المحاضرة السابعة
المؤشرات والمراجع
Pointers and References

المؤشرات Pointers

مفهوم المؤشرات

- تعد المؤشرات من أهم وأقوى الخصائص المتوفرة في لغة C++ وهي التي تميز هذه اللغة عن باقي لغات البرمجة.
- في الوقت نفسه تعتبر المؤشرات من أصعب المواضيع في لغة C++ من حيث الاستخدام والسيطرة. حيث تتطلب من المبرمج حضور الذهن والمتابعة الدقيقة للمؤشرات المستخدمة في برامجه كما تحتاج المؤشرات لدقة عند التعامل معها.
- تكمن فائدة استخدام المؤشرات في البرمجة عموماً في:

a. إمكانية التعامل مع المتغيرات بالعناوين أي عناوينها في الذاكرة بشكل مباشر.

b. الوصول إلى عناصر المصفوفة .

c. تمرير المعاملات إلى دالة باستخدام العنوان (Passing By Reference) .

d. إرسال المصفوفة أو سلاسل الحروف إلى الدوال .

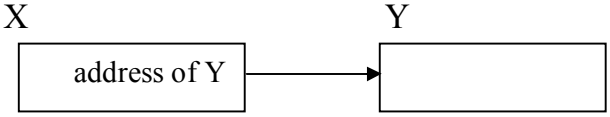
e. التحكم بمساحة اكبر من الذاكرة.

<u>أو يمكن القول باختصار</u>

أن المؤشرات تستخدم في البرامج التي نحتاج فيها إلى حجز وإلغاء حجز الذاكرة أثناء تنفيذ البرنامج

تعريف المؤشر :-

هو عبارة عن موقع في الذاكرة يحمل عنوان موقع آخر وبالتالي يمكن أن نقول أن الموقع الأول عبارة عن مؤشر يؤشر إلى الموقع الآخر .



X تؤشر إلى Y

الإعلان عن المؤشرات

- مثل بقية المتغيرات يجب التصريح عن المؤشرات قبل استخدامها، ويتم التصريح عن المؤشر بوضع إشارة * قبل اسم المؤشر.
- يمكن أن يتم التصريح عن المؤشر ليؤشر على قيمة أو متغير من أي نوع من أنواع البيانات الأساسية في لغة C++

الصيغة العامة للإعلان عن المؤشر

data type *pointer name ;

حيث :-

data type هي نوع البيانات التي يشير إليها المؤشر , pointer name هو اسم المؤشر .

أمثلة :

1. int * p;

إعلان عن مؤشر يشير إلى متغير من النوع الصحيح

2. int i,j,a[10],b[2],*p,*q;

إعلان عن مجموعة من المتغيرات والمصفوفات والمؤشرات من النوع الصحيح

3. char *r;

إعلان عن مؤشر يشير إلى متغير من النوع الحرفي

4. float *m, *n

إعلان عن مؤشرين باسم m,n ليؤشرا على قيمتين حقيقيتين.

☑	يفضل إعطاء قيمة ابتدائية للمؤشر، وعادة ما يتم إعطاء المؤشرات القيمة الابتدائية NULL والتي تعني أن يؤشر المؤشر على "لا شيء". والقيمة NULL يمكن إسنادها إلى أي نوع من أنواع المؤشرات. وبالطبع يمكن إسناد القيمة الابتدائية NULL إلى المؤشر أثناء التصريح كما هو الحال مع باقي أنواع المتغيرات كما في المثال التالي: `int *ptr=NULL;`
☑	كما تستخدم القيمة NULL لأغراض أخرى في البرنامج فمثلاً لاختبار فيما إذا كان المؤشر ptr يؤشر على قيمة فعلية أو أنه يؤشر على "لا شئ" نستخدم عبارة الشرط التالية: `if (ptr!=NULL) ...`
☑	عند إسناد قيمة أو متغير إلى مؤشر يجب أن تكون هذه القيمة أو المتغير من نفس نوع بيانات المؤشر وغير ذلك يؤدي إلى خطأ قواعدي في البرنامج

معامل العنوان (&) ومعامل المحتوى (*)

مع المؤشرات يتم استخدام أداتان مهمتان في لغة C++ وهما

1. الأداة **&** والتي تستخدم لإعطاء عنوان متغير في الذاكرة على سبيل المثال

```
mem=&x;
```

فإذا كان المتغير x واقعاً في الموقع 1000 من الذاكرة وكانت قيمة x في الموقع هي 3 فإن جملة التعيين السابقة تعطي المتغير mem في موقعه القيمة 1000 وهي عنوان x في الذاكرة وعليه فإن معنى الجملة [mem=&x] هو أعطى المتغير mem في موقعه عنوان x .

