Ch.3 C Program Components
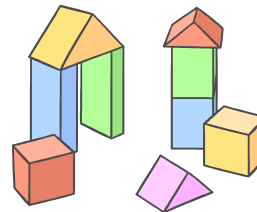
What you will learn in this chapter

* annotation
* Variables , constants
* Function
* sentence
* Output function printf ()
* Input function scanf ()
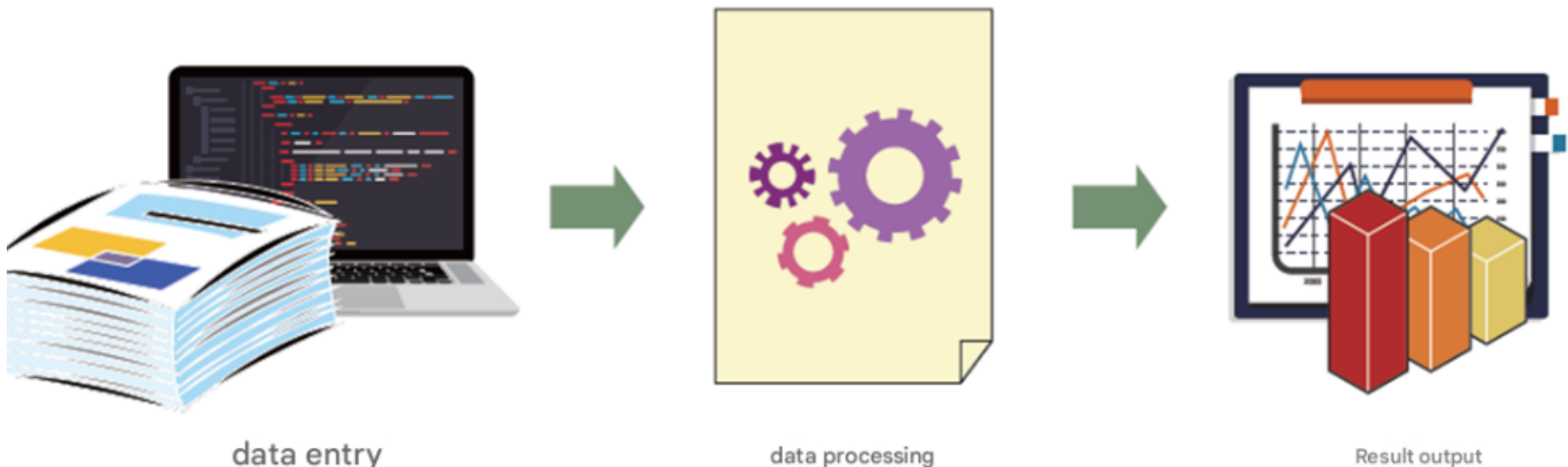* Arithmetic operations
* Assignment operation

In this chapter, we will look at the components that make up a C program .

General program form

- receiving data (input stage),
- processing the data (processing stage)
- and then displaying the results on the screen (output stage) .

data entry　　　　　　data processing　　　　　　Result output

Addition Program #1

add1.c

```c
1   /* Program to calculate the sum of two numbers */   — annotation
2   #include <stdio.h>                                   — preprocessor
3
4   int main(void)                                       — function
5   {
6       int x;      // Variable to store the first integer
7       int y;      // Variable to store the second integer
8       int sum;    // Variable that stores the sum of two integers
9
10      x = 100;
11      y = 200;                                          — calculation
12
13      sum = x + y;
14      printf("Sum of two numbers: %d", sum);
15
16      return 0;
17  }
```

variable declaration

Sum of two numbers : 300

Comment

```c
/* Program to calculate the sum of two numbers * /
#include < stdio.h >
int main( void )
{
    int x; // variable to store the first integer
    int y; // variable to store the second integer
    int sum; // Variable that stores the sum of two integers
x = 100;
y = 200;
sum = x + y;
    printf ( " Sum of two numbers : %d" , sum);
    return 0;
}
```

Comments are text that explains the code .

2 ways to comment

```
/* single line comment * /


/* several
In a line
Comment done* /
```

```
// This entire line is a comment .
int x; // From here to the end of the line is a comment .
```

The importance of comments

- When someone else looks at the program , it is much easier to understand the contents of the program if there are comments . If a lot of time has passed , even the creator may not remember the contents well .

- A good comment does not repeat or explain the code . It should clearly state the intent of the code .

Annotation style

```
/*
File name : add.c
Description : Program to add two numbers
Author : Hong Gil-dong
* /
```

```
/************************************
* File name : add.c
* Description : Program to add two numbers
* Author : Hong Gil-dong
************************************ /
```

Indentation

- *Indentation :* Indenting sentences on the same level a few ch aracters from the left end .


```
#include <stdio.h>

int main(void)
{
    int x;      // Variable to store the first integer
    int y;      // Variable to store the second integer
    int sum;    // Variable that stores the sum of two integers

    ...

    return 0;
}
```

Insert blank lines to differentiate between meanings.

Explain the intent of the program in comments.

If the content is the same, indentation is applied.

Without comments and indentation ..

```c
#include < stdio.h >
int main( void ) {
int x; int y; int sum; x = 100; y = 200; sum = x + y;
printf ( " Sum of two numbers : %d" , sum); return
0;
}
```

Running is However,
it is difficult to know
what kind of processing
the program is doing, a
nd it is also difficult
to distinguish sentences
on the same level beca
use there is no indentati
on .

Preprocessor

- stdio.h contains definitions of library functions for standard input and output .

#include <stdio.h>

- Preprocessor
- # Starts with a symbol

Preprocessor

hello.c
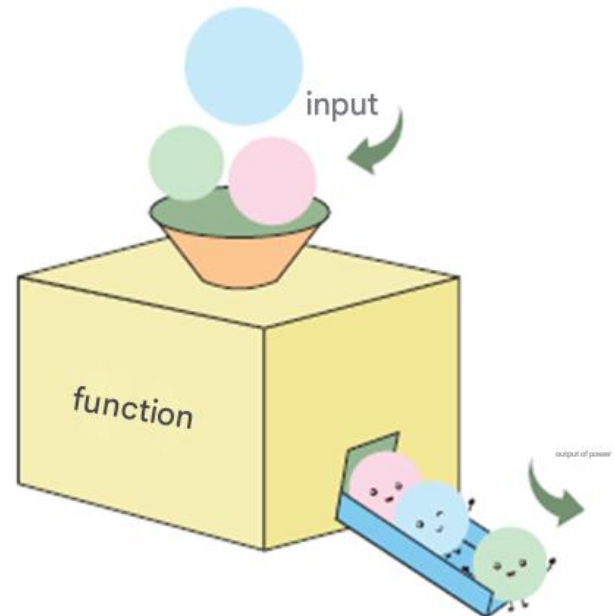
stdio.h

Function

- Function : A set of processing steps that perform a specific function, grouped in parentheses and named accordingly.

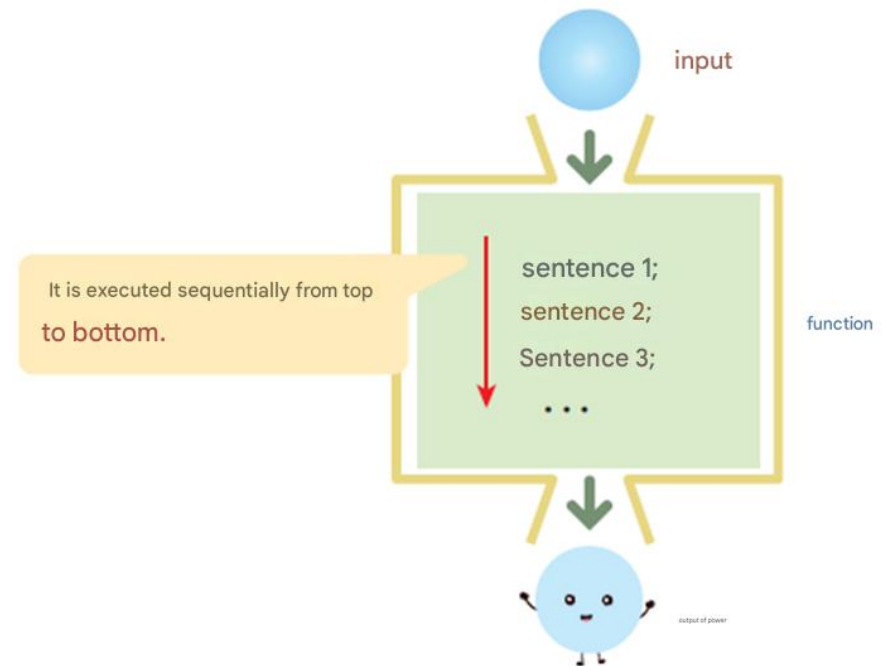- (parts) Functions are the basic units that make up a program.

```
int main(void)
{
        ...
        ...
}
```
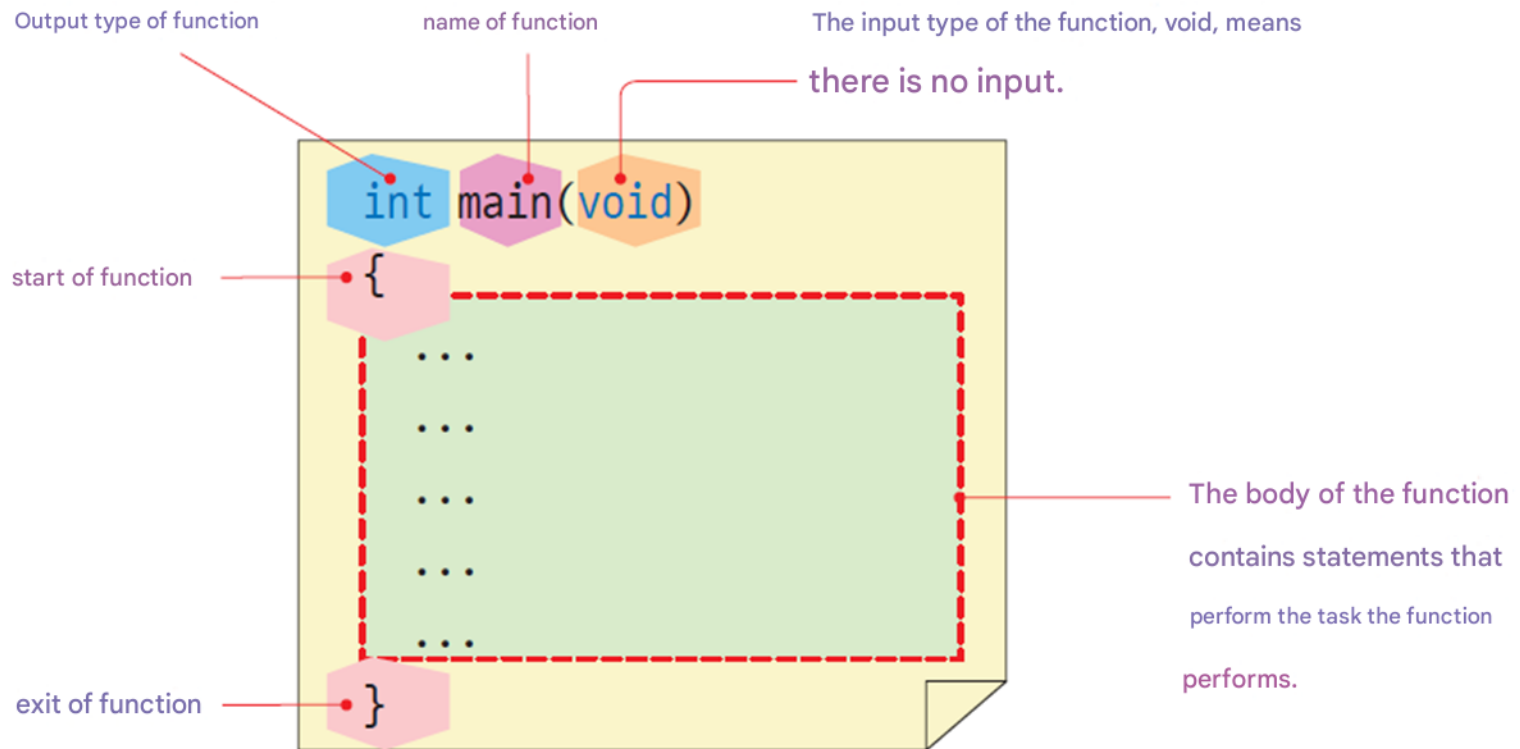
What's inside the function

