اساسيات البرمجة باستخدام لغة البيسك المرئي 2015

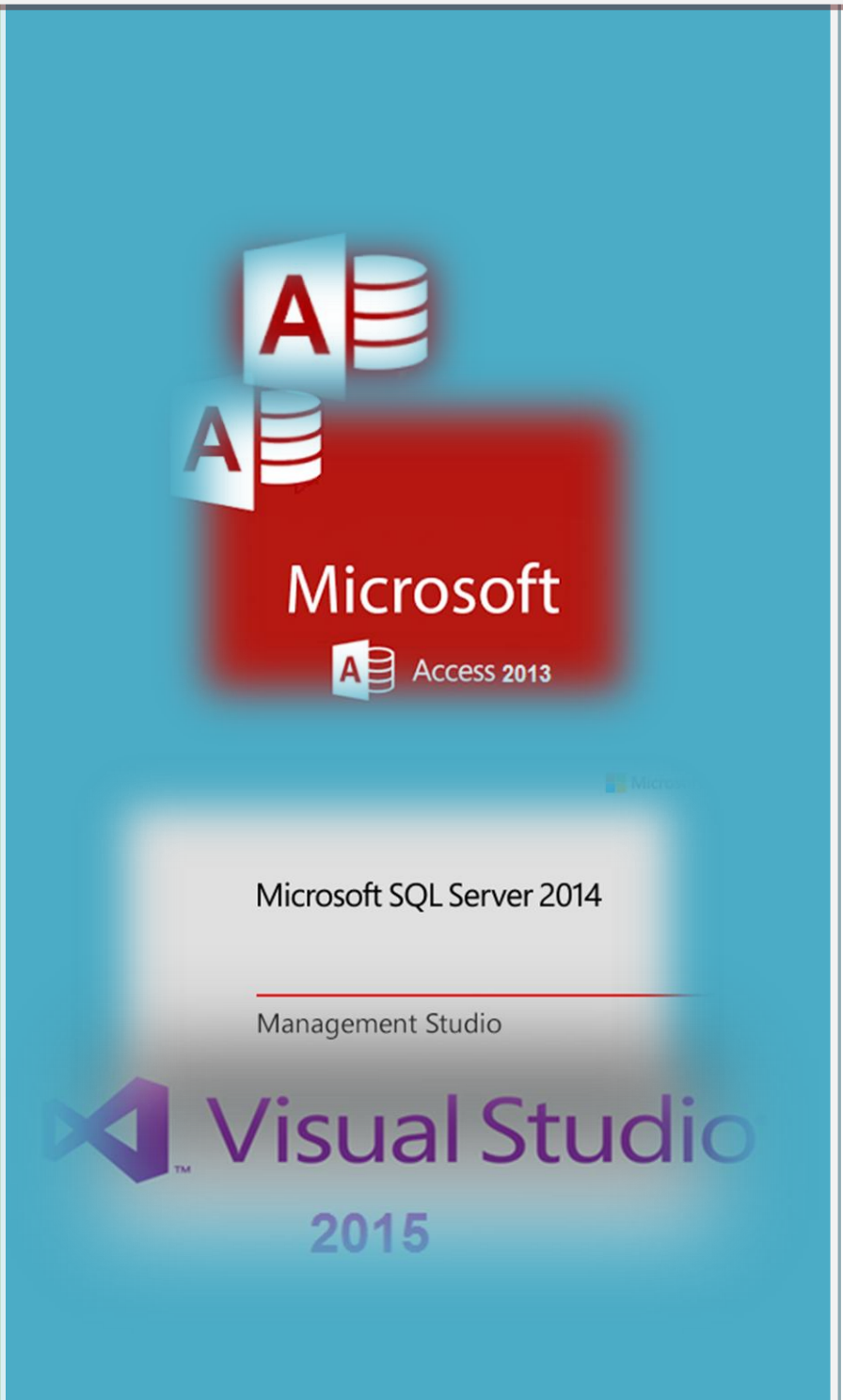
Programing Fundamental Using visual Basic.Net 2015

برمجة تطبيقات واجهات المستخدم النصية (Console Application)

تصميم وبرمجة تطبيقات واجهات المستخدم الرسومية (GUI)

الاتصال بقواعد البيانات (Access 2013 & Sql Server 2014)

Visual Studio.net 2015



إعداد: سالم مسعود الدروقي

قسم الحاسوب-كلية التربية -الخمس

2016م

بسم الله الرحمن الرحيم

وما أوتيتم من العلم إلا قليلا

صدق الله العظيم

الإهداء

إلى روح أبي الطاهرة..... رحمة الله

إلى والدتي الغالية .... أطال الله في عمرها

وإهداء خاص إلى عائلتي (زوجتي الغالية وأبنائي والأعزاء)

لعلي أجد عدراً أقدمه لهم لانشغالي عنهم طيلة فترات إعداد هذا الكتاب....

إلى كافة الزملاء بكلية التربية/ الخمس

إلى جيل الأساتذة الذين علموني طيلة فترات دراستي

وإلى كل من يقرأ هذا الكتاب

إلى كل هؤلاء أهدي ثمرة هذا الجهد المتواضع....

المؤلف: سالم الدروقي

ملخص الكتاب

يتألف هذا الكتاب من اثنا عشر فصلاً وفق الاتي:

الفصل الأول: يقدم هذا الفصل مجموعة من المفاهيم الأساسية في دراسة لغات البرمجة من خلال تقديم نبده مختصرة عن الحاسوب وفكرة عملة وخطوات حل المسائل البرمجية باستخدام الحاسوب، كما يقدم هذا الفصل نبده عن اساسيات تعلم لغة البيسك من خلال الحديث عن مكونات هذه اللغة والكلمات المحجوزة فيها وطرق تعريف المتغيرات والثوابت وانواعها.

الفصل الثاني: يقدم هذا الفصل شرحاً وافياً لجمل الإدخال والإخراج في لغة البيسك المرئي تحت بيئة تطبيقات الكونسل (Console Application)، كما يقدم هذا الفصل مجموعة من التمارين المحلولة لتوضيح طريقة كتابة جمل الإدخال والإخراج في البرنامج مع توضيح فكرة عملها من خلال كتابة خوارزمية البرنامج ورسم المخطط الانسيابي له.

الفصل الثالث: يقدم هذا الكتاب شرحاً وافياً لطريقة استخدام جمل الشرط في البرنامج وهي جملة "IF" وكذلك جملة "Select Case"، وكذلك يقدم مجموعة من التمرين المحلولة لتوضيح طريقة عمل الجمل الشرطية في العديد من الحالات البرمجية.

الفصل الرابع: يقدم هذا الفصل شرحاً وافياً لمفهوم التكرار في البرمجة من خلال دراسة جمل التكرار في لغة البيسك، كما يقدم العديد من الأمثلة المحلولة لتوضيح فكرة عمل كل حلقة من حلقات التكرار.

الفصل الخامس: يقدم هذا الفصل شرحاً وافياً عن طريقة استخدام المصفوفات في لغة البيسك المرئي، كما يقدم مجموعة من التمارين المحلولة التي توضح فكرة عمل المصفوفات وطرق تخزين واسترجاع البيانات في المصفوفات.

الفصل السادس: يقدم هذا الفصل شرحاً وافياً عن طريقة استخدام البرامج الفرعية (الإجراءات والدوال) في لغة البيسك، كما يقدم مجموعة من التمارين المحلولة التي توضح فكرة عمل البرامج الفرعية وطرق تعريفها واستدعائها في البرنامج.

