الفصل الخامس

الدوال
Functions, Procedures

Programming Concepts 5

فى هذا الفصل

في هذا الفصل نشرح موضوع من أساسيات البرمجة التركيبية لأي لغة وهو إنشاء دوال فرعية FUNCTIONS أو Procedures و ذلك من خلال النقاط التالية :

- معني الدالة و البرنامج الفرعي (Function & Procedure)
- الفرق بين الدالة Functionو البرنامج الفرعي Procedure (الأجراء)
- كيفية إنشاء وأستعمال برنامج فرعي Procedure (الأجراء)
- كيفية انشاء و استعمال دالة Function
- ملفات الدوال و البرامج الفرعية Modules
- برنامج من النوع Console Application

معني الدالة و البرنامج الفرعي FUNCTION & SUB

إذا سبق لك دراسة لغة من لغات البرمجة الأخري مثل لغة C و غيرها فأنت تعرف أن أي برنامج يتكون من برنامج رئيسي و برامج فرعية و دوال يتم استدعائها من البرنامج الرئيسي أي أن الدالة و البرنامج الفرعي هما مجموعة من الأوامر التي تكتب داخل اطار اسم برنامج فرعي يسمي دالة FUNCTION أو SUB يتم إنشائه مرة واحدة و يمكن استدعائة أكثر من مرة مما يوفر اعادة كتابة هذه الأوامر أكثر من مرة.

الفرق بين الدالة FUNCTION و البرنامج الفرعي Procedure

الفرق الأساسي بين ال FUNCTION و Procedure (الإجراء) هو أن الـ FUNCTION لابد أن تعيد قيمة عند استدعائها مثل دالة حساب AVGأو SIN أو أي دالة من دوال اللغة.

بينما يقوم البرنامج الفرعي Procedure (الإجراء) بتنفيذ مجموعة من السطور دون أن تعيد قيمة بينما ينفذ الجمل لتحقيق حدث و سوف يتضح ذلك عند التعامل معهما .

ما هي الدوال Methods

الدالة Method هي مجموعة من سطور البرمجة التي تتحد معا لتحقيق وظيفة معينة ، ويمكنك أن تقول أنها برنامج فرعي يؤدي وظيفة محددة

وهي تتصف بما يلي:

1- برنامج فرعي يمكن استدعئه أكثر من مرة
2- هي بلوك من الكود لتحقيق وظيفة محددة
3- وهي غالبا ما تحتوى على مدخلات Input ومخرجات output (ولكن ليس بالضرورى)
4- أحيانا تسمى "subroutines" or "functions"

لماذا نحتاج إلى الدوال

1. Modularity جعل البرنامج مركب

بالطبع يصبح البرنامج كبير ويحتوى على عدد كبير من السطور ..، وبالتالى من

المفضل تقسيم هذا البرنامج إلى برامج صغيرة Methods واستدعائها ، وبالتالى يتم تقسيم وظائف البرنامج إلى دوال Methods

2. Code Re-use اعادة الاستخدام

من أشهر قواعد البرمجة والتى حققتها الدوال methods نظرية كتابة الكود مرة واحدة ثم اعادة استخدامه أكثر من مرة. وهى من القواعد المشهورة جدا والمهمة جدا ، وهو اتجاه تطوير البرمجة من أول إنشائها وحتى الآن وفى المستقبل ، ومن أول ما حقق هذا المفهوم وجود الدوال Methods

3. Abstraction إخفاء الأوامر

من فوائد الدوال Methods هو إنشاء دوال تحقق وظائف معينة وعند الاحتياج إلى وظائف هذه الدوال يتم استعمالها دون الرجوع إلى سطور هذه الدوال ، وسوف تتضح هذه النقطة أكثر مع درس Objects and Classes.

التعامل مع البرنامج الفرعي Procedure (الإجراء)

يتم التعامل مع البرنامج الفرعي Procedure أو الدالة خلال مرحلتين هما :

إنشاء البرنامج الفرعي (الإجراء)

استدعاء البرنامج الفرعي (الإجراء)

فى لغة VB.NET

(أ) إنشاء البرنامج الفرعي SUB

لتوضيح ذلك تابع الخطوات التالية :

من القائمة الرئيسية اختر PROJECT و منها اختر ADD MODULE كما بالشكل (1-5)

و بالتالي تكون خطوات التعامل مع البرنامج الفرعي هي :

1- إضافة ملف أوامر أول مرة (MODULE).

2- كتابة اسم و سطور البرنامج الفرعي S1.

3- استدعاء البرنامج الفرعي بكتابة اسمه من المكان المطلوب أستدعائه منه إرسال معاملات .

في المثال السابق تم أنشاء برنامج فرعي Procedure (الإجراء) بسيط لا يحقق شيء أكثر من إظهار رسالة ثابتة و لكن يمكن تغيير هذه الرسالة بإضافة معاملات للبرنامج الفرعي و لتحقيق ذلك تابع ما يلي :

1- أعد كتابة البرنامج الفرعي S1 ليصبح كما يلي :

```
Sub s1(ByVal st As String)
    MsgBox(st)
End Sub
```

- الجديد في هذه السطور هو إضافة المتغير ST من نوع STRING و هذا يعني أن S1 لها معامل من نوع STRING يتم ارساله إليها عند الاستدعاء، ولابد من إرسال المعامل وألا تحصل على رسالة خطأ.
- في السطر رقم 2 تم أستعمال الدالة MSGBOX لعرض رسالة و لكن هذه الرسالة هي محتوي المتغير ST الذي يأخذ قيمة عند استدعاء S1 من المعامل.

