الفصل السادس

المصفوفات ARRAYS

في هذا الفصل Programming Concepts

6

في هذا الدرس نشرح موضوع المصفوفات Arrays بأنواعها و كيفية تمثيلها و استخدامها و ذلك من خلال النقاط التالية :

- معني المصفوفات و استخدامها .
- أنواع المصفوفات Arrays types.
- مصفوفة بعد واحد One Dimension.
- مصفوفة بعدين Two Dimension .
- خصائص المصفوفات

معني المصفوفة

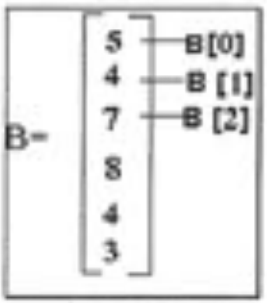
كما تعلم من الرياضيات أن المصفوفة Arrayعبارة عن مجموعة من القيم (العناصر) من نفس النوع و تأخذ أسم و تأخذ عدد للعناصر.

والمطلوب كيفية تمثيلها و استعمالها وكيفية التعامل مع عناصرها و للمصفوفة تطبيقات كثيرة لا تصلح إلا بها مثل عمليات البحث عن قيمة ضمن مجموعة قيم Search ومثل عمليات ترتيب مجموعة من القيم سواء تصاعدى أو تنازلى Sorting ومثل إيجاد أكبر قيمة ضمن مجموعة قيم Max وايجاد أقل قيمة Min والكثير من التطبيقات وخاصة فى مجال الدرسات العليا والأبحاث.

أنواع المصفوفات :

(أ) مصفوفة بعد واحد One Dimension كما بالشكل (6.1)

A= [5 4 7 8 4 3]



الشكل (1-6)

- المصفوفة A مصفوفة صفOne Row وهي ذات بعد واحد حيث لا يوجد صفوف و أعمدة بل هي صف واحد فقط .
- المصفوفة B مصفوفة عمود واحد One Column ، لذلك يسمي كلا منهما مصفوفة البعد الواحد و يتم ترتيب العناصر بأسم المصفوفة و ترتيب العنصر مثل ، B[2] A[1] وهكذا.

(ب) مصفوفة متعددة الأبعاد :

و فيها يوجد أكثر من صف Row وأكثر من عمود Column كما بالشكل (2-6) .

$$C= \begin{bmatrix} 5 & 70 & 10 \\ 20 & 40 & 5 \\ 30 & 40 & 7 \\ 5 & 30 & 5 \\ 9 & 40 & 6 \\ 30 & 10 & 7 \end{bmatrix}$$

الشكل (2-6)

و في هذه المصفوفة يوجد 6 صفوف Rows و 3 أعمدة Columns لذلك تسمي 6 ⊕ 3 ويتم ترتيب العناصر بالصف Row ثم العمود Column كما بالشكل التالي :

- C (0, 0) العنصر الأول الذي يقع في الصف الاول رقم 0 و العمود الاول رقم 0
- C (0, 1) العنصر الثاني الذي يقع في الصف الاول رقم 0 و العمود الثانى رقم 1
- C (0, 2) العنصر الثالث الذي يقع في الصف 0 و العمود 2
- C (0,3) العنصر الرابع الذي يقع في الصف 0 و العمود 3
- C (1,0) العنصر الخامس الذي يقع في الصف 1 و العمود 0
- C (1,1) العنصر السادس الذي يقع في الصف 1 و العمود 1

و هكذا حتى باقي العناصر

مصفوفة البعد الواحد One Dimension :

و يتم التعامل مع هذه المصفوفة بالخطوات التالية :

1- الإعلان عن المصفوفة.

2- استعمال عناصر المصفوفة بملئها بالقيم أو استعمال القيم أو طباعة القيم.