الفصل السابع: يتناول هذا الفصل بالشرح طريقة إنشاء البرامج ذات الواجهات الرسومية في لغة البيسك المرئي من خلال تقديم شرح وافي لمجموعة من أهم الأدوات المستخدمة في تصميم الواجهات الرسومية وطرق استخدامها وأهم خصائصها.

الفصل الثامن: يعتبر هذا الفصل مكملاً للفصل الذي يسبقه حيث يقدم هذا الفصل شرحاً وافياً لدوال الإدخال الإخراج في لغة البيسك المرئي عند التعامل مع الواجهات الرسومية، كما يقدم شرحاً لطريقة تصميم وبرمجة مجموعة متنوعة من التطبيقات ذات الواجهات الرسومية.

الفصل التاسع: يقدم هذا الفصل شرحاً وافياً لأساسيات التعامل مع قواعد البيانات من خلال تقديم تطبيق عملي لتصميم قاعدة بيانات لنظام تسجيل طلابي، وكذلك يقدم هذا الفصل شرحاً مفصلاً لطريقة تصميم تطبيق يحتوي على واجهات رسومية باستخدام لغة الفيجوال بيسك لكي يتم شرح طريقة ربط هذا التطبيق بقاعدة البيانات في الفصول التالية لهذا الفصل.

الفصل العاشر: يقدم هذا الفصل شرحاً وافياً لطريقة إنشاء قاعدة بيانات التي تم تصميمها في التاسع باستخدام نظام إدارة قواعد البيانات "Microsoft Access 2013"، كما يقدم شرحاً تفصيلياً لطريقة ربط قاعدة البيانات من نوع "Access" مع واجهات التطبيق الذي تم تصميمه في الفصل الذي يسبق هذا الفصل.

الفصل الحادي عشر: يقدم هذا الفصل شرحاً وافياً لطريقة إنشاء قاعدة بيانات التي تم تصميمها في التاسع باستخدام نظام إدارة قواعد البيانات "Microsoft sql server 2014"، كما يقدم شرحاً تفصيلياً لطريقة ربط قاعدة البيانات من نوع " sql server" مع التطبيق الذي تم تصميمه في الفصل التاسع.

الفصل الثاني عشر: يقدم هذا الفصل شرحاً مختصراً لطريقة تصميم وبرمجة التقارير عند التعامل مع قواعد البيانات.

فهرس الموضوعات

المقدمة

بسم الله الرحمن الرحيم والصلاة والسلام على خاتم الأنبياء والمرسلين سيدنا محمد النبي الامي الكريم وعلى اله وصحبة اجمعين ومن اتبع هداه الى يوم الدين..... وبعد

تعتبر لغة البيسك المرئي ويطلق عليها ايضاً "فيجوال بيسك" وتختصر إلى "VB" من اللغات الأساسية التي تدرس بشكل واسع في كافة الجامعات والمعاهد العليا وكذلك المتوسطة في ليبيا، وذلك نظراً لأهمية هذه اللغة لكونها لغة برمجة سهلة الفهم و التعلم والاستخدام و الذي يُرجع إلى قربها الشديد من لغة الإنسان (اللغة الانجليزية) من خلال استخدامها للمصطلحات الإنجليزية دون اللجوء إلى التغيير فيها او الاختصار، كما تعتبر لغة البيسك المرئي لغة برمجة لها امكانياتها الواسعة في تصميم وتنفيذ العديد من أنواع البرمجيات التي يتطلبها السوق المحلي، ونظراً للنقص الشديد في المراجع العربية المُقدمة في هذه اللغة فقد رئينا أن نقوم بتأليف هذا الكتاب ليكون عوناً ومرجعً لكافة طلبة وطالبات اقسام الحاسوب في مرحلة التعليم العالي في ليبيا، وكذلك أعضاء هيأة التدريس، حيث يقدم هذا الكتاب لغة البيسك من خلال تقديم العديد من الأمثلة المحلولة باستخدام واجهة المستخدم النصية التابعة لحزمة "Visual Studio" والتي ركزنا فيها على الشرح التفصيلي لطريقة استخدام جمل اللغة المختلفة، وكذلك فكرة عمل كل جملة من هذه الجمل، كما يقدم هذا الكتاب مجموعة من التطبيقات المحلولة باستخدام واجهة المستخدم الرسومية والتي سبقها شرحاً وافياً لطريقة تصميم وبرمجة واجهات المستخدم الرسومية (GUI) في لغة البيسك المرئي، كما تضمن هذا الكتاب شرحا تفصيليا لتطبيق نظام طلابي مُصغر يوضح اهم واشهر الطرق لربط تطبيق فجوال بيسك مع قواعد البيانات "Access" من خلال الإصدار 2013 و كذلك "Sql Server" من خلال الإصدار 2014، وكذلك قمنا من خلال هذا التطبيق بشرح طرق إجراء العمليات المختلفة على البيانات المخزنة في قواعد البيانات.

وفي الختام نرجو من الله العلي القدير أن نكون قد وفقنا في تقديم هذا العمل، ونسأل الله أن ينفع به البلاد والعباد.

الفصل السادس
البرامج الفرعية (الإجراءات والدوال)

6) البرامج الفرعية:

من خلال متابعة كافة الأمثلة التي تم تقديمها في الفصول السابقة نلاحظ أن كل برنامج من هذه البرامج يكتب بشكل كامل داخل البرنامج الرئيسي (Sub Main) ويتم تنفيذ أسطر هذا البرنامج ابتداءً من السطر الأول إلى أن يصل إلى السطر الاخير، في هذا الفصل سوف ندرس فكرة إمكانية كتابة شفرات برمجية (برامج فرعية) خارج البرنامج الرئيسي (Sub Main) يتم تنفيذها فقط حين استدعائها من البرنامج الرئيسي (الدالة الرئيسية).

```
Module Module1

    Sub Main()
        Statement1
        Statement2
        Statement3
    End Sub

End Module
```

البرنامج الرئيسي

والبرنامج الفرعي هو عبارة عن مجموعة من التعليمات البرمجية المجمعة تحت اسم واحد يطلق عليها اسم برنامج فرعي حيث يؤدي هذا البرنامج وظيفة محددة وله بداية وله نهاية و يكتب داخل الهيكل العام للبرنامج بين كلمتي (Module …. End Module) وخارج البرنامج الرئيسي (Sub Main) ولا يتم تنفيذه إلا بعد استدعائه من خلال كتابة اسمه في البرنامج الرئيسي وبعد تنفيذه يتم الرجوع إلى نقطة الاستدعاء في البرنامج الرئيسي.

```
Module Module1
    Sub Main()
        Statement1
        Statement2
        Name()     ' استدعاء البرنامج الفرعي
        Statement3
     End Sub

    Sub  Name ()
     Statement1
     Statement2
     Statement3
    End sub
End Module
```

البرنامج الرئيسي

برنامج فرعي

تساعد البرامج الفرعية في تنظيم وتنسيق هيكلية البرنامج الرئيسي من خلال تقسيم البرنامج إلى مجموعة برامج فرعية بحيث يكون لكل منها وظيفة محددة، كما تساعد في التقليل من تكرار كتابة

الشفرات البرمجية (في حالة الرغبة في تنفيذها أكثر من مره) من خلال كتابتها مرة واحدة داخل برنامج فرعي وإعادة تنفيذها أكثر من مرة (دون إعادة كتابتها) من خلال إعادة كتابة اسمها فقط، ويوجد في لغة الفيجوال بيسك نوعان من البرامج الفرعية وهما الإجراءات (Sub procedures) والدوال (Functions):

1.6) الإجراءات Sub procedures:

