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.
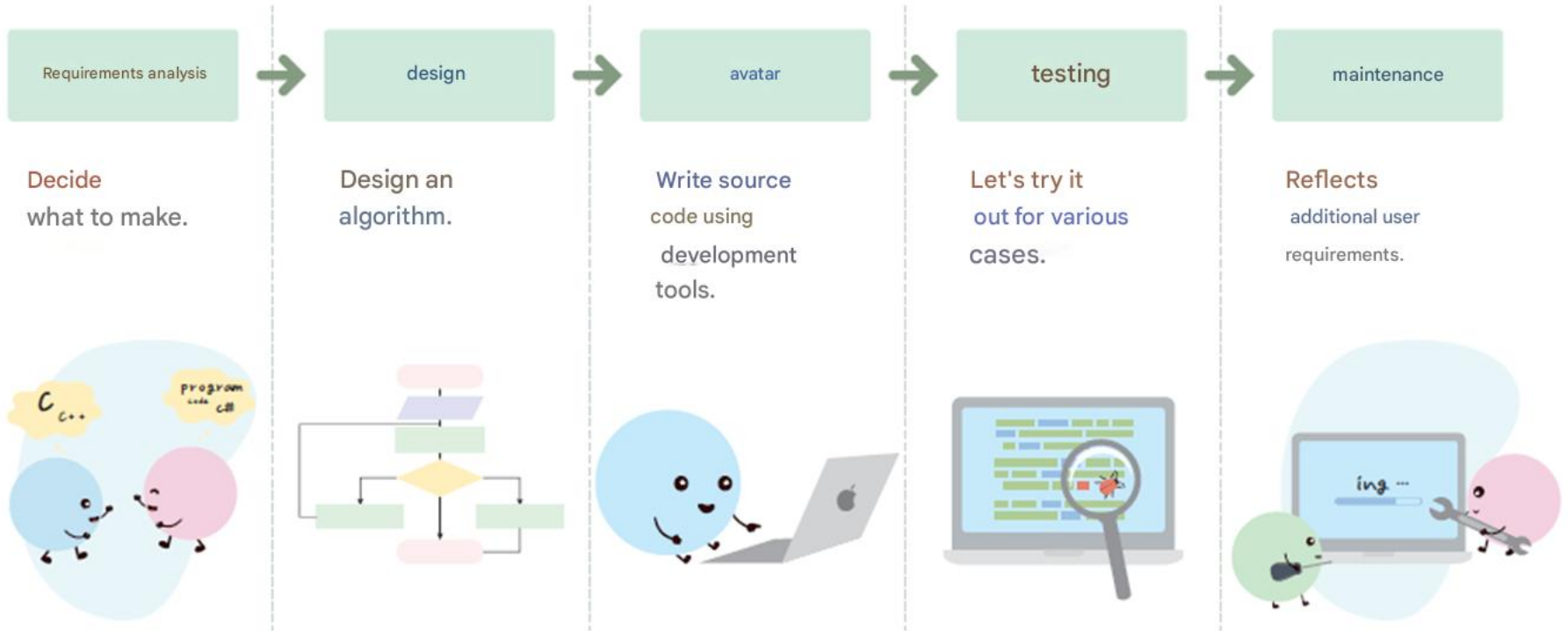
Design

- 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.
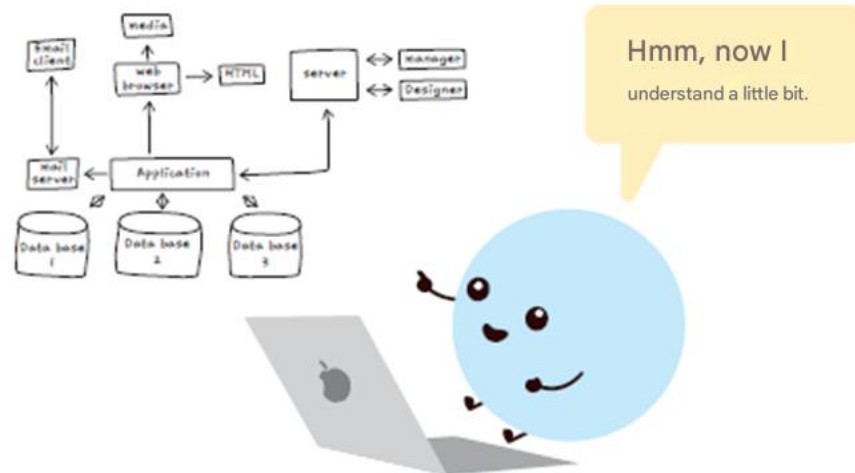
Write Souce code

- 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

editor

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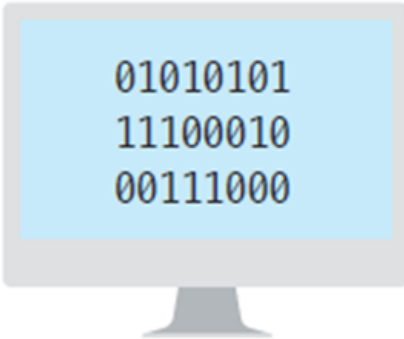
Compile