فى لغة VB.NET

الإعلان عن المصفوفة

- يتم الإعلان عن المصفوفة كما يتم الإعلان عن المتغيرات و بنفس درجات الإعلان فيمكن الإعلان عنها و كأنها محلية Local داخل برنامج فرعي Sub أو على مستوي form أو على مستوي التطبيق كما يحدث مع المتغيرات و ذلك حسب المكان و بالصورة التالية :

```
Dim a(10) As Integer
```

أو يتم استعمال الصورة الحديثة التالية:

```
Dim Arr As Integer()=New Integer(10) {}
```

في هذه الصورة استعملت كلمة DIM لإعلان عن المصفوفة و المتغير A هو أسم المصفوفة و العدد 10 هو عدد عناصر المصفوفة و النوع Integer هو نوع عناصر المصفوفة و يمكن هذا النوع أن يتغير ليصبح من أي نوع (STRING,.....) حسب استعمال المصفوفة وفى الصورة الثانية تم الإعلان عن مصفوفة بالاسم Arr من النوع Integer ثم حجز هذه الأماكن فى الذاكرة باستعمال الكلمة New والإشارة إلى هذه الاماكن باسم المصفوفة.
و لتوضيح ذلك تابع المثال التالي :

مثال :

1- قم بانشاء تطبيق جديد ولكن من النوع ConsoleApplication كما سبق.

2- داخل الدالة الرئيسية Sub Main() اكتب السطور التالية:

```
1.  Dim a(10) As Integer
2.  Dim i As Integer
3.  For i = 0 To 10
4.  a(i) = i
5.  Next
6.  For i = 0 To 10
7.  Console.WriteLine(a(i))
8.  Next
```

فى هذه السطور :

- ق السطر رقم 1 تم الإعلان عن مصفوفة بالاسم a عدد عناصرها 10 من النوع Integer وق السطر رقم 2 تم الإعلان عن متغير بالاسم I من النوع Integer.
- ق السطر رقم 3 بداية التكرار For من القيمة 0 إلى القيمة 10.
- ق السطر رقم 4 يتم وضع قيمة المتغير I ق عنصر المصفوفة رقم I.
- فمثلا عندما تكون قيمة المتغير I هى 0 يتم وضع هذه القيمة ق العنصر رقم 0 ق المصفوفة أى a(0) و عندما تكون قيمة المتغير I هى 1 يتم وضع هذه القيمة ق العنصر رقم 1 ق المصفوفة أى a(1) وهكذا حتى تملئ المصفوفة بالقيم من 0 إلى 10.
- ق السطر رقم 6 يبدأ التكرار For مرة أخرى من القيمة 0 إلى القيمة 10.
- ق السطر رقم يتم استعمال الدالة ()Writeln لطباعة قيم المصفوفة a(I) مع تغيير قيمة I بالتكرار For.

تمرين (1) :

صمم برنامج يستقبل من المستخدم مجموعة من الحروف ثم يطبعها له بالعكس أي إذا استقبل الحروف a,b,c,d يطبعها لك d,c,b,a.

تمرين (2) :

صمم برنامج يستقبل مجموعة من الأسماء ثم يضبعها للمستخدم بعكس الترتيب.

فى لغة C#

يتم التعامل مع هذه المصفوفة بالخطوات التالية :

1- الإعلان عن المصفوفة.

2- استعمال عناصر المصفوفة بملئها بالقيم أو استعمال القيم أو طباعة القيم.

الإعلان عن المصفوفة

يتم الإعلان عن المصفوفة كما يتم الإعلان عن المتغيرات و بنفس درجات الإعلان بالصورة التالية :

```
int[] myArray = new int [5];
```

في هذه الصورة المتغير A هو أسم المصفوفة و العدد 10 هو عدد عناصر المصفوفة

والنوع int هو نوع عناصر المصفوفة و يمكن لهذا النوع أن يتغير ليصبح من أي نوع (STRING,.....) حسب استعمال المصفوفة وفى الصورة الثانية تم الاعلان عن مصفوفة بالاسم Arr من النوع Integer ثم حجز هذه الأماكن فى الذاكرة باستعمال الكلمة New والإشارة إلى هذه الأماكن باسم المصفوفة.

ويمكن اعطاء قيم ابتدائية لعناصر المصفوفة كما فى الصور التالية:

```
int[] myArray = new int[] {1, 3, 5, 7, 9};

string[] weekDays = new string[]
                {"Sun","Sat","Mon","Tue","Wed","Thu","Fri"};

int[] myArray = {1, 3, 5, 7, 9};
string[] weekDays =
{"Sun","Sat","Mon","Tue","Wed","Thu","Fri"};

int[] myArray;
myArray = new int[] {1, 3, 5, 7, 9};   // OK
myArray = {1, 3, 5, 7, 9};   // Error
```

و لتوضيح ذلك تابع المثال التالي :

مثال :

1- قم بإنشاء تطبيق جديد ولكن من النوع ConsoleApplication كما سبق

2- داخل الدالة الرئيسية Main() اكتب السطور التالية:

```
public static void Main()
{
// Declare and initialize an array:
string[] WeekDays = new string []
{"Sun","Sat","Mon","Tue","Wed","Thu","Fri"};

// Pass the array as a parameter:
PrintArray(WeekDays);
}
```

فى هذه السطور:

تم الاعلان عن مصفوفة بالاسم WeekDays من النوع string .

