『程式』是什麼?

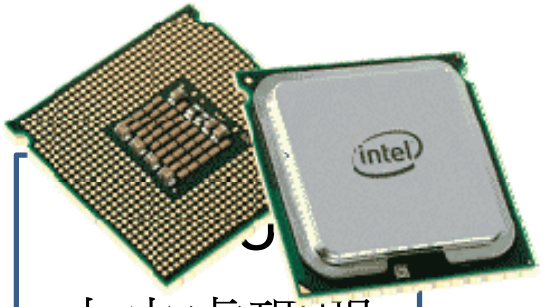
『寫程式』到底在做什麼事?

丁培毅

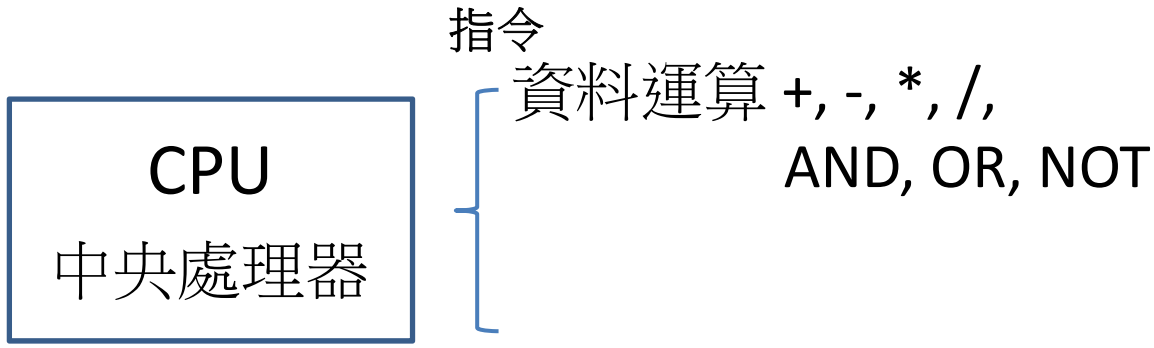
NTOUCSE 程式設計體驗

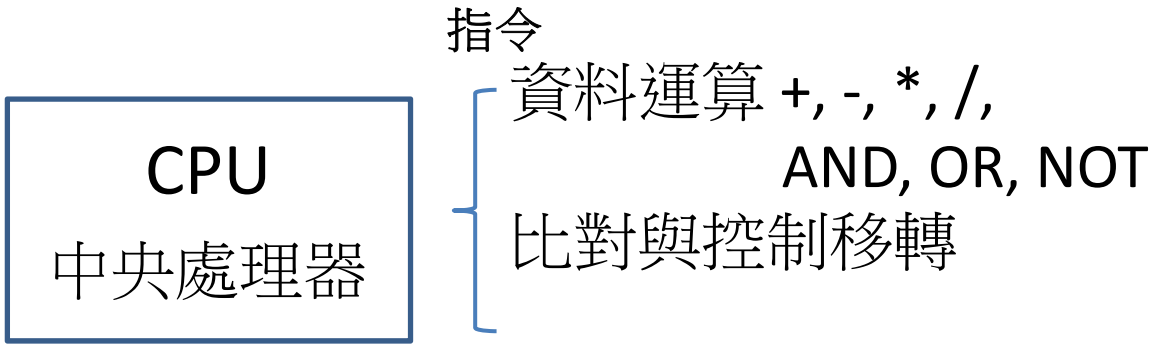
CPU

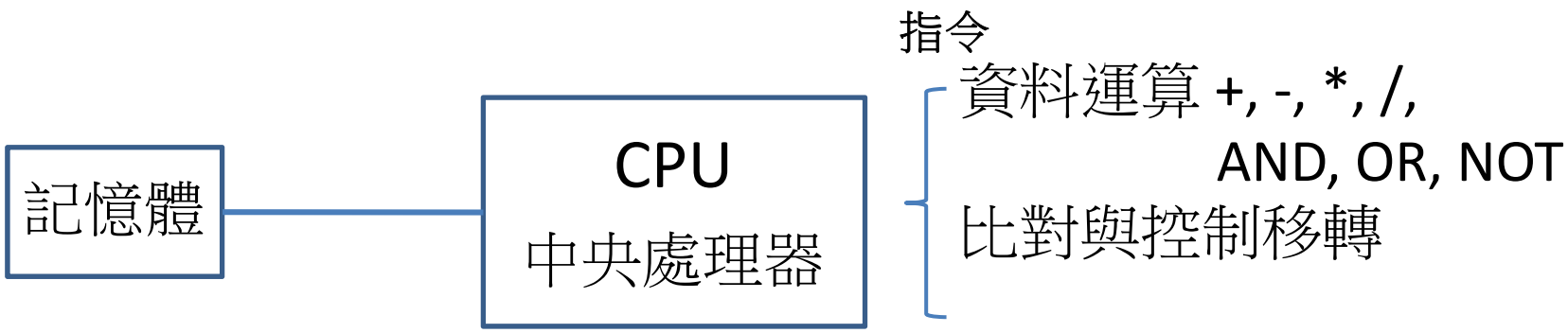
中央處理器

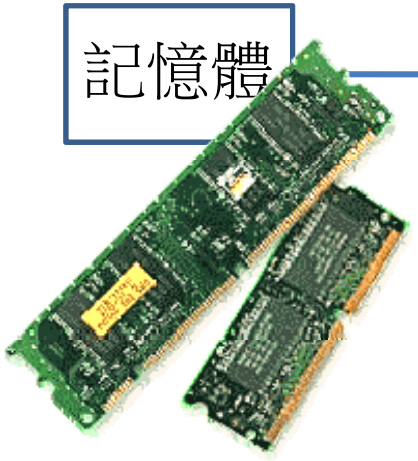


中央處理器


指令
資料運算 +, -, *, /,
AND, OR, NOT
CPU
中央處理器


指令
CPU
中央處理器
資料運算 +, -, *, /,
AND, OR, NOT
比對與控制移轉


記憶體
CPU
中央處理器
指令
資料運算 +, -, *, /,
AND, OR, NOT
比對與控制移轉



記憶體

CPU

中央處理器

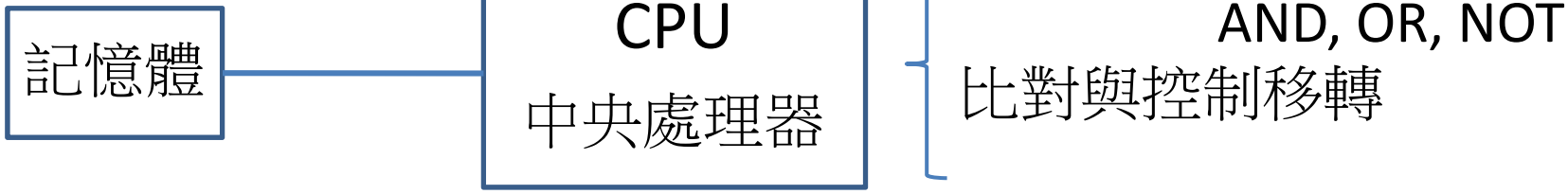
指令

資料運算 +, -, *, /,

AND, OR, NOT

比對與控制移轉

CPU 運作時儲存
資料與程式的裝置



指令
- 資料運算 +, -, *, /,
  AND, OR, NOT
- 比對與控制移轉

CPU 運作時儲存
資料與程式的裝置

指令

- 資料運算 +, -, *, /,
  AND, OR, NOT
- 比對與控制移轉
- 資料搬移

CPU 運作時儲存
資料與程式的裝置

記憶體

CPU
中央處理器

指令
- 資料運算 +, -, *, /, AND, OR, NOT
- 比對與控制移轉
- 資料搬移

CPU 運作時儲存
資料與程式的裝置

記憶體

CPU
中央處理器

指令
- 資料運算 +, -, *, /, AND, OR, NOT
- 比對與控制移轉
- 資料搬移

CPU 運作時儲存
資料與程式的裝置

記憶體

CPU
中央處理器

指令
- 資料運算 +, -, *, /,
  AND, OR, NOT
- 比對與控制移轉
- 資料搬移

CPU 運作時儲存
資料與程式的裝置

記憶體

CPU
中央處理器

指令
- 資料運算 +, -, *, /, AND, OR, NOT
- 比對與控制移轉
- 資料搬移

CPU 運作時儲存
資料與程式的裝置

記憶體

CPU
中央處理器

指令
- 資料運算 +, -, *, /, AND, OR, NOT
- 比對與控制移轉
- 資料搬移

光碟

硬碟

儲存大量資料的裝置

CPU 運作時儲存
資料與程式的裝置

記憶體

CPU
中央處理器

指令
- 資料運算 +, -, *, /, AND, OR, NOT
- 比對與控制移轉
- 資料搬移

光碟

硬碟

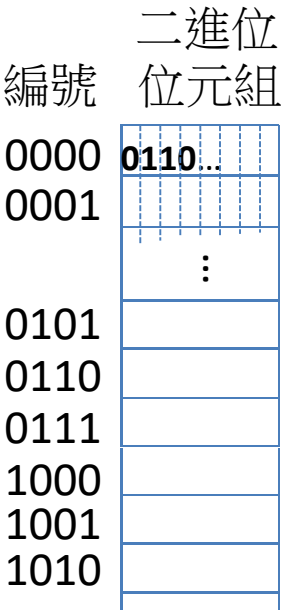
儲存大量資料的裝置

網路

存取別的機器上的資料

資料存放到記憶體/由記憶體取出

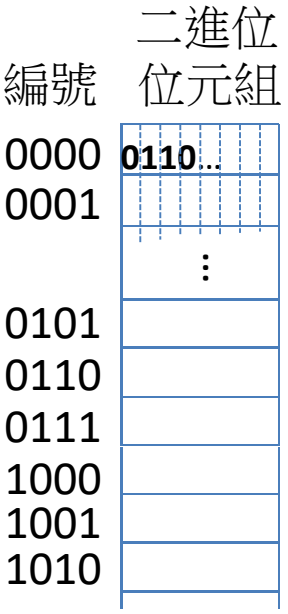
記憶體模型



資料存放到記憶體/由記憶體取出

- 記憶體是 CPU 不可缺少的夥伴, 沒有它的話, CPU 就像是重度失憶症患者, 做完的事、算出來的結果, 沒有一個記得, 其實該怎麼完成任何一件事的步驟也是不記得的, 所有的指令和資料都不知道記在哪裡

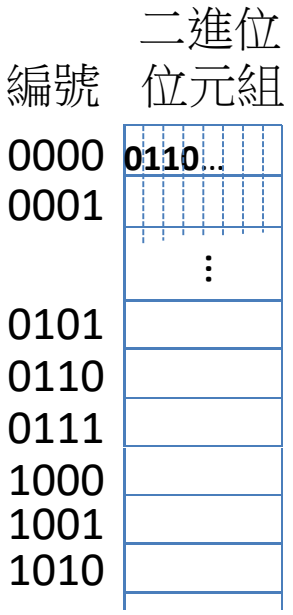
記憶體模型



資料存放到記憶體/由記憶體取出

- 記憶體是 CPU 不可缺少的夥伴, 沒有它的話, CPU 就像是重度失憶症患者, 做完的事、算出來的結果, 沒有一個記得, 其實該怎麼完成任何一件事的步驟也是不記得的, 所有的指令和資料都不知道記在哪裡
- 程式裡看到的記憶體

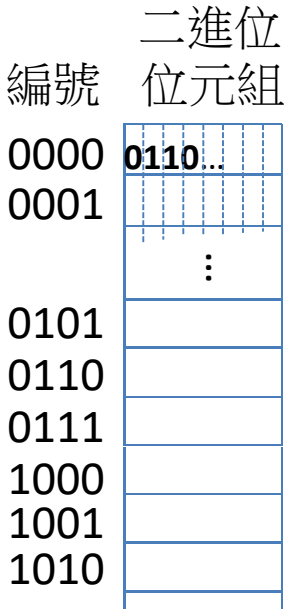
記憶體模型



資料存放到記憶體/由記憶體取出

- 記憶體是 CPU 不可缺少的夥伴, 沒有它的話, CPU 就像是重度失憶症患者, 做完的事、算出來的結果, 沒有一個記得, 其實該怎麼完成任何一件事的步驟也是不記得的, 所有的指令和資料都不知道記在哪裡
- 程式裡看到的記憶體 。。。 變數

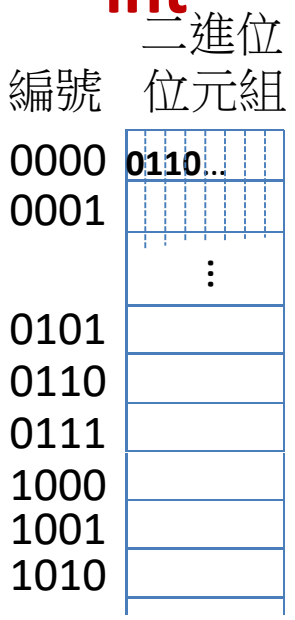
記憶體模型



資料存放到記憶體/由記憶體取出

- 記憶體是 CPU 不可缺少的夥伴, 沒有它的話, CPU 就像是重度失憶症患者, 做完的事、算出來的結果, 沒有一個記得, 其實該怎麼完成任何一件事的步驟也是不記得的, 所有的指令和資料都不知道記在哪裡
- 程式裡看到的記憶體

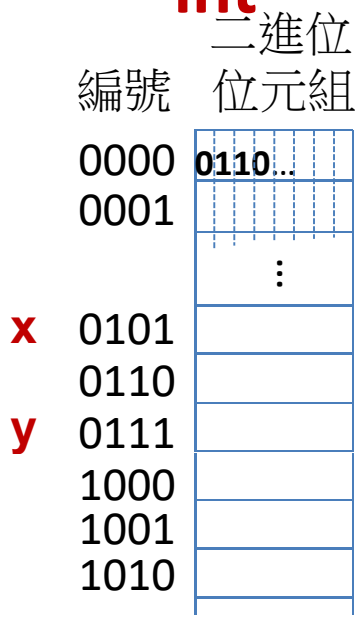
int x, y;　　取名字, 規劃存放格式 ------ **int**



資料存放到記憶體/由記憶體取出

- 記憶體是 CPU 不可缺少的夥伴, 沒有它的話, CPU 就像是重度失憶症患者, 做完的事、算出來的結果, 沒有一個記得, 其實該怎麼完成任何一件事的步驟也是不記得的, 所有的指令和資料都不知道記在哪裡
- 程式裡看到的記憶體

int x, y;　　取名字, 規劃存放格式 ------ **int**



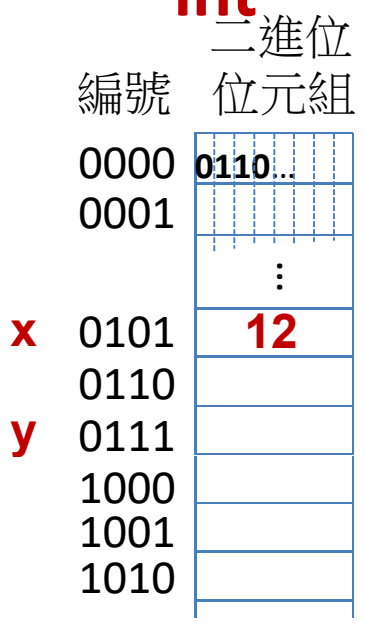
資料存放到記憶體/由記憶體取出

- 記憶體是 CPU 不可缺少的夥伴, 沒有它的話, CPU 就像是重度失憶症患者, 做完的事、算出來的結果, 沒有一個記得, 其實該怎麼完成任何一件事的步驟也是不記得的, 所有的指令和資料都不知道記在哪裡
- 程式裡看到的記憶體

```
int x, y;
x = 12;
```

取名字, 規劃存放格式 ------ **int**

存放資料 ------ **x =**



資料存放到記憶體/由記憶體取出

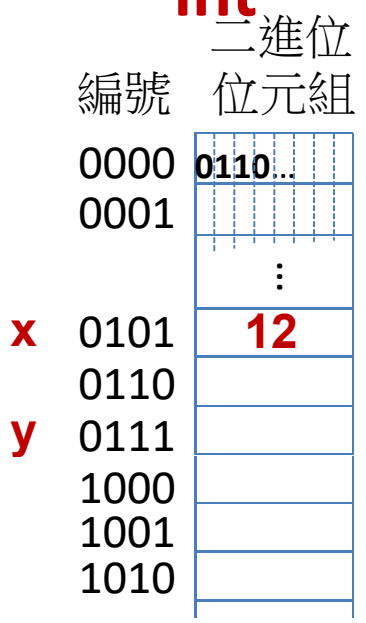
- 記憶體是 CPU 不可缺少的夥伴, 沒有它的話, CPU 就像是重度失憶症患者, 做完的事、算出來的結果, 沒有一個記得, 其實該怎麼完成任何一件事的步驟也是不記得的, 所有的指令和資料都不知道記在哪裡
- 程式裡看到的記憶體

```
int x, y;

x = 12;

y = x + x / 3;
```

