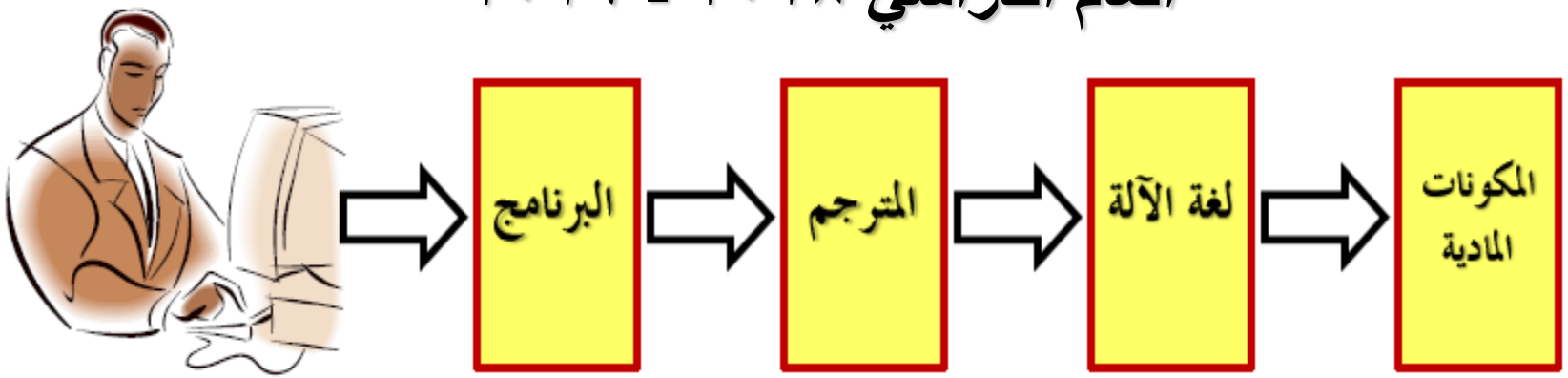
جامعة حماة
كلية العلوم في مصياف / السنة الأولى

المادة: البرمجة و الخوارزميات

المحاضرة الخامسة: المصفوفات في لغة البرمجة C++

Arrays in C++

العام الدراسي ٢٠١٨ - ٢٠١٩



مقدمة عن المصفوفات في لغة Arrays in C++

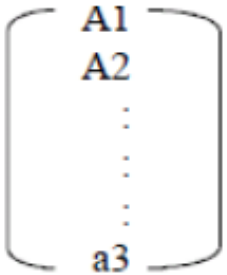
- إن طرق التعامل مع أسماء المتغيرات و الثوابت العددية و الرمزية،التي درسناها سابقاً، تعد صالحة للتعامل مع عدد محدود من هذه الثوابت و المتغيرات، سواء في عمليات الإدخال و الإخراج أو في العمليات الحسابية والمنطقية، وعندما يصبح عدد المتغيرات كبيرا جدا،تصبح تلك الطرق غير عملية، فمثلا لو أردنا إدخال مائة قيمة لمتغيرات البرنامج من x1 إلى X100....x2, ، فكم الحيز المطلوب لعمليات الإدخال و الإخراج و العمليات الحسابية و المنطقية لهذه المتغيرات يصبح كبيراً جداً؟
- من جهة أخرى فإننا نوفر مخزناً خاصاً لكل متغير نتعامل معه، أثناء تنفيذ البرنامج، و ذلك لحفظ قيمته في مخزن، و من ثم لاستعمال قيمته في عمليات أخرى تالية، ومن ناحية ثالثة، فان من الصعوبة بمكان، بل من المستحيل استعمال اسم المتغير العددي و الرمزي كمصفوفة ذات بعدين، و ثلاثة أبعاد...الخ.
- للأسباب الثلاثة الواردة أعلاه، جاءت فكرة استعمال متغير جماعي يضم تحت اسمه و يتم ترقميه، subscripted variable عددا من العناصر يسمى بالمتغير الرقمي وقد نسمّيه، subscript بين قوسين مربعين [ ] يوضع بينهما قيمة العداد المرقم أحيانا، ويمكننا تشبيه المتغير المرقم بمقسم الهاتف لمؤسسة ما، فهو الدليل index لمقسم واحد، تنضم تحته عدد من الأرقام الفرعية للموظفين وكل رقم من هذه الأرقام مستقل ومتميز عن الأرقام الفرعية الأخرى، و له مخزن خاص في الذاكرة، الآن انه كغيره من الأرقام الفرعية تابع للرقم العام لمقسم المؤسسة.