قم بإنشاء دالة جديدة بالاسم PrintArray() كما فى السطور التالية :

```
static void PrintArray(string[] w)
{
for (int i = 0 ; i < w.Length ; i++)
Console. WriteLine (w[i] + "{0}", i < w.Length - 1 ? " "
: "");
Console.WriteLine();
}
}
```

في هذه السطور:

تم استعمال التكرار For مؤة أخرى من القيمة 0 إلى عدد عناصر المصفوفة.

تم استعمال الدالة ()Writeln لطباعة قيم المصفوفة a(i) مع تغيير قيمة I بالتكرار For.

وبالتالي عند تشغيل هذا البرنامج تحصل على قيم عناصر المصفوفة مطبوعة.

تمرين (1) :

صمم برنامج يستقبل من المستخدم مجموعة من الحروف ثم يطبعها له بالعكس أي إذا استقبل الحروف a,b,c,d يطبعها لك d,c,b,a.

تمرين (2) :

صمم برنامج يستقبل مجموعة من الأسماء ثم يطبعها للمستخدم بعكس الترتيب.

في لغة Java

لإنشاء مصفوفة نستعمل ثلاثة خطوات هي:

1- الإعلان عن متغير المصفوفة Array Variables .

2- تعريف (إنشاء) عنصر المصفوفة array object .

3- تخزين القيم أو البيانات داخل المصفوفة والتعامل معها .

1. الإعلان عن متغير مصفوفة Array Variable

ويتم ذلك بطريقتين هما:

الطريقة الأولى:

```
String ourteam [];
int Codes [];
```

فى هذه الطريقة يتم كتابة نوع عناصر المصفوفة فى الأول (String , int) ثم كتابة اسم المصفوفة [our team, codes] ثم وضع الأقواس [ ] وبالتالى يتم الإعلان عن مصفوفة بالاسم ourteam من النوع string والذى يفرق بين المصفوفة والمتغير العادى هو الأقواس [ ] التى تدل على أن اسم المتغير هو مصفوفة.

الطريقة الثانية:

```
String [ ] ourteam;
int [ ] codes;
```

فى هذه الطريقة يتم كتابة نوع المصفوفة ثم الأقواس بالشكل [ ] string ثم اسم المصفوفة وهى our team فى السطر الأول.

بالمثل المثال الثانى هو [ ] int ثم اسم المصفوفة وهو Codes.

وفى الحالتين تم الإعلان عن متغير من نوع هذه المصفوفة.

تعريف عنصر المصفوفة Array Object

بعد الإعلان عن متغير مصفوفة يتم تعريف (إنشاء) عنصر المصفوفة ويتم بطريقتين:

- استعمال كلمة new لإنشاء العنصر.
- تعريف عنصر object المصفوفة مع إعطائها قيم ابتدائية بدون استعمال new.

استعمال new

ويتم استعمال new لتعريف object عنصر كما يلى:

```
String jobs= new string [10]
```

وفى هذا السطر يتم تعريف متغير بالاسم jobs من نوع مصفوفات حرفيات [ ] string كما سبق حجز او تعريف مصفوفة عناصر جديدة باستعمال new فى الطرف الثانى من النوع string وعدد العناصر 10 وبالطبع يمكن إجراء هذا السطر على سطرين كما يلى:

```
String [ ] jobs;
jobs = new string [10];
```

وعند استعمال new مع المصفوفة لابد من تحديد عدد العناصر مثل 10.

تعريف عنصر المصفوفة وإعطائها قيم ابتدائية

يمكن تعريف عنصر مصفوفة بدون استعمال new وذلك بإعطاء المصفوفة قيم ابتدائية وبالتالى لا نحتاج لتحديد عدد العناصر لأننا نضع العناصر نفسها كما يلى:

```
String [ ] Jobs = {"Eng", Doc', "Tech");
```

فى هذا السطر تم تعريف مصفوفة حرفيات بالاسم Jobs ثم إعطائها قيم ابتدائية وهى الكلمات ...Eng وفى هذه الحالة لا نحتاج لتحديد عدد العناصر لأن العناصر موجودة بالفعل.

