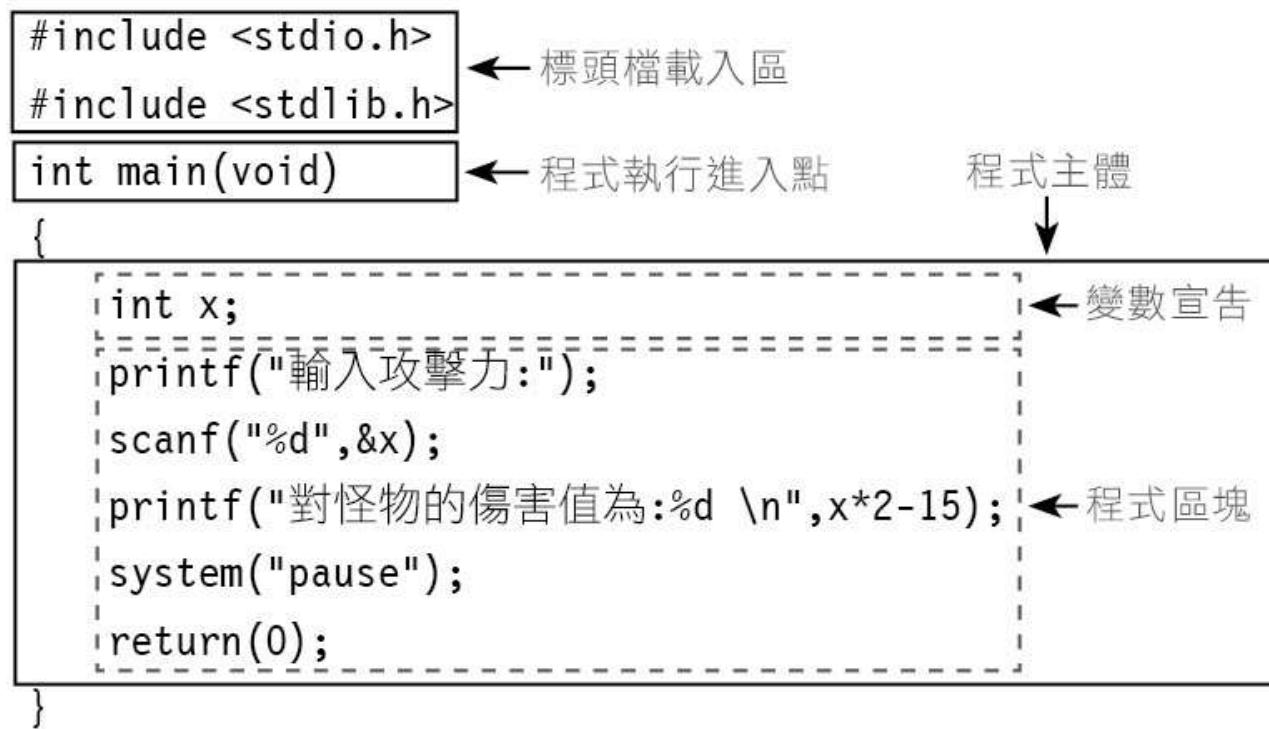


Course 02

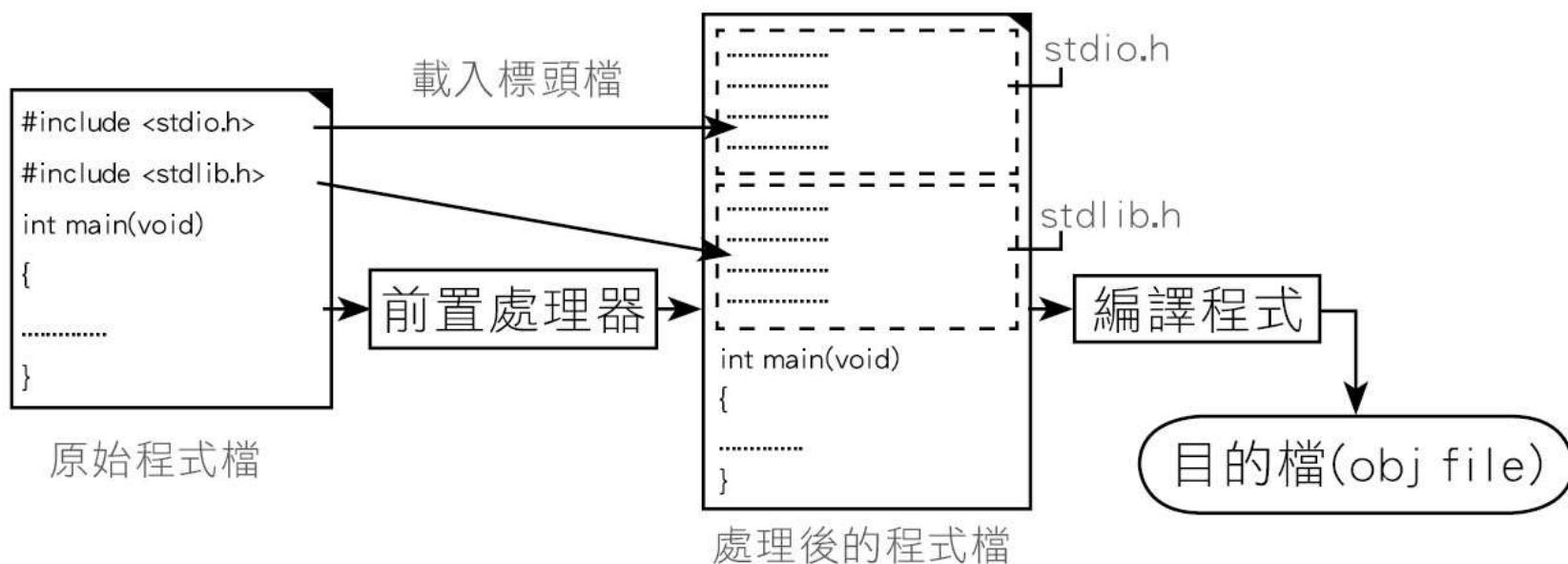
資料型態
輸入與輸出

C 程式的基本架構

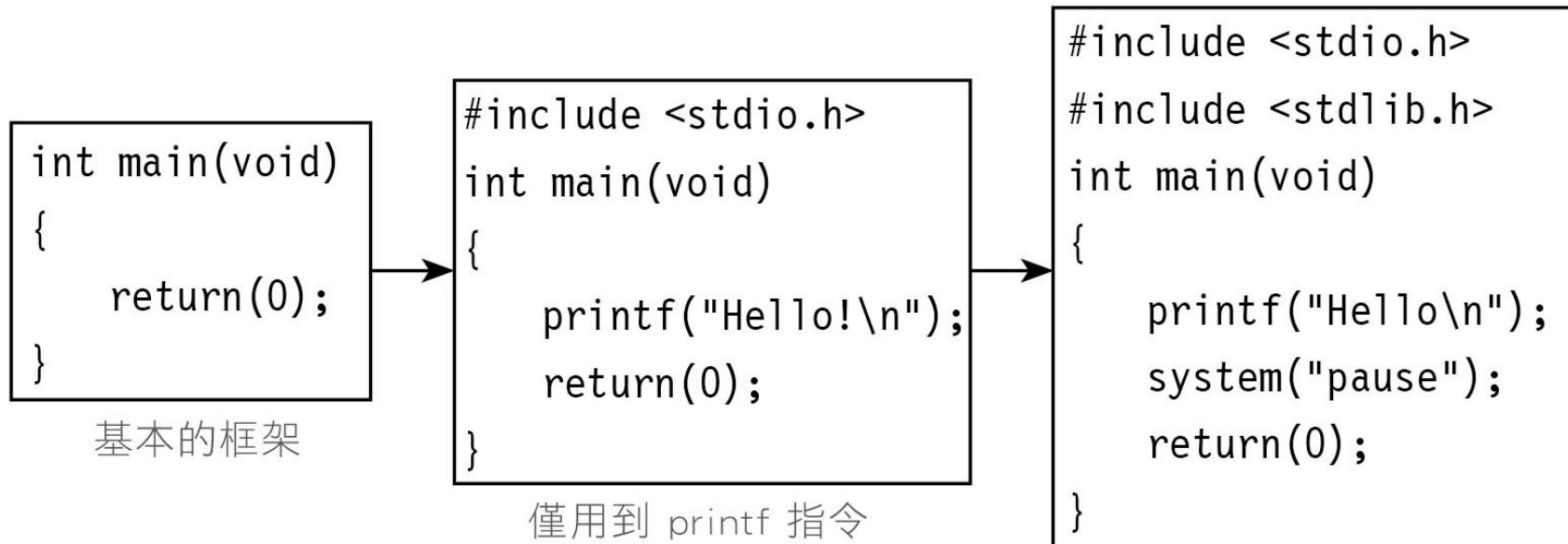
- 基本架構非常的單純，分成三個部分
 - 標頭檔載入區
 - 程式執行進入點(main 函式)
 - 程式主體(main 之後以 {} 所包含的區域)



- 可擴充性 – C 語言的重要特性
 - 程式的功能不夠時，透過載入的方式來擴充
 - 載入的過程是透過編譯器的前置處理器 (preprocessor) 來執行
 - 以「#」為開頭的 #include 就是前置處理器的指令之一



- 極端的問題 – 沒有做任何事情的 C 程式
 - ANSI C 的標準 – `int main(void) { return(0); }`
 - 程式主體增加 `printf("Hello!\n");`
 - 標頭檔載入區就需要 `#include <stdio.h>`
 - 要將程式暫停，增加 `system("pause")`
 - 標頭檔載入區再增加 `#include <stdlib.h>`



- 未來將常用函式定義檔
 - 除了 stdio.h 與 stdlib.h
 - #include <math.h> – 數學計算函式
 - #include <time.h> – 時間處理函式
 - #include <string.h> – 字串處理函式
 - #include <ctype.h> – 字元轉換函式

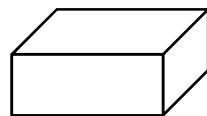
常數與變數

Constant and Variable

變數

- 變數 – 內容可隨時改變的數。四個衍生的問題
 - 變數是怎樣產生的
 - 內容放在哪裡
 - 內容如何被指定
 - 內容如何被改變
- 變數是怎樣產生的

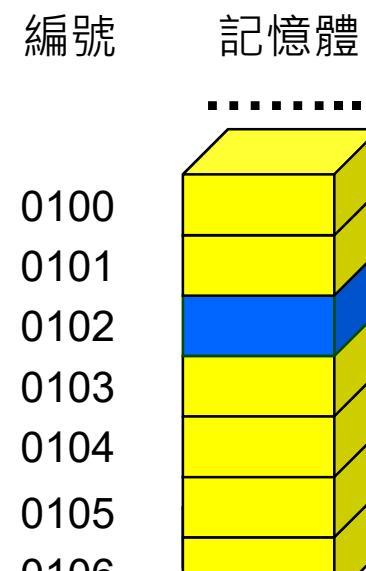
變數：可儲存資料的抽屜



不可能無中生有
就能儲存資料

宣告(declaration)

抽屜必須對應到記憶體
的某一空間：
這個程序稱為**宣告**
**(程式向作業系統申請
一個記憶體)**

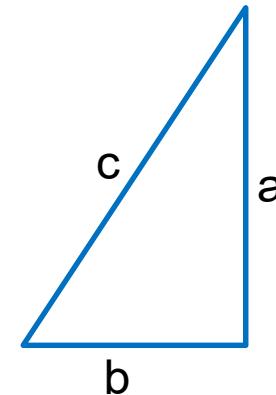


知道直角三角型的任兩邊，
計算第三邊的長度

$$c^2 = a^2 + b^2$$

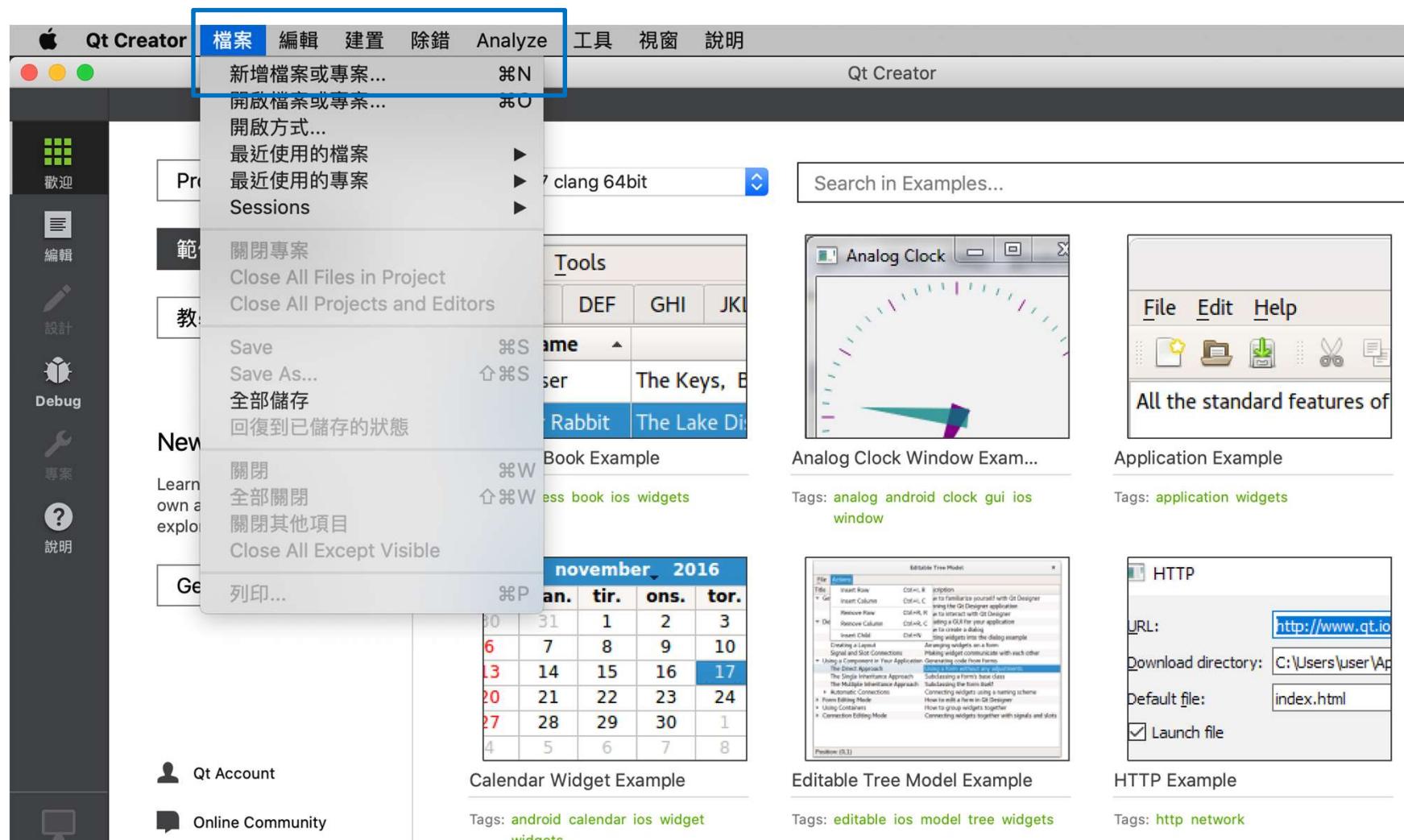
假設 $a=4$, $b=3$, 則

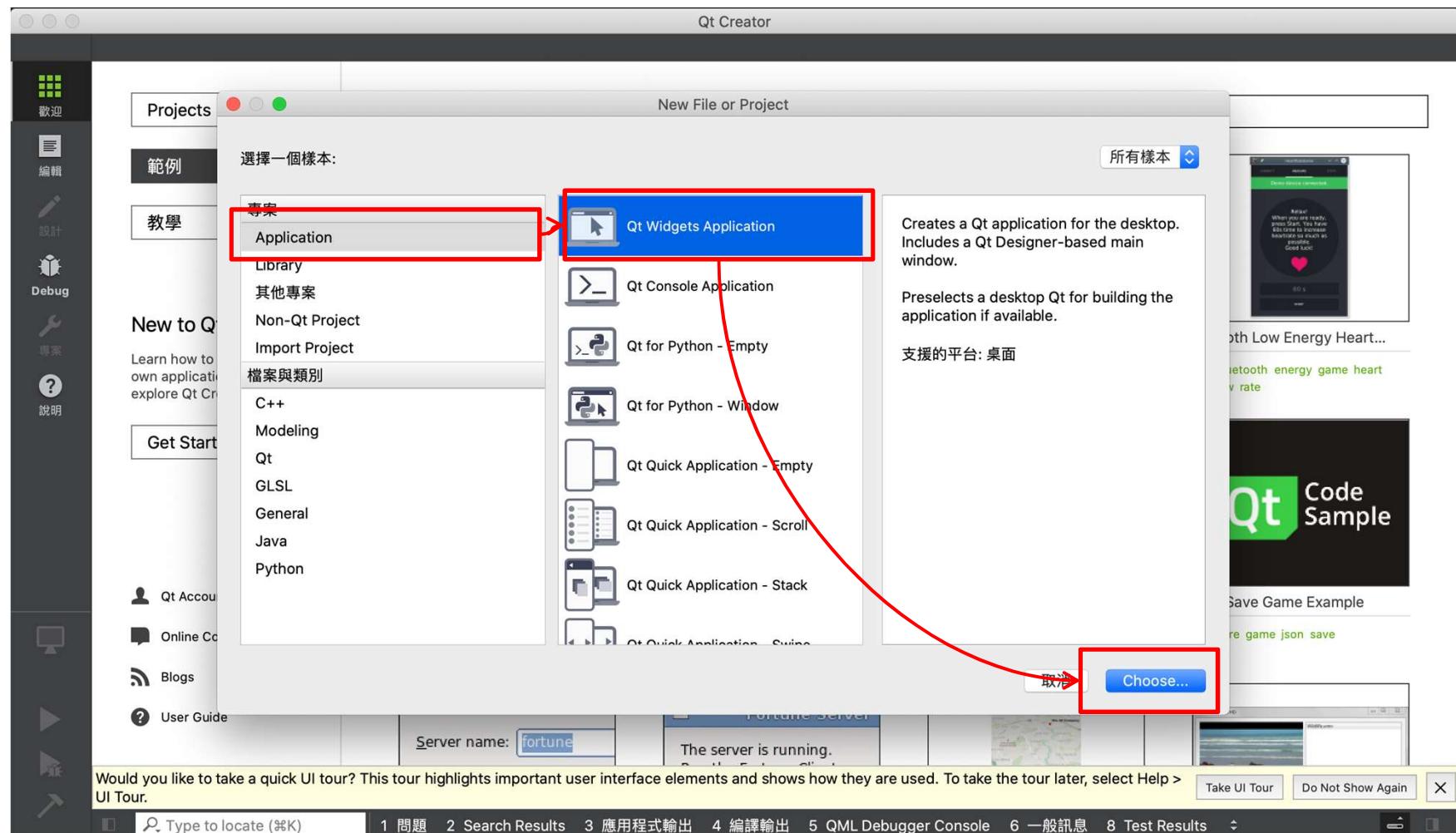
$$c = \sqrt{a^2 + b^2} = \sqrt{3^2 + 4^2} = 5$$