- 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;
}
```
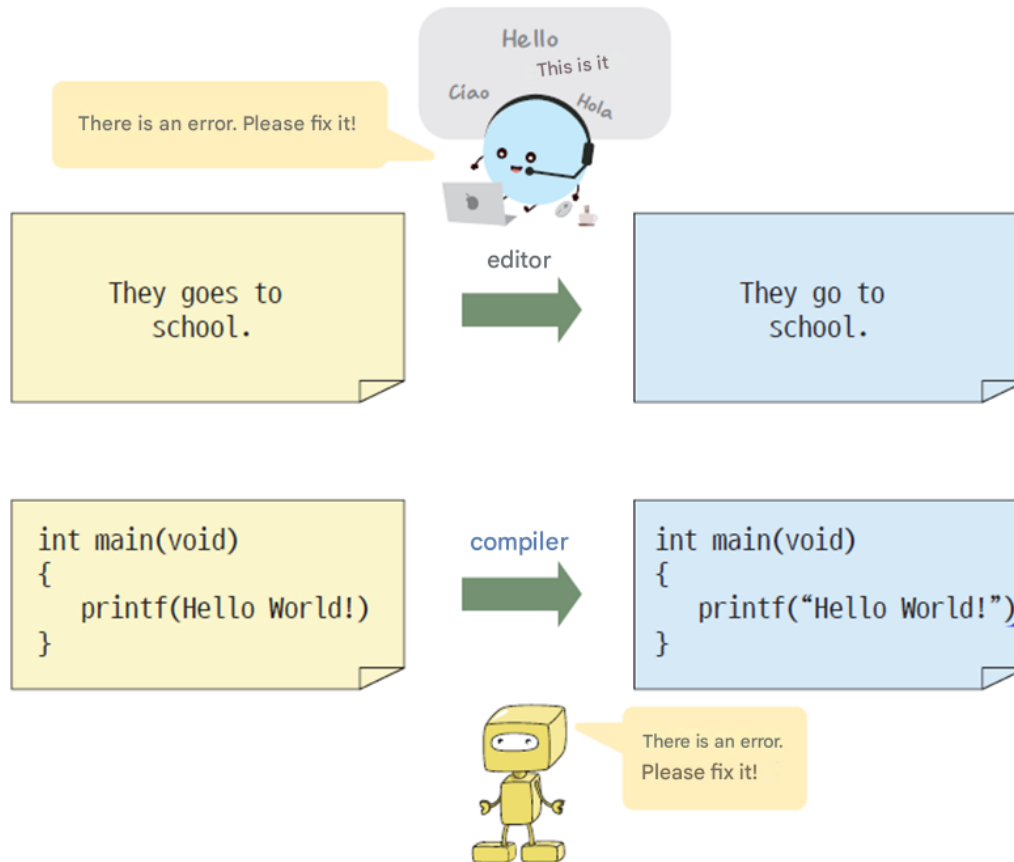
compiler

```
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
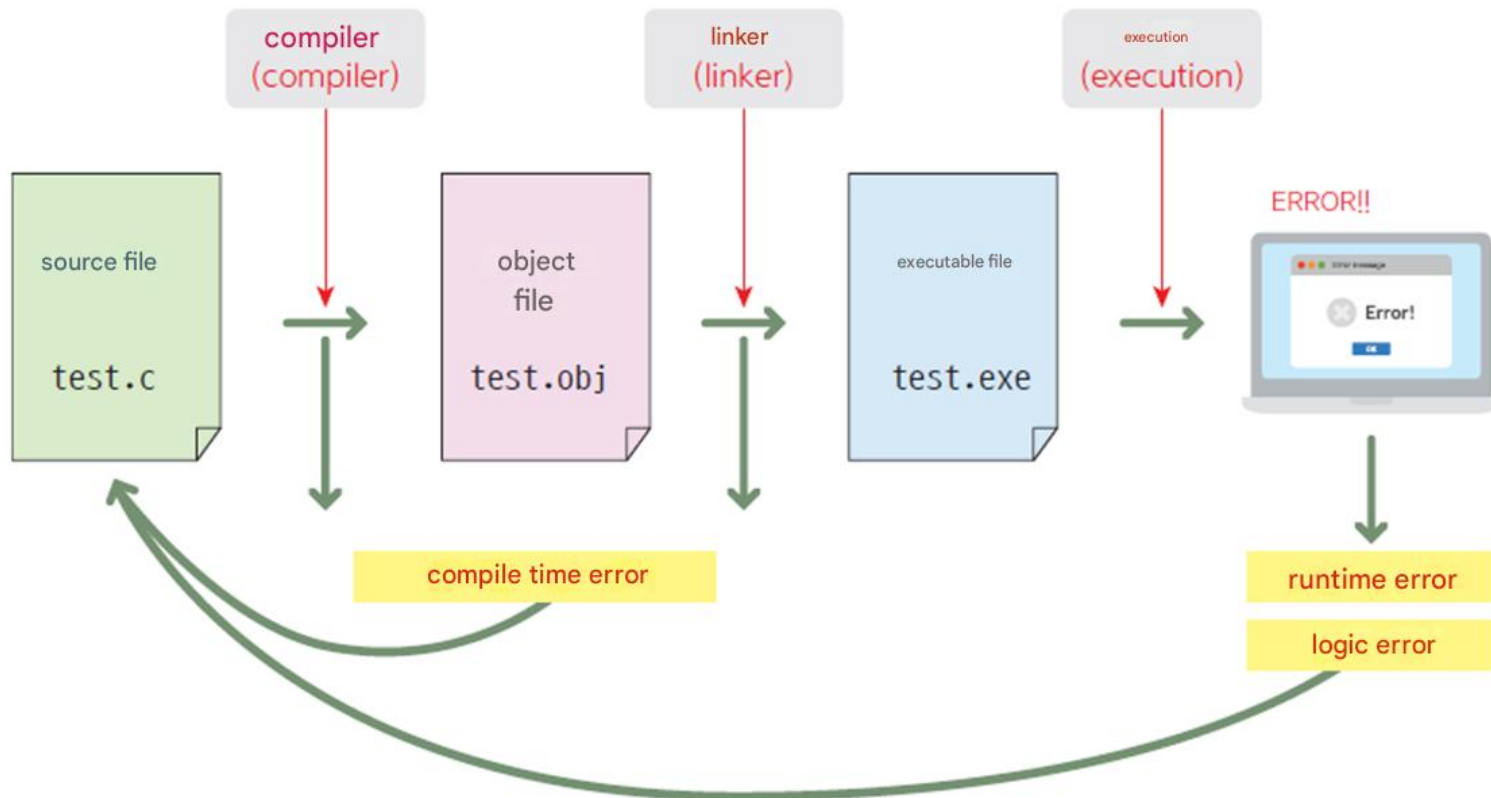- The program that performs linking is called *a linker*