فى حالة استعمال New وعند إعطاء المصفوفة قيم ابتدائية يتم تسجيل قيم افتراضية حسب نوع المصفوفة فمثلاً مصفوفة القيم الرقمية تأخذ العناصر0 ومصفوفة الحرفيات تأخذ 0/ ومصفوفة القيم المنطقية Boolean تأخذ القيمFalse ومصفوفة العناصر Objects تأخذ null.

التعامل مع عناصر المصفوفة Accessing Array Elements

ويتم التعامل مع عناصر المصفوفة بكتابة رقم عنصر المصفوفة بين أقواس المصفوفة [ ]، كما بالشكل.

```
Array Name [ No]
```

وفى هذه الصورة Array Name اسم المصفوفة وno ترتيب العنصر فى المصفوفة مع ملاحظة أن ترتيب عناصر المصفوفة تبدأ من القيمة 0 ، وتقوم لغة Java باختبار قيمة ترتيب العنصر قبل التعامل معه للتأكد من وجود هذا العنصر ضمن عناصر المصفوفة وليس خارجها ولا يمكن تسجيل قيمة فى عنصر غير موجود ضمن عناصر المصفوفة (كما يحدث فى لغة C/C++) فلا يحدث خطأ.

ولتوضيح ذلك تابع المثال التالى:

```
String [ ] My Array = new String [20];
          My Array [20] = "Java lang";
```

فى السطر الأول يتم الإعلان وتعريف مصفوفة بالاسم My Array من نوع حرفيات string وحجز 20 عنصر باستعمال new.

فى السطر الثانى يتم تسجيل العبارة .Java lang فى المصفوفة My Array فى العنصر رقم 20 وهذا السطر عند ترجمة البرنامج يعطى خطأ وذلك لان هذه المصفوفة تبدأ من العنصر رقم 0 وتنتهى عند العنصر 19 وعدد العناصر 20 عنصر لذلك لا يوجد عنصر رقمه 20.

وللتعامل مع عدد عناصر المصفوفة يمكن استعمال الدالة LENGTH كما فى السطر التالى:

```
int L= MyArray. Length;
```

فى هذا السطر يتم حساب عدد عناصر المصفوفة باستعمال الدالة Length مع المصفوفة وحفظ هذه العدد فى المتغير L.

مثال

فى هذا المثال نحاول تجميع النقاط السابقة عن المصفوفة وتوضيحها كما فى السطور التالية:

```
1  public class Myfriends {
2
3      String [ ] Names = { "Khaled", "Alaa", "Amr", "Omr"};
4  void printNames ( )
5  {
6    int I = 1;
7    System.out.println (Names [I]);
8    I++;
9    System.out.println (Names [I]);
10   I++;
11 System.out.println (Names [I]);
12   I++;
13   System.out.println (Names [I]);
14   }
15   public static void main (String args [ ])
16   {
17
18   Myfriends FD= new Myfriends ( );
19   FD. printNames ();
20   }
21   }
```

فى هذه السطور:

- فى السطر رقم 1 يتم تعريف فصيلة البرنامج (Class) بالاسم Myfriends .
- فى السطر رقم 3 يتم تعريف مصفوفة حرفيات (string) بالاسم Names مع إعطائها قيم ابتدائية عبارة عن مجموعة أسماء.

- فى السطر رقم 4 يتم تعريف دالة عضوه دالة فى الفصيلة بالاسم ()printNames.
- فى السطر رقم 5 يتم الإعلان عن متغير i بقية ابتدائية 0.
- فى السطر رقم 6 يتم طباعة عنصر المصفوفة رقم i اى رقم 0 أى الاسم Khaled المخزن فى العنصر.
- فى السطر رقم 7 يتم زيادة قيمة المتغير i بمقدار 1 لتصبح i تساوى 1.
- فى السطر رقم 8 يتم طباعة عنصر المصفوفة رقم i أى رقم 1 أى Alaa وهكذا السطور حتى السطر رقم 12.
- وفى السطر رقم 13 تنتهى الدالة ()Print Names التى تقوم بطباعة الأسماء.
- فى السطر رقم 15 تبدأ الدالة الرئيسية ()main التى يبدأ التنفيذ منها وإذا كان هناك اكثر من فصلة class يتم تنفيذ الفصيلة التى بها الدالة main كما أشرنا من قبل.
- فى السطر رقم 18 يتم تعريف عنصر (object) بالاسم FD فى الفصيلة My friends وذلك لتتمكن ممن التعامل مع أعضاء الفصيلة My friends فى دوال وبيانات.
- فى السطر 19 تم استدعاء الدالة ()PrintNames مع هدف الفصيلة FD وبالتالى تقوم الدالة بطباعة الأسماء المعرفة داخل المصفوفة Names وتحصل على قائمة بالأسماء كما فى نتيجة التنفيذ.

