Course03

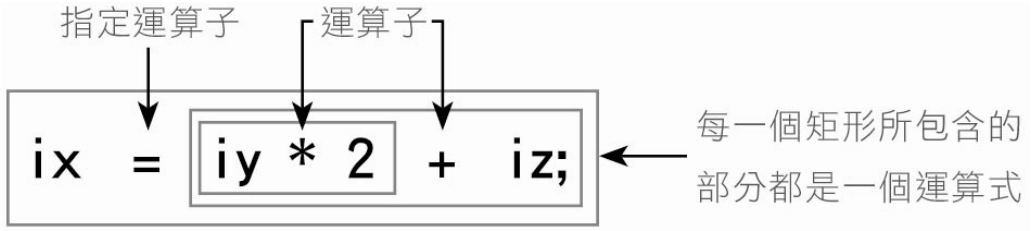
程式羅輯

選擇性敘述

運算式與運算子

- C 所提供的運算子可細分成以下 3 種
  - 算數運算子
    - ix = ia + ib; // + 運算子
    - fy = fc * fd – 2.0f; // - 與 * 運算子
  - 邏輯運算子
    - a = x && y; //* && 為 AND 運算子的符號
    - b = w || z ; // || 為 OR 運算子的符號
  - 關係運算子
    - ix >= iy // >= 代表大於等於
    - cz == ca // == 代表判斷關係上的等於

- 「=」在 C 語言的定義下稱為指定運算子
  - 將等號右邊的運算結果(右值)「指定」到等號左邊的變數(左值)內



- 算數運算子
  - + 、- 、 * 、/
    - 注意除號應用在整數上的問題
  - 正號「+」，負號「-」
    - 與加減符號是一樣的，編譯器會自動幫你進行判斷
    - 基本規則：正負號的計算優先於四則運算
    - EX： **iy = -10 - -ix;**
      - 先計算 -ix, 然後 -10 減去 -ix 的內容,最後在存到 iy
  - %：餘數運算子
    - 就是求取兩個數相除後的餘數
    - 唯一的要求是除數與被除數都必須是整數

指定與算數運算子

- 四則運算是
  - 先乘除後加減
  - 正負號的順序高於四則運算
  - 括號運算子 ( ) 擁有最高的優先權

優先順序	運算子	類別	結合律	範例
高	( )	括號運算子	由左至右	(ix - iy) /(ix - ia)
	+(正號)、-(負號)	一元運算子	由右至左	-ix + iy
	*、/、%	算數運算子	由左至右	ix * iy % iz
	+、-	算數運算子	由左至右	ix + iy - iz
低	=	指定運算子	由右至左	ix = iy + iz

關係與邏輯運算

/Course/101001001/c/Course03_Logic_Selective

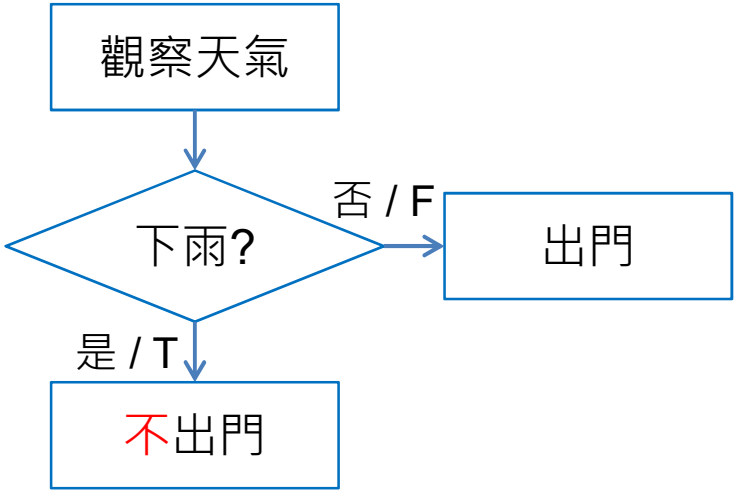
- C 語言提供了 6 個關係運算子

關係運算子	意義	範例	說明
>	大於	ia > ib	判斷 ia 是否大於 ib
>=	大於等於	ca >= cb	判斷 ca 是否大於等於 cb
<	小於	ia < ib	判斷 ia 是否小於 ib
<=	小於等於	ca <= cb	判斷 ca 是否小於等於 cb
==	等於	fx == fy	判斷 fx 是否等於 fy
!=	不等於	dx != dy	判斷 dx 是否不等於 dy

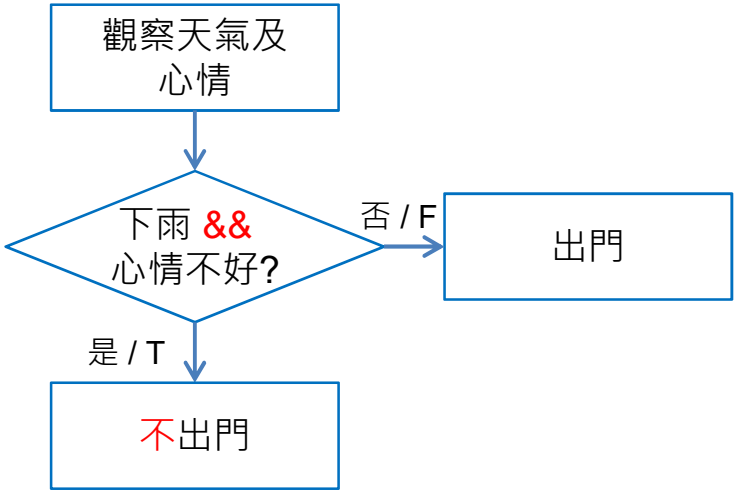
- 關係運算子中的等於「==」比指定運算子等號多一個等號「=」，使用時絕對不要弄錯。
- 比較的結果只有兩種
  - 真 (true) 表示比較成立 。(**1** 表示真)
  - 假 (false) 則表示比較不成立。(**0** 代表假)

邏輯運算

- 邏輯運算就是判斷一個條件是否成立
  - 如果今天下雨，我就不出門。
    - 「今天下雨」就是一個條件
    - 今天真的下雨了，條件成立，以真(true)來表示
    - 沒有下雨就是條件不成立，就以假(false)來表示



- 同時判斷多個條件才能發揮它的真正實力
  - 如果今天下雨**且**我的心情不好，我就不出門
    - 今天下雨與心情不好，當這兩個都成立時，就不出門
    - 但，就算今天下雨了，但是心情很好，還是會出門



- 每一個條件是否成立是以 true 或是 false 來表示
- 組合**邏輯條件(true/false)**的就是邏輯運算子
  - and（且），C 語言以 && 來代表
  - or（或），C 語言以 || 來代表
  - not（否），C 語言以 ! 來代表
- && 與 || 必須連接兩個運算元，而 ! 為一元運算子

