程式設計

第4章 函式
Function

蘇維宗(Wei-Tsung Su)
suwt@au.edu.tw
564D

目標

函式的宣告與定義

函式呼叫與記憶體

函式的使用時機

遞迴函式(Recursion)

函式

簡單來說，函式是一組執行特定功能的程式碼集合。

例如，整數加法(addi)的輸入是加數(x)與被加數(y)，輸出為結果(x+y)。

函式宣告(Declaration)

函式宣告包含(函式名稱、輸入參數型別、輸出結果型別)

```
1.  // 功能：整數加法(addi)
2.  // 輸入：兩個整數(int, int)
3.  // 輸出：整數(int)
4.  int addi(int, int);
```

輸出結果型別　　函式名稱　　輸入結果型別

函式定義

函式定義包含(函式名稱、輸入參數型別與**名稱**、輸出結果型別、**程式實體**)

```
1.  int addi(int x, int y) {
2.    int z = x + y;
3.    return z; //回傳值須為整數型態
4.  }
```

函式呼叫

呼叫函式時需給定輸入並取得輸出結果

```
1.  //呼叫者(caller)
2.  int main() {
3.    int num1 = 2, num2 = 5;
4.    int num3 = addi(num1, num2);
5.    printf("%d\n", num3);
6.    return EXIT_SUCCESS;
7.  }
```

```
1.  //被呼叫者(callee)
2.  int powi(int x, int y) {
3.    int z = x + y;
4.    return z; //回傳值須為整數型態
5.  }
```

重要:

1. 呼叫函式之前，必須先宣告 (或定義)該函式
2. 在C語言中，不能宣告(定義)名稱相同的函式，否則會出現重複宣告 (定義)

函式呼叫(續)

呼叫函式之前，必須先宣告(或定義)該函式

```
1.   //在呼叫函式前宣告(正確)
2.   int addi(int, int);
3.
4.   int main () {
5.     ...
6.     int z = addi(x,y);
7.     ...
8.   }
9.
10.  int addi(int x, int y) {
11.    ...
12.  }
```

```
1.   //在呼叫函式前定義(正確)
2.   int addi(int x, int y) {
3.     ...
4.   }
5.
6.   int main () {
7.     ...
8.     int z = addi(x,y);
9.     ...
10.  }
```

```
1.   //沒有在呼叫函式前宣告或定義(錯誤)
2.   int main () {
3.     ...
4.     int z = addi(x,y);
5.     ...
6.   }
7.
8.   int addi(int x, int y) {
9.     ...
10.  }
```

函式呼叫(續)

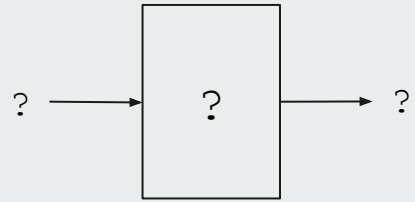
在C語言中，不能宣告(定義)名稱相同的函式，否則會出現重複宣告(定義)

```
1.   //重複宣告(錯誤)
2.   int addi(int, int);
3.   int addi(float, float);
4.   int main () {
5.     ...
6.     int z = addi(x,y);
7.     ...
8.   }
9.
10.  int addi(int x, int y) {
11.    ...
12.  }
```

```
1.   //重複宣告(錯誤)
2.   int addi(int, int);
3.   float addi(int, int);
4.   int main () {
5.     ...
6.     int z = addi(x,y);
7.     ...
8.   }
9.
10.  int addi(int x, int y) {
11.    ...
12.  }
```

```
1.   //重複宣告(錯誤)
2.   int addi(int, int);
3.   float addi(float, float);
4.   int main () {
5.     ...
6.     int z = addi(x,y);
7.     ...
8.   }
9.
10.  int addi(int x, int y) {
11.    ...
12.  }
```