هي عبارة عن برامج فرعية مكونة من جملة أو مجموعة جمل برمجية لها بداية ولها نهاية بحيث تكتب بين جملتي (Sub, End Sub) و تنفذ وظيفة معينة ولا تعيد قيمة بعد انتهاء تنفيذها، حيث ترجع التحكم إلى السطر المنادي (السطر الذي تم عنده الاستدعاء) (Calling code) و تنقسم الإجراءات حسب طريقة تعريفها إلى نوعين:

اولاً: إجراءات لا تحتوي على معاملات Non Parameterized Procedure:

وهي إجراءات لا تحتاج إلى إرسال قيم معينة (ثوابت أو متغيرات) أثناء استدعائها والشكل العام لتعريفها يتمثل فيما يلي:

```
Access_Modifier  Sub sub_name ()
   Statement1
   Statement2
End Sub
```

حيث أن:

- Access Modifier: حدود الإجراء (قابيله الرؤية) والتي تحدد مستوى التعامل مع الإجراء وتأخذ إحدى القيم التالية:
  - Public (عام): بحيث انه يمكن التعرف على الإجراء (رؤية الإجراء) واستدعائه من أي مكان داخل الـ "solution".
  - Friend (صديق): بحيث أنه يمكن التعرف على الإجراء (رؤية الإجراء) واستدعائه من أي مكان في المشروع.
  - Private (خاص): بحيث أنه يمكن التعرف على الإجراء (رؤية الإجراء) واستدعائه من أي مكان في الموديل فقط، مع ملاحظة أنه في حالة عدم تحديد مستوى الوصول فإن البرنامج سوف يعتبره عام (Public).
- Sub: كلمة محجوزة للإعلان عن برنامج فرعي من نوع "إجراء ".
- Sub_Name: الاسم المميز للإجراء.

مثال: اكتب برنامج يقوم بطباعة الرسالة التالية (WellCome to first Sub) عن طريق إجراء.

```
1: Module Module1
2:     Sub x()
3:         Console.WriteLine("WellCome to first Sub")
4:     End Sub
5:    Sub Main()
6:        x()
7:         Console.ReadKey()
8:     End Sub
9:End Module
```

شرح البرنامج:

السطر الأول: تم فيه الإعلان عن موديل تحت اسم Module1.

السطر الثاني: تم فيه الإعلان عن إجراء فرعي من نوع Sub ومستوى الوصول "Private" تحت اسم "x".

السطر الثالث: تم فيه كتابة شفرة عرض الرسالة المطلوبة.

السطر الرابع: تم فيه الإعلان عن نهاية الإجراء باستخدام الكلمة المحجوزة End Sub.

السطر الخامس: تم فيه تعريف البرنامج الرئيسي (الدالة الرئيسية) في البرنامج داخل الموديل.

السطر السادس: تم فيه كتابة شفرة استدعاء الإجراء وذلك بكتابة اسمه متبوعا بقوسين ()X.

السطر السابع: تمت فيه كتابة السطر الذي يقوم بإيقاف البرنامج على شاشة التنفيذ لكي يتمكن المستخدم من متابعة الناتج.

السطر الثامن: تم فيه الإعلان عن نهاية الدالة الرئيسية باستخدام الكلمات المحجوزة "End Sub".

السطر التاسع: تم فيه الإعلان عن نهاية الموديل باستخدام الكلمة المحجوزة "End Module".

مع ملاحظة أن ترتيب تنفيذ الاسطر كما يلي: (1، 5، 6، 2، 3، 4، 7، 8، 9).

ثانياً: إجراءات تحتوي على معاملات Parameterized Procedure:

وهي إجراءات تحتاج إلى إرسال (تمرير) قيم (معاملات) معينة (ثوابت أو متغيرات) أثناء استدعائها والشكل العام لتعريفها يتمثل فيما يلي:

```
Access_Modifier  Sub subname (Parameter_List)
   Statement1
   Statement2
End Sub
```

ملاحظة: يمكن تصنيف البارامترات بناءً على آلية التمرير إلى تمرير بالقيمة "ByVal" وتمرير بالمرجع "ByRef" وتمرير مصفوفة "ParamArray".

مثال: اكتب برنامج يقوم بقراءة اسمك ثم يقوم بعرض رسالة ترحيب مع طباعة الاسم امامها وذلك باستخدام إجراء.

```
 Module Module1
1:    Private Sub x(msg As String)
2:        Console.WriteLine("wellCome  " & msg)
3:    End Sub
4:    Sub Main()
5:        Dim ms As String
6:        ms = Console.ReadLine
7:        x(ms)
8:        Console.ReadKey()
9:    End Sub
 End Module
```

السطر الأول: تم فيه الإعلان عن إجراء تحت اسم X ويحتاج إلى معامل واحد من النوع النصي "String".

السطر الثاني: تمت فيه كتابة جملة الطباعة لكي تتم طباعة الجملة "WellCome" مدموجة مع قيمة المعامل التي ستخصص له قيمة أثناء الاستدعاء في الدالة الرئيسية.

السطر الخامس: تم فيه الإعلان عن متغير من النوع النصي (يجب أن يكون من نفس نوع المعامل الخاص بالإجراء)

السطر السادس: تم فيه كتابة جملة القراءة التي تسمح للمستخدم بإدخال قيمة وتخزينها في المتغير "ms".

السطر الثامن: وتمت فيه كتابة السطر الذي يقوم باستدعاء الإجراء "x" مع تمرير (إرسال) قيمة المتغير "ms" لتكون مناظرة للمعامل |"msg" الموجود في الإجراء ليتم عرضها في جملة الطباعة.

ملاحظات:

1- تسمى المعاملات الموجودة في الدالة الرئيسية والتي سيتم إرسالها في جملة الاستدعاء بالمعاملات الفعلية اما المعاملات الموجودة في الإجراء تسمى بالمعاملات الصورية.

2- وجود المعاملات في المثال السبق إجباري، ويجب إرسال قيمتها إلى الإجراء، وفي حالة الرغبة في جعل المعامل إختياري نقوم بتعريفة بإضافة كلمة "Optional" قبل كتابة اسم المعامل كما يتم كتابة القيمة الافتراضية بعد علامة "=" بعد نوع المتغير في حالة عدم وجود قيمة مرسلة مع جملة الاستدعاء على النحو التالي:

```
Optional  parametername  As  datatype = defaultvalue
```

2.6) الدوال Function:

هي عبارة عن برامج فرعية مكونة من جملة أو مجموعة جمل برمجية لها بداية ولها نهاية بحيث تكتب بين جملتي (Function ،End Function) يتم استدعائها لتنفذ وظيفة معينة وبعد انتهاء تنفيذها تقوم بإرجاع قيمة للبرنامج المنادي (السطر الذي تم عنده الاستدعاء) (Calling code) من نفس نوع الدالة، وكما هو الحال في الإجراءات فإن الدوال يمكن أن تنقسم إلى قسمين:

اولاً: دوال لا تحتوي على معاملات Non Parameterized Functions:

وهي الدوال التي لا تحتاج إلى إرسال قيم (معاملات) أثناء الاستدعاء والشكل العام لها على النحو التالي:

```
Function Function_Name () As Type
  <Function Body>
  Return value or variable
End Function
```

مثال: اكتب برنامج يقوم بجمع العددين (10،15) باستخدام دالة (لا تحتوي على معاملات).

```
Module Module1
   Function add() As Integer
        Dim x As Integer = 10
        Dim y As Integer = 15
        Dim z As Integer = x + y
        Return z
   End Function
  Sub Main()
        Dim R As Integer
        R = add()
        Console.WriteLine("R=" & R)
        Console.ReadLine()
  End Sub
End Module
```

