الفصل الرابع

جمل التكرار والتحكم فى مسار البـرنامج

Loops &Control Statements

Programming Concepts فى هذا الفصل

4

هي من أشهر الجمل في جميع لغات البرمجة حيث تسمح بتكرار عملية أو مجموعة سطور أكثر من مرة وكذلك جمل التي تحدد مسار البرنامج و تحقق رغبات المستخدم و هي جمل التحكم في مسار البرنامج حيث نشرح النقاط التالية

- جمل التكرار..... , FOR
- جمل التكرارFor Each
- الأشكال المختلفة للجملة FOR
- الجملة WHILE ...WEND
- جملة while
- جملة do ..... while

التكرار Looping

من العمليات المسموح بها في جميع اللغات عملية التكرار وهي تكرار تنفيذ عملية معينه أو مجموع سطور أكثر من مرة وهي مفيدة بدرجة انها لا يمكن الاستغناء عنها في كثير من البرامج.

وينقسم التكرار Looping إلى نوعين:

- تكرار معلوم العدد: وفيه معروف عدد مرات التكرار وتستعمل له الجملة for بأشكالها المختلفة.
- تكرار غير معلوم: واحياناً يسمى تكرار مشروط لان التكرار يتوقف على شرط معين وفيما يلي شرح لكلاً النوعين والجمل المختلفة :-

في لغة VB.NET

الجملة FOR

وتأخذ الصورة العامة التالية

```
For counter = start To end [Step step]
[statements]
[Exit For]
[statements]
Next [counter]
```

و هي تستعمل في تكرار تنفيذ عملية معينة عدد من المرات .

فمثلا لطباعة كلمة VISUAL BASIC 20 مرة سوف نكتب 20 سطر لتحقيق ذلك بدون استعمال جمل التكرار في حين تستعمل جملة تكرار مع سطر واحد لتنفيذ ذلك.

و يتضح ذلك من السطر التالي :

```
Dim i
For i = 0 To 10

   MsgBox(" Visual Basic.Net")
Next (i)
```

في هذه السطور تم استعمال جملة التكرار FOR .

لتنفيذ السطر رقم 2 عدد 10 مرات و هذا ما يوفر علينا كتابة سطر 2 عدد 10مرات ولتوضيح فيم بإنشاء برنامج جديد ووقع زر أمر Button1 وكتابة السطور السابقة به ثم نفذ البرنامج وأنقر الزر تلاحظ ظهور مربع الرسالة Msgbox بالعبارة 10 مرات.

أنواع الجمل

توجد أكثر من جملة للتكرار هي :

```
1. For ..Next
2. Do...      Loop
3. While... End While
```

جملة FOR

نستخدم جملة FOR لتكرار تنفيذ عملية عدد من المرات معلومة العدد و نأخذ أكثر من صورة نعرضها فيما يلي:

الصورة الأولى :

```
1: Dim i
2:       For i = 0 To 20
3:         MsgBox("I=" & i)
4:    Next i
```

هذه هي أبسط صور التكرار حيث تبدأ في السطر رقم 2 بكلمة FOR ثم متغير I (أي متغير) و هو العدد يبدأ بالقيمة 0 ثم TO و القيمة 20 و بالتالي يكون معني السطر رقم 2 هو كرر ما يلي ابتداء من القيمة 0 إلي 20 أي 20 مرة .

- السطر رقم 3 يقوم بطباعة قيمة المتغير i كل مرة ولكن بالصورة I=0 وهكذا.
- السطر رقم 4 كلمة NEXT التي تحدد الجمل المطلوب تكرارها.

و بالتالي لتكرار جمل معينة نكتبها بين كلمتي FOR و كلمة NEXT .

الصورة الثانية :

```
1: Dim i
2:       For i = 0 To 20 step 2
3:         MsgBox("I=" & i)
4:    Next i
```

```
K=O
DO   WHILE     K <10
Msgbox k
K=K+1
LOOP
```

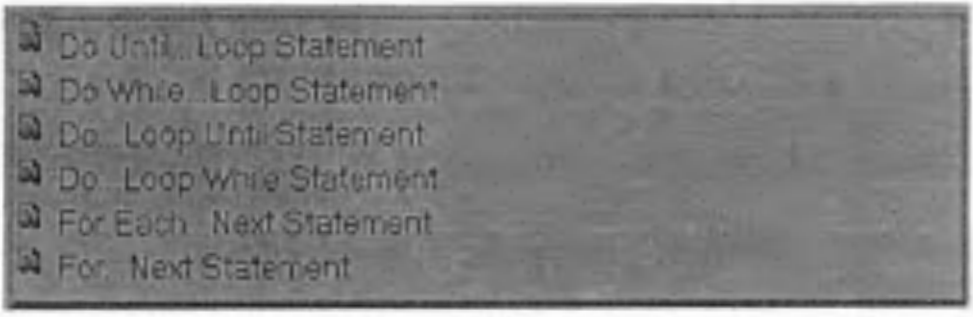
في هذه السطور يتم طباعة قيم K من 0 إلى 10 طالما أن K<10 كما في جملة الشرط.

و لكن يفضل أستعمال DO WHILE مع حالات عدم معرفة عدد المرات مثل شرط EOF الذي يعبر عن نهاية الملف الذي لا يعرف فيه عدد السجلات.

ويمكن اضافة الصيغة من قائمة Snippet التى أشرنا اليها مما يوفر عليك حفظ الصيغة

الاستفادة من الـ Snippet

توفر .NET قائمة Snippet تعرض جميع الصور المتاحة للتكرار Loops كما في الشكل (4-1)



الشكل(1-4)

وعند اختيار هذه الصور تحصل على تركيبها كما في السطور التالية

```
Do Until False

Loop
```

```
Do While True

Loop
```

```
Do

Loop Until True
```

```
Do

Loop While True
```

```
For Each Item As String In CollectionObject
Next
```

```
For index As Integer = 1 To 10
Next
```

```
While True
End While
```

فى لغات C-Like

تلتقى لغات C-Like مثل لغة Java و C# و C++ فى الأوامر وكيفية تنفيذها وأن كانت الأوامر التالية تنفذ باستعمال لغة Java مباشرة كما فى يل

التكرار باستعمال for Loop

تستعمل هذه الجملة لتكرار تنفيذ عملية اكثر من مرة وهى ابسط وأشهر أنواع جمل التكرار وتأخذ الصورة التالية:

وتأخذ الصورة العامة التالية :

```
for ([initializers]; [expression]; [iterators])
  statement
```

فى هذه الصورة تأخذ جملة for الأجزاء التالية:-

for: الأمر نفسه المستعمل فى التكرار

initializers: القيمة الابتدائية التى يبدأ بها التكرار.

expression: شرط التكرار الذى يستمر التكرار طالما هذا الشرط صحيح.

