الدرس السادس: المؤشرات™

Pointers

1- تعريف المؤشرات

المؤشرات متغيرات تستخدم لتخزين عناوين متغيرات أخرى في ذاكرة الحاسب . وبالتالي يمكن من خلال هذه المؤشرات الوصول إلى المتغيرات بطريقة غير مباشرة.

وتستخدم المؤشرات أساسا في إنشاء القوائم المترابطة (Linked Lists) والتحكم في الكائنات التي يتم إنشاؤها بطريقة ديناميكية أثناء تنفيذ البرنامج.

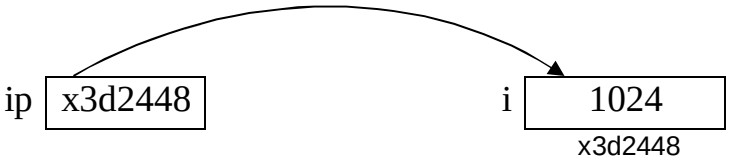
في لغة C++، يتم تعريف المؤشرات باستخدام الصيغة التالية:

```
Data_Type  *Pointer_Name;
```

كما يحبذ إعطاء قيمة ابتدائية للمؤشر عند تعريفه بأن يشير إلى كائن معين، شريطة أن يكون هذا الكائن من نفس نوع المؤشر ويتم ذلك باستخدام معامل العنوان "**&**" كما في الكود التالي:

```
int  i = 1024;
int  *ip = &i;
```

إذا فرضنا أن عنوان المتغير i هو x3d2448، فإن العلاقة بين i و ip يمكن تمثيلها كما يلي:



2- التعامل مع المؤشرات

يمكن إسناد قيمة مؤشر إلى مؤشر آخر من نفس النوع كما في المثال التالي:

```
int  i = 1024;
int  *ip = &i;
int  *ip2 = ip;
```

وبذلك يشير المؤشر ip2 إلى نفس المتغير الذي يشير إليه المؤشر ip وهو i في هذه الحالة.

يمكن الوصول إلى بيانات الكائن الذي يشير إليه المؤشر باستخدام معامل المحتوى "*" كما في المثال التالي:

```
int  i = 1024;
int  *ip = &i;
int  k = *ip;      // k = 1024
```

قاعدة عامة

- التركيب **&i** يعني **عنوان المتغير** i ويتم تخزينه داخل مؤشر من نفس نوع المتغير i.
- التركيب ***ip** يعني **محتوى** المكان الذي يشير إليه المؤشر ip.
- إذا التقى معامل العنوان (**&**) ومعامل المحتوى (*****) يقوم كل منهما بإلغاء مفعول الآخر مما يعني أن:

$$*\&p = \&*p = p$$

تمرين رقم 1: ما هي الأخطاء الواردة في الكود التالي ؟

```
1   int  a = 100;
2   int  *pa = a;
3   float  *fp = &a;
```

الحل

- في السطر الثاني وقع إسناد قيمة صحيحة (a) إلى المؤشر pa وهذا خطأ، الصحيح هو:

```
int  *pa = &a;
```

- في السطر الثالث وقع إسناد عنوان متغير من فئة الأعداد الصحيحة (int) إلى مؤشر من فئة عدد عشري (float) وهذا أيضا خطأ، الصحيح هو:

```
int  *fp = &a;
```

تمرين رقم 2: ما هي المخرجات التي ستظهر على الشاشة عند تنفيذ البرنامج التالي:

```
#include <iostream>
using namespace std;
int main( )
{
      int value1 = 5, value2 = 15;
      int  *p1 = &value1;
      int  *p2 = &value2;
      *p1 = 10;
      *p2 = *p1;
      p1 = p2;
      *p1 = 20;
      cout << "value1 = " << value1 << "\t"<<"value2 = " << value2;
      cout<<endl;
      return 0;
}
```

الحل (مخرجات البرنامج)

```
value1 = 10     value2 = 20
Press any key to continue . . .
```

تمرين رقم 3: ما هي المخرجات التي ستظهر على الشاشة عند تنفيذ البرنامج التالي:

```
#include <iostream>
using namespace std;
int main( )
{
      int i = 15, j = 8;
      int *p, *q  = &j;
      p = &i;
      *p = 5;
      (*p) ++;
      j = *q + 5;
      cout << "i = " << i << "\t"<<"j = " <<j;
      cout<<endl;
      return 0;
}
```

الحل (مخرجات البرنامج)

```
i = 6   j = 13
Press any key to continue . . . _
```

ملاحظة: تتيح لغة C++ التعريف بـ**مؤشرات عامة** من نوع **void** أي بإمكانها أن تؤشر إلى أي نوع من البيانات : عدد صحيح، عدد حقيقي (عشري)، حرف، ...الخ.

مثال

```
char  c;
void  *q;
q = &c;
```

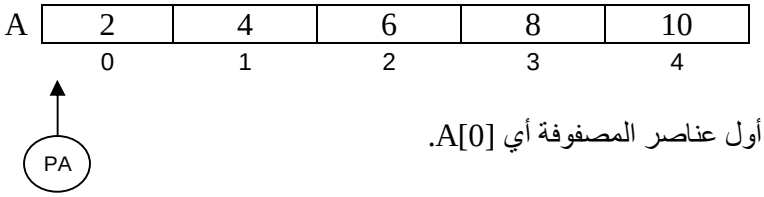
4- المؤشرات و المصفوفات

المصفوفة عبارة عن مجموعة متتابعة من القيم التي تنتمي إلى نفس النوع (int، float، char، ...).

من التطبيقات المهمة للمؤشرات معالجة المصفوفات. فعندما نضع المؤشر عند بداية المصفوفة، من السهل التحرك خلال القائمة و معالجة عناصرها بطريقة متتابعة.