2- عدل دالة زر أمر استدعاء S1 ليصبح كما يلي :

```
S1 (" THIS MSG PASSED TO S1 AS A PARAMETER")
```

- في هذا السطر تم استدعاء S1 كما سبق و لكن الجديد هو ارسال معامل لها هو الرسالة المطلوب اظهار ما بين " " و ذلك لأن المعامل من النوع STRING و بالتالي عند الاستدعاء ترسل الرسالة إلي المعامل ST في البرنامج الفرعي (الإجراء) الذي يأخذ هذا المعامل (الرسالة) ويضعه أمام الدالة MSGBOX فتظهر الرسالة .

و بنفس الأسلوب يمكن زيادة أو تغيير المعاملات من حيث النوع و العدد و لتوضيح ذلك تابع المثال التالي :

```
1: sub s2 (no as integer , msg  as  string )
2: for I=0 to no
3: Msgbox msg
4: next I
5: end sub
```

في هذه السطور

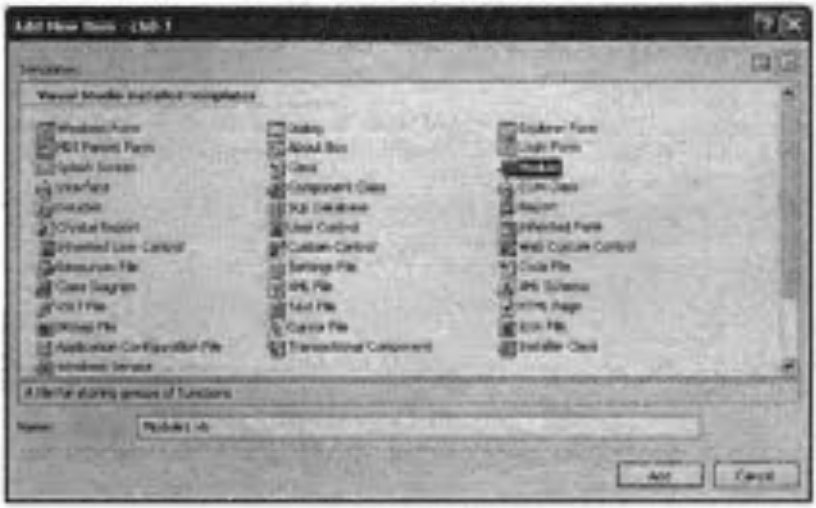
- في السطر رقم 1 يبدأ البرنامج الفرعي بأسم s2 مع تعريف معاملين هما no من نوع رقم integer و msg من نوع حرفي string.
- في السطر رقم 2 يبدأ التكرار باستعمال for و لكن يبدأ من 0 إلى قيمة المتغير no التي يرسلها برنامج الاستدعاء الفرعي .
- في السطر رقم 3 يتم عرض رسالة هي قيمة المتغير msg و هو المعامل التالي القادم من استدعاء البرنامج الفرعي (الإجراء).
- و باقي السطور كما سبق.
- وقع زر أمر على خلفية البرنامج (From 1) و أكتب في دالتة السطر التالي:
- S2 10 , " VISUAL BASIC.Net ..."
- في هذا السطر تم استدعاء البرنامج الفرعي S2 مع أرسال معاملين له هما القيمة 10 و الرسالة ... VISUAL BASIC.Net تذهب إلى المعامل الأول في البرنامج الفرعي وهو NO القيمة 10.
- و المعامل الثاني هو الرسالة و بالتالي يتم طباعة هذه الرسالة عدد NO من المرات كما في نص البرنامج الفرعي S2.

مما سبق نلاحظ إمكانية تصميم (إنشاء) برنامج فرعي من نوع SUB (الأجراء) و ذلك بالأشكال التالية :

- بدون معاملات.

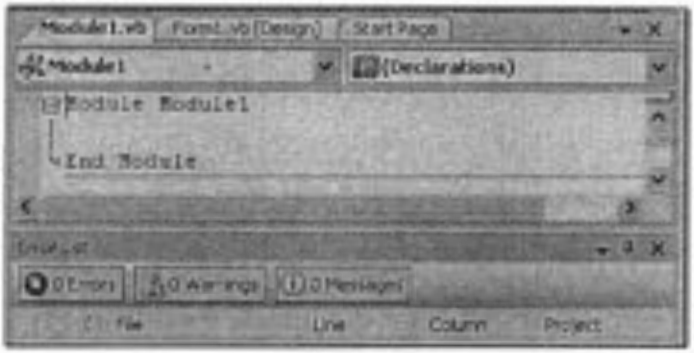
الشكل (5-1)

1- تحصل على شكل به مجموعة من الـ Modules وتحصل على صفحة بيضاء لكتابة سطور كما في الشكل (5-2) .



الشكل (5-2)

2- اضغط زر OPEN يتم إضافة ملف أوامر يظهر في مربع المشروع كما يتم فتحه. كما في الشكل (5-3) .



