المحاضرة السادسة

المصفوفات والسلاسل النصية
Arrays and Strings

عند التعامل مع حجم كبير من البيانات كنا نستخدم الكثير من المتغيرات لتخزين ومعالجة تلك البيانات , لأن كل متغير يخزن فيه قيمة واحدة فقط , ولكن هذا يجعل البرنامج طويلا ومعقداً في بعض الأحيان

ولكن باستخدام المصفوفة يستطيع المبرمج استعمال متغيرات قليلة وذلك بتقسيم كل متغير إلى عدد من العناصر المتسلسلة

تعريف المصفوفة :-

هي عبارة عن منطقة في الذاكرة تتكون من عدد محدد Finite ومتجانس Homogenous من المواقع المتجاورة

أ- التجانس ويعني أن هذا الجزء من الذاكرة يستخدم لتمثيل نوع واحد من البيانات Data Type

ب- محدد Finite هو عبارة عن رقم صحيح يستخدم لتحديد عدد المواقع المطلوبة ويسمى هذا الرقم بالدليل subscript

أنواع المصفوفات :-

يمكن تقسيم المصفوفات إلى نوعين هما :

1. **المصفوفات ذات البعد الواحد One Dimensional Arrays**

تعريف المصفوفة ذات البعد الواحد:-

هي عبارة عن صف أو عمود يحتوي على مجموعة من عناصر البيانات متحدة النوع والاسم

الصيغة العامة للإعلان عن المصفوفة ذات البعد الواحد:

Data Type Array Name [Index] ;

حيث :-

a. Data Type: نوع بيانات المصفوفة

b. Array Name: اسم المصفوفة ويراعى فيه شروط تسمية المتغيرات

c. Index: دليل المصفوفة (عدد عناصرها) وهي عبارة عن قيمة صحيحة يمكن أن تكون محدده(ثابتة) أو مدخلة من قبل المستخدم(متغيرة)

أمثلة :

1. int x [50];

إعلان عن مصفوفة حجمها 50 وبياناتها من النوع الصحيح

2. float y [20];

إعلان عن مصفوفة حجمها 20 وبياناتها من النوع الحقيقي

3. char name [15] ;

إعلان عن مصفوفة حجمها 15 وبياناتها من النوع الحرفي

4. int x [3] = { 5,10,15} ;

إعلان عن مصفوفة حجمها 3 وبياناتها من النوع الصحيح مع إعطائها قيم ابتدائية

5. char y [4] = { 'a','b','c','d' } ;

إعلان عن مصفوفة حجمها 4 وبياناتها من النوع الحرفي مع إعطائها قيم ابتدائية

الوصول إلى عناصر المصفوفة (Array Elements Allocation)

للوصول إلى المواقع داخل المصفوفة يستخدم الدليل الذي تضاف له وظيفة أخرى إذ انه يعمل كمرجع للخلايا المكونة للمصفوفة حيث ترقم الخلايا بالصورة التالية :

المواقع في الذاكرة

0	1	2	.....	n-2	n-1

عناصر المصفوفة

1 2 3 n-1 n

وبالتالي للوصول إلى أي موقع لإجراء عملية إدخال أو معالجة أو إخراج أو اتخاذ قرار نستخدم اسم المصفوفة زائداً دليل الموقع المراد الوصول إليه .

يبين الشكل التالي مصفوفة من الأعداد الصحيحة اسمها C. وهي تتضمن 10 عناصر:

C[0]	-5
C[1]	8
C[2]	32
C[3]	11
C[4]	0
C[5]	-14

C[6]	36
C[7]	-76
C[8]	1
C[9]	9

☒	يمكن الرجوع إلى أي عنصر من العناصر السابقة بإعطاء اسم المصفوفة متبوعاً برقم موضع العنصر داخل قوسين من الشكل []
☒	العنصر الأول في المصفوفة هو دائما رقم صفر، لذلك يتم الرجوع إلى العنصر الأول في المصفوفة C على الشكل التالي C[0]، والعنصر الثاني C[1]، أما من أجل الرجوع إلى العنصر السابع نكتب C[6]، وبشكل عام نستطيع القول أننا نرجع إلى العنصر رقم i بأن نكتب C[i-1].
☒	نسمي رقم الموضع الذي نضعه بين قوسين بالدليل (subscript)، ويجب أن يكون الدليل عبارة عن عدد صحيح أو أي تعبير يعطي قيمة صحيحة.

