C 程式語言

C 語言

- C 語言是 1972 年由 Dennis Ritchie 在貝爾實驗室，而發展出的程式語言。

- 如今，許多新的程式語言也都直接或間接的受到 C 語言的影響，例如直接由 C 語言衍生出的有 C++ 、 Objective-C 、 Java 、 C# 等。

C 語言

- main() 是 C 語言的主程式，每一個 C 的程式都必須有一個 main 函數，而且只能有一個
- 函數名稱後面接著一對小括號，而下面的一對大括號 {} 則是 main 函數的程式主體
- 程式主體的每一行敘述必須以分號；做為結束
- 註解是給人看的，程式開發軟體不會去處理註解
- 多行註解要成對出現 /*　　*/　　單行註解　//

C 語言

- 程式註解（Comments）是程式中十分重要的部分，可以提供程式內容的進一步說明，良好註解不但能夠了解程式目的，並且在程式維護上，也可以提供更多的資訊。

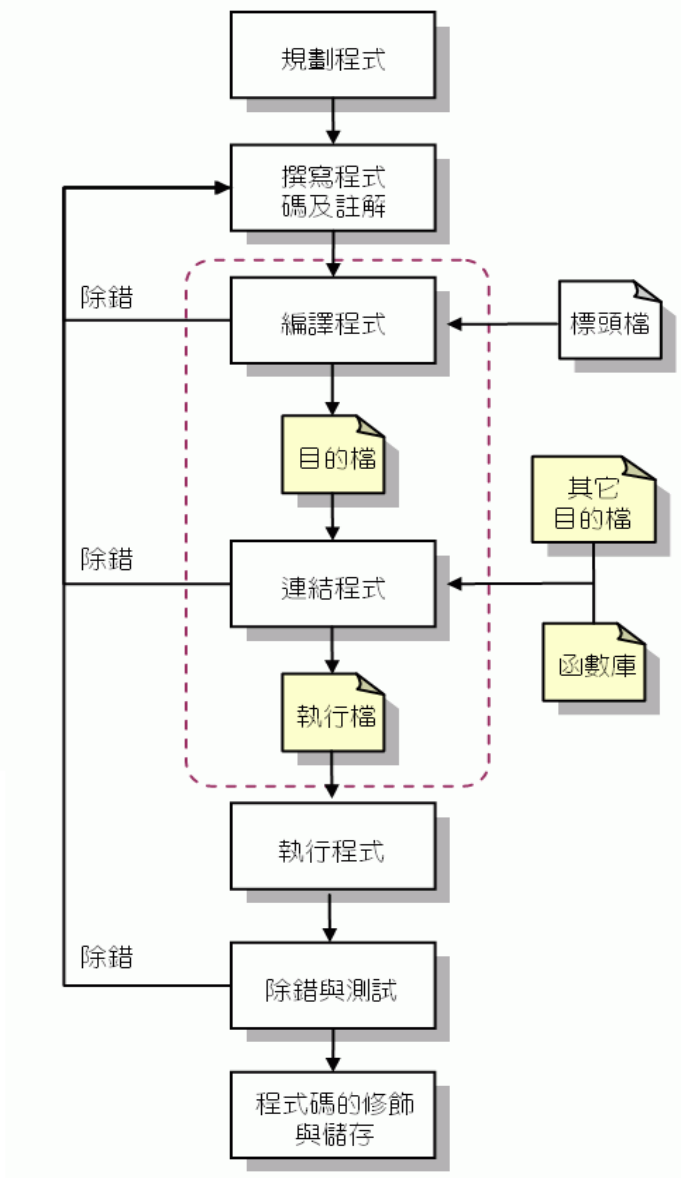
- C語言的註解可以跨很多列，如下所示：

```
/* -------------------------
   程式範例: Ch2_3.c
 ------------------------- */
```