取名字, 規劃存放格式 ------ **int**

存放資料 ------ **x =**

取出資料 ------ **x, y**



資料存放到記憶體/由記憶體取出

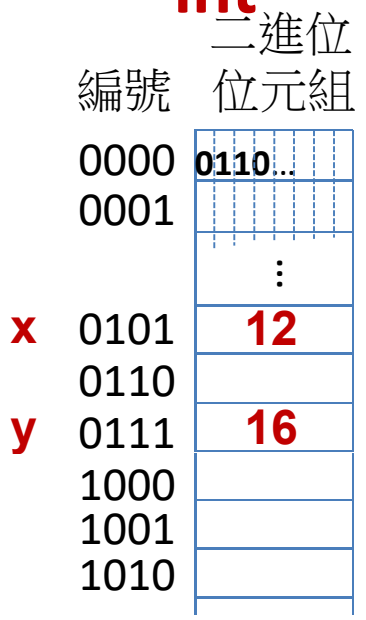
- 記憶體是 CPU 不可缺少的夥伴, 沒有它的話, CPU 就像是重度失憶症患者, 做完的事、算出來的結果, 沒有一個記得, 其實該怎麼完成任何一件事的步驟也是不記得的, 所有的指令和資料都不知道記在哪裡
- 程式裡看到的記憶體

```
int x, y;

x = 12;

y = x + x / 3;
```

取名字, 規劃存放格式 ------ **int**

存放資料 ------ **x =**

取出資料 ------ **x, y**



資料存放到記憶體/由記憶體取出

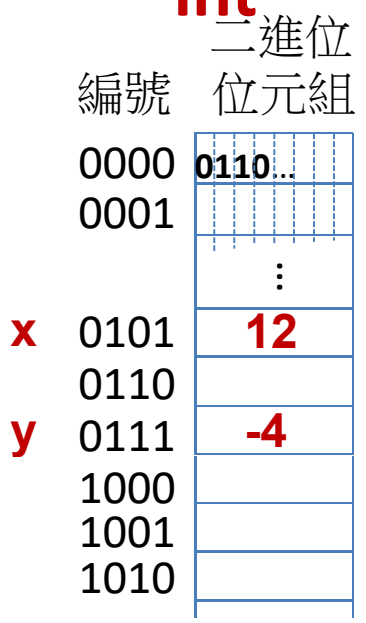
- 記憶體是 CPU 不可缺少的夥伴, 沒有它的話, CPU 就像是重度失憶症患者, 做完的事、算出來的結果, 沒有一個記得, 其實該怎麼完成任何一件事的步驟也是不記得的, 所有的指令和資料都不知道記在哪裡
- 程式裡看到的記憶體

```
int x, y;

x = 12;

y = x + x / 3;

y = x - y;
```

取名字, 規劃存放格式 ------ **int**

存放資料 ------ **x =**

取出資料 ------ **x, y**



資料存放到記憶體/由記憶體取出

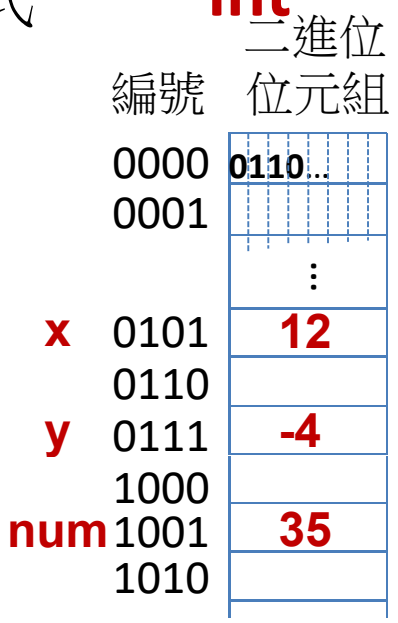
- 記憶體是 CPU 不可缺少的夥伴, 沒有它的話, CPU 就像是重度失憶症患者, 做完的事、算出來的結果, 沒有一個記得, 其實該怎麼完成任何一件事的步驟也是不記得的, 所有的指令和資料都不知道記在哪裡
- 程式裡看到的記憶體

```
int x, y;

x = 12;

y = x + x / 3;

y = x - y;

int num = 35;
```

取名字, 規劃存放格式 ------ **int**

存放資料 ------ **x =**

取出資料 ------ **x, y**



資料存放到記憶體/由記憶體取出

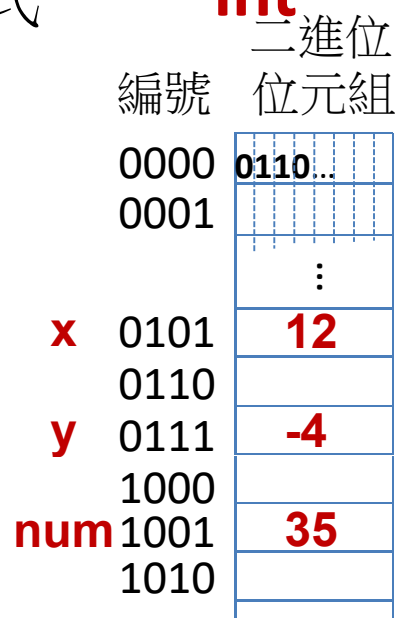
- 記憶體是 CPU 不可缺少的夥伴, 沒有它的話, CPU 就像是重度失憶症患者, 做完的事、算出來的結果, 沒有一個記得, 其實該怎麼完成任何一件事的步驟也是不記得的, 所有的指令和資料都不知道記在哪裡
- 程式裡看到的記憶體

```
int x, y;
x = 12;
y = x + x / 3;
y = x - y;
int num = 35;
printf("我們班上有 %d 位同學\n", num);
```

取名字, 規劃存放格式 ------ **int**

存放資料 ------ **x =**

取出資料 ------ **x, y**



資料存放到記憶體/由記憶體取出

- 記憶體是 CPU 不可缺少的夥伴, 沒有它的話, CPU 就像是重度失憶症患者, 做完的事、算出來的結果, 沒有一個記得, 其實該怎麼完成任何一件事的步驟也是不記得的, 所有的指令和資料都不知道記在哪裡
- 程式裡看到的記憶體

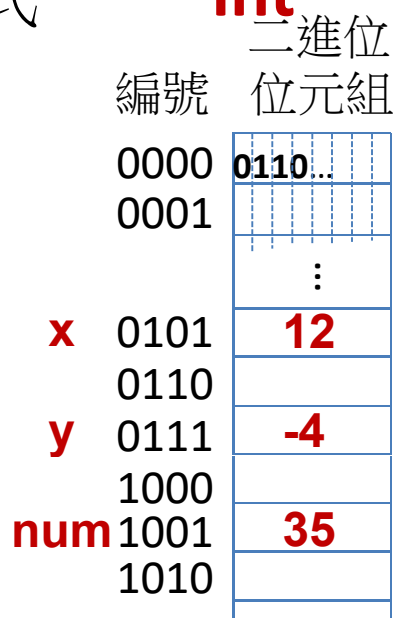
```
int x, y;
x = 12;
y = x + x / 3;
y = x - y;
int num = 35;
printf("我們班上有 %d 位同學\n", num);
```

取名字, 規劃存放格式 ------ **int**

存放資料 ------ **x =**

取出資料 ------ **x, y**

x, y, num 稱為變數



要求 **CPU** 把資料顯示在螢幕上

要求 **CPU** 把資料顯示在螢幕上

```
printf("Hello World!\n");
```

要求 **CPU** 把資料顯示在螢幕上

```
#include <stdio.h>


    printf("Hello World!\n");
```

要求 **CPU** 把資料顯示在螢幕上

```
#include <stdio.h>


    printf("                    ");
```

要求 **CPU** 把資料顯示在螢幕上

```
#include <stdio.h>


    printf("Hello World!\n");
```

要求 **CPU** 把資料顯示在螢幕上

```
#include <stdio.h>


    printf("Hello World!\n");
    printf("My name is Bob.\n");
```

要求 CPU 把資料顯示在螢幕上

```
#include <stdio.h>

int main()
{
    printf("Hello World!\n");
    printf("My name is Bob.\n");
    return 0;
}
```

要求 CPU 把資料顯示在螢幕上

```
#include <stdio.h>

int main()
{
    printf("Hello World!\n");
    printf("My name is Bob.\n");
    return 0;
}
```

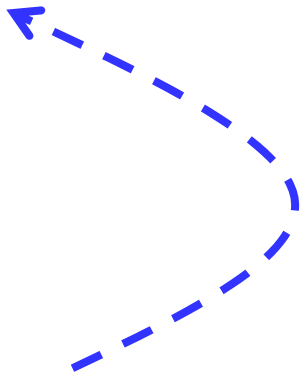
要求 CPU 把資料顯示在螢幕上

```
#include <stdio.h>

int main()
{
    printf("Hello World!\n");
    printf("My name is Bob.\n");
    return 0;
}
```

```
q:\>testPrintf
Hello World!
My name is Bob.
```

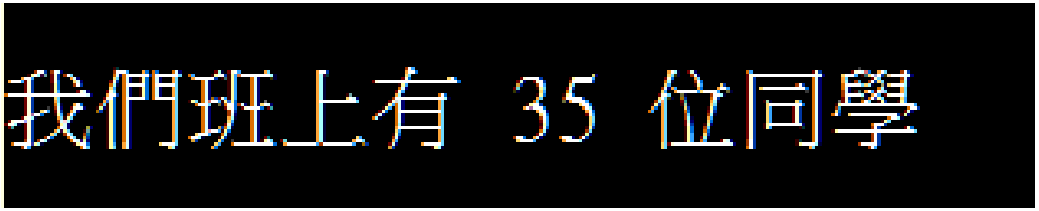
要求 CPU 把資料顯示在螢幕上

```
#include <stdio.h>

int main()
{


    return 0;
}
    int num = 35;
    printf("我們班上有 %d 位同學\n", num);
```



要求 CPU 把資料顯示在螢幕上

```
#include <stdio.h>

int main()
{


    return 0;
}
    int num = 35;
    printf("我們班上有 %d 位同學\n", num);
```



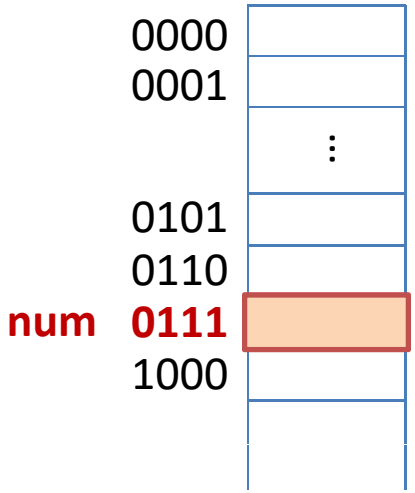
要求 CPU 由鍵盤讀取資料

- 操作者由鍵盤輸入資料以後, CPU 把資料依照指定的格式放進記憶體 (變數) 裡

要求 CPU 由鍵盤讀取資料

- 操作者由鍵盤輸入資料以後, CPU 把資料依照指定的格式放進記憶體 (變數) 裡

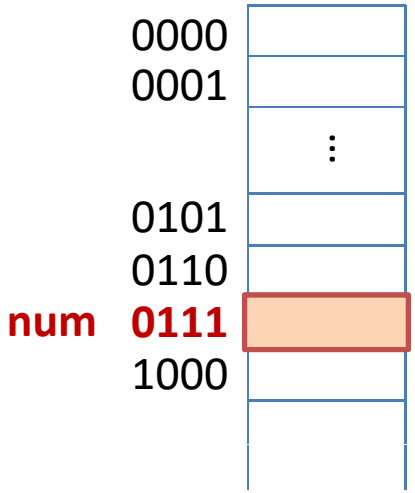
```
int num;
```



要求 CPU 由鍵盤讀取資料

- 操作者由鍵盤輸入資料以後, CPU 把資料依照指定的格式放進記憶體 (變數) 裡

```
int num;
scanf("%d", &num);
```

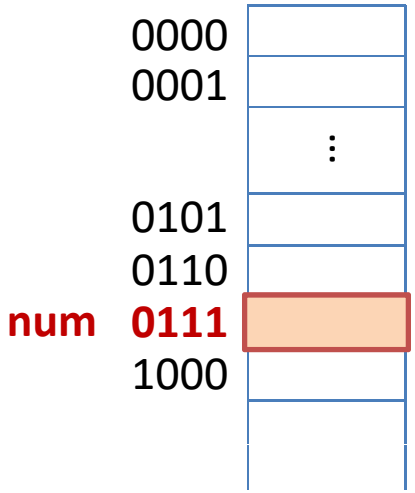


要求 CPU 由鍵盤讀取資料

- 操作者由鍵盤輸入資料以後, CPU 把資料依照指定的格式放進記憶體 (變數) 裡

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

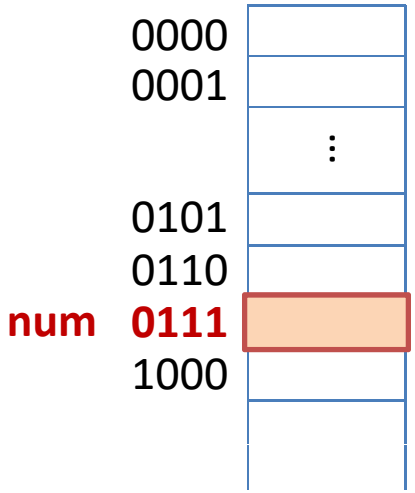
```
int num;
scanf("%d", &num);
```



要求 CPU 由鍵盤讀取資料

- 操作者由鍵盤輸入資料以後, CPU 把資料依照指定的格式放進記憶體 (變數) 裡

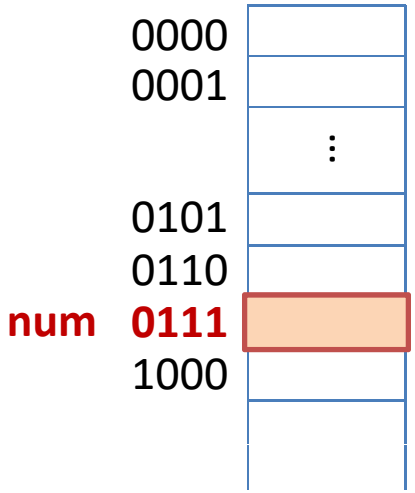
```
#include <stdio.h>
int main()
{
    int num;
    scanf("%d", &num);
    return 0;
}
```



要求 CPU 由鍵盤讀取資料

- 操作者由鍵盤輸入資料以後, CPU 把資料依照指定的格式放進記憶體 (變數) 裡

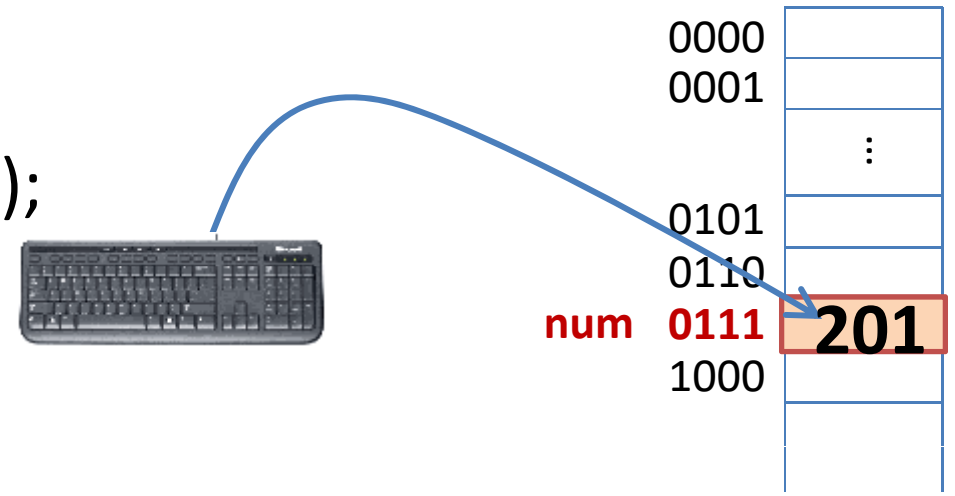
```
#include <stdio.h>
int main()
{
    int num;
    scanf("%d", &num);
    return 0;
}
```