Q) So what is inside the function ?

A) Inside the function, the processing steps (state ments) that the functio n processes are listed in curly brackets.

input

It is executed sequentially from top to bottom.

sentence 1;
sentence 2;
Sentence 3;
. . .

function

output of power

Structure of a function

Output type of function

name of function

The input type of the function, void, means there is no input.

```
int main(void)
{
    ...
    ...
    ...
    ...
    ...
}
```

start of function

The body of the function contains statements that perform the task the function performs.

exit of function

Function

- The statement that performs the task must be placed inside a function.
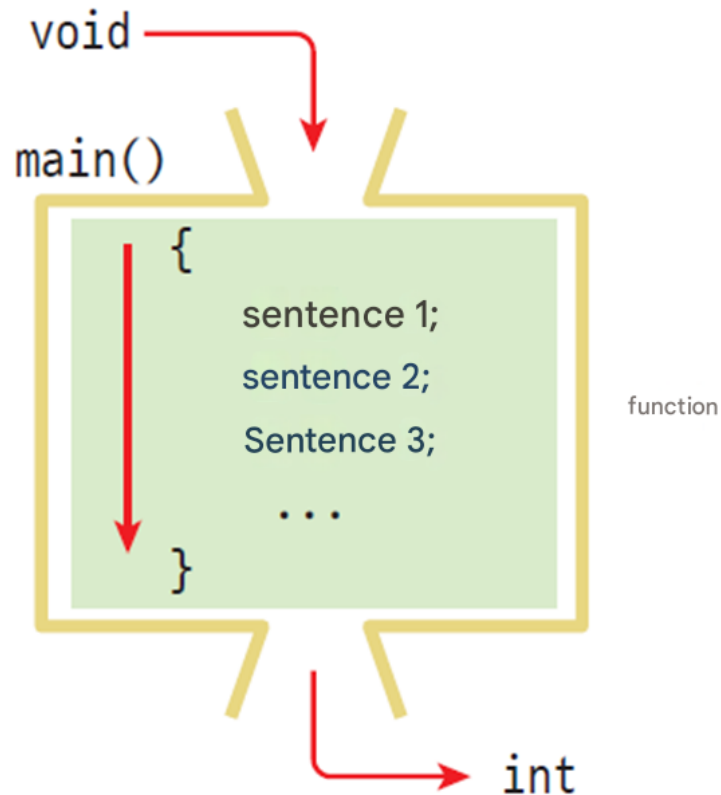
return statement

- return is a keyword that returns a value while terminating a function .
- To return a value, write the return value after return .

```c
#include < stdio.h >

int main( void ) {

        …

        …
        return 0;

}
```
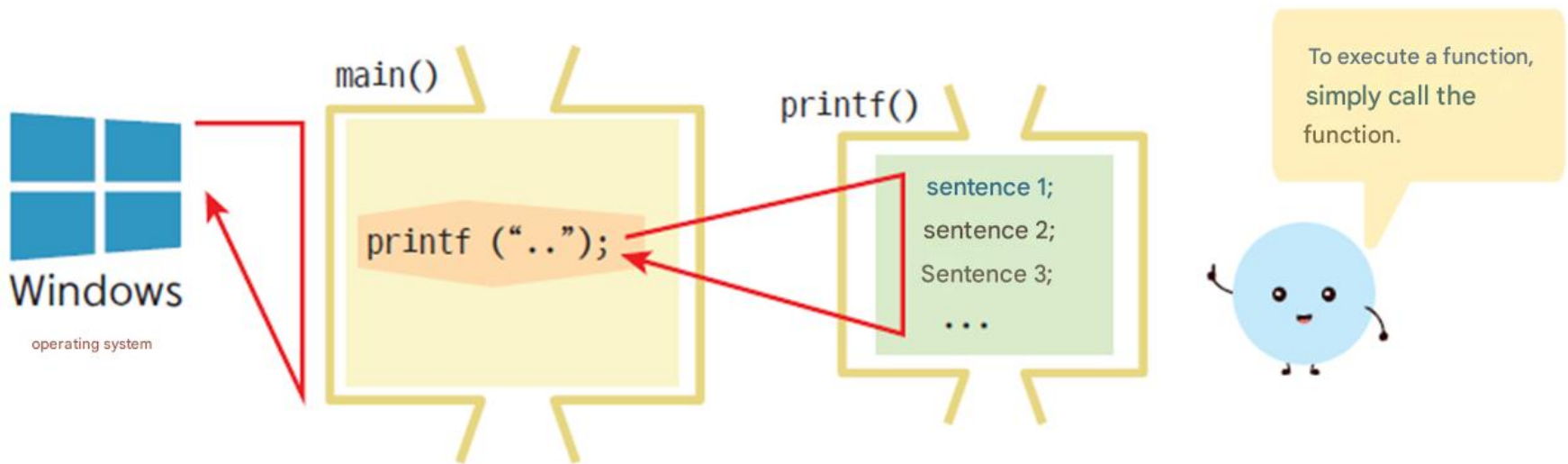
Who will call main() ?

Variable

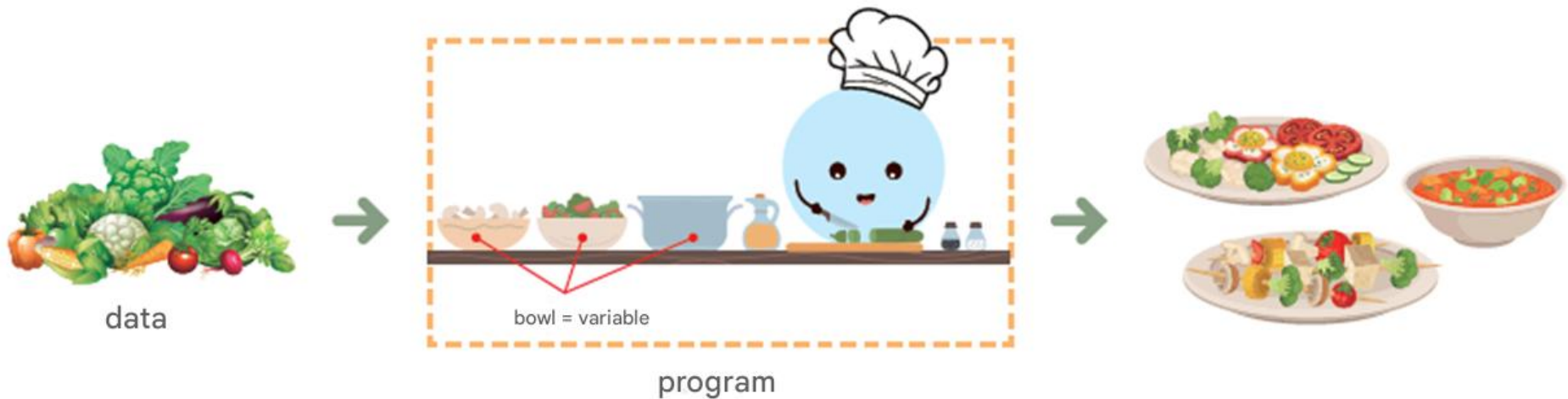- Memory space used for temporarily storing data used by a program.
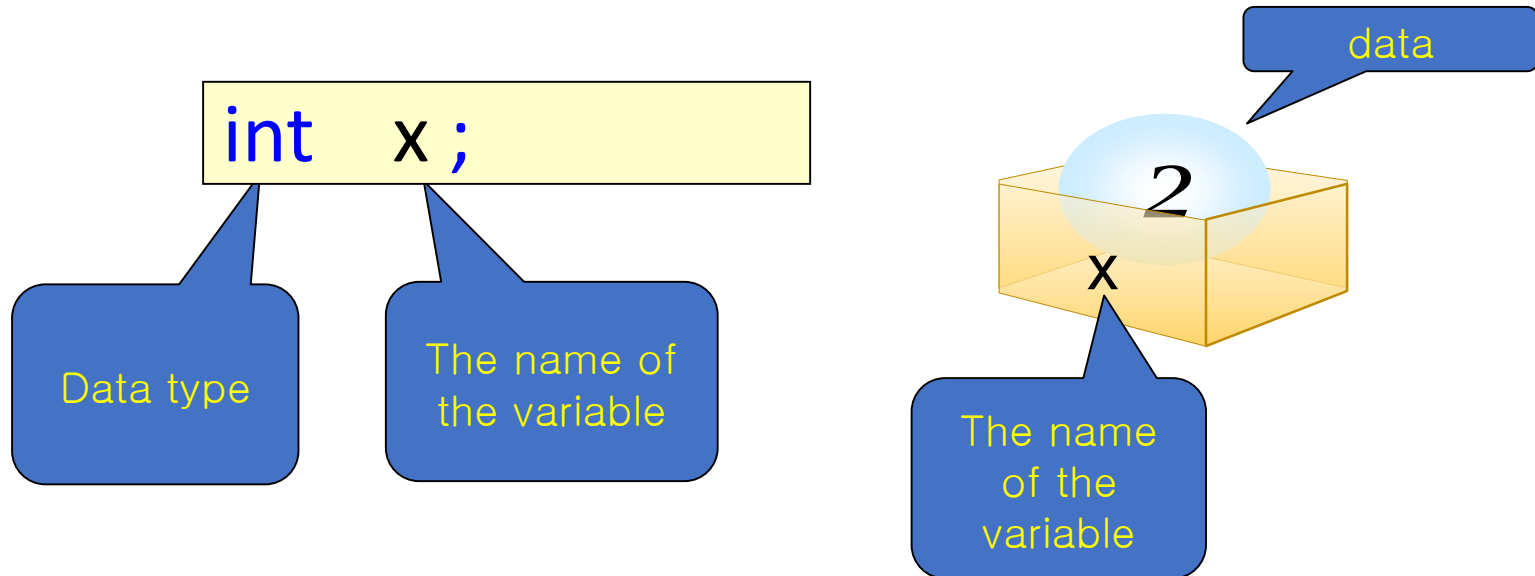
Why are variables needed ?

