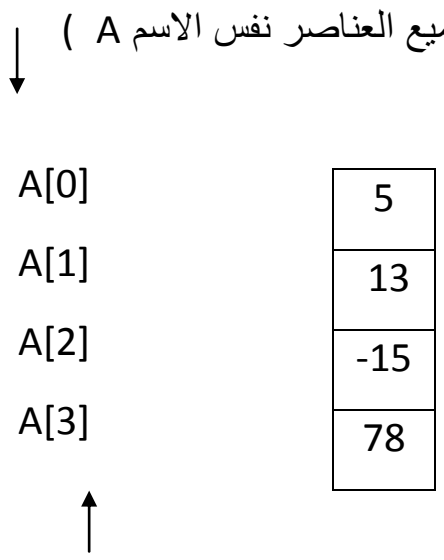
4-المصفوفات

المصفوفات:

المصفوفة هي عبارة عن مجموعة من خانات الذاكرة المتتالية التي لها نفس الاسم ونفس النمط . ومن أجل الرجوع الى خانة معينة من هذه الخانات ضمن المصفوفة ورقم موضع الخانة (العنصر) ضمن المصفوفة وذالك داخل قوسين متوسطين من الشكل ([ ]) .

والشكل التالي يمثل مصفوفة من الإعداد الصحيحة التي أسمها A وهي تتضمن أربعة عناصر .



{رقم موضع العنصر من عناصر المصفوفة (رقم موضع الخانة)}

العنصر الاول من المصفوفة هو دائماً العنصر ذو الرقم صفر وبالتالي يتم الرجوع اليه من المصفوفة A مثلاً على الشكل التالي A[0] وبشكل عام نستطيع القول أننا نرجع الى العنصر ذو الرقم I بكتابة [I-1] اسم المصفوفة

نسمي رقم الموضع الذي نضعه ضمن قوسين متوسطين بالدليل Subscript ويجب أن يكون الدليل عبارة عن عدد صحيح أو تعبير يعطي قيمة صحيحة حيث يتم حساب قيمة التعبير أولاً من أجل تحديد المطلوب ، على سبيل المثال c=2 وb=3 وبالتالي يكون العنصر A[b+c] يمثل العنصر A[5] .

- **التصريح عن المصفوفات :**

تشغل المصفوفات أجزاء محددة من الذاكرة لذلك نقوم بتحديد نمط عناصر المصفوفة وعددها الى المترجم الذي يقوم بدوره بحجز الحجم المناسب في الذاكرة . وبتصريح الشكل العام التالي:

[عدد عناصر المصفوفة] اسم المصفوفة نمط معطيات المصفوفة

مثلاً : int A[5]

يمكن حجز أمكنة لعدة مصفوفات باستخدام تصريح وحيد ، فعلى سبيل المثال ; Y[13] , int x[10]

ويمكن التصريح عن المصفوفات تحتوي معطيات من أنماط أخرى مثل ; float x[100] ، ; char y [100] ، .....

ويسمى هذا النوع من المصفوفات بالمصفوفات ذات البعد الواحد .

- **أمثلة عن طرق اعطاء قيم ابتدائية لعناصر المصفوفة :**

1. يمكن اعطاء قيمة ثابتة لكامل العناصر المصفوفة فعلى سبيل المثال نعطي الصفر لكامل عناصر المصفوفة على الشكل التالي int A[10]={0}.

2. يمكن اعطاء قيم ابتدائية لعناصر المصفوفة أثناء التصريح عنها مثلا ; int A[5]={10,2,34,6,18}

3. يمكن أعطاء قيم ابتدائية لعناصر المصفوفة بالشكل التالي :

```
# include < iostream.h >
main()
{
int a[5]
for(int i=; i <5 ;i ++ )
a[i] = 0 ;
return 0 ;
}
```

مثال 1 :

أكتب برنامج يقوم بطباعة عناصر مصفوفة .

```
# include < iostream.h >
main()
{
int a[5] = {10,2,12,30,67} ;
for (int i =0; i<5 ;i++)
     cout <<"a["<<i<<"] = "<<a[i]<<"\ n ";
return;
}
```

ملاحظات :

1 ـ يسبب التصريح التالي :

```
int [5] ={1,2,34,56,24,14};
```

خطأً قواعدياً لاننا أعطينا ستة قيم لمصفوفة مؤلفة من خمس عناصر فقط .

