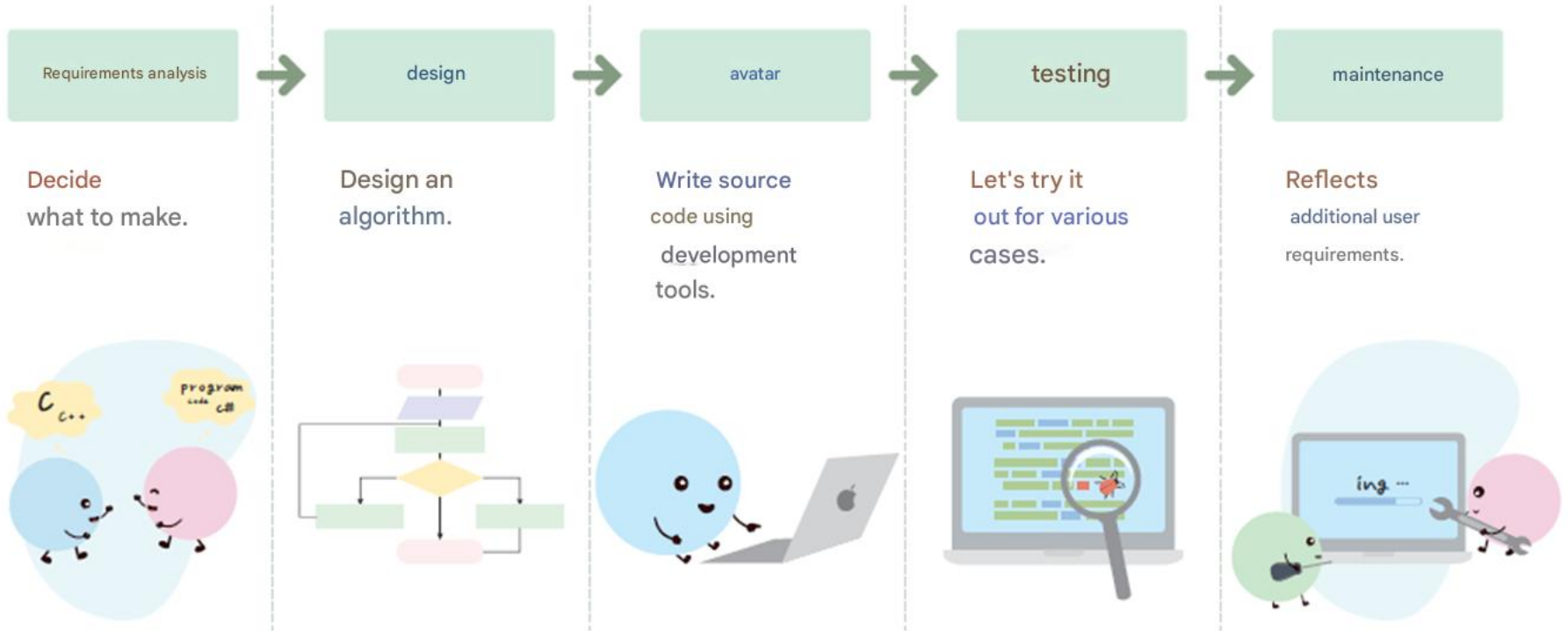


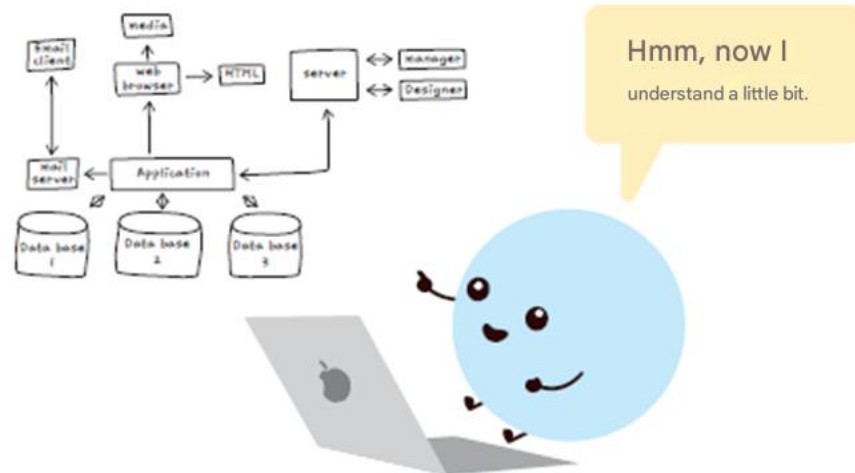
Ch.2 Program Writing Process

Program Development process



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 Source 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



editor

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

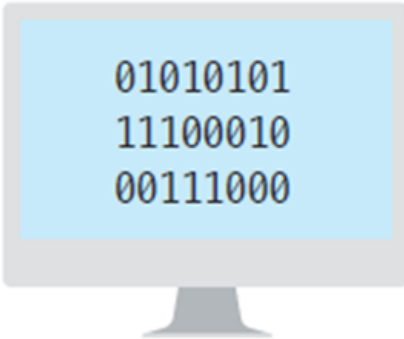
Compile

- 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;
}
```



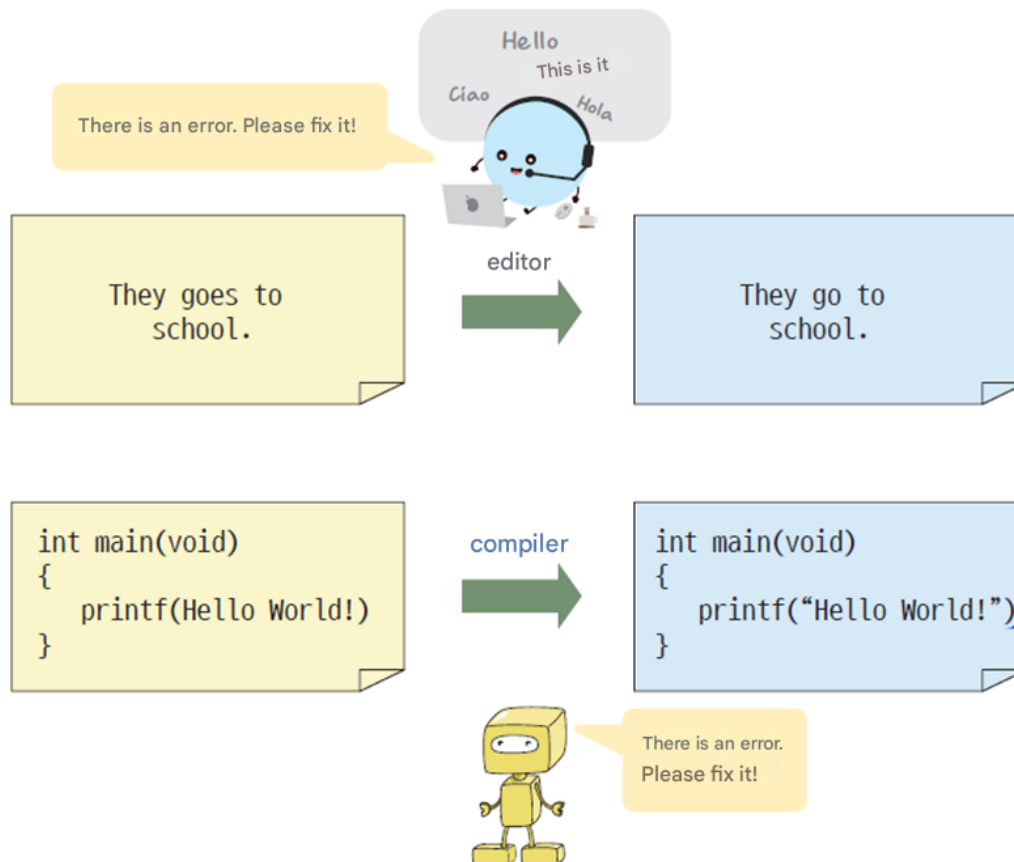
compiler