استعمال أوامر التكرار for مع المصفوفة

لاحظنا فى المثال السابق كيف تم التعامل مع عناصر المصفوفة بتحديد ترتيب عنصر المصوفة كما فى السطر.

```
Names[3]
```

أى العنصر رقم 4 ، ولكن أشرنا إلى أنه يصعب أو يستحيل استعمال هذه الطريقة مع المصفوفات ولابد من استعمال جملة لتكرار for بدلا من ذلك.

وبالتالى يتم تعديل البرنامج السابق ليصبح كما فى السطور التالية:

```
1  public class Myfriends {
2
3      String [ ] Names = { "khaled", "Alaa", "Am", "Omr"};
4  void printNames ( )
5   {
6    int i ;
7    for (i=0;i<4;i++)
8    System.out.println (Names [i]);
9      }
10   public static void main (String args [ ])
11   {
12
13   Myfriends FD= new Myfriends ( );
14   FD. printNames ();
15   }
16   }
```

الجديد فى هذه السطور عن برنامج المصفوفات التقليدى هو استعمال التكرار for Loop .

فى السطر رقم 4 للتكرار من 0 حتى Names. Length وهو عدد عناصر المصفوفة:

- فى السطر رقم 5 يتم طباعة عناصر المصفوفة Names[I]> أى العناصر المخزنة وبالتالى يتم طباعة الأسماء، وذلك يوفر إعادة كتابة أمر الطباعة وذلك باستعمال تكراره مع for بعدد عناصر المصفوفة.

وعند تنفيذ البرنامج تحصل على نفس النتيجة السابقة.

كما يتضح لنا أهمية استعمال جملة التكرار for مع المصفوفات .

نسخ المصفوفات

يمكن استعمال متغير مصفوفة للإشارة إلى مصفوفة أخرى كما فى الشكل.

```
double[] a = new double[3];
double[] b = a;
```

فى هذه السطور يتم الإعلان عن مصفوفة a ثم الإعلان عن مصفوفة b واستخدام متغيرها للإشارة إلى نفس المصفوفة a.

طول المصفوفة

```
for (int i = 0; i < a.length; i++)
    {
        b[i] = a[i];
       }
```

فى هذه السطور:

يتم الاستفادة من الخاصية length الموجودة فى المصفوفة لتحديد طولها أى عدد عناصرها بدلا من تحديد هذا الطول برقم ، ثم القيام بنسخ عناصر المصفوفة a إلى المصفوفة b.

ايجاد أكبر قيمة

يمكن استعمال المصفوفات لإيجاد أكبر قيمة من مجموعة قيم وذلك كما فى السطور التالية :

```
double max = A[0];
    for (int i = 1; i < A.length; i++) {
      if (A[i] > max)
        max = A[i];
    }
```

فى هذه السطور:

يتم وضع قيمة أول عنصر من المصفوفة فى المتغير mx بافتراض أنه أكبر عنصر .

ثم استعمال تركيب التكرار for فى المرور على جميع عناصر المصفوفة وفى كل مرة يتم مقارنة قيمة المتغير max بعنصر المصفوفة الحالى .

إذا كانت القيمة الموجودة فى المصفوفة أكبر من قيمة المتغير mx يتم التبديل بحيث ينتهى التكرار وقد تم وضع أكبر عنصر فى المتغير mx.

البحث عن قيمة داخل المصفوفة

من العمليات المشهورة للمصفوفات القيام بالبحث عن قيمة داخل مجموعة قيم.