2 ـ يسبب التصريح التالي :

```
int n[5]={1,2,9,5};
```

اعطاء قيمة الصفر للعنصر الخامس من قبل المترجم .

3 ـ اذا تم حذف حجم المصفوفة أثناء التصريح عنها فان عدد عناصر هذه المصفوفة يصبح مساوياً لعدد القيم الابتدائية المعطاة ضمن القائمة الملحقة بالتصريح . لذلك يقوم التصريح التالي :

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

بخلق مصفوفة مؤلفة من ستة عناصر .

التصريح عن متحول ثابت :

يكون الشكل العام للتصريح عن المتحول ثابت كالتالي :

```
Const نوع المعطيات اسم المتحول = القيمة ;
```

مثال :

```
Const int size=10;
```

يفيد السطر السابق في التصريح عن متحول ثابت Size وذلك باستخدام الكلمة المحجوزة const 10

ملاحظة هامة :

يجب اعطاء قيمة ابتدائية للمتحولات الثابتة عند التصريح عنها **ولا يمكن تغيير هذه القيمة بعد ذلك** ، تسمى المتحولات الثابتة أيضاً بالثوابت constants .

ومن **الاخطاء البرمجية الشائعة** اعطاء قيمة لثابت من خلال تعليمة تنفيذية مثل :

```
main ( )
{
   const int n ;
  n = 9 ;
   return 0 ;
}
```

وبالتالي تعطي التعليمات السابقة خطأً قواعدياً نتيجة اسناد متحول ثابت ، ويكون التصحيح كما يلي :

```
# includde < iostream. h>
main ()
{
   const int n = 9
   cout <<" the value of constant is : "<< n ;
   return 0 ;
}
```

- **المصفوفات والثوابت :**

يمكن وضع المتحولات الثابتة في أي مكان يمكن أن نضع فية تعبيراً ثابتاً ، فمثلاً يمكن استخدامها في تحديد حجم المصفوفة .

مثال :

```
contest int size = 10 ;

int s [size ] ;
```

تفيد التعليمات السابقة في تحديد حجم مصفوفة S باستخدام الثابت SIZE .

ويفيد استخدام المتحولات الثابتة لتحديد حجم المصفوفات في جعل البرامج أكثر قابلية لتغيير الحجم . فمثلاً حلقة FOR تقوم بتعبئة 10 عناصر يمكن تعديلها لتقوم بتعبئة 1000 عنصر وذلك بتغيير قيمة الثابت المرتبطة به أما في حالة عدم أستخدام الثوابت فيتطلب التعديل السابق عدة تعديلات في أماكن مختلفة من البرنامج .

مثال 1:

اكتب برنامج لطباعة عناصر مصفوفة .

```
# include< iostream.h>
main( )
{
  const int arrasize = 10 ;
  int a [arrasize];
  for (int i=0 ;i<arrasize ; i++)
  {
    a[i]=2+2*i;
```

```
    cout<<a[i]<<"\n";
  }
   return 0;
}
```

مثال 2 :

أكتب برنامج لحساب مجموع عناصر مصفوفة .

```
# include< iostream.h>
main( )
{
  const int arrasize = 10 ;
   int a [arrasize]={1,12,5,4,8,9,7,32,65,91};
   int sum =0;
   for (int I =0;i< arrasize ; i++)
       sum +=a[i];
   cout << "sum = "<<sum;
   return 0;
 }
```

- **مصفوفات الحروف:**

سوف نتعرض الان الى تخزين سلاسل الحروف فى مصفوفات من النمط Char حيث أن أي سلسلة حروف مثلا السلسلة "first" هى فى الواقع عبارة عن مصفوفة حروف . يمكن إعطاء

قيمة ابتدائية لمصفوفة حروف باستخدام سلاسل الحروف فعلى سبيل المثال يقوم التصريح بالشكل التالى

Char str 1[ ] =" first "

