Samsung
Innovation
Campus

Chapter 5.

C Programming

Coding&Programming

Enabling People

SAMSUNG

CONTENTS

MISSION FOR EVERYDAY

MISSION

NOISSIM

DAY 1

First Encounter with C Language!

- You can find out the structure and features of C language, and install Integrated Development Environment (IDE).
- You can write and compile source code.

READY

DAY 2

What is Source Code?

- You can understand the principle of writing source code.
- You can apply main() function and the function of 'include'.

READY

DAY 3

Oh! I can Display on Screen!

- You can print execution result on screen.

READY

DAY 4

Let's Name the Data!

- You can find out how to use variable, and program by declaring variable

READY

DAY 5

A Large Number of Data should be Lined UP!

- You can find out character defined data type and define several characters.
- You can save and use character by using array.

READY

DAY 6

Let's Play with Variable!

- You can do input/output (I/O) programming using variable.
- You can convert data type of variable.

READY

DAY 7

What If There's Array in Array?

- You can find out the principle of two-dimensional array and program it.

READY

DAY 8

Making File at My Disposal

- You can find out the method and mode to generate file and write a program.

READY

DAY 9

Let's Bring Out Data from File!

- You can find out and use the method to search file, extract data, and save at variable.

READY

DAY 10

Add, Subtract, Multiply, and Divide

- You can find out numerical data type and write a program using type conversion.

READY

DAY 11

True or False?

- You can find out and use the types and the meanings of comparison operator.

READY

DAY 12

Select by Condition

- You can find out and use conditional statement that decides and operates by condition.

READY

DAY 13

When There are More Conditions?

- You can find out and use nest-

READY

DAY 14

Think Logically

- You can find out logical operator and use it in conditional statement.

READY

DAY 15

Do, Do Again, and Keep Repeating

- You can find out and use while loop among repetition statements

READY

DAY 16

Repeat Hard as Much as Set Number!

- You can find out and use for loop among repetition statements.

READY

DAY 17

Judy's Project

- You can solve team project with

READY

DAY 18

My Own Project

- You can solve your own project with C language programming.

READY

BOOK STRUCTURE

'Judy's C Note' is composed of total 18 DAY chapters of 3 STEP. It takes a format of storytelling solving problem according to the story so the students can learn C language in fun way, and it is divided into 4 main contents including 'Learning Contents', 'Exercise', 'Learn More!', and "TELL ME SOMETHING'. The entire structure of this book is as following.

Prologue
The need for learning C programming is developed in cartoon format

Introduction for each chapter
Story related to the learning contents for each chapter(DAY) is introduced in cartoon format

Learning Contents
Learning contents are composed in format of copying the provided source code of a program.

Exercise
Problems to check of the students are provided. Problems are provided by the level of difficulty from the low level to high level so the students can improve the problem solving ability.

Learn Morel
Contents which have not dealt in the Learning Contents are provided in depth, and contents required to know when programming with C language are introdueced.

TELL ME SOMETHING
Career exploration contents are provided to experience dream coming true indirectly and to improve the spirit of challenge though real stories of C language and experiences of 9 mentors from Samsung Electronics

* In the books for instructors, 'commentary' about the parts students feel difficult and 'additional explanation' and 'answers' about core code are added.

HOW TO STUDY

01 The book is produced for Orwell DevC++(5.4.2) version. We inform you in advance that the result could be different from the book if you use different version.

02 Each lesson is connected by DAY so the class should proceed in order.

STEP 1 → **DAY 1 - DAY 7**
It consists of contents from installation of program to input and print of data which is the basic of learning program. In STEP 1, it is important to understand the overall structure of C language programming.

STEP 2 → **DAY 8 - DAY 13**
It consists of contents about how to use file and solve problem using conditional statement. In STEP 2, it is important to show how to solve program according to various conditions with programming.

STEP 3 → **DAY 14 - DAY 18**
In STEP 3, finding repetitive pattern and drawing logical thought to express these with programming are important. You should try to solve problem together through project class based on what you learned earlier.

03 Coding of source code should be done after finding the process for problem solving.

04 You should have time to share your source code and thoughts about why you program as the way you did since programming can be written creatively.

05 DAY 17 Project is a team assignment for problem solving, and can proceed by dividing functions among the team members (one function per one member). It provides the opportunity of collaboration through the process of sharing thoughts of each other, writing program for each function, and completing the result. The project let you experience in advance that programming to solve a big project is progressed through collaboration.

06 DAY 18 Project is a step to write a program by designing your own project lastly after improving ability to solve problem. Try to stretch your ideas with a program.

C language which codes source in CUI may make the students feel boring so the instructor should give attention to lead the class interestingly by giving examples from our surroundings.

HOW TO TEACH

For instructor

DAY	Title	Objectives	Contents	Instructor's Activities
1	**First Encounter with C Language!**	· ou can find out the structure and features of C language, and install Integrated Development Environment (IDE).	· Introduction to C language and installation process of Integrated Development Environment (IDE) and orwell devc++	· Explain the basic structure of C programming by taking example of source code of C language · Explain the features of C language and the need for IDE. · Install orwell DEVc++ together.
2	**What is Source Code?**	· You can understand the principle of writing source code. · You can apply main() function and the function of 'include'.	· Basic principle of writing source code of C language · Main()function, sequential processing method, use of annotation, function	· explain the principle of writing source code of C language and the function of #include<stdio.h>. · After completing and compiling the source code, check the execution result. · Find the cause of error and explain how to modify.
3	**Oh! I can Display on Screen!**	· You can print the execution result on screen.	· printf( ) function · pecify various output formats by combining escape character, format specifier, and constant.	· Explain basic method of using printf() function, which prints the result on screen. · Check the execution result of specifying various output formats.
4	**Let's Name the Data!**	· You can find out how to use variable, and program by declaring variable.	· Variable and data type · Variable declaration and variable assignment · Memory addressing of variable	· Explain the meaning of variable · Guide to declare integer variable
5	**A Large Number of Data should be Lined UP!**	· You can find out character defined data type and define several characters. · You can save and use character by using array.	· Data type specifying character · Saving a number of characters and saving character using array	· Compare and explain declaration of character variable and integer variable. · Explain the difference between character and char acter string, and let students find the inconvenience by saving several characters. · Explain the need and structure of array, and guide to save character string.
6	**Let's Play with Variable!**	· You can do input/output (I/O) programming using variable. · You can convert data type of variable.	· Writing program using scanf(), printf() function by declaring variable · Converting data type of varable · Two-dimensional array	· Convert data type of variable within scanf(), printf() function, and check the result through input/output program to compare. · Compare and explain one-dimensional array and
7	**What If There's Array in**	· ou can find out the principle of two-dimensional array and program it.	· Writing a program for input of character string including long sentence and blank space using array	· Compare and explain one-dimensional array and two-dimensional array. · Let students give presentations about what is convenient about using two-dimensional array. · Explain the concept of file, the reason of using
8	**Making File at My Disposal**	· You can find out the method and mode to generate file and write a program.	· You can find out the method and mode to generate file and write a program. · Writing a program to generate fil · Writing a program to write contents after generating file	· Explain the concept of file, the reason of using file, and the procedure. · Explain the file pointer procedure and file related library functions (fopen(), fclose(), fprint()). · Check if the students can open file and print it on screen.
9	**Let's Bring Out Data from File!**	· You can find out and use the method to search file, extract data, and save at variable.	· Writing a program to search and print the contents of file · Writing a program to read contents of file at one time and print. · Writing a program to search the contents of file and use as internet search word.	· Explain how to use fscanf() function and format specifier. · Check if the students can read the contents of file by applying file input/output function and search.

DAY	Title	Objectives	Contents	Instructor's Activities
10	**Add, Subtract, Multiply, and Divide**	• You can find out numerical data type and write a program using type conversion.	• Writing a program using the four fundamental arithmetic operations • Writing a program of operation using type conversion between	• Explain about the type of arithmetic operator, declaration of real type variable, and data assignment. • Compare and explain the result of numeric operations of integertype, real type, and mixed type.
11	**True or False?**	• ou can find out and use the types and the meanings of comparison operator.	• Writing a program to distinguish true and false using comparing operator • Writing a program to compare the size of numbers	• Explain the type and the meaning of comparing operators, and especially the difference between '==' and '=' by comparing them. • Explain that the result of comparing operator is expressed as true and false, and is used to distinguish the true/false of conditional statement (if statement) and repetition statement (while loop and for loop)
12	**Select by Condition**	• You can find out and use conditional statement that decides and operates by condition.	• Writing a program using if • Writing a program using if-else	• Explain by taking an example of a case applying different processing method by condition. • Explain the structures and the considerations of if statement and if-else statement. • Let the students to write a sentence to
13	**When There are More Conditions?**	• You can find out and use nested if statement.	• statement • Writing a program to find maximum/minimum value with nested if statement	• Explain about the structure of nested if statement • Let the students write the nested conditional statement using a brace ({}) to nested if statement.
14	**Think Logically**	• You can find out logical operator and use it in conditional statement.	• Writing a program ending PC using logical operator • Writing a program to check pass word using logical operator • Writing a program to check leap year using logical operator • Bitwise logical operator	• Explain the type and the meaning of logical operator. • Explain by taking an example of program deciding true and false using a number of conditions. • Let the student write a program using logical operator required for problem solving properly. • Explain the type and the meaning of bit and bitwise logical operator and let the student use these properly in the process of problem solving.
15	**Do, Do Again, and Keep Repeating**	• You can find out and use while loop among repetition statements.	• Repetition statement (while loop), increment and decrement operator • Endless loop and breaking out of endless loop (break, continue, goto) • Writing a program to check if password matches • Writing a program to match last name and first name from the list file	• Explain about repetition statement, loop, the need of repetition statement, the types and the meanings of increment and decrement operator, and how to use it. • Explain the structure of repetition statement (while loop) and the case of developing endless loop. • Explain about jump statement related to breaking out of endless loop (break, continue, goto)
16	**Repeat Hard as Much as Set Number**	• You can find out and use for loop among repetition statements.	• Repetition statement (for loop) • Writing a program to convert capital and small letters of alphabet • Writing a program to print magic square using repetition statement	• Explain the structure of repetition statement (for loop). • Let the students compare the for loop and the while loop and find out the difference. • Check if the students can write a program using repetition statement (for loop).
17	**Judy's Project**	• You can solve team project with C language programming.	• Writing a program to register and search stray dog • Writing a program to make a list of stray dog	• Explain the considerations so it can be a project satisfying the given conditions. • Let the students communicate and share freely about the function of writing program for problem solving through team activity. • Let the students complete the project by cooperating with team.
18	**My Own Project**	• You can solve your own project with C language programming.	• Writing a program to solve your own special project	• Let the students communicate and share freely about the function of writing program for problem solving through team activity. • Let the students perform their own project individually based on shared information

CHARACTERS

Judy

She is a curious and easy-going high school student. As she participated in the C language class by chance, she tries to study enthusiastically but it's not easy. At first, she makes careless mistakes sometimes when programming, but she shows an excellent ability to understand as she studies in depth due to her habit of thinking focused on principle.

David

He is Judy's friend in middle school. He is just good at using program, such as photoshop, but is mistaken for being good at programming. He starts studying C language with Judy on an impulse, but he gradually finds enjoyment in C language. He is originally apathetic, but as he meets several mentors, he finds his own solution with a positive attitude. He is quick to understand and has a good ability to solve problem.

Charlie

He is David's cousin. He's so-called the genius of C language programming. He is still young, but has a hobby of making educational robot and moving it with programming. With an outstanding programming skill, he teaches C language to Judy and David.

Nick

He is a young farmer who runs farm in a suburb. He manages the farm with a program he made. He knows well about control statement.

Jessie

She is a young artist who creates the media art using the program such as processing. She uses repetition statement a lot for making art works.

SYNOPSIS

One day, Judy finds out an interesting fact about programming: Clanguage is a programming language which is the matrix of smartphone OS, and learning Clanguage will let her be able to make mobile games that she enjoys on her own. Besides, Judy always wanted to make a program for community service.

Judy who wants to learn Clanguage right away goes to visit her friend in middle school, David. David knows the best about computer among Judy's friends, but he is disconcerted when Judy suddenly tells him to teach Clanguage to her. It's because being good at using the computer program and being good at programming are different. Instead, he introduced his cousin, Charlie, who is called the genius of programming to her. Charlie taught the basics of programming with Clanguage to them, and Judy and David get to know about basic structure and rules of Clanguage, data, variable, and array, etc.

After learning the basics of programming, they go to visit Nick with an introduction from Charlie. Nick is a young farmer who uses a quadcopter for farming. Judy and David are highly inspired by Nick, actively using Clanguage to farming which does not seem relevant to programming. Especially, David, who were indifferent to Clanguage saying it's difficult, finds enjoyment in it as he makes a program of which the result changes by condition. Judy and David who learned control statement from Nick meet with another mentor, Jessie. Jessie, a role model that Judy dreams of, is a charming and confident artist who loves her work. Jessie uses conditional statement of Clanguage in an art field called Media Art. When she shows making various art works with Clanguage, Judy and David are amazed by the infinite potentials of programming.

Completing the practice of programming with three mentors, they gained confidence in Clanguage. Now, Judy is trying to make the program she needed on her own. David is also highly motivated because he learned that Clanguage is not just difficult but it can be used in real-life in any degree. We look forward to what wonderful programs that Judy and David will make using Clanguage and how they will surprise their friends.

TELL ME SOMETHING

The real story of C language from the mentors of Samsung Electronics

Let's meet the employees of Samsung Electronics who are using C language in various fields as Judy and David meet with three mentors (Charlie, Nick, and Jessie).

Let's dream of your future that you will create through the stories of mentors for SIC friends!


When on earth is Judy coming?
(Turning head)
David! Sorry I'm late!
Hello!
hehe
How could you come 20 minutes late after asking for my help?
glaring
(Running)
(Running)
(Running)
I'm so sorry, I got up late.
By the way, long time no see.
(Walking)
Since we haven't met after you move, is it in about two months?
Hmm…


Hehe…
Sorry that I haven't been in touch. I should have called you often.
Oh, well… Let's order something to eat first.
COFFEE
By the way, what can I do for you?
(Drinking juice with a straw)
Well, the thing is… I have to do some-thing and I don't know how to do it…
What is it? You've got better grade than me. Is there anything I can do to help you?
Actually… I have to learn C language programming!
(Spouting juice)


…w..what do you mean all of sudden?
Here's napkin..
(Drying off with napkin…)
What is that?
There's a program I want to make, and I heard I can make it by programming with C language.
Umm… I'll tell you later.
(Grinning)
Well, how am I supposed to help you?
I don't know anything about programming.
What? But aren't you good at doing a lot of things with computer? Since you've made a video with our field trip pictures before, I thought you are good at programming, too!
(Sigh…) No, I have used various programs or computer games, but not programming. Isn't it what adults or geniuses do?
Is it? I heard from someone that it's not that difficult so I thought you know how to do it.
Ummm… Come to think of it, there's Charlie.
Charlie? Do you have a friend who knows how to do programming?
No, not a friend… I have a cousin who said he was programming to make some kind of robot in school.


Really? Then…
(A smile)
Please introduce your cousin to me. Okay?
But..he's way too young and…
(Big smile)
Still, he'll be better than me! Please just ask him once, okay?
Well, even if I introduce him to you, I don't know how to ask…
You should come with me, then. Wouldn't it be good if you learn programming with me?
Ahem
What? It's too sudden.
I don't need to learn…
David, I'll surely return your favor later, please? Come with me to introduce him to me, please?
(Bigger Smile)
HoorayAh, okay… I'll make a call first.
…
…Okay! Sure!
You can come any time!


?
Are you sure?
You are not busy?
I'm on a break, not busy at all! And I get to see you, too, after a long time!
Come soon!
Um···
(Heart pounding)
How did it go? Did he say he's busy?
He said to come anytime···
Hooray!
Thank you!
I knew you can make it happen!
(Blushing)
Did···did you?


Well, then hurry up!
Huh?! Right now?!
Why? Do you have another plan today?
No··· I don't···
Then, that's great! Let's go!
hand
Yay! I'm so excited! Let's start C programming lesson!
Wait! Let my hand go!

Chapter 5.

C Programming

STEP 1.

DAY 1.

First Encounter with C Language!

01 What is C Language? C Programming Language

Learning Contents

You don't know what programming is? Then, I should first let you know what programming is. Program made with C language looks like this.

Source code

```
#include <stdio.h> int
main()
{
        printf("Write C language program!\n"); return 0;
}
```

Program written with C language looks like this.

Result

Write C language program!

By the way, who created C language?

It's created by Dennis Ritchie in 1970s. He created it to make an operating system called Unix

He created a language to make a program?!

02 Characteristics of C Language

Learning Contents

C language can make a program using a few keywords and a number of functions defined in advance. I'll tell you briefly about the characteristics of C language. For now, it may sound difficult, but you will understand eventually as you study programming.

- You can do various operations as well as the four fundamental arithmetic operations.
- Special actions that need to be used repeatedly can be made into functions which can continue to be used conveniently.
- You can save and process data easily by using array.
- You can handle memory effectively using pointer.
- You can program easily using various standard input/output functions and library functions.

So what can you do with C language?

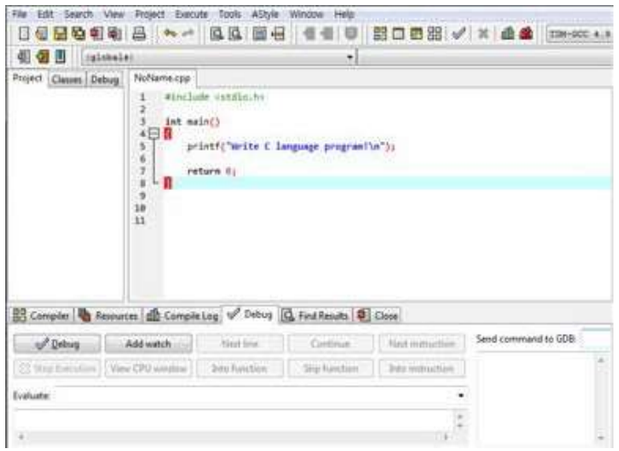
C language is rather easy with simple structure. Because C language shows fast speed and high performance, it can make almost all software including various operating systems, application programs, or even web services.

03

Learning Contents

Integrated Development Environment

For actual programming, we need programming tool. We have to install IDE, a program to make programs with C language. Then, shall we find out about installation?



It's a screen of Orwell DevC++ Integrated Development Environment. It has a various functions.

→ Writing source code → debugging → tracking value of a variable are possible.

What is IDE?

IDE refers to Integrated Development Environment. It is a program composed to write and compile source code required for making a program, and to use function of checking and modifying error easily. Installation Step 3

So, it's a program for making programs. Then, what's a program for making program for making program??

01

Follow

Installing C language program

Now, let's install IDE. Installation of program is very easy so you can do it soon!

❶ Search

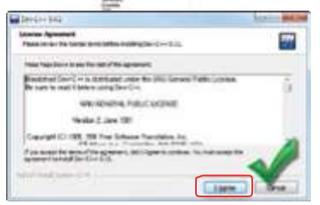
Search Orwell DevC++ from search site, or enter sourceforge.net/projects/orwelldevcpp/ to address bar of web browser and find download page.

❷ Download

Download installation file by clicking on download.

❸ Installation Step 1

Run the installation file and set language to Korean, right?



❹ Installation Step 22

Press Agree button for GPL license.

❺ Installation Step 3

Now, you can select components to install, but since there are still so much you don't know, it is convenient to choose default. Without setting anything, just click Next> button.

❻ Installation Step 4

A screen to set installation path shows up, and select this also as default. Click Install button without setting.

❼ Installation Step 5

Now, the installation is completed. You can do this.

Incidentally, this is not the only program to develop C language, right?