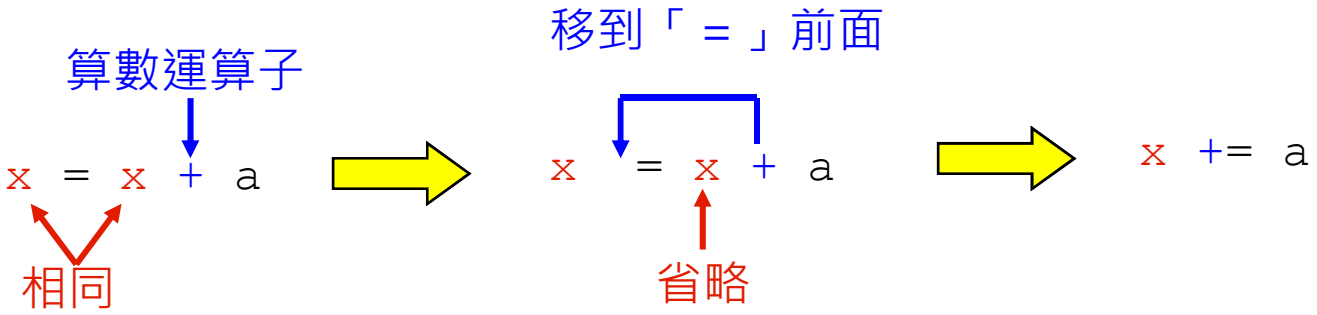
- 算數、關係與邏輯運算子間的運算優先順序

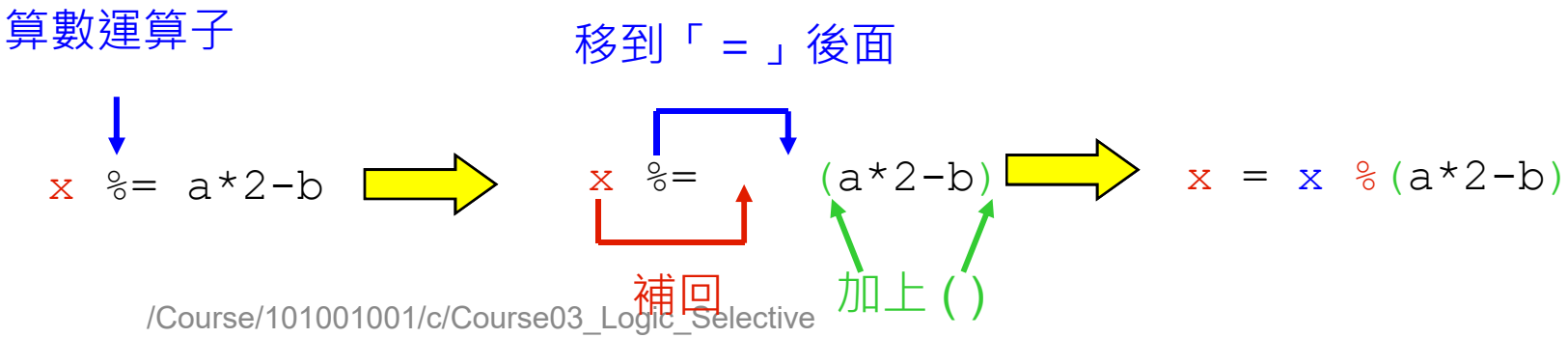
優先順序	運算子	類別	結合律	範例
高	( )	括號運算子	由左至右	(ix - iy) /(ix - ia)
	!、+(正)、-(負)	一元運算子	由右至左	-ix + iy 或 !(ix>iy)
	*、/、%	算數運算子	由左至右	ix * iy % iz
	+、-	算數運算子	由左至右	ix + iy - iz
	>、>=、<、<=	關係運算子	由左至右	ca > cb 或 cx <= cy
	==、!=	關係運算子	由左至右	cGet == ca 或 cSet != cx
	&&	邏輯運算子	由左至右	(ix>iy) && (ix>iz)
	||	邏輯運算子	由左至右	(ix<iy) || (ix>=iz)
低	=	指定運算子	由右至左	ix = iy + iz

- 關係運算子的優先順序大於邏輯運算子的 && 與 ||
  - 執行 ix>iy || ix>iz 時
    - 先執行 ix>iy 與 ix>iz 的運算，然後再執行兩者的 || 運算。
  - 如果怕混淆，可以寫成 (ix>iy) || (ix>iz)

運算式的規則

- 遞增運算子(++)與遞減運算子(--)
  - 來自變數內容遞增 1 與遞減 1 的精簡版寫法
- 算數運算子與指定運算子「＝」的簡便寫法
  - 例如：**x = x + a 可以寫成 x += a**





- 遞增遞減運算子與其他運算子混用的解讀
  - 例如：++ix*10 或 ix--<=6*--iy
  - 記住一下的處理規則
    - ++ 或 -- 出現在變數的左邊
      - 具有僅次於括弧 ( ) 的優先權，會先被執行
    - ++ 或 -- 出現在變數的右邊
      - 優先權比「=」還低，會到最後才執行

運算子的運算優先順序

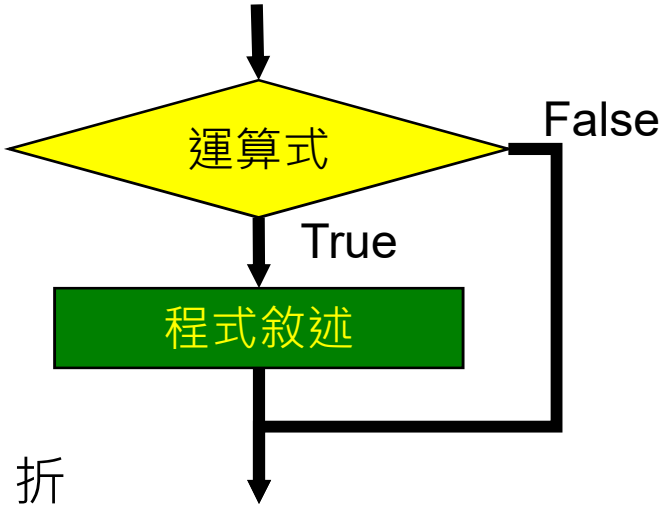
優先順序	運算子	類別	結合律	範例
1	( ) 與 [ ]	括號運算子	由左至右	(ix - iy) /(ix - ia)
2	~、!、+(正)、-(負)	一元運算子	由右至左	-ix + ~iy 或 !(ix>iy)
2	++、--(變數左邊)	遞增遞減運算子	由右至左	++ix、--iy
3	*、/、%	算數運算子	由左至右	ix * iy % iz
4	+、-	算數運算子	由左至右	ix + iy - iz
5	<<、>>	位元運算子	由左至右	ix<<3 或 iy>>2
6	>、>=、<、<=	關係運算子	由左至右	ca > cb 或 cx <= cy
7	==、!=	關係運算子	由左至右	cGet == ca
8	& (and)	位元運算子	由左至右	cht = ch1 & ch2
9	^ (xor)	位元運算子	由左至右	cht = ch1 ^ ch2
10	| (or)	位元運算子	由左至右	cht = ch1 | ch2
11	&&	邏輯運算子	由左至右	(ix>iy) && (ix>iz)
12	||	邏輯運算子	由左至右	(ix<iy) || (ix>=iz)
13	?:	條件運算子	由右至左	ix = ch > 'a' ? 10 : 5
14	=	指定運算子	由右至左	ix = iy + iz
15	++、--(變數右邊)	遞增遞減運算子	由左至右	ix++、iy--

