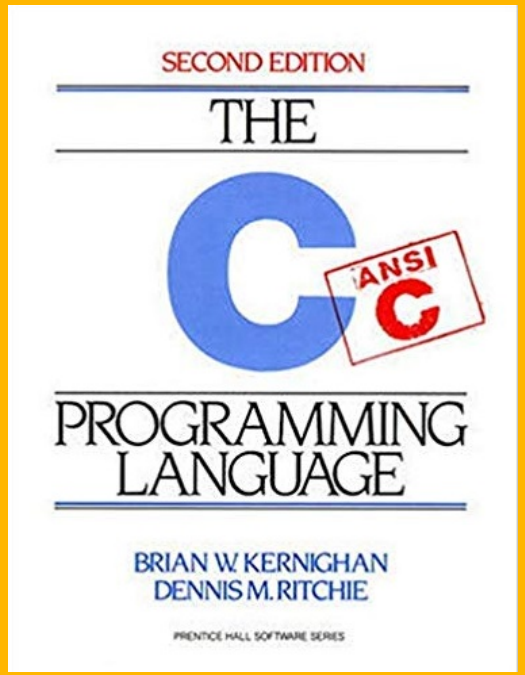


Introduction to Computer Science Fall 2022 #16

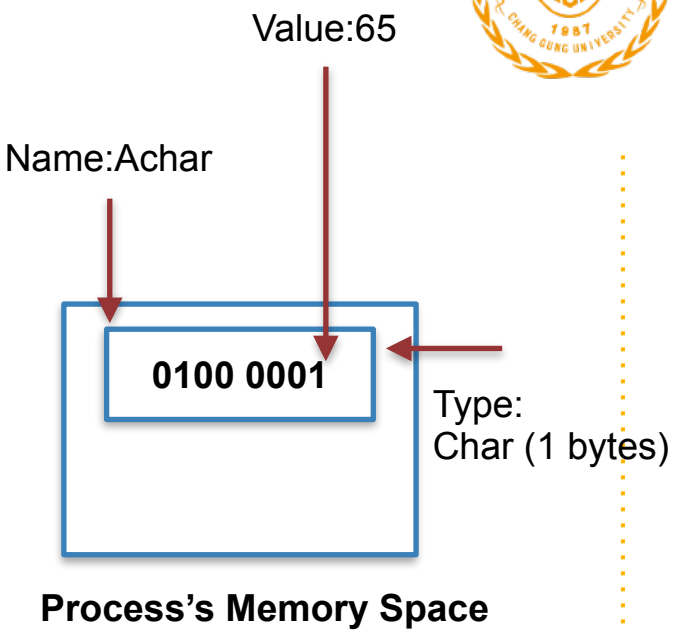
Chi-Jen Wu

期末考資訊

- 2023年 1月 12 號 星期四 下午三點
- **手寫考卷**
- **五題C語言程式題**
  - Variables & Flow of Control
  - Function & Arrays (strings)
  - 包含以上範圍

Variables

- 值 (Value)
  - 在記憶體實際的二進位值
- 名字 (Name)
  - 在程式裡的代號
- 型別 (Type)
  - 在記憶體所佔的空間



Topics

- Problem Solving with Programming Language
- C Programming
  - C Basics
    - Variables
    - Flow of Control
    - Function Basics
    - Programming with Arrays
    - Strings
    - Structures
    - Streams and File I/O
- Google Cloud Platform/Cloud Shell Editor (gcc/g++/Makefile)
- Google Cloud Platform/Cloud Source Repositories (git)

和Variables有關的事

- 注意變數型別，assignment and output
- Integer (整數)
  - char
    - printf(“%**c**”), printf(“%**d**”), printf(“%**i**”)
  - Int
    - printf(“%**d**”), printf(“%**i**”), printf(“%**u**”)
  - Long
    - printf(“%**lu**”), printf(“%**d**”), printf(“%**i**”)

```c
#include <stdio.h>
int main() {
    int a = 100;
    long b = -100;
    printf("a in int: %i\n", a);
    printf("b in long: %lu\n", b);
    return 0;
}
```

unsigned long

input

```
a in int: 100
b in long: 18446744073709551516


...Program finished with exit code 0
Press ENTER to exit console.
```

記憶體存什麼就是直接轉出來
所以必須知道你的程式在幹什麼

```c
#include <stdio.h>
int main() {
    printf("string in int: %d\n", "1");
    return 0;
}
```

input

```
main.c:3:29: warning: format ‘%d’ expects argument of type ‘int’, but argument 2 has type ‘char *’
[-Wformat=]
    3 |     printf("string in int: %d\n", "1");
      |                            ~^     ~~~
      |                             |     |
      |                             int   char *
      |                            %s
string in int: 1955016708


...Program finished with exit code 0
Press ENTER to exit console.
```

Printf 不會管你給的是什麼
他就是轉成%d然後印出來

signed int

```c
#include <stdio.h>
int main() {
    printf("string in int: %s\n", "1234");
    return 0;
}
```

input

```
string in int: 1234


...Program finished with exit code 0
Press ENTER to exit console.
```

你必須關注變數的型別Type
和他對應的output

You should know that!

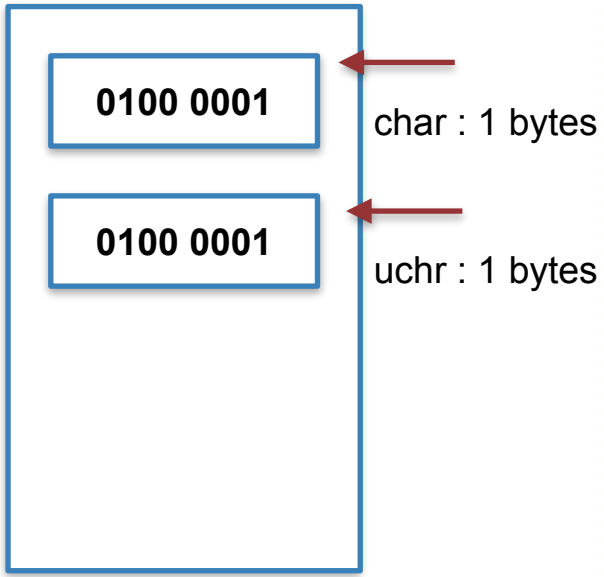
- Integer
  - Overflow
  - MIN & MAX
  - Rounding
- floating-point
  - 浮點數的陷阱
  - Rounding
  - 無法取%
  - 無法用邏輯運算子 (& | ~ ^)，(其實是可以)

```c
#include <stdio.h>
int main() {
    unsigned char uval = 65;
    char val = 65;
    printf("%d\n", uval *=2);
    printf("%d\n", val *=2);
    uval = 193;
    printf("%d\n", uval *=2);
    // 1 1000 0010 = 386 (overflow)
    //   1000 0010 = 130
    return 0;
}
```

input

```
130
-126
130


...Program finished with exit code 0
Press ENTER to exit console.
```

要能自動產生這個圖



```c
#include <stdio.h>
int main() {
    unsigned char x = 150;
    unsigned char y = 250;
    unsigned char z = 0;
    unsigned char z1 = 0;
    z = (x+y);
    // overflow
    z1 = (y - x)/2;
    printf("x=%d, y=%d\n", x, y);
    printf("%i, %i\n", z, z/2);
    printf("%i, %i\n", z1, z1 + x);

    return 0;
}
```

input

```
x=150, y=250
144, 72
50, 200


...Program finished with exit code 0
Press ENTER to exit console.
```

計算平均要小心！

Integer overflow problem

你可以這樣做！

Integer Rounding

Type	Storage size	Value range
char	1 byte	-128 to 127 or 0 to 255
unsigned char	1 byte	0 to 255
signed char	1 byte	-128 to 127
int	2 or 4 bytes	-32,768 to 32,767 or -2,147,483,648 to 2,147,483,647
unsigned int	2 or 4 bytes	0 to 65,535 or 0 to 4,294,967,295
short	2 bytes	-32,768 to 32,767
unsigned short	2 bytes	0 to 65,535
long	8 bytes or (4bytes for 32 bit OS)	-9223372036854775808 to 9223372036854775807
unsigned long	8 bytes	0 to 18446744073709551615

Storage size 小的 <— 大的 就會產生 Rounding

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

    int i = 0x00123456;
    long l = 0x0012345600123456;
    printf("string in int: %u\n", i);
    printf("string in int: %lu\n", l);
    i = l;
    printf("string in int: %u\n", i);
    return 0;
}
```

input

```
string in int: 1193046
string in int: 5124093553816662
string in int: 1193046