That's right. But for studying, free version of Orwell DevC++(5.4.2) is convenient. Development program for professionals is really expensive.

02 Follow

Setting an Environment

Now then, we will set user environment to program more conveniently.

❶ Setting a language for menu

When running for the first time, set the language as Korean. By any chance, are you more familiar with English?

❷ Setting a font

This part is about setting the font to show the source code. Select the font you like according to personal preference.

❸ Click Next button to move on to the next step.

Click Next button to move on to the next step.

❹Setting a library cache

This part is about setting the cache for the library of functions that are used often. Since you don't have to use it yet, click Next button.

❺ Completing setting

If you click OK button now, setting is all done.

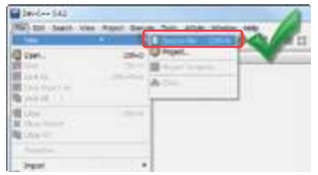
❻ Completing environment setting

Once you complete setting, screen like this appears. Now, preparation has been completed.

03 Follow

Basics of writing the source code

Now let's program in earnest. I'll tell you briefly about how to write and save source code with C language.



❶ Writing the source code 11

After running DevC++, if you click [File]-[New]-[Source File], the source code editing screen shows up.

❷ Writing the source code 2

Write a source code like the source code editing screen shown on the left. Special characters, signs, or even capital and small letters should be written exactly the same.

❸ Saving the source code 1

If you finished writing the source code as it is, now you need to save it. Click [File]-[Save] to open Save File window.

❹ Saving the source code 2

Save the source code as "NoName.cpp" to the desired location. Of course file name can be saved as the name you want.

❺ Checking the source code file

If saving is completed, “NoName.cpp” file is shown as DevC++ icon.
It’s not difficult, is it?

By the way, what is the term, source code, we have been talking about?

It is a code that becomes the source to make program. If you write a source code correctly, execution file which can be executed through the program called Complier is created. The source code becomes a program.

Is it like source code corresponding to blueprint and program being the finished product?

It’s similar..

04 Follow

Compile

Source code writing is finished, but one more step is required to execute the program. The process of converting the source code written with C language to the executable file is called compile.

❶ Compile

While the source code is opened, click [execute]-[compile]. When you get used to this, you can press the shortcut key, F9.

❷ Checking the compile result

Then, compile beings automatically. If written source code is correct, compile will be completed without any special message.

❸ Checking the execution file

If compile is completed normally, the contents of source code saved in “Noname1.cpp”file is made into an executable file, called “noname1.exe.”

Is there any way to check if compile is done well?

The contents or the result of compile can be seen if you press “compile log” tab at the bottom.

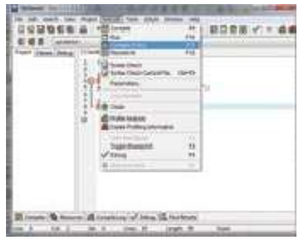
What if compile is not done normally?

Then, you should check the source code, correct the mistakes, and then compile again.

05 Follow

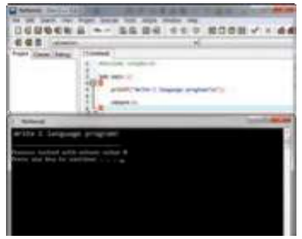
Executing right after compiling

Now compiling is completed, I’ll tell you how to execute the completed program.



❶ Executing after compiling 1

Click [execute]-[execute after compile]. Or, you can press shortcut key, F11, to execute the program. If there is no error in source code, the program will be executed right after compiling.



❷ Executing after compiling 2

Congratulations on your first program coding.

Exercise 01

Now then, shall we find out what we can do with C language? Write the source code as written below, and execute the program after compiling.

Source code

```
#include <stdlib.h>
int main()
{
        system("start http://www.Samsung.com");
        return 0;
}
```

Result

→ The website, http://www.Samsung.com/, is executed in web browser.

Exercise 02

How's that? Isn't that amazing? Let me show you something else. Write the source code as written below, and execute the program after compiling.

Source code

```
#include <stdlib.h>
int main()
{
        system("start http://csr.samsung.com/");
        return 0;
}
```

Result

→ http://csr.samsung.com is executed in web browser.

Exercise 03

This time, let's make a special function to change screen with C language. Write the source code as written below, and execute the program after compiling.

Source code

```
#include <stdlib.h>
int main()
{
        system("start \magnify\ C:\\Windows\\System32\\Magnify.exe");
        return 0;
}
```

Result

→ Magnify.exe (Magnifier for enlarging and reducing screen) program is executed.

What is the Source Code?

01

Learning Contents

The Basics of Writing the Source Code

The source code is written using alphabet, numbers, and special characters representing special meaning. Let me show you an example.

Screen printing of character string

```
/*Source code printing character to screen*/
#include <stdio.h>
int main()
{
    printf("Introduction to C language! \ n");
    printf("Start the C language program! \ n");

    return 0;

}
```

- This is comment statement
- It is used for screen printing
- It is the beginning part of main() function.
- It is executed as the first.
- It is executed as the second.
- It is the ending part of main() function

Result

```
Introduction to C language!
Start the C language program!
```

Oh! But when I execute it, something shows up and disappears soon.

I'll show you how to solve that part.

02

Learning Contents

Making the result screen not disappear

To avoid the result screen disappear right away, you need to add one simple instruction. Do you see the part added to the code written earlier? Compile it and try executing again.

Screen printing of character string

```
/*Source code printing character to screen*/
#include <stdio.h>
#include <stdlib.h>
int main()
{
    printf("CIntroduction to C language!\n");
    printf("CStart the C language program!\n");

    system("pause");
    return 0;

}
```

- This is comment statement
- It is used for screen printing
- It is the beginning part of main() function.
- The source code is written using alphabet, numbers, and special characters representing special meaning. Let me show you
- It is the ending part of main() function

Result

```
Introduction to C language
Start the C language program
```

03

Learning Contents

What is main() function?

The source code of C language can be divided into a few main parts. Among those, the part called main() function is the most critical part.

int main()	➢ C language is executed from the part entered as main().
return ();	➢ When return(); part is executed, main() function ends.

int main()
{
...
return 0;
}
	➢ Thus, main() function, which is the main body of the program written with C language, is mostly made in this wa

By the what, what is function? I think I've heard it somewhere.

Is it the same as the function used in math?

Yes, it's similar. By definition, 'function' means 'the useful thing that they do or are intended to do.' In programming, actually, they play a role of making particular functions to operate. I'll tell you about function later in detail.

04

Learning Contents

Sequential Processing Method

I'll show you how to write the contents of main() function. In C language, one small processing unit is called statement, and you must put semicolon (;) at the end of each statement.

C language, fundamentally, processes the statement of input source code one at a time from the top.

printf(“Introduction to C language! \ n”); printf(“Start the C language program! \ n”);	➢ It is executed first because it is on the top line. ➢ It is executed second because it is on the next line.

The printf functions to show the contents on the result screen. I’ll tell you in detail a little bit later. Don’t forget to put semicolon (;) at the end of all instructions.

Why are there so many parentheses and marks?

It is written as such to distinguish the elements required in each instruction. There are a few rules in special characters. The characters, such as (), {}, [], “”, and ‘’, must be used in pairs to indicate the meaning set in advance. When you show one alphabet letter, you have to use single quotation marks (‘’); when you show character string, you have to use double quotation marks (“”). (‘A’, ‘c’, “ABC”, “Hello”)

You should always use semicolon (;) at the end of a line of instruction to show the instruction is over.

It is not that different from how we use those when we write.

05

Learning Contents

Writing source code in an easy way to see

In many cases, it is not easy to recognize the source code when it gets long and complex. I’ll tell you how to write the source code in an easy way to see.

You can space, indent, or change line any time within the block or in the middle of statement. If you utilize this well, you can write source code in a little easier way to read.

int main(){ printf(“Introduction to C language! \ n”); return 0;}	➢ If you don’t change line, it looks complicated like this.

int main()
{
printf(“Introduction to C language! \ n”);
return 0;
}
	➢ If you change line properly, it is much easier to recognize.

Only if you use spacing (Space Bar), indentation (Tap), and line change (Enter) well, you can write a neat source code with less errors.

When you write source code, do you have to use only small letter?
It seems fine to use capital letter, right?

C language recognizes capital letter and small letter of alphabet as different letters. Thus, if you write “printf” as “PRINTF”, it is not executed.

06

Contents

The use of comment

To write source code easy to see, you have to make good use of comment. If you take a look at the example we learned earlier, do you see the part written as /* */? That is comment.

The contents written in /* and */ are the part that is not executed. It is a kind of memo.

/* Source code printing character to screen*/	➢ This part is skipped without being executed.
/* Source code printing character to screen*/	➢ Comment can be written over a number of lines.

If you mark what content does the source code show with comment, it is helpful when you read it later or when you modify. If you use only one line for comment, you can put // in front of the sentence.

07

Contents

What is include?

Now then, I have only to explain #include part shown at the top in the example we made earlier, right? This part is a little difficult.

#include <stdio.h>	➢ This instruction means to remember the header file, stdio.h, in which various useful functions, such as printf() function, are collected by reading it before compiling. The work of reading the contents of other file to be referred as such is called include. ➢ The line with # mark is read first before processing other codes by a special part called preprocessor. The line with # mark should be entered only in one line without semicolon at the end or line change.

To interpret English sentence, you need to prepare English dictionary. Is it in this manner?

Yes, it is if figuratively speaking. You can just understand it as an instruction to prepare so “stdio.h” file can be referred in library folder defined in advance.

Exercise 01

Now then, let's find out about the considerations as you write the source code on your own. If you enter the following source code, an error will occur. Guess which part is wrong.

Source code

```
#include <stdio.h>
int main()
{
        printf("Hello World!');
        return 0;
}
```

- It is a program to print a sentence, 'Hello world!'.
- But when executed, it is not compiling due to error.

▶ TIP

Carefully examine the special characters used in sentence.

Details of error printed when compiling

row	colum	Unit	message
4	10	C:\Users\user\Desktop\csw\02\testerror1.cpp	[Warning] missing terminating " character [enabled by default]
4	3	C:\Users\user\Desktop\csw\02\testerror1.cpp	[Error] missing terminating " chracter In
		C:\Users\user\Desktop\csw\02\testerror1.cpp	function 'int main( )':
5	4	C:\Users\user\Desktop\csw\02\testerror1.cpp	[Error] expected primary-expression before 'return'

Answer

Exercise 02

This time, also, I will show you a source code with error. Can you tell which part is wrong?

Source code

```
#include <stdio.h>
int main()
{
        printf('Hello World!');
        return 0;
}
```

- It is a program to print a sentence, 'Hello world!'.
- But when executed, it is not compiling due to error.

Details of error printed when compiling

row	colum	Unit	message
4	10	C:\Users\user\Desktop\csw\02\testerror1.cpp	[Warning] character constant too long for its type [enabled by default]
	24	C:\Users\user\Desktop\csw\02\testerror1.cpp	
			In function 'int main( )':
4		C:\Users\user\Desktop\csw\02\testerror1.cpp	[Error] invaild conversion from 'int' to 'const char*'[-fpermissive]
1	0	C:\Users\user\Desktop\csw\02\testerror1.cpp	In file included from C:\Users\user\Desktop\csw\02\testerror1.cpp
391	15	c:\program files\dev-cpp\mingw64\x86_64-w64...	[Error] initializing argument1 of 'int printf(const char*, ...)' [-fpermis- sive]

Answer

Exercise 03

This one is also incorrect source code. Check which part is wrong carefully.

Source code

```
#include <stdio.h>
int main()
{
        printf("Hello World!")
        return 0;
}
```

- It is a program to print a sentence, 'Hello world!'.
- But when executed, it is not compiling due to error.

▶ TIP

What did I say about what must be placed at the end of the instruction sentence?

Details of error printed when compiling

row	colum	Unit	message
		C:\Users\user\Desktop\csw\02\testerror1.cpp	In function 'int main( )':
5	3	C:\Users\user\Desktop\csw\02\testerror1.cpp	[Error] expected ';' before 'return'

Answer

Exercise 04

This one is also similar problem. It seems right this time…what is missing?

Source code

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

- It is a program to print a sentence, 'Hello world!'
- But when executed, it is not compiling due to error.

▶ TIP

To use printf() statement, you need something to refer to. Depending on the cases, compiling error may not occur. It's because compiler include the necessary header file to printf() on its own. Compare with the source code written earlier, and find which part is included by computer.

Details of error printed when compiling

row	colum	Unit	message
		C:\Users\user\Desktop\csw\02\testerror1.cpp	In function 'int main( )':
3	24	C:\Users\user\Desktop\csw\02\testerror1.cpp	[Error] 'printf' was not declared in this scope

Answer

Exercise 05

This time, it really seems like a well-made code. But compiling does not work again. What's the problem?

Source code

```
#include <stdio.h>
int main()
{
      Printf("Hello World!");
      return 0;
}
```

· It is a program to print a sentence, 'Hello world!'

➜But when executed, it is not compiling due to error.

▶TIP
Big one… Small one… This is the hint!

Details of error printed when compiling

row	colum	Unit	message
		C:\Users\user\Desktop\csw\02\testerror1.cpp	In function 'int main( ):'
4	24	C:\Users\user\Desktop\csw\02\testerror1.cpp	[Error] 'printf' was not declared in this scope

Answer

Exercise 06

What about code like this? Will it be compiled and executed normally? Try it yourself.

Source code 1

```
#include <stdio.h>
int main(){printf("Hello World!"); return 0;}
```

Source code 2

```
#include <stdio.h>
int main(){
printf("Hello World!");
return 0;}
```

Source code 3

```
#include <stdio.h>
                int                main()                {

                                printf("Hello World!");
return 0;

}
```

Answer

Learn more! What is function?

The need of function

As we do C programming so far, we have used printf() and main(), introduced in page 23, and we'll learn more about function.
In programming, why do we make functions and use these?

01 For the same repetitive works, it is convenient to make and use function.

Shall we take an example of remote controller? When we watch TV, we use remote controller to turn ON/OFF as well as to set the TV in the form we want by using the function adjusting channel, volume, and screen details. If we save the functions that are used often in remote controller and use these by pressing the button of the corresponding function, it will be very useful since we don't have to do the same motion every time.

Like this, the procedure of using the saved operation in need corresponds to function in program. In function, the part to process can be written in advance and be used whenever it is necessary by calling function name; thus, repetitive and frequently used procedures are written and used as function.

02 Just like a program to carry out the function of ON/OFF, channel, volume, and screen adjusting of remote controller, function is a small unit of program to solve one problem.

03 In C program, the functions required for problem solving can be called several times into a function, called main, and be used; different types of functions can be connected to exchange data with each other and are reusable.

04 The function returns the resulting value no matter what after execution. You remember wring return statement. The return statement notifies the end of function and returns the value behind.

As a complex and difficult problem is processed by dividing into small units, function can be modularized into the small units of programs within a program; thus, a complex problem can be solved easily.

The types of function

Function can be classified into two main types: standard library function and user defined function.

01 Standard library function, including printf(), scanf(), getchar(), fscanf(), feof(), and fopen(), refers to a function which is provided by writing in advance in C program. The standard library function must be included before compiler supports and uses.

02 In writing a program, there is a function used by defining out of necessity by user, and it is called the user defined function. To use the user defined function, first you need to define the definition of function in the form below.

```
Function data type function name (factor1, factor2…)
{ Data processing instruction return
  return value; }
```

```
int sum3num(int a, int b, int c)
{
      int sum =a + b + c;
      return sum3num;
}
```

Let's write a program to double the sum of two numbers using the user defined function.

	1	#include <stdio.h>	
Global variable declaration	2	int sum;	➢ sum is declared before main function and used in all areas of the program.
Function call	3	void s();	➢ User defined s function declaration
main function	6	int main()	
	7	{	
	8	int a, b;	
	9	printf("Enter two numbers.\n");	➢ Local variable a, b declaration
	10	scanf("%d,%d", &a, &b);	
	11	sum = a + b;	
	12	printf("The sum of two numbers %d %d is %d.\n", a, b, sum);	➢ A global variable, sum, is not re-de clared in main() function and used.
	13	s();	
	14	printf("Double of the sum is %d.\n", sum); return 0;	
	15	}	➢ When s function is called, it is moved to row 20 and the sum is multiplied by 2; then, it returns to where it was called by row 23 and printf function of row 16 is executed.
	16	void s()	
	17	{	
	18	sum = sum * 2;	
	19	}	
	20		
	21		➢ A global variable, sum, is also used in s function
	22		
	23		

❶ Declare the variable to be used at the beginning part of the function first, and then write the operation code.
Function declared first can be used in function using after declaration, but function declared later cannot be used in function before declaration.

❷ Local variable can only be used in a certain declared area, and when it departs from the function while it is used declaring in function, it disappears from the memory.

❸ Global variable can be used outside the function, and can be used in all areas. When executing the program,

❹ it always remains in the memory.

Now I can put it up on the screen

01

Learning Contents

Printing on the screen

You practiced using printf() statement, right? The printf() is an instruction to print the contents in the parenthesis as a character string on the screen.

Source code

```
#include <stdio.h>
Int main()
{
        printf("print");
        return 0;
}
```

- It is a source code to print the character string, “print,” on the screen

Result

```
print
```

02

Learning Contents

Printing with a line change

What should be done to change the line of sentence to be printed on the screen with printf()? Well, it can be done as following.

Source code

```
#include <stdio.h>
int main()
{
        printf("CStart the C language program! \ nprint\n");
        return 0;
}
```

- If you use the mark to change the line of character string, ‘ \ n’, you can change the line.

Result

```
Start the C language program!
print
```

Can I just use printf() twice to change the line?

If you don’t enter \ n, it is printed in one line even though you wrote each with printf().

Exercise 01

Then, what results shall we get if we write the source code in this form? Try it and write the result.

▶ TIP

I know what \ n is but what does \ t mean?

Source code

```
#include <stdio.h>
int main()
{
        printf(“Start the C language program! \ t”);
        printf(“print \ n”);  return 0;
        return 0;
}
```

Result

Exercise 02

This time, let’s find out how to print using format specifier! Write the following source code and try executing it.

Source code

```
#include <stdio.h>
int main()
{
        printf(“%c is character \ n”, ‘A’);
        printf(“%s is character string \ n”, “helloworld”);
        printf(“%d is integer \ n”, 123);
        printf(“%f is fraction \ n”, 11.6);
        return 0;}
```

Result

Format specifier?

Data used for programming has a certain format. For example, there are character, integer, and fraction, etc. The type of such data is called data type.

To print data, you have to use format specifier according to such a data type. You can just remember %c, %s, %d, and %f are the format specifier meaning character, character string, integer, and fraction, respectively.

Exercise 03

What will happen if you use more than two format specifiers for one print instruction? Try it!

Source code

```
#include <stdio.h>
int main()
{
        printf(“%c is character, %d is integer \ n”, ‘A’, 123);
        printf(“%d is integer, %f is fraction \ n”, 5, 11.6); return 0;
}
```

Result

Exercise 04

This time, we’ll learn about numeral system. There are various numeral systems, such as binary notation, and octal notation, etc. .

Source code

```
#include <stdio.h>
int main()
{
        printf(“21 of decimal number is %d. \ n”, 21);
        printf(“21 of decimal number is %o in octal number. \ n”, 21);
        printf(“21 of decimal number is %x in hexadecimal number. \ n”,
        21); 21); return 0;
}
```

Result

When printing some integer value with decimal, ‘%d’ is used, and use ‘%o’ for octal and ‘%x’ for hexadecimal.

Ugh, I have to do math in programming, too…

Exercise 05

Then, shall we use various format specifiers? Try it!

▸TIP

You can just print the contents you want, but if you use format specifier, you can form the contents and format as you want.