```
01010101
11100010
00111000
```

Compile error

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



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

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

```
int main(void)
```

```
{
```

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

```
    return 0;
```

```
}
```

source file



```
010101000001111101010
```

```
101010100000111110101
```

```
010101010000011111010
```

```
101010101000001111101
```

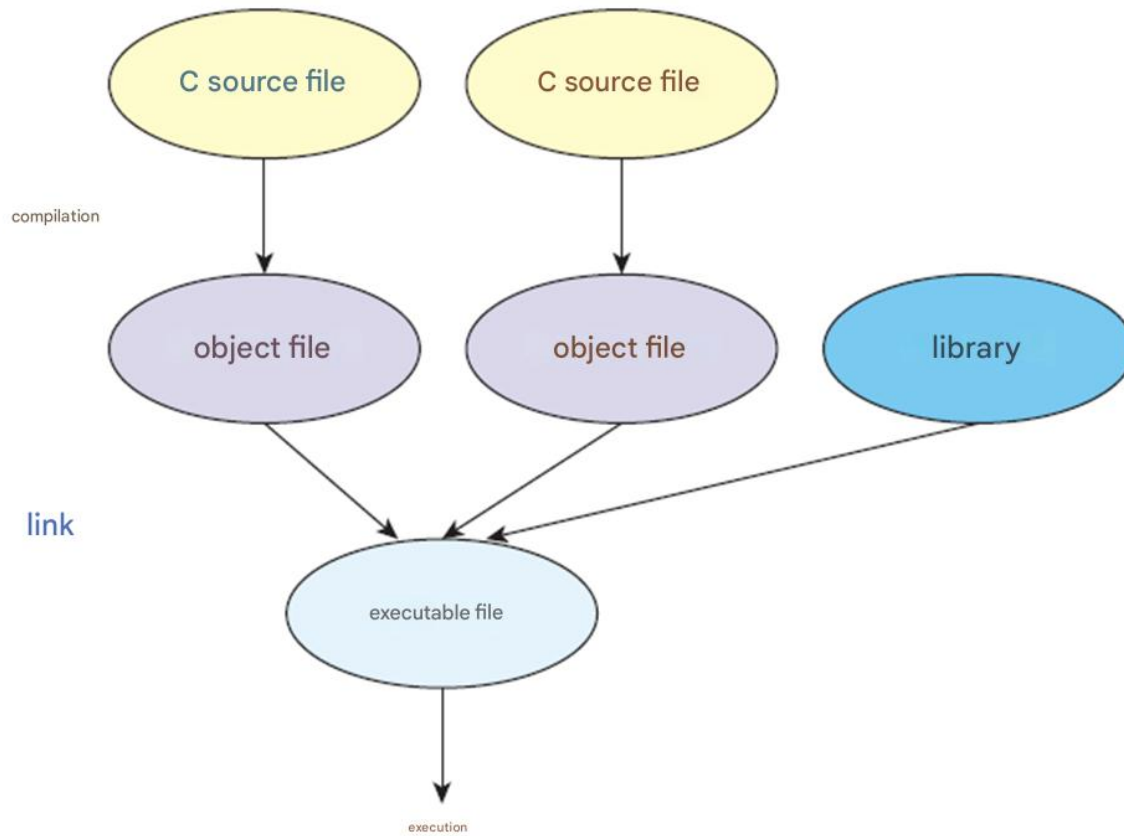
```
010101010100000111110
```