دالة الجمع

الدالة الرئيسية

ثانياً: دوال تحتوي على معاملات Parameterized Functions:

وهي الدوال التي تحتاج إلى إرسال (تمرير) قيم (معاملات) أثناء استدعائها والشكل العام لها على النحو التالي:

```
Function Function_Name (Parameter_list) As Type
    <Function Body>
     Return value or varible
End Function
```

مثال: اكتب برنامج يقوم بجمع العددين (10،15) باستخدام دالة بحيث يتم إرسال العددين أثناء الاستدعاء.

```
Module Module1
Sub Main()
        Dim R As Integer
        R = add(10, 15)
        Console.WriteLine("R=" & R)
        Console.ReadLine()
End Sub

Function add(x As Integer, y As Integer) As Integer
        Dim z As Integer = x + y
        Return z
End Function

End Module
```

مثال: اكتب البرنامج السابق بحيث يتم إدخال قيم الاعداد عن طريق المستخدم ويتم تمريرها إلى الدالة.

```
Module Module1
    Sub Main()
        Dim a, b, r As Integer
  1:    a = Console.ReadLine
  2:    b = Console.ReadLine
  3:    r = add(a, b)
        Console.WriteLine("R=" & R)
        Console.ReadLine()
    End Sub
    Function add(x As Integer, y As Integer) As Integer
        Dim z As Integer = x + y
        Return z
    End Function
End Module
```

حيث تم في الاسطر رقم "1 و 2" إدخال قيمة a وقيمة b على التوالي، كما تم في السطر رقم "3" استدعاء الدالة "add" وتمرير قيم المتغيرات لها (a,b) والتي تكون مناظرة لمعاملات الدالة (x,y) ليتم حساب المجموع وتخزينه في المتغير المحلى "z" لتعود الدالة بقيمة المتغير "z" من خلال السطر " Return z " ، كما تم في السطر رقم "3" تخزين القيمة المرجعة من الدالة في المتغير "R" ليتم طباعتها في السطر التالي.

ويمكن تغير الشفرة المكتوبة داخل الدالة على النحو التالي لتعطي نفس النتيجة:

```
Function add(x As Integer, y As Integer) As Integer
        Return x + y
End Function
```

3.6) استدعاء البرامج الفرعية (الإجراءات والدوال) داخل البرنامج:

كما هو في الحال في الدوال الجاهزة فإنه يتم استدعاء البرامج الفرعية المعرفة من قبل المبرمج عن طريق كتابة اسمها في البرنامج متبوعا بقوسين يتم بينهما كتابة قيم المعاملات المطلوب تمريرها إلى الدالة أن وجدت والصيغة العامة لاستدعاء البرنامج الفرعي كالآتي:

```
SubPrograme_name(Parameter_list) // في حالة وجود معاملات//
SubPrograme_name() // في حالة عدم وجود معاملات//
```

1.3.6) طرق تمرير المعاملات إلى البرامج الفرعية Passing Parameter:

كما ذكرنا سابقاً أن هنالك بعض البرامج الفرعية (Sub or function) يتم تعريفها لكي تستقبل قيمة أو مجموعة من القيم يتم تمريرها لها أثناء استدعائها من خلال كتابة قيم أو متغيرات في جملة الاستدعاء تكون مناظرة للمتغيرات المعرفة في رأس البرنامج الفرعي، وتسمي المعاملات (المتغيرات أو الثوابت) الموجودة في البرنامج الرئيسي (المكتوبة امام جملة الاستدعاء) بالمعاملات الفعلية بينما تسمى المعاملات (المتغيرات أو الثوابت) المناظرة لها والمعرفة في رأس البرنامج الفرعي (الدالة أو الإجراء) بالمعاملات الصورية أو الشكلية مع ملاحظة أنه يجب أن تتطابق المعاملات الفعلية (المكتوبة في جملة الاستدعاء) والمعاملات الصورية(الموجودة في رأس الدالة أو الإجراء) من حيث أنواعها وعددها وترتيبها، وتوجد هنالك عدة طرق لعملية تمرير المعاملات إلى البرنامج الفرعي اهمها:

الطريقة الأولى: تمرير المعاملات بالقيمة (Pass-By-Value): وفيها يتم إرسال نسخة من المعامل الفعلي إلى المعامل الصوري (الشكلي)، بمعني أن كلا المعاملين (الفعلي و الشكلي) لا يشتركان في موقع واحد في الذاكرة وبالتالي فإن أي تغير في قيمة المعامل الشكلي (داخل البرنامج الفرعي) لا ينتج عنها تغير في المعامل الفعلي (داخل البرنامج الرئيسي) وهذا النوع من تمرير المعاملات يتم تطبيقه من خلال كتابة الكلمة المحجوز"ByVal" امام اسم المعامل أثناء تعريف الإجراء أو الدالة، والمثال التالي يوضح فكرة إرسال المعاملات بالقيمة:

مثال: المثال التالي يقوم باستدعاء برنامج فرعي من نوع إجراء (Procedure) يقوم بحساب مربع عدد "X" يتم إرساله له أثناء الاستدعاء، ومن خلال ناتج تنفيذ البرنامج و من خلال طباعة قيمة المتغير "X" قبل إرساله إلى الإجراء وطباعته بعد إرساله إلى الإجراء نلاحظ أن قيمته لم تتأثر بالعملية التي حدثت داخل الإجراء نظراً لان العملية التي حدثت على المتغير داخل الإجراء تم تخزينها في موقع اخر من الذاكرة بعيد عن المتغير الفعلي.

```
Module Module1
    Sub Main()
        Dim x As Integer = 10
        Console.WriteLine("The value of x before change is " & x)
   change(x)
        Console.WriteLine("The value of x before change is " & x)
        Console.ReadLine()
    End Sub
    Sub change(ByVal y As Integer)
        y = y * y
    End Sub
End Module
```

```
The value of x before change is 10
The value of x after change is 10
```

من خلال المثال السابق نلاحظ أن قيمة المتغير "X" لم تتغير قبل جملة الاستدعاء وبعد جملة الاستدعاء وذلك نظراً لاستخدام اسلوب التمرير بالقيمة.

الطريقة الثانية: تمرير المعاملات بالعنوان (تمرر المعاملات بالمرجع، تمرير المعاملات بالإشارة) (Pass-By-Reference): وفيها يتم إرسال عنوان المعامل الفعلي (الموجود في جملة الاستدعاء) في الذاكرة إلى المعامل الصوري المناظر له في البرنامج الفرعي وبالتالي فإنه وفي هذه الحالة فإن المعامل الفعلي والمعامل الشكلي يشيران إلى نفس موقع الذاكرة (يشتركان في موقع الذاكرة) وينتج عن ذلك أن أي تغير يحدث في قيمة المعامل الشكلي فإنه سيتم تطبيق هذا التغير على المعامل الفعلي في الدالة الرئيسية وهذا النوع من تمرير المعاملات يتم تطبيقه من خلال كتابة الكلمة المحجوز "ByRef" امام اسم المعامل أثناء تعريف البرنامج الفرعي والمثال التالي يوضح فكرة تمرير المعاملات بالمرجع:

مثال: المثال التالي يقوم باستدعاء برنامج فرعي من نوع إجراء (Procedure) يقوم بحساب مربع عدد "x" يتم إرساله له أثناء الاستدعاء، ومن خلال ناتج تنفيذ البرنامج و من خلال طباعة

قيمة المتغير "x" قبل إرساله إلى الإجراء وطباعته بعد إرساله إلى الإجراء نلاحظ أن قيمته تتأثر بالعملية التي حدثت داخل الإجراء نظراً لأن كلا المتغيرين (الفعلي والشكلي) يشيران إلى موقع واحد في الذاكرة.

```
Module Module1
    Sub Main()
        Dim x As Integer = 10
       Console.WriteLine("The value of x before change is " & x)
      change(x)
        Console.WriteLine("The value of x before change is " & x)
        Console.ReadLine()
    End Sub
    Sub change(ByRef y As Integer)
        y = y * y
    End Sub
End Module
```

```
The value of x before change is 10
The value of x after change is 100
```

من خلال المثال السابق نلاحظ أن قيمة المتغير "X" تغيرات بعد جملة الاستدعاء عنها قبل جملة الاستدعاء وذلك نظراً لاستخدام اسلوب التمرير بالمرجع.