بإعطاء قيم ابتدائية لكل عنصر من عناصر المصفوفه str1 حيث يقابل كل منها احد حروف السلسلة "first" ويتحدد عدد عناصر المصفوفة str1 بواسطة المترجم وذلك حسب طول السلسلة المعطاة . من المهم أن نلاحظ أن السلسلة "first" تحتوى على خمسة حروف إضافة الى حرف خاص يحدد نهاية السلسلة وهو الحرف الصفري null character لذلك تتألف المصفوفة str1 من ستة عناصر ويتمم تمثيل الحرف الصفري على الشمل '0 \'. وهذا يعنى أن كافة الحروف تنتهى بالحرف الصغرى ويتم بالتالى التصريح عن المصفوفات التى تتعامل مع هذه السلاسل بحيث تكون ذات حجم كافي لتخظين حروفها إضافة الى الحرف الصفري

يمكن أيضا إعطاء قيم ابتدائية لمصفوفات الحروف باستخدام ثوابت الحروف المفردة ضمن قائمة للقيم الابتدائية . فمثلا يمكن كتابة التصريح السابق على الشكل التالى

```
char str1 [ ] ={' f ',' i ',' r ',' s ',' t ','\0'};
```

وعلى اعتبار ان سلاسل الحروف هي عبارة عن مصفوفات للحروف فيمكن الوصول الى كل حرف من حروفها بشكل منفصل مباشرة باستخدام ذليل عناصر المصفوفة فعلى سبيل المثال يمثل العنصر str1[0] الحرف 'f ويمثل الحرف 't العنصر str1 [4]

يمكن ايضا إدخال السلاسل مباشرة الى مصفوفات الحروف باستخدام لوحة المفاتيح وذلك بواسطة cin >> فمثلا التصريح التالى

Char str2 [10] ;

يقوم بإنشاء مصفوفة حروف قادرة عل تخزين سلسلة من 9 أحرف والحرف الصفري ايضا . وتمكن التعليمة التالية :

```
cin >>Str2;
```

على قراءة سلسلة من الحروف من لوحة المفاتيح وتخزينها فى str 2 أما التعليمة التالية

```
cout >> Str2 ;
```

فتساعد على طباعة المصفوفة str2

ملاحظة:

عند قراءة سلسلة حروف من لوحة المفاتيح لم يتم كتابة حجم المصفوفة وإنما فقط إسمها وبالتالى فى حالة عدم التزويد بمصفوفة ذات حجم كافي لاستيعاب الحروف المدخلة من قبل المستخدم بواسطة لوحة المفاتيح تؤدى الى ضياع فى معطيات البرنامج بالاضافة الى اخطاء التنفيذ علما أن cin يقوم بقراءة الحروف المدخلة حتى يصل الى فراغ ولا يهتم بحجم المصفوفة وكذلك الطباعة coutلاتهتم بحجم المصفوفة ويتم طباعة الحروف حتى الوصول الى الحرف الصفري .

مثال توضيحى :

```
# include <iostream.h >
main ( )
{
char str1[10],str2[]="first program";
cin>>str1;
cout<<"str1:"<<str1<<"\n"<<"str2:"<<str2<<"\n";
for(int i=0;str[i]!="\0";i++)
cout<<str[i]<<" ";
Return 0;
}
```

hello there

str 1 : hello

str 2 : first program

h e l l o

فرز المصفوفات :

تعتبر عملية فرز المعطيات (أي وضعها حسب ترتيب معين تصاعدي أو تنازلى مثلا) من أهم التطبيقات الحسابية وبالتالى سوف نقوم بشرح طريقة فرز تدعى بالفرز الفقاعي bubble sort او الفرز بالغوص sinking cort وذلك لان القيم الصغيرة تقوم تدريجيا بشق طريقها تصاعديا الى قمة المصفوفة بينما تقوم القيم الكبيرة بالغوص الى اسفل المصفوفة وتعتمد هذه الطريقة في الفرز على القيام بأكثر من مرور على العناصر وفى كل مرة يتم مقارنة زوجين متتالين من عناصر المصفوفة إذا كان هذان الزوجان مرتبين تصاعديا (أو لهما نفس القيمة) فأننا ندعهما على حالهما وإذا كان مرتين تنازليا فإننا نقوم بالمبادلة بينعما ضمن المصفوفة

