1

Lecture 08

函式

2

目前學習到的內容

這些內容也是在接觸一個新程式語言時除了語法外，所需學習的主題

- 前半學期主要介紹的在流程的建立與記憶體內資料的儲存與應用
- 流程
  - 循序執行
    指令由上而下逐行執行 → 流程
  - 分支: if(...) { ... } else ..., switch ... case ...
    指令因為不同的條件而執行不同的分支或不同的流程
  - 迴圈: for、while、do ... while ...
    根據條件判斷的成立與否決定是否要執行或重複執行迴圈內的流程
- 資料
  - 變數：變數具有名稱與型別，不同型別有其最適合儲存的資料
  - 陣列：多筆資料的組合，並可以依據需求而有不同的排列方式
    - 線性排列、數列：一維陣列
    - 矩陣排列：二維陣列
    - 多維陣列

3

函式

function

4

函式簡介

- 在程式語言上，函式 (function) 為獨立的程式運作單元 (unit)，或稱為模組 (module)、副程式 (subroutine)、程序 (procedure)、 ...
- 函式常用來包裝某一流程並給予名稱，使主程式看起來較為簡單易懂

```
answer=0;
for(int i=0;i<100;++i) {
    answer += a[i];
}
```

→

```
answer = sum(a, 100);
```

- 函式是用來創造「黑箱」(black box) – 只看得見進去黑箱的東西，和黑箱產出的東西，黑箱裡怎麼運作是看不到、也不想看到的 ...

```
y = exp(x);
```

- 當需要使用函式時，會需要兩個要件：
  - 函式的原型宣告 (prototype) → 常由 標頭檔 (header file) 取得
  - 函式的定義 (definition) → 常由函式庫 (library) 取得

5

1-1.cpp

```
#include <iostream>
```

← 將 iostream 這個標頭檔內含有以定義標準輸入輸出之類別、物件、與函式等等的宣告。

```
using namespace std;
```

← 標準函式庫裡的東西都定義在 std 這個命名空間裡, 用來避免名稱的衝突

```
int main() {
    // 輸出 "Hello World" 至螢幕上
    cout << "Hello World";

    /* 函式執行完畢，回傳0 */
    return 0;
}
```

← 使用標準程式庫的 cout 物件輸出

6

函式的來源

- 標準函式庫 (standard library)
  - 為程式語言制定者/組織/委員會認為必要提供的基本功能
  - e.g. 基本輸入輸出功能 (cin, cout)
- 自訂函式
- 第三方 (third party) 提供免費的或商業的程式庫
  - e.g. SVG 繪圖、 OpenGL、Unity3D、 ...

7

C/C++ 標準函式庫

C/C++ Standard Library

8

C/C++ 標準函式庫 (C與C++通用函式庫)

http://www.cplusplus.com/reference/clibrary/

C++標頭檔	用途	C++標頭檔	用途
<cmath>	**數學運算函式庫**	<cassert>	診斷/除錯函式庫
<cstdlib>	**標準函式庫**	<cerrno>	錯誤狀態函式庫
<cstring>	**字串函式庫**	<csignal>	訊號的處理函式庫 (e.g. CTRL-BREAK)
<cstdio>	標準輸入輸出函式庫		
<ctime>	時間函式庫		
<cstdarg>	不定個數的函式參數處理函式庫	<cctype>	字元處理
<climits>	整數的極限定義	<clocale>	在地化/本土化
<cfloat>	浮點數的特性定義	<cuchar>	萬國碼字元轉換函式庫
<cstdint>	一些額外的標準準數型別的定義	<cwchar>	寬字元型別操作函式庫
<cstddef>	一些額外標準型別的定義	<cwctype>	寬字元型別轉換函式庫

