Arrays

- 函式的參數傳遞有三種
  - Call by value (傳值) //C和C++有支援
    - 函數呼叫: function(a, b)
    - 函數定義: void function(int x, int y)
  - Call by address (傳位址) //C和C++有支援
    - 函數呼叫: function(&a, &b)
    - 函數定義: void function(int * x, int *y)
  - Call by reference (傳參考) //只有C++有支援
    - 函數呼叫: function(a, b)
    - 函數定義: void function(int &x, int &y)
- Call by address (傳位址)和Call by reference (傳參考)具有相同結果， Call by reference (傳參考)主要簡化Call by address (傳位址)的符號運算

Arrays

- 函式的參數傳遞有三種
  - Call by value
    - 函數呼叫: function(a, b)
    - 函數定義: void function(int x, int y)
    - 主要把數值拷貝到函示，函示與主程式的變數互不相干

```
6  void main ()
7  {
8      int x=100;
9      int y=addbyone(x);
10     printf("x=%d\n",x);
11     system("pause");
12 }
13
14 int addbyone (int x)
15 {
16     x++;
17     printf("x=%d\n",x);
18     return x;
19 }
```

```
x=100
```

```
x=101
```

Arrays

- 函式的參數傳遞有三種
  - Call by address
    - 函數呼叫: function(&a, &b)
    - 函數定義: void function(int * x, int *y)
    - 呼叫函數主要傳給函數位址(&x)，函數則以指標指導相對應的變數(*xptr)，函數運算會會更改相對應的變數內容

```
6  void main ()
7  {
8      int x=100;
9      int y=addbyone(&x);
10     printf("x=%d\n",x);
11     system("pause");
12 }
13
14 int addbyone (int* xptr)
15 {
16     (*xptr)++;
17     printf("*xptr=%d\n",*xptr);
18     return *xptr;
19 }
```

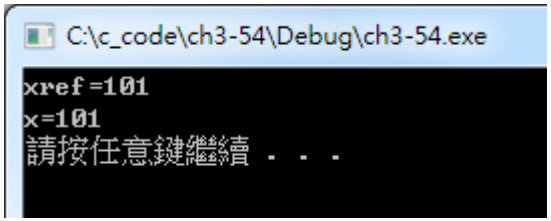
```
x=101
```

```
*xptr=101
```

Arrays

- 函式的參數傳遞有三種
  - Call by reference
    - 函數呼叫: function(a, b)
    - 函數定義: void function(int &x, int &y)
    - 呼叫函數主要傳給函數參考變數或物件(x)，函數會以位址(&xref)建立起相連等號，並表示使用相同記憶體空間，函數運算會會更改相對應的變數內容
    - 因爲C沒有支援, 需要把main.c改成main.cpp

```
6  void main ()
7  {
8      int x=100;
9      int y=addbyone(x);
10     printf("x=%d\n",x);
11     system("pause");
12 }
13
14 int addbyone (int &xref)
15 {
16     xref++;
17     printf("xref=%d\n",xref);
18     return xref;
19 }
```



Arrays

- 程式利用%p轉換指定詞 (一個用來列印位址的特殊轉換指定詞) 印出array，&array[0]和&array，來驗證陣列名稱確實是此陣列第一個元素所在的位址。
- %p轉換指定詞通常會將位址以十六進制數的形式印出來。

```c
#include <stdio.h>
#include <stdlib.h>

int main (void)
{
    char array[5];
    printf("    array = %p\n&array[0] = %p\n   &array = %p\n",
        array,&array[0],&array);

    system("pause");
    return 0;
}
```

C:\c_code\ch3-55\Debug\ch3-55.exe

```
    array = 003AFD34
&array[0] = 003AFD34
   &array = 003AFD34
請按任意鍵繼續 . . .
```

Arrays

- 傳遞陣列引數給函式
  - 陣列(a[5])自動以Call by reference (傳參考) 來呼叫傳遞
    - 函數呼叫: **modifyArray(a)**
    - 函數定義: **void modifyArray(int b[])**
      - 參數b接收一個整數陣列
      - 陣列的中括號裡不需要指定陣列的大小

Arrays

- 傳遞陣列引數給函式

```c
#include <stdio.h>
#include <stdlib.h>
#define SIZE 5

void modifyArray(int b[], int size);
void modifyElement(int e);

int main( void )
{
    int a[SIZE] = {0,1,2,3,4};
    int i;

    printf("Effects of passing entire array by reference:\n\nThe"
       "values of the original array are:\n");

    for (i=0;i<SIZE;i++)
    {
       printf("%3d",a[i]);
    }
    printf("\n");

    modifyArray(a, SIZE);
    printf("The values of the modified array are:\n");
    for (i=0;i<SIZE;i++)
    {
       printf("%3d",a[i]);
    }

    printf("\n\n\nEffects of passing array element"
       "by value:\n\nThe value of a[3] is %d\n",a[3]);

    modifyElement(a[3]);
    printf("The value of a[3] is %d\n", a[3]);

    system("pause");
    return 0;
}
```

Call by value

C:\c_code\ch3-57\Debug\ch3-57.exe

```
Effects of passing entire array by reference:

Thevalues of the original array are:
  0  1  2  3  4
The values of the modified array are:
  0  2  4  6  8


Effects of passing array elementby value:

The value of a[3] is 6
Value in modifyElement is 12
The value of a[3] is 6
請按任意鍵繼續 . . .
```

Arrays

```
38
39 void modifyArray(int b[],int size)
40 {
41     int j;
42
43     for (j=0;j<size;j++)
44     {
45         b[j] *=2;
46     }
47 }
48
49 void modifyElement(int e)
50 {
51     printf("Value in modifyElement is %d\n", e *= 2);
52 }
53
```

C:\c_code\ch3-57\Debug\ch3-57.exe

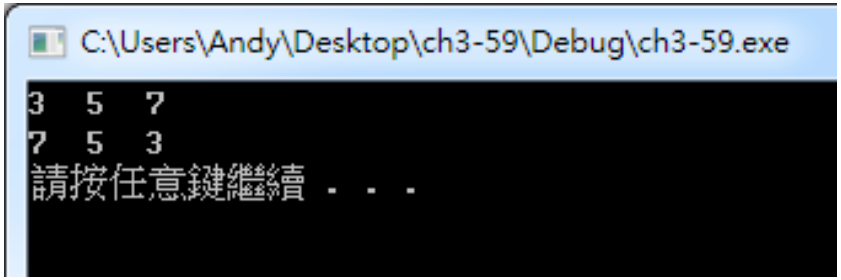
```
Effects of passing entire array by reference:

Thevalues of the original array are:
  0  1  2  3  4
The values of the modified array are:
  0  2  4  6  8


Effects of passing array elementby value:

The value of a[3] is 6
Value in modifyElement is 12
The value of a[3] is 6
請按任意鍵繼續 . . .
```

Arrays

- **Call by Address**

```c
#include <stdio.h>
#include <stdlib.h>

void inverse(int *);

int main()
{
    int a[3]={3,5,7},i;
    for (i=0;i<3;i++)
        printf("%d  ",a[i]);
    printf("\n");

    inverse(a);

    for (i=0;i<3;i++)
        printf("%d  ",a[i]);
    printf("\n");

    system("pause");
    return 0;
}
```

```c

void inverse(int *b)
{
    int tmp[3],i;
    for (i=0;i<3;i++)
        tmp[2-i]=b[i];
    for (i=0;i<3;i++)
        b[i]=tmp[i];
}
```



Homework

- P6-15(Fig 6.10)
- P6-27(Fig 6.16)
- 6.11