يقوم البرنامج التالي بمقارنة العنصرين a[0] و a [1] ثم العنصرين a[1] و a [2] وهكذا حتى نهاية المصفوفه بمقارنة العنصرين a [8] و a [9] وعلى اعتبار أن المصفوفة تحتوى على عشرة عناصر فالبرنامج يقوم بتسع مقارنات تشق خلالها القيمة الكبرى طريقها الى الاسفل بينما تصعد القيمة الصغرى مكانا واحدا وهذا يعنى أن القيمة الكبرى سوف تصل الى الموضع a [9] بعد نهاية المرور الاول أما القيمة الكبرى الثانية سوف تصل الى الموضع

a[8] بعد نهاية المرور الثاني وهمذا حتى المرور التاسع حيث توضع القيمة التاسعة فى الموضع a[1] ويؤدى ذلك لبقاء القيمة الصغرى فى الموضع a[0] إذا نحتاج الى تسعة مرورات لفرز مصفوفة مولفة من عشر عناصر

تتم عملية الفرز من خلال بنية التكرار for المتداخلة وتجرى عملية المبادلة بين العناصر وفقا للتعليمات التالية

hold = a [i] ؛

```
a [i]  = a [i+ l] ;

a [i+ l] = hold ;
```

ونستخدم المتحول الاضافي Hold لتخزين إحدى القيمتين المراد مبادلتهما مؤقتا

```
a [ i ] = a [i+1];

a[ i +1 ] = a[ i];
```

فإذا كانت القيمة a[i] تساوى 10 وقيمة a[i+l] تساوى 8 فإن التعليمة الاولى تجعل قيمة العنصرين مساوية للقيمة العنصرين مساوية للقيمة 8 مما يسبب ضياعا للقيمة 10

```
#include < iomanip.h >
main ( )
{
const int size = 10 ;
int a[ size] ; int hold ;
for ( int i = 0; i< size ; i++)
{
cout << setw (5) << '' a['' << i <<''] ='' ;
cin >> a[ i] ;
cont << endi;
}
for ( int pass = 1 ; pass < size – pass ; i++ )
```

```
if ( a[ i ]> a [ i+ 1 ])
{
hold = a [ i ] ;
a [ i ] =a [ i+1 ] ;
 a[ i+1 ] = hold ;
}
for ( i=0 ; i < size ;i + + )
 cout < < setw (4) < < a[i] ;
return 0;
}
```

ملاحظة :

يتميز الفرز الفقاعي بسهولة البرمجة ولكنة أسلوب فرز بطيء وخصوصا مع المصفوفات الكبيرة

- المصفوفات المتعددة الأبعاد :

يمكن للمصفوفات فى لغة ++cأن تأخذ عدة أبعاد (بعدين وأكثر وصولا إلى 12 دليلا 9 ومن بين الاستخدامات الشائعة المصفوفات الثنائية أو الجداول التي تتألف من الأسطر والاعدمدة . وبالتالي للحصول على عنصر ما من بين العناصر يجب أن نحدد الدليلين : رقم السطر ورقم العمود الذي ينتمي لها العنصر . فمثلا إذا كان لدينا مصفوفة a مؤلفة من ثلاثة أسطر و أربعة أعمدة أي مصفوفة 3X4 فإننا نحدد كل عنصر من عناصر المصفوفة

بـ a [i] [j] حيث أن a اسم المصفوفة و i ، j هما الدليلان المحددان للعنصر المطلوب . حيث تأخذ عناصر السطر الأول القيمة صفر للدليل i أما عناصر العمود الأول فتأخذ

القيمة صفر للدليل j والتالي يمثل [0] [0] a العنصر الأول من السطر الأول والعمود الأول .

