تعلم برمجة Pascal الدرس الاول - مقدمه

عن **باسكال** :

باسكال لغة برمجه انشئت بواسطة Niklaus Wirth في عام ١٩٧٠ . كان اسمها سابقاً Blaise Pascal ، عالم الرياضيات الفرنسي المشهور . لقد صنعت هذه اللغه لتعليم البرمجه ولكي تكون محل ثقه لدى المبرمجين . منذ ذلك الوقت قد اصبحت لغة البرمجه **باسكال** اكثر من انها فقط لغه اكاديميه و لكنّها استخدمت تجارياً ايضا .

ما الذي احتاجه لأبدأ مع **باسكال** :

قبل بداية تعلم برمجة الباسكال ، تحتاج إلى مترجم **باسكال** (compiler) . هذا الدرس يستخدم (مترجم **باسكال** المجاني) : http://www.freepascal.org/ . يمكنك الحصول على قائمه من مترجمات **باسكال** في
http://www.freebyte.com/programming/pascal/#pascal_compilers .

برنامجك الاول :

اول شئ يجب ان تفعله ان تفتح مترجم الباسكال .

دائماً نبدأ البرنامج بكتابة اسمه . ادخل program و اسم البرنامج بعدها . سوف نقوم بتسمية برنامجنا الاول "Hello" لانه سيقوم بطباعة الجمله المشهوره "world Hello" على الشاشه ، سوف تكون طريقة الكتابه بهذه الطريقه :

كود

```
program Hello;
```

بعدها يجب علينا ان نطبع الكلمتين begin و end . بعدها نقوم بكتابة شيفرة البرنامج بين هاتين الكلمتين . تذكر وضع النقطه بعد الكلمه end ، هكذا ستكون صيغة ما كتبناه حتى الآن :

كود

```
program Hello;
```

```
begin
end.
```

الآن نكتب الشيفره التي تطبع الجمله على الشاشه :

كود

```
program Hello;

begin
  Write('Hello world');
end.
```

يجب ان تكون الجمله بين علامة الاقتباس الفرديه ' . واي شي يكون بين علامة الاقتباس الفرديه عباره عن سلسله (كلمات و حروف) . كل السلاسل يجب ان تكون بهذا الشكل . علامة الفاصله المنقوطه ؛ هي نهاية السطر . يجب ان تتذكر دائما ان تضع هذه العلامه في نهاية سطر .

الامر Readln يستخدم لانتظار المستخدم ان يقوم بالضغط على زر الادخال لكي ينتهي البرنامج .

كود

```
program Hello;

begin
  Write('Hello world');
  Readln;
end.
```

الآن يجب ان تخزّن البرنامج بإسم hello.pas

تجميع البرنامج و ترجمته :

برنامجنا الاول الآن جاهز للترجمه . عندما تقوم بترجمة البرنامج ، سيقوم المترجم بقرائة الملف المصدري (الشفره التي قمت بكتابتها) و يقوم بتطبيقها . اذا كنت تستخدم مترجم من نوع IDE قم بالضغط على CTRL+F9 انها دائماً تستخدم لترجمة و تشغيل البرنامج في المترجمات من نوع IDE . اذا كنت تستخدم مترجم يعتمد على سطر الاوامر مثل Free Pascal ادخل الامر التالي :

كود

```
fpc hello.pas
```

اذا تلقيت اي خطأ عند الترجمه يجب ان تقرأ هذا الدرس مره اخرى بعنايه لاكتشاف اين هو الخطأ .
مستخدمين IDE سوف يجدون ان برنامجهم تمت ترجمته و تم تشغيله في وقت قصير .
مستخدمين سطر الاوامر يجب ان يدخلون اسم البرنامج في متلقي الاوامر لكي يعمل .

يجب ان تجد الجمله "Hello world" عندما تقوم بتشغيل برنامجك و عند الضغط على زر الادخال سوف يتم الخروج من البرنامج .
مبروك ! لقد كتبت اول برنامج لك في **باسكال** .

المزيد من الاوامر :
الامر Writeln مثل الامر Write بإستثناء انه يقوم بنقل المؤشر إلى سطر جديد . هنا شفره لبرنامج يقوم بطباعة الكلمه Hello و بعدها world في السطر الذي يليه :

كود

```
program Hello;

begin
  Writeln('Hello');
  Write('world');
  Readln;
end.
```

استخدام الاوامر من الوحدات :

الاوامر التي تقوم ببنائها في مترجمك اساسيه جداً و انت تحتاج لاستخدام المزيد . الوحدات يمكن ان تُدرَج في برنامجك لكي تعطيك القدره على استخدام المزيد من الاوامر . الوحده crt احد اقوى الوحدات المفيده الموجوده . الامر ClrScr موجود من ضمن الوحده crt و استخدامها هو تنظيف الشاشه . كيفية استخدام هذا الامر :

كود

```
program Hello;

uses
  crt;

begin
  ClrScr;
  Write('Hello world');
  Readln;
end.
```

التعليقات :

التعليقات جمل تستخدم للتعليق على احد اجزاء البرنامج . المترجم لا يعتبر التعليقات جزء من الشيفره والتعليقات فقط للاشخاص الذين يطّلعون على الشيفره المصدريه للبرنامج ، التعليقات يتم اضافتها بين العلامتين { } . يفضل دائما ان تقوم بكتابة تعليقات في اول الملف تشرح فيهاوظيفة الملف و ان تقوم بكتابة التعليقات بجانب الاوامر صعبة الفهم لكي تقوم بشرحها .