結構化程式設計的概念

if 敘述

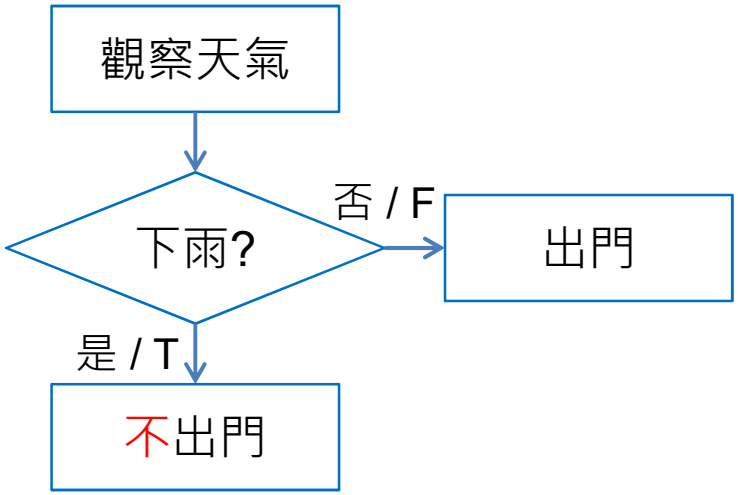
- if 敘述的最基本語法

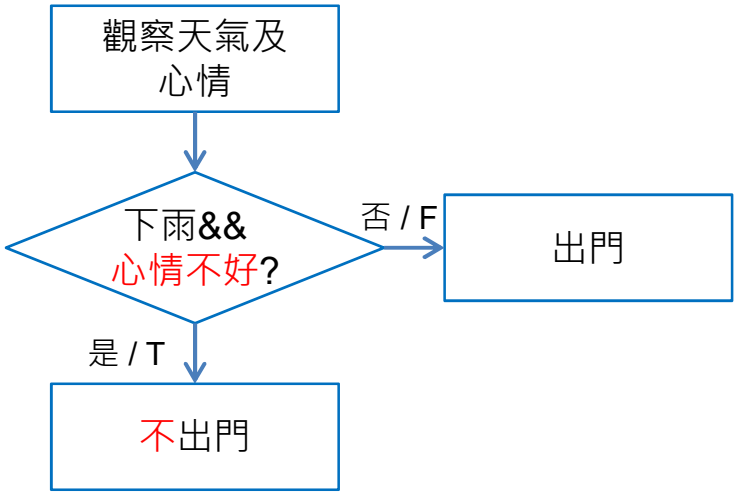
```
if ( 運算式 ) 敘述;
```



- 假設
  - iTotal 是購買總金額
  - iCount 是購買總瓶數
  - 購買總數超過 20 瓶販售總價錢可以打 85 折
  - 程式碼可寫成：

```
if ( iCount > 20 ) iTotal = iTotal * 0.85;
```