Source code

```
#include <stdio.h>
int main()
{
        printf(“ID number: %d \ n”, 131120);
        printf(“Name: %s \ n”, “Eunmi Hong”);
        printf(“bloodtype: %c \ n”, ‘A’);
        printf(“welcome! \ n”);}

}
```

Result

Exercise 06

Now it’s the last problem! To have “Date of Birth: December 23, 2002” as the result below, what instruction should be put in the blank? Make it on your own!

Source code

```
#include <stdio.h>

int main()
{
        printf(“ID number: %d \ n”, 131120);
        printf(“Name: %s \ n”, “Pro Hong”);
        printf(“Bloodtype: %c \ n”, ‘A’);

        Printf(“Date of birth:%dyear%dmonth%dday/
printf(“Welcome! \ n”);

}
```

Result

```
ID number: 131120
Name: Pro Hong
Bloodtype: A
Date of Birth: December 23, 2002

Welcome!
```

DAY 4.

Let’s name the data!

01

Learning Contents

Assignment of variable

Look at this code here. Doesn’t it look different from the code we have been using previously?

Source code

```
#include <stdio.h>
int main()
{
        int a = 65      ;
        printf("%d\n",a);
        return 0;
}
```

Result

65

Umm… I see a. I think this a here mean 65.

That’s right. This is a code to print variable a after making variable called a and assigning the number, 65 to it.

Variable? Assigning?

The term is a little difficult, isn’t it? Saving the number, 65, to computer memory is called assignment. The a is the name of variable, and it plays a role of showing where the number, 65, is saved.

When the variable name is not used

I can’t find where I put it.

When the variable name is used

I can find where I put it easily since there’s name on it

It sounds like an address since it plays a role of showing where it is.

By the way, why is it called variable, not address or name?

Variable mean the value can continue to be changed. For now, 65 is assigned to a, but it can be changed to different value any time.

Tell me more about how to assign.

Do you see the part, 'int a=65', in the code? That is the part assigning value to variable. The meaning of '=' in C language is a little different from '=' used in math. In math, if you write 'a=b', it meana a is equal to b; however, it means to save (assign) the right one (b) to the left one (a). Thus, this '=' is called assign operator.

Then, is there a different mark with the meaning of equal used in math?

Yes. In C language, '==' is used for the mark with the meaning that the left and the right are equal..

'a==b,' means a is equal to b, and 'a=b,' means saving (assigning) b to a, right?

That's right. It is often confused so keep that in mind.

02

Learning Contents

It's called variable because it's changeable!

I said earlier that variable is a changeable number, right? Then, what will happen if the value of data assigned to a variable is changed? Try it yourself.

Source code

```
#include <stdio.h>
int main()
{
        int a = 65;
        printf("%d\n",a);
        a = 56;
        printf("%d\n",a);
        return 0;
}
```

Result

```
65
56
```

The value assigned to a is changed! By the way, what is int in front of variable name?

That is a mark showing the data type of variable. The type of data assigned to variable is called data type. When you make variable, you have to define the form of variable according to this data type.

Shouldn't computer takes care of that?

There are such program languages, but unfortunately, C language

But why is there no data type in the part, 'a=56;' that changes the contents of variable?

It's because data type requires to be written only once when making the variable for the first time.

I see. I should keep in mind what kind of data type is.

Yes. There are a good number of variable types, but remember just two types for now. The int is used when you make a variable to save data in form of integer. The variable to save data in form of character is made with char.

01

Learning Contents

Memory address of variable

If you make variable and assign data, computer saves the data to memory. I'll show you the code to check the memory address of the saved data so try copying it.

Source code

▶ TIP

int a = 65;

Memory value is generally expressed in hexadecimal.

```
#include <stdio.h>

{
        int a = 65;
        printf("%p\n",&a);
        return 0;
}
```

- & is an operator that shows memory address, not a variable value placed behind.
- %p is a format specifier mark that prints memory address in hexadecimal.

Result

```
0022FEBC
```

Exercise 01

I made a code with some problem here. Look at the error code and find which part is wrong; then compile it after fixing the error.

Source code

```
#include <stdio.h>
int main()
{

        a=1;

    printf("%d\n", a);

    return 0;
}
```

Result

```
1
```

Exercise 02

This one is also an incorrect code. Can you see which part is incorrect? Fix the incorrect part and compile it.

Source code

```
#include <stdio.h>
int main()
{
    int a=1; printf("%d\n",
    a);

    int a=2; printf("%d\n",
    a);

    return 0;
}
```

Result

```
1
2
```

Exercise 03

Will this be a code working properly? Compile as it is and try executing it.

Source code

```
#include <stdio.h>
int main()
{
    int a=1;
    int b=2;
    int c=3;
    int d=4;

    printf(" %d\n %d\n %d\n %d\n", a, b, c, d);

    return 0;
    return 0;
}
```

Why does return0; show up twice?

Result

```
1
2
3
4
```

Exercise 04

Compile this code also as it is and try executing it.

▶ TIP

When you make variables with the same data type, it is convenient to make all at once like this

Source code

```
#include <stdio.h>
int main(
{
    int a=1,b=2,c=3,d=4;

    printf(" %d\n %d\n %d\n %d\n", a, b, c, d);

    return 0;

    return 0;
}
```

Result

```
1
2
3
4
```

Exercise 05

This time, it is a code which gives a little different result. What instruction should be put in the blank? Try making on your own

Source code

```
#include <stdio.h>
int main()
{
        int a=1,b=2,c=3,d=4;
        a=b=c=d;
        printf("%d\n%d\n%d\n%d\n",a,b,c,d);
        return 0;
}
```

Result

```
4
4
4
4
```

Exercise 06

What does the result that came out after compiling this code mean? Try writing down.

Source code

```
#include <stdio.h>
int main()
{
        int a=2,b=0;

        b = a == 2;
        printf("%d\n",b);
        b = a == 1;
        printf("%d\n",b);

        return 0;
}
```

Result

```
1
0
```

▶ TIP

As I told you earlier, 'a==b' is a comparing operator that indicates 'a is equal to b'.

If there are assign operator '=' and comparing operator '==' together, which one is executed first?

You can find it out if you try yourself! Try it and write which one is executed first!

DAY 5.

A large number of data should be lined up!

01
Learning Contents

Data type that saves character

The following is a code of program printing after saving alphabet A to variable. It's not much different except the data type.

Source code

```
#include <stdio.h>
int main()
{
        char c = 'A';
        printf("%c",c);
        return 0;
}
```

Result

A

'One' character can be saved to the variable declared as char. To print a character with printf() instruction, the mark in form of %c is used.

One character per one variable is too inconvenient!

Then, how many variables should be used to express a word?

02
Learning Contents

Source code

```
#include <stdio.h>
Int main()
{
        char c1 = 'H',c2 = 'e',c3 = 'l',c4 = 'l',c5 = 'o';
        printf("%c%c%c%c%c",c1,c2,c3,c4,c5);

        return 0;
}
```

Result

Hello

This seems too inconvenient. There must be some other convenient way! Charlie, are you sure you know well?

There is a way to save a long character. Just, you have to use a new concept, called array.

Array?

Array refers to a format which has a number of data in one variable.

03
Learning Contents

Saving character using array

The following is a code that prints the word, "Hello", more conveniently using array. Isn't this much better? Oh, don't forget to use %s, not %c, when printing the array of a character.

Source code

```
#include <stdio.h>
int main()
{
        char c[6]="Hello";
        printf("%s",c);
        return 0;

}
```

Result

Hello

Array is expressing a number of variables as one name. You can think the number in [] when making array is deciding in advance that variable as much as the number in the array will enter. So that means that the number in [] is the size of array.

By the way, why did you set to put 6 variables when hello is 5 letters?

It's because compiler add ' \ 0' (NULL character) which means the end of character automatically when expressing character string. Think that it makes one more array than the number of letter in general when making character sting in this format.

Ugh…it's complicated!

Actually, there's an easier way. It calculates the size of variable automatically when making variable if it is not small. Like this: char c[]="Hello";.

What? You should tell us the easier way first!

But there's a big difference between learning with knowing the principle and without.

04

Learning Contents

Using array

The data saved in variable made with array can be printed as variable itself, but each data in variable can be handled individually. Take a look at the code below.

Source code

```
#include <stdio.h>
Int main()
{
        char c[6]="Hello";

        printf("%c",c[0]);
        printf("%c",c[1]);
        printf("%c",c[2]);
        printf("%c",c[3]);
        printf("%c",c[4]);
        printf("%c",c[5]);

        return 0;
}
```

Result

```
Hello
```

If you put [] behind the variable name of array and put number inside it, you can use the data saved in the order corresponding to the number. You can also print or change the data. The number in [] which is the order of data is called index of array. In C language, index starts from 0, not 1. Computer counts everything from 0.

Exercise 01

This source code does not cause an error, but also does not operate properly. Make it print 'abcd' by modifying the code.

Source code

```
#include <stdio.h>
Int main()
{
        char c[4]={'a','b','c','d'};

                printf("%c%c%c%c",c[1],c[2],c[3],c[4]);

        return 0;
}
```

Result

```
abcd
```

Exercise 02

To get a result, 'SAMSUNG INNOVATION CAMPUS!', fill in the blank of the following source code.

▶ TIP

There are various possible answers, but write the code using the character string.

Source code

```
#include <stdio.h>
int main()
{
        char [ ]={ ];

        printf(" ",c);

        return 0;
}
```

Result

```
SAMSUNG INNOVATION CAMPUS!
```

Exercise 03

This one is also similar problem. Fill in the blank of the following source code to get the result shown below!

Source code

```
#include <stdio.h>
int main()
{
        char c[  ]={"                          "};

        printf("%s\n",c);

        c[6]=  ;

        printf("%s",c);

        return 0;

}
```

Result

```
SAMSUNG INNOVATION CAMPUS
SAMSUNG
```

▶ TIP

The character that signifies null (\ 0) means the end of character in character string.

Exercise 04

This time, the execution result of source code comes out abnormally. Think of the reason and write down.

Source code

```
#include <stdio.h>
int main()
{
        char c[]="ABC";

        printf("%d\n",c[0]);
        printf("%d\n",c[1]);
        printf("%d\n",c[2]);

        return 0;
}
```

Result

```
65
66
67
```

▶ TIP

Computer only knows about numbers whatever the form of data is. Execute it by comparing the difference of conversion format, %c and %d.

Let's play with variable!

01

Learning Contents

Try using variable

This source code is a code to print as it is after name and age are entered. Two variables are used here. Try copying it.

Source code

```
#include <stdio.h>
int main()
{
     char name[100] = {0,};
     int old = 0;

     printf("name:");
scanf("%s",name);
printf("age:");
scanf("%d",&old);

printf(“Hello.%s, you have become %d years old already!\n”,name,old);

     return 0;
}
```

Result

```
Name:
Age:
Hello. [input name], you have become [input age] years old
already!
```

Why did you make the array called name in this way? Something’s a little different.

It is a method to make all characters have a value of 0 after making array of 100 characters with the name of ‘name’. This method can only be used when making array. It is a way to put name entered with instruc- tion, scanf(“%s”,&name);, to array made as such.

name	← scanf("%s",name);

Aha! The instruction, scanf(), is entered the value.

That’s right. If entered character string with scanf(“%s”, name), charac- ter enters one by one from the room 0. Instead, you have to define the data type of data to be entered just like variable. It is a similar form with format specifier using in printf(). Character string is ‘%s’ and integer type is ‘%d’.

Ah. Then just like printf(), I can put the name of variable to be entered behind it, right?

▶ TIP

If you want to know more about & placed in front of array, read ‘Learn more!’ on page 57.

Yes. If I add one more, when you enter or print array, you should put & in front of the array name originally. However, when you use the first value of the array, like &name[0], you don’t have to put & and index ([]) and just use as ‘name’.

name	&name[0]

By the way, you should take a note that computer recognizes spacing as the end of the data since it is only entered up to the blank when entered data with scanf.

02

Learning Contents

What if data type is different?

This time, we’ll do a fun experiment with variable. What will happen if variable with different data type is assigned to already-made variable?

Source code

```
#include <stdio.h>
int main()
{
     int number = 81;
     char old;

     old = number;

     printf("%d->%c",number,old);

     return 0;
}
```

▶ TIP

Alphabet A corresponds to 65, B corresponds to 66, and Q corresponds to 81.

Result

```
81->Q
```

Huh? What? I get totally different result, not an error?

In character data type, ‘81’ is not recognized as the number but as the character corresponding to the numerical code. Since the character corresponds to 81 is Q, Q is shown.

Then what should I do to show the number itself as the character?

There’s also the way for that.

03

Learning Contents

Converting the data type

The following is a code which includes a method to convert variable declared as integer to character..

SOURCE CODE

```
#include <stdio.h>
int main()
{
        int number = 81;
        char old[3] ={0,};

        sprintf(old,"%d",number);
        printf("%d->%s",number,old);

        return 0;
}
```

Result

81->81

Sprint? It seems similar to printf.

If printf functions as showing the character string on screen, sprint functions as saving the character string to array, not screen.

```
char name[5] = "JACK";
printf("hello %s", name);
```

- printing hello JACK on the monitor

```
char buf[20] = {0,};
char name[5] = "JACK";
sprintf(buf,"hello %s", name);
```

- Char buf[20]-> hello JACK is saved

Isn't the direction for use also similar? But there is something a little different. In sprintf, the name of array to save should come as the first element. You have to make enough size of array to put all characters to save, of course.

```
char buf[20]  = {0,};
sprintf(buf,"hello %s", name);
```

Oh, I suppose. Then, similar to printf, character string to save and format specifiers, such as %d or %s, should come after that, right?

```
sprintf(buf,"hello %s", name);
```

Yes, from the third element, also, you have to write variables to show as for- mat specifiers, right?

```
char name[5] = "JACK";
sprintf(buf,"hello %s", name);
```

That's right. Then, what does "sprintf(old,"%d",number);" mean?

I suppose it means to save the value saved in number variable to old array as character string in form of decimal.

You are correct! Don't forget that number variable was originally number since it is integer type (int), but now it is converted to character string so it is saved at old!

01

Follow

Making a simple program!

Well then, let's make a program for computer shutdown timer by applying what we have studied so far. Let's try together!

Source code

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
      int time = 0;
      char cmd[100] = {0,};

           Printf("Please enter the time for computer shutdown timer(unit=sec):
");
      scanf("%d",&time);
      sprintf(cmd, "start shutdown -s -t %d",time);
      system(cmd);

      return 0;
}
```

Result

Please enter the time for computer shutdown timer(unit=sec):

Exercise 01

This is a modified program for computer shutdown timer. Execute the following source code, and write down the reason for doing ‘cmd[mode-1] at ‘sprintf (cmd,”start shutdown -%c”,cmd[mode-1]);’.

Source code

```
#include <stdio.h>
#include <stdlib.h>
 int main()
{
        char cmd[2] = {'p','r'};
        char str[100]={0,};
        int mode;

        printf(“Select the mode of computer shutdown timer. \ n”);
        printf(“1. Shut down now \ n”);
        printf(“2. Rebooting \ n”);
        printf(“[Enter the number]: \ n”);

        scanf("%d",&mode);
        sprintf(str,"start shutdown -%c",cmd[mode-1]);
        system(str);
        return 0;
}
```

Result

```
Select the mode of computer shutdown timer.
1. Shut down now
2. Rebooting

[Enter the number]:
```

Exercise 02

Let’s make a program for video retrieval in computer this time. Can you fill in the blank so the execution result comes out as following?

Source code

```
#include <stdio.h>
#include <stdlib.h>
int main()
{

        char cmd[100] = {0,};
        char q[100] = {0,};
        printf(“Enter the title of video you want to retrieve: \ n”);
        scanf("%s",q);

        sprintf(cmd,"start http://www.youtube.com/results\?search_query=%s",q);

        system(cmd);

        return 0;

}
```

Result

```
Enter the title of video you want to retrieve:
```

If you want to enter character string including blank character with scanf, you can use the format, such as ‘scanf(*%[^ \ n]”,q)’. Then, all characters are saved at variable until \ n(line feed-line change) character comes out.

Exercise 03

This time I’ll make a map search program. Fill in the blank so the execution result comes out as following.

Source code

```
#include <stdio.h>
#include <stdlib.h>
 int main()
{

        char cmd[100]= {0,};
        char q[100] = {0,};


        printf(Enter the region you want to search: \ n”);


        sprintf(cmd,"start https://maps.google.com/maps\?q=%s",q);

        system(cmd);

        return 0;

}
```

Result

```
Enter the region you want to search:
```

Exercise 04

Shall we make an image search program? Find the image you want to search by executing the source code.

Source code

```
#include <stdio.h>
#include <stdlib.h>
int main()
{

        char cmd[100]= {0,};
        char q[100] = {0,};


        printf(Enter the name of image you want to search: \ n");

        scanf("%s",q);

        sprintf(cmd, "start https://www.google.com/search?q=%s^&tbm=isch",q);

        system(cmd);

        return 0;

}
```

Result

Enter the name of image you want to search:

Learn more! How is the data entered with scanf()function?

Scanf should hand over memory address value instead of variable name. Do you remember printing memory address of variable in copying 01 (page 39) of DAY 4?

Source code

```
#include <stdio.h>
int main()
{
        int a = 65;
        printf("%p\n",&a);
        return 0;
}
```

- & is an operator which shows memory address, not variable value that comes after.
- %p is a format specifier mark which prints memory address in hexadecimal.

&& is an operator which changes the name of variable that comes after to the memory address of the corresponding variable. Shall we find out more about what address is?

We learned in DAY 1 that one of the roles of variable name is to show where in the memory the data is saved. The 'old' in the code on the right is actually the address of memory (0x00) which saved data, 4. When the variable name is used as it is, it means the value of data that variable saves. However, if & is put in front of a, it shows the address value of memory saved instead of value that saves..

```
int old = 4;
```

old	4
&old	0x04

int old;	scanf("%s",&old);

Scanf() requires the address value of variable to save input value so & should be put in front of the variable.

Then, what happened to the case below? What is the reason & is not put?

char **name[5]** = {0,};	scanf("%s",**name**);

It's because name has the same meaning as %name[0].

Shall we find out more in detail? We learned that number has to be written in [] to see the value saved in array. This is what's called index. However, what value does it mean if just variable name is used as be- low, not indicating the index value? The entire "JACK" character string?

```
char name[5] = "JACK";
```

name	?????

name[0]	'J'
name[1]	'A'
name[2]	'C'
name[3]	'K'
name[4]	'\0'

It is easy to be mistaken that it means the entire character string since it is used as the right when printing the character string using printf earlier. But name means the address value of memory that saves the first data the array indicates, not the entire character string that array is saving. In other words, it is the same as &name[0]. If you execute the code below, you can see that the same value is printed for both &name[0] and name.

Code	Print Value
printf("%s", name);	JACK

SOURCE CODE

```
#include <stdio.h>
int main()
{
        char name[5]= "JACK";

        printf("%p\n", &name[0]);
        printf("%p\n", name);

        return 0;
}
```

Then, why the character string, JACK, not the address value, comes out in printf statement? It's because the role of %s format signifier is made to print all saved values starting from memory space that the address value indicates up to the space that the null character, \ 0, is saved appears.

Address value	Data
0x01	J
0x02	A
0x03	C
0x04	K
0x05	\0

The role of %s
It prints all values saved starting from where the name indicates until where the null character, \ 0, appears.

* What will happen if the null character, ' \ 0' is removed?
Try to solve the exercise 5 of DAY 7 (p.66).

In other words, if just variable name is used in array, not giving index ([]), this means address value of the first data of the array. In scanf(), only address value should be given, but since name is already an address value, & does not have to be added.

Summary

1. Scanf requires the address value of variable to save input value.
2. If index is not given to variable name in array, the first value of the array is the address value of the saved space.

char name[5] = "JACK";

name	&name[0]

※What would happen if &name is added to name? if you add & operator to address value, it still shows the address value.

What If There's Array in Array?

01

Learning Contents

Two-dimensional array

I'll put a sentence composed of three words in one variable using two-dimensional array. Watch carefully and follow me.

SOURCE CODE

```
#include <stdio.h>
int main()
{
        char str[3][100] = {"SAMSUNG","INNOVATION","CAMPUS"};

        printf("%s\n",str[0]);
        printf("%s\n",str[1]);
        printf("%s\n",str[2]);

        return 0;
}
```

➢ t can be written as below.