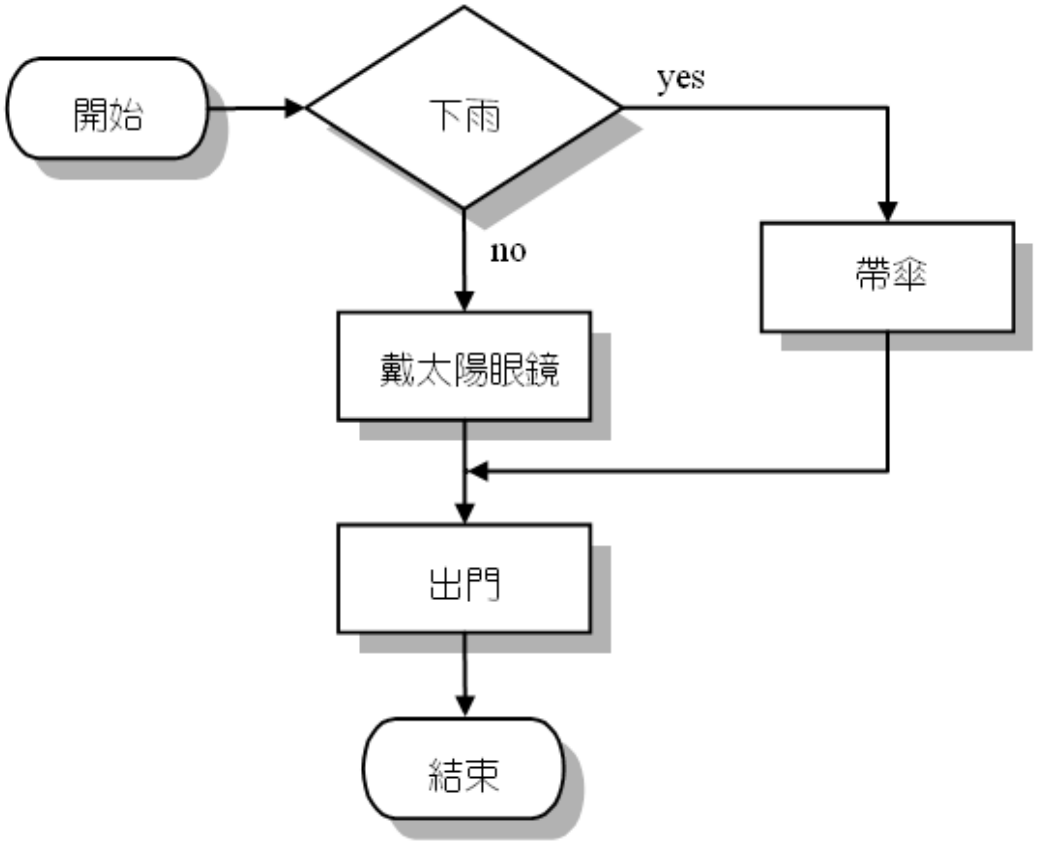
- printf("第一個C應用程式\n");　　// 顯示訊息

程式規劃與實作的流程

- 規劃程式
- 撰寫程式碼及註解
- 編譯程式碼
- 執行程式
- 除錯與測試
- 程式碼的修飾與儲存



流程圖



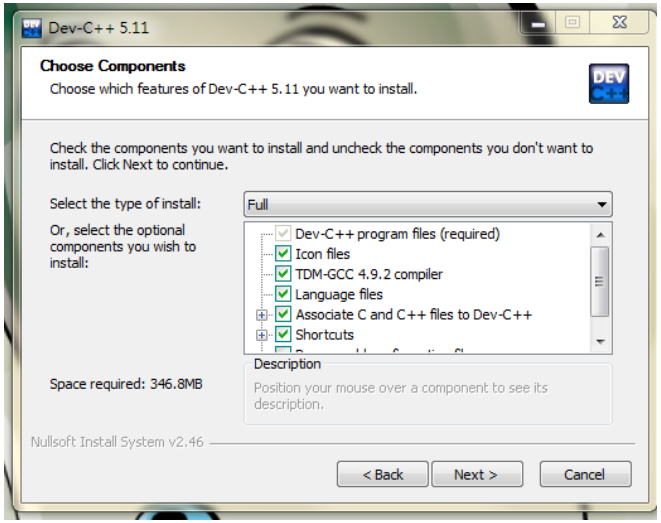
如果下雨，則帶傘，否則載太陽眼鏡。不管是否下雨，最後都要出門

C程式語言

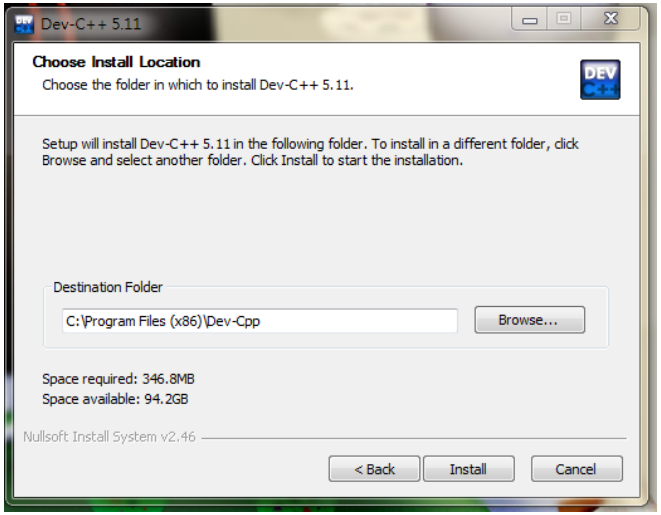
- Dev-C++ 有一段時間都沒有改版更新（從 2005 年 2 月 22 日起），Orwell Dev-C++ 是 Dev-C++ 的衍生版本，此版本直到現在仍然持續提供錯誤更新和新版編譯器（支援 64 位元編譯器），和提供免安裝的可攜式版本。

