1

Lecture 11

指標與陣列

2

回顧

- 自訂函式由 1) 原型宣告、與 2) 函式定義等兩部份組成。
  - 原型宣告中包含了 a) 函式會回傳資料的型別、b) 函式名稱、以及 c) 函式需傳入的參數個數與個別的型別。

```
void funA();
void nStar(char symbol, int x=20);
double f(double);
int sum(int);
void do(int);
```

  - 函式定義則主要定義了函式的實際運作。
  - 自訂函式的原型宣告必需出現在第一次使用該函式前。
  - 函式定義可以代替原型宣告。

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

3

10-5.cpp

此處為函式定義 (function definition)

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

此行為函式的界面 (interface)
-編譯器會檢查與出現過的原型宣告是否一致
- 所有資料的「進出」都應透過此界面

此處為函式本體 (function body)
-定義實質函式的運作

注意到 sum 函式裡面一共宣告了三個變數: x, i, ans. 這三個變數稱為**區域變數**，僅作用於此函式內而不會影響到其它函式內相同名稱的變數。

4

10-7.cpp

執行結果

```
1, 2, 3, 4
100, 2, 3, 4
100, 200, 3, 4
100, 200, 300, 4
100, 200, 300, 400
```

```
#include <iostream>
using namespace std;
void test(int a=1, int b=2, int c=3, int d=4) {
  cout << a << ", " << b << ", ";
  cout << c << ", " << d << endl;
}
int main() {
  test();
  test(100);
  test(100,200);
  test(100,200,300);
  test(100,200,300,400);
  return 0;
}
```

錯誤宣告

```
void test(int a=1, int b, int c=2, int d) {...}
void test(int a=1, int b=2, int c, int d) {...}
```

錯誤呼叫

```
test(,,,100)
test(100,, 300, 400)
test(100, , , 400)
test(100, 200, 300, )
```

5

函式內之變數

- 函式內所宣告的變數稱為區域變數 (local variable) ，不同函式的區域變數各自獨立，可使用同樣名稱而不互相干擾。
- 因此，當在程式設計初期在分析時，即可將不同的工作步驟寫為不同函式並交由不同人去將函式實作出來，而不用耽心程式會互相衝突。

```
double power(double x, int n) {
      int i;
      double ans = 1;
      for(i=0;i<n;i++) {
            ans = ans * x;
      }
      return ans;
}
```

```
int doSum(int n) {
      int i;
      int ans = 0;
      for(i=1;i<=n;i++) {
            ans = ans + i;
      }
      return ans;
}
```

6

隨堂練習

a) 請寫一函式 bool isRightTriangle (int a, int b, int c); 用來判斷傳入的三角形三邊長 a, b, c, 是否構成直角三角形。若是直角三角形回傳 true，否則回傳 false.

b) 請寫一主程式，利用三層迴圈，找出所有整數邊長界於 1 – 50 之間的直角三角形。(Hint: 每一層迴圈代表一個邊長)

7

參考輸出

```
3       4       5
5       12      13
6       8       10
7       24      25
8       15      17
9       12      15
9       40      41
10      24      26
12      16      20
12      35      37
14      48      50
15      20      25
15      36      39
16      30      34
18      24      30
20      21      29
21      28      35
24      32      40
27      36      45
30      40      50
共有 20 個直角三角形
```

8

今日主題

- 指標 (pointer)
- 指標與陣列 (pointer vs. array)

9

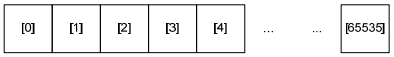
指標

Pointer

10

指標/指位器 (Pointer)

- 變數
  - int a; → 整數型別，名稱為 a
  - 變數是為了使用記憶體資源來儲存資料與進行運算
  - 所有的變數都佔有記憶體空間
- 記憶體
  - 可視為一個很大的一維陣列，單位是 byte



  - 1GB → 1,024 MB → 1,048,576 KB → 1,073,741,824 Bytes

11

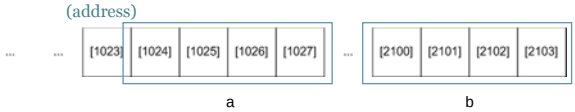
問題

- 一個 4KB 的電腦，其記憶體位置(編號)從 0 至？
  - 4 x 1024 – 1 = 4095
- 一台 1MB的電腦，其記憶體位置從 0 至？
  - 1 x 1024 x 1024 – 1 = 1048575

12

指標 (Pointer)

- 變數宣告
  - int a; → 整數型別，名稱為 a，由型別知其佔記憶體大小。sizeof()
  - float b; → 浮點數，名稱為 b ，由型別可知其佔記憶體大小。
- 記憶體
  - 在記憶體 (陣列) 裡每個元素都有一個索引 ← 稱為記憶體位置 (address)

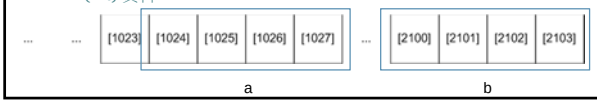


  - a 變數佔了四個 byte, 從位址 1024 開始
  - b 變數佔了四個 byte, 從位址 2100 開始

13

指標/指位器 (Pointer)

