1

回顧 - 參數預設值或內定值

- 函式的參數可以帶預設值或是內定值

```
void nStar(char symbol='*', int x=10) {
  for(int i=0;i<x;i++) cout << symbol;
  cout << endl;
}

nStar('x', 20);
nStar('+');
nStar();
```

2

回顧 - 遞迴函式

- 函式可以呼叫自己 → 遞迴

```
int arithmeticSequence (int i) {
    if(i==1) return 5;
    return arithmeticSequence(i-1)+3;
}

int geometricSequence (int i) {
    if(i==1) return 5;
    return 1.3 * geometricSequence(i-1);
}
```

$a_i = a_{i-1} + d$
$a_1 = 5, d = 3$

$a_1 = 5, r = 1.3$
$a_i = r \cdot a_{i-1}$

3

回顧 - 函式覆載 (overloading)

- 函式的名稱可以重覆，只要函式名稱＋參數的組合唯一即可。

```
double area(double r) {
       return r * r * 3.1415926535897932385;
}
double area(double width, double height) {
       return width * height;
}

void area(double r);
int area(int r);
float area(float r=2.0);
double area(double a, double b=2.0);
double area(double a, double b, double c, double d);
double area(double a, double b=1.0, double c);
```

4

回顧 - 變數的視野/範疇

- 函式的名稱可以重覆，只要函式名稱＋參數的組合唯一即可。
- 全域變數可以宣告在所有的函式之外，可以被其以下的函式所使用
- 全域變數可用來在不同的函式之間交換資料，不過儘量不要這樣作，因為會造成程式除錯的困難。

```
int a;
void fun() {
       cout << a++;
}
int main() {
       a=3;
       fun();
       ++a;
       fun();
       cout << a;
       return 0;
}
```

3
5
6

5

Lecture 10

函式呼叫時的資料傳遞 (I) – 傳值
參考型別
函式呼叫時的資料傳遞 (II) – 傳參考
基本檔案操作

6

函式呼叫時的資料傳遞 (I)

傳值呼叫 (call by value)

7

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)。

一般參數列可以將資料傳入進行運算，也可用來將資料傳出！
(這個範例僅將資料傳入，而沒有資料能從參數列傳出！)

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);

8

10-1.cpp

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

int b = a;

- 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 b ，然後才開始執行 test 函式的內容。
- 所傳入的參數 (e.g. a)，是用來**初始化**在函式裡的區域變數 (b)。
- test 函式結束時將 b+1 的值透過 return 的功能傳回主程式中，並在此範例中存入主程式中的 b 變數，然後test函式中的 b 變數即被消滅。
- 因為是**傳值呼叫**，無論test函式裡的 b 變數如何改變，都不會影響主程式裡 a 變數的值。

a=3
b=7

9

10-1.cpp 說明

以下的 main::、test:: 用來強調該變數所在的區域或是說該變數所在的範疇 (scope)，並不是正確的語法 。

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

1: int **main::**a = 3;
2: int **main::**b;
3:
  _: int **test::**b = **main::**a; (test::b = 3)
  a: **test::**b = **test::**b * 2; (test::b = 6)
  b: return **test::**b + 1; (return 7)
  main::b = 回傳值; (main::b = 7)
4: cout << "a=" << **main::**a << endl;(印出 main::a)
5: cout << "b=" << **main::**b << endl;(印出 main::b)

10

10-2.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;
}
```

11

10-2.cpp (cont'd)

```
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。

12

10-2.cpp (cont'd)

以上程式中，真正重要且有執行的敍述與順序有：

```
main:: int main::a = 3;
main:: int main::b = 7;
main:: cout << "a=" << main::a << ", b=" << main::b << endl;
main:: swap(a, b);
  swap:: int swap::a = main::a;
  swap:: int swap::b = main::b;
  swap:: cout << swap::a << ", " << swap::b << endl;
  swap:: int swap::tmp = swap::a;
  swap:: swap::a = swap::b;
  swap:: swap::b = swap::tmp;
  swap:: cout << swap::a << ", " << swap::b << endl;
main:: cout << "a=" << main::a << ", b=" << main::b << endl;
```

13

參考型別

14

10-3.cpp

```
#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;
}
```

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

```
3
3

4
```

15

10-3.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!
  (有點類似在宣告常數變數(const)時，一定要指定常數值)
- 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。