Object file

source file

object file

Link
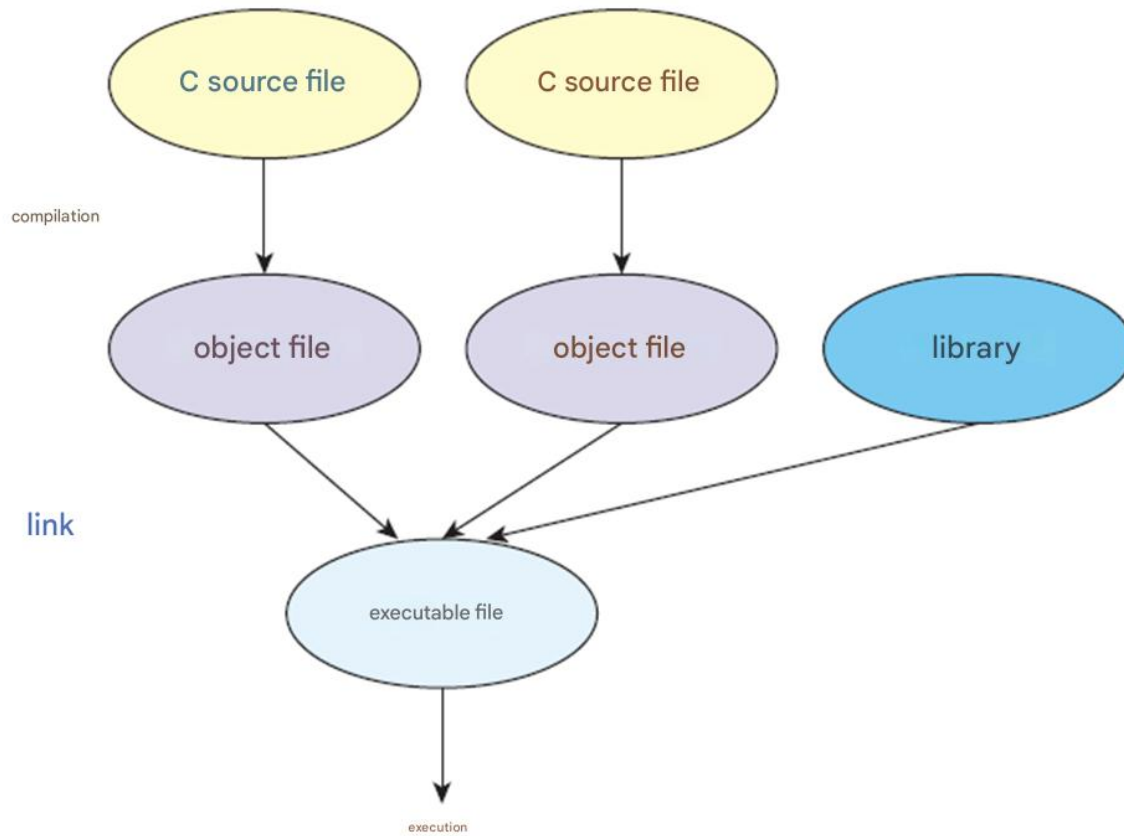
Running and Debugging

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.

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 flying 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.
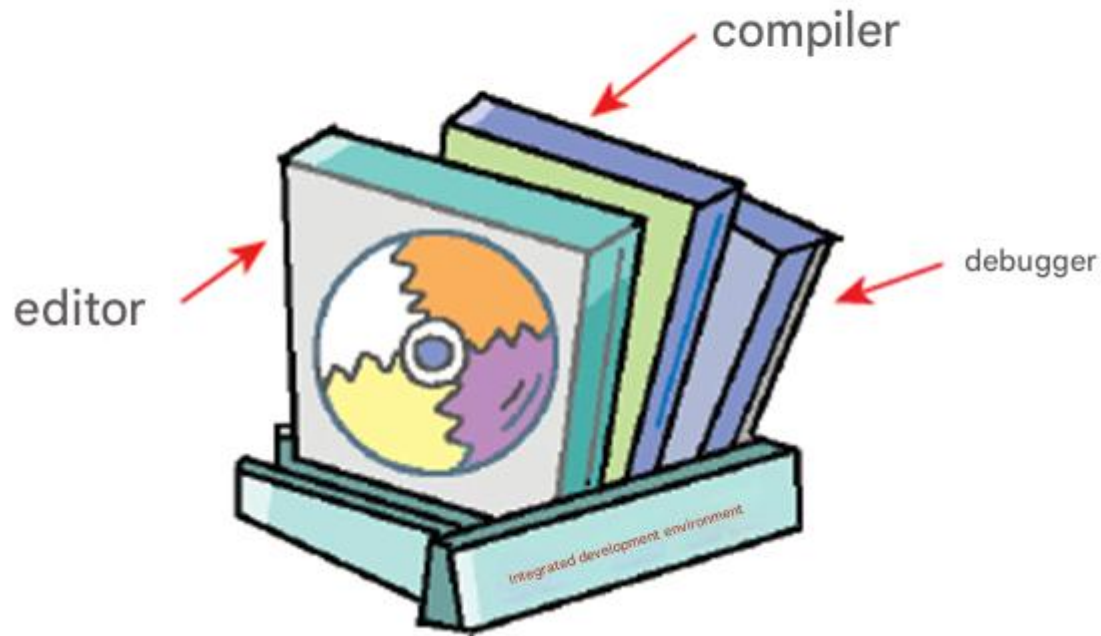
Maintenance of software

- Why Software Maintenance Is Necessary
  - Because bugs may remain even after debugging.
  - 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