2. الأداة * والتي تستخدم لإعطاء قيمة المتغير للمؤشر (المشار إليه) أي قيمة x في المثال السابق .

والمثال التالي يوضح العملية كما يلي :

```
y=*mem;
```

وفي هذه الحالة تكون القيمة المخزنة في موقع المتغير y هي قيمة x نفسها أي 3 أي يصبح معنى الجملة هو أعطى y القيمة المخزنة لدى موقع العنوان mem

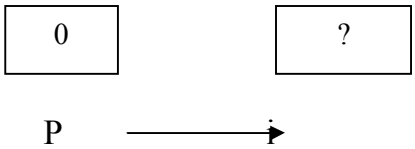
وللإيضاح أكثر نأخذ المثال التالي :

لنفرض

```
int i,*p;
p=&i;
```

وعليه فيوضع عنوان i في المؤشر p فإن هذا المؤشر سوف يشير إلى المتغير i

كما يلي :



إن وضع عنوان i في المؤشر p لم يؤثر على محتوى الموقع المخصص للمتغير i

ويمكن وضع العنوان في المؤشر أثناء الإعلان كما يلي :

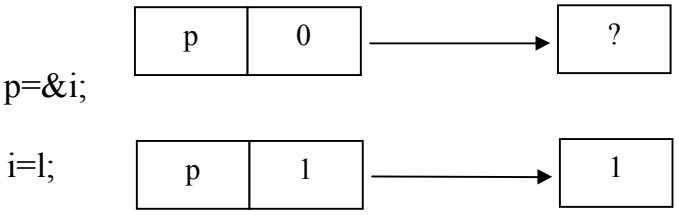
```
int i;
int *p=&i;
```

بعد أن يشير المؤشر إلى المتغير يمكن الوصول إلى القيمة المخزنة في العنوان باستخدام معامل المحتوى (*) فإذا كانت p تشير إلى المتغير i فانه بالإمكان طباعة محتوى الموقع (قيمة i) كما يلي :

```
cout<<*p;
```

ومادامت p تشير الى i فإن *p هي نفسها قيمة i (وكأنه اسم مرادف لـــــ i)

انظر الشكل التالي :




```
cout<<i ;     // print 1
cout<<*p;  // print 1
```

تمتلك لغة C++ المقدرة على نسخ المؤشرات (على أن تكون من نفس النوع) باستخدام أمر المساواة

فلو فرضنا أن q,p,j,i تم الإعلان عنها كما يلي :

```
int i,j,*p,*q
```

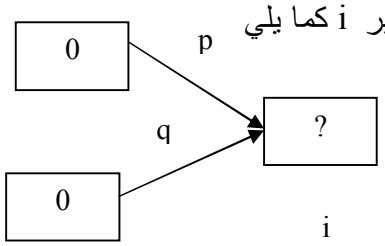
فإن الجملة التالية تضع عنوان المتغير i في المؤشر p

```
P=&i;
```

واستخدام التعليمة التالية

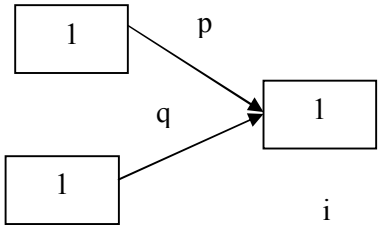
```
q=p;
```

سيؤدي إلى أن تشير المؤشرات إلى المتغير i كما يلي

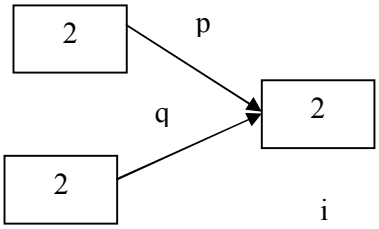


بعد ذلك يمكن إعطاء المتغير قيم باستخدام المؤشر كما يلي :

```
*p=1;
```



*q =2;