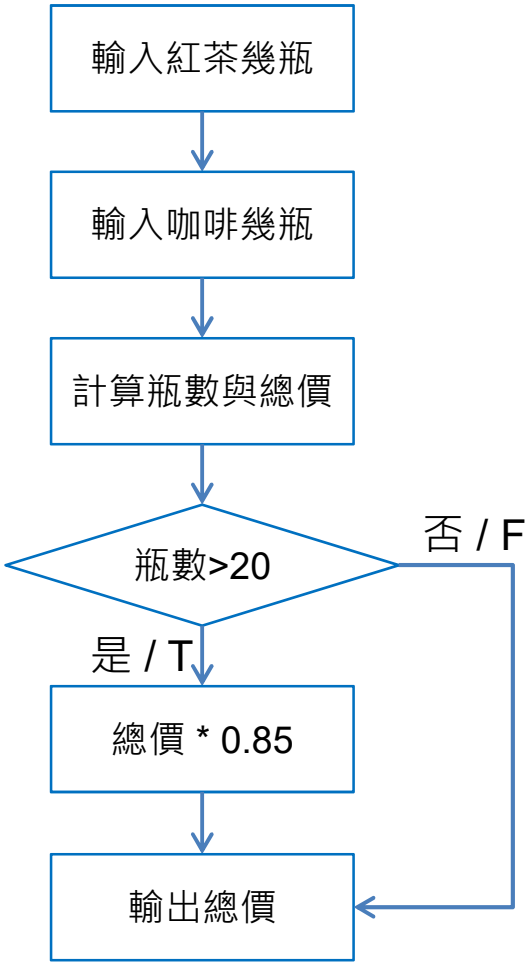

```
if(下雨)
{
        不出門;
}
```

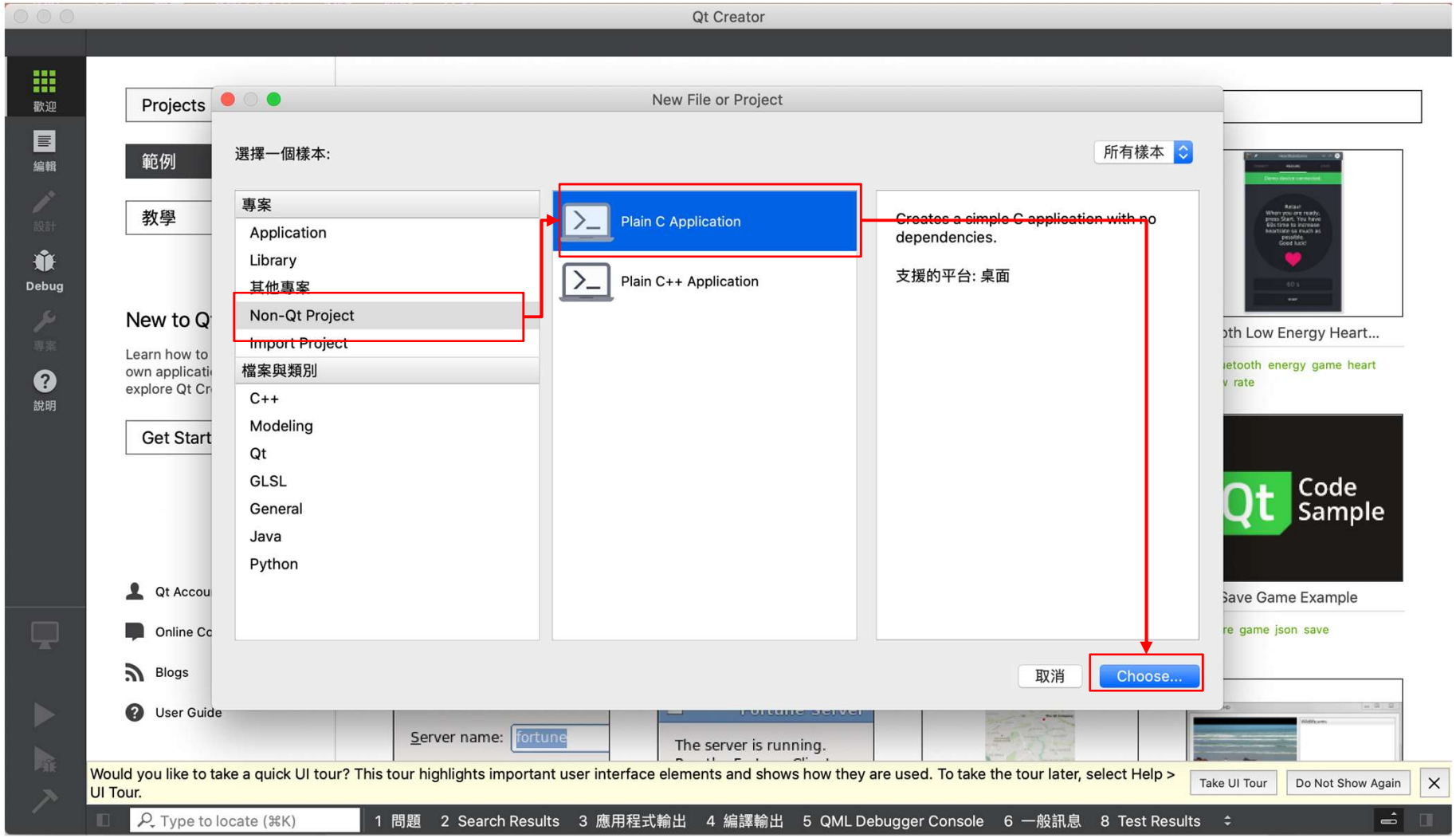
```
if(下雨&&心情不好)
{
        不出門;
}
```

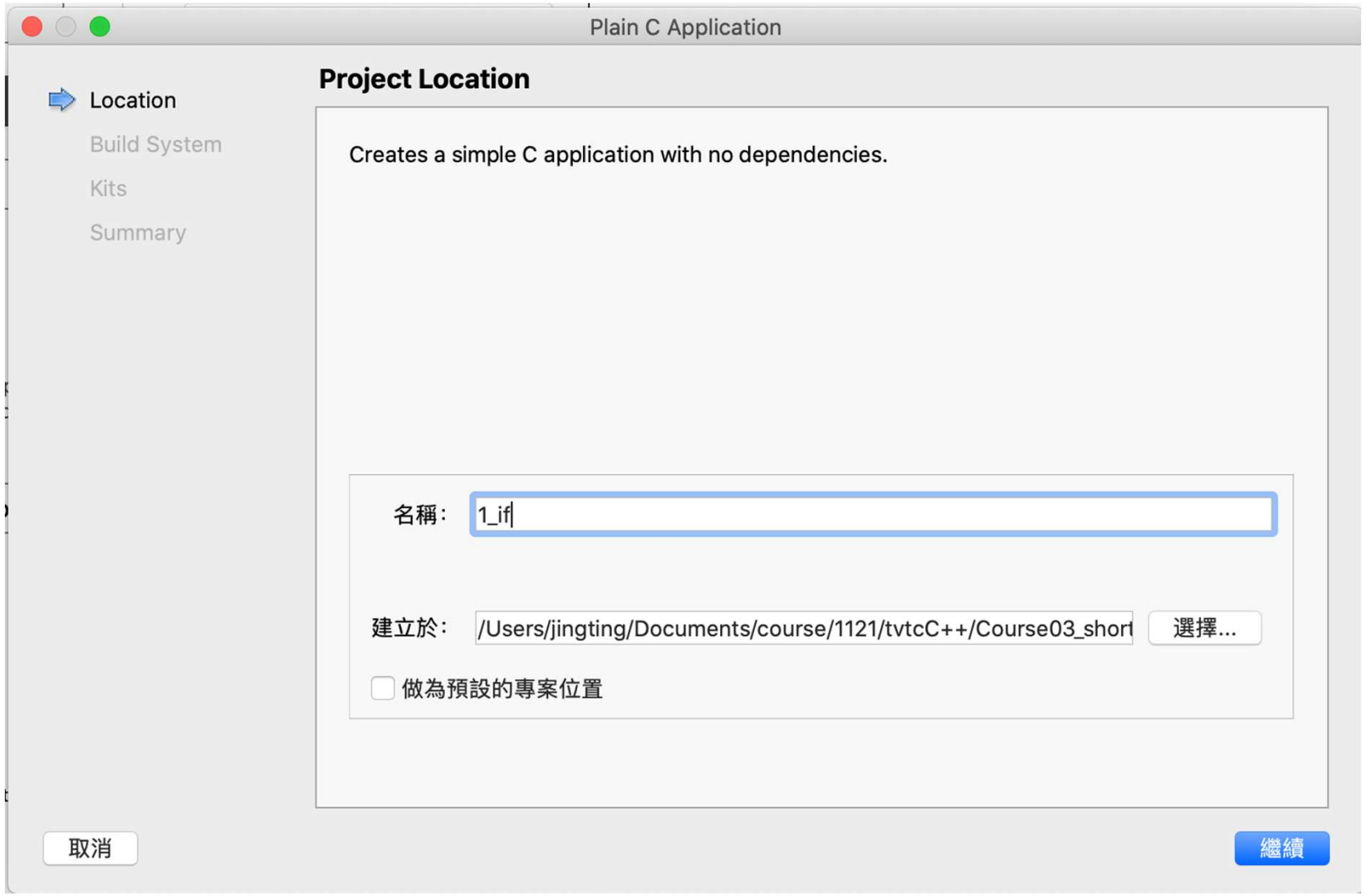
程式

