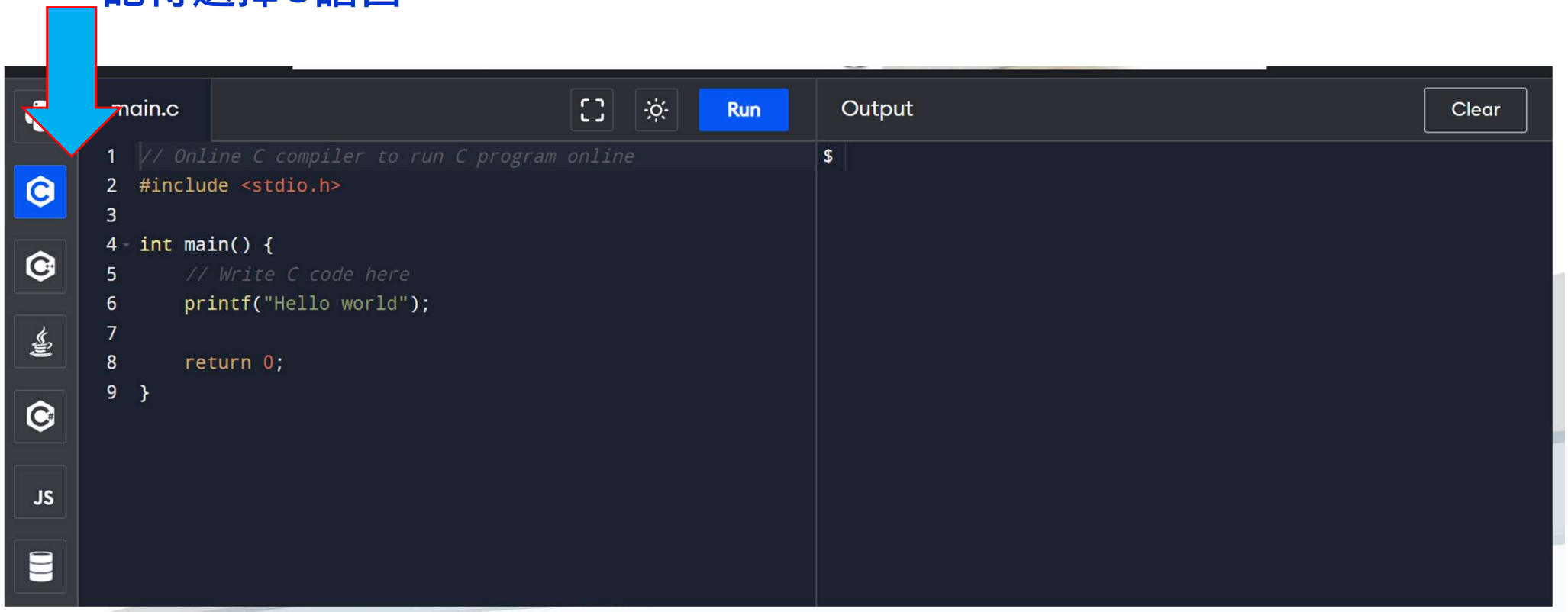
>>C語言 基本程式設計

C語言練習平台

- 可使用 online c complier 來簡單練習 C語言。
- https://www.programiz.com/c-programming/online-compiler/
- https://www.tutorialspoint.com/compile_c_online.php
- 練習在螢幕顯示文字 (printf) 指令
- 記得選擇C語言



基本概念

- C語言的程式是由 函式(function)所構成的。
- 一定要有 main()
- 記得在使用任何 function 之前, 要 include 該函式所屬的表頭檔.h
- include 怎麼用？
- 表頭檔是什麼？ C 檔與 Header 檔的差別
- 如果沒有在使用 function 前先 include 會如何？
- 養成好習慣，消除所有的 warning (*example*)

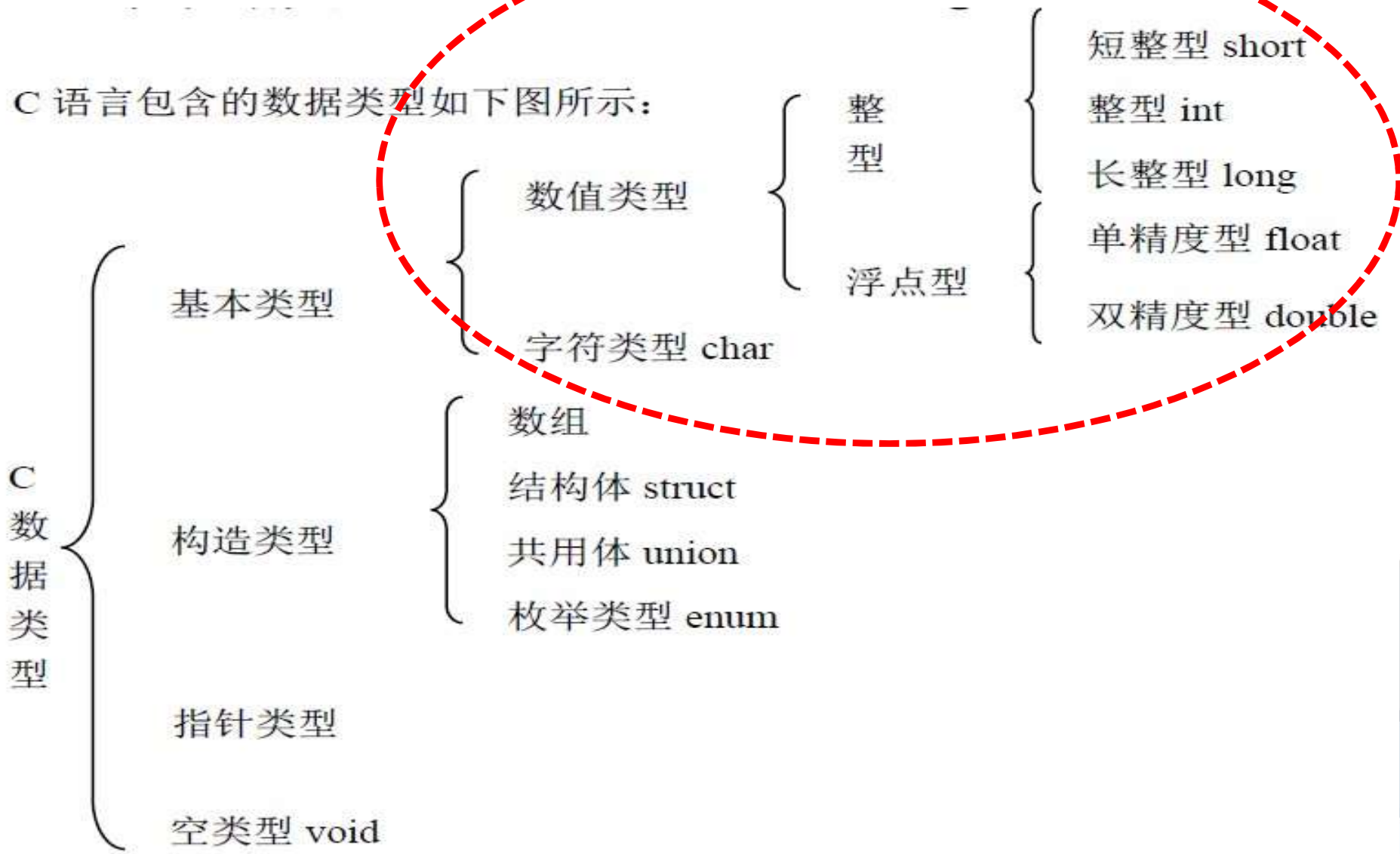
```
#include <文件名>
#include "文件名"
```

```
1  // Online C compiler to run C program online
2  #include <stdio.h>
3  #include <stdlib.h>
4
```