安裝 DEV Cpp

安裝模式選擇Full

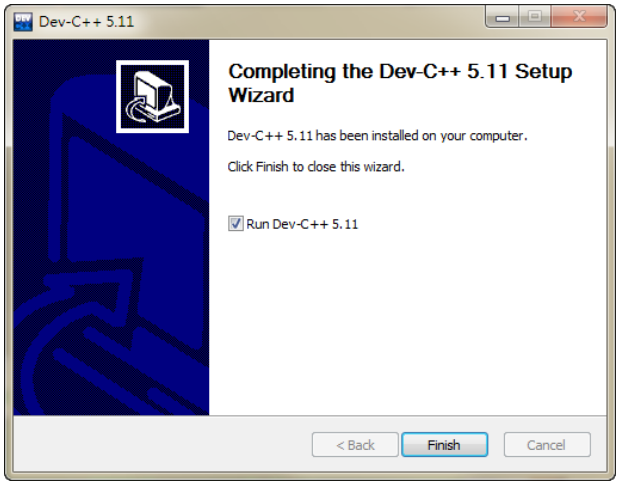


安裝路徑使用預設

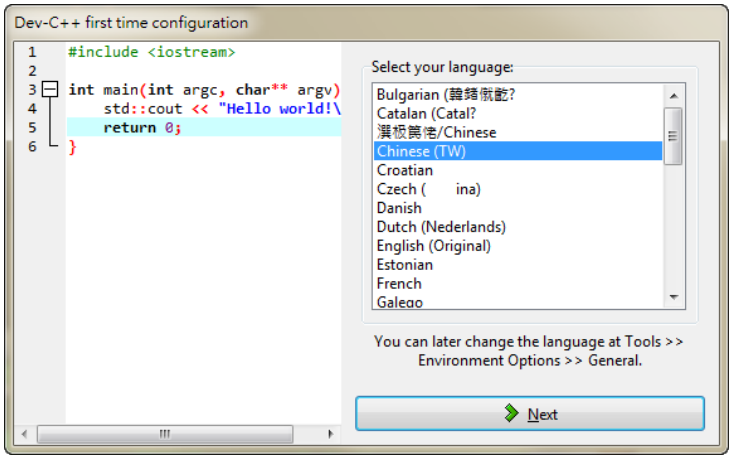


安裝 DEV Cpp

安裝完畢, 點選Finish

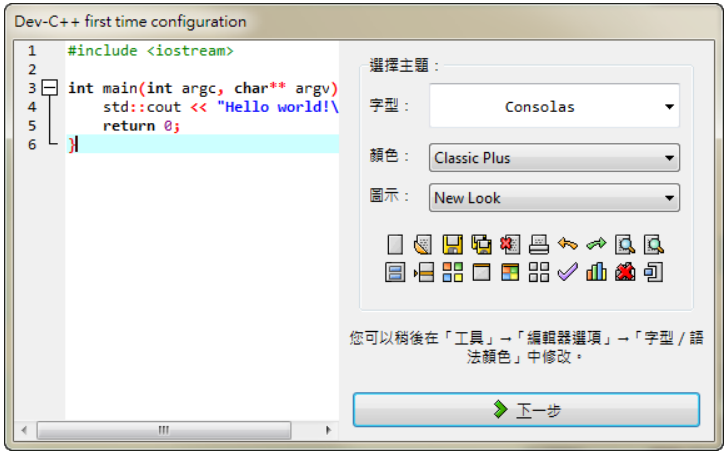


支援選擇Chinese(TW)

