函數

為什麼要用函數?

- 把大問題拆成很多小任務來完成
- 先思考程式的架構，再顧慮細節
- 可重複使用
- 有許多現成可用的函數可利用，不須重寫

用到函數的程式範例

```c
#include <stdio.h>
#define WIDTH 40
void starbar(void); /* function prototype */
int main(void){
      starbar();
      printf("%s\n", "NTHU CS");
      printf("%s\n", "101, Sec. 2, Kuang Fu Rd.");
      printf("%s\n", "Hsinchu, 300 Taiwan");
      starbar();
      return 0;
}
void starbar(void){ /* function definition */
      int count;
      for (count = 1; count <= WIDTH; count++)
            putchar('*');
      putchar('\n');
}
```

執行結果

```c
#include <stdio.h>
#define WIDTH 40
void starbar(void); /* function prototype */
int main(void){
      starbar();
      printf("%s\n", "NTHU CS");
      printf("%s\n", "101, Sec. 2, Kuang Fu Rd.");
      printf("%s\n", "Hsinchu, 300 Taiwan");
      starbar();
      return 0;
}
```

```
****************************************
NTHU CS
101, Sec. 2, Kuang Fu Rd.
Hsinchu, 300 Taiwan
****************************************
```

宣告函數

- function prototype (原形)

```
void starbar(void);

int main(int argc, char *argv[]);
```

- 讓編譯器知道有哪些可用的函數
- 定義函數的輸入、輸出

定義函數實體

- function definition

```
void starbar(void){ /* function definition */
   int count;
   for (count = 1; count <= WIDTH; count++)
       putchar('*');
   putchar('\n');
}
```

- 描述黑盒子的行為是什麼

傳入參數

```
void f(int x);
void f(int* x);
void f(int& x);
```

pass by value

之後會介紹

區域變數

```c
#include <stdio.h>
void f(int x);
int main(void){
        int x = 1;
        printf("main: %d\n", x);
        f(x);
        printf("main: %d\n", x);
        return 0;
}
void f(int x){
        x++;
        printf("f: %d\n", x);
}
```

輸出：

```
main: 1
f: 2
main: 1
```

傳回參數

```
int imin(int n, int m){
      int min;
      if (n < m) {
            min = n;
      }
      else {
            min = m;
      }
      return min;
}
```

自動型別轉換

```
int what_if(int n)
{
      double z = 100.0/ (double) n;
      return z; /* what happens? */
}
```

遞迴函數
Recursive Function

Tail Recursion