Chapter 02

程式架構的認識與實作

學習綱要

- 2-1 程式語言與演進
- 2-2 專案架構、程式架構及語言架構
- 2-3 演算法的認識
- 2-4 程式設計步驟
- 2-5 開發環境介面

學習目標

- 認識程式語言的演進。
- 認識 C 語言的架構與專案架構。
- 認識 C 語言程式架構。
- 能以演算法寫出程式架構。
- 瞭解程式設計步驟。
- 能以 CodeBlocks 編輯、編譯與執行程式。

2-1 程式語言與演進

程式語言

人與人之間溝通的工具稱爲語言，世界上因有許多民族，因其發源地不同，所以就有許多語言。例如，華語、英語及德語等。其次，人與電腦溝通的工具，則稱爲電腦程式語言。那麼爲什麼沒有電視語言、冰箱語言或冷氣語言呢？那是因爲這些機器的功能較爲簡單，只要幾個按鈕就能發揮其功能。但是電腦的功能就非常多，多到連用整個鍵盤的所有按鍵都無法表現其功能，所以必須使用一些類似單字所組成的片語與敘述來發揮其所有功能，這些單字與片語的集合就稱爲電腦程式語言，簡稱程式語言。就如同人類也無法用 26 個字母表達所有感受與思維，必須藉助這些字母的排類組合，先組成單字，再由單字組合成片語與句字，才能充分表達其思維。

程式語言的演進

電腦語言依其演化先後，大致可分爲低階語言 (Low-level programming language）與高階語言（High-level programming language），分別說明如下:

▶ 低階語言

低階語言依其演化先後，又可分爲機器語言（Machine language）與組合語言 (Assembly language)，分別說明如下：

機器語言

機器語言是由一堆 0 與 1 所組成的單字。例如，要在螢幕輸出 A，所需鍵入單字如下：

```
B2 41 B4 02 CD 21 CD 20
```

由於這些單字的來籠去脈過於複雜，絕非一般人所能體會。只因爲要於螢幕顯示一個 A，就要鍵入以上 8 個單字，那如果要計算一個正方形面積，其所需鍵入的單字可能比火車還長，還不如用筆算還快，哪需要計算機呢！其次，電腦剛發明時並沒有鍵盤，要鍵入以上單字，必須用 8 個位元一組的指撥開關，分別輸入 8 次 (共 8 個單字)。例如，B2 是由 (10110010)

所組成，則必須先將 8 個指撥開關分別設定爲 (on,off,on,on,off,off,on,off)，由此可見當時使用機器語言的艱辛，眞是未蒙其利，先受其害。

組合語言

當時學習與使用機器語言是艱辛的，而且電腦的體積又非常龐大，通常只有作爲學術機關研究用。但是當時科學家並未氣餒，爲了推廣這一神奇而且好用的工具，所以有組合語言的發展。所謂組合語言就是使用一些助憶碼（Assembly mnemonic），這些助憶碼類似一些簡單的英文單字，以便協助使用者記憶與書寫所要完成的程式。例如，上例中要將字元 A 顯示於螢幕，所需組合語言如下：

```
MOV DL,41
MOV AH,2
INT 21
INT 20
```

其中，MOV 是 MOVE 的縮寫，41 是字元 A 的 ASCII 碼，DL 與 AH 則是 CPU 的內部暫存器，INT 21 是呼叫字元顯示副程式，INT 20 則是呼叫程式結束處理副程式。也就是程式發展已經有一脈絡可循，一切遵循一些簡單規則，方便使用者的查詢與記憶。由於有了組合語言的好工具，學習電腦終於可以突破實驗室，推廣到大學以上的資訊電子相關科系。

▶ 高階語言

有了組合語言，電腦使用的方便性可說向前邁向一大步，但是電腦的使用卻僅限於資訊電子相關科系，因爲不同的 CPU 就要有不同的組合語言，也就是說此程式語言沒有硬體相容性。其次，這些助憶碼也要花費很多時間學習，而且每一小小功能的程式長度都要超過一頁，所以科學家並不以此爲滿足，所以有高階語言的研發。例如，前例要在螢幕顯示字元 A，若使用 C 語言，則只要撰寫如下：

```
printf("%c",'A');
```