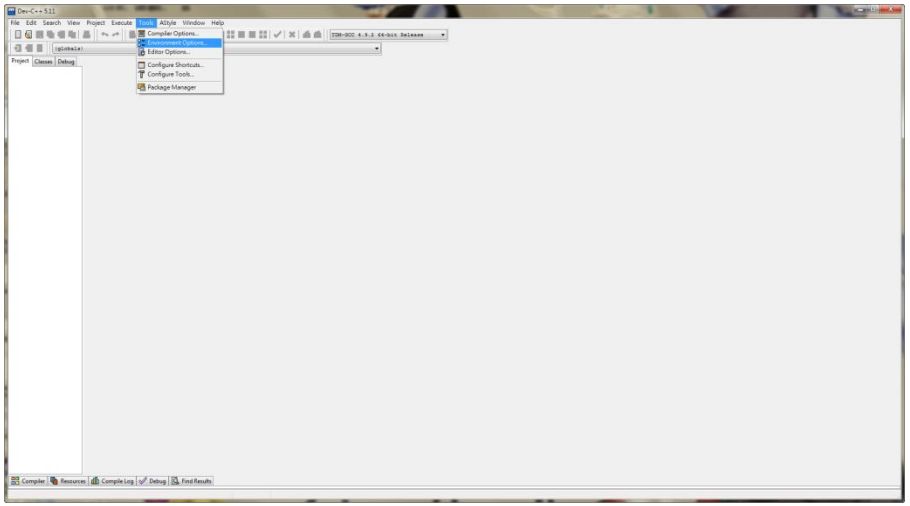


安裝 DEV Cpp

圖示選擇 New Look



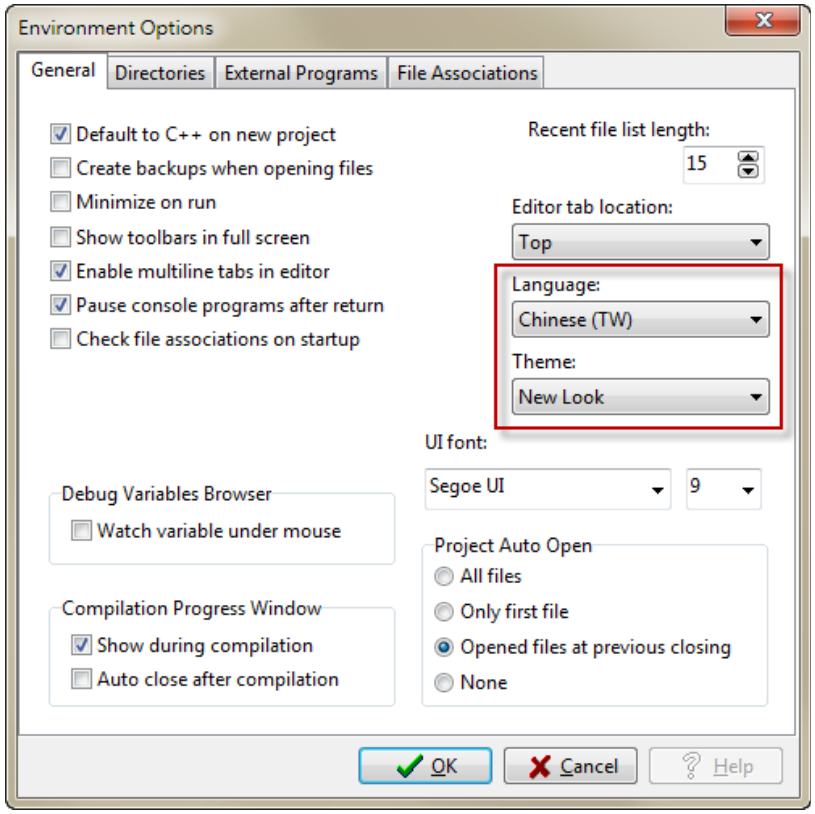
英文介面, 請選擇工具列 Tools>Environment Options



安裝 DEV Cpp

語言選擇 Chinese(TW)

圖示選擇 New Look

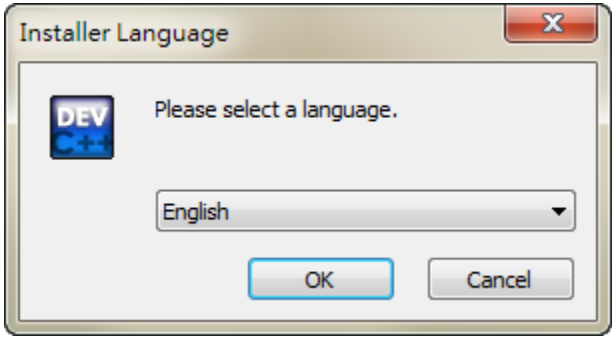


安裝 DEV Cpp

- 取得 DEV Cpp

網址

http://sourceforge.net/projects/orwelldevcpp/



第一個C程式語言

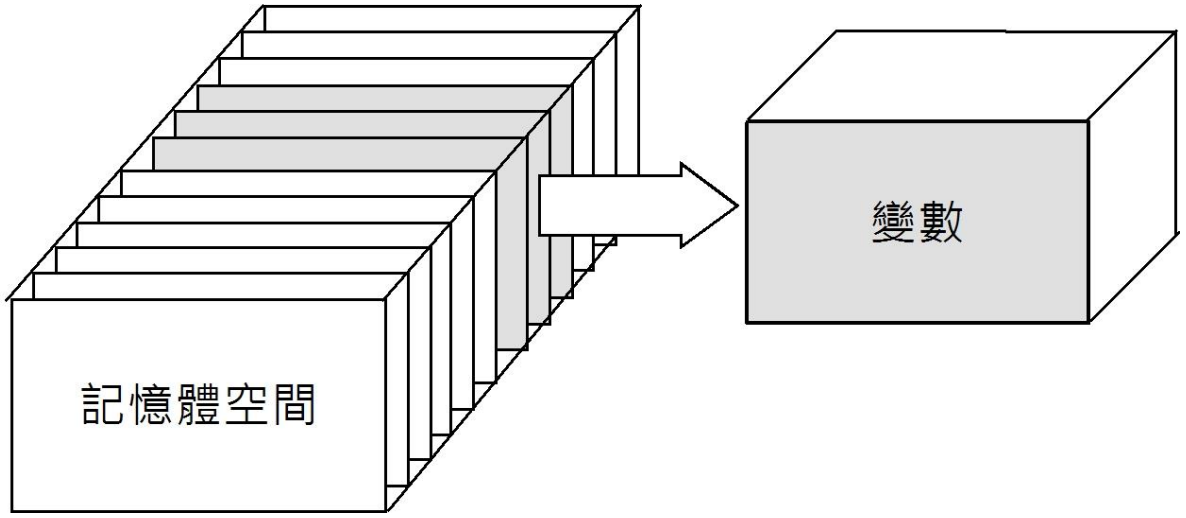
- 以Dev C++ 的環境撰寫第一個C語言：

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
   printf("Hello C!\n");
   printf("Hello World!\n");