16

10-4.cpp

```
#include <iostream>
using namespace std;
int main() {
  int a = 3;
  int &b = a;
  int &c = b;
  int d = c;

  cout << a << ", " << b << ", " << c << ", " << d << endl;
  d = 10;
  cout << a << ", " << b << ", " << c << ", " << d << endl;
  c = 5;
  cout << a << ", " << b << ", " << c << ", " << d << endl;

  return 0;
}
```

```
3, 3, 3, 3
3, 3, 3, 10
5, 5, 5, 10
```

17

函式呼叫時的資料傳遞 (II)

傳參考呼叫 (call by reference)

18

10-5.cpp

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

int &b = a;

- 此範例為**傳參考**的函式呼叫。
- 在 test 函式被呼叫時, 主程式傳入的 a 變數成為 b 這個參考型別的變數初始值 ，因此 b 變數在 test 函式中，為主程式中的 a 變數的分身。
- test 函式結束時將 b+1 的值透過 return 的功能傳回主程式中，並在此範例中存入主程式中的 b 變數。
- 因為是傳參考呼叫，所以在 test 函式中對於 b 變數的改變，實際上就是對主程式中的 a 變數作改變。

```
a=6
b=7
```

19

10-5.cpp 說明

以下的 main::、test:: 用來強調該變數所在的區域或是說該變數所在的範疇 (scope)、:: 前面則提示所對應的程式敘述所在位置，並不是正確的語法。

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

```
_   int &b = a;
```

```
1: int main::a = 3;
2: int main::b;
3:
    _: int test::&b = main::a;      (test::b = 3)
    a: test::b = test::b * 2;       (test::b = 6)
    b: return test::b + 1;          (return 7)
    main::b = retval;               (main::b = 7)
4: cout << "a=" << main::a << endl;(印出 main::a)
5: cout << "b=" << main::b << endl;(印出 main::b)
```

20

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
- ***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*

21

10-6.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;
}
```

22

10-6.cpp (cont'd)

```
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。

23

10-6.cpp (cont'd)

以上程式中，真正重要且有執行的敘述與順序有：

```
main:: int main::a = 3;
main:: int main::b = 7;
main:: cout << "a=" << main::a << ", b=" << main::b << endl;
main:: swap(a, b);
  swap:: int swap::&a = main::a;
  swap:: int swap::&b = main::b;
  swap:: cout << swap::a << ", " << swap::b << endl;
  swap:: int swap::tmp = swap::a;
  swap:: swap::a = swap::b;
  swap:: swap::b = swap::tmp;
  swap:: cout << swap::a << ", " << swap::b << endl;

main:: cout << "a=" << main::a << ", b=" << main::b << endl;
```

24

10-7.cpp

A. 看起來，area 不管值是多少，函式呼叫完成後都會是圓面積。
B. 看起來，不管 perimeter 是多少，函式呼叫完成後都會是圓周長。
C. 所以，它們都應該使用傳參考呼叫。
D. 看起來，circle函式沒有回傳值。

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

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

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

  return 0;
}
```

25

先想原型宣告

A. 看起來，area 不管值是多少，函式呼叫完成後都會是圓面積。
B. 看起來，不管 perimeter 是多少，函式呼叫完成後都會是圓周長。
C. 所以，它們都應該使用傳參考呼叫。
D. 看起來，circle函式沒有回傳值。

```
double r, area, perimeter;
circle(r, area, perimeter);   // 如何寫這個函式？
```

回傳值型別
函式名稱(參數1宣告, 參數2宣告, …)

宣告 = 變數型別 變數名稱
e.g. int a;