برامج متكاملة عن المصفوفة ذات البعد الواحد

مثال1:

عرف مصفوفة تستخدم لتمثيل 5 قيم صحيحة حيث أن القيمة رقم 1 هي 20 ورقم 3 هي 15 والقيمة رقم 2 تساوي القيمة رقم 1 زائدا القيمة رقم 3 والقيمة الرابعة هي 3-2-4 والقيمة الخامسة تساوي القيمة الأكبر ما بين الثانية والرابعة أي 4>2=5 أو 2<4=5 ؛ ثم أطبع عناصرها

```
#include <iostream.h>
int main( )
{
   int i,x[5];
   x[0]=20;
   x[2]=15;
   x[1]=x[0]+x[2];
   x[3]=x[1]-x[2];
   if(x[1]>x[3])
   x[4]=x[1];
   else
   x[4]=x[3];
   for(i=0;i<5;i++)
```

```
    cout<<"x["<<i+1<<"]="<<x[i]<<"\n";
     return 0;
}
```

مثال2:

اكتب برنامجاً بلغة C++ لقراءة عناصر مصفوفة ذات بعد واحد تتكون من 10 عناصر ثم حساب وطباعة مجموع عناصر هذه المصفوفة.

```
#include <iostream.h>
int main()
{
  int a[10];
    int sum,i;
    sum=0;
  for(i=0;i<10;i++)
  {
     cin>>a[i];
     sum +=a[i];
  }
  cout<<"sum="<<sum<<"\n";
  return 0;
}
```

مثال3:

اكتب برنامج باستخدام المصفوفات لحساب متوسط 5 قيم مدخلة من النوع الحقيقي ؟

```
#include<iostream.h>
int  main( )
{
float x[5] , sum=0 , avg;
int i;
for(i=0;i<5;i++)
{
cout<<"enter elemant["<<i+1<<"] \n";
cin>>x[i];
sum+=x[i];
}
avg=sum/5;
cout<<"the average="<<avg<<"\n";
return 0;
```

```
}
```

مثال4: اكتب برنامج لإدخال 5 قيم حقيقية ثم طباعتها بصورة مرتبة تصاعديا؟

```
#include <iostream.h>
const int n=5;
int main( )
{
float x[n],temp;
int i,j;
for(i=0;i<n;i++)
{
cout<<"enter elemant["<<i+1<<"]\n";
cin>>x[i];
}
for(i=0;i<n-1;i++)
for(j=i+1;j<n;j++)
{
if(x[i]>x[j])
{
temp=x[i];
x[i]=x[j];
x[j]=temp;
}
}
cout<<"elements after sorting\n";
for(i=0;i<n;i++)
cout<<x[i]<<"\n";
    return 0;
}
```

تمرين

اكتب برنامجاً بلغة C++ لحساب مجموع مصفوفتين a وb وتخزين المجموع في المصفوفة c ثم طباعة عناصر المصفوفة الناتجة c علماً بأن عدد عناصر كلاً من المصفوفتين c وb هو 5.

2. المصفوفات متعددة الأبعاد Multi Dimensional Arrays

المصفوفات متعددة الأبعاد هي التي تتكون من عدة أبعاد. وعادة ما يتم التعامل مع المصفوفات ذات البعدين

لقلة التطبيقات التي تحتاج لمصفوفات ذات أكثر من بعدين.

المصفوفة ذات البعدين تتألف من مجموعة من الصفوف rows ومجموعة من الأعمدة columns وفيها يتم إعطاء المصفوفة دليلين n * m ويساعدنا هذا النوع من المصفوفات في تمثيل المصفوفات الرياضية على الذاكرة بافتراض أن (n تمثل دليل الصفوف و m تمثل دليل الأعمدة)

الصيغة العامة للإعلان عن المصفوفة ذات البعدين:

data type array name [row size] [column size];

نوع المصفوفة حجم الصفوف اسم المصفوفة حجم الأعمدة

☒	يتم التصريح عن المصفوفات ذات البعدين بنفس طريقة التصريح عن المصفوفات ذات البعد الواحد غير أننا نحدد عدد الصفوف والأعمدة في المصفوفات ذات البعدين

أمثلة :

1. int y[4][3]

تصريح عن المصفوفة الصحيحة y كمصفوفة ذات بعدين وتتكون من 4 صفوف وثلاثة أعمدة

2. y[3][2]=23;

إسناد القيمة 23 لعنصر الصف الرابع والعمود الثالث من المصفوفة y

3. int y[4][3]={{5,0,-4},{-2,3,1},{4,7,6},{9,8,-1}};

إسناد قيم ابتدائية لعناصر المصفوفة y أثناء التصريح. حيث المصفوفة y عناصرها كما يلي:

$$y = \begin{bmatrix} 5 & 0 & -4 \\ -2 & 3 & 1 \\ 4 & 7 & 6 \\ 9 & 8 & -1 \end{bmatrix}$$

حيث يتم تجميع عناصر كل صف ضمن قوسين.