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

變數

- 變數需記住的資料是儲存在哪裡，就是電腦的「記憶體」（Memory），變數是一個名稱，用來代表電腦記憶體空間的一個位址



變數

- C程式使用變數前，我們需要2項準備工作，如下所示：
  - 替變數命名，例如：name和height等變數名稱。
  - 指定變數儲存資料的型態，例如：整數和字元等型態。

變數的命名規則

- 變數名稱可以是英文字母、數字或底線
  - 名稱中不能有空白字元
  - 第一個字元不能是數字
  - 不能使用到關鍵字

intel_4x
_AMD
2dos
my dogs
goto

關鍵字（keyword）

- 下表為 C 語言的關鍵字

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

變數的設值方式

- 宣告的時候設值

```
int num=2;              /* 宣告變數，並直接設值 */
```

- 宣告後再設值

```
int num1,num2;          /* 宣告變數 */
char ch;
num1=2;                 /* 將整數變數num1的值設為2 */
num2=30;                /* 將整數變數num2的值設為30 */
ch ='m';                /* 將字元變數ch的值設為'm' */
```

格式特定字元

格式特定字元

接下來，若要輸出某一變數的值，則需要一對應的格式特定字元。首先，要看變數所屬的資料型態是什麼，再利用此資料型態所對應的**格式特定字元（format specified character）**。格式特定字元是以 % 開頭，之後接英文字母。

輸入格式符號	輸入資料型態
%d	整數
%f	浮點數
%c	字元
%s	字串
%e	科學符號
%u	不帶符號10進位整數
%o	8進位整數
%x	16進位整數

程式錯誤的分類

- 語法錯誤（syntax error）
  - 程式含有不合語法的敘述，它無法被編譯程式翻譯

- 語意錯誤（semantic error）
  - 語意錯誤(又稱邏輯錯誤)，就是程式的執行結果非我們所願

語法錯誤

- 下面是有語法錯誤的程式：

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

int main(void)
{
  int num;                                          /* 宣告整數num  /*
  num=2;                                            /* 將num設值為2 /*
  printf("I have %d dogs. \n", num);                //印出字串及變數內容
  printf("You have %d dogs, too. \n, num);          /* 印出字串及變數內容  */
  system("pause")
  return 0;
)
```

語法錯誤

- 下面是有語法錯誤的程式：

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

int main(void)
{
  int num;                    /* 宣告整數num   /*
  num=2;                      /* 將num設值為2 /*
  printf("I have %d dogs. \n", num);          //印出字串及變數內容
  printf("You have %d dogs, too. \n, num);    /* 印出字串及變數內容  */
  system("pause")
  return 0;
)
```

語意錯誤

- 下面是語意錯誤的程式：

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

int main(void)
{
   int num=-2;                /* 宣告整數變數 num，並設值為-2 */

   printf("I have %d dogs.\n", num);
   system("pause");
   return 0;
}
```

提高程式的可讀性

- 程式碼縮排，可提高可讀性

```
02  #include <stdio.h>
03  #include <stdlib.h>
04
05  int main(void)
06  {
07     int i;
08     for(i=1;i<=2;i++)
09     {
10        printf("Cats are running, ");
11        printf("dogs are chasing.\n");
12     }
13     system("pause");
14     return 0;
15  }
```

```
02  #include <stdio.h>
03  #include <stdlib.h>
05  int main(void)
06  {
07  int i;
08  for(i=1;i<=2;i++)
09  {
10  printf("Cats are running, ");
11  printf("dogs are chasing.\n");
12  }
13  system("pause");
14  return 0;
15  }
```

基本資料型態

- C 語言是一種「強調型態」（Strongly Typed）的程式語言，所以變數一定需要宣告使用的資料型態，因為資料型態可以告訴編譯程式變數準備儲存哪種資料和配置多大的記憶體空間。

基本資料型態

- 各種基本資料型態所佔的記憶體空間及範圍：

資料型態		型態說明	位元組	表示範圍
整數類型	long int	長整數	4	-2147483648 到 2147483647
	int	整數	4	-2147483648 到 2147483647
	short int	短整數	2	-32768 到 32767
	char	字元	1	0 到 255 (256 個字元)
浮點數類型	float	浮點數	4	1.2e-38 到 3.4e38
	double	倍精度浮點數	8	2.2e-308 到 1.8e308

整數型態 int

- 整數型態可分為
  - 長整數 (long int)
  - 整數 (int)
  - 短整數 (short int)
- 下面為整數型態宣告的範例：

