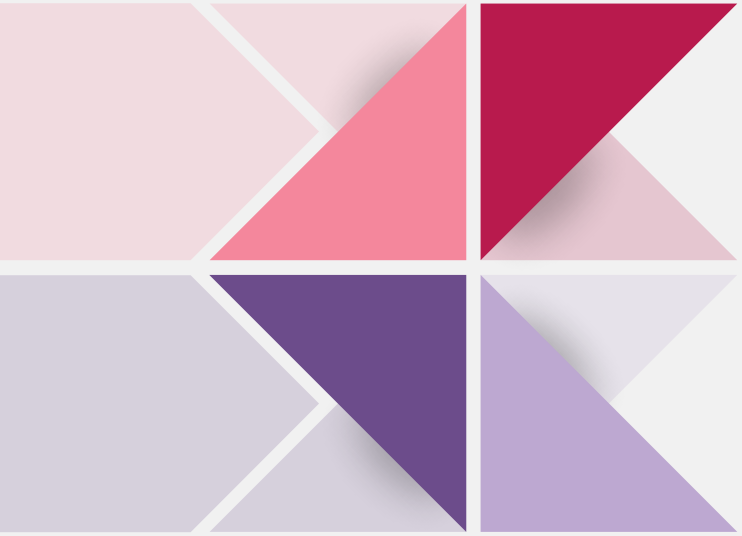
Introduction of C

Index



01
History

History

Computer Science Father

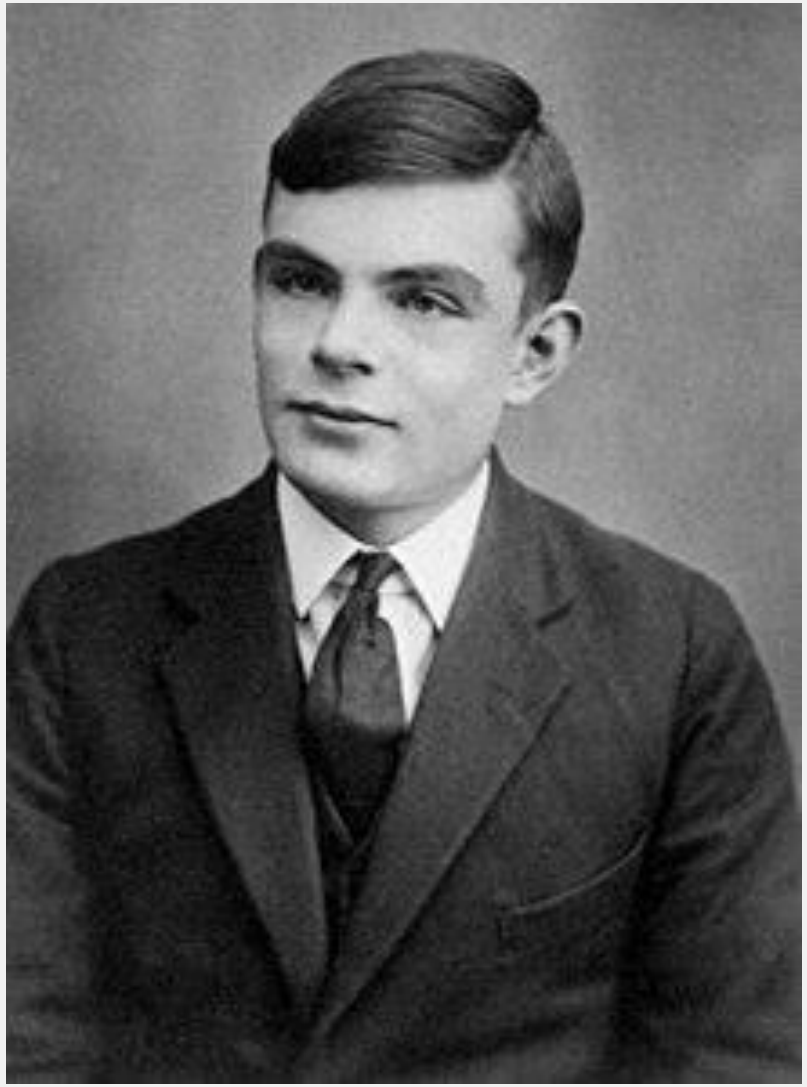
Alan Mathison Turing

艾倫·麥席森·圖靈

1912.06 - 1954.06 (41歲)

研究領域: 數學、密碼分析、邏輯學及電腦科學 、數理生物學

運動領域: 是一位世界級的長跑運動員，馬拉松最好成績是2小時46分03秒(手動計時)，比1948年奧林匹克運動會金牌成績慢11分鐘。1948年的一次越野賽跑中，他跑贏了同年奧運會銀牌得主湯姆·理查茲。

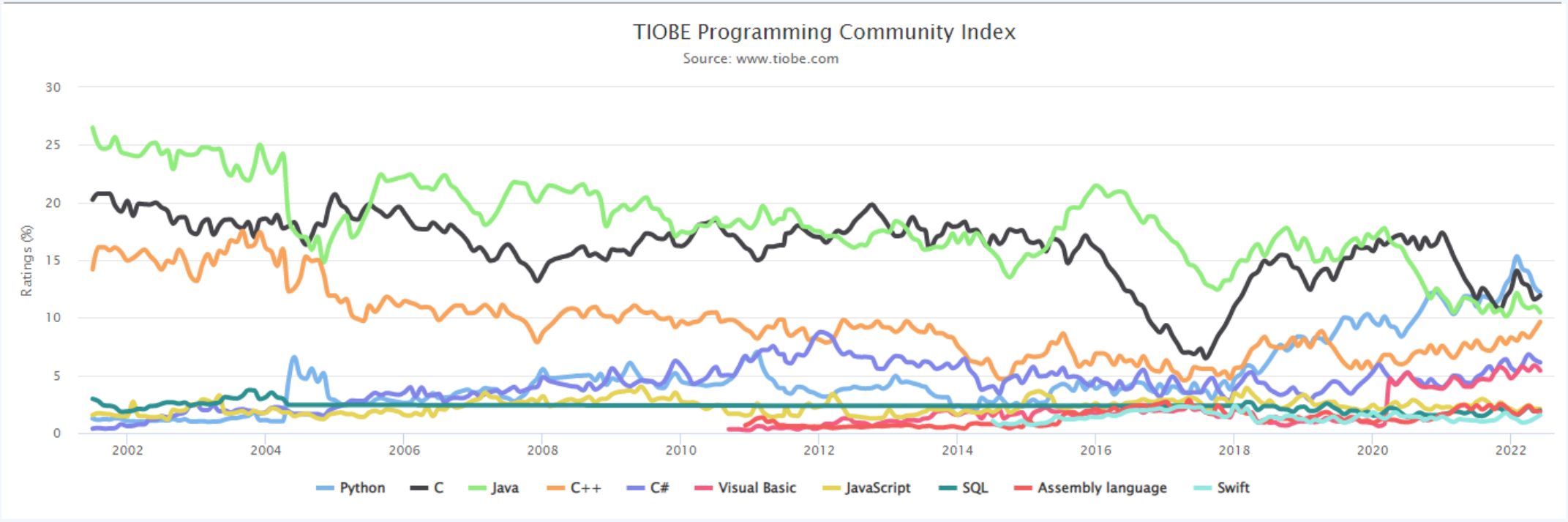


Link

History

Why C?

Jun 2022	Jun 2021	Change	Programming Language	Ratings	Change
1	2	▲	Python	12.20%	+0.35%
2	1	▼	C	11.91%	-0.64%
3	3		Java	10.47%	-1.07%
4	4		C++	9.63%	+2.26%