數據類型

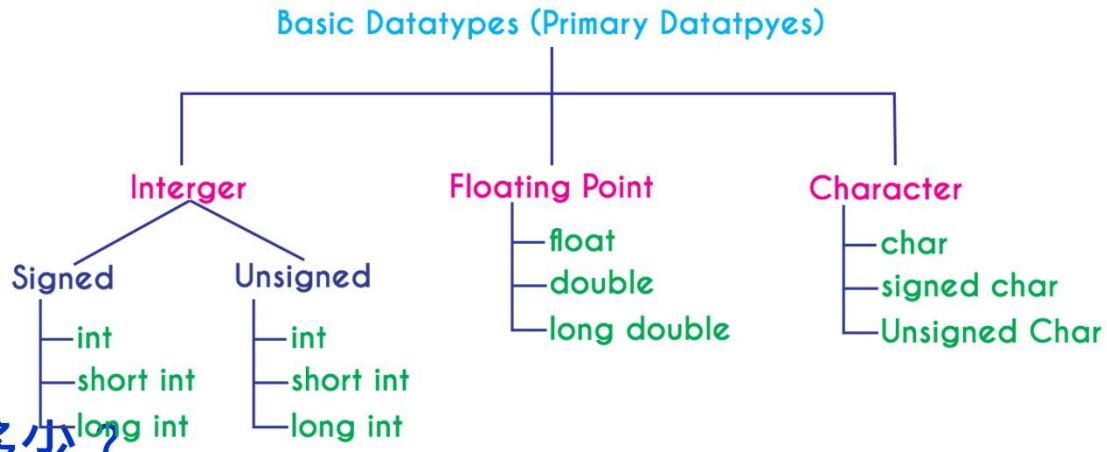
先瞭解基本類型！！

C 语言包含的数据类型如下图所示：



數據類型

- 數據類型佔用的位元
- 儲存類型
- 不同類型轉換
- 如何知道二進位值/16進位值為多少？
- 不同平台所佔空間可能不一樣



C Basic Data Types	32-bit CPU		64-bit CPU	
	Size (bytes)	Range	Size (bytes)	Range
char	1	-128 to 127	1	-128 to 127
short	2	-32,768 to 32,767	2	-32,768 to 32,767
int	4	-2,147,483,648 to 2,147,483,647	4	-2,147,483,648 to 2,147,483,647
long	4	-2,147,483,648 to 2,147,483,647	8	-9,223,372,036,854,775,808-9,223,372,036,854,775,807
long long	8	9,223,372,036,854,775,808-9,223,372,036,854,775,807	8	9,223,372,036,854,775,808-9,223,372,036,854,775,807
float	4	3.4E +/- 38	4	3.4E +/- 38
double	8	1.7E +/- 308	8	1.7E +/- 308

進位轉換（2進位 16進位）

2進位 vs. 16進位

2進位	16進位
0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111,1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
1111011 = 01111011 (寫成8個數字)	7B (將2-bit 每四個數字對照上列轉換一次)
小數部份(2-bit 每四個數字轉換一次)	
0.11 = 0.1100 (小數寫成4個數字)	0.C
整數與小數並存(將2-bit 每四個數字轉換一次)	
1111011.11 = 01111011.1100	7B.C

- **2進位的整數轉換成 10進位的整數.**

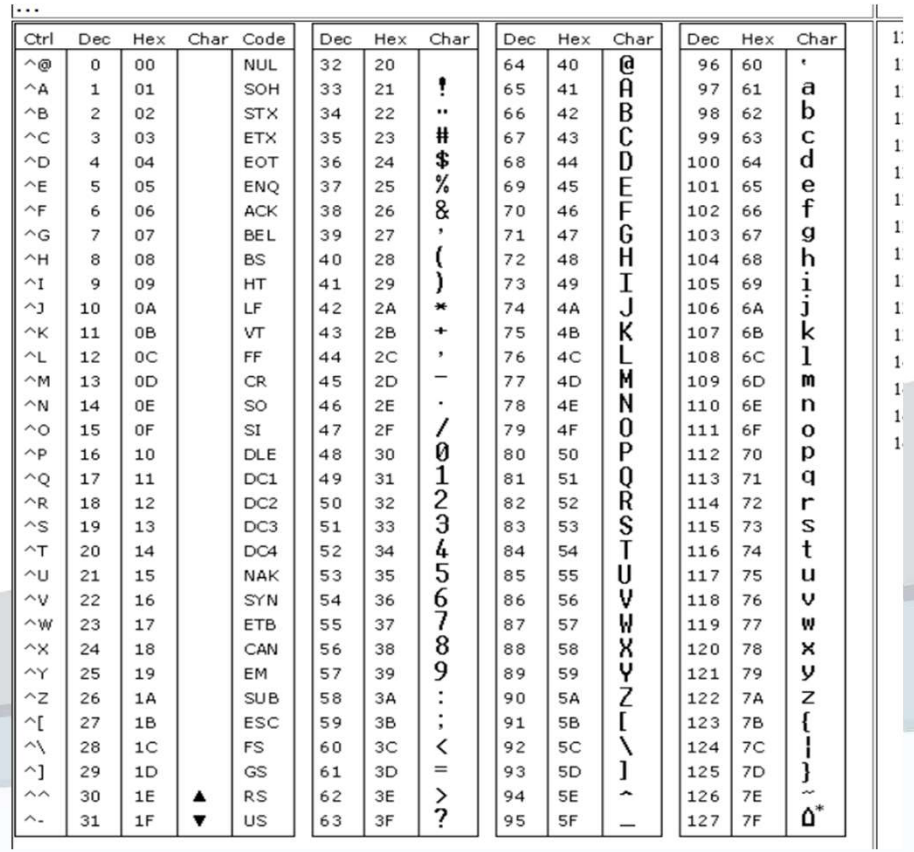
  (2 進位) 10101 = (10 進位) $1 \times 2^0 + 0 \times 2^1 + 1 \times 2^2 + 0 \times 2^3 + 1 \times 2^4 = 1 + 4 + 16 = 21$.

