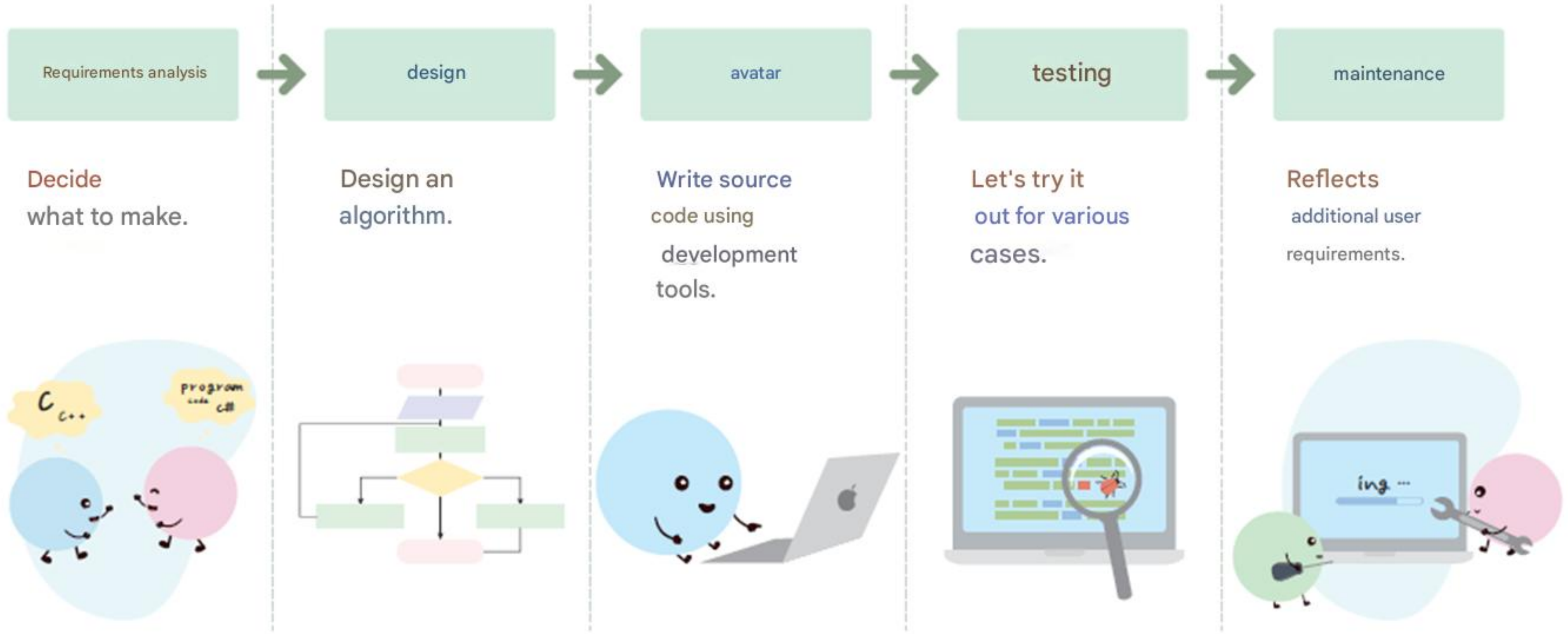
Ch.2 Program Writing Process

Program Development process



Requirements analysis → design → avatar → testing → maintenance

Decide what to make.

Design an algorithm.

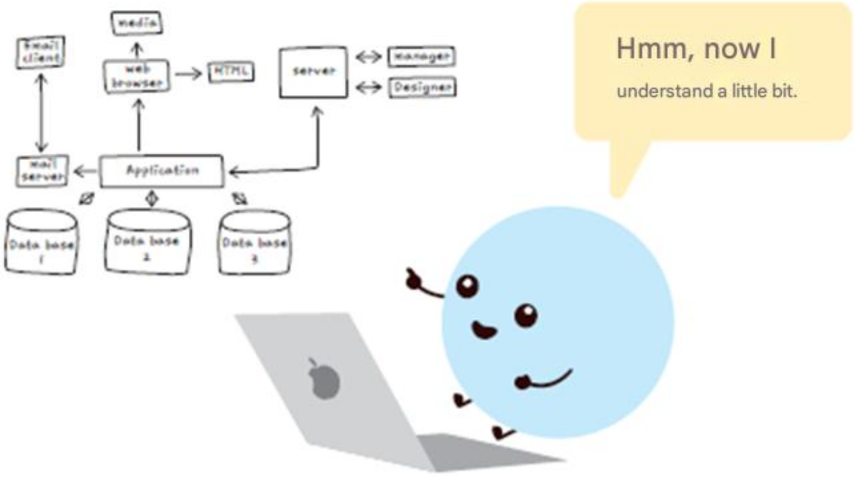
Write source code using development tools.

Let's try it out for various cases.

Reflects additional user requirements.

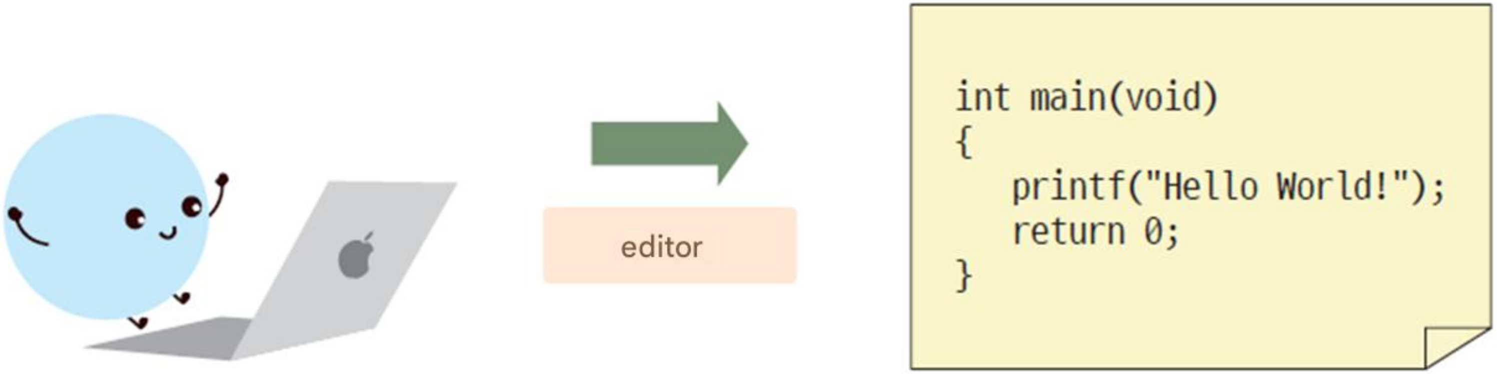
Design

- Steps to develop an algorithm to solve a problem
- Using flowcharts and pseudocode as tools
- Algorithms are independent of programming language
- An algorithm focuses on the steps that must be taken to achieve a desired result.



Write Souce code

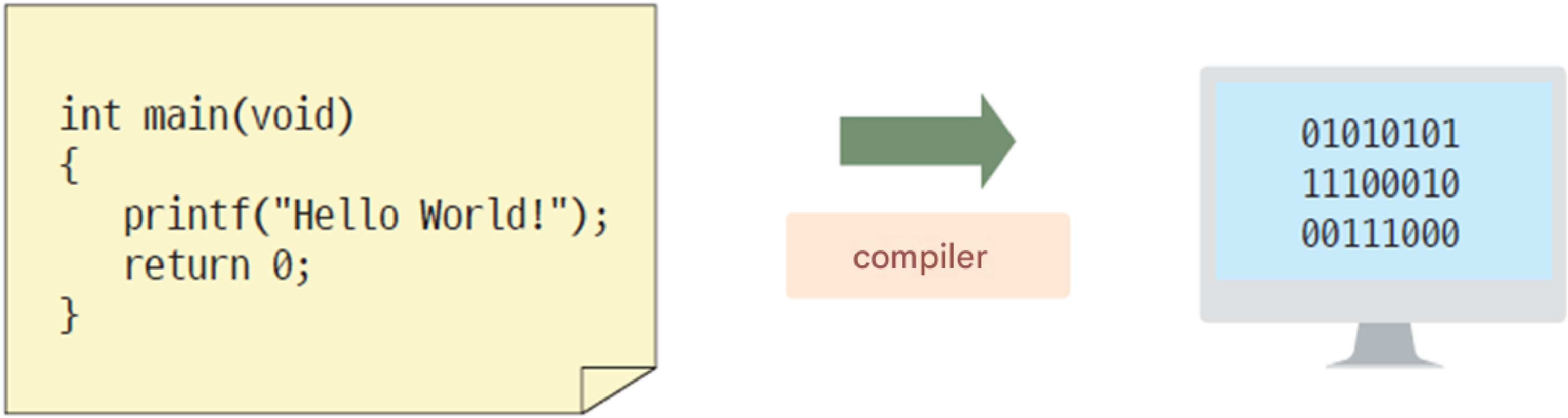
- Describe each step of the algorithm using a programming language.
- An algorithm written in the grammar of a programming language *is called a source program .*
- Source programs are usually written using a text editor or integrated development environment .
- Source file Name : (Example) test.c



```
int main(void)
{
    printf("Hello World!");
    return 0;
}
```

Compile

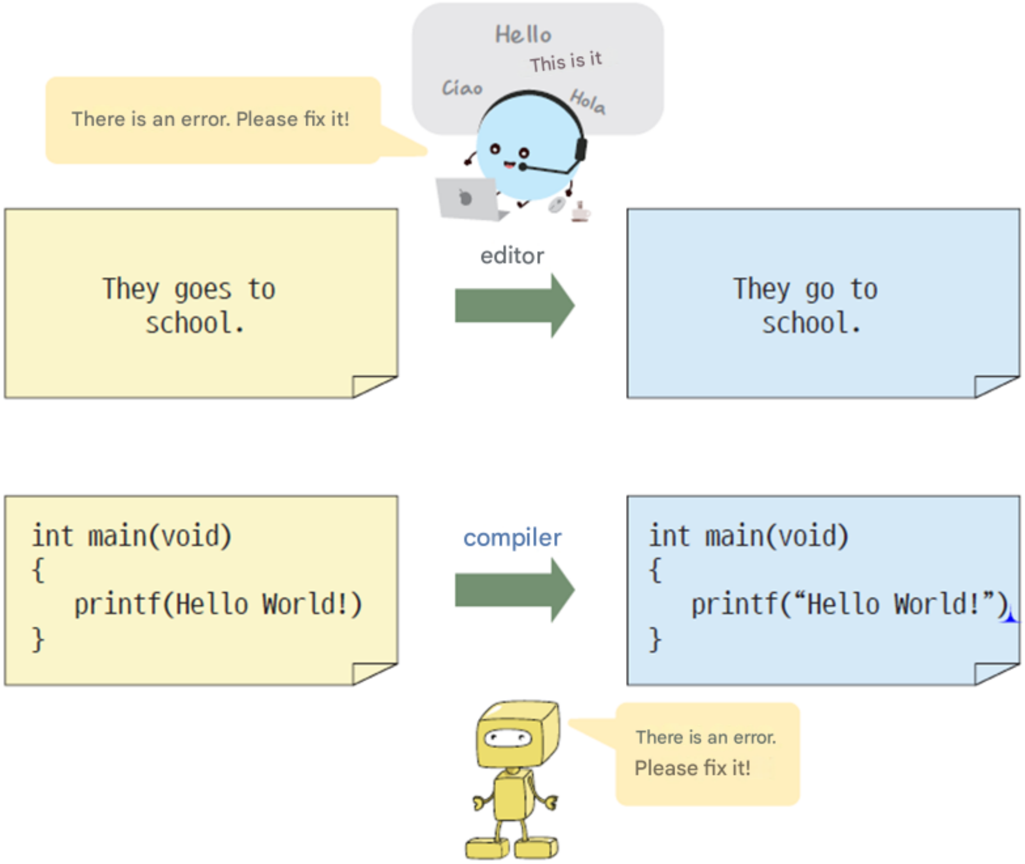
- The task of converting a source program into an object file.
- Object file name : (example) test.obj



```
int main(void)
{
    printf("Hello World!");
    return 0;
}
```

Compile error

- Compile error : Syntax error
  - (Example) He go to school;



They goes to
school.

They go to
school.

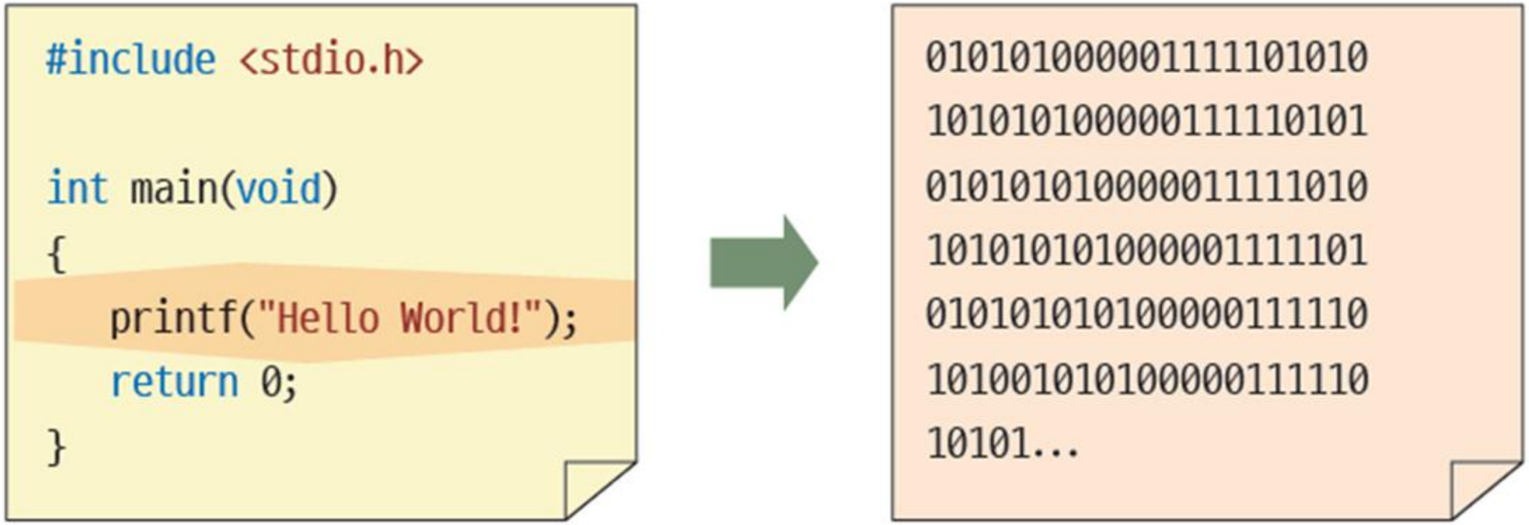
```
int main(void)
{
    printf(Hello World!)
}
```

```
int main(void)
{
    printf(“Hello World!”)
}
```

Link

- Linking a
- Executable file name : (e.g.) test.exe
- *Library :* Pre-written functions that programmers often use
  - (Example) Input/output functions , file processing , mathematical function calculations
- The program that performs linking is called *a linker* .