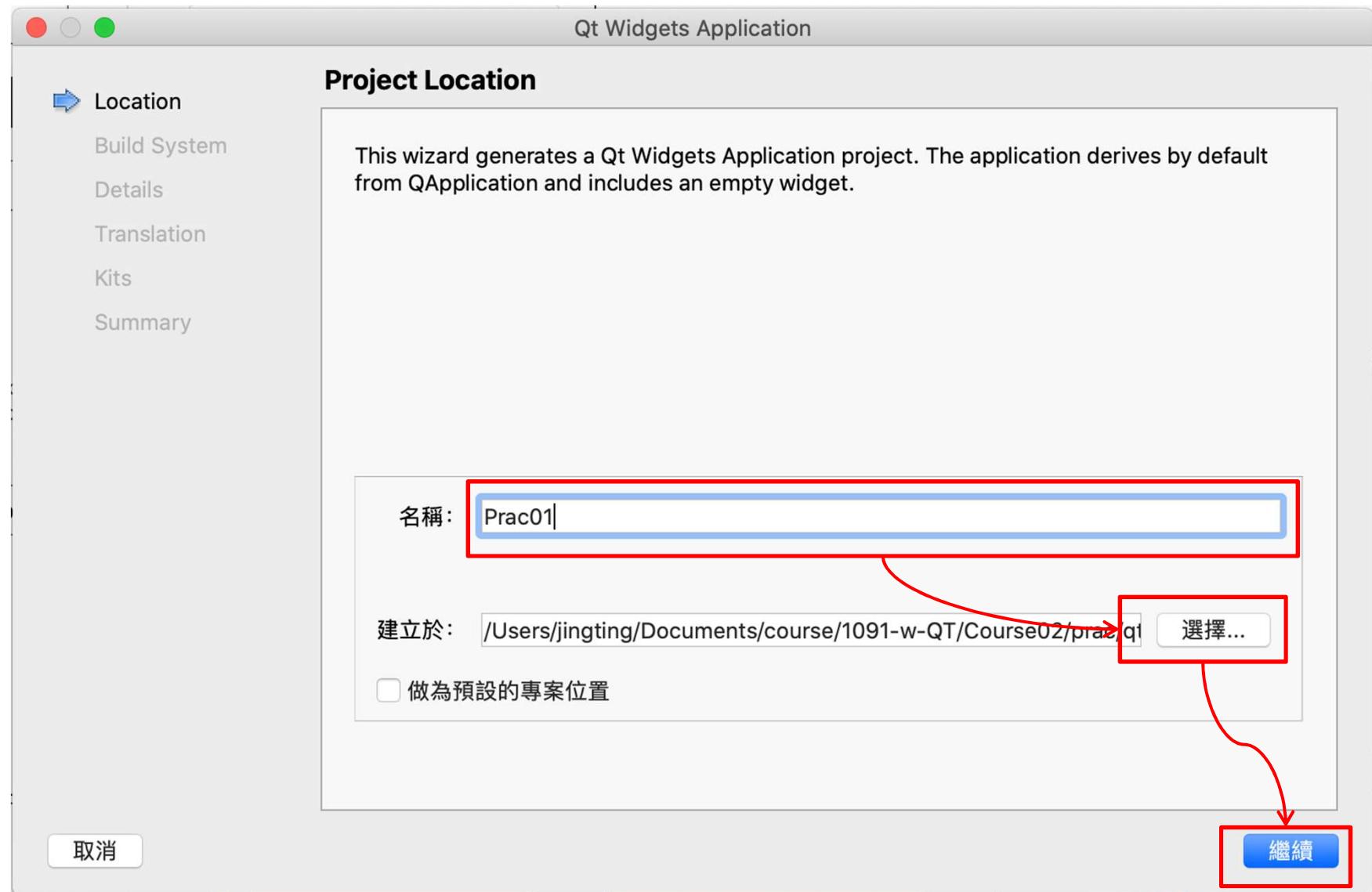
- 變數
 - 輸入 a/b
 - 輸出 c
- GUI
 - 輸入元件: LineEditor x2
 - 輸出元件: Label x1
 - 觸發元件: PushButton