Description of the first program

main.c

```c
#include <stdio.h>

int main (void)
{

    printf( "Hello World!" ) ;

    return 0;

}
```
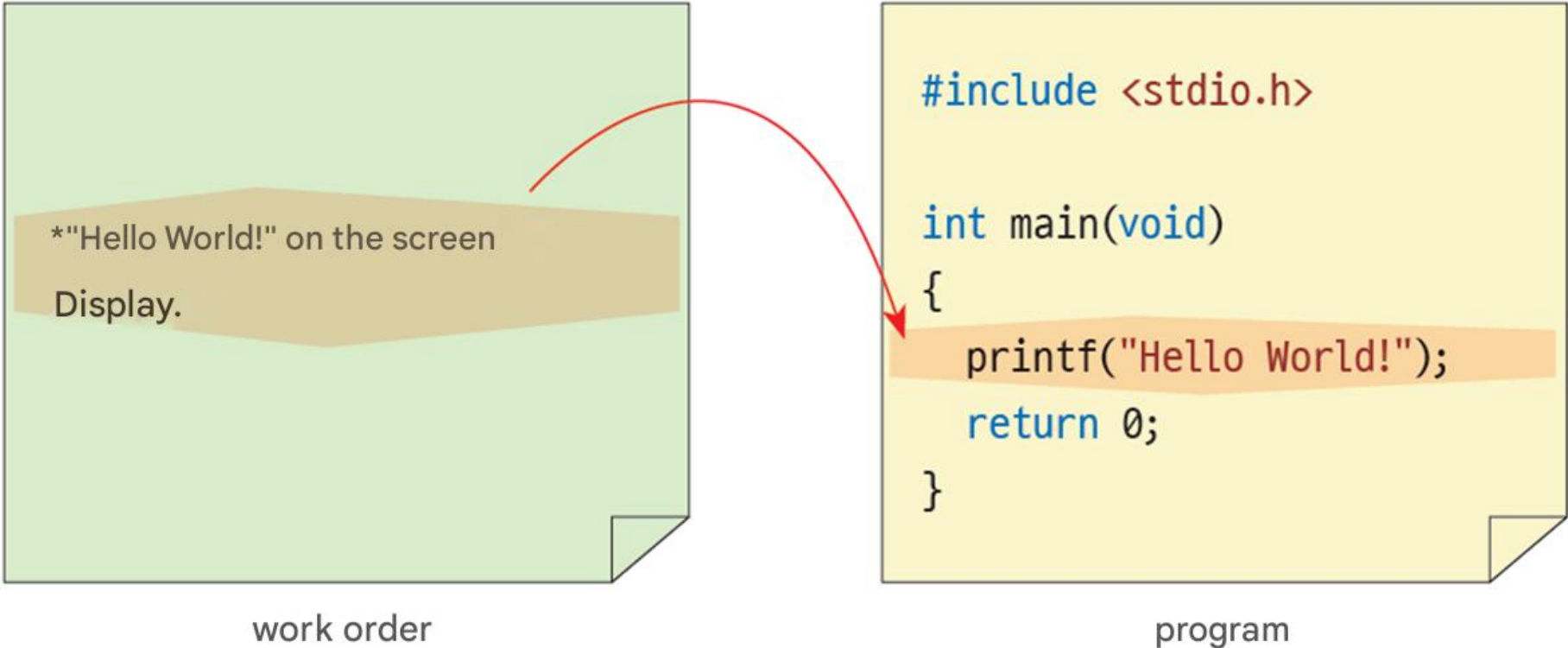
Hello World!

Program == Work Instructions


```
*"Hello World!" on the screen
Display.
```

work order

```c
#include <stdio.h>

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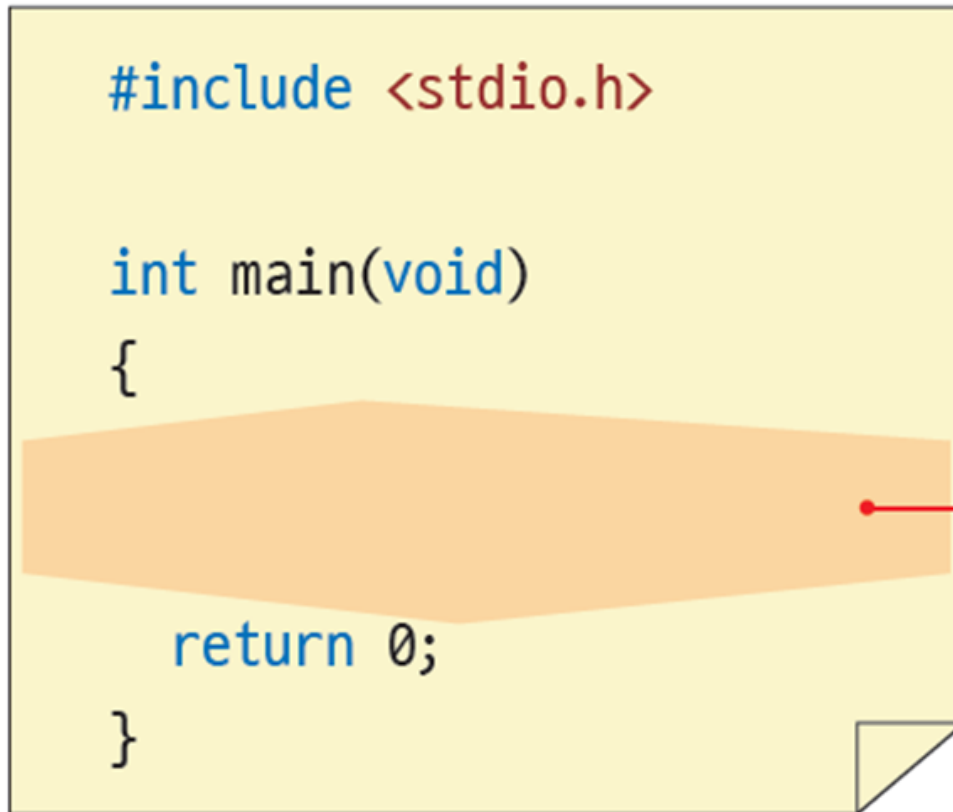
program

Where to write down your work


```c
#include <stdio.h>

int main(void)
{

    return 0;
}
```

program

Write the sentence that performs the desired task here.

Brief source description


```c
#include <stdio.h>              Includes header files.