又例如，面積的計算，程式如下：

```
int a=15;
int b=8;
int c;
c=a*b;
printf("面積=%d\n",c);
```

以上程式與人類學習數學的習慣非常接近，已經可以被大部分人接受。

程式語言的選擇

電機電子群的專長是電機電子設備的控制，學習程式語言的目的主要是銜接二、三年級的嵌入式晶片控制。目前所有的嵌入式系統的程式語言都是使用 C 語言，所以本書選擇 C 語言作爲電機電子群程式設計實習的語言。其次，物件導向已經是程式語言領域的標準配備，C++ 是繼承 C 語言的物件導向語言，所以本書第十二章導入物件導向程式設計語言時，選擇 C++ 闡述物件導向的程式設計。

2-2 專案架構、程式架構及語言架構

專案架構

一個可執行的 C 語言稱爲一個專案，每個專案可由至少一個 main() 函式、若干函式、組合語言函式、C 標準函式庫、自訂函式庫所組成，且由 main() 函式當作起動函式，如 2-1 圖所示。

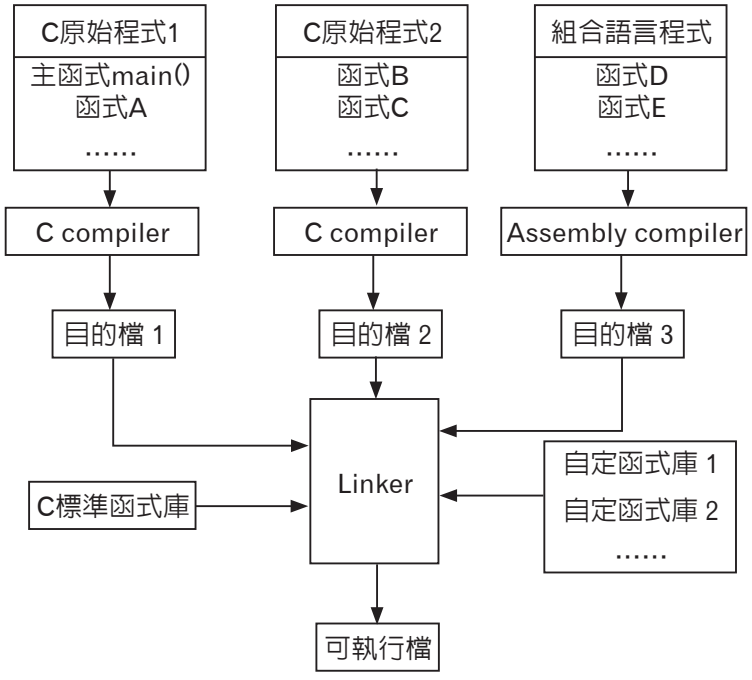


圖 2-1 C 程式專案架構

依據模組化原則，大的專案都要先分解爲很多單一函式。所以實用的 C 程式都是由很多單一功能函式組合而成，因爲單一功能函式容易撰寫、除錯與編譯，這些單一功能都先使用編譯程式 (Compiler) 編譯成目的檔「*.obj」，如上圖的目的檔 1、或目的檔 2。每當要改變程式功能時，只要修改對應的函式，編譯此函式，不用全部重新編譯，當然效率也高。最後階段再使用連結程式（Linker），連結所有目的檔「*.obj」、對應的標頭檔 (C 標準函式庫共有 29 個標頭檔) 或已經編譯完成的自訂函式庫，即可完成可執行檔。

程式架構

以上專案架構有點複雜，早期學習 C 語言程式設計，都要自己使用 Compiler 編譯程式，將程式編譯成目的程式 (「*.obj」)。若需修改，則修改後再編譯。最後再使用 Linker 連結所有目的程式 (「*.obj」) 成爲可執行檔案「*.exe」。所幸，目前有 Code::Blocks 統整與管理以上瑣事。以下是使用 Code::Blocks 開啓新專案時，系統準備好的基本程式架構。

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    printf("Hello world!\n");
    return 0;
}
```

▶ 標頭檔

所有 C 語言程式都要載入兩個標頭檔

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

這些標頭檔內含許多由 C 語言事先編寫好的函式，例如內含許多輸出入函式供我們呼叫，只要載入它們兩個，我們就可以輕鬆作出輸出入的功能。例如：本例的

