الدرس ٨

سنتناول في هذه المحاضرة

١- عبارة الإختيار المفردة if
٢- عبارة الإختيار المزدوجة if ... else .
٣- العملية الشرطية الثلاثية (؟:).
٤- عبارة التكرار while.

بسم الله وبه نستعين.

عبارة الأختيار if

تستخدم عبارات الإختيار لإختبار قيم معينة . على سبيل المثال ، افترض أننا نريد أن نختبر قيمة معينة ولتكن
درجة معينة لأحد الطلاب ، بحيث نقارنها بالـ ٦٠ ، عبارة شبه الكود سوف تكون :

كود

```
If student's grade is greater than or equal to 60
    Print "Passed"
```

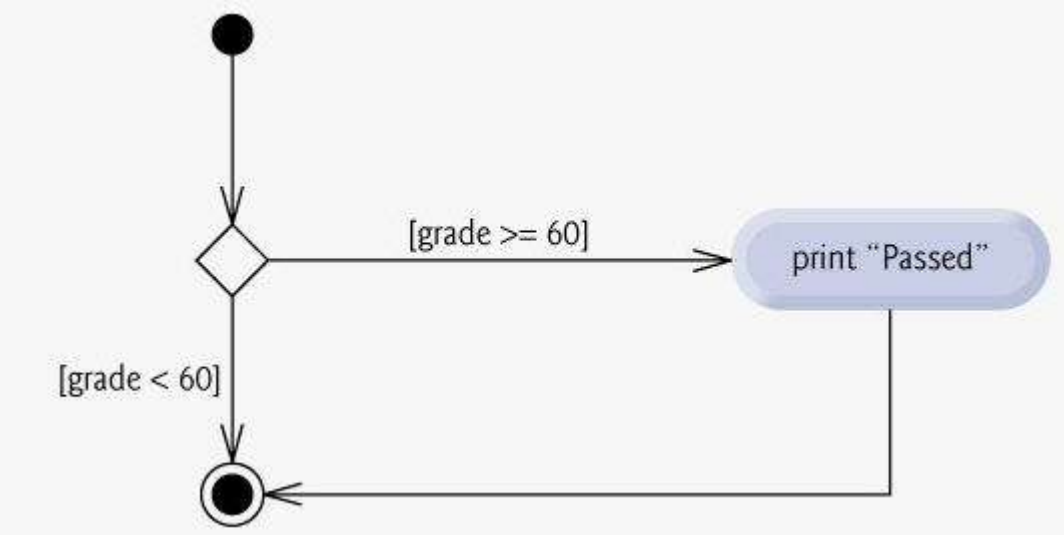
هذا السطر يرى هل درجة الطالب تساوي الـ ٦٠ أو اكبر منها ، فإذا كانت كذلك فإنه يقوم بطباعة العبارة ، واذا لم تكن لا يفعل شي.

كود C++ المناظر للسطر السابق (سطر شبه كود) كالتالي :

كود

```
if ( grade >= 60 )
    cout << "Passed";
```

الشكل ٦-١ :



الشكل السابق (٦-١) ، كما ذكرنا سابقا ، فإن هذا الرسم عبارة عن نموذج UML يمثل مخطط الحركة لسير العبارة if الماضية .
كما تلاحظون الجديد في هذا الرسم هو المعين ، وهو مفترق اتخاذ القرار ، ايضا يمكن تسميته برمز القرار . يحتوي هذ المعين على الشرط (شرط حارس) يحدد خط السير بحيث يخرج من هذا المعين ، على الأقل ، سهمين ، لكل حالة خط سير.
الرسم واضح (يمكنكم تأمله).
لو حاولنا معرفة قيمة الشرط (كل عبارة في الـ C++ تمثل قيمة) ، فسوف نتوصل إلى ان التعبير الشرطي يأخذ احدى قيمتين ، إما الصفر أو غير الصفر لذلك تم إيجاد المفتاحية (bool) وذلك يحتوي
إحدى قيمتين ، إما صحيح (عدد صحيح غير الصفر) أو خاطيء (الصفر).