ويتم استعمال الأقواس { } لتكوين البلوك الذى ينفذ داخل التكرار وإذا كانت جملة واحدة لا تحتاج إلى أقواس.

iterators : مقدار الزيادة

statement: هى الجملة المطلوب تنفيذها داخل التكرار.

مع ملاحظه فصل إجزاء جملة for بالفاصلة المنقوطة ;

الجديد في هذه الصورة :

هو كتابة STEP 2 في السطر رقم 1 و هو تغيير خطوة العد فبدلا من أن يكون العد هو 0 و 1 و 2 و 3 يصبح 0 و 2 و 4 و 6 و 8 و هكذا و يمكنك تغيير قيمة الخطوة حسب العملية المطلوبة

الصورة الثالثة:

```
1: Dim i
2:       For i = 10 To 0 step  -1
3:         MsgBox("I=" & i)
4:   Next
```

•في السطر رقم 1 تم ابتداء العداد من القيمة 10 إلي القيمة 0 مع تحديد الخطوة بمقدار 1- (يمكن تنفيذ الخطوة) و بالتالي يصير العد بالطريقة 10 و 9 و 8 و7....

أي عد تنازلي و باقي السطور كما سبق.

الصورة الرابعة :

وتسمى هذه الصورة Nested Loops أو التكرار المتداخل حيث يتم استعمال تكرار داخل تكرار.

```
Dim i, k, m, m2, m3
      For i = 0 To 5
         For k = 0 To 4
            m = "I=" & i
            m2 = " k=" & k
            m3 = m + m2
            MsgBox(m3)
         Next (k)
      Next (i)
```

- في السطر رقم 1 بداية تكرار بجملة FOR بمتغير I يبدأمن 0 إلي 5
- في السطر رقم 2 بداية تكرار آخر بجملة FOR بمتغير K يبدأ من 0 إلي 4
- في السطر رقم 3 يتم طباعة قيم I,K وهي أول مرة 0,0
- في السطر رقم 4 جملة NEXT K التي تؤدي إلي ذهاب التكرار إلي السطر رقم 2 الذي يزيد من قيمة K لتصبح 1 ثم طباعتها في سطر 3 ثم سطر 4 يعيد التكرار مرة أخري

لزيادة قيمة K و هكذا حتي تصل قيمة K إلي 4 فينتقل التنفيذ إلي سطر 5 الذي يؤدي إلي الذهاب إلي سطر رقم 1 الذي يزيد من قيمة المتغير I لتصبح 1 ثم يعاد تنفيذ التكرار الرافع و هكذا.

كل قيمة من المتغير I يقابلها 4 قيم من المتغير K و بالتالي يكون شكل القيم كما يلي:

0,0	0,1	0,2	0,3	0,4
1,0	1,1	1,2	1,3	1,4
2,0	2,1	2,2	2,3	2,4
3,0	3,1	3,2	3,3	3,4

و هذه الصورة تسمي NESTED LOOPS أي التكرار المتداخل .

و يكثر استعمالها مع المصفوفات

يمكنك تجربة الصور المختلفة و ذلك بإنشاء برنامج جديد و توقيع زر أمر عليه و كتابة سطور الصورة داخل دالة زر الأمر ثم تنفيذ البرنامج و تجربة و مشاهدة نتيجة التنفيذ حتي يتكون لديك تصور عن معني الجمل.

جملة DO.....WHILE

الجملة DO WHILE تستخدم لتكرار عملية عدد من المرات غير معلومة العدد و لكن يعتمد هذا التكرر علي شرط WHILE حيث تأخذ الصورة العامة التالية :

```
Do [{While | Until} condition]
[statements]
[Exit Do]
[statements]
Loop
Or, you can use this syntax:
Do
[statements]
[Exit Do]
[statements]
Loop [{While | Until} condition]
```

في هذه الصورة :

يتم تكرار ما بين DO و الفرق الآخر LOOP طالما أن الشرط مكتوب أمام كلمة WHILE صحيح فإذا أصبح غير صحيح يتوقف التكرار .

ويمكن تنفيذ هذه الصورة علي التكرار العددي كما بالشكل التالي :

- ويتم استعمال التكرار for بكثرة مع المصفوفات لتجنب التعامل مع كل عنصر على حدة.

لا تنتهي جملة for أو جملة if بالفاصلة المنقوطة ; وإلا يسير البرنامج في خطوات خطأ، فمثلاً لا تكتب السطور التالية:

```
for (i =0; i<20; i++); //Logic Error
System. Out. Print In ("Java –");
```

في هذه الحالة لا يتم تنفيذ الدالة Print In() عشرة مرات بل يتم تنفيذ التكرار منفرد بدون التأثير على دالة Print In() ثم تنفذ الدالة مرة واحدة.

و هي تستعمل في تكرار تنفيذ عملية معينة عدد من المرات .

فمثلا لطباعة كلمة Java عدد 20 مرة سوف نكتب 20 سطر لتحقيق ذلك بدون استعمال جمل التكرار في حين تستعمل جملة تكرار مع سطر واحد لتنفيذ ذلك.

و يتضح ذلك من السطر التالي :

```
        for (int i = 1; i <= 10; i++)
System. Out. Print In ("Java –");
```

ولتوضيح ذلك قم بانشاء برنامج جديد واكتب سطوره كما في الشكل.

```
1  //import java.io.*;
2  class For1 {
3
4      public static void main(String[] args) {
5          // Create application frame.
6  int i;
7                  for ( i = 1; i <= 10; i++)
8          System.out.println("Java ");
9
10     }
11 }
12
```

- زيادة مقدار الزيادة .

وفيه يتم تغيير مقدار الزيادة ويأخذ الصورة التالية:

```
for(I=0;I<20;I+=2)
```

وفي هذه الصورة يتم زيادة قيمة المتغير I بمقدار 2 كل مرة.

ولتوضيح ذلك قم بتعديل البرنا مج السابق ليصبح كما فى الشكل.

```
1  class For1 {
2
3    public static void main(String[] args) {
4       // Create application frame.
5  int i;
6              for ( i = 0; i <= 20; i+=2)
7          System.out.println("   Java  i= "+i);
8
9     }
10 }
```

ف هذه السطور:

قم بترجمة وتنفيذ البرنامج لحصل على نتيجة التنفيذ كما فى الشكل.