مثال :

كود

```
{This program will clear the screen, print "Hello world" and wait for the user to
press enter.}

program Hello;
```

```
uses
  crt;

begin
  ClrScr;{Clears the screen}
  Write('Hello world');{Prints "Hello world"}
  Readln;{Waits for the user to press enter}
end.
```

المسافات :

قد تلاحظ اننا تركنا قبل الاوامر ٣ اسطر . هذه الطريقه مفيده في جعل البرنامج سهل القرائه . الكثير من المبتدئين لا يفهم سبب وضع تلك المسافات ولكن اذا بدأ بكتابة البرامج الكبيره سوف يفهم فائدتها .

مصدر هذا الدرس : http://www.sepsis.za.net/programming_pascal01.html

مترجم الدرس : MaaSTaaR

تعلم برمجة Pascal الدرس الثاني - الالوان ، الاحداثيات ، النوافذ و الصوت

الالوان :

لتغيير لون النص الذي يُطبع على الشاشه يمكننا استخدام الامر TextColor .

مثال :

كود

```
program Colors;

uses
  crt;

begin
  TextColor(Red);
  Writeln('Hello');
  TextColor(White);
  Writeln('world');
end.
```

الامر TextBackground يقوم بتغيير لون خلفية النص . اذا كنت تريد تغيير لون خلفية الشاشه إلى لون معيّن يجب ان تستخدم الامر ClrScr .

كود

```
program Colors;

uses
  crt;

begin
```

```
  TextBackground(Red);
  Writeln('Hello');
  TextColor(White);
  ClrScr;
end.
```

إحداثيات الشاشه :

يمكنك ان تضع السهم في اي مكان تريده في الشاشه بإستخدام الامرGoToXY . في الدوس ، الشاشه بعرض ٨٠ حرف و ارتفاعها ٢٥ حرف . يتغير الطول و العرض في المنصات الاخرى ، قد تتذكر الرسوم البيانيه في الرياضيات . احداثيات الشاشه كذلك تعمل بنفس الطريقه . في المثال الذي سوف نعرضه سوف نعرف كيف نحرّك المؤشر إلى العمود العاشر في الصف الخامس

كود

```
program Coordinates;

uses
  crt;

begin
  GoToXY(10,5);
  Writeln('Hello');
end.
```

النوافذ :

النوافذ ستسمح لك بتعريفها على الشاشه لكي تحتل المنطقه التي تطلبها . اذا قمت بإنشاء النافذه و قمت بتنظيف الشاشه سوف يتم تنظيف ما على الشاشه فقط. الامر Window يأخذ ٤ بارامترات و هم إحداثيات (اعلى ، يسار ، يمين ، اسفل) .

كود

```
program Coordinates;

uses
  crt;

begin
  Window(1,1,10,5);
  TextBackground(Blue);
  ClrScr(Blue);
end.
```

الصوت :

الامر Sound يُصدر صوت بالتكرار الذي تطلبه منه الذي تطلبه منه . الامر Delay يأخذ الوقت بالثواني التي تخبره بها . يستخدم هذا الامر بين الامرين Sound و NoSound ليُصدر الصوت الاخير من مقدار معين من الوقت

كود

```
program Sounds;

uses
  crt;

begin
  Sound(1000);
  Delay(1000);
  NoSound;
end.
```

مصدر هذا الدرس : http://www.sepsis.za.net/programming_pascal02.html

مترجم الدرس : MaaSTaaR

تعلم برمجة Pascal الدرس الثالث - المتغيرات و الثوابت

ماهي المتغيرات؟

المتغيرات عباره عن اسماء تخزّن في ذاكرة الحاسوب. هذا الاسم يستخدم لتخزين المعلومات في الذاكره .

يمكننا استخدام انواع مختلفه من المعلومات في المتغيرات ، مثلاً الارقام و السلال و غيره .

استخدام المتغيرات :

يجب دائماً ان نعلن عن المتغيرات قبل استخدامها . نستخدم الكلمه الاساسيه var لعمل ذلك .

دائماً يجب اختيار نوع البيانات للمتغير . وهذه هي انواع المتغيرات المختلفه .

: Byte

من ٠ إلى ٢٥٥

: Word

من ٠ إلى ٦٥٥٣٥

: ShortInt

من -١٢٨ إلى ١٢٧

: Integer

من -٣٢٧٦٨ إلى ٣٢٧٦٧

: LongInt

من -٤٢٢٨٢٥٠٠٠٠ إلى ٤٢٢٨٢٤٩٠٠٠

: Real

قيم النّقطة المتغيّرة

: Char

حرفاً واحداً

String :

فوق ٢٥٥ حرف

Boolean :

true او false

في هذا المثال سوف نتعرف على طريقة الاعلان عن متغير من نوع intger اسمه i :

كود

```
program Variables;

var
  i: Integer;

begin
end.
```

لإعطاء قيمه للمتغير يتم استخدام عامل الاسناد :=

كود

```
program Variables;

var
  i: Integer;

begin
  i := 5;
end.
```

يمكنك انشاء متغيرين او اكثر من نفس النوع إذا وضعت الفاصله بعد المتغير الاول و قمت بكتابة اسم المتغير الثاني بعدها . كذلك يمكنك انشاء متغيرات مختلفه بدون الحاجه إلى استخدام الجمله الاساسيه var اكثر من مره :

كود

```
program Variables;

var
  i, j: Integer;
  s: String;

begin
end.
```

اذا كنت تريد ادراج سلسله في متغير يجب استخدام علامة الاقتباس الفرديه ' ، النوع Boolean فقط يمكن اسناد قيمتين له و هم True و False :

كود

```
program Variables;

var
  i: Integer;
  s: String;
  b: Boolean;

begin
  i := -3;
  s := 'Hello';
  b := True;
end.
```