Object file



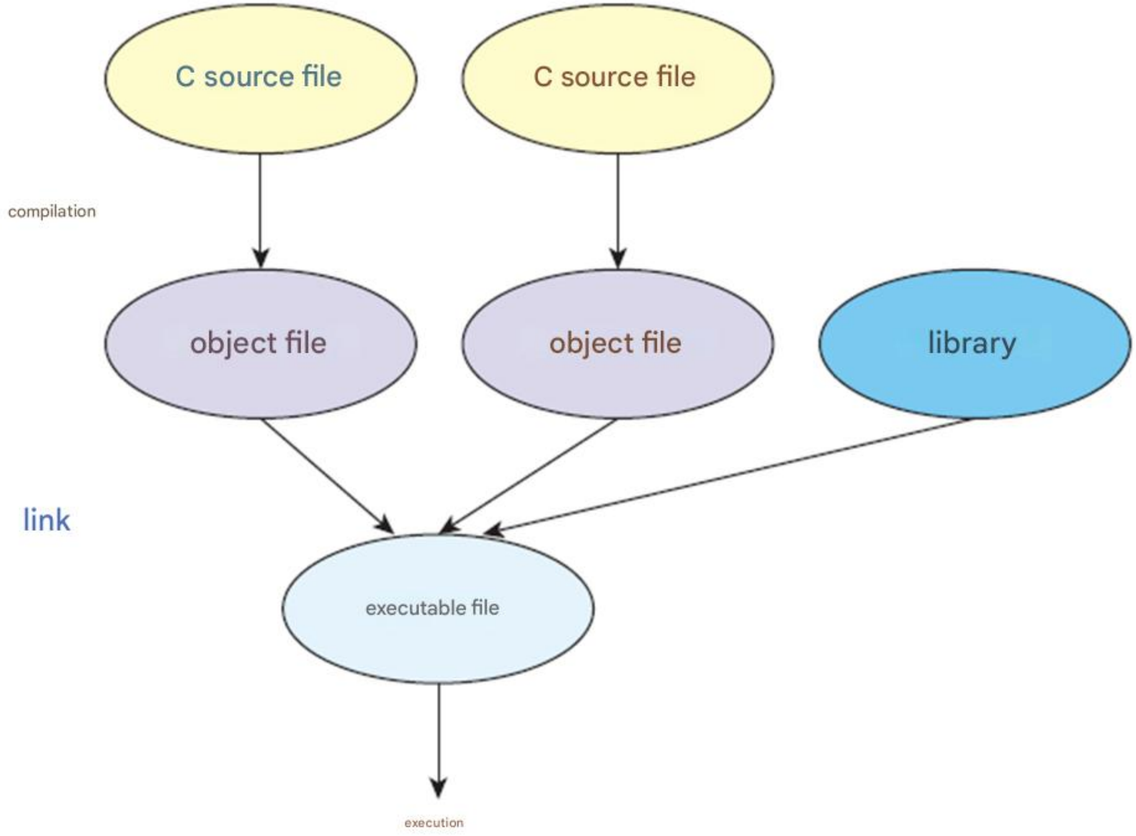
```c
#include <stdio.h>

int main(void)
{
    printf("Hello World!");
    return 0;
}
```

source file

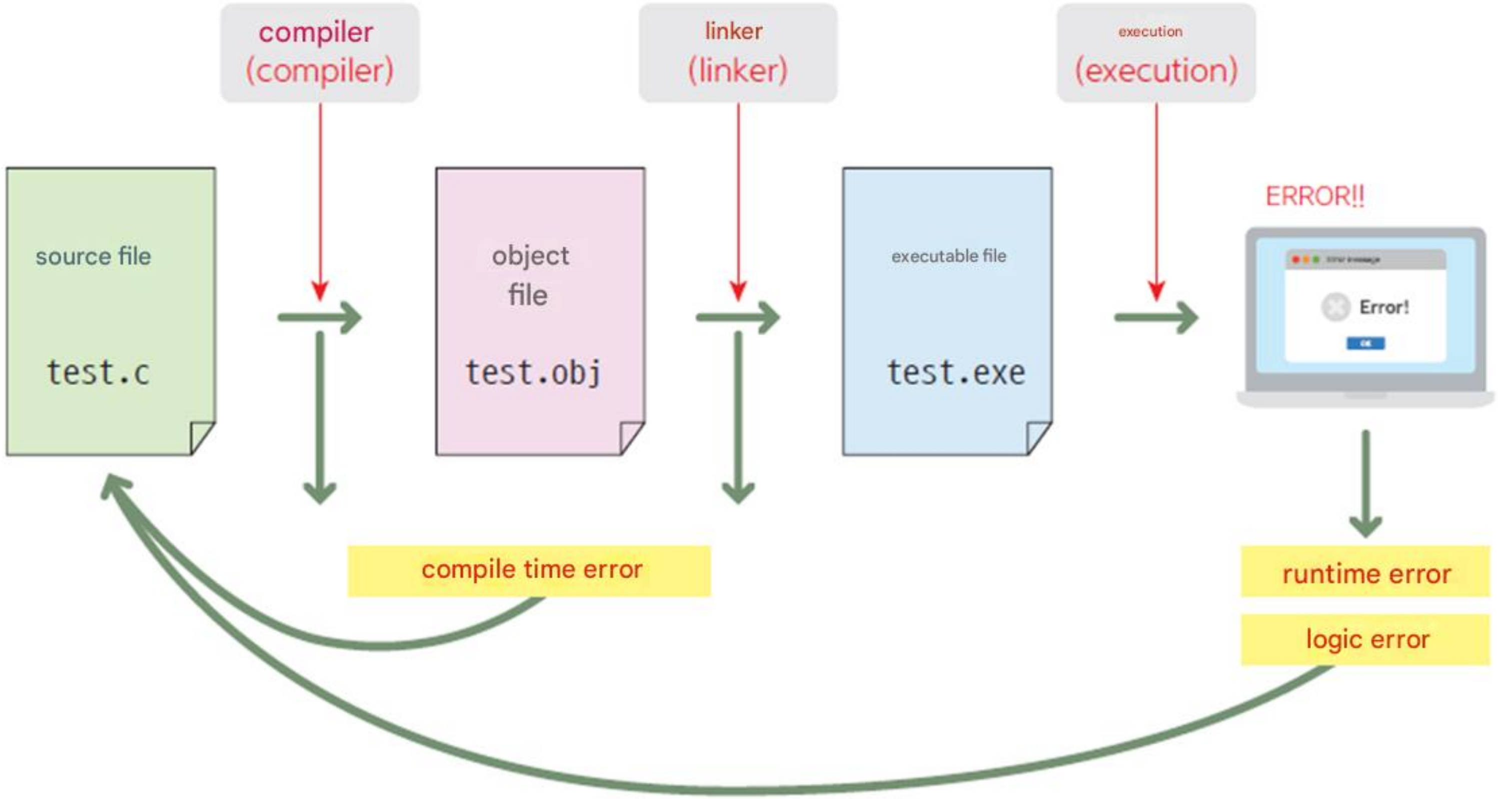
```
010101000001111101010
101010100000111110101
010101010000011111010
101010101000001111101
010101010100000111110
101001010100000111110
10101...
```

object file

Link


C source file
C source file
compilation
object file
object file
library
link
executable file
execution

Running and Debugging



Running and Debugging

- Run time error :
  - division by zero
  - Accessing an invalid memory address
- Logical error : Grammatically correct, but logically incorrect

① Prepare bowl 1 and bowl 2 .
② Add flour , milk , and eggs to bowl 1 and mix well .
③ Place bowl 2 in the oven. Bake at 350 degrees for 30 minutes .

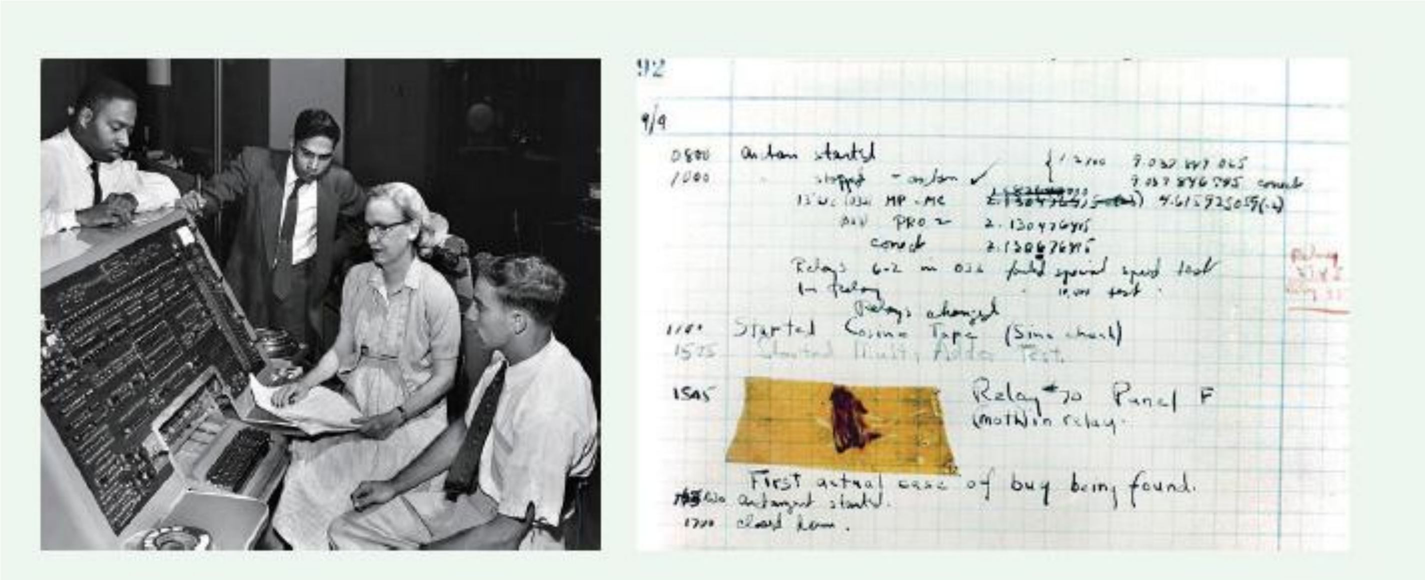
If you accidentally put an empty bowl in the oven, it is a logical error.

Debugging

- Catching errors that exist in the source code

Origin of Debugging

- 1945 , the Mark II computer malfunctioned due to a moth fly ing into its relay unit, which was called a “computer bug . ”
- called Grace , a female computer scientist . Hopper collected moths, recorded them, and reported this as “ debugging” work.



92

9/9

0800 Antan started
1000 " stopped - antan ✓ {1.2700 9.037 847 025
9.037 846 795 correct
13" UC (032) MP - MC ~~1.982147000~~ ~~2.130476415~~ (-3) 4.615925059(-2)
(033) PRO 2 2.130476415
correct 2.130676415
Relays 6-2 in 033 failed special speed test
In relay 10,000 test.
Relays changed
1100 Started Cosine Tape (Sine check)
1525 Started Multi Adder Test.
1545 Relay #70 Panel F
(moth) in relay.
First actual case of bug being found.
1630 Antangent started.
1700 closed down.

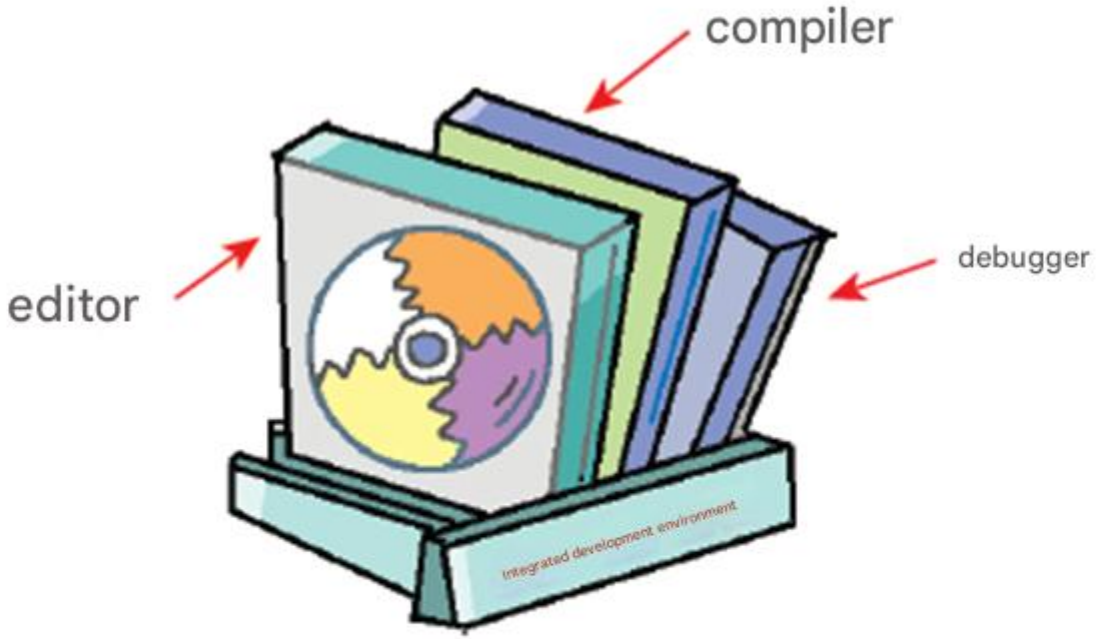
Relay 2145
Relay 3370

Maintenance of software

- Why Software Maintenance Is Necessary
  1. Because bugs may remain even after debugging.
  2. Because user needs can be added after the software is developed.
- Maintenance costs account for more than 50% of total costs