要求 CPU 由鍵盤讀取資料

- 操作者由鍵盤輸入資料以後, CPU 把資料依照指定的格式放進記憶體 (變數) 裡

```
#include <stdio.h>
int main()
{
    int num;
    scanf("%d", &num);
    return 0;
}
```



CPU 執行算術與邏輯運算

CPU 執行算術與邏輯運算

- +, -, *, /, AND, OR, NOT, XOR

CPU 執行算術與邏輯運算

- +, -, *, /, AND, OR, NOT, XOR

```
int x = 13, y = 3;
y = x + x / y;
y = x - y;
```

CPU 執行算術與邏輯運算

- +, -, *, /, AND, OR, NOT, XOR

```
int x = 13, y = 3;
y = x + x / y;
y = x - y;
```

整數除法: 商為 4

CPU 執行算術與邏輯運算

- +, -, *, /, AND, OR, NOT, XOR

```
int x = 13, y = 3;
y = x + x / y;
y = x - y;
```

整數除法: 商為 4 , 先乘除後加減

CPU 執行算術與邏輯運算

- +, -, *, /, AND, OR, NOT, XOR

```
int x = 13, y = 3;
y = x + x / y;
y = x - y;
```

整數除法: 商為 4 , 先乘除後加減

浮點數

```
double r, s = 12.5;
r = s * s;
```

CPU 搬動資料

CPU 搬動資料

- 記憶體和記憶體之間, 記憶體和暫存器之間

CPU 搬動資料

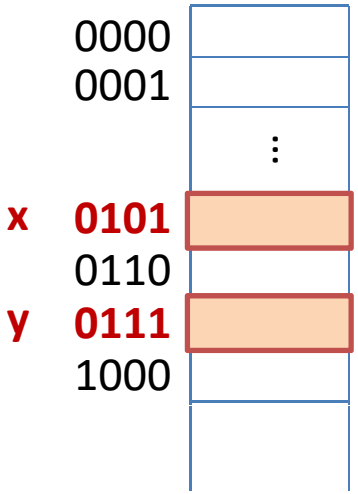
- 記憶體和記憶體之間, 記憶體和暫存器之間

```
int x = 13, y = 3;
y = x + x / y;
```

CPU 搬動資料

- 記憶體和記憶體之間, 記憶體和暫存器之間

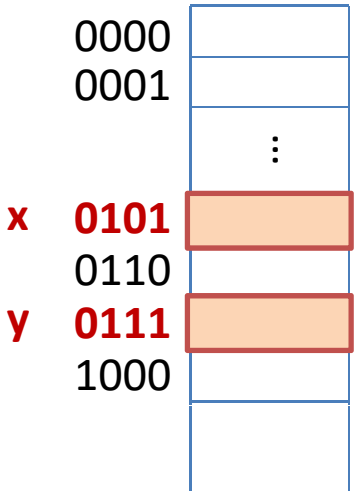
```
int x = 13, y = 3;
y = x + x / y;
```



CPU 搬動資料

- 記憶體和記憶體之間, 記憶體和暫存器之間

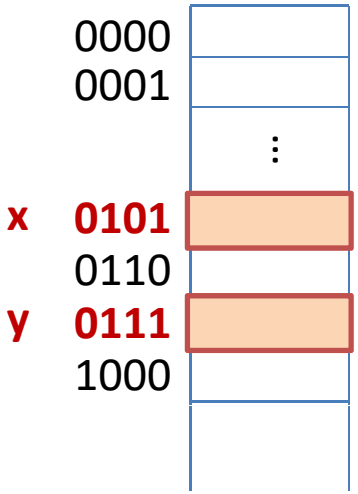
```
int x = 13, y = 3;
y = x + x / y;
```

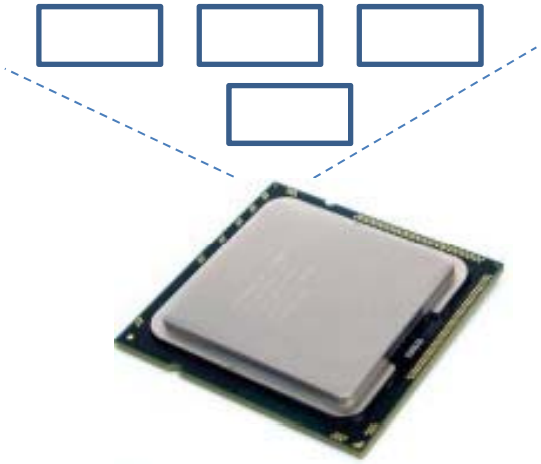


CPU 搬動資料

- 記憶體和記憶體之間, 記憶體和暫存器之間

```
int x = 13, y = 3;
y = x + x / y;
```

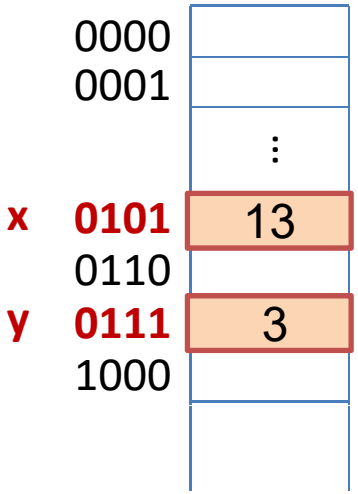


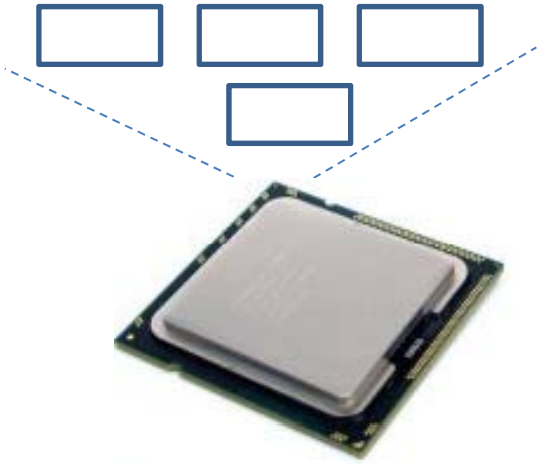


CPU 搬動資料

- 記憶體和記憶體之間, 記憶體和暫存器之間

```
int x = 13, y = 3;
y = x + x / y;
```

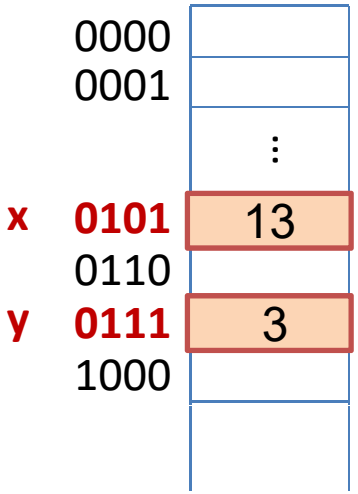


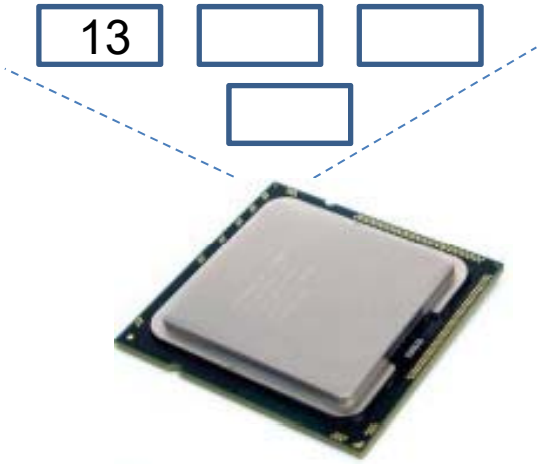


CPU 搬動資料

- 記憶體和記憶體之間, 記憶體和暫存器之間

```
int x = 13, y = 3;
y = x + x / y;
```

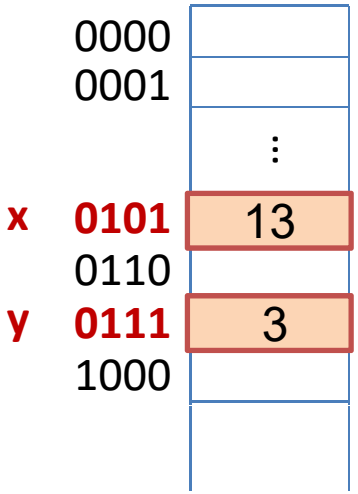


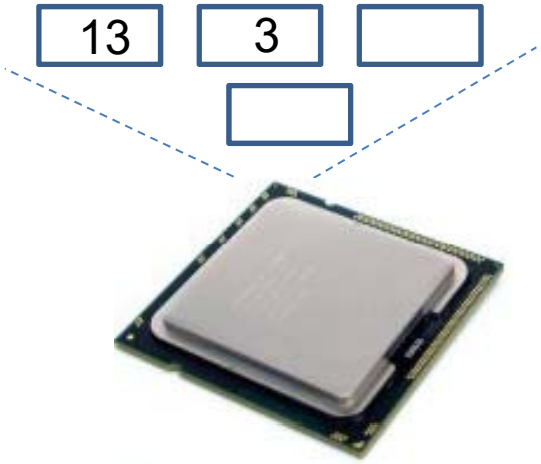


CPU 搬動資料

- 記憶體和記憶體之間, 記憶體和暫存器之間

```
int x = 13, y = 3;
y = x + x / y;
```

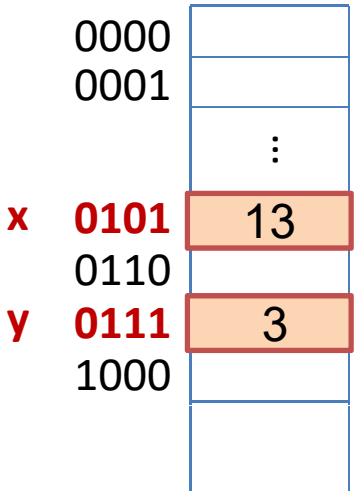


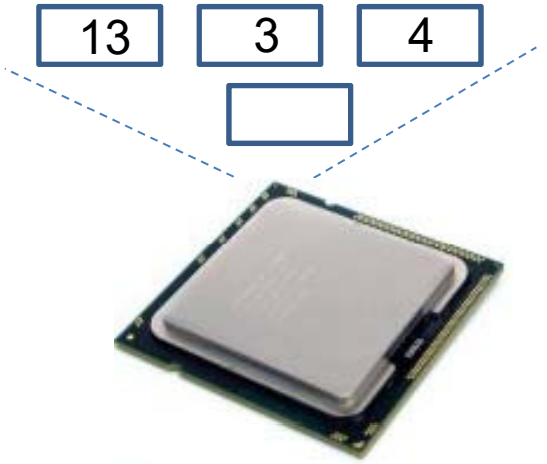


CPU 搬動資料

- 記憶體和記憶體之間, 記憶體和暫存器之間

```
int x = 13, y = 3;
y = x + x / y;
```

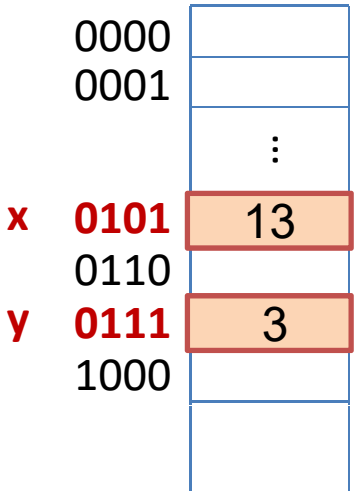


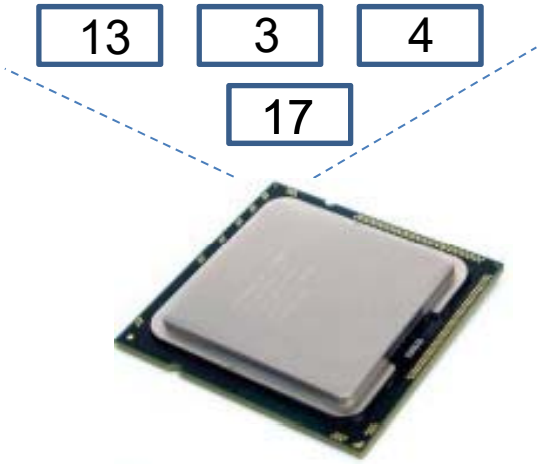


CPU 搬動資料

- 記憶體和記憶體之間, 記憶體和暫存器之間

```
int x = 13, y = 3;
y = x + x / y;
```

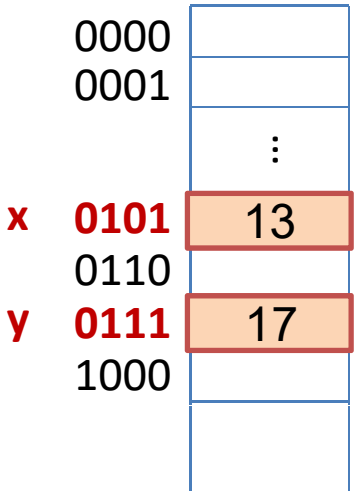


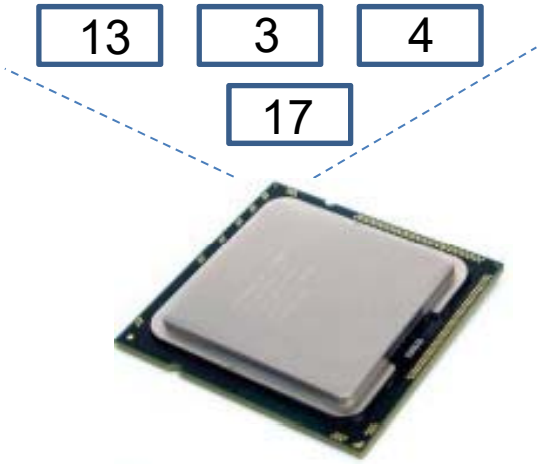


CPU 搬動資料

- 記憶體和記憶體之間, 記憶體和暫存器之間

```
int x = 13, y = 3;
y = x + x / y;
```

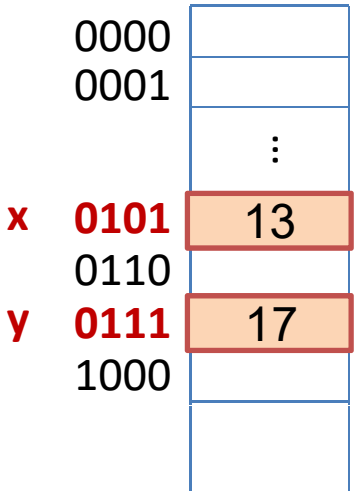


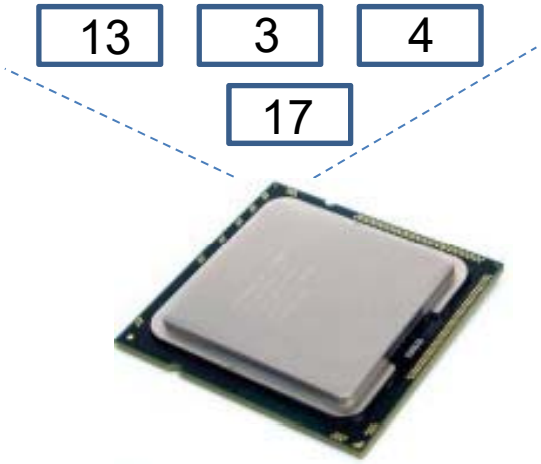


CPU 搬動資料

- 記憶體和記憶體之間, 記憶體和暫存器之間

```
int x = 13, y = 3;
y = x + x / y;
```





- 記憶體和硬體裝置 (硬碟、鍵盤、網路...) 之間

測試比對與控制移轉

測試比對與控制移轉

- CPU 自己沒有智慧, 但是下面這段程式裡面有智慧行為裡最基本的 條件判斷

測試比對與控制移轉

- CPU 自己沒有智慧, 但是下面這段程式裡面有智慧行為裡最基本的 條件判斷

```
int x = 13, y = 3;
y = x + x / y;
y = x - y;
if (y < 0) {
    printf("y 存放的數值是負數\n");
}
else {
    printf("y 存放的數值大於或等於 0\n");
}
```

C 程式架構

- 每一個 C 程式裡都一定有一個 **main** 函式

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

C 程式架構

- 每一個 C 程式裡都一定有一個 **main** 函式

```
#include <stdio.h>
int main()
{

    return 0;
}
```

