الدرس التاسع

سوف نتناول في هذه المحاضرة مايلي :

١. صياغة الخوارزميات : تكرار العداد المحدد.
٢. صياغة الخوارزميات : تكرار الحارس المحدد من قبل المستخدم.
٣. صياغة الخوارزميات : عبارات التحكم المتداخلة.
٤. عمليات الإسناد.
٥. العملية المزيدة والمنقصة .

بسمه نبدأ وبه نستعين

صيغة الخوارزميات : تكرار العداد المحدد

لتوضيح هذه الخوارزمية بشكل أفضل ، نأخذ المثال التالي :

لدينا صنف يتكون من عشر طلاب ، وتم أخذ درجاتهم (تكون بين الصفر والمائة) إحسب واعرض المجموع الكلي للدرجات ومتوسط الدرجات (المعدل)

كما هو معلوم ، المفهوم الرياضي ، أن المعدل هو مجموع الدرجات مقسوم على عدد الطلاب في الخوارزم يقوم بإدخال كل الدرجات ، ويحسب المعدل ويطبع النتيجة.

خوارزم شبه الكود مع تكرار العداد المحدد:

كود ١-٧

١

كود

```
// Fig. 4.8: GradeBook.h
2  // Definition of class GradeBook that determines a class average.
3  // Member functions are defined in GradeBook.cpp
4  #include <string> // program uses C++ standard string class
5  using std::string;
6
7  // GradeBook class definition
8  class GradeBook
```

```
9  {
10 public:
11    GradeBook( string ); // Constructor initializes course name
12    void setCourseName( string ); // function to set the course name
13    string getCourseName(); // function to retrieve the course name
14    void displayMessage(); // display a welcome message
15    void determineClassAverage(); // averages grades entered by the user
16 private:
17    string courseName; // course name for this GradeBook
18 }; // end class GradeBook
```

كود ٢-٧

كود

```
1  // Fig. 4.9: GradeBook.cpp
2  // Member-function definitions for class GradeBook that solves the
3  // class average program with counter-controlled repetition.
4  #include <iostream>
5  using std::cout;
6  using std::cin;
7  using std::endl;
8
9  #include "GradeBook.h" // include definition of class GradeBook
10
11 // Constructor initializes courseName with string supplied as argument
12 GradeBook::GradeBook( string name )
13 {
14    setCourseName( name ); // validate and store courseName
15 } // end GradeBook Constructor
16
```

```
17 // function to set the course name;
18 // ensures that the course name has at most 25 characters
19 void GradeBook::setCourseName( string name )
20 {
21    if ( name.length() <= 25 ) // if name has 25 or fewer characters
22       courseName = name; // store the course name in the object
23    else // if name is longer than 25 characters
24    { // set courseName to first 25 characters of parameter name
25       courseName = name.substr( 0, 25 ); // select first 25 characters
26       cout << "Name \"" << name << "\" exceeds maximum length (25).\n"
27          << "Limiting courseName to first 25 characters.\n" << endl;
28    } // end if...else
29 } // end function setCourseName
30
31 // function to retrieve the course name
32 string GradeBook::getCourseName()
33 {
34    return courseName;
35 } // end function getCourseName
36
37 // display a welcome message to the GradeBook user
38 void GradeBook::displayMessage()
39 {
40    cout << "Welcome to the grade book for\n" << getCourseName() << "!\n"
41       << endl;
42 } // end function displayMessage
43
44 // determine class average based on 10 grades entered by user
45 void GradeBook::determineClassAverage()
46 {
47    int total; // sum of grades entered by user
48    int gradeCounter; // number of the grade to be entered next
49    int grade; // grade value entered by user
50    int average; // average of grades
51
52    // initialization phase
```

```
53    total = 0; // initialize total
54    gradeCounter = 1; // initialize loop counter
55
56    // processing phase
57    while ( gradeCounter <= 10 ) // loop 10 times
58    {
59       cout << "Enter grade: "; // prompt for input
60       cin >> grade; // input next grade
61       total = total + grade; // add grade to total
62       gradeCounter = gradeCounter + 1; // increment counter by 1
63    } // end while
64
65    // termination phase
66    average = total / 10; // integer division yields integer result
67
68    // display total and average of grades
69    cout << "\nTotal of all 10 grades is " << total << endl;
70    cout << "Class average is " << average << endl;
71 } // end function determineClassAverage
```