عبارة الإختيار if ... else

عرفنا فيما سبق أن عبارة الإختيار if تؤدي عملية واحدة (في حال تحقق الشرط) ، ولذلك سميت بعبارة الإختيار المفردة.

العبارة if ... else عملية مزدوجة ، اي تستطيع أن تحدد مسارين (قرارين) ، قرار في حال تحقق الشرط ، وقرار عندما يكون الشرط خاطيء. على سبيل المثال ، عبارة شبه الكود التالية :

كود

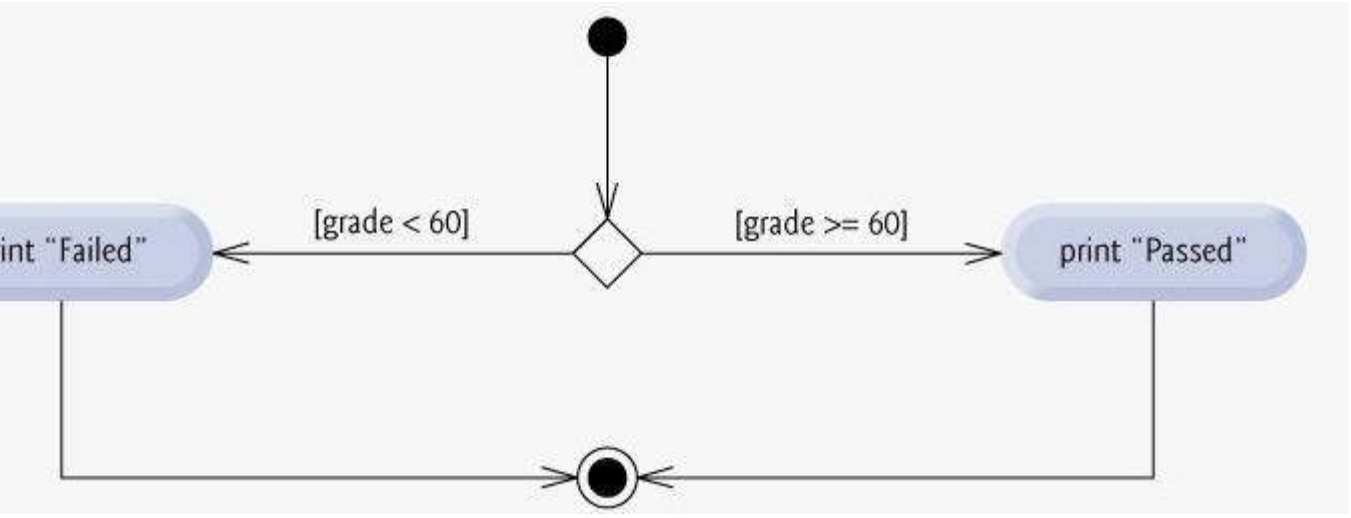
```
If student's grade is greater than or equal to 60
        Print "Passed"
    Else
        Print "Failed"
```

يتم طباعة " passed " في حال تحقق الشرط ، وإذا لم يتحقق الشرط يطبع " failed".

لممارسة البرمجة بشكل جيد :
جسم الـ else من المستحسن جدا أن يسبق بفراغ لتسهل قراءة الكود ويسهل تتبعه ، لكل مستوى يوضح أمامه فراغ يزيد عن الذي قبله (عادةً ما يكون ثلاث فراغات).

الشكل ٦-٢ :

خطأ!



الشكل السابق يوضح سير التحكم في عبارة الـ if ... else ، كما تلاحظون ، اتضح لنا خط سير جديد
بينما كان في الرسم السابق لا يوجد إلى خيار واحد ، الآن نرى إلزام في إختيار أحد المسارين (كما هو واضح من الرسم السابق).

العملية الشرطية الثلاثية (:؟)

هي عملية شرطية مشابهة إلى حد كبير بالعبارة if ... else ، وتسمى بالعملية الثلاثية (لأنها تحتوي على ثلاث معاملات)

المعامل الأول هو الشرط نفسه ، المعامل الثاني هو العبارة التي سوف تنفذ عند تحقق الشرط ، المعامل الثالث هي

العبارة التي تنفذ عندما يكون الشرط خاطيء. ولنأخذ مثال على ذلك ، لدينا عبارة الإخراج التالية :

كود

```
cout << ( grade >= 60 ? " passed " : " failed " );
```

grade >= ٦٠ : يمثل المعامل الأول وهو الشرط

" passed " : المعامل الثاني ، العبارة التي سوف تنفذ عند تحقق الشرط .