```
char      str[3][100]      =
          {{'S','A','M','S','U',
'N ', 'NG',' \ 0 ' } , { 'I ', 'N ', ' N ', ' O ', ' V ',
' A ', 'T', ' I ','0','N ',
'\0'},{'C','A','M','P','U','S','\0'}};
```

Result

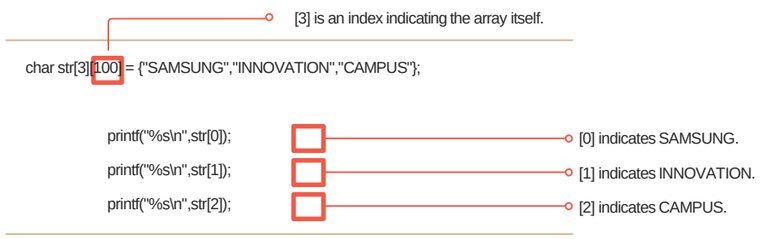
SAMSUNG
INNOVATION
CAMPUS

There are two indexes in array, aren't there?

Yes. That's why it's called two-dimensional array. You can use it by saving a number of words in one variable name. The array called str in the code can save a sentence composed of 100 letters or up to 3 words.

Is it like an array saving array?

Oh, you understand it pretty well.



※[0][1] indicates 'S' of SAMSUNG, [1][1] indicates 'I' of INNOVATION, and [2][1] indicates 'C' of CAMPUS.

02

Learning Contents

Even long sentence can be written freely!

This time, I'll show you a code writing longer sentence. Try it yourself.

SOURCE CODE

```
#include <stdio.h>
int main()
{
        char str[3][100] = {0,};

        printf("When is the most important time?");
        scanf("%s",str[0]);
        printf("Who is the most necessary person?");.
        scanf("%s",str[1]);
        printf("What is the most important work?");.
        scanf("%s",str[2]);

        printf(" \ n \ n  The most important time is %s \ n",str[0]);
        printf(" the most necessary person is %s, and \ n",str[1]);
        printf("the most important work is %s. \ n",str[2]);

        return 0;
}
```

➢ Reset method of two-dimensional array is the same as that of one-dimensional array.

Result

When is the most important time? Now
Who is the most necessary person? You
What is the most important work? What you are doing right now

The most important time is now,
the most necessary person is you, and
the most important work is what you are doing rightnow

Where did you find these cringeworthy phrases?

Ah, you don't like these

Why? I like these! A man who sometimes gets in the mood is attractive.

(groaning)…it doesn't suit me…

03

Learning Contents

Spacing can also be done in any degree!

This time, I'll fix the previous code a little bit so it can be entered with character string including a blank space. I used new instruction. Take a look.

SOURCE CODE

```
#include <stdio.h>
int main()
{
        char str[3][200] = {{0,}};

        printf("When is the most important time? ");.
        scanf("%[^\n]",str[0]);
        getchar();
        printf("Who is the most necessary person? ");.
        scanf("%[^\n]",str[1]);
        getchar();
        printf("What is the most important work? ");.
        scanf("%[^\n]",str[2]);

      printf(" \ n \ n The most important time is %s \ n",str[0]);
      printf(" the most necessary person is %s, and \ n",str[1]);
      printf("the most important work is %s. \ n",str[2]);

        return 0;
}
```

Result

```
When is the most important time? N ow
Who is the most necessary person? Y ou
What is the most important work? What you are doing right now

What is the most important work? What you are doing right The most important time is n ow,
the most necessary person is y ou, and
the most important work is what you are doing right now.
```

What is getchar()? I haven't seen this before. There's nothing written in the ().

It's similar to scanf(), but getchar() has a function of being entered only one character. The reason there's nothing in the () is because it is a function which operates even without parameter.

getchar()	A function without parameter
scanf("%s",name);	A function with parameter

Parameter?

The elements that go into the (), divided by comma(,) when using printf() or scanf() are called parameter.

Is that so? I have been using without knowing the name. Then, where is the character entered with getchar() saved? The scanf uses parameter to set the place to save.

The getchar() plays a role of returning the input character right away without saving it. Thus, it's used as following when saving the input character.

```
char c;
c = getchar();
```

Using like this is expressed as calling function and saving return value of function to variable.

Ah, Now I get it.

04

Learning Contents

What will come out?

Shall we try a fun experiment since we learned it? What result will come out if we save return value of printf or scanf, just like getchar? Let's try it together!

▶ TIP

The printf() returns the number of printing character, and the scanf() returns the number of reading format signifier.

SOURCE CODE

```
#include <stdio.h>
int main()
{
        int r = 0;
        int c = printf("InputNumber:");
        int d = scanf("%d",&r);
        printf("return value of printf: %d \ return value of nscanf: %d \ n", c, d);

        return 0;
}
```

Result

Exercise 01

The following code does not cause an error but it does not operate normally.
Take a look carefully to find the incorrect part and fix it.

SOURCE CODE

```
#include <stdio.h>
int main()
{
    char str[3][10] = {"juior","INNOVATION","CAMPUS","world"};
    printf("%s\n", str[0]);
    printf("%s\n", str[1]);
    printf("%s\n", str[2]);
    printf("%s\n", str[3]);
    return 0;
}
```

Result(when normal running)

```
SAMSUNG
INNOVATION
CAMPUS
world
```

Exercise 02

It's a similar problem. Can you find the incorrect part and fix it?

SOURCE CODE

```
#include <stdio.h>
int main()
{
    char str[10][5] = {"hello","SAMSUNG","INNOVATION","CAMPUS","Good!"};
    printf("%s\n", str[0]);
    printf("%s\n", str[1]);
    printf("%s\n", str[2]);
    printf("%s\n", str[3]);
    printf("%s\n", str[4]);
    return 0;
}
```

Result(when normal running)

```
hello
SAMSUNG
INNOVATION
CAMPUS
Good!
```

Exercise 03

It's a similar problem. Can you find the incorrect part and fix it?

SOURCE CODE

```
#include <stdio.h>
int main()
{
    char str[3][10] = {"hello, SAMSUNG","INNOVATION CAMPUS","Good!"};
    printf("%s\n", str[0]);
    printf("%s\n", str[1]);
    printf("%s\n", str[2]); return 0;
}
```

Result(when normal running)

```
hello, SAMSUNG
INNOVATION CAMPUS
Good!
```

Exercise 04

Fill in the blank of the following code to get the result below.

▶ TIP

Don't forget ' \ 0' comes at the end of character string.

SOURCE CODE

```
#include <stdio.h>
int main()
{
    char str[ ]  ={"HELLO WOLRD","SAMSUNG INNOVATION CAMPUS"};
    str[ ] = [ ]
    str[ ] = [ ]
    printf("%s",str[0]);
    printf("%s",str[1]);

    return 0;
}
```

Result(when normal running)

```
HELLO SAMSUNG
```

Exercise 05

Now is the last problem. When you solve this problem there's nothing more I can teach you. Fill in the blank to get the result below.

SOURCE CODE

```
#include <stdio.h>

int main()

{

      char str [    ]={"HELLO","SAMSUNG","SOFTWARE"}
      ;
      str[    ] = [    ];
      str[    ] = [    ];

      printf("%s",str[0]);

      printf("%s",str[1]);

      printf("%s",str[2]);


    return 0;

}
```

Result

HELLO SAMSUNG SOFTWARE

Learn more! What is the difference between scanf() function and getchar() function?

getchar()	h(……)
scanf()	hello world

The getchar() is a function that does a similar work as scanf(). If scanf() can be entered character string, getchar() can be entered one character. Then, why is getchar() used between scanf() in the code on the left?

Example

```
#include <stdio.h>

int main()
{
      char str[3][200] = {{0,}};

      printf("When is the most important time? ");.
      scanf("%[^\n]",str[0]);
      getchar();
      printf("Who is the most necessary person?");.
      scanf("%[^\n]",str[1]);
```

The contents we are entering with keyboard is saved temporarily in the memory space called input buffer.

Scanf format specifier	input	value result
scanf("%s",name);	hello world	Saving hello to name
scanf("%d",name);	hello world	Saving nothing to name array

Actually, scanf() reads the value saved in this input buffer and if there's a value that fits the format specifier(%), it brings the corresponding value from the buffer, and if there's a value that does not fit the format specifier, it ends the function.

Keyboard input	input buffer	scanf("%[^\n]",str[0])
jack + Enter key input (\n)	\n k c a j	Wait until enter key is input
	\n k c a	→ j
	\n k c	→ a
	\n k	→ c
	\n	→ k
	\n	scanf ends!

%[^ \ n] of scanf("%[^ \ n]",str[0]); means to bring all character input until line change (\ n) appears. Thus, when line change(\ n) is met in the input buffer, the function ends and \ n remains in the input buffer.

Keyboard input	input buffe	scanf("%[^\n]",str[0])
hi! + Enter key input (\n)	\n	Wait until enter key is input
	\n ! i h \n	
	\n ! i h \n	scanf ends!

However, if scanf("%[^ \ n]". str[1]) is used, the function ends right away since the first character of the input buffer reading again is \ n. Therefore, there's a need to remove \ n of input buffer.

Using getchar() between scanf() can be used effectively to solve problem above because it bring the \ n remaining in input buffer.

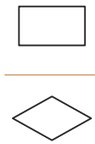
Learn more! What is flowchart?

01 The methods to express algorithm in the problem-solving process include natural language, flowchart, mnemonic code, and programming language, etc. Flowchart is expressing the method and procedure of processing a certain problem using computer with signs. It is expressed using agreed signs according to logical flow in the step before coding the program. It is convenient to understand the overall flow of program, find and modify the logical error.

※Algorithm is a logical procedure to solve problem, and refers to what constructed the works to be processed for problem solving clearly by each step.

02 **The rules of writing flowchart**

❶ It should be written to express the direction of flow line from top to bottom, and from left to right and not to cross each other.
❷ It is expressed using agreed signs and figures
❷ Frequently used signs of flowchart are as following.

Signs	Names	Use
	Terminal	It expresses the beginning and the end of flow chart
	Process	It expresses all kinds of operation (calculation), data movement, and process
	decision	It shows the logical branch by decision of comparison and condition
	Input/ output	It shows the general input/output of data
	document	It shows printing of document form with paper media.
	flowline	It indicates the flow of the process

Practice flow chart

01 Shall we express the process from waking up in the morning to going to school in flow chart?

02 Shall we express the process of turning on the power of cell phone, finding the camera function, taking a picture of discussing in the team activity, and sending it to team members in flow chart?

Samsung Innovation Campus

Chapter 5.

C Programming

STEP 2.

DAY 8.

Making File at My Disposal

01

Learning Contents

Let's Make a File!

Umm, then let's start from this. You should be able to save the output into file if you want to make something using C language. I'll tell you how to save file though making simple webpage. I always check the current condition of my farm with web page. Would you make it as it is written below?

Source code

▶ TIP

When fopen() function is not successful at creating file, NULL is returned..

```
#include <stdio.h>
 int main()
{
        FILE *f = fopen("sic.html","w");
        fclose(f);

        return 0;

}
```

➢ FILE*f declared the variable f as a pointer to indicate the file to open with fopen() function.

Result

➔ The jsa.html file is created in the folder with source code file.

What is FILE?

You know that data type is written in front of variable name when declaring variable to save numbers or characters, right? FILE is also one of the data type, such as int or float.

Since the name of data type is FILE… By any change, is it a data type that saves to computer file we use?

Um, it's similar. However, it does not save the file itself, but it's more appropriate to see it as a data type which plays a role of showing where the door to file is.

I don't understand.

Do you see the part, fopen? This is the function that opens the door to file. This door has to be opened to read or write the contents of file. The fclose below that is a function which works as closing door to file.

This door is called file descriptor, and FILE is a data type that means this file descriptor.

Then, what is * in front of the variable name?

That is called pointer. It is a little difficult concept but I'll try to explain easily. When you declare variable, it becomes pointer variable that can save address if you put * in front of or behind the variable name.

Umm… Then FILE*f is a door for reading and writing file; in other words, it is a variable that saves "ad- dress" which shows where the file descriptor is, isn't it?

That's it.

But, what is the reason for closing the door that is opened?

Good question. It varies by the environment a little bit, but when you open the door to file, other program can't open file until the door is closed. Thus, when the file is all used, the door must be closed. Besides, When making new file or adding new contents to existing file, the file is actually created or the contents is save at the moment of closing the door, not at the moment of writing; therefore, you should not forget to close the file.

Learn more!

Let's learn more about fopen("jsa.html","w"). The first parameter in () is written with storage route of file (when the route is omitted, the folder with source code becomes the storage route). The second parameter in () is written about with what mode the file is opened. The w means opening with writing mode, and when opened with w mode, you can only write but not read. The mode to open file is as following.

"r"	Opening file for read-only (the contents cannot be added to the file._
"w"	Making new file for write-only (when the file with the same name already exists, the con- tents of the existing file is deleted.)
"a"	It makes new file for write-only, but when there is a file with the same name, it does not delete and add the contents to the existing file.
"r+"	It can be read or written.
"w+"	It can be written or read.
"a+"	It can be read or written, and when there is the existing file, the contents are added to the existing file.

02
Learning Contents

Let's add contents!

If you made file, now you need to make contents to be added to file, right? The part adding contents to file is not much different from printing on screen. You'll know if you follow me.

Source code

```
#include <stdio.h>
int main(){

        FILE *f = fopen("sic2.html","w");

        fprintf(f, "Hello Samsung Innovation Campus!");

                fclose(f);

}               return 0;
```

Result

→ Let's click and execute the jsa2.html file which is created after compiling.
Web browser is executed and the phrase, "Hello Samsung Innovation Campus", is shown on screen.

If file is opened by using file descriptor, you can print contents in file using fprintf. If printf plays a role of printing on screen, fprintf plays a role of printing in file.

In a sense, it is similar to how we edit document; for example, opening file to modify and then closing again.

It is.

By the way, what file format is HTML?

It is a file format typically used in web page. You can say it's similar to C language, but is dissimilar in that it is a markup language which is interpreted and executed by web browser right away without compiling process. It is more convenient than C language, but C language is much faster. HTML requires time for web browser to interpret before it is executed, but C lan- guage is executed right away without interpreting process since it's already compiled with machine language.

Exercise 01

Then, shall we practice? Make html file with your friends' names by looking at the source code and filling in the blank, and then make your friends' names appear by putting these in file.

Source code

```
#include <stdio.h>
 int main()
{
        FILE *f = fopen(" sic.html          ","w");

        fprintf(f," sic  ");

        fclose(f);

        return 0;
}
```

Result

➔ html file is created with your friends' names.
If the created html file is executed, your friends' name shows up on web browser.

Exercise 02

This one is a code which makes html file when it is executed and then checks the contents by opening the file directly in web browser. Try to follow me.

Source code

```
#include <stdio.h>
#include <stdlib.h>
 int main()
{
        FILE *f = fopen("sic3.html","w");

        fprintf(f, "Hello Samsung Innovation Campus!");

        fclose(f);

        system("sic.html");

        return 0;
}
```

Result

➔ If it is executed after compiling, file is created, web browser opens automatically, and "Hello Samsung Innovation Campus!" appears.

Exercise 03

Shall we make a file by getting the contents of webpage entered directly with keyboard? You can put anything for input contents.

Source code

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
        FILE *f = fopen("sic5.html","w");
        char html[100] = {0,};
        scanf("%[^\n]",html);

        fprintf(f, "%s", html);

        fclose(f);

        system("jsa5.html");

        return 0;
}
```

Result

➔ Contents entered with keyboard shows up on web browser.

Exercise 04

Well then, fill in the blank of source code so both previously input contents and newly input contents can show up on webpage.

▸ TIP

When using file descriptor, if you use "a" instead of "w", you can add the contents to existing file without deleting the file even though there is file with the same name.

Source code

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
        FILE *f = fopen("sic5.html"," a ");
        char html[100] = {0,};

        scanf("%[^\n]",html);

        fprintf(f, "%s", html);

        fclose(f);

        system("sic5.html");

        return 0;
}
```

Result

➔ Newly input contents are added to the contents previously input to sic5.html and show up on web browser.

Exercise 05

This time, let's write a source code to pick out and save the contents you want to show on web browser among a number of contents.

Source code

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
	FILE *f = fopen("sic6.html","w");
	int mode = 0;

	char menu[4][100] = {"JIM", "SIC", "Mr. Samsung", "Great SIC"};

	printf("Plese select the contents you want to show on web browser\n");
	printf("[1]Popular singer of the month\n");
	printf("[2]Popular drama of the month\n");
	printf("[3]Sports player of the month\n");
	printf("[4]Popular movie of the month\n");
	printf("Enter the number(1-4):");

	scanf("%d",&mode);

	fprintf(f,"<html>"); fprintf(f,"<marquee
	direction=UP>"); fprintf(f,"%s", menu[mode-1]);
	fprintf(f,"</marquee>"); fprintf(f,"</html>");
	fclose(f);

	system("sic6.html");

	return 0;
}
```

Result

→ the contents selected with numbers
show up on web browser.

DAY 9.

Let's Bring Out Data from File!

01

Learning Contents

Let’s display the file contents!

Since you need contents to input, inquire into your friends’ names and hobbies and record these in a notepad as the figure below. Recorded file must be saved in the same folder as source code. If it’s done, write and execute the following source code.

Source code

```
#include <stdio.h>
int main()
{
    FILE *f = fopen("Survey.txt","r"); char s[10] = {0};

    fscanf(f,"%s",s);
    printf("%s\n",s);


    return 0;
}
```

- fscanf() is a function which receives input in the unit of variable from the opened file.Variables are separated with blank, tap, and line feed.

Result

```
Name
```

‘r’ is used in file descriptor this time?

That means the file is opened as read-only. If you open file as read-only, you can read the contents but can’t add the contents.

It’ll be easier when you just read the file.

The fscanf is similar to scanf, but it functions as reading data from file, not from input device. When reading for the first time, it disregards all blanks until actual character, not blank, appears.

It seems that format specifier, such as %s, is used the same as scanf, right?

Yes. If the contents read are different from format specifier, i t can’t read the data.

02

Learning Contents

A number of contents at a time

Once you made the file, now you need to make contents to put in file. The part adding contents to file is not much different from printing on the screen. You’ll see if you copy after it.

Source code

```
#include <stdio.h>
int main()
{
    FILE *f = fopen("Survey.txt","r");
    char name[10] = {0,};
    char age[10] = {0,};
    char favo[10] = {0,};

    fscanf(f,"%s ",name);
    fscanf(f,"%s ",age);
    fscanf(f,"%s ",favo);

    printf("%s %s %s\n",name,age,favo);
    fclose(f);

    return 0;
}
```

Result

```
name age hobby
```

If you read about file opened with fscanf several times, it starts reading from the part it last read so it can read a number of contents.

03

Learning Contents

More conveniently

But if contents are read in the same way as earlier, it would be inconvenient to write a program reading a number of contents. This time, let's read a number of contents at a time.

Source code

```
#include <stdio.h>
int main()
{

    FILE *f = fopen("survey.txt","r");

    char name[10] = {0,};
    char age[10] = {0,};
    char favo[10] = {0,};
    fscanf(f,"%s %s %s",name,age,favo);
    printf("%s %s %s\n",name,age,favo);

    fclose(f);

    return 0;
}
```

Result

```
name age hobby
```

If a number of format specifiers are given at a time, fscanf can assign to variable in order of being read.