المصفوفات في لغة Arrays in C++

- مثلا العناصر التالية :(من اليمين إلى اليسار):

a[n] ...a[2], a[1], a[0]

- تابع للمتغير الجماعي [ ]a: فالعنوان الأول يكون address للعنصر الأول و كل عنصر من هذه العناصر له عنوان في الذاكرة والثاني للثاني والثالث للثالث ... وهكذا.
- يستعمل المتغير الجماعي [المرقم] أو المصفوفة، في لغة C++ حجز جماعي مسبق في الذاكرة لجميع عناصره، فلو كان يتبعه خمسون عنصرا، فانه يحجز له ٥٠ مخزنا، على الأقل في الذاكرة .
- من الفوائد المهمة للمتغيرات المرقمة والمصفوفات: هو استعمالها في الترتيب التصاعدي والتنازلي للعناصر والقيم المختلفة، وعمليات ترتيب الأسماء الأبجدي النصوص الرمزية، وفي عمليات ضرب المصفوفات، و إيجاد معكوس المصفوفة وعملياتها الأخرى، وفي التحليل العددي ... الخ.
- المتغير المرقم (المصفوفة) ذو البعد الواحد one-dimensional Array: ويمثل المتغير المرقم ذو البعد الواحد هو مصفوفة ذات بعد واحد أو متجه vector في الجبر على النحو الأفقي [a1 a2 ...a3] أو العمودي



١١

المصفوفات في لغة Arrays in C++

ويأخذ المرقم المتغير في c++ الشكل العام التالي:

```
Type-specifier array-name[size];
```

- يبدأ العداد المرقم عادة من الصفر، آي العنصر الأول من المصفوفة وهكذا فمثلا المصفوفة التالية:

```
Int a[20];
```

اسمها a[ ] و العنصر الأول a[0] والثاني a[1] وقد حجز لها ٢٠ موقعا لعشرين عنصرا من النوع الصحيح.

- والمصفوفة التالية: مصفوفة رمزية ، اسمها name يحجز لها خمسة عشر عنصرا من النوع الرمزي.