الحسابات مع المتغيرات :

يمكن استخدام المتغيرات في العمليات الحسابيه . على سبيل المثال يمكن أن تسند القيمة بمتغيّر ثمّ تضيف الرقم ١ له . العمليات الحسابيه التي يمكن استخدامها :

+ : اضافه (زائد)

- : طرح

* : ضرب

/ : قسمه

div : قسمة عدد صحيح

mod : باقي قسمة العدد الصحيح

المثال التالي يقوم ببعض العمليات الحسابيه :

كود

```
program Variables;

var
  Num1, Num2, Ans: Integer;

begin
  Ans := 1 + 1;
  Num1 := 5;
  Ans := Num1 + 3;
  Num2 := 2;
  Ans := Num1 - Num2;
  Ans := Ans * Num1;
end.
```

يستخدم النوع Strings في تخزين السلاسل مثل الحروف والارقام المختلفه و غيره ، و يمكنك استخدام العمليات الحسابيه مع هذا النوع من المتغيرات حيث لا يتم جمع الرقمين المخزنين على

صيغة String بمعنى جمع ، مثلاً اذا قمت بإضافة السلسله التي قيمتها ١ إلى السلسله التي قيمتها ١ سوف يكون الناتج ١١ و ليس ٢ .

كود

```
program Variables;

var
  s: String;

begin
  s := '1' + '1';
end.
```

يمكنك قراءة محتوى المتغيرات بإستخدام Readln و ReadKey . الامر الثاني و هو ReadKey من الوحده crt و هي تقرأ حرف واحد فقط . سوف ترى ان طريقة عمل ReadKey مختلفه عن Readln .

كود

```
program Variables;

uses
   crt;

var
  i: Integer;
  s: String;
  c: Char;

begin
  Readln(i);
  Readln(s);
  c := ReadKey;
```

```
end.
```

طباعة المتغيرات على الشاشه طريقه سهله . اذا كنت تريد طباعة المتغيرات على الشاشه يمكنك استخدام الامر Writeln :

كود

```
program Variables;

var
  i: Integer;
  s: String;
begin
  i := 24;
  s := 'Hello';
  Writeln(i);
  Writeln(s,' world');
end.
```

الثوابت :

الثوابت لا تختلف عن المتغيرات إلا انه قيمتها لا يمكن ان تتغير أي قيمه ثابته . يمكن اسناد القيمه إلى الثابت عند الاعلان عن الثابت . const هي الكلمه الاساسيه للاعلان عن الثوابت .

كود

```
const
  pi: Real = 3.14;

var
  c, d: Real;

begin
```

```
  d := 5;
  c := pi * d;
end.
```

مصدر هذا الدرس : http://www.sepsis.za.net/programming_pascal03.html

مترجم الدرس : MaaSTaaR

تعلم برمجة Pascal الدرس الرابع - التعامل مع السلاسل و التحويلات

التعامل مع السلاسل :

يمكنك في السلاسل ان تقوم بإستخراج حرف معيّن من سلسله معينه

كود

```
program Strings;

var
  s: String;
  c: Char;

begin
  s := 'Hello';
  c := s[1];{c = 'H'}
end.
```

كذلك يمكنك معرفة طول سلسله بإستخدام الامر Length :

كود

```
program Strings;

var
  s: String;
  l: Integer;

begin
  s;= 'Hello';
  l := Length(s);{l = 5}
end.
```

للبحث عن شئ في سلسله يمكنك استخدام الامر Pos .

البارامترات :

١: كلمة البحث

٢: السلسله التي سيتم البحث بها

كود

```
program Strings;

var
  s: String;
  p: Integer;

begin
  s := 'Hello world';
  p := Pos('world',s);
end.
```

الامر Delete يقوم بحذف الحروف من السلسله .

البارامترات :

١: السلسله التي سيتم الحذف منها

٢: بدء الحذف من اين

٣: كمية الاحرف التي سيتم حذفها

كود

```
program Strings;

var
  s: String;
```

```
begin
  s;= 'Hello';
  Delete(s,1,1);{s = 'ello'}
end.
```

الامر Copy يقوم بنسخ ما هو مطلوب من السلسله

البارامترات :

١: السلسله التي سيتم النسخ منها

٢: بدء النسخ من اين

٣: كمية الاحرف التي سيتم نسخها

كود

```
program Strings;

var
  s, t: String;

begin
  s;= 'Hello';
  t := Copy(s,1,3);{t = 'Hel'}
end.
```

الامر Insert يقوم بإضافة الحروف المطلوبه في السلسله المختاره .

البارامترات :

١: الحروف التي سوف تضاف للسلسله

٢: المتغير الذي سيتم الاضافه إليه

٣: من اين سيتم اضافة الحروف

كود

```
program Strings;
```

```
var
  s: String;

begin
  s := 'Hlo';
  Insert('el',s,2);
end.
```

التحويلات :

الامر Str يقوم بتحويل المتغير من نوع integer إلى string .

كود

```
program Convert;

var
  s: String;
  i: Integer;

begin
  s;= '123';
  Str(i,s);
end.
```

الامر Val يقوم بتحويل المتغير من نوع string إلى integer

كود

```
program Convert;

var
```

```
  s: String;
  i: Integer;

begin
  i;= 123;
  Val(s,i,i);
end.
```

الامر Int يعطيك العدد قبل الفاصله في عدد حقيقي

كود

```
program Convert;

var
  r: Real;

begin
  r := Int(3.14);
end.
```

الامر Frac يعطيك العدد بعد الفاصله في عدد حقيقي

كود

```
program Convert;

var
  r: Real;

begin
  r := Frac(3.14);
```

```
end.
```

الامر Round سيكمل عدد حقيقي إلى اقرب عدد صحيح

كود

```
program Convert;

var
  i: Integer;

begin
  i := Round(3.14);
end.
```

الامر Trunc سيعطيك العدد قبل فاصلة كعدد صحيح .