```
short int sum;          /* 宣告sum為短整數 */
int num=15;             /* 宣告num為整數，並設值為15 */
```

無號整數

- 加上unsigned，整數資料型態便可成為無號整數
  - 無號整數即沒有負數的整數

資料型態	型態說明	位元組	表示範圍
unsigned long int	無號長整數	4	0 到 4294967295
unsigned int	無號整數	4	0 到 4294967295
unsigned short int	無號短整數	2	0 到 65535

```
unsigned int num;          /* 宣告num為無號整數 */
unsigned short int sum;    /* 宣告sum為無號短整數 */
```

溢位 (overflow)

- **溢位**：當儲存的數值超出容許範圍時

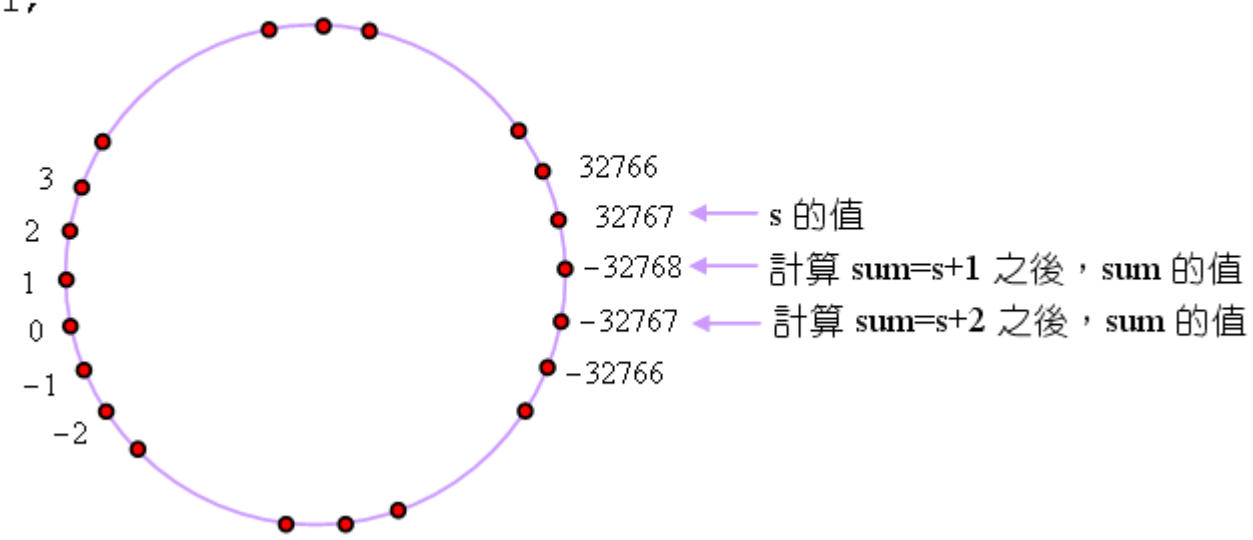
```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
  short int sum,s=32767;                    /* 宣告短整數變數sum與s */
  sum=s+1;
  printf("s+1= %d\n",sum);                  /* 列印出sum的值 */
  sum=s+2;
  printf("s+2= %d\n",sum);                  /* 列印出sum的值 */

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

溢位 (overflow)

- 下圖說明溢位的發生：

```
short sum,s=32767;
sum=s+1;
```



字元型態 char

- 字元型態佔 1 個位元組，用來儲存字元
- 宣告字元變數，並設值給它：

```
char ch;          /* 宣告字元變數ch */
ch='A';           /* 將字元常數'A'設值給字元變數ch */
```

- 在宣告的同時便設定初值

```
char ch='A';      /* 宣告字元變數ch，並將字元常數'A'設值給它 */
char ch=97;       /* 將ch設值為ASCII碼為97的字元 */
char ch='7';      /* 將ch設值為字元常數'7' */
char ch=7;        /* 將ch設值為設值為ASCII碼為7的字元 */
```

跳脫字元

- 反斜線「\」稱為跳脫字元
- 反斜線「\」加上控制碼，稱為跳脫序列

跳脫序列	所代表的意義	十進位 ASCII
\a	警告音(alert)	7
\b	倒退一格(backspace)	8
\n	換行(new line)	10
\r	歸位(carriage return)	13
\0	字串結束字元(null character)	0
\t	跳格(tab)	9
\\	反斜線(backslash)	92
\'	單引號(single quote)	39
\"	雙引號(double quote)	34

