1

回顧 – 指標與陣列

- 所有變數都存在電腦的記憶空間，其所在地方稱為記憶體位址 (地址, address)。
- 指標 (pointer)，即為用來記錄地址的特殊型別。
- 指標的宣告

```
int A=3; int *ptrA:
double B=4.0, *ptrB;
float  C; float *ptrC;
```

- 指標變數亦有型別，用來讓電腦知道所記錄的地址 (所指向的位置上) 存放資料的型別。

- 取址運算子，用來取一個變數的記憶體位置。

```
ptrA = &A;
ptrB = &B;
ptrC = &C;
```

- 欲存取一記憶體位址上的資料或值，則使用取值運算子 (deference)。

```
cout << (*ptrA)<<endl;
(*ptrB)=B+3.0;
*ptrC = 5.0;
```

2

回顧 – 指標與陣列

- 一維陣列裡所存放的資料，常使用索引用來指定要存取那一個元素。
- int q[5] = {0, 1, 2, 3, 4};

```
cout << q[2];
q[3] = 7;
q[1] += 5;
```

- 陣列的名字本身即為一指標，而其後附加的索引 i 即代表要存取由該記憶體地址開始的第 i 個元素。

- 所以，我們可以亦可用指標來存取陣列中的元素。

```
int *r; r=q;
cout << r[4];
r[1] = 7;
```

- 也因此，我們可以將陣列當成函式呼叫的參數傳入函式加以使用。

3

隨堂練習

1. 請寫一函式 void plusOne(int n, int *a); ，會將傳入的陣列 a 中的前 n 個元素都加 1。
2. 請寫一函式 void minus (int n, int *a, int b); ，會將 a 陣列中的前 n 個元素都減掉 b 值。
3. 請寫一函式 void printArray(int n, int *a); 會將 a 陣列中的前 n 個元素印出來。
4. 請寫一主程式，宣告一個10個元素的陣 列 q，並填入 1, 2, 3, … 10 的值。想辦法利用 plusOne() 函式以及 minus()函式將 q 陣列的內容變成 [2,3,4,5,6,2,3,4,5,6]，並利用 printArray 將過程和最後結果印出來。

4

Lecture 12

參考 (reference)
函式的參數傳遞

5

今日內容

- 參考 (Reference)
- 函式的參數傳遞
  - 傳值
  - 傳址
  - 傳參考

6

參考

Reference

7

參考 (Reference)

* 在程式執行時，為**取值運算子**
* 在變數宣告時，代表其**宣告的變數為指標變數**

  int b = 4; int *a = &b; (*a) = 3;

  int *q, r, s; ← q 為指標變數; r, s 為兩個整數變數

  int *q, *r, *s; ← q, r, s 為三個指標變數

  cout << b;

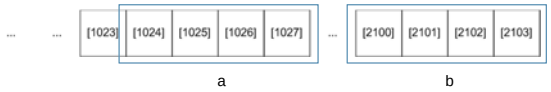
&在程式執行時，為取址運算子

& 在變數宣告時？

8

指標 (Pointer) 相關之運算子

- **&**
  - 宣告時
    - 宣告一個變數為其它的變數的參考 (reference)
    - 可以把它想為一個變數的分身、
  - 執行時
    - 取得一變數之記憶體位置 (e.g. **&**a, **&**b)，稱為**取址**運算子。



9

12-1.cpp

變數 b 為變數 a 的參考
也可說變數 b 為變數 a 的別名、分身

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

int main() {
  int a = 3;
  int &b = a;

  cout << a << endl;
  cout << b << endl;
  b = 4;
  cout << a << endl;
  return 0;
}
```

```
3
3

4
```

10

12-1.cpp 的說明

- 宣告 b 為一參考型別的變數，其參考的變數為 a
- 參考型別在宣告時，**一定**要指定參考那一個變數
- 宣告出來的參考型別變數 b, 為原來變數 a 的分身。
- 有時, 亦可說 b 為 a 的別名 (alias)
- 所以，改變參考型別變數 b 的值，被參考到的變數 a 的值也會一起跟著改變。

11

12-2.cpp

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

int main() {
  int a = 3;
  int &b = a;

  cout << &a << endl;
  cout << &b << endl;

  return 0;
}
```

```
0026F8F8
0026F8F8
```