" failed " : المعامل الثالث ، العبارة التي سوف تنفذ عندما يكون الشرط خاطيء.

شرح العبارة : عندما يتحقق الشرط فإن العبارة " passed " سوف تطبع على الشاشة ، وعندما يكون

الشرط خاطيء تطبع العبارة "failed " على الشاشة.

عندما تستخدم هذه العبارة الشرطية بشكل اوسع ، سوف تلاحظ أنه يمكنك ان تستخدمها في مواضع يصعب على عبارة if ... else تحل محلها.

عبارات الـ if ... else المتداخلة :

عبارات الـ if ... else المتداخلة ، هي عبارة عن وضع عبارة مرة أخرى في الخيار else. على سبيل المثال ، في شبه الكود التالي ، سوف يقوم بطباعة " A " عندما تكون الدرجة أكبر من أو تساوي ٩٠، ايضا

يقوم بطباعة " B " عندما تكون الدرجة أكبر من أو تساوي ٨٠، يقوم بطباعة " C " عندما تكون الدرجة أكبر

من أو تساوي ٧٠، يقوم بطباعة " D " عندما تكون الدرجة أكبر من أو تساوي ٦٠، أخيرا، يقوم بطباعة " F " عندما تكون الدرجة أقل من ٦٠ .

كود

```
If student's grade is greater than or equal to 90
    Print "A"
Else
    If student's grade is greater than or equal to 80
       Print "B"
    Else
      If student's grade is greater than or equal to 70
         Print "C"
      Else
         If student's grade is greater than or equal to 60
               Print "D"
         Else
               Print "F"
```

كود C++ يمثل الخوارزم السابق.

كود

```
if ( studentGrade >= 90 ) // 90 and above gets "A"
  cout << "A";
else
  if ( studentGrade >= 80 ) // 80-89 gets "B"
    cout << "B";
  else
    if ( studentGrade >= 70 ) // 70-79 gets "C"
      cout << "C";
```

```
else
  if ( studentGrade >= 60 ) // 60-69 gets "D"
    cout << "D";
  else // less than 60 gets "F"
    cout << "F";
```

اثناء التنفيذ ، عندما يفحص الشرط أي عبارة ويتحقق أنها صحيحة ، فإنه سوف يتجاوز جميع العبارات الإخرى ، ليخرج بذلك عن العبارة الشرطية.
يفضل المبرمجي كتابة العبارة السابقة بهذا الشكل :

كود

```
if ( studentGrade >= 90 ) // 90 and above gets "A"
  cout << "A";
else if ( studentGrade >= 80 ) // 80-89 gets "B"
  cout << "B";
else if ( studentGrade >= 70 ) // 70-79 gets "C"
  cout << "C";
else if ( studentGrade >= 60 ) // 60-69 gets "D"
  cout << "D";
else // less than 60 gets "F"
  cout << "F";
```

فكرة مفيدة في تحسين الأداء :

هناك أحتمال كبير بأن الشرط في عبارة if ... else المتداخلة يتحقق في أول السلسلة ، وعندها سوف يخرج التحكم من السلسلة بأكملها ، وهذا أسرع من عبارا if ... else مفكوكة . لذلك وددنا أن نشير إلى ذلك لتحسين سرعة البرنامج.

مشكلة تعلق الـ else .. وتسمى (dangling - else problem) :

عندما يقوم المترجم بترجمة كود الـ C++ فإنه دائما ما يربط عبارة else بالعبارة التي قبلها مباشرة ،

الا

في حالات معينة وهي أنه هناك تحديد للمسار عن طريق الأقواس {} . على سبيل المثال ، لنأخذ هذا الكود :

كود

```
if ( x > 5 )
  if ( y > 5 )
    cout << "x and y are > 5";
else
  cout << "x is <= 5";
```

أنظر إلى هذا الكود ، ثم خمن سير عملة ، وماذا يعمل في كل حالة ؟ !!

