1

Lecture 15

函式覆載
遞迴函式
指標的算術運算

2

函式覆載

Function Overloading

3

函式覆載
Function overloading

- 函式的介面 (Interface) 由三部份組成
  - 傳回值型別 函式名稱 (參數 1 型別 參數1名稱, 參數 2 型別參數3名稱, 參數 3 型別 參數3名稱, 參數 4 型別 ...)

函式的原型宣告

```
void nStar(char symbol, int x=20);
```

函式的定義

```
void nStar(char symbol, int x){
  for(int i=0;i<x;i++) {
    cout << symbol;
  }
  cout << endl;
}
```

4

函式覆載
Function overloading

- 在 C 與大部份的程式語言，函式名稱不可以一樣。
  - double **sum**(const int N, double *A);
  - × int **sum**(const int N, int *A);
- 在 C++ 中，函式名稱可以重複，只要**函式名稱**與**參數個數及型別**的組合唯一即可。
  - ✓ double sum(const int N, double *A);
  - ✓ int sum(const int N, int *A);
  - ✓ int sum(int N, int A, int B, int C, int D);

5

15-1.cpp (1/2)

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

double area(double r) {
  return r * r * 3.1415926535897932385;
}
double area(double width, double height) {
  return width * height;
}
double area(double up, double down, double height) {
  return (up + down) * height * 0.5;
}
```

6

15-1.cpp (2/2)

```
int main() {
  cout << "r=5 的圓面積為" << area(5) << endl;
  cout << "10x20 的矩形面積為" <<area(10, 20)<< endl;
  cout << "上底2, 下底4, 高為3 的梯形面積為" <<
          area(2, 4, 3) << endl;

  return 0;
}
```

r=5 的圓面積為78.5398
10x20 的矩形面積為200
上底2, 下底4, 高為3 的梯形面積為9

在此範例中，有三個同名的函式 area，但它們的參數個數不一樣，所以編譯器不會弄不清楚應該執行那一個函式。

7

函式覆載
Function overloading

- **由前例可以看出，在 C++ 裡函式名稱可以不是唯一。其限制：**
  - 函式名稱不同時，沒有問題。
  - 函式名稱相同時，**沒有預設值**的參數個數不同時，沒有問題。
  - 函式名稱相同、參數個數相同、任一參數或以上之型別不同時，沒有問題。
  - 函式名稱相同、參數個數相同、且型別皆相同時則不行。
  - 亦即：函式名稱與傳入參數個數與到別之組合必需唯一。
  - **總而言之，要讓編譯器不會無法決定應該要呼叫那一個函式！**
- **注意，如果不同函式之間僅有回傳參數之型別不同時，不可共存於同一個程式中。編譯器無法僅靠回傳形別區分不同函式。**
  - `void myFunc(int, int, int);`
  - `int myFunc(int, int, int);`

8

範例

```
void myFunc();
void myFunc(int);
void myFunc(int, int, int);
void myFunc(double, int, int);
void myFunc(double, double, double);
int myFunc(float, double, int);
void myFunc(int, int, int, int);
void myFunc(double, double, double, double);

int myFunc();
double myFunc(double, double, double);
```

此兩個函式原型宣告不合法，會和前面的衝突。

9

15-2.cpp

此例中，編譯器無法判斷是要呼叫
`area(5.0)`
或是
`area(5.0, 1.0)`

```
#include <iostream>
using namespace std;
double area(double r) {
  return r * r * 3.1415926535897932385;
}
double area(double width, double height=1.0) {
  return width * height;
}
int main() {
  cout << "area(5) = " << area(5) << endl;
  cout << "area(10, 20) = " << area(10, 20) << endl;
  return 0;
}
```

10

pow 函式

- 在標準函式庫中的 pow 函式，亦為一個多重覆載的函式，它有以下的原型宣告：