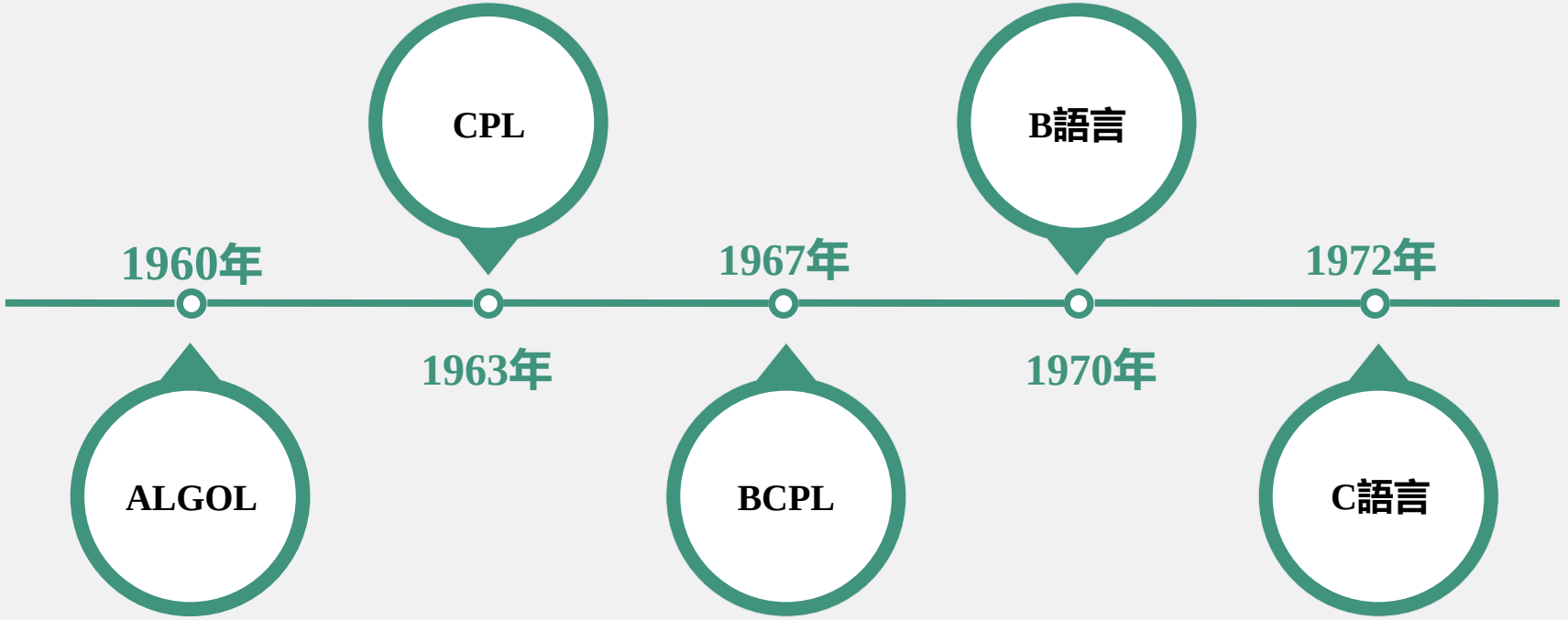


[Link]

History

Inspiration

C是美國貝爾實驗室的Dennis Ritchie以B語言為基礎所開發的，現今Windows與UNIX系列的作業系統內大部分也是用C語言設計的



History

Inspiration

Martin Richards

Ken Thompson Dennis Ritchie

DEC PDP-7

Link

History

Inspiration

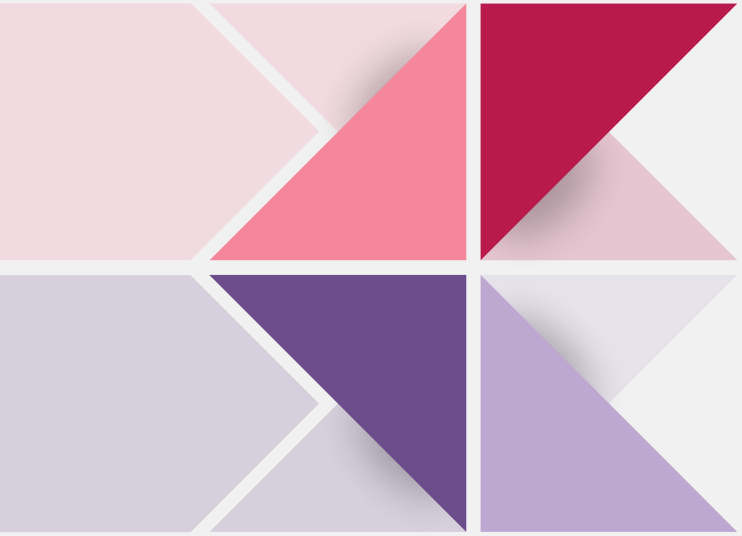
優點

1. 簡潔方便 - 只有32個關鍵字，9個控制語句
2. 允許直接存取記憶體位址，控制硬體
3. 可移植性高 - 不同機器上的C編譯程式有86%是公用程式碼，編譯程式方便移植

…

缺點

語法限制較不嚴格，使得程式安全性較低，因此從應用的角度來看，C語言使用者對於程式設計的熟練度要高一點



02
Compile and Execution

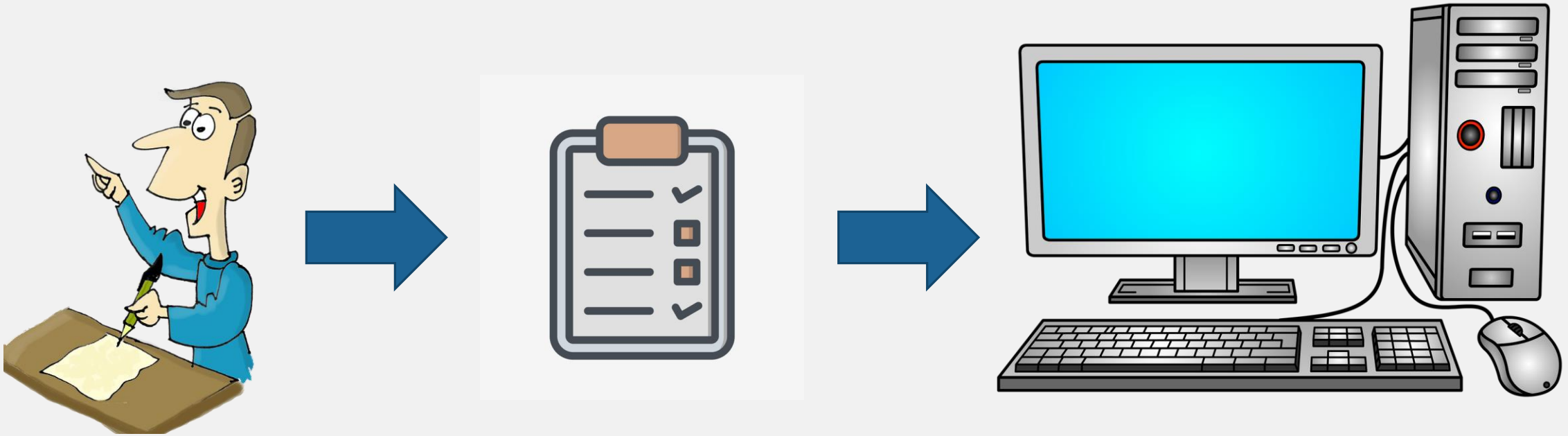
Compile and Execution

Process

使用電腦完成想要的目的

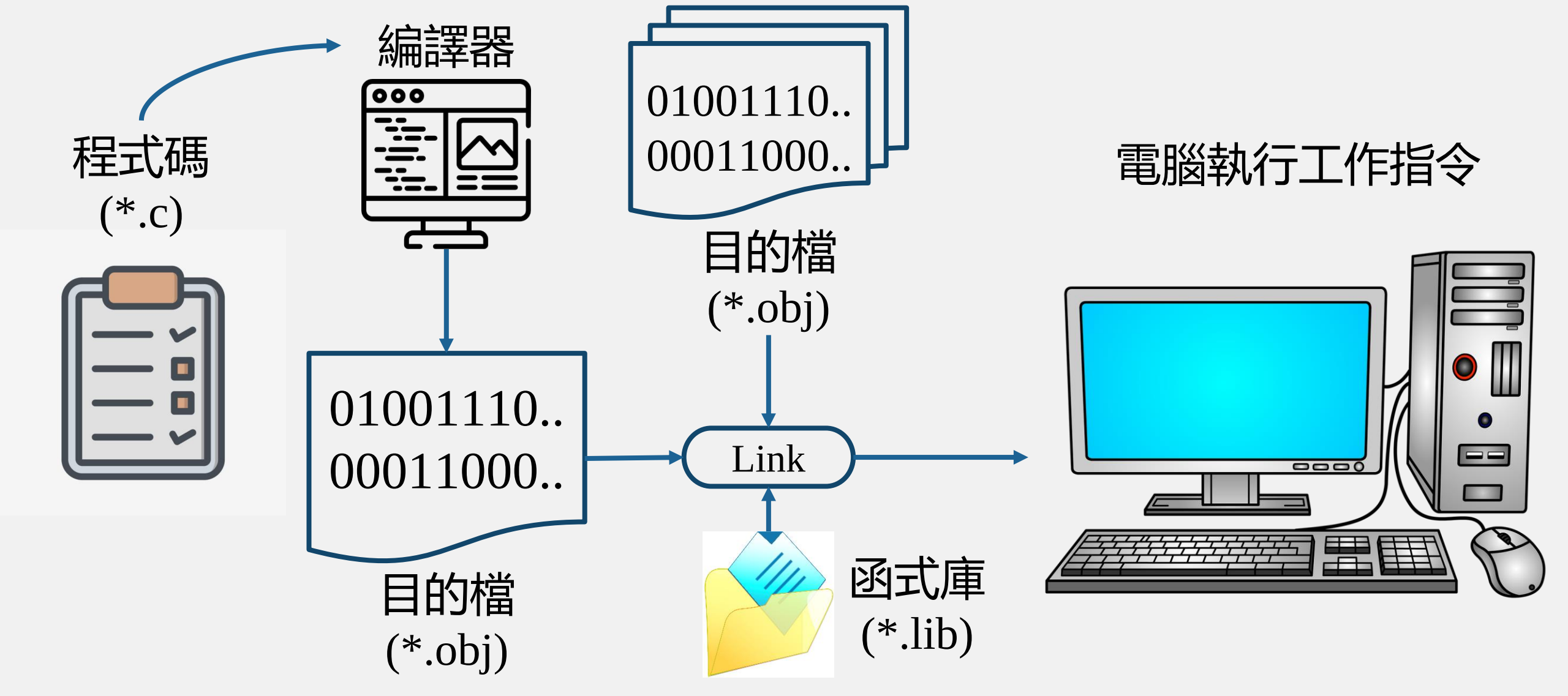
程式決定指令執行內容 (程式碼)