- الطباعة من الأكبر إلى الأصغر ويأخذ الصورة التالية:

```
for(I=20;I>0;I--)
```

وفى هذه الصورة يبدأ التكرار من القيمة 20 ثم يتناقص بمقدار 1 طالما أكبر من 0

التكرار المتداخل Nested Loops

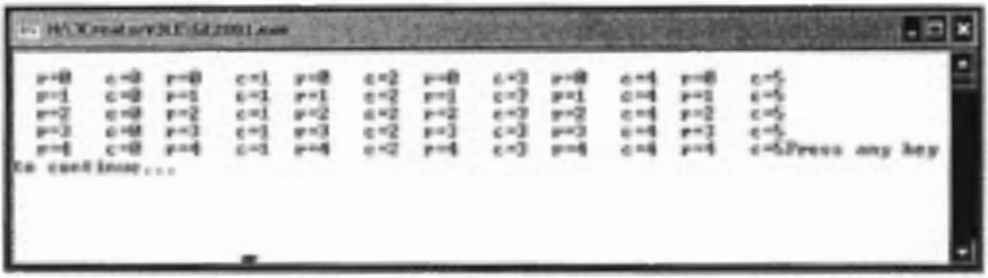
وفيه يتم استعمال تكرار for داخل تكرار for اخر كما يلى:

```
for(r=0;r<5;r++)
for(c=0;c<6;c++)
System.out.println("r="+r+"c="+c);
```

ولتوضيح ذلك قم بتعديل البرنامج السابق ليصبح كما فى الشكل

```
1  public class For4 {
2
3      public static void main(String[] args) {
4          // Create application frame
5  int r,c;
6      for(r=0;r<6;r++)
7      {
8        System.out.println("");
9
10
11 for(c=0;c<6;c++)
12 System.out.print(" r="+r+"  c="+c);
13 }
14    }
15 }
16
```

قم بترجمة وتنفيذ البرنامج تحصل على نتيجة التنفيذ كما فى الشكل.



مثال جدول الضرب

من الأمثلة الواضحة جدا على استعمال التكرار المتداخل Nested Loops هو طباعة جدول الضرب المشهور ، ولتوضيح ذلك قم بتعديل البرنامج السابق ليصبح كما فى الشكل.

```
1  public class For4 {
2
3      public static void main(String[] args) {
4          // Create application frame
5  int r,c;
6      for(r=1;r<13;r++)
7      {
8        System.out.println("");
9
10
11 for(c=r;c<13;c++)
12 System.out.print(r+"*"+c+"="+(r*c)+" ");
13
14 }
15    }
16 }
17
```

التكرار باستعمال While – Loop

الجملة while

تستعمل هذه الجملة لتكرار تنفيذ مجموعة من السطور عدد من المرات غير محدد مسبقا ولكن يتوقف على شرط ولذلك يفضل استعمالها فى حالة عدم معرفة عدد مرات التكرار وتاخذ الجملة التركيب التالى:

```
while
(expression)
statement
```

وفى هذه الصورة يتم اختبار الشرط expression إذا كان صحيح يتم تنفيذ الجمل Statements ويستمر التنفيذ حتى يصبح الشرط غير صحيح.

ولتوضيح ذلك تابع المثال التالى:

```
int n = 1;

      while (n < 6)
      {
         System.out.Println (n);
         n++;
}
```

فى هذه السطور

- فى السطر رقم 1 تم إنشاء فصيلة جديدة بالاسم usingWhile ثم الدالة الرئيسية main()
- فى السطر رقم 3 تم الإعلان عن مصفوفة قيم صحيحة int.
- فى السطر رقم 5 يتم استعمال التكرار while باختبار قيمة المتغير I طالما أقل من عدد عناصر المصفوفة نفذ ما يلى.
- فى السطر رقم 7 يتم طباعة عصر المصفوفة رقم I ثم يتم زيادة قيمة المتغير I حتى تنتهى عناصر المصفوفة.

ولتوضيح ذلك قم بتعديل البرنامج السابق ليصبح كما فى الشكل.

```
public class For4 {

   public static void main(String[] args) {
      // Create application frame
int n = 1;

      while (n <=10)
      {
        System.out.println ("     n=" +n);
        n++;
}
}
}
```

وبمراجعة أجزاء جملة الـ While تلاحظ أنها تحتوى على نفس أجزاء التكرار for

استخدام do --- While

يستخدم هذا التكرار مثل While ولكن يختلف عنه في انه يبدأ أولاً بتنفيذ الجمل ثم يختبر الشرط في While في نهاية التركيب فإذا كان صحيح يعاد التكرار مرة أخرى وإلا يتوقف وبالتالي يتم تنفيذ الجمل مرة واحدة على الأقل حتى لو كان الشرط خطأ، وتأخذ الصورة العامة التالية.

```
Do {
Statements
} While (Condation);
```

في هذه الصورة يتم تنفيذ الجمل statements أولاً ثم اختبار الشرط الموجود مع While فإذا كان صحيح يعود التكرار إلى do ويتم تنفيذ الجمل مرة أخرى وهكذا.

مثال:

في السطور التالية نعرض مثال بسيط لاستعمال do---While لطباعة مجموعة عبارات كما في الشكل.

```
public class For4 {

   public static void main(String[] args) {
      // Create application frame.


   int x = 1;
   do {
   System.out.println ("this is no "+x);
   x++;
   } while (x<=20);


}
}
```

فى هذه السطور

- يتم تعريف فصيلة جديدة بالاسم for4.
- ثم يبدأ التكرار فى السطر رقم 4 بالأمر do فيتم طباعة الرسالة this is no بالإضافة لقيمة المتغير x وهى أول مرة 1.
- ثم يتم زيادة قيمة x وفى السطر 7 يتم استعمال باقى تركيب التكرار وهو الأمر while الذى يختبر قيمة المتغير x فإذا كانت أقل من 20 يعود التكرار مرة أخرى إلى أول التكرار بالسطر 4

الخروج من جمل التكرار Breaking Out Of loops

يمكن استعمال الأمر Break للخروج من التكرار قبل إنهاء الشرط الخاص به ويظهر كما فى السطور التالية:-

```
int i =0;
While (i <50)
      {
if (i >20) break;
System.out.Println (i);
  }
```

فى هذا المثال

- يتم التكرار من 0 حتى 50 وليكن شرط لها يقطع هذا التكرار عند أول قيمة أكبر من 20 وهى 21.