كود

```
program Convert;

var
  i: Integer;

begin
  i := Trunc(3.14);
end.
```

الحاسوب يستخدم الارقام من ٠ إلى ٢٥٥ (١ بايت) لتمثيل الحروف وهذه الحروف تسمّى بحروف ASCII

الامر Ord يحول الحروف إلى ارقام و الامر Chr يحول الارقام إلى حروف . استخدم العلامه # قبل

الحرف لتحويلها إلى احرف

كود

```
program Convert;

var
  b: Byte;
  c: Char;

begin
  c := 'a';
  b := Ord(c);
  c := Chr(b);
  c := #123;
end.
```

إضافات :

الامر Random يعطيك رقم عشوائي من الصفر إلى الرقم الذي تعطيه اياه و الامرRandomize يستخدم لعمل ارقام اكثر عشوائيه بالإعتماد على ساعة النظام.

كود

```
program Rand;

var
  i: Integer;

begin
  Randomize;
  i := Random(101);
end.
```

مصدر هذا الدرس : http://www.sepsis.za.net/programming_pascal04.html

مترجم هذا الدرس : MaaSTaaR

تعلم برمجة Pascal الدرس الخامس - اتخاذ القرار

: if then else

الجمله الشرطيه if تسمح لك بإتخاذ القرارات في برنامجك . المثال التالي يسأل المستخدم ان يدخل رقم و يخبر المستخدم اذا كان الرقم اكبر من ٥ .

كود

```
program Decisions;

var
  i: Integer;

begin
  Writeln('Enter a number');
  Readln(i);
  if i > 5 then
     Writeln('Greater than 5');
end.
```

و القرارات التي يمكنك اتخاذها مع الجمله الشرطيه if :

> : اكبر من

< : اصغر من

>= : اكبر من او يساوي

<= : اصغر من او يساوي

= : يساوي

<> : لا يساوي

المثال السابق كان يخبرنا فقط اذا كان الرقم اكبر من ٥ ، اذا كنا نريد ان يخبرنا انه ليس اكبر من ٥ يجب ان نستخدم else .

كود

```
program Decisions;

var
  i: Integer;

begin
  Writeln('Enter a number');
  Readln(i);
  if i > 5 then
     Writeln('Greater than 5')
  else
     Writeln('Not greater than 5');
end.
```

إذا كانت الحاله صحيحه (True) اختر الجزء الذي بعد then و اذا لم تكن صحيحه (False) اختر الجزء الذي بعد else . ذلك لان الحاله i > ٥ تعتبر من النوع Boolean

كود

```
program Decisions;

var
  i: Integer;
  b: Boolean;

begin
  Writeln('Enter a number');
  Readln(i);
  b := i > 5;
end.
```

اذا كنت تريد ان تستخدم اكثر من حاله واحده يجب عليك ربط حاله بالاخرى . لربط الحالات يمكنك استخدام AND او OR . اذا استخدمت AND يجب ان تكون الحالتين صحيحتين و اذا استخدمت OR يجب ان تكون حاله واحده او الحالتين صحيحتين .

كود

```
program Decisions;

var
  i: Integer;

begin
  Writeln('Enter a number');
  Readln(i);
  if (i > 1) and (i < 100) then
     Writeln('The number is between 1 and 100');
end.
```

اذا كنت تريد كتابة امرين او اكثر بعد الجمله الشرطيه يجب استخدام begin و end؛ .

كود

```
program Decisions;

var
  i: Integer;

begin
  Writeln('Enter a number');
  Readln(i);
  if i > 0 then
     begin
       Writeln('You entered ',i);
```

```
      Writeln('It is a positive number');
    end;
end.
```

كذلك يمكنك استخدام if داخل if

كود

```
program Decisions;

var
  i: Integer;

begin
  Writeln('Enter a number');
  Readln(i);
  if i > 0 then
    Writeln('Positive')
  else
    if i < 0 then
      Writeln('Negative')
    else
      Writeln('Zero');
end.
```

: Case

الامر Case يشبه إلى حد ما if لكنه يقبل العديد من الحالات مره واحده

كود

```
program Decisions;
```

```
uses
  crt;

var
  Choice: Char;

begin
  Writeln('Which on of these do you like?');
  Writeln('a - Apple:');
  Writeln('b - Banana:');
  Writeln('c - Carrot:');
  Choice := ReadKey;
  case Choice of
     'a': Writeln('You like apples');
     'b': Writeln('You like bananas');
     'c': Writeln('You like carrots');
  else;
     Writeln('You made an invalid choice');
  end;
end.
```

مصدر هذا الدرس : http://www.sepsis.za.net/programming_pascal05.html

مترجم هذا الدرس : MaaSTaaR

تعلم برمجة Pascal الدرس السادس - التكرار

التكرار يستخدم متى اردت اعادة تطبيق الشيفره اكثر من مره .
مثال : اذا اردنا ان نطبع الجمله Hello على الشاشه ١٠ مرات سوف نحتاج إلى كتابة الامر Writeln عشرة مرات . يمكنك بإستخدام التكرار ان تكتب الامر مره واحد و سوف يتم طباعة الجمله ١٠ مرات .

هناك ٣ انواع من التكرار و هم for , while , repeat .