الشكل (3-5)

3- في مربع المشروع اضغط MODULE1 مرتين تحصل في الجانب الأيسر على خلفية بيضاء

4- في الصفحة البيضاء أكتب SUB S1 ثم أضغط مفتاح ENTER تلاحظ كتابة END SUB كما بالشكل التالي:

```
Module Module1
   Sub s1()

   End Sub
End Module
```

بهذا تم إنشاء برنامج فرعي Procedure (الإجراء) بالاسم S1 و لكنة لا يؤدي وظيفة لأنه لا يحتوي على اي أوامر و لتحقيق وظيفة ما أكتب بين SubS1() و END SUB السطر التالي :

```
MSGBOX " THIS LINE CALLED FROM SUB S1 "
```

و لاختبار هذا البرنامج الفرعي Procedure (الإجراء) تابع ما يلي :

- استدعاء البرنامج الفرعي Procedure CALLING.
- وقع زر أمر على خلفية البرنامج Form ثم أكتب في دالته السطر التالي:

```
S1()
```

- في هذا السطر تم استدعاء البرنامج الفرعي بكتابة اسمه S1.
- نفذ البرنامج ثم اضغط الزر CALL S1 تلاحظ ظهور الرسالة التي تم كتابتها في البرنامج الفرعي S1 وهي الرسالة THIS LINE CALLED FROM SUB S1.

- بمعامل واحد.
- أكثر من معامل.

كما تلاحظ أنه بعد إنشاء البرنامج الفرعي يمكن استدعائه من أي مكان بكتابة اسمه فقط إذا لم يكن له معاملات ، و كتابة اسمه و أمامه المعاملات بترتيبها إذا كان له معاملات .
و أيضا تلاحظ إمكانية أنشاء البرنامج الفرعي مرة واحدة و أستدعائة أكثر من مرة وأيضا البرنامج الفرعي من نوع sub (الإجراء) لا يعيد قيمة مثل الدوال كما سيلي.

القيمة الافتراضية للمعامل Optional Argumen

يمكن تحديد معامل أو أكثر على أنه Optional وفى هذه الحالة عند استدعاء البرنامج الفرعى Sub يمكن ارسال قيمة المعامل أو عدم إرسالها.
بشرط اعطاء هذا المعامل قيمة افتراضية عند الانشاء كما يظهر من السطور التالية:

```
Sub s1(Optional ByVal st As String = "hi")
    MsgBox(st)
End Sub
```

فى هذه السطور فى السطر الاول تم تعريف البرنامج الفرعى S1 مع اضافة معامل له بالاسم st كما سبق ولكن الجديد هو اضافة كلمة Optional التى تعنى ان ارسال قيمة هذا المعامل عند الاستدعاء ليس إجبارى مع تحديد قيمة افتراضية لهذا المعامل وهى "hi"
فى هذه الحالة يمكن استدعاء هذا الإجراء Sub بصورتين.
الأولى :عدم إرسال معامل بكتابة اسم الإجراء ()S1 فقط كما فى السطر التالى:

```
S1()
```

الثانية: إرسال معامل بكتابة اسم الإجراء ()S1 مع ارسال المعامل كما فى السطر التالى:

```
S1("new Value Sent")
```

من فضلك

- جرب اضافة زرى أمر Button وكتابة الأمر الأول فى دالة الزر الأول Button1
- وكتابة الأمر الثانى فى دالة الزر الثانى Button2.

ثم نفذ البرنامج وجرب استعماله وشاهد النتيجة

الفرق بين ByVal وByRef

فى الامثلة لاحظت كتابة الكلمة ByVal تلقائيا أمام المعاملات كما فى المثال السابق حيث كتبت أمام المعامل st وهذه الكلمة اختصار للعبارة By Value أى بالقيمة.

وهى تعنى أن استدعاء هذا المعامل يتم بإرسال قيمته إلى البرنامج الفرعى الذى يأخذ هذه القيمة ويجرى عليها العمليات المحددة فى سطوره.

فكما سبق فى المثال السابق يتم ارسال قيمة المعامل st وهى عبارة وفى البرنامج الفرعى S1 يتم اظهر هذه العبارة باستعمال الدالة Msgbox.

ولكن السؤال هو هل يمكن استعمال كلمة أخرى غير ByVal ؟

نعم توجد الكلمة ByRef وهى اختصار للعبارة By Reference وهى لا تؤدى إلى إرسال قيمة المعامل بل تؤدى إلى ارسال عنوان المتغير للتعامل معه بدلا من القيمة.

ولتوضيح ذلك تابع المثال التالى:

1- قم بكتابة البرنامجين الفرعين Inc1(),Inc2() داخل Module1 كما فى السطور التالية:

```
Sub inc1(ByVal v1 As Integer)
    v1 += 1
End Sub
Sub inc2(ByRef v1 As Integer)
    v1 += 1
End Sub
```

فى هذه السطور فى البرنامج الفرعى الأول تم تحديد المعامل مع تحديد كيفية التعامل معه وهى التعامل بالقيمة.

برنامج فرعي من نوع function (الدالة)

يأخذ البرنامج الفرعي من نوع function نفس خطوات الإنشاء و لكنه يختلف عن النوع sub في أنها لابد أن تعيد قيمة مثل الدوال الجاهزة في اللغة مثل دالة abs و دالة sin و غيرها من الدوال .

و لتوضيح ذلك تابع ما يلي :

مثال :

1- ابدأ برنامج جديد من النوع Windows Application.

2- أضف ملف أوامر MODULE.

3- أفتح ملف الأوامر لحصل علي صفحة بيضاء لكتابة أوامر.

4- أكتب السطور التالية :

```
Function f1(ByVal N1 As Integer, ByVal N2 As Integer)
    Dim s1 As Integer
    s1 = N1 + N2
    Return s1
End Function
```

في هذه السطور

- في السطر رقم 1 تم أنشاء دالة (FUNCTION) بالاسم F1 و لها معاملين الأول N1 و الثاني N2 و كلاهما من النوع Integer يتم استقبالهما من استدعاء الدالة (FUNCTION) التى تأخذ الاسم F1.
- في السطر رقم 2 يتم جمع المعاملين N2 , N1 ووضع النتيجة في المتغير S1.
- في السطر رقم 3 يتم اعادة قيمة S1 باستعمال الأمر Return وهو نتيجة استدعاء الدالة .