```
printf("Hello world!\n");
```

就可以輸出「Hello world!」。我們作爲程式設計師，只要靈活呼叫這些函式，就可以完成許多常見的工作。

▶ 新增/刪除檔案

若要新增檔案，也是點選功能表「File/New/File」即可新增 C 函式或組合函式；若要修改函式，也是先點選函式名稱再修改。

▶ 執行程式

完成程式的編輯，最後點選「Build/Build and Run」即可編譯所有函式、連結所有目的檔，且執行程式。

以上是最簡單的 C 程式，我們剛剛強調 C 語言是由許多函式組合而成，以下則是自己新增函式的 C 程式架構，程式分爲四大區塊，分別是函式引用區、全域變數宣告區、函式實作區、主程式區，如表 2-1 所示：

▶ 表 2-1　C 程式架構

標題	程式
函式引用區（載入所需標頭檔）	`#include <stdio.h>` `#include <stdlib.h>`
全域變數宣告區	`int s;`

函式實作區	`int  add (int a,  int b ){`
	`    int  c;`
	`    c = a + b;`
	`    return (c);`
	`}`

主程式區 main()	`int main() {`
	`    s = add (6, 2);`
	`    printf("%d",s);`
	`    return 0;`
	`}`

圖 2-2 是專案管理窗格，表示此專案，目前含兩個檔案，分別是 main.c 與 m3.c。

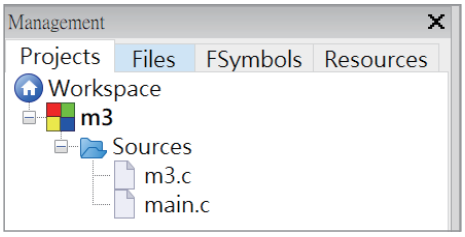


圖 2-2　專案管理窗格

圖 2-3 是以上專案的檔案結構。

圖 2-3　專案檔案結構

.cbp 是專案檔案名稱，main.c 是一個函式檔，目的檔 (.obj) 在 obj 資料夾裡面，執行檔 (*.exe) 在 bin 資料夾裡面。

▶ C語言架構

C 語言的指令分爲資料型態的宣告、運算子、決策指令、迴圈指令、函式、指標，其語言指令架構如表 2-2 。

▶ 表 2-2　C 語言指令架構表

C語言指令架構		
指令分類	指令	本書章節
資料型態的宣告	常數 :const #define char,int,float,double,short,long,void,struct	3,4,9
運算子	算術運算 :=,+,-,*,/,%,++,-- 關係運算 :>,<,>=,<=,==,!= 邏輯運算：!,&&,|| 位元運算：!,&,|,^,>>,<<	6
決策指令	if,else,switch,case	7
迴圈指令	for,while,continue,break	8
函式	函式	5,11
指標	指標	10

以上指令對應章節如表 2-2 的「本書章節」，我們將會在以下各章節陸續介紹。

自我練習

1. 以 C 語言開發程式的敘述，下列何者錯誤？（統測 111）
   (A) 程式需要經過編譯及連結產生可執行檔，才能夠執行
   (B) 使用 #include 前置處理命令時，命令結尾需要加上分號
   (C) main ( ) 是一個函式，程式執行時從 main ( ) 函式開始執行
   (D) 單行註解可用 2 個斜線 (//) 開頭

2-3 演算法的認識

演算法基本概念

演算法（algorithm）是指電腦程式完成一項工作所需要『步驟』的集合。演算法嚴謹定義如下：

在有限 (finite) 的步驟 (step) 所構成的集合中，依照給定輸入 (input) ，依序執行每個明確 (definite) 且有效 (effective) 的步驟，以便能夠解決特定的問題，而且步驟的執行必定會終止 (terminate)，並產生輸出 (output)。

演算法的流程表示

常見的演算法流程表示有三種，分別是自然語言、虛擬碼與流程圖。分別說明如下：

▶ 自然語言(Natural language processing)

自然語言就是使用我們日常生活的文字表示或已經熟悉的數學語言。例如，以下是求 9 開根號的演算法。

1. 使用迴圈從 1 到 9 分別求其平方。（此稱爲循序法）
2. 若其平方大於等於 9，此數即爲所求。例如：1 的平方是 1，2 的平方是 4，3 的平方是 9，9 已經大於等於 9，所以 3 即爲所求，演算結束。

▶ 虛擬碼(Pseudocode)

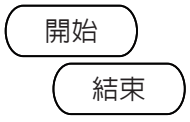
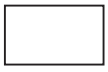
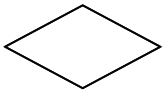
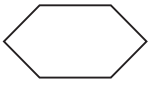
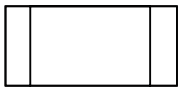
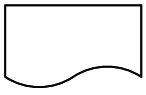
在自然語言中，有時嵌入一些慣用常用程式敘述，例如 for 代表迴圈，if 代表決策，這樣可以縮短文字長度，也會讓敘述更加清楚易懂。例如，以下是求 9 開根號的演算法。