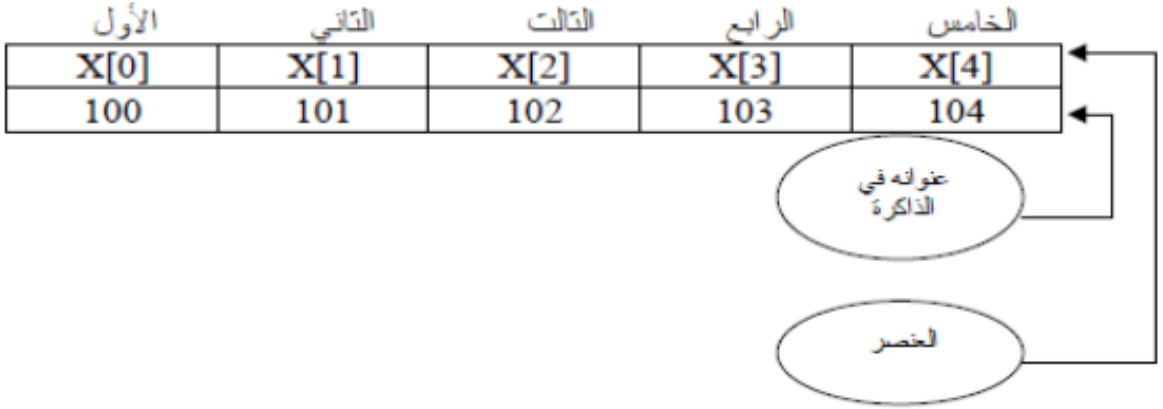
```
Char name[15];
```

عنوان عناصر المصفوفة في الذاكرة
Array Elements in Memory Addressing

- ذكرنا من قبل أن آي متغير أو عنصر من متغير ذاتي مرقم ، يحتل موقعا من الذاكرة يستعمل عادة مؤشرا لكل متغير أو عنصر، ليكون دليلا على استعمال هذه المتغيرات و العناصر بسهولة ويسر، و المثال التالي يوضح هذه العملية بالنسبة للمصفوفة ذات بعد واحد.

Int x[5];

- يمكن تمثيل عناصر المصفوفة المعلن عنها، مع عناوينها بالشكل التوضيحي xالتالي (من اليسار إلى اليمين)



- إذا فرضنا أن عنوان موقع العنصر الأول x[0] في الذاكرة هو ١٠٠ ، فان عناوين العناصر الأخرى تكون على التوالي ١٠١ و ١٠٢ و ١٠٣ و ١٠٤.
- يمكن تشبيه العلاقة بين قيمة العنصر وعنوانه بالعلاقة بين علامة طالب و رقمه الجامعي ، إذ علامته هي قيمة نشطه كعنصر، ليس لها علاقة برقم مقعده الجامعي.

١٨

المصفوفات أحادية البعد Array في لغة البرمجة C++

- **المصفوفة ذات البعد الواحد**: هي بنية معطيات نظامية متجانسة تكون جميع عناصرها من نوع قاعدي واحد مرّكب يعرّف في بداية البرنامج، وتتميز بإمكانية الوصول المباشر إلى أي عنصر فيها دون المرور ببقية العناصر الأخرى و ذلك بمساعدة الفهرس Index الذي يحدد موقع العنصر الذي نرغب بتنفيذ العمليات عليه ضمن المصفوفة x[i]، ويتم التعريف عنها على الشكل التالي في بداية البرنامج:

Type *arrrayname*[size of array]

تعريف المصفوفة ذات البعد واحد

- ***Arrrayname:*** اسم المصفوفة الذي سنتعامل معه في البرنامج أي اسم ممكن.
- **size of array: الحجم** الذي ستشغله المصفوفة في الذاكرة وقد يكون أي رقم حسب احتياجك
- **Type:** نوع عناصر المصفوفة التي سوف نعرفها قد تكون حرفية أو رقمية. إذا عرّفنا مصفوفة من نوع integer فإن جميع عناصرها تكون integer ولا يجوز تخزين أحرف في داخلها .

- مثال: تعريف مصفوفة أحادية البعد فيها خمس أعداد صحيحة: **int arr[5];**
- المصفوفات: هي مجموعة خلايا متتالية في الذاكرة تحجز لغرض خزن معلومات معينة في داخلها كأن نخزن في داخلها أرقام أو أحرف وتبقى هذه القيم المخزنة داخل المصفوفة حتى نغلق البرنامج.
- يجب الإعلان عن عدد المواقع التي نحتاجها في العمل في بداية البرنامج حتى يحجزها المترجم للمصفوفة ولا يخزن قيم أخرى في داخلها و تبقى محجوزة فقط لعناصر المصفوفة. ويكون الإعلان عليها هكذا

مثال: برنامج يطبع درجات طالب في ستة مقررات مدخلة بشكل مباشر.

- تعريف المكتبات التي سيستخدمها البرنامج
- بداية الدّالة الرئيسية
- التعريف بالمصفوفة وإدخال عناصر المصفوفة بشكل مباشر.
- طباعة عناصر المصفوفة.
- نهاية البرنامج.

```
##include <iostream>
using namespace std;
void main ()
{
int a[6]={40,60,50,70,80,90};
for(int I=0;I<6;I++)
        cout<<a[I]<<endl;
}
```

```
40
60
50
70
80
90
```

مثال: برنامج يقرأ ١٠ أرقام، ثم يقوم بطباعتها بالعكس

- تعريف المكتبات التي سيستخدمها البرنامج
- بداية الدّالة الرئيسية
- التعريف بالمصفوفة
- إدخال عناصر المصفوفة
- طباعة عناصر المصفوفة عكسياً
- نهاية البرنامج

```
#include <iostream>
using namespace std;
void main()
{
int x[10];
    for(int i=0;i<10;i++)
    cin>>x[i];
    for(int i=10;i>0;i--)
    cout<<x[i-1];
}
```

مثال: برنامج يطبع معدل ومجموع درجات طالب في خمسة مقررات مدخلة بشكل مباشر.

