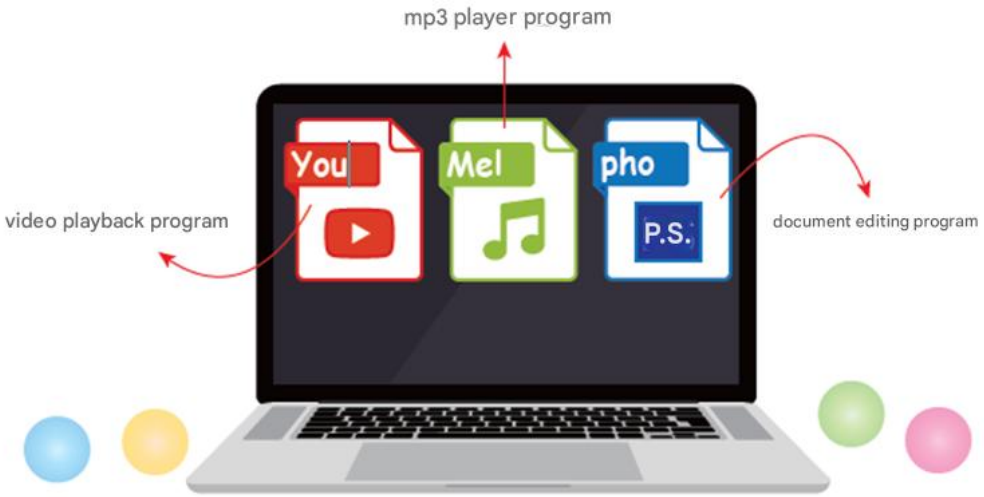
Ch.1 Programming concept

What you will learn in this chapter

- Programming Concepts
- Programming language
- Algorithm
- Scratch

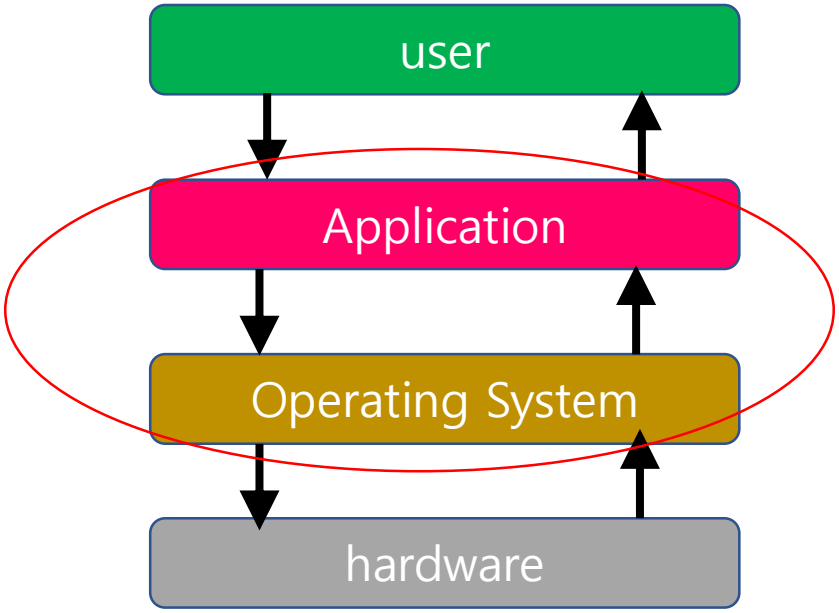
What is a program ?

- A computer is a general-purpose machine . -> You can do m any things using a computer .
- What makes computers general-purpose is the concept of pr ograms .



What if there is no program on the co mputer ?

