جَامعة
المَنارة
MANARA UNIVERSITY

كلية الهندسة المعلوماتية

مقرر برمجة_1

Dr.Bassam Hasan

محاضرات الأسبوع الأول + الثاني

الفصل الأول 2024-2025

المصفوفات Arrays

نوع معطيات يستخدم لتخزين قيم متعددة ضمن متحول واحد، نوع المعطيات هذا يسمى مصفوفة array ويعرف بأنه بنية معطيات مركبة.
وسنميز في بنية المعطيات المركبة هذه بين المصفوفات وحيدة البعد والمصفوفات متعددة الأبعاد.

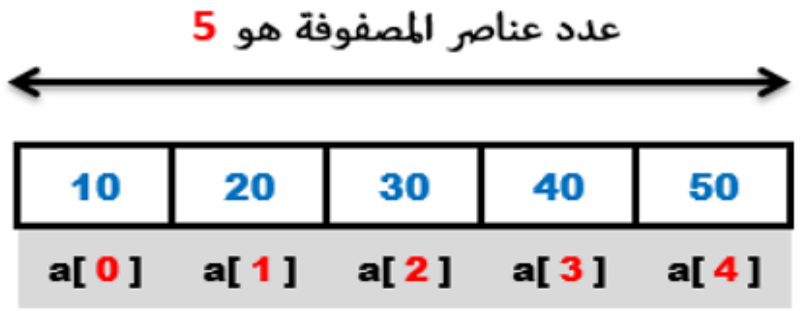
المصفوفات أحادية البعد(One dimensional array)

تسمى المصفوفة وحيدة البعد أو الأنساق ،لها نفس نوع البيانات ، تسمى عناصر النسق، ويتم ترقيمها بالتتابع من الصفر.

لنفترض الآن أننا قمنا بتعريف مصفوفة نوعها `int` , إسمها `a` , و تتألف من 5 عناصر.

```
int a[] = { 10, 20, 30, 40, 50 };
```

يمكنك تصور شكل المصفوفة `a` في الذاكرة كالتالي.



بما أن المصفوفة تتألف من 5 عناصر, تم إعطاء العناصر أرقام indexes بالترتيب من 0 إلى 4.

إذاً هنا أصبح عدد عناصر المصفوفة يساوي 5 و هو ثابت لا يمكن تغييره لاحقاً في الكود.

و للوصول لقيمة أى عنصر نستخدم index العنصر الذى تم إعطاؤه له.

الإعلان عن الصف Declaring array

يتم الإعلان عن الصف في لغة C++ بالشكل التالي :

Type array-name [size] ;

حيث:

type : نوع عناصر الصف .

size : عدد عناصر الصف .

array-name : اسم الصف .

double a[4] ;

يعلن عن صف اسمه a ، عدد عناصره 4 ، نوع عناصره حقيقي .

هذا ويمكن أن يعلن عن الصف السابق بالشكل :

```
const  int size = 4 ;
double  a[ size ] ;
```

حيث أعلن بداية عن size على أنه ثابت صحيح و قيمته 4 .

يمكن التصريح عن المصفوفة وتهيئتها في نفس الوقت على النحو التالي:

```
int a [5] = {10, 20, 30, 40, 50};
```

إذا هيّأت مصفوفة بسرد جميع عناصرها فلا يلزمك تضمين عدد عناصر المصفوفة داخل القوسين المعقوفين، إذ سيحسُب تلقائيًا. في المثال التالي، تعداد المصفوفة هو 5:

```
int array [] = {10, 20, 30, 40, 50};
```

كذلك نستطيع تهيئة العناصر الأولى فقط، مع تخصيص مساحة للمزيد من العناصر، وفي هذه الحالة يلزمك كتابة طول المصفوفة بين القوسين المعقوفين. حيث نخصص مصفوفةً خماسية (تحتوي 5 عناصر) مع تهيئتها جزئيًّا، سيُهيّئ بقية العناصر بالقيمة الافتراضية لنوع العنصر (في هذه الحالة، تلك القيمة هي 0).

int array [5] = {10,20};

10	20	0	0	0

إدخال و إخراج عناصر الصف
Input and output the array elements

يمكن إدخال عناصر الصف و تخزينها في ذاكرة الحاسب باستخدام أي من بنى التكرار، كذلك الأمر بالنسبة لعملية الإخراج أو الطباعة وبشكل عام يتم استخدام البنية التكرارية **for** .