Integrated Development Environment

- Integrated development environment (IDE) = editor + compiler + debugger

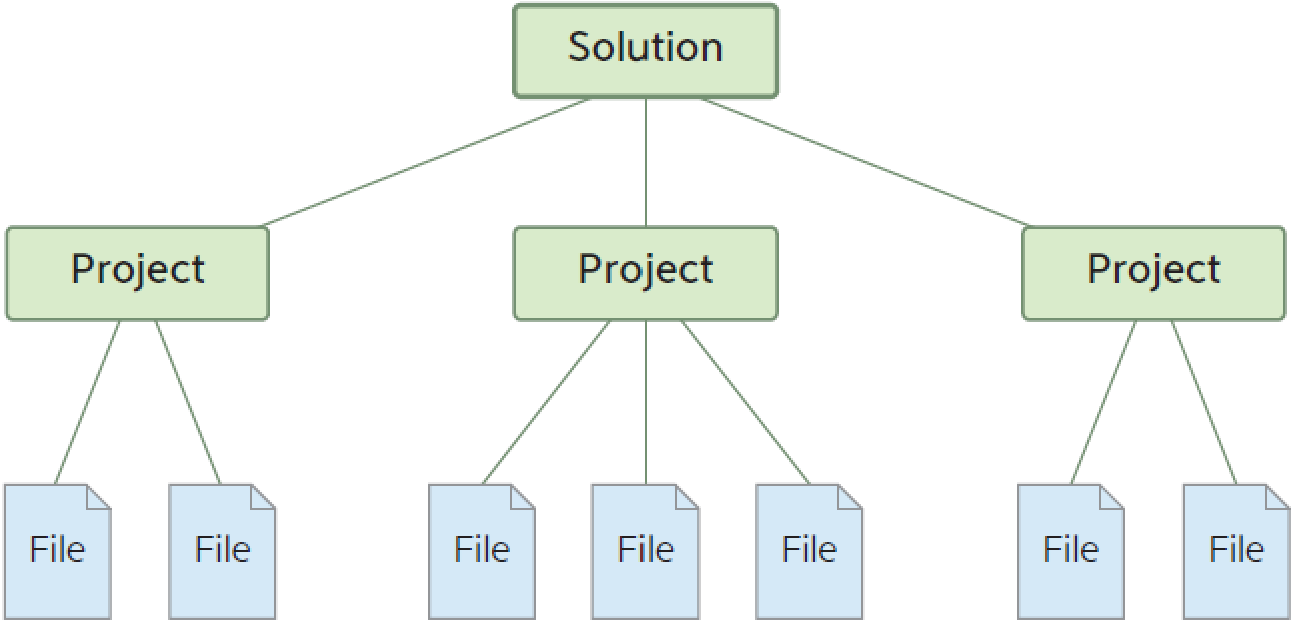


Integrated development environment examples

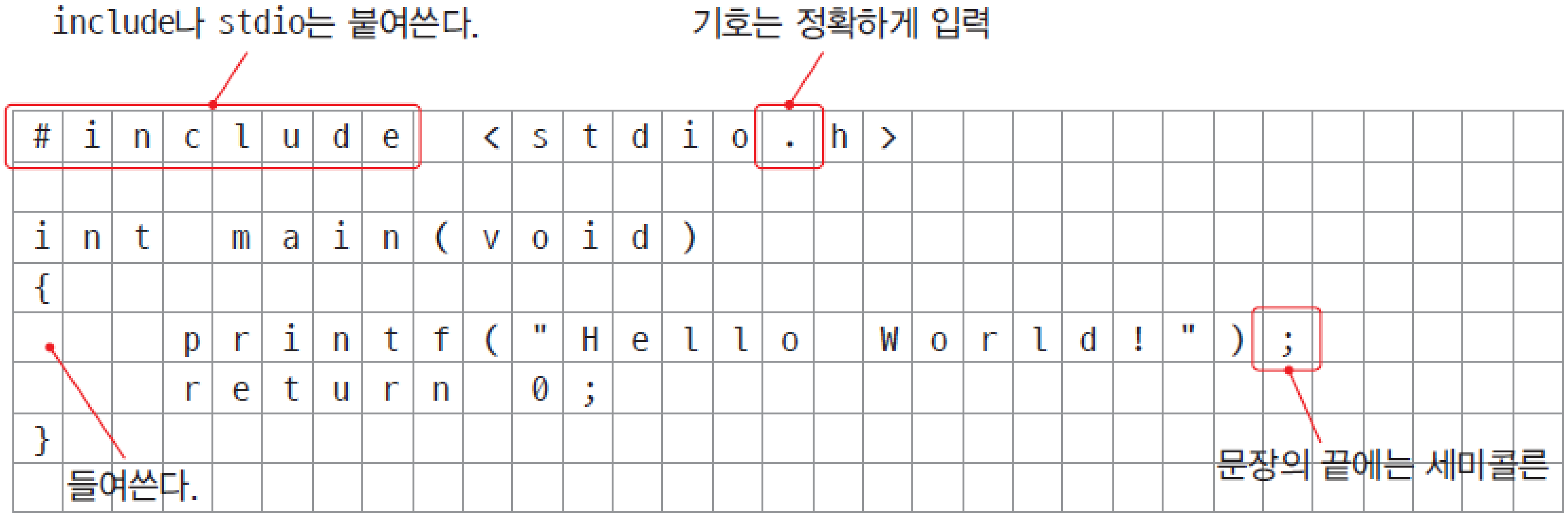
- Visual Studio : Microsoft
- Eclipse : Open Source Project
- Dev-C++: Open source project

Solutions and Projects

- Solution ; problem A container containing the projects needed to be resolved.
- Project : A container that contains several items needed to create a single executable file .



Program input



```
#include <stdio.h>

int main(void)
{
   printf("Hello World!");
   return 0;
}
```

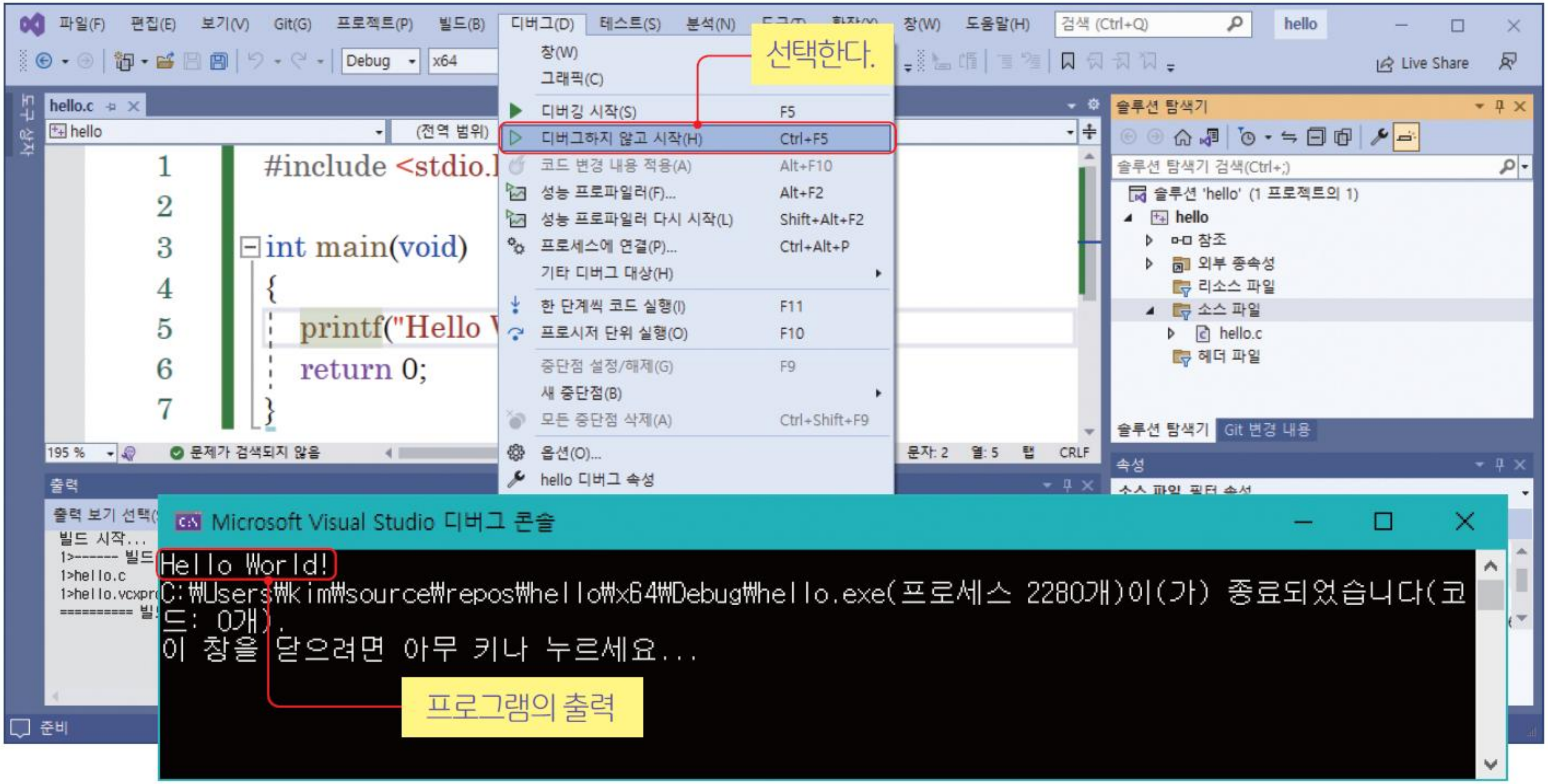
include나 stdio는 붙여쓴다.

기호는 정확하게 입력

들여쓴다.

문장의 끝에는 세미콜른

Running the program



```
#include <stdio.l

int main(void)
{
    printf("Hello \
    return 0;
}
```

```
빌드 시작...
1>------ 빌드
1>hello.c
1>hello.vcxpro
========== 빌
```

```
Hello World!
C:₩Users₩kim₩source₩repos₩hello₩x64₩Debug₩hello.exe(프로세스 2280개)이(가) 종료되었습니다(코드: 0개).
이 창을 닫으려면 아무 키나 누르세요...
```

선택한다.

프로그램의 출력

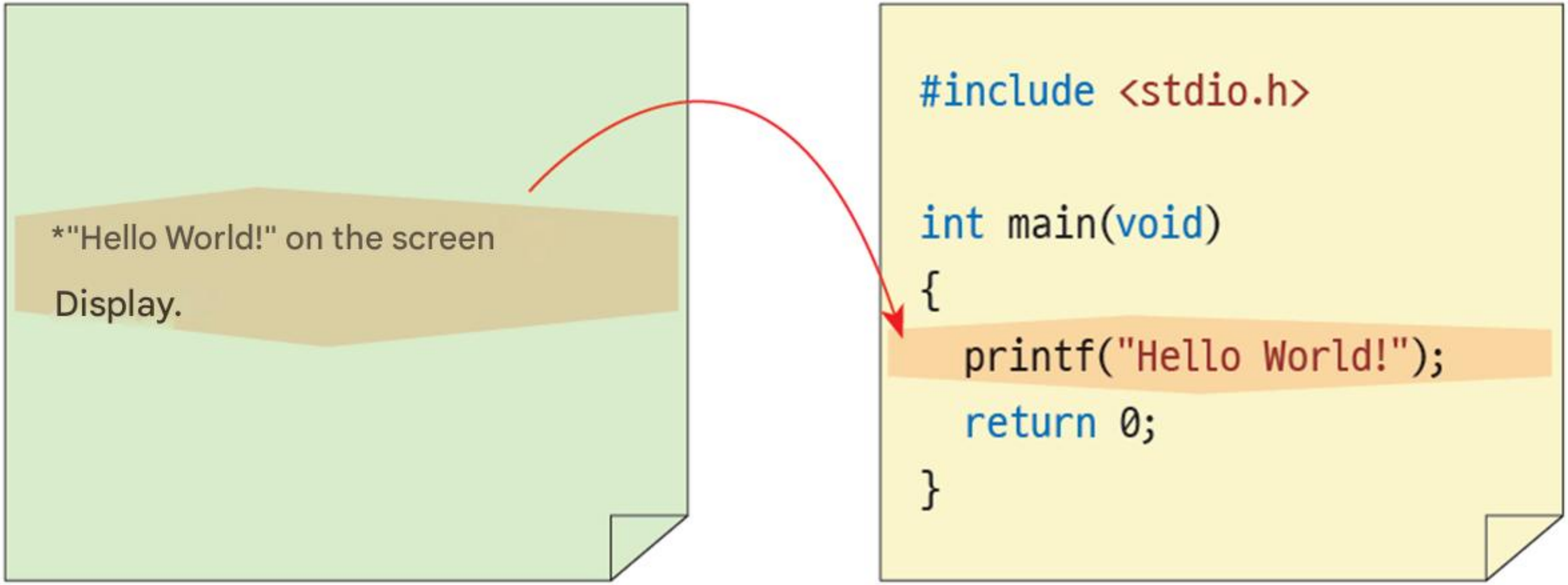
Description of the first program

hello.c

```c
#include <stdio.h>

int main (void)
{
 printf( "Hello World!" ) ;
 return 0;
}
```

```
Hello World!
```

Program == Work Instructions



*"Hello World!" on the screen

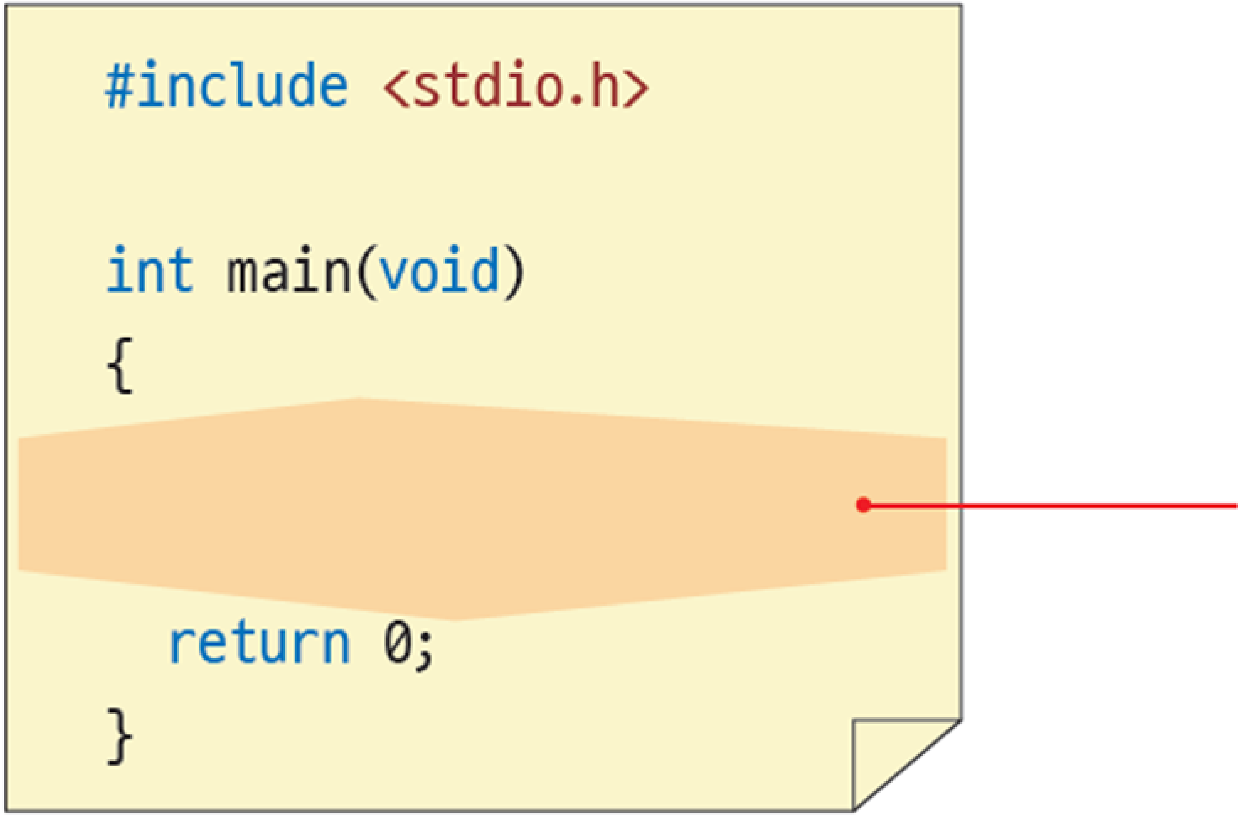
Display.

```c
#include <stdio.h>

int main(void)
{
  printf("Hello World!");
  return 0;
}
```

work order

program

Where to write down your work

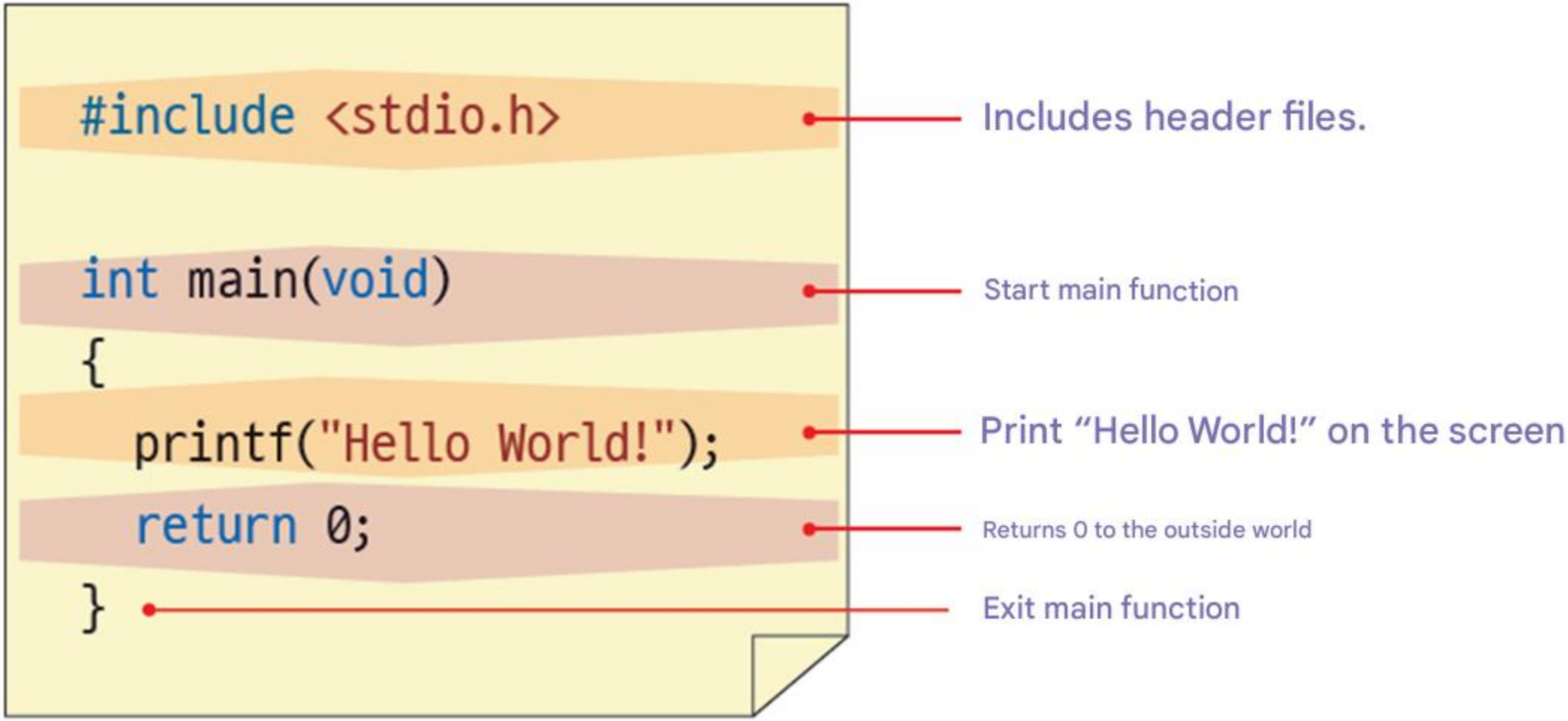


```c
#include <stdio.h>

int main(void)
{

    return 0;
}
```

