



Beginning

Programming

مُتَرْجِم

ترجمة

أحمد موسى

Adrian Kingsley-Hughes



Beginning Programming

How to buy This Book

الموقع الرسمي لـ Adrian

Adrain Web site Here

<http://kingsley-hughes.com>

Book Web Page

<http://www.kingsley-hughes.com/books/begprog.php>

you will find Links in The above Page

ستجد الروابط فى الصفحة بالأعلى

بسم الله وبالله التوفيق

كتاب

Wrox.Beginning.Programming.
Apr.2005

لمؤلفيه

Adrian and Kathie Kingsley-Hughes

Beginning Programming

Published by
Wiley Publishing, Inc.
10475 Crosspoint Boulevard
Indianapolis, IN 46256
www.wiley.com

Copyright © 2005 by Wiley Publishing, Inc., Indianapolis, Indiana

Published simultaneously in Canada

ISBN-13: 978-0-7645-8406-0

ISBN-10: 0-7645-8406-5

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

1MA/SU/QT/QV/IN

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except as permitted under Sections 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923, (978) 750-8400, fax (978) 646-8600. Requests to the Publisher for permission should be addressed to the Legal Department, Wiley Publishing, Inc., 10475 Crosspoint Blvd., Indianapolis, IN 46256, (317) 572-3447, fax (317) 572-4355, e-mail: brandreview@wiley.com.

LIMIT OF LIABILITY/DISCLAIMER OF WARRANTY: THE PUBLISHER AND THE AUTHOR MAKE NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE ACCURACY OR COMPLETENESS OF THE CONTENTS OF THIS WORK AND SPECIFICALLY DISCLAIM ALL WARRANTIES, INCLUDING WITHOUT LIMITATION WARRANTIES OF FITNESS FOR A PARTICULAR PURPOSE. NO WARRANTY MAY BE CREATED OR EXTENDED BY SALES OR PROMOTIONAL MATERIALS. THE ADVICE AND STRATEGIES CONTAINED HEREIN MAY NOT BE SUITABLE FOR EVERY SITUATION. THIS WORK IS SOLD WITH THE UNDERSTANDING THAT THE PUBLISHER IS NOT ENGAGED IN RENDERING LEGAL, ACCOUNTING, OR OTHER PROFESSIONAL SERVICES. IF PROFESSIONAL ASSISTANCE IS REQUIRED, THE SERVICES OF A COMPETENT PROFESSIONAL PERSON SHOULD BE SOUGHT. NEITHER THE PUBLISHER NOR THE AUTHOR SHALL BE LIABLE FOR DAMAGES ARISING HEREFROM. THE FACT THAT AN ORGANIZATION OR WEBSITE IS REFERRED TO IN THIS WORK AS A CITATION AND/OR A POTENTIAL SOURCE OF FURTHER INFORMATION DOES NOT MEAN THAT THE AUTHOR OR THE PUBLISHER ENDORSES THE INFORMATION THE ORGANIZATION OR WEBSITE MAY PROVIDE OR RECOMMENDATIONS IT MAY MAKE. FURTHER, READERS SHOULD BE AWARE THAT INTERNET WEBSITES LISTED IN THIS WORK MAY HAVE CHANGED OR DISAPPEARED BETWEEN WHEN THIS WORK WAS WRITTEN AND WHEN IT IS READ.

For general information on our other products and services or to obtain technical support, please contact our Customer Care Department within the U.S. at (800) 762-2974, outside the U.S. at (317) 572-3993 or fax (317) 572-4002.

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print may not be available in electronic books.

Trademarks: Wiley, the Wiley Publishing logo, Wrox, the Wrox logo, Programmer to Programmer, and related trade dress are trademarks or registered trademarks of John Wiley & Sons, Inc. and/or its affiliates, in the United States and other countries, and may not be used without written permission. All other trademarks are the property of their respective owners. Wiley Publishing, Inc., is not associated with any product or vendor mentioned in this book.

كل حركات التطور والإزدهار بدأت بالترجمة
ويذكر أن أحد الأمراء أنه كان يعطى للمترجم وزن الكتاب المترجم ذهباً
فليته هذا كان عصره !!

About the Authors

عن المؤلف

ادريان و كاثى كنجسلى - هوجس . لقد عملا فى ميدان الحاسب والبرمجة لأكثر من عشر سنوات , أول إستخدام لمهاراتهم كان فى حل مشاكل العالم الواقعى للمنشات الصناعية وقبل أن ينتقلوا إلى تعليم الآخرين فقد أصبحوا مبرمجين . فى السنوات الأخيرة كانت لديهم الفرصة لتعليم البرمجة للآلاف للمبرمجين الطموحين من خلال الدورات التدريبية عبر الإنترنت . هناك أكتشفوا متعة تدريس البرمجة للبشر الذين اتوا من جميع أنحاء مجالات الحياة وليس لديهم خبرة فى البرمجة أو خلفية سابقة .

أدريان و كاثى يعيشان شمال ويلز الجميلة , المملكة المتحدة , ويقسمان وقتهما بين عالم التعليم الإلكتروني والأنشطة الترفيهية فى الهواء الطلق.

عن المترجم

أحمد موسى , ينتسب إلى عالم البرمجة من قريب . والله الموفق .



أحمد موسى محمود
Microsoft Certified Professional
Web Developer

المهنة	مدرّب / مبرمج / مترجم في عالم الحاسب الآلي
العنوان	جمهورية مصر العربية
البريد الإلكتروني	ahmed_moosa83@yahoo.com
المدونة باللغة الإنجليزية	www.weblogs.asp.net/ahmedmoosa
المدونة باللغة العربية	www.ahmedmoosa.wordpress.com
لمزيد من التفاصيل يرجى المراسلة بذلك	

يرجى سرعة الاتصال عند الحالات التالية :

- العثور على خلل في الترجمة (يرجى ذكر رقم الصفحة والتعليق في رسالة)
- تريد المشاركة في ترجمات كتب تالية (وهذا يتم التجهيز له إن شاء الله)
- تواجد فرصة عمل مميزة (😊)

عن الترجمة داخل الكتاب

أولاً وقبل كل شيء . إن الكمال لله وحده .

ثانياً:

كان الهدف من طريقة سير الترجمة ان يخرج الكتاب طبق الأصل ولكن بلغة عربية . لذا تجد الصور والأكواد قد صورت وتم قصها من الكتاب الأصلي . حتى الغلاف , تم كتابة كلمة ترجمة وتركه كما هو كما تراه بالسابق . وأسماء المنتجات من البرامج نادراً ما تجدها مترجمة , بل تترك كما هي بحروف إنجليزية . كذلك الأسماء للأشخاص والشركات والمواقع والمننديات .

كما يوجد كلام بين أقواس فيه من كلام المؤلف وفيه من كلام المترجم ربما لتوضيح غامض . وللتفريق بينهما خصص المترجم كلامه وليس بكثير بين هذا الشكل من الأقواس **فقط**

[.....]

فكل ما تجده داخل هذا الشكل من الأقواس هو من كلام المترجم وكما ذكرنا ليس بكثير .

ثالثاً :

ترجمة الحاسب أمر محير , لذا فهناك مصطلحات لا تفهم إلا بالإنجليزية أو بالعربية المفصلة تفصيلاً مطولاً . لذلك لو أطلق المترجم العنان لأصبح كتابين داخل كتاب , ولكن تم كتابة مصطلحات شائعة الاستخدام والمألوف والمألّف من خبرة المترجم في عالم البرمجة أى ربما لا تجده في مكان آخر . فربما تجد انه تم كتابة المصطلح وترجمته وكذلك المصطلح نفسه باللغة الإنجليزية بجوار بعضهم البعض .

Credits

Acquisitions Editor

Katie Mohr

Senior Development Editor

Kevin Kent

Development Editor

Howard A. Jones

Production Editor

Felicia Robinson

Technical Editor

Wiley-Dreamtech India Pvt Ltd

Copy Editor

Foxxe Editorial Services

Editorial Manager

Mary Beth Wakefield

Vice President & Executive Group Publisher

Richard Swadley

Vice President and Publisher

Joseph B. Wikert

Project Coordinator

Erin Smith

Graphics and Production Specialists

April Farling

Lauren Goddard

Denny Hager

Clint Lahnen

Lynsey Osborn

Rashell Smith

Ron Terry

Quality Control Technicians

Amanda Briggs

John Greenough

Joe Niesen

Carl William Pierce

Brian H. Walls

Proofreading and Indexing

TECHBOOKS Production Services

Contents

Introduction	xvii
Chapter 1: What Is Programming?	1
The History of Programming	1
What Is Programming?	4
Why So Many Programming Languages?	4
Different Code, Same Results	6
BASIC	7
Atari BASIC	7
C	7
C++	7
COBOL	8
FORTRAN	8
Java	8
JavaScript	9
Mathematica	9
Pascal	9
Perl	9
Python	9
QBASIC	9
Smalltalk	9
Visual Basic	9
VRML	10
Programs Needed to Create Programs	11
Development Environment	11
Compilers	11
Summary	12
Chapter 2: Why Learn to Program?	13
Why Program?	13
The "Career" Programmer	14
Problem Solving	16
How Big Is the Project?	17
Is the Project Time Critical?	17
Work or Hobby?	17

Because You Want to Contribute	18
Just for Fun	18
Fame	18
What to Learn?	19
Programming Avenues	19
Traditional Programming	19
Web Programming	21
Programming Myths and Facts	23
Myth 1 — Programming Tools Cost a Lot of Money	23
Myth 2 — You Need a Degree in Computer Science to be a Programmer	24
Myth 3 — Learning to Program Takes Years	24
Myth 4 — Programming Is Just for Young People	24
Myth 5 — I'll Need a Top-of-the-Range Super-Duper Computer to Program On	24
Myth 6 — Programming Is Addictive!	24
Myth 7 — Programming Languages Change All the Time	25
Myth 8 — Once You've Learned One Programming Language, Learning Others Is Easier	25
Summary	25
Chapter 3: How Computers “Read” Code	27
Reading Code	27
Top Down	28
Breaking Up Code	31
The Statement	35
Functions/Procedures	36
The Sentences and Paragraphs of Programming	37
Lines of Code	37
Paragraphs of Code	38
Data Storage	39
Data	40
Summary	42
Chapter 4: From Concepts to Code — The Language of Code	43
Binary	44
Interpreting Binary	44
Large Numbers	46
Bit Grouping	47
Bit	47
Nybble	47
Byte	48
Halfword	48
Word	49
Doubleword	50

Binary Math	50
Using Windows Calculator	51
Why Binary?	54
Hexadecimal	54
Interpreting Hexadecimal	55
Hexadecimal and Windows Calculator	57
Representing Characters	57
Operators	65
Arithmetic Operators	65
Assignment Operators	66
Comparison Operators	66
Logical Operators	67
String Operators	68
Summary	69
 Chapter 5: The Tools for Programming	 71
 Make Your Workspace Your Own	 71
The Keyboard	71
Workspace	73
Desk	73
Monitor	73
Fonts	73
Choose Your Language	75
Learning to Program	76
School/College	77
Tips	77
Work-Based Training	77
Tips	78
Hobby Programmer	78
Tip	79
The Languages	79
How I Will Teach You to Program	79
Why Not Buy a Book Covering a Specific Language?	80
The Tools	80
General Tools and Utilities	80
Text Editor	80
Utilities	83
Programming Tools	87
Java	89
C++	91
Scripting Languages	94
Summary	95

Chapter 6: Simple Coding	97
Commenting Code	97
VBScript Comments	98
Things to Watch For	100
Quick Exercise	100
JavaScript Comments	101
Things to Watch For	102
Quick Exercise	102
C++ Comments	103
Things to Watch For	103
Quick Exercise	104
Variables	104
Variables in Action	107
Variable Run Through	107
Naming Variables	118
Naming of Variables	121
Quick Exercise	124
Strings	124
What Are Strings?	124
String Manipulation	126
Processing Inputs	130
Variable Manipulation — Simple Math	132
Summary	136
Chapter 7: The Structure of Coding	137
The Purpose of Structure	137
Benefits	138
Examining Structure	138
Quick Introduction to C++	138
Examine Source Code	138
Compiling C++	141
Functions	149
More Functions	152
Quick Exercise	156
Conditionals	157
Programming Decisions	157
Conditions	158
More on Conditionals	163
Quick Exercise	171
Loops	173
For Loops	173
Infinite Loops	175
While Loops	175

Do While loop	176
Quick Exercise	177
Arrays	179
Two-Dimensional Array	180
Multidimensional Array	181
Quick Exercise	182
Summary	183
Chapter 8: Problem Solving	185
The Basics of Problem Solving	186
Be Clear about the Requirements	186
Research	191
Breaking the Problem up into Smaller Problems	194
What Are the Issues That the Application Needs to Deal With?	195
Moving on to the Coding Phase	196
Improving the Code	204
Summary	212
Chapter 9: Debugging	213
To Err Is Human	213
Errors, Errors, Errors!	214
Different Kinds of Error	214
Compiler Error	214
Runtime Errors	228
Logic Errors	232
Spotting Errors	236
Read Each Line After You Press Enter	236
Check the Preceding Statements	237
Keep the Layout Clear	237
Comments, Comments, Comments!	237
Remove Ambiguity in Code	238
Semicolons	238
Test the Code	239
Keep Track of Variables	239
Summary	245
Chapter 10: Interface	247
What Is an Interface?	247
The Importance of an Interface	250
What Is an Interface?	250
Does All Software Have an Interface?	251

Examining the Interface	251
Text-Based Interface	251
Program Overview	254
Proper Prompting for Input	256
Annotating Output	259
Confirming Exit	260
Adding Simple Help	261
Confirmations	267
Moving Away from the Text-Based Interface	268
Buttons	268
Menus	270
Check Box	271
Radio Buttons	272
Single-Line Text Box	273
Multiline Text Box	274
Drop-Down Menu	275
Putting It All Together	275
Simple Applications	276
More Complicated Applications	280
Summary	283
 Chapter 11: Putting It All Together	 285
Planning a Programming Project	285
Without Planning	285
More Code, Less Features	285
More Bugs	286
Project Takes Longer	286
Missing Features	286
Planning	286
The Idea	286
Documenting the Idea	287
Maturing Time	289
The Requirements	290
Requirements	290
Programming Stage	293
Programming the Basics	293
Testing	293
Commenting Code	294
Testing	296
The Route to Better Testing	297
But What About . . .	298
Questions for Those Previewing Software	298

Additional Features	299
Tweak the Code	300
Final Testing	300
Summary	301
 Chapter 12: Interacting with Files	 303
The Principles of Saving Data	303
The File Life Cycle	304
Working with Files	306
The Tools	306
Getting Started	306
Creating a File with VBScript	307
Basics	307
Creating a Folder	310
Creating Multiple Files	311
Making Use of Conditionals	312
Making Use of Variables	313
Adding Flexibility — Prompt for File and Folder Names	314
Check for Duplicate Files	317
Editing an Existing File	317
In Action	319
Appending a File	319
Open File for Reading	320
ReadAll, ReadLine, and Read Methods	321
ReadAll	321
ReadLine	322
Read	323
Deleting Files and Folders	325
Delete Files	325
Delete Folders	325
Summary	326
 Chapter 13: The Windows Registry	 327
The Windows Registry	327
What Is the Windows Registry?	327
Definition	328
The Layout of the Windows Registry	329
Regedit and Regedit32	329
Backing Up the Registry	330
Windows XP	330
Restoring the Registry	339

Working with the Registry	341
Finding a Subtree, Key, Subkey, or Value	341
Adding a New subkey	343
Adding a New Value	344
Changing an Existing Value	346
Renaming an Existing Subkey or a Value	347
Deleting an Existing Subkey or a Value	348
Manipulating the Windows Registry Using Programming	349
VBScript Registry Editing	349
JScript Registry Editing	354
Possible Uses for the Windows Registry	355
In Closing	356
Summary	357
 Chapter 14: Organizing, Planning, and Version Control	 359
Organize, Organize, Organize!	359
Organize Yourself	359
Stages of Planning	360
Plan Your Time	360
Organize Your Workspace	360
The Main Event — Organize Your PC	362
Create a Workspace	362
Folders, Folders, Folders	364
Group by Language	364
Group by Project	365
Folder Contents Note	366
Filename Control	368
More Version Control Tips	370
Add Version Information to the Tombstone Comment Block	370
Using Windows Search	371
Add Summary Information to the File	373
Version Control — Looking Beyond Release	376
Software Version Control	377
Summary	377
 Chapter 15: Compiling Code and Alternatives to Compiling	 379
Compiling Code	379
Are All Compilers the Same?	381
Error Handling	387
What about Different Languages?	391

Benefits of Compiling	394
Protection of Intellectual Property	395
Speed	395
Increased Functionality	396
Security	396
Debugging	397
Alternatives to Compiled Code	397
Make Code Hard to Follow	397
Obscure Variable Names	397
Whitespace	398
Script Encoding	398
Summary	404
 Chapter 16: Distributing Your Project	 405
Types of Distribution	405
Physical Distribution	405
Floppy Disk	406
CDs	409
DVD	411
Burning Discs	412
CD/DVD Burner	412
Burning Software	414
CD/DVDs	419
Labeling	420
Packaging	420
Virtual Distribution	421
Pros of Virtual Distribution	421
Cons of Virtual Distribution	422
Considerations for Virtual Distribution	422
Full-Time Job versus Hobby	423
Supported versus Unsupported	424
Summary	425
 Appendix A: Glossary	 427
 Appendix B: Web Resources	 433
Programming Tools	433
Java Tools	435
Java Sites	436

C++ Tools	437
C++ Sites	438
BASIC Tools	439
BASIC Sites	440
Web Scripting Languages	441
CD Burning	441
Compression Tools	442
Miscellaneous Tools	444
Miscellaneous Sites	446
Index	447

Introduction

المقدمة

فى وجود العديد من الحاسبات فى متداول الحياة العامة , فى العمل , فى المنزل , وفى المكتبات والمدارس , فكان مما لا مفر منه أن الأشخاص الذين كانوا سعداء ان يكونوا مجرد "مستخدمين" فسيودون ان يتخذوا الخطوة التالية وهى تعلم كيف يقوم الحاسب بتقديم اعمال لهم بطريقة أخرى . فأدركوا سريعاً أنهم بحاجة ان يتعلموا فن كيف تصبح مبرمجاً . واول خطوة قاموا بعملها هى اقتناء مثل هذا الكتاب .

المشكلة فى معظم كتب البرمجة هى انهم يفترضون أن القارئ أحد إثنين :

- أن يكون متحمس جداً لدرجة انه قد قرأ العديد من المواد العلمية فى البرمجة قبل أن يأتى للكتاب .
- أن لديه خلفية فى الحاسبات وقد تعرض للبرمجة من قبل .

نحن لا نفترض أى شئ - كل ما نفعله هو اننا نبحث فى البرمجة من البداية .

Who This Book Is For

لمن هذا الكتاب

معظم كتب برمجة الحاسب تهدف إلى الأشخاص الذين أخذوا بالفعل طريق مبرمجين والذين يريدون ان يصلقوا مهاراتهم فى مهنة معينة . هؤلاء الأشخاص يعرفون الاختلاف بين الدالة وبين نظم الصفوف أو المصفوفات . ويعرفون أيضاً أن المبرمجون الجيدون يقومون بتعريف متغيرات وترك تعليقات وغيرها من الممارسات البرمجية التى سنعرفها فيما بعد .

هذا الكتاب مختلف - هذا الكتاب موجه للذين يريدون تعلم كيف يكونوا مبرمجين ولكن ليس لديهم خلفية قد كشفت لهم البرمجة , أو للمبرمجين , وللمعلمين , للتلاميذ , للممرضون , للمحامون , لسائقوا عربات اللورى , للطيارون , أو الأشخاص الذين يرون ان جعل الحاسب يقوم بأعمال لهم هى ميزة يريدون ان يقتنصوها ويحصلوا عليها .

من قبل , كانت كتب البرمجة تركز على الأشخاص الذين يريدون أن يحصلوا على مهنة مبرمجين , هذا الكتاب لهم ولغيرهم !

وأيضاً إذا إقتنيت هذا الكتاب لتصبح مبرمجاً محترفاً فلا مانع . هذا الكتاب سيعطيك التأسيس الجيد الذى تحتاجه لتضييق مجال الدراسة وتركز على مهمة تخصصك فى لغات البرمجة بشكل خاص أو المهام الذى تريدها . فى هذه الصفحات لن تجد " كيف تصبح خبيراً فى البرمجة فى خمس دقائق " أو " كيف تكتب تطبيقات تهزم العالم " . ولكن ما تجده هو معلومات ثابتة تمكنك من اخذ مهارتك فى الإتجاه الذى تحتاجه .

What This Book Covers

ماذا يغطي هذا الكتاب

الغرض من هذا الكتاب هو السماح للمهتمين بالبرمجة ان يجمعوا مهاراتهم و الخبرات التى يحتاجونها لتحقيق أهدافهم . التركيز الأساسى لهذا الكتاب على المهارات التى هى من صميم تكوين المبرمج . فيعرض لك الكتاب نظرية البرمجة ويلقى نظرة على هذه النظرية فى الفعل فى شكل كود حقيقى . فاستخدم لغات برمجة متنوعة خلال هذا الكتاب لأساعدك على فهم وتوضيح هذه المفاهيم .

اللغات التى سنبحث فيها تشمل على :

- ☐ C++
- ☐ Java
- ☐ VBScript
- ☐ JavaScript

يعرض هذا الكتاب امثلة متعددة لهذه اللغات فى الفعل إضافة إلى تزويدك بالفرصة لإستخدام أدوات برمجة وبرامج ترجمة الكود المتنوعة .

How This Book Is Structured

كيف تم إعداد هذا الكتاب

تم إعداد هذا الكتاب ليرشدك من خلال الخطوات التى تحتاجها لتصبح أكثر احترافاً فى المهارات التى تجعلك مبرمج حاسب محترف . وأوصيك ان تبدأ من الفصل الأول والمضى قدماً إلى نهاية الكتاب . فصلاً بفصل .

- **الفصل الأول :** "ما هى البرمجة ؟ " هذا الفصل يتكلم عن ما هية البرمجة , وماذا يعنى أن تصبح مبرمجاً .
- **الفصل الثانى :** "لماذا تتعلم البرمجة ؟ " هذا الفصل يتكلم عن الأسباب التى لاحصر لها و التى تجعل الكثيرون يريدون ان يتعلموا البرمجة ويوضح أيضاً الإتجاهات المختلفة التى يمكن ان يأخذوها ليصبحوا مبرمجين .
- **الفصل الثالث :** "كيف تقرأ الحاسبات الكود" هذا الفصل يتكلم عن كيفية تخزين او معالجة الحاسبات للكود . ويفترض ان يعطى هذا للقارئ رؤية على ما يحدث وراء الكواليس فى الحاسب .
- **الفصل الرابع :** "من مفاهيم الكود إلى لغة الكود" هذا الفصل يتكلم عن أساسيات لغات الحاسب و كيف تمثل النصوص والارقام فى العالم الرقمى . هنا سنلقى نظرة على كيف ان أشكال أنظمة الأرقام الثنائية والسادسى عشر هى جزء أساسى من الكود وكيف أن ASCII تسمح للحر وف الأبجدية أن تمثل كنظام أرقام ثنائى .

- **الفصل الخامس : "أدوات البرمجة "** هذا الفصل يتكلم عن الأدوات التى تحتاجها لتعمل مع مصدر الكود وتصبح مبرمجاً فعالاً . ستتعرف على محررات النصوص , المفسرات [محولات الكود] , وبرامج مرفقة اخرى تجعل معالجة الكود بها المزيد من السهولة وأقل ضغطاً .
- **الفصل السادس: "الكود البسيط "** فى هذا الفصل سنمر خلال عملية كتابة الكود الذى يعمل بالفعل وينفذ مهمة . وسنتعرض لعدد من المفاهيم الأساسية فى البرمجة التى تعتمد عليها لغات البرمجة .
- **الفصل السابع : "تركيب الكود "** هذا الفصل يتكلم فى كيفية أخذ الكود من مجرد أسطر مكتوبة لتعطي تركيبة معينة تمكّنك من تنفيذ مهام باستخدام مدخلات متنوعة ويسمح لجمل الكود المعينة أن تعمل بناءً على إختبار شرطاً ما .
- **الفصل الثامن : "حل المشكلات "** كتابة الكود كلها يعتبر هدفها حل المشكلات , وفى هذا الفصل سترى أفضل ما يمكن فعله مع عملية حل المشكلات . هذا الفصل يتكلم عن كيفية تقطيع كتل الكود فى خطوات متسلسلة صغيرة وسهلة إدارياً .
- **الفصل التاسع : " التصحيح "** أى وقت تكتب فيه كود فهناك إحتمال أن يحتوى على مشكلة . هذا الفصل يتكلم عن الأخطاء المتنوعة التى يمكن أن ترحف إلى الكود وتوقعه.
- **الفصل العاشر : "الواجهة "** بينما نحن فى إنشاء الكود ينبغى أن نضع أعيننا على كيف يبدو مظهر البرنامج للمستخدم النهائى . هذا الفصل يزودك بالتلميحات , والحيل , ونصائح مفيدة لجعل تطبيقاتك سهلة الإستخدام .
- **الفصل الحادى عشر : "وضع الكل مع بعضهم "** إلى هذا الحد تكلمنا عن المراحل الفردية فى البرمجة . فى هذا الفصل , سنحضرهم جميعاً معاً , لتمر بمشروع برمجة من المفاهيم الاساسية إلى نهاية المنتج . و هذا يعطيك فرصة جيدة لتجمع كل مهاراتك مع بعضها البعض لتختبر عملية البرمجة من البداية للنهاية .
- **الفصل الثانى عشر : "التفاعل مع الملفات "** هذا الفصل يعطيك الخبرة فى العمل مع ملفات النظام ويعرض لك كيفية إنشاء , تعديل , وحذف الملفات والمجلدات .
- **الفصل الثالث عشر: "الريجسترى The Windows Registry "** هى مساحة تخزين أساسية لأعدادات التطبيقات . وهذا الفصل يزودك بالخبرة بتصفحه وكيفية كتابة كود يدخل , ويقرأ , ويعدل فى برنامج Registry .
- **الفصل الرابع عشر: "التحكم فى تخطيط وتنظيم الإصدار "**. تعقب مشروعك والكود الذى بداخله أمر هام إذا لم تجد عملية كتابة الكود مجهده وعمل شاق . هذا الفصل يركز على كيفية التحكم فى مصدر الكود وكيفية معالجة الملفات التى تنشأها .
- **الفصل الخامس عشر: "تفسير الكود والبدائل للتفسير "** هذا الفصل يعرض لك عملية تفسير الكود ويبحث فى المنافع التى تاتى من وراء تفسير الكود إلى المبرمج .
- **الفصل السادس عشر : " توزيع ونشر المشروع "** فى هذا , الفصل الأخير , نتكلم عن الطرق التى يمكنك ان تنشر بها الكود أو التطبيقات إلى المستخدم النهائى . سترى انماط متنوعة يمكنك أن تستخدمها لتذهب بالكود الخاص بك إلى كل من يريد أن يستخدمه .

What You Need to Use This Book

ماذا تحتاج لتستخدم هذا الكتاب

كل الأدوات والمواد العلمية التي تحتاجها للعمل مع هذا الكتاب متاحة بسهولة على الإنترنت او بالفعل مثبتة على حاسبك .

مبدئياً ستستخدم محرر النصوص لإنشاء الكود . برنامج Windows Notepad هو برنامج مثالي لهذا . ولكن إذا أردت برنامجاً أكثر قوة يمكنك أن تستخدم بدائل متنوعة , البعض مجاني , والبعض تجاري .

الأداة التي سأستخدمها تسمى UltraEdit , وهي برنامج تجاري قوي يمكن تحميله من الموقع www.ultraedit.com . وهي اداة تعتمد على تحرير النصوص وهو ما يفعله المبرمجون . ليست مجانية ولكن تقوم بالعمل بشكل جيد . إذا لم تكن في حاجة لإستخدامها . فانت في حل من امرك فيمكنك ان تستخدم بدائل أخرى .

تحتاج أيضاً ان تملك مفسر Compiler . المفسر الرئيسي الذي سأستخدمه مفسر مجاني وهو مفسر Borland C++ . ويمكنك ان تقوم بتحميل هذا المفسر من هنا :

www.borland.com/products/downloads/download_cbuilder.html

[المفسر Compiler هو برنامج صغير يقوم بتحويل الكود إلى لغة الآلة ليفهمه الحاسب]

الأدوات الأخرى التي ستحتاجها سنعرضها على حسب الحاجة .

Conventions

الإتفاقيات

لأساعدك لتحصل على أكبر قدر من الفهم مع النص وتعقب ما يحدث , لقد إستخدمت بعض الإتفاقيات خلال هذا الكتاب .

Boxes like this one hold important, not-to-be forgotten information that is directly relevant to the surrounding text.

المربعات التي تشبه هذه هامة يجب حفظها

النصائح ، التلميحات ، و الحيل ، والتعليقات الجانبية للمناقشة الحالية كتبت ووضعت في خط مائل مثل هذا

للتنسيقات داخل النص .

- لقد جعلت الكلمات الهامة بشكل مائل .
- اظهر لك إختصارات لوحة المفاتيح مثل هذا : Ctrl + A
- اظهر لك إسماء الملفات , وعناوين الإنترنت و الكود خلال نص مثل هذا
persistence.properties
- لقد عرضت الكود في بطريقتين مختلفتين :

In code examples we highlight new and important code with a gray background.

The gray highlighting is not used for code that's less important in the present context or that has been shown before.

Source Code

مصدر الكود

كما عملت داخل الكتاب مع الامثلة , فمتاح لك إما ان تكتب كل الكود بيدك أو تستخدم ملفات مصدر الكود الذى بصطحبة الكتاب . فستجد كل ملفات مصدر الكود المستخدم فى هذا الكتاب متاح للتحميل فى الموقع www.wrox.com . مجرد أن تكون فى الموقع , ببساطة حدد المكان بعنوان الكتاب (إما بإستخدام صندوق البحث أو بإستخدام قائمة العناوين) وإضغط على تحميل الكود Download Code فى صفحة تفاصيل الكتاب لتحصل على كل ملفات مصدر الكود لهذا الكتاب .

بسبب أن الكثير من الكتب لها نفس العنوان , ربما تجد من السهل أن تبحث بواسطة رقم ISBN الرقم القياسى الدولى للكتاب : والرقم لهذا الكتاب هو :

ISBN is 0-7645-8406-5.

مجرد ان تقوم بتحميل مصدر الكود , قم بفك الضغط بإداة الضغط المفضلة لديك . بدلاً من ذلك يمكن أن تذهب إلى صفحة الكود الرئيسية فى موقع Wrox تجدها هنا :

www.wrox.com/dynamic/books/download.aspx

تجد هناك الأكواد المتاحة لهذا الكتاب وكل كتب شركة Wrox .

Errata

الأخطاء

لقد بذلنا كل جهد للتأكد من أن هذا الكتاب ليس فيه خطأ فى النص أو فى الكود . مع ذلك , لا أحد كامل , والأخطاء تحدث , فإذا وجدت خطأ فى أحد كتبنا , مثل خطأ إملائي أو خطأ فى جزء من الكود , سنكون شاكرين لك لإعطائنا خلفية عن ذلك , بإرسائل الإخطاء ستحفظ الكثير من الوقت والحيرة على قارئ آخر وفى نفس الوقت ستساعدنا بتقديم معلومات أكثر جودة .

لتجد صفحات الأخطاء لهذا الكتاب أدخل للموقع www.wrox.com . وحدد الكتاب كما ذكرنا من قبل , ستجد هناك رابط لصفحة الأخطاء . ويمكنك أن تصل إلينا

www.wrox.com/contact/techsupport

p2p.wrox.com

For author and peer discussion, join the P2P forums at p2p.wrox.com. The forums are a Web-based system for you to post messages relating to Wrox books and related technologies and interact with other readers and technology users. The forums offer a subscription feature to e-mail you topics of interest of your choosing when new posts are made to the forums. Wrox authors, editors, other industry experts, and your fellow readers are present on these forums.

At <http://p2p.wrox.com> you will find a number of different forums that will help you not only as you read this book, but also as you develop your own applications. To join the forums, just follow these steps:

1. Go to p2p.wrox.com and click the Register link.
2. Read the terms of use and click Agree.
3. Complete the required information to join as well as any optional information you wish to provide and click Submit.
4. You will receive an e-mail with information describing how to verify your account and complete the joining process.

You can read messages in the forums without joining P2P; but in order to post your own messages, you must join.

Once you join, you can post new messages and respond to messages that other users post. You can read messages at any time on the Web. If you would like to have new messages from a particular forum e-mailed to you, click the Subscribe to this Forum icon by the forum name in the forum listing.

For more information about how to use the Wrox P2P, be sure to read the P2P FAQs for answers to questions about how the forum software works as well as many common questions specific to P2P and Wrox books. To read the FAQs, click the FAQ link on any P2P page.

[الفقرة السابقة بإختصار هي أن الشركة وفرت منتدى لتلقى وعرض الأسئلة والأجوبة للعملاء , فتشرح لك عملية التسجيل وبعدها يمكنك ان تقرأ وتعرض مشكلاتك والرد عليها , والسلام]

1

What Is Programming?

ماهى البرمجة ؟

إذا كنت ستقتنى كتاب يتحدث عن البرمجة فلا بد من أن يكون لديك فكرة جيدة عن ماهية البرمجة . وعلى كلاً , لماذا ستريد أن تتعلم شيئاً وانت لا تعرف ما هيته ؟ , ومع ذلك العديد من الطلاب الجدد في عالم البرمجة الرائع , أو حتى الممارسين لهذا الفن ممن تعلموا القليل من البرمجة للحصول على وظيفة عادية في البرمجة ومن ثم يبدؤون ببناء خبراتهم البرمجية من هناك . فربما يكون من المنفعة على عجلة أن تعرف تاريخ البرمجة . وما هي البرمجة ؟ وأين هي الآن ؟

The History of Programming تاريخ البرمجة

يمتد تاريخ البرمجة لسنوات عديدة لأكثر مما يتخيل الناس . معظم الناس يعتقدون ان البرمجة هي إختراع من إختراعات اواخر القرن العشرين . فى الحقيقة , تاريخ البرمجة الحديثة ولغات البرمجة يعود منذ ما يقرب من 60 عاماً إلى منتصف سنة 1940 .

ومع ذلك قبل أن نستأنف القصة فى عام 1940 , نحتاج أن نعود بالزمن إلى الوراء . فى طريق العودة إلى 1822 و "تشارلز باباج " . بينما كان يدرس فى جامعة كامبردج فى بريطانيا . وجاء " تشارلز باباج " بناء على إدراك أن العديد من اجهزة حساب الوقت . مثال الجداول الفلكية . ومخططات المد والجزر , والخرائط الملاحية جميعها تحتوى على أخطاء حرجة وربما سهو . وهذه الأخطاء تتسبب فى ضياع العديد من السفن فى البحر جنباً إلى جنب لحياة الناس والبضائع . ولأعتبره أن أخطاء البشر هى المصدر لعدم الدقة فى هذا الأمر . فكانت فكرته اخذ التخمين فى إنشاء وإصلاح مثل هذه الخرائط والمخططات بواسطة صنع آلات بخارية تعمل بالطاقة . أو شئ من هذا القبيل . وأطلق عليها محرك الفرق . ولعمل ذلك كان لابد من إشغال وقته لبقية حياته . وكما انه أعلم الحكومة البريطانية ليحصل على الدعم المالى فطلب منحة لإبحاث الكمبيوتر . وبعد عشر سنوات من العمل على هذا المحرك . أدرك فى نهاية المطاف انها آلة ذات غرض واحد تمكنه من تنفيذ عملية واحدة . وكما انه أدرك ان هناك قيود كثيرة . فتخلى عنها لصالح إنشاء المحركات التحليلية المتنوعة . إحتوى هذا المحرك على المكونات للحاسبات الحديثة . وقاده ذلك إلى أن لُقّب " بأبو الحاسب [الكمبيوتر] " . ولم تلقى هذه المحركات التحليلية الإقبال الواسع بسبب أنه أصيب

بالإصابة التي تصيب مبرمجي الحاسب وعلماء الحاسب عبر العصور وهى عدم القدرة على توضيح أوراقه ونقل أفكاره .

إستمر العمل على المحركات التحليلية إلى سنة 1942 . عندما تعبت الحكومة البريطانية من عدم إحرار تقدم والحصول على نتائج . تخلت عن الفكرة وسحبت التمويل . ومع ذلك إستمر هو بالعمل عليه حتى 1847 بينما عاد إلى محرك الاختلاف وبين ذلك وبين 1849 أنهى 21 رسم تفصيلي لتركيبة الإصدار الثانى من المحرك . ولم يحصل حقيقةً على إنهاء جهازه . فى الحقيقة كانت حتى بعد مماته فى 1871 حيث قام " إبنه " و " هنري بريفوست " ببناء العديد من النسخ من الوحدات الرياضية لمحرك الاختلاف . وإرسالها للعديد من المعاهد حول العالم تشمل جامعة هارفارد للتأكد من صحتها .

وأستمر التقدم خلال الثمانينات . وفى 1854 وصف " تشارلز بول " النظام المنطقى الرمضى . والذي يحمل اسمه (Boolean logic) الذى ظل إستخدامه إلى يومنا هذا . (والذى نغطيه فيما بعد فى هذا الكتاب) . أخذ هذا النظام مصطلحات منطقية مثال اكبر من و أصغر من ويساوى ولا يساوى والآنشأ نظام رمضى ليعرض به هذه المصطلحات .

وقال إن الحاجة هى أم الاختراع . ووصلت الحاجة إلى ذروتها فى 1880 عندما أراد الكونجرس الأمريكى إضافة العديد من الأسئلة حول التعداد السكانى . وزيادة سكان الولايات المتحدة تعنى أن معالجة البيانات أخذت وقتاً أطول وأطول . وقد تم تقدير أن بيانات التعداد لـ 1890 لن تعالج قبل تعداد 1900 ما لم يتم عمل شئ لتحسين وتسريع هذه العملية .

لذلك , كانت المسابقة لحشد الإهتمام بمجال الحاسب وتقديم معدات معالجة بيانات جيدة للحكومة وفاز بهذه المسابقة "هيرمان هوليريث" . وبعد إثبات نجاح هذه التقنية . ذهب لمعالجة معلومات التعداد فى بلاد أخرى . وفى وقت لاحق أسس شركة وسماها " شركة هوليريث تابولاتينج " . هذه الشركة أصبحت فى نهاية المطاف واحدة من ثلاثة قد إنضموا معاً 1914 وتم إنشاء ما يسمى CTR (the Calculating Tabulating Recording Company) [أوشركة تسجيل الحسابات المجدولة] . هذا الإسم ربما لم تسمع به ولكن فى السنوات العشر الأخيرة تم إعادة تسمية الشركة بإسم (International Business Machines,) أو IBM [والترجمة آالات الأعمال الدولية] وهذا الإسم لاشك انك قد سمعت به .

إمتلك " هوليريث " إسم آخر بجوار إسمه إسم آله التى كانت الأولى فى الظهور لتغطى غلاف المجالات .

إستمر التقدم ولكن بدا بطئ قليلاً وبمنتصف عام 1920 كانت ادوات عمليات الحاسب الرقمية قليلا ما تستخدم فى الأعمال التجارية أو المصانع أو الهندسيات . وفى الحقيقة , كانت الأداة الأكثر إستخداماً هى Humble analog slide rule . وبدأت الأشياء فى التغير فى 1925 , ومع ذلك . عندما وبينما كانوا فى MIT [ربما يقصدب "MIT" معهد ماساتشوستس للتكنولوجيا هذا حسب الترجمة والله اعلم] , بنى " فانيفار بوش " محلل تفاضل ذو نطاق واسع يدمج التكامل مع القدرة على التمييز . وهذا الإنشاء الواسع قد تم تمويله بواسطه مؤسسة " روكيفيلير " . وكان الحاسب "Computer" الاكبر فى العالم .

اللاعب التالى الرئيسى فى هذه القصة هو العالم الألمانى " كونراد زوس " . فى 1935 , قام هذا العالم بتطوير أول حاسب له بإسم الحاسب زد 1 Z-1 computer . وما كان أكثر من رائع أنه بنى الحاسب فى غرفة المعيشة وكان هذا أول حاسب يستخدم التعاقب فى العمل [التتابع] وإعتمد ذلك على النظام الثنائى الحسابى Binary . وكان الأول من نوعه وكان هذا تبشير بعصر الحاسبات الحديثة .

وأستمر هذا العالم الألمانى فى عمله وطور Z-2 فى عام 1938 بمساعدة من " هيلموت شريبير " . ودعى إلى مساعدة مالية من الحكومة الألمانية لتطوير وتشيد آله . ولكن رُفض طلبه لأنه سيأخذ وقتاً

لإنهائه أطول من الوقت المتوقع لإنهاء الحرب . وفي نهاية المطاف هرب هذا العالم إلى "هينترشتاين" وذلك عند قرب الحرب من الإنتهاء . ومؤخراً جعل طريقه إلى سويسرا حيث أعاد بناء حاسبه Z-4 في جامعة زيورخ .

وكما /عطى" زوس" الميلاد للبرمجة الحديثة في عام 1946 عندما طور أول لغة برمجة في العالم , تدعى Plankalkül . بل وصل إلى حد كتابة كود للعب الشطرنج ضد نفسه , وكانت هذه اللغة فتحاً . واحتوت على ما تراه في اللغات الحديثة مثال ذلك الجداول و قواعد البيانات .

فذهب "زوس" في وقتٍ لاحق إلى تأسيس شركة حاسب التي اندمجت في نهاية المطاف إلى "شركة سيمينز" . شهد عام 1945 تطور هام آخر في العمليات على الحاسب وكانت المقدمة للكلمة التي تشعنا بالضيق لمجرد سماعها وهي "Bugs" [هي "الشوائب" كترجمة وهي الأعطال والأخطاء وكل شيء يعترض تنفيذ العمليات , ومنها وصل إلى الأمر 'Debug' وهناك ترجمات تقول جراثيم] , في 1945 , كانت "جرايس موري هوبير" (لقبت مؤخراً " العميد هوبير ") تعمل في جامعة هارفارد على ما يسمى Mark II Aiken Relay Calculator .

وكانت الآلات في تلك الأيام تُظهر الأخطاء المستمرة , وخلال واحدة من هذه الحوادث في سبتمبر 1945 كشف متخصص فني في تحقيق اجراه أن هناك "moth" [الترجمة غريبة ولكن إعتبر أنه وجد شيء غير طبيعي والأقرب "جرثومة"] محصورة بين الدوائر ("أظهر السجل انه كان في نقاط التقوية #7 في لوحة F ") , فقام هذا الفني بحذف ما وجده وأضاف ذلك إلى الدفتر الذي يُعد مستند لأستخدام الحاسب ومشكلاته . المدخل في هذا الكتاب يقول "الحالة الحقيقة للشوائب قد عُثر عليها " . قادت هذه الجملة إلى مولد ما يعرف بأن الآلات يتم عمل Debugged لها [تصحيح أخطاء] . وظهرت المصطلحات " debugging a Computer" و مؤخراً ولد مصطلح "debugging a computer program" .

يكشف التاريخ أنه على الرغم من أن *Grace Hopper* كانت حريصة على الإعتراف انها كانت بعيدة عندما حدث ذلك بالفعل , إلا أنها كانت واحدة من القصص التي ترويها في كثير من الأحيان .

أخذت الأشياء في السرعة وشهد عام 1949 تطوير لغة برمجة تدعى Short code . وكان لابد أن نجعل هذا الكود ككود مقروء آلياً يكتب يدوياً (تعرف العملية بـ Compiling). [المقصود بها ترجمة الكود المكتوب يدوياً لتفهمه الآله ومعنى "ترجمة" هي تحويل لغة إلى لغة أخرى فالكود مكتوب بلغة نفهمها ولا يفهما الحاسب فيتم الترجمة للحاسب بلغته وهي لغة الأرقام] .

في 1954 بدأت شركة IBM في تطوير لغة تدعى FORTRAN (وهي إختصار FORmula TRANslator) لاحظ الحروف الكبيرة ومعناها ترجمة الصيغ حيث أنها كانت صيغ رياضية , ونُشرت FORTRAN في عام 1959 وحققت نجاح فوري وذلك لسهولة إستخدام نظام الإدخال والإخراج بها . وأيضاً بسبب أن الكود كان قصير ويمكن تتبعه (وظلت في الإستخدام في بعض الاماكن ليومنا هذا) .

وكانت Fortran أيضاً اللغة التجارية الأولى عالية المستوى البرمجي . يعني ذلك انها كانت أيسر لأن يقرأها البشر وذلك لإتباعها للقواعد النحوية والإملائية المألوفة [Grammar – Syntax] .

في 1958 . نشرت لغتا Fortran II و ALGOL وكما جرى العمل على إخراج LISP , بينما في عام 1959 أنشأت لغة منتشرة وأكثر حياة وهي COBOL (Common Business Oriented Language) وقد أنشأت في إتحاد اللغات وأنظمة البيانات (CODASYL) [تم تأسيس هذا الإتحاد لتكنولوجيا المعلومات لوضع معايير قياسية للغة البرمجة التي يمكن إستخدامها في العديد من الحاسبات "يمكنك البحث للإستزاده"] . إستخدمت COBOL كلغة مؤسسات تجارية في بداية الأمر في شركات كبرى ثم ظلت مستخدمه في كثير من الشركات في يومنا هذا .

واخذت الأشياء فى تطور سريع إلى حد خطير وظهرت العديد من اللغات وكذلك تغييرات على لغات موجودة بالفعل. وخلال 1968 بدأ العمل على لغة تسمى Pascal وتم إطلاقها فى عام 1970 وظلت مستخدمه على نحو واسع فى الأغراض التعليمية. وشهد 1970 إشهار لغتين أثاروا عالم الحاسب وكانتا Smalltalk و B-language. وكانت Smalltalk ذو اهمية لإعتمادها على ما يسمى Objects [أثرت عدم ترجمتها لكن ورد فيها أنها أشياء . كائنات . اجسام] وكانت أيضاً لغة B-Language مهمة لما تؤدى إليه.

إلى ماذا تؤدى لغة B-Language ؟ حسناً , فى عام 1972 أنشأ Dennis Ritchie لغة برمجة تعتمد على لغة B-Language التى سميت مؤخراً بلغة C (بعد تسميتها NB لبعض الوقت) , أحضرت C معها البساطة والكفاءة والمرونة وكانت صاحبة التبشير لعصر جديد فى البرمجة التى تسمح لك بفعل الكثير مما تريد بكتابة كود أقل [كلمة كود أرى أقرب ترجمة لها قانون وذلك لتعلقها بالرياضيات فى الأصل وبناء الحاسب بقوانين رياضية والله اعلم] وأكثر سرعة ويسر من ذى قبل .

شهد عام 1975 إطلاق لغة TinyBasic بواسطة Dr Wong . اخذت tinyBasic فى الذاكرة مايساوى 2 KB وكانت تُحمّل على الحاسب من خلال شريط ورقى . وكانت الرائدة أيضاً بسبب انها اول نوع يعرف بالبرامج المجانية (لذلك , كان يمكن إستخدامها بدون الحاجة إلى شرائها) .

[الجملة التالية وردت كدعابة من المؤلف وأثرت عدم ترجمتها لتحفظ برونقها]

The text strings "All Wrongs Reserved" and "Copyleft" were found within this program.

مصادفةً , عام 1975 كان أيضاً عام الشابين Bill Gates , Paul Allen حين كتبوا وباعاً أيضاً إصدارهم من لغة Basic إلى معهد ماساتشوستس للتكنولوجيا (MIT)

إستمرت الأشياء فى التغير إلى حد خطير خلال 1970 و 1980 و 1990 مع الكثير والكثير من التطور الذى قادنا إلى ما نحن عليه الآن . مع العدد الضخم من لغات البرمجة المختلفة حيث تمتلك كلاً منهما قوتها وضعفها , وكما رأينا أيضاً ظهور الإنترنت الذى أتى إلينا بواسطة إستضافة العديد من اللغات . الميزة الأخرى للإنترنت هى امكانية مشاركة المعلومات والبرامج مع الآخرين , مما يعنى زيادة نمو البرمجة ولغات البرمجة . ونمو التبادل المجانى للمعلومات والأفكار والتطبيقات .

لقد أخذنا رحلة مختصرة خلال تاريخ البرمجة مع تسليط الضوء على الأحداث المهمة التى قادتنا إلى ما نحن عليه الآن . وجاء الوقت لمعرفة ما هى البرمجة ؟

What Is Programming?

ماهى البرمجة ؟

ليس هناك إجابة واحدة لهذا السؤال . البرمجة لها تعريفات مختلفة لأشخاص مختلفة [يقصد هنا أن كل شخص يراها بتعريف غير الآخر]. ومع ذلك , فهناك تعريف أشعر بالراحة والسعادة معه وأحسبه جيداً هو :

البرمجة هي القدرة على مخاطبة الحاسب بلغة تُفهم وباستخدام قواعد إملائية ونحوية يمكن إتباعها لتحصل على أداء جيد للمهام التى تريد .

ضعها فى مصطلحات بسيطة , هذا كل شئ , وهكذا البرمجة تكون ! تكتب كود والحاسب يترجم طلبك ويفعل لك شئ ما .

فعل شئ ما جزء حيوى فى البرمجة . فانت دوماً تبرمج الحاسب لفعل شئ ما . (حتى الإنتظار ") فالبرمجة كلها تأتى من فعل شئ والتقدم الى الأمام . فحينما تبرمج لايهم إن كانت مشكلتك بسيطة أو معقدة . ففى الحالتين دوماً تعطى الحاسب تعليمات لينفذها . وعادة ما تعطى هذه التعليمات واحدة كل مرة . وحتى على الرغم من انه يبدو انك تريد من الحاسب أن ينفذ الكثير من الأشياء المختلفة فى وقت واحد . إلا انك تعطيه التعليمات خطوة بخطوة بالترتيب الى ما تريد .

أيضاً لابد أن يكون الكود صحيحاً ولا لبس فيه . ولا يمكن أن يحتوى على أخطاء , أو أعطال أو غموض . فوجود أياً من هذه الأمور يتسبب فى فشل الكود وحدث خلل فى البرنامج . فليس للحاسب القدرة على تصحيح الأخطاء بنفسه أو أن يخمن المقصود من الكود .

شئ آخر ستلاحظه فى بدايتك للبرمجة , فإنك نادراً ما ستكتب كود لينفذ شئ واحد وواحد فقط , حتى أبسط المشاريع تقوم على خطوات متعددة :-

- يتم تنفيذ البرنامج
- التحقق من البيانات الأولية.
- تغيير هذه البيانات .
- الإيضاح الكامل بعد التشغيل. [اى تنفيذ البرنامج لما نريد]
- إنهاء البرنامج .

Why So Many Programming Languages?

لماذا نجد العديد من لغات برمجة ؟

لكنك إذا إعطيت الحاسب تعليمات فى شكل يفهما . كيف أتت هذه اللغات العديدة المختلفة ؟ وهل بالتأكد ستكتب كود يفهمه الحاسب ؟

حسناً . هذا صحيح . فالأنواع المختلفة من الحاسبات تفهم أيضاً انواع مختلفة من الكود . (كالحاسب العادى يختلف عن اجهزة MAC) . فكل نوع من هذه الحاسبات يفهم لغة واحدة , وليست اللغة التى قمت بكتابة الكود بها . فالكود الذى كتبته ليس هو الكود الذى سيفهمه الحاسب . فهناك برنامج (سواء كان Interpreter أو compiler) مطلوب لتحويل الكود من صيغته النصية إلى صيغة رقمية [Binary Language] يستطيع الحاسب إلى حل رموز هذه الصيغ . وهذا هو الجزء العظيم عن البرمجة .

يعمل الحاسب على قراءة هذه التعليمات [الأوامر] فى صيغتها الرقمية binary , التى ربما تكون مألوفة لك . لأنه نظام أرقام ولكن تختلف عن الأرقام الأساسية التى نعرفها من 0 إلى 9 بأنها محصورة بين الرقمين 0 و 1 .
والشكل التالى يعرض الأرقام العشرية يقابلها النظام الثنائى :

Decimal digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9	
Binary bits: 0, 1	
Decimal	Binary
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001

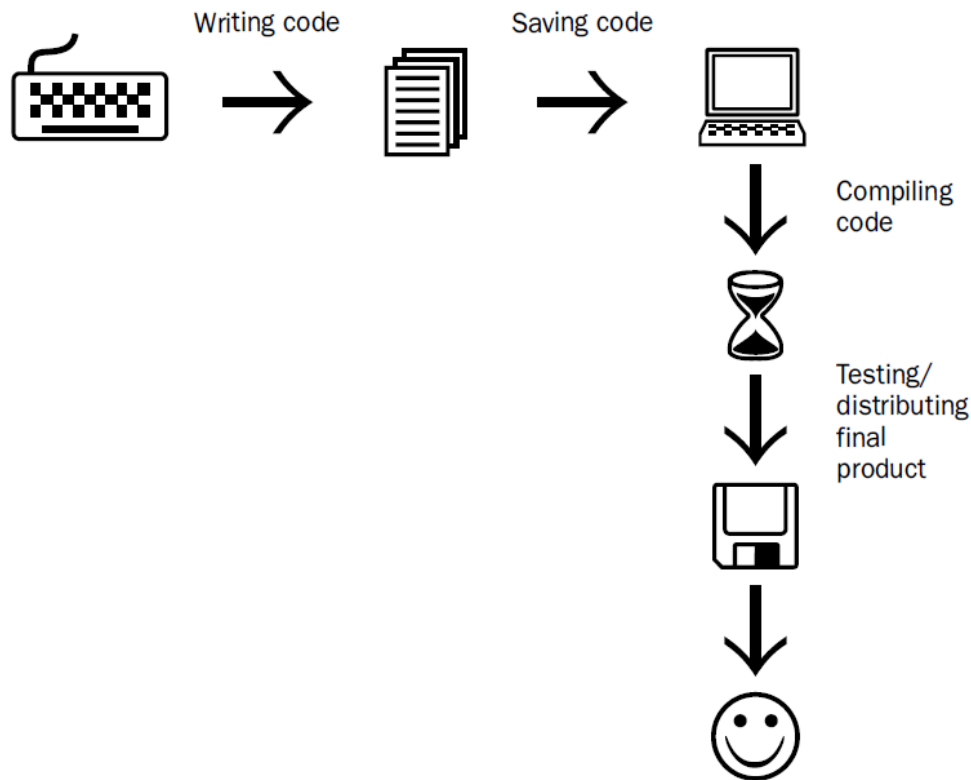
والآن إلقى نظرة إلى هذا الشكل :-

```
01000010011010011011011100110000101110010011110010010000001101001011100110010000
0011101000110010101100100011010010110111101110101011100011001000000110000101101110
01100100001000000110100101100110001000000111011101100101001000000110100001100
00001101001011011100010000001100010011010010110111001100001011100100111100100100
00000111011101101001011101000110100000100000110111101110101011100100010000011
00011011011110110110101110000011101010111010001100101011100100111001100100000111
010101110011011010010110111001100111001000001101001011101000010000001100001011
0100011011000010000001110100011010000110010100100000011101000011010010110110101
10010100100000011101000110100001101001011011100110011101110011001000000111011101
10111101110101011011000110010000100000011001110110010101110100001000000111011001
1000101011100100111100100100000011000110110111101101011100000110110001101001011
0001101100001011101000110010101100100001000000110100101101110011001000110010101
1001010110010000100000001010000110000101101100011101000110100001101111011101010
110011101101000001000000110111101110101011101000010000001101011011001010111100101
1000100110111101100001011100100110010000111001100100000011101110111101110101011
01100011001000010000001100010011001010010000001100001001000000110110001101111
011101000010000001110011011010010011011010111000001101100011001010111001000010000
1001010010010110
```

النظام الثنائى Binary نظام ممل . وإذا كان علينا ان نعمل بهذا النظام فسنستخدم أشياء معقدة طوال الوقت . هل تتسائل ماذا يمثل الشكل بالأعلى ؟ هذه هى آخر فقرة قد كتبت بإستخدام النظام الثنائى Binary وقد إستخدم لعرض الحروف نظام يسمى ASCII (American Standard Code for Information Interchange) لاتقلق فيما يعنى هذا المصطلح (فيما بعد ستكون قادراً على العودة وقرءاته مرة ثانية) . إلى الآن كن شاكراً إلى ان البرمجة لا تعنى كتابة سلسلة متدفقه لا تنتهى من الأصفار والاحاد .

عندما تبدأ البرمجة , أو بالأخص عند كتابة الكود . ما تفعله فعلاً هو كتابة كود يستطيع برنامج آخر من فهمه . ليترجم هذا الكود إلى شئ يفهمه الحاسب . ومن هنا جاء إسم Interpreter [أى مترجم] . لذلك عندما تكتب كود فلن تتبع قواعد الحاسب بل ستتبع قواعد المترجم (أو ما يشبهه وهو Compiler) الذى بدوره سترجم الكود إلى ما يفهمه الحاسب .

فالشكل التالي يعرض عملية كتابة الكود :



Different Code, Same Results

كود مختلف ونفس النتائج

لكي نريك كيف يكون الكود مختلف ولكن نفس النتائج . إلقى نظرة على المثال التالي في لغات مختلفة . ربما سمعت عن بعضها .

BASIC

```
10 print "Hello World!"
20 goto 10
```

Atari BASIC

```
10 REM HELLO.BAS
20 POKE 764,255
30 PRINT "Hello World"
40 IF PEEK(764)=255 THEN GOTO 30
```

C

```
#include <stdio.h>

main()
{
    for(;;)
    {
        printf ("Hello World!\n");
    }
}
```

C++

Old version of C++ code:

```
#include <iostream.h>

main()
{
    for(;;)
    {
        cout << "Hello World! ";
    }
}
```

Newer version of the code:

```
#include <iostream>

int main()
{
    std::cout << "Hello, World!\n";
}
```

COBOL

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID. HELLOWORLD.
000300 DATE-WRITTEN. 02/09/04 17:24.
000400* AUTHOR FRED F
000500 ENVIRONMENT DIVISION.
000600 CONFIGURATION SECTION.
000700 SOURCE-COMPUTER. RM-COBOL.
000800 OBJECT-COMPUTER. RM-COBOL.
000900
001000 DATA DIVISION.
001100 FILE SECTION.
001200
100000 PROCEDURE DIVISION.
100100
100200 MAIN-LOGIC SECTION.
100300 BEGIN.
100400 DISPLAY * * LINE 1 POSITION 1 ERASE EOS.
100500 DISPLAY "HELLO, WORLD." LINE 15 POSITION 10.
100600 STOP RUN.
100700 MAIN-LOGIC-EXIT.
100800 EXIT.
```


FORTRAN

```
c
c Hello, world.
c
Program Hello

implicit none
logical DONE

DO while (.NOT. DONE)
write(*,10)
END DO
10 format('Hello, World.')
END
```

Java

```
class HelloWorld {
public static void main (String args[]) {
for (;;) {
System.out.print("Hello World ");
}
}
}
```

JavaScript

```
<title>
Hello World in JavaScript
</title>
<script>
alert("Hello, World!")
</script>
```

Mathematica

```
While[True, Print["Hello, World!"]]
```

Pascal

```
Program Hello (Input, Output);

Begin
Writeln ('Hello World!');
End.
```

Perl

```
print "Hello, World!\n" while (1);
```

Python

```
while (1) :
print "Hello World";
```

QBASIC

```
begin:
print "Hello World!"
goto begin
```

Smalltalk

```
Transcript show: 'Hello World';cr
```

Visual Basic

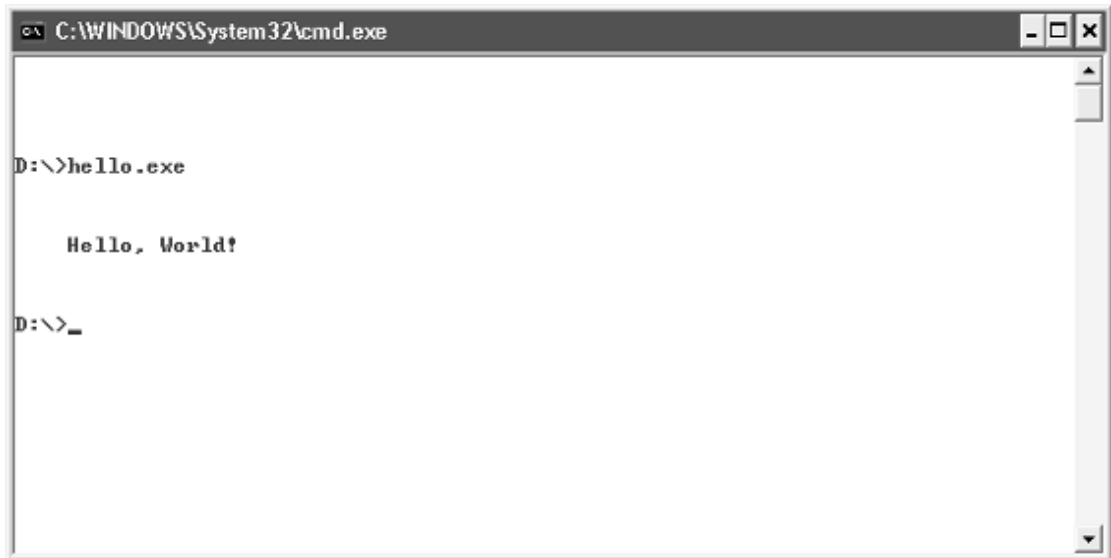
```
Private Sub Form_Load()
Static I
I = 1
for I = 1 to 10
msgbox "Hello, World!"
Next I
end sub
```

VRML

```
#VRML V1.0 ascii

AsciiText {
  string "Hello, World!"
  justification LEFT
}
```

كل هذا الكود الذى تم كتابته فى لغات متنوعة يفعل شئ واحد وهو عرض "Hello , World !" فى شكل أو آخر . وهذا مثال لما كتب فى لغة C++



مثال آخر بلغة JavaScript



لماذا جملة "Hello World" ؟

لماذا تستخدم هذه الجملة بكثرة فى امثلة البرمجة ؟ الأصل والتاريخ وراء هذه الجملة هو ولو بمحمل الإسطورة والكذب , مع ذلك , يبدو أن اول درس إحتوى على إشارة إلى جملة "Hello, World !" كان فى لغة B-Language التى تسبق C .

ويبدو أنها كانت من قبل مستخدمه كطريقة بسيطة لمعرفة ان كل شئ يعمل على ما يرام , وأن اللغة تعمل . لذلك إستخدمتها , ومن انا حتى أكسر التقاليد !

Programs Needed to Create Programs

البرامج التي تحتاجها لإنشاء برامج

وعليه , هل تستطيع بناء برنامج من لا شيء ؟ حسناً , تستطيع , والأشخاص الذين سبقوك إحتاجوا أن يفعلوا ذلك . ولكن بعد إنشاء أول لغة برمجة كان من أول الأشياء التي فكر بها علماء الحاسب هي البدء بإنشاء برامج تجعل كتابة برامج أخرى على قدر من السهولة . ومنذ أن حدث هذا فقد أصبح أيسر وأسرع أن تكتب كود في لغة عادية . وبمجرد إستخدام لغة واحدة لإنشاء ادوات تعمل مع لغة مختلفة تماماً أو لغة جديدة . هكذا كانت لغة واحدة تستخدم لإنشاء ادوات تعمل مع لغات مختلفة تماماً (أو جديدة) فقد مكن هذا الأمر اللغات الجديدة من أن يكون بها المزيد من التعقيد والتطور . مما يعنى أنه قد إنتهى بهم الأمر لأداء المزيد من المهام .

هناك نوعان من البرامج التي تساعدك لتكتب كود وتنشئ برنامج :

- A Development Environment بيئة التطوير أو بيئة إنشاء البرامج .
- Compiler المفسر .

الـ *Interpreter* المذكور من قبل لا حاجة له في كتابة كود , بل يستخدم لتنفيذ هذا الكود .

Development Environment

بيئة التطوير

بيئة التطوير هذه هي برنامج تستخدمه لتكتب الكود بداخله . بعض اللغات تحتاج إلى بيئة تطوير خاصة (على سبيل المثال : Visual Basic) . بينما البعض الآخر يحتاج فقط إلى محرر نصوص (JavaScript مثال على اللغة التي يمكن ان تعمل بإستخدام لا شيء سوا محرر النصوص) , كما أن هناك أيضاً عدد كبير من اللغات التي يمكن أن تستخدم معها أى بيئة عمل تقوم بإختيارها . (C++ واحدة من هذه اللغات) .

بيئات التطوير نناقشها بتفصيل أكثر فيما بعد , والآن , إعرف انها موجودة فهذا يكفي .

Compilers

المفسرات

هو البرنامج الذي يحول الكود الذي كتبته إلى كود يفهمه الحاسب . وتعرف هذه العملية بـ Compilation . التفسير .

فهو يقرأ الكود الذي كتبته ويفصحه من الأخطاء ويتأكد من انك تتبع كل القواعد . وإذا وجد مشكلة في الكود . فسيخبرك الـ compiler بها (إذا كان Compiler جيد) وبالتالي يعلق العملية . وإذا لم توجد مشكلة . فسيخرج لنا الـ Compiler برنامج يمكن تشغيله (أو تنفيذه) لك . وسينفذ التعليمات التي أعطيتها إياها من خلال الكود , يعرف هذا الملف في الحاسب بالملف التنفيذي ولديه إمتداد .exe . كما في الشكل :



hello.exe

Summary

الملخص

فى هذا الفصل : ألقينا نظرة على حقيقة ما هية البرمجة , والآن لا أتوقع بعد إنتهاء هذا الفصل إنه لابد وأن يكون لديك أى فكره عن كتابة برنامج , ولكن يجب أن تكون قد حصلت على لمسة فنيه أو ملخص للتالى : -

- تاريخ البرمجة .
- المقدمة لبعض لغات البرمجة .
- نبذات من بعض الكود .
- نظرة عامة على عملية البرمجة .من كتابة الكود إلى التنفيذ .
- معرفة لبعض مصطلحات البرمجة الهامة .

فى الفصل القادم , سنرى لماذا تريد أن تتعلم البرمجة حقاً !

2

Why Learn to Program?

لماذا نتعلم البرمجة ؟

لماذا نتعلم البرمجة ؟ سؤال جيد جداً ! تعلم البرمجة ليس شيئاً يمكن ان تفعله فى خمس دقائق , كما ان هناك العديد من الوسائل التى يمكن ان تأخذها والوظائف التى يمكن أن تختار منها لتدخل البرمجة حتى تبدو معقده .

وإذا ما بدأت تعلم البرمجة , متى تقف ؟ كم عدد اللغات ؟ وأى من الوسائل الجديدة ستتبع ؟

فى هذا الفصل , سأساعدك لتجيب على قليل من الأسئلة ليمكنك التخطيط لمسارك البرمجى فى وقتاً مبكراً .
القليل من التخطيط فى المراحل المبكرة لن يساعدك فقط فى إتخاذ إختيارات صحيحة فى البداية . بل سيحفظ لك الوقت والجهد أيضاً , وسيسمح لك بالتركيز على ما تريد فعله بدلاً مما تعتقد انك تحتاج إليه . وكما أنه سيدعك ترى أى أبواب البرمجة يمكن فتحه وسيكون من إهتماماتك أن تتعلم أكثر فيما يخص مهنتك .

أنا أقول أن " مهنة البرمجة " غالباً . وعندما أقول هذا فأنا لا أعنى انه يجب عليك ان تصنع من البرمجة مهنة بالشعور التقليدى عندما تستخدمها لجنى المال (على الرغم من انه ليس هناك خطأ فى هذا) . ما اعنيه أن تعلم البرمجة هو مهنة مصغرة فى حد ذاتها . ستبدأ بتعلم القليل ثم لفترة ستبدأ باستدعاء هذه المهارات . فى أثناء ذلك , ستتعلم أكثر . فكلما برمجت كلما تعلمت . على طول الطريق , ستصنع بعض القرارات (كأى اللغات ستتعلم وأى الوسائل ستتبع) . وهو ما سيصب فى مهنتك الرمجية فى المستقبل أكثر وأكثر .

بدون ذكر حقيقة ان تعلم البرمجة يمكن أن يكون دراسة على مدى الحياة..... على أكثر الأسباب مما يجعلها مهنة.

لذلك , وقبل أن نجد انفسنا عالقين مع بعض الاكواد , لنلقى نظرة على الأسباب التى وراء تعلمك البرمجة؟

Why Program?

لماذا نبرمج ؟

حتى إن كنت بالفعل قد قررت بأنك تريد ان تتعلم البرمجة , فليست فكرة سيئة أن تفكر في الأسباب وراء رغبتك لتجميع الأصفار والاحاد !

لقد وجدت أصناف عديدة للأشخاص الذين يريدون تعلم البرمجة , ها هي بعض هذه الأصناف :-

The “Career” Programmer

المهنة مبرمج

بشكل غير مفاجئ , السبب الأول أن الناس يريدون تعلم البرمجة لتغيير مهنة . فإن أى شخص يعمل مع الحاسب أو يتصفح مواقع الإنترنت لا يستطيع المساعدة في ذلك لكنه يلاحظ ان هناك برامج وصفحات إنترنت في كل مكان , كل هذه البرامج والمواقع والصور قد كتبت بواسطة احد الأشخاص , كما أن كل البرامج التي تتحكم بكل شئ و هي وراء كل هذه المشاهد و لا تراها . فإن تعلم البرمجة ينظر إليه انه المفتاح الذي يفتح الباب لكل هذا . وصنع العديد من الفرص المتاحة .

وكل هذا صحيح تماماً .

إذا كنت في وظيفة لا تريد ان تكون فيها . وتعتقد ان البرمجة تزودك بالمفتاح الذى سيساعدك بفتح باب جديد , أنت بلا شك محق . تعلم البرمجة سيفعل هذا , وسيعطيك أيضاً إختيارات , يمكنك أن تأخذ التوجيه التقليدى وتبحث عن عمل خلال المنشآت الصناعية . فإن طلب المبرمجين يزداد وينهار مع الصحة الإقتصادية والإزدهار . ولكن حتى في الأوقات الصعبة فإن المبرمج سيجد عملاً لأن البرمجة ينظر إليها كأنها مهارات خاصة , والمبرمجين في شركات البرمجة هم الرجال و النساء الحرفيين بالفعل هم على الجبهة يصنعوا المنتجات . لذلك , بينما نحزم وربما نشدد في بعض الأجزاء من الشركة أن وظيفة المبرمج غالباً ماتكون أكثر أماناً .

الإختيار الآخر المتاح أن تراهن وتصبح مبرمج مستقل وتعمل لحساب نفسك , وسيتيح لك هذا الطريق إختيار وإلتقاط الشركات والوظائف التي لديك إهتمام بها .

الكثير من المبرمجين المستقلين يتمتعون بالمرونة . لذلك في أغلب الأحيان إذا أحببت أن تغير المشهد وتقابل وتعمل مع أشخاص جدد يمكنك أن تبحث عنهم , وتأخذ الوظيفة التي تتحرك بها . بينما إذا لم تحب هذا يمكنك أن تبحث عن فرص عمل تتيح لك العمل من خلال المنزل , وتعمل ساعات عملك الخاصة . فإن عقود البرمجة العادية تكون بمدة من إسابيع قليلة إلى أشهر إلى سنوات عديدة .

إذا كان التوظيف الذاتى إختيار فى إهتمامك , تأكد أيضاً خلال تعلمك لكيف تبرمج ان تدرس بعناية الشركات , المحاسبة , القانون المتعلق بالشركات والعقود .

لا تترك هذه الأشياء للحظ .

الإختيار الآخر المتاح أن تطلق العنان لإبداعك وجانب الإختراع عندك , لإنشاء وتطوير وإطلاق تطبيقات برامجك الخاصة . ولعمل ذلك بنجاح , فعموماً تحتاج أن تجد البيئة الملائمة حسب الحاجة . ثم إنشئ البرامج التي تنتج ما تحتاج إليه . أو إن كان هناك بالفعل برنامج يلبي إحتياجاتك تريد أن تجرى عليه تغيرات على طريقتك الخاصة . فإليك بعض الطرق القليلة التي تعطى لأصحاب الأعمال الصغيرة الفرصة للمنافسة :

- **البرامج مخفضة الثمن :-** بسبب أنك أصغر مما يكون من الشركات الكبرى . سوف تخدم السوق بسعر مخفض , ولكونك مخفض الثمن فهذا متاح لتسويق أوسع ويكون متاح أكثر لعدد أكبر من الأشخاص مما يزيد المبيعات .
- **معرفة السوق :-** هل تعرف ماذا يريد الناس ؟ هل يمكن أن ترى حاجتهم إليه ؟ هل تعرف حقاً كم سيدفعون ؟ هي أسئلة مهمة . وإذا حصلت على إجابة لها فإنت بصدد الحصول على مزايا في وظيفتك .
- **مزايا أكثر محدده :** لذلك أكثر البرامج التجارية ما تأتي إلى السوق إلا بمزايا جديدة تعتقد أن المستخدم يحتاج إلى هذه المزايا الجديدة . والمشكلة أن رجال التسويق ليسوا هم الأشخاص الذين سيستخدمون البرامج في المسار الحياتي . وإذا كنت بالفعل تعمل في مجال يخدم بواسطة البرامج وتعتقد انه يمكن ان تقوم بتحسين هذه البرامج فلديك مميزات هائلة .
- **الدعم الجيد :-** ولأنك تعرف البرنامج الذي انشأته بجدارة , فانت في أفضل الوظائف لتكون قادراً على توفير الدعم الممتاز له .
- **مصاريف توزيع أقل :-** إن إطلاق برنامج جديد يكلف الكثير من المال . إنشاء أقراص للبرنامج , صناديق , مواد ترويجيه , وإحضاره إلى المتاجر , وهكذا , وبمجرد وجوده في المتجر فسيقطع المتجر أيضاً من الثمن , ليضاف على المستخدم النهائي . وانت , في الجانب الآخر يمكن أن تطلق البرنامج من خلال الإنترنت بأقل سعر وتحفظ هذا الإدخال لزيائتك .
- **أصغر , أسرع , أكثر تركيزاً :-** ولأنك لن تسعد رجال التسويق ببهجة برنامجك , يمكنك التركيز على برنامجك ليؤدي وظيفته ويؤديها أيضاً جيداً . والحقيقة المعروفة جيداً أن بعض معظم برامج الشوائب القوية المتاحة تأتي في شكل برامج مشاركة من شركات صغيرة أو أفراد .
- حتى إن كان هناك برامج تفي بكل الأغراض , فهناك دوماً مكاناً للتحسين , خذ محرر النصوص , رجوعاً إلى نظام التشغيل فستجد انه محمل ببرنامج محرر للنصوص (Windows Notepad)

```

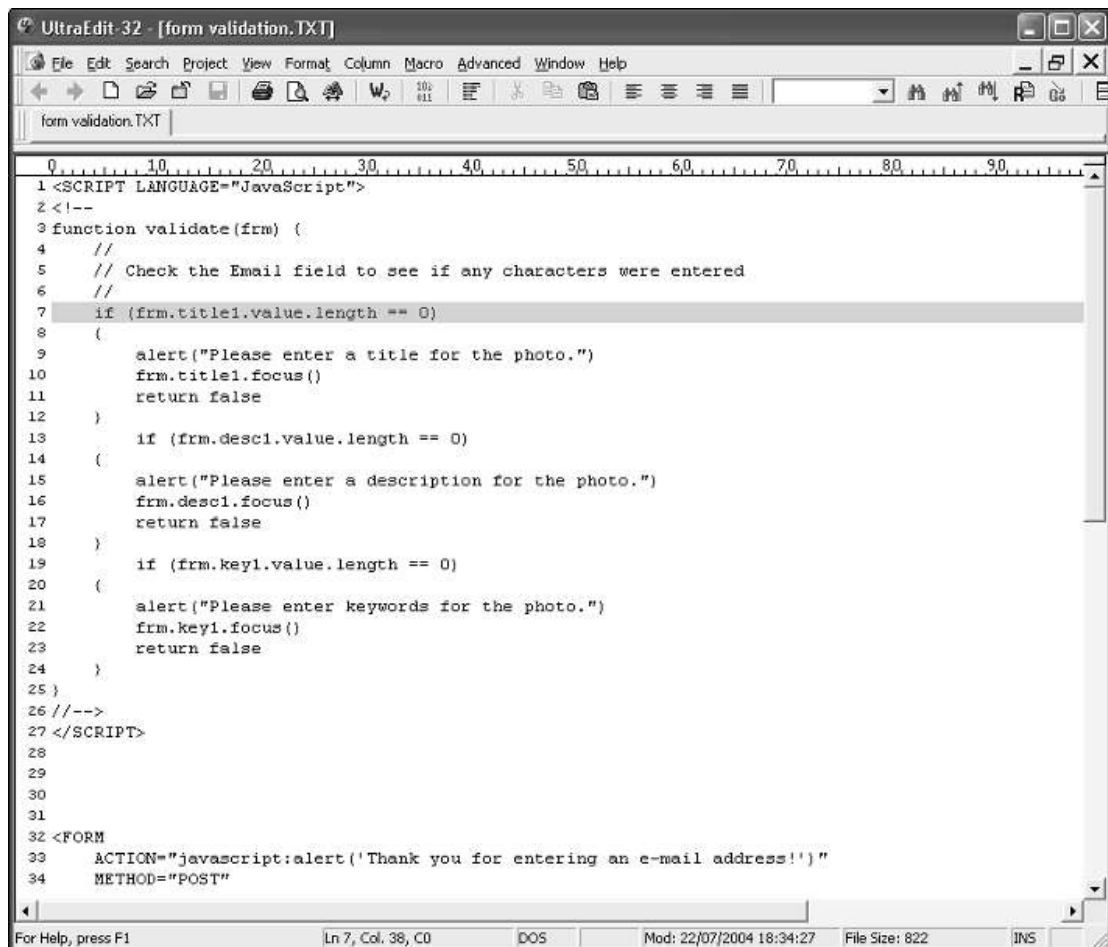
form validation.TXT - Notepad
File Edit Format View Help
<SCRIPT LANGUAGE="JavaScript">
<!--
function validate(frm) {
    //
    // Check the Email field to see if any characters were entered
    //
    if (frm.title1.value.length == 0)
    {
        alert("Please enter a title for the photo.")
        frm.title1.focus()
        return false
    }

    if (frm.desc1.value.length == 0)
    {
        alert("Please enter a description for the photo.")
        frm.desc1.focus()
        return false
    }

    if (frm.key1.value.length == 0)
    {
        alert("Please enter keywords for the photo.")
        frm.key1.focus()
        return false
    }
}
//-->
</SCRIPT>

```

ستعتقد من هذا انه لا حاجة أياً كان إلى محرر نصوص آخر , وأن السوق (على الأقل سوق الـ Windows) سيكون جيداً ومحققاً تماماً في هذا , ليس كذلك , فواحد من برامج المشاركة المفضلة لدى هو في الحقيقة محرر نصوص يسمى UltraEdit (www.ultraedit.com) يظهر في الشكل التالي :-



```
UltraEdit 32 - [form validation.TXT]
File Edit Search Project View Format Column Macro Advanced Window Help
form validation.TXT
0 10 20 30 40 50 60 70 80 90
1 <SCRIPT LANGUAGE="JavaScript">
2 <!--
3 function validate(frm) {
4     //
5     // Check the Email field to see if any characters were entered
6     //
7     if (frm.title1.value.length == 0)
8     {
9         alert("Please enter a title for the photo.")
10        frm.title1.focus()
11        return false
12    }
13    if (frm.desc1.value.length == 0)
14    {
15        alert("Please enter a description for the photo.")
16        frm.desc1.focus()
17        return false
18    }
19    if (frm.key1.value.length == 0)
20    {
21        alert("Please enter keywords for the photo.")
22        frm.key1.focus()
23        return false
24    }
25 }
26 //-->
27 </SCRIPT>
28
29
30
31
32 <FORM
33     ACTION="javascript:alert('Thank you for entering an e-mail address!')"
34     METHOD="POST"
```

هذا البرنامج الرائع كان من حوالى 1994 وإستمر فى النمو بنجاح , وكان من إبداعات رجل إسمه "إيان مياد " , لقد أخذ الفكرة من تطوير محرر النصوص الذى يفعل ما يريده الناس منه وإنتهى به إلى إنشاء هذه الجائزة UltraEdit .

UltraEdit هو محرر نصوص رائع وهو الوحيد الذى أستخدمه الآن وذلك لأنه يفعل كل شئ أريده من محرر النصوص . كل مبرمج يحتاج إلى محرر نصوص جيد , وهذا هو الخاص بى , إلقى نظرة على الملحق B [فى آخر الكتاب] لقائمة بالأدوات , بما فيها محرر النصوص , معظم العروض لما يطرح مجاناً للتجربة , لذلك إعطهم بعض الإختبارات واحصل على الذى يفى بإحتياجاتك .

لا تعتقد أنه ليس هناك أمل لفكرتك لأنها موجودة من قبل , فقط تعديل أو تحسين واحد يمكن ان يحدث فرق ضخم (لكن إن كان بإمكانك إضافة عدة تحسينات , سيكون أفضل بكثير) .

Problem Solving

حل المشكلات

سبب آخر يجعل الأشخاص يريدون تعلم البرمجة وهو رؤيتهم للبرمجة على انها أداة يمكن أن تساعدكم في حل مشاكلهم سواء في العمل أو في حياتهم الشخصية , يرون المشكلة فيعتقدون أن البرمجة يمكن أن تحلها . المقاييس المتنوعة لمثل هذه المشاريع يمكن أن تكون ضخمة , والمشكلة يمكن أن تكون واحدة صغيرة(مثال التطبيق الذي يجرى عملية حسابية بسيطة) , ويبدو من هذا أنها مشكلة يمكن أن تحل بأسطر قليلة من الكود (في الحقيقة لقد رأيت أناساً تمكنوا من حل مشكلة عويصة ظلوا في مواجهتها شهور أو سنين وتم حلها بسطر واحد فقط من الكود) . أو تكون المشكلة أكثر تعقيداً وتحتاج إلى مجموعة كبيرة من الأكواد (على سبيل المثال تطبيق يجعل العمل أو الهواية تعتمد على الكمبيوتر لا على الورق)

هناك عوامل كثيرة ستجعلك تقرر ما إذا كان حل المشكلة يحتاجك ان تتعلم البرمجة او أن تحضر احداً ما يعرف بالبرمجة .

How Big Is the Project?

ما مدى كبر المشروع ؟

في المشروع الابتدائي , أفضل , و أحسن طريق لتعلم البرمجة هو أخذ خطوات صغيرة وصنع قطع صغيرة [يقصد برامج صغيرة] . وفي هذا الطريق ستحصل على تعزيز إيجابي لأنك في حقيقة الأمر تكتب البرامج , و تحل المشكلات , وترى إحراز التقدم لنفسك , فالعمل على مشاريع صغيرة يعطيك تأسيس برمجى جيد , وذلك لأنك ستتعامل مع الأساسيات بكثرة .

لا شئ يسبب الحرج أكثر من تواجد المبرمجين المبتدئين في وظيفة تحتوى على مشاريع ضخمة وتكون مسئولة منهم فعندها لا يبدو أنهم سيحرزون أى تقدم .

إذا كنت تريد العمل على مشاريع ضخمة ولا تريد أن تضيع وقتك في مشاريع صغيرة , فأفضل عمل لك هو تحليل برنامجك إلى اجزاء . وتتعلم ماذا تريد أن تتعلم لأداء البرنامج .

مشكلة أخرى في قفز خطوات إلى مشروع ضخم , فكما تعلم , كثيراً ما ستكتشف قيامك بعمل أشياء خطأ في البداية ويستوجب عليك الرجوع للخلف لإعادة كتابة مجموعة ضخمة من الكود . فالتخطيط لمشروع كبير وكتابة الكود له لا بد لها من ممارسات لتمنعك من كتابة كود غير صحيح وتأخذك في دوران للخلف ومن ثم إلى طريق مسدود . الأفضل كلما تحسنت برمجتك , فقد أصبحت ذو كفاءة عالية في كتابة كود . فالتطبيق الذي يأخذ في البداية 1000 سطر من الكود ربما سيأخذ فقط 300 بعد زيادة معرفتك قليلاً [يقصد هنا فارق الخبرة] سبعمائة سطر ستقل من الكود مما يعنى تطبيق أسرع وأسطر قليلة للتصحيح وإحتمالات أخطاء أقل .

Is the Project Time Critical?

هل وقت المشروع حرج ؟

مشاريع الوقت الحرج تكون عادةً الأسوأ لتبدء بها مهنتك البرمجية , ضغط الوقت ويتبعه خوف من الفشل . مما يجعله بالكاد موقف خاسر لمعظم الأشخاص . أضف إلى هذه الحقيقة أنك إذا ما ألقيت العبء على نفسك وإستنفذت الوقت . ستشعر بالضيق حياله وربما لن تنفذ أى شئ فى تعليمك للبرمجة .

فقط تعرف كيف تعمل تحت ضغط , بعض الناس تزدهر تحت الضغط , فإن كنت واحداً منهم فقد يكون هذا تحدى تريده لتتعلم البرمجة سريعاً , ومع ذلك معظم الأشخاص غير مرتبطين بهذه الطريقة من العمل , ويجدون ان ضغوط المواعيد النهائية تؤثر على أدائهم وتعوق تعلمهم . ولهذا لا أوصى بها .

Work or Hobby?

عمل أم هواية

هل المشروع هو مشروع متعلق بعمل أم هواية ؟ مشاريع الهواية هى المشاريع التى تكرر لها الوقت حين تجد الوقت متاح . ولكن المشروع المتعلق بالعمل ربما تجد به ضغط عليك لتصنع تقدم فيه .

كقاعدة : كلما قل الضغط عليك كان أفضل .يُمكنك هذا من التعلم بدون ضغوط تؤثر على تعلمك .

ومع ذلك , لا تقلل من المشاريع المتعلقة بالعمل , ربما يوفر لك العمل الوقت المناسب وكذلك المصادر التى تحتاجها لتصبح مبرمج .

إحرص على المتابعة ما إذا كان رب العمل يقدم أى دورات تدريبية متعلقة بالعمل .

Because You Want to Contribute

لأنك تريد أن تساهم

Open source أصبحت المصادر المفتوحة أكثر وأكثر شيوعاً هذه الأيام , وهو مصطلح تم إستخدامه لتحديد ان التطبيق العام أو جزء من البرامج متاح مجاناً للآخرين ليلقوا النظر عليه .ويجربوه ,ويضيفوا التحسينات عليه , عادة ما تكون برامج المصادر المفتوحة برامج مجانية , وتمتلك وجودها على أجهزة المستخدم ومهداه إلى كل المبرمجين من جميع أنحاء العالم .

بعض الأشخاص الذين إستخدموا برامج المصادر المفتوحة (وخاصة من يمتلكون نشاطاً فى مجتمع إختبار الأكواد أو إقتراح مزايا جديدة) بدأوا بالشعور بوجوب المساهمة , ولما لا ؟ فلديهم حق الدخول إلى الكود ومجتمع المبرمجين الذين سيبدلون الوقت والجهد (وأحياناً المال) للمشروع . معظم الأشخاص ذوى العلاقة بهذه النوعية من البرامج يجودون بالوقت والمعرفة وتقديم العون إلى المبتدئين .

إذا كنت ذو علاقة بمجتمع كهذا , إلقى نظرة بجوارك لترى ما إذا كانوا يقدمون الدعم لبناء المبرمجين _____ ربما تكون محظوظ ! .

Just for Fun

فقط للمرح

كان لي من المحادثات طويلاً أكثر من مرة في الأسطر التالية :-
أنا : أها , انت تريد ان تتعلم البرمجة ؟

الطالب : نعم

أنا : لماذا ؟ هل للعمل أم للهواية لمشروع في خاطرك ؟

الطالب : هممم حسناً أنا أريد فقط أن اتعلم البرمجة أعتقد أنه سيكون الأمر مسلياً .

هذا ليس جنوناً كما يبدو , بعض الأشخاص يقوم بحل الكلمات المتقاطعة للمرح فقط . أو بلعب الشطرنج أو فقط يجرى أو بالسير على الأقدام أو يعمل على سيارته أو شاحنته . أو بالذهاب للسينما أو بلعب ألعاب الحاسب . لماذا لا تكون البرمجة للمرح ؟ إنها تحدى وتحفيز عقلى . فأنت تتعلم شيئاً ما , وتقود إلى أشياء أخرى . فإذا كنت واحد من الذين يتعلمون البرمجة "فقط للمرح" فلديك ميزة هائلة لأنك ستتهج المادة في حالة هادئة , ستعطيها كل ما لديك طالما تريد , ثم تتركها لفترة حتى تكون مستعداً ثم تعود لتستزيد . في هذه الحالة , ستفاجأ بكمية ما تعلمت وكم ستشعر بقلة الجهد , فالأشخاص الذين يتعلمون للمرح محظوظون .

Fame

الشهرة

هل تتعلم البرمجة من أجل الشهرة ؟ إنساها .

What to Learn?

ماذا تتعلم ؟

ملخص تاريخ البرمجة الذى أعطيتك إياه فى الفصل السابق لابد ان يكون قد جعلك مدرك لحقيقة أن البرمجة لها نطاق هائل — لغات مختلفة , تطبيقات مختلفة , أسباب مختلفة . فالكمل يمكن ان يكون هائل . وتحتاج اعمار مديدة لتعلم و دراسة كل هذا .

ولأنه من الهام لك أن تعمل مبكراً على منطقة إهتماماتك فى البرمجة .

الأقرب أن تعمل على تنقية "البرمجة" فى شئ أكثر تحديداً , هذا الأفضل . بعدها ستضيق تركيزك وإهتمامك لتتعلم أشياء محدده لا أن تقضى عمرك فى تعلم أشياء عامة .

Programming Avenues

دروب البرمجة

يوجد منطقتين رئيسيتين فى البرمجة يمكن ان تختار منهم :

تذكر , ليس مجرد إقرارك بإحد دروب البرمجة الآن لا يعنى أنه ليس بإمكانك ان تغير رأيك وتأخذ درب آخر . ولا يعنى أيضاً أنك لن تحصل على اكثر من لغة برمجة فى جعبتك قريباً .

الإقدام إلى اخذ القرار الآن يساعدك أن تركز إنتباهك على ما اهم ما تحصل عليه . فعندما تستخدم لغة واحدة بارتياح , عندها يمكن أن تجرب مع الآخرين بسهولة .

- البرمجة التقليدية .
- برمجة الإنترنت (Web) .

كلاً منهما له فروع كثيرة , دعنا نلقى نظرة القليل الآن .

Traditional Programming البرمجة التقليدية

هو مصطلح تم إعطائه لبرمجة التطبيقات التي تعمل على أجهزة الحاسبات . ربما يبدو أنه الإمتياز الغريب أن تجعل البرامج تعمل على الحاسب على أية حال . ولكن مساعدة ان نميزها عن التطبيقات الهامة للإنترنت .

ليس من الدقة تقنياً أن نقول أن كل البرامج تعمل على الحاسبات وذلك لأن العديد من المواد تجدها من خلال المنزل تحتوى على كود مما يمكن ان نعتقده برنامج . مواد كمشغل الأقراص الرقمية , شاشة الألعاب , الآلات الحاسبة , حتى السيارات — ليس من هؤلاء ما يعد من الحاسبات , ولكنهم بلا شك قد يحتوى الواحد منهم على تطبيق صغير وكود يحدد ماذا تفعل هذه المادة وكيف تتصرف .

البرمجة التقليدية قد أسست جيداً وحتى مؤخراً كانت منطقة معظم المبرمجين يختارونها , إلى حدٍ ما لأنها كانت الجزء الأكبر ولكن أيضاً لأن الإنترنت كانت فى طفولتها .

البرمجة التقليدية عادة ما تقسم بناءً على ما هي لغات البرمجة التي يستخدمها الناس . ومع ذلك ولوجود لغات برمجة كثيرة لتغطيتها وإختلاف بينهم غير ملحوظ , فسأقسم البرمجة التقليدية إلى نوعين من البرمجة :

- تجارى / تطبيق
- تعليمى / أكاديمى .

Commercial/Application Programming برمجة التطبيقات التجارية

البرمجة التجارية هي نوع من البرمجة كانت نتيجة لإنشاء لمعظم التطبيقات التي نراها حولنا . يوجد هناك العديد من المقاييس للبرمجة التجارية ولكن لأغراضنا هنا لا نريد أن نقلق لأجلها .

هناك عدد من التطبيقات مناسبة تجارياً إلى برمجة تطبيقات التجارية :-

- C / C++
- Java
- Visual Basics

يوجد هناك لغات كبيرة لكونها مليئة بالمزايا والقوة . ومع ذلك , الذهاب يد بيد مع هذه المزايا الكثيرة والمرونة , فيه كثير من التعقيد , وهذه اللغات من المحتمل أن تكون من أصعب اللغات للتعلم , فالثلاثة , Microsoft Visual Basic إلى حد كبير هي الأسهل , C / C++ هما الأصعب .

صعوبة لغة البرمجة تأتي من كيفية إبتعادها من اللغات الطبيعية المكتوبة . إلقى نظرة إلى هذين المثالين لنرى الإختلاف :

هذا هو الكود فى C++

```
#include <iostream.h>
int main()
{
    cout << "Hello, World!\n";
}
```

هذا هو الكود في Java

```
class helloworldjavaprog
{
    public static void main(String args[])
    {
        System.out.println("Hello, World!");
    }
}
```

وهذا الكود في Visual Basic :-

```
Private Sub Form_Load()
    msgbox "Hello, World!"
End Sub
```

الإختلاف يجب أن يكون واضحاً في الحال! في C++, Java يوجد الكثير من الرموز داخل الكود , بينما الكود في Visual Basic فيعتمد أساسياً على إستخدام الكلمات , مما يجعل الناس دائماً يجدونها أسهل للقراءة. السؤال التالي الواضح هو " أى اللغات يجب ان أختار؟ " حسناً , إذا أردت أن تطور برنامج ليستخدم في تطبيقات الـ Windows , عندها سيكون من الأفضل أن تتعلم لغتا C++ أو Visual Basic , فتعلم C++ ستعطيك مجال أوسع وإمكانية الحصول على الكثير من الفرص , بالرغم من انها ستأخذ الكثير من الوقت للتعلم. النقطة الأخيرة التي نأخذها في الإعتبار في Visual Basic تجعل إنشاء واجهات المستخدم (ما يراها المستخدم النهائي من التطبيق) أكثر يسر . إضافة المفاتيح , مربعات النصوص , ووظائف اخرى كثيرة مع Visual Basic اكثر مما في لغات أخرى .

مع ذلك , كل هذا ليس مفقداً في لغة C++ عند العمل مع واجهات المستخدم , يمكن ان تنشأ واجهات مستخدم فعالة في C++ , ولكن المشكلة أنك تحتاج إلى بيئة تطوير متكاملة IDE تستخدمها ذات نكهة "مرئية" لـ C++ أو Java لفعل ذلك . المشكلة الرئيسية في هذا ان معظمهم يأتي بقسيمة شراء عالية. سيكون جيداً حين تستخدم حقبة التطوير الخاصة بالأعمال حيثما ستدفع لنفسها [يقصد أن الأعمال ستأتي بدخل يساعدك في دفع ثمن البرنامج] . لكن لا تفعل كهواية .

يوجد هناك بيئات تطوير مجانية وغير مجانية لـ Java , C++ يمكنك من إنشاء واجهات مستخدم فعالة , قم بفحص الملحق B [في آخر الكتاب] .

فإذا كنت تطور تطبيق لنظام التشغيل غير Windows عندها يمكنك أن تغير اللغة التي تخطط لإستخدامها .

مازالت C++ كلغة جيدة لتختارها ولكن Visual Basic لا , لأنها تعد لتطبيقات الـ Windows فقط , إختيارك للغة يصب في اللغة التي تستطيع ان تكتب بها كود يعمل على أنظمة التشغيل المتنوعة والبرامج وستكون Java . فهي أسهل من C++ واكثر قوة , لذلك لا تقارن الوظائف للحصول على منافع أكثر .

Learning/Academic Programming

إذا كنت تتعلم البرمجة تماماً من الصفر , ثم تريد أن تجعل الأمر سهلاً على نفسك , بالطبع , ستكون هناك القيود , مثال ذلك سعر البرنامج وما تريد فعله . ولكن مبدئياً إذا كنت تهتم بتعلم البرمجة أكثر من حقيقة إنشائك لتطبيق معين , فمن المنطقي أن يجعل ذلك الحياة أسهل ما يمكن .

هناك لغات صممت خصيصاً لأغراض التعلم . واحدة كمثال لغة Smalltalk . مشكلة هذه اللغات بينما تم تصميمها لتقدم أقصى منفعة أكاديمية [جامعية] . فيتعلم الطلاب لغة لن تكون في استخداماتهم في العالم الواقعي . لذا يتعلم الطلاب المفاهيم والأساسيات . ولكن لا شيء في الواقع يمكن أن يأخذه معهم في النهاية . لفعل أي شيء , بعدها سيجب عليهم أن يذهبوا بعيداً لتعلم لغة برمجة مناسبة .

هناك بعض اللغات جيدة لتستخدم لغرض التعلم . و لغة Microsoft Visual Basic واحدة منهم . هذه اللغة سهلة التعلم وتناسب كل المبتدئين لعدة أسباب منها :-

- سهولة الاستخدام — دوماً ميزة هائلة !
- لغة واقعية يمكن إستخدامها لمهام البرمجة الإحترافية .
- منخفضة الثمن .
- موثقة جيداً وتأتي بالكثير من امثلة التطبيقات والكود .

قم بفحص موقع مايكروسوفت لتحصل على تفاصيل أكثر (msdn.microsoft.com/vbasic) .

مع ذلك , منخفضة الثمن لا تعني انها بلا مقابل . لذلك , عندما ننظر إلى برمجة مواقع الإنترنت سأريك كيف تحصل على مذاق Visual Basic بلا مقابل [مجاني] .

Web Programming

برمجة الانترنت

شهدت الإنترنت نمواً ضخماً عبر سنوات عديدة . وأحد الأسباب الرئيسية التي جعلت الناس يمتلكون الحاسبات هو الدخول على الإنترنت , كثير من الأشخاص يعد الإنترنت الإستخدام الرئيسي لهم .

قدم الإنترنت ثورة في كيفية نقل المعلومات وفي نفس الوقت تغير البرمجة . البرمجة لم تعد لتجعل التطبيق يعمل فقط , فقد أصبحت الان أداة حيوية في التحكم في المعلومات لتجعلها أكثر إقناعاً .

عندما تأتي لبرمجة الإنترنت , هناك مساحتين منفصلتين من البرمجة :

- Server Side Programming [برمجة جانب الخادم]
- Client Side Programming [برمجة جانب المستخدم]

Server Side Programming

برمجة تنفيذ جانب الخادم

البرمجة جانب جهاز الخادم هي كتابة كود مشابه كثيراً لأي نوع من الكود , ومع ذلك , فبدلاً من تنفيذ الكود لأداء أي مهمة على الحاسب للمستخدم النهائي . فهو ينفذ المهام على الحاسبات المتصلة بالإنترنت التي تمسك بصفحات مواقع الإنترنت (هذه الحاسبات يشار إليها في العام بإسم الخادم Server). ويمكن أن تستخدم هذه البرمجة لتنفيذ نطاق أوسع من المهام .

الأكثر شيوعاً هو إرسال صفحة إنترنت إلى المتصفح لتعرض للشخص كصفحة إنترنت عادية . وهناك المزيد مما تستطيع وتفعله برمجة جانب الخادم:-

- تتحكم فى كيف تعرض الصفحة .
- تتحكم فى كيفية عرض البيانات .
- تخزين إعدادات المظهر للمستخدم .
- تتصل بقاعدة البيانات .

وهناك العديد من اللغات التى تستخدم للبرمجة جانب جهاز الخادم :-

- ASP (Active Server Side) تتحكم فى كيفية عرض الصفحة بالتعايش مع المستضيف, طورت بواسطة مايكروسوفت. وهو نظام تجارى يعمل مع العديد من أنظمة التشغيل الخادم التابعة لمايكروسوفت .
- PHP (Perl Hypertext Preprocessor) مشابهة لـ ASP.net . وتعد مصدر مفتوح ومتاحة بلا مقابل .
- SQL (Structured Query Language) لغة تستخدم للقراءة من او الكتابة إلى , ومعالجة قواعد البيانات على مواقع الإنترنت .

لقد تم إستبعاد لغات العلامات هنا المستخدمة فى عرض صفحات الإنترنت .

دائماً مايكون فى الصورة اكثر مما تقابلة العين , لناخذ ASP على سبيل المثال هناك لغات متعددة تستخدم لقيادة ASP . لذلك الأشياء لا تكون كما تبدو .

لن نغطى ادوات Server side فى اى تفاصيل فى هذا الكتاب لأنهم خارج نطاق المادة العلمية .

Client-Side Programming

برمجة تنفذ جانب العميل

تختلف البرمجة جانب العميل [المستخدم] عن البرمجة فى جانب الخادم فى أن هذا الكود يوجد فى صفحة الإنترنت المرسله إلى المتصفح . يقرأ هذا الكود بواسطة المتصفح ويترجمه ومن ثم ينفذ الكود . سننظر إلى هذا فى كثير من التفاصيل فيما بعد , ومع ذلك , دعنا الآن نعرفك بلغتين هامتين بدأتا الحياة وأصبحا اكثر شعبية فى اللغات التى تنفذ فى جانب العميل [اى على جهاز العميل] ولكنهم الآن قد أنجزا حالة عظيمة يمكن ان تذهب إلى أبعد من الويب (كما سترى قريباً) . هذه اللغات هى :-

- JavaScript طورت بواسطة شركة Netscape .
- VBScript لغة تنفذ جانب العميل طورت بواسطة مايكروسوفت وتعتمد على لغتها المشهورة Visual Basic .

JavaScript هى لغة برمجة نالت العديد من الشعبية على الإنترنت لكونها مدعومه دعماً واسعاً من الكثير من المتصفحات وأنظمة التشغيل بينما VBScript هى لغة لـ Windows ومتصفح Internet Explorer الخاص بـ مايكروسوفت .

ستجد الفرصة الجيدة لتستخدم الإثنين خلال دورة هذا الكتاب

كلاً من هاتين اللغتين أكثر قوة ولكن لديهما مزايا إضافية :-

- السرعة وسهولة التعلم .
- ممتازة للأغراض التعليمية .
- مطابقة للعالم الواقعى .

معظم لغات البرمجة التي تعمل في جانب العميل ليست لغات برمجة صارمة ولكن لغات كتابات [يقصد انها لغات تشبه الكتابة العادية وليست بصعوبة وقواعد اللغات الأخرى] . ولكن القواعد التي تتبعها هذه اللغات وكيفية عملها تختلف قليلاً وهذا الاختلاف يكون أكاديمي بعض الشيء .

وقبل أن ننهي هذا الفصل , دعنا نلقي نظرة على بعض الأساطير التي تحيط بالبرمجة ولغات البرمجة .

Programming Myths and Facts أساطير البرمجة والحقائق

دعنا نأخذ جولة في الأساطير والإعتقادات حول البرمجة ونرى ما له جوهر وما هو أساطير مدنية .

Myth 1 — Programming Tools Cost a Lot of Money

الأسطورة الأولى — ادوات البرمجة تكلف الكثير من المال

خطأ !

هذه هي الأسطورة الأولى حول البرمجة . كثير من الأشخاص بالخارج ممن يحبون تعلم البرمجة ولكن لديه خوف كبير من أن أدوات البرمجة التي سيحتاجونها ستكلفهم الكثير من الثروة . وهذا ليس صحيحاً , وليس شيئاً تم مؤخراً . فلسنوات مضت , صنعت الجامعات والكليات أدوات ومترجم أكواد متاحة بلامقابل من أجل المبرمجين طلاب العلم , لذلك , أليس هذا وقت جيداً لدخول البرمجة !

ضع في خاطرك أن تعلم البرمجة وإنشاء برامج تجارية امرين منفصلين وأن العديد (لكن ليس الكل) من التراخيص للأدوات التي بلا مقابل تمنع إستخدامها لأغراض تجارية . إذا كان لديك شك , قم بفحص الترخيص او إتصل بالمؤلف او الناشر .

Myth 2 — You Need a Degree in Computer Science to be a Programmer

الأسطورة الثانية — تحتاج إلى درجة علمية [شهادة جامعية] في علوم الحاسب لتكون مبرمجاً

خطأ

مطلقاً ليست كذلك , أى خبرة سابقة أو خلفية في البرمجة أو أى تعاملات مع الحاسب , لا شك في ذلك , لكن الفكرة أن أى شخص يستطيع أن يدخل البرمجة يحتاج إلى شهادة جامعية أو مؤهلات أخرى ليدخل سلم البرمجة هي فكرة لا أساس لها .

Myth 3 — Learning to Program Takes Years

الأسطورة الثالثة — تعلم البرمجة يأخذ سنين

صحيح وخطأ

البرمجة حقل ضخم من الدراسة وتستطيع ان تقضى بقية عمرك فى تعلم , تهذيب , إتقان وممارسة مهاراتك . فعندما تتقن لغة واحدة , فهناك الكثير من اللغات الأخرى , ولا تقلق من نفاذ اللغات — فسيقدم لك المطورون واحدة جديدة لك أيضاً (مثل لغات برمجة نظام .Net . من مايكروسوفت) .

ومع ذلك , تعلم البرمجة لا تأخذ وقتاً طويلاً جداً . فى الحقيقة , ستفاجأ فى كيفية سرعة الدخول لمرحلة البدء , بعض نواحى البرمجة تأخذ وقتاً للتعلم , ولكن هذا نفس الشئ مع أى شئ . الدخول إلى البرمجة ليس شاقاً .

Myth 4 — Programming Is Just for Young People

البرمجة للشباب فقط

خطأ !

أى شخص يستطيع أن يدخل البرمجة — صغير أو كبير . طالما أنك مهتم بالأمر , لديك إحساس بالمغامرة , وستضع فيها بعض الوقت . يمكنك تعلم البرمجة .

Myth 5 — I'll Need a Top-of-the-Range Super-Duper Computer to Program On

سأحتاج إلى حاسب اعلى جودة وكفاءة لأعمل بالبرمجة

خطأ

إمتلاكك لحاسب بمواصفات عالية تعتبر ميزة إلى حد ما من البرمجة . بيئة التطوير التى تستخدمها ربما ستعمل سريعاً وتترجم الأكواد بأكثر سرعة . ولكن لهذه المنافع الصغيرة هناك أيضاً عقبات .

العقبة الرئيسية هى إنك إذا أردت أن تصمم برنامج على جهاز عالى فسيجب عليك ان تختبره على أجهزة أقل لتتأكد أنه سيعمل عليهم .

Myth 6 — Programming Is Addictive!

الأسطورة السادسة — البرمجة إدمان

صحيحة !

بالتأكيد هى كذلك ! هناك إنتعاش ضخم يمكن ان تمتلكه من جعلك الحاسب يفعل ماتريد . والإحساس بالقوة الذى تعطيك أياه حتى المبتدئين يشعرون بالإندهاش .

Myth 7 — Programming Languages Change All the Time

لغات البرمجة كثيراً ما تتغير .

خطأ !

كأى شئ حسن فى عالم الحاسب , تمر لغات البرمجة فى عجلة التطور والتغير , ومع ذلك , إذا إخترت أن تتعلم لغة مألوفة (Visual basic , C++ أو حتى JavaScript أو VBScript) فإن التغيرات التى ستجربها ستكون صغيرة , وفى العادة , تكون التغيرات الرئيسية فى التطبيقات مثال الواجهة IDE أو المترجم (التطبيق الذى يحول مصدر الكود داخل التطبيق) .

Myth 8 — Once You've Learned One Programming Language, Learning Others Is Easier

الإسطورة الثامنة — ما إن تتعلم لغة برمجة واحدة , تعلم اللغات الأخرى سيكون أسهل

صحيحة !

ما إن تفهم أساسيات لغة واحدة , تعلم اللغات الأخرى لن يكون صعب , وهذا بسبب أن المفاهيم العامة منقولة من لغة إلى أخرى . بعض اللغات متشابهة جداً (وأحياناً الاختلافات الصغيرة فى التشابهة يمكن حقيقتاً أن تجعل الأشياء أكثر صعوبة) , بينما يمكن أن تكون اللغات الأخرى مختلفين تماماً .

مع ذلك , يمكنك أن تعتمد على أن المتغير هو المتغير , والمعاملات الرياضية ستكون فى العام متشابهة من لغة برمجة لأخرى , كما الحال فى العديد من مفاهيم البرمجة الأخرى , لذلك فى حين قد يكون عليك أن تتعلم كيفية فعل شئ ما فى لغة برمجة جديدة , ستكون على الأقل على دراية بماهية هذا الشئ .

Summary

الملخص

فى هذا الفصل , عرضنا بعضاً من معظم الأسباب القاطعة من أجل تعلم البرمجة ولماذا يكون هذا شئ جيد , هناك أيضاً الكثير من الأسباب الجيدة لتعلم البرمجة , فسواء اتيت إلى هذا الكتاب بسبب قد أدرجته داخل قائمة الأسباب أو سبب آخر , فإنه من الجيد أن يكون لك هنا أحد الإتجاهين !

فى الفصل القادم , سنلقى نظرة على كيفية تخزين الحاسبات وقرائة الكود .

3

How Computers “Read” Code

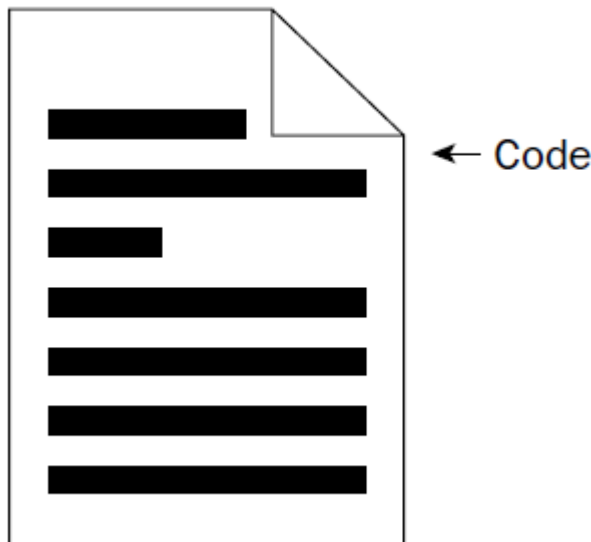
كيف تقرأ الحاسبات الكود

هذا الفصل يبحث في كيفية تعامل الحاسبات مع الكود الذى نعطيها إياه , إذا كان لديك فهم عن كيف يقرأ الكود بواسطة الحاسب , ستكتب افضل , و بأكثر إيجاز للكود الصحيح من البداية ! .

Reading Code

قراءة الكود

لمساعدتك لفهم كيف يقرأ الحاسب الكود , وتُقدِّمُ إليه , سنبدأ بالنظر إلى صفحة لكود عام , يظهر فى الشكل

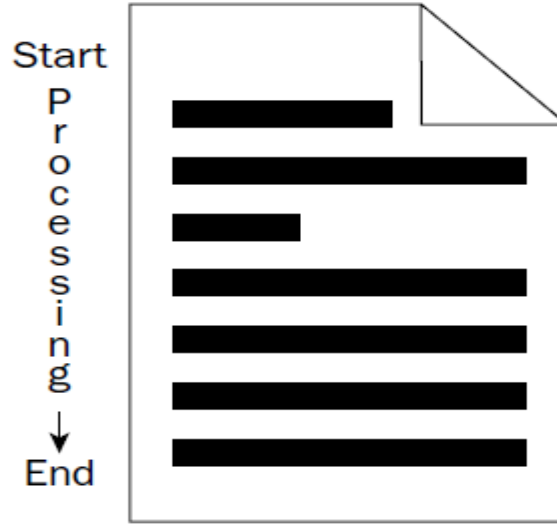


هذه صفحة لكود , لا تفكر فى انها جزء من الكود لتطبيق , ولكن فكر فى أنها كل الكود . ليس هناك كود حقيقى , مجرد اسطر فارغة تمثل الكود , ولكن ستظل تعطيك بقدر كافى الفكرة فى كيف تعمل الأشياء .

Top Down

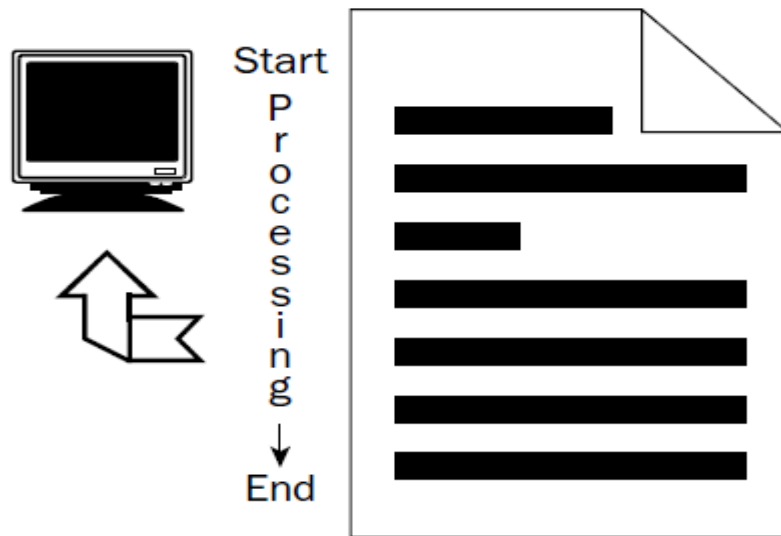
أعلى لأسفل

معظم الأشخاص يعتقدون ان الحاسب يقرأ الكود من أعلى لأسفل , يبدأ من المقدمة ويقرأ حتى يصل إلى النهاية , كما في الشكل :

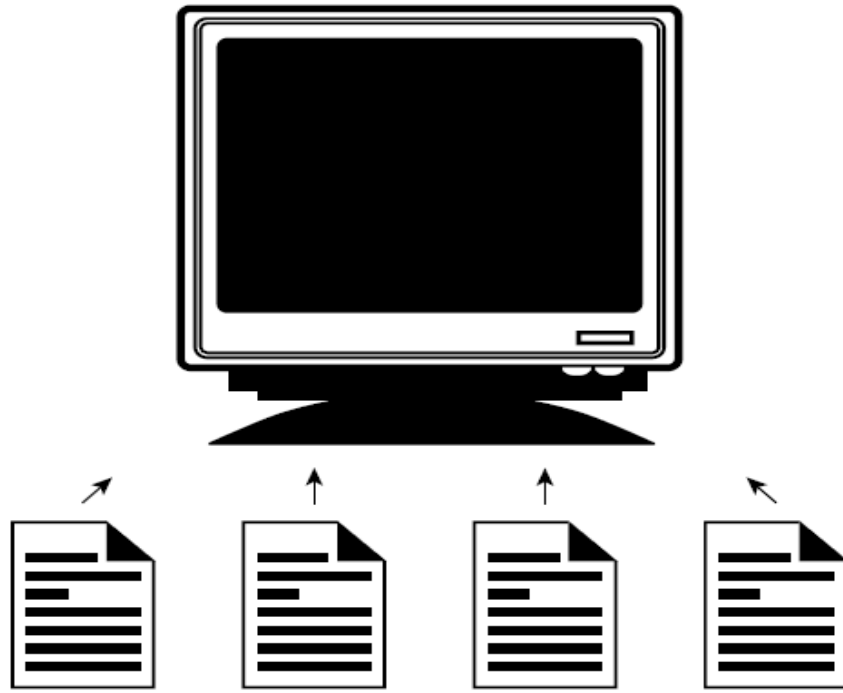


في هذا السيناريو , عندما يبدأ الكود في التنفيذ يبدأ الحاسب بقراءة الكود من الأعلى ويستمر إلى الأسفل مروراً بكل سطر في الكود حتى يصل إلى النهاية .

عملية "الأعلى - إلى الأسفل" أو "أعلى - أسفل" تستخدم لتبين كيف تجري الأمور , وأخذت في الاعتبار كلاً من السرعة و السهولة , ومع ذلك , تكون فقط سريعة وسهلة عندما يكون الكود صغير وغير معقد (على الأقل بمعايير هذه الأيام) . تكون المتاعب في فعل هذا هي أن الكود قد تمت معالجته وتم تحميله بالكامل في الذاكرة (انظر الشكل)



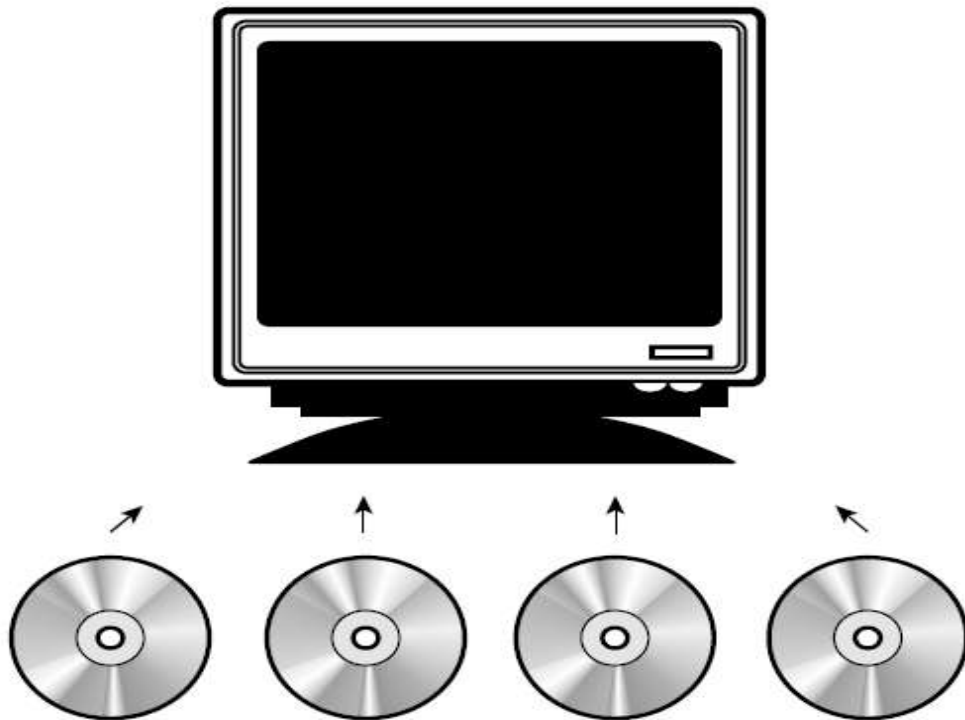
كلما زاد الكود المحمل في الذاكرة , كلما زادت حاجتك لمواجهة موارد النظام , (راجع الشكل)



More code = More resources!

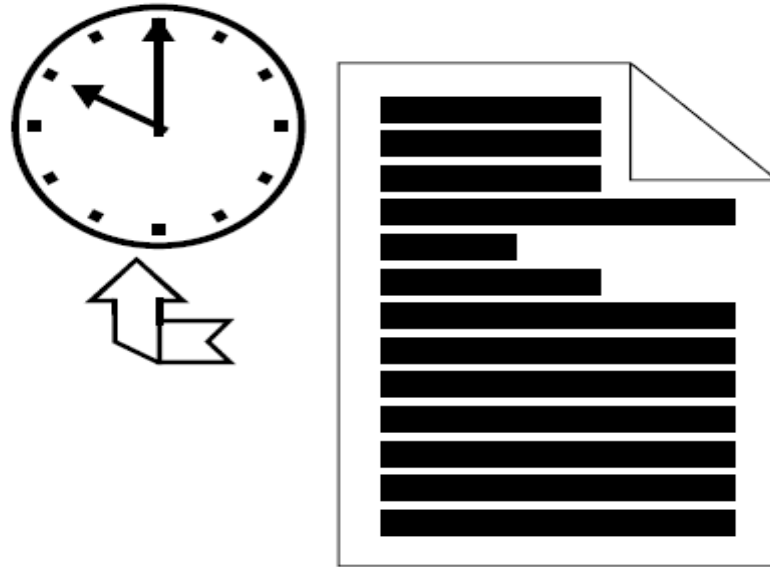
[يقصد هنا كثرة الكود في الذاكرة يقابلها إستهلاك أكثر في الموارد بالتالى سيؤثر سلباً على الأداء]

لذلك لكثرة البرامج التى تعمل بها , فأعظم متطلبات فى النظام من حيث الذاكرة والمعالج , كما فى الشكل



More programs = More resources!!!!

أيضاً , كبر حجم البرنامج الذى تقوم بتحميله , سيأخذ تحميله أطول وأطول فى كل مرة تريد أن تقوم بتشغيل هذا البرنامج . إنظر الشكل



More code = More processing time!

هناك الكثير من العقبات , أولها هي أنك يجب أن تكتب كود بالطريقة الصعبة لإحتمال إختباره متى تنتهيه .
إختبار اجزاء منفردة من الكود أصبح كابوس , فإذا تركت الإختبار حتى النهاية , يكون لديك الكثير من الكود
لتمر خلاله لتجد الشوائب , إنظر الشكل



Top down code is much harder to debug!

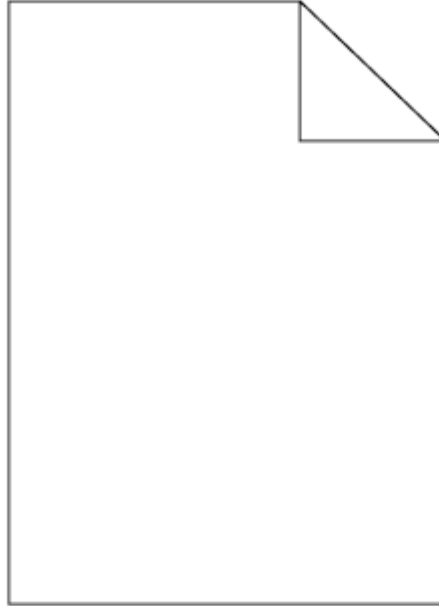
كما سترى فى الفصول اللاحقة , عملية أعلى-أسفل تظل تمتلك مكانها عندما تاتى إلى المشاريع الصغيرة ولكن
سرعان ما تصبح قابلة للتطبيق فى المشاريع الكبيرة أو الكود الأكثر تعقيداً . حتى إعطاء هذه الحقيقة هي ان
عملية الأعلى – إلى – أسفل كانت حل مبسط أكثر من غيرها , فلم تبقى فى نطاق العمل لوقت طويل .

Breaking Up Code

تجزئة الكود

بدلاً من جعل الكود أكثر تعقيداً بواسطة كتابة كود نعتقد ان الحاسب سيقراه ويقوم بتشغيله في مرة واحدة . فمن الجيد تقطيع الكود والسماح لأجزاء مختلفة من الكود لمعالجة أجزاء مختلفة من المشروع .

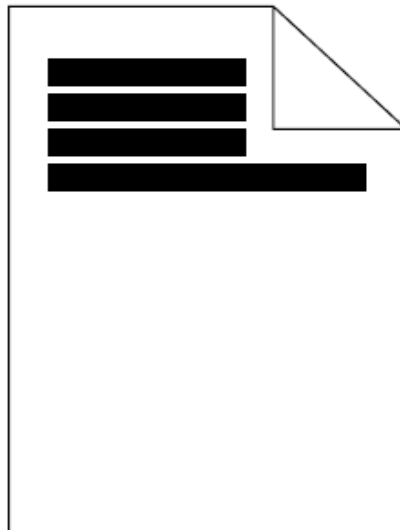
لنقل , على سبيل المثال , انك تعمل على برنامج بسيط يعمل على كم الضريبة المستحقة للشراء .



Tax calculator project - no code

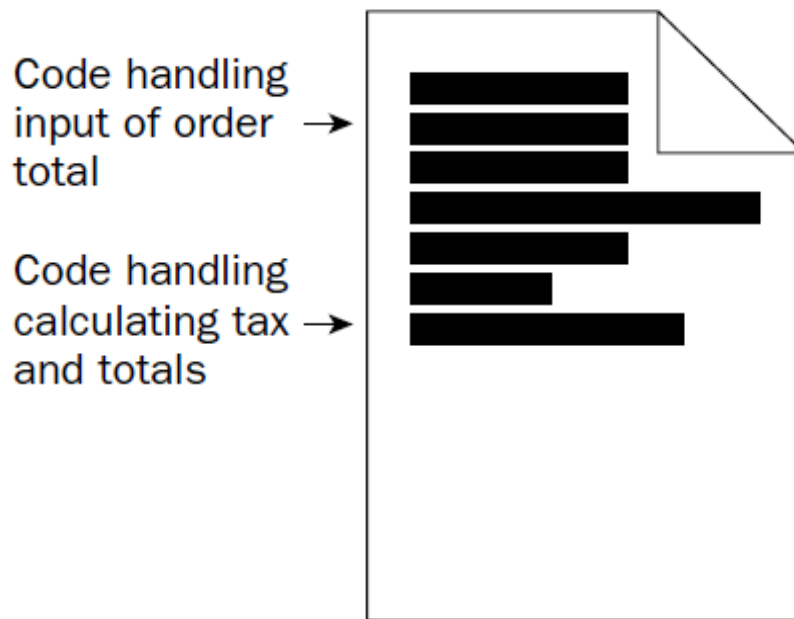
ربما قد ينتهي بك الحال مع موقف يكون لديك جزء واحد يعالج المدخلات لترتيب الإجمالي , كما في الشكل :

Code handling
input of order
total →



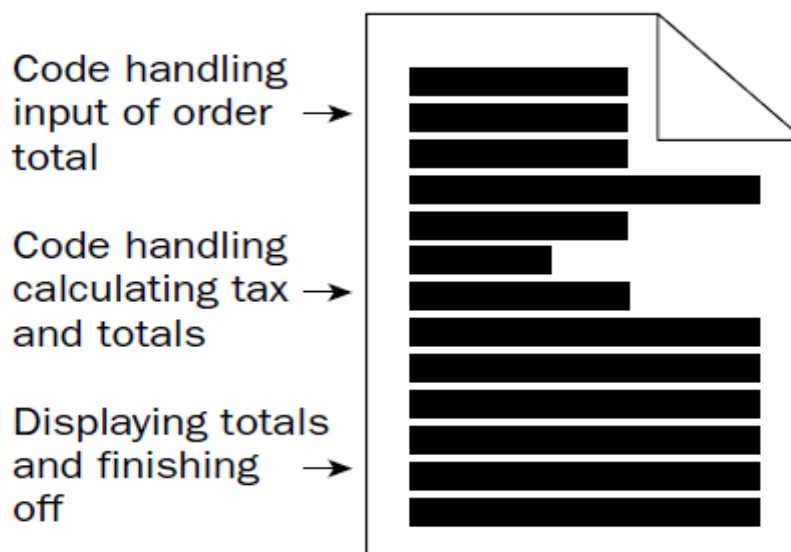
Tax calculator project

جزء واحد يعالج العمل على الضريبة بإضافتها على المجموع , كما يعرض فى الشكل



Tax calculator project

وأخيراً (إنظر الشكل التالى) قمنا بإضافة الكود لمعالجة عرض المجموع وتنفيذ أى توضيح نهائى .

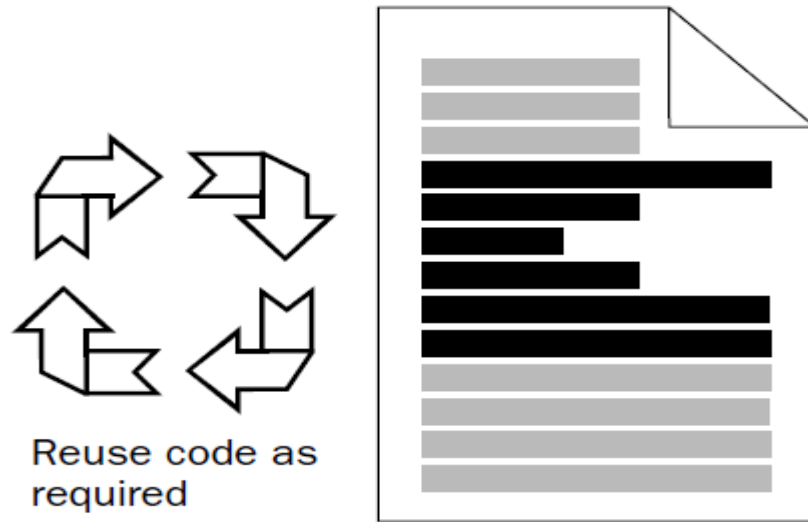


Tax calculator project

هناك مميزات ضخمة لكتابة الكود بهذه الطريقة :-

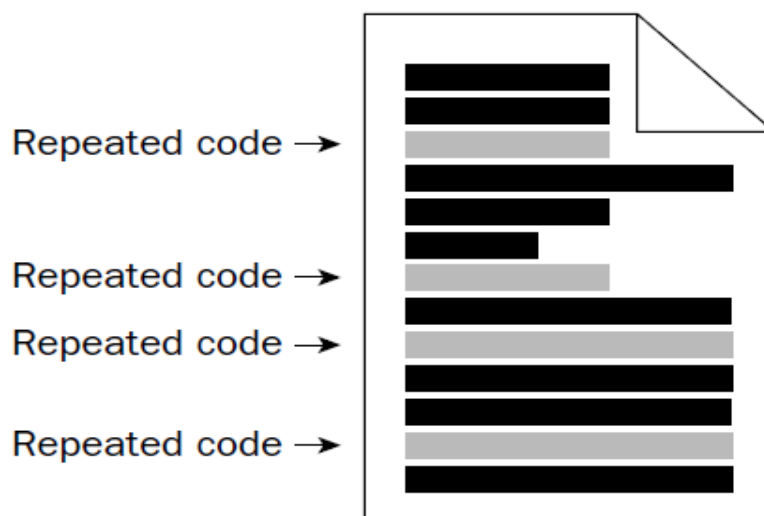
- تقطيع الكود إلى اجزاء صغيرة , لذا من السهل كتابته , وتستطيع ان تختبره فى وقت مبكر .
- تصحيح أى خطأ فى الكود به الكثير من التيسير لأنك تعرف أى جزء من الكود يعالج أى جانب من البرنامج .
- إمكانية إعادة استخدام أجزاء محددة من الكود فى مشاريع أخرى باكثر سهولة وستكون قائمة بذاتها .
- بناء مشروع كبير من حزم مترابطة صغيرة يمكنك من بناء مشروع على مر الوقت وإضافة مزايا اكثر كلما تحسنت مهاراتك .

تقطيع الكود فى حزم صغيرة يمكنك من فعل بعض الأشياء الأخرى الذكية , اولها إمكانية أن تكتب جزء من الكود يمكن أن يعمل ويعاد تشغيله مرات على حسب الطلب . مثال بسيط لهذا أن جزء الكود ربما يستخدم لمعالجة إضافة رقمين , فى طريقة أعلى- اسفل البرمجة كنت بحاجة لتوقع كم عدد الأرقام التى تريد أن تضيفهم معاً وتكرر الكود لكل خطوة . إنظر الشكل التالى

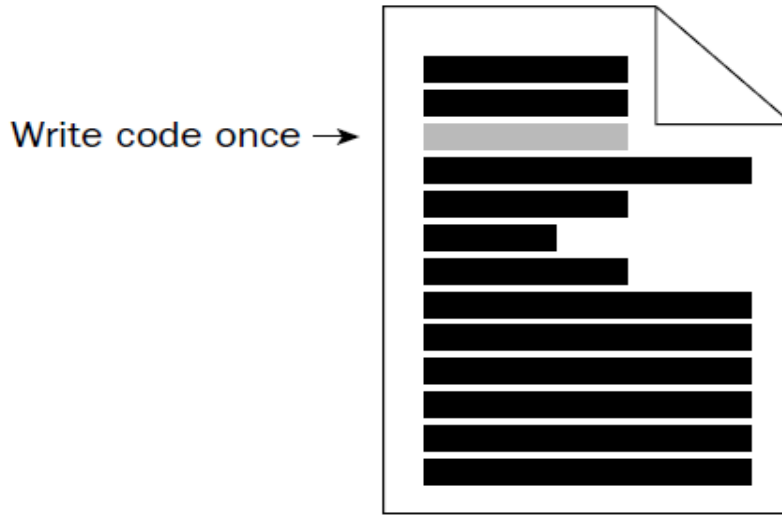


بهذه الطريقة , يمكن ان تكتب قطعة من الكود تعالج إضافة الأرقام ومجرد تكراره حسب الطلب . مما يجعل الكود الشامل أصغر وأكثر إستجابة وأيسر لتتم قرائته ومتابعته (الكود الأكبر , الأكثر بطئ , بسبب كثرة إستهلاكه من الذاكرة ووقت المعالج , وكأى شخص يمتلك حاسب شخصى لفتره سيعرف الإثنين بعرض مختصر)

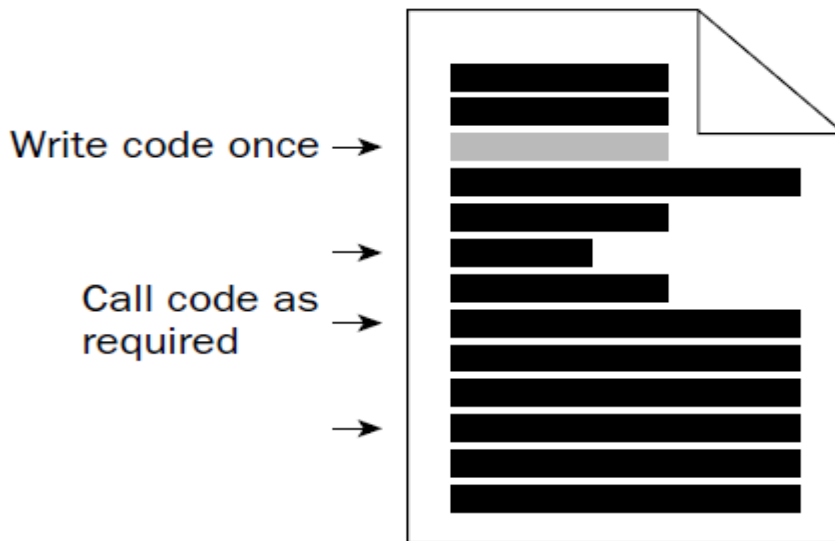
يجعل هذا أيضا الكود أكثر مرونة , السبب فى هذا هو الحال قد لا يكون واضحاً على الفور , ولكن يأخذ فى الإعتبار الكود الذى كتبته الذى يقوم بالعملية الحسابية للأرقام . دعنا نقول أنك تريد أن تحدث بعض أشياء بعد تمام كل عملية حسابية (ربما تريد تغيير المجموع فى الشاشة) .بعدها طريقة واحدة لترمز لهذا هى إضافة الكود الذى ينفذ ويُحدّث فى كل نقطة فى البرنامج كلما احتجت لذلك .إنظر الشكل



هذا إصراف , ليس فقط لوقتك (عليك أن تكتبه فيه ثم إمكانية تصحيح الكود عدة مرات بسبب وقوعك في الخطأ) ولكن أيضاً لموارد النظام . وهناك طريقة حسنة لكتابة كود هو عمل التالي , عندما تكتب الكود في سؤال واحد , كما يعرض الشكل



ما ان تقوم بتشغيل الكود , بعدها تقوم بتشغيل الكود من اى مكان حسب المتطلبات وهذا يسمى "إستدعاء" الكود . إنظر الشكل



تنفيذ الكود حسب الطلب هي خطوة قوية جداً عندما تكتب كود , وسترى هذا على أرض الواقع كثيراً بمجرد أن تبدأ بكتابة كود حقيقي . لماذا ؟ لأنه يعني أن لأى مهمة معينة تريد ان تكتب الكود لها مرة واحدة وتشير إليها عند الحاجة فيما بعد . فلا حاجة لتكرار نفسك . ولكن هناك منافع أخرى أيضاً . أولها في إنه اذا وجدت نفسك تريد ان تغير في الكود وتحتاج فقط ان تغيره مرة واحدة , مما سيحفظ الكثير من الوقت على المدى البعيد . في حالة أنك تملك كود يحدث المجموع في الشاشة وتريد أن تغير برنامجك بطريقة تشمل تغيير الكود لتحديث المجموع . الآن , أيهما سيكون أسهل , البحث خلال الكود وتحديث مساحة واحدة من الكود أو أن تبحث كل الكود وتجد كل نسخة عند تنفيذ التحديث وتغيير كل واحدة على حده ؟ أعتقد اننا يمكن أن نوافق ان تغيير الكود مرة ليس فقط الطريقة الأسهل ولكن أيضاً الأكثر قبولاً عقلاً .

عندما نتحدث كثيراً عن تقطيع الكود وكم هذا مهم لكتابة كود جيد , الآن دعني أقدم لك بعض المصطلحات الهامة المرتبطة بالكود وتركيب الكود .

يقطع الكود في تركيبتين إثنين :-

- **The Statement**
- **The function Or Procedure**

The Statement

The statement هي ما يعادل برمجة الجملة , وهي السطر الواحد لتعليمات الحاسب ويمكن ان يفهم بواسطة النظام . بشكل مبسط , المصطلحات الغير برمجية هي جملة سيتكون منها شئ ما كالذى يظهر في الشكل :-

```
Print current document  
  
Change text to red and make it bold  
  
Save current document
```

[يقصد بالشكل بالأعلى أن الكود سيكون بالشكل الذى ترى في اللغة العادية أى في الكلام العادى بعيداً عن البرمجة وقواعدها]

الشكل التالى هو مثال للجملة في C++

C++ code snippet

```
#include <iostream.h>  
int main()  
{  
A statement → cout << "Hello, World!\n";  
}
```

والشكل التالى يعرض مثال في لغة Visual Basic

Visual Basic code snippet

```
Private Sub Form_Load()  
A statement → msgbox "Hello, World!"  
End Sub
```

يمكن أن يكون لديك جملة واحدة كما يظهر بالأمثلة , أو أكثر كما يعرض فى الشكل التالى

C++ code snippet

```
#include <iostream.h>
int main()
{
    cout << "Hello, World!\n";
    cout << "Another statement here\n";
    cout << "And another!\n";
    cout << "Statements are what ... \n";
    cout << "... drive programs!\n";
}
```

كل الجمل هي تعليمات [أوامر] لبعض الوصف ويفعل شئ واحد من إثنين :

- فعل شئ ما .
- الإنتظار .

لديك جمل , ولكن لتعطى أعظم تحكم , يمكنك تنظيم الجمل فى دوال أو إجراءات (Functions Or Procedure)

Functions/Procedures

الدوال والإجراءات

إذا كانت الـ Statement هي جمل البرمجة , إذا فالدوال والإجراءات هي الفقرات .

مصطلح آخر ربما تسمعه يستخدم للدوال أو الإجراءات هو " Routine " ليست شائعة الاستخدام ولكن تعطى فى العام فى برمجة Pascal بعيداً

الشكل التالى يظهر دالة فى C++

C++ code snippet

```
#include <iostream.h>
int main()
{
    cout << "Hello, World!\n";
}
```

A function {

والشكل التالى يعرض دالة فى Visual Basic

Visual Basic code snippet

```
Private Sub Form_Load()
    msgbox "Hello, World!"
End Sub
```

A function {

في لغات معينه (مثال Pascal) الدالة والأجراء يمكن أن تكون لها معاني كثيرة . وللأرتياح , سأستخدم كلمة دالة "function" فقط .

الدالة هي الطريقة النهائية لفرز الكود إلى قطع قد أعيد إستخدامها . ربما في كثير من الأحيان عندما تحتاج إلى جملة واحدة ولكن في العام سيجعلك الكود الخاص بك تستخدم الكثير من الجمل وحينها سيكون عليك أن تجمعهم في دوال .

The Sentences and Paragraphs of Programming

الجمل والفقرات من البرمجة

قبل أن نكمل , دعنا نأخذ القليل من الدقائق القصيرة لنلخص ما قد قيل بالفعل , أعتقد انه من الهام فعل هذا هنا لأننا قمنا بتغطية بعض الإهتمامات والمصطلحات الهامة التي ستكون مفاتيح طريقتك البرمجية مؤخراً في هذا الكتاب . فإذا تحدثنا كلنا لغة واحدة (لا بقصد التورية) ونمضي قدماً , ستكون الأشياء أسهل وستكون هناك فرصة أقل ليكون هناك خلط في المفاهيم .

Lines of Code

أسطر الكود [سطور الكود]

سطور الكود التي تكتبها تسمى جمل statement . وهذه هي عادة بناء الكتل الصغير من الكود . كما يعرض في الشكل

Visual Basic code

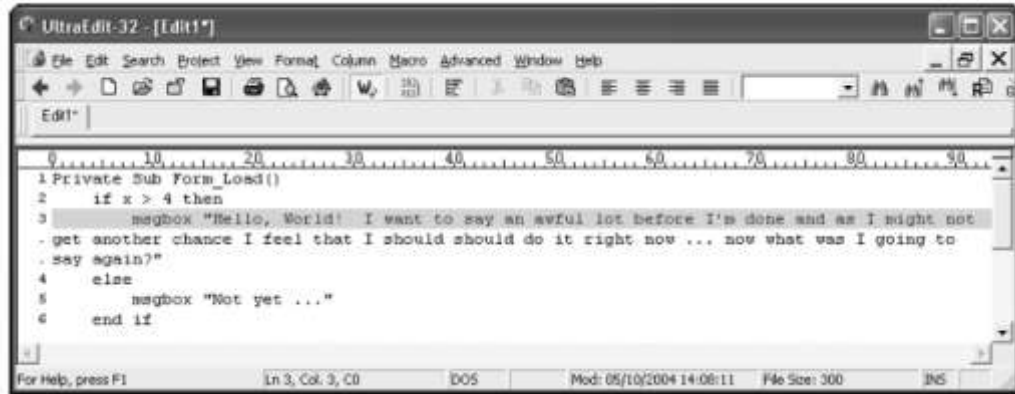
```
Private Sub Form_Load()  
→   if x > 4 then  
→       msgbox "Hello, World!"  
Statements →   else  
→       msgbox "Not yet ..."  
→   end if  
End Sub
```

طول السطر ليس له صلة ولا يؤثر , فكل من السطور القصير والطويل بكونها تعادل جمل . كما يعرض في الشكل .

Visual Basic code

```
Private Sub Form_Load()  
if x > 4 then  
    msgbox "Hello, World! I want to say an awful lot before I'm done and as I might not get another ... ummmmmmm"  
else  
    msgbox "Not yet ..."  
end if  
End Sub
```

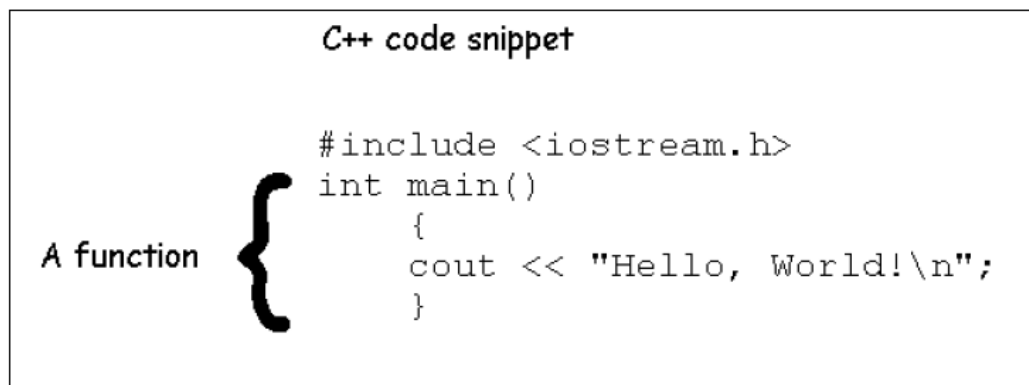
لاحظ أحياناً عندما تكتب جمل الكود التي ربما (تعتمد على طول السطر ونوعه ومحرر النصوص المستخدم) تتحرف إلى سطرين أو أكثر , ولكن ضع في إعتبارك انها مازالت فقط جملة واحدة . كما يعرض بالشكل التالي



Paragraphs of Code

فقرات الكود

جمل الكود أو statement تنظم في فقرات تسمى دوال functions . وما تحتوية الدوال يعتمد على اللغة المستخدمه وماتريد أن ينفذه الكود , والكثير من لغات البرمجة تضع القيود لما يمكن أن تحتويه إضافة إلى التدقيق الإملائي [طريقة تركيب الجمل] الذى تحتاجه لكتابة الكود بشكل صحيح . كما يعرض فى الشكل :



الـ syntax فى الكود يعادل ما يسمى Grammar فى اللغة

كم عدد الجمل وكم عدد الدوال التى توجد فى مشروع يعتمد هذا على نوع المشروع الذى تتعده وكم يكون تعقيده — عادةً المشاريع الكبيرة (هكذا الكثير منهم يكون كذلك والمزيد من المزايا التى يمتلكها التطبيق) سنجد انها ستحتوى على المزيد من الجمل والدوال , لذلك بإلقاء نظرة بسيطة على كود المشروع يمكنك ان تخبرنا تقريباً , كم سيكون كبر حجم التطبيق النهائى .

المبرمجون الجيدون يحاولون ان يحفظوا الكود النهائى المخرج أكثر إيجاز على قدر الإمكان. ومع ذلك فى المراحل المبكرة من البرمجة ربما كان المبرمجون أكثر تحرراً مع الكود ولا يزعجوا لجعل الكود موجزا كلما أمكن , تضحيةً "بالحجم" للكود للقراءة .

Data Storage

تخزين البيانات

مؤخراً في هذا الفصل , دعنا نلقى نظرة على كيف يخزن الكود قبل أن يعمل .

أحد أبسط الطرق في وضع الكود داخل النظام الذى سيعيد به ويكتب فيه عند الحاجة . لن هذا يتطلب تخزينه قبل الإستخدام لكنه محمل بالسلبات :

- الوقت المأخوذ لعملية الإدخال .
- المشاكل الناتجة من الأخطاء المختزلة فى الكود .
- مشاكل توزيع الكود (يجب ان يوزع الكود فى نموذج الكتاب , عمل نسخ منه وسهولة القرصنة)
- مشاكل أخرى لا تعد ولا تحصى وقضايا أكثر تعدداً لتذكر هنا .

تذكر أن الكود يبدأ الحياة عند شخص ما يجلس فى مقدمة لوحة المفاتيح ليكتبه . هناك الكثيرون ممن يعتقدون أن هذا هو سبب لوجود الكثير من الشوائب والمشاكل فى البرامج الحديثة . وبسبب العامل البشرى . ربما تكون هذه القضية , ولكن ليس هناك طريقة أخرى لتكتب كود يبدو لجعل نفسه معروف لنا (على الأقل مازلت أكتب الكود بالطريقة القديمة ولم أستخدم نظام Star Trek الرائع) .

لجعل التطبيقات بالفعل يعاد إستخدامها , فحتاج إلى طريقة تخزن بها الكود ليكون متاح عند الحاجة إليه . على مر السنين , كان هناك الكثير من الحلول لمثل هذه المشاكل :

- ☐ Paper punch cards
- ☐ Magnetic tape reels
- ☐ Magnetic cassettes
- ☐ Floppy disks (51/4 inch)
- ☐ Floppy disks (31/2 inch)
- ☐ Hard drives
- ☐ Networked systems
- ☐ CDs
- ☐ DVD
- ☐ Flash drives
- ☐ Internet distribution

كلاً من هذه الأنظمة يمكنك من تخزين الكود بطريقة يمكن الوصول إليه وتشغيله شئ واحد ستلاحظه هو انه , كلما يتحرك الوقت , تزداد المساحة التخزينية . فى اخر 10 إلى 20 سنة رأينا مساحات Hard drive تنمو من عشرات الميجا بايت إلى مئات الجيجا بايت — هذا يعنى أن أجهزة القرص الصلب (Hard Drive) الحديثة بها عشرات الالاف من المساحة أكثر من التى كانت فى العقود السابقة .

كلاً من هذه الأنظمة يمكنك من التخزين والإسترجاع لكود برمجة التطبيقات . تخزين تعنى انك فقط يجب ان تكتب الكود أو (اكثر دقة) تنشأ الكود مرة واحدة , وتخزينه للإستعمال بالمستقبل . ومع ذلك , الإسترجاع يعادله فى الأهمية , وهذه هى البرمجة الحديثة , وفى الحقيقة , جميع عمليات الحاسب الحديثة , تدور حول — القدرة على التخزين , والوصول , وتعديل البيانات .

كما رأينا الزيادة فى مساحات التخزين . رأينا التزايد الضخم فى الملائمة حول التخزين والإسترجاع للبيانات — تحميل البرامج المخزنة على القرص الصلب Hard Drive تهزم العبث مع البطاقات الورقية أو حتى الـ floppy disks.

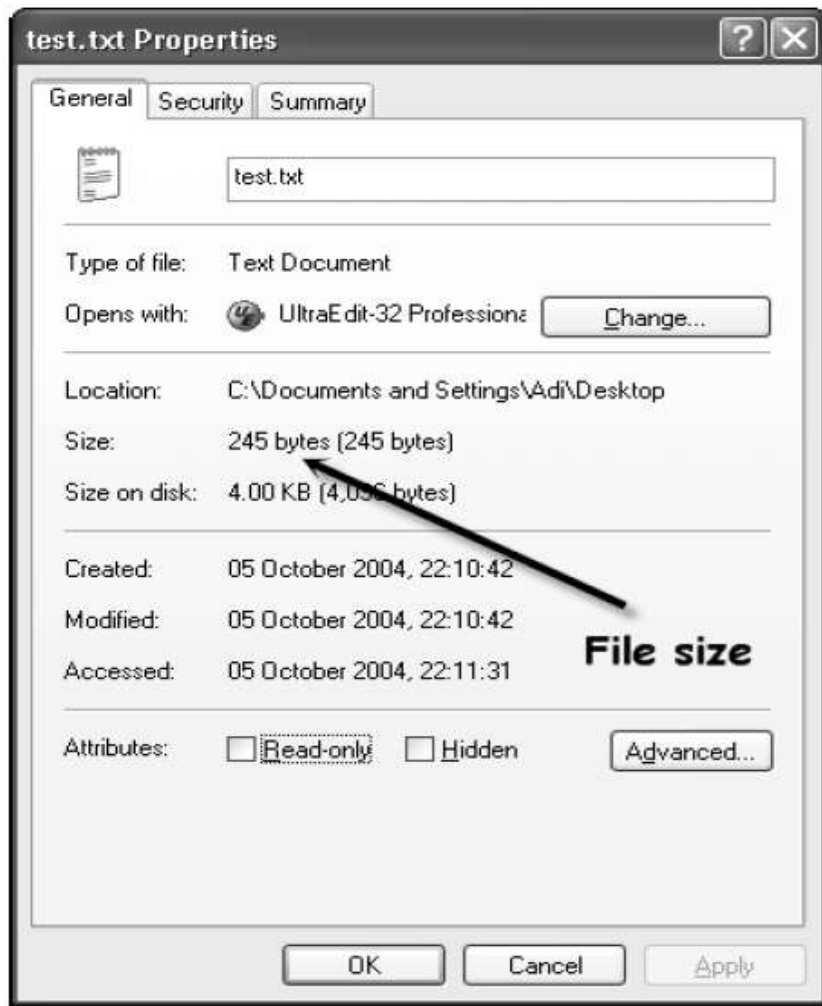
Data

البيانات

ولكن كيف تخزن البيانات فى هذه الوسائل ؟ تخزن فى أبسط ما يمكن — Binary [النظام الثنائى] . الذى يقرأ بمسلسل من رقمى الصفر والواحد . فإذا قمت بتخزين هذه الفقرة فى ملف نصى و سيخزن على القرص الصلب الخاص بى سيكون على هذا الشكل :