صفر إعطاء قيم ابتدائية لعناصر المصفوفة المتعدد الأبعاد بنفس أسلوب المصفوفات ذات البعد فمثلا يمكن إعطاء قيم ابتدائية للمصفوفة [2] [2] a بالشكل التالي

```
int  a [2] [2] ={ {2.4} ,{5.9} };
```

حيث يتم تجميع عناصر كل سطر ضمن قوسين كبيرين . مما يدب على أن القيم 2 و 4 هي قيم العنصرين [0] [0] a و [1] [0] a والقيم 5 و 9 هي [1] [0] a a [1] [1]

```
# include < iostream.h >
# include< iomanip.h >
main( )
{
const int  size 1 = 3 ;
const int  size 2=2;
int a [ size 1 ] [ size 2] ;
for ( int i=0 ; i < size 1 ; ++ )
for ( intj=0 ; j < size 2 ;j ++ )
{
cout < < setw ( 5 ) < < ''a[''< <'' ][''< <j< <'']='';
cin > > a[ i ] { j ] ;
cout < < endl ;
}
```

retum0 ;

في حالة كانت القيم الابتدائية غير كافية لعناصر السطر فانه يتم إعطاء القيمة صفر لبقية العناصر .

مثال : int a [2] [2] ={ {3} ,{4.6} };

يعطى العنصر [0] [0] a القيمة 3 [1] [0] a القيمة 6 أما العنصر[1] [0] a فتسند له قيمة الصفر من قبل المترجم

مثال

اكتب برنامج لقراءة عناصر مصفوفة ثنائية مدخلة من قبل المستخدم

تمارين عامة :

1- اكتب برنامج لقراءة صف ذو بعد واحد ثم طباعته على الشا شة

```
#inlude <iostream.h>
#inlude <iomanip.h>
main ( )
{
const int size =3;
```

```
int a[size ];
for ( int I = 0 : < size : i++ )
for (i=0;<size ;i++)
cout <<''a[''<<''a[''<<i<<'']=''<<a[i] <<setw(5);
retum 0;
}
```

2- اكتب برنامج لقراءة قيم صف ذو بعد واحد ثم احسب مجموع ومتوسط عناصر هذا الصف بالإضافة إلى اكبر واصغر عنصر

```
#inlude <iostream.h>
main ( )
const int size =3;
for ( int I = 0 : < size : i++ )
    cout <<''a[''<<i<<'']= ;
cin>>a[i];
}
int sum= 0,max =a[0],min=a[0];
for(i=0;i<size;i++)
{
```

```
sum+=a[i];
if(max<a[i])max=a[i];
if(min>a[i])min=a[i];
}
cout<<''sum is:''<<sum<< end1;
cout<<average is: ''<< sum/size << endl;
cout <<''max is:''<<max<<end1;
cout<<''min is :''<<min <<end1;
```

3- اكتب برنامج لقراءة عناصر صفين بعد أحادي ثم احسب مجموع هذين الصفين وحداؤهما

```
#inlude <iostream.h>
#inlude <iomanip.h>
main ( )
const int size =3;
{
int a[size],b[size],c[size];
{
cout<<''a[''<<i<<]='';
```

```
cin>>a[i];
}
for(i=0;i<size;i++)
{
cout <<"b["<<i<<"]=";
cin>>b[i];
}
for(i=0;i<size;i++)
{
c[i]=a[i]+b[i];
cout << setw(10) <<"c["<<"]="<<c[i];
mul+=a[i]*b[i];
cout<<setw(15)<<"mul is:"<<end1;
retum 0;
```

4- اكتب برنامج لقراءة قيم مصفوفة ذات بعدين ثم أطبع هذه القيم حسب الشكل الرياضي المتعارف علية

مثال :

9	5	1
3	7	4
2	6	8

```
#inlude <iostream.h>
#inlude <iomanip.h>
main ( )
const int size 1=3;
const int size 2=4;
int a[size1][size2];
for(int i=0 ;i<size1;i++)
   for(int j =0;j<size2;j++)
{
  cout <<''a[''<<i<<''][''<<j<<'']='';
cin>>a[i][j];
}

for(int j=0; j<size2 ;j++)
     cout << setw(5) <<a[i][j];
cout<<end1;
}
return 0;
}
```

5- أكتب برنامج لقراءة قيم مصفوفة مربعة ثم احسب مجموع عناصر القطر الرئيسي ومجموع عناصر القطر الثانوي

ملاحظة :

- عناصر القطر الرئيسي هي [j] [i] a حيث j = I

- عناصر القطر الثانوي [j] [i] a حيث i+ j = n- 1 ، n بعد المصفوفة .