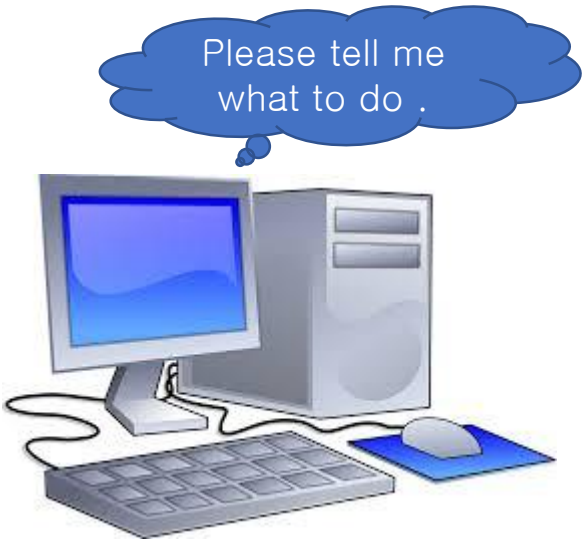
- Without computer hardware and no programs, a computer is just a u seless machine that generates some heat and noise .
- “ Windows” and additionally installing various applications .

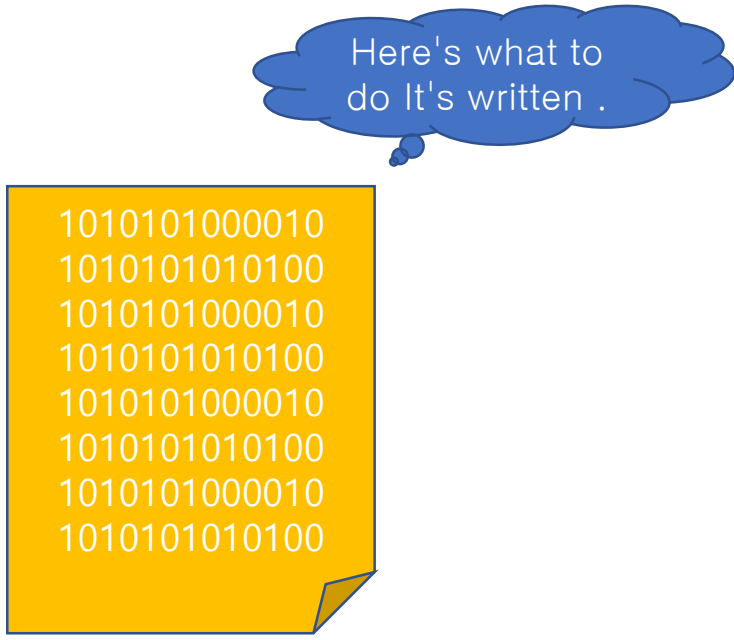


Should I install the program ?

Q) Why don't computers operate right away like home appliances without installing programs, but instead require users to install programs, which is inconvenient ?

A) To make computers into general-purpose machines
. Computers can perform a variety of tasks simply by changing the program .





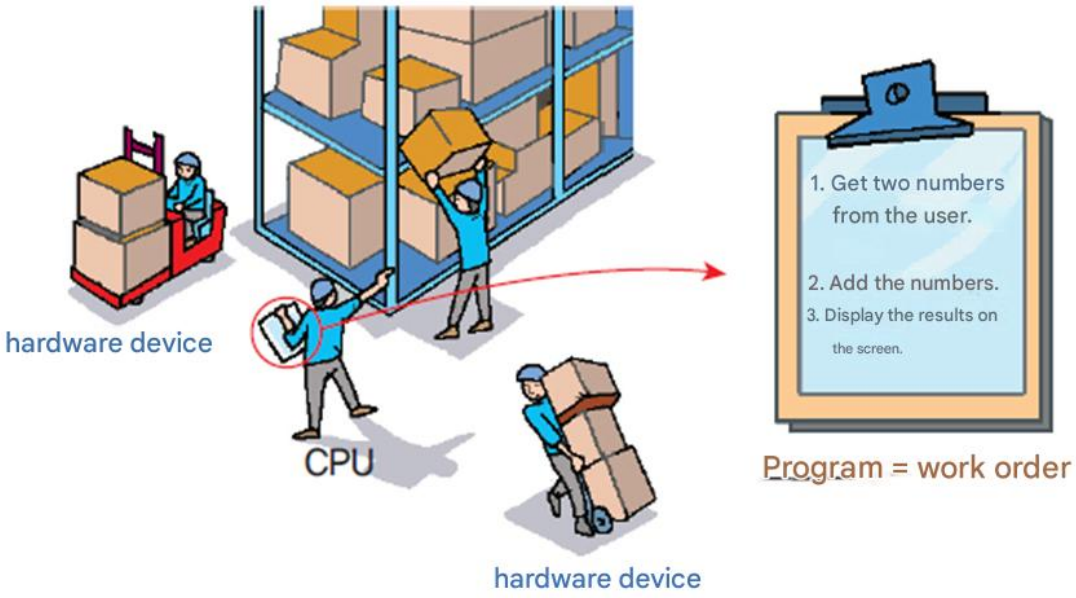
Difference between calculator and com puter