Write the sentence that performs the desired task here.

program

Brief source description



```c
#include <stdio.h>

int main(void)
{
   printf("Hello World!");
   return 0;
}
```

Includes header files.

Start main function

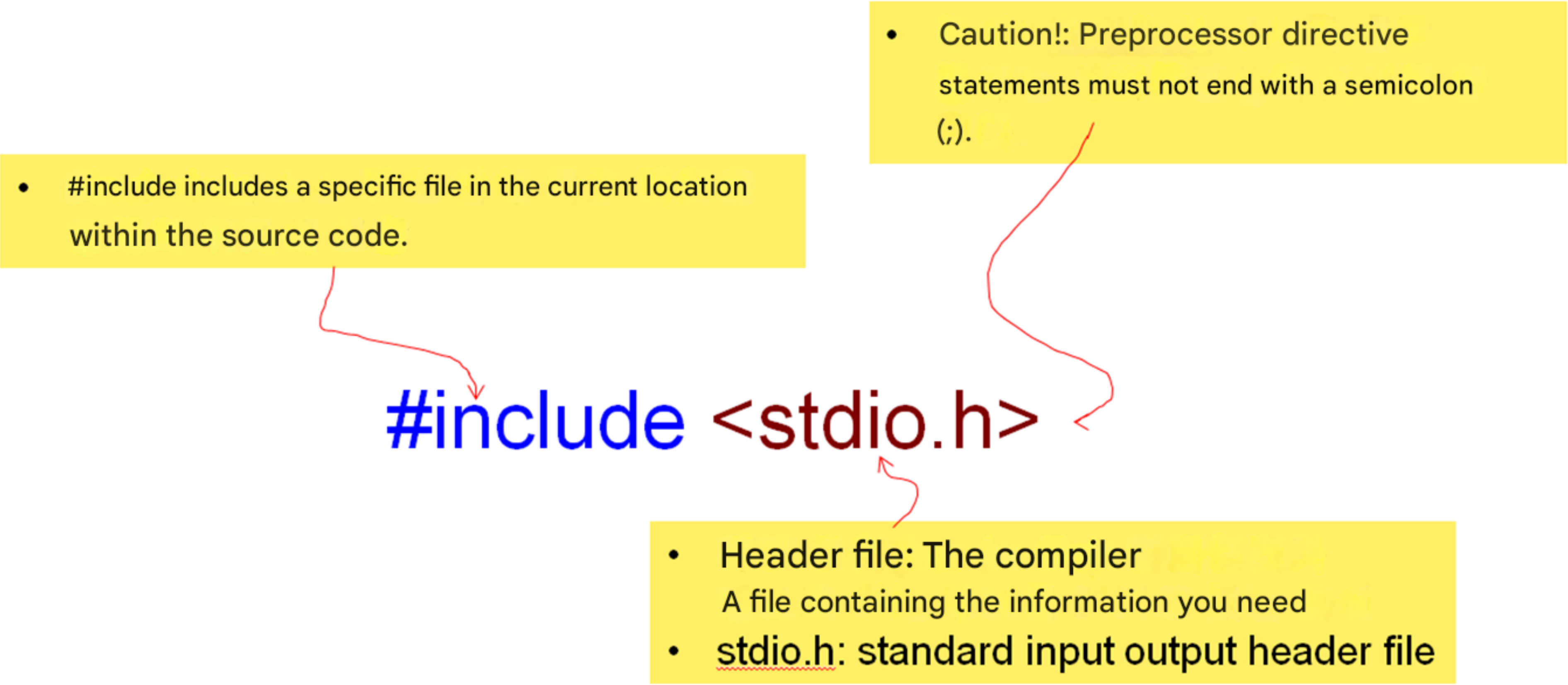
Print “Hello World!” on the screen

Returns 0 to the outside world

Exit main function

program

Including header files



- #include includes a specific file in the current location within the source code.

- Caution!: Preprocessor directive statements must not end with a semicolon (;).

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

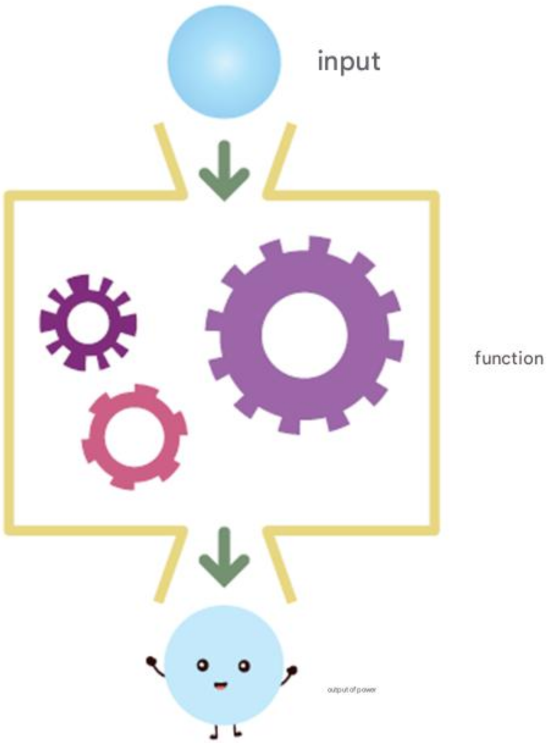
- Header file: The compiler
  A file containing the information you need
- stdio.h: standard input output header file

Function

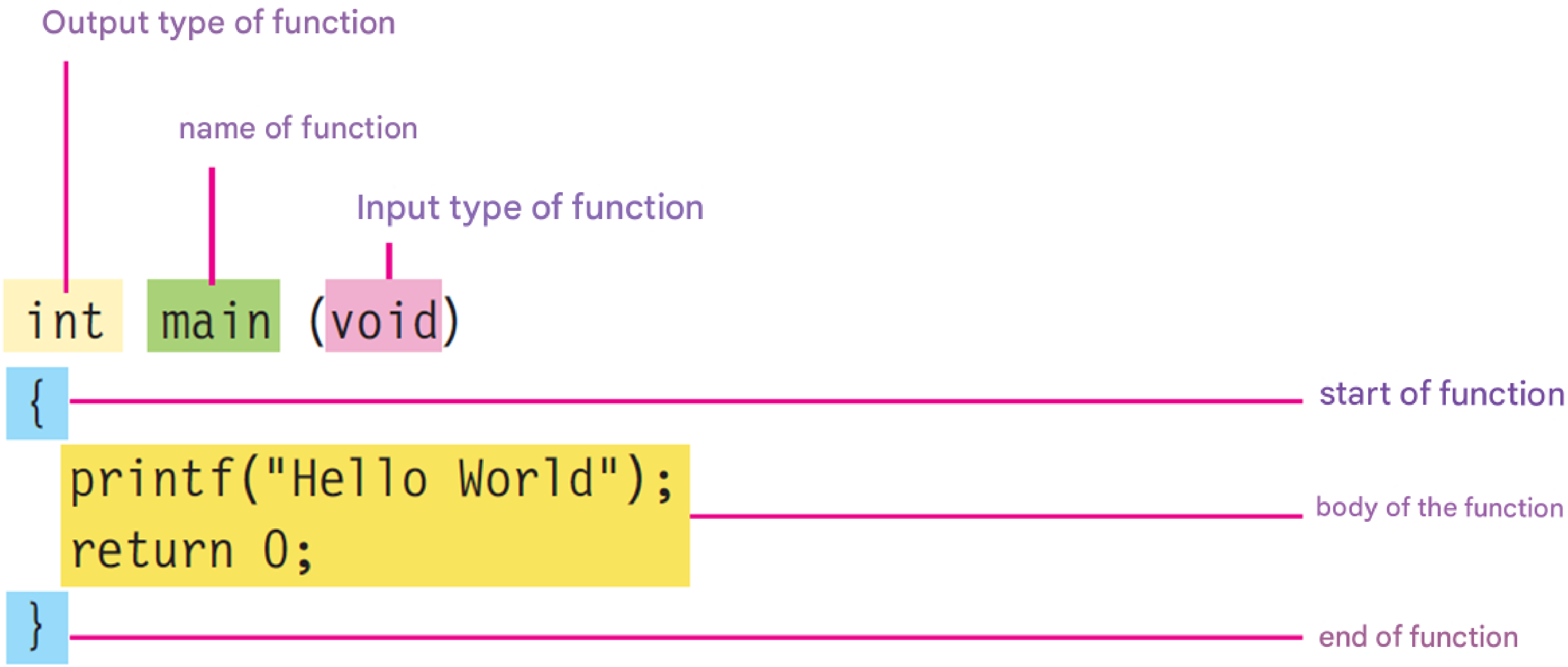
- Function : A standalone piece of code written to perform a specific task.
- (Reference) Mathematical function

$$y = x^2 + 1$$

- program = set of functions



A brief description of the function



Output type of function

name of function

Input type of function

```
int main (void)
{
   printf("Hello World");
   return 0;
}
```

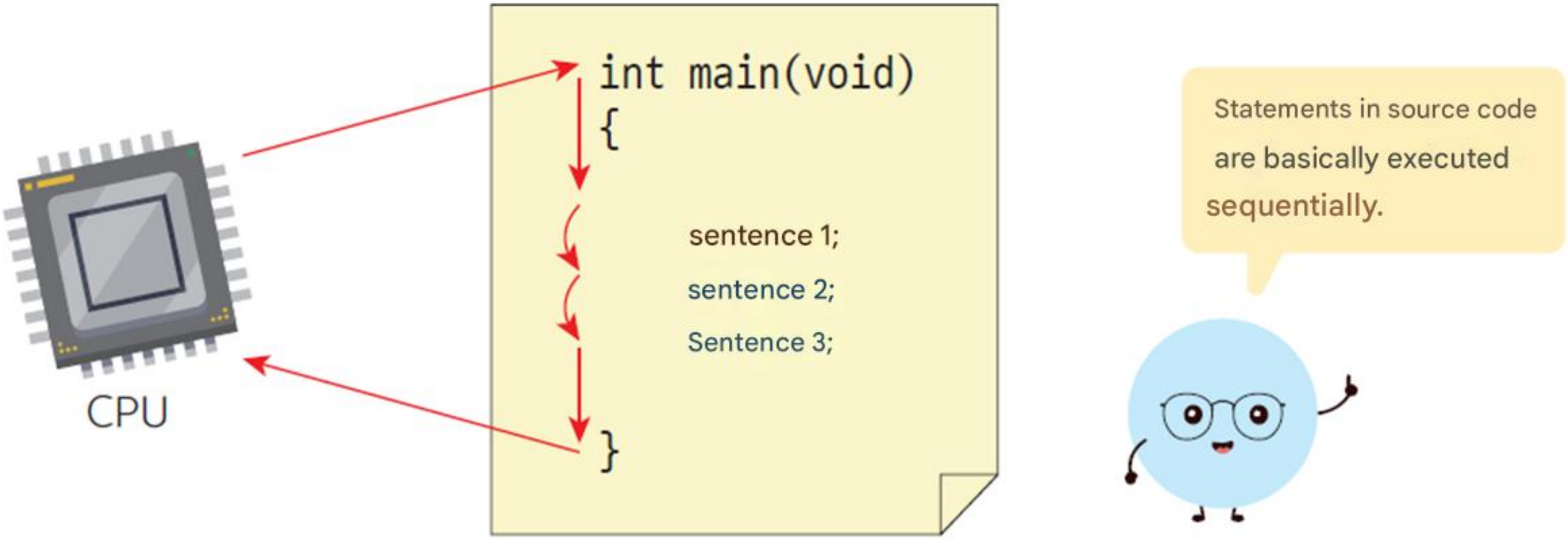
start of function

body of the function

end of function

Sentence (Imperative)

- A function consists of several statements .
- Sentences are executed sequentially .
- There must be a ; at the end of a sentence .