電腦執行工作指令



Compile and Execution

Process



Compile and Execution

Environment

[Programiz]

Programiz
C Online Compiler

main.c

Run

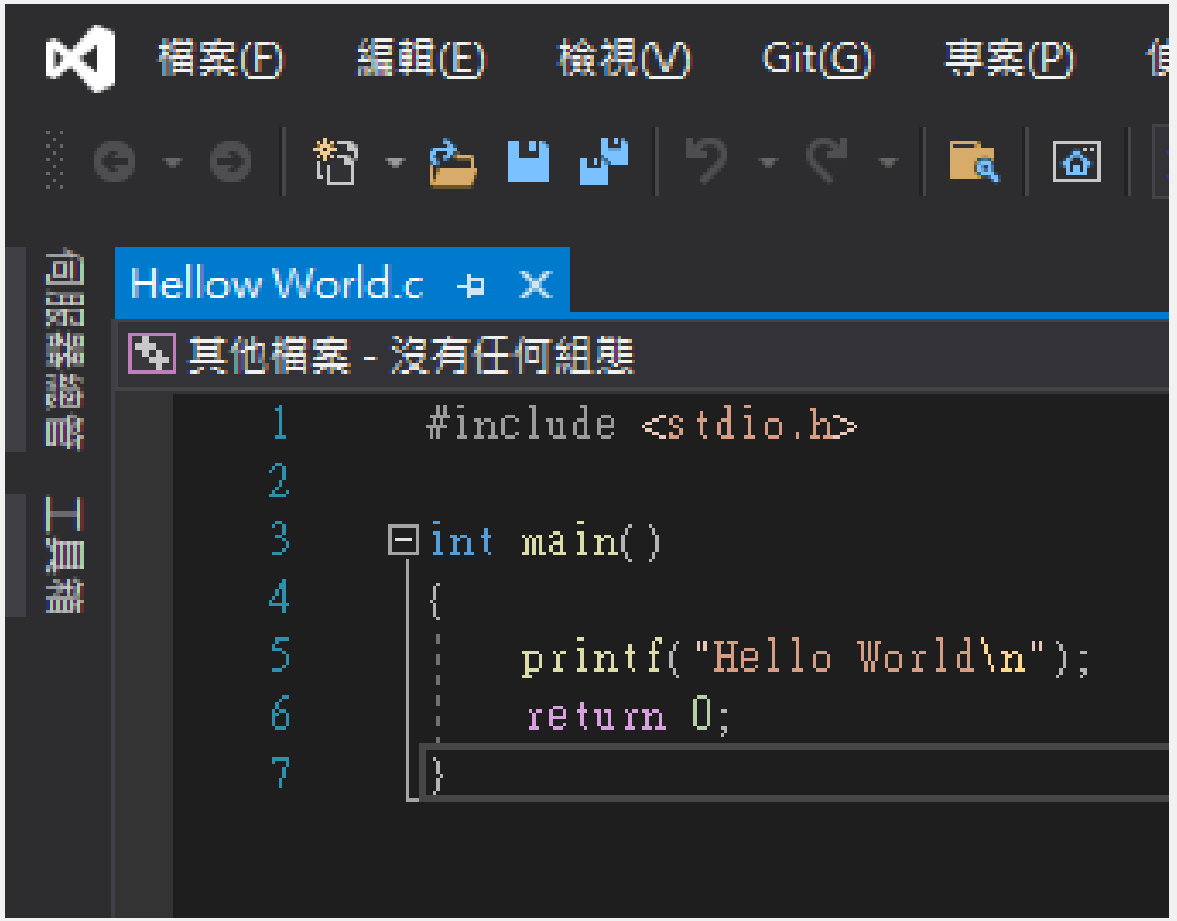
```c
// Online C compiler to run C program online
#include <stdio.h>

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

    return 0;
}
```

Output

Clear

```
/tmp/o9Eqf17PxZ.o
Hello world
```

[Visual Studio]