...Program finished with exit code 0
Press ENTER to exit console.
```

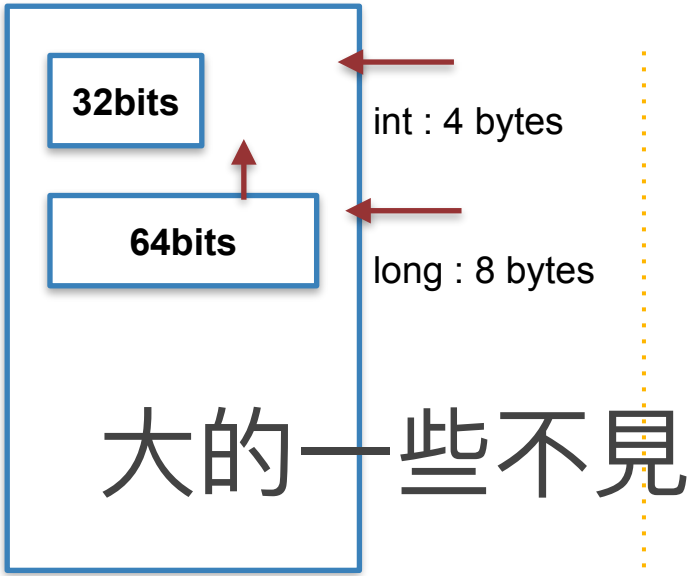
大的放小的

Rounding

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

    int i = 0x00123456;
    long l = 0x0012345600123456;
    printf("string in int: %u\n", i);
    printf("string in int: %lu\n", l);
    i = l;
    printf("string in int: %u\n", i);
    return 0;
}
```

input

```
string in int: 1193046
string in int: 5124093553816662
string in int: 1193046


...Program finished with exit code 0
Press ENTER to exit console.
```



Process's Memory Space

```c
#include <stdio.h>
int main() {
    float a;
    float b;

    a = 1.9f - 0.01f;
    b = 0.9f;
    printf("a in float: %.8f\n", a);
    printf("b in float: %.8f\n", b);

    return 0;
}
```

input

a 是多少，b 是多少？

```
...Program finished with exit code 0
Press ENTER to exit console.
```

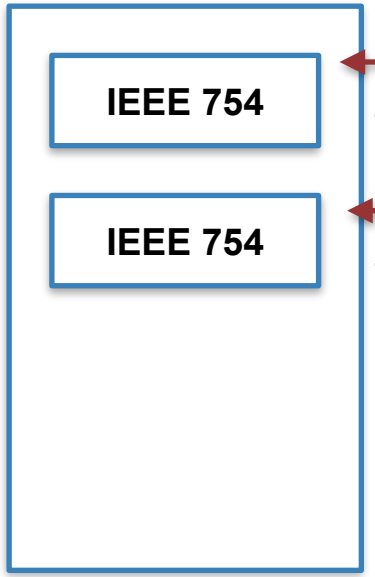
floating-point
浮點數的陷阱

絕對和你想的不一樣，
他存成什麼只有他自己
知道！！！

```c
#include <stdio.h>
int main() {
    float a;
    float b;

    a = 1.000001f - 0.1f;
    b = 0.900001f;
    printf("a in float: %f\n", a);
    printf("b in float: %f\n", b);
    if (a == b)
        printf("a = b\n");

    return 0;
}
```

input

```
a in float: 0.900001
b in float: 0.900001


...Program finished with exit code 0
Press ENTER to exit console.
```



floating-point
浮點數的陷阱

```c
#include <stdio.h>
int main() {
    float a;
    float b;
    int ia;
    int ib;
    ia = 1.000001f * 10000000;
    printf("%d\n", ia);
    ia -= 0.1f * 10000000;
    ib = 0.900001f * 10000000;
    printf("a in float: %d\n", ia);
    printf("b in float: %d\n", ib);
    if (a == b)
        printf("a = b\n");
```

input

```
10000010
a in float: 9000010
b in float: 9000010
a = b


...Program finished with exit code 0
Press ENTER to exit console.
```

floating-point
浮點數的陷阱

浮點數是不好處理的！

在可能的狀態下
建議轉成整數操作

定點數

- 浮點數轉整數
  - * 100000000000000？
  - 還是會有很多不確定的問題
- 大部分會轉成定點數
- 將小數點固定在最右邊
- 也可以透過**IEEE 745**的規格
  - 將浮點數轉化成整數儲存！
  - 大部分對付浮點數的作法

係數1/100

小數數值	用整數表示的值
0.00	0
0.5	50
0.99	99
2	200
−14.1	−1410
314.160	31416

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

    float x = 0.1;
    printf("\t %lu %lu %lu", sizeof(x), sizeof(0.1), sizeof(0.1f));
    return 0;
}
```

float double float

input

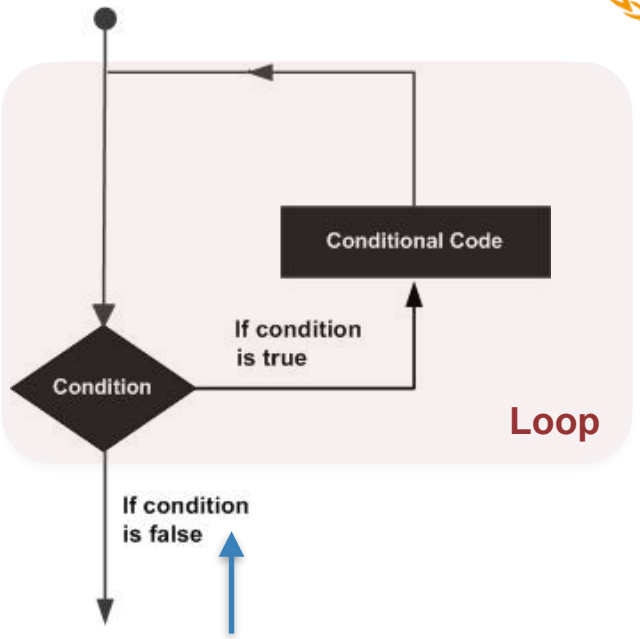
```
        4 8 4

...Program finished with exit code 0
Press ENTER to exit console.
```

都是 floating-point

但是他儲存的storage size 卻不一樣！

Flow of Control

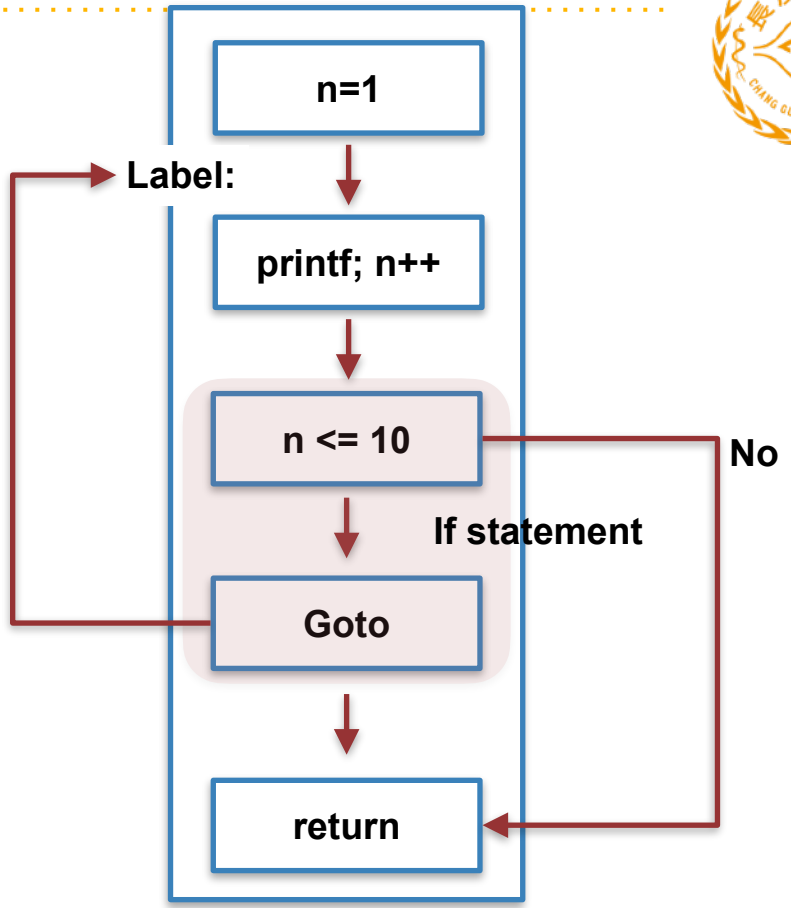
- while loop
- For loop
- Do …. While loop
- Nested loops
- Break
- Continue



利用 if 和 goto 也可以產生Loop
所有的Loop都可以轉換成這種形式