أتوقع أن هذا الكود يبدو للجميع كالتالي : يتحقق التحكم من أن x أكبر من الـ ٥ ، فإذا نجح ذلك فإنه سوف

يقوم بعملية تحقق أخرى وهي (هل المتغير y اكبر من الـ ٥ أيضا) ، وعند تحقق هذا الشرط سوف يقوم بطباعة العبارة " x and y are > ٥ " .

واذا لم يتحقق الشرط الأول (x > ٥) فإن العبارة " x is <= ٥ ".

أبدا ، ليس كذلك ، الفريق العربي للبرمجةارة الـ else المتعلقه في أخر القطعة ، سوف تنسب لعبارة الـ if التي تسبقها مباشرة.

إنظر للشكل الذي سوف ينفذها المترجم وفقا له :

كود

```
if ( x > 5  )
  if ( y > 5 )
    cout << "x and y are > 5";
```

```
else
   cout << "x is <= 5";
```

لذلك من المستحسن أن نجبر التفيذ لكي نحتاط من الوقوع في الخطأ ، وذلك عن طريق وضع الأقواس كما في الشكل التالي :

كود

```
if ( x > 5 )
{
   if ( y > 5 )
      cout << "x and y are > 5";
}
else
   cout << "x is <= 5";
```

القطع (blocks) :

عن القاء نضرة على كثير من أكواد الـ C++ ، تلاحظ (سواء كان في عبارة if أو if.. else) أن كل من if أو else تحتوي على جسم ، يحتوي هذا الجسم على عبارة واحدة على الأقل ، وغالبا ما يكون محاط

بالأقواس {} . يسمى هذا الجسم بالقطعة (block) في معظم الأحوال وأحيانا يسمى بالـ العبارات

المركبة compound statements. مثال على ذلك :

كود

```
if ( studentGrade >= 60 )
   cout << "Passed.\n";
```

```
else
{
   cout << "Failed.\n";
   cout << "You must take this course again.\n";
}
```

في الـ else نرى قطعة تحتوي على عبارتين .
في هذه الحالة ، اذا لم يتحقق الشرط فإن العبارة سوف تنفذ هاتين العباراتين .
لنتخيل أنه لا يوجد أقواس {} ، فإن العبارة الثانية في القطة سوف تنفذ في كل الأحوال.

خطأ برمجي شائع :
نسيان أحد الأقواس أو كلاهما للقطعة يؤدي إلى خطأ نصي أو خطأ منطقي في البرنامج

لممارسة البرمجة بشكل جيد :
بعض المبرمجين يقومون بكتابة الأقواس قبل كتابة العبارات ، حرصا منهم على وجودها ليتجنبوا بذلك الوقوع في الخطأ .

خطأ برمجي شائع :
وضع فاصلة منقوطة بعد الشرط if أو الجزء else ، فإن ذلك يلغي العبارة ، وتصبح عبارة أحداهما فاضية (statement null).

عبارة التكرار while

تسمح هذه العبارة بتكرار جزء معين ضمن شرط معين ، في حال عدم تحقق هذا الشرط ، فإن حلقة التكرار تنكسر.

على سبيل المثال ، نأخذ شبه الكود التالي :