- 指標/指位器
  - `int a, b;`
  - `int *ptrA, *ptrB;`
  - ptrA 是一個變數，其型別為整數的指標 (int *)，此變數的值為一個記憶體位址 (address)，且該記憶體位址上所儲存的資料為整數 (int) 資料。我們常說 ptrA 指向一個整數。
  - ptrB 是一個變數，其型別為整數的指標 (int *)，此變數的值為一個記憶體位址 (address)，且該記憶體位址上所儲存的資料為整數 (int) 資料。



14

11-1.cpp

```
a=3
b=3
ptrA=0021F878
ptrB=0021F86C
```

```
#include <iostream>
using namespace std;
int main() {
  int a = 3;
  float b = 3;
  int *ptrA = &a;
  float *ptrB = &b;
  cout << "a=" << a << endl;
  cout << "b=" << b << endl;
  cout << "ptrA=" << ptrA << endl;
  cout << "ptrB=" << ptrB << endl;
  return 0;
}
```

- 每個變數「通常」有獨立的記憶體位置
- 在印出指標的值時，是以16進位方式印出的。

15

11-1.cpp 的說明

變數名稱	變數所佔記憶體位置	變數記錄的值
a	21F878	3
b	21F86C	3.0
ptrA	??????	0021F878
ptrB	??????	0021F86C

16

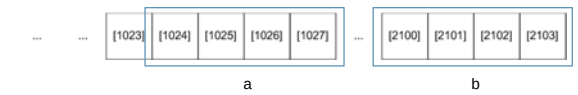
指標 (Pointer) 相關之運算子

- *****
  - 宣告時
    - int *ptrA; → 1) 宣告名稱為 ptrA 的變數為一指標變數，2) 其內容存放的值為一個記憶體位置，3) 其指到的位址所存放的資料為整數型別 int 的資料。
  - 執行時
    - C = (*ptrA); → 將 ptrA 所指到的資料取出並存入 C。
    - *ptrA = 3; → 將 3 存入 ptrA 所指到的變數裡。
    - * 稱為 dereference (反參照、或翻為**取值**運算子)

17

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)
  - 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。



18

11-2.cpp

```
int **pptrA;
```
- 宣告一變數，名稱為 pptrA
- pptrA 為一個指標變數
- 其指向的資料型別為 int*
- int* *pptrA;

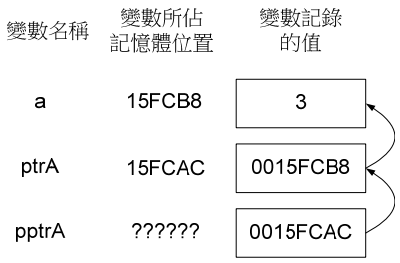
```
a=3
ptrA=0015FCB8
pptrA=0015FCAC
```

```
#include <iostream>
using namespace std;
int main() {
  int a=3;
  int *ptrA  = &a;
  int **pptrA = &ptrA;

  cout << "a=" << a << endl;
  cout << "ptrA=" << ptrA << endl;
  cout << "pptrA=" << pptrA << endl;

  return 0;
}
```

- 每個變數「通常」佔據獨立的記憶體位置。
- 每個變數都可以透過取址運算子得到其記憶體位址。

19

11-2.cpp 的說明



20

指標 (Pointer) 相關之運算子

- *
  - 宣告時
    - int *ptrA; → 1) 宣告名稱為 ptrA 的變數為一指標變數，2) 其內容存放的值為記憶體位置，3) 其指到的位址所存放的資料為整數型別的資料。
  - 執行時
    - (*ptrA); → 得到 ptrA 所指到的記憶體位址內的資料。
    - *ptrA = 3; → 將 3 存入 ptrA 所指到的記憶體位址。
    - * 稱為 dereference (翻為**取值**運算子)

21

11-3.cpp

```
#include <iostream>
using namespace std;
int main() {
  int a=3;
  int *ptrA = &a;

  cout << a;
  cout << ptrA;
  cout << *ptrA;
  *ptrA = 4;
  cout << a;

  return 0;
}
```

輸出 a 的值
輸出 ptrA 的值
輸出 ptrA 指到的記憶體的值
將 ptrA 指到的記憶體內容填入4
輸出 a 的值, 此時 a 的值為 ?