```c
#include <stdio.h>
int main() {
    int n = 1;
label:
    printf("%d ",n);
    n++;
    if (n <= 10)
        goto label;
    return 0;
}
```

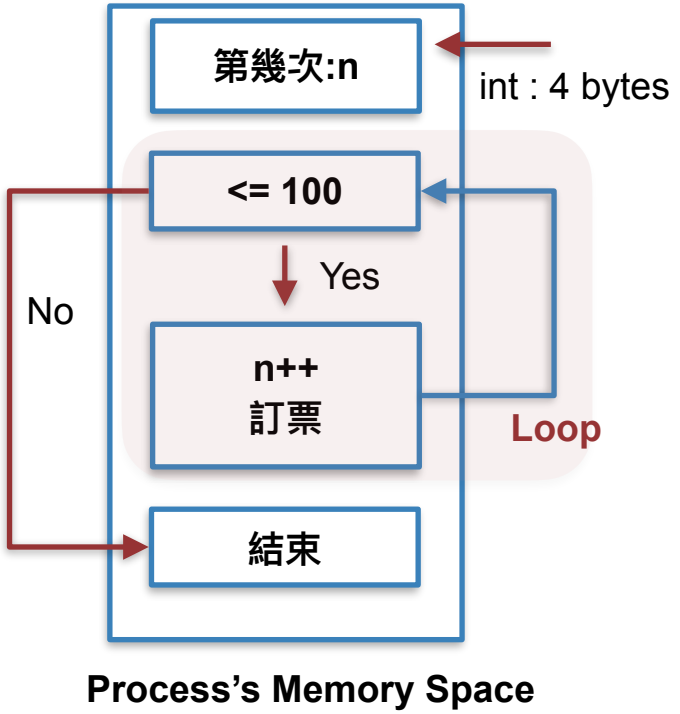
```
1 2 3 4 5 6 7 8 9 10

...Program finished with exit code 0
Press ENTER to exit console.
```



Process's Memory Space

Flow of Control: 迴圈

- 假設你要去聽IU演唱會100次
- 要訂票100次
- 寫一隻程式去訂票
  - 要能做100次訂票動作！
  - 所以要計算定幾次了
  - 每次都要做定票動作



while

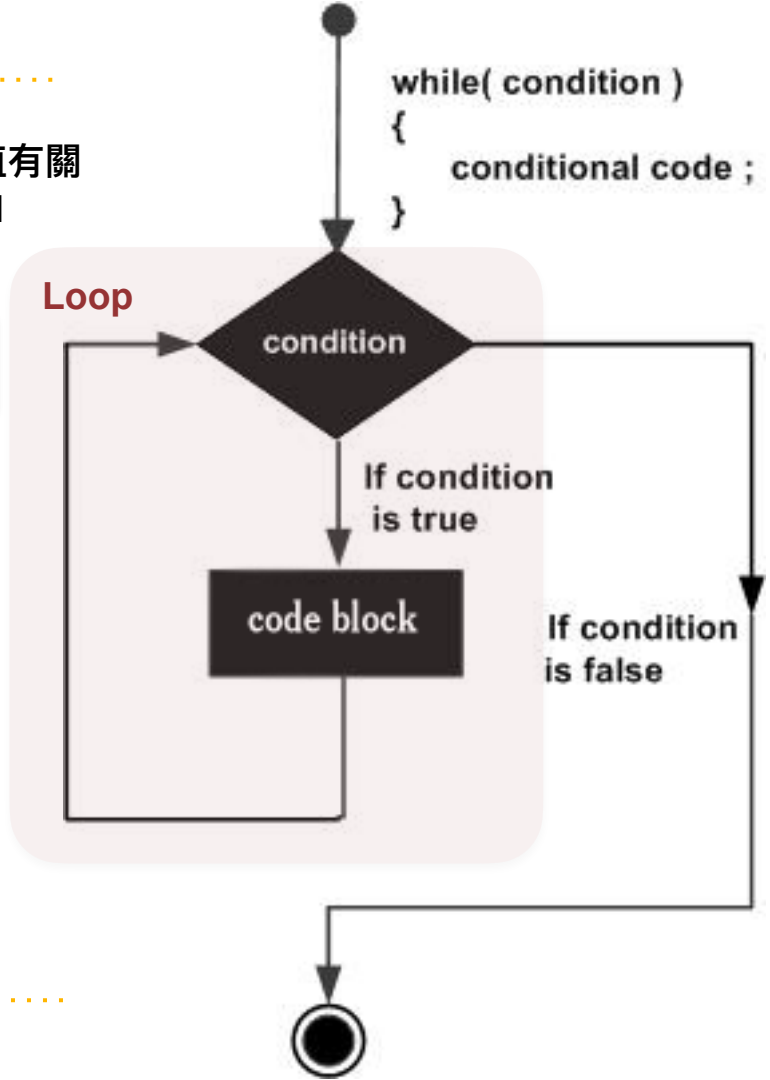
和 n 的初始值有關
n = 0 or n = 1

第幾次:n < 100

條件是否成立

```
while(condition) {
    statement(s);
}
```

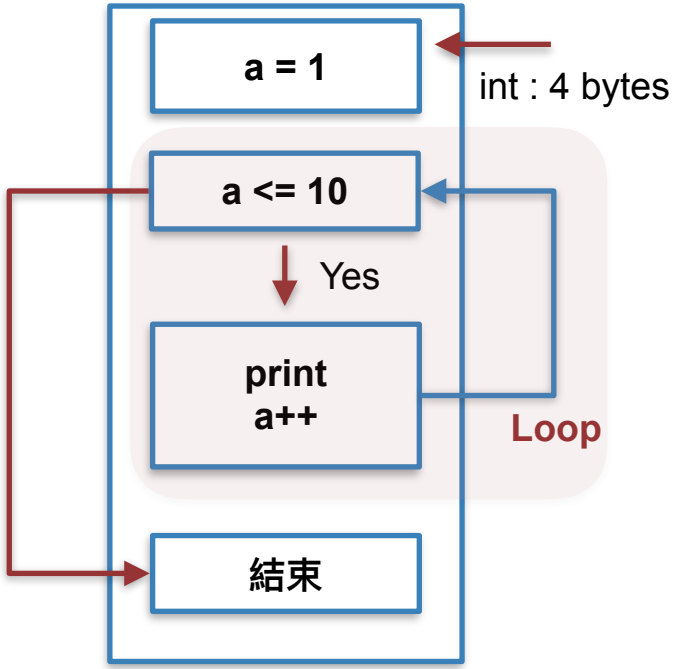
要重複的動作



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

    int a = 1;
    while( a <= 10 ) {
        printf("value of a: %d\n", a);
        a++;
    }

    return 0;
}
```

input

```
value of a: 1
value of a: 2
value of a: 3
value of a: 4
value of a: 5
value of a: 6
value of a: 7
value of a: 8
value of a: 9
value of a: 10


...Program finished with exit code 0
Press ENTER to exit console.
```