```
for i=1 to 9{
    y=i*i
    if y>=9 then
        print(i)
        end for
}
```

流程圖(flow chart)

流程圖 (flow chart) 是利用各種方塊圖形、線條及箭頭等符號來表達解決問題的步驟及進行的順序，常用的流程符號如表 2-3：

▶ 表 2-3　流程圖符號表

編號	符號	意義
1		起迄符號。 代表流程圖的開始或結束，此符號若是開始，則僅有一出口，若是結束則僅有一入口。
2		輸入與輸出符號。 用來填入輸入與輸出的符號，此符號有一入口與一出口。
3		處理符號。 用來填入處理程序，此符號有一入口與一出口。
4		決策符號。 用來表示決策分歧點，此符號的出口至少有兩個，分別是 true 或 false。
5		重複符號。 聲明重複指令的起點與離開條件，通常配合下面的連結符號表示重複的範圍。
6		連結符號。 可配合上面的重複符號或移至另一頁的起點。
7		副程式。(副程式又稱函式) 用來表示另一個副程式。
8		程式流向符號。 用來連結以上符號，說明程式的流向。
9		代表輸出至螢幕。 將資料由螢幕輸出。
10		代表輸出至印表機。 將資料由印表機輸出。

例如，圖 2-4 是求 9 開根號的演算法。

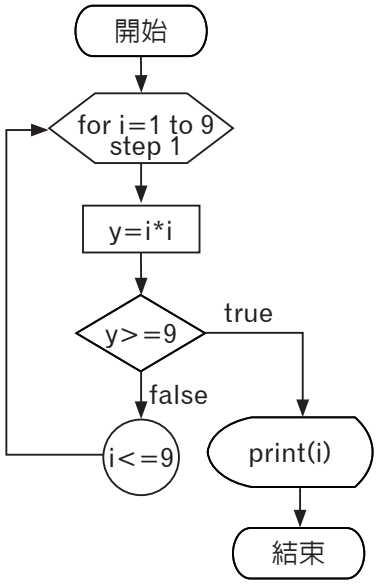


圖 2-4　開根號流程圖

又例如，圖 2-5 的流程圖，可以將輸入成績分類，分爲 A、B、C、D 等。

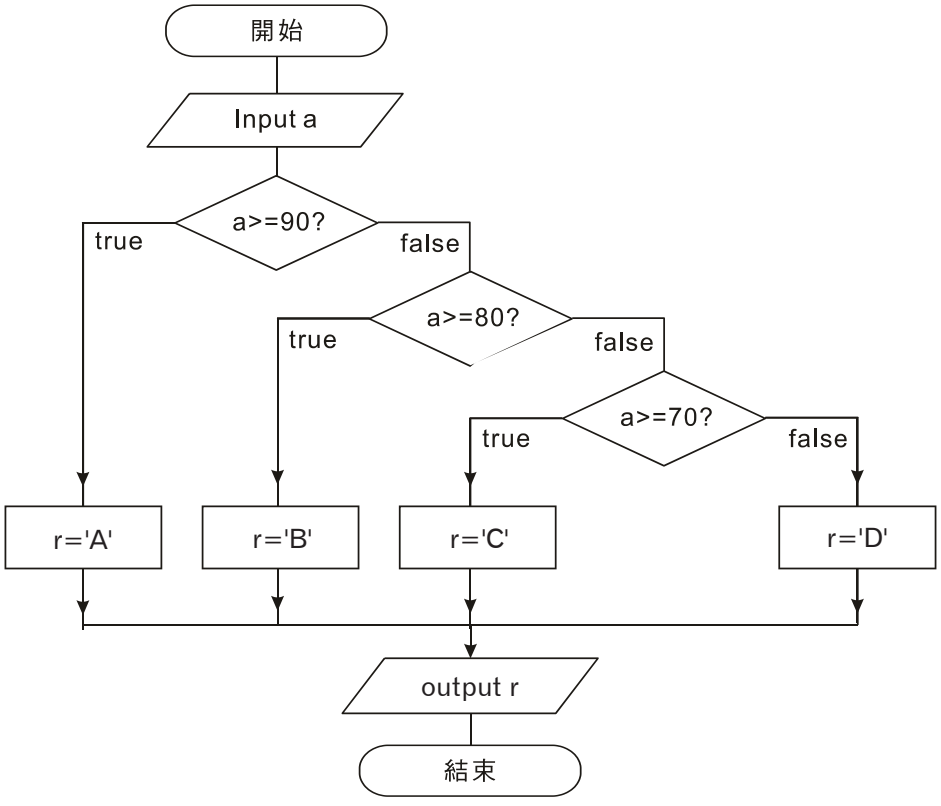


圖 2-5　成績分類流程圖

2-4　程式設計步驟

範例2-4a

已知長方形的長與寬，請寫一個程式，求其面積。

程式設計解題步驟

1. 資料的數位化
   (1) 本題若使用掌上型計算機，其方法是直接將鍵入「長 * 寬 =」，就可得到答案。例如，鍵入「15*8=」，就可得到答案「120」。
   (2) 但使用 C 語言，必須先將所要計算的值先以一個英文代號儲存，例如 a,b 或 score，以上代號在程式設計的領域，稱爲「變數」。例如：