9

C++ 容器類別庫

C++標頭檔	功能概述
<array>	陣列容器類別庫
<deque>	雙端佇列容器類別庫
<forward_list>	單向鍊結容器類別庫
<list>	雙向鍊結容器類別庫
<map>	映射容器類別庫
<queue>	單端佇列容器類別庫
<set>	集合 (樹狀結構) 容器類別庫
<stack>	堆疊容器類別庫
<unordered_map>	雜湊映射容器類別庫
<unordered_set>	雜湊集合容器類別庫
<vector>	向量容器類別庫

- 容器類別都是用來像陣列一般地儲存大量資料所使用的。
- 這些類別的實作方法會在「資料結構與演算法」課程介紹其資料儲存的方法與相關操作的演算法。
- 這些類別如果會使用的話，會比使用之前所介紹的陣列方便許多。

10

C++ 輸入輸出類別庫

C++標頭檔	功能概述
<fstream>	檔案串流類別庫
<iomanip>	輸入輸出操作類別庫
<ios>	輸入輸出基礎類別庫
<iosfwd>	
<iostream>	標準輸入輸出物件庫
<istream>	輸入串流類別庫
<ostream>	輸出串流類別庫
<sstream>	字串串流類別庫
<streambuf>	串流緩衝區類別庫

- 這些類別是用來在C++中進行輸入輸出所使用的。
- C語言則是使用定義在<cstdio>裡的函式進行類似的操作。
- 我們已經使用了其中的 <iostream>
- 之後如果時間夠時會介紹如何使用 <fstream> 裡的類別進行檔案操作。

11

其它C++類別庫

C++標頭檔	功能概述	C++標頭檔	功能概述
<algorithm>	演算法函式庫	<complex>	複數類別庫
<functional>	泛函類別庫	<ratio>	分數運算類別庫
<chrono>	時間類別庫	<limits>	數值極限類別庫
<iterator>	迭代器類別庫	<string>	C++字串類別庫
<random>	亂數產生器類別庫	<tuple>	多元組類別庫
<regex>	正則運算式類別庫	<valarray>	數值陣列類別庫
<initializer_list>	初始化用類別庫	<numeric>	通用數值操作函式庫
		<utility>	工具類別函式庫
<memory>	動態記憶體管理類別庫		
<new>	動態記憶體管理函式庫	<locale>	本土化/在地化類別庫
<system_error>	系統錯誤處理類別庫	<typeindex>	號轉數字類別庫
		<typeinfo>	資料型別資訊類別庫
<exception>	例外處理函式庫	<type_traits>	編譯時期取得變數型別類別庫
<stdexcept>	標準例外處理類別庫		

12

數學函式庫

#include <cmath>

注意，在電腦裡的三角函數以**弧度**為單位，而非我們一般使用的角度。

- 角度：以 360 度為一個圓
- 弧度：以 2π 為一個圓
- 角度轉弧度
  - 角度 * π / 180
- 弧度換角度
  - 弧度 * 180 / π

分類		
三角函數	sin	asin
	cos	acos
	tan	atan, atan2
	sinh, cosh, tanh	
指數與對數	exp	log
		log10
次方與根號	pow	sqrt
進位與捨位	ceil	floor
取絕對值	abs, fabs, ...	

http://www.cplusplus.com/reference/clibrary/cmath/

13

8-1.cpp

```
#include <iostream>
#include <iomanip>
#include <cmath>
using namespace std;
int main() {
  double theta;
  double rad;
  double coeff = 3.14159265358979323846 / 180.0;
  for (theta = 0; theta <= 90; theta = theta + 10) {
      rad = theta * coeff;
      cout   << setw(5) << theta << ": "
             << setw(12) << sin(rad) << ", "
             << setw(12) << cos(rad)
             << endl;
  }
  return 0;
}
```

程式執行結果

```
 0:           0,           1
10:    0.173648,    0.984808
20:     0.34202,    0.939693
30:         0.5,    0.866025
40:    0.642788,    0.766044
50:    0.766044,    0.642788
60:    0.866025,         0.5
70:    0.939693,     0.34202
80:    0.984808,    0.173648
90:           1, 6.12323e-017
```