C 程式架構

- 每一個 C 程式裡都一定有一個 **main** 函式

```
#include <stdio.h>
int main()
{

    return 0;
}
```

C 程式架構

- 每一個 C 程式裡都一定有一個 **main** 函式

```
#include <stdio.h>
int main()
{
    printf("Hello World!\n");
    return 0;
}
```

C 程式架構

- 每一個 C 程式裡都一定有一個 **main** 函式

```
#include <stdio.h>
int main()
{
    printf("Hello World!\n");
    return 0;
}
```

大括號裡面就放你要求 CPU 執行的動作序列

C 程式架構

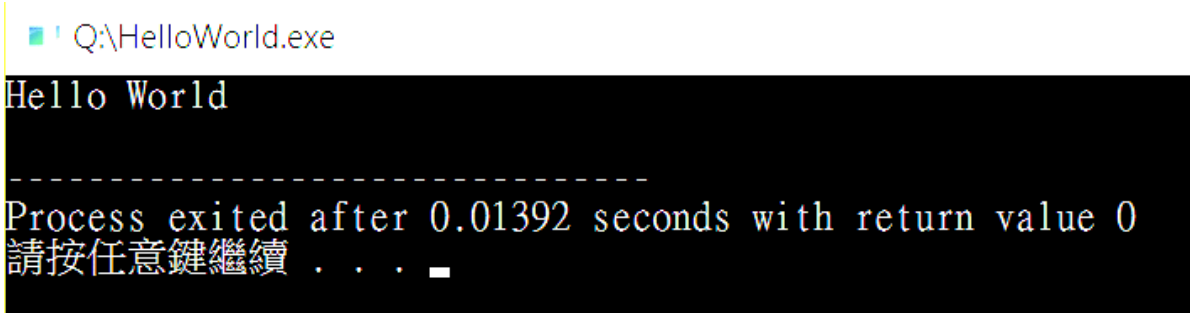
- 每一個 C 程式裡都一定有一個 **main** 函式

```
#include <stdio.h>
int main()
{
    printf("Hello World!\n");
    return 0;
}
```

大括號裡面就放你要求 CPU 執行的動作序列

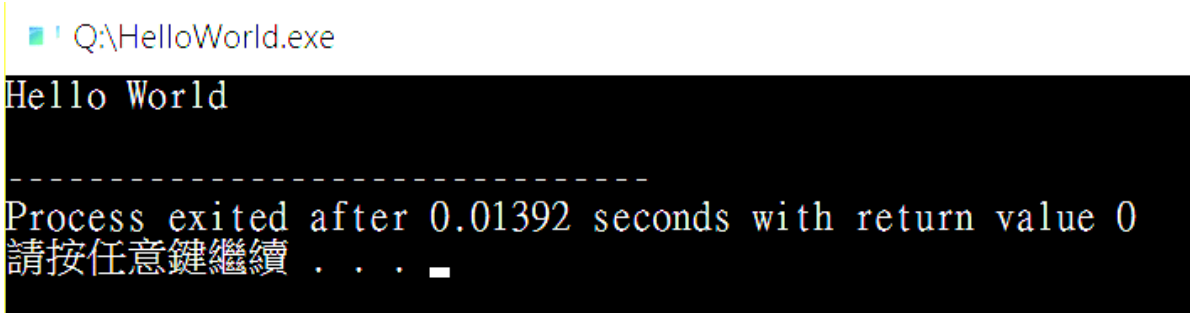
程式練習一: Hello World

- 請寫一個 C 程式, 在螢幕上列印出 Hello World



程式練習一: Hello World

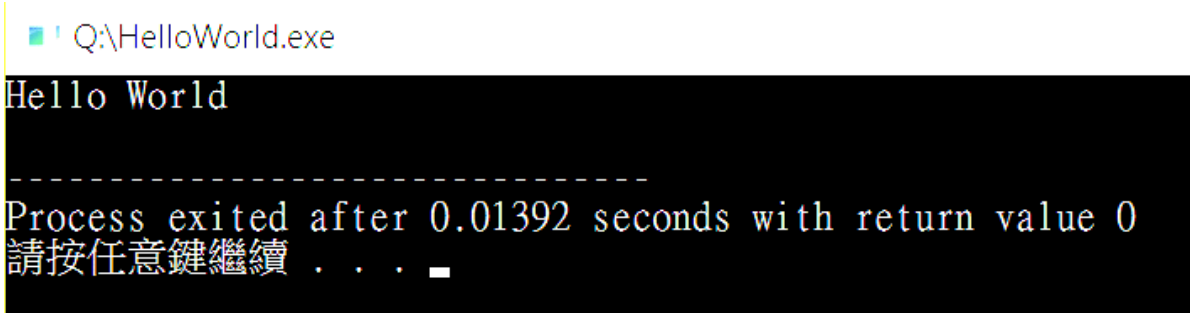
- 請寫一個 C 程式, 在螢幕上列印出 Hello World



- 請注意練習

程式練習一: Hello World

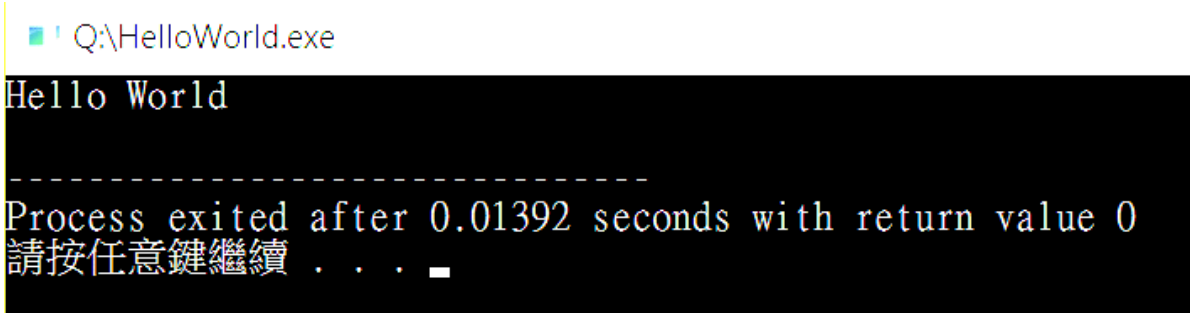
- 請寫一個 C 程式, 在螢幕上列印出 Hello World



- 請注意練習
  - DevCpp 程式開發環境

程式練習一: Hello World

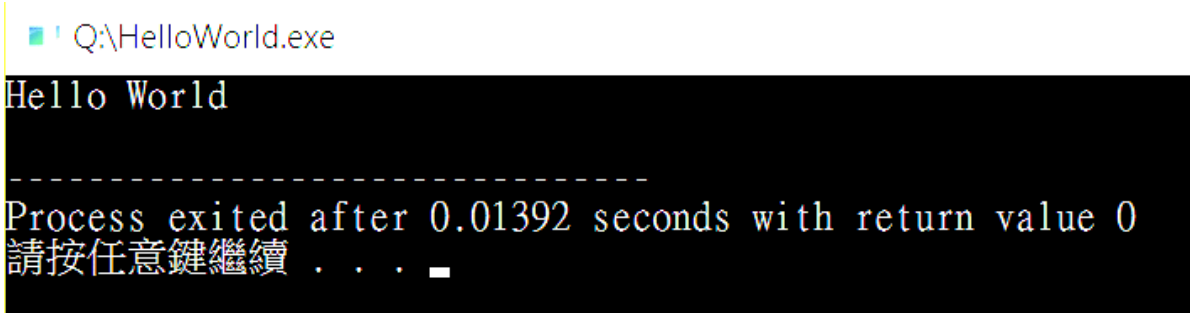
- 請寫一個 C 程式, 在螢幕上列印出 Hello World



- 請注意練習
  - DevCpp 程式開發環境
    - 開新... 原始碼

程式練習一: Hello World

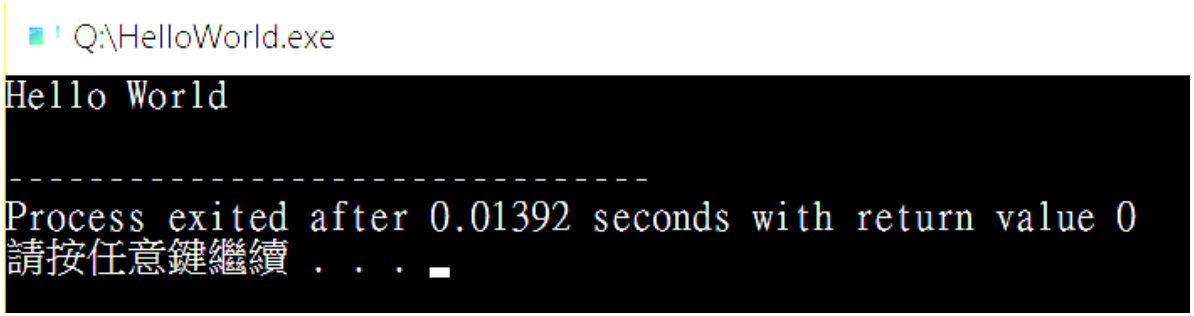
- 請寫一個 C 程式, 在螢幕上列印出 Hello World



- 請注意練習
  - DevCpp 程式開發環境
    - 開新... 原始碼
    - 執行 (編譯並執行)

程式練習一: Hello World

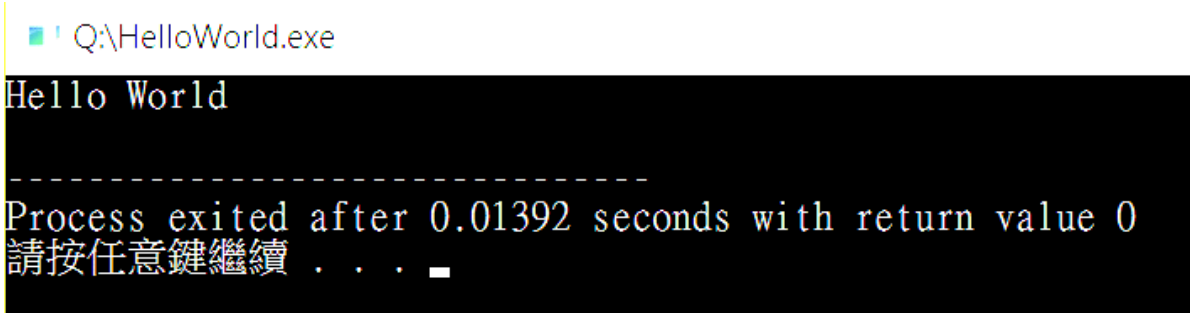
- 請寫一個 C 程式, 在螢幕上列印出 Hello World



- 請注意練習
  - DevCpp 程式開發環境
    - 開新... 原始碼
    - 執行 (編譯並執行)
  - 練習 e-Tutor 繳交與評測環境

程式練習一: Hello World

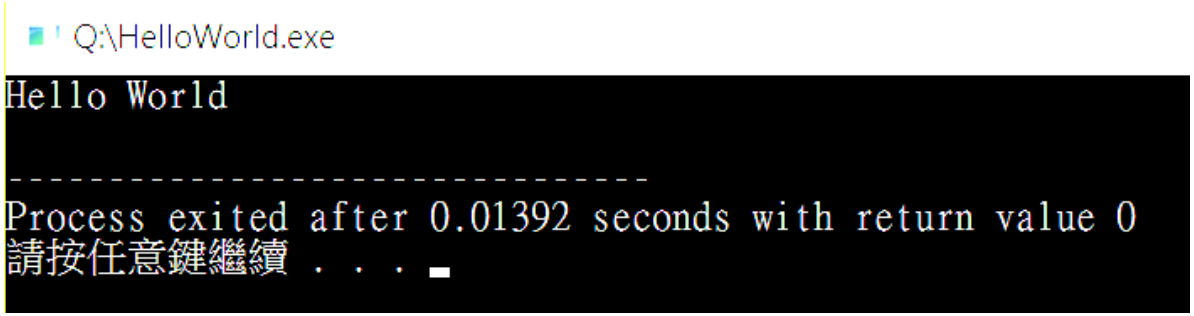
- 請寫一個 C 程式, 在螢幕上列印出 Hello World



- 請注意練習
  - DevCpp 程式開發環境
    - 開新... 原始碼
    - 執行 (編譯並執行)
  - 練習 e-Tutor 繳交與評測環境
  - 記住基本的 C 程式架構

程式練習一: Hello World

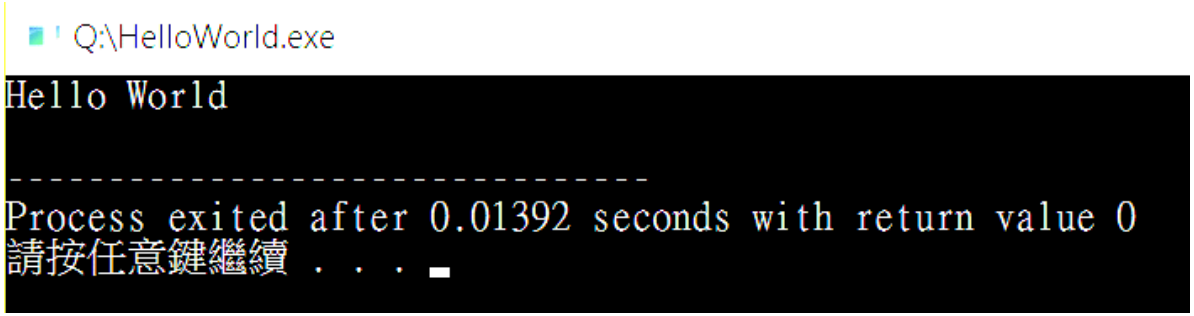
- 請寫一個 C 程式, 在螢幕上列印出 Hello World



- 請注意練習
  - DevCpp 程式開發環境
    - 開新... 原始碼
    - 執行 (編譯並執行)
  - 練習 e-Tutor 繳交與評測環境
  - 記住基本的 C 程式架構
    - main 函式

程式練習一: Hello World

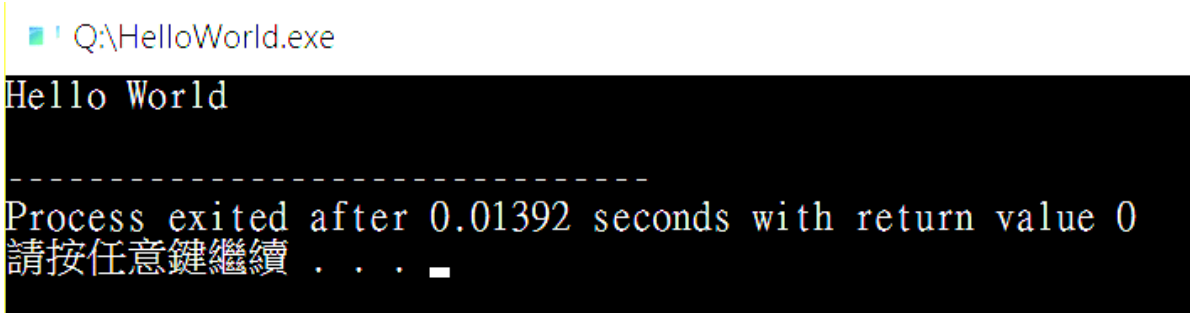
- 請寫一個 C 程式, 在螢幕上列印出 Hello World



- 請注意練習
  - DevCpp 程式開發環境
    - 開新... 原始碼
    - 執行 (編譯並執行)
  - 練習 e-Tutor 繳交與評測環境
  - 記住基本的 C 程式架構
    - main 函式
    - return 0; 敘述

程式練習一: Hello World

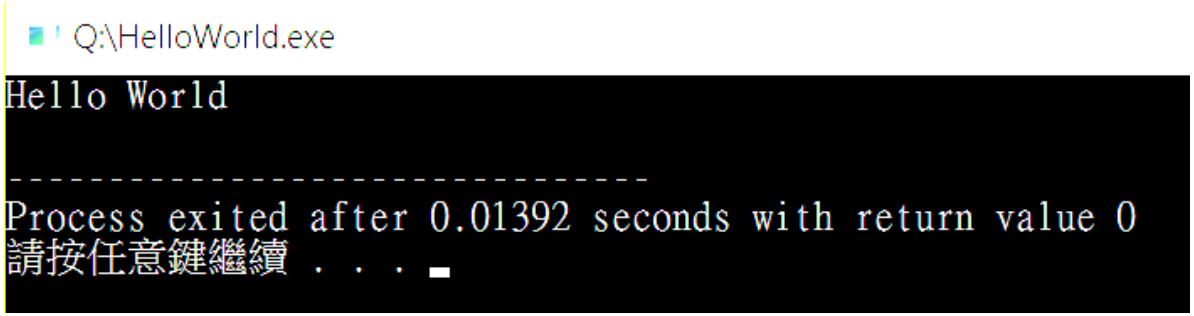
- 請寫一個 C 程式, 在螢幕上列印出 Hello World



- 請注意練習
  - DevCpp 程式開發環境
    - 開新... 原始碼
    - 執行 (編譯並執行)
  - 練習 e-Tutor 繳交與評測環境
  - 記住基本的 C 程式架構
    - main 函式
    - return 0; 敘述
    - printf(...); 在螢幕輸出文字的工具