Process's Memory Space

for loop

```
for( init; condition; increment )
{
    conditional code ;
}
```

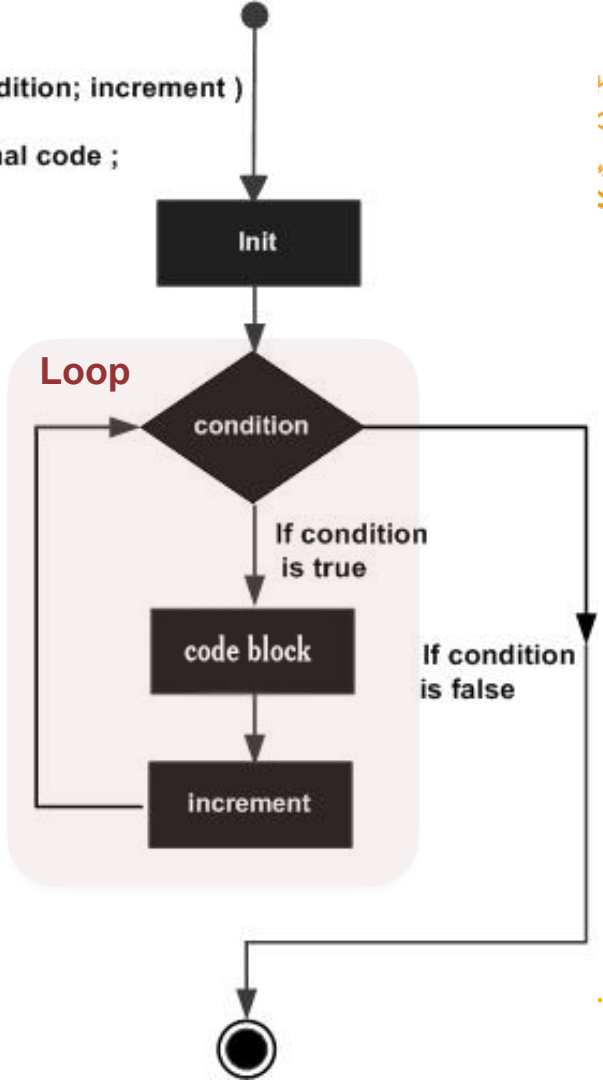
條件是否成立

初始輪迴參數值

```
for ( init; condition; increment ) {
   statement(s);
}
```

修改下一個輪迴參數值

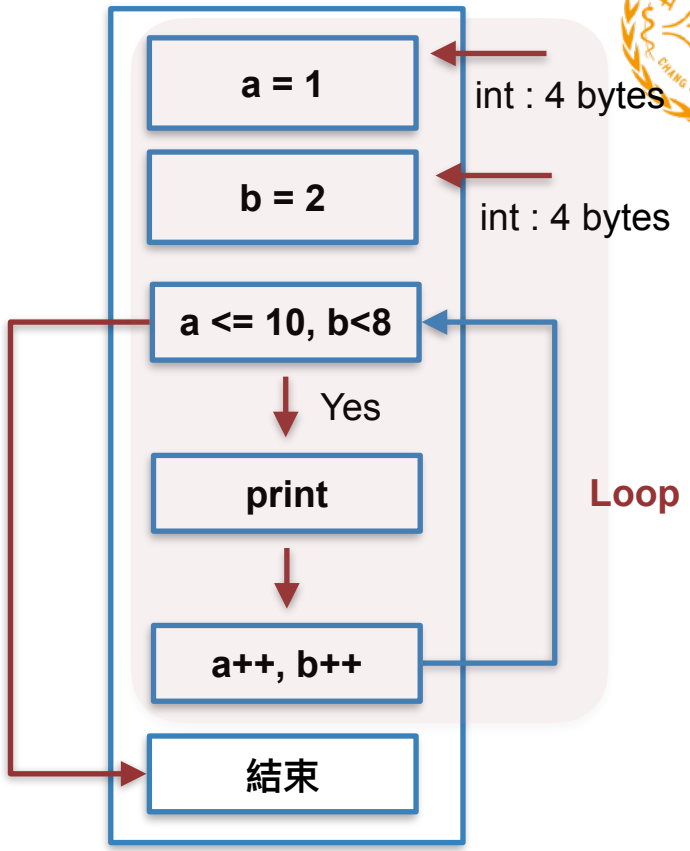
要重複的動作



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

    for ( int a=1, b=2; a <= 10, b<8; a++, b++ ){
        printf("value of a, b: %d, %d\n", a, b);
    }

    return 0;
}
```

input

```
value of a, b: 1, 2
value of a, b: 2, 3
value of a, b: 3, 4
value of a, b: 4, 5
value of a, b: 5, 6
value of a, b: 6, 7


...Program finished with exit code 0
Press ENTER to exit console.
```



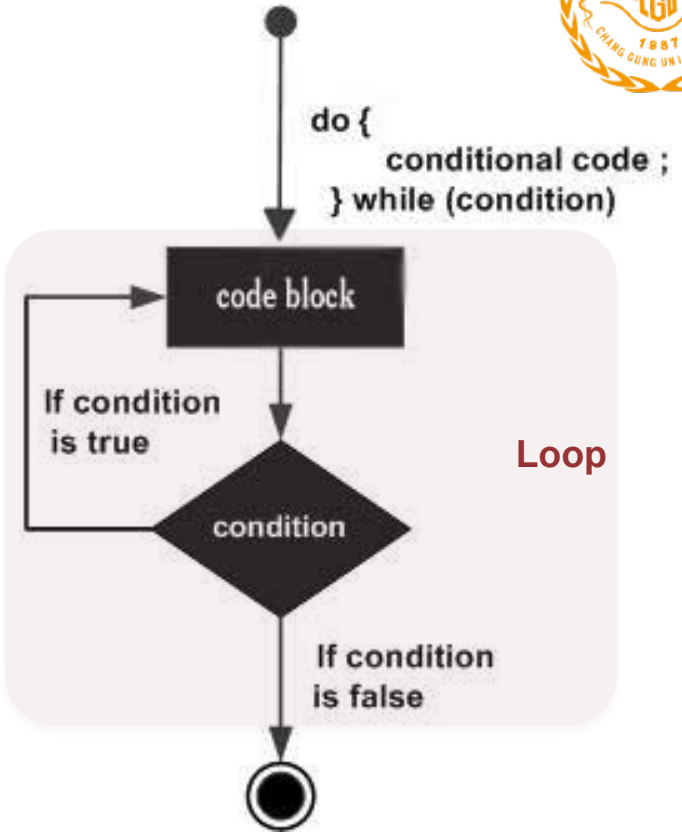
Process's Memory Space

Do …. While

要重複的動作

```
do {
    statement(s);
} while( condition );
```

條件是否成立

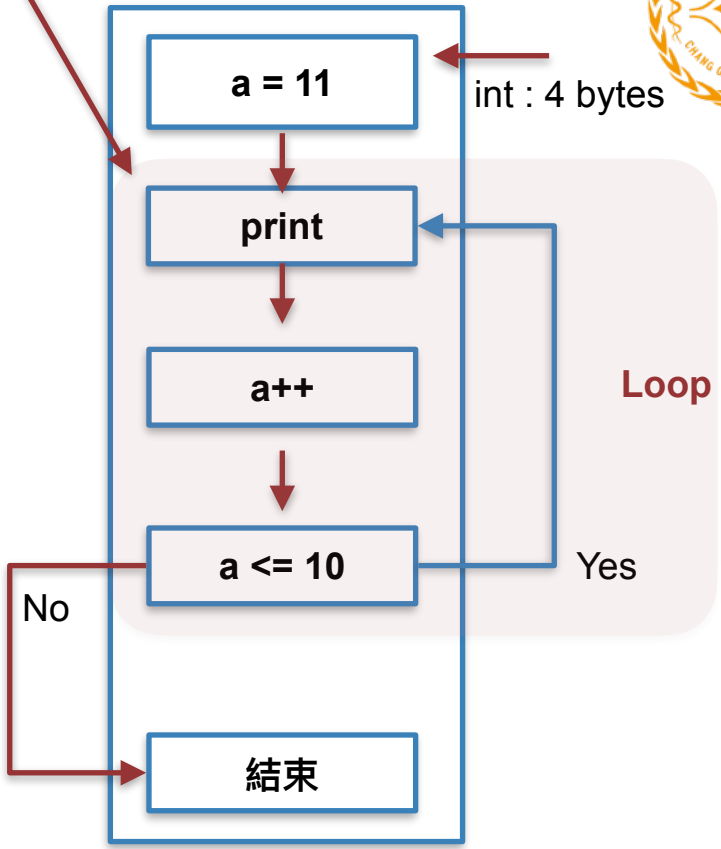


鐵定會做一次

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

    int a = 11;

    do {
        printf("value of a: %d\n", a);
        a++;
    } while ( a <= 10 );

    return 0;
}
```

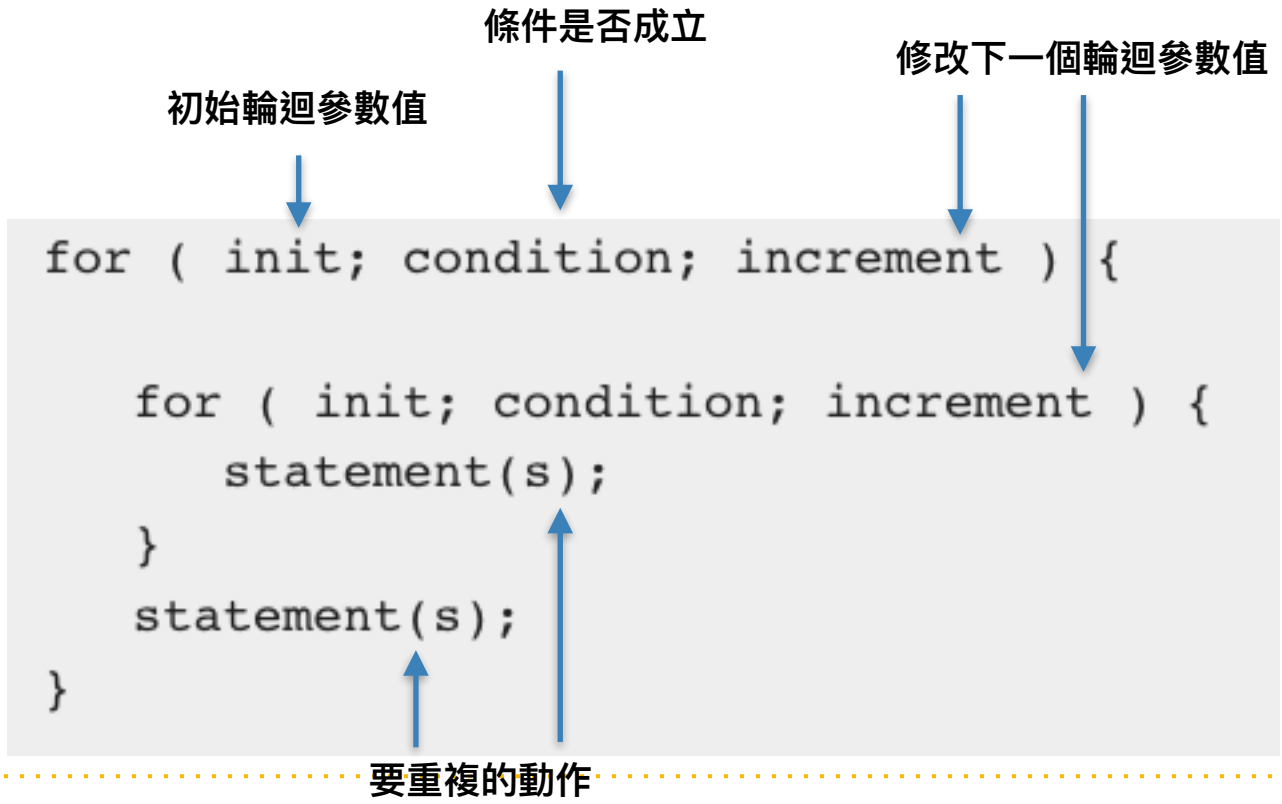
input

```
value of a: 11


...Program finished with exit code 0
Press ENTER to exit console.
```