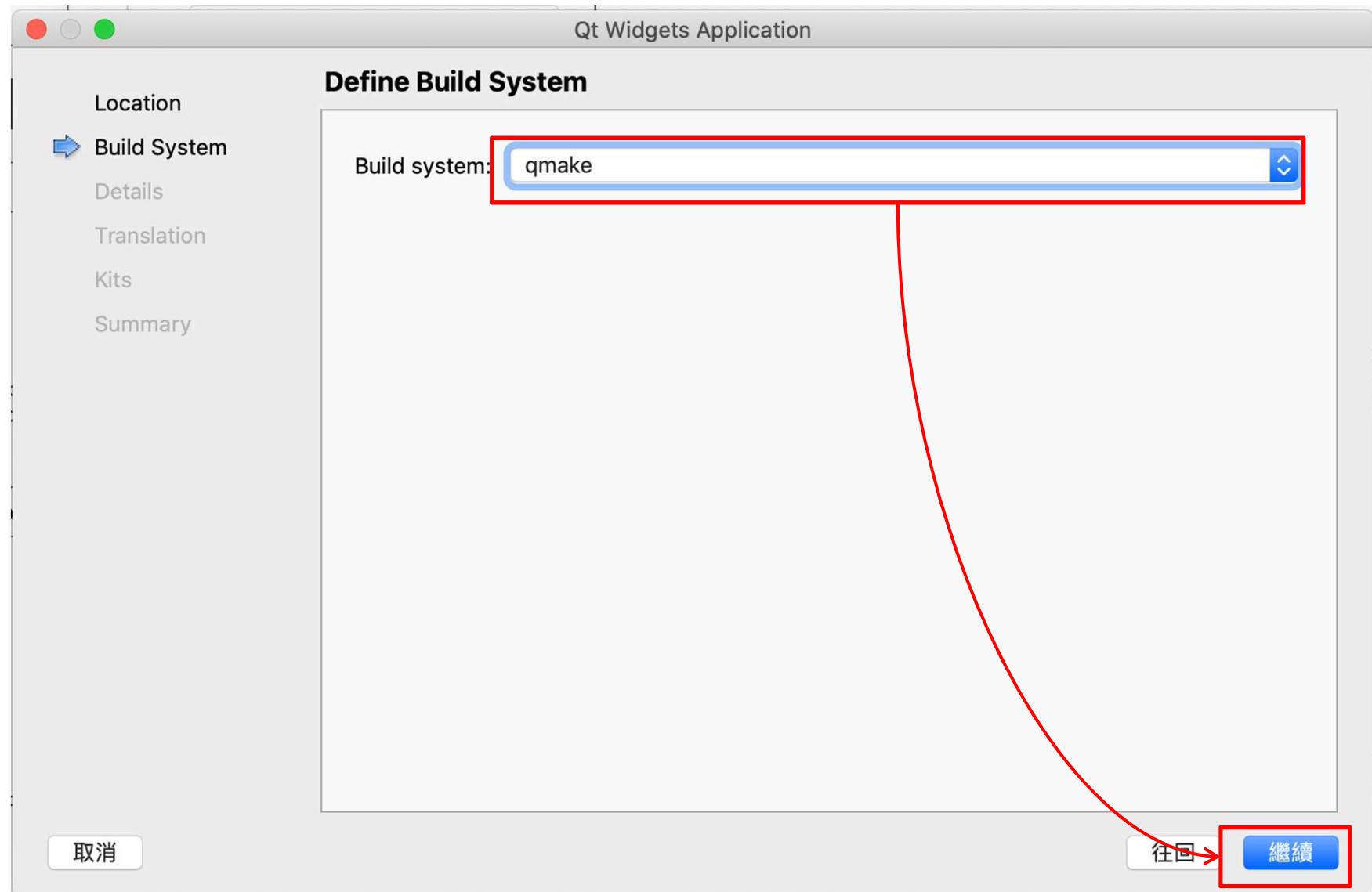
1. 建立專案

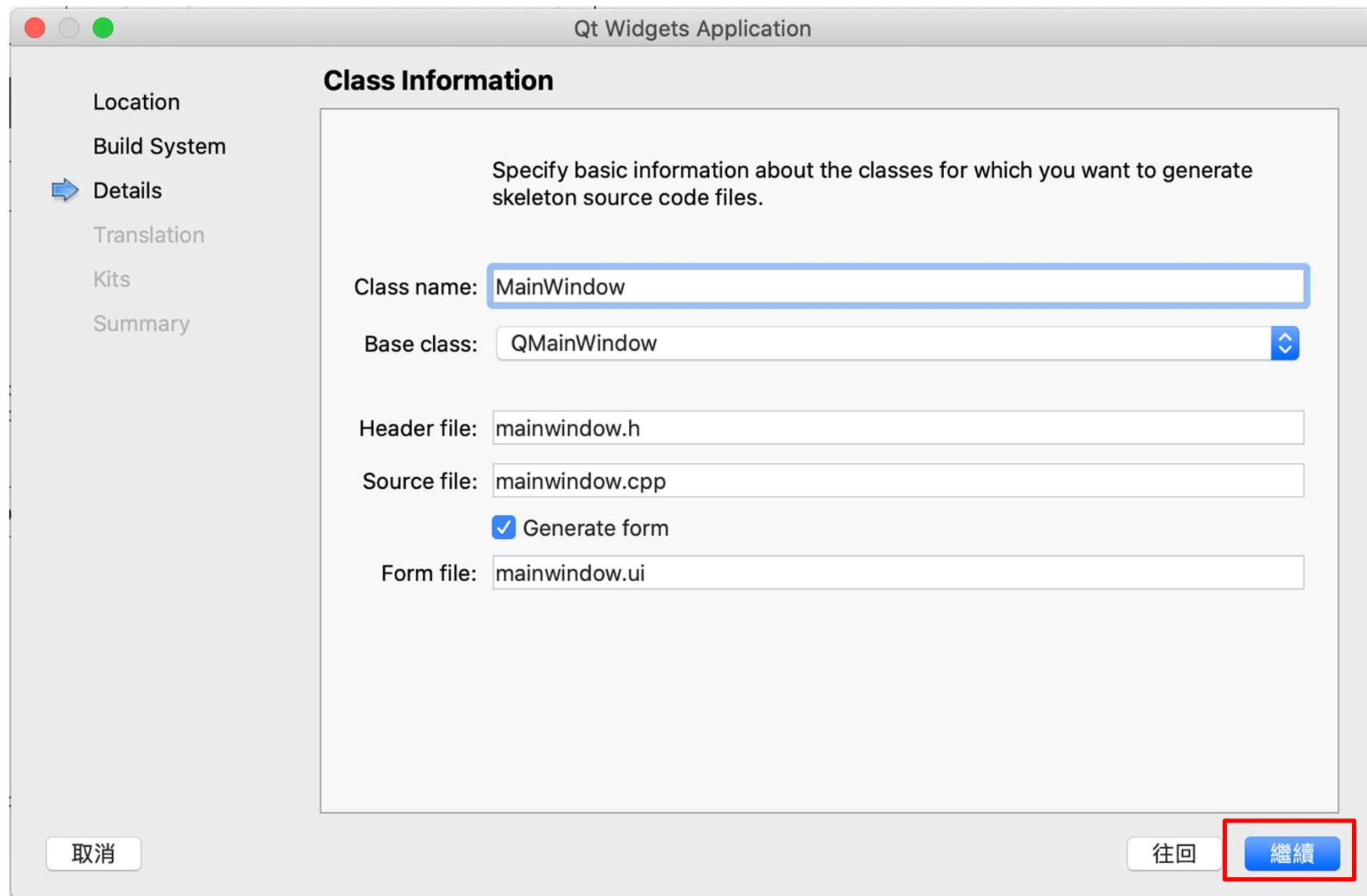


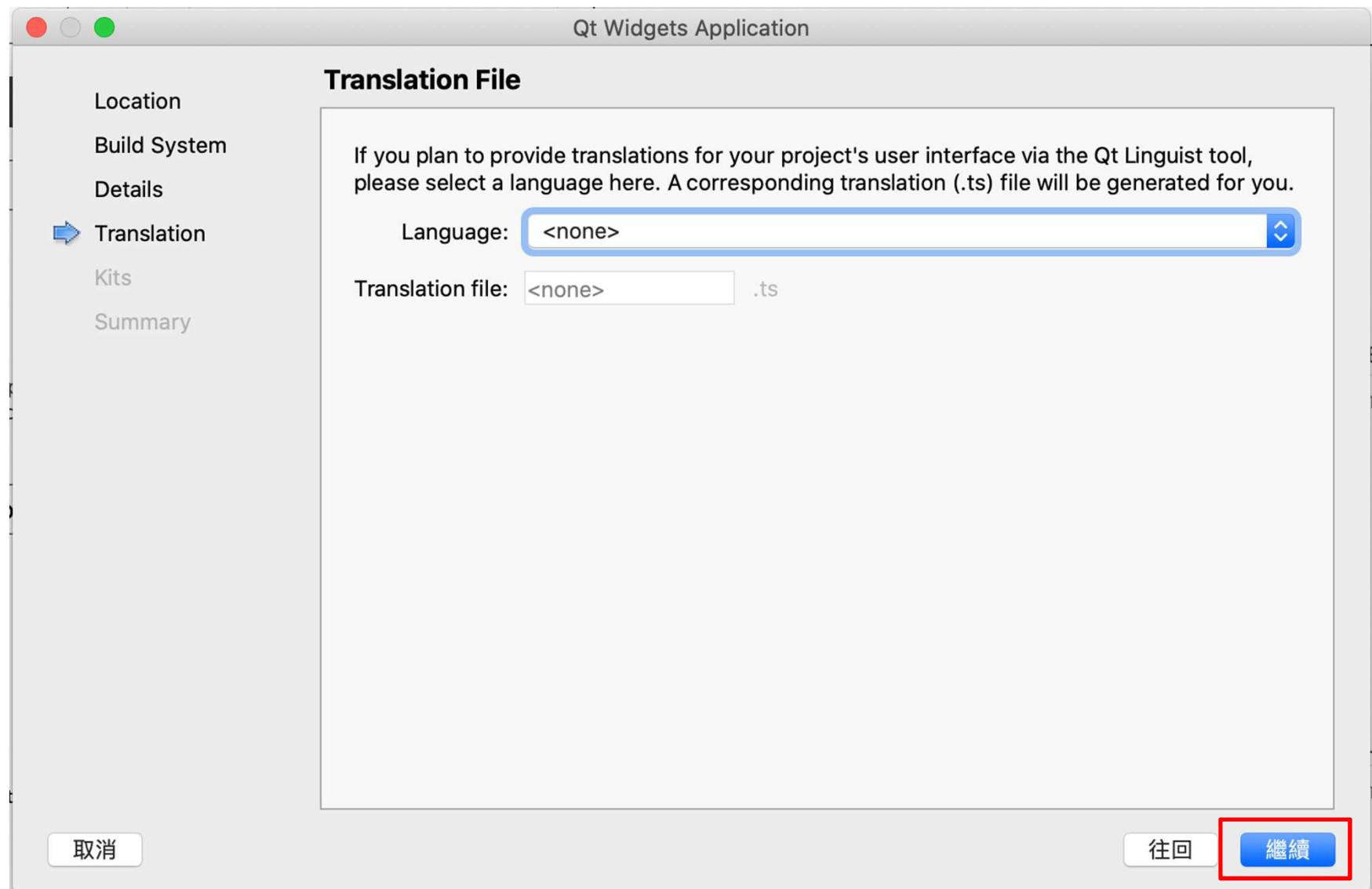


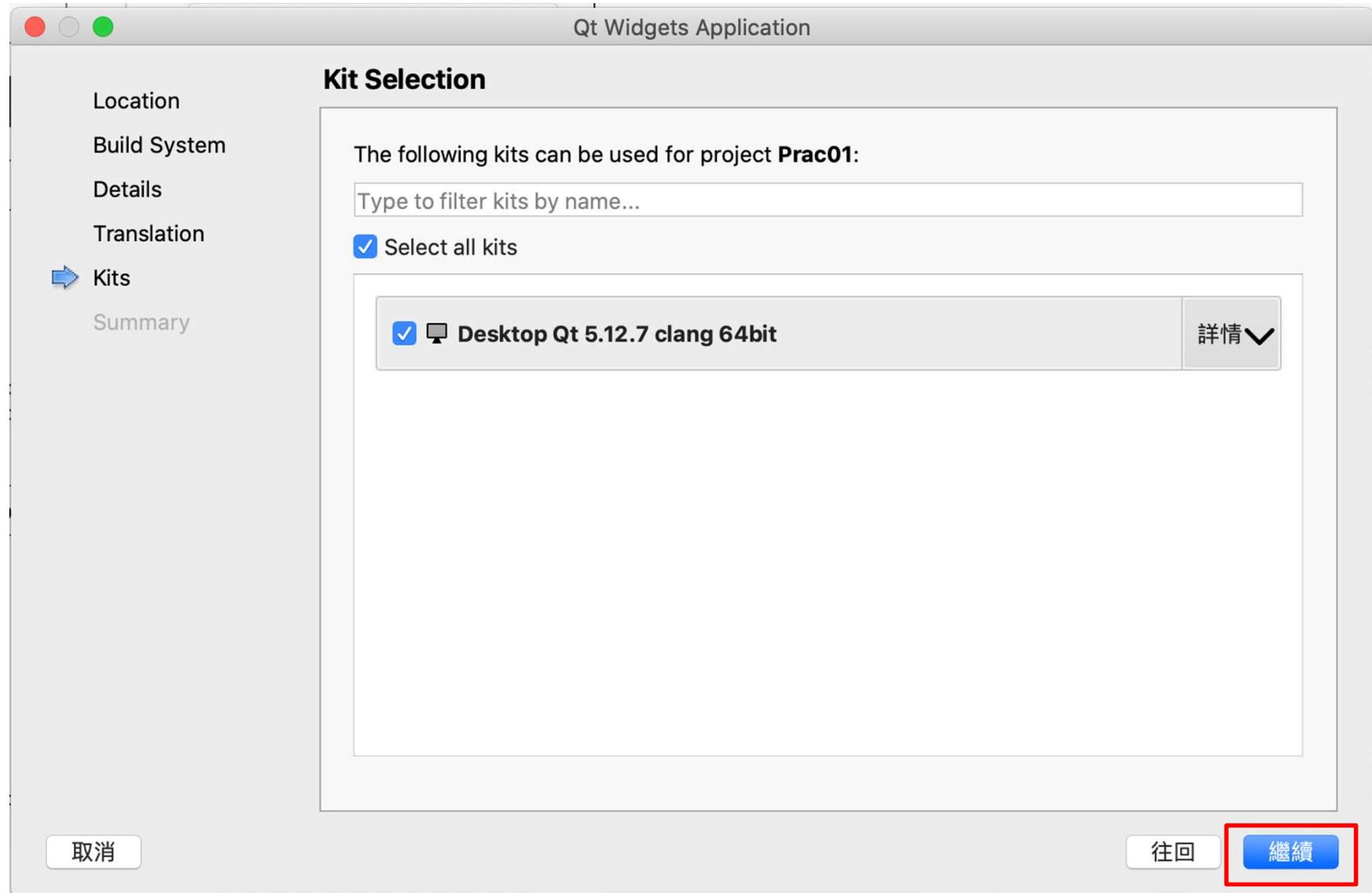
/Course/101001101/c/Cours1e02_DataTypeAndIO

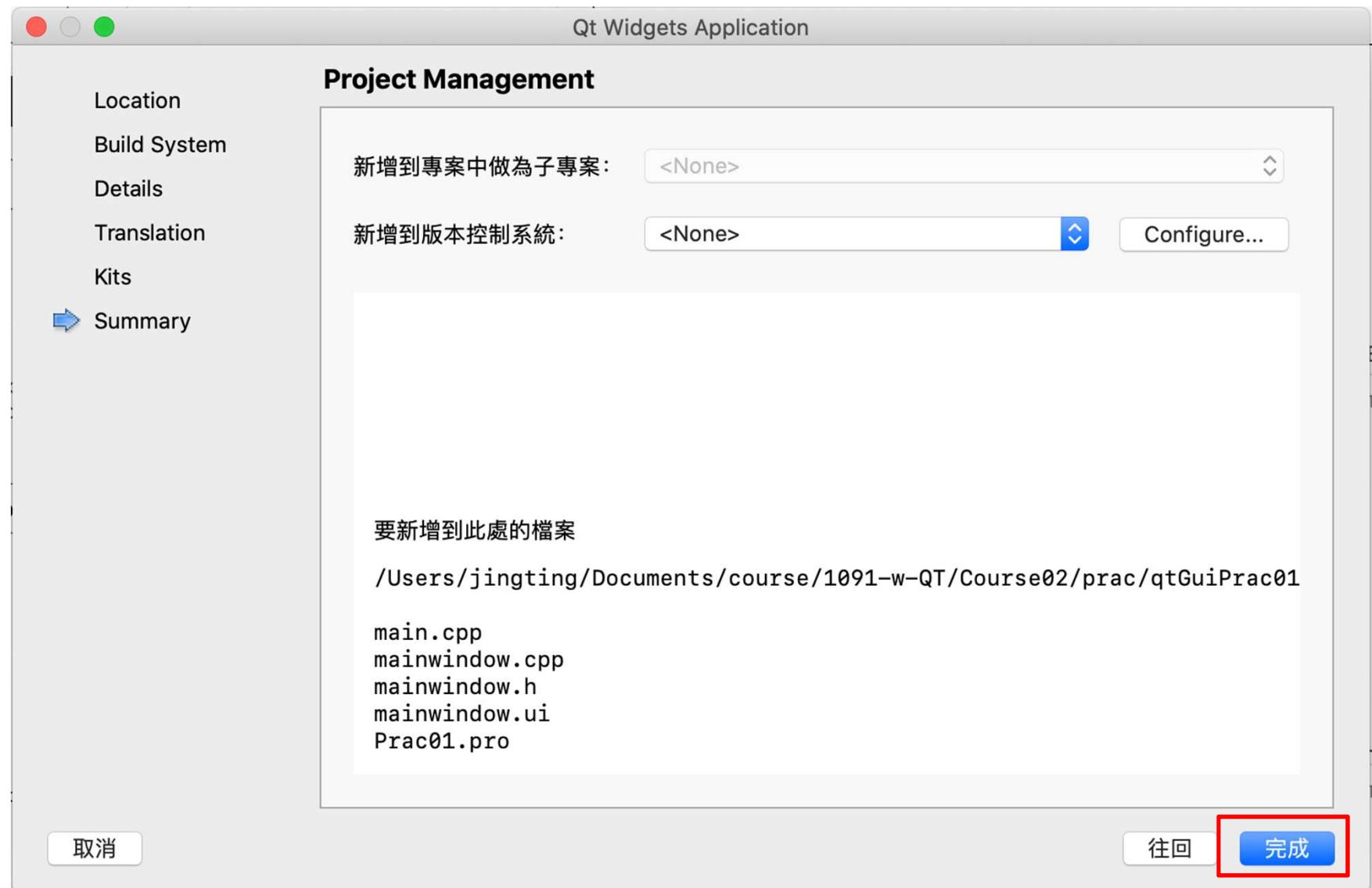




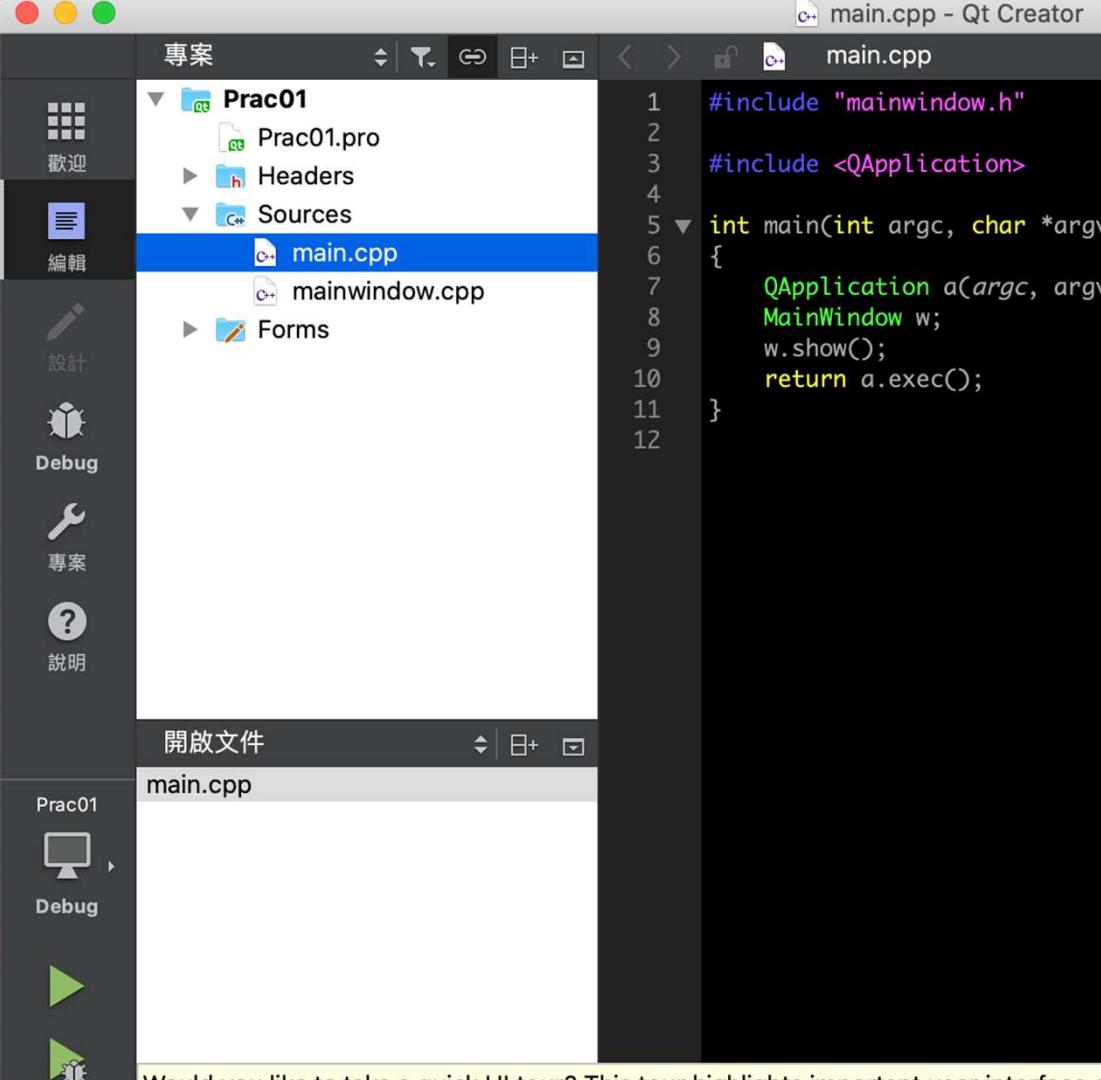






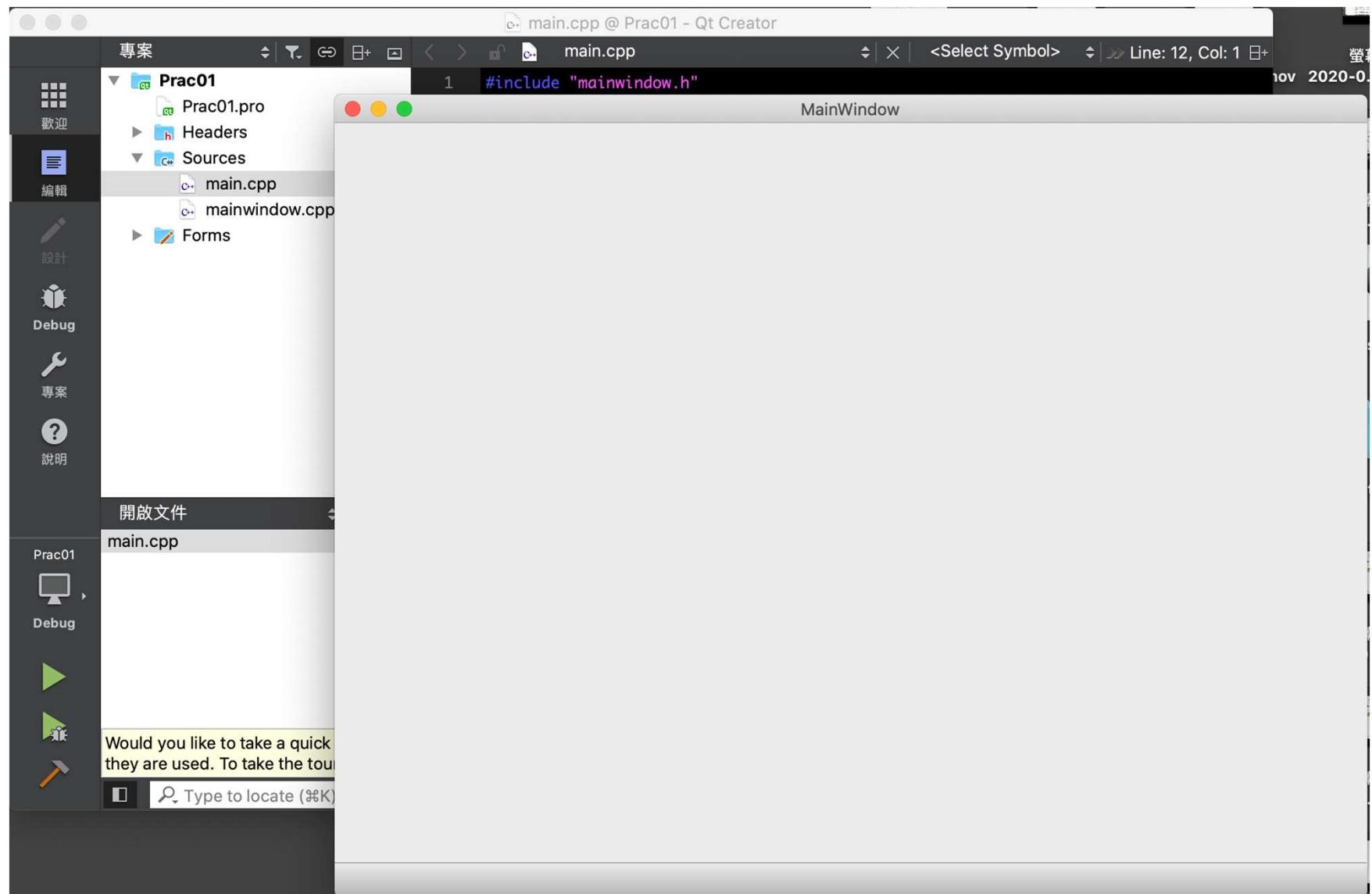


執行看看



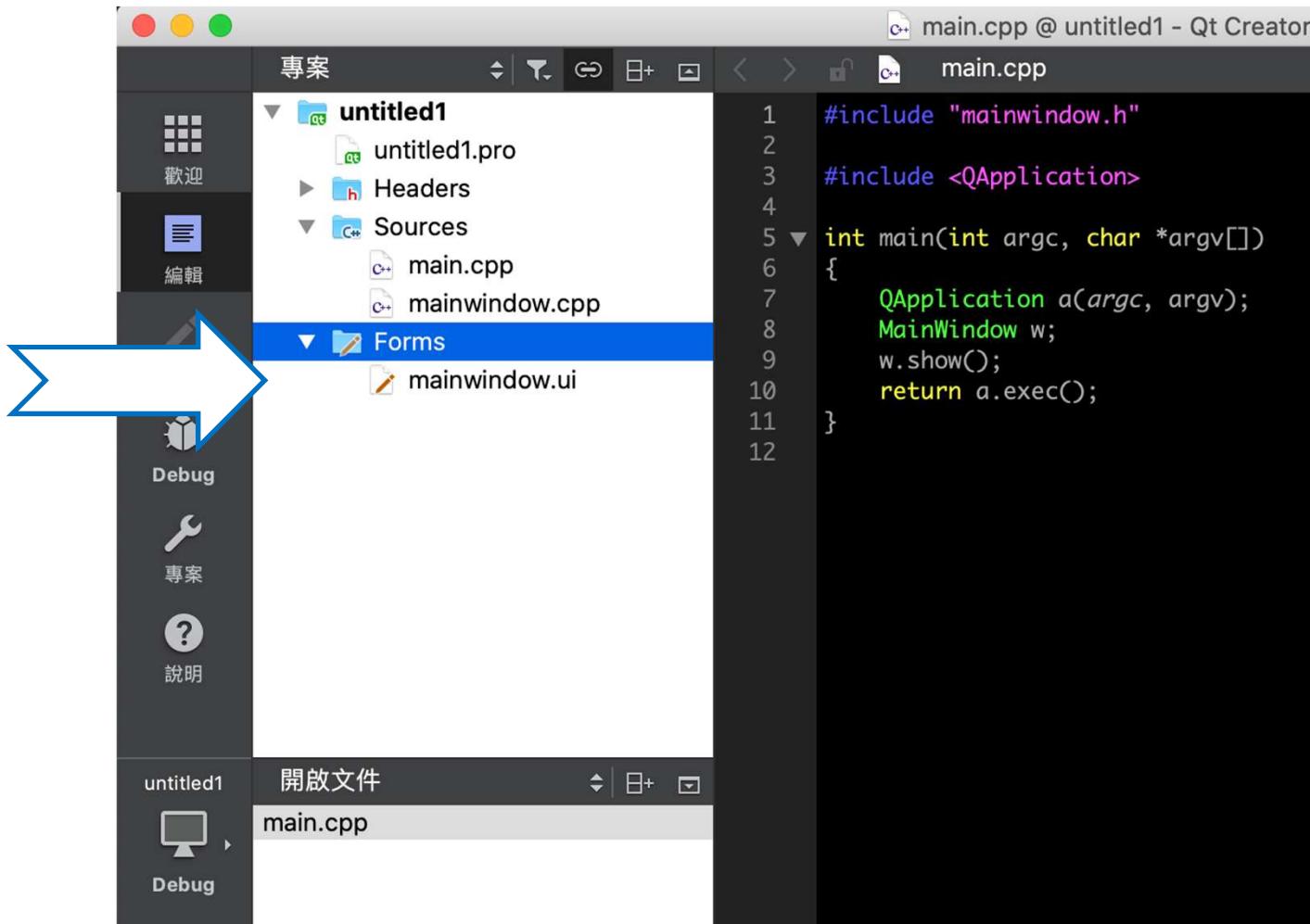
```
#include "mainwindow.h"  
  
#include <QApplication>  
  
int main(int argc, char *argv)  
{  
    QApplication a(argc, argv);  
    MainWindow w;  
    w.show();  
    return a.exec();  
}
```