كود ٣-٧

كود

```
1  // Fig. 4.10: fig04_10.cpp
2  // Create GradeBook object and invoke its determineClassAverage function.
3  #include "GradeBook.h" // include definition of class GradeBook
4
5  int main()
6  {
7     // create GradeBook object myGradeBook and
8     // pass course name to Constructor
9     GradeBook myGradeBook( "CS101 C++ Programming" );
```

```
10
11    myGradeBook.displayMessage(); // display welcome message
12    myGradeBook.determineClassAverage(); // find average of 10 grades
13    return 0; // indicate successful termination
14 } // end main
```

نستخدم شبه الكود لوضع جميع الإجراءات في قائمة مرتبة. نحن نستخدم تكرار العداد المحدد لإدخال الدرجات في كل مرة درجة واحدة. في طريقتنا هذه نستخدم
متغير يسمى العداد counter. والعداد هو عبارة عن متغير يتم تحديده مسبقا بحيث يمثل عدد التكرارات في حلقة التنفيذ.

في هذا المثال ، التكرار ينتهي عندما التكرار يتجاوز العشرة.

فكرة مفيدة في هندسة البرمجيات :

من واقع الخبرة ، العقبة في تطوير أو انشاء برنامج هو كتابة الخوارزم لتنفيذ هذا البرنامج ، بحيث يكون مستقيم ومباشر ، وخطواته واضحة.

لاحظ الخوارزم في الشكل التالي :

كود

```
1  Set total to zero
2  Set grade counter to one
3
4  While grade counter is less than or equal to ten
5     Prompt the user to enter the next grade
6     Input the next grade
7     Add the grade into the total
8     Add one to the grade counter
9
10 Set the class average to the total divided by ten
11 Print the total of the grades for all students in the class
12 Print the class average
```

المجموع عبارة عن متغير يستخدم لجمع المجموع لعدة قيم, العداد عبارة عن متغير يقوم بالعد كل مرة (في هذه الحالة ، عدد المرات يكون ١٠).
كما تلاحظون ، فإن المجموع من المفترض أن يحتوي على قيمة مخزنة مسبقا في موقع الذاكرة total.

تحسين التدقيق في الصنف :
قبل شرح خوارزم الصنف، لدينا صنفنا GradeBook في الشكل ٣.١٦ ، دالة العضو setCourseName تدقق في إسم الدورة وتقارن عدد رموزة بالعدد ٢٥.
ماسوف نفعله نستبدل عبارة if بعبارة if ... else في السطور (٢١-٢٨) الكود ٢-٧ .

تطبيق تكرار العداد المنظم في الصنف GradeBook:
الصنف GradeBook (الكود ٢-٧ ،الكود ٢-٧) يحتوي باني (مصرح عنه في السطر ١١ من الكود ١-٧ ومعرف في السطور ١٢- ١٥ من الكود ٢-٧) ذلك يسند القيمة لـ courseName
متغير المثال للصنف (معلن عنه في السطر ١٧ الكود ١-٧) السطور ١٩-٢٩ ، ٣٢-٣٥ ، ٣٨-٤٢ من الكود ٢-٧ تعرف دالة العضو setCourseName و getCour , diplay على
التوالي.السطور ٤٥- ٧١ تعرف دالة العضو determineClassAverage ، التي تطبق خوارزم معدل الصنف الذي تم وصفه بشبه الكود في الشكل ٧ز٤.

السطور ٤٧-٥٠ نعلن عن متغيرات محلية average, total, grade , gradeCounter، لتكون من نوع عدد صحيح. المتغير grade يخزن مايدخله المستخدم. لاحظ تلك التصريحات السابقة تظهر في جسم دالة العضو determineClassAverage - نحن لا نحفظ أي معلومات حول درجات الطالب في متغيرات عينة الصنف.