```
101001010100000111110
```

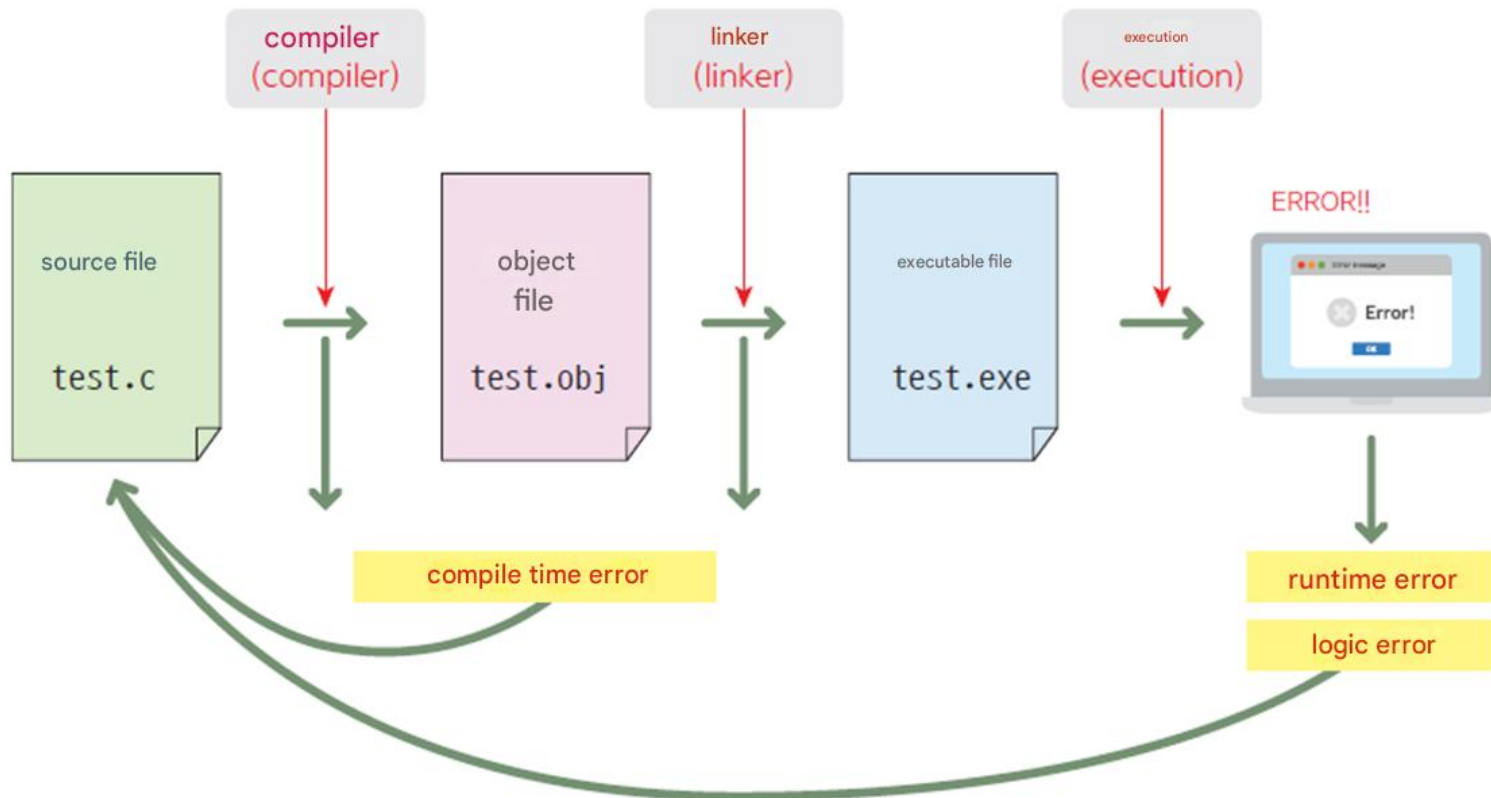
```
10101...
```

object file

Link



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 .



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.



9/2
9/9

0800 Action started
1000 stopped - action ✓
1300 (1300) HP - MC 2.13047695
PRO - 2.13047695
correct 2.13047695
Relays 6-2 in 0.24 full speed speed test
in relay 11.00 test

1100 Started Cassio Tapes (Sine check)
1500 Started Quality Audio Test

1545

Relay 70 Panel F
(moth in relay)

First actual case of bug being found.
1700 Action started.
1700 closed down.

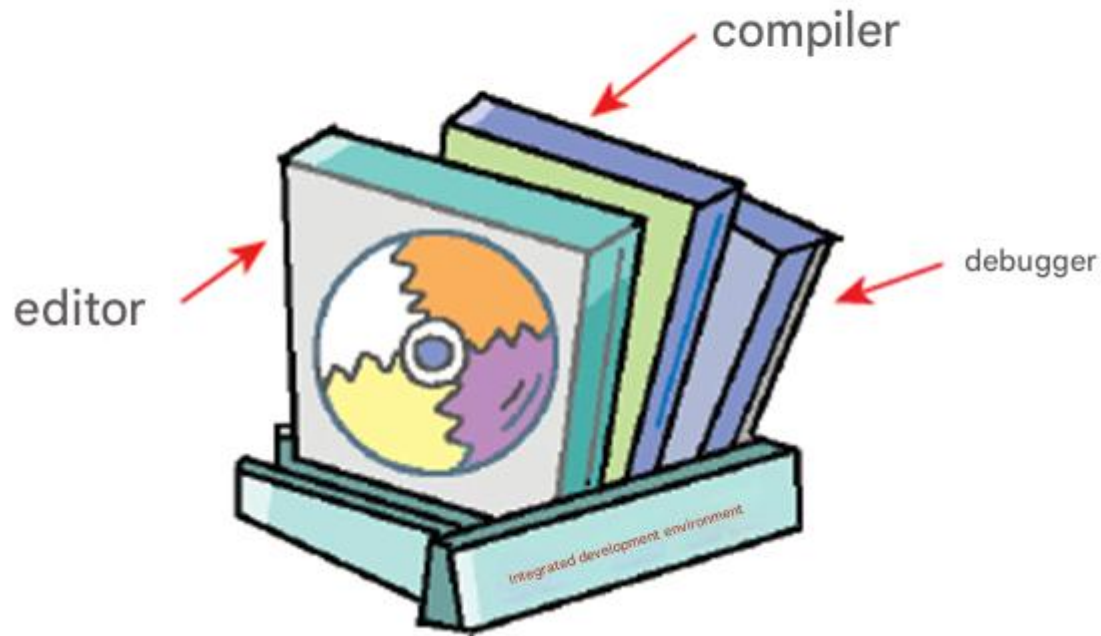
Maintenance of software

- 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



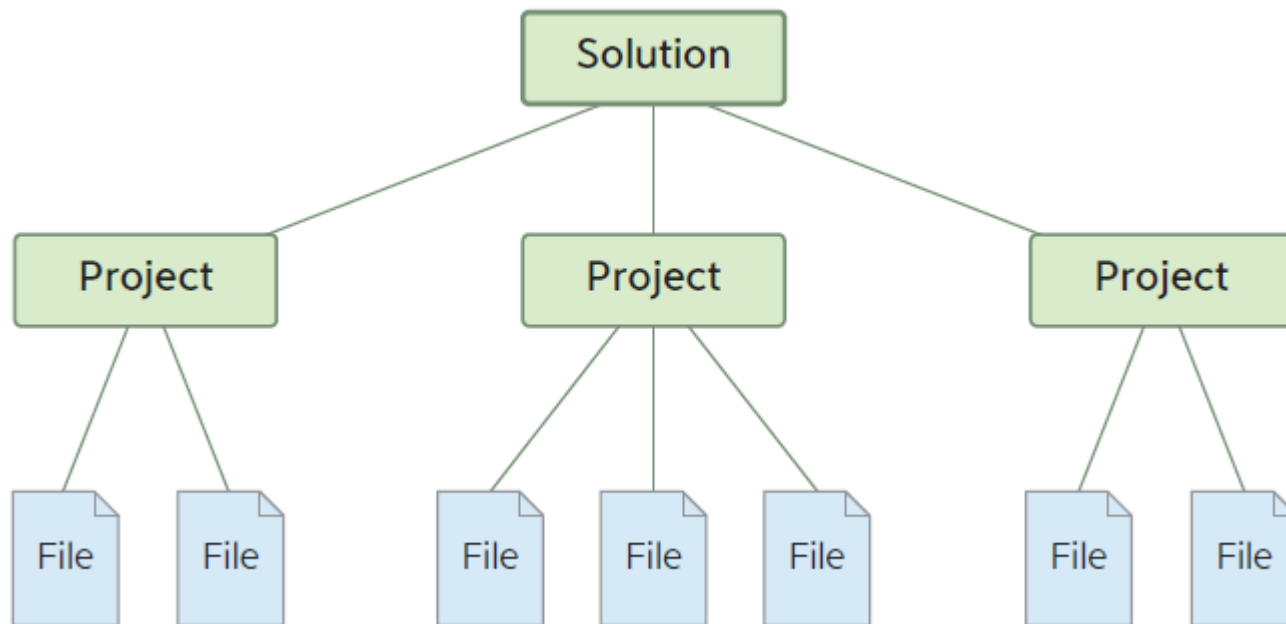
Integrated development environment examples

- 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 .



Description of the first program

hello.c

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

```
int main (void)
```

```
{
```

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

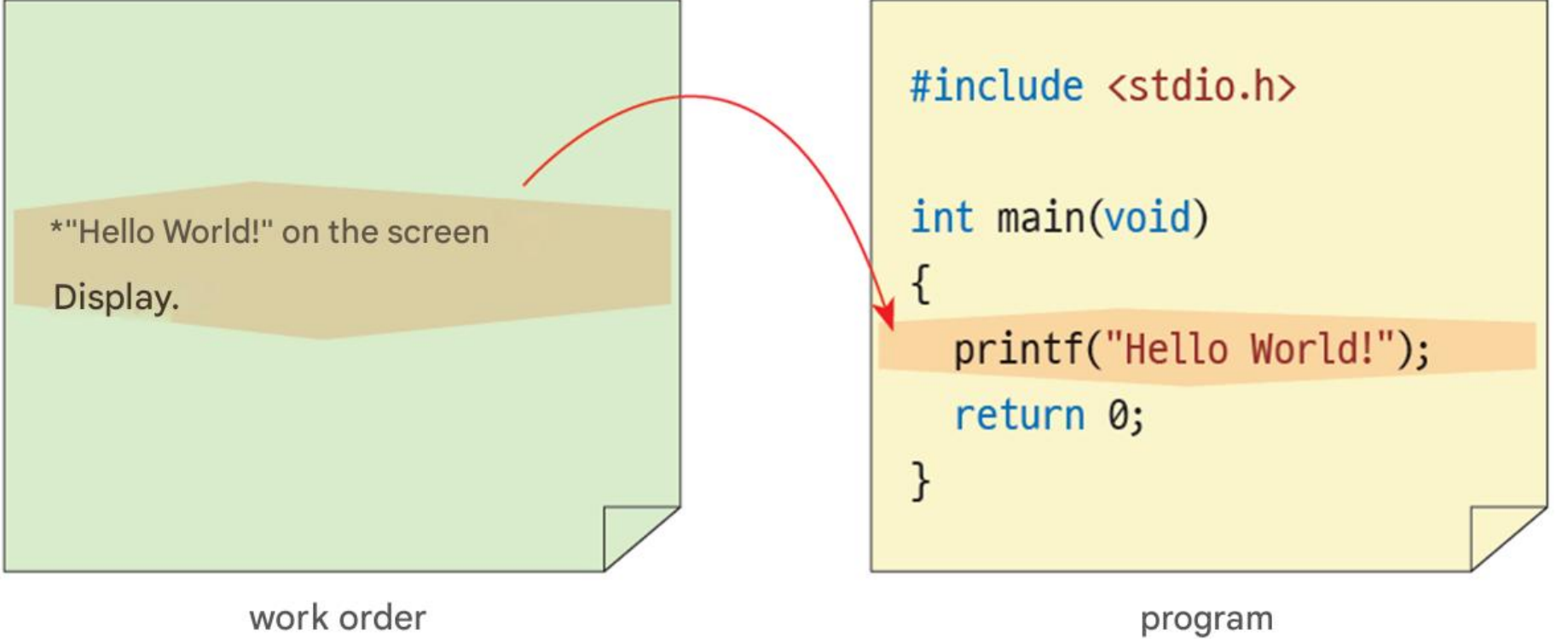
```
    return 0;
```