☑	تأخذ الأداتان & و* أولوية في التنفيذ على سائر العمليات(المؤثرات) المختلفة

مثال يوضح كيفية نسخ المؤشرات :

```
#include <iostream.h>
int main()
{
int i =5;
int *p,*q;
p=&i;
*p = i**p;
q = p;
*q =*q+*p;
cout<<" i= "<<i<<" \n";
cout<<" *p= "<<*p<<" \n";
cout<<" *q= "<<*q<<" \n";
return 0;
}
```

مثال

تتبع البرنامج التالي ، و حاول أن تستنتج شكل المخرجات

```
#include <iostream.h>
int main ()
{
int value1 = 5, value2 = 15;
int* p1;
int* p2;
p1 = &value1;
```

```
p2 = &value2;
*p1 = 10;
*p2 = *p1;
 p1 = p2;
*p1 = 20;
 cout << "value1==" <<value1<<"\t value2=="<< value2<<endl;
 return 0;
}
```

خرج البرنامج هو :

```
Value1==10        value2==20
```

العمليات الحسابية على المؤشرات:

تستطيع المؤشرات الإشارة إلى المتغيرات المفردة كما وتشير إلى عناصر المصفوفة array

لنفترض أن a,p عرفت كما يلي:

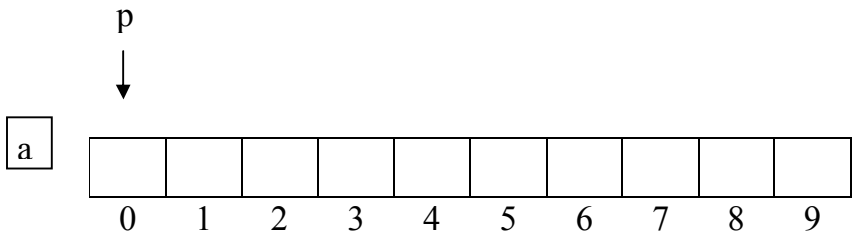
```
int a[10],*p;
```

فبهذا يمكن استخدام المؤشر p للإشارة إلى العنصر الأول من المصفوفة a (a[0])

كما يلي:

p=&a[0]; وهي مكافئة للعبارة p=a;

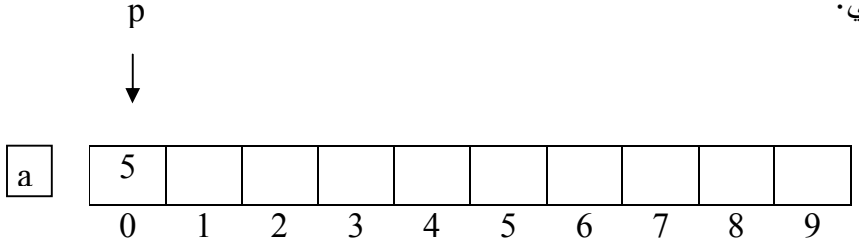
أي



وبهذا يمكن الوصول إلى a[0] عن طريق p وبإمكاننا تخزين القيمة 5 على سبيل المثال كعنصر في الموقع الأول من المصفوفة كما يلي:

```
*p=5;
```

فيصبح الشكل كما يلي:



☑	يعتبر اسم المصفوفة مؤشر ثابت على أول عنصر من عناصرها

يمكن تنفيذ ثلاث عمليات حسابية أساسية على المؤشر وهي:

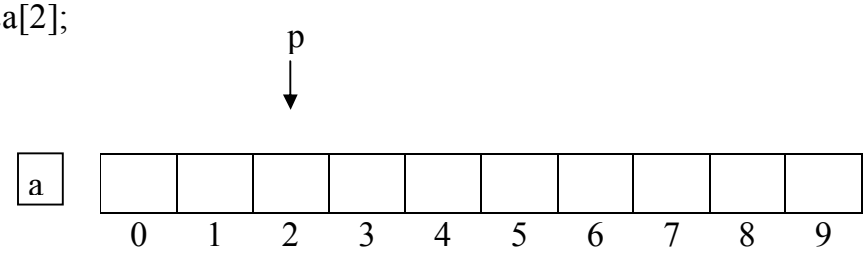
1. زيادة رقم صحيح إلى المؤشر.
2. طرح رقم صحيح من المؤشر.
3. طرح مؤشر من مؤشر.