لممارسة البرمجة بشكل جيد :
فصل الإعلانات عن العبارات الأخرى في الدوال بسطر فارغ لتسهيل قراءة الكود

في السطور ٥٣-٥٤ متغير المجموع الكلي total يعطى قيمة أولية وهي الصفر والـ gradeCounter يعطى القيمة (١) . لاحظ ذلك ان المتغيرات total والـ gradecounter معطاه قيم أولية قبل أن أن يتم استخدامهم في العمليات الحسابية. متغيرات العداد عادة تكون معطاه قيمة أولية للصفر أو الواحد يعتمد على طريقة استخدامه. المتغير grade و averageلا يحتاجان إعطاء قيم أوليه هنا

خطأ برمجي شائع :
عدم إعطاء العداد قيمة أولية والـ total يؤدي إلى خطأ منطقي.

فكرة مفيدة في منع الاخطاء :
أعطاء قيمة أولية للعداد والمجموع ، أما عند الإعلان عنهم أو بعبارة اسناد أخرى . المجموع عادةً يكون مسند للصفر . العداد عادة يكون مسند للصفر أو الواحد، يعتمد على كيفية استخدامه

لممارسة البرمة بشكل جيد :
الإعلان عن كل متغي في سطر منفصل مع التعليق عليه يجعل البرنا أكثر وضوح.

في السطر ٥٧ ، العبارة while سوف تتكرر طالما قيمة الـ gradeCounter اصغر من أو يساوي الـ ١٠.

ملاحظات على قسمة وتقريب العدد الصحيح :
في حساب المعدل في الدالة determineClassAverage الناتج من العملية total/10 ستعطي النتيجة ٨٤ بدلا من ٨٤.٦ (سطر ٦٦ الكود ٢-٧).

خطأ برمجي شائع :
انتهاء قيمة العداد (العداد المعلوم القيمة) بقيمة تفوق القيمة المحدده ، مثلا لو لدينا عدد طلاب عشرة ، وانتهى العداد بالقيمة ١١ ، لذلك خرج من حلقة التكرار لأنه نافى الشرط.

الحسابات التي تنتج عن هذا العدد تكون عادة خاطئة ، والأخطاء تسمى off_by_one_error.

صياغة الخوارزميات : تكرار الحارس المحدد

إذا اردنا أن نجعل برنامج الدرجات عام وغيرمحدد عدد الطلاب فيه نأخذ المثال التالي :

قم بإنشاء برنامج لحساب معدل الدرجات المدخلة لعدد اختياري (من قبل المستخدم) من الطلاب .

كما تلاحظون ، لا يوجد أي تحديد أو اشارة إلى عدد الطلاب كما في المثال السابق، إذن كيف ينتهي هذا البرنامج ؟
ينتهي هذا البرنامج عن طريق المستخدم ، حيث تحدد قيمة معينة عند ادخالها ينتهي البرنامج ، وتسمى هذه القيمة بـ "قيمة الحارس".
تسمى قيمة الحارس في مواطن اخرى بـ " قيمة العلم " و " قيمة الإشارة "
يؤخذ في عين الإعتبار ، أنه لا يوجد احتمال أن قيمة الحارس يمكن ان تكون ضمن المدخلات ، وأيضا أن لا تخزن مثل البيانات
وأن تطبع مع المخرجات ، فهي لها وظيفة واحدة وهي إنهاء التكرار.

إنشاء خوارزم شبه الكود للبرنامج السابق ، مع توضيح خطوات تحسينة التدريجية

المرحلة الأولى :
بدون تفصيل ، نريد أن نختصر وظيفة البرنامج قدر الإمكان ، وفي برنامجنا السابقة لا يمكن اختصار خوارزم شبه الكود
في عبارة مفردة واحدة ، يأتي تفصيلها فيما بعد لتحسين الخوارزم وإستخلاص كود C++ منه.