Char 類型

`char` 類型可用來儲存可顯示字元集之成員的整數值。 該整數值是對應所指定字元的 ASCII 碼。

Microsoft **特定的**

`unsigned char` 類型的字元值範圍是十六進位的 0 到 0xFF。 `signed char`的範圍0x80 0x7F。 這些範圍會分別轉譯為十進位的 0 到 255 及十進位的 -128 到 +127。 /J 編譯器選項會將預設值從 `signed` 變更為 `unsigned` 。



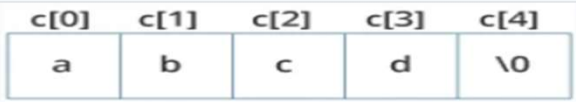
Ctrl	Dec	Hex	Char	Code	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
^@	0	00		NUL	32	20		64	40	@	96	60	`
^A	1	01		SOH	33	21	!	65	41	A	97	61	a
^B	2	02		STX	34	22	"	66	42	B	98	62	b
^C	3	03		ETX	35	23	#	67	43	C	99	63	c
^D	4	04		EOT	36	24	$	68	44	D	100	64	d
^E	5	05		ENQ	37	25	%	69	45	E	101	65	e
^F	6	06		ACK	38	26	&	70	46	F	102	66	f
^G	7	07		BEL	39	27	'	71	47	G	103	67	g
^H	8	08		BS	40	28	(	72	48	H	104	68	h
^I	9	09		HT	41	29	)	73	49	I	105	69	i
^J	10	0A		LF	42	2A	*	74	4A	J	106	6A	j
^K	11	0B		VT	43	2B	+	75	4B	K	107	6B	k
^L	12	0C		FF	44	2C	,	76	4C	L	108	6C	l
^M	13	0D		CR	45	2D	-	77	4D	M	109	6D	m
^N	14	0E		SO	46	2E	.	78	4E	N	110	6E	n
^O	15	0F		SI	47	2F	/	79	4F	O	111	6F	o
^P	16	10		DLE	48	30	0	80	50	P	112	70	p
^Q	17	11		DC1	49	31	1	81	51	Q	113	71	q
^R	18	12		DC2	50	32	2	82	52	R	114	72	r
^S	19	13		DC3	51	33	3	83	53	S	115	73	s
^T	20	14		DC4	52	34	4	84	54	T	116	74	t
^U	21	15		NAK	53	35	5	85	55	U	117	75	u
^V	22	16		SYN	54	36	6	86	56	V	118	76	v
^W	23	17		ETB	55	37	7	87	57	W	119	77	w
^X	24	18		CAN	56	38	8	88	58	X	120	78	x
^Y	25	19		EM	57	39	9	89	59	Y	121	79	y
^Z	26	1A		SUB	58	3A	:	90	5A	Z	122	7A	z
^[	27	1B		ESC	59	3B	;	91	5B	[	123	7B	{
^\	28	1C		FS	60	3C	<	92	5C	\	124	7C	|
^]	29	1D		GS	61	3D	=	93	5D	]	125	7D	}
^^	30	1E	▲	RS	62	3E	>	94	5E	^	126	7E	~
^-	31	1F	▼	US	63	3F	?	95	5F	_	127	7F	⌂

字串 類型 string

- 在 C 語言中，字串實際上是使用空字符 \0 結尾的一維字串數組。因此，\0 是用於標記字串的結束。
- 空字符（Null character）又稱結束符，縮寫 NUL，是一個數值為 0 的控制字符，\0 是轉義字符，意思是告訴編譯器，這不是字符 0，而是空字符。

字串初始化

```
char c[] = "abcd";
char c[50] = "abcd";
char c[] = {'a', 'b', 'c', 'd', '\0'};
char c[5] = {'a', 'b', 'c', 'd', '\0'};
```



String Initialization in C

註解

・註解可以用兩種方法

// -> 單行註解

/* */ -> 多行註解

```
mcp.portMode(MCP23017Port::B, 0b11111111);  //Port B as input
mcp.portMode(MCP23017Port::A, 0b11111111);  //Port B as input

mcp.writeRegister(MCP23017Register::GPIO_A, 0x00);  //Reset port A
mcp.writeRegister(MCP23017Register::GPIO_B, 0x00);  //Reset port B
```

單行註解

```
47  /* Private function prototypes -----------------------------------------------*/
48  /* Private functions ---------------------------------------------------------*/
49
50  /******************************************************************************/
51  /*            Cortex-M3 Processor Exceptions Handlers                         */
52  /******************************************************************************/
53
54  /**
55    * @brief  This function handles NMI exception.
56    * @param  None
57    * @retval None
58    */
59  void NMI_Handler(void)
60  {
61  }
62
```

多行註解

輸入與輸出 input/output：*printf()*

printf 函数的原型为：

```
# include <stdio.h>
int printf(const char *format, ...);
```

Format specifiers

- `%d` for `int`
- `%f` for `float`
- `%lf` for `double`
- `%c` for `char`

1) printf("字符串\n");
2) printf("輸出控制符"，輸出變數);
3) printf("輸出控制符1 輸出控制符2 ..."，輸出變數1，輸出變數2，...);

範例：

```c
// Online C compiler to run C program online
#include <stdio.h>
#include "stdlib.h"

int main() {
    // Write C code here
    printf("Hello world\n");

    int i=99;
    printf("The integer is %d \n",i);
    float j=0.99;
    printf("The float number is %f\n",j);
    char c='k';
    printf("The char is %c\n",c);

    return 0;
}
```

```
/tmp/Dd2wDNV4zj.o
Hello world
The integer is 99
The float number is 0.990000
The char is k
```