It's like assigning to variable by reading contents until blank appears and then assigning to the next variable if other content appears after the blank, right?

That's right. it reads up to the number of format specifier, but you have to be careful because reading stops if contents read do not fit the format specifier.

Okay, I get it!

04

Learning Contents

Actual Use!

Since you seem to understand somewhat now, I think you should practice how to use in reality. If you write and execute the source code as shown below, you'll understand the contents as well as the title can be expressed this time.

Source code

```
#include <stdio.h>
int main()

    char name[10] = {0,};
    char age[10] = {0,};
    char favo[10] = {0,};

    printf("%s %s %s\n",name,age,favo);
    fscanf(f,"%s %s %s",name,age,favo);
    printf("%s %s %s\n",name,age,favo);

    fclose(f);

    return 0;
}
```

Result

```
Name Age Hobby
Cho, Sijung 18 volunteering
```

Aha! If I repeat like this, I can express all data!

Yes. It's not that difficult, right?

This is fun.

Exercise 01

Do you understand now? Then, let's write the source code by filling the blank to show all the list in survey.txt written earlier.

Source code

```
#include <stdio.h>
int main()
{
    FILE *f = fopen("survey.txt","r");

    char name[10] = {0,};
    char age[10] = {0,};
    char favo[10] = {0,};

    fclose(f);

    return 0;
}
```

Result

Name	Age	Hobby
Cho, Sijung	18	Volunteering
Oh, Yoonkyung	18	Drawing
Yoon, Jihyun	17	Boxing
Hur, Hyung	18	Reading a book
Kim, Bonyun	17	Listening to music
Ko, Byungwook	19	Computer game
Song, Jaeran	16	Composition

Exercise 02

Shall we handle number, not character string? Try to fill in the blank of the code to save and print the data corresponding to age of the data, saved to survey.txt, to old which is int type variable.

Source code

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    FILE *f = fopen("survey.txt","r");

    char name[10] = {0,};
    char  age[10] = {0,};
    char favo[10] = {0,};

    int old = 0;

    fscanf(f,"%s %s %s",name,age,favo);

    printf("%s %s %s\n",name,age,favo);

    fclose(f);

    return 0;
}
```

Result

Name	Age	Hobby
Cho, Sijung	18	Volunteering
Oh, Yoonkyung	18	Drawing
Yoon, Jihyun	17	Boxing
Hur, Hyung	18	Reading a book
Kim, Bonyun	17	Listening to music
Ko, Byungwook	19	Computer game
Song, Jaeran	16	Composition

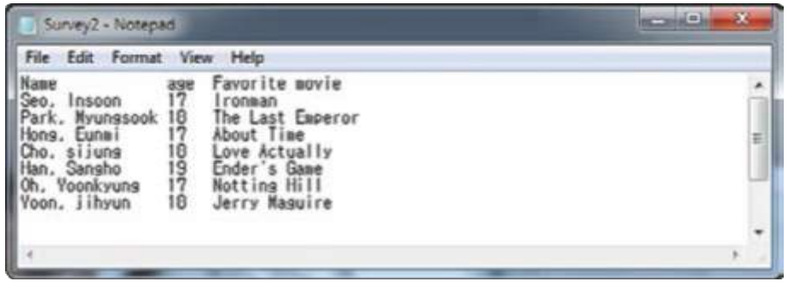
You learned from Charlie that the format specifier that can receive input of integer is %d, right?

Yes. I remember. When saving to integer variable, I have to put & in front of variable name, like scanf. Is that correct?

That's correct!

Exercise 03

Let's make a program to search videos by the movie title after making questionnaire about favorite movies of your friends this time. You have to make it be able to search including the spacing.



▶TIP

fprintf: : print file
sprintf: print character string

▶TIP

System() function receives the name of process to execute and execute the process.

Source code

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    FILE *f = fopen("Survey2.txt","r");

    char   name[20] = {0,};
    char    age[20] = {0,};
    char    mov[20] = {0,};
    char  cmd[100] = {0,};

    fscanf(f,"%s %s %[^\n]",name,age,mov);

    sprintf(cmd, "start https://www.google.com/search?q=%s^&tbm=vid",mov);
    system(cmd);

    fclose(f);

    return 0;
}
```

Result

· video search results about movie titles show up on web browser in order.

DAY 10.

Add, Subtract, Multiply, and Divide

01

Learning contents

Addition and Subtraction

So far you have learnt how to read and write data, and now it's time to learn how to process the data in earnest. However, it's not too difficult, and since we'll start from addition and subtraction, you don't have to be scared. Take a look at the following source code and write as it is written.

Source code

```
#include <stdio.h>
int main()
{
        int r = 10 + 20;

        printf("10 + 20 = %d \n", r);

        printf("10 + 20 = %d \n", 10+20);

        return 0;
}
```

Result

```
10 + 20 = 30
10 + 20 = 30
```

It really is addition and subtraction!

Yes. The formula can show the result right away or it can be saved at variable first and then show.

02

Learning contents

Multiplication and Division

Shall we do multiplication and division as well as addition and subtraction using variable this time? It'll be better if you try yourself than just listen to the explanation.

Source code

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

```
int main()
{

        int a = 2, b = 1;

        printf("%d plus %d = %d \ n",a,b,a+b);
        printf(%d minus %d = %d \ n",a,b,a-b);
        printf(%d times %d = %d \ n",a,b,a*b);
        printf((%d divided by %d = %d \ n",a,b,a/b);
        printf((the reaminder of %d divided by %d = %d \ n",a,b,a%b);

        return 0;

}
```

➜a/b finds the quotient diving a by b.
➜a%b finds the remainder of a divided by b

Result

```
2 plus 1 = 3
2 minus 1 = 1
2 times 1 = 2
2 divided by 1 = 2
The remainder of 2 divided by 1 = 0
```


The four fundamental arithmetic operations are exactly the same as math.


By the way, what is %? It doesn't seem like a percent…


It's called modular operator which is used to find the remainder of dividing a number. It is an operator used surprisingly often in programming.


It feels awkward calling mathematical sign as operator.


Generally, +(addition), -(subtraction), /(division), *(multiplication), and %(remainder) used in C language are called arithmetic operator.


I should keep that in mind.


Yes. Actually, there are many types of arithmetic operators other than these so you should study on your own later.

03
Learning Contents

New data type

Shall we try calculation including decimal point this time? It's similar to what we have done so far except for one big difference. Write the following source code as it is and execute it.

Source code

```
#include <stdio.h>
int main()
{
        int i1 = 3, i2 = 2;
        float f1 = 3, f2 = 2;

        printf("3 divided by 2 in integer type = %d \ n",i1/i2);

        printf("3 divided by 2 in float type = %f \ n",f1/f2);

        return 0;
}
```

Result

```
3 divided by 2 in integer type = 1
3 divided by 2 in float type = 1.500000
```


Uh? I see float, not int.


That's right. It is the float data type which can assign floating point number. The variable made with int can only handle the integers, but float can handle data including decimal points.


Then, the data which can have floating point, such as division, should make variable as float, shouldn't it?


Yes, it should. That's why it's one of the most frequently used data types.


Then, %f used in printf is the format specifier which shows the float variable, right?


That's right! It is set to show down to 6 places of decimals by default.

04
Learning Contents

What if calculating between different data types?

By the way, don't you wonder what result would come out when calculating the data saved as int with the data saved as float? You should try it yourself.

Source code

```
#include <stdio.h>
int main()
{
        int i1 = 3, i2 = 2;
        float f = 2, r = 0;

        r = i1 / i2;
        printf("When saving the result of operating int and int to float variable = %f \ n",r);
        r = i1 / f ;
        printf("When saving the result of operating int and float to float variable = %f \ n",r);

        return 0;
}
```

Result

```
When saving the result of operating int and int to float variable = 1.000000
When saving the result of operating int and float to float variable = 1.500000
```

Uh? Both are diving 3 by 2, but the result are different?

Isn't it interesting? The first result is printing by assigning the result of dividing int type by int type to float type variable. In the operation between int types, the result value also always comes out in int type. Even though the resulting value is saved to float which can express decimal point, it is not calculated down to decimal point.

Then, why does the result of operation between two different data types (int type and float type) come out normally?

It's because when operating between different data types, data types are converted in accordance with data type with larger expression range. Since float type has larger expression range than int type, when you operate int type and float type together, int type is automatically converted to float type to calculate.

Ah, that's why the result is integer, 1, when operated between int types, and the result is real number, 1.5, when operated int type and float type.

05
Learning Contents

Calculating Precisely!

Then, what should be done to express the result precisely as real number even though division is done between data saved as int type? Let's find out by trying yourself.

Source code

```
#include <stdio.h>
int main()
{
        int i1 = 3, i2 = 2;
        float r = 0;
        r = i1 / i2;
          printf("When saving the result of operating int and int to float variable = %f \ n",r);

        r = (float)i1 / i2 ;
        printf("The operation result after type conversion of one int = %f \ n",r);


        return 0;
}
```

· explicit type conversion
It is instructing type conversion forcibly by using typecast operator. (ex: (float);1)

Result

```
When saving the result of operating int and int to float variable = 1.000000
The operation result after type conversion of one int = 1.500000
```

Wow, Division is done between int types and the resulting value comes out in float type.

Do you see adding (float) in front of i1? It is called typecast operator, which is an operator that converts data type of variable behind to the data type in (). Since i1 is converted to float type to calculate, the resulting value came out by mistake.

Then, why is only data type of i1 converted? Shouldn't both be converted?

We just learnt about that if two different data types are operated, other variables also convert data type in accordance with data type which expresses the largest range.

Ah, that's right! So if you just convert one, other variables just change automatically! It's really convenient.

Exercise 01

Okay, then should I give a question? What does the format specifier, %,2f, in the following source code mean? You'll see if you write it down yourself.

Source code

```
#include <stdio.h>
int main()
{
        int i1 = 3, i2 = 2;
        float r = 0;

        r = (float)i1 / i2 ;
        printf("%d Division  %d = %.2f\n",i1,i2,r);
        r = i1 / i2 ;
        printf(
                                    );

        return 0;
}
```

Result

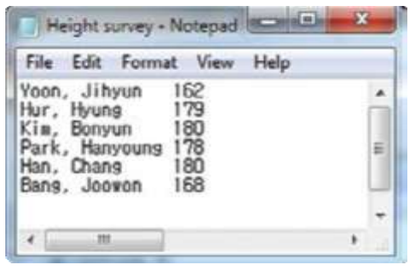
```
3 division 2 = 1.50
3 division 2 = 1.00
```

Exercise 02

Shall we make a proper program? Write down the heights of your friends on the notepad as the figure below, and save it as height survey.txt file. Let's make a program to find the average height with this file

Source code

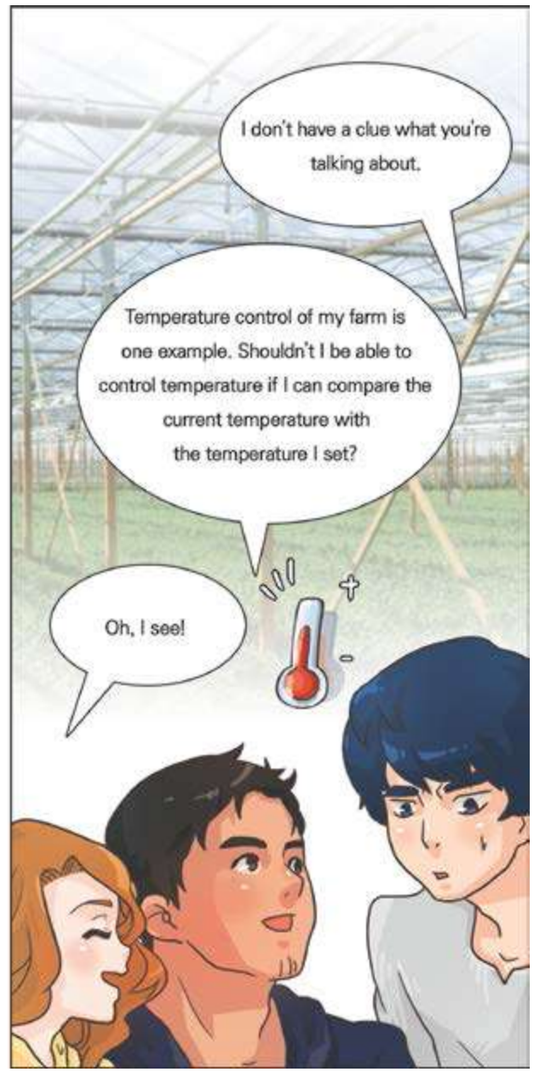
```
#include <stdio.h>
#include <stdlib.h>
int main()
{
        FILE *f = fopen("Height survey.txt","r");
        char name[6][100] = {0};
        int h[6] = {0,}; int
        count = 0;
        float result =  0.0;

        fscanf(f,"%s %d",name[count], &h[count]);
        count = count + 1;
                                    
        result = ( h[0] + h[1] + h[2] + h[3] + h[4] + h[5] ) /
        printf("The average height of %d people is %.2f.\n",count,result);
        fclose(f);
        return 0;
}
```



Result

The average height of 6 people is 173.17.

True or False?



01

Learning Contents

What is comparison operator?

How are the data compared in C language? It's not that different from calculating. It's using mathematical signs.

Source code

```
#include<stdio.h>
int main()
{
        printf("%d\n", 3>0);
        printf("%d\n", 5<0);
        printf("%d\n", 3>=5);
        printf("%d\n", 3<=5);
        printf("%d\n", 3==5);
        printf("%d\n", 3!=5);
        return 0;
}
```

➜3>0: Since the condition that 3 is bigger than 0 is valid, it is processed as true(1)
➜5<0: Since the condition is not valid, it is processed as false (0).
➜3>=5: Since the condition is not valid, it is processed as false (0).
➜3<=5: Since the condition is valid, it is processed as true (1).
➜3==5: Since the condition is not valid, it is processed as false (0).
➜33!=5: Since 3 and 5 are not equal, it is processed as true (1).

Result

```
1
0
0
1
0
1
```

I'm unfamiliar with true and false.

In programming, general, true and false is classified into 1 and 0, respectively. An operator which compares two data and processes as true and false is called comparison operator or relational operator.

But, it seems a little different from the signs used in mathematics.

That's right. Since there are some differences, you should know it from memory.

Bigger?	>
Smaller?	<
Equal?	= =
Not equal?	! =
Bigger or equal?	> =
Smaller or equal?	< =

'==' used for 'equal?' and '=' used for assigning value to variable should be distinguished carefully.

Yes. It's often confusing.

By the way, what happens if comparison operators are used at the same time?

In that case, calculate bigger or smaller, such as '<', '>', '<=', and '>=' first and then calculate to decide if they are equal, such as '!=' and '=='. Besides, don't forget that '<=' or '>=' is not spaced.

02

Learning Contents

Using comparing operator

Usually, comparing operator is well used in conditional (if) statement or repetition (while, etc.) statement, but I'll tell you about it later. For now, let's practice how to use by print statement first.

Source code

```
#include <stdio.h>
int main( )
{
        int a=9, b=4;
        printf("%d\n", a>b);
        printf("%d\n", a<b);
        printf("%d\n", a>=b+3);
        printf("%d\n", a<=b);
        printf("%d\n", a==b);
        printf("%d\n", a!=b+5);
        return 0;
}
```

➜>b: since the condition, 9>4, is valid, it is processed as 1.
➜a<b: since the condition, 9<4, is not valid, it is processed as 0.
➜a>=b+3: since the condition, 9>=4+3, is valid, it is processed as 1.
➜a<=b: since the condition, 9<=4 is not valid, it is processed as 0
➜a==b: since the condition, 9==4 is not valid, it is processed as 0.
➜a!=b+5: since the condition, 9!=4+5, is not valid, it is processed as 0.

Result

```
1
0
1
0
0
0
```

Since the result of comparison operation is always true or false, it is used to decide the condition in conditional statement or repetition statement.

Exercise 01

Then should I give a question? Fill in the blank of the following source code, and print the result by comparing if a is smaller than, equal to, or smaller than b.

Source code

```
#include <stdio.h>
int main()
{
        int a=7, b=3;
        printf("%d\n", b<10);
        printf("%d\n",    );
        printf("%d\n",    );
        return 0;
}
```

▸ TIP

Remember that when there are comparing operator and arithmetic operator together, arithmetic operator is processed first.

Result

```
1
0
1
```

Exercise 02

Well, this one is also a question of filliing in the blank. In the blank, fill the code using operator comparing if a is equal to b-2

Source code

```
#include <stdio.h>
int main()
{
        int a=7, b=3;
        printf("%d\n", a==b);
        printf("%d\n'
        printf("%d\n", a=b);
        return 0;
}
```

Result

```
0
0
3
```

Umm… why does 3 come out at the last line? It should be 0 or 1.

Hehe, I know why. Look at the code. a=b is not a comparison operator!

Exercise 03

Fill the code that can compare if a+2 is not equal or equal to b in the blank this time. The problem seems too easy.

Source code

```
#include <stdio.h>
int main()
{
        int   a=6,   b=10;
        printf("%d\n",    );
        printf("%d\n",    );
        return 0;
}
```

Result

```
0
1
```

Exercise 04

Shall we solve a little difficult problem? The following is the source code comparing three numbers in a number of ways. Try to put code using comparing operator in the blank. You can put the code comparing if the result of comparing if a is bigger than b is equal to b for the first blank, the code comparing if the result of comparing b is smaller than c is equal to a for the second blank, and the code comparing if the result of comparing b is bigger than c is not equal to a. Can you do it?

▸ TIP

When a number of comparison operators are used at the same time, you've learned the priorities earlier, right? When the comparison operators have the same priority, you can think that it is processed from the left.

Source code

```
#include <stdio.h>
int main()
{
        int a=3, b=2, c=1;
        printf("%d\n", a>b>c);
        printf("%d\n",    );
        printf("%d\n",    );
        printf("%d\n", a<b>=c);
        printf("%d\n",    );
        return 0;
}
```

Result

```
0
0
0
0
1
```

Exercise 05

This problem is filling in the blank to get a source code which, after two integers are entered, prints 1 if the integer entered first is bigger than the integer entered after it, and prints -1 if the former is smaller than the latter.

▶ TIP

How about using subtraction since true and false is 0 and 1, respectively?

Source code

```
#include <stdio.h>

int main( )

{
        int a, b;
        printf("Please enter two numbers ");
        scanf("%d %d", &a, &b);
        printf("%d, %dResult of = ", a, b);
        printf("%d\n",          );
        return 0;

}
```

Result

```
Enter two numbers. 5 7
The result of 5, 7 = -1
Enter two numbers. 7 5
The result of 7, 5 = 1
```

DAY 12.

Selecting by condition

01

Learning Contents

The basics of if statement

You know that if you use comparison operator we learned earlier, you can find out a certain condition through comparison, right? Now, I'll tell you how to execute the code selectively by the condition. Copy the following source code.

Source Code

```
#include <stdio.h>
int main( )
{
        int a=91, b=7;
        printf("a=%d, b=%d\n", a, b);
        if(a%b==0)
        {
                printf("%d: It is a multiple of  %d.\n", a, b);
        }
}
```

➜ if(a%b==0)
If the value dividing a by b is 0 (that is, if a is a multiple of b), the code block in '{}' below the if statement is executed. If it is not 0, the contents of code block are not executed.

Result

a=91, b=7
91: it is a multiple of 7.

The part below if of this source code can omit the block signs like this.

```
if(a%b==0)
 printf("%d: It is a multiple of  %d.\n", a, b);
 return 0;
```

If here is the similar to as if we use.

It is! It's easy if you think as executing code under the premise of 'if~then'. Since it is a code executed only when the condition is true, certain part can be repeated or stopped executing if you use control statement properly.

You are saying that the code in {} is executed only when the condition is true?