int main(void)                  Start main function
{
    printf("Hello World!");     Print "Hello World!" on the screen
    return 0;                   Returns 0 to the outside world
}                               Exit main function
```

program
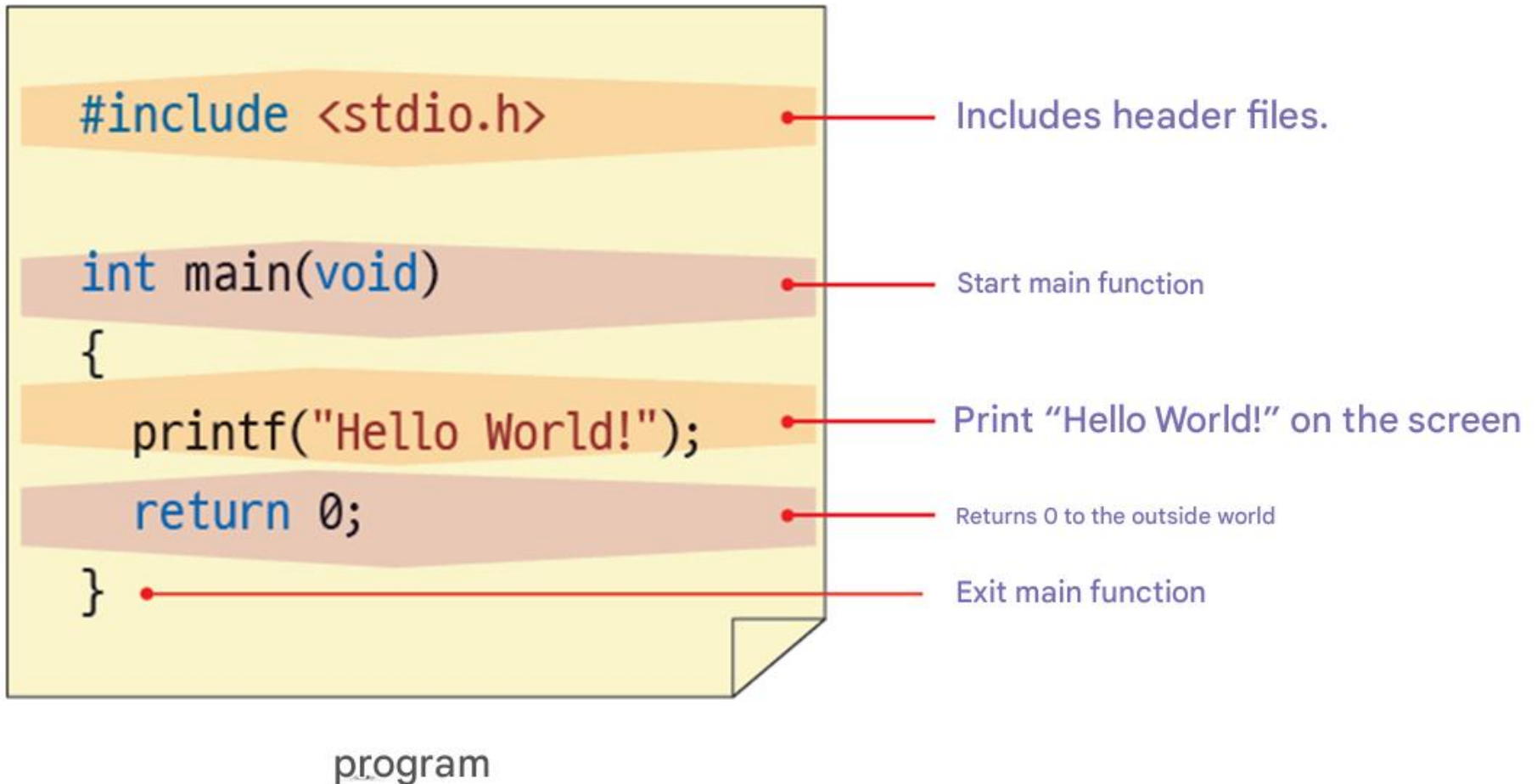
Including header files

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

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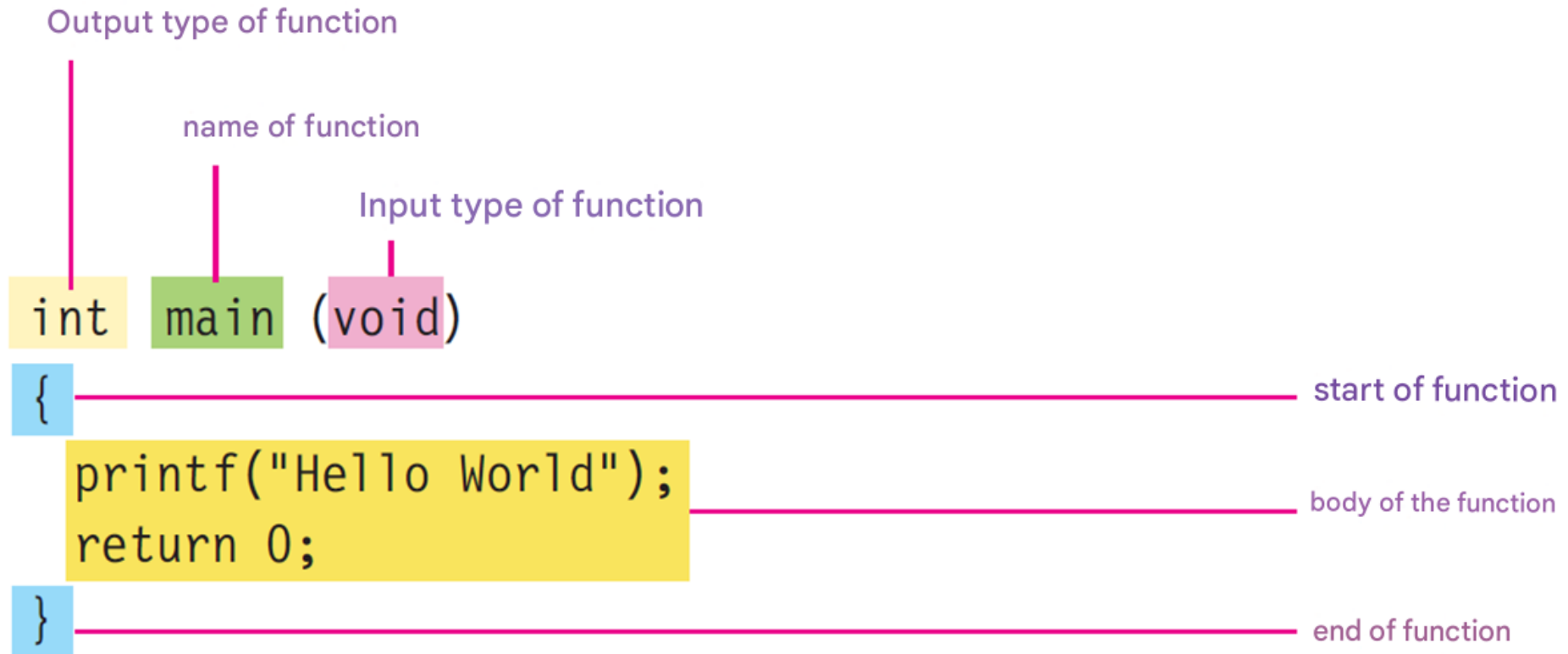
```