والمثال التالى يوضح ذلك:

```
static int find(int[] A, int N)
    {

        for (int index = 0; index < A.length; index++)
        {
           if ( A[index] == N )
              return index;  // N has been found at this index!
        }

        return -1;
    }
```

في هذه السطور:

يتم إنشاء دالة تستقبل اسم المصفوفة والقيمة المطلوب البحث عنها ثم يتم استعمال تركيب التكرار for للمرور على العناصر ومقارنتها بالقيمة حتى نجدها.

فاذا وجدت يتم اعادة ترتيب هذه القيمة في المصفوفة وإلا يتم إعادة القيمة -1.

ترتيب عناصر المصفوفة

من العمليات المشهورة أيضا استعمال المصفوفات في ترتيب مجموعة قيم ولتوضيح ذلك تابع السطور التالية :

```
static void selectionSort(int[] A)
{
    for (int lastPlace = A.length-1; lastPlace > 0; lastPlace--)
    {
 position lastPlace.
        int maxLoc = 0;  // Location of largest item seen so far.
        for (int j = 1; j <= lastPlace; j++)
      {
        if (A[j] > A[maxLoc])
        {
             maxLoc = j;
        }
      }
```

```
        int temp = A[maxLoc];  // Swap largest item with A[lastPlace].
        A[maxLoc] = A[lastPlace];
        A[lastPlace] = temp;
    } // end of for loop

}
```

المصفوفة ذات البعدين :

فى VB.NET

يتم الإعلان عنها بالشكل التالي :

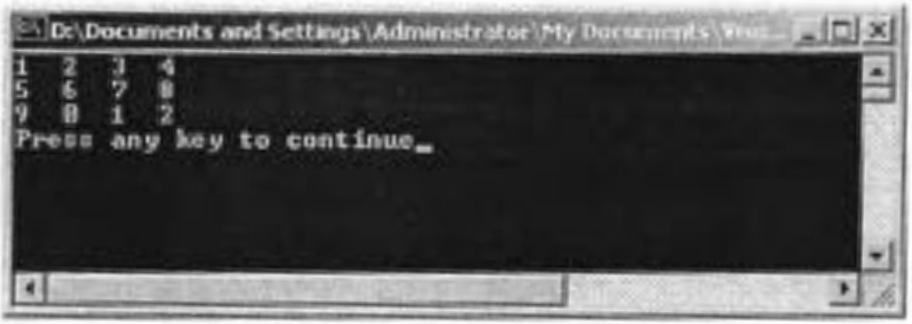
```
Dim RC(3, 4) As Integer
```

و فيها يتم الإعلان عن مصفوفة بالاسم RC بعدد صفوف 3 و عدد أعمدة 4 من النوع INTEGER أي الأرقام الصحيحة .

و بنفس القاعدة يمكن تغيير عدد العناصر حسب الطلب و كذلك تغيير نوع المصفوفة و لتوضيح ذلك تابع المثال التالي :

مثال :

عند تنفيذ هذا المثال تحصل علي نتيجة التنفيذ عبارة عن برنامج يطلب منك إدخال قيم عناصر مصفوفة 3*4 بعد إدخال العناصر يقوم البرنامج بطباعة العناصر بشكل مصفوفة كما بالشكل (2-6) .



الشكل (2-6)

جرب استعمال البرنامج ثم تابع معنا خطوات التصميم .

خطوات التصميم :

إبدأ برنامج جديد من النوع ConsoleApplication كما سبق .

- اكتب السطور التالية داخل الدالة الرئيسية (Sub Main():

```
Dim c, r
   Dim AR(3, 4) As Integer
   For r = 1 To 3
      For c = 1 To 4
         AR(r, c) = InputBox("ENTER NO:")
      Next c
   Next r
   For r = 1 To 3

      For c = 1 To 4
         Console.Write(AR(r, c))
         Console.Write("  ")
      Next c
      Console.WriteLine()

   Next r
```

في هذه السطور:

- يتم استعمال جملتي تكرار FOR لاستقبال عناصر المصفوفة وكذلك لطباعة صفوف وأعمدة المصفوفة كما يلي:

- في السطرين 1 يتم الاعلان عن مصفوفة بالاسم AR عدد عناصر 3*4 من النوع Integer.
- في السطر رقم 2 تبدأ التكرار الخارجي بلكمة For بالمتغير R.
- في السطر رقم 3 تبدأ التكرار الداخلي بالمتغير 2.
- في السطر رقم 4 يتم استقبال قيم المصفوفة من المستخدم باستعمال INPUTBOX مع وضع القيمة في عناصر المصفوفة AR (R,C) حسب قيمة R,C

الحالية. وبهذه يستمر استقبالنا عناصر المصفوفة حتى ينتهي التكرار الداخلي C والتكرار الخارجي R.