كود

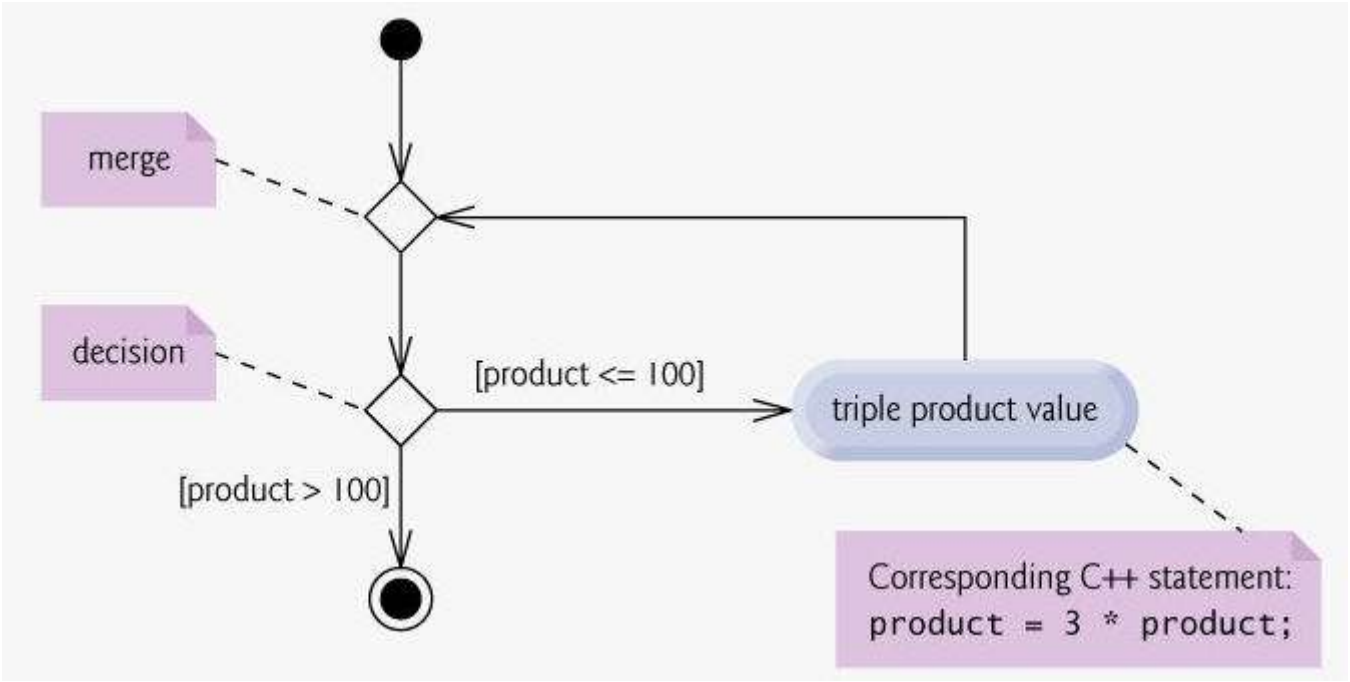
```
While there are more items on my shopping list
    Purchase next item and cross it off my list
```

نأخذ مثال شفوي أخر : لنفرض أننا نريد أن نحصل على الأسس الأولى ، بحيث يكون الناتج أقل من

١٠٠ أو يساويها .
سوف نضع عبارة التكرار while ونضع خلالها بأن الـ product <= ١٠٠ ، ونقوم في كل مره بضرب الـ product بقيمتة السابقة (اول قيمة هي ٣)
سوف نحصل في أول دورة على قيمة للـ product وهي القيمة ٩ ، وفي الدورة الثانية نحصل على القيمة ٢٧ ، ثم ٨١ ، ثم ٢٤٣ ، عند هذه النقطة لا يتحقق الشرط
وتنكسر حلقة التكرار ، وتحتوي الـ product على القيمة ٢٤٣. بعدها ينتقل التحكم إلى ما بعد العبارة while.

الشكل ٦-٣ :

خطأ!



في الشكل السابق ، قمنا بتوضيح سير حلقة التكرار السابقة ، كما تلاحظون ، هناك معينان ، المعين الأول يمثل معين الدمج
وذلك لعمل نقطة التقاء لتكوين الحلقة ، والمعين الثاني يمثل الشرط الذي يتكرر التنفيذ بناءا عليه.
ومن تجدر الإشارة إليه
هنا ، أن معين الدمج ليس له نظير عند كتابة الكود ، فهو هنا لغرض إيجاد نقطة التقاء لتكوني حلقة التكرار.

ملاحظه في الأداء :
نود التنويه هنا بأن الملاحظات حول تحسينات الأداء في ما سبق ، البعض عندما يراها صغيرة (يتصاغر تاثيرها) يهملها ولا يأخذها بالحسبان
لكن هذه الأشياء عندما تؤخذ بعين الإعتبار ، فإن الكثير من التحسين لسرعة الأداء سوف يحدث ، ولا أبالغ اذا قلت بأنها سوف تحدث تغييرا
ضخما في الأداء .

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