浮點數型態 float

- 浮點數佔 4 個位元組

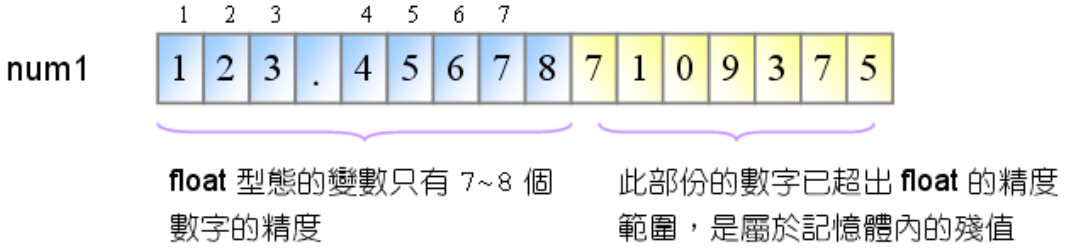
```
float num;                  /* 宣告浮點數變數num */
float num=5.46F;            /* 宣告浮點數變數num，並設值為5.46F */
```

- 要印出浮點數，可用「%f」格式碼

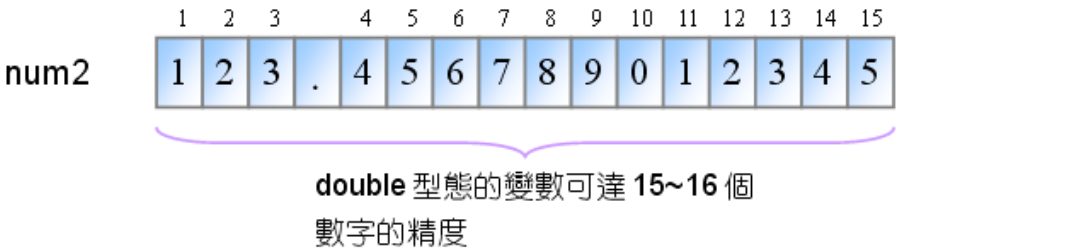
倍精度浮點數型態double

- float 只有 7~8 個位數的精度，double可達 15~16 個位數

```
float num1=123.456789012345F;
```

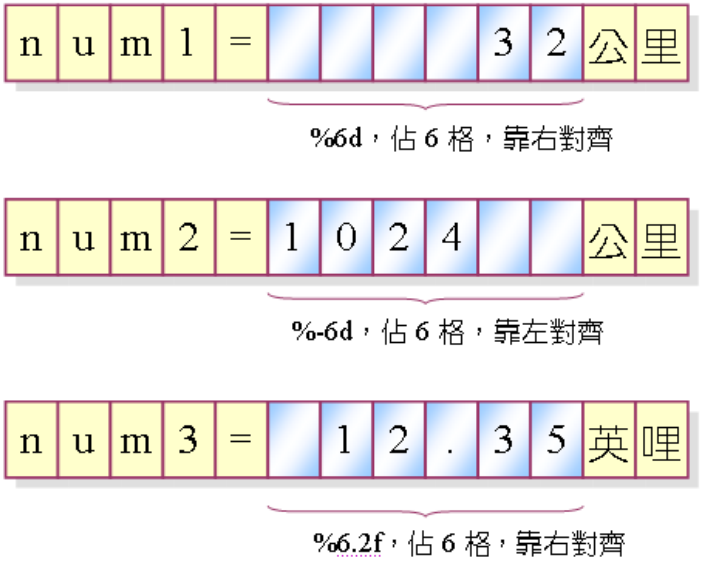



```
double num2=123.456789012345;
```



