第十三章

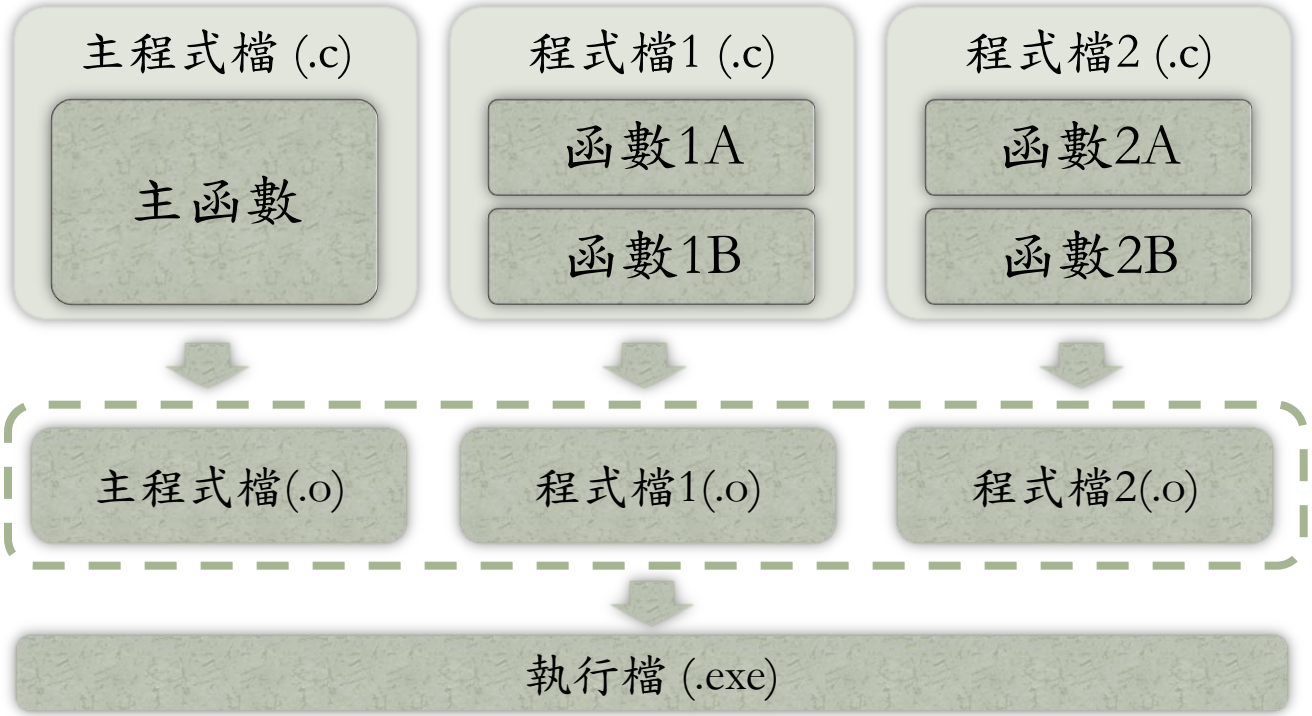
大型程式的發展

大綱

- 模組化程式
- 程式碼、目的碼、執行檔
- 變數範圍
- 條件式編譯
- 命令列參數

程式的模組化

- 將具有特定功能的程式碼，單獨編寫成一個程式單元
- 程式模組化的概念可由下圖來表示



同一個檔案內有多個函數

- 下面的範例是將多個函數寫在一個檔案內

```
01 #include <stdio.h>
02 #include <stdlib.h>
03
04 void func_one();
05 void func_two();
06
07 int main()
08 {
09   func_one();
10   func_two();
11
12   system("PAUSE");
13  return 0;
14 }
15
```

```
16 void func_one()
17 {
18     printf("One\n");
19 }
20
21 void func_two()
22 {
23     printf("Two\n");
24 }
25
```

各別編譯的實作

- 將程式拆解成數個模組後各別編譯

main.c

```
01 #include <stdio.h>
02 #include <stdlib.h>
03
04 void func_one();
05 void func_two();
06
07 int main()
08 {
09   func_one();
10   func_two();
11
12   system("PAUSE");
13  return 0;
14 }
15
```