```
void
circle(
      double r,
      double &area,
      double &perimeter
);
```

26

10-7.cpp (cont'd)

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

27

小結

- 函式呼叫僅使用傳值法 (C/C++ 的內定方法) 時，我們只能傳回一個運算結果 (透過 return)。
- 傳值呼叫**不可能**意外改到呼叫者 (e.g. main) 內的變數值。
- 傳參考時，乃將呼叫者的變數 (e.g. a)，在被呼叫函式中創造出一個別名，所以在函式裡改變變數值，則呼叫者的變數也會被改變。
- 傳參考呼叫方法則可能可以傳回一個以上的運算結果，並可以改變呼叫者內的變數值。
- 傳值呼叫，該參數為函式的輸入；傳參考呼叫，該參數可能為輸入值也可能為輸出值。10-6.cpp 中的 swap 函式 a b 兩變數是傳入值、也是傳出值。10-7.cpp中的circle，r 為傳入值、area、perimeter為傳出值。

28

基本檔案操作

基本的檔案讀取與寫入

29

檔案輸入與輸出
file I/O (input/output)

- 為什麼需要電腦檔案？
  - 所有我們目前學到的，都是在利用記憶體 (變數) 來進行運算，而運算的資料來源為：
    - 1) 內建於程式內的變數 (data = 3;)
    - 2) 由使用者輸入 (cin >> data);
  - 運算得到的結果常常是輸出至螢幕上
    - cout << result;
- 問題
  - 運算結果無法長期保存下來、無法檢查
  - 使用者會忘記要輸入什麼資料
  - 重複輸入資料很煩、會出錯。
- 在 C/C++ 來說，電腦檔案是一種串流 (stream) 或資料流。

30

檔案串流
file stream

- 檔案串流，即為其來源或目的為電腦檔案
  - cin: 來源為鍵盤輸入的串流物件
  - cout: 來源為螢幕輸出的串流物件
  - 檔案串流需有相關的檔案名稱！
- 檔案輸入輸出串流
  - **檔案輸入串流：ifstream** (input file stream)
  - **檔案輸出串流：ofstream** (output file stream)
  - **檔案輸入輸出串流：fstream**(file stream) – 可設定作輸入、輸出的串流

31

10-8.cpp – 檔案輸出

```
#include <iostream>
#include <fstream>
using namespace std;
int main() {
  ofstream file;
  file.open("d:\\test.txt");
  if(!file) {
      cout << "檔案開啟錯誤!";
      return 255;
  }
  file << "Hello File!";
  file.close();
  return 0;
}
```

含括入檔案串流的相關原型宣告

宣告 file 為一**輸出**檔案串流物件
開啟 d:\test.txt
檢查檔案串流開啟狀態，一個好的程式應該要檢查檔案是否有開啟成功，以免後面的程式出錯。

將 "Hello File!" 插入至檔案串流
關閉檔案串流

32

檔案串流物件

- cin, cout, ofstream, ifstream 都是一種串流物件
- 之前 cin, cout 的操作 <<, >> 都可直接套用在檔案操作上。
  - `file << "Hello File!";`
  - `file << "\nA=" << A << ", " << "B=" << B;`
  - << 和>>稱為串流的插入與抽取運算子
- 檔案串流只增加了
  - #include <**fstream**>
  - ifstream abc; 或是 ofstream abc; ← 用來產生輸入/輸出串流物件
  - abc.open("c:\\test.dat"); ← 將串流物件關連於一個電腦檔案
  - abc.close(); ← 關閉檔案
  - 其中 abc 為任意合法的變數名稱

34

串流的狀態
stream state

- 串流有以下的狀態，通常是經過輸入或輸出的操作後會被設定。
  - **eof** (end of file): 串流已至檔案結尾，表所有資料都已經流過了，不會再有更多資料進來了。
  - **fail**: 失敗，表示有輸入或操作的動作失敗。例如：
    - cin >> a; (其中 a 為整數，但輸入的資料為 "$AC"，無法轉換)
  - **bad**: 壞掉，表串流本身之完整性 (integrity) 有問題，通常代表即使將狀態清除後換一種操作仍然會發生問題。
- 可透以下相關函式檢查或設定
  - **bad()**: 回傳 true 時表示 bad 的狀態發生了。
  - **fail()**: 回傳 true 時表示 bad 或 fail 的狀態發生了。
  - **eof()**: 回傳 true 時表示已到達檔案結尾。
  - **clear()**: 將以上三種不好的狀態清除。
  - **good()**: 回傳 true 表示以上三種狀態都沒有發生, 可以進行操作。
- 前範例中僅使用了 eof() 檢查是否到達檔案結尾。

35

10-10.cpp

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

int main() {
  int i, ans, a;
  unsigned long sum;
  char comma;
  ifstream inp;

  inp.open("d:\\test.dat");
  if(!inp) {
      cout << "檔案開啟錯誤";
      return 255;
  }
  cout << "請問您要加總到的數目:";
  cin >> ans;
  for(i=1;i<=ans;i++) {
      inp >> a >> comma >> sum;
  }
  cout <<"總和為: " << sum << endl;
  inp.close();
  return 0;
}
```