كود

determine the class average for quiz

المرحة الثانية :

التفصيل التمهيدي للعبارة السابقة ، يجب أن نحللها بطريقة منطقية وبالإهتمام في تنظيم الخطوات

كود

Initialize variables

Input, sum and count the quiz grades

Calculate and print the total of all student grades and the class average

هذا التحسين يستخدم فقط مع التركيب المتسلسل - قائمة الخطوات يجب أن تنفذ بالترتيب، واحد تلو الأخرى .

فكرة مفيدة في هندسة البرمجيات :

الكثير من البرمج تكون مقسمة منطقيا إلى ثلاث مراحل : مرحلة إعطاء قيم أولية ، مرحلة المعالجة التي تضمن إدخال البيانات ، وتبسيط متغير البرنامج (مثل العدادات والمجموعات الكلية) بشكل متماثل، ومرحلة الإنها التي تحسب وتخرج النتائج النهائية.

التحسين النهائي :

في هذا التحسين ، ندخل في تفاصيل كل عبارة وتبسيط كل متغير وكتابة كافة العمليات التي تجرى عليه.

في هذه الخطوة ، فصلنا جميع عبارات الكود السابق لنحصل على الخوارزم بالشكل التالي:

كود

1 Initialize total to zero

```
2  Initialize counter to zero
3
4  Prompt the user to enter the first grade
5  Input the first grade (possibly the sentinel)
6
7  While the user has not yet entered the sentinel
8        Add this grade into the running total
9        Add one to the grade counter
10       Prompt the user to enter the next grade
11       Input the next grade (possibly the sentinel)
12
13 If the counter is not equal to zero
14       Set the average to the total divided by the counter
15       Print the total of the grades for all students in the class
16       Print the class average
17 else
18       Print "No grades were entered"
```

خطا برمجي شائع :

محاولة القسمة على الصفر عادة ما يسبب خطأ وقت التشغيل فادح .

لمنع حدوث الخطأ :

فحص القيمة التي في المقام بحيث يعطي البرنامج رسالة إذا كان صفر .

ملاحظة : الكثير من الخوارزميات تحتاج لخطوات تحسين كثيرة ، ولكننا هنا أعطينا مجرد مثال لتبسيط الطريقة.

ملاحظة في هندسة البرمجيات :

الكثير من المبرمجين يحرصون على تعلم طريقة كتابة الخوارزميات ، وتمرسوا عليها في بداية تطبيقاتهم

ولكنهم حينما اكتسبوا الخبرة والتمرس في البرمجة ، لا يقومون بكتابتها لأن ذلك يؤدي إلى تأخير إنتاجهم

دون جدوى ، ويؤدي أيضا إلى وجود تعقيدات ومشاكل لا داعي لها.

تطبيق التكرار المحدد بالحارس في الصنف GradeBook:

الكود ٧-٤

كود

```
1  // Fig. 4.12: GradeBook.h
2  // Definition of class GradeBook that determines a class average.
3  // Member functions are defined in GradeBook.cpp
4  #include <string> // program uses C++ standard string class
5  using std::string;
6
7  // GradeBook class definition
8  class GradeBook
9  {
10 public:
11    GradeBook( string ); // Constructor initializes course name
12    void setCourseName( string ); // function to set the course name
13    string getCourseName(); // function to retrieve the course name
14    void displayMessage(); // display a welcome message
15    void determineClassAverage(); // averages grades entered by the user
16 private:
17    string courseName; // course name for this GradeBook
18 }; // end class GradeBook
```