輸入與輸出 input/output：

Format specifiers

控制符	说明
%d	按十进制整型数据的实际长度输出。
%ld	输出长整型数据。
%md	m 为指定的输出字段的宽度。如果数据的位数小于 m，则左端补以空格，若大于 m，则按实际位数输出。
%u	输出无符号整型（unsigned）。输出无符号整型时也可以用 %d，这时是将无符号转换成有符号数，然后输出。但编程的时候最好不要这么写，因为这样要进行一次转换，使 CPU 多做一次无用功。
%c	用来输出一个字符。
%f	用来输出实数，包括单精度和双精度，以小数形式输出。不指定字段宽度，由系统自动指定，整数部分全部输出，小数部分输出 6 位，超过 6 位的四舍五入。
%.mf	输出实数时小数点后保留 m 位，注意 m 前面有个点。
%o	以八进制整数形式输出，这个就用得很少了，了解一下就行了。
%s	用来输出字符串。用 %s 输出字符串同前面直接输出字符串是一样的。但是此时要先定义字符数组或字符指针存储或指向字符串。

輸入與輸出 input/output： *scanf()*

- C語言可以用 scanf（ ）來輸入 ->通過鍵盤給程序中的变數賦值
- 記得變數前加上 取址符號 &
- 透過%c, %d, %s等來決定輸入的資料型態

Declaration

Following is the declaration for scanf() function.

```
int scanf(const char *format, ...)
```

Example:

int var;

scanf("%d", &var);

The scanf will write the value input by the user into the integer variable ***var.***

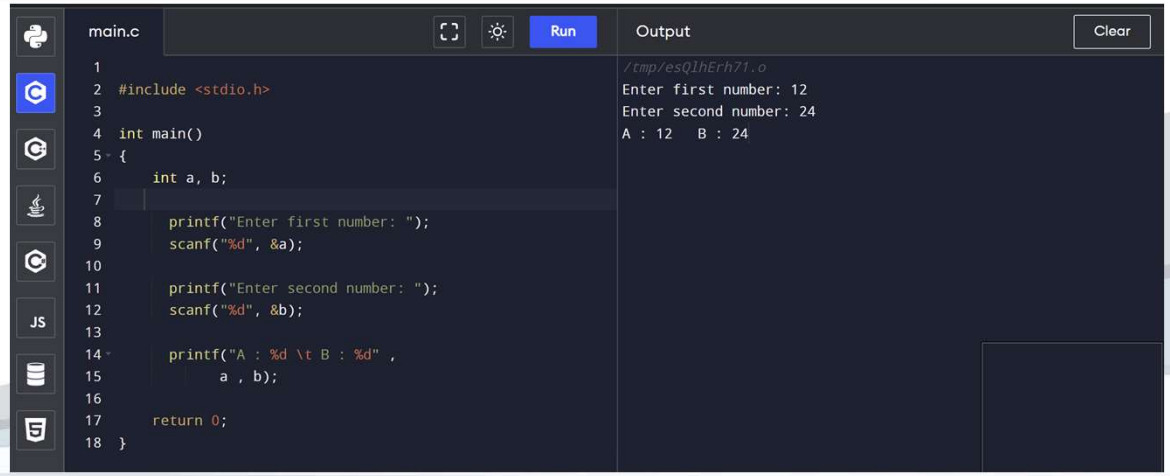
%d *to accept input of integers.*

%f *to accept input of real number.*

%c *to accept input of character types.*

%s *to accept input of a string.*

範例:



輸入與輸出 input/output：練習題

- 編寫一個程式，先提示使用者輸入名（%s 字串），在提示使用者輸入姓（%s 字串），最後印出使用者的名字與分別的字母數目。

提示:可以使用 string.h 內的 strlen() 來獲得字串的長度。（#include）

螢幕輸出結果:

```
James Yao
    5   3
```

C 库函数 - strlen()

C 标准库 - <string.h>

描述

C 库函数 **size_t strlen(const char *str)** 计算字符串 **str** 的长度，直到空结束字符，但不包括空结束字符。

声明

下面是 strlen() 函数的声明。

```
size_t strlen(const char *str)
```

Arithmetic Operators 算術運算子

Operator	Description	Example
+	Adds two operands.	A + B = 30
−	Subtracts second operand from the first.	A − B = -10
*	Multiplies both operands.	A * B = 200
/	Divides numerator by de-numerator.	B / A = 2
%	Modulus Operator and remainder of after an integer division.	B % A = 0
++	Increment operator increases the integer value by one.	A++ = 11
--	Decrement operator decreases the integer value by one.	A-- = 9

練習: Arithmetic Operators

```
3  main() {
4
5      int a = 21;
6      int b = 10;
7      int c ;
8      c = a + b;
9      printf("Line 1 - Value of c is %d\n", c );
10     c = a - b;
11     printf("Line 2 - Value of c is %d\n", c );
12     c = a * b;
13     printf("Line 3 - Value of c is %d\n", c );
14     c = a / b;
15     printf("Line 4 - Value of c is %d\n", c );
16     c = a % b;
17     printf("Line 5 - Value of c is %d\n", c );
18     c = a++;
19     printf("Line 6 - Value of c is %d\n", c );
20     c = a--;
21     printf("Line 7 - Value of c is %d\n", c );
22 }
```

Relational Operators 關係運算子

Operator	Description	Example
==	Checks if the values of two operands are equal or not. If yes, then the condition becomes true.	(A == B) is not true.
!=	Checks if the values of two operands are equal or not. If the values are not equal, then the condition becomes true.	(A != B) is true.
>	Checks if the value of left operand is greater than the value of right operand. If yes, then the condition becomes true.	(A > B) is not true.
<	Checks if the value of left operand is less than the value of right operand. If yes, then the condition becomes true.	(A < B) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand. If yes, then the condition becomes true.	(A >= B) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand. If yes, then the condition becomes true.	(A <= B) is true.