12

12-2.cpp 的說明

- 宣告 b 為一參考型別的變數，其參考的變數為 a
- 分別將 a, b 兩個變數的記憶體位址印出來，發現它們完全一樣。故說 b 為 a 的分身或別名。
- 也因此，改變其中一個變數的值，另一個出來的值也跟著改變，因為他們實質上為同一個變數，只是他們有著不一樣的名字。

13

指標與參考的用途

- 指標 (pointer)
  - 用來在不同的函式之間傳遞資料
  - 用來進行動態的記憶體配置
- 參考 (reference)
  - 用來在不同的函式之間傳遞資料

14

函式的資料傳遞

15

函式定義

```
int sum(int x) {
  int i, sum = 0;
  for(i=1;i<=x;i++) {
      sum += i;
  }
  return sum;
}
```

函式定義的介面決定了資料如何**進入**此函式、以及運算結果回傳給呼叫者 (caller)。

此參數列可以將資料傳入進行運算，也可用來將資料傳出！(這個範例不行！)

此為函式傳出資料的途徑一，使用 **=** 將回傳的資料存入一指定變數。
e.g. q = sum(200);

16

12-3.cpp

```
#include <iostream>
using namespace std;
int test(int c) {
  c = c * 2;
  return c+1;
}
int main() {
  int a = 3;
  int b;
  b = test(a);
  cout << "a=" << a << endl;
  cout << "b=" << b << endl;
  return 0;
}
```

- 在 C/C++ 裡, 參數的傳遞內定的方法是傳值呼叫 (call by value)。
- 在 test 函式被呼叫時, 主程式傳入的 a 變數的值被**複製**一份到函式裡區域變數 c ，然後才開始執行 test 函式的內容。
- test 函式結束時將 c+1 的值透過 return 的功能傳回主程式中，並在此範例中存入 b 變數，然後 c 變數即被消滅。
- 因為是傳值呼叫，無論 c 變數如何改變，都不會影響主程式裡 a 變數的值。

```
a=3
b=7
```

17

參數傳遞的方式有三種

- **傳值 (call by value)**
  - 此為預設方式, 將傳入的變數值**拷貝**一份到函式內的區域變數 (local variable).
- **傳址 (call by address)**
  - 使用指標型別的變數, 將變數的記憶體位址拷貝一份到函式內的區域指標變數
- **傳參考 (call by reference)**
  - 使用參考型別的變數, 將函式的區域參考變數參考到呼叫者的變數

18

12-4.cpp

```
#include <iostream>
using namespace std;
void swap(int a, int b);

int main() {
  int a = 3;
  int b = 7;
  cout << "a=" << a << ", b=" << b << endl;
  swap(a, b);
  cout << "a=" << a << ", b=" << b << endl;
  return 0;
}
```

19

12-4.cpp

```
void swap(int a, int b) {
  cout << a << ", " << b << endl;
  int tmp = a;
  a = b;
  b = tmp;
  cout << a << ", " << b << endl;
}
```

```
a=3, b=7
3, 7
7, 3
a=3, b=7
```

此範例亦為傳值呼叫 (call by value)，所以雖然在 swap 函式裡成功將傳入的兩個變數交換了，在主程式中的兩個變數並沒有交換。

20

12-5.cpp

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

void swap(int &a, int &b);

int main() {
  int a = 3;
  int b = 7;

  cout << "a=" << a << ", b=" << b << endl;
  swap(a, b);
  cout << "a=" << a << ", b=" << b << endl;

  return 0;
}
```

21

12-5.cpp

```
void swap(int &a, int &b) {
    cout << a << ", " << b << endl;
    int tmp = a;
    a = b;
    b = tmp;
    cout << a << ", " << b << endl;
}
```