控制輸出欄位的寬度

- 設定欄位的寬度：




```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
   int num1=32, num2=1024;
   float num3=12.3478f;
   printf( “num1=%6d km\n",num1);      /* 以「%6d」格式印出num1 */
   printf("num2=%-6d km\n",num2);     /* 以「%-6d」格式印出num2 */
   printf("num3=%6.2f mile\n",num3);    /* 以「%6.2f」格式印出num3 */

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

輸入函數scanf()

- 由鍵盤輸入一個整數的範例：

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
   int num;

   printf(" Please input an integer :");
   scanf("%d",&num);                  /* 由鍵盤輸入整數，並指定給num存放 */
   printf("num=%d\n",num);       /* 印出num的內容 */

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

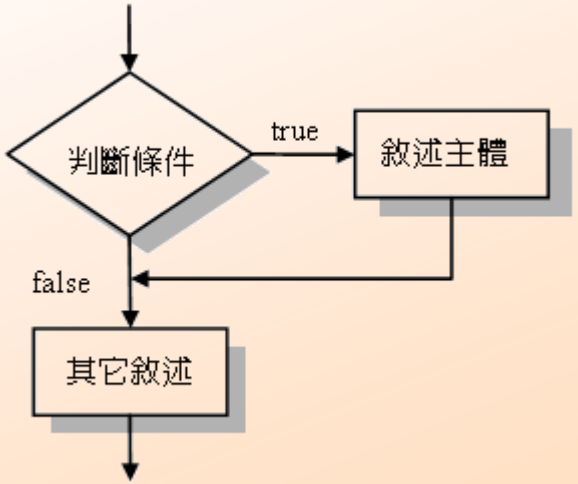
if 敘述的使用

- if 敘述可依據條件式是否成立，來決定程式的流程

if 敘述的語法

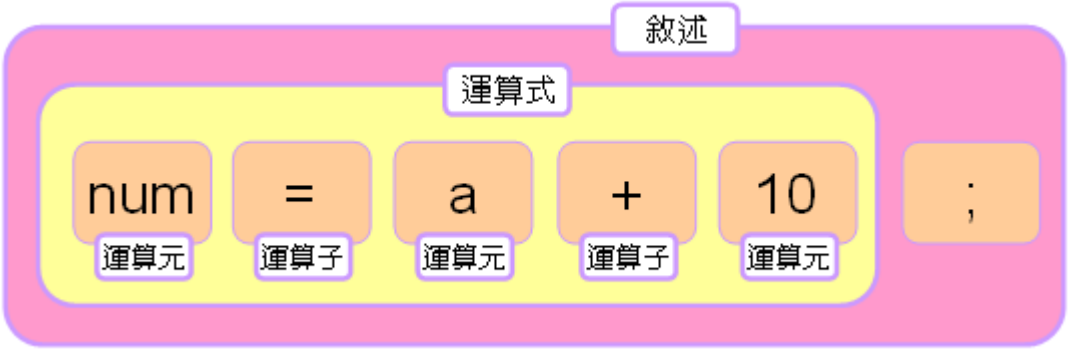
```
if(判斷條件)
{
    敘述 1;
    敘述 2;
      ...
    敘述 n;
}
```

if 敘述的主體



運算式、運算元與運算子

- 運算式由運算元與運算子組成
  - 運算式：expression
  - 運算元：operand，如變數sum，或常數10等
  - 運算子：operator，如「+」、「-」、「*」與「/」等符號



運算式與運算子

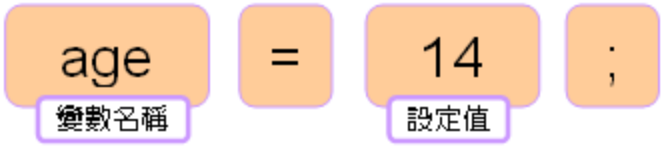
- C 所提供的運算子可分
  - 算數運算子
    - x = a + b;　　// + 運算子
    - y = c * d;　　// * 運算子
  - 邏輯運算子
    - a = x && y;　　//* && 為 AND 運算子的符號
    - b = w || z ;　　// || 為 OR 運算子的符號
  - 關係運算子
    - x >= y　　// >= 代表大於等於
    - z == a　　// == 代表判斷關係上的等於

設定運算子

- 「設定」運算子可將變數設值

設定運算子	意義	範例	說明
=	設定	a=5	設定 a 的值等於 5

- 等號（=）是「設定」的意思，如下面的範例：



一元運算子

- 一元運算子（unary operator）只需要一個運算元
  - +3; /* 表示正3，3 為運算元 */
  - -a; /* 表示負a ，a 為運算元 */
  - !a; /* NOT運算，若a為0，則!a為1，若a不為0，則!a為0*/

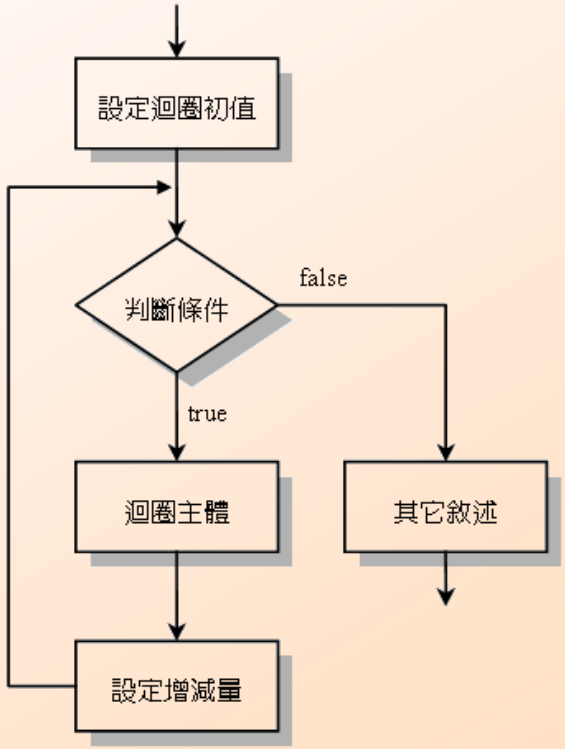
一元運算子	意義	範例	說明
+	正號	a=+5	同 a=5，相當於設定 a 等於正 5
-	負號	a=-3	設定 a 等於-3
!	NOT，否	a=!b	把 b 的值取 NOT，再設給 a 存放

使用for 迴圈

for 迴圈的語法

```
for(設定迴圈初值; 判斷條件; 設定增減量)
{
    迴圈主體;
}
```

這兒不可以加分號



-The End-