實作計算三角形面積的函式



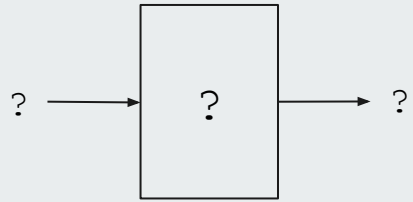
三角形面積公式

$$area = (base * height)/2$$

輸入	輸出
10 5	25.00
5 5	12.50
3 4.5	6.75

輸出無條件捨去到小數點後2位

實作一個可計算二維歐幾里德距離的函式



二維歐幾里得距離公式

$$d = \sqrt{(x1 - x2)^2 + (y1 - y2)^2}$$

輸入	輸出
0 0 0 2	2.00
0 2 0 2	0.00
1 2 3 4	2.82

輸出無條件捨去到小數點後2位

函式呼叫與記憶體

交換兩個變數數值

請問下面的程式會印出什麼?

```
1.  int main() {
2.    int x = 5;
3.    int y = 2;
4.    printf("%d %d\n", x, y);
5.    x = y;
6.    y = x;
7.    printf("%d %d\n", x, y);
8.    return EXIT_SUCCESS;
9.  }
```

Line 4	
x	5
y	2

Line 5	
x	2
y	2

Line 6	
x	2
y	2

交換兩個變數數值(續)

```
 1.  int main() {
 2.    int x = 5;
 3.    int y = 2;
 4.    printf("%d %d\n", x, y);
 5.    int t = x;
 6.    x = y;
 7.    y = t;
 8.    printf("%d %d\n", x, y);
 9.    return EXIT_SUCCESS;
10.  }
```

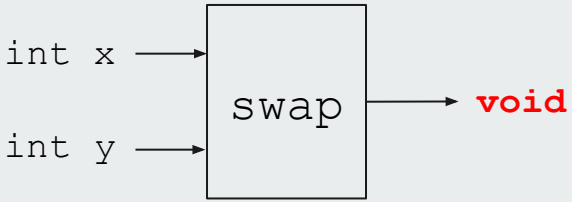
Line 4	
x	5
y	2

Line 5	
x	5
y	2
t	5

Line 6	
x	2
y	2
t	5

Line 7	
x	2
y	5
t	5

實作交換兩個變數數值的函數



例如，輸入x=2， y=5，經過函式運算後使得x=5， y=2。

註:void代表沒有回傳值

輸入	輸出
5 10	10 5
1 9	9 1
5 6	6 5

函式呼叫的記憶體使用

呼叫函式時需給定輸入並取得輸出結果

```
1.  //呼叫者(caller)
2.  int main() {
3.    int x = 5, y = 2;
4.    printf("%d %d\n", x, y);
5.    swap(x, y);
6.    printf("%d %d\n", x, y);
7.    return EXIT_SUCCESS;
8.  }
```

```
1.  //被呼叫者(callee)
2.  void swap(int x, int y) {
3.    int t = x;
4.    x = y;
5.    y = x;
6.  }
```

重要:

1. 呼叫函式時會在記憶體堆疊(stack)中新增該函式所使用到的區域變數與函式返回位址
2. 如果以傳值方式呼叫函式(call by value)，則caller只會把變數的數值傳遞給callee，因此在callee中對該變數的改變並不影響caller中對應的變數。

函式呼叫的記憶體使用

呼叫函式時需給定輸入並取得輸出結果

```
1.  //呼叫者(caller)
2.  int main() {
3.    int num1 = 2, num2 = 5;
4.    int num3 = addi(num1, num2);
5.    printf("%d\n", num3);
6.    return EXIT_SUCCESS;
7.  }
```

```
1.  //被呼叫者(callee)
2.  int powi(int x, int y) {
3.    int z = x + y;
4.    return z; //回傳值須為整數型態
5.  }
```

重要:

1. 呼叫函式時會在記憶體堆疊(stack)中新增該函式所使用到的區域變數與函式返回位址
2. 如果以傳值方式呼叫函式(call by value), 則caller只會把變數的數值傳遞給callee, 因此在callee中對該變數的改變並不影響caller中對應的變數。

函式的使用時機

函式可以將一個大型的程式切割成多個小程式，主要使用的時機包含

1. 讓主程式變的簡潔且容易閱讀(readable)
2. 重複使用相同的程式碼(reusable)
3. 容易多人合作開發專案(collaboration)

如何運用函式來提高程式的可讀性與程式碼的再使用率?