It only performs its assigned functions . Its functions cannot be changed .

You can easily change the function being performed .

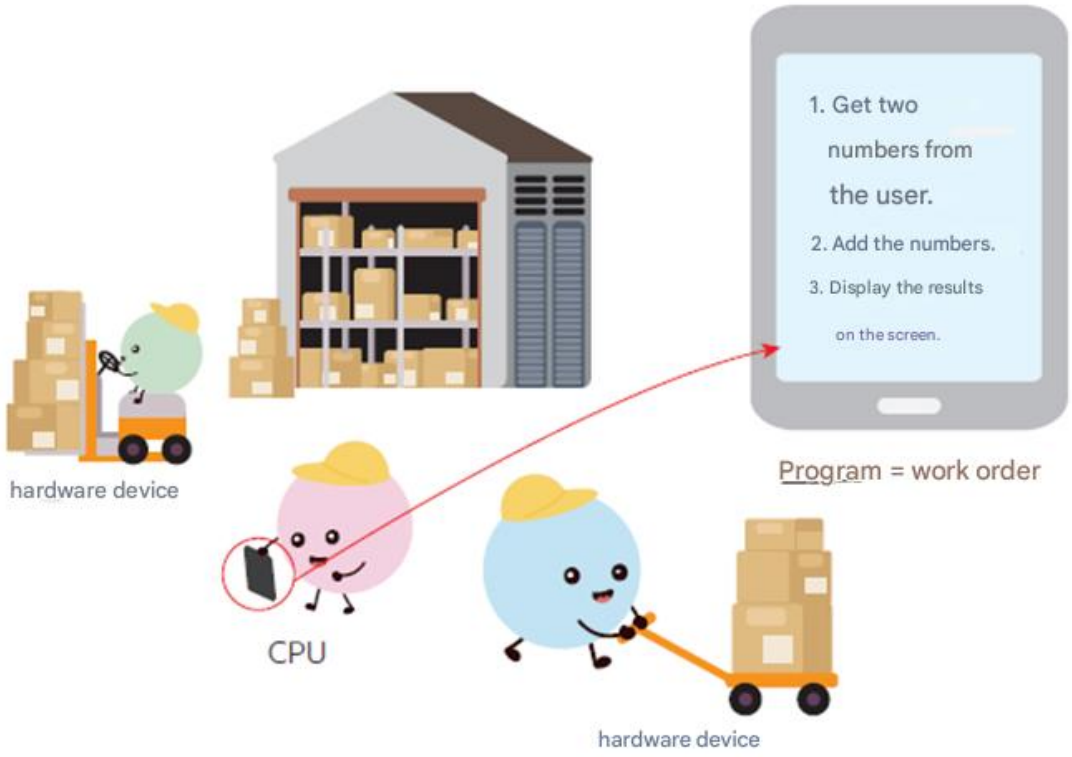
Definition of computer

- A computer is not simply a machine that computes .
- A computer in the modern sense can be said to be a machi ne that processes data according to a program