- تعريف المكتبات التي سيستخدمها البرنامج
- بداية الدّالة الرئيسية
- التعريف بالمصفوفة وإدخال عناصرها
- حساب مجموع الدرجات
- حساب متوسط درجات الطالب
- طباعة مجموع الدرجات
- طباعة متوسط الدرجات
- نهاية البرنامج

```
#include <iostream>
using namespace std;
main ()
{
int a[5]={87,67,81,90,55};
int s = 0;  float avg = 0.0 ;
for(int i=0;i<5;i++)
s+ = a[i];
avg=s/5;
cout<<"Average="<<avg<<endl;
cout <<"Sum=" << s <<endl;
}
```

```
Average= 76
Sum=380
```

مثال: مصفوفتين A[10] و B[10] من الأعداد الحقيقية. المطلوب:

١. إدخال عناصر المصفوفتين A و B .
٢. طباعة عناصر المصفوفة C التي يكون عناصرها ناتج طرح العناصر المتقابلة بالفهرس في المصفوفتين A و B على سطر واحد.
٣. طباعة مجموع SUMعناصر المصفوفة C على سطر جديد.
٤. طباعة متوسط Avg عناصر المصفوفة C على سطر جديد.

```
#include <iostream>

using namespace std;

void main()

{   float Sum=0.0, Avg=0.0;

float A[10],B[10],C[10];

 for ( int i=0;i<10;i++)  {

cout<<" Input  A [ "<<i<<" ] ";

cin>>A[i];

cout<<" Input  B [ " <<i<<" ] ";

cin>>B[i];

C[i]=A[i]-B[i];

 Sum=Sum+C[i];  }

Avg=Sum/10;

cout<<“C[“<<10<<“]=“;

for(int i=0;i <10;i++)

cout<<C[i]<<“ , “;

cout<<"\nSum ="<<Sum<<endl;

cout<<"\nAvg ="<<Avg<<endl;

}
```

مثال: أوجد العنصر الأصغر MIN في المصفوفة التالية

```
#include <iostream>
using namespace std;
void main()
{
  int x[10];
  int min;
    for(int i=0;i<10;i++) {
  cout<<"\nx["<<i<<"]=";
  cin>>x[i];  }
    min=x[0];
for(int i=0;i<10;i++)
    if (x[i]<min)
    min=x[i];
  cout<< "MIN="<<min;
}
```

تعريف المصفوفة والمتغيرات

إدخال عناصر المصفوفة

إيجاد العنصر الأصغر

طباعة العنصر الأصغر

مثال: طباعة عناصر المصفوفة و العنصر الأكبر MAX

```
#include <iostream>
using namespace std;
void main()
{
  float A[100];
  float max;
  for(int i=0;i<100;i++) {
  cout<<"\nA["<<i<<"]=";
  cin>>A[i];  }
  max=x[0];
for(int i=0;i<100;i++)
  if (A[i]>max)
  max=A[i];
  for(int i=0;i<100;i++)
  cout<<A[i]<<", ";
cout<< "\nMAX="<<max;
}
```

تعريف المصفوفة والمتغيرات

إدخال عناصر المصفوفة

إيجاد العنصر الأكبر

طباعة عناصر المصفوفة .