14

8-2.cpp

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

int main() {
  double a;
  a = pow(2.0, 3);          // 計算 2 的 3 次方
  cout << "2^3 = " << a;
  cout << "\nlog10(2) = " << log10(2.0);
  cout << "\nlog10(2^3) = " << log10(a);
  cout << "\nsqrt(3.0) = " << sqrt(3.0);
  cout << "\npow(3.0, 0.5) = " << pow(3.0, 0.5);
  return 0;
}
```

程式執行結果

```
2^3 = 8
log10(2) = 0.30103
log10(2^3) = 0.90309
sqrt(3.0) = 1.73205
pow(3.0, 0.5) = 1.73205
```

15

8-3.cpp

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

int main() {
  double a;
  a = log10(3.0);
  cout << "a = log10(3.0) = " << a << endl;
  cout << "pow(10.0, a) = " << pow(10.0, a) << endl;
  cout << "exp(1)=" << exp(1.0) << endl;
  cout << "log(exp(1)) = " << log(exp(1.0)) << endl;

  return 0;
}
```

程式執行結果

```
a = log10(3.0) = 0.477121
pow(10.0, a) = 3
exp(1)=2.71828
log(exp(1)) = 1
```

16

8-4.cpp

```
#include <iostream>
#include <cmath>
using namespace std;
int main() {
  double a = sqrt(2.0);
  cout << "a=sqrt(2) = " << a << endl;
  cout << "floor(a) = " << floor(a) << endl;
  cout << "ceil(a) = " << ceil(a) << endl;
  a=-a;
  cout << "a=-sqrt(2) = " << a << endl;
  cout << "floor(a) = " << floor(a) << endl;
  cout << "ceil(a) = " << ceil(a) << endl;
  cout << "fabs(a) = " << fabs(a) << endl;
  return 0;
}
```

```
a=sqrt(2) = 1.41421
floor(a) = 1
ceil(a) = 2
a=-sqrt(2) = -1.41421
floor(a) = -2
ceil(a) = -1
fabs(a) = 1.41421
```

floor: 傳回最接近且小於傳入參數的整數
ceil: 傳回最接近且大於傳入參數的整數
fabs: 取絕對值

17

C 標準函式庫

#include <cstdlib>

分類	函式名稱	說明
C 字串轉換	atof, strtod	字串轉浮點數
	atoi	字串轉整數
	atol, strtol	字串轉長整數
亂數	srand	設定亂數產生的種子
	rand	產生一亂數
整數相關	abs, labs	取整數的絕對值
進階函式	bsearch	在陣列中找尋資料
	qsort	對陣列中的資料排序
重要常數	RAND_MAX	亂數的最大值

http://www.cplusplus.com/reference/clibrary/cstdlib/

18

8-5.cpp

```
#include <iostream>
#include <cstdlib>
using namespace std;
int main() {
  int i, seed;
  cout << "請輸入亂數種子: ";
  cin >> seed;
  srand(seed);
  for(i=0;i<10;i++) {
      cout << rand() << "," << rand()%6 << endl;
  }
  return 0;
}
```

```
請輸入亂數種子: 1
18467,5
26500,4
15724,5
29358,0
24464,4
28145,5
16827,1
491,1
11942,1
5436,3
```

```
請輸入亂數種子: 2
29216,3
17795,0
19650,0
26431,4
18316,1
28189,1
606,4
17829,5
30367,5
28961,4
```

使用取餘數的方式限制得到的亂數大小
此方法適用於整數!

srand() 用來設定亂數的種子，而之後透過 rand() 得到的亂數值則根據此種子所決定的。

19

8-6.cpp

```
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;
int main() {
  int i;
  srand((unsigned int) time(0));
  cout << endl;
  for(i=0;i<10;i++) {
      cout << ((double) rand())/RAND_MAX*6-3 << endl;
  }
  return 0;
}
```

```
-0.259926
-2.30583
-2.79821
0.261025
2.25968
1.96725
-2.95935
-1.686
-1.23719
0.356243
```

```
-0.250404
-0.816401
1.53841
-1.24104
2.47813
1.47139
0.159215
1.73818
-0.863277
0.34434
```