```
}
```



Hello World!

Program == Work Instructions



*"Hello World!" on the screen
Display.

work order

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

```
int main(void)
```

```
{
```

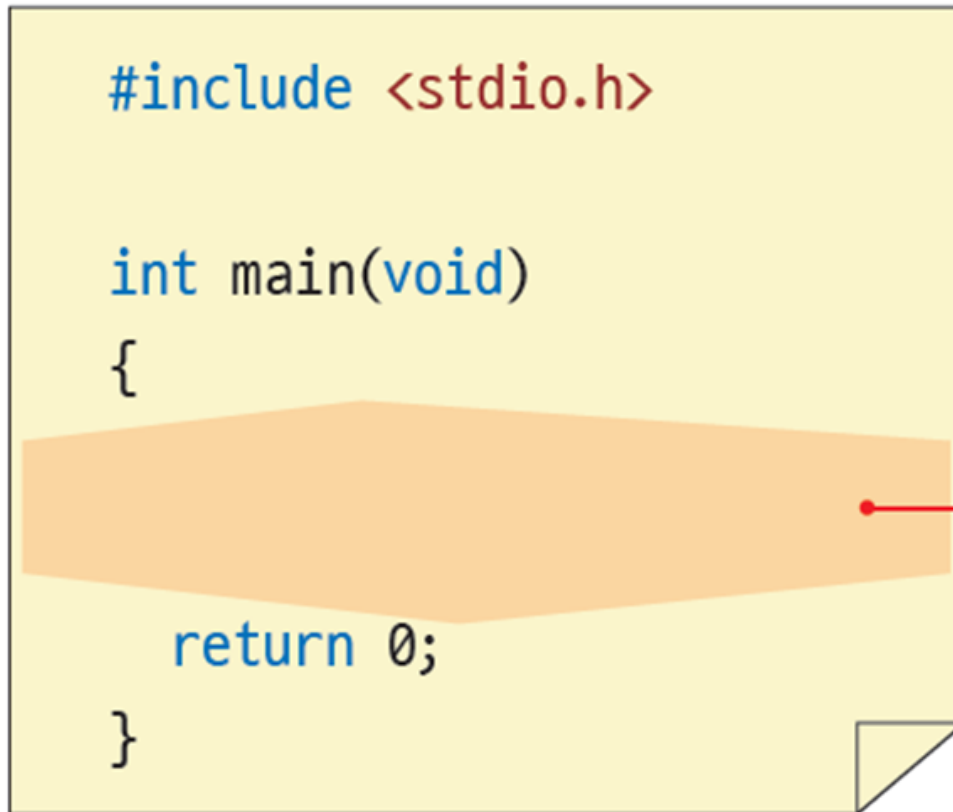
```
    printf("Hello World!");
```