#include <stdio.h>
```

- 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

Input

Function

Output
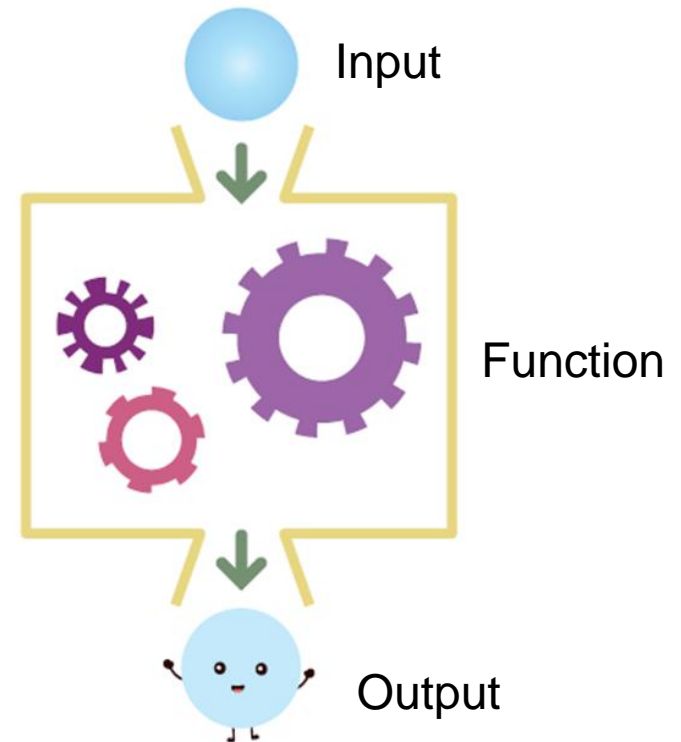
A brief description of the function

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

- 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
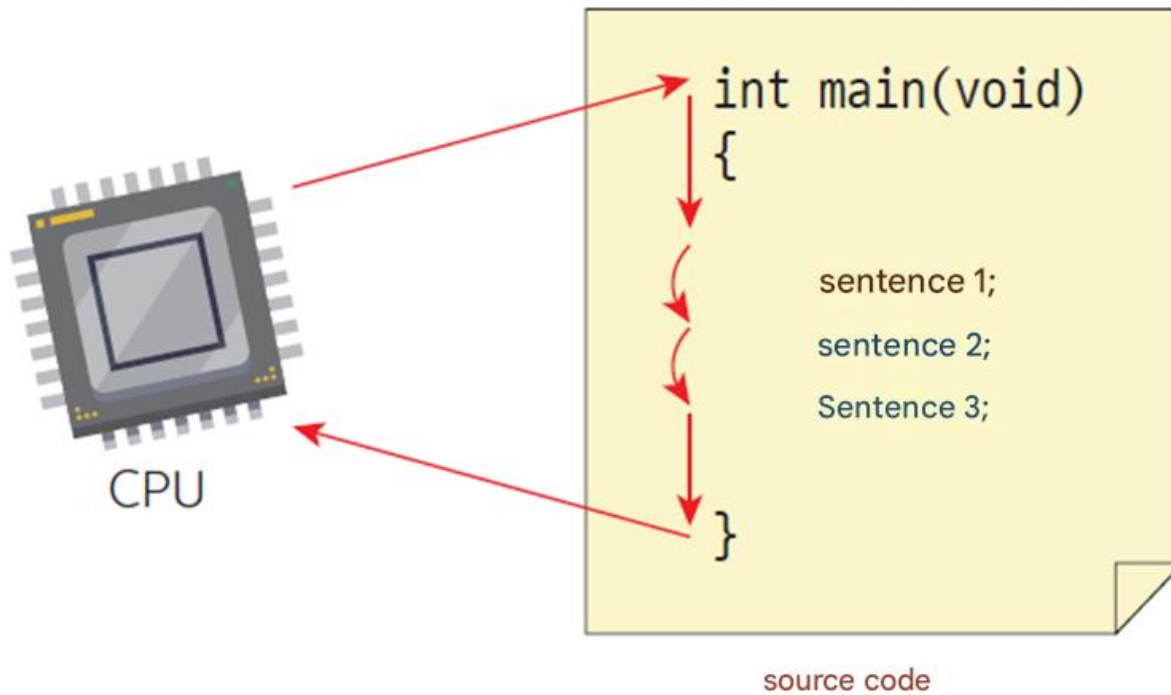
source code

Statements in source code are basically executed sequentially.

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