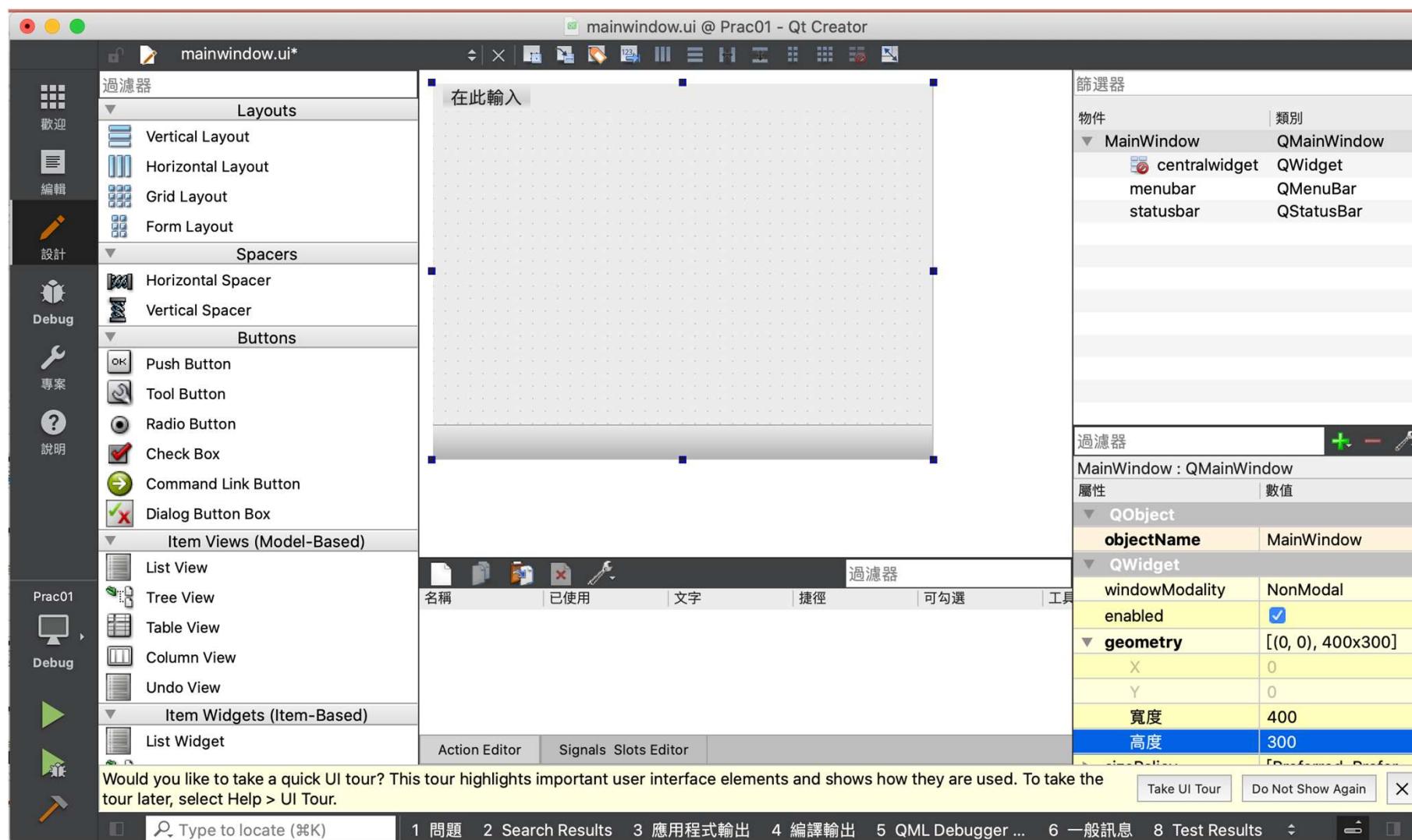
Would you like to take a quick UI tour? This tour highlights important user interface elements they are used. To take the tour later, select Help > UI Tour.

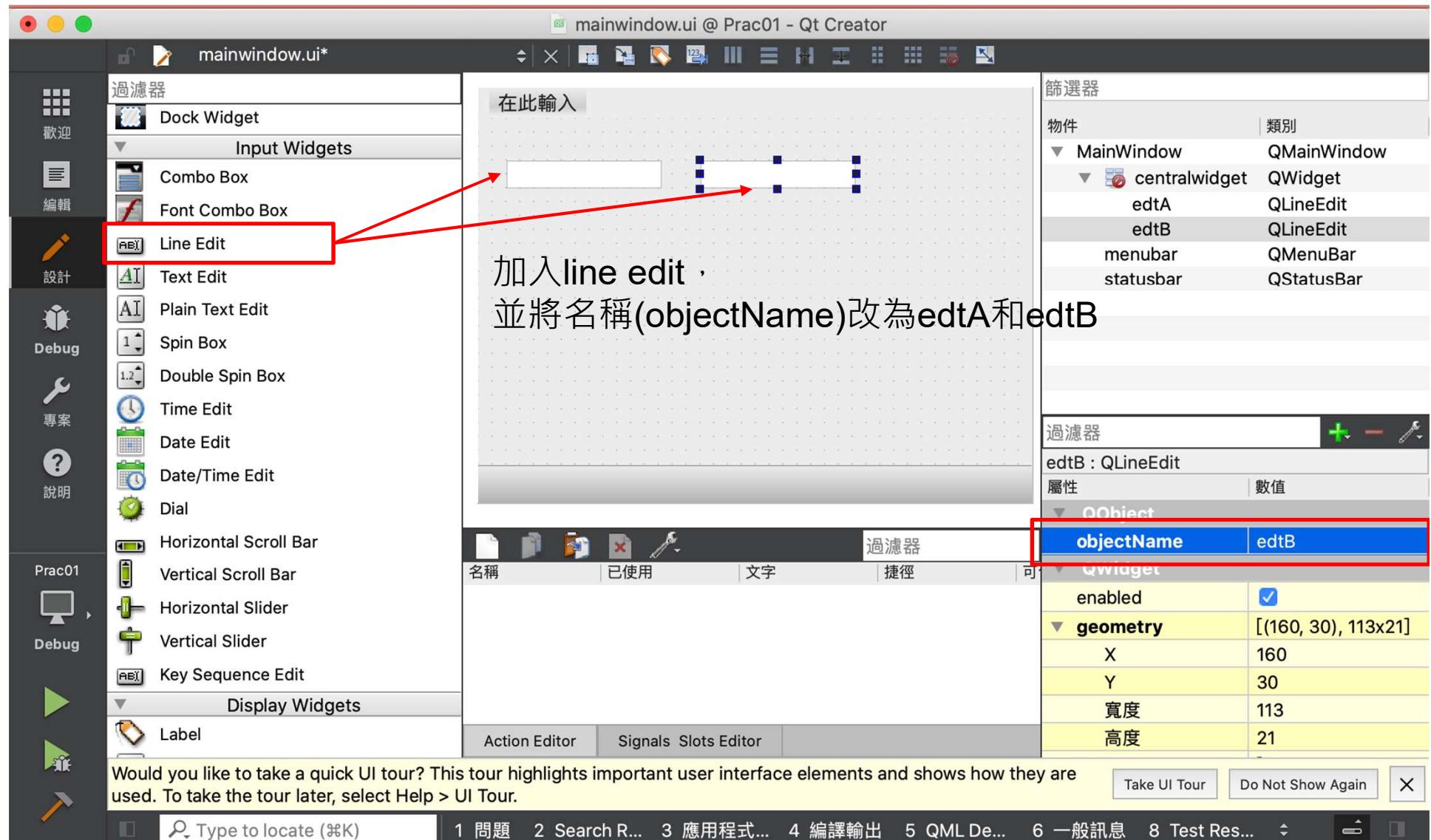


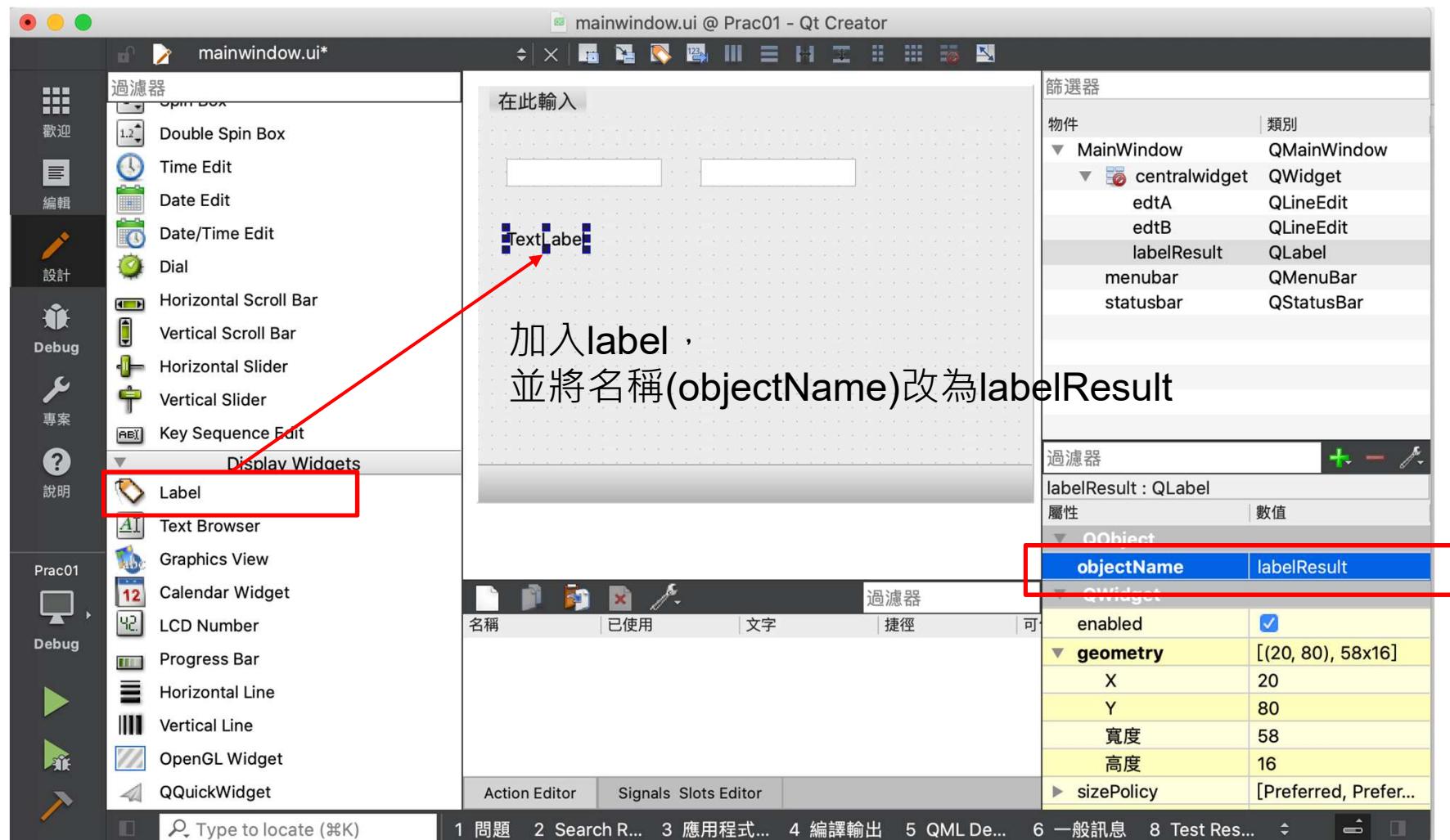
/Course/101001101/c/Cours1e02_DataTypeAndIO