لنفترض التعريف التالي:

```
int a[10],*p,*q;
```

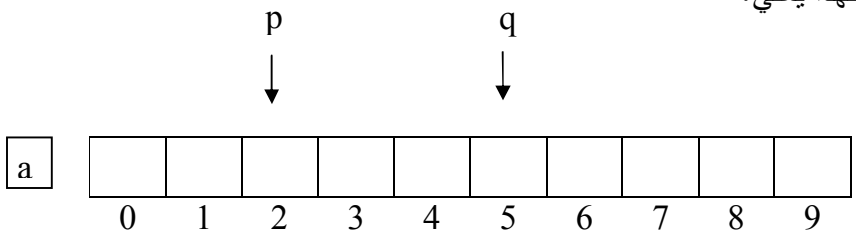
فإذا كان المؤشر p يشير إلى a[i] فإن زيادة j إلى المؤشر تعني أن المؤشر سوف يشير إلى العنصر a[i+j] ويمكن توضيح العملية كما يلي:

```
p=&a[2];
```



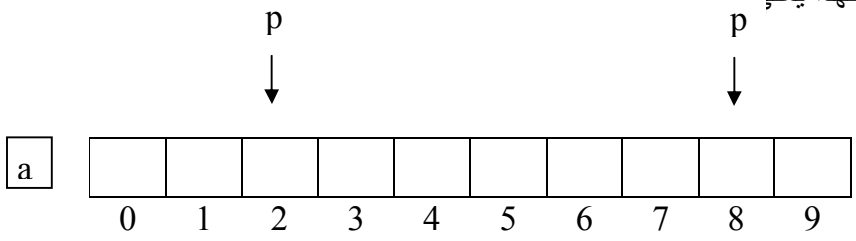
إذا جعلنا `q=p+3;`

فهذا يعني:



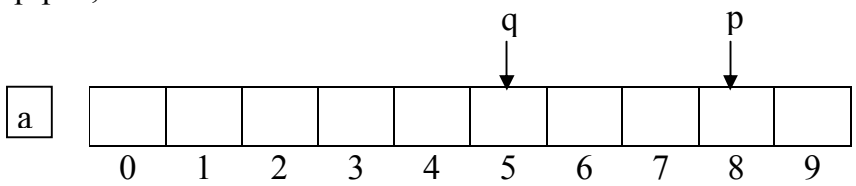
إذا جعلنا `p+=6;`

فهذا يعني:



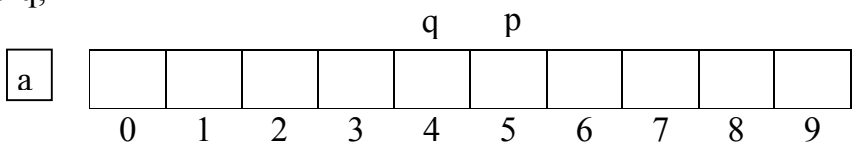
أما عملية الطرح من المؤثر فيمكن توضيحها بالشكل التالي:

```
p=&a[8];
q=p-3;
```



كما يمكن طرح مؤشر من مؤشر آخر كما يلي:

```
p=&a[5];
q=&a[1];
q=p-q;
```



يمكن بواسطة المؤشرات حساب مجموع عناصر مصفوفة

مثال :-

باستخدام المؤشرات احسب وأطبع مجموع عناصر مصفوفة حجمها عشرة عناصر تستخدم لتخزين بيانات من النوع الصحيح

```
#include <iostream.h>
int main()
{
int a[10],sum,*p,i;
sum=0;
for(i=0;i<10;i++)
{
cout<<"enter  a["<<i+1<<"]:";
cin>>a[i];
}
for(p=&a[0];p<&a[10];p++)
 sum+=*p;
cout<<"sum="<<sum<<endl;
return 0;
}
```

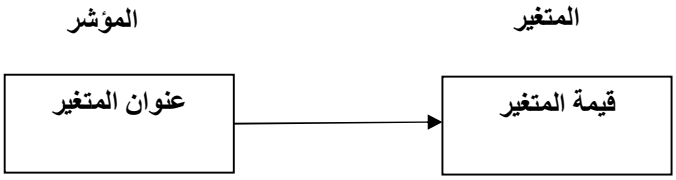
مثال يوضح استخدام المؤشرات مع المصفوفات:

```
#include<iostream.h>
{
int main()
{
int a[5]={5,10,15,20,25};
int *p;
p=a; //p=&a[0];
cout<<"the first method to print array elements \n";
for(int i=0;i<5;i++)
cout<<*(p+i)<<"\t";
cout<<"\n";
cout<<"the second method to print array elements\n";
for(p=a;p<&a[5];p++)
cout<<*p<<"\t";
cout<<"\n";
return 0;
}
```