التكرار For :
هكذا يتم استخدام التكرار for

كود

```
program Loops;

var
  i: Integer;

begin
  for i := 1 to 10 do
    Writeln('Hello');
end.
```

اذا كنت تريد استخدام اكثر من امر بعد الكلمه for فيجب ان تضع الاوامر بوسط الامر begin و end

كود

```
program Loops;

var
  i: Integer;
```

```
begin
  for i := 1 to 10 do
     begin
        Writeln('Hello');
        Writeln('This is loop ',i);
     end;
end.
```

التكرار While :

التكرار While يكرر الاوامر ما دام الشرط صحيح . طريقة الاستخدام

كود

```
program Loops;

var
  i: Integer;

begin
  for i := 1 to 10 do
     begin
        Writeln('Hello');
        Writeln('This is loop ',i);
     end;
end.
```

التكرار Repeat until :

التكرار Repeat until يشبه التكرار while و لكنه يختبر الحاله في اسفل التكرار .

كود

```
program Loops;
```

```
var
  i: Integer;

begin
  i := 0;
  repeat
     i := i + 1;
     Writeln('Hello');
  until i = 10;
end.
```

اذا كنت تريد استخدام اكثر من حاله في التكرار while او rapeat اضف الاقواس بين الحاله

كود

```
program Loops;

var
  i: Integer;
  s: String;

begin
  i := 0;
  repeat
     i := i + 1;
     Write('Enter a number: ');
     Readln(s);
  until (i = 10) or (s = 0);
end.
```

Break و Continue :

الامر Break يخرج من من التكرار في اي وقت . في المثال التالي البرنامج لا يطبع اي شئ لانه يخرج من التكرار قبل تنفيذ العمليه.

كود

```
program Loops;

var
  i: Integer;

begin
  i := 0;
  repeat
    i := i + 1;
    Break;
    Writeln(i);
  until i = 10;
end.
```

الامر Continue يقفز إلى اعلى التكرار

كود

```
program Loops;

var
  i: Integer;

begin
  i := 0;
  repeat
    i := i + 1;
    Continue;
    Writeln(i);
```

```
  until i = 10;
end.
```

مصدر هذا الدرس : http://www.sepsis.za.net/programming_pascal06.html

مترجم هذا الدرس : MaaSTaaR

تعلم برمجة Pascal الدرس السابع - المصفوفات

المصفوفات احد انواع المتغيرات ، ما يميز المصفوفات عن باقي المتغيرات انه يمكنها تخزين اكثر من قيمه في متغير واحد .

المصفوفات تعرّف غالباً مثل طريقة تعريف المتغيرات و لكن يجب ان تذكر عدد البيانات التي سوف يتم تخزينها .

كود

```
program Arrays;

var
  a: array[1..5] of Integer;

begin
end.
```

يمكننا ان نصل إلى كل العناصر بإستخدام اسم المتغير و من ثم وضع رقم العنصر بين القوسين []

كود

```
program Arrays;

var
  a: array[1..5] of Integer;

begin
  a[1] := 12;
  a[2] := 23;
  a[3] := 34;
  a[4] := 45;
  a[5] := 56;
```

```
end.
```

لسهوله اكثر في استدعاء المصفوفات للقرائه استخدام التكرار لكي يتم قرائة الخمس عناصر من خلال سطر واحد .

كود

```
program Arrays;

var
  a: array[1..5] of Integer;
  i: Integer;

begin
  for i := 1 to 5 do
     Readln(a[i]);
end.
```

تصنيف المصفوفات :

احياناً قد تريد ترتيب قيم المصفوفات بطريقه معينه . لعمل ذلك يمكنك استخدام تصنيف bubble . تصنيف bubble واحده من الطرق الكثيره لتصنيف المصفوفات و هو الاكثر شعبيه . في تصنيف bubble اكبر اعداد ينتقلون إلى آخر المصفوفه .

كود

```
program Arrays;

var
  a: array[1..5] of Integer;
  i, j, tmp: Integer;

begin
```

```
  a[1] := 23;
  a[2] := 45;
  a[3] := 12;
  a[4] := 56;
  a[5] := 34;
  for i := 1 to 4 do
    for j := 2 to 5
      if a[j] > a[j + 1] then
        begin
          tmp := a[j];
          a[j] := a[j + 1];
          a[j + 1] := tmp;
        end
end.
```

مصفوفات D٢ :

يمكن ان يكون لدى المصفوفات بعدان بدلاً من واحد ، او بعباره اخرى يمكن ان يكون لديهم صفوف و اعمده بدلاً من صفوف فقط .

كود

```
program Arrays;

var
  a: array [1..3,1..3] of Integer;

begin
end.
```

للوصول إلى القيم يجب ان تستخدم اسم المتغير ثم عددين بين الاقواس [] .

كود

```
program Arrays;

var
  r, c: Integer;
  a: array [1..3,1..3] of Integer;

begin
  for r := 1 to 3 do
     for c := 1 to 3 do
        Readln(a[r,c]);
end.
```

مصدر هذا الدرس : http://www.sepsis.za.net/programming_pascal07.html

مترجم هذا الدرس : MaaSTaaR

تعلم برمجة Pascal الدرس الثامن - Types، Records و Sets

الانواع :

من الممكن ان تقوم بإنشاء نوع متغيرات خاص بك بإستخدام الامر type .