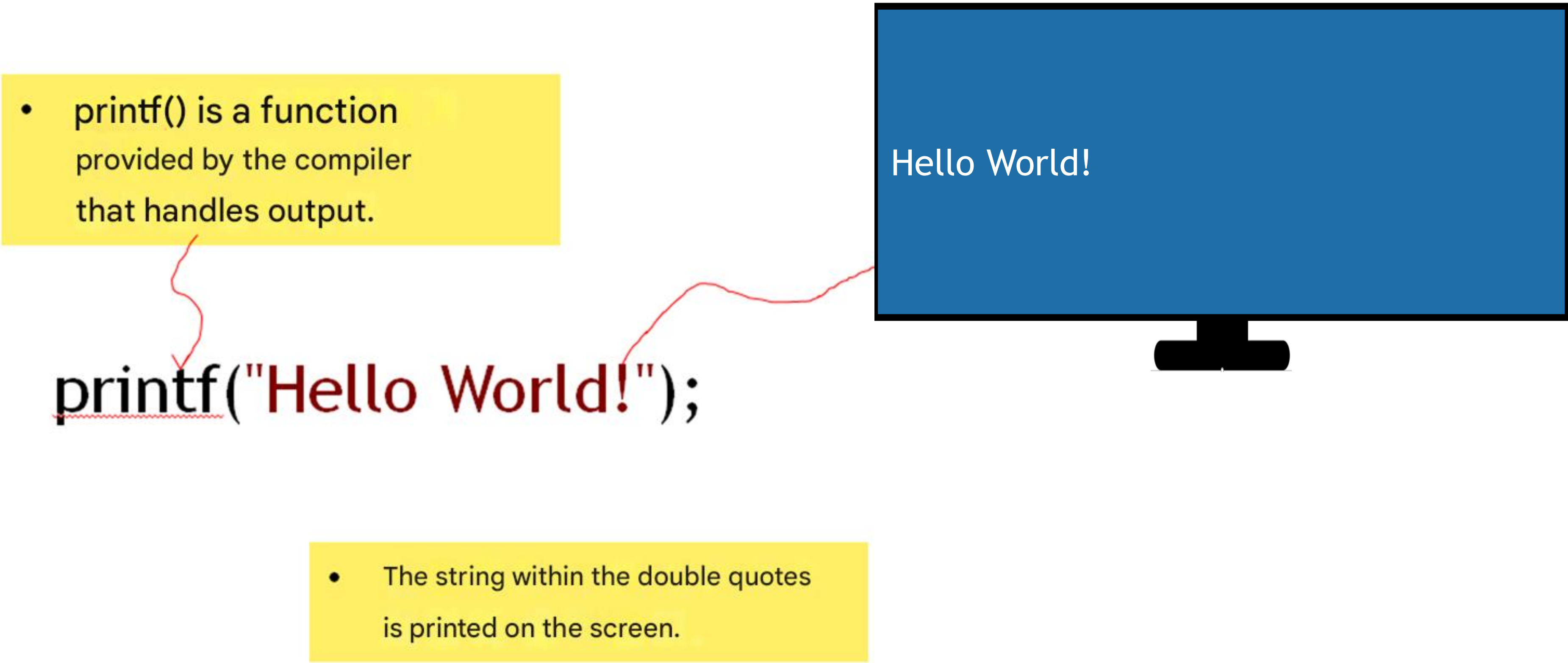


```
int main(void)
{
        sentence 1;
        sentence 2;
        Sentence 3;
}
```

CPU
Statements in source code are basically executed sequentially.

source code

printf () call



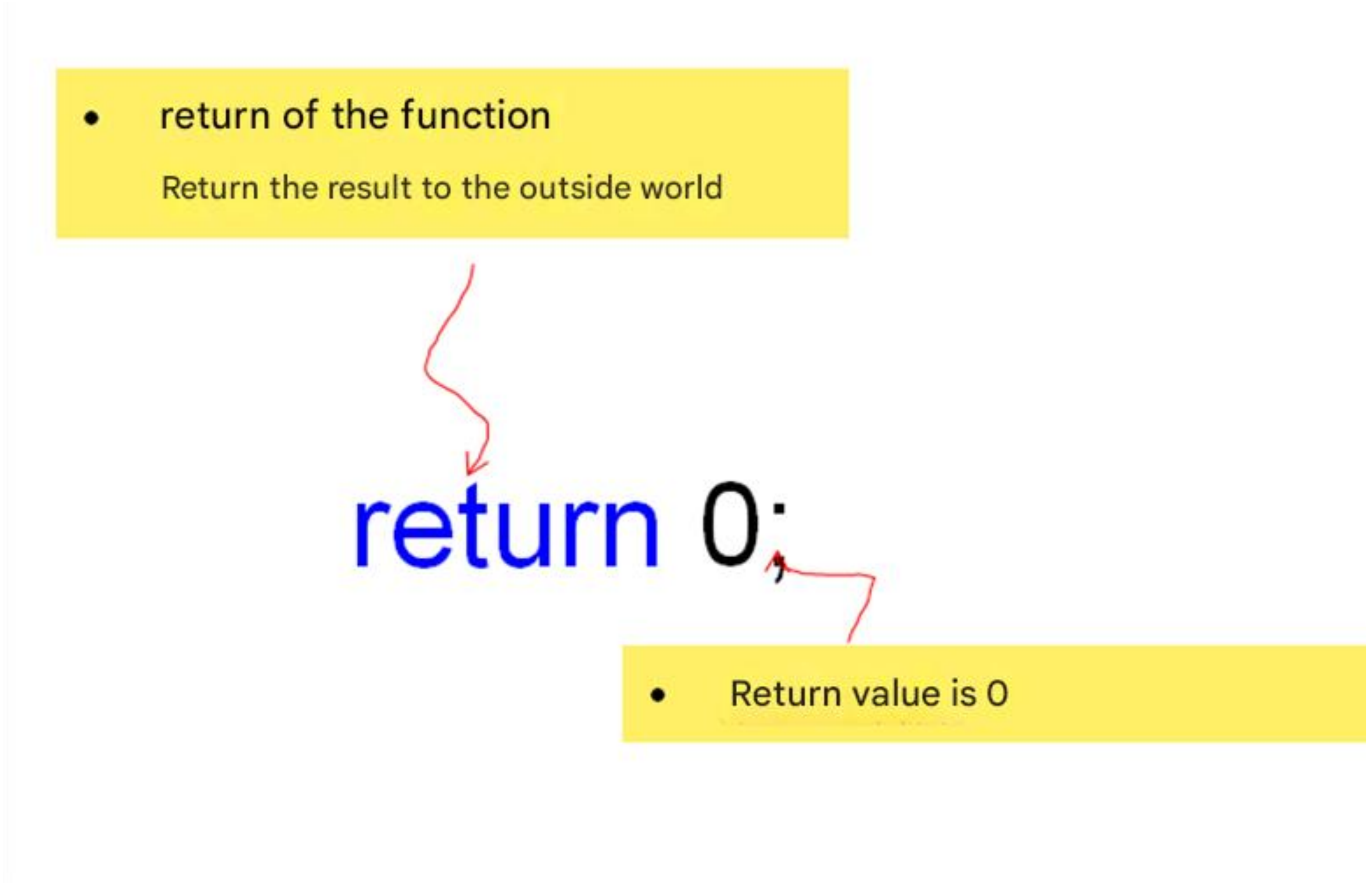
- printf() is a function provided by the compiler that handles output.

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

```
Hello World!
```

- The string within the double quotes is printed on the screen.

Return value



- return of the function

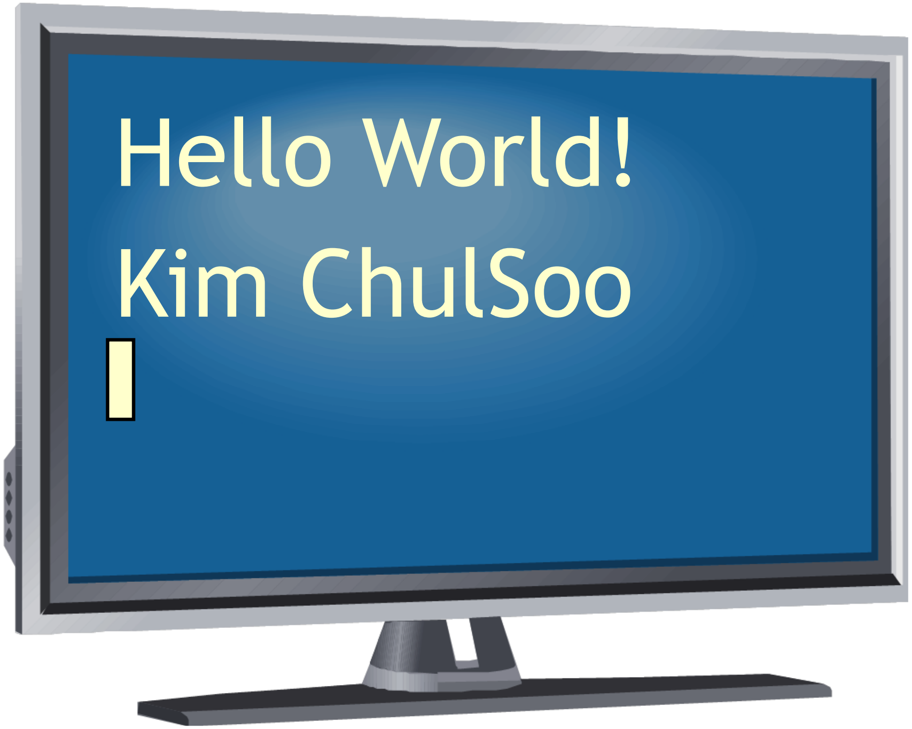
  Return the result to the outside world

```
return 0;
```

- Return value is 0

Application #1

- Let's create a program that produces the following output .



```
Hello World!
Kim ChulSoo
```

First version

- Take advantage of the fact that sentences are executed sequentially

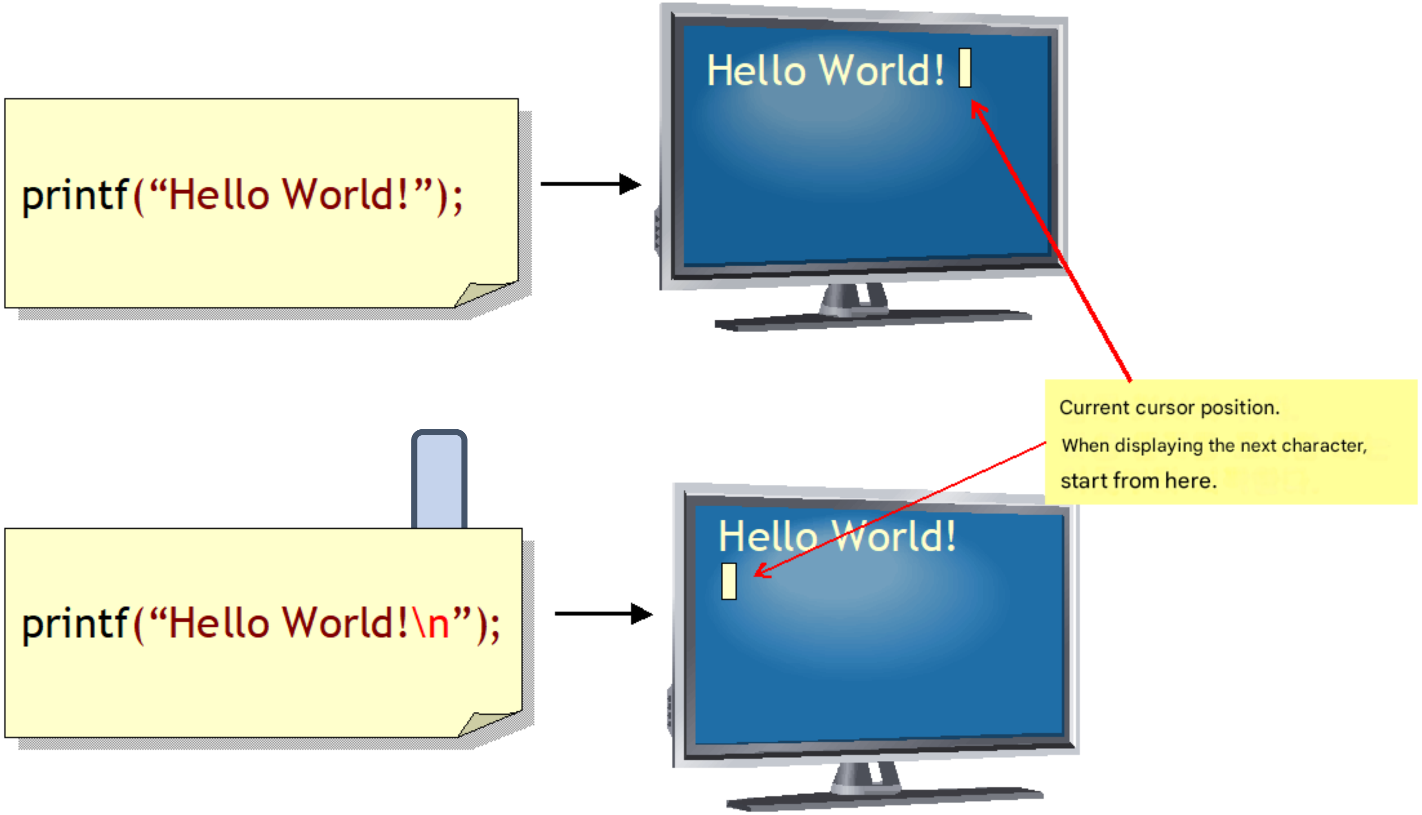
```
#include < stdio.h >

int main( void )
{
  printf ( "Hello World!" );
  printf ( "Kim ChulSoo " );

return 0;
}
```

2 of Sentences are executed sequentially.

```
Hello World!  Kim ChulSoo
```

New line character

- The new line character ₩n moves the cursor to the next line on the screen .



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

```
Hello World!
```

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

```
Hello World!
```

Current cursor position.

When displaying the next character, start from here.

Changed program

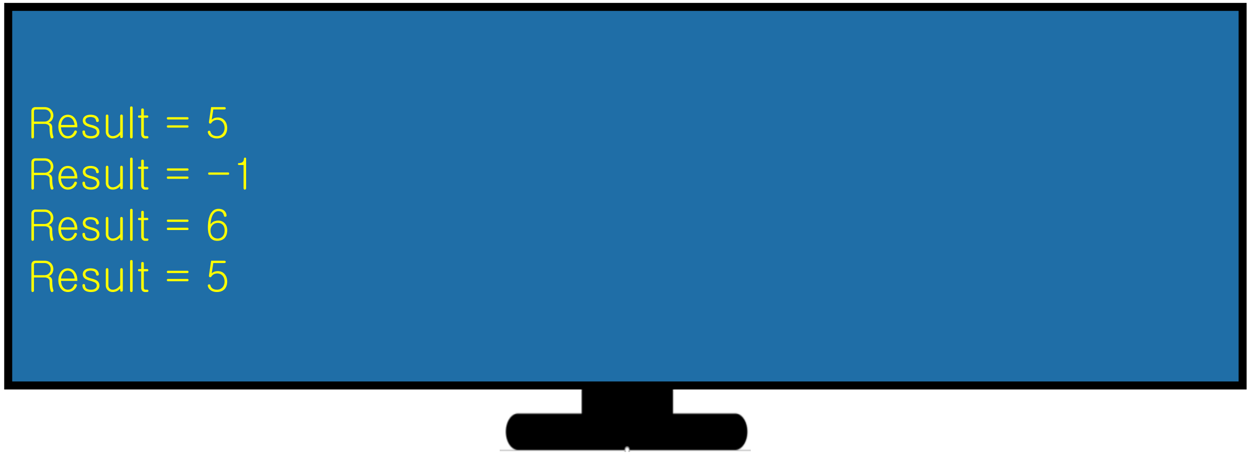
- Adding a new-line character gives us the result we want .

```
#include < stdio.h >

int main( void )
{

  printf ( "Hello World!\n" );
  printf ( "Kim ChulSoo \n" );

return 0;
}
```

```
Hello World!
Kim ChulSoo
▯
```

Lab: Simple Let's do the math

- addition, subtraction , multiplication , and division calculations .