```c
#include < stdio.h >

int main( void )
{

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

  return 0;
}
```
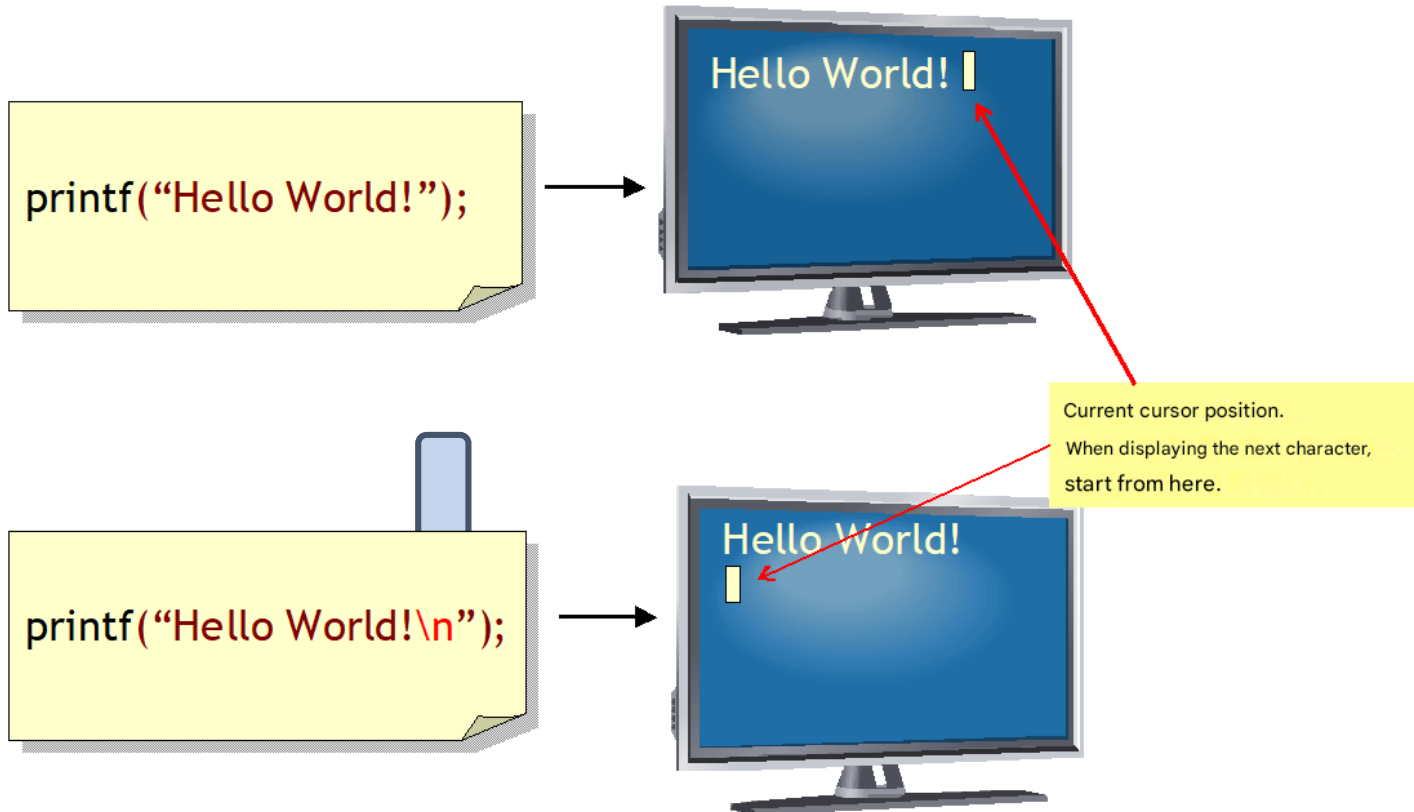
2 of Sentences are executed sequentially.

Hello World! Kim ChulSoo

New line character

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

Changed program

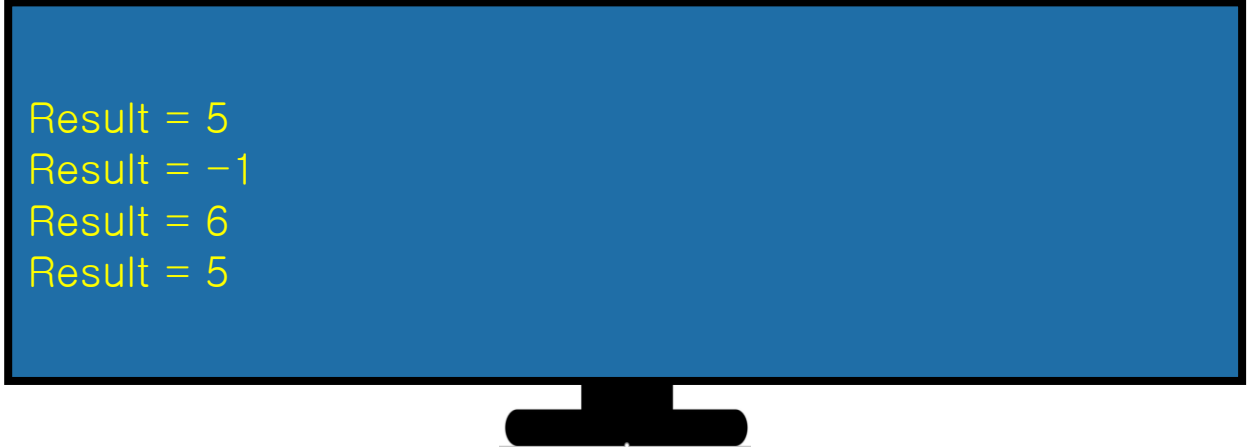- Adding a new-line character gives us the result we want .

```c
#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

```c
#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.
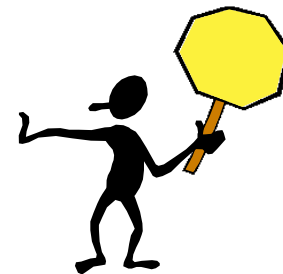
Error correction process

Error #1

Error #2

Error #3

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

```c
#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

Debugger

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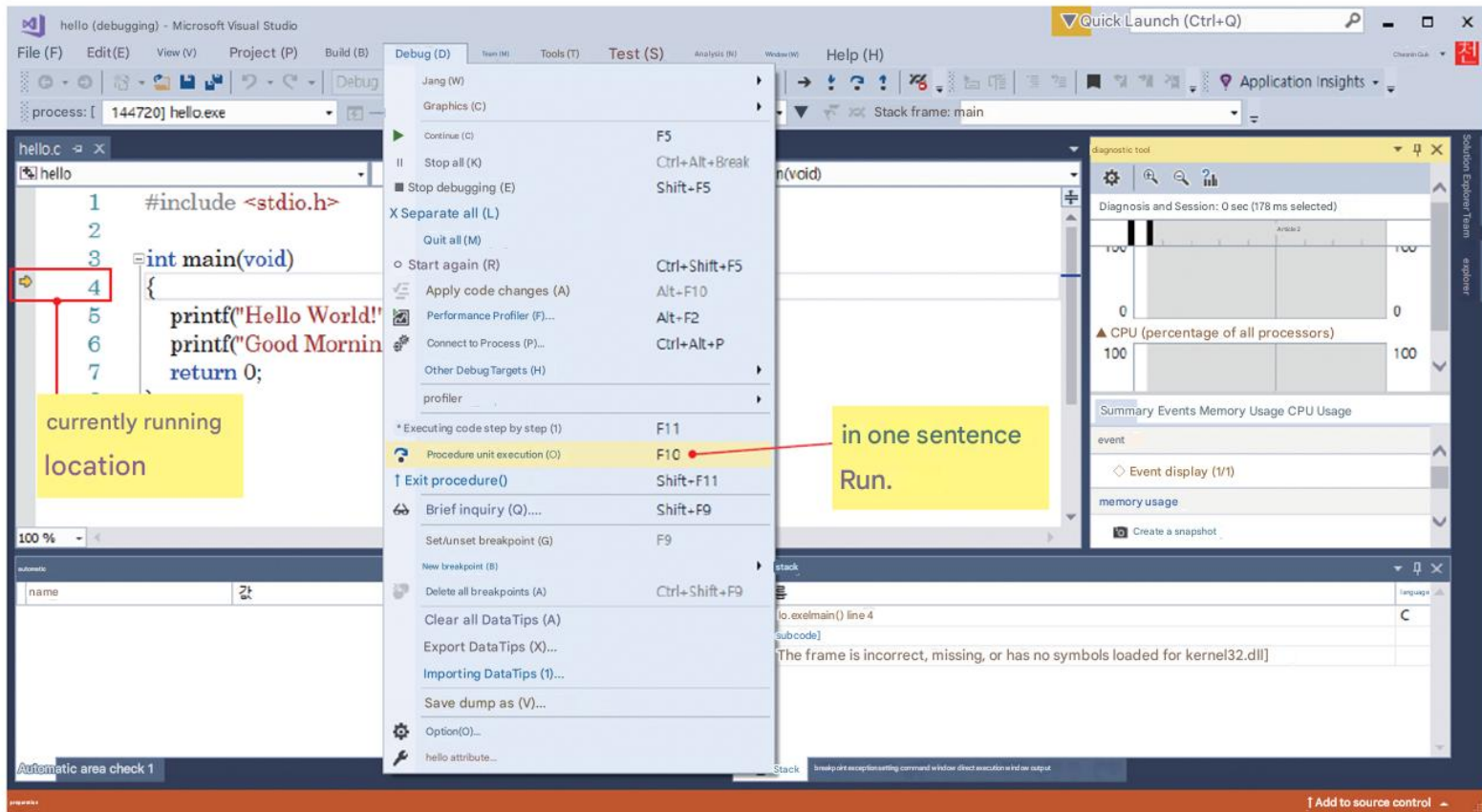
Defining commands for the debugger