الكود ٧-٥

كود

```
1  // Fig. 4.13: GradeBook.cpp
2  // Member-function definitions for class GradeBook that solves the
3  // class average program with sentinel-controlled repetition.
```

```
4  #include <iostream>
5  using std::cout;
6  using std::cin;
7  using std::endl;
8  using std::fixed; // ensures that decimal point is displayed
9
10 #include <iomanip> // parameterized stream manipulators
11 using std::setprecision; // sets numeric output precision
12
13 // include definition of class GradeBook from GradeBook.h
14 #include "GradeBook.h"
15
16 // Constructor initializes courseName with string supplied as argument
17 GradeBook::GradeBook( string name )
18 {
19    setCourseName( name ); // validate and store courseName
20 } // end GradeBook Constructor
21
22 // function to set the course name;
23 // ensures that the course name has at most 25 characters
24 void GradeBook::setCourseName( string name )
25 {
26    if ( name.length() <= 25 ) // if name has 25 or fewer characters
27       courseName = name; // store the course name in the object
28    else // if name is longer than 25 characters
29    { // set courseName to first 25 characters of parameter name
30       courseName = name.substr( 0, 25 ); // select first 25 characters
31       cout << "Name \"" << name << "\" exceeds maximum length (25).\n"
32          << "Limiting courseName to first 25 characters.\n" << endl;
33    } // end if...else
34 } // end function setCourseName
35
36 // function to retrieve the course name
37 string GradeBook::getCourseName()
38 {
39    return courseName;
```

```
40 } // end function getCourseName
41
42 // display a welcome message to the GradeBook user
43 void GradeBook::displayMessage()
44 {
45    cout << "Welcome to the grade book for\n" << getCourseName() << "!\n"
46       << endl;
47 } // end function displayMessage
48
49 // determine class average based on 10 grades entered by user
50 void GradeBook::determineClassAverage()
51 {
52    int total; // sum of grades entered by user
53    int gradeCounter; // number of grades entered
54    int grade; // grade value
55    double average; // number with decimal point for average
56
57    // initialization phase
58    total = 0; // initialize total
59    gradeCounter = 0; // initialize loop counter
60
61    // processing phase
62    // prompt for input and read grade from user
63    cout << "Enter grade or -1 to quit: ";
64    cin >> grade; // input grade or sentinel value
65
66    // loop until sentinel value read from user
67    while ( grade != -1 ) // while grade is not -1
68    {
69       total = total + grade; // add grade to total
70       gradeCounter = gradeCounter + 1; // increment counter
71
72       // prompt for input and read next grade from user
73       cout << "Enter grade or -1 to quit: ";
74       cin >> grade; // input grade or sentinel value
75    } // end while
```

```
76
77    // termination phase
78    if ( gradeCounter != 0 ) // if user entered at least one grade...
79    {
80       // calculate average of all grades entered
81       average = static_cast< double >( total ) / gradeCounter;
82
83       // display total and average (with two digits of precision)
84       cout << "\nTotal of all " << gradeCounter << " grades entered is "
85          << total << endl;
86       cout << "Class average is " << setprecision( 2 ) << fixed << average
87          << endl;
88    } // end if
89    else // no grades were entered, so output appropriate message
90       cout << "No grades were entered" << endl;
91 } // end function determineClassAverage
```

الكود ٦-٧

كود

```
1  // Fig. 4.14: fig04_14.cpp
2  // Create GradeBook object and invoke its determineClassAverage function.
3
4  // include definition of class GradeBook from GradeBook.h
5  #include "GradeBook.h"
6
7  int main()
8  {
9     // create GradeBook object myGradeBook and
10    // pass course name to Constructor
```

```
11    GradeBook myGradeBook( "CS101 C++ Programming" );
12
13    myGradeBook.displayMessage(); // display welcome message
14    myGradeBook.determineClassAverage(); // find average of 10 grades
15    return 0; // indicate successful termination
16 } // end main
```

البرنامج السابق يعرض لنا صنف الـ GradeBook الذي يحتوي على دالة العضو الممثلة في الخوارزم السابق .
عند معرفة مهمة البرنامج فإننا نتوقع أن يخرج لنا النتيجة تحتوي على نقطة عشرية لتوضيح المعدل بشكل دقيق (نتيجة ٣.٤٥)، لكن لن يحدث
هذا مع المتغيرات من النوع int لأن حاصل قسمة عددين صحيحين سوف يعطي عدد غير صحيح (بدون كسور). لذلك وجدت عملية
الـ cast والمتغيرات من نوع float والـ double. فعن طريق هذه التقنيات سوف نحصل على اعداد دقيقة تحتوي على
نقطة عشرية.