- من السطر 7 حتى السطر 12 يتم طباعة عناصر المصفوفة بشكل مصفوفة.
- وعند تنفيذ هذا البرنامج تحصل نتيجة التنفيذ كما في الشكل (3-6)

الشكل (3-6)

ملحوظة هامة:

يتصاحب استعمال المصفوفات Arrays باستعمال التكرار For لأنه بدون استعمال التكرار For يتم التعامل مع عناصر المصفوفة عنصر بعنصر ولذلك عند التعامل مع مصفوفة متعددة الابعاد يتم استعمال أكثر من For.

خصائص المصفوفات

بالاضافة لما شرحناها من طرق التعامل مع المصفوفات توجد بعض الخصائص للمصفوفات تفيد عند التعامل معها منها ما يلي:

الدالة GetUpperBound()

تقوم هذه الدالة بحساب ترتيب العنصر الأخير في المصفوفة ولتوضيح ذلك تابع المثال التالي:

مثال :

1- أبدأ برنامج جديد من النوع WindowsApplication كما سبق .

2- قم باضافة زر أمر Button وبداخل الحدث Button1_Click اكتب السطور التالية:

```
1:    Dim output As String
2:    Dim i As Integer
3:    Dim array1, array2 As Integer()
4:    array1 = New Integer() {10, 20, 30, 40, 50, _
   60, 70, 90, 70, 60, 50}
5:    array2 = New Integer(array1.GetUpperBound(0)) {}
6:    For i = 0 To array2.GetUpperBound(0)
7:    array2(i) = 2 + 2 * i
8:    Next
9:    output = "Index " & vbTab & "First " & vbTab & _
   "Secand" & vbCrLf
10:     For i = 0 To array1.GetUpperBound(0)
11:  output &= i & vbTab & array1(i) & vbTab & array2(i)&vbCrLf
12:     Next
13:       MessageBox.Show(output, " Values", _
14:       MessageBoxButtons.OK)
```

في هذه السطور:

- في السطر رقم 3 يتم الاعلان عن array1,array2 كمصفوفات من نوع القيم الصحيحة Integers.
- في السطر رقم 4 يتم إعطاء قيم لعناصر المصفوفة array1 مجموعة من القيم.
- في السطر رقم 6 يتم ابتداء التكرار for ابتداء من 0 إلى نهاية المصفوفة array2.
- GetUpperBound(0).
- في السطر رقم 7 يتم وضع قيم في المصفوفة array2 بمعلومية متغير التكرار I .
- في السطر رقم 9 يتم تكوين المتغير output بكلمة Index ثم مسافة ثم مسافة tab ثم كلمة First ثم مسافة كلمة Secand .
- في السطر رقم 10 يتم دمج عناصر المصفوفات array1,array2 مع ما سبق وذلك من خلال التكرار for.

في النهاية تم MessageBox.Show() لعرض النتائج.

نفذ البرنامج تحصل على النتيجة التالية كما في الشكل (4-6).

الشكل (4-6)

فى لغة C#

المصفوفة ذات البعدين :

يتم الإعلان عنها بالشكل التالي :

```
int[,] myArray = new int[4,2];
```

و فيها يتم الإعلان عن مصفوفة بالاسم myArray بعدد صفوف 4 و عدد أعمدة 2 من النوع int أي الأرقام الصحيحة .

و بنفس القاعدة يمكن تغيير عدد العناصر حسب الطلب و كذلك تغيير نوع المصفوفة و لتوضيح ذلك تابع الأمثلة التالية :

```
int[,] myArray = new int[4,2];
int[,,] myArray = new int [4,2,3];

int[,] myArray = new  int[,] {{1,2}, {3,4}, {5,6}, {7,8}};

int[,] myArray = {{1,2}, {3,4}, {5,6}, {7,8}};

int[,] myArray;

myArray = new int[,] {{1,2}, {3,4}, {5,6}, {7,8}};   // OK

myArray = {{1,2}, {3,4}, {5,6}, {7,8}};   // Error

myArray[2,1] = 25;
```

فى لغة Java

المصفوفات متعددة الأبعاد Multi dimensional Array

الإعلان عن مصفوفة متعددة الأبعاد.