```
a=15
b=8
```

   設定變數的目的是將資料與程式分開，這樣當資料改變時，還可重複計算。計算過程都是以變數與程式指令描述，此稱爲程式設計。程式設計的優點是有一致性，只要第一次對，往後都對。掌上型計算機就沒有一致性了，每次都要按兩次，才能確認計算結果是否正確，且沒無法重複使用。以上變數的命名請看 4-1 節。
   (3) 選用資料型態。C 語言的資料型態請看 3-2 節，本例先選用 int(整數型態)，且每一個敘述都要以分號 (;) 結束，所以程式如下：

```
int a=15;
int b=8;
```

2. 寫出演算法。本例是輸入長方形的長與寬來計算面積，所以演算法如下：

```
面積=a*b
```

3. 使用變數與程式指令，完成以上演算法。本例是計算面積，面積也要使用變數儲存，本例使用 c，所以是

```
c=a*b;
```

4. 輸出結果。

```
printf("面積=%d\n",c);
```

5. 以上全部程式如下：

```
int a=15;
int b=8;
int c;
c=a*b;
printf("面積=%d\n",c);
printf("%d",c);
printf(c)
```

6. 配合 C 語言的程式架構，全部程式如下：

```
#include <stdio.h>
#include <stdlib.h>
int main(){
    int a=15;
    int b=8;
    int c;
    c=a*b;
    printf("面積=%d\n",c);
    return 0;
}
```

2-5 開發環境介面

目前 C/C++ 語言的程式開發環境有 Code::Blocks、Dev-C++，因爲 APCS（Advance Placement Computer Science, 大學程式設計先修檢測）考場採用 Code::Blocks，所以本書介紹 Code::Blocks 如下：

Code::Blocks