```
01000010011101010111010000100000011010000110111101110111001000
00011010010111001100100000011101000110100001100101001000000110
010001100001011101000110000100100000011001101110100011011101
11001001100101011001000010000001101110110110001000000110100
01101000011001010111001101100101001000000110110101100101011001
00011010010110000100111111001000000010000001001001011101000010
0000011010010111001100100000011001101110100011011110111001001
10010101100100001000000110100101101110001000000111010001101000
01100101001000000111001101101001011011010111000001101100011001
01011100110111010000100000011001100110111101110010011011010010
00000111000001101111011100110111001101101001011000100110110001
10010100100000100101100010000001100010011010010110111001100001
01110010011110010010111000100000001000000101011101101000011000
01011101000010000001101001011100110010000001100010011001010110
10010110111001100111001000000111001001100101011000010110010000
10000001101001011100110010000001100001001000000111001101100101
01110010011010010110010101110011001000000110111101100110001000
00011110100110010101110010011011110111001100100000011000010110
11100110010000100000011011110110111001100101011100110010111000
10000000100000010010010110011000100000010010010010000001110111
01100001011100110010000001110100011011110010000001110011011101
00011011110111001001100101001000000111010001101000011010010111
00110010000001110000011000010111001001100001011001110111001001
10000101110000011010000010000001101001011011100010000001100001
00100000011101000110010101111000011101000010000001100110011010
01011011000110010100101100001000000111011101101000011000010111
01000010000001110111011011110111010101101100011001000010000001
10001001100101001000000111001101110100011011110111001001100101
01100100001000000110111101101110001000000110110101111001001000
00011010000110000101110010011001000010000001100100011100100110
10010111011001100101001000000111011101101111011101010110110001
10010000100000011000100110010100111010
```

ينشأ عن هذا ملف حجمه 245 Bytes — 1 بايت لكل حرف فى الملف , كما فى الشكل



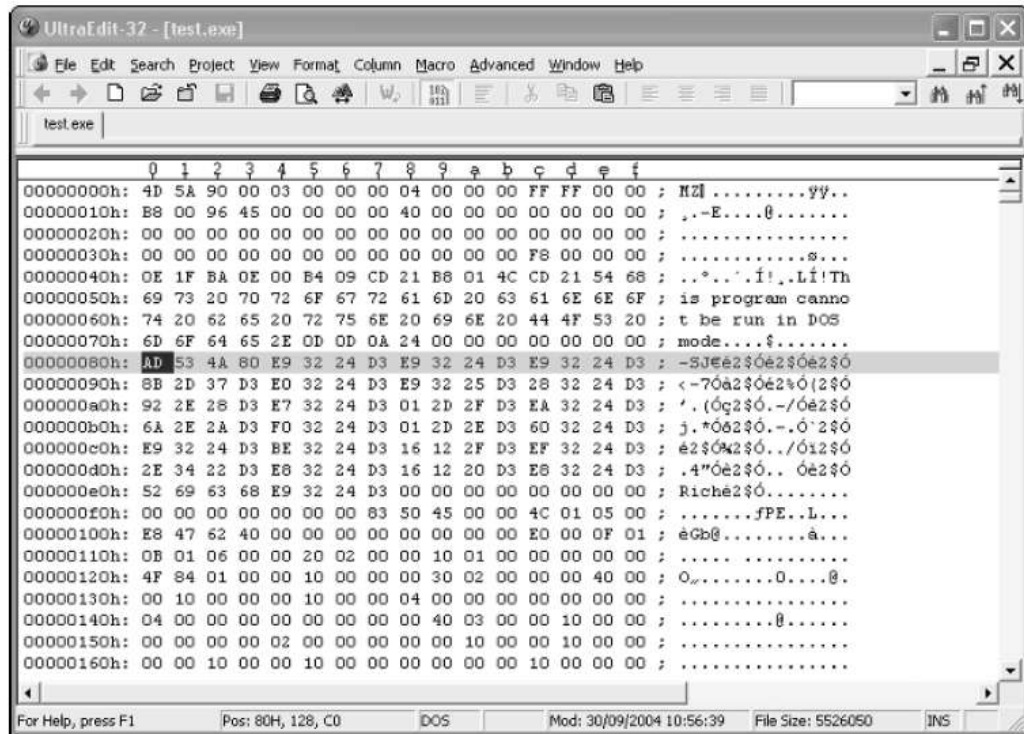
نفس الشيء البرامج — ولكن الكود يمكن أن يكون احد نموذجين .

- Plain Text [نص عادي]
- Compiled [مترجم]

النص العادي يقرأ بواسطة الإنسان — أنت هنا تقرأ نص عادي , بعض الكود الذي ستمر به سيخزن ويعمل في نموذج النص العادي — JavaScript ,VBScript أمثلة جيدة لذلك .

الكود المترجم مختلف . فيكون كود قد تم تشغيله من قبل في برنامج يسمى Compiler [مترجم] . يعالج الـ Compiler كود النص العادي ويغيره إلى شيء ما يسمى "Object code" . الشكل الذي سيأتي يعرض مثال لكود مترجم محمل في محرر نصوص .

Object Code هو Machine Code [كود الآلة] . وهو ما يقرأه النظام . معالج الحاسب يقرأه ويعالج الأرقام المحددة من الصفر والواحد في وقت واحد (على سبيل المثال . ربما يكون قد صمم ليقرأ 32 رقم ثنائي , أو Bits في نفس الوقت)
ولأن النظام قد صمم لمعرفة عدد bits (وأيضاً أي Bits) ويخبره بماذا يفعل . يمكنه البحث عن التسلسل الصحيح من Bits ويؤدي العملية التالية . ثم يقرأ التعليمات التالية [الأوامر] , وهكذا .
وهناك مزايا و عيوب لكليهما



الميزة الرئيسية للكود المخزن كنص عادي هي سهولة القراءة , والتغيرات التي يمكن أن تحدث له بسهولة وسرعة . وليس هناك أدوات خاصة مطلوبة لذلك . ومع ذلك , سهولة قراءة الكود , وجعل التغيرات كذلك هي ضعف بسبب أنها تسمح للأخرين ان ينسخوا الكود ويقومون بالتعديل فيه , و أكثر من ذلك إدعاء ملكيته.

الكود المترجم أسرع من كود النص العادي ويقدم الحماية الفعلية لمصدر الكود من النسخ أو التعديل . لايمكنك ان تنظر إلى الكود المترجم وتسترد مصدر الكود بأي طريقة فيقدم هذا صفقة عظيمة من الحماية من العيون المتطفلة .

ومع ذلك . هناك أيضاً ضعف على هذا النحو , لذلك تحتاج أدوات خاصة لترجم وتجميعه للتحويل بطريقة واحدة . فتحتاج إلى حفظ نسخة من مصدر الكود في حالة إحتياجك لعمل تغيير في وقت لاحق لهذا الكود .في الحقيقة , ستحتاج إلى الإحتفاظ بنسخ عديدة لمصدر الكود . ومن الناحية المثالية , ستحفظ نسخة منه في كل وقت تجرى عليه بعض التغييرات — هذه الطريقة , إذا أجريت ببعض التغييرات تسبب مشكلة , وسيكون عليك أن ترجع إلى النسخة الأصلية من إصدار العمل . هذا هو السيناريو المثالي ولكن يكون في الغالب ليس عملياً أن تحفظ كل هذه النسخ .

Summary

الملخص

في هذا الفصل , تكلمنا عن كود البرمجة , ماذا يكون وكيف يخزن , ربما ليس حيوياً أن تفهم كل شئ عن كيف يخزن الكود في النظام وإسترجاعه مؤخراً . وفي أى تنسيق يكون الكود , فهم التنسيقات المختلفة المتاحة يساعدك لصنع إختيار تعليمي لما يناسبك .

هذا الفصل أيضاً أعطاك الكثير من المصطلحات ستقابلك عند قراءة كتب البرمجة , الحديث إلى المبرمجين , أو عرض المعلومات على الإنترنت .

4

From Concepts to Code — The Language of Code

من المفاهيم إلى كتابة كود — لغة الكود

في هذا الفصل سنقوم بفحص العديد من عناصر البرمجة التي تسمح لك بأخذ مفاهيم البرمجة وتحويلها إلى كود . لغة البرمجة المثالية ستكون واحدة عندما تستخدم اللغة الإنجليزية (أو اللغة التي أنت بها أكثر دراية) لإعطاء الحاسب تعليمات عادية يمكنه تتبعها . كمثال هذا :

```
Print this document.  
Make that item green.  
Add the last two numbers together.
```

هكذا ستكون , نظرياً , اللغة الأسهل للإستخدام لقرئها من لغة البشر , وستكون لغة عالية المستوى (اللغة الأعلى هي القريبة من اللغة الحقيقية) . ولكن هذا النوع من اللغة غير واقعي بسبب غموض تحدث وكتابة اللغة والتعقيد في الدماغ ضرورياً للتعامل معها بفهم . على سبيل المثال , الجمل الثلاثة التالية هم نفس الشيء — كيف يمكنك ان تجعل الحاسب يميز الثلاثة ؟

```
Print this document.  
Print the current document.  
Send this document to the printer.
```

هناك الكثير من الاحتمالات , كلاً منها سيفهما ولكن الحاسب لن يفعل .

الفهم المعقد هو هدف نهاية المطاف , بالطبع . لاحظ انه في *sci-fi* (*Star Trek* , *Star Wars*) والكثير من العروض) تفاعل البشر مع الحاسبات واجهزة الإنسان الآلى بإستخدام اللغة المنطوقة ؟ كونك قادراً على ذلك الان فهو جميل !

بما أننا لا نستطيع إستخدام لغتنا العادية لتتفاعل مع الحاسبات , فعلياً أن نستخدم لغات خاصة . في هذا الفصل , فإننا لم نبحث في لغات البرمجة حتى الآن , وبدلاً من ذلك سنبحث عن العناصر المكونة لمثل هذه اللغات . وبداننا بالبحث في أبسط نظام يستخدمه الحاسب للإتصال — Binary

Binary

النظام الثنائي [المزدوج]

هو نظام رقمي , يختلف عن نظام الأعداد العشرية العادية (العشرة الأساسية) الذي إعتدنا عليها ولكن بدلاً من أن يكون لديه عشرة أرقام (0,1,2,3,4,5,6,7,8,9) فلديه اثنين فقط (0 و 1) .
إسم آخر للنظام الثنائي هو Base 2 . إعطاء الإسم لهذه الأرقام هو bits .

معظم الناس يجدون الدخول في نظام Binary حقيقياً مربك لأنه يستخدم زوجين من الأرقام , فهي على خلاف ذلك مألوفة لهم بطريقة مألوفة . ولو كان بدلاً من استخدام 0 و 1 كانوا قد إستخدموا _ and _ أو _ and _ سيكون هناك أقل إرتباك .

Interpreting Binary

تفسير النظام الثنائي

أسهل طريقة لتكتشف كيف يعمل هذا النظام هي مقارنته بنظام الأعداد العشرية العادية .

Decimal	Binary
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
10	1010
11	1011

لاحظ كيف يحدث التزايد . منذ وجود 2 Bits فقط . يظهر الرقم ليصبح الأكبر (للحصول على المزيد من الأرقام) الأسرع , لكن لا تدع هذا يشعرك بالغباء , تقرأ الـ Binary من اليمين إلى اليسار , مع الزيادة في قيمة الـ bits من اليمين لليسار .

طريقة بسيطة للنظر إلى الإرقام الثنائية Binary أن ترجع إلى الخلف إلى طريقة تعلم الأطفال الأرقام حين كانوا صغاراً . إليك الرقم التالي

1010

ممثلة في النظام العشري , سيكون هذا الرقم كالتالي :

Thousands	Hundreds	Tens	Units
1	0	1	0

[الشكل السابق يمثل الرقم في صورة أحاد , عشرات , مئات وآلاف]

قراءةً يعطينا الرقم العشري كـ ألفاً وعشرة .

فى النظام الثنائى , تغيرت رؤس الأعمدة تماماً. وترتفع لكن ليس فى قوة 10 ولكن فى قوة 2 (هكذا , يتضاعف)

Eights	Fours	Twos	Units
1	0	1	0

لذلك يعطى قراءة هذا التالى :

1	x	8	=	8
0	x	4	=	0
1	x	2	=	2
0	x	1	=	0
			Decimal Total	10

يصل الإجمالى فى العمود جهة اليمين ليعطيك الإجابة التى تبحث عنها , عشرة .

هذا يعنى ان الرقم يكون 1 ثمانية و 1 إثنان يعطى فى النظام العشري الرقم 10 .

لقراءة رقم ثنائى أكبر , فقط تحتاج إلى شبكة أكبر , إليك هذا الرقم الثنائى التالى :

10010111

إنشأ شبكة (تذكر أن كل عمود يمثل مضاعفة العمود السابق)

One-Hundred-Twenty-Eights	Sixty-Fours	Thirty-Twos	Sixteens	Eights	Fours	Twos	Units
1	0	0	1	0	1	1	1

ولتعمل مع الرقم فى النظام العشري , إنشأ شبكة كما فعلنا بالسابق وضع وحدات النظام الثانى فى أسفل الجانب الأيسر.

1	x	128	=	128
0	x	64	=	0
0	x	32	=	0
1	x	16	=	16
0	x	8	=	0
1	x	4	=	4
1	x	2	=	2
1	x	1	=	1
			Decimal Total	151

لذلك فى النظام العشري , الرقم الثنائى 10010111 هو 151.

Large Numbers

الأرقام الكبيرة

أرأيت أرقام الأطفال , وأول شئ يفعلونه هو محاولة كتابة أكبر أرقام على قدر إستطاعتهم . بإستخدام 8bits , أكبر رقم ممكن هو 11111111 . سيعمل على هذا الشكل :

1	x	128	=	128
1	x	64	=	64
1	x	32	=	32
1	x	16	=	16
1	x	8	=	8
1	x	4	=	4
1	x	2	=	2
1	x	1	=	1
			Decimal Total	255

أضف المزيد من الـ bits , سيصبح الرقم أكبر .

1111111111111111

1	x	32768	=	32768
1	x	16384	=	16384
1	x	8192	=	8192
1	x	4096	=	4096
1	x	2048	=	2048
1	x	1024	=	1024
1	x	512	=	512
1	x	256	=	256
1	x	128	=	128
1	x	64	=	64
1	x	32	=	32
1	x	16	=	16
1	x	8	=	8
1	x	4	=	4
1	x	2	=	2
1	x	1	=	1
			Decimal Total	65535

Bit Grouping

تجمع الـ Bit

تجمع الـ bits لديه إسماء ليجعلهم أيسر للتذكر وللمناقشة , دعنا نأخذ جولة سريعة فى نظام التسمية .

Bit

هو الرقم الثنائى الواحد , هذا هو الـ bit :

0

كما هو الحال :

1

أقصى قيمة للـ bit فى النظام العشرى تتضح بـ 1 .

Nybble

تجمع أربعة من الـ Bits يسمى Nybble (أحياناً يسمى Nibble) , ها هو Nybble :
1010

أقصى قيمة للـ Nybble فى النظام العشرى هى 15

1111

1	x	8	=	8
1	x	4	=	4
1	x	2	=	2
1	x	1	=	1
Decimal Total				15

Byte

تجمع ثمانية من الـ bits يسمى Byte
10101010

أقصى قيمة للـ byte فى النظام العشرى 255

11111111

1	x	128	=	128
1	x	64	=	64
1	x	32	=	32
1	x	16	=	16
1	x	8	=	8
1	x	4	=	4
1	x	2	=	2
1	x	1	=	1
TOTAL				255

Halfword

تجمع 16 من الـ bits يسمى Halfword

1010101010101010

أقصى قيمة للـ Halfword فى النظام العشرى هى 65536

1111111111111111

1	x	32768	=	32768
1	x	16384	=	16384
1	x	8192	=	8192
1	x	4096	=	4096
1	x	2048	=	2048
1	x	1024	=	1024
1	x	512	=	512
1	x	256	=	256
1	x	128	=	128
1	x	64	=	64
1	x	32	=	32
1	x	16	=	16
1	x	8	=	8
1	x	4	=	4
1	x	2	=	2
1	x	1	=	1
			Decimal Total	65535

Word

تجمع 32 من الـ bits يسمى Word كما هو معروض هنا :

10101010101010101010101010101010

أقصى قيمة للـ Word فى النظام العشرى هى 4294967295

11111111111111111111111111111111

Binary Math

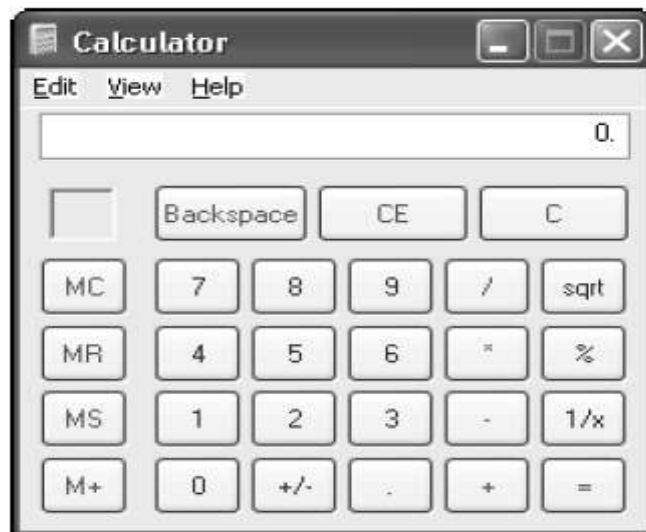
يمكنك العمل مع أرقام النظام الثنائي تشابهاً بطريقة العمل مع الأرقام العادية . مع ذلك , لنطاق الكتاب , الخوض في كتل نظرية رياضيات النظام الثنائي ليس ضرورياً . وسأريك طريقة أيسر للعمل مع النظام الثنائي باستخدام أداة لديها قوة عظيمة ومرونة — ولديها ميزة إضافية في كونها بلا مقابل . معروضة في الشكل التالي :



نعم , حاسبة نظام التشغيل Windows , فإذا استخدمت نظام التشغيل Windows إذاً فهي التطبيق المثالي للاستخدام , وإذا لم تكن مستخدم للـ Windows . إذا فنظام التشغيل الخاص بك لابد أن يأتي معه حاسبة مرنة بدرجة كافية للعمل مع النظام الثنائي Binary .

Using Windows Calculator

هاهنا إرشاد سريع لإستخدام حاسبة نظام التشغيل windows مع النظام الثنائي Binary . وإحتمالات إنك إذا لم تستخدم حاسبة الـ windows في الغالب , فبدلاً من ذلك يكفيك النظر إليها في هذا الشكل , ستكون في الشكل التالي :



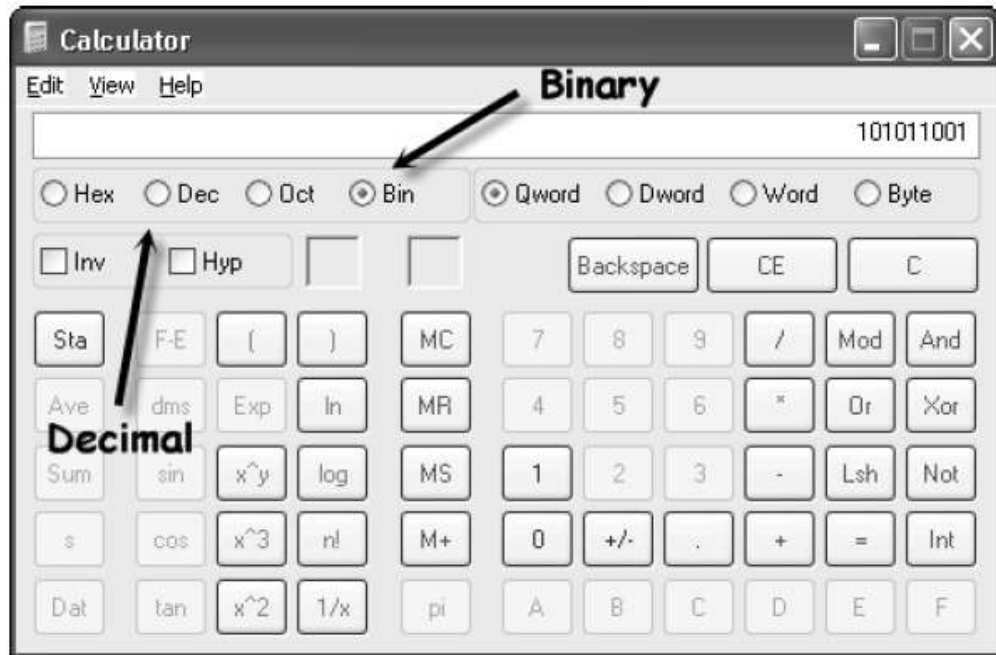
هذه هي حاسبة الـ windows بوضعها القياسي . ولكي تكون قادراً على جعلها مناسبة للعمل مع النظام الثنائي binary ستحتاج إلى تحويلها إلى الوضع العلمي scientific ولتحويلها للوضع العلمي إضغط على قائمة View ثم scientific , كما هو موضح في الشكل :



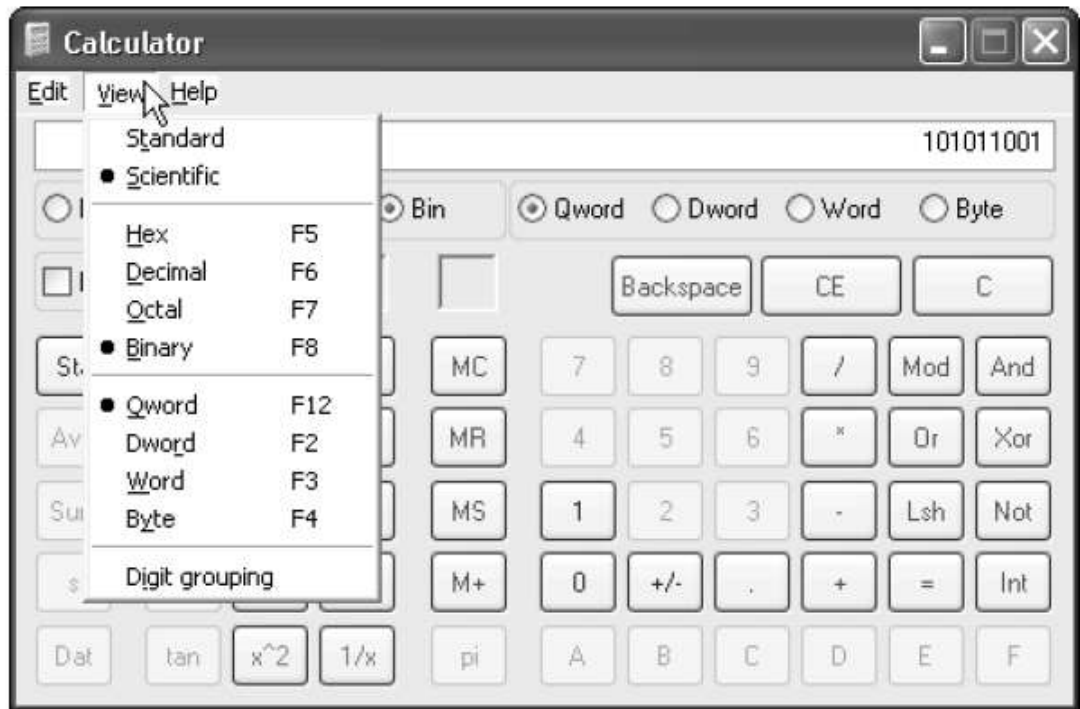
تغيرت الحاسبة للوضع العلمي وعرضت لك مجموعة كاملة من مفاتيح جديدة ومزايا .

إذا كنت تتشارك الحاسب مع شخص آخر غيرك يستخدم الألة الحاسبة لبعض المهام البسيطة , فتذكر أن تحوّل الآلة الحاسبة إلى الوضع القياسي بعدما تنتهي من العمل — فقط في حالة إعطائه / إعطائها شيء من الخوف !

الآن في الوضع العلمي , ستلاحظ الأدوات لتغيير تنسيقات الأرقام . النظام العشري Decimal ممثل في Dec . بينما النظام الثنائي binary ممثل في Bin , إشارة إلى الشكل :



يمكنك أيضاً أن تقوم باستخدام القائمة لتغيير التنسيق , إضغط view كما في الشكل :

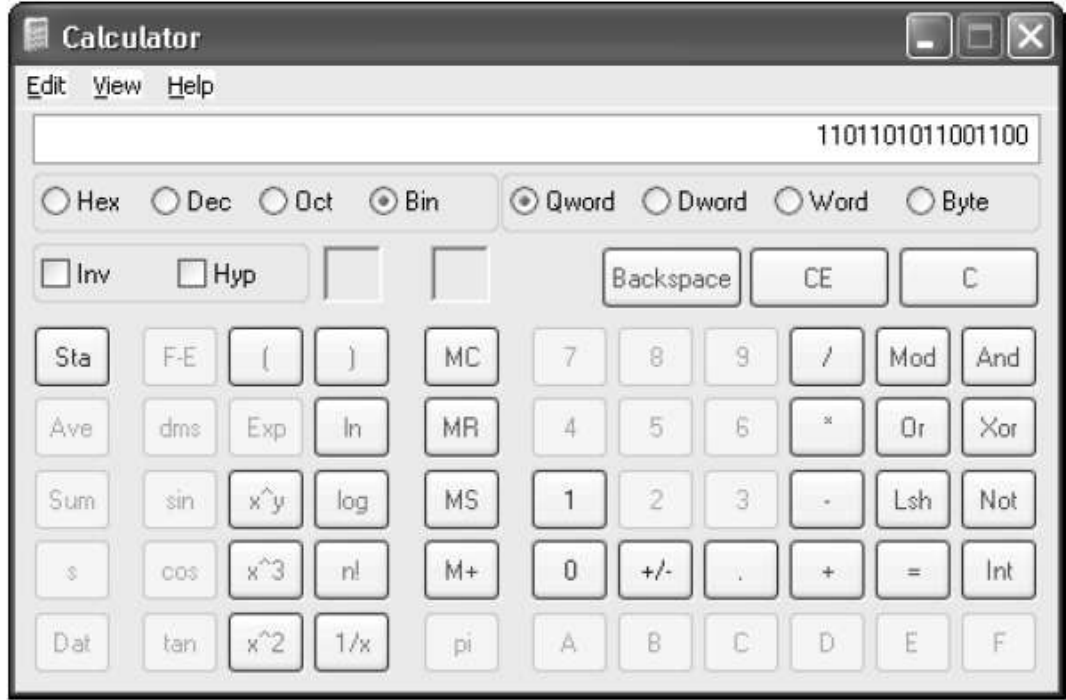


[ملحوظة: تغيير الشكل في Windows 7 فستجد بدلا منه الوضع programmer]

ضع حاسبة الـ windows في النظام العشري العادي Decimal وأكتب بداخلها رقم . لترى هذا الرقم في النظام الثنائي binary. فقط حول الآلة الحاسبة من Dec إشارة إلى الشكل :



.....إلى Bin , كما بالشكل



..... والآن سيظهر لك الرقم العشري فى نموذج النظام الثنائى .

ولأى عملية رياضية تريد أن تنفذها , فقط إستخدم الآله الحاسبة وغير وضع العرض لتراها فى نظام binary . أو العكس . الجمع , الطرح , الضرب و القسمة , كلهم ممكن العمل معهم فى حاسبة الـ windows . لاجابة لإحضار ورقة وقلم لتقوم بعمله يدوياً .

لاحظ كيف تنسق الأرقام الأخرى مشتملة على الإختيارات فى الآله الحاسبة , Hex تمثل hexadecimal و Oct هي Octal , Octal خارج نطاق هذا الكتاب , وحقيقتاً ليست لها صلة بالموضوع . ولكن هذا الفصل يقضى بعض الوقت فى نظام السداسى عشر HexaDecimal فى مقطع لاحقاً .

Why Binary?

السؤال الشائع الذى يُسأل "لماذا هناك حاجة لإستخدام Binary ؟ " لماذا فقط لا نستخدم Decimal ؟

الإجابة بسيطة ——— فهى تسمح للمعلومات أن يتم تمثيلها بحالتين — on أو Off . هذا يعنى أن فى نظام الحاسب (أو أى نظام إلكترونى آخر) يمكن أن يتم تمثيل المعلومات بعمليات التبديل وتغييرها من on إلى off والعودة مرة أخرى ويمكنها من العمل مع الـ binary . هذا هو الأساس لكيفية عمل المعالج والذاكرة (كلا من Random Access Memory أو RAM , و Hard Drive) .

تمثل الـ binary أيضاً ذبابات إلكترونية داخل الأسلاك — الأساس للشبكات والإنترنت .

ربما تبدو binary ثقيلة ولكنها تشكل الإشارة المرسله اسفل الجهاز العصبى لكل شئ تقريباً يتم فعله مع النقل الإلكترونى , والمعالجة , وتخزين كل البيانات , وجميع من حولك . تتدفق الـ binary فى دوائر , خلال أسلاك , وخلال الهواء .

Hexadecimal

السداسى عشر

HexaDecimal هو نظام أرقام مشابهة للنظام العشري والنظام الثنائى ولكن على وجه أكثر تعقيداً .

لماذا أكثر تعقيداً ؟ حسناً , النظام الثنائى لديه رقمين , النظام العشري لديه عشرة أرقام , على الجانب الآخر السداسى عشر لديه ستة عشر رقماً .

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F.

فى بعض التمثيل تستخدم الحروف \$, #, @, !, ~, و% بدلاً من ABCDEF (على التوالى) .

Interpreting Hexadecimal

تفسير السداسى عشر

دعنا نلقى نظرة على نظام الأرقام الـ HexaDecimal (Hex) ونقارنه بالنظام العشري Decimal , سيساعدك هذا على تعريف نفسك بالنظام .

Decimal	Hexadecimal
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C
13	D
14	E
15	F
16	10
...	...
30	1E
31	1F
32	20
33	21

لاحظ كيف تحدث الزيادة , ولوجود 16 رقماً , يظهر نمو الرقم ببطئ أكثر من النظام العشري . وكما فى النظام الثنائى , فإن HexaDecimal تقرأ من اليمين لليسار, مع زيادة الـ bits فى القيمة من اليمين لليسار.

كلمة *HexaDecimal* غريبة الأطوار لأن الـ *Hexa* مأخوذة من *Hexa* اليونانية للرقم 6 وكلمة *Decimal* مأخوذة من اللاتينية للرقم 10 , الأقدم , وأكثر دقة . وكان المصطلح اللاتيني *SexDecimal* , ولكن كان منبوزاً لأنه كان يعتقد أنه كان أكثر صعوبة (ناهيك عن انه أيضا لديه معنى بديلاً وهو *Base 60*)

أرقام السداسي عشر إما أن يكون لديه بادئ بـ 0x أو اللاحقة h , على سبيل المثال :

0x3F7D

ولغرضنا , سنقوم بنسيان هذا , لأنها تجعل الحياة أكثر تعقيداً , والمبتدأون يملون بهذا .

لتحويل قيمة من نظام السداسي عشر *HexaDecimal* إلى النظام الثنائي *Binary* . فقط تقوم بترجمة كل رقم في *HexaDecimal* إلى ما يساوي 4 bits في النظام الثنائي . لذا الرقم السابق يترجم كالتالي :

3	F	7	D
0011	1111	0111	1101

وأيضاً من السهل تحويل أى قيمة حرفية في النظام الثنائي إلى النظام السداسي عشر . خذ التالي :

101111

أضف أصفار إلى جهة اليسار لذلك يقسم هذا الرقم من الـ bits إلى أربعة بدون باقى .

00101011

الآن , قطع الإعداد إلى تجمع 4 bits (a nybble)

0010 1111

ثم قم بتحويل كل تجمع إلى رقم *Hexa Decimal* مناسب :

2 F

هذا يعطى:

2f

قمنا به !

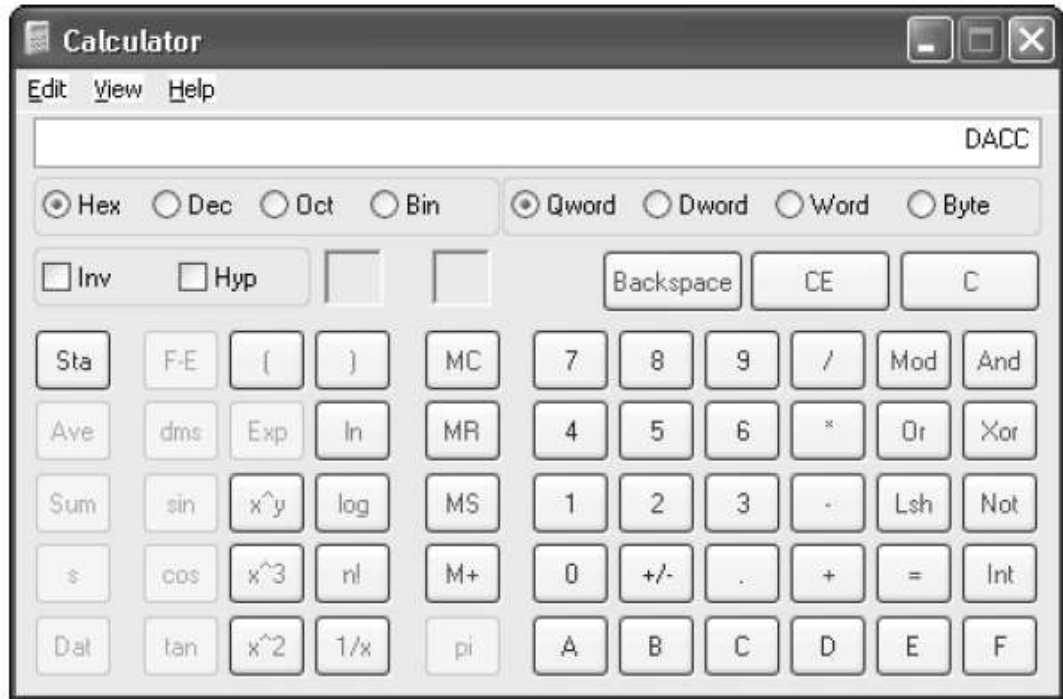
فى الحقيقة , أى *Nybble* تهتم بإختيارها ممثلة بشكل مباشر برقمين *HexaDecimal*

وهذا هو السبب لظهور الـ *byte* عادةً كإثنين من *Nybble* .

Hexadecimal and Windows Calculator

النظام السداسى عشر وحاسبة الـ windows

بنفس الطريقة التى قد عملت بها مع النظام الثنائى بإستخدام حاسبة الـ windows , يمكن أن تفعل نفس الشئ بالضبط مع النظام السداسى عشر . يمكنك أن تحول تنسيق الرقم بإستخدام المفتاح لواجهة الحاسبة أو من خلال قائمة النظام . يمكن تحويل الأرقام من وإلى Hexa بنفس الطريقة من Binary كما هو موضح بالشكل :



Representing Characters

تمثيل الحروف

إلى هذا الحد إلقينا النظر إلى أكثر من طريقة فى تمثيل الأرقام ولكن , كما تدرك فى الغالب , هناك أكثر من الأرقام ليتم إجراء عمليات حسابية عليها . هناك النصوص , الأرقام , الرموز , وهكذا الحروف , كما تسمى فى العام . كيف لك أن تذهب من النظام الثنائى (النظام من قبل كافة أجهزة الكمبيوتر التى تعمل على المستوى الأساسى) إلى الحروف التى أكتب بها الآن وتراها تحيط بك فى كل الأوقات عند إستخدام الحاسب ؟ .

للتعامل مع تمثيل الأحرف فقد تم تقديم نظام يسمى ASCII (كما تنطق AS-Kee) . ASCII وهى اختصار لـ *American Standard Code for Information Interchange* . منذ تمكنت الحاسبات من فهم الأرقام . كود ASCII يعد تمثيل رقمى للحرف مثال "a" أو " " ~ " او إجراء بعض الوصف لأمر ما .

ASCII هى طريقة قياسية لترميز الأحرف الكبيرة والصغيرة فى الإبجدية الإنجليزية , وكذلك الأرقام والحروف الخاصة . ويتم بإستخدام النظام الثنائى , وتحتاج فقط إلى 7 bits . لأنها تستخدم 7 bits , وتمثل تحديداً بـ 128 حرف.

DECIMAL	OCTAL	HEX	BINARY	VALUE
000	000	000	00000000	NUL (Null char.)
001	001	001	00000001	SOH (Start of Header)
002	002	002	00000010	STX (Start of Text)
003	003	003	00000011	ETX (End of Text)

004	004	004	00000100	EOT	(End of Transmission)
005	005	005	00000101	ENQ	(Enquiry)
006	006	006	00000110	ACK	(Acknowledgment)
007	007	007	00000111	BEL	(Bell)
008	010	008	00001000	BS	(Backspace)
009	011	009	00001001	HT	(Horizontal Tab)
010	012	00A	00001010	LF	(Line Feed)
011	013	00B	00001011	VT	(Vertical Tab)
012	014	00C	00001100	FF	(Form Feed)
013	015	00D	00001101	CR	(Carriage Return)
014	016	00E	00001110	SO	(Shift Out)
015	017	00F	00001111	SI	(Shift In)
016	020	010	00010000	DLE	(Data Link Escape)
017	021	011	00010001	DC1	(XON) (Device Control 1)
018	022	012	00010010	DC2	(Device Control 2)
019	023	013	00010011	DC3	(XOFF) (Device Control 3)
020	024	014	00010100	DC4	(Device Control 4)
021	025	015	00010101	NAK	(Negative Acknowledgement)
022	026	016	00010110	SYN	(Synchronous Idle)
023	027	017	00010111	ETB	(End of Trans. Block)
024	030	018	00011000	CAN	(Cancel)
025	031	019	00011001	EM	(End of Medium)
026	032	01A	00011010	SUB	(Substitute)
027	033	01B	00011011	ESC	(Escape)
028	034	01C	00011100	FS	(File Separator)
029	035	01D	00011101	GS	(Group Separator)
030	036	01E	00011110	RS	(Request to Send Record separator)
031	037	01F	00011111	US	(Unit Separator)
032	040	020	00100000	SP	(Space)
033	041	021	00100001	!	(exclamation)
034	042	022	00100010	"	(double quote)
035	043	023	00100011	#	(number sign)
036	044	024	00100100	\$	(dollar sign)
037	045	025	00100101	%	(percent)
038	046	026	00100110	&	(ampersand)
039	047	027	00100111	'	(single quote)
040	050	028	00101000	((left/opening parenthesis)
041	051	029	00101001)	(right/closing parenthesis)
042	052	02A	00101010	*	(asterisk)
043	053	02B	00101011	+	(plus)
044	054	02C	00101100	,	(comma)
045	055	02D	00101101	-	(minus or dash)
046	056	02E	00101110	.	(dot)
047	057	02F	00101111	/	(forward slash)
048	060	030	00110000	0	
049	061	031	00110001	1	
050	062	032	00110010	2	
051	063	033	00110011	3	
052	064	034	00110100	4	
053	065	035	00110101	5	
054	066	036	00110110	6	
055	067	037	00110111	7	
056	070	038	00111000	8	
057	071	039	00111001	9	

058	072	03A	00111010	:	(colon)
059	073	03B	00111011	;	(semi-colon)
060	074	03C	00111100	<	(less than)
061	075	03D	00111101	=	(equal sign)
062	076	03E	00111110	>	(greater than)
063	077	03F	00111111	?	(question mark)
064	100	040	01000000	@	(AT symbol)
065	101	041	01000001	A	
066	102	042	01000010	B	
067	103	043	01000011	C	
068	104	044	01000100	D	
069	105	045	01000101	E	
070	106	046	01000110	F	
071	107	047	01000111	G	
072	110	048	01001000	H	
073	111	049	01001001	I	
074	112	04A	01001010	J	
075	113	04B	01001011	K	
076	114	04C	01001100	L	
077	115	04D	01001101	M	
078	116	04E	01001110	N	
079	117	04F	01001111	O	
080	120	050	01010000	P	
081	121	051	01010001	Q	
082	122	052	01010010	R	
083	123	053	01010011	S	
084	124	054	01010100	T	
085	125	055	01010101	U	
086	126	056	01010110	V	
087	127	057	01010111	W	
088	130	058	01011000	X	
089	131	059	01011001	Y	
090	132	05A	01011010	Z	
091	133	05B	01011011	[(left/opening bracket)
092	134	05C	01011100	\	(back slash)
093	135	05D	01011101]	(right/closing bracket)
094	136	05E	01011110	^	(caret/circumflex)
095	137	05F	01011111	_	(underscore)
096	140	060	01100000		
097	141	061	01100001	a	
098	142	062	01100010	b	
099	143	063	01100011	c	
100	144	064	01100100	d	
101	145	065	01100101	e	
102	146	066	01100110	f	
103	147	067	01100111	g	
104	150	068	01101000	h	
105	151	069	01101001	i	
106	152	06A	01101010	j	
107	153	06B	01101011	k	
108	154	06C	01101100	l	
109	155	06D	01101101	m	
110	156	06E	01101110	n	
111	157	06F	01101111	o	

112	160	070	01110000	p	
113	161	071	01110001	q	
114	162	072	01110010	r	
115	163	073	01110011	s	
116	164	074	01110100	t	
117	165	075	01110101	u	
118	166	076	01110110	v	
119	167	077	01110111	w	
120	170	078	01111000	x	
121	171	079	01111001	y	
122	172	07A	01111010	z	
123	173	07B	01111011	{	(left/opening brace)
124	174	07C	01111100		(vertical bar)
125	175	07D	01111101	}	(right/closing brace)
126	176	07E	01111110	~	(tilde)
127	177	07F	01111111	DEL	(delete)

لاحظ كيف تستخدم الحروف السالفة 7 bits لقيم حرفية فى النظام الثنائي لتمثيل الأحرف . كما قلت فى وقت سابق . وهذا يسمح بتمثيل 127 حرف , وتمثل مجموعة من الحروف شكل ما يعرف بمجموعة أحرف .

لماذا 7 bits ؟ كل الحروف الأساسية أبقت على أول 127 من الشقوق [slots فتحات] لتقليل مساحة الاستخدام التى نحتاجها للبيانات . فى الحقيقة ,الكثير من التطبيقات فى وقت مبكر إستبعدت الصفر الثامن . ولزيادة حروف أكثر , كلهجات الأحرف والرموز الرياضية . وغالباً ما يضاف الـ bit الثامن , توفير 256 حرف فى الجميع , وهناك مجموعات أحرف كثيرة . تمثل كلا منهم حروفاً مختلفة . هذا بالإضافة إلى مجموعة الأحرف يعرف بـ *extended character set* .

VALUE	DECIMAL	HEX
--	128	80
--	129	81
--	130	82
--	131	83
--	132	84
--	133	85
--	134	86
--	135	87
--	136	88
--	137	89
--	138	8A
--	139	8B
--	140	8C
--	141	8D
--	142	8E
--	143	8F
--	144	90
--	145	91
--	146	92
--	147	93
--	148	94
--	149	95
--	150	96

--	151	97
--	152	98
--	153	99
--	154	9A
--	155	9B
--	156	9C
--	157	9D
--	158	9E
--	159	9F
	160	a0
ı	161	a1
ç	162	a2
£	163	a3
¤	164	a4
¥	165	a5
—	166	a6
§	167	a7
¨	168	a8
©	169	a9
®	170	AA
«	171	AB
¬	172	AC
—	173	AD
®	174	AE
-	175	AF
°	176	B0
±	177	B1
—	178	B2
—	179	B3
—	180	B4
μ	181	B5
¶	182	B6
•	183	B7
•	184	B8
—	185	B9
o	186	BA
»	187	BB
¹ / ₄	188	BC
¹ / ₂	189	BD
³ / ₄	190	BE
¿	191	BF
À	192	C0
Á	193	C1
Â	194	C2
Ã	195	C3
Ä	196	C4
Å	197	C5
Æ	198	C6
Ç	199	C7
È	200	C8
É	201	C9
Ê	202	CA
Ë	203	CB
Ì	204	CC

í	205	CD
î	206	CE
ï	207	CF
Ð	208	D0
Ñ	209	D1
Ò	210	D2
Ó	211	D3
Ô	212	D4
Õ	213	D5
Ö	214	D6
×	215	D7
Ø	216	D8
Ù	217	D9
Ú	218	DA
Û	219	DB
Ü	220	DC
Ý	221	DD
Þ	222	DE
ß	223	DF
à	224	E0
á	225	E1
â	226	E2
ã	227	E3
ä	228	E4
å	229	E5
æ	230	E6
ç	231	E7
è	232	E8
é	233	E9
ê	234	EA
ë	235	EB
ì	236	EC
í	237	ED
î	238	EE
ï	239	EF
ð	240	F0
ñ	241	F1
ò	242	F2
ó	243	F3
ô	244	F4
õ	245	F5
ö	246	F6
÷	247	F7
ø	248	F8
ù	249	F9
ú	250	FA
û	251	FB
ü	252	FC
ý	253	FD
þ	254	FE
ÿ	255	FF

الرمز ——— ويشير إلى أنه لا يوجد حرف معين .

باستخدام ASCII يمكنك ان تأخذ القيم الحرفية النصية وتحويلهم إلى قيم حرفية من الأصفار الأحاد والعودة مرة أخرى . فلا يكون هناك فقد للبيانات ولا أى غموض .

خذ القيمة الحرفية التالية :

Programming is great!

بالرجوع إلى جدول ASCII , هذه الأحرف يمكن أن تحول إلى binary Bytes . وهنا قد أتممت القيام به لاجل :

```
010100000111001001101111011001110111001001100001011011010110110110100101101110011
001111001000000110100101110011001000000110011101110010011001010110000101110100001000
01
```

Character	Binary
P	01010000
R	01110010
O	01101111
G	01100111
R	01110010
A	01100001
M	01101101
M	01101101
I	01101001
N	01101110
G	01100111
	00100000
I	01101001
S	01110011
	00100000
G	01100111
R	01110010
E	01100101
A	01100001
T	01110100
!	00100001

ولتحويل سلسلة النظام الثنائي للخلف إلى ASCII ليس صعباً . فإذا كان عليك ان تفعل ذلك يدوياً , الخدعة هي أنك تقطع سلسلة النظام الثنائي إلى Bytes قبل العمل عليها . العمل على قطار طويل من الاصفار والآحاد عمل صعب , ومجال الأخطاء عالى جداً .

```
01010000 01110010 01101111 01100111 01110010 01100001 01101101 01101101 01101001
01101110 01100111 00100000 01101001 01110011 00100000 01100111 01110010 01100101
01100001 01110100 00100001
```


ثم تعمل خلال الجدول

```
P      r      o      g      r      a      m      m
01010000 01110010 01101111 01100111 01110010 01100001 01101101 01101101

i      n      g      i      s      g
01101001 01101110 01100111 00100000 01101001 01110011 00100000 01100111

g      e      a      t      !
01110010 01100101 01100001 01110100 00100001
```

ليس هناك فقد في البيانات أثناء العملية .
الشيء الجيد في البرمجة الحديثة هي أنك نادراً ما تقلق أو تتضايق لأستخدام تمثيل النظام الثنائي للأحرف .
معظم هذا يتم لك حديثاً . ومع ذلك , هناك أوقات تأتي في متناول اليدين عند معرفة العلاقة بين النظام الثنائي
والأحرف (لشيء واحد , ستعطيك رؤية لكيفية تخزين البيانات في الذاكرة في القرص الصلب) . إنظر الشكل :



من الهام أن نتذكر أن عرض الأحرف الحقيقية ليس لديه شيء يفعله مع كود النظام الثنائي . تترجم أرقام
binary (أو Hexa Decimal أو Octal) , وتعرض الأحرف المناسبة بواسطة التطبيق أو نظام
التشغيل .

إلى هذا الحد , فقد رأينا تنسيق الأرقام في العميات ورأينا كيف يمكن إستخدام الأرقام التمثيل الأحرف للعرض
, المعالجة, النقل , والتخزين . التالي , سنلقى نظرة إلى المعاملات الرياضية والمنطقية .

Operators

المعاملات

المزيد من البرمجة عن معالجة البيانات , كلا من النصية والرقمية في الطبيعي . المعاملات تمكّنك من العمل مع البيانات ومقارنة المعلومات .

دعنا نبدأ بأخذ نظرة على بعض المعاملات الأساسية لتألف ماذا تعمل وكيف تعمل ؟ . ومن الهام أن تدرك ان ليس كل المعاملات لديها نفس الرمز في كل لغات البرمجة (وهذا من أكبر احوار العثرات تطرح الطلاب أرضاً عندما يحاولون تعلم لغة برمجة ثانية) إلى الآن , إذا فهمت ما تفعله هذه , سوف تساعدك في وقت لاحق إلى حد كبير .

تتنوع المعاملات إلى خمسة تصنيفات , كلا منهم سيوصف في المقاطع التالية

- ☐ Arithmetic
- ☐ Assignment
- ☐ Comparison
- ☐ Logical
- ☐ String

Arithmetic Operators

المعاملات الرياضية

هذه المعاملات هي مثل المعاملات التي تألفها , لأن معاملات الرياضة تستخدمها للعمليات الرياضية العادية . هناك سبعة معاملات رياضية شائعة :

Operator	Description	Example	Result
+	Addition	If $x = 6$ $x + 4$	10
-	Subtraction	If $x = 6$ $x - 2$	4
*	Multiplication	If $x = 6$ $x * 2$	12
/	Division	If $x = 6$ $x / 3$	2
++	Increment (add one)	If $x = 6$ $x++$	7
--	Decrement (subtract one)	If $x = 6$ $x--$	5
%	Modulus (division remainder)	$7\%2$ $10\%4$ $12\%6$	1 2 0

Assignment Operators

معاملات التكليف او الإسناد

[نقصد هنا بالإسناد أو التكليف هي إعطاء قيمة لشيء ما ويميل المترجم إلى كلمة إسناد فأختر ما شئت]

تستخدم معاملات الإسناد لإسناد قيمة . وسيستخدم هذا حصرياً عندما نأتى لإستخدام المتغيرات (المتغير x في الأمثلة السابقة هي مثال على المتغير الذى أسند إليه القيمة 6)
وهناك ستة معاملات إسناد أو تكليف :

Operator	Example	Is Equivalent to ...
=	$x = 6$ $x = y$	$x = 6$ $x = y$
+=	$x += y$	$x = x + y$
-=	$x -= y$	$x = x - y$
*=	$x *= y$	$x = x * y$
/=	$x /= y$	$x = x / y$
%=	$x \% = y$	$x = x \% y$

في الأمثلة السابقة , كلا من x , y هما متغيرات , ولا بد أن يكون للمتغيرات إسم مميز فريد (إذا لم تعيد إستخدامهم) من أجلنا هنا , x,y على ما يرام !

Comparison Operators

معاملات المقارنة

تستخدم معاملات المقارنة لمقارنة شيئين . ستجد عندما تكون بصدد العمل مع الكود في الفصل السابع " The Structure of Coding " فسيكون هذا مفيد جداً .

هناك ستة معاملات للمقارنة . كلاً منهم يرجع بالقيمة true أو False , إعتماًداً على ناتج المقارنة سواء كان صحيحاً أو خطأ .

Operator	Description	Example
==	Equal to	$5 == 8$ (returns false) $8 == 8$ (returns true) $"cat" == "dog"$ (returns false) $"cat" == "cat"$ (returns true)
!=	Not equal to	$5 != 8$ (returns true) $8 != 8$ (returns false)
>	Greater than	$5 > 8$ (returns false) $8 > 3$ (returns true)
<	Less than	$5 < 8$ (returns true) $8 < 3$ (returns false)
>=	Greater than or equal to	$5 >= 8$ (returns false) $8 >= 3$ (returns true) $8 >= 8$ (returns true)
<=	Less than or equal to	$5 <= 8$ (returns true) $8 <= 3$ (returns false) $3 <= 3$ (returns true)

Logical Operators

المعاملات المنطقية

تمكنك المعاملات المنطقية من إدخال مسألة منطقية إلى الكود الذى تكتبه وتمكنك من عمل تركيب بين المعاملات السابقة .

هناك ثلاثة معاملات منطقية :

Operator	Description	Example
<code>&&</code>	And	<pre>x = 7 y = 2 (x < 12 && y > 1) returns true In English, the preceding states "x is less than 12, and y is greater than 1." x = 7 y = 2 (x < 12 && y < 1) returns false</pre>
<code> </code>	Or	<pre>x=6 y=1 (x==6 y==5) returns true In English, the preceding states "x is equal to 5, and y is equal to 5." x=6 y=1 (x==5 y==5) returns false x=6 y=1 (x==6 y==1) returns true</pre>
<code>!</code>	Not	<pre>x=6 y=3 !(x==y) returns true In English, the preceding states "x is not equal to y." x=3+3 y=4+2 !(x==y) returns false</pre>

String Operators

معاملات القيم الحرفية

هى معاملات تعمل مع القيم الحرفية , القيم الحرفية هى عادةً قصاصات من النصوص , وها هم مثالين لقيم حرفية :

```
string1 = "Hello "
string2 = "World!"
```

باستخدام معاملات القيم الحرفية , يمكن أن يرتبط (يتسلسل) نص بأخر .

```
string3 = string1 + string2
```

سيمسك المتغير `String3` القيمة `Hello World!`

شيء واحد يبدو للقادمين الجدد إلى القيم الحرفية string هي إيجاد صعوبة في التغلب على المسافات . هذا هو الحال , تتبع المسافات بين القيم الحرفية والتأكد من أن نهاية القيمة الحرفية منطقية .

على سبيل المثال , في المثال السابق يمكن أن يكون لدى التالي :

```
string1 = "Hello"  
string2 = "World!"
```

ثم يكون على أن أسلسلهم مع بعضهم [أي يجمع بينهم]

```
string3 = string1 + string2
```

ومع ذلك , الآن سيمسك المتغير الجديد string3 القيمة HelloWorld! [لاحظ انها بدون مسافات]

أمر واحد لإصلاح هذا هو إضافة مسافة عند تنفيذ عملية الربط بينهم .

```
string3 = string1 + " " + string2
```

الآن سيكون المتغير string3 صحيحاً ويكون لديه القيمة Hello World! [لاحظ المسافة بينهم] سنلقى نظرة على هذا في تفاصيل أكثر , في فصول لاحقة . ولكن الآن إذا كانت لديك الأساسيات مثبتة سيكون أسهل وأملاً في أن يزيد الفهم كثيراً .

Summary

الملخص

في هذا الفصل , أخذنا نظرة في كيفية الانتقال من المفاهيم إلى الكود . وفحصنا بعض من تنسيقات الإرقام التي ربما ستمر بها مروراً بكيفية يمكن أن تستخدم الـ binary, Hexadecimal , Octal لتمثيل الأحرف باستخدام ASCII . وأيضاً قمنا بفحص بشكل مختصر لماذا تكون الـ binary هامة عند العمل مع عمليات الحاسب , وكيف تكون هي الجوهر في الغالب لكل شيء تفعله مع البيانات في العالم الإلكتروني .

أيضاً ألقينا نظرة على المعاملات وكيف يمكنك من أخذ بيانات بسيطة وتبدأ العمل معها . وهو في نهاية المطاف , كل شيء عن عمليات الحاسب [أو ما يسمى في الترجمة "حوسبة"]

الفصل التالي يلقي نظرة على ادوات البرمجة .

5

The Tools for Programming

أدوات البرمجة

فى هذا الفصل , سنلقى نظرة على بعض التطبيقات التجارية و المجانية , التى ستستخدمها خلال ما تبقى من هذا الكتاب .

معظم ما ستحتاجه من برامج فى هذا الكتاب ستكون مجانى ومتاح للتنزيل من خلال الإنترنت .

ما ستختاره لتعمل به يعتمد عليك , ولكن هذا الفصل ينظر إلى البرمجة من وجهة نظر لغات البرمجة المتعدده , حتى تحصل على تغطية واسعة و كذلك خبرة .

Make Your Workspace Your Own

إجعل مساحة العمل خاصة بك .

المساحة التى تعمل فيها بقدر ماهى اداة برمجية فهى جزء من برنامج أو مجموعة برامج تحتاجها , فى الحقيقة . ربما تكون أحد أهم العوامل فى مهنتك البرمجية فيما بعد . لان قلة الجودة فى مساحة العمل وقلة إتبارحك بها تعنى أن إستمتاعك بالبرمجة سيقول الوقت الذى تقضيه للعمل سيقول أيضاً .

The Keyboard

لوحة المفاتيح

لايهم أى نوع من البرمجة تخطط ان تعمل به , أو أى لغات البرمجة تختارها لتعمل بها , شئ واحد يجب ان يكون فى ذهنك هو ان ستكتب كود , عندما تأتى للبرمجة , ستقضى الوقت خلف لوحة المفاتيح , مثال التى فى الشكل التالى :

هناك أنواع كثيرة مختلفة من لوحات المفاتيح . اليوم . يبدو أنهم قد جاءوا فى كل الأشكال والأحجام ونوع الإتصال (سلكى , لاسلكى) . فى العام , نمط وتخطيط أهم لوحات المفاتيح , تأتى فى ثلاثة تصانيف :

- ☐ Traditional
- ☐ Ergonomic
- ☐ Bizarre/futuristic



شكل لوحة المفاتيح

كن حذراً من أى شئ غريب , فقد وضعت لوحات المفاتيح بطريقة غريبة يمكن ان يكون من الصعب الإعتياد عليها وعندما تكون بعيداً عن نظام الحاسب فهذا يجعلك أقل كفاءة فى التعامل معها.

إحصل على لوحة مفاتيح تشعرك بالإرتياح معها , فإن لوحات المفاتيح من النوع Ergonomic تجعل الكتابة لوقت طويل أسهل , ولكن تتطلب كثيراً " إعادة التعلم " لكيفية الكتابة . وعلى المدى القصير قد يبدو أماً حقيقياً . ولكن على المدى البعيد ستشعر أنك بالفعل قمت بحفظ أماكن الحروف فى لوحة المفاتيح وتكتشف أن هذا كان أمراً مفيداً .

أياً كان ما اخترته للعمل معه , إليك بعض المؤشرات :

- تأكد من أن لوحة المفاتيح مريحة الإستخدام . يتطلب البعض منها كثرة الضغط على المفاتيح أكثر من غيرها وإذا كانت لوحتك غير مريحة للكتابة بشكل عام , ستكون جلسات كتابة الكود الطويلة أقل متعة .
- لا تقفز إلى لوحات المفاتيح المستحدثة المريحة مباشرة . ودائماً حاول أن تجرب نمط جديد من لوحات المفاتيح قبل أن تلزم واحدة .
- إذا كانت لوحة المفاتيح الحالية لديك متهاكة أو بها مفاتيح لا تعمل بطريقة جيدة , فاستبدلها .

Workspace مساحة العمل

تعود على العمل على لوحة المفاتيح فى وقت مبكر , وقم بتخصص مساحة العمل الخاصة بك . إجعلها مريحة كلما أمكن وأحضر مقعد جيد لتجلس عليه . إذا كان ظهرك لا يشعر بالراحة أو إذا كانت لوحة المفاتيح فى إرتفاع خاطئ منك . عندها ستعانى من قلة التركيز , والوقت الذى ستقضيه فى البرمجة سيكون أقصر وأقل راحة . وهذا بلا شك سيؤثر على تركيزك , و تعلمك وأدائك .

تأكد من ان لديك إضاءة جيدة , لو امكن , منفذ هواء جيد , لإنك ستظل جالساً لوقت طويل. حاول أن تجعل المكان دافئ . ولكن ليس حار . أضف النباتات وأى شئ آخر يشعرك بالسعادة والراحة .

Desk المكتب

تأكد من أن لديك مساحة وفيرة في مكتبك عندما تبرمج . الفوضى سيئة للعقل وتجعل الأمور صعبة عليك أكثر من اللازم . كمبرمج مبتدئ , ستراجع الكتب الكثيرة (على أمل أن ذلك أحدهم) وغيرها من المواد العلمية . إجعل بها مساحة للمكتب والورق الذي ستستخدمه .

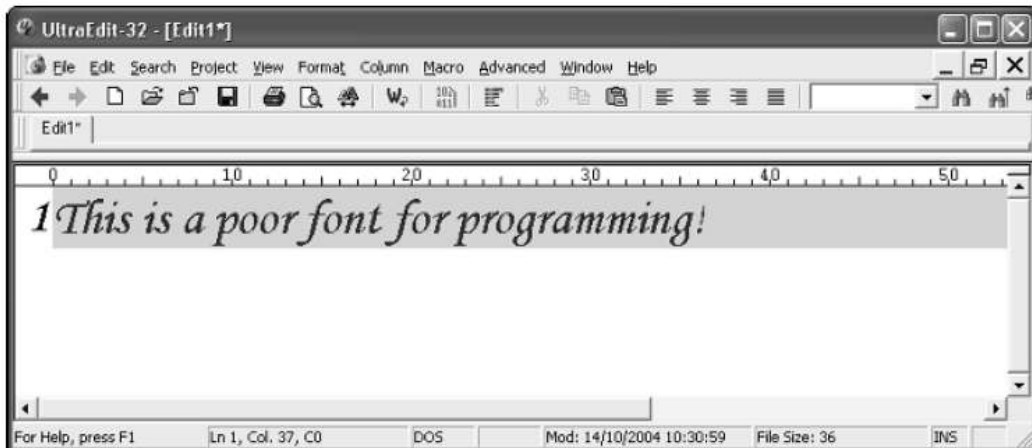
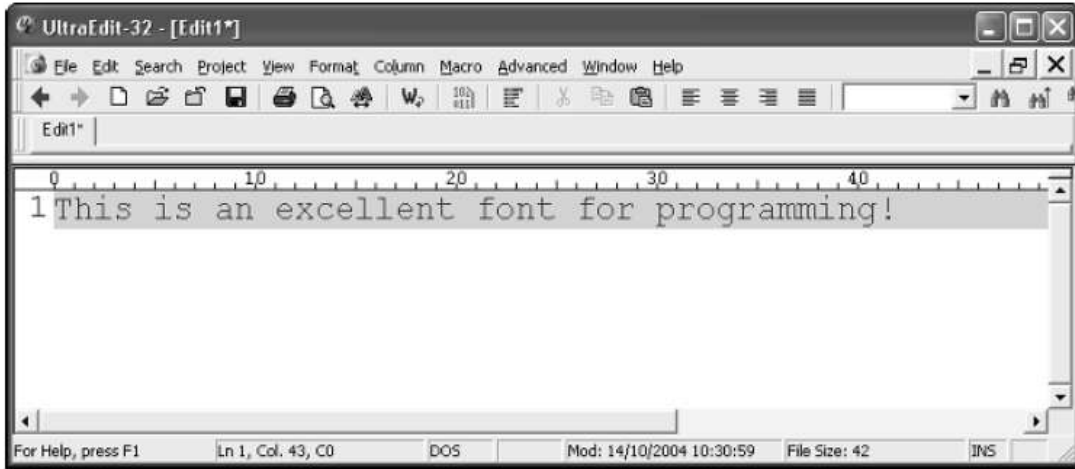
أيضاً ستجد ان إضافة الملاحظات سيكون جيداً . لذا تأكد من أن لديك قلم ومخزون وفير من الورق والمسودات اللاصقة . دفتر Hardbound مثالي لتتابع مشروعاتك و كذلك تقدمك .

Monitor الشاشة

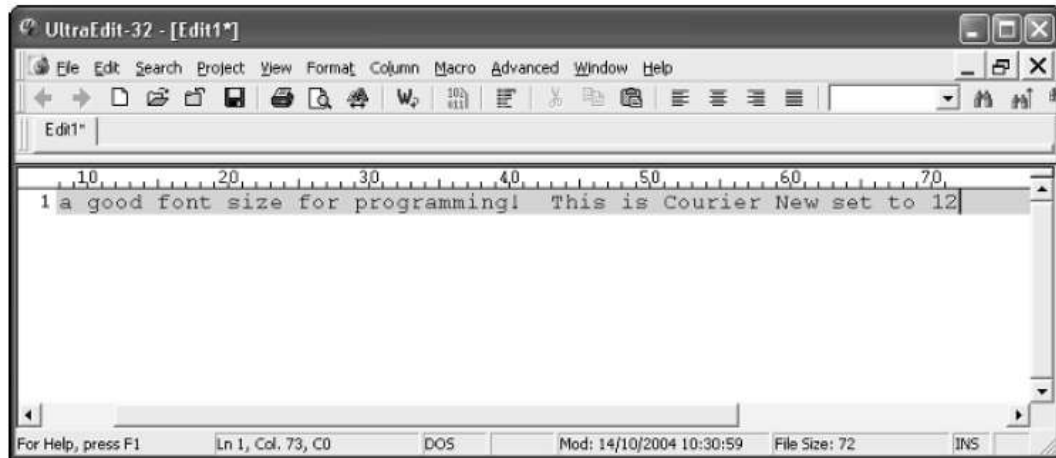
تأكد من انك في زاوية مريحة من لوحة المفاتيح والشاشة . فاذا كان هناك إشعاع يخرج من الشاشة فإن هذه مشكلة . فستحتاج إلى إعادة ضبط النظام أو تثبيت أداة تنقية الاشعاع . لأنك ستكتب نصوص كثيرة فإحتمال أكيد ان تقرأ ما قمت بكتابته . لذا تأكد من أن خط التطبيق الذي تستخدمه أسهل للقراءة .

Fonts الخطوط

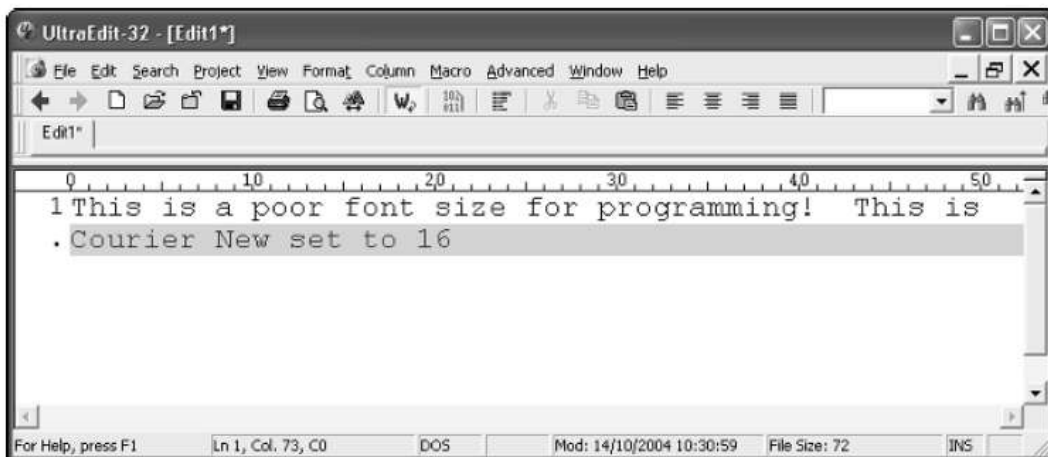
قم بتغيير حجم الخط وأكتب ما يوفرلك أفضل قراءة . الخطوط البسيطة كـ Arial أو Courier New أفضل بكثير (إنظر الشكل) من الخطوط الخيالية (إنظر الشكل الذي يليه) مثال cursive types.



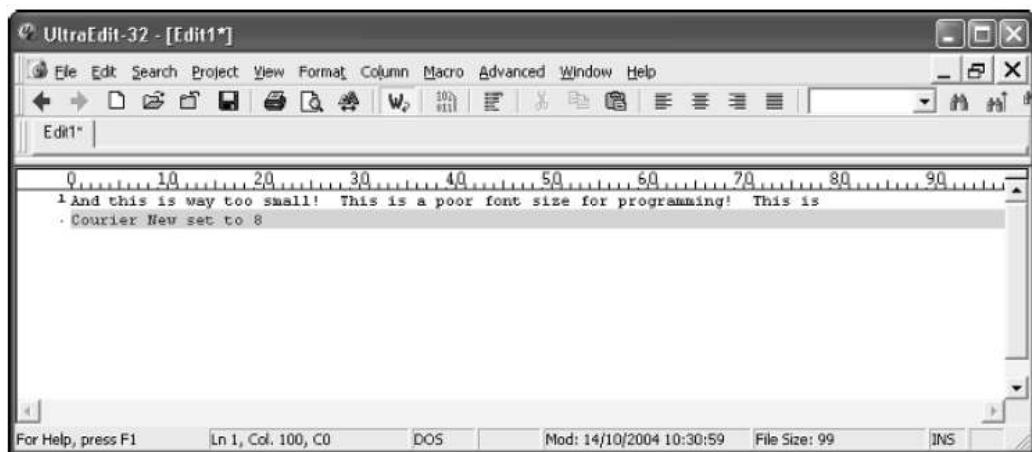
تأكد من أن حجم الخط مناسب أيضاً, فإذا كنت تستخدم خط مثل courier New, إذاً فحجم الخط 12 أو 14 مناسب . كما يعرض في الشكل :



حجم الخط 16 وما علا عن ذلك أكبر بكثير لايمكنك من أن ترى كمية كافية من الكود في الشاشة أمامك كما هو موضح بالشكل :

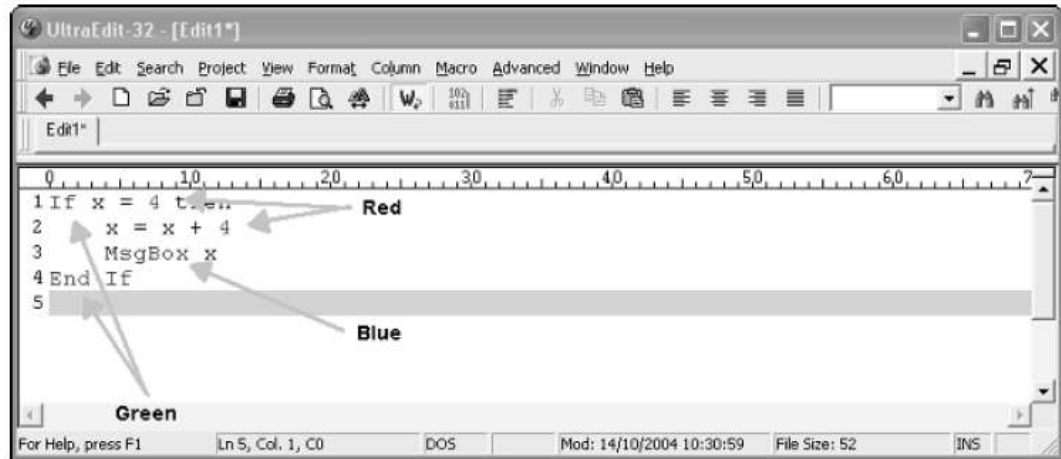


حجم الخط الصغير جداً يكون غير واضح للقراءة . وخاصة سيكون من الصعب أن تلاحظ المعاملات (/ , * , + , -) وهكذا الأقواس النصية والأقواس المتعرجة . كما هو بالشكل :

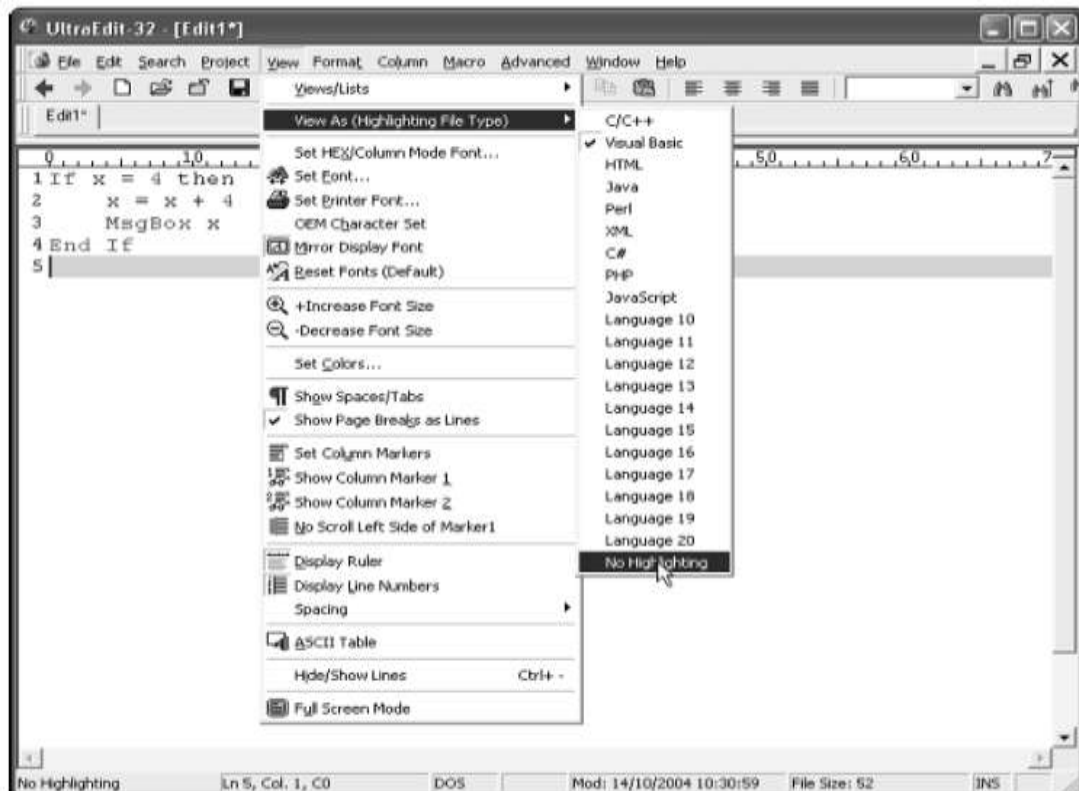


ما لم يكن البرنامج يقدم بعض النوع من تسليط الضوء على تركيب الجملة (كما هو الحال , فإن تطبيقات البرمجة تسلط الضوء على جوانب مختلفة من الكود بمختلف الألوان) فمن الجيد ان تترك الخط بلونه الأسود . فتسليط الضوء على قواعد بناء الجملة يعد مكافأه جيدة , ولكن هذا يحتاج ان تتعلم ماذا تعنى الألوان المختلفة وما علاقتها بقواعد لغة البرمجة التى قمت بإختيارها .

محرك النصوص الذى سأستخدمه هو UltraEdit, فيمكنك إعدادة ليستخدم خاصية تسليط الضوء على قواعد بناء الجملة كما هو موضح بالشكل ولاحظ الألوان .



إما للمبتدئين , فالإضطرار إلى التعامل مع الألوان و أيضاً فهم الكود يمكن أن يمثل لهم معضلة , ربما يكون من الأفضل إيقاف خاصية تسليط الضوء Highlight . ونجد فى UltraEdit كما مع معظم التطبيقات الأخرى انه يتم إستخدام خاصية تسليط الضوء فهى خاصية من السهل أن توقفها إلى حين أن تكون مستعداً لها كما بالشكل :



Choose Your Language

إختيار لغتك

هناك الكثير من اللغات التي يمكن ان تختار منهم , الشئ الحسن أن هذا الكثير يمكن ان ينفذ نفس المهام , وإذا إخترت بحرص , إعتيماً على ما تعتقد انك تريده , ستجد واحدة تلائم إحتياجاتك (على أية حال , لفترة بسيطة , حتى تحتاج لفعل شئ آخر مختلف).

الفصل الثاني " Why Learn to Program? " تكلمنا عن الانواع المختلفة من لغات البرمجة وفيما يتم إستخدامها . فى هذا الفصل , سافترض أن تعلم البرمجة هو الهدف الرئيسى لديك , وسأتكلم عن كيف يمكنك أن تفعل أفضل ما فى هذه اللغات المتنوعة .

أحد العوامل التي تتحكم فى أى لغة ستستخدمها فى برمجة تطبيقاتك هي كيفية تعلمك البرمجة .

Learning to Program

تعلم البرمجة

تعلم البرمجة مماثل كثيراً لتعلم أى لغة أجنبية , فإنك مع اللغة الأجنبية , ستجد ان بعض العناصر التي تتعلمها مألوفاً لك , فى حين أن اللغات الأخرى ستكون متشابهة بشكل خارق للعادة , ولكن بطريقة مختلفة مما تتوقع . فيقول بعض الناس أن تعلم لغة ثالثة أسهل من تعلم لغة ثانية , وهذا صحيح . فمعظم الناس يعتقدون ذلك , وهذا لأن إمتلاك لغة ثانية فى حوزتك , يعنى انه يمكنك ان تستنتج من خلفيتك العامة كيف تتعلم التالية , ومع ذلك , أنا أؤمن ان هذا ليس دقيقاً تماماً . أعتقد أن السبب فى لماذا أصبح التعلم أسهل هو انك تتعلم ألا تتوقع ان اللغات تتبع قواعد مشابهة . فعند وجود قواسم مشتركة , تكون الحياة أسهل , وإذا لم يوجد الكثير منها , فستصاب بالإحباط والإنزعاغ منها .

فإن تعلم البرمجة يختلف عن البرمجة الواقعية فى عدد من الطرق . فكثر الاختلاف تعتمد على المسار الذي تأخذه فى التعلم :

School/College

المدرسة / الكلية

المدرسة او الكلية هي المسار التقليدى لتعلم البرمجة , فقبل أن تلمس أى كود , تدخل فى تفاصيل تاريخ البرمجة وكيف تأثرت البرمجة فى الوقت الحاضر بشدة جراء التقدم الذى تحقق على مر السنين . وما تلاحظه أن الكثير من الأشياء التي نعتقد كونها مفاهيم "حديثة" , هي فى الحقيقة موجودة منذ وقت طويل .

فإن المرور على تاريخ البرمجة يمكن ان يكون أكثر تشوقاً , والتعرض إلى اللغات والأفكار القديمة ليس فقط يجعلك تقدر كم سهولة البرمجة اليوم , ولكن يساعدك فى فهم الكثير من منطق وتركيب الكود الحديث . بعد تغطية تاريخ البرمجة من المحتمل أنك ستلقى نظرة فى شئ من التفصيل على المفاهيم التي وراء الكود , وطريقة العمل مع الكود وتجهيز ومعالجة البيانات وسيتم تغطيته بالتفصيل .

عليك ان تقوم بكتابة بعض الكود , فالبرمجة التي تقوم بكتابتها ستدعم المادة التي تقوم بتغطيتها , لتعطى معنى حقيقى للمفاهيم .

كمية البرمجة التي يمكن بالفعل ان تستدعيها إلى مشكلات العالم الحقيقى تعتمد كثيراً على أى اللغات المستخدمه فى الفصل الدراسى , فإن اللغات مثل C++ , java تمكنك من إنشاء تطبيقات واقعية يمكن ان تضعها لتعمل على حل المشكلات الواقعية , بينما لغات أخرى (مثل Smalltalk) لا تستطيع .

فإذا كنت محظوظ بدرجة كافية لتدرس لغة لديها أهمية في العالم الواقعي , يمكنك أن تطبق ما تتعلمه بأسرع ما يمكن . في الحقيقة , أود أن أوصيك تماماً أن تفعل ذلك لأنك ستوسع معرفتك البرمجية أسرع بكثير من البرمجة في الواقع , فكلما تبرمج أكثر, كلما تم ترسيخ المعرفة عندك , من خلال تحديد مهام لنفسك , فأنت بذلك تزيد نطاق المعرفة لديك بشكل أكثر, ومن ثم توسيع مهاراتك وإكتساب خبرات جديدة , هذه الخبرة لا تقدر بثمن عندما تستدعيها إلى المشاريع التي ربما تقوم بها وأيضاً تعطيك ميزة عندما تقع في إختبار .

Tips

نصائح

- خذ الملاحظات الوفيرة .
- مارس بانتظام واعمل على مشاريع خاصة بك "غير مرتبط بالساعات" .
- الكثير من الطلاب لهم الحق في ترخيص "تراخيص الطلاب" للبرمجيات وأدوات البرمجة — إن استطعت , خذ الميزة من هذه العروض فربما يمكنك ان تكسب ألاف الدولارات .
- إذا اتاحت لك الفرصة لتستخدم أكثر من لغة , إقتنصها .

Work-Based Training

التدريب في العمل

الكثير والكثير من الشركات يرون تدريب موظفيهم جزء ذا قيمة يحافظ على مرونة القوى العاملة ويمنحهم الميزة على منافسيهم . وبسبب هذا , أصبح تدريب العاملين أكثر شيوعاً .

يختلف التدريب في العمل عن مسار البرمجة في المدارس والكلية التقليدية . فتجدهم أولاً يدرسون تاريخ البرمجة و يبدو على أنه ليس له صلة بالموضوع لأنه يأخذ الكثير من الوقت لتغطيته , ولا يرى ان له إستخدام هام في وقت التدريب .

إختلاف آخر أن مكان عمل التدريب عادةً ما يركز بقوة على اللغة التي يحتاجها العمل . فليس من المرجح أن يتعلم أحد أكثر من لغة في وقت واحد. ربما تمنح الإختيار , فكل الإختيارات ستكون لغات واقعية ذات مغزى تجاري .

الوقت الذي سياخذه التدريب سيكون أيضاً قصير , قصير جداً , ولكن هذا محتمل لأن التدريب سيكون أكثر تخصصاً لأداء مهام محددة , ستغطي في البداية الأساسيات التي تحتاجها, ولكن فيما بعد ستغطي عناصر البرمجة على حسب الحاجة , و يسمح هذا للتدريب أن يتسلسل بشكل جيد مما يعني أنك ستحصل على خليط جيد من الخبرة النظرية والعملية.

وستزود بالكثير من المواد العلمية التي تحتاجها , مثال :

- ☐ Texts (النصوص)
- ☐ Software (البرمجيات)
- ☐ Sample code (نماذج الكود)
- ☐ Support (الدعم)

وإذا أعطيت حق الدخول إلى موارد الإنترنت المجانية أو المدفوعة للمنتديات , فإستخدامها جيداً إن أتيح لك — فيمكن ان يكون هذا بمثابة مورد عظيم لتعزيز مهنتك البرمجية , ويمنحك سبق هائل .

Tips نصائح

- إذا كان لديك الفرصة لتعلم لغات أخرى , قم بالبحث أولاً ثم اختر أيهم أكثر إهتماماً ومنفعة لك .
- إذا منحت الدخول إلى نظام " التعلم بالسرعة التي تشاء " , إستخدمه بالطريقة التي لا تشعر بك بإنك وصلت للذروة .
- إحفظ كل المواد العلمية التي تعطى لك (كتب , برمجيات , ملاحظات) وصنفهم لديك , لأن في هذا تعزيز أكبر لمهاراتك .
- إحفظ بكل ملفات صادر الكود التي تنتجها . و ستكون طريقة عظيمة لتحكم على تقدمك في وقت لاحق.

Hobby Programmer مبرمج الهواى

المبرمجين الهواة هم مبرمجون محظوظون لأنهم يجدون التحفيز من أنفسهم ولرغبتهم في التعلم , هذا يعنى انهم عادةً لن يقفوا تحت ضغط الوقت أو ضغط الإنتاج , فهذا يهدأ الحالة في واحدة من أفضل الحالات للتعلم وإستيعاب المعلومات وإحراز تقدم . المبرمجون الهواة أيضاً , بشكل غريب , أكثر تحفيزاً للتعلم و تجربة أشياء جديدة أكثر من التي تعلموها في بيئة أكثر تحكماً .

وهناك عيوب أيضاً , يجب أن تزود بكل المواد العلمية , التي ستكلفك جزءاً من المال .و سيكون عليك بناء وقت تعليمك على كل شئ تفعله . وهذا يعنى أحياناً أنك لا تحصل على ماتريد في جميع جوانب البرمجة.

كمبرمج هاوى , يمكنك ان تعمل في البرمجة كلما أحببت , أو تستغرق وقتاً أقل للأشياء التي تريد . يمكن أيضاً أن تصنع إختيارات على ما تريد أن تتعلم , وليس هناك ضغط عليك لتعلم كل شئ او في ترتيب معين . فتحتاج ان يتم تغطية الأساسيات مبدئياً , لذلك تكون بمكانة البرمجة الفعلية . ولكن فقط بعد أن يكون لديك الاحترافية لتعلم ما يهكم .

Tip نصائح

- حاول أن تحصل على الممارسات البرمجية . مبدئياً يومياً — فكلما مارست أكثر كلما تحسنت معرفتك البرمجية وسيكون ذلك أسرع في إحراز تقدم .
- البرنامج رشقات نارية في البداية — لاتزيد عن ساعة .
- إذا وجدت مشكلة أصابتك بالإحباط .إبتعد عنها لفترة ثم عد إليها بعد ذلك , وربما تجد ان إيجاد فاصل قصير فقط من ما تفعل قد يحسن تركيزك ومهاراتك .
- إحفظ بملاحظات لما تفعل وأين تفعل . وهنا لا شئ أسوء من نسيان أين أنت .
- إذا وجدت أن الأمور أصبحت صعبة , النصيحة العظيمة هي التراجع قليلاً والذهاب إلى الخلفية المألوفة مرة أخرى قبل الإنتقال إلى مساحات جديدة .

The Languages اللغات

إذا كنت جديد للبرمجة , إذا فأول شيء تحتاجه هو إختيار لغة تبدأ لتعلمها , هناك لغات متعددة مناسبة تماماً للتعلم . اللغات التي سنستخدمها في هذا الكتاب هي :

- ❑ C++
- ❑ Java
- ❑ VBScript
- ❑ JavaScript

فقد تم إختيارهم لأسباب متنوعة :-

- يمكن إستخدامهم في العالم الواقعي .
- سهولة تعلمهم .
- يمكن إستخدامهم لتوضيح مفاهيم البرمجة .
- يمكن تعلمهم من خلال أدوات مجانية .

How I Will Teach You to Program كيف سأعلمك البرمجة

قبل أن نذهب أبعد من ذلك , دعنا نلقى نظرة على ما سنتعلمه في صفحات هذا الكتاب .

- **الإساسيات** : هذا الكتاب يعلمك مبادئ البرمجة , وهذه المبادئ هي أساسيات المعرفة التي تحتاجها ولا يهم أي لغة قد إخترتها, ولا يهم أيضاً ماذا تريد ان تفعل . فهذا معظمه سيكون مطابق لأي لغة قمت بإختيارها لتستخدمها .
- **مبادئ اللغات المتنوعة** : يغطي هذا الكتاب مبادئ هذه اللغات , سيعطيك السبق في إستخدام اللغات والتطبيقات .
- **أمثلة تطبيقات** : في عملية تعلم البرمجة , سننشأ بعض الأكواد الشيقة التي يمكن أن تستخدمها وتعديل فيها لمشاريع اللاحقة .

هناك بعض الأشياء التي لن نقوم بتغطيتها , هذا الكتاب ليس الدليل الكامل لـ C++ أو كل شيء تريد أن تعرفه عن Java — فلو كان كذلك , لكان هذا الكتاب كبير جداً! وأيضاً , كمبتدئ للبرمجة , لا تحتاج إلى هذا النوع من المعلومات إلى الان .

Why Not Buy a Book Covering a Specific Language? لما لا تشتري كتاب يغطي لغة محددہ ؟

تقريباً كل الكتب الموجودة في السوق التي تتحدث على لغة معينة لا تناسب تماماً المبتدئين .

لماذا ؟ مشكلة الكتب التي تقوم بتغطية لغة برمجة معينة تفترض أنك بالفعل مبرمج ولا تحتاج إلى أي تغطية للمبادئ . تلك الكتب التي تغطي مبادئ فقط في شكل تمهيدي ولا تمنحك تغطية كافية لكثير مما تحتاجه لخلق أساس متين .

المشكلة الأخرى مع النصوص والكتيبات التي تغطي لغات برمجة محددة , هي انها , في أغلب الأحيان , تغطي لغات تجاريه , و يعني هذا انك قد تحتاج أن تنفق المال في حقبة تجارية مرتفعه الثمن و بدوره فهو أمر شاق على المتعلم .

كن حذراً من شراء الكتب التى لديها " *learning edition* " [طبعة للتعليم] نسخ من تطبيقات البرمجيات مجمعة فى إسطوانة CD — معظم حزم هذه البرمجيات معطلة بطريقة ما ولا تقدم لك الأدوات التى تحتاجها للقيام بالبرمجة المناسبة .

The Tools

الأدوات

دعنا الآن نلقى نظرة على الأدوات التى تحتاجها لتبدأ تعلم البرمجة !

سوف نقسم الادوات إلى فئتين إثنيتين مريحتين :

- الادوات والخدمات العامة .
- أدوات برمجة معينة .

General Tools and Utilities

الأدوات والخدمات العامة

هناك بعض الأدوات التى تجعل البرمجة أسهل , وقد أدرجتها فى الأقسام الفرعية التالية :

Text Editor

محرر النصوص

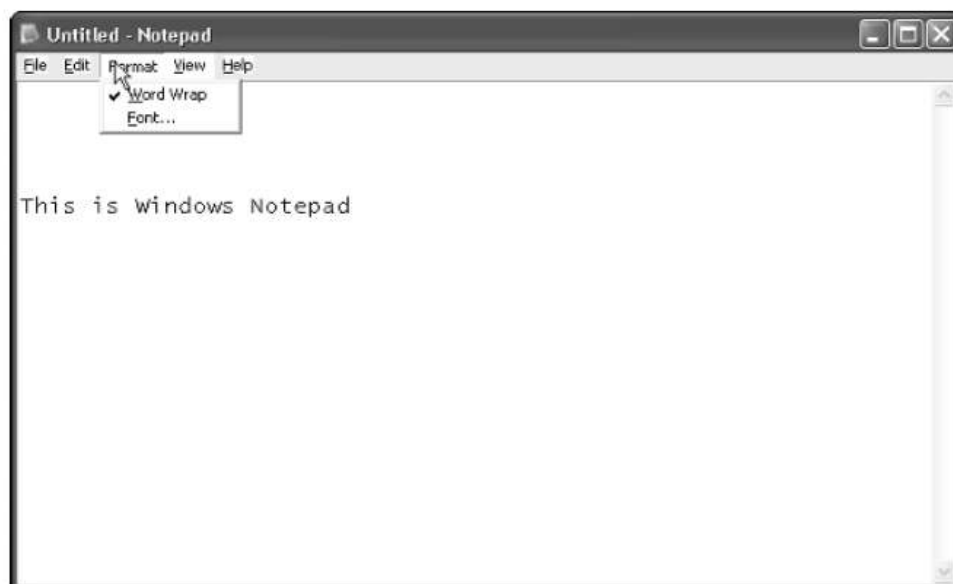
محرر النصوص هو أداة يدوية وذلك لأنك يمكنك أن تستخدمه للغات برمجة متنوعة مختلفة . هناك الكثير لتختار منهم (فمنها المجانى والتجارى) , وأوصى بالتالى :

- ☐ Windows Notepad (free) مجانى
- ☐ UltrEdit (commercial) تجارى

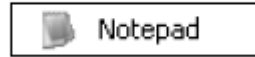
Windows Notepad

[برنامج محرر نصوص ملحق بنظام التشغيل Windows]

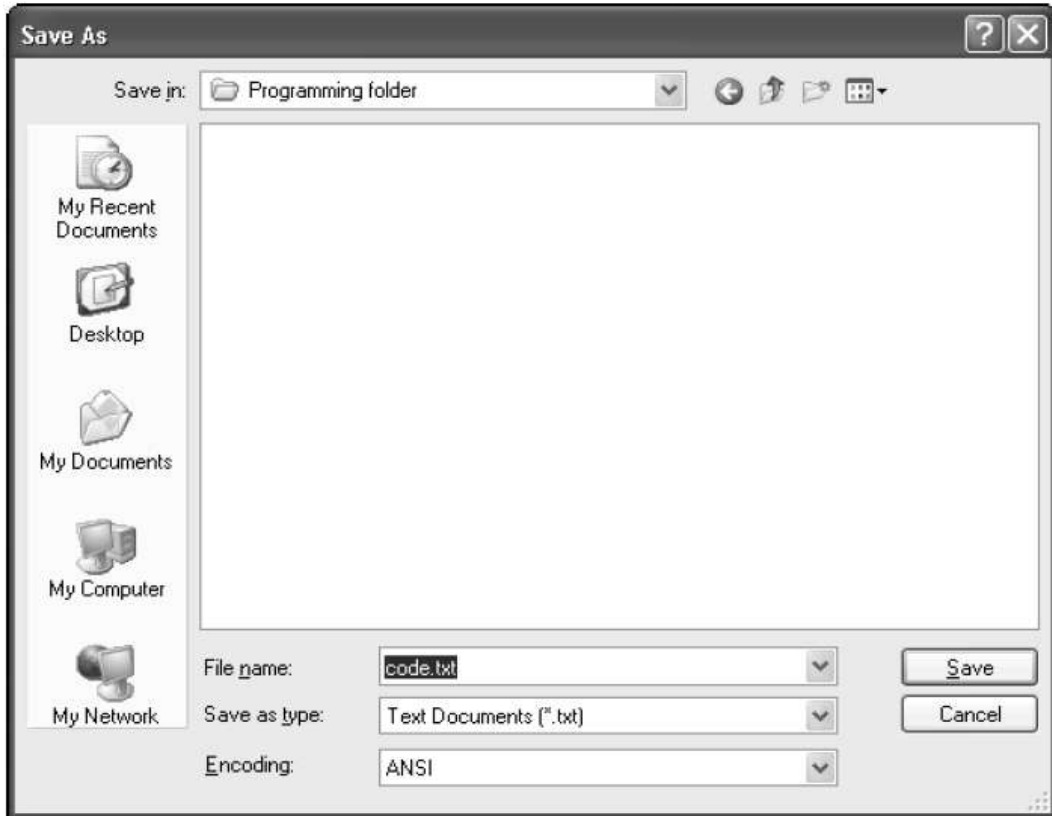
هذا هو محرر النصوص القياسى الذى يأتى مثبتاً و ملحق بنظام التشغيل Windows (كل إصدار من الـ windows يشتمل على إصدار من Notepad (كما يعرض بالشكل) .



لإيجاد **NotePad** اضغط قائمة **Start** ثم **All Programs** (**windows XP** — فى أنظمة تشغيل
اخرى اضغط **Programs**) ثم اضغط على **Accessories** ثم أيقونة **Notepad** (كما يظهر بالشكل)



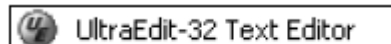
NotePad هو محرر نصوص بدائى بلا أى تسهيلات برمجية خاصة , وليس لديه ميزات خاصة صممت
للبرمجة , ومع ذلك فـ **NotePad** كان راسخاً ومفضلاً للمبرمجين لسنوات . لتسهيل الحفظ الأساسى (كما
يظهر بالشكل)



- السعر : مجانى .
- الإتاحة : مجاناً مع أنظمة التشغيل windows من مايكروسوفت .
- المزايا : بدائى جداً .

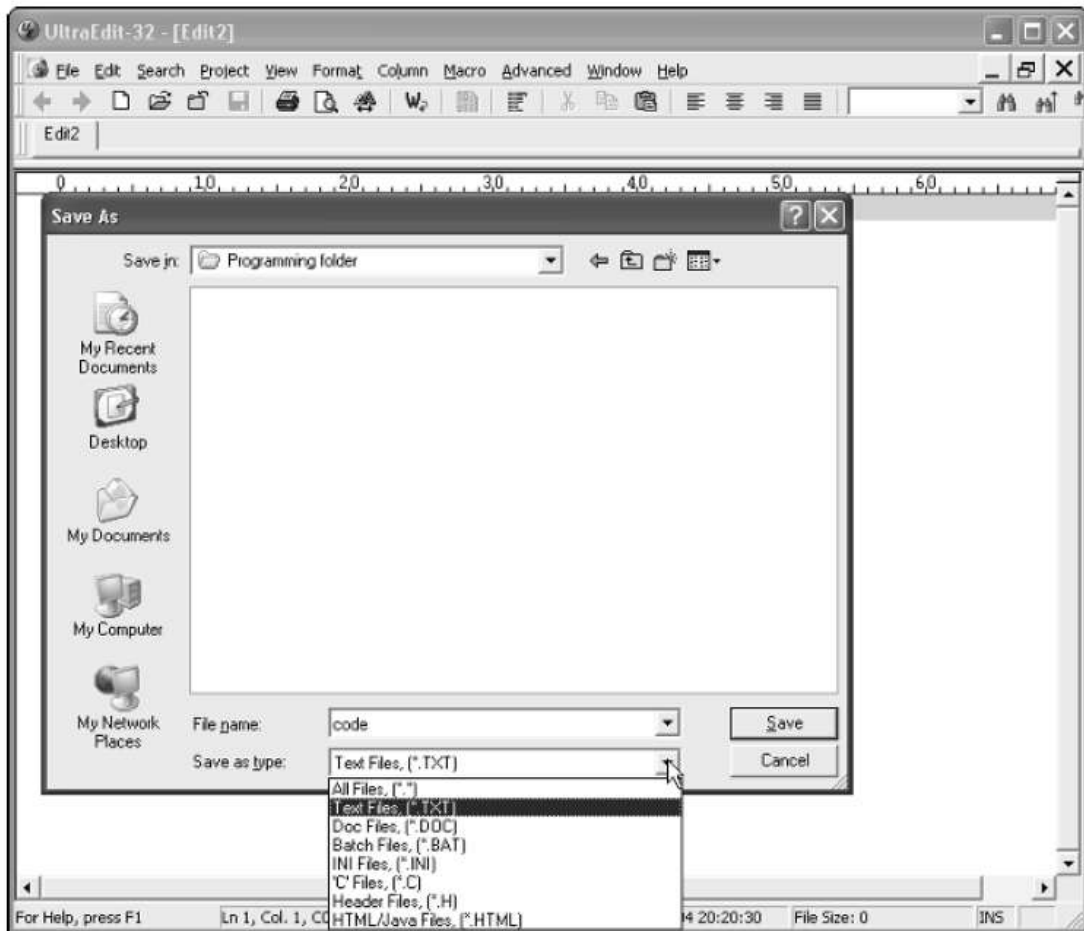
UltraEdit

فى رأى هو أفضل محرر نصوص متاح فى الأسواق , (معروض فى الشكل التالى)



ليس مجاناً , ولكن يأتى كعلبة معبأة بالمزايا المحملة التى تعد مثالية للمبرمجين — مزايا مثال :

- ☐ Syntax highlighting (تسليط الضوء على تركيب الجملة)
- ☐ Line numbering (ترقيم الأسطر)
- ☐ Advanced save features (مزايا حفظ متقدمة) (إنظر الشكل)



التكلفة : \$ 35 .

الإتاحة : www.ultraedit.com

المزايا : كامل المزايا

Utilities المرافق أو الخدمات

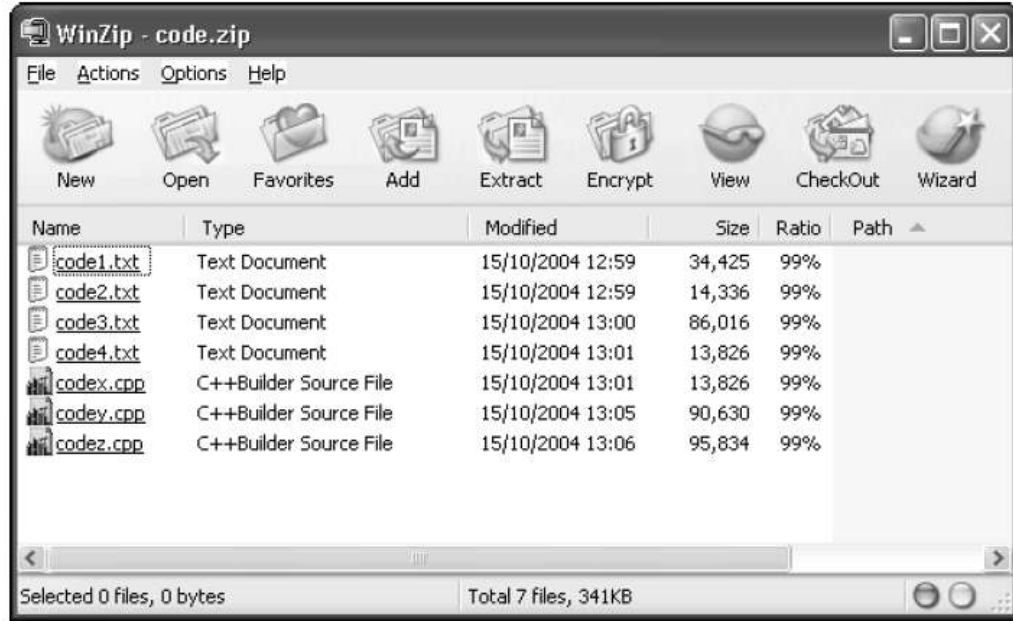
هي ادوات تساعد في جعل وظيفتك أسرع وأسهل , فليس لهم صلة مباشرة بالبرمجة ولكن تجعل وقت البرمجة الفعلي أسرع وأسهل , وتمنعك من القتال مع نظامك .

Winzip

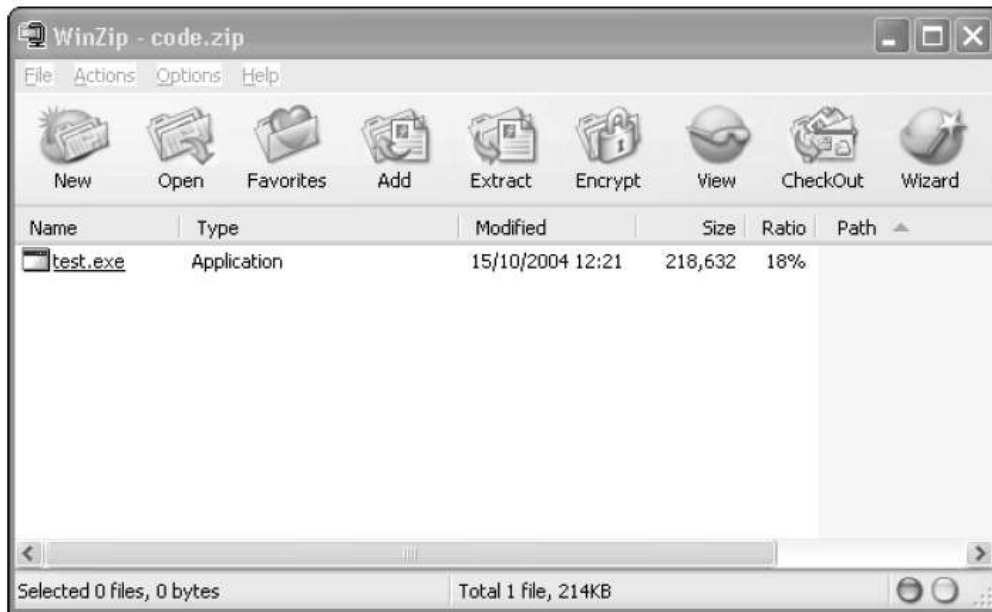
[برنامج لضغط الملفات ومعنى ضغط أى تقليل المساحة]

يمكن ان ينتج المبرمج اكثر من ملف , ويمكن ان يبدأ بنائهم سريعاً ويصبح من الصعب التصفح والتخزين .

أداة الضغط يمكن ان تساعد في تخزين كل الملفات المتعلقة بمشروع معين (أو جانب من مشروع) كلهم في مكان واحد , وأفضل أداة ضغط متاحة هي winzip (كما يظهر في الشكل)



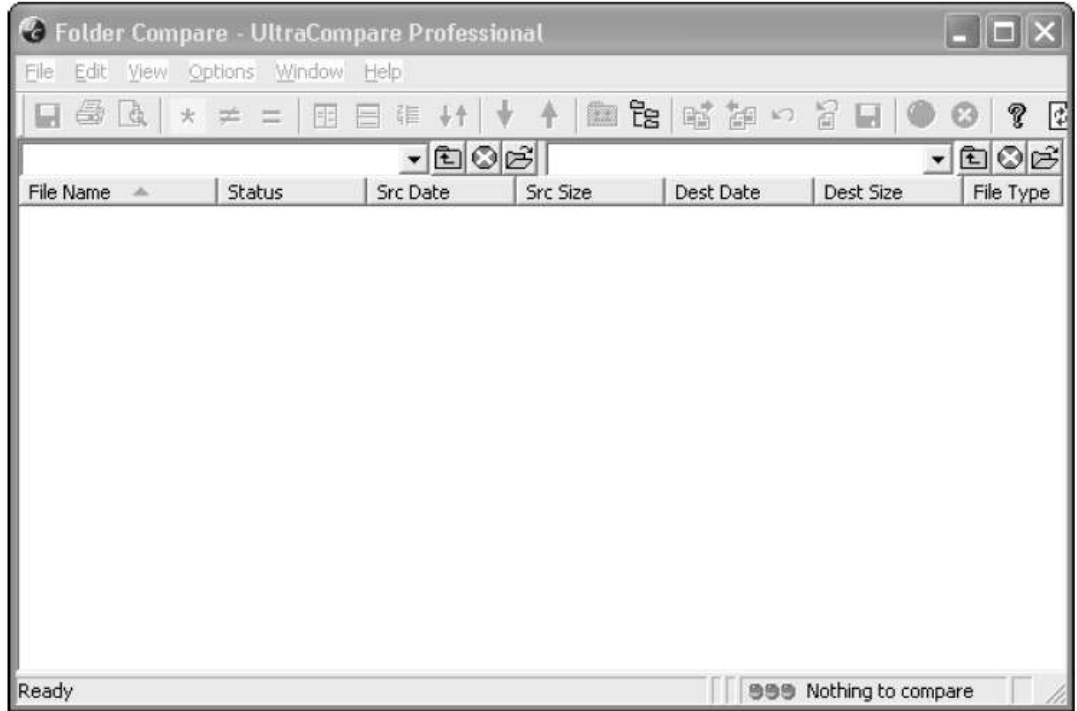
ضغط ملفات المشروع في ملف مضغوط واحد ميزة هائلة , تقلل الفوضى — فكل الملفات التي تحتاجها يمكن أن تجدها في مكان واحد , مما يجعل العمل أكثر رتابة . وبهذه الطريقة يمكنك أن تحفظ كل الملفات التي تنشأها , وتحفظ التغيرات التي أجريتها إلى الكود على طول الطريق . فإن الملفات المضغوطة أيضاً (واضح نوعاً ما) تحفظ الكثير من مساحة القرص , مثال , الملف المضغوط zip . المفصل لديه تقريباً 315 kb , ولكن الضغط يجعله ينكمش إلى ما لا يصدق 2kb , ضغط النصوص لا يصدق ولكن هناك أيضاً حفظ مساحة عظيمة على القرص الصلب من إختبار ضغط الملفات التي أنشأتها على طول الطريق , وهنا 215 Kb تنفيذ تم ضغطه نزولاً إلى 170kb (إنظر الشكل) . ليس جيداً أن تقتصد في الضغط الذي حققته ومن ثم ضغط مصدر الكود , ولكن يظل حافظاً للمساحة .



- السعر : \$ 29 .
- الإتاحة : www.winzip.com
- المزايا : كامل المزايا

UltraCompare

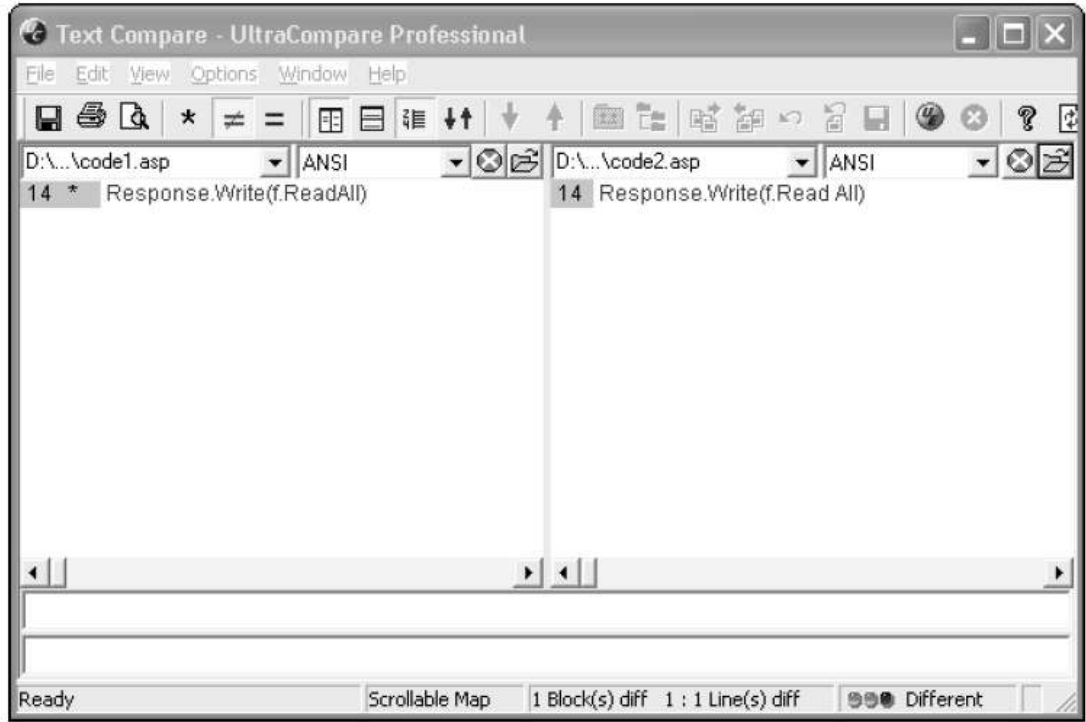
عندما تتعامل مع المزيد من الملفات المختلفة التي تحتوى على كود فبينما يوجد إختلاف بسيط بين واحد وآخر . عندها يمكن أن يكون فى متناول اليد أن تقوم بفحص الملفات سريعاً تحسباً للتغيرات . فوجدت اداه واحدة مناسبة لفعل هذا وهى **UltraCompare** (من صنع محرر النصوص UltraEdit) ويظهر فى الشكل التالى



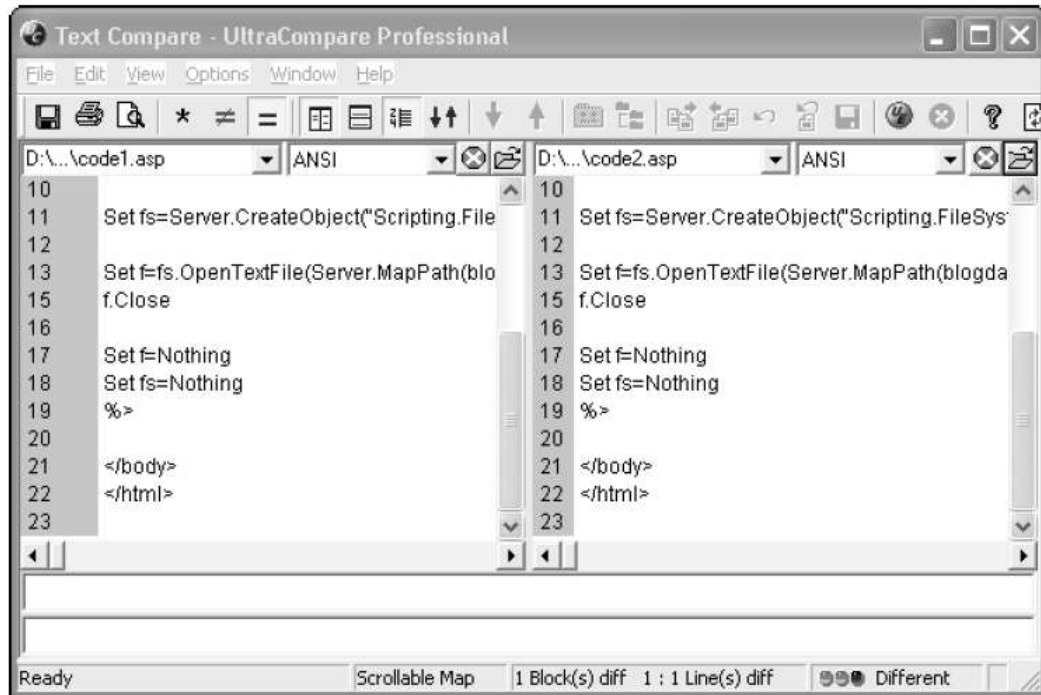
UltraCompare لديها العديد من مزايا المقارنة المفيدة :

- ☐ **File compare.** تفحص الإختلاف فى الملفات
- ☐ **Folder compare.** تفحص الإختلاف فى الملفات المحتواه فى مجلدات
- ☐ **Binary compare.** تفحص ملفات النظام الثنائى من التغيرات

كونك قادراً على إكتشاف التغيرات فى الملفات , كما يظهر فى الشكل , فيعد هذا مفيداً عندما تبحث عن التغيرات الحاصلة, ويمكن أن يكون ذو قيمة عالية جداً عندما تبحث عن الأخطاء الذى تسببت تجاه الكود .



UltraCompare تمكنك من إكتشاف إختلافات متعددة بين الملفات , وحتى للبحث عن خطوط فقط مماثلة في الملفات وتجاهل التغييرات . (إنظر الشكل)



- السعر : \$ 30 (نسخة تجريبية متاحة للتنزيل)

- الإتاحة : www.ultraedit.com

- المزايا : مقارنة الملفات والمجلدات وملفات النظام الثنائي binary

Snagit

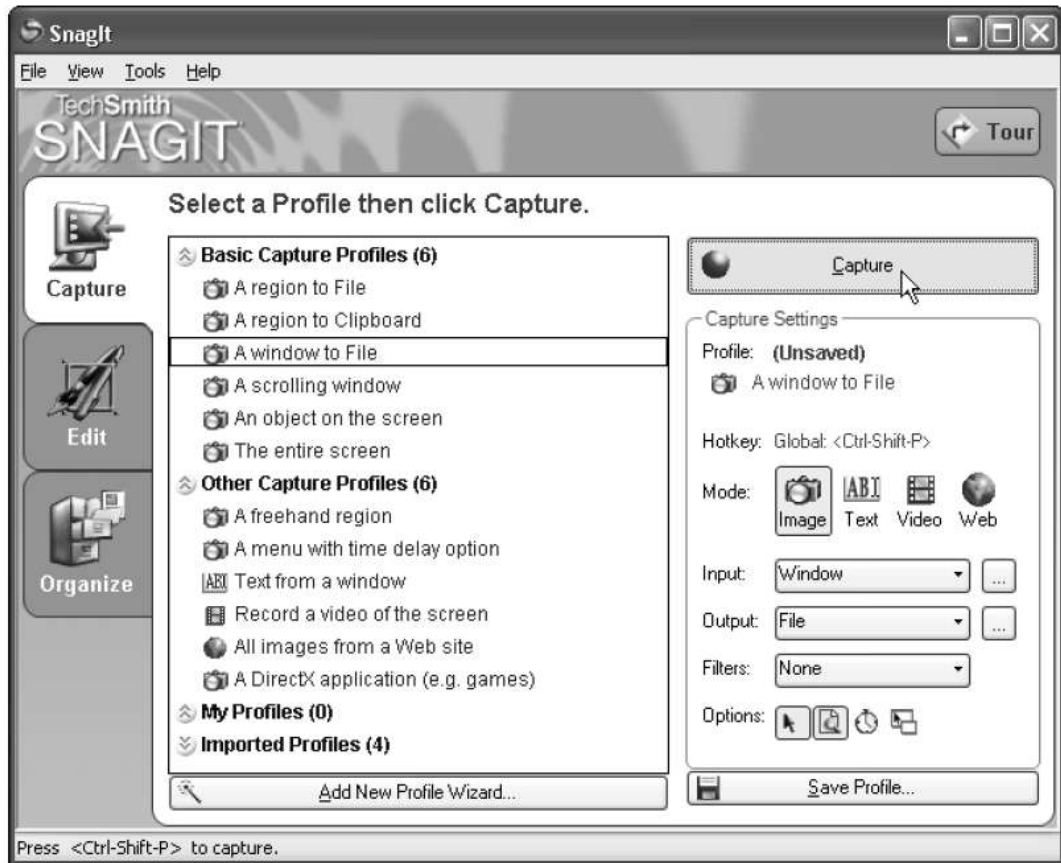
برنامج إلتقاط صور

أحياناً يكون من المفيد حقاً ان تأخذ لقطة للشاشة لما تفعله وتحفظه , وذلك لأنك قد تستخدمها في تجهيز ملفات المساعدة لمن يستخدمون تطبيقك , أو ربما فقط لتسجيل التقدم للتطبيق , أياً كان السبب , فالقدرة على أخذ الصور بإحتراف يمكن ان يكون شئ رائع .

أحد أفضل الأدوات لأخذ صورة هي Snagit بواسطة TechSmith ويمكن ان يرى في الشكل التالي , ويمكنك هذا من أخذ صور بجودة إحترافية تساعدك في التوثيق أو في نشرها على الإنترنت .

هذا البرنامج يسمح بأخذ نوع من الأنواع المختلفة للصور واللقطات .

- ☐ Full-screen capture
- ☐ Windows
- ☐ Menus
- ☐ Screen objects
- ☐ Specific regions
- ☐ Capture text
- ☐ Capture video



يمكنك بعدها أن تخرج هذه الصور في تنسيقات ملفات متنوعة .

السعر : 39.95 (نسخة تجريبية متاحة للتنزيل)

الإتاحة : www.snagit.com

المزايا : كامل الوظائف لإمكانية قص الشاشة والمرئيات .

Programming Tools

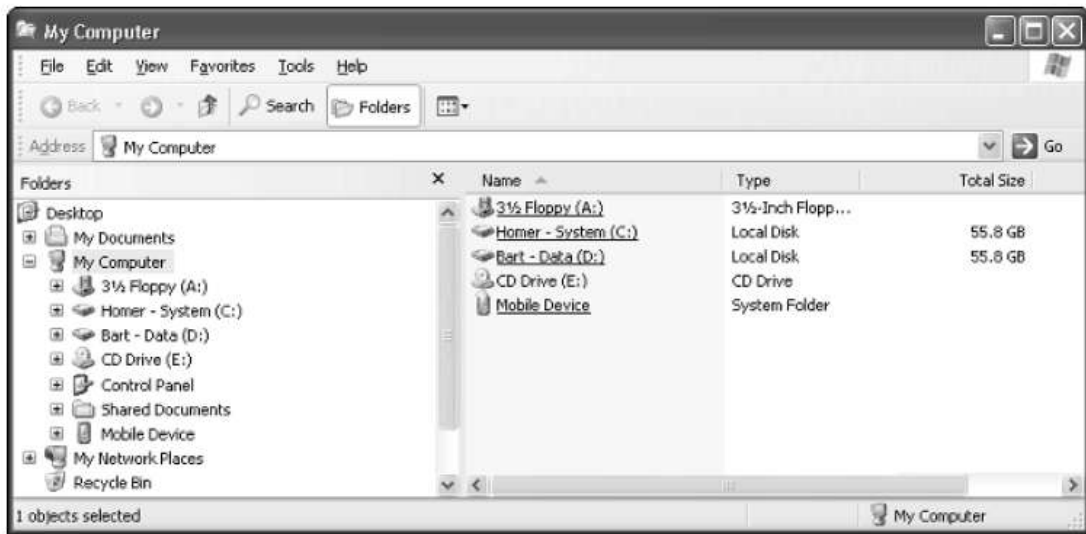
أدوات البرمجة

حان الوقت الآن أن نتكلم عن أدوات البرمجة . فأدوات البرمجة هي الأدوات الفعلية التي ستستخدمها لتكتب , أو تنشأ , أو تنفذ كود وسوف نستخدمها بقدر كبير مع ما تبقى من الكتاب .

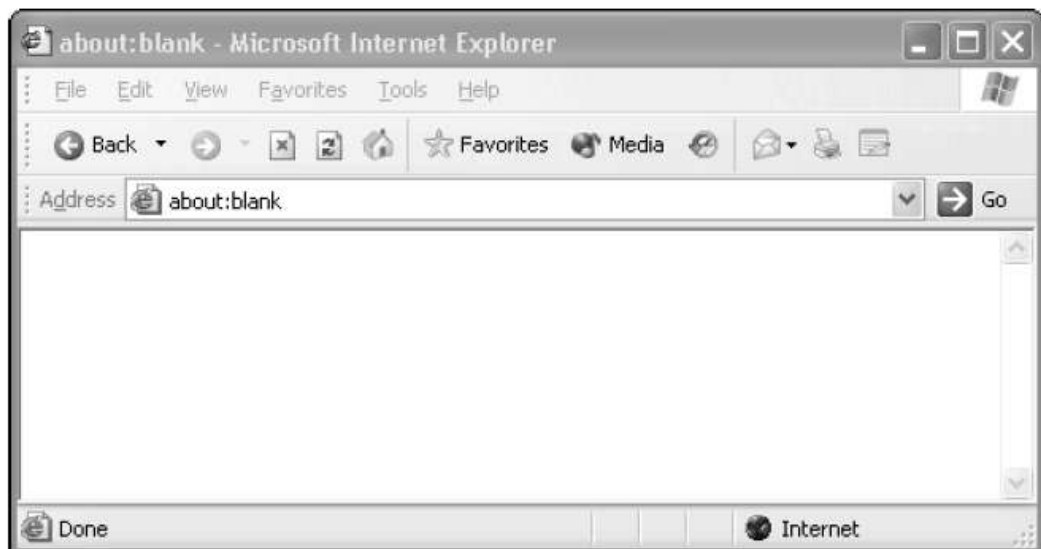
بعض الأدوات التي تحتاجها هي مجرد ادوات تستخدمها لتنفيذ الكود الذي أنشأته في محرر النصوص . بينما الآخرين مخصصين للغة التي تستخدمها .

إنطلاقاً من هذه النقطة , يفترض هذا الكتاب أن لديك ثقة إلى حد ما في العمل مع نظام التشغيل Windows , فسوف نقوم بإستخدام الكثير من مكونات نظام التشغيل windows . في العادة :

❑ Windows Explorer (يظهر في الشكل)



❑ Internet Explorer , يظهر بالشكل



❑ The command window تظهر بالشكل

```
Command Prompt

D:\Programming folder>dir/p
Volume in drive D is Bart - Data
Volume Serial Number is C879-C57D

Directory of D:\Programming folder

15/10/2004  13:50    <DIR>        -
15/10/2004  13:50    <DIR>        -
10/06/2004  12:17                302 code1.asp
15/10/2004  13:49                303 code2.asp
                2 File(s)          605 bytes
                2 Dir(s)  50,530,238,464 bytes free

D:\Programming folder>
```

دعنا نلقى نظرة على الأدوات التي سنستخدمها .

Java

Java هي لغة برمجة طورت بواسطة Sun Microsystems في 1995 وقد صممت في المقام الأول لإستخدامها في الإنترنت . إنشأت java تعمداً لتكون مشابهة للغة C++ ولكن أسهل للتعلم والإستخدام من C++ ويمكن ان تستخدم java لتنشأ تطبيقات كاملة الوظائف لتعمل على حاسب واحد أو توزع على أكثر من خادم وحاسب على نظام الشبكة .

القوة الرئيسية لـ java أن تعرف بإنها لغة بيئة مستقلة , يعنى هذا انها صممت خصيصاً لتعمل مع أنواع الأنظمة المختلفة (PC , Macintosh كمثيل) وأنظمة التشغيل المختلفة , فالكود بنفسه يسمى ByteCode , ولا يعتمد الكود على جوانب محدده من اجزاء الحاسب الملموسة أو برامج الحاسب — لأن تنفيذ الكود يتم التحكم فيه من خلال بيئة تنفيذ خاصة .

لتبدأ البرمجة في java تحتاج إلى شينين :

- **The Java Software Development Kit (SDK)** : يستخدم هذا لتطوير تطبيقات الـ java وبه أيضاً أدوات ومعلومات تحتاجها . وملف تحميله يسمى J2SE (Java 2, Special Edition) ويظهر في الشكل



- **Java Virtual Machine : The Java Virtual Machine (VM) runtime environment**
يتطلب أن يكون موجوداً على كل جهاز سينفذ كود الـ Java. وملف تنزيله يسمى Java VM (يشتمل هذا على SDK, ولكن أي جهاز سينفذ الكود الخاص بك ستحتاج أن يكون مثبتاً)



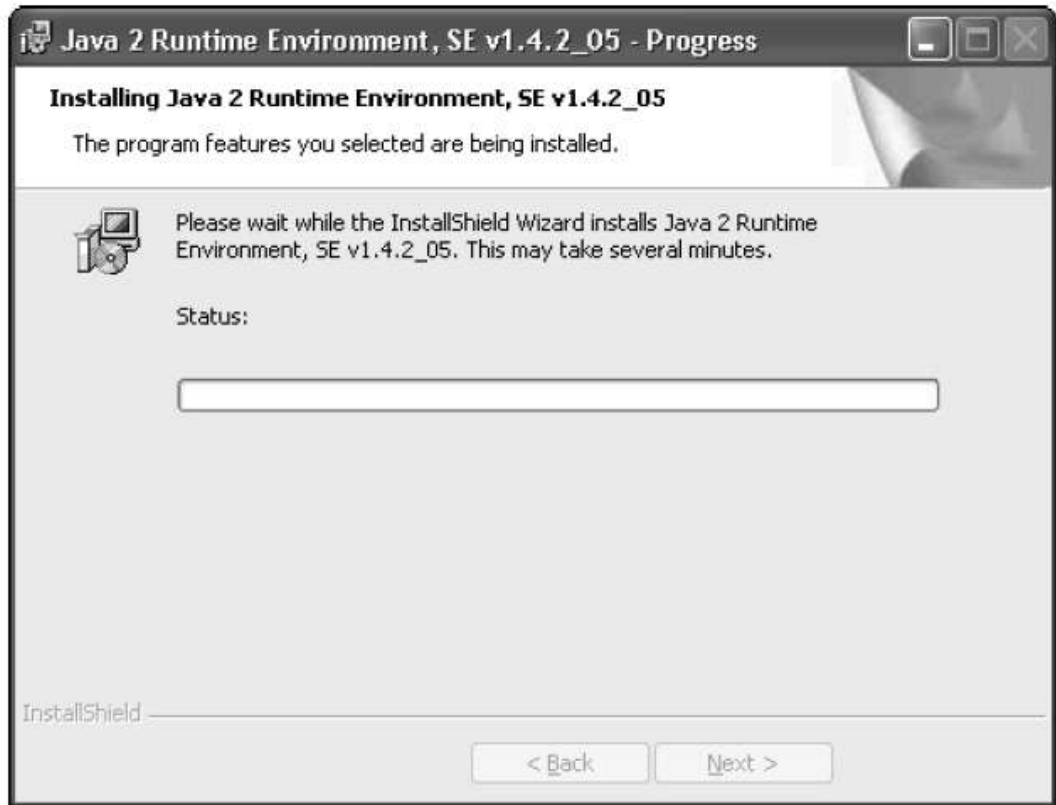
لا يأتي نظام التشغيل Windows XP مثبتاً معه Java VM ولكن كل إصدارات الـ windows تأتي بذلك .
كلاهما متاح للتنزيل من الموقع الرسمي لـ Java الذي يمكن زيارته من هنا :

<http://java.sun.com/j2se>

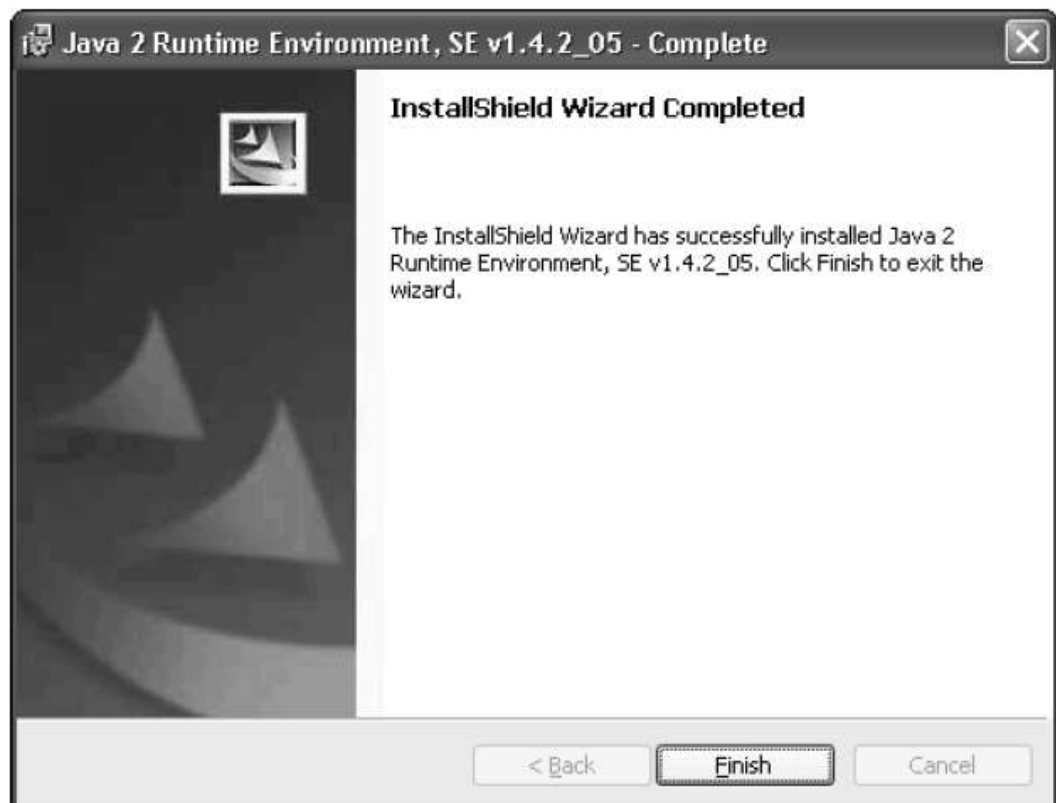
تثبيت ملفات هذه التنزيلات سهلة , الضغط مرتين عليهم , كما يظهر بالشكل



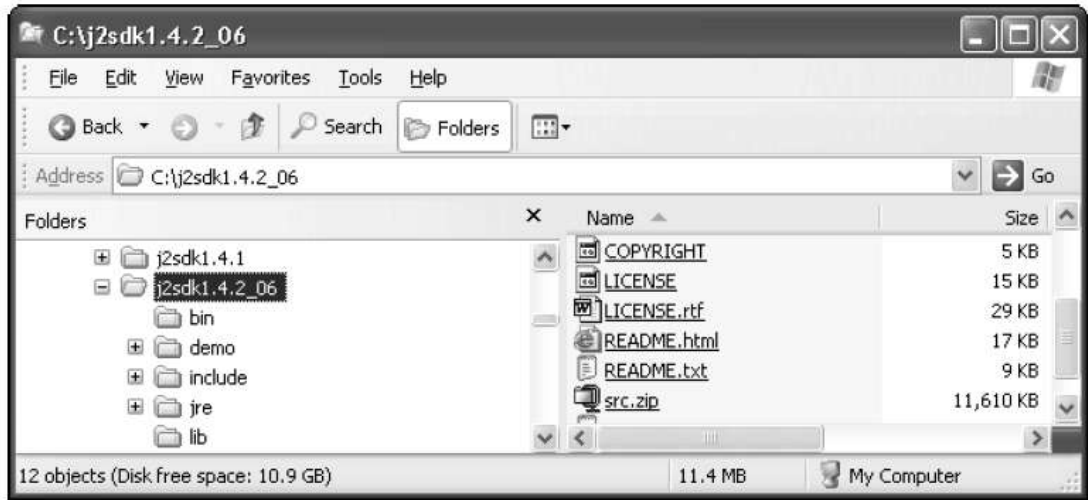
بعدها توافق على الترخيص وتختار أين تثبته على حاسبك الشخصي . سيتم التثبيت بعدها , كما في الشكل :



ربما يأخذ التنصيب الكثير من الدقائق ويتم بتنصيب بيئة التنفيذ VM , كما يظهر بالشكل :



بمجرد أن يثبت البرنامج , إستخدم مسكتشف الـ Windows Explorer لتبحر داخل مسار تثبيت البرنامج (الإفترضى هو c:\j2sdk1.4.2_06) لتفحص ما إذا كان قد تم تثبيته كما بالشكل



بمجرد ان يثبت هذا ليكون جاهز ليستخدم مؤخراً .

C++ لغة برمجة

الكثير من الناس يعتقدون أن كونك قادراً على البرمجة بإستخدام C++ فأنت تحتاج تطبيقات مرتفعة الثمن — مفسر أو مترجم اكواد وبيئات تطوير . هذا ببساطة ليس صحيحاً , فأحد الآثار الجانبية الرئيسية للغة لكى تصبح شعبية ويتم إستعمالها كلغة أكاديمية هى وجود وفرة من الأدوات المجانية التى تتشكل حولها — فقد تم إنشائها بواسطة المعلمين والأساتذة للإستخدام الأكاديمى .

وقبل أن نذهب أبعد من هذا , كلمة واحدة عن *academic use* : عندما نقول أن برنامج او تطبيق ما للإستخدام الأكاديمى يعنى هذا أنه يمكن أن يستخدم مجاناً ولإغراض تعليمية — لذلك فإن اداة البرمجة أو المفسر يوصف من هذا القبيل أن يتم إستخدامهم بحرية خلال سياق التعلم . وهذه الأدوات يمكن ان تستخدم بواسطة الطلاب و الأساتذة على حد سواء لزيادة معرفتهم فى هذا الموضوع .

ولكن هناك حدود لما يمكن أن تفعله . هاهى بعض الحدود الشائعة موجودة فى البرنامج الأكاديمى بانها :

- برمجيات لا يمكن أن تستخدمها لأغراض تجاريه : ما يعنى عادةً انه مسموح لك بإنشاء التطبيقات بإستخدام الأدوات ولكن محرم عليك بيع أى شئ انشأته , وبشكل عام التخلّى عن ما انشأته قانونى .
- لا يوجد دعم للأدوات : ربما يكون هناك ترتيب من أنظمة الدعم الرسمية فى المكان ولكن توقع أن تحصل على ما تدفع له — فى هذا الحالة , لا شئ .
- على مسئوليتك الخاصة : الجملة المشتركة مع جميع البرامج فى الوقت الحاضر , ولكن مرة اخرى توقع أن يطبق ذلك على نحو مضاعف لبرامج الأكاديمية الحرة .

توقع أن يكون التوثيق بدائى , ليس دائماً نفس الحالة ولكن غالباً ما يكون صحيحاً .

لتكون واثقاً تماماً من حقوقك , إفحص إتفاقية الترخيص المصاحبة للتطبيق , وإذا كنت فى أى شك لما يمكنك فعله وما لا يمكنك , إذا إتصل بالشركة أو المؤلف للتطبيق .

إبقى قانونى !

الآن مع الخروج من الطريق , دعنا نلقى نظرة على بعض اداوت C++ المتاحة .

Borland C++ compiler

أحد أفضل مفسرات الأكواد المتاحة المجانية هو Borland . فهو متاح مجاناً كمفسر أسطر الأوامر من Borland C++ Builder application .

التحميل متاح حالياً على :

www.borland.com/products/downloads/download_cbuilder.html

الإصدار الحالي هو إصدار 5.5 وحجم التنزيل هو 8.7MB .

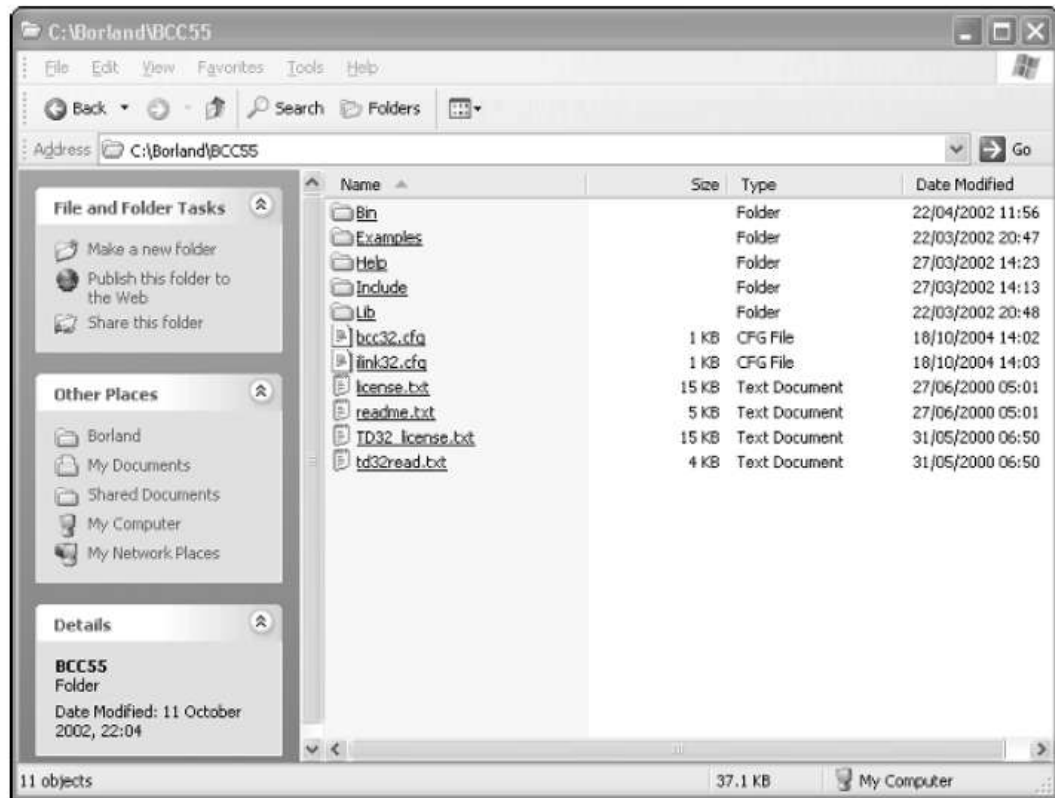
تنزيل آخر إصدار للبرنامج (كما يظهر بالشكل) من الموقع .



بمجرد ان تنزل الحزمة , قم بالضغط مرتين لتقوم بتشغيله وتبدأ عملية التثبيت , كما يعرض بالشكل



الموقع الافتراضى لتنصيب التطبيق هو C:\Borland\BCC55 (كما بالشكل) إذهب إلى هذا المسار إذا لم يكن لديك إختيار



لتجعل البرنامج يعمل فعليك بإنشاء ملفين Configuration جديدين. لا تقلق , لا يجب ان تفعل أى شئ معقد .

الملفين هما :

bcc32.cfg

ilink32.cfg

محتوي هذه الملفات كما يلى , أولاً bcc32.cfg

```
-I*c:\Borland\Bcc55\include*  
-L*c:\Borland\Bcc55\lib*
```

والثانى , ilink32.cfg:

```
-L*c:\Borland\Bcc55\lib*
```

إنشأ هذه الملفات فى تطبيق مثل UltraEdit أو Windows NotePad وإحفظهم فى

C:\Borland\BCC55.

هذا التطبيق هو Command Line فقط — ليس هناك مساعدة لكتابة الكود على كل حال , وكل الكود سينشأ فى محرر النصوص . هذا الكود يجهز ويجمع بواسطة التطبيق وقد تم إنشاء ملف تنفيذى . المفسر للأكواد فى أثناء العمل يرى على هذا الشكل

```

Command Prompt
D:\>bcc32
Borland C++ 5.5 for Win32 Copyright (c) 1993, 2000 Borland
Syntax is: BCC32 [ options ] file[s]
-3 * 80386 Instructions -4 * 80486 Instructions
-5 Pentium Instructions -6 Pentium Pro Instructions
-Ax Disable extensions -B Compile via assembly
-C Allow nested comments -Dxxx Define macro
-Exxx Alternate Assembler name -Hxxx Use pre-compiled headers
-Ixxx Include files directory -K Default char is unsigned
-Lxxx Libraries directory -M Generate link map
-N Check stack overflow -Ox Optimizations
-P Force C++ compile -R Produce browser info
-RT * Generate RTTI -S Produce assembly output
-Txxx Set assembler option -Uxxx Undefine macro
-Ux Virtual table control -X Suppress autodep. output
-aN Align on N bytes -b * Treat enums as integers
-c Compile only -d * Merge duplicate strings
-exxx Executable file name -fxx Floating point options
-gN Stop after N warnings -iN Max. identifier length
-jN Stop after N errors -k * Standard stack frame
-lx Set linker option -nxxx Output file directory
-oxxx Object file name -p Pascal calls
-tVxxx Create Windows app -u * Underscores on externs
-v Source level debugging -wxxx Warning control
-xxx Exception handling -y Produce line number info
-zxxx Set segment names
D:\>

```

Scripting Languages

لغات البرمجة الكتابية

هذا المقطع الأخير يعرض سريعاً جولة لما تحتاج إليه عند العمل مع لغات البرمجة مثال vbscript أو JavaScript . بساطة هذه اللغات إن مثلها لا تحتاج معه إلى الكثير لتكون قادراً على العمل معهم .
فالأاساسيات هي :

☐ Text editor. لتنشأ الكود

☐ Compliant Web browser. لتنفيذ الكود

هكذا تكون !

المتصفحات المتوافقة تعتمد على اللغة المستخدمة :

- VBScript : Internet Explorer (يجب أن يعمل على نظام التشغيل windows)
- JavaScript : ما يقرب من جميع المتصفحات , مشتملة على Internet Explorer , NetScape , Mozilla , Opera , Navigator .

Summary

الملخص

في هذا الفصل قمنا بجولة بالأدوات التي تحتاجها لتبدأ العمل مع الكود . ألقيت نظرة على الأدوات التي تحتاجها للعمل مع اللغات المتنوعة التي قمنا بتغطيتها في هذا الكتاب .

أقترح عليك أن تقوم بتنزيل ما تحتاجه الآن لأن ذلك سيحفظ عليك الكثير من الوقت , الجهد والمتاعب في الوقت اللاحق . ويعنى هذا ان كل شئ على ما يرام لتبدأ بكتابة الكود في الحال .

كمستعرض نصوص , أو صى بـ UltraEdit بسبب المزايا المذهلة المتاحة للمبرمج , ولكن ليس هناك متطلبات لتستخدمه ويمكن لك دوماً أن تختار محرر النصوص المثبت عندك في نظام التشغيل أو قم بتحميل أحد الإصدارات المجانية المتاحة الكثيرة . المهم في هذا أنه لابد من وجود واحد وتجد نفسك الراحة في استخدامه .

6

Simple Coding

هذا الفصل ستحصل حقاً على المشاركة في الأكواد الجادة ! فهناك المزيد الذى سنقوم بتغطيته خلال هذا الفصل , ولكن لا تقلق — فحتى لو لم تكن قد قمت بكتابة كود من قبل فسوف نأخذه بسهولة ويسر , وسنقوم بعمل كل شئ خطوة بخطوة .

فى هذا الفصل , سنتعرض للبرمجة ومفاهيم كتابة الكود التالية :

- ❑ Commenting code (كتابة تعليق على الكود)
- ❑ Variables (المتغيرات)
- ❑ Strings (القيم الحرفية)
- ❑ Processing input (معالجة المدخلات)
- ❑ Outputs (المخرجات)
- ❑ Simple math (عملية رياضية بسيطة)

هذا مزيد من المواد نقوم بتغطيتها هنا , لذلك دعنا نبدأ !

Commenting Code

كتابة تعليق على الكود

هو عملية ترك تلميحات على ما يعنيه الكود خلال الكود الفعلى نفسه . هذه التعليقات يمكن أن تكون لك أو لمبرمجين آخرين سيقومون بالإطلاع على الكود الخاص بك . لا يتم قراءة التعليقات بواسطة المفسر أو المترجم وقت تجهيز الكود — فهناك منافع لقراءة البشر الكود .

أمر غريب أن نبدأ هذا الفصل بالحديث عن كتابة أشياء فى الكود التى ليس لها تأثير فيما سيفعله الحاسب بالفعل . فالتعليقات التى تكتبها فقط ذو منفعة (وللعين) للبشر فى مقابل كونها للكمبيوتر .

أحد اهم الأشياء التى تفعلها أثناء البرمجة وهكذا تتضاعف أثناء تعلمك , هى ترك تعليق على الكود .

علامات إنشاء التعليقات ليست فقط لإضافة تعليقات على الكود الخاص بك , لكن هي طريقة عظيمة لإزالة أسطر من الكود والتي قد تسببت في إثارة المتاعب أو لم يعد لها حاجة بها . وذلك دون أن يجب عليك حذفها فعلياً . [يقصد انك يمكن ان تحدد مجموعة الأسطر من الكود وتجعلها كتعليق ويتم هذا بواسطة علامات تضعها في بداية ونهاية هذه الأسطر وستعرفها في وقتها]

يعتمد كيفية ترك تعليق على الكود على اللغة التي تستخدمها . ففي هذا الفصل , سنلقى نظرة على VBScript, JavaScript, والقليل من C++ ولذلك سنرى كيف نترك تعليق في هذه اللغات .

VBScript Comments

VBScript أحد أسهل اللغات للتعلم , وذلك لأن تركيبها مشابهة كثيراً للإنجليزية .

التعليقات في VBScript سهلة , فهي تجعل من استخدام الفاصلة العليا لتنشأ تعليق . لذلك , فإذا كنت تكتب تعليق على الكود سيكون مثل هذا

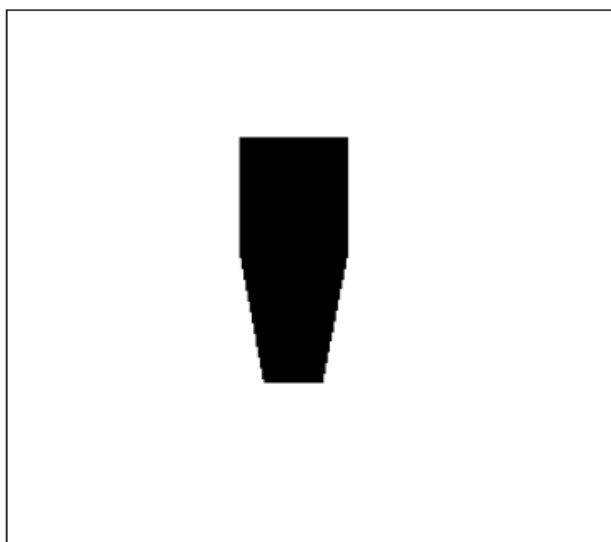
```
' This line is a comment.
' As is this one and the one below.
' I am a comment too!
```

ليس هناك أى تركيبات محددة لما يلي الفاصلة العليا , لذلك كل التعليقات التالية صحيحة :

```
'           Lots of whitespace at the beginning
' comment starts lowercase
' Symbols * ( ) / &f^ ( )
```

[يقصد هنا أنه لا يوجد قواعد لكتابة تعليق غير انك تكتب الفاصلة ثم بعدها تكتب ما شئت وبأى طريقة]

ما تحتاج أن تتأكد منه هو الفاصلة التي تستخدمها في انها الفاصلة البسيطة وليست المتعرجة التي تراها في بعض معالجات النصوص . الشكل التالي يوضح صورة مكبرة لنموذج هذه الفاصلة .



بينما الشكل التالي يظهر الفاصلة المتعرجة خطأ .



نمط آخر من التعليق يمكن إستخدامه وهو امر REM . لـ VBScript هي :

```
REM This is a comment.  
REM It stands for "remark."
```

نحن نسمى REM أمر VBScript بسبب أن كليهما هي والفاصلة بالفعل يفعلون شيئاً ما — فهما يخبران المترجم لتجاهل ما يتبعهم ! وهذا يجعلها أمر اوجملة .

تأتي REM من Remark,., وأعطى هذا ظهور للجمل التي ربما قد سمعت المبرمجون يقولونها مثال :

“That line is remmed out.”

“I remmed out the line of code that was causing problems.”

يمكنك مطابقة الخط بين الفاصلة والـ Remark في الكود الخاص بك , ولن يتسبب هذا في أى مشاكل للجميع .

```
REM This is a remark.  
' This is a comment.  
REM Another remark  
' Another comment
```

نوع واحد من تعليق الكود ينبغي ان يتواجد وهو علامة كتلة التعليق الذي تظهر في بدايه الكود لتعطى تمهيد لما يفعل الكود . نمطين من التعليق يعنى نوعين من كتل علامات التعليق

واحد يستخدم علامة الفاصلة :

```
' Tombstone comments  
' Widget 1.0.2  
' Author: A. N. Other  
' 22-10-04  
'  
' Code starts below
```

وأخرى تستخدم جملة أو أمر REM :

```
REM Tombstone comments
REM Widget 1.0.2
REM Author: A. N. Other
REM 22-10-04
REM
REM Code starts below
```

كما سترى فى وقت لاحق , ستكون قادراً على إضافة تعليقات فى آخر الأسطر التى تحتوى على كود , مثل هذا المثال :

```
x = x + 1 ' This takes x and adds 1 to the value.
```

حتى من الان , لا تقلق كثيراً بسبب ذلك — فكلما تمرنت جيداً فى كتابة الكود , ستحصل على الخبرة فى كتابة التعليق .

Things to Watch For

أشياء للملاحظة

تأكد من أنك تستخدم الفاصلة وليس علامة التنصيص وهما هاتين علامتين " " وتسمى double quotes او فاصلة الربط أو شئ آخر مثال ذلك

```
* This is not a comment.
, This isn't a comment either.
```

تأكد أيضاً ان تجعل مسافة بين الفاصلة أو REM وبين التعليق . وإذا لم تفعل , فلن يتم تجاهل التعليق .

```
'Not a valid comment
REManother invalid comment
```

كلا التاليين هما أيضاً تعليقات صحيحة لأن أول فاصلة أو REM تحول الثانية ببساطة إلى تعليق ليتم تجاهلها

```
REM REM A valid comment
' ' Another valid comment
```

وهذا أيضاً صحيح :

```
* REM valid too
REM * Also valid!
```

فهناك مشكلة شائعة هي ان من كان على دراية بلغة HTML (Hypertext Markup Language) فيحاول أن يستخدم العلامات التى تستخدم فى إنشاء تعليقات HTML (كهذه <!-- -->) بدلاً من هؤلاء الصحاح . فإنه لا يعلم أنه لن يتم عملهم خلال الكود كتعليق وسيكون هذا سبباً فى إحداث عطل .

Quick Exercise

تمرين سريع

العمل خلال هذه الأسئلة فقط لمجرد جعلك تتأكد ان لديك وضوح في إستخدام تعليقات VBScript

ماذا تستخدم ليتم تعريف تعليق في VBScript ؟

اي من التالي ليس تعليق VBScript صحيح ؟

```
'Comment 1
' Comment 2
REM comment 3
REM_Comment4
" Comment 5
REM    Comment 6
' REM Comment 7
REM ' comment 8
<!-- Comment 9 -->
```

هل هذا التعليق صحيح ؟

```
x = x + 1 ' A comment following code
```

JavaScript Comments

تعليقات لغة JavaScript

علامات تعريف التعليق في JavaScript هي علامات بسيطة — في البداية سنلقى نظرة على شرطين مائتين جهة اليمين , كما يلي :

//

فقط ضعهما في أى تعليق تريد أن تنشأه

```
// Comments go here.
```

يمكنك ان تفعل كل هذه الأشياء مع علامات التعليق لـ JavaScript كما يمكنك في VBScript .

ها هي مجموعة من التعليقات :

```
// Comments
// More comments
// Even more
```

يمكنك أن تنشأ كتلة من التعليق أيضاً هكذا :

```
// Tombstone comments
// Widget 1.0.2
// Author: A. N. Other
// 22-10-04
//
// Code starts below.
```

إذا كنت تتوى فعل شئ ما مثل هذا , يستحسن أن تستخدم علامات التعليق متعدد الأسطر :

```
/*
Comments go here.
*/
```

وهى عظمة لبناء كتل التعليق

```
/*
Tombstone comments
Widget 1.0.2
Author: A. N. Other
22-10-04

Code starts below.
*/
```

تذكر ان يكون الترتيب كما يلى :

يفتح بشرطة مائلة ثم علامة النجمة

```
/*
```

ويغلق بعلامة النجمة ثم الشرطة المائلة :

```
*/
```

أضف التعليق بين الإثنين

```
/*
Comments go here.
*/
```

Things to Watch For

أشياء للملاحظة

الأسباب الرئيسية للأعطال أن الناس انفسهم يستخدمون علامات خاطئة [invalid غير صحيحة]

```
\\ Invalid comments!!!!
```

كما إنهم لا يتركون مسافة بين العلامة والتعليق

```
//Invalid comments
```

أو انهم يستخدموا التركيبات الغريبة :

```
/*Invalid!  
/*Also invalid
```

مع تعليقات متعددة الأسطر , تحتاج أن تتأكد من ترتيب علامة الشرطة المائلة والنجمة عند الإفتتاح والإغلاق.

```
*/  
Invalid comments!  
/*
```

Quick Exercise

تمارين سريعة

العمل مع هذه الأسئلة السريعة فقط لجعلك تتأكد من انك قد فهمت تعليقات JavaScript

كيف تقوم بتعريف تعليق في JavaScript ؟

أى من التاليين ليس بتعليق JavaScript صحيح ؟

```
\\ Comment 1  
// Comment 2  
/*  
Comments 3  
*/  
//Comment 4  
/*Comment 5  
/* Comment 6
```

C++ Comments

تعليقات لغة C++

تتبع علامات إنشاء تعليق في C++ نفس القواعد مع JavaScript وتشمل أيضاً تعليق ذو سطر واحد وآخر متعدد الأسطر :

```
// Single line comment
```

و

```
/*  
Multiline comments  
Look like this  
*/
```

تطبق نفس قواعد JavaScript , ويمكنك إستخدام كتل التعليق في نفس الطريقة (في الحقيقة , كتل التعليقات في C++ مهمة جداً وتستخدم على نطاق أوسع)

```
/*
C++ Project
Widget 3.1.9
Author: A. N. Other
22-10-04

Code starts below.
*/
```

Things to Watch For

أشياء للملاحظة

مكان المشكلة في تعليقات C++ هي نفس الشيء بالضبط في JavaScript , الإختلاف الرئيسي هو أنك إن إرتكبت أخطاء سيلتقطها المفسر Compiler بدلاً من المترجم Interpreter , وسيفصل المفسر موقع الخطأ , لكنها قد غامضة بعض الشيء فيما يتعلق بالخطأ الفعلي :

أعطال تعليقات C++ هي :
إستخدام أسطر خطأ

```
\\ This will cause problems!!!!
```

عدم ترك مسافة بين الأسطر والتعليق

```
//This is incorrect!
```

مع التعليقات متعددة الأسطر , تحتاج أن تتأكد من ترتيب الأسطر وعلامة النجمة في الإفتتاح والإغلاق

```
/*
This comment block will
throw errors.
*/
```

Quick Exercise

تمارين سريعة

العمل مع هذه الأسئلة السريعة فقط يجعلك تتأكد من أنك قد فهمت تعليقات C++

صح / خطأ : علامات تعليقات C++ مثل علامات تعليق VbScript ؟

كيف تقوم بتعريف تعليق في C++ ؟

أى من التاليين ليس بتعليق C++ صحيح ؟

```
// Comment 1

\/ Comment 2

/*
Comments 3
*/

*/
Comments 4
/*

* Comment 5

REM Comment 6
```

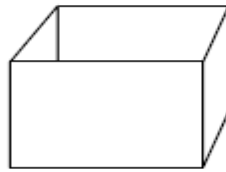
Variables

المتغيرات

المتغيرات هي حجر الزاوية لمعظم لغات البرمجة لأنها تسمح لك بالعمل ومعالجة البيانات .

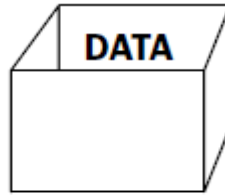
هناك الكثير من التعريفات العائمة عن ماذا يكون المتغير ؟ , البعض معقد جداً , والبعض مبسط جداً . ربما تسمع شئ من هذا القبيل في كونه "موقع في الذاكرة يمكن الإشارة والوصول إليه " , في حين أنه غير دقيق , فهي حقاً طريقة معقدة في تعريف مصطلح .

فأنا أفضل كثيراً في أن أعتقد ان المتغير هو مكان للبيانات له اسم يمكنك الإشارة والرجوع إليه , أنشأت لك تمثيل تصويري في الشكل التالي :



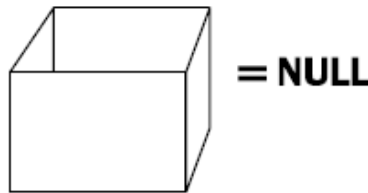
Variable

فتحمل البيانات داخل المتغير ويمكن الوصول إليه ومعالجته من خلال الكود , كما يظهر بالشكل



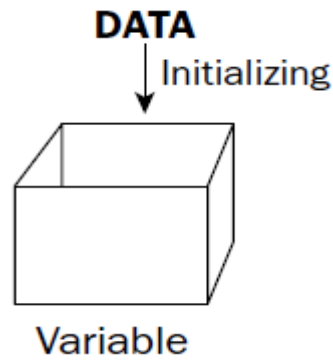
Variable

فارغ Null, عديم الوجود , المتغير (كما هو مبين بالشكل التالي) مثل السلة أو العلبه تنتظر أن تملئ — تنتظر أن تعطى غرض ما .



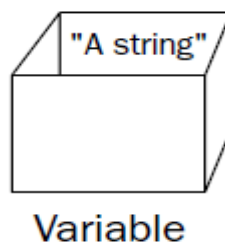
Variable

فعندما تحمل المتغير بقيمة , فيسمى هذا Intializing . [أى إعطاء قيمة ابتدائية للمتغير] كما بالشكل



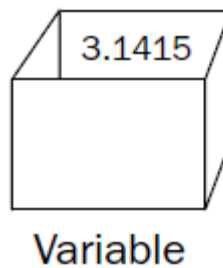
يمكن ان يمسك المتغير بيانات من كل الأنواع , عدد قليل من الأنواع الأكثر شيوعاً لما يمكن أن تمسك به المتغيرات من البيانات هي :

❑ Test strings (القيم الحرفية) (انظر الشكل)

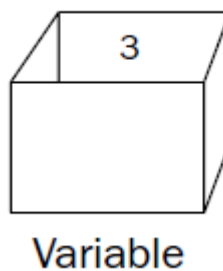


❑ Numbers (الإرقام)

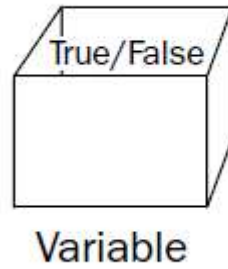
❑ Floating point (انظر الشكل) النقطة العائمة



❑ Integers (انظر الشكل) (الارقام الصحيحة)



❑ Boolean values — (انظر الشكل) True أو False القيم المنطقية

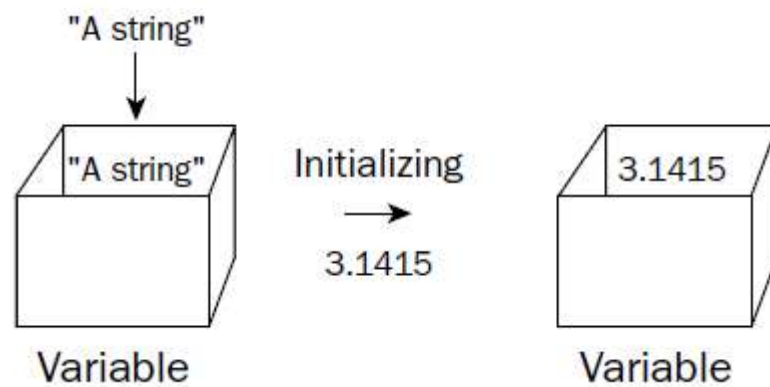


الشئ العظيم فى المتغيرات انها شديدة التنوع , يمكنك تغيير ما يحمله المتغير من البيانات

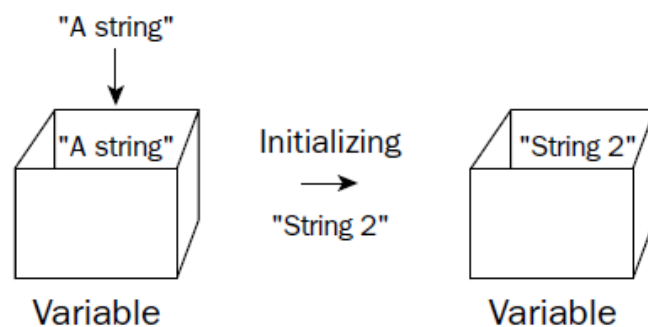
هناك طريقتين يمكن ان يتم تحديث المتغير بهما :

- يمكن ان يتغير المتغير تماماً (إنظر الشكل التالى) . نوع البيانات يمكن ان يبقى نفس الشئ أو يمكن أن يتغير. هذا مثال للمتغير عادةً ما يعاد إستخدامه لأغراض مختلفة خلال الكود .

بشكل عام , هذا غير موصى به لأنه كما سترى فى وقت لاحق , أنه يشجع المشاكل ويجعل الأخطاء أكثر إحتمالاً فى الوقوع .



- يمكن أن يتم تحديث المتغير , هذا يعنى أن البرنامج يعمل مع البيانات المحملة بالمتغير ويجرى بعض التغييرات إليها . وهى أغلب الطريق شيوعاً التى يستخدم فى الكود (إنظر الشكل)



Variables in Action

المتغيرات في العمل

دعنا نحرز بعض التقدم البرمجي ونستخدم بعض المتغيرات . لهذا الجزء من الكتاب , سننظر إلى JavaScript لأنها سريعة , وبسيطة , ولا يطلب منا تفسير للكود, فإنه يوضح النقطة على نحو فعال , وتعمل بطريقة متشابهة في كل المتصفحات .

Variable Run Through

المتغير خلال التشغيل

حسناً , دعنا نشغل بعض الكود الذي يبين كيف يعمل الكود . سنصنع بعض التقدم هنا ونقدم بعض مفاتيح المفاهيم التي من المحتمل أنك لم ترها من قبل . ومع ذلك , هذه الصفحات القليلة التالية ستعطيك بعض الأيدي في البرمجة للخبرة .

Creating a Template

إنشاء قالب

منذ أن تبدأ باستخدام JavaScript , فأول شيء تحتاجه هو قالب ليسمح لك بتشغيل الكود في المتصفح . لما سنفعله هنا النموذج التالي سيكون جيداً :-

```
<html>
<head>
<title>JavaScript Test Template</title>
<script language="JavaScript">
<!--
/*
Code goes below this comment block
*/

// -->
</script>
</head>
<body>

</body>
</html>
```

شاهد تعليقات HTML قد استخدمت داخل <script> و</script> في هذا المثال ؟ هذا حسن حيث أن الكود لن يصدر أي أعطال في بعض المتصفحات .

معظم محتويات الملف هي HTML , إذا كانت لديك خبرة في إنشاء صفحات الإنترنت . إذا الكثير من ذلك سيكون مألوفاً لك بالفعل , وإليك خطوات داخل الكود .

هناك ثلاثة أسطر يبدأ بهم الملف , إفتتاح رأس الصفحة (حيث سيكون الكود الخاص بك) , ومن ثم إضافة عنوان للصفحة .

