المحاضرة الأولى

مدرس المقرر م. رود الأصفر

المصفوفات:

هي عبارة عن مجموعة ذات حجم ثابت من العناصر التي لها نفس نوع البيانات ولكنها تختلف في القيم المُسندة إليها.

يحتاج المبرمج في كثير من الأحيان إلى تخزين البيانات بعد قراءتها بهدف الرجوع إليها في إجراء بعض العمليات، وعندها يحتاج إلى التصريح وبشكل منفصل عن عدة متحولات لها نفس نوع البيانات مثلاً num1, num2, ...num50 ، وهنا تبرز أهمية المصفوفات في تخزين قيم عدة متحولات من نفس النوع ضمن تسلسل معين مما يسهّل الوصول إليها والتعامل معها لاحقاً، حيث يتم تعريف متحول واحد من نوع مصفوفة وليكن على سبيل المثال numbers ، ثم يتم التعامل مع عناصر هذه المصفوفة بشكل مستقل عن طريق الـ Index (الدليل)الخاص بهذا العنصر

numbers[0], numbers[1], numbers[2],.....

أنواع المصفوفات:

- المصفوفات أحادية البعد (vector).
- المصفوفات ثنائية البعد.

التصريح عن مصفوفة أحادية البعد: يتم التصريح عنها بالشكل التالي:

```
dataType[]  arrayName = new dataType[n];
```

حيث:

dataType : نوع عناصر المصفوفة (int , float , string, ...).

arrayName : اسم المصفوفة.

n : عدد عناصر المصفوفة، ويسمى طول المصفوفة(length)

مثال:

```
int marks [ ] = new int [6] ;
```

هنا تم التصريح مصفوفة اسمها marks مؤلفة من 6 أعداد صحيحة.

للوصول إلى أي عنصر من عنصر المصفوفة لا بد من استخدام دليل يدل على هذا العنصر، علماً ان ترتيب عناصر المصفوفة يبدأ من الصفر، وبالتالي فإن:

ترتيب آخر عنصر من عناصر المصفوفة = عدد العناصر - ١

- يمكن اسناد قيم ابتدائية لعناصر المصفوفة مباشرة أثناء التصريح عنها كما في المثال التالي:

```
int marks [ ] = new int [6] {77, 99, 85, 66, 90} ;
```

في هذا المثال تم اسناد قيم لعناصر المصفوفة أثناء التصريح عنها، وبالتالي فإن:

marks[0]=77
marks[1]=99
marks[2]=85
marks[3]=66
marks[4]=90

إدخال قيم عناصر المصفوفة:

يتم التعامل مع عناصر المصفوفة كأي مجموعة من المتغيرات المشتركة بالنوع، لكن كونها تقع ضمن مصفوفة فإن هذا الأمر يقدم مجموعة من الميزات كإمكانية العودة إليها باستمرار كما ذكرنا سابقا، وكذلك إمكانية إدخال أو قراءة قيمها بشكل أبسط وأسرع من إدخال قيمة كل متغير على حدا، الأمر الذي يصبح معقدا جدا عند التعامل مع أعداد ضخمة من المتغيرات، يمكننا استخدام الحلقات للتعامل مع عناصر المصفوفة مثل For, Foreach، مع ملاحظة أن الحلقة foreach تستخدم فقط لقراءة عناصر المصفوفات وليس لتعديل هذه العناصر.

مثال:

```
static void Main(string[] args)
{

   string[ ] courses = new string[ 4] {"Programming 2" ,
                "Math 2" , "English", "Arabic" } ;

   foreach (  string  c  in  courses  )
   {
       Console.WriteLine( c  ) ;
   }
     }
```

مثال: اكتب برنامج يطلب من المستخدم إدخال عناصر مصفوفة مؤلفة من ١٠ عناصر، ثم يقوم بطباعة مجموع عناصر المصفوفة على الشاشة.

```
static void Main(string[] args)
{

   int[ ] myarr = new int[10];
   int elements_sum = 0;
   int  i ;

   Console.WriteLine("Enter the elements of array");
   for (  i = 0  ;  i < myarr.length  ;  i++)
   myarr [ i ] = int.Parse(Console.ReadLine());


   for (  i = 0 ; i < numbers.length ;  i++)
   elements_sum + = myarr [ i ] ;

   Console.WriteLine("Sum of array elements = {0}",
                        elements_sum ) ;
}
```

مثال: اكتب برنامج يطلب من المستخدم إدخال عناصر مصفوفة مؤلفة من ٣٠ عنصر، ثم يقوم بطباعة عدد مرات تكرار كل عنصر من عناصر المصفوفة.

```
static void Main(string[] args)
{
   int[ ] numbers = new int[30];
   int i , j , count = 0;

   Console.WriteLine("Enter the elements of array");
   for (  i = 0 ; i < numbers.length ; i++)
     numbers [ i ] = int.Parse(Console.ReadLine());

   for ( i = 0 ; i < numbers.length ;  i++)
   {
     for (  j = 0 ; j < numbers.length ;  j++)
     if (numbers[ i ] == numbers[ j ])
       count++;
     Console.WriteLine("count of element  {0} = {1} " ,
                numbers[i] , count);
     count = 0;
       }
 }
```

المصفوفات ثنائية البعد: تتكون من أكثر من سطر وأكثر من عمود ، ويتم التصريح عنها بالشكل التالي:

```
dataType[ , ]  arrayName = new dataType[n,m];
```

حيث:

dataType	:	نوع عناصر المصفوفة .
arrayName	:	اسم المصفوفة.
n	:	عدد أسطر المصفوفة.
m	:	عدد أعمدة المصفوفة.