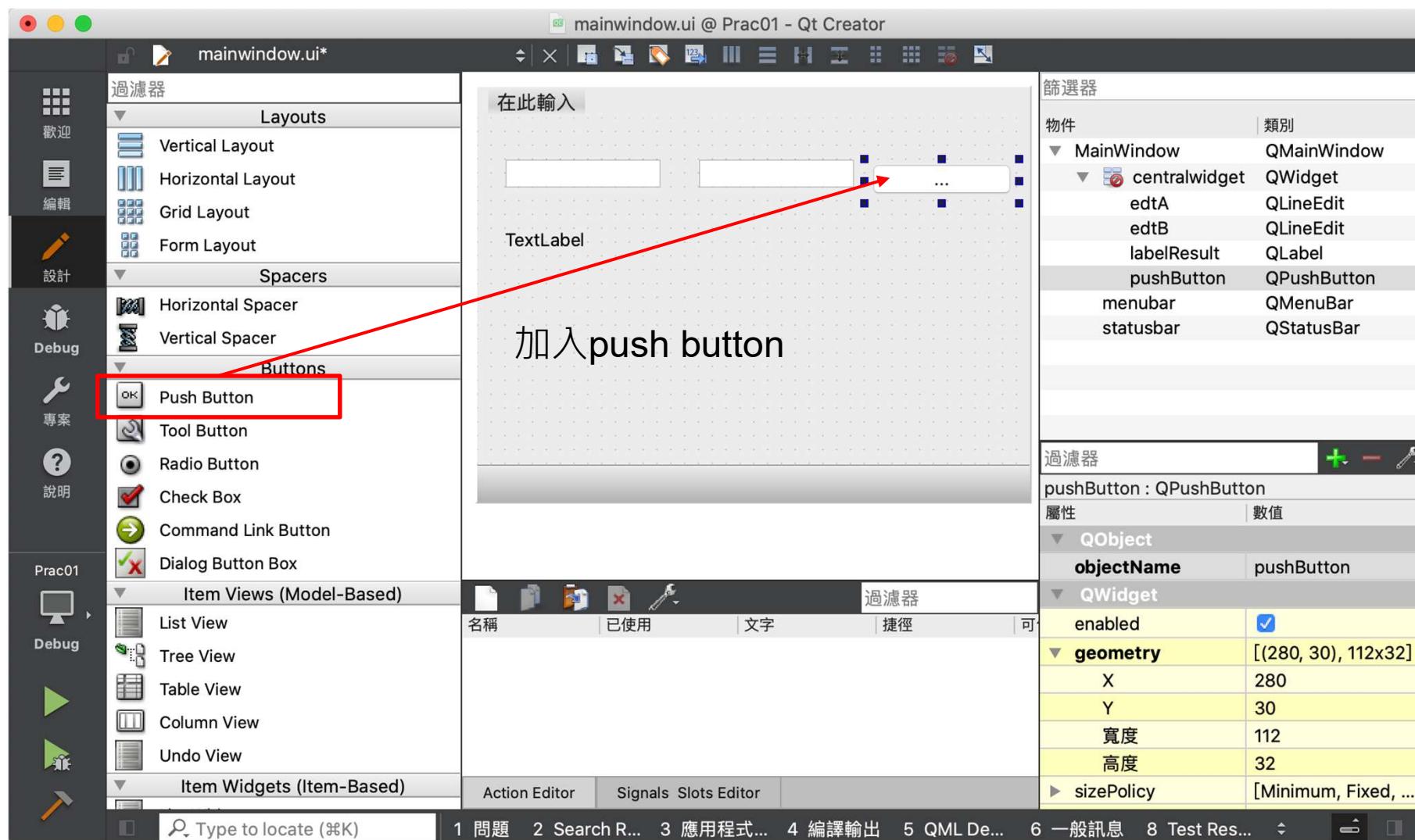
2. 設計UI



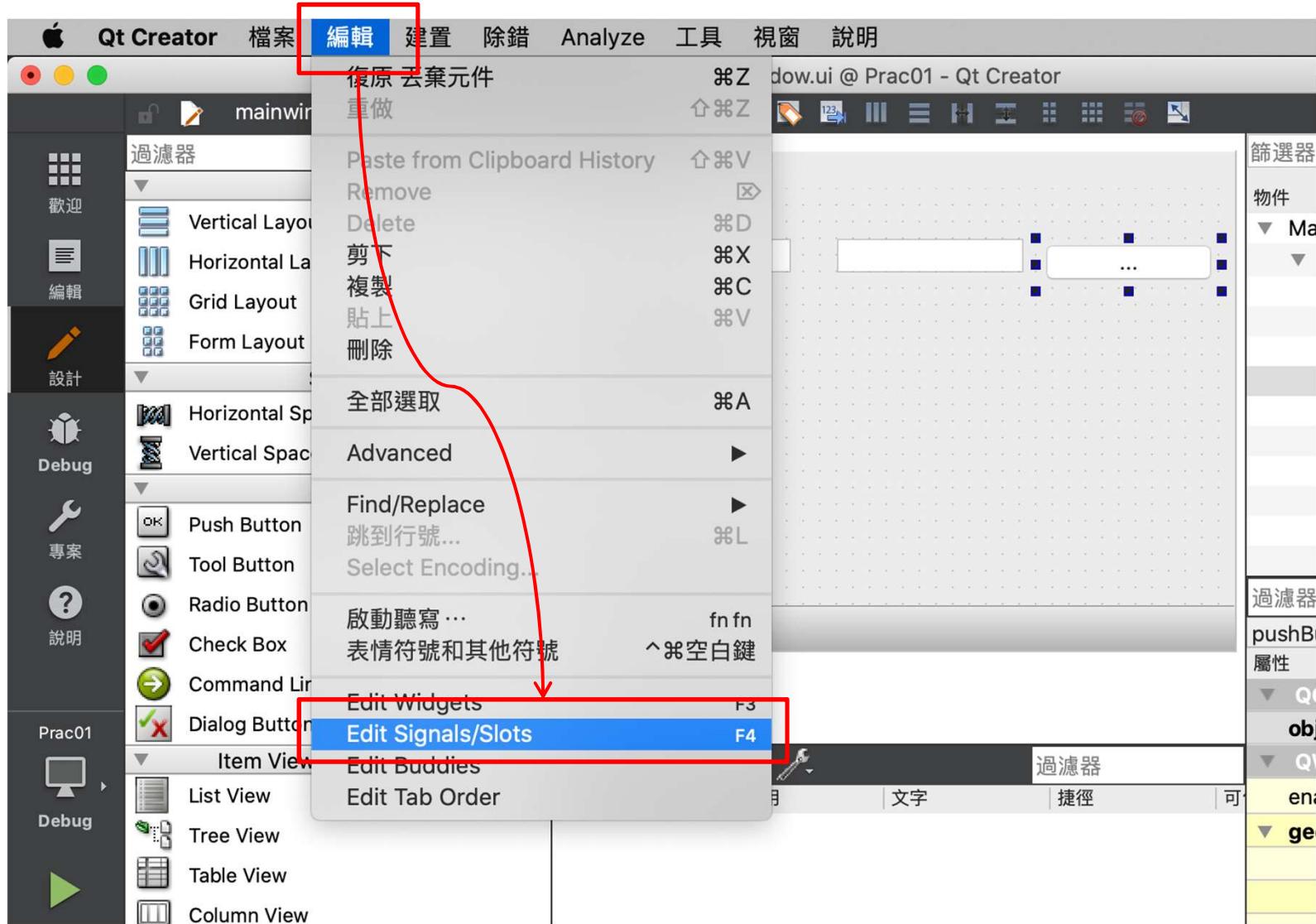


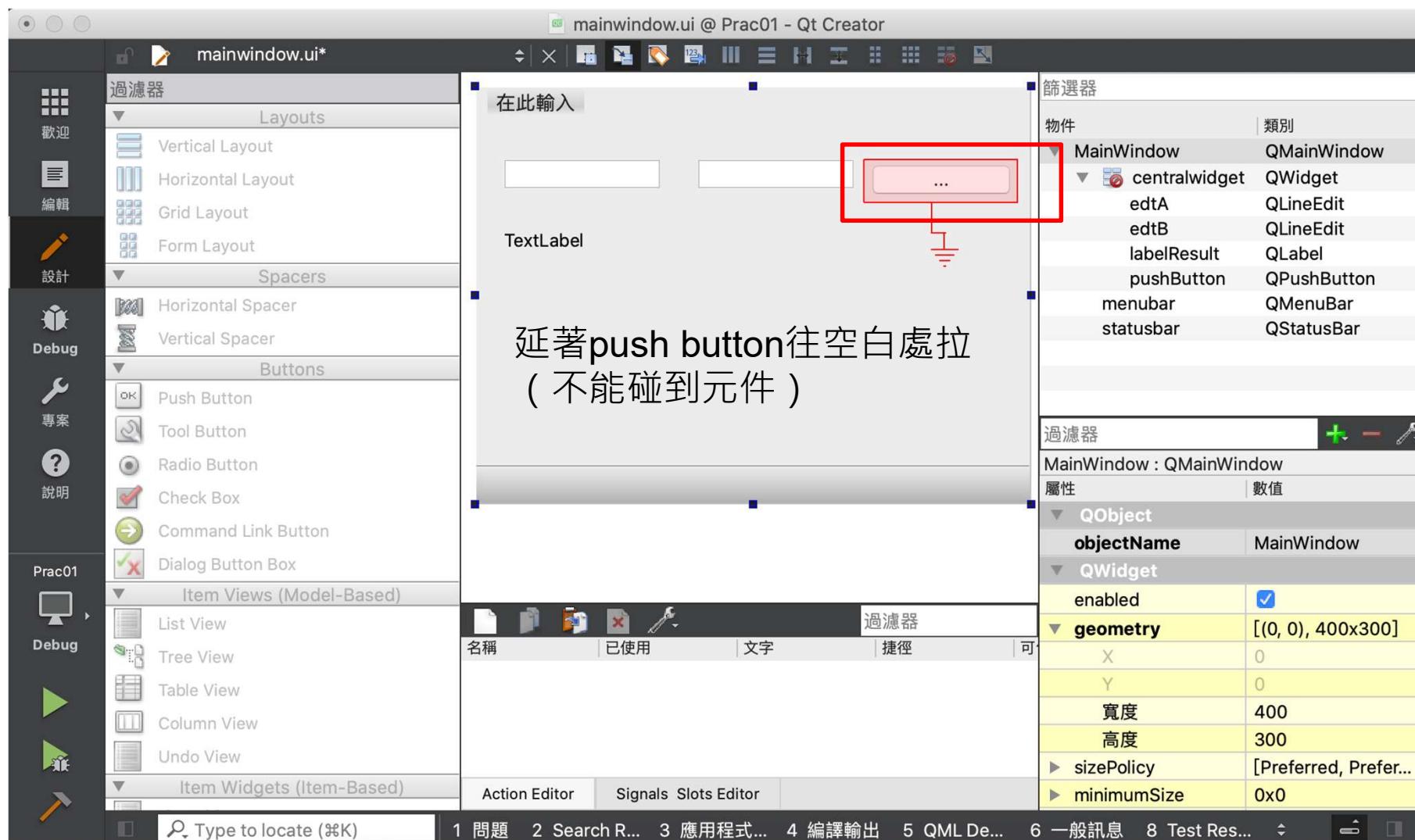


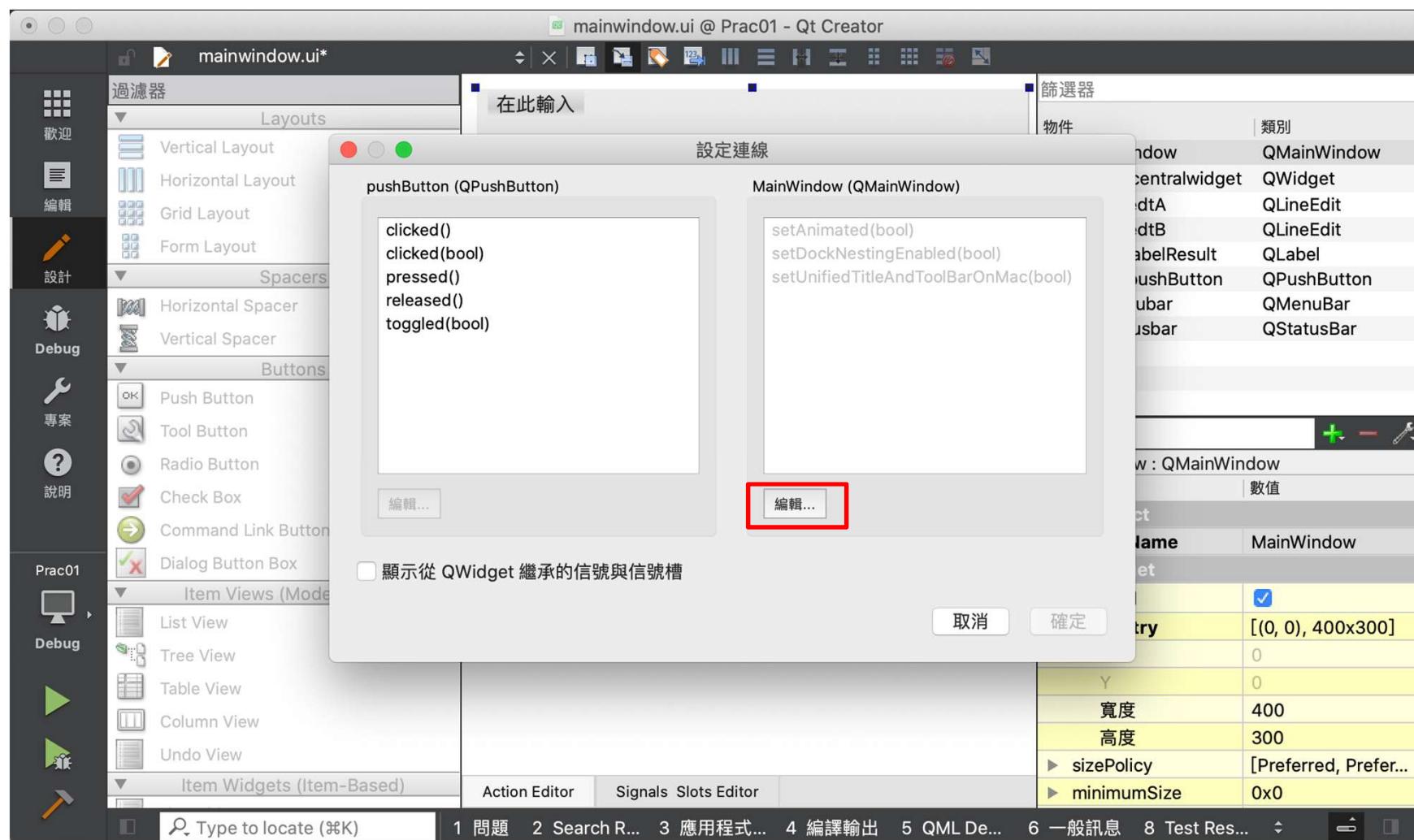


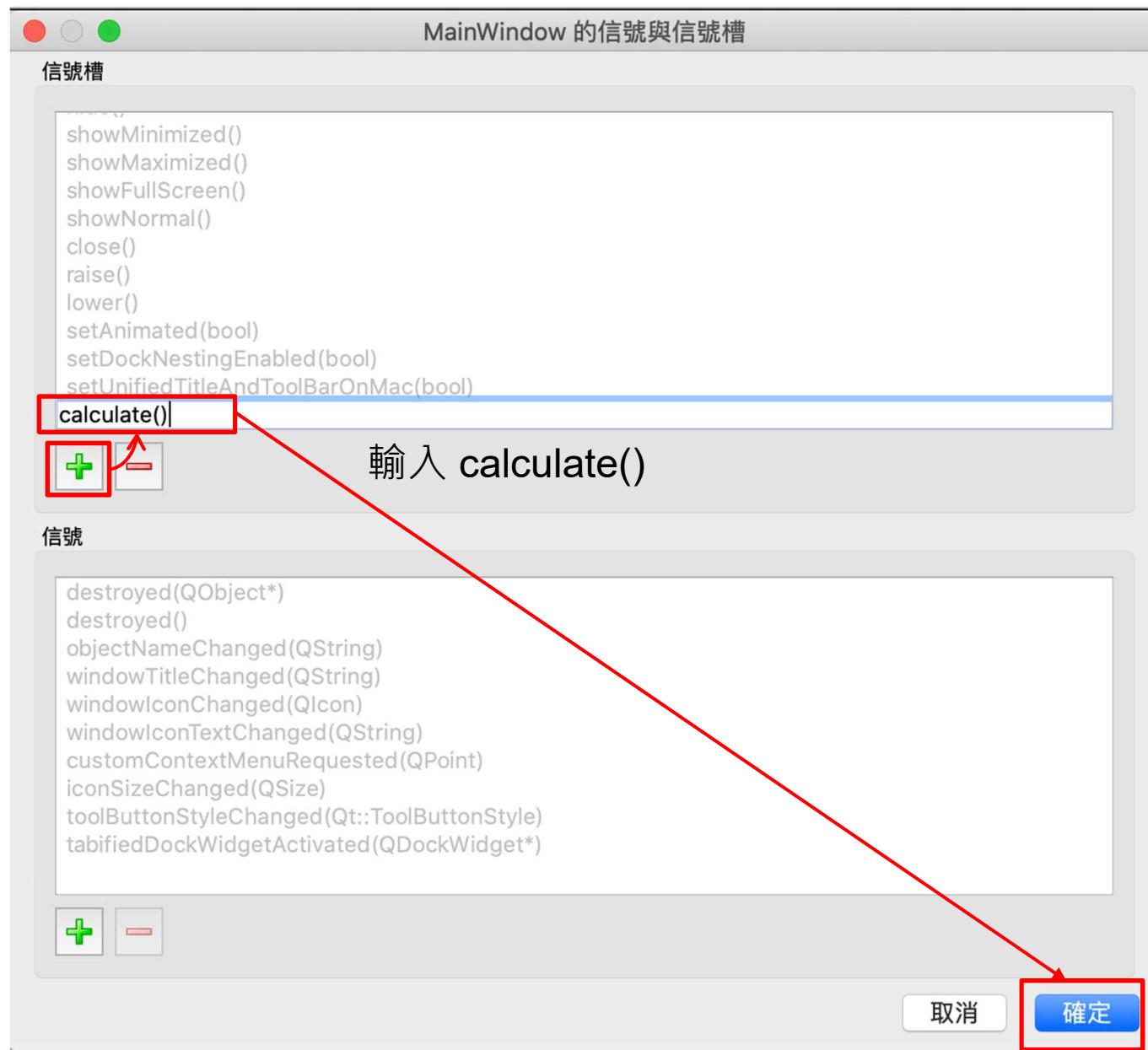


3. 連結Signals/slots

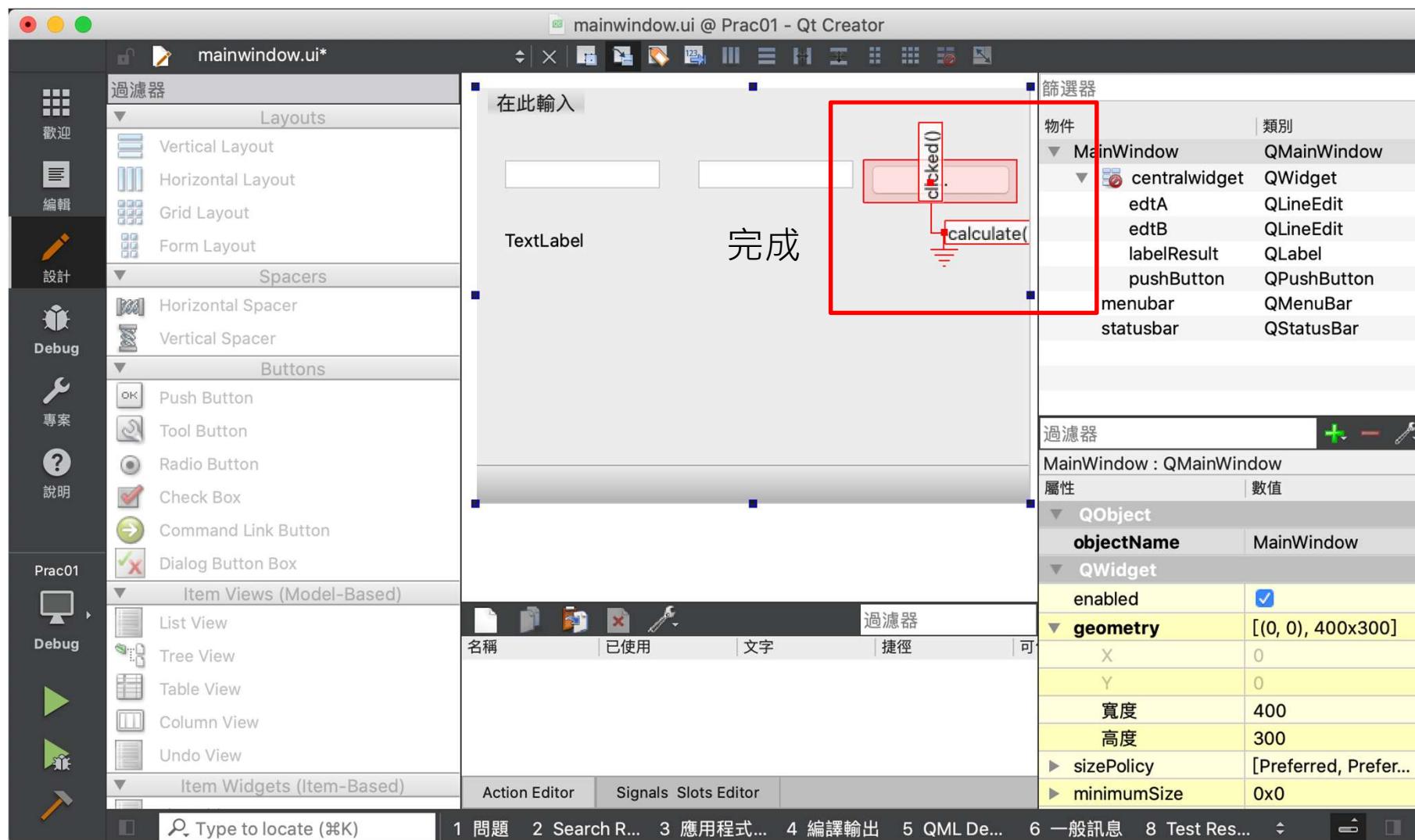




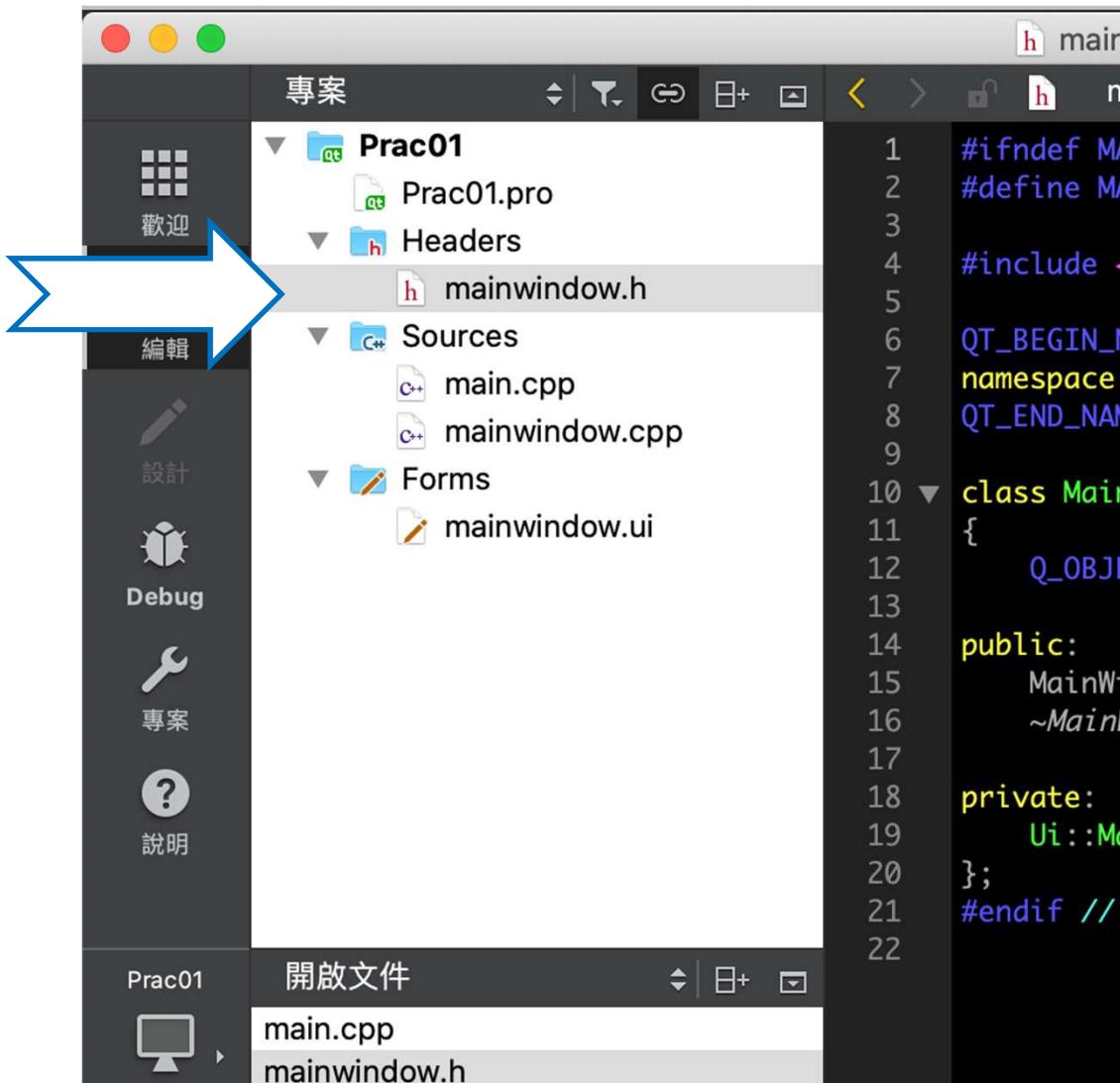








4. 宣告



The screenshot shows the Qt Creator IDE interface. On the left is a vertical toolbar with icons for Welcome, Edit (highlighted with a blue arrow), Design, Debug, Projects, and Help. The main window has a title bar with the project name "Prac01". The left pane displays the project structure:

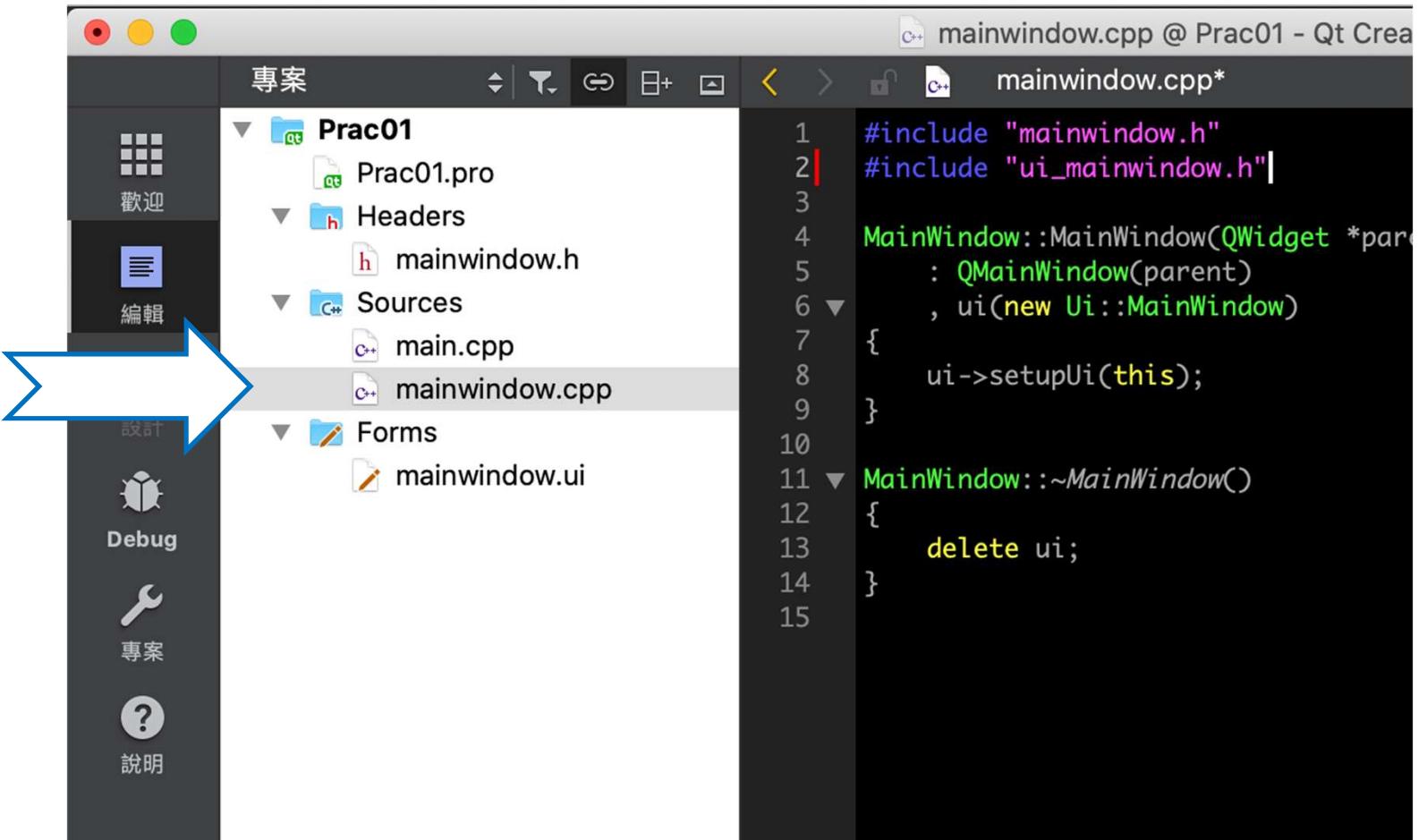
- Project: Prac01
- Prac01.pro
- Headers folder: mainwindow.h
- Sources folder: main.cpp, mainwindow.cpp
- Forms folder: mainwindow.ui

The right pane shows the code editor with the following content:

```
1 ifndef MAINWINDOW_H
2 define MAINWINDOW_H
3
4 include <QtWidgets>
5
6 QT_BEGIN_NAMESPACE
7 namespace Main {
8 QT_END_NAMESPACE
9
10 class MainWindow : public QMainWindow
11 {
12     Q_OBJECT
13
14 public:
15     MainWindow();
16     ~MainWindow();
17
18 private:
19     Ui::MainWindow *ui;
20 };
21
22 #endif // MAINWINDOW_H
```

```
1 #ifndef MAINWINDOW_H
2 #define MAINWINDOW_H
3
4 #include < QMainWindow>
5
6 QT_BEGIN_NAMESPACE
7 namespace Ui { class MainWindow; }
8 QT_END_NAMESPACE
9
10 class MainWindow : public QMainWindow
11 {
12     Q_OBJECT
13
14 public:
15     MainWindow(QWidget *parent = nullptr);
16     ~MainWindow();
17
18 public slots:
19     void calculate();
20
21 private:
22     Ui::MainWindow *ui;
23 };
24 #endif // MAINWINDOW_H
25
```

5. 輸入程式

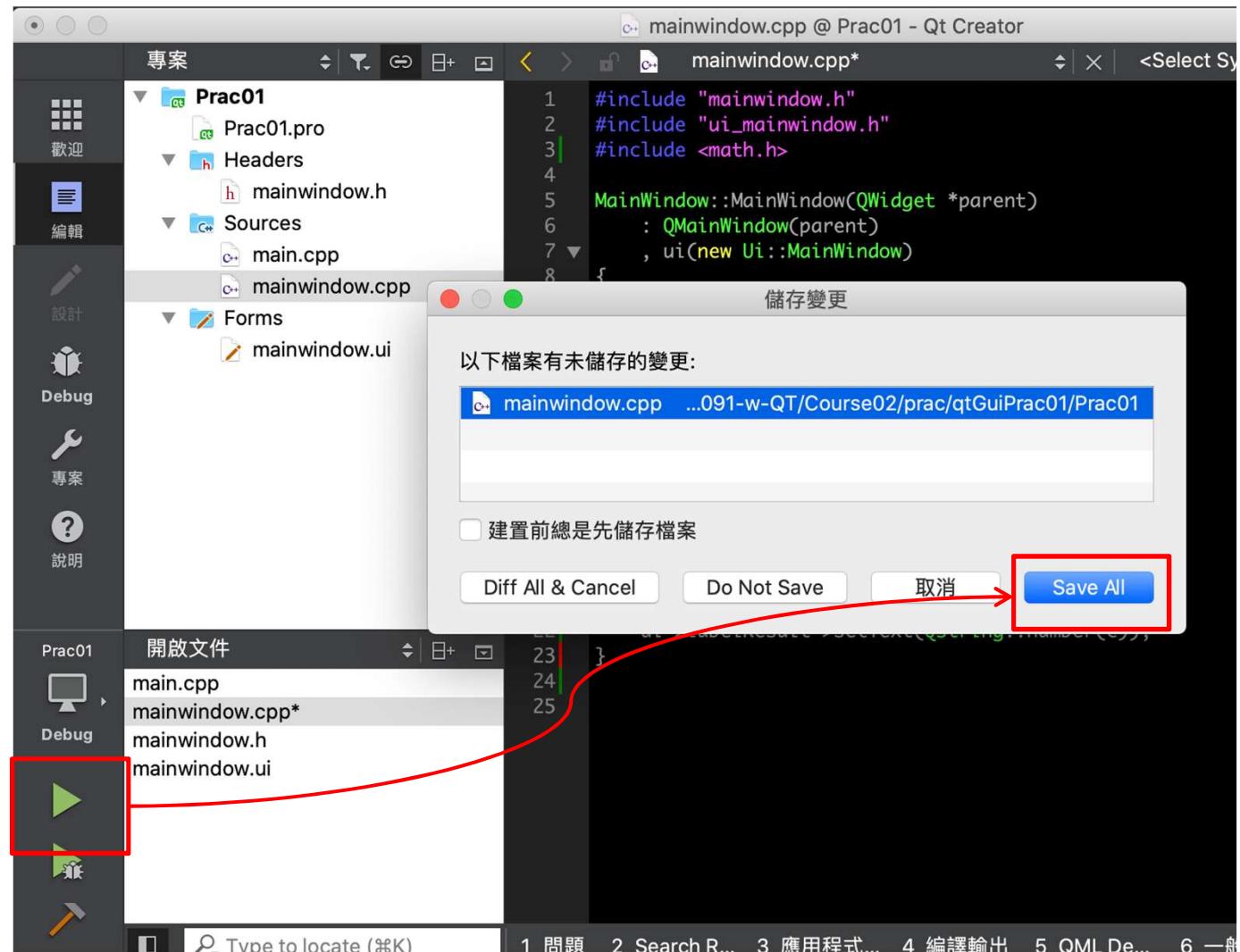