النّوع الاول الذي يمكن ان تعمله هو نوع records (سجلات) . السجلات هي عباره عن متغيّران او اكثر من انواع مختله في واحد . على سبيل المثال ان يكون المستخدم طالب و لديه رقم و اسم الطالب . كيف سوف ننشئ نوعاً من البيانات :

كود

```
program Types;

Type
  Student = Record
     Number:
     Name;
  end;

begin
end.
```

بعد ان تنشئ النوع يجّب ان تعلن عن متغير من النوع الذي قمنا بإنشاءه لكي يمكننا استخدامه :

كود

```
program Types;

Type
  StudentRecord = Record
     Number:
     Name;
  end;
```

```
var
  Student: StudentRecord;

begin
end.
```

للوصول إلى العدد و الاسم من السجل يجب ان تتبع الآتي :

كود

```
program Types;

Type
  StudentRecord = Record
     Number:
     Name;
  end;

var
  Student: StudentRecord;

begin
  Student.Number := 12345;
  Student.Name := 'John Smith';
end.
```

النوع الآخر هو set ، النوع set ليس له فائده كبيره ، و اي شئ يمكنك عمله مع set يمكن ان يعمل بسهوله في طريقة اخرى . على سبيل المثال لدينا نوع set يدعى Animal و المعلومات المخزنه : cat , dog و rabbit

كود

```
program Types;

Type
  Animal = set of (dog, cat, rabbit);

var
  MyPet: Animal;

begin
  MyPet := dog;
end.
```

لا يمكنك استخدام Readln او Writeln في sets . يمكنك ان تنشئ مدى من المتغيرات من 'a' إلى 'z' . في هذا النوع يمكنك ان تختبر إذا كان المتغير من ضمن المدى :

كود

```
program Types;

uses
  crt;

Type
  Alpha = 'a'..'z';

var
  Letter: set of Alpha;
  c: Char;

begin
  c := ReadKey;
  if c in [Letter] then
     Writeln('You entered a letter');
end.
```

مصدر هذا الدرس : http://www.sepsis.za.net/programming_pascal08.html

مترجم هذا الدرس : MaaSTaaR

تعلم برمجة Pascal الدرس التاسع - الإجراءات و الدوال

الإجراءات :
الإجراءات يعتبر برنامج فرعي . يمكن ان يستدعى من الجزء الرئيسي للبرنامج . يتم الإعلان عن الاجراء خارج الجسم الرئيسي للبرنامج بإستخدام الكلمه procedure . يجب ان تعطي الاجراء اسم مميز . الاجراءات لها بدايه و نهايه خاصه بها . في هذا المثال سوف نتعرف على طريقة الاعلان و الاستدعاء لاجراء اسمه Hello يطبع الكلمه "Hello" على الشاشه .

كود

```
program Procedures;

procedure Hello;
begin
  Writeln('Hello');
end;

begin
end.
```

لإستخدام الاجراء يجب ان نقوم بإستدعاءه من خلال اسمه المميز في جسم البرنامج (مكان كتابة شيفره البرنامج)

كود

```
program Procedures;

procedure Hello;
begin
  Writeln('Hello');
end;

begin
```

```
  Hello;
end.
```

الاجراءات يجب ان تكون دائماً فوق المكان الذي يتم استدعاءه اي يجب ان قوم بالاعلان عن الاجراء ثم استدعاءه ولا يجوز استدعاء الاجراء قبل الاعلان عنه . في المثال التالي سوف يتم استدعاء اجراء داخل اجراء آخر .

كود

```
program Procedures;

procedure Hello;
begin
  Writeln('Hello');
end;

procedure HelloCall;
begin
  Hello;
end;

begin
  HelloCall;
end.
```

يمكن ان تكون للاجراءات بارامترات مثل اي امر تستخدمه . يجب ان تعطي اسم البارامتر و نوعه و سوف يتم استخدامه مثل اي متغير . اذا كنت تريد ان تستخدم اكثر من بارامتر في اجراء واحد يجب ان تكون الفاصله المنقوطه ؛ هي التي تفصل بينهم

كود

```
program Procedures;
```

```
procedure Print(s: String; i: Integer);
begin
  Writeln(s);
  Writeln(i);
end;

begin
  Print('Hello',3);
end.
```

المتغيرات العامّه و المحليه :

المتغيرات التي نستخدمها حاليّاً تعتبر متغيرات عامه ، اي يمكننها استخدامها في اي وقت و في اي مكان في البرنامج . المتغيرات المحليه يمكن استخدامها فقط بداخل الاجراء و المتغيرات المحليه لا تأخذ مكانها من الذاكره إذا لم يبدأ الاجراء . يتم الاعلان عن المتغيرات المحليه تحت إعلان اسم الاجراء .

كود

```
program Procedures;

procedure Print(s: String);
var
  i: Integer;
begin
  for i := 1 to 3 do
     Writeln(s);
end;

begin
  Print('Hello');
end.
```

الدوال :

الدوال تشبه الاجراءات بإستثناء انها تعيد قيمه . الكلمه function تستخدم بدل الكلمه procedure عند تعريف الدوال . لتعريف ما هو نوع البيانات للقيمه العائده يجب ان تستخدم النقطتين : و بعدها نوع البيانات و قبل النقطتين اسم الدالّه .

كود

```
program Functions;

function Add(i, j:Integer): Integer;
begin
end;

begin
end.
```

عند اسناد قيمة الداله لمتغير سيجعل المتغير يساوي القيمه المعاده من الداله . اذا استخدمت الداله في شئ مع مثل Writeln سوف يتم طباعة قيمة العائد . لاسناد قيمة العائد انشئ اسم الداله تساوي القيمه التي تريدها ان تكون هي العائد.

كود

```
program Functions;

var
  Answer: Integer;

function Add(i, j:Integer): Integer;
begin
  Add := i + j;
end;
```

```
begin
  Answer := Add(1,2);
  Writeln(Add(1,2));
end.
```

يمكنك الخروج من اجراء او داله في اي وقت بإستخدام الامرExit .