• Variables serve to temporarily store data values .

Types of variables

- A variable can be thought of as a box that holds data .

int   x ;

Data type

The name of the variable

data

2

x

The name of the variable

Types of variables
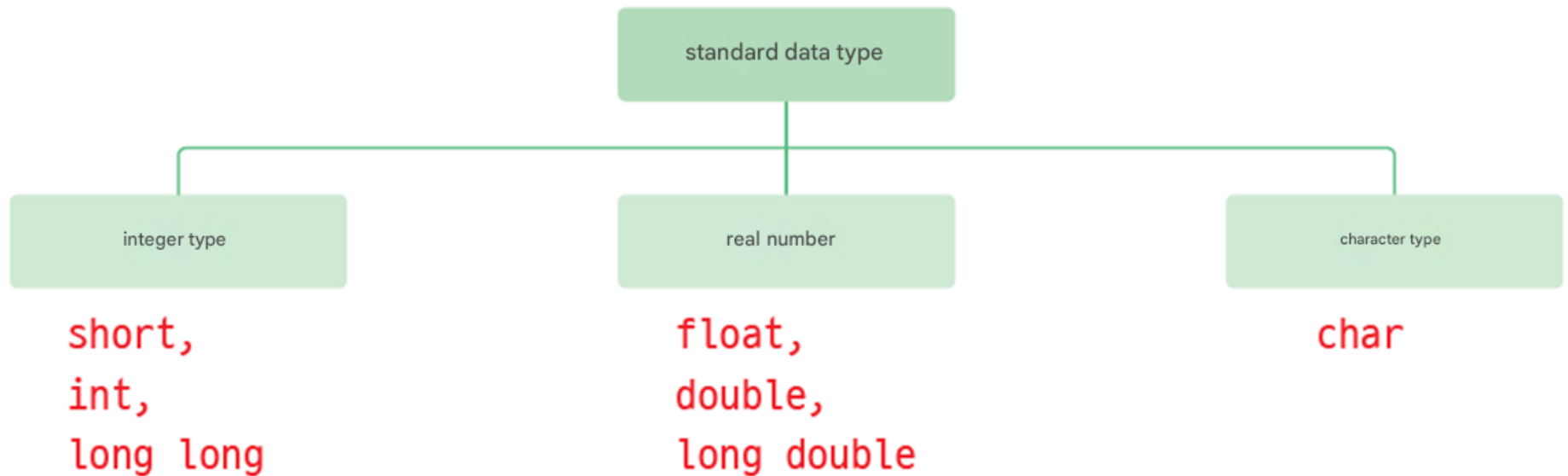
- Variables have several types depending on the type of data .
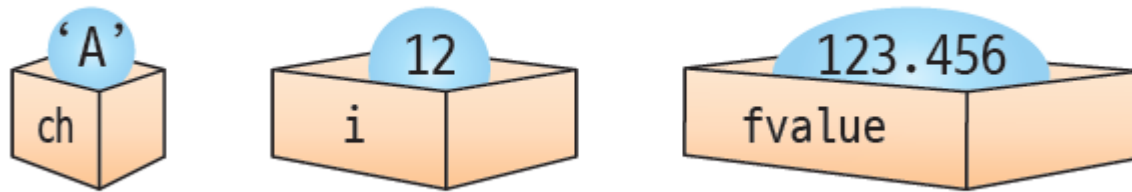
Data type

- It specifies whether the data to be stored in the variable is an integer, a real number , or some other data .
- Data types include integers , floating point numbers (real numbers), and character types .

standard data type		
integer type	real number	character type
short, int, long long	float, double, long double	char

Declaring variables

- Variable declaration : Telling the compiler in advance what type of variable will be used.

From now on
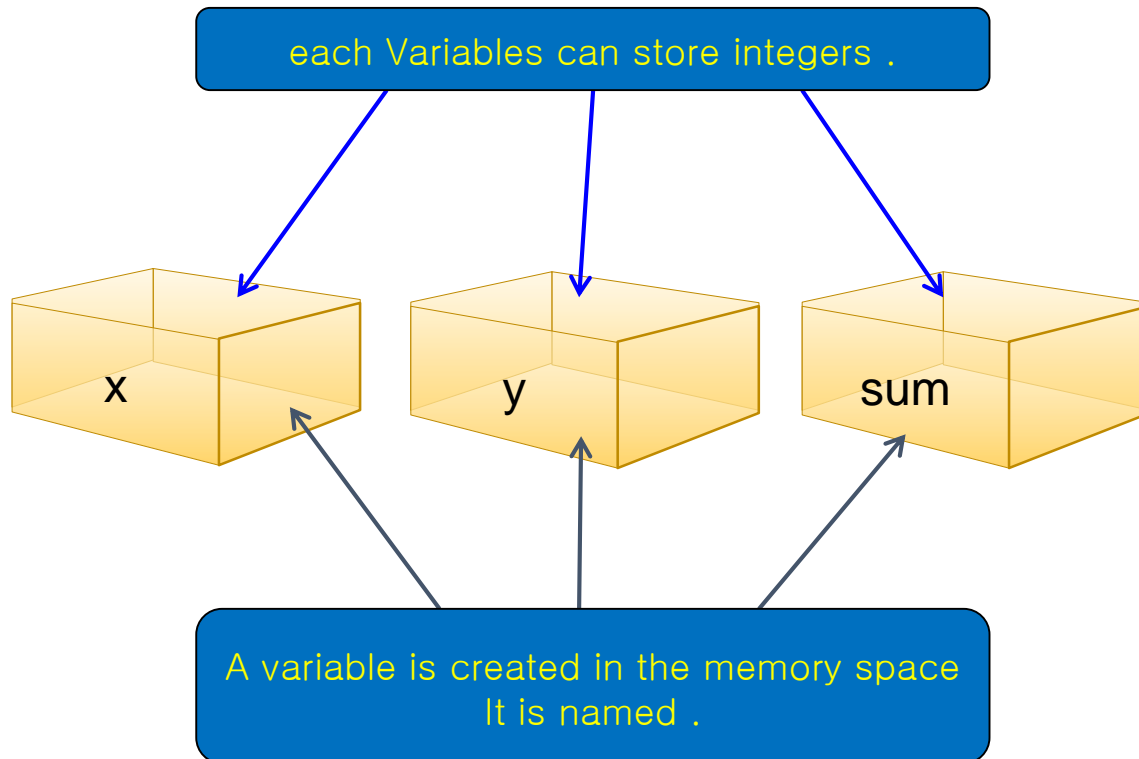In the program
Let me introduce the
variables that will be used.

```
int x;
int y;
int sum;
```
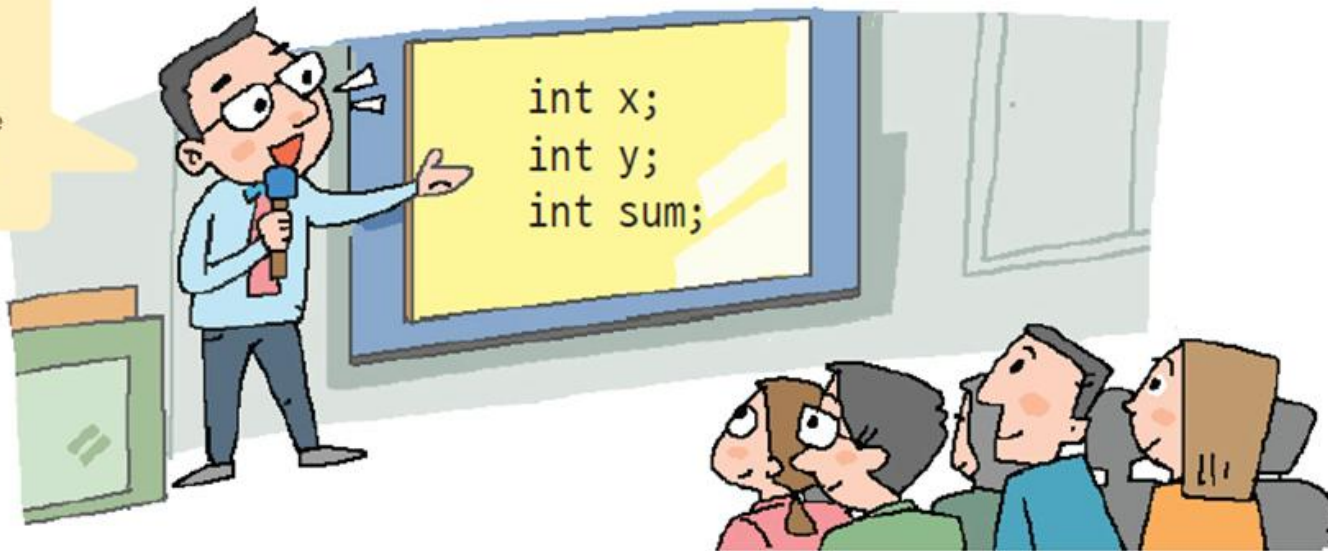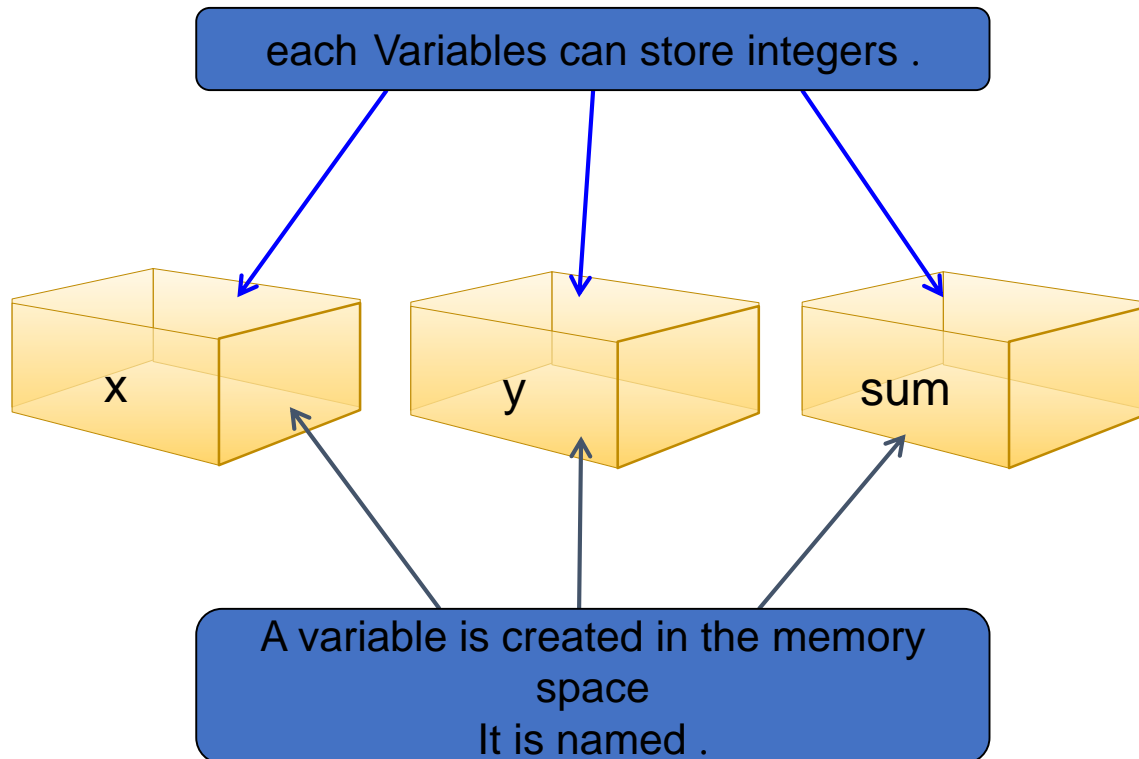
compiler