5- وقع زر أمر Button على خلفية البرنامج Form و أكتب في دالته السطور التالية :

```
R=F1 (5,7)
MSG BOX " RETURENED VALUE IS : " & R
```

- في السطر رقم 1 تم استدعاء الدالة F1 مع إرسال القيمة 7.5 كمعاملات للدالة فيتم أرسالها إلي سطور الدالة التي تستقبلها في المعاملين N1,N2 و تجمعهما و تعيد النتيجة التي تستقبل في المتغير R .
- في السطر رقم 2 يتم عرض القيمة المرتجعة و هي القيمة R مع رسالة تفيد ذلك.

- نفذ البرنامج و جرب النقر على زر الأمر تلاحظ ظهور الرسالة راجع الخطوات تجد أنها :

1. إضافة ملف أوامر Modale.
2. كتابة سطور الدالة FUNCTION بمعاملتها و عملياتها و أعادة النتيجة باستعمال الامر Return (في هذا المثال اسم الدالة F1).

1- استدعاء الدالة من أي مكان بكتابة أسم الدالة و معاملاتها ووضع النتيجة في متغير و الفرق الجوهري بين الدالة FUNCTION و البرنامج الفرعي SUB هو أن الدالة لابد أن تعيد قيمة بينما البرنامج الفرعي SUB (الإجراء) لا تعيد قيمة.

2- كما تكمن أهمية البرامج الفرعية Procedures و الدوال FUNCTION في إمكانية إنشائها مرة واحدة و استدعائها أكثر من مرة مما يوفر أعادة كتابة السطور مرة أخرى .

برنامج من النوع Console Application

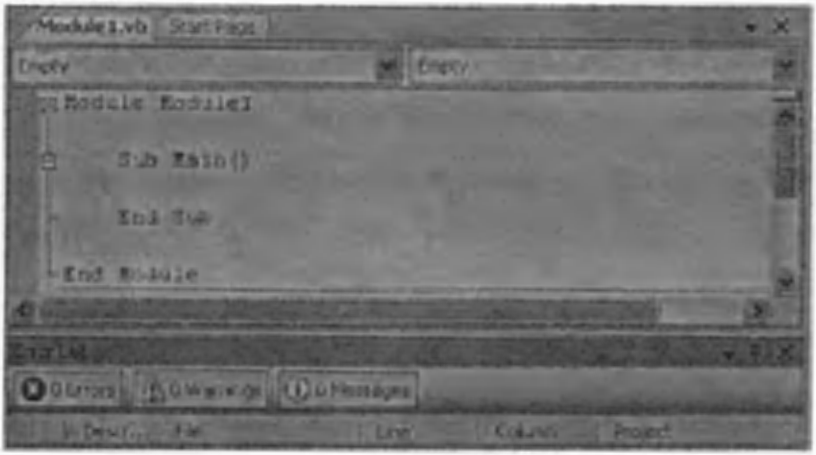
يوفر Visual Basic.Net أنواع اخرى من التطبيقات غير النوع التقليدى الذى نتعامل معه حتى الآن وهو Windows Application ومن هذه الأنواع برنامج من النوع Console Application وهو عبارة عن برنامج بدون واجهة (Form) وبالتالى عند تنفيذه تحصل على النتيجة فى شاشة خارجية مثل شاشة نظام التشغيل Dos ويفيد ذلك فى حالة عدم احتياجك لواجهة البرامج Form فقط تريد تحقيق عمليات فى الخلفية بدون خلفية أو تريد اختبار عمليات

ولتوضيح ذلك تابع الخطوات التالية:

1- قم بانشاء تطبيق جديد ولكن هذه المرة حدد النوع ConsoleApplication كما فى الشكل (4-5) .

الشكل (5-4)

2-اضغط الزر OK تحصل على تطبيق من هذا النوع كما في الشكل (5-5) .



الشكل (5-5)

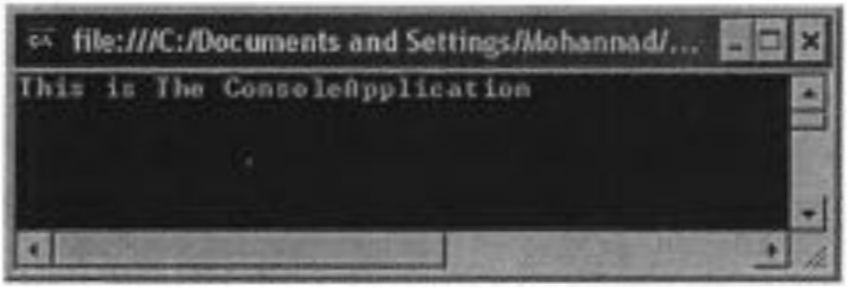
- في هذا الشكل نلاحظ وجود إجراء Sub بالاسم ()Main وهو الذي ينفذ عند تشغيل البرنامج اكتب السطر التالي:

```
Console.WriteLine("This is The ConsoleApplication ")
```

في هذا السطر يتم استدعاء الدالة WriteLine من الفصيلة Console

لطباعة العبارة This is The ConsoleApplication

3- شغل البرنامج بالضغط على Ctrl+F5 تحصل على نتيجة التنفيذ كما في الشكل (6-5)



الشكل (6-5)

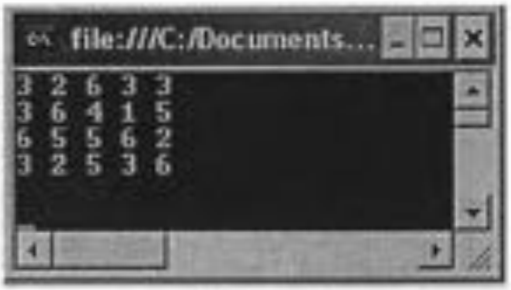
4- اضغط اى مفتاح تعود إلى بيئة التطوير.