練習:Relational Operators

```c
#include <stdio.h>

main() {

   int a = 21;
   int b = 10;
   int c ;

   if( a == b ) {
      printf("Line 1 - a is equal to b\n" );
   } else {
      printf("Line 1 - a is not equal to b\n" );
   }

   if ( a < b ) {
      printf("Line 2 - a is less than b\n" );
   } else {
      printf("Line 2 - a is not less than b\n" );
   }

   if ( a > b ) {
      printf("Line 3 - a is greater than b\n" );
   } else {
      printf("Line 3 - a is not greater than b\n" );
   }

}
```

Logical Operators 邏輯運算子

Operator	Description	Example
&&	Called Logical AND operator. If both the operands are non-zero, then the condition becomes true.	(A && B) is false.
||	Called Logical OR Operator. If any of the two operands is non-zero, then the condition becomes true.	(A || B) is true.
!	Called Logical NOT Operator. It is used to reverse the logical state of its operand. If a condition is true, then Logical NOT operator will make it false.	!(A && B) is true.

練習: Logical Operators

```c
#include <stdio.h>

main() {

   int a = 5;
   int b = 20;
   int c ;

   if ( a && b ) {
      printf("Line 1 - Condition is true\n" );
   }

   if ( a || b ) {
      printf("Line 2 - Condition is true\n" );
   }

   /* lets change the value of  a and b */
   a = 0;
   b = 10;

   if ( a && b ) {
      printf("Line 3 - Condition is true\n" );
   } else {
      printf("Line 3 - Condition is not true\n" );
   }

   if ( !(a && b) ) {
      printf("Line 4 - Condition is true\n" );
   }

}
```

Bitwise Operators 位元運算子

Operator	Description	Example
&	Binary AND Operator copies a bit to the result if it exists in both operands.	(A & B) = 12, i.e., 0000 1100
|	Binary OR Operator copies a bit if it exists in either operand.	(A | B) = 61, i.e., 0011 1101
^	Binary XOR Operator copies the bit if it is set in one operand but not both.	(A ^ B) = 49, i.e., 0011 0001
~	Binary One's Complement Operator is unary and has the effect of 'flipping' bits.	(~A) = ~(60), i.e,. -0111101
<<	Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand.	A << 2 = 240 i.e., 1111 0000
>>	Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand.	A >> 2 = 15 i.e., 0000 1111

練習: Bitwise Operators

```c
#include <stdio.h>

main() {

   unsigned int a = 60; /* 60 = 0011 1100 */
   unsigned int b = 13; /* 13 = 0000 1101 */
   int c = 0;

   c = a & b;        /* 12 = 0000 1100 */
   printf("Line 1 - Value of c is %d\n", c );

   c = a | b;        /* 61 = 0011 1101 */
   printf("Line 2 - Value of c is %d\n", c );

   c = a ^ b;        /* 49 = 0011 0001 */
   printf("Line 3 - Value of c is %d\n", c );

   c = ~a;           /*-61 = 1100 0011 */
   printf("Line 4 - Value of c is %d\n", c );

   c = a << 2;      /* 240 = 1111 0000 */
   printf("Line 5 - Value of c is %d\n", c );

   c = a >> 2;      /* 15 = 0000 1111 */
   printf("Line 6 - Value of c is %d\n", c );
}
```

Assignment Operators 指定運算子

Operator	Description	Example
=	Simple assignment operator. Assigns values from right side operands to left side operand	C = A + B will assign the value of A + B to C
+=	Add AND assignment operator. It adds the right operand to the left operand and assign the result to the left operand.	C += A is equivalent to C = C + A
-=	Subtract AND assignment operator. It subtracts the right operand from the left operand and assigns the result to the left operand.	C -= A is equivalent to C = C - A
*=	Multiply AND assignment operator. It multiplies the right operand with the left operand and assigns the result to the left operand.	C *= A is equivalent to C = C * A
/=	Divide AND assignment operator. It divides the left operand with the right operand and assigns the result to the left operand.	C /= A is equivalent to C = C / A
%=	Modulus AND assignment operator. It takes modulus using two operands and assigns the result to the left operand.	C %= A is equivalent to C = C % A

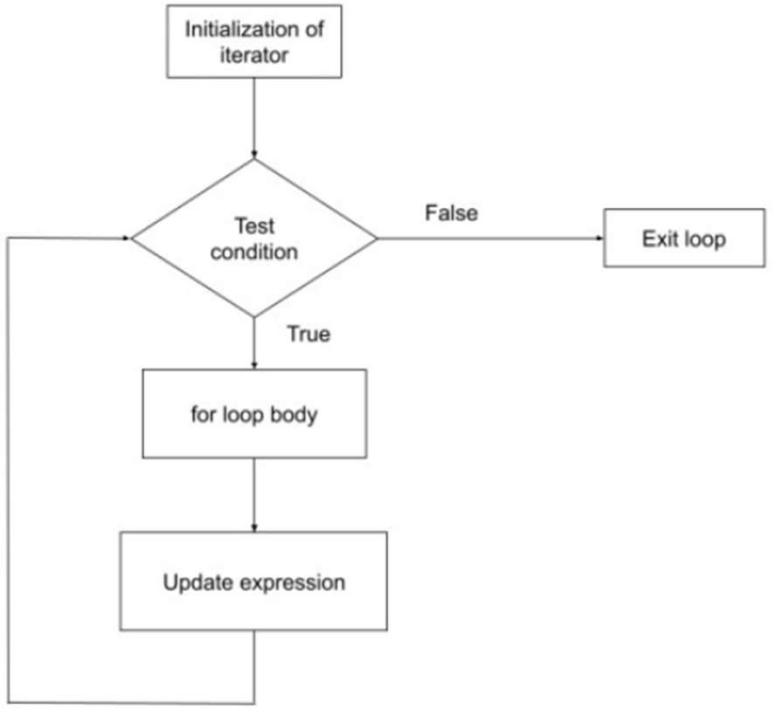
Assignment Operators 指定運算子

```c
#include <stdio.h>

main() {

   int a = 21;
   int c ;

   c =  a;
   printf("Line 1 - =  Operator Example, Value of c = %d\n", c );

   c +=  a;
   printf("Line 2 - += Operator Example, Value of c = %d\n", c );

   c -=  a;
   printf("Line 3 - -= Operator Example, Value of c = %d\n", c );

   c *=  a;
   printf("Line 4 - *= Operator Example, Value of c = %d\n", c );

   c /=  a;
   printf("Line 5 - /= Operator Example, Value of c = %d\n", c );

   c  = 200;
   c %=  a;
   printf("Line 6 - %= Operator Example, Value of c = %d\n", c );

   c <<=  2;
   printf("Line 7 - <<= Operator Example, Value of c = %d\n", c );

   c >>=  2;
   printf("Line 8 - >>= Operator Example, Value of c = %d\n", c );

   c &=  2;
   printf("Line 9 - &= Operator Example, Value of c = %d\n", c );

   c ^=  2;
   printf("Line 10 - ^= Operator Example, Value of c = %d\n", c );

   c |=  2;
   printf("Line 11 - |= Operator Example, Value of c = %d\n", c );
}
```