此範例中使用 time(0) 取得目前電腦的時間，並用來作為亂數種子。

在此利用**RAND_MAX**為亂數最大可能值，來限制亂數的範圍為 0 - 1 之間 (含)，再作放大 (*6) 與平移 (-3) 使生出來的亂數在 -3 ~ 3 之間

22

8-8.cpp

```
Hello
Hello World
str3 == str4
Hello World! Hell
```

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

int main() {
  char str1[] = "Hello";
  char str2[] = "World";
  char str3[100];
  char str4[]="Hello World";

  strcpy(str3, str1);
  cout << str3 << endl;

  strcat(str3, " ");
  strcat(str3, str2);
  cout << str3 << endl;

  if(strcmp(str3, str4)==0)
    cout<<"str3 == str4\n";

  strcat(str3, "! ");
  strncat(str3, str1, 4);
  cout << str3 << endl;

  return 0;
}
```

20

8-7.cpp

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

int main() {
    char str[500];
    int a;
    double b;

    cout << "請輸入一字串: ";
    cin >> str;
    a = atoi(str);
    b = atof(str);

    cout << "a: " << a;
    cout << "\na%6=" << a%6;

    cout << "\nb: " << b;
    cout << "\nb/7=" << b/7;

    return 0;
}
```

```
請輸入一字串: 12.3456
a: 12
a%6=0
b: 12.3456
b/7=1.76366
```

23

自訂函式

user-defined functions

21

C 字串函式庫

#include <cstring>

- 用來對C-字串 (字元的一維陣列) 進行操作處理
- 不能把這些函式套用在 string 型別的C++字串變數上！

函式名稱	說明
strncpy	字串拷貝
strncat	字串串接
strcmp	傳回字串比較結果
strlen	傳回字串長度
strchr	在字串中找尋一字元,並傳回其位置
strrchr	從字串尾端開始往回找尋一字元,並傳回其位置
strstr	在字串中找尋一字串,並傳回其起始位置
strtok	將字串依指定字元符號作切割

http://www.cplusplus.com/reference/clibrary/cstring/

24

自訂函式 (user-defined functions)

- 函式是一個程式的單元 (unit) 或是模組 (module)，為多行程式敍述的組合，用來**簡化**程式開發。
  - **常常**需要加總一個陣列
  - **常常**需要從一個陣列裡找最大值
  - **常常**需要計算一個函數的值
- 我們可以把自訂函式想像成是在自己創造程式裡的指令，而運用我們自己創造出來的「指令」(其實是自訂函式)，我們可以「比較容易寫出」較為複雜的程式。

25

8-9.cpp

執行結果
M1 M2 A1 B1 A2 M3 B1 M4

其中 funA() 和 funB() 有沒有很像兩個新的指令呢？

```
#include <iostream>
using namespace std;
1 void funA();
  void funB();

int main() {
  cout << "M1 ";
  cout << "M2 ";
2 funA();
  cout << "M3 ";
2 funB();
  cout << "M4 ";

  return 0;
}
```

1 函式的原型宣告 (prototype)

2 函式的呼叫 (invocation, call)

3 函式的定義 (definition)

```
3 void funA() {
    cout << "A1 ";
2   funB();
    cout << "A2 ";
  }
  void funB() {
    cout << "B1 ";
  }
```

26

8-10.cpp

記得電腦是由上而下在看你的程式，所以在此程式中原型宣告的目的之一在於告訴電腦有一個叫作 f 的函式，這個函式在呼叫時需要傳入一個 double 型別的參數，並且這個函式執行完後，會回傳一個 double 的資料。

```
#include <iostream>
using namespace std;
double f(double);   ←函式的原型 (prototype)宣告
int main() {
  double a, q;
  for(q=-5;q<=5;q++) {
    a = f(q);
    cout << q << ", " << a << endl;
  }
  return 0;
}
double f(double x) {
  return x*x+3*x+2;
}
```