The screenshot shows the Qt Creator IDE interface. On the left, there is a vertical toolbar with icons for '歡迎' (Welcome), '編輯' (Edit), '設計' (Design), 'Debug', '專案' (Project), and '說明' (Help). A large blue arrow points from the '編輯' icon towards the main window. The main window has a title bar 'mainwindow.cpp @ Prac01 - Qt Crea'. Below the title bar is a toolbar with standard file operations. The central area shows a project tree for 'Prac01' containing 'Prac01.pro', 'Headers' (with 'mainwindow.h'), 'Sources' (with 'main.cpp' and 'mainwindow.cpp' selected), and 'Forms' (with 'mainwindow.ui'). The right side is a code editor displaying the contents of 'mainwindow.cpp'. The code is as follows:

```
1 #include "mainwindow.h"
2 #include "ui_mainwindow.h"
3
4 MainWindow::MainWindow(QWidget *parent
5     : QMainWindow(parent)
6     , ui(new Ui::MainWindow)
7 {
8     ui->setupUi(this);
9 }
10
11 MainWindow::~MainWindow()
12 {
13     delete ui;
14 }
```

```
1 #include "mainwindow.h"
2 #include "ui_mainwindow.h"
3 #include <math.h>
4
5 MainWindow::MainWindow(QWidget *parent)
6     : QMainWindow(parent)
7     , ui(new Ui::MainWindow)
8 {
9     ui->setupUi(this);
10 }
11
12 ~MainWindow()
13 {
14     delete ui;
15 }
16
17 void MainWindow::calculate()
18 {
19     int a = ui->edtA->text().toInt();
20     int b = ui->edtB->text().toInt();
21     double c = sqrt(pow(a,2)+pow(b,2));
22     ui->labelResult->setText(QString::number(c));
23 }
24
```

6. 執行



```
: QMainWindow(parent)
, ui(new Ui::MainWindow)
```

```
ui->set
```

MainWindow

```
inWindow:
```

3

4

...

```
delete
```

5

```
id MainWi
```

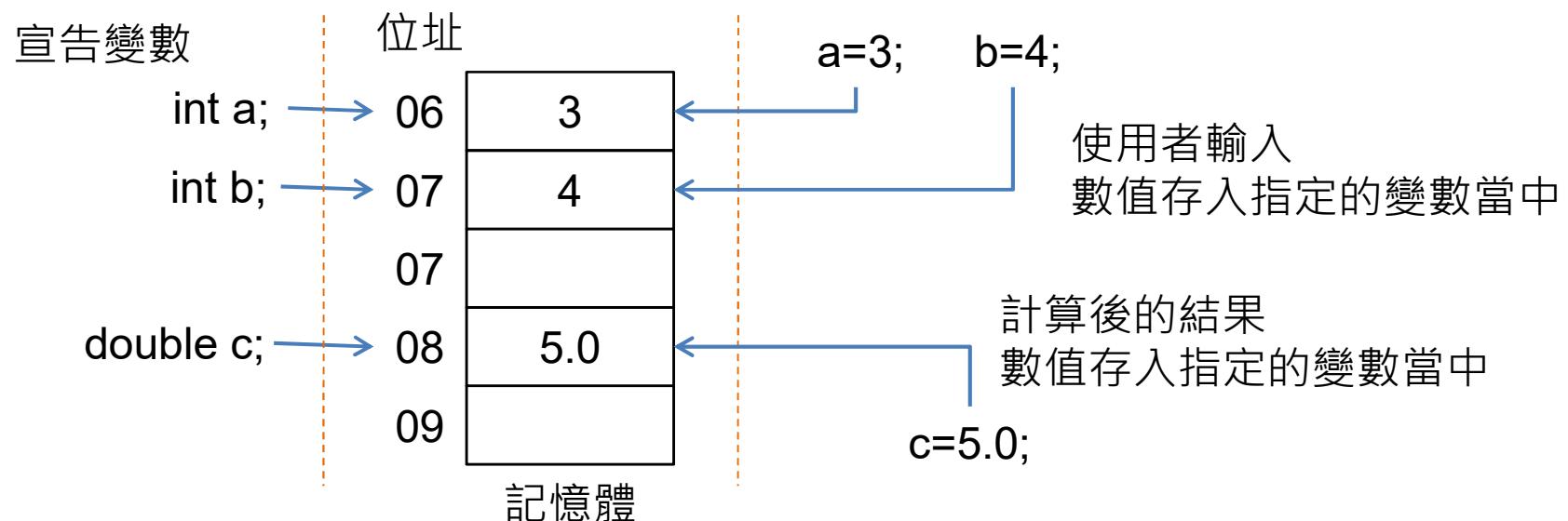
```
int a =
```

```
int b =
```

```
double
```

```
ui->lab
```

- 以上面程式為例
 - int a, b; 與 double c;
 - 變數宣告的程序
 - 宣告二個整數變數與一個浮點數變數
 - 只要宣告成功，變數名稱就會與記憶體中某一個記憶空間建立關連