```
<html>
<head>
<title>JavaScript Test Template</title>
```

الثمانية أسطر التالية يسمون كتلة كود Code block — فهي الكتلة التي ستمسك بكود JavaScript الذي سنتشأه .

```
<script language="JavaScript">
<!--
/*
Code goes below this comment block
*/

// -->
</script>
```

الخمس أسطر الباقية يغلقان الرأس لملف HTML , ويفتح , ويغلق جسم الصفحة (وهو المكان الذي سيكون به محتويات الصفحة) وأخيراً إغلاق الصفحة

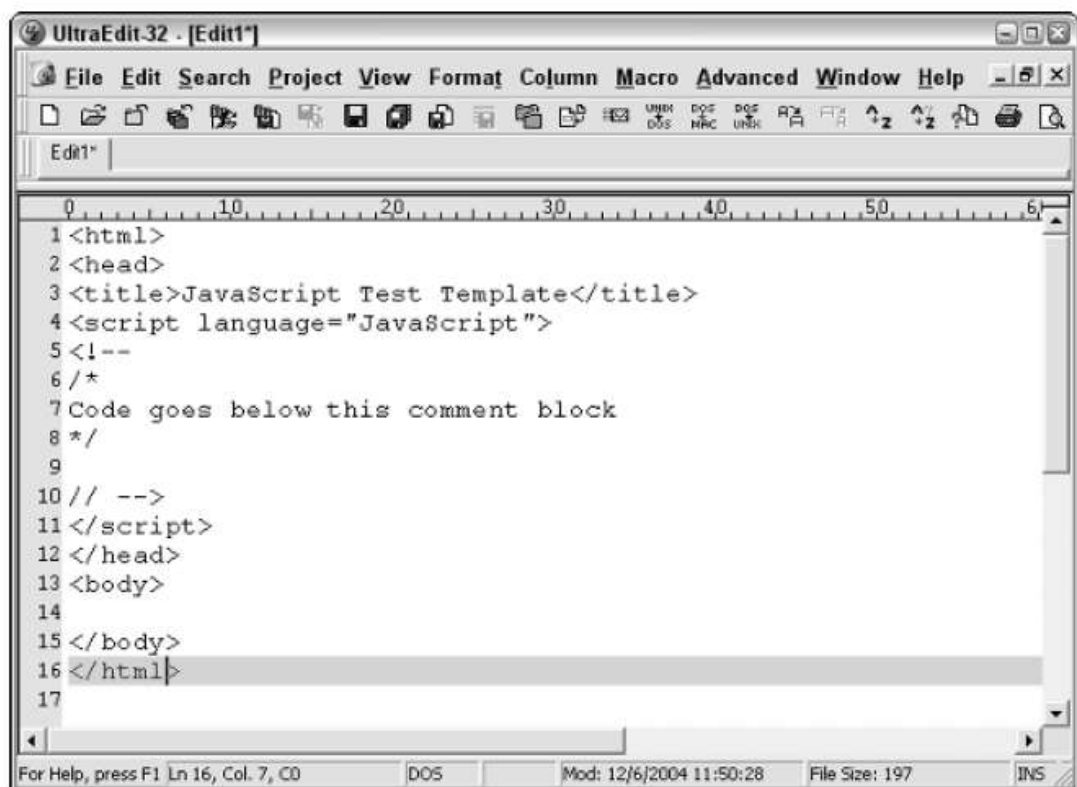
```
</head>
<body>

</body>
</html>
```

خذ هذا الكود , وإنسخه في ملف نصي واحفظه كـ jstemplate.htm . وهذا هو الملف الذي يمكن ان تفتحه في المستعرض لإختبار الكود .

المتصفح الذي أوصى به لتستخدمه لتشغيل أمثلة JavaScript هو Internet Explorer (على الرغم من الكود لابد وأن يعمل مع معظم المتصفحات الحديثة الأخرى) .

بمجرد ان تكمل هذا الملف , إفتحه في محرر نصوص , (windows NotePad أو UltraEdit , على سبيل المثال) الشكل التالي يظهر الكود في UltraEdit .



Writing Code

كتابة الكود

حسناً , دعنا نكتب كود ينظر إلى المتغيرات في العمل .

منذ أن تعاملنا مع المتغيرات , دعنا ننشأ متغيرين إثنيين , للتبسيط , سنقوم بتسميتهم X , Y . ولفعل ذلك نستخدم الكلمة المحجوزة `var` للإعلان (كلمة أخرى لتنشأها) المتغير .

```
<html>
<head>
<title>JavaScript Test Template</title>
<script language="JavaScript">
<!--
/*
Code goes below this comment block
*/
var x
var y
// -->
</script>
</head>
<body>
</body>
</html>
```

كما يمكننا إعلان المتغيرين في سطر واحد , كما يلي

```
<html>
<head>
<title>JavaScript Test Template</title>
<script language="JavaScript">
<!--
/*
Code goes below this comment block
*/
var x, y
// -->
</script>
</head>
<body>
</body>
</html>
```

لهذه اللحظة , كلا المتغيرين تم إعلانهما , ولكن غير معرفين وبالتالي فارغين .

ولتعريفهم , كتبنا التالي :

```
<html>
<head>
<title>JavaScript Test Template</title>
<script language="JavaScript">
<!--
/*
Code goes below this comment block
*/
var x = 3
var y = 4
// -->

</script>
</head>
<body>
</body>
</html>
```

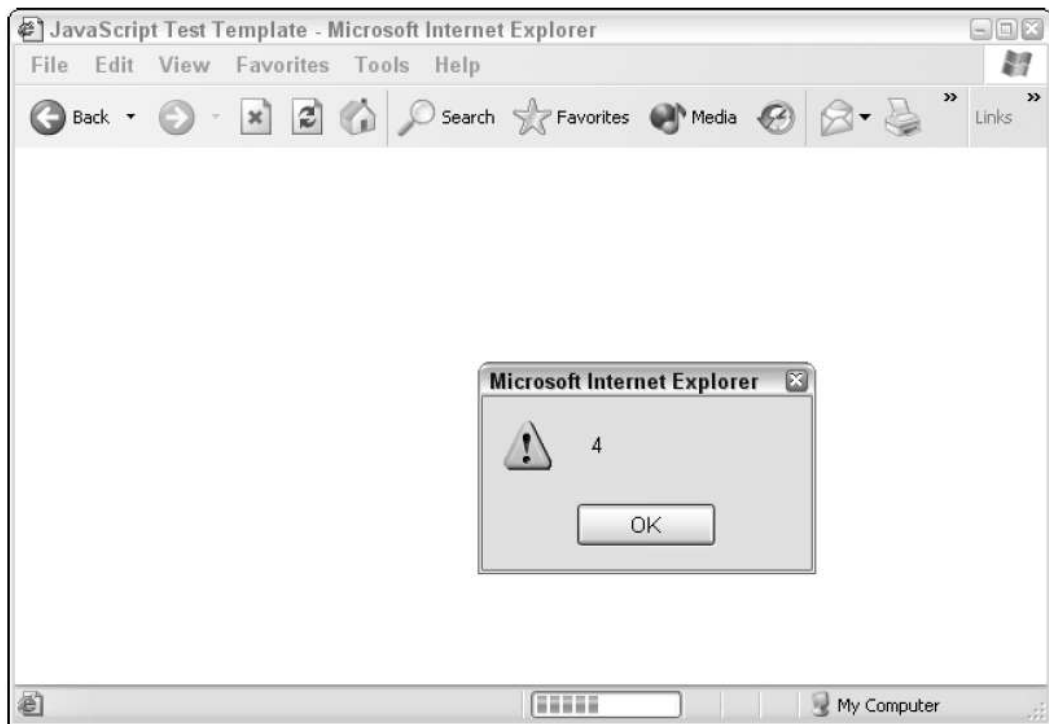
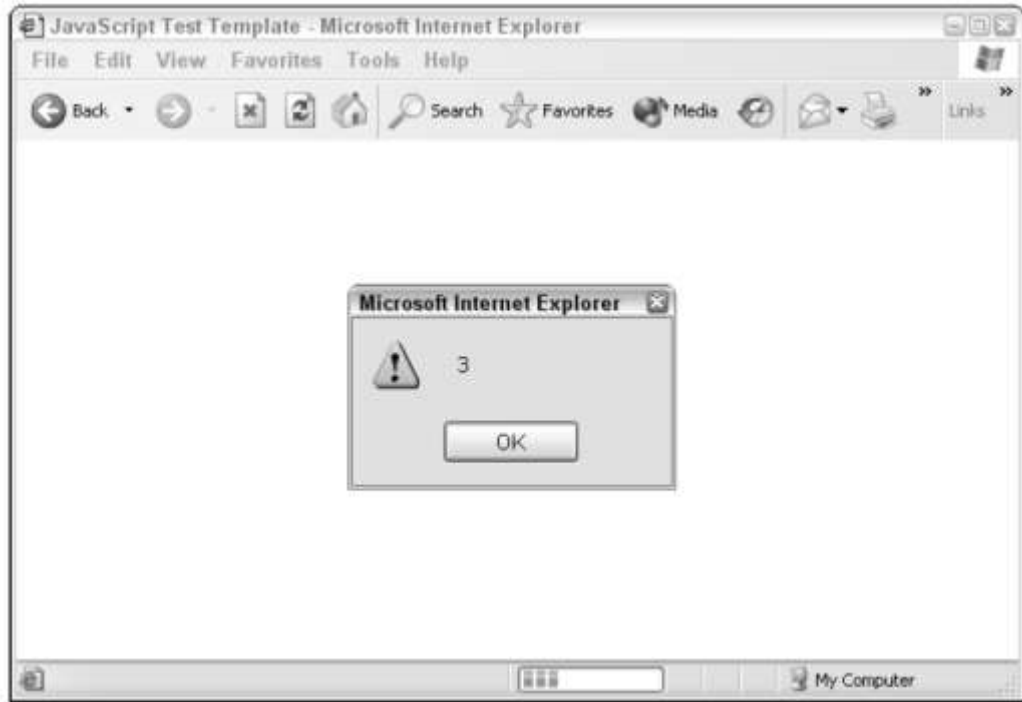
الآن x يحمل القيمة 3 و y يحمل القيمة 4 , هذه أرقام يحملونها (أرقام صحيحة من كل الأرقام) وليست الحروف الأبجدية 3 و 4 . وإذا كنت تريد المتغيرات أن تمسك الحروف الأبجدية 3 و 4 . إذاً ستحتاج أن تكتبهم كالتالى :

```
<html>
<head>
<title>JavaScript Test Template</title>
<script language="JavaScript">
<!--
/*
Code goes below this comment block
*/
var x = "3"
var y = "4"
// -->
</script>
</head>
<body>
</body>
</html>
```

يمكن الآن أن تثبت أن المتغيرات تسمك القيم بجعل الكود يعرض القيم فى مربع رسالة كما يلى :

```
<html>
<head>
<title>JavaScript Test Template</title>
<script language="JavaScript">
<!--
/*
Code goes below this comment block
*/
var x = 3
var y = 4
alert(x)
alert(y)
// -->
</script>
</head>
<body>
</body>
</html>
```

إحفظ الملف وقم بتحميله فى متصفح . وعندما ستحمل الصفحة ستجد رسالتين قد عرضا لك كما يظهر بالشكلين التاليين



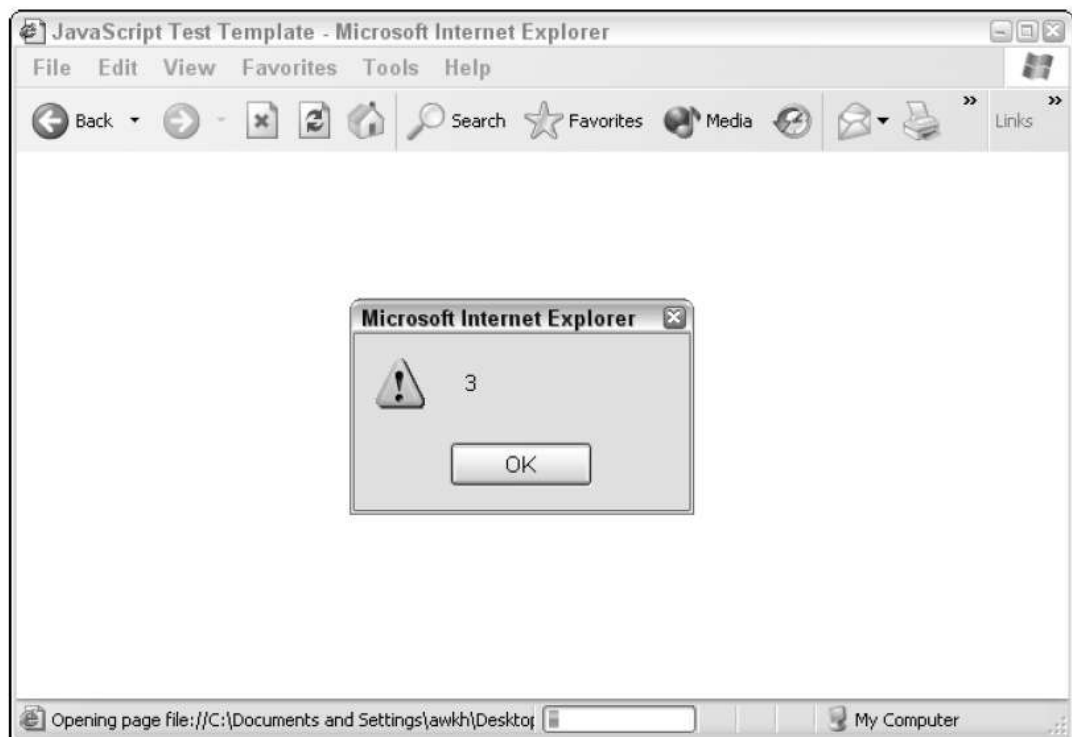
تعرض الرسالتين قيمة المتغيرات . الآن يمكنك أن تعمل على القيم الممسكة بواسطة المتغيرات وإستخدام الرسائل التي نعرضها لتعرض لك تغير القيم .

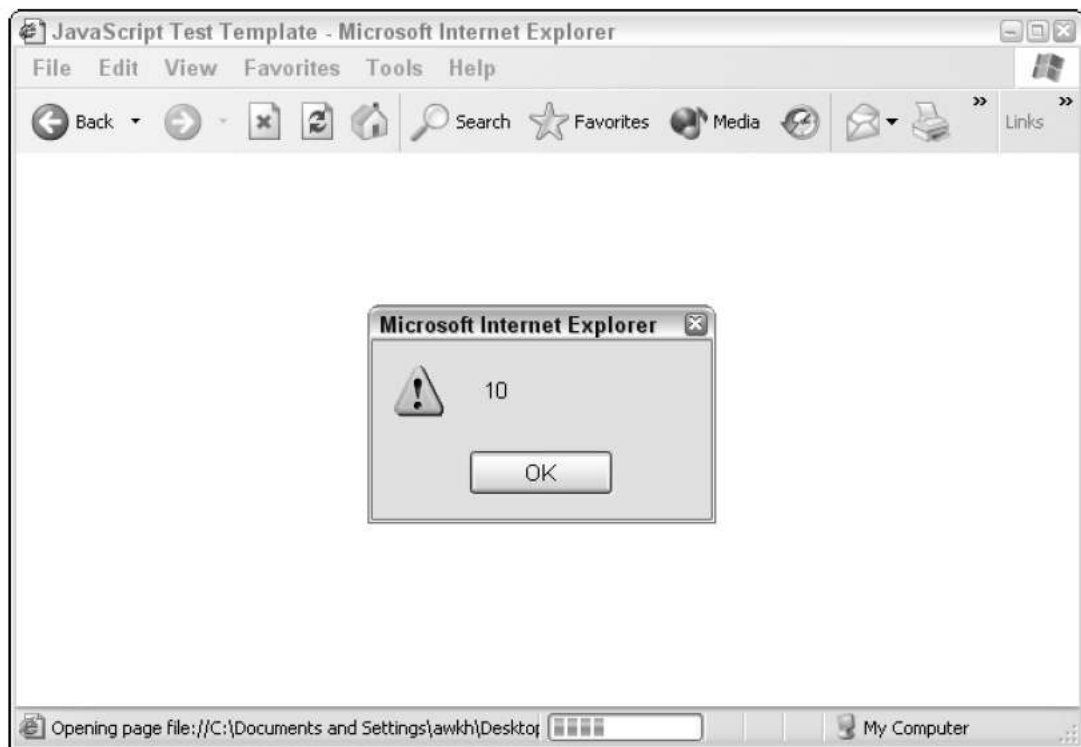
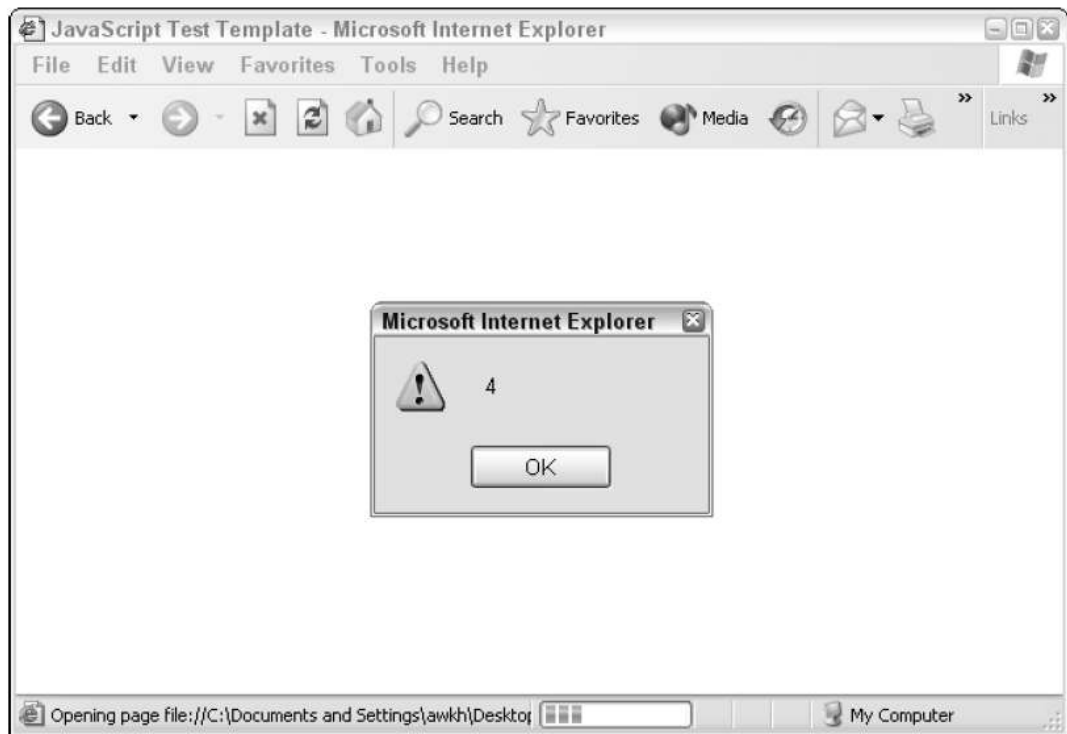
إلقى نظرة على الكود المتغير.

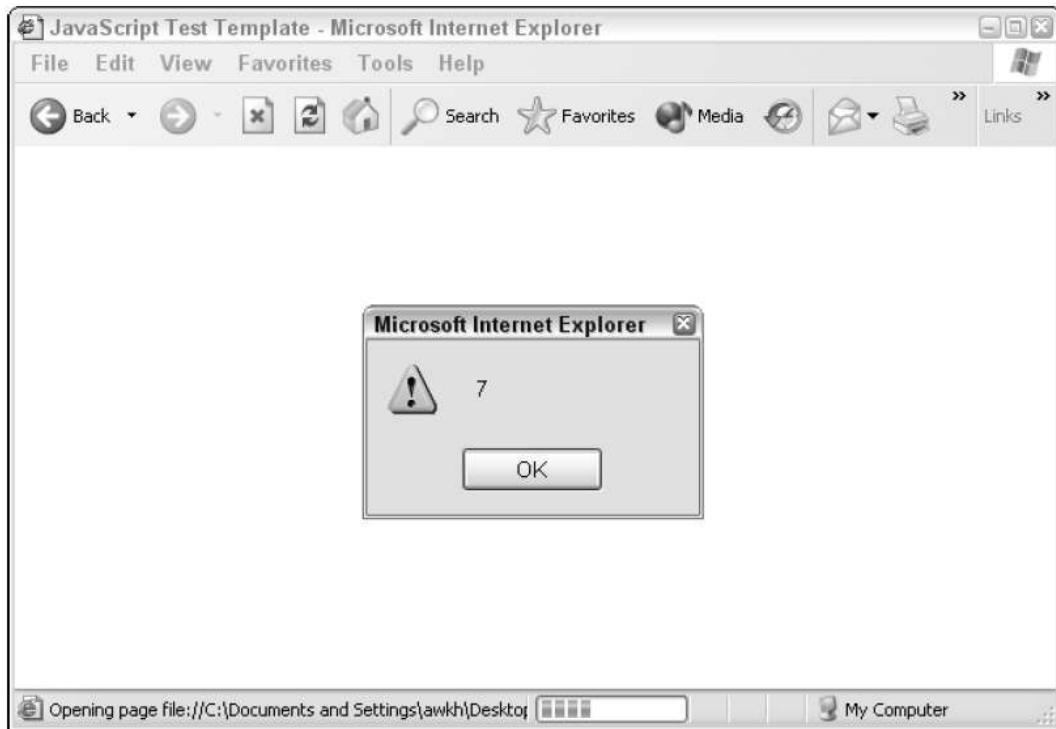
من الآن سأظهر فقط الجزء الخاص بـ JavaScript للكود ونتجاهل مكونات HTML – فقط تذكر أنك تحتاج إلى HTML كذلك .

```
<script language="JavaScript">
<!--
/*
Code goes below this comment block
*/
var x = 3
var y = 4
alert(x)
alert(y)
var x = 10
var y = 7
alert(x)
alert(y)
// -->
</script>
```

ماذا حدث هنا ؟ حسناً , إلقى نظرة على الرسائل المنبثقة التي قد تولدت (إنظر الأشكال التالية)







إعلنت القيم للمتغيرات ثم عرضت في الأشكال . هي نفس القيم كمن قبل . ولكن إنظر إلى الكود , بعد هذا , قمنا بإعلانهم مرة أخرى , هذه المرة بقيم مختلفة وعرضناهم مرة أخرى في رسائل منبثقة التي تعرض بوضوح القيم التي تم تغييرها وإستبدالها بالقيم الجديدة .

في ذلك المثال إستبدلنا الأرقام الصحيحة , ولكن لا يوجد قاعدة تقول أننا يجب أن نفعل هذا , الكود التالي إستبدل قيم الأرقام الصحيحة بنص حرفي .

```
<script language="JavaScript">
<!--
/*
Code goes below this comment block
*/
var x = 3
var y = 4
alert(x)
alert(y)
var x = "Hello"
var y = "World!"
alert(x)
alert(y)
// -->
</script>
```

دعنا نعود للقيم الرقمية للحظة , أريد ان أريك أنه يمكنك فعل أكثر من مجرد عرض أرقام ثابتة فقط عند إستخدام المتغيرات . إلقى نظرة على الكود التالي :

```
<script language="JavaScript">
<!--
/*
Code goes below this comment block

*/
var x = 3
var y = 4
var answer
var answer = x + y
alert(answer)
// -->
</script>
```

هنا قمنا بتقديم متغير جديد اسمه answer :

```
var answer
```

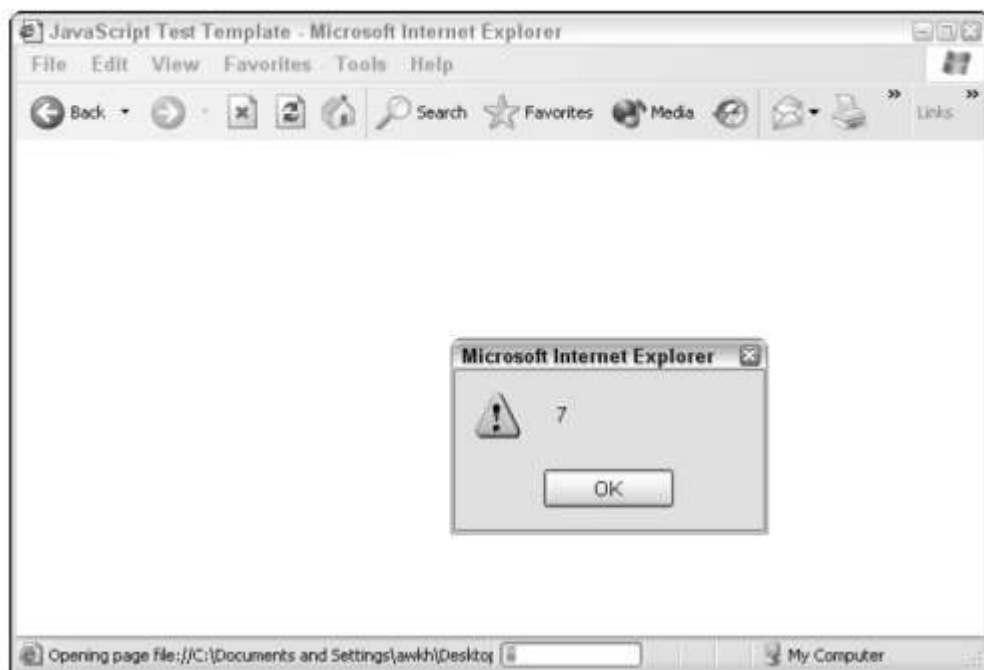
ثم عرفنا answer كمجموع X و Y :

```
answer = x + y
```

وعرضنا answer في رسالة منبثقة Pop UP

```
alert(answer)
```

ناتج هذه الرسالة يعرض في الشكل



كما ترى , ما تم عرضه هو جمع $4+3$. من هذا يمكن أن ترى اننا قد بدأنا البرمجة الصحيحة لإننا لدينا القدرة على معالجة البيانات وتخزينها .

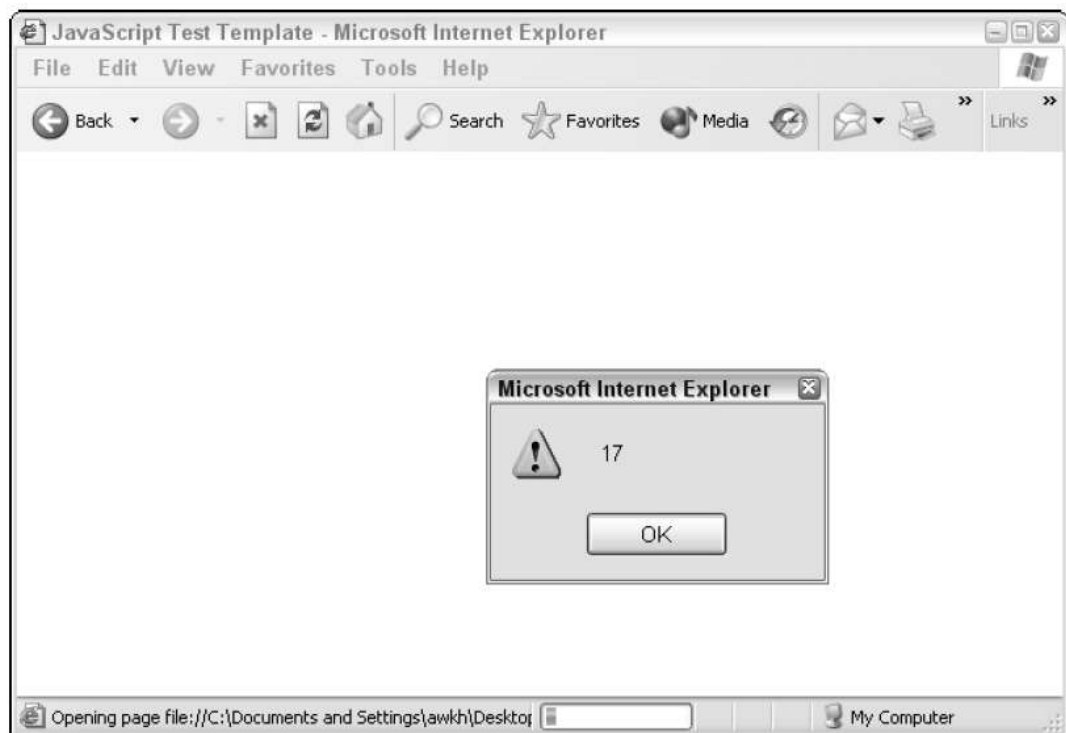
يمكن أن تعيد إستخدام المتغيرات أيضاً , إلقى نظرة على الكود التالي , الذى يعيد إستخدام المتغيرات مرات ومرات .

```
<script language="JavaScript">
<!--
/*
Code goes below this comment block
*/
var x = 3
var y = 4
var answer
var answer = x + y + x + y + x
alert(answer)
// -->
</script>
```

الشكل التالي يعرض قيمة `answer` .

وهذا مساوياً لفعل التالي

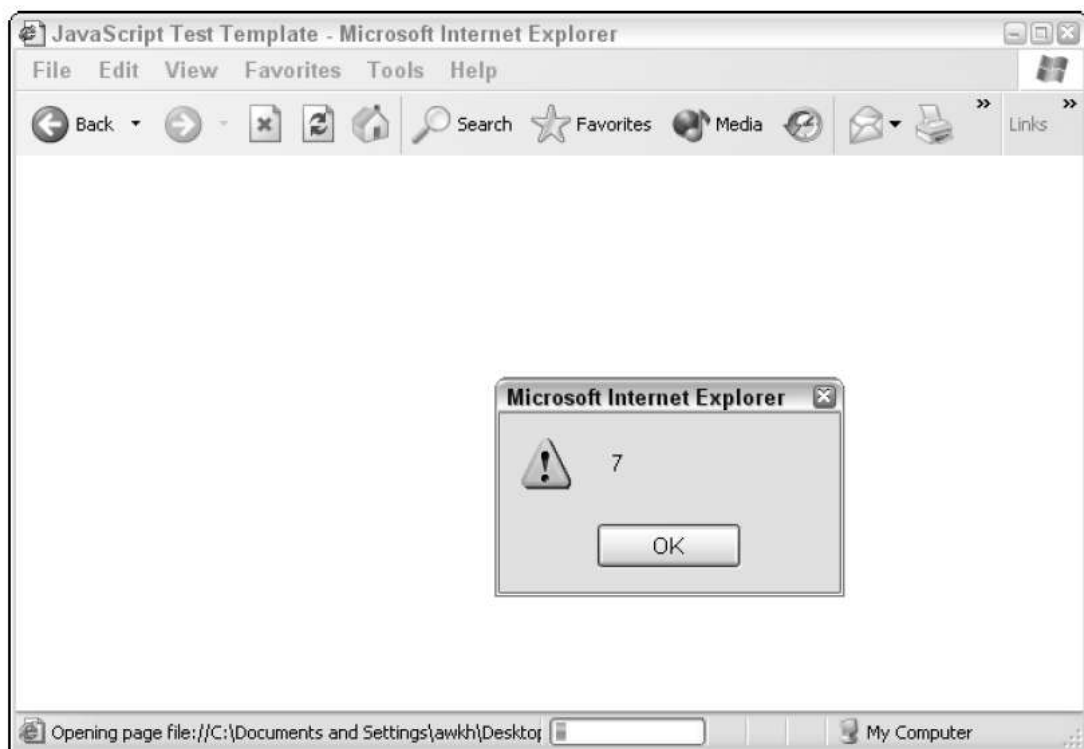
```
<script language="JavaScript">
<!--
/*
Code goes below this comment block
*/
var x = 3
var y = 4
var answer
var answer = 3 + 4 + 3 + 4 + 3
alert(answer)
// -->
</script>
```



الطريقة الأسرع والأبسط لإعادة تعريف المتغير مع إستجابته للمجموع هي :

```
<script language="JavaScript">
<!--
/*
Code goes below this comment block
*/
var x = 3
var y = 4
var x = x + y
alert(x)
// -->
</script>
```

رسالة pop-up تعطى القيمة لـ x الآن كـ 7 وليس 3 (إنظر الشكل) . هذه القيمة الجديدة ستبقى حتى تتغير القيمة للمتغير مرة أخرى أو إعادة ضبطه أو إعادة تنفيذ الكود .



هل لاحظت كيف قمنا بإضافة كلمة var قبل كل متغير ؟ في بعض اللغات عند أول إستخدام للمتغير تحتاج أن تعرفه تصريحياً قبل أن تستخدمه , و بعد ذلك يمكن أن تستخدمه بدون فعل ذلك , في JavaScript تفعل فقط مثل هذا :

```
var myVariable
```

وبعد ذلك يمكن ان تستخدم فقط المتغير بدون إضافة var

```
myVariable = "something ..."
```

JavaScript هي لغة أقل تشدداً وتسمح لك إلا تفعل ذلك . وهي احد المناطق التي ستجد بها التنوع من لغة إلى لغة . وهذه الأشياء التي تتغير من لغة إلى لغة يمكن ان تجعل التعلم شائك .

Naming Variables

تسمية المتغيرات

وحتى الان قد شهدت ثلاثة متغيرات :

```
x  
y  
answer
```

فى سياق الكود , ما تعتقد فى هذه المتغيرات ؟ هل هم فعالين ؟ هل يخبرونك بأى شئ مهم ؟ وهل هم ذو معنى ؟

حسناً , المتغير `answer` هو لبعض الدرجات , ولكن `x` , `y` غامضين تماماً .
يمكنك ان تترك تعليق لجعل المتغير بارزاً وأكثر دلالة على معناه .

```
<script language="JavaScript">  
  <!--  
  /*  
  Code goes below this comment block  
  */  
  var x = 3 // variable x - arbitrary number  
  var y = 4 // variable y - arbitrary number  
  var answer = x + y // variable answer - sum of x + y  
  alert(answer)  
  // -->  
</script>
```

هذا يعمل , لكنه مرهق ومن أجل ان تكون فعاله سيجب عليك أن تترك تعليق على كل سطر فى الكود الذى يستخدم المتغيرات , او فعل ذلك عندما يتم استخدامه اكثر من مرة واحدة , وهذا يضيف إلى الكود الكثير من التراكم .

وهناك طريقة جميلة جداً من كونها اكثر منهجية ومستفضية حول تسمية المتغيرات فى المقام الأول .
وقبل أن ننظر إلى تسمية المتغيرات وجعلهم أيسر للفهم , تحتاج أن تعرف أولاً ما يمكن وما لا يمكن استخدامه فى الاسم .

وما هى جولة سريعة فى ما تفعل وما لا تفعل فى تسمية المتغيرات التى تطبق فى جميع المجالات لمعظم اللغات .

Don'ts

لا تفعل

- أسما المتغيرات لا يمكن أن تحتوى على أرقام فى البداية , كما يظهر بالشكل .

```
123var  
999var  
911variable  
1stvar  
01variable  
001string
```



- أسماء المتغيرات لا يمكن أن تحتوى على معاملات رياضية . كما يظهر بالشكل

```
.variable  
string.1  
op,001  
"variable"
```



- أسماء المتغيرات لا يمكن أن تكون منشأة بإسم كلمة محجوزة منسوبة للغة كما هو موضح بالشكل

```
var  
alert  
prompt
```



- لا يمكن أن تحتوى إسماء المتغيرات على مسافة . كما موضح بالشكل


```
a variable  
string 001  
first var
```



Dos **إفعل**


- يمكن أن يحتوى إسم المتغير على أرقام طالما أنه ليس فى البداية . كما هو موضح بالشكل

```
var001  
string911var  
s001
```




- Underscores (_) مسموح بها فى تسمية المتغيرات , كما هو موضح بالشكل .

```
string_001  
var_name  
string_001_var
```



- يمكن ان يكون إسم المغير مختلط فى حالة الأحرف من حيث كونها كبير أو صغير كما بالشكل

```
StRing001  
Var_01AAb  
Avariable_01
```



Naming of Variables

تسمية المتغيرات

هناك طرق عديدة تجعل اسم المتغير أكثر وضوحاً وأكثر فاعلية لأي مبرمج يأتي بعد ذلك بفترة ويقراً الكود .

دعنا نلقى نظرة على بعض المخططات التي يمكن ان نستخدمها

Clearer Naming

توضيح الاسم

اول شيء يمكن أن تفعله لجعل اسم المتغير أوضح هو استخدام تسمية أوضح . نأخذ مثالنا السابق

```
x
y
answer
```

أيهما تعتقد انه أوضح ؟ حسناً answer بالطبع هو .

في تطبيق ما , ربما يكون لديك جملة أو امر بشكل الشكل :

```
z = x * y
```

حيث ان z هي ناتج x , y مضروبين في بعضهما . كعملية الرياضية أكثر وضوحاً, ولكن ماذا يمثل كل من المتغيرات فهذا مفقود . وللتحسين يجب أن يكون هكذا

```
totalvalue = subtotalprice * taxrate
```

هذا أكثر وضوحاً ويجعل الكود منطقياً رياضياً ليسهل تتبعه , وهذا يجعل رصد الأخطاء أسهل , خذ نظرة على هذا :

```
totalvalue = subtotalprice - taxrate
```

طرح معدل الطريبة من المجموع أليس يولد إحساس بما يعنى هذا ؟ فعندما نسمى المتغيرات أوضح , فقط من خلال مراجعة بعض الأخطاء البرمجية تصبح أكثر وضوحاً.

Capitalization

الكتابة بحرف كبير

شيء آخر يجعل قراءة الكود أسهل هو حرصك على أن تكتب اول حرف بحروف كبيرة لكل كلمة في اسم المتغير . إنظر إلى هذين السطرين من الكود :

```
totalvalue = subtotalprice - taxrate
```

```
TotalValue = SubTotalPrice - TaxRate
```

أعتقد أنك ستوافق على أن الثانية أسهل لأن تقرأ لأن الحروف الكبيرة تلفت العين إلى الكلمة جيداً .

التعديل الذى يجعل الكود أسهل ليقراً وربما أقل عملاً هو عدم الكتابة بأحرف كبيرة للحرف الأول من المتغير . إنظر إلى هذا

```
TotalValue = SubTotalPrice - TaxRate  
totalValue = subTotalPrice - taxRate
```

حاول أن تفعل هذا كلما امكن لأنه حقاً يجعل الكود أكثر يسر ليقراً و يترجم .

Use Underscores إستخدام العلامة (_)

طريقة أخرى لتوضيح المتغيرات هو إستخدام هذه علامة (_) فى إسماء المتغيرات لزيادة الإستخدام لتكبير حالة الأحرف .

والنمط الجيد هو إستخدام هذه العلامات (_) بين الكلمات التى اولها حرف كبير . بسبب ان هذا أيضاً يساعد على القراءة للكود . إنظر على هذا الكود التالى :

```
totalvalue = subtotalprice - taxrate  
TotalValue = SubTotalPrice - TaxRate  
totalValue = subTotalPrice - taxRate  
totalValue = subTotal_Price - taxRate
```

كلما كبر إسم المتغير كلما ساعدك هذا النمط كثيراً :

إلقى نظر على هذا مع وبدون علامات _ :

```
totalValueBeforeTax  
totalValue_Before_Tax  
cpuCoreTempCelsius  
cpuCore_Temp_Celsius  
minFileSizeMb  
minFile_Size_Mb
```

أعتقد انك ستوافق على أن إسم المتغير الذى يحتوى على كلا من تكبير الأحرف وعلامات _ يكون أسهل ليقراً من الذى يكون بدونهما .

Naming Notation

صدق اولاً تصدق أن هذا التدوين لتسمية المتغيرات . ويسمى هذا التدوين الهنجارى أو المجرى و طور بواسطة مبرمج مايكروسوفت إسمه Charles Simonyi , فقط لأنه هنجارى , ومن هنا جاءت تسميته .

الغرض من التدوين الهنجارى كان لإحضار الترتيب الى الفوضى فى إنشاء وتمثيل إسماء المتغيرات . ويعمل بواسطة تقد إسم المتغير بالحروف التى تحدد نوع المعلومات التى يسمك بها المتغير .

على سبيل المثال , إذا كان لديك متغير يمسك بقيمة نصية , يمكن أن تسميه شئ من هذا القبيل :

```
stringUser_Name
```

سيعمل هذا ولكن المشكلة تأتي عندما يبدأ المبرمجون بإختصارهم (كما يميل المبرمجون إلى فعل ذلك) قريباً سينتهي بك الحال لذلك :

```
strgUser_Name  
sUser_Name  
strUser_Name
```

وإحتمالية المزيد , أفضل شئ هو تقديم أحرف بادئه قياسية .

وهناك الكثير من الاحرف البادئه , الكثير منهم لا يحتاج ان يزعجك الآن والبعض من غير المرجح ان يكون كذلك للإستخدام على كل حال , ولكن ها هنا القليل لتجد البداية

Prefix	Type	Example
Str	String	strUser_Name
C	Character	cStudent_Grade
Dt	Date (and time)	dtStart
Cur	Currency	curTotal
B	Boolean (true and false)	bIs_Valid
Int	Integer	intAge

هناك الكثير من هذه الأنواع , ولكن للآن هذا كافى للحصول عليه . وكما تحتاج إلى الكثير للأنواع الكثير والبادئات , وسنقدمهم كلما تقدمنا .

Quick Exercise

تمارين سريع

إليك بعض الأسئلة السريعة لتختبر معرفتك بالمتغيرات وإستخدامهم ,

- 1 - إشرح فى مصطلح بسيط ما هو المتغير ؟
- 2 - حدد المتغيرات فى الكود

```
<script language="JavaScript">  
<!--  
/*  
Code goes below this comment block  
*/  
var curTotal_Price = 3  
var curTotal_Price = 4  
curTotal = curTotal_Price + curTotal_Price  
alert(curTotal)  
// -->  
</script>
```

- 3 - صف الطرق التى تجعل المتغيرات أكثر سهولة لأن تقرأ وتفهم .
- 4 - إنظر إلى الكود التالى وتتبع المتغير الجيد والأكثر وصفاً

```

<script language="JavaScript">
<!--
/*
Code goes below this comment block
*/
var name = "A. N. Other"
var zipcode = 09090
var age = 27
var member = true
// -->
</script>

```

Strings القيم الحرفية

دعنا نحول اهتمامنا بعيداً عن المتغيرات ونبدأ بالنظر إلى ما يمسك به المتغير وهي القيمة . وحتى الآن لقد بحثنا بشكل رئيسي في المتغيرات التي تحمل الأرقام (أو أكثر دقة , الأرقام الصحيحة) دعنا الآن نلقى نظرة على نوع آخر من البيانات التي يمكن ان تحملها المتغيرات — القيم الحرفية .

What Are Strings? ما هي Strings ؟

تقنياً , String هي التسلسل لإثنين أو أكثر من الحروف . ويمكن أن تحتوى على حروف رقمية (حروف وأرقام) إضافة إلى المسافات والرموز (مثال التشكيل للحروف) .

حرف واحد هو مجرد حرف , على الرغم من أن بعض اللغات تنظر للحرف الواحد ك String وتقريباً جميعهم لا يمانع من التعامل مع الحرف الواحد ك String .

هاهي بعض المتغيرات التي تحمل قيم من نوع string

```

var strTest_String1 = "Hello, World!"
var strTest_String2 = "123xyz"
var strTest_String3 = "[*^£/\~"

```

من الهام لك أن تدرك أن الـ strings هي نفسها التي داخل علامة التنصيص ولاحظ أن علامة التنصيص ليست تابعة لها . لذلك , القيم الحرفية الحقيقية تكون :

```

Hello, World!
123xyz
[*^£/\~

```

يمكن أن تحصل على كود JavaScript لعرض نص تشبة كثيراً نفس طريقة عرض الأرقام . إلقى نظرة على الكود التالي :

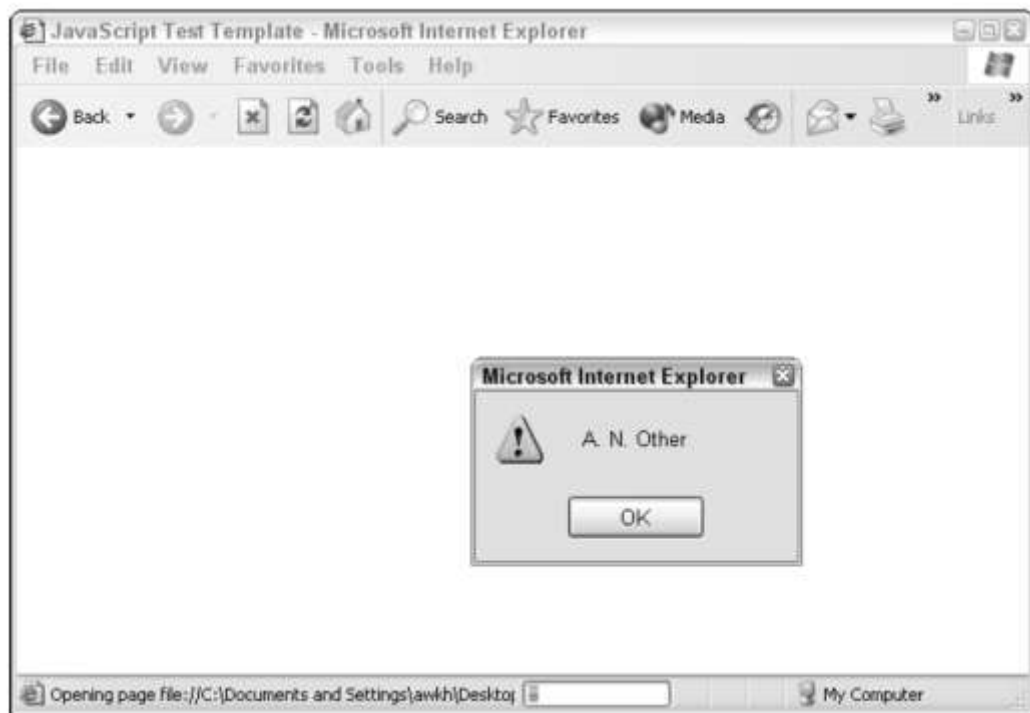
```

<script language="JavaScript">
<!--
/*
Code goes below this comment block
*/
var strName = "A. N. Other"
alert(strName)
// -->
</script>

```

مخرجات هذا تظهر في الشكل التالي .

لأشئ جديد , إذا كان الكود يمكن ان يعرض أرقام , إذاً عرض النص ليس معضلة , ولكن الآن حان الوقت لنلقى نظرة على كيف تستطيع المتغيرات معالجة البيانات والتسلسل الحرفي النصي الذي يحتويه .



String Manipulation

معالجة القيم الحرفية

إلقى نظرة على الكود التالي :

```

<script language="JavaScript">
<!--
/*
Code goes below this comment block
*/
var strFirst_Name = "Andy"
var strMiddle_Initials = "N"
var strLast_Name = "Other"
alert(strFirst_Name)
alert(strMiddle_Initials)
alert(strLast_Name)
// -->
</script>

```

تنفيذ هذا الكود سينتج عنه ثلاثة رسائل منبثقة كما يظهر أمامك بالشكل

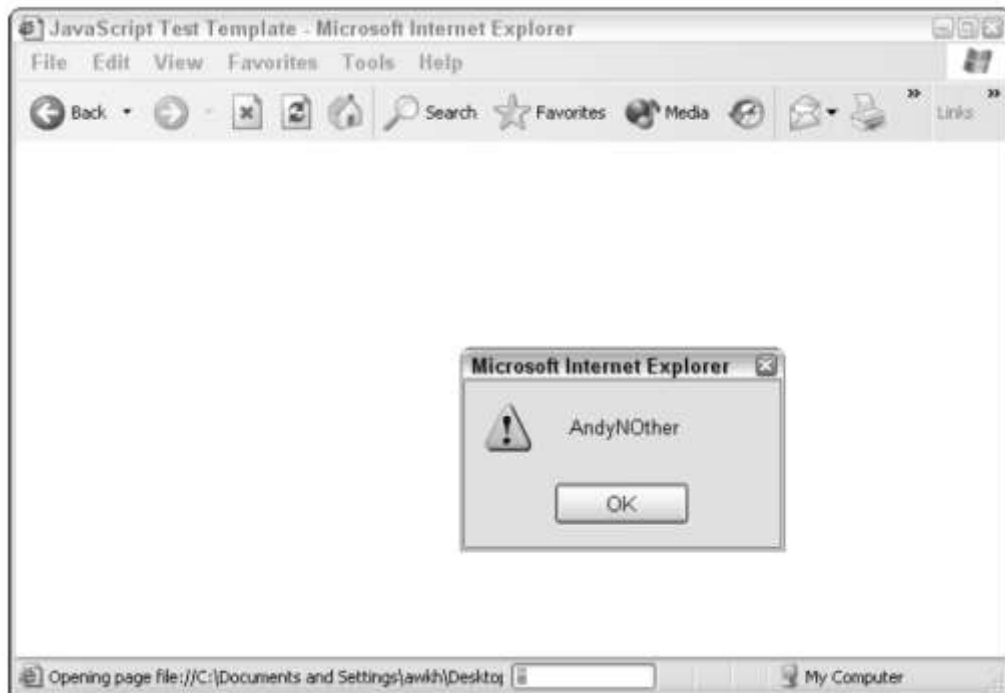


الان , هذا شيق , ولكن ليس وظيفياً جداً , سيكون جيد جداً اذا تمكنت من أن تعالج البيانات وتقوم بربط الثلاثة أجزاء في إسم واحد معاً لتظهرهم في رسالة واحدة popup . الن يكون هذا جيداً ؟ بالطبع, يكون . عملية ربط القيم الحرفية النصية معاً تعرف ك concatenation تسلسل ويعتبر دالة معالجة قيم حرفية نصية أساسية .

تسلسل [أى جمعهم في سلسلة] القيم الحرفية في JavaScript تقتضى استخدام معامل + . لذلك لتسلسل القيم الحرفية السابقة في متغير واحد ومن ثم عرضه يمكن كتابة التالى :

```
<script language="JavaScript">
<!--
/*
Code goes below this comment block
*/
var strFirst_Name = "Andy"
var strMiddle_Initials = "N"
var strLast_Name = "Other"
alert(strFirst_Name + strMiddle_Initials + strLast_Name)
// -->
</script>
```

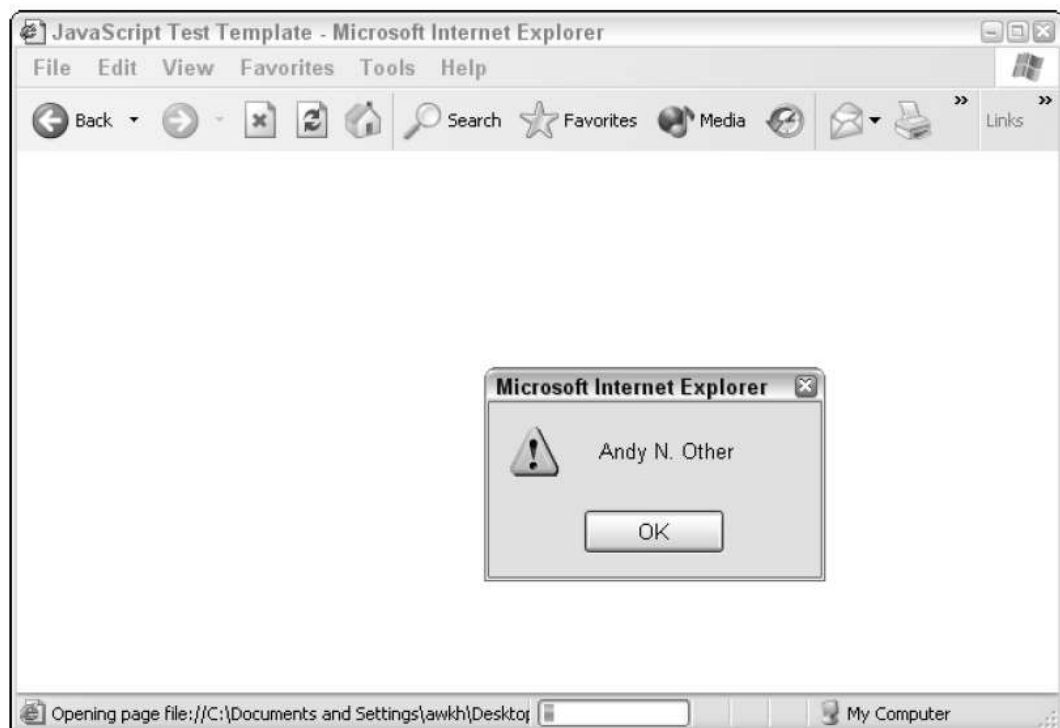
مخرجات الرسالة من هذا كما يظهر بالشكل



هل لاحظت شيئاً خطأ خلال المخرجات ؟ لاحظ أنه ليس هناك مسافات بين القيم الحرفية الفردية المتسلسلة مع بعضهما . ولإضافة مسافات , يمكنك وضع القيم الحرفية في جملة التسلسل هكذا :

```
<script language="JavaScript">
<!--
/*
Code goes below this comment block
*/
var strFirst_Name = "Andy"
var strMiddle_Initials = "N"
var strLast_Name = "Other"
alert(strFirst_Name + " " + strMiddle_Initials + ". " + strLast_Name)
// -->
</script>
```

الشكل التالي يظهر المخرجات الإن في التنسيق المناسب .



هناك تحسينات يمكنك فعلها مع الكود . أولاً , إذا كنت تريد العمل بصفاء مع المتغيرات , يمكنك جعل مسافة واحدة كمتغير و نقطة وقوف كمتغير آخر .

```
<script language="JavaScript">
<!--
/*
Code goes below this comment block
*/
var strFirst_Name = "Andy"
var strMiddle_Initials = "N"
var strLast_Name = "Other"
var strSpace = " "
var strFull_Stop = "."
alert(strFirst_Name + strSpace + strMiddle_Initials + strFull_Stop + strSpace +
strLast_Name) // this should be typed as one line of code
// -->
</script>
```

هذا الكود الآن سيفعل بالضبط نفس الشيء كالكود السابق , ولكنه أرتب قليلاً .

ربما تريد ان تربط أجزاء الإسم فى قيمة حرفية واحدة يمكنك أن تستخدمهم فى اكثر من مكان , فى هذه الحالة , إفعل التالى , فإذا كنت بحاجة إلى إعادة استخدام الاسم كاملاً عدة مرات , ستحفظ الكثير من الكتابة

```
<script language="JavaScript">
<!--
/*
Code goes below this comment block
*/
var strFirst_Name = "Andy"
var strMiddle_Initials = "N"
var strLast_Name = "Other"
var strSpace = " "
var strFull_Stop = "."
var strFull_Name = strFirst_Name + strSpace + strMiddle_Initials + strFull_Stop +
strSpace + strLast_Name // this should be typed as one line of code
alert(strFull_Name)
// -->
</script>
```

نرى أنها ضخمة , سطر كود كريه ؟ طالما أنها تعمل , فيجب ان تقرأ عبر الصفحة إلى هذا الحد أو التفاف السطر يمكن أن يجعله صعباً ان يكره ويصعب المشكلة . يمكنك ان تستخدم الربط أو التسلسل لتربط السطر فى مرحلتين كما يلى :

```
<script language="JavaScript">
<!--
/*
Code goes below this comment block
*/
var strFirst_Name = "Andy"
var strMiddle_Initials = "N"
var strLast_Name = "Other"
var strSpace = " "
var strFull_Stop = "."
var strFull_Name = strFirst_Name + strSpace + strMiddle_Initials
strFull_Name = strFull_Name + strFull_Stop + strSpace + strLast_Name
alert(strFull_Name)
// -->
</script>
```

الربط تم فى مرحلتين

```
var strFull_Name = strFirst_Name + strSpace + strMiddle_Initials
```

بنهاية هذه المرحلة , المتغير strFull_Name يحمل القيمة التالية :

Andy N

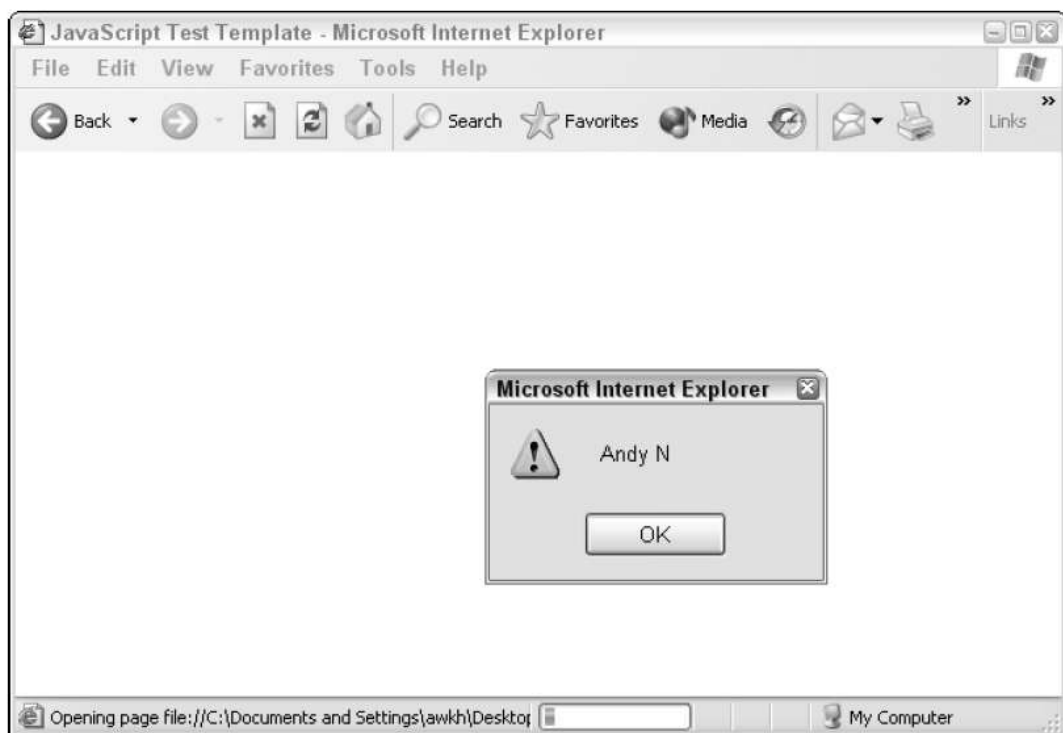
يمكنك ان تختبر هذا إذا أردت أن تضيف السطر التالى للكود :


```

<script language="JavaScript">
<!--
/*
Code goes below this comment block
*/
var strFirst_Name = "Andy"
var strMiddle_Initials = "N"
var strLast_Name = "Other"
var strSpace = " "
var strFull_Stop = "."
var strFull_Name = strFirst_Name + strSpace + strMiddle_Initials
alert(strFull_Name)
strFull_Name = strFull_Name + strFull_Stop + strSpace + strLast_Name
alert(strFull_Name)
// -->
</script>

```

تشغيل هذا , يمكن أن تأكد القيمة للمتغير , كما هو معروض بالشكل



السطر الثانى يأخذ المتغير ويشرع إلى نهاية ما تبقى .

```
strFull_Name = strFull_Name + strFull_Stop + strSpace + strLast_Name
```

Processing Inputs

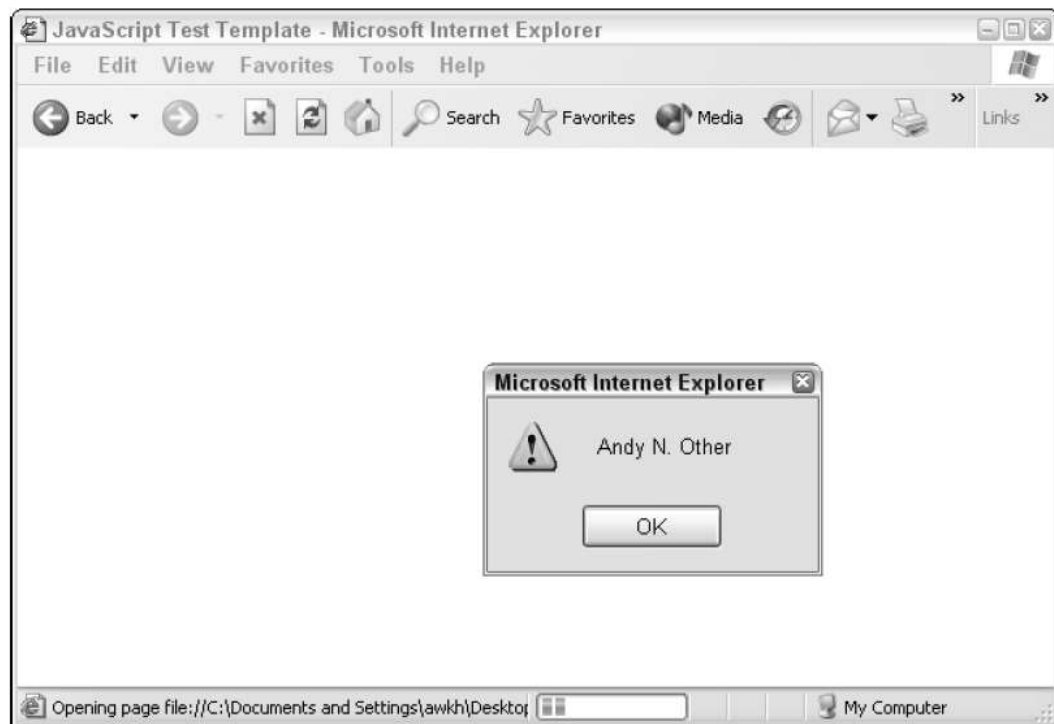
تجهيز المدخلات

المتغيرات هامة عند التعامل مع المدخلات في برنامجك , سنكمل في تفصيل عن المدخلات لاحقاً في هذا الكتاب ولكن إلى الآن سننظر فقط إلى كيف تستخدم المتغيرات للتعامل مع المدخلات والقليل عن كيف يمكن أن نصنع منها .

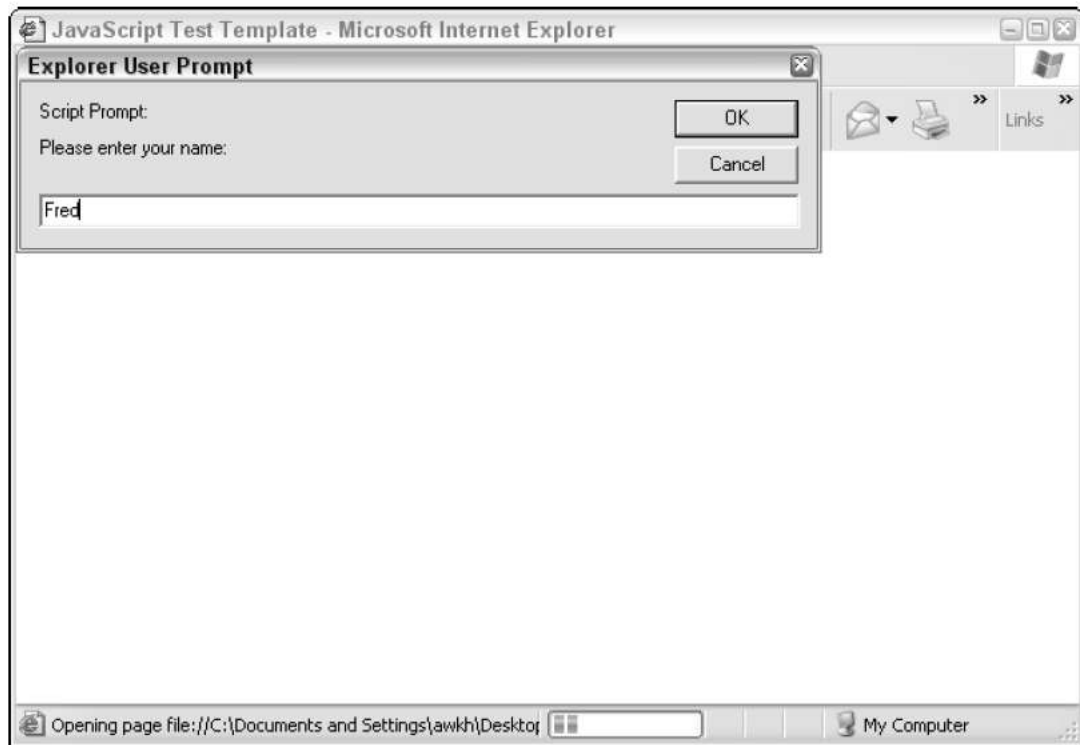
يمكن أن يأخذ كود JavaScript المدخلات من خلال إستخدام ما يسمى JavaScript Prompt . وها هو كود مبسط على Prompt [prompt هو أمر بإظهار رسالة تقوم بتلقي مدخلات المستخدم] .

```
<script language="JavaScript">
<!--
/*
Code goes below this comment block
*/
var strYour_Name = prompt('Please enter your name:', 'Enter your name')
var strGreeting = "Welcome " + strYour_Name + "!"
alert(strGreeting)
// -->
</script>
```

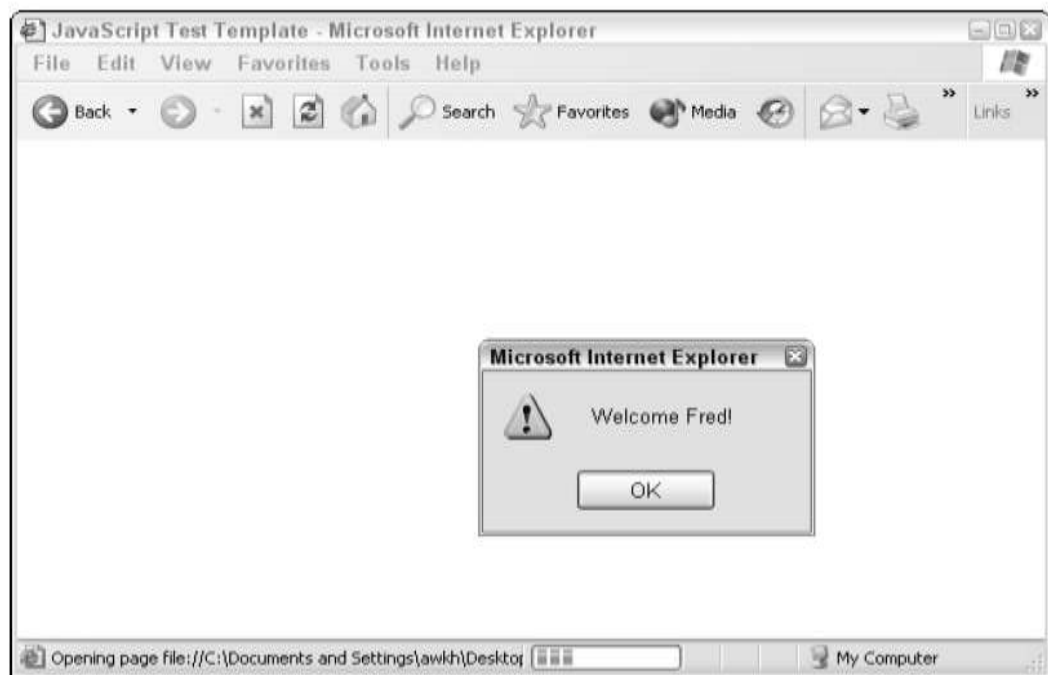
الشكل التالي يظهر كيف أن كل المتغيرات تأتي مع بعضها لبناء الإسم الكامل



تنفيذ هذا الكود , فيعرض Prompt ويسألك أن تدخل إسمك (اي نص سيعمل معك) . النص الذي تدخله سيجمل بواسطة المتغير strYour_Name .



هذا إذا تم ربطه مع التحيّة وتم تخزينه في متغير يسمى `strGreeting` ثم عرضها في رسالة `popup` كما يلي في الشكل:



بمجرد أن تأخذ المدخل من المستخدم وتم تخزينه في متغير يمكن بعدها معالجته كأي قيمة حرفية أخرى لديك في متغير

Variable Manipulation — Simple Math

دعنا ننهي هذا الفصل بإلقاء نظرة على بعض تجهيز المتغير وتنفيذ بعض الرياضيات البسيطة . على طول الطريق , سنقطف بعض المعلومات عن كيف تتصرف المتغيرات .

إلقى نظرة على الكود التالي :

```
<script language="JavaScript">
<!--
/*
Code goes below this comment block
*/
var intNum1 = 7
var intNum2 = 12
var intNum3 = 2
var intNum4 = 15
var intNum5 = prompt('Please enter a number:', 'Enter a number')
// -->
</script>
```

يمكننا الآن ان نبدل الكود ونبدأ بفعل بعض الرياضيات .

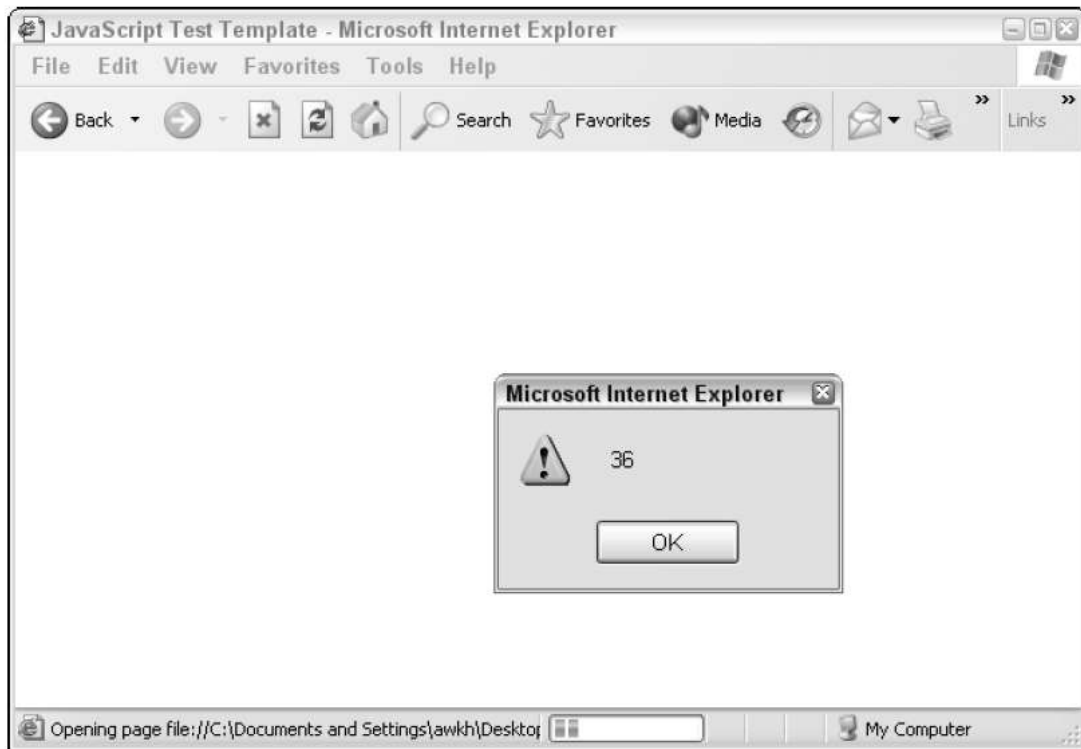
أبسط الرياضيات هي فقط إضافة أول أربع أرقام صحيحة :

```
<script language="JavaScript">
<!--
/*
Code goes below this comment block
*/
var intNum1 = 7
var intNum2 = 12
var intNum3 = 2
var intNum4 = 15
var intNum5 = prompt('Please enter a number:', 'Enter a number')
alert(intNum1 + intNum2 + intNum3 + intNum4)
// -->
</script>
```

هذا بسيط , ولكن نلاحظ كيف أننا لا نفعل أى شئ مع المدخلات من رسالة prompt , intNum5 . فدعنا نضيف ذلك إلى التنبيه الأخير ونرى ماذا سيحدث .

```
<script language="JavaScript">
<!--
/*
Code goes below this comment block
*/
var intNum1 = 7
var intNum2 = 12
var intNum3 = 2
var intNum4 = 15
var intNum5 = prompt('Please enter a number:', 'Enter a number')
alert(intNum1 + intNum2 + intNum3 + intNum4 + intNum5)
// -->
</script>
```

هذا فقط يضيف كل الأرقام ويعرض الناتج في رسالة popup , كما في الشكل ,



قم بتشغيل الكود الآن , أدخل رقم في رسالة prompt وألقى نظرة على الإجابة , ويجب أن ترى شيئاً غريب قد حدث .

ما يجب ملاحظته انك مهما قمت بالإدخال في prompt , فبدلاً من إضافته إلى مجموع الإرقام , فيتم ربطه برقم المجموع . لذا , إذا كان مجموع intNum1, intNum2, intNum3 و intNum4 هو 36 . فمهما ادخلت في prompt يتم ربطه لهذا , لذلك إذا أدخلت الرقم 100 سيكون المجموع 36100 . هذا واضحاً أنه غير صحيح وخطأ في الكود . ويكشف عن وجود مشكلة محتملة في النظام (في هذه الحالة في المترجم interpreter, ولكن يمكن أن يكون المفسر Compiler) يترجم نوع البيانات المخزنة في المتغير.

دعنا نفحص ما حدث .

في هذه الحالة , يأخذ الكود ما نريد أن نترجمه كأرقام صحيحة ويترجمهم بدلاً من ذلك كقيم حرفية . المشكلة ان معامل الإضافة ومعامل الربط نفس الشيء .

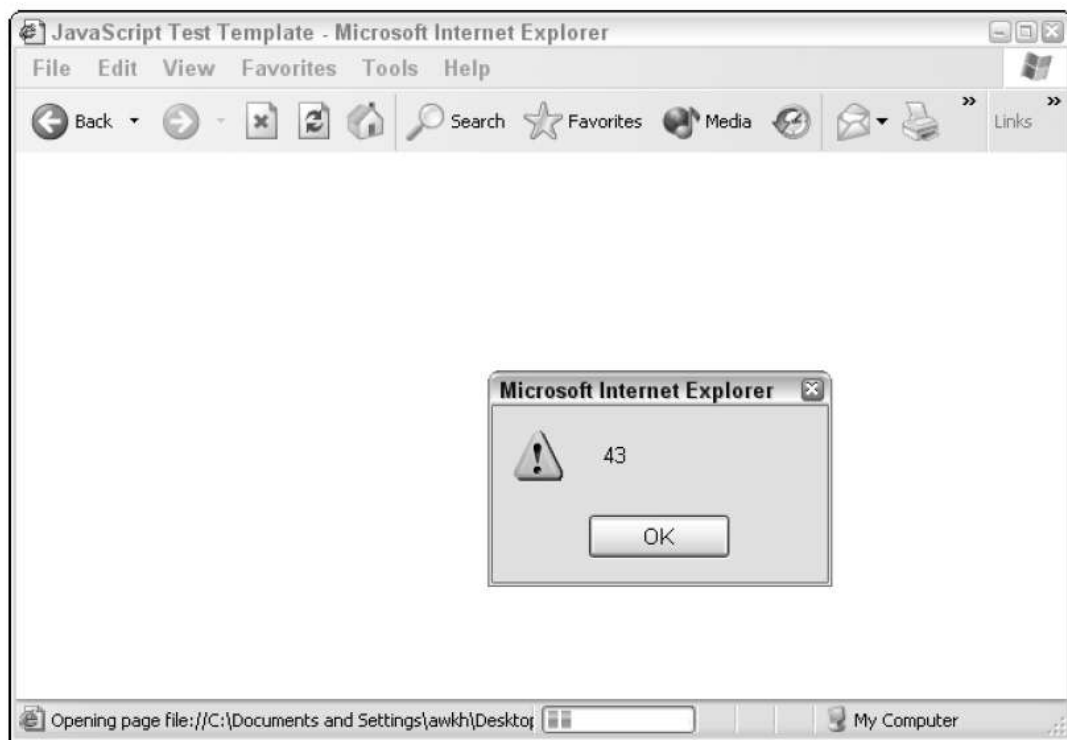
هناك طرق للتغلب على هذا . مع JavaScript واحدة من أسهل الطرق للتغلب على هذا هو أخذ القيمة المخزنة في المتغير وبضربها بالرقم 1 . لا يؤثر هذا على الرياضيات ولكن يجرى تغيير على كيفية عرض القيمة للمتغير , ويعالجها بعد ذلك كرقم وليس قيمة حرفية .

```

<script language="JavaScript">
<!--
/*
Code goes below this comment block
*/
var intNum1 = 7
var intNum2 = 12
var intNum3 = 2
var intNum4 = 15
var intNum5 = prompt('Please enter a number:', 'Enter a number')
intNum5 = intNum5 * 1
alert(intNum1 + intNum2 + intNum3 + intNum4 + intNum5)
// -->
</script>

```

الشكل التالي يظهر الكود و قد تمت ترجمته الآن القيمة كرقم بشكل صحيح .



تذكر أن تفحص هذه الأنواع من المشاكل في أى مخرجات يعطيك إياها الكود . وينبغي أن تكون بمثابة تذكير صارخ لك إلى كم هو سهل أن تحصل على مخرجات لا تتوقعها .

Summary

الملخص

هذا كافٍ إلى الآن , لقد قمت بتغطية المزيد من الأساسيات في الفصل الآتي ستستخدم الكثير من المتغيرات في لغات متنوعة . إلى الآن , قد إستلمت معلومات كافية عن المتغيرات ليتمكنك أن تتحرك مساحات جديدة وتتقدم مع الكود الخاص بك .

في هذا الفصل , ألقيت نظرة على العمود الفقري لمعظم لغات البرمجة — المتغير . يحمل المتغير البيانات ويستخدم في أوامر العمليات في الكود لتجهيز البيانات . تعلمنا أن تجعل المتغيرات واضحة , لا أسماء كريهة , وإضافة تعليق إلى الكود لحذف أى غموض ويجعل الكود أسهل للتبع .

كان هناك الكثير من المعلومات في هذا الفصل والكثير من الكود , أقترح أن تنظر إلى أمثلة الكود وتجربها مع ما تضيفه أنت . تذكر انه كلما كانت تجربتك مع الكود كلما تعلمت أكثر . كلما ستتذكر أكثر , وستكون أسرع في التقدم .

التجربة , التجربة , التجربة !

7

The Structure of Coding

تركيبة الكود

حتى الآن بحثنا في كود بسيط جميل , ولا شيء عن الكود البسيط سوا انه يمكن أن يفعل مجرد أشياء بسيطة. إذا أردت من الكود أن يفعل المزيد , تحتاج أن تعرف أكثر عن انماط الكود التي تبني على ما بحثت فيها حتى الآن وترتقى إلى مستوى أعلى .

في هذا الفصل , سنلقى نظرة على كيف يمكن لك أن تحصل على كود ينفذ عمل حقيقي . ولتكون قادراً على فعل هذا تحتاج إلى إضافة تركيبات أو هياكل إلى الكود الخاص بك ليكون الكود قادر أن يفعل المزيد .

سنقوم بتغطية أربعة أنواع من الهياكل أو التركيبات التي تضيف القوة للكود :

- ☐ Functions (الدوال)
- ☐ Conditionals (الأوامر الشرطية)
- ☐ Loops (اوامر التكرار)
- ☐ Arrays (المصفوفات)

معرفة كيف تستخدم هذه التركيبات بتحكم سيمكنك من كتابة كود يكون قادراً على تنفيذ من المهام الكثيرة اكثر من الكود البسيط الذي رأيناه حتى الآن .

The Purpose of Structure

الغرض من التركيبات

من وجهة نظر التعلم لكتابة الكود, لا شيء أفضل من الكود البسيط . فإنه جيد التتبع , سريع الكتابة , يتيح لك تجربة عالية للحصول على شيء ما للعمل , ويعطيك الممارسة لكتابة كود حقيقي والدخول إلى عقلية الكود . ومع ذلك , أخيراً (في كثير من الأحيان يكون سريع) ستجد أنك تريد فعل أشياء تكون أكثر تعقيداً أو تعكس ما يمكن ان تنجزه من خلال الكود البسيط .

ولكن هناك سبب آخر هام لإضافة التركيبات إلى الكود — لأنه يسمح لك بفعل المزيد من العمل مع أقل كود . لنرى , مشكلة أخرى مع الكود البسيط في كونه غير محكم وكلما أردت المزيد من الكود كلما كانت الحاجة إلى كتابة كود أطول. فكلما كان الكود الذى كتبته أطول , كلما كان هناك المجال الأكبر لوقوع الأخطاء وزيادة صعوبة إيجاد هذه الأخطاء .

وكما يمكنك تنظيم الرسائل في الكلمات والجمل والفقرات , تحتاج إلى تنظيم التعليمات البرمجية في هياكل لسهولة إدارتها .

Benefits

المنافع

كلما تم التقدم خلال هذا الفصل , ستلاحظ ان التركيبات التى تم تطبيقها على الكود لديها الكثير من المنافع , وإليك البعض الذى لا شك فى انك قد لاحظته :

- كود أقل يساوى زيادة قوة .
- تجعل الكود أسهل للقراءة والتتبع .
- تمكّنك من تقطيع الكود فى أجزاء منطقية .
- يجعله من السهل لإكتشاف مكان وجود الإخطاء.
- يجعل إعادة إستخدام الكود أسهل .

Examining Structure

إختبار تركيبة الكود

و سنستخدم لغة C++ لنرى التركيب , لذا تحتاج التالى يكون جاهز فى النظام الخاص

- ❑ A text editor (Windows Notepad, UltraEdit, أو شئ مشابه لذلك) محرر النصوص
- ❑ A C++ compiler (عد للفصل الخامس لمزيد من المعلومات حول إعداداته للعمل)

Quick Introduction to C++

مقدمة سريعة إلى C++

قبل أن ننتقل إلى أبعد من هذا, دعنا نأخذ مقدمة سريعة لـ C++ لتحصل على السرعة الكافية للمتابعة معنا . إذا كنت قد عملت خلال الكود من دى قبل , إذاً فمبادئ تركيبات C++ التى ستراها هنا لن تكون تحدياً كبير لك

Examine Source Code

فحص مصدر الكود

أساس تركيب كود C++ الذى سنعمل معه يظهر أمامك هنا

```
// C++ code template
#include <iostream.h>
void main()
{
    // Code goes here!!!!
}
```

فربما ترى المزيد من الأشياء الجديدة هنا . فلا تقلق , وهنا ما يعنى الجميع . دعنا نعمل معه سطر بسطر .

```
// C++ code template
#include <iostream.h>
void main()
{
    // Code goes here!!!!
}
```

السطر الأول هو تعليق بسيط . فقط كما فى JavaScript , أى شئ بعد // سيتم إهماله بواسطة المفسر compiler الذى سيحول الكود إلى برنامج .

```
// C++ code template
#include <iostream.h>
void main()
{
    // code goes here!!!!
}
```

السطر الثانى هو أمر للمفسر وليس حقيقةً فيه أى شئ يفعله مباشرةً مع الكود الذى سنكتبه . يخبر السطر المفسر أن يشمل الملف, `iostream.h`, فى الكود الذى تكتبه . يسمى هذا ملف الرأس `header file` ويضاف للكود ليوفر الدعم لـ IO, أو Input , Output إدخال وإخراج . بدون هذا , لا يمكنك أن تعمل مع إدخال لوحة المفاتيح وإخراج الشاشة .

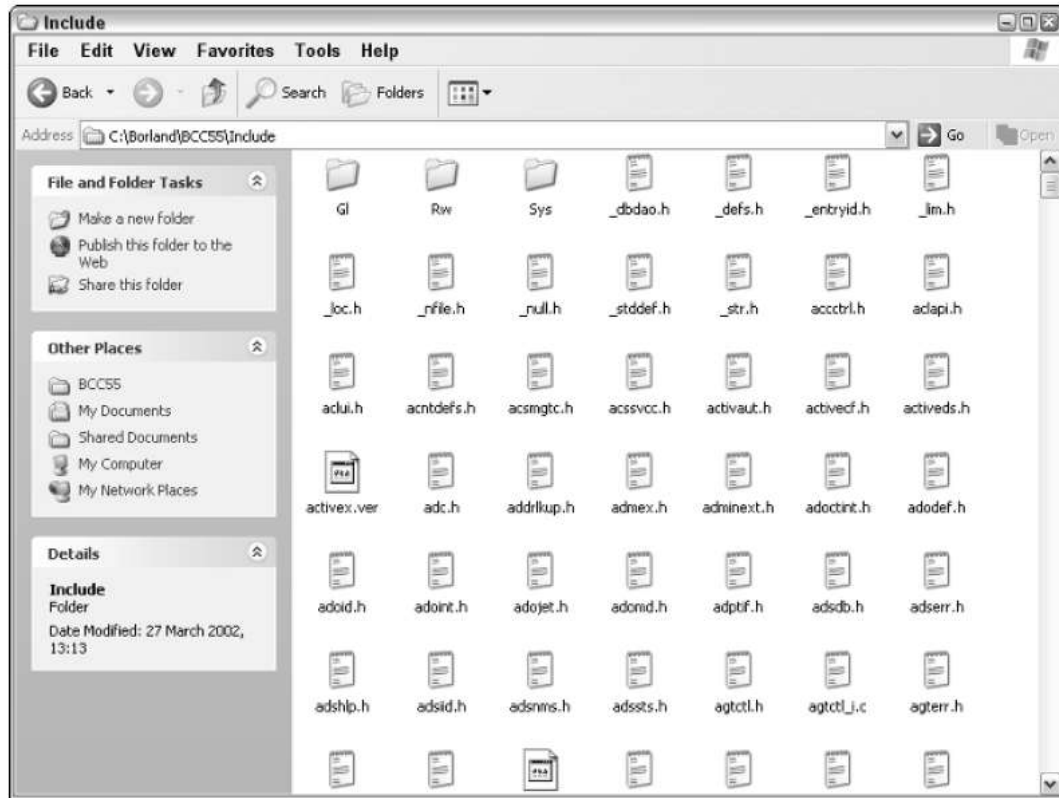
شئ آخر من المحتمل أن تستخدمه يسمى `string.h` , ويضيف الدعم للعمل مع القيم الحرفية للكود . فإذا أردت يمكنك ان تضيف هذا الكود حتى إذا لم يكن عليك أن تستخدمها هنا , أكتب التالى :

```
// C++ code template
#include <iostream.h>
#include <string.h>
void main()
{
    // Code goes here!!!!
}
```

إذا صادفت أن تستخدم أى ملفات رأس لا تحتاجها , سيتجاهلهم المفسر (سأزودك حتماً بهذا لاحقاً فى هذا الفصل) .

إذا كان لديك فضول فمن المحتمل أن تجد كل رؤس الملفات هذه (`iostream.h`, `string.h` والكثير) مخزنة داخل الحاسب الشخصى مع المفسر . يمكنك رؤية الشكل التالى

لا تلتفت فيما تعنى هذه الملفات وماذا تحتوى . فقط فكر فيهم كأنهم كود مصنع من قبل يمكنك ان تضيفه إلى تطبيقك عندما تحتاج العمل معهم لداء مهام أساسية لك . ولا تحتاج ان تعرف كيف يتم عملهم — هذه وظيفة المفسر !



```
// C++ code template
#include <iostream.h>
void main()
{
    // code goes here!!!!
}
```

يستخدم هذا السطر المظلل لتعريف شئ ما سنقوم بتغطيته في لحظات — وهو الدالة . هناك جزئين من الجملة:

- `main()` . وهذا يعلن الاسم للدالة , فى هذه الحالة , تسمى `main` . الأقواس فى النهاية غرضها سنراها قريباً .
- `void` . وهى كلمة محجوزة تستخدم لتحديد ان الدالة التى سنكتبها ستأخذ القيم الممررة إليها من دالة أخرى (تسمى `parameters` او معلمات) ولا تقوم بتمرير هذه المعلمات إلى دالة أخرى . لا حاجة إليه , ولكن تساعد فى معرفة ماذا تفعل الدالة وسواء كانت مرتبطة بدالة أخرى .

وتكمن هذه المعلومات فى هذه المرحلة — والأمثلة والممارسات السابقة مؤخراً ستجعل هذا واضحاً .

```
// C++ code template
#include <iostream.h>
void main()
{
    // Code goes here!!!!
}
```

هذا السطر هو بداية الكود الفعلي للدالة . فلا بد من بدايتها بفتح أقواس متعرجة { } . ولا يجب عليها أن تكون في سطر واحد منفصل , ولكن هذا يساعدك ان تعرف انها هناك وتجنباً للأخطاء .

```
// C++ code template
#include <iostream.h>
void main()
{
    // Code goes here!!!!
}
```

التالى هو ما سيذهب إليه الكود . وليس هناك كود , فقط تعليق , ولكن يبين ما يسير إليه الكود . يمكن أن تضيف الكثير من الأسطر من الكود هنا كلما زادت الحاجة لذلك , وأيضاً إستخدم التعليقات لإخبار نفسك (فى المستقبل) ما يفعله الكود .

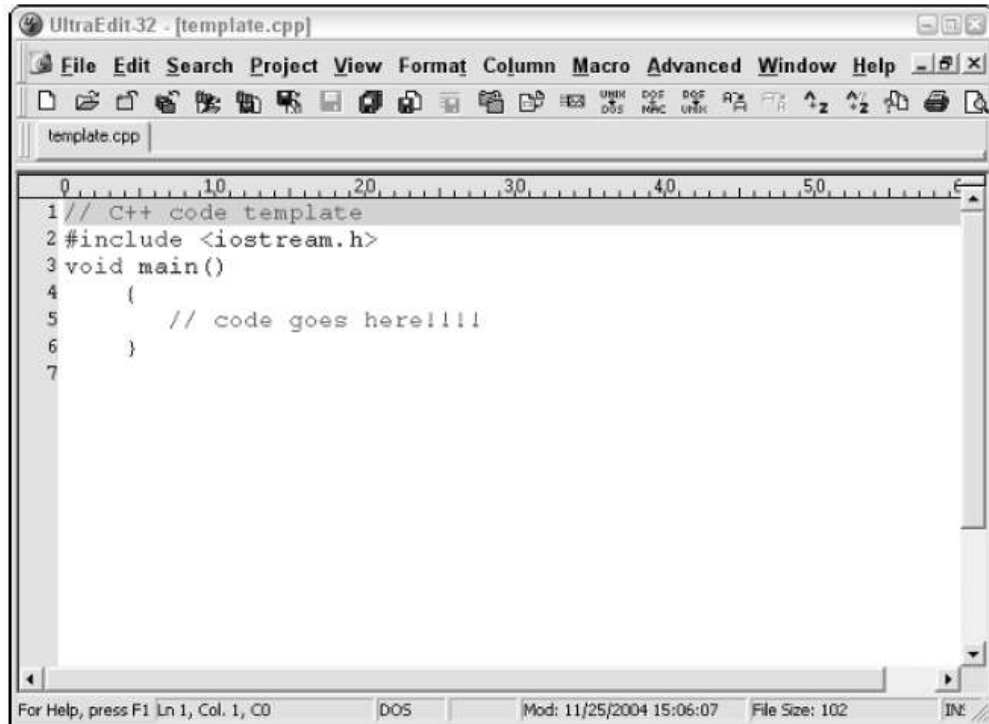
```
// C++ code template
#include <iostream.h>
void main()
{
    // Code goes here!!!!
}
```

هذه هي نهاية الدالة . وهو قوس متعرج للإغلاق , وهو مطلوب للضرورة . فقط مع إفتتاح القوس المتعرج ولا يشترط ان تكون في سطر منفصل , ولكن تساعد إن كانت كذلك .

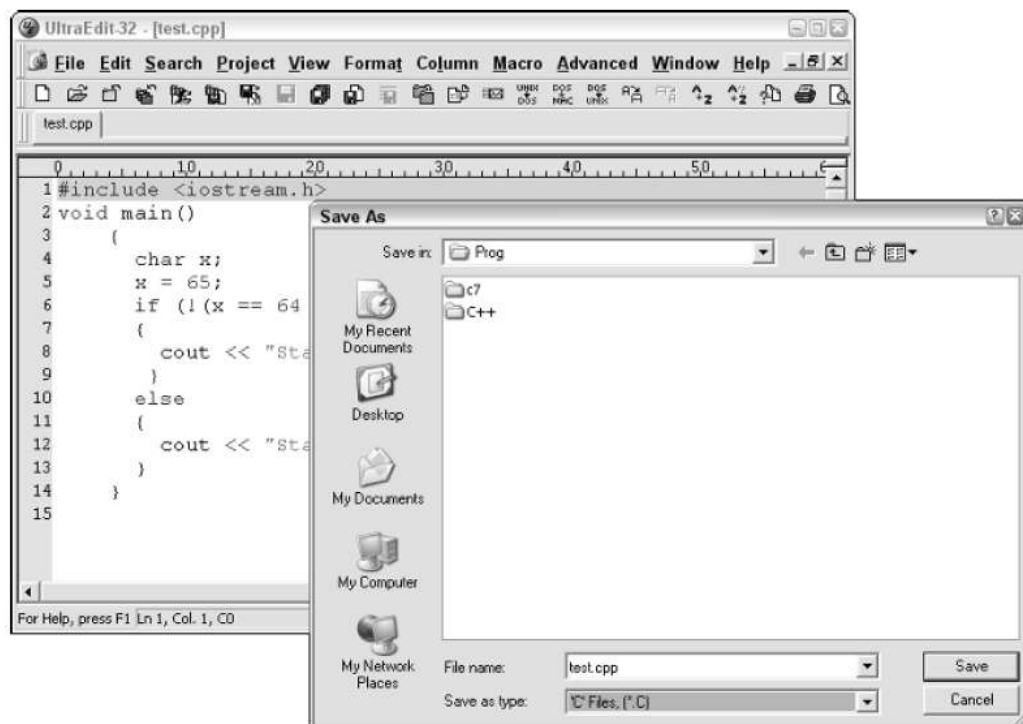
Compiling C++

تفسير C++

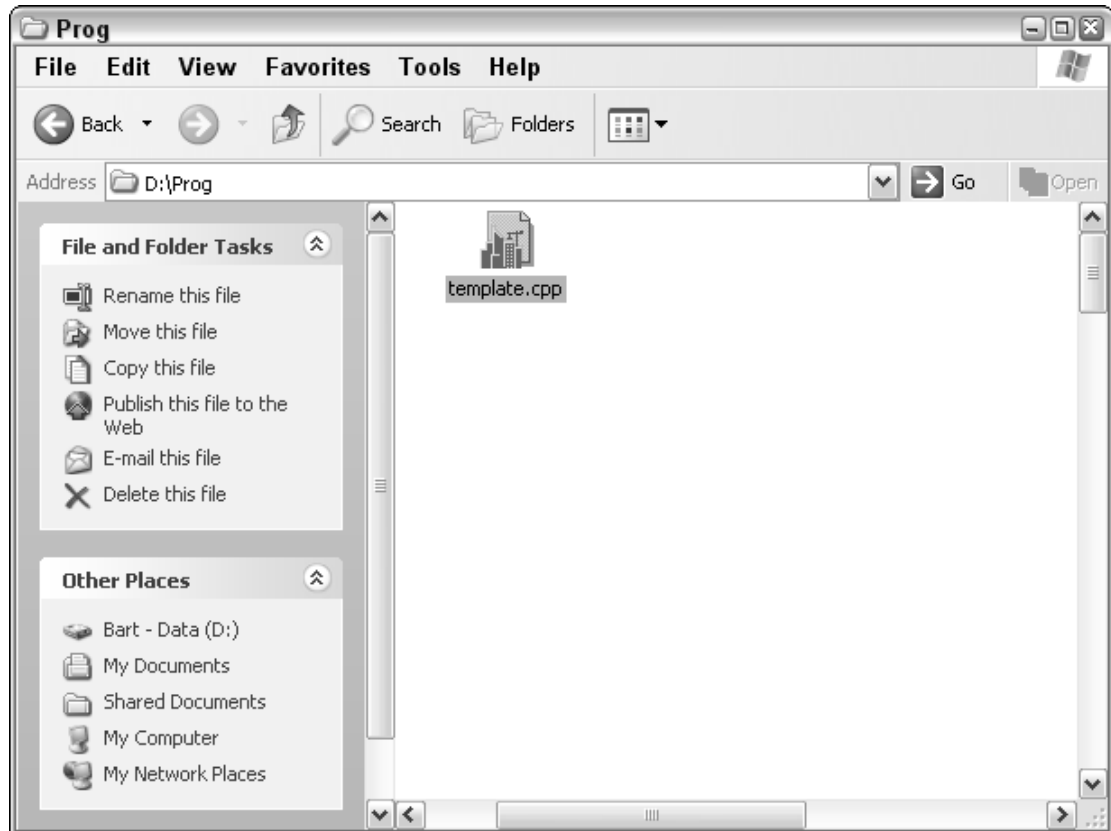
يمكننا الآن ان نفرس هذا الكود . ولفعل هذا نحتاج إلى المفسر borland . إكتب الكود السابق فى محرر نصوص (إنظر الشكل) وإحفظه .



عندما تحفظ هذا الكود تذكر أن تعطيه إمتداد الملف `.cpp`. وهذا يخبر المفسر أن هذا الملف يحتوى على مصدر كود لـ `c++`. مصدر الملف يعرض فى الشكل , وهذا يسمى `test.cpp`

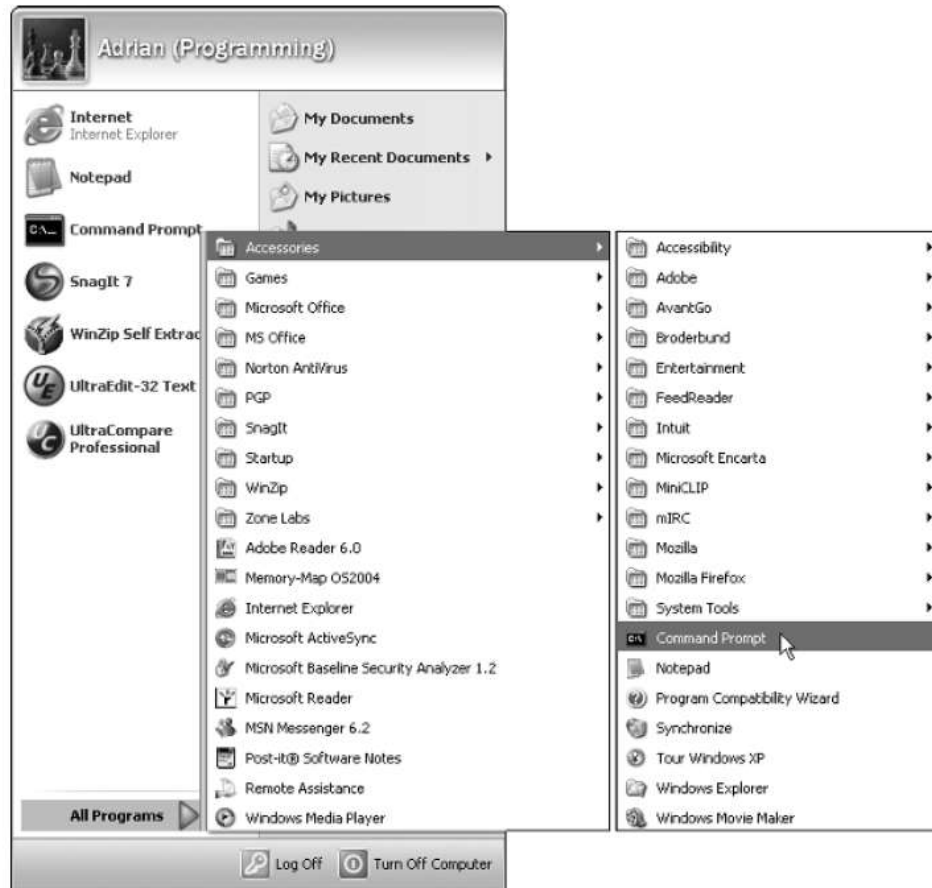


لقد قمت بحفظ هذا الملف فى مجلد يسمى `prog` فى القرص (`d:\`) لجهازى, وسميته `template.cpp`.
 ويظهر بوضوح فى الشكل التالى :



الآن إلى تفسيره , فالتفسير سهل — فقط تحتاج أن نستدعي المفسر وتخبره بمكان مصدر الكود .

اولاً إفتح نافذة Command prompt أو Command Window . الإختصار له عادةً تجده في مجلد Accessories في قائمة start (إضغط start ثم All program) أو إضغط start , program لكل الإنظمة غير windows XP , ثم إذهب إلى Accessories وإضغط على Command Prompt (إنظر الشكل)



يمكنك ان تضغط على قائمة Start , ثم Run وتكتب التالي :

cmd

بعدها اضغط OK , كلا الطريقتين ستحضر لك النافذة , كما هو موضح بالشكل

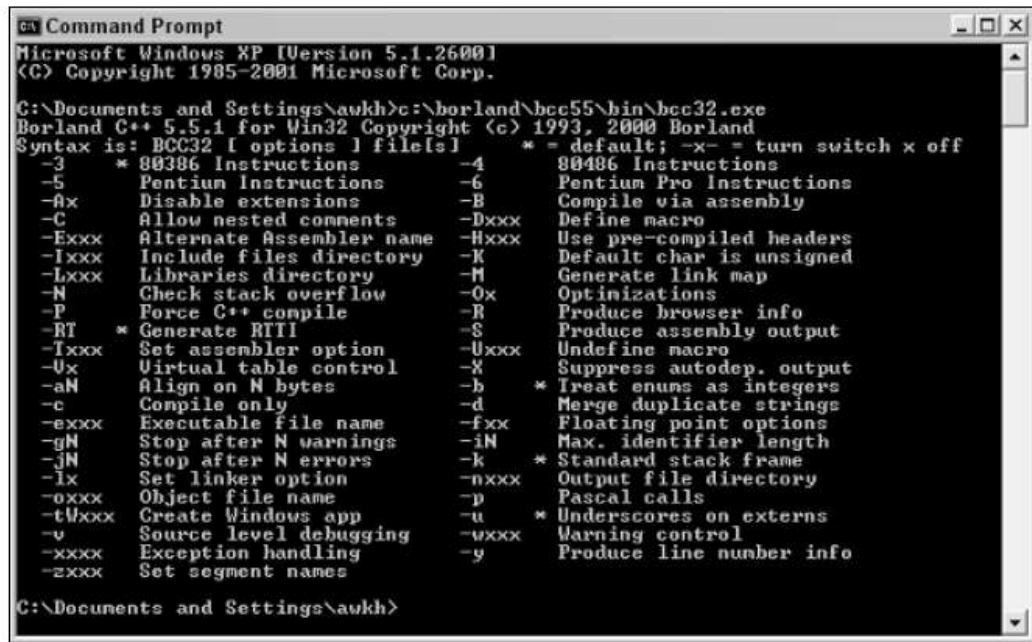


الآن إستدعى المفسر compiler وأشر إلى مصدر الكود , ولعمل هذا تحتاج ان تعرف شيئين :

- أين قمت بتنصيب المفسر Compiler.
 - أين مكان ملف مصدر الكود Source Code.
- فإذا قمت بتنصيب المفسر فى الموقع الافتراضى , بعدها سيكون مخزن فى

```
c:\borland\bcc55\bin\bcc32.exe
```

وإذا لم تفعل , بعدها سيكون عليك أن تذهب فى البحث عنه ! وفى كلتا الحالتين , هذا هو المسار للمفسر واكتب هذا فى (أو إنسخه فى) نافذة Command Prompt وسينفذ الكود (إنظر الشكل)




```
Command Prompt
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\aukh>c:\borland\bcc55\bin\bcc32.exe
Borland C++ 5.5.1 for Win32 Copyright (c) 1993, 2000 Borland
Syntax is: BCC32 [ options ] file[s]
          * = default; -x- = turn switch x off
-3      * 80386 Instructions          -4      80486 Instructions
-5      Pentium Instructions          -6      Pentium Pro Instructions
-Ax     Disable extensions           -B      Compile via assembly
-C      Allow nested comments        -Dxxx   Define macro
-Exxx   Alternate Assembler name     -Hxxx   Use pre-compiled headers
-Ixxx   Include files directory       -K      Default char is unsigned
-Lxxx   Libraries directory           -M      Generate link map
-N      Check stack overflow          -Ox     Optimizations
-P      Force C++ compile            -R      Produce browser info
-RT     * Generate RTTI              -S      Produce assembly output
-Txxx   Set assembler option         -Uxxx   Undefine macro
-Ux     Virtual table control         -X      Suppress autodep. output
-aN     Align on N bytes              -b      * Treat enums as integers
-c      Compile only                 -d      Merge duplicate strings
-exxx   Executable file name         -fxx    Floating point options
-gN     Stop after N warnings         -iN     Max. identifier length
-jN     Stop after N errors           -k      * Standard stack frame
-lx     Set linker option             -nxxx   Output file directory
-oxxx   Object file name             -p      Pascal calls
-tWxxx  Create Windows app           -u      * Underscores on externs
-v      Source level debugging        -wxxx   Warning control
-xxxx   Exception handling            -y      Produce line number info
-zxxx   Set segment names

C:\Documents and Settings\aukh>
```

ومع ذلك , هذا لا يفعل شيئاً سوا أنه يحضر قائمة من التعليمات عن كيفية إستخدام المفسر . وإذا أردت ان يتم تفسيره فقط , ثم أشر إلى الملفات . ولفعل هذا , إكتب المسار للمفسر متبوعة بمسار ملف مصدر الكود . كما موضح بالشكل



```
Command Prompt

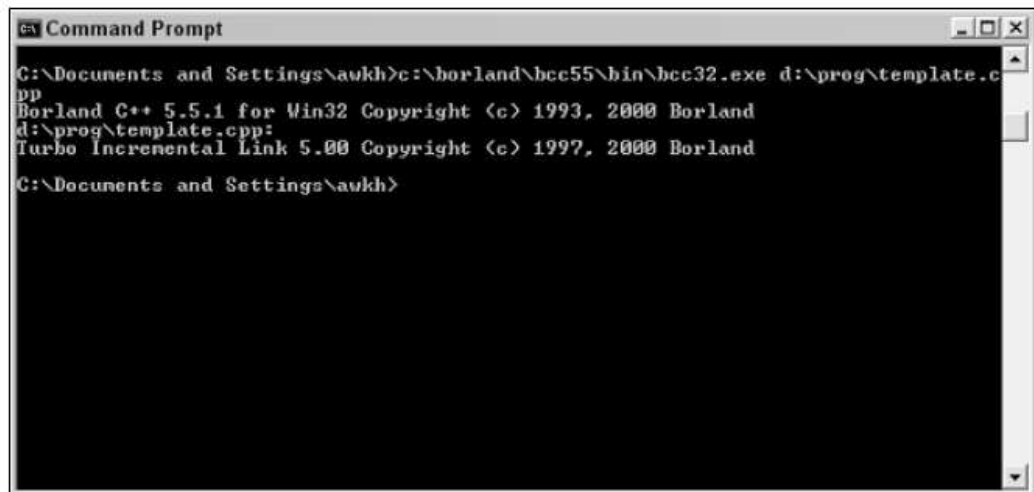
C:\Documents and Settings\aukh>c:\borland\bcc55\bin\bcc32.exe d:\prog\template.c
pp

C:\Documents and Settings\aukh>
```

منذ أن كان مسارنا لملف مصدر الكود هو `d:\prog\template.cpp` , وكل ما على فعله كتابة هذا

```
c:\borland\bcc55\bin\bcc32.exe d:\prog\template.cpp
```

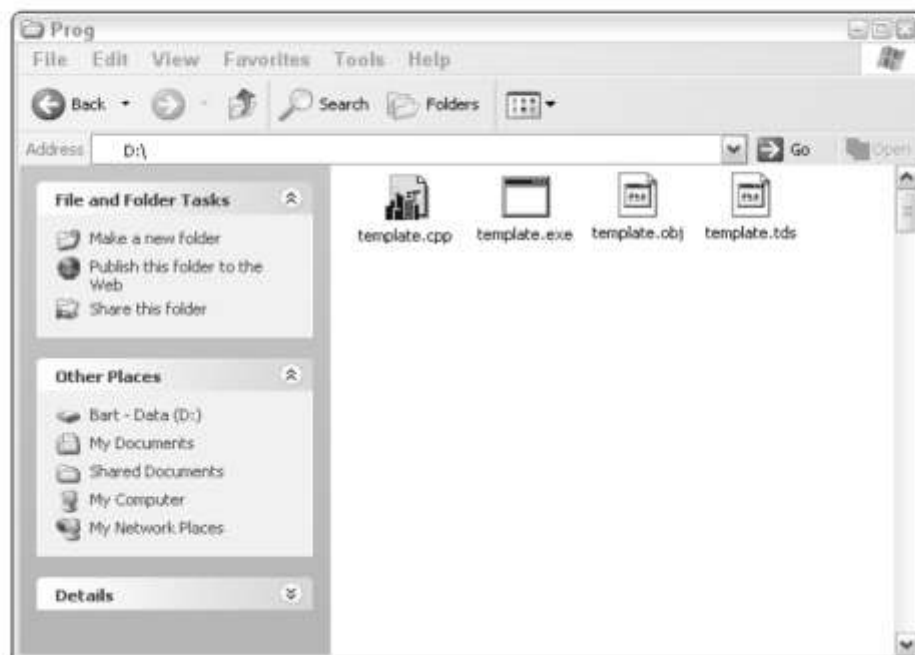
وإضغط Enter .
والآن يفسر الملف (إنظر الشكل)



بعد إتمام التفسير , (ينبغي أن يأخذ ثوانى قليلة فقط) , ويمكن بعدها فحص المجلد الذى قد تم تخزين مصدر الكود به , أترى شيئاً هناك ؟ لا , من المحتمل لا , لأن الملف فى المحتمل فى المسار الذى يكون فيه مصدر الكود — فى هذه الحالة هو `d:\` .

ستجد الآن ملفات قليلة جديدة هناك (الشكل التالى) وسيسمون بإسم `template` وسيكونون ثلاثة

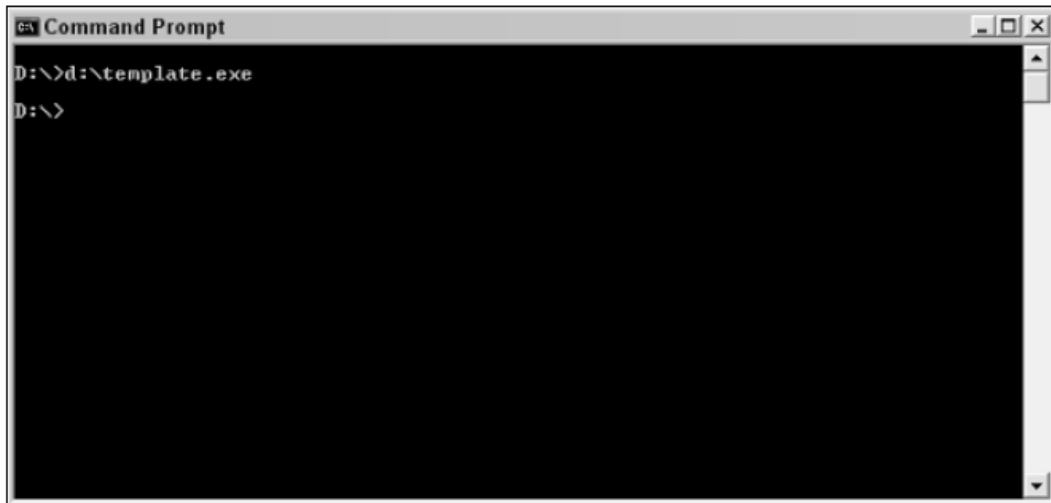
- ☐ `template.exe`
- ☐ `template.obj`
- ☐ `template.tds`



كل ما ستكون مندهشاً به هو الملف التنفيذي `template.exe`, Executable File . ويمكن تشغيله من خلال الضغط مرتين أو من خلال `command prompt` . فإذا قمت بتشغيله من خلال الضغط مرتين , إذاً فستظهر رسالة وتختفي , ولتشغيله من خلال نافذة `Command prompt` أكتب :

```
d:\template.exe
```

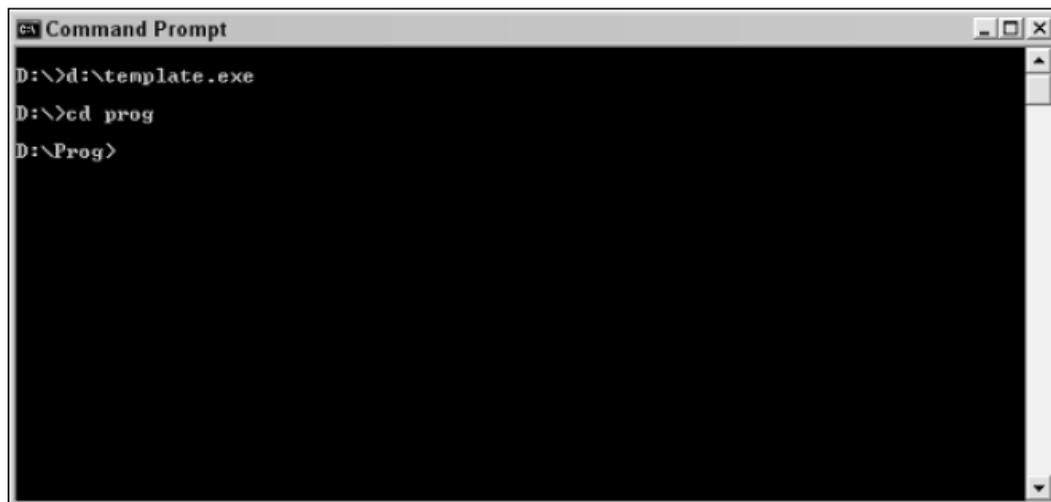
ثم اضغط `enter` . سيعمل الكود ولكن لن يفعل أى شئ : (أنظر الشكل)



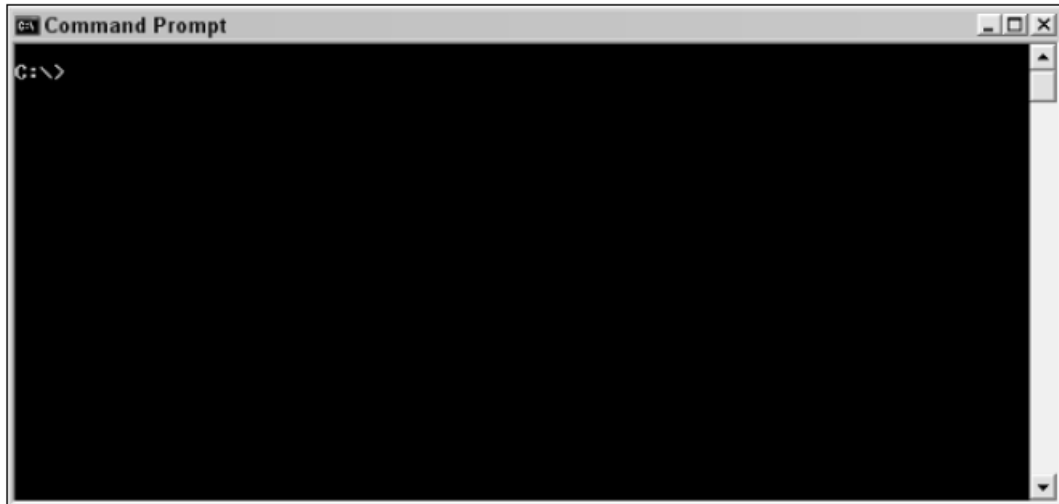
ولتحديد المكان المحدد لمخرجات الملف إلى المجلد المناسب , تحتاج إلى تنقله إلى المجلد قبل تشغيل المفسر . يمكن عمل ذلك بكتابة التالي :

```
cd prog
```

ثم اضغط `Enter` (كما فى الشكل)



هذه القضايا تغير أمر المسار (المسار هو اسم آخر للمجلد) . يجب أن تكون فى القرص المناسب لفعل ذلك , وبالتالي , سيجب عليك أن تغير اسم القرص أيضاً . لذلك , إذا بدأت مع نافذة `command prompt` فى `c:\` (كما ترى بالشكل) إذاً ينبغى أن أغير إلى `d:\prog` بكتابة التالي

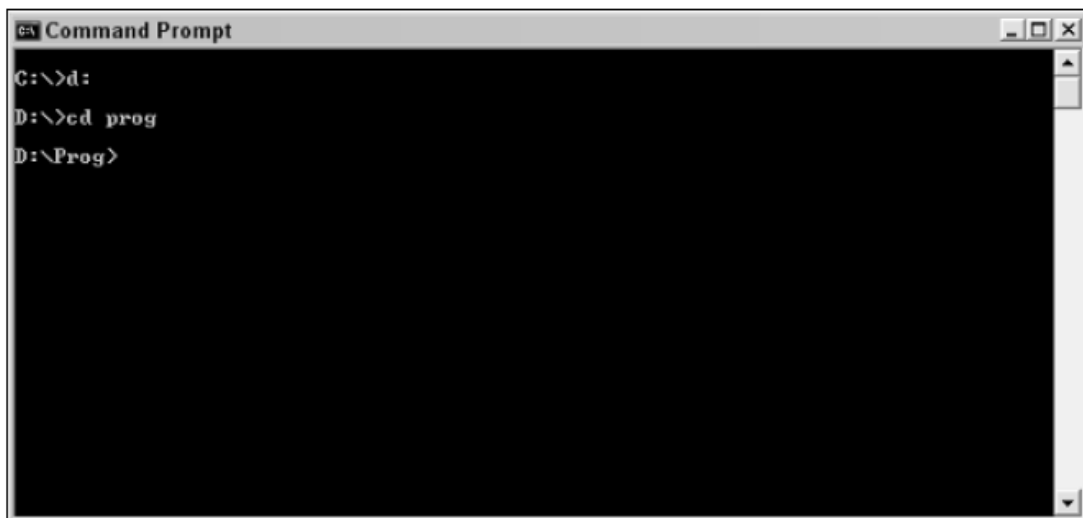


ثم اضغط Enter , متبوعة بـ

```
cd prog
```

متبوعة بالضغط على enter مرة أخرى .

الشكل يتأكد إن كان في المجلد المناسب



الآن كل ما عليك كتابته هو

```
c:\borland\bcc55\bin\bcc32.exe template.cpp
```

ومخرجات الملف قد وضعت في d:\prog تلقائياً

يمكنك إختصاره أكثر , لدى windows القدرة على تخزين معلومات المسار خلال نظام التشغيل , لذا . عندما تحاول تشغيل ملف تنفيذي يقوم بفحص كل المسارات المحدده لهذا الملف التنفيذي ويقوم بتشغيله إن وجده . فإذا التطبيق لا يمكن إيجاده إذا رسالة عطل سيتم إظهارها .

ليتم إعداد ذلك في windows xp , إفعّل التّالي (تحتاج أن تقوم بتسجيل الدخول تحت إسم Administrator
للتّمكن من فعل ذلك)

1. إضغط بيمين الفأرة على My Computer، ثمّ إضغط على Properties.
2. إضغط على الجزء الخاص -Advanced.
3. إضغط على Environment variables .
4. إضغط على PATH، ثمّ إضغط على Edit لتضع القيمة البديله .
5. إذهب إلى نهاية المسار الموجود وأضف العلامة ; (إذا لم تكن موجودة) , ثم أضف المسار للمفسر.

c:\borland\bcc55\bin

ولتشغيل المفسر الآن تحتاج أن تكتب

bcc32

ثمّ إضغط Enter . سهل جداً، هذا الأمر الجديد يظهر في الفعل في الشكل التّالي

```
Command Prompt
D:\Prog>bcc32
Borland C++ 5.5 for Win32 Copyright (c) 1993, 2000 Borland
Syntax is: BCC32 [ options ] file(s)
-3 * 80386 Instructions          -4 * 80486 Instructions
-5 Pentium Instructions         -6 Pentium Pro Instructions
-Ax Disable extensions         -B Compile via assembly
-C Allow nested comments      -Dxxx Define macro
-Exxx Alternate Assembler name -Hxxx Use pre-compiled headers
-Ixxx Include files directory  -K Default char is unsigned
-Lxxx Libraries directory      -M Generate link map
-N Check stack overflow        -Ox Optimizations
-P Force C++ compile          -R Produce browser info
-RT * Generate RTTI           -S Produce assembly output
-Txxx Set assembler option     -Uxxx Undefine macro
-Ux Virtual table control      -X Suppress autodep. output
-aN Align on N bytes          -b * Treat enums as integers
-c Compile only               -d Merge duplicate strings
-e Exeutable file name        -fxx Floating point options
-gN Stop after N warnings     -iN Max. identifier length
-jN Stop after N errors       -k * Standard stack frame
-l Set linker option           -nxxx Output file directory
-oxxx Object file name        -p Pascal calls
-tUxxx Create Windows app     -u * Underscores on externs
-v Source level debugging     -wxxx Warning control
-x Exception handling          -y Produce line number info
-zxxx Set segment names

D:\Prog>
```

تهانينا , لقد قمت بكتابة وتفسير تطبيق C++ . أحسنت ! , حسناً , فهو لا يفعل شئ حتى الآن , ولكن في نفس الوقت لقد انجزت الكثير .لقد قمت بـ :

- كتابة الكود .
- حفظه في التنسيق المناسب .
- حفظه في المكان المناسب.
- استخدام نافذة Command Prompt .
- تشغيل المفسر compiler .
- توجيه المفسر لمكان مصدر الكود .
- إختبار الملف التنفيذي الذي تم إنشاؤه .
- تعلم كيف تبحر خلال الملفات والمجلدات من خلال Command Prompt .
- تعديل معلومات المسار Path في windows .

اعتقد أن هذا كثير جداً !

Functions

الدوال

إذا كان هذا أول تطبيق تكتبه في C++ . إذاً الشيء التالي الذي تريد ان تفعله ان تكتب المزيد من الكود وتقوم بتفسيره . انا لا ألوم عليك , فكتابة كود متعة وإدمان , وخاصةً عندما تحرز تقدم وتحصل على نتائج .

تسلح بكل المعلومات الموجودة , يمكننا الآن ان نبدأ بإلقاء نظرة على ما حددنا النظر إليه في السابق – الدوال .

لقد رأيت الدوال بالفعل :

```
void main()
{
    // Code goes here!!!!
}
```

هذه الدالة فارغة , ولكن الدالة مع ذلك , لها اسم هو main , ولها نطاق تحتوى الكود بداخله , على الرغم من أن هذه لا تفعل ذلك .

الدوال التي تسمى main لها معنى خاص في C++ — فهم الدوال التي سيتم تشغيلها تلقائياً . لذا فأى كود يوضع بداخلهم يتم تشغيله عند تشغيل الملف التنفيذي (بعدما يتم تفسيره بالطبع) .

يمكننا الآن إضافة بعض الكود ونبدأ بصنع الملف التنفيذي الذى سيقوم بعمل شئ ما !

دعنا نبدأ مع المحبوب والشعبى للأبد كود ! Hello , World .

```
// C++ code template
#include <iostream.h>
void main()
{
    cout << "Hello, World!" << endl;
}
```

هناك فقط سطر واحد فى هذا الكود يختلف عن ما رأينا من دى قبل :

```
// C++ code template
#include <iostream.h>
void main()
{
    cout << "Hello, World!" << endl;
}
```

هناك خمسة أجزاء لهذا السطر نحتاج أن نلقى نظرة عليهم:

```
cout
<<
"Hello, World!"
endl
;
```

دعنا نبدأ مع cout . هذه هي Instruction [احد التعليمات أو ما تسمى تعليمه أو يمكن أن تسميها أمر [(يحدث هذا لإحتواءه مع ملف iostream.h) الذى يقوم بإخراج النص إلى الشاشة — لا شئ معقد هناك .

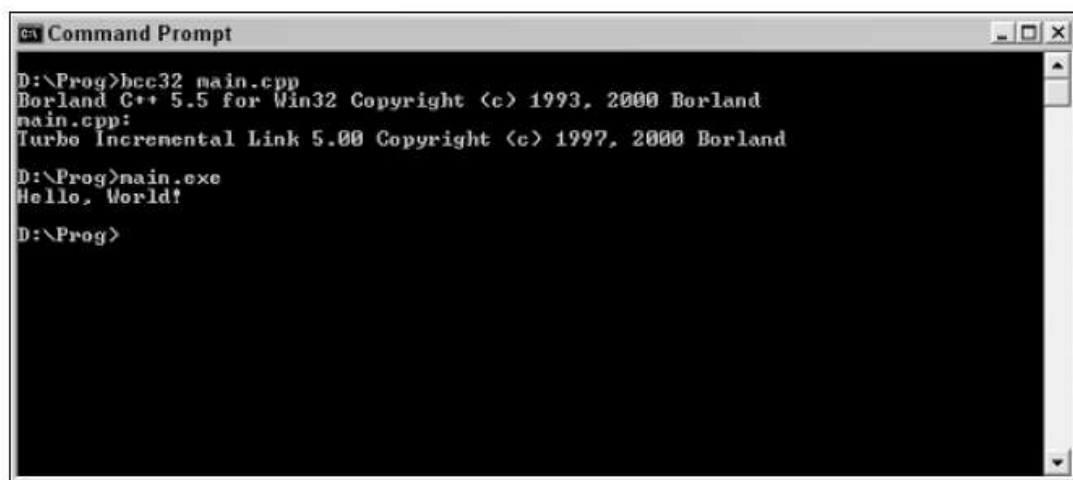
فى التالى دعنا نلقى نظرة على << . وتسمى معامل الإدخال insertion Operator . ومهمته هى إرسال شئ ما (بيانات) من جزء فى البرنامج إلى آخر . فى هذه الحالة , يستخدم لتمرير القيم الحرفية التى تتبعها إلى الأمر cout لإظهارها فى الشاشة .

الجزء التالى هو القيم الحرفية التى سيتم عرضها , هذا بسيط , فالقيم الحرفية "Hello, World!" . كل شئ داخل علامات التنصيص "" يتم عرضه فى الشاشة .

نأتى بعد ذلك إلى معامل إدخال آخر , يعنى هذا ان لا شئ آخر سيتم تمريره إلى أمر cout . وهذا بعض الشئ هو endl . وتستخدم لإدراج سطر جديد .

مؤخراً , لدينا إغلاق الجملة وهى (;) وتسمى semicolon . وخلافاً لـ javascript , هذا مطلوب فى C++ .

إذا قمت بتشغيل هذا الكود فى compiler وتم تفسيره به , ستحصل على ما تتوقعه . يقوم البرنامج بالعمل ويقوم بتشغيل الدالة التى تسمى main . وينفذ الجملة ويعرض النص على الشاشة (كما فى الشكل)



```
Command Prompt
D:\Prog>bcc32 main.cpp
Borland C++ 5.5 for Win32 Copyright (c) 1993, 2000 Borland
main.cpp:
Turbo Incremental Link 5.00 Copyright (c) 1997, 2000 Borland
D:\Prog>main.exe
Hello, World!
D:\Prog>
```

كل شئ سيعمل كما تتوقع ان يكون .

مجرد مناجاة سريعة ... تذكر انى قلت سابقاً انه يمكنك إضافة Header files التى ليس لها حاجة لمصدر الكود ولا تؤثر فى الكود النهائى . حسناً , عندما قمت بتفسير الكود السابق كان الكود النهائى 112,640 bytes , ولكن إذا أضفت سطر للكود ليشمل رأس string.h (وهو ما لا يستخدمه الكود) سيظل الملف فى نفس الحجم بالضبط .

الآن ما قمنا بإعداده ماهو إلا دالة بسيطة .

```
// C++ code template
#include <iostream.h>
void main()
{
    cout << "Hello, World!" << endl;
}
```

ولكن دالة تقوم بفعل اكثر من شئ واحد , . وهاك دالة أخرى لديها جملتين بسيطتين :

```
// C++ code template
#include <iostream.h>
void main()
{
    cout << "Hello, World!" << endl;
    cout << "How are you?" << endl;
}
```

More Functions

المزيد من الدوال

دالة واحد للبداية , ولكن لأداء بعض العمليات البرمجية تحتاج أكثر من دالة .

إلقى نظرة على هذا الكود :

```
#include <iostream.h>

void hello()
{
    cout << "Hello, World!" << endl;
}

void hello2()
{
    cout << "Hello, World, MK II!" << endl;
}

void main()
{
    cout << "main function" << endl;
}
```

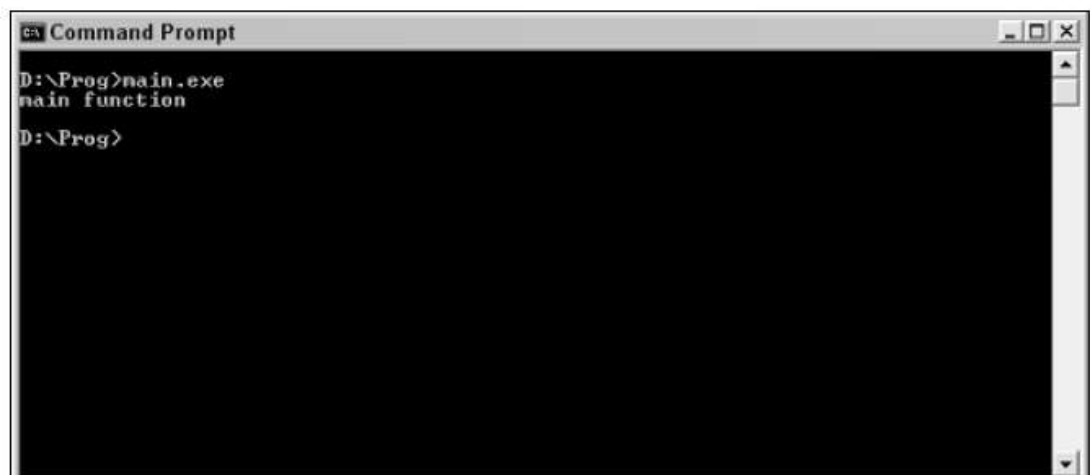
هذا الكود لديه ثلاثة دوال :

hello

hello2

main

ومع ذلك , إذا قمت بتفسير هذا الكود وقمت بتشغيله ستجد شئ ما شيق , كما يظهر بالشكل ,



بمجرد ان تعمل الدالة main , التى تعمل كأفتراضياً , ولجعل الآخرين يقوموا بالعمل , ضعهم كلهم فى الكود هكذا :

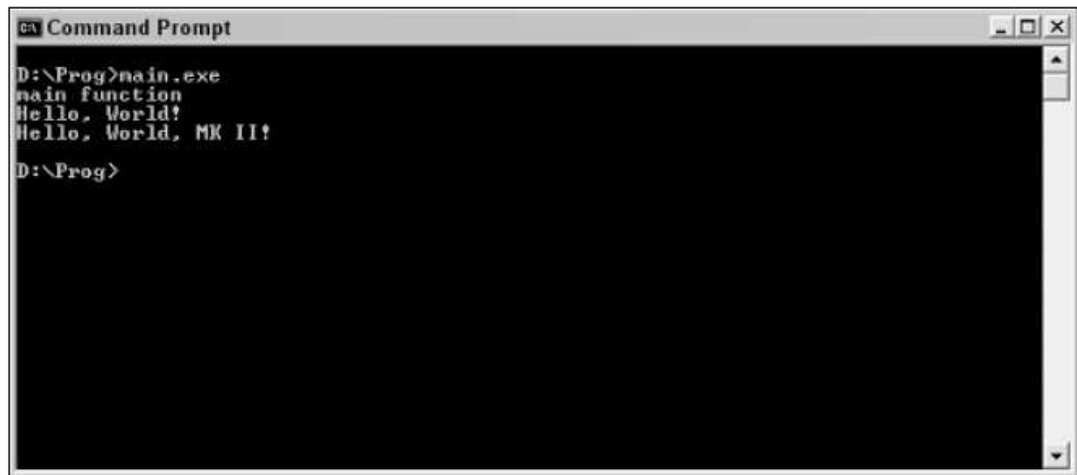
```
#include <iostream.h>

void hello()
{
    cout << "Hello, World!" << endl;
}

void hello2()
{
    cout << "Hello, World, MK II!" << endl;
}

void main()
{
    cout << "main function" << endl;
    hello();
    hello2();
}
```

إذا قمت بتفسير هذا وقمت بتشغيل الكود , ستحصل على المخرجات هكذا كما بالشكل :



```
CA Command Prompt
D:\Prog>main.exe
main function
Hello, World!
Hello, World, MK II!
D:\Prog>
```

ما يفعله هذين السطرين هو إعطاء الأمر بتشغيل الدوال الأخرى بعد تشغيل الدالة main . منطقياً , ربما تعتقد ان هذا الكود سيعمل

```
#include <iostream.h>

void hello()
{
    cout << "Hello, World!" << endl;
    hello2();
}

void hello2()
{
    cout << "Hello, World, MK II!" << endl;
}

void main()
{
    cout << "main function" << endl;
    hello();
}
```

ومع ذلك , هناك مشكلة مع هذا — تستدعى دالة لم تعرف بعد (او تجهز بواسطة المفسر) . هذا ينتج عنه عطل ذلك لأن المفسر يحب أن يرى الدالة قبل إستدعائها , لذلك بينما ينتج الكود السابق للأعطال , الدالة التالية لن ينتج عنها اعطال لان الدالة Hello2 تنادى قبل أن تظهر فى الكود .

```
#include <iostream.h>

void hello2()
{
    cout << "Hello, World, MK II!" << endl;
}

void hello()
{
    cout << "Hello, World!" << endl;
    hello2();
}

void main()
{
    cout << "main function" << endl;
    hello();
}
```

يمكنك أيضاً أن تستدعى الدوال اكثر من مرة

```
#include <iostream.h>

void hello2()
{
    cout << "Hello, World, MK II!" << endl;
}

void hello()
{
    cout << "Hello, World!" << endl;
    hello2();
}

void main()
{
    cout << "main function" << endl;
    hello();
    hello2();
}
```

نتائج هذا الكود تظهر فى الشكل التالى :



```
Command Prompt
D:\Prog>main.exe
main function
Hello, World!
Hello, World, MK II!
Hello, World, MK II!
D:\Prog>
```

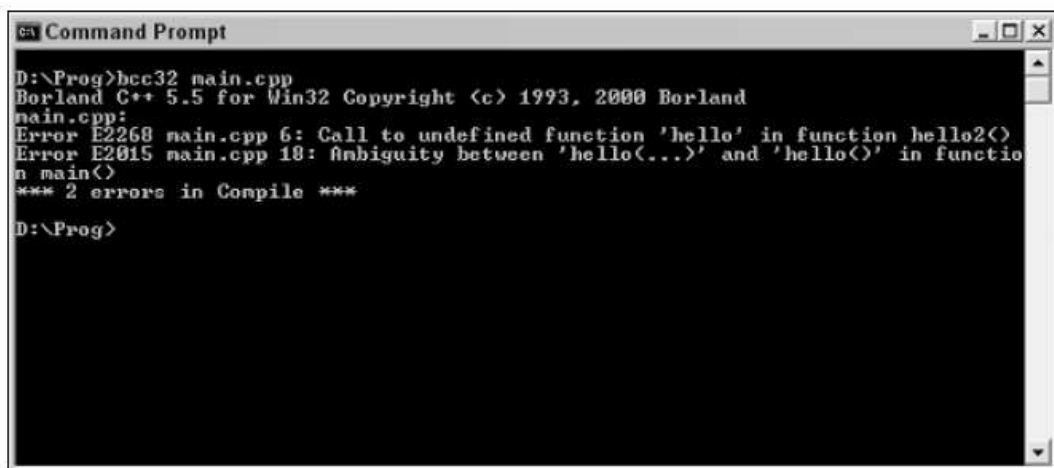
لا تحتاج ان تكون حذراً عندما تستدعي نفس الدالة أكثر من مرة ,لأن المفسر ربما يشكو من الغموض في الكود .
الكود التالي ينتج عنه عطل كما يظهر .

```
#include <iostream.h>

void hello2()
{
    cout << "Hello, World, MK II!" << endl;
    hello();
}

void hello()
{
    cout << "Hello, World!" << endl;
    hello2();
}

void main()
{
    cout << "main function" << endl;
    hello();
}
```



```

C:\ Command Prompt
D:\Prog>bcc32 main.cpp
Borland C++ 5.5 for Win32 Copyright (c) 1993, 2000 Borland
main.cpp:
Error E2268 main.cpp 6: Call to undefined function 'hello' in function hello2()
Error E2015 main.cpp 18: Ambiguity between 'hello<...>' and 'hello<>' in function main()
*** 2 errors in Compile ***
D:\Prog>
```

ستقوم باستخدام الكثير من الدوال خلال هذا الدرس في هذا الفصل , لذا سنترك المناقشة في الدوال لهذا الآن . فقط تذكر على الرغم من أن , الدوال أقسام فرعية من الكود , وكما أن الـ Statements اوامر تعادل الجمل , فإن الدوال تعادل الفقرات , وللتحرك بين الدوال نستخدم الإستدعاء Calling. وتذكر أن تظهر الدالة قبل ان تستدعيها وستكون بأمان من الخطأ .

Quick Exercise

تمارين سريع

إلقى نظرة على امثلة الكود التالي وحدد (بدون تفسيرهم وتشغيلهم إذا امكن)ما هو الخطا في كلا منها .

Code Sample 1:

```
#include <iostream.h>

void hello()
{
    cout << "main function" << endl;
}
```

Code Sample 2:

```
#include <iostream.h>

void main()
{
    cout << "main function" << endl;
    hello();
    hello2();
}

void hello()
{
    cout << "Hello, World!" << endl;
    hello2();
}

void hello2()
{
    cout << "Hello, World, MK II!" << endl;
}
```

Code Sample 3:

```
void main()
{
    cout << "main function" << endl;
}
```

Conditionals

الأوامر الشرطية

إذا كان هناك شيء يجعلك تقدر قوة البرمجة , فهي الأوامر الشرطية . يمكنك الأوامر الشرطية من تشغيل الكود اعتماداً على الخلفية عن بعض ما يسمى **parameter** لذا يمكنك أن تختار أي الجمل ليتم تنفيذه وإيهما ليتم تجاهله , تدور الأوامر الشرطية حول الإجابة عن الأسئلة , وهذه الأسئلة حصرياً **true/false** أو **yes/No** . الإجابات لأسئلة بسيطة .

ليست فقط البرمجة هي التي تعتمد على الأوامر الشرطية فحسب — بل في الحياة يوماً بعد يوم فنحن أنفسنا نستخدم الأوامر الشرطية . ها هو مثال بسيط

```
For breakfast, would you rather have toast or cereal?
```

هذا أمر شرطى , والإجابة تقرر ما ستفعله بعد . يمكن أن تقول نعم لـ **toast** , نعم لـ **Cereal** , أو نعم لكلاهما , أو لا لكلاهما . في الحقيقة , هذه هي الإجابات الممكنة فقط . فإذا اخترت أن لا تجيب على السؤال 'إذاً في نظرية كلا الحالتين تقول لكلاهما لا , فإذا أضفت وظيفة أخرى فعليك تغيير الـ **parameters** للأسئلة.

فكما علمت الآن , الأوامر الشرطية تدور حول صنع القرار , والشئ الجيد فى التعامل مع الحاسبات والبرمجة أنك إن طلبت منه أن يأتى لك بقرار يعتمد على بعض البيانات , فإن كان عنده ما طلبت , سيعطيك .

Programming Decisions

قرارت البرمجة

كل لغات البرمجة لديها طريقة لتختبر بها الأشياء ثم تصنع القرارات اعتماداً على هذا الاختبار . فى C++ الطريقة لفعل ذلك هو استخدام جملة if .

هاك مثال مبسط فى C++ لجملة if .

```
if (condition)
{
    // do statements here if the condition is true
}
else
{
    // do these statements if the condition is false
};
```

جملة if تختبر الشرط المعطى (مزيد من المعلومات حول هذه الشروط قريباً) وإما أن يكون true أو false . فإذا كان الشرط true . إذاً ستنفذ جملة واحدة او مجموعة من الجمل, بينما إن كان ناتج الشرط false , فسيتم تشغيلها جملة اخرى او مجموعة من الجمل.

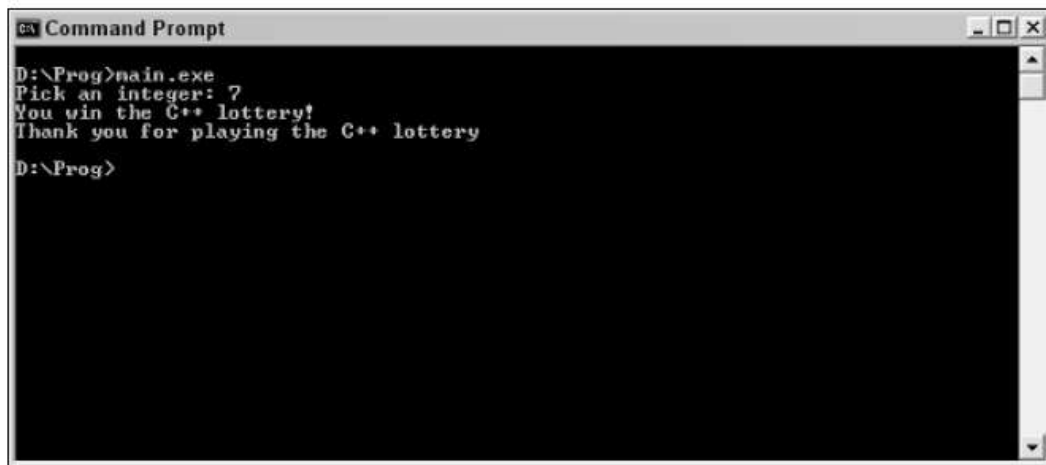
Conditions

الشروط

إلقى نظرة على الكود التالى

```
#include <iostream.h>
void main()
{
    int x;
    cout << "Pick an integer: ";
    cin >> x;
    if (x == 7)
    {
        cout << "You win the C++ lottery!" << endl;
        cout << "Thank you for playing the C++ lottery" << endl;
    }
    else
    {
        cout << "Thank you for playing the C++ lottery" << endl;
    }
}
```

إذا انشأت هذا الكود , إحفظه , ثم قم بتفسيره , وقم بتشغيل التطبيق . ستحصل على البرنامج الذى يبدوا على هذا الشكل :



```
Command Prompt
D:\Prog>main.exe
Pick an integer: 7
You win the C++ lottery!
Thank you for playing the C++ lottery
D:\Prog>
```

و يستخدم هذا الكود الأوامر الشرطية , فدعنا نبجر خلال الكود ونلقى نظرة على كيفية العمل .

```
#include <iostream.h>
void main()
{
    int x;
    cout << "Pick an integer: ";
    cin >> x;
    if (x == 7)
    {
        cout << "You win the C++ lottery!" << endl;
        cout << "Thank you for playing the C++ lottery" << endl;
    }
    else
    {
        cout << "Thank you for playing the C++ lottery" << endl;
    }
}
```

الأسطر الأولى القليلة مجرد كود C++ بدائي . ينبغي عليك ان تعرف ماذا يعنى كل هذا الكود من الان.

```
#include <iostream.h>
void main()
{
    int x;
    cout << "Pick an integer: ";
    cin >> x;
    if (x == 7)
    {
        cout << "You win the C++ lottery!" << endl;
        cout << "Thank you for playing the C++ lottery" << endl;
    }
    else
    {
        cout << "Thank you for playing the C++ lottery" << endl;
    }
}
```

التالى هى إفتتاح القوس المتعرج للدالة , دالة main

```
#include <iostream.h>
void main()
{
    int x;
    cout << "Pick an integer: ";
    cin >> x;
    if (x == 7)
    {
        cout << "You win the C++ lottery!" << endl;
        cout << "Thank you for playing the C++ lottery" << endl;
    }
    else
    {
        cout << "Thank you for playing the C++ lottery" << endl;
    }
}
```

الآن يعلن هذا السطر عن المتغير الذي سنستخدمه . و يسمى المتغير X .

```
#include <iostream.h>
void main()
{
    int x;
    cout << "Pick an integer: ";
    cin >> x;
    if (x == 7)
    {
        cout << "You win the C++ lottery!" << endl;

        cout << "Thank you for playing the C++ lottery" << endl;
    }
    else
    {
        cout << "Thank you for playing the C++ lottery" << endl;
    }
}
```

يرسل هذا الأمر سطر نصي إلى الشاشة . ولاحظ أني لم أضف endl إلى نهاية السطر .

```
#include <iostream.h>
void main()
{
    int x;
    cout << "Pick an integer: ";
    cin >> x;
    if (x == 7)
    {
        cout << "You win the C++ lottery!" << endl;
        cout << "Thank you for playing the C++ lottery" << endl;
    }
    else
    {
        cout << "Thank you for playing the C++ lottery" << endl;
    }
}
```

الان وقد أتينا إلى شئ جديد , فى الجمل السابقة رأينا cout . اما فى هذه الجملة رأينا cin . بينما cout هو امر يرسل القيم الحرفية للشاشة . فإن cin يستخدم لإستقبال ادخالات من لوحة المفاتيح . هذه المدخلات يتم إدراجها فى المتغير x بإستخدام معامل الإدراج او الأذخال >> .

وما ان تقوم بتشغيل الكود , ستلاحظ أن هذا الرقم الذى ادخلته قد وضع فى نفس السطر كالنص المخرج ليسألك عن إدخال رقم صحيح . وهذا هو احد الآثار الجانبية لعدم استخدام `endl` .

```
#include <iostream.h>
void main()
{
    int x;
    cout << "Pick an integer: ";
    cin >> x;
    if (x == 7)
    {
        cout << "You win the C++ lottery!" << endl;
        cout << "Thank you for playing the C++ lottery" << endl;
    }
    else
    {
        cout << "Thank you for playing the C++ lottery" << endl;
    }
}
```

الآن إلى طعم الأوامر الشرطية — وهو الاختبار . هذا السطر من الكود يختبر إذا كانت القيمة للمتغير `x` هو 7 . وهذا اختبار بسيط , ولكن يمكنك هذا لاختبار الشرط وتقوم بتشغيل الجمل اعتماداً عليه .

```
#include <iostream.h>
void main()
{
    int x;
    cout << "Pick an integer: ";
    cin >> x;
    if (x == 7)
    {
        cout << "You win the C++ lottery!" << endl;
        cout << "Thank you for playing the C++ lottery" << endl;
    }
    else
    {
        cout << "Thank you for playing the C++ lottery" << endl;
    }
}
```

هذه الجمل تعمل إذا تحقق الشرط وقد اتى بـ `true` . فالقيمة المدخلة بالفعل هي 7 .

```
#include <iostream.h>
void main()
{
    int x;
    cout << "Pick an integer: ";
    cin >> x;
    if (x == 7)
    {
        cout << "You win the C++ lottery!" << endl;
        cout << "Thank you for playing the C++ lottery" << endl;
    }
    else
    {
        cout << "Thank you for playing the C++ lottery" << endl;
    }
}
```

وهاهى `else` . تعرض الجمل التى سيتم تشغيلها إذا لم يتحقق الشرط أى كان ناتج الشرط بـ `False` .


```
#include <iostream.h>
void main()
{
    int x;
    cout << "Pick an integer: ";
    cin >> x;
    if (x == 7)
    {
        cout << "You win the C++ lottery!" << endl;
        cout << "Thank you for playing the C++ lottery" << endl;
    }
    else
    {
        cout << "Thank you for playing the C++ lottery" << endl;
    }
}
```

هذا هو . وعلى الرغم من اختلاف الطرق لفعل نفس الشيء في C++ . فالكود الذي أنشأته في المثال التالي يفعل نفس الشيء كما المثال السابق , ولكن لا يقوم بعمل else .

```
#include <iostream.h>
void main()
{
    int x;
    cout << "Pick an integer: ";
    cin >> x;
    if (x == 7)
    {
        cout << "You win the C++ lottery!" << endl;
        cout << "Thank you for playing the C++ lottery" << endl;
    }
}
```

الاختلاف الرئيسي في هذا الكود من الكود السابق هو هذه الأسطر :

```
#include <iostream.h>
void main()
{
    int x;
    cout << "Pick an integer: ";
    cin >> x;
    if (x == 7)
    {
        cout << "You win the C++ lottery!" << endl;
        cout << "Thank you for playing the C++ lottery" << endl;
    }
}
```

ما حدث مع هذا الكود هو أنه : إذا كان الشرط صحيح إذا يتم تنفيذ الجملتين , ولكن إذا كان الشرط غير متحقق فقط فالسطر الأخير فقط هو من سينفذ .

ولكن, كما ترى , هذا النوع من الكود يمكن ان يكون غامض , فماذا يحدث في حالة مثال هذه ؟

```
#include <iostream.h>
void main()
{
    int x;
    cout << "Pick an integer: ";
    cin >> x;
    if (x == 7)
    {
        cout << "You win the C++ lottery!" << endl;
        cout << "What about this statement..." << endl;
        cout << "Thank you for playing the C++ lottery" << endl;
    }
}
```

هاهى هذه الجمل تم تشغيلها ؟

الإجابة تكون إذا تحقق الشرط , إذاً كل الجمل ستتفد , ولكن إن لم يتحقق سينفذ الكل ما عدا الأولى . فيمكن أن يكون هذا الاختصار مفيد مما يعنى انك يمكنك ان تكتب كود أقل , ولكن أيضاً يكون اكثر غموضاً من كتابة الكل ويكون أكثر مللاً .

More on Conditionals

الأوامر الشرطية

إلقى نظرة على هذا الكود الذى ينشأ حاسبة مبسطة فى C++

```
#include <iostream.h>
void main()
{
    float num1;
    float num2;
    char op;
    float ans;
    cout << "Please enter a number: ";
    cin >> num1;
    cout << "Please enter another number: ";
    cin >> num2;
    cout << "Press A to add the two numbers."
        << endl
        << "Press S to subtract the two numbers."
        << endl
        << "Press M to multiply the two numbers."
        << endl
        << "Press D to divide the two numbers."
        << endl;
    cin >> op;
    if (op == 65)
        ans = num1 + num2;
    if (op == 83)
        ans = num1 - num2;
    if (op == 77)
        ans = num1 * num2;
    if (op == 68)
        ans = num1 / num2;
    cout << "The answer is " << ans << endl;
}
```

هذا الكود يمكنك من اخذ رقمين إثنين يمكنك إدخالهم واداء عمليات عليهم إعتماًداً على المدخلات . يستخدم كثيراً ما قمنا بالبحث فيه فى هذا الفصل . دعنا نلقى نظرة على بعض تسليط الضوء على الكود .