- 紅茶與咖啡的售價分別是 25 與 35 元，先詢問客人紅茶與咖啡各要買多少瓶，當紅茶與咖啡購買的總數超過 20 瓶時，則販售總價錢可以打 85 折。

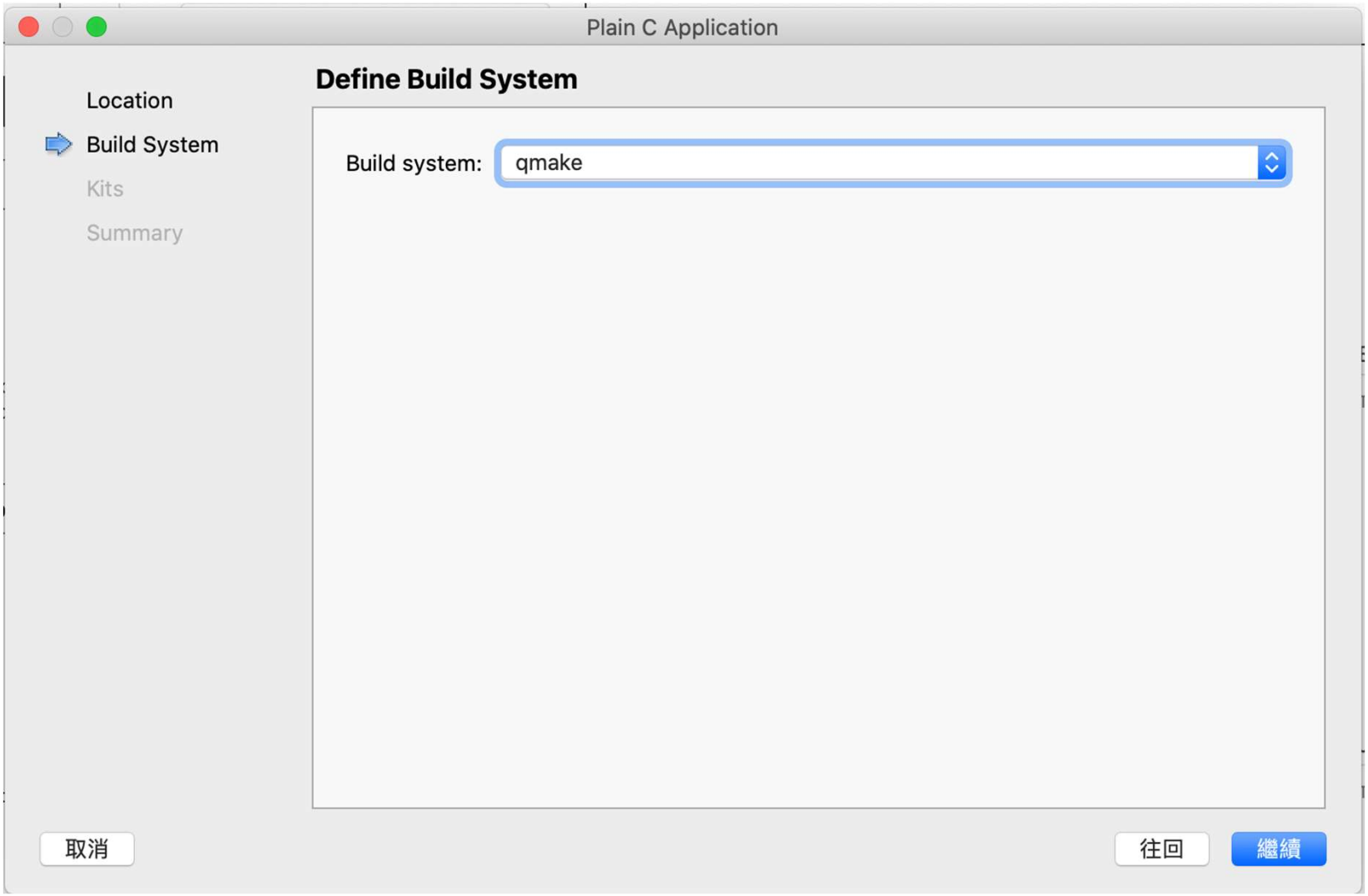
- 思考
  - 輸入
    - 紅茶幾瓶，咖啡幾瓶
  - 輸出
    - 價錢
  - 條件
    - 總數超過20瓶，可以打85折
    - 需要多一個變數!!

