程式語言一　講義

程式基本結構

Hanly《C 語言詳論》6e

§ 1　名詞速查表

中文	English	一句話定義	科目	章節
前處理指令	preprocessor directive	`#` 開頭的指令，編譯前先處理，如 `#include`	程式語言一	程式基本結構
標頭檔	header file	宣告函式庫功能的檔案，如 `<stdio.h>` 提供 printf	程式語言一	程式基本結構
進入點	entry point	程式開始執行的地方，C 是 `main()` 第一行	程式語言一	程式基本結構
回傳值	exit status	`main` 的 `return` 值，0 代表正常結束、非 0 代表錯誤	程式語言一	程式基本結構
連結	linking	把你的程式和函式庫實作（如 printf）合併成執行檔	程式語言一	程式基本結構
格式指定符	format specifier	printf/scanf 中決定如何顯示/讀取的符號，如 `%d`、`%f`	程式語言一	程式基本結構
跳脫字元	escape sequence	反斜線開頭的特殊字元，如 `\n` 換行、`\t` Tab	程式語言一	程式基本結構
取址運算子	address-of operator	`&`，取變數位址；scanf 要靠它把讀到的值寫回變數	程式語言一	程式基本結構
未定義行為	undefined behavior	規格未定義的情況（如格式符型別不符），結果不可預測	程式語言一	程式基本結構

§ 2 核心概念

核心概念

一個最小的 C 程式有四個**必備元素**，缺一不可：引入函式庫（`#include`）、定義唯一的 `main()`、在 `main` 裡執行動作、用 `return 0` 告訴系統正常結束。

```c
#include <stdio.h>       // 引入標準輸入輸出函式庫
int main(void) {         // 程式入口：從這裡第一行開始跑
    printf("Hello, World!\n");    // 執行動作：輸出到螢幕
    return 0;             // 回報作業系統：正常結束
}
```

每個程式**有且只有一個** `main`；作業系統從 `main` 的第一行開始執行。這五行你每一行都要能解釋它在做什麼。

§ 3 主要內容

3.1 #include 與前處理

`#` 開頭的指令在編譯**之前**由前處理器處理。`#include <檔>` 的作用是「把該檔內容複製貼上到這裡」。

標頭檔	提供
`<stdio.h>`	printf, scanf, puts, fgets（輸入輸出）
`<math.h>`	sqrt, pow, sin, abs（數學）
`<string.h>`	strlen, strcpy, strcmp（字串）
`<stdlib.h>`	malloc, free, rand, exit（通用）

`< >` 找系統標準路徑；`" "` 先找目前目錄再找系統路徑（自己寫的標頭用這個）。**忘了** `#include <stdio.h>` 直接用 printf，編譯器會警告 `implicit declaration of function 'printf'`（有的警告通過、有的報錯，都是錯寫法）。

3.2 main 與 return

`int main(void)` 是標準寫法（`int main()` 也接受）。`return 0` 代表正常結束、非 0 代表錯誤；終端機 `echo $?` 可查上一個程式的回傳值。C99 後沒寫 `return` 編譯器會自動補 `return 0`，但明確寫出是好習慣。

3.3 編譯的四個階段

`gcc hello.c -o hello` 一步完成四階段。知道**哪個階段抓哪種錯**很有用：

階段	做什麼	在這抓到的錯
前處理 Preprocessing	展開 `#include`／`#define`、移除註解	`gcc -E` 可看結果
編譯 Compilation	C → 組合語言	**語法錯誤**
組譯 Assembly	組合語言 → 機器碼 `.o`	（極少直接報錯）
連結 Linking	合併你的 `.o` 與函式庫	**找不到函式**（undefined reference，如 printf 沒連到）

3.4 註解：說明「為什麼」而非「做什麼」

```
// 單行註解 (C99 起)
/* 多行註解
  可跨好幾行 */
```

好註解解釋**原因**：`// 用海龍公式，因為只知道三邊長`。壞註解只是重複程式碼：`// 把 a 加 1` 配 `a = a + 1;`（沒資訊量）。

3.5 printf 格式化輸出

格式符	型別	範例 → 輸出
`%d`	int	`printf("%d", 42)` → `42`
`%f`	float/double	`printf("%f", 3.14)` → `3.140000`
`%.2f`	float/double	`printf("%.2f", 3.14)` → `3.14`
`%c`	char	`printf("%c", 'A')` → `A`
`%s`	字串	`printf("%s", "Hi")` → `Hi`

跳脫字元：`\n` 換行、`\t` Tab、`\\` 印出反斜線。`%d` 等會被對應引數替換，其餘字元**原樣輸出、不自動加空格**：`printf("A%dB%dC", 1, 2)` → `A1B2C`。

3.6 scanf：從鍵盤讀入

```c
int age;
printf("請輸入你的年齡：");
scanf("%d", &age);      // 變數前要加 & !
printf("你 %d 歲\n", age);
```

`scanf` 的變數前**一定要加** `&`（取址運算子）——scanf 要拿到變數的位址才能把讀到的值寫回去。忘了加是最常見的錯，且編譯器不一定警告，可能直接 crash。格式字串保持簡潔，別亂加 `\n`（`scanf("%d\n", &x)` 會害你要多按一次 Enter）。

3.7 完整範例：BMI 計算機

```c
#include <stdio.h>
int main(void) {
    float height, weight, bmi;
    printf("身高(m)："); scanf("%f", &height);
    printf("體重(kg)："); scanf("%f", &weight);
    bmi = weight / (height * height);
    printf("你的 BMI = %.1f\n", bmi);
    return 0;
}
```