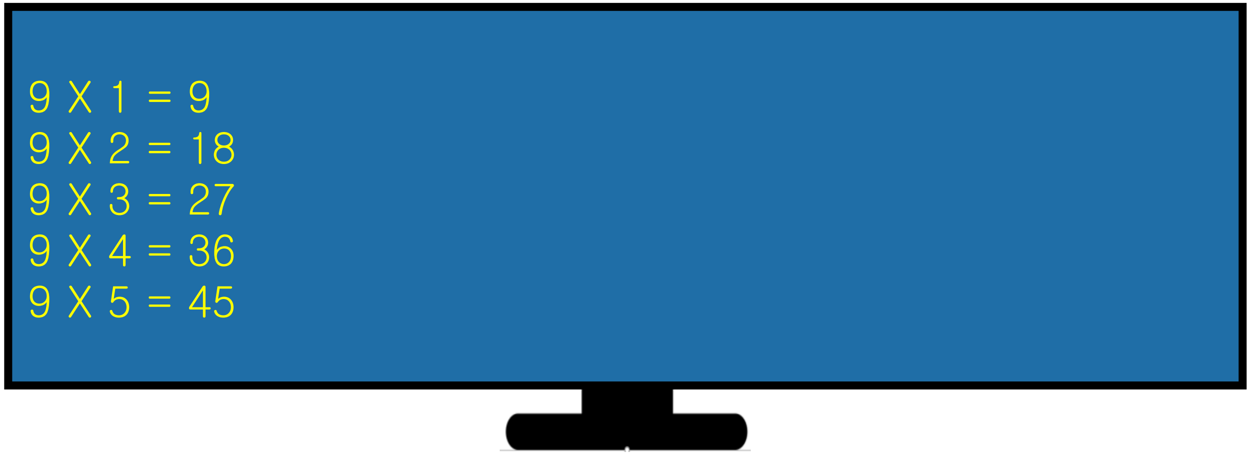
```
Result = 5
Result = −1
Result = 6
Result = 5
```

Solution

```
#include < stdio.h >

int main( void )
{
 printf ( “ Result =%d\n” , 2 + 3);
 printf ( “ Result =%d\n” , 2 - 3);
 printf ( “ Result =%d\n” , 2 * 3);
 printf ( “ Result =%d\n” , 2 / 3);
 return 0;
}
```

Lab: Multiplication Table Let's print it out .

- Let's write a program that prints part of the 9th digit of the multiplication table .



```
9 X 1 = 9
9 X 2 = 18
9 X 3 = 27
9 X 4 = 36
9 X 5 = 45
```

Solution

```
#include < stdio.h >

int main( void )
{
printf ( “9
printf ( “9
printf ( “9
printf ( “9
printf ( “9
return 0;
}
```

Let's modify the code to print all 9 columns .

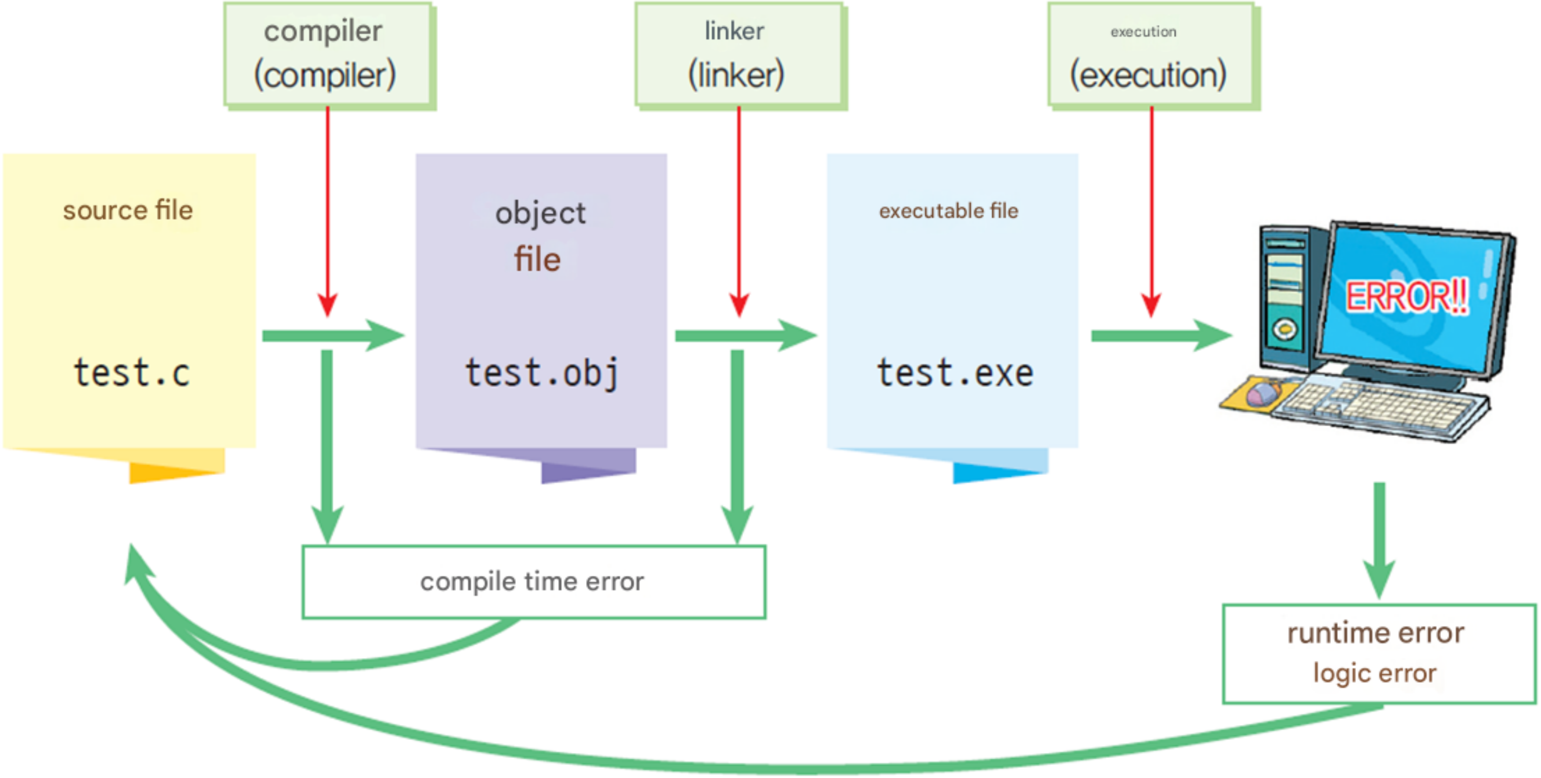
Error Fixing and Debugging

- Errors may occur during compilation or execution .
- Errors and Warnings
  - error : fatal error
  - Warning : Minor error

Types of errors

- Types of errors
  - Compile time errors : mostly grammatical errors
  - Runtime errors : Errors such as division by 0 (zero)
  - Logical error : An error that is logically incorrect and results in an outcome that is not what was intended.

Error correction process



compiler
(compiler)

linker
(linker)

execution
(execution)

source file
test.c

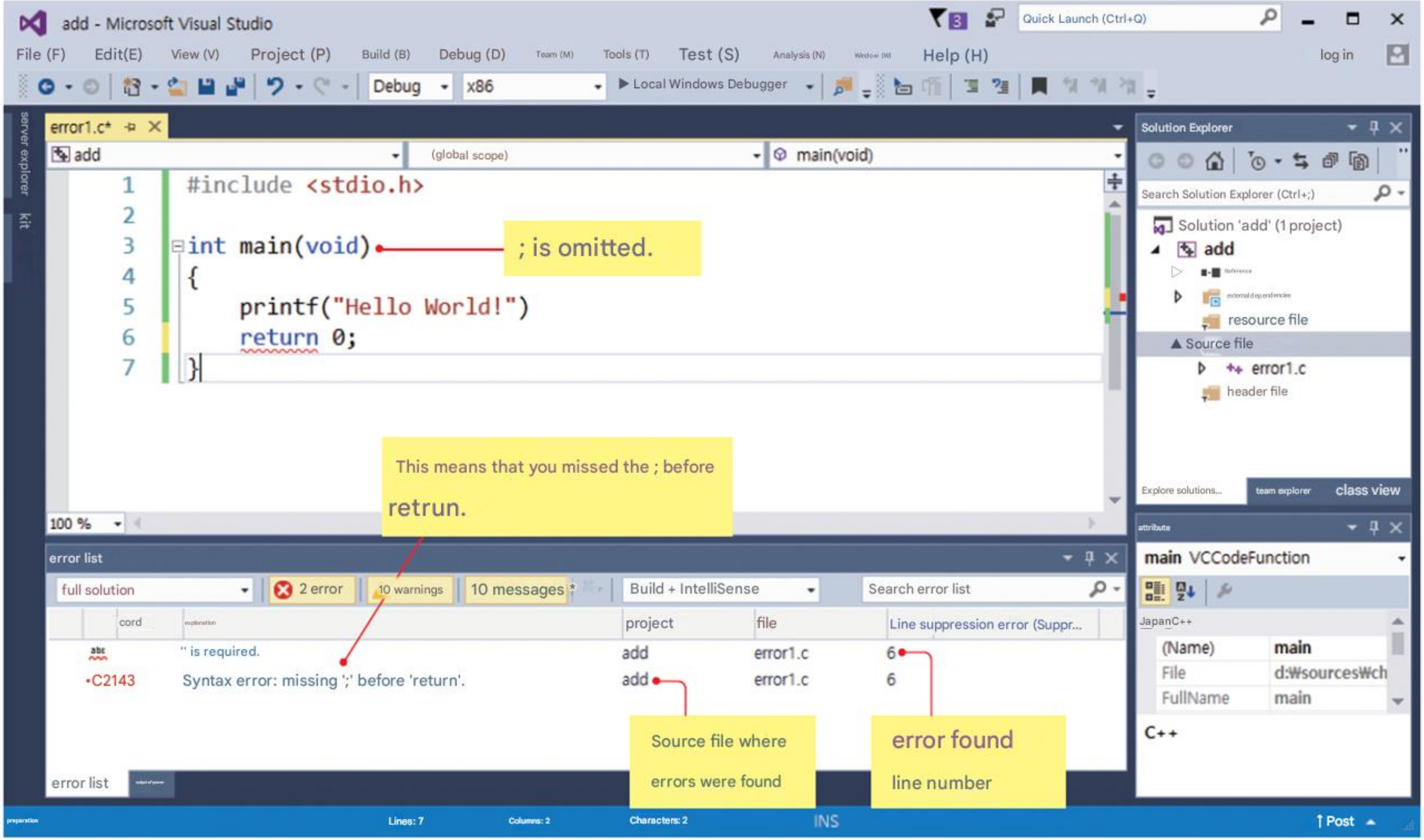
object
file
test.obj

executable file
test.exe

compile time error

runtime error
logic error

Error #1



```c
1  #include <stdio.h>
2
3  int main(void)
4  {
5      printf("Hello World!")
6      return 0;
7  }
```

; is omitted.

This means that you missed the ; before retrun.

error list

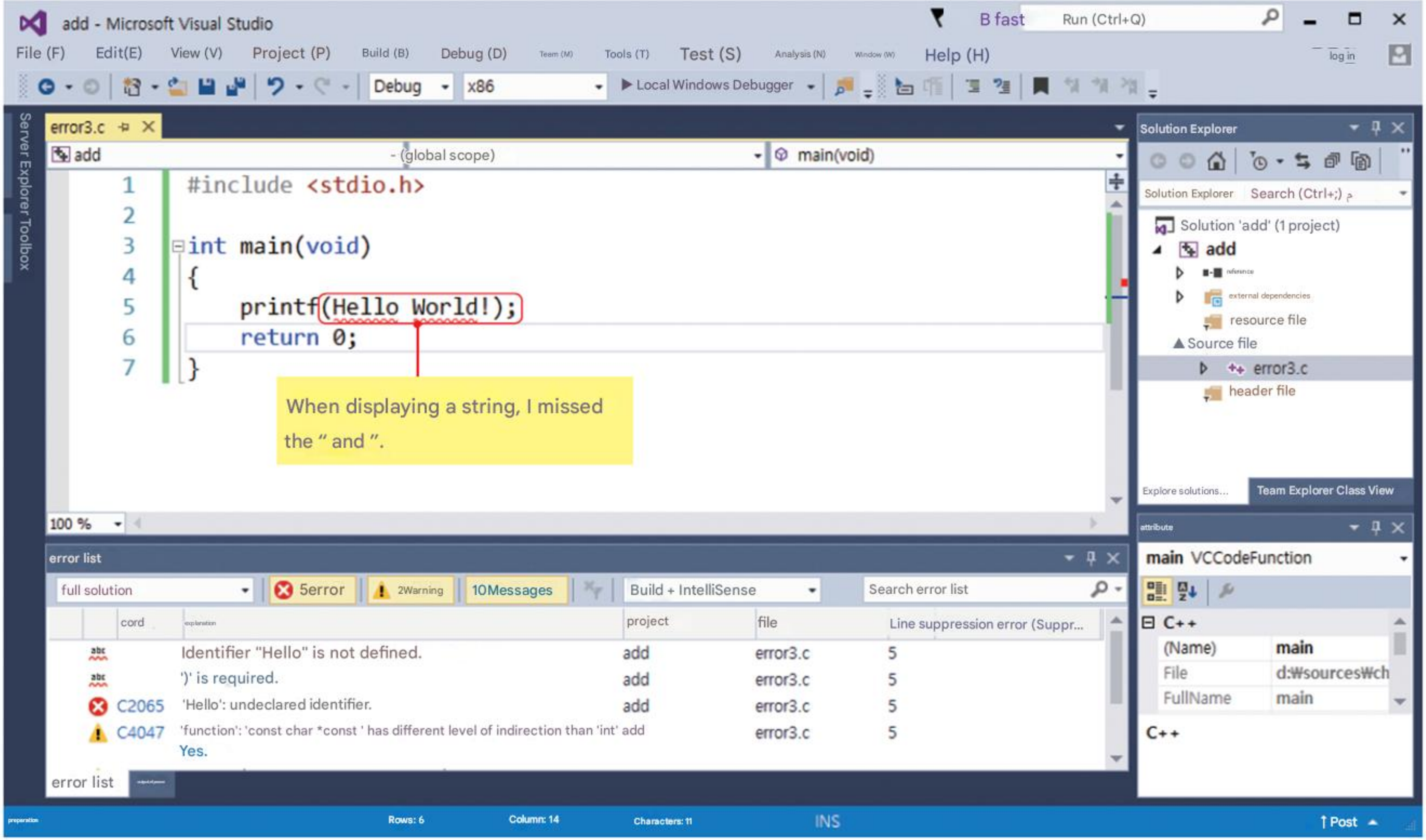
full solution | ✖ 2 error | 10 warnings | 10 messages | Build + IntelliSense | Search error list

	cord	explanation	project	file	Line suppression error (Suppr...
	abc	'' is required.	add	error1.c	6
	•C2143	Syntax error: missing ';' before 'return'.	add	error1.c	6

Source file where errors were found

error found line number

Error #2



```
#include <stdio.h>

int main(void)
{
    printf(Hello World!);
    return 0;
}
```

When displaying a string, I missed the “ and ”.