Code::Blocks 是目前最熱門的 C/C++ 自由整合式開發環境（IDE），Code::Blocks 官網（https://www.codeblocks.org/），其下載與安裝方式如下：

▶ 下載軟體

點選「Downloads」，畫面如圖 2-6：

圖 2-6　Code::Blocks 下載點一

點選「Download the binary release」，畫面如圖 2-7：（往下捲動）

File	Download from
codeblocks-20.03-setup.exe	FossHUB or Sourceforge.net
codeblocks-20.03-setup-nonadmin.exe	FossHUB or Sourceforge.net
codeblocks-20.03-nosetup.zip	FossHUB or Sourceforge.net
codeblocks-20.03mingw-setup.exe	FossHUB or Sourceforge.net
codeblocks-20.03mingw-nosetup.zip	FossHUB or Sourceforge.net
codeblocks-20.03-32bit-setup.exe	FossHUB or Sourceforge.net
codeblocks-20.03-32bit-setup-nonadmin.exe	FossHUB or Sourceforge.net
codeblocks-20.03-32bit-nosetup.zip	FossHUB or Sourceforge.net
codeblocks-20.03mingw-32bit-setup.exe	FossHUB or Sourceforge.net
codeblocks-20.03mingw-32bit-nosetup.zip	FossHUB or Sourceforge.net

圖 2-7　Code::Blocks 下載點二

初學者請直接點選「codeblocks-20.03mingw-setup.exe」，因爲它才有內含「C/C++」編譯軟體，否則需自備編譯軟體。

▶ 安裝軟體