ليكن لدينا الصف : int a [4] ;

يتم إدخال عناصر الصف :

```
for ( int i = 0 ; i < 4 ; i++ )
cin >> a[i] ;
```

أما الإخراج فيكون بالشكل :

```
for ( int i = 0 ; i < 4 ; i++ )
cout  >> a[i] ;
```

قراءة مصفوفة من لوحة المفاتيح

Example:

```
int marks [10];
    for(int i=0;i<10;i++)
     {
      cin>>marks[i];
     }
```

مثال اكتب برنامج لإدخال 5 ارقام وتخزينها في مصفوفة ثم يطبع عناصر المصفوفة

```
#include <iostream>
using namespace std;

int main()
{
    int numbers[5];
    cout << "Enter 5 numbers: " << endl;
      //  store input from user to array
     for (int i = 0; i < 5; ++i) {
           cin >> numbers[i];
      }
      cout << "The numbers are: ";
      //  print array elements
      for (int n = 0; n < 5; ++n) {
       cout << numbers[n] << "  ";
      }
  return 0;
}
```

Output

```
Enter 5 numbers:
11
12
13
14
15
The numbers are: 11  12  13  14  15
```

مثال

اكتب برنامج حساب متوسط عناصر مصفوفة.

يتم في البرنامج التالي حساب متوسط عناصر مصفوفة
يتم إدخالها من لوحة المفاتيح

```
#include <iostream>
using namespace std;
int main(){
const int Size = 12;
int a[Size];
doule ave,
Int sum=o;
for (int i = 0; i < Size ; i++)
{cin>>a[i];
sum += a[i];}
ave=sum / Size;
cout << " average values is " << ave <<endl;
return (0);}
```

فرز الصف Array sort

عملية فرز المعطيات (أي وضعها حسب ترتيب معين تصاعدي أو تنازلي)

```
#include <iostream>
using namespace std;
int main()
{const int n=5;
int x[n ]={10,8,6,4,2},temp;
for(   int i=1   ;   i < n ;  i++  )
{     for(int j=0   ;   j<n-1  ;   j++)
{            if(x[j]>x[j+1])

              {temp=x[j];
               x[j]=x[j+1];
              x[j+1]=temp;}
}
}
cout<<"the array after sorting:\n";

for(int i=0;i<n;i++)
cout<<x[i]<<" ";
return 0;
}
```

Output

data items in original order:
10 8 6 4 2
data items in ascending order:
2 4 6 8 10

الفرز الفقاعي Bubble sort

تعتمد طريقة الفرز الفقاعي على القيام بأكثر من مرور على عناصر النسق .

في كل مرور يتم مقارنة زوجين متتالين من عناصر النسق, إذا كان هذان الزوجان مرتبين تصاعدياً) أو متساويين (فإننا نبقيهما على حالهما,

أما إذا كانا مرتبين تنازلياً فإننا نقوم بالمبادلة بينهما ضمن النسق (فرز فقاعي تصاعدي)

مثال

أكتب برنامجاً يقوم بمايلي :