Process's Memory Space

nested loop

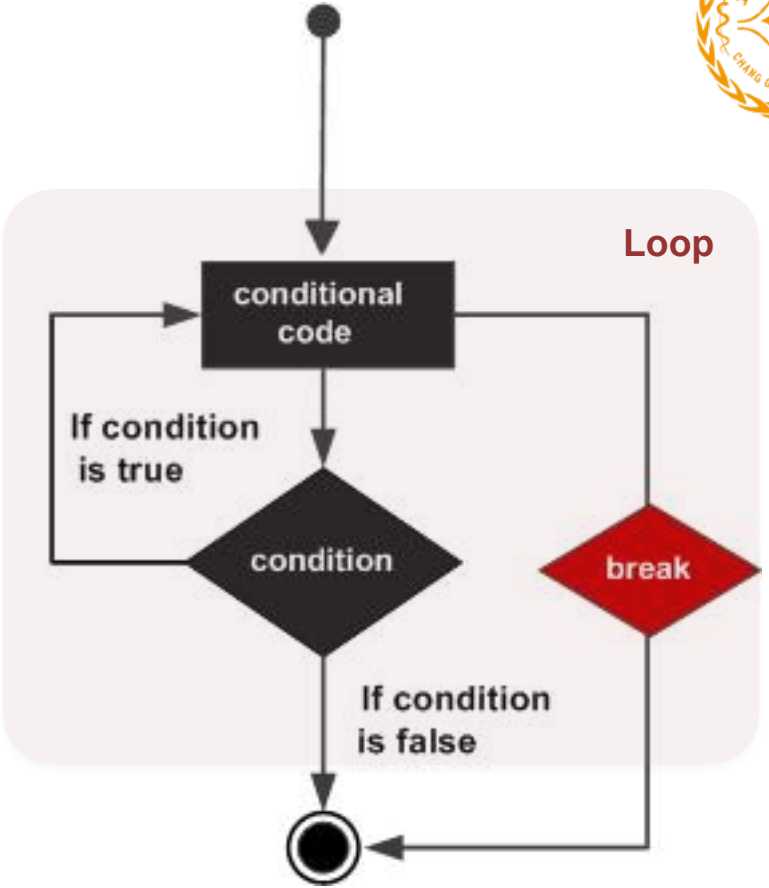

條件是否成立
初始輪迴參數值
修改下一個輪迴參數值
for (init; condition; increment) {
for (init; condition; increment) {
statement(s);
}
statement(s);
}
要重複的動作

迴圈中有迴圈

```
for ( init; condition; increment ) {

    for ( init; condition; increment ) {
        statement(s);
    }                                    Loop
    statement(s);
}                                        Loop
```

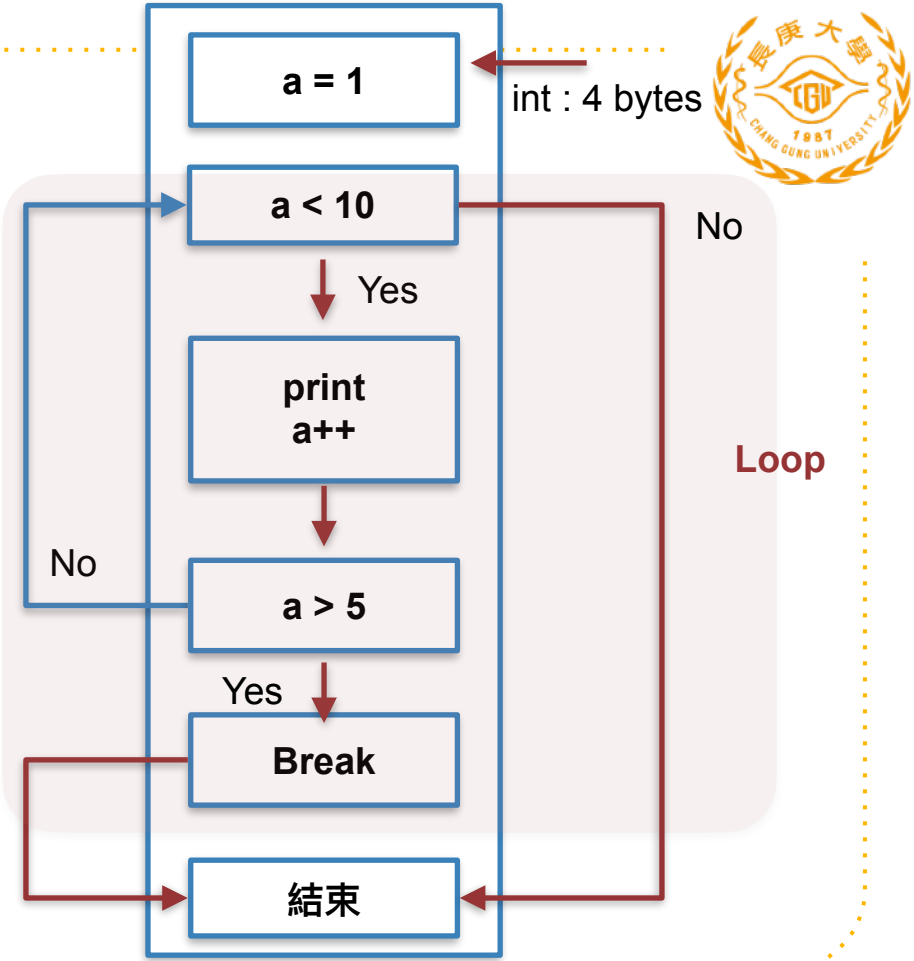
Break

- 原本訂票100次
- 但是假如錢不夠時
  - 不能訂了
  - 要終止訂票迴圈



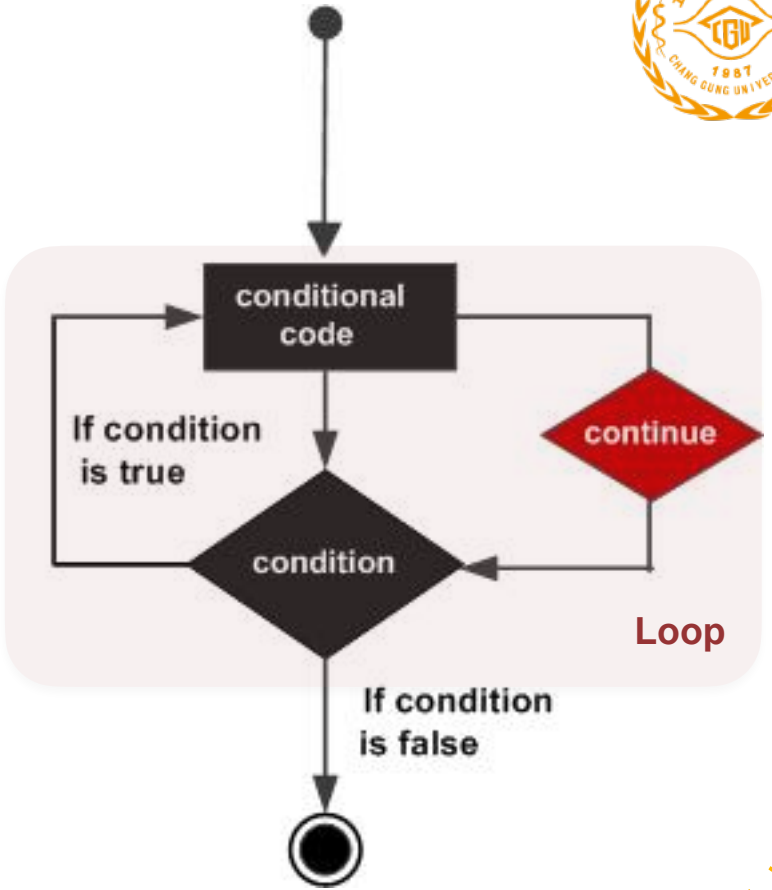
```c
int main() {
    int a = 1;
    while (a < 10) {
        printf("value of a: %d\n", a);
        a++;

        if (a > 5) {
            break;
        }
    }

    return 0;
}
```

```
value of a: 1
value of a: 2
value of a: 3
value of a: 4
value of a: 5
```