猜猜這個程式在做什麼?

bina.c

```
 1.   #include<math.h>
 2.
 3.   int main() {
 4.     char bin1[10];
 5.     char bin2[10];
 6.     scanf("%s", bin1);
 7.     scanf("%s", bin2);
 8.
 9.     int num1=0, num2=0;
10.     for(int i=0;i<=7;i++) {
11.       num1 += (bin1[i]-'0') * pow(2, 7-i);
12.     }
13.   for(int i=0;i<=7;i++) {
14.       num2 += (bin2[i]-'0') * pow(2, 7-i);
15.     }
16.
17.     int result = num1 + num2;
18.     for(int i=7;i>=0;i--) {
19.       printf("%d", (result >> i) & 1);
20.     }
21.     printf("\n");
22.
23.     return EXIT_SUCCESS;
24.   }
```

```
> gcc -Wall bina.c -lm
```

二進制相加

輸入兩個8位元的2進位無號整數，輸出兩個8位元2進位無號整數相加的結果。

輸入	輸出
00000001 00000010	00000011
00100010 10100101	11000111
11010001 01000000	00010001

先把大問題切成幾個小問題

二進制相加：輸入兩個8位元的2進位無號整數，輸出相加的結果。

Step 1. 以字串的方式輸入兩個8位元的2進位無號整數

Step 2. 將兩個字串轉換成無號整數

Step 3. 將兩個無號整數相加得到結果

Step 4. 將結果以8位元的2進位無號整數方式輸出

猜猜這個程式在做什麼? (主程式功能不易閱讀)

bina.c

Step 1

```
1.   #include<math.h>
2.
3.   int main() {
4.     char bin1[8];
5.     char bin2[8];
6.     scanf("%s", bin1);
7.     scanf("%s", bin2);
8.
```

Step 2

```
9.     int num1=0, num2=0;
10.    for(int i=0;i<=7;i++) {
11.      num1 += (bin1[i]-'0') * pow(2, 7-i);
12.    }
```

Step 2 使用了重複的程式碼

Step 2

```
13.    for(int i=0;i<=7;i++) {
14.      num2 += (bin2[i]-'0') * pow(2, 7-i);
15.    }
```

Step 3

```
16.
17.    int result = num1 + num2;
```

Step 4

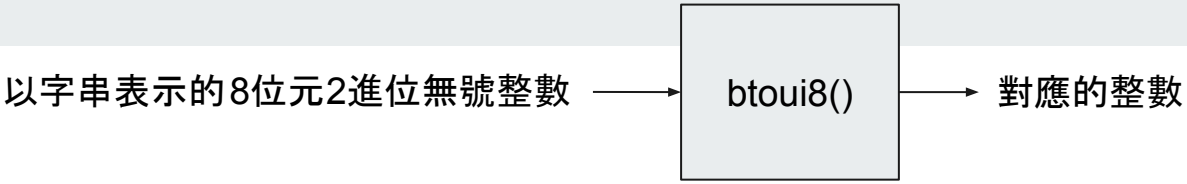
```
18.    for(int i=7;i>=0;i--) {
19.      printf("%d", (result >> i) & 1);
20.    }
21.    printf("\n");
22.
23.    return EXIT_SUCCESS;
24.  }
```

```
> gcc -Wall bina.c -lm
```

透過撰寫函式來改善這些問題

撰寫函式要注意的事

1. 函數的名稱要有意義。例如，格式化輸出與輸入：printf()、scanf()。
2. 函式必須在使用之前被**定義**或**宣告**。
3. 不能有相同名稱的函式(即使函式的輸入/輸出不同)。// C語言
4. 因為編譯器可能會自動轉型，所以要注意輸入/輸出的型態是否正確。
5. ...