```
    return 0;
```

```
}
```

program

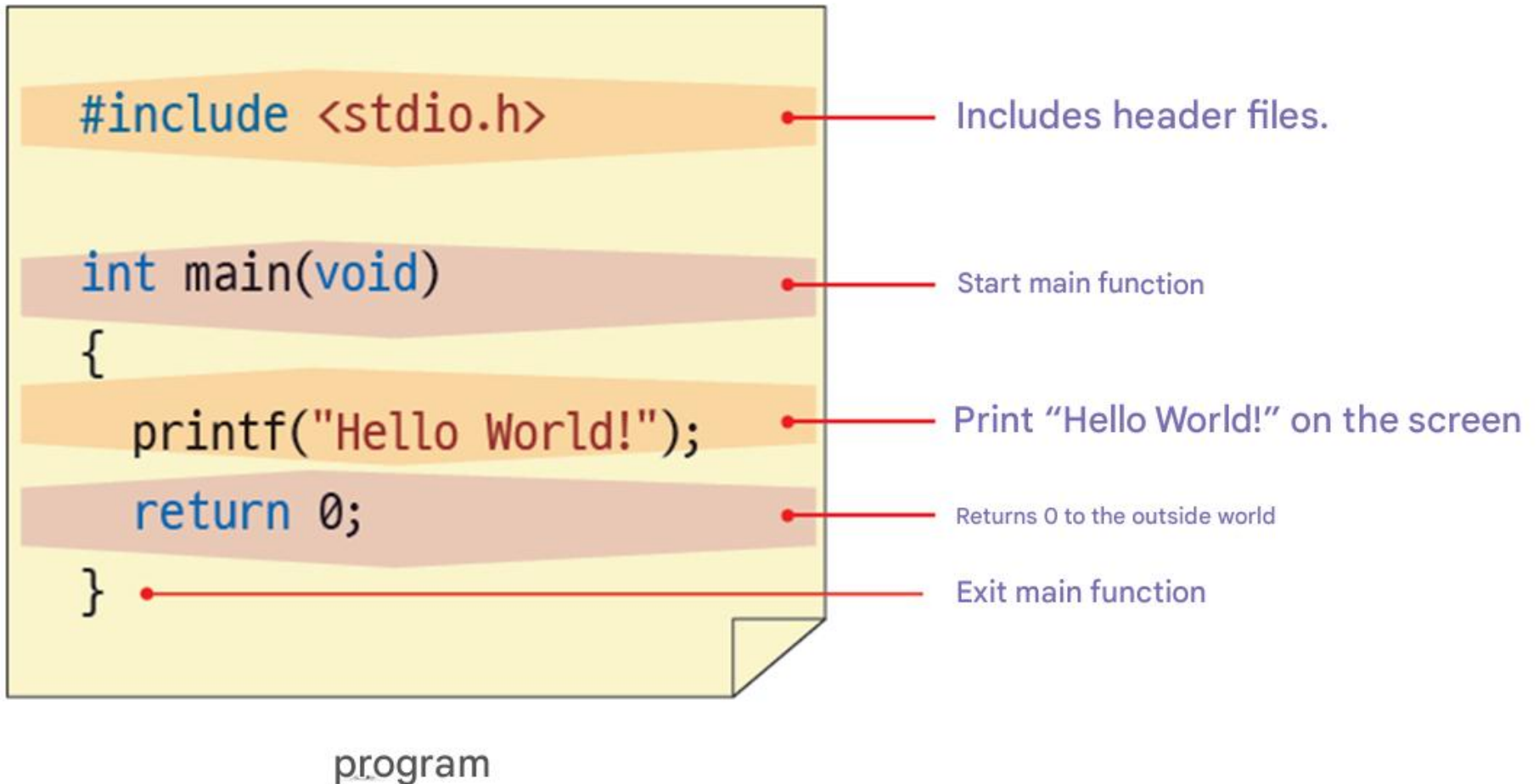
Where to write down your work



Write the sentence that
performs the desired task
here.

program

Brief source description



Including header files

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

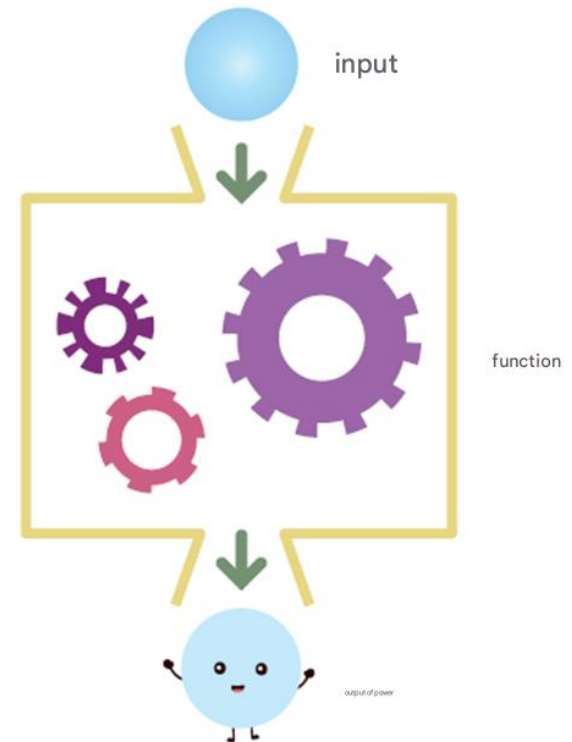
- Caution!: Preprocessor directive statements must not end with a semicolon (;).

`#include <stdio.h>`

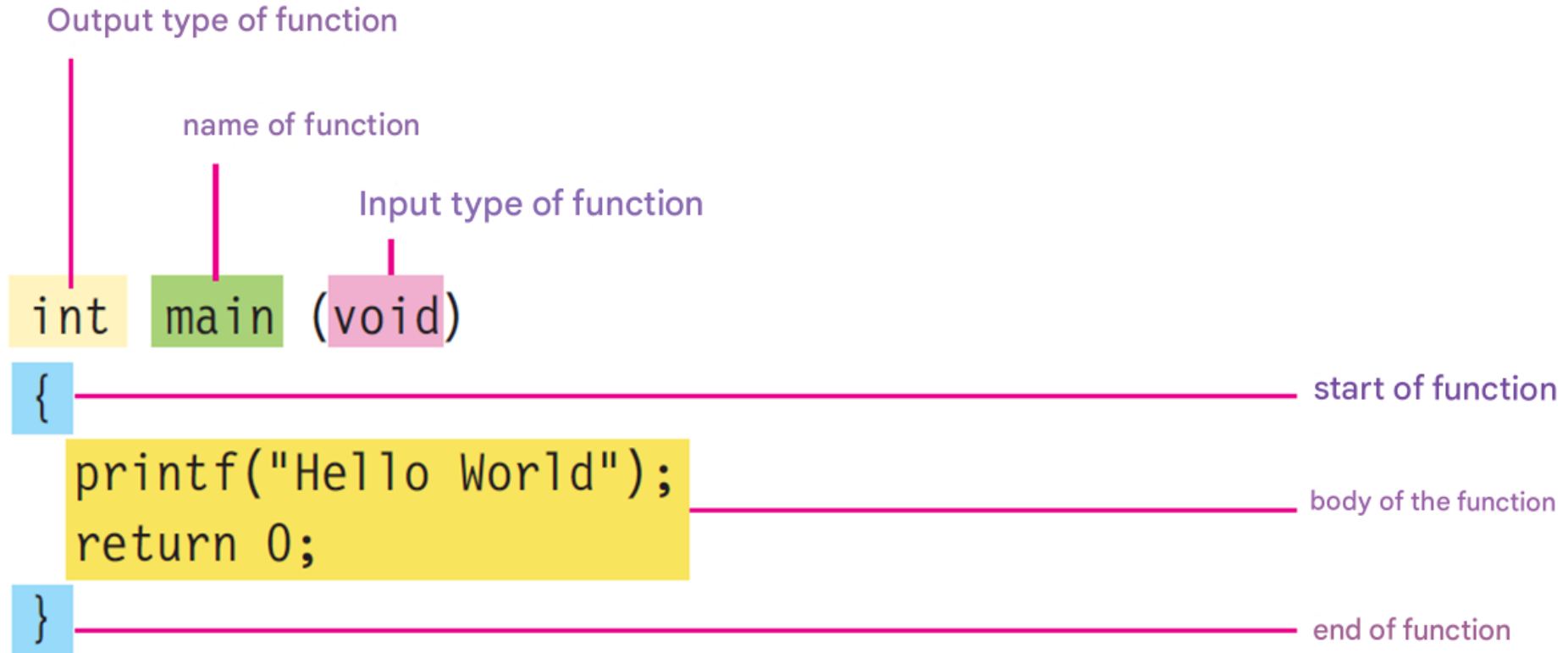
- Header file: The compiler
A file containing the information you need
- stdio.h: standard input output header file

Function

- Function : A standalone piece of code written to perform a specific task.
- (Reference) Mathematical function
$$y = x^2 + 1$$
- program = set of functions

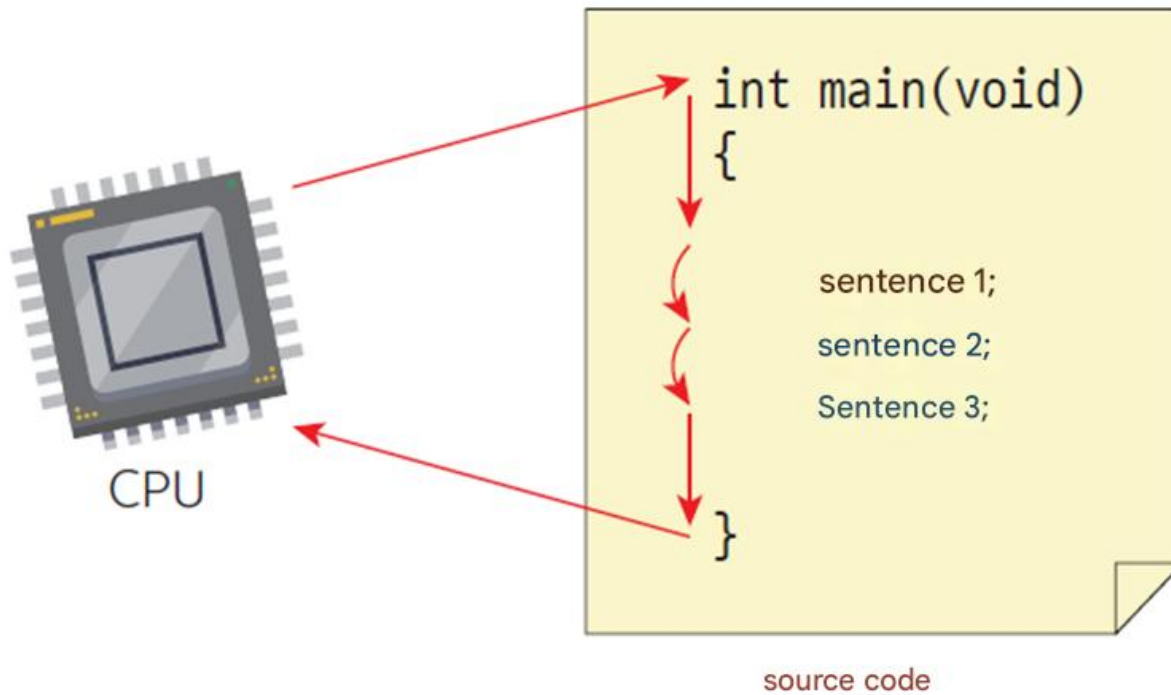


A brief description of the function

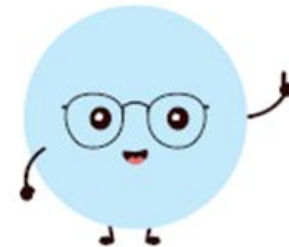


Sentence (Imperative)

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



Statements in source code are basically executed sequentially.



printf() call

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

printf("Hello World!");

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



Hello World!

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 .



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;
```

```
}
```