```
#include <iostream.h>
void main()
{
    float num1;
    float num2;
    char op;
    float ans;
    cout << "Please enter a number: ";
    cin >> num1;
    cout << "Please enter another number: ";
    cin >> num2;
    cout << "Press A to add the two numbers."
        << endl
        << "Press S to subtract the two numbers."
        << endl
```

```

        << "Press M to multiply the two numbers."
        << endl
        << "Press D to divide the two numbers."
        << endl;
    cin >> op;
    if (op == 65)
        ans = num1 + num2;
    if (op == 83)
        ans = num1 - num2;
    if (op == 77)
        ans = num1 * num2;
    if (op == 68)
        ans = num1 / num2;
    cout << "The answer is " << ans << endl;
}

```

هذه الأسطر الأربعة تعرف المتغيرات التي سنستخدمها في الكود . نجد نوعين هنا :

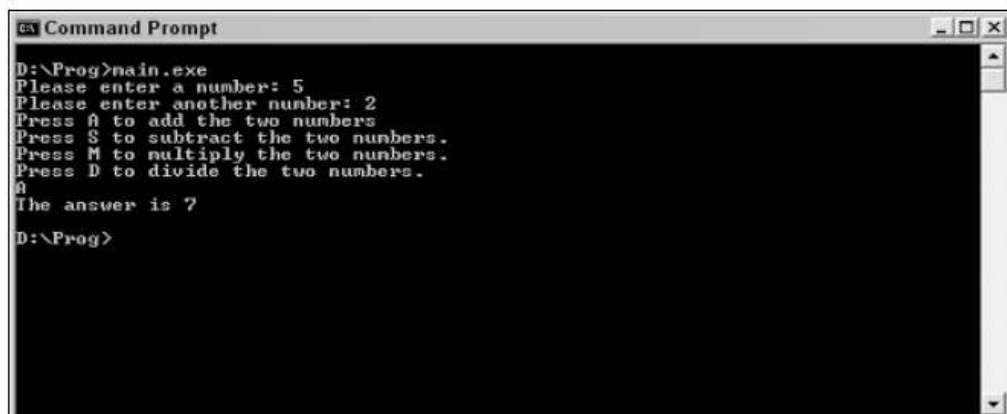
- Float : رقم النقطة العائمة (رقم يمكن ان يستخدم العلامة العشرية للضبط, مثال 3.141592654)
- Char : إختصار Character . كما سنرى قريباً , وتم تخزينهم في طريقة عجيبة في الكود . إليك القليل من تركيب cin , cout يمكنك ان تقوم بإدخال الرقمين

```

#include <iostream.h>
void main()
{
    float num1;
    float num2;
    char op;
    float ans;
    cout << "Please enter a number: ";
    cin >> num1;
    cout << "Please enter another number: ";
    cin >> num2;
    cout << "Press A to add the two numbers."
        << endl
        << "Press S to subtract the two numbers."
        << endl
        << "Press M to multiply the two numbers."
        << endl
        << "Press D to divide the two numbers."
        << endl;
    cin >> op;
    if (op == 65)
        ans = num1 + num2;
    if (op == 83)
        ans = num1 - num2;
    if (op == 77)
        ans = num1 * num2;
    if (op == 68)
        ans = num1 / num2;
    cout << "The answer is " << ans << endl;
}

```

هذه الأسطر تعطينا مخرجات منسقة جيداً كما ترى بالشكل



```

D:\Prog>main.exe
Please enter a number: 5
Please enter another number: 2
Press A to add the two numbers
Press S to subtract the two numbers.
Press M to multiply the two numbers.
Press D to divide the two numbers.
A
The answer is 7
D:\Prog>

```

```

#include <iostream.h>
void main()
{
    float num1;
    float num2;
    char op;
    float ans;
    cout << "Please enter a number: ";
    cin >> num1;
    cout << "Please enter another number: ";
    cin >> num2;
    cout << "Press A to add the two numbers."
        << endl
        << "Press S to subtract the two numbers."
        << endl
        << "Press M to multiply the two numbers."
        << endl
        << "Press D to divide the two numbers."
        << endl;
    cin >> op;
    if (op == 65)
        ans = num1 + num2;
    if (op == 83)
        ans = num1 - num2;
    if (op == 77)
        ans = num1 * num2;
    if (op == 68)
        ans = num1 / num2;
    cout << "The answer is " << ans << endl;
}

```

هذا الإدخال الثالث للوحة المفاتيح إلى البرنامج , هذه المرة سيتعرف على العملية المنفذة على الرقمين التي أدخلتها سابقاً .

```

#include <iostream.h>
void main()
{
    float num1;
    float num2;
    char op;
    float ans;
    cout << "Please enter a number: ";
    cin >> num1;
    cout << "Please enter another number: ";
    cin >> num2;
    cout << "Press A to add the two numbers."
        << endl
        << "Press S to subtract the two numbers."
        << endl
        << "Press M to multiply the two numbers."
        << endl
        << "Press D to divide the two numbers."
        << endl;
    cin >> op;
    if (op == 65)
        ans = num1 + num2;
    if (op == 83)
        ans = num1 - num2;

```

```

    if (op == 77)
        ans = num1 * num2;
    if (op == 68)
        ans = num1 / num2;
    cout << "The answer is " << ans << endl;
}

```

هذه هي القوة الدافعة الرئيسية للكود , ينظر الكود إلى المعامل الذى تم إدخاله وينفذ على الأرقام اعتماداً على ذلك .
 شئ آخر لتلاحظه فى كيفية الإدخال من لوحة المفاتيح يكون حرف (A,S,M, أو D) , ولكن فى هذا الكود قمنا
 بفحص الأرقام 65,83,77 أو 68 . وهى كذلك لأنها عندما يتم إدخال الحرف كـ char . فتمثيل النظام العشري
 فى كود ASCII قد تم تخزينه . لا عليك أن تقلق كثيراً عن هذا , ولكن الحروف تمثل كما يلى :

A	B	C	D	E	F
65	66	67	68	69	70

G	H	I	J	K	L
71	72	73	74	75	76

M	N	O	P	Q	R
77	78	79	80	81	82

S	T	U	V	W	X
83	84	85	86	87	88

Y	Z	Space			
89	90	32			

A	b	c	d	e	f
97	98	99	100	101	102

G	h	i	j	k	l
103	104	105	106	107	108

M	n	o	p	q	r
109	110	111	112	113	114

S	t	u	v	w	x
115	116	117	118	119	120

Y	z				
121	122				

```

#include <iostream.h>
void main()
{
    float num1;
    float num2;
    char op;
    float ans;
    cout << "Please enter a number: ";
    cin >> num1;
    cout << "Please enter another number: ";
    cin >> num2;
    cout << "Press A to add the two numbers."
    << endl
    << "Press S to subtract the two numbers."
    << endl
    << "Press M to multiply the two numbers."
    << endl
    << "Press D to divide the two numbers."
    << endl;
    cin >> op;
    if (op == 65)
        ans = num1 + num2;
    if (op == 83)
        ans = num1 - num2;
    if (op == 77)
        ans = num1 * num2;
    if (op == 68)
        ans = num1 / num2;
    cout << "The answer is " << ans << endl;
}

```

الجملة الأخيرة هي التي ستعرض النتيجة على الشاشة . الشكل التالي يظهر التطبيق في الفعل .

```

D:\Prog>main.exe
Please enter a number: 34
Please enter another number: 12
Press A to add the two numbers
Press S to subtract the two numbers.
Press M to multiply the two numbers.
Press D to divide the two numbers.
A
The answer is 46
D:\Prog>

```

هذا البرنامج يعمل جيداً ولكن هناك مشكلة — فحص الناتج وبدلاً أن تقوم بالضغط على أحد المفاتيح التي تصف الحروف (A,S,M أو D) , إضغظ شئ آخر غيرهم (إنظر الشكل)

```

D:\Prog>main.exe
Please enter a number: 12
Please enter another number: 21
Press A to add the two numbers
Press S to subtract the two numbers.
Press M to multiply the two numbers.
Press D to divide the two numbers.
x
The answer is 6.08504e-39
D:\Prog>

```

الإجابة بالتأكيد ليست صحيحة , والكود لا يمكنه الإمساك بالمدخلات التي قمت بإدخالها , و نحتاج إلى فعل شيء ما حيال هذا الأمر .

```
#include <iostream.h>
void main()
{
    float num1;
    float num2;
    char op;
    float ans;
    cout << "Please enter a number: ";
    cin >> num1;
    cout << "Please enter another number: ";
    cin >> num2;
    cout << "Press A to add the two numbers."
        << endl
        << "Press S to subtract the two numbers."
        << endl
        << "Press M to multiply the two numbers."
        << endl
        << "Press D to divide the two numbers."
        << endl;
    cin >> op;

    if (op == 65)
    {
        ans = num1 + num2;
        cout << "The answer is " << ans << endl;
    }
    if (op == 83)
    {
        ans = num1 - num2;
        cout << "The answer is " << ans << endl;
    }
    if (op == 77)
    {
        ans = num1 * num2;
        cout << "The answer is " << ans << endl;
    }
    if (op == 68)
    {
        ans = num1 / num2;
        cout << "The answer is " << ans << endl;
    }
    if (op != 65 && op != 83 && op != 77 && op != 68)
    {
        cout << "No valid operation was chosen!" << endl;
    }
}
```

ليس فقط علينا أن نضيف تسليط الضوء على الكود , ولكن علينا ان نضيف الأقواس المتعرجة لجعل الكود أقل غموضاً.
لكن ماذا تفعل الجملة المضافة ؟ حسناً , هذه تتحكم في القيم الحرفية لهذا الشرط :

```
if (op != 65 && op != 83 && op != 77 && op != 68)
```

هناك معاملين إثنين هنا لم ترهم من قبل :

!=

&&

المعامل != يأتي من لا تساوى بينما && تمثل and و . وبالتالي , فى الإنجليزية , ما يقول :

إذا كان op لا يساوى 65 و op لا تساوى 83 و op لا تساوى 77 و op لا تساوى 68 , إذاً

واليك طريقة أخرى لتمثل لا تساوى :

```
if (!(op == 83))
```

الشرط هنا :

```
If op is not equal to 83 then . . .
```

وهذا مساوياً لكتابة :

```
if (op != 83)
```

لذا , فهو يختبر كل المدخلات الصحيحة ولو لم يكن هناك أحدهم موجود , إذاً يقوم بتشغيل الجملة . فهو موجز وذكى جداً .

ليكون الشرط صحيح , كل الشروط التى يتحتويها لابد ان تكون لها متحقة أى true ولو كان هناك واحد منهم غير متحقق أى false , إذاً فالنتيجة تغلب عليهم وتكون غير متحقق أى false .

جنباً إلى جنب && , معاملاً آخر ستجده مفيداً فى الأمر الشرطية هو المعامل الشرطى || . ويمثل هذا أو ويمكن استخدامه فقط كهذه && .

واليك مثال :

```
if (op != 65 || op != 83 || op != 77 || op != 68)
```

يقال هذا فى الإنجليزية :

إذا كان op لا يساوى 65 أو op لا تساوى 83 أو op لا تساوى 77 أو op لا تساوى 68 , إذاً

هذا مختلف جداً من استخدام && مع الاختلاف الرئيسى كون الشرط عام وشامل ليكون غير متحقق , كل الشرطيات تحتاج أن تكون false , على الجانب الآخر , ستعود بالقيمة true .

Quick Exercise

إلقى نظرة على امثلة الكود التالى وحدد (بدون تفسيرهم وتشغيلهم إذا امكن) أى الجمل ينفذ .

Code Sample 1

```
#include <iostream.h>
void main()
{
    int x;
    x = 7;
    if (x == 7)
    {
        cout << "Statement 1" << endl;
    }
    else
    {
        cout << "Statement 2" << endl;
    }
}
```


Code Sample 2

```
#include <iostream.h>
void main()
{
    int x;
    x = 7;
    if (!(x == 7))
    {
        cout << "Statement 1" << endl;
    }
    else
    {
        cout << "Statement 2" << endl;
    }
}
```

Code Sample 3

```
#include <iostream.h>
void main()
{
    char x;
    x = 65;
    if (x == 64 || x != 65 || x == 66)
    {
        cout << "Statement 1" << endl;
    }
    else
    {
        cout << "Statement 2" << endl;
    }
}
```

Code Sample 4

```
#include <iostream.h>
void main()
{
    char x;
    x = 65;
    if (x == 64 && x == 65 && x == 66)
    {
        cout << "Statement 1" << endl;
    }
    else
    {
        cout << "Statement 2" << endl;
    }
}
```

Code Sample 5

```
#include <iostream.h>
void main()
{
    char x;
    x = 65;
    if (!(x == 64 && x == 65 && x == 66))
    {
        cout << "Statement 1" << endl;
    }
    else
    {
        cout << "Statement 2" << endl;
    }
}
```

Loops

اوامر حلقات التكرار

مهمة اخرى هامة حقاً تحتاج ان تكون قادراً على أدائها مع الكود وهى أن تنتظر خلال نفس القطعة من الكود عدد من المرات. التطبيقات المفيدة بأوامر التكرار ربما لا تظهر لك حالياً , ولكن الان دعنا ننظر كيف يتم التكرار .

For Loops

حلقات التكرار بـ for

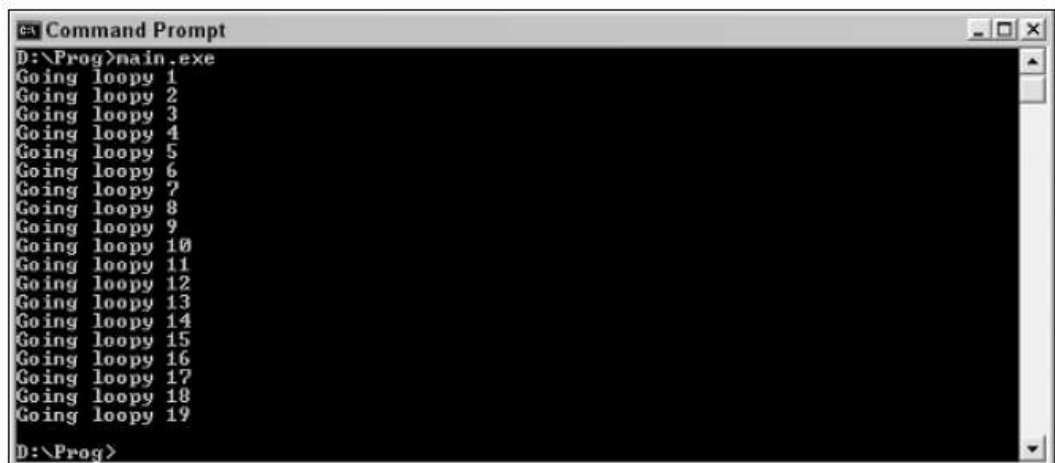
نوع واحد من حلقات التكرار التى ستمر بها هو حلقات التكرار for .

```
for ( <initialisation> ; <terminating condition> ; <increment> )  
{  
    // statement (or block of statements)  
}
```

إليك بعض الكود الذى يعمل على حلقات التكرار

```
#include <iostream.h>  
void main()  
{  
    int counter;  
  
    for (counter = 1; counter < 20; counter++)  
    {  
        cout << "Going loopy " << counter << endl;  
    }  
}
```

إذا قمت بتفسير هذا الكود وقمت بتشغيله , ستحصل على المخرجات كما يظهر فى الشكل :



```
Command Prompt  
D:\Prog>main.exe  
Going loopy 1  
Going loopy 2  
Going loopy 3  
Going loopy 4  
Going loopy 5  
Going loopy 6  
Going loopy 7  
Going loopy 8  
Going loopy 9  
Going loopy 10  
Going loopy 11  
Going loopy 12  
Going loopy 13  
Going loopy 14  
Going loopy 15  
Going loopy 16  
Going loopy 17  
Going loopy 18  
Going loopy 19  
D:\Prog>
```

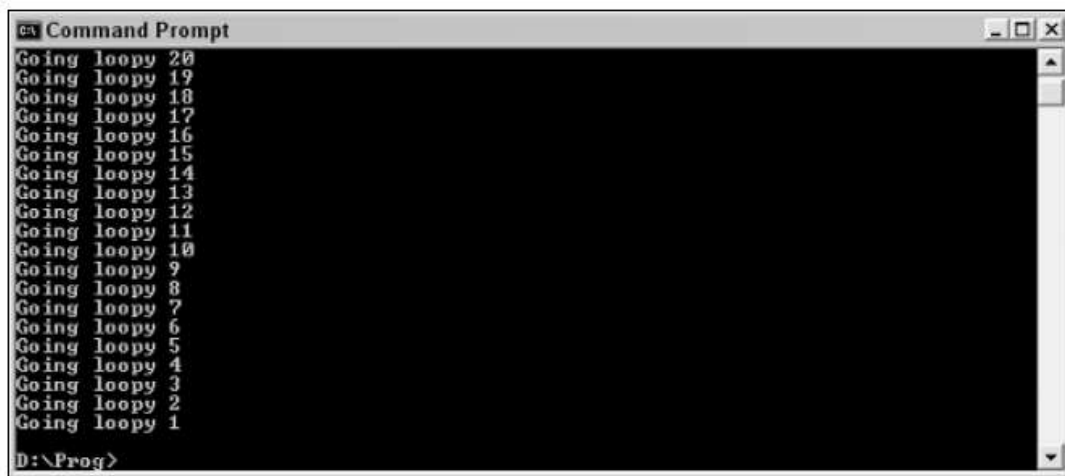
حلقات التكرار, تسمى حلقات تكرار for , ويتم التحكم فيها من خلال هذا السطر من الكود :

```
for (counter = 1; counter < 20; counter++)
```

يعتمد هذا على ثلاثة اجزاء :

- **counter = 1** وهذه هي النقطة الابتدائية للتكرار , أو حيثما نريد ان نبدأ . هذه هي القيمة التي أعطيت لمتغير التحكم . في هذه الحالة القيمة لـ **counter** تحمل الرقم 1 .
- **counter < 20** هذا هو شرط إنهاء التكرار . فعندما ينتج عن الشرط القيمة **false** , يتوقف التكرار . في هذه الحالة , عندما يكون **counter** أقل من 20 , سيتكرر حلقة التكرار .
- **counter++** هذه هي الزيادة , هنا يعني أن قيمة **counter** ستزيد بمقدار واحد خلال كل حلقة دوران . **counter--** ستعني ان القيمة ينبغي ان تنقص بمقدار 1 لكل حلقة دوران .

الكود الذى يتبع العمل إلى الخلف , من 20 إلى 0 . لاحظ الشكل كيف انه لا يتجه إلى الأسفل إلى الصفر وذلك بسبب أن الشرط قد تم إعداده ليعمل طالما أن الـ **counter** أكبر من صفر .



```
cs Command Prompt
Going loopy 20
Going loopy 19
Going loopy 18
Going loopy 17
Going loopy 16
Going loopy 15
Going loopy 14
Going loopy 13
Going loopy 12
Going loopy 11
Going loopy 10
Going loopy 9
Going loopy 8
Going loopy 7
Going loopy 6
Going loopy 5
Going loopy 4
Going loopy 3
Going loopy 2
Going loopy 1
D:\Prog>
```

```
#include <iostream.h>
void main()
{
    int counter;

    for (counter = 20; counter > 0; counter--)
    {
        cout << "Going loopy " << counter << endl;
    }
}
```

تعديل واحد سيمكنك من ان تتم التكرار إلى أن تصل إلى الصفر .

```
#include <iostream.h>
void main()
{
    int counter;

    for (counter = 20; counter >= 0; counter--)
    {
        cout << "Going loopy " << counter << endl;
    }
}
```

Infinite Loops

التكرار إلى ما لا نهاية

الشيء الرئيسي للملاحظة في حلقات التكرار هو ما نسميه حلقات التكرار إلى ما لا نهاية . حيث ترتكب خطأ في الكود فسينتج أن شرط الإنهاء لن يتم تحققه , مما سينتج عنه أن حلقات التكرار ستستمر للأبد (أو إلى أن توقفه أو يتعطل الحاسب).

إليك مثال على مثل هذا النوع من حلقات التكرار :

```
#include <iostream.h>
void main()
{
    int counter;

    for (counter = 20; counter <= 20; counter--)
    {
        cout << "Going loopy " << counter << endl;
    }
}
```

هذا الكود سيتكرر للأبد بسبب أن شرط الإنهاء لا يمكن أن **false** بسبب أن النقصان لقيمة **counter** تأخذه بعيداً من كونها أكبر من 0 , ليست أقرب .

While Loops

حلقات التكرار بـ while

هي نوع آخر من حلقات التكرار التي ربما تجده مفيداً في عملك .

```
while ( <condition> )
{
    // statement (or block of statements)
}
```

يختلف هذا عن **for** لأنه سيستمر حتى يتم مقابلة الشرط بإدخالات أو بعض الوسائل الأخرى , بدلاً من الزيادة أو النقصان للقيمة التي تتم داخل الكود .

إليك مثال:

```
#include <iostream.h>
void main()
{
    int num;
    cout << "Enter a number greater than 7: ";
    cin >> num;
    while (num <= 7)
    { cout << "Try again: ";
      cin >> num;
    }
}
```

و يظهر هذا الكود في الفعل في الشكل :

```
C:\WINDOWS\System32\cmd.exe
D:\Prog>main.exe
Enter a number greater than 7: 1
Try again: 6
Try again: 4
Try again: 2
Try again: 7
Try again: 8
D:\Prog>
```

مبدئياً , هذا الكود يعمل على تكرار هذه الجملة حتى يحصل على المدخلات المطلوبة :

```
#include <iostream.h>
void main()
{
    int num;
    cout << "Enter a number greater than 7: ";
    cin >> num;
    while (num <= 7)
    { cout << "Try again: ";
      cin >> num;
    }
}
```

Do While loop

حلقة تكرار أخرى فى c++ هى حلقات التكرار DO while

```
do
{
    // statement (or block of statements)
}
while ( <condition> )
```

تشبه كثيراً حلقة تكرار While , وإليك مثال سيجعله أكثر وضوحاً

```
#include <iostream.h>
void main()
{
    int num;
    do
    {
        cout << "Enter a number greater than 7: ";
        cin >> num;
    }
    while (num <= 7);
}
```

فى حالة انك ادخلت قيمة تفى بهذا (أى رقم صحيح أكبر من 7) يتم إنهاؤها بشكل جيد . (كما يظهر بالشكل)

```
C:\WINDOWS\System32\cmd.exe
D:\Prog>main.exe
Enter a number greater than 7: 1
Try again: 4
Try again: 5
Try again: 6
Try again: 7
Try again: 9
D:\Prog>
```

Quick Exercise

تمرين سريع

إلقى نظرة على امثلة الكود التالى وحدد (بدون تفسيرهم وتشغيلهم إذا امكن) الإجابة للأسئلة .

Code Sample 1

كم عدد المرات سيتكرر هذا الكود ؟

```
#include <iostream.h>
void main()
{
    int counter;

    for (counter = 5; counter < 18; counter++)
    {
        cout << "Going loopy " << counter << endl;
    }
}
```

Code Sample 2

كم عدد المرات سيتكرر الكود ؟

```
#include <iostream.h>
void main()
{
    int counter;

    for (counter = 10; counter > 0; counter--)
    {
        cout << "Going loopy " << counter << endl;
    }
}
```

Code Sample 3

هل هذا حلقة تكرار لا نهائية ؟

```
#include <iostream.h>
void main()
{
    int counter;

    for (counter = 10; counter <= 5; counter--)
    {
        cout << "Going loopy " << counter << endl;
    }
}
```

Code Sample 4

هل هذا حلقة تكرار لا نهائية ؟

```
#include <iostream.h>
void main()
{
    int counter;

    for (counter = 10; counter <= 15; counter--)
    {
        cout << "Going loopy " << counter << endl;
    }
}
```

Code Sample 5

كم الأرقام التى ستفى بحلقة التكرار while ؟

```
#include <iostream.h>
void main()
{
    int num;
    cin >> num;
    while (num <= 14)
    { cout << "Try again: ";
      cin >> num;
    }
}
```

Code Sample 6

كم الأرقام التى ستفى بحلقة التكرار while ؟

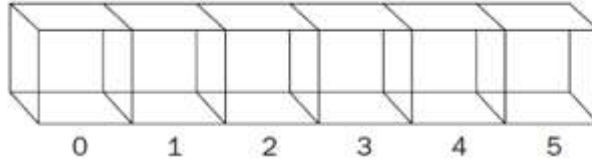
```
#include <iostream.h>
void main()
{
    int num;
    cin >> num;
    while (num < 12)
    { cout << "Try again: ";
      cin >> num;
    }
}
```

Arrays

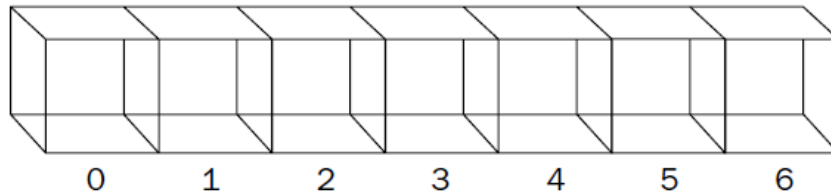
المصفوفات

إلى هذا الحد , المتغيرات المستخدمة في هذا الكود قد إستخدمت لتحمل قيمة واحدة . ومع ذلك , فهناك حالات يكون أكثر إفادة أن تجمع أكثر من قيمة معاً تحت إسماً واحداً . ولهذا كانت المصفوفات .

فيمكن ان تفكر في المصفوفات على أنها تسلسل من علب البريد , كلاً منها له رقم يعرف به . (إنظر الشكل)



هذه العلب البريدية معرفة الهوية برقم , تبدأ الأرقام بـ 0 مروراً بزيادة 1 على من تلاه . فإذا كنت منشأ لمصفوفة تحمل سبعة درجات في مادة , سترقم من 0 إلى 6 (كما في الشكل)



تحتاج المصفوفات إن يتم إعلانها بنفس طريقة إعلان المتغيرات . فيمكنك ان تسمى المصفوفة grades , والعنصر الفردي في هذه المصفوفة يمكن ان يشار إليه بـ grades[0] , grades[1] , عل طول الطريق إلى grades[6] . هنا هو كود سياخذ سبع درجات ومن ثم يعرض لهم جنباً إلى جنب مع المتوسط .

```
#include <iostream.h>
void main()
{
    float grades[7];
    int i;
    float tot;
    cout << "Please enter grade 1: ";
    cin >> grades[0];
    cout << "Please enter grade 2: ";
    cin >> grades[1];
    cout << "Please enter grade 3: ";
    cin >> grades[2];
    cout << "Please enter grade 4: ";
    cin >> grades[3];
    cout << "Please enter grade 5: ";
    cin >> grades[4];
    cout << "Please enter grade 6: ";
    cin >> grades[5];
    cout << "Please enter grade 7: ";
    cin >> grades[6];
    cout << "Your grades are: " << endl;
    for (i = 0; i < 7; i++)
    {
        cout << grades[i] << endl;
        tot += grades[i];
    }
    cout << "Average: " << (tot/7) << endl;
}
```

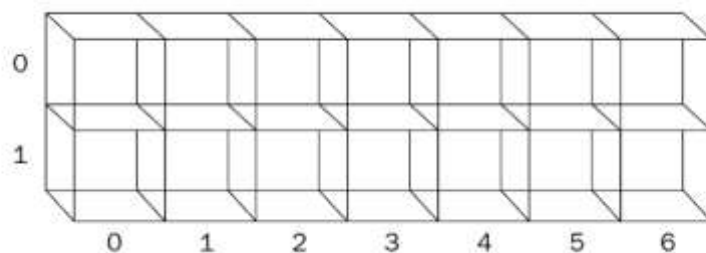

التطبيق المنفذ يظهر في الشكل

```
C:\WINDOWS\System32\cmd.exe
D:\Prog>grades.exe
Please enter grade 0: 99
Please enter grade 1: 75
Please enter grade 2: 93
Please enter grade 3: 91
Please enter grade 4: 48
Please enter grade 5: 90
Please enter grade 6: 88
Your grades are:
99
75
93
91
48
90
88
D:\Prog>
```

Two-Dimensional Array

مصفوفة ثنائية الأبعاد

المصفوفة التي رأينا حتى الآن تسمى مفردة الأبعاد , هذا النوع من المصفوفات يحتوى على سطر واحد لأطار البيانات. المصفوفة ثنائية الأبعاد تمثل بواسطة شبكة مستطيلية الإطار . (إنظر الشكل)



لنعلن عن مصفوفة كمثال التي في الشكل السابق , فسيكون عليك أن تستخدم هذا السطر من الكود :

```
int array_values[5][7];
```

هذا النوع من المصفوفات سيكون مفيداً حين تريد ان تقول.." تخزن النتائج من سبع درجات لخمس مواد مختلفة "

فيمكن أن يتم إعداد القيم باستخدام الكود هكذا :

```
array_values[2][6] = 56;
```

ويتم الوصول إلي القيم هكذا :

```
x = 2;
y = 6;
cout << array_values[x][y];
```

ويلعب التكرار مع المصفوفات دوراً هاماً, إلقى نظرة على هذا الكود , وانظر كيف يمكن أن نستخدم التكرار لمعالجة المصفوفة :

```
for (x = 0; x <= 6; x++)
    for (y = 0; y <= 4; y++)
    {
        cout << "Enter a value for the element " << x << " : " << y << " : ";
        cin >> array_values[y][x];
    }
```

Multidimensional Array

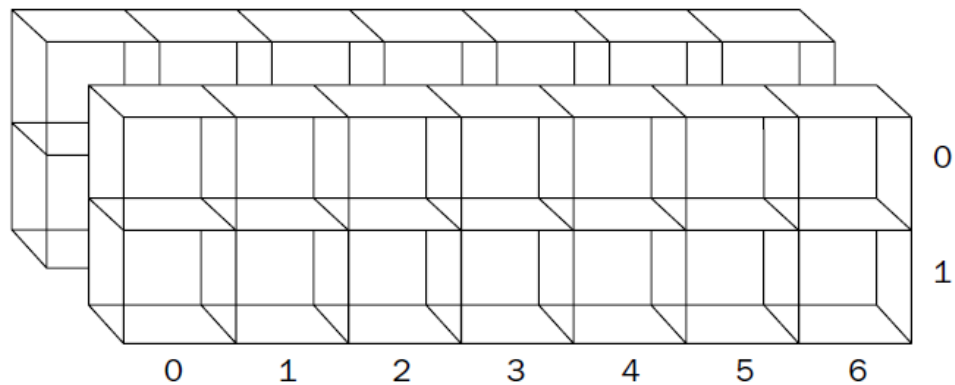
مصفوفة متعددة الأبعاد

هي مصفوفة لثلاثة أو أكثر من الأبعاد . الشكل التالي هو تمثيل لمصفوفة ثلاثية الأبعاد .

يمكنك ان تعلن عنهم بالكود مثل هذا :

```
int letters[5][5][7];
```

على الجانب الآخر, العمل مع هذا النوع من المصفوفات هو نفس طريقة العمل مع المصفوفات احادية وثنائية الأبعاد التي رأيناها .



Quick Exercise

تمرين سريع

إلقى نظرة على امثلة الكود التالي وحدد (بدون تفسيرهم وتشغيلهم إذا امكن) الإجابة للأسئلة .

Code Sample 1

ما الخطأ في هذا الكود ؟

```

#include <iostream.h>
void main()
{
    float grades[6];
    int i;
    float tot;
    cout << "Please enter grade 1: ";
    cin >> grades[1];
    cout << "Please enter grade 2: ";
    cin >> grades[2];
    cout << "Please enter grade 3: ";
    cin >> grades[3];
    cout << "Please enter grade 4: ";
    cin >> grades[4];
    cout << "Please enter grade 5: ";
    cin >> grades[5];
    cout << "Please enter grade 6: ";
    cin >> grades[6];
    cout << "Please enter grade 7: ";
    cin >> grades[7];
    cout << "Your grades are: " << endl;
    for (i = 0; i < 7; i++)
    {
        cout << grades[i] << endl;
        tot += grades[i];
    }
    cout << "Average: " << (tot/7) << endl;
}

```

Code Sample 2

صف هذه المصفوفة ؟

```
int letters[5][4][7];
```

Code Sample 3

ماذا تفعل هذا الكود ؟

```
array_values[3][2][6] = 22;
```

Summary

الملخص

فى هذا الفصل , كان لدينا المزيد من الأيدى فى التجربة مع كلا من C++ والتنوع فى تركيبات الكود المختلفة . ولقد أنجزنا المزيد فى هذا الفصل , وينبغى ان تعطى نفسك الفرصة للإعادة لإتمامه بنجاح .

فى هذا الفصل ألقينا نظرة على :

- ❑ Functions (الدوال)
- ❑ Conditionals (الأوامر الشرطية)
- ❑ Loops (حلقات التكرار)
- ❑ Arrays (المصفوفات)

إذا عملت خلال كل الأمثلة , وما أقترحه عليك لتفعله الان هو أخذ نظرة سريعة على الأمثلة فى هذا الفصل والبحث فى طرق التعديل فيهم . وإنظر إن كان يمكنك ان تأخذ الكود وتجعله يقوم بالمزيد . على سبيل المثال , خذ الكود من الأمثلة السابقة وقم بتقطيعه إلى دوال منفصلة (مثال الحاسبة مثالى لهذا الأمر)

أيضاً امل ان هذا الفصل قد فتح شهيتك لجعلك مهياً للكود والبرمجة .

8

Problem Solving

حل المشكلات

البرمجة تقريباً عن حل المشكلات , الكثير من الناس يبدو أنهم يفكرون فى أن البرمجة عن تعلم كيف تكتب كود . ومن ثم إختبار الكود قبل تفسيره وتحويله إلى تطبيقات مفيدة , وجعل هذه التطبيقات متاحة للآخرين . كل هذا صحيح , وهذه هى الخطوات الصحيحة التى يجب أن تأخذها كمبرمج , ولكن تفتقد للنقطة الحقيقية للبرمجة — والتي تعنى بها البرمجة وهى حل المشكلات . ولغة البرمجة هى ببساطة الآداة التى تستخدمها لاداء هذه الوظيفة .

البرمجة تشبه كتابة رواية , إذا أردت ان تكون الملك "ستفين" التالى أو "توم كلانسي" أو "جون جريشام" أو "جي. كي. رولنج" , فلا تحتاج إن تذهب إلى أى مكان إذا كانت كل إهتماماتك فى التدقيق الإملائى والقواعد وإستخدام معالج نصوص ! فهى إستخدام الادوات التى لديك لحل المشكلة .

على سبيل المثال , فإذا كنت تكتب كود يضيف ضريبة المبيعات لأجمالى أمر البيع . فالكود الذى تحتاج أن تكتبه لحل المشكلة هو كيفية حساب ضريبة إجمالى المبيعات وكيف يتم إضافة هذه الضريبة إلى الإجمالى . فإذا قمت بإنشاء تطبيق أكثر تعقيداً , كمثال لعبة , إذاً فهذا يحل المزيد من المشاكل القليلة . بعضها مفصل هنا :

- جعل الأشياء تتحرك فى الشاشة .
- جعل اللعبة تتفاعل مع اللاعب .
- الإستجابة لمدخلات اللاعب .
- تتبع أهداف اللعبة .
- تسجيل النقاط .
- إنشاء مستويات أصعب / أسهل للعبة .
- كيف تقوم بحفظ اللعبة والسماح بأن يعاد تحميلها مرة اخرى .

لذا , كما ترى , يحتاج المبرمج أن يكون موجد حلول جيد للمشاكل , وفى هذا الفصل سنلقى نظرة على حل المشكلات وكما هو جيد حل المشكلات البرمجية بتطبيق المنطق عليهم .

The Basics of Problem Solving

مبادئ حل المشكلات

لإن حل المشكلات هي مهارة جوهرية يحتاج المبرمج أن يتقنها , سنقضى الآن بعض الوقت بالنظر إلى الأنماط التي تجعل حل المشكلات أسرع , أسهل , وأكثر دقة .

لنبحث في حل المشكلات , تحتاج مثال لتعمل من خلاله , خذ في إعتبارك التالي . لقد إقتربت من تطبيق آلة حاسبة , يريد العميل منك ان تنشأ تطبيق لديه القدرة على تحويل درجات الحرارة بين وحدتين قياس شائعتين :

❑ Fahrenheit to centigrade (or Celsius) تحويل الفهرنهايت إلى الدرجة المئوية

❑ Centigrade (or Celsius) to Fahrenheit تحويل الدرجة المئوية (او السيليلوز) إلى الفهرنهايت

لا يبدو للعميل الإهتمام لكيفية فعل هذا , هو فقط يريد التطبيق حيث يمكنه إدخال درجة حرارة واحدة ويقوم بتحويلها إلى أخرى .

كيف يمكنك ان تنهج الكتابة مثل هذا التطبيق ؟ سأفترض انك لا تعرف الكثير عن عملية التحويل ولكن يمكنك كتابة كود مبسط في C++ .

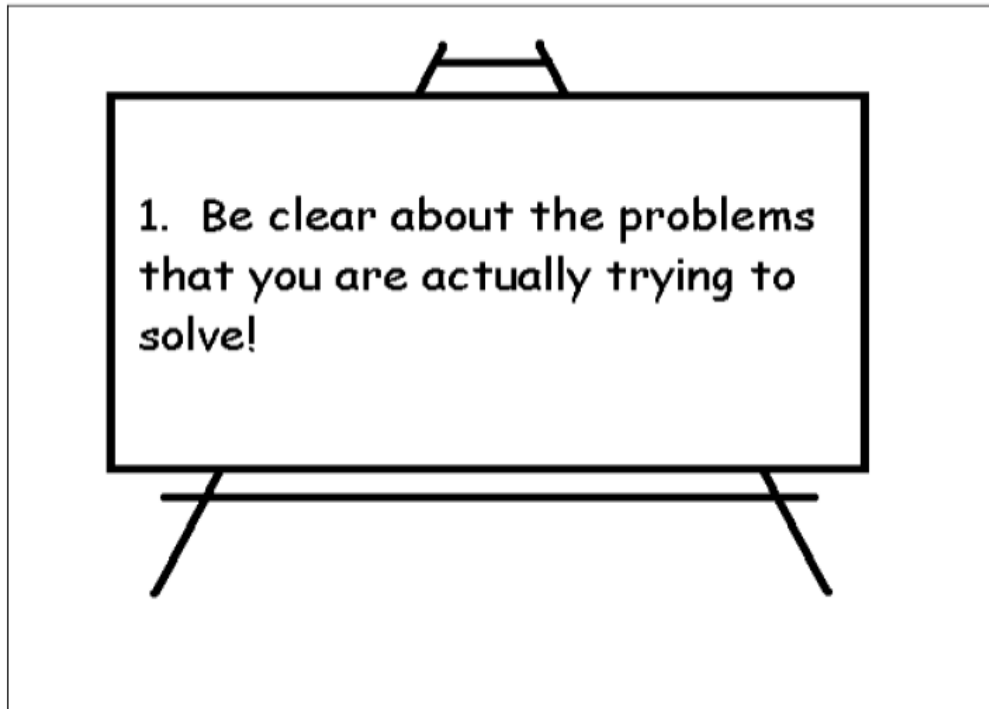
دعنا الآن نلقى نظرة على المشاكل التي تحتاج إلى حلها لتكون قادراً على كتابة هذا التطبيق وكيف تحلها.

إذا كنت تعرف كيف تحل هذا النوع من التحويل , حاول أن تنساه لفترة إن امكن , العمل خلال هذه العملية وإكتشاف كيف نأتي للطريقة المنطقية لحل هذا النوع من المشكلة .

Be Clear about the Requirements

كن واضحاً مع المتطلبات

الخطوة الأولى لحل أى مشكلة هو التأكد من انك تعرف بالضبط ما هي المشكلة التي تحاول أن تجد لها حل . كما بالشكل :



لذا , ما هى المشكلة التى تحاول ان تجد لها حلاً هنا ؟ حسناً , ربما تكون مندهشاً لتعرف أن هناك القليل !
قبل أن تعالج أى مشكلة برمجية , الشئ الأول الذى تحتاج أن تفعله هو التأكد من معرفة ماذا يريد العميل (أو أنت , إذا حدث وأردت أن تكتب تطبيق لنفسك) . ويسمى هذا متطلبات العميل .



و ينطوى هذا على إستجواب العميل عن ماذا يريد من البرنامج أن يفعله بالفعل . فيمكن ان يكون بعضاً الناس لديه غموض فى مرحلة التخطيط ويصبحوا أكثر إختياراً وتحديداً بمجرد أن نضع المزيد من العمل فى المشروع , الحصول على ما يريده العميل من عمل البرنامج او التطبيق . يمكن أن يتغير فيما بعد حسب الحاجة ولكن إذا كنت أنت لست مفيداً بوقت فستكون قادراً على ان تتولى مسؤولية المزيد من العمل .

هذه المرحلة كلها عن توجيه أسئلة والحصول على الإجابات , أسئلة مثل :

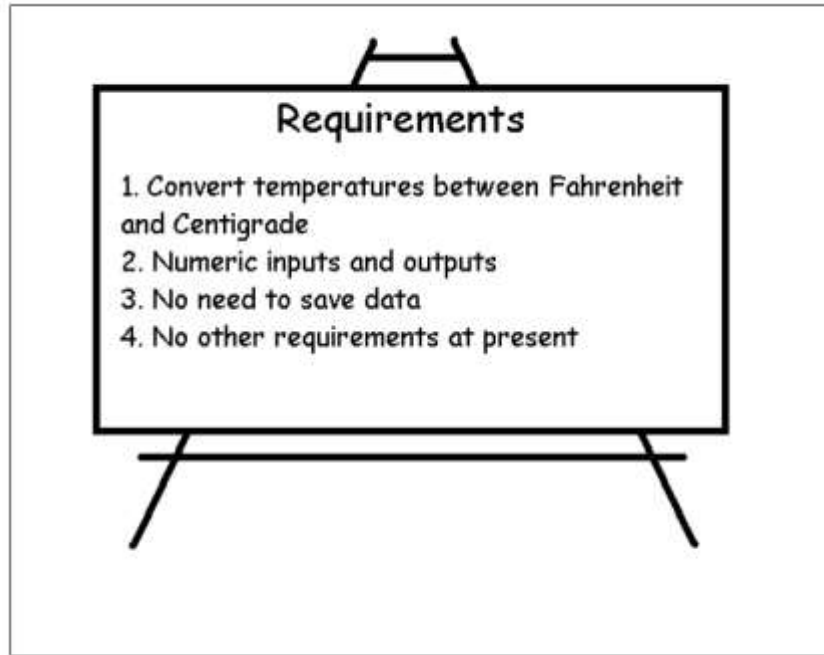
- ماذا تريد من التطبيق أن يفعل ؟
- ما هى المدخلات التى تريد ان يأخذها التطبيق ؟
- ما هى المخرجات التى تريدها من التطبيق ؟
- هل هناك شئ آخر تريد ان يفعله التطبيق ؟

إذا قمت بوضع هذه الأسئلة للعميل ينبغى ان تحصل على الإجابة التى تساعدك فى المزيد من التخطيط.

هذه المرحلة كلها عن توجيه أسئلة والحصول على الإجابات , أسئلة مثل :

- ماذا سيفعل التطبيق ؟ "أريد التطبيق ان يقوم بتحويل درجات الحرارة بين الفهرنهايت والدرجات المئوية " .
- إحتياجات المدخلات ؟ "أريد ان أكون قادراً على إدخال درجات الحرارة وإمتلاك برنامج ليفعل ذلك التحويل " .
- المخرجات الممكنة . " مخرجات رقمية إلى الشاشة , ليس هناك حاجة ان يقوم التطبيق بتخزين البيانات فى ملف " .
- إستخدامات أخرى . " ليس فى الحاضر "

يمكنك تنظيم هذه الخلفية في نظام منطقي (إنظر الشكل)

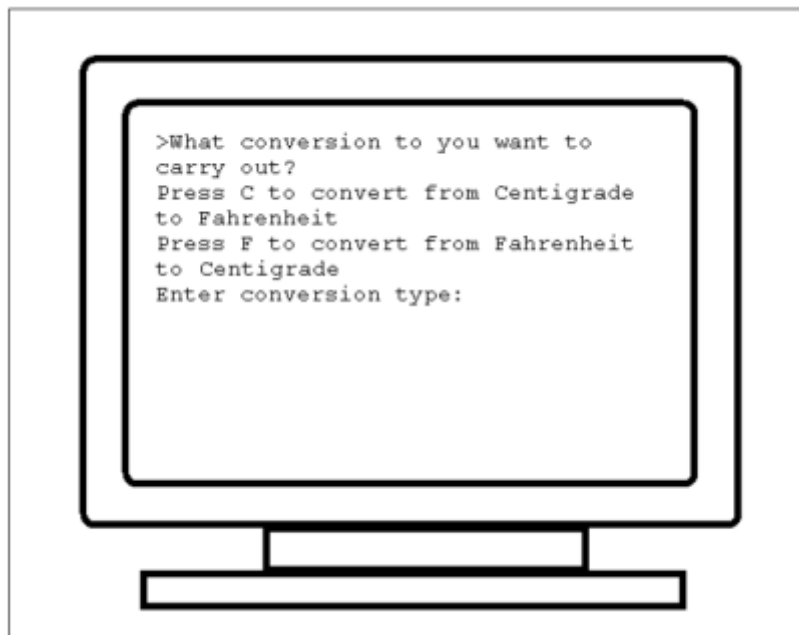


تبدو واضحة بدرجة كافية , ولكن هناك نقطة هامة أريد ان اوضحها لك قبل المعالجة , وهو ما يتم فعله مع المدخلات والمخرجات . إسأل العميل :

هل تريد ان يكون المستخدم قادراً على تحديد نوع المدخلات التي تعطي للبرنامج أو هل تريد أن تكون المدخلات رقم يتم تحويله لكلا من الفهرنهايت والدرجة المئوية تلقائياً ؟

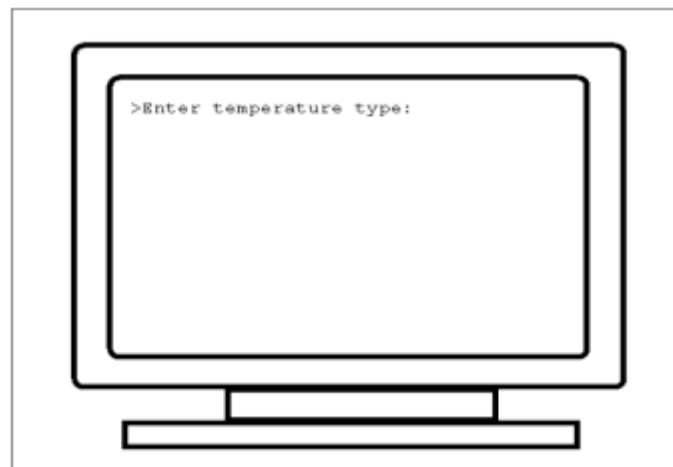
كما ترى , لقد بدأت بالفعل في التفكير في المشكلة كأنها تطبيق في شكله النهائي , وبقدر ما أرى , هناك طريقتين بسيطتين يمكن للتطبيق أن يعمل بهما :

- يمكن أن يسأل التطبيق المستخدم عن نوع التحويل الذي يريد هو او هي تنفيذه (إنظر الشكل) , وبعد , اعتماداً على المدخلات , يتم طبقاً له تنفيذ العملية الحسابية (إنظر الشكل الذي يليه)



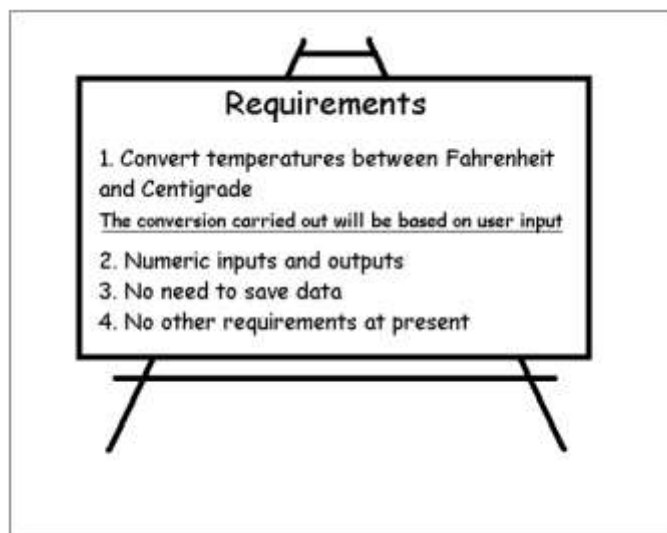


- يمكن ان يسأل التطبيق عن المدخلات (إنظر الشكل) والمخرجات فى وحدات درجات الحرارة(إنظر الشكل الذى يليه) .



هناك طريقتين يمكن للبرنامج أن يعمل بهما , فى حالة أن العميل قرر انه سيكون من الأفضل ان يسأل التطبيق المستخدم عن نوع التحويل الذى سينفذه ويقوم بإخراج القيم بناءً على المدخلات .

هذا يقلل بشكل كبير متطلبات المشروع (إنظر الشكل)



لذا , فلدينا الان المبادئ الرئيسية للبرنامج فى متناول اليد , ويمكن تلخيصهم . وسيكون هذا بمثابة برنامج للمضى قدماً .

Research

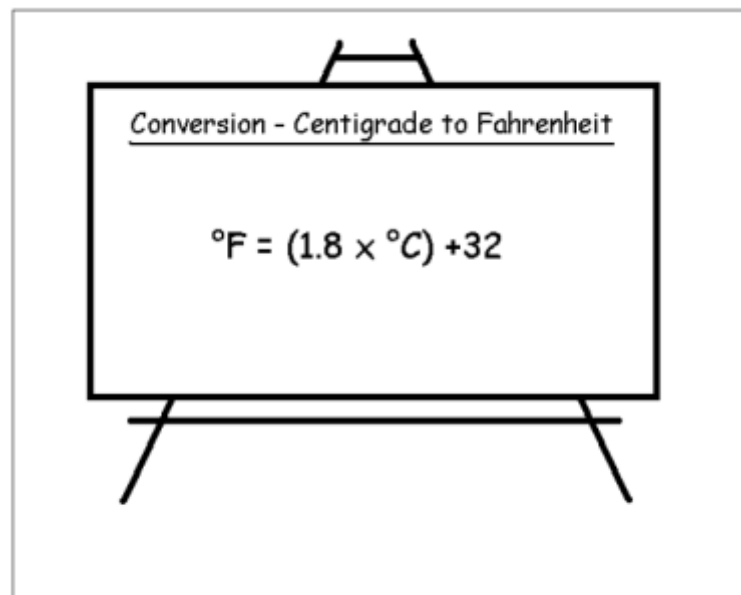
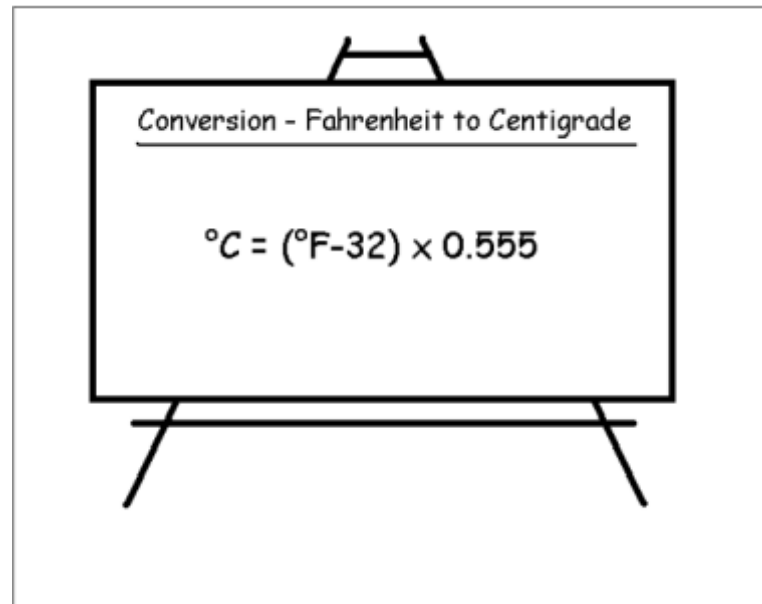
البحث

لقد عرفت كيف تبرمج ولكن هل تعرف كيف تقوم بالتحويل ؟ غالباً ما يتم سؤال المبرمجين عن فعل أشياء ربما لا يعرفون كيف يقومون بعملها . وبالتالي لتكون وظيفة المبرمج جيدة إما فى عمل البحث العلمى او فى توجيه الأسئلة.

هنا قد سالنا للتحويل بين وحدتين لقياس درجات الحرارة . أعرف أن هذا مر عليه وقتاً طويلاً , لقد تم إخبارى كيف أفعل هذا وقد كتبتة فى المدرسة و تم تنفيذ العملية الحسابية بناءً عليه , ولكن لاكون صادقاً أنا لم أعد أذكر كيف قمت بعملها . لذا سأقوم ببعض البحث .

هناك طرق عديدة لأنفذ هذا البحث. يمكن ان أسال العميل ولكن انا مقتنع تماماً أنى يمكن ان أجد كيفية عمل هذا النوع من التحويل على الإنترنت , أو فى كتاب , او بسؤال شخص آخر . ومع ذلك , هناك بعض الأوقات خصوصاً إذا كنت تعمل مع تطبيقات معينة , عندها تود ان يكون لديك الحق فى إمتلاك البحث فى المراجع العلمية التى يمكن أن تستخدمها فى حل المشكلة وتكون فى متناول يدك , على سبيل المثال , تحويل درجات الحرارة أعرف أنى يمكن أن أجده بنفسى , ولكن سيكون الحال مختلف إذا سألت ان تكتب كود يعمل بقوة ليخرج انواع مختلفة من محركات الجازولين — ومن المرجح أن تجد هذا بسهولة على الإنترنت , وسأحتاج عندها الوصول إلى العمليات الحسابية والأشكال المطلوبة .

حسناً , تحويلات درجات الحرارة , مجرد كيف تقوم بتحويل الفهرنهايت إلى الدرجة المئوية والدرجة المئوية إلى فهرنهايت ؟ . قم بتشغل المتصفح , وقم بعمل زيارة سريعة لمحرك البحث , وأدخل بعض معايير البحث القليلة (قريباً يصبح المبرمجون جيّدون فى البحث فى شبكة الويب). سريعاً لقد وجدت السؤال عن تحويل الفهرنهايت إلى الدرجة المئوية (إنظر الشكل) و من الدرجة المئوية إلى فهرنهايت (إنظر الشكل الذى يليه)



يبدو واضحاً لى , السؤال نفسه يبدو سهل التتبع , ومع ذلك , دائماً ما يساعدك أن تحشو بعض الأرقام فى معادلات مثل هذا فقط ليكون مألوفاً لديك لكيفية العمل . (غنظر الشكل الشكلىن التالىين)

Conversion - Fahrenheit to Centigrade

$$^{\circ}\text{C} = (^{\circ}\text{F} - 32) \times 0.555$$

So, if $^{\circ}\text{F} = 25$

$$(25 - 32) \times 0.555 = -3^{\circ}\text{C}$$

التحويل من فهرنهايت إلى الدرجة المئوية

Conversion - Centigrade to Fahrenheit

$$^{\circ}\text{F} = (1.8 \times ^{\circ}\text{C}) + 32$$

So, if $^{\circ}\text{C} = 25$

$$(1.8 \times 25) + 32 = 77^{\circ}\text{F}$$

التحويل من الدرجة المئوية إلى فهرنهايت

هذه الخطوة هامة لإنك بحاجة لتفهم جيداً كيف تقوم بعمل أى معادلة فى العالم الحقيقى , فنتحتاج أن تفعل هذا لأسباب متعددة هامة :

- لإيكنك أن تكتب كود ليحل مشكلة إذا كنت لاتعرف ماذا يفترض من الكود ان يفعل .
- إذا كنت تعرف كيف تحل المشكلة فى العالم الحقيقى سيدعك هذا تحول هذا الشئ الأساسى فى العمل مع الكود فهذا إيكنك أن تحل المشكلة فى عالم الحاسب .
- بواسطة إداركك الجيد لكيفة حل المشكلة إيكن ان تحصل على تقدير للمخرجات التى ستنتج عن الكود والذى سيساعدك لمعرفة متى يعمل الكود ومتى لا يعمل .

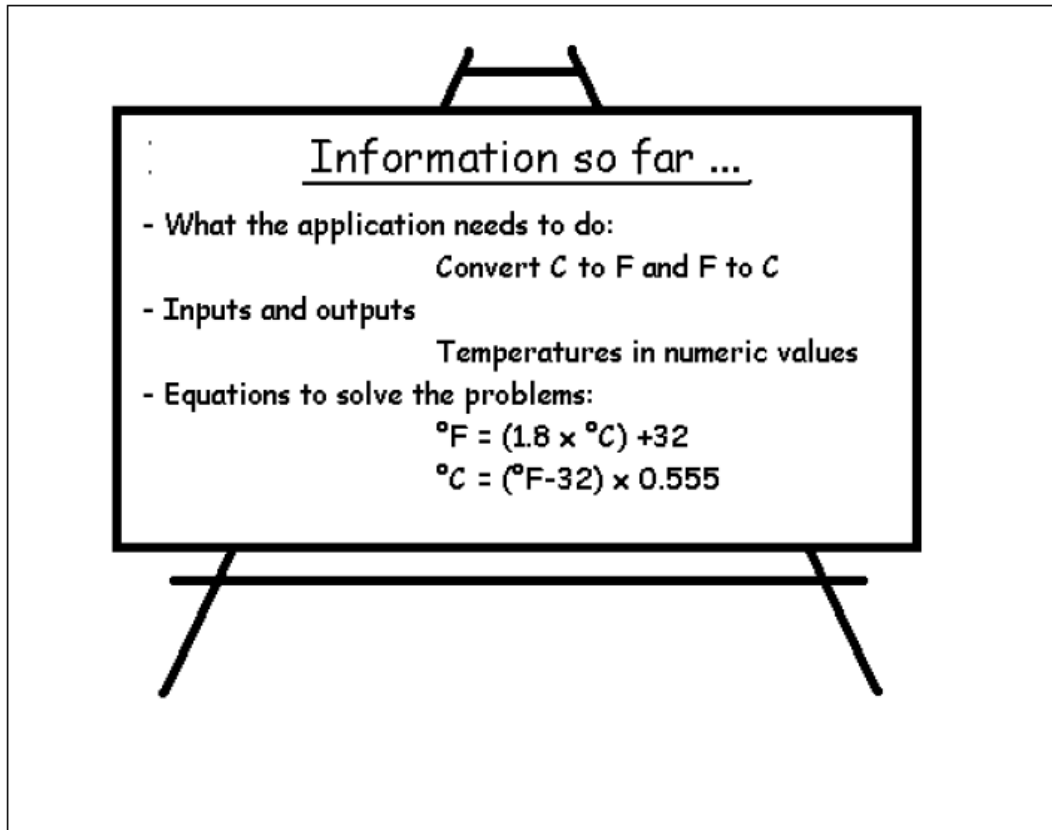
ألقينا النظر بالفعل على أحد المدخلات التى إيكن أن تستخدمها فى الكود , و قمت بتقريب الارقم إلى ارقام صحيحة ولكن هذا فى نفسه سيخدمك جيد لتبدأ عند إختبار الكود مع المدخلات حيث ستكون المخرجات صحيحة .

Breaking the Problem up into Smaller Problems

تقطيع المشكلة إلى مشاكل صغيرة

إلى هذا الحد لقد جمعنا المزيد من المعلومات التى تحتاجها لعمل هذا المشروع (إنظر الشكل) . إنت تعرف :

- ماذا يحتاج البرنامج ان يفعل .
- ماذا ينبغي على المخرجات أن تكون .
- المعادلات التى تحتاجها لتنفيذ هذه الوظائف .



لديك المعلومات التى تحتاجها ولكن الآن ما تحتاجه هو التفكير فى المشكلة (أو القضية) للبرنامج الفعلى الذى ستكتب له الكود لتحلها .

What Are the Issues That the Application Needs to Deal With?

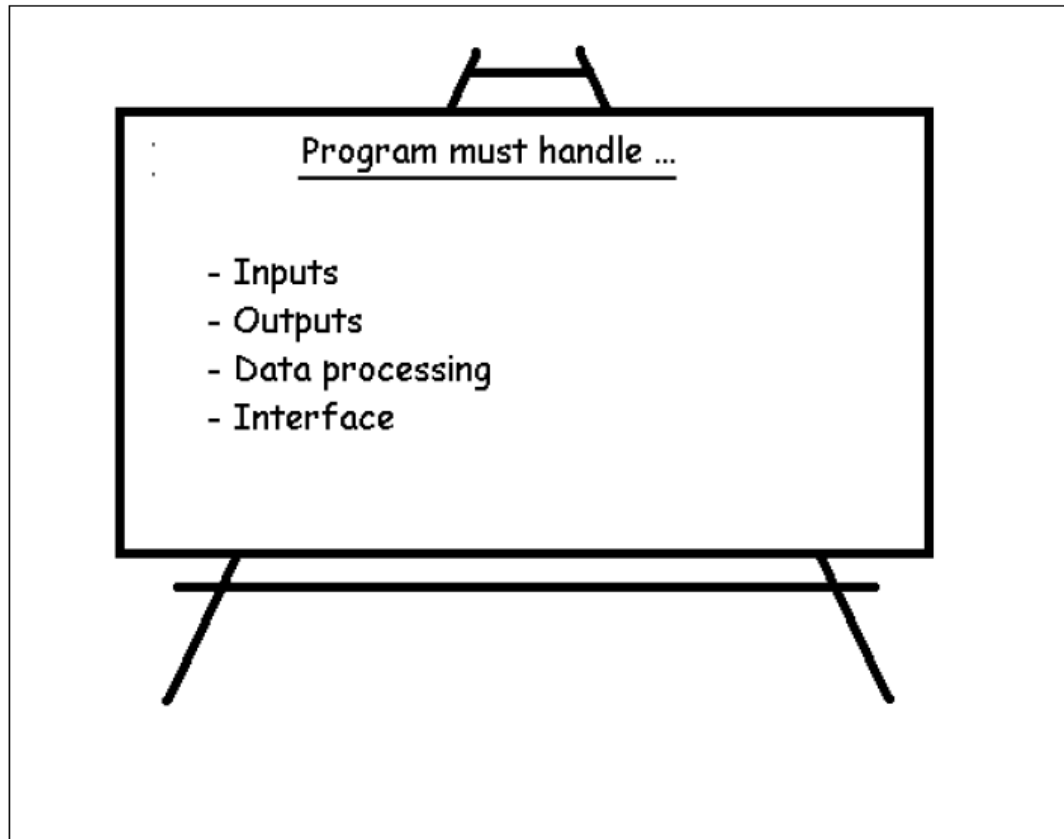
ما هي القضايا التي يحتاج ان يتعامل معها التطبيق ؟

لقد قمت بكتابة بعض التطبيقات لذلك إلى الآن لديك فكرة جيدة لبعض المشكلات أو القضايا التي ستتعامل معهم في الكود . بعضها مجرد قضايا معنية في البرمجة . و المشاكل الأخرى محددة في كتابة التطبيق ومتطلباته .

إليك بعض قضايا تطبيقاتك التي عليك ان تتعامل معها :

- **المدخلات من المستخدم :** لقد تم الاتفاق بالفعل على أن التطبيق سيكون ذو طبيعة رقمية في الأساس ولكن أيضاً سنحتاج طريقة لتحديد سواء كانت درجات الحرارة المدخلة فھرنهايت أو درجات مئوية .
- **المخرجات في الشاشة :** سيحتاج التطبيق أن يخرج درجات الحرارة التي تم تحويلها إلى الشاشة . مرة أخرى سيكون في الأساس قيم رقمية ولكن تحتاج أيضاً أن تضيف نص ليتحفظ بالرقم في سياقه .
- **المعالجة :** يحتاج الكود القدرة على تنفيذ تحويلين إثنين . ونحتاج أن ننقل إلى معادلة الكود التي فصلناها مسبقاً .
- **الواجهة :** سيحتاج التطبيق لواجهة لنوع واحد أو آخر ليكون من السهولة تصفح البرنامج وإستخدامه .

وهذا قد لخص في الشكل :



ما سبق هو مبادئ برنامج تحويل درجات الحرارة . وهذه النماذج هي الكثير من الكود الذي تحتاج ان تكتبه .

Moving on to the Coding Phase

الانتقال إلى مرحلة الكود

إلى هذا الحد , لقد أتممنا الكثير من العمل , سألنا الكثير من الأسئلة , وقمنا بالبحث في النظرية وراء التطبيق — وقد قمنا بفعل هذا كله بدون كتابة سطر واحد من الكود . الآن ما أتممنا فعله أننا جاهزون لننتقل للمرحلة التمهيدية للكود فقط . ولأننا سنستخدم C++ لهذه المشروع , يمكن ان نبدأ بإنشاء تخطيط مبدئي لتطبيق تحويل درجات الحرارة .

المكان الأفضل لتبدأ مع أى مشروع هو قالب كود أساسى :

```
// Temperature conversion application
#include <iostream.h>
void main()
{
    // code goes here!!!!
}
```

الان , لقد عرفنا أن هذا البرنامج يحتاج أن يقبل على الأقل مدخلين إثنين من لوحة المفاتيح :

- درجات الحرارة التى نحتاج أن نحولها .
- مدخلات لتحديد أى التحويلات نحتاج لتكون من الدرجة المئوية إلى فهرنهايت أو من فهرنهايت إلى الدرجة المئوية .

لذلك , يمكننا أن نبدأ بإعلان المتغيرات التى سنحتاجها . لدينا بالفعل بعض القرارات لنصنعها :

- أى نموذج إدخال البيانات لدرجات الحرارة سيكون ؟ ستكون أرقام صحيحة (كل الأرقام) أو أرقام تقبل الكسور ؟
- كيف سيحدد المستخدم سواء كان التحويل من فهرنهايت إلى درجة مئوية أو الدرجة المئوية إلى فهرنهايت ؟

هذه الأسئلة للعميل — خاصة في أسئلة الأرقام الصحيحة ضد الأرقام ذات الفاصلة العائمة لأن هذا له تأثير على الدقة . لأن العميل يريد من التطبيق معالجة درجة حرارة الهواء ولا يحتاج إلى دقة عالية , فالأرقام الصحيحة ستكون جيدة هنا .

فمن أجل إختيار نوع التحويل للمدخلات , سيقوم بهذا حرف واحد بسيط (C ليحول من الدرجة المئوية إلى فهرنهايت وحرف F للتحويل من فهرنهايت إلى الدرجة المئوية)

مرة أخرى , تأكد من تلخيص هذه المعلومات لذا لا تنسى هذا لاحقاً . (إنظر الشكل)

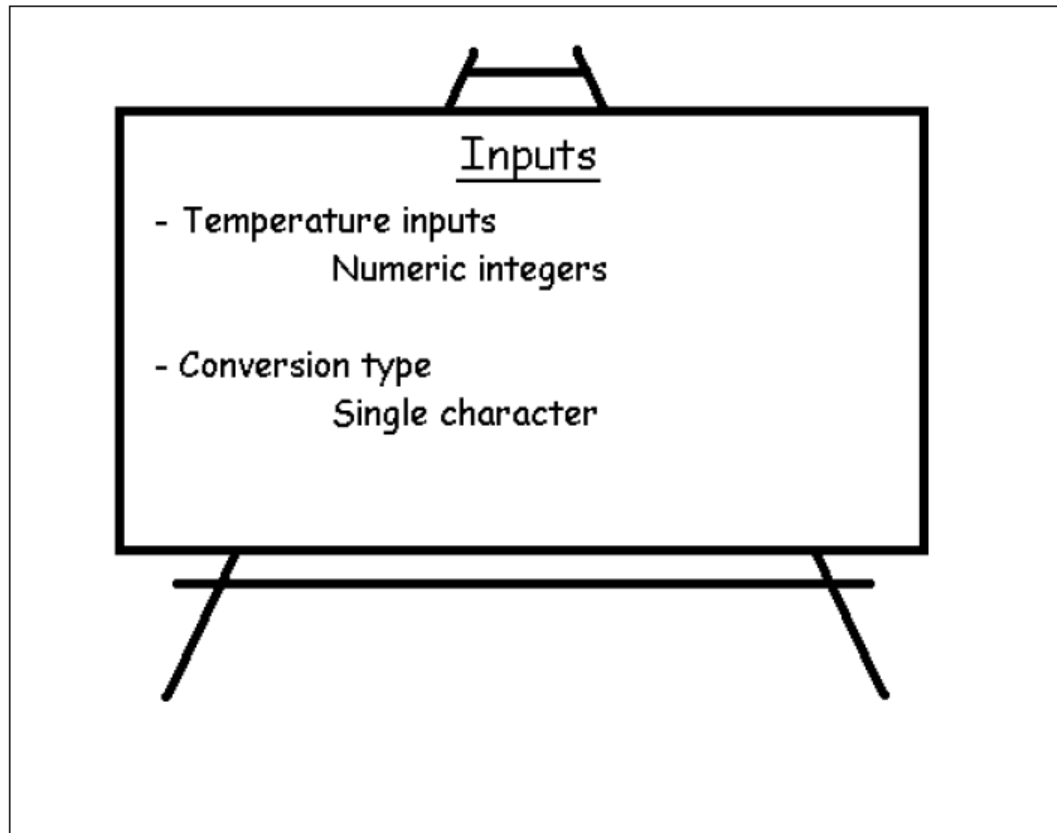
يمكن أن نبدأ بإدماج هذا الكود . أولاً , يمكننا ان نعلن عن المتغيرات للمدخلين . سيكون هكذا :

```
TempInput
ConversionType

// Temperature conversion application
#include <iostream.h>
void main()
{
```



```
int TempInput;  
char ConversionType;  
}
```



بسبب أن البرنامج يعتمد على المدخلات . يمكن ان نستخدم cin لتبدأ السماح للمدخلات التي تدخلها على التطبيق . الأول , مدخلات درجات الحرارة :

```
// Temperature conversion application  
#include <iostream.h>  
void main()  
{  
    int TempInput;  
    char ConversionType;  
    cin >> TempInput;  
}
```

يمكننا أن نسمح للمدخلات لنوع التحويل . إلى هذه المرحلة لا نحتاج أن نكون قلقين عن ما سنستخدم ؟

```
// Temperature conversion application  
#include <iostream.h>  
void main()  
{  
    int TempInput;  
    char ConversionType;  
    cin >> TempInput;  
    cin >> ConversionType;  
}
```

تذكر أيضاً أنك تحتاج إلى إضافة تعليقات لتتأكد أن المشروع يبقى في التعقب ويمكنك من تذكر كل شيء تفعله

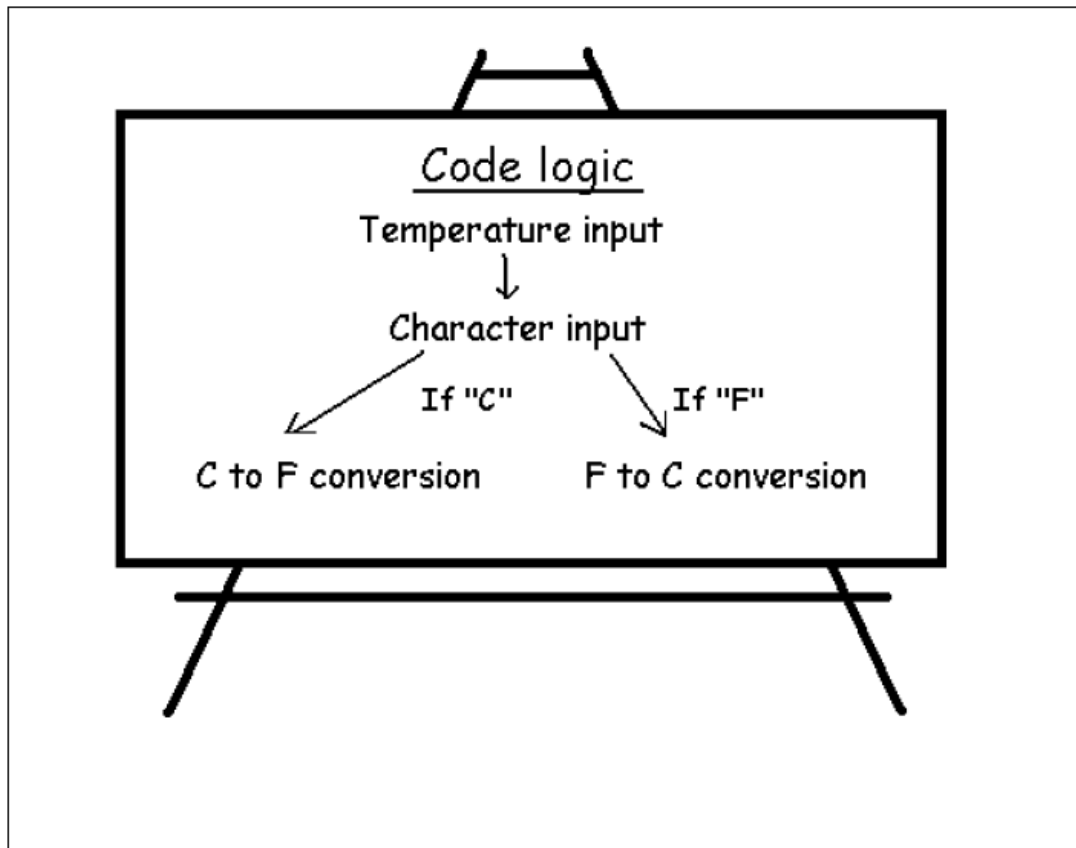
```
// Temperature conversion application
#include <iostream.h>
void main()
{
    // Input temperature, numeric input
    int TempInput;

    // Input conversion type, single character
    // "C" will be used to represent C to F.
    // "F" will be used to represent F to C.
    char ConversionType;

    // Input the temperature
    cin >> TempInput;

    // Input conversion type
    cin >> ConversionType;
}
```

تم إعلان المتغيرات وأخذ المدخلات وإضافة التعليقات . الآن نحتاج ان نضيف بعض التركيب للكود حتى يمكن لنا تنفيذ التحويل على درجات الحرارة المدخلة . حيث أن التحويلات التي سنتنفذ تعتمد على المدخلات . كما هو ملخص في الشكل التالي :



فإذا كان المدخل هو C , فنحتاج إلى تنفيذ احد التحويلات , وإذا كانت F , سنحتاج إلى فعل الآخر . الأمر المنطق هنا يشير أن أمر IF الشرطية مناسب لجملة .

```
// Temperature conversion application
#include <iostream.h>
void main()
{
    // Input temperature, numeric input
    int TempInput;

    // Input conversion type, single character
    // "C" will be used to represent C to F.
    // "F" will be used to represent F to C.
    char ConversionType;

    // Input the temperature
    cin >> TempInput;

    // Input conversion type
    cin >> ConversionType;

    // Conditionals below
    if (ConversionType == 'C')
        // Calculation code here
    if (ConversionType == 'F')
        // Calculation code here
}
```

تذكر أنه عندما تدخل الحروف لا يمكننا أن نفحص مباشرة وأن تختبر كود النظام العشري ASCII لهذا الحرف . وهذا هو :

❑ C = 67

❑ F = 70

يمكننا الآن ان نضيف هذا إلى الكود :

```
// Temperature conversion application
#include <iostream.h>
void main()
{
    // Input temperature, numeric input
    int TempInput;

    // Input conversion type, single character
    // "C" will be used to represent C to F.
    // "F" will be used to represent F to C.
    char ConversionType;

    // Input the temperature

    cin >> TempInput;

    // Input conversion type
    cin >> ConversionType;

    // Conditionals below
    if (ConversionType == 'C')
        // Calculation code here for C to F
    if (ConversionType == 'F')
        // Calculation code here for F to C
}
```

الآن نحن جاهزون للانتقال إلى الجزء الرئيسي من الكود — سيكون هذا هو الكود الذي يأخذ المدخلات ويقوم بتحويلها إلى وحدة قياس درجات الحرارة المناسبة .

الشيء الأول الذي نحتاجه هو متغير آخر , وسيستخدم للإسك بدرجة الحرارة المخرجة , التي ستكون رقم صحيح آخر :

```
// Temperature conversion application
#include <iostream.h>
void main()
{
    // Input temperature, numeric input
    int TempInput;

    // Output temperature, numeric output
    int TempOutput;

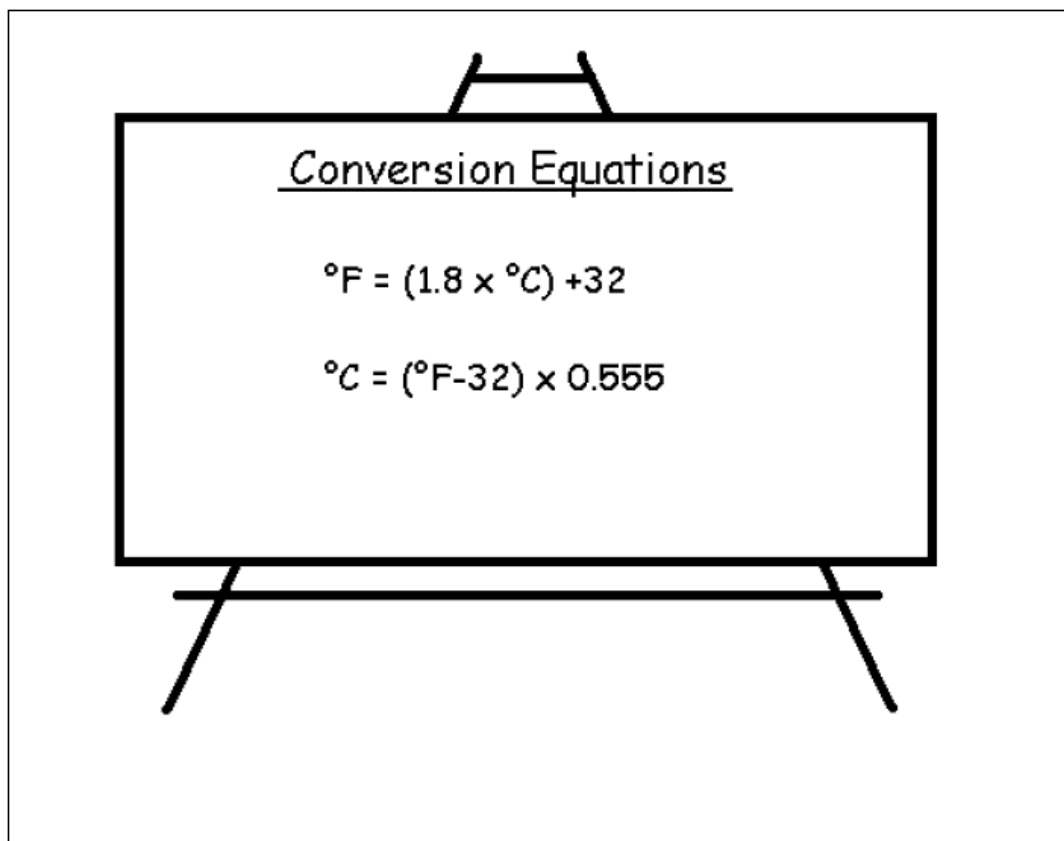
    // Input conversion type, single character
    // *C* will be used to represent C to F.
    // *F* will be used to represent F to C.
    char ConversionType;

    // Input the temperature
    cin >> TempInput;

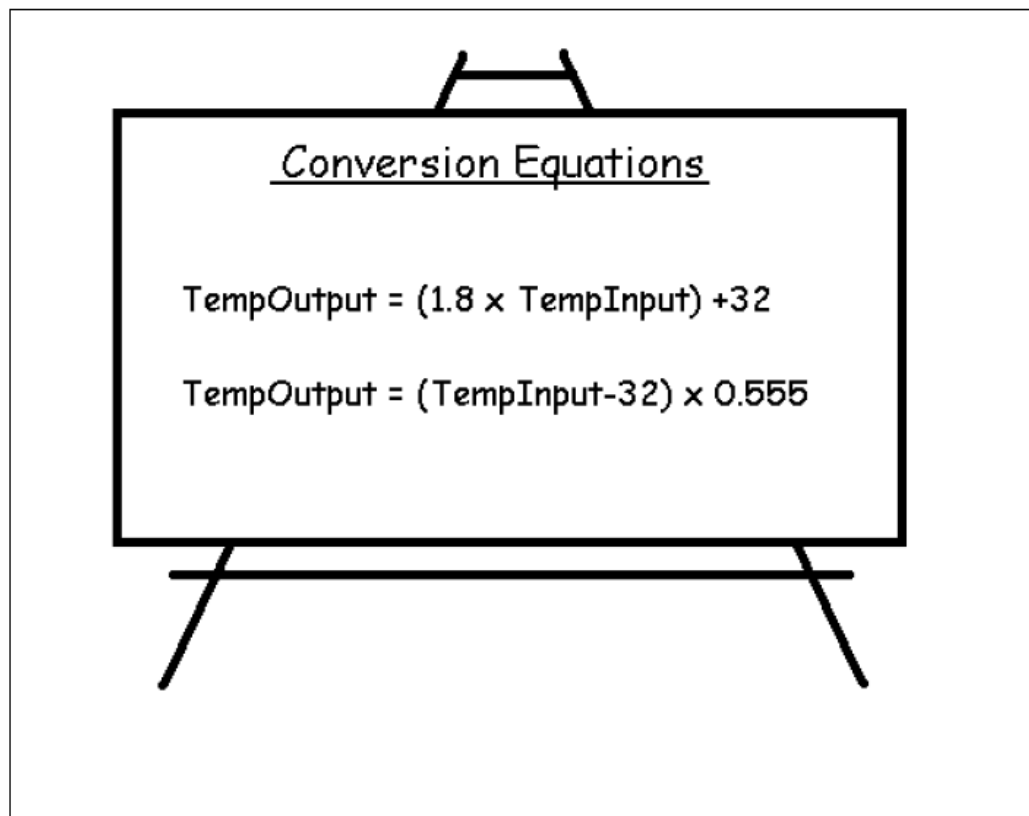
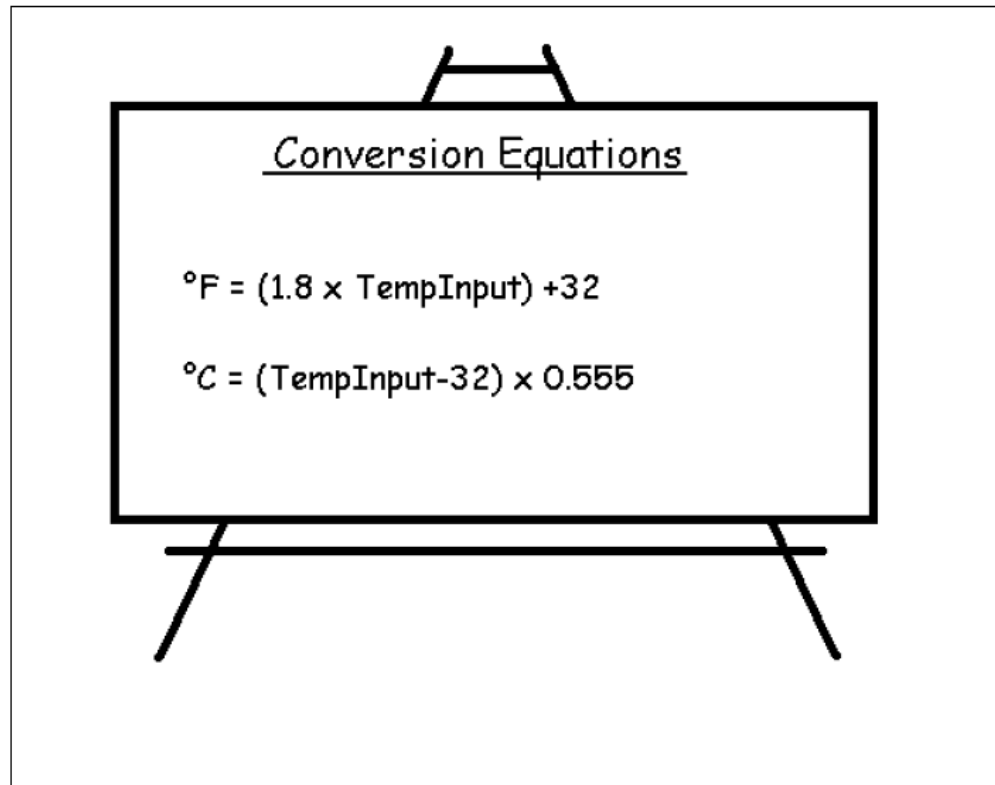
    // Input conversion type
    cin >> ConversionType;

    // Conditionals below
    if (ConversionType == 67)
        // Calculation code here for C to F
    if (ConversionType == 70)
        // Calculation code here for F to C
}
```

لذا , إلى تحويل درجات الحرارة من المدخلات إلى المخرجات . ولفعل هذا نحتاج إن نشير إلى معادلات التحويل التي لدينا مسبقاً (إنظر الشكل)



ما يمكنك فعله الآن هو اخذ المدخلات ومتغيرات الإخراج ودمجهم في المعادلة . تذكر ان المدخلات التي تحتاج أن تدخلها هي جسم المعادلة (إنظر الشكل) والمخرجات هي حل هذه المعادلة (إنظر الشكل)



يمكن الآن إدماج الكود , تذكر أن علامة * تستخدم في الرياضيات لتمثل علامة الضرب وليست X .

```
// Temperature conversion application
#include <iostream.h>
void main()
{
    // Input temperature, numeric input
    int TempInput;

    // Output temperature, numeric output
    int TempOutput;

    // Input conversion type, single character
    // "C" will be used to represent C to F.
    // "F" will be used to represent F to C.
    char ConversionType;

    // Input the temperature
    cin >> TempInput;

    // Input conversion type
    cin >> ConversionType;

    // Conditionals below
    if (ConversionType == 67)
        // Calculation code here for C to F
        TempOutput = (1.8 * TempInput) + 32;
    if (ConversionType == 70)

        // Calculation code here for F to C
        TempOutput = (TempInput - 32) * 0.555;

}
```

مع إضافة مخرج واحد بسيط إلى الشاشة , يمكنك الآن أن تفسر الكود وتختبره مع العديد من المدخلات وفحص المخرجات :

```

// Temperature conversion application
#include <iostream.h>
void main()
{
    // Input temperature, numeric input
    int TempInput;

    // Output temperature, numeric output
    int TempOutput;

    // Input conversion type, single character
    // "C" will be used to represent C to F
    // "F" will be used to represent F to C
    char ConversionType;

    // Input the temperature
    cin >> TempInput;

    // Input conversion type
    cin >> ConversionType;

    // Conditionals below
    if (ConversionType == 67)
        // Calculation code here for C to F
        TempOutput = (1.8 * TempInput) + 32;
    if (ConversionType == 70)
        // Calculation code here for F to C
        TempOutput = (TempInput - 32) * 0.555;
    cout << TempOutput << endl;
}

```

إحفظ مصدر الكود , وقم بتسمية tempconv.cpp ويمكن تفسيره على الشكل التالي :

```

C:\WINDOWS\System32\cmd.exe
D:\Prog>hcc32 tempconv.cpp
Borland C++ 5.5 for Win32 Copyright (c) 1993, 2000 Borland
tempconv.cpp:
Turbo Incremental Link 5.00 Copyright (c) 1997, 2000 Borland
D:\Prog>

```

بعد عملية التفسير , إختبر الكود مع بعض المدخلات وإفحصه لكي تتأكد أن المخرجات تعمل كما ينبغي لكل تحويل . إلقى نظرة على الشكل لتعدد المدخلات والمخرجات .

```
C:\WINDOWS\System32\cmd.exe
D:\Prog>hcc32 tempconv.cpp
Borland C++ 5.5 for Win32 Copyright (c) 1993, 2000 Borland
tempconv.cpp:
Turbo Incremental Link 5.00 Copyright (c) 1997, 2000 Borland
D:\Prog>tempconv
100
C
The answer is 212
D:\Prog>
```

Improving the Code

تحسين الكود

لديك كود يعمل ومخرجاته صحيحة, الأمر التالي الذي تحتاج أن تفعله ان يكون التطبيق أسهل لإستخدام . خذ خطوة للخلف من التطبيق وإلقى نظرة منتقده , أعتقد كحد أدنى انه يحتاج التالي :

- نص مبدئي يعرض يخبر المستخدم ما هو التطبيق .
- تعليمات مبدئية للإستخدام .
- تعليق على المدخلات .
- تعليق على المخرجات .

```
// Temperature conversion application
#include <iostream.h>
void main()
{
    // Input temperature, numeric input
    int TempInput;

    // Output temperature, numeric output
    int TempOutput;

    // Input conversion type, single character
    // "C" will be used to represent C to F.
    // "F" will be used to represent F to C.
    char ConversionType;

    // Welcome message
    cout << "Tempperature Converter" << endl;
    cout << "-----" << endl;

    // Initial instruction for use
```



```

    cout << "Enter a temperature and choose the conversion you want to carry out:"
    << endl;
    // Input the temperature
    cout << "Enter a temperature: ";
    cin >> TempInput;

    // Input conversion type
    cout << "Enter conversion type." << endl;
    cout << "Type C followed by ENTER for Centigrade to Fahrenheit conversion." <<
    endl;
    cout << "Type F followed by ENTER for Fahrenheit to Centigrade conversion." <<
    endl;
    cin >> ConversionType;

    // Conditionals below
    if (ConversionType == 67)
        // Calculation code here for C to F
        TempOutput = (1.8 * TempInput) + 32;
    if (ConversionType == 70)
        // Calculation code here for F to C
        TempOutput = (TempInput - 32) * 0.555;
    cout << "Output: " << TempOutput << endl;
}

```

فكرة جيدة أن تقوم بتفسيره مرة أخرى وإلقاء نظرة على ما إذا كان الكود يعمل وإذا كان تعليق الشاشة والتعليمات لها معنى ودقيقة (انظر الشكل)

```

C:\WINDOWS\System32\cmd.exe
tempconv.cpp:
Turbo Incremental Link 5.00 Copyright (c) 1997, 2000 Borland

D:\Prog>tempconv
100
C
The answer is 212

D:\Prog>bcc32 tempconv.cpp
Borland C++ for Win32 Copyright (c) 1993, 2000 Borland
tempconv.cpp:
Turbo Incremental Link 5.00 Copyright (c) 1997, 2000 Borland

D:\Prog>tempconv
Temperature Converter
Enter a temperature and choose the conversion you want to carry out:
Enter a temperature: 100
Enter conversion type.
Type C followed by ENTER for Centigrade to Fahrenheit conversion.
Type F followed by ENTER for Fahrenheit to Centigrade conversion.
C
Output: 212

D:\Prog>

```

تذكر أنك تبحث عن الوضوح والمنطق في التعليمات البرمجية . إخبار المستخدمين ماذا يفعلون قبل كل خطوة واخبرهم بعد كل خطوة . أيضاً , إنظر إلى الغموض في التعليمات البرمجية والشروحات للمخرجات . هناك جانب واحد للكود أعتقد أنه يمكننا فعله مع التحسينات وهذا آخر مخرج :

Output: 99

يوجد في هذا غموض , وتريد أن تتجنب هذا بأي ثمن .

أحد التحسينات هو جعل المخرجات واضحة ككونها درجة حرارة .

```

// Temperature conversion application
#include <iostream.h>
void main()
{
    // Input temperature, numeric input
    int TempInput;

    // Output temperature, numeric output
    int TempOutput;

    // Input conversion type, single character
    // "C" will be used to represent C to F.
    // "F" will be used to represent F to C.
    char ConversionType;

    // Welcome message
    cout << "Temperature Converter" << endl;
    cout << "-----" << endl;

    // Initial instruction for use
    cout << "Enter a temperature and choose the conversion you want to carry out:"
    << endl;
    // Input the temperature
    cout << "Enter a temperature: ";
    cin >> TempInput;

    // Input conversion type
    cout << "Enter conversion type." << endl;
    cout << "Type C followed by ENTER for Centigrade to Fahrenheit conversion." <<
    endl;
    cout << "Type F followed by ENTER for Fahrenheit to Centigrade conversion." <<
    endl;
    cin >> ConversionType;

    // Conditionals below
    if (ConversionType == 67)
        // Calculation code here for C to F
        TempOutput = (1.8 * TempInput) + 32;
    if (ConversionType == 70)
        // Calculation code here for F to C
        TempOutput = (TempInput - 32) * 0.555;
    cout << "Temperature output: " << TempOutput << endl;
}

```

فهذا يساعد , ولكن جزئياً فقط , خذ نظرة قريبة إلى المخرجات مرة أخرى :

```

D:\Prog>tempconv2
Temperature Converter
-----
Enter a temperature and choose the conversion you want to carry out:

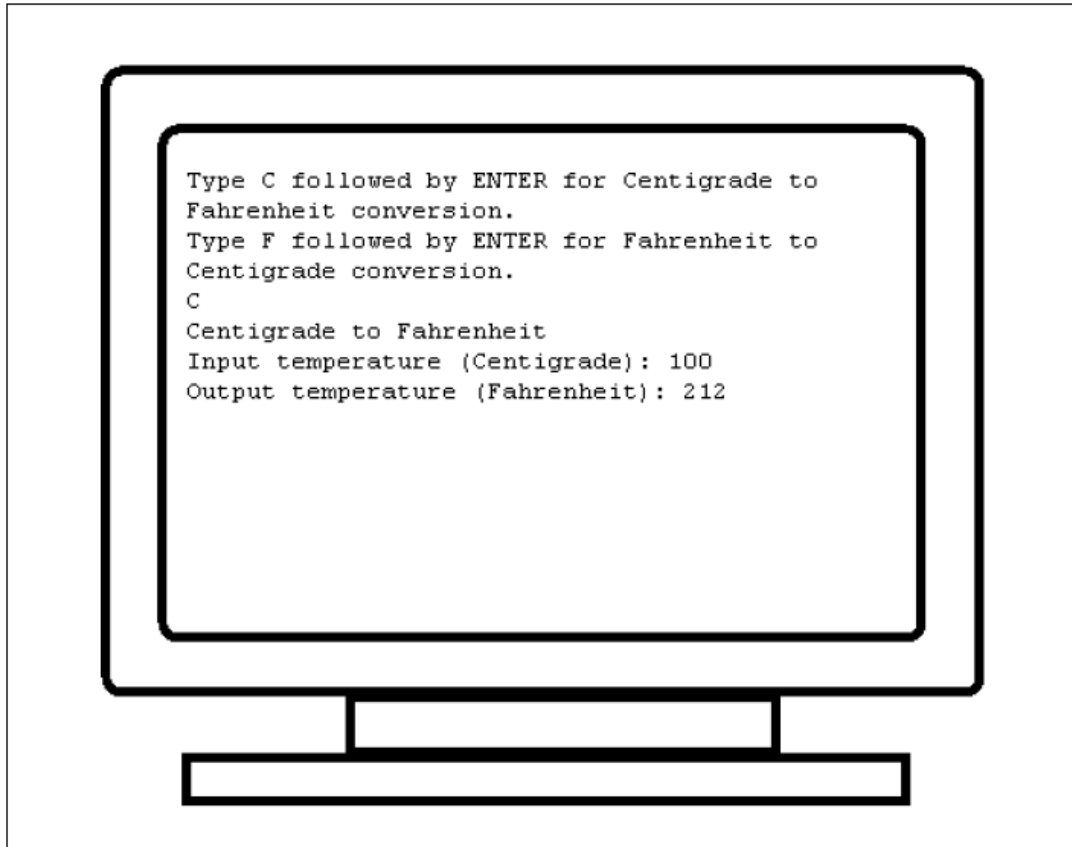
Enter a temperature: 100
Enter conversion type.
Type C followed by ENTER for Centigrade to Fahrenheit conversion.
Type F followed by ENTER for Fahrenheit to Centigrade conversion.
C
Temperature output: 212

```

هناك غموض في الكود في نهايته , حيث أنه لا يعطى المستخدم خلفية كافية . مثالياً , ينبغي ان تكون هناك خلفية تعتمد على اى الاختيارات التى قام بها المستخدم والتنسيق الجيد للمخرجات . فإنت تريد أن تكون المخرجات أوضح مما يكون كلما امكن .

عندما تبحث عن مدخلات جيدة , فإنها فكر طيبة في ان تخرج في النموذج الأولى للمخرجات , البحث عن طرق متنوعة يمكن ان ينسق بها المخرجات , و يمكن ان يكون جيداً على الورق . أو مذكرات , أو على الحاسب في حقة صور .

التخطيط الذي أعتقد أن يعمل جيداً لهذا في الشكل التالي :



لإعادة إنشاء هذا في الكود الآن ، هناك طرق عديدة يمكنك من هذا , ولكن الطريقة الأسهل هي صنع إستخدام عظيم للجمل المنفذة بواسطة أمر الشرط if . ولتتفرع من هذه الأولى قم بإضافة أقواس متعرجة :

```

// Temperature conversion application
#include <iostream.h>
void main()
{
    // Input temperature, numeric input

    int TempInput;

    // Output temperature, numeric output
    int TempOutput;

    // Input conversion type, single character
    // "C" will be used to represent C to F.
    // "F" will be used to represent F to C.
    char ConversionType;

    // Welcome message
    cout << "Temperatature Converter" << endl;
    cout << "-----" << endl;

    // Initial instruction for use
    cout << "Enter a temperature and choose the conversion you want to carry out:"
<< endl;
    // Input the temperature
    cout << "Enter a temperature: ";
    cin >> TempInput;

    // Input conversion type
    cout << "Enter conversion type." << endl;
    cout << "Type C followed by ENTER for Centigrade to Fahrenheit conversion." <<
endl;
    cout << "Type F followed by ENTER for Fahrenheit to Centigrade conversion." <<
endl;
    cin >> ConversionType;

    // Conditionals below
    if (ConversionType == 67)
        // Calculation code here for C to F
        {
            TempOutput = (1.8 * TempInput) + 32;
        }
    if (ConversionType == 70)
        // Calculation code here for F to C
        {
            TempOutput = (TempInput - 32) * 0.555;
        }
    cout << "Temperature output: " << TempOutput << endl;
}

```

الآن هذه الدوال يمكنها معالجة المخرجات المناسبة إلى الشاشة . أولاً، أضيف إلى كل تأكيد تخبر المستخدم ماذا يحدث :

```

// Temperature conversion application
#include <iostream.h>
void main()
{
    // Input temperature, numeric input

```

```

int TempInput;

// Output temperature, numeric output
int TempOutput;

// Input conversion type, single character
// "C" will be used to represent C to F.
// "F" will be used to represent F to C.
char ConversionType;

// Welcome message
cout << "Temperature Converter" << endl;
cout << "-----" << endl;

// Initial instruction for use
cout << "Enter a temperature and choose the conversion you want to carry out:"
<< endl;
// Input the temperature
cout << "Enter a temperature: ";
cin >> TempInput;

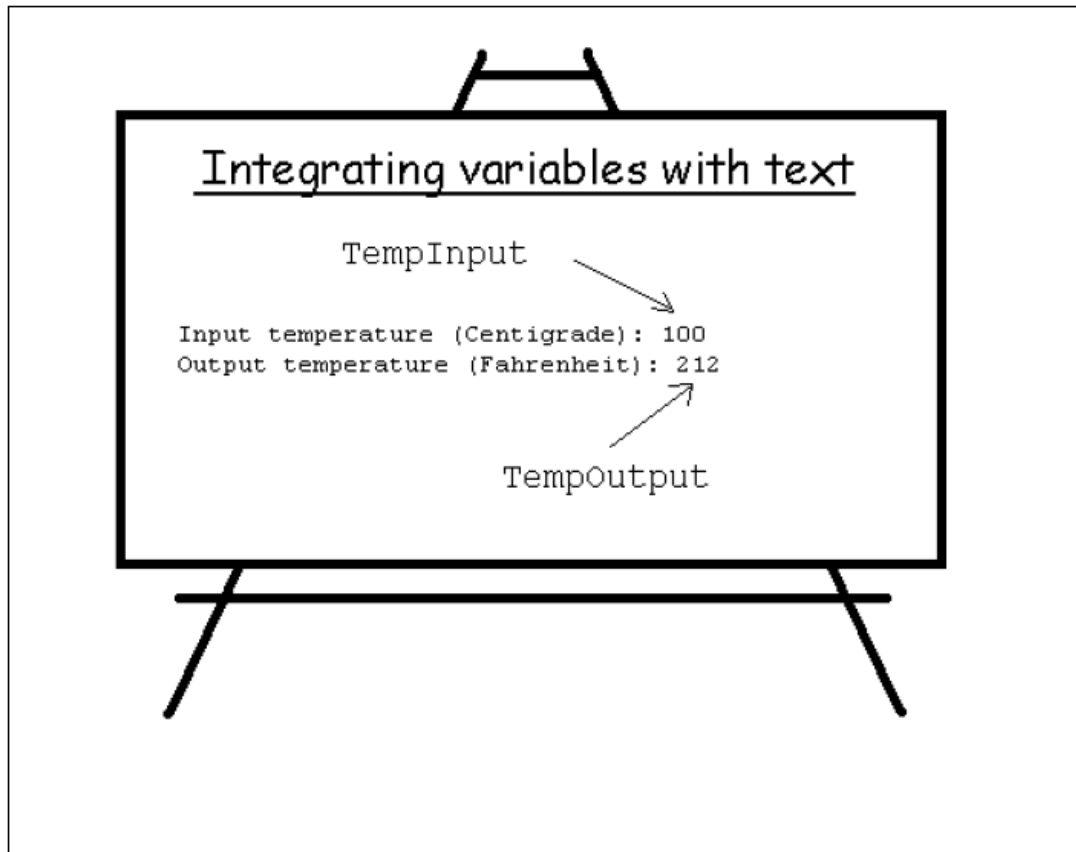
// Input conversion type
cout << "Enter conversion type." << endl;
cout << "Type C followed by ENTER for Centigrade to Fahrenheit conversion." <<
endl;
cout << "Type F followed by ENTER for Fahrenheit to Centigrade conversion." <<
endl;
cin >> ConversionType;

// Conditionals below
if (ConversionType == 67)
    // Calculation code here for C to F
    {
        cout << "Centigrade to Fahrenheit conversion:" << endl;
        TempOutput = (1.8 * TempInput) + 32;
    }
if (ConversionType == 70)
    // Calculation code here for F to C
    {
        cout << "Fahrenheit to Centigrade conversion:" << endl;
        TempOutput = (TempInput - 32) * 0.555;
    }
cout << "Temperature output: " << TempOutput << endl;
}

```

بعد فعل ذلك , الشئ المنطقي هو أخذ ناتج المخرجات وتضعها داخل دوال .

التنسيق السليم للمخرجات يمكن ان ينجز بإدماج المتغيرات مع المخرجات المختبرة لإنجاز المخرجات المرغوبة , كما يظهر في الشكل .



يمكن أن تترجم هذه التعليمات للعمل مع كود C++ كما يلي :

```
// Temperature conversion application
#include <iostream.h>
void main()
{
    // Input temperature, numeric input.
    int TempInput;

    // Output temperature, numeric output
    int TempOutput;

    // Input conversion type, single character
    // "C" will be used to represent C to F.
    // "F" will be used to represent F to C.
    char ConversionType;

    // Welcome message
    cout << "Temperature Converter" << endl;
    cout << "-----" << endl;

    // Initial instruction for use
    cout << "Enter a temperature and choose the conversion you want to carry out:"
    << endl;
    // Input the temperature
    cout << "Enter a temperature: ";
    cin >> TempInput;

    // Input conversion type
```

```

        cout << "Enter conversion type." << endl;
        cout << "Type C followed by ENTER for Centigrade to Fahrenheit conversion." <<
endl;
        cout << "Type F followed by ENTER for Fahrenheit to Centigrade conversion." <<
endl;
        cin >> ConversionType;

        // Conditionals below
        if (ConversionType == 67)
            // Calculation code here for C to F
            {
                cout << "Centigrade to Fahrenheit conversion:" << endl;
                TempOutput = (1.8 * TempInput) + 32;
                cout << "Input temperature (Centigrade): " << TempInput << endl;
                cout << "Output temperature (Fahrenheit): " << TempOutput << endl;
            }
        if (ConversionType == 70)
            // Calculation code here for F to C
            {
                cout << "Fahrenheit to Centigrade conversion:" << endl;
                TempOutput = (TempInput - 32) * 0.555;
                cout << "Input temperature (Fahrenheit): " << TempInput << endl;
                cout << "Output temperature (Centigrade): " << TempOutput << endl;
            }
    }
}

```

يمكنك الآن ان تفسر هذا الكود وإلقى نظرة فى الفعل كما يظهر فى الشكل :

```

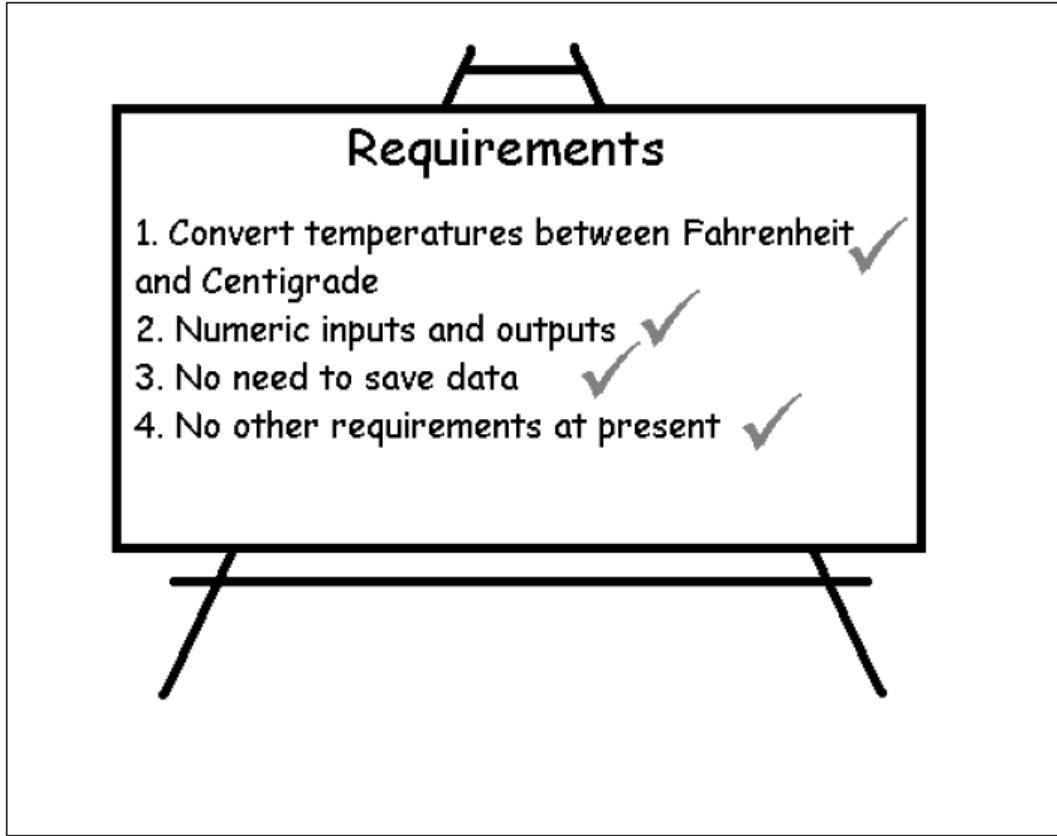
C:\WINDOWS\System32\cmd.exe
D:\Prog>bcc32 tempconvb.cpp
Borland C++ 5.5 for Win32 Copyright (c) 1993, 2000 Borland
tempconvb.cpp:
Turbo Incremental Link 5.00 Copyright (c) 1997, 2000 Borland

D:\Prog>tempconvb
Temperature Convertor
Enter a temperature and choose the conversion you want to carry out:
Enter a temperature: 100
Enter conversion type.
Type C followed by ENTER for Centigrade to Fahrenheit conversion.
Type F followed by ENTER for Fahrenheit to Centigrade conversion.
C
Centigrade to Fahrenheit conversion:
Input temperature (Centigrade): 100
Output temperature (Fahrenheit): 212
D:\Prog>

```

دعنا الآن نتوقف لننظر على البرنامج الذى أنشأناه ونرى إن كان يفى بمتطلبات المنتج المنصوص عليها فى البداية . إلقى نظرة على الشكل :

لذلك لابد من موافقة كل المتطلبات , لقد اخذنا المتطلبات المنصوص عليها مبدئياً , وبواسطة توجيه بعض الأسئلة البسيطة . قم بتقطع المشكلة إلى مشاكل صغيرة (مشاكل فرعية , إذا احببت) وتناول بعد ذلك كل منها بمفردها , استخدم البرمجة كأداة لتبنى بها الحلول . فقط كمثال الكاتب الذى يستخدم التدقيق الإملائى والنحوى لكتابة القصص . يستخدم المبرمج اداة البرمجة لإختياراته أوإختياراتها لبناء أجوبة للأسئلة المطروحة .



Summary

الملخص

في هذا الفصل ألقيت نظرة على كيف ينبغي أن تنهج حل المشاكل في التعامل مع البرمجة . كل المشاكل ستكون مختلفة . وكما أنه ليس هناك حل " واحد يناسب جميع المشكلات " . مع ذلك , بإتباع القواعد المبدئية , يمكنك تحويل معرفتك البرمجية إلى أداة لحل المشاكل وإنشاء كود ينتج نتائج حقيقية .

تذكر :

- إحصل على المتطلبات بوضوح .
- قم بتقطيع المشاكل إلى مشاكل صغيرة .
- قم بالبحث عن الجانب الذي لا تفهمه .
- ضاعف الفحص لجوانب من البرنامج التي لا يتوفر لديك فهم كافى لها مع العمل .
- إكتب الكود الذي يحل المشكلة .
- تأكد أن التطبيق ينتج النتائج بدقة .
- تأكد من أن التطبيق يعطى المستخدم المعلومات لكيفية إستخدامه . إضافة إلى ما تعنيه النتائج .
- إبتعد عن كل شئ غامض .

ضع هذه النقاط في الإعتبار . لتكون برمجتك مركزة أكثر , و سيقوم التطبيق الذي أنشأته بحل المشاكل المصمم لحلها .

9

Debugging

التصحيح

فى هذا الفصل , سنلقى نظرة على الأخطاء — شئ ما من المحتمل أنك شاهدته بقدر كبير كلما كتبت كود فى الفصول السابقة !

الأخطاء هى حقيقة الحياة عندما نأتى للبرمجة . الخدعة هى معرفة كيف تتابعهم وكيف تصلحهم قبل أن يضع الأشخاص الآخرين أيديهم فى البرنامج .

فى هذا الفصل , هنا سنلقى نظرة على الأخطاء البرمجية المتنوعة , وماذا يسببهم , وكيف تصلحهم . سبحث أيضاً فى بعض علوم النفس وراء اخطاء البرمجة وسبحث أيضاً فى لماذا ان بعضهم شائع أكثر من الآخر .

To Err Is Human

البشر يخطئون

دعنى أكون صادقاً معك , بينى وبينك ولا يكن احداً معاً .

هناك الكثير من المبرمجين من يعطى شعور بأنه يرى ان كل سطر من الكود والجميل التى قد أنشأوها تعمل من أول مرة , وكل مرة , بدون فشل , وبدون أخطاء , وهذا بدوره يضع الكثير من الضغط النفسى والضغط العصبى للذين هم جدد للبرمجة لفعل نفس الشئ . ويحصلون على الإنطباع بان إرتكاب الإخطاء والحصول على رسائل الخطأ هو شئ يخلج منه . وان هذه هى علامة الفشل وشئ ما محرج.

لاوضح هذا لك على الفور — الأخطاء هى حقيقة الحياة , وإن أحداً من المبرمجين أخبرك بأنه يمكنك أن تكتب كود كامل ممتاز طول الوقت فإنهم إما يكذبون وإما لم يقوموا بالكثير من البرمجة — على الأقل , البرمجة الجادة . فإذا كان المبرمجون يخرجون كود كامل , فإن شركات البرامج ستقتنص هؤلاء الناس . وتوفر لهم وظائف بأعلى أجرا , وتطلق النار على كل المبرمجين الأقل كفاءة , وسيتم إلغاء كل الإختبار و إصدارات البرنامج التجريبية. وتوفر الملايين من الدولارات , فإن إصدار برنامج متكامل بمعنى الكلمة لا يحتاج إلى حزم من الخدمات , أو صيانة بيانات , أو إضافة فيما بعد وبهذا توفر أموال أكثر . شركات البرمجة الكبرى لا تفعل هذا لأن "المبرمج الكامل" هو إسطورة نتجت عن مبرمجين آخرين لتعزيز وضعهم .

كل شخص يرتكب أخطاء , حتى في أبسط الأمور , تنسى شئ بسيط أو يحدث لك صرف إنتباه في وسط الجملة أو انك لم تنتهيه بالشكل الصحيح . أنا أرتكب أخطاء وسعيد أن اعترف بهذا , هناك أكواد في هذا الكتاب حيث اننى مبدئياً تركت شيئاً ما أو ربما جاءتني مكالمات هاتفية أو بريد إلكتروني وأنا في وسط الجملة ونسيت أن أكملها . أيضاً فقدت تعقب المتغيرات التي أستخدمها ثم إستخدمت واحد خطأ . هذه هي الحياة , إنها مزعجة , وإتمنى أن أكتب كود كامل كل الوقت لأن هذا سيكون معزز لى جداً . ويحفظ لى المزيد من الوقت . ولكن أنا بشر , أنا أقبل هذا , وأحسك ان تقبل بهذا أيضاً .

مبدئياً , أنت ملزم أن تصنع الأخطاء في الكود الخاص بك ببساطة لأنك تتعلم . التدقيق وبناء الجملة يأخذ وقت ليتم فهمه . المعاملات الغير مألوفة , وممارسات أساسيات الكود ربما تكون غريبة بالنسبة لك , إلى أن تجد الراحة معهم , فالأخطاء ستحدث . وإنظر إلى الأخطاء على أنها مؤشر التعلم . تظهر لك مساحات تحتاج ان تزيد فيها الإنتباه والتركيز أكثر , في حين ان الجمل الخالية من الأخطاء تنبأ بأن العمل على ما يرام .

واظب على البرمجة , وستلاحظ انك كلما تقدمت في المستوى كلما قلت اخطاءك — فهذا تقدم جيد وشئ ينبغي ان تفتخر به . ولكن لا تشغل بالك بمن يحاول إن يصبح 100% خالياً أخطاء . إترك هذا إلى المبرمجين " الكاملين " الذي يشعرون بأن لديهم ما يظهروه .

Errors, Errors, Errors! اخطاء , أخطاء , أخطاء !

إرتكاب أخطاء ليس شيئاً نفكر فيه طوال اليوم , نحن نرتكب أخطاء ونأمل أن نصحيحها بعض الإسئلة الشائعة التي تسأل من الذين يبدؤان في البرمجة هي :

- هل الأخطاء كلها تنشأ نفس الشئ ؟
- هل هناك أنواع مختلفة من الأخطاء ؟

كلاهما سؤالين ممتازين . الإجابة على ذلك هي ان كل الأخطاء لم تصنع بنفس الطريقة ولهذا فهناك أنواع مختلفة من الأخطاء التي يمكن ان نجدها من خلال كود الحاسب .

دعنا نأخذ نظرة على أنواع الأخطاء التي يمكن ان تعرض من خلال كود الحاسب .

Different Kinds of Error الأنواع المختلفة من الأخطاء

كما ستري , هناك الكثير من انواع الأخطاء التي يمكن أن توجد في أسطر الكود . لحسن الحظ , يمكن أن تصنف هذه الأخطاء إلى مجموعات فيسهل علينا مناقشتهم .

هناك ثلاثة Classes أنواع من الأخطاء سنقوم بالبحث خلالهما هنا , ها هي :

- ❑ Compiler errors (أخطاء أثناء عملية التفسير)
- ❑ Runtime errors (أخطاء أثناء عملية التشغيل)
- ❑ Logic errors (اخطاء منطقية)

دعنا نلقى نظرة على كلا من هذه الأنواع من الأخطاء بدوره .

Compiler Error

أخطاء برنامج تحويل الكود

عادةً ما يكون هذا النوع من الأخطاء التي يراها معظم المبرمجين . هي أخطاء تظهر بواسطة البرنامج الذي يقوم بتحويل الكود compiler ويسمى مفسر مدمت في وقت تنفيذ العملية أو تحويل مصدر الكود إلى تطبيق مستقل بذاته .

نحن بالفعل بحثنا داخل هذا البرنامج , ولديك بعض أيادي الخبرة في أخذ الكود من مصدر الكود إلى تطبيق تنفيذي .

اهم ما تلاحظه هو أنك ستستقبل أخطاء منه فقط في حالة إستخدامك له . لذا , إذا كنت تستخدم Java, C++ , أو أى لغة أخرى تستخدم المفسر , ربما ترى أخطاء المفسر , وإذا كنت تستخدم لغة لا تحتاج التفسير (مثال JavaScript , VBScript) فلن ترى اخطاء المفسر لأنه لا يوجد مفسر معنياً في هذا . (ولكن لا تقل أنك لاترى أنه لا يوجد أخطاء قد ظهرت عند تحميل المترجم interpreter للكود)

أخطاء المفسر هي اخطاء موجودة في الكود قام المفسر بالتقاطها . المفسرات المختلفة تدقق في الكود بمستويات مختلفة في عملية البحث عن الأخطاء . في حين أن معظم يقوم بالإمساك ببعض الأخطاء الأكثر شيوعاً . فمن الهام لك أن تدرك حقاً من البداية أنك لن تمسك بكل الأخطاء .

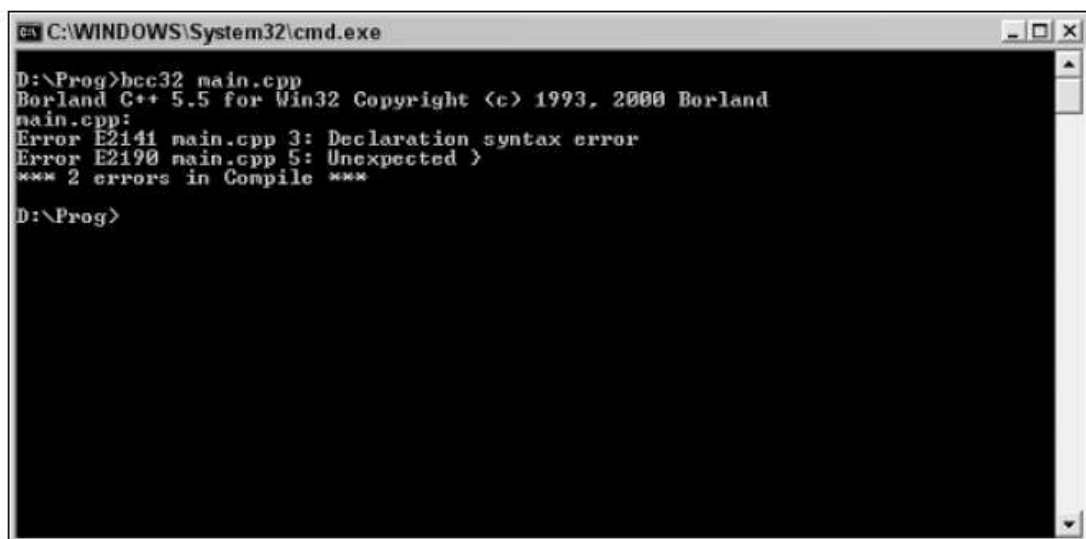
إلقى نظر على هذا الكود البسيط لـ C++

```
#include <iostream.h>
void main()
{
    cout << "Hello, World!" << endl;
}
```

قم بتعريفه ؟ هذا هو الكود الذي يعرض " Hello, World! " على الشاشة عند تنفيذه . ولكن هل لاحظت مشكلة الكود ؟ أمعن النظر في السطر الثالث , هناك عند الأقواس المتعرجة كيف ينبغي ان تكون :

```
#include <iostream.h>
void main()
{
    cout << "Hello, World!" << endl;
}
```

لاحظ انها ليست اقواس متعرجة , ولكن أقواس مفتوحة . ماذا تعتقد ان يفعل المفسر إذا حاولت أن تقوم بتفسيره ؟ حسناً , لن تحبه بالتأكيد! الشكل التالي يظهر ماذا عليه أن يقول :



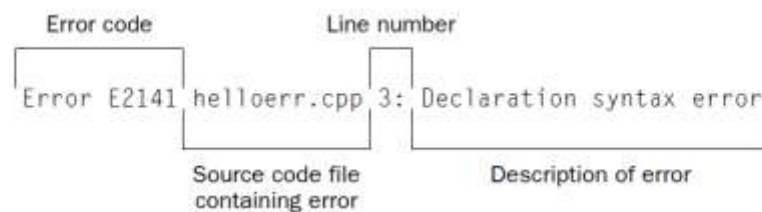
```
C:\WINDOWS\System32\cmd.exe
D:\Prog>bcc32 main.cpp
Borland C++ 5.5 for Win32 Copyright (c) 1993, 2000 Borland
main.cpp:
Error E2141 main.cpp 3: Declaration syntax error
Error E2190 main.cpp 5: Unexpected >
*** 2 errors in Compile ***
D:\Prog>
```

```
D:\Prog>bcc32 helloerr.cpp
Borland C++ 5.5 for Win32 Copyright (c) 1993, 2000 Borland
helloerr.cpp:
Error E2141 helloerr.cpp 3: Declaration syntax error
Error E2190 helloerr.cpp 5: Unexpected }
*** 2 errors in Compile ***
```

الرسالة تقول أن هناك إثنين من الأخطاء وهم في الحقيقة خطأ واحد فقط , ولكن دعنا نمنع النظر في الأخطاء التي تقول إنها موجودة . إليك السطرين الإثنين حيث يوجد الأخطاء المعلنه :

```
Error E2141 helloerr.cpp 3: Declaration syntax error
Error E2190 helloerr.cpp 5: Unexpected }
```

هناك أربعة أجزاء لرسائل الأخطاء , كما يظهر في الشكل



لذلك , لديك رسالة عطل تخبرك بنوعين من الأخطاء في سطرين منفصلين من الكود . أسطر الكود في السؤال هي :

Line 3:

```
#include <iostream.h>
void main()
{
    cout << "Hello, World!" << endl;
}
```

Line 5:

```
#include <iostream.h>
void main()
{
    cout << "Hello, World!" << endl;
}
```

من الجيد دائماً أن تنتظر إلى السطر الأول المحدد خلال الكود . كما ترى , في هذه الحالة المفسر محق , لقد رصد خطأ في بناء الكود , وهو مثل خطأ نحوي في الكتابة . ما عرفناه أن هناك تحديد وليس بتحدى هائل . ولكن ما الخطأ في السطر 5 ؟ حسناً , هذا الخطأ هو نتيجة ثانوية للخطأ في السطر 3 . لأننا إرتكبنا خطأ ومن المحتمل أننا لم نستخدم الأقواس المتعرجة . فقد أتى المفسر إلى السطر الخامس ورأى قوس الأغلاق المتعرج بدون رؤية قوس متعرج مفتوح . هذا بنفسه أيضاً خطأ , ولكن احدهم ينشأ بسبب خطأ سابقاً في الكود . إلقى نظرة على هذا الكود , إختلاف آخر في كود " Hello, World!" هذه المرة مع خطأ مختلف .

```
#include <iostream.h>
void main()
{
    cout << "Hello, World!" << endl;
}
```

كيف نكتشف الخطأ ؟

حسناً , إذا حاولت أن تفسر الكود , فستحصل على النتيجة المعروضة في الشكل التالي :



```
C:\WINDOWS\System32\cmd.exe
D:\Prog>bcc32 main.cpp
Borland C++ 5.5 for Win32 Copyright (c) 1993, 2000 Borland
main.cpp:
Error E2141 main.cpp 4: Declaration syntax error
Error E2190 main.cpp 5: Unexpected }
*** 2 errors in Compile ***
D:\Prog>
```

```
D:\Prog>bcc32 helloerr2.cpp
Borland C++ 5.5 for Win32 Copyright (c) 1993, 2000 Borland
helloerr2.cpp:
Error E2141 helloerr2.cpp 4: Declaration syntax error
Error E2190 helloerr2.cpp 5: Unexpected }
*** 2 errors in Compile ***
```

الأخطاء مرة أخرى .

السطر الرابع

```
#include <iostream.h>
void main()

    cout << "Hello, World!" << endl;
}
```

السطر الخامس

```
#include <iostream.h>
void main()

    cout << "Hello, World!" << endl;
}
```


هذه المرة السطر الأول تم إعلانه كخطأ وهذا مفضل . هذه جملة أمر وهي في نفسها صحيحة . دعنا ننتقل إلى السطر التالي ونلقى نظرة عليه . وهو إغلاق القوس المتعرج وينبغي أن يعطينا مفتاح لحل اللغز ... إلقى نظرة عودةً إلى السطر الخامس . هل تلاحظ شيء ما ؟ نعم , إفتتاح القوس المتعرج الآن مفقود .

أحياناً تحصل الأخطاء من المفسر واضحة وتشير مباشرةً إلى السطر الذي به المشكلة , بينما في أوقات أخرى تحتاج أن تلتفت قليلاً إلى أن تصل .

دعنا نبحث في بعض الأخطاء الأخرى . ليكون لديك بعض الممارسة .
ما الخطأ في هذا الكود ؟

```
#include <iostream.h>
void main()
{
    cout << "Hello, World!" << endl;
}
```

لست متأكداً ؟ دعنا نقوم بتفسيره . المخرجات تظهر بالشكل



```
C:\WINDOWS\System32\cmd.exe
D:\Prog>bcc32 main.cpp
Borland C++ 5.5 for Win32 Copyright (c) 1993, 2000 Borland
main.cpp:
Error E2380 main.cpp 4: Unterminated string or character constant in function ma
in()
Error E2379 main.cpp 5: Statement missing ; in function main()
*** 2 errors in Compile ***
D:\Prog>
```

```
D:\Prog>bcc32 helloerr3.cpp
Borland C++ 5.5 for Win32 Copyright (c) 1993, 2000 Borland
helloerr3.cpp:
Error E2380 helloerr3.cpp 4: Unterminated string or character constant in function
main()
Error E2379 helloerr3.cpp 5: Statement missing ; in function main()
*** 2 errors in Compile ***
```

الأخطاء موجودة في السطر الرابع والخامس . الخطأ في السطر الرابع يقول "سلسلة غير منتهية" وهو مصطلح آخر يظهر حيث احتمالاً أنك قد نسيت ان تغلق أقواس القيم الحرفية باستخدام العلامات ". هل يمكن ان ترى مثال لهذه المشكلة في السطر الرابع ؟ نعم , علامة تنصيب واحدة قد استخدمت بدلاً من علامتين تنصيب .

نتجت هذه المشكلة في الخطأ الثاني أيضاً حيث أن المفسر ويتوقع أن يكون هناك جملة أخرى في الكود لأن الأولى تحتوى على خطأ . تقول القاعدة إنك إذا حصلت على رسائل اخطاء قد عرضت لك , أفحص الخطأ الأول الذي وجدته وأعد تشغيل برنامج المفسر مرة أخرى — ربما يحتوى الكود على خطأ واحد , والذي بدوره تسبب في سلسلة من الأخطاء . فحص هذا الخطأ ربما يتسبب في إخفاء الآخرين .

إليك كتلة من الكود تحتوى على مشكلة :

```
#include <iostream.h>
void main()
{
    cout << "Hello, World!" <<< endl
}
```

ولأننا قد ألقينا نظرة على نفس القطعة من الكود منذ قليل الآن , فهناك احتمالات أن يقفز إليك الخطأ ! ومع ذلك , دعنا نفسره على أي حال ونرى ما سيحدث , مخرجات التفسير تظهر في الشكل

```
C:\WINDOWS\System32\cmd.exe
D:\Prog>bcc32 main.cpp
Borland C++ 5.5 for Win32 Copyright (c) 1993, 2000 Borland
main.cpp:
Error E2188 main.cpp 4: Expression syntax in function main()
Error E2379 main.cpp 5: Statement missing ; in function main()
*** 2 errors in Compile ***
D:\Prog>
```

```
D:\Prog>bcc32 helloerr4.cpp
Borland C++ 5.5 for Win32 Copyright (c) 1993, 2000 Borland
helloerr4.cpp:
Error E2188 helloerr4.cpp 4: Expression syntax in function main()
Error E2379 helloerr4.cpp 5: Statement missing ; in function main()
*** 2 errors in Compile ***
```

هناك خطأين قد تم إدراجهم , السطر الرابع والسطر الخامس . يقول برنامج المفسر أن السطر الرابع به خطأ في بناء الجملة لذا هذه بداية جيدة للبحث في الكود :

```
#include <iostream.h>
void main()
{
    cout << "Hello, World!" <<< endl
}
```

وهناك هو عليه , فقط بعد <<< "Hello, World!" ينبغي في الحقيقة أن يكون << . دعنا نفحصه ونعيد إعادة تشغيل برنامج المفسر ليقوم بإعادة تفسير الكود .

```
#include <iostream.h>
void main()
{
    cout << "Hello, World!" << endl
}
```

ولكن مازال هناك خطأ ! مازال بسبب الجملة المفقودة في السطر الخامس كما يظهر في الشكل

```
C:\WINDOWS\System32\cmd.exe
D:\Prog>bcc32 main.cpp
Borland C++ 5.5 for Win32 Copyright (c) 1993, 2000 Borland
main.cpp:
Error E2379 main.cpp 5: Statement missing ; in function main()
*** 1 errors in Compile ***
D:\Prog>
```



```
D:\Prog>bcc32 helloerr4.cpp
Borland C++ 5.5 for Win32 Copyright (c) 1993, 2000 Borland
helloerr4.cpp:
Error E2379 helloerr4.cpp 5: Statement missing ; in function main()
*** 1 errors in Compile ***
```

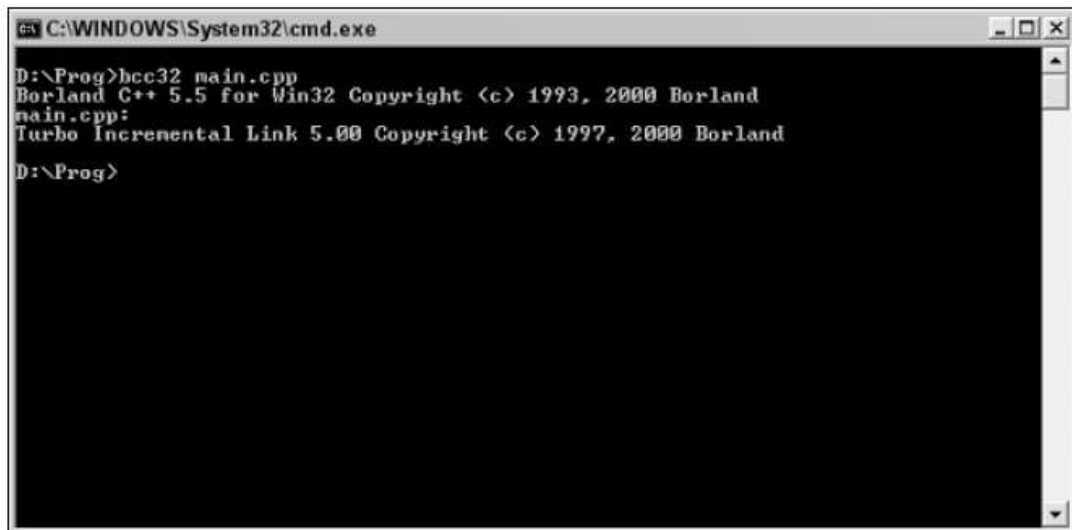
السطر الخامس هو إغلاق القوس المتعرج فقط , لذلك فالمشكلة لابد ان تكون أعلاه . إبحث بحرص خلال الجملة عن ما هو واضح : إستخدام خاطئ لعلامات التنصيص , تعليمات أوامر برمجية, إنهاء السطر ... وها هي ! علامة إنهاء السطر ; semicolon مفقودة .

دعنا نضيف هذا ونحاول مرة أخرى

```
#include <iostream.h>
void main()
{
    cout << "Hello, World!" << endl;
}
```

الان يعمل كل شئ كما هو متوقع

كما بالشكل



```
C:\WINDOWS\System32\cmd.exe
D:\Prog>bcc32 main.cpp
Borland C++ 5.5 for Win32 Copyright (c) 1993, 2000 Borland
main.cpp:
Turbo Incremental Link 5.00 Copyright (c) 1997, 2000 Borland
D:\Prog>
```

إلى هذا الحد بحثنا داخل تركيبات بسيطة من الكود , حيث من السهل ان نكتشف الأخطاء في الكود. لإن بإمكاننا أن نقرأ و نراجع الكود في دقائق قليلة. الان , دعنا ننقل إلى أخطاء برنامج المفسر في كميات كبيرة من الكود .

تذكر كود الحاسبة في الفصل السابع بعنوان "تركيبات الكود", ها هو مرة أخرى , ولكن هذه المرة يحتوى على خطأ :

```

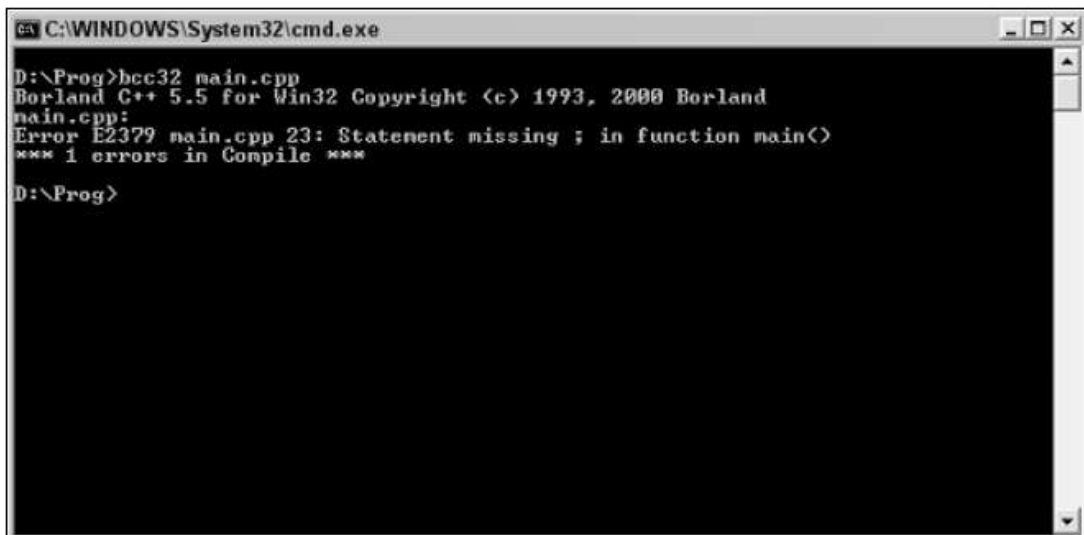
#include <iostream.h>
void main ()
{
    float num1;
    float num2;
    char op;
    float ans;
    cout << "Please enter a number: ";
    cin >> num1;
    cout << "Please enter another number: ";
    cin >> num2;
    cout << "Press A to add the two numbers."
        << endl
        << "Press S to subtract the two numbers."
        << endl
        << "Press M to multiply the two numbers."
        << endl
        << "Press D to divide the two numbers."
        << endl;
    cin >> op;

    if (op == 65)
        ans = num1 + num2;
    if (op == 83)
        ans = num1 - num2;
    if (op == 77)
        ans = num1 * num2;
    if (op == 68)
        ans = num1 / num2;
    cout << "The answer is " << ans << endl;
}

```

ليس من السهل ان تكتشف الخطأ هذه المرة , أليس كذلك ! وهذا ليس كود خاص مطول بدرجة كبيرة . — فقط 30 سطر طوال ولكن بالفعل يوحى بعض الشيء أنه غير عملي .

الطريقة الأفضل لإكتشاف هذا الخطأ هي تفسير الكود ونرى على ماذا سنحصل .
مخرجات التفسير تظهر في الشكل



```

C:\WINDOWS\System32\cmd.exe
D:\Prog>bcc32 main.cpp
Borland C++ 5.5 for Win32 Copyright (c) 1993, 2000 Borland
main.cpp:
Error E2379 main.cpp 23: Statement missing ; in function main()
*** 1 errors in Compile ***
D:\Prog>

```

```

D:\Prog>bcc32 calcerr.cpp
Borland C++ 5.5 for Win32 Copyright (c) 1993, 2000 Borland
calcerr.cpp:
Error E2379 calcerr.cpp 23: Statement missing ; in function main()
*** 1 errors in Compile ***

```

فالرسالة تقول أن المشكلة مع السطر 23 . دعنا نلقى نظرة عليه .

```
#include <iostream.h>
void main ()
{
    float num1;
    float num2;
    char op;
    float ans;
    cout << "Please enter a number: ";
    cin >> num1;
    cout << "Please enter another number: ";
    cin >> num2;
    cout << "Press A to add the two numbers."
        << endl
        << "Press S to subtract the two numbers."
        << endl
        << "Press M to multiply the two numbers."
        << endl
        << "Press D to divide the two numbers."
        << endl;
    cin >> op;
    if (op == 65)
        ans = num1 + num2;
    if (op == 83)
        ans = num1 - num2;
    if (op == 77)
        ans = num1 * num2;
    if (op == 68)
        ans = num1 / num2;
    cout << "The answer is " << ans << endl;
}
```

لا شيء خطأ بهذا السطر لأنه لا حاجة لعلامة الإنهاء ; semicolon هناك ولإنها ليست جملة أو امر مكتمل . لذلك , يبدو أنها حولها . ومن المحتمل أنك قد رأيتها . إنه السطر الذي يعلو سطر الخطأ المظلل فهو الذي يفتقد علامة إنهاء السطر ; . [إحفظ اسم هذه العلامة لإنهاتستخدم في أكثر من لغة مثال , JavaScript , C# , Java , PHP وإينما وجدتها فلها نفس الاستخدام وهو لتعريف المفسر أن هذه نهاية الجملة] وإليك خطأ مرة أخرى , هذه المرة أصعب قليلاً ولكنها مشكلة شائعة جداً مع الكود

```
#include <iostream.h>
void main ()
{
    float num1;
    float num2;
    char op1;
    float ans;
    cout << "Please enter a number: ";
    cin >> num1;
    cout << "Please enter another number: ";
    cin >> num2;
    cout << "Press A to add the two numbers."
        << endl
        << "Press S to subtract the two numbers."
        << endl
        << "Press M to multiply the two numbers."
        << endl
        << "Press D to divide the two numbers."
        << endl;
    cin >> op;
    if (op == 65)
        ans = num1 + num2;
    if (op == 83)
        ans = num1 - num2;
    if (op == 77)
        ans = num1 * num2;
    if (op == 68)
        ans = num1 / num2;
    cout << "The answer is " << ans << endl;
}
```

هل لاحظت الخطأ ؟ صدقاً , أنا لا أتوقع ان تكتشف هذه المشكلة لأنها مخفية . الطريقة الأفضل لإكتشاف هذا الخطأ هو تفسير الكود (تجد المخرجات فى الشكل التالى)

```
C:\WINDOWS\System32\cmd.exe
D:\Prog>bcc32 main.cpp
Borland C++ 5.5 for Win32 Copyright (c) 1993, 2000 Borland
main.cpp:
Error E2451 main.cpp 20: Undefined symbol 'op' in function main()
*** 1 errors in Compile ***
D:\Prog>
```

```
D:\Prog>bcc32 calcerr1.cpp
Borland C++ 5.5 for Win32 Copyright (c) 1993, 2000 Borland
calcerr1.cpp:
Error E2451 calcerr1.cpp 20: Undefined symbol 'op' in function main()
*** 1 errors in Compile ***
```

الخطأ موصوف فى السطر 20 للكود . دعنا نلقى نظرة على السطر 20 .

```

#include <iostream.h>
void main ()
{
    float num1;
    float num2;
    char op1;
    float ans;
    cout << "Please enter a number: ";
    cin >> num1;
    cout << "Please enter another number: ";
    cin >> num2;
    cout << "Press A to add the two numbers."
        << endl
        << "Press S to subtract the two numbers."
        << endl
        << "Press M to multiply the two numbers."
        << endl
        << "Press D to divide the two numbers."
        << endl;
    cin >> op;
    if (op == 65)
        ans = num1 + num2;
    if (op == 83)
        ans = num1 - num2;
    if (op == 77)
        ans = num1 * num2;
    if (op == 68)
        ans = num1 / num2;
    cout << "The answer is " << ans << endl;
}

```

كما تذكر , هذا السطر يستخدم لأخذ مدخلات من لوحة المفاتيح ويعطي ما يأخذه لمتغير . في هذه الحالة يسمى `op` .

هذه الجملة تبدو كاملة , ولكن المفسر يدعى أن `op` رمز غير معرف . هل اعلنا عن هذا المتغير سابقاً في هذا الكود ؟ دعنا نلقي نظرة :

```

#include <iostream.h>
void main {}
{
    float num1;
    float num2;
    char op1;
    float ans;
    ...
}

```

هناك مشكلة . لقد اعلنا عن متغير واحد (`op1`) وإستخدمنا آخر (`op`) . هذه مشكلة تحدث غالباً لأنه من الصعب ان تتقفي أثر وتعقب مختلف أسماء المتغيرات المستخدمة في الكود . لحسن الحظ , يلتقطه المفسر . وإذا فهمت المشكلة , فمن السهل إصلاحه .

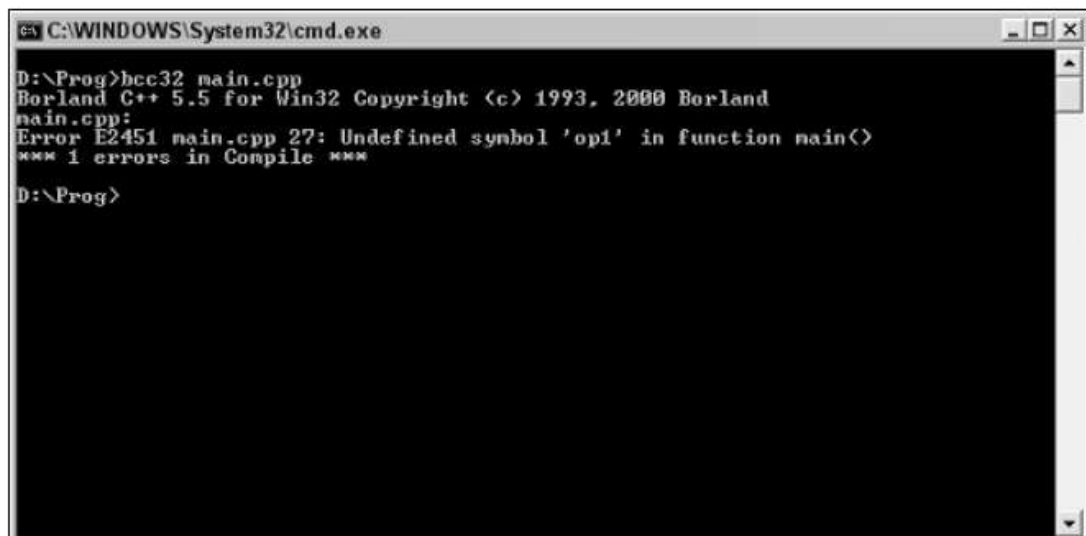
الكود الذى يلى يحتوى على خطأ مشابه :

```

#include <iostream.h>
void main ()
{
    float num1;
    float num2;
    char op;
    float ans;
    cout << "Please enter a number: ";
    cin >> num1;
    cout << "Please enter another number: ";
    cin >> num2;
    cout << "Press A to add the two numbers."
    << endl
    << "Press S to subtract the two numbers."
    << endl
    << "Press M to multiply the two numbers."
    << endl
    << "Press D to divide the two numbers."
    << endl;
    cin >> op;
    if (op == 65)
        ans = num1 + num2;
    if (op == 83)
        ans = num1 - num2;
    if (op == 77)
        ans = num1 * num2;
    if (op1 == 68)
        ans = num1 / num2;
    cout << "The answer is " << ans << endl;
}

```

فإذا حاولت ان تفسر هذا الكود تحصل على العطل التالي (انظر الشكل)



```

C:\WINDOWS\System32\cmd.exe
D:\Prog>bcc32 main.cpp
Borland C++ 5.5 for Win32 Copyright (c) 1993, 2000 Borland
main.cpp:
Error E2451 main.cpp 27: Undefined symbol 'op1' in function main()
*** 1 errors in Compile ***
D:\Prog>

```

```

D:\Prog>bcc32 calcerr2.cpp
Borland C++ 5.5 for Win32 Copyright (c) 1993, 2000 Borland
calcerr2.cpp:
Error E2451 calcerr2.cpp 27: Undefined symbol 'op1' in function main()
*** 1 errors in Compile ***

```

الإن السطر 27 محدد كسطر يحتوى على مشكلة . كما ترى , ما حدث هو أننا أعلننا عن المتغير op ولكن داخل الجملة فقد إستخدمنا المتغير op1 بالخطأ .

تفسير الكود هي طريقة عظيمة لتكتشف من الذى يستضيف الأخطاء فى الكود .

ولكن ليس فقط الأخطاء التى تنتج عن المفسر . فيمكن أن ينتج عنه تحذيرات .

إلقى نظرة على الكود المعدل

```
#include <iostream.h>
void main ()
{
    float num1;
    float num2;
    char op;
    int var1;
    float ans;
    var1 = 7;
    cout << "Please enter a number: ";
    cin >> num1;
    cout << "Please enter another number: ";
    cin >> num2;
    cout << "Press A to add the two numbers."
    << endl
    << "Press S to subtract the two numbers."
    << endl
    << "Press M to multiply the two numbers."
    << endl

    << "Press D to divide the two numbers."
    << endl;
    cin >> op;
    if (op == 65)
        ans = num1 + num2;
    if (op == 83)
        ans = num1 - num2;
    if (op == 77)
        ans = num1 * num2;
    if (op == 68)
        ans = num1 / num2;
    cout << "The answer is " << ans << endl;
}
```

فى هذا الكود قد اعلنا عن متغير يسمى var1 ثم أسندنا له قيمة (يمكن أن يرى السطر 7)

مع ذلك , إذا نظرت خلال الكود ستجد ان لم يستخدم فى أى مكان فى التطبيق . يمكن أن يختار المفسر ليتجاهله ولكن بدلاً من أعطائك تحذير عن المشكلة (إنظر الشكل)



```
C:\WINDOWS\System32\cmd.exe
D:\Prog>bcc32 main.cpp
Borland C++ 5.5 for Win32 Copyright (c) 1993, 2000 Borland
main.cpp:
Warning W8004 main.cpp 32: 'var1' is assigned a value that is never used in func
tion main()
Turbo Incremental Link 5.00 Copyright (c) 1997, 2000 Borland
D:\Prog>
```

```
D:\Prog>bcc32 calcerr3.cpp
Borland C++ 5.5 for Win32 Copyright (c) 1993, 2000 Borland
calcerr3.cpp:
Warning W8004 calcerr3.cpp 32: 'var1' is assigned a value that is never used in
function main()
Turbo Incremental Link 5.00 Copyright (c) 1997, 2000 Borland
```

الإختلاف الآخر بين الخطأ والتحذير انه عندما يلقي المفسر خطأ فإنه يوقف التفسير وينتج عنه تحذير . أما مع التحذيرات , لا يوقف المفسر التفسير ويظل يبني الملف للتنفيذ بطريقة عادية . على الرغم من انه يحتوى على متغير غير مستخدم ولا قيمة مسنده له, ويعمل التطبيق بطريقة عادية (يمكن ان نراه فى الشكل)

```
C:\WINDOWS\System32\cmd.exe
D:\Prog>bcc32 main.cpp
Borland C++ 5.5 for Win32 Copyright (c) 1993, 2000 Borland
main.cpp:
Warning W8004 main.cpp 32: 'var1' is assigned a value that is never used in func
tion main()
Turbo Incremental Link 5.00 Copyright (c) 1997, 2000 Borland

D:\Prog>main.exe
Please enter a number: 5
Please enter another number: 8
Press A to add the two numbers
Press S to subtract the two numbers.
Press M to multiply the two numbers.
Press D to divide the two numbers.
D
The answer is 0.625
D:\Prog>
```

تستطيع , إذا أردت , أن تختار ان تتجاهل التحذيرات , ولكن إهتماماً بجودة الكود ففكرة جيدة أن تصحح المشكلات .

مع ذلك , إذا كنت تقوم بتفسير مؤقت للكود وستستخدم المتغير فيما بعد , يمكن أن تتجاهل التحذير .

اخطاء وتحذيرات المفسر هي خطك الأول للدفاع ضد الاخطاء الحاصلة فى الكود وتسبب المشاكل .

Runtime Errors

أخطاء وقت التشغيل

هي أخطاء تظهر فى برنامجك عندما تقوم بتشغيله . مع تطبيق قمت بتفسيره , كتطبيق C++ , اخطاء وقت التشغيل فى العام تعنى انه عليك أن تعود للوراء إلى مصدر الكود لتكتشف ما هي المشكلة . مع اللغات التي تترجم باستخدام Interpreter , الخطأ من هذا النوع يعنى ان عليك أن تعود للوراء وتفحص الكود .

عندما تستخدم لغة كلغة مترجمة فى مقابل لغة مفسره , الأخطاء التي عادةً سيتم إنقائها وتسلط الضوء عليها بواسطة المفسر لن تظهر حتى يتم تشغيل الكود وعندها تصبح أخطاء وقت التشغيل .

هذه الأنواع من المشاكل التي تتسبب فى أخطاء وقت التشغيل فى الكود مشابهة للتي تتسبب فى اخطاء المفسر . ومع ذلك , الكثير من اللغات التي لا تتطلب مفسر لا تراعى هذه الامور كإعلان المتغيرات ونادراً ما تظهر أن هناك متغيرات غير مستخدمه .

إليك بعض الكود فى VBScript الذي يعالج المتغيرات بشكل إعتباطى للغاية حتى أنه يعمل بدون عرض اخطاء :


```

dim x, y, z
x = "Hello, "
y = "Hello, World!"
zz = "World!"
msgbox(x + zz)

```

هنا ثلاثة متغيرات (x,y,z) تم إعلانهم وإعطائهم قيمة ابتدائية , ولكن الثلاثة لهم قيم ليست نفس الشيء (x,y, zz تم تهيئتهم) . بعدها نستمر فقط في استخدام x و zz . المتغير zz لم يعلن عنه في البداية ولكن مع ذلك لا يزال من الممكن استخدامه والكود يعمل بشكل جيد (كما ترون بالشكل)



هذا الكود لا ينتج عنه خطأ , لا يمكك المتغير z بأى قيمة , ولكن مازال يستخدم على الرغم من ذلك (مخرجات الشاشة تظهر فى الشكل)

```

dim x, y, z
x = "Hello, "
y = "Hello, World!"
zz = "World!"
msgbox(x + z)

```



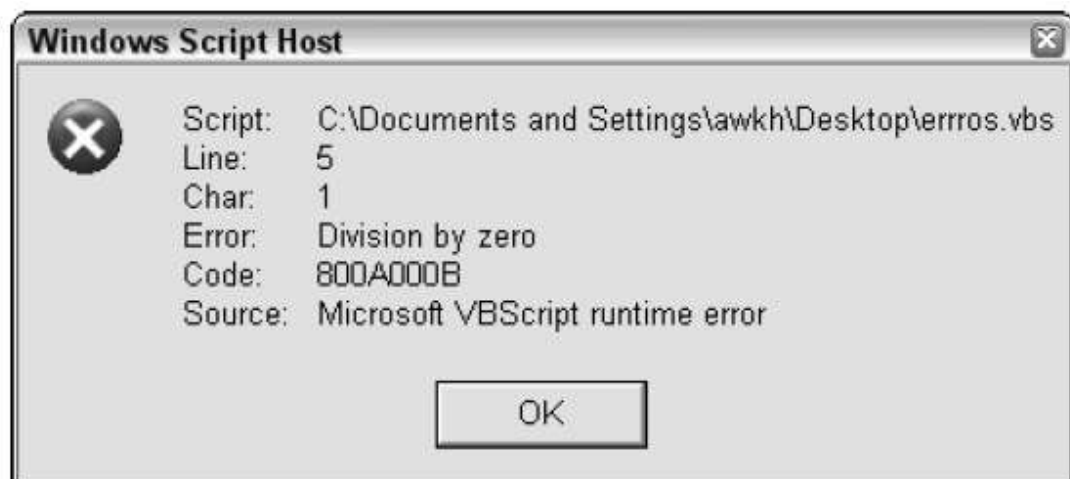
كما ترى , VBScript متسامحة جداً فى الأخطاء بالمقارنة بـ c++ . ولكن هذا لا يعنى انك لن تحصل على اى اخطاء . حاول تشغيل هذا الكود

```

dim x, y, z
x = 1
y = 2
zz = 3
msgbox(zz / z)

```

عند تشغيل هذا الكود سيظهر خطأ وقت التشغيل , كما ترى فى الشكل

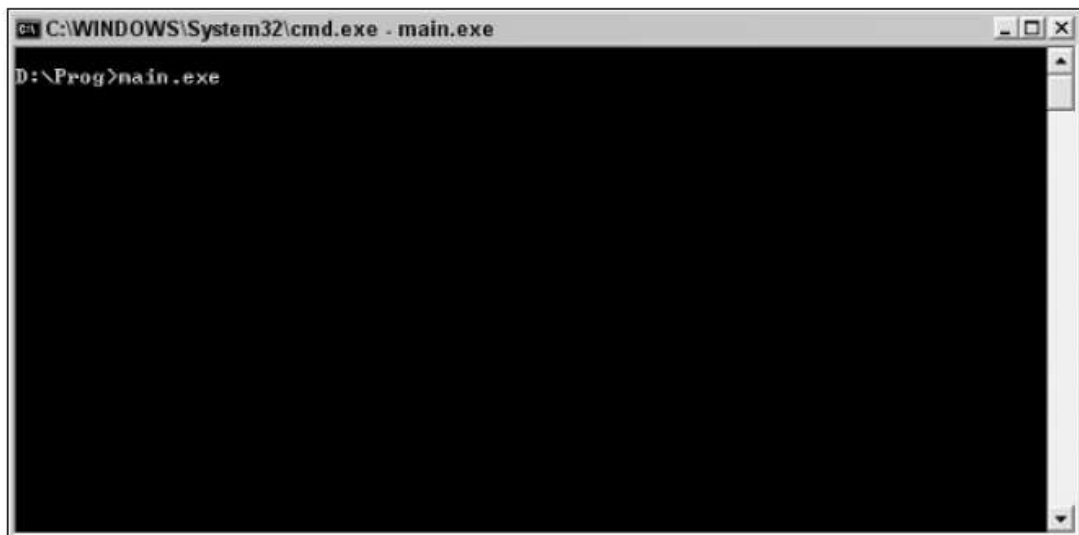


يحدث هذا الكود بسبب اننا إستخدمنا متغير ليس له قيمة معرفة . أى يحتوى على قيمة فارغة null. ولكن لإننا نحاول أن ننفذ عملية رياضية معه , فهي تهمل كأنها صفر .

إلقى نظرة على هذا الكود C++

```
#include <iostream.h>
void main ()
{
    float num1;
    float num2;
    float ans;
    cin >> num1;
    cin >> num2;
    ans = num1 / num2;
    cout << "The answer is " << ans << endl;
}
```

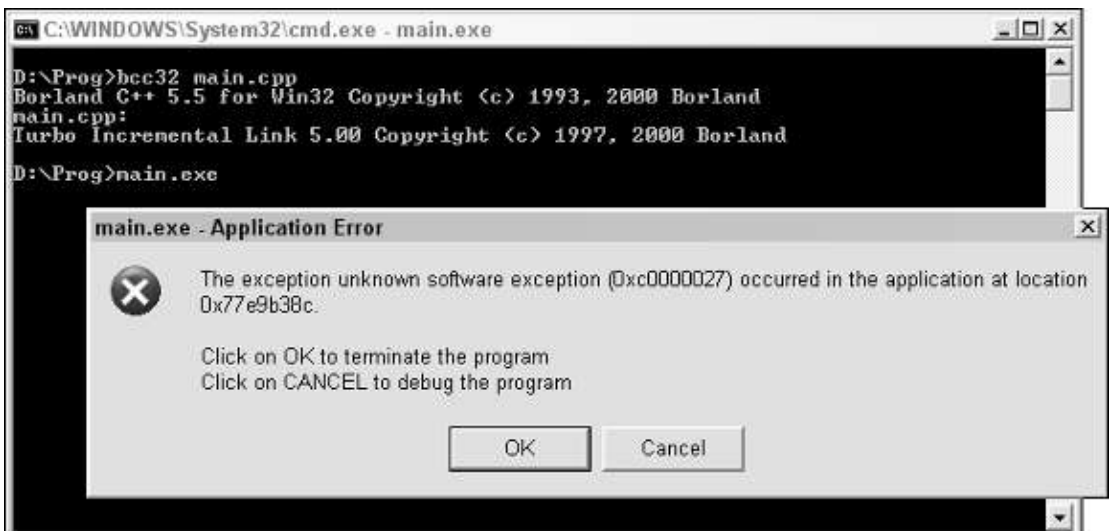
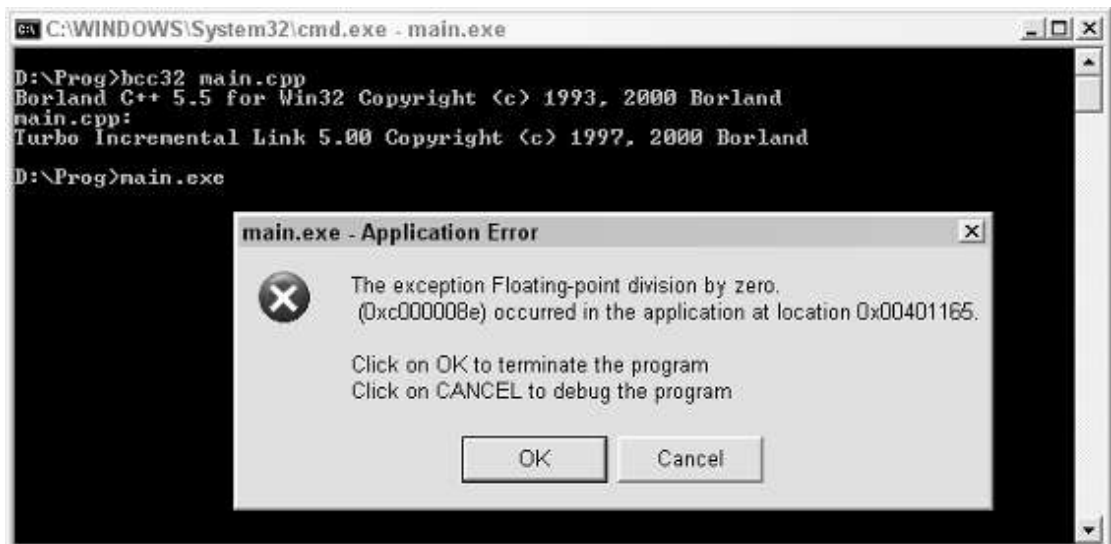
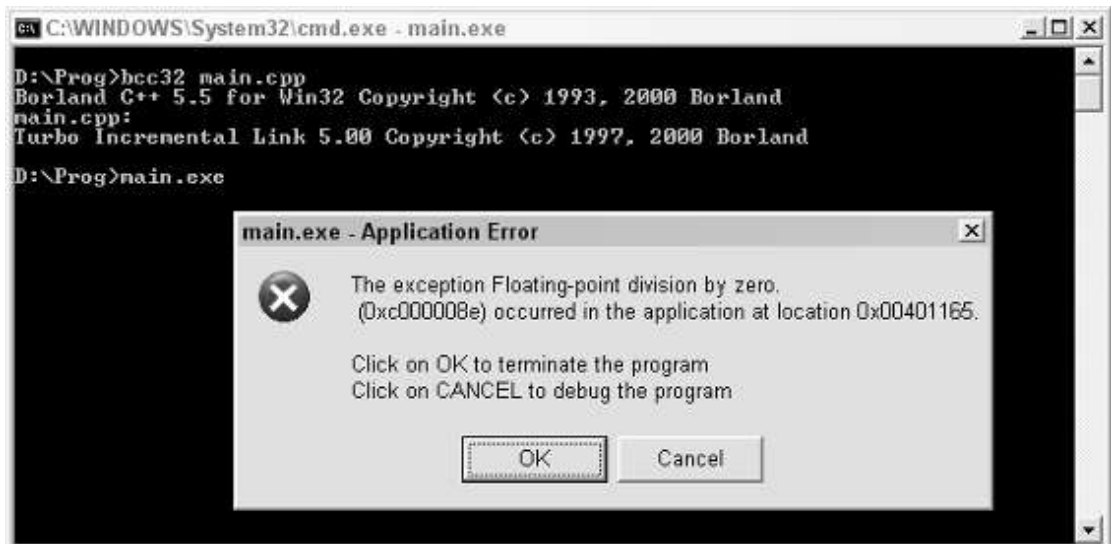
يمكنك ان تفسر هذا الكود . عندما تقوم بتشغيل هذا البرنامج فهو يأخذ رقمين ويقسمهم . فستجد أن كل شئ يعمل جيداً إلى ان تحاول أن تجرى عملية القسمة على الصفر (إنظر الشكل)



ماذا عن هذا الكود , حيث تم معرفة القيم مسبقاً , ولا مفر من أن الكود فى نهاية المطاف سيحاول أن ينفذ القسمة على الصفر . هل يمسك المفسر هذا ؟

```
#include <iostream.h>
void main ()
{
    float num1;
    float num2;
    float ans;
    num1 = 3;
    num2 = 0;
    ans = num1 / num2;
    cout << "The answer is " << ans << endl;
}
```

سيقوم برنامج المفسر بتفسير هذا الكود بدون مشاكل ولكن إذا حاولت أن تقوم بتشغيل البرنامج سينتج عن ذلك خطأ وقت التشغيل , كما يظهر فى الأشكال .



ربما بدأت تلاحظ شيء ما شيق — الخطأ الذي ينتج عن الكود المترجم interpreted في وقت التشغيل يظهر معلومات أكثر عن الخطأ من الذي ينتج عن التطبيق المفسر Compiled في وقت التشغيل.

فتجد من حسن الحظ أن مع الكود المترجم تحصل على رقم السطر الذى نتج عنه الخطأ . وهذا السطر ربما لا يأخذك فى كثير من الأحيان إلى الخطأ مباشرةً (كرقم السطر الذى يعطيه المفسر) ولكن يعطيك مكان تبدأ منه البحث عن الأخطاء .

ربما لاحظت الآن كم الفائدة التى تعود عليك عندما تكون قادراً على الانتقال إلى السطر الكود بواسطة معارفك لرقمه . وهذه هى احد المزايا التى لا توجد فى برنامج محرر النصوص قليل المزايا مثال windows Notepad . وهذا هو السبب الرئيسى وراء وصيتى لك أن تستخدم محرر نصوص قد تم تصميمه للمبرمجين ولديه مزايا مفيدة , من بين أمور أخرى , متاحة .

تعقب الأخطاء وقت التشغيل فى الكود الذى تم تفسيره امر صعب فى حين ان التطبيق يعطيك عنوان الذاكرة ليس أكثر . و يمكن ان يكون التصحيح إستهلاك وقت وربما تجده أمر صعب . فنجد أن أدوات التطوير الحديثة تجعل الحياة أسهل . (الإصدار المتكامل من برنامج Borland يسمى أيضاً C++ builder , يأتى مع ادوات التصحيح المتطورة .مع ذلك , بسبب إرتفاع ثمن البرنامج وعمق التغطية . فهو خارج المجال للمناقشة هنا)

مع ذلك سنلقى نظرة على المخططات schemes لتساعدك فى التعامل مع الأخطاء فيما بعد فى هذا الكتاب

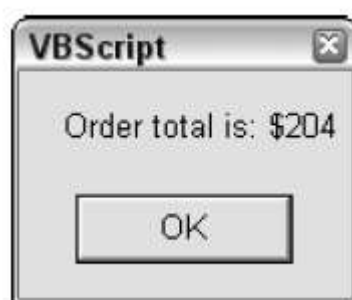
Logic Errors

الأخطاء المنطقية من بين أصعب الأخطاء التى لا يمكن ان تكتشفها ببساطة لأنها اخطاء ليست لها تأثير على الكود فى وقت التشغيل أو وقت التفسير , ولكنها اخطاء تنتج من كيفية عمل الكود .فعادةً , تريد أن يقوم الكود بفعل شئ واحد , ولكن كتبت شئ ما آخر بالخطأ .

مثال بسيط : لديك تطبيق من المفترض أن يضيف تكلفة الشحن إلى الأمر المطلوب . ربما يبدو الكود شئ ما كهذا (من الأبسط أن تجعله أسهل للتتبع)

```
Dim subtotal, shipping, ordertotal
shipping = 5
subtotal = 199
ordertotal = subtotal + shipping
msgbox("Order total is: $" & ordertotal)
```

مخرجات تشغيل هذا الكود تظهر فى الشكل



فهذا يعمل كما تتوقع . لقد أضفت تكلفة الشحن إلى إجمالى امر البيع . هذا النوع "واضح" من الرياضيات هل أنت مستعد لخطأ منطقي . إليك احدهم .