ملاحظة: في حالة لم يتم تحديد طريقة التمرير هل هي بالقيمة ام بالمرجع فإن البرنامج افتراضياً سيعتبر عملية التمرير بالقيمة.

2.3.6) البارامترات (المعاملات) الاختيارية Optional parmeters:

توفر لنا لغة الفيجوال بيسك إمكانية جعل عملية تمرير المعاملات إلى البرامج الفرعية (الدوال و الإجراءات) عملية اختيارية من خلال تعريف المعاملات على أساس أنها معاملات اختيارية باستخدام الكلمة المحجوزة "Optional" مع إعطاء هذا المتغير (المعامل) قيمة افتراضية أثناء التعريف، حيث يقوم البرنامج الفرعي باستقبال قيمة المتغير الاختياري من سطر الاستدعاء في حالة تمت كتابتها، اما في حالة عدم كتابة قيمة للمتغير الاختياري في سطر الاستدعاء فإن البرنامج يقوم بتفعيل القيمة الافتراضية المعرفة مسبقاً امام المتغير الاختياري والشكل العام لتعريف برنامج فرعي بمعاملات افتراضية كالتالي:

```
Sub sub_name(Optional parameter1 As datatype = defaultvalue)
```

مثال: في المثال التالي تم تعريف إجراء يحتوي على المعامل "Y" على أساس أنه معامل اختياري وتم إعطاءه قيمة افتراضية "5" في حالة لم يتم تمرير قيمة لهذا الإجراء مناظرة للمعامل "Y" فإنه سيتم تمرير القيمة الافتراضية تلقائيا.

```
Module Module1
    Sub Main()
        change()
        Console.ReadLine()
    End Sub
    Sub change(Optional ByVal y As Integer = 5)
        y = y * y
        Console.WriteLine("The value of x before change is " & y)
    End Sub
End Module
```

ناتج التنفيذ في حالة عدم تمرير قيمة:

```
The value of y= 25
```

اما في حالة تمرير القيمة 4 مثلا للإجراء فسيكون ناتج التنفيذ كالتالي:

```
The value of y= 16
```

---

1. اكتب برنامج يقوم باستدعاء برنامج فرعي من نوع إجراء (sub) يقوم بحساب وطباعة مساحة مثلث من خلال المعادلة (مساحة المثلث = نصف القاعدة* الارتفاع) علما بأن قاعدة المثلث تساوي "5.2" متر وارتفاعه "6.5" متر.

```
Module Module1
    Sub Main()
        Dim b As Single = 5.2
        Dim h As Single = 6.5
        triangle_area(b, h)
        Console.ReadLine()
    End Sub
    Sub triangle_area(ByVal b As Single, ByVal h As Single)
        Dim area As Single = (0.5 * b) * h
        Console.WriteLine("The value of area = " & area)
    End Sub
End Module
```

2. اكتب برنامج يقوم بقراءة نصف قطر دائرة (R) ثم يقوم باستدعاء برنامج فرعي من نوع دالة يقوم بحساب مساحه الدائرة (Area) على أن تتم الطباعة في الدالة الرئيسية (main).

```
Module Module1

    Sub Main()
        Dim R, area As Decimal
        Const pi = 3.14
        Console.Write("Enter R=")
        R = Console.ReadLine()
        area = cir_area(R, pi)
        Console.WriteLine("Area=" & area)
        Console.ReadLine()
    End Sub
    Function cir_area(Rad As Decimal, pi As Decimal) As
Decimal
        Return pi * (Rad ^ 2)
    End Function
End Module
```

3. اكتب برنامج يقوم بقراءة عدد صحيح (n) ثم يقوم باستدعاء إجراء يقوم بطباعة الأعداد من 1 إلى العدد المدخل من قبل المستخدم (n) على النحو التالي:

```
1
1 2
1 2 3
. . . n
```

```
Module Module1
    Sub Main()
        Dim n As Integer
        Console.Write("Enter n=")
        n = Console.ReadLine()
        num(n)
        Console.ReadLine()
    End Sub
    Sub num(ByVal x As Integer)
        For i = 1 To x
            For j = 1 To i
                Console.Write(" " & j)
            Next
            Console.WriteLine()
        Next
    End Sub
End Module
```

4. اكتب برنامج يقوم بقراءة "10" اعدد ثم يقوم باستدعاء داله تقوم بحساب مجموع القيم لها على أن تتم الطباعة في البرنامج الرئيسي.

```
Module Module1
    Dim funsum As Integer = 0
    Sub Main()
        Dim x, sum As Integer
        sum = 0
        For i = 1 To 10
            x = Console.ReadLine
            sum = sumation(x)
        Next

        Console.WriteLine("sum=" & sum)
        Console.ReadLine()
    End Sub
    Function sumation(num As Integer) As Integer

        funsum = funsum + num
        Return funsum
    End Function
End Module
```

5. اكتب برنامج يقوم بتعريف دالة تقوم بقراءة بيانات موظف في شركة ما (الاسم، الحالة الاجتماعية (0 أو 1)، الراتب الأساسي) وكذلك دالة أخرى تقوم بحساب صافي المرتب بحيث يتم إضافة نسبة 10 % على الراتب الأساسي إذا كان الموظف متزوج (الحالة الاجتماعية =1) و 5 % إذا كان غير متزوج (الحالة الاجتماعية =0) ثم يقوم بطباعة اسم الموظف و الراتب الأساسي وصافي المرتب عن طريق إجراء.

```
Module Module1
    Sub Main()
        Dim net_in, s As Decimal
        Dim stat As Integer
        Dim name As String
        Console.WriteLine("Enter name")
        name = Console.ReadLine
        Console.WriteLine("Enter Salary")
        s = Console.ReadLine
        Console.WriteLine("Enter social state")
        stat = Console.ReadLine
        net_in = netmony(s, stat)
        print(name, s, net_in)
        Console.ReadLine()
    End Sub
    Function netmony(Sal As Decimal, st As Integer) As Decimal
        Dim com, net As Decimal
        com = 0
        net = 0
```

```
        If st = 1 Then
            com = Sal * 0.1
        ElseIf st = 0 Then
            com = Sal * 0.05
        End If
        net = Sal + com
        Return net
    End Function
  Sub print(nam As String, sal As Decimal, net_sal As Decimal)
        Console.WriteLine("Emp name:" & nam)
        Console.WriteLine("Emp Salary=" & sal)
        Console.WriteLine("net salary=" & net_sal)
  End Sub
End Module
```