main VCCodeFunction

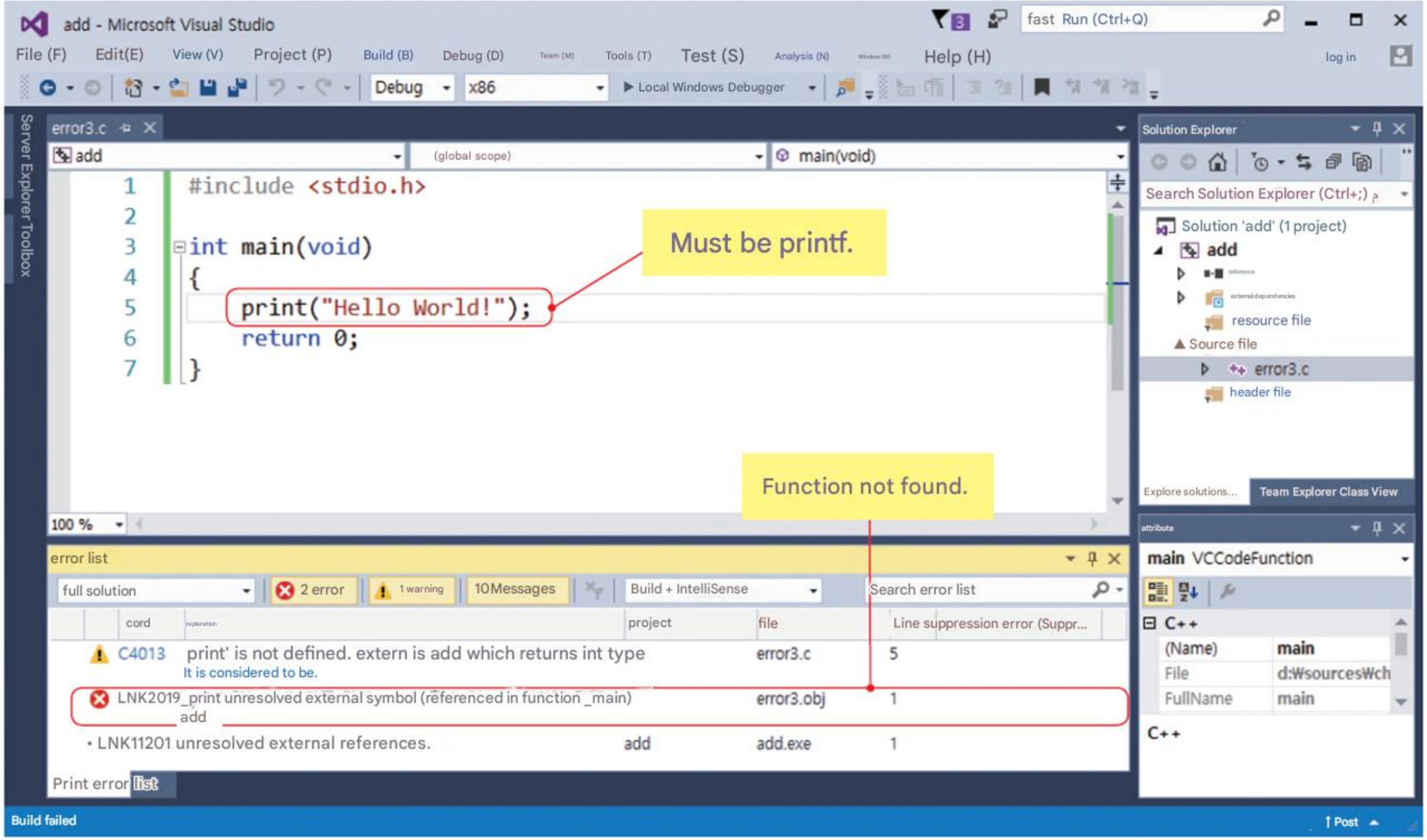
C++

(Name)	main
File	d:₩sources₩ch
FullName	main

C++

	cord	explanation	project	file	Line suppression error (Suppr...
abc		Identifier "Hello" is not defined.	add	error3.c	5
abc		')' is required.	add	error3.c	5
✖	C2065	'Hello': undeclared identifier.	add	error3.c	5
⚠	C4047	'function': 'const char *const ' has different level of indirection than 'int' add Yes.		error3.c	5

Error #3



```
#include <stdio.h>

int main(void)
{
    print("Hello World!");
    return 0;
}
```

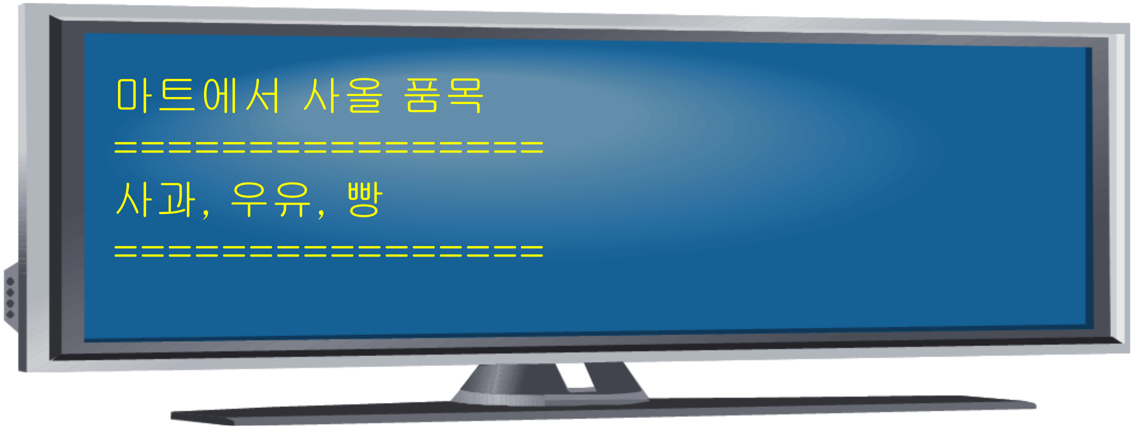
Must be printf.

Function not found.

	cord	explanation	project	file	Line suppression error (Suppr...
⚠	C4013	print' is not defined. extern is add which returns int type It is considered to be.		error3.c	5
⊗	LNK2019	_print unresolved external symbol (referenced in function _main) add		error3.obj	1
•	LNK11201	unresolved external references.	add	add.exe	1

Logical error

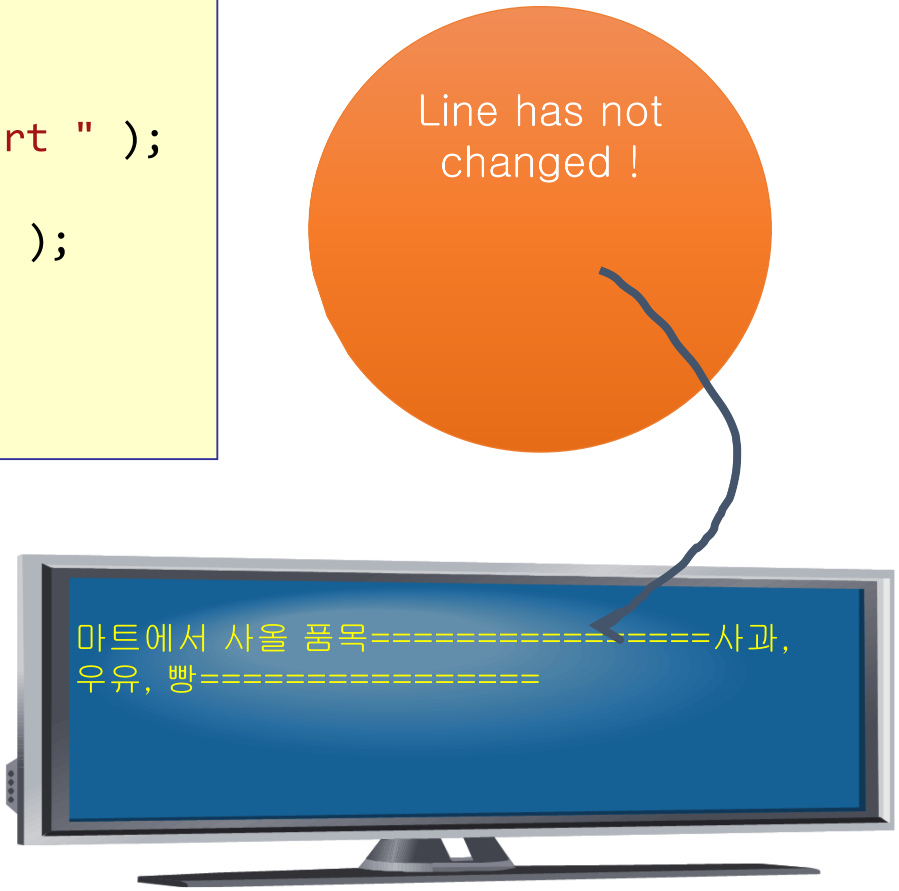
- Let's write a program that produces the following output :



```
마트에서 사올 품목
================
사과, 우유, 빵
================
```

Program with logic errors

```c
#include < stdio.h >

int main( void )
{
 printf ( " Items to buy at the mart " );
 printf ( "================" );
 printf ( " apple , milk , bread " );
 printf ( "================" );
 return 0;
}
```



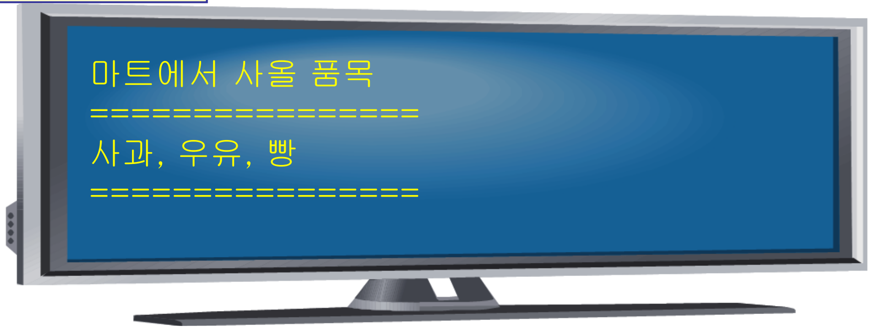
Line has not changed !

```
마트에서 사올 품목================사과,
우유, 빵================
```

Program with corrected logic errors

```
#include < stdio.h >

int main( void )
{
 printf ( " Items to buy at the mart \n" );
 printf ( "================\n" );
 printf ( " apple , milk , bread \n" );
 printf ( "================\n" );
 return 0;
}
```

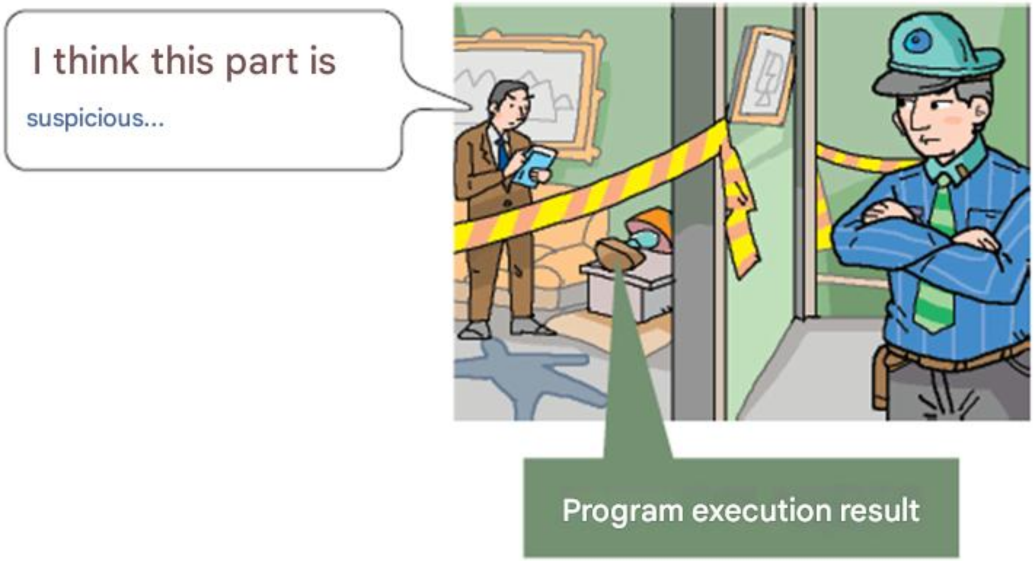
Fix logic error !!

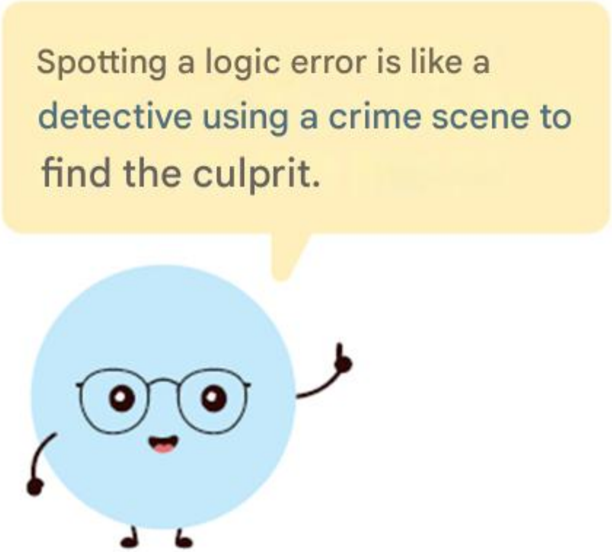


```
마트에서 사올 품목
================
사과, 우유, 빵
================
```

Debugging

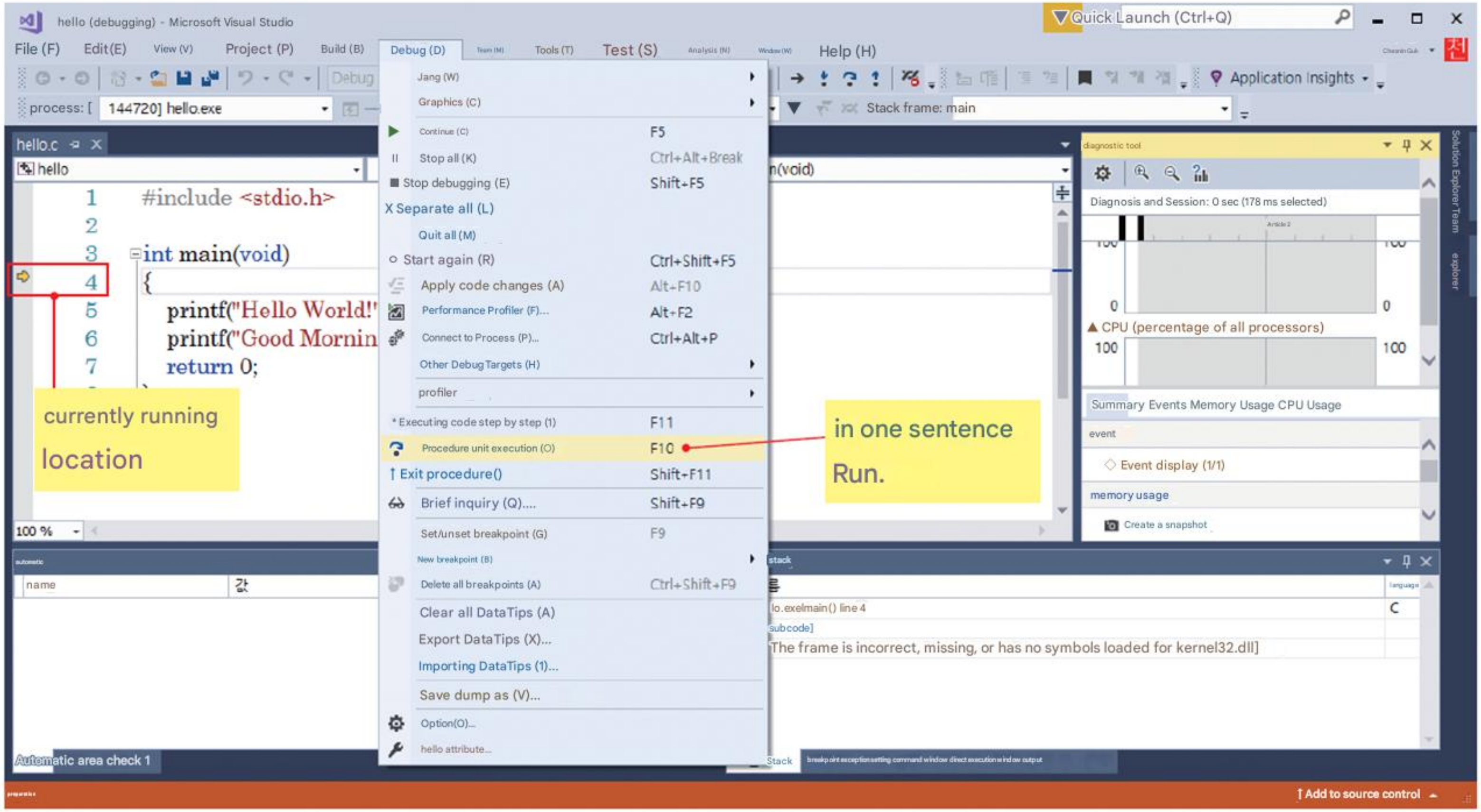
- Debugging : The process of finding logic errors




Spotting a logic error is like a detective using a crime scene to find the culprit.