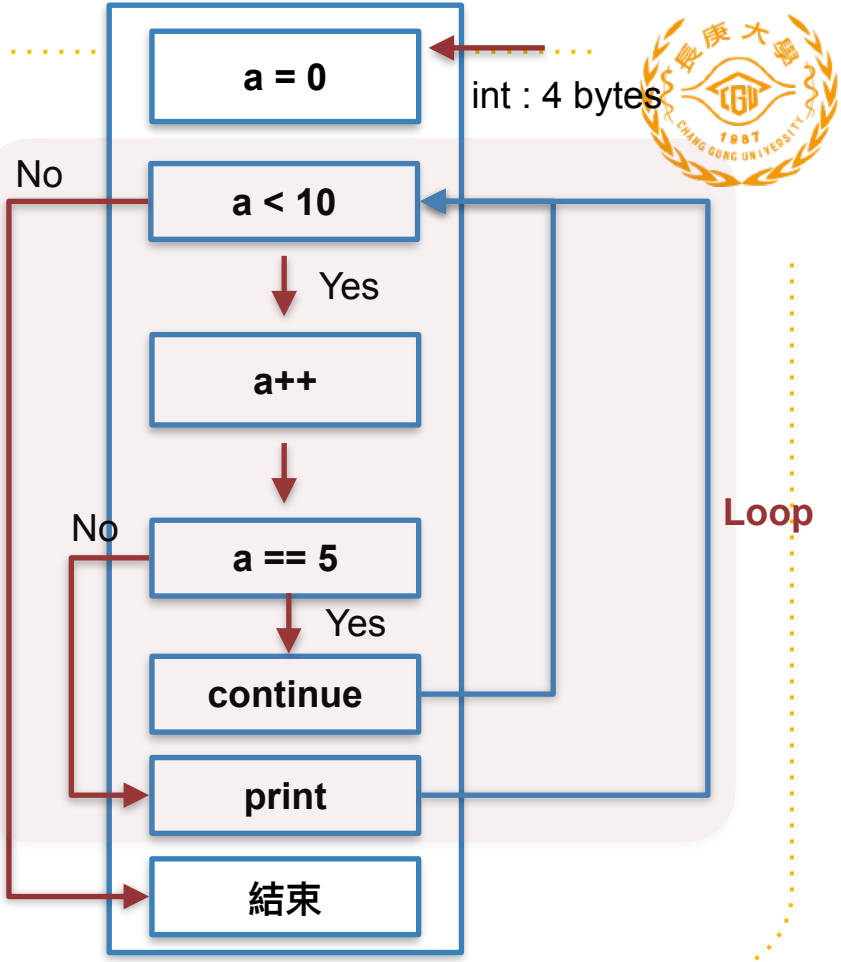
Continue

- 原本訂票100次
- 但是第4場不要訂
  - 第4場要跳過
  - 其他都要定



```c
int main() {
    int a = 0;
    while (a < 10) {
        a++;
        if (a == 5) {
            continue;
        }
        printf("value of a: %d\n", a);
    }

    return 0;
}
```

```
value of a: 1
value of a: 2
value of a: 3
value of a: 4
value of a: 6
value of a: 7
value of a: 8
value of a: 9
value of a: 10
```



```c
#include <stdio.h>
int main() {
    unsigned long factorial = 1;
    for (int i=1; i <=20; i++) {
        factorial *= i;
        printf("%i! = %lu\n", i, factorial);
    }

    return 0;
}
```

input

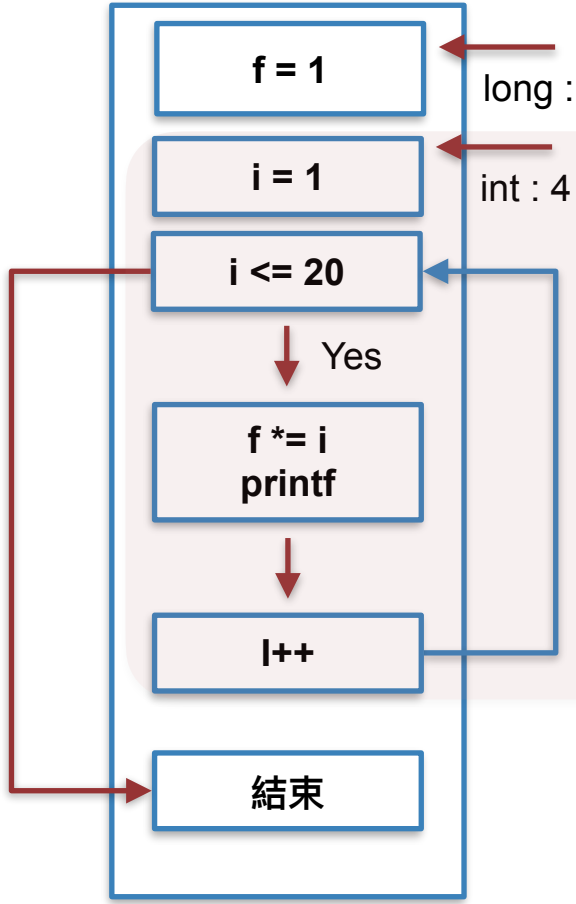
```
1! = 1
2! = 2
3! = 6
4! = 24
5! = 120
6! = 720
7! = 5040
8! = 40320
9! = 362880
10! = 3628800
11! = 39916800
12! = 479001600
13! = 6227020800
14! = 87178291200
15! = 1307674368000
16! = 20922789888000
17! = 355687428096000
18! = 6402373705728000
19! = 121645100408832000
20! = 2432902008176640000

...Program finished with exit code 0
Press ENTER to exit console.
```

Factorial
算階乘
20!

```c
#include <stdio.h>
int main() {
    unsigned long factorial = 1;
    for (int i=1; i <=20; i++) {
        factorial *= i;
        printf("%i! = %lu\n", i, factorial);
    }

    return 0;
}
```

input

```
1! = 1
2! = 2
3! = 6
4! = 24
5! = 120
6! = 720
7! = 5040
8! = 40320
9! = 362880
10! = 3628800
11! = 39916800
12! = 479001600
13! = 6227020800
14! = 87178291200
15! = 1307674368000
16! = 20922789888000
17! = 355687428096000
18! = 6402373705728000
19! = 121645100408832000
20! = 2432902008176640000

...Program finished with exit code 0
Press ENTER to exit console.
```



i 包含在迴圈裡喔

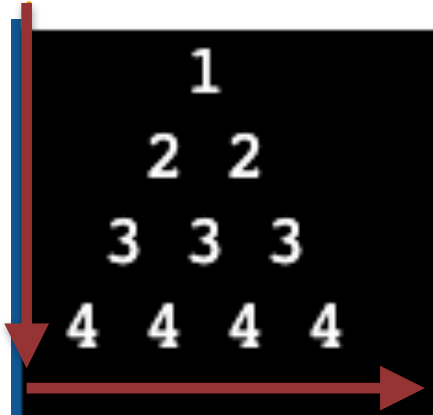
Loop

Process's Memory Space

印出數字金字塔


1
2 2
3 3 3
4 4 4 4
...Program finished with exit code 0
Press ENTER to exit console.