有了指標後，一個變數 (a) 可以被其它變數 (*ptrA) 所改變或使用。

22

練習

```
#include <iostream>
using namespace std;
int main() {
  float b;
  // 1. 請宣告一ptrB 變數, 其型別為指向 float 型別的指標
  // 2. 請宣告一ptrC 變數, 其型別為指向 int 型別的指標

  // 3. 請透過取址運算子(&), 將 b 變數的記憶體位置存入ptrB
  // 4. 請透過取址運算子(&), 將 b 變數的記憶體位置存入ptrC
  // 5. 請利用取值運算子(*), 將3.14 存入 ptrB 所指向的記憶體位置
  // 6. 請將 b 變數的值列印出來
  // 7. 請將ptrC 指標所指向記憶體位置之值列印出來
  return 0;
}
```

```
float *ptrB;
int *ptrC;
ptrB = &b;
ptrC=(int*)&b;
(*ptrB) = 3.14;
cout << b;
cout << (*ptrC);
```

23

請解釋或回答

- double *q;
- int a;
- int *b = &a;
- int *c = &b;
- *q = 3.1415;

- 以下兩個述敍何者 ok ?
  - c=6.02;
  - *c = 9.03;

24

陣列與指標

Array vs. Pointer

25

11-4.cpp

```
i=0, A[i]=5, &A[i]=0029FA28, b[i]=5
i=1, A[i]=4, &A[i]=0029FA2C, b[i]=4
i=2, A[i]=3, &A[i]=0029FA30, b[i]=3
i=3, A[i]=2, &A[i]=0029FA34, b[i]=2
i=4, A[i]=1, &A[i]=0029FA38, b[i]=1
```

```
#include <iostream>
using namespace std;
int main() {
  int A[5] = {5,4,3,2,1};
  int *b=&A[0];

  for(int i=0;i<5;i++) {
      cout << "i=" << i;
      cout << ", A[i]=" << A[i];
      cout << ", &A[i]=" << &A[i];
      cout << ", b[i]=" << b[i] << endl;
  }
  return 0;
}
```

26

11-4.cpp 的說明

- `int *b=&A[0]`
  令b指標指向A陣列的第1個元素(索引為0)，即其陣列的開頭
- `cout << ", &A[i]=" << &A[i];`
  可以利用取址運算子 `&` 取出陣列每個元素的記憶體位置，可以從程式執行結果發現陣列的每個相鄰元素其記憶體位置為連續的。
- `cout << ", b[i]=" << b[i] << endl;`
  使用指標時，除了可使用取值運算子 `*` 取得其指向的記憶體內的資料外，亦可把它當陣列使用，取出連續記憶體內的資料。

27

11-4.cpp 的說明

變數名稱	陣列元素	變數所佔記憶體位置	變數記錄的值
A	A[0]	0029FA28	5
	A[1]	0029FA2C	4
	A[2]	0029FA30	3
	A[3]	0029FA34	2
	A[4]	0029FA38	1
b		????	0029FA28

28

11-5.cpp

```
i=0, A[i]=5, &A[i]=0029FA28, b[i]=5
i=1, A[i]=4, &A[i]=0029FA2C, b[i]=4
i=2, A[i]=3, &A[i]=0029FA30, b[i]=3
i=3, A[i]=2, &A[i]=0029FA34, b[i]=2
i=4, A[i]=1, &A[i]=0029FA38, b[i]=1
```

```
#include <iostream>
using namespace std;
int main() {
  int A[5] = {5,4,3,2,1};
  int *b=A;

  for(int i=0;i<5;i++) {
      cout << "i=" << i;
      cout << ", A[i]=" << A[i];
      cout << ", &A[i]=" << &A[i];
      cout << ", b[i]=" << b[i] << endl;
  }
  return 0;
}
```

29

11-5.cpp

- 陣列 int A[20];
  - A[0], A[1], … A[19]
- A這個名字本身即為一個指標，指到陣列最開始元素位置
  - A[0] 與 *A 的動作完全一樣，即取得 A 這個指標所指到的位置上之資料
  - A[1] 則是取出 A 這個指標指向的元素的下一個元素資料 … 會等同於 *(A+1) ← 我們下次再講到指標的算術運算 …
- 指標可當成一維陣列來存取、使用

30

練習

```
int *b;
```

```
for(int i=0;i<10;i++) {
    cout << a[i] << " ";
}
```

```
b = &A[2];
```

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

int main() {
  int A[] = {1,2,3,4,5,6,7,8,9,10};
  //1. 宣告一個整數指標變數b(一指標變數,其指向資料為整數型別)

  //2. 請寫一個迴圈, 將 A 陣列完整列印出來
  //3. 請將 b 指標指向 A 陣列內第 3 個元素

  //4. 請寫迴圈, 把 b 指標當成一維陣列使用, 印出b[0]至b[4]

  return 0;
}
```

```
for(int i=0;i<5;i++) {
    cout << b[i] << " ";
}
```

31

指標與陣列

- 在前面已經看到我們可以把陣列的名字當指標用，也可以把指標當陣列使用。
- 所以，我們可以把陣列當成是指標變數傳遞給函式去使用。
- 在以下範例中，即將陣列當成是指標傳入函式中。由於指標指向陣列的開頭，函式裡可以存取到陣列中的每一個元素。

32

11-6.cpp

```
5 4 3 2 1
5 3 3 2 1
5 3 3
3 2
```

```
#include <iostream>
using namespace std;
void printArray(int n, int *a);
int main() {
  int A[5] = {5,4,3,2,1};

  printArray(5, A);
  A[1] = 3;
  printArray(5, A);
  printArray(3, A);
  printArray(2, &A[2]);

  return 0;
}
```

```
void printArray(int n, int *a) {
  for(int i=0;i<n;i++) {
    cout << a[i] << " ";
  }
  cout << endl;
}
```

33

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。