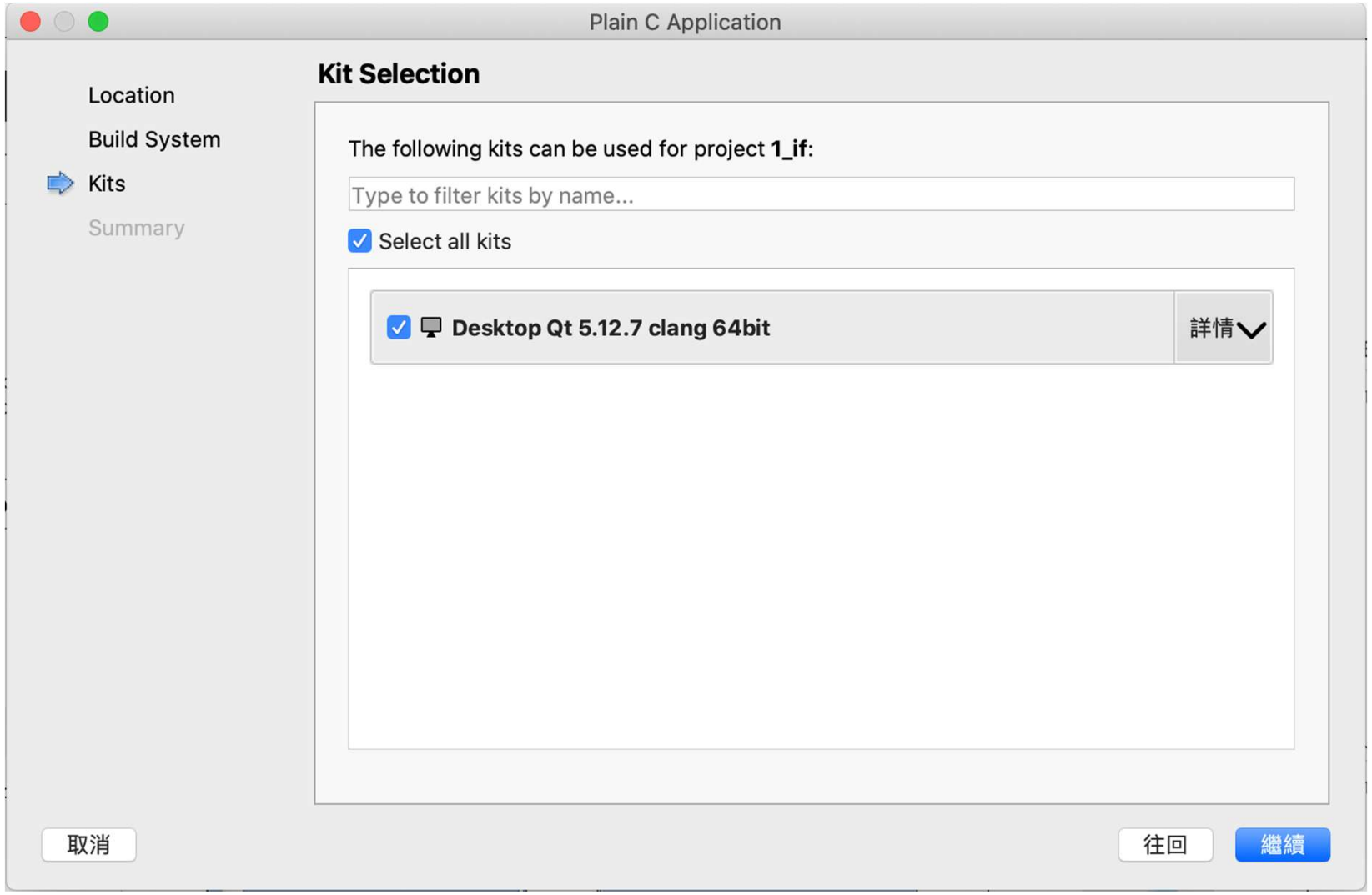


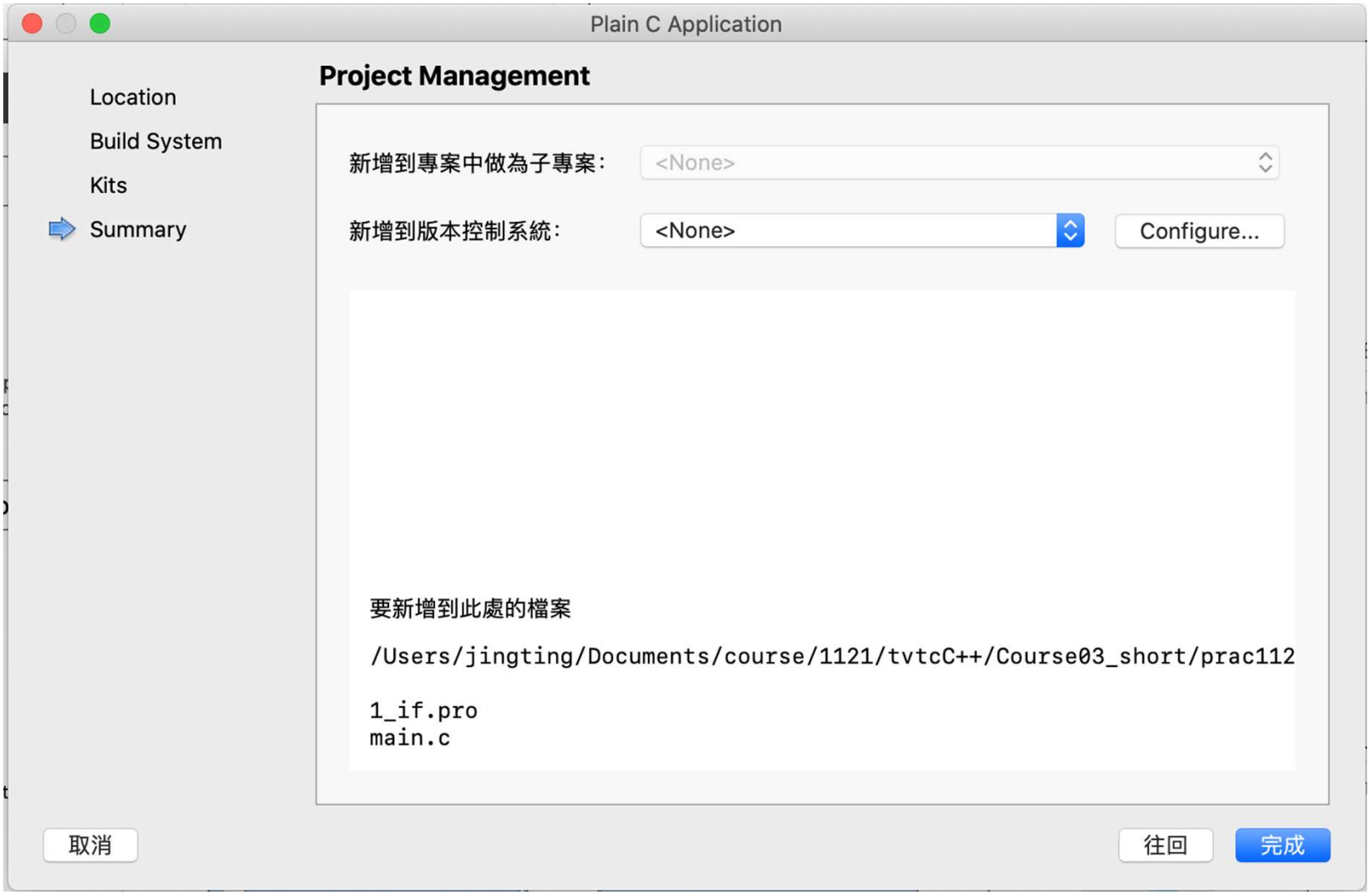

Qt Creator
New File or Project
選擇一個樣本:
所有樣本
專案
Application
Library
其他專案
Non-Qt Project
Import Project
檔案與類別
C++
Modeling
Qt
GLSL
General
Java
Python
Plain C Application
Plain C++ Application
Creates a simple C application with no dependencies.
支援的平台: 桌面
取消
Choose...
Projects
範例
教學
New to Q
Learn how to
own applicati
explore Qt Cr
Get Start
Qt Accou
Online Co
Blogs
User Guide
歡迎
編輯
設計
Debug
專案
說明
Server name:
fortune
The server is running.
Code Sample
Save Game Example
Would you like to take a quick UI tour? This tour highlights important user interface elements and shows how they are used. To take the tour later, select Help > UI Tour.
Take UI Tour
Do Not Show Again
Type to locate (⌘K)
1 問題
2 Search Results
3 應用程式輸出
4 編譯輸出
5 QML Debugger Console
6 一般訊息
8 Test Results

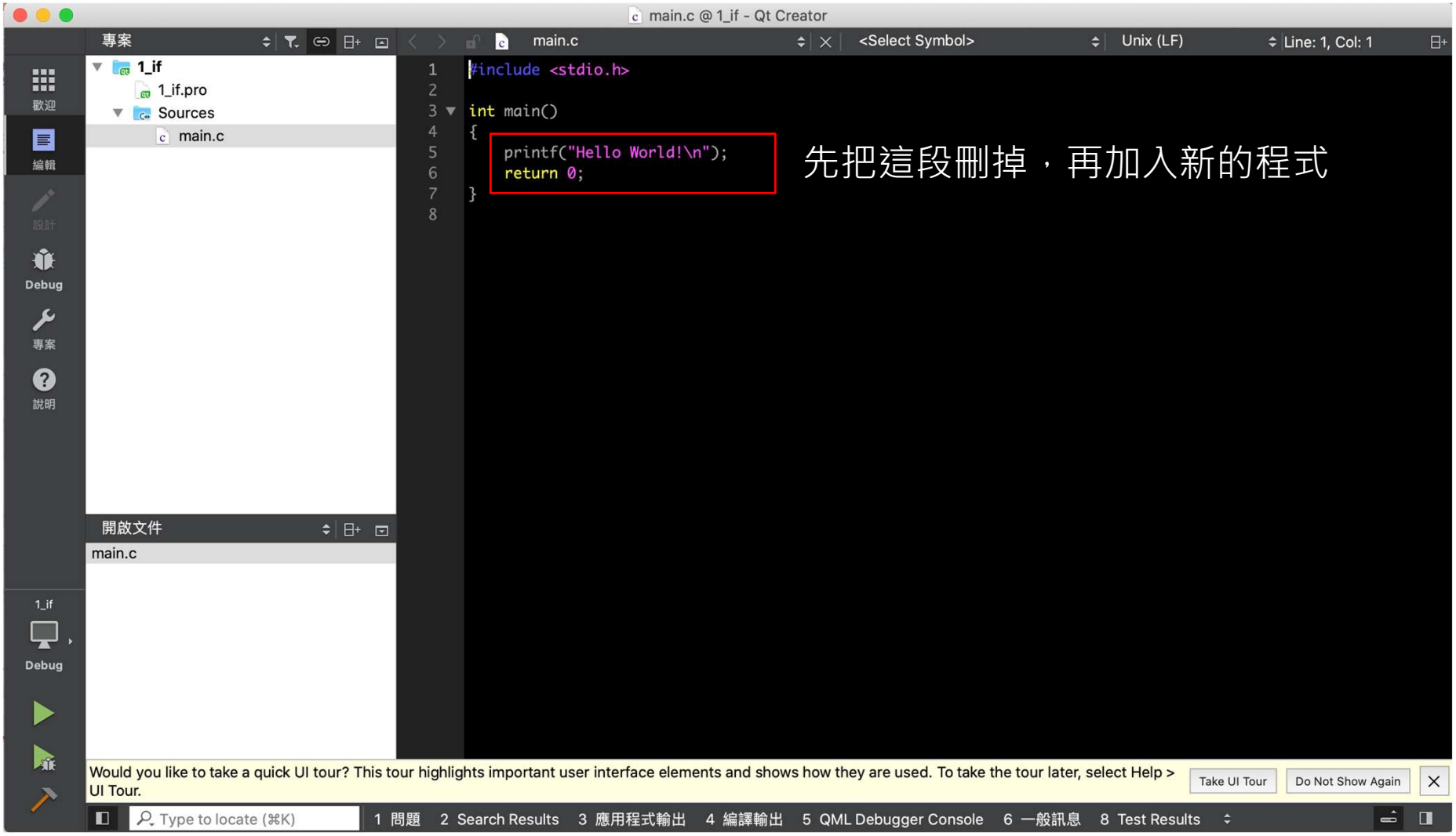


/Course/101001001/c/Course03_Logic_Selective


Plain C Application
Location
Build System
Kits
Summary
Define Build System
Build system:
qmake
取消
往回
繼續