فى Oracle PL/SQL

ويوجد أكثر من أمر تكرار تنفيذ العمليات وهي :

التكرار LOOP---if condition –Exit

التكرار LOOP----Exit when

التكرار while----loop

التكرار For ----loop

التكرار WHILE...LOOP

فى هذه الجملة يتم إضافة شرط التكرار مع While لتكوين تكرار while بدلا من تحديد شرط باستعمال جملة IF أو EXIT WHEN وبالتالى يختلف تركيب التكرار وذلك كما بالشكل:

```
WHILE condition LOOP  <-------- Condition is evaluated at the beginning of each iteration.
  statement1;
  statement2;
  . . .
END LOOP;
```

فى هذه الصيغة يبدأ التكرار بشرط التكرار condition مع أمر التكرار while ثم تبدأ جمل التكرار ككل جمل التكرار بالامر .LOOP ثم الجمل المطلوب تكرارها statement1,... ثم ينتهى التكرار كما هو معتاد بالامر END LOOP وبهذا الاسلوب لا تحتاج شرط للخروج لأن الشرط كتب فى أول التكرار مع الامر while ولتوضيح ذلك تابع السطور التالية:

```
DECLARE
MY_VAR NUMBER:=0;
BEGIN
 While MY_VAR <11
LOOP
DBMS_OUTPUT.PUT_LINE('MY_VAR='|| MY_VAR);
 MY_VAR := MY_VAR+1;
 END LOOP;
END;
```

فى هذا السطور:

- تم إضافة شرط التكرار وهو MY_VAR<11 مع بداية التكرار مع While ثم كتابة loop التي تبدأ طالما أن الشرط صحيح ويتم تنفيذ السطر رقم 2 ثم إعادة التكرار وهكذا حتى يصبح الشرط غير صحيح أي تزيد قيمة المتغير MY_VAR عن 11.
- ويختلف هذا التكرار عن التكرار السابق في أن التكرار WHILE لا يبدأ بالتكرار ولا مرة إلا إذا كان الشرط صحيح ، بينما التكرار السابق يقوم بتنفيذ الأوامر مرة واحدة على الأقل ثم يختبر الشرط ليعيد التكرار أو يتوقف.

جملة التكرار FOR

هو من أشهر الطرق في جميع لغات البرمجة حيث يستعمل التكرار تنفيذ عملية عدد من المرات معروفة ، ويأخذ الشكل التالى:

```
FOR counter IN [REVERSE]
    lower_bound..upper_bound LOOP
  statement1;
  statement2;
  . . .
END LOOP;
```

فى هذه الصورة:

يبدأ التكرار بالأمر FOR ثم المتغير المستعمل فى التكرار هو أي متغير ثم IN لتحدد أن قيمة التكرار لابد أن تقع فيما يلي:

- وهو 1—1 أي من القيمة 1 إلى القيمة 1 بمقدار زيادة 1 كل مرة ، ثم الأمر LOOP لتكرار تنفيذ الجمل التالية.
- ثم تكتب الجمل المطلوب تكرارها وفى نهاية الجمل يكتب END LOOP لإنهاء جمل التكرار .
- ويمكن تغير قيم التكرار بالعكس أي بدلاً من تكرارها من القيمة 1 إلى 1 ويتم تكرارها من 1 إلى 1وذلك بإضافة كلمة REVERSE كما بالشكل:

FOR MY_VAR IN REVERSE 1—1 LOOP

```
|
END LOOP;
```

ولتوضيح استعمال FOR تابع المثال التالى:

ويلاحظ ذالك فى الشكل:

```
REM NumericLoop.sql
REM Chapter 1, Oracle9i PL/SQL Programming by Scott Urman
REM This block illustrates a numeric FOR loop.


BEGIN
  FOR v_LoopCounter IN 1..50 LOOP
```

التكرار LOOP---if condition –Exit

وفيه يتم تكرار الأوامر التي بينهما بدون توقف ولإيقاف هذا التكرار لابد من إضافة شرط داخل التكرار كما يلي:

```
LOOP                                    -- delimiter
  statement1;                           -- statements
  . . .
 IF  condition THEN
  EXIT;                                 -- EXIT statement
END IF ;
END LOOP;                               -- delimiter
```

في هذه الصيغة

تبدأ هذه الصيغة بالأمر LOOP الذي يعني كرر الجمل التالية وتنتهي الصيغة بالأمر END LOOP الذي ينهي الجمل المطلوب تكرارها وبينهما الجمل.

كما يجب أن يحتوي هذا النوع من جمل التكرار على أمر شروط وخروج من التكرار وفي هذه الصيغة نستعمل الأمر IF ثم الشرط ثم الأمر EXIT إذا تحقق الشرط ولتوضيح ذلك تابع المثال التالي:

```
DECLARE
MY_VAR Number (10) :=1 ;
BEGIN
LOOP
DBMS_OUTPUT.PUT_LINE('MY_VAR ='||MY_VAR);
MY_VAR := MY_VAR +1;
IF MY_VAR = 9     THEN
EXIT;
END IF;
END LOOP;
END;
```

في هذه السطور تم استعمال التكرار LOOP—EXIT كما يلي:

- في السطر رقم 2 إعلان عن متغير MY-VAR من نوع رقمي.
- في السطر رقم 4 بداية التكرار باستعمال الأمر LOOP.
- في السطر رقم 5 يتم إضافة 1 لقيمة المتغير MY-VAR إذا كانت 9.
- ينفذ السطر 7 وهو الأمر EXIT ويتم الخروج من التكرار.

- وإلا يستمر تنفيذ الأوامر وعدم الخروج إلى السطر رقم 9 END LOOP الذى يعيد التنفيذ مرة أخرى إلى بداية التكرار وهو السطر 4 وهكذا حتى تصل قيمة المتغير إلى 9.

جملة التكرار LOOP----EXIT WHEN

يتم استعمال الامر EXIT WHEN بدلا من استعمال IF وهى أسهل وتأخذ الصورة العامة التالية:

```
LOOP                                    -- delimiter
  statement1;                           -- statements
  . . .
  EXIT [WHEN condition];                -- EXIT statement
END LOOP;                               -- delimiter
```

فى هذه الصيغة تم استبدال جملة الشرط IF وأمر الخروج EXIT بالأمر EXIT WHEN ولتوضيح ذلك تابع السطور التالية:

```
REM SimpleLoop.sql
REM This block contains a simple loop.


DECLARE
  v_LoopCounter BINARY_INTEGER := 1;
BEGIN
  LOOP
    INSERT INTO temp_table (char_col)
      VALUES (v_LoopCounter);
    v_LoopCounter := v_LoopCounter + 1;
    EXIT WHEN v_LoopCounter > 50;
  END LOOP;
END;
/
```

فى هذا السطور:

فى هذا السطر تم وضع الشرط مع الأمر WHEN بدلاً من الأمر IF وهذا الأسلوب أوقع وأوضح من استعمال IF ويؤدى إلى نفس النتيجة حيث يؤدى إلى إضافة الأرقام من 1 إلى 50 فى الجدول Temp_table

```
    INSERT INTO temp_table (char_col)
      VALUES (v_LoopCounter);
  END LOOP;
END;
```