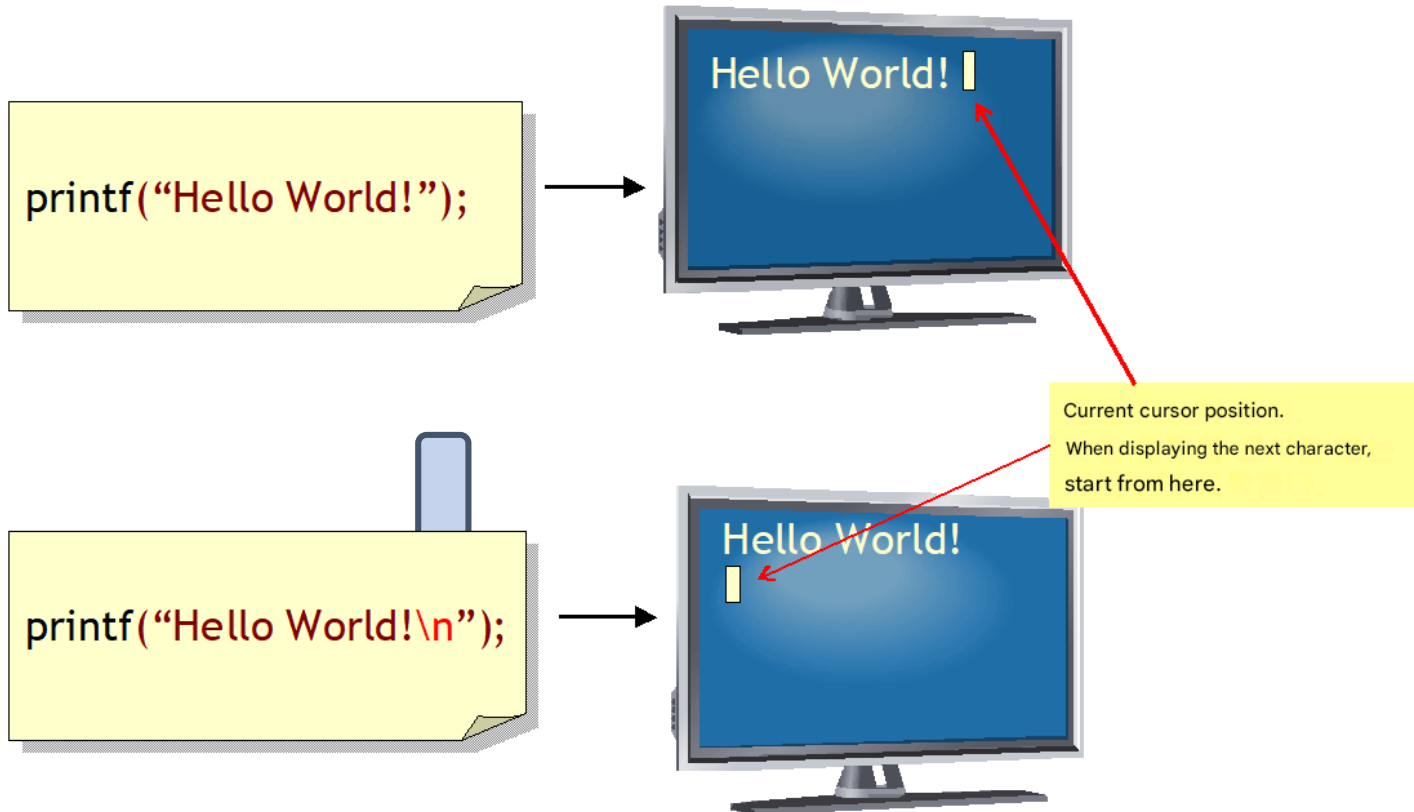
2 of Sentences are executed sequentially.



Hello World! Kim ChulSoo

New line character

- The new line character `\n` moves the cursor to the next line on the screen



Changed program

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

```
#include < stdio.h >

int main( void )
{

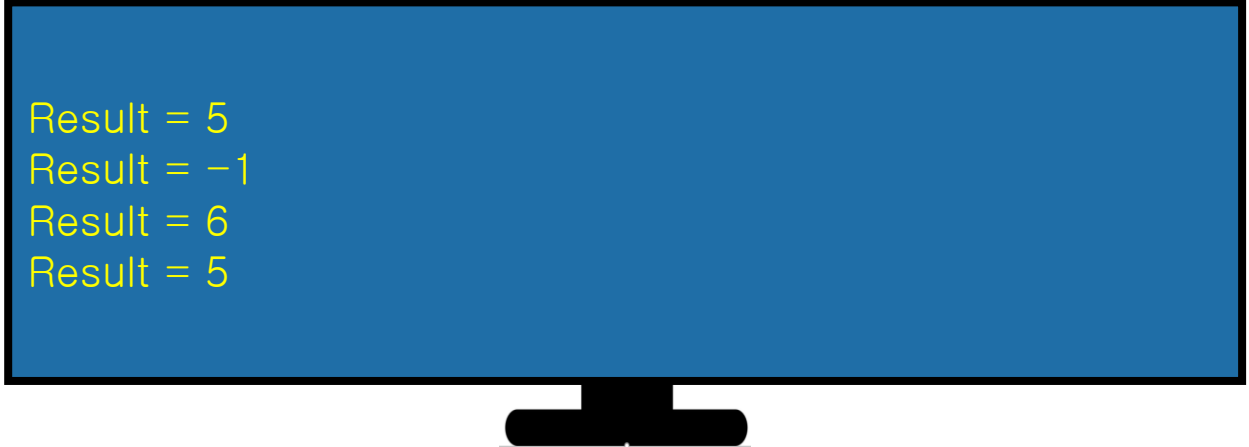
    printf ( "Hello World!\n" );
    printf ( "Kim ChulSoo \n" );

    return 0;
}
```