ومن أجل الوصول إلى أي عنصر من عناصر المصفوفة ذات البعدين يتم تحديد رقم الصف ورقم العمود الواقع فيه هذا العنصر.

والشكل التالي يوضح مصفوفة ذات بعدين:

	العمود 1	العمود 2	العمود 3	العمود 4

☐الصف 1	a[0][0]	a[0][1]	a[0][2]	a[0][3]
☐الصف 2	a[1][0]	a[1][1]	a[1][2]	a[1][3]
☐الصف 3	a[2][0]	a[2][1]	a[2][2]	a[2][3]

☒	يمكن تحديد أي من عناصر المصفوفة ذات البعدين باستخدام دليلين, دليل لرقم الصف ودليل آخر لرقم العمود فمثلاً لتحديد العنصر الموجود في الصف الثاني والعمود الثالث من المصفوفة x نكتب x[1][2] لاحظ هنا أن الصف الأول يأخذ الرقم 0 والعمود الأول يأخذ الرقم 0 أيضاً.

☒	إذا لم يتم إعطاء قيم كافية لعناصر المصفوفة فأنه يتم إسناد القيمة 0 إلى باقي العناصر التي لم يتم إسناد قيم لها. فمثلاً التصريح: ☐ int b[2][2]={{1},{3,4}} يعطي العنصر b[0][0] القيمة 1 والعنصر b[1][0] القيمة 3 والعنصر b[1][1] القيمة 4 وباقي العناصر القيمة 0.

برامج متكاملة عن المصفوفة ذات البعدين

متال1:

برنامج يقوم بقراءة مصفوفة ذات بعدين مع طباعة البيانات المدخلة على هيئة المصفوفة الثنائية؟

```
#include<iostream>
const int  row=3;
const int col=4;
int main( )
{
   int a[row][col],i,j;
   cout<<"enter the elements of array:\n";
   for(i=0;i<row;i++)
   {
        for(j=0;j<col;j++)
        {
         cout<<"a["<<i<<","<<j<<"]=";
         cin>>a[i][j];
```

```
    }
  }
   cout<<"the arry looks like:\n";
   for(i=0;i<row;i++)
   {
   for(j=0;j<col;j++)
   cout<<a[i][j]<<"\t";
   cout<<"\n";
   }
 return 0;
 }
```

متال2

اكتب برنامج يقوم بجمع مصفوفتين صحيحتين من النوع 3*3 لأعداد يتم إدخالها من قبل المستخدم ثم طباعة المصفوفات على هيئة المصفوفة الثنائية؟

```
#include <iostream.h>
int main( )
{
int a[3][3],b[3][3],c[3][3],i,j;
cout<<"first  array a[3][3]:\n";
for(i=0;i<3;i++)
{
for(j=0;j<3;j++)
cin>>a[i][j];
cout<<"\n";
}
cout<<"second array b[3][3]:\n";
for(i=0;i<3;i++)
{
for(j=0;j<3;j++)
cin>>b[i][j];
cout<<"\n";
}
cout<<"the first  array a[3][3] look like :\n";
for(i=0;i<3;i++)
{
for(j=0;j<3;j++)
cout<<a[i][j]<<"\t";
cout<<"\n";
}
cout<<"the second array b[3][3] look like :\n";
for(i=0;i<3;i++)
```

```
{
for(j=0;j<3;j++)
cout<<b[i][j]<<"\t";
cout<<"\n";
}
cout<<"the sum of tow array c[3][3] look like:\n";
for(i=0;i<3;i++)
{
for(j=0;j<3;j++)
{
c[i][j]=a[i][j]+b[i][j];
cout<<c[i][j]<<"\t";
}
cout<<"\n";
return 0;
}
```

تمرين

1- اكتب برنامج لحساب حاصل ضرب المصفوفتين b(3*4), a(4*3) والناتج c(4*4) ثم طباعة المصفوفات على هيئة المصفوفة الثنائية؟؟

2- اكتب برنامجاً لقراءة عناصر مصفوفة X(4x3) ثم عمل ما يلي:

أ- طباعة المصفوفة بشكل مناسب.