الحل:

```
#inlude <iostream.h>
const int size =4;
main ( )
{
        int b [size][size];
        int I,j,sum1=0,sum2=0;
        for (i=0;i<size;i++)
        for(j=0;j<size;j++)
{
cout<<'' sum master : ''<< sum1<<end1;
cout<<''sum primary :''<<sum2<<end1;
return 0;
}
```

6- اكتب برنامج لقراءة قيم مصفوفة ذات بعدين ومن ثم قراءة قيمة عددية ما والتحقق من وجودها ضمن قيم المصفوفة أم لا .

```
#inlude <iostream.h>
const int size =3;
enum bool {true, false};
main( )
                    int b [size][size]; int I,j,x,
                    bool f =false ;
                   for(i=0;i<size;i++)
                      for(j=0;<size;j++)
                       {
                       cout <<''b[''<<i<<''][''<<j<<'']='';
                        cin >>b[i][i];
```

7- أكتب برنامج لحساب منقول مصفوفة ذات بعدين

```
#inlude <iostream.h>
#inlude <iomanip.h>
main ( )
{
const int size =3;
```

```
int b [size][size],c[size][size];int i , j ;
for(i=o ;i<size;i++)
for(j=0;<size;j++)
{
cout<< '' b [''<<i<<''][''<<j<<'']='';
cin>>b[i][j];
}
for(i=o;i<size;i++)
   for(j=o;j<size;j++)
      c[i][j]=b[j][i];
for(i=o;<size;i++)
```

برنامج لحساب منقول مصفوفة ذات بعدين

```
for(j[O;j<size;j++)
   cout <<c[i][j];<<setw (5);
cout <<end1;
}
return 0;
```

8- اكتب برنامج لعكس قيم صف a ذو بعد واحد جديد b أي يصبح أول عنصر من a آخر عنصر من b وهكذا .

مثال [1 , 5, 9 ,4 , 7] a ⟶ [7, 4 , 9 , 5 , 1] b

```
#inlude <iostream.h>
#inlude <iomanip.h>
main ( )
{
const int n =5;
int a[n], b [n];
for(int i=o;i<n;i++)
{
  cout <<''a[''<<i<<'']='';
cin>>a[i];
}
for(i=o;<n;i++)
{
   b[i]=a[ (n-1) –i];
cout <<''b[''<<i<<'']=''<<b[i] <<setw(5);
}
return 0;
```

9- أكتب برنامج للتحقق من تناظر مصفوفة مربعة .

```
#inlude <iostream.h>
#inlude <iomanip.h>
main ( )
{
const int n =3;
int a[n] [n];
 bool f=true;
for(inti=0;i<n;i++)
   for(int j=0; j;<n;j++)
{
   cout <<''a[''<<i<<''][''<<j<<'']='';
         }
    for( i0;<n;i++)
        for(int j=0; j;<n;j++)
            if(a[i][j]!=a[j][i])
                  f=false
    if(f= = true)
        cout <<''mathed'';
else
}
for (i=0;i<n;i++)
  for(int j=0; j<n ;j++)
```

```
        if(a[i][j]!a[i][j]
            f=false
if( f = = true )
    cout << ''mathed'';
else
   cout<<''no mathed'';
return 0;
}
```

10- نقول عن جملة أو عدد انه palindrome إذا أمكن قراءتها من البداية الى النهاية وبالعكس

مثال 12321 - 555 - 45554 - 121 - radar .....

أكتب برنامج يقوم بإدخال سلسلة (من الحروف أو من الأعداد الصحية) مؤلفة من خمس خانات كحد أقصى ويتحقق فيما إذا كان هذا العدد هو palindrome أو لا.

```
#inlude <iostream.h>
#inlude <iomanip.h>
main ( )
{
   const int n=5;
   char s [n];
   int i= o ;bool f = true;
   cin >>s;
```

```
   while(s[i]!=' 0')
{
   if(s[i]!=s[(n-1)-i])
        f=false;
    i=i+1;
}
if(f = = false )
    cout <<'' not palindrome';
else
    cout <<''palindrome '';
return0;
}
```