```
Dim subtotal, shipping, ordertotal
shipping = 5
subtotal = 199
ordertotal = subtotal - shipping
msgbox("Order total is: $" & ordertotal)
```

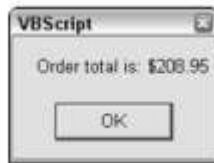
مازال الكود يعمل على الإجمالى , ولكن الإجابة ليست ما تتوقعه على كل حال (إنظر الشكل)



ربما تعتقد ان هذا النوع من الأخطاء يكون واضح جداً لدرجة أن لا أحد سيرتكبه بدون إكتشافه . حسناً , انت مخطئ . التاريخ ملئ بالقصص لمثل هذه الأخطاء . مؤخراً . في المملكة المتحدة عند ظهور اليورو كعملة جديدة ظهر معها قصص كثيرة لأناس قد يريدون أن تتم برمجة تسجيل أموالهم ليقبل اليورو جنباً إلى جنب مع الجنية البريطاني . فقد قاموا ببناء معدل صرف إفتراضى لعملية التحويل . وهذه هى عملية حسابية بسيطة حيث تأخذ الإجمالى فى عملة واحدة وتقوم بتحويله إلى أخرى . مع ذلك , المسجلين الذين اعدوا البرمجة لبعض سلسلة المحلات التى تحتوى على ثغرات خطيرة وهذا يعنى أن التحويلات ستنفذ بشكل خاطئ ويتم دفع العملاء باليورو لمشترياتهم مبلغاً أقل بكثير مما ينبغى عليهم أن يدفعوه .

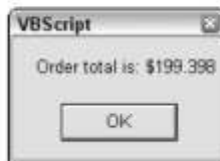
وإليك بعض الكود الذى يقوم بحساب ضريبة مبيعات فى 7 بالمائة (المخرجات تظهر فى الشكل)

```
Dim subtotal, salestax, ordertotal
salestax = 5
subtotal = 199
ordertotal = ((subtotal / 100) * salestax) + subtotal
msgbox("Order total is: $" & ordertotal)
```



هذا النوع من الكود معرض لإخطاء رياضية , وإليك بعض ما يكلفك المال فى المبيعات المفقودة .

```
Dim subtotal, salestax, ordertotal
salestax = 5
subtotal = 199
ordertotal = ((subtotal / 100) / salestax) + subtotal
msgbox("Order total is: $" & ordertotal)
```



والبعض الآخر يكلف العملاء (انظر الشكل)

```
Dim subtotal, salestax, ordertotal
salestax = 5
subtotal = 199
ordertotal = ((subtotal * 100) * salestax) + subtotal
msgbox("Order total is: $" & ordertotal)
```



هل لاحظت شيئاً من نتائج الأمثلة الثلاثة ؟ لاحظ كيف يعطى المثالين فى البداية الجواب فى النطاق الذى تتوقع تقريباً، وربما لا يلقى إنتباه للخطأ. بينما الثالث يعطى جواب شنيع مما يدل على ان هناك خطأ , كقاعدة , كلما زادت الأخطاء المنطقية الفظيعة , كلما برزت الأخطاء وكلما كان إكتشافها بسهولة .

نوع اخر من الأخطاء المنطقية هى حلقات التكرار اللانهاية , وقد رايتها من قبل فى الفصل السابع

```
#include <iostream.h>
void main()
{
    int counter;

    for (counter = 20; counter <= 20; counter--)
    {
        cout << "Going loopy " << counter << endl;
    }
}
```

فستستخدم حلقات التكرار لتكرار جملة او عدة جمل إلى أن تقابل شرط , فنجد فى التكرار اللانهاية , ان الشرط لن يتم مقابلته لأنه لن يتحقق . و سيستمر البرنامج فى عملية تكرار الجمل حتى يتم كسر العملية أو ان يتم إنهاء البرنامج . على الجانب الآخر , سيعمل التكرار بلا نهاية , وليس هناك حاجة لمثل هذا التكرار . ومع ذلك , فان حلقات التكرار الانهائية مشكلة واضحة . فستكون فى العادة من السهل إكتشافهم .

الأوقات التى يكون من الصعب فيها إكتشاف الحلقات اللانهاية عند تحكمهم فى بعض الدوال التى ليس تظهر بوضوح امامك على الشاشة . وحلقات التكرار هذه ستستهلك مساحة الذاكرة وتضعف قوة المعالج ولن يكون من السهل إكتشافهم .

وأرى أنه رغم بساطة حلقات التكرار التى تضيفها إلى الكود لا مانع من أن تفحصه جيداً لتتأكد انه يعمل بشكل صحيح وأنها خالية من إمكانية حدوث التكرار اللانهاية .

مشكلة منطقية شائعة أخرى , خاصة فى لغة ك++ , وتحدث عندما تستخدم معامل خاطئ. لنأخذ المعامل = و == . فلهما معانى مختلفة (= يمكن ان يستخدم لإسناد قيمة لمتغير بينما == تستخدم كمعامل مقارنة لفحص إذا كان المتغير مساوياً لقيمة أخرى) .

تجد بالأسفل كود الحاسبة الذى كان لدينا مسبقاً وبه إثنين من هذه الأخطاء . هل يمكن أن تكتشفهم ؟

```
#include <iostream.h>
void main ()
{
    float num1;
    float num2;
    char op;
    float ans;
    cout << "Please enter a number: ";
    cin >> num1;
    cout << "Please enter another number: ";
    cin >> num2;
    cout << "Press A to add the two numbers."
        << endl
        << "Press S to subtract the two numbers."
        << endl
        << "Press M to multiply the two numbers."
        << endl
        << "Press D to divide the two numbers."
        << endl;
    cin >> op;
    if (op == 65)
        ans = num1 + num2;
    if (op == 83)
        ans = num1 - num2;
    if (op == 77)
        ans = num1 * num2;
    if (op == 68)
        ans = num1 / num2;
    cout << "The answer is " << ans << endl;
}
```

الأخطاء التي فى السطر 23 و 28 :

```
cin >> op;
if (op == 65)
    ans = num1 + num2;
if (op = 83)
    ans = num1 - num2;
if (op == 77)
    ans == num1 * num2;
if (op == 68)
    ans = num1 / num2;
cout << "The answer is " << ans << endl;
}
```

هل إكتشفتهم ؟ ليس أمراً سهلاً , أليس كذلك ؟

الطرق المؤثرة الوحيدة التى تمنع الكود الذى يحتوى على أخطاء منطقية من أن تخرج للعالم الواسع هى من خلال تحليل الكود وبحرص , وشمولية الإختبار للكود , وإستخدام مدخلات متنوعة وفحصهم فى العمل بشكل إستقلالى للمخرجات . لذا , مع حساب تكاليف الشحن , سيتعامل مع الشحن المناسب لمجموعة من المجاميع الفرعية وإدخالهم , وقم بفحص المخرجات لتتأكد من أن انها أكثر دقة .

Spotting Errors إكتشاف الأخطاء

إكتشاف الأخطاء عند كتابة الكود هى أحد المهارات المفيدة للغاية لدى المبرمج .

دعنا نلقى نظرة على بعض المخططات التى تجعل إكتشاف الأخطاء سهلة , قبل أن تنتقل إلى وقت المفسر أو وقت التشغيل .

Read Each Line After You Press Enter إقرأ كل سطر بعد ان تضغط enter

يبدو واضح , ولكن أنا مندهش كيف أن كثيراً من الناس لا يفعلون هذا.

فهى فقط تأخذ لحظات , وكل ما تفعله هو اخذ نظرة سريعة على الجملة فحسب . فإسال نفسك :

- هل يبدو الكود صحيح ؟
- هل تكتشف أخطاء مطبعية او أى أخطاء اخرى ؟
- هل ينجز الكود ما قمت بإنشائه من أجله ؟
- هل إستخدمت علامات نهاية السطر المناسبة ؟

كلما كنت جاداً مع الكود , ستجد أنك ذهنيأ تترجم الكود الذى كتبته فى الإنجليزية كما تقرأه . وهذه طريقة عظيمة لإكتشاف الأخطاء لأنه ربما يكون خطأ الكود غير واضح , ولكن عندما تترجمه إلى الإنجليزية فربما يظهر لك الكثير من المشكلات .

Check the Preceding Statements

إفحص الجمل السابقة

هل لهم علاقة مع ما تحاول فعله بالكود , أو بهم تأثير أو تغيير لمجرى الكود ؟ قم بقراءة الجمل السابقة لتذكير نفسك بما تحاول إنجازه مع الكود .

وأرى ان تنفذ هذا النوع من المراجعة على الأقل بعد كتابة بعض الجمل. ولكن إن امكن وهذا من الجيد إن تقوم بالمراجعة مبدئياً بعد كل جملة لأن هذا يساعدكفى كتابة كود ينقلك للإمام بالحل المكتمل فيما بعد .

Keep the Layout Clear

إبقى التخطيط واضح

تذكر أن تبقى تخطيط الكود تحت التحكم . تأكد انك تستخدم الأقواس المتعرجة الصحيحة والأقواس فى الكود

```
void functionname()
{
    // Code goes here.
}
```

ربما تجد من السهل أن تحفظ هذا قالب الكود جاهز فى متناول اليد لذا يمكنك ان تنسخه وتلصقه للمكان الذى تريده .

عدد الساعات التى يمكن ان يضيعها الناس فى البحث عن الخطأ الذى له علاقة بما تفعله فى قالب الكود مذهل , فتأكد عند إنشاء دالة جديدة أن تضيف إسم الدالة وأن تفتح وتغلق الأقواس المتعرجة بمجرد بدء تشغيل الدالة.

عند فتح قوس يجب إغلاقه مباشرةً ثم كتابة الكود بينهما ولا تعتمد على أنك ستذكر أن تغلقه بعد كتابة محتوى الدالة .

Comments, Comments, Comments!

تعليقات , تعليقات , تعليقات

تذكر عند إضافة تعليقات ان تستخدم علامات التعليق المناسبة للغة التى تستخدمها .

لا تفعل ما يلى :

- استخدام علامات تعليقات خاطئة .
- تنسى أن تضيف علامات التعليق .
- تنسى أن تضيف علامات التعليق للأسطر اللاحقة (استخدم التعليقات المتعددة لو كان مناسباً)

إذا قمت بتحويل سطر من الكود إلى تعليق لتستبدله بآخر أو فقط للتخلص منه . تذكر ان تضيف تفصيل التعليق فى لماذا قمت بحذفه ومتى — فكرياً جداً ستتعب وتتسائل لماذا تم حذف هذا السطر من الكود.

Remove Ambiguity in Code

أَمْحِ الغموض من الكود

أحسبك تذكر في الفصل السابع كيف قلت لك ان المثالين التاليين من الكود فنياً هم واحد وينتج عنهما نفس النتائج :

Code 1:

```
#include <iostream.h>
void main()
{
    int x;
    cout << "Pick an integer: ";
    cin >> x;
    if (x == 7)
    {
        cout << "You win the C++ lottery!" << endl;
        cout << "Thank you for playing the C++ lottery" << endl;
    }
    else
    {
        cout << "Thank you for playing the C++ lottery" << endl;
    }
}
```

Code 2:

```
#include <iostream.h>
void main()
{
    int x;
    cout << "Pick an integer: ";
    cin >> x;
    if (x == 7)
        cout << "You win the C++ lottery!" << endl;
        cout << "Thank you for playing the C++ lottery" << endl;
}
```

فمن ناحية فنية , كلا المثالين معرف في دالة ولكن محدودة الإستخدام للأقواس المتعرجة مع غياب `else` في المثال الثاني مما يجعله أكثر تعقيداً لأن يقرأ او يتم متابعته .

في المجمل , إذا كان أن تختار بين إيجاز الكود و إزالة الغموض , دوماً إختار إزالة الغموض — في حين أنى أعرف ما يفعله الكود , وربما شخص آخر يطلع عليه وربما لا . وحيث انك تتعلم فأقترح عليك بشدة أن تحذو حذوى .

Semicolons

[يقصد بها العلامات بهذا الشكل ; التى تكتب فى آخر كل جملة]

هدفها إنهاء السطر للجملة . وليس للتجميل أو لتزين الكود .

فتكون بعد إتمام الجمل . كن حريص جداً عندما تضيفهم حول حلقات التكرار .

```
for(x = 0; x < 100; x++);
{
    cout << "Hello, World!" << endl;
}
```

هذه الكتلة من الكود بها علامتين ; semicolon . التى فى السطر الثالث هى الجيدة .

```
for(x = 0; x < 100; x++);  
{  
    cout << "Hello, World!" << endl;  
}
```

أما التى فى السطر الأول وفقد وضعت خطأً . وإذا أبقيت عليها فى مكانها التى هى فيه الآن فبذلك سيتكرر الكود مرة واحدة فقط .

```
for(x = 0; x < 100; x++);  
{  
    cout << "Hello, World!" << endl;  
}
```

كن على دراية جيدة باستخدام علامات إنهاء السطر فى وقت مبكر من مهنتك البرمجية , وستحفظ لنفسك المزيد من الوقت الضائع والجهد فيما بعد .

Test the Code

إختبر الكود

يبدو ان هذه نقطة أخرى واضحة , وستفاجأ حين تعرف أن الكثير من المبتدئين يكتبون الأسطر والأسطر من الكود قبل إخذ نظرة حقيقية عليه ويرون إن كان سيعمل أما لا .

إن امكنك , قم بتقطيع الكود إلى دوال تعمل بمفردها أو على الأقل يمكنك ان تقوم بتشغيلها بمفردها لتختبرها . وربما يكون عليك أن تصطنع إدخالات لقيم متغيرات وتنشأ مخرجات مصطنعة لتفحص ان كل شئ على ما يرام .

Keep Track of Variables

تتبع المتغيرات

إذا كان لديك كود لديه تعاملات كثيرة مع المتغيرات , إذاً ربما تجد من الجيد ان تتابع بشكل دوري ماذا يعنى كل متغير فى كل نقطة فى الكود .

الطريقة الأسهل لفعل هذا هو إخراجهم فى طريقة واحدة أو بأخرى . عرضهم على الشاشة عادةً ما يكون جيداً للمشاركة البسيطة أو المتوسطة التعقيد . هل تذكر كود الحاسبة فى الفصل السابع الذى يجرى عمليات على الأرقام التى تعتمد على المدخلات ؟ من المفيد مع مثل هذا الكود أن تعرض مخرجات الكود على الشاشة لقيم المتغيرات المستخدمة ليتمكنك فحصهم لترى هل القيم كما تتوقع أم لا .

```
#include <iostream.h>  
void main ()  
{  
    float num1;  
    float num2;  
    char op;  
    float ans;
```

```

cout << "Please enter a number: ";
cin >> num1;
cout << "Please enter another number: ";
cin >> num2;
cout << "Press A to add the two numbers."
    << endl
    << "Press S to subtract the two numbers."
    << endl
    << "Press M to multiply the two numbers."
    << endl
    << "Press D to divide the two numbers."
    << endl;
cin >> op;
cout << "----- start test output -----"
    << endl
    << "num1: " << num1
    << endl
    << "num2: " << num2
    << endl
    << "op: " << op
    << endl
    << "----- end test output -----"
    << endl;
if (op == 65)
    ans = num1 + num2;
if (op == 83)
    ans = num1 - num2;
if (op == 77)
    ans = num1 * num2;
if (op == 68)
    ans = num1 / num2;
cout << "The answer is " << ans << endl;
}

```

مخرجات هذا الكود تظهر في الشكل

```

C:\WINDOWS\System32\cmd.exe
D:\Prog>bcc32 main.cpp
Borland C++ 5.5 for Win32 Copyright (c) 1993, 2000 Borland
main.cpp:
Turbo Incremental Link 5.00 Copyright (c) 1997, 2000 Borland
D:\Prog>main.exe
Please enter a number: 4
Please enter another number: 56
Press A to add the two numbers
Press S to subtract the two numbers.
Press M to multiply the two numbers.
Press D to divide the two numbers.
A
----- start test output -----
num1: 4
num2: 56
op: A
----- end test output -----
The answer is 60
D:\Prog>

```

في كثير من الأحيان نجد من الهام ان نلقى نظرة على المتغيراتبشكل دورى , لأن هذا يساعدك لتكتشف الأخطاء أول بأول . إليك مثال آخر ينتفع من هذه المعالجة :

```

#include <iostream.h>
void main()
{
    int x;
    int y;
    y = 7
    cout << "Pick an integer: ";
    cin >> x;
    if (x == y)
    {
        cout << "You win the C++ lottery!" << endl;
        cout << "Thank you for playing the C++ lottery" << endl;
    }
    else
    {
        cout << "Thank you for playing the C++ lottery" << endl;
    }

    cout << "----- start test output -----"
    << endl
    << "Winning condition: " << y
    << endl
    << "----- end test output -----"
    << endl;
}

```

الشكل التالي يظهر الكود بعد التشغيل

```

C:\WINDOWS\System32\cmd.exe
D:\Prog>main.exe
Pick an integer: 6
Thank you for playing the C++ lottery
----- start test output -----
Winning condition: 7
----- end test output -----
D:\Prog>

```

الشيء الوحيد الذي يمكنك القيام به هو أن يكون هذا النوع من الإخراج معتمداً على قيم المتغير الذي قمنا باختباره . فإذا تم إعداد المتغير إلى قيمة واحدة , فنتنتج مخرجات واحدة , و إذا إعطيته قيمة أخرى .حدث القلق .

```

#include <iostream.h>
void main()
{
    int x;
    int y;
    y = 7

```

```

int outputvariable;
outputvariable = 1;
cout << "Pick an integer: ";
cin >> x;
if (x == y)
{
    cout << "You win the C++ lottery!" << endl;
    cout << "Thank you for playing the C++ lottery" << endl;
}
else
{
    cout << "Thank you for playing the C++ lottery" << endl;
}
}

if (outputvariable == 1)
{
    cout << "----- start test output -----"
    << endl
    << "Winning condition: " << y
    << endl
    << "----- end test output -----"
    << endl;
}
)

```

إذا أردت ان تمنع المخرجات من وصولها للشاشة , فقط قم بتغيير المتغير `outputvariable` لأي شيء آخر غير 1 (يمكنك ان تختار 0 لأن تمثل صفر) .

```

#include <iostream.h>
void main()
{
    int x;
    int y;
    y = 7;
    int outputvariable;
    outputvariable = 0;
    cout << "Pick an integer: ";
    cin >> x;
    if (x == y)
    {
        cout << "You win the C++ lottery!" << endl;
        cout << "Thank you for playing the C++ lottery" << endl;
    }
    else
    {
        cout << "Thank you for playing the C++ lottery" << endl;
    }
    if (outputvariable == 1)
    {
        cout << "----- start test output -----"
        << endl
        << "Winning condition: " << y
        << endl
        << "----- end test output -----"
        << endl;
    }
}

```

هذا التخطيط يمكن أن يستخدم لمخرجات متعددة

```

#include <iostream.h>
void main ()
{
    int outputvariable;
    outputvariable = 1;
    float num1;
    float num2;
    char op;
    float ans;
    cout << "Please enter a number: ";
    cin >> num1;
    cout << "Please enter another number: ";
    cin >> num2;
    cout << "Press A to add the two numbers."
        << endl
        << "Press S to subtract the two numbers."
        << endl
        << "Press M to multiply the two numbers."
        << endl
        << "Press D to divide the two numbers."
        << endl;
    cin >> op;
    if (outputvariable == 1)
    {
        cout << "----- start test output -----"
            << endl
            << "num1: " << num1
            << endl
            << "----- end test output -----"
            << endl;
    }
    if (outputvariable == 1)
    {
        cout << "----- start test output -----"
            << endl
            << "num2: " << num2
            << endl
            << "----- end test output -----"
            << endl;
    }
    if (outputvariable == 1)
    {
        cout << "----- start test output -----"
            << endl
            << "op: " << op
            << endl
            << "----- end test output -----"
            << endl;
    }
    if (op == 65)
        ans = num1 + num2;
    if (op == 83)
        ans = num1 - num2;
    if (op == 77)
        ans = num1 * num2;
    if (op == 68)
        ans = num1 / num2;
    cout << "The answer is " << ans << endl;
}

```

يمكن ان ترى هذا الكود فى الفعل

```

C:\WINDOWS\System32\cmd.exe
D:\Prog>main.exe
Please enter a number: 23
Please enter another number: 33
Press A to add the two numbers
Press S to subtract the two numbers.
Press M to multiply the two numbers.
Press D to divide the two numbers.
A
----- start test output -----
num1: 23
----- end test output -----
----- start test output -----
num2: 33
----- end test output -----
----- start test output -----
op: A
----- end test output -----
The answer is 56
D:\Prog>

```

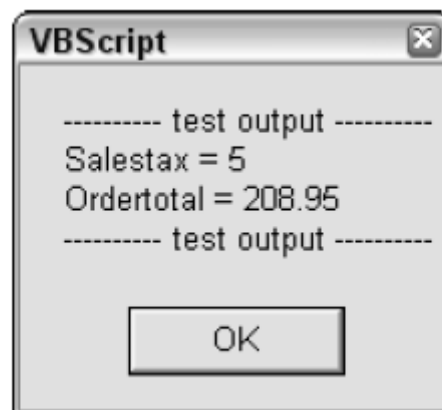
الكود الذى كان لدينا مسبقاً يحسب ضريبة المبيعات الذى إستفنعنا بها من هذا النوع من المعالجة .

```

Dim subtotal, salestax, ordertotal
salestax = 5
subtotal = 199
ordertotal = ((subtotal / 100) * salestax) + subtotal
msgbox("----- test output -----" & chr(13)
& "Salestax = " & salestax & chr(13)
& "Ordertotal = " & ordertotal & chr(13)
& "----- test output -----")
msgbox("Order total is: $" & ordertotal)

```

مخرجات الإختبار تظهر فى الشكل



إنظر كيف إستخدمنا حرف فاصل الاسطر chr(13) لتنسيق صندوق الرسالة . وهذا يجعله أسهل ليقرأ .

Summary

الملخص

فى هذا الفصل , لقد رأينا المزيد من الكود — البعض يعمل والكثير منه به أخطاء . الغرض من هذا الفصل هو إلقاء نظرة على الإخطاء فى البرمجة ويعطيك بعض الخبرة فى معالجة الكود الذى به خطأ .

ربما تتعجب لما نحتاج أن نرى الكود الذى به أخطاء لتكون قادراً على كتابة كود ليقوم بعمل ذلك لك . حسناً , السبب فى إمكانية تحكمك بمثل هذه الاخطاء , يسمح لك برؤية أين مكان الخطأ والإثار الجانبية لهذه الأنواع من الأخطاء .

الشئ الرئيسى لتذكره هنا مع الأخطاء هو أن كل شخص يرتكب أخطاء , والخدعة هى ان تفهم ما السبب ورائهم , ليسهل عليك أن تكتشفهم , وتكون قادراً على التعامل معهم بفاعلية .

أيضاً , خذ الوقت لتقرأ كل وأى رسالة خطأ تظهر لك — أنا أعرف أن لديك إتجاهاً لا إرادى أن تتخلص منهم كلما امكن . وهذا خطأ لأن رسائل الخطأ تحتوى على معلومات هامة !

10

Interface

الواجهة

المبرمج الذى يفكر فقط فى كيفية عمل البرنامج بشكل جيد ليس مبرمج جيد . المبرمج الجيد ليس فقط يحتاج أن يكتب كود جيد يجعل التطبيق يعمل كما هو مطلوب — فالمبرمج أيضاً يحتاج إلى أن يكون قادراً على جعل التطبيق سهل الاستخدام للمستخدم النهائى . وإذا كان برنامجك مشابهة لكثير من البرامج التى تؤدي نفس المهام , فتحتاج أن تقنع المستخدم أن يختار برنامجك من بين البرامج الأخرى .

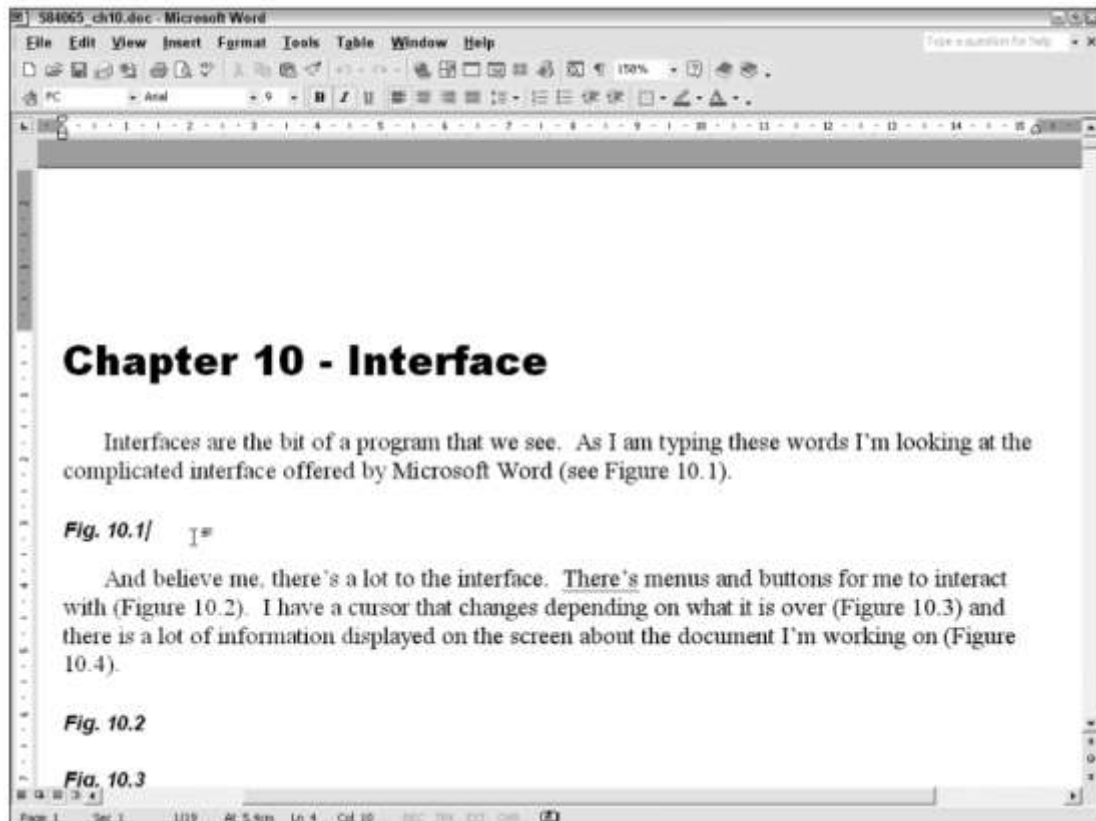
هذا الفصل يلقي نظرة على كيف تنشأ واجهة جيدة وكيف تتجنب الواجهة السيئة ! وهو أيضاً يفحص المكونات الفردية للواجهة والنظر فى كيف يمكنك إحضارهم معاً لإستخدام التطبيق بسهولة وطريقة أكثر منطقية .

What Is an Interface?

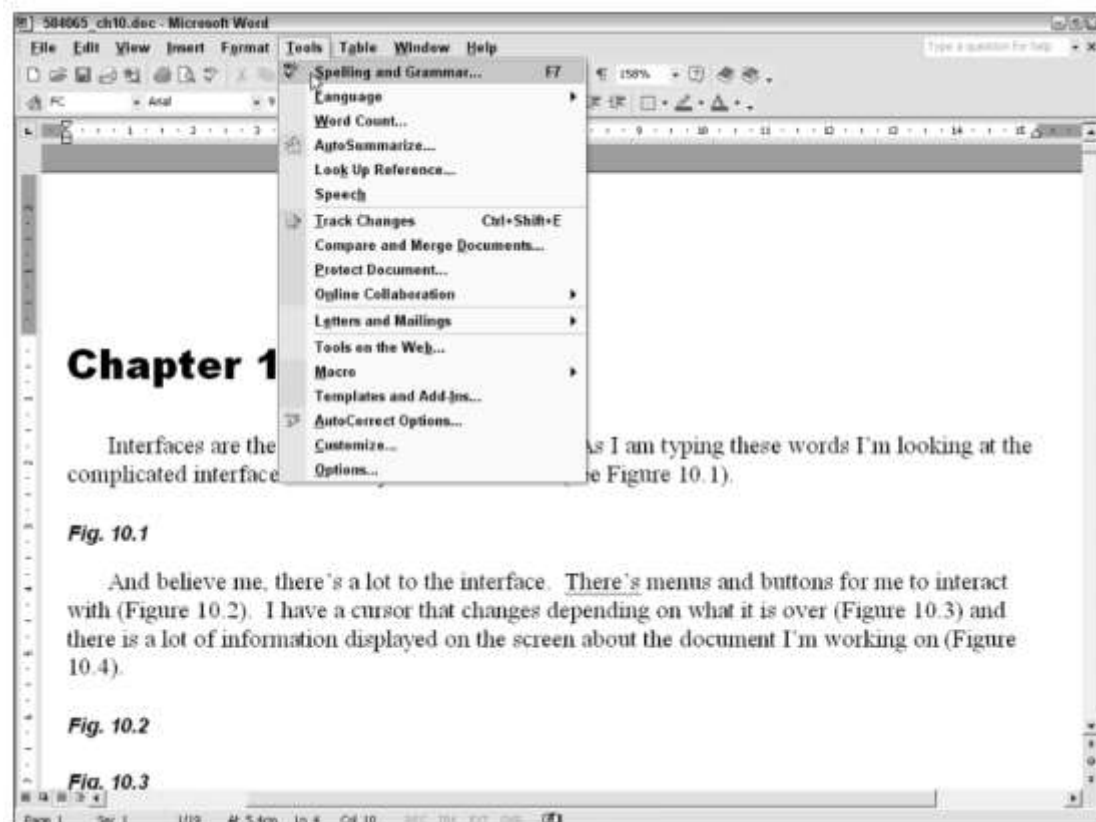
ما هى الواجهة ؟

الواجهات هى جزء من البرنامج الذى تراه . كما أكتب هذه الكلمات فأنا انظر إلى الواجهة المعقدة الذى يقدمها Microsoft Word كما يظهر فى الشكل .

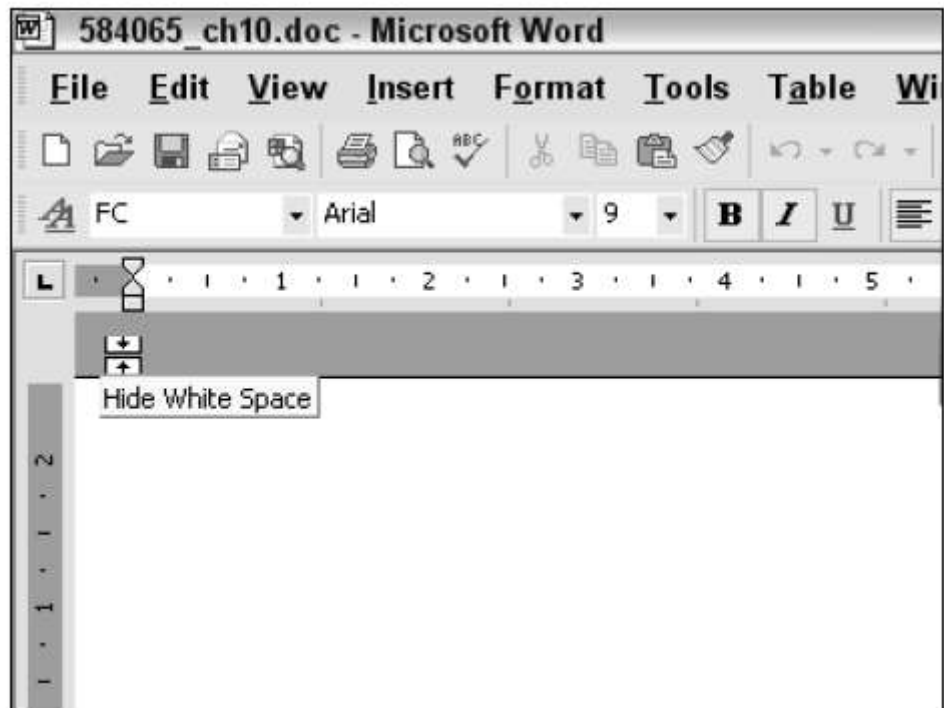
وصدقنى , هناك المزيد الموجود فى الواجهة , هناك قوائم ومفاتيح لى لأتفاعل معها , ويظهر فى الشكل . فلدى مؤشر الذى يتغير اعتماداً على ما هو عليه (انظر الشكل) , وهناك المزيد من المعلومات المعروضة فى الشاشة عن المستند الذى اعمل عليه (انظر الشكل)



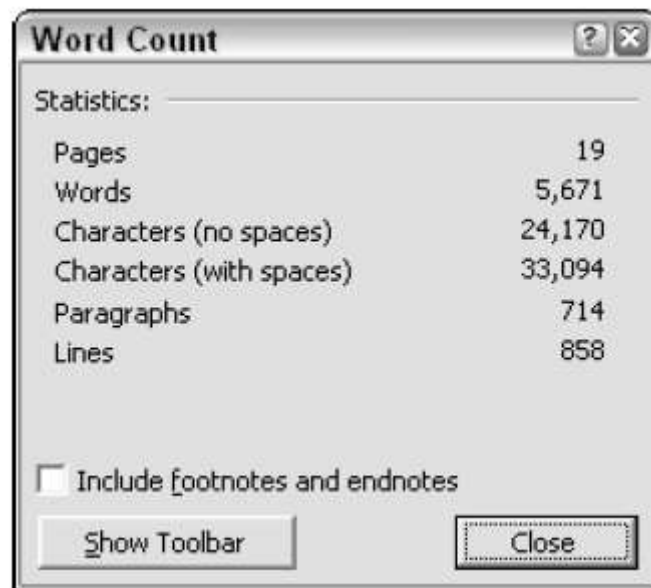
الواجهة



القوائم



المؤشر



معلومات إضافية

الواجهة هي مفتاح للمستخدم تتيح استخدام جيد للبرنامج الذي أنشأته .

The Importance of an Interface

أهمية الواجهة

نحن محاطون بالواجهات في برامج الحاسب التي بالفعل لا نعطيها المزيد من التفكير . فسواء كان في حاسب شخصي . كل تفاعلاتك مع البرنامج هي واجهة باستخدام كلاً من الفأرة ولوحة المفاتيح (وحدات إدخال أخرى موجودة ولكن معظمهم عادةً ما يضاهي لوحات المفاتيح وأجهزة فأرة بالطريقة التي تعمل بها) لتحصل على برنامج يقوم بما تقوم أنت به , فتمرر له التعليمات بهذه الطريقة .

What Is an Interface?

ما هي الواجهة

الواجهة هي جزء من البرنامج الذي يراه المستخدم النهائي على الشاشة عند عمل البرنامج . الشكل التالي يظهر أبسط واجهة مستخدم ممكنة على تطبيق .



كل ما هنالك هو مفتاح واحد . هذا الزر [المفتاح] مميز بوضوح ومن غير المحتمل أن يتم إفتقاده . ومع هذا , فهذه ليست واجهة مستخدم جيدة لإسباب نأخذها في وقت لاحق . ومع ذلك , وليست إلى حد ما أسوأ المحتمل . الشكل التالي يظهر واجهة مستخدم مروعة .



هذه الواجهة تعرض بشكل سيئ ومن الصعب أن تعمل معها لإن العناصر التي تعمل قد عرضت بشكل سيئ .
سنلقى نظرة قريباً على ما يجعل الواجهة جيدة وما يجعلها سيئة .

Does All Software Have an Interface?

هل البرامج لها واجهات ؟

المعظم له , ربما يكون لديك برامج لا تتضمن على واجهة . ولكن في العام الواجهة مفيدة حتى إن كان كل ما تفعله هو تمرير معلومات تأكيدية للفعل الذي قام المستخدم بطلبه كتشغيل البرنامج الذي قام بتنفيذه . أيضاً , إذا كان التطبيق لا يحتاج إلى واجهة , ما هو رد الفعل إذا حدث خطأ ما و لم ينفذ الحدث ؟

إذا كنت تعتقد أن لديك برنامج لا يحتاج إلى نموذج على الشاشة يتفاعل معها المستخدم النهائي , أرى ان تعيد التفكير بحرص في هذا وتعيد فحص السبب في أن البرنامج لا يحتاج إلى واجهة .

Examining the Interface

فحص الواجهة

دعنا نلقى نظرة على الواجهات في الواقع . سنبدأ بأخذ نظرة على الأنواع الابتدائية من الواجهة التي من المحتمل أن تصادفها — وهي واجهة تعتمد على النص .

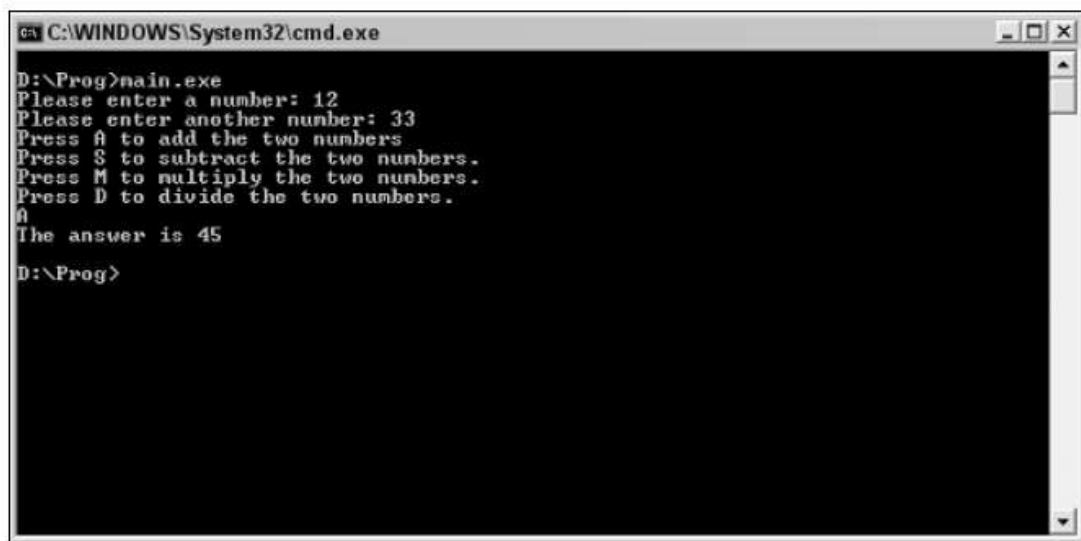
Text-Based Interface

واجهة تعتمد على النص

تذكر تطبيق حاسبة C++ التي قد عملت معها في الفصل السابع "The Structure of Coding" ذلك التطبيق به واجهة متعمدة على نص .
هذا هو الكود الذي يستخدم لقيادة الواجهة .

```
#include <iostream.h>
void main()
{
    float num1;
    float num2;
    char op;
    float ans;
    cout << "Please enter a number: ";
    cin >> num1;
    cout << "Please enter another number: ";
    cin >> num2;
    cout << "Press A to add the two numbers,"
        << endl
        << "Press S to subtract the two numbers."
        << endl
        << "Press M to multiply the two numbers."
        << endl
        << "Press D to divide the two numbers."
        << endl;
    cin >> op;
    if (op == 65)
        ans = num1 + num2;
    if (op == 83)
        ans = num1 - num2;
    if (op == 77)
        ans = num1 * num2;
    if (op == 68)
        ans = num1 / num2;
    cout << "The answer is " << ans << endl;
}
```

عندما يتم تشغيل البرنامج , يعرض التعليمات عن كيفية إستخدام البرنامج (إنظر الشكل) فهي تخبر المستخدم ما ذا يفعل بالرقمين اللذين قام بإدخالهما .

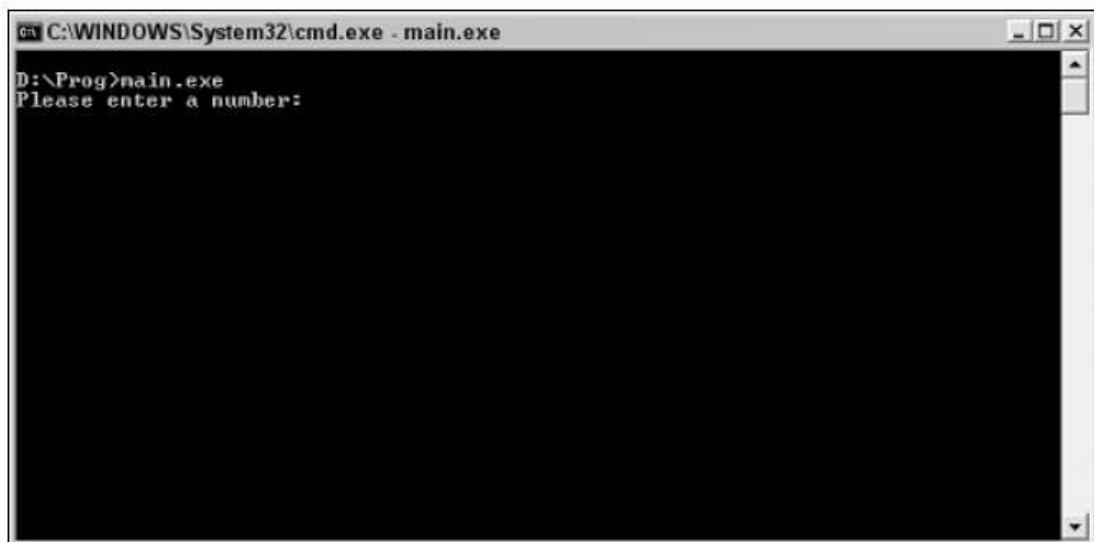


```
C:\WINDOWS\System32\cmd.exe
D:\Prog>nain.exe
Please enter a number: 12
Please enter another number: 33
Press A to add the two numbers
Press S to subtract the two numbers.
Press M to multiply the two numbers.
Press D to divide the two numbers.
A
The answer is 45
D:\Prog>
```

ومع ذلك , فهذه ليست واجهة جيدة , دعنى أشرح لماذا .

لبنبدأ معها , عندما تقوم بتشغيل البرنامج فليس هناك تحديد إبتدائي لما يفعله التطبيق أو ماذا يفترض من المستخدم أن يفعل معه . نحن نعرف كيف يعمل البرنامج لإننا نحن من كتبناه وقد رأينا مصدر الكود ونعرف المتوقع منه , ولكن ضع نفسك فى مكان الشخص الآخر الذى يستخدم هذا البرنامج للمرة الاولى ولا يعرف ما يتوقع من ذلك أو ما يفعله .

ما الذى عليك فعله فيما تضعه كقيمة إبتدائية (إنظر الشكل) ؟



```
C:\WINDOWS\System32\cmd.exe - main.exe
D:\Prog>nain.exe
Please enter a number:
```

لا يعطيك هذا المزيد من المعلومات , الا يفعل ؟

هذا هو الفرق بين أداة وحيدة تقوم بإنشائها لنفسك وتطبيق تريد ان يستخدمه الآخرون .ونحن كمبرمجين لدينا رؤية داخل التطبيق لا يراها الآخريين .فنحن نبرمجه بسبب اننا نعرف كيف يعمل . نحن نعرف المدخلات

المتوقعة وماذا يفعل البرنامج . إمتلاك كل المعلومات لا يجعلنا بمأمن من الأخطاء فى الكود ولكن يعطينا صفقة عظيمة من المزايا عن الآخرين الذين سيستخدمون البرنامج .

بينما تطبيق الحاسبة لا يقدم المزيد من المعلومات للمستخدم النهائي , فليس أسوأ ما يكون . إلقى نظرة على هذا المثال الذى يفعل نفس الشئ ولكن بدون أى تأجيل لإدخال مدخلات .

```
#include <iostream.h>
void main()
{
    float num1;
    float num2;
    char op;
    float ans;
    cin >> num1;
    cin >> num2;
    cin >> op;
    if (op == 65)
        ans = num1 + num2;
    if (op == 83)

        ans = num1 - num2;
    if (op == 77)
        ans = num1 * num2;
    if (op == 68)
        ans = num1 / num2;
    cout << ans << endl;
}
```

هذا الكود , عندما يتم تفسيره , يفعل نفس الشئ الذى يفعله الكود السابق , ولكن هذه المرة ليس هناك نموذج لكيفية إستخدام البرنامج . (إنظر الشكل)



مع الكود فى هذا الشرط , إذا كنت لا تعرف متى تدخل أرقام ومتى تضغط المفتاح الصحيح , وليس لديك فكرة كيف يعمل التطبيق ولا تعرف حتى ماذا تعنى المخرجات . عندها يكون هذا التطبيق فى حقيقة الأمر يترك المستخدم فى الظلام .

لذا , دعنا نلقى نظرة على طرق التحسين على البرنامج ويجعله أسهل للاستخدام وأكثر حداثة .

Program Overview

نظرة على البرنامج

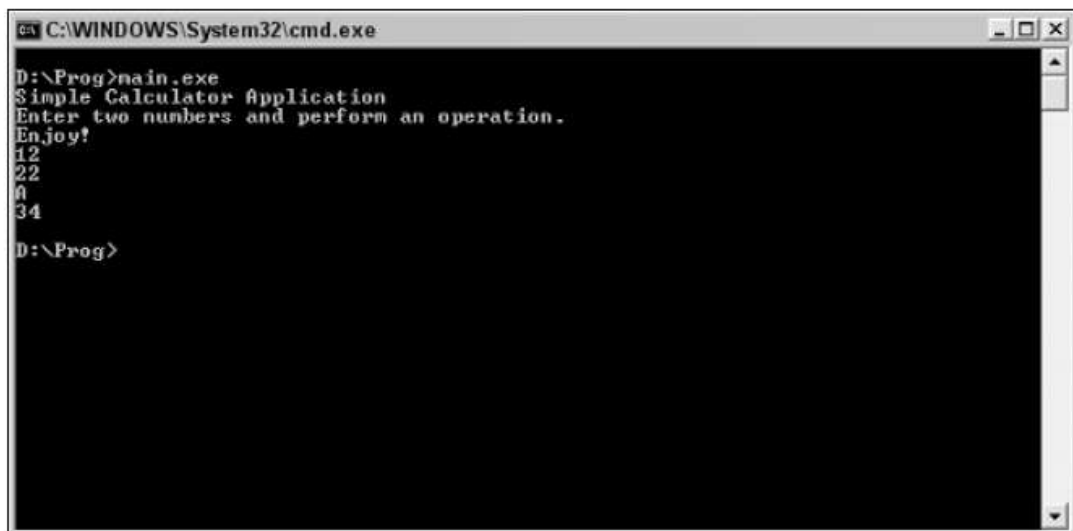
أول شيء يحتاجه البرنامج هو مراجعة عامة على ما يفعله. وهذا يعد المشهد لما يفعله المستخدمون في وقت لاحق , وإعطائهم الثقة في أنهم قد إختاروا التطبيق الصحيح . وأنه جاهز للعمل.

ها هو مثال مبسط على مراجعة البرنامج

```
#include <iostream.h>
void main()
{
    float num1;
    float num2;
    char op;
    float ans;
    cout << "Simple Calculator Application" << endl;
    cout << "Enter two numbers and perform an operation." << endl;
    cout << "Enjoy!" << endl;

    cin >> num1;
    cin >> num2;
    cin >> op;
    if (op == 65)
        ans = num1 + num2;
    if (op == 83)
        ans = num1 - num2;
    if (op == 77)
        ans = num1 * num2;
    if (op == 68)
        ans = num1 / num2;
    cout << ans << endl;
}
```

مخرجات هذا الكود تظهر في الشكل , فهو واضح , وبشكل مصغر أيضاً , ويعطى المستخدمون فكرة عن ماذا يتوقع التطبيق منهم .



```
C:\WINDOWS\System32\cmd.exe
D:\Prog>main.exe
Simple Calculator Application
Enter two numbers and perform an operation.
Enjoy!
12
22
0
34
D:\Prog>
```

قم بتبسيط الأمور قدر المستطاع — لا تعطى المستخدم كميات هائلة من النصوص ليقراها. فهذا غير مشجع ومن المحتمل ألا يقرأها كلها . لأنها معلومات كثيرة جداً .

```

#include <iostream.h>
void main()
{
    float num1;
    float num2;
    char op;
    float ans;
    cout << "Simple Calculator Application" << endl;
    cout << "Enter two numbers and perform an operation." << endl;
    cout << "Add!" << endl;
    cout << "Subtract!" << endl;
    cout << "Multiply!" << endl;
    cout << "Divide!" << endl;
    cout << "You need to enter two numbers." << endl;
    cout << "And choose an operation to carry out on them." << endl;
    cout << "Swift and accurate!" << endl;
    cout << "Tell your friends!" << endl;
    cout << "Enjoy!" << endl;
    cin >> num1;
    cin >> num2;
    cin >> op;
    if (op == 65)
        ans = num1 + num2;
    if (op == 83)
        ans = num1 - num2;
    if (op == 77)
        ans = num1 * num2;
    if (op == 68)
        ans = num1 / num2;
    cout << ans << endl;
}

```

تظهر المخرجات في هذا الشكل , أعتقد أنك توافقني الرأي ان هذا مثال لطريقة إعطاء معلومات كثيرة جداً لقيم ابتدائية .

```

C:\WINDOWS\System32\cmd.exe
D:\Prog>main.exe
Simple Calculator Application
Enter two numbers and perform an operation.
Add!
Subtract!
Multiply!
Divide!
You need to enter two numbers.
And choose an operation to carry out on them.
Swift and accurate!
Tell your friends!
Enjoy!
12
22
+
34
D:\Prog>

```

Proper Prompting for Input

المساحة التالية للتحسين أن برنامج الحاسبة الخاص بنا يمكن ان يكون لديه تاجيلات لإدخالات المستخدم المطالبة بالمدخلات يجب أن تكون واضحة , فعندما تسأل عن أرقام , كن دقيق فيما تبحث عنه . هل تبحث عن أرقام صحيحة أو أرقام تقبل الكسور ؟ هل هناك حدود ؟ قم بتوضيح ذلك جيداً .

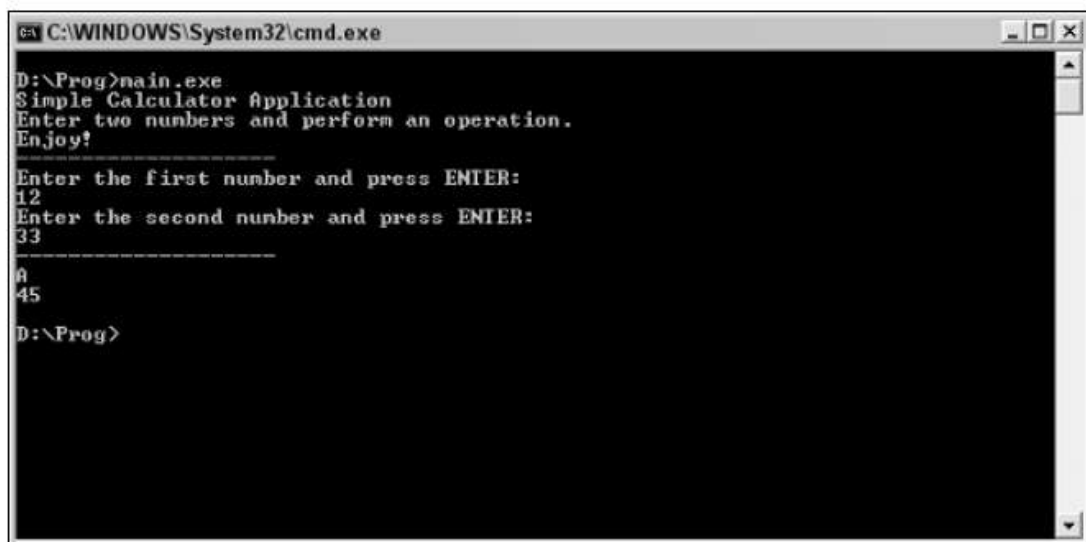
كن حريصاً مع التعبير أيضاً. إذا قلت "أرقام صحيحة" ربما يكون هناك اناس يتضايقون بمثل هذا المصطلح .

إذا وجدت انه يجب عليك ان تعطى المزيد من التعليمات والشروط في نوع المدخلات التي يمكن أن يدخلها المستخدم إلى برنامجك إذاً ربما يكون عليك أن تكتب برنامج أكثر تقييداً . على سبيل المثال , إذا كان برنامجك يأخذ فقط أرقام صحيحة , هل هناك سبب جيد في لماذا او هل هو تصميم محدد لا ينبغي حقاً ان يكون ؟

الكود التالي هو تحسين ضخم لأن به تأجيل لإخذ مدخلات المستخدم ويوضح انها الأرقام .

```
#include <iostream.h>
void main()
{
    float num1;
    float num2;
    char op;
    float ans;
    cout << "Simple Calculator Application" << endl;
    cout << "Enter two numbers and perform an operation." << endl;
    cout << "Enjoy!" << endl;
    cout << "-----" << endl;
    cout << "Enter the first number and press ENTER:" << endl;
    cin >> num1;
    cout << "Enter the second number and press ENTER:" << endl;
    cin >> num2;
    cout << "-----" << endl;
    cin >> op;
    if (op == 65)
        ans = num1 + num2;
    if (op == 83)
        ans = num1 - num2;
    if (op == 77)
        ans = num1 * num2;
    if (op == 68)
        ans = num1 / num2;
    cout << ans << endl;
}
```

نرى ذلك في الواقع في الشكل



```
C:\WINDOWS\System32\cmd.exe
D:\Prog>nain.exe
Simple Calculator Application
Enter two numbers and perform an operation.
Enjoy!
-----
Enter the first number and press ENTER:
12
Enter the second number and press ENTER:
33
-----
A
45
D:\Prog>
```

هذا التعديل يغطي المدخلات الرقمية , التي تتحكم في تنفيذ العملية . مرة أخرى يحتاج هذا ان يكون واضحاً ودقيق .

هذا ما كان لدينا من قبل

```
#include <iostream.h>
void main()
{
    float num1;
    float num2;
    char op;
    float ans;
    cout << "Simple Calculator Application" << endl;
    cout << "Enter two numbers and perform an operation." << endl;
    cout << "Enjoy!" << endl;
    cout << "-----" << endl;
    cout << "Enter the first number and press ENTER:" << endl;
    cin >> num1;
    cout << "Enter the second number and press ENTER:" << endl;
    cin >> num2;
    cout << "Press A to add the two numbers."
        << endl
        << "Press S to subtract the two numbers."
        << endl
        << "Press M to multiply the two numbers."
        << endl
        << "Press D to divide the two numbers."
        << endl;
    cout << "-----" << endl;
    cin >> op;
    if (op == 65)
        ans = num1 + num2;
    if (op == 83)
        ans = num1 - num2;
    if (op == 77)
        ans = num1 * num2;
    if (op == 68)
        ans = num1 / num2;
    cout << ans << endl;
}
```

هل لاحظت شيئاً غير طبيعي في هذه التعليمات التي إستخدمها ؟ هل لاحظت انهم كانوا حقيقتاً بهم شيئاً من الدقة ! لقد ادخلت الحرف المناسب ثم بعدها عليك أن تضغط Enter للتطبيق ليستمر .

```
#include <iostream.h>
void main()
{
    float num1;
    float num2;
    char op;
    float ans;
    cout << "Simple Calculator Application" << endl;
    cout << "Enter two numbers and perform an operation." << endl;
    cout << "Enjoy!" << endl;
    cout << "-----" << endl;
    cout << "Enter the first number and press ENTER:" << endl;
```

```

cin >> num1;
cout << "Enter the second number and press ENTER:" << endl;
cin >> num2;
cout << "Press A followed by ENTER to add the two numbers."
<< endl
<< "Press S followed by ENTER to subtract the two numbers."
<< endl
<< "Press M followed by ENTER to multiply the two numbers."
<< endl
<< "Press D followed by ENTER to divide the two numbers."
<< endl;
cout << "-----" << endl;
cin >> op;
if (op == 65)
    ans = num1 + num2;
if (op == 83)
    ans = num1 - num2;
if (op == 77)
    ans = num1 * num2;
if (op == 68)
    ans = num1 / num2;
cout << ans << endl;
}

```

الشكل التالي يعرض المخرجات من هذا الكود .

```

C:\WINDOWS\System32\cmd.exe
D:\Prog>main.exe
Simple Calculator Application
Enter two numbers and perform an operation.
Enjoy!
-----
Enter the first number and press ENTER:
12
Enter the second number and press ENTER:
22
Press A followed by ENTER to add the two numbers
Press S followed by ENTER to subtract the two numbers.
Press M followed by ENTER to multiply the two numbers.
Press D followed by ENTER to divide the two numbers.
-----
S
-10
D:\Prog>

```

Annotating Output

تذييل المخرجات

تقديم حاشية للمدخلات لا يقل أهمية عن نفس الشئ للمخرجات , فأنت لا تريد مخرجات على الشاشة تبقى غير معروفة .

```

#include <iostream.h>
void main()
{
    float num1;
    float num2;

```

```

char op;
float ans;
cout << "Simple Calculator Application" << endl;
cout << "Enter two numbers and perform an operation." << endl;
cout << "Enjoy!" << endl;
cout << "-----" << endl;
cout << "Enter the first number and press ENTER:" << endl;
cin >> num1;
cout << "Enter the second number and press ENTER:" << endl;
cin >> num2;
cout << "Press A followed by ENTER to add the two numbers."
<< endl
<< "Press S followed by ENTER to subtract the two numbers."
<< endl
<< "Press M followed by ENTER to multiply the two numbers."
<< endl
<< "Press D followed by ENTER to divide the two numbers."
<< endl;
cout << "-----" << endl;
cin >> op;
if (op == 65)
    ans = num1 + num2;
if (op == 83)
    ans = num1 - num2;
if (op == 77)
    ans = num1 * num2;
if (op == 68)
    ans = num1 / num2;
cout << "The answer is " << ans << endl;
)

```

Confirming Exit

تأكيد الإنهاء

عندما ينتهي برنامجك , فكرة جيدة ان تخبر المستخدم انه أتم العمل وسيتم الأغلاق , على الجانب الآخر فربما يكون المستخدم متعجباً إذا ظل هو او هي بالعمل معه واغلق دون أن يدري.

```

#include <iostream.h>
void main()
{
    float num1;
    float num2;
    char op;
    float ans;
    cout << "Simple Calculator Application" << endl;
    cout << "Enter two numbers and perform an operation." << endl;
    cout << "Enjoy!" << endl;
    cout << "-----" << endl;
    cout << "Enter the first number and press ENTER:" << endl;
    cin >> num1;
    cout << "Enter the second number and press ENTER:" << endl;
    cin >> num2;
}

```

```

cout << "Press A followed by ENTER to add the two numbers."
<< endl
<< "Press S followed by ENTER to subtract the two numbers."
<< endl
<< "Press M followed by ENTER to multiply the two numbers."
<< endl
<< "Press D followed by ENTER to divide the two numbers."
<< endl;

cout << "-----" << endl;
cin >> op;
if (op == 65)
    ans = num1 + num2;
if (op == 83)
    ans = num1 - num2;
if (op == 77)
    ans = num1 * num2;
if (op == 68)
    ans = num1 / num2;
cout << "The answer is " << ans << endl;
cout << "Thanks for using the calculator." << endl;
cout << "Application now exiting ... " << endl;
}

```

تظهر مخرجات هذا في الشكل

```

C:\WINDOWS\System32\cmd.exe
D:\Prog>main.exe
Simple Calculator Application
Enter two numbers and perform an operation.
Enjoy!

Enter the first number and press ENTER:
12
Enter the second number and press ENTER:
22
Press A followed by ENTER to add the two numbers
Press S followed by ENTER to subtract the two numbers.
Press M followed by ENTER to multiply the two numbers.
Press D followed by ENTER to divide the two numbers.
-----
S
The answer is -10
Thanks for using the calculator.
Application now exiting ...

D:\Prog>

```

Adding Simple Help

إضافة مساعدة بسيطة

كل البرامج إلا البرامج البسيطة يمكن أن تقدم بعض المساعدة لكيفية إستخدامهم .

مع تطبيق مثل هذا الطريقة نجد من الأفضل ان نستهل بنظام مساعدة من أجل المستخدم لنبلغه بإدخال قيمة؟ , الحرف , الذي بدوره قد تم توضيحه مع نظام المساعدة .

ليعمل هذا مع ذلك , فنحتاج إلى إعادة تركيب الكود الذي لدينا , فنقوم بتقطيع بعض الكود إلى دوال منفصلة .


```

#include <iostream.h>
void calc()
{
    float num1;
    float num2;
    char op;
    float ans;
    cout << "Simple Calculator Application" << endl;
    cout << "Enter two numbers and perform an operation." << endl;
    cout << "Enjoy!" << endl;
    cout << "-----" << endl;
    cout << "Enter the first number and press ENTER:" << endl;
    cin >> num1;
    cout << "Enter the second number and press ENTER:" << endl;
    cin >> num2;
    cout << "Press A followed by ENTER to add the two numbers."
        << endl
        << "Press S followed by ENTER to subtract the two numbers."
        << endl
        << "Press M followed by ENTER to multiply the two numbers."
        << endl
        << "Press D followed by ENTER to divide the two numbers."
        << endl;
    cout << "-----" << endl;
    cin >> op;
    if (op == 65)
    {
        ans = num1 + num2;
    }
    if (op == 83)
    {
        ans = num1 - num2;
    }
    if (op == 77)
    {
        ans = num1 * num2;
    }
    if (op == 68)
    {
        ans = num1 / num2;
    }
    cout << "The answer is " << ans << endl;
    cout << "Thanks for using the calculator." << endl;
    cout << "Application now exiting ... " << endl;
}

void helpsystem()
{
    char op;
    cout << "Help System" << endl;
    cout << "Are you ready to use this application?" << endl;
    cout << "Press Y followed by ENTER to continue." << endl;
    cout << "Press N followed by ENTER to exit." << endl;
}

```

```

        cout << "Press ? followed by ENTER for help." << endl;
        cin >> op;
        if (op == 89)
        {
            calc();
        }
        if (op == 63)
        {
            helpsystem();
        }
        if (op == 78)
        {

        }

    }

void main()
{
    char op;
    cout << "Simple Calculator Application" << endl;
    cout << "Enter two numbers and perform an operation." << endl;
    cout << "Enjoy!" << endl;
    cout << "Are you ready to use this application?" << endl;
    cout << "Press Y followed by ENTER to continue." << endl;
    cout << "Press N followed by ENTER to exit." << endl;
    cout << "Press ? followed by ENTER for help." << endl;
    cin >> op;
    if (op == 89)
    {
        calc();
    }
    if (op == 63)
    {
        helpsystem();
    }
    if (op == 78)
    {

    }

}

```

قد حدث هنا المزيد , إليك جولة سريعة .
 1 - تم تقطيع الكود إلى ثلاثة دوال .

```

main()
calc()
helpsystem()

```

2 - الدالة main() الآن فقط تعالج عرض النص المبدئى للتطبيق وأيضاً تسألك إذا كنت مستعد لتستخدم التطبيق , أو تريد أن تغلق , أو إذا ما تريد المساعدة , الحرف y يمثل 89 فى النظام العشرى , بينما N هي 78 و 63 . على التوالى .

```

void main()
{
    char op;
    cout << "Simple Calculator Application" << endl;
    cout << "Enter two numbers and perform an operation." << endl;
    cout << "Enjoy!" << endl;
    cout << "Are you ready to use this application?" << endl;
    cout << "Press Y followed by ENTER to continue." << endl;
    cout << "Press N followed by ENTER to exit." << endl;
    cout << "Press ? followed by ENTER for help." << endl;
    cin >> op;
    if (op == 89)
    {
        calc();
    }
    if (op == 63)
    {
        helpsystem();
    }
    if (op == 78)
    {
    }
}
}

```

3 - لاحظ كيف انهيينا التطبيق — فقط لا تعطى البرنامج أى شئ آخر ليفعله إستجابة لـ مدخلات N (78) ويتم إغلاقه .

```

if (op == 78)
{
}

```

4 - الدالة Calc() هي المسئولة عن العملية الرياضية الرئيسية التى تتم بواسطة التطبيق . هذا الكود مشابهة للكود الذى إستخدمناه مسبقاً . ولكن لدينا إستخدام مرتباً للأقواس المتعرجة فى الأوامر الشرطية .

```

void calc()
{
    float num1;
    float num2;
    char op;
    float ans;
    cout << "Simple Calculator Application" << endl;
    cout << "Enter two numbers and perform an operation." << endl;
    cout << "Enjoy!" << endl;
    cout << "-----" << endl;
    cout << "Enter the first number and press ENTER:" << endl;
    cin >> num1;
    cout << "Enter the second number and press ENTER:" << endl;
    cin >> num2;
    cout << "Press A followed by ENTER to add the two numbers."
        << endl
        << "Press S followed by ENTER to subtract the two numbers."
        << endl
        << "Press M followed by ENTER to multiply the two numbers."

```

```

        << endl
        << "Press D followed by ENTER to divide the two numbers."
        << endl;
    cout << "-----" << endl;
    cin >> op;
    if (op == 65)
    {
        ans = num1 + num2;
    }
    if (op == 83)
    {
        ans = num1 - num2;
    }
    if (op == 77)
    {
        ans = num1 * num2;
    }
    if (op == 68)
    {
        ans = num1 / num2;
    }
    cout << "The answer is " << ans << endl;
    cout << "Thanks for using the calculator." << endl;
    cout << "Application now exiting ... " << endl;
}

```

5 - الدالة `helpsystem()` تعالج عرض معلومات المساعدة التي نرودها . و تسمح أيضاً أن يتم إعادة عرض المساعدة التطبيق المستخدم أو الإغلاق .

```

void helpsystem()
{
    char op;
    cout << "Help System" << endl;
    cout << "Are you ready to use this application?" << endl;
    cout << "Press Y followed by ENTER to continue." << endl;
    cout << "Press N followed by ENTER to exit" << endl;
    cout << "Press ? followed by ENTER for help." << endl;
    cin >> op;
    if (op == 89)
    {
        calc();
    }
    if (op == 63)
    {
        helpsystem();
    }
    if (op == 78)
    {

    }
}

```

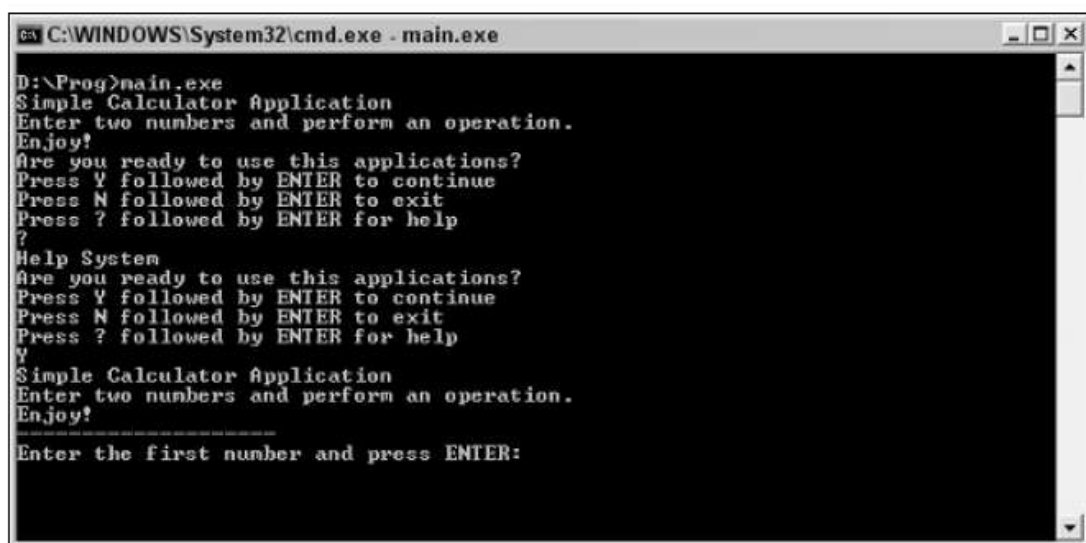
6 - هناك أيضاً بعض التغيرات الثانوية لاستيعاب إعادة استخدام المتغيرات .

علينا ان نفعل شئ من إعادة التركيب لإستيعاب نظام المساعدة ولكن هذا جيد لأن هذا أمر طبيعي . فقد تم تقطيع الكود الان إلى مكونات مجزئة مما يمكننا بناء وإضافة المزيد من المزايا .

لفعل هذا , قمنا بإضافة دالة خاصة تتعامل مع مخرجات المساعدة حيث يكون البرنامج قادراً على أن يعطينا إياها . ويمكن ان نوطن التعليمات من أجل البرنامج

```
void helpsystem()
{
    char op;
    cout << "Help System" << endl;
    cout << "-----" << endl;
    cout << "To work this application you enter two numbers and then choose" <<
endl;
    cout << "whether you want those numbers added together, subtracted, " <<
endl;
    cout << "multiplied, or divided. You choose this operation by inputting" <<
endl;
    cout << "A, S, M, or D, respectively." << endl;
    cout << "The result is displayed on-screen." << endl;
    cout << "Are you ready to use this application?" << endl;
    cout << "Press Y followed by ENTER to continue." << endl;
    cout << "Press N followed by ENTER to exit." << endl;
    cout << "Press ? followed by ENTER for help." << endl;
    cin >> op;
    if (op == 89)
    {
        calc();
    }
    if (op == 63)
    {
        helpsystem();
    }
    if (op == 78)
    {
    }
}
```

إذا تم تفسير هذ الكود الآن وتم تشغيله سترى عمل هذه الدوال الثلاث وستعطينا بعض المدخلات البسيطة . (إنظر الشكل)



```
C:\WINDOWS\System32\cmd.exe - main.exe
D:\Prog>main.exe
Simple Calculator Application
Enter two numbers and perform an operation.
Enjoy!
Are you ready to use this applications?
Press Y followed by ENTER to continue
Press N followed by ENTER to exit
Press ? followed by ENTER for help
?
Help System
Are you ready to use this applications?
Press Y followed by ENTER to continue
Press N followed by ENTER to exit
Press ? followed by ENTER for help
Y
Simple Calculator Application
Enter two numbers and perform an operation.
Enjoy!
Enter the first number and press ENTER:
```

Confirmations

التأكيدات

طريقة أخرى جيدة للتأكد من أن المستخدم قد أدخل البيانات المناسبة إلى التطبيق هي عرض ما أدخله لتذكره بما أدخل (على الرغم من هذا المثال ستظل الأرقام على الشاشة)

```
void calc()
{
    float num1;
    float num2;
    char op;
    float ans;
    cout << "Simple Calculator Application" << endl;
    cout << "Enter two numbers and perform an operation." << endl;
    cout << "Enjoy!" << endl;
    cout << "-----" << endl;
    cout << "Enter the first number and press ENTER:" << endl;
    cin >> num1;
    cout << "Enter the second number and press ENTER:" << endl;
    cin >> num2;
    cout << "You entered " << num1 << " and " << num2 << endl;
    cout
        << "Press A followed by ENTER to add the two numbers."
        << endl
        << "Press S followed by ENTER to subtract the two numbers."
        << endl
        << "Press M followed by ENTER to multiply the two numbers."
        << endl
        << "Press D followed by ENTER to divide the two numbers."
        << endl;
    cout << "-----" << endl;
    cin >> op;
    if (op == 65)
    {
        ans = num1 + num2;
    }
    if (op == 83)
    {
        ans = num1 - num2;
    }
    if (op == 77)
    {
        ans = num1 * num2;
    }
    if (op == 68)
    {
        ans = num1 / num2;
    }
    cout << "The answer is " << ans << endl;
    cout << "Thanks for using the calculator." << endl;
    cout << "Application now exiting ... " << endl;
}
```

Moving Away from the Text-Based Interface

الانتقال بعيداً عن الواجهة المعتمدة على النصوص

إلى هذا الحد فقد بحثنا في الواجهات المعتمدة على النصوص . بقدر ما بهم من فائدة . إلا أن معظم التطبيقات هذه الأيام تستخدم نظام الواجهة الرسومية . كما يوجد في Microsoft Windows .

ليس لدينا الوقت أو المجال هنا لنذهب في التفاصيل في كيفية إنشاء واجهة رسومية لبرنامجك في لغة مثل C++ — فهذا يتطلب المزيد من التغطية أكثر من اننا يمكن أن نغطيهم هنا والأدوات المستخدمة التي هي حالياً خارج نطاق الكتاب .

مع ذلك , فهي فكرة جيدة ان نكتشف العناصر الأساسية للواجهة وتنظر على كيف تصنع منهم استخدام جيد — ولنتأكد من أنك مستعد أن تبني انظمة رسومية , فليس هناك احد في الجوار يمكن أن يخبرك كيف تستخدم هذه العناصر الفردية !

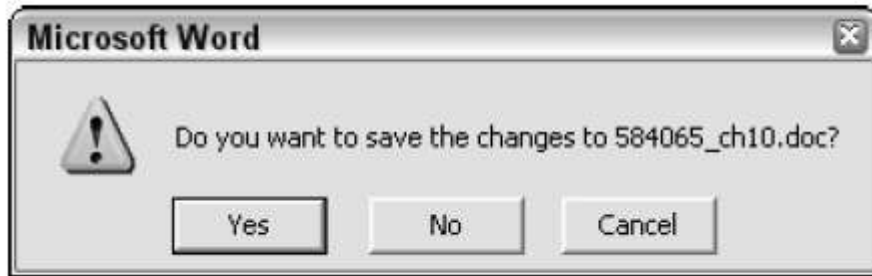
Buttons

الزر / المفتاح

أحد أعظم وأسرع طرق الإدخال الشائعة . (إنظر الشكل)

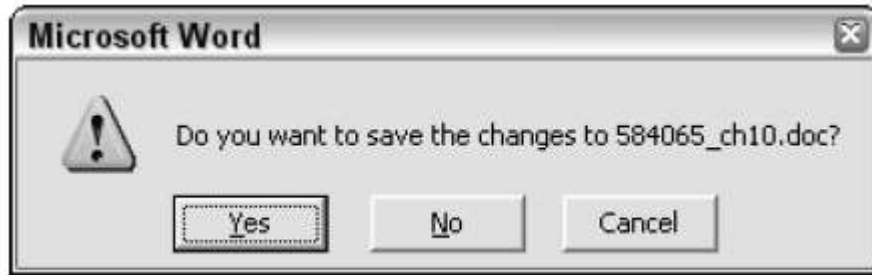


تستخدم الأزرار لإعداد فعل في الحركة . الضغط على زر هو بداية الفعل . يمكن أيضاً أن تكون طريقة للبرنامج أن يسأل مدخلات من المستخدم في قرار يجب أن يأخذه .



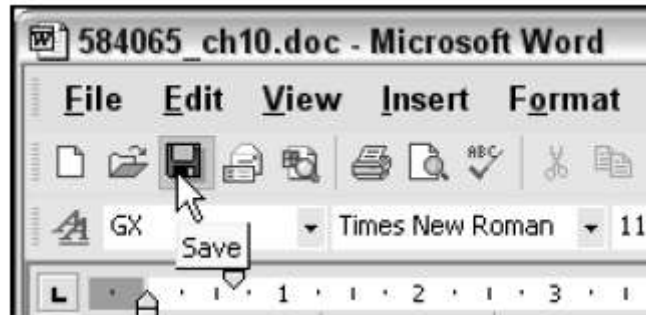
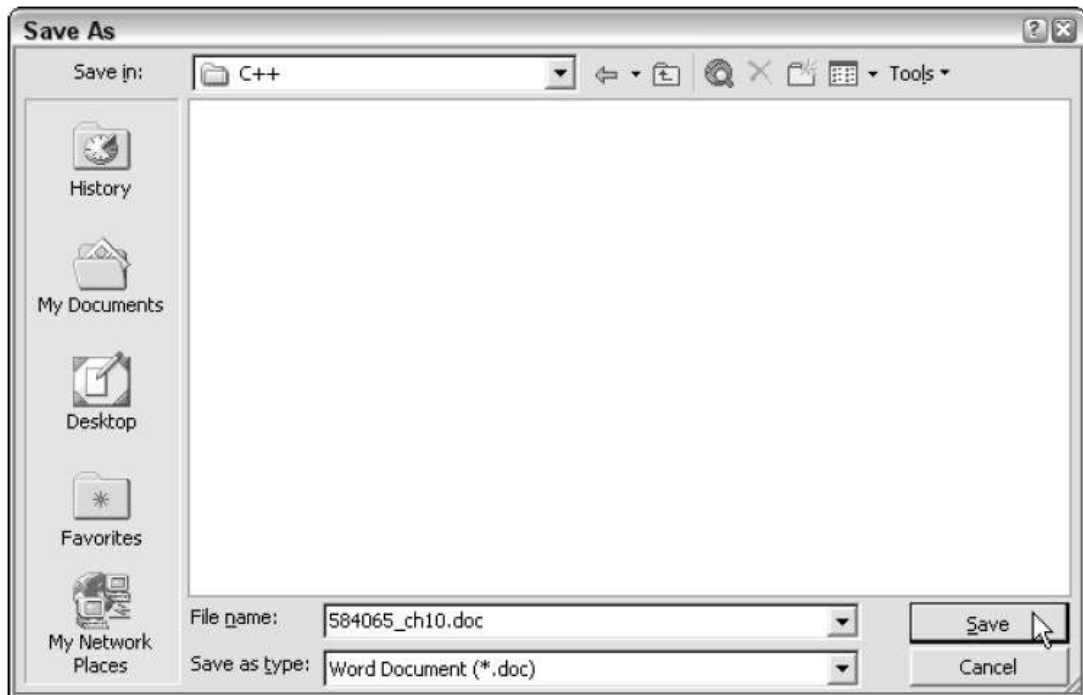
المنطق خلف استخدام الأزرار مقابلةً بضغط المفاتيح هو أن الناس يمكن أن يستخدموا الفأرة للضغط على كائن ما بشكل أسرع من كتابة حرف أو تركيبة من الحروف على لوحة المفاتيح .

بينما الأزرار قدمت لجعل الانتقال داخل التطبيق أسهل بإعطاء المستخدم شئ ما يقوم بالضغط عليه. معظم الأزرار يمكن ان تعمل بواسطة لوحة المفاتيح أيضاً . إنظر الحرف الذى تحته خط فى الشكل الذى يعرض ان الزر يمكن أن يضغط بواسطة الحرف مع الزر alt .



بعض الأزرار , مثل OK و Cancel , فهناك زر لوحة المفاتيح إفتراضى لتشغيلهم — Enter و Esc , على التوالى .

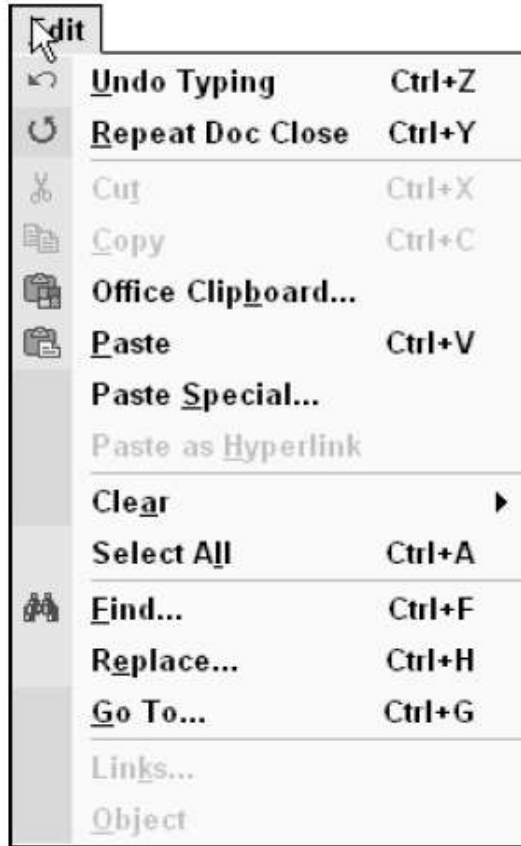
عادةً ما تميز الازرار بالنص (إنظر الشكل) , على الرغم من أن بعض الأزرار يمكن ان يمثل بصورة (إنظر الشكل الذى يليه)



Menus

القوائم

هنا شئ آخر يحتاج مقدمة غير حقيقة . القائمة المنبثقة تعرض للمستخدم قائمة بالإختيارات من أيهم يختار (إنظر الشكل)



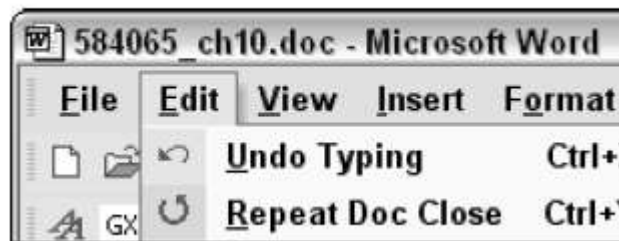
القائمة تظهر فى الشكل التالى



عنصر القائمة يظهر فى الشكل



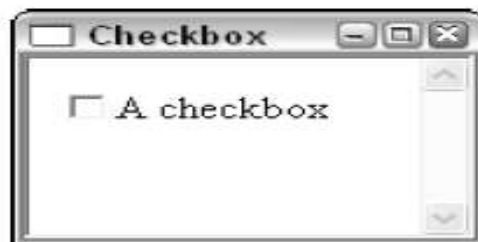
كما فى الأزرار , الفكرة هى أنه أسرع للإبحار خلال القائمة باستخدام الفأرة وصنع إختيارات بواسطة الضغط عليها فهى أسرع من الإبحار باستخدام ضغطات المفاتيح . مع ذلك , فقط مع الأزرار , فلدى القوائم خطوط تحتيه على بعض الحروف — إضغط Alt , وهذه الأحرف سوف تنشط القائمة أو عنصر القائمة (إنظر الشكل)



Check Box

مربع إختيارى

هو طريقة للمستخدم ليصنع إختيارات . الإختيار من هذه الوظائف (إنظر الشكل)



كما يمكن ان يأتى فى مجموعة حيث يمكن أن تختار إختيارات متعددة تزامنياً (إنظر الشكل)



صندوق الإختيارات مميزة بنص (إنظر الشكل)

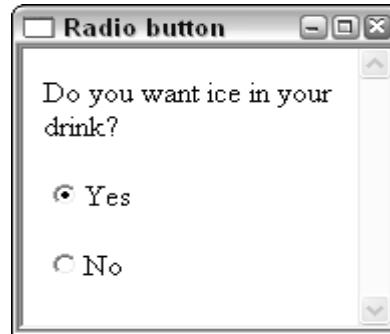


Radio Buttons

هى مثل السابقة , الإختلاف الوحيد هو أنك يمكن لك ان تختار إختيار واحد فقط فى اى مرة . (إنظر الشكل)



فهذا مفيد جداً عندما تريد ان تعطى المستخدم إختيارات ليختار منها إختيار واحد فقط (كما بالشكل)



كما أن بها علامة نصية بنفس الطريقة كما فى CheckBox . قم بعمل إختبار وصفى وتأكد من أنه لا يوجد غموض .

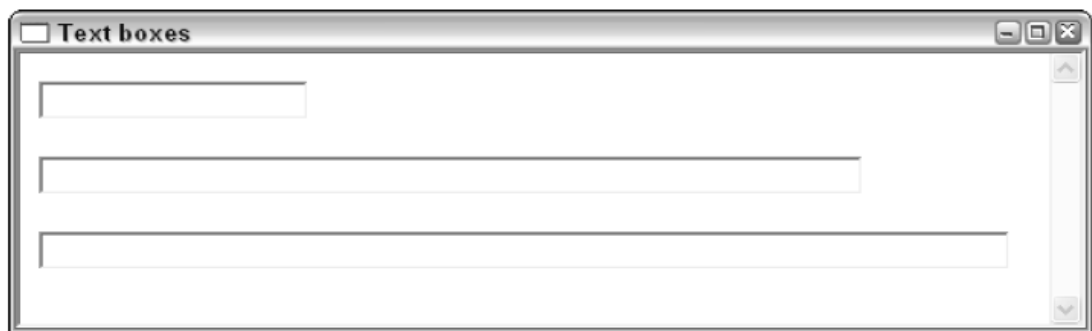
Single-Line Text Box

مربع نصى ذو سطر واحد

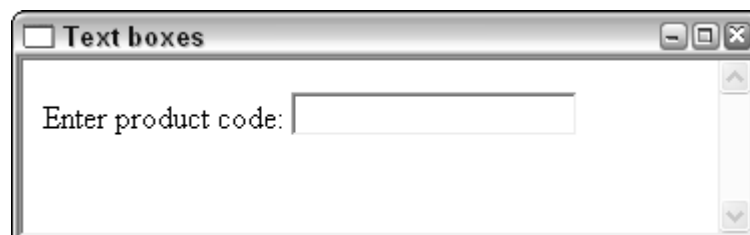
الطريقة الأبسط الرئيسية للمستخدم ليدخل الحروف الأبجدية إلى البرنامج (إنظر الشكل)



فهي تأتى فى تنوع من الطول , ولكن إذا كان بالإمكان أن يصبخوا أكثر طويلاً فلا يمكن ان يكونوا أكثر عرضاً (إنظر الشكل)



تحتاج هذه الخانات النصية أو مربعات المص إلى أن تعلم أو تميز بوضوح (ترى الشكل)



يمكن أيضاً أن يكون لهم قيمة افتراضية قد تم اعداد برمجتها من قبل

Multiline Text Box

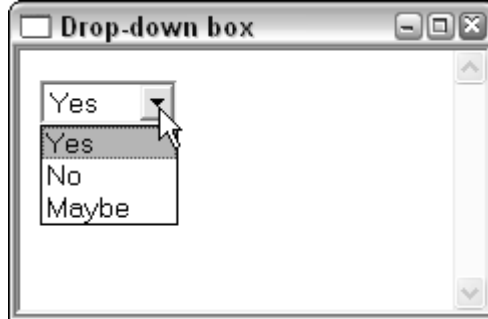
مربع النص المتعدد الأسطر

مشابه لما سبقه ذو السطر الواحد , الإختلاف فى أنه يمكن أن يأخذ أسطر متعددة (إنظر الشكل)

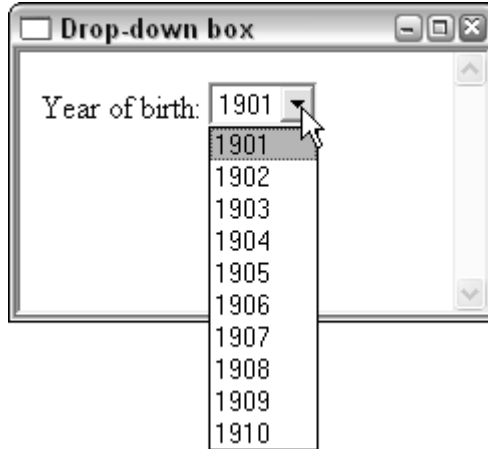
هذه المربعات النصية المتعددة الأسطر يمكن أن تعد لأشكال وأحجام متنوعة (إنظر الشكل)

Drop-Down Menu

هى قائمة محتويات فى جسم التطبيق . وتمكنك من إعطاء المستخدم طريقة سريعة وسهلة ليحدد إختيار من مجموعة إختيارات (إنظر الشكل)



القائمة فى الشكل السابق تأخذ مساحة أقل فى الشاشة بالمقارنة إلى إستعمال نوع آخر من عنصر الواجهة , مثال Radio Buttons (إنظر الشكل)



Putting It All Together

وضعهم كلهم معاً

لقد قمنا بتغطية معظم العناصر الأساسية , لذا الآن فهذا الوقت لنلقى نظرة على كيفية وضعهم معاً بفاعلية .

لفعل ذلك , سنفحص بعض الواجهات التى ربما تكون مألوفة لك ونرى كيف نستخدمهم بفاعلية.

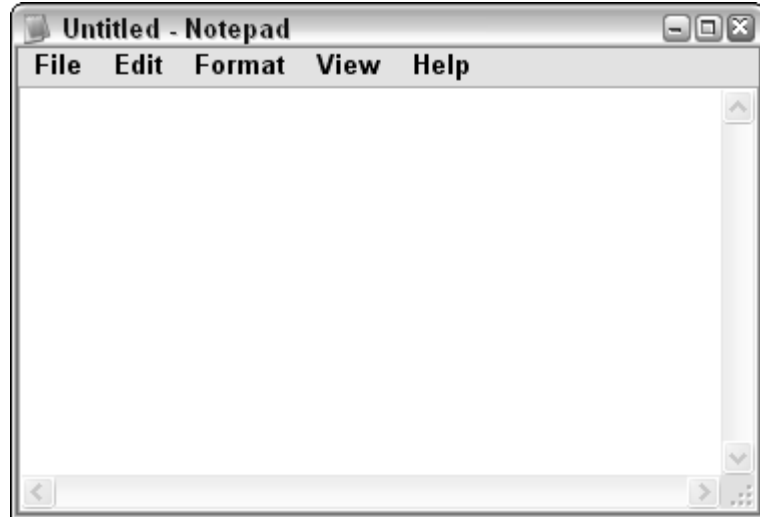
شئ واحد مهم لتتذكره وهو ان ليس هناك طريقة واحدة لتفعل أياً من هذا , ما تهدف إليه هو واجهة منطقية وسهلة الإستخدام . ربما تمر على بعض البرامج القديمة جداً , فتجد واجهة مستخدم غير قياسية و لا يوجد شئ خاطئ فى هذا طالما أنه سهل الإستخدام , ولا يتطلب المزيد من التعلم لتستخدمه .

تذكر أن الناس يحتاجون إستخدام التطبيق , ولا يحتاجون إلى تعلم واجهة جديدة .

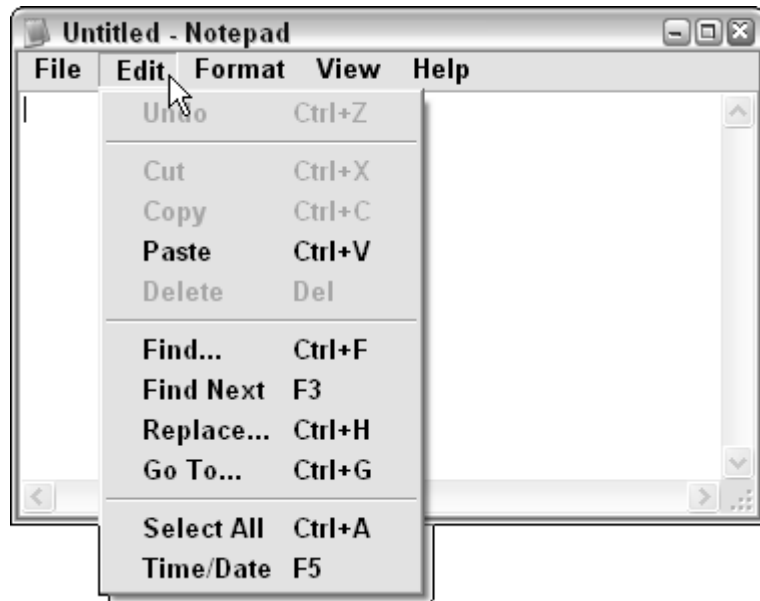
Simple Applications

تطبيقات بسيطة

دعنا نلقى نظرة على تطبيق بسيط — windows Notepad (إنظر الشكل)



نجد في هذه الواجهة ,تطبيق يقاد بالقوائم (يرى بالشكل) ,ولا توجد ازرار مرئية مبدئياً .



كل القوائم يتم الدخول عليها بإستخدام كلاً من الفأرة ولوحة المفاتيح

- ❑ **File:** ALT + F
- ❑ **Edit:** ALT + E
- ❑ **Format:** ALT + O
- ❑ **View:** ALT + V
- ❑ **Help:** ALT + H

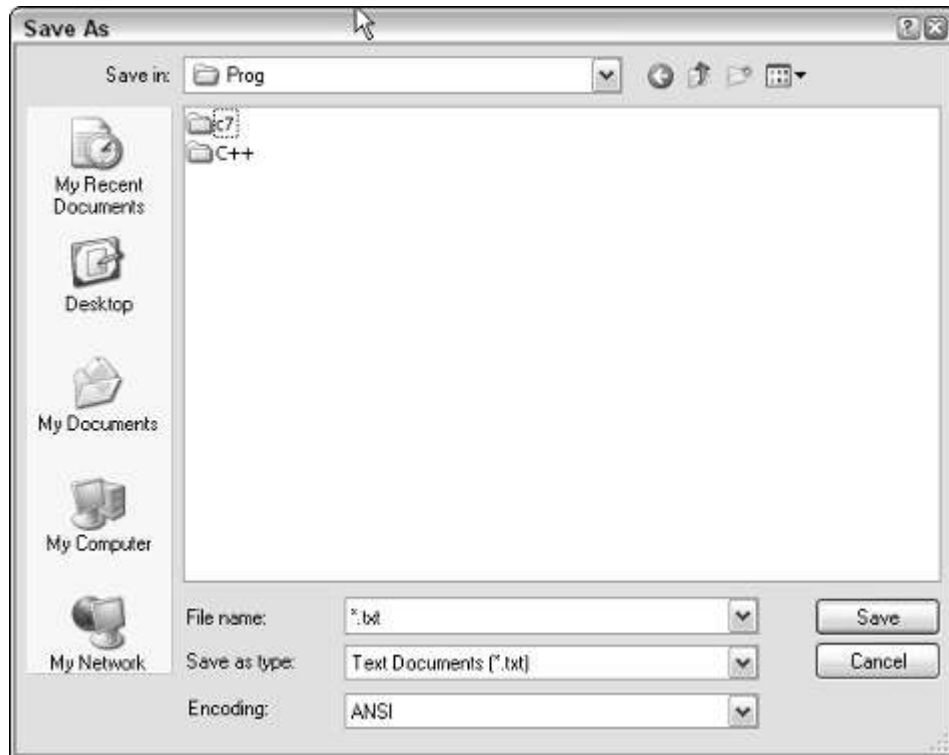
بعض العناصر في القوائم الفرعية تدخل بإستخدام إصطحاب Ctrl مع المفاتيح الساخنة , مثال

❑ **Save:** CTRL + S

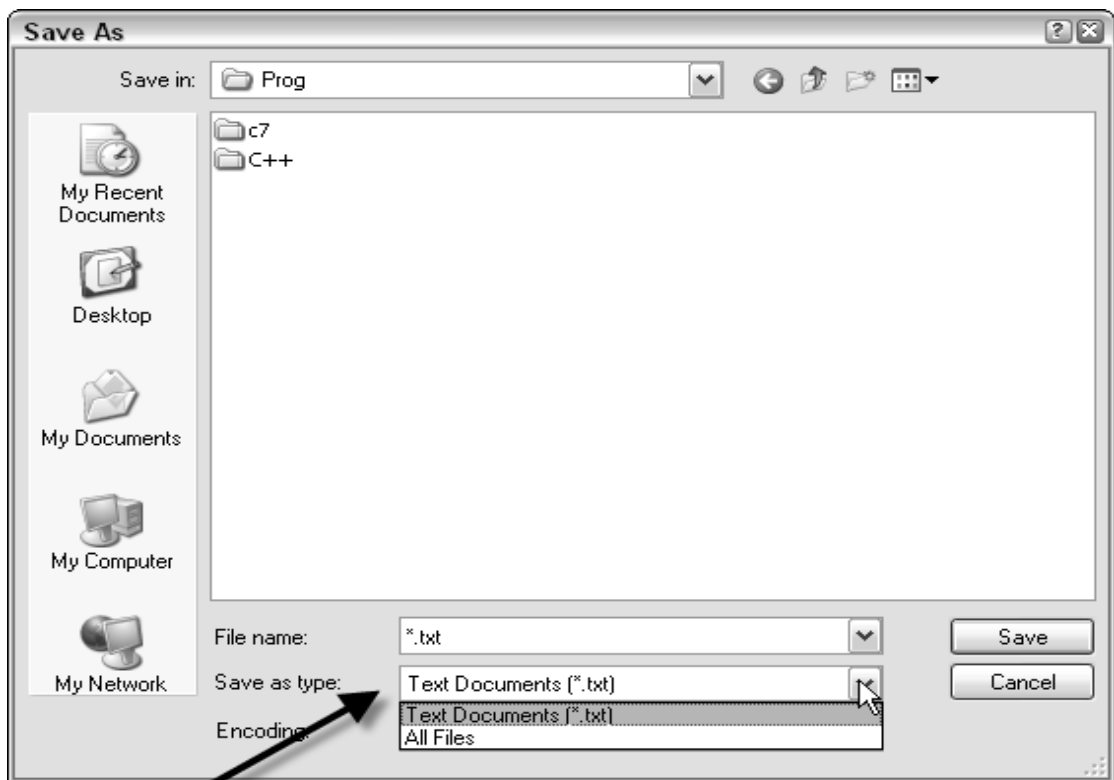
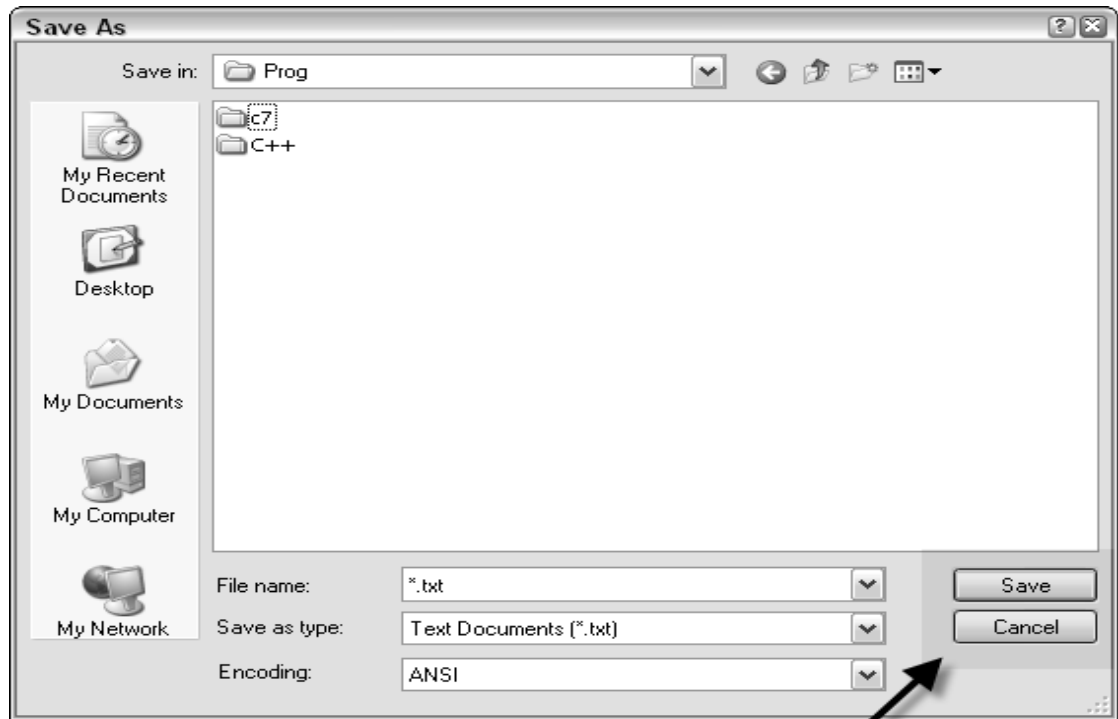
❑ **New:** CTRL + N

❑ **Open:** CTRL + O

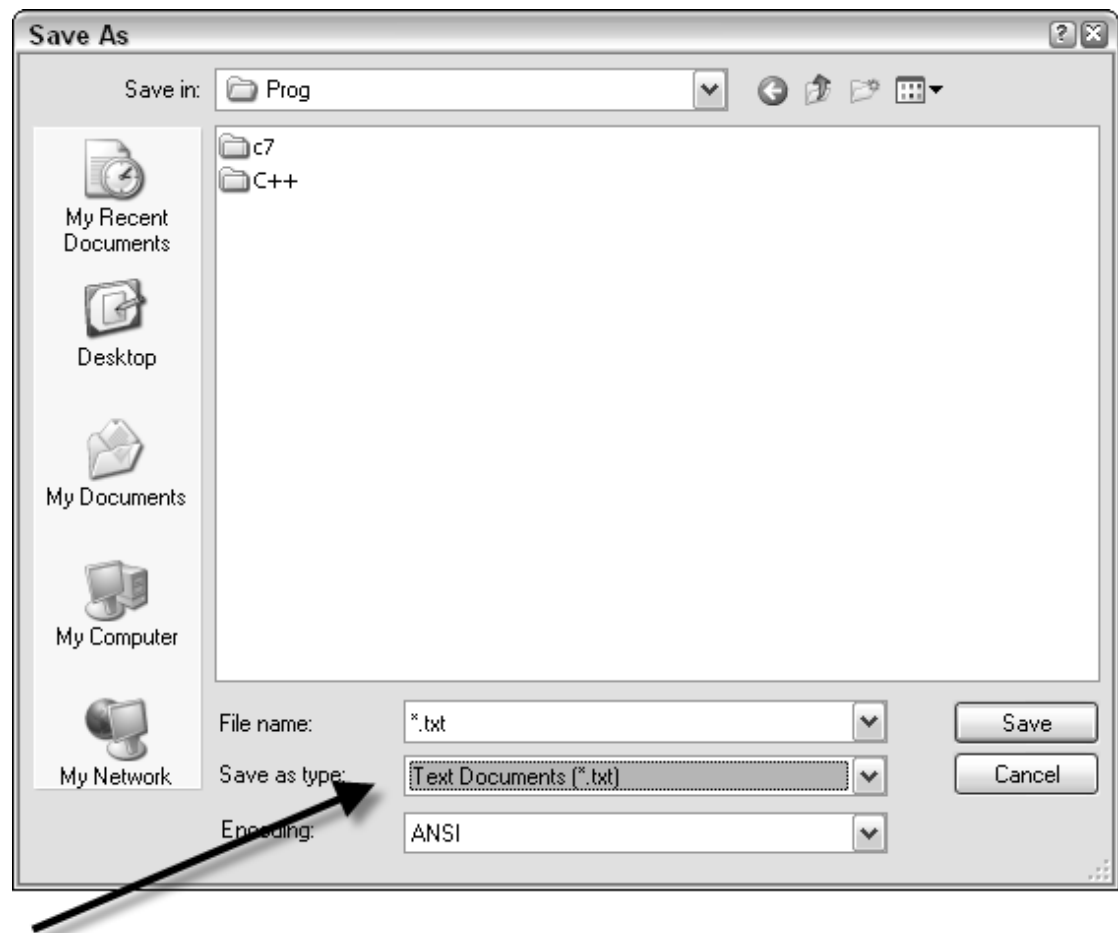
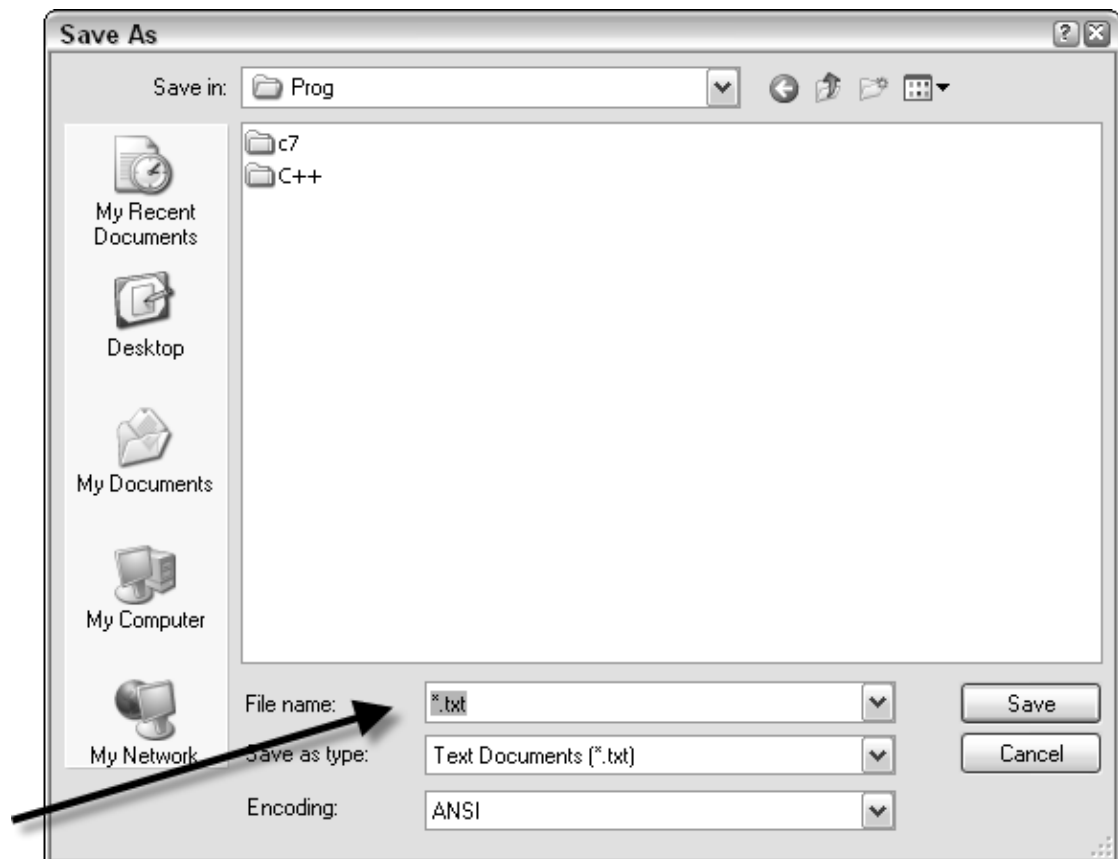
إذا حاولت ان تحفظ الملف , ستحصل على مربع حوارى يجعل لك إستخدام متنوع لعناصر من الواجهة (إنظر الشكل)



فهذه تحتوى على أزرار Buttons متعددة (إنظر الشكل) وقائمة منسدلة Drop-Down (إنظر الشكل التالى) ليساعدك لتبهر داخل ملفات النظام وتختار الموقع للملف لتحفظ الملف فيه , إسم الملف , والنوع .



وبينما انت في نافذة المربع الحوارى dialog box , حاول الضغط على مفتاح Tab ولاحظ ماذا يحدث .
 لاحظ كيف يركز (او العنصر الذى يمكن ان تستخدمه) التحركات من عنصر (مثال إسم الملف — إنظر الشكل) إلى آخر (نوع الملف — إنظر الشكل)



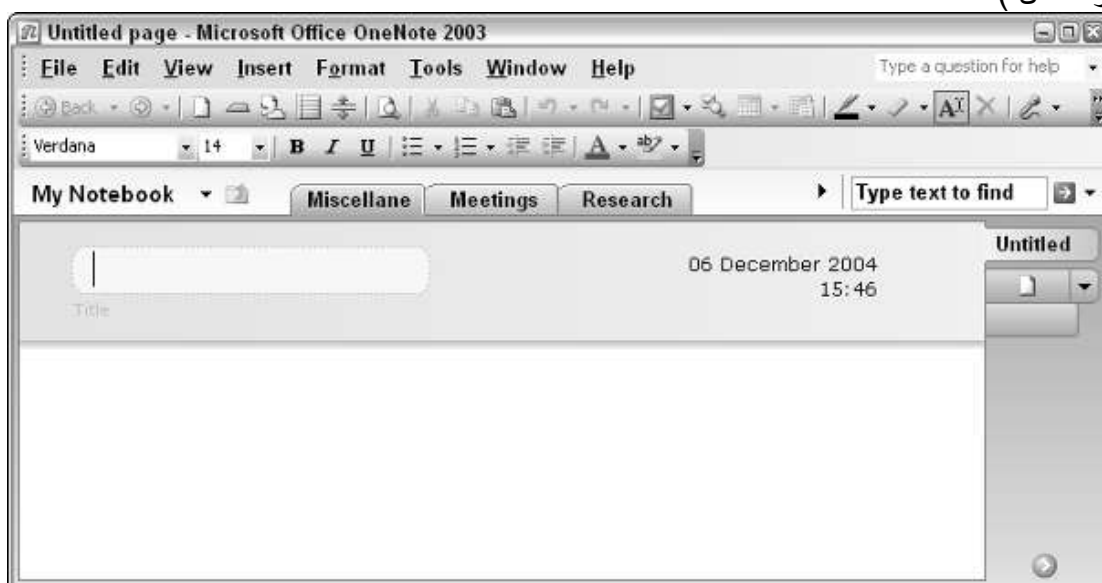
هذا التحرك من عنصر واحد إلى آخر يسمى **tabbing** , والترتيب في التركيز ليتحرك من عنصر واحد إلى آخر يعرف بـ **tab Order** .

بيئات البرمجة المختلفة تمكنك من إعداد هذا في طرق مختلفة , ولكن شئ واحد ستحتاجه لتكون حذراً فيه هو ان تركيز التحرك من عنصر واحد إلى غيره في طريق منطقي ولا يقفز ويتخطى على كل الأماكن .

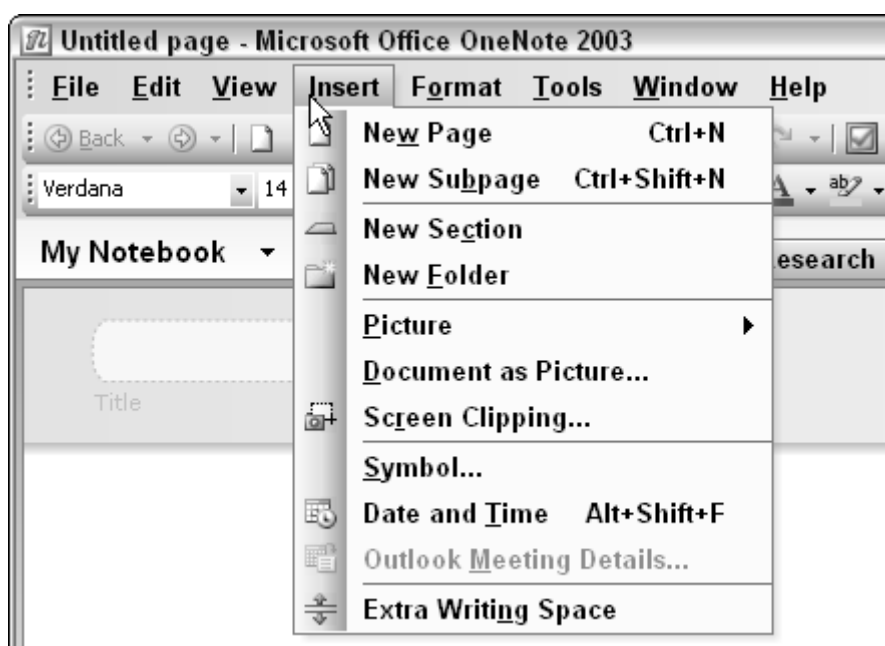
More Complicated Applications

المزيد من التطبيقات المعقدة

إذا أردت واجهة أكثر تعقيداً , فعليك أن تبحث في تطبيق أكثر تعقيداً كمثل Microsoft OneNote 2003 (إنظر الشكل)



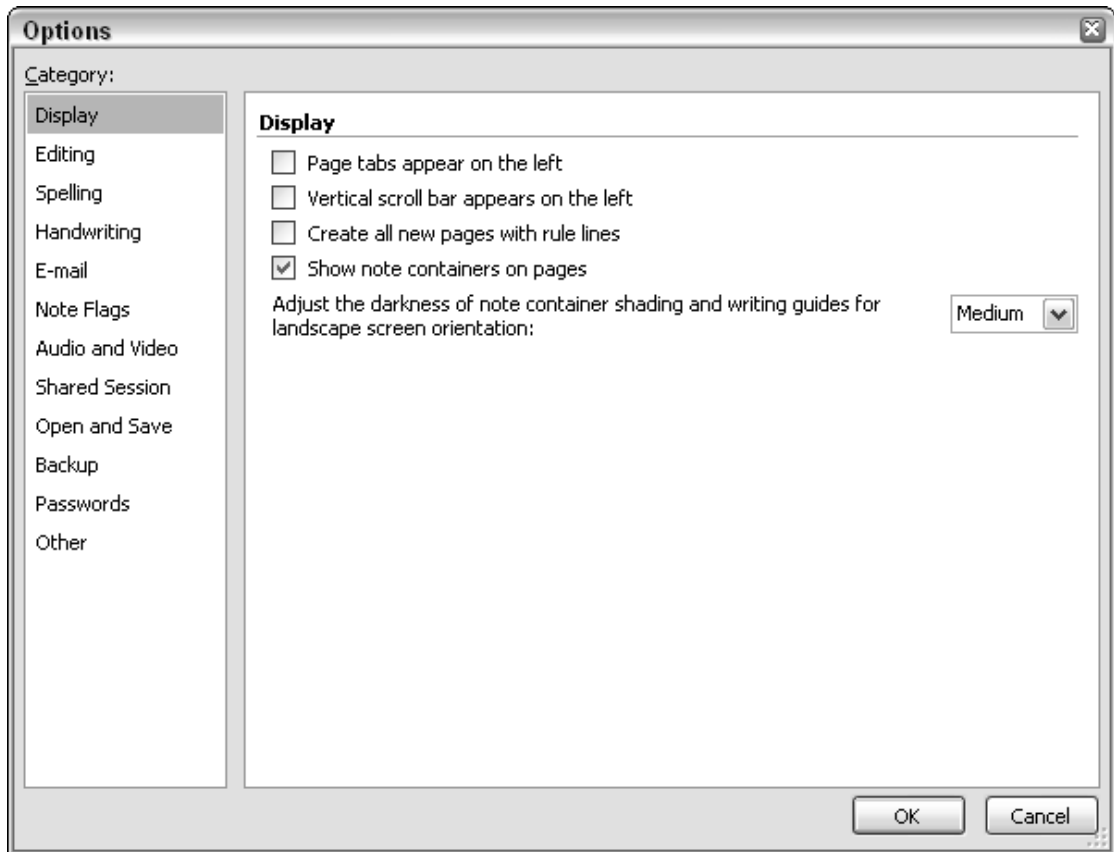
هذا البرنامج لديه العديد من القوائم والأزرار (إنظر الشكل)



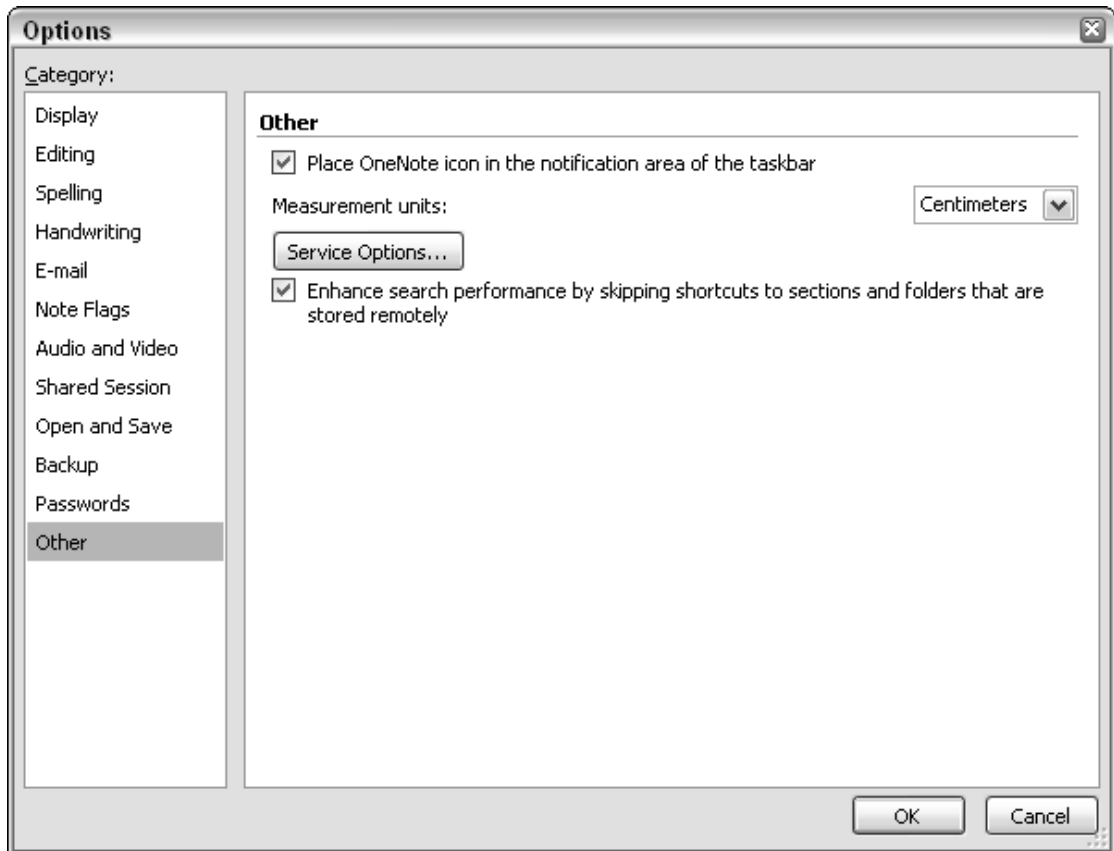
الأزرار هنا تعتمد على الصور , وعندما تتحرك بالفأرة فوقهم فهناك نص ينبثق (يسمى tooltip) يعطيك وصف مختصر على ما تفعله الأداة . (إنظر الشكل)



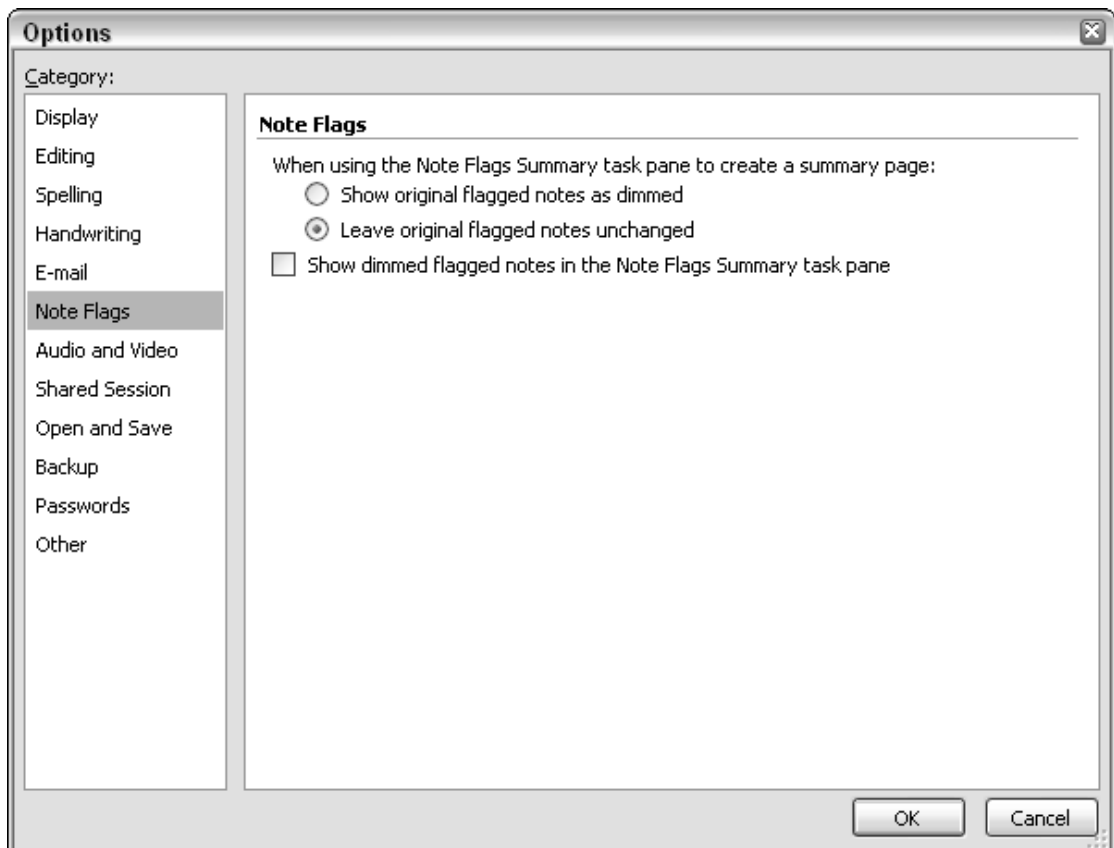
الضغط على الأدوات في القوائم ومن ثم Options ... يحضر لك نافذة Options (إنظر الشكل)



فيها أزرار Buttons ومربعات إختيار Check box لك لتختار الإعدادات التي تريد (إنظر الشكل)



بعض هذه الوظائف أيضاً يتيح لك استخدام Radio buttons , تستخدمها حيث يمكنك فقط تحديد إختيار واحد للبعض (إنظر الشكل)



Summary الملخص

فى هذا الفصل , لم تتعرض فقط إلى ممارسات كتابة كود ولكن قدمت لك مفاهيم واجهة المستخدم . وذلك فى كيفية رؤية المستخدم لبرنامجك , وكمبرمج فجزء من وظيفتك أن تجعل البرنامج أسهل للآخرين ليستخدموه كلما امكن . تذكر , أنت تعرف كيف يعمل البرنامج — الآخرين لا يعرفون .

لقد بحثنا فى تفصيل كيف يمكن أن تضيف واجهة بسيطة إلى تطبيقات C++ وكيف تصنع نظام معتمد على النص للمستخدم النهائى , بإضافة تأجيلات إدخال وشاشات تأكيد .

أيضاً رأينا طرق بسيطة لإضافة نظام مساعدة إلى تطبيقات C++ .

فى نهاية الفصل , إلقينا نظرة على العناصر لواجهة المستخدم التى هى نموذج جزء من الواجهة الرسومية للـ windows . اخذت جولة على العناصر وبحثنا فى كيف يمكن ان تستخدمها فى التطبيق لجعل خبرة المستخدم ممتعة وسهلة كلما امكن . الآن ماهى العناصر وكيف يتم إستخدامهم , ستلاحظ إستخدامهم فى تطبيقات أخرى لذا عندما تأتى لأستخدامهم بنفسك فى تطبيقات فى المستقبل , ستكون قادراً على إستخدامهم بإحتراف بحق من البداية .

11

Putting It All Together

وضعهم معاً

هذا الفصل يتكلم عن التخطيط والتنفيذ للمشروع البرمجي . من البداية للنهاية . ليس هناك أى كود فى هذا الفصل . بدلاً من ذلك نجد التركيز على التركيب , التخطيط , التنفيذ للمشروع البرمجي .

هذا الفصل يفترض أنك إتبعنا هذا الكتاب من البداية حتى هذا الفصل وإحتمالاً أنك أيضاً قرأت كتاباً أو اخذت دورة فى لغة البرمجة التى تهتم بها .

لا يوجد شئ أفضل لتختبر عضلاتك البرمجية أكثر من العمل على تحديات ومشاريع معقدة , وهذا الفصل يساعدك لتفعل هذا .

Planning a Programming Project

تخطيط مشروع برمجي

عندما تواجه مشروع برمجي , تجد ان معظم الناس يعتقدون ان العمل بأكمله يتم خلف لوحة المفاتيح فى كتابة الكود , ربما يكون هذا جيداً لبعض المشاريع الصغيرة , منذ ان يشتمل المشروع على واحدة أو اثنين أو ثلاثة من المزايا فإن التخطيط وقتها هو مفتاح النجاح للمشروع .

Without Planning

بدون تخطيط

دعنا نلقى نظرة سريعة على ما يحدث للمشروعات الغير مخطط لها أو أن المخطط غير صحيح . أمل من هذا أن تحصل على التحفيز لتخطط لمشروعك ولترى أيضاً بعض المساحات التى تحتاج إلى تركيز أكثر من غيرها عندما تاتى لتفصيل التخطيط .

More Code, Less Features

المزيد من الكود , القليل من المزايا

المشاريع الغير مخطط لها أو التخطيط الغير جيد عادةً ما يكون ينتج عنه كتابة الكود بطريقة غير فعالة. الكود الغير فعال في العام يعنى كتابة المزيد من الكود لأداء وظيفة ما أكثر مما هو مطلوب . فإذا كان كتابة الكود مخطط لها. سبب أن الكود يصبح ممتلئ بالغرور هو أنه بينما الكو يقوم بتنفيذ العمل لميزات خاصة خلال مشروعك فلن يكون أقل كفاءة . إطار العمل يصبح غير متقن كما قمت بإضافة ميزات لمشروعك بطريقة عشوائى , موضة خاصة .

السبب الآخر وراء الكود الغير متقن فى المشاريع الغير مخطط لها هو ان المبرمج يكتب دوال مضاعفة لتنفيذ مهام مشابه أو محدده .

إضافة إلى أن الكود الغير ضرورى يعتبر أيضاً عائقاً للتصحيح الفعال للكود .

More Bugs

المزيد من الشوائب

المشاريع المخطط لها بطريقة سيئة عادةً ينتج عنها تطبيقات سيئة لأن عدد عظيم من الشوائب المتنوعة فان الكود يتراكم على طول الطريق , الدوال للمزايا الخاصة كتبت ونشأت عشوائياً , والأخطاء التى عادةً تنشأ نتيجة من مجموعة واحدة من الدوال المتعلقة بميزة واحدة متعارضة مع مجموعة دوال أخرى كتبت لمزايا أخرى .

التخطيط المناسب يزيل هذه المشاكل لأنك تعرف مقدماً ما هى المزايا المطلوبة , ويمكنك ان تخطط كيف يتم التفاعل مع بعضهم البعض .

Project Takes Longer

المشروع يأخذ وقتاً طويلاً

إذا كنت تقاتل مع المزيد من الشوائب والمزيد من الكود بسبب التخطيط السيئ , فالكود الخاص بك سيأخذ وقتاً أطول .

عادةً , المشروع الغير مخطط له سيأخذ بصورة ملحوظة أطول ليتم إتمامه من البرنامج المخطط له مقارنة بالحجم وبساطة التعقيد . لأنك ستبرمج مزايا منفصلة مستقلة عن المشروع ككل ولا تخطط كيف يتم دمجهم من البداية .

سبب آخر فى لماذا المشاريع الغير مخطط لها تأخذ اطول هو انها تعاني من زحف المزايا الغير ضرورية . Feature creep يحدث عند إضافة مزايا إضافية إلى المشروع فى وسط الكود . يمكن ان يكون إستخدام غير ضرورى من الوقت والجهد إذا كان السبب الوحيد الذى يحتاجه هو إضافتهم .لأنهم يتم التغافل عنهم بسبب عدم وجود تخطيط .

Missing Features

مزايا مفقودة

جنباً إلى جنب مع زحف المزايا الغير ضرورية تأتي مزايا مفقودة — المزايا التي تريد ان تشتمل عليها او نسيت أن تشتمل عليها بسبب أنك لا تمتلك خطة تسير عليها . يمكن أن يكون هذا مشكلة ضخمة , خاصة إذا لم تجد عن المزايا المفقودة حتى تدخل مرحلة الإختبار , ولأن هذا يعنى انه سيكون عليك أن ترجع أو تعود لمرحلة كتابة الكود وإرجاء الاختبار حتى إضافة المزايا الإضافية . إما أن , او يمكنك تركها .

Planning

التخطيط

الآن يمكنك ان ترى بعض العقبات للمشروع فيها في المشروع التي لم يكن من المحتمل أن يخطط لها, دعنا نلقى نظرة على كيف تخطط لمشروع وإنشاء تخطيط مناسب ليساعدك لتحصل على أفضل ما في برمجتك .

دعنا الآن نمر خلال الخطوات و مراحل التخطيط لمشروع البرمجة .

The Idea

الفكرة

يبدأ المشروع البرمجي من فكرة . البرمجة حزئياً هي مهارة إبداعية — فتشارك في إنشاء شيء ما يفعل شيء ما .

الفكرة يمكن ان تكون احد الأنواع الكثيرة :

- **التحسين** : - ترى شيء ما موجود بالفعل , ويمكنك أن ترى كيف يمكن أن ينجز أفضل بكثير أو أكثر تسهلاً أو في وقت أقل أو في خطوات أقل .
- **فكرة جديدة** : ترى فجوة حيث يريد الناس حل مشكلة ولا يوجد شيء متاح حالياً ليساعدهم (او لو كانت هناك, مرتفعة الثمن) . ترى طريقة لتغيير هذا .
- **فجوة التخصص/ السوق** : ترى فجوة تحتاج إلى ملئها في السوق , وترى طريقة لفعلها . عامة فكرة كهذا تبدأ الحياة كتحسين أو فكرة جديدة , ولكن مع مرور الوقت تأتي إلى التفكير الفعلي عن برمجة حل الذي لم يعد فكرة جديدة .

اول شيء ينبغي أن تفعله مع فكرة أن توثقها لأنك بالتأكيد يمكن أن يحدث لك شيء من إثنين :

- أن تنسى . يقال أن كل شخص يأتى على الأقل بفكرة واحدة في السنة يمكن أن تجعله مليونير ولكن لا يضعونها في الفعل — عادةً لأنهم ينسونها !
- كما تفكلا في فكرتك ومن المحتمل أن تناقشها مع احد الآخرين , الفكرة الإبتدائية ستتغير ومن الممكن في هذه المرحلة في العملية للجزء الجيد من الفكرة الأصلية لتفقد .

Documenting the Idea

توثيق الفكرة

وفيما يلي قائمة مراجعة بسيطة لكنها فعالة تساعدك أن توثق وتعد أفكار مشروعك الإبتدائية . سيساعدك هذا أن تسجل فكرة الإبتدائية لذا أيضاً يمكن أن تعمل عليها وتناقشها بدون أن تفقد جوانب حيوية من المشكلة .

قائمة مراجعة مبدئية

التاريخ :

الوقت :

بعبارات عامة , ما هي فكرتك ؟

.....
.....
.....

ما هي المشكلة الرئيسية التي يقابلها البرنامج أو ماذا تحتاج ان تقوم بملئه ؟

.....
.....
.....

هل ستكون معالجة البيانات , أو تمتة مهمة ؟

.....
.....
.....

هل هذا تخصص يحتاج ان تقى به او أنه ينطبق على كثير من الناس ؟ إذا كان يحتاج تخصص , من سيجده مفيداً ؟ وإذا لم يكن لتخصص , فصل من ربما يجده مفيداً ؟

.....
.....
.....

هل هناك إحتياجات ثانوية تعتقد أن البرنامج سيقابلها ؟

.....
.....

هل هناك شئ خاص جعلك تأتى بالفكرة ودفع ذلك فى المقام الأول ؟ إذا كان نعم , ما هو دافع الفكرة؟

.....
.....
.....

دون المزاي العديدة لفكرتك التي يمكنك فعلها :

.....
.....
.....
.....
.....

ما الذى يجعل فكرتك مختلفة / أفضل من البرامج الموجودة ؟

.....
.....
.....

هل هناك أى جوانب من البرنامج انت غير واضحاً بها ؟

.....
.....
.....

هل تفكر فى ان يكون البرنامج سيكون تجارى ام مجانى ؟
مجانى / تجارى

لماذا اخترت مجانى أو تجارى ؟

.....
.....
.....

إذا كان تجارى , كم الثمن الذى تعتقد أن يدفعه الناس لشرائه , ولماذا ؟

.....
.....
.....

كيف تفكر فى كيفية توزيعك للبرنامج ؟
على Internet / DVD / CD / Floppy

هل تنوى أن توفر مساعدة ودعم للبرنامج إذا انت قمت بإطلاقه ؟

.....

ما نوع الدعم ؟
بريد إلكترونى / موقع أو منتدى / التليفون

بعد إكمال هذه القائمة , ينبغى أن تسجل فكرتك الابتدائية أو المبدئية التى يمكن أن تعيد النظر فيما بعد إليها
لتنعش الذاكرة . وعندما تأتى للتعديل , إستخدم القائمة التالية لتدون ملاحظاتك للتغيرات , وكن متأكداً أن تقوم
بتخزينها مع الأولى

قائمة التغيرات // الإضافات

التاريخ :

الوقت :

ما هى التغيرات / الإضافات التى تخطط لها ؟

.....
.....
.....

ما أثر ذلك على فكرتك الابتدائية ؟

.....
.....
.....

Maturing Time

منذ أن توثق فكرتك الابتدائية , قاوم الإغراء لتشغيل الحاسب لبداية كتابة الكود إذا لم يكن المشروع بسيطاً تتعده . دع الفكرة قائمة معك لفترة (على الأقل يوم) وبحث ذلك مراراً . خلال هذا الوقت , فكر في الفكرة ككل ولتري إذا مازلت تفكر انها فكرة جيدة وما إذا كانت المنافع التي وضعتها في القائمة الابتدائية مازالت مطبقة . إحياناً الفكرة التي تبدو مبدئياً جيدة لا تبدو بنفس الجودة حين تعيد النظر فيها في وقت لاحق .

إستخدم الوقت للقيام للعض البحث , تحقق بمعرفة ما إذا كان هناك برنامج يقوم بما تخطط له . فإذا وجدت البعض , فابحث ما إذا كان يقوم باداء جيد أفضل من الذي يمكن ان تقوم أنت به .

- كم يتضح لك لتنفيذه في ما وضعت في القائمة الابتدائية في البرنامج الذي وجدته ؟
- كم يكلف ؟
- هل يبدو أنه واسع الاستخدام ؟
- هل يبدو أنه أفضل من فكرتك ؟

بمجرد أن تكمل البحث , إنظر كيف يغير هذا من خطتك المبدئية . إلى هذه المرحلة , من المحتمل أن يكون لديك شعور بواحد من ثلاث :

- إنك لم تجد في برنامج يقابله و ستتعهد بتنفيذه .
- انك وجدت برنامج يفعل ما خطط له , ولكن أكثر إرتفاعاً في الثمن / لديه مزايا أقل /ليس جيداً كما خطط له أن يكون .
- أنك قررت أن فكرتك ليست جديرة بالمتابعة .

The Requirements

المتطلبات

إذا قررت أن تتابع المشروع , إذاً فالمهمة التالية لك أن تفعلها هو تفصيل المتطلبات للمشروع . المتطلبات للمشروع تفصل وصف ما تريد أن يفعله المنتج في النهاية .

من الهام في هذه المرحلة أن تكون واضحاً ودقيق على قدر الإمكان . إلى هذه المرحلة لا تكن قلق جداً على "كيف" سيعمل البرنامج , فقد ركز على " ماذا " سيقوم بفعله .

إليك بعض الأسئلة لتساعدك أن تحدد مجموعة من المتطلبات :

Requirements

المتطلبات

التاريخ :

الوقت :

ما هي الوظيفة الابتدائية التي يقوم بها التطبيق ؟

.....
.....
.....

ما هي الوظيفة الثانوية التي يقوم بها التطبيق ؟ مرة أخرى, كن واضحاً قدر ما تستطيع .

.....
.....
.....

هل الوظيفة الابتدائية والثانوية سيضملمان داخل تطبيق واحد أم يقسمان إلى تطبيقات منفصلة ؟ كن واضحاً قدر ما تستطيع .

.....
.....
.....

إذا كان التطبيق متورط في معالجة بيانات , من أين ستأتي البيانات ؟(على سبيل المثال , هل ستدخل من لوحة المفاتيح أو بمعالجة ملف ؟)

.....
.....
.....

ما شكل البيانات التي يأخذها للتجهيزها ؟

.....
.....
.....

إذا كان التطبيق سيقوم بمعالجة بيانات , قم بتقسيم خطوات المعالجة إلى أجزائها التكوينية .

.....
.....
.....

ما هو الشكل الذي تأخذه البيانات المعالجة (بيانات المخرجات) ؟

.....
.....
.....

كم يحتاج المستخدم النهائي للتفاعل مع البرنامج ؟

.....
.....
.....

ما نوع الواجهة التي يحتاجها برنامجك ؟ هل ستكون بدائية جداً (Command Line) أو احدث وبها إستخدام للأزرار , صناديق إختيارات وقوائم , وعناصر واجهة متقدمة أخرى ؟

.....
.....
.....

لو نعم , أى العناصر التي ترى أن تستخدم ؟

.....
.....
.....

كيف ستعطى المستخدم كيفية إستخدام البرنامج ؟ (لمحة جيدة هنا أن تفكر فى كيفية توفير البرامج الأخرى لكيفية الإستخدام والخلفية للمستخدم)

.....
.....
.....

هل يحتاج التطبيق إلى حفظ ملفات ؟
نعم / لا

لو نعم , أى نوع من الملفات , وماذا ستحتوى هذه الملفات ؟

.....
.....
.....

هل يحتاج برنامجك أن يتفاعل مع برامج أخرى ؟
نعم / لا

لو نعم , أى واحد ؟

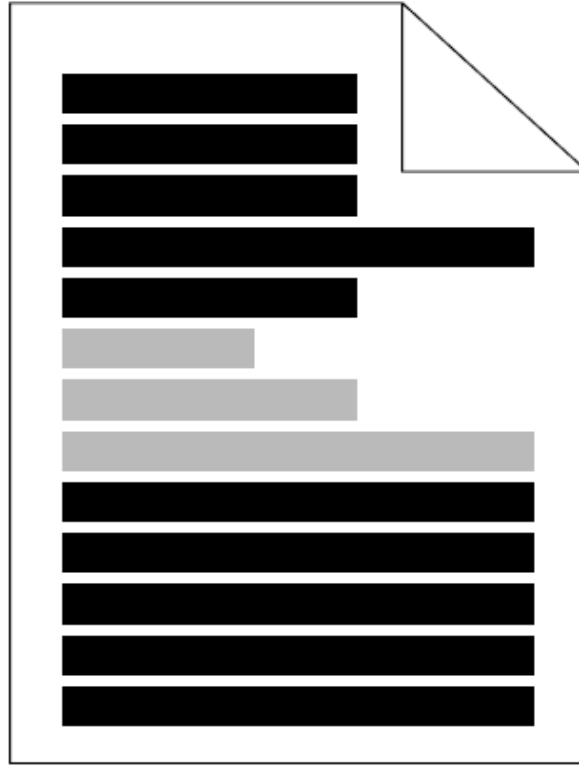
.....
.....
.....

هل يحتاج برنامجك أن يطبع ؟
نعم / لا

لو نعم , ماذا ؟

.....
.....
.....

مجرد ان تضع المتطلبات فيكون لديك خريطة بتفاصيل الوظيفة والعمل الذي سينفذها مشروع برنامجك . إلى هذه المرحلة , لديك فكرة جيدة لكيف سيعمل التطبيق وماذا تحتاج الوظائف للتنفيذ . إذا كنت غير واضح عن نواحي أكيدة من المشروع (على سبيل المثال , تعرف أن البرنامج سيحتاج ان يعالج البيانات ولكن حتى الآن غير متأكد من الخطوات التي ستأخذها العملية للتنفيذ , يمكنك الآن تركز الوقت والطاقة لتعبد البحث في هذا في مزيد من التفصيل . هذا مصور في الشكل



Some gray areas are inevitable early on in a project

لست بالضرورة في حاجة إلى خطوات المعالجة مفصلة حتى تبدأ كتابة الكود للبرنامج . وهذا لأنه أكثر المحتمل أن تكون دالة أو قبضة من الدوال يتحكم في العملية . بينما تظل تعمل معهم , يمكنك ان تكتب دوال لا تفعل شيئاً أو تنفذ بعض المهام البسيطة على البيانات . بالطبع , قبل أن تكمل كتابة الكود فتحتاج كل المراحل مفصلة بدرجة كافية لك أيضاً يمكنك تحويل الخطوات إلى تعليمات كود , ولكن إقبل إلى الآن أن هناك مساحات باهتة لا مفر وسيكون عليك العودة لملئها بتفصيل في وقت لا حق .

Modifying the Requirements

تعديل المتطلبات

كمجرد فكرة مبدئية فهي تنضج وتتغير , لذا ستضع المتطلبات المبدئية للمشروع . ربما تقرر أن :

- إضافة مزايا ومتطلبات جديدة .
- تعديل المتطلبات الموجودة مسبقاً .
- حذف المتطلبات .
- تأجيل متطلبات إلى إصدارات المستقبل .

بينما تكون بعض التغييرات متوقعة , كن حذراً عند إجراء تغييرات كبيرة — ربما ينتهي بك الحال إلى برنامج جديد أو برنامج آخر مختلف إختلافاً جذرياً .

Choosing the Language

إختيار اللغة

قم بإختيار اللغة التى تحتاجها لإحضار المشروع من الناحية النظرية إلى الواقع . فالمشروع البسيط تناسبه لغة برمجة كتابيه مثال JavaScript أو VBScript , ولكن أى شئ معقد سيتطلب منك إستخدام لغات بها المزيد من التطور كاللغات المتنوعة مثل Java أو C++ , فى رأيك أن التطبيق سيعتمد بقوة على واجهة مستخدم محترفة , إذاً فربما تحتاج إلى لغة "مرئية " مثال visual Basic .

ستحدد مستويات خبراتكم كيف يمكنك المضى قدوماً الآن . إسأل نفسك إذا كان لديك المهارات الضرورية لتنفيذ مشروع ما أو إذا كنت تعتقد أنك تحتاج إلى ترقية مهاراتك . فينبغى من تعلم شئ جديد من خلال دورة تدريبية فى البرمجة . ولكن إذا شعرت أنك تحتاج مهارات هامة تعزيزية فربما عليك أن تأجل المشروع لبعض الوقت

Programming Stage

مرحلة البرمجة

لديك فكرتك موثقة , ولديك التغيرات والتعديلات للفكرة , وقد وثقت وفصلت متطلبات المشروع ووصفتم . وألقيت نظرة على مفتاح بداية العمليات خلال الفكرة ووزعت خطوات المعالجة . بعد إتمام كل هذا فهذا وقت جيد لتبدأ البرمجة .

Programming the Basics

برمجة الأساسيات

دوماً يبدأ مشروع البرمجة بالعمل على أساسيات التطبيق أولاً . المكان الأفضل لتبدأ هو إضافة بعض التركيبات إلى المشروع . ربما تكون المساحة الجيدة التى تركز عليها مبدئياً هى :

- كتابة كود مرتبط بما يحدث عندما يبدأ التطبيق فى بدء التشغيل لدى المستخدم النهائى .
- أى مطالبات أو تعليمات تعطى من التطبيق إلى المستخدم النهائى .
- كتابة كود مرتبط بما يحدث عند إغلاق التطبيق , بواسطة المستخدم النهائى .
- كتابة هذا النوع من الكود الذى يسمح لك بتطوير الهيكل العام لبيئة العمل للكود الذى يمكن إرفاق المزيد من الكود والإستمرار فى تطوير التطبيق .

Testing

الإختبار

إختبر الكود بعد كل خطوة تخطوها , لأنك ستعمل على جوانب محددة من البرنامج , فينبغى عليك تشغيل أجزاء من الكود منفصلة عن الكل (كما فعلنا فى الفصل السابق) سيساعدك هذا فى إعطائك الثقة بأنك تحرز تقدم وأن الكود يعمل .

إفحص أى مشكلة تمر بها فى الكود . فهذا لن يقلل عملية التصحيح فى نهاية المشروع فقط , ولكن سيمسح لك أيضاً بأن تستمر فى عملية كتابة الكود وإختبار كود آخر تقوم بكتابته .

وإذا مررت بشائبة وإخترت أن تتركها حتى نهاية المشروع , فقم بتفصيل ملاحظة لها (خاصةً إذا كانت أخطاء فى معالجة البيانات أو عملية أخرى) وهذا لأنك بلا شك لن تتذكرهم عندما يأتى الوقت لفحصهم .

Error Documenting

توثيق الأخطاء

التاريخ :

الوقت :

صف الخطأ الذى قابلته , كن واضحاً قدر الإمكان وصف ما تأثيره

.....
.....
.....

أى مقطع من الكود تعتقد انه يسبب مشكلة ؟

.....
.....
.....

لماذا قررت انك لن تصلحه فى الوقت الحالى ؟

.....
.....
.....

Commenting Code

كتابة تعليق على الكود

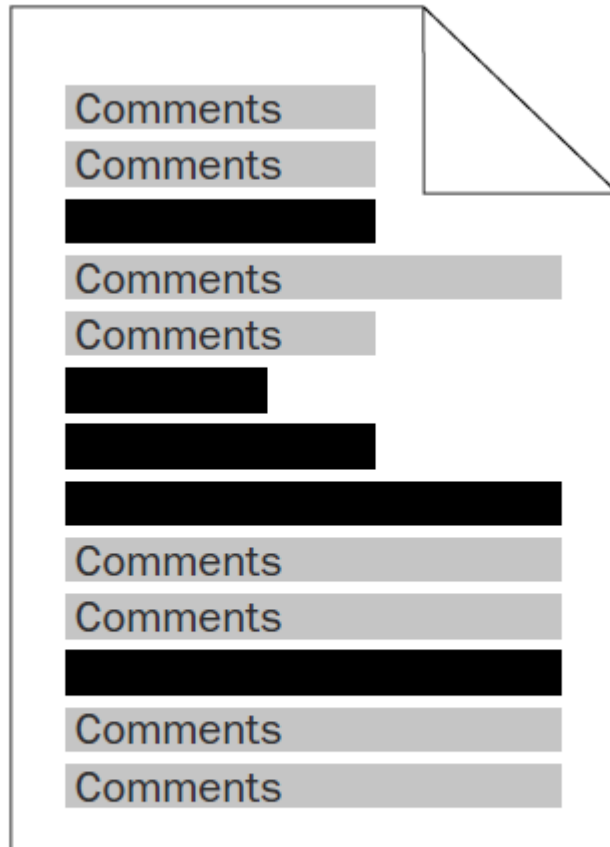
شئ آخر ينبغى ان تتذكر ان تفعله هو ان تكتب تعليق على الكود بشكل جيد (كما هو مبين بالشكل) إفعل هذا لكل الدوال , المتغيرات , وحتى كل كتلة من الدوال , وإفعل هذا كلما كتبت كود .

لا تنخدع فى التفكير فى أنك لا تحتاج إلى كتابة تعليق , ترك تعليق على الكود حتى إن كان الكود واضحاً لك لأنك من كتبته . ربما يكون واضحاً الآن , ولكن بعد أيام قلائل او أسابيع ستحطم رأسك عجباً فيما يقوم هذا الكود بفعله .



Comment your code regularly and thoroughly

أحد الأشياء التي يقلق منها المبتدئون للبرمجة هي إدراج المزيد من التعليقات على الكود (كما هو مبين في الشكل) . لا تقلق لهذا كله — ستكون بكتابة تعليقات يمكن تحريرها فيما بعد أفضل بكثير من عدم كتابتك تعليقات كافية وفقد الأثر للمشروع .
لقد رأيت مبتدئين يكتبون ثلاثة , أربعة , وخمسة أسطر من التعليقات لسطر أو سطرين من الكود ولا شيء خاطئ في هذا مطلقاً .



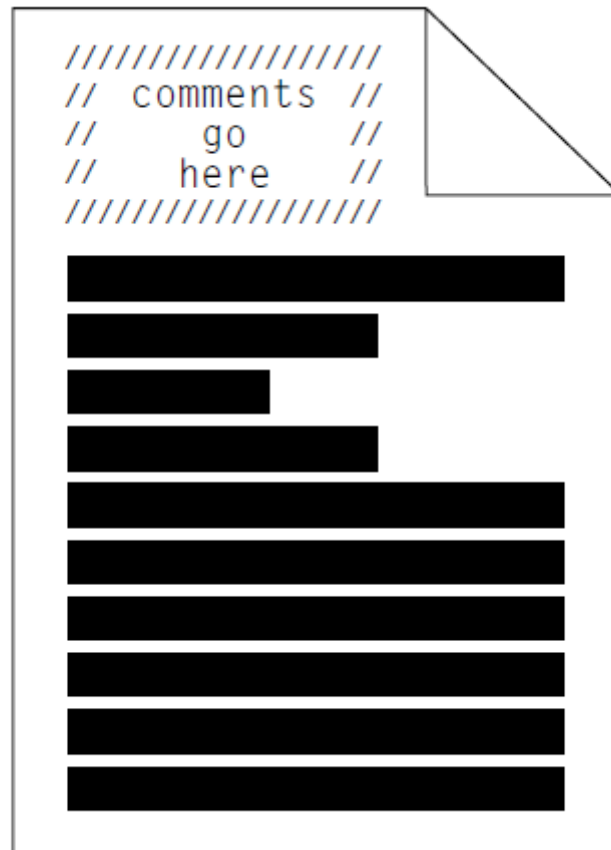
Don't worry if initially comments feel like they outweigh code sometimes –that's normal!

Commenting Tips

تلميحات عن التعليقات

إليك تلميحات لتحصل على تعليقات فعالة .

- في بداية الكود , أضف تعليق يفصل المشروع ككل (إنظر التبيين في الشكل) . إلى هذا أضف الاسم والتاريخ . تعليقات أعلى الصفحة شائعة وتعرف **tombstone comments** بسبب الطريقة التي يتم ظهورها بالصفحة .



- فصل ما تفعله كل دالة . أفعل هذا فى الإنجليزية السهلة لذا فلا ينبغي عليك ان تفك شفرة البرمجة لتفهم الكود .

التعليق الجيد : هذه الدالة تأخذ إجمالى السعر النهائى وتضيف ضريبة المبيعات إن وجدت .
التعليق السيئ : هذه الدالة تنفذ العملية المدرجة بالأسفل . السطر السادس صعب !

- فصل ما يعنى كل متغير إذا لم يتغير تفصيله من قبل .

التعليق الجيد : المتغير x هو إجمالى القيمة قبل الضريبة , y هو إجمالى القيمة بعد الضريبة .
التعليق السيئ : المتغيران x, y يشيران إلى إجمالى القيمة .

- وثق كتل الدوال .

التعليق الجيد : الثلاث دوال التالية على إجمالى القيمة على الترتيب, إضافة الضريبة و حساب تكلفة التسليم .
التعليق السيئ : إجمالى القيمة , الضريبة و تسليم البضائع .

Testing

الإختبار

تهانينا !

لقد وصلت إلى علامة فارقة هامة — فالمشروع أصبح جاهزاً ليتم إختباره من جهة الآخرين ! هنئ نفسك في نهاية الأمر وأشعر بالفخر لإنجازاتك , لا يهم كم كان كبير أو صغر المشروع الذى قمت به .

لا تترك مرحلة الإختبار حتى يتم تكملة المشروع . خطط أن تنفذ جلسات إختبارات متعددة خلال عملية البرمجة . سيساعدك هذا للتأكد من أنك فى المسار الصحيح ولا تساهم فى بناء مشاكل فيما بعد .

الإختبار ينبغى أن يعتمد على الخطوات التالية :

- إختبار الأخطاء الفعلية للكود التى تسببت فى وجود المشاكل . هذا بالفعل إختبار للكود لنرى إما أن البرنامج يعمل كما هو متوقع ولا توجد أى شوائب مع الكود تسبب تصدع أو ظهور أخطاء أخرى .
- إختبار الأخطاء المنطقية فى الكود . هذه الأخطاء هى الأصعب فى إكتشافها . تحدث الأخطاء المنطقية عند تركيب منطق الكود بشكل خاطئ . مثال مبسط على هذا : عند توقع أن يجمع الكود رقمين معاً ولكن بدلاً من ذلك يطرحهم — هذا هو الخطأ المنطقى , إذا كانت المسألة المنطقية بسيطة , إذا فهذه الأنواع من الأخطاء يمكن من السهل إكتشافها , ولكن كلما تعقدت المسألة المنطقية , فيكون من الصعب إكتشافها وكذلك يصعب تصحيحها . فلا تقلل من شأن الأثر المدمر لهذا النوع من الخطأ الذى يمكن ان يكون فى التطبيق — وتقلل من صعوبة إمكانية إكتشافها . هناك حالات عدة حيث يمكن للتطبيق التجارى (مثل الذى يشارك فى الفواتير) أن يحتوى على خطأ يكلف العميل أو الشركة المزيد .

قم بإختبار بالمدخلات المناسبة وبعدها مروراً بفحص المخرجات لتكون فى أفضل طريقة لفحص هذه الأنواع من الأخطاء .

للمشروعات الكبيرة , بعض المبرمجين يبنون إختبارات التطبيقات التى ترسل المدخلات إلى البرنامج الرئيسى لفحص المخرجات ليرى ما إذا كان ما يتوقعوا . هذا النظام يقلل المجال لوجود أخطاء منطقية .

- فحص أخطاء التصميم , هناك مشاكل فى طريقة عمل البرنامج . يظهر هذا بوضوح عند إستخدام وإختبار التطبيق . قم بتدوين ملاحظات مفصلة لأى مشكلة تواجهها لذا يمكنك ان تراجعهم فيما بعد .
- المزايا المفقودة . لا يهم مدى توثيقك للفكرة فمن المحتمل ان تجد أنك فقدت شئ ما فى مكان ما خلال الدورة البرمجية . بدون قلق , يمكنك ان تعود للوراء وتقوم بإضافتها . مرة أخرى فقط دون ملاحظتك لما تتركه ولن تنساه مرة أخرى .

The Route to Better Testing

الطريق إلى أفضل إختبار

كيفية إختبارك تكاد تكون بنفس أهمية الأمر الذي تقوم بالإختبار من أجله. فهناك قاعدة عامة تقول أن المبرمج أسوأ شخص يمكن ان يختبر التطبيق (خصوصاً التطبيقات التي قام بتطويرها). وهذا لأن المبرمج لديه فكرة جيدة في كيف ينبغي أن يعمل البرنامج. و ما الأشياء التي يفعلها , وأى المدخلات التي يتوقعها البرنامج .

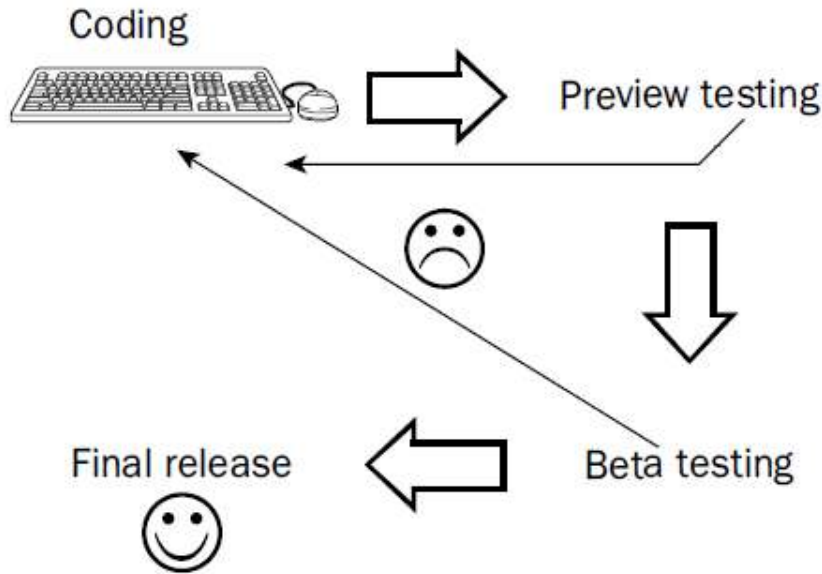
فالطريقة الأفضل بكثير من إختبار التطبيق بنفسك هي أن تدع الآخرين يختبروه لك بإنشاء برنامج معاينة لك وأخر تجربة .

ربما قد سمعت عن المصطلحات program preview أو Beta release من قبل (من المحتمل على الإنترنت او في مجلات الحاسب) , وإذا لم تسمع بهم , دعنى أشرح لك ماذا نعنى بكلاً منها .

Preview release هي فقط — معاينة . مع إصدار المعاينة تدع الآخرين يرون برنامجك قبل إستعداده بالكامل . وفي مقابل السماح لهم بالمعاينة , تسألهم عن ردود الافعال عن البرنامج . يمكن أن يقترب البرنامج من النهاية أو لم يقترب من الإنتهاء . فالفكرة ليست في انك تعطى الآخرين برنامجك بصورته الكاملة , ولكن بدلاً من ذلك تدعهم يرون التقدم الذي تصنعه في البرنامج . وإلى هذه المرحلة ينبغي أن تفتح تقرير للشوائب والمقترحات لأى مزايا و تغيرات جديدة .

Beta release تأتي عند إطلاق برنامج تعتقد أنه تام إلى حد كبير . ولكن أنت تريد جمهور أوسع لإختبار البرنامج وإعطاء رد الفعل . خاصةً تقارير الشوائب والمشاكل . إلى هذه المرحلة ينبغي أن تستمر في تقبل رد الفعل عن قضايا أخرى مثال ذلك التحسينات أو المزايا الجديدة , ولكن من المحتمل وربما الأفضل أن تترك هذا إلى إصدار المستقبل (إذا لم تكن مقنعة حقاً)

العملية تم تبينها في الشكل



But What About . . .

ولكن ماذا عن

انت تعمل على بعض البرامج الشبقة وتريد ردود الفعل عليها . لذا فقد أطلقتها للآخرين — ولكن بمجرد أن يحصلوا على البرنامج لماذا ينبغي عليهم ان يعودوا مرة أخرى ؟ وإذا كنت تخطط أن تبيع البرنامج , فلماذا ينبغي لك أن تعطيه بلا مقابل . في Beta أو Preview ؟

الإجابة عن هذه الأسئلة تعتمد على ما إذا كنت تخطط لبيع البرنامج أو إعطائه بلا مقابل .

Beta and Preview Strategy for Free Software

خطة البرنامج المجاني

إذا كنت تخطط أن تطلق البرنامج بالمجان او للتبرع , إذاً فلا تحتاج أن تقلق على النسخة بنوعيتها فلن تؤثر على الإصدارات اللاحقة . في الحقيقة , سيحرص الآخريين على العودة مرة أخرى للحصول على تحديثات أو آخر إصدار .

Beta and Preview Strategy for Commercial Software

خطة البرامج التجارية

مع البرنامج الذى تخطط أن تبيعه فى آخر الأمر, عليك أن تقوم بفعل أشياء جزئية قليلة مختلفة لإيقاف الأشخاص من الإستمرار فى إستخدام نسخة المعاينة أو التجربة ولا يدفعون لك ليستمروا فى الإستخدام . إليك بعض الخطط التى يمكن أن تساعدك للتأكد من أن مرحلة الإختبار لا تؤثر فى المبيعات .

- ضع حدود على بعض المزايا للبرنامج — قم بحذف مكونات حيوية مثال إمكانية حفظ او تصدير ملفات.
- ضع مزايا مزعجة فى البرنامج (مثال شاشة تدمر أو عداد تنازلى يؤجل عمله مع البرنامج لبعض الثوانى)
- حدد وقت للبرنامج فيتوقف عن العمل بعد وقت محدد او بعد مرات معينة من التشغيل .
- قيد الإختبارات المبدئية لتحديد مجموعة من المستخدمين .
- إذا كان البرنامج ترقية , ضع قيود على الإصدار لمن يستخدمون برنامجك بالفعل.
- ضع تنبيه فى البرنامج يذكر المستخدم بانه يستخدم نسخة معاينة أو نسخة تجريبية .

بإستخدام مثل هذا المخطط الذى قد فصلنا , ستكون قادراً على الحصول على مميزات أكثر لإصداراتك وتحصل على ردود الإفعال وتقارير الشوائب من الآخرين ولا يؤثر ذلك على المبيعات .

Questions for Those Previewing Software

الأسئلة لبرنامج المعاينة

إليك قائمة بالأسئلة التى أقترح أن تسألها للأشخاص ممن يوافقون على مراجعة وإختبار برنامجك .

بعض الأشخاص ربما لا يكونوا قادرين على الإجابة على كل الأسئلة المعروضة , ولكن لا أحد سيجيب إذا لم تسألهم فى المقام الأول .

PC System Questions

أسئلة نظام الحاسب الشخصي

نوع الحاسب .
وحدة المعالجة المركزية / المعالج [البروسيسور]
ذاكرة المرور العشوائية [الرام]
المساحة الفارغة على القرص الصلب
نظام التشغيل (أدرج أى نسخ حقب خدمية مثبتة)

Problems

المشاكل

هل واجهت مشاكل ؟
لوكانت الإجابة بنعم ، صف المشكلة فى تفاصيل أكثر إن أمكن .
هل تستخدم أى تطبيقات أخرى فى نفس الوقت .

Usability

كم مدى سهولة البرنامج فى الإستخدام ؟
هل وجدت مساحات مزعجة ؟
لو نعم ، لماذا ؟
كيف تعتقد أن يتم تحسينها ؟

Suggestions

الإقتراحات

هل يمكن أن تحسن البرنامج ؟
أى المزايا الإضافية التى ستضيفها ؟
هل هناك أى مزايا تود حذفها ؟

General questions

هل وجدت البرنامج مفيد ؟
هل ستشتري البرنامج ؟
لو نعم ، كم تود أن تدفع فيه ؟
لو لا ، لماذا ؟
إذا لم تكن متأكداً ، لماذا ؟

Additional Features

مزايا إضافية

إذا كنت فى مرحلة المعاينة ، يمكن الآن أن تفكر فى المزايا الإضافية لبرنامجك .

الشئ الأول الذى أقترح أن تفعله الآن أن تنشأ نسخة احتياطية لإسترجاع الكود للوقوف على هذه المرحلة .
فإذا كنت تعمل مع لغة تتطلب أن يتم تفسيرها إحتفظ نسخة من الملف التنفيذى كما فى مصدر الكود . وينبغى عليك أن تحفظ نسخة إحتياطية لإسترجاع مصدر الكود أول بأول على أى حال (على الأقل مرة كل يوم)
ولكن هذا هام جداً خصوصاً الآن لأنها تمثل أساس المشروع . فمن الهام أن تحفظ كل المواد العلمية إلى هذه المرحلة فى حالة حدوث أى تغيرات مستقبلية تعطل العمل — إذا حدث هذا ، سابقاً على الأقل ، فيمكنك أن تعود للوراء حيثما كنت .

بمجرد أن يكون لديك نسخة إسترجاع آمنة ، يمكنك بعدها أن تذهب إلى العمل على الكود مرة أخرى وتشرع فى عمل التغييرات بناءً على ردود الفعل .

تحتاج الآن أن تصحح أى شائبة أو خلل أو ثغرة توجد فى الكود . إفحص بحذر هل التصحيح الذى أجرىته يصح بالفعل المشكلة .

فكرة جيدة أن تحاول نسخ الخطأ أو الشائبة أو الخلل الموجود فى الكود قبل محاولة إصلاحه وهذا لأنه يسمح لك فى وقت لاحق أن ترى بوضوح ما إذا كنت بالفعل قد أصلحته !

هذه المرحلة من الكود لا ينبغي ان تستغرق وقتاً طويلاً مادامت الخطوة الأولى (إذا لم يكن عليك أن تضيف بعض المزايا الصعبة حقاً) . وبإضافة ميزة ,قم بإختبارها وانتقل لمابعدھا .وقم بحفظ قائمة بالتغيرات التى تجربها و لماذا إخترت أن تتركها حتى إصدار المستقبل من البرنامج .

بمجرد أن تصلح كل الشوائب التى وجدتها فى الإختبار وإضافة أى مزايا جديدة , تحتاج أن تعود للوراء إلى إختبار آخر بسبب أنك بالفعل مدمت تعمل على الكود فبلا شك قد أضفت شوائب جديدة إلى الكود . فتحتاج أن يتم إقتناصها والقضاء عليها , لذلك تحتاج مساعدة من الآخرين مرة أخرى .

فهذا وقت جيد للعودة للوراء إلى مرحلة الإختبار — وهذا هو وقت إختبار الـBeta ! مع ذلك , قبل أن تفعل هذا , حان الوقت أن تعدل الكود .

Tweak the Code

إقترب الكود الآن من أن يكتمل (مع حذف أى شائبة يحتويها الكود) لذلك فهذا وقت ممتاز أن تلف الكود لتبسيطه قليلاً

الشئ الأول لتفعله هو النظر إلى الدوال أو الكود الغير مستخدم خلال البرنامج . فيمكن ان تكون مهمة شاقة , ولكن ربما تقلل حجم التطبيق النهائى الذى أنشأته بشكل كبير جداً , وهذا سيقفل كمية موارد النظام التى سيتسهلكها التطبيق فى حالة الإستخدام . مر بالكود دالة بدالة وإنظر إذا ما كان يمكنك تتبع المسائل المنطقية به . وتذكر أن تلقى نظرة على أى شئ فى غير محله , وإنظر إذا كان يتم إستخدامه .

فبينما تمر بالكود , ربما ترى أسطر من الكود تحتاج أن تكثفها أو تضغطها (خاصة الكود الذى كتبته فى وقت مبكر — لا شك أن كتابتك للكود قد تحسنت خلال دراسة المشروع) . هذا سيبسط الكود , كما أنه يحسن الأداء والسرعة .

إذا كنت تستخدم لغة تترجم فضلاً عن انها تفسر (مثال VBScript أو JavaScript) وقت جيد الآن أن تقوم بحذف كل التعليقات التى لديك فى كود الـBeta (ولكن إحفظ نسخة احتياطية كمرجع بالطبع) . هذا أيضاً سيبسط الأمور ويحسن وقت التحميل والأداء .

أنت الآن مستعد أن تنتقل إلى الإختبار النهائى لذا أى شوائب يتم إيجادها تدمر .

Final Testing

الإختبار النهائى

مرحلة الإختبار النهائى هى فقط التركيز على إيجاد وإزالة الشوائب . إلى هذه المرحلة , لا تهتم بمزايا جديدة أو أفكار جديدة . فسيتم وضعهم فى قائمة الإنتظار ويمكن أن تشملهم فى الإصدارات التالية .

إحصل على الإختبار النهائى من المختبرين (يفضل نفس المختبرين الذين كانوا لديك فى المعاينة) وأحكم حكم نهائى على الكود . قم بعمل إختبار نهائى بنفسك أيضاً. بعدها, أن أمكن , ضع التطبيق فى جانب واحد لبعض الأيام قبل أن تعود إليه وتختبره مرة اخرى .

عندما تتأكد أن الشوائب قد غادرت , فأنت بعدها بمثابة التفكير فى الإصدار النهائى الكامل (إذا لم تقم من قبل بإنشاء ملف مساعدة قبل الإصدار — هذا إختيارى)

يبدأ بعض المبرمجين العمل على ملفات المساعدة فى وقت مبكر من عملهم على التطبيقات , ولكن أنا أميل أن أعتقد ان هذه فكرة سيئة . لأن البرنامج يمكن أن يتغير ولكن التوثيق سيظل ثابت , إثارة بلبله , من الأفضل ترك إنشاء ملفات المساعدة حتى يقترب التطبيق من الانتهاء .

و كل ما تبقى عليك أن تفعله هو أن تطلق العنان لتطبيقك للعالم المنتظر بالخارج .

Summary

الملخص

فى هذا الفصل , أخذنا جولة خلال عملية إنشاء التطبيق , بدأنا من نقطة الفكرة المبدئية , بنائها وأخذها مروراً إلى التطبيق النهائى . أنا لم أركز على الكود ولكن على العملية التى تحتاج أن تتبعها وأيضاً كيف توثق ما قمت بفعله لذا فتعرف أين أنت وماذا قمت بفعله وما عليك فعله الآن , وماذا تركت .

حاولت أن اكون اكثر تفصيلاً على قدر الإمكان , حيث جعلت العملية عامة ومناسبة لأى نوع من المشاريع لديك. ربما تحتاج أن تخصص الخطوات اعتماداً على ما تفعله لأنه ليست كل الخطوات ستكون ذات صلة بكل انواع المشاريع . ليس هذا فقط , ولكن ينبغى أن تبقى مع ما تشعر بالراحة معه . كل هذا جيد وعادى جداً , طالما أنك تخطط بتروى ولا تسرع , ستكون بخير .

12

Interacting with Files

التفاعل مع الملفات

هذا الفصل يستعرض طرق التفاعل مع نظام الملفات في حاسبك الشخصي . سنتعلم أن تستخدم مبادئ البرمجة البسيطة لتبدأ بإنشاء نموذج تطبيق سيحفظ بالفعل البيانات إلى القرص الصلب لنستردها فيما بعد .

عملية حفظ وإسترجاع البيانات هي عادةً ميزة متقدمة مظم المبتدئين لا يأتون جهتها , ولكن في الحقيقة هي جانب مهم في عمليات الحاسب الحديثة . ولهذا قد قررت في هذا الكتاب أنك تحصل على الفرصة للعمل مع كود يحفظ معلومات ويتم إسترجاعها فيما بعد .

The Principles of Saving Data

مبادئ حفظ البيانات

كما أكتب هذه الصفحة (بإستخدام Microsoft Word) , فإننا بين حين وآخر أحفظ البيانات إلى القرص الصلب . معظم الناس عندما يحفظون الملفات هم حقاً لا يفكرون فيما يقومون بفعله — هم فقط يحفظون ملف لأنهم يريدون أن يدخلوا لهذه البيانات فيما بعد و إعادة كتابة هذه البيانات كل مرة عمل شاق أو ربما يريدون أن يرسلوا هذه البيانات إلى شخص آخر . هذا ما سأقوم بفعله . سأرسله إلى برنامج المحرر الخاص بي . الذي بدوره سيمررها إلى أشخاص آخرين حيث سينتهى بها الحال في نهاية المطاف إلى الطابعة . إلى هذه النقطة سيتم تحويله من كونه صيغة رقمية Digital إلى شكل تناظري analog . وأخيراً ستلتقط هذا الكتاب وتقرأه . كل هذا ممكن لأنني كتبت الفصل , ثم حفظته .

حفظ الملفات عن كل ما يتم حفظه . عندما أحفظ ملف معالجة نصوص , فأنا أحفظ المزيد من النصوص فقط . بعض الأشياء التي أحفظها :

- تنسيق الكلمات , الفقرات , والصفحات .
- إدراج أى صورة في الصفحة .
- معلومات عن المؤلف .
- معلومات عن متى تم آخر حفظ .
- معلومات اللغة .
- قوالب مخصصة .

من المحتمل أن ترى كم مدى تعقيد التطبيق . والأكثر تعقيد في الملف هو أنه سيكون عليك الحفظ . مع ذلك , عليك أن تبدأ في مكان ما ومعالج النصوص ليس المكان الذي أوصى به ! دعنا نبدأ بالعمل مع ملفات معتمدة على النصوص والبناء من هناك .

سنلقى أيضاً نظرة على بعض التطبيقات البسيطة وإستخدامهم كأرضية لنبنى عليها وننظر كم بساطة الشئ , مثال حفظ ملف نصي , يمكن أن يكون مكان جيداً أن نبدأ بحفظ ملفات مع بعض التركيبات المعقدة.

The File Life Cycle

دورة حياة الملف

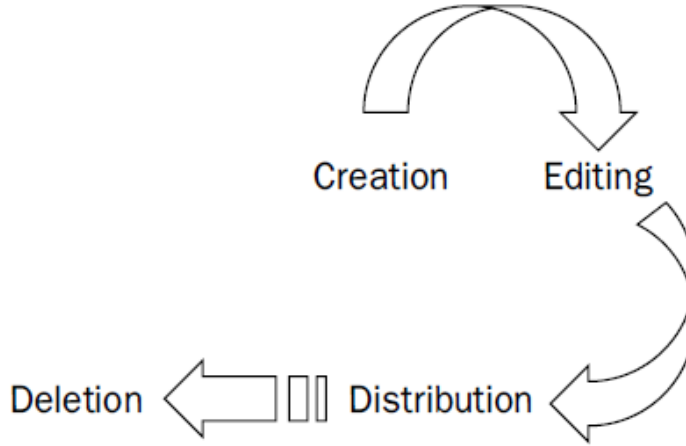
قبل أن ننظر إلى الكود لنعمل مع الملفات فكرة جيدة ان نبدأ بالنظر إلى دورة حياة الملف وكيف أن للملفات المختلفة دورة حياة مختلفة .

إذا فكرت فيها للحظة, أنا متأكد أنك ستوافق ان معظم الملفات لديها دورة حياة تعتمد على أربع مراحل

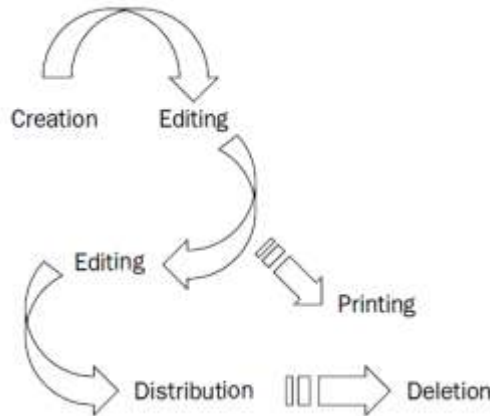
- إنشاء الملف .
- تحرير الملف .
- التوزيع .
- الأرشفة / الحذف .

هذا هو المسار الطبيعي لمعظم الملفات وبينما تم فحص البداية والنهاية , ما يحدث بين إنشاء الملف وجعله من المحفوظات أو حذفه يمكن أن يكون إما بسيط أو معقد .

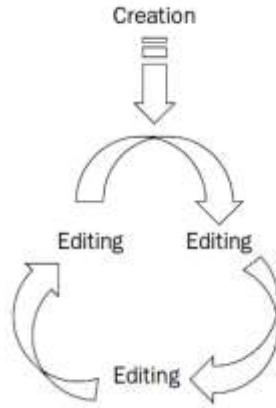
دعنا نلقى نظرة على بعض الأمثلة . الشكل التالي دورة حياة قياسية لملف ملاحظات بسيط الذي ربما تنشأه ليتحفظ فيه ببعض الأشياء . تم إنشاء الملف وبعدها ربما لم يتم عمل تحرير له بعد على كل حال , وتم المضي قدما إلى مرحلة الحذف .



الشكل التالي يظهر دورة الحياة أكثر تعقيداً — فهذا وقت العمل على ملف معالجة نص أو ربما يكون برنامج جدول أو صور . هذا الملف يمر به تحريرات متعددة وهناك إمكانية وجود توزيعات متعددة (ربما يكون مطبوع عدة مرات أو وزع بواسطة البريد الإلكتروني لأخرين ليقروا) . عادةً هذه الملفات , تكون فقط مثل ملفات الملاحظات , وقد عاشوا حياتهم الطويلة وإما أن يحذف (لم يعد مطلوب) أو ان يكون ضمن المحفوظات في حالة انك تحتاج ان تجعله مرة لك في المستقبل .



فى الشكل التالى قد صورت لك دورة حياة ملف مثال ملف قاعدة بيانات . هذا النوع من الملفات يمر بعدد غير معدود من التحديثات بمجرد إنشائه وله حياة طويلة جداً . الكثير من ملفات قواعد البيانات لا تحذف أو تأرشف فى حد ذاتها , ولكن ربما تحذف أو تأرشف جوانب محدده من البيانات بمرور الوقت والعمل عليها .



تركيبات الملفات المعقدة هى طريقة خارج نطاق هذ الكتاب لإن المهارة التى تحتاجها ليست فقط أن تكتب تطبيقات تستخدمها , ولكن أيضاً لتخطيط العمل الداخلى الفعلى لتنسيق الملف . فى هذا الفصل , سنركز على تنسيقات الملف البسيطة المعتمدة على النصوص . ومع ذلك , كما سترى قريباً , حتى الملفات ذات التنسيقات البسيطة تقدم لك إمكانيات ضخمة بدون بذل الكثير من العمل — وهو الشئ الذى يراقبه المبرمج دوماً .

Working with Files

العمل مع الملفات

أنا هنا أبسط لك الأمور قليلاً , ولكن بفعل هذا يمكننا أن نبدأ بكتابة كود يقوم بتحويل الثلاث مراحل المميزة الإنشاء , التحرير , و الحذف من النظرى إلى العملى أو الواقع .

The Tools

الأدوات

كل لغة برمجة تعالج الملفات بطريقة مختلفة , البعض منها بسيط والبعض الآخر معقد . أسهل وأوضح لغات البرمجة لمعالجة الملفات هى Visual Basic , ولكن لأنها منتج تجارى سيكون علينا أن نستخدم ثانى أفضل لغة مجانية متاحة — VBScript . VBScript هى لغة برمجة كتابية , ولكن بتشغيلها من خلال برنامج خدمى يسمى Windows Script Host (ميزة متاحة فى كل النسخ الحديثة Microsoft Windows) , يمكننا استغلال هذه اللغة وسهولة الحصول عليها للعمل كلغة البرمجة .

هناك مميزات عديدة فى إستخدام هذه اللغة :

- مجانية : دوماً سبب عظيم .
 - لا شئ للتنشيط : كل ماتحتاجه هو أن تكتب كود , و تحفظه , وتقوم بتشغيله وسترى النتائج فوراً . ليس هناك تطبيقات تحتاج أن تقوم بتنشيطها.
 - ليس برنامج معقد للتعلم : نصف المعركة مع لغة برمجة جديدة هو أن تشغل طريقك حول بيئة العمل الفعلية — لتعمل مع VBScript تحتاج أن تعرف كيف تستخدم محرر النصوص ! هذا سريع وسهل .
 - لا تفسير: لأنك لا تحتاج إلى تفسير الكود قبل تشغيله , يمكن أن ترى النتائج للكود أسرع وأسهل.
 - أسهل لفحص الأخطاء : لأنه لا يجب عليك أن تستمر بالتقليب بين مصدر الكود والكود المفسر كل الوقت , يمكنك أن تكتشف وتصحح الأخطاء فى المزيد من السرعة .
- لذا , كمستخدم windows , كل ماتحتاجه هو محرر نصوص وتكون جاهز لتبدأ البرمجة .

Getting Started

البداية

منذ أن بدأت دورة حياة الملف طبيعياً مع إنشاء الملف , سنبدأ هناك . لا يجب على البرنامج دوماً أن ينشأ ملفات يعمل معها مع ذلك — البرنامج الخدمي المرفق الذي كتبتة ليعمل windows Registry (إنظر الفصل 13 , "The Windows Registry") للعمل مع الملفات الموجودة بالفعل . ربما تريد أن تكتب برنامج مرفق يبحث في القرص الصلب عن الملفات المؤقتة الغير مفيدة فيحذفها . في حالة أن برنامجك لا يحتاج ان بهتم بإنشاء ملف أو تعديله ويمكنه فقط ان يحذف ملفات . ولكن سنبدأ من البداية ونلقى نظرة عملية أنشاء ملف .

Creating a File with VBScript

إنشاء ملف بـVBScript

عندما تقوم بتشغيل كود VBScript من خلال نظام Windows Script Host (هذا هو , قم بتشغيله على النظام الخاص بك مباشرة بدلاً من تمرير إلى الإنترنت في المستعرض) هو مدهش وسهل لإنشاء ملفات .

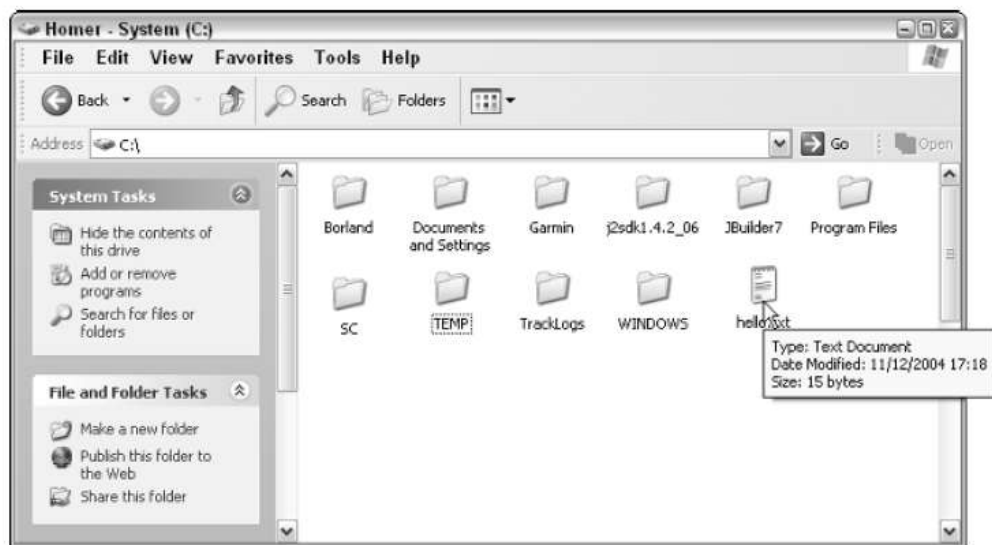
Basics

المبادئ

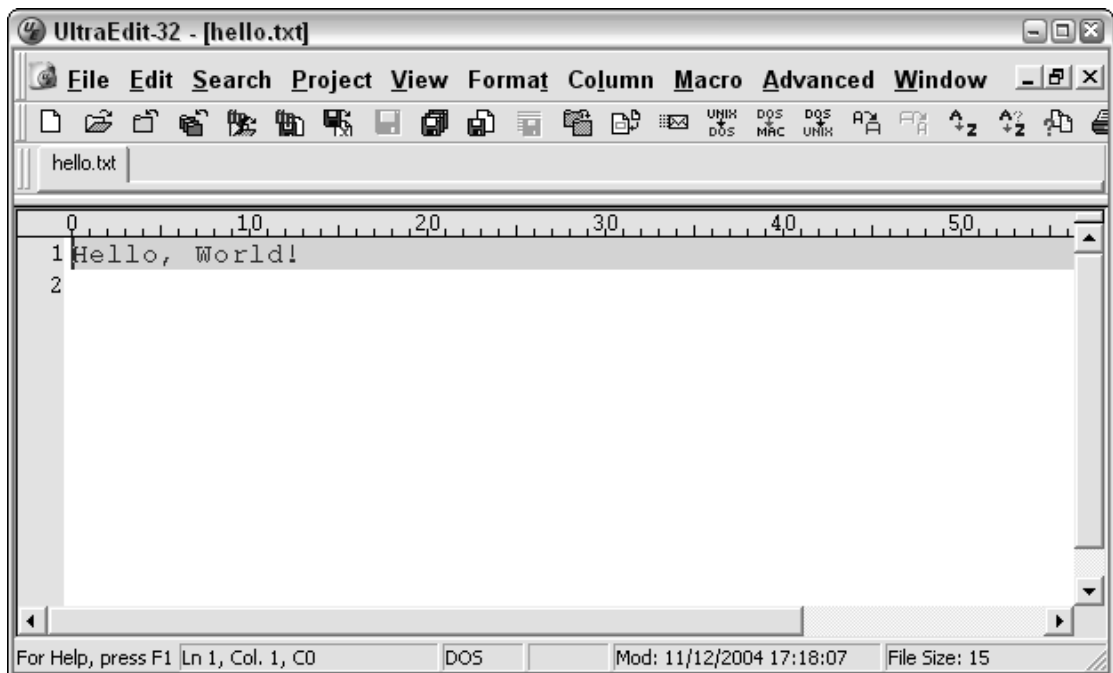
من المحتمل أن تفاجئ مجرد أن ترى الكود البسيط الذي نحتاجه , في الحقيقة نحن نحتاج إلى خمسة أسطر من الكود