ب- طباعة عدد الأرقام الموجبة وعدد الأرقام السالبة وعدد الأرقام الصفرية في المصفوفة

ج- وطباعة عدد الأرقام الزوجية والفردية.

السلاسل النصية
strings

السلسلة النصية هي عبارة عن مصفوفة حرفية حيث انه لا يوجد متغير من نوع string في لغة C++

الصيغة العامة للإعلان عن المصفوفة الحرفية (السلسة النصية)

```
char var [size];
```

حيث :

char تعني نوع البيانات حرفي

var اسم المصفوفة الحرفية (السلسة النصية)

size حجم السلسة النصية

مثال 1: char name[20];

إعلان عن سلسلة نصية حجمها 20 حرفاً

خواص السلاسل النصية

1. يتم إلحاق عنصر إضافي لنهاية السلسلة وقيمته '\0' أو NULL حيث أن العدد الكلي لحروف السلسلة دائماً يزيد بواحد على طول السلسلة.
2. يمكن بدأ السلسلة بحروف (إدخال السلسلة أثناء التصريح) كما يلي:

```
Char str[]="Riyadh";
```

3. يمكن إخراج السلسلة بالكامل كهدف واحد مثال:

```
cout<<str;
```

سيطبع حروف السلسلة حتى يصل إلى '\0' أو NULL.

4. السلسلة بكاملها يمكن إدخالها كهدف واحد كما يلي:

```
cin>>str;
```

5. دوال التعامل مع السلاسل النصية مضمنة في المكتبة string

مثال 2: برنامج لطباعة سلسلة نصية محددة في التعريف

```
#include <iostream.h>
int main()
{
char s[]="C++ Language";
cout<<s<<"\n";
return 0;
}
```

مثال3: برنامج يوضح كيفية إضافة الرقم الصفري إلى نهاية السلسلة:

```
#include <iostream.h>
int main()
{
char str[]="Riyadh";
for(int i=0;i<7;i++)
cout<<"str["<<i<<"]="<<str[i]<<endl;
 return 0;
}
```

دوال التعامل مع السلاسل النصية

1. دالة الطول strlen

وهي اختصار للعبارة string length حيث تستخدم لإيجاد طول السلسلة النصية والتي تقع قبل الرمز '\0' والذي يستخدم من قبل المترجم للإشارة إلى نهاية السلسلة .

الصيغة العامة لها :

strlen (string name);

حيث string name اسم السلسلة

فمثلا العبارة "How are you"

تحتوي على 11 رمز أي 9 حروف ومسافتين خاليتين .

مثال : برنامج يوضح كيفية استخدام الدالة strlen() حيث يتم تعريف سلسلة محددة في البرنامج وطباعة طولها

```
#include<iostream.h>
#include<string.h>
main()
{
char str[]="How are you";
cout<<"the length of string="<<strlen(str)<<"\n";
return 0;
}
```

2. دالة الوصل (الدمج) strcat

وهي اختصار للعبارة string concatenation حيث تستخدم لوصل سلسلة نصية بأخرى .

الصيغة العامة لها :

stract (string1,string2)

حيث string2,string1 سلاسل نصية

وهي تعنى وصل السلسلة string2 عند نهاية السلسلة string1

☒	يجب حجز الطول المناسب للسلسلة string1 لأنه بعد عملية الوصل تحتوي على طول السلسلتين معا .

مثال :

```
#include <iostream.h>
#include <string.h>
main()
{
char s1[]="Teachers";
char s2[]=" College";
```

```
strcat(s1,s2);
cout<<s1<<"\n";
cout<<"The length of new string="<<strlen(s1)<<"\n";
 return 0;
}
```

3. دالة النسخ strcpy()

هذه الدالة اختصار للعبارة string copy

الصيغة العامة لها :

strcpy(string1,string2)

تستخدم لنسخ السلسلة string2 في السلسلة string1 حيث تفقد السلسلة string1 القيمة القديمة أي القيمة التي قبل النسخ .

☒	يجب حجز الطول المناسب للسلسلة string1 لأنه بعد عملية النسخ تحتوي على يصبح محتواها هو السلسلة string2 .

مثال :

```
#include <iostream.h>
#include <string.h>
main()
{
char s1[]="King Saud University";
char s2[]="Teachers College";
cout<<"s1:"<<s1<<"\n";
cout<<"s2:"<<s2<<"\n";
strcpy(s1,s2);
cout<<"copying s2 into s1:"<<s1<<"\n";
return 0;
}
```