函式呼叫 (function call, function invocation)

←函式的定義

執行結果

-5, 12
-4, 6
-3, 2
-2, 0
-1, 0
0, 2
1, 6
2, 12
3, 20
4, 30
5, 42

27

8-10.cpp 的說明

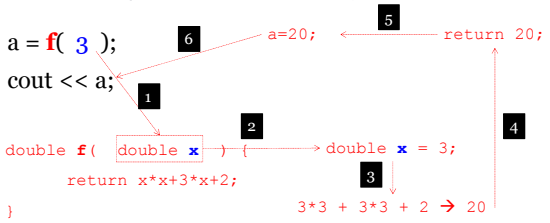
- 程式中 f(x) 即為一個函式 (類似於數學上的函數)

```
double f(double x) {
    return x*x+3*x+2;
}
```

- 函數傳入 0 或多個參數 (parameter, argument)，運作完成後回傳 (return) 0 或 1 個計算結果。
  - 在以上範例，傳入一個參數 x, 並傳出 x^2+3x+2 的計算結果。

28

函式呼叫時所發生的事情:



29

8-11.cpp

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

int sum(int);

int main() {
  cout << "sum(10)=" << sum(10) << endl;
  cout << "sum(100)=" << sum(100) << endl;

  return 0;
}
```

原型宣告的目的在於知會編譯器
1. 函式名稱 (sum)
2. 函式的傳入參數個數(1 個)
3. 個別參數的型別 (int)
4. 函式回傳的資料的型別 (int)

至於個別參數的名字不是很重要，可省略。但寫的好處是別人看到變數的名稱就大概知道要傳什麼資料進去。

此處為函式呼叫，編譯器會根據原型宣告檢查
1. 參數個數
2. 個別傳入參數的型別是否正確

30

8-11.cpp (continued)

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
- 注意到參數應給一變數名稱，以供函式內執行時使用這個變數的名字可以和原型宣告時使用的名字不同。

此處為函式本體 (function body)
- 定義實質函式的運作 (由傳入的參數到傳出的結果之流程)

注意到 sum 函式裡面一共宣告了三個變數: x, i, ans. 這三個變數被稱為**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。

31

自訂函式

- 自訂函式可分為兩部份：
  - 原型宣告 (prototype declaration)
  - 函式定義 (function definition)
- 自訂函式寫出來應該都會被呼叫到 (不然為什麼要寫它？)
  - 函式呼叫 (function invocation)

32

原型宣告
prototype declaration

- 原型宣告的目的在於告知編譯器一個函式的：
  1. 函式名稱
  2. 函式的傳入參數個數
  3. 個別參數的型別
  4. 函式回傳的資料的型別
- 原型宣告必需在函式第一次被使用 (被呼叫) 之前出現
- 原型宣告時之語法 (詳細說明見函式定義)
  回傳資料型別 函式名稱 (*參數列*) ;

```
double f(double);
int sum(int);
void do(int);
```

- 原型宣告之參數列個別參數的名稱不重要，可有可無。

33

原型宣告
prototype declaration

- 以下各框框內的原型宣告皆代表同一函式，不同框框代表不同函式的原型宣告。注意到原型宣告以分號作結尾。
- 在宣告函式時，可在最前面加上 inline ，此關鍵字提示編譯器此函式可考慮將其內容作「行內展開」 (inline expansion) 在呼叫函式處，以期加速程式的執行。
- 注意到相同名稱、傳入參數型別不同時，視為不同函式。

```
inline int sum(int);
inline int sum(int i);
inline int sum(int j);
inline int sum(int k);
```