下載的檔案即是「.exe」執行檔，請在檔案總管「下載區」點按程式兩下，就可自動執行，開啓「Code::Blocks」，畫面如圖 2-8：

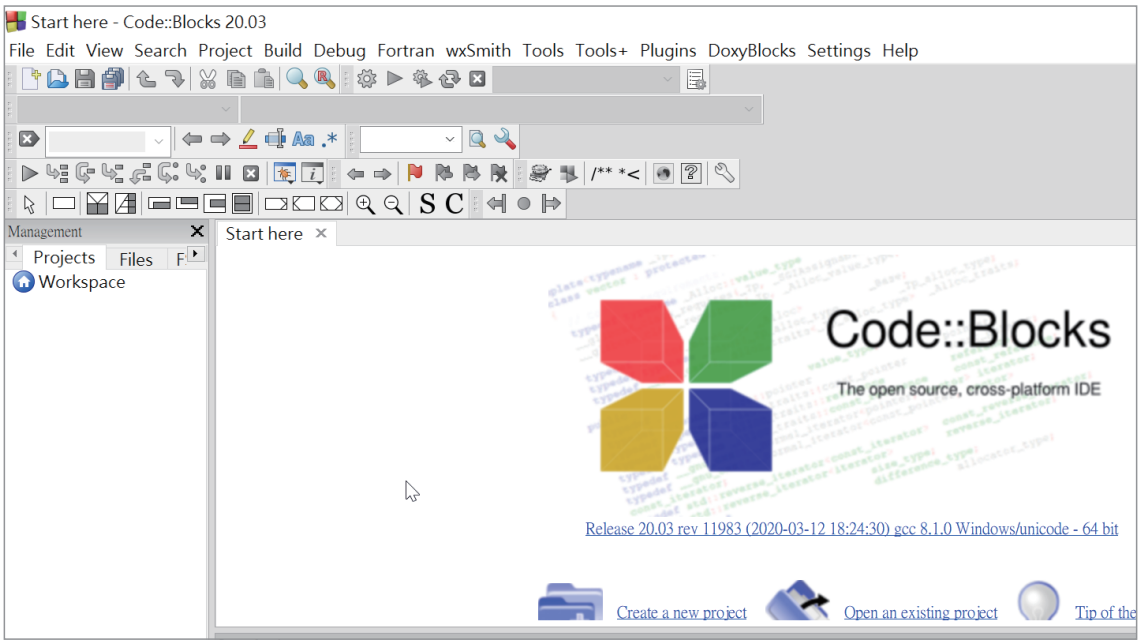


圖 2-8　Code::Blocks 初始畫面

範例2-5a

試寫一程式，將 2-4 節的面積計算程式輸入與執行。

操作步驟

1. 開新專案。點選功能表的「File → New → Project」，畫面出現要求點選專案類型，本例點選『Console application』／『Go』，如圖 2-9。

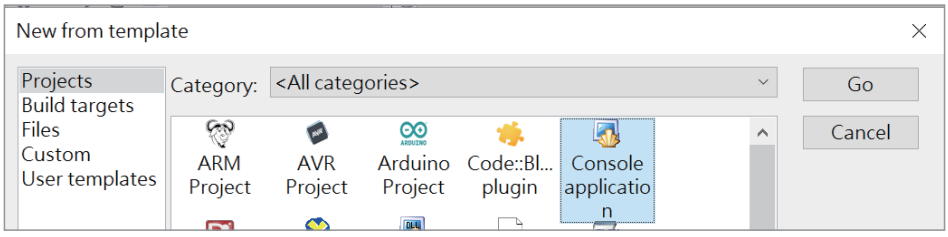


圖 2-9　開新專案一

2. 出現歡迎畫面，如圖 2-10，請點選「skip」。

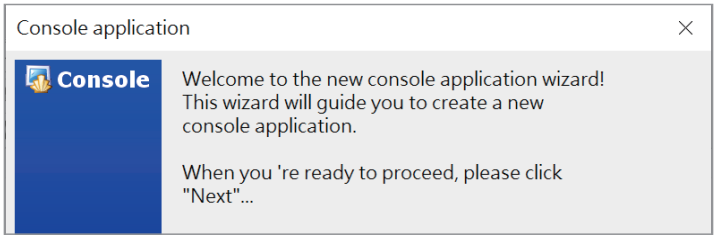


圖 2-10　開新專案二

3. 接著，要求點選開發 C 或 C++，本例點選「C」，如圖 2-11。

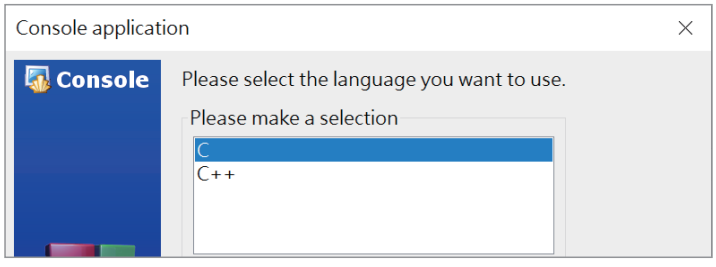


圖 2-11　開新專案三

4. 點選專案名稱與資料夾位置，本例於「Project title:」輸入 a1，資料夾位置點選 D：\test，如圖 2-12。

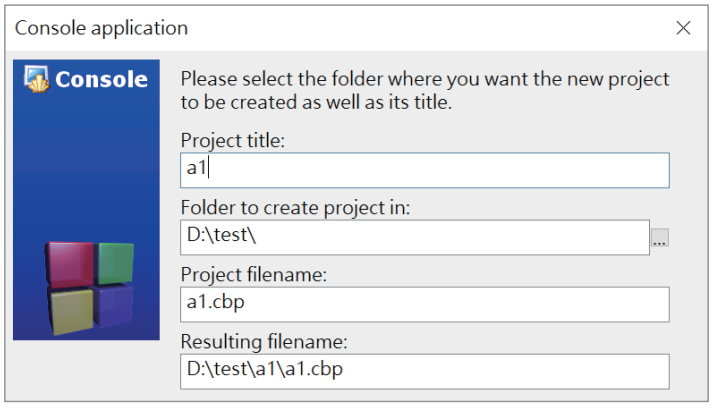


圖 2-12　開新專案四

5. 點選輸出路徑，本例使用預設值，如圖 2-13。

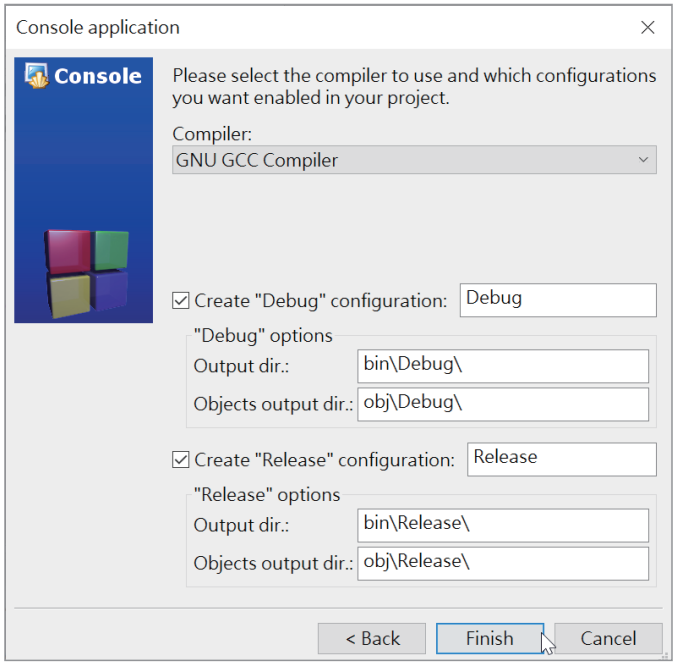


圖 2-13　開新專案五

6. 完成以上設定，並展開下圖 Sources 的 main.c，畫面如圖 2-14。

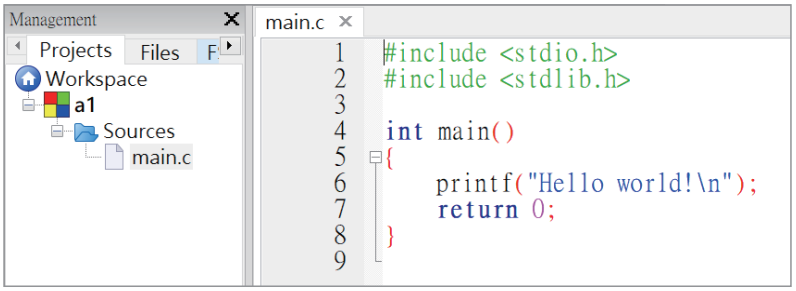


圖 2-14　main.c 程式架構

7. 編譯程式。點選功能表的「Build → Build」。
8. 執行程式。點選功能表的「Build → Run」，畫面如圖 2-15。
（以上 Step 7、8 兩個步驟，亦可點選功能表或工具列的「Build and Run」。）