10-9.cpp – 將運算結果輸出至檔案

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

int main() {
  int i;
  unsigned long sum=0;
  ofstream outp;

  outp.open("d:\\test.dat");
  if(!outp) {
      cout << "檔案開啟錯誤";
      return 255;
  }
  for(i=1;i<=20000;i++) {
    sum += i;
    outp<< i <<", "<<sum<<"\n";
  }
  outp.close();

  return 0;
}
```

1, 1
2, 3
3, 6
4, 10
5, 15
.
.

HW09

Due: 6/5/2015

1.一元二次方程式的解

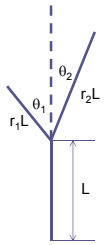
請撰寫一程式，由使用者提供一元二次方程式 $ax^2 + bx + c = 0$ 中的係數a、b、與c，並計算該方程式的解。其中，主程式已完成如下頁，請根據主程式去撰寫其中的函式 solve，使其能達到求解的功能。參考輸入輸出如下下頁。

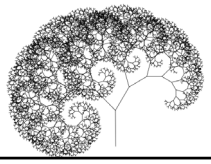
$$ax^2 + bx + c = 0$$

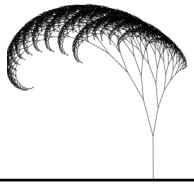
$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

2. 碎形幾何 (fractal)

- 碎形幾何為由相同的「樣式」所組成的圖形。如下圖形為例，皆由簡單的「Y」重覆繪製而成。其中，主幹的長度定義為 L，每一個主幹會長出左、右兩分支，如右圖所示。其中左分支長度為r_1L、偏轉角度為θ_1；右分支長度為r_2L、偏轉角度為θ_2。
- 透過變化r_1 、θ_1、r_2、θ_2等四個參數，我們可以得到許多非常不同的圖形。







```
int main() {
  double a, b, c;
  cout << "請輸入一元二次方程式 ax^2 + bx + c = 0 中的 a, b, 與 c: ";
  cin >> a >> b >> c;
  int result;
  double x1, x2;

  result = solve(a, b, c, x1, x2);

  cout << a << "x^2 + " << b << "x + " << c << "=0 這個方程式";
  switch (result) {
    case 0:
      cout << " 沒有實數解。";
      break;
    case 1:
      cout << " 有一實根為: " << x1;
      break;
    case 2:
      cout << " 有兩實根為: " << x1 << ", " << x2;
      break;
    default:
      cout << " \n程式發生異常 ...";
      break;
  }
  return 0;
}
```


2. 碎形幾何 (fractal)

- 以下將定義本作業所需撰寫的程式：
- 本程式會有主程式 main 以及一個函式 tree 。
- 請宣告五個全域變數，一個為 SVG 繪圖使用、另外四個則為前面介紹中的四個參數r_1 、θ_1、r_2、θ_2 。第一個變數主要由 tree 函式使用以繪圖；後面四個變數的值將在主程式中設定，並在之後的 tree 函式中被使用。
- 主程式中，讓使用者輸入一檔案名稱，爾後開啟使用者指定的檔案，並讀取檔案中的四個參數 r_1 、θ_1、r_2 、θ_2 後並將其值儲存於全域變數中。接著將檔案關閉，並將使用者指定的參數列印出來。接著呼叫 tree 函式 (以下會介紹）根據使用者指定的四個參數以繪製碎形幾何之圖形。
- 繳交作業時，除了作業的程式碼外，請額外再多附三個參數檔，其中的參數是你認為你很喜歡的結果。

```
請輸入一元二次方程式 ax^2 + bx + c = 0 中的 a, b, 與 c: 1 3 2
1x^2 + 3x + 2=0 這個方程式 有兩實根為: -1, -2
```

```
請輸入一元二次方程式 ax^2 + bx + c = 0 中的 a, b, 與 c: 1 0 -2
1x^2 + 0x + -2=0 這個方程式 有兩實根為: 1.41421, -1.41421
```