ويمكن استعمال النوع من البرامج لاختبار قواعد البرمجة بسهولة حيث أنه أسرع حيث لا يحتاج إلى تحميل واجهة البرنامج Form والأدوات ولتوضيح ذلك تابع الخطوات التالية:

1- عدل سطور المثال السابق ConsoleApplication لتصبح كما في السطور التالية:

```
1:Dim rObject As Random = New Random()
2:Dim rNumber As Integer
3:Dim Resurt As String = ""
4:Dim i As Integer
5:For i = 1 To 20
6:rNumber = rObject.Next(1, 7)
7:Resurt &= rNumber & " "
8:If i Mod 5 = 0 Then
9:Resurt &= vbCrLf
10:End If
11:Next
12:Console.WriteLine(Resurt)
```

2- نفذ هذا البرنامج تحصل على النتيجة كما في الشكل التالي (7-5):



الشكل (7-5)

فى لغات C-Like

فى لغة Java

لا توجد Procedures توجد دوال فقط ويطلق عليها methods أو functions

وفى هذا المثال يتم انشاء دالة Method بسيطة لتوضيح فقط كيفية انشاء واستدعاء الدالة Method ولتحقيق ذلك اكتب السطور التالية:

```
1  public class Method1 {
2      public void Hello()
3      {
4          System.out.println("Hello from Method 1 ");
5          }
6      public static void main(String[] args)
7      {
8      Method1 Obj1=new Method1();
9      Obj1.Hello();
10     }
11 }
12
```

فى هذه السطور

فى السطر رقم 2 تم إنشاء دالة Method جديدة بالاسم ()Hello.

فى السطر رقم 4 وداخل هذه الدالة .. تقوم الدالة بطباعة العبارة "Hello from Method1".

فى السطر رقم 8 يتم تعريف متغير هدف من الفصيلة وهذا ما سوف نتناوله فى فصول البرمجة باستعمال الاهداف OOP.

فى السطر رقم 9 يتم استدعاء الدالة الجديدة ()Hello.

1.Types of methods أنواع الدوال

بعد إنشاء دالة بسيطة وتناول كيفية الانشاء وكيفية الاستدعاء .. تعالى معنا نستمل معا باقي عناصر هذا الموضوع وذلك بتناول ... أنواع الدوال.

وفيها يحدد نوع الدالة بنوع القيم المرتجعة من الدالة فاذا كانت الدالة تعيد قيمة من نوع int كانت الدالة من النوع int واذا كانت الدالة تعيد قيمة من النوع char كانت الدالة من النوع char وهكذا ، ولتوضيح ذلك تابع معنا الأمثلة التالية:

مثال :

المثال التالى يوضح كيفية انشاء دالة Method من النوع int كما فى السطور التالية :

```
1  public class Method1 {
2      public int Sum(int a ,int b )
3      {
4  int s;
5  s=a+b;
6  return s;
7          }
8      public static void main(String[] args)
9      {
10       int r;
11     Method1 Obj1=new Method1();
12     r=Obj1.Sum(10,15);
13
14       System.out.println("the sum is :"+r);
15     }
16 }
```

فى هذه السطور

فى السطر رقم 2 تم انشاء الدالة ()Sum من النوع int لانها تعيد قيمة int وهما معاملان من النوع int .

فى السطر رقم 5 يتم جمع قيم المعاملان ووضع النتيجة فى المتغير s .

فى السطر رقم 6 يتم استعمال الامر return لاعادة قيمة المتغير s وبالتالى عند استدعاء هذه الدالة تعود قيمة المتغير s .

فى السطر رقم 12 تم استدعاء الدالة ()Sum مع ارسال القيمتين 10و15 كمعاملات للدالة وبالتالى يتم إرسال هذان القيمتين إلى الدالة التى تقوم بجمعهما واعادة النتيجة التى توضع فى المتغير r .

وفى النهاية يتم طباعة قيمة المتغير r وهى القيمة المرجعة من استدعاء الدالة .

مثال : دالة تعيد قيمة حرفية

وهذا المثال يستعمل التركيب Switch لاختبار الأرقام (0,1,2,...) وتحويلها إلى العبارة المقابلة فمثلاً يحول الرقم 1 إلى one وهكذا.وهو يشابه فكرة تفقيط الشيكات.