ويتم ذلك باسم المصفوفة ونوعها وتحديد عدد الصفوف وعدد الأعمدة كما يلى:

```
int XY [ ] [ ] = New int [5] [6];
```

فى هذا السطر تم الإعلان عن مصفوفة بالاسم XY ذات بعدين من النوع int ثم استعمال new لتحديد عدد الصفوف بالعدد 5 والأعمدة بالعدد 6.

ويتم تسجيل أو التعامل مع أى عنصر بتحديد الصف والعمود فمثلاً لوضع قيمة فى العنصر الموجود فى الصف الثانى (2) والعمود الثالث (3) نكتب السطر التالى:

```
XY [2] [3] = 90;
```

وللتعامل مع المصفوفة ذات البعدين تستعمل اثنين تكرار متداخلة باستعمال for loop والذى يسمى Nested كما سيل فى فقرة التكرار Loops.

بلوك الأوامر Block Statements

كما يوجد فى لغة C ما يسمى بلوك كذلك يوجد فى لغة Java وهو عبارة عن أى مجموعه من الجمل أو الأوامر محصورة بين الأقواس { } كما بالشكل التالى:

```
{
    :
    :
    :
Statement 1
Statement 2
    :
    :
}
```

فى هذا الشكل يتم حصر جمل Statement 1 و Statement 2 بين الأقواس ليتعامل كبلوك.

ولكن السؤال ما الفائدة وراء إنشاء بلوك جمل ؟

الفائدة هي إمكانية الإعلان عن متغيرات داخل البلوك ليس لها علاقة بما قبل او بعد البلوك مثل تعريف متغير داخل بلوك داخل دالة فهو خاص بها.

مثال :

في هذا المثال نوضح كيفية استعمال التكرار المتداخل مع المصفوفات وكيفية التعامل مع عناصر المصفوفات بدقة اكتب سطور هذا البرنامج كما في الشكل.

```
1  import java.io.*;
2  public class MyFriends {
3
4      public static void main (String args [ ]) throws IOException
5
6      {
7        int x,y;
8          InputStreamReader inStream = new InputStreamReader( System.in );
9        BufferedReader stdin = new BufferedReader( inStream );
10
11       String inData;
12   char  [ ][ ]  arr=new char [40][40];
13     int r=0,c=0;
14
15     for (r=0;r<40;r++)
16     for (c=0;c<40;c++)
17     arr[r][c]=' ';
18   do
19   {
20
21   for (r=0;r<40;r++)
22   {
23     System.out.println ("");
24
25     for (c=0;c<40;c++)
26     System.out.print (arr[r][c]);
27     }
28        System.out.print ("X=");
29        System.out.print("Enter the u r Name :");
30     inData = stdin.readLine();
31     x = Integer.parseInt( inData );
32
33        System.out.print(" Y=");
34        inData = stdin.readLine();
35      y = Integer.parseInt( inData );
36      arr[x][y]='X';
37    }while (x>0);
38    }
39 }
```

في هذه السطور:

في السطر رقم 12 تم الاعلان عن مصفوفة من نوع حروف عدد صفوفها وأعمدتها 40

فى السطور 15و16و17 يتم ملئ عناصر هذه المصفوفة بالحرف . وذلك باستخدام التكرار المتداخل مع المصفوفة.

فى السطر رقم 18 يبدأ تكرار do.

فى السطور من 21 إلى 27 يتم طباعة عناصر المصفوفة وهى طباعة 40x40 من النقط مما يعطى شكل ورقة رسم بيانى.

فى السطر رقم 30 يتم استقبال قيمة x.

فى السطر رقم 35 يتم استقبال قيمة y.

فى السطر رقم 36 يتم وضع الحرف x فى عنصر المصفوفة x,y .

ثم يعاد طباعة المصفوفة بالشكل الجديد مما يؤدى إلى طباعة ورقة رسم بيانى موقع عليها هذه النقطة، ويستمر ذلك حتى يدخل المستخدم قيمة سالبة.

نفذ البرنامج لتحصل على برنامج رسم بيانى يستقبل منك النقطة (r,c) المطلوب توقيعها ثم يقوم البرنامج برسم علامة x فيها وهكذا حتى تدخل قيمة سالبة كما فى الشكل.