تمثيل الأرقام الدقيقة التي تحتوي على فواصل ومتطلبات في الذاكرة :
في السطر ٥٥ من البرنامج السابق ، تلاحظون لدينا نوع جديد من البيانات وهو النوع double
قبل شرح هذا النوع ، نريد ان نذكر نوع يماثله تماما وهو الـ float وهو نوع عرف قبل الـ double وتمسية
الـ double على اساسه. هذا النوع (float) يمثل الأعداد الحقيقية بفواصل عشرية تقدر بسبع خانات على جميع انظمة
الـ ٣٢ بت . ويخزن لهذا النوع حجم ذاكرة يفوق الحجم المخزن للأعداد الصحيحة (int). نأتي لنوعنا السمتخدم في هذا
البرنامج وهو النوع (double) ، هذا النوع هو نفس النوع float ولكنه يفوقه في المواصفات (ضعف الذاكرة و ضعف
عدد الخانات).
الأن عند طباعة نتيجة المعدل ، سوف يظهر لنا المعدل بعد بفاصله عشرية وخانات.

التحويل بين الأناع الأساسية بوضوح وبشكل كلي :
ذكرنا فيما سبق أن قسمة عددين صحيحين سوف يكون الناتج عدد صحيح ، ورأينا أننا عرفنا الـ average بأنه عدد حقيقي
فحتما النتيجة سوف تحتوي على خطأ (وهو فقدها للكسور) ولتظمين الكسور في ذلك يجب التحويل بين النوعين (ناتج القسمة ونوع المتغير المسند إليه)
في العبارة average = total / grandCounter. لذلك سوف نقوم بإجراء عملية الـ cast
الـ C++ تعطي عملية التحويل الأحادية لإجراء عملية التحويل بين طرفي التعبير الحسابي السابق (السطر ٨٨).
عملية الـتحويل (casting) تنشي نسخة موقته (من نوع double) للمتغير total لإنجاز هذه المهمة. وتلقائيا سوف يتحول
الـ gradeCounter إلى عدد حقيقي ، فيحصل التوافق بين الطرفين.

خطأ برمجي شائع :
عملية الـ cast يمكن ان تستخدم للتحويل بين الأنواع الأساسية ، مثل الـ int والـ double وأنواع المترابطة (ذات العلاقة مع بعض) من الأصناف.

اذا تم التحويل بين نوعين غير تلك ، فسوف ينتج خطأ في الترجمة أو وقت التشغيل (على حسب النظام المستخدم).

في السطر ٨٦ هناك استدعاء للدالة setprecision وتم تمرير القيمة ٢ لها . هذه الدالة تقوم بإعداد العدد المطبوع بعدها لأن يحتوي على خانتين فقط بعد الفاصلة
وهذا ماتمثله القيمة المرسلة .
في نفس السطر السابق ايضا ، هناك دالة اخرى اسمها fixed وهي تقوم بإعداد العدد لظهورة بشكل علمي (بأن يكون مقرب ومختصر).
الدالتين السابقتين (fixed والـ setprecision) يتطلبان وجود ملف الترويسة iomanip (السطر ١٠).

صياغة خوارزميات : عبارات التحكم المتداخلة

هيكلة عبارات تحكم متداخلة (ومترابطة) ، وليكن لدينا المهمة التالية :

" برنامج يقوم بحساب عدد الطلاب الذين اجتازو الإختبار ، ويقوم المستخدم بإدخال ١ إذا الطالب اجتاز الإختبار ، و ٢ إذا الطالب لم يجتز الإختبار "

تحليل البرنامج كالآتي :

١. أدخل نتيجة كل إختبار (١ أو ٢) ، حيث يقوم البرنامج بعرض رسالة تطلب ذلك ، صيغتها " enter the result ".
٢. قم بحساب عدد النتائج (نتائج الإختبار).
٣. اعرض الخلاصة ، عدد الطلاب الذين اجتازو واللذين لم يجتازو.
٤. اذا كان عدد الذين اجتازو اكثر من ٨ فإن البرنامج يطبع رسالة صيغتها كالآتي : " Raise tuition".

