C++程式設計

單元四：基本指標變數

指標變數(一)

- 指標變數是C語言中特殊的資料存取方式，也是C語言的特色。
- 一般變數存放的是資料本身，而指標變數則是存放記憶體位址，然後再透過這個位址去指向到資料真正存在的地方。
- 因為指標變數是對實際記憶體位址做操作，所以使用時要非常小心，以免造成系統程式的錯亂。

指標變數(二)

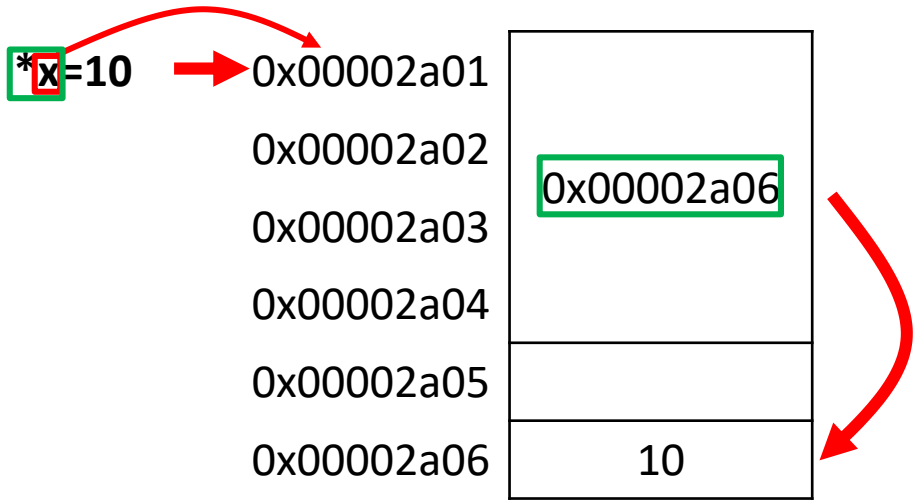
- 指標變數在使用時一定要指向到已存在的變數位址，否則程式會無法編譯。
- 為避免錯誤，指標變數在宣告後，於未指定明確位址之前，可先令其等於0或NULL。
- 因指標變數本身所存放的是記憶體位址，因此，若系統為32位元時，其大小固定為4Bytes、若為64位元則為8Bytes。