That's right. Basically, conditional statement is used in such form, but if the contents in code block is one sentence that ends with only one semicolon or is logically one unit, {} can be omitted.

```
if(conditional expression or value to use
for condition check)
{   //Start code block, Start executing
when true
Execution contents 1;
Execution contents 2;
 Execution contents 3;
}  // the end of code block
```

Umm… it's a little confusing.

Haha, it's confusing for me, too. Actually, although one instruction is carried out, it is more convenient to bind the conditional statement as a code block than to omit braces.

02

Learning Contents

Else comes with if

Then, what would happen when conditional statement is made but the condition is false? You can make nothing to happen, but can't you also put code that is executed only when the condition is false? Write the following source code as it is and try executing it.

Source Code

```
#include <stdio.h>
int main( )
{
        int a=91, b=7;
        printf("a=%d, b=%d\n", a, b);
        if(a%b==0)
        {
                printf("%d: It is a multiple of %d.\n", a, b);
        }
        else
        {
                printf("%d: It is not divided by %d.\n", a, b);
        }
        return 0;
}
```

➜ if(a%b==0)
If the value dividing a by b is 0 (that is, if a is a multiple of b), the code block in '{}' below the if statement is executed. .

➜If the value dividing a by b is not 0, the code block below else is executed.

Result

a=91, b=7
91: It is a multiple of 7.

This code can also omit the code block sign only if it is not confusing.

```
 if(a%b==0)
  printf("%d: It is a multiple of  %d.\n", a, b);
 else
  printf("%d: It is not divided by %d.\n", a, b);
  return 0;
}
```

Ah, I think I know. 'else' is the part executed only when the condition is false, isn't it?

Yes, it is. Directions are also very similar to if.

```
if(conditional expression or value to use for condition check)
{   // the start of the code block when the condition is true
Execution contents … ;

}  // the end of code block
else
{   // the start of the code block when the condition is false
Execution contents … ;

} // the end of code block
```

03

Learning Contents

Finding error with if~else

Shall we try a simple application? If you use if and else well, it can be used to find the error of the code

Source Code

```
#include <stdio.h>
int main()
{
        FILE *f = fopen("unknown.txt","r+");

        if(f == 0 )
        {
                printf("There is no file to read.\n");
                return -1;
        }
        else
        {
                printf("File is opened successfully.\n");
        }

        if( fclose(f) == 0 )
        {
                printf("File is closed successfully.\n");
        }
        else
        {
        printf("File is not closed successfully..\n");
        }
```

➜fopen() function returns '0' value when file is not opened successfully.

➜fclose() function returns '0' value when file is closed successfully.

▶ TIP

'return' ends the function itself is included and returns the value to where the function is called. 'return 0;' returns 0 value and 'return -1;' returns -1 value. When ending the program, return 0; means the success of normal ending and return -1; means the failure of normal ending.

Result

```
File is opened successfully.
File is closed successfully.
```

If you use fopen() function and fclose() function well, you can make a good program that you can know immediately if function works properly.

Exercise 01

Then, let's try a simple practice. To make a program that is entered with age to decide if someone attained adulthood if the input age if over 19, what source code should be put in the blank?

▶ TIP

Since July 1, 2013, by the revision of the civil law, anyone over the age of 19 is designated as an adult and is enfranchised.

Source Code

```
#include <stdio.h>
 int main()
{
        int a;
        printf("What is your age?\n");
        scanf("%d", &a); scanf("%d", &a);
        [          ]
                printf("You attained adulthood.");          return 0;
}
        [          ]
```

Result

```
What is your age?
19
You attained adulthood.
```

Uh? Why do I get an error?

You don't put semicolon(;) for the condition of if statement. I got this wrong all the time in the past. Just in case, check if you have put it.

Since it's 'over' the age of 19, I have to use '>=' for the condition comparison operator.

Exercise 02

Shall we make a program to find bigger number among two numbers entered? What code shall be put in the blank?

Source Code

```
#include <stdio.h>
int main()
{
        int a, b;
        printf("Enter two numbers. "); scanf("%d %d", &a, &b);
        scanf("%d %d", &a, &b);
        if(a>b)
        [          ]
                printf("Bigger number found : %d\n", a);
```

```
        }
        [          ]
        {
                printf("Bigger number found : %d\n", b);
        }
        return 0;
}
```

Result

```
Enter two numbers. 12 11
Bigger number found: 12
```

Exercise 03

Let's make a program, entered with one integer, decides if the number is a multiple of 4, and prints 'YES' if it is and 'NO' if it is not. Do you know what source code to put in the blank?

Source code

```
#include <stdio.h>
int main()
{
        int a;
        printf("Enter a random number. "); scanf("%d", &a);
        scanf("%d", &a);
        [          ] printf("%d: a multiple of  4 NO\n", a); printf("%d: a multiple of 4 YES\n", a);
        [    ]printf("%d: a multiple of 4 YES\n", a); return 0;

}
```

Result

```
Enter a random number. 24
24: a multiple of 4 YES
```

How should a multiple of 4 be decided?

Isn't it a multiple of 4 if the remainder of a number divided by 4 is 0?

Ah! I can use %! How typical honor student you are!

Exercise 04

This time, it's a program, entered with two integers, checks if each of the two numbers is a multiple of 6. It's more difficult than the one earlier, but you should write down what to put in the blank.

▸ TIP

Since there are two numbers, you need to write conditional statement twice.

Source code

```
#include <stdio.h>
int main()
{
        int a, b;
        printf("Enter two numbers. "); scanf("%d %d", &a, &b);
        scanf("%d %d", &a, &b);
        [          ] printf("%d: a multiple of 6 NO\n", a);
        [    ] printf("%d: a multiple of 6 YES\n", a);
        [          ] printf("%d: a multiple of 6 NO\n", b);
        [    ] printf(%d: a multiple of 6 TES\n", b);
        return 0;
}
```

Result

Exercise 05

This is the last problem. The admission fee of the amusement park in this neighborhood is 7,500 won per person. Let's make a program to enter the amount of money you have, and if two people can enter with that amount, it prints 'You can enter.', and if the amount is not enough, it shows the amount of money short.

Source cod

```
#include <stdio.h>
int main()
{
        int money=0;
        printf("How much do you have?\n");
        scanf("%d", &money);
        if(money>=7500*2)
        {
                printf("%dwon can enter.", money);}              }
        else
        {
                [                  ]
                printf("You are short of  %d won.", money);
}
        }
        return 0;
}
```

Result

```
How much do you have?
12500
You are short of 2500 won.
```

DAY 13.

When there are more conditions?

01
Learning Contents

What if conditional statement is nested?

When there is one condition, 'if' can be used, but actually in programming, there are more cases with a number of conditions. In that case, what should be done? The answer is simple. You have to use 'if' several times! Try to copy the following source code. It is a program to decide if input number is a multiple of 15

Source code

```
#include <stdio.h>
int main()
{
    int a;
    printf("Enter a random number. "); scanf("%d", &a);
    scanf("%d", &a);
    if(a%3==0)
    {
        if(a%5==0)
        {
            printf("%d: it is a multiple of 15.\n", a);
        }
    }
    return 0;
}
```

➜ int a; declare as integer variable (0)

➜ if(a%3==0) Check if divided by 3

➜ if(a%5==0) Check if divided by 5

If integer entered with overlapping use of selection execution structure is divided by 3, divide it again with 5 and print if it is a multiple of 15.

Result

```
Enter a random number. 15
15: it is a multiple of 15.
```

Like this, adding another conditional statement in the conditional statement is nested if statement. It is checking if one condition is true, and if the condition is false, it checks another condition.

But the code looks complicated.

Yes, so when you write nested if statement, you should write the code block accurately.

By the way, when I enter a number which is not a multiple of 15 in this program, no result comes out.

Yes. Then, what code should be added?

else

02
Learning Contents

Nested if-else statement

That's right. Then let's find out about the structure using nested if-else in if statement. Write the following source code and then compile it

Source code

```
#include <stdio.h>
int main()
{
        int a;
        printf("
       scanf("%d", &a);
       if(a%3==0)
       {
              if(a%5==0)
              {
                     printf("%d: It is a multiple of 15.\n", a);
              }
              else
              {
                     printf("%d: It is a multiple of 3 which is not divided
 by 15.\n", a);
              }
       }
        return 0;
}
```

➜ if(a%3==0)

Selection execution structure is used by overlapping. If input integer is divided by 3, then it is divided by 5 again and the program prints "it is a multiple of 15" if the result is true. If not, it prints "It is a multiple of 3 which is not divided by 15."

Result

```
Enter a random number. 18
18: It is a multiple of 3 which is not divided by 15.
```

03
Learning Contents

New nested structure

This code is a little different in structure from the previous one. If another if-else is used in if structure for the previous one, this one is in form that if is in else statement. You can use them properly depending on the condition you want to decide.

Source code

```
#include <stdio.h>
int main()
{
        int a;
        printf("Enter a random number. ");
        scanf("%d", &a);
        if(a%3==0)
        {
              printf("%d: It is a multiple of 3.\n", a);
        }
        else
        {
              if(a%5==0)
              {
                    printf("%d: It is a multiple of 3 which is not divided
by 5.\n", a);
              }
        }
        return 0;
}
```

➜ if(a%3==0)

Selection execution structure is used by overlapping. If input integer is divided by 3, the program prints "it is a multiple of 3" Or else the integer is divided by 5 again and when the condition is satisfied, it prints "It is a multiple of 3 which is not divided by 3."

Result

```
Enter a random number. 15
15: It is a multiple of 3.
```

04
Learning Contents

Omitting code block

There is the case it is possible to omit code block in the Nested if statement, just like typical conditional statement. It's a little more convenient when coding, but you should be careful in using it since it can be confusing later. Try it yourself.

Source code 1

```
#include <stdio.h>
int main()
{
        int a;
        printf("Enter a random number. "); scanf("%d", &a);
        scanf("%d", &a);
        if(a%3==0)
        {
              printf("%d: It is a multiple of 3.\n", a);
        }
        else
        {
              if(a%5==0)
              printf("%d: It is a multiple of 3 which is not divided by 5.\n", a);
        }
        return 0;
}
```

Result

```
Enter a random number. 25
25: It is a multiple of 5 which is not divided by 3.
```

Source Code2

```
#include <stdio.h>
int main()
{
	int a;
	printf("Enter a random number. ");
	scanf("%d", &a);
	if(a%3==0)
		printf("%d: It is a multiple of 3.\n", a);
	else
		if(a%5==0)
			printf("%d: It is a multiple of 5 which is not divided by 3..\n", a);
	return 0;
}
```

Result

```
Enter a random number. 25
25: It is a multiple of 5 which is not divided by 3.
```

Source Code 3

```
#include <stdio.h>
int main()
{
	int a;
	printf("Enter a random number. ");
	scanf("%d", &a);
	if(a%3==0)
		printf("%d: It is a multiple of 3.\n", a);
	else if(a%5==0)
		printf("%d: It is a multiple of 5 which is not divided by 3.\n", a);
	return 0;
}
```

Result

```
Enter a random number. 25
25: It is a multiple of 5 which is not divided by 3.
```

The results are the same whatever ways you use.

Yes. However, you should pay attention because error can occur if the range of condition is overlapped or omitted since nested if statement is executed in order.

Exercise 01

Nested if statement is a little complicated so you'll need a practice. Let's say there is a school that awards a certification of merit to the person with the test score over 90, and awards a certification and a prize to the person with the test score over 95. Then, can you make a program that decides whether the person will receive a certification of merit or receive a certification and a prize when entered the score? Try it by filling in the blank.

Source Code

```
#include <stdio.h>
int main()
{
	int a;
	printf("Enter the score. ");
	scanf("%d", &a);
	                    
	{
		                
			printf("The person is a subject of awarding a certification of merit and a prize.\n");
		}
		else
		{
			printf("The person is a subject of awarding a certification of merit.\n");
		}
	}
	return 0;
}
```

Result

```
Enter the score. 95
The person is a subject of awarding a certification of merit and a prize.
```

Exercise 02

Then, let's make a program that shows a special sentence only to the person who received the perfect score (100) among people who received the score over 60. Fill in the blank, and execute it.

Source Code

```
#include <stdio.h>
int main()
{
	int a;
	printf("Enter the score. "); scanf("%d", &a);
	 scanf("%d", &a);
	                    
	{
		if(a==100)
			printf("Wow~it's a perfect score.\n", a);
		                
			printf("wow~excellent.\n", a);
	}
```

```
        else
                printf("Try a little harder.\n", a);
return 0;
}
```

Result

```
Enter the score. 100
Wow~ it's a perfect score.
```

Exercise 03

The program we'll make this time is a program to organize class A for students with scores of 90 or over, class B for those of 85 or over, and class C for those below 85 among students with scores of 80 or over; and to fail those with scores below 80. Can you fill in the blank?

Source Code

```
#include <stdio.h> int
main()
{
        int a;
        printf("Enter the score. ");
        scanf("%d", &a);
        [          ]
        {
                [          ]
                {
                        printf("%d is class A.\n", a);
                }
                else [          ]
                {
                        printf("%d is class B.\n", a);
}
                }
                [          ]
                {
                        printf("%d is class C.\n", a);
                 }
        }
        [     ]
        {
                printf("%d is failed.\n", a);
        }
        return 0;
}
```

Result

```
Enter the score.85
85 is class B.
```

Exercise 04

Let's get off the score problem now. This time, we'll make a program to find the smallest number after entering 3 numbers. Think carefully and fill in the blank.

Source Code

```
#include <stdio.h>
int main()
{
        int a, b, c, min; printf("Enter 3 numbers. ");
        scanf("%d %d %d" , &a, &b, &c);
        [          ]
        {
                [          ]
                [          ]
            else
                        min=c;
        }
        else
        {
            if[     ]
                        min=b;
            else
                        min=c;
        }
        printf("The smallest number is %d.\n",
        return 0;
}
```

Result

```
Enter 3 numbers. 85 90 100
The smallest number is 85.
Enter 3 numbers. 85 100 90
The smallest number is 85.
Enter 3 numbers. 90 100 85
The smallest number is 85.
```

Exercise

05

Since it's the last problem, I'll give a little difficult one. This one is a program that you enter 3 real numbers, not integers, and then it prints by rounding off the biggest number to four decimal places. Make the program to operate well by filling in the blank.

Source Code

```
#include <stdio.h>
int main()
{
        float a, b, c;
      printf("Enter 3 float numbers. ");
        scanf("%f %f %f", &a, &b, &c);
        [          ]
        {
              [          ]rintf("The biggest number: %.3f\n", a);
else printf("The biggest number: %.3f\n",  c);
        }
        [      ]
        {
              [          ]rintf("The biggest number: %.3f\n", b);
else printf("The biggest number: %.3f\n", c);
        }

        return 0;
}
```

Result

```
10.1234 12.1357 13.1369
The biggest number: 13.137 30.5 20.5 10.5
The biggest number: 30.500
90.5 100.5 88.7
The biggest number: 100.500
```

Chapter 5.

C Programming

STEP 3.

DAY 14.

Think logically

01
Learning Contents

Learning Contents

I heard from Nick that you learned up to conditional statement! Today I'll tell you about logical operator! Since it's the best you try yourself, copy the following code exactly!

Source code

```
#include <stdio.h>
int main()
{
        int a=0, b=1;

        printf("a && b = %d\n", a && b);
        printf("a || b = %d\n", a || b);
        printf("!a = %d\n", !a);
        printf("!b = %d\n", !b);
          return 0;
 }
```

➜ a && b
When both a and b are true (1), it shows true result.

➜ a || b
When any one of a and b is true (1), it shows true result

➜ !a
It shows the opposite value of a.

Result

```
a && b = 0 a || b = 1
!a = 1
!b = 0
```

How is it? Do you think you understand it once you give it a try?

No… It seems somewhat like comparing operator, but …

Yes! The part that result comes out as true or false seems similar but it's actually different! Logical operator plays a role of showing new result with two values that are true or false. I'll explain one by one so listen carefully!

&& is also called and operation, and 'a and b' is an operator which gives a true result value when both a and b are true. || is also called or operation, and 'a or b' is an operation which give a true result value when one of a and b is true. !, also called not operation, shows the opposite value of operand value as a result, similar to comparing operation. Now you understand all, right?

(In a low voice) Why is she so excited?

Oh, and don't forget that it's like whether it's 1 or 10, all are recognized as 1 if it is not 0 in logical operation!

* If you want to know about bitwise operator, see 'Learn more!' on page 130.

02
Learning Contents

The Use of Logical Operator

Let's make a program to end the PC! You have made something similar, right? This time, we'll make in a form that PC is turned off when the alphabet Y is entered, and is not turned off when N is entered. Since the capital letter and small letter are treated as different letters in C language, you should make it work normally whether capital or small letter is entered!

Source code

```
#include <stdio.h>
#include <stdlib.h>
 int main()
{
        char s = '\0';

        printf("If you want to turn off the PC (Y/y) \n If you don't want, press (N/n)! \n");

        scanf("%s", &s);

        if( s=='Y' || s=='y')
        {
                printf("Turn off PC.\n");
                system("start \"shutdown\" C:windows\\system32\\shutdown.exe /s");

        }
        else if( s=='N' || s=='n' )
        {
                printf("Do not turn off PC.\n");

        }
        else
        {
                printf("You entered wrong letter.\n please enter (Y/y) or (N/n). \n");
        }
        return 0;
}
```

Result

```
If you want to turn off the PC, enter (Y/y)
 If you don't want, enter (N/n)!
```

Ah! It's convenient I can operate the same work whether I enter small letter or capital letter since || is used!

03
Learning Contents

The Master of Logical Operator!

Now is the turn to make a program to check password! You have seen this often in the website, right? We'll set the limit of the number of entering password to 3, and let it verify the password if right password is entered within the limit. In addition, the program must end automatically if the number of entering wrong password becomes more than the limit of the number of entering the password. Let's try it!

Source code

```
#include <stdio.h>
int main()
{
        int input = 0;
        int passwd = 1234;
        int itry = 3;

        printf("Please enter the password\n");

        scanf("%d",&input);

        if((itry = itry -1) && (input == passwd))
        {
                printf("The password is correct.");
                return 0;
        }
         printf("You have %d chances of entering.\n",itry);
        scanf("%d",&input);

        if((itry = itry -1) && (input == passwd))
        {
                printf("The password is correct.");
                return 0;
        }
                printf("You have %d chances of entering.\n",itry);
                scanf("%d",&input);
                scanf("%d",&input);

        if(input == passwd)
        {
                printf("The password is correct.");
                return 0;
        }
        printf("You entered wrong password for more than 3 times.\n",itry);

        return -1;
}
```

Result

```
Please enter the password
```

04
Learning Contents

Exceptions

What do you think? Isn't it fun? But there is something you need to consider when using logical operator. I'll modify the code we just made a little. Write the following code as it is and enter the wrong password more than 3 times.

Source Code

```
#include <stdio.h>
int main()
{
        int input = 0;
        int passwd = 1234;
        int itry = 3;

        printf("Please enter the password\n");
        scanf("%d",&input);
        if((input == passwd) && (itry = itry -1))
        {
                { printf("The password is correct.");
                return 0;
        }
          printf("You have %d chances of entering.\n",itry);
           scanf("%d",&input);
          if((input == passwd) && (itry = itry -1))
          {
                {printf("The password is correct.");
                return 0;
        }
          scan
           scanf("%d",&input);
          if((input == passwd) && (itry = itry -1))
          {
                printf("The password is correct.");
                return 0;
        }
                printf("You entered wrong password for more than 3 times.\n",itry);
                return -1;
}
```

Result

```
Please enter the password
```

Uh? The limit of the number of entering does not go down even though the wrong password is entered. Why does it happen?

In && operation, one of the two values is false, that is 0, the resulting value is always 0. Thus, if the left of && is 0, the computer does not operate the right one, process the resulting value to 0, and move on to the next one! That's why the part, itry=itry□1, on the right of && operator does not operate! You should always be careful on this part when you do && operation in C language!