one.c

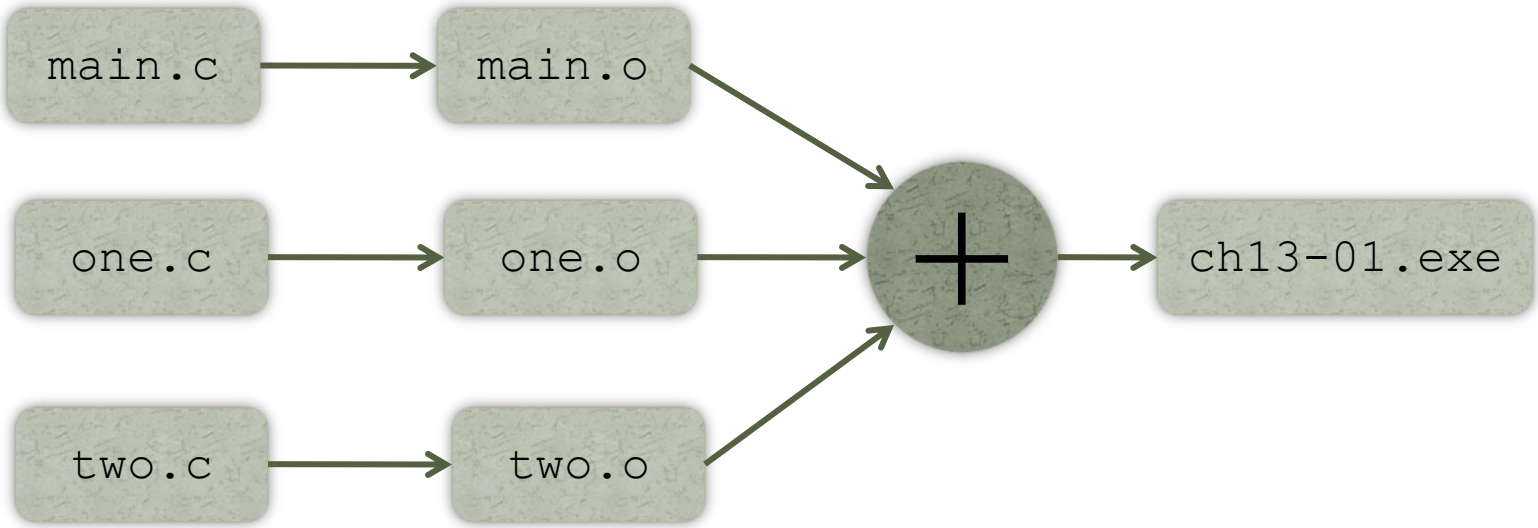
```
01 #include <stdio.h>
02 #include <stdlib.h>
03
04 void func_one()
05 {
06     printf("One\n");
07 }
```

two.c

```
01 #include <stdio.h>
02 #include <stdlib.h>
03
04 void func_two()
05 {
06     printf("Two\n");
07 }
```

程式碼、目的碼、與執行檔

- 每個程式碼經過編譯後產生對應的目的碼
- 連接各個目的碼可以組合成一個可執行檔



使用全域變數

- 在各函數外宣告的變數為全域變數
- 利用extern可在函數內使用外部程式檔案宣告的全域變數

```c
01 #include <stdio.h>
02 #include <stdlib.h>
03
04 void func_one()
05 {
06     extern int money;
07     money += 1;
08 }
```

one.c

```c
01 #include <stdio.h>
02 #include <stdlib.h>
03
04 int money = 1;
05 void func_one();
06
07 int main()
08 {
09   func_one();
10   printf("%d",money);
11   system("PAUSE");
12   return 0;
13 }
```

main.c

條件式編譯

- 條件式編譯
  - 可根據條件判斷來決定某個部份的程式碼是否要編譯
  - #if、#else 及 #endif 的使用格式如下：

#if、#else及#endif 的使用格式

```
#ifdef 識別字
    /* 如果識別字有被定義過，則編譯此部份的程式碼 */
#else
    /* 否則編譯此部份的程式碼 */
#endif
```

條件式編譯的範例

- #ifdef、#else 與 #endif 的使用範例

```c
#include <stdio.h>
#include <stdlib.h>
#include "config.h"

double area();

int main()
{
  int r = 3;
  printf("PI=%.6f\n",PI);
  printf("PERI=%.6f\n",PERI(r));
  printf("AREA=%.6f\n",area(r));

  system("PAUSE");
  return 0;
}
```

main.c

```c
#include "config.h"
#ifndef PI
   #define PI 3.14
#endif

double area(int r)
{
   return PI*r*r;
}
```

area.c

```c
#ifndef PI
   #define PI 3.14159
#endif

#define PERI(x) 2*PI*x
```

config.h

#if、#else、#elif 與 #endif 指令

- #if、#else、#elif與#endif指令
  - 其功能和選擇性敘述中的if-else指令類似
  - #if為真，則編譯其後的敘述，否則編譯#elif或#else後面的敘述

#if、#else及#endif 的使用格式

```
#if 運算式1
    /* 若運算式1的結果成立，則編譯此區段的敘述 */
#elif 運算式2
    /* 若運算式2的結果成立，則編譯此區段的敘述 */
#elif 運算式3
    :
#endif
```

命令列參數的使用

- 假設在命令提示字元模式下鍵入

```
C:\>type a.txt
```

  - a.txt可視為是一個參數，傳入type 命令中
- 從命令列裡傳遞參數給 main() 函數

命令列引數的使用格式

```
int main(int argc, char *argv[])
{
        /* main() 函數裡的程式碼 */
}
```

  - 其中argc為參數個數，argv[]為參數字串陣列

命令列引數的使用範例

- 印出 argc 與字串陣列 argv[] 的值

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc,char *argv[])
{
   int i;
   printf("ARGC=%d\n",argc);

   for (i=0;i<argc;i++)
   {
       printf("ARGV[%d]=%s\n",i,argv[i]);
   }

   system("PAUSE");
   return 0;
}
```

```
C:\>ch13-04 first second
ARGC=3
ARGV[0]=ch13-04
ARGV[1]=first
ARGV[2]=second
請按任意鍵繼續 . . .
```

命令列引數的應用

- 將數字參數加總後顯示在畫面上

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc,char *argv[])
{
   int i,sum=0;

   for (i=1;i<argc;i++)
        sum += atoi(argv[i]);

   printf("SUM=%d\n",sum);

   system("PAUSE");
   return 0;
}
```

```
C:\>ch13-05 1 3 5
SUM=9
請按任意鍵繼續 . . .
```

Any question?