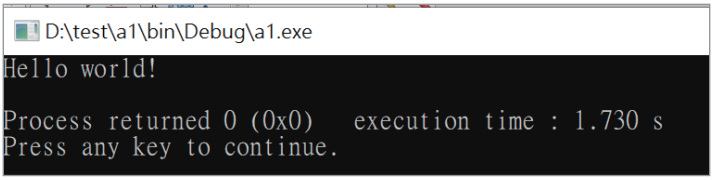


圖 2-15　執行結果

9. 以上「printf("Hello world! \n");」，就是本例所執行的程式，往後要寫任何的程式，只要將此敘述刪除，並在此位置寫下程式即可。
10. 開啓檔案總管，全部檔案如圖 2-16：（本例於 Step 4 的「Project title」輸入爲 a1，資料夾位置點選 test，系統會自動建立 a1 資料夾）。

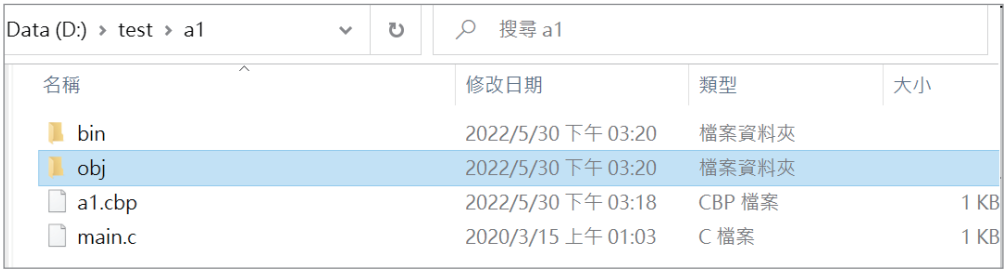


圖 2-16　C 專案檔案結構

11. 調整編輯器字型大小。點選功能表「Settings」/「Editor」，畫面如圖 2-17，繼續點選「Choose」，即可依照指示完成字型大小的設定。

圖 2-17　Code::Block 編輯字型設定

12. 以下是開新專案時，main.c 的程式樣版。

```
#include <stdio.h>
#include <stdlib.h>
int main(){
    printf("Hello world!\n");
    return 0;
}
```

13. 以下程式可計算長方形的面積。程式執行結果

```
int a=15;
int b=8;
int c;
c=a*b;
printf("面積=%d\n",c);
```

14. 將 printf() 函式刪除，並鍵入以上程式。全部程式如下：

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int a=15;
    int b=8;
    int c;
    c=a*b;
    printf("面積=%d\n",c);
    return 0;
}
```

15. 執行程式。點選功能表「Build/Build and Run」，並觀察執行結果。若有錯誤，請修改，再執行程式。

自我練習

1. 請寫一個程式，可以指派長方體的長、寬、高，並計算表面積與體積。