بعد قراءة المهمة ، نهتم بالآتي :

١. البرنامج سوف يقوم بفحص كل نتيجة لعشرة طلاب . ويتم تكرار العداد المحدد بواحد كل مرة (وذلك لأن عدد التكرار محدد سلفا).
٢. نتيجة الإختبار إما ان تكون واحد أو اثنين . اذا الرقم ليس واحد ، تلقائيا نفترض أنه يكون ٢.
٣. عدادين لحفظ مسار نتائج الإختبار - واحد لحساب عدد اللذين اجتازو ، والآخر للذين لم يجتازو.
٤. بعدها يقرر البرنامج ، أي العددين أكثر (اللذين اجتازو واللذين لم يجتازو).

الخطوة الأولى لإنشاء خوارزم البرنامج ، عبارة مختصرة لمهمة البرنامج :

كود

Analyze exam results and decide whether tuition should be raised

المرحلة التمهيدية لإنهاء الخوارزم :

التفصيل التمهيدي للعبارة السابقة

يجب أن نحللها بطريقة منطقية وبالإهتمام في تنظيم الخطوات

كود

Initialize variables

Input the 10 exam results, and count passes and failures

Print a summary of the exam results and decide if tuition should be raised

التحسين النهائي :

في هذا التحسين ، ندخل في تفاصيل كل عبارة وتبسيط كل متغير وكتابة كافة العمليات التي تجرى عليه.

في هذه الخطوة ، فصلنا جميع عبارات الكود السابق لنحصل على الخوارزم بالشكل التالي:

كود

```
1  Initialize passes to zero
2  Initialize failures to zero
3  Initialize student counter to one
4
5  While student counter is less than or equal to 10
6      Prompt the user to enter the next exam result
7      Input the next exam result
8
9      If the student passed
10         Add one to passes
11     Else
12         Add one to failures
```

```
13
14      Add one to student counter
15
16  Print the number of passes
17  Print the number of failures
18
19  If more than eight students passed
20      Print "Raise tuition"
```

سوف نقوم فيما يلي بتحويل الخوارزم السابق إلى كود C++، كالآتي :

كود ٧-٧

كود

```
1  // Fig. 4.16: Analysis.h
2  // Definition of class Analysis that analyzes examination results.
3  // Member function is defined in Analysis.cpp
4
5  // Analysis class definition
6  class Analysis
7  {
8  public:
9     void processExamResults(); // process 10 students' examination results
10 }; // end class Analysis
```

كود ٧-٨

كود

```
1  // Fig. 4.17: Analysis.cpp
2  // Member-function definitions for class Analysis that
3  // analyzes examination results.
4  #include <iostream>
5  using std::cout;
6  using std::cin;
7  using std::endl;
8
9  // include definition of class Analysis from Analysis.h
10  #include "Analysis.h"
11
12  // process the examination results of 10 students
13  void Analysis::processExamResults()
14  {
15     // initializing variables in declarations
16     int passes = 0; // number of passes
17     int failures = 0; // number of failures
18     int studentCounter = 1; // student counter
19     int result; // one exam result (1 = pass, 2 = fail)
20
21     // process 10 students using counter-controlled loop
22     while ( studentCounter <= 10 )
23     {
24        // prompt user for input and obtain value from user
25        cout << "Enter result (1 = pass, 2 = fail): ";
26        cin >> result; // input result
27
28        // if...else nested in while
29        if ( result == 1 )          // if result is 1,
30           passes = passes + 1;      // increment passes;
31        else                        // else result is not 1, so
32           failures = failures + 1;  // increment failures
33
```

```
34       // increment studentCounter so loop eventually terminates
35       studentCounter = studentCounter + 1;
36    } // end while
37
38    // termination phase; display number of passes and failures
39    cout << "Passed " << passes << "\nFailed " << failures << endl;
40
41    // determine whether more than eight students passed
42    if ( passes > 8 )
43       cout << "Raise tuition " << endl;
44 } // end function processExamResults
```