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;
}
```

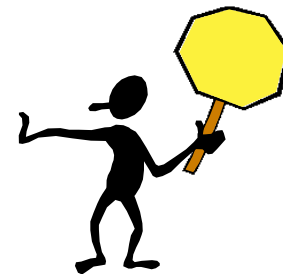
Error Fixing and Debugging

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

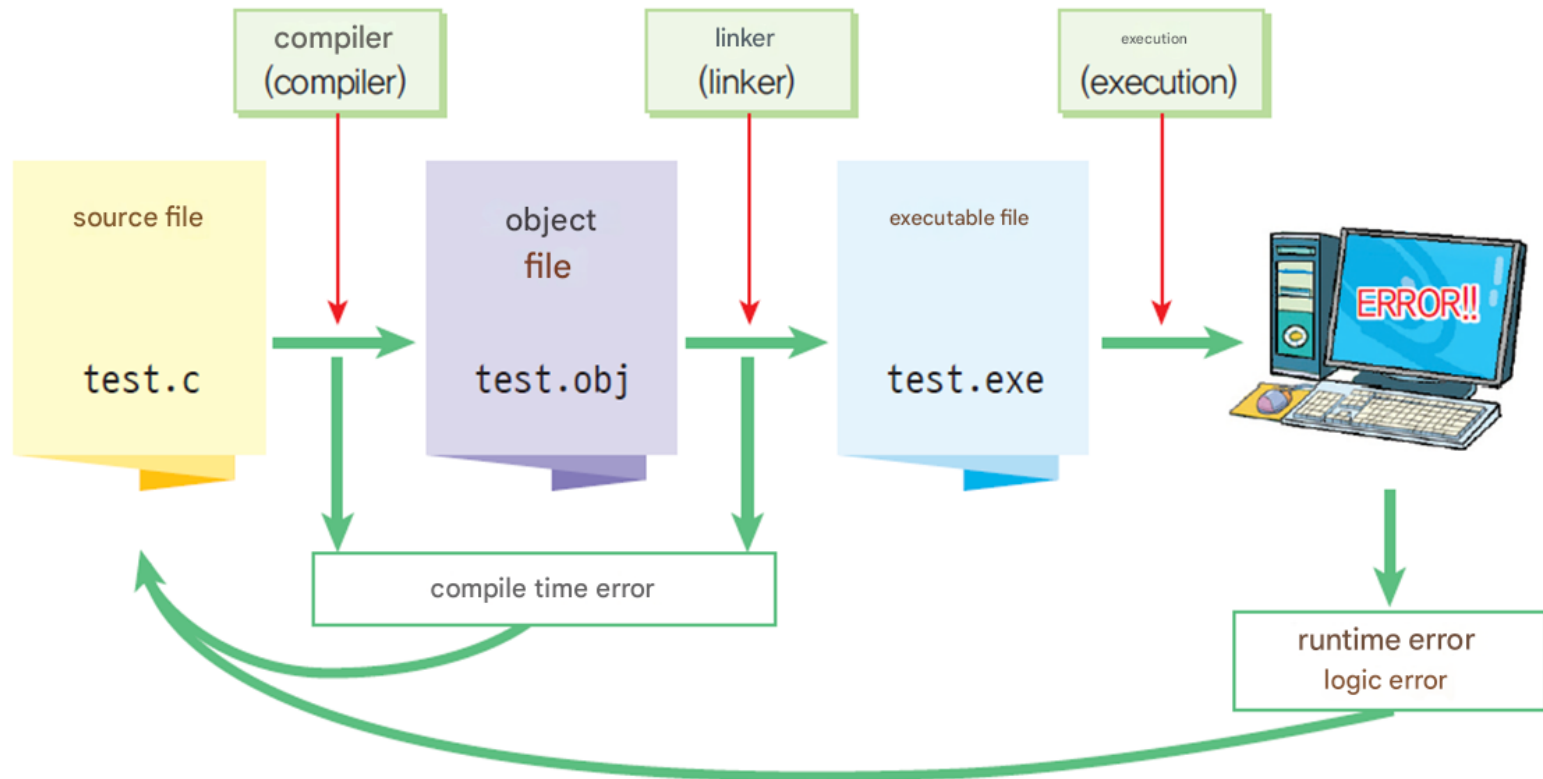


Types of errors

- 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



Error #1

The screenshot shows the Microsoft Visual Studio IDE with a C++ project named 'add'. The main window displays the source file 'error1.c' with the following code:

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

A red dot on line 3, column 24, indicates a missing semicolon. A yellow box with the text "; is omitted." points to this location. Another yellow box with the text "This means that you missed the ; before retrun." points to the error list entry.

The error list at the bottom shows the following entry:

code	description	project	file	line
C2143	" is required.	add	error1.c	6
	Syntax error: missing ';' before 'return'.	add	error1.c	6

Annotations highlight the error list entry and the line number in the error list.

Source file where errors were found

error found line number

Error #2

The screenshot shows the Microsoft Visual Studio IDE with a C program named `error3.c` open. The code is as follows:

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

A red box highlights the `printf(Hello World!);` line, and a yellow callout points to it with the text: "When displaying a string, I missed the \" and \".".

The error list at the bottom shows the following errors:

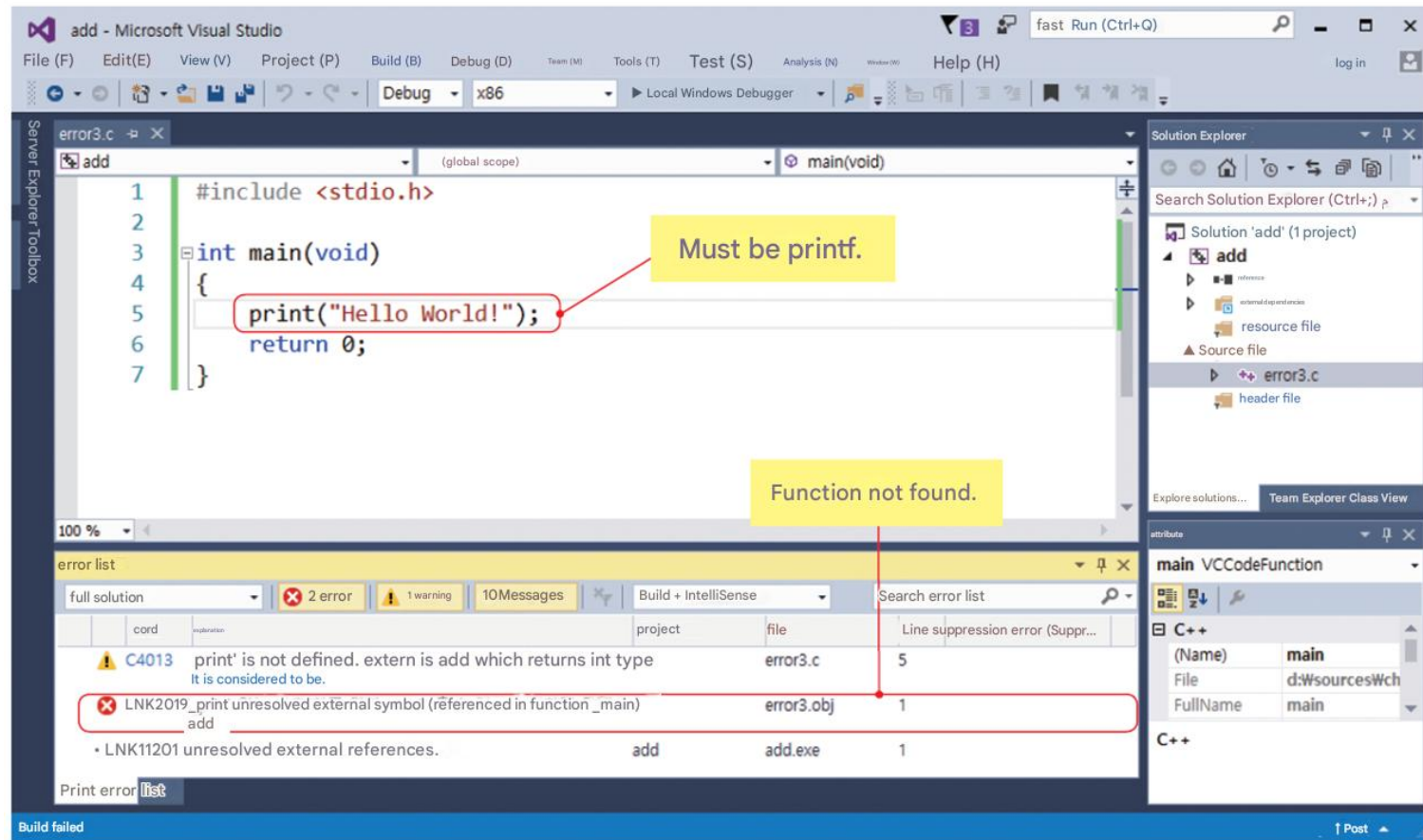
Code	Explanation	Project	File	Line	Suppression
abc	Identifier "Hello" is not defined.	add	error3.c	5	
abc	'/' is required.	add	error3.c	5	
C2065	'Hello': undeclared identifier.	add	error3.c	5	
C4047	'function': 'const char *const ' has different level of indirection than 'int' add Yes.	add	error3.c	5	

The Solution Explorer on the right shows the project structure:

- Solution 'add' (1 project)
 - add
 - reference
 - external dependencies
 - resource file
 - Source file
 - error3.c
 - header file

The status bar at the bottom shows: Rows: 6, Column: 14, Characters: 11, INS.

Error #3



Logical error

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




Program with logic errors

```
#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

I think this part is
suspicious...