Compile and Execution

Environment - Visual Studio

優點

整合了編譯器與介面

缺點

啟動慢，檔案開啟速度慢

下載

Compile and Execution

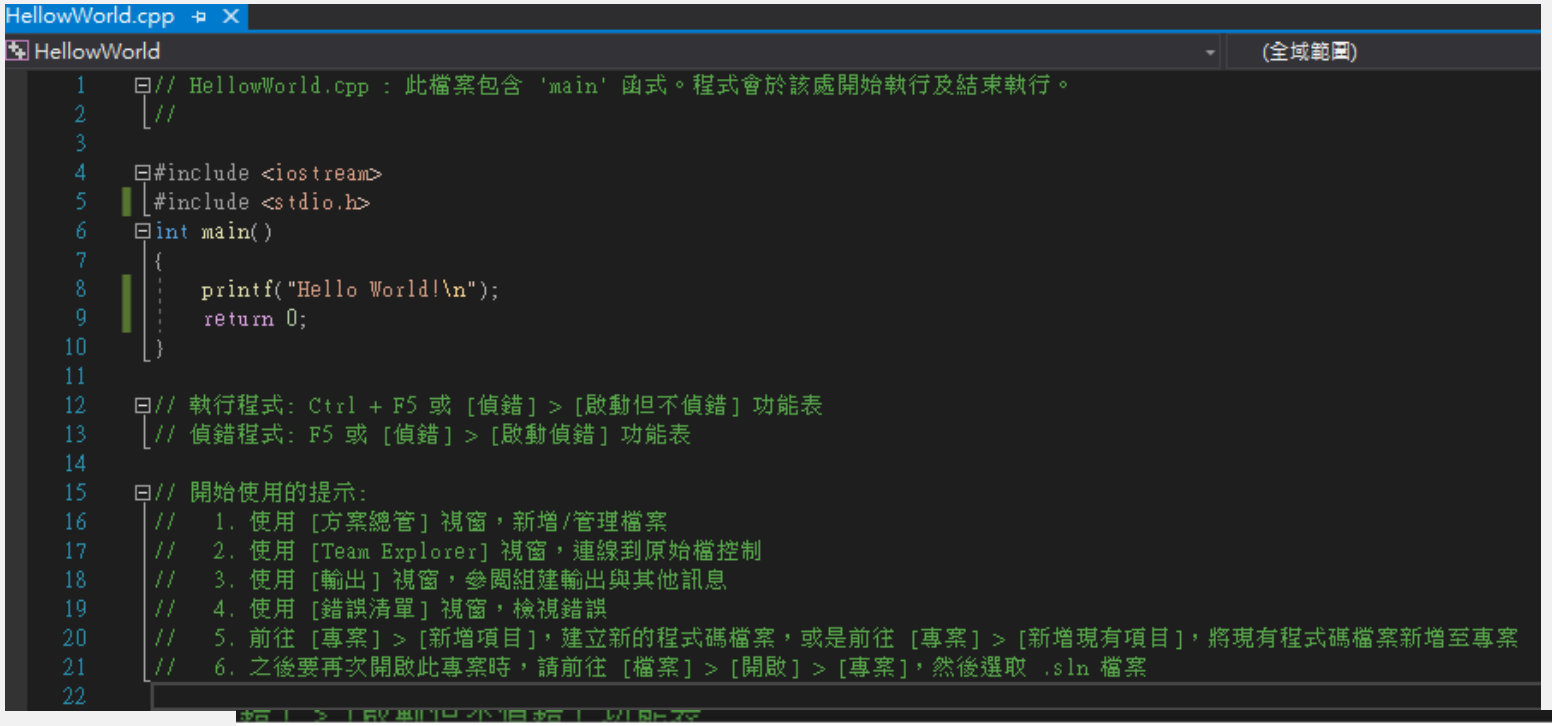
Environment - Visual Studio

Compile and Execution

Environment - Visual Studio

Compile and Execution

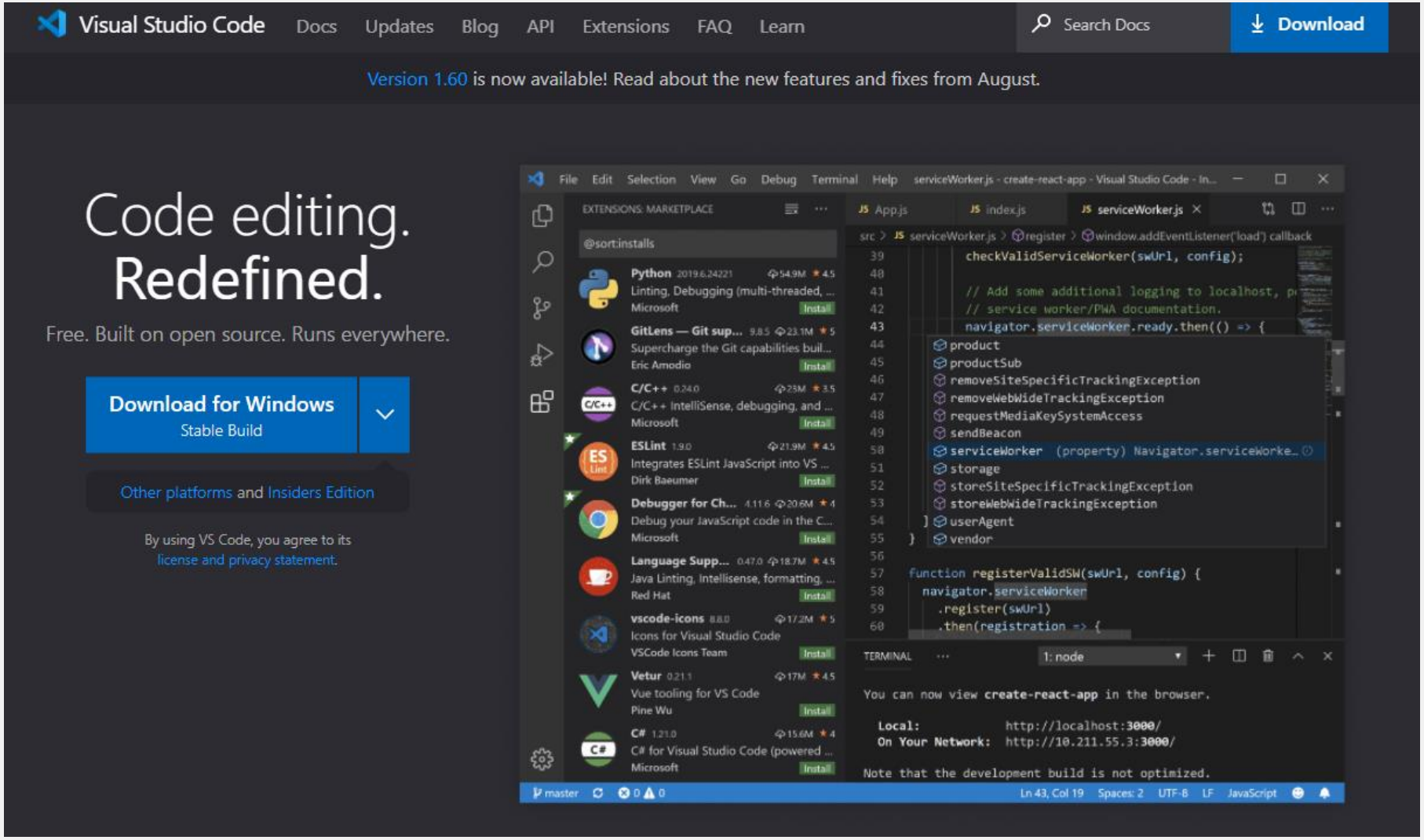
Environment - Visual Studio



Compile and Execution

Environment - Visual Studio Code (VS Code)

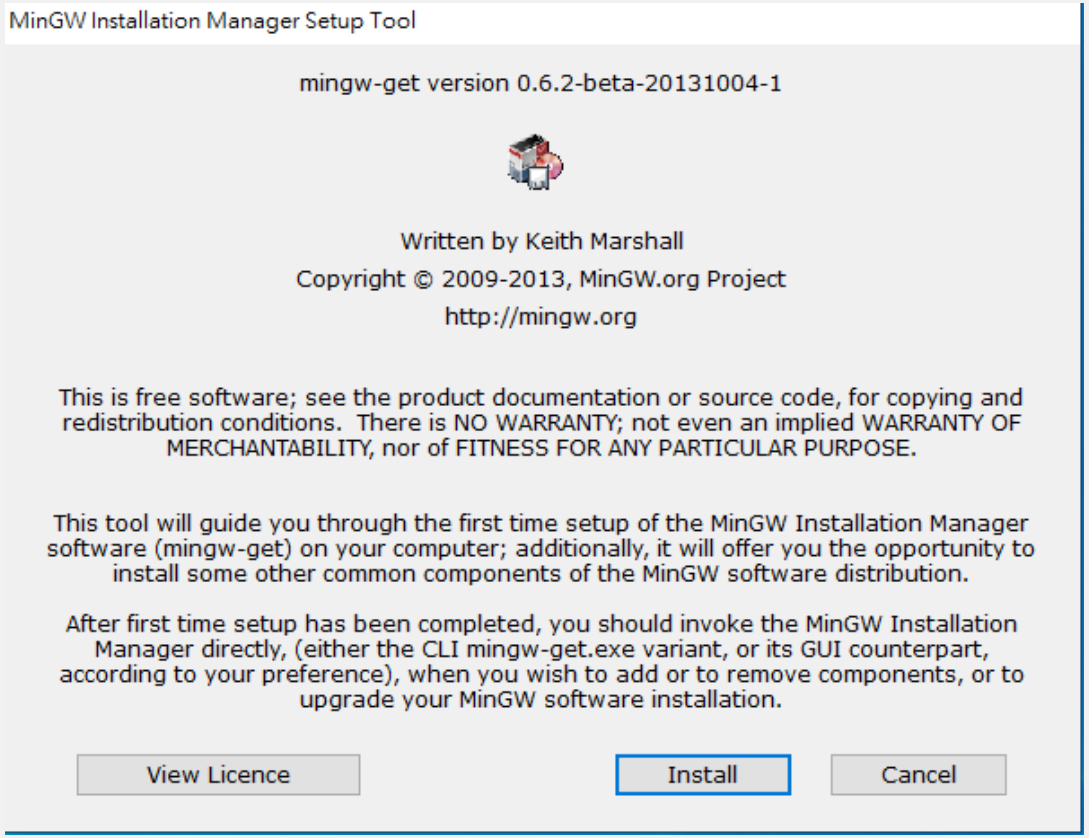
[Visual Studio Code]

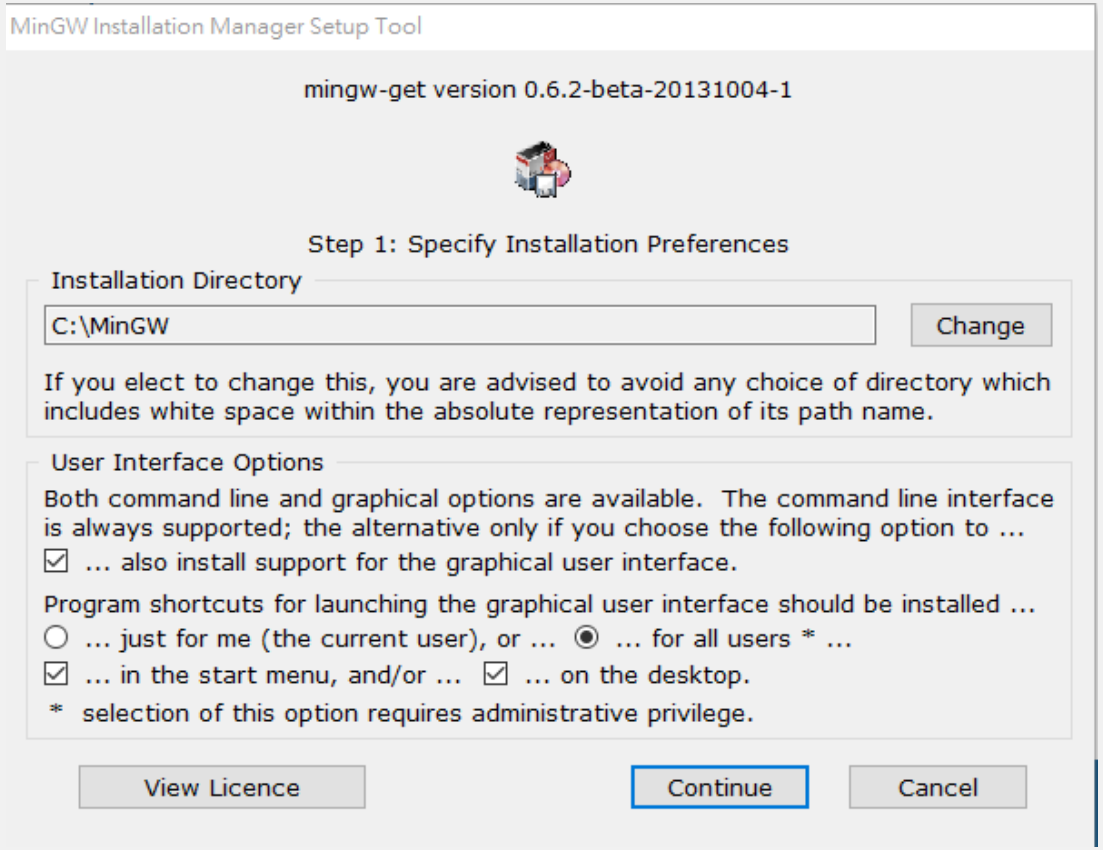


Compile and Execution

Environment - Compile (Mingw)