Declaring variables

Declaring variables

```
int x ; // first Integer Save Variable
int y ; // second Integer Save Variable
int sum; // two Integrity Sum Save Variable
```

each Variables can store integers .

x   y   sum

A variable is created in the memory space
It is named .

Declaring multiple variables on one line

```
int x, y, sum; // possible !!
```

each Variables can store integers .

x   y   sum

A variable is created in the memory space
It is named .

Name of variable

- Identifier : The programmer can name variables as he pleases , but he must follow some rules . Just as names such as " Hong Gil-dong" and " Kim Young-hee" identify people, variable names identify variables .
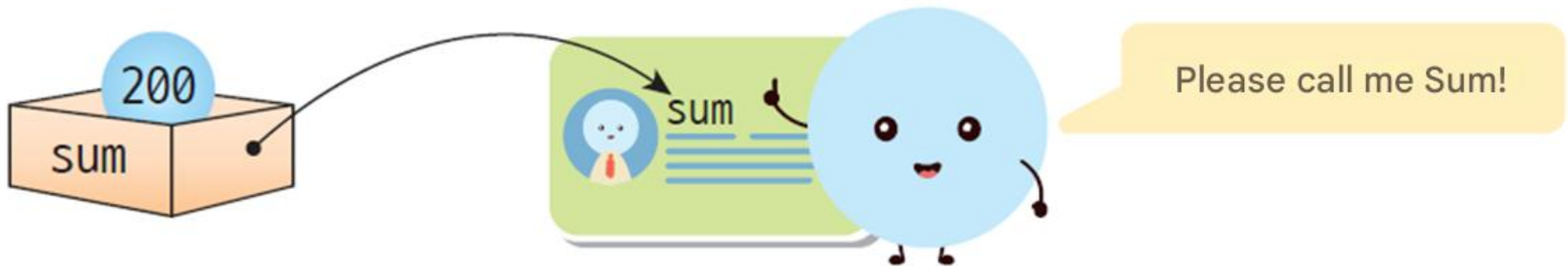
Name of variable

- **Rules for creating**
  - Identifiers consist of English letters, numbers , and the underscore c haracter _ .
  - There must be no spaces in the middle of the identifier .
  - The first character of an identifier must be a letter or the underscore symbol _ . An identifier cannot start with a number .
  - Uppercase and lowercase letters are distinguished . Therefore, the va riables index , Index, and INDEX are all different variables .
  - Identifiers that are identical to keywords in the C language are not allowed .

Function name

Variable name

```
int sub(void)
{
        int x;

}
```

Keyword

- Keyword : A special word that has its own meaning in the C language . Also called reserved words .

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

The name of the variable

- *sum // Starts with an English alphabet letter*
- *_count // can start with an underscore character .*
- *number_of_pictures // You can put an underscore character in the middle .*
- *King3 // You can also put numbers in, as long as it's not the very first one .*

- *2nd_base(X) // Cannot start with a number .*
- *money# // Symbols such as # cannot be used .*
- *double // double is a keyword in the C language .*

Good variable names

- You should choose a name that best describes the role of the variable .
  - i , j, k (X)
  - year, month, date (O)

- How to create a multi-word name
  - underscore Method : bank_account
 - Capitalize the first letter of the word : BankAccount

a, b, c, d,…

number,
average,
sum,…
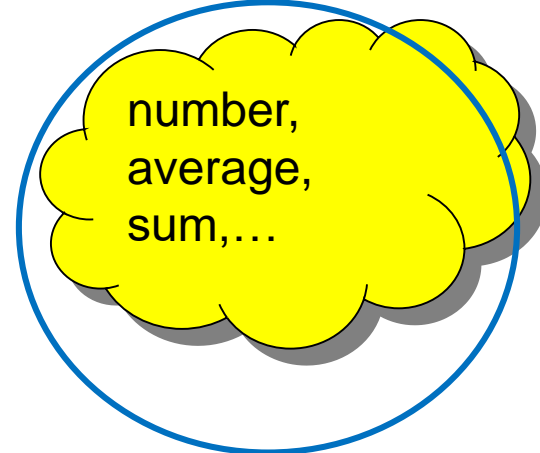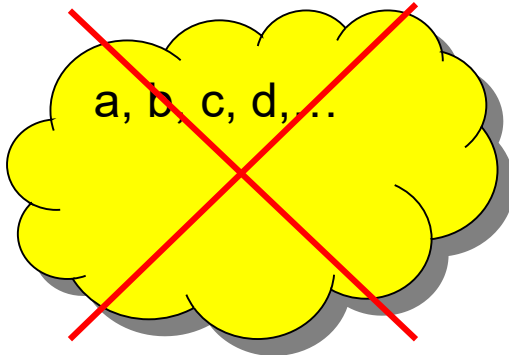
Initialization of variables

- You can give initial values to variables .
    - int x = 10;
    - int y = 20;
    - int sum = 0;

- If the variables are of the same type , you can declare and initialize them on the same line .
    - int width = 100, height = 200;

- Initializing as follows is not syntactically incorrect, but should be avoided :
    - int width, height = 200;

width is not initialized .

formula

- Expression : Operands and Operators consisting of expression
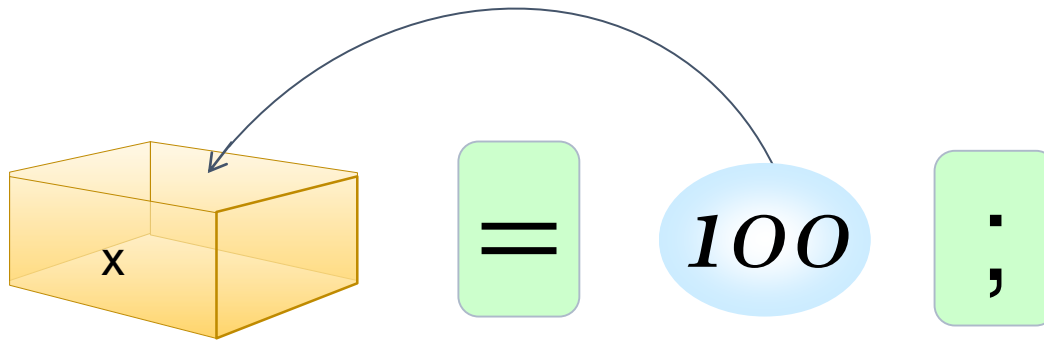- A formula has **a result** .