程式練習一: Hello World

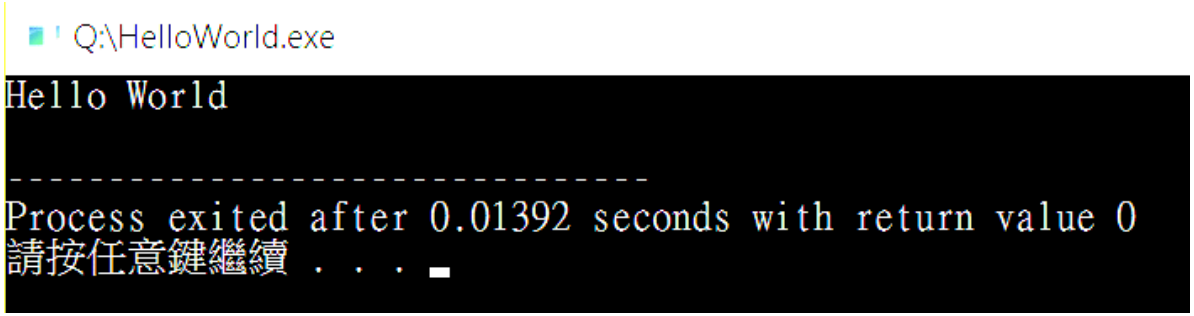
- 請寫一個 C 程式, 在螢幕上列印出 Hello World



- 請注意練習
  - DevCpp 程式開發環境
    - 開新... 原始碼
    - 執行 (編譯並執行)
  - 練習 e-Tutor 繳交與評測環境
  - 記住基本的 C 程式架構
    - main 函式
    - return 0; 敘述
    - printf(...); 在螢幕輸出文字的工具
    - #include <...> 前處理器指令

程式練習一: Hello World

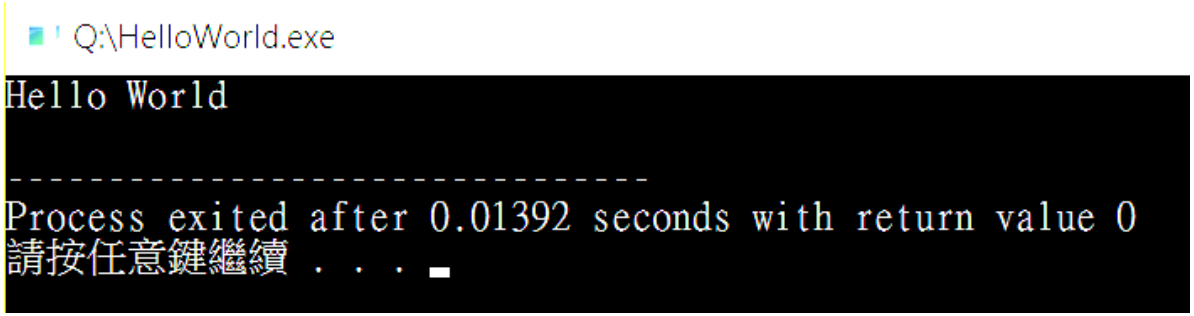
- 請寫一個 C 程式, 在螢幕上列印出 Hello World



- 請注意練習
  - DevCpp 程式開發環境
    - 開新... 原始碼
    - 執行 (編譯並執行)
  - 練習 e-Tutor 繳交與評測環境
  - 記住基本的 C 程式架構
    - main 函式
    - return 0; 敘述
    - printf(...); 在螢幕輸出文字的工具
    - #include <...> 前處理器指令
  - 如果打英文覺得有點慢的話

程式練習一: Hello World

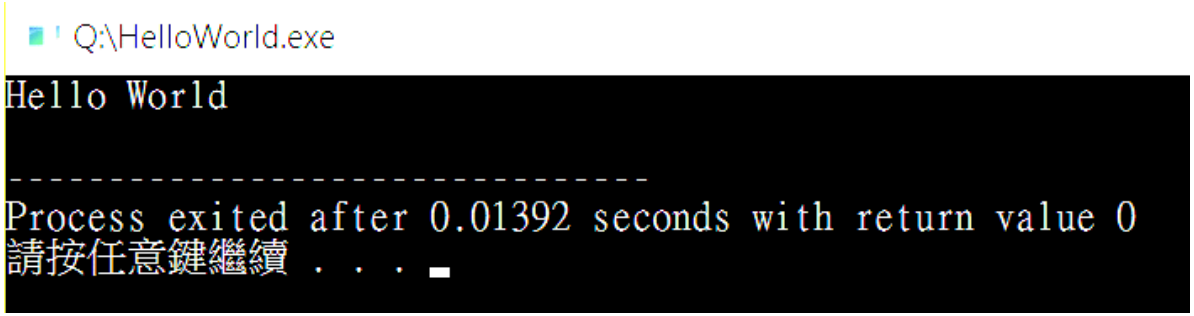
- 請寫一個 C 程式, 在螢幕上列印出 Hello World



- 請注意練習
  - DevCpp 程式開發環境
    - 開新... 原始碼
    - 執行 (編譯並執行)
  - 練習 e-Tutor 繳交與評測環境
  - 記住基本的 C 程式架構
    - main 函式
    - return 0; 敘述
    - printf(...); 在螢幕輸出文字的工具
    - #include <...> 前處理器指令
  - 如果打英文覺得有點慢的話
    - 編輯/插入文字

程式練習一: Hello World

- 請寫一個 C 程式, 在螢幕上列印出 Hello World



- 請注意練習
  - DevCpp 程式開發環境
    - 開新... 原始碼
    - 執行 (編譯並執行)
  - 練習 e-Tutor 繳交與評測環境
  - 記住基本的 C 程式架構
    - main 函式
    - return 0; 敘述
    - printf(...); 在螢幕輸出文字的工具
    - #include <...> 前處理器指令
  - 如果打英文覺得有點慢的話
    - 編輯/插入文字
    - 工具/編輯器選項/插入程式碼

程式練習二: 需要幾片地磚

程式練習二: 需要幾片地磚

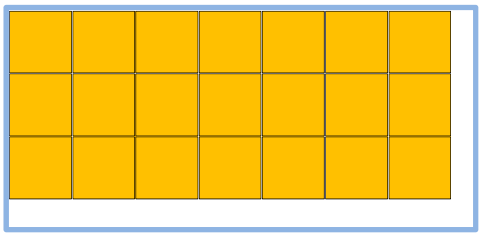
- 有一塊長寬都超過 500 公分的長方形空地, 希望算出最多能鋪上幾塊完整的正方形地磚?

程式練習二: 需要幾片地磚

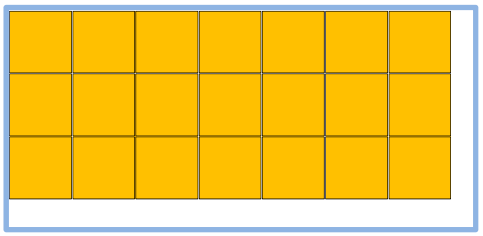
- 有一塊長寬都超過 500 公分的長方形空地, 希望算出最多能鋪上幾塊完整的正方形地磚?

程式練習二: 需要幾片地磚

- 有一塊長寬都超過 500 公分的長方形空地, 希望算出最多能鋪上幾塊完整的正方形地磚?

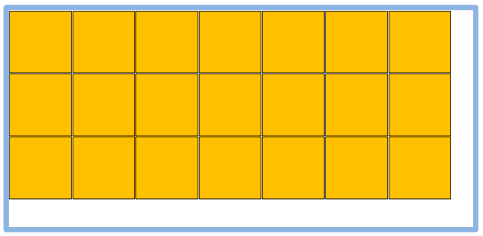


程式練習二: 需要幾片地磚



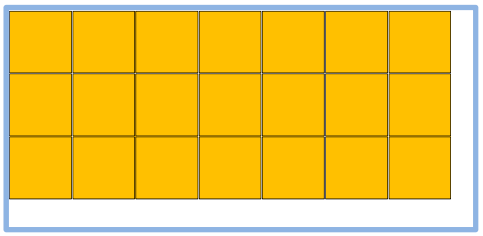
- 有一塊長寬都超過 500 公分的長方形空地, 希望算出最多能鋪上幾塊完整的正方形地磚?
- 程式要求使用者輸入兩個**整數**代表地板的長度與寬度 (單位為公分), 再輸入一個**整數**代表正方形地磚的邊長 (單位為公分), 程式請計算總共需要幾塊完整的地磚?

程式練習二: 需要幾片地磚



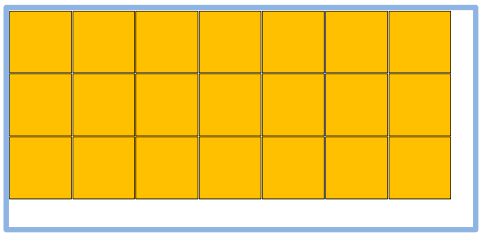
- 有一塊長寬都超過 500 公分的長方形空地, 希望算出最多能鋪上幾塊完整的正方形地磚?
- 程式要求使用者輸入兩個**整數**代表地板的長度與寬度 (單位為公分), 再輸入一個**整數**代表正方形地磚的邊長 (單位為公分), 程式請計算總共需要幾塊完整的地磚?
- **輸入測試資料:**
  1230 2100↵
  20↵

程式練習二: 需要幾片地磚



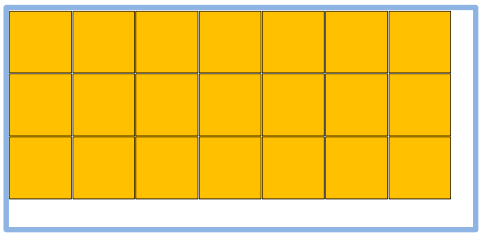
- 有一塊長寬都超過 500 公分的長方形空地, 希望算出最多能鋪上幾塊完整的正方形地磚?
- 程式要求使用者輸入兩個**整數**代表地板的長度與寬度 (單位為公分), 再輸入一個**整數**代表正方形地磚的邊長 (單位為公分), 程式請計算總共需要幾塊完整的地磚?
- **輸入測試資料:**
  1230 2100↲
  20↲
- **輸出測試資料:**
  6405↲

程式練習二: 需要幾片地磚



- 有一塊長寬都超過 500 公分的長方形空地, 希望算出最多能鋪上幾塊完整的正方形地磚?
- 程式要求使用者輸入兩個**整數**代表地板的長度與寬度 (單位為公分), 再輸入一個**整數**代表正方形地磚的邊長 (單位為公分), 程式請計算總共需要幾塊完整的地磚?
- **輸入測試資料:**
  1230 2100↵
  20↵
- **輸出測試資料:**
  6405↵
- **計算方法:**
  1230/20 ⇒ 61 餘 10
  2100/20 ⇒ 105 餘 0
  61 * 105 ⇒ 6405

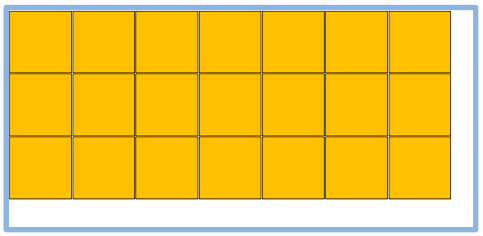
程式練習二: 需要幾片地磚



- 有一塊長寬都超過 500 公分的長方形空地, 希望算出最多能鋪上幾塊完整的正方形地磚?
- 程式要求使用者輸入兩個**整數**代表地板的長度與寬度 (單位為公分), 再輸入一個**整數**代表正方形地磚的邊長 (單位為公分), 程式請計算總共需要幾塊完整的地磚?
- **輸入測試資料:**
  1230 2100↵
  20↵
- **輸出測試資料:**
  6405↵
- **計算方法:**
  1230/20 ⇒ 61 餘 10
  2100/20 ⇒ 105 餘 0
  61 * 105 ⇒ 6405

- **設計方法 ---** 如果給你一張紙, 請你用左邊的方法重新算一次 (不要用心算), 你需要在紙上紀錄一些資料, 例如:

程式練習二: 需要幾片地磚



- 有一塊長寬都超過 500 公分的長方形空地, 希望算出最多能鋪上幾塊完整的正方形地磚?
- 程式要求使用者輸入兩個**整數**代表地板的長度與寬度 (單位為公分), 再輸入一個**整數**代表正方形地磚的邊長 (單位為公分), 程式請計算總共需要幾塊完整的地磚?
- **輸入測試資料:**
  1230 2100↵
  20↵
- **輸出測試資料:**
  6405↵
- **計算方法:**
  1230/20 ⇒ 61 餘 10
  2100/20 ⇒ 105 餘 0
  61 * 105 ⇒ 6405

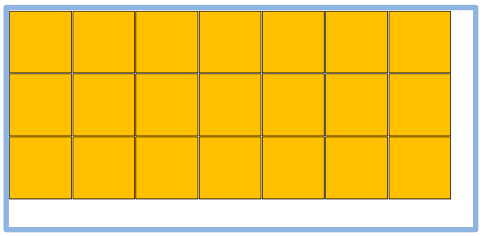
- **設計方法 ---** 如果給你一張紙, 請你用左邊的方法重新算一次 (不要用心算), 你需要在紙上紀錄一些資料, 例如:

$$1230 \div 20 = 61$$

$$2100 \div 20 = 105$$

$$61 \times 105 = 6405$$

程式練習二: 需要幾片地磚



- 有一塊長寬都超過 500 公分的長方形空地, 希望算出最多能鋪上幾塊完整的正方形地磚?
- 程式要求使用者輸入兩個**整數**代表地板的長度與寬度 (單位為公分), 再輸入一個**整數**代表正方形地磚的邊長 (單位為公分), 程式請計算總共需要幾塊完整的地磚?
- **輸入測試資料:**
  1230 2100↵
  20↵
- **輸出測試資料:**
  6405↵
- **計算方法:**
  1230/20 ⇒ 61 餘 10
  2100/20 ⇒ 105 餘 0
  61 * 105 ⇒ 6405
- **設計方法 ---** 如果給你一張紙, 請你用左邊的方法重新算一次 (不要用心算), 你需要在紙上紀錄一些資料, 例如:

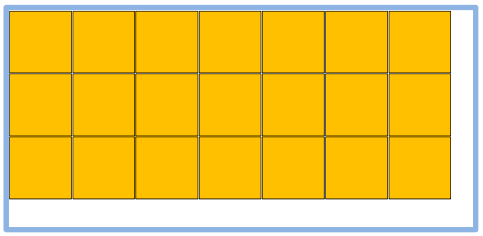
length

1230 ÷ 20 = 61

2100 ÷ 20 = 105

61 × 105 = 6405

程式練習二: 需要幾片地磚

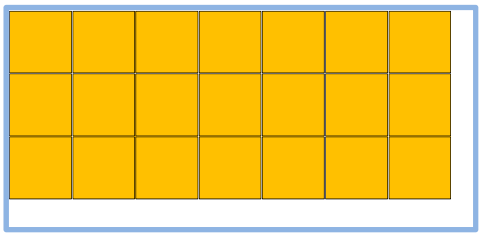


- 有一塊長寬都超過 500 公分的長方形空地, 希望算出最多能鋪上幾塊完整的正方形地磚?
- 程式要求使用者輸入兩個**整數**代表地板的長度與寬度 (單位為公分), 再輸入一個**整數**代表正方形地磚的邊長 (單位為公分), 程式請計算總共需要幾塊完整的地磚?
- **輸入測試資料:**
  1230 2100↵
  20↵
- **輸出測試資料:**
  6405↵
- **計算方法:**
  1230/20 ⇒ 61 餘 10
  2100/20 ⇒ 105 餘 0
  61 * 105 ⇒ 6405
- **設計方法 ---** 如果給你一張紙, 請你用左邊的方法重新算一次 (不要用心算), 你需要在紙上紀錄一些資料, 例如:

length
1230 ÷ 20 = 61
width
2100 ÷ 20 = 105

61 × 105 = 6405

程式練習二: 需要幾片地磚

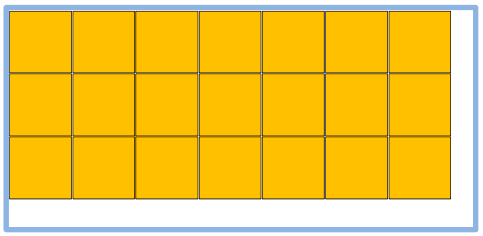


- 有一塊長寬都超過 500 公分的長方形空地, 希望算出最多能鋪上幾塊完整的正方形地磚?
- 程式要求使用者輸入兩個**整數**代表地板的長度與寬度 (單位為公分), 再輸入一個**整數**代表正方形地磚的邊長 (單位為公分), 程式請計算總共需要幾塊完整的地磚?
- **輸入測試資料:**
  1230 2100↵
  20↵
- **輸出測試資料:**
  6405↵
- **計算方法:**
  1230/20 ⇒ 61 餘 10
  2100/20 ⇒ 105 餘 0
  61 * 105 ⇒ 6405
- **設計方法 ---** 如果給你一張紙, 請你用左邊的方法重新算一次 (不要用心算), 你需要在紙上紀錄一些資料, 例如:

length　　side
1230 ÷ 20 = 61
width
2100 ÷ 20 = 105

61 × 105 = 6405

程式練習二: 需要幾片地磚



- 有一塊長寬都超過 500 公分的長方形空地, 希望算出最多能鋪上幾塊完整的正方形地磚?
- 程式要求使用者輸入兩個**整數**代表地板的長度與寬度 (單位為公分), 再輸入一個**整數**代表正方形地磚的邊長 (單位為公分), 程式請計算總共需要幾塊完整的地磚?
- **輸入測試資料:**
  1230 2100↵
  20↵
- **輸出測試資料:**
  6405↵
- **計算方法:**
  1230/20 ⇒ 61 餘 10
  2100/20 ⇒ 105 餘 0
  61 * 105 ⇒ 6405
- **設計方法 ---** 如果給你一張紙, 請你用左邊的方法重新算一次 (不要用心算), 你需要在紙上紀錄一些資料, 例如:

length　　side　　num_x

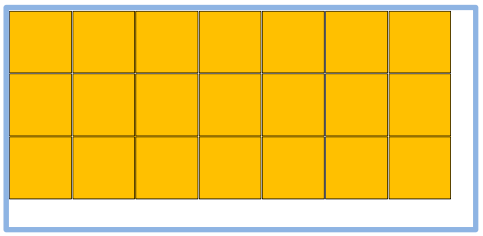
1230 ÷ 20 = 61

width

2100 ÷ 20 = 105

61 × 105 = 6405

程式練習二: 需要幾片地磚

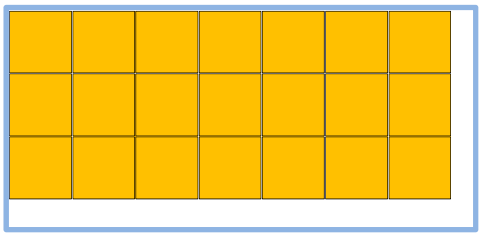


- 有一塊長寬都超過 500 公分的長方形空地, 希望算出最多能鋪上幾塊完整的正方形地磚?
- 程式要求使用者輸入兩個**整數**代表地板的長度與寬度 (單位為公分), 再輸入一個**整數**代表正方形地磚的邊長 (單位為公分), 程式請計算總共需要幾塊完整的地磚?
- **輸入測試資料:**
  1230 2100↵
  20↵
- **輸出測試資料:**
  6405↵
- **計算方法:**
  1230/20 ⇒ 61 餘 10
  2100/20 ⇒ 105 餘 0
  61 * 105 ⇒ 6405
- **設計方法 ---** 如果給你一張紙, 請你用左邊的方法重新算一次 (不要用心算), 你需要在紙上紀錄一些資料, 例如:

length　side　num_x
1230 ÷ 20 = 61
width　num_y
2100 ÷ 20 = 105

61 × 105 = 6405

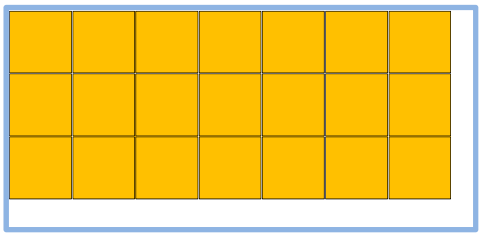
程式練習二: 需要幾片地磚



- 有一塊長寬都超過 500 公分的長方形空地, 希望算出最多能鋪上幾塊完整的正方形地磚?
- 程式要求使用者輸入兩個**整數**代表地板的長度與寬度 (單位為公分), 再輸入一個**整數**代表正方形地磚的邊長 (單位為公分), 程式請計算總共需要幾塊完整的地磚?
- **輸入測試資料:**
  1230 2100↵
  20↵
- **輸出測試資料:**
  6405↵
- **計算方法:**
  1230/20 ⇒ 61 餘 10
  2100/20 ⇒ 105 餘 0
  61 * 105 ⇒ 6405
- **設計方法 ---** 如果給你一張紙, 請你用左邊的方法重新算一次 (不要用心算), 你需要在紙上紀錄一些資料, 例如:

length　side　num_x
1230 ÷ 20 = 61
width　num_y
2100 ÷ 20 = 105
total_num
61 × 105 = 6405

程式練習二: 需要幾片地磚



- 有一塊長寬都超過 500 公分的長方形空地, 希望算出最多能鋪上幾塊完整的正方形地磚?
- 程式要求使用者輸入兩個**整數**代表地板的長度與寬度 (單位為公分), 再輸入一個**整數**代表正方形地磚的邊長 (單位為公分), 程式請計算總共需要幾塊完整的地磚?
- **輸入測試資料:**
  1230 2100↵
  20↵
- **輸出測試資料:**
  6405↵
- **計算方法:**
  1230/20 ⇒ 61 餘 10
  2100/20 ⇒ 105 餘 0
  61 * 105 ⇒ 6405

- **設計方法 ---** 如果給你一張紙, 請你用左邊的方法重新算一次 (不要用心算), 你需要在紙上紀錄一些資料, 例如:

length 1230 ÷ side 20 = num_x 61

width 2100 ÷ 20 = num_y 105

61 × 105 = total_num 6405

int length, width, …;

程式撰寫範例一

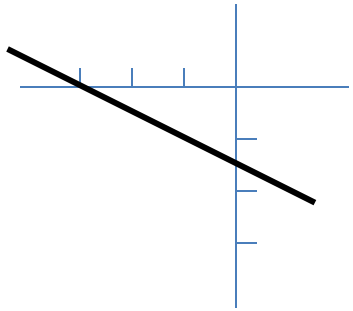
- 請寫一個程式計算 直線 $ax+by+c=0$ 和 **X 軸** 的交點

程式撰寫範例一

- 請寫一個程式計算 直線 a x + b y + c = 0 和 **X 軸** 的交點
- 例如 a = 1, b = 2, c = 3

程式撰寫範例一

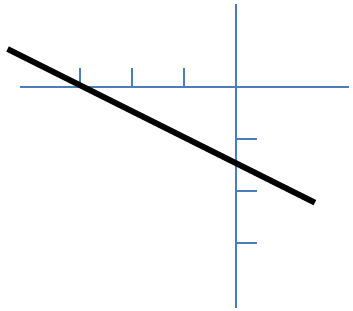
- 請寫一個程式計算 直線 a x + b y + c = 0 和 **X 軸** 的交點
- 例如 a = 1, b = 2, c = 3



程式撰寫範例一

- 請寫一個程式計算 直線 a x + b y + c = 0 和 **X 軸** 的交點
- 例如 a = 1, b = 2, c = 3

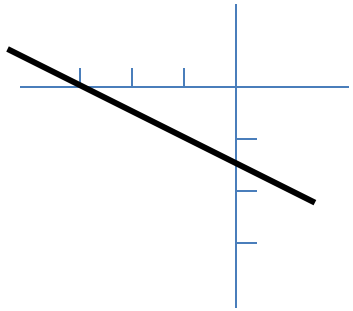
  和 X 軸的交點為 **(-3.0, 0.0)**



程式撰寫範例一

- 請寫一個程式計算 直線 a x + b y + c = 0 和 **X 軸** 的交點
- 例如 a = 1, b = 2, c = 3

  和 X 軸的交點為 **(-3.0, 0.0)**

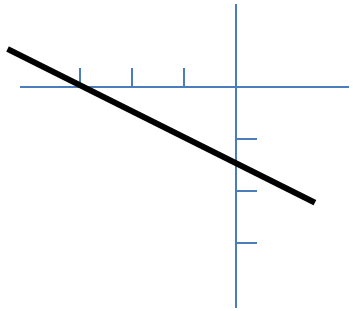


```
int main() {



    return 0;
}
```

程式撰寫範例一

- 請寫一個程式計算 直線 a x + b y + c = 0 和 **X 軸** 的交點
- 例如 a = 1, b = 2, c = 3

  和 X 軸的交點為 **(-3.0, 0.0)**

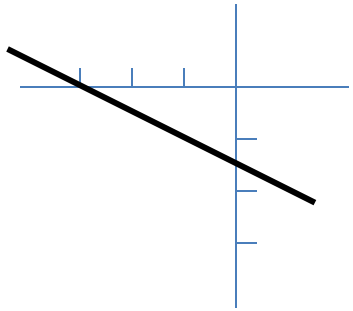


```
int main() {
    double a=1.0, b=2., c=3, x, y=0;



    return 0;
}
```

程式撰寫範例一

- 請寫一個程式計算 直線 a x + b y + c = 0 和 **X 軸** 的交點
- 例如 a = 1, b = 2, c = 3

  和 X 軸的交點為 **(-3.0, 0.0)**

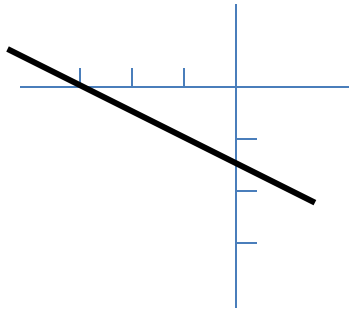


```
int main() {
    double a=1.0, b=2., c=3, x, y=0;
    x = - (b * y + c) / a;

    return 0;
}
```

程式撰寫範例一

- 請寫一個程式計算 直線 a x + b y + c = 0 和 **X 軸** 的交點
- 例如 a = 1, b = 2, c = 3

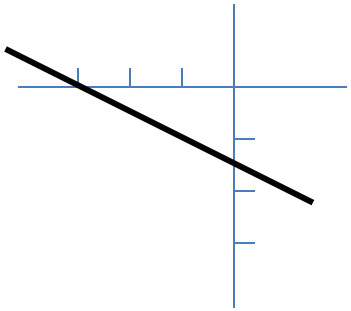
  和 X 軸的交點為 **(-3.0, 0.0)**



```
int main() {
    double a=1.0, b=2., c=3, x, y=0;
    x = - (b * y + c) / a;
    printf("和 X 軸的交點為 (%f, 0.0)\n", x);
    return 0;
}
```

程式撰寫範例一

- 請寫一個程式計算 直線 a x + b y + c = 0 和 **X 軸** 的交點
- 例如 a = 1, b = 2, c = 3

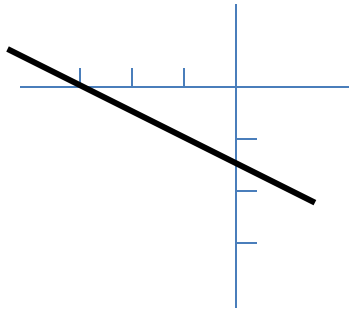


和 X 軸的交點為 **(-3.0, 0.0)**

```
#include <stdio.h>
int main() {
    double a=1.0, b=2., c=3, x, y=0;
    x = - (b * y + c) / a;
    printf("和 X 軸的交點為 (%f, 0.0)\n", x);
    return 0;
}
```

程式撰寫範例一

- 請寫一個程式計算 直線 a x + b y + c = 0 和 **X 軸** 的交點
- 例如 a = 1, b = 2, c = 3



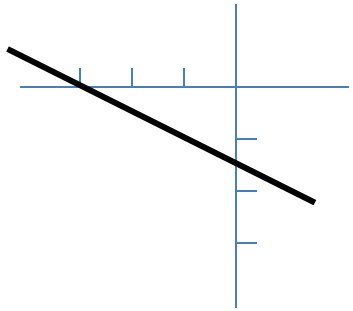
和 X 軸的交點為 **(-3.0, 0.0)**

```
#include <stdio.h>
int main() {
    double a=1.0, b=2., c=3, x, y=0;
    x = - (b * y + c) / a;
    printf("和 X 軸的交點為 (%f, 0.0)\n", x);
    return 0;
}
```

```
q:\>test01
和 X 軸的交點為 (-3.000000, 0.0)
```

程式撰寫範例一

- 請寫一個程式計算 直線 a x + b y + c = 0 和 **X 軸** 的交點
- 例如 a = 1, b = 2, c = 3



和 X 軸的交點為 **(-3.0, 0.0)**

```
#include <stdio.h>
int main() {
    double a=1.0, b=2., c=3, x, y=0;
    x = - (b * y + c) / a;
    printf("和 X 軸的交點為 (%f, 0.0)\n", x);
    return 0;
}
```

```
q:\>test01
和 X 軸的交點為 (-3.000000, 0.0)
```

格式命令

範例一 (cont'd)

- -3.000000 格式和要求不太一樣, 調整一下

```
#include <stdio.h>
int main() {
    double a=1, b=2, c=3, x, y=0;
    x = - (b * y + c) / a;
    printf("直線 %.1f x + %.1f y + %.1f = 0 ", a, b, c);
    printf("和 X 軸的交點為 (%.1f, 0.0)\n", x);
    return 0;
}
```

範例一 (cont'd)

- -3.000000 格式和要求不太一樣, 調整一下

```
#include <stdio.h>
int main() {
    double a=1, b=2, c=3, x, y=0;
    x = - (b * y + c) / a;
    printf("直線 %.1f x + %.1f y + %.1f = 0 ", a, b, c);
    printf("和 X 軸的交點為 (%.1f, 0.0)\n", x);
    return 0;
}
```

```
q:\>test02
直線 1.0 x + 2.0 y + 3.0 = 0 和 X 軸的交點為 (-3.0, 0.0)
```

範例一 (cont'd)

- 係數 **a**, **b**, **c** 可不可以不要是固定的?