١- يطبع صفاً له القيم { 37 , 45 , 89 , 12 , 10 , 8 , 2 , 6 , 4 }

٢- يقوم بحساب مجموع أول خمس عناصر من الصف ومتوسط تلك العناصر

٣- طباعة الصف بعد ترتيب الخمس عناصر الأولى .

```
#include <iostream>
using namespace std;
int main() {
    const int n=10;
int a[n]={ 4 ,6 ,2 ,8 ,10 ,12 ,89 ,68 ,45 ,37 } ;
int temp,sum=0;float ave;
cout<<"the array is:"<<endl;
for(int i=0;i<n;i++)cout<<a[i]<<" ";
cout<<endl;

for(int i=0;i<5;i++)sum=sum+a[i];
ave=float(sum)/5;
cout<<"sum="<<sum<<"  and   "<<"  ave="<<ave;
cout<<endl;

for(int i=0 ;i<4;i++){
for(int j=1;j<5;j++)
{if(a[j]>a[j+1])   {temp=a[j];
                    a[j]=a[j+1];
                   a[j+1]=temp;}}}

cout<<"المصفوفة بعد ترتيب العناصر الخمسة الأولى تصاعدياً :"<<endl;
for(int i=0;i<n;i++)cout<<a[i]<<"    ";
cout<<endl;
return 0;
}
```

الخرج

```
the array is:
4 6 2 8 10 12 89 68 45 37
sum=30  and     ave=6
المصفوفة بعد ترتيب العناصر الخمسة الأولى تصاعدياً :
4    2    6    8    10    12    89    68    45    37
```

البحث ضمن الصف Search of an array

```
#include <iostream>
using namespace std;
int main() {
    const int n=5;
    int a[];
     int key;

cout<<"enter 5 value:";

for(int i=0;i<n;i++) cin>>a[i];

cout<<endl;
cout <<"enter the key to search for"<<endl;
cin>> key;
for(int i=0;i<n;i++)
  if(a[i]==key){cout<< "key is present";}

return 0;
}
```

```
enter 5 value:10 7 3 1 8

enter the key to search for
3
key is present
```

غالباً ما يتعامل المبرمج مع كميات كبيرة من المعطيات المخزنة في صفوف، ويصدف أن يحتاج إلى تحديد فيما إذا كان الصف يحتوي على قيمة تتطابق مع قيمة أخرى (القيمة المفتاح)، وعملية إيجاد قيمة ضمن صف مطابقة للقيمة المفتاح تسمى عملية البحث .

سنتطرق إلى طريقتين من طرق البحث هما :

البحث الخطي

البحث الثنائي.

أ- طريقة البحث الخطي Liner search

تعتمد طريقة البحث الخطي على مقارنة كل عنصر من عناصر الصف مع القيمة المفتاح التي نبحث عنها، وفي هذه الحالة يكون احتمال وجود العنصر في بداية الصف مساوياً لاحتمال وجوده في نهايته.

تعمل طريقة البحث الخطي بشكل جيد مع المصفوفات صغيرة الحجم وهي غير فعالة مع المصفوفات كبيرة الحجم .

البحث ضمن الصف Search of an array

طريقة البحث الثنائي Binary search

تعتمد طريقة البحث الثنائي على

1- يجب أن تكون المصفوفة مرتبة تصاعديا

2- تقسيم المصفوفة على 2

3- حذف نصف عدد عناصر النسق المرتب الذي نبحث ضمنه بعد كل عملية مقارنة,

4- حيث تقوم خوارزمية هذه الطريقة بمقارنة قيمة العنصر الواقع في منتصف النسق مع القيمة المفتاح التي نبحث عنها, فإذا تساوت القيمتان فهذا يعني أنه تم إيجاد القيمة التي نبحث عنها واذا لم تحدث المساواة فيتم إعادة صياغة المسألة لتصبح مكافئة لمسألة إيجاد عنصر ضمن أحد نصفي النسق السابق

5- إذا كانت القيمة المفتاح تقل عن قيمة العنصر الواقع في منتصف النسق فإنها تقع في النصف القيمة التي نبحث عنها تقع في النصف الأول من النسق وذلك على افتراض أن عناصر النسق مرتبة تصاعدياً.

إذا لم تكن القيمة المفتاح مساوية لقيمة العنصر الوسط في النسق الجزئي المختار نتيجة المرحلة الأولى فإن الخوارزمية تتكرر على أحد أرباع النسق الأصلي وهكذا.