Plain C Application
Location
Build System
Kits
Summary
Kit Selection
The following kits can be used for project 1_if:
Type to filter kits by name...
Select all kits
Desktop Qt 5.12.7 clang 64bit
詳情
取消
往回
繼續


Plain C Application
Location
Build System
Kits
Summary
Project Management
新增到專案中做為子專案:
<None>
新增到版本控制系統:
<None>
Configure...
要新增到此處的檔案
/Users/jingting/Documents/course/1121/tvtcC++/Course03_short/prac112
1_if.pro
main.c
取消
往回
完成


main.c @ 1_if - Qt Creator
專案
main.c
<Select Symbol>
Unix (LF)
Line: 1, Col: 1
歡迎
編輯
設計
Debug
專案
說明
1_if
1_if.pro
Sources
main.c
開啟文件
main.c
1_if
Debug
#include <stdio.h>
int main()
{
printf("Hello World!\n");
return 0;
}
先把這段刪掉，再加入新的程式
Would you like to take a quick UI tour? This tour highlights important user interface elements and shows how they are used. To take the tour later, select Help > UI Tour.
Take UI Tour
Do Not Show Again
Type to locate (⌘K)
1 問題
2 Search Results
3 應用程式輸出
4 編譯輸出
5 QML Debugger Console
6 一般訊息
8 Test Results

```c
#include <stdio.h>

int main(int argc, char *argv[])
{
    int NumOfTea = 0;
    int NumOfCoffee = 0;
    int amount, totalPrice;
    printf("Number of Tea: ");
    scanf("%d",&NumOfTea);
    printf("Number of Coffee: ");
    scanf("%d",&NumOfCoffee);
    totalPrice = NumOfTea*25 + NumOfCoffee*35;
    amount = NumOfTea + NumOfCoffee;
    if(amount>20)
    {
        totalPrice *= 0.85;
    }
    printf(" Total Price: %d \n",totalPrice);
    return 0;
}
```

宣告變數

輸入資訊

計算

判斷及處理

輸出結果

進一步思考

- 多一個且打折條件
  - 除了購買總數超過20瓶外，還必須購買總價超過 (含) 650 元，才享有打 85 折的優惠

```
if( iCount > 20 && iTotal >= 650 ) iTotal = iTotal * 0.85;
```

- 多一個或打折條件
  - 只要購買總數超過20瓶，或是購買總價超過(含) 600 元，就享有打 85 折的優惠

```
if( iCount > 20 || iTotal >= 600 ) iTotal = iTotal * 0.85;
```

邏輯問題

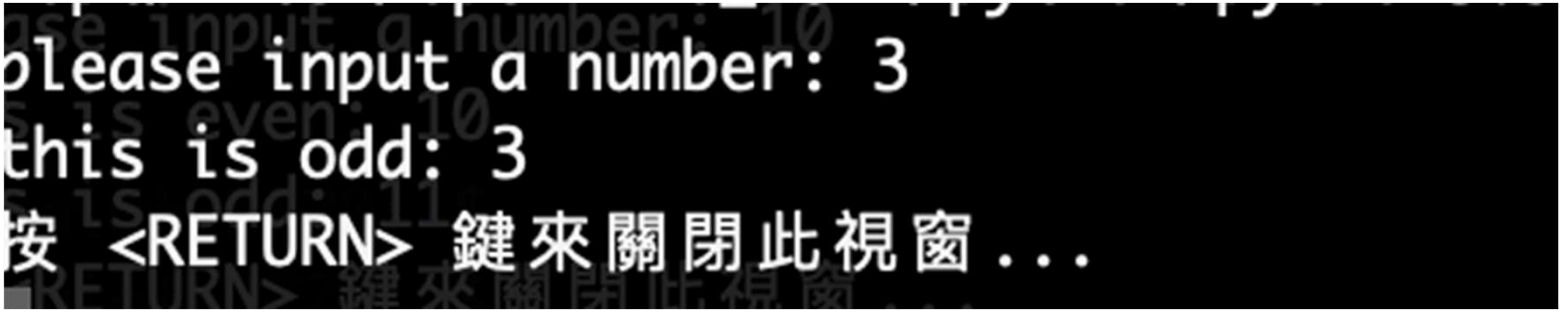
- 試寫一個程式，讓使用者輸入一個整數之後，判斷其為偶數就加1，奇數就乘以2，並印再畫面上

- 思考
  - 輸入
    - 任一個整數
  - 輸出
    - 「奇數」或「偶數」
  - 判斷
    - 除以二的餘數為零，就是偶數，加1
    - 除以二的餘數為一，就是奇數，乘以2

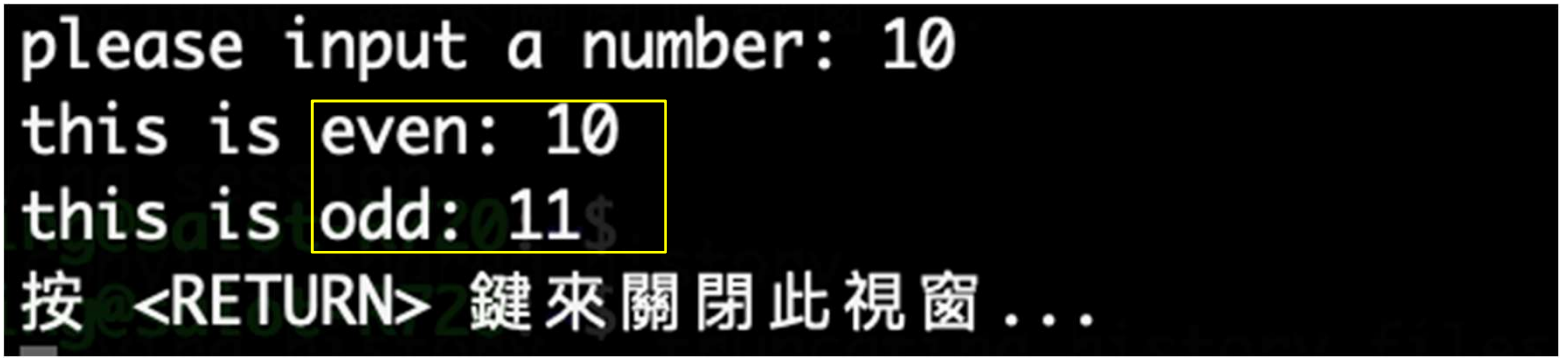
```c
#include <stdio.h>

int main()
{
    int number;
    printf("please input a number: ");
    scanf("%d",&number);
    if(number%2==0)
    {
        printf("this is even: %d\n",number);
        number += 1;
    }
    if(number%2==1)
    {
        printf("this is odd: %d\n",number);
        number *=2 ;
    }
}