範例一 (cont'd)

- 係數 **a**, **b**, **c** 可不可以不要是固定的? 由鍵盤輸入

範例一 (cont'd)

- 係數 **a**, **b**, **c** 可不可以不要是固定的? 由鍵盤輸入

```
#include <stdio.h>
int main() {
    double a, b, c, x, y=0;



    x = -(b * y + c) / a;
    printf("直線 %.1f x + %.1f y + %.1f = 0 ", a, b, c);
    printf("和 X 軸的交點為 (%.1f, 0.0)\n", x);
    return 0;
}
```

範例一 (cont'd)

- 係數 **a**, **b**, **c** 可不可以不要是固定的? 由鍵盤輸入

```
#include <stdio.h>
int main() {
    double a, b, c, x, y=0;
    printf("請輸入直線 a x + b y + c = 0 的係數: ");

    x = -(b * y + c) / a;
    printf("直線 %.1f x + %.1f y + %.1f = 0 ", a, b, c);
    printf("和 X 軸的交點為 (%.1f, 0.0)\n", x);
    return 0;
}
```

範例一 (cont'd)

- 係數 **a**, **b**, **c** 可不可以不要是固定的? 由鍵盤輸入

```
#include <stdio.h>
int main() {
    double a, b, c, x, y=0;
    printf("請輸入直線 a x + b y + c = 0 的係數: ");
    scanf("%lf%lf%lf", &a, &b, &c);
    x = -(b * y + c) / a;
    printf("直線 %.1f x + %.1f y + %.1f = 0 ", a, b, c);
    printf("和 X 軸的交點為 (%.1f, 0.0)\n", x);
    return 0;
}
```

範例一 (cont'd)

- 係數 **a**, **b**, **c** 可不可以不要是固定的? 由鍵盤輸入

```
#include <stdio.h>
int main() {
    double a, b, c, x, y=0;
    printf("請輸入直線 a x + b y + c = 0 的係數: ");
    scanf("%lf%lf%lf", &a, &b, &c);
    x = -(b * y + c) / a;
    printf("直線 %.1f x + %.1f y + %.1f = 0 ", a, b, c);
    printf("和 X 軸的交點為 (%.1f, 0.0)\n", x);
    return 0;
}
```

```
q:\>test03
請輸入直線 a x + b y + c = 0 的係數: 1.2 -2.5 5.9
直線 1.2 x + -2.5 y + 5.9 = 0 和 X 軸的交點為 (-4.9, 0.0)
```

範例一 (cont'd)

- 係數 **a**, **b**, **c** 可不可以不要是固定的? 由鍵盤輸入

```
#include <stdio.h>
int main() {
    double a, b, c, x, y=0;
    printf("請輸入直線 a x + b y + c = 0 的係數: ");
    scanf("%lf%lf%lf", &a, &b, &c);
    x = -(b * y + c) / a;
    printf("直線 %.1f x + %.1f y + %.1f = 0 ", a, b, c);
    printf("和 X 軸的交點為 (%.1f, 0.0)\n", x);
    return 0;
}
```

"%lf %lf %lf"

```
q:\>test03
請輸入直線 a x + b y + c = 0 的係數: 1.2 -2.5 5.9
直線 1.2 x + -2.5 y + 5.9 = 0 和 X 軸的交點為 (-4.9, 0.0)
```

範例一 (cont'd)

- 係數 **a**, **b**, **c** 可不可以不要是固定的? 由鍵盤輸入

```
#include <stdio.h>
int main() {
    double a, b, c, x, y=0;
    printf("請輸入直線 a x + b y + c = 0 的係數: ");
    scanf("%lf%lf%lf", &a, &b, &c);
    x = -(b * y + c) / a;
    printf("直線 %.1f x + %.1f y + %.1f = 0 ", a, b, c);
    printf("和 X 軸的交點為 (%.1f, 0.0)\n", x);
    return 0;
}
```

"%lf□%lf□%lf"

```
q:\>test03
請輸入直線 a x + b y + c = 0 的係數: 1.2 -2.5 5.9
直線 1.2 x + -2.5 y + 5.9 = 0 和 X 軸的交點為 (-4.9, 0.0)
```

範例一 (cont'd)

- 係數 **a**, **b**, **c** 可不可以不要是固定的? 由鍵盤輸入

```
#include <stdio.h>
int main() {
    double a, b, c, x, y=0;
    printf("請輸入直線 a x + b y + c = 0 的係數: ");
    scanf("%lf%lf%lf", &a, &b, &c);
    x = -(b * y + c) / a;
    printf("直線 %.1f x + %.1f y + %.1f = 0 ", a, b, c);
    printf("和 X 軸的交點為 (%.1f, 0.0)\n", x);
    return 0;
}
```

"%lf %lf %lf"

```
q:\>test03
請輸入直線 a x + b y + c = 0 的係數: 1.2 -2.5 5.9
直線 1.2 x + -2.5 y + 5.9 = 0 和 X 軸的交點為 (-4.9, 0.0)
```

範例一 (cont'd)

- 當使用者可以自由輸入 **a**, **b**, 與 **c** 時, 前一頁的程式有可能發生執行時的錯誤

範例一 (cont'd)

- 當使用者可以自由輸入 **a**, **b**, 與 **c** 時, 前一頁的程式有可能發生執行時的錯誤

```
q:\>test03
請輸入直線 a x + b y + c = 0 的係數: 0 1 2
直線 0.0 x + 1.0 y + 2.0 = 0 和 X 軸的交點爲 (-1.$, 0.0)
```

範例一 (cont'd)

- 當使用者可以自由輸入 **a**, **b**, 與 **c** 時, 前一頁的程式有可能發生執行時的錯誤

```
q:\>test03
請輸入直線 a x + b y + c = 0 的係數: 0 1 2
直線 0.0 x + 1.0 y + 2.0 = 0 和 X 軸的交點爲 (-1.$, 0.0)
```

- 仔細想一下, 輸出的 x 發生錯誤, 有問題的敘述可能是

```
x = -(b * y + c) / a;
```

範例一 (cont'd)

- 當使用者可以自由輸入 **a**, **b**, 與 **c** 時, 前一頁的程式有可能發生執行時的錯誤

```
q:\>test03
請輸入直線 a x + b y + c = 0 的係數: 0 1 2
直線 0.0 x + 1.0 y + 2.0 = 0 和 X 軸的交點爲 (-1.$, 0.0)
```

- 仔細想一下, 輸出的 x 發生錯誤, 有問題的敘述可能是

$$x = -(b * y + c) / a;$$

當係數 **a** 是 0 時, 除以 **0** 是會發生錯誤的!!

範例一 (cont'd)

- 當使用者可以自由輸入 **a**, **b**, 與 **c** 時, 前一頁的程式有可能發生執行時的錯誤

```
q:\>test03
請輸入直線 a x + b y + c = 0 的係數: 0 1 2
直線 0.0 x + 1.0 y + 2.0 = 0 和 X 軸的交點爲 (-1.$, 0.0)
```

- 仔細想一下, 輸出的 x 發生錯誤, 有問題的敘述可能是

```
x = -(b * y + c) / a;
```

當係數 **a** 是 0 時, 除以 **0** 是會發生錯誤的!!

那怎麼辦?!

範例一 (cont'd)

- 修改的
  方法

範例一 (cont'd)

- 修改的方法

```
if (a == 0) {
    printf("和 X 軸沒有交點\n");
}
else {
    x = -(b * y + c) / a;
    printf("和 X 軸的交點為 (%.1f, 0.0)\n", x);
}
```

範例一 (cont'd)

- 修改的方法

```
if (a == 0) {
    printf("和 X 軸沒有交點\n");
}
else {
    x = -(b * y + c) / a;
    printf("和 X 軸的交點為 (%.1f, 0.0)\n", x);
}
```

- 當 **a** 的數值很小的時候, 算出來的結果也很嚇人

範例一 (cont'd)

- 修改的方法

```
if (a == 0) {
    printf("和 X 軸沒有交點\n");
}
else {
    x = -(b * y + c) / a;
    printf("和 X 軸的交點為 (%.1f, 0.0)\n", x);
}
```

- 當 **a** 的數值很小的時候, 算出來的結果也很嚇人

```
q:\>test03
請輸入直線 a x + b y + c = 0 的係數: 1e-300 1 2
直線 0.0 x + 1.0 y + 2.0 = 0 和 X 軸的交點為 (-19999999999999998000000
0000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000.0, 0.0)
```

範例一 (cont'd)

- 修改的方法

```
if (a == 0) {
    printf("和 X 軸沒有交點\n");
}
else {
    x = -(b * y + c) / a;
    printf("和 X 軸的交點為 (%.1f, 0.0)\n", x);
}
```

- 當 **a** 的數值很小的時候, 算出來的結果也很嚇人

```
q:\>test03
請輸入直線 a x + b y + c = 0 的係數: 1e-300 1 2
直線 0.0 x + 1.0 y + 2.0 = 0 和 X 軸的交點為 (-19999999999999998000000
0000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000.0, 0.0)
```

範例一 (cont'd)

- 如果改成

```
if (a == 0) {
    printf("和 X 軸沒有交點\n");
}
else {
    x = -(b * y + c) / a;
    printf("和 X 軸的交點為 (%.1e, 0.0)\n", x);
}
```

範例一 (cont'd)

- 如果改成
```
if (a == 0) {
    printf("和 X 軸沒有交點\n");
}
else {
    x = -(b * y + c) / a;
    printf("和 X 軸的交點為 (%.1e, 0.0)\n", x);
}
```
- 輸出的結果會變成

範例一 (cont'd)

- 如果改成
```
if (a == 0) {
    printf("和 X 軸沒有交點\n");
}
else {
    x = -(b * y + c) / a;
    printf("和 X 軸的交點為 (%.1e, 0.0)\n", x);
}
```
- 輸出的結果會變成

```
q:\>test03
請輸入直線 a x + b y + c = 0 的係數: 1e-300 1 2
直線 0.0 x + 1.0 y + 2.0 = 0 和 X 軸的交點爲 (-2.0e+300, 0.0)
```

範例一 (cont'd)

- 如果改成
```
if (a == 0) {
    printf("和 X 軸沒有交點\n");
}
else {
    x = -(b * y + c) / a;
    printf("和 X 軸的交點為 (%.1e, 0.0)\n", x);
}
```
- 輸出的結果會變成

```
q:\>test03
請輸入直線 a x + b y + c = 0 的係數: 1e-300 1 2
直線 0.0 x + 1.0 y + 2.0 = 0 和 X 軸的交點爲 (-2.0e+300, 0.0)
```

範例一 (cont'd)

- 如果改成
```
if (a == 0) {
    printf("和 X 軸沒有交點\n");
}
else {
    x = -(b * y + c) / a;
    printf("和 X 軸的交點為 (%.1e, 0.0)\n", x);
}
```
- 輸出的結果會變成

```
q:\>test03
請輸入直線 a x + b y + c = 0 的係數: 1e-300 1 2
直線 0.0 x + 1.0 y + 2.0 = 0 和 X 軸的交點爲 (-2.0e+300, 0.0)
```

- 更有意義的改法

範例一 (cont'd)

- 如果改成
```
if (a == 0) {
    printf("和 X 軸沒有交點\n");
}
else {
    x = -(b * y + c) / a;
    printf("和 X 軸的交點為 (%.1e, 0.0)\n", x);
}
```
- 輸出的結果會變成

```
q:\>test03
請輸入直線 a x + b y + c = 0 的係數: 1e-300 1 2
直線 0.0 x + 1.0 y + 2.0 = 0 和 X 軸的交點爲 (-2.0e+300, 0.0)
```

- 更有意義的改法　if (a < 1e-10 && a >-1e-10) … else …

10^{-10}　　-10^{-10}

- 看完前面的範例以後, 應該要很清楚地知道：

- 看完前面的範例以後, 應該要很清楚地知道：
  - 程式 是操控電腦完成某項工作時一步一步的指令

- 看完前面的範例以後, 應該要很清楚地知道：
  - **程式** 是操控電腦完成某項工作時一步一步的指令
  - **寫程式** 是你在教電腦一步一步執行你想要它完成的工作

- 看完前面的範例以後, 應該要很清楚地知道：
  - **程式** 是操控電腦完成某項工作時一步一步的指令
  - **寫程式** 是你在教電腦一步一步執行你想要它完成的工作
- 想要順利地寫出一個程式, 你**需要能夠手動一步一步解決問題**, 把動作的順序記錄下來, 完整地用 C 語言的語法表示出來 .... 就是你的程式了

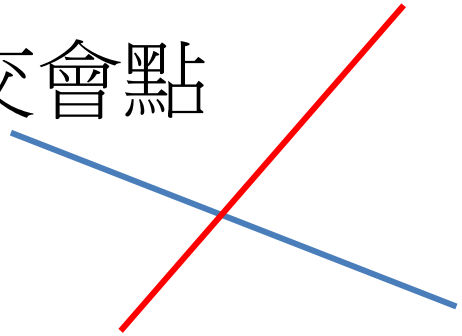
程式撰寫範例二

- 請寫一個程式計算 直線 $ax+by+c=0$ 和 直線 $dx+ey+f=0$ 的交點

程式撰寫範例二

- 請寫一個程式計算 直線 $ax+by+c=0$ 和 直線 $dx+ey+f=0$ 的交點

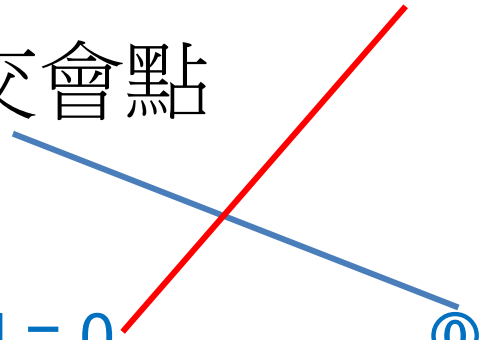
- 重點在於手動算出這兩條直線的交會點



程式撰寫範例二

- 請寫一個程式計算 直線 a x + b y + c = 0 和 直線 d x + e y + f = 0 的交點

- 重點在於手動算出這兩條直線的交會點



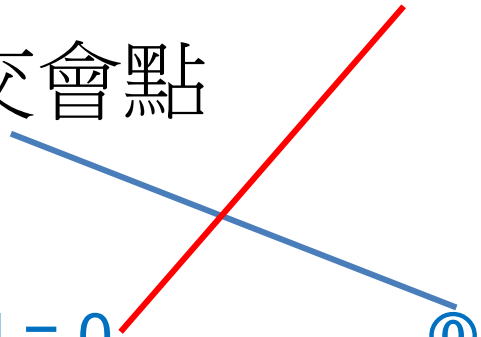
d != 0 時 d x + e y + f = 0 ⇒ x + e/d y + f/d = 0 … ⓪

⇒ a x + a e/d y + a f/d = 0 … ①

程式撰寫範例二

- 請寫一個程式計算 直線 a x + b y + c = 0 和 直線 d x + e y + f = 0 的交點

- 重點在於手動算出這兩條直線的交會點



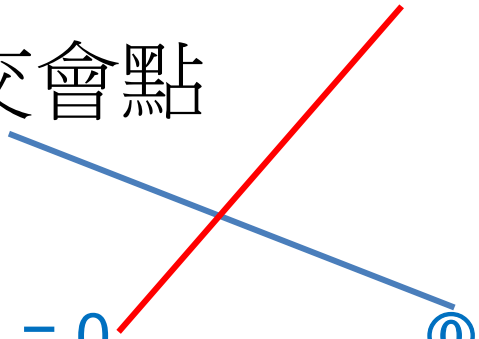
d != 0 時 d x + e y + f = 0 ⇒ x + e/d y + f/d = 0 … ⓪

⇒ a x + a e/d y + a f/d = 0 … ①

a x + b y + c = 0 … ②

程式撰寫範例二

- 請寫一個程式計算 直線 a x + b y + c = 0 和 直線 d x + e y + f = 0 的交點
- 重點在於手動算出這兩條直線的交會點



d != 0 時 d x + e y + f = 0 ⇒ x + e/d y + f/d = 0 … ⓪

⇒ a x + a e/d y + a f/d = 0 … ①

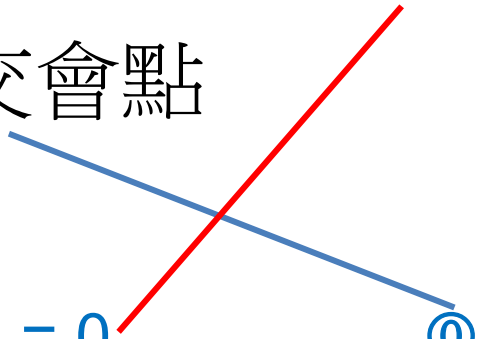
a x + b y + c = 0 … ②

① – ② ⇒ y = (c – a f / d) / (a e / d – b)

程式撰寫範例二

- 請寫一個程式計算 直線 a x + b y + c = 0 和 直線 d x + e y + f = 0 的交點

- 重點在於手動算出這兩條直線的交會點



d != 0 時 d x + e y + f = 0 ⇒ x + e/d y + f/d = 0 … ⓪

⇒ a x + a e/d y + a f/d = 0 … ①

a x + b y + c = 0 … ②

① – ② ⇒ y = (c – a f / d) / (a e / d – b)

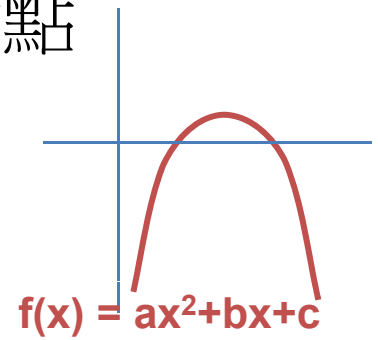
y 代回 ⓪ ⇒ x = – e / d (c – a f / d) / (a e / d – b) – f / d

程式撰寫範例三

- 讓我們寫一個程式計算一元二次方程式
  $a\,x^2 + b\,x + c = 0$ 的兩個根 (拋物線與 x 軸交點)

程式撰寫範例三

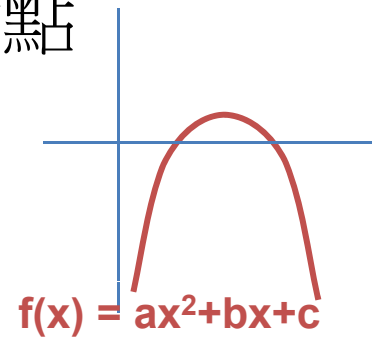
- 讓我們寫一個程式計算一元二次方程式
  $a\,x^2 + b\,x + c = 0$ 的兩個根 (拋物線與 x 軸交點)
- 重點在於手動算出 拋物線 與 X-軸 的交會點