一般變數和指標變數的差異

一般變數—

變數所指向位址就是存放資料所在位址

指標變數—



變數所在位址存放位址

再透過這個位址指向去存取資料

指標變數宣告

- 語法格式：

```
資料型態 *變數名稱;
資料型態* 變數名稱;
```

- 範例—

```
int *p; 或 int *p=NULL;
int* p; 或 int* p=NULL;
```

指標變數的初始化—指向變數

- 宣告時即指向一變數位址

```
int  x=10;
int  *p=&x;
```

&的作用在此處為"取址"運算，即取得變數所在的記憶體位址

- 執行階段才指向一變數位址

```
int  x=10;
int  *p=NULL;


p=&x
```

指標變數的初始化—指向一維陣列

- 宣告時即指向一維陣列

```
int x[5];
int *p=x;
```

- 執行階段才指向一維陣列

```
int x[5];
int *p=NULL;


p=x; 或 p=&x[0];
```

以指標變數讀取一維陣列值

- 宣告一指標變數並指向已存在之陣列

```
int x[4];
int *p=x;
```

- 指定後，可形成下列對應關係

```
x[0]=*(p+0)
x[1]=*(p+1)
x[2]=*(p+2)
x[3]=*(p+3)
```

指標變數的初始化—指向二維陣列

- 宣告時即指向二維陣列

```
int x[3][5];
int *p=&x[0][0];
```

- 執行階段才指向二維陣列

```
int x[5];
int *p=NULL;


p=&x[0][0];
```

以指標變數讀取二維陣列值

- 宣告一指標變數並指向已存在之陣列

```
int  x[3][3];
int  *p=&x[0][0];
```

- 指定後，可形成下列對應關係

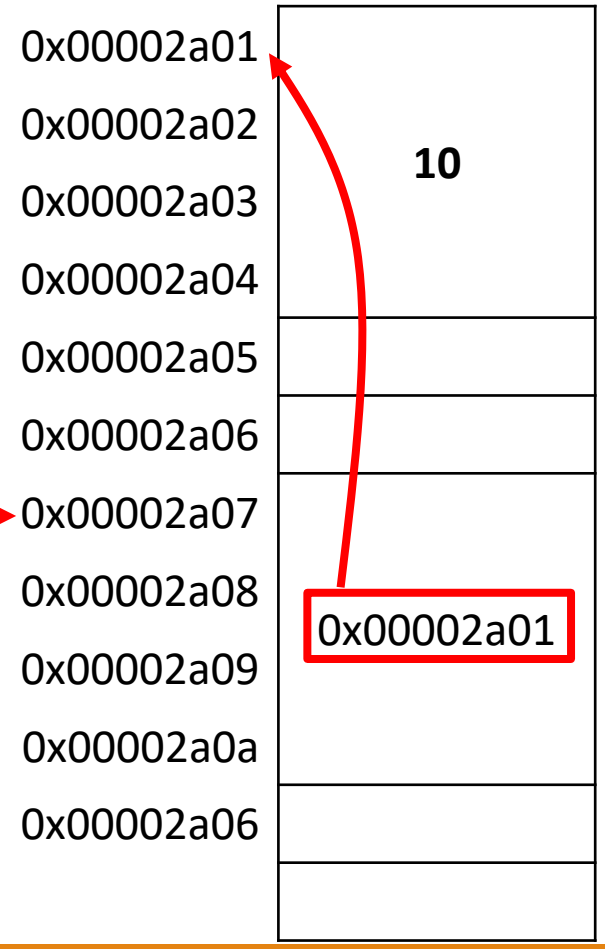
x[0][0]=*(p+0)、x[0][1]=*(p+1)、x[0][2]=*(p+2)
x[1][0]=*(p+3)、x[1][1]=*(p+4)、x[1][2]=*(p+5)
x[2][0]=*(p+6)、x[2][1]=*(p+7)、x[2][2]=*(p+8)

指標變數的運作

```
void  main(){

   int   x=10;
   int  *p=&x;

}
```



&x=0x00002a01

x=10

&p=0x00002a07

p=0x00002a01

*p=x=10

&p取得指標位址

p　指標位址的內容

*p指標指向的位址存放的值

指標變數的觀察(一)

```
#include <iostream>
using namespace std;
main(){
	int a=10,b=5;
	int *p1=NULL,*p2=NULL;

	cout<<"原宣告資料--------"<<endl;
	cout<<"\t變數 a 的位址為："<<&a<<" 、其內含值為："<<a<<endl;
	cout<<"\t變數 b 的位址為："<<&b<<" 、其內含值為："<<b<<endl;
	cout<<"\t指標變數 p1 的位址&p1=："<<&p1<<" 、內含值p1="<<p1<<endl ;
	cout<<"\t指標變數 p2 的位址&p2=："<<&p2<<" 、內含值p2="<<p2<<endl<<endl ;
```

```
原宣告資料--------
        變數 a 的位址為: 0x70fe0c 、其內含值為: 10
        變數 b 的位址為: 0x70fe08 、其內含值為: 5
        指標變數 p1 的位址為: 0x70fe00 、內含值為: 0
        指標變數 p2 的位址為: 0x70fdf8 、內含值為: 0
```

指標變數的觀察(二)

```
        p1=&a; p2=&b;
        cout<<"執行 p1=&a、p2=&b 之後\n";
        cout<<"\t變數 a 的位址為："<<&a<<" 、其內含值為："<<a<<endl;
        cout<<"\t變數 b 的位址為："<<&b<<" 、其內含值為："<<b<<endl;
        cout<<"\t指標變數 p1 的位址 &p1="<<&p1<<" 、內含值為 p1="<<p1<<"、指向位址的資料
*p1="<<*p1<<endl ;

        cout<<"\t指標變數 p2 的位址 &p2="<<&p2<<" 、內含值為 p2="<<p2<<"、指向位址的資料
*p2="<<*p2<<endl <<endl;
```

```
執行 p1=&a、p2=&b  之後
        變數 a 的位址為: 0x70fe0c 、其內含值為: 10
        變數 b 的位址為: 0x70fe08 、其內含值為: 5
        指標變數 p1 的位址 &p1=0x70fe00 、內含值為 p1=0x70fe0c、指向位址的資料 *p1=10
        指標變數 p2 的位址 &p2=0x70fdf8 、內含值為 p2=0x70fe08、指向位址的資料 *p2=5
```