مثال:

```
int  arr [ , ] = new  arr [5 , 3] ;
```

عرفنا مصفوفة اسمها arr مؤلفة من 5 أسطر و ٣ أعمدة ، عناصر هذه المصفوفة أعداد صحيحة.

للوصول إلى أي عنصر من عنصر المصفوفة الثنائية نحتاج إلى دليل للأسطر ودليل للأعمدة.

وكما في المصفوفات الأحادية يمكن إسناد القيم مباشرة لعناصر المصفوفة عند التصريح عنها.

```
int  arr [ , ] = new  arr [2 , 3] { {1,2,3} , {4,5,6} } ;
```

مثال: اكتب برنامج يطلب من المستخدم ادخال عناصر مصفوفة ثنائية البعد على أن يتم تحديد عدد العناصر من قبل المستخدم أيضاً، ثم:

- طباعة عناصر المصفوفة بشكلها الرياضي
- طباعة اكبر عدد في المصفوفة، وكذلك أصغر عدد.
- طباعة مجموع عناصر المصفوفة بكاملها ثم المتوسط الحسابي للعناصر.
- طباعة مجموع عناصر القطر الرئيسي، ومجموع عناصر القطر الثانوي.
- طباعة مجموع عناصر السطر الاول والعمود الأول.

```
static void Main(string[ ] args)
{
  Console.WriteLine("Please enter the number of  Rows:");
  int n = int.Parse(Console.ReadLine());

  Console.WriteLine("Please enter the number of Columns:");
  int m = int.Parse(Console.ReadLine());

  int [ , ] x = new int [n , m] ;
  int  sum = 0, sum1 = 0, sum2 = 0, sum3 = 0, sum4 = 0;

  Console.WriteLine("Please enter elements of  Array ");
  for ( int i= 0 ;  i < n ;  i++)
   for ( int  j = 0 ;  j < m ;  j++)
     {
        Console.Write  (" x[{0},{1}]=  ", i , j ) ;
         x[ i , j ] = int.Parse(Console.ReadLine());
      }

  for (int  i = 0; i < n; i++)
  {
    for ( int j = 0; j < m ; j++)
     {
           Console.Write  (x[i , j ] + "\t");
     }
    Console.WriteLine( );
  }

 int min = x[0, 0];
 int max = x[0, 0];

 for (int   i = 0 ; i < n ; i++)
 for (int  j = 0 ; j < m ; j++)
   if (x[ i, j ] < min)
            min = x[i, j];
  Console.WriteLine("minimum element in Array =  {0}" , min);

 for (int  i = 0; i < n; i++)
  for (int  j = 0; j < m; j++)
    if (x[ i ,  j ] > max)
           max = x[i, j];
 Console.WriteLine("maximum element in Array  =  {0}" , max);

for (int  i = 0 ; i < n ; i++ )
for ( int  j = 0 ;  j <m ;  j++)
       sum += x[ i , j ] ;
Console.WriteLine("Sum of  Array elements  =  " + sum ) ;
```

```
int  z = n * m;
double  avg = (double) sum / z;
Console.WriteLine("Avarage of Array elements  =  "+ avg);


for ( int  i = 0 ; i < n ; i++)
   sum1 +=  x [ i , i ] ;
Console.WriteLine("Sum of  the main  = " + sum1) ;

int k = m - 1;
for (int i = 0 ; i < n ; i++)
{
  if (k >= 0)
        sum2 += x[i, k];
  k-- ;
 }
Console.WriteLine("Sum of  secondary =  " + sum2);


for (int  j = 0 ; j < m ; j++)
    sum3 +=  x[ 0, j ] ;
Console.WriteLine("Sum of  the first row =  " + sum3);

for (int i = 0 ; i < n ; i++)
sum4 += x[ i , 0] ;
Console.WriteLine("Sum of  the first column = " + sum4);
}
```

مثال: اكتب برنامج يطلب من المستخدم إدخال عناصر مصفوفة محارف أبعادها ٣*٣، ثم يقوم بالبحث ضمن عناصر المصفوفة عن عنصر يحدده المستخدم ويعيد عدد مرات تكرار هذا العنصر في حال وجوده.

```
static void Main(string[] args)
{
  char[,] myarr = new char[3, 3];
  int c = 0;

  Console.WriteLine("Please enter the char which you
                     search");
  char k = char.Parse(Console.ReadLine());

  Console.WriteLine("Enter the elements of array:");
  for(int i=0 ; i<3 ; i++)
```

```
for(int j=0 ; j<3 ; j++)
    myarr[i, j] = char.Parse(Console.ReadLine());

for (int i = 0; i < 3; i++)
for (int j = 0; j < 3; j++)
  if (myarr[i, j] == k)
            c++;

if (c == 0)
  Console.WriteLine("Not Found");
else
  Console.WriteLine("Found, " + c + "Times");
}
```