Exercise 01

Well then, shall we practice what we've learnt? It's a program to decide if input number is a multiple of 12. Can you fill in the blank?

▶ TIP

A multiple of 12 is a number whose remainder is 0 when divided by either 3 or 4.

Source Code

```
#include <stdio.h>
int main()
{
        int a;
        printf("Enter a random number..");
        scanf("%d", &a);
        if(a%3==0 □ a%4==0)
        {
                printf("%d => It is a multiple of 12.\n", a);
        }
        else
        {
                {printf("%d => It is not a multiple of 12.\n", a);
        }
        return 0;
}
```

Result

```
Enter a random number..45 45 => It is
not a multiple of 12.
```

Exercise 02

Good job! Then the next problem! What code should go into the blank to get x, y, z like the result when a, b, c are given? Fill in the blank using logical operator and comparing operator!

Source Code

```
#include <stdio.h>
int main()
{
        int a=10, b=10, c=5;
        int x, y, z;
        x=(a==b) □ b==c);
        y=(a>b) □ >c);
        z= □ a+b>c);
        printf("a=%d b=%d c=%d\n", a, b, c);
        printf("x= %d\n", x);
        printf("y= %d\n", y);
        printf("z= %d\n", z);
        return 0;
}
```

Result

```
a=10 b=10 c=5
x=0
y=1
z=0
```

Exercise 03

Let's make a program to check if the last names of two people are the same after two names are entered! Don't forget you are dealing with characters, not numbers!

Source Code

```
#include <stdio.h>
int main()
{
	char in1[10];
	char in2[10];
	printf("Enter the first name. :");
	scanf("%s", in1);

	printf("Enter the second name. :"); scanf("%s", in2);
	scanf("%s", in2);

	if((                                    )
	{
		printf("The two names, %s, %s, you enter have them same last name.\n",in1 ,in2);
	}
	else
	{
		printf("The two names, %s, %s, have different last name.\n",in1 ,in2);
	}
	return 0;
}Result
```

▸ TIP

For English, one character takes up the size of one array in the char typed array, but for Korean, one character takes up the size of two arrays. Be sure to remember since it's an important difference!

Result

```
Enter the first name : Gil-Dong Hong Enter the second name : Gil-
Soon Hong
The two names, Gil-Dong Hong, Gil-Soon Hong, you enter have the same last name.
```

Exercise 04

Now is the turn to make a program to decide if a person is a teenager. Make the program operate well by filling in the blank!

Source Code

```
#include <stdio.h>
int main()
{
	char ans;

	printf("Are you a teenage girl?\n");
printf("If yes, enter Y; if no, enter N ~\n");

	ans =      

	if(ans=='y' || ans=='Y')
	{
		printf("You are a teenage girl.\n");
	}
	else if(                                    )
	{
		printf("You are not a teenage girl.\n");
	}
	else
	{
		printf("It's a wrong entry.\n");
	}

	return 0;
}
```

▸ TIP

C language makes a decision by distinguishing capital and small letter of input value. To get a re- sult regardless of capital and small letter, you have to use logical operator. Think of a function which reads one character from keyboard.

Result

```
Are you a teenage girl?
If yes, enter Y; if no, enter
N~Y You are a teenage girl.
```

Exercise 05

Then let's make a program to find out if entered year is leap year! Think carefully since you just have to fill in the blank!

Source Code

```
#include <stdio.h>
int main()
{
	int year;

	printf("Enter the year (number only): "); scanf("%d", &year); scanf("%d",
	&year);

	if( )                                    
		printf("%d is a leap year.\n", year);
else
printf("%d is not a leap year.\n", year);

	return 0;
}
```

Result

```
Enter the year (number only) : 2014
2014 is not a leap year.
```

Uh, what is a leap year?

Huh? You don't know what a leap year is? Ah, I thought the farmer told about things like this! Leap year refers to the year with February 29 by adding one more day to solar calendar once every 4 years. The year which is a multiple of 400 or the year which is a multiple of 4 but not a multiple of 100 is a leap year. Okay?

Learn more! How is digital world expressed?

Have you heard about bit? Bit is an acronym of Binary Digit. It is a unit of binary system which expresses the entire numbers in the world with 0 and 1. In binary system, a single figure is expressed with the two numbers, either 0 or 1. A number used in binary system as such is called 1 bit.

How can the entire numbers be expressed with binary numeral which only has 0 and 1? 0 and 1 is calculated the same as 0 and 1 of decimal numeral. As 9 plus 1 is expressed to 10 by raising one digit, 1 plus 1 is expressed to 10 by raising a digit, 10 plus 1 is to 11, and 11 plus 1 is expressed to 100. In other word, 2 of decimal numeral becomes 10 of binary numeral, and 3 of decimal numeral becomes 11 of binary numeral.

Binary numeral	Decimal Numeral
0	0
0 + 1 = 1	1
1 + 1 = 10	2
10 + 1 = 11	3
11 + 1 = 100	4

Why dose computer use such an inconvenient and simple calculation? Actually, binary system is a more convenient way to calculate than decimal system. A mathematician in 17~18 centuries, Leibniz, discovered that calculation can be done more simply and quickly through binary system and organized the system about binary system, also called Bit.

Cards for hundreds place	0	1	2							
Cards for tens place	0	1	2	3	4	5	6	7	8	9
Cards for ones place	0	1	2	3	4	5	6	7	8	9

→ Total 23

The numbered cards required to express 0~255 to decimal number

Let's take an example of Bit by comparing it to card.
If it is supposed that all numbers from 0 to 255 should be expressed using cards numbered from 0 to 9, the following cards are required.

Cards for ten millions place	0	1								
Cards for millions place	0	1								
Cards for hundred thousandsplace	0	1								
Cards for ten thousands place	0	1								
Cards for thousands place	0	1								
Cards for hundreds place	0	1								
Cards for tens place	0	1								
Cards for ones place	0	1								

→ Total 16

The number cards required to express 0~255 to binary numberal

If you extend the number you have to express to up to 1023, it makes a big difference. In case of expressing to decimal numeral, 10 more cards are needed since one digit is increased and since you only need one card for the thousands place, total 32 cards are required.

Cards for thousand place	0	1								
Cards for hundred place	0	1	2	3	4	5	6	7	8	9
Cards for tens place	0	1	2	3	4	5	6	7	8	9
Cards for ones place	0	1	2	3	4	5	6	7	8	9

→ Total 32

The numbered cards required to express 0~1023 to decimal number

In case of binary numeral, two digits are increased but the extra cards you need is only 4. You need the total 20 cards. As the number you express become bigger, the gap of card you need becomes more and more widen.

Cards for thousand millions place	0	1
Cards for hundred millions place	0	1
Cards for ten millions place	0	1
Cards for millions place	0	1
Cards for hundred thousands place	0	1
Cards for ten thousands place	0	1
Cards for thousands place	0	1
Cards for hundreds place	0	1
Cards for tens place	0	1
Cards for ones place	0	1

→ Total20

The number cards required to express 0~1023 to binary numberal

It's easy to understand if you think of the number of this card as the capacity of memory or hard disk we use. Will you store data using decimal system or binary system?
Binary system is much simple in calculation. In decimal numeral, to get an answer of 2^10, 2 has to be multiplied 10 times. However, a little different calculation can be used for binary numeral. It is adding ten 0 on the right of 1.

2^10 in decimal = 1024 in decimal	You can find out the result by multiplying 2 ten times.
10^1010 in binary = 100000000000 in binary	You can find out the result just by adding ten 0 on the right of 10.

When using decimal numeral, to multiply 2^10 to some number, the value of 2^10 needs to be multiplied to the corresponding value again. However, in binary numeral, calculation is done by just adding ten 0 on the right of the value to multiply, just like the example earlier. It's because in binary numeral, whenever the digit increases by one, the result like multiply 2 comes out.

For operation of multiplying 2^n to some number, binary numeral can be calculated much faster than decimal numeral since the result comes out if original number is moved to the left by n. This is called LEFT SHIFT operation.

LEFT SHIFT Operation
10 x 2^10 => 0000000000001010 -> 0010100000000000

C language code of LEFT SHIFT is <<. It is used as following.

```
printf("%d", 10 << 10 );
```

10<<10 means left shifting 10 by 10. In other word, the result of multiplying 2^10 to 10 comes out. You can think of division as the opposite of multiplication. When dividing by 2^n, original number is moved to the right by n.

RIGHT SHIFT Operation
(10 x 2^10) / 2^5 -> 0010100000000000 -> 0000001010000000

C language code of RIGHT SHIFT is >>. It is used as following.

```
printf("%d", 10 >> 10 );
```

10>>10 means right shifting 10 by 10, and the result of dividing 10 by 2^10 comes out.

In addition, Leibniz discovered all four fundamental arithmetic operation including subtraction can be done using just addition by proving subtraction can be operated with addition using two's complement.

Now, shall we find out about logical operation using binary system?

& operation is a bit logical operator that 1 comes out only when both of two numbers are 1. When it is com-posed of a number of bits, the result of logical operation for each bit place is shown.

Operation to perform	11 & 4	11 & 14
Operation process	Decimal 11: 1 0 1 1 Decimal 4: 0 1 0 0 Operation result: 0 0 0 0	Decimal 11: 1 0 1 1 Decimal 14: 1 1 1 0 Operation result: 1 0 1 0
C language Code	printf("%d\n",11&4); → Result: 0	printf("%d\n",11&14); → Result: 10

| operation is a bit logical operator that 1 comes out if one of two numbers is 1. Similarly, when it is composed of a number of bits, the result of logical operation for each bit place is shown.

Operation to perform	11 | 4	11 | 14
Operation process	Decimal 11: 1 0 1 1 Decimal 4: 0 1 0 0 Operation result: 1 1 1 1	Decimal 11: 1 0 1 1 Decimal 14: 1 1 1 0 Operation result: 1 1 1 1
C language Code	printf("%d\n",11|4); → Result 15	printf("%d\n",11|14); → Result: 15

~ is a bit operation changing 1 to 0 and 1 to 0. It's like ON/OFF of button. It can be useful for making comple- ment since it reverses each bit.

Operation to perform	~11	~15
Operation process	Decimal 11: 1 0 1 1 Operation result: 0 1 0 0	Decimal 15: 1 1 1 1 Operation result: 0 0 0 0
C language Code	printf("%d\n",~11); → Result: -12	printf("%d\n",~15); → Result -16

If you take a look at operation result, you can check the result is 1 less than changing the value before operation to negative number. Computer uses two's complement as a method to express negative number. As explained above, the method to make complement is reversing each place and then adding 1. Since 1 is not added after reversing each bit in the example above, value smaller by 1 came out.

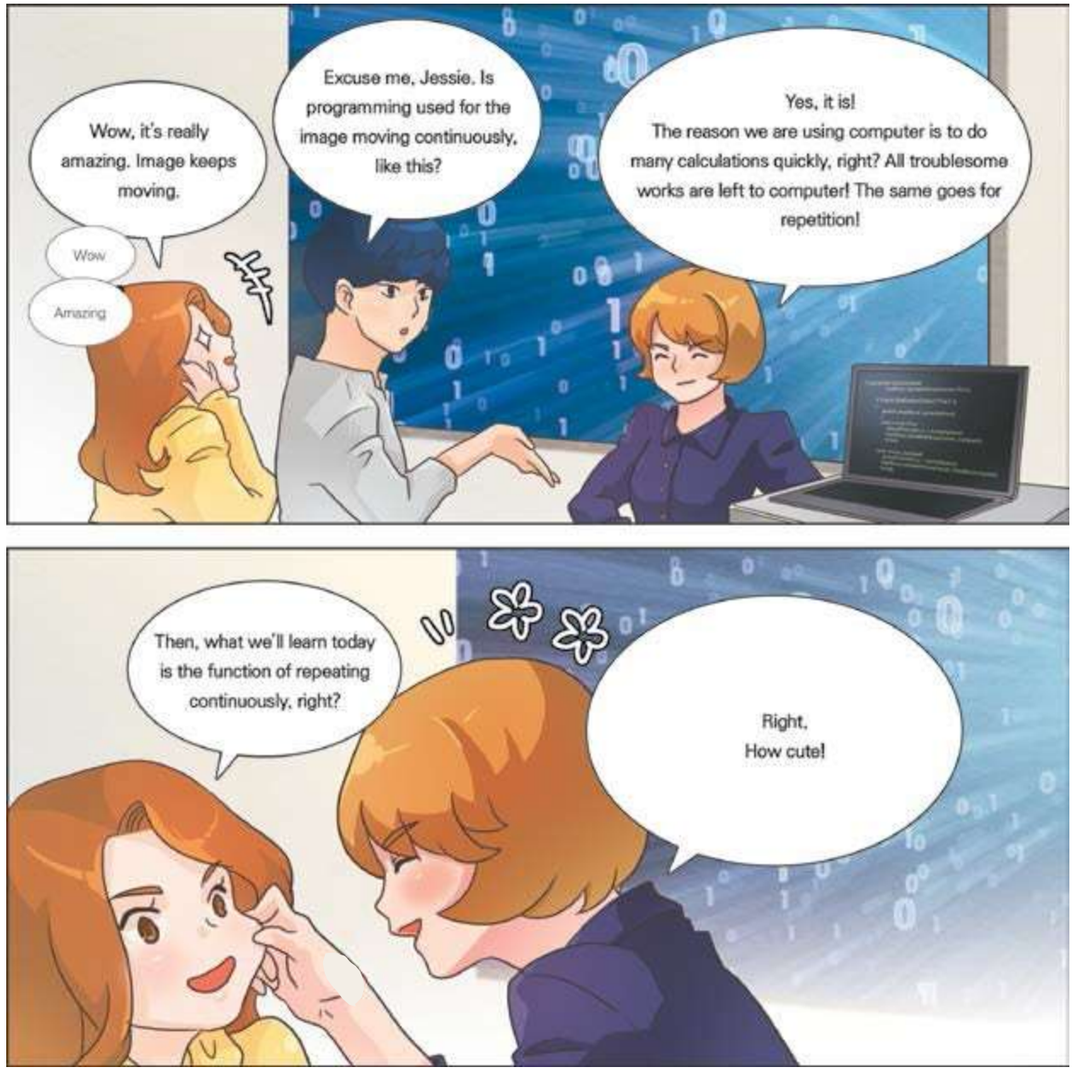
^ (XOR) is a bit logical operator that the result is 1 if one of the two is 1 and the result is 0 if both are 1 or both are 0. When it is composed of a number of bits, the result of logical operation for each bit place is shown.

Operation to perform	10 ^ 5	10 ^ 10
Operation process	Decimal 10: 1 0 1 0 Decimal 5: 0 1 0 1 Operation result: 1 1 1 1	Decimal 10: 1 0 1 0 Decimal 10: 1 0 1 0 Operation result: 0 0 0 0
C language Code	printf("%d\n",10^5); → Result:15	printf("%d\n",10^10); → Result: 0

The present computer uses this binary system as a base. It is processing all calculation with Bit unit. Computer, smartphone, and internet, etc. we uses nowadays can be said as the world made by 0 and 1.

DAY 15.

Do, Do Again, and Keep Repeating



01

Learning contents

The Basics of Repetition tatement

Shall we start? As we learned the last time, we'll make a program that repeats continuously. Like this, repeating the operation which is carried out once is called loop in programming! It's the term we'll keep using from now on so make sure to remember and copy the source code!

Source Code

```
#include <stdio.h>
int main()
{
        int count = 10;

        while(count)
        {
                count = count - 1;
                printf("count %d\n",count);
        }

       return 0;
}
```

Result

```
count 9
count 8
count 7
count 6
count 5
count 4
count 3
count 2
count 1
count 0
```

Wow! One line of source code is repeated ten times!

Right! Do you see how the number of iterations is set?

it is set with this count variable, right?

That's correct! I'll give you one question. What result would come out if the initial value of count variable is -10, not 10, and variable increases by 1 whenever the count phrase is printed?

Uh… I'm not sure.

Was it too difficult? Then, I'll tell you in detail. The while loop does not execute the contents in code block if the value in (), parameter, becomes 0 and execute the next code of the code block!

```
while(count)
{
    statement1;
    statement2;
    ...
    statement5;
}
statement6;
```

And when parameter is not 0, the contents in code block is executed once first and then check the value of parameter again. Thus, it's possible to repeat also in case of negative number!

```
while(count)
{
    statement1;
    statement2;
    ...
    statement5;
}
statement6;
```

Ah, so if the parameter in () is not 0, it keep repeating until the parameter becomes 0!

That's right! If parameter does not become 0 forever or is added as a constant not a variable due to incorrect coding, the code block is repeated endlessly! This is the horrible endless loop!

```
while(1)
{
    statement1;
    statement2;
    ...
    statement5;
}
statement6;
```

What happens if it becomes endless loop?

It executes the code block infinitely until the program is ended from the outside! It is indeed a nightmare!

Could it be the reason for computer program being down sometimes!

There's such case! I've made it a lot. Hehe!] By the way, there is another type of while loop which is do~while loop. If used in this type, code block below do loop is executed once unconditionally before checking the parameter of while loop. You will get to use it often!

```
do
{
Execution state-
ment;
}
while(condition)
```

02

Learning Contents

Endless Loop

Endless loop can be made by a mistake but it is also used to make a program that operates continuously. Let's make a program that uses endless loop! The following code is a program that shows the input value by squaring. Check the code by trying on your own!

Source Code

```
#include <stdio.h>
int main()
{
      int input = 0;

      while(1)
      {
        printf("Enter the number you want to square:");
        scanf("%d",&input);
        printf("The square of %d is %d.\n",input,input*input);
      }

      return 0;
}
```

Result

```
Enter the number you want to square: 10
The square of 10 is 100
Enter the number you want to square: 100
The square of 10 is 10000
Enter the number you want to square:
```

In this way, the new value can be entered again although the value is entered and the result comes out. The programs which wait for entry continuously or repeat motion endlessly are constructed using endless loop. Operating system or communication program as well as the media art work we saw earlier are such cases!

By the way, how do we end it? We cannot do only calculation of square forever.

03

Learning Contents

Breaking Out of the Endless Loop

You cannot end the program like this every time, right? Let's make a function of breaking out of endless loop in the program itself! It is a program to end the endless loop when input number corresponds to the password!

Source Code

```
#include <stdio.h>t
main()
{
        int count = 0;
        int passwd = 1234, input = 0;

        while(1)
        {
                scanf("%d",&input);
                count++;
                if(input == passwd)
                {
                        break;
                }
        }

          printf("You've broken the endless loop on the %d attempt.\n",count);

        return 0;
}
```

Result

```
12
34
1234
You've broken the endless loop on the third attempt.
```

Break statement functions as stopping the repetition and getting out of code block! This program repeats endlessly since there is the constant, 1, as a parameter of while loop, but the endless loop can be broken when the password is right by conditional statement!

If repetition statement is broken, the next syntax, printf() is executed.

That's right!

```
while(1)
{
        scanf("%d",&input);
        getchar(); count++;
        if(input == passwd)
        {
                break;

        }
}

printf("You've broken the endless loop on the %d attempt.\n",count);

return 0;
```

04 Learning Contents

By the way, Jessie, what is 'count++'? Is it like a beef grade?

Ah, I didn't tell you about that part! 'count++' has the same meaning as 'count=count+1'! It's called increment and decrement operator.

Count++;	count = count + 1;

Then, is there a '- -'?

Of coure! You can tell right away once you see but 'count--' has the same meaning as 'count=coun-1'.

count--;	count = count - 1;

Wow, it could be more convenient if I have known this earlier!

* If you want to know about increment and decrement operator, see 'Find out more!' on page 145.

Like the break statement earlier, an instruction that plays a role of getting out of repetition is called the jump statement. Let's use 'continue statement', one of the jump statement' that has a similar function. The following source code checks the input number from the ones place so if the number matches, it prints the matching digit; if not, number is entered again. Try it yourself!