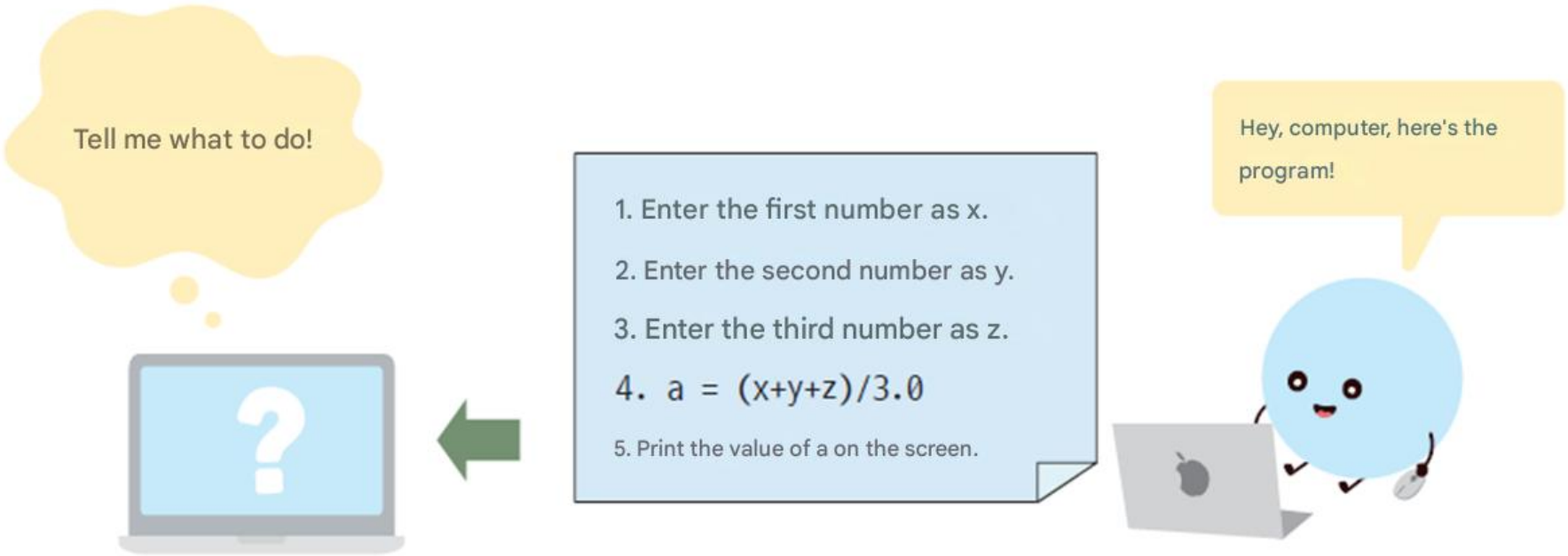
What's inside the program ?

- A program can be thought of as a work instruction for a specific task .
- To perform a task, you need to list instructions . A program contains instructions .