```
long double pow ( long double base, long double exponent );
long double pow ( long double base, int exponent );
double pow ( double base, double exponent );
double pow ( double base, int exponent );
float pow ( float base, float exponent );

pow(3.0, 4.0)
pow(3.0, 4)
pow(3, 4.0)
pow(3, 4)
pow(3.0f, 4)
pow(3.0f, 4.0f)
```

此函式呼叫會有問題，會無法判斷要型別轉換為
a) pow(double, double)
b) pow(double, int)
c) pow(float, float)

11

遞迴函式

Recursive Function

12

遞迴函式
Recursive Function

- 何謂遞迴函式?
  - 一個函式呼叫本身，即被稱為遞迴函式。
  - 透過遞迴 (recursion) 的技巧來解決計算上的問題。
  - 有些數學上的定義本來就以遞迴定義。
    - 等差級數、 $a_i = a_{i-1} + b$
    - 等比級數、 $a_i = b \times a_{i-1}$
    - Fibonacci 級數、 ... $a_i = a_{i-1} + a_{i-2}$
- 範例：
  - 15-4.cpp: 計算階乘
    - 5! = 5 * 4!、4! = 4 * 3!、3! = 3 * 2! 、...
  - 15-5.cpp: Fibonacci 數列
    - 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ...

13

15-4.cpp

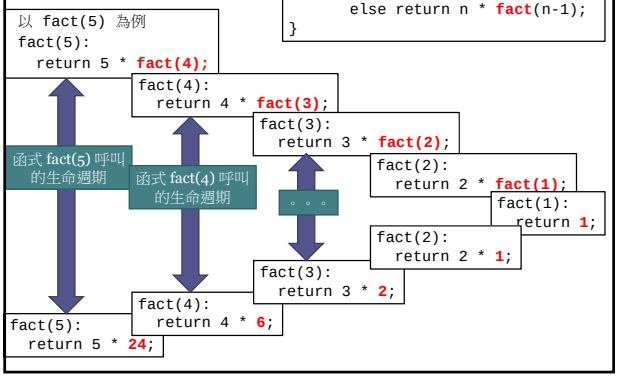
```
#include <iostream>
using namespace std;
int fact(int n) {
  if(n==1) return 1;
  else return n * fact(n-1);
}
int main() {
  int i;

  for(i=1;i<=10;i++) {
      cout << i << "! = " << fact(i) << endl;
  }
  return 0;
}
```

```
1! = 1
2! = 2
3! = 6
4! = 24
5! = 120
6! = 720
7! = 5040
8! = 40320
9! = 362880
10! = 3628800
```

14

15-4.cpp

```
int fact(int n) {
      if(n==1) return 1;
      else return n * fact(n-1);
}
```



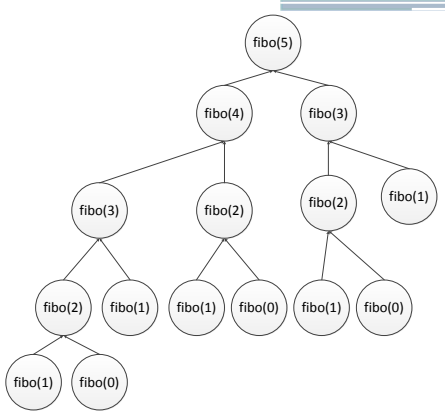
15

15-5.cpp

```
#include <iostream>
using namespace std;
int fibo(unsigned int n) {
  if(n==0) return 0;
  else if(n==1) return 1;
  else return fibo(n-1) + fibo(n-2);
}

int main() {
  cout << fibo(5) << endl;

  return 0;
}
```

```
1
1
2
3
5
8
13
21
34
55
```

16



17

撰寫遞迴函式的注意事項

- 遞迴函式必需有「終止」條件，電腦不能無窮遞迴下去!
  - if(n==1) return 1; else ...
  - if(n==0) return 0; else if(n==1) return 1; else return ...
- 遞迴的「深度」不能太深，電腦程式執行會發生錯誤
  - stack overflow! (堆疊溢位)
- 遞迴的寫法會讓程式看起來比較簡短，但執行效率不見得比較高；遞迴常可使用迴圈取代。

18

指標的算術運算

19

指標

- 用來存取記憶體內容的工具
  - `int A=4, *ptrA = &A;`
  - `cout << *ptrA;` `// 印出 4`
  - `*ptrA = 7;`
  - `cout << A;` `// 印出 7`
- 一維陣列 vs. 指標
  - `int a[5] = {1, 2, 3, 4, 5};`
  - `int *ptrA = a;`
  - `cout << ptrA[3];` `// 印出 4`
- 我們也可以對指標作加減的計算

20

15-6.cpp