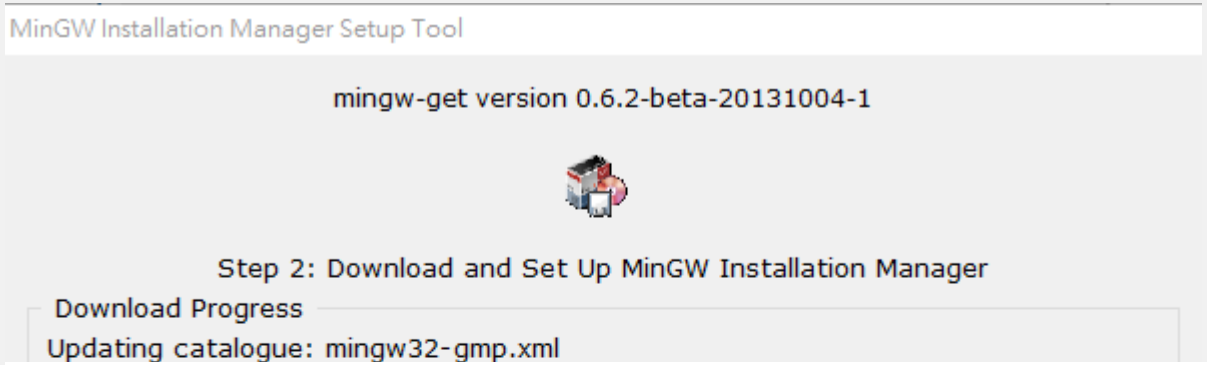
Link

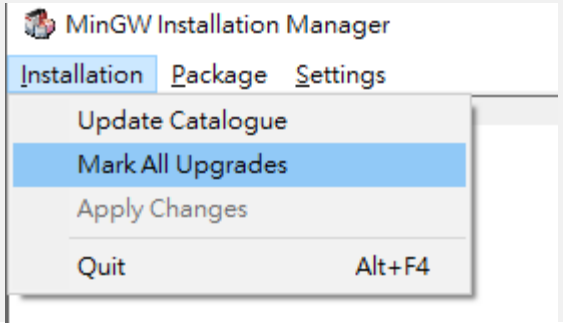


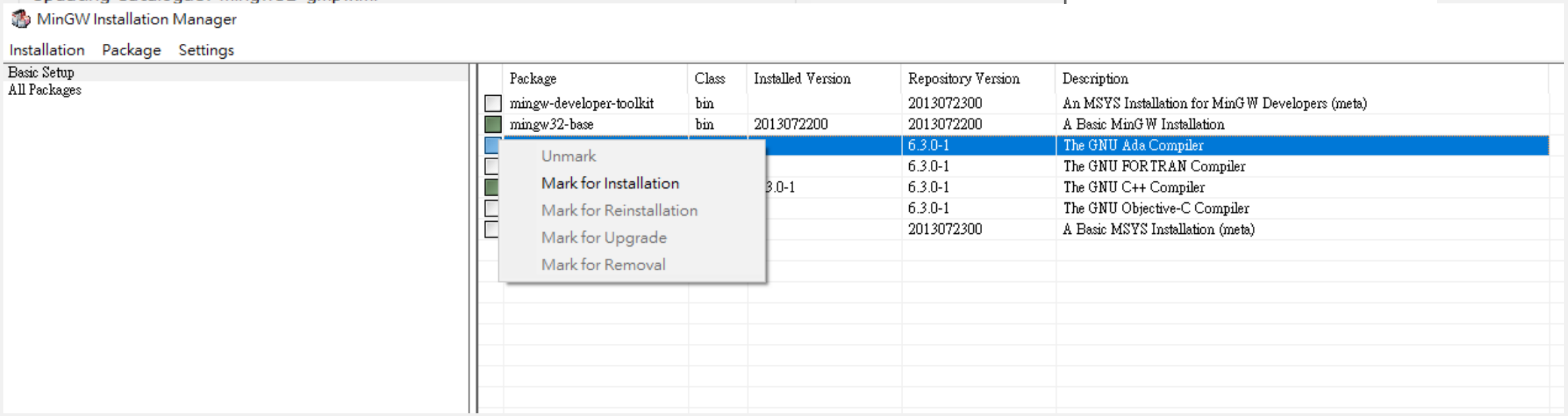


Compile and Execution

Environment - Compile (Mingw)