ولتحقيق ذلك قم بكتابة السطور التالية:

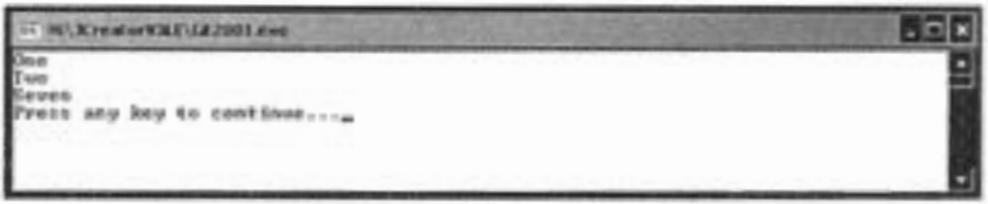
```
1  public class Method2 {
2
3      String Convfunc (int v) {
4  switch (v)
5    {
6  case 0 : return "Zero";
7
8  case 1 : return "One";
9  case 2 : return "Two";
10 case 3 : return "Three";
11 case 4 : return "Four";
12 case 5 : return "Five";
13 case 6 : return "Six";
14 case 7 : return "Seven";
15 case 8 : return "Eight";
16 case 9 : return "Nine";
17   default : return "";
18 }
19   }
20 public static void main(String[] args)
21   {
22   Method2 CN= new Method2();
23  String wordNum = CN. Convfunc (1);
24   System.out.println (wordNum);
25
26    wordNum = CN. Convfunc (3);
27   System.out.println (wordNum);
28
29    wordNum = CN. Convfunc (7);
30   System.out.println (wordNum);
31   }
32 }
```

فى هذه السطور

- فى السطر رقم 1 يتم إنشاء فصيلة بالاسم Method2.
- وفى السطر رقم 2 يتم إنشاء دالة عضوه بالفصيلة بالاسم Convertfunc.
- وفى السطور من 5 إلى 17 تم اختبار قيمة المتغير v الذى تستقبله الدالة وإعادة أحد الكلمات المقابلة حيث ان الدالة تعيد كلمة (string) حسب نوعها وبالتالى إذا كان الرقم 0 تعيد الدالة كلمة Zero وإذا كان 1 تعيد one وهكذا.

- وفى السطر 17 الكلمة default تعتبر مسافة " " إذا لم تتحقق أى حالة.
- فى السطر 20 تبدأ الدالة ()main التى تتولى سير البرنامج.
- فى السطر 22 يتم تعريف المتغير CN من نوع الفصيلة Convert Numb كعنصر object
- فى السطر 23 يتم تعريف المتغير Word Num من نوع string مع تخزين فيها نتيجة استدعاء الدالة Convert func مع إرسال القيمة 1 كمعامل فتقدم الدالة بتحويل 1 إلى بالمثل 3,2 وتصبح النتيجة one two three توضع فى المتغير word Num.
- فى السطر رقم 24 يتم طباعة محتوى المتغير word Num وهو one two three.

قم بكتابة البرنامج كما فى الشكل ثم قم بترجمة وتنفيذ البرنامج
تحصل على نتيجة التنفيذ كما فى الشكل (8-5)



الشكل (8-5)

دوال من غير قيم مرتجعة Methods with no returning values

كما اتفقنا يحدد نوع الدالة حسب القيمة المرتجعة ، فماذا إذا كانت الدالة لا تعيد قيمة... فى هذه الحالة تسمى الدالة بلا عائد أو void ، وهى الدوال التى تقوم بتنفيذ عملية محددة بدون اعادة نتيجة مثل دوال الرسم ودوال الصوت ...

وهذا ما تم فى المثال الاول حيث تم إنشاء دالة لا تعيد قيمة .. فقط تطبع رسالة وبالتالى كان نوعها void.

وهذا ما يقابل برنامج فرعى من نوع procedure فى لغة VB.NET حيث لا تحتوى لغات C Like برامج فرعية من نوع procedure ولكن فى المقابل تحتوى على دالة من نوع void بنفس الوظيفة

فى Oracle PL/SQL

إعداد الدوال والبرامج الفرعية Building procedures & functions

وفى هذه الفترة سوف نتناول خطوات ومتطلبات إعداد الدوال والبرامج الفرعية procedures & functions وذلك من خلال النقاط التالية:

- خطوات إنشاء Procedures.
- إرسال معاملات للبرامج الفرعية Parameter passing.
- إنشاء دوال Functions.

إنشاء Procedure

```
    PROCEDURE DEPT_SEARCH IS
    BEGIN
      SELECT DNAME INTO :TEXT_ITEM6  FROM DEPT WHERE
DEPTNO=:TEXT_ITEM5;
    END;
```

فى هذه السطور تعدل بما يناسب الكود

- فى السطر الأول تم انشاء PROCEDURE بالاسم DEPT_SEARCH مع تحديد معامل بالاسم P_DEPTNO من النوع NUMBER.
- وهذا يعنى أنه عند استدعاء هذه الدالة لابد من ارسال معامل من النوع NUMBER.
- فى السطر الثانى وبعد IS تم الاعلان عن متغير بالاسم V_DNAME من نوع حقل DNAME الموجود فى جدول DEPT.
- فى السطر الثالث يبدأ البرنامج الفرعى بالأمر BEGIN.
- فى السطر رقم 4 تم استدعاء أمر SELECT لعرض اسم الادارة DNAME فى المتغير V_DNAME.
- فى السطر رقم 5 يتم وضع قيمة المتغير V_DNAME فى مربع النص TEXT_ITEM5: حتى يظهر على الشاشة وبالتالى أصبح لدينا برنامج فرعى PROCEDURE تأخذ رقم الإدارة و تعرض اسم الإدارة.