```
double f(double);
double f(double x);
double f(double xyz);
double f(double qq);
```

```
double f(float);
double f(float x);
double f(float xyz);
double f(float qq);
```

34

函式定義 - overview
function definition

函式定義 (function definition) 語法如下：

回傳資料型別 函式名稱 (*參數列*) {
函式本體 (function body)
}

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

```
int plus(int x, int y) {
  return x+y;
}
```

```
void show(int x, int y) {
  cout << x+y;
}
```

35

函式定義 - return data type
function definition

- **回傳資料型別** (return datatype)
  - 定義了函式回傳的資料型別 (double, int, int*, float*, char, ...)
  - 亦可宣告為 void, 代表無回傳資料

```
int plusOne(int x) {
    int y = x + 1;
    return y;
}
```

```
void printTwoNumbers(int x, int y) {
    cout << x << "+" << y << "=" << x+y << endl;
}
```

36

函式定義 - function name / function identifier
function definition

- **函式名稱** (function identifier)
  - 函式名稱的限制同變數名稱的限制
  - 可使用英文字母、底線符號、與數字
  - 第一個字必需是英文字母或是底線符號

```
void printTwoNumbers(int x, int y) {
    cout << x << "+" << y << "=" << x+y << endl;
}
```

```
int plusOne(int x) {
    int y = x + 1;
    return y;
}
```

37

函式定義 - arguments
function definition

- **參數列 (argument list / parameters)**
  - 定義了函式執行時所需要傳入的資料數量與每個資料的型別。
  - 亦可宣告為 void 或省略, 代表無傳入的資料。
  - 參數列所列之變數可視為函式內的部份**變數宣告**
  - **亦可為參數設定內定值／預設值**

```
void helloWorld(void) {
    cout << "Hello World";
}
```

```
void printTwoNumbers(int x, int y=5) {
    cout << x << "+" << y << "=" << x+y << endl;
}
```

38

函式定義 - function body
function definition

- **函式本體 (function body)**
  - 定義了函式被呼叫時的執行動作、或是由輸入參數計算成回傳結果中間的流程
  - 可宣告自己所需使用的變數，稱為區域變數 (local variable)
  - 透過 return 指令將資料回傳給呼叫的程式 (caller)，並繼續執行呼叫者下一行的程式
  - 若函式無回傳值，則無需有 return 指令出現。

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

```
void hi(char *x) {
  cout << x << endl;
}
```

```
void hi(char *x) {
  cout << x << endl;
  return
}
```

39

自訂函式

- 自訂函式分為兩部份：**原型宣告**與**函式定義**。
- 若函式定義出現在第一次使用前，則原型宣告可省略
  - i.e. 函式定義直接作為原型宣告。
- 函式原型宣告與函式定義應一致 (名稱、參數個數、回傳型別 ...)
  - 原型宣告時參數名稱不重要，可以和函式定義的參數名稱不同。
  - 一個程式裡可使用多個函式，函式裡也還可以再呼叫其它函式
  - main () 事實上也是一個函式，只是它代表了程式主要開始的地方。

```
int sum(int);
int sum(int i);
int sum(int j);
int sum(int k);
```

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

40

8-12.cpp

```
#include <iostream>
using namespace std;
void nStar(char symbol, int x=20) {
  for(int i=0;i<x;i++) cout << symbol;
  cout << endl;
}

int main() {
  for(int i=1;i<=5;i++)
    nStar('+', i);
  cout << "\n";
  for(int j=0;j<5;j++)
    nStar('*');
  return 0;
}
```

執行結果