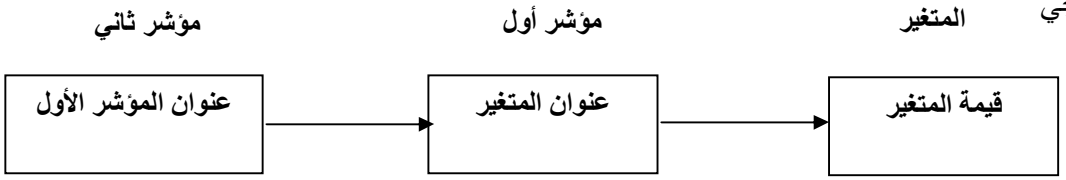
مؤشرات المؤشرات pointers to pointers

عند اعتبارنا المؤشر كمتغير فإنه يمكننا أن نشير له بمؤشر آخر:

1. تأشير منفرد



2. تأشير زوجي



ويتم الإعلان عن مؤشر المؤشر بالشكل التالي int **p;

وهو مؤشر لمؤشر من نوع عددي صحيح

مثال:

```
#include<iostream.h>
int main()
{
int v,*a,**b;
v=20;
```

```
a=&v;
b=&a;
cout<<"**b="<<**b<<endl;
return 0;
}
```

الحجز وإلغاء الحجز للذاكرة

لغة C++ تسمح بإستخدام المؤشر new لحجز الذاكرة الديناميكية والمؤشر delete لتنظيف الذاكرة

يمكن استخدام المؤشرnew لحجز ذاكرة موقع واحد حسب الصيغة التالية:

```
data type *p-var = new data type (value);
```

مثال: احجز ذاكرة لموقع واحد من نوع int وضع فيه القيمة 8

```
int *x =new int(8);
```

كما يمكن حجز ذاكرة لعدة مواقع(n موقع) كما في حالة المصفوفات وذلك حسب الصيغة التالية:

```
data type *p-var = new data type [n];
```

حيث n عدد المواقع

مثال: احجز ذاكرة لعشرة مواقع لبيانات من النوع الحقيقي:

```
float *y =new float[10];
```

كما ويستخدم المؤشر delete لحذف موقع واحد حسب الصيغة التالية:

```
Delete p-var
```

مثال:

```
int *x =new int(8);
delete x;
```

أما في حالة حذف عدة مواقع فتستخدم الصيغة التالية :

```
delete[ ] p-var;
```

حيث [ ] الرمز الدليلي للمصفوفة

مثال:

```
float *y =new float[10];
delete [ ] y;
```

مثال:-

اكتب برنامج لإدخال m من العناصر في مصفوفة ثم طباعتها:

```
#include<iostream.h>
int main()
{
int m,i;
cout<<" enter m\n";
cin>>m;
int *x = new int [m];
for(i=0;i<m;i++)
```

```
cin>>*(x+i);
cout<<" array elements is \n";
for(i=0;i<m;i++)
cout<<*(x+i)<<"\n";
delete [ ] x;
return 0;
}
```

المراجع References

تعريف المرجع:

هو عبارة عن اسم آخر لمتغير موجود وهي ميزة جديدة في لغة C++ تستخدم في بعض التطبيقات مثل الاستدعاء بالعنوان في الدوال والتي سوف نتعرف عليها لاحقاً عندما نتحدث عن الدوال

الصيغة العامة للإعلان عن مرجع

```
data type    var-name;
data type    & reference = var-name;
```

مثال1:

برنامج يوضح مفهوم المرجع

```
#include<iostream.h>
int main()
{
int i=6;
int &r=i;
r=13;
cout<<"i="<<i<<"\n";
i=15;
cout<<"r="<<r<<"\n" ;
return 0;
}
```

☑	يتم التعامل مع المرجع في كافة العمليات الرياضية مثل المتغير العادي.
☑	المرجع يأخذ قيمة ابتدائية مثل المؤشر.