تنفيذ البرنامج الفرعى PROCEDURE CALLING

يكتب سطر استدعاء البرنامج الفرعى DEPT_SEARCH كما فى السطر.

```
DEPT_SEARCH();
```

إرسال المعاملات Passing parameters

من الضرورى جداً الإعلان عن متغيرات كمعاملات داخل أقواس الدالة function أو البرنامج الفرعى procedure وذلك لأن المعاملات توسع استعمالها فمثلاً الدالة sin(x) الموجودة باللغة يمكن استدعائها عدد كبير من المرات مع إرسال أى قيمة مكان المتغير كمعامل للدالة وبالتالى يمكن للدالة حساب SIN() ((جا)) أي قيمة ، فى حين لو كانت هذه الدالة بدون معاملات فبالتأكيد فهى تحسب قيمة جا SIN() لقيمه واحدة فقط ولابد من إنشاء دالة SIN() لكل قيمه وهذا هو المستحيل، والمعاملات تأخذ ثلاثة أنواعها هي:

النوعIN : وهو الذى يستقبل قيمة عند استدعاء الدالة ولا يمكن إعادة قيمة فيه لمكان الاستدعاء.

النوع OUT : وهو الذى يأخذ قيمة تعود لمستدعى الدالة فقط ولا يستعمل لاستقبال قيم عند الاستدعاء.

النوع INOUT: وهو النوع الذى يسمح باستقبال القيم عند الاستدعاء فيه وكذلك إعادة قيم للخروج فيه.

ويتضح ذلك عند استدعاء البرنامج الفرعى الذى أنشأناه من قبل حيث يتم استدعائه بدون معاملات ولكن يتم تعريف المعاملات عند الانشاء ثم استعمالها عند الاستدعاء ولتوضيح ذلك تابع الخطوات التالية:

فى هذا السطر تم استدعاء البرنامج MYPROC1 مع إرسال القيمة 398 5 كمعامل له وبالتالى يتم استقبال القيمة في المتغير DEL-EMPID المعرف كمعامل من نوع IN ثم استعماله داخل السطور لمسح الموظف الذي كوده 398 5 وبالتالى يمكن استدعاء البرنامج MYPROC1 اكثر من مرة مع تغيير كود الموظف المطلوب حذفة

5- اضغط الزر OK تظهر صفحة كتابة الاوامر CODE EDITOR اكتب بها سطور البرنامج الفرعى التالى:

```
PROCEDURE DEPT_SEARCH(P_DEPTNO NUMBER) IS
V_DNAME DEPT.DNAME%TYPE;
BEGIN
  SELECT DNAME INTO V_DNAME FROM DEPT WHERE DEPTNO=P_DEPTNO;
  :TEXT_ITEM5:=V_DNAME;
END;
```

فى هذه السطور:

- فى السطر الأول تم انشاء PROCEDURE بالاسم DEPT_SEARCH مع تحديد معامل بالاسم P_DEPTNO من النوع NUMBER.
- وهذا يعنى أنه عند استدعاء هذه الدالة لابد من إرسال معامل من النوع NUMBER.
- فى السطر الثانى وبعد IS تم الاعلان عن متغير بالاسم V_DNAME من نوع حقل DNAME الموجود فى جدول DEPT.
- فى السطر الثالث يبدأ البرنامج الفرعى بالامر BEGIN.
- فى السطر رقم 4 تم استدعاء أمر SELECT لعرض اسم الإدارة DNAME فى المتغير V_DNAME.
- فى السطر رقم 5 يتم وضع قيمة المتغير V_DNAME فى مربع النص TEXT_ITEM5: حتى يظهر على الشاشة .

وبالتالى أصبح لدينا برنامج فرعى PROCEDURE تأخذ رقم الإدارة و تعرض اسم الإدارة.

استدعاء البرنامج الفرعى Procedure

ويتم استدعاء البرنامج الفرعى DEPT_SEARCH بنفس الطريقة ولكن مع تحديد معامل ثانى وهو معامل اسم الإدارة وذلك كما فى الشكل :

```
DEPT_SEARCH(:TEXT_ITEM4,:TEXT_ITEM5);
```

- فى هذه السطور تم استدعاء البرنامج الفرعى DEPT_SEARCH بمعامل IN وهو TEXT_ITEM4 الذى يكتب المستخدم به كود الإدارة
- والمعامل الثانى TEXT_ITEM5 المعامل من النوع OUT الذى يظهر به اسم الإدارة.

- وبهذا اصبح البرنامج الفرعي DEPT_SEARCH غير مرتبط بعناصر معينة بل يصلح للاستعمال في اى مكان

انشاء معامل من نوع IN OUT

بالإضافة لإمكانية إنشاء معامل من IN للادخال و معامل من النوع OUT للاخراج يمكن إنشاء معامل من النوع IN OUT لاستعماله للإدخال والإخراج في نفس الوقت ولتوضيح ذلك تابع الخطوات التالية:

1- قم بتعديل سطور البرنامج لتصبح كما في الشكل (9-5).

```
PROCEDURE DEPT_SEARCH(P_INOUT  IN OUT NUMBER) IS
BEGIN
  SELECT SAL INTO P_INOUT FROM EMP WHERE EMPNO=P_INOUT;

END;
```

الشكل(9-5)

في هذه السطور:

2- اكتب سطور استدعاء البرنامج الفرعي لتصبح كما في الشكل :

```
DEPT_SEARCH(:TEXT_ITEM4);
```

في هذه السطور

إنشاء دالة CREATING PL/SQL FUNCTION

الدالة Function في جميع اللغات بما فيها PL/SQL هي برنامج فرعي مثل PROCEDURE ولكن يختلف عنه في أن الدالة لا يمكن أن تعيد أكثر من قيمة فمثلاً الدالة SQRT(X) الموجودة باللغة تعيد قيمة واحدة فقط وهي الجذر التربيعي للقيمة X وهكذا أي دالة أخرى.