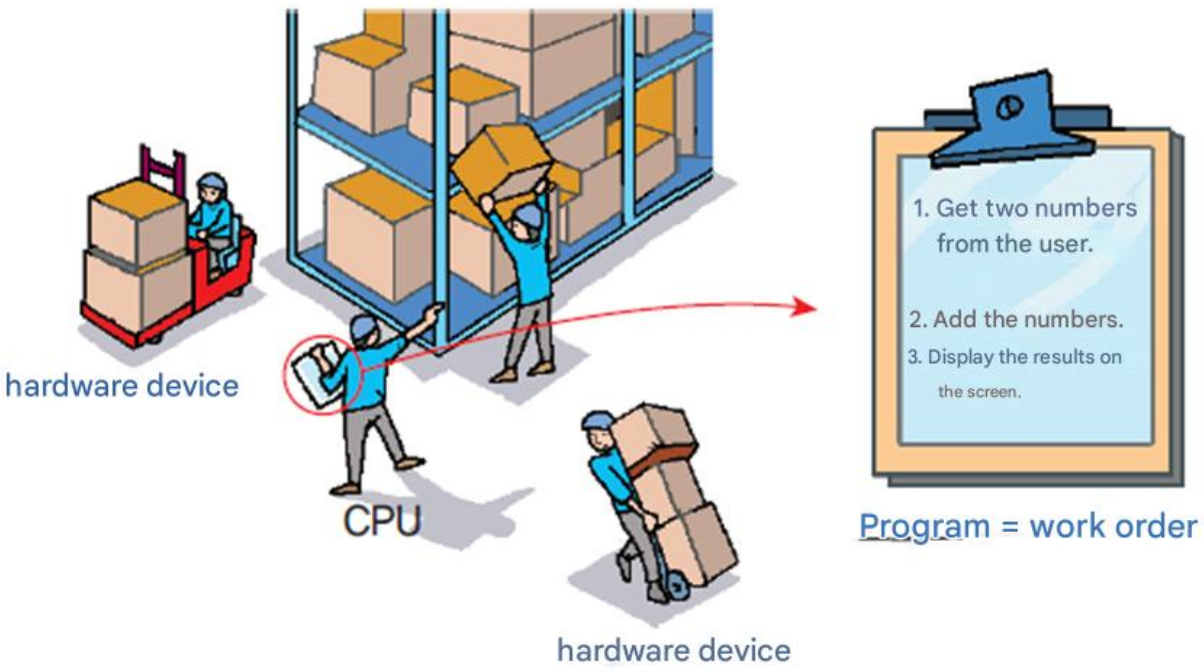
Example of a program

- three numbers and calculates their average . The pro gram can be made up of the following instructions . These instructions are called commands .



Program == Work Instructions

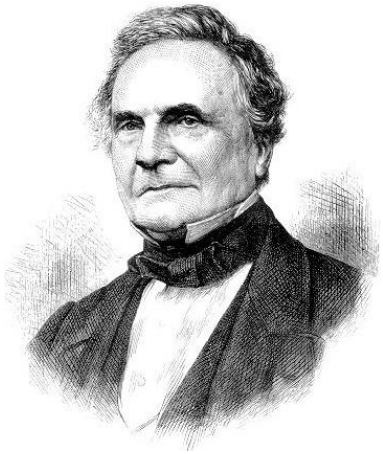
- Program : A document that tells the computer what to do.



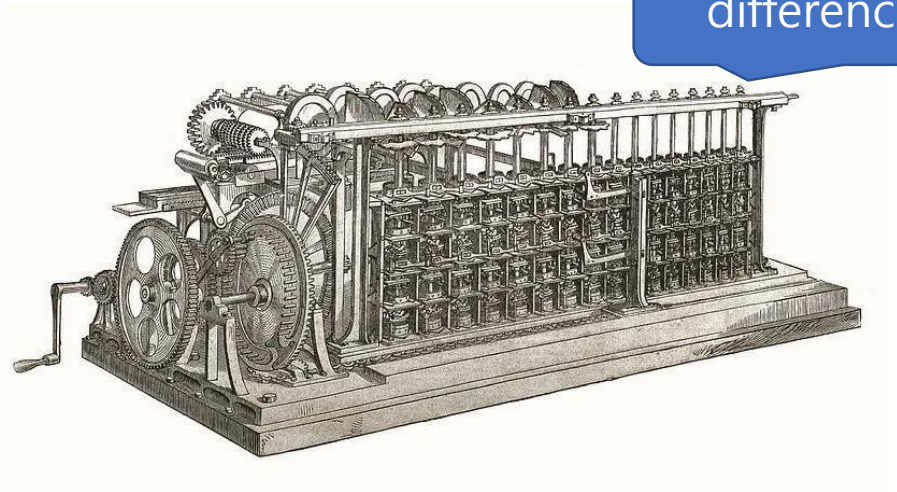
History of the program

It never actually got made !

- The first programmable machine : the Analytical Engine
- Created by : Charles Babbage
- Thousands of gears , wheels , axles , levers, etc. are powered by steam.