كود

```
program Procedures;

procedure GetName;
var
  Name: String;
begin
  Writeln('What is your name?');
  Readln(Name);
  if Name = '' then
     Exit;
  Writeln('Your name is ',Name);
end;

begin
  GetName;
end.
```

مصدر هذا الدرس : http://www.sepsis.za.net/programming_pascal09.html

مترجم هذا الدرس : MaaSTaaR

تعلم برمجة Pascal الدرس العاشر - الملفات النصيه

الملفات النصيه عباره عن ملفات تحتوي على اسطر من النصوص . متى اردت الوصول إلى ملف في **باسكال** يجب عليك ان تنشئ متغير ملف اولاً

كود

```
program Files;

var
  f: Text;

begin
end.
```

بعد الاعلان عن المتغير يجب ان تسند اسم الملف للمتغير :

كود

```
program Files;

var
  f: Text;

begin
  Assign(f,'MyFile.txt');
end.
```

لإنشاء ملف جديد فارغ استخدم الامر Rewrite . هذا الامر ينشئ اي ملف غير موجود .

كود

```
program Files;

var
  f: Text;

begin
  Assign(f,'MyFile.txt');
  Rewrite(f);
end
```

الامرين Write و Writeln يعملون على الملفات على طريقه واحده على الشاشه بإستثناء البارامتر الاضافي الذي يخبرهم للكتابه في الملفات :

كود

```
program Files;

var
  f: Text;

begin
  Assign(f,'MyFile.txt');
  Rewrite(f);
  Writeln(f,'A line of text');
end.
```

اذا كنت تريد قرائه محتويات ملف موجود سابقاً اولاً يجب ان تستخدم الامرReset بدلاً من Rewrite . ثانيا استخدم الامر Readln لقرائه السطور من الملف . سوف تحتاج إلى التكرار while لقرائه جميع السطور .

كود

```
program Files;

var
  f: Text;
  s: String;

begin
  Assign(f,'MyFile.txt');
  Reset(f);
  while not eof(f) do
     Readln(f,s);
end.
```

الامر Append يفتح الملف و يسمح لك ان تكتب المزيد من النصوص في آخر الملف.

كود

```
program Files;

var
  f: Text;
  s: String;

begin
  Assign(f,'MyFile.txt');
  Append(f);
  Writeln('Some more text');
end.
```

يجب عليك اغلاق الملف بعد الانتهاء من استخدامه . اذا لم تغلقه قد تخسر بعض النصوص التي قمت بكتابتها فيه .

كود

```
program Files;

var
  f: Text;
  s: String;

begin
  Assign(f,'MyFile.txt');
  Append(f);
  Writeln('Some more text');
  Close(f);
end.
```

يمكنك تغير اسم ملف مع الامر Rename و حذف الملفات مع الامر Erase

كود

```
var
  f: Text;

begin
  Assign(f,'MyFile.txt');
  Rename(f,'YourFile.txt');
  Erase(f);
  Close(f);
end.
```

للبحث عن الملف اذا كان موجود ام لا ، يجب اولاً ان تتبع ذلك بإستخدام {$I-} . بعد ذلك يجب ان تقوم بإستخدام الامر Reset على الملف بعدها استخدام الجمله الشرطيه if IOResult = 2 then و معناها اذا كان الملف غير موجود ، و استخدام الشرط If IOResult = 0 و معناها اذا كان الملف

موجود . و اذا كانت قيمة IOResult ليست ٢ او ٠ نستخدم الامر Halt لإنهاء البرنامج . الـ IOResult سوف يفقد قيمته اذا تم استخدامه لمره واحده لذلك يجب علينا ان نضعه في متغير قبل استخدامه . يجب عليك كذلك استخدام {$I+} لإعادة تتبع الاخطاء .

كود

```
program Files;

var
  f: Text;
  IOR: Integer;

begin
  Assign(f,'MyFile.txt');
{$I-}
  Reset(f);
{$I+}
  IOR := IOResult;
  if IOR = 2 then
     Writeln('File not found');
  else
     if IOR <> 0 then
        Halt;
  Close(f);
end.
```

مصدر هذا الدرس : http://www.sepsis.za.net/programming_pascal10.html

مترجم هذا الدرس : MaaSTaaR

تعلم برمجة Pascal الدرس الحادي عشر - ملفات المعلومات

ملفات المعلومات تختلف عن ملفات النصوص بأشياء بسيطه . ملفات المعلومات هي الوصول العشوائي يعني انه لا يجب ان تقرأ سطر بعد سطر بدلاً من ذلك يمكن ان تصل إلى اي جزء من اجزاء الملف في اي وقت . و طريقة تعريف ملفات البيانات :

كود

```
program DataFiles;

var
  f: file of Byte;

begin
end.
```

يجب ان تستخدم Assign بنفس طريقة استخدامها في ملفات النصوص.

كود

```
program DataFiles;

var
  f: file of Byte;

begin
  Assign(f,'MyFile.txt');
end.
```

يمكنك استخدام Rewrite لانشاء ملف جديد غير موجود . الاختلاف بين ملفات النصوص و ملفات المعلومات عند استخدام الامر Rewrite هو ان ملفات المعلومات يمكنك القرائه منها و الكتابة عليها .

كود

```
program DataFiles;

var
  f: file of Byte;

begin
  Assign(f,'MyFile.txt');
  Rewrite(f);
end.
```

الامر Reset مثل الامر Rewrite بإستثناء انه لا ينشئ الملف اذا كان لم يكن موجوداً :

كود

```
program DataFiles;

var
  f: file of Byte;

begin
  Assign(f,'MyFile.txt');
  Reset(f);
end.
```

عندما تريد الكتابه على الملف يجب ان تستخدم الامر Write ، يجب اولاً ان تضع الذي تريد كتابته على الملف في متغير ، قبل الكتابه او القرائه من ملفات المعلومات يجب ان تستخدم الامر Seek للبحث عن المكان الصحيح لبداية الكتابه . يجب ان تتذكر ملف المعلومات يبدأ من الوضع ٠ و ليس ١ .