```
請輸入一元二次方程式 ax^2 + bx + c = 0 中的 a, b, 與 c: 1 0 2
1x^2 + 0x + 2=0 這個方程式 沒有實數解。
```

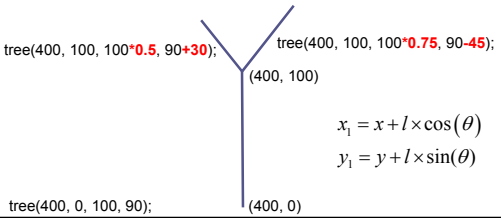

2. 碎形幾何 (fractal)

- 此程式中需要遞迴呼叫 tree 函式以繪製圖形。以下為 tree 這個函式的原型宣告：
  - void tree(double x, double y, double L, double theta);
  - x, y 為「Y」中主幹的根部座標、L為主幹長度、theta為主幹方向。
- 在 tree 函式的內部, x, y 為主幹的起點座標
  - 首先計算主幹的終點座標 (x_1, y_1)
  - 繪製直接 (x, y) – (x_1, y_1)
  - 如果主幹長度 < 5 ，則 return; (不再繪製分支)
  - 呼叫 tree 函式，繪製左分支
  - 呼叫 tree 函式，繪製右分支
- 注意SVG繪圖時的座標系統與我們習慣的座標系統不同。

43

2. 碎形幾何 (fractal)

參數：
r_1=0.5, θ_1=30
r_2=0.75, θ_2=45



$$x_1 = x + l \times \cos(\theta)$$
$$y_1 = y + l \times \sin(\theta)$$

46

參考執行結果

請輸入端點荷重大小 (lb.)：100
請輸入可容許變位量 (in.)：1
請輸入樑的長度 (in.)：1000

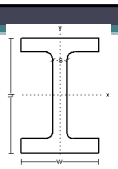
S24x121 變位量為 0.363648 in.，總重量為 10083.3 lb.
S24x106 變位量為 0.390859 in.，總重量為 8833.33 lb.
S24x100 變位量為 0.480806 in.，總重量為 8333.33 lb.
S24x90 變位量為 0.510723 in.，總重量為 7500 lb.
S24x80 變位量為 0.547203 in.，總重量為 6666.67 lb.
S20x96 變位量為 0.6881 in.，總重量為 8000 lb.
S20x86 變位量為 0.727295 in.，總重量為 7166.67 lb.
S20x75 變位量為 0.897755 in.，總重量為 6250 lb.
S20x66 變位量為 0.965652 in.，總重量為 5500 lb.

總共試了 31 種梁，其中 9 種可以使用。

44

隨堂練習

47



Properties in imperial units of American Standard Beams according ASTM A6 are indicated below.

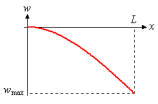
Designation	Dimensions					Static Parameters			
						Moment of Inertia		Elastic Section Modulus	
Imperial (in x lb/ft)	Depth h (in)	Width w (in)	Web Thickness s (in)	Sectional Area (in^2)	Weight (lb/ft)	I_x (in^4)	I_y (in^4)	W_x (in^3)	W_y (in^3)
S 24 x 121	24.5	8.050	0.800	35.6	121	3160	83.3	258	20.7
S 24 x 106	24.5	7.780	0.620	31.2	106	2940	77.1	240	19.6
S 24 x 100	24	7.425	0.745	29.3	100	2390	47.7	199	13.2
S 24 x 90	24	7.125	0.625	26.5	90	2250	44.9	187	12.6
S 24 x 80	24	7.000	0.500	23.5	80	2100	42.2	175	12.1
S 20 x 96	20.3	7.200	0.800	28.2	96	1670	50.2	165	13.9
S 20 x 86	20.3	7.060	0.660	25.3	86	1580	46.8	155	13.3
S 20 x 75	20	6.385	0.635	22.0	75	1280	29.8	128	9.32
S 20 x 66	20	6.255	0.505	19.4	66	1190	27.7	119	8.85

http://www.engineeringtoolbox.com/american-standard-beams-d_1320.html

45

選擇 I 型樑

請撰寫一程式，讓使用者輸入一懸臂梁的自由端端點垂直荷重、可接受的最大變位量、以及懸臂梁的長度。之後列出檔案中所定義各種 I 型梁斷面，那一些可以滿足所指定的條件，以及其重量。請假設楊氏模數 E 為 $200x10^9$ (GPa)。請注意所給梁的檔案使用英制單位！



$$w(x) = -\frac{P x^2 (3L - x)}{6EI} \qquad w_{max} = w(L) = -\frac{PL^3}{3EI}$$

http://www.efunda.com/formulae/solid_mechanics/beams/casestudy_display.cfm?case=cantilever_endload