```
+
++
+++
++++
+++++

********************
********************
********************
********************
********************
```

在此程式中，函式 nStar 定義在第一次使用之前，故可以省略函式的原型宣告，並**以函式定義作為函式的原型宣告**。

在函式被執行時，傳入的變數名稱不重要、重要的是位置。第一個位置的資料被存入函式內的 symbol 變數；第二個位置的資料被存入函式內的 x 變數

41

參數預設值

- 在前範例中，我們給了第二個參數一個預設值 20，因此當呼叫此函式而沒有給第二個參數時，nStar函式執行時，會令 x 為 20 來執行函式的內容。
  - nStar('*');
- 定義函式時，一個參數若有預設值，則其之後的所有參數都必需有預設值!!
- 在函式呼叫時，任意兩逗點間或逗號與括號間都必需有值，因此有預設值的參數必需集中於參數列的後面
  - nStar(, '+');

```
void nStar(int x=20, char symbol) {
  for(int i=0;i<x;i++)
    cout << symbol;
  cout << endl;
}
```

42

8-13.cpp

```
#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;
}
```

執行結果

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

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, )
```

43

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;
}
```

44

使用函式的好處

- 將具有特定功能的敍述組合獨立為函式，可提高程式的可讀性。
- 也可把函式想成是自己在創造程式語言的指令。
  - e.g. nStar
- 將程式模組化，讓程式可重複使用 (code reuse)，可提升程式寫作的效率。
- 將程式分解為多個獨立函式，當有錯誤時，可較容易找出問題在那一個函式，提高除錯的效率。
- 各函式間互相獨立,可由不同程式設計人員完成。

45

8-14.cpp

```
#include <iostream>
using namespace std;
bool isTriangle(double, double, double);
int main() {
    double a, b, c;
    cout << "請輸入三邊長: ";
    cin >> a >> b >> c;
    cout << a << ", " << b << ", " << c;
    if( isTriangle(a, b, c) )
        cout << " 可";
    else
        cout << " 不可";
    cout << "構成一個三角形。";

    return 0;
}
```

46

8-14.cpp (continuted)

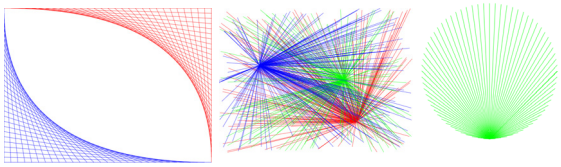
```
bool isTriangle(double a, double b, double c) {
    if(c >= (a+b)) return false;
    if(a >= (b+c)) return false;
    if(b >= (a+c)) return false;
    if((a-b) >= c) return false;
    if((b-c) >= a) return false;
    if((a-c) >= b) return false;