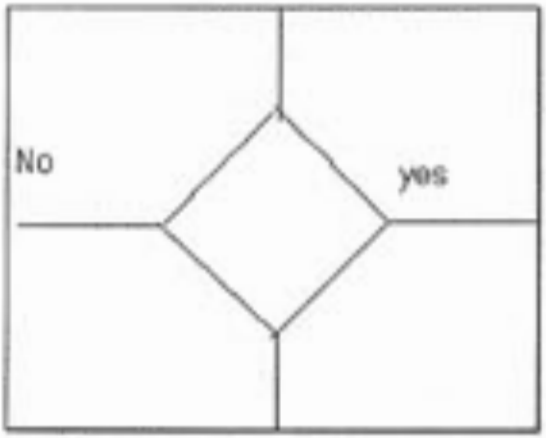
جمل التحكم في مسار البرنامج

في هذه الفقرة نتناول الجمل التالية:

- جملة ..... If
- جملة ....If.....Else
- جملة ....If ..... else if ....
- جملة select case....end select
- جملة GOTO

هذه الجمل هي التي تحدد مسار البرنامج هل يسير في الخط الطبيعي أم يتفرع إلى دالة فرعية وهي التي تحقق رغبات المستخدم حيث لا يمكن أن يكون هناك برنامج بل جمل تحكم .

و يتضح ذلك من خريطة التدفق الموضحة في الشكل (2-4):



الشكل (2-4)

في هذا الشكل نلاحظ وجود جملة شرط تحدد مسار البرنامج فإذا كان yes يتجه التنفيذ إلى اتجاه معين و إذا كان NO أتجه اتجاه آخر .

و فيما يلي نشرح جمل التحكم :

في لغة VB.NET

جملة IF

تستخدم جملة if كما هي بدون else إذا كان هناك اختبار تريد اختباره إذا تحقق ننفذ جمل معينة و إلا يسير البرنامج طبيعي و لا ينفذ هذه الجمل و يأخذ ذلك الصور العامة التالية:

```
If condition Then [statements] [Else elsestatements]
Or, you can use the block form syntax:
If condition Then
[statements]
[ElseIf condition-n Then
[elseifstatements] ...
[Else
[elsestatements]]
End If
```

وجملة if تأخذ شكلين:

الصورة الأولى:

If Condation then Statement

وهي صورة بسيطة لا تحتاج الى End If وتستعمل عندما تكون نتيجة الشرط عبارة عن جملة واحدة

الصورة الثانية

```
if ... condation ... then
Statements 1
Statements 2
End if
```

في هذه الصورة إذا كان الشرط condation صحيح ننفذ الجمل statements 1 , statements 2 و ينهي ذلك كلمة end if .

تركيب IF ...ELSE IF

يستعمل هذا التركيب عندما يكون هناك أكثر من احتمالين (شرطين) نريد اختبارهما و يأخذ الصورة :

```
IF CONDATION
ELSE IF CONDATION THEN
```

```
ELSE IF CONDATION THEN
ELSE
END IF
```

في هذه الصورة

اذا تحقق الشرط الاول يختبر الشرط الثانى وهكذا.

لاحظ المثال الموجود بالمسار EX12 بأسم NESTED_IF.

التركيب SELECT ... CASE

يستعمل هذا التركيب محل التركيب السابق و هو اختيار أكثر من حالتين و هو أسهل في التركيب و الأستعمال عن التركيب السابق .

و يأخذ الصورة العامة التالية:

```
Select Case testexpression
[Case expressionlist-n
[statements-n]] ...
[Case Else
[elsestatements]]
End Select
```

والصورة التالية هى نفس الصورة بشكل أبسط.

```
SELECT CASE VAR
CASE 1
Statement 1
CASE 2
Statement 2
CASE3
Statement 3
Else case4
Statement 4
END SELECT
```

في هذه الصورة

يتم اختبار المتغير var ثم تحديد قيمته حسب الحالات الموجودة فى cases وبالتالى إذا كانت قيمة المتغير VAR 1 يتم تنفيذ الجملة State ment 1 وهكذا مع Case 2 وباقى الحالات وإذا لم تتحقق أحد الحالات تنفذ الجملة Else Case.

كما يأخذ هذا التركيب أكثر من صورة سوف نوضحها مع الأمثلة :

الصورة الاولى

```
SELECT  CASE VAR
CASE 1

CASE 2

CASE3

Else case

 END SELECT
```

وهى الصورة التقليدية والتى شرحناها فى الصورة السابقة

الصورة الثانية:

```
SELECT CASE VAR
CASE Is > 5
Statement 1
CASE Is > 10
Statement 2
CASE3 Is > 15
Statement 3
Else case
Statement 4
END SELECT
```

وفى هذه الصورة يتم اختبار قيمة المتغير هل هى أكبر أم أصغر من قيمة معينة ففى هذه الحالة إذا كانت قيمة المتغير VAR أكبر من 5 تنفذ الحالة الأولى وتنفذ الجملة Statement 1 وهكذا باقى الحالات:

الصورة الثالثة:

```
SELECT CASE VAR
CASE VAR =1 TO 5
Statement 1
CASE VAR =5 TO 10
Statement 2
CASE3 VAR =10 TO 15
Statement 3
Else case
Statement 4
 END SELECT
```

هذه الصورة جيدة جداً حيث تقوم باختيار قيمة المتغير VAR والحالات هى مدى فمثلاً.

الحالة الاولى: تقول إذا كانت قيمة المتغير تقع بين القيمة 1 أو القيمة 5 تنفذ الجملة Statement 1 وفى هذه الصورة يتم اختبار قيمة المتغير هل هى تقع بين قيمتين أم لا وهكذا باقى الحالات.

مثال :

- قم بتصميم مثال يستقبل درجة الطالب ثم يختبر هذه الدرجة
- اذا كانت أقل من 50 تخرج رسالة اعتذار أن الطالب لم ينجح
- اذا كانت تبدأ من 50 وأقل من 64.9 تخرج رسالة تفيد أن الطالب نجح بتقدير مقبول
- اذا كانت تبدأ من 65 وأقل من 74.9 تخرج رسالة تفيد أن الطالب نجح بتقدير جيد وهكذا باقى التقديرات

خطوات التصميم

1- إبدأ برنامج جديد بالخطوات المعتادة.

2- وقع أداة كتابة TextBox وأداة زر Button1.