```
Dim fso, TestFile
Set fso = CreateObject("Scripting.FileSystemObject")
Set TestFile = fso.CreateTextFile("c:\hello.txt", True)
TestFile.WriteLine("Hello, World!")
TestFile.Close
```

خمس أسطر , خذ هذا الكود , ضعه في محرر نصوص مثل Windows Notepad وأحفظه بإمتداد ملف .vbs (إسم الملف لا يهم) بعدها اضغط مرتين على الملف لتشغيله وسينشأ ملف بإسم Hello.txt في المسار C: (إنظر الشكل)



خذ نظرة مقربة للملف (وستلاحظ أنه يحتوي على السطر " Hello, World! " إنظر الشكل)



دعنا نأخذ جولة داخ الكود وننظر كيف يعمل

```
Dim fso, TestFile
Set fso = CreateObject("Scripting.FileSystemObject")
Set TestFile = fso.CreateTextFile("c:\hello.txt", True)
TestFile.WriteLine("Hello, World!")
TestFile.Close
```

السطر الأول من الكود يعلن بوضوح المتغيرات التي سنستخدمها فيما بعد . فنياً , لا نحتاجه ويمكن الإستغناء عنه ولكن إعلانهم بوضوح يقلل خطورة المتغير إملائياً فيما بعد .

```
Dim fso, TestFile
Set fso = CreateObject("Scripting.FileSystemObject")
Set TestFile = fso.CreateTextFile("c:\hello.txt", True)
TestFile.WriteLine("Hello, World!")
TestFile.Close
```

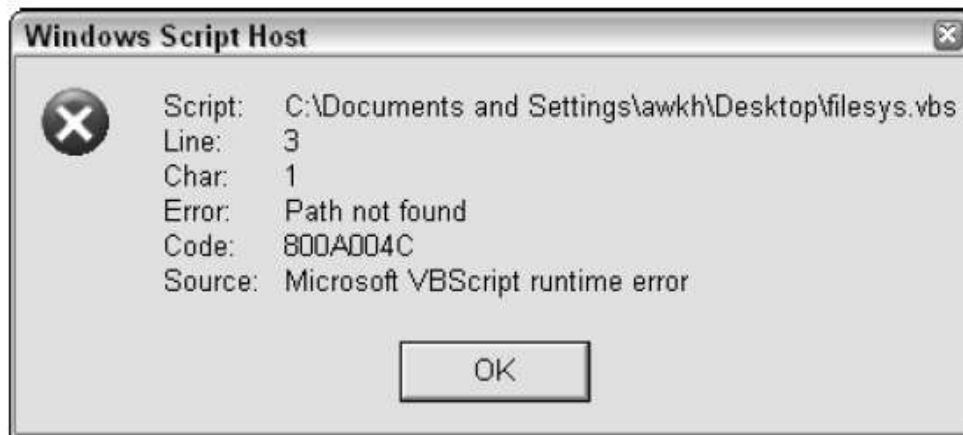
هذا السطر هو قوة الكود ويقدم الكائن الذي يمكنك من الدخول إلى نظام الملفات , هذا السطر أيضاً ليس ضرورياً ولكنه يسمح لنا ان نكتب كود أقل فيما بعد .

```
Dim fso, TestFile
Set fso = CreateObject("Scripting.FileSystemObject")
Set TestFile = fso.CreateTextFile("c:\hello.txt", True)
TestFile.WriteLine("Hello, World!")
TestFile.Close
```

في هذه الحالة , فيما بعد في السطر التالي . هذا السطر يستخدم الكائن من السطر الثاني لإنشاء ملف إختبار . ملف الإختبار محدد بالمتغير TestFile . windows Registry الذي اعلنا عن الموقع و الإسم للملف . يمكنك , إن أردت أن , تختار إسم وموقع مختلف . دعنا نقول انك تريد ان تنشأ ملف في مكان آخر أو في مجلد مختلف أو تعطيه إسم مختلف , إذا فهذه مجرد حالة تغيير المسار المستخدم :

```
...
Set TestFile = fso.CreateTextFile("d:\hello\hello.txt", True)
...
```

أحد الأشياء التي يجب أن يكون في مكانه هو المجلد — هذه الجملة لا يمكن أن تنشأ مجلد للملف . لذا إذا أعلنت عن مسار غير موجود , ستحصل على رسالة خطأ عندما تحاول أن تشغيل الكود , كما هو مبين في الشكل



إذا كان الملف بالفعل موجود , هذه الجملة ستسبب في تغطية الموجود من قبل . إذا أردت أن تمنع ذلك يمكنك تغيير True في النهاية إلى False .

```
***  
Set TestFile = fso.CreateTextFile("c:\hello.txt", False)  
***
```

إذا كان الملف بالفعل موجود , سينتج عن الكود رسالة خطأ .

```
Dim fso, TestFile  
Set fso = CreateObject("Scripting.FileSystemObject")  
Set TestFile = fso.CreateTextFile("c:\hello.txt", True)  
TestFile.WriteLine("Hello, World!")  
TestFile.Close
```

السطر التالي هو السطر الذي يعالج الملف الذي أنشأ مع النص الذي تريده ليحتويه . هذا هو كتابة السطر في نفس الوقت بإستخدام دالة WriteLine . ويمكن تكراره حسب الحاجة .

```
Dim fso, TestFile  
Set fso = CreateObject("Scripting.FileSystemObject")  
Set TestFile = fso.CreateTextFile("c:\hello.txt", True)  
TestFile.WriteLine("Hello, World!")  
TestFile.WriteLine("Another line")  
TestFile.WriteLine("This is getting boring now.")  
TestFile.WriteLine("That's it, I'm outta here. . .")  
TestFile.Close
```

مؤخراً , ينتهي الكود بإغلاق الملف . هذه هي طريقة حفظ الكود للتحديثات إلى الملف .

Creating a Folder

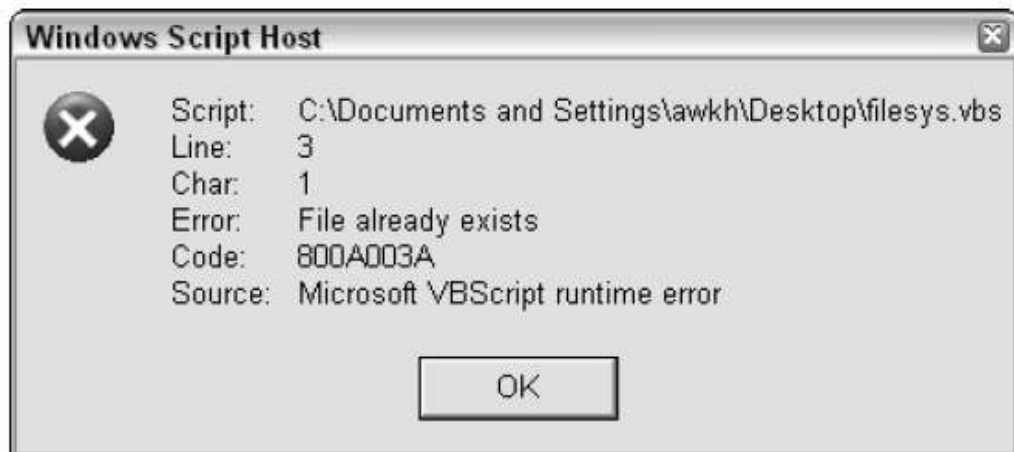
إنشاء مجلد

لقد قلنا سابقاً أن الكود إلى هذا الحد لا يمكنه أن ينشأ مجلد ليضع الملفات بداخله. ولكن لحسن الحظ كتابة كود ليفعل هذا ليس صعباً أن يتم إنجازه .

لإنشاء مجلد , إستخدم دالة CreateFolder . التالي هو مثال تدعك تنشأ مجلد

```
Dim fso, fld
Set fso = CreateObject("Scripting.FileSystemObject")
Set fld = fso.CreateFolder("c:\New Test Folder")
```

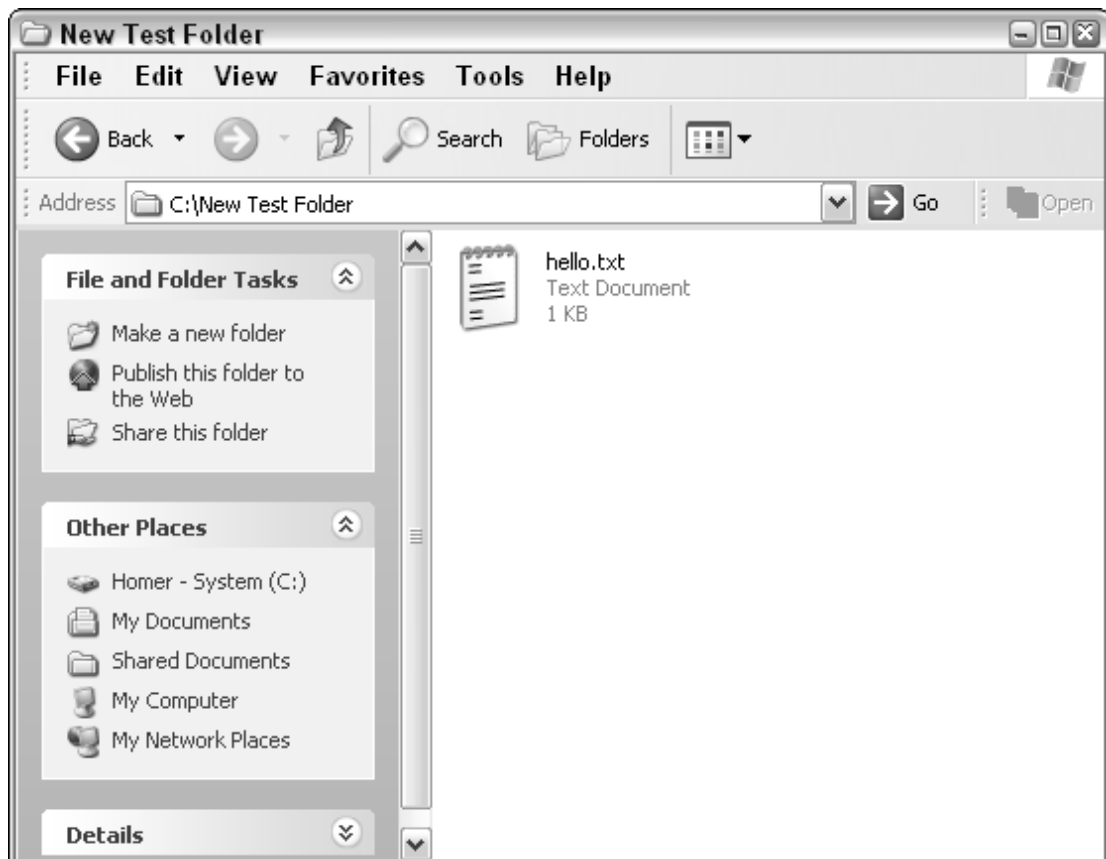
الجزء الرئيسي للكود هو السطر الثالث , وهو الأمر الذي سينشأ المجلد . إذا كان المجلد موجود من قبل , سيعرض لك خطأ , كما هو مبين بالشكل



إدماج السطرين من الكود , يمكنك أن تكتب كود ينشأ مجلد ثم تضع الملف في المجلد الجديد الذي تم إنشاؤه .

```
Dim fso, fld, TestFile
Set fso = CreateObject("Scripting.FileSystemObject")
Set fld = fso.CreateFolder("c:\New Test Folder")
Set fso = CreateObject("Scripting.FileSystemObject")
Set TestFile = fso.CreateTextFile("c:\New Test Folder\hello.txt", True)
TestFile.WriteLine("Hello, World!")
TestFile.Close
```

إحفظ الكود في ملف بإمتداد ملف vbs . وقم بتشغيله وإفحص لترى ما إذا كان المجلد والملف قد تم إنشاؤهم . هنا لقد فحصت النظام الخاص بي ووجدت كل شيء موجود (انظر الشكل)



Creating Multiple Files

إنشاء ملفات متعددة

يمكن أن يتم تعديل الكود لينشئ ملفات متعددة في نفس المجلد في نفس الكود .

```
Dim fso, fld, TestFile, TestFile2, TestFile3
Set fso = CreateObject("Scripting.FileSystemObject")
Set fld = fso.CreateFolder("c:\Folder2")
Set fso = CreateObject("Scripting.FileSystemObject")
Set TestFile = fso.CreateTextFile("c:\Folder2\hello.txt", True)
TestFile.WriteLine("Hello, World!")
TestFile.Close
Set TestFile2 = fso.CreateTextFile("c:\Folder2\another file.txt", True)
TestFile2.WriteLine("I don't have anything to say any more!")
TestFile2.Close
Set TestFile3 = fso.CreateTextFile("c:\Folder2\and another.txt", True)
TestFile3.WriteLine("D'oh!")
TestFile3.Close
```

بدأ الكود أن يبديوا معقد , وأيضاً أكثر عرضة للأخطاء مثل محاولة إنشاء مجلد حيث يكون موجوداً من قبل . فكرة جيدة أن تعدل الكود ليفحص أولاً ما إذا كان المجلد موجود من قبل وتحاول فقط أن تنشأه إذا لم يكن موجود .

Making Use of Conditionals

هنا سنحتاج إلى الأوامر الشرطية لفحص المجلد ونصنع إختيار إعتماًداً ما إذا كان الأمر الشرطى True أو False

```
Dim fso
Set fso = CreateObject("Scripting.FileSystemObject")

If (fso.FolderExists("c:\Folder")) Then
    MsgBox("Folder exists.")
Else
    MsgBox("Folder doesn't exist.")
End If
```

هذا الكود يفحص وجود المجلد c:\folder . إذا كان موجود سيعرض رسالة واحدة (إنظر الشكل)



وعرض رسالة اخرى مختلفة إذا كان المجلد غير موجود (إنظر الشكل)



عرض الرسائل هو احد الأشياء الهامة , ولكن يمكنك الآن أن تجعل الكود ينشأ المجلد إذا لم يكن بالفعل موجود , فضلاً عن توليد رسائل خطأ أو مربع حوارى .

```
Dim fso, fld
Set fso = CreateObject("Scripting.FileSystemObject")

If (fso.FolderExists("c:\Folder")) Then
    MsgBox("Folder exists.")
Else
    Set fld = fso.CreateFolder("c:\Folder")
End If
```

وهذا مجال لخطأ مثل الذى يتسلل إلى الكود

```
Dim fso, fld
Set fso = CreateObject("Scripting.FileSystemObject")

If (fso.FolderExists("c:\Folder")) Then
    MsgBox("Folder exists.")
Else
    Set fld = fso.CreateFolder("c:\Folder2")
End If
```

Making Use of Variables

إستخدام المتغيرات

حالياً يفحص إمكانية وجود ملف واحد , و ينشأ واحد آخر بإسم مختلف — وصعوبة إمكانية كشف الخطأ .

إستخدام متغير ليمسك بإسم الملف ليقفل مجال الخطأ

```
Dim fso, fld, fldName, drvName
drvName = "c:\\"
fldName = "folder"
Set fso = CreateObject("Scripting.FileSystemObject")

If (fso.FolderExists(drvName & fldName)) Then
    MsgBox("Folder exists.")
Else
    Set fld = fso.CreateFolder(drvName & fldName)
End If
```

هذا الكود الآن ينفذ بطريقة محترفة وذكية , وأذكى بدرجة كافية لفحص وجود المجلد قبل محاولة إنشاؤه وكتابة كود بطريقة نقل بها المجال لوقع خطأ كحد ادنى .

يمكن ان نأخذ هذا الكود ونقوم بتطبيقه على المثال السابق الذى ينشأ الملفات .

```
Dim fso, fld, fldName, drvName, TestFile
drvName = "c:\\"
fldName = "folder"
Set fso = CreateObject("Scripting.FileSystemObject")

If (fso.FolderExists(drvName & fldName)) Then
    MsgBox("Folder exists.")
Else
    Set fld = fso.CreateFolder(drvName & fldName)
End If

Set TestFile = fso.CreateTextFile(drvName & fldName & "\hello.txt", True)
TestFile.WriteLine("File contents here")
TestFile.Close
```

Adding Flexibility — Prompt for File and Folder Names

إضافة بعض المرونة — التأجيل لإدخال إسم الملف والمجلد

يعمل هذا جيداً , ولكن المشكلة الوحيدة أسماء المجلدات وإسماء الملفات , ومحتوى الملف الذى تم إصلاحه , وهذا موقف حرج لأنه يجعل الكود فى العام غير مرن إلى حد كبير ويحد من الإداء الوظيفى .

أفضل خطة لذلك هو أستخدام المدخلات لإسم المجلد والملف . ها هو الكود بعد التعديل ليسأل عن إدخال إسم الملف ليحفظه .

```

Dim fso, fld, fldName, drvName, TestFile, fileName
drvName = "c:\"
fldName = "folder"
fileName = inputbox("Enter the name of the file:", "Filename")
Set fso = CreateObject("Scripting.FileSystemObject")

If (fso.FolderExists(drvName & fldName)) Then
    msgbox("Folder exists.")
Else
    Set fld = fso.CreateFolder(drvName & fldName)
End If

Set TestFile = fso.CreateTextFile(drvName & fldName & "\" & fileName & ".txt",
True)
TestFile.WriteLine("File contents here")
TestFile.Close

```

يمكنك الآن ان تأخذ الفكرة وتطبقها على إسم القرص وإسماء المجلدات .

```

Dim fso, fld, fldName, drvName, TestFile, fileName
drvName = inputbox("Enter the drive to save to:", "Drive letter")
fldName = inputbox("Enter the folder name:", "Folder name")
fileName = inputbox("Enter the name of the file:", "Filename")
Set fso = CreateObject("Scripting.FileSystemObject")

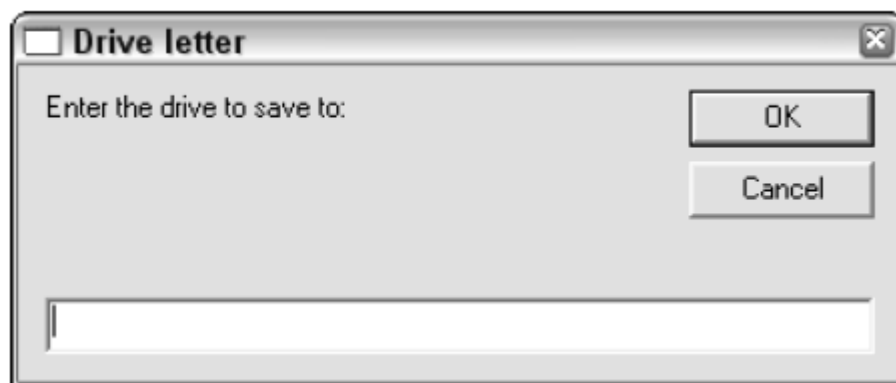
If (fso.FolderExists(drvName & fldName)) Then
    msgbox("Folder exists.")
Else
    Set fld = fso.CreateFolder(drvName & fldName)
End If

Set TestFile = fso.CreateTextFile(drvName & fldName & "\" & fileName & ".txt",
True)
TestFile.WriteLine("File contents here")
TestFile.Close

```

دعنا نحفظ الملف (مرة أخرى , قم بتسميته اى شئ مادام انه بإمتداد .vbs) وقم بتشغيله .

اول شئ ستراه هو تأجيل لسؤالك عن إسم القرص حيث سيتم حفظ الملف (إنظر الشكل)



وهذا ينتج عن هذا السطر من الكود

```

Dim fso, fld, fldName, drvName, TestFile, fileName
drvName = inputbox("Enter the drive to save to:", "Drive letter")
fldName = inputbox("Enter the folder name:", "Folder name")
fileName = inputbox("Enter the name of the file:", "Filename")
Set fso = CreateObject("Scripting.FileSystemObject")

...

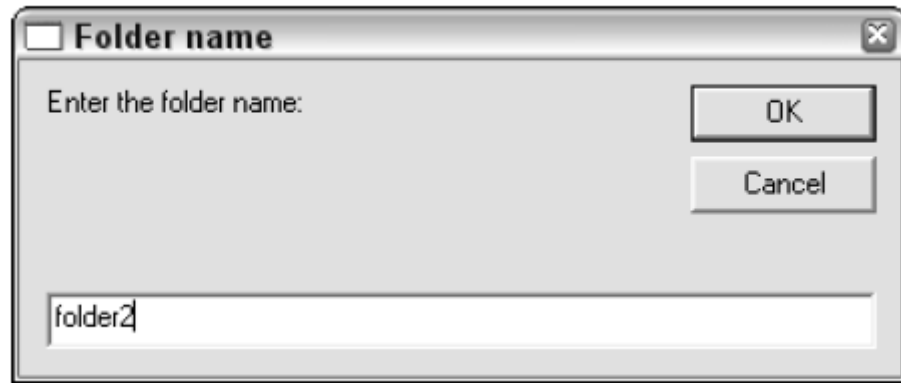
```

إدخال اسم القرص حيث ستحفظ الملف , على سبيل المثال

c:\

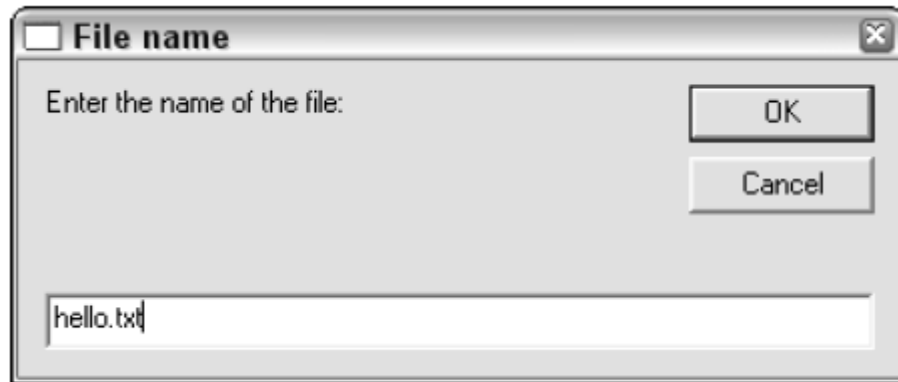
d:\

ثم تحصل على تأجيل لسؤالك عن اسم المجلد (إنظر الشكل)



```
Dim fso, fld, fldName, drvName, TestFile, fileName
drvName = inputbox("Enter the drive to save to:", "Drive letter")
fldName = inputbox("Enter the folder name:", "Folder name")
fileName = inputbox("Enter the name of the file:", "Filename")
Set fso = CreateObject("Scripting.FileSystemObject")
...
```

ثم , تأجيل آخر لسؤالك عن اسم الملف (إنظر الشكل)



```
Dim fso, fld, fldName, drvName, TestFile, fileName
drvName = inputbox("Enter the drive to save to:", "Drive letter")
fldName = inputbox("Enter the folder name:", "Folder name")
fileName = inputbox("Enter the name of the file:", "Filename")
Set fso = CreateObject("Scripting.FileSystemObject")
...
```

الآن مع كل المتغيرات المدرجه , سيعمل الأمر الشرطى لفحص ما إذا كان المجلد المحدد موجود .

```
...
If (fso.FolderExists(drvName & fldName)) Then
    msgbox("Folder exists.")
Else
    Set fld = fso.CreateFolder(drvName & fldName)
End If
...
```


إذا كان المجلد غير موجود , فينشأ :

```
...  
If (fso.FolderExists(drvName & fldName)) Then  
    MsgBox("Folder exists.")  
Else  
    Set fld = fso.CreateFolder(drvName & fldName)  
End If  
...
```

فلا حين أنه موجود , رسالة تأجيل تعرض بدلاً من ذلك (انظر الشكل)

```
...  
If (fso.FolderExists(drvName & fldName)) Then  
    MsgBox("Folder exists.")  
Else  
    Set fld = fso.CreateFolder(drvName & fldName)  
End If  
...
```



Check for Duplicate Files

فحص مضاعفة الملفات

يمكن أن نقوم بتعديل الكود قليلاً لنبحث ما إذا كان الملف المحدد بإسمه موجود من قبل .

```
Dim fso, fld, fldName, drvName, TestFile, fileName  
drvName = InputBox("Enter the drive to save to:", "Drive letter")  
fldName = InputBox("Enter the folder name:", "Folder name")  
fileName = InputBox("Enter the name of the file:", "Filename")  
Set fso = CreateObject("Scripting.FileSystemObject")  
  
If (fso.FolderExists(drvName & fldName)) Then  
    MsgBox("Folder exists.")  
Else  
    Set fld = fso.CreateFolder(drvName & fldName)  
End If  
  
If (fso.FileExists(drvName & fldName & "\" & fileName & ".txt")) Then  
    MsgBox("File already exists.")  
Else  
    Set TestFile = fso.CreateTextFile(drvName & fldName & "\" & fileName & ".txt",  
    True)  
    TestFile.WriteLine("File contents here")  
    TestFile.Close  
  
End If
```

Editing an Existing File

تحرير ملف موجود من قبل

لدينا الآن الكود الفعال الذى ينشأ الملفات فى الموقع حيث سيحفظ الملف . فلدينا هذا الملف ويمكننا أن نكتب كود قادراً على فتح ملف موجود بالفعل.

```
Dim fso, opnFile
Set fso = CreateObject("Scripting.FileSystemObject")
Set opnFile = fso.OpenTextFile("c:\test.txt", 2, True)
opnFile.Write "Hello world!"
opnFile.Close
```

مرة اخرى , خمسة أسطر من الكود هو كل ما نحتاجه .

```
Dim fso, opnFile
Set fso = CreateObject("Scripting.FileSystemObject")
Set opnFile = fso.OpenTextFile("c:\test.txt", 2, True)
opnFile.Write "Hello world!"

opnFile.Close
```

السطر الأول من الكود يعلن كل المتغيرات التى نحتاجها . ليسوا مطلوبين , ولكن سيجعل الكود أسهل لنتبعه وأقل عرضة للخطأ .

```
Dim fso, opnFile
Set fso = CreateObject("Scripting.FileSystemObject")
Set opnFile = fso.OpenTextFile("c:\test.txt", 2, True)
opnFile.Write "Hello world!"

opnFile.Close
```

السطر الثانى هناك يبسط إستخدام المتغير لتقصير السطر الثالث للراحة ولتقليل الأخطاء الممكنة فى الكود فيما بعد .

```
Dim fso, opnFile
Set fso = CreateObject("Scripting.FileSystemObject")
Set opnFile = fso.OpenTextFile("c:\test.txt", 2, True)
opnFile.Write "Hello world!"

opnFile.Close
```

السطر الثالث يعرف الدالة التى نحتاجها لفتح الملف . و يعرف أيضاً الملف ليتم فتحه . هذا اكثر قليلا على هذا الأمر . قريب من النهاية لاحظ الرقم 2 . فهو يعد احد ثلاثة إختيارات للأمر :

- فتح الملف للقراءة فقط . لا يمكنك ان تكتب إلى هذا الملف .
- فتح الملف للكتابة .
- فتح الملف والكتابة إلى نهاية الملف (فى معنى آخر إلحاق بالملف) [يقصد تكملة او وضع زيادات]

True تعرف كيف ينبغى فتح الملف , True تحدد Unicode , False , ASCII , و UseDefault , إقتراضى النظام .

فى هذا المثال , إختارنا أن نكتب إلى الملف

```
Dim fso, opnFile
Set fso = CreateObject("Scripting.FileSystemObject")
Set opnFile = fso.OpenTextFile("c:\test.txt", 2, True)
opnFile.Write "Hello world!"

opnFile.Close
```

السطر التالى هو السطر الذى سيكتب إلى الملف .

```

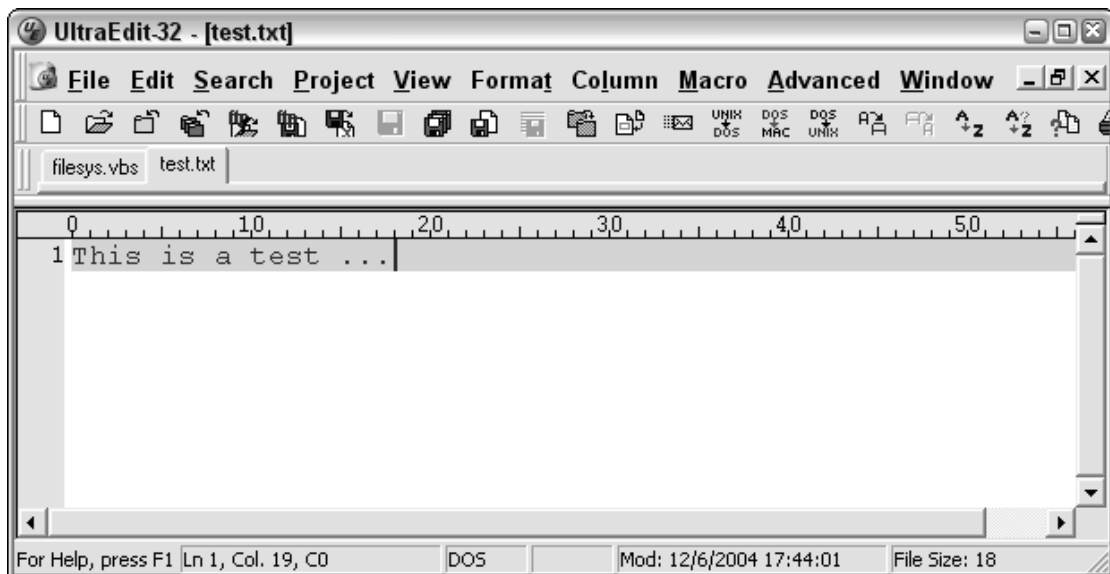
Dim fso, opnFile
Set fso = CreateObject("Scripting.FileSystemObject")
Set opnFile = fso.OpenTextFile("c:\test.txt", 2, True)
opnFile.Write "Hello world!"
opnFile.Close

```

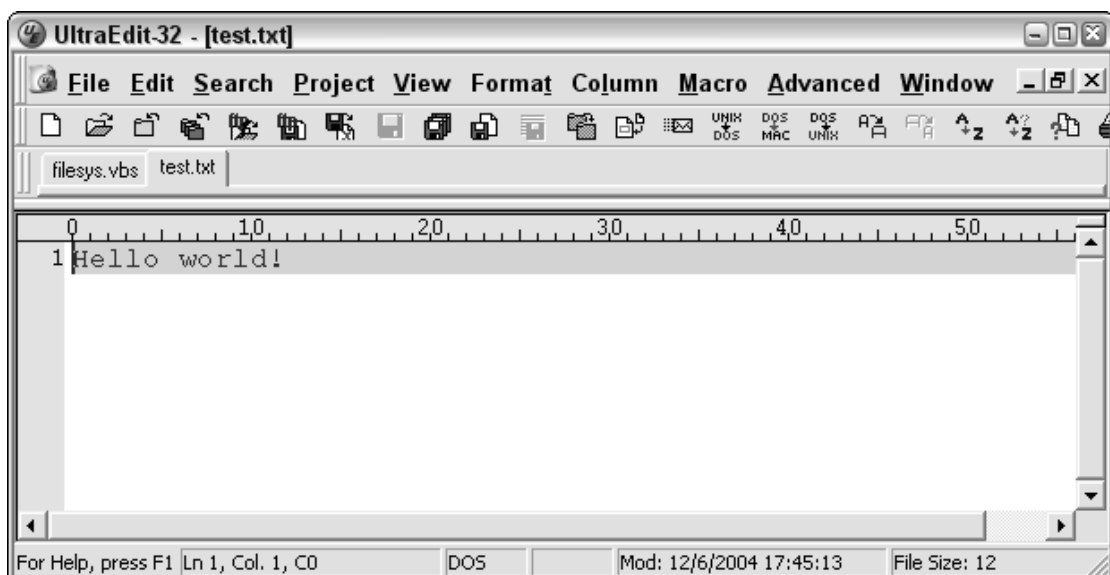
مؤخراً , يغلق الملف .

In Action فى الفعل

أولاً , إنشاء ملف فى المسار C:\ يسمى test.txt . داخل هذا الملف ضع أى نص تريد . لقد أضفت فقط بعض السطور كما ترى فى الشكل



الآن , يحفظ الكود فى ملف بإمتداد .vbs , قم بتشغيله . وينبغي أن تعمل بخفاء , ولكن إذا ألقيت نظرة على الملف test.txt , ينبغي ان تلاحظ أن محتوى الملف قد تم تغييره بواسطة الكود (كما هو مبين بالشكل)

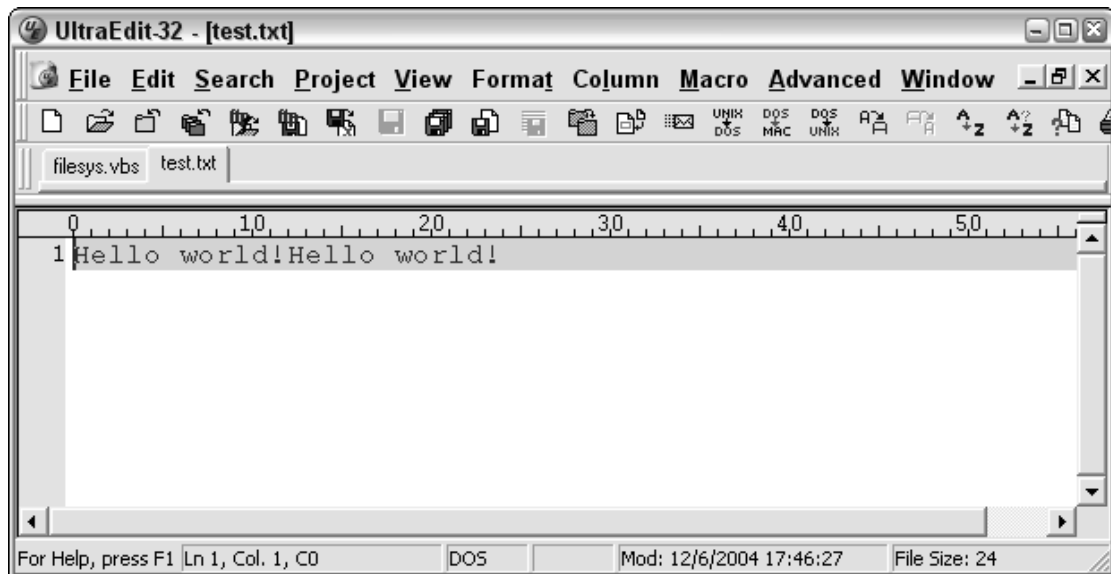


Appending a File

تغيير واحد صغير على الكود يعنى انه بدلاً من تحديث الملف , سنلحق بالملف ونضيف إليه نص في آخره .

```
Dim fso, opnFile
Set fso = CreateObject("Scripting.FileSystemObject")
Set opnFile = fso.OpenTextFile("c:\test.txt", 8, True)
opnFile.Write "Hello world!"
opnFile.Close
```

بتشغيل هذا الكود , النسخة الموجودة من الملف test.txt قد تم فتحها وقد تم إضافة جملة Hello World إلى نهاية الملف , كما هو مبين في الشكل :



Open File for Reading

فتح ملف للقراءة

إذا أردت فقط ان تفتح ملف للقراءة, سيكون عليك إستخدام الكود التالي :

```
Dim fso, opnFile
Set fso = CreateObject("Scripting.FileSystemObject")
Set opnFile = fso.OpenTextFile("c:\test.txt", 1, True)
```

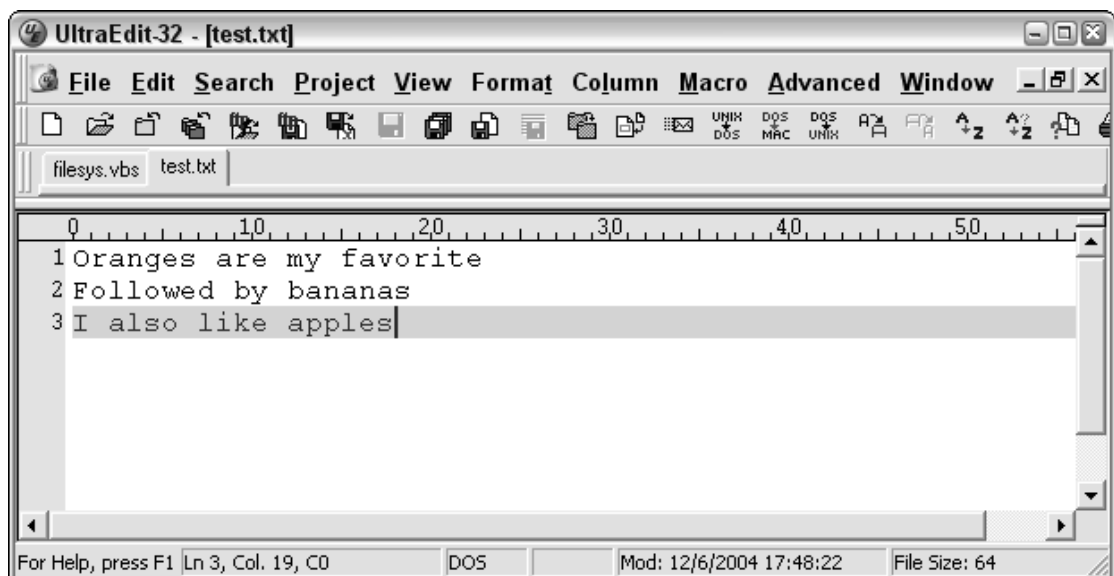
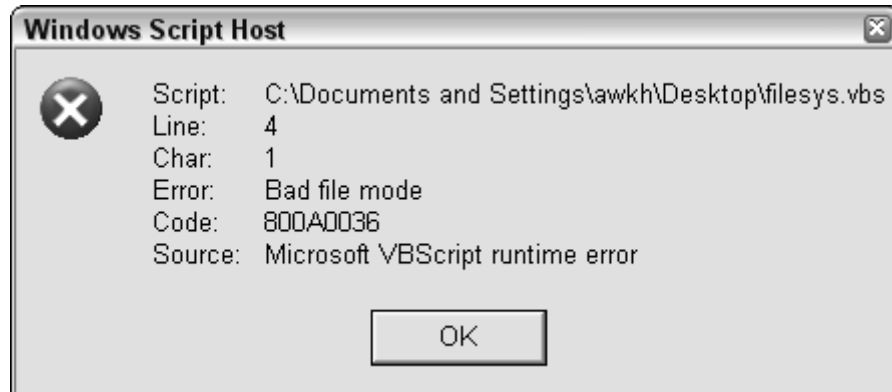
مع ذلك , ضع في بالك محاولة الكتابة إلى الملف الذى تم فتحه للقراءة فقط . بكتابة كود مثال التالى , سينتج عنه خطأ قد نتج (كما مبين في الشكل)

```
Dim fso, opnFile
Set fso = CreateObject("Scripting.FileSystemObject")
Set opnFile = fso.OpenTextFile("c:\test.txt", 1, True)
opnFile.Write "Hello world!"
opnFile.Close
```

ReadAll, ReadLine, and Read Methods

دوال قراءة الكل , قراءة سطر , أو قراءة

يمكن ان تختار ان تقرأ ملف أيضاً . لهذا , دعنا ننشئ ملف لديه القليل من المحتويات , شئ ما مثل الذى ما قد فعلته فى الشكل .



إستخدام هذا , يمكننا فحص الاختلاف بين دالة ReadAll و دالة ReadLine .

ReadAll

قراءة الكل

إليك مثال للكود الذى يستخدم دالة ReadAll

```
Dim fso, opnFile
Set fso = CreateObject("Scripting.FileSystemObject")
Set opnFile = fso.OpenTextFile("c:\test.txt", 1, True)
MsgBox(opnFile.ReadAll)
```

تشغيل هذا يسبب عرض محتويات الملف فى الرسالة المنبثقة , كما هو مبين بالشكل



ReadLine قراءة سطر

فهي مشابهة لدالة ReadAll , غير أن سطر واحد فقط هو الذى يتم قراءته بدلاً من كل الملف :

```
Dim fso, opnFile
Set fso = CreateObject("Scripting.FileSystemObject")
Set opnFile = fso.OpenTextFile("c:\test.txt", 1, True)
MsgBox(opnFile.ReadLine())
```

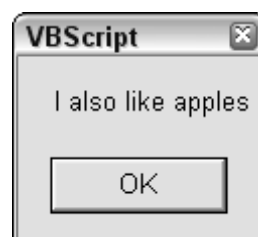
تشغيل هذا يتسبب فى ان السطر الأول من الملف يعرض فى رسالة منبثقة , كما هو مبين فى الشكل



لتقرأ تقدم الأسطر , أضف المزيد من جمل ReadLine إلى نهاية الكود . التالى سيقراً ثلاثة أسطر من الملف النصى :

```
Dim fso, opnFile
Set fso = CreateObject("Scripting.FileSystemObject")
Set opnFile = fso.OpenTextFile("c:\test.txt", 1, True)
MsgBox(opnFile.ReadLine())
MsgBox(opnFile.ReadLine())
MsgBox(opnFile.ReadLine())
```

هذا الكود ينتج عنه رسائل , كل منها يعرض سطر فى المرة , الشكل التالى يعرض الثالث



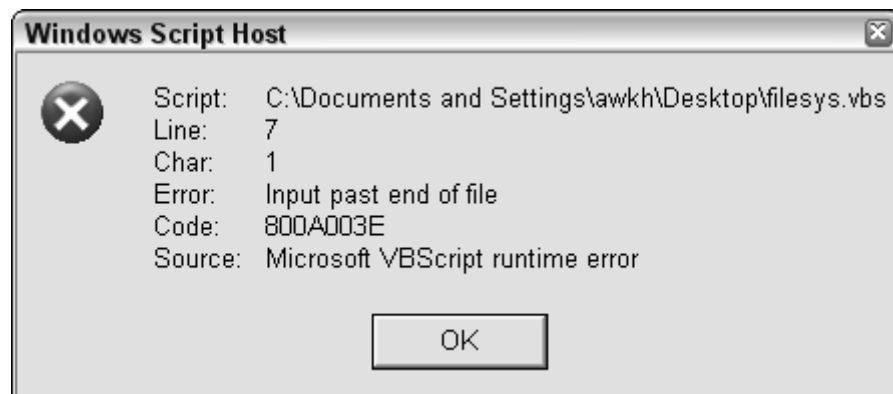
إذا أضفت المزيد من جمل ReadLine أكثر مما لديك , سينتج عنه خطأ :

```

Dim fso, opnFile
Set fso = CreateObject("Scripting.FileSystemObject")
Set opnFile = fso.OpenTextFile("c:\test.txt", 1, True)
MsgBox(opnFile.ReadLine())
MsgBox(opnFile.ReadLine())
MsgBox(opnFile.ReadLine())
MsgBox(opnFile.ReadLine())
MsgBox(opnFile.ReadLine())
MsgBox(opnFile.ReadLine())
MsgBox(opnFile.ReadLine())
MsgBox(opnFile.ReadLine())
MsgBox(opnFile.ReadLine())
MsgBox(opnFile.ReadLine())

```

رسالة الخطأ (تعرض في الشكل) تعرض رقم السطر الذي تسبب في الخطأ وعطل تشغيل الكود .



Read قراءة

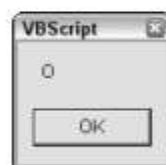
دالة Read تمكنك من تحديد عدد الأحرف التي ينبغي أن تقرأ من الملف .

الكود التالي سيقراً حرف واحد من ملف (مخرجات الكود معروضه في الشكل)

```

Dim fso, opnFile
Set fso = CreateObject("Scripting.FileSystemObject")
Set opnFile = fso.OpenTextFile("c:\test.txt", 1, True)
MsgBox(opnFile.Read(1))

```



بينما سيقراً هذا 20 حرف (مخرجات الكود تعرض في الشكل)

```
Dim fso, opnFile
Set fso = CreateObject("Scripting.FileSystemObject")
Set opnFile = fso.OpenTextFile("c:\test.txt", 1, True)
MsgBox(opnFile.Read(20))
```



هذا الكود يقرأ 2000 حرف و يتجاوز طول الملف , ولكن لا ينتج عنه أى خطأ . هو فقط يعرض بداخله محتويات الملف (إنظر الشكل)

```
Dim fso, opnFile
Set fso = CreateObject("Scripting.FileSystemObject")
Set opnFile = fso.OpenTextFile("c:\test.txt", 1, True)
MsgBox(opnFile.Read(2000))
```



يغطي هذا المزيد لنفعله مع قراءة وإنشاء الملفات والمجلدات .

Deleting Files and Folders

حذف الملفات والمجلدات

مؤخراً , دعنا نلقى نظرة على كيف ننجز الخطوة الأخيرة فى دورة حياة الملف — حذف الملفات والمجلدات باستخدام الكود . أولاً , دعنا ننظر على كيفية حذف ملف .

Delete Files

حذف الملفات

حذف الملفات فقط يأخذ ثلاثة أسطر من الكود (أقل من الذى نحتاجه إلى إنشاء ملف) الملفات تحذف باستخدام دالة DeleteFile .

```
Dim fso
Set fso = CreateObject("Scripting.FileSystemObject")
fso.DeleteFile("c:\test.txt")
```

السطر الأخير من الكود هو السطر الذى يقوم بالحذف الفعلى , وهنا قد حددنا ان هذا الملف قد حدد للحذف . هذا الأمر يجب أن يحتوى على المسار الذى سيحفظ (إذا لم يكون الملف المحذوف فى نفس المجلد) , وإسم الملف يجب أن يحتوى على إمتداد الملف إذا تم إستخدامه .

يمكن ان يتم تعديل الكود ليقبل إسم الملف من خلال استخدام رسالة التاجيل :


```
Dim fso, fileDel
fileDel = InputBox("Enter filename to Delete:", "File deletion")
Set fso = CreateObject("Scripting.FileSystemObject")
fso.DeleteFile(fileDel)
```

تذكر أن تشتمل على المسار الكامل للملف وإمتداد الملف , كما هو مبين في الشكل



Delete Folders

حذف المجلدات

حذف المجلدات مشابه لحذف الملفات . حذف المجلدات ينفذ بإستخدام دالة DeleteFolder . الكود مشابهة للكود المستخدم لحف الملف .

```
Dim fso
Set fso = CreateObject("Scripting.FileSystemObject")
fso.DeleteFolder("c:\testfolder")
```

هذه الدالة لا تميز بين المجلدات التى بها محتويات والتى ليس بها وسيحذف كلاهما .

Summary

الملخص

في هذا الفصل , لقد ألقيت نظرة على كود يستخدم للدخول على نظام الملفات . ونظرنا إلى دورة حياة الملف للملفات في العادة وإستخدمنا هذا كقالب لتصنيف المهارات التي نحتاجها لنعمل بفعالية مع الملفات في نظام الملفات .

رأيت كيف تستخدم VBScript خلال Windows Script Host :

- إنشاء ملفات .
- إنشاء مجلدات .
- تحرير ملفات .
- حذف ملفات .
- حذف مجلدات .

كل هذا مفتاح مهارة للتجربة والخبرة , وكذلك عملك من خلال الأمثلة في هذا الفصل , أرى ان تقوم بالتجربة والتعديل في الكود مع الأمثلة المزودة وأن تعمل على مشاريعك الخاصة المتوسطة لتحسين مهاراتك وثقتك بنفسك — فقط خذ حذرك عندما تأتي لتحرير وحذف الملفات ! .

13

The Windows Registry

هذا الفصل يبحث في أحد أسرار نظام التشغيل windows : Windows Registry . الحصول على أساسيات هذا المخزون من المعلومات الواردة داخل نظام التشغيل سيعطيك السبق في مجال البرمجة , خاصة إذا كنت تنتهي بالبرمجة بأحدى لغات البرمجة "المرئية" المصممة لإنشاء تطبيقات للـ windows .

The Windows Registry

قبل أن ننتقل لنلقى نظرة على Windows Registry , دعنا نوضح أكثر هنا أننا نتحدث عن شئ يطبق فقط في أنظمة التشغيل Windows . ولا يطبق على أحد الأنظمة الأخرى مثل Mac,Unix,Linux .

انظمة التشغيل windows التى تستخدم Registry هى :

- ☐ Windows 95
- ☐ Windows 98
- ☐ Windows Me
- ☐ Windows NT 4
- ☐ Windows 2000
- ☐ Windows XP
- ☐ Windows CE
- ☐ All Pocket PC operating systems

لا يطبق الـ Registry على نظام التشغيل DOS .

What Is the Windows Registry?

ما هو Windows Registry ؟

Registry كان فى الجوار من لحظات الآن , كان اول تقديم له فى نظام windows 95 , ولكن من الغريب كم الناس الذين لا يعرفون عنها او العلم فقط بوجودها بسبب المشاكل التى اعطتها لهم .

هذا المقطع يلقى نظرة عن قرب على ماذا يقوم Registry بالفعل .

Definition

التعريف

هو قاعدة بيانات مركزية تستخدم بواسطة نظام التشغيل windows لتخزين مجموعة وكبيرة من المعلومات عن نظام التشغيل نفسه .

هناك ثلاثة انواع رئيسية من المعلومات فى Registry :

- التطبيقات : يستخدم Registry ليقوم بتخزين كمية هائلة من المعلومات عن التطبيقات . وهذا يشمل , ولكن ليس محدود عليهم :
 - الإعدادات المبدئية الافتراضية .
 - إعدادات المستخدم المخصصة .
 - إعدادات التكوين او مكونات (Configuration) .
 - معلومات عن نوع الملف .
- الأجهزة HardWare : هو أيضاً مفتاح التحكم فى معظم الاجهزة التى للمستخدم الحق فى إستخدامها من خلال windows . يوجد هنا المزيد من المعلومات .
- معلومات عن المستخدم : يمكن ان يكون لديك ملفات تعريفية مختصرة متعددة على أنظمة التشغيل Windows بسبب ان كل المعلومات المتعلقة بالملفات التعريفية بالمستخدم توجد فى الـ Registry . هذا يجعل إنشاء ملف تعريفى جديد او التعديل فى ملف موجود مسبقاً او حذف ملف قديم من أسهل ما يكون .

فيحتوى الـ Registry على المزيد من المعلومات التى يرجع إليها باستمرار نظام التشغيل — إعدادات عامة لكيف يبدو الـ windows وكيف يعمل , ملفات تعريفية للمستخدم , بيانات التطبيقات . وما يستخدمه التطبيقات لفتح ملفات , أو الأجهزة الموجودة على نظام التشغيل , او حتى اى الأيقونات المستخدمه . هذا مزيد من المعلومات يمكن الوصول إليها على الطائر [لم يحضرنى معنى آخر] . وبسبب إستخدام تصميم قاعدة البيانات فى الـ Registry فيزيد من سرعة العملية .

قبل windows Registry , كل هذه المعلومات والإعدادات كانت تخزن فى ملف نصى . الكثير يعرفها بـ ملفات التهيئة initialization Files (بإمتداد .ini) . معظم التطبيقات لديها ملفات التهيئة الخاصة بها مبعثر على النظام فى كل مكان (البعض فى مجلدات التطبيق فى الأسئلة , ولكن فى الحقيقة يمكن أن يكونوا فى أى مكان) وكان هذا بمثابة كابوس لإدارة وتعقب هذه الملفات والسبب فى أسف الكثير عندما يأتى إلى تحديث التطبيقات او صنع تغييرات .

مازال هناك القليل من ملفات .ini فى الحاسبات الحديثة ولكن كمية هائلة من الذى يستخدم فيه يتم فعله بالـ Registry .

ايضاً Registry يستبدل ملفات أخرى النظام الموجوده على انظمة التشغيل السابقة , , إشتمالاً على :

autoexec.bat

config.sys

system.ini

win.ini

إذا قمت ببعض البحث داخل النظام الخاص بك , ربما تجد بعض هذه الملفات ولكن تبقى كملاذ أخير للتطبيقات التي حقاً لا يمكن ان تعمل بدونها .

Windows Registry : يخزن في ملفات ذات نظام ثنائي binary التي لا يمكن تحريرها مباشرةً بدون استخدام تطبيق معين , أيضاً , على الرغم من أن كل الإصدارات من windows منذ Windos 95 يستخدم Registry , التنسيق الذي يأخذه ليس متطابق , على الرغم من أن هناك قواسم مشتركة بينها .

The Layout of the Windows Registry

تخطيط Windows Registry

إضافة إلى انه قاعدة بيانات , فهو منظم في شكل تنسيق متسلسل . وهذا يعني أنه يتبع تخطيط منطقي . عندما ننظر إليه وتصفح (كما يفعل قريباً) لا حظ انه مشابه لتصفح ملفات النظام , حيث أن عندنا مجلدات , ومجلدات فرعية , وملفات .

البيانات التي يحتويها الـ Registry مرتبة منطقياً , يمكنك ان تستخدم معالم قليلة لتجد ما تبحث عنه بسهولة .

أسهل طريقة لتتعرف على الـ Registry هو إلقاء نظرة عليه . لفعل هذا تحتاج ان تعرف نفسك على عرضه وتحريره .

Regedit and Regedit32

أنت تقوم بتحريره طوال الوقت ولكن من المحتمل أنك لا تعرف ان تقوم بفعل هذا . كل مرة تجرى تغيير على تطبيق , تثبيت او إزالة تثبيت لتطبيق , او تعديل ملفات المستخدم التعريفية فانت تجرى تغيير في الـ Registry

من المقبول ان أسهل طريقة لإجراء تغييرات على الـ Registry هي السماح للتطبيقات المثبتة على الحاسب أن تنفذ التغييرات فضلاً عن تحريرها يدوياً . هذه نصيحة جيدة لمعظم الناس , ولكن كمبرمج ستجد في نهاية المطاف أنك تريد أن تخزن بعض المعلومات عن المنتج او عن المستخدم , وربما تأتي إلى إستنتاج إن الـ Registry هـ والمكان المناسب لحفظ هذه المعلومات . لفعل هذا , فتحتاج إلى التعرف على مبادئ الـ Registry . ولفعل ذلك تحتاج إلى التدريب العملي عليه .

هناك برنامجين يمكن أن نستخدمهم لعرض وتحرير الـ Registry ومن المحتمل أن تجدهم مثبتين في الحاسب بالفعل . ويسميان Regedit , RegEdt32

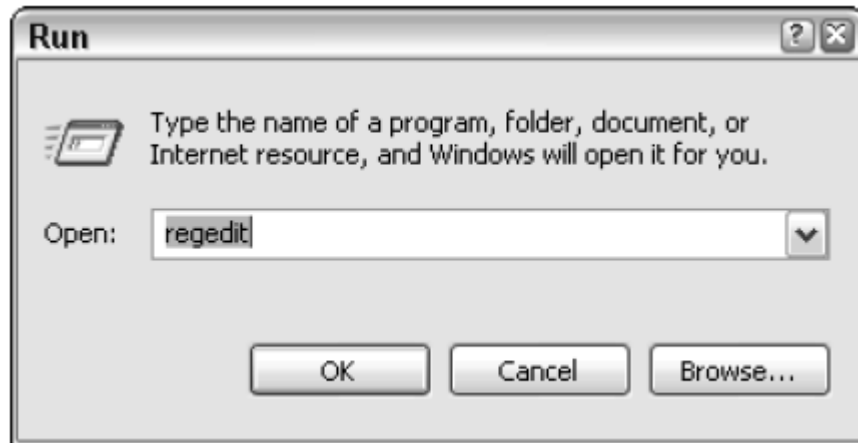
أسهل طريقة لتصل إلى هذه التطبيقات هو استخدام امر Run الموجودة في قائمة Start . اضغط على Start ثم Run . في المربع الذي يعرض لك (كما في الشكل) اكتب :

Regedit

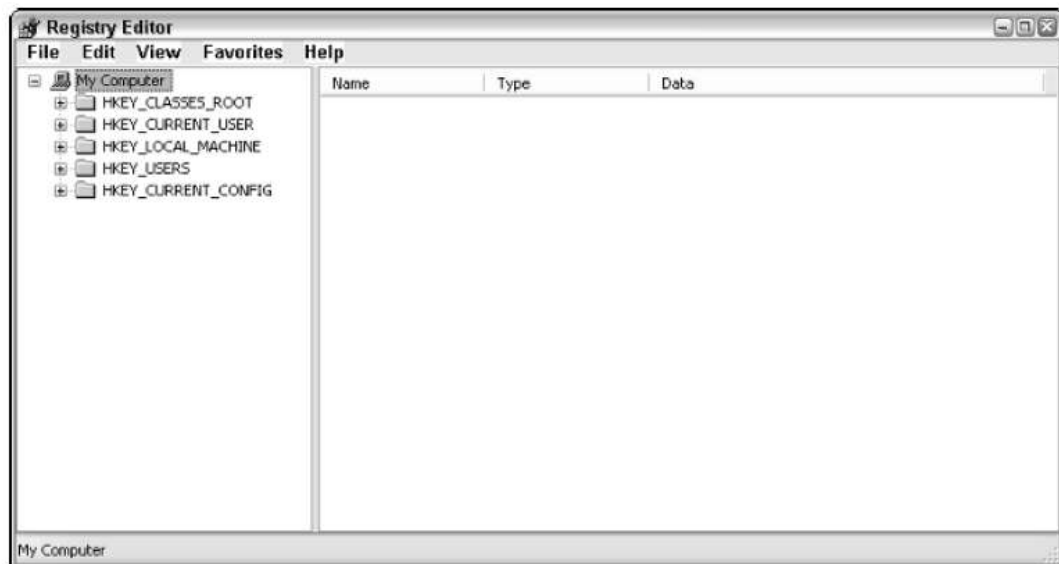
وإضغط Ok لـ Regedit أو اكتب :

regedt32

وإضغط Ok لـ Regedt32 .



البرنامج المحمل يظهر في الشكل . في windows XP , لا يهم إن كتب Regedit أو Regedt32 لأن windows NT هو مجرد ملف صغير يقوم بتحميل Regedit.exe . مع ذلك , في windows 2000 و , وأرى ان تستخدم regedt32 : إنظر الشكل



في Windows 95 و 98 و windows ME إستخدم Regedit .

[في Windows 7 يمكنك ان تستخدم الإثنين Regedit , Regedt32]

بإستخدام هذا المحرر يمكنك أن تبجر داخله بإستكشاف التفريعات التي أمامك . مع ذلك , قبل النظر إلى هذا , فكرة جيدة أن نعرف كيف يتم إسترجاع الـ Registry .

Backing Up the Registry

لأنه مفتاح التشغيل السليم لنظام التشغيل windows . من الهام أن تأخذ حذرك منه وحمايته من التلف . أفضل طريقة لفعل هذا أن تحفظ نسخة احتياطية منه . دعنا نلقى نظرة سريعة على كيف نقوم بعمل نسخة احتياطية الـ Registry لأنظمة تشغيل متنوعة .

Windows XP

عند عمل نسخة احتياطية للـ Registry في Windows xp . لديك إختيارين متاحين :

- عمل نسخة احتياطية لمدخلات محددة (مثال Subkeys) . وهذا ينشأ عنه ملف صغير ومحدد من السهل إستبداله .
- عمل نسخة احتياطية لكل الـ Registry . وهذا ينشأ عنه ملف كبير يحتوى على كل الـ Registry . إستبداله يكون كبير , وتشارك أكثر من عملية .

Back up Subkeys

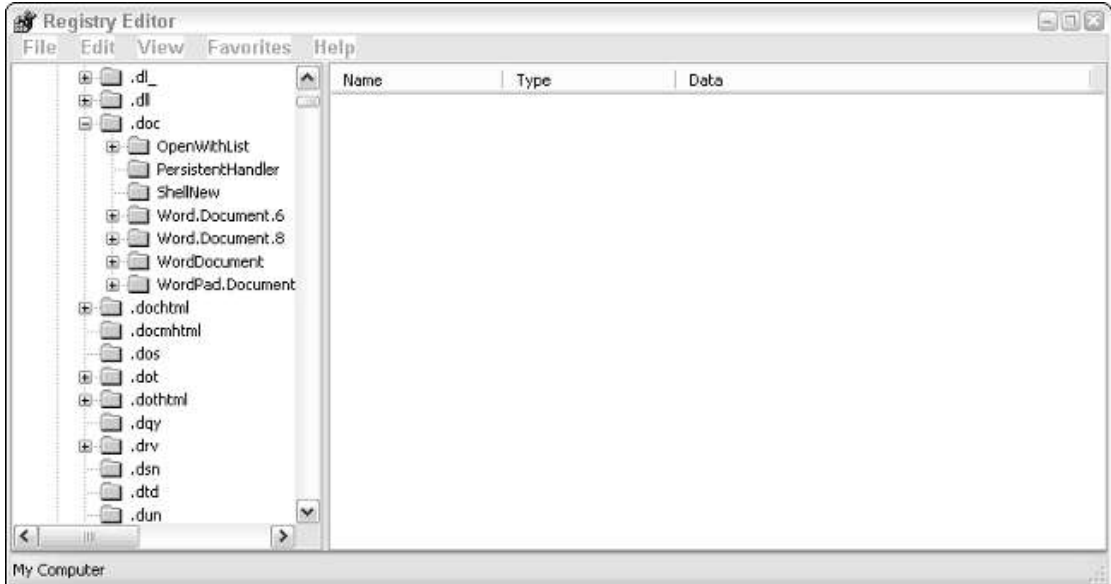
يمكن ان تستخدم الخطوات التالية لتصدير Registry Subkey قبل تحريره .

- 1 - اضغط Start , ثم اضغط Run .
- 2 - فى المربع المفتوح , أكتب Regedit .

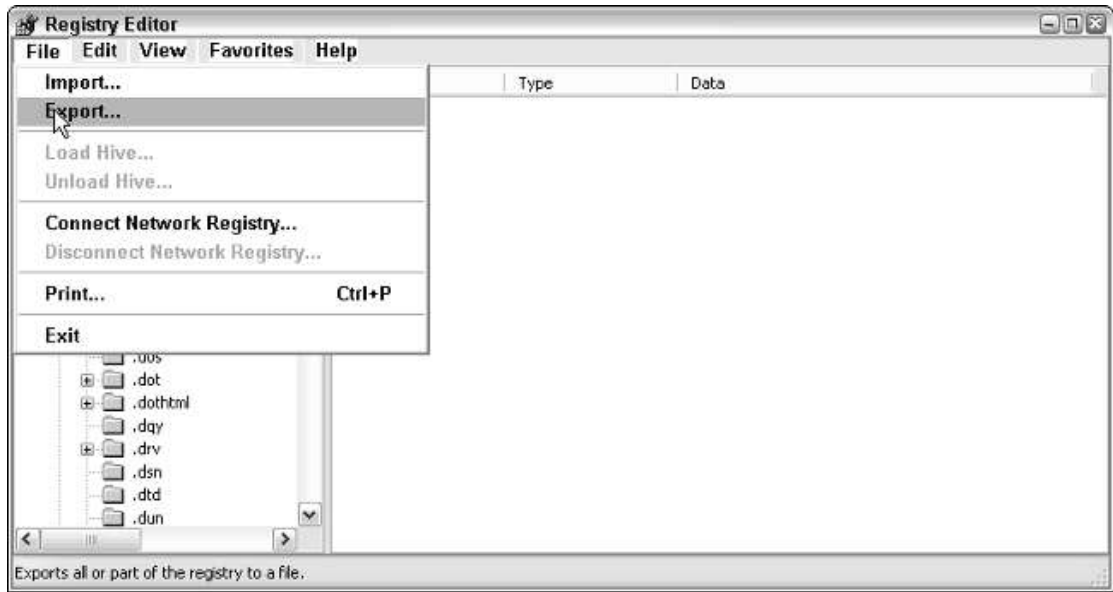
regedit

ثم اضغط OK .

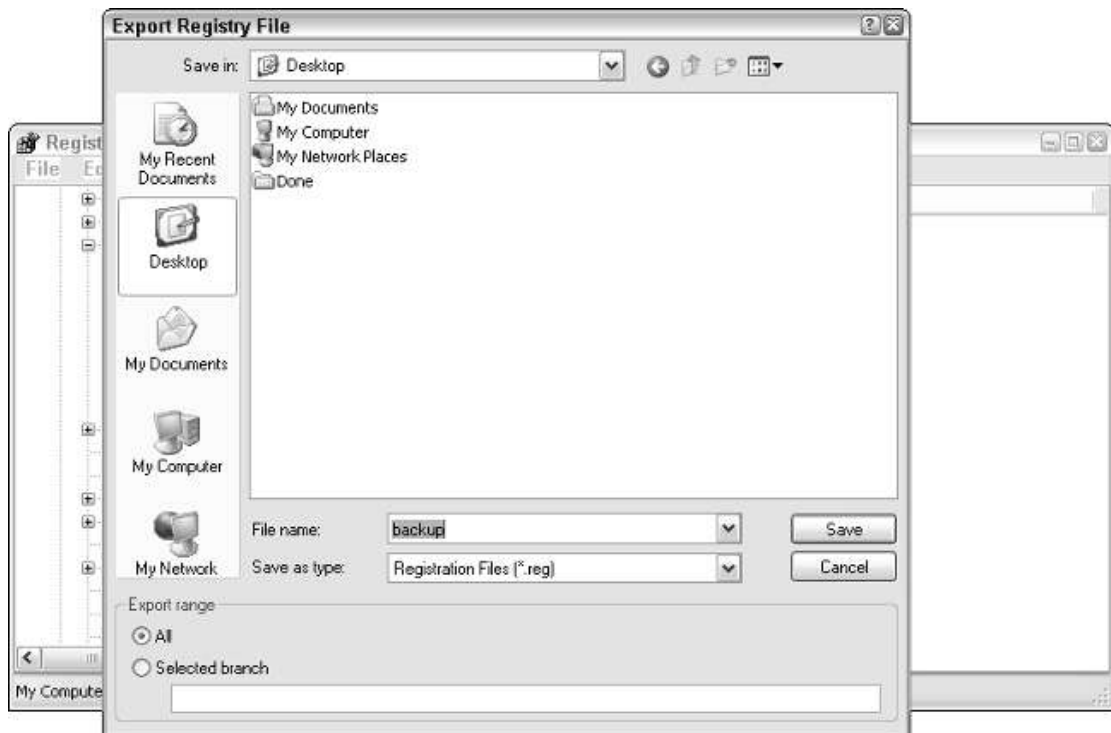
- 3 - إبحر خلاله بالضغط على المفاتيح والمفاتيح الفرعية (إنظر الشكل)



- 4 - حدد ثم اضغط على المفتاح الفرعى الذى يحتوى على القيمة التى تريد ان تحررها .
- 5 - فى قائمة File , اضغط على Export (إنظر الشكل)



- 6 - في مربع الحفظ , حدد الموقع حيث تريد أن تحفظ مدخلات ملف التسجيل (.reg) .
 اكتب اسم الملف في مربع اسم الملف . ثم اضغط إحتفظ (إنظر الشكل)



- 7 - لا تستخدم هذه الطريقة لعمل نسخة إحتياطية للكل Registry — تبذروا فاشلة .

Back Up the Whole Registry

عمل نسخة إحتياطية لكل الـ Registry

إستخدام اداة Backup لعمل نسخة إحتياطية .

لابد أن يكون لديك تصريح كـ Administrator أو Backup Operator على حاسبك لعمل نسخة إحتياطية من الملفات والمجلدات .

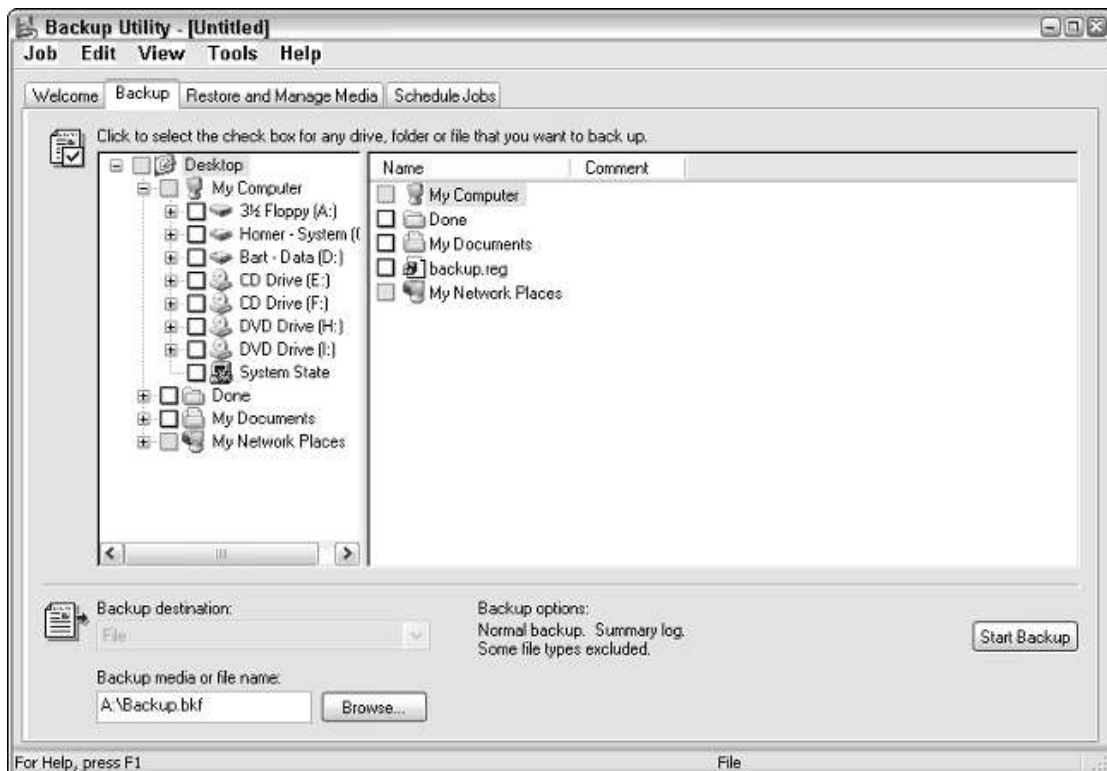
1 - إضغط Start , أشر إلى All Porgrams , ثم إلى Accessories , ثم إلى System Tools ومن ثم إضغط Backup . فتبدأ نافذة عمل نسخة إحتياطية أو إسترجاع فى الظهور .



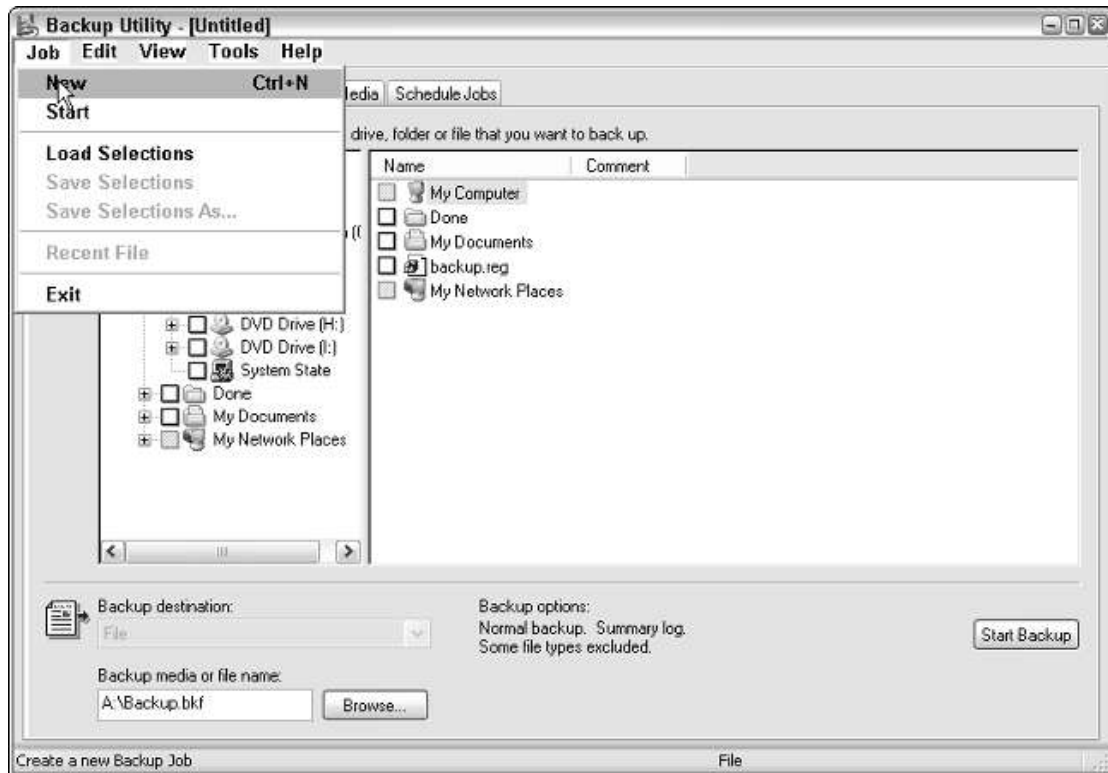
2 - التالى , إضغط على Advanced Mode (أنظر الشكل)



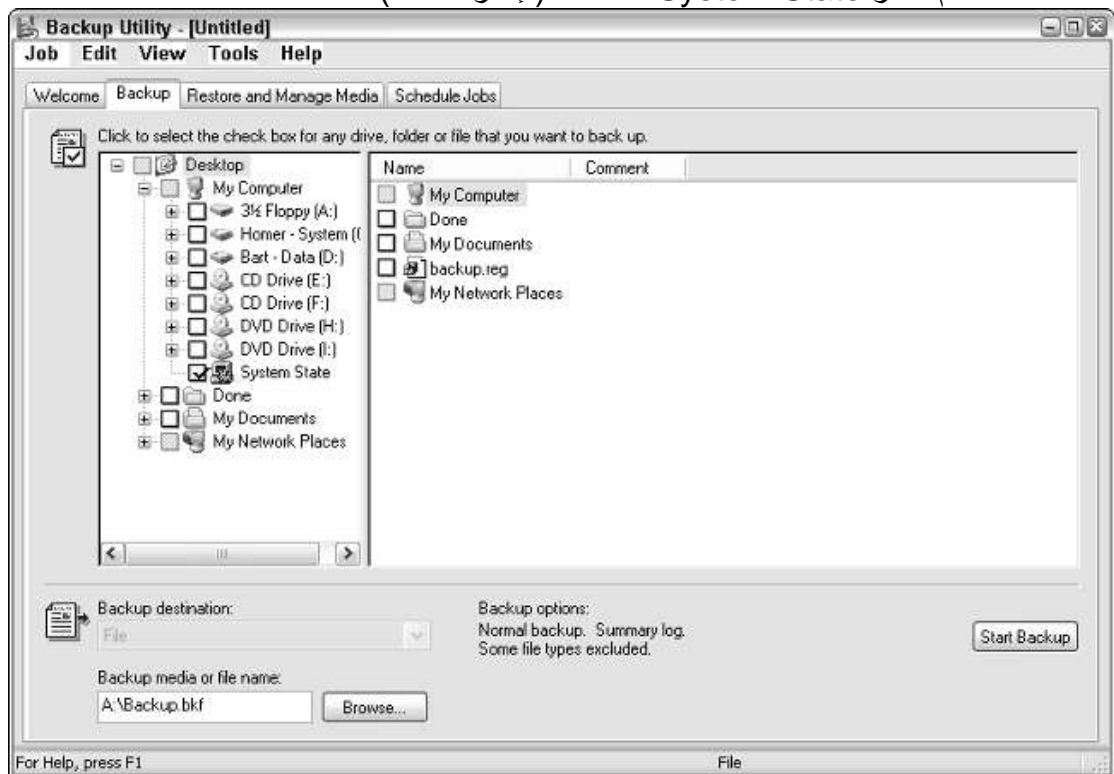
3 - إضغط على الجزء الخاص بـ Backup (انظر الشكل)



4 - فى قائمة Job , إضغط New (إنظر الشكل)

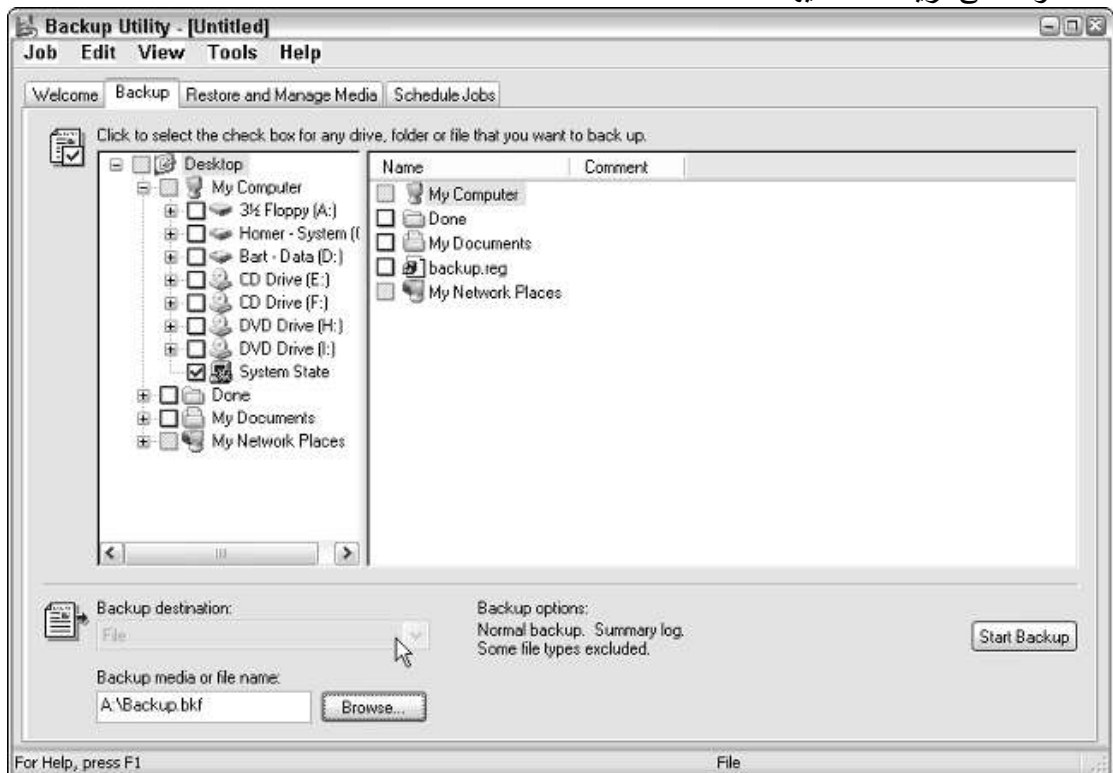


5 - حدد العلامة امام الأمر System State لتحدها (انظر الشكل)

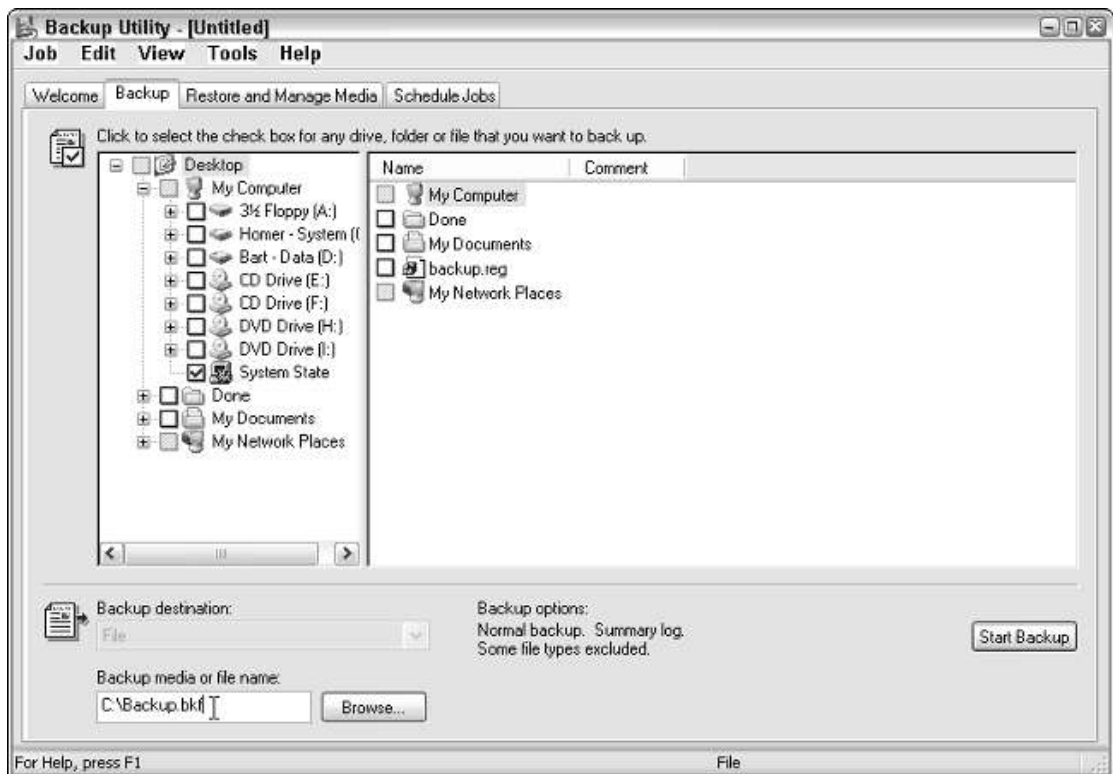


بيانات System State تشتمل على عدة عناصر مثل Windows Registry , قاعدة بيانات التسجيل Com+ . ملفات تحت حماية ملف windows , وملفات الإقلاع .

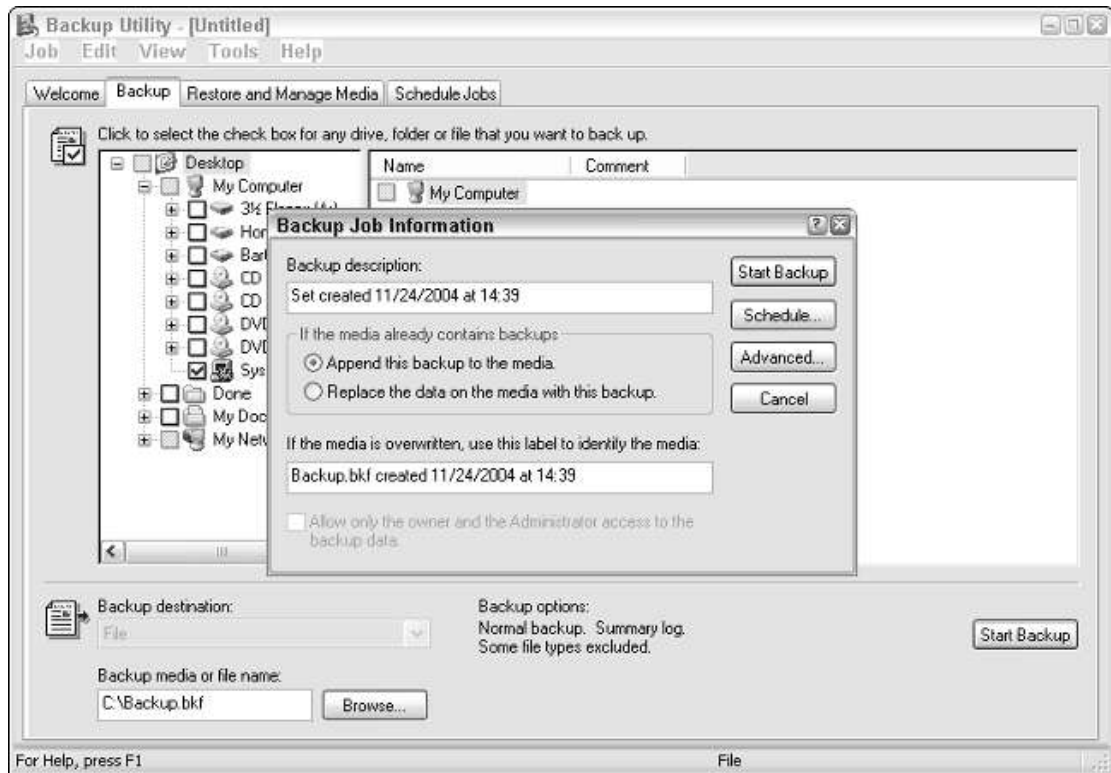
6 - في قائمة الجهة المقصودة للحفظ النسخة الاحتياطية , إضغط على Backup Destination أى الجهة المقصودة التى تريد الحفظ فيها .



7 - إذا قمت بالضغط على File فى الخطوة السابقة , أكتب المسار الكامل وإسم الملف الذى تريد أن فى خانة Backup media or filename box



8 - اضغط على Start Backup . يسبب ظهور مربع حوارى لمعلومات وظيفة Backup . كما مبين بالشكل



9 - تحت الأمر If the media already contains backups, استخدم أحد الخطين التاليين

○ إذا أردت أن تلحق هذه النسخة بالنسخة السابقة , اضغط على الأمر Append This Backup to The media

○ إذا أردت ان تستبدل النسخة السابقة بهذه النسخة, اضغط على الأمر Replace the data on the media with this backup

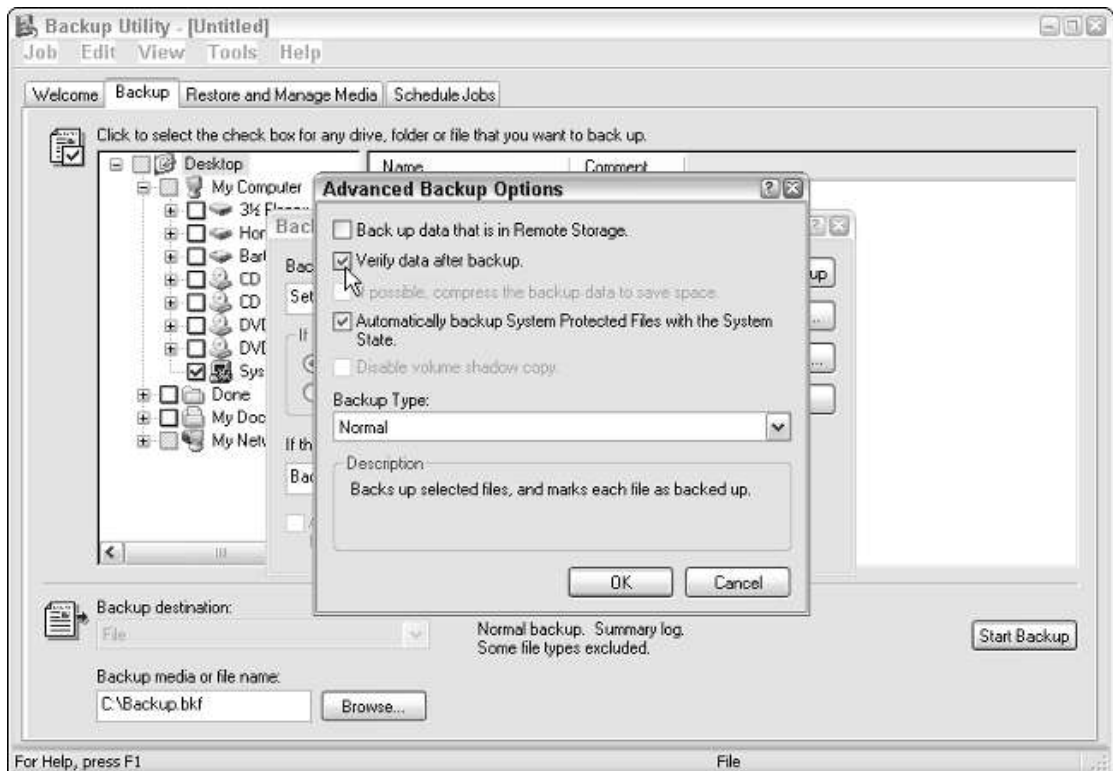
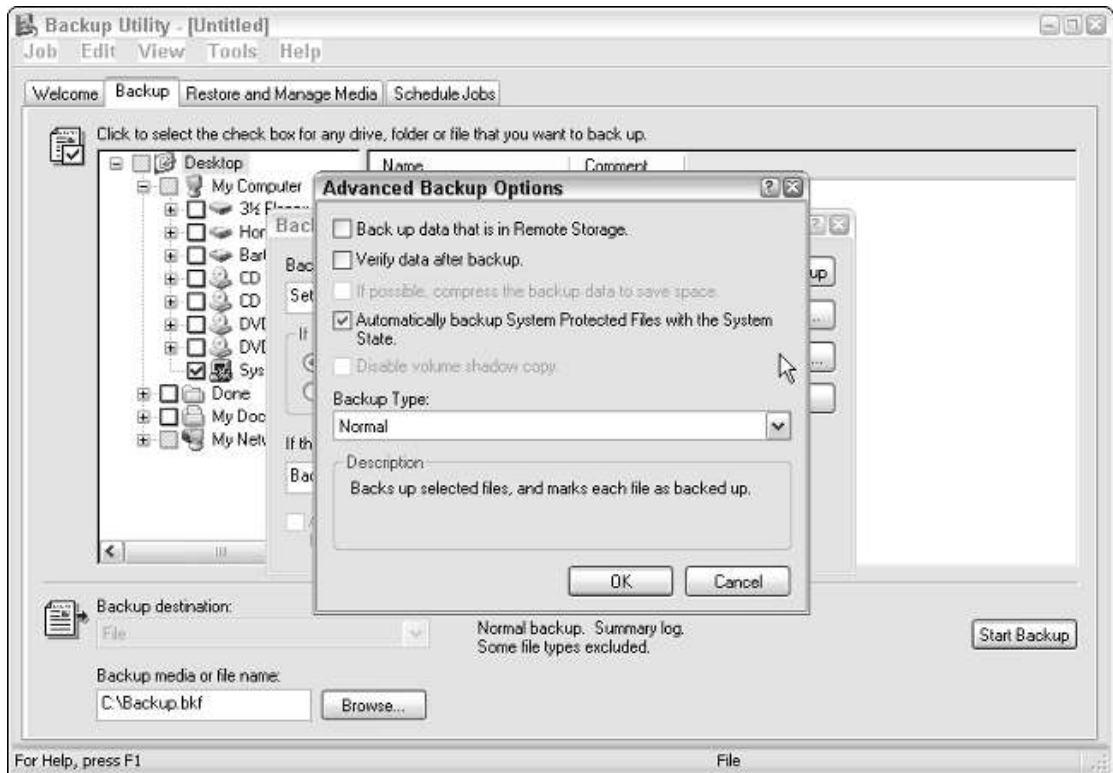
10 - اضغط على Advanced .

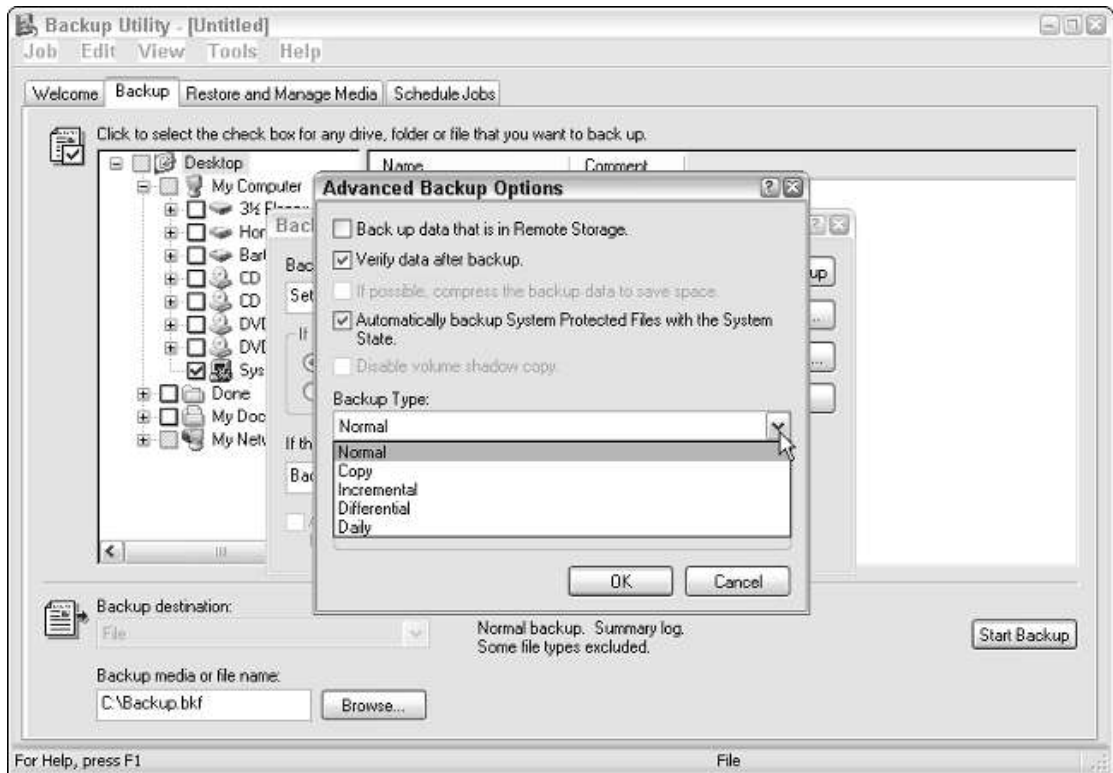
11 - حدد الأمر Verify data after backup (الشكل التالي)

12 - فى خانة Backup Type , اضغط على نوع النسخ الذى تريده . عندما تضغط على Backup Type , وصف لنوع النسخ يظهر تحت كلمة description (انظر الشكل الذى يليه) .

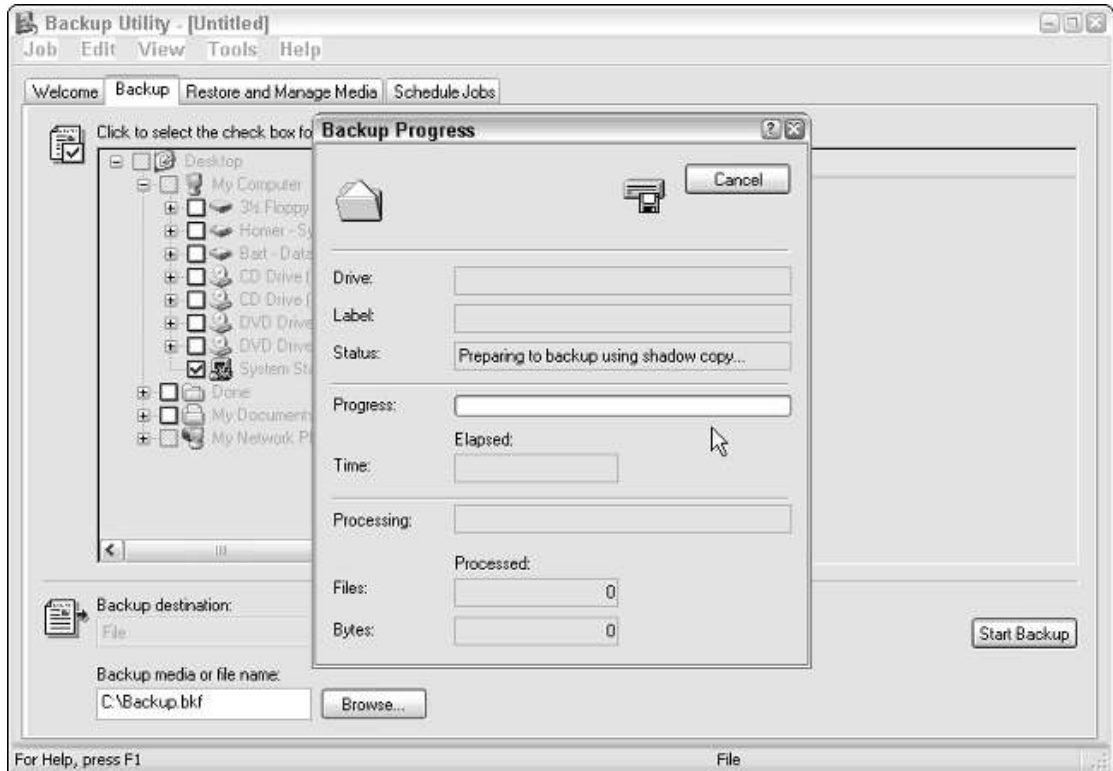
الإختيارات المتاحة لك هى :

- ☐ Normal
- ☐ Copy
- ☐ Differential
- ☐ Incremental
- ☐ Daily





13 - اضغط على OK , ومن ثم اضغط على Start Backup , فيظهر مربع حوارى لتقدم عملية النسخ , وتبدأ عملية النسخ (انظر الشكل)



14 - عند إتمام عملية النسخ , اضغط على close .

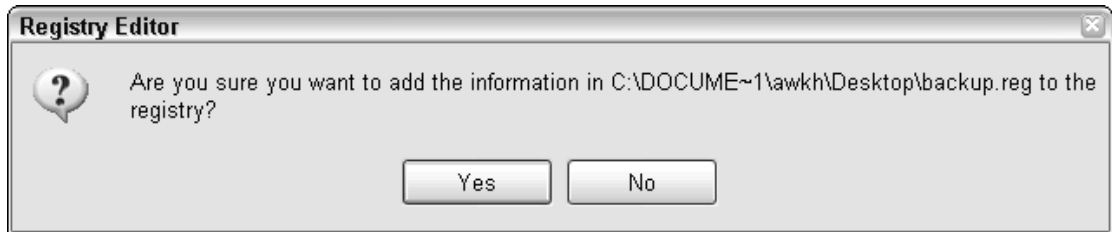
Restoring the Registry

عملية إسترجاع النسخة [أى العكس]

كيفية إسترجاع الـ Registry تعتمد على الطريقة التى إختترتها فى عملية عمل Backup .

Restoring Subkeys

لإسترجاع Registry Subkeys الذى قمت بتصديره , بالضغط مرتين على ملف مدخلات التسجيل (.reg) الذى قمت بحفظته . سيعرض لك رسالة تأكيد , فتحتاج أن تضغط على Ok لتكملة عملية الإستعادة او الأسترجاع , كما هو مبين بالشكل .

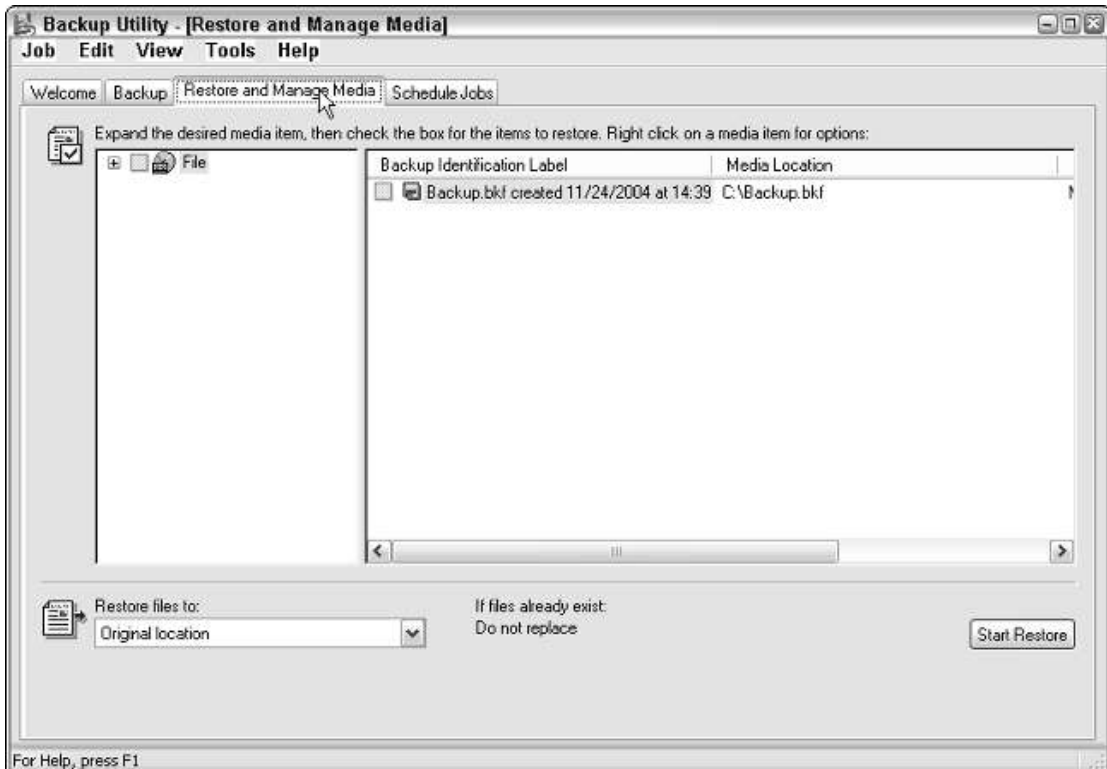


Restoring the Whole Registry

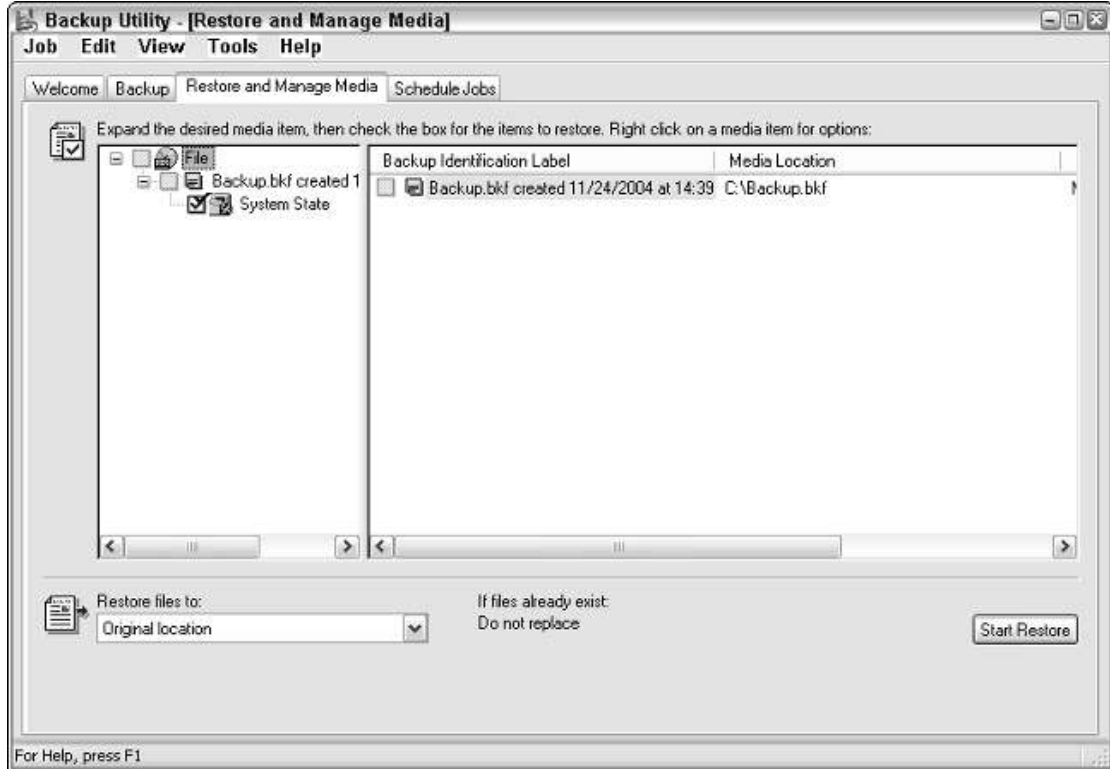
إسترجاع كل الـ Registry

إليك دليل خطوة بخطوة إستعادة الـ Registry :

- 1 - إضغط start , ثم All programs , ثم Accessories , ثم System Tools , ثم Backup , وسيبدأ لك بعملية إسترجاع او عمل النسخة الإحتياطية .
- 2 - إضغط على Advanced Mode
- 3 - إضغط على الجزء الخاص Restore and Manage Media فى النافذة . كما بالشكل .



- 4 - إستخدام أحد الخطوات التالية :
- إذا أردت ان تسترجع من ملف نسخة احتياطية , إضغط على الأمر File , وحدد ملف النسخة التي تريد . (انظر الشكل)



- إذا كنت تريد الإستعادة من الجهاز المخصص للنسخة الإحتياطية , حدد مكان هذا الجهاز وحدد الإسم الذي تريد أن تستخدم .
- 5 - في حالة الحاجة لإستعادة الـ Registry , حدد مربع التحديد الخاص بالأمر System State . (انظر الشكل)
- 6 - في مربع Restore File to , إختار Original location .
- 7 - إضغط على Start Restore .
- 8 - ستستقبل الرسالة التالية , وستحتاج أن تضغط على Ok لأنك تريد ان تستعيد معلومات عن حالة النظام مع النسخة الإحتياطية المحددة .
- Warning**
Restoring System State will always overwrite current System State unless restoring to an alternative location.
- 9 - في رسالة تأجيل التأكيد للإستعادة , إضغط OK , سيظهر لك مربع توضيح تقدم سير العملية لتعلن بداية الإستعادة .
- 10 - عند إتمام عملية الإستعادة , إضغط Close . فعندما تظهر لك رسالة تأجيل تسألك غلق الحاسب وإعادة تشغيله , تذكر أن تضغط Yes .

إمتلاك نسخة إحتياطية من Registry موصى به قبل تصفحه , المستخدمين الخبراء يرون أن ذلك غير ضروري , ولكن إذا كان لديك أى شك أو عدم تأكد , قم بعمل نسخة إحتياطية .

Working with the Registry

العمل مع Windows Registry يشمل على هذه الستة أنواع من الإجراءات . كلاً منهم يناقش بالتفصيل فى الصفحات التالية :

- ☐ Finding a subtree, key, subkey, or value
- ☐ Adding a new subkey
- ☐ Adding a new value
- ☐ Changing an existing value
- ☐ Renaming an existing subkey or a value
- ☐ Deleting an existing subkey or a value

[في الحقيقة أثرت عدم ترجمتهم , حتى لا يفقدوا المعنى ولكن المقصود هو إضافة وتغيير قيمة وإعادة تسمية وحذف سواء كانت قيمة أو ما يسمى Subkey والفكرة انه يشبه المتغير وله قيمة وهي Value فتدور الدائرة حول تحريرهم من خلال الأوامر التي ذكرناها]

Finding a Subtree, Key, Subkey, or Value

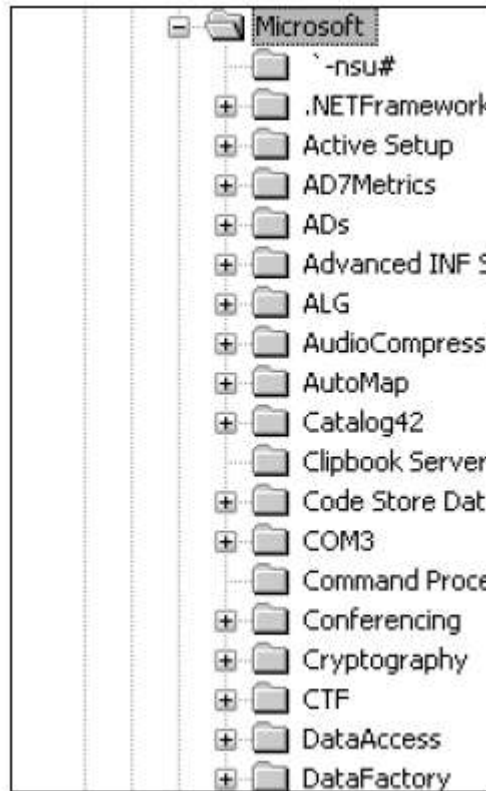
هناك خمسة تفرعات بشكل شجري , وكلأ منهم يبدأ بالبادئة HKEY . في المثال التالي (انظر الشكل) ،
HKEY_LOCAL_MACHINE هي SubTree ، SOFTWARE هي key ، Microsoft هي Subkey .

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft



في محرر Registry , يمكن ان تبحث من خلال Keys و SubKeys بنفس الطريقة التي تتصفح بها الملفات والمجلدات في windows Explorer .

Keys و subkeys قد تم إدراجهم في شجرة المجلدات في اللوحة جهة اليسار للمحرر(انظر الشكل) . إذا ضغطت على Key أو SubKey في جهة اليسار, ستظهر معلومات على الإسم والنوع والبيانات في اللوحة جهة اليمين .(انظر الشكل)



Name	Type	Data
ab(Default)	REG_SZ	(value not set)

كما في Windows Explorer , كل المجلدات يمكن ان نستكشف ما بداخلها بالضغط على علامة (+) وبعد أن يتم التفرع نجد هذه العلامة أستخدمت بعلامة سالبة (-). والضغط عليها سيعيد إخفاؤهم مرة أخرى.

عندما اقول انك لا بد ان تفرع عنصر ما , فاقصد أن تضغط على علامة الموجب المجاورة للعنصر.

لتحديد موقع المفتاح الفرعي Microsoft الذي ذكرته مسبقاً , إتبع الخطوات .

- اضغط على Start , ثم Run , اكتب Regedit واضغط OK .

- قم بتفريع الأمر HKEY_LOCAL_MACHINE

- قم بتفريع الأمر SOFTWARE .

- اضغط على Microsoft .

عندما تضغط على المفتاح الفرعي Microsoft , تظهر لك القيمة فى اللوحة جهة اليمين , إذا أردت ان تعرض المستوى الأقل للمفاتيح الفرعية , إذاً فحتاج أن تقوم بتفريع Microsoft . إذا أردت ان نحدد موقع القيمة , اضغط على المفتاح الفرعى الذى يحتوى على القيمة . ثم إعرض المحتويات فى اللوحة جهة اليمين .

Adding a New subkey

إضافة مفتاح فرعى

الآن سنقوم بإضافة مفتاح فرعى جديد . وسنقوم بتسميته NewTestSubkey وسنقوم بإضافته إلى المفتاح الفرعى Microsoft .

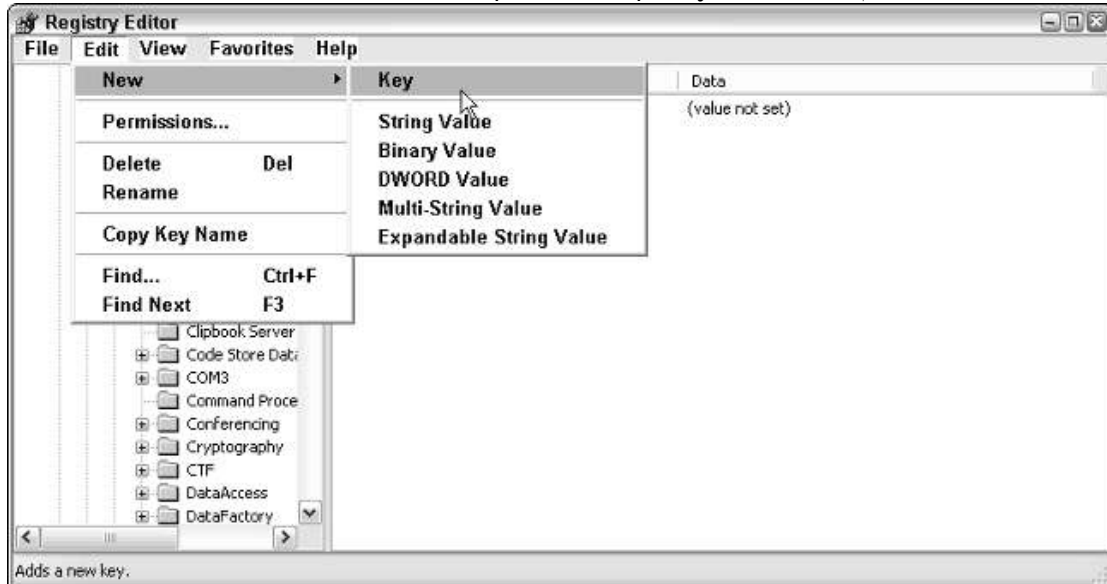
لفعل هذا نتبع الخطوات :

1 - قم بتفريع HKEY_LOCAL_MACHINE.

2 - قم بتفريع SOFTWARE .

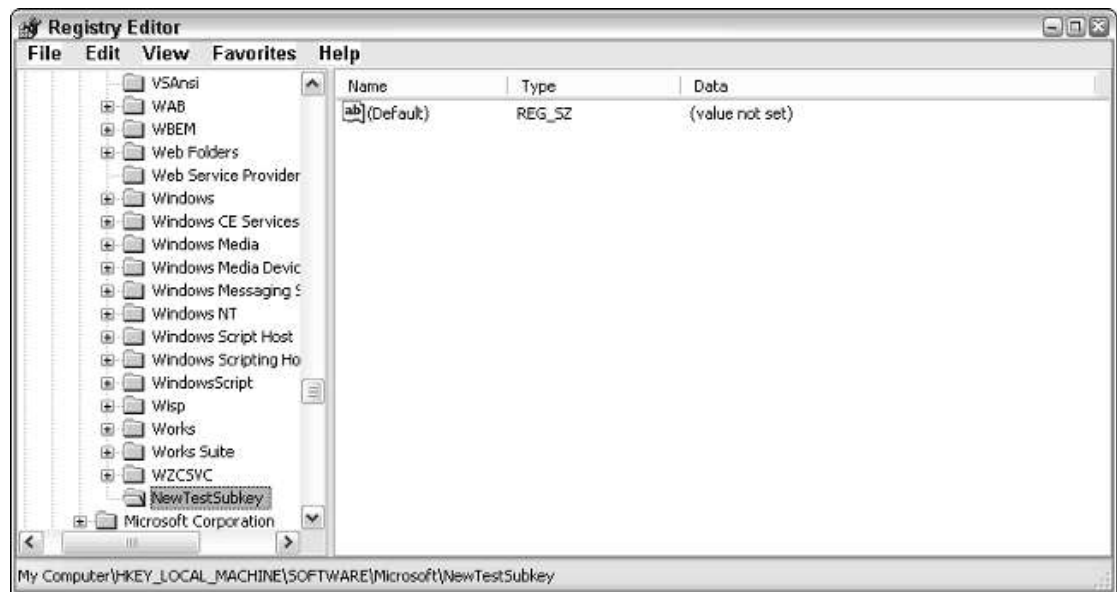
3 - اضغط على المفتاح الفرعى Microsoft .

4 - فى قائمة New , Edit , ثم اضغط على Key (انظر الشكل)



5 - إكتب NewTestSubkey , ثم اضغط على Enter . الآن قد تم إنشاء المفتاح الجديد .

ليس هناك أى مزايا للحفظ — كل التغييرات اقد تم تنفيذها فى الحال. وهذه احد المزايا التى تجعلها أكثر خطورة .

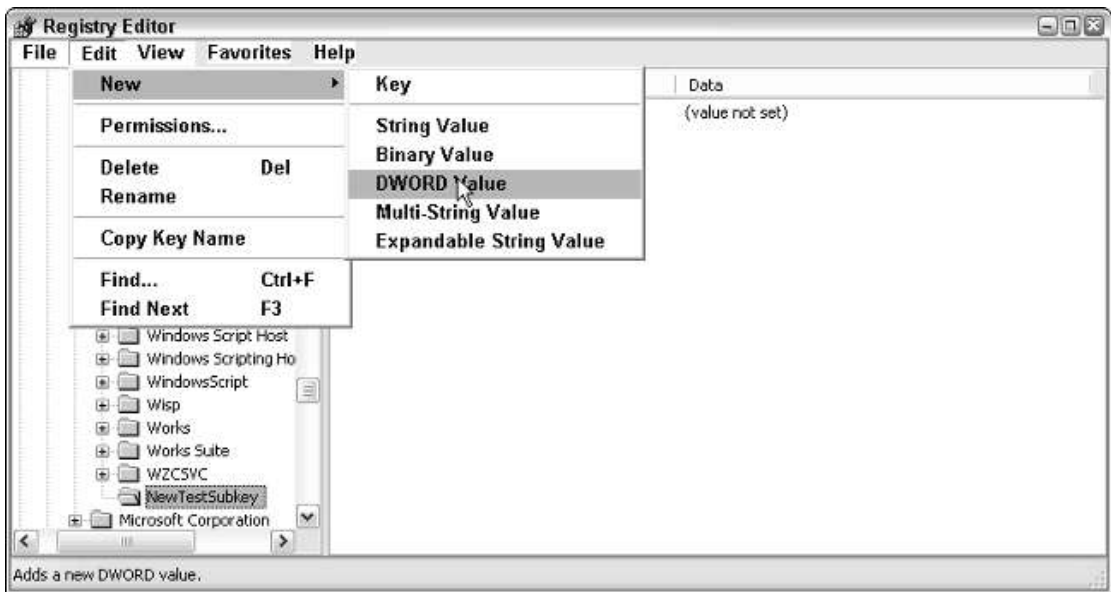


Adding a New Value

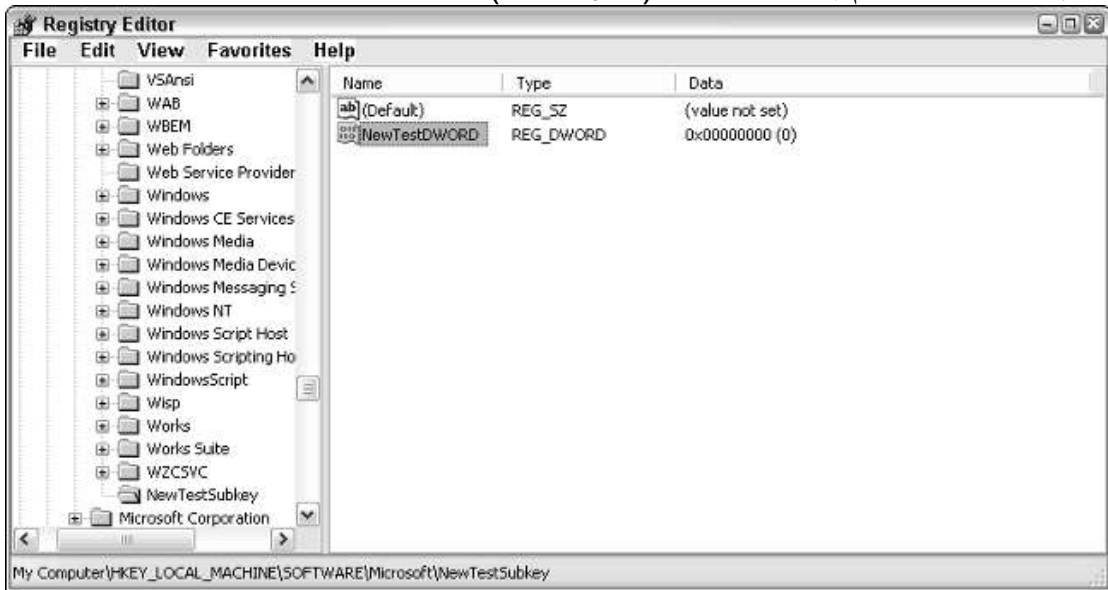
الآن سنقوم بإضافة قيمة جديدة. وسنقوم بإضافة Dword ويسمى NewTestDWORD ونعطيها القيمة 1 في المفتاح NewTestSubkey الذي أنشأها .

Dword تأتي من Double Word , التي تعني جزء من البيانات وهي 4Bytes , وتحجز للأرقام في الـ Registry

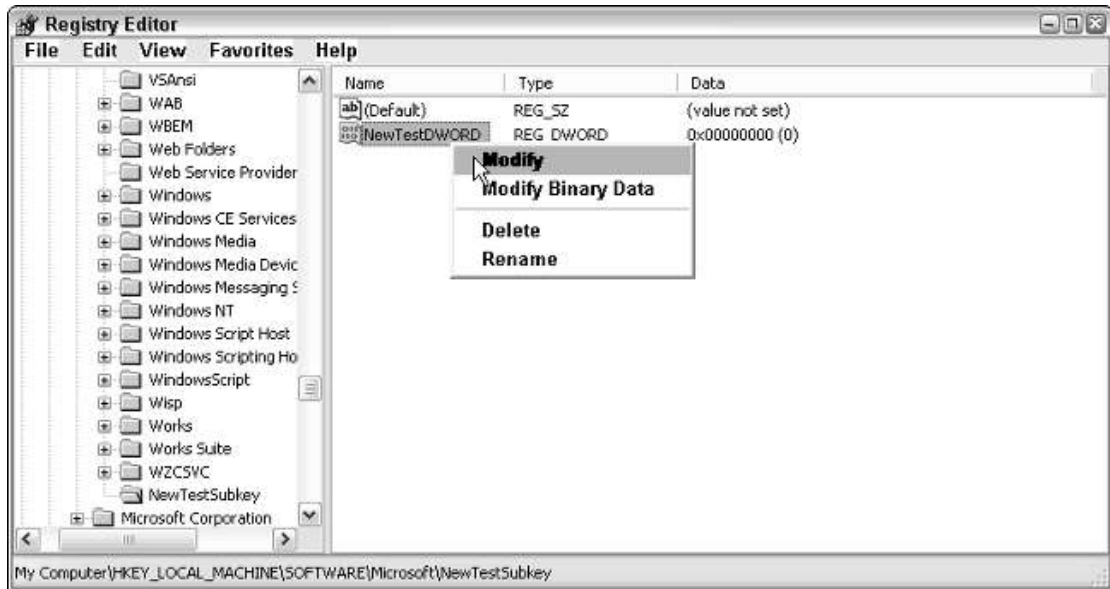
- 1 - قم بتفريع HKEY_LOCAL_MACHINE.
- 2 - قم بتفريع SOFTWARE.
- 3 - قم بتفريع Microsoft
- 4 - اضغط على المفتاح الفرعي NewTestSubkey .
- 5 - في قائمة Edit , New , ثم اضغط على Dword (انظر الشكل)



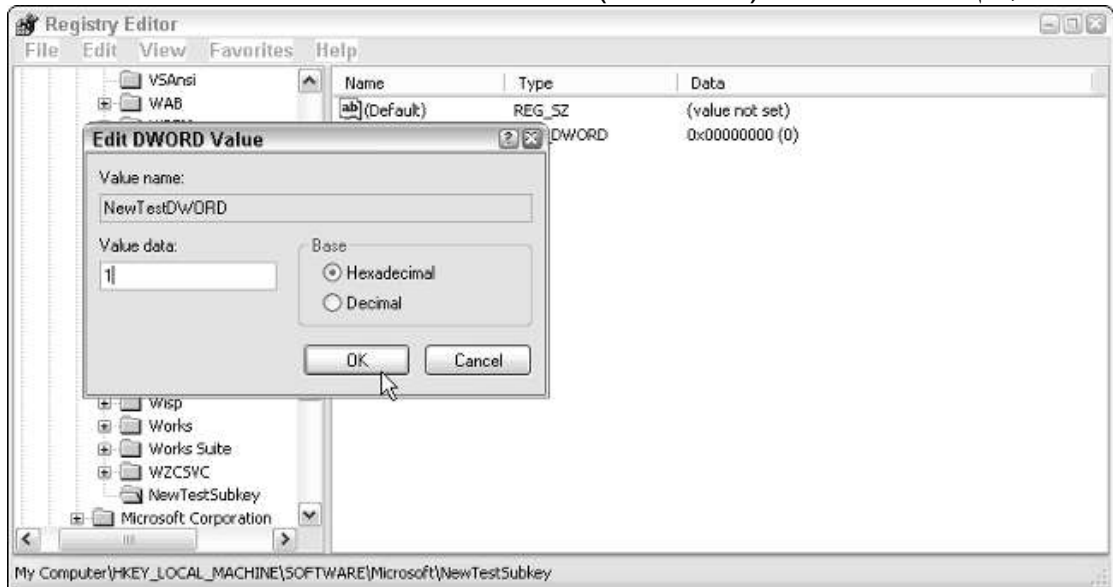
6 - اكتب NewTestDWORD, ثم اضغط Enter (انظر الشكل)



7 - قم بالضغط على المفتاح الأيمن للفأرة على NewTestDWORD , ثم اضغط Modify (انظر الشكل)



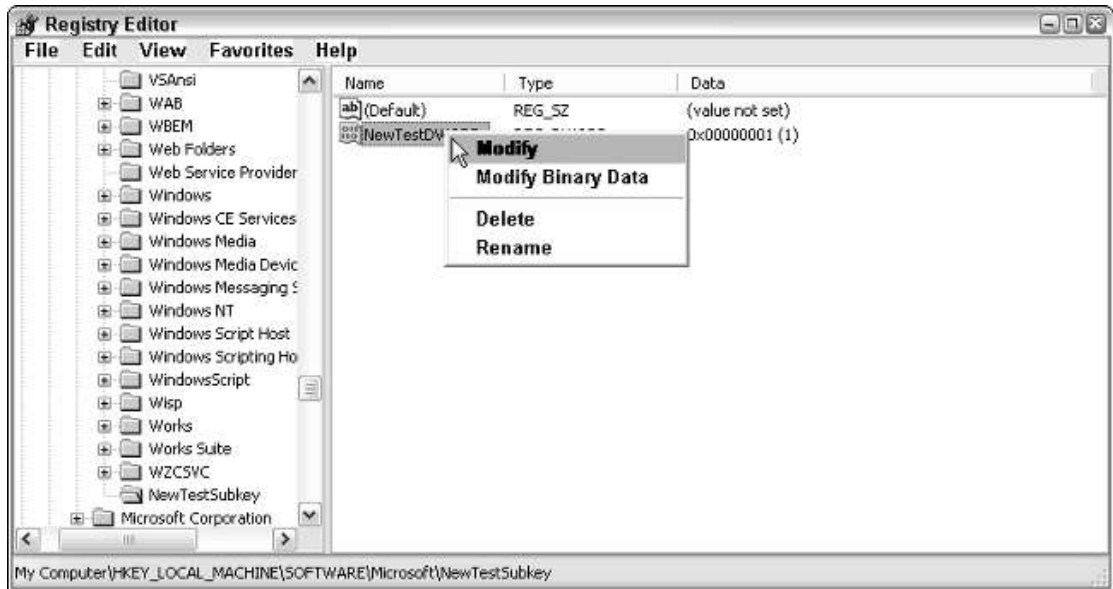
8 - اكتب 1 , ثم اضغط على Ok (انظر الشكل)



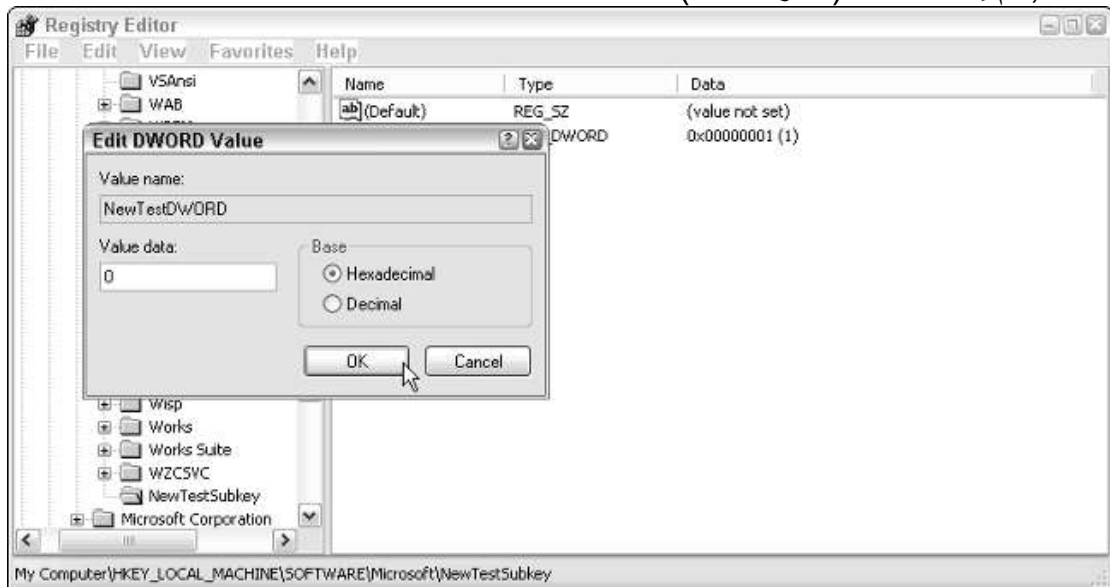
Changing an Existing Value

الان دعنا نلقى نظرة على كيفية تغير قيمة موجودة مسبقاً . لنقوم بتغير بيانات قيمة للـ NewTestDWORD إلى 0 في NewTestSubkey , إتبع الخطوات :

- 1 - قم بتفريع HKEY_LOCAL_MACHINE.
- 2 - قم بتفريع SOFTWARE.
- 3 - قم بتفريع Microsoft
- 4 - اضغط على المفتاح الفرعي NewTestSubkey .
- 5 - اضغط بالمفتاح الأيمن على NewTestDWORD , ثم اضغط على Modify (انظر الشكل) .



6 - اكتب 0 , ثم اضغط OK (انظر الشكل)

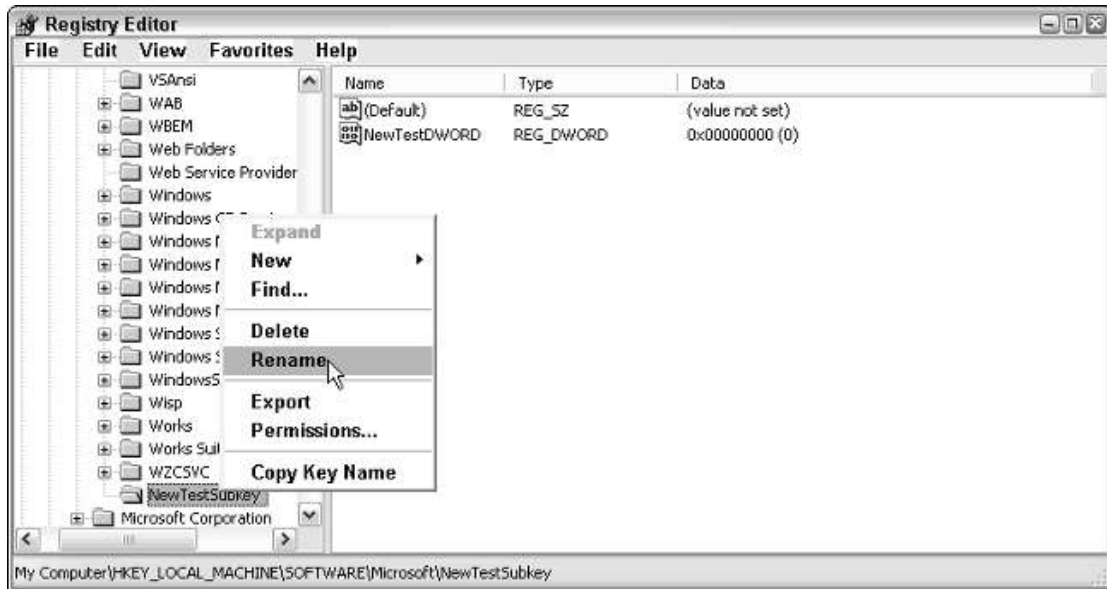


Renaming an Existing Subkey or a Value

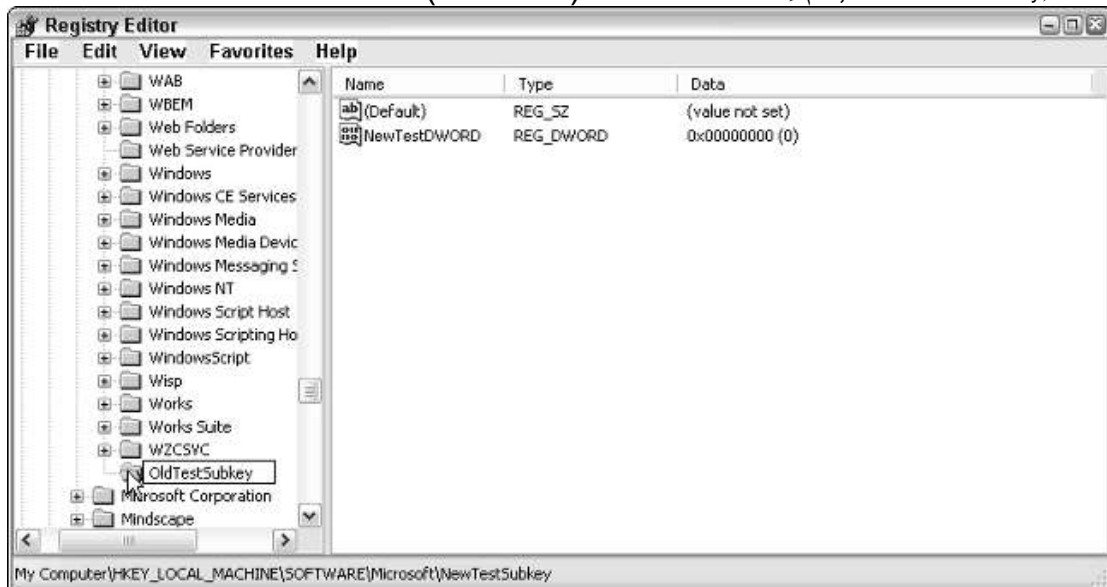
إعادة تسمية مفتاح فرعي أو قيمة .

دعنا الآن نلقى نظرة على كيف نعيد تسمية المفتاح الفرعي NewTestSubkey إلى OldTestSubkey , ولنفعل هذا نتبع الخطوات :

- 1 - قم بتفريع HKEY_LOCAL_MACHINE.
- 2 - قم بتفريع SOFTWARE.
- 3 - قم بتفريع Microsoft
- 4 - اضغط بالمفتاح الأيمن على NewTestSubkey , ثم اضغط على Rename (انظر الشكل) .



5 - اكتب OldTestSubkey , ثم اضغط ENTER (كما بالشكل)



Deleting an Existing Subkey or a Value

حذف مفتاح فرعى أو قيمة .

اخيراً , لنلق نظرة على كيف نقوم بحذف NewTestDWORD فى المفتاح الفرعى OldTestSubkey

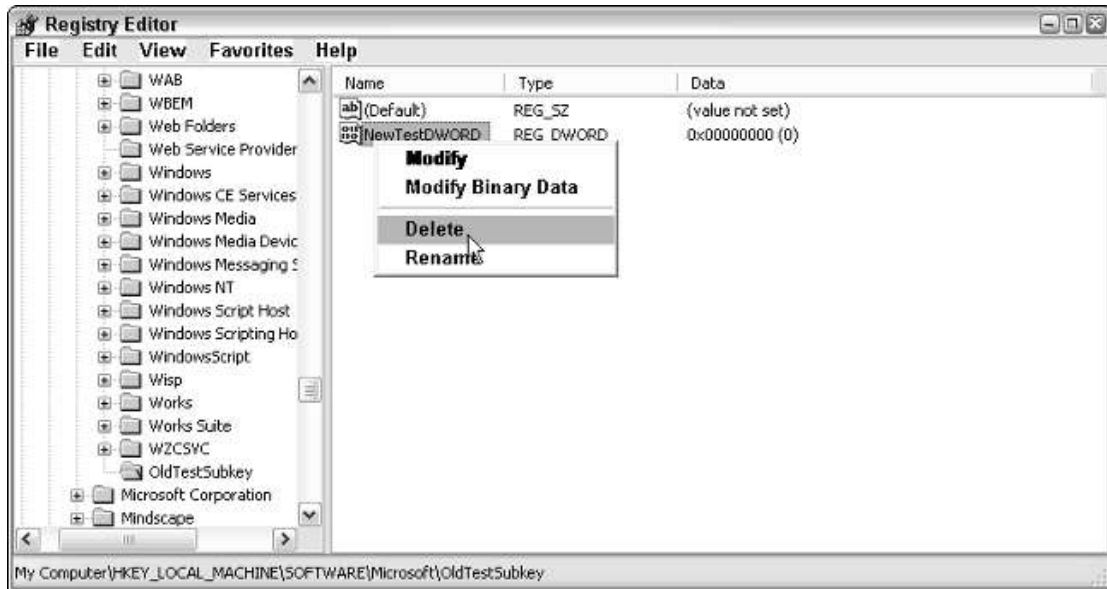
1 - قم بتفريع HKEY_LOCAL_MACHINE.

2 - قم بتفريع SOFTWARE.

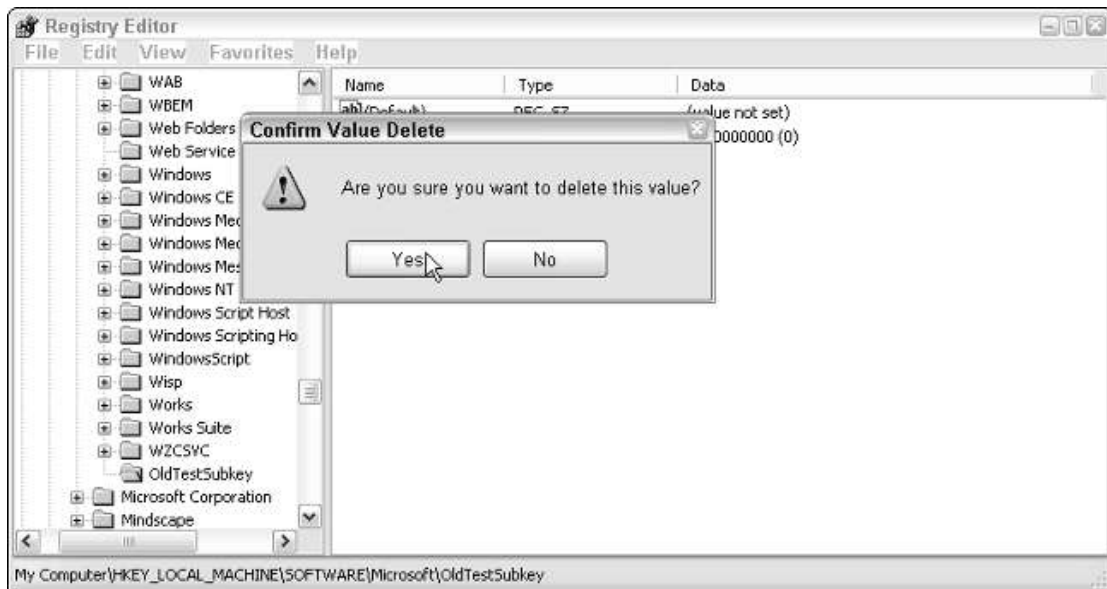
3 - قم بتفريع Microsoft

4 - نضغط على OldTestSubkeysubkey.

5 - اضغط بالمفتاح الأيمن على NewTestDWORD , ثم اضغط على Delete (انظر الشكل) .



6 - إضغط على yes لتأكد أنك تريد ان تحذف القيمة (انظر الشكل)



7 - هذا هو — لقد ذهبت القيمة , يمكنك الان فعل نفس الشئ مع المفتاح الفرعى OldTestSubkey

Manipulating the Windows Registry Using Programming

معالجته برمجياً

كل هذا جيد جداً ويقبل التحرير اليدوي للـ Registry من خلال Regedit . ولكن كميرمج لن يكون لديك الوقت لتمر على جهاز كل مستخدم ونقوم بعمل هذا التحرير يدوياً . أفضل طريقة هي عمل كود ينفذ هذه المهام لك .

ولفعل هذا بطريقة جيدة إستخدم كود VBScript و كود JScript (هي تحسينات مايكروسوفت على JavaScript , وهي مطابقة تقريباً لـ JavaScript) . ولتشغيل هذا الكود . سنستخدم أداة تسمى Windows Script Host التي يمكنك من تشغيل كود JavaScript و VbScript كلغة برمجة من خلال الـ windows .

ما سنفعله هنا هو كتابة بعض الكود التي يضيف Subkeys إلى Windows Registry . إقرأ هذا SubKey , ثم حذفه . للوضوح والممارسة , سنستخدم كلا من Jscript و VbScript لهذه الوظيفة .

VBScript Registry Editing

قم بتشغيل محرر النصوص المفضل لديك وإنشئ مستند فارغ جديد . بمجرد أن يكون لديك هذا , يمكنك ان تبدأ بكتابة الكود الذي تحتاجه .

الشيء الأول الذي تحتاجه لتعلن عن المتغير الذي ستستخدمه . وهو WshShell .