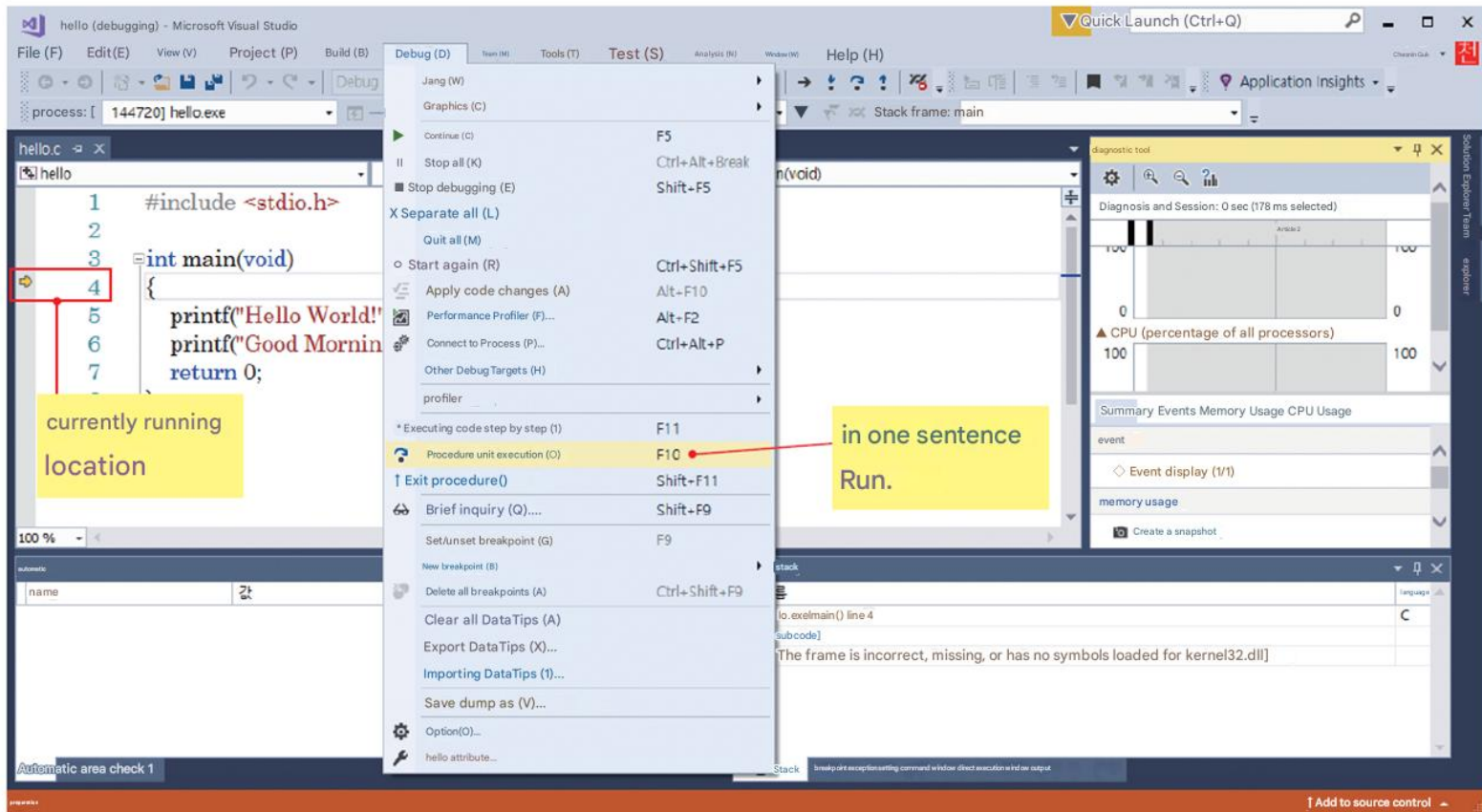
Program execution result

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



Debugger

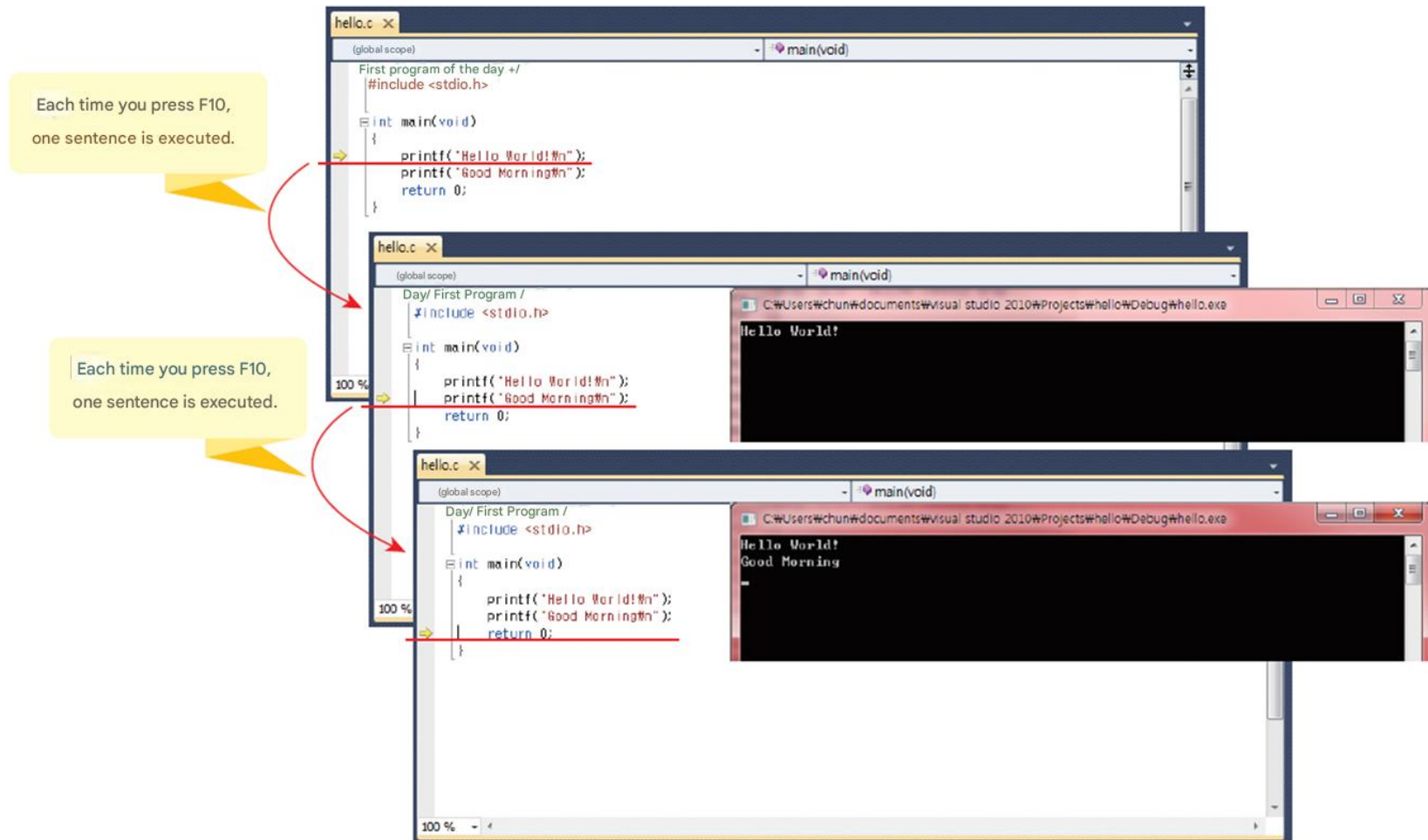
- A tool to find the cause of an error by executing a program one sentence at a time.



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



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

```
1 #include <stdio.h>
2
3 int Main(void)
4 (
5     printf(Hello? \n);
6     printf(There are many errors in this code \n) -
7     print(I will fix everything.\n);
8     return 0;
9 )
```

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