3- اكتب فى دالة زر الامر السطور التالية:

```
Dim t
t = Val(TEXTBOX1.TEXT)
Select Case t
   Case t = 0 To 49.9
      MsgBox("SORRY....")
   Case t = 50 To 64.9
      MsgBox("OK YOU PASSED...")
   Case t = 65 To 74.9
      MsgBox("OK YOU ARE GOOD...")
   Case t = 75 To 84.9
      MsgBox("OK YOU ARE VERY GOOD...")
   Case t = 85 To 100
      MsgBox ("OK YOU ARE EXCELLENT...")
   Case Else
      MsgBox("OUT OF RANG..")
End Select
```

جملة GOTO

تستخدم GOTO لتغير مسار البرنامج وتوجه تنفيذ البرنامج إلى خطوة معينة وإن كانت غير مستحبة فى عالم البرمجة وتأخذ الشكل التالى:

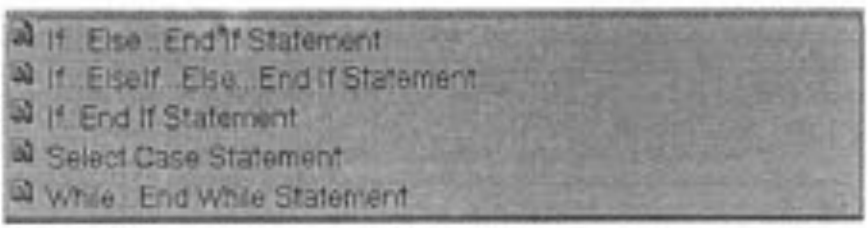
```
GoTo line
```

وتوضع فى الأمثلة كما فى الشكل التالى:

```
IF A=10 THEN GOTO R
--
--
--
--
R:
--
--
--
```

الاستفادة من الـ Snippet

تعرض قائمة Snippet جميع الصور المتاحة لجمل الشرط Condations كما فى الشكل



كما يلى:

```
    If True Then

Else

End If
```

هذه الصيغة التقليدية لجملة الشرط If Else

```
    If True Then
ElseIf False Then
    Else
    End If
```

تنفذ الجملتين STATEMENT1, STATEMENT2 و إلا (إذا كان الشرط غير صحيح) تنفذ الجمل التي بعد ELSE و هي في الصورة STATEMENT3, STATEMENT4.

و لتوضيح ذلك تابع المثال التالي:

```
1. If (x>y)
2. {
3. System.out.Println ("x is greater than y – ");
4. System.out. Println ("so you can go -- ");
5. }
6. else
7. {
8. System.Out.println ("x is less than y – ");
9. System.Out.println ("so you can not go -- ");
10. }
```

في هذه السطور:

يتم اختبار الشرط (x>y) فإذا كان صحيح يتم تنفيذ الجمل التى في السطور 3,4 وإلا (else) تنفذ الجمل في السطور 8,9.

مؤثر الشرط Conditional Operator

بالإضافة لجملة if يوجد مؤثر الشرط الذى يحقق عمل لأمر if --- else بشكل بسيط كما يلى:

```
Condition ? trueresult: false result;
```

في هذا السطر:

Condition: هو الشرط المطلوب اختباره.

True result: ينفذ إذا كان الشرط صحيح.

False result: ينفذ إذا كان الشرط خطأ.

وبالتالى يقول هذا المؤثر إذا كان الشرط Condition صحيح قم بتنفيذ الجملة المكتوبة في مكان True result وإلا قم بتنفيذ الجملة الأخرى false result ويظهر ذلك من المثال التالى:

```
k= (x > y) ? 10:20:
```

فى هذا المثال تأخذ k القيمة 10 إذا كانت x أكبر من y

وتأخذ k القيمة 20 إذا كانت x أصغر من y

وهذا المؤثر يكافئ جملة If..Else كما يلى:

```
if (x > y)
k=10;
else
k=20;
```

التركيب الشرطى Switch

هذا التركيب يستعمل بدلاً من استعمال اكثر من جملة else --- if وذلك لتحديد قيمة من عدة قيم فمثلاً إذا كانت لديك قيمة وتريدا اختبارها واحتمال ان تأخذ اكثر من نتيجة نستعمل لذلك التركيب Switch. الذى يأخذ الشكل التالى:

```
switch (expression)
{
   case constant-expression:
      statement
      jump-statement
   [default:
      statement
      jump-statement]
}
```

فى هذه الصورة

تبدأ بالكلمة switch وبداخلها المتغير expression الذى يتم اختباره ثم تبدأ الحالات المتوقعة لقيمة هذا المتغير VAR.

- الحالة الأولى Case Value One تقول فى حالة القيمة الأولى تنفذ النتيجة الأول do result one وهكذا.
- وفى النهاية default التى تنفذ فى حالة عدم تحقق أى حالة مع ملاحظة أن الكلمات المحجوزة فى هذه الصورة هى:

switch, case, break, default.

الكلمة switch : لاختبار قيمة المتغير.

الكلمة case : لتحديد الحالات المختلفة للمتغير.

فى هذه الصيغة يتم إضافة ElseIf لإضافة شرط آخر.

```
If True Then

End If
```

هذه الصيغة أبسط صور الشرط والتى تحتوى على شرط واحد.

```
Select Case VariableName
   Case 1
   Case 2
   Case Else
End Select
```

فى هذه الصيغة يتم عرض التركيبSelect Case الذى يسمح بتحديد اختيار من مجموعة اختيارات.

فى لغات C-Like

فى هذه الفقرة نتناول الجمل التالية:

جملة ..... if

جملة ...if.....Else

جملة ....if .....End if .....else if

جملة switch case....

جملة goto

ويمكن تنفيذها فى Java,

جملة IF

تستخدم جملة if كما هي بدون else إذا كان هناك اختبار نريد اختباره إذا تحقق ننفذ جمل معينة و إلا يسير البرنامج طبيعي و لا ينفذ هذه الجمل و يأخذ ذلك الصور العامة التالية:

```
if (expression)
   statement1
[else
   statement2]
```

وجملة if تأخذ شكلين:

الصورة الأولى: If Condation then Statement

وهى صورة بسيطة لا تحتاج الى End If وتستعمل عندما تكون نتيجة الشرط عبارة عن جملة واحدة.

الصورة الثانية:

```
if (condation )
{
Statements 1
Statements 2
}
```

مثال:

```
if (x>y)
System.out. Println ("x is greater than y");
```

فى هذا السطر يتم استعمال if لاختبار الشرط (x>y) فإذا كان صحيح اى ان قيمة x اكبر من قيمة y يتم تنفيذ الجملة التالية للشرط وهى طباعة العبارة الموجودة.