- **內容如何被改變**

- **等號規則 – 指定**

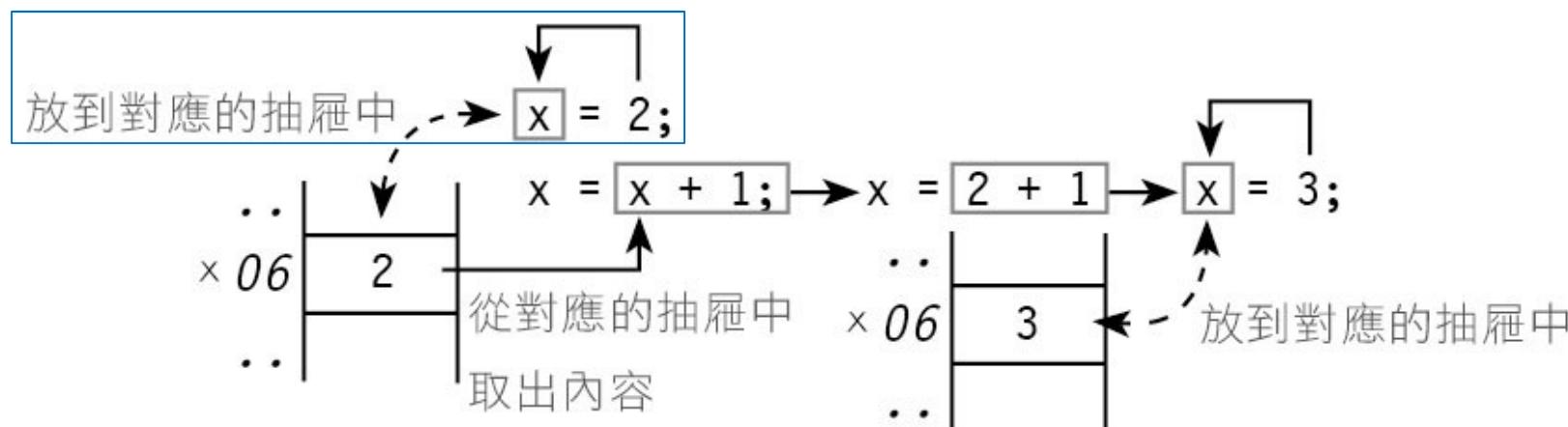
- 等號的左邊只能是變數，負責儲存運算的結果

- 只能有一個，不能是運算式

- 等號右邊是運算式、數值或是變數

- 右邊的運算結果儲存到等號左邊的變數中

- $x = x + 1;$



- 變數的使用，思考邏輯應該是
 - 這個變數應該設定成怎樣的內容
 - 這個變數的內容應該等於怎樣的運算結果
 - 這些運算的必須儲存到哪一個變數內
- 設定初始值(非常重要！！)
 - 變數宣告後第一次存入的內容，稱為變數的初始值
 - 編譯器的任務僅從作業系統當中取得記憶體
 - 將給定初值養成習慣
- 變數值的內容有兩種方式
 - 在宣告時指定
 - 在需要時指定（非宣告階段）

宣告變數的語法與變數命名規則

- C 語言變數宣告的語法

資料型態 變數名稱；

- 資料型態(data type)
 - 程式語言本身所提供的資料格式
 - int (整數) 、 double (浮點數)
- 變數名稱
 - 每個程式片段當中唯一
 - 不能使用識別字(identify)及關鍵字(keyword)
 - 使用英文及底線_(under)開頭，後面接數字或英文字母
 - 通常使用小寫

- 變數命名的規定
 - 變數的名稱不可以是關鍵字(Keywords)

auto	break	case	char	const	continue	default	do
double	else	enum	extern	float	for	goto	if
int	long	register	return	short	signed	sizeof	static
struct	switch	typedef	union	unsigned	void	volatile	while

_Bool	_Complex	_Imaginary	inline	restrict			
x	y						
_Alignas	_Alignof	_Atomic	_Generic	_Noreturn	_Static_assert	_Thread_local	

- 變數的名稱在程式中必須唯一

- 識別字(identifier)
 - 程式所有用到的變數、函式或自訂資料型態...等的名稱都稱為識別字
 - printf、scanf 也是識別字

常數的宣告與使用

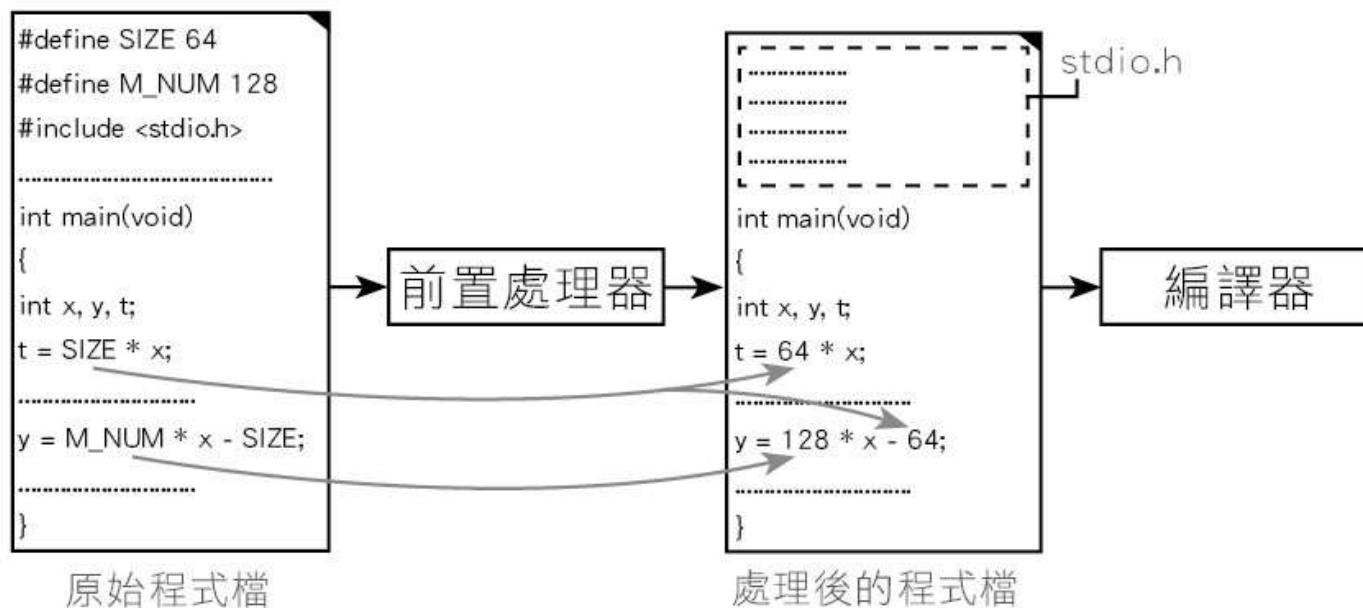
- 常數
 - 程式中會使用到的一些**固定不變**的數值、名稱或是字元
 - 圓周率、星期名稱與地名等的數值或是文字的組合
 - 線上遊戲中的組隊的人數上限、角色創立的最大數量...
- C 語言提供兩種常數的
 - 符號常數 (Symbolic Constant)
 - `#define TEAM_SIZE 8`
 - 常數變數
 - `const int team = 8;`

- 符號常數以 `#define` 這個前置處理器指令來宣告

`#define 常數名稱 常數值` ← 這裡不用分號

- 常數名稱
 - 可為任何字母的組合
 - 習慣上會使用全部大寫的字母來命名
 - 避免與關鍵字、變數或函式名稱產生不必要的衝突
 - 當需要使用多個單字來組合時，大都以下底線來連接
- 常數值
 - 舉凡整數、浮點數、單一字元、字串都可以是常數值

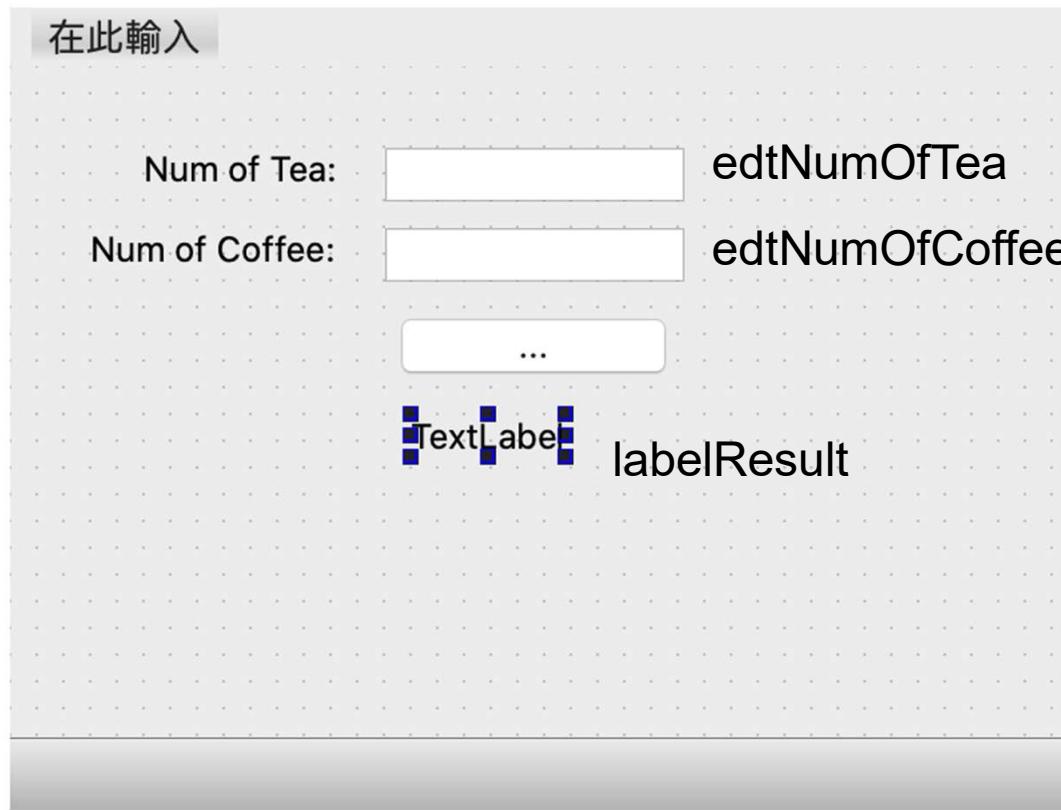
- `#define` 是前置處理器指令
 - 後面是不需要使用分號當結尾
 - 常數名稱與常數值之間，絕對不要自動幫他補上等號
 - 語法規定沒有，就別自己亂加
 - `#define CITYCODE = 24`
- 符號常數的處理方式
 - 前置處理器是以「尋找/取代」的方式來處理符號常數
 - 最後進入編譯階段的程式碼，是不會有符號常數

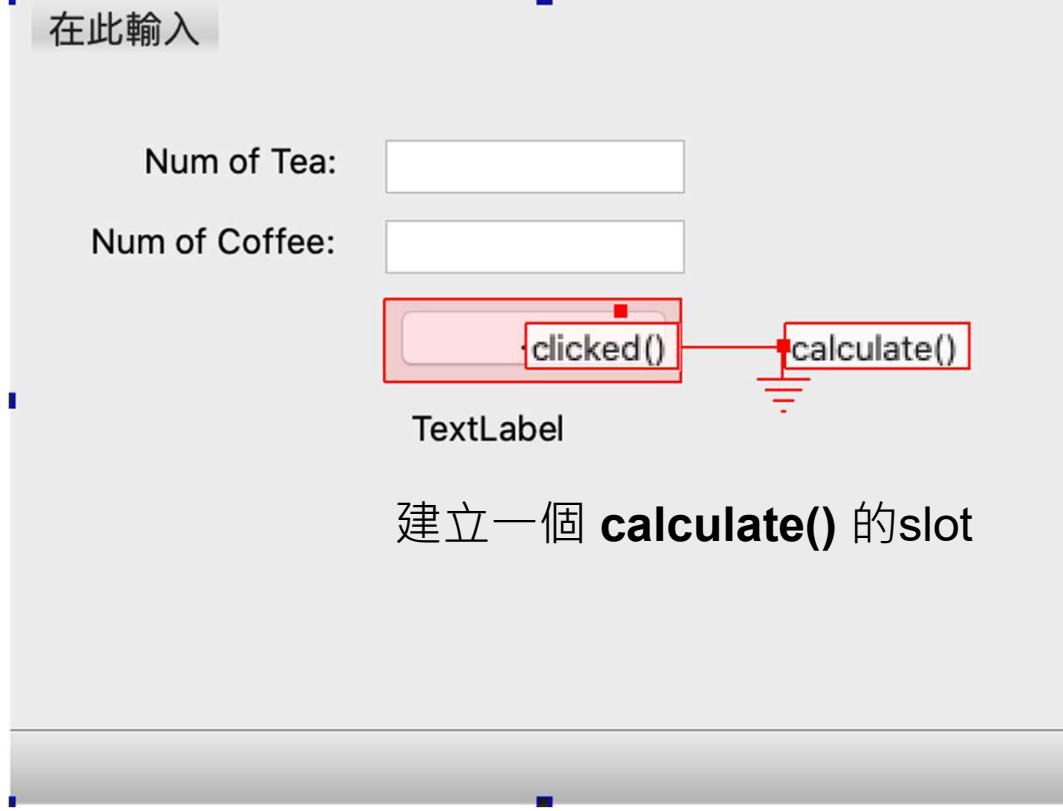


- 常數變數
 - 在變數宣告的敘述前面加上 const
 - 並在變數的後面指定初始值
const 資料型態 變數名稱 = 初始值;
- 例如： const int DefaultSTR = 25;
- 常數，內容就不能被修改
 - C 語法規定一定要在宣告時給予初始值
 - 不能用其他的敘述去改變常數變數的內容

練習

- 紅茶與咖啡每杯的售價各是20與35元，先詢問顧客要購買紅茶與咖啡各幾杯，然後告知顧客購買的總價
 - 將紅茶及咖啡的價錢定義為常數PRICE_OF_TEА及PRICE_OF_COFFEE
 - 若咖啡的價錢固定為紅茶的1.5倍，則將咖啡的價錢定義為常數公式
- Hint:
 - 定義兩個常數為紅茶及咖啡的價格
 - 宣告兩個變數儲存紅茶與咖啡的數量
 - 定義咖啡價格為紅茶1.5倍的價格
 - 可能會有小數出現!!





建立一個 **calculate()** 的slot

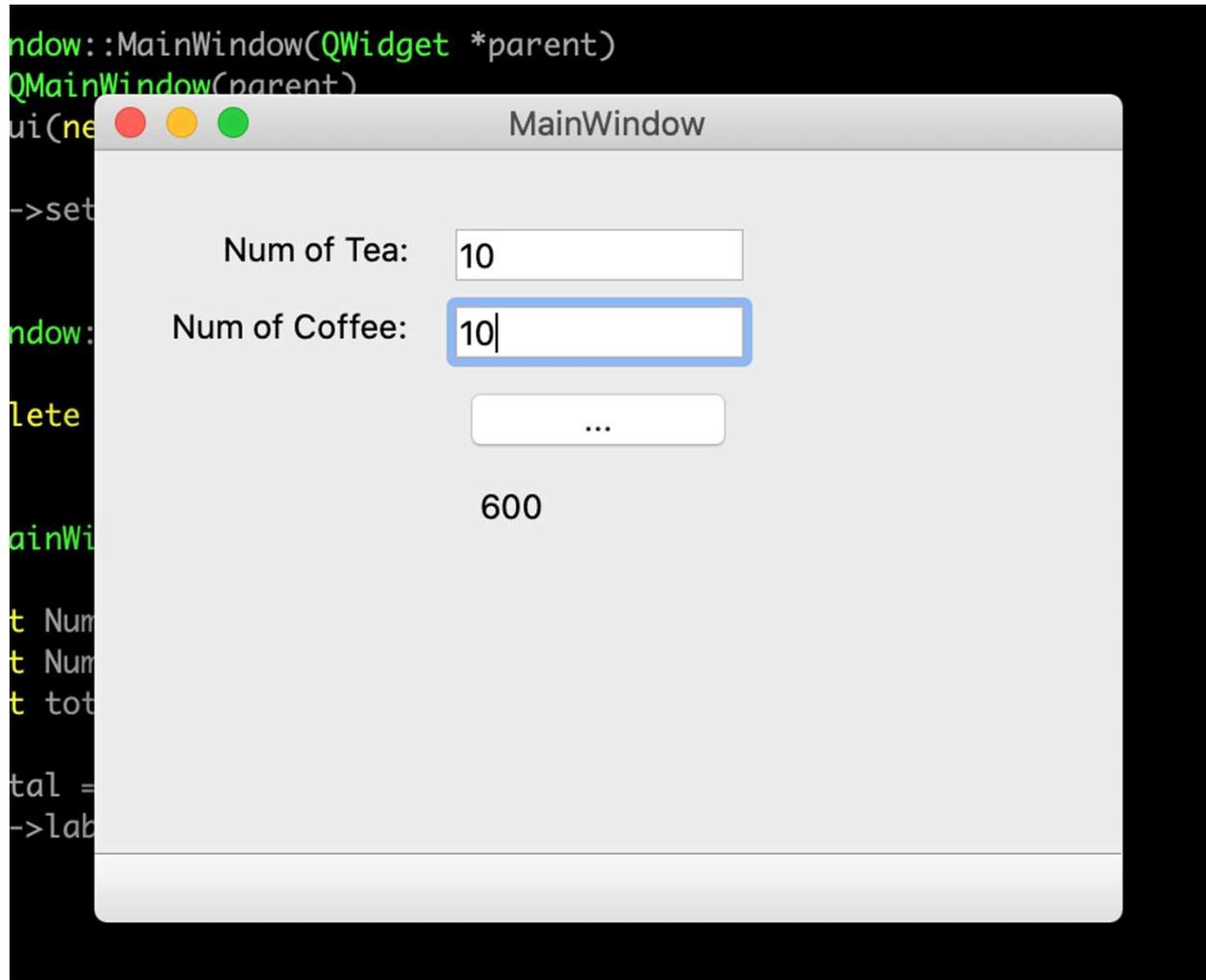
```
1 #ifndef MAINWINDOW_H
2 #define MAINWINDOW_H
3
4 #include < QMainWindow>
5
6 QT_BEGIN_NAMESPACE
7 namespace Ui { class MainWindow; }
8 QT_END_NAMESPACE
9
10 class MainWindow : public QMainWindow
11 {
12     Q_OBJECT
13
14 public:
15     MainWindow(QWidget *parent = nullptr);
16     ~MainWindow();
17
18 public slots:
19     void calculate();
20
21 private:
22     Ui::MainWindow *ui;
23 };
24 #endif // MAINWINDOW_H
25
```