6. اكتب برنامج يقوم باستقبال قيمة عائد مشروع تجاري وكذلك قيمة المصروفات ثم يقوم باستدعاء دالة تقوم بتقسيم الارباح على شركاء المشروع الثلاثة بنسبة 1:2:3.

```
Module Module1
    Sub Main()
        Console.WriteLine("Enter  total income")
        Dim income As Decimal = Console.ReadLine
        Console.WriteLine("Enter  total Masrofat")
        Dim M As Integer = Console.ReadLine
        Dim net_In As Decimal = income - M
        Console.WriteLine("total income = " & income)
        Console.WriteLine("net_income =  " & net_In)
        Console.WriteLine("total Masrofat=" & M)
        ok(net_In)
        Console.ReadLine()
    End Sub
    Sub ok(net_income As Decimal)
        Dim n1, n2, n3 As Integer
        n1 = net_income * 1 / 6
        n2 = net_income * 2 / 6
        n3 = net_income * 3 / 6
        Console.WriteLine("first = " & n1)
        Console.WriteLine("Second = " & n2)
        Console.WriteLine("third = " & n3)
    End Sub
End Module
```

7. اكتب برنامج يقوم بقراءة اسم الموظف وراتبه الأساسي ثم يقوم باستدعاء دالة تقوم بحساب صافي المرتب حيث أن قيمة الخصم تبلغ 5% من إجمالي الراتب الأساسي وفي حالة تجاوز

الراتب قيمة "1000" دينار يقوم باستدعاء دالة أخرى تقوم بحساب صافي المرتب حيث أن قيمة الخصم تبلغ 10%، ثم يقوم استدعاء دالة تقوم بطباعة الاسم وصافي المرتب.

```
Module Module1
    Sub Main()
        Dim name As String
        Dim sal As Integer
        Dim net As Decimal
        name = Console.ReadLine
        sal = Console.ReadLine
        If (sal <= 1000) Then
            net = tax5(sal)
            Print(name, net)
        Else
            net = tax10(sal)
            Print(name, net)
        End If
        Console.ReadLine()
    End Sub

    Function tax5(num As Integer) As Decimal
        Dim tax, net_sal As Decimal
        tax = num * 0.05
        net_sal = num - tax
        Return net_sal
    End Function
    Function tax10(num As Integer) As Decimal
        Dim tax, net_sal As Decimal
        tax = num * 0.1
        net_sal = num - tax
        Return net_sal
    End Function

    Sub print(nam As String, net As Decimal)
        Console.WriteLine("name: " & nam)
        Console.WriteLine("net= " & net)
    End Sub
End Module
```

4.6) البرامج الفرعية والمصفوفات:

عندما درسنا موضوع المصفوفات عرفنا أن المصفوفات تستخدم للتقليل من عدد المتغيرات في البرنامج من خلال تخزين عدد من القيم المتشابه (من نوع واحد) تحت اسم واحد وبالتالي توفر إمكانية إدخال عدد كبير من القيم دون الحاجة لتعريف عدد مماثل لها من المتغيرات، وكما هو الحال عند استخدام البرامج الفرعية (الإجراءات / الدوال) وكما نلاحظ في كافة الأمثلة السابقة أن عدد المعاملات التي يتم تمريرها إلى البرنامج الفرعي محدود بعدد معين من البارامترات وفي حالة الرغبة في إرسال (تمرير) عدد كبير من القيم فانه يجب استخدام (تعريف) عدد مماثل (مساوي) له من المعاملات وهذا غير مجدي في حالة الرغبة في إرسال عدد كبير من القيم، ومن هنا جاءت فكرة استخدام المصفوفات مع البرامج الفرعية وذلك للتقليل من عدد المتغيرات المعرفة كبارامترات (معاملات) في البرنامج الفرعي من خلال تعريف برنامج فرعي يستقبل عدد من القيم المجمعة تحت اسم واحد متمثل في مصفوفة من البارامترات.

1.4.6) البرامج الفرعية والمصفوفات ذات البعد الواحد:

لا يوجد هنالك فرق كبير بين تعريف برنامج فرعي يحتوي على مجموعة معاملات وبين برنامج فرعي يحتوي على مصفوفة من المعاملات حيث أن الفرق الوحيد يكمن في تغير قائمة المعاملات بمصفوفة من المعاملات أثناء التعريف.

الشكل العام لتعريف إجراء يقوم باستقبال مصفوفة ذات بعد واحد:

```
Access_Modifier  Sub subname (ParamArray Array_name() as data_type)
 statement1
 statement1
End Sub
```

الشكل العام لتعريف دالة تقوم باستقبال مصفوفة وتقوم بإرجاع قيمة واحدة:

```
Function function_Name (ParamArray array_name() As data_type) As Type


        Return value or varible
End Function
```

الشكل العام لتعريف دالة تقوم باستقبال مصفوفة وتقوم بإرجاع مصفوفة:

```
Function function_Name (ParamArray array_name() As data_type) As Type

        Return array_name
End Function
```

مثال 1: اكتب شفرة الإعلان عن إجراء يقوم باستقبال مصفوفة مكونه من 5 قيم ثم يقوم بطباعة هذه القيم.

```
Public Sub Print(ParamArray x() As Integer)

        For i = 0 To 4
            Console.Write(x(i) & " ")
        Next
End Sub
```

مثال 2: اكتب شفرة الإعلان عن دالة تقوم باستقبال مصفوفة مكونه من 5 قيم ثم تقوم بإرجاع مجموع عناصر هذه المصفوفة.

```
Function summation(ParamArray s() As Integer) As Integer()
        Dim sum As Integer
        For i = 0 To 4
            sum = sum + s(i)
        Next
        Return s
End Function
```

الشكل العام لاستدعاء البرنامج الفرعي وإرسال مصفوفة من البارامترات له:

استدعاء الإجراء (Calling Sub):

```
Sub_name(array_name)
```

استدعاء الدالة (Calling Function):

استدعاء دالة تقوم باستقبال مصفوفة وتعيد قيمة واحدة:

```
Variable_name= method_name(array_name)
```

استدعاء دالة تقوم باستقبال مصفوفة وتعيد مصفوفة:

```
Array1_name= method_name(array2_name)
```

---

1. اكتب برنامج يقوم بتخزين قيم لعناصر مصفوفة ثم يقوم باستدعاء إجراء يقوم بطباعة هذه القيم وكذلك يقوم باستدعاء دالة تقوم بحساب مجموع هذه القيم على أن تتم طباعة مجموع القيم في البرنامج الرئيسي.

```
Module Module1
    Sub Main()
        Dim arr(4) As Integer
        Dim summ As Integer
        arr(0) = 10
        arr(1) = 12
        arr(2) = 8
        arr(3) = 11
        arr(4) = 95
        print(arr)
        Console.WriteLine()
        summ = summmation(arr)
        Console.WriteLine("summ=" & summ)
        Console.ReadKey()
    End Sub
    Public Sub print(array() As Integer)
        For i = 0 To 4
            Console.Write(array(i) & " ")
        Next i
    End Sub
    Function summmation(array() As Integer) As Integer
        Dim sum As Integer
        For i = 0 To 4
            sum = sum + array(i)
        Next i
        Return sum
    End Function
End Module
```

2. اكتب برنامج يقوم باستدعاء دالة تقوم بقراءة عناصر مصفوفة مكونة من 5 عناصر من النوع الصحيح ثم يقوم باستدعاء دالة أخرى تقوم بحساب المتوسط الحسابي لمجموع عناصر المصفوفة كما يستدعي إجراء يقوم طباعة عناصر المصفوفة.

```
1. Module Module1
2. Sub Main()                                              الدالة الرئيسية
3.   Dim arr(4) As Integer
4.   Dim averg As Decimal
5.   arr = readd()
6.   averg = average(arr)
7.   print(arr)
8.   Console.WriteLine()
9.   Console.Write("average=" & averg)
10. Console.ReadKey()
11. End Sub
12. Public Sub print(array() As Integer)                   اجراء طباعة المصفوفة
13.  For i = 0 To 4
14.  Console.Write(array(i) & " ")
15.  Next i
16. End Sub
17. Public Function average(arr() As Integer) As Decimal   دالة لإيجاد متوسط
18.  Dim sum As Integer                                    عناصر المصفوفة
19.  Dim avg As Decimal
20.  For i = 0 To 4
21.   sum = sum + arr(i)
22.  Next
23.  avg = sum / 5
24.  Return avg
25. End Function
26. Function readd() As Integer()                          دالة قراءة
27.  Dim array(4) As Integer                               المصفوفة
28.  For i = 0 To 4
29.  Console.Write(" array element " & i & "=")
30.  array(i) = Console.ReadLine
31.  Next i
32. Return Array
33. End Function
34. End Module
```

في المثال السابق احتجنا بالإضافة إلى الدالة الرئيسية إلى تعريف ثلاثـة برامج فرعية تتمثل في (دالة لقراءة عناصر المصفوفة، دالة لحساب المتوسط الحسابي لعناصر المصفوفة، إجراء لطباعة عناصر المصفوفة).