```
Dim WshShell
```

فتحتاج أن تعرف المتغير WshShell كما يلي :

```
Dim WshShell
```

```
Set WshShell = WScript.CreateObject("WScript.Shell")
```

يمكنك هذا من إنشاء نسخة من الكائن WScript.Shell , ويمكنك من تشغيل الكود على نظام التشغيل نفسه . الغرض من WScript.Shell هو سابقة التعريف . وبمجرد استخدام هذا يمكنك من الدخول على Registry من خلال نظام التشغيل windows .

باستخدام تلك المتغيرات فإنت تمد نفسك بإختصار جيد لكتابة الكود كما ستري قريباً . الآن يمكنك ان تنشأ الـ Subkeys .

هناك بعض القواعد لإنشاء مفتاح فرعي . القاعدة الأولى هي ان هذه المفاتيح الفرعية subkeys اعلى في التسلسل الهرمي ويجب ان يتم إنشاؤه قبل الذين يلونه . إذا حاولت مخالفة هذه القاعدة , فهناك خطأ سيحدث .

قاعدة أخرى هامة وهي وجوب التحدث مع نظام التشغيل بطريقة يفهمها . هذا يعني ان عليك ان تفعل الأشياء بطريقة محددة . قاعدة أخرى هي وجوب الإشارة إلى جذور المفاتيح (مفاتيح المسار الأعلى في الـ Registry) باستخدام الإختصارات . الإختصارات لجذور المفاتيح كما يلي :

Root key name	Abbreviation
HKEY_CURRENT_USER	HKCU
HKEY_LOCAL_MACHINE	HKLM
HKEY_CLASSES_ROOT	HKCR
HKEY_USERS	HKEY_USERS
HKEY_CURRENT_CONFIG	HKEY_CURRENT_CONFIG

يجب أن تستخدم هذه الإختصارات في الكود — استخدام شيء آخر أيضاً , مثال الاسم بالكامل , سينتج عنه ان الكود لن يعمل كما نتوقع .

فإذا أردت ان تضيف مفاتيح إلى HKEY_CURRENT_USER/Software , لذا تحتاج ان تختصر هذا كما يلي:

```
HKCU\Software
```

أولاً تحتاج أن تضيف مفتاح فرعى يسمى TestKey تحت Software . سر الكود اذى يفعل هذا هو :

```
Dim WshShell
Set WshShell = WScript.CreateObject("WScript.Shell")

WshShell.RegWrite "HKCU\Software\TestKey\*", 0, "REG_BINARY"
```

هناك أربعة أجزاء فى هذه الجملة المنفردة تحتاج ان تلقى نظرة عليهم .
أولاً , لاحظ إستخدام المتغير Wshshell فى البداية . تم هذا لتقليل كمية الكود الذى تكتبه بدون هذا , سطر الكود سيكون هكذا :

```
WScript.CreateObject("WScript.Shell").RegWrite "HKCU\Software\TestKey\*", 0, "REG_BINARY"
```

أعتقد انك موافق ان هذا غير عملى , طويل . وكثير ليكتب (خاصةً أكثر من مرة) وهذا الكود الذى لدينا أسهل بكثير للتتبع .
الجزء التالى لننظر عليه هو RegWrite . هى دالة تسمح لك بإنشاء مفتاح جديد , إضافة إسم قيمة أخرى لمفتاح موجود من قبل (وإسناد قيمة إليه) , او تغيير قيمة لقيمة موجودة مسبقاً .

دالة Regwrite لديها قواعد إملائية فى بناء الجملة تحتاج ان تتبعها :

```
object.RegWrite(strName, anyValue [,strType])
```

- **Object** : وهو Wshshell وهو مطلوب لإتمام العمل .
- **StrName** : وهى قيمة سلسلة حرفية تشير إلى اسم المفتاح , أو إسم القيمة , او القيمة التى تريد أن تنشأ , إضافة , أو تغيير .
- **AnyValue** : هو إسم المفتاح الجديد الذى تريد أنشائه , أو إسم القيمة التى تريد أن تضيفها إلى المفتاح الموجود من قبل , أو القيمة الجديدة التى تسندها إلى إسم قيمة موجودة من قبل .
- **StrType** . هى قيمة حرفية إختيارية تحدد نوع البيانات . (على الرغم من أننا دوماً سنستخدمه).

هذه القواعد الإملائية فى بناء الجملة تستخدم لتحديد إسم المفتاح بنهاية strName مع علامة الشرط المائل . لا تدرج هذه العلامة إذا كنت تحدد إسم القيمة .

هناك أربعة انواع بيانات ممكنة يمكن ان تحددوها مع strType , وقد أدرجتها فى الجدول التالى

Type	Description	Form
REG_SZ	A string	A string
REG_DWORD	A number	An integer
REG_BINARY	A binary value	An integer
REG_EXPAND_SZ	An expandable string (For example, "%windir%\paint.exe", which would resolve out the folder that Windows is installed into and find the appropriate application)	A string

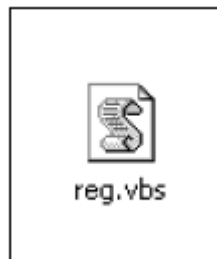
حفاً تحتاج أن تنشأ ثلاثة مفاتيح فرعية , وستفعل هذا بإستخدام الكود التالى :

```
Dim WshShell
Set WshShell = WScript.CreateObject("WScript.Shell")

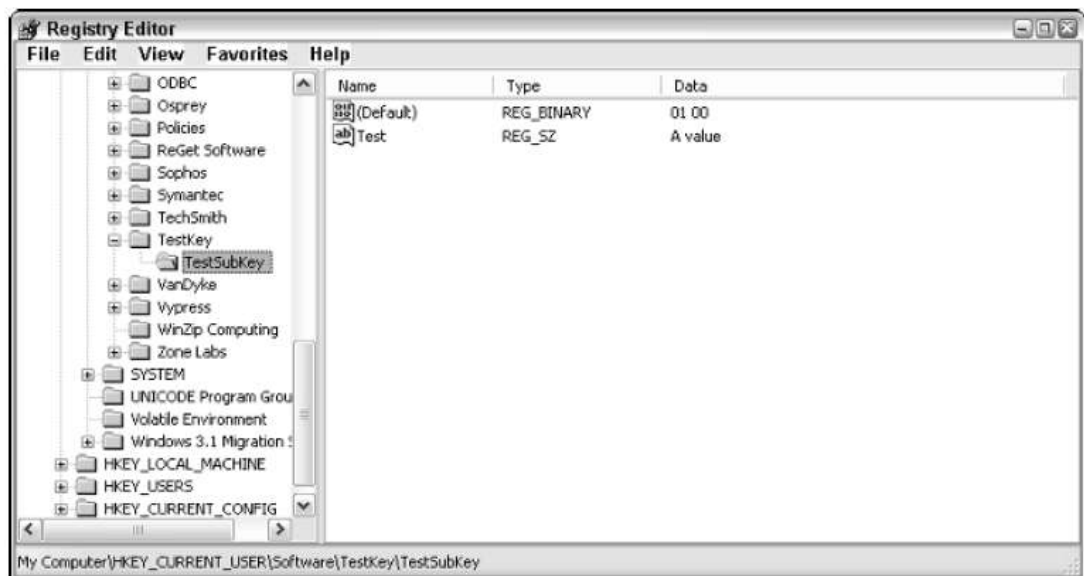
WshShell.RegWrite "HKCU\Software\TestKey\", 0, "REG_BINARY"
```

```
WshShell.RegWrite "HKCU\Software\TestKey\TestSubKey\", 1, "REG_BINARY"
WshShell.RegWrite "HKCU\Software\TestKey\TestSubKey\Test", "A value", "REG_SZ"
```

إلى هذه النقطة يمكنك ان تحفظ هذا الملف .و يمكنك أن تستدعيه وقتما أردت ولكن تحتاج أن يكون الملف بإمتداد .vbs للدلالة على أنه ملف VBScript . في الشكل , لقد قمت بحفظ الملف كـ Reg.vbs . لاحظ الأيقونة لهذا الملف قد تغيرت :



يمكنك أن تقوم بتشغيل الملف , إضغط عليه مرتين . مرئياً , ينبغي ألا ترى شيئاً , ولكن إذا فتحت به Regedit وتفحصته ستري أن المفاتيح والقيم قد تم إنشاؤهم . المفاتيح المنشأ تعرض في الشكل



الآن يمكنك أن تنظر على الكود الضروري لقراءة مفاتيح الـ Registry .
لفعل هذا تستخدم دالة اخرى — دالة RegRead .
في الواقع هذا الكود يبدو كبير ولكن هو سهل جداً للنتبع .

```
Dim WshShell
Set WshShell = WScript.CreateObject("WScript.Shell")

WshShell.RegWrite "HKCU\Software\TestKey\", 0, "REG_BINARY"
WshShell.RegWrite "HKCU\Software\TestKey\TestSubKey\", 1, "REG_BINARY"
WshShell.RegWrite "HKCU\Software\TestKey\TestSubKey\Test", "A value", "REG_SZ"

WScript.Echo WshShell.RegRead("HKCU\Software\TestKey\TestSubKey\Test")
```

هناك جزئين من هذه الجملة , يتسبب التالي في عرض ما قمت بقراءته من الـ Registry على الشاشة .

WScript.Echo

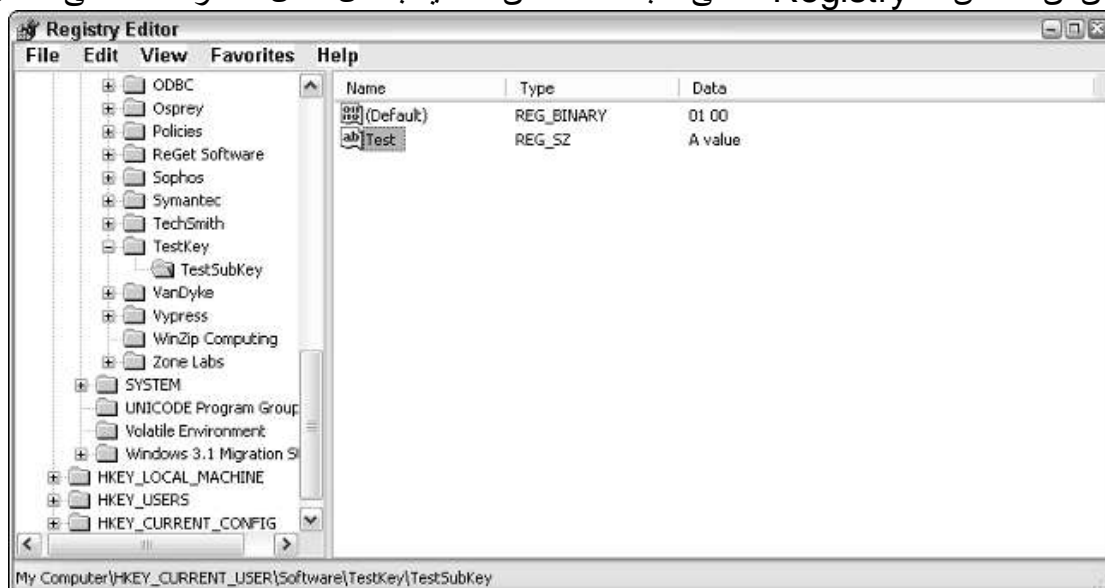
هو الجزء من الجملة الذى يقرأ المفتاح المناسب فى الـ Registry .

```
WshShell.RegRead("HKCU\Software\TestKey\TestSubKey\Test")
```

إحفظ الملف وقم بتشغيله , هذه المرة العملية لن تذهب بدون ملاحظة شئ ما — مربع حوارى قد عرض (شاهد الشكل)



يمكن ان تفحص الـ Registry الفعلى لتثبت لنفسك أن العملية بالفعل تعمل — وقد أتمته فى الشكل



مؤخراً , يمكنك ان تقوم بتنظيف نفسك . وهذا يتم دالة RegDelete . هذا سهل الإستخدام , ويمكنك بسهولة أن تحذف كل المفاتيح الفرعية التى قد أنشأتها .

```
Dim WshShell
Set WshShell = WScript.CreateObject("WScript.Shell")

WshShell.RegWrite "HKCU\Software\TestKey\", 0, "REG_BINARY"
WshShell.RegWrite "HKCU\Software\TestKey\TestSubKey\", 1, "REG_BINARY"
WshShell.RegWrite "HKCU\Software\TestKey\TestSubKey\Test", "A value", "REG_SZ"

WScript.Echo WshShell.RegRead("HKCU\Software\TestKey\TestSubKey\Test")

WshShell.RegDelete "HKCU\Software\TestKey\TestSubKey\Test"
WshShell.RegDelete "HKCU\Software\TestKey\TestSubKey\"
WshShell.RegDelete "HKCU\Software\TestKey\"
```

إحفظ الملف وقم بتشغيله مرة اخرى , سينشأ هذا مفاتيح Registry . إقرأهم ومن ثم نظف وإترك الأشياء كما كانت قبل تشغيل الكود .

إذا قمت بفحص الـ Registry , ستلاحظ أن كل المفاتيح التى قمت بإنشائها قد ذهبت , قوية جداً فى الواقع .

ولكن مع أى قوة تهورية , تحتاج ان تكون حذراً عند حذف مفاتيح Registry بسبب انه من السهل تعطيل جهازك بواسطة حذف الشئ الذى تحتاجه !

JScript Registry Editing

الكود لفعل نفس الشئ فى jscript هو بشكل مدهش مشابه لكود VBScript . هناك بعض الاختلافات , ولكن لاشئ رئيسى .

```
var WshShell = WScript.CreateObject("WScript.Shell");

WshShell.RegWrite ("HKCU\\Software\\TestKey\\", 0, "REG_BINARY");
WshShell.RegWrite ("HKCU\\Software\\TestKey\\TestSubKey\\", 1, "REG_BINARY");
WshShell.RegWrite ("HKCU\\Software\\TestKey\\TestSubKey\\Test", "A value",
"REG_SZ");

WScript.Echo (WshShell.RegRead ("HKCU\\Software\\TestKey\\TestSubKey\\Test"));

WshShell.RegDelete ("HKCU\\Software\\TestKey\\TestSubKey\\Test");
WshShell.RegDelete ("HKCU\\Software\\TestKey\\TestSubKey\\");
WshShell.RegDelete ("HKCU\\Software\\TestKey\\");
```

بعض الاختلافات التى نلاحظها هى :

- أولاً : كل جملة تنتهى بعلامة Semicolon ; . هذا قياسى لكلا من JavaScript و Jscript ولكن إختيارياً . يفعل .مع ذلك , تجعل الكود أسهل للتبع , الكود التالى سيعمل جيداً كذلك :

```
var WshShell = WScript.CreateObject("WScript.Shell")

WshShell.RegWrite ("HKCU\\Software\\TestKey\\", 0, "REG_BINARY")
WshShell.RegWrite ("HKCU\\Software\\TestKey\\TestSubKey\\", 1, "REG_BINARY")
WshShell.RegWrite ("HKCU\\Software\\TestKey\\TestSubKey\\Test", "A value",
"REG_SZ")

WScript.Echo (WshShell.RegRead ("HKCU\\Software\\TestKey\\TestSubKey\\Test"))

WshShell.RegDelete ("HKCU\\Software\\TestKey\\TestSubKey\\Test")
WshShell.RegDelete ("HKCU\\Software\\TestKey\\TestSubKey\\")
WshShell.RegDelete ("HKCU\\Software\\TestKey\\")
```

- ثانياً , إعلان المتغيرات مختلف — ولكن أن تعرفهم بالفعل !.
- ثالثاً , تتطلب القيم الحرفية لعلامتين إثنين // — هذا بسبب أن علامة واحدة تمثل حرف هروب التى تختلف إعتماًداً على ما يتبعها . حاول أن تكتب علامة واحدة وسيحدث هناك خطأ .
- أخيراً , لاحظ إستخدام الأقواس . مرة أخرى , أشياء رئيسية لـ JavaScript/Jscript .

As an aside, here is a list of JavaScript and JScript escape characters:

\b	Backspace
\f	Formfeed
\n	New line
\r	Carriage return
\t	Horizontal tab
\'	Single quote
\"	Double quote
\uHHHH	Unicode encoded character (where HHHH is character code)
\###	Latin encoded character (where ### is character code)

إختلاف إخر فى إمتداد الملف — بدلاً من .vbs , تستخدم .js . الأيقونة هى نفس الشئ (كما تراها فى الشكل)



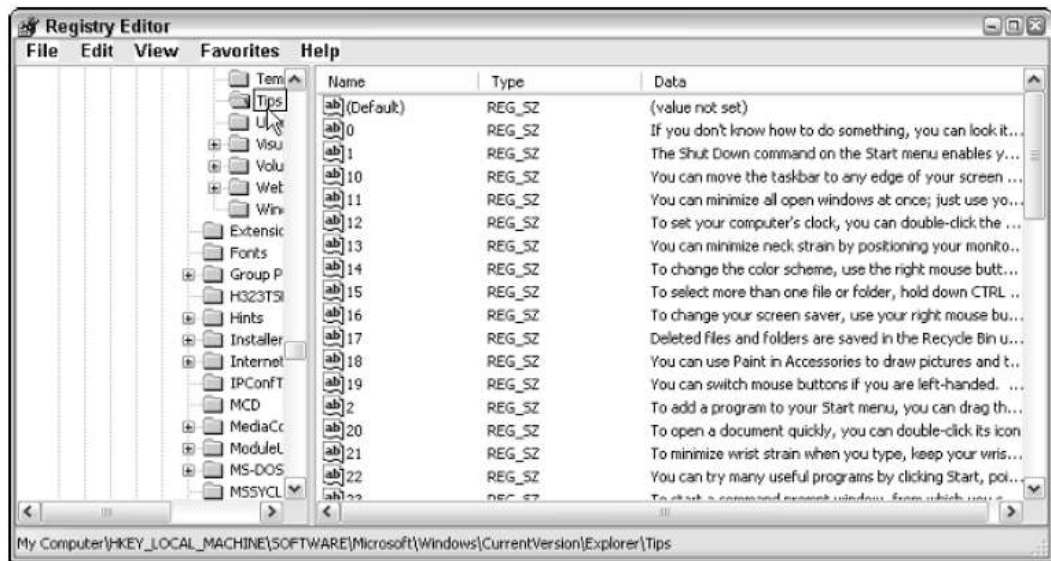
Possible Uses for the Windows Registry

الاستخدامات الممكنة للـ Windows Registry .

قبل ان نتقرب دعنا نلقى نظرة على نوع المعلومات التي يمكنك أن تقوم بتخزينها فيه .

ينبغي ان يكون جلياً لك أن تعرف ان المعلومات المطولة ليست شيئاً مناسباً للـ Registry . لسنا نقول ان الـ Registry لا يمكن ان يمسك معلومات مطولة , إلقى نظرة على

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\explorer\tips\
وسترى بعض التلميحات هنا مطولة جداً في الواقع (ترى بالشكل)



قبل كل شيء هي إعدادات , وليست ملفات , وأحد الأشياء التي لا ينبغي أن تستخدم الـ Registry هي تخزين ملفات . وهذا من أجل نظام الملفات .

إليك هنا بعض انواع المعلومات المفيدة لـ Registry . هذه ليست قائمة نهائية ولكن هي فقط لتعطيك فكرة :

- ❑ Storing user licensing information
- ❑ Storing menu/option defaults
- ❑ Storing specific file paths relating to your application
- ❑ Storing version information
- ❑ Storing information on when the application was last run

In Closing

فى الخاتمة

- وهناك استخدامات أخرى لا تعد ولا تحصى عن الـ Registry التى ستكون محددة بالتطبيق — فهو يعتمد على ماذا تفعل . مع ذلك , دعنا نتركك مع بعض المعتقدات عن الـ Registry وإستخدامه.
- إستخدام Registry هو خطر محتمل ويمكن أن يسبب ضرر شديد للنظام — انت خلال مرحلة البرمجة و الآخرين خلال مرحلة الإختبار . فتأكد ان قد أخذت كل الإحتياطات ضد تدمير وإختبره مراراً قبل أن تدع الآخرين ليستخدموا برامجك .
 - لا تستخدم Registry فى أشياء تافهة .
 - الكثير من أدوات البرمجة , خاصةً المجانية , لا تسمح لك بالدخول الميسر للـ Registry . عامةً , أداة البرمجة الأكثر ثمناً , تعطيك دخلاً أسهل إلى Windows Registry (ماعدا للـ Windows Script Host , فهى مجانية وتسمح لك بحق جيد فى الدخول للـ Registry)
 - أحد الاستخدامات العظيمة للـ Registry , خاصةً مع رفقة Windows Script Host , هو أن تنشأ أدوات بسيطة لتقوم بالتعديلات السريعة للنظام (إذا كنت تعرف ماذا تفعل) . على سبيل المثال , يمكن أن تستخدم ما قد أتممناه اليوم بسهولة قم بتعديل الكود الذى قمت بالعمل معه لتغير تلميحات الـ Windows المحتواه فى Registry , أو حتى حذفهم تماماً .
 - أخيراً , لا تقلق كثيراً لإستخدام Registry — إذا كنت قلق عن التسبب فى مشاكل على النظام الخاص بك أو النظام الخاص بالآخرين , إتركه وحده !

Summary

الملخص

فى هذا الفصل , أخذنا جولة خلال الـ Windows Registry وبحثنا فى كيفية تخزين بيانات متعلقة بنظام التشغيل windows . معرفة تم تخزين البيانات هى مفتاح للدخول والوصول إليها .

وبحثنا أيضاً فى كيفية أخذ خطوات لحماية Registry من الضرر بواسطة عمل نسخة إحتياطية — فقط فى حالة أن الأمور جرت على نحو خاطئ .

مؤخراً , إنتقلنا إلى البحث فى بعض البرمجة ونظرنا فى طريق قوية وبسيطة للوصول للبيانات المخزنة فى Registry وأيضاً طرق للإضافة , والتعديل , وحذف البيانات منها .

14

Organizing, Planning, and Version Control

التنظيم , التخطيط , و التحكم فى الإصدار

هذه هى حقيقة الحياة هذه الأيام كمستخدم كمبيوتر نحن نقوم بتوليد كميات من الملفات فى تعاملاتنا يوماً بعد يوم مع الحاسبات , هى أيضاً حقيقة الحياة كمبرمج (أو مبرمج نشط على الأقل) . كلما كتبت كود كلما إزداد الكود فى نهاية المطاف .

إذا لم تكن حذراً فى أخذ الخطوات الصحيحة فى وقت مبكر , ستجد نفسك تبحث فى ما يمكن ان يكون مئات أو الآلاف من الملفات لمحاولة للبحث عن ملف واحد خاص فيما بعد . كما يمكن أن تتخيل , يمكن ان تكون هذه مهمة تتطلب الوقت والجهد وتقديم كلاهما كل من الملل والإحباط . فلا حاجة فيها وغير ضرورى وربما ينتج عنها فقدان كود جيد .

هناك مشكلة أخرى متعلقة بها , أحد ما يحصل فى كثير من الاحيان : حفظ بإستبدال الكود القديم بالجديد , كود غير المختبر , ويسبب هذا مشاكل أسفل السطر عندما تريد العودة إلى الكود القديم .

الدخول إلى أى نوع من مصدر الكود يفسد عادةً مشاكل التهجى أسفل السطر , لذا فى هذا الفصل سأريك كيف تتحكم فى الكود وتبقيه تحت السيطرة , فيمكنك ان تجد الملفات التى تريد متى تريد .

Organize, Organize, Organize!

تخطيط , تخطيط , تخطيط

أول عنصر فى جدول الأعمال عن التحكم فى الإصدار هو تجد نفسك منظم . وهذا يعنى تنظيم نفسك , ومساحة العمل الخاص بك , وحاسبك الشخصى للوظيفة .

Organize Yourself

نظم نفسك

أفضل موضع للبدء أن تنظم نفسك , لا تقلق على الرغم من — أنه ليس هناك تغيرات كبيرة فى الحياة سيكون عليك فعله !

العنصر الأول لتنظيم نفسك هو الحصول على خطة لكل المشروع والاتجاه الذي تعتقد انه سيأخذه . راجع الفصل الحادى عشر , "Putting it All Together" , لتفاصيل أكثر عن كيف تخطط المشروع البرمجة . مر خلال هذا بحرص وخطط للمشروع أفضل ما يمكن . فكلما كان التخطيط أوضح تحسنت الأمور لك .

Stages of Planning

مراحل التخطيط

- التخطيط هو عملية متعددة الخطوات . الخطوات الرئيسية هي :
- الفكرة : وثق الفكرة التي لديك بشكل وافى . هذه هي البذور مما سيجعل مشروعك ينمو , لذا تأكد من انك قد رثقتها بوضوح . من السهل أن تتغير الفكرة ويمكن ان تفقد ما طورته .
 - المتطلبات : وهى عمل قائمة بما تتوقع أن النظام سيكون فى حاجة إليه . قائمة المزايا كثيرا ما تنكمش وتنمو كلما تمر خلال مرحلة التخطيط ومقارنةً بأحلامك لمهاراتك البرمجية الفعلية ولكن وهذا طبيعى ولا شئ هناك لتقلق منه . تذكر , يمكنك دوماً ان تضيف ميزة لبرنامج قمت بالفعل بكتابته , فى حين انك كنت قد اتعبت نفسك بمشروع كبير جداً فى البداية ربما لن تكمله .

Plan Your Time

خطط لوقتك

وثمة قضية أخرى حاسمة , لقد تم تحذيرنا جميعاً عن أخذ أكثر من اللازم . من إمتلاك عيون أكبر من معدتنا أو قضم أكثر مما يمكن مضغه . وهذا يمكن ان يؤثر سلباً على ناتج مشروعك .

مفتاح النجاح هو أنك تكون صادق فى مهاراتك والوقت المتاح لديك . إنطلق بالعمل على مشاريع صغيرة وإستمر فى البناء تدريجياً . بهذه الطريقة , يمكن ان تبني عضلاتك البرمجية هكذا يمكنك أن تأخذ مشاريع أكبر , بينما أيضاً تحسن مهاراتك التخطيطية .

Organize Your Workspace

نظم مساحة العمل الخاصة بك

الشئ التالى الذى تحتاج أن تفعله هو تنظيم مساحة العمل . مختلف الأشخاص يمكن ان يقومون بالعمل فى انواع مساحات مختلفة , وبينما بعض الأشخاص يعتقدون أن مساحة العمل المشوشة تشجع على عدم التنظيم , وعدم إنتظام التفكير , الآخرون لديهم مشاكل قليلة فى العمل مثل هذه الظروف والعثور على نظافة معقمة جداً ومصطنعة.

انت تعرف عن نفسك أكثر منى أى شخص تكون , ولن أسالك لتتغير أو اخبرك ان تكون شخص مرتب لتصبح مبرمج . ما سأقوله لك انك تحتاج مساحة عمل يمكنك من العمل لفترات ممتدة وتعمل هذا فى راحة .

اقل ما أقوله انك تحتاج :

- **مساحة عمل مريحة** . يشمل هذا مقعد مريح وعلى إرتفاع صحيح (هكذا تقريباً أن تكون رجليك مسطحة على الأرض عندما تجلس عليه) , ولوحة مفاتيح فى أرتفاع مناسب (تقريباً هكذا أن يكون ساعدك موازية للأرض) . الغرفة يجب ان تكون دافئة , ولكن ليست دافئة جداً .
- **مكان للكتب والملاحظات** . من المحتمل أن يكون لديك كتب , ورق , ومفكرات محيطة بك التى سترجع إليها . أقترح انك إن لم تكن الشخص الذى لا يحب أن تكون المنضده حوله مرتبة ان يكون لديك مساحة كافية لهذه الأساسيات . أيضاً حافظ ان يكون هناك قلم رصاص أو جاف ومفكرة قريباً منك لتدون ملاحظاتك .

- **حد ادنى من الإلهاء** . حاول ان تعمل فى مكان حيث يكون لديك حد ادنى من الإلهاء من الآخرين . إستخدم الموسيقى إذا وجدت ذلك يساعدك فى إنشاء مجال من المساحة الشخصية حولك . إن أمكن , حاول ان تكون بعيداً عن الهاتف (إغلقهم أو أخفض صوته) , وأغلق تطبيق البريد الإلكتروني وغيرها من المصادر الملهية .

أشياء أخرى تمكن ان تساعدك ولكن ليس أساسية لمساحة العمل الخاصة بك :

- **شاشة كبيرة** . كلما زاد حجم الشاشة , كلما ان تجد الراحة مع الشاشة , عندما تقوم بالبرمجة ربما تجد أنك سواء كنت فى تطبيق واحد لفترة طويلة (مثال محرر النصوص , المفسر , والتطبيق الذى تم تفسيره) . كلما كبر حجم الشاشة كلما كان من الممكن إحتوائها للمزيد . بينما تافظ على حجم النص مناسب ومريح .

الحد الأدنى لوضوح الشاشة الذى أقترحه للبرمجة هو 1024 x 768 . أى شئ تحت هذا اجده غير عملى . فى الأشكال التالية , قد تضمنت تخطيطين إثنين تعرض الأحجام المتعلقة لمختلف عناصر وضوح الشاشة 800 x 600, 1024 x 768, and 1200 x 1024 .





- **تخصيص لوحات مفاتيح/ فأرات** . هناك عدد من لوحات المفاتيح والفأرات فى السوق لديها تنوع من الأزرار وبكرات التمرير فيهم قد صممت لذا يمكنك تخصيص وظائفها إلى فعل معين (مثال قص , لصق , التنقل بين التطبيقات أو أياً كان) . لبعض الأجهزة الطرفية الجيدة لديها مزايا فى متناول أيدي المبرمجين , إلقى نظرة على www.microsoft.com/hardware و www.logitech.com .

يمكن أن يكونوا حقاً مفيدين للبرمجة لأنك يمكنك ان تبرمج مهام عليهم التى على جانب الآخر تأخذ الفأرة حركات كبيرة او ضغطات على المفاتيح متعددة .

العقبة الأساسية لهذه الأنظمة أنه من الطبيعى تأخذ لحظات لتستخدمهم . لدى ازرار فى الفأرتى الخاصة بى ولوحة مفاتيح لوظائف مثال قص , نسخ , لصق , تنقلات التطبيقات , التمرير أسفل الصفحة , واكثر بكثير ولكن ظلت ناسيهم انهم هناك ورجعت إلى الأقل كفاءة ولكنها الطريقة الأكثر رسوخا للقيام بهذه الأمور . العقبة الأخرى لهذه هى انك بالفعل إن كنت تستخدمهم ومن ثم تنتقل بين الأنظمة , حقيقياً ستفتقدهم وسوف تنخفض الإنتاجية بشكل كبير.

The Main Event — Organize Your PC

الحدث الرئيسى — نظم حاسبك .

لقد أصبحت حالك منظمة . ولديك مساحة العمل كما أردت أن تكون (او على الأقل البعض قريباً مما أردت أن تكون — معظم الأشياء تنطوي على حل وسط) . الآن هو الوقت للحدث الرئيسى وهو البحث فى طرق تنظيم حاسبك الشخصى .

Create a Workspace

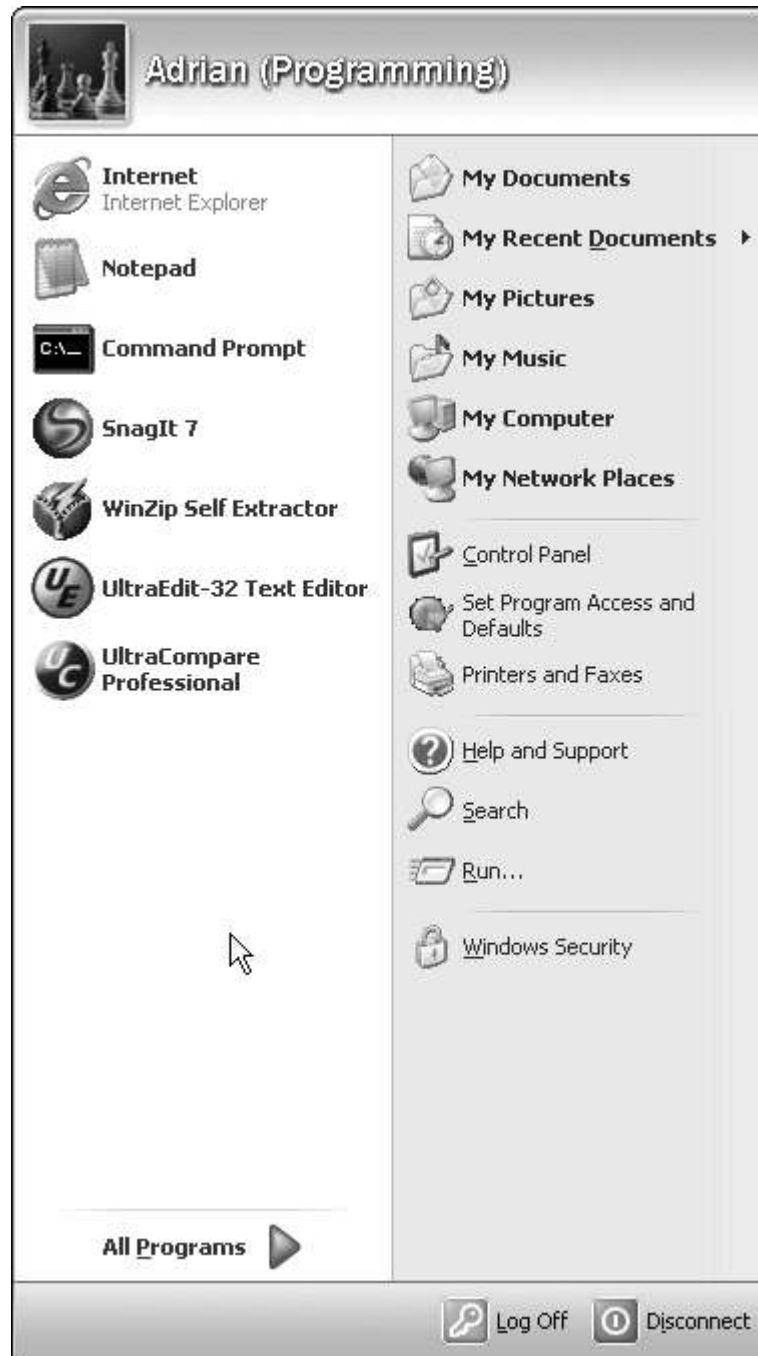
إنشئ مساحة عمل

بمجرد أن تنشئ مساحة عمل حولك , تنشئ مساحة عمل رقمية لك أيضاً . مكان واحد جيد لتبدأ بإنشاء لنفسك ملف برمجى على حاسبك الشخصى . Microsoft Windows تعطيك القدرة على إنشاء ملفات تعريفية على حاسبك الشخصى يكون لديها إعدادات تكوينية مختلفة . وإمتلاك ملف تعريفى معين للبرمجة ربما يكون فكرة جيدة . فيمكنك ان تضع فيها البرامج والتطبيقات الخاصة بالبرمجة فى قائمة Start وتقوم بحذف التطبيقات الغير ضرورية والآلهائية (العاب , تطبيق البريد الإلكتروني , وهكذا)

الشكل يعرض تخصيص قائمة Start التى قمت بإنشائها .

القيام بفعل هذا يعطيك مميزات أخرى . أولاً , حذف التطبيقات الغير ضرورية يعطيك ذاكرة أكبر لتكريسها للتطبيقات المطلوبة . هذا يعنى ان حاسبك الشخصى سيعمل بأكثر سلاسة وسرعة — وهكذا . تطبيقات أقل للعمل يعنى أيضاً عدد أقل من الصراعات والحوادث — الشئ الذى ستريد ان تتجنبه عندما تيرمج هو كالحوادث مما يعنى ضياع العمل .

لا يهم كم يكون النظام مستقر أو آمن , إجعل عندك عادة بحفظ العمل بشكل متكرر . إذا كان محرر النصوص او بيئة التطوير التى تستخدمهم لديهم دعم ميزة الحفظ التلقائى لحفظ نسخة من عملك بشكل دورى , إستفد من هذه الميزة . ليس هناك أكثر تثبيطاً وإزعاج من ضياع كل عمل بالظهر !.



ميزة اخرى هى إمتلاك مساحة عمل جاهزة لمشروعك . يمكن استخدام Desktop لحفظ ملفاتك .



لا تحفظ الملفات فقط على windows Desktop لأن هذا سينتج عنه فوضى وقريباً سيصبح من المستحيل أن تتصفحه . فحتاج أن تقضى بعض الوقت لتنظم قليلاً .

Folders, Folders, Folders

مجلدات , مجلدات , مجلدات

أحد مفاتيح مكونات حفظ عملك منظم هو المجلدات . هناك طرق كثيرة يمكن لك أن تنهجها ولكن بعض معظم المخططات الناجحة التي وجدتها توصف في المقاطع التالية .

Group by Language

التجمع بواسطة اللغة

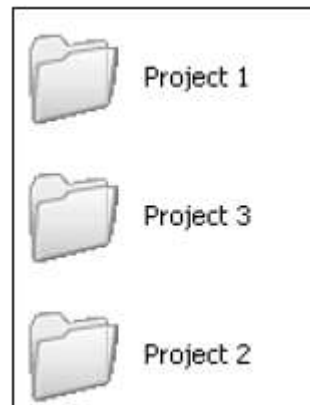
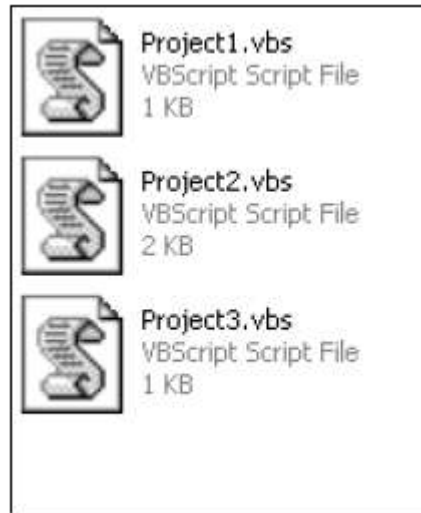
أحد أبسط الطرق لتجميع ملفاتك , واحد الذين يعملون في البداية بطريقة جيدة جداً , هي تجميع ملفات بواسطة لغات البرمجة المستخدمة .

هناك مخطط بسيط ويعمل كالتالي :

1 - تنشئ مجلد للغة برمجة معينة, لنقل , في هذا المثال , VBScript . إعطى هذا المجلد اسم واضح ولكن دقيق يلخص اللغة التي تستخدمها . (انظر الشكل)



2 - داخل هذا المجلد لديك أحد إختيارين . يمكنك تجميع ملفاتك في مجلد واحد (انظر الشكل) او تنظيم في مجلدات فرعية للتوضيح (انظر الشكل) . من المحتمل أن ستجد أنك إذا قمت بتجميع كل الملفات في مجلد واحد .فقریباً يصبحوا من الصعب إدارتهم إذا لم تكن واضحاً جداً في كتابة أسماء الملفات . مثلي في الشكل . حيث أنني قد أعطيت كل الملفات أسم دقيق وواضح .



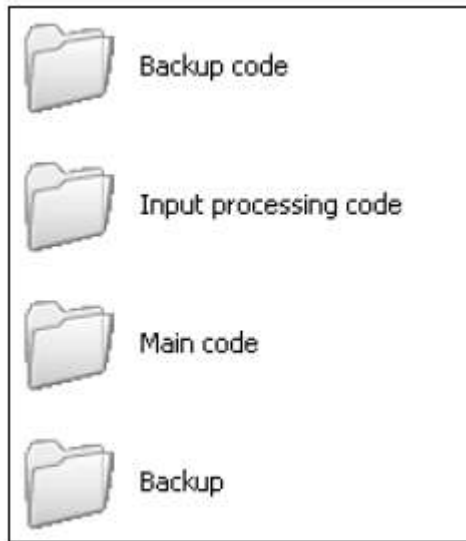
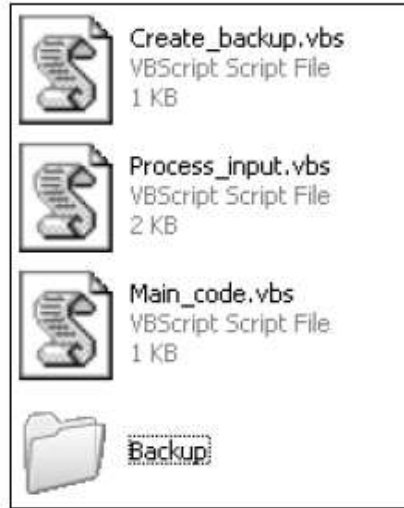
3 - عندما تنتهي مشروع او مجرد أن تريد ان تنشئ نسخة احتياطية , يمكن نسخ أو نقل الملفات ذات الصلة إلى أرشيف مضغوط مثال أرشيف zip . (انظر الشكل) . يمكن أيضاً ان تحفظ ملفات الأرشيف هذه في جهازك الخاص أو تقم بنقلهم إلى وحدة تخزين أخرى مثال CD أو DVD .



Group by Project

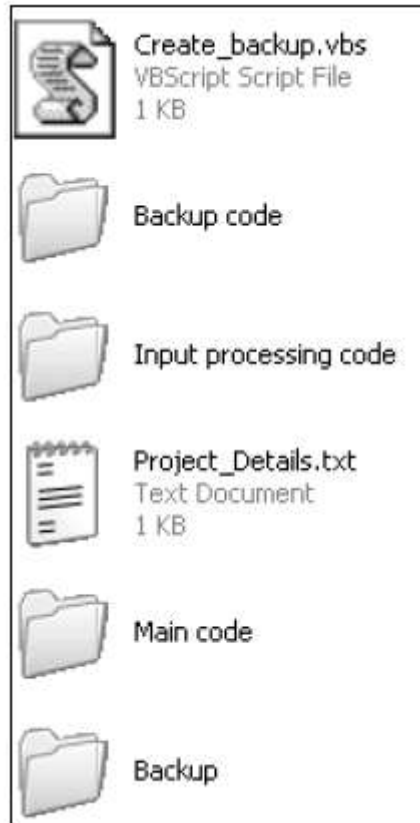
تجميع بواسطة المشروع

تجميع الملفات من خلال اللغة ربما تكون طريقة جيدة للمبتدئين في العمل . ولكن قريباً جداً ستجد ان لديك تمييز للغة واحدة عن اخرى ومنذ أن يحدث هذا فهو الوقت لتبدأ بتنظيم الملفات بناءً على المشروعات . على العموم لقد بدأت مع مجلد واحد للمشروع . ولكن غالباً ما يصبح مقيد أو اكثر كبراً ليتم إدارته . ومن ثم أنوى ان احذف وأرشفة ملفات النسخ الاحتياطية القديمة إلى مجلد آخر . (انظر الشكل) , فهي فكرة جيدة على كل حال . او تقسم المشروع إلى مشروعات فرعية مجلدات فرعية . (انظر الشكل) , مما يعنى أن لدى بعض الملفات والمجلدات .



Folder Contents Note

نصيحة جيدة اخرى يمكن أن أمدك بها هي صنع ملف نصي في المجلد الى تعمل عليه , وفي هذا الملف ادرج ما يفعله كل ملف ويتبع أى مشروع . انا أنوى أن أسمى هذا الملف شيئاً ما كـ **Project_Details.txt** (إنظر الشكل) وأحفظه كنص عادى حتى يسهل فتحه وتحريره بسرعة وسهولة وبأى محرر نصوص .



ملفات معالجة النصوص ربما تمكنك من تنسيق خيالى للمستند , ولكن هى تتطلب برامج معينة مثبتة على الحاسب لتعمل . والإختيار النهائى هو إما أن تستخدم نص عادى أو تنسيق معالج نصوص معين كما يحلو لك , ولكن اجد أن السرعة والبساطة مع النص العادى دوماً تفوز معى !

فى العام , التخطيط للمستند الذى استخدمه يميل إلى يتبع هذا التنسيق :

```
Project Title
Short project description
Date started
Date last edited
Status
-----

Folder 1
-----

Filename1 - What the file does
Filename2 - What the file does

-----

Folder 2
-----

FilenameA - What the file does
FilenameB - What the file does

-----

Special notes
-----

EOF
```

- دعنا نلقي نظرة على المعلومات التي ستكون في هذا الملف :
- عنوان المشروع : عنوان المشروع الذي تعمل عليه .
 - وصف مختصر للمشروع : إعطى وصف مختصر للمشروع هنا , إلقيه مختصراً , إذا كان لديك أى ملفات أخرى انشأت تعود للمشروع , ادرجهم هنا ليسهل إيجادهم عند الحاجة . هذا مكان جيد لتدرج لغات البرمجة المستخدمة وانظمة التشغيل (إن كانت معروفة)
 - تاريخ البدء : التاريخ الذي بدأت به العمل مع المشروع .
 - تاريخ آخر تحرير : تاريخ آخر تحرير قمت بعمله في مجلد المحتويات .
 - الحالة : حالة المشروع ., على سبيل المثال , لم يبدأ , فى تقدم , تحت الاختبار , وتم إنهائه .
 - مجلد 1 ومجلد 2 وهكذا . إدراج أسماء المجلدات , أسفل ما تدرج أسماء الملفات (إسم الملف 1 , إسم الملف 2 , إسم الملف 3 وهكذا) جنباً إلى جنب وصف لكل من الملفات وما تفعله بالفعل .
 - ملاحظات خاصة : أى ملاحظات أخرى لها صلة بالمشروع .
 - نهاية الملف . يدعك تعرف أثناء قرأتك وتحريرك للملف متى ينتهى . هذه أضافة بسيطة تجعل القراءة اسهل .

إليك نموذج ملف ملاحظات .

```
"Hello, World!" application
A simple "Hello, World!" application written in C++. This should work on all the
latest version of Microsoft Windows.
Date started - 11/04/2004
Date last edited - 11/05/2004
Status - Undergoing testing
-----
Working_Files
-----

folder_info.txt - This file
hello.cpp - Main C++ source code file
hello.cpp.bak - Current backup C++ source code file
hello.exe - Compiled code

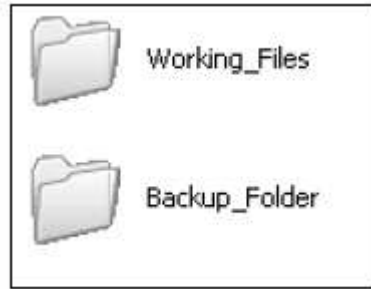
-----
Backup_Folder
-----

hello_110504.cpp - Backup of C++ source code file
hello_110404.exe - Executable file. Does not display message
hello_110404_old.cpp - Backup of C++ source code file

-----
Program tested on Windows XP and 2000. Undergoing testing on Windows Me.
-----
EOF
```

يمدك هذا بالمعلومات الأساسية عن الملفات التي تحتويها المجلدات . لاحظ أيضاً اننى قد إتضمنت إسم ملف معلومات المجلد فى القائمة . هذا جيد للتمام ولذا تتم القائمة وتعرض كل الملفات .

الشكل التالى يعرض تركيب المجلدات والملفات ممثلة بملف معلومات المجلد



Filename Control

إلى جانب تسمية المجلدات فعاله , من المهم أن تستخدم أسماء ملفات فعاله . أن انوى ان اجد أن إتفاقية تسمية الملفات يساعدك لحفظ الأشياء مرتبة .

- **filename.ext** : أستخدم هذه لأسماء الملفات القياسية , إمتداد الملف يحدد باللغة المستخدمة. حافظ على إسم الملف أن يكون مختصر . ويحتاج أن يكون ذا وصف جيد مدمت تحفظ ملف معلومات المجلد

- **filename_mmdyy.ext** : نسخة إحتياطية من الملف (لنقول من اليوم السابق أو قبل أى تغييرات رئيسية . mmdyy تمثل تاريخ الملف . إذا قمت بعمل أكثر من واحدة فى اليوم إذاً يمكنك ان تضيف حرف أو رقم فى النهاية لتعطى مجال اكبر , على سبيل المثال ,
 filename_mmdyyA.ext or filename_mmdyy01.ext.

- **filename_mmdyy.ext.bak** : لاحظ مضاعفة الإمتداد هنا , هذه هى نسخة تقدم العمل الحالى . هذا جيد طالما انك تعمل و هى النسخة السابقة المحفوظة من الملف . التطبيقات مثل UltraEdit يمكن ان تفعل هذا لك تلقائياً . إذا قمت بعمل أكثر من واحدة فى اليوم , إذاً يمكنك ان تضيف حرف أو رقم فى النهاية لتعطى مجال اكبر , على سبيل المثال ,

filename_mmdyyA.ext.bak أو filename_mmdyy01.ext.bak.

- **filename_mmdyy_old.ext** : الإصدار القديم من النسخة الإحتياطية , وتظل موجودة فى حالة انها فى قيد الإفادة . وهم فقط جديرون بالحفظ إذا كانوا يعملون على امثلة من الكود . وإذا لم تكن , فمن الأفضل حذفهم . إذا قمت بعمل أكثر من واحدة فى اليوم , إذاً يمكنك ان تضيف حرف أو رقم فى النهاية لتعطى مجال اكبر , على سبيل المثال ,

filename_mmdyyA_old.ext or filename_mmdyy01_old.ext.

كن منضبطاً وممنهجاً فى تسمية الملفات . ربما تاخذ منك بعض الوقت اتجدها معك فى العمل . ولكن على المدى الطويل ستحفظ لك الوقت وتقلل المشاكل والأخطاء . أحد الأشياء التى يعتمد عليها النظام ألا تستخدم علامة (_) فى تسمية الملفات عامةً . هذه العلامة نادراً ما يستخدمها معظم الأشخاص فى انها ليست مشكلة . مع ذلك , إذا كان لديك ملف مسى به هذه العلامة , لا تتردد فى تبديلها برمز آخر .

بعض الأمثلة التى تشمل ذلك :

```
filename-mmdyy.ext
filename$mmdyy.ext
filename+mmdyy.ext
filename#mmdyy.ext
```

كل هذه الأمثلة ساعمل جيداً فقط إذا كنت بالفعل تسخدم (_) فى أسماء الملف فى ملفات النظام . الشكل التالى يعرض مختلف تسمية الملفات فى نطاق العمل .



إذا كان لابد , أو كنت تشعر أنها تضيف الوضوح لملفاتك , يمكنك أن تخلط بينهم في نفس المجلد . يمكنك أيضاً أن تعطي معاني مختلفة لرموز مختلفة . كما فعلت في الشكل التالي . طالما أنك تذكر أن توثق هذا في ملف معلومات المجلد , سيكون هذا جيد ولن تنسى المعنى أو تقع في مشكلة .



More Version Control Tips

نصائح عن التحكم في الإصدار

دعنا نلقي نظرة على بعض النصائح والحيل الأخرى لتساعدك في تعقب مصدر الكود لتجعله من السهل إيجاداه .وتساعدك على حراستك من الضياع .

Add Version Information to the Tombstone Comment Block

إضافة معلومات عن الإصدار إلى علامات كتل التعليق

هي حيلة بسيطة معظم القادمين الجدد إلى البرمجة ينسون فعل هذا . تذكر أنه يمكنك أن تضيف معلومات الإصدار إلى علامات التعليقات من الكود . هذا يعني انه لن يكون لديك فقط ملف بالإصدارات وتغييرات الإصدار ولكن أيضاً سيكون لديك قائمة بهذة التغييرات في مصدر الكود الفعلى .

بساطة وتفصيل تعليقات الإصدار التي تضيفها تعتمد على مدى شعورك . إليك مثال لتعليقات مبسطة

```
// Tombstone comments
// Widget 1.0.2
// Author: A. N. Other
// 22-10-04
//
// Code starts below.
```

إليك هنا المزيد من التعليقات المفصلة في كتلة تعليقات .

```
// Tombstone comments
// Widget 1.0.2
// Author: A. N. Other
// Simple application that displays random messages on-screen
// Started: 22-10-04
// Last revision: 04-11-04
// Last revision by: A. N. Other
// Project status: In progress
//
// Code starts below.
```

هناك كلا المميزات والعيوب في استخدام التعليقات بهذه الطريقة , الميزة الرئيسية أنك تحفظ المعلومات بشكل صحيح مكان ما تحتاجها — بالملف الذي يحتوى على مصدر الكود . بتلك الطريقة هناك فرصة لئلا في أن المعلومات سيتم فصلها من الكود . يمكن ان يكون ميزة هائلة إذا قمت بغير قصد بحفظ الملف في مكان ما غير متوقع (أو تم نقل الملف بدون قصد) .

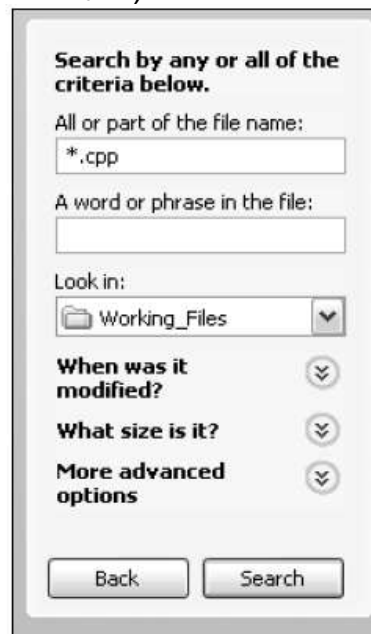
العيوب الذي يكون لديك أن تصلح المعلومات خلال الملف الفعلي في مقابل ملف منفصل . إذا قمت بإصلاح ملف معلومات المجلد , إذاً يمكن أن يبدو هذا كعمل شاق ولكن في الواقع هي حالة من قص ولصق المعلومات من ملف إلى آخر .

وهناك أثر جانبي آخر مفيد في حفظ معلومات الإصدار من خلال مصدر الكود الفعلي , وتكون إذا تم نقل أو حفظ الملف بغير قصد إلى مكان خاطئ فيمكنك استخدام أداة البحث ليساعدك للبحث عنه .

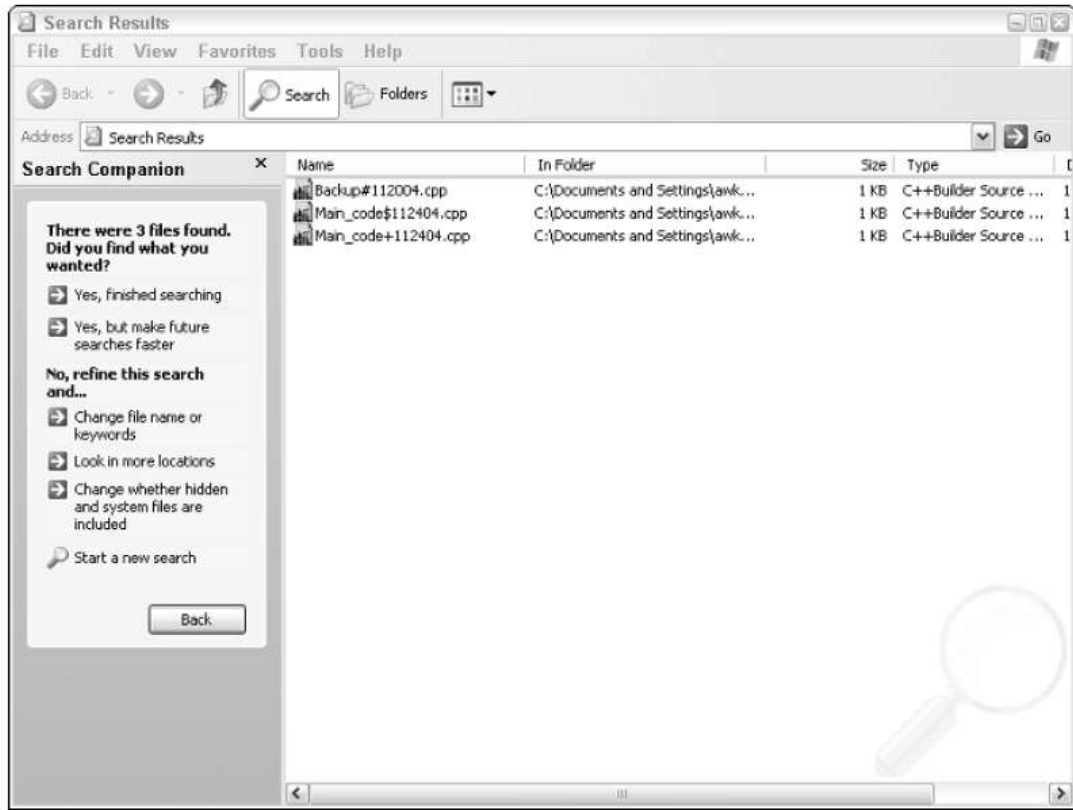
Using Windows Search

إستخدام البحث

دعنا نقول أنك تبحث عن ملف يحتوى على مصدر كود بـ C++ وقد وضعته على القرص الصلب . وانت تعرف أن الملف له إمتداد .cpp , لذا يمكنك ان تبحث عنه (انظر الشكل)



سيقوم هذا بسحب كل الملفات التي بإمتداد .cpp (انظر الشكل)



هذا مفيد ولكن إذا كان لديك المزيد من الملفات في النظام .إذاً حتى كل التضيق للملفات بالإمتداد المناسب ربما لا يساعد . مع ذلك , إذا قمت باضافة تعليقات إلى مصدر الكود الفعلى , يمكنك أن بحث عنها كجزء من البحث لما يحتويه الملف . (انظر الشكل)

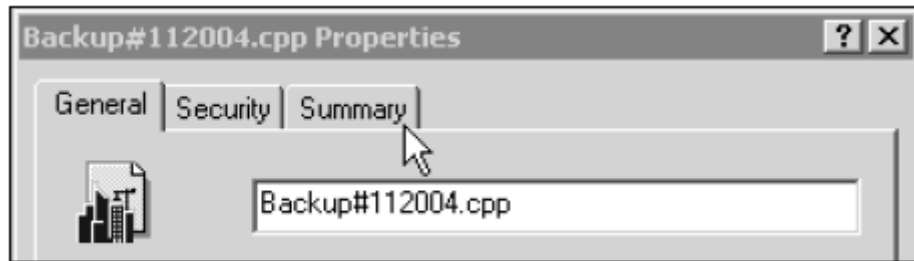
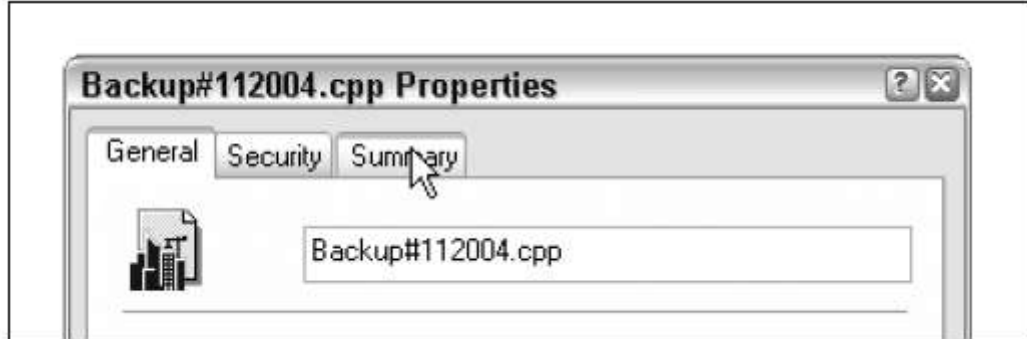


يمكن أن يكون هذا مساعدة كبيرة ويمكنك من تضيق معيار البحث . يساعدك لتكشف الملف الفعلى من بعد . لايد أن تكون مبدع مع متطلبات البحث , ولكن إذا مازال يمكنك أن تجد ملف معلومات المجلد ذات الصلة إذاً يمكنك أن تستخدم المعلومات التي يحتويها لتضيق معيار البحث لك وتقلل الملفات الموجودة خلال البحث إلى شيء ما أكثر تحكماً فيه . (كما نأمل , إلى واحد فقط , أصح واحد) .

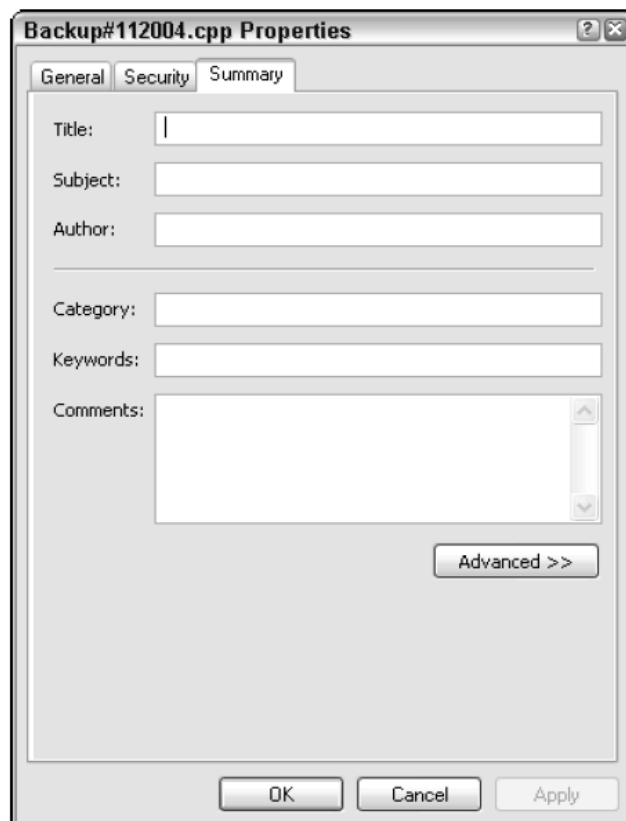
Add Summary Information to the File

إضافة ملخص المعلومات إلى الملف

عندما تضغط بيمين الفأرة على الملف في Windows XP , 2000 , ربما تلاحظ تبويب في أعلى نافذة الخصائص تحت عنوان summary . مثل الذي يعرض في الشكل التالي (XP) والشكل التالي في (2000



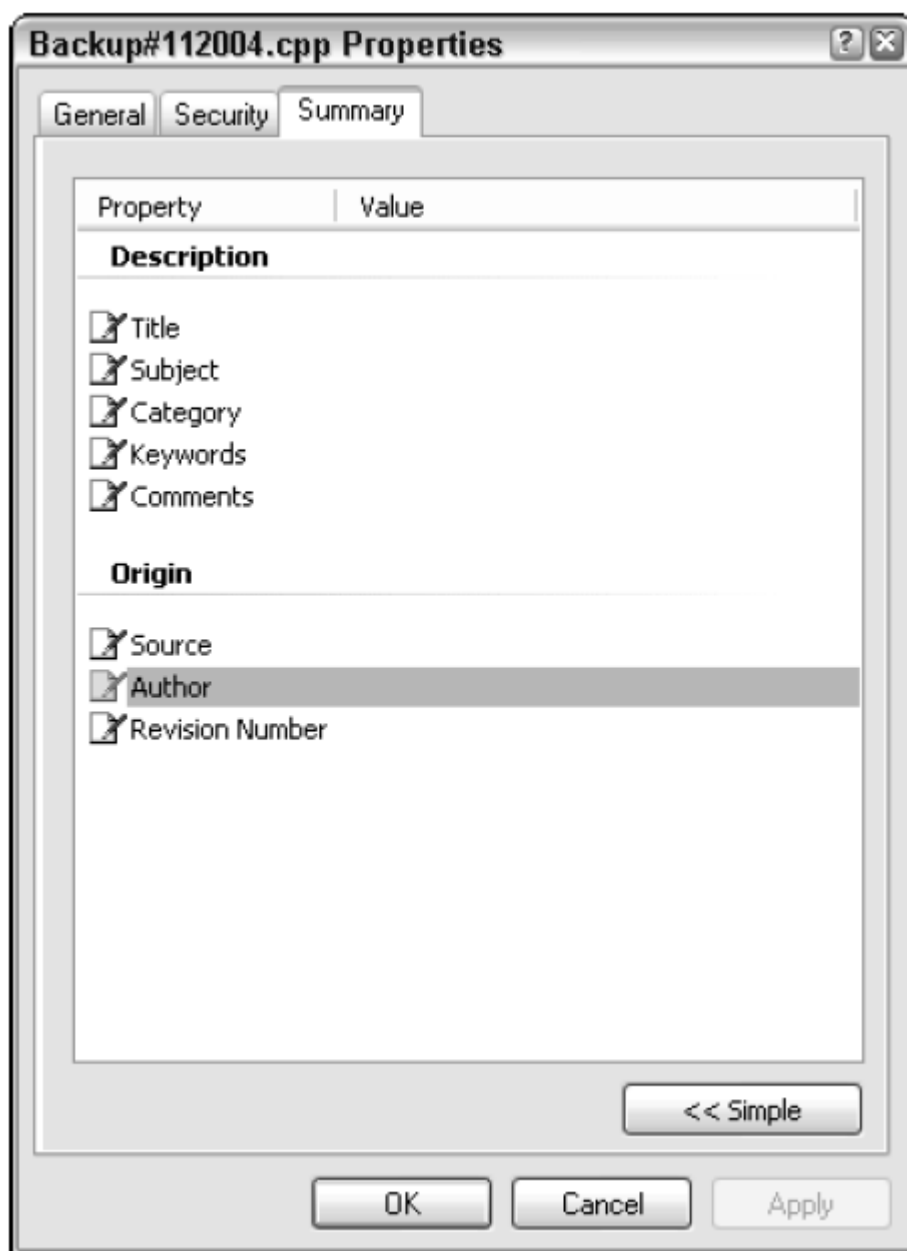
إذا قمت بالضغط على تبويب Summary , ستري أنه يحتوى على عدة مربعات نصية يمكن ان تقوم بملئها (انظر الشكل)



هذه الحقول هي :

- ☐ Title (العنوان)
- ☐ Subject (موضوع)
- ☐ Author (المؤلف)
- ☐ Category (التصنيف)
- ☐ Keywords (الكلمات المفتاحية)
- ☐ Comments (التعليقات)

ستلاحظ أيضاً زرار تحت إسم >> Advanced . بضغظ هذا الزرار يغير مربع الخصائص (انظر الشكل) وإضافة حقول زيادة لتقوم بملئها .



☐ Source (المصدر)

☐ Revision number (رقم المراجعة)

كما يمكن الإخبار من عناوين الحقول . هذه مثالية لإدخال تفاصيل عن مصدر الكود الخاص بك . الشكل التالي يعرض مربع Summary مكتمل .

The screenshot shows a Windows-style dialog box titled "Backup#112004.cpp Properties". It has three tabs: "General", "Security", and "Summary", with "Summary" currently selected. The dialog contains a table with two columns: "Property" and "Value".

Property	Value
Description	
<input checked="" type="checkbox"/> Title	Backup#112004
<input checked="" type="checkbox"/> Subject	C++ Backup program
<input checked="" type="checkbox"/> Category	C++
<input checked="" type="checkbox"/> Keywords	C++ Backup 112004
<input checked="" type="checkbox"/> Comments	
Origin	
<input checked="" type="checkbox"/> Source	
<input checked="" type="checkbox"/> Author	A N Other
<input checked="" type="checkbox"/> Revision Number	1.2.01

At the bottom of the dialog, there are three buttons: "OK", "Cancel", and "Apply". Above the "OK" button, there is a button labeled "<< Simple".

عندما تنهى اتمام ملئ الحقول , اضغط OK . وستحفظ المعلومات تلقائياً وتكون متاحة للمراجعة والتحرير في مراحل فيما بعد .

يمكن أن تبحث عن المعلومات التي قد خزنتها في لوحة المخلص Summary بنفس الطريقة التي تبحث بها عن نص آخر يمكن أن يحتويه الملف . إذا أضفنا الكلمة Bimbamboozle إلى لوحة الملخص . سنقوم بالبحث عنا طبقاً لنفس النص (انظر الشكل)

The screenshot shows a small search dialog box with the text: "What word or words do you remember in the name or contents of the file?". Below this text is a text input field containing the word "Bimbamboozle". Under the input field, there are two radio buttons: "Only search file names" (which is unselected) and "Search anywhere in the file" (which is selected). At the bottom of the dialog, there are two buttons: "Back" and "Search".

يمكنك ان تختار لوحة الملخص للتفاصيل , يمكنك أن تقوم بمضاعفة المعلومات هنا أو تستخدمها فقط هنا . اى نهج تختار متوقف عليك ولكن إستخدام الملخص يعنى حفظ الكود نظيف .

Version Control — Looking Beyond Release

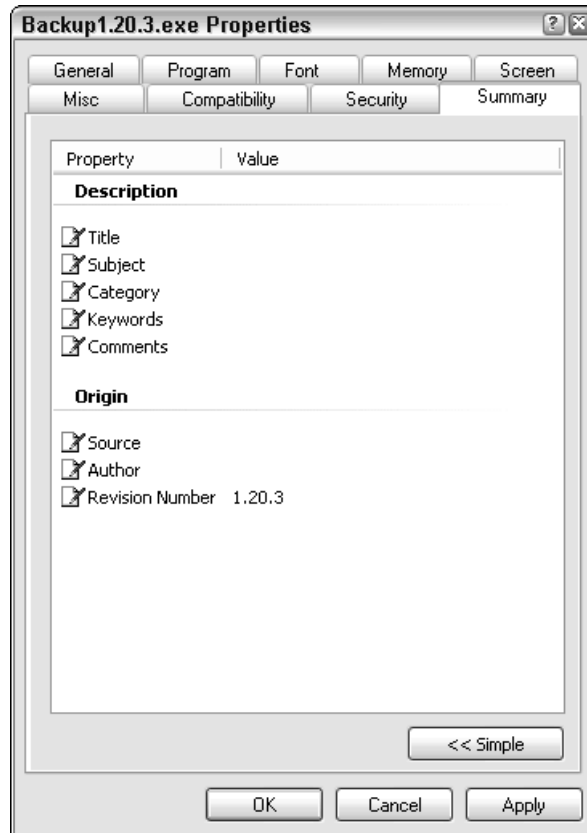
تذكر أن التحكم في الإصدار لا ينتهى بالإصدار الأول من برنامجك .إذا كان أي شيء, فيزيد عبء العمل الخاص بك , بسبب انك لن يكون لديك تحكم فى التغيرات التى تجرى على مصدر الكود . ولكن ستحتاج أن تتعقب الإصدارات المختلفة للكود النهائى (سواء كود مترجم أو كود مفسر),

أفضل طريقتين لتعقب أثر الكود المفسر هما :

1 - قم بإضافته إلى إسم الملف (انظر الشكل) للملف التنفيذى والملف المضغوط إذا إخترت ان تضغط الملف التنفيذى .



2 - إضافته للوحة الملخص Summary (انظر الشكل) إذا قمت بعمل هذا , تذكر أن تضغط الملف , خاصة إذا كنت تقوم بتحميل الملفات إلى الإنترنت , بسبب أن معلومات الملخص ربما لا تنجو خلال العملية.



الإختبار الثالث متاح إذا كان المفسر أو البرنامج الذى تستخدمه إنشاء تثبيت التطبيق يمكنك من إضافة معلومات الإصدار إلى الملف عند إنشائه ,إذا كان هذا ممكن , قم بإستخدامه فى الإفضلية فى لوحة الملخص .

إضافة إلى تعقب أثر إصدار مصدر الكود والملف التنفيذى , تأكد أن تضيف معلومات عن الإصدار إلى التطبيق الفعلى لذا نجد الأشخاص الذين يقوم لتشغيل التطبيق فيعرفون أى إصدار معهم .

Software Version Control

صدق او لاتصدق , هناك تطبيقات برمجية يمكن ان تساعدك فى تعقب كل الكود والملفات التنفيذية التى أنشأتها عند البرمجة , لا يزال أفضل , هناك برامج مجانية تدعك تفعل هذا !

وهنا لسرد سريع لبعض من أفضل ما هو متاح

- GNU Revision Control System, <ftp://ftp.gnu.org/pub/gnu> . قوى جداً , حتى الآن سهل الإستخدام فى التحكم فى إصدار النظام . هذا البرنامج فريد فى الحقيقة فبدلاً من تخزين كامل الإصدارات للكود , فهو يتعقب التغييرات والتخزين بمفرده للكل . وهى طريقة جيدة لتعقب الكود . ويعمل على Windows وما قبله أنظمة Dos .
- JediVCS (Version Control System), <http://www.freevcs.de> . السابق يعرف بـ FreeVCS , JediVCS هو برنامج مصدر مفتوح نظام التحكم فى الإصدار الذى يأتى إما إصدار وحيد يعمل مع أى لغة أو إصدارات محددة لـ VC++ Expert , BCB ,Delphe .
- SourceJammer, <http://sourcejammer.org> . هو نظام تحكم فى الإصدار وقد كتب بلغة Java , لذا ينبغى ان يعمل على أى نظام لديها Java VM (virtual Machine)
- Arch, <http://wiki.gnuarch.org> . هذا نظام آخر للتحكم فى الإصدار متاح لأنظمة تشغيل متنوعة .
- TortoiseCVS
.. يدعك تجرى احداث تحكم فى الإصدار من خلال مدير الملفات لمستكشف الـ Windows , فإنت تعمل مع الملفات مباشرة . عادةً من خلال الضغط بيمين على قائمة Context . فهو يوفر مميزات هائلة فى أنه سهل الإستخدام , ولكن يمكن أن يكون أبطئ , لأنها متكاملة بشكل كبير فى نظام التشغيل . ومع هذا , فهى مجانية وسهلة الإستخدام وتستحق التدقيق بها.

Summary

الملخص

فى هذا الفصل قمنا بالبحث فى كيف تقوم بتنظيم نفسك والنظام للإستعداد للبرمجة وأيضاً كيف تتعقب الإصدارات المختلفة للكود والملف التنفيذى التى تنتجها والمزيد المزيد من البرمجة .

بينما تنظيم نفسك إختيارياً , تنظيم الحاسب هام إذا كنت ستكتب كود فعال . الكود الغير منظم هو مسار جيد للمشاكل . لذا تجنب إلم الرأس مبكراً وأعد كل الملفات التى قد أنشأتها .

15

Compiling Code and Alternatives to Compiling

تفسير مصدر الكود الذى قضيت فيه أيام , أسابيع , او حتى شهور أستعباداً هى عامة ترى هامة لأنها ليست فقط تمكّنك من إختبار الكود كتطبيق , ولكن يعمل أيضاً كعلامة فارقة — كل تفسير لمصدر الكود تقوم بعمله يعنى انك اقتربت من النهاية .

فى هذا الفصل , نأخذ نظرة مقربة على عملية التفسير وفحص المميزات والعيوب للعملية إضافة إلى النظر إلى طرق لجعل عملية التفسير أمثل واكمل .

بالإضافة إلى ذلك , نفحص أيضاً بدائل التفسير متاحة لك إذا إخترت إن تستخدم لغة كتابات برمجية بدلا من ذلك .

Compiling Code

تفسير الكود

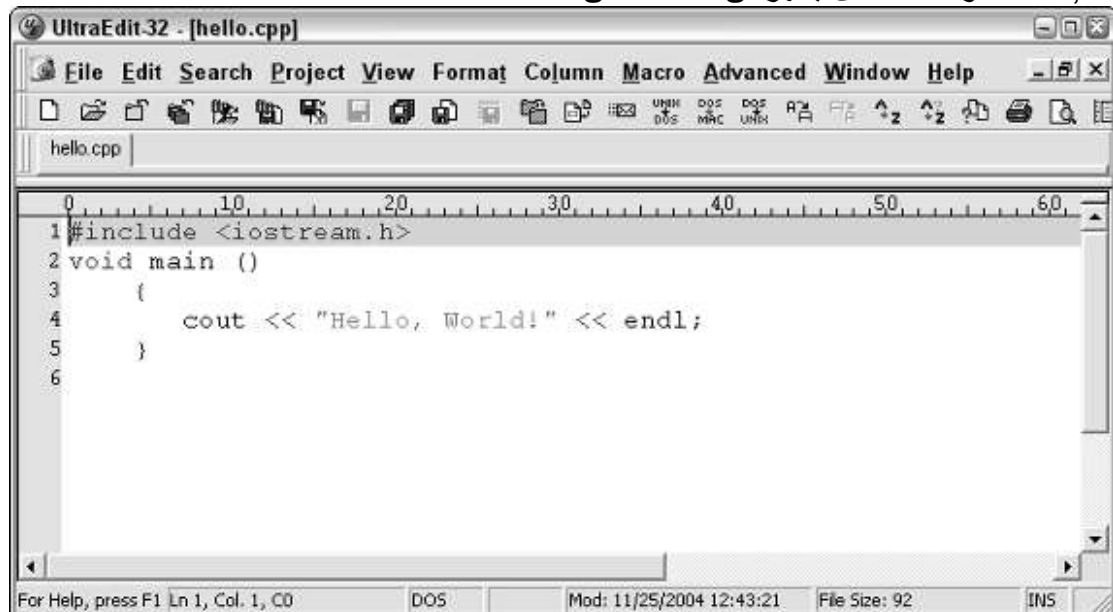
لقد رأينا بالفعل كيف نقوم بتفسير كود C++ بإستخدام المفسر المجاني Borland C++ command-line . عملية بسيطة نسبيا ومباشرة .

- 1 - تكتب مصدر الكود داخل محرر النصوص .
- 2 - الملف الذى يحتوى على مصدر الكود يحفظ بإمتداد مناسب .
- 3 - يأخذ المفسر الملف ومصدر الكود ويتم معالجتهم , وينشئ ملف تنفيذى , الملف المنشئ هو تطبيق مستقل يمكن أن يعمل على أنظمة أخرى .

بينما كل المفسرات تنشئ ملف تنفيذى يمكن ان تعمل , ليس كلهم ينشئ ملف تنفيذى بصدق " مستقل " . بعض المفسرات , مثل الذى يعمل مع Visual Basic أو تلك التى يتم شحنها مع Windows وهو المفسر Borland . تقوم بإنشاء ملفات تنفيذية تعتمد على ملفات إضافية اكون مثبتة على النظام الذى يقوم بتشغيل الملف التنفيذى . الملف التنفيذى المنشأ بواسطتهم يعنى ان يتطلب مكتبة وقت التشغيل Runtime Library أن تكون مثبتة على النظام الذى سيقوم بتشغيل الملف . مناقشة هذه الانواع من المكتبات هو أبعد منا هنا . ولكن من الهام لك ان تدركه هنا هو ان كل الملفات التنفيذيه ليست كلها مستقلة .

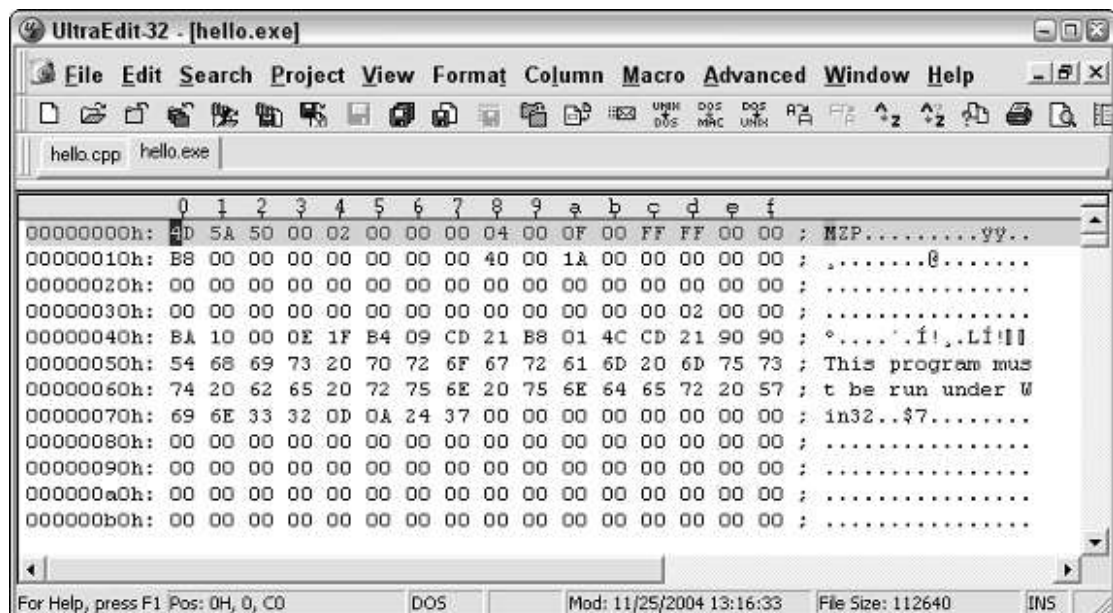
بعبارة أخرى , عملية التفسير تأخذ الكود المكتوب على شكل نص عادي بواسطة البشر ومقروء بواسطة البشر (أو على الأقل , مقروء للبعض) ومن ثم يقوم بتحويله إلى شكل يفهمه الحاسب .

لذا , تأخذ الكود مثل الذى يظهر فى الشكل التالى



```
UltraEdit 32 - [hello.cpp]
File Edit Search Project View Format Column Macro Advanced Window Help
hello.cpp
0 10 20 30 40 50 60
1 #include <iostream.h>
2 void main ()
3 {
4     cout << "Hello, World!" << endl;
5 }
6
For Help, press F1 Ln 1, Col. 1, C0 DOS Mod: 11/25/2004 12:43:21 File Size: 92 INS
```

ثم يقوم بتحويله إلى كود مثل الذى يظهر أمامك فى الشكل التالى



```
UltraEdit 32 - [hello.exe]
File Edit Search Project View Format Column Macro Advanced Window Help
hello.cpp hello.exe
0 1 2 3 4 5 6 7 8 9 a b c d e f
00000000h: 4D 5A 50 00 02 00 00 00 04 00 0F 00 FF FF 00 00 ; MZP.....yy..
00000010h: B8 00 00 00 00 00 00 00 40 00 1A 00 00 00 00 00 ; .....@.....
00000020h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; .....
00000030h: 00 00 00 00 00 00 00 00 00 00 00 00 02 00 00 00 ; .....
00000040h: BA 10 00 0E 1F B4 09 CD 21 B8 01 4C CD 21 90 90 ; .....f!..Li!!
00000050h: 54 68 69 73 20 70 72 6F 67 72 61 6D 20 6D 75 73 ; This program mus
00000060h: 74 20 62 65 20 72 75 6E 20 75 6E 64 65 72 20 57 ; t be run under W
00000070h: 69 6E 33 32 0D 0A 24 37 00 00 00 00 00 00 00 00 ; in32..$7.....
00000080h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; .....
00000090h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; .....
000000a0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; .....
000000b0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; .....
For Help, press F1 Pos: 0H, 0, C0 DOS Mod: 11/25/2004 13:16:33 File Size: 112640 INS
```

يقوم المفسر بتحويل مصدر الكود الذى كتبته إلى كود يفهمه الحاسب .

ولكن هناك اكثر من ذلك . لاحظ كيف تترجم بعض السطور من مصدر الكود إلى الكثير والكثير من اسطر التعليمات فى نهاية التطبيق . خمسة أسطر من الكود (ويشمل هذا الأقواس المتعرجة) . مثال ذلك:

```
#include <iostream.h>
void main()
{
    cout << "Hello, World!" << endl;
}
```

يترجم إلى مئات الأسطر من التعليمات في المخرج النهائي . السؤال الواضح هنا هو "لماذا " ؟

حسناً , الإجابة تحت سطح أمور أكثر تعقيداً مما يبدو في البداية . حتى اللغات المعقدة مثال C++ في الحقيقة تبقى المبرمج من واقع أشد قسوة من كم تعقيد التعليمات البرمجية عندما تكون لغة مفهومة لدى الحاسب . لذا مجرد كتابة سطر كود يخرج نص على الشاشة يشمل على الكثير من العمل خلف المشهد التي بالفعل لا تقوم بتغطيتها في الكود الذي تكتبه .

خذ الامر البرمجي cout . هذا امر في C++ يأخذ بعض البيانات ويخرجها على الشاشة . ولكن ما يقف وراء كلمة cout هو طريقة إختزال تمثل المزيد من الكود . فأنت تكتب Cout في مصدر الكود فيعرف المفسر ماذا تريد ان تفعل . في هذه الحالة يخرج شيئاً ما إلى الشاشة . ومن ثم يكتب لك الكود لمعالجة هذا .

لذا , معظم الكود الذي تكتبه هو فقط يتكون من تدوين مختصر للمفسر لينشئ كود ويدخله في التطبيق . وهذا يوقفك أن تكتب الكود بنفسك — وهذا يعني أنك لن يكون عليك ان تدير العجلة كل مرة تريد ان تفعل نفس الشيء .

وهذا يقودنا إلى عدة أسئلة شيقة تحتاج إلى إجابة .

Are All Compilers the Same?

هل كل المفسرات مثل بعضها ؟

سؤال رائع ! دعنا ننظر فقط في لغة C++ . إذا قمت بعمل بحث في الإنترنت , ستمر بمختلف المفسرات التي يمكن ان تستخدمها (كثيراً منهم مجاني) .

إليك زوجين من البدائل لمفسر Borland C++ الذي قمنا بإستخدامه في هذا الكتاب .

❑ Digital Mars, www.digitalmars.com

❑ DJGPP, www.delorie.com/djgpp

ولكن السؤال هو , هل كل المفسرات متشابهة ؟ حسناً , هناك طريقة واحدة لنعرف وهذه هي لنلقى نظرة عليها.

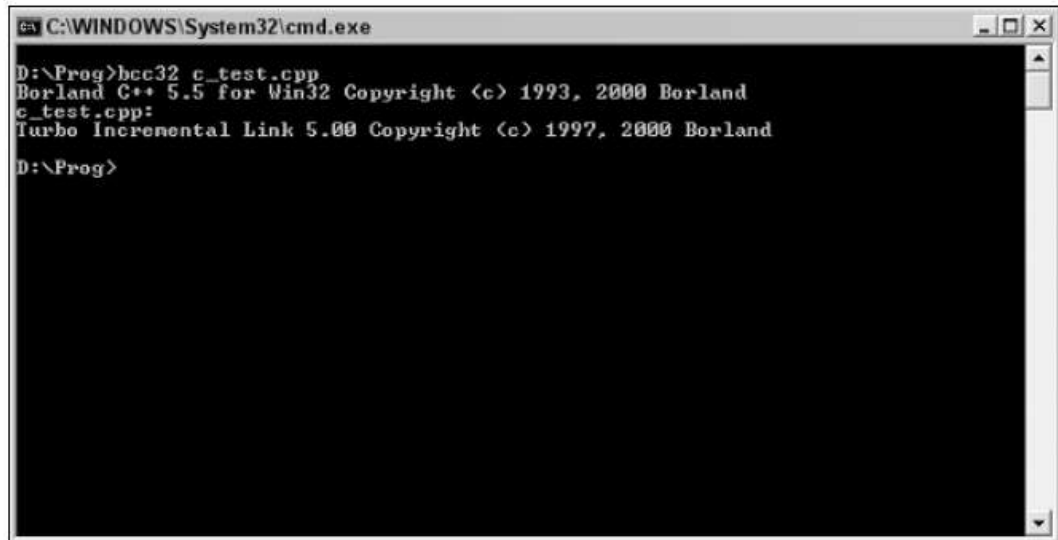
لفعل هذا سأقوم بتفسير بعض الكود البسيط بإستخدام مفسر Borland ومفسر Digital Mars ونرى إذا كان هناك إختلافات .

التثبيث لهذين البرنامجين مختلف تماماً , مع المفسر Digital Mars كل ما تحتاجه هو فك الضغط للملفات من التحميل إلى مجلد. مع المفسر DJGPP الإعداد معقد جداً , ويجب أن تتبع التعليمات المزودة لك بحرص . أنا احذرك الآن أن DJGPP ليس من اجل التردد وأكثر صرامة من أى من الآخرين . وبالتالي يجعله اقل تسامحاً مع الكود . هي مفسر جيد . ولكن أنا أوصي به للمبتدئين .

مثال الكود الأول الذى سنستخدمه هو أبسط واحد معتمداً على مثال Hello , world ! (يحتوى الملف على كود C++ لتطبيق يسمى c-test.cpp)

```
#include <iostream.h>
void main ()
{
    cout << "This is a compiler test." << endl;
}
```

انت تعرف كيف تقوم بتفسير هذا بواسطة Borland compiler . (انظر الشكل)



```
C:\WINDOWS\System32\cmd.exe
D:\Prog>bcc32 c_test.cpp
Borland C++ 5.5 for Win32 Copyright (c) 1993, 2000 Borland
c_test.cpp:
Turbo Incremental Link 5.00 Copyright (c) 1997, 2000 Borland
D:\Prog>
```

ينشأ هذا ملف 110 kb .

الآن لتفسير هذا الكود باستخدام Digital Mars compiler على النظام الخاص بى , لقد قمت بتثبيت المفسر على مجلد اسمه dm على d: . أسهل طريقة لتجعل المفسر يعمل ان تقوم بنسخ مصدر الكود إلى المجلد Bin الموجود خلال المجلد dm . ومن ثم يمكنك أن تفتح command Prompt وتقوم بالوصول إليه .

قم بالتغيير إلى D: (انظر الشكل)



```
C:\WINDOWS\System32\cmd.exe
C:\>d:
D:\>
```

قم بالتغيير على المجلد المناسب ثم المجلد الفرعى . (انظر الشكل)

```
C:\WINDOWS\System32\cmd.exe
C:\>d:
D:\>cd dn\bin
D:\dn\bin>
```

الان يمكنك تشغيل المفسر مع الأمر التالي .

dmc c_test.cpp

```
C:\WINDOWS\System32\cmd.exe
C:\>d:
D:\>cd dn\bin
D:\dn\bin>dmc c_test.cpp
link c_test,,,user32*home132\noi;
D:\dn\bin>
```

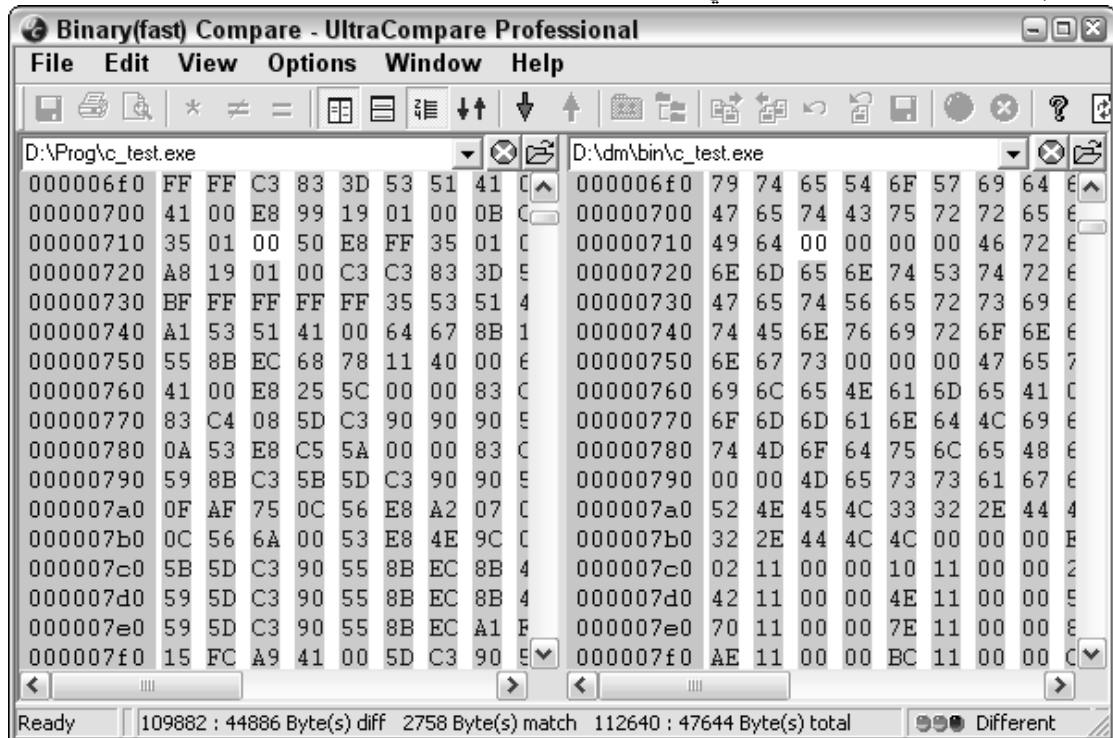
هذا يقوم أيضاً بإنشاء ملف تنفيذي , ولكن هناك إختلافين تم ملاحظتهما :

- المفسر Digital Mars حقيقياً أسرع مفسر . وينبغي ان تلاحظ أنه قام بتفسير مصدر الكود أسرع بكثير من المفسر Borland .
- الملف التنفيذي أقل بكثير من الملف التنفيذي المنشئ بواسطة المفسر Borland , فى هذه الحالة 47 K

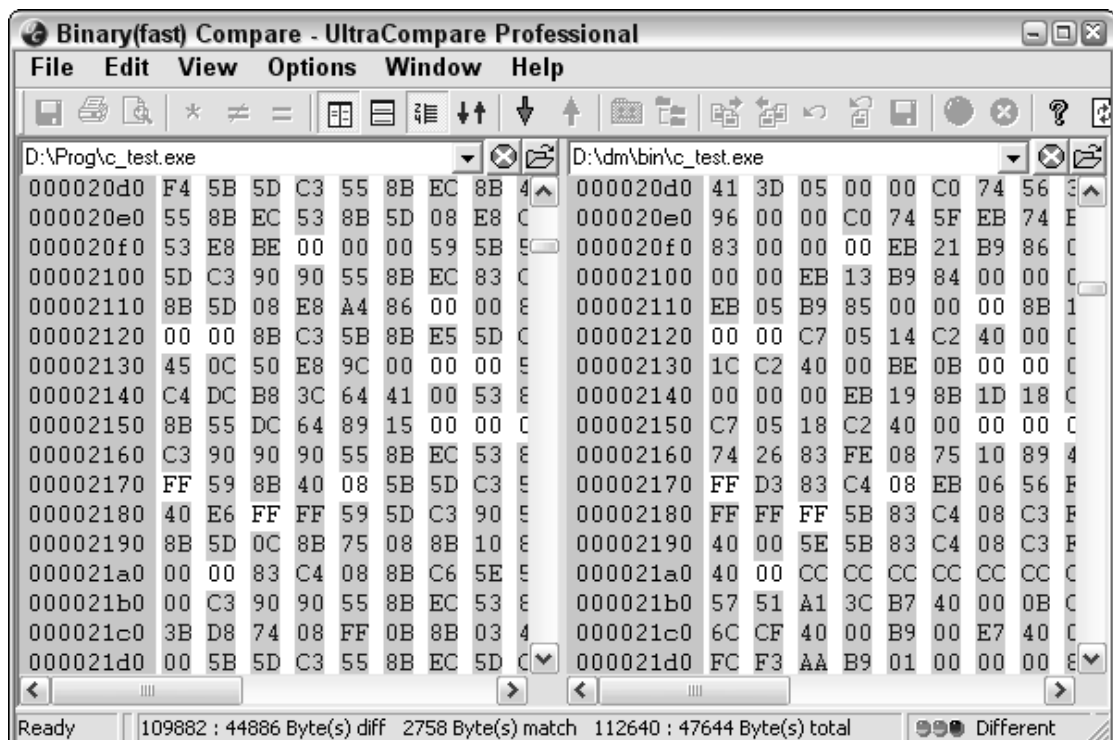
لذا , يمكنك فى الحال أن ترى انه لإنتاج نفس النتيجة (فى هذه الحالة , نتيجة بسيطة) الإثنين يقومان بإنشاء مخرجات ملفات مختلفة .

لمعلومات أكثر عن إستخدام مفسر Digital Mars , إكتب DMC فى نافذة command Prompt وأقرأ التعليمات على الشاشة .

إذا كان لديك تشوش عقلي , انت من المحتمل تعتقد ان الملفين كلاهما نفس الشيء وأن مفسر Borland يقوم بإضافة معلومات إضافية للملف وإنما كان . يمكنني ان أوكد لك أن الملفين مختلفين تماماً . الشكل التالي يعرض الملفات التنفيذية التي صنعت بواسطة كلا المفسرين محملة في محرر النصوص UltraEdit (www.ultraedit.com) وهو برنامج يمكن أن يسلط الضوء على الاختلافات في الملفين . لقد قمت باختيار مواقع عشوائية في الملف . الاختلافات تعرض بواسطة الكتل الملونة . بينما النقاط المتطابقة تظهر بدون ألوان . كما ترى ., هناك اختلافات كبيرة جدا في الملفات.



مع ذلك , هناك انماط للتشابه خلال الملفات , كما يظهر بوضوح في الشكل التالي . الملفات مختلفة , ولكن أيضاً بها بعض التشابه .



لذا , حتى على الرغم من أن الملفات بالتأكيد ليست واحدة . فإنهم ليسو غير متشابهين تماماً .

دعنا نلقي نظرة على ما يحدث عندما نقوم بتفسير شيئاً ما هو اكبر قليلاً مع المفسر digital Mars . دعنا نرى إذا كان هناك إختلافات .

عن هذه التجربة , سنستخدم الكود البسيط للأله الحاسبة التي لديها الواجهة الكاملة .

```
#include <iostream.h>
void calc()
{
    float num1;
    float num2;
    char op;
    float ans;
    cout << "Simple Calculator Application" << endl;
    cout << "Enter two numbers and perform an operation." << endl;
    cout << "Enjoy!" << endl;
    cout << "-----" << endl;
    cout << "Enter the first number and press ENTER:" << endl;
    cin >> num1;
    cout << "Enter the second number and press ENTER:" << endl;
    cin >> num2;
    cout << "You entered " << num1 << " and " << num2 << endl;
    cout
        << "Press A followed by ENTER to add the two numbers."
        << endl
        << "Press S followed by ENTER to subtract the two numbers."
        << endl
        << "Press M followed by ENTER to multiply the two numbers."
        << endl
        << "Press D followed by ENTER to divide the two numbers."
        << endl;
    cout << "-----" << endl;
    cin >> op;
    if (op == 65)
    {
        ans = num1 + num2;
    }
    if (op == 83)
    {
        ans = num1 - num2;
    }
    if (op == 77)
    {
        ans = num1 * num2;
    }
    if (op == 68)
    {
        ans = num1 / num2;
    }
    cout << "The answer is " << ans << endl;
    cout << "Thanks for using the calculator." << endl;
    cout << "Application now exiting ... " << endl;
}

void helpsystem()
```

```

    {
        char op;
        cout << "Help System" << endl;
        cout << "-----" << endl;
        cout << "To work this application you enter two numbers and then choose" <<
endl;
        cout << "whether you want those numbers added together, subtracted, " <<
endl;
        cout << "multiplied, or divided. You choose this operation by inputting" <<
endl;
        cout << "A, S, M, or D respectively." << endl;
        cout << "The result is displayed on-screen." << endl;
        cout << "Are you ready to use this applications?" << endl;
        cout << "Press Y followed by ENTER to continue." << endl;
        cout << "Press N followed by ENTER to exit." << endl;
        cout << "Press ? followed by ENTER for help." << endl;
        cin >> op;
        if (op == 89)
        {
            calc();
        }
        if (op == 63)
        {
            helpsystem();
        }
        if (op == 78)
        {
        }
    }
}

void main()
{
    char op;
    cout << "Simple Calculator Application" << endl;
    cout << "Enter two numbers and perform an operation." << endl;
    cout << "Enjoy!" << endl;
    cout << "Are you ready to use this applications?" << endl;
    cout << "Press Y followed by ENTER to continue." << endl;
    cout << "Press N followed by ENTER to exit." << endl;
    cout << "Press ? followed by ENTER for help." << endl;
    cin >> op;
    if (op == 89)
    {
        calc();
    }
    if (op == 63)
    {
        helpsystem();
    }
    if (op == 78)
    {
    }
}

```


عندما يتم تفسير الكود بواسطة المفسر Borland التطبيق النهائي يكون 148kb. وعندما نفسر هذا الكود بواسطة المفسر Digital Mars التطبيق النهائي يكون 70 kb . أقل بقريب من 80kb . التطبيقات الصغيرة تفسر أسرع . تأخذ مساحة تخزين أقل , تحمل بأكثر سرعة , وتستخدم موارد نظام أقل عند التشغيل .

لذا في الإجابة على سؤال هل كل المفسرات متشابهة , أعتقد أننا يمكننا بإمان أن نقول انه بينما تعمل التطبيقات الفعلية بنفس الطريقة , الإسلوب المتبع لإنجاز العمل في كلا المفسرين مختلف تماماً .

Error Handling

معالجة الخطأ

ماذا عن معالجة الأخطاء ؟ هل المفسرات تعالج اخطاء بنفس الطريقة ؟

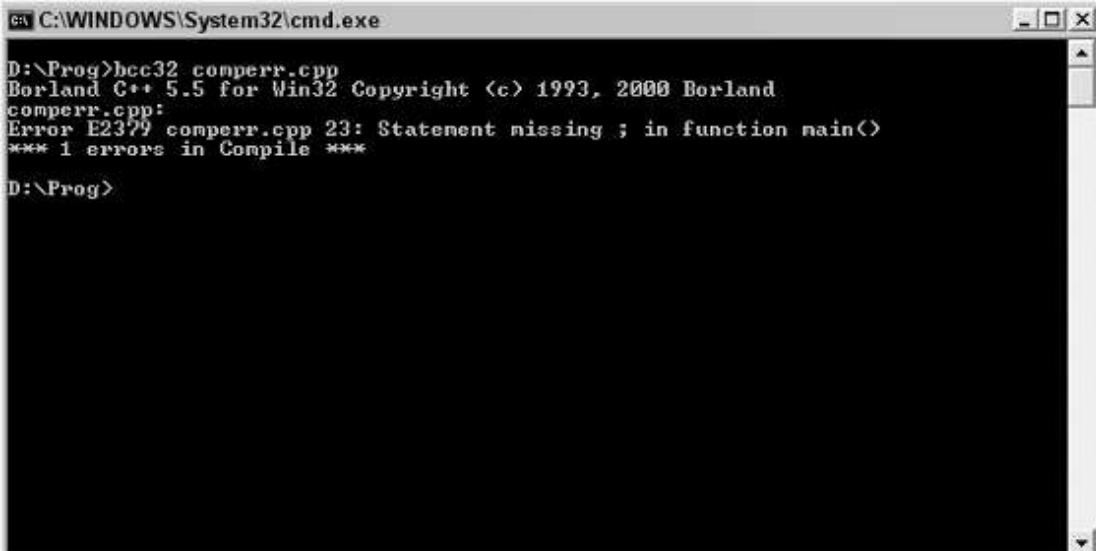
بسبب أن الأخطاء هي شيئاً ما يواجهها المفسر ولا تكون متوقعة حقاً أن تمر خلال مرآهنة ان المفسرات لن يقوموا بإخراج نفس رسالة الخطأ كلمة كلمة . بعد كل شيء , كل مفسر مبرمج على ان يصدر رسائل اخطاء مخصصة به وأيضاً يعطى اكواد أخطاء تساعده المستخدم ان يكتشف مصدر الخطأ بنفسه .

دعنا نعيد زيارة الكود من الفصل التاسع "Debugging"

```
#include <iostream.h>
void main ()
{
    float num1;
    float num2;
    char op;
    float ans;
    cout << "Please enter a number: ";
    cin >> num1;
    cout << "Please enter another number: ";
    cin >> num2;
    cout << "Press A to add the two numbers."
    << endl
    << "Press S to subtract the two numbers."
    << endl
    << "Press M to multiply the two numbers."
    << endl
    << "Press D to divide the two numbers."
    << endl;
    cin >> op;
    if (op == 65)
        ans = num1 + num2;
    if (op == 83)
        ans = num1 - num2;
    if (op == 77)
        ans = num1 * num2;
    if (op == 68)
        ans = num1 / num2;
    cout << "The answer is " << ans << endl;
}
```

هذا الكود يحتوى على خطأ (فى حالة انك مهتم , يمكنك إكتشافه) ما يمكنك فعله هو القيام بتفسير الكود وسيعطيك بعض المناظر التى تتسبب فى الخطأ .

أولاً , قم بتفسيره بواسطة المفسر Borland و نبه نفسك للخطأ الذى تحصل عليه من هذا المفسر (انظر الشكل)



```
C:\WINDOWS\System32\cmd.exe

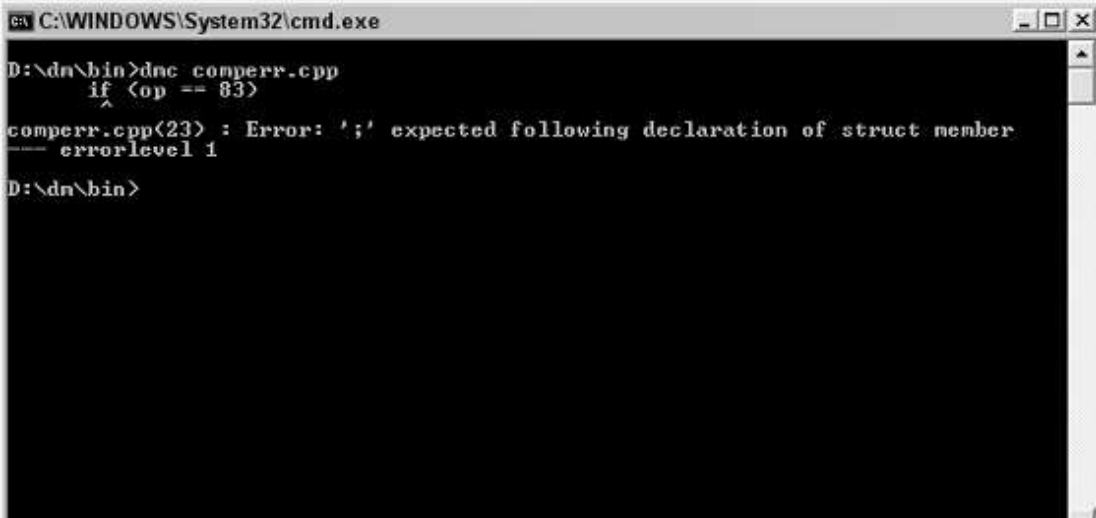
D:\Prog>bcc32 comperr.cpp
Borland C++ 5.5 for Win32 Copyright (c) 1993, 2000 Borland
comperr.cpp:
Error E2379 comperr.cpp 23: Statement missing ; in function main()
*** 1 errors in Compile ***

D:\Prog>
```

```
D:\Prog>bcc32 calcerr.cpp
Borland C++ 5.5 for Win32 Copyright (c) 1993, 2000 Borland
calcerr.cpp:
Error E2379 calcerr.cpp 23: Statement missing ; in function main()
*** 1 errors in Compile ***
```

هناك خطأ واحد فى السطر 23 . تذكر ماذا كان ؟

قبل اخذ نظرة , دعنا نفسر بعض الكود بإستخدام مفسر Digital Mars . المخرجات التى ستحصل عليها فى الشكل التالى .



```
C:\WINDOWS\System32\cmd.exe

D:\dn\bin>dnc comperr.cpp
      if <op == 83>
      ^
comperr.cpp(23) : Error: ';' expected following declaration of struct member
----- errorlevel 1

D:\dn\bin>
```

```
D:\dm\bin>dmc calcerr.cpp
if (op == 83)
^
calcerr.cpp(23) : Error: ';' expected following declaration of struct member
--- errorlevel 1
```

في الحال ينبغي ان ترى أن هذا المفسر يعطى مخرجات مختلفة . إضافة إلى انه يعطى رقم السطر ووصف للخطأ (الوصف للخطأ ليس ذا معنى للمبتدئ كالذى يعطى بواسطة المفسر Borland اعتقد) هو ايضاً يعطينا لمحة عن الكود — السطر 23 في الكود .

هل ترى الخطا حتى الآن ؟ دعنا نركز على الكود في ناحية الخطأ .

```
...
cin >> op;
if (op == 65)
    ans = num1 + num2
if (op == 83)
    ans = num1 - num2;
if (op == 77)
    ans = num1 * num2;
...
```

لاحظ إين هي العلامة Semicolon المفقودة الان ؟ في الجملة التي تسبق السطر 23 .

```
...
cin >> op;
if (op == 65)
    ans = num1 + num2
if (op == 83)
    ans = num1 - num2;
if (op == 77)
    ans = num1 * num2;
...
```

تذكر التحذير الذى يعطيه الكود مع المفسر Borland لأن هناك متغير قد تم الإعلان عنه ولم يستخدم ؟

```
#include <iostream.h>
void main ()
{
    float num1;
    float num2;
    char op;
    int var1;
    float ans;
    var1 = 7;
    cout << "Please enter a number: ";
    cin >> num1;
    cout << "Please enter another number: ";
    cin >> num2;
    cout << "Press A to add the two numbers."
        << endl
        << "Press S to subtract the two numbers."
        << endl
        << "Press M to multiply the two numbers."
        << endl
```

```

        << "Press D to divide the two numbers."
        << endl;
    cin >> op;
    if (op == 65)
        ans = num1 + num2;
    if (op == 83)
        ans = num1 - num2;
    if (op == 77)
        ans = num1 * num2;
    if (op == 68)
        ans = num1 / num2;
    cout << "The answer is " << ans << endl;
}

```

يعطى المفسر Borland هذه الرسالة (الشكل التالي)

```

C:\WINDOWS\System32\cmd.exe
D:\Prog>bcc32 comperr2.cpp
Borland C++ 5.5 for Win32 Copyright (c) 1993, 2000 Borland
comperr2.cpp:
Warning W8004 comperr2.cpp 32: 'var1' is assigned a value that is never used in
function main()
Turbo Incremental Link 5.00 Copyright (c) 1997, 2000 Borland
D:\Prog>_

```

```

D:\Prog>bcc32 calcerr3.cpp
Borland C++ 5.5 for Win32 Copyright (c) 1993, 2000 Borland
calcerr3.cpp:
Warning W8004 calcerr3.cpp 32: 'var1' is assigned a value that is never used in
function main()
Turbo Incremental Link 5.00 Copyright (c) 1997, 2000 Borland

```

الكود مازال يفسر وملف تنفيذي قد تم إنشائه , فقط هنا يحذرك المفسر من المتغير الغير مستخدم .

المفسر Digital Mars يختار أن يتجاهل القضية معاً وينشأ ملف تنفيذي بشكل طبيعي (انظر الشكل)

```

D:\dm\bin>dmc calcerr3.cpp
link calcerr3,,user32+kernel32/noi;

```

```
C:\WINDOWS\System32\cmd.exe
D:\dn\bin>gcc comperr2.cpp
link comperr2,,,user32+kernel32/NOI;

D:\dn\bin>
```

What about Different Languages?

المفسرات لمختلف اللغات تتطلب مصدر كود مختلف كمدخلات للمفسر . هذا يعنى أن المخرجات لابد ان تكون مختلفة .

ناخذ , على سبيل المثال , لغة Java . إذا كنت تنشأ تطبيق بسيط "Hello , World ! " فى هذه اللغة فمصدر الكود الذى تحتاجه يبدو كما يلى :

```
public class HelloWorld
{
    public static void main(String args[])
    {
        System.out.println("Hello, World!");
    }
}
```

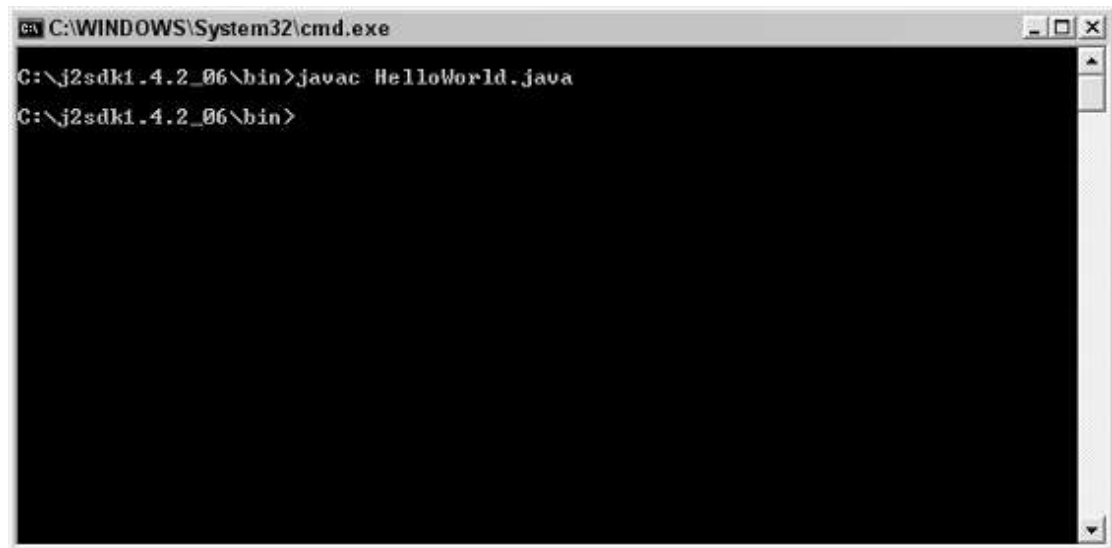
إذا قمت بحفظ هذا فى مجلد Bin فى مجلدات تثبيت Java الموجود لديك فى النظام . إعطيه اسم HelloWorld (ليطابق الاسم اسم التصنيف العام) . وتأكد ان تحفظ الملف بإمتداد java . وهذا يعنى إنك قمت بتغيير الاسم للتصنيف العام , سيكون على اسم الملف أن يتم تغييره . لذا , فى المثال التالى اسم الملف سيكون JavaTest.java

```
public class JavaTest
{
    public static void main(String args[])
    {
        System.out.println("Hello, World!");
    }
}
```

إرجع إلى <http://java.sun.com> لتعليمات التثبيت وآخر ملفات التحميل .

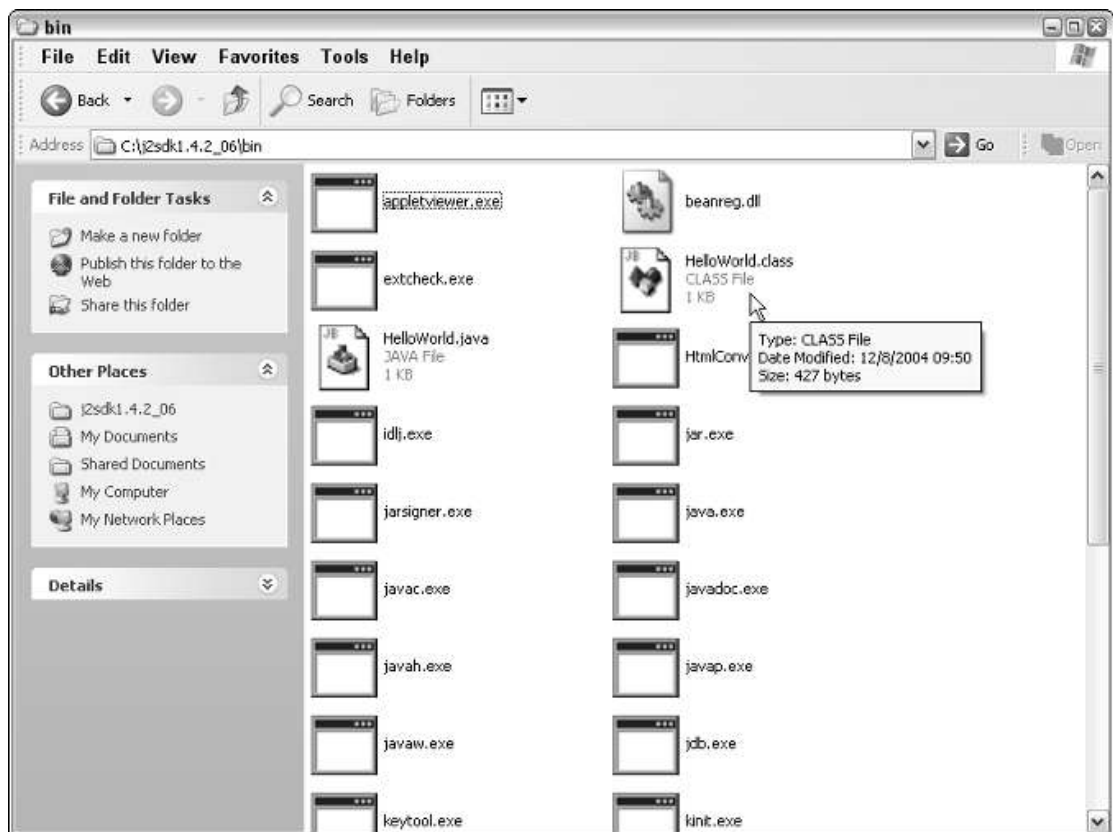
مع وجود هذا الملف فى مجلد bin لبيئة التطوير لـ Java , يمكنك الآن أن تقوم بتفسير الملف .

تفسير Command-Line لكود Java مشابه لتفسير كود c++ . توجه إلى المجلد bin من خلال نافذة Command Prompt واكتب التالي (كما في الشكل)



```
javac HelloWorld.java
```

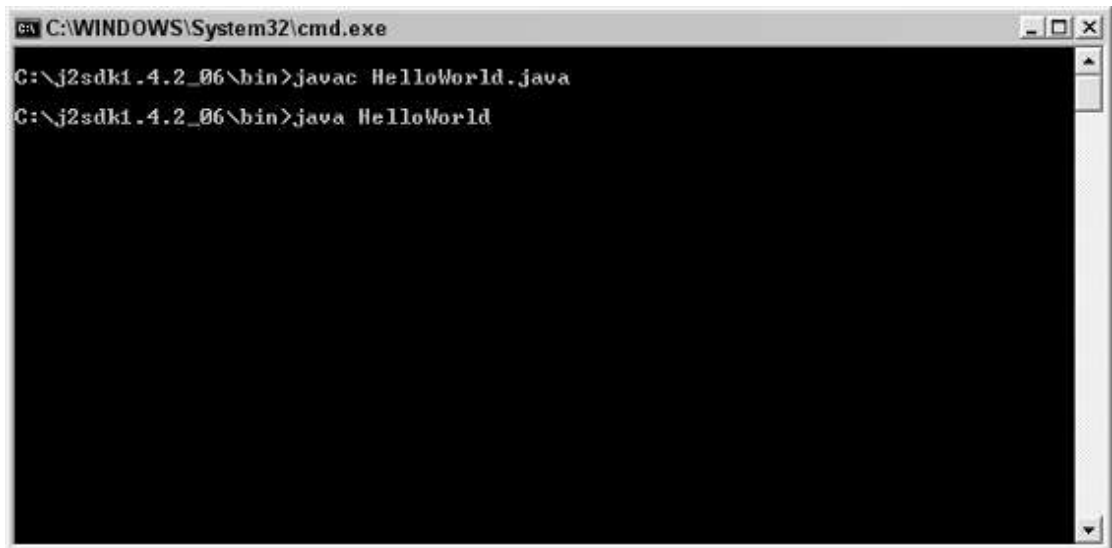
سيتم تفسير كود Java بصمت, ويتم إنشاء ما يعادل في Java للملف التنفيذي . وسيكون هذا بإسم HelloWorld.class كما في الشكل .



كود جافا المفسر يسمى CodeByte . ويفترض أن يكون هذا برنامج مستقل . مما يعني أنه يمكن أن يعمل على أي نظام حيث بيئة العمل لجافا (يعرف أيضاً بـ JVM أو Java Virtual Machine) يمكن أن يتم تحميلها وتثبيتها .

كود جافا المفسر ليس كود مستقل بل يتطلب أن تكون بيئة تشغيل جافا مثبتة على الأجهزة التي ستقوم بتشغيله . فليسيت هذه صفقة كبير في العام لأن استخدام جافا واسع الإنتشار , ولكن إذا كان احد مستخدميك يعمل على نظام windows XP , إذاً فهناك حاجة إلى ان يزور موقع جافا (<http://java.sun.com>) ويقوم بتنزيل احر إصدار للجهاز لوهمى لجافا Java virtual Machine .

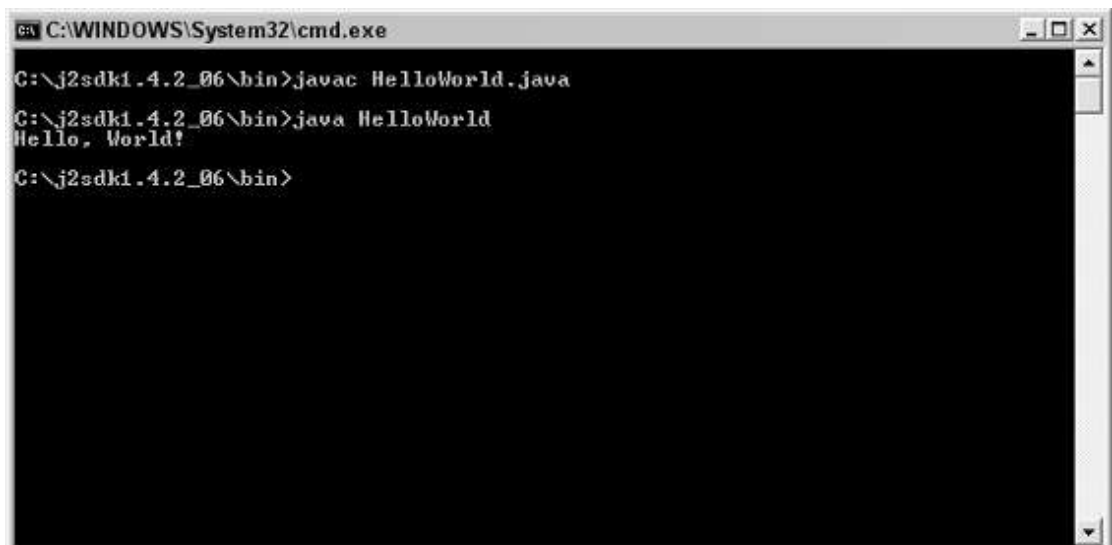
دعنا نقوم بتشغيل الكود , لتشغيل الكود , أكتب التالي في نافذة Command Prompt (انظر الشكل)



```
C:\WINDOWS\System32\cmd.exe
C:\j2sdk1.4.2_06\bin>javac HelloWorld.java
C:\j2sdk1.4.2_06\bin>java HelloWorld
```

java HelloWorld

تطبيق Hello World سيعمل الآن . وتظهر الرسالة على الشاشة كما بالشكل



```
C:\WINDOWS\System32\cmd.exe
C:\j2sdk1.4.2_06\bin>javac HelloWorld.java
C:\j2sdk1.4.2_06\bin>java HelloWorld
Hello, World!
C:\j2sdk1.4.2_06\bin>
```

ملفات أنواع Java صغيرة. في هذه الحالة، مصدر الكود لهذا كان 151 byte (بايت وليس كليوبات) مما يعادل 151 حرف من النص (بعضها مسافات فارغة تستخدم في التنسيق لتخطيط ملف مصدر الكود) بينما نوع أو صنف المخرج النهائي هو فقط 427 bytes. هذا الكود الصغير يجعل كود جافا أسرع بكثير ويجعله مثالياً للاستخدام عبر الإنترنت. حيث أن حجم الملف الذي تحتاج أن ترسله لشخص ما يشكل عاملاً حاسماً.

مع ذلك، لا بد من أن النسخ الاحتياطي وجعل عمل كل ما هو وراء الكواليس كله من آلة جافا الافتراضية (JVM). التي في نفسها ليست بالصغيرة (متطلبات النظام لجافا تحت Windows يستدعي لـ 125 MB من المساحة الفارغة لتحميل JVM، والحقبة النهائية في نفسها تستخدم ما يقرب من 10 MB)

إذا ألقيت نظرة على المخرجات من مفسر الجافا. (انظر الشكل). ستجد أن مخرج CodeByte الفعلي عندما تنظر إلى تنسيق النص هو بشكل مفاجئ مشابهة إلى الكود الأصلي. يمكنك جعل المخرج قيم حرفية، إسماء ملف المصدر، والأوامر خلال الكود.

```

UltraEdit-32 - [HelloWorld.class]
File Edit Search Project View Format Column Macro Advanced Window Help
HelloWorld.class
0 1 2 3 4 5 6 7 8 9 a b c d e f
00000000h: 5A FE BA BE 00 00 00 2E 00 1D 0A 00 06 00 0F 09 ; Ep*%.....
00000010h: 00 10 00 11 08 00 12 0A 00 13 00 14 07 00 15 07 ; .....
00000020h: 00 16 01 00 06 3C 69 6E 69 74 3E 01 00 03 28 29 ; .....<init>...()
00000030h: 56 01 00 04 43 6F 64 65 01 00 0F 4C 69 6E 65 4E ; V...Code...LineN
00000040h: 75 6D 62 65 72 54 61 62 6C 65 01 00 04 6D 61 69 ; umberTable...mai
00000050h: 6E 01 00 16 28 5B 4C 6A 61 76 61 2F 6C 61 6E 67 ; n...{Ljava/lang
00000060h: 2F 53 74 72 69 6E 67 3B 29 56 01 00 0A 53 6F 75 ; /String;)V...Sou
00000070h: 72 63 65 46 69 6C 65 01 00 0F 48 65 6C 6C 6F 57 ; rceFile...HelloW
00000080h: 6F 72 6C 64 2E 6A 61 76 61 0C 00 07 00 08 07 00 ; orld.java.....
00000090h: 17 0C 00 18 00 19 01 00 0D 48 65 6C 6C 6F 2C 20 ; .....Hello,
000000a0h: 57 6F 72 6C 64 21 07 00 1A 0C 00 1B 00 1C 01 00 ; World!.....
000000b0h: 0A 48 65 6C 6C 6F 57 6F 72 6C 64 01 00 10 6A 61 ; .HelloWorld...ja
000000c0h: 76 61 2F 6C 61 6E 67 2F 4F 62 6A 65 63 74 01 00 ; va/lang/Object..
000000d0h: 10 6A 61 76 61 2F 6C 61 6E 67 2F 53 79 73 74 65 ; .java/lang/Syste
000000e0h: 6D 01 00 03 6F 75 74 01 00 15 4C 6A 61 76 61 2F ; m...out...Ljava/
000000f0h: 69 6F 2F 50 72 69 6E 74 53 74 72 65 61 6D 3B 01 ; io/PrintStream;.
00000100h: 00 13 6A 61 76 61 2F 69 6F 2F 50 72 69 6E 74 53 ; ..java/io/PrintS
00000110h: 74 72 65 61 6D 01 00 07 70 72 69 6E 74 6C 6E 01 ; tream...println.
00000120h: 00 15 28 4C 6A 61 76 61 2F 6C 61 6E 67 2F 53 74 ; ..(Ljava/lang/St
00000130h: 72 69 6E 67 3B 29 56 00 21 00 05 00 06 00 00 00 ; ring;)V.!.....
00000140h: 00 00 02 00 01 00 07 00 08 00 01 00 09 00 00 00 ; .....
00000150h: 1D 00 01 00 01 00 00 00 05 2A B7 00 01 B1 00 00 ; .....*...±..
00000160h: 00 01 00 0A 00 00 06 00 01 00 00 00 01 00 09 ; .....
00000170h: 00 0B 00 0C 00 01 00 09 00 00 00 25 00 02 00 01 ; .....%.....
00000180h: 00 00 00 09 B2 00 02 12 03 B6 00 04 B1 00 00 00 ; ....*...%...±...
00000190h: 01 00 0A 00 00 00 0A 00 02 00 00 00 05 00 08 00 ; .....
000001a0h: 06 00 01 00 0D 00 00 00 02 00 0E ; .....

```

For Help, press F1 Pos: 0H, 0, C0 DOS Mod: 12/8/2004 09:50:40 File Size: 427 INS

Benefits of Compiling

منافع التفسير

لقد قضينا بعض الوقت فى البحث خلال عملية التفسير وكيف أن المفسرات المختلفة تعالج مصدر الكود لإنشاء الملف التنفيذي .ولكن لماذا تهتم لترجمة التعليمات البرمجية فى المقام الأول ؟ لماذا لا يتم معالجة كل الكود كنص عادى وقت التشغيل كما تكون الكتابات ؟

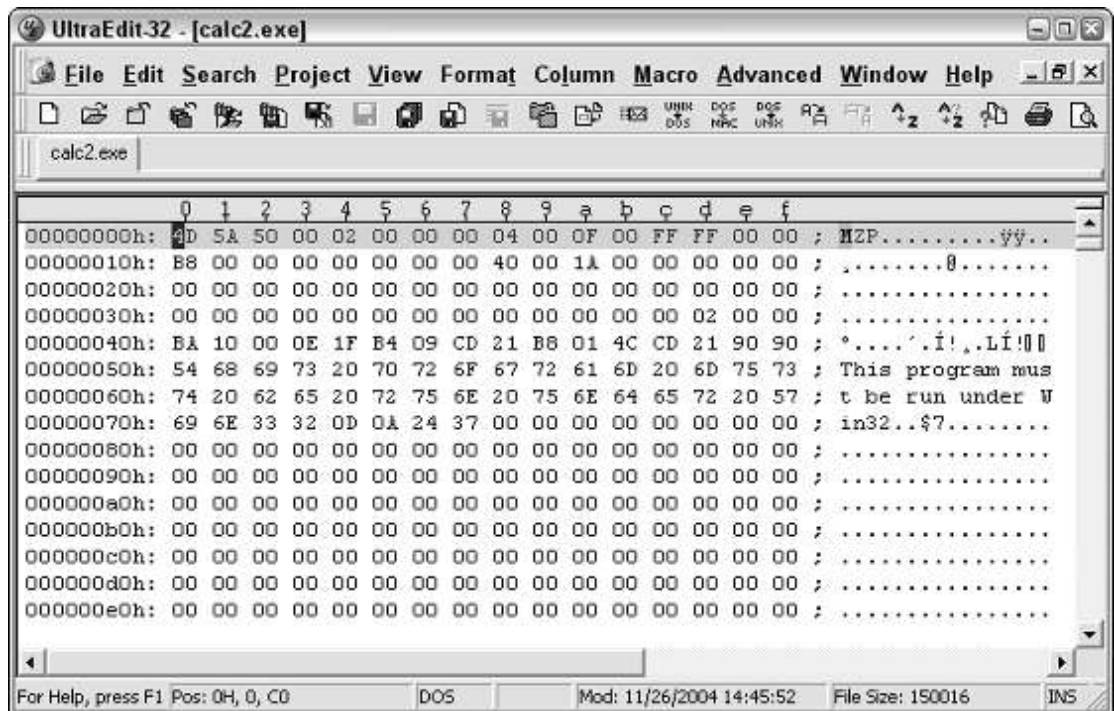
دعنا نلقى نظرة على لماذا التفسير أو الترجمة هامة .

Protection of Intellectual Property حماية الملكية الفكرية

احد الاختلافات الرئيسية بين الكود المفسر والكود المستخدم فى لغة البرمجة الكتابية هو أن الكود المكتوب يخزن كنص عادى . للمبرمج ليست مشكلة , فى الحقيقة , يمكن ان تكون ميزة لإن الملف الذى تقوم بتشغيله والملف الذى تضع فيه مصدر الكود هما نفس الملف . فإنت تعمل مع ملف واحد ,وعندما تاتى لإختبار الكود فليس هناك مرحلة تفسير/ ترجمة ليمر بها .

مع ذلك , الجانب الآخر لهذه الحجة ان ذلك الكود الذى يراه كل الناس هو الكود الذى قمت بكتابته. طالما أن الشخص يفهم اللغة . لا شئ يوقف هذا الشخص من نسخه , تغيير , أخذ أفضل الأجزاء , ودمجه مع مشروعه /مشروعها .

تفسير الكود يقدم لك , كمبرمج , الأمان من العيون المتطفلة .والقدرة على النسخ واللصق السريع للآخرين . تفسير الكود يقوم بتحويل الكود من النص العادى إلى كود الآلة (انظر الشكل) مما يعنى أنه قد تم حمايته من أن يقرأ أو يفهم بواسطة الآخرين . وذلك لأن عملية التفسير هى إتجاه واحد ولا يمكن التراجع (أو هندسة العكس) . هذا يعنى أنه بدون مصدر الكود فتطبيقك لا يمكن ان يأخذ للوراء إلى الكود الذى تم كتابته به . وهذا يوقف الآخرين من تغيير الكود وسرقة أفكارك .



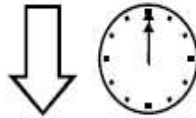
Speed السرعة

سبب آخر لماذا التفسير (أو على نحو أدق , إستخدام لغة تحتاج للتفسير) هام فى أن الكود المفسر Compiled Code يتم تشغيله بواسطة الحاسب أسرع من الكود المترجم Interpreted Code.

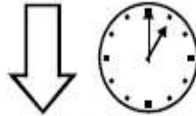
والسبب الرئيسي وراء هذا التحسن في الأداء هو ان Compiled code أو الكود المفسر قد تم تحويله بالفعل إلى كود الآلة قبل الحاجة إليه , مما يعنى أن وقتاً ثميناً قد تم حفظه ويمكن أن يتم تحميل التطبيقات على الفور .

سبب آخر فى أن الكود المفسر Compiled Code أسرع من نظيره الكود المترجم Interpreted Code هو أن قبل يتم تشغيل الكود المترجم فإنه يتم تشغيل برنامج المترجم Interpreter وقد تم تحميله فى الذاكرة , مرة أخرى يأخذ الوقت , كما فى الشكل التالى

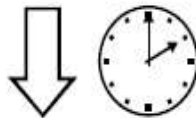
User runs script



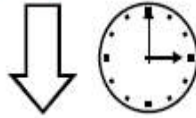
Operating system
loads interpreter



Interpreter checks
script for errors



Script processed



Script finally
loaded!



منافع أخرى للسرعة تأتي من حقيقة أن المفسر compiler يمكن أن يحسن الكود لأنظمة تشغيل محددة , معالجات , أو حاسبات مقدر أن يعمل الكود عليها . يمكن ان يوفر هذا التسحين تحسينات كبيرة في الأداء . ولكن هذه المزايا هي عادةً ما تعرض في المفسرات التجارية (وبالتالي يدفع لها)

Increased Functionality

زيادة وظائف

لإن معظم الكود المترجم هو كود برنامج نصي , الذي صمم في بداية الأمر ليستخدم في عالم الإنترنت من خلال مستعرض الإنترنت Web Browser . وظيفة هذه اللغات قليل مقارنة بتلك اللغات الكاملة مثل C++ . فهناك العديد من الوظائف محدودة أو مفقودة في مثل هذا الكود :

- القدرة على إنشاء واجهة متقدمة .
- معالجة ملفات .
- الوصول إلى الأجهزة الطرفية . [يقصد الأجهزة المتصلة بالحاسب]
- القدرة على الاندماج مع أنظمة التشغيل .
- معالجة متقدمة لموارد المعالجات والذاكرة .
- معالجة متقدمة للصور .

Security

الأمن

لإن الناس لا يملكون القدرة على تغيير كود البرنامج المفسر بشكل مباشر , فهناك مجال أقل للذين يريدون أن يأتوا إلى الكود ويقومون بتغييره . لماذا يريد شخص ما تغيير الكود ؟ هناك أسباب متنوعة هذه الأيام :

- سرقة معلومات .
- تسبب في التدمير .
- سرقة موارد النظام (مثال نطاق تردد الإنترنت)

النص العادي , الكود الغير مفسر عرضة للإنتهاكات او شئ من هذا القبيل و لا ينبغي أن يستخدم في تطبيقات لديها صلاحيات دخول كبيرة إلى النظام .

Debugging

التصحيح

سبب عظيم آخر لتفسير الكود هو أن عملية التفسير تجد المزيد من الشوائب والأخطاء في الكود . مع الكود المترجم Interpreter Code تحتاج ان تقوم بتشغيل الكود لتقوم بفحصه , ولكن هذا ليس تصحيحاً دقيقاً كالذي تم تنفيذه من قبل Compiler .

Alternatives to Compiled Code

بدائل الكود المفسر

افترض أنك تريد ان تستخدم لغة كتابات نصية لأداء وظيفة ما ولكن تريد حماية اكثر قليلاً من الآخرين الذين يريدون التطفل على الكود الخاص بك وسرقة أفكارك (او حتى تغيير الكود) . إليك بعض الحيل التي يمكنك من جعل الكود أقل ودأً وتعاطفاً مع الآخرين .

Make Code Hard to Follow

إجعل الكود صعب التتبع

لماذا تجعل الكود الخاص بك سهل القراءة للآخرين ؟ تبدو نتيجة عكسية إذا كنت لا تريد لن يتم قراءة الكود الخاص بك بواسطة الآخرين .

إليك بعض النصائح التي تجعل الكود يصعب على الآخرين قراءته .

Obscure Variable Names

غموض تسمية المتغيرات

خلال التطوير يمكنك أن تستخدم أسماء من السهل تتبعها لجعل الكود أسهل , ولكن سيجعل الأمور أسهل للآخرين ليقروا الكود . قبل أن تطلق الكود الى التداول في الأسواق , يمكنك أن تغير المتغيرات التي استخدمتها ليكون أكثر غموضاً وأصعب للتتبع بواسطة الآخرين . تقدم ميزة البحث والإستبدال في برامج محررات النصوص فهي نغالية لهذه العملية .

حتى التغيرات البسيطة للكود تجعله صعب التتبع , قارن .