程式撰寫範例三

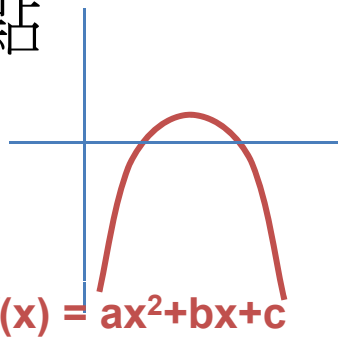
- 讓我們寫一個程式計算一元二次方程式
  $a\,x^2 + b\,x + c = 0$ 的兩個根 (拋物線與 x 軸交點)
- 重點在於手動算出 拋物線 與 X-軸 的交會點

判別式 $d = b^2 - 4\,a\,c$



程式撰寫範例三

- 讓我們寫一個程式計算一元二次方程式
  $a\,x^2 + b\,x + c = 0$ 的兩個根 (拋物線與 x 軸交點)
- 重點在於手動算出 拋物線 與 X-軸 的交會點

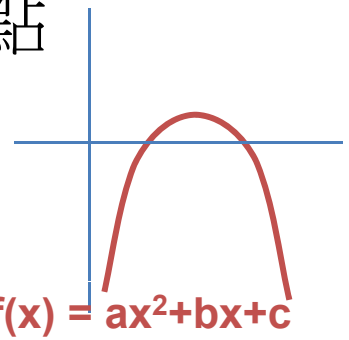


判別式 $d = b^2 - 4\,a\,c$

d>0 時
$$x_1 = (-b + \sqrt{d}\,) / (2\,a)$$
$$x_2 = (-b - \sqrt{d}\,) / (2\,a)$$

程式撰寫範例三

- 讓我們寫一個程式計算一元二次方程式
  $a\,x^2 + b\,x + c = 0$ 的兩個根 (拋物線與 x 軸交點)
- 重點在於手動算出 拋物線 與 X-軸 的交會點



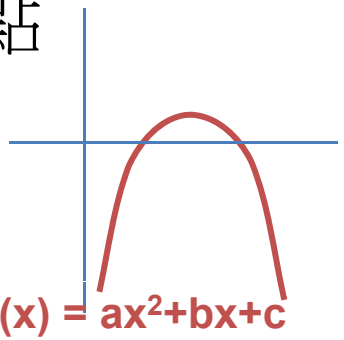
判別式 $d = b^2 - 4\,a\,c$

d>0 時
$x_1 = (-b + \sqrt{d}) / (2\,a)$
$x_2 = (-b - \sqrt{d}) / (2\,a)$

d==0 時只有一個交點 $x = -b / (2\,a)$

程式撰寫範例三

- 讓我們寫一個程式計算一元二次方程式
  $a\,x^2 + b\,x + c = 0$ 的兩個根 (拋物線與 x 軸交點)
- 重點在於手動算出 拋物線 與 X-軸 的交會點



判別式 $d = b^2 - 4\,a\,c$

d>0 時
$x_1 = (-b + \sqrt{d}\,) / (2\,a)$
$x_2 = (-b - \sqrt{d}\,) / (2\,a)$

d==0 時只有一個交點 $x = -b / (2\,a)$

d<0 時沒有交點

設計這個程式的步驟

1. 設計基本程序

設計這個程式的步驟

1. 設計基本程序
   a. 讀取三個係數

設計這個程式的步驟

1. 設計基本程序
   a. 讀取三個係數
   b. 檢查是拋物線還是直線

設計這個程式的步驟

1. 設計基本程序
   a. 讀取三個係數
   b. 檢查是拋物線還是直線
      如果是直線的話, 直接算和 x 軸的交點

設計這個程式的步驟

1. 設計基本程序
   a. 讀取三個係數
   b. 檢查是拋物線還是直線
      如果是直線的話, 直接算和 x 軸的交點
   c. 如果是拋物線的話, 分辨有沒有實數根

設計這個程式的步驟

1. 設計基本程序
   a. 讀取三個係數
   b. 檢查是拋物線還是直線
      如果是直線的話, 直接算和 x 軸的交點
   c. 如果是拋物線的話, 分辨有沒有實數根
      計算判別式

設計這個程式的步驟

1. 設計基本程序
   a. 讀取三個係數
   b. 檢查是拋物線還是直線
      如果是直線的話, 直接算和 x 軸的交點
   c. 如果是拋物線的話, 分辨有沒有實數根
      計算判別式
   d. 判別式小於 0 時沒有實數解, 列印訊息

設計這個程式的步驟

1. 設計基本程序
   a. 讀取三個係數
   b. 檢查是拋物線還是直線
      如果是直線的話, 直接算和 x 軸的交點
   c. 如果是拋物線的話, 分辨有沒有實數根
      計算判別式
   d. 判別式小於 0 時沒有實數解, 列印訊息
      等於 0 時只有一個實數解, 計算, 列印

設計這個程式的步驟

1. 設計基本程序
   a. 讀取三個係數
   b. 檢查是拋物線還是直線
      如果是直線的話, 直接算和 x 軸的交點
   c. 如果是拋物線的話, 分辨有沒有實數根
      計算判別式
   d. 判別式小於 0 時沒有實數解, 列印訊息
      等於 0 時只有一個實數解, 計算, 列印
      大於 0 時有兩個實數解, 計算, 列印

設計這個程式的步驟

1. 設計基本程序

   a. 讀取三個係數

   b. 檢查是拋物線還是直線

      如果是直線的話, 直接算和 x 軸的交點

   c. 如果是拋物線的話, 分辨有沒有實數根

      計算判別式

   d. 判別式小於 0 時沒有實數解, 列印訊息

      等於 0 時只有一個實數解, 計算, 列印

      大於 0 時有兩個實數解, 計算, 列印

2. 變數設計

設計這個程式的步驟

1. 設計基本程序

   a. 讀取三個係數

   b. 檢查是拋物線還是直線

      如果是直線的話, 直接算和 x 軸的交點

   c. 如果是拋物線的話, 分辨有沒有實數根

      計算判別式

   d. 判別式小於 0 時沒有實數解, 列印訊息

      等於 0 時只有一個實數解, 計算, 列印

      大於 0 時有兩個實數解, 計算, 列印

2. 變數設計

```
double a, b, c;
```

設計這個程式的步驟

1. 設計基本程序

   a. 讀取三個係數

   b. 檢查是拋物線還是直線

      如果是直線的話, 直接算和 x 軸的交點

   c. 如果是拋物線的話, 分辨有沒有實數根

      計算判別式

   d. 判別式小於 0 時沒有實數解, 列印訊息

      等於 0 時只有一個實數解, 計算, 列印

      大於 0 時有兩個實數解, 計算, 列印

2. 變數設計

   double **a**, **b**, **c**;

   double **d**;

設計這個程式的步驟

1. 設計基本程序

   a. 讀取三個係數

   b. 檢查是拋物線還是直線

      如果是直線的話, 直接算和 x 軸的交點

   c. 如果是拋物線的話, 分辨有沒有實數根

      計算判別式

   d. 判別式小於 0 時沒有實數解, 列印訊息

      等於 0 時只有一個實數解, 計算, 列印

      大於 0 時有兩個實數解, 計算, 列印

2. 變數設計

```
double a, b, c;
double d;
double x;
```

設計這個程式的步驟

1. 設計基本程序

   a. 讀取三個係數

   b. 檢查是拋物線還是直線

      如果是直線的話, 直接算和 x 軸的交點

   c. 如果是拋物線的話, 分辨有沒有實數根

      計算判別式

   d. 判別式小於 0 時沒有實數解, 列印訊息

      等於 0 時只有一個實數解, 計算, 列印

      大於 0 時有兩個實數解, 計算, 列印

2. 變數設計

   double **a**, **b**, **c**;

   double **d**;

   double **x**;

   double **x1**, **x2**;

設計這個程式的步驟

1. 設計基本程序

   a. 讀取三個係數 scanf("%lf%lf%lf",

   b. 檢查是拋物線還是直線

      如果是直線的話, 直接算和 x 軸的交點

   c. 如果是拋物線的話, 分辨有沒有實數根

      計算判別式

   d. 判別式小於 0 時沒有實數解, 列印訊息

      等於 0 時只有一個實數解, 計算, 列印

      大於 0 時有兩個實數解, 計算, 列印

2. 變數設計

   double **a**, **b**, **c**;

   double **d**;

   double **x**;

   double **x1**, **x2**;

設計這個程式的步驟

1. 設計基本程序

   a. 讀取三個係數 scanf("%lf%lf%lf",

   b. 檢查是拋物線還是直線

      如果是直線的話, 直接算和 x 軸的交點

   c. 如果是拋物線的話, 分辨有沒有實數根

      計算判別式

   d. 判別式小於 0 時沒有實數解, 列印訊息

      等於 0 時只有一個實數解, 計算, 列印

      大於 0 時有兩個實數解, 計算, 列印

2. 變數設計

   double **a**, **b**, **c**;

   double **d**;

   double **x**;

   double **x1**, **x2**;

```
if (d<0)
{
…
}
else if (d==0)
{
…
}
else // if (d>0)
{
…
}
```

設計這個程式的步驟

1. 設計基本程序

   a. 讀取三個係數 scanf("%lf%lf%lf",

   b. 檢查是拋物線還是直線

      如果是直線的話, 直接算和 x 軸的交點

   c. 如果是拋物線的話, 分辨有沒有實數根

      計算判別式

   d. 判別式小於 0 時沒有實數解, 列印訊息

      等於 0 時只有一個實數解, 計算, 列印

      大於 0 時有兩個實數解, 計算, 列印

2. 變數設計

   double **a**, **b**, **c**;

   double **d**;

   double **x**;

   double **x1**, **x2**;

```
if (d<0)
{
…
}
else if (d==0)
{
…
}
else // if (d>0)
{
…
}
```

```
x = - b / (2 * a);
```

設計這個程式的步驟

1. 設計基本程序

   a. 讀取三個係數 scanf("%lf%lf%lf",

   b. 檢查是拋物線還是直線

      如果是直線的話, 直接算和 x 軸的交點

   c. 如果是拋物線的話, 分辨有沒有實數根

      計算判別式

   d. 判別式小於 0 時沒有實數解, 列印訊息

      等於 0 時只有一個實數解, 計算, 列印

      大於 0 時有兩個實數解, 計算, 列印

2. 變數設計

   double **a**, **b**, **c**;

   double **d**;

   double **x**;

   double **x1**, **x2**;

```
if (d<0)
{
…
}
else if (d==0)
{
…
}
else // if (d>0)
{
…
}
```

```
x = - b / (2 * a);

x1 = (- b + sqrt(d)) / (2 * a);
```

設計這個程式的步驟

1. 設計基本程序

   a. 讀取三個係數 scanf("%lf%lf%lf",

   b. 檢查是拋物線還是直線

      如果是直線的話, 直接算和 x 軸的交點

   c. 如果是拋物線的話, 分辨有沒有實數根

      計算判別式

   d. 判別式小於 0 時沒有實數解, 列印訊息

      等於 0 時只有一個實數解, 計算, 列印

      大於 0 時有兩個實數解, 計算, 列印

2. 變數設計

   double **a**, **b**, **c**;

   double **d**;

   double **x**;

   double **x1**, **x2**;

```
if (d<0)
{
…
}
else if (d==0)
{
…
}
else // if (d>0)
{
…
}
```

```
x = - b / (2 * a);

x1 = (- b + sqrt(d)) / (2 * a);
x2 = (- b - sqrt(d)) / (2 * a);
```

設計這個程式的步驟

1. 設計基本程序

   a. 讀取三個係數 scanf("%lf%lf%lf",

   b. 檢查是拋物線還是直線

      如果是直線的話, 直接算和 x 軸的交點

   c. 如果是拋物線的話, 分辨有沒有實數根

      計算判別式

   d. 判別式小於 0 時沒有實數解, 列印訊息

      等於 0 時只有一個實數解, 計算, 列印

      大於 0 時有兩個實數解, 計算, 列印

2. 變數設計

```
double a, b, c;
double d;
double x;
double x1, x2;
```

```
if (d<0)
{
…
}
else if (d==0)
{
…
}
else // if (d>0)
{
…
}
```

```
x = - b / (2 * a);
printf("%.3f\n", x);
x1 = (- b + sqrt(d)) / (2 * a);
x2 = (- b - sqrt(d)) / (2 * a);
```

設計這個程式的步驟

1. 設計基本程序

   a. 讀取三個係數 scanf("%lf%lf%lf",

   b. 檢查是拋物線還是直線

      如果是直線的話, 直接算和 x 軸的交點

   c. 如果是拋物線的話, 分辨有沒有實數根

      計算判別式

   d. 判別式小於 0 時沒有實數解, 列印訊息

      等於 0 時只有一個實數解, 計算, 列印

      大於 0 時有兩個實數解, 計算, 列印

2. 變數設計

```
double a, b, c;
double d;
double x;
double x1, x2;
```

```
if (d<0)
{
…
}
else if (d==0)
{
…
}
else // if (d>0)
{
…
}
```

```
x = - b / (2 * a);
printf("%.3f\n", x);

x1 = (- b + sqrt(d)) / (2 * a);
x2 = (- b - sqrt(d)) / (2 * a);
printf("%.3f, %.3f\n", x1, x2);
```

程式撰寫範例四

- 請寫一個程式計算等差級數 (等差數列的和)
- 重點在於如何手動算出 等差級數 的結果

程式撰寫範例四

- 請寫一個程式計算等差級數 (等差數列的和)
- 重點在於如何手動算出 等差級數 的結果

計算 $a_1 + a_2 + a_3 + \ldots + a_n$

其中 $a_i = a_1 + (i-1) * d$, d 是公差

程式撰寫範例四

- 請寫一個程式計算等差級數 (等差數列的和)
- 重點在於如何手動算出 等差級數 的結果

計算 $a_1 + a_2 + a_3 + \ldots + a_n$

其中 $a_i = a_1 + (i-1) * d$, d 是公差

$a_1 \quad + \quad a_2 \quad + \quad a_3 \quad + \ldots + \quad a_n$

程式撰寫範例四

- 請寫一個程式計算等差級數 (等差數列的和)
- 重點在於如何手動算出 等差級數 的結果

計算 $a_1 + a_2 + a_3 + \ldots + a_n$

其中 $a_i = a_1 + (i-1) * d$, d 是公差

$$a_1 + a_2 + a_3 + \ldots + a_n$$
$$a_n + a_{n-1} + a_{n-2} + \ldots + a_1$$

程式撰寫範例四

- 請寫一個程式計算等差級數 (等差數列的和)
- 重點在於如何手動算出 等差級數 的結果

計算 $a_1 + a_2 + a_3 + \ldots + a_n$

其中 $a_i = a_1 + (i-1) * d$, d 是公差

$$\begin{array}{ccccccc} a_1 & + & a_2 & + & a_3 & + \ldots + & a_n \\ a_n & + & a_{n-1} & + & a_{n-2} & + \ldots + & a_1 \\ \hline a_1+a_n & & a_1+a_n & & a_1+a_n & & a_1+a_n \end{array}$$

程式撰寫範例四

- 請寫一個程式計算等差級數 (等差數列的和)
- 重點在於如何手動算出 等差級數 的結果

計算 $a_1 + a_2 + a_3 + \ldots + a_n$

其中 $a_i = a_1 + (i-1) * d$, d 是公差

梯形公式

$$\frac{(a_1 + a_n)\, n}{2}$$

$$\begin{array}{ccccccc} a_1 & + & a_2 & + & a_3 & + \ldots + & a_n \\ a_n & + & a_{n-1} & + & a_{n-2} & + \ldots + & a_1 \\ \hline a_1+a_n & & a_1+a_n & & a_1+a_n & & a_1+a_n \end{array}$$

程式撰寫範例四

- 請寫一個程式計算等差級數 (等差數列的和)
- 重點在於如何手動算出 等差級數 的結果

計算 $a_1 + a_2 + a_3 + \ldots + a_n$

其中 $a_i = a_1 + (i-1) * d$, d 是公差

梯形公式

$$\frac{(a_1 + a_n)\, n}{2}$$

$$\begin{array}{ccccccc} a_1 & + & a_2 & + & a_3 & + \ldots + & a_n \\ a_n & + & a_{n-1} & + & a_{n-2} & + \ldots + & a_1 \\ \hline a_1+a_n & & a_1+a_n & & a_1+a_n & & a_1+a_n \end{array}$$

轉換成程式