في أسوأ الحالات تقوم خوارزمية البحث الثنائي بعشر مقارنات للبحث عن عنصر ضمن مصفوفة مؤلفة من 1024 عنصرًا, إذ أن التقسيم المتكرر على 2 يحذف نصف عدد العناصر بعد كل عملية مقارنة .

(1 , 2 , 4 , 8 , 16 , 32 , 64 , 128 ,256 , 512)

Search of an array البحث ضمن الصف

Binary search طريقة البحث الثنائي

```
#include <iostream>
using namespace std;
int main()
{  const int n=8;
int target;
int a[n]={22,33,44,55,66,77,88,99};
cout<< "enter the value you search for:";
cin>>target;                    العدد الذي تبحث عنه
int location , left=0, right=n-1; int found=0;

while(!found && left<=right){
location=( left + right)/2;
found=(a[location]==target);
if(a[location]<target)
left=location+1;
else right=location-1; }

if(found)cout<<target<<"is at:"<<location<<"\n";
if(!found)cout<<target<<"is not found.\n";
   return 0;}
```

انتبه : المصفوفة مرتيه تصاعدياً

يدل على موقع العنصر الذي ستبحث عنه في المصفوفة المؤشر الأوسط: location

يدل على موقع العنصر الاول في المصفوفة: left

يدل على موقع العنصر الاخير في المصفوفة: right

المصفوفات متعددة الأبعاد

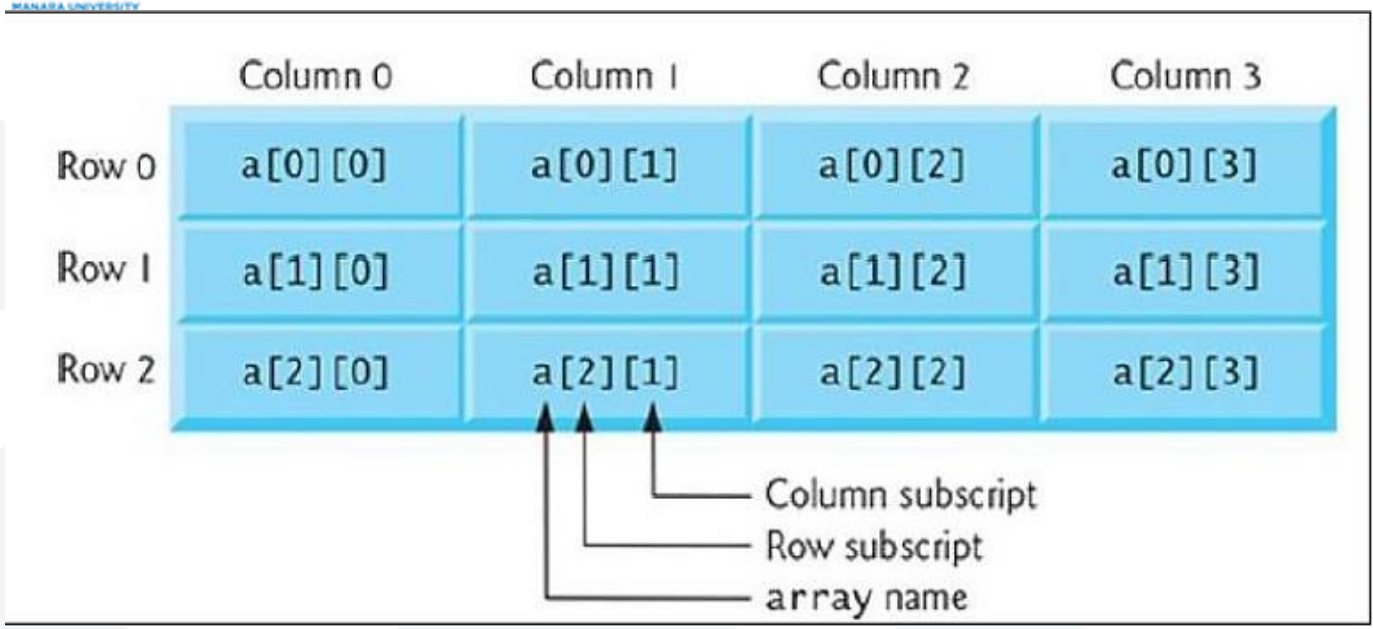
يتم إدخال عناصرمصفوفة ثنائية البعد a[4][3] الذي بالشكل باستخدام الحلقات التكرارية

```
for ( int i = 0 ; i < 4 ; i++ ){
      for(int j=0; j< 3 ; j++){
      cin >> a[i] [j] ;}
}
```