```
Dim subtotal, salestax, ordertotal
salestax = 5
subtotal = 199
ordertotal = ((subtotal / 100) * salestax) + subtotal
MsgBox("Order total is: $" & ordertotal)
```

و هذا

```
Dim xxyyxx, chhhcch, yyffyyfy
chhhcch = 5
xxyyxx = 199
yyffyyfy = ((xxyyxx / 100) * chhhcch) + xxyyxx
MsgBox("Order total is: $" & yyffyyfy)
```

تأكد أن ق قمت بحفظ نسخة من الكود الأصلي في حالة أنك تريد أن تقوم ببعض التغيرات على الكود أو تبديل في المستقبل — لاجدوى من جعل الأمور صعبه عليك .

Whitespace المسافات الفارغة

المسافات الفارغة في الكود تجعله أسهل للتتبع لأن التخطيط يضيف هيكل مرئي لمعنى الكود . بعض التعديلات والتبديلات في هذه المسافات البيضاء أو الفارغة يمكن ان يجعل الكود أصعب للقراءة .

مع ذلك , حذف المسافات الفارغة التي بداخل الكود ليس ذلك هو الشيء الفعال الذي يمكنك فعله (لإننا إعتدنا أن نقرأ من اليسار لليمين أى من الحافة اليسرى للصفحة). في الحقيقة , الشيء الأكثر فعالية هو إضافة مسافات زيادة إلى الكود . قارن :

```
Dim subtotal, salestax, ordertotal
salestax = 5
subtotal = 199
ordertotal = ((subtotal / 100) * salestax) + subtotal
MsgBox("Order total is: $" & ordertotal)
```

مع :

```
Dim subtotal, salestax, ordertotal
salestax = 5
        subtotal = 199

ordertotal = ((subtotal / 100) * salestax) + subtotal
MsgBox("Order total is: $" & ordertotal)
```

إرفق هذه التقنية مع الكود السابق وستحصل على طريقة فعالة لحماية الكود :

```
Dim xxyyxx, chhhcch, yyffyyfy
        chhhcch = 5
xxyyxx = 199

yyffyyfy = ((xxyyxx / 100) * chhhcch) + xxyyxx
MsgBox("Order total is: $" & yyffyyfy)
```

هذا الكو يضيف زيادات في تحميل الكود , لأن كل حرف , حتى المسافات الفارغة تعنى وجود Byte زيادة في الكود . ولكن مادامت لن تفرط في استخدام هذه الطريقة . فلا ينبغي أن يضيف عدد كبير من Byte إلى الكود (1.000 مسافة فارغة تساوى 1 kb) .

Script Encoding

الشئ العظيم فى استخدام نظام التشغيل Microsoft Windows هو أن هناك المزيد من الأدوات الرائعة التى يمكنك أن تقوم بتحميلها وتنزيلها واستخدامها من موقع مايكروسوفت . أحد هذه الأدوات هى Microsoft Script Encoder .

هذه التطبيقات تأخذ كود مكتوب باستخدام VBScript أو Jscript (إصدار مايكروسوفت من JavaScript) . ويقوم بترميزه لتجعل من المستحيل إلى حد كبير على شخص آخر أن يقرأهم . ولكن لا يزال مفهوم بالنسبة لنظام التشغيل windows .

التطبيقات هى كتابة على شكل رموز , ترميز الكتابة فى مقابل التشفير . التشفير مختلف عن الترميز . ولكن الاختلاف خفى . مع الإثنين تقوم بإخذ الملف ثم تقو بتغيير تنسيقه . ولكن مع التشفير فالملف غير مستخدم إلا بعد أن يتم فك التشفير عنه . الترميز يغير الملف من وجهة نظر القارئ من البشر , ولكن يظل الحاسب يفهمه ويعرف ماذا يفعل هذا الكود الذى داخل الملف . إذا تم تشفير الملف , لن يكون من المتاح الدخول إلى المحتويات من خلال الأشخاص أو التطبيقات بدون كلمة سر أو كلمة مرور .

ملف تثبيت Microsoft Script Encoder متاح مجاناً من مايكروسوفت كملف تحميل صغير (تقريباً 130 kb) من الموقع التالى : www.microsoft.com/scripting

وهو متاح باللغات التالية :

- الإنجليزية .
- الصينية (التقليدية)
- الألمانية .
- الكورية .

وتثبيته سهل . حدد الملف المراد تحميله (الإصدار باللغة الإنجليزية يسمى sce10en.exe , بينما الإصدار باللغة الصينية sce10cht.exe , والإصدار بالألمانية هو sce10de.exe , و الإصدار بالكورية , sce10ko.exe) قم بالضغط مرتين عليه . ثم إتبع الخطوات .

الموقع الافتراضى للتثبيت لتطبيق Command-Line وملفات المساعدة هى
C:\Program files\Windows Script Encoder.

إضافة إلى تثبيت تطبيق سطر الأوامر Command-Line وملف المساعدة , فتنصيب التطبيق يقوم بتعديل Windows Registry لذا فإن النظام يقوم بتعريف ملفات .vbe و .jse . (ملفات الكتابات ذات الترميز VBScript , ملفات الكتابات ذات الترميز Jscript , على التوالى) .

تثبيت Microsoft Script Encoder لا يتطلب تشغيل ملفات الكتابات المرمزة . فقط يحتاج المبرمج أن يثبت المرمز Encoder.

[المرمز : نقصد به تحويل النص العادى إلى رموز]

Microsoft Script Encoder أو مرمز الكتابات التابع لمايكروسوفت هو مرفق يتم جلبه من الخارج ولا يأتى مع واجهة الـ windows . وهذا يعنى أنه ليس سهلاً فى استخدامه كمثّل استخدام التطبيقات ذات الواجهة الرسومية . مع ذلك . ليس من الصعب استخدامها .

وهي لا تقوم بالترميز بنفسها ولكن بدلاً من ذلك تستخدم ملف وحدة كتابات وقت التشغيل scripting runtime module file (يسمى Sccrun.dll) موجودة بالفعل في windows لتقوم بالترميز من أجلك . هذا هو الملف الذي يترجم Interpreter ويقوم بالفعل بتشغيل Javascript , VBScript . وكل ما تفعله الكتابات المرمز المنفذة هو تزويد Comman-Line آلية لتنفيذ الترميز.

لأن برنامج التثبيت لا يضيف المسار إلى المرمز إلى متغير بيئة النظام Path . وهو ليس متاح من كل مجلد من نافذة Command Prompt . لتتأكد من أن استخدام مرمز الكتابات التابع لمايكروسوفت يعد سلساً وخالياً من المتاعب . نوضح أن تفعل أحد الأشياء التالية :

- إضافة المسار لرمز الكتابات (المسار الافتراضي هو C:\Program Files\Windows Script Encoder) إلى متغير النظام Path (استشر دليل النظام الخاص بك أو ملفات المساعدة لتعرف كيف تفعل هذا)
 - نسخ الملف التنفيذي لرمز الكتابات (screnc.exe) للمجلد الذي ستستخدمه للكتابات التي سيتم ترميزها .
 - قم بنسخ كل الكتابات النصية التي تريد أن تقوم بترميزها إلى المجلد المستخدم بواسطة مرمز الكتابات التابع لمايكروسوفت .
- للملحوظ والبساطة , ستقوم بنسخ كل الكتابات النصية التي ستقوم بترميزها إلى المجلد المستخدم بواسطة مرمز الكتابات . هذه أسهل طريقة لاستخدام المرمز .

أفضل طريقة لتعريف نفسك بقواعد بناء الجمل في Microsoft Script Encoder هو استخدامه . قم بفتح نافذة Command Prompt وإذهب إلى المسار C:\Program Files\Windows Script Encoder . وهناك اكتب التالي :

```
screnc /?
```

سيجعل التطبيق يدرج لك المساعدة . يمكن أن يكون هذا مرجع مفيد . عندما تكون في عجلة من أمرك . تعرض المخرجات في الشكل التالي :

```
C:\WINDOWS\System32\cmd.exe

C:\Program Files\Windows Script Encoder>screnc /?

Usage:   screnc [/?] [/s] [/f] [/xl] [/l ScriptLanguage] [/e DefaultExtension]
         <source> <destination>

Encode embedded script.

/? -      Help
/s -      Silent: display no messages
/f -      Force: allow file(s) overwrite <source == destination>
/xl -     Exclude Language: does not add the language directive in asp files
/l ScriptLanguage -
         Script Default Language: specify the default script language to be
         used when encoding
/e DefaultExtension -
         Default Extension: override actual file extension. Control the
         encoder to be loaded.
<source>
         The file to encode. It can have wildcard characters.
<destination>
         The destination file. When <source> contains wildcard characters,
         <destination> is the directory where to place the encoded
         files; files will keep the same name. When <source> and
         <destination> are the same /f must be used.

Examples:
screnc test.html encode.html
         encode test.html into encode.html
screnc /f test.html
         encode and override test.html
screnc /e html test.txt test1.txt
         treat test.txt as an html file, encode it into test1.txt
screnc test.asp c:\myDir\test.asp
         encode test.asp into c:\myDir\test.asp
screnc *.asp c:\myDir
         encode all the asp files in the current directory and place
         then in c:\myDir
screnc -s -f *.sct
         encode all the scriptlet files in the current directory and
         override them. No message is displayed
screnc -e asp *.* c:\myDir
         encode *all* the files in the current directory as asp files.
         Place these files in c:\myDir
screnc -e asp -xl *.inc c:\myDir
         encode all the files with .inc extension as asp files.
         Does not add the @LANGUAGE directive at the top of the file
screnc -l vbscript test.html encode.html
         encode test.html into encode.html. When a script tag with
         no language attribute is found, VBScript is assumed to be
         the default language. If -l is not specified, JScript is
         assumed for html and VBScript is the default for asp

C:\Program Files\Windows Script Encoder>
```

قاعدة بناء الجملة الأساسية لـ Microsoft Script Encoder هي :

```
screnc inputfile outputfile
```

- **Inputfile** . مطلوب . وهذا هو اسم الملف الذي سيتم ترميزه, ويمكن ان يحتوى على أى معلومات ضرورية عن المسار لها علاقة بالدليل الحالى .
- **Outputfile** . مطلوب . وهذا هو اسم الملف الذى سينتج عن العملية . ويمكن أن يشتمل على أى معلومات عن المسار لها علاقة بالدليل الحالى .

ما يلى هي امثلة بسيطة على استخدام Microsoft Script Encoder . كلاً منهم مصاحب بشرح مختصر للنتائج .

لترميز الملف المدخل unencoded.htm وإنتاج ملف مخرج باسم encoded.htm, استخدم :

```
screnc unencoded.vbs encoded.vbe
```


لذا , على سبيل المثال , دعنا نقول أن لدينا ملف VBScript يحتوى على الكود التالى :

```
' Adrian Kingsley-Hughes
' 02 Dec 2004
' This script prompts the user for his or her name.
' It incorporates various greetings depending on input by the user.
'
' Added alternative greeting
' Changed variable names to make them more readable

Dim PartingGreeting
Dim VisitorName

VisitorName = PromptUserName

If VisitorName <> "" Then
    PartingGreeting = "Hello, " & VisitorName & ". Nice to have met you."

Else
    PartingGreeting = "I'm glad to have met you, but I wish I knew your name."
End If

MsgBox PartingGreeting

Function PromptUserName

    ' This Function prompts the user for his or her name.
    ' It incorporates various greetings depending on input by the user.
    Dim YourName
    Dim Greeting

    YourName = InputBox("Hello! What is your name?")

    If YourName = "" Then
        Greeting = "OK. You don't want to tell me your name."
    Else
        Greeting = "Hello, " & YourName & ", great to meet you."
    End If

    MsgBox Greeting

    PromptUserName = YourName

End Function
```

إذا قمنا بحفظ هذا الملف بإمتداد كـ unencoded.vbs وقمنا بتشغيل الأمر المعطى بالأعلى لترميزه داخل ملف يسمى encoded.vbs . فهذا الملف سيحتوى على التالى :

يمكنك ان تتحكم في اى اجزاء يتم ترميزه باستخدام علامات Script من خلال الكود :

لاحظ أن هذه العلامات تبدأ بعلامات التعليق في VBScript — لا تتركها تفوتك !

هذا يمكنك ان تقوم بترميز جزء فقط للكوود :

```

**Start
Encode**#@~^IAMAAA==@##@qW,./b/kDWMHls+~@!@*PEE,KtOU@##@PP,~KlMYbxLM.nOYrXT-',Jun^VW
S~rP'PjrkkoKdGl:UPL~JcPhKl+~OKPt~
+,~d~zW!Rr@#@P@#@#@#@3Vkn@#@P~,nl.ObxLMMnOYro,'PrqEhPTVCN,YG~4l\Q~:OYPHG;~,4lY
~q~Ab/4P~3+A~HWE,~lh+cE@#@#@3N,q0###@#@#@Hko$WXPkCMYkLMM++DrUo@#@#@#@#@#@#@#@wEUMd
rWPk,K:wo!k+.lmhQ#@#@#@#@#@,PP,B,Ptb/~s!x^ObWx,2DK:wdd~Y4+,Ed+~6W.PD4~bD~Um:~
@#@#@P~P,v,q0,kmWmwK.Ld+dP7l.rKE/,LDQ+YbUL/,NDwnx[ro~W~kw;O,4X~O4+~EknMR@#@@P,PPGks~
5KE.lm:n~#@#@P,~PGk;!,~OYbxL@#@#@#@#@P~P,eW!DHCs+Px~&x2ED$K6crCUVVK*,~Q4lOpb/~zKED,
Uls+grb@#@#@#@#@,P~P(W,5GEMHls+~x,JJ~P4+U@#@#@~,P~,P,PMMAoOkO~^,J6FcPPtGE,NwOPs1Y~YG
~D+sV,h+,XG;MPxChDR@#@#@~,P~AVk+@#@#@P,~P,P~PVDnnDkxT~',JCDssWBPrP'PeG!DHlsnPLPES,oD
ncDPow,hD+O,XKERr@#@#@P,P3x9P(W@#@#@#@#@P,PhkL$WXPVDn+Or
@#@#@#@#@#@P,P~KMW:2O~/nDgCs+~(PIWEMlmh~@#@#@#@#@Ax[~wExlOkKxyNoAAA==^#~@

```

يسمح لك هذا بعمل شيئاً شيقاً — أضف ملاحظة بحقوق النسخ حيث لا يستطيع المستخدم أن يقوم بتغييرها لأنهم لو قاموا بتغييرها عندها سيلتقطها الكود .

```

Dim strCopyright
strCopyright = "This script is copyright to me, 2004!"
'***Start Encode**
If strCopyright = "This script is copyright to me, 2004!" Then
    MsgBox "Copyright notice is unaltered. Script cleared to continue ..."
    MsgBox "Script continues ..."
Else
    MsgBox "Copyright notice is altered!!! Script halted!"
End If

```

سطر الكود الذى يفحص ملاحظة حقوق النسخ وسوف يتم تشفيرها دون تغيير ويفحصها عند تشغيل الكود , ويمكن أن يلتقط التغير الذى تم هنا — فهى طريقة سهلة لتتأكد من أن ملاحظات حقوق النسخ لن تتغير .

Encoded Scripts—Dos and Don'ts

قبل وبعد الترميز هناك بعض إفعال ولا تفعل ينبغى أن تتبعهم لتتأكد من أن الكود خالى من المتاعب بعد الترميز .

إفعل

- إلفظ النص العادى , غير مرمز , كنسخة إلفطاففة للكوځ .
- كن منلفبها أنك ففففف إفففف .vbe .لفففف VBScript المرمزة و إفففف .jse .لفففف
- Jscript المرمزة .
- قم بفففف الفففففف قبل الففففف — بفف الففففف لا فاففة لنا فففف وفففف ان فففف فى كلفة نص الكوځ .
- إفففف الكوځ قبل و بفف الففففف لففففف من انه فففف .

لا تفعل

- لا تفعل أى ففففف فى الكوځ بففما ففف ففففف .
- لا ففففف الإفففف والففففف ففف ففففف نص الكوځ .
- لا ففففف أن ففففف ففففف الكوځ ماففة بالماففة أمان . ففففف ففففف لففففف الففففف للكوځ (والففى لم أقم بفففففها هنا) .

Summary

الملفص

فى ففذا الفففف , لفځ رأفنا منافف كففرة لففلفة ففففف الكوځ وكفف أن المفففراف المففلفة فففف أن ففففا ملفاف مفرفة مففلفة لففف مفرر الكوځ . على الرفف من أن لغة البرمفة المففففة واففة . وأففا رأفنا كفف أن المفففراف Compilers أسرع من فففرها . وكفف ان برنامففن للففففف ففففف باففراف ملفاف ففففففة مففلفة باففام مففلفة .

ولفځ فافففة أن لغاف البرمفة المففلفة ففففف ملفاف ففففففة مففلفة — البفف منفف مفففف بفففه , بففما ففففف الأفرفن مفففباف مفاةة ففف ففففف لكى فففف .

لفځ ألقىف ففطرة على الففف فففف الكوځ فففف الفففف من قبل مفففففف الأكواځ . كما أننا بففففا فى فففف إففففاف Microsoft Script Encoder لففففف الكوځ لفففففك لففففف من لاففن فففففون أن ففففففوا الكوځ الفففف بك .

16

Distributing Your Project

نشر وتوزيع المشروع

لقد كان طريقاً طويلاً , ولكن أخيراً قد أكملناه . وقد صممت المشروع , خطة العمل قد وضعت , وكتب الكود , واختبر , وفسر , وتم تصحيحه , وإصلاحه , وإعادة تفسيره . فأصبح لديك المشروع.

ولكن ماذا بعد ؟ ماذا تفعل الان ؟

هذا الفصل يفحص الاختيارات المتاحة ل لتوزيع ونشر المشروع للآخرين فيمكن للآخرين أن يستنفعوا ويستفيدوا بما قد قمت بفعله .

إليك ملاحظة أخيرة عن حقوق توزيع التطبيق الخاص بك , تذكر ان تفحص إتفاقية الترخيص لأي مفسر تستخدمه لتتأكد من ما هية الحقوق عندما تأتي لمرحلة التوزيع والنشر . بعض إتفاقية التراخيص ربما تسمح لك بتوزيع ونشر الكود بدون قيود , بينما الآخرين يحظر الإصدار "التجاري" (هذا هو , فهو يمنعك من أن يسرق ما قمت بفعله) .

إفحصه وإبقى قانوني !

Types of Distribution

أنواع التوزيع

هذا هو الوقت في التفكير في توزيع و نشر مشروعك . الإختيار الرئيسي الذي يجب أن تقوم به هو أن تقرر الطريق الأفضل ليقع المشروع الخاص بك في أيدي المهتمين به .

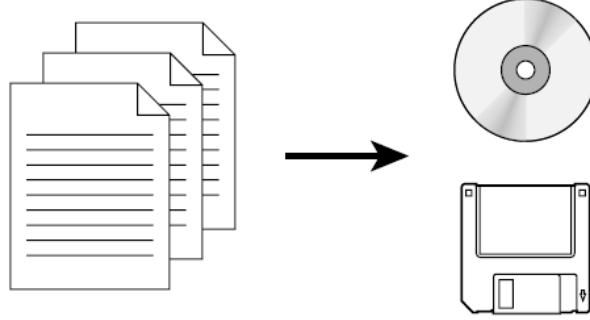
هناك نوعين من التوزيع الواقعي متاحين لك , وكلاهما يناقش بالتفصيل في المقاطع التالية :

- ☐ Physical distribution (التوزيع المادي)
- ☐ Virtual distribution (التوزيع الوهمي أو الظاهري)

Physical Distribution

التوزيع الطبيعي

التوزيع الطبيعي كان , حتى مؤخراً , هو الطريقة الأساسية في توزيع كل البرامج . التوزيع الطبيعي يشمل على نسخ المشروع على وسائل نقل بيانات من نوع أو من آخر وتوزيع وسيلة نقل البيانات هذه إلى من يهتم فيها . (كما هو مبني بالشكل التالي)



هناك عدد من وسائل التخزين المختلفة يمكن ان تستخدمها بسهولة في نشر وتوزيع مشروعك :

- ☐ Floppy disk
- ☐ CD
- ☐ DVD

كل نوع من وسائل تخزين هذه لديه عيوب ومميزات .

Floppy Disk

يظهر في الشكل التالي , يستخدم كوسيلة أساسية لتوزيع المشروع .

هناك عدد من العيوب والمميزات في استخدام Floppy Disk .

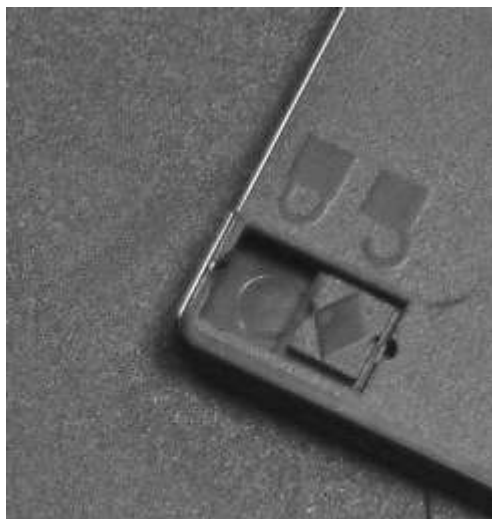
Advantages

المميزات

- **منخفض الثمن** . فتلكلته زهيدة وواسع الانتشار .
- **متوفر على نطاق واسع** . هي أقراص تقوم بشراءها من متاجر الحاسب . المكتبات , وحتى الكثير من المتاجر العامة . ويمكن شراء وبكميات كبيرة وواسعة.
- **إعادة الاستخدام** . يمكن ان يعاد استخدامها وبسهولة . ولكن يمكن ان يقلل ليمنع من الكتابة عليه مرة أخرى من خلال استخدام الجزء الخاص Lock (مبين في الشكل التالي) .
- **سهل التشخيص** أي تحديد هوية . يمكن أن تشخصها بسهولة بطابعات مرفقه مع هذه الأقراص . هذه البطاقات يمكن ان شرائها بكميات كبيرة ويمكن طباعتهم على معظم طابعات المنزل .
- **مظم الحاسب لديها مشغل Floppy** . فهي شائعة في معظم أنظمة الحاسب (المشغل العادة الذي يظهر في الشكل) .



Floppy Disk
قرص التخزين



LOCK
قفل



Floppy Drive
المشغل

Disadvantages

العيوب

- **المساحة** . فقيرة نسبياً — 1.44 MB . في العام يعني هذا أن كل ولكن الأبسط والأصغر من المشاريع يتطلب أكثر من Floppy واحد ليحمل عليه البيانات . المساحة الضئيلة تعني ان المستخدم النهائي سيحوز على أقراص متعددة .
- **بطئ النسخ عليه** . يمكن ان يأخذ من دقيقتين إلى ثلاث دقائق لكي يملئ بالبيانات . فإذا كان يحتاج إلى تهيئة قبل أن تستخدمه , إذاً الوقت المستغرق لإنشاء قرص واحد يمكن ان يكون أكثر من خمس دقائق . هذه هي 5 دقائق لمدة تقل عن 11 / 2 MB !
- **بطئ في القراءة** . قرانته ونسخ البيانات منهم هي عملية بطيئة جداً . تتطلب الكثير من الدقائق .
- **لا يمكن الاعتماد عليها** . فهي عرضة للتلف من الأوساخ والأتربة , والغبار , الحرارة , المجالات المغناطيسية , حوادث المستخدم . والكثير .
- **تغطية الكتابة** . يمكن أن تستبدل الكتابة عليه بواسطة المستخدم النهائي بدون قصد . ويمكن أن يصاب الفيروسات ويدمر .
- **مشغلات هذه الأقراص** . بينما لدى كل الأنظمة floppy Drive , والبعض ليس لديه . والأجهزة التي بدون هذا المشغل (خصوصاً الحاسبات المحمولة) قد أصبحت كثيرة وأكثر شيوعاً .

CDs

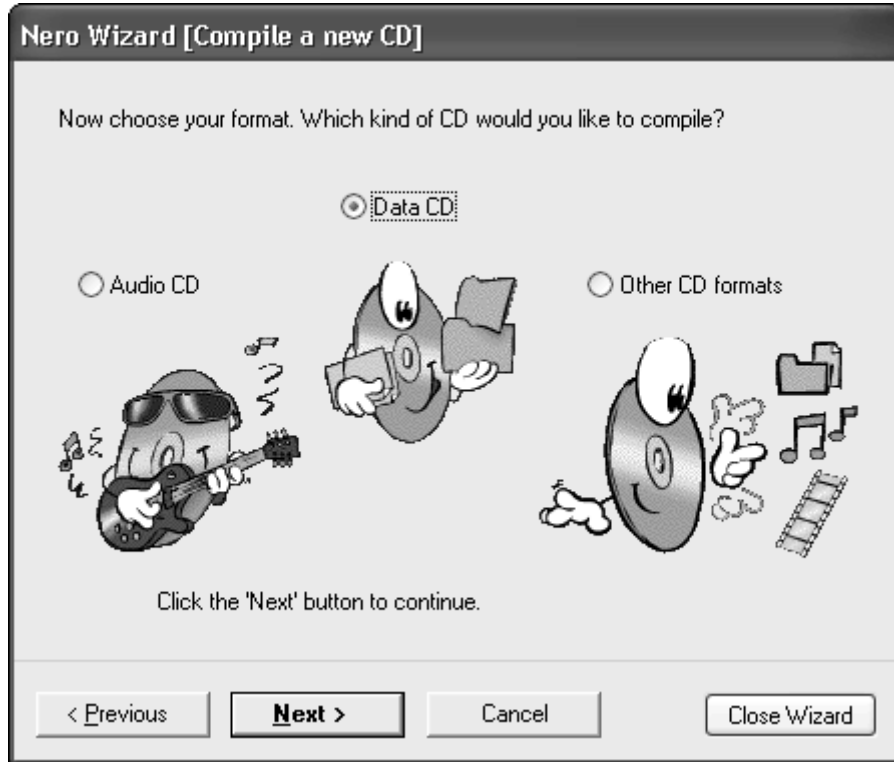
الأقراص المدمجة

من أكثر من عشر سنوات , إستبدلت الأقراص المدمجة CDs (Compact Disks) الأقراص العادية Floppy Disks لمعظم أغراض توزيعات البرمجيات . (في الشكل)



Advantages المميزات

- **المساحة** . له مساحات متعددة . تبدأ من 700 . وهذا معادل لأكثر من 480 قرص من نوع Floppy .
- **منخفض الثمن** . تكلف القليل من البنسات Pennies ومتوفر في كميات كبيرة .
- **يمكن الاعتماد عليه** . قوية ويمكن أن تبقى على قيد الحياة في الظروف القاسية وعلى المدى الطويل .
- **إنتشار محركات أو مشغلات الأقراص** . الكثير من الحاسبات لديه هذا المشغل . مما يعطى توزيع لها على نطاق واسع .
- **وضع علامات أو بطاقات** . هناك العديد من النظمة التي تدعك تشخص وتضع علامات على هذه الأقراص بسهولة بإستخدام طابعات المنزل .
- **إستبدال الكتابة** . لا يمكن ألها أن تستبدل بواسطة المستخدم النهائي , مما يجعلها بمأمن من التلف بواسطة الإستبدال للبيانات او بالفيروسات المدمرة .
- **سهل الإنشاء** . سريعة , سهلة , منخفض الثمن لإنشاءه , مع سهولة إتاحة مشغل النسخ عليه . حرق البرمجيات عليه , أو أقراص تقبل الكتابة , الشكل التالى يوضح نافذة الإنشاء لتسجيل برنامج على CD/DVD من خلال برنامج Nero (لمزيد من المعلومات www.nero.com)



- **سرعة الإنشاء** . يمكن أن يحرق القرص فى مشغلات حرق حديثة والتكرار والمضاعفة سريعة وكذلك سهلة .
- **سريع بالنسبة للمستخدم النهائي** . سريع ليقرأ , ويمكن أن تنتسخ منه البيانات بشكل سريع أيضاً .

Disadvantages

العيوب

- **المساحة** . كما يمكن للمساحة ان تكون ميزة , يمكن أن تكون عيب — كتابة ملف صغير للقرص فى حين انه يملئ بـ 700 MB فهذا إهدار للمساحة .
- **متغير الجودة** . يمكن أن يكون متغير الجودة , وبعض العلامات التجارية أفضل من غيرها , وبعض العلامات يمكن أن تكون مسببة للمشاكل مع مشغلات محددة . إختبر علامات تجارية متعددة وقم بإنتقاء الأفضل لك . الكثير من المعلومات يمكن أن تجدها هنا www.cdfreaks.com
- **وضع البطاقات** . هى وظيفة أبطأ من وضع بطاقات على floppy Disks لأنه فى الحقيقة يجب عليك أن تحصل على بطاقات تثبت بشكل صحيح مركزى . الكثير من الأدوات والإجهزة وجدت لتساعدك فى فعل هذا بدقة .

لنطاق واحد فى وضع علامات على الأقراص , إنظر الموقع www.neato.com.

- **التكلفة** . بينما ان الأقراص المدمجة مخفضة الثمن , الحقائق ووضع العلامات سيزيد من التكلفة فى استخدام الأقراص .
- **بعض نقاط الضعف** . بينما هى قوية جداً , فهى عرضة لأنواع معينة من الضرر , خاصة الشرخ والتلف من جانبوضع العلامات لجانب من القرص (انظر الشكل)

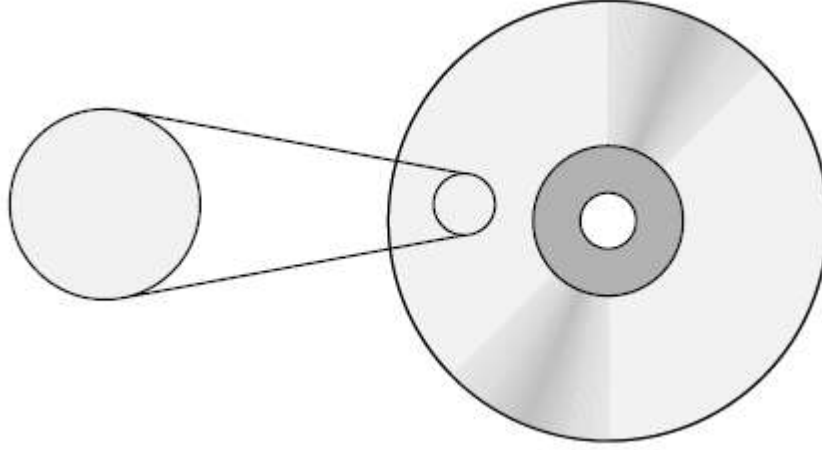


مثال للخدوش فى الاسطوانة

DVD

الأقراص الرقمية متعددة الاستخدامات

DVDs (يأتى هذا المصطلح من *digital versatile disc* وليس الإعتقاد الشائع , *digital video disc*) وأحدث وسيلة تخزين منها تصل غلى السوق , كأى وسيلة تخزين أخرى , وهى أيضاً لها عيوب ومميزات . تظهر فى الشكل .



Advantages

المميزات

- **المساحة** . تقدم مساحة هائلة — 4.5 GB بما يعادل ستة ونصف من الأقراص المدمجة CD . وأكثر من Floppy 3.250 . [فى الحقيقة فى يومنا هذا قد أصبحت المساحة أضعاف ماذكر فهذا الكتاب قد كتب فى 2005 ونحن فى 2010 . فهناك خمس سنوات من التطوير]
- **السرعة** . هى أسرع من CD .

Disadvantages

العيوب

- **التكلفة** . بينما التكلفة بالـ Mega byte من DVD تكون قليلة . فالتكلفة الإجمالية أعلى من الأقراص العادية CD . مما يجعلها غير مناسبة إذا لم تكن فى حاجة المساحة الضخمة التى تقدمها هذه النوعية من الأقراص . تذكر أيضاً أن تأخذ فى الحسبان ثمن الحقائق والأغلفة على تكلفته الأصلية , كما أن هناك وضع علامات وبطاقات تعريفية .
- **المساحة** . كما فى CDs . فربما نضطر أن نضع كمية ضئيلة من البيانات فى DVD . القاعدة العامة هى الاحتفاظ بهذه النوعية من الأقراص فى حين أنك تحتاج مساحة كبيرة لتخزين بياناتك بما يقرب من إسطوانتين CD (1.4 GB) .
- **عرضة للتلف** . مثل CDs , معرضة للتلف , خاصة فى جانب وضع العلامات .
- **توفر مشغل هذه الأقراص** . تذكر أنه ليست كل الأنظمة سيكون لديها مشغل DVD — خاصة Laptop أو الأقل من ثلاث أو أربع سنوات [فى الواقع الإن الأجهزة المحمولة أصبح طبيعياً وجود مثل هذه الأقراص DVD]

Burning Discs

حرق الأقراص

[الحرق هنا ليس إشعال النار فى الإسطوانة وإنما المراد به النسخ عليها]

حرق الأقراص (كلا CD , DVD) كلاهما سريع وسهل . كل ماتحتاجه هو ان تبدأ مفن الرسم الخاص بك .

- ☐ A suitable CD/DVD burner (writable drive)(المشغل)
- ☐ Software (البرنامج)
- ☐ Discs (الأقراص)
- ☐ Labeling gear (optional) (البطاقات وهو اختياري)

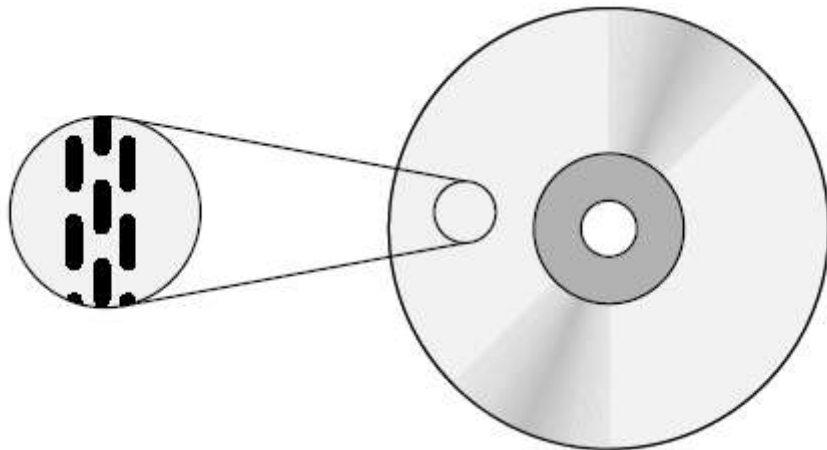
CD/DVD Burner

هذا هو المشغل الذى ستقوم بإستخدامه لإنشاء الأقراص من نوع CD/DVD . هناك الكثير من الأسماء التى يسمى هذا المشغل , أكثر إثنين شائعين .

- ☐ CD/DVD writers
- ☐ CD/DVD burners

يمكنك ان تختار منهم واحد تقوم بإنشاء CD/DVD به , أو كلاهما . إذا كنت تفكر فى شراء مشغل جديد . الأفضل لك أن تحصل على مشغل يقوم بإنشاء كلا من CDs/ DVDs .

قبل حرق الإسطوانة , لون القرص موحد , بعد حرق الإسطوانة , يقوم الليزر بحرق البيانات الرقمية إلى سطح الإسطوانة , فيغير الصبغة ويعطى الليزر المقروء شيئاً ما ليلتقطه . إنظر الشكل



المشغل الذى يقوم بحرق كلا من CDs/ DVDs يستطيع أن يقرأ أيضاً .

المشغلات المختلفة يمكن أن تحرق الأقراص بسرعات مختلفة , إذا كان هناك مشغل قام بحرق بيانات إلى إسطوانة في نفس الوقت الذي يقرأ فيه المشغل القياسي البيانات من الإسطوانة (المعدل قريب من 9MB في الدقيقة) إذاً فالمشغل يعرف بإسم One Speed أو 1x .
سرعة القراءة مساوية للسرعة التي يعمل مشغل الأسطوانات الصوتية في مشغل CD player .
القرص ذو سعة 700 MB يمكن أن يحتوى على 80 دقيقة من الصوت .

المشغلات ذات السرعة من النوع 1x Speed فهي بطيئة جداً , وتأخذ 80 دقيقة لحرق قرص / إسطوانة واحدة — بطيء جداً ومرهق فعلاً , للتغلب على هذا , قد أطلقت المشغلات الأسرع والأسرع وهذه الأيام يمكنك أن تحصل على مشغل يكتب أقراص على سرعات 52x , أو 52 سرعة , ويقلل وقت حرق القرص أقل من دقيقتين .

الأقراص التي تقبل إعادة الكتابة (الأقراص التي يكتب عليها ثم تسمح أكثر من مرة) يجب أن تحرق على سرعات أقل .

عامّةً , ترد مواصفات مشغلات شيئاً ما مثل 52x 32x 52x , مما يعنى أن 52 Speed تكتب للأقراص العادية , 32 تكتب للأقراص التي تقبل المسح وإعادة الكتابة , ويقرأ CD بسرعة 52 .

مشغلات الأقراص الرقمية DVD لا يقوم بالحرق بنفس السرعة . مشغلات DVD الجيدة التي تحرق الأقراص ربما في 8x أو 16x . مما يعنى أن القرص الكامل يمكن أن يحرق في 8 إلى 16 دقيقة , على التوالي .

هذه السرعة هي بالنسبة إلى سرعة القراءة القياسية لـ DVD — حوالى 1.5 MB بالثانية .

هناك أشياء أخرى يمكن أن نبحث فيها عن المشغلات "Burner" التي تقوم بحرق الإسطوانات . احد هذه المزايا يساعد في منع إضاعة الأقراص خلال مقاطعة البيانات من التدفق إلى القرص . هذه الميزة تعرف بإسماء متنوعة لتشمل Burnproof , buffer underrun protection . هذه الميزة متاحة في المشغلات عالية الجودة (والتي تكلف في العام فقط بعض الدولارات أكثر من المشغلات العادية) ولكن يرد لك الثمن في توفير الوقت والحد من فساد الأقراص .

Burning Software

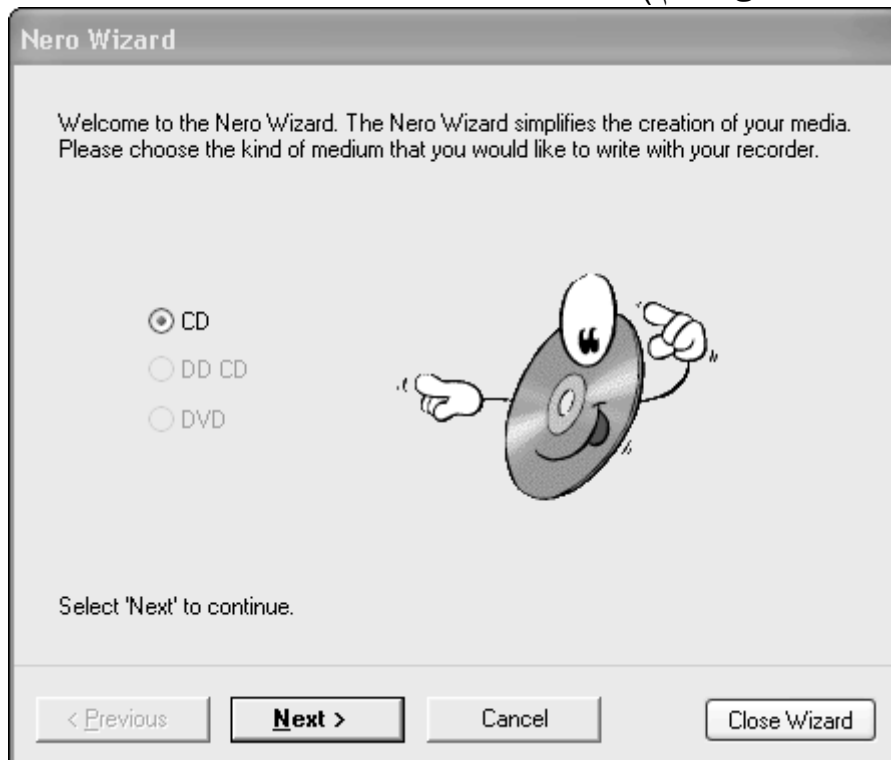
حرق البرامج

مجرد أن يكون لديك الجهاز [ويقصد هنا المشغل] . فهو الوقت لتفكر في البرنامج . هناك المئات من البرمجيات التي يمكن أن تستخدم لحرق إسطوانات ولكن الأفضل الذي يقدم نتائج محترفة في أقل ثمن هو Nero بواسطة Ahead Software , انظر الشكل .

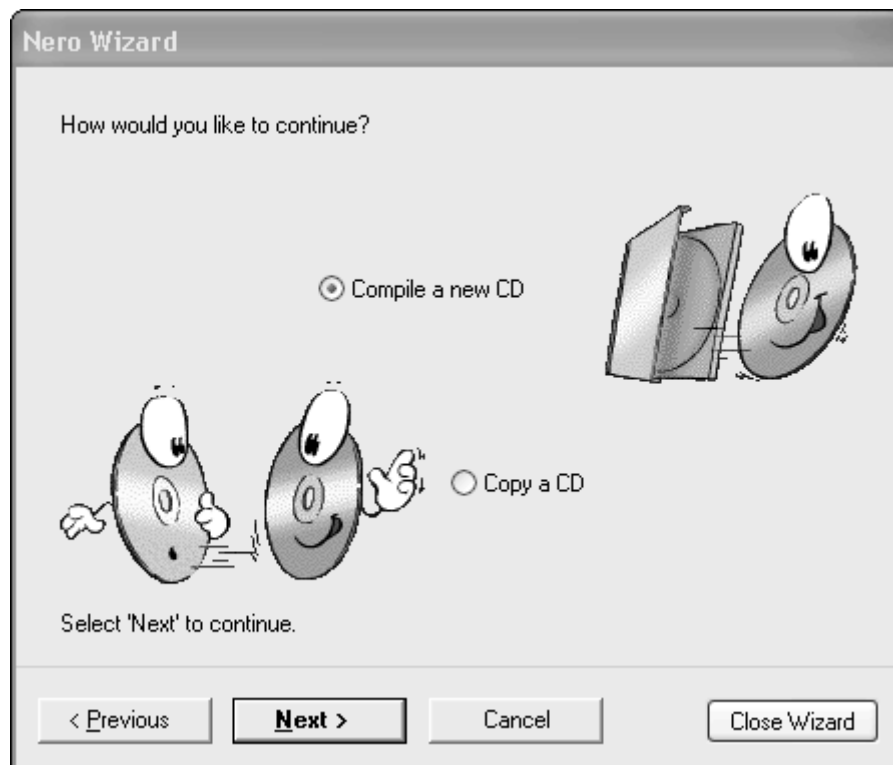


Nero يقوم بإنشاء CDs و DVDs بسهولة ! العملية كما يلي :

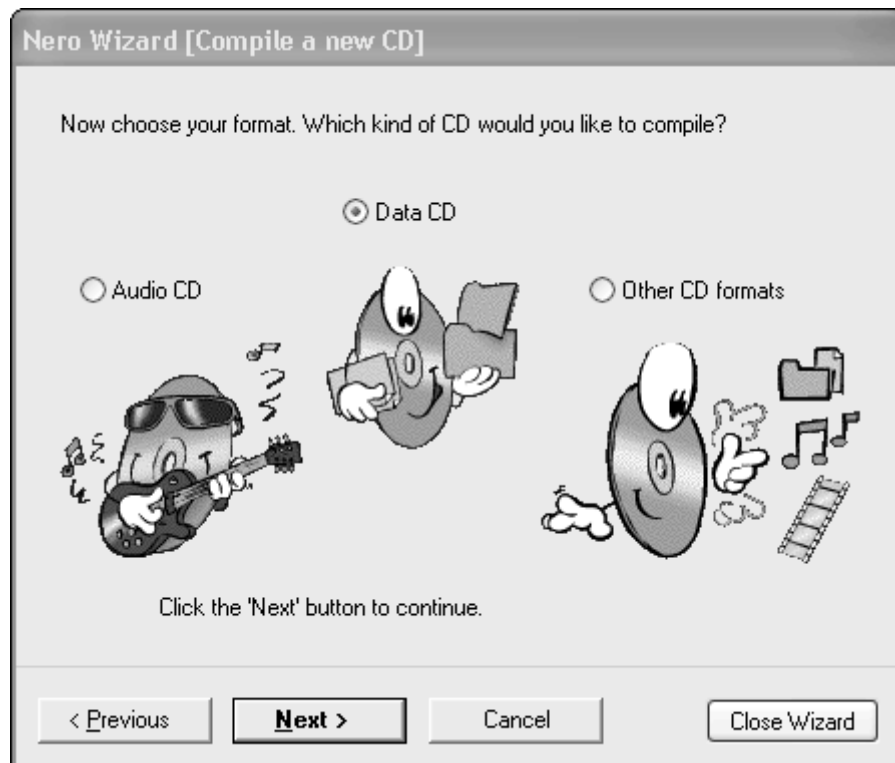
1 - قم بإختيار نوع القرص الذي تريد أن تنشأه (كما في الشكل) . هنا فقط يمكن أن ننشأ CDs (لأنه ليس هناك DVD Burner في النظام)



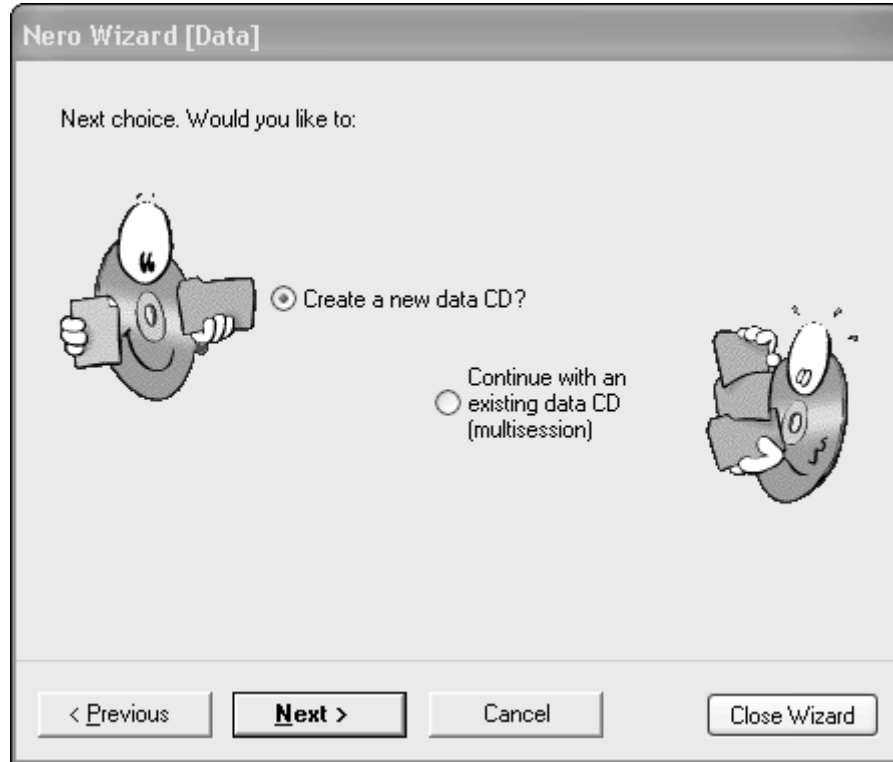
2 - التالى , إختار ما تريد فعله . فى الشكل لقد أخترت . compile a new CD



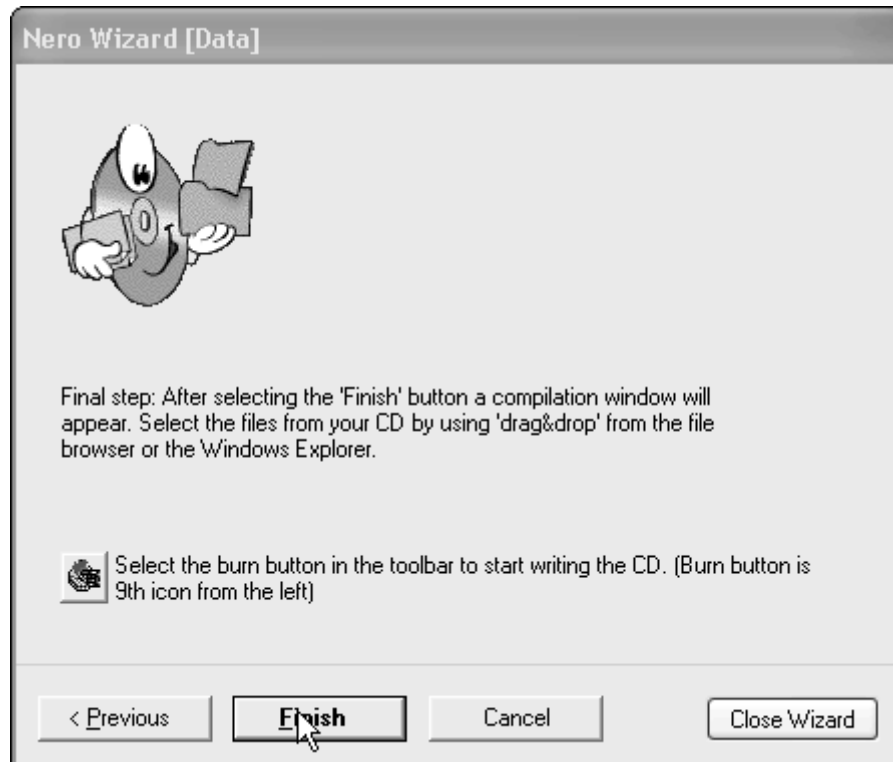
3 - اختر نوع القرص , فأنت تريد ان تنشئ قرص بيانات قياسى , كما هو مبين فى الشكل



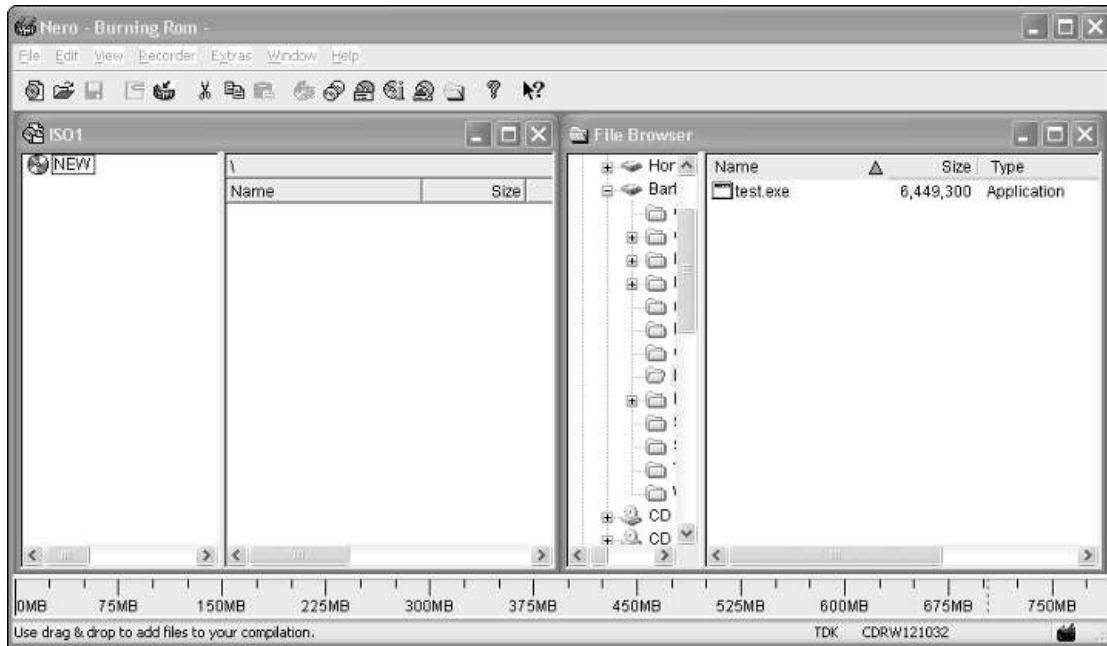
4 - إختار ما إذا كنت تريد إنشاء قرص جديد او وتستمر مع قرص موجود مسبقاً(انظر الشكل) . ستختار أن تنشئ قرص جديد .



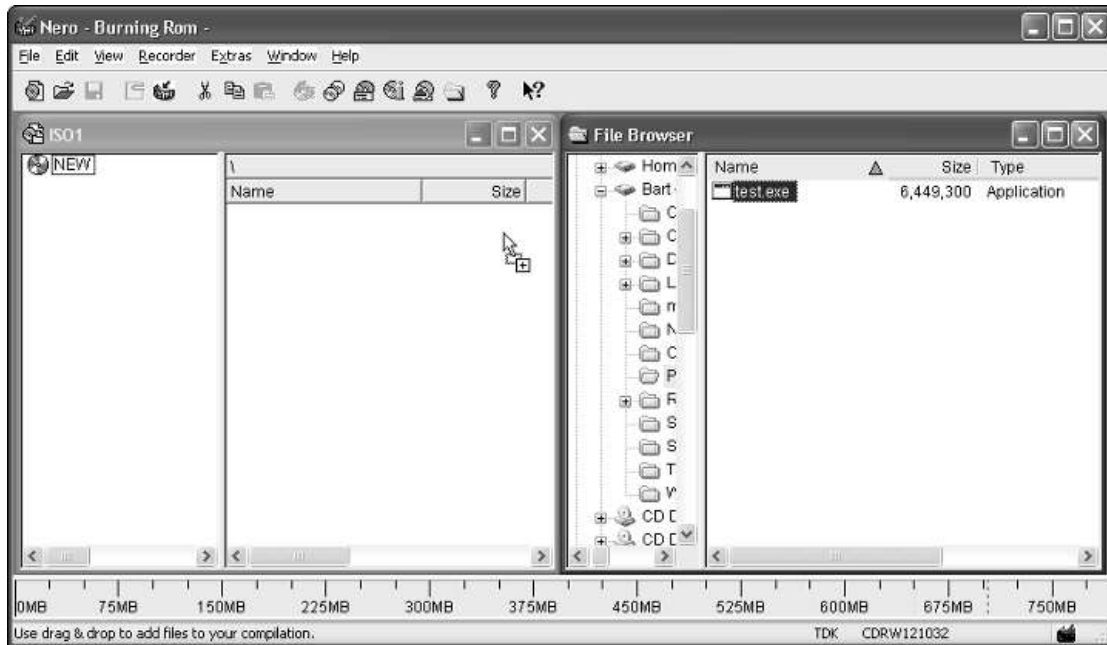
5 - اضغط finish , هذا لن ينهي العملية , بدلاً من ضبط الإعدادات للـ CD/DVD التي ستحرق .(انظر الشكل)



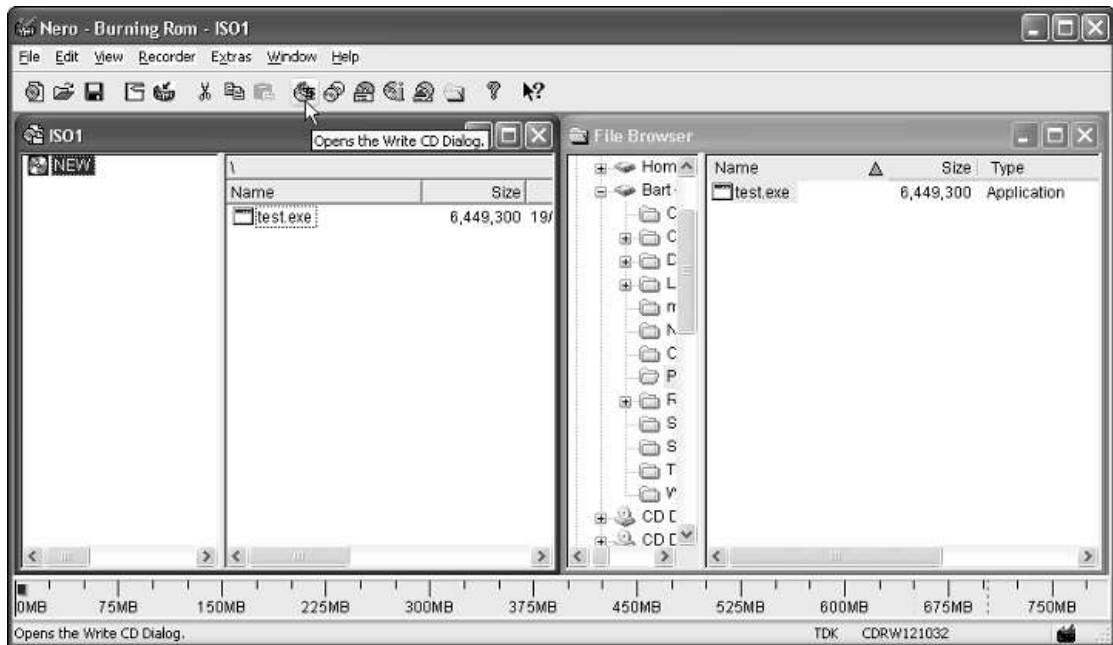
6 - الآن أنت مستعد ان تضيف الملفات التي تريد ان تنسخها إلى الإسطوانة . على الجهة اليمنى من نافذة البرنامج Nero , توجه إلى المجلد الذي يحتوى على الملف أو الملفات التي تريد ان تنسخها إلى الإسطوانة (انظر الشكل).



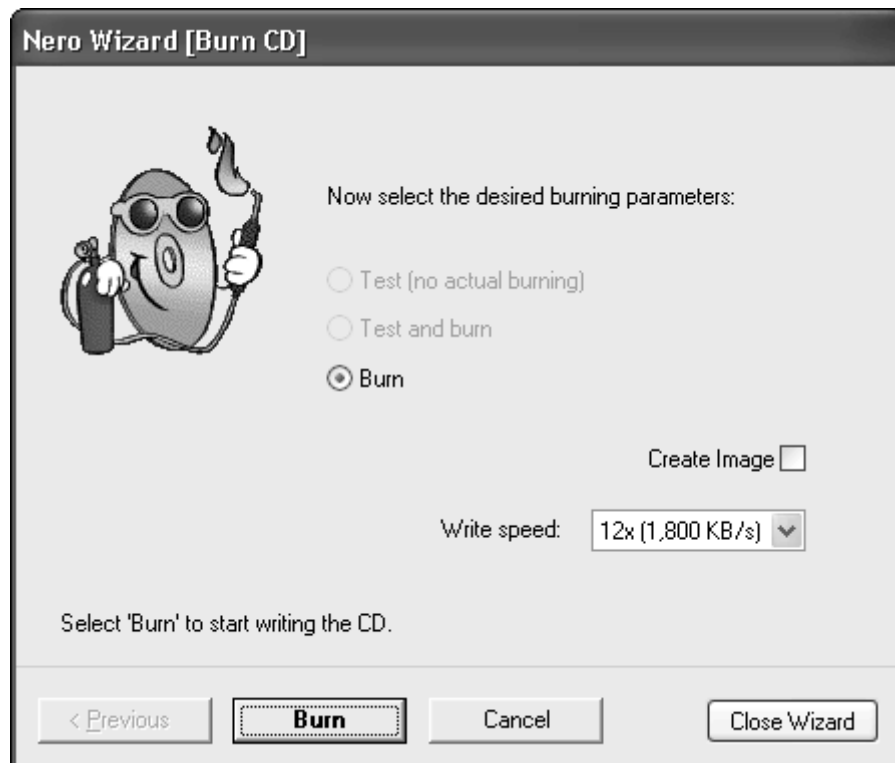
7 - قم بسحب الملفات والمجلدات التي تريد ان تنسخها من الجانب الأيمن إلى الجانب الأيسر , كما هو مبين في الشكل .



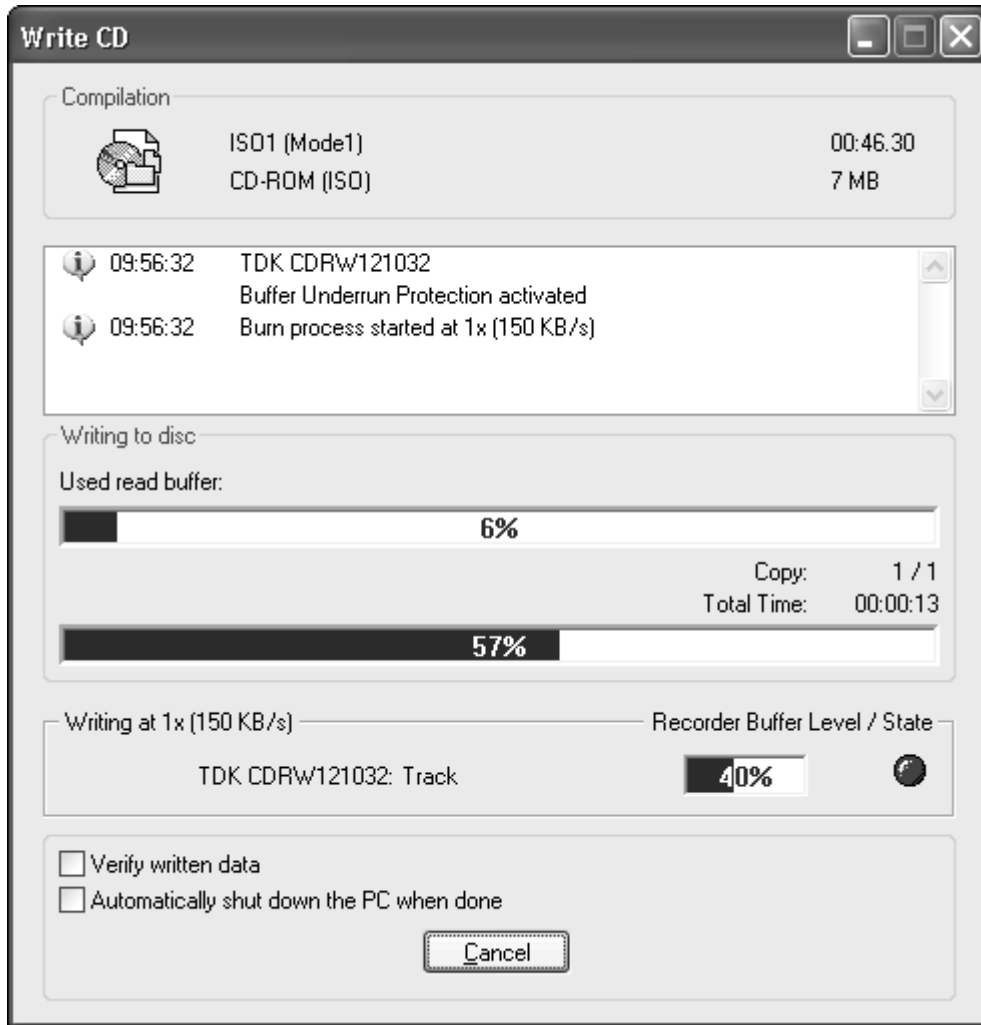
8 - بمجرد أن تنسخ الملفات , فحتاج أن تتأكد من أن لديك إسطوانة فارغة في المشغل ومن ثم تضغط على الأيقونة Burn CD تظهر في الشكل .



9 - بمجرد ان تضغط على ايقونة Burn CD , ستعرض لك رسالة تأكيد تسألك عن تأكيد ما تم سرعة الإحتراق . وهذا مبين في الشكل



10 سيتم حرق الإسطوانة الآن , وبمجرد أن يتم هذا الأمر (انظر الشكل) سيتم طرد ها للخارج.



11 -تمت العملية, ويمكنك أن تستمر وتضع البطاقات على الإسطوانة وضعه في حقيبة للحماية .

CD/DVDs

هما احد العناصر الرئيسية في توزيع البرمجيات . هناك مئات من الأقراص المختلفة المتاحة , ولكنهم في العام يتم إنقسامهم إلى ثلاثة تصانيف :

- ☐ Cheap discs (منخفضة الثمن)
- ☐ Fast discs (سريعة الأداء)
- ☐ Rerecordable discs (تقبل الحذف والنسخ مرة أخرى)

Cheap Discs الأقراص منخفضة الثمن

هي فقط — أقراص تشتري بالجملة منخفضة الثمن . يمكن ان يكون هذا جيداً , ولكن يمكن ان يكون هذا إقتصاد خاطئ لأنك يمكن أن تحصل على عدد من محاولات الفشل عند إستخدامهم . المشكلة الأخرى انهم أقل تسامحاً مع التلف والتأثر بعوامل البيئة (خاصة حرارة الشمس)

المشكلة الأخرى مع الأقراص منخفضة الثمن هي أنهم في الطبيعيلا يتم تعاملهم مع أقراص ذات سرعة قليلة . مما يعنى أنك سينتهى بك الحال أن تحرقهم على سرعة أبطىء من المشغل المصنف لتقليل الخطأ والأقراص عديمة القيمة .

Fast Discs

الأقراص السريعة

هي أقراص عالية الجودة , وهى مناسبة للمشغلات عالية الجودة أيضاً . هي أعلى ثمناً وأعلى جودة . القدرة على نسخ الإسطوانات على سرعة عالية يعنى أنك تقضى وقتاً أقل على النسخ أو الحرق الفعلى, لذلك يمكنك أن تقوم بنسخ أكثر من واحدة فى وقت أقل .

Rerecordable Discs

الأقراص التى تمسح ويكتب عليها

هي أقراص مرتفعة الثمن أكثر من القياسية "ذات الإستخدام الواحد " . فهي مفيدة عند عمل نسخ احتياطية لك أو أن تعطى برنامج إلى شخص ما سيقوم برد الإسطوانة لك مرة أخرى , ولكن ليست جيدة أن يوزع عليها البرنامج لأن المستخدمين يمكن ان يقوموا بالتغيير فيها أو حذفها .

العقبة الأخرى أنها أبطىء فى عملية النسخ من الأقراص العادية . ويعنى هذا ان هناك تقدير أقل فى إستخدامهم لتوزيع البرامج .

Labeling

وضع بطاقات أو علامات

ليس متطلب منك ان تضع بطاقات على الأقراص بأى طريقة من الطرق . مع ذلك , الأقراص التى لا يوضع عليها بطاقات تبدو غير محترفة ويمكن أن تفقد بسهولة أو الخلط بينها وبين الأقراص الأخرى . إذا كنت تخطط ان تقوم بنسخ أقراص متعددة للتوزيع . لننظر فى إمكانية الإستثمار فى وضع البطاقات . إستخدام البطاقات يجعل الإقراص أكثر احترافية ومفهوم ما بها .

لا تضع بطاقات على الأقراص غير المخصصة والمصممة لذلك . وضع أى شئ غير غير مصمم لذلك يمكن ان يتسبب فى تلف القرص وأيضاً تحدث خلل فى وزن القرص فى التوزيع . ويعنى هذا انها ربما تكون مشوشة جداً فى المشغل ومن الممكن تدميره .

أيضاً , خطورة أخرى فى إستخدام بطاقات غير صحيحة يمكن أن تؤتي ثمارها في محرك الأقراص والاضرار به.

Packaging التعبئة والتغليف

لديك إختيارات كثيرة عندما تأتى إلى التعبئة والتغليف لبرنامج فى حزمة واحدة :

- ☐ Simple packaging
- ☐ Customized packaging
- ☐ Elaborate packaging

Simple Packaging التغليف البسيط

هى الحد الأدنى الذى يمكن ان تستعمله فى حماية القرص وتزويد المستخدم النهائى بمعلومات أساسية . [تم ترك جملة لصعوبة الترجمة]

ما تضعه على البطاقات والعلامات متوقف عليك ولكن الحد الأدنى

- ☐ Program name (إسم البرنامج)
- ☐ Version number (رقم الإصدار)
- ☐ Author contact (Web address, e-mail address) (بيانات الاتصال بالمؤلف)
- ☐ Basic copyright information (معلومات عن حقوق النسخ)
- ☐ Type of disc (CD or DVD) (نو القرص)
- ☐ Basic setup or installation instructions (تعليمات تثبيت أو إعدادات أساسية)



Customized Packaging

التغليف المخصص

تعنى ان هناك مجهود أعلى مع التغليف . يمكن ان يعنى شعارات أو صور وربما دليل مستخدم للمساعدة أو تعليمات تثبيت منفصلة .

تذكر أن هذا كله يضيف المزيد من التعقيد والتكلفة للمشروع . ولكن إذا قمت بعمل تغييرات إلى تطبيقك ربما تحتاج إلى ان تقوم بتغيير النسخة المغلفة , لذا لا تطبع أكثر مما تحتاجه .

Elaborate Packaging

التغليف المفصل

وهذا يعنى أن تكون أكثر تفصيلاً والحصول على صناديق وأغطية . وهذا عادةً ما يضيف كمية كبيرة إلى التكلفة إذا لم يتعامل مع الأحجام المرتفعة وليس مناسباً للأحجام الصغيرة والمتوسطة في التوزيع .

Virtual Distribution

من المحتمل أنك قد خمنت , أنه يشمل على الإنترنت كوسيلة توزيع لمشروعك . قبل أن يصبح الإنترنت التيار الرئيسى . فكان التوزيع العادى هو الاختيار الوحيد المتاح المفتوح امام معظم الناس , ولكن حتى هذا الوقت كان المبرمجون الرواد يسخرون قوتهم في التوزيع الإلكتروني بواسطة تحميل أو رفع برامجهم إلى BBS, bulletin boards متاح للآخرين أن يقوموا بتزيله والإحتفاظ به . فالأمور أكثر تفصيلاً الآن وكما دائماً , لها سلبيات وإيجابيات في استخدام الإنترنت كنظام توزيع للبرمجيات .

Pros of Virtual Distribution

الإيجابيات

هناك عدد هائل من المنافع في استخدام توزيع البرمجيات الظاهري العملى .

- **منخفض الثمن** . منخفض الثمن جداً , لا تحتاج إلى مشغل نسخ إسطوانات , إسطوانات , بطاقات , كل ما تحتاجه هو الإتصال بالإنترنت ومكان ما تقوم بتحميل أو رفع برنامجك عليه .
- **متاح 7/24** : أى شخص يريد برنامجك يمكنه الحصول عليه بسهولة وسرعة . لا حاجة أن ينتظر إلى أن يستلم البرنامج عبر البريد .
- **السرعة** . هناك المزيد من منافع السرعة — سرعة تحديث المنتج , لا حاجة للملل من حرق الإسطوانات , سرعة جعل الإصدار الجديد متاح , الإنترنت هو كل ما يخص السرعة , وهو أسرع طريقة لجعل البرامج والمشاريع متاحة للجميع .
- **جماهير ضخمة** . لم يكن هناك جمهور ضخم ليستفاد منه !
- **الاتصال المباشر مع الجمهور** . يدعك الموقع تتحدث مباشرة مع المستخدمين

Cons of Virtual Distribution

السلبيات

قبل أن تستخدم الإنترنت كوسيلة لتوزيع مشروعك , هناك بعض الأمور التى تحتاج ان تأخذها فى الاعتبار .

- **التوزيع الغير مصرح** . مم لا شك فيه أن الإنترنت قد شجع توزيع البرامج الغير مرخصة إذا لم تريد أن تحصل على المزيد من المال فى برنامج معين . من الصعب حماية من هذا (وحتى ذلك الحين من المستحيل تقريبا ان تجعل أي شيء يضمن لك الحماية) .
- **لا وصول لغير مستخدمى الإنترنت** . المشكلة التى فى إستخدام الإنترنت كوسيلة لتوزيع البرمجيات هو انك لا تستفيد من الأسواق الصغيرة الغير متصلة بالإنترنت .
- **الإخبار تنتقل بسرعة** . أحد الأشياء التى يجب أن تكون حذراً حيالها عند استخدام الإنترنت هو أنك إذا أطلقت برنامج به بعض الأخطاء أو المشاكل فإن الإخبار تنتقل بسرعة كبيرة جداً.
- **كن مستعد لإعادة النظر** . احد الأمور التى يحبها مستخدمى الانترنت هى مراجعة الأمور .وإذا كان برنامجك حائزاً على تقييم أعلى سترى انه إستعرض فى أماكن . خذ السلبى مع الإيجابى وإستخدم الخلفية للتحسين من المنتج .
- **الدعم** . البرنامج يمكن أن يكون ضحية لنجاحه , ويمكن أن تحصل على جمهور واسع أكثر مم هو منتظر . كثرة المستخدمين تعنى أن توفر مزيد من الدعم . وهذا يمكن أن يكون إستنزاف لوقتك ومواردك . إحسب وخطط للدعم لأى برنامج تقوم بإطلاقه .

Considerations for Virtual Distribution

الإعتبارات للتوزيع الظاهرى

قبل أن تغرق وتغوص فى أى شكل من التوزيع , ولكن التوزيع من خلال الإنترنت بشكل خاص ,إليك بعض الأمور التى تأخذ فى الإعتبار .

لا تتعجل ما تفعله , خذ وقتك وإنظر كيف الإخريين يقوموا بتوزيع منتجاتهم عبر الإنترنت .

Free versus Commercial?

المجانى مقابل التجارى ؟

هل قمت بالتخطيط لجعل برنامجك متاح مجانى للجميع للتنزيل على الإنترنت ويستخدموه أم سيكون عليهم أن يدفعوا نظير إستخدامهم برنامجك ؟ هناك طرق كثيرة يمكن أن تتبعها للدفع للبرنامج .ويمكن أن تعمل على نظام تبرع .

أخذ المدفوعات ليس صعباً , وهناك الكثير من المواقع التى يمكن أن تساعدك . إليك البعض :

- ☐ www.paypal.com
- ☐ www.download.com
- ☐ www.handango.com
- ☐ www.tucows.com
- ☐ www.winplanet.com
- ☐ www.shareware.com

على كل الأحوال , مساحة أخرى تحتاج أن تفكر فيها هى ترخيص برنامجك . ترخيص البرمجيات يمكن أن يكون بسيط أو معقد . مع المنتج التجارى ربما كنت أفضل حالا وأكثر أمناً إذا بحثت عن نصيحة قانونية لإن

القوانين تتغير من ولاية إلى ولاية ومن بلد إلى بلد . وإذا كنت ستقوم بتوزيع المنتج مجاناً إذاً يمكن ان تحميه بإستخدام ترخيص GNU General Public .

نسخة من هذا الترخيص تجدها فى الرابط الذى امامك . www.fsf.org/licenses/gpl.html .

تذكر أن التوزيع تحت هذا الترخيص لا يجعلك , كمؤلف , تفقد حقوق النسخ , فأنت تحتفظ بحقوق النسخ والقدرة على تولى مسؤولية البرنامج . بناء على اتفاقية Berne , كل شئ يكتب , تلقائياً يحفظ حق النسخ له منذ أن توضع فى شكل ثابت بواسطة الشخص أو الشركة التى وضعت فى هذا الشكل . لذا , فأنت لا تحتاج أى تفعل أى شئ حيال أن تدعى حقوق النسخ على ما تكتبه , مادام أن أى شخص لا يمكن أن يدعى أملاك عملك .

تسجيل حقوق التأليف والنشر فى الولايات المتحدة هو فكرة جيدة جداً, لمزيد من المعلومات وكيفية فعل هذا يرجى زيارة www.copyright.gov اجر هذا بيذا من أقل من 30 \$ لتسجيل مبدئى .

وإذا كنت مهتماً فى نشر وتوزيع برنامجك خلال مصدر الكود , إذاً مكان عظيم على شبكة الإنترنت لهذا هو : www.sourceforge.net هنا يمكن أن تنظم الكود , التطبيقات , تقارير الشوائب والأخطاء , والمراجعات .

Full-Time Job versus Hobby وظيفة الدوام الكامل مقابل الهواية

هل تبرمج وتطلق البرامج كهواية او تفعل هذا وترى أنها وظيفة دوام كامل ؟ هذا يعتمد على مدى الجهد والتزام ما تريد ان تضعه لتعزيز برمجياتك , وتحصل على مبيعات لا حدود لها .

ونصحتى ان يكون التعامل بحذر, خذ وقتك وقم بالمزيد من البحث العلمى , كما أخذت وقتك فى تعلم البرمجة , فانه يأخذ وقت لتعزيز وبيع المنتجات خلال الإنترنت . كن حذراً ألا تدخل حقل الإلغام القانونية . وخذ كل خطوة بعد ان تبحث فيها جيداً , وإذا شعرت انك بحاجة لشئ ما , إسعى لنصيحة قانونية .

النصيحة القانونية يمكن أن تكون مرتفعة الثمن — خذ هذا فى الاعتبار قبل البحث عنها , خاصة إذا كنت تظن ان العودة ستجعل الأمر أقل ثمناً .

Supported versus Unsupported المتوفر له الدعم مقابل ما لا يتوفر له دعم

معظم المنتجات الناجحة ستحتاج إلى بعض الدعم لأنه لا يهم كم هو برنامجك جيد أو كم برمجتك جيدة (حتى لو كان لا تشوبه شائبة) . بعض الناس سيكون لديهم مشاكل مع عملك ببساطة , بسبب الصراعات مع الأجهزة الحسية للحاسب Hardware أو برامج أخرى موجودة على النظام الخاص بهم .

من ناحية , إذا كان البرنامج مجانى , يمكن ان تقرر ألا تقدم أى دعم , العبارة الرئيسية فى إتفاقية الترخيص للبرنامج "المخاطرة خاصة" و "كما هو" . من ناحية اخرى , إذا كان البرنامج لديه تتبع جيد , يمكن أن تنشئ منتدى أو موقع لقائى يستطيع فيه المستخدمين أن يساعدوا الذين لديهم متاعب أو مشاكل , هذه الوصفة قد نجحت مع الكثير من مطورين البرمجيات .
إذا كان البرنامج هو تطبيق تجارى او إذا قررت أن تقدم الدعم للتطبيق المجانى , إليك بعض الاختيارات المتاحة لك .

Detailed FAQ

أسئلة واجوبة تفصيلية

يمكنك ان تجعل من المتاح قائمة بالأسئلة والأجوبة المفصلة (Frequently asked Questions) FAQs التي تغطي المشاكل التي تنتبأ بحدوثها في التطبيق . يمكنك إضافة إلى هذه المشاكل التي يقوموا بإيجادها بأنفسهم على ما يبدو لك .

القائمة الجيدة والشاملة بالأسئلة والأجوبة يمكن أن تستخدم في زيادة مسائل الدعم للآخرين . ويحررك من إفاد نفس الأسئلة مرات ومرات (بالرغم من أنك لن تستطيع ان تجعل المستخدمين يقرأون هذه الأسئلة قبل سؤالك أن تجيب على سؤال قد تم تغطيته في القائمة)

E-mail Support

دعم البريد الإلكتروني

توفير الدعم بالبريد الإلكتروني هو احد الطرق المتاحة لك . يرسل الأشخاص أسئلتهم , وترد عليهم .

إذا قمت بإختيار هذه الطريقة , تأكد بأن لديك قائمة من المعلومات الأساسية التي تحتاجها من المستخدمين في أول رسالة بريد إلكتروني (خلاف , سيكون عليك تبادل عدد من الرسائل للحصول على المعلومات التي تريد)

بعض المعلومات الأساسية التي يمكن أن تحتاجها هي :

- ☐ The product (المنتج)
- ☐ Product version (إصدار المنتج)
- ☐ Operating system (نظام التشغيل)
- ☐ Other software installed (البرامج الاخرى المثبتة على الحاسب)
- ☐ Hardware (CPU, RAM, and so on) (مكونات الحاسب المعالج, الذاكرة, وهكذا)

إذا اخترت الدعم من خلال البريد الإلكتروني , تأكد من ان لديك عنوان بريد إلكتروني محدد قد تم إعداده لدعم الرسائل . آخر شيء تريد ان حساب البريد الإلكتروني الشخصي مملوء بأمور الدعم! بإمتلاك حساب بريد منفصل , يمكنك ان تختار ان تتعامل مع قضايا الدعم عندما تريد .

Forum/Web Support

منتديات / مواقع الدعم

هذه الأيام , يبدو ان كل شخص لديه موقع او منتدى جميل — لذا لماذا لا تستخدم هذه الطريقة لتدعم منتجك ؟

- الموقع والمنتدى لديه مزايا متعددة أكثر من الطرق الأخرى :
 - يمكن للمستخدمين أصحاب المشكلة نشر سؤالهم في اي وقت , ويمكن ان تنشر لهم ردك متى تريد .
 - إجابتك على شخص واحد ربما تساعد الآخرين مع نفس المشكلة .
 - مستخدمين آخرين ربما يساعدون المستخدمين الجدد الذين لديهم مشاكل .
 - تبقى دعمك مفتوح , تجعل دعمك اكثر وداً , وتظهر أنك تأخذ المشاكل على نحو جيد وتتعامل معهم .

Telephone Support الدعم بالهاتف

كن حذراً جداً عند فعل هذا — الهاتف يمكن ان يأخذ كل وقتك . فهو جيد للشركات الكبرى التى لديها مبيعات كافية لتدبير موظفين دعم .

Summary الملخص

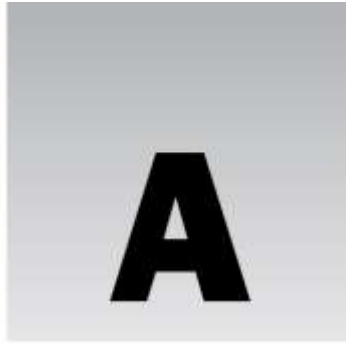
فى هذا الفصل , لقد ألقينا نظرة على المرحلة النهائية للكثير من مشاريع البرمجيات — توزيع التطبيقات أو المنتجات .

لقد بحثنا فى انماط التوزيع العادى والظاهرية وقمنا بفحص الإيجابيات والسلبيات لكلاً منهم بالتفصيل. ولقد شاهدت أيضاً الأختلاف فى التكلفة المتعلق بكلاً من طرق التوزيع والنشر .

وخلاصة القول أن أى واحدة تختار فذلك يعتمد على عدد العوامل , مثال ما هى تكلفة الجمهور و ما هو حجم التوزيع الذى تخطط له . الإجابة على مثل هذه الأسئلة تساعدك على أن تسلك الإختيار الصحيح لإحتياجاتك .

أيضاً لقد شاهدت آلية الدعم التى ربما تحب أن تقدمها وفحص التكلفة والإيجابيات والسلبيات لكل واحدة حتى إذا لم تكن فى حاجة إلى توفير أى دعم لكل مشاريعك .

ما تختار ان تفعله يعتمد على ما تخطط له — إذا كنت بصدد مشروع صغير يعمل على حل مشكلة لك , وتريد أن ينتفع به الآخرين , إذاً فالطريق الذى تأخذه سيكون مختلف تماماً عن الطريق الذى يأخذه شخص آخر قد قضى أشهر للعمل على مشروع كبير الحجم ويأمل أن يجنى بعض المال منه . إختيار بحرص بناءً على متطلباتك وتخطيطك .



Glossary

قائمة بالمصطلحات

هذا الملحق هو قائمة بالمصطلحات لأهم مصطلحات البرمجة , معظمها تم إستخدامه فى هذا الكتاب . والبعض لا وقد أدرجتها لأكمال الفائدة .

لا تحاول أن تحفظ هذه المصطلحات فى الذاكرة او تقلق على انك تتذكر كل شئ يذكر فى هذه القائمة — خذ وقتك . كلما زادت الممارسة والبرمجة , كلما كنت افضل , وسترى أنك بمرور الوقت ان هذه المصطلحات قد علفت فى ذهنك .

كن على راحة إن أردت أن تنشأ بعض الملاحظات على الهوامش وأضف مصطلحاتك إلى القائمة .

المصطلح	الوصف
Address	عبارة عن موقع الذاكرة .
Algorithm خوارزمية	التعريف الدقيق لخطوات حسابية
Argument	القيمة التى يتم تمريرها إلى الدالة
Array المصفوفة	تسلسل متوالى لعناصر مرتبطة من نفس النوع
ASCII	الكود الأمريكى القياسى لتبادل المعلومات هو كود 7-bit لتمثيل الحروف
ASP	Active Server Page تقنية تطوير مواقع إنترنت طورت بواسطة مايكروسوفت ومبدئياً تستخدم VBScript
Assignment Operator	هو معامل يستخدم لإسناد قيمة إلى متغير فهو متفاوت من لغة برمجة إلى لغة برمجة .
Backdoor	هى وسيلة وصول للبرنامج تجاوزاً لكل إحتياطات الأمن
Backup	هى نسخى تحفظ من الملفات الهامة فى حالة الضياع , حادثة إستبدال , أو تلف , أو نسخة تحفظ كأرشيف .
Beta Test	إختبار البرنامج ليكتشف أى شوائب مفقودة . فإختبارها يتم بواسطة أناس غير مبرمجين ولا يعرفون كيف يتم إستخدام البرنامج .
Binary	نظام أرقام يستخدم الأعداد من 1 إلى 9 فقط (تسمى bits)

المصطلح	الوصف
Bits	الأصفار والآحاد التى يتكون منها نظام الارقام الثنائى Binary
Block Comment	هو كتلة من النص المعرف كتعليقات للبشر ليقرأوه بدلاً من التعليمات للحاسب . فى C++ و JavaScript كتلة التعليق معرفة ببداية بـ /* وتنتهى بـ */ .
Boolean (bool)	نوع المنطقى مبنى بالداخل . يمكن أن قيمته (1) True أو (0) False
Boot up	عملية بدء تشغيل الحاسب
Bug	مصطلح برمجة شائع يستخدم لوصف خطأ قد حدث
Byte	8bits
C	لغة برمجة
C++	المتابعة للغة C++ . وهى لغة برمجة غرضها عام وواسعة الإنتشار . وتعتبر لغة قوية جداً ولغة برمجة متعددة الإستعمال
Call	مصطلح يستخدم لوصف تشغيل دالة من أخرى .
CD-ROM	القرص المدمج للقراءة فقط
CD-RW	القرص المدمج للقراءة / للكتابة
Comment	النص المدرج داخل مصدر الكود وليس فيه منفعة للحاسب ولكن مصمم كملاحظات للمبرمج ليقراها فيما بعد او مبرمجين آخرين ربما يقرأوها فيما بعد . وقد صممت التعليقات لتجعل الكود أسهل أن يقرأ
Compiler	برنامج يقوم بتحويل مصدر الكود إلى كود مقروء بواسطة الآلة
Concatenate	هو إرتباط إثنين او اكثر من السلاسل النصية او القيم الحرفية لجعلهم واحد
Constant ثابت	حرفياً , يستخدم لتعريف متغير لا يمكن تغييره
Curly Braces	{ و } وتستخدم فى JavaScript , Java , C++ وغيرهم . وتستخدم لتعريف كتل من الكود خلال مصدر الكود .
Cursor	المؤشر الذى فى شاشة الحاسب
DataBase	قواعد البيانات هى تجميع من المعلومات قد تم تنظيمه لذا فيسهل الوصول إليه وإدارته وتحديثه
Debug	عملية حذف الشوائب من الكود
Demo	هذا الاسم قد أعطى للبرنامج التجربة أو التطبيق البرمجى وعادةً ما يكونوا محددين بوقت أو أقل وظيفياً
Donateware	برنامج مجاني الإستخدام ولكن المؤلف يسأل التبرع إذا أحببت البرنامج
DVD	Digital versatile disc قرص رقمى متعدد الإستعمال
End User	الشخص الذى يستخدم التطبيق المطور بواسطة المبرمج . فالمستخدم النهائى ليس لديه معرفة بالعمل الداخلى للتطبيق . وهما الأشخاص الذى يجب أن تأخذهم فى إعتبارك بينما تطور تطبيق .
Equality Operator	وتستخدم لإختبار مساواة قيمة بقيمة أخرى . المعامل == يستخدم فى C++ . إذا كان لديك متغيرين (x , y) وكلاهما قد إسند إليهم القيمة 7 فعندها x==y سيكون الناتج عنها True
Error	شئ ما غير متوقع . يحدد سواء كان خطأ فى الكود أو خطأ فى المدخلات او المخرجات
Escape Charcter	الحرف \ (أيضاً تسمى Backslash أى شارطة مائلة) . ويستخدم هذا كحرف اولى فى تمثيل الحروف التى لا يمكن أن تمثل بواسطة ASCII — على سبيل المثال : سطر جديد (\n) . ومسافة أفقية (\t) .
Exception	مصطلح اخر للخطأ
Executable	برنامج كامل يمكن تشغيله
False	قيمة منطقية . عادةً ما تمثل بـ 0 . تعلن فى الكود ان 1 + 1 = 3 فهذا مثال على جملة false .

المصطلح	الوصف
Field	مساحة نصية ربما تتطلب بعض المدخلات (مثل نص أو أرقام)
File	تسلسل من Bytes أو كلمات تحتوى على معلومات فى الحاسب . الملفات عادةً ما تخزن فى المشغل الصلب Hard Drive أو وحدة تخزين خارجية
Floppy Disk	وسيلة تخزين قابلة للنقل فى شكل وسيلة مغناطيسية يسكن فى حقيبة بلاستيكية جامدة تقاس بـ 3.5 بوصة مربعة .
Function	تجمع متسلسل لجمل البرمجة
GigaByte	1,073,741,824 bytes
Grammer	وصف لقواعد بناء الجملة اللغة
Hard Drive (Hard Disk)	مصطلح قديم للتخزين فى الحاسب . البيانات التى تحفظ إلى الهارد سيكون متاح بعد غلق الجهاز الحاسب وإعادة تشغيله .
Help file	الملف الذى يحتوى على معلومات عن إستخدام التطبيق
HexaDecimal	نظام رقمى من 16 رقم . بدلاً من عشرة أعداد , فهو يستخدم 16 (0 إلى 9 و A , B , C , D , E , F)
HTML	Hypertext Markup Language لغة تعليمات النص المتشعب وهى لغة تستخدم فى إنشاء صفحات الإنترنت
Hungarian notation	إتفاقية كتابة كود , بين أمور أخرى , توصف كيف ترمز معلومات نوع بيانات إلى أسماء متغيرات , تجعل الكود أسهل للتبع
I/O	مدخلات / مخرجات
If statement	جملة تختار ما بين بدلين يعتمد على شرط .
Inequality operator	معاملات لا تساوى تستخدم لإختبار قيمة واحدة فى انها لا تساوى قيمة أخرى . فى ++C يستخدم المعامل != . إذا كان لديك متغيرات (x , y) وكلاهما أسند إلى القيمة 7 فعند $x \neq y$ سيكون False .
Input	البيانات المدخلة إلى التطبيق (عادةً خلال لوحة المفاتيح)
Interpreted Code	الكود الذى لا يحتاج إلى تفسير , بدلاً فهو يترجم كما يطلب إلى كود مقروء بواسطة الآلة باستخدام المترجم .
Interpreter	تطبيق يأخذ مصدر البيانات (عامّة كود كتابات) ويحوّله إلى كود مقروء بواسطة الآلة عند الطلب , JavaScript هى مثال على اللغات التى تستخدم المترجم Interpreter
JavaScript	Script Language أو Interpreted Language من شركة NetScape
KiloByte	1.024 Byte
Linux	نظام تشغيل يعتمد على Unix , عادةً مجاناً
Liteware	برنامج قد أنشئ وأُتيح لك مع فقد بعض الوظائف الموقوفة حتى تدفع للبرنامج
Machine Code	الكود المكتوب فى النظام الثنائى (0 و 1) وهذا هو الكود الذى يفهمه الحاسب
Markup	تسلسل من الحروف أو الرموز فى ملف نصى تحدد كيف ينبغى لهذا الملف أن يعرض ويترجم .
MegaByte	1,048,576 bytes
Null	الصفّر , 0 , لا شئ .
Nybble	4 bytes
Open Source	برنامج بمصدر الكود المتاح للإستخدام أو التعديل كمستخدم أو مطور آخر لما يراه مناسباً . يختلف هذا عن التطبيقات التجارية , حيث يبقى مصدر الكود سر .
Operating System	هو البرنامج الذى , بعدما يتم تحميله على ذاكرة الحاسب , يدير كل البرامج الأخرى على الحاسب .

المصطلح	الوصف
Operator	تدوين تقليدي للعمليات المبنية بداخلها , مثال / , * , - , + .
Output	البيانات المرسله خارج التطبيق . يمكن أن تكون للشاشة , ملف , أو طابعة .
Parameter	متغير يتم الإعلان عنه في دالة وتمثل Argument
PHP	Hypertext Preprocessor وتستخدم كبديل لتقنية مايكروسوفت ASP . لإنتاج صفحات إنترنت على الخادم التي ترسل للمستخدم لتعرض من خلال مستعرض الإنترنت .
Plain Text	أسم آخر للحروف الإبدئية النصية الغير منسقة
Platform	أسم آخر للنظام الذي يقوم بتشغيل التطبيق . انظمة التشغيل المختلفة والحاسبات المختلفة تصنف Different Platform
Postcardware	برنامج مجاني يطلب من المستخدم أن يرسل بطاقة بريدية إلى المؤلف للبرنامج
Procedure	إسم آخر للدالة . وهي مجموعة من الجمل البرمجية المجمعة مع بعضها للتشغيل .
Programming	علم كتابة تعليمات يمكن أن يفهما الحاسب (على الرغم من أن البعض يسمونها فن)
Pseudocode	تفصيل ولكن وصف مقروء لما يفعله البرنامج أو الخوارزمية العادية , وهذا ما يعبر عنه رسميا لغة طبيعية بدلاً من لغة برمجية .
RAM	ذاكرة المرور العشوائية (Random Access Memory) . الذاكرة المؤقتة التي يستخدمها الحاسب في الإمساك بالبرامج والبيانات حال إستخدامها . البيانات هنا تحذف عند غلق الحاسب .
Runtime	عند تشغيل البرنامج
Script Language	لغة Interpreted وليست Compiled [الترجمة متشابهة]
Setup	المصطلح الشائع يستخدم لوصف عملية تثبيت برنامج على الحاسب
Shareware	برنامج يوزع على الإنترنت أو على إسطوانات , الدفع مطلوب لتستخدم البرنامج , على الرغم من ان هناك فترات تجربة .
Software	مجموعة من البرامج .
Source Code	الكود المكتوب بواسطة المبرمج , عادةً يكون في شكل نص عادي.
Source file	الملف الذي يحتوي على الكود . هذا الكود يحتاج إلى تفسير او ترجمة ليعمل .
Statement	وحدة أساسية في الدالة . الجمل تتحكم في تدفق التنفيذ .
Statement Delimiter	حرف أو مجموعة من الحروف تستخدم لتحديد نهاية جملة الكود . C++ , JavaScript يستخدمان علامة ; (على الرغم من أن إستخدامها في JavaScript إختياري)
String	تسلسل من الحروف
Switch Statement	جملة تحديد من قائمة لبدائل تعتمد على قيمة .
Syntax	مجموعة من القواعد يمكن ان يتم مقارنتها للتهجي والقواعد الإملائية والنحوية للغة منطوقة / مكتوبة التي تصف كيف أن مصدر كود البرنامج ينبغي ان يكتب لذا يمكن أن يعمل بدون أخطاء
Text	تسلسل من الحروف يمكن أن يقرأ بواسطة البشر

المصطلح	الوصف
Text editor	برنامج لعرض , تحرير و إنشاء ملفات النصوص العادي . النص الذي في هذه الملفات غير منسق بأى طريقة وبشكل مبسط هو تسلسل من القيم الحرفية , المسافات , علامات الرجوع .
True	قيمة منطقية , عادةً تمثل بـ 1 . تعلن في الكود ان $1 + 1 = 2$ سيكون مثال على جملة صحيحة True
Undefined	متغير لا يمسك بأى قيمة
Value	قيمة المتغير هي البيانات التي يحملها أو يمسك بها
Variable	كائن مسمى , وهو عامةً يعتبر جزء مسمى من الذاكرة يستخدم ليحمل قيمة أو يمسك قيمة
VBScript	لغة مترجمة فهي على نحو تام تعتمد على Visual Basics وهي لغة تستخدم عادةً مع ASP
Version	رقم (أو أرقام) مرتبط بالكود أو بالبرنامج يحدد عمره . آخر رقم إصدار . فهو أحدث كود أو برنامج .
Visual Basic	لغة برمجة لمايكروسوفت مشابهة لـ Basic
Visual C++	تحسين مايكروسوفت من C++
While Statement	جمل تكرارية تحتوى على طالما ان الشرط متحقق
Whitespace	الحروف الممثلة فقط بواسطة المسافات التي تاخذ في الشاشة او الصفحة . معظم الامثلة الشائعة هي () , سطر جديد ($\backslash n$) , مسافة إنتقال $\backslash t$
Widget	مصطلح يستخدم لوصف كائن سرى , كائن ليس لديه إسم أو شئ نسيت ان تضع له إسم .
Zero	مصطلح آخر يستخدم للكلمة Null

B

Web Resources

موارد الإنترنت

أصبح الإنترنت المورد الهائل لكثير من الأشياء — ففكرة إمتلاك مكتبة فى كل منزل تكاد تكون حقيقة واقعة . مع ذلك , فالمشكلة أنه ليس هناك موظف مكتبة ليرشدك داخل هذه المكتبة , ويصل بها إلى مستوى معين من الجودة فى المواد المتاحة .

هذا الملحق يدرج المواقع المفضلة لدى على الإنترنت المرتبطة بالبرمجة فى بعض الطرق . لقد تم تجميعهم فى تصانيف ولكن تم إدارجهم فى هذا التصنيف دون ترتيب معين .

إستمع !

معظم تراخيص البرامج المجانية مدرج فى هذا الملحق فقط تتبع استخدام المنتجات الغير تجارية . من فضلك قم بفحص الترخيص والاتفاقيات بحرص . وإذا كان لديك شك فى إستخدام برنامج سواء كان إستخدامه قانونياً , إستشر الصانع له أو المؤلف لهذا البرنامج .

Programming Tools

ادوات البرمجة

Site	الوصف
www.ultraedit.com	<p>برنامج مشاركة Shareware من أفضل محرري النصوص المتاحة . فهو يقدم العديد من المزايا التي يجدها المبرمج مفيدة له في عمله وتشمل : أرقام الأسطر دعم ملفات كبيرة (أكبر من 4 GB !) تراجعات متعددة فاحص قواعد إملائية 100.000 تسليط الضوء على قواعد بناء الجملة دعم شامل للماكرو أو المختزلات إتفاقية نقل الملفات للعملاء المدمجة به FTP دعم المشاريع / مساحة العمل محرر ساحر بحث / إستبدال قوى أيضاً المتاح هو تطبيق جديد يسمى UltraCompare . هذه الأداة هي أداة مقارنة لملفات ويمكن ان تعمل مع الملفات من نوع ملفات نصوص ملفات نظام أرقام ثنائي مجلدات UltraCompare تأتي مجهزة مع قوة مدمجة وقدرات أداة ملفات .</p>
www.textpad.com	<p>برنامج مشاركة قوى جداً , وملئ بالوظائف , ومدعماً لملفات عدة , ويقدم مصحح ومدقق إملائي .</p>
www.crimsoneditor.com	<p>برنامج مجاني قوى جداً ومحرر نصوص محترف لـ windows . صغير وسريع ووجيز مفتقر للمزايا . فهو يقدم المزايا التالية : مدقق إملائي تراجع متعدد مرفق به برنامج Macro أو المختزلات</p>
www.jedit.org	<p>مجاني محرر نصوص قوى ولديه ميزة العمل على أنظمة تشغيل متعددة Mac OS X OS/2 Unix VMS Windows بعض من المزايا التي يقدمها تشمل : ماكرو مدمج به مزايا إضافية يمكن ان تضاف إليه من خلال Plug-ins تسليط الضوء على أكثر من 80 لغة . دعم لعدد هائل من ترميز الحروف .</p>

Java Tools

المصطلح	الوصف
www.borland.com/products/downloads/download_jbuilder.html	مجاني مفسر لبيئة تطوير Java قوى , وسهل الإستخدام وسريع مؤسسة JBuilder تعمل على الانظمة التالية : Windows Solaris Linux
www.JCreator.com	مجاني بيئة تطوير Java مدمج معه المزايا التالية : قوالب المشروع ادارة المشروع تكملة الكود تلقائياً تسليط الضوء على القواعد الإملائية مصحح يعمل على الأنظمة التالية : Windows XP Windows 2000 Windows ME Windows 9x
www.netbeans.com	مجاني سريع , وسهل الإستخدام بيئة تطوير Java متكاملة , التي تعمل مع معظم انظمة تشغيل . يمكن ان يستخدم لإنشاء تطبيقات Java , خدمات إنترنت وتطبيقات هاتف .
Java.sun.com/j2se	مجاني هذا هو رابط مجاني لبيئة تطوير لشركة Sun للغة Java إضافة إلى تقديم بيئة تطوير قوية , وايضاً توفر مفسر سريع وسهل الإستخدام . هذا الموقع أيضاً يحتوى على مصفوفة واسعة من المستندات وامثلة الكود هذا الموقع جيد ليزار ويطلع عليه .
www10.software.ibm.com/developerworks/opensource/jikes/	مجاني هو مفسر جافا تم صنعه بواسطة IBM يدعم الكثير من أنظمة التشغيل المتنوعة : Windows Solaris Sparc Linux OS/2 AIX
hsqldb.sourceforge.net/	مجاني هو مصدر مفتوح مجاني قاعدة بيانات SQL إنشئ في جافا

Java Sites

مواقع جافا

الموقع	الوصف
java.sun.com	هو المورد الرئيسي لكل ما يخص أشياء Java
www.javaworld.com	أخبار Java , موارد , روابط , اكواد , منتديات , المزيد
www.javalobby.org	مورد على الإنترنت لمستخدمي Java في كل مكان
www.blackdown.org/java-linux.html	مورد لمستخدمي Linux المرادين الدخول إلى جافا
www.mindview.net/Books/TIJ	التفكير في جافا thinking in java هو كتاب الكتروني مجاني كله عن الجافا

C++ Tools

ادوات C++

الموقع	الوصف
upp.sourceforge.net	مجاني Ultimate++ مختلف عن بيئات تطوير C++ لأنه بيئة تطوير Windows/ Linux . الفكرة أنه يعمل مع بيئة متوافقة مع أنظمة التشغيل لتقليل التكلفة . Ultimate ++ تعتمد على : محرم مصدر نظام إداري مصحح مصمم تخطيط خارجي مصمم صور محرم لغة
www.bloodshed.net/devcpp.html	مجاني مملوء بمزايا بيئة تطوير متكاملة للغات البرمجة C/C++ مزايا بيئة التطوير تشمل : تكملة الكود . مصحح تسليط الضوء على القواعد الإملائية مدير ادوات دعم طباعة بحث / إستبدال قوى قائمة بالدوال
www.borland.com/products/downloads/download_cbuilder.html	مجاني (مفسر فقط) مبدئي , ولكن مملوء بمزايا مفسر Command-Line ليس فيه تكلف , ولكن سريع . ويشتمل على ادوات تصحيح
www.cs.virginia.edu/~lcc-win32	مجاني كل ما تحتاجه لتبدأ في برمجة C++: مولد كود

	بيئة تطوير متكاملة مستندات مساعدة ودليل مستخدم
sunset.backbone.olemiss.edu/~bobcook/eC	مجاني مفسر C++ مع اختلاف — أن هذا أحد المصممين خصيصاً للإستخدام في المدارس الثانوية و كليات الصغيرة
www.digitalmars.com	مجاني مفسر C / C++ . مناسب لـ : Win32 Win16 DOS DOS32 كل شيء تحتاجه لتبنى تطبيقات C++ المزيد من مصدر الكود والمستندات المشتمله به .

C++ Sites

مواقع C++

الموقع	الوصف
www.cprogramming.com	موقع دروس C++ ممتاز
www.glenmccl.com/tutor.htm	موقع اخر دروس جيد
devcentral.iticentral.com/articles/C++/default.php	موقع اخر به دروس لمفاهيم متقدمة في C++
bdn.borland.com/cpp/	مجتمع Borland C++
www.developerfusion.com/forums/forum-26	منتدى
www.cuj.com	جريد خاصة بـ C/C++ User's Journal Web site

BASIC Tools

ادوات بيسك

الموقع	الوصف
www.basic4gl.net	<p>مجاني</p> <p>لغة برمجة Basic لنظام التشغيل Windows.</p> <p>سهلة التعلم وسهلة الاستخدام ويمكن ان تستخدم لإنشاء ألعاب , ثلاثية الأبعاد , وخدمات ومرفقات</p>
www.maxreason.com/software/xbasic/xbasic.html	<p>مجاني</p> <p>مفسر Basic متقدم مع اداة تطوير واجهة رسومية قوية .</p> <p>كانت تجارية ولكن الان مجانية .</p> <p>تنشأ أداة البرمجة هذه مصدر كود مناسب لـ :</p> <p>Windows Linux Unix</p>
www.basicguru.com/rapidq/	<p>مجاني</p> <p>Rapid-Q هي عبر لغة البرمجة Basic وقادرة على إنتاج برامج رسومية وبرامج تعمل على DOS</p> <p>تدعم مباشرة :</p> <p>MySQL DirectX Direct3D Sockets Some COM Component/object programming والمزيد</p>
www.nicholson.com/rhn/basic/	<p>مجاني</p> <p>صغير ولكن ومتعدد الإستعمال يمكن أن يستخدم مع الكثير من برنامج Basci . مع القليل أو عدم التعديل .</p> <p>الإصدارات المتاحة لـ :</p> <p>Windows Mac OS Linux Sun OS SGI</p>
www.yabasic.de	<p>مجاني</p> <p>يعمل على أنظمة Windows</p>

www.libertybasic.com	مشاركة يمكن ان يستخدم لإنشاء ألعاب , تطبيقات أعمال تجارية , والمزيد .
www.freebyte.com/programming/compilers/envelop.html	مجاني بسيط , أداة برمجة Basic قوية , سهلة وسريع الإستخدام

BASIC Sites

الموقع	
www.qb4all.com	كل شئ عن من يريدون ان يبدوا برمجة بـ Quick Basic
www.programmersheaven.com/zone6	موقع دروس
www.codepedia.com/1/BeginnersGuideToBasic	موقع لدروس مبتدئين لـ Basic

Web Scripting Languages

الموقع	الوصف
asp2php.naken.cc	مجاني ASP إلى PHP , سريع , سهل .
www.php.net	منافسة لـ ASP وهى لبرمجة مواقع الإنترنت
www.perl.org	لغة برمجة مواقع إنترنت أخرى
www.zope.org	تطبيق خادم مفتوح المصدر , ابناء أنظمة إدارة المحتوى , شبكات داخلية , وتطبيقات مخصصة

CD Burning

حرق الإسطوانة

الموقع	الوصف
www.nero.com	برنامج مشاركة يقدم العديد من المزايا : دعم نطاق واسع لتسجيلات الإسطوانات BURNProof recorders دعم نسخ الإسطوانات من نوع : (ISO, VideoCD, AudioCD, Bootable CD, UDF) نسخ احتياطي تلقائي دعم للإسطوانات الوهمية دعم لطباعة البطاقات والعلامات على الإسطوانات Nero يقوم بعمل كل ماتريد ان تفعله مه برمج حرق الإسطوانات
www.roxio.com	منتج تجارى أداة حرق إسطوانات قوية إنشاء انواع العديد من مختلف أنواع الإسطوانات السريعة والسهلة .

Compression Tools

ادوات ضغط

الموقع	الوصف
www.winzip.com	<p>مجاني</p> <p>اداة ضغط قوية سهلة الإستخدام وتستخدم خوارزمية الضغط Zip المشهورة</p> <p>ويشمل على المزايا التالية :</p> <p>سرعة الضغط / وفك الضغط للملفات والمجلدات</p> <p>ينشئ تطبيق إعداد تثبيت</p> <p>دعم للملفات الكبرى</p> <p>تشفير AES لحفظ البيانات لك .</p> <p>دعم ملفات إنترنت</p> <p>متكامل مع windows</p> <p>دعم لتقسم الملف</p>
www.winace.com	<p>مشاركة</p> <p>اداة ضغط أخرى قوية وملينة بالمزايا .</p> <p>تشمل على المزايا التالية :</p> <p>دعم لخوارزميات الضغط التالية :</p> <p>ACE, ZIP, LHA,MS-CAB, JAVA JA</p> <p>يمكن ان يقوم بفك تشفير الملفات التالية :</p> <p>ACE, ZIP, LHA,MS-CAB, RAR, ARC, ARJ, GZip, TAR, ZOO, JAR</p> <p>دعم للغات متعددة</p> <p>ينشئ تطبيق تثبيت قوى .</p> <p>يدعم الإسطوانات والأقراص</p> <p>يدعم إنشاء الأرشيف الذاتي [الذى لا يحتاج إلى وجود البرنامج مثبتاً]</p> <p>حماية بكلمة سر</p> <p>يمكن إصلاح الملفات المدمرة ZIP.</p> <p>دعم للسحب والإسقاط</p> <p>عارض سريع لأنوا الملفات الشائعة</p> <p>تحسينات الإرشيف</p>
www.pkzip.com	<p>مشاركة</p> <p>اداة ضغط أصلية , يقدم هذا الموقع إلى أدوات مختلفة متنوعة لها مزايا مختلفة</p> <p>الضغط</p> <p>الأمان</p> <p>مضاد الفيروسات</p> <p>دعم جيد لأنظمة التشغيل</p> <p>Windows, DOS, VM, VSE, Unix,iSeries, zSeries</p>

www.pb-sys.com	مجاني TaskZip هي أداة نسخ احتياطي , سهلة الإستخدام
www.ultimatezip.com	مشاركة منخفضة الثمن , ولكن أكثر وظيفياً وأداة ضغط منافسة . فهو يبدو ك Windows Explorer . ومما يجعله سهل الإستخدام . المزايا المشتملة عليها : يدعم تنسيق : ZIP, RAR, ACE, BH, CAB, JAR, LHA (LZH), GZIP, TAR, BZIP2, ARC, ARJ, XXE, UUE يدعم الملفات الكبيرة (أكبر 4 GB) يدعم تشفير AES تقسيم الملفات ميزة نسخ احتياطي ميزة إصلاح ZIP
www.7-zip.org	مجاني (التبرع مقبول والترخيص يحتاج للدعم) أداة أرشف ملفات مع معدل ضغط عالي المزايا تشمل على : القدرة على فك ضغط الملفات بدون وجود البرنامج مثبت على الحاسب مدمج مع نظام التشغيل windows
www.stuffit.com	مشاركة أداة ضغط قوية , بدأت الحياة كأداة MAC ولكن الجديد منها يدعم أنظمة تشغيل مختلفة . تشمل المزايا : إنشاء أرشيف . ملفات مضغوطة , وملفات مرمزة . تسمح لك بتصفح المحتويات . القدرة على البحث عن الملفات ميزة البريد الإلكتروني و نقل الملفات للعميل من خلال FTP للإنترنت.

Miscellaneous Tools

ادوات متنوعة

الموقع	الوصف
www.freebyte.com/hjinstall	<p>برنامج مجاني يعمل على :</p> <p>Windows 95 Windows 98 Windows ME Windows NT 4.0 Windows 2000 Windows XP</p> <p>ويمكن أن يستخدم لإنشاء انواع مختلفة من البرامج :</p> <p>Internet-distributed CD-Rom/DVD Single-floppy-disk Multi-floppy-disk Network installations</p>
www.gregorybraun.com/iconEx.html	<p>مشاركة</p> <p>برنامج Icon Extractor برنامج مرفق وصغير يحمل ويعرض كل الإيقونات الموجودة في الملفات ويسمح لك بحفظها كملف بإمتداد ico .</p> <p>ويمكنك هذا الملف والتعديل فيه بواسطة أى برنامج آخر .</p>
www.icongrabber.com-http.com	<p>مشاركة</p> <p>برنامج Icon Grabber خاص ايضاً بالبحث عن الإيقونات وعرضها لك من كل ملفات الحاسب .</p>
www.mirekw.com/winfreeware/mwsnap.html	<p>مجاني</p> <p>برنامج MWSnap صغير , برنامج قوى لقص الصور وتقطيعها للجزء المحدد من الشاشة .</p>
www.progency.com/other.html#screenrip32	<p>مجاني</p> <p>ScreenRip32 هو برنامج مرفق لقص صور من الشاشة بطرق متعددة .</p>
www.axialis.com/axcursors/	<p>مشاركة</p> <p>Axialis AX-Cursors 4.5 هو برنامج قوى لتحرير المؤشر كما أنه سهل الإستخدام ويستخدم في إنشاء وتعديل وإنتاج مؤشرات .</p>

Miscellaneous Sites

الموقع	
www.kingsley-hughes.com	موقع تعليمي , به المزيد من معلومات عن البرمجة , والحيل والخدع والتلميحات .
msdn.microsoft.com	شبكة مطوري مايكروسوفت — مورد ضخم للمطورين من كل نوع .
www.w3schools.com	دروس لكل ما يتعلق بالإنترنت : HTML XHTML XML CSS JavaScript VBScript SQL ASP PHP المزيد , موقع ممتاز .
www.dcs.ed.ac.uk/home/mxr/gfx/utls-hi.html	كمية من المعلومات في تنسيقات الصور الموقع يشمل : كود برمجة أسئلة وأجوبة روابط
http://www.chami.com/tips	كمية كبيرة من تلميحات البرمجة للغات البرمجة المتنوعة . موقع ممتاز
www.webopedia.com/quick_ref/fileextensions.asp	تنسيق بيانات , وإمتداد ملفات
condor.stcloudstate.edu/help/faqs/fileformats.html	كمية كبيرة من تنسيقات الملفات المختلفة
www.filext.com	محرك بحث لإمتدادات الملفات . سيجد لك نوع الملف ممثل بامتداد الملف او البرنامج الذي تستخدمه .

Index

SYMBOLS & NUMERICS

&& (and) operator, 171
' (apostrophe) in VBScript, 98, 99, 100
= (assign value to variable), 235
== (check variable against value), 235
***/** (close multiline comment), 102, 103
- (collapse item), 342
[] (curly braces), 171, 237
**** (double backslash), 355
+ (expand item), 342
// (forward slashes)
 in C++, 103
 in JavaScript, 101
<< (insertion operator), 150, 151
***** (multiplication) operator, 202
!= (not equal to) operator, 171
/* (open multiline comment), 102, 103
|| (or) operator, 171
; (semicolon)
 in C++, 150, 151, 238–239
 in JavaScript, 354
_ (underscore)
 in name of file, 369
 in name of variable, 121, 122–123
7-Zip compression tool, 443

A

academic use, limits on, 92
Active Server Pages (ASP), 22
addictive, programming as, 24
Additions/Changes Checklist, 289
airflow, 73
Allen, Paul (programmer), 3
alphanumeric character, variable to hold, 110
ambiguity, removing
 from code, 238
 from output screen, 206–211
American Standard Code for Information Interchange (ASCII),
 6, 57–64
Analytical Engine, 1–2
and (&&) operator, 171
annotating output, 259–260

apostrophe (') in VBScript, 98, 99, 100
appending file, 319–320
applications. *See* software applications
Arch version control system, 377
archiving file, 265
arithmetic operators
 description of, 65
 in name of variable, 119
arrays
 description of, 180–181
 exercises, 183–184
 multidimensional, 182–183
 two-dimensional, 182
ASCII (American Standard Code for Information Interchange),
 6, 57–64
ASP (Active Server Pages), 22
assigning value to variable (=), 235
assignment operators, 66
Atari BASIC, “Hello, World!” code in, 7
author, Web site of, 446
Axialis AX-Cursors 4.5 cursor-editing and -managing program,
 445

B

Babbage, Charles (inventor), 1–2
backing up
 encoded script, 403
 source code, 299
 Windows Registry
 in its entirety, 332–338
 subkeys, 330–332
 in Windows XP 330–331
Backup Job Information dialog box, 336
Backup or Restore Wizard, 332, 333, 339
Backup Progress dialog box, 338
BASIC
 “Hello, World!” code in, 7
 Web sites related to, 439–440
beta release, 297–298
binary
 bit grouping, 47–50
 description of, 44
 interpreting, 44–46

- binary (continued)**
 - large numbers, 46–47
 - purpose of, 54
- binary data, 40**
- binary language, 5**
- binary math**
 - overview of, 50–51
 - Windows Calculator, using, 51–54
- bit, 44, 47**
- bit grouping, 47–50**
- B-language, 3**
- Boole, Charles (symbolic logic system inventor), 2**
- Boolean value, as variable, 106**
- Borland C++ compiler**
 - in action, 93–94
 - Digital Mars compared to, 381–386
 - downloading and installing, 92–93
 - error handling, 388–391
- breaking up code, 31–34**
- browser**
 - testing JavaScript examples in, 108
 - as tool, 88
- buffer underrun protection feature, 413**
- burning disc**
 - CD/DVD for, 419–420
 - drive for, 412–413
 - labeling and, 420
 - requirements for, 412
 - software for, 414–419
- Burnproof feature, 413**
- Bush, Vannevar (inventor), 2**
- button. See also radio button**
 - description of, 268–269
 - example of, 277–278, 280–281
- byte, 48**
- bytecode, 89, 393**

C

- C++ Builder, 232**
- C programming language**
 - commercial programming and, 20
 - development of, 3
 - "Hello, World!" code in, 7
- C++ programming language**
 - advantages of, 79
 - arrays, 180–183
 - calculator, creating, 163–171
 - code
 - for temperature conversion program, 196–204
 - for temperature conversion program, improving, 204–212
 - code structure, 138–141
 - commenting code, 103–104
 - commercial programming and, 20–21
 - compiling code, 141–149
 - conditionals, 157–171
 - description of, 91–92
 - functions, 36, 38, 141, 149–156
 - "Hello, World!" code in, 7, 10
 - If statement, 158
 - logic errors, 235–236
 - loops, 174–178
 - overview of, 138
 - runtime errors, 230–232
 - statement in, 35

- text-based interface, 251–254
- Web sites related to, 437–439

calc() function, 264–265

calculator application

- annotating output, 259–260
- breaking problem into smaller steps, 194–195
- coding phase, 196–204
- confirmations, 267
- confirming exit from, 260–261
- creating, 163–171
- help, adding, 261–266
- improving code, 204–212
- program overview, including, 254–256
- prompting user for inputs, 256–259
- requirements for, 186–191
- research for, 191–194

calling

- code, 34
- compiler, 144
- function, 154–156

capitalization scheme for naming variables, 122

CD

- advantages of, 410
- burning, 412–419
- description of, 409
- disadvantages of, 410–411

cd (change directory) command, 147

chair, 73, 360

change directory (cd) command, 147

char variable, 164–167

character

- alphanumeric, variable to hold, 110
- ASCII, 57–60
- description of, 57
- extended character set, 60–62
- keyboard input and, 167–168
- in string, 124–125

cheap disc, using for burning, 419

check box

- description of, 271–272
- example of, 282

checking

- for duplicate files, 317
- preceding statements to spot errors, 237
- variable against value (==), 235

cin instruction, 161, 165–166

client requirements

- clarifying, 186–191
- reviewing, 211–212

client-side programming, 22–23

closing

- HTML file, 108
- multiline comment (*/), 102, 103

COBOL

- development of, 3
- "Hello, World!" code in, 8

code. See also programming stage of project; structure of code

- calling, 34
- commenting
 - C++, 103–104
 - encoding and, 403
 - exercises, 100–101, 104
 - JavaScript, 101–103
 - overview of, 97–98
 - programming stage of project and, 294–296
 - spotting errors and, 237

- temperature conversion program, 198
- VBScript, 98–101
- compiled
 - alternatives to, 397–403
 - description of, 41, 42
 - runtime errors and, 230–232
 - version control and, 376–377
- compiler, 11
- compiling
 - benefits of, 394–397
 - C++, 141–149
 - debugging and, 397
 - functionality and, 396
 - intellectual property protection and, 395
 - process for, 379–381
 - security and, 396–397
 - speed and, 395–396
 - spotting errors and, 226
 - temperature conversion program, 203, 211
- copying, 42
- development environment and, 11
- in different languages, 6–10
- functions and procedures, 36–37, 38–39
- improving for temperature conversion program, 204–212
- inefficiency in, 285
- JavaScript
 - to display text string, 125
 - processing inputs, 130–132
 - string manipulation, 126–130
 - variable manipulation, 132–135
 - variables in action, 109–118
- logic of, 198–199
- making hard to follow, 397–403
- reading
 - breaking up code, 31–34
 - overview of, 27
 - top-down, 28–30
- reusing, 32–33
- saving, 299
- statement, 35–36, 37–38
- storage of, 39–42
- syntax of, 39
- testing
 - in browser, 108
 - to spot errors, 239
- tweaking, 300
- code-writing process, 6**
- collapsing item (-), 342**
- Command Prompt, 143**
- command window**
 - opening, 143–144
 - view of, 88
- commenting code**
 - C++, 103–104
 - encoding and, 403
 - exercises, 100–101, 104
 - JavaScript, 101–103
 - overview of, 97–98
 - programming stage of project and, 294–296
 - spotting errors and, 237
 - temperature conversion program, 198
 - VBScript, 98–101
- commercial/application programming**
 - beta and preview strategy for, 298
 - description of, 20
 - free tools and, 23
- comparison operators, 66–67**
- compiled code**
 - alternatives to, 397–403
 - description of, 41, 42
 - interpreted code compared to, 395–396
 - runtime errors and, 230–232
 - version control and, 376–377
- compiler**
 - Borland C++
 - Digital Mars compared to, 381–386
 - downloading and installing, 92–94
 - calling, 144
 - description of, 11, 25
 - error handling and, 387–391
 - executable file and, 379
 - language and, 391–394
 - license agreement, 405
- compiler error**
 - description of, 214
 - in large segment of code, 221–226
 - opening curly brace missing, 217–218
 - opening parenthesis instead of curly brace, 215–2
 - semicolon missing, 220–221
 - too many < in code, 219–220
 - unterminated string, 218–219
 - warnings, generating, 226–228
- compiling code**
 - benefits of, 394–397
 - C++, 141–149
 - debugging and, 397
 - functionality and, 396
 - intellectual property protection and, 395
 - process for, 379–381
 - security and, 396–397
 - speed and, 395–396
 - spotting errors and, 226
 - temperature conversion program, 203, 211
- compression tools, 83–84, 442–444**
- computer**
 - for programming, 24
 - workspace, creating, 362–363
- computer science, degree in, 24**
- concatenating strings in JavaScript, 126–130**
- conditionals**
 - choice of folder and, 310–311
 - exercises, 172–173
 - overview of, 157
- confirmations, adding to interface, 267**
- confirming exit from application, 260–261**
- converting**
 - ASCII to binary string, 63
 - binary string to ASCII, 64
- copyright issues, 423**
- copyright notice, 403**
- cost of programming tools, 23**
- Courier New font, 73**
- cout instruction**
 - cIn and, 165–166
 - cIn compared to, 161
 - compiler and, 381
 - description of, 150, 151
- .cpp file extension, 141**
- CreateFolder method, 310–311**
- Crimson Editor (text editor), 434**
- curly braces ({}), 171, 237**

D

data storage, 39–42

debugging. See also errors

- breaking up code and, 32
- compiling code and, 397
- poor planning and, 286
- reproducing error before fixing, 300
- testing after, 300
- top-down processing and, 30

decimals, in binary, 44

decision making, 157–158

declaring

- array, 181, 182
- path for file, 309
- variable, 109, 160

defining variable, 109–110

deleting

- file, 325
- folder, 325
- registry keys, 354

design errors, checking for, 297

desk, 73

desktop, saving files to, 363

development environment, 11

dialog box, example of, 277–279

Difference Engine, 1, 2

Digital Mars compiler

- Borland C++ compiler compared to, 381–386
- error handling, 388–391

distraction, minimizing, 360

distribution

- full-time job versus hobby, 423
- license agreement and, 405
- physical, 405–412
- supported versus unsupported, 424–425
- types of, 405
- virtual, 421–424

DJGPP compiler, 381

Do While loop, 177–178

documenting

- errors, 294
- ideas, 287–289

double backslash (\\), 355

doubleword, 50

downloading

- Borland C++ compiler, 92
- Java SDK and VM, 89–90

drop-down menu

- description of, 275
- example of, 277–278

DVD

- advantages and disadvantages of, 412
- burning, 412–419

DWORD value, 344

E

editing. See also text editor

file

- appending file, 319–320
- overview of, 317–319

Windows Registry

- in JScript, 354–355
- overview of, 329–330
- in VBScript, 349–354

else statement, 162

e-mail support, 424

encryption compared to encoding, 399

endl insertion operator, 150, 151, 160, 161

ergonomic keyboard, 72

Error Documenting, 294

error handling and compiler, 387–391

error messages

- Borland C++ compiler, 388, 390
- Digital Mars compiler, 388–389
- interpreting, 216–217
- terminated string, 219

errors. See also debugging

- compiler type, 214–228
- documenting, 294
- as expected, 213–214
- logic type, 232–236
- reproducing before fixing, 300
- runtime type, 228–232
- spotting
 - checking preceding statements, 237
 - commenting and, 237
 - keeping layout clear, 237
 - keeping track of variables, 239–244
 - reading each line after pressing enter, 236
 - removing ambiguity in code, 238
 - semicolons, 238–239
 - testing code and, 239
- types of, 214
- warnings compared to, 226–228

escape characters in JavaScript and JScript, 355

executable file

- compiler and, 379
- .exe extension, 11, 146, 150

exercises

- arrays, 183–184
- commenting code
 - C++, 104
 - JavaScript, 102–103
 - VBScript, 100–101
- conditionals, 172–173
- functions, 156–157
- loops, 178–180
- variables, 124

expanding item (+), 342

F

FAQ (Frequently Asked Questions), 424

fast disc, using for burning, 420

feature creep, 286

features

- adding, 299–300
- missing, 286, 297

file

- appending, 319–320
- checking for duplicate, 317
- .cpp extension, 141
- creating, 306–309
- deleting, 325
- editing, 317–319
- .exe (executable), 11, 146, 150
- executable, and compiler, 379
- .ini (initialization), 328
- iostream.h, 139
- .java extension, 391

- .js extension, 355
- .jse extension, 403
- multiple, creating, 311
- naming conventions, 368–370
- opening for reading, 320
- organizing
 - folder contents note, 366–368
 - language, grouping by, 364–365
 - overview of, 359
 - project, grouping by, 365–366
- prompting user for name of, 314–317
- Read method, 323–324
- ReadAll method, 321–322
- reading, 321
- ReadLine method, 322–323
- .reg (registration entries), 339
- saving
 - to desktop, 363
 - frequency of, 362
 - principles of, 303–304
- size of, and compiler, 382, 383, 387, 393–394
- string.h, 139, 151
- .txt (text), 366
- .vbe extension, 403
- .vbs extension, 307
- .zip extension, 83
- file life cycle, 304–305**
- find and replace feature, 397**
- flexibility of code, 33**
- float variable, 164–166**
- floating point number**
 - integer compared to, 196
 - as variable, 105–106
- floppy disk**
 - advantages of, 406
 - disadvantages of, 408–409
 - drive, 408
 - examples of, 407
- folder**
 - conditionals and choice of, 312–313
 - contents note, 366–368
 - creating, 310–311
 - creating file and, 309
 - deleting, 325
 - grouping
 - by language, 364–365
 - by project, 365–366
 - prompting user for name of, 314–317
 - variables and, 313
- fonts, 73–75**
- For loop, 174–176**
- formatting message box, 244**
- FORTTRAN language**
 - development of, 3
 - “Hello, World!” code in, 8
- forum, support using, 424–425**
- forward slashes (/)**
 - in C++, 103
 - in JavaScript, 101
- free software, beta and preview strategy for, 298**
- freelance programmer, 14**
- freeware, 3**
- Frequently Asked Questions (FAQ), 424**
- functions**
 - C++, 36, 38, 141, 149–156
 - calling, 154–156

- commenting, 296
- defining, 140
- description of, 36–37, 38–39
- exercises, 156–157
- main(), 150, 152–156, 263–264
- overview of, 149–156
- planning project and, 292
- removing unused, 300

G

Gates, Bill (programmer), 3

general tools

- SnagIt, 86–87
- UltraCompare, 84–86
- Windows Notepad, 80–82
- Winzip, 83–84

GNU General Public License, 423

GNU Revision Control System, 377

grouping folders

- by language, 364–365
- by project, 365–366

H

halfword, 48–49

hard drive, storage capacity of, 40

hardware and Windows Registry, 328

header file, 139–140, 151

hello function, 152–156

hello2 function, 152–156

“Hello, World!”

- code examples, 7–10
- origin and history of, 10

help, adding to interface, 261–266

help file, creating, 301

helpsystem() function, 265–266

hexadecimal number system

- interpreting, 55–56
- overview of, 54–55
- Windows Calculator and, 57

history of programming, 1–4, 77

HJ-Install freeware installer, 444

hobby, programming as, 78–79, 423

hobby project, 17–18

Hollerith, Herman (inventor), 2

Hopper, Grace Murray (programmer), 3

HTML comment tags, 100, 107

Hungarian Notation, 123–124

I

IBM, 2, 3

Icon Extractor utility, 445

Icon Grabber utility, 445

IDE, 25

idea

- documenting, 287–289
- maturing time for, 289
- types of, 286–287

If statement

- description of, 158
- temperature conversion program, 199, 207–211

infinite loop, 176, 234–245

.ini (initialization) file, 328

Initial Checklist, 287–289

initial instruction for use for temperature conversion program, 204–205

initialization file (.ini), 328

initializing variable, 105

inputs

- buttons and, 268–269
- folder and filenames, 314–317
- processing, 130–132
- prompting user for, 256–259
- from user for temperature conversion program, 188, 195, 196–197

insertion operator

- <<, 150, 151
- endl, 150, 151, 160, 161

installing

- Borland C++ compiler, 92–94
- Digital Mars compiler, 381
- Java SDK and VM, 90–91
- Microsoft Script Encoder, 399

integer. *See also* number

- code interpreting as string, 134–135
- floating-point number compared to, 196
- replacing with text string, 114
- as variable, 106

integrating variables with test output, 209–210

intellectual property, protection of, 395

interface

- annotating output, 259–260
- appalling, 250–251
- buttons, 268–269
- check boxes, 271–272
- confirmations, 267
- confirming exit from application, 260–261
- description of, 247, 250
- drop-down menus, 275
- goal of, 275
- help, adding, 261–266
- importance of, 251
- menus, 270–271
- Microsoft OneNote 2003, 280–282
- Microsoft Word, 247–250
- multiline text boxes, 274
- problem solving and, 195
- program overview, including, 254–256
- prompting for input, 256–259
- radio buttons, 272–273
- simple, 250
- single-line text box, 273–274
- text-based, 251–254
- Windows Notepad, 276–280

Internet. *See also* Web sites

- virtual distribution and, 421–424
- Web programming and, 21–22

Internet Explorer

- testing JavaScript examples in, 108
- as tool, 88

interpreted code compared to compiled code, 395–396

interpreted language and runtime errors, 228, 232

interpreter, 6, 11

interpreting

- binary, 44–46
- error messages, 216–217
- hexadecimal number system, 55–56

iostream.h file, 139

J

Java

- advantages of, 79
- commercial programming and, 20, 21
- compiler and, 391–394
- description of, 89
- “Hello, World!” code in, 8
- Software Development Kit (SDK), 89
- Web sites related to, 435–437

.java file extension, 391

Java Virtual Machine (VM) runtime environment, 89, 393, 394

JavaScript

- advantages of, 79
- client-side programming and, 23
- code
 - to display text string, 125
 - processing inputs, 130–132
 - string manipulation, 126–130
 - variable manipulation, 132–135
 - variables in action, 109–118
- commenting code, 101–103
- escape characters in, 355
- “Hello, World!” code in, 9, 10
- parentheses in, 355
- plain text and, 41
- template, creating, 107–108
- tools needed for, 94–95

JBuilder Foundation IDE/compiler, 435

JCreator IDE, 435

JEdit text editor, 434–435

JediVCS (Version Control System), 377

.js file extension, 355

JScript (Microsoft)

- description of, 349
- Registry editing, 354–355

.jse file extension, 403

K

keeping code layout clear, 237

keeping track of variables, 239–244

key, finding, 341–343

keyboard

- buttons and, 268–269
- choosing, 71–73
- custom, 362
- height of, 360
- input from, 167–168
- menu and, 271

keyword

- in name of variable, 120
- var, 109, 117–118
- void, 140

L

labeling

- button, 269
- check box, 272
- disc, 420–421
- radio button, 273
- text box, 274

language

- changes in, 25

- choosing
 - to learn, 75–76
 - for project, 293
- code and, 6–10
- compiler and, 391–394
- of computer, 4–5
- grouping folders by, 364–365
- knowledge of as transferable, 25
- platform-independent, 89
- scripting, tools needed for, 94–95
- Web scripting sites, 441

layout

- for folder contents note, 366–367
- of Windows Registry, 329

learning programming

- career programmer and, 14–16
- foreign language learning and, 76
- for fun, 18
- as hobby, 78–79
- knowledge as transferable to other language, 25
- to make contribution, 18
- overview of book, 79–80
- problem solving and, 16–17
- reasons for, 13–14
- school/college setting, 77
- time requirement, 24
- work-based setting, 77–78

learning/academic programming, 21

legal advice, 423

license agreements, 405

licensing software, 423

life cycle of file, 304–305

lighting, 73

line break character, 244

line numbering and text editor, 232

logic errors

- description of, 232–236
- testing for, 296

logical operators, 67–68

loops

- description of, 235
- Do While type, 177–178
- exercises, 178–180
- For type, 174–176
- infinite, 176, 234–245
- semicolon and, 238–239
- While type, 176–177

M

main() function

- calculator application, 263–264
- description of, 150
- example of, 152–156

manipulating

- stored data, 114–115
- strings, 126–130
- variables, 132–135

Mathematica, “Hello, World!” code in, 9

Mead, Ian (programmer), 16

memory

- breaking up code and, 33
- top-down processing and, 28–29

menu

- description of, 270–271, 275
- example of, 276, 277–278, 280

message box, formatting, 244

methods

- CreateFolder, 310–311
- Read, 323–324
- ReadAll, 321–322
- ReadLine, 322–323
- RegDelete, 353–354
- RegRead, 352–353
- RegWrite, 351

Microsoft

- Developer Network Web site, 446

JScript

- description of, 349
- Registry editing, 354–355

OneNote 2003 interface, 280–282

Script Encoder, 398–403

VBScript

- advantages of, 79
- appending file, 319–320
- checking for duplicate files, 317
- client-side programming and, 23
- commenting code, 98–101
- creating file, 307–309
- creating folder, 310–311
- deleting file, 325
- deleting folder, 325
- editing file, 317–319
- folder, choice of and conditionals, 312–313
- folder, variables and, 313
- haphazard example of, 228–229
- logic error, 232–234
- multiple files, creating, 311
- opening file for reading, 320
- plain text and, 41
- prompting for file and folder name, 314–317
- reading file, 321–324
- Registry editing, 349–354
- runtime errors, 229–230
- tools needed for, 94–95
- working with files and, 306

Visual Basic

- advantages to beginner of, 21
- commercial programming and, 20–21
- function in, 37
- “Hello, World!” code in, 9
- statement in, 35, 37

Windows

- desktop, saving files to, 363
- profile, creating, 362–363
- Properties screen, Summary tab, 373–375
- Search feature, 371–372

Windows Registry

- backing up in Windows XP, 330–338
- description of, 327
- editing in JScript, 354–355
- editing in VBScript, 349–354
- file creation and, 306
- finding subtree, key, subkey, or value, 341–343
- information in, 328–329
- layout of, 329
- Regedit and Regedit32, 329–330
- restoring, 339–340
- subkey, adding, 343–344
- uses for, 355–357
- value, adding, 344–346
- value, changing existing, 346–347

- Microsoft (continued)**
 - value or subkey, deleting, 348–349
 - value or subkey, renaming, 347–348
 - Windows Script Host, 306, 307, 357
 - Word
 - file, saving, 303–304
 - interface, 247–250
- minimizing distraction, 360**
- mixed case, in name of variable, 121**
- modifying requirements of project, 292**
- monitor**
 - positioning, 73
 - size of, 361
- mouse**
 - buttons and, 268
 - custom, 362
 - menu and, 271
- multidimensional array, 182–183**
- multiline comment**
 - in C++, 103
 - in JavaScript, 101–102
- multiline text box, 274**
- multiplication (*) operator, 202**
- MWSnap utility, 445**
- myths and facts, 23–25**

N

- naming**
 - files, conventions for, 368–370
 - variables
 - capitalization scheme, 122
 - clearer scheme for, 121–122
 - dos and don'ts for, 118–121
 - Hungarian Notation, 123–124
 - underscore and, 122–123
- Nero (Ahead Software) CD creation tool, 414–419, 441**
- Netbeans IDE, 436**
- Netscape. See JavaScript**
- not equal to (!=) operator, 171**
- Notepad text editor (Windows)**
 - description of, 15, 80–82
 - interface, 276–280
- null variable, 105**
- number. See also integer**
 - floating point, 105–106, 196
 - in name of variable, 118–119, 120
 - as variable, 105–106
- numerical inputs, modification for, 256–257**
- nybble, 47–48**

O

- object code, 41**
- OneNote 2003 (Microsoft) interface, 280–282**
- opening**
 - file for reading, 320
 - HTML file, 107
 - multiline comment (/*), 102, 103
- open-source software, 18**
- operators**
 - arithmetic, 65
 - assignment, 66
 - categories of, 65
 - comparison, 66–67

- logic errors and, 235–236
- logical, 67–68
- string, 68–69
- options window (Microsoft OneNote 2003), 281**
- or (||) operator, 171**
- organizing**
 - computer workspace, creating, 362–363
 - files
 - filename control, 368–370
 - folder contents note, 366–368
 - language, grouping by, 364–365
 - overview of, 359
 - project, grouping by, 365–366
 - version control
 - compiled code and, 376–377
 - software for, 377
 - Summary information and, 373–375
 - tombstone comment block and, 370–371
 - Windows Search and, 371–372
 - workspace, 360–362
- outputs to screen**
 - ambiguity in, removing, 206–211
 - annotating, 259–260
 - checking variables and, 239–244
 - Java compiler, 394
 - temperature conversion program, 188, 195

P

- packaging disc for distribution, 420–421**
- parentheses in JavaScript and JScript, 355**
- Pascal**
 - development of, 3
 - "Hello, World!" code in, 9
- path, declaring for file, 309**
- path information, storing in Windows, 148**
- PC System Questions for software previewer, 298–299**
- Perl, "Hello, World!" code in, 9**
- Perl Hypertext Preprocessor (PHP), 22**
- physical distribution. See also burning disc**
 - CD, 409–411
 - DVD, 411–412
 - floppy disk, 406–409
 - labeling and, 420–421
 - overview of, 405–406
 - packaging and, 420–421
- pkzip compression tool, 443**
- plain text, 41, 42**
- Plankalkül programming language, 2**
- planning project**
 - help file, creating, 301
 - idea
 - documenting, 287–289
 - maturing time for, 289
 - types of, 286–287
 - language, choosing, 293
 - requirements
 - drawing up set of, 290–292
 - modifying, 292
 - stages of, 360
- planning time, 360**
- platform-independent language, 89**
- pop-up box, displaying variable values in, 110–114**
- pop-up menu, 270**
- preview release, 297–299**

- problem solving**
 - basics of, 186
 - breaking problem into smaller steps, 194–195
 - coding phase, 196–204
 - improving code, 204–212
 - learning programming and, 16–17
 - overview of, 185
 - requirements, clarifying, 186–191
 - research and, 191–194
 - reviewing client requirements, 211–212
- procedures, 36–37, 38–39**
- processing inputs, 130–132**
- processing steps of project, 292**
- profile, creating, 362–363**
- program overview, including in interface, 254–256**
- programming career**
 - definition of, 13
 - examples of, 14–16
- programming, definition of, 4**
- programming stage of project**
 - basics of application and, 293
 - commenting code, 294–296
 - features, adding, 299–300
 - testing
 - final stage of, 300–301
 - preview release, 297–299
 - steps for, 296–297
 - while writing code, 293–294
 - tweaking code, 300
- programming tools**
 - C++, 91–94
 - command window, 88
 - Internet Explorer, 88
 - Java, 89–91
 - scripting languages, 94–95
 - Windows Explorer, 87–88
- programs. See software applications**
- project. See also planning project; programming stage of project**
 - additional features for, 299–300
 - folder contents note, 366–367
 - grouping folders by, 365–366
 - size of, 17
 - time-critical, 17
 - without planning, 285–286
 - work- or hobby-related, 17–18
- prompting user**
 - for file and folder name, 314–317
 - for input, 256–259
- Properties screen, Summary tab (Windows), 373–375**
- prototyping output, 207–211**
- punctuation, in name of variable, 119**
- Python, “Hello, World!” code in, 9**

Q

QBASIC, “Hello, World!” code in, 9

R

- radio button**
 - description of, 272–273
 - example of, 282
- Rapid-Q programming language, 439–440**
- Read method, 323–324**
- ReadAll method, 321–322**

- reading**
 - code
 - breaking up code, 31–34
 - overview of, 27
 - top-down, 28–30
 - each line after pressing enter to spot errors, 236
 - opening file for, 320
 - registry keys, 352–353
- ReadLine method, 322–323**
- redefining variable, 117**
- .reg (registration entries) file, 339**
- RegDelete method, 353–354**
- Regedit and Regedit32, 329–330**
- registering copyright, 423**
- registration entries (.reg) file, 339**
- RegRead method, 352–353**
- RegWrite method, 351**
- REM statement (VBScript), 98–101**
- removing**
 - ambiguity
 - from code, 238
 - from output screen, 206–211
 - unused functions from code, 300
- representing characters, 57–64**
- reproducing error before fixing, 300**
- requirements**
 - of client
 - clarifying, 186–191
 - reviewing, 211–212
 - of project
 - drawing up set of, 290–292
 - modifying, 292
- rerecordable disc, using for burning, 420**
- research**
 - maturing time for idea and, 289
 - problem solving and, 191–194
- reserved word, in name of variable, 120**
- resources. See Web sites**
- restoring Windows Registry, 339–340**
- reusing**
 - code, 32–33
 - variables, 115–116
- rewritable CD, 413**
- Ritchie, Dennis (programmer), 3**
- room for work, 360**
- root keys, abbreviations for, 350**
- running code on demand, 34**
- runtime errors, 228–232**

S

- saving data**
 - to desktop, 363
 - file life cycle, 304–305
 - frequency of, 362
 - principles of, 303–304
 - in Windows Registry, 343
- Scientific command, View menu (Windows Calculator), 51–52**
- screen resolution, 361**
- ScreenRip32 utility, 445**
- screenshot, taking, 86–87**
- script block, 108**
- Script Encoder (Microsoft), 398–403**
- security**
 - compiling code and, 396–397
 - encoding and, 398–403

- self-employment, 14**
- semicolon (;)**
 - in C++, 150, 151, 238–239
 - in JScript, 354
- server-side programming, 22**
- 7-Zip compression tool, 443**
- Short Code language, 3**
- Simonyi, Charles (programmer), 123**
- single-line text box, 273–274**
- size**
 - of file and compiler
 - Borland C++ compared to Digital Mars, 382, 383, 387
 - Java, 393–394
 - of font, 74–75
- Smalltalk**
 - development of, 3
 - "Hello, World!" code in, 9
- Snagit, 86–87**
- software applications. *See also* calculator application; interface; planning project; programming stage of project**
 - academic use, limits on, 92
 - burning CD/DVD, 414–419
 - "learning edition" copies of, 80
 - licenses for, 433
 - licensing, 423
 - for version control, 377
 - Windows Registry and, 328
- software maker, niche for, 14–15**
- source code. *See* code**
- SourceJammer version control system, 377**
- space, in name of variable, 120**
- speed**
 - CD/DVD burner and, 413
 - compiling code and, 395–396
 - virtual distribution and, 422
- spotting errors**
 - checking preceding statements, 237
 - commenting and, 237
 - keeping layout clear, 237
 - keeping track of variables, 239–244
 - reading each line after pressing enter, 236
 - removing ambiguity in code, 238
 - semicolons, 238–239
 - testing code and, 239
- SQL (Structured Query Language), 22**
- statement**
 - checking preceding to spot errors, 237
 - description of, 35–36, 37–38
 - else, 162
 - If
 - description of, 158
 - in temperature conversion program, 199, 207–211
 - REM, 99
 - semicolon and, 238–239
- storage capacity, 40**
- storing path information in Windows, 148**
- streamlining code, 300**
- string operators, 68–69**
- string.h file, 139, 151**
- strings**
 - code interpreting integer as, 134–135
 - description of, 124–125
 - manipulating, 126–130
- strType value, data types for, 351**

- structure of code**
 - arrays, 180–183
 - benefits of, 138
 - conditionals, 157–171
 - functions, 149–156
 - loops, 174–178
 - purpose of, 137–138
- Structured Query Language (SQL), 22**
- Stuffit compression tool, 444**
- subkey**
 - adding, 343–344
 - backing up, 330–332
 - creating, 350
 - deleting, 348–349
 - finding, 341–343
 - renaming, 347–348
 - restoring, 339
- subtree, finding, 341–343**
- Sun Java development kit, 436**
- support for software, 424–425**
- syntax**
 - definition of, 39
 - Microsoft Script Encoder, 400–401
- syntax error, 216**
- syntax highlighting feature, 75, 76**
- system state data, 334**

T

- Tab key, 279–280**
- tab order, 280**
- tabbing, 280**
- TaskZip compression tool, 443**
- telephone support, 425**
- temperature conversion program**
 - breaking problem into smaller steps, 194–195
 - coding phase, 196–204
 - improving code, 204–212
 - requirements for, 186–191
 - research for, 191–194
- template, creating**
 - C++, 196
 - JavaScript, 107–108
 - spotting errors and, 237
- test string, 105**
- testbed application, 296**
- testing**
 - after debugging, 300
 - beta release, 297–298
 - code
 - in browser, 108
 - to spot errors, 239
 - conditional, 161–162
 - encoding and, 403
 - final stage of, 300–301
 - logic errors and, 236
 - preview release, 297–299
 - principles or equations, 193–194
 - during programming stage of project, 293–294
 - steps for, 296–297
 - temperature conversion program, 204
 - top-down processing and, 30
- text box**
 - multiline, 274
 - single-line, 273–274

text display, for temperature conversion program, 204

text editor

- development environment and, 11
- find and replace feature, 397
- line numbering and, 232
- UltraEdit, 15–16, 75, 76, 82–83, 433–434
- Web sites for, 433–435
- Windows Notepad, 15, 80–82, 276–280

text string, replacing integer value with, 114

text-based interface

- annotating output, 259–260
- confirmations, 267
- confirming exit from application, 260–261
- description of, 251–254
- help, adding, 261–266
- program overview, including, 254–256
- prompting for input, 256–259

Textpad text editor, 434

time

- planning, 360
- for processing and top-down method, 30
- for project and planning, 286

time-critical project, 17

TinyBASIC, 3

tombstone comment block

- C++, 103
- description of, 295
- JavaScript, 101–102
- VBScript, 99
- version control and, 370–371

tools

- BASIC, 439–440
- compression, 442–444
- general
 - Snagit, 86–87
 - UltraCompare, 84–86
 - UltraEdit, 82–83
 - Windows Notepad, 80–82
 - Winzip, 83–84
- Java, 435–436
- miscellaneous, 444–445
- programming
 - C++, 91–94, 437–438
 - command window, 88
 - Internet Explorer, 88
 - Java, 89–91
 - scripting languages, 94–95
 - Web sites for, 433–435
 - Windows Explorer, 87–88

ToolTip, 281

top-down processing, 28–29

TortoiseCVS version control system, 377

traditional programming, 19–21

tweaking code, 300

two-dimensional array, 182

.txt (text file), 366

U

Ultimate++ IDE, 437

UltimateZip compression tool, 443

UltraCompare, 84–86, 384

UltraEdit text editor

- description of, 15–16, 82–83, 433–434
- syntax highlighting feature, 75, 76

underscore (_)

- in name of file, 369
- in name of variable, 121, 122–123

unterminated string, 219

updating variable, 106–107

user information and Windows Registry, 328

user, prompting

- for file and folder name, 314–317
- for input, 256–259

V

value

- adding, 344–346
- changing existing, 346–347
- deleting, 348–349
- finding, 341–343
- renaming, 347–348
- strType, data types for, 351

var keyword, 109, 117–118

variables

- char, 164–167
- code for variables in action, 109–118
- commenting, 296
- declaring, 109, 160
- exercises, 124
- float, 164–166
- folder name and, 313
- integrating with test output, 209–210
- keeping track of, 239–244
- manipulating, 132–135
- manipulating strings, 126–130
- naming
 - capitalization scheme, 122
 - clearer scheme for, 121–122
 - dos and don'ts for, 66, 118–121
 - Hungarian Notation, 123–124
 - underscore and, 122–123
- obscure names for, 397–398
- overview of, 104–107
- processing inputs and, 130–132
- strings and, 124–125
- for temperature conversion program, 196–197, 200

.vbe file extension, 403

.vbs file extension, 307

VBScript (Microsoft)

- advantages of, 79
- checking for duplicate files, 317
- client-side programming and, 23
- commenting code, 98–101
- file
 - appending, 319–320
 - creating, 307–309
 - deleting, 325
 - editing, 317–319
 - opening for reading, 320
 - reading, 321–324
- folder
 - choice of and conditionals, 312–313
 - creating, 310–311
 - deleting, 325
 - variables and, 313
- haphazard example of, 228–229
- logic error, 232–234
- multiple files, creating, 311

VBScript (Microsoft) (continued)
 plain text and, 41
 prompting for file and folder name, 314–317
 Registry editing, 349–354
 runtime errors, 229–230
 tools needed for, 94–95
 working with files and, 306

version control
 compiled code and, 376–377
 software for, 377
 Summary Information and, 373–375
 tombstone comment block and, 370–371
 Windows Search and, 371–372

View menu commands, Scientific (Windows Calculator), 51–52

virtual distribution
 cons of, 422
 free versus commercial, 422–423
 pros of, 421–422

Visual Basic (Microsoft)
 advantages to beginner of, 21
 commercial programming and, 20–21
 function in, 37
 “Hello, World!” code in, 9
 statement in, 35, 37

void keyword, 140

VRML, “Hello, World!” code in, 10

W

warnings, dealing with, 226–228

Web programming, 21–23

Web scripting languages, 441

Web sites
 author, 446
 BASIC tools, 439–440
 Borland C++ compiler, 92–93
 C++ tools, 437–438
 CD burning, 441
 compilers, 381
 compression tools, 442–444
 distributing free software, 423
 Java, 89–90, 435–437
 Java tools, 435–436
 Microsoft
 Developer Network, 446
 Script Encoder, 399
 Visual Basic, 21
 peripherals, 362
 programming tools, 433–435
 registering copyright, 423
 resources, miscellaneous, 446
 software version control, 377
 tools, miscellaneous, 444–445
 UltraCompare, 86
 UltraEdit text editor, 15
 virtual distribution, 423
 Web scripting languages, 441
 Winzip, 84

Web support, 424–425

While loop, 176–177

whitespace in code, 298

Winace compression tool, 442

Windows (Microsoft)
 Calculator
 binary and, 51–54
 hexadecimal system and, 57
 desktop, saving files to, 363
 Explorer, 87–88
 Notepad text editor
 description of, 15, 80–82
 interface, 276–280
 profile, creating, 362–363
 Properties screen, Summary tab, 373–375
 Registry
 backing up in Windows XP, 330–338
 description of, 327
 editing in JScript, 354–355
 editing in VBScript, 349–354
 file creation and, 306
 finding subtree, key, subkey, or value, 341–343
 information in, 328–329
 layout of, 329
 Regedit and Regedit32, 329–330
 restoring, 339–340
 subkey, adding, 343–344
 uses for, 355–357
 value, adding, 344–346
 value, changing existing, 346–347
 value or subkey, deleting, 348–349
 value or subkey, renaming, 347–348
 Script Host, 306, 307, 357
 Search feature, 371–372
 storing path information in, 148

Winzip (compression tool), 83–84, 442

word, 49–50

Word (Microsoft)
 file, saving, 303–304
 interface, 247–250

word processor file, saving, 303–304

work project, 17–18

work-based training, 77–78

workspace
 chair, lighting, and air flow, 73
 desk, 73
 digital, creating, 362–363
 fonts, 73–75
 keyboard, 71–73
 monitor, 73
 organizing, 360–362

Y

YABasic Interpreter, 440

Z

.zip file, 83

Zope server, 441

Zuse, Konrad (scientist), 2–3