函式宣告(函式原型)

函式宣告只需要包含(名稱、輸入參數型態、輸出結果型態)

```
1.  // 將以字串表示的8位元2進位無號整數轉換成整數
2.  // 輸入：8位元組的字串 ; Output: 8 bits unsigned integer
3.  int btoui8(char*);
```

輸出結果型態 介面　　**名稱**　　**輸入參數型態** 介面

函式定義

函式定義需要包含(名稱、輸入參數型態與**名稱**、輸出結果型態、**程式實體**)

```
1.  int btoui8(char* bin) {
2.    int num = 0;
3.    for(int i=0;i<=7;i++) {
4.      num += (bin[i]-'0') * pow(2, 7-i);
5.    }
6.    return num; //回傳值須為整數型態
7.  }
```

猜猜這個程式在做什麼?

binaf1.c

```
 1.  #include<math.h>
 2.  int btoui8(char*);
 3.  int main() {        // main() is caller
 4.    char bin1[8];
 5.    char bin2[8];
 6.    scanf("%s", bin1);
 7.    scanf("%s", bin2);
 8.    // btoui8() is callee
 9.    int result = btoui8(bin1) + btoui8(bin2);
10.    for(int i=7;i>=0;i--) {
11.      printf("%d", (result >> i) & 1);
12.    }
13.    printf("\n");
14.    return EXIT_SUCCESS;
15.  }
16.
17.  int btoui8(char* bin) {
18.    int num = 0;
19.    for(int i=0;i<=7;i++) {
20.      num += (bin[i]-'0') * pow(2, 7-i);
21.    }
22.    return num;
23.  }
```

```
> gcc -Wall binaf1.c -lm
```

猜猜這個程式在做什麼? (續)

binaf1.c

```
1.   #include<math.h>
2.   int btoui8(char*);
3.   int main() {
4.     char bin1[8];
5.     char bin2[8];
6.     scanf("%s", bin1);
7.     scanf("%s", bin2);
8.
9.     int result = btoui8(bin1) + btoui8(bin2);
10.    for(int i=7;i>=0;i--) {
11.      printf("%d", (result >> i) & 1);
12.    }
13.    printf("\n");
14.    return EXIT_SUCCESS;
15.  }
16.
17.  int btoui8(char* bin) {
18.    int num = 0;
19.    for(int i=0;i<=7;i++) {
20.      num += (bin[i]-'0') * pow(2, 7-i);
21.    }
22.    return num;
23.  }
```

```
> gcc -Wall binaf1.c -lm
```

請將Step 4的程式碼改寫成函式呼叫

如何多人同時開發程式專案?

主程式檔案與輔助函式檔案

功能函式介面定義清楚後，就可以讓A開發函式，B開發主程式，最後再進行整合。例如，

- A負責開發以下兩個函式(標頭檔`binui.h`、原始檔`binui.c`)
  - `int btoui8(char*)` //將字串轉成8位元無號整數
  - `void bprint8(int)` //將整數以8位元無號整數印出
- B負責開發主程式(`main.c`)
  - 輸入兩個8位元的2進位無號整數，輸出相加的結果。

函式開發

標頭檔(binui.h)宣告功能函式

```
1.   int btoui8(char *);
2.   void bprint8(int);
```

原始檔(binui.c)實作函式

```
1.   #include<stdlib.h>
2.   #include<stdio.h>
3.   #include<math.h>
4.
5.   int btoui8(char* bin) {
6.     int num=0;
7.     for(int i=0;i<=7;i++) {
8.       num += (bin[i]-'0') * pow(2, 7-i);
9.     }
10.    return num;
11.  }
12.
13.  void bprint8(int num) {
14.    for(int i=7;i>=0;i--) {
15.      printf("%d", (num >> i) & 1);
16.    }
17.    printf("\n");
18.  }
```