指標變數的觀察(三)

```
        *p1=35;*p2=99;
        cout<<"執行 *p1=35、*p2=99  之後\n";
        cout<<"\t變數 a 的位址為："<<&a<<" 、其內含值為："<<a<<endl;
        cout<<"\t變數 b 的位址為："<<&b<<" 、其內含值為："<<b<<endl;
        cout<<"\t指標變數 p1 的位址 &p1="<<&p1<<" 、內含值為 p1="<<p1<<"、指向位址的資料
*p1="<<*p1<<endl ;

        cout<<"\t指標變數 p2 的位址 &p2="<<&p2<<" 、內含值為 p2="<<p2<<"、指向位址的資料
*p2="<<*p2<<endl <<endl;
```

```
執行 *p1=35、*p2=99  之後
        變數 a 的位址為: 0x70fe0c 、其內含值為: 35
        變數 b 的位址為: 0x70fe08 、其內含值為: 99
        指標變數 p1 的位址 &p1=0x70fe00 、內含值為 p1=0x70fe0c、指向位址的資料 *p1=35
        指標變數 p2 的位址 &p2=0x70fdf8 、內含值為 p2=0x70fe08、指向位址的資料 *p2=99
```

指標變數的觀察(四)

```
        p1=p2;
        cout<<"執行 p1=p2  之後\n";
        cout<<"\t變數 a 的位址為："<<&a<<" 、其內含值為："<<a<<endl;
        cout<<"\t變數 b 的位址為："<<&b<<" 、其內含值為："<<b<<endl;
        cout<<"\t指標變數 p1 的位址 &p1="<<&p1<<" 、內含值為 p1="<<p1<<"、指向位址的資料
*p1="<<*p1<<endl ;

        cout<<"\t指標變數 p2 的位址 &p2="<<&p2<<" 、內含值為 p2="<<p2<<"、指向位址的資料
*p2="<<*p2<<endl <<endl;

        return 0;
}
```

```
執行 p1=p2  之後
        變數 a 的位址為: 0x70fe0c 、其內含值為: 35
        變數 b 的位址為: 0x70fe08 、其內含值為: 99
        指標變數 p1 的位址 &p1=0x70fe00 、內含值為 p1=0x70fe08、指向位址的資料 *p1=99
        指標變數 p2 的位址 &p2=0x70fdf8 、內含值為 p2=0x70fe08、指向位址的資料 *p2=99
```

指標變數與陣列的觀察(一)

```
#include<iostream>
using namespace std;
main(){
	int a[]={1,2,3,4,5};
	int *p=NULL;
	int i;
	cout<<"\n 原陣列資料："<<endl;
	for(i=0;i<5;i++)
		cout<<"\t a[" << i << "] 的位址為" << &a[i] <<"、a[" << i << "]="<<a[i]<<endl;
	cout<<"\n 執行 p=a 之後："<<endl;
	p=a;
	for(i=0;i<5;i++)
		cout<<"\t p+"<< i <<" 位址的內容為 " << p+i <<"、*(p+"
<<i<<")="<<*(p+i)<<endl; ;
	return 0;
 }
```

指標變數與陣列的觀察(二)

```
原陣列資料:
        a[0] 的位址為0x70fde0、a[0]=1
        a[1] 的位址為0x70fde4、a[1]=2
        a[2] 的位址為0x70fde8、a[2]=3
        a[3] 的位址為0x70fdec、a[3]=4
        a[4] 的位址為0x70fdf0、a[4]=5

執行 p=a 之後:
        p+0 位址的內容為 0x70fde0、*(p+0)=1
        p+1 位址的內容為 0x70fde4、*(p+1)=2
        p+2 位址的內容為 0x70fde8、*(p+2)=3
        p+3 位址的內容為 0x70fdec、*(p+3)=4
        p+4 位址的內容為 0x70fdf0、*(p+4)=5
```

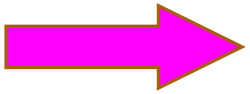
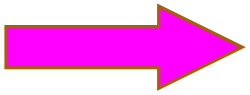
程式執行*(p+1)中之+1，指的是位址的偏移量，所以若是整數指標，實際上是把原p指向的位址+4(因整數占4Bytes)

指標變數的運算

- 範例程式—

```
int  x[3]={1,2,3};
int  *p=x; //&x=ff2c00
```

- 執行程式—

*p+1;		x[0]+1;
*(p+1);		ff2x00+4 即指向x[1]

動態配置

- 語法格式：

```
資料型態 *變數名稱=new 資料型態(初值);
資料型態 *變數名稱=new 資料型態[大小];
```

- 範例—

```
int *p=new int(9);
int *p=new int[3]; //等同宣告p[3]陣列
```

刪除動態配置

- 語法格式：

```
delete 原動態配置之指標變數名;
delete [] 原動態配置之指標變數名;
```

- 範例—

```
delete p;
delete [] p;
```