Source Code

```
#include <stdio.h>
int main()
{
        int count = 0;
        char passwd[4] = "123";
        char input[4] = {0,};

        while(1)
        {
                count++;
                printf("enter the 3-digit password:\n");
                scanf("%s",input);

                if(input[2] != passwd[2])
                {
                        continue;
                }

                printf("ones place number is correct.\n");
```

▶ TIP

In 'char passwd[4] = "123";', passwd is reset as character. You should be careful because if the value which is more than 3 digits is entered, NULL character cannot enter the end of character string and thus it does not print normally.

```
        if(input[1] != passwd[1])
        {
            continue;
        }
        printf("tens place number is correct.\n");
        if(input[0] != passwd[0])
        {
            continue;
        }

        printf("all place numbers are correct.\n"); break;

    }
    printf("You've broken the endless loop in %d attempt.\n",count);
    return 0;
}
```

Result

```
Enter the 3-digit password:
123
Ones place number is correct.
Tens place number is correct.
All place numbers are correct.
You've broken the endless loop in the first attempt.
```

Do you understand the function of continue statement? It plays a role of returning to the very first of the loop, not executing the rest of code block, after breaking the loop! Like this, if input and password does not correspond, break is not executed and endless loop is repeated continuously!

```
while(1)
while(1)
{
    if( input != passwd )
    {
        continue;
    }
    break;
}
```

Uh? It's somewhat similar to if~else, isn't it?

Yes, it is! We can use if~else in the code we just made instead of continue. You're smarter than I thought!

God, is that a complement?

There are things like this in jump statement. It will be useful if you know these!

break	It stops the loop and executes the next syntax outside the {}.
continue	It stops the loop and returns to the beginning of the loop.
goto	It moves to where the label is and executes the next syntax.

Exercise 01

Then, what will happen in this case that another while loop is in the code block repeating with while loop! Try it yourself and check the result!

Source Code

```
#include <stdio.h>
int main()
{
    char star[3]= {'$','#','*'};

    while(1)
    {
        while(1)
        {
            while(1)
            {
                printf("%c",star[2]);
            }
            printf("%c",star[1]);
        }
        printf("%c",star[0]);
    }

    return 0;
}
```

Why don't you apply while you're at it! Can you make different pattern to repeat by adding the jump statement, without modifying this code?

The first pattern: #

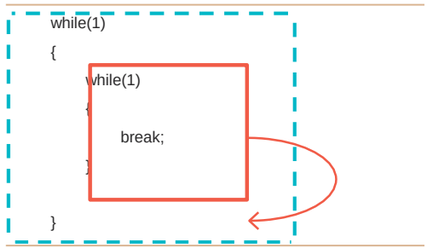
The second pattern: $

The third pattern: *#

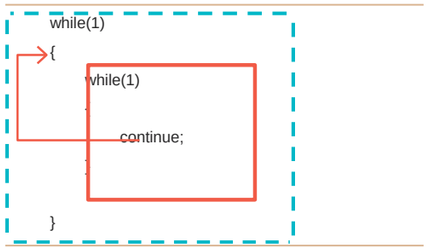
The fourth pattern: *#$

How to use goto statement

When while loop is nested like practice 01, you can break or restart the corresponding loop if break statement or continue statement is used, but you cannot break out of the while loop outside the corresponding loop. What should be done to break out of all while loops at once in nested while loop?

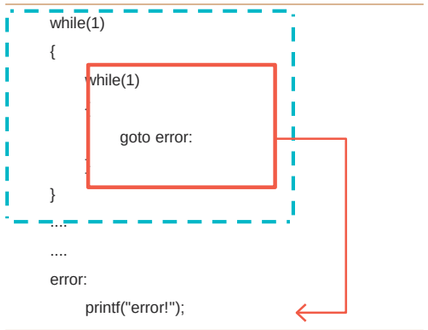


When using break



When using continue

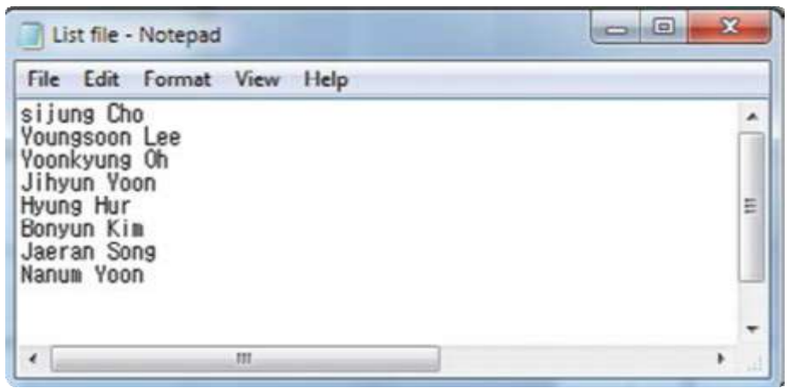
In this case, goto statement can be used. 'goto' statement unconditionally moves to where the label is and start the code again from that position. Thus, it can be useful in the nested loop structure. However, it can be the code that is difficult to understand the order of executing program if you abuse the goto statement so you should refrain from using it as much as you can.



When using goto

Exercise 02

This time, let's solve a unique problem! Write down the names of your classmates by using notepad and save it. The code below is a program that prints the name of friends whose last name is the same as input last name. Take a look at this and fill in the blank!



▶ TIP

When you make list file with notepad, you need to finish the input with Enter key after entering the last data.

Source Code

```
#include <stdio.h>
int main()
{
	int count = 0;
	char name[20] = {0,};
	char list[20] = {0,};

	FFILE *f = fopen("List file.txt","r");
	    printf("Enter the last name you want to search:\n");
	    scanf("%s",name);

	while(1)
	{
		fscanf(f,"%s",list);
		if(feof(f)) break;
		if((name[0]==list[0]) && (name[1]==list[1]))
		{
			[          ]
			printf("%s\n",list);
		}

	}
	printf("Total %d people is searched.\n",count);

	fclose(f);

	return 0;
}
```

Result

```
Enter the last name you want to search
Yoon
Jihyun Yoon
Nanum Yoon
Total 2 people are searched.
```

· Name which is the same as input last name and the number of searched list are printed.

By the way, what is feof?

feof functions as returning 1 when the file that file descriptor variable in () indicates is read to the end. Thus, if you put conditional expression in while() loop and use with NOT(!) operation, it is repeated continuously until the file is read until the end. Do you understand?

Yes! You sure know everything.

Exercise 03

We are going to use the list of friends' name we made earlier this time! Let's find the friends whose first name as well as last name correspond. Fill in the blank!

Source Code

```
#include <stdio.h>
int main()
{
	int idx  = 0;
	int flag  = 0;
	char list[20] = {0,};
	char target[20] = {0,};

	FILE *f = fopen("List File.txt","r");

	scanf("%s",target);

	while(          )
	{
		fscanf(f,"%s",list);

		idx = 0;
		flag = 1;

		while(              )
		{
			if(target[idx] == list[idx])
			{
				idx++;
				continue;
			}

			flag = 0;
			break;
		}

		if(flag == 1)
		{
			printf("Input name is found: %s\n",list);
 return 0;
		}
	}

	printf("Input name is not found.");

	fclose(f);

	return 0;
}
```

Result

```
Gildong Hong
Input name is not found.
```

Make sure to remember that the character that means the end of character string is the null character (\ 0)!

Learn more! Using Increment and Decrement Operator

Increment and decrement operator is also called shortcut assignment operator. The shortcut assignment operators are the operators made to briefly express the four fundamental arithmetic operations and assignment operation. The shortcut assignment operator is as following.

Expression	Actual Contents	Expression	Actual Contents
a++;	a = a + 1;	a-=b;	a = a - b;
a--;	a = a - 1;	a*=b;	a = a * b;
a+=b;	a = a + b;	a/=b;	a = a / b;

The contents of ++ and -- which mean the increase or decrease vary depending on it is written in front of or behind the variable.

++Operator
It is called Increment Operator and the value of corresponding variable increases by 1.

Type	example	Meaning
Pre	++a	a=a+1
Post	a++	a=a+1

--Operator
It is called Decrement Operator and the value of corresponding variable decreases by 1.

Type	example	Meaning
Pre	--a	a=a-1
Post	a--	a=a-1

Pre-increment/decrement
As a form that increment and decrement operator is positioned in front of variable, it first changes (increase or decrease) the valuable by 1 and executes the instruction.

Post-increment/decrement
As a form that increment and decrement operator is positioned behind the variable, it changes (increases or decreases) the value of variable by 1 after executing the instruction.

Ex) example of pre-increment/decrement	Result
a = 1; printf("%d",++a);	2
a = 1; printf("%d",--a);	0

Ex) example of post-increment/decrement	Result
a = 1; printf("%d",a++);	1
a = 1; printf("%d",a--);	1

The shortcut assignment operator acts as preventing mistake of omitting assignment operator when increasing or decreasing the value of variable which is handled as ending condition in repetition statement.
In the example on the right, since assignment operator (=) is omitted in x, the value of parameter does not change and falls into endless loop. If you get in the habit of using shortcut assignment operator in increment and decrement expression of variable used as ending condition of r epetition statement like this, it will be very helpful in preventing mistakes.

Source Code

```
x=0;

while( x < 10 )

{
    x+1;

}
```

DAY 16.

Repeat Hard as Much as Set Number!

01
Learning Contents

Another repetition statement

This time, I'll tell you about repetition statement using for loop! It's similar to while loop in function itself but it's a little more convenient when using! Make the following source code as it is!

Source Code

```
#include <stdio.h>
int main()
{
        int i = 0;


        for(i = 1; i <= 10; i++)
        {
                printf("%3d",i);
        }

        return 0;
}
```

Result

```
= Print number from 1~10=
1 2 3 4 5 6 7 8 9 10
```

The 'for loop' is written as following.

```
for(i = 1; i <= 10; i++)
      1      2      3
{

        printf("%3d",i);

}
```

1. iThe first part of parameter is the part resetting the initial value of variable to use in condition of ending repetition. Usually, i is used but it doesn't matter you change this to whatever you want!
2. The second part is where the condition to check until when the repetition will continue goes in. If conditional expression of this part becomes false (=0), repetition ends.
3. The third part is where increment and decrement expression of variable that is used as condition goes in. It'll be more convenient if increment and decrement operator is used.

The code coded like this checks the condition if the value of variable i starting from 1 is less than or equal to 10 and increase the value of variable i by 1 as it repeats the code block continuously while the condition is true. Do you understand?

By the way, what does %3d which goes in printf statement mean? I have never seen this before.

Ah, that is a format specifier in a sense of writing decimal integer up to three digits (hundreds place).

It seems similar to while loop…what's the difference?

For while loop, there is only ending condition as parameter. Increment and decrement expression goes in the code block, right? But for for loop, initial value, ending condition, and increment and decrement expression all go in the parameter so it's easy to know the structure of repetition statement at once.

Ah, so when it is only subjected for a fixed range or when the range has to be checked carefully, for statement will be more convenient, right?

That's right! You have to choose according to the purpose of use!

02
Learning Contents

Using repetition statement

Shall we practice one more time? This time, let's make a repetition statement with character! We'll make a program to print from alphabet A to Z. Take a look at the following source code and make it so you can print the number corresponding to each alphabet letter!

Source Code

```
#include <stdio.h>
int main()
{
	char ch = '\0';

	for(ch='A'; ch<='Z'; ch++)
	{
		printf("%6c:%6d\n", ch, ch);
	}

	return 0;
}
```

Result

```
A:65
B:66
C:67
⋮
Y:89
Z:90
```

Uh… Why do we get characters when the numbers are increasing? I don't understand.

I think we have learnt before that characters are expressed as numbers in C language.

That's right! C language essentially expresses character using the rule called 'American Standard Code Information Interchange (ASCII) code'. A specific number is set by each character registered in ASCII code! That's why big and small of value between character exist and you can change to different character though operation!

Then, the number next to alphabet on resulting screen corresponds to the letter!

That's correct! In for loop, initial value of ch became A so it prints from a in order!

03
Learning Contents

Array and Repetition

Shall we make a repetition statement using array? The following code is a program to print the value saved in two-dimensional array by squaring. Look at it carefully and copy it so you won't get confused!

Source Code

```
#include <stdio.h>
int main()
{
	int i = 0;
	Int j = 0;
	int array[3][3] = {{1,2,3},{4,5,6},{7,8,9}};
	for(i = 0; i < 3; i++)

		for(j = 0; j <3; j++)
		{
			array[i][j] *= array[i][j];
			printf("array[%d][%d] = %2d\n", i , j ,array[i][j]);
		}

	return 0;
}
```

Result

```
array[0][0] =  1
array[0][1] =  4
array[0][2] =  9
array[1][0] = 16
array[1][1] = 25
array[1][2] = 36
array[2][0] = 49
array[2][1] = 64
array[2][2] = 81
```

When using two-dimensional array, it's more convenient if repletion statement is written in a form to put another for loop in the for loop. Why? You don't seem to understand.

Haha…

I don't understand. Can you explain in plain words?

Well, it'll be easier to understand if I organize it like this. Since two-dimension array is a structure of array holding array, all data can be repeated if the loop repeating the range of the second index is made to repeat by the range of the first index! Just like this!

```
array[ i ][ j ]

array[ 0 ][ 0 ]
array[ 0 ][ 1 ]
array[ 0 ][ 2 ]

array[ 1 ][ 0 ]
array[ 1 ][ 1 ]
array[ 1 ][ 2 ]

array[ 2 ][ 0 ]
array[ 2 ][ 1 ]
array[ 2 ][ 2 ]
```

```
for(i = 0; i < 3; i++)

for(j = 0; j < 3; j++)
```

Oho. I get it.

If you get used to the process of placing repetition statement in repetition statement like this, you can handle three-dimensional or four-dimensional array easily, too! Now do you understand?

I think I understand with my head… but I'm not sure.

Then, let's practice by solving problem!

Exercise 01

If you use repetition statement, you can make a form using characters. Make a program that prints stars until 15th row using printf as following! You have to put star as much as the number of each row, which means 1 start for the first row, 5 stars for 5th row, and 15 stars for 15th row! Can you try this?

Result

```
*
**
***
****
*****
******
*******
********
*********
**********
***********
************
*************
**************
***************
```

Ugh, do I have to make it without an example code now?

Think of what you have learnt earlier! You can do it!

Exercise 02

Now let's do it in opposite! Flip the form so 15 stars are in the first row and one star is on 15th row! You can do it well, right?

Result

```
***************
**************
*************
************
***********
**********
*********
********
*******
******
*****
****
***
**
*
```

Exercise 03

Good job! I'll show you one method of applying code you made. Fill in the blank with the code for making a shape with stars that we've made earlier and execute it. You'll get a wonderful result!

Source Code

```
#include <stdio.h>
#include <stdlib.h>
int main()
{

        FILE *f = fopen("stars.html","w");

        int i = 0;
        int j = 0;

        fprintf(f,"<html>");
        fprintf(f,"<marquee direction=UP>");

        fprintf(f,"*****");

        fprintf(f,"</marquee>");
        fprintf(f,"</html>");

                (  );

        system("stars.html");

        return 0;
}
```

Result

➜ The shape of * printed in web browser moves up on the screen.

Wow! It's like a media art work that you made!

Exercise 04

Now shall we solve a unique problem? Fill in the blank to make a program to change all the small letters stored in array to capital letters!

Source Code

```
#include <stdio.h>
int main()
{
        int i = 0;
        int j = 0;

        char list[5][10] = {"apple","orange","banana","cocoa","pican"};

        for(i = 0; i < 5; i++)
        {
                printf("%s\n",list[i]);
        }

        return 0;
}
```

Uh, how can you make small letters to capital letters?

I told you earlier but all characters in C language have a corresponding number. The characters a through z correspond to number increasing by 1. Then you can just check how many integers should be added to the small a to make the capital A, right?

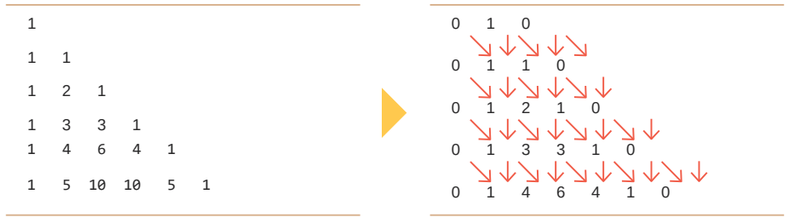
Then, how do you know the corresponding word is completed when changing small letter to capital letter?

Let me give you a hint! The end of character string all end with \ 0!

Exercise

05

Now it is the last problem! The following table is an array called Pascal's triangle. 1 comes in the first row, and the value adding the above number in the same column and the left number of that comes from the next row. If there's no number right above in the same column, consider it as 0.



Do you understand? Let make a program repeating array which is repeated like this until 10th row!

Source Code

```
#include <stdio.h>
int main()
{
	int i = 0;
	int j = 1;

	long long b[100][100] = {0,};

	for(j = 1; j <= 10; j++ )
	{
		for(i = 0; i < j; i++)
		{
			
		}
		printf("\n");
	}

  return 0;
}
```

DAY 17.

Judy's Project

Exercise 01

Listening to Judy, there are a lot of functions required! There should be a function to register the stray dog as well as a function to search by breeds, age, and color. Let's make a part to select which function to do once program is executed!

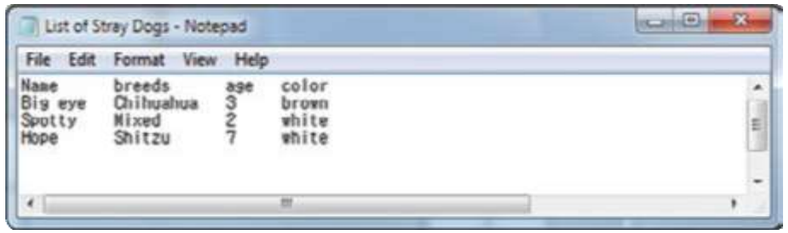
Source Code

Result

```
Select a function
1. Register the stray dog
2. Search by breeds
3. Search by age
4. Search by color
```

Exercise 02

Next, let's make a function to make a list of stray dogs! You should make a function to save as file if the name, breeds, age, and color are entered. Once you finish making the program, check it by registering the following list.



Source Code

Result

```
Select a function
1. Register the stray dog
2. Search by breeds
3. Search by age
4. Search by color
Enter the name of the stray dog to register
Enter the breeds of the stray dog to register
Enter the age of the stray dog to register
Enter the color of the stray dog to register
The entry is completed
```

Exercise 03

Now let's make a function to search by breeds from the saved list!
I'll be better if it prints name, breeds, age, and color of the searched stray dog.

Source Code

Result

```
Select a function
1. Register the stray dog
2. Search by breeds
3. Search by age
4. Search by color
Enter the breeds to search
Chihuahua Search result
Big eye Chihuahua 3 brown
The search is completed
```

Exercise 04

Now it's almost done! Let's make a function to search by age and color!

Source Code

Result

```
Select a function
1. Register the stray dog
2. Search by breeds
3. Search by age
4. Search by color
Enter the age to search 5
Search result
There is no search result
The search is completed
```

DAY 18.

My Own Project

Exercise 01

This time, let's make a special program of your own! Write down about what program you want to make!

NOTE

Exercise 02

Go over what functions and instruction you need to make the program and write down what variables you need.

NOTE

Exercise 03

Draw a simple image of the structure of the entire program. Then, write down what instructions to use in each part.

NOTE

Exercise 04

Now, it’s time to write program on your own! Make a program on your own and try executing it. Good luck!

NOTE

Samsung Innovation Campus aims to create a better world through youth education.

If you have any questions, please feel free to contact us.

SIC Program related enquiry : Seonghee Kang (sh1128.kang@samsung.com)

Enabling People

SAMSUNG