分析一下

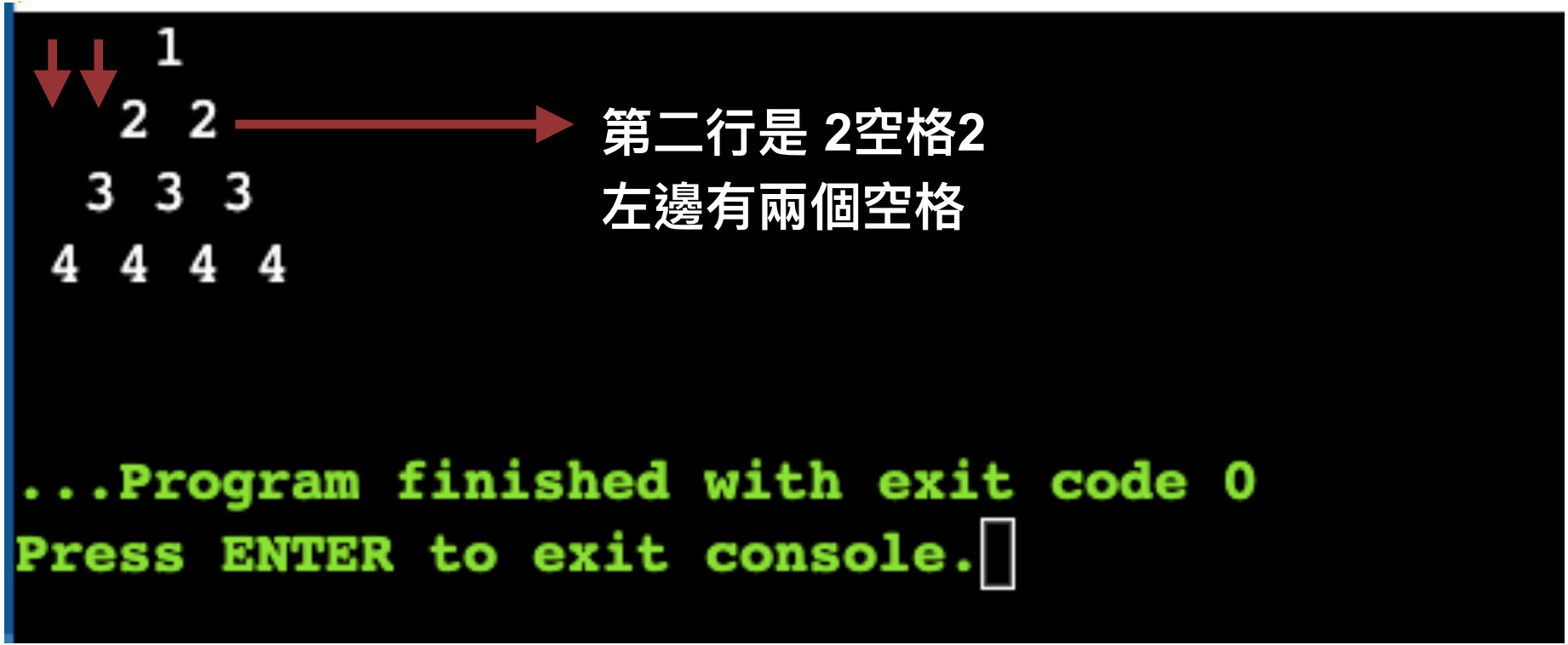


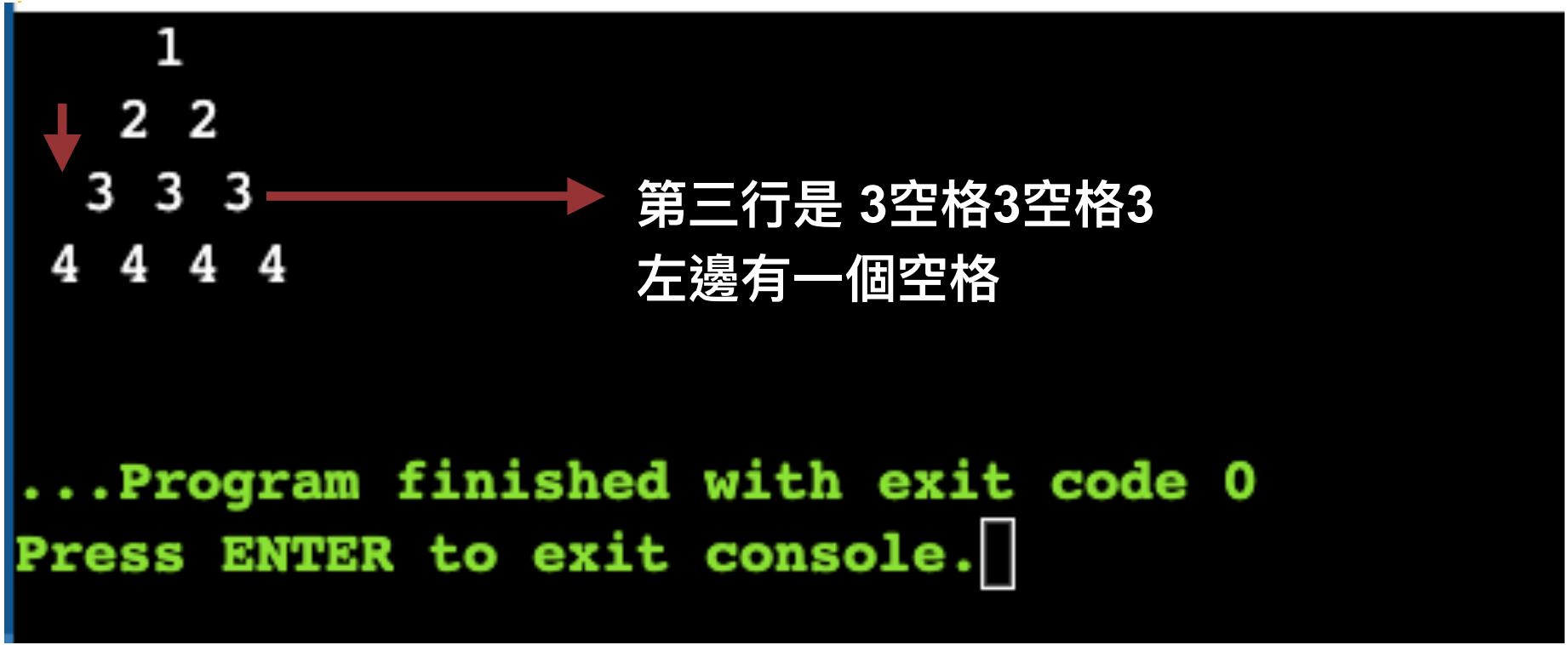
從上面印下來
從左邊印過去

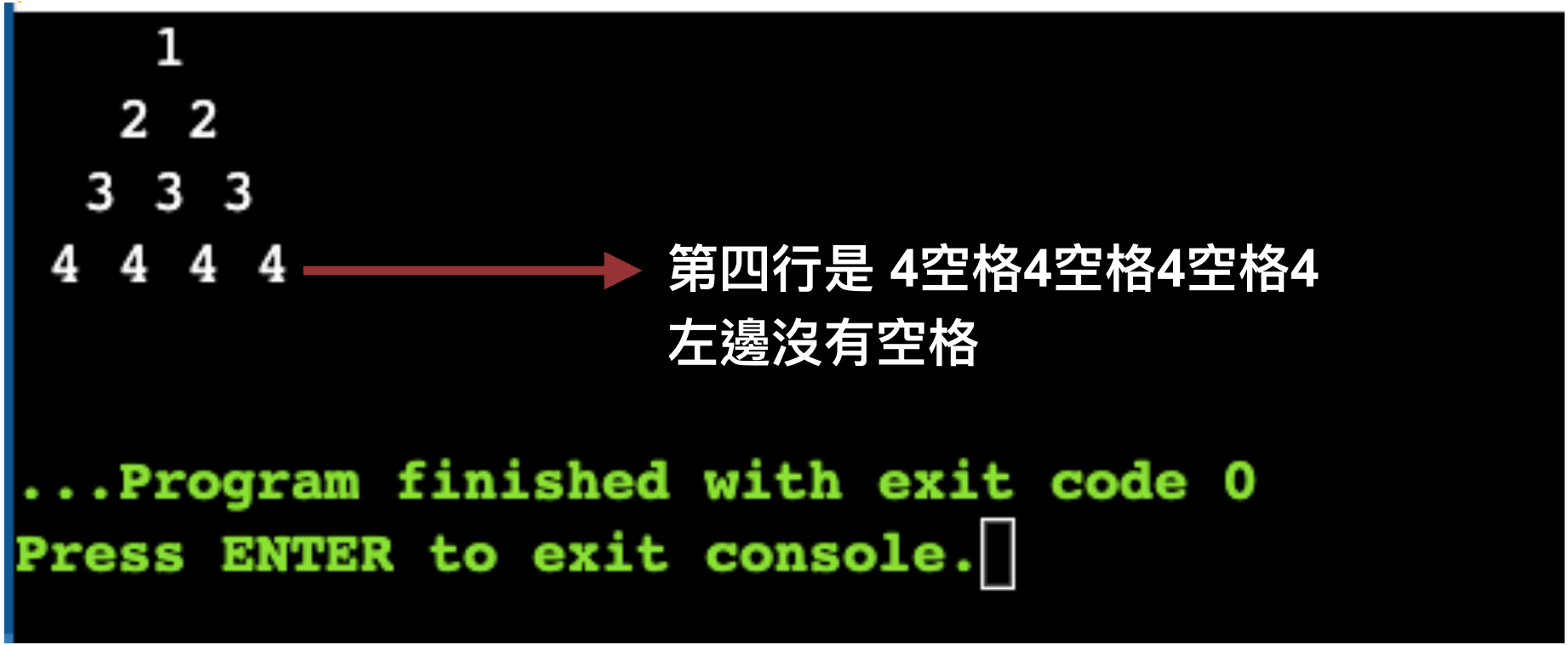
從上面印下來
印到4
所以印了四行，每次加一

```
...Program finished with exit code 0
Press ENTER to exit console.
```


1
2 2
3 3 3
4 4 4 4
第一行是 1
左邊有三個空格
...Program finished with exit code 0
Press ENTER to exit console.