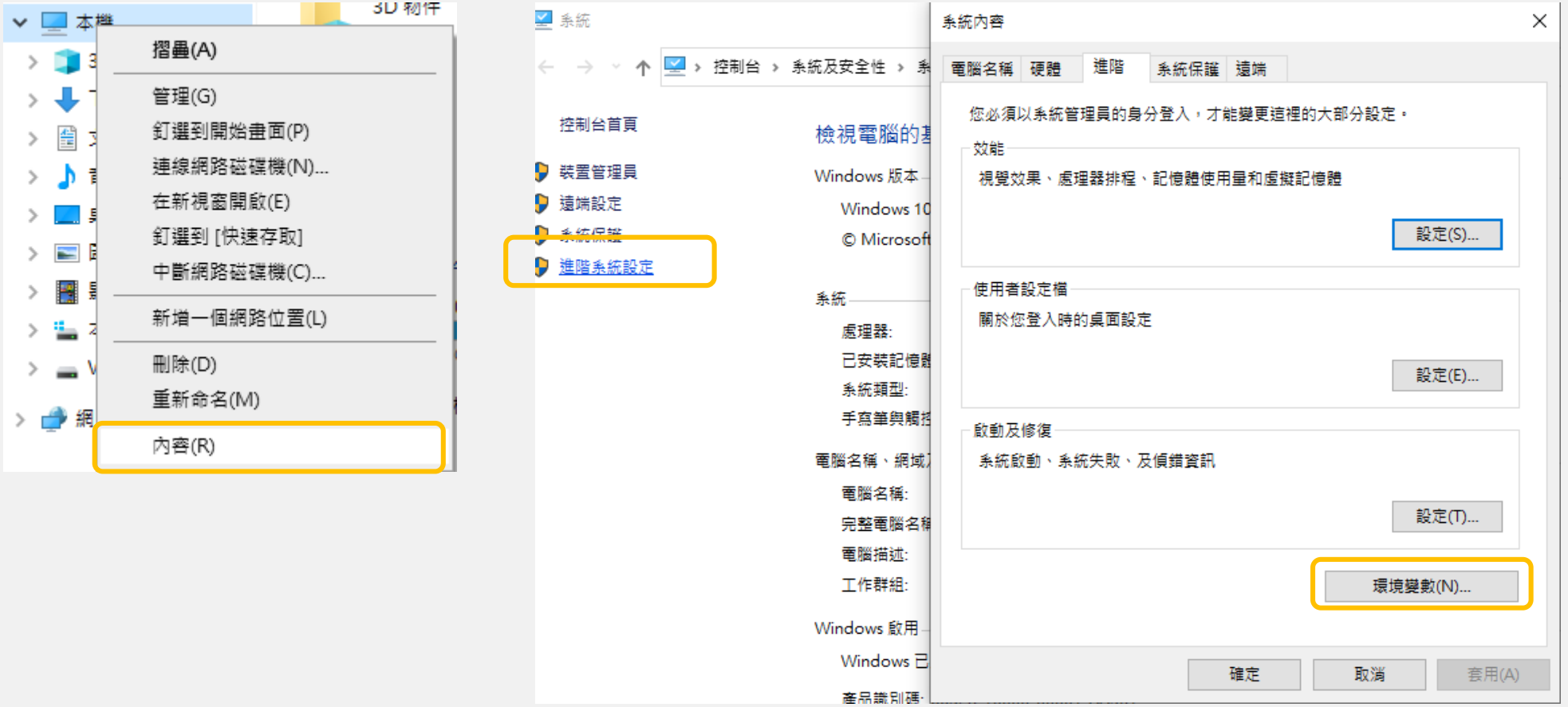






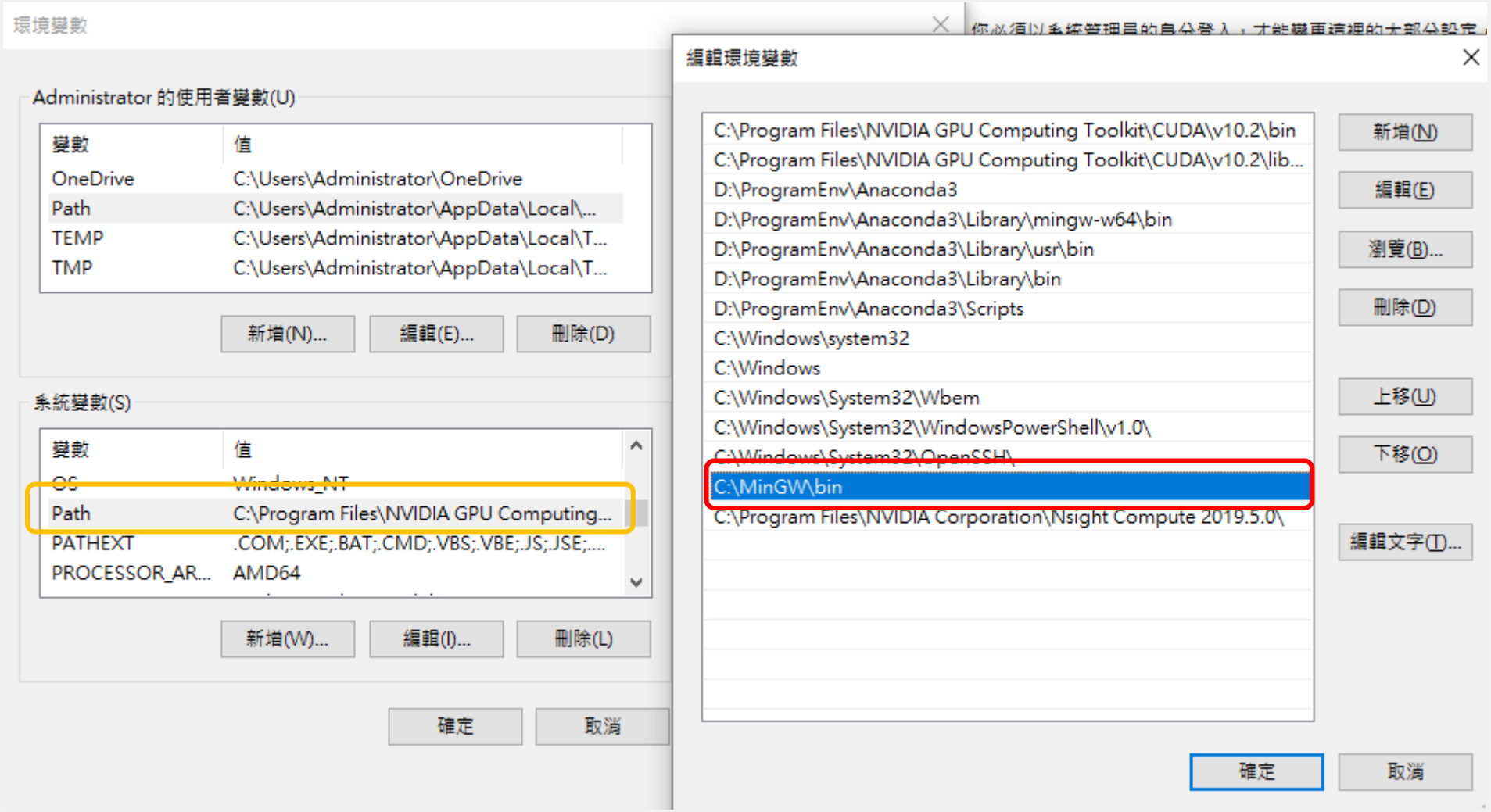
Compile and Execution

Environment - Compile (Mingw)



Compile and Execution

Environment - Compile (Mingw)



Compile and Execution

Environment - Compile (Mingw)

Compile and Execution

Environment - Execution

編譯

gcc "filename.c" -o "execute_filename"

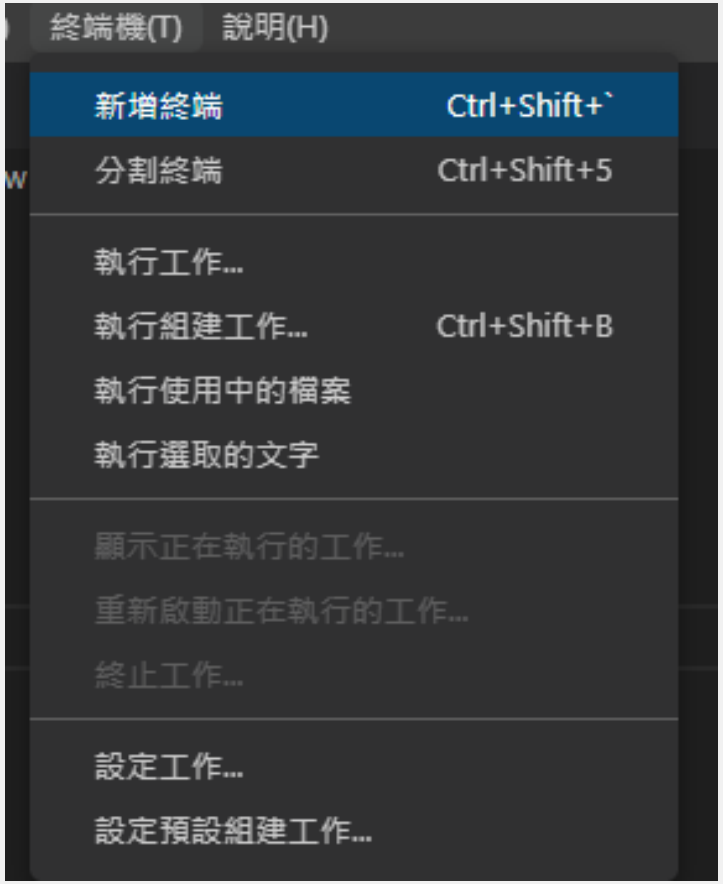
執行

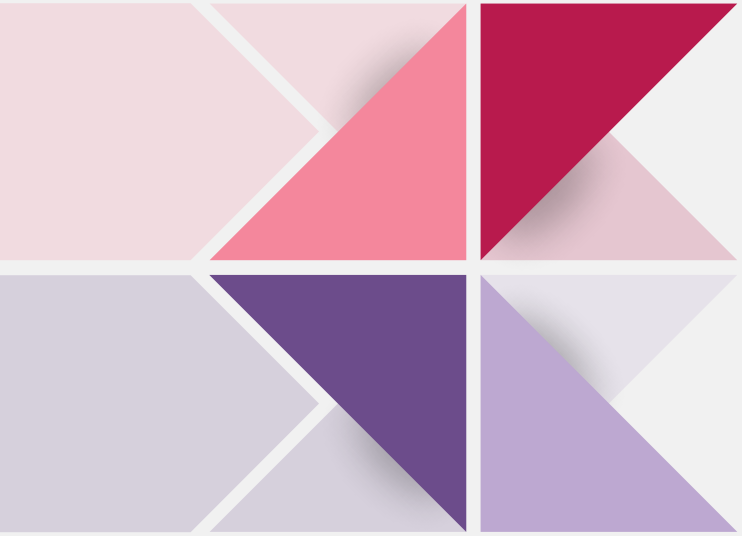
execute_filename (.exe)

```
C:\Users\Administrator\Desktop\temp>hello
hello
C:\Users\Administrator\Desktop\temp>hello.exe
hello
```

Compile and Execution

Environment - Execution





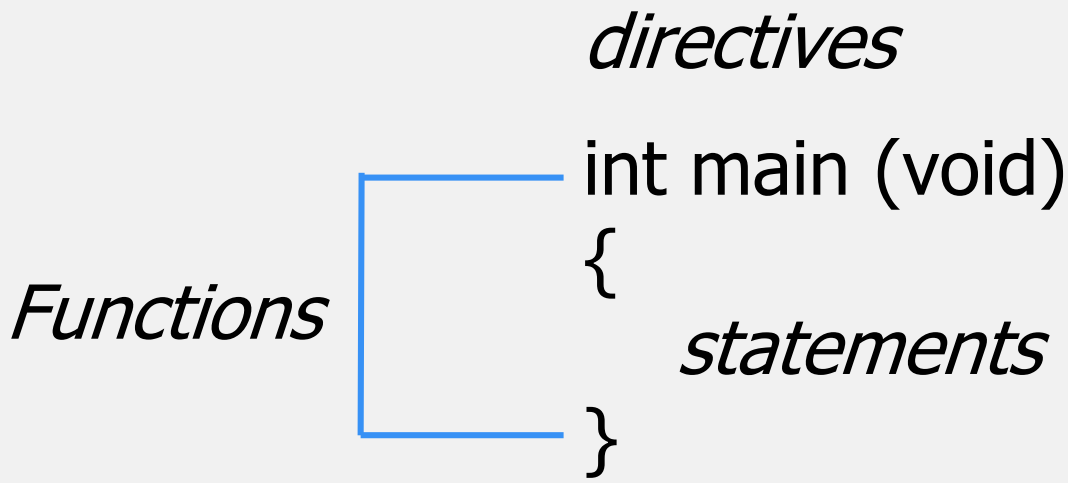
03
Let's go C

Let' go C

Introduction

A simple C program relies on three key language features:

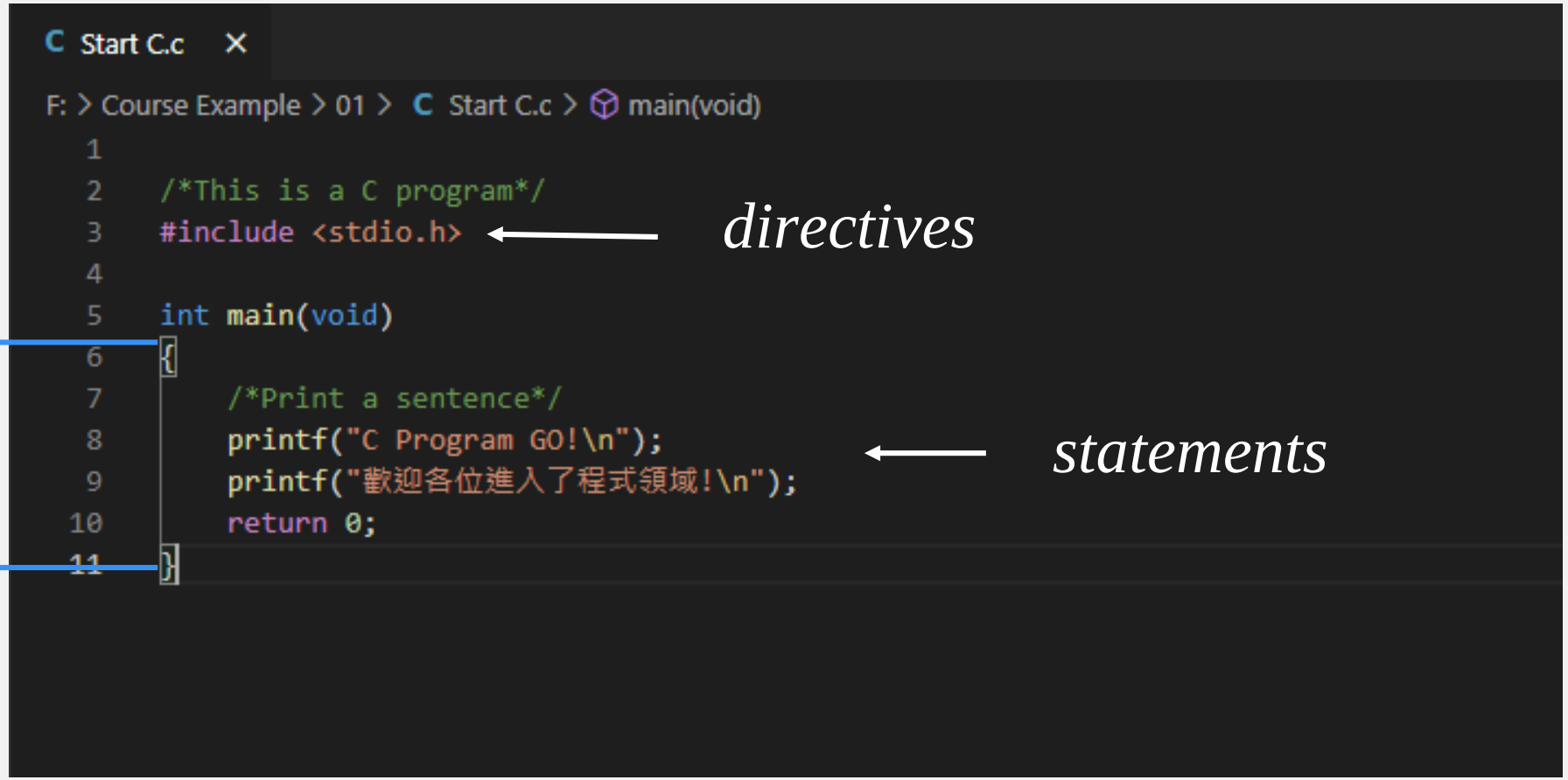
- Directives
- Functions
- Statements



Let' go C

Introduction

An example: File.c

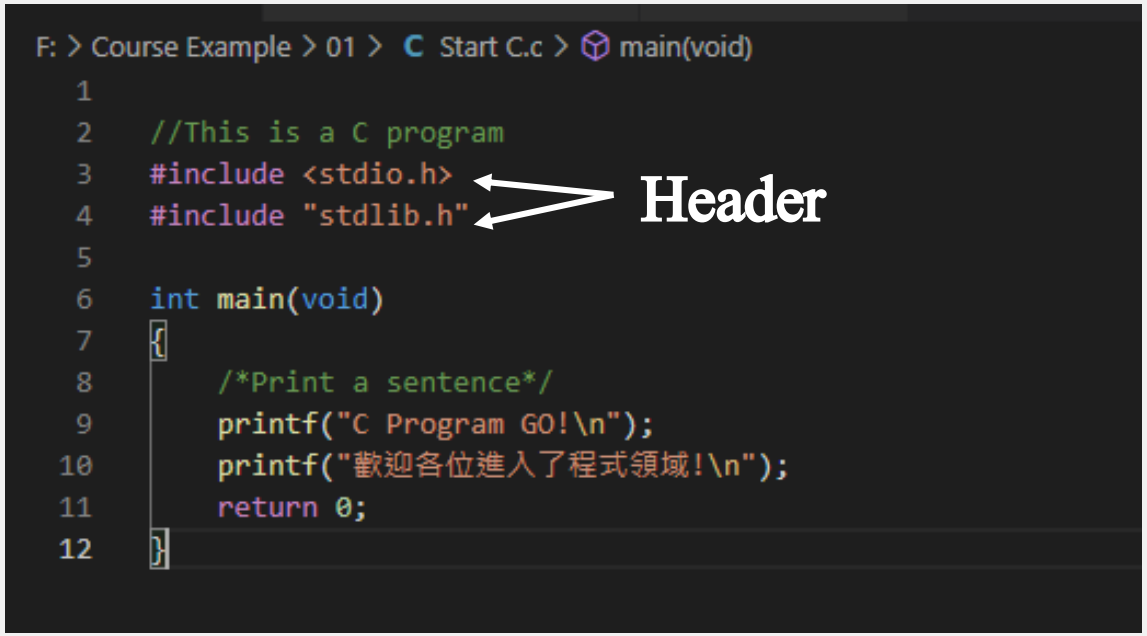


Let' go C

Introduction

Directives

- Before a C program is compiled, it is first edited by a preprocessor
- Commands intended for the preprocessor are called directives
- Example:
  - #include <file>
    - ✓ Search from environment path
  - #include "file"
    - ✓ Search from current path

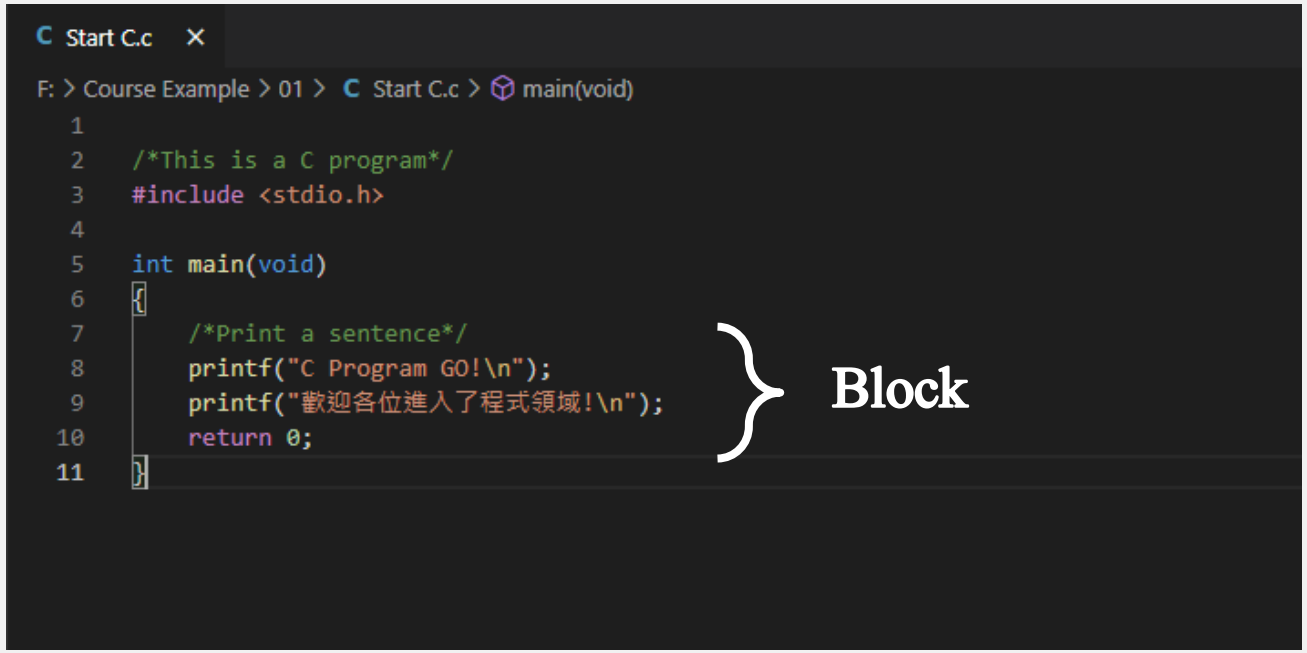


Let' go C

Introduction

main() function

- The main function is mandatory and gets called automatically while the program is executed
  - ✓ main() + curly brackets
  - ✓ Block is the statement in the curly brackets



Let' go C

Introduction

Statement

- It is a command to be executed
- Each statement ends with a semicolon
  - ✓ Exception: compound statement

Let' go C

Introduction - keywords & Identifiers

keywords

- They are predefined and reserved words for programming with special meanings to the compiler
- They are parts of the syntax and cannot be as the identifiers

```
int money;
```