迴圈 Loop(for，while，do while)

・for 語法

```c
#include<stdio.h>
int main()
{
    int number;
    for(number=1;number<=10;number++)   //for loop to print 1-10 numbers
    {
        printf("%d\n",number);      //to print the number
    }
    return 0;
}
```

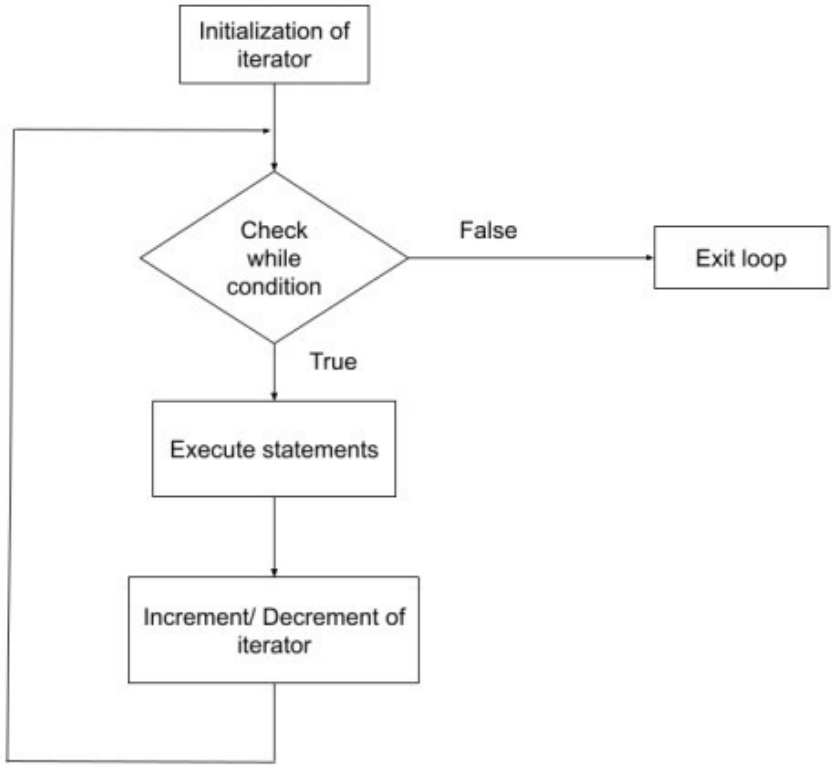


Reference: https://www.studytonight.com/c/loops-in-c.php

迴圈 Loop(for，while，do while)

- while 語法

```c
#include<stdio.h>

void main( )
{
    int x;
    x = 1;
    while(x <= 10)
    {
        printf("%d\t", x);
        /* below statement means,
        do x = x+1, increment x by 1 */
        x++;
    }
}
```

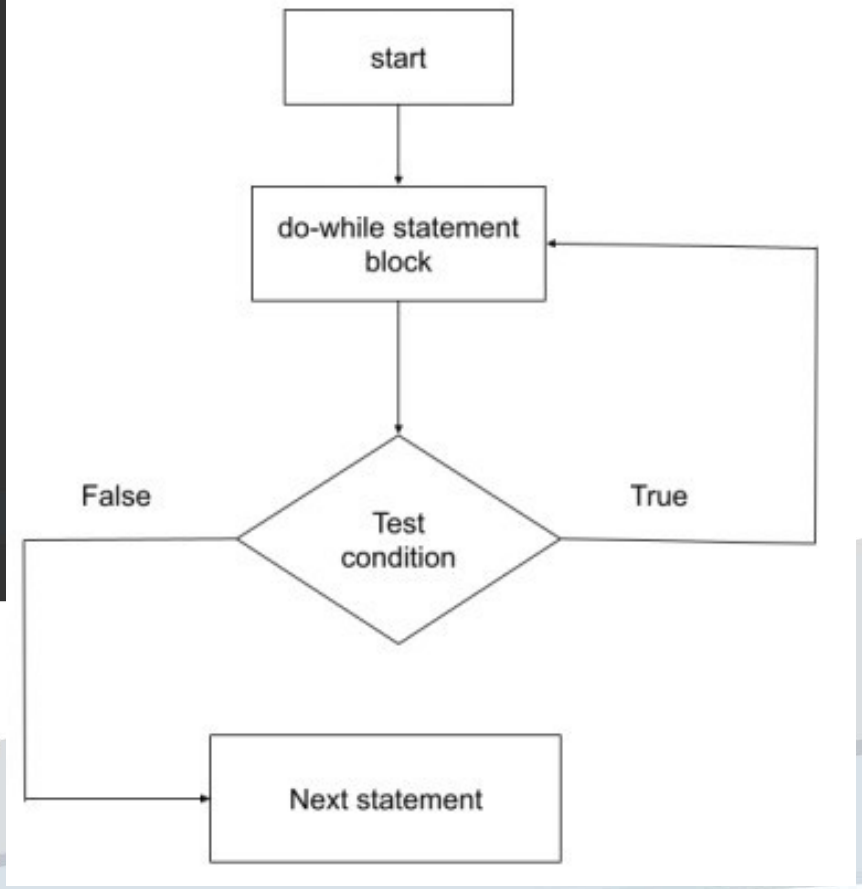


Reference: https://www.studytonight.com/c/loops-in-c.php

迴圈 Loop(for，while，do while)

・do while 語法

```c
#include<stdio.h>

void main()
{
    int a, i;
    a = 5;
    i = 1;
    do
    {
        printf("%d\t", a*i);
        i++;
    }
    while(i <= 10);
}
```



Reference: https://www.studytonight.com/c/loops-in-c.php

迴圈 Loop 練習題1

Use nested loops to produce the following pattern:

```
A
BC
DEF
GHIJ
KLMNO
PQRSTU
```