Formula when x is 3
The value of x ^{2} - 5x + 6
Calculate .
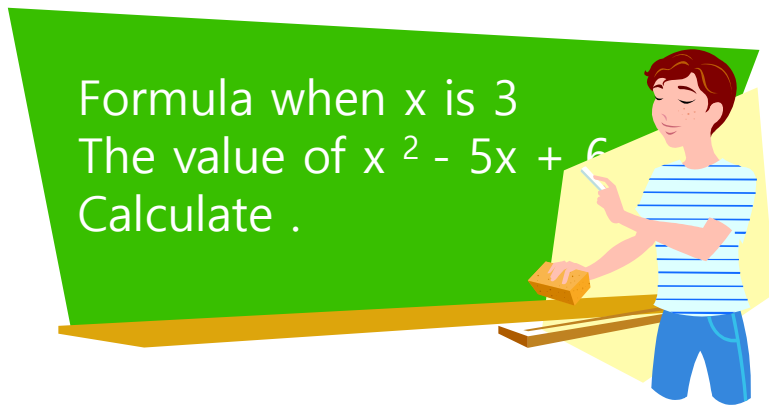
```
int x, y;

x = 3;
y = x * x - 5 * x + 6;
printf ("%d\n", y);
```
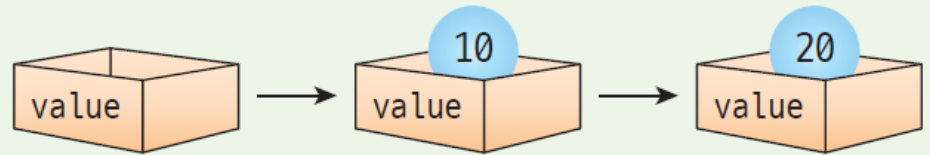
Storing values in variables

x = 100;

- Assignment operation : An operation that stores a value in a variable.
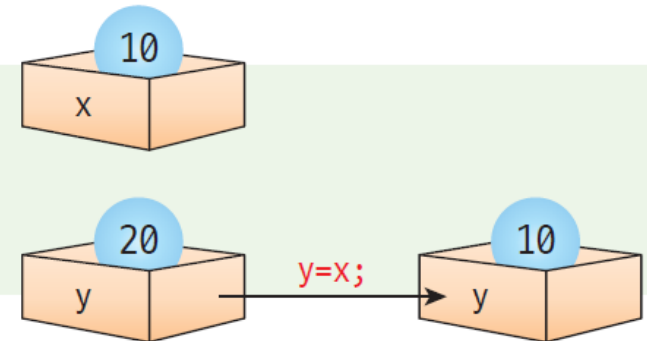- Assignment operation = assignment operation = assignment operation

various Assignment operation

- A variable can store a value using the = symbol, and the value of the variable can be changed as many times as you like .

```
int value;
value = 10;
value = 20;
```

-
```
int x = 10;
int y = 20;
y = x;              // y는 10이 된다.
```
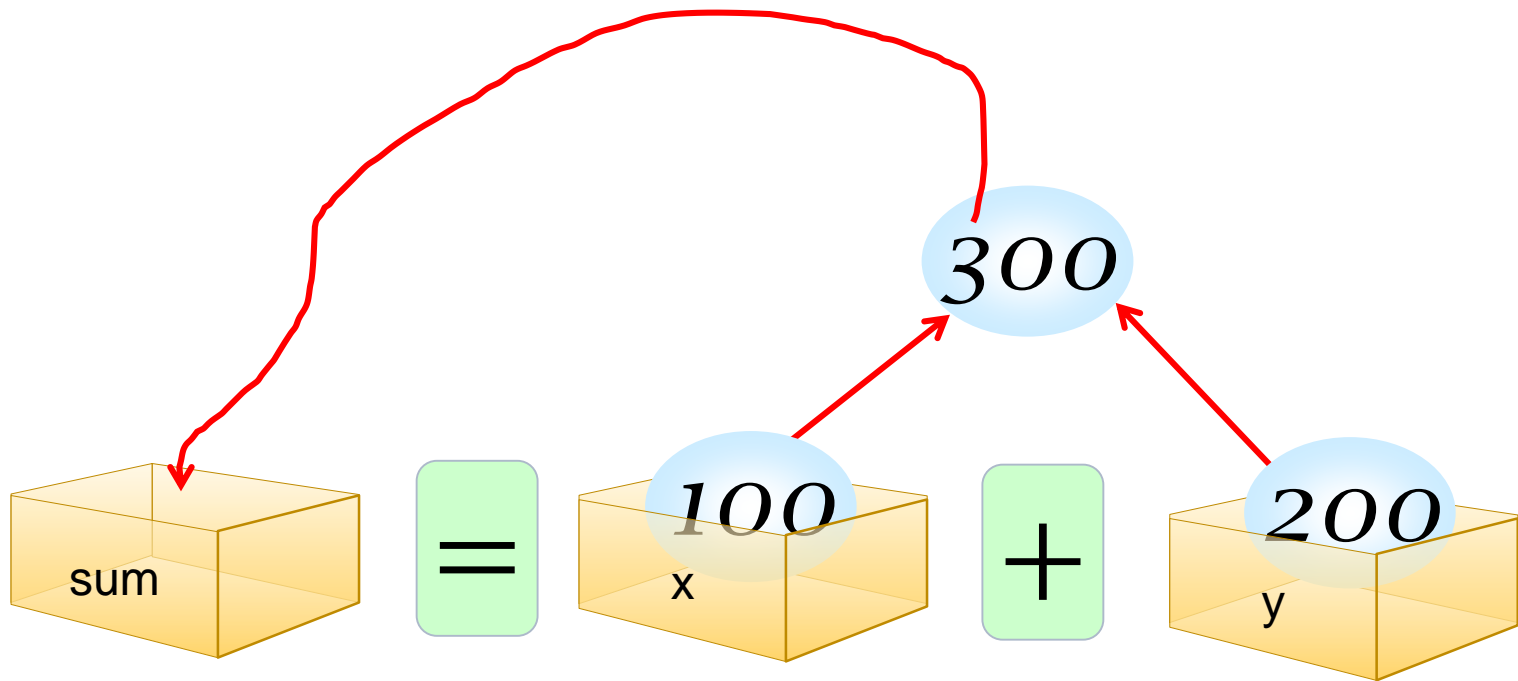
Arithmetic operations

- Arithmetic operators are similar to the operation symbols commonly used in mathematics .
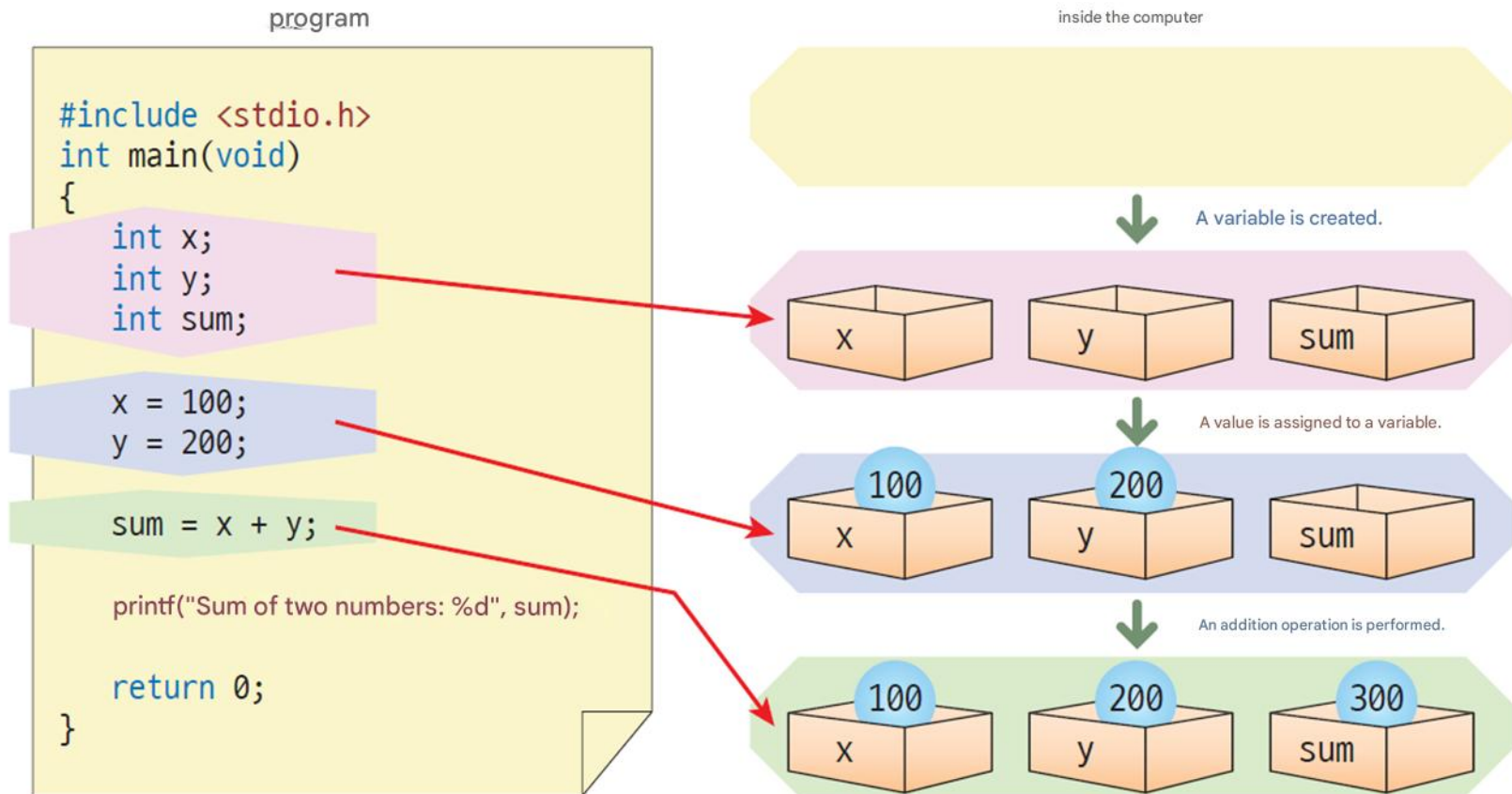
calculation	operator	C formula	symbols in mathematics
addition	+	x+y	x+y
subtraction	−	x − y	x − y
multiplication	*	x*y	xy
division	/	x / y	x/y or s or x÷y y
remain	%	x%y	x mod y

Arithmetic operations

sum = x + y;

Organize

program

```
#include <stdio.h>
int main(void)
{
    int x;
    int y;
    int sum;

    x = 100;
    y = 200;

    sum = x + y;

    printf("Sum of two numbers: %d", sum);

    return 0;
}
```

inside the computer

A variable is created.

| x | y | sum |

A value is assigned to a variable.

100   200

| x | y | sum |

An addition operation is performed.