كود ٧-٩

كود

```
1  // Fig. 4.18: fig04_18.cpp
2  // Test program for class Analysis.
3  #include "Analysis.h" // include definition of class Analysis
4
5  int main()
6  {
7     Analysis application; // create Analysis object
8     application.processExamResults(); // call function to process results
9     return 0; // indicate successful termination
10 } // end main
```

يمكنك في كل مرة تنفيذ الكود ورؤية المخرجات

عمليات الإسناد

الـ C++ تعطي عدة عمليات لإختصار عمليات الإسناد مثلا ، العبارة c = c + 3 يمكن أن نختصرها كالآتي c += 3 عن طريقة عملية اسناد الجمع (=+).
وظيفة هذه العملية هي اضافة القيمة التي على اليسار إلى القيمة التي على اليمين وتسند الناتج للمتغير الذي على اليسار.
varible operator eexpression = variable:

وهكذا باقي العمليات (*، / و %، - ، وعمليات اخرى نأخذها فيما بعد) ، والشكل التالي يفصل ما سبق :

Assignment operator	Sample expression	Explanation	Assigns
Assume: `int c = 3, d = 5, e = 4, f = 6, g = 12;`			
+=	`c += 7`	`c = c + 7`	10 to c
-=	`d -= 4`	`d = d - 4`	1 to d
*=	`e *= 5`	`e = e * 5`	20 to e
/=	`f /= 3`	`f = f / 3`	2 to f
%=	`g %= 9`	`g = g % 9`	3 to g

العملية المزيدة والمنقصة

هي عمليات احادية ، اي تقوم بالتعامل مع الواحد فقط ، تزيد او تنقص عدد مقدارة ١.
نمثل العملية المزيدة كالتالي (++) ونمثل العملية المنقصة كالتالي (--).
المزيدة تزيد القيمة بواحد كل مره ، والمنقصة تنقص من القيمة المقدار واحد كل مرة.
وللتفصيل في هاتين العمليتين ، هناك اربعة انواع (اشكال لهما)، لتوضيح ذلك ، نأخذ المتغير b ونطبق كالآتي :

عملية المزيدة السابق (b++) : تقوم بزيادة المتغير بواحد ومن ثم تقوم بالعملية (الإسناد أو الطباعة).
عملية المزيدة اللاحقة (++b) : تقوم بالعملية (الإسناد أو الطباعة) ومن ثم تقوم بعملية الزيادة بواحد.
عملية المنقصة السابقة (b--) : تقوم بطرح المتغير بواحد ومن ثم تقوم بالعملية (الإسناد أو الطباعة).
عملية المنقصة اللاحقة (--b) : تقوم بالعملية (الإسناد أو الطباعة) ومن ثم تقوم بعملية الطرح بواحد.

واليكم الشكل التالي لزيادة التوضيح

Operator	Called	Sample expression	Explanation
++	preincrement	++a	Increment a by 1, then use the new value of a in the expression in which a resides.
++	postincrement	a++	Use the current value of a in the expression in which a resides, then increment a by 1.
--	predecrement	--b	Decrement b by 1, then use the new value of b in the expression in which b resides.
--	postdecrement	b--	Use the current value of b in the expression in which b resides, then decrement b by 1.

خطأ برمجي شائع :
وضع الفراغ خلال هذه العمليات.

أخيرا ، الشكل التالي يوضح اولويات وترابطيات العمليات التي اخذناها من بداية الدورة ، وهي ضرورية
اثناء تتبع البرنامج لعدم الوقوع في حساب خاطيء

خطأ!

Operators						Associativity	Type
()						left to right	parentheses
++	--	static_cast< *type* >()				left to right	unary (postfix)
++	--	+	-			right to left	unary (prefix)
*	/	%				left to right	multiplicative
+	-					left to right	additive
<<	>>					left to right	insertion/extraction
<	<=	>	>=			left to right	relational
==	!=					left to right	equality
?:						right to left	conditional
=	+=	-=	*=	/=	%=	right to left	assignment

انتهت المحاضرة
http://www.arabteam2000-forum.com