طباعة العنصر الأكبر

مثال: رتّب واطبع عناصر المصفوفة التالية تصاعدياً.

```
#include <iostream>
using namespace std;
void main()
{
  float Arr[200];
  float min,temp;
  for(int i=0;i<200;i++)  {
cout<<"\nArr["<<i<<"]=";
  cin>>Arr[i];  }
    min=x[0];
for(int i=0;i<200;i++)
    if (Arr[i]<min) {
          temp=Arr[i];
          Arr[i]=min;
          min=temp;  }
  for(int i=0;i<200;i++)
    cout<<Arr[i]<<", ";
}
```

تعريف المصفوفة والمتغيرات

إدخال عناصر المصفوفة

إيجاد العنصر الأصغر واستبدال مكانه

طباعة عناصر المصفوفة مرّتبة تصاعديا

مثال على المصفوفات ذات البعد الواحد و الدوال

حساب المتوسط عن طريق الدوال

```
#include <iostream>
using namespace std;
float Avg ();
void main ()
{
int a[5]={87,67,81,90,55};
cout<<“Average=“<<Avg(a,5);
}
```

```
float Avg (int x[ ], int n)
{
int s = 0;
float avg=0.0;
for(int i=0;i<n;i++)
 s+= x[i];
avg=s/5;
return avg ;
}
```

مثال على المصفوفات و الدوال

طباعة عناصر مصفوفة بواسطة الدوال ذات المتغيرات

```
#include <iostream>
using namespace std;
void printarr (int arr[ ], int size);
void main ()
{
int A[5]={87,67,81,90,55};
int B[7]={22,80,60,21,90,66,99};
printarr(A,5);
printarr(B,7);
}
```

```
void printarr (int arr[ ], int size);
{
    for(int i=0,i<size,i++)
    {
    cout<<arr[i]<<endl;
    }
}
```

مثال على المصفوفات و الدوال

إدخال وطباعة عناصر مصفوفة ذات بعد واحد بواسطة الدوال ذات المتغيرات

```
#include <iostream>
#include <conio>
using namespace std;
int inarr (int arr[ ] , int s1);
void prarr (int arr[ ] , int s1);
int main ()   {
const n=5 ;
Int A [n] ;
inarr ( &A, &n );
clscr() ;
prarr ( A, n );
return0;
}
```

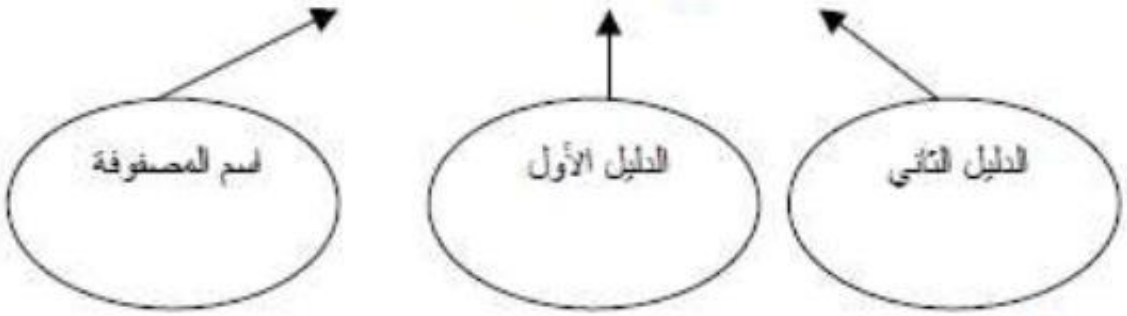
```
int inarr (&int arr[ ] , &int s1 )
{
   for (int i=0, i<s1, i++)
  {  cout << " \n arr [" << i << " ] = " ;
     cin >> arr [ i ] ;   }
return (arr [ ]);
}

void prarr ( int arr[ ], int s1 )
{
   for (int i=0, i<s1, i++)
   cout << arr [ i ] << " , " ;
}
```

المصفوفة ذات البعدين في C++
Two-Dimensional Arrays

- تشبه المصفوفة ذات البعدين في طريقة تعاملها المصفوفة ذات البعد الواحد إلا أن لها عدادين (index 2) دليلين أو مرقمين إحداهما عداد للصفوف والآخر عداد للأعمدة ويأخذ الإعلان عن المصفوفة الشكل العام التالي:

```
Type-specifier arraey_name [index 1][index 2];
```



- فمثلا المصفوفة :

```
Int x[2][3];
```

int هي مصفوفة صحيحة العناصر أبعادها هي عدد الصفوف= ٢، وعدد الأعمدة = ٣

لاحظ أن عدد الصفوف يوضع بين قوسين وحده ، وكذلك عداد الأعمدة .