كود

```
program DataFiles;

var
  f: file of Byte;
  b: Byte;

begin
  Assign(f,'MyFile.txt');
  Reset(f);
  b := 1;
  Seek(f,0);
  Write(f,b);
end.
```

الامر Read يستخدم للقراءه من ملفات المعلومات .

كود

```
program DataFiles;

var
  f: file of Byte;
  b: Byte;

begin
  Assign(f,'MyFile.txt');
  Reset(f);
  Seek(f,0);
  Read(f,b);
end.
```

يجب عليك ان تغلق ملفات المعلومات بعد الانتهاء منها مثل ملفات النصوص .

كود

```
program DataFiles;

var
  f: file of Byte;
  b: Byte;

begin
  Assign(f,'MyFile.txt');
  Reset(f);
  Seek(f,0);
  Read(f,b);
  Close(f);
end.
```

الامر FileSize يمكن استخدامه مع الامر FilePos لمعرفة متى وصلت إلى نهاية الملفات . الامر FileSize يعيد الرقم الفعلي لعدد التسجيلات و يبدأ من ١ و ليس ٠ . الامر FilePos يخبرنا عن الوضع الذي هو عليه الملف الآن .

كود

```
program DataFiles;

var
  f: file of Byte;
  b: Byte;

begin
  Assign(f,'MyFile.txt');
  Reset(f);
```

```
  while FilePos(f) <> FileSize(f) do
    begin
      Read(f,b);
      Writeln(b);
    end;
  Close(f);
end.
```

الامر Truncate يحذف أي شئ في الملف من الوضع الحالي .

كود

```
program DataFiles;

var
  f: file of Byte;

begin
  Assign(f,'MyFile.txt');
  Reset(f);
  Seek(f,3);
  Truncate(f);
  Close(f);
end.
```

احد اكثر الاشياء افادة في ملفات المعلومات هو امكانية استخدام التسجيلات و ملفات المعلومات مع بعضها البعض .

كود

```
program DataFiles;
```

```
type
  StudentRecord = Record
  Number: Integer;
  Name: String;

var
  Student: StudentRecord;

begin
  Assign(f,'MyFile.txt');
  Reset(f);
  Student.Number := 12345;
  Student.Name := 'John Smith';
  Write(f,Student);
  Close(f);
end.
```

مصدر هذا الدرس : http://www.sepsis.za.net/programming_pascal11.html

مترجم الدرس : MaaSTaaR

تعلم برمجة Pascal الدرس الثاني عشر - انشاء وحداتك الخاصّه

لابد و انك تعلم ما هي الوحدات ، تذكر ذلك عندما استخدمنا الوحده crt ، قد تضطر يوماً إلى كتابة العديد من الاجراءات و الدوال . يمكنك انشاء وحدتك الخاصه ووضعك اجراءاتك و دوالك فيها .

لإنشاء وحدتك الخاصه يجب اولا ان تنشئ ملف **باسكال** جديد بإسم MyUnit.pas . السطر الاول يجب ان يبدأ بالكلمه unit و بعدها اسم وحدتك . اسم الوحده يجب ان تكون مثل اسم الملف .

كود

```
unit MyUnit;
```

في السطر التالي نقوم بإضافة الكلمه interface . بعدها يجب علينا وضع اسم الاجراء الذي سوف نستخدمه من هذه الوحده . على سبيل المثال قمنا بإنشاء داله بإسم NewReadln تشبه Readln و لكنها تسمح لك بتحديد عدد الحروف التي يمكن ادخالها .

كود

```
unit MyUnit;

interface

function NewReadln(Max: Integer): String;
```

السطر الذي يليه نضيف الكله implementation . هذا هو المكان الذي يمكنك ان تكتب في كامل شيفرتك و اجراءاتك و دوالك . سوف نحتاج إلى استخدام الوحده crt لإنشاء NewReadln . بعد الانتهاء من الوحده يجب ان نكتب الامر end للانتهاء .

كود

```
unit MyUnit;
```

```
interface

function NewReadln(Max: Integer): String;

implementation

function NewReadln(Max: Integer): String;
var
  s: String;
  c: Char;
begin
  s := '';
  repeat
    c := ReadKey;
    if (c = #8){#8 = BACKSPACE} and (s >< '') then
      begin
        Write(#8+' '+#8);
        delete(s,length(s),1);
      end;
    if (c >< #8) and (c >< #13){#13 = ENTER} and (length(s) < Max) then
      begin
        Write(c);
        s := s + c;
      end;
  until c = #13;
  NewReadln := s;
end;
end.
```

بعدها قم بتخزين الوحده . الآن يمكننا استخدام وحدتنا MyUnit التي قمنا بإنشائها من خلال استدعاء الوحده و استخدام الداله التي فيها NewReadln .

كود

```
program MyProgram;

uses
  MyUnit;

var
  s: String;

begin
  s := NewReadln(10);
end.
```

مصدر هذا الدرس : http://www.sepsis.za.net/programming_pascal12.html

مترجم هذا الدرس : MaaSTaaR

كلمه اخيره بواسطة المترجم :

هكذا نكون قد انتهينا و لله الحمد من سلسلة دروس لغة البرمجه **باسكال** ، آملين انكم قد استفدتم و استمتعتم من هذه السلسله ، اعتذر على اي خطأ في الترجمه و دعواتكم .

الدورة من http://www.arabteam2000-forum.com