迴圈 Loop 練習題2

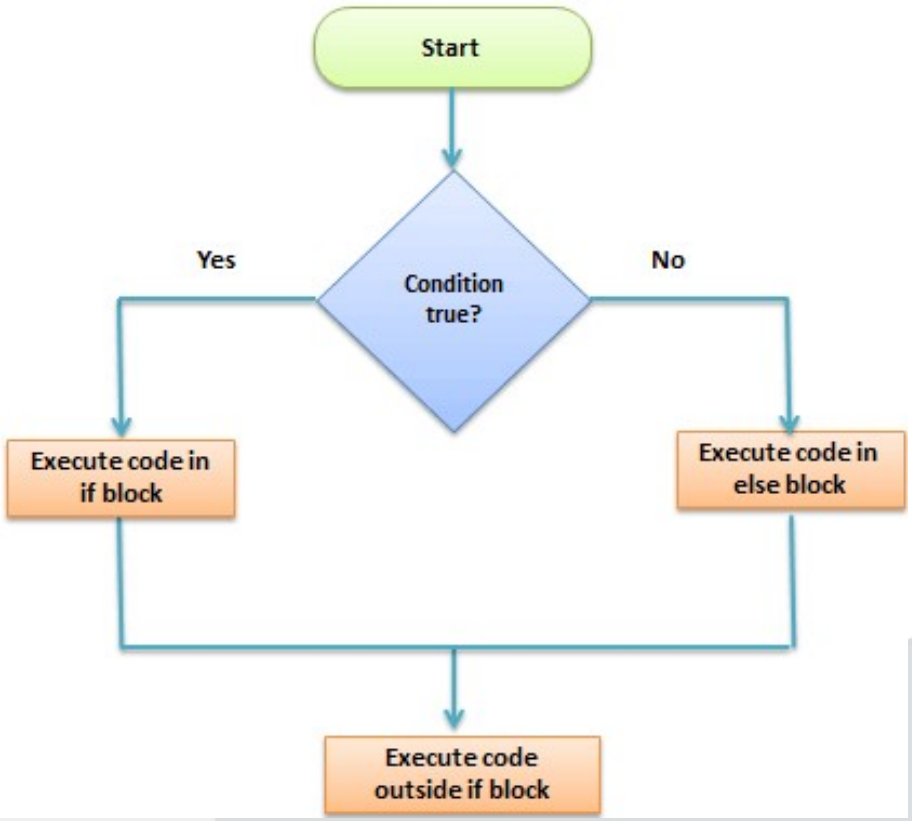
Write a program that prints a table with each line giving an integer, its square, and its cube. Ask the user to input the lower and upper limits for the table. Use a `for` loop.

預計程式執行結果:

```
Enter two number: the min and the max number
The 1 number is ..(After inputing number , then press Enter..
10
The 2 number is ..(After inputing number , then press Enter..
15
-----------------------------------------------
n       n*n     n*n*n
10      100     1000
11      121     1331
12      144     1728
13      169     2197
14      196     2744
15      225     3375
-----------------------------------------------
```

If …..else…

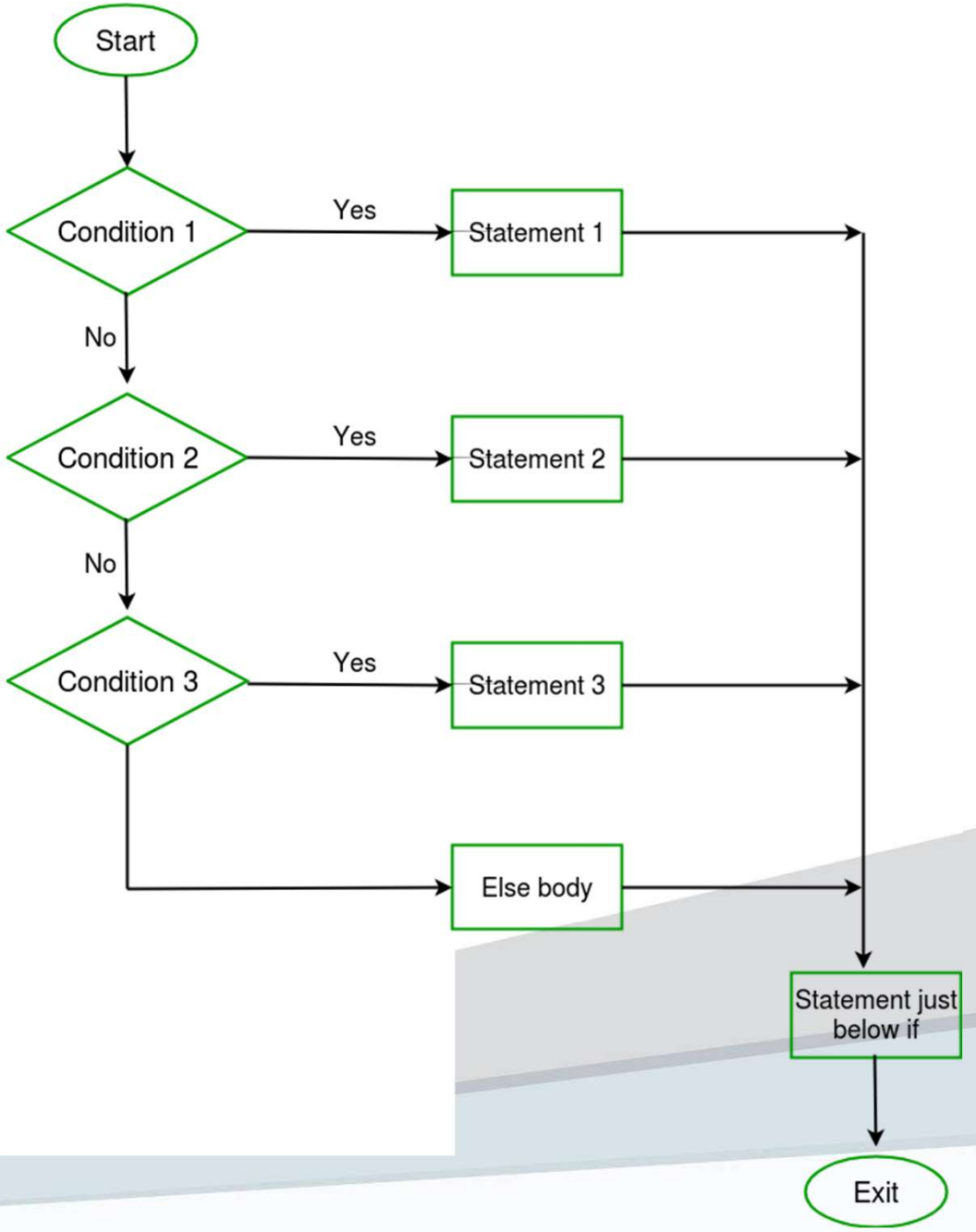
```c
#include <stdio.h>

int main () {

   /* local variable definition */
   int a = 100;

   /* check the boolean condition */
   if( a < 20 ) {
      /* if condition is true then print the following */
      printf("a is less than 20\n" );
   } else {
      /* if condition is false then print the following */
      printf("a is not less than 20\n" );
   }

   printf("value of a is : %d\n", a);

   return 0;
}
```