المصفوفات ذات البعدين Array في لغة البرمجة C++

- **المصفوفة ذات البعدين**: هي بنية معطيات نظامية متجانسة تكون جميع عناصرها من نوع قاعدي واحد مرّكب يعرّف في بداية البرنامج، وتتميز بإمكانية الوصول المباشر إلى أي عنصر فيها دون المرور ببقية العناصر الأخرى و ذلك بمساعدة الأدلة (الفهارس) Index's التي تحدد موقع العنصر (رقم السطر والعمود) الذي نرغب بتنفيذ العمليات عليه ضمن المصفوفة x[i] [j]، ويتم التعريف عنه في بداية البرنامج:

تعريف المصفوفة ذات البعد ين

Type of array *arrrayname*[row][columns]

- ***Arrrayname:*** اسم المصفوفة الذي سنتعامل معه في البرنامج أي اسم ممكن.
- **row:** عدد سطور المصفوفة (البعد الثاني)
- **Columns:** عدد أعمدة المصفوفة (لأننا نتعامل مع مصفوفة ثنائية الإبعاد)، (البعد الأول).
- **Type:** نوع عناصر المصفوفة التي سوف نعرفها قد تكون حرفية أو رقمية.

- يجب الإعلان عن عدد المواقع التي نحتاجها في العمل في بداية البرنامج حتى يحجزها المترجم للمصفوفة ولا يخزن قيم أخرى في داخلها و تبقى محجوزة فقط لعناصر المصفوفة. ويكون الإعلان عنها هكذا:
- مثال: تعريف مصفوفة ثنائية الأبعاد فيها خمسون عدد صحيح (خمسة سطور و عشرة أعمدة):
- **int arr[5][10};**
- نستخدم في المصفوفات ذات البعدين حلقتين متداخلتين ليتم قراءة البعد الأول i والبعد الثاني j .

مثال١ : طباعة عناصر القطر الرئيسي في أي مصفوفة هو ثابت **(i==j)**

مثال٢ : طباعة عناصر القطر الثانوي في أي مصفوفة هو ثابت **(i+j=size)**

مثال١

```
#include <iostream>
using namespace std;
void main ()
{
Int a[5][5]={11,15,20,30,12,27,44,13,33,27,
             44,55,22,11,10,45,22,18,22,50,
             33,34,22,40,32};
for(i=0;i<5;i++) {
        for(j=0;j<5;j++)
        if(i==j)
        cout<<a[i][j]<<"\n";  }
}
```

مثال٢

```
#include <iostream>
using namespace std;
void main ()
{
int M[5][5];
for(int i=0;i<5;i++) {
for(int j=0;j<5;j++)
cin>>M[i][j]; }
for(i=0;i<5;i++) {
for(j=0;j<5;j++)
if(i+j=4)
cout<<M[i][j]<<"\n";
}}
```

القطر الرئيسي

(0,0)	(0,1)	(0,2)	(0,3)	(0,4)
(1,0)	(1,1)	(1,2)	(1,3)	(1,4)
(2,0)	(2,1)	(2,2)	(2,3)	(2,4)
(3,0)	(3,1)	(3,2)	(3,3)	(3,4)
(4,0)	(4,1)	(4,2)	(4,3)	(4,4)