- 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

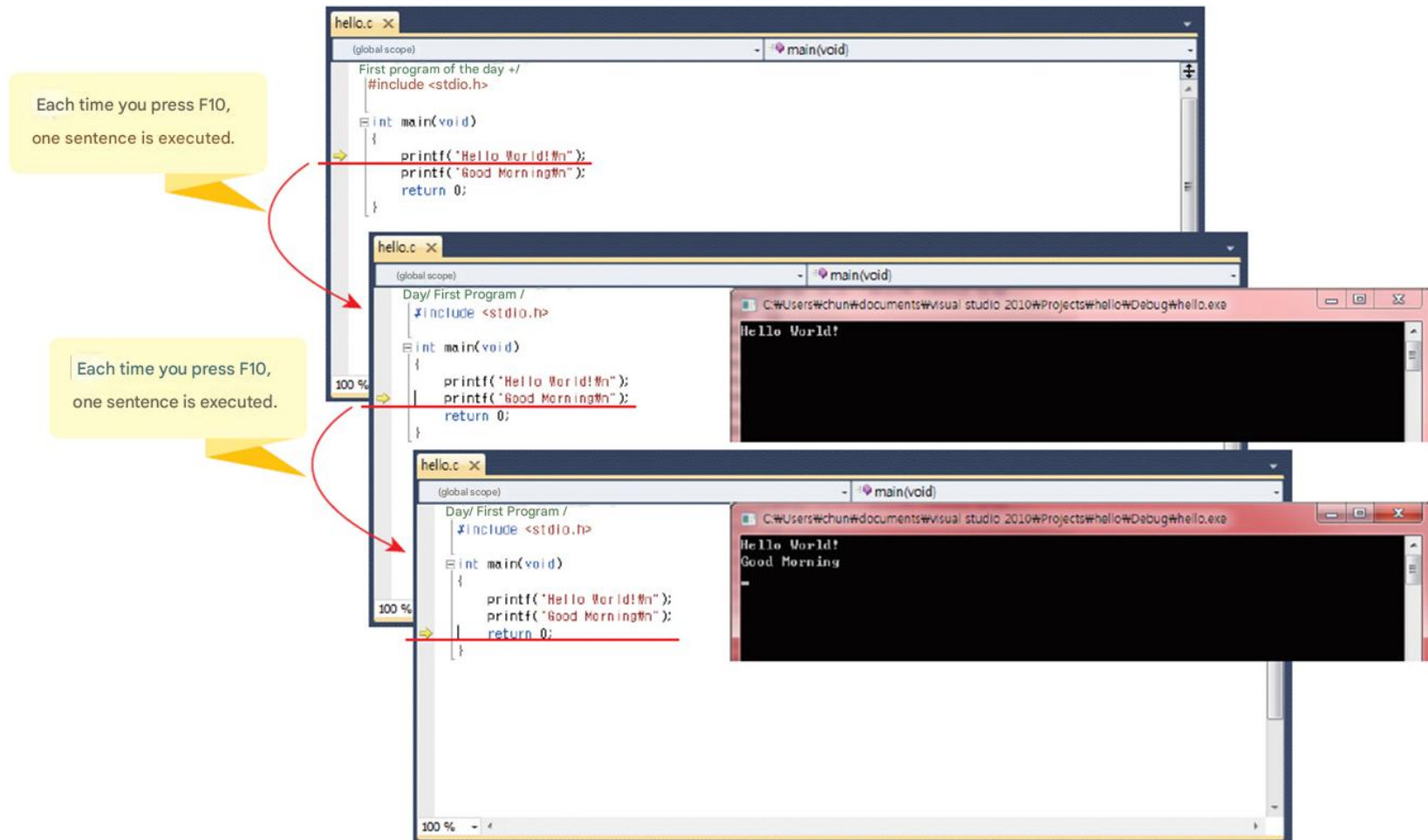|	
Start Debug→Go(F5)	Run the program in debugging mode.
Restart(Ctrl + Shift + F5)	Rerun the program.
Stop Debugging(Shift + F5)	Stop debugging.
Break Execution	If you press this button while the program is running, execution will stop at the current location.
Step Into(F11)	Executes a single statement. 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.

• 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;
)
```

Solution bug.c

```c
1  #include <stdio.h>
2
3  int Main(void)          — main
4  (                       — (should be {not {).
       5 printf(Hello? \n);
6      printf(There are many errors in this code \n) -        ── There must be a ; at the end of a sentence.
       7 print(I will fix everything.\n);
8      return 0;                                    Strings are enclosed in quotes.
9  )
```

It should be printf, not print.

- Program with fixed errors

```c
#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