ويتم إنشاء الدالة باستعمال الأمر Create function في كلا من بيئة البرنامج SQL Pluse والبرنامج Procdure Builder كما في السطور التالية:

```
CREATE [OR REPLACE] FUNCTION function_name
    [ (argument1 [mode1] datatype1 [,argument2 [mode2] datatype2, ...] ) ]
RETURN datatype
IS/AS
    [ Local Declaration ]
Begin
      Executable Statements
[Exception
      exception handlers ]
END [function_name];
```

ولتوضيح ذلك تابع المثال التالي:

```
1: Create function doub_salary (EID IN NUMBER (1 ))
2: RETURN NUMBER
3: IS
4: OLDSA NUMBER;
5: BEGIN
6: Select salary into OLDSA FROM EMP
7: WHERE EMPNO = EID;
8: OLDSA = OLDSA * OLDSA;
9: RETURN OLDSA;
1 : END;
```

ملحوظة 1:

البرنامج PROCEDURE قد يستقبل معامل أو اكثر أوقد لا يستقبل وكذلك يعيد نتيجة أو أكثر أو لا يعيد نهائياً .

الدالة function قد تستقبل معامل أو أكثر أوقد لا تستقبل ولكن لا تستطيع إلا أن تعيد قيمة واحدة.

ملحوظة 2:

في حالة الدالة function لا يصلح أن تحدد نوع المعامل IN أو OUT أو IN OUT لأن الدالة لا تعيد إلا قيمة واحدة وتعيدها كما في المثال السابق باستعمال الأمر RETURN.

ملحوظة 3

فى حالة الدوال Function لابد أن تكون نوع المتغيرات و المعاملات من الأنواع الأساسية فى Oracle ،الدالة لابد أن تعمل على قيمة واحدة single row function وليس group لا ينفذ لك return واحدة فى الدالة لكن يمكن تواجد أكثر من جملة return لا تنفذ إلا واحدة.

القيم الافتراضية لمعاملات الدوال default parameters

من الامكانيات المتاحة عند التعامل مع الدوال إمكانية إعطاء قيم ابتدائية لمعاملات الدوال default parameters هذه القيم تفيد فى حالة استدعاء الدالة بدون قيم معاملات تأخذ المعاملات القيم الابتدائية وفى حالة إرسال قيم للمعاملات تأخذ معاملات الدالة القيم الجديدة ولتوضيح ذلك تابع المثال التالى:

مثال قيم المعاملات الافتراضية default parameters

فى هذا المثال نعرض كيفية إنشاء اجراءات بقيم افتراضية للمعاملات:

```
/*
 * AddNewBook.sql
 * This procedure will insert a new book into the books table.
 * It also demonstrates default parameters.
 */

CREATE OR REPLACE PROCEDURE AddNewBook(
  p_ISBN IN books.ISBN%TYPE,
  p_Category IN books.category%TYPE := 'Oracle Server',
  p_Title IN books.title%TYPE,
  p_NumPages IN books.num_pages%TYPE,
  p_Price IN books.price%TYPE,
  p_Copyright IN books.copyright%TYPE DEFAULT
    TO_NUMBER(TO_CHAR(SYSDATE, 'YYYY')),
  p_Author1 IN books.author1%TYPE,
  p_Author2 IN books.author2%TYPE := NULL,
  p_Author3 IN books.author3%TYPE := NULL) AS
```

```
BEGIN
  -- Insert a new row into the table using the supplied
  -- parameters.
  INSERT INTO books (isbn, category, title, num_pages, price,
                     copyright, author1, author2, author3)
  VALUES (p_ISBN, p_Category, p_Title, p_NumPages, p_Price,
          p_Copyright, p_Author1, p_Author2, p_Author3);
END AddNewBook;
/
```

في هذه السطور:

تم إنشاء procedure لها معاملات وتم اعطاء المعامل p_Category قيمة ابتدائية وهي القيمة 'Oracle Server' وبالتالي عند الاستدعاء إذا لم يتم إرسال قيمة لهذا المعامل يتم استعمال القيمة الافتراضية وإذا تم إرسال قيم يتم استعمال القيم الجديدة.

وفيما يلي أكثر من حالة لاستدعاء هذا الإجراء لتجربة المعاملات ذات القيم الافتراضية.

الاستدعاء الأول

```
-- We can avoid passing author2 and author3 since they have
-- default values:
BEGIN
  AddNewBook('00000000', 'Oracle Basics', 'A Really Nifty Book',
             500, 34.99, 2004, 1);
END;
/
ROLLBACK;
```

في هذه السطور يتم استدعاء الإجراء AddNewBook مع إرسال قيم لجميع المعاملات وتم تجاهل القيم الافتراضية التي فرضت عند تعريف الإجراء Procdure في حين كان من الممكن تجاهل هذه المعاملات وبالتالي تخزين القيم الافتراضية.

الاستدعاء الثاني

```
-- The same example, using named notation:
BEGIN
  AddNewBook(p_ISBN => '0000000',
             p_Category => 'Oracle Basics',
             p_Title => 'A Really Nifty Book',
```

```
        p_NumPages => 500,
        p_Price => 34.99,
        p_Copyright => 2004,
        p_Author1 => 1);
END;
/
ROLLBACK;
```

في هذه السطور:

تم الاستدعاء بقيم مختلفة .

الاستدعاء الثالث

```
-- Using all the default values:
BEGIN
  AddNewBook(p_ISBN => '00000000',
        p_Title => 'A Really Nifty Book',
        p_NumPages => 500,
        p_Price => 34.99,
        p_Author1 => 1);
END;
/

ROLLBACK;
```

في هذه السطور تم الاستفادة بجميع القيم الافتراضية Default Paramters