```

/Course/101001001/c/Course03_Logic_Selective

輸入3—奇數



輸入10—偶數



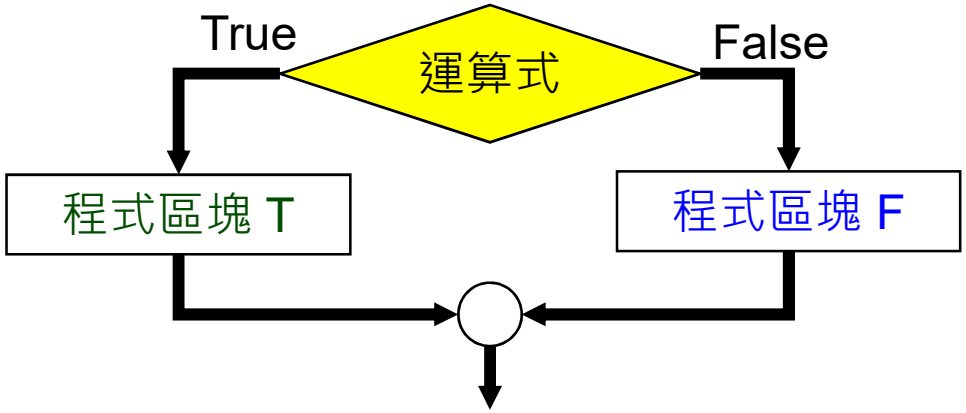
if...else 敘述

- 計算奇數的程式
  - 用了兩行 if 敘述
    - if(number%2==0) …
    - if(number%2==1)…

- 過多的判斷式使程式不易讀且沒有效率!!
  - 一個數不是奇數就是偶數，所以當第一次執行 ix%2 時，就可以判斷 ix 為奇數或是偶數
  - 更有效率的 if 敘述 – if...else 敘述

- 語法：

```
if( 運算式 )
{
    程式區塊T
}
else
{
    程式區塊F
}
```



- 規則
  - 運算式的結果為真 (true) 時：執行「程式區塊T」中的程式碼
  - 運算式的結果為假 (false) 時：執行「程式區塊F」中的程式碼
  - 程式區塊中敘述只要超過一行以上
    - 一定要使用大括弧 { }
    - 如果只有一行敘述，則可以省略大括弧
  - 運算式為 false，而且不需執行任何的敘述時
    - 可省略包含 else 之後的所有部分
    - 此時就是 if 敘述的基本型 **if(運算式) {程式區塊}** 。

程式練習

- 試寫一個程式，讓使用者輸入一個整數之後，判斷其為偶數就加1，奇數就乘以2，並印再畫面上

- 思考
  - 輸入
    - 任一個整數
  - 輸出
    - 「奇數」或「偶數」
  - 判斷
    - 除以二的餘數為零，就是偶數，加1
    - 除以二的餘數為一，就是奇數，乘以2

```c
#include <stdio.h>

int main()
{
    int number;
    printf("please input a number: ");
    scanf("%d",&number);
    if(number%2==0)
    {
        printf("this is even: %d\n",number);
        number += 1;
    }
    else
    {
        printf("this is odd: %d\n",number);
        number *=2 ;
    }
}

```

/Course/101001001/c/Course03_Logic_Selective

```
please input a number: 10
this is even: 10
按 <RETURN> 鍵來關閉此視窗...
```

```
please input a number: 3
this is odd: 3
按 <RETURN> 鍵來關閉此視窗...
```

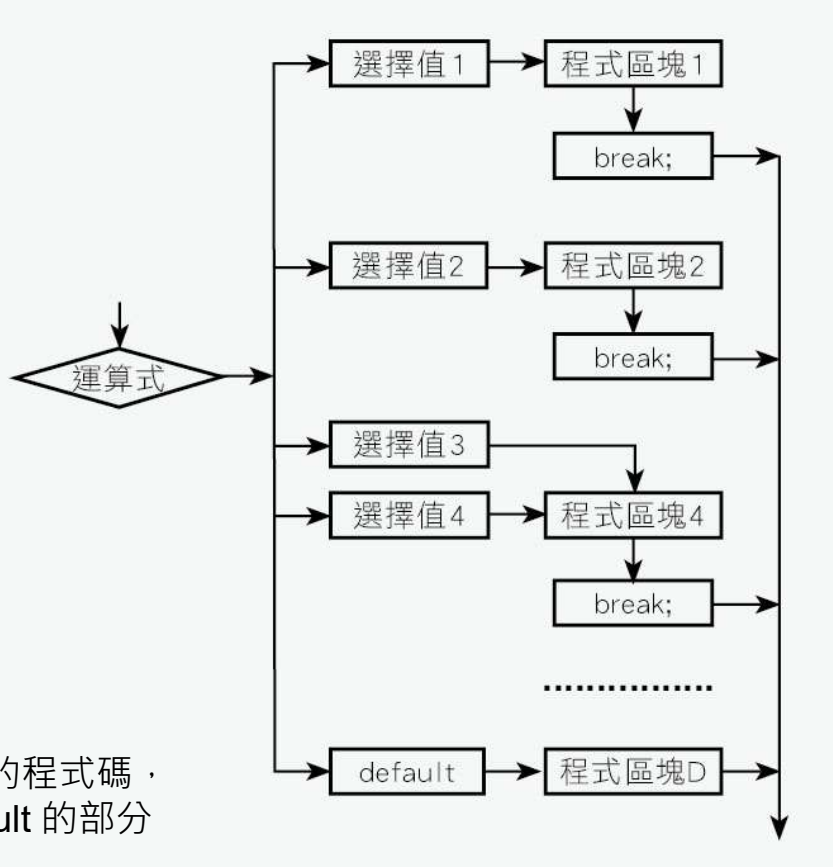
多重選擇 switch-case

- switch-case 語法

```
switch (運算式) {
    case 選擇值1:
         程式區塊1
         break;
    case 選擇值2:
         程式區塊2
         break;
    case 選擇值3:
    case 選擇值4:
         程式區塊4
         break;
    ..................
    default:
         程式區塊D
}
```

如果不需執行任何的程式碼，
可以省略整個 default 的部分

這裡不需要分號



限制：case 後的選擇值：只能是常數或是字元

程式

- 請使用switch-case，輸入1, 2, 3，對應輸出A, B, C，除此之外的數字都顯示input error

```c
#include <stdio.h>

int main(int argc, char *argv[])
{
    int num;
    printf("input a number: ");
    scanf("%d",&num);
    switch(num)
    {
        case 1: printf("A");
                break;
        case 2: printf("B");
                break;
        case 3: printf("C");
                break;
        default:
                printf("input error");
    }

    return 0;
}

```

```
input a num: 1
A
請按任意鍵繼續 . . .
```

```
input a num: 5
input error
請按任意鍵繼續 . . .
```

break

- break 在 switch-case 敘述的功能
  - 讓程式執行流程離開該 case
  - 並脫離 switch-case 敘述的執行範圍

- 將 switch-case 看成一個程式區塊
  - 大括弧所涵蓋的範圍
  - 讓程式的執行流程跳脫目前**大括弧所涵蓋**的程式區塊