主程式開發

主程式開發者只需要專注在功能的開發(假設函式已完成)。

因為會呼叫自行開發的函式，所以在主程式中會以

```
#include"binui.h"
```

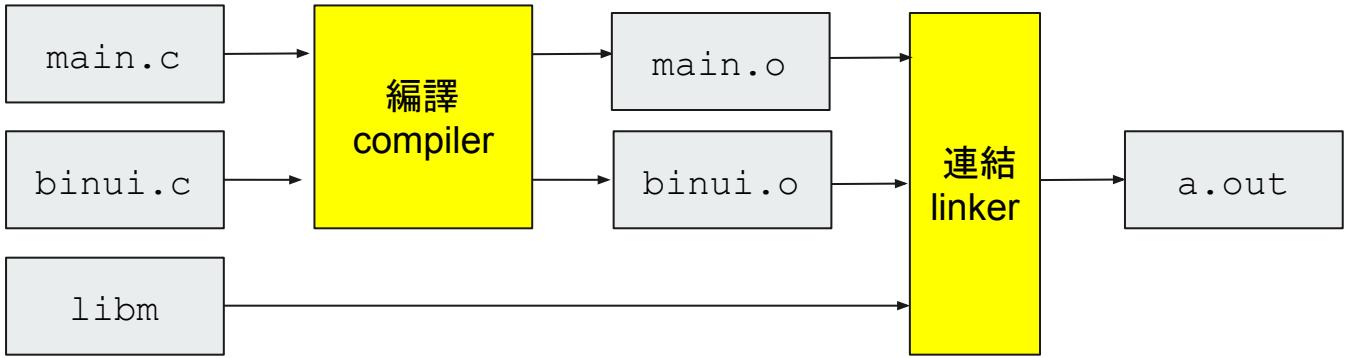
來引用。

```
1.  #include<stdlib.h>
2.  #include<stdio.h>
3.  #include"binui.h"
4.
5.  int main() {
6.    char bin1[10];
7.    char bin2[10];
8.    scanf("%s", bin1);
9.    scanf("%s", bin2);
10.   int result = btoui8(bin1) + btoui8(bin2);
11.   bprint8(result);
12.   return EXIT_SUCCESS;
13. }
```

如何編譯多個原始檔案的程式專案?

```
$ gcc -Wall main.c binui.c -lm
```

加入-lm是因為使用到math.h所對應的函式庫libm



練習：簡易計算機

試撰寫一程式，讓使用者選擇運算子(+、-、*、/)。如果使用者輸入的不是這些運算子輸出ERROR，否則讓使用者輸入兩個數字後印出計算後的結果。

註1: 以迴圈讓使用者可以持續計算，直到輸入Q為止。

註2: **將所有數學運算以函式的方式實作在一個獨立的原始檔中。**

輸入	輸出
+ 3 5	8
E	ERROR
* 5 4	20
Q	BYE

遞迴函式(Recursive FunctiOn)

遞迴函式

遞迴函式指的是自己呼叫自己的函式，對於某些問題來說非常有用。

例如，輸入n計算1+2+3+…+n的總和。假設函式`int sum(int n)`可以回傳1+2+3+…+n的總和，近一步觀察可以發現

```
sum(n)= n + sum(n-1)
      = n + (n-1) + sum(n-2)          // sum(n-1) = (n-1) + sum(n-2)
      = n + (n-1) + (n-2) + sum(n-3) // sum(n-2) = (n-2) + sum(n-3)
      ...
      = n + (n-2) + … + 2 + sum(1)    // 當n=1時，sum(1)=1 (我們稱n=1時為遞迴停止條件)
      = n + (n-2) + … + 2 + 1
```

如何以遞迴實作sum(n)

```
1.  int sum(int n) {
2.    if(n == 1) {
3.      return 1;
4.    } else {
5.      return n + sum(n-1);
6.    }
7.  }
```

遞迴函式(續)

大部分問題都能夠用迴圈或遞迴函式來解決，但有些問題用遞迴函式來解決會比較直覺，例如費式係數(fibonacci number)。

$$F(n) = F(n-1) + F(n-2), F(0) = 0, F(1) = 1$$

遞迴函式因為是函式呼叫，所以跟迴圈比起來會使用較多的記憶體。

遞迴停止條件要正確，否則遞迴函式會無法結束(類似無窮迴圈)。

練習：費式係數

輸入正整數n，輸出費式係數F(n)。

輸入	輸出
0	0
1	1
2	1
10	55