100   200   300

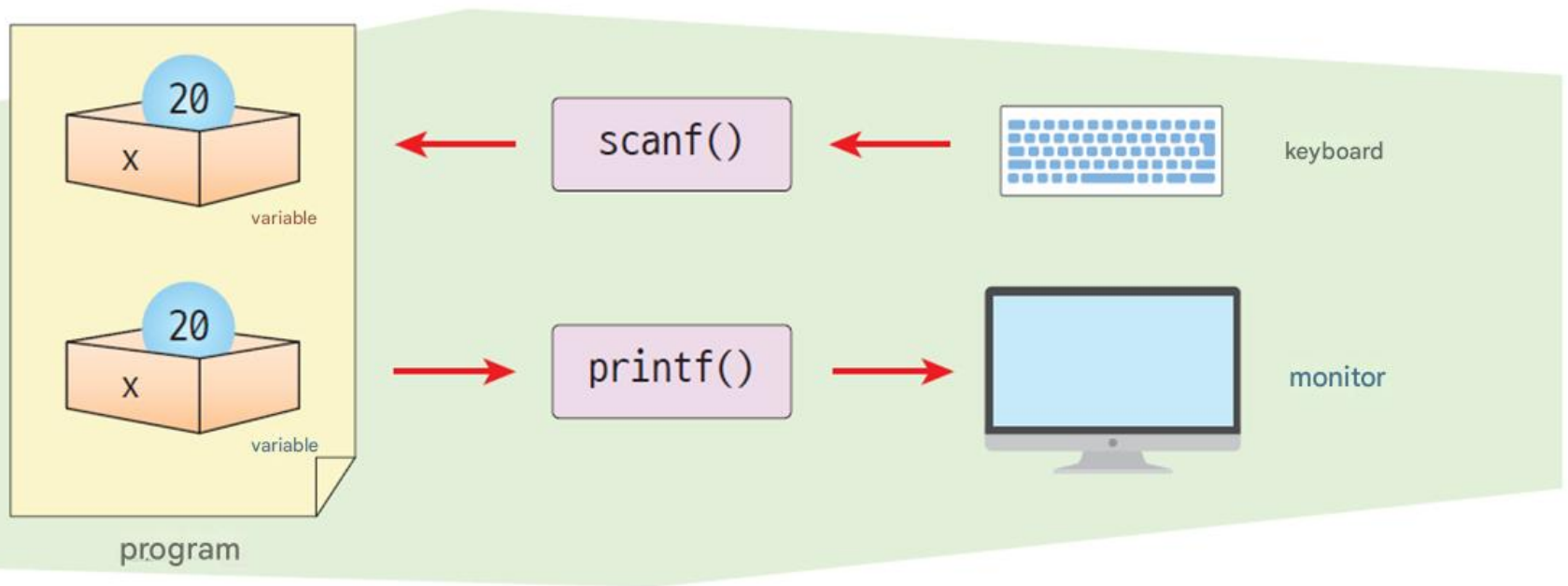| x | y | sum |

Library functions

- Library functions : Library functions are functions that the compiler provides for programmers to use.

  - printf (): Standard output function for printing to the monitor.
  - scanf ( ): Standard input function for input from the keyboard.
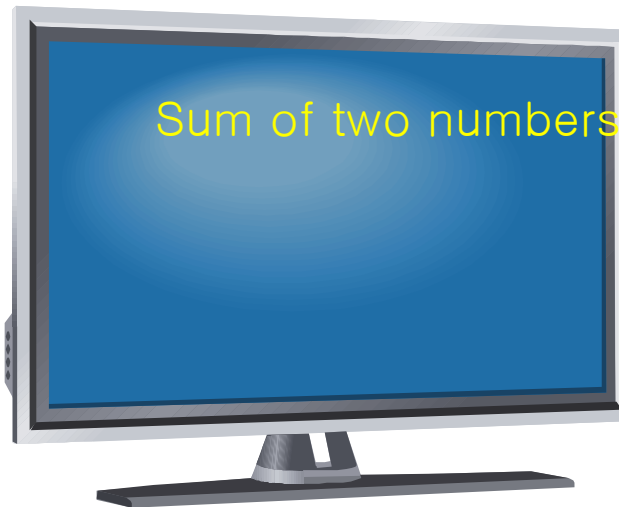
String output

printf ("Hello World!\n ");

- string : "Hello World!\n" and A list of several characters together

Hello World!

Output variable values

printf (" Sum of two numbers : %d \n, sum");
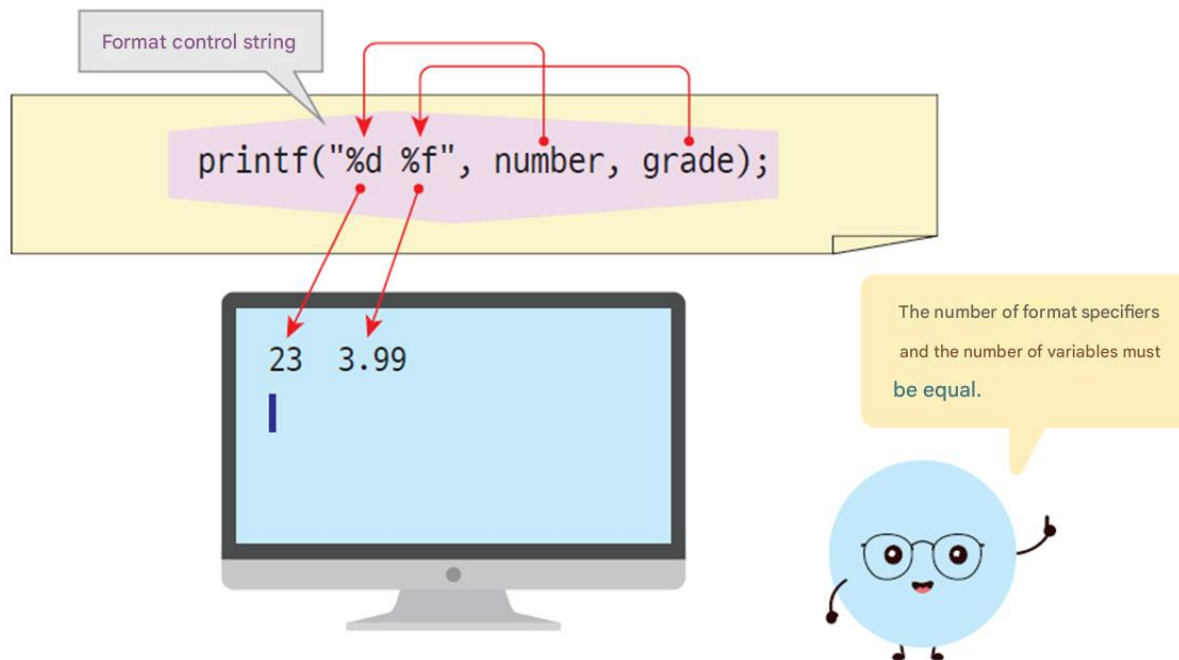
Sum of two numbers : 30

Format specifier

- Format specifier : Specifies the format in which values are printed in printf () .

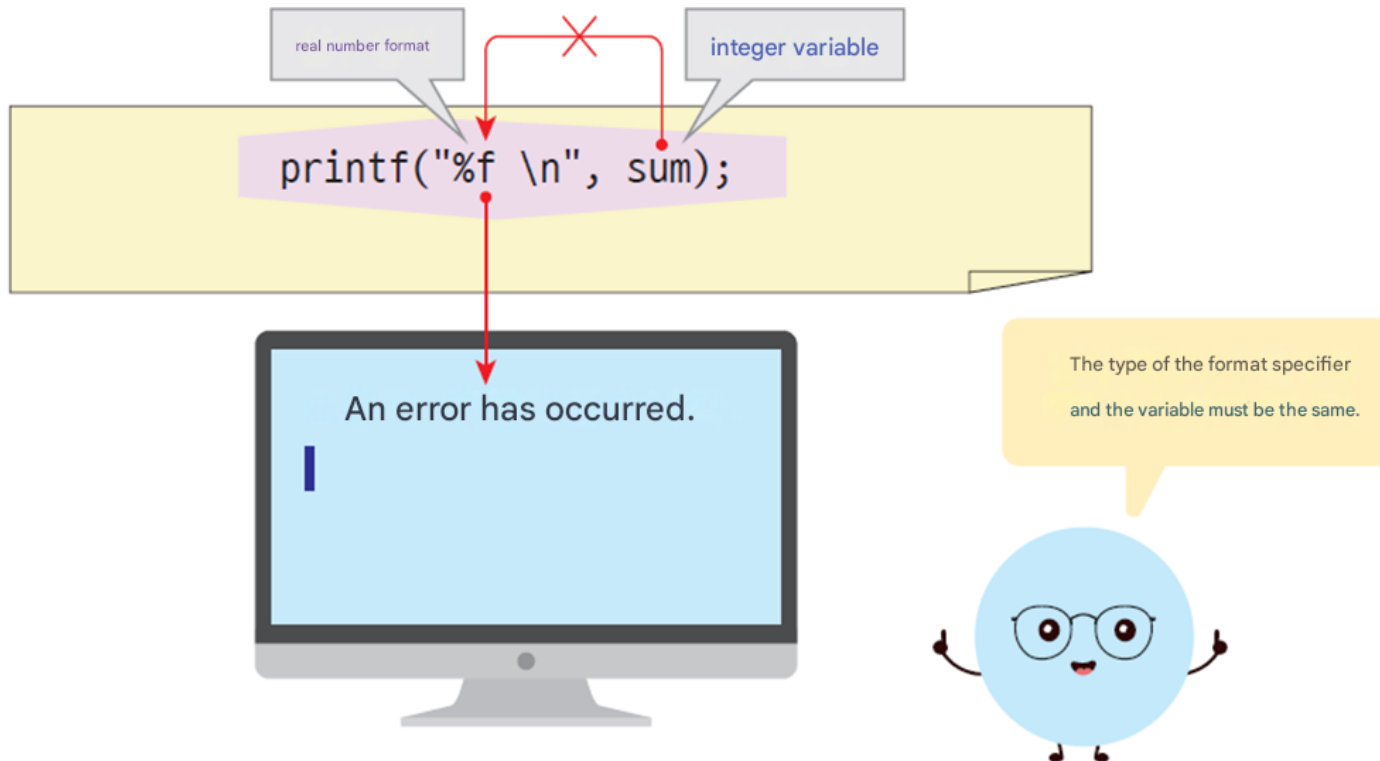| format specifier	meaning	yes	execution result
%d	Output as a decimal integer	`printf("%d \n", 10);`	10
%f	Printed by mistake	`printf("%f \n", 3.14);`	3.14
%c	output as text	`printf("%c \n", 'a');`	a
%s	output as string	`printf("%s \n", "Hello");`	Hello

Output multiple variable values

• You can think of it as the value of the variable being substituted in the place of the format specifier and then printed .