    return true;
}
```

雖然函式只能回傳一個值，但可以有多個 return 出現在函式中。

47

作業七

Due: 5/22/2015



48

7-1.cpp: 自訂函式

- 請撰寫兩函式，其原型宣告如下：
  double min(double x, double y, double z);
  double max(double x, double y, double z);
- 其中，min 函式會回傳所傳入三個數字中最小者， 而 max 函式會回傳三個數字中最大者。
- 請撰寫一主程式，運用前面所寫出來的函式，讓使用者輸入三個數字後，印出來其中最大值與最小值。

請輸入三個數字：1 1 5

您輸入的數字中，最大為：5, 最小為：1

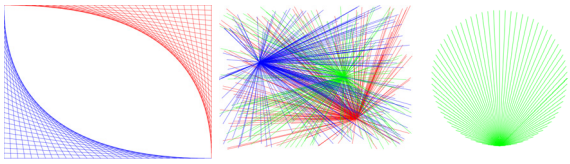
請輸入三個數字：17.1 17.2 100.5

您輸入的數字中，最大為：100.5, 最小為：17.1

49

7-2.cpp: 電腦繪圖藝術

- 請撰寫一程式，首先讓使用者輸入所要產出的創作類型：1) 規則藝術、2) 不規則藝術、3) sincos 藝術，接著分別輸入所需要的資料 (後述)，爾後透過SVG繪圖產出所選定的藝術圖形。

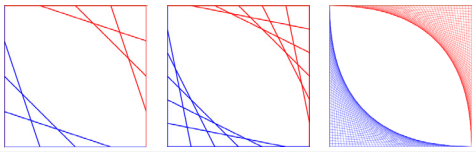


50

1) 規則藝術

- 所需輸入的資料為「每邊的線條數, n」
- n 包含了水平與垂直兩條線
- 自己決定整個圖的寬度與高度，並利用迴圈與 n 計算出每條線的起點與終點

n=5 n=7 n=50

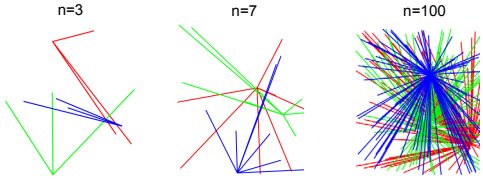


51

2) 不規則藝術

- 所需輸入的資料為「每種顏色的線條數目, n」
- 每種顏色 (紅、綠、藍) 使用亂數產生一固定的起點，再用亂數隨機產生 n 個在繪圖範圍內的終點，並將起點與終點連線起來 (看起來像是彩球或是煙火)。

n=3 n=7 n=100

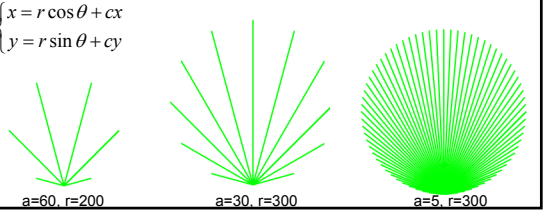


52

3) sincos藝術

- 所需輸入的資料為「角度間隔量 , a」與「半徑, r」。
- 所繪製的線條為綠色，所有線條的起點為圓的最下方，終點則由圓的最右方開始繞圓一圈，每次的圓心角差異量即為角度間隔量。
- 圓周上各點的座標可由直角座標 – 極座標轉換公式求得

$$\begin{cases} x = r\cos\theta + cx \\ y = r\sin\theta + cy \end{cases}$$



a=60, r=200 a=30, r=300 a=5, r=300

53

隨堂練習

質數判斷

a) 請撰寫一函式，其原型宣告如下:
   bool isPrime(unsigned long p);
   此函式傳入一無號長整數p，若 p 為質數時回傳 true，否則回傳 false。注意此函式內不會列印任何東西!

b) 請撰寫一主程式，讓使用者輸入一正整數 n，接著利用 a) 所寫的 isPrime 函式，輸出介於 1 – n 之間的所有質數，且每一列最多輸出 10 個數字，最後並輸出總有找到多少個質數。

Hint: 可參考範例程式6-1.cpp

本程式會找出介於 1 ~ n 之間的所有質數。
請輸入 n: 1000
2, 3, 5, 7, 11, 13, 17, 19, 23, 29,
31, 37, 41, 43, 47, 53, 59, 61, 67, 71,
73, 79, 83, 89, 97, 101, 103, 107, 109, 113,
127, 131, 137, 139, 149, 151, 157, 163, 167, 173,
179, 181, 191, 193, 197, 199, 211, 223, 227, 229,
233, 239, 241, 251, 257, 263, 269, 271, 277, 281,
283, 293, 307, 311, 313, 317, 331, 337, 347, 349,
353, 359, 367, 373, 379, 383, 389, 397, 401, 409,
419, 421, 431, 433, 439, 443, 449, 457, 461, 463,
467, 479, 487, 491, 499, 503, 509, 521, 523, 541,
547, 557, 563, 569, 571, 577, 587, 593, 599, 601,
607, 613, 617, 619, 631, 641, 643, 647, 653, 659,
661, 673, 677, 683, 691, 701, 709, 719, 727, 733,
739, 743, 751, 757, 761, 769, 773, 787, 797, 809,
811, 821, 823, 827, 829, 839, 853, 857, 859, 863,
877, 881, 883, 887, 907, 911, 919, 929, 937, 941,
947, 953, 967, 971, 977, 983, 991, 997,
總共有: 168 個質數。