فى هذا المثال يتم تنفيذ جملة واحدة كنتيجة للشرط ولكن إذا أردت تنفيذ أكثر من جملة تحصرهم بين الأقواس ( ) لتصبح بلوك فينفذ كنتيجة للشرط .بالشكل التالى:

```
if (x>y)
{
System.out. Println ("x is greater than y");
System.out. Println ("x is greater than y");

}
```

جملة if ----else

يستعمل هذا التركيب لاختبار شرط معين فإذا كان صحيح ينفذ ما بعد الـif وإذا كان غير صحيح ينفذ ما بعد else ويظهر ذلك من الشكل التالي :

```
if (expression)
   statement1
[else
   statement2]
```

في هذه الصورة يتم اختيار الشرط CONDATION بجملة IF فإذا كان الشرط صحيح

الكلمة break : للخروج من التركيب الشرطى switch عند تحقق حالة.

الكلمة default : لتنفيذ مجموعة من الجمل فى حالة عدم تحقق أى حالة.

مثال:

```
public class Switchcase {

   public static void main(String[] args) {
   int v=1;
      switch (v)
{
   case 0 :
    System.out.println ( "Zero");
   break;
   case 1 :
    System.out.println ( "One");
    break;
   case 2 :
    System.out.println ( "Two");
    break;
   case 3 :
    System.out.println ( "Three");
    break;
   case 4 :
    System.out.println ( "Four");break;
    case 5 :
    System.out.println ( "Five");break;
   case 6 : System.out.println ( "Six");break;
   case 7 : System.out.println ( "Seven");break;
   case 8 : System.out.println ( "Eight");break;
   case 9 : System.out.println ( "Nine");break;
   default :System.out.println ( " ");break;
   }
   }
}
```

فى Oracle PL/SQl

التحكم في مسار البرنامج Flow of process

غالباً ما تحتاج التحكم فى سير البرنامج بتوجيهه إلى عملية معينة بشرط معين أو تكرار تنفيذ عمليات معينة أكثر من مرة.

- ويتم الاختبار وتوجيه التنفيذ باستعمال الأمر if...then...else end if.
- ويتم تكرار تنفيذ العمليات باستعمال جمل تكرار وفيها for,loop,while وغيرها.
- ونبدأها بجمل الشرط.

1. الأمر IF

ونأخذ جملة IF الشكل العام

```
IF condition THEN
  statements;
[ELSIF condition THEN
  statements;]
[ELSE
  statements;]
END IF;
```

في هذه الصورة:

يتم استعمال الأمر IF لاختبار الشرط Condition فإذا كان صحيح يتجه التنفيذ إلى ما بعد then وإلا يتم توجيه التنفيذ إلى ما بعد ELSE.

- وفي النهاية يتم إنهاء بلوك IF بالكلمة END IF.
- ويمكن تكرار الشرط بكتابة ELSE IF بدلاً من ELSE.

```
Set serveroutput on;
DECLARE
  v_TotalStudents NUMBER;
BEGIN
  -- Retreive the total number of students from the database.
  SELECT COUNT(*)
    INTO v_TotalStudents
    FROM students;
    Dbms_output.put_line('number of students='|| v_TotalStudents);
  -- Based on this value, insert the appropriate row into temp_table.
  IF v_TotalStudents = 0 THEN
    INSERT INTO temp_table (char_col)
      VALUES ('There are no students registered');
  ELSIF v_TotalStudents < 5 THEN
    INSERT INTO temp_table (char_col)
      VALUES ('There are only a few students registered');
  ELSIF v_TotalStudents < 10 THEN
    INSERT INTO temp_table (char_col)
      VALUES ('There are a little more students registered');
  ELSE
    INSERT INTO temp_table (char_col)
      VALUES ('There are many students registered');
  END IF;
END;
/
```

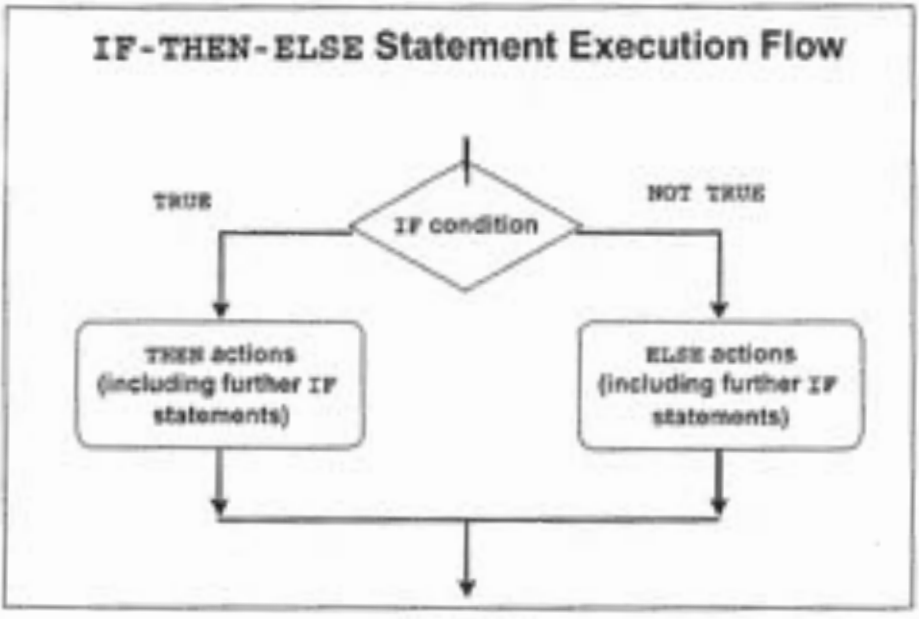
فى هذه السطور:

يتم استعمال الأمر IF لاختبار الشرط Condition فإذا كان صحيح يتجه التنفيذ إلي ما بعد then وإلا يتم توجيه التنفيذ إلي ما بعد ELSE ويبدأ ذلك باختبار عدد الطلاب فى الجدول students باستعمال الأمر SELECT مع الدالة COUNT ووضع النتيجة فى المتغير v_TotalStudents ثم استعمال الأمر IF لاختباره فاذا كان العدد أقل من 0 يتم استعمال الامر INSERT لاضافة العبارة 'There are no students registered' فى الجدول TABLE_TEMP ثم يتم استعمال الأمر ELSIF شرط آخر وهو اختبار قيمة المتغير v_TotalStudents إذا كانت أقل من 5 وهكذا .

وفى النهاية يتم إنهاء بلوك IF بالكلمة END IF ويمكن تكرار الشرط بكتابة ELSE IF بدلاً من ELSE.

والشكل التالى يعرض مخطط يوضح منطق تنفيذ جملة IF فى حالة ما إذا كان الشرط صحيح أو غير صحيح ويوضح ما تنفذ نتيجة IF ومتى تنفذ نتيجة ELSE.