Format control string

```
printf("%d %f", number, grade);
```

23   3.99

The number of format specifiers and the number of variables must be equal.

Caution !

- The data types of the format and variables must match.

real number format

integer variable

```
printf("%f \n", sum);
```

An error has occurred.

The type of the format specifier
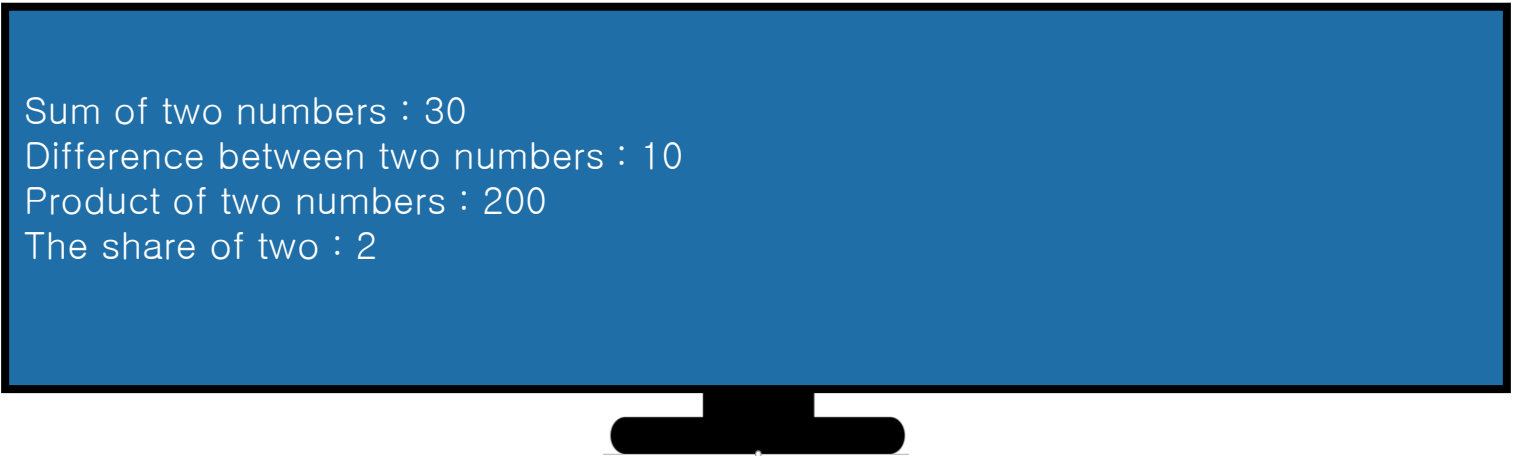and the variable must be the same.

Field width and precision

- When printing using printf () , you can specify the size of the field in which data is printed .

output statement	output result	explanation
printf("%10d", 123);	⬜⬜⬜⬜⬜⬜⬜ 1 2 3	Width 10, right aligned
printf("%-10d", 123);	1 2 3 ⬜⬜⬜⬜⬜⬜⬜	Width 10, left aligned

output statement	output result	explanation
printf("%f", 1.23456789);	1 . 2 3 4 5 6 8 ⬜⬜	6 decimal places
printf("%10.3f", 1.23456789);	⬜⬜⬜⬜⬜ 1 . 2 3 5	3 decimal places
printf("%-10.3f", 1.23456789);	1 . 2 3 5 ⬜⬜⬜⬜⬜	Left Align
printf("%.3f", 1.23456789);	1 . 2 3 5 ⬜⬜⬜⬜⬜	Display only decimal places

Lab: Four basic arithmetic calculation

- stores 20 and 10 in variables x and y , calculates x+y , xy, x * y, x/y, stores them in variables, and prints these variables on the screen .

Sum of two numbers : 30
Difference between two numbers : 10
Product of two numbers : 200
The share of two : 2

Solution

```c
// Program to calculate addition, subtraction, multiplication and division between integers
#include < stdio.h >

int main( void )
{
    int x; // variable to store the first integer
    int y; // variable to store the second integer
    int sum, diff, mul , div; // Variables that store the results of operations between two
integers

  x = 20; // store 2 in variable x
  y = 10; // store

  sum = x + y; // Store the result of (x + y)
  diff = x - y; // Store the result of (x – y)
  mul = x * y; // variable Store the result of (x * y)
  div = x / y; // Store the result of  (x / y)
```

Solution

```
printf ( " Sum of two numbers : %d\n" , sum); // Print the value of sum
printf ( " Difference between two numbers : %d\n" , diff); // Print the value of diff
printf ( " Product of two numbers : %d\n" , mul ); // variable Print the value of mul
printf ( " Quotient of two numbers : %d\n" , div); // Print the value of div

return 0;
}
```
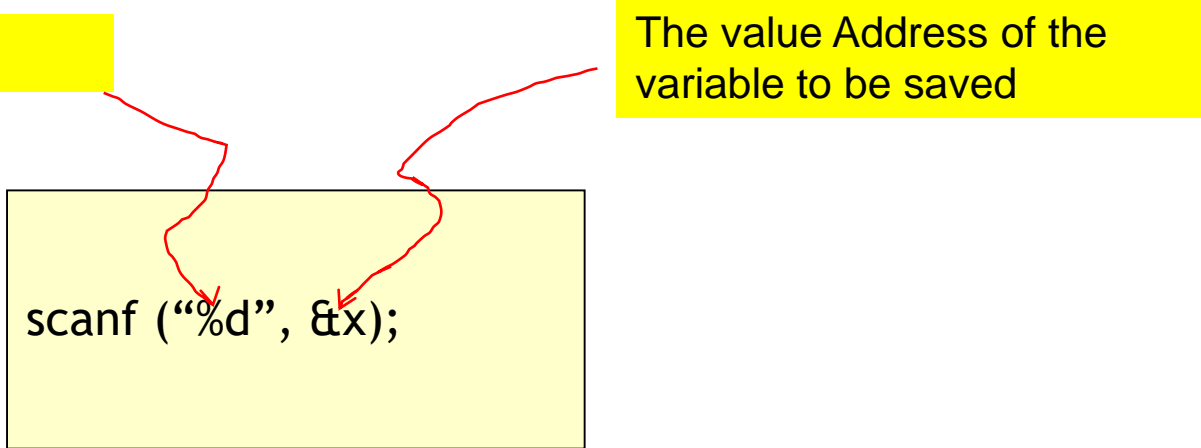
scanf ()

- Gets a value from the keyboard and stores it in a variable .
- Requires the address of the variable .

Format specifier

The value Address of the variable to be saved

scanf ("%d", &x);

Why do I need an address?

- when we buy a product on the Internet and have it delivered to our home, we have to tell the shopping mall our address .

The address of the variable

was sent with & attached!

Shipping is possible as long as you

have an address.

& x

The & operator calculates
the address of a variable .

Format specifiers for scanf ()

- mostly It is similar to printf () .

format specifier	meaning	yes
%d	Enter a decimal integer	`scanf("%d", &i);`
%f	Enter a real number of float type.	`scanf("%f", &f);`
%lf	Enter a real number of type double.	`scanf("%lf", &d);`
%c	Enter one character.	`scanf("%c", &ch);`
%s	Enter a string.	`char s[10]; scanf("%s", s);`

Not studied yet!
Don't worry too much!

When entering a mistake Things to note

- Float type uses %f

```
float ratio = 0.0;
scanf ( "%f", &ratio );
```

```
double scale = 0.0;
scanf ( "%lf", &scale );
```

wrong If you use it, an error will occur .

- double type uses %lf

scanf()

When using scanf () What if I get a compilation error ?

⚠ C6031   반환 값이 무시되었습니다. 'scanf'.

❌ C4996   'scanf': This function or variable may be unsafe. Consider using scanf_s instead. To disable deprecation, use _CRT_SECURE_NO_WARNINGS. S

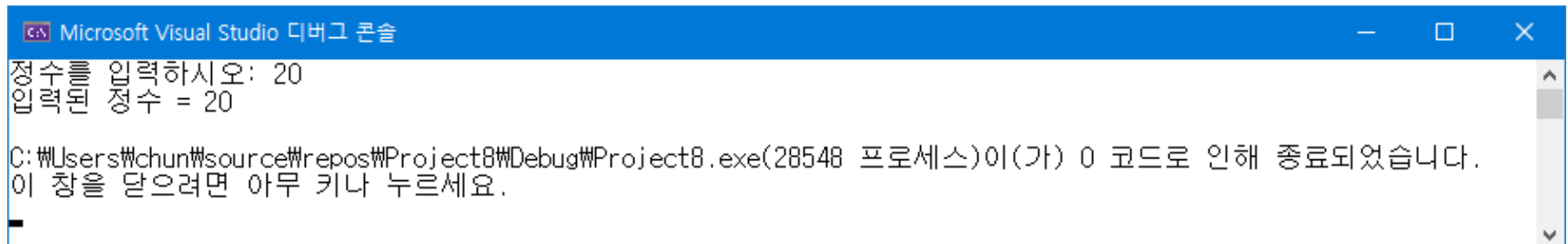scanf ( ) This means that you should use scanf_s () because it is unsafe .

if If this error occurs, define _CRT_SECURE_NO_WARNINGS as shown on the following page .