Reference:https://www.guru99.com/c-if-else-statement.html

If ….elseif …else…

```c
// C program to illustrate nested-if statement

#include <stdio.h>

int main()
{
    int i = 20;

    // Check if i is 10
    if (i == 10)
        printf("i is 10");

    // Since i is not 10
    // Check if i is 15
    else if (i == 15)
        printf("i is 15");

    // Since i is not 15
    // Check if i is 20
    else if (i == 20)
        printf("i is 20");

    // If none of the above conditions is true
    // Then execute the else statement
    else
        printf("i is not present");

    return 0;
}
```



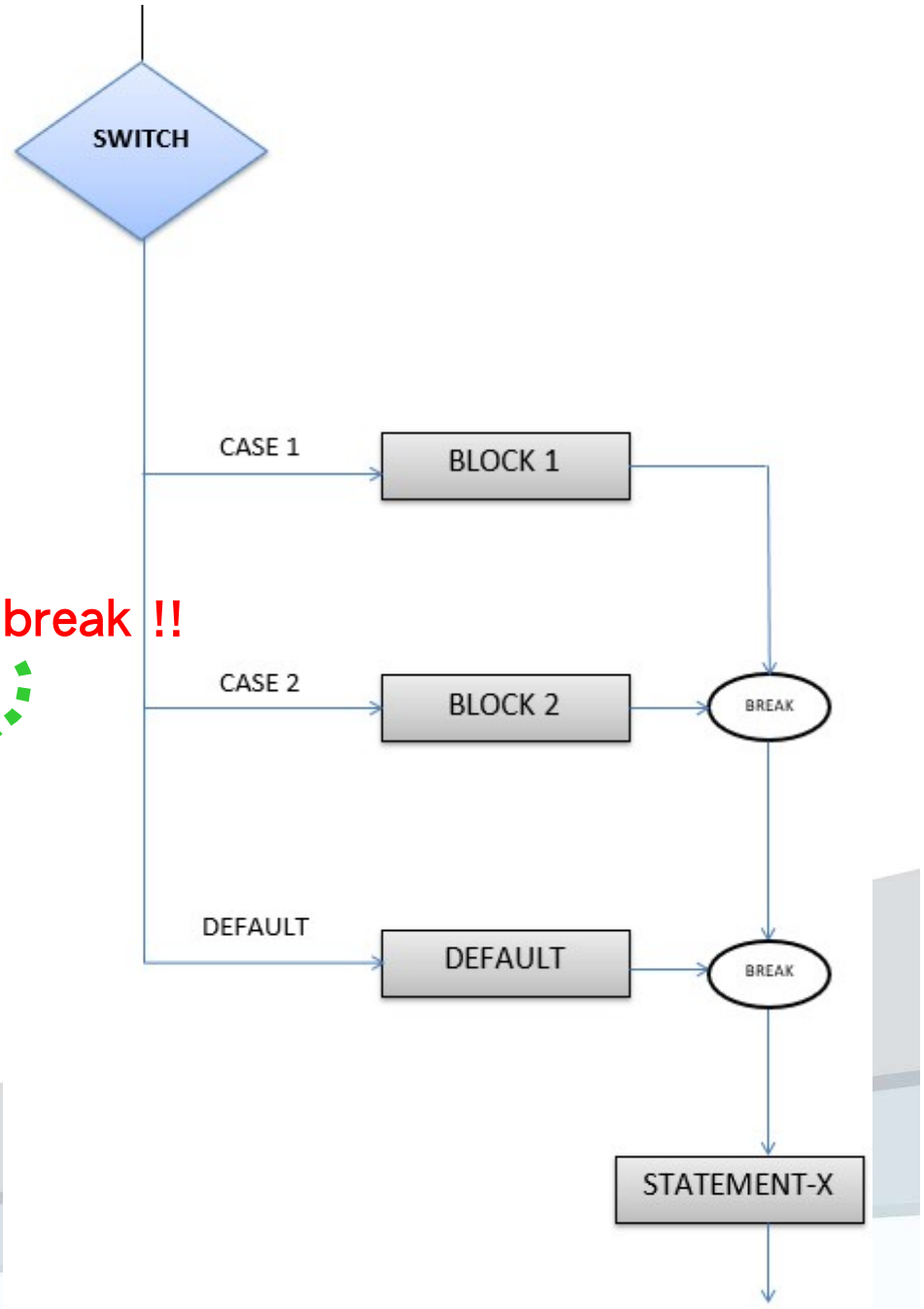
Reference:https://www.geeksforgeeks.org/c-c-if-else-if-ladder-with-examples/

switch

- *可以用 if ….else if…. else 取代*
- 記得加 break。

```c
#include <stdio.h>
    int main() {
        int num = 8;
        switch (num) {
            case 7:
                printf("Value is 7");
                break;
            case 8:
                printf("Value is 8");
                break;
            case 9:
                printf("Value is 9");
                break;
            default:
                printf("Out of range");
                break;
        }
        return 0;
    }
```

記得加 break !!



Ref: https://www.guru99.com/c-switch-case-statement.html

流程控制練習題1

Write a program that reads input until encountering the # character and then reports the number of spaces read, the number of newline characters read, and the number of all other characters read.

預計程式執行結果:

```
 Enter some words, and press '#' to exit
James James#
all word have 1 spaces, 0 newline, and the other chars =10


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

流程控制練習題2

Write a program that requests the hours worked in a week and then prints the gross pay, the taxes, and the net pay. Assume the following:

a. Basic pay rate = $10.00/hr

b. Overtime (in excess of 40 hours) = time and a half

c. Tax rate: #15% of the first $300

20% of the next $150

25% of the rest

Use `#define` constants, and don't worry if the example does not conform to current tax law.

預計程式執行結果:

```
Enter the hours perweek..
50
the salary= 650.0, and the tax= 125.0

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