لنفترض أن A مصفوفة من النوع الصحيح و PA مؤشر من النوع صحيح :

```
int  A[5] = {2, 4, 6, 8, 10};
int  *PA = A ;    // اسم المصفوفة يمثل عنوانها في الذاكرة
```



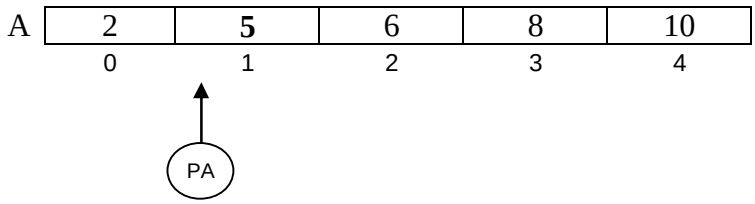
المؤشر PA يؤشر إلى أول عناصر المصفوفة أي A[0].

وبذلك يكون محتوى PA* هو قيمة أول عنصر في المصفوفة (أي 2 في هذه الحالة).

بعد تنفيذ الأمرين:

```
PA++;       // زيادة قيمة المؤشر بواحد بحيث يصبح يشير إلى العنصر الثاني
(*PA)++;    //  5  زيادة قيمة العنصر الذي يشير إليه PA بواحد بحيث يصبح
```

يصبح الوضع كما يلي:



تمرين رقم 1 : ما هي المخرجات التي ستظهر على الشاشة عند تنفيذ البرنامج التالي:

```
#include <iostream>
using namespace std;
int main( )
{
      int  Array[5] = {31, 45, 77, 52, 93};
      int  *PArray = Array ;
      for (int i = 0; i < 5; i++)
            cout<<*(PArray + i)<<"\t";
      cout<<endl;
      return 0;
}
```

الحل (مخرجات البرنامج)

```
31      45      77      52      93
Press any key to continue . . .
```

تمرين رقم 2 : ماذا سيظهر على الشاشة عند تنفيذ البرنامج التالي ؟

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

# define TAB '\t';  // يحتوي الثابت TAB على مسافة حقل

int main( )
{
      int A[5];
      int *p;
      p = A; *p = 10;
      p++;          *p = 20;
      p = &A[2];    *p = 30;
      p = A + 3;    *p = 40;
      p = A;        *(p+4) = 50;

      for(int i = 0; i<5; i++)
            cout<<A[i]<<TAB;
      cout<<endl;
      return 0;
}
```

الحل (مخرجات البرنامج)

```
10      20      30      40      50
Press any key to continue . . .
```

5- المؤشرات و السلاسل الحرفية

يمكن تعريف السلسلة الحرفية كمصفوفة من الحروف أو كمؤشر نحو مجموعة من الحروف.

مثال

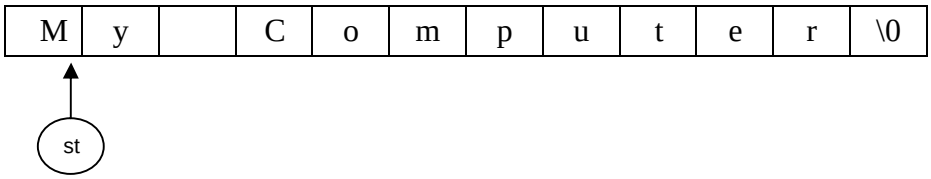
```
#include <iostream>
using namespace std;

int main( )
{
      char *st = "My Computer";
      st++;
      cout<<st;
      cout<<endl;
      return 0;
}
```

مخرجات البرنامج

```
y Computer
Press any key to continue . . .
```

في هذا البرنامج وقع استخدام مؤثر الزيادة بواحد ++ مع المؤشر st فأصبح يشير إلى الحرف الثاني من السلسلة (أي الحرف y).



ملاحظات

1- يمثل اسم المصفوفة عنوانها في الذاكرة وبالتالي يعامل كمؤشر نحو أول عناصر المصفوفة.

2- يمكن تمرير سلسلة حرفية كاملة إلى دالة باستخدام مؤشر نحو هذه السلسلة.

مثال

```
#include <iostream>

using namespace std;

void DisplayString(char *ps)
{
      while (*ps != '\0')
      {
            cout<<*ps;
            ps++;
      }
}
```

```
int main ( )
{
	char str[] = "Actions speak louder than words";
	DisplayString(str);
	cout<<endl;
	return 0;
}
```

مخرجات البرنامج

```
Actions speak louder than words
Press any key to continue . . .
```

تمرين :

1- قم بإنشاء دالة اسمها

Copy(char *st1,char *st2)

وتقوم بنسخ محتوى السلسلة الحرفية str2 إلى السلسلة str1 (بدون استخدام الدالة strcpy).

2- أكتب برنامجا بلغة C++ يتم فيه استخدام الدالة Copy لنسخ محتوى سلسلة معينة إلى سلسلة أخرى ثم طباعة السلسلتين على الشاشة.

- **الحل**

```
#include <iostream>
using namespace std;
void Copy(char *st1, char *st2)
{
	while (*st2 != '\0')
	{
		*st1 = *st2;
		st2++;
		st1++;
	}
	*st1 = '\0';
}
int main()
{
	char str1[40];
	char *str2 = "Give respect, take respect ";
	Copy(str1,str2);
	cout<<"str1 = "<<str1;
	cout<<endl;
	cout<<"str2 = "<<str2;
	cout<<endl;
	return 0;
}
```

- **مخرجات البرنامج**

```
str1 = Give respect, take respect
str2 = Give respect, take respect
Press any key to continue . . .
```