```
a=3, b=7
3, 7
7, 3
a=7, b=3
```

此範例亦為傳參考呼叫 (call by reference)，所以在 swap 函式裡成功將傳入的兩個變數交換了，在主程式中的兩個變數也成功地交換，因為函式裡的 a, b 為主程式中 a, b 的分身。

22

12-6.cpp

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

void swap(int *a, int *b);

int main() {
  int a = 3;
  int b = 7;

  cout << "a=" << a << ", b=" << b << endl;
  swap(&a, &b);
  cout << "a=" << a << ", b=" << b << endl;

  return 0;
}
```

23

12-6.cpp

```
void swap(int *a, int *b) {
  cout << *a << ", " << *b << endl;
  int tmp = *a;
  *a = *b;
  *b = tmp;
  cout << *a << ", " << *b << endl;
}
```

```
a=3, b=7
3, 7
7, 3
a=7, b=3
```

此範例亦為傳址呼叫 (call by address)，在 swap 函式得到兩個變數的地址，並將此兩地址上的變數值作交換，所以主程式中的兩個變數值也跟著被交換了。

24

小結

- 預設傳值呼叫的方法，我們只能傳回一個運算結果。
- 傳值呼叫不可能意外改到呼叫程式 (e.g. main) 內的變數值。
- 傳址或傳參考呼叫方法則可能可以傳回一個以上的運算結果，並可以改變呼叫程式內的變數值。

25

12-7.cpp

```
int main() {
  double r, area, perimeter;
  cout << "請輸入半徑:";
  cin >> r;

  circle(5.0, area, perimeter);     // 如何寫這個函式？

  cout << "面積 = " << area << endl;
  cout << "圓周 = " << perimeter << endl;

  return 0;
}
```

26

12-7.cpp

```
void circle(double r, double &area, double &perimeter)
{
    const double pi = 3.141592654;
    area = pi*r*r;
    perimeter = 2*pi*r;
}
```

因為在前面的呼叫，傳入的參數不是指標變數，但我們又要把計算結果回傳回去，所以必需使用傳參考的方式將計算結果回傳。

27

隨堂練習

1. 請寫一函式 cube，傳入一正方體的一邊長，並回傳正方體的體積、表面積、以及總邊長。
2. 請寫一主程式，利用 1) 函式作計算，並列出邊長為 1, 2, 3, … 18, 19, 20 的正方體的體積、表面積、以及總邊長。

28

```
1, 1, 6, 12
2, 8, 24, 24
3, 27, 54, 36
4, 64, 96, 48
5, 125, 150, 60
6, 216, 216, 72
7, 343, 294, 84
8, 512, 384, 96
9, 729, 486, 108
10, 1000, 600, 120
11, 1331, 726, 132
12, 1728, 864, 144
13, 2197, 1014, 156
14, 2744, 1176, 168
15, 3375, 1350, 180
16, 4096, 1536, 192
17, 4913, 1734, 204
18, 5832, 1944, 216
19, 6859, 2166, 228
20, 8000, 2400, 240
```

29

作業十

30

1. 二次多項式的根

- 請寫一函式 f，計算以下的函數值：
  $f(x) = ax^2 + bx + c$
- 請寫一函式 findRoot，其傳入 a, b, c 後會計算出此 f(x)=0 的兩個實根。findRoot函式具有以下原型宣告：

```
void findRoot(double a, double b, double c,
        doouble &r1, double &r2);
```

- 請寫一程式讓使用者輸入二次多項式中的係數 a, b, 與 c，由程式判斷所輸入的 a, b, c 是否具有實根。如果有實根則使用 findRoot 找出實根並利用 f 檢查計算出的根是否正確。

參考輸出

```
請輸二次多項式的係數: a, b, c: 2 6 2

此二次多項式的根:
r1: -0.381966
r2: -2.61803
f(r1)=2.22045e-016
f(r2)=0
```

```
請輸二次多項式的係數: a, b, c: 2 2 2
你所輸入a, b, c 之二次多項式沒有實根!

請輸二次多項式的係數: a, b, c: 3 4 1

此二次多項式的根:
r1: -0.333333
r2: -1
f(r1)=0
f(r2)=0
```