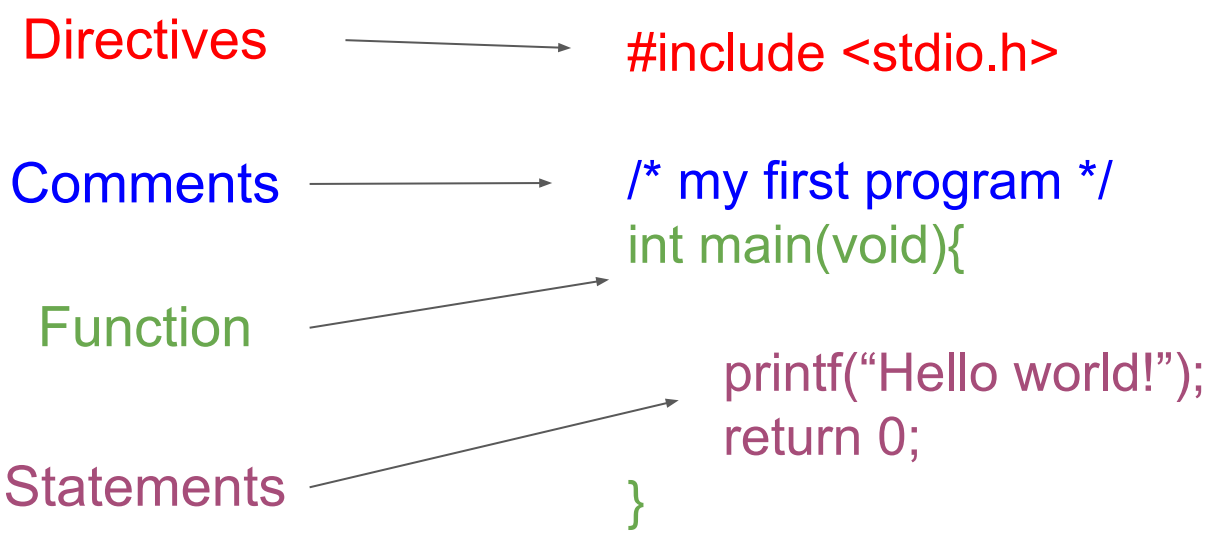
# C語言基本架構

# C程式的基本架構


Directives
#include <stdio.h>
Comments
/* my first program */
int main(void){
Function
printf(“Hello world!”);
return 0;
Statements
}

# Directives

- C語言進行compiled前會先進行preprocess (由preprocessor處理)
- 某些刻意要交給preprocessor執行的指令稱為directives
- e.g. #include <stdio.h>
- <stdio.h> 是個標頭檔(header file)裡面是有關C語言進行I/O的函數庫(library)
- Directives 一定都是 # 開頭
- e.g. #define PI 3.14159f

```
#include <stdio.h>

/* my first program */
int main(void){

        printf(“Hello world!”);
        return 0;
}
```

# Comments (註解)

- 用來幫程式進行補充說明的地方
- 程式在進行compilation的過程中會捨棄掉comments的部分
- C語言Comments可用兩種方式
  - /* here are some comments */ : 可以跨行
  - // here are some comments : 不可以跨行，只能註解一行且從 // 之後開始註解 (C99 新增功能)


```
#include <stdio.h>

/* my first program */
int main(void){

        printf(“Hello world!”);
        return 0;
}
```

# Functions (函式)

- Function 是由一連串 Statements 組合起來，且被賦予名稱
- Library Function 是由先人所事先撰寫好的 function 集合 (standard library 或 third-party libaray) e.g. stdio.h 中的 printf
- Function 裡通常會進行計算並且用 “return” 關鍵字來回傳結果給呼叫該 Function 的 Statement

```
#include <stdio.h>

/* my first program */
int main(void){

        printf(“Hello world!”);
        return 0;
}
```

# The **main** Function

- main 是每個C程式裡**一定要有的**
- 電腦執行 C 程式時會先找main function 裡的內容先開始執行 aka 進入點
- 如同其他 function, main 也會回傳一個值來告訴電腦執行是否成功、或出現異常, 約定成俗規定 0 代表成功, 非 0 值代表出現異常

```
#include <stdio.h>

/* my first program */
int main(void){

        printf(“Hello world!”);
        return 0;
}
```