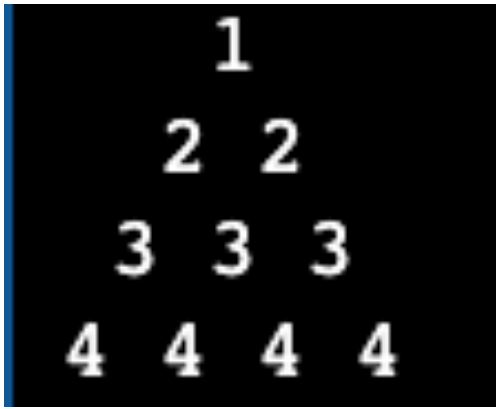

1
2 2
3 3 3
4 4 4 4
第二行是 2空格2
左邊有兩個空格
...Program finished with exit code 0
Press ENTER to exit console.


1
2 2
3 3 3
4 4 4 4
第三行是 3空格3空格3
左邊有一個空格
...Program finished with exit code 0
Press ENTER to exit console.


1
2 2
3 3 3
4 4 4 4
第四行是 4空格4空格4空格4
左邊沒有空格
...Program finished with exit code 0
Press ENTER to exit console.

印出數字金字塔

- For i=1 to n, 需要一個 loop
  - 計算幾個空白
  - 計算 印 i 次 i



印出數字金字塔

```c
#include <stdio.h>

int main() {
    int n = 4;

    for (int i=1;i<=n;i++) {
        printf("%d\n", i);
    }
}
```

印出第i層

```
1
2
3
4


...Program finished with exit code 0
Press ENTER to exit console.
```

```c
#include <stdio.h>

int main() {
    int n = 4;
    int spc = n;

    for (int i=1;i<=n;i++) {
        for (int k=spc;k>=1;k--) {
            printf(" ");
        }
        printf("%d\n", i);
        spc--;
    }
}
```

```
   1
  2
 3
4


...Program finished with exit code 0
Press ENTER to exit console.
```

印出計算的空白和第i層

```c
#include <stdio.h>

int main() {
    int n = 4;
    int spc = n;

    for (int i=1;i<=n;i++) {
        for (int k=spc;k>=1;k--) {
            printf(" ");
        }
        for (int j=1;j<=i;j++) {
            printf("%d ",i);
        }
        printf("\n");
        spc--;
    }
}
```

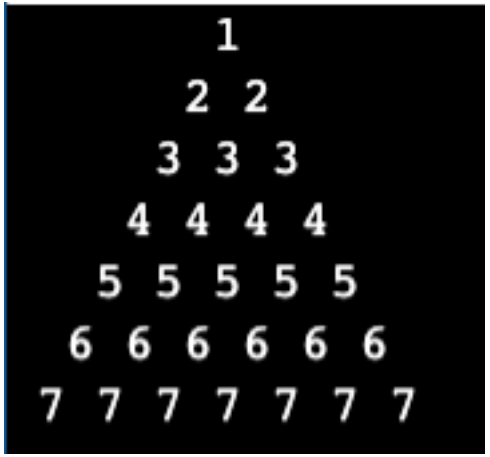
```
   1
  2 2
 3 3 3
4 4 4 4


...Program finished with exit code 0
Press ENTER to exit console.
```

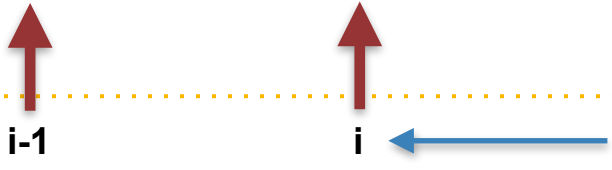
第i層 i 要印i次

Can we do better?

印出數字金字塔
只用兩個 Loop

- For i=1 to n, 需要一個 loop
  - 1 行是 n space + 1 次 “1”
  - 2 行是 n-1 space + 2 次 “2”
  - 3 行是 n-2 space + 3 次 “3”
  - 4 行是 n-3 space + 4 次 “4”





HW#7，12/29 12:00PM
Commit to your GitHub

- 印出數字金字塔
  - **兩個loop**
- **main1.c**

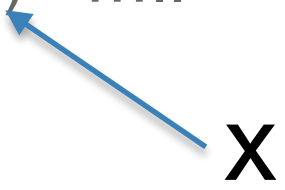
- **請注意！作業沒有輸入！**
- **如果程式需要輸入零分計算！**

```c
#include <stdio.h>

int main() {
    int n = 7;
    
    
    
    
    
    
    
    
    
    
    
    
    
    
    
    
    
}
```

```
      1
    2 2
  3 3 3
4 4 4 4
5 5 5 5 5
6 6 6 6 6 6
7 7 7 7 7 7 7


...Program finished with exit code 0
Press ENTER to exit console.
```

HW#7，12/29 12:00PM
Commit to your GitHub

- 計算PI by Gregory-Leibniz Series
- PI = (4/1) - (4/3) + (4/5) - (4/7) + (4/9) - …..
- 請利用 for loop 來計算 pi
- double PI = 4.0f;
- **找到最小的 x** 計算出來 = 3.14159 (**精準五位數**)
- **main2.c**



- **請注意！作業沒有輸入！**
- **如果程式需要輸入零分計算！**

HW#7，12/29 12:00PM
Commit to your GitHub

- 請注意！作業沒有輸入！
- 如果程式需要輸入零分計算！

- 九九乘法表
  - 兩個loop
  - **請改成一個loop**

- **main3.c**

```c
#include <stdio.h>

int main() {

    for (int y=1;y<=9;y++) {
        for (int x=1;x<=9;x++) {
            printf("%d*%d=%d\t",y,x,y*x );
        }
        printf("\n");
    }

    return 0;
}
```

```
1*1=1   1*2=2   1*3=3   1*4=4   1*5=5   1*6=6   1*7=7   1*8=8   1*9=9
2*1=2   2*2=4   2*3=6   2*4=8   2*5=10  2*6=12  2*7=14  2*8=16  2*9=18
3*1=3   3*2=6   3*3=9   3*4=12  3*5=15  3*6=18  3*7=21  3*8=24  3*9=27
4*1=4   4*2=8   4*3=12  4*4=16  4*5=20  4*6=24  4*7=28  4*8=32  4*9=36
5*1=5   5*2=10  5*3=15  5*4=20  5*5=25  5*6=30  5*7=35  5*8=40  5*9=45
6*1=6   6*2=12  6*3=18  6*4=24  6*5=30  6*6=36  6*7=42  6*8=48  6*9=54
7*1=7   7*2=14  7*3=21  7*4=28  7*5=35  7*6=42  7*7=49  7*8=56  7*9=63
8*1=8   8*2=16  8*3=24  8*4=32  8*5=40  8*6=48  8*7=56  8*8=64  8*9=72
9*1=9   9*2=18  9*3=27  9*4=36  9*5=45  9*6=54  9*7=63  9*8=72  9*9=81
```

HW#7，12/29 12:00PM
Commit to your GitHub

- 把一個大於零的數的千位數字和個位數字調換
  - int i = 12345;
  - 12345 —> 15342
  - 123 —> 3120
  - 12 —> 2010
  - 1 —> 1000
- **main4.c**

請把 i 預設為 12345
但是也要考慮 以下情況

- **請注意！作業沒有輸入！**
- **如果程式需要輸入零分計算！**

Conclusion

- C Basics
  - Variables
    - floating-point
  - Flow of Control
    - **While loop**
    - **For**
    - **Do while**
    - **Break**
    - **Continue**

Thanks!

Open for any questions

CJ Wu

cjwu@mail.cgu.edu.tw

```c
#include <stdio.h>

int main() {
    int n = 4;
    int spc = n;

    for (int i=1;i<=n;i++) {
        for (int k=spc;k>=1;k--) {
            printf(" ");
        }
        for (int j=1;j<=i;j++) {
            printf("%d ",i);
        }
        printf("\n");
        spc--;
    }
}
```

```
   1
  2 2
 3 3 3
4 4 4 4


...Program finished with exit code 0
Press ENTER to exit console.
```