宣告

```
1 #include "mainwindow.h"
2 #include "ui_mainwindow.h"
3
4 #define PRICE_OF_TEA 25
5 #define PRICE_OF_COFFEE 35
6
7 MainWindow::MainWindow(QWidget *parent)
8     : QMainWindow(parent)
9     , ui(new Ui::MainWindow)
10 {
11     ui->setupUi(this);
12 }
13
14 ~MainWindow()
15 {
16     delete ui;
17 }
18
19 void MainWindow::calculate()
20 {
21     int NumOfTea = ui->edtNumOfTea->text().toInt();
22     int NumOfCoffee = ui->edtNumOfCoffee->text().toInt();
23     int total;
24
25     total = PRICE_OF_TEA*NumOfTea + PRICE_OF_COFFEE*NumOfCoffee;
26     ui->labelResult->setText(QString::number(total));
27 }
28
```

前端處理程式

函數，實現slot



```
1 #include <stdio.h>
2 #define PRICE_OF_TEА 25
3 #define PRICE_OF_COFFEE PRICE_OF_TEА*1.5
4
5 int main(int argc, char *argv[])
6 {
7     int NumOfTeа, NumOfCoffee;
8     int totalPrice;
9     printf("Number of tea: ");
10    scanf("%d", &NumOfTeа);
11    printf("Number of COFFEE: ");
12    scanf("%d", &NumOfCoffee);
13    totalPrice =
14        PRICE_OF_TEА*NumOfTeа+PRICE_OF_COFFEE*NumOfCoffee;
15    printf("total price = %d", totalPrice);
16    return 0;
17 }
```

咖啡的單價為小數(浮點數)，
所以總價的部份用整數
可能會有問題
羅輯的迷思

基本資料型態

基本資料型態

- 不同的資料型態 (data type) 應付不同的需求
- 資料型態
 - 字元
 - 字元 (char)
 - 整數
 - 短整數(short int or short)
 - 整數(int)
 - 長整數(long int or long)
 - 整數部分，提供了「unsigned」關鍵字可將整數設定成沒有正負號的整數，
 - 浮點數
 - 單倍精確度浮點數 (float)
 - 雙倍精確度浮點數 (double)

對每一種資料型態來說，必須記住的資訊

	位元組	資料型別的宣告關鍵字	表示範圍
有正負號 整數	字元	1	char
	短整數	2	short
	整數	4	int
	長整數	4	long
無正負號 unsigned	字元	1	unsigned char
	短整數	2	unsigned short
	整數	4	unsigned int
	長整數	4	unsigned long
浮點數	單倍精確度	4	float
	雙倍精確度	8	double

- 整數，三種資料型態：
 - 短整數 (short)
 - 使用 2 位元組來儲存內容
 - 使用 short 或是 short int 來宣告
 - short x; short int x; 都能宣告 x 為短整數
 - 整數 (int)
 - 使用 2/4 位元組來儲存內容
 - 使用 int，如：int ix;
 - 長整數 (long)
 - 使用 4 位元組來儲存內容
 - 使用 long 或是 long int 來宣告
 - long lx; short int lx; 都能宣告 lx 為長整數。
 - 某些編譯器會設定 8 個位元組給長整數使用，
 - 使用時可以先用 sizeof 查詢一下使用位元組數

- 無正負號的整數 – `unsigned`
 - 整數的三個型態可以利用 `unsigned` 改成無正負號的整數
 - 加大其在正數的表示範圍。
 - 例如：
 - `unsigned short usV; // 宣告 usV 為沒有正負號的短整數`
 - `unsigned int uiV; // 宣告 uiV 為沒有正負號的整數`
 - `unsigned long ulV; // 宣告 ulV 為沒有正負號的長整數`

- 浮點數：
 - 單倍精確度浮點數 (float)
 - 使用 4 位元組來儲存內容，型態名稱為 **float**
 - 如：float fx; 宣告 fx 為單倍精確度浮點數。
 - 所能表達的精確度只到小數點下第 6 與第 7 位，之後就會有產生誤差
 - 雙倍精確度浮點數 (double)
 - 使用 8 位元組來儲存內容，型態名稱為 **double**
 - 如：double dV; 宣告 dV 為雙倍精確度浮點數
 - 精確度到達小數點下第 15 到第 16 位
- C 語言對帶有小數點的數值，預設都是以 **double** 型態來表示
- 數值後面加上 f 或 F，就會被視為 float 型態
 - 如：float fV = 1.23f
 - 也可以利用科學記號表示法來表達
 - 如：4.56E-3f = 4.56*10-3f = 0.00456f

- 字元
 - 指一個英文/數字字母
 - 來自英文字 character，因此 char 就是型態名稱
 - 僅使用 1 個位元組來儲存資料
 - 指定資料給字元變數時，用兩個單引號 ‘’來包住單一字元
 - `char cx = 'b'` 表示將字母 b 存放到字元變數 cx 裡
- NOTE
 - 存放在 cx 中非字母 b，而是字母 b 所對應的 ASCII 碼為 98
 - 雖然型態名稱是字元，但字元變數裡面實際儲存的是數值
 - 以整數來看，其數值的範圍是 $-128 \sim 127 (-2^7 \sim 2^7 - 1)$
 - 如果加上 `unsigned` 則範圍變成 $0 \sim 255 (0 \sim 2^8 - 1)$

- 字元所能儲存的都是單一的英文字母或符號
 - ASCII 編碼中的符號（參考附錄C）
- 有些字元在是控制字元，無法顯示
 - 嘿一聲 (bell)
 - 倒退鍵 (backspace)
 - 換行
- 有些則是因為該字元已經被 C 語言拿去使用
 - 單、雙引號等
 - 利用「跳脫字元」

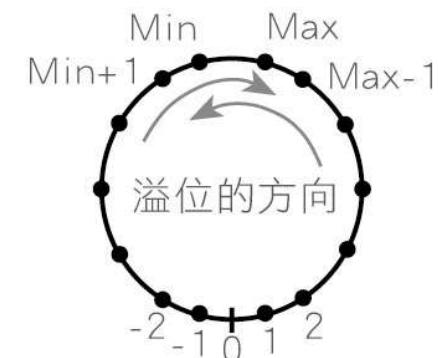
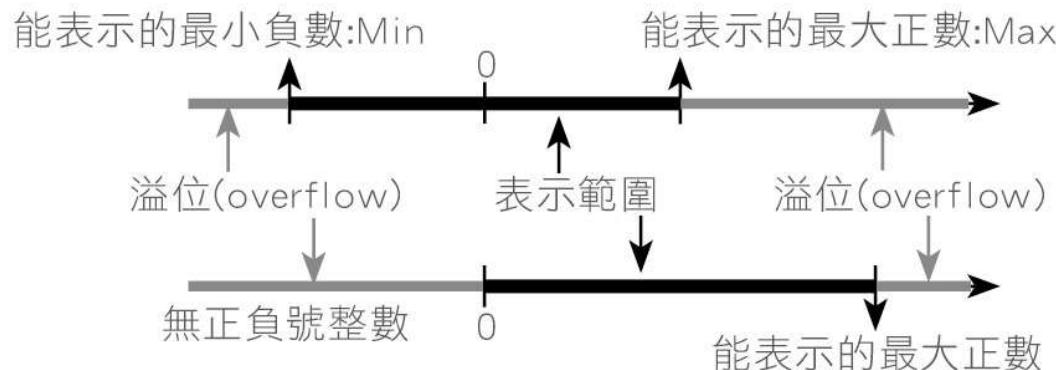
- 這類特殊的字元必須利用反斜線「\」來表達
 - 「\」稱為跳脫字元 : escape character
 - 反斜線與控制碼的組合稱為跳脫序列 (escape sequence)
 - 當遇到反斜線時 (跳脫字元) , 之後跟隨的字元將當成命令來執行
 - 例如：
 - printf 中的 \n 就是一個跳脫序列
 - n 就被當成換行 (new line) 命令來執行

表 2-1 常用的跳脫序列

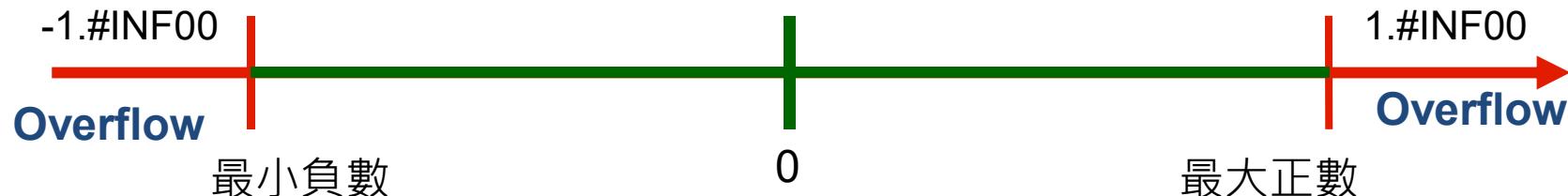
跳脫序列	代表的意義	ASCII的十進位/十六進位值
\0	字串結束字元(null character)	0 / 0x0 (數字0 不是字母O)
\a	嗶一聲 (bell)	7 / 0x7
\b	倒退鍵 (backspace)	8 / 0x8
\t	跳格 (Tab)	10 / 0x9
\n	換行 (new line)	10 / 0xA
\r	歸位(carriage return)	13 / 0xD
\"	雙引號	34 / 0x22
'	單引號	39 / 0x27
\\	反斜線	92 / 0x5C

溢位 Overflow

- 溢位：當儲存的數超出變數容許範圍時
- 整數的溢位



- 浮點數的溢位



- 編譯器對於溢位的處理都浮現的問題
 - 程式在執行階段發生溢位時，編譯器所產生的執行碼是不會幫你做額外的檢查
 - 程式並不會因為溢位而立即停止，結果通常是
 - 導致不當的記憶體存取，程式因此被迫停止執行
 - 變數溢位所導致的程式錯誤是程式語意錯誤的主要來源之一
- 變數內容的改變一定要注意是否超出該資料型態的表示範圍
- 不能使用過大的型態來取代，一定要思考適合的型態!!

程式

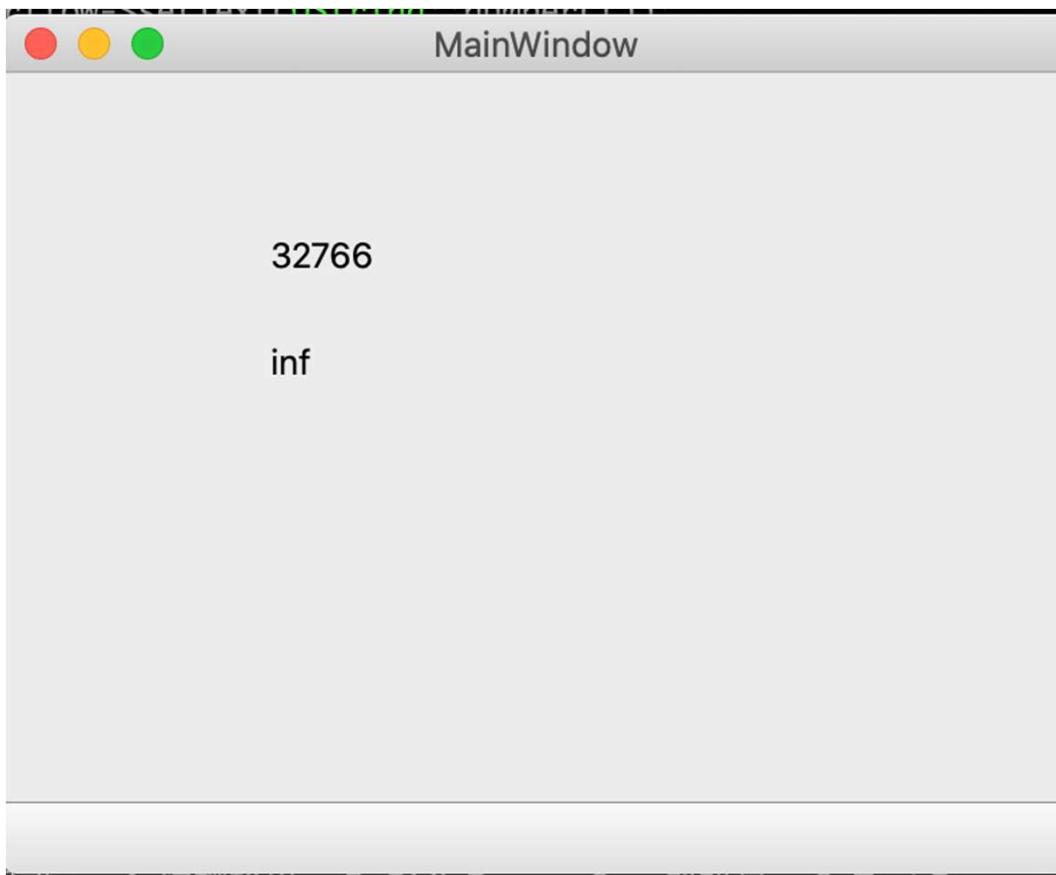
- 溢位觀察
 - 宣告一個短整數，初值為-32766，並將其減4
 - 宣告一個浮點數，初值為4.4E+38f，並將其乘以2

在此輸入

TextLabel **labelIntOverflow**

TextLabel **labelFloatOverflow**

```
1 #include "mainwindow.h"
2 #include "ui_mainwindow.h"
3
4 MainWindow::MainWindow(QWidget *parent)
5     : QMainWindow(parent)
6     , ui(new Ui::MainWindow)
7 {
8     ui->setupUi(this);
9     short s = -32766;
10    ⚠ 10   float f = 4.4E+38;
11    s = s-4;
12    f = f*2;
13    ui->labelIntOverflow->setText(QString::number(s));
14    ui->labelFloatOverflow->setText(QString::number(f));
15 }
16
17 MainWindow::~MainWindow()
18 {
19     delete ui;
20 }
21
```



基本算數運算

- 運用變數解決問題，最常用的就是數學上的四則運算
 - 「+」、「-」、「*」、「/」這四個符號可以直接用在程式中
 - 但括弧要注意一下
 - 數學上寫 $(a+b)(a-b)$ ，代表 $(a+b)$ 與 $(a-b)$ 相乘
 - 程式則必須寫成 $(a+b)*(a-b)$
- 當不同型態的變數或數值被四則運算組合在一起時
 - 規則：運算符號的兩邊必須是相同的資料型態才會進行計算

- C 語言會自動幫數值或是變數的內容進行型態的轉換
 - 規則：根據型態所能表示範圍的大小「**將小的轉成大**」
 - 目的：**提高計算結果的精確度**
- 資料型態表示範圍由小到大依序是
 - **char**、**short**、**int**、**long**、**float** 與 **double**
- 數值預設的資料型態如下：
 - 整數
 - 如：123或56，預設的資料型態為 **int**
 - 後面加上 L 或是 l，如：123l 或 56L，數值的型態為 **long**
 - 浮點數
 - 如 1.23 或 3.1415，預設的資料型態為 **double**
 - 後面加上 f，如：1.23f 或 3.14f 則該數值的型態為 **float**

- 強制型態轉換
 - 在變數之前加上「**(資料型態名稱)**」，就可以將該變數轉成括弧內所指定的型態
 - `int x=(float)10;` 或是 `(float) x`
- 以除法為例，要得到帶有小數的計算結果
 - 數值可以先寫成帶有小數的形式
 - 相除的兩邊都是整數變數，但是又希望得到帶有小數的結果
 - 必須使用強制型態轉換的功能
- 範例
 - `ix = 4; iz = 2; iy = 3; fy = 3.0f;`
 - `fr = iz/iy`，寫成 `fr = (float)iz/iy`
 - `fr` 的儲存內容變成 `0.666666` (**why?**)
 - 寫成 `fr=(float)(iz/iy)`，並不會得到這樣的結果 (**why?**)

程式

- 型態轉換
 - 宣告兩整數變數iz及iy，初值分別為2及3，將iz/iy，觀察其結果

在此輸入

TextLabel

labelCast1

TextLabel

labelCast2

```
1 #include "mainwindow.h"
2 #include "ui_mainwindow.h"
3
4 MainWindow::MainWindow(QWidget *parent)
5     : QMainWindow(parent)
6     , ui(new Ui::MainWindow)
7 {
8     ui->setupUi(this);
9
10    int iz=2;
11    int iy=3;
12    ui->labelCast1->setText(QString::number( (float)iz/iy   ));
13    ui->labelCast2->setText(QString::number( (float)(iz/iy) ));|
14 }
15
16 ~MainWindow()
17 {
18     delete ui;
19 }
20
```