用到：`#include`、`main`、`printf`（`%.1f` 小數一位）、`scanf`（記得 `&`）、變數運算、`return 0`。

3.8 Warning 與 Error

Error 編譯失敗、產不出執行檔（如語法錯、缺 `;`）。**Warning** 能編譯但有疑慮（如格式符型別不符），**不要忽略**——它常是潛在 bug。所以標準編譯指令加 `-Wall`。

§ 4 語法與函式速查

```c
#include <stdio.h>      // 引入函式庫（前處理）
int main(void){ ... return 0; }    // 入口；0=正常結束

printf("格式字串", 引數...);    // 輸出；%d %f %.2f %c %s，\n \t \\
scanf("%d", &x);                 // 讀入；變數前要 &

gcc -E hello.c          // 只看前處理結果
gcc -v hello.c -o hello    // 看完整編譯過程
gcc -Wall -std=c11 hello.c -o hello    // 標準編譯
```

§ 5 常見錯誤

常見錯誤

- 忘了 `#include <stdio.h>` 就用 printf：`implicit declaration` 警告或報錯。
- `scanf` 變數前漏 `&`：可能直接 crash，編譯器常不警告。
- 格式符與型別不符（用 `%d` 印 float）：未定義行為，輸出垃圾值。
- `scanf("%d\n", &x)` 多了 `\n`：害使用者要多按一次 Enter。
- 忽略 warning：warning 常是真 bug，要加 `-Wall` 並清掉。
- 改了程式碼沒存／沒重新 `gcc` 就執行：跑到舊版。
- 註解只重複「做什麼」：沒資訊量，應說「為什麼」。

§ 6 練習題

練習 1（一般題）：預測 printf 輸出

寫出下列各行的輸出（注意空格與跳脫字元）。

```
printf("%d+%d=%d\n", 3, 4, 3+4);
printf("Hello\tWorld\n123");
```

> **引導步驟**
>
> 1. 每個 `%d` 依序被引數替換，其餘字元原樣輸出。
> 2. 第三個引數先算 `3+4`。
> 3. `\t` 是 Tab、`\n` 是換行，不會照字面印。

解答

```
3+4=7
Hello   World
123
```

`%d` 替換成 3、4、7；`\t` 產生 Tab 間距、`\n` 換行後接 `123`。

> **易錯**
>
> · 以為 `%d` 之間會自動加空格（不會）。
> · 把 `\t`、`\n` 當成字面字元印出。

練習 2（一般題）：抓 bug

下列程式為何可能 crash？指出問題並改正。

```
int x;
scanf("%d", x);
printf("%d\n", x);
```

引導步驟

1. scanf 要把讀到的值寫回 `x`，它需要什麼？
2. `x` 沒初始化，把它的值當位址用會怎樣？

解答

`scanf("%d", x)` 漏了 `&`。scanf 要變數的**位址**才能寫回；傳 `x`（未初始化的值）當位址會寫到亂位址，可能 crash。改成 `scanf("%d", &x);`。

易錯

· 以為編譯器會擋下來（不一定會警告）。

練習 3（重要題）：讀程式追蹤

不執行，逐行追蹤並寫出輸出。

```c
#include <stdio.h>
int main(void) {
    int a = 5, b = 2;
    printf("%d\n", a / b);
    printf("%.2f\n", (float)a / b);
    return 0;
}
```

引導步驟

1. `a / b` 兩個 int 相除 → 整數除法，去小數。
2. `(float)a / b`：a 先轉 float，整個運算變浮點。
3. `%.2f` 印到小數兩位。

解答

```
2
2.50
```

`5 / 2` 整數除法得 2（非 2.5）；`(float)5 / 2 = 2.5`，`%.2f` 印成 `2.50`。

易錯

· 以為 `a / b` 會得到 2.5（兩個 int 相除是整數除法）。

練習 4（重要題）：手寫

寫一支程式：讀入兩個整數，印出它們的和與平均（平均印到小數一位）。

> **引導步驟**
>
> 1. `#include <stdio.h>`、`int main(void)`。
> 2. 宣告兩個 int，用 `scanf("%d", &x)` 各讀一個（記得 `&`）。
> 3. 和用 `%d`；平均要 `(a+b)/2.0` 才不是整數除法，用 `%.1f`。

解答

```c
#include <stdio.h>
int main(void) {
    int a, b;
    scanf("%d", &a);
    scanf("%d", &b);
    printf("和 = %d\n", a + b);
    printf("平均 = %.1f\n", (a + b) / 2.0);
    return 0;
}
```

`(a + b) / 2.0` 用 `2.0` 觸發浮點除法，否則整數除法會去掉小數。

易錯

- 平均寫 `(a+b)/2`（整數除法，小數不見）。
- scanf 漏 `&`。

§ 7 自我檢核

- [ ] 能說出最小 C 程式的四個必備元素。
- [ ] 知道 `#include` 在前處理階段做「複製貼上」，分得清 `< >` 與 `" "`。
- [ ] 知道 `return 0` 的意義，main 從第一行開始執行。
- [ ] 能說出編譯四階段，及語法錯（編譯）、找不到函式（連結）各在哪階段。
- [ ] 會用 printf 格式符（%d %f %.2f %c %s）與跳脫字元（\n \t）。
- [ ] 知道 scanf 變數前要加 `&`，並說出原因。
- [ ] 分得清 warning 與 error，知道要加 `-Wall` 且不忽略警告。
- [ ] 知道兩個 int 相除是整數除法，要浮點得轉型或用 `2.0`。