القطر الثانوى

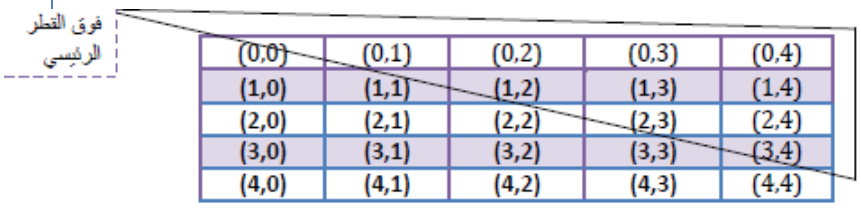
(0,0)	(0,1)	(0,2)	(0,3)	(0,4)
(1,0)	(1,1)	(1,2)	(1,3)	(1,4)
(2,0)	(2,1)	(2,2)	(2,3)	(2,4)
(3,0)	(3,1)	(3,2)	(3,3)	(3,4)
(4,0)	(4,1)	(4,2)	(4,3)	(4,4)

مثال٣: طباعة العناصر الواقعة فوق القطر الرئيسي في أي مصفوفة هو ثابت **(i<j)**

مثال٤: طباعة العناصر الواقعة تحت القطر الثانوي في أي مصفوفة هو ثابت **(i>j)**

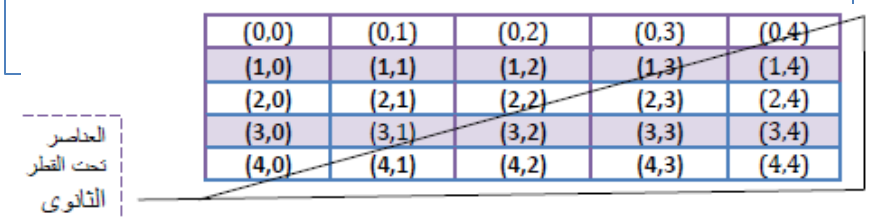
مثال ٣

```
#include <iostream>
using namespace std;
void main ()
{
int a[5][5];
for(int i=0;i<5;i++) {
for(int j=0;j<5;j++)
cin>>a[i][j]; }
for(i=0;i<5;i++) {
for(j=0;j<5;j++)
if(i<j)
cout<<a[i][j]<<"\n";
}}
```



مثال ٤

```
#include <iostream>
using namespace std;
void main ()
{
int a[5][5];
for(int i=0;i<5;i++) {
for(int j=0;j<5;j++)
cin>>a[i][j]; }
for(i=0;i<5;i++) {
for(j=0;j<5;j++)
if(i+j>4)
cout<<a[i][j]<<"\n";
}}
```



مثال ٥ على المصفوفات

برنامج يقوم بإدخال وطباعة ثلاث علامات لخمس طلاب :

- بالنسبة لعناوين العناصر المصفوفة متعددة الأبعاد في الذاكرة، لا يختلف عما ذكرنا بالنسبة للمصفوفات ذات البعد الواحد، ولذلك لو فرضنا ، في المثال السابق أن العنصر x[0,0] كان عنوانه ١٠٠ مثلا فان عناوين العناصر التالية: حسب ترتيبها المذكور أعلاه هي ١٠٠ ـ١٠١ـ ١٠٢ لعناصر الصف الأول و ١٠٣ـ١٠٤ـ ١٠٥ لعناصر الصف الثاني.

```
#include <iostream>
Using namespace std;
void main ()
{
int a,b,m[5][3];
for(a=0;a<5;a++)  {
            for(b=0;b>3;b++)
            { cin>>m[a][b]; }
                        }
for(a=0;a<5;a++)   {
            for(b=0;b>3;b++)
            { cout<<m[a[b]; }
}
}
```

مثال ٦ على المصفوفات ذات البعدين و الدوال

طباعة متوسط عناصر مصفوفة ذات بعدين بواسطة الدوال

```
#include <iostream>
#include <conio>
using namespace std;
float AV(int x[ ] [ ])
{
float avg=0.0;
for(int I=0;I<5;I++)
{  for(int j=0;j<7;j++)
s+= X[I][j];
 }
avg=s/(7*5);
return avg ;
}
```

```
void main ()
{
int X[5][7];
for(int I=0;I<5;I++)
 {
   for(int j=0;j<7;j++)
   cin>>X[I][j];
  }
clscr();
cout <<“Average=“<< AV(x,7);
}
```