Integer Accepting program

```c
#define _CRT_SECURE_NO_WARNINGS
#include < stdio.h >

int main( void )
{
    int x; // variable to store integer
    printf ( " Enter an integer : " );
    scanf ( "%d" , &i );
    printf ( " Integer entered = %d \n" , i );
return 0;
}
```
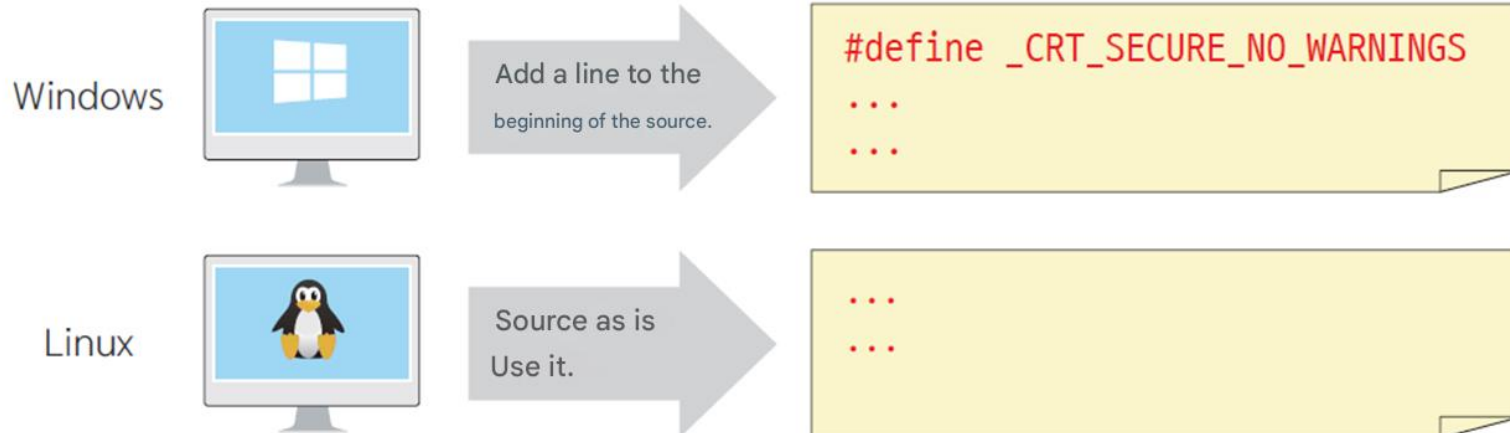

Microsoft Visual Studio 디버그 콘솔

정수를 입력하시오: 20
입력된 정수 = 20

C:\Users\chun\source\repos\Project8\Debug\Project8.exe(28548 프로세스)이(가) 0 코드로 인해 종료되었습니다.
이 창을 닫으려면 아무 키나 누르세요.

scanf () Summary

Windows — Add a line to the beginning of the source.

```
#define _CRT_SECURE_NO_WARNINGS
. . .
. . .
```

Linux — Source as is Use it.

```
. . .
. . .
```

Addition Program #2

- Let's get input from the user .

```
Enter the first number : 10
Enter the second number : 20
Sum of two numbers : 30
```

Algorithm


```c
#include <stdio.h>
int main(void)
{
    int x, y;

    scanf("%d", &x);

    scanf("%d", &y);

    sum = x + y;

    printf("d", sum);

    return 0;
}
```
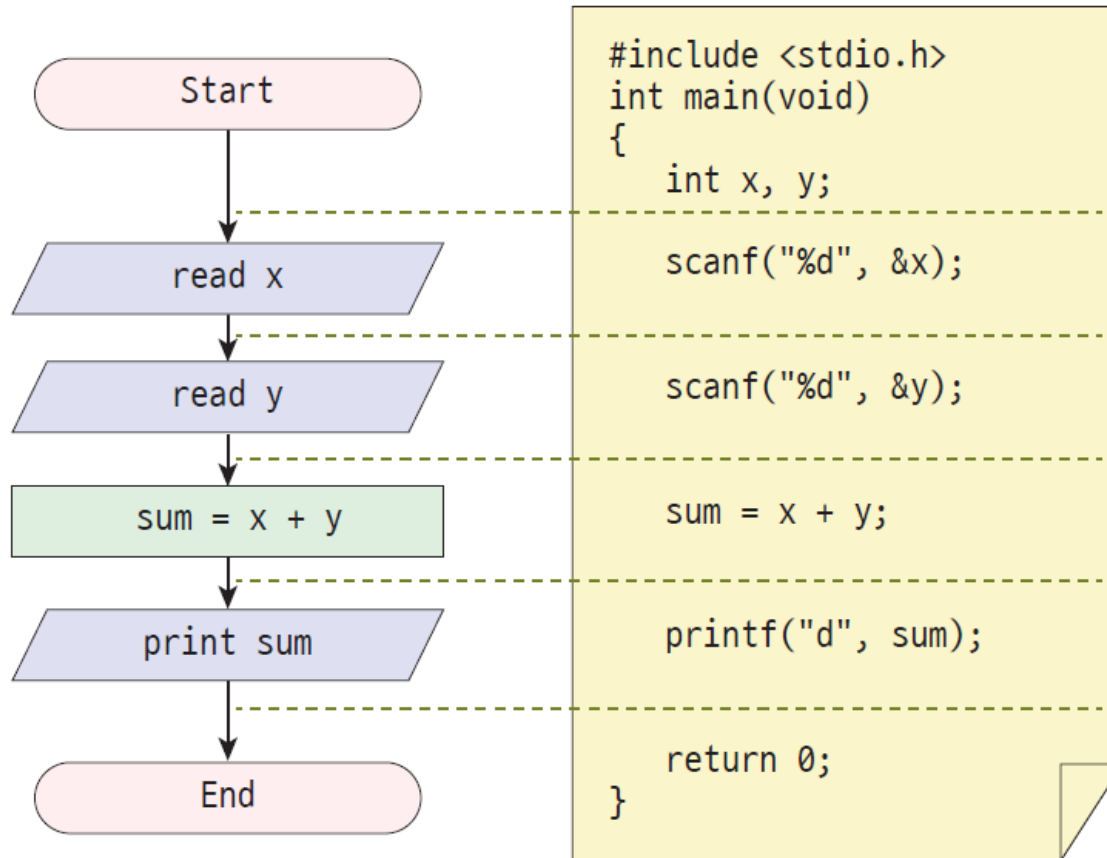
Second addition program

```c
// Calculate and output the sum of
#define _CRT_SECURE_NO_WARNINGS

#include < stdio.h >

int main( void )
{
 int x; // variable to store the first integer
 int y; // variable to store the second integer
 int sum; // Variable to store the sum of two integers

 printf ( " Enter the first number : " ); // Output input guidance message
 scanf ( "%d" , &x); // Receive an integer and store it in

 printf ( " Enter the second number : " ); // Output input guidance message
 scanf ( "%d" , &y); // Receive an integer and store it in

sum = x + y; // Add two variables .
 printf ( " Sum of two numbers : %d" , sum); // Print the value of sum in decimal

 return 0; // return 0 to the outside
}
```
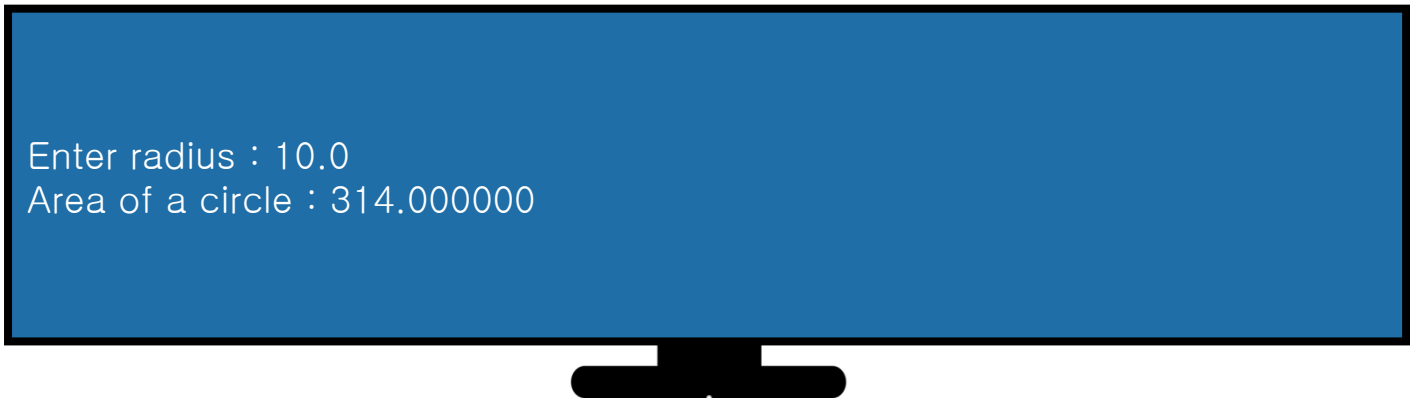
Circle Area Calculation Program

- It receives the radius of a circle from the user , calculates the area of the circle , and then displays it on the screen .

Enter radius : 10.0
Area of a circle : 314.000000

Circle Area Calculation Program

```c
#define _CRT_SECURE_NO_WARNINGS
#include < stdio.h >

int main( void )
{
    float radius; // radius of the circle
    float area; // area

    printf ( " Enter the radius : " );
    scanf ( "%f" , &radius);
    area = 3.14 * radius * radius;
    printf ( " Area of the circle : %f\n" , area);

    return 0;
}
```

Currency conversion program

- Let's write a program that converts the won input by the user into dollars and outputs it .

Enter the exchange rate : 1400
Enter the amount in Won : 1000000
1,000,000 Won is equal to 714.285,714 US Dollars .

```c
/* Program to calculate exchange rates * /
#define _CRT_SECURE_NO_WARNINGS
#include < stdio.h >

int main( void )
{
    double rate; // won / dollar exchange rate
    double usd ; // dollar
    int krw ; // Won is declared as an integer variable

    printf ( " Enter the exchange rate : " ); // Input guidance message
    scanf ( "%lf" , &rate); // Input exchange rate

    printf ( " Enter the amount in Korean Won : " ); // Input guidance message
    scanf ( "%d" , & krw ); // Enter the amount in Korean Won

    usd = krw / rate; // Convert to dollar

    printf ( " %d won is %lf dollar .\n" , krw , usd ); // Print calculation result

    return 0; // Return the function result value
}
```

Program to calculate average

- Write a program that receives three double type real numbers from the user , calculates the sum and average , and displays them on the screen .

Enter 3 numbers : 10.2 21.5 32.9
Total =64.60
Average =21.53

Program to calculate average

```c
#define _CRT_SECURE_NO_WARNINGS
#include < stdio.h >

int main( void )
{
    double num1, num2, num3;
    double sum, avg;

    printf ( " Enter 3 real numbers : " );
    scanf ( "% lf % lf % lf " , &num1, &num2, &num3); // Input 3 real numbers

sum = num1 + num2 + num3;
avg = sum / 3.0;

    printf ( " total = %.2lf\n" , sum); // Display decimal point with 2 digits
    printf ( " average =%.2lf\n" , avg);

    return 0;
}
```

Q & A