```
0013F7A0:1
0013F7A8:2
0013F7B0:3
0013F7B8:4
0013F7C0:5
```

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

int main() {
      double q[] = {1,2,3,4,5};
      double *ptr = q;
      for(int i=0;i<5;i++) {
            cout << ptr << ":" << *ptr << endl;
            ptr++;
      }

      return 0;
}
```

21

指標的算術運算

- 指標在作加減法時，是以指標所指向的資料型別決定它實際的位址變化，以方便指向陣列中的資料：
  - double 佔 8 個 bytes，位址變化以 8 bytes 為單位
  - int 佔 4 個 bytes，位址變化以 4 bytes 為單位

```
int a[10] = {1,2,3,4,5,6,7,8,9,10};
int *b = a;           // b 指標指向 1
b++;                  // b 指標指向 2
b++;                  // b 指標指向 3
b+=2;                 // b 指標指向 5
b-=3;                 // b 指標指向 2
cout << b[3];         // 印出 5 !
```

22

15-7.cpp

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

int main() {
  double q[] = {1,2,3,4,5,6,7,8,9,10};
  double *ptr = q + 9;;
  for(int i=0;i<10;i++) {
    cout << ptr << ":" << *ptr << endl;
    *ptr+=10;

    ptr--;
  }
  for(int i=0;i<10;i++)
    cout << q[i] << " ";
  return 0;
}
```

```
002BF940:10
002BF938:9
002BF930:8
002BF928:7
002BF920:6
002BF918:5
002BF910:4
002BF908:3
002BF900:2
002BF8F8:1
11 12 13 14 15 16 17 18 19 20
```

23

```
   int a[10] = {1,2,3,4,5,6,7,8,9,10};
   int *pa = a, *pb = a+9, *pc = &a[5];

1  cout << *pa++;
6  cout << *pc;
9  cout << --*pb;
   pa += 5;
7  cout << *pa--;
   pb -= 2;
8  cout << *pb;
6  cout << *pc++;
6  cout << --*pc;
9  pa += 3; cout << *pa;
3  pb -= 5; cout << *pb;
3  pc -= 4; cout << *pc;
```

24

指標相減

- 當拿兩個指標相減時，會得到介於兩個指標之間的資料個數 (而不是記憶體位址的差別)
- Example:

```
int a[10] = {1,2,3,4,5,6,7,8,9,10};
int *b = &a[3];
int *c = &a[8];
cout << c – b << endl;    // 印出 5
亦即 c == b+5
```

25

隨堂練習

26

指標運算

- 請利用指標運算的方式，完成以下函式的定義 (主程式已貼在 BB 上)

```
void printArray(int n, int *a);
void reverse(int n, int *a, int *b);
int even(int n, int *a, int *b);
```

- 其中 printArray 用來印出 a 陣列中 頭 n 個元素
- reverse 將 a 陣列的內容反過來存到 b 陣列裡
- even 用來將 a 陣列中的頭 n 個元素其值為偶數的放到 b 陣列中，並回傳偶數個數。

27

主程式與輸出結果

```
int main() {
      int a[10] = {1,2,3,4,5,6,7,8,9,10};
      int b[10];
      int c[10];

      printArray(10, a);
      reverse(10, a, b);
      printArray(10, b);
      int nEven = even(10, a, c);
      printArray(nEven, c);

      return 0;
}
```

```
1 2 3 4 5 6 7 8 9 10
10 9 8 7 6 5 4 3 2 1
2 4 6 8 10
```