تسلسل تنفيذ البرنامج السابق:

يبدأ تنفيذ البرنامج بالسطر رقم 3 والذي تم فيه حجز مواقع العناصر لمصفوفة "arr".

السطر رقم 4 تم فيه حجز موقع لتخزين متوسط مجموع عناصر المصفوفة.

السطر رقم 5 تم فيه استدعاء دالة القراءة لينتقل التنفيذ إلى دالة القراءة في السطر رقم 26.

السطر رقم 26 وتم فيه الإعلان عن دالة تحت اسم read لا تستقبل أي معاملات وترجع مصفوفة من القيم حيث يبدا جسم الدالة من السطر رقم 27.

السطر رقم 27 وتم فيه الإعلان عن مصفوفة تحت اسم "array" ليتم فيها تخزين القيم داخل الدالة قبل ارسالها إلى الدالة الرئيسية.

الاسطر من 28 إلى 31 تم فيها استخدام حلقة التكرار "for" في إدخال عناصر إلى المصفوفة "array".

السطر رقم 32 ثم فيه ارجاع القيم التي تم تخزينها في المصفوفة "array" إلى البرنامج الرئيسي (نقطة الاستدعاء) ليتم تخزينها في المصفوفة "arr" في البرنامج الرئيسي من خلال جملة التخصيص في السطر رقم 5 لينتقل التنفيذ إلى السطر رقم 6.

السطر رقم 6 تم فيه استدعاء الدالة الخاصة بحساب المتوسط الحسابي لعناصر المصفوفة كما تم فيه إرسال القيم المخزنة في المصفوفة "arr" إلى الدالة لحساب متوسط هذه القيم لينتقل التنفيذ بعدها إلى السطر رقم 17.

السطر رقم 17 وتم فيه الإعلان عن دالة تحت اسم "average" تستقبل مصفوفة من المعاملات (القيم في المصفوفة arr) وترجع قيمة واحدة من النوع الكسري والتي تمثل المتوسط الحسابي للمصفوفة حيث يبدا جسم الدالة من السطر رقم 18.

السطر 18 و 19 تم فيه الإعلان عن عدد اثنان من المتغيرات المحلية (خاصة بالدالة) متغير لتخزين مجموع عناصر المصفوفة (sum) ومتغير لتخزين متوسط مجموع العناصر (avg).

الاسطر من 20 الى 22 تم فيها استخدام حلقة التكرار "For" في حساب مجموع عناصر المصفوفة.

السطر 23 تم فيه حساب المتوسط الحسابي لمجموع العناصر وتخزين النتيجة في المتغير "avg".

السطر رقم 24 ثم فيه ارجاع القيمة التي تم تخزينها في المتغير "avg" إلى البرنامج الرئيسي (نقطة الاستدعاء) ليتم تخزينها في المتغير "Average" في البرنامج الرئيسي من خلال جملة التخصيص في السطر رقم 6 لينتقل التنفيذ إلى السطر رقم 7.

السطر رقم 7 تم فيه استدعاء الإجراء الخاص بطباعة عناصر المصفوفة، كما تم فيه إرسال القيم المخزنة في المصفوفة "arr" إلى الإجراء الخاص بالطباعة لينتقل التنفيذ بعدها إلى السطر رقم 12.

السطر رقم 12 وتم فيه الإعلان عن إجراء تحت اسم "print" والذي يستقبل مصفوفة من القيم (القيم في المصفوفة arr) حيث يبدا جسم الإجراء من السطر رقم 12.

السطر رقم 13،14، 15 تم فيه استخدام حلقة "for" لطباعة كافة عناصر المصفوفة.

---

3. اكتب برنامج يقوم باستقبال مصفوفة احادية البعد مكونة من 10 عناصر من النوع الصحيح ثم يقوم باستدعاء إجراء لطباعة المصفوفة وكذلك استدعاء دالة تقوم بحساب اكبر واخرى لحساب اصغر قيمة كما يقوم باستدعاء دالة تقوم بطباعة اصغر واكبر قيمة.

```
Module Module1
    Sub Main()    ' بداية الإجراء الرئيسي
        Dim x(9) As Integer
        Dim mx, mn As Integer
        For i = 0 To 9
            Console.WriteLine("Enter Element " & i & ":")
            x(i) = Console.ReadLine
        Next
        printArray(x)     ' استدعاءاجراء طباعة المصفوفة
        mx = maximum(x)   ' استدعاء دالة ايجاد اكبر قيمة
        mn = minmum(x)    ' استدعاء دالة ايجاد اصغر قيمة
        print(mx, mn)     ' استدعاء إجراء طباعة اكبر واصغر قيمة
        Console.ReadLine()
    End Sub

    Public Function maximum(a() As Integer) As Integer
        Dim max As Integer
        max = a(0)
        For i = 0 To 9
            If a(i) > max Then
```

```
                max = a(i)
            End If
        Next
        Return max
    End Function
    Public Function minmum(a() As Integer) As Integer
        Dim min As Integer
        min = a(0)
        For i = 0 To 9
            If a(i) < min Then
                min = a(i)
            End If
        Next
        Return min
    End Function
    Public Sub print(x As Integer, y As Integer)
        Console.WriteLine("--------------------")
        Console.WriteLine("The max Element=" & x)
        Console.WriteLine("The min element=" & y)
    End Sub
    Public Sub printArray(arr() As Integer)
        Console.WriteLine("--------------------")
        For i = 0 To arr.Length - 1
            Console.Write(arr(i))
        Next
        Console.WriteLine()
    End Sub

End Module
```

2.4.6) البرامج الفرعية والمصفوفات ذات البعدين:

يمكن للبرنامج الفرعي (دالة /إجراء) استقبال مصفوفة ذات بعدين، كما يمكن أن يرجع مجموعة من القيم متمثلة في مصفوفة ذات بعدين.

الشكل العام لتعريف إجراء يستقبل مصفوفة ذات بعدين من البارامترات:

```
Access_Modifier  Sub subname (Array_name(,) as data_type)
      Statements1
      Statements2
End Sub
```

الشكل العام لتعريف دالة تستقبل مصفوفة ذات البعدين وتقوم بإرجاع قيمة واحدة:

```
Function function_Name (array_name(,) As data_type) As Type

     <Function Body>

        Return value or varible
End Function
```

الشكل العام لتعريف دالة تستقبل مصفوفة ذات بعدين وتقوم بإرجاع مصفوفة ذات بعدين:

```
Function function_Name (array_name(,)As data_type) As Type(,)

<Function Body>

        Return array_name
End Function
```

الشكل العام لاستدعاء البرنامج الفرعي وارسال مصفوفة ثنائية من البارامترات له:

لا يوجد هنالك أي اختلاف بين استدعاء برنامج فرعي يستقبل مصفوفة احادية من البارامترات أو مصفوفة ثنائية.