# Statements

- Statement 是指程式可以在跑的時候所執行的指令
- **C 要求每個 Statements 後面都一定要有分號【;】**
  - 除了有一個例外稱為 Compound Statement (複合型 Statement) aka block
  - Compound statement (block) 是由 { 以及 } 所包圍的又一連串 Statements

```
// Compound statement
{
        statement_1;
        statement_2;
        …. ;
}
```

```
#include <stdio.h>

/* my first program */
int main(void){

        printf(“Hello world!”);
        return 0;
}
```

# 變數 (Variables) 以及賦值 (Assignment)

- 在程式運行的當中可能會需要**暫時地儲存資料**
- 這些在C語言的**儲存位置**稱為變數(Variables)

```
#include <stdio.h>

/* my second program */
int main(void){
        int a;
        a = 3;
        printf(“number of a is : %d”, a);
        return 0;
}
```

# 變數 (Variables) - 型別 Types

- 每個 variable 都會有 type，這些 type 的資訊可以告訴 compiler 以及電腦說這個 variable 它須要多少(預約) memory 空間，以及這些 type 可以進行什麼操作
- 在C語言當中數值方面可以分為: Type Integer(整數型別)、Type Float (浮點數型別)
- 兩者之間最大差別在可不可以儲存小數
- 下一章會在更詳細介紹

# 變數 (Variables) - 宣告 Declarations

- 每個 variable 使用前都必須先 declaration
- Declaration 的用意在告訴電腦說【有這個 variable 的存在，並且幫我留memory 空間】，有點類似現實生活中去餐廳訂位: 顧客姓氏，用餐人數
- 若沒有事先 declare 的動作 compiler 就會回報錯誤，有助於 debug
- C語言中可以在一行內 declare 多個 variables

# 賦值 (Assignment)

- 把值(value)給variable的動作稱為Assignment(賦值)
- 每個variable在被可 assign 的動作前必須先被 declare

<Notice!>

- 在C語言裡Assignment所使用的符號【=】跟數學符號的 "等於" 一樣，但 "等於" 在C語言裡是用【==】
- 因此在C語言遇到【=】的時候讀做 "assign" 或 "賦值"，遇到【==】的時候才念作 "等於"

# 賦值 (Assignment) 與型別 (Type)

- 通常同 type 的內容或變數才能夠 assign
- C 語言允許某些不同的基本 type 能夠 assign 彼此，但可能會與原本內容不一樣的結果
- Assignment 的右手邊可以是個運算式(expression)

```
#include <stdio.h>

/* my second program */
int main(void){
        int a, b, sum;
        a = 3, b = 4;
        sum = a+b+5;
        return 0;
}
```

# Initialization 初始化

- C語言當中在宣告某些 variable 的時候會把 variable 預先設定成 0，**但大部分不會**
- 如果一個 variable 沒有一個預設的值，稱之為 uninitialized (未初始化)
- 因此直接使用這些 uninitialized variable 做計算的話，會出現一些不可預測的結果
- 因此**培養在 declaration 後馬上 initialize 的好習慣，有助於減少 bug 發生**

# Initialization 初始化

- Initialization 的動作可以放在 declaration 的那一行
- 但請注意雖然 Initialization 可放在 declaration 的那一行，但不代表 initialization 是屬於 declaration 的一部分，兩者機制不同 (講到 pointer 時更能夠看出其中差別)

```
#include <stdio.h>

/* my second program */
int main(void){
        int a = 3, b = 4, sum;
        sum = a+b+5;
        return 0;
}
```

# Keywords 關鍵字 (保留字)

- Keywords 指 C 語言會有一些特定的詞是C語言本身會用到的, programmer 不能夠用這些 keywords 當作變數名稱去運用 e.g., 我們不能夠用 int 當作 variable

# Keywords

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

*C99 only

# Identifier 識別字

- Programmer 在程式中幫 variables, functions, macros 以及其他東西取名的名字稱為 Identifier 識別字
- identifier 的名字一開始可以用字母或者是下底線【_】取名
  - 但下底線【_】開始的通常是key word，避免混淆variable名稱還是不要用下底線開始
  - 合法的名字: times10, get_next_char, _done (避免)
- identifier 不要以數字為開頭
  - 不合法的名字: 10times, get-next-char
- C語言是 case-sentitive i.e. C語言用大小寫命名是不一樣的

# 參考資料


- C語言教學手冊(四版) 洪維恩著
- 成大資工程式設計(一)講義　　蔡孟勳教授