C keywords			
auto	break	case	char
continues	do	default	const
double	else	enum	extern
for	if	goto	float
int	long	register	return
signed	static	sizeof	short
struct	switch	typedef	union
void	while	volatile	unsigned

Let' go C

Introduction - keywords & Identifiers

Identifier

- It refers to name for entity such as variable, function, structure …
- It must be unique for an entity with unique name to identify

```
int money;
```

Rules for naming a identifier

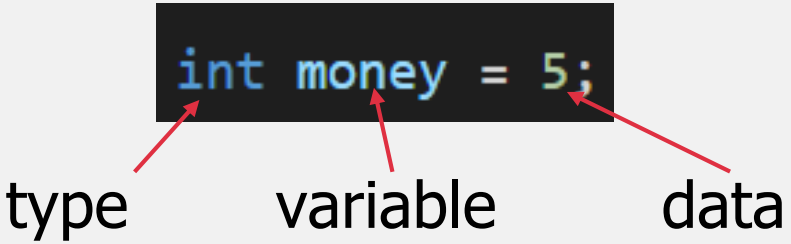
- ✓ A valid identifier have digits, letters, and underscores
- ✓ The first letter should be either a letter or an underscore
- ✓ The keywords cannot be used as identifiers
- ✓ The name of identifier should be meaningful (Not necessary)

Let' go C

Introduction - Variables & Constants

Variables

- Each variable should be given a unique name to indicate the storage area
- It is a symbolic representation of a memory location and is also a container to store data



Rules for naming a variable

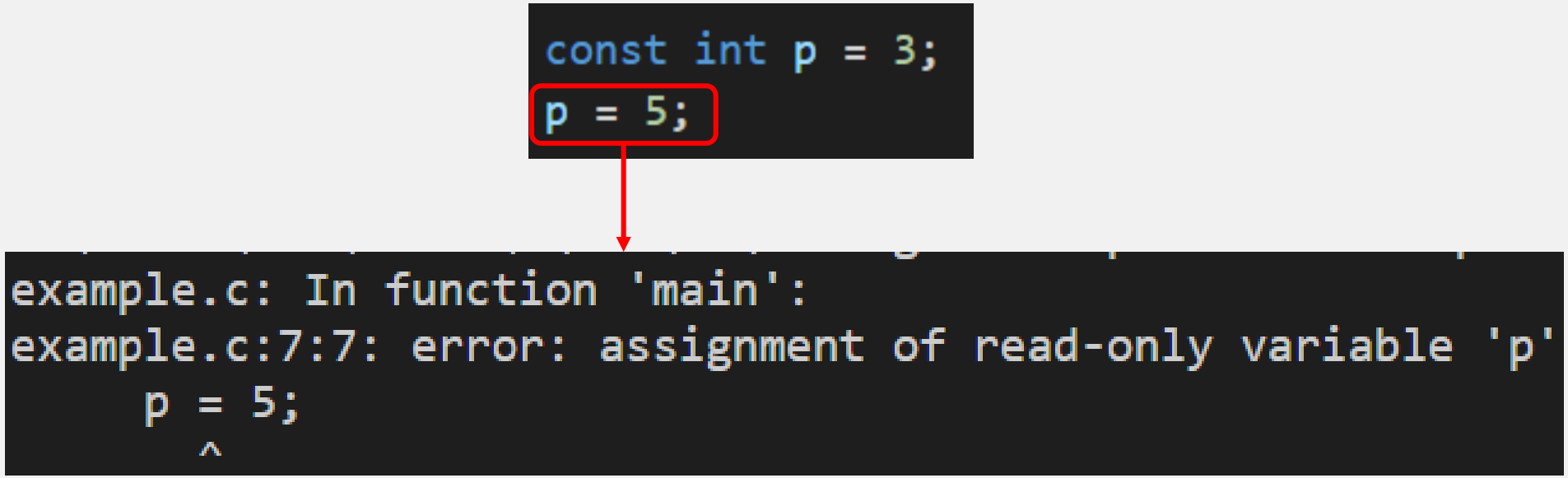
✓ Same as naming a identifier

Let' go C

Introduction - Variables & Constants

Constants

- It means a unchangeable variable is created using keyword "const"



Let' go C

Introduction - Data Type

Data Type

- It is the declaration for variable
- It determines the type and size of data associated with variable

Type	Size (bytes)
int	at least 2, usually 4
char	1
float	4
double	8
short int	usually 2
unsigned int	at least 2, usually 4
long int	at least 4, usually 8

Type	Size (bytes)
long long int	at least 8
unsigned long int	at least 4
unsigned long long int	at least 8
singed char	1
unsigned char	1
long double	at least 10, usually 12 or 16

Let' go C

Introduction - Input / Output (I/O)

Output - printf()

- The one of main output function is printf()
- The printf function must be supplied with a format string, followed by any values that are to be inserted into the string during printing

printf(*string*, *expr* 1, *expr* 2, ...);

- The format string may contain both ordinary characters and conversion specifications, which begin with the % character

```
int p = 3;
printf("The value is %d\n", p);
```

Let' go C

Introduction - Input / Output (I/O)

Output - printf()

- Conversion specifications

Type	Specifier
int	%d, %i
char	%c
float	%f
double	%lf
short int	%hd
unsigned int	%u
long int	%ld, %li

Type	Size (bytes)
long long int	%lld, %lli
unsigned long int	%lu
unsigned long long int	%llu
singed char	%c
unsigned char	%c
long double	%Lf

Let' go C

Introduction - Input / Output (I/O)

Output - printf()

- A conversion specification can have the form %m.pX or %-m.pX, where the m and p are integer constants and X is a letter

```
printf("%10.2f\n", i);   // m = 10, p = 2, X = f

printf("%10f\n", i);   // m = 10, p = missing, X = f

printf("%f\n", i);   // m and p are miss, X = f
```

Let' go C

Introduction - Input / Output (I/O)

Output - printf()

- A conversion specification can have the form %m.pX or %-m.pX, where the m and p are integer constants and X is a letter

```
printf("%10.2f\n", i);   // m = 10, p = 2, X = f

printf("%10f\n", i);   // m = 10, p = missing, X = f

printf("%f\n", i);   // m and p are miss, X = f
```

- m specifies the minimum number of characters to print

```
int i = 123;
printf("%10d\n", i);   // It will print • • • • • • • 123
```

- Putting a minus sign in front of m causes left justification

```
printf("%-10d\n", i);   // It will print 123 • • • • • • •
```

Let' go C

Introduction - Input / Output (I/O)

Output - printf()

- p, depending on the choice of X, indicates the minimum number of digits to display (extra zeros are added to the beginning of the number if necessary)

```c
int i = 314;
printf("i = %.4d\n", i);

float f = 3.141592;
printf("f = %.4f\n", f);

printf("f = %10.4f\n", f);
```

```
i = 0314
f = 3.1416
f =     3.1416
```

Let' go C

Introduction - Input / Output (I/O)

Input - scanf()

- The one of main output function is scanf()
- It can reads formatted input from the standard input, such as keyboards

scanf(format, *exp* 1, *exp* 2, …);

- Same as printf function, the format string may contain both ordinary characters and conversion specifications, which begin with the % character
- In exp, the & symbol which normally precedes each variable in a scanf call is usually but not always required

scanf("%d,%d", &x, &y);

Let' go C

Introduction - Input / Output (I/O)

Input - scanf()

- How does scanf work?

```
scanf("%d%d%f%f", &i, &j, &x, &y);
```

```
••1¤-20•••.3¤•••-4.0e3¤
ssrsrrrsssrrssssrrrrrr (s = skipped; r = read)
```

Let' go C

Introduction - Comments

Comments

- It is a hint added by the programmer for making code easier to read
- Two ways to add comments
  - ✓ // - Single line comment
  - ✓ /*...*/ - Multi-line comments

```c
// Define a variable p
int p = 3;
/* Print value of p
   in the screen*/
printf("The value is %d\n", p);
```

Operators

- It is a symbol to operate on a value or a variable
- C has a rich collection of operators to perform various operators, such as
  - ✓ arithmetic operators
  - ✓ relational operators
  - ✓ assignment operators
  - ✓ increment and decrement operators
  - ✓ logical operators
  - ✓ …

Let' go C

Introduction - Operators

+	Addition	>>	Bit right shift
-	Subtraction	++	Prefix increment
*	Multiplication	--	Prefix decrement
/	Division	>	Greater than
%	Remainder	>=	Greater than or equal
+	Positive	<	Less than
-	Negative	<=	Less than or equal
~	Complement	==	Equality
&	And	!=	Inequality
|	Or	!	Not
^	XOR	&&	Logical And
=	Assignment	||	Logical Or
<<	Bit left shift		

Let' go C

Introduction - Examples

1. Write a program to add numbers and print the result

```
Please enter first fraction: 5/6
Please enter second fraction: 3/4
The sum is 38/24
```

2. Write a program to read a number i satisfying $100 \leq i \leq 999$ and then print each digit in one line

```
Please input a number (100-999)
253
2
5
3
```