استدعاء الإجراء:

```
Sub_name(array_name)
```

استدعاء الدالة:

استدعاء دالة تقوم باستقبال مصفوفة وتعيد قيمة واحدة

```
variable_name= method_name(array_name)
```

استدعاء دالة تقوم باستقبال مصفوفة وتعيد مصفوفة:

```
Array1_name= method_name(array2_name)
```

1. اكتب برنامج يقوم بإدخال مصفوفة ثنائية البعد (3*3) ثم يقوم باستدعاء برنامج فرعي من نوع إجراء يقوم بطباعة هذه المصفوفة.

```
Module Module1
    Sub Main()
        Dim x(2, 2) As Integer
        For i = 0 To 2
            For j = 0 To 2
                Console.WriteLine("Enter Element (" & i & "," & j & "):")
                x(i, j) = Console.ReadLine
            Next j
        Next i
        printArray(x)

        Console.ReadLine()
    End Sub
    Public Sub printArray(arr(,) As Integer)
        Console.WriteLine("--------------------")
        For i = 0 To 2
            For j = 0 To 2
                Console.Write(arr(i, j) & " ")

            Next j
            Console.WriteLine()
        Next i
    End Sub
End Module
```

2. اكتب برنامج يقوم باستقبال مصفوفة ذات بعدين (3*4) ثم يقوم باستدعاء دالة تقوم بحساب أكبر قيمة في المصفوفة وكذلك دالة لحساب اصغر قيمة.

```
Module Module1
    Sub Main()
        Dim array(2, 3) As Integer
        For i = 0 To 2
            For j = 0 To 3
                Console.WriteLine("Enter Element (" & i & "," & j & "):")
                array(i, j) = Console.ReadLine
            Next j
        Next i
        Console.WriteLine("--------------------------")
        Console.WriteLine("Max=" & maximum(array))
        Console.WriteLine("min=" & minmum(array))
```

```
        Console.ReadKey()
    End Sub
    Public Function maximum(arr(,) As Integer) As Integer
        Dim max As Integer
        max = arr(0, 0)
        For i = 0 To 2
            For j = 0 To 3
                If arr(i, j) > max Then
                    max = arr(i, j)
                End If
            Next j
        Next i
        Return max
    End Function
    Public Function minmum(arr(,) As Integer) As Integer
        Dim min As Integer
        min = arr(0, 0)
        For i = 0 To 2
            For j = 0 To 3
                If arr(i, j) < min Then
                    min = arr(i, j)
                End If
            Next j
        Next i
        Return min
    End Function
End Module
```

3. اكتب برنامج يقوم باستدعاء برنامج فرعي يقوم بقراءة مصفوفة ذات بعدين ثم يقوم بطباعة عناصر المصفوفة في البرنامج الرئيسي.

```
Module Module1
    Sub Main()
        Console.WriteLine("--------------------")
        Dim x(2, 2) As Integer
        x = read()
        For i = 0 To 2
            For j = 0 To 2
                Console.Write(x(i, j) & " ")
            Next j
            Console.WriteLine()
        Next i
```

```
        Console.ReadLine()
    End Sub
    Public Function read() As Integer(,)
        Dim array(2, 2) As Integer
        For i = 0 To 2
            For j = 0 To 2
                Console.WriteLine("Enter Element (" & i & "," & j & "):")
                array(i, j) = Console.ReadLine
            Next j
        Next i
        Return array
    End Function
End Module
```

5.6) التحميل الزائد للدوال والإجراءات Sub & Function Overloading:

تمكننا لغة الفيجوال بيسك من كتابة أكثر من برنامج فرعي بنفس الاسم في فئة (Class) واحدة أو في موديل (Module) واحد وهذا ما يعرف بالتحميل الزائد للدوال والاجراءات أي أن التحميل الزائد للدوال أو الإجراءات عبارة عن كتابة أكثر من دالة أو أكثر من إجراء تحمل نفس الاسم في فئة واحدة أو في موديل واحد مع اختلاف توقيع كل دالة أو إجراء (signature) لكي يتم التميز بينها حيث يتمثل توقيع الدالة أو الإجراء في قائمة البارامترات الموجودة بين قوسي الدالة أو الإجراء ، ولكي نستطيع تعريف أكثر من دالة بنفس الاسم أو أكثر من إجراء بنفس الاسم داخل فئة واحدة أو داخل موديل واحد فإن هذه الدوال أو الإجراءات المعرفة يجب أن تختلف في احدى ثلاثة اشياء:

1. عدد المعاملات Number of parameters:

يتيح اسلوب التحميل الزائد للمبرمج إمكانية تعريف دالة أو إجراء بنفس الاسم شريطة أن تختلف في عدد المتغيرات المعرفة في رأس الدالة أو الإجراء.

```
Public Sub max(a As Integer, b As Integer)
     <Procedure body>
End Sub

Public Sub max(x As Integer, y As Integer, z As Integer)
    < Procedure body>
End Sub
```

مثال: البرنامج التالي يقوم بتعريف إجراءين بنفس الاسم مع الاختلاف في عدد المعاملات.

```
Module Module1
    Sub Main()
        disp("a")
        disp("a", 10)
        Console.ReadLine()
    End Sub
    Public Sub disp(c As Char)
        Console.WriteLine(c)
    End Sub
    Public Sub disp(c As Char, num As Integer)
        Console.WriteLine(c & " " & num)
    End Sub
End Module
```

2. نوع بيانات المعاملات Data type of parameters:

يتيح اسلوب التحميل الزائد للمبرمج إمكانية تعريف دالة أو إجراء بنفس الاسم وبنفس عدد البارامترات شريطة أن تختلف في نوع المتغيرات المعرفة في رأس الدالة أو الإجراء .

```
Public Function max(arr(,) As Decmial) As Integer
    < function  body>
    Return value
End Function

Public Function max(arr(,) As Integer) As Integer
        < function  body>
        Return value
End Function
```

مثال: تعريف دالتين بنفس الاسم مع الاختلاف في نوع البارامترات.

```
Module Module1
    Sub Main()
        disp("a")
        disp(10)
        Console.ReadLine()
    End Sub
    Public Sub disp(c As String)
        Console.WriteLine(c)
    End Sub
    Public Sub disp(num As Integer)
        Console.WriteLine(num)
End Sub
```

3. ترتيب نوع بيانات المعاملات Sequence of Data type of parameters:

يتيح اسلوب التحميل الزائد للمبرمج إمكانية تعريف دالة أو إجراء بنفس الاسم ونفس عدد ونوع المتغيرات شريطة أن تكون مختلفة في ترتيب نوع المتغيرات المعرفة في رأس الدالة أو الإجراء.

```
Public Sub max(a As Integer, b As Decmial)
     <Procedure body>
End Sub

Public Sub max(x As decmial, y As Integer)
    < Procedure body>
End Sub
```

مثال: تعريف إجرائيين بنفس الاسم مع الاختلاف في ترتيب نوع البارامترات.

```
Module Module1
    Sub Main()
        disp("a", 5, 3.2)
        disp(10, "a", 5.4)
        Console.ReadLine()
    End Sub
    Public Sub disp(c As String, x As Integer, y As Double)
        Console.WriteLine(c)
    End Sub
    Public Sub disp(num As Integer, c As String, y As Double)
        Console.WriteLine(num)
    End Sub
End Module
```