Debugger

- A tool to find the cause of an error by executing a program one sentence at a time.



```
#include <stdio.h>

int main(void)
{
    printf("Hello World!"
    printf("Good Mornin
    return 0;
```

currently running location

in one sentence Run.

lo.exelmain() line 4

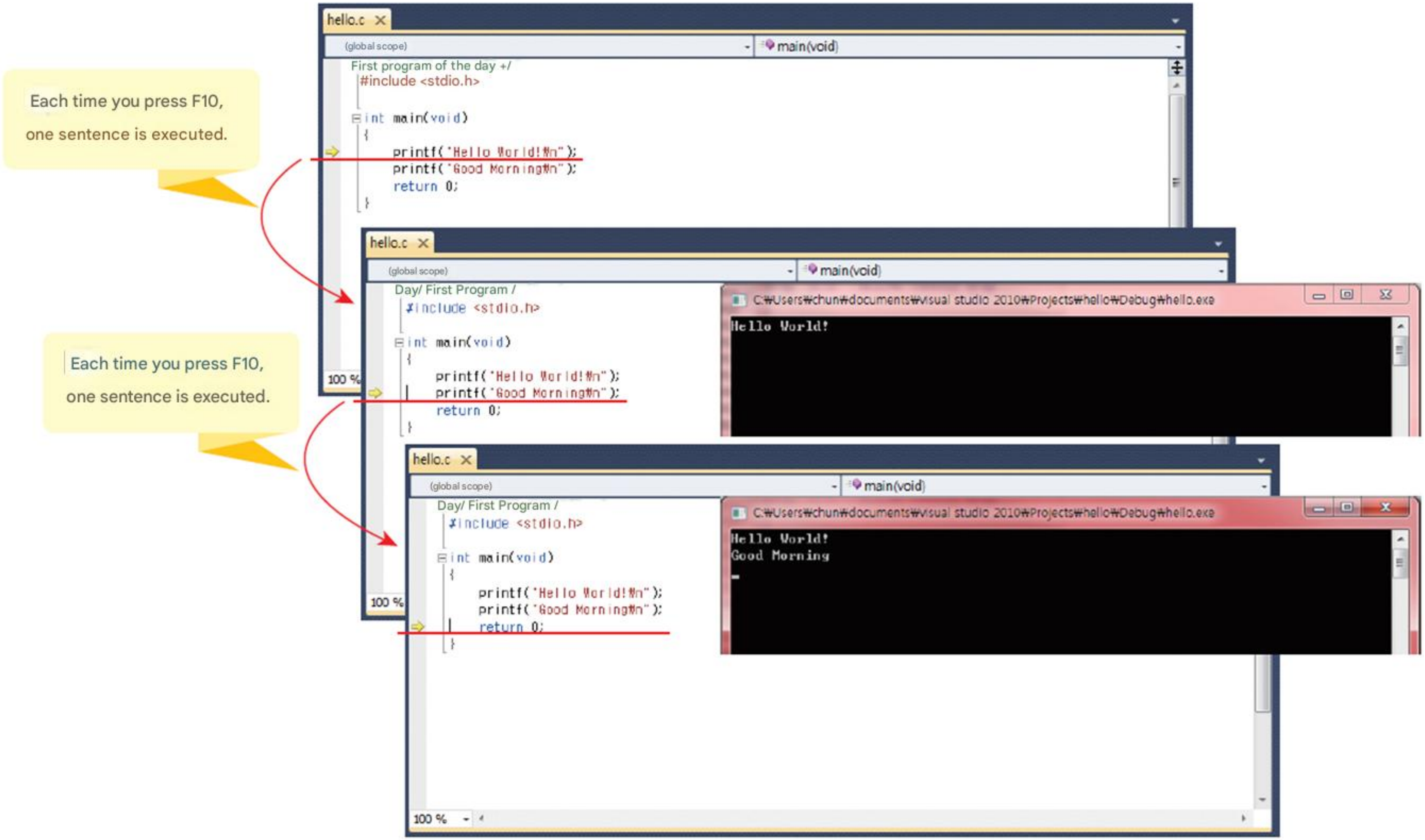
The frame is incorrect, missing, or has no symbols loaded for kernel32.dll]

hello (debugging) - Microsoft Visual Studio
Quick Launch (Ctrl+Q)
File (F)
Edit(E)
View (V)
Project (P)
Build (B)
Debug (D)
Tools (T)
Test (S)
Help (H)
Application Insights
process: [144720] hello.exe
Stack frame: main
hello.c
hello
n(void)
Jang (W)
Graphics (C)
Continue (C) F5
Stop all (K) Ctrl+Alt+Break
Stop debugging (E) Shift+F5
X Separate all (L)
Quit all (M)
Start again (R) Ctrl+Shift+F5
Apply code changes (A) Alt+F10
Performance Profiler (F)... Alt+F2
Connect to Process (P)... Ctrl+Alt+P
Other Debug Targets (H)
profiler
Executing code step by step (I) F11
Procedure unit execution (O) F10
Exit procedure() Shift+F11
Brief inquiry (Q).... Shift+F9
Set/unset breakpoint (G) F9
New breakpoint (B)
Delete all breakpoints (A) Ctrl+Shift+F9
Clear all DataTips (A)
Export DataTips (X)...
Importing DataTips (1)...
Save dump as (V)...
Option(O)...
hello attribute...
diagnostic tool
Diagnosis and Session: 0 sec (178 ms selected)
100
0
CPU (percentage of all processors)
Summary Events Memory Usage CPU Usage
event
Event display (1/1)
memory usage
Create a snapshot
100 %
automatic
name
stack
subcode]
C
Automatic area check 1
Stack
Add to source control
Solution Explorer Team explorer

Defining commands for the debugger

- F5 (Go): Run
- F10 (Step Over): Execute one sentence at a time (functions are also treated as one sentence)
- F11 (Step Into): Execute one statement at a time (step into a function)
- F9 (Breakpoint): Set a breakpoint in the current sentence

Debugger execution process



Each time you press F10, one sentence is executed.

```
First program of the day +/
#include <stdio.h>

int main(void)
{
    printf("Hello World!₩n");
    printf("Good Morning₩n");
    return 0;
}
```

Each time you press F10, one sentence is executed.

```
Day/ First Program /
#include <stdio.h>

int main(void)
{
    printf("Hello World!₩n");
    printf("Good Morning₩n");
    return 0;
}
```

C:₩Users₩chun₩documents₩visual studio 2010₩Projects₩hello₩Debug₩hello.exe

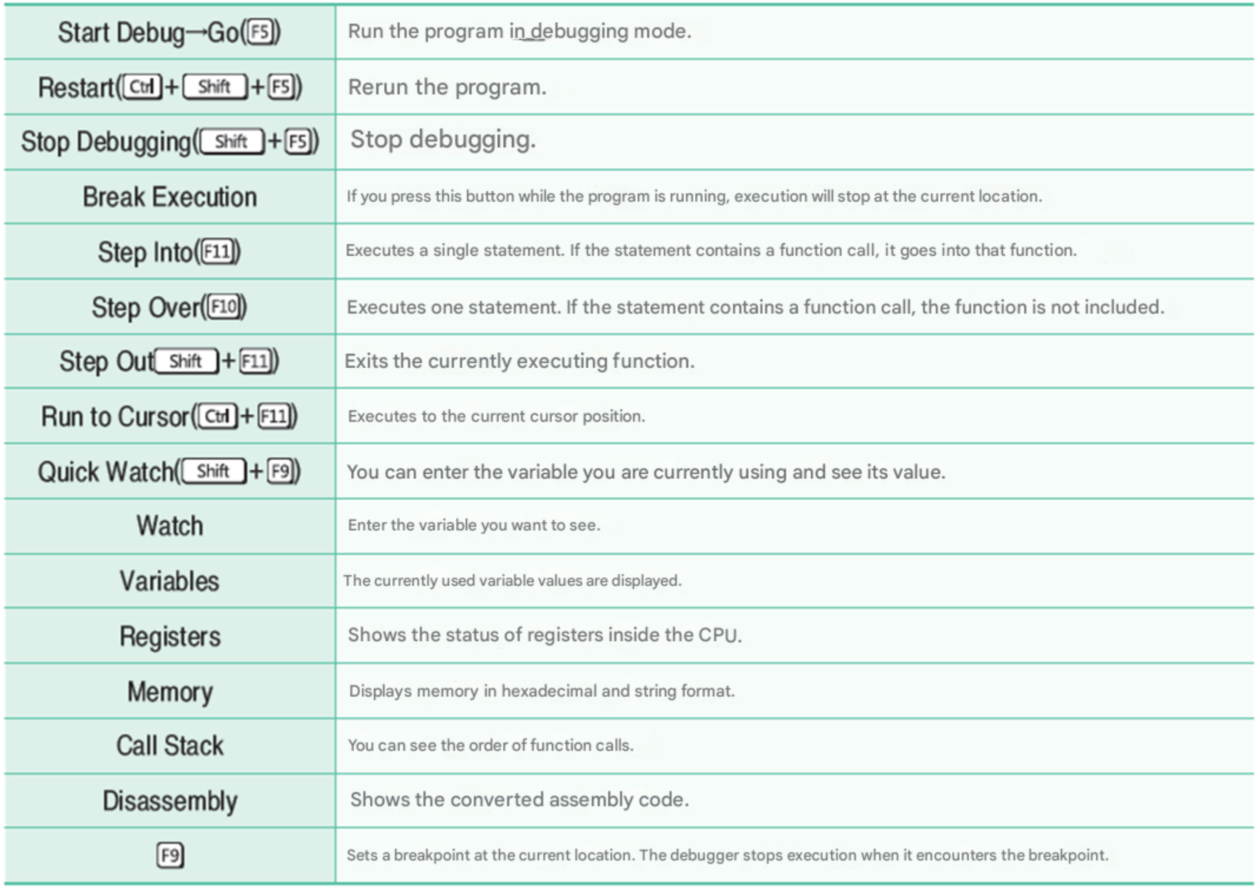
```
Hello World!
```

```
Day/ First Program /
#include <stdio.h>

int main(void)
{
    printf("Hello World!₩n");
    printf("Good Morning₩n");
    return 0;
}
```

C:₩Users₩chun₩documents₩visual studio 2010₩Projects₩hello₩Debug₩hello.exe

```
Hello World!
Good Morning
_
```

Note : Debugger features



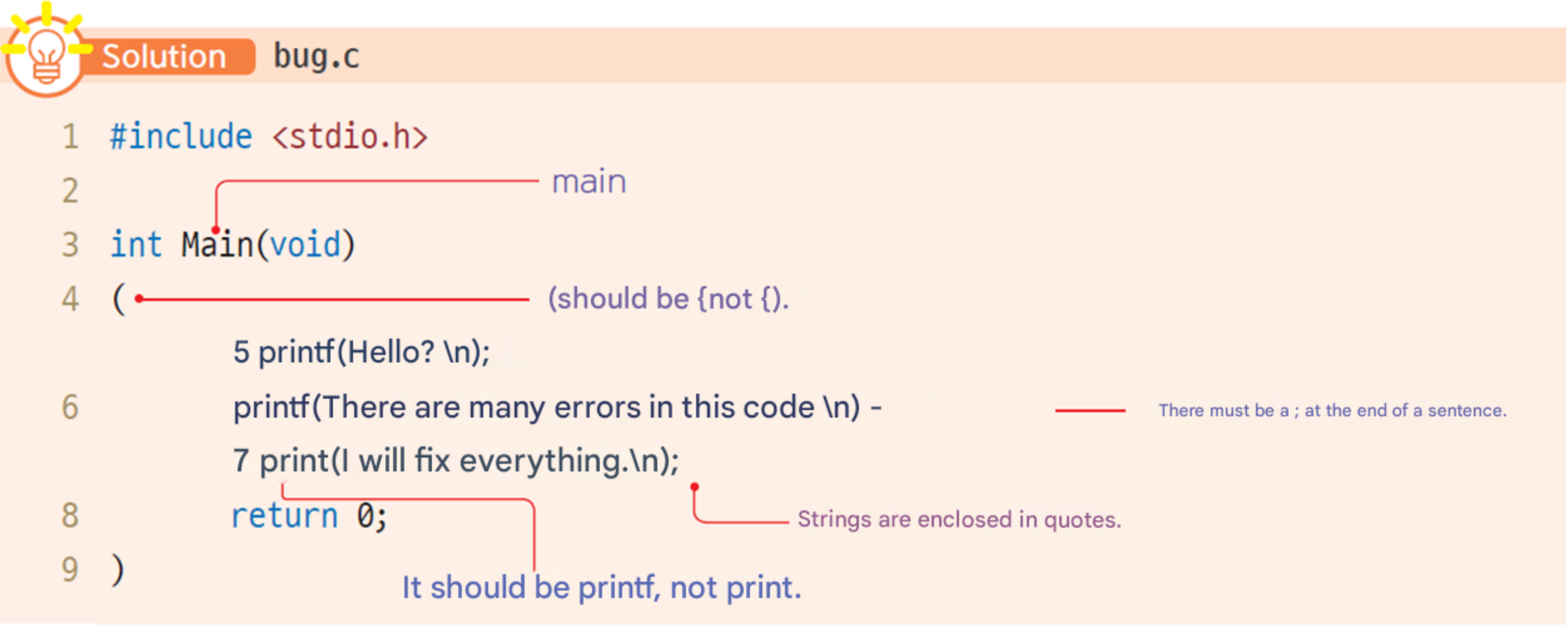
Start Debug→Go(F5)	Run the program in debugging mode.
Restart(Ctrl+Shift+F5)	Rerun the program.
Stop Debugging(Shift+F5)	Stop debugging.
Break Execution	If you press this button while the program is running, execution will stop at the current location.
Step Into(F11)	Executes a single statement. If the statement contains a function call, it goes into that function.
Step Over(F10)	Executes one statement. If the statement contains a function call, the function is not included.
Step Out Shift+F11)	Exits the currently executing function.
Run to Cursor(Ctrl+F11)	Executes to the current cursor position.
Quick Watch(Shift+F9)	You can enter the variable you are currently using and see its value.
Watch	Enter the variable you want to see.
Variables	The currently used variable values are displayed.
Registers	Shows the status of registers inside the CPU.
Memory	Displays memory in hexadecimal and string format.
Call Stack	You can see the order of function calls.
Disassembly	Shows the converted assembly code.
F9	Sets a breakpoint at the current location. The debugger stops execution when it encounters the breakpoint.

Mini Project

- Error Let's fix it !

```
#include < stdio.h >

int Main(void)
(
    printf ( hello ?₩n);
    printf ( There are many errors in this code ₩n)
    print( I will fix everything .₩n);
   return 0;
)
```

Mini Project



Solution bug.c

```
#include <stdio.h>

int Main(void)
(
        printf(Hello? \n);
        printf(There are many errors in this code \n) -
        print(I will fix everything.\n);
        return 0;
)
```

main

(should be {not {).

There must be a ; at the end of a sentence.

Strings are enclosed in quotes.

It should be printf, not print.

Mini Project

- Program with fixed errors

```
#include < stdio.h >

int main( void )
{
        printf ( " Hello ? \n" );
         printf ( " There are many errors in this code \n" );
         printf ( " I will fix it all .\n" );
         return 0;
}
```

Q & A