طباعة عناصر مصفوفة ثنائية البعد

يتم طباعة عناصرمصفوفة ثنائية البعد a[4][3] باستخدام الحلقات التكرارية

```
for ( int i = 0 ; i < 4 ; i++ ){
      for(int j=0; j< 3 ; j++){
      cout >> a[i] [j] ;}
}
```



الإعلان عن المصفوفة متعددة الأبعاد

double a[3] 4] ;

يعرف مصفوفة ذات بعدين عدد أسطرها 3 وعدد أعمدتها 4

المصفوفات متعددة الأبعاد

مثال اكتب برنامج لطباعة عناصر مصفوفة اعداد صحيحة ثنائية البعد مؤلفة من سطرين وثلاث اعمدة

```
#include <iostream>
using namespace std;
int main()
{int x[2][3]={{2,3,4},{8,9,10}};
for(int i= 0;i<2;i++)
{for(int j= 0;j<3;j++)
  {cout<<x[i][j]<<"  ";}
cout<<endl;}
return 0;}
```

الخرج

```
2  3  4
8  9  10
```

ملاحظة

```
#include <iostream>
using namespace std;
int main()
{int x[2][3]={{2,3,4},{8,9,10}};
for(int i= 0;i<2;i++)
{for(int j= 0;j<3;j++)
  cout<<x[i][j]<<"  ";
}
return 0;}
```

الخرج

```
2  3  4  8  9  10
```

المصفوفات متعددة الأبعاد

```
#include <iostream>
using namespace std;
int main()
{
int x[3][5];
 for(int i=0;i<3;i++){
      for(int j=0;j<5;j++)
      cin>>x[i][j];}
for(int i=0;i<3;i++) {
    for(int j=0;j<5;j++){
   cout<<x[i][j]<<" "; }
cout<<endl; }
int sum=0;
 for(int i=0;i<3;i++) {
      for(int j=0;j<5;j++) {
       sum=sum+x[i][j]; } }
  cout<<"sum="<<sum<<endl;
 cout<<"the elements which column index is greater than row index
:"<<endl;
 for(int i=0;i<3;i++) {
  for(int j=0;j<5;j++) {
      if(j>i)
      cout<<x[i][j]<<" " ;
      else
      cout<<'0'<<"  "; }
  cout<<endl; }
  return 0; }
```

مثال

اكتب برنامج يقوم بإدخال مصفوفة ثنائية من الاعداد الصحيحة حجمها 3 x5و المطلوب:

1 - طباعة المصفوفة .

2 - طباعة مجموع عناصر المصفوفة .

3 - طباعة العناصر التي ًكون دليل العمود فيها أكبرمن دليل السطر

مثال

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{int const n_rows = 3;
   int const n_cols = 7;
   int const m[n_rows][n_cols] = {
      { 1, 2, 3, 4, 5, 6, 7 },
      { 8, 9, 10, 11, 12, 13, 14 },
      { 15, 16, 17, 18, 19, 20, 21 },
   }
   for( int y = 0; y < n_rows; ++y )
   {
      for( int x = 0; x < n_cols; ++x )

      {cout << setw( 4 ) << m[y][x];}
      cout << '\n';
   }
return 0;}
```

الخرج

```
   1   2   3   4   5   6   7
   8   9  10  11  12  13  14
  15  16  17  18  19  20  21
```