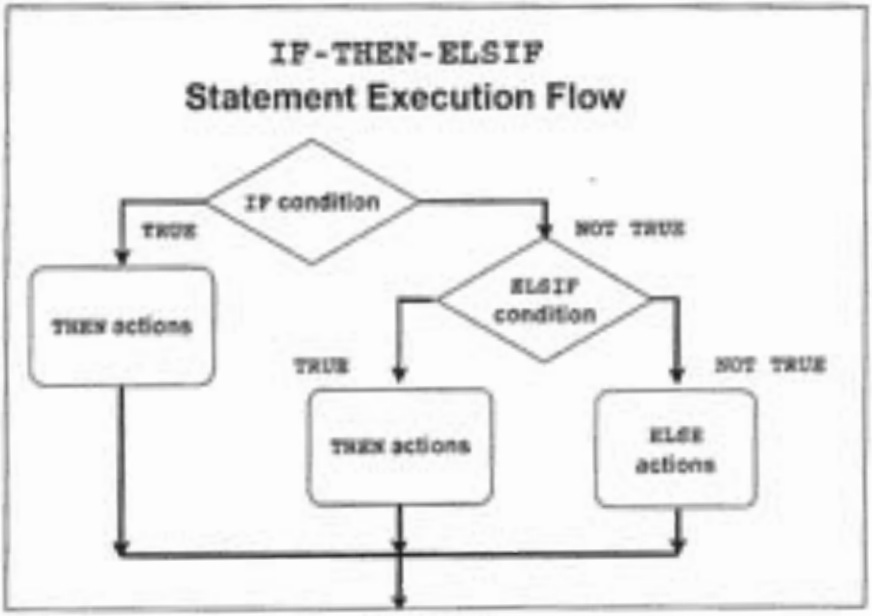
كما فى الشكل (3-4)



الشكل(3_4)

الأمر IF ELSIF

يستعمل هذا الأمر في حالة تعدد الشروط بمعنى أنك لديك أكثر من اختبار ويتضح ذلك من الشكل (4-4).



شكل (4_4)

مثال:

```
SET SERVEROUTPUT ON ESCAPE OFF

DECLARE
  v_price books.price%TYPE;
  v_isbn books.isbn%TYPE := '12345678';
BEGIN
  SELECT price
  INTO v_price
  FROM books
  WHERE isbn = v_isbn;

  DBMS_OUTPUT.PUT_LINE('Starting price: '||v_price);

  IF v_price < 40
  THEN
    DBMS_OUTPUT.PUT_LINE('This book is already discounted');
  ELSIF v_price BETWEEN 40 AND 50
```

```
  THEN
    v_price := v_price - (v_price * .10);

    UPDATE books
    SET price = v_price
    WHERE isbn = v_isbn;
  ELSIF v_price > 50
  THEN
    v_price := v_price - (v_price * .10);
    UPDATE books
    SET price = v_price
    WHERE isbn = v_isbn;
  END IF;

  DBMS_OUTPUT.PUT_LINE('Ending price: '||v_price);

  ROLLBACK;
EXCEPTION
  WHEN OTHERS
  THEN
    DBMS_OUTPUT.PUT_LINE(SQLERRM);
    ROLLBACK;
END;
/
```

الشكل(4_4)

في هذه السطور:

- تم وضع قيمة الحقل price أي سعر الكتاب لكتاب محدد في المتغير v_price ثم تم اختباره.
- باستعمال الأمر if ثم أجريت عليه عدة اختبارات باستعمال ELSIF وفي كل مرة يتم خصم قيمة من هذا السعر حسب السعر.

وبالتالي يكون الغرض من هذه السطور هي تحديد سعر الكتاب بعد الخصم.

استعمال Case

بالإضافة لأوامر IF المختلفة لاختبار المتغير وتحديد القيمة وعلى أساسها التصرف يوجد الأمر CASE الذي يسمح لك باختيار أكثر من قيمة للمتغير وهو أبسط من استعمال IF ELSEIF.

ويأخذ هذا الأمر الصيغة العامة التالية:

```
CASE selector
   WHEN expression1 THEN result1
   WHEN expression2 THEN result2
   ...
   WHEN expressionN THEN resultN
  [ELSE resultN+1;]
END;
```

في هذه الصيغة:

- يتم اختبار قيمة المتغير selector باستعمال الأمر CASE ثم تحديد حالة من حالاته.
- ويتم ذلك باستعمال الامر WHEN ثم اختبار الشرط ثم الأمر THEN ثم نتيجة تحقق الشرط result1.
- وهكذا حتى تنتهي الحالات وفي النهاية ELSE.
- التي تنفذ حالة عدم تحقق جميع الحالات السابقة.
- ثم END لإنهاء بلوك .

ولتوضيح ذلك تابع المثال التالي:

```
SET SERVEROUTPUT ON;
DEFINE P_GRADE=A;
DECLARE
V_GRADE CHAR(10) := UPPER('&P_GRADE');
V_APPRAISAL VARCHAR2(20);
BEGIN
V_APPRAISAL := CASE V_GRADE
WHEN 'A' THEN 'EXCELLENT'
WHEN 'B' THEN 'VERY GOOD'
WHEN 'C' THEN ' GOOD'
ELSE 'NO SUCH GRADE'
END;
DBMS_OUTPUT.PUT_LINE('GRADE:'|| V_GRADE ||' APPRAISAL '|| V_APPRAISAL);
END;
```

فى هذه السطور:

تم تعريف متغير بالاسم p_grade ووضع الحرف A فيه وذلك فى بيئة SQL PLUS ثم ابتداء بلوك PL/SQL ثم وضع قيمة هذا المتغير فى متغير البلوك v_grade بعد تحويله الى حرف كبير باستعمال الدالة UPPER() .

- ثم تم تعريف متغير من نوع varchar2.
- ثم تم استعمال الامر CASE لاختبار قيمة المتغير v_grade.
- ثم تم استعمال أكثر من WHEN لتحديد أحد الحالات.

وفى النهاية يتم طباعة النتيجة وهى EXCELENT حيث أن قيمة المتغير هى A.

الأمر GoTo

هذا الأمر يؤدى إلى انتقال التنفيذ إلى سطر محدد ويفيد عند استعمال الأمر If فإذا تحقق شرط معين يتم الانتقال بالتنفيذ إلى سطر معين ولا يتم تنفيذ سطور أخرى .

ولتوضيح ذلك تابع السطور التالية:

```
SET SERVEROUTPUT ON
BEGIN
   GOTO l_Last_Line;
 DBMS_OUTPUT.PUT_LINE('GOTO didn"t work!');
   RETURN;
   <<l_Last_Line>>
 DBMS_OUTPUT.PUT_LINE('Last Line');
 END;
/
```

فى هذه السطور:

يتم استدعاء الأمر GOTO الى يؤدى إلى الانتقال إلى آخر سطر وبالتالى يطبع العبارة Last Line.