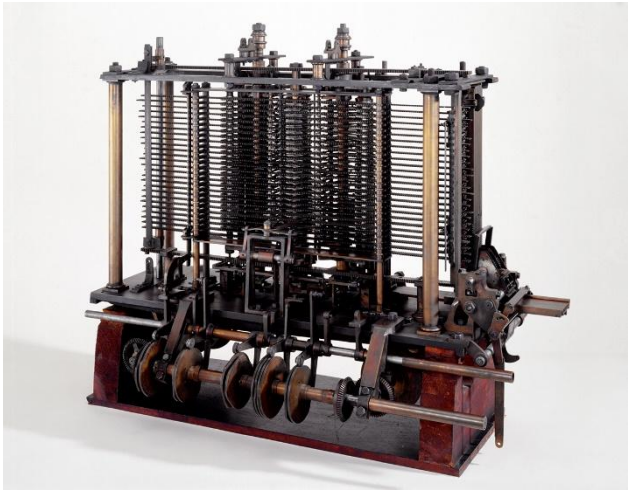


difference organ



Babbage's Analytical Engine

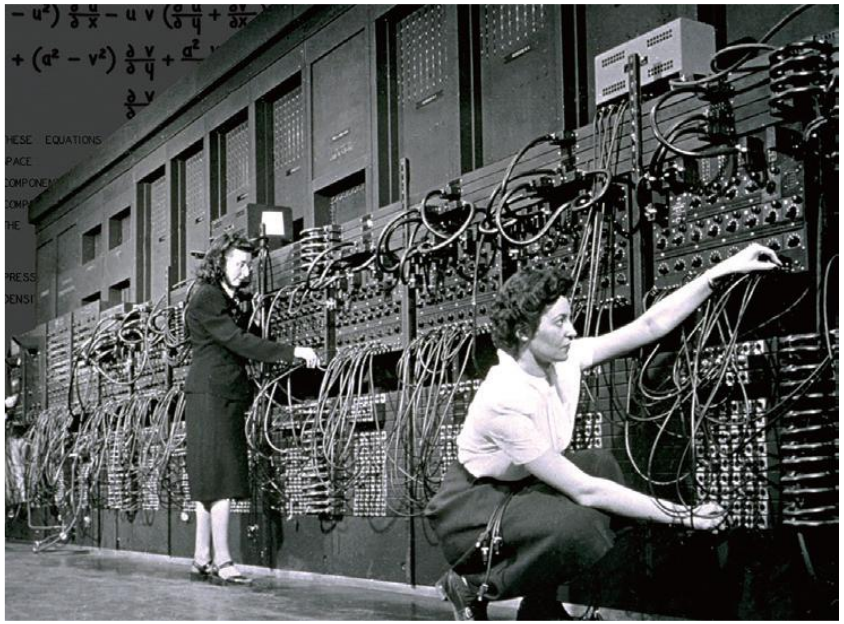
- Babbage's Analytical Engine was designed to be steam-powered and comprised of thousands of gears , wheels , axles , and levers .

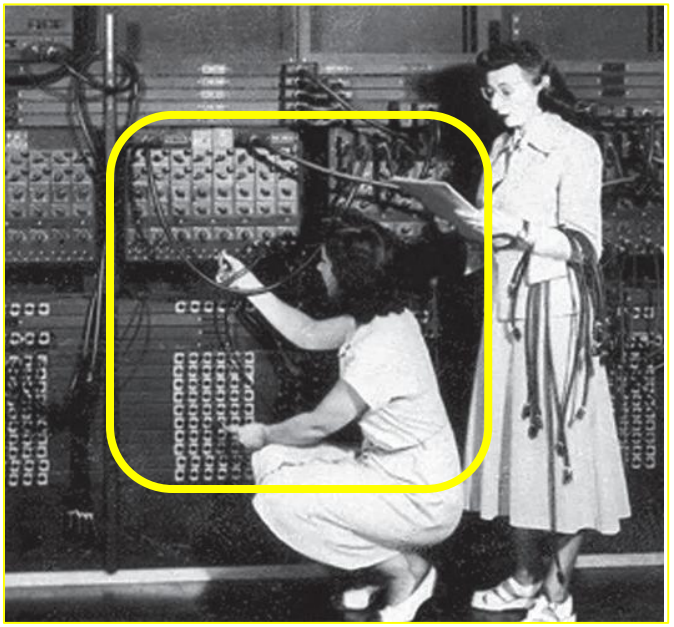


* Central processing unit (responsible for calculations , called mill)
* Memory (where numbers are temporarily stored in intermediate stages , called store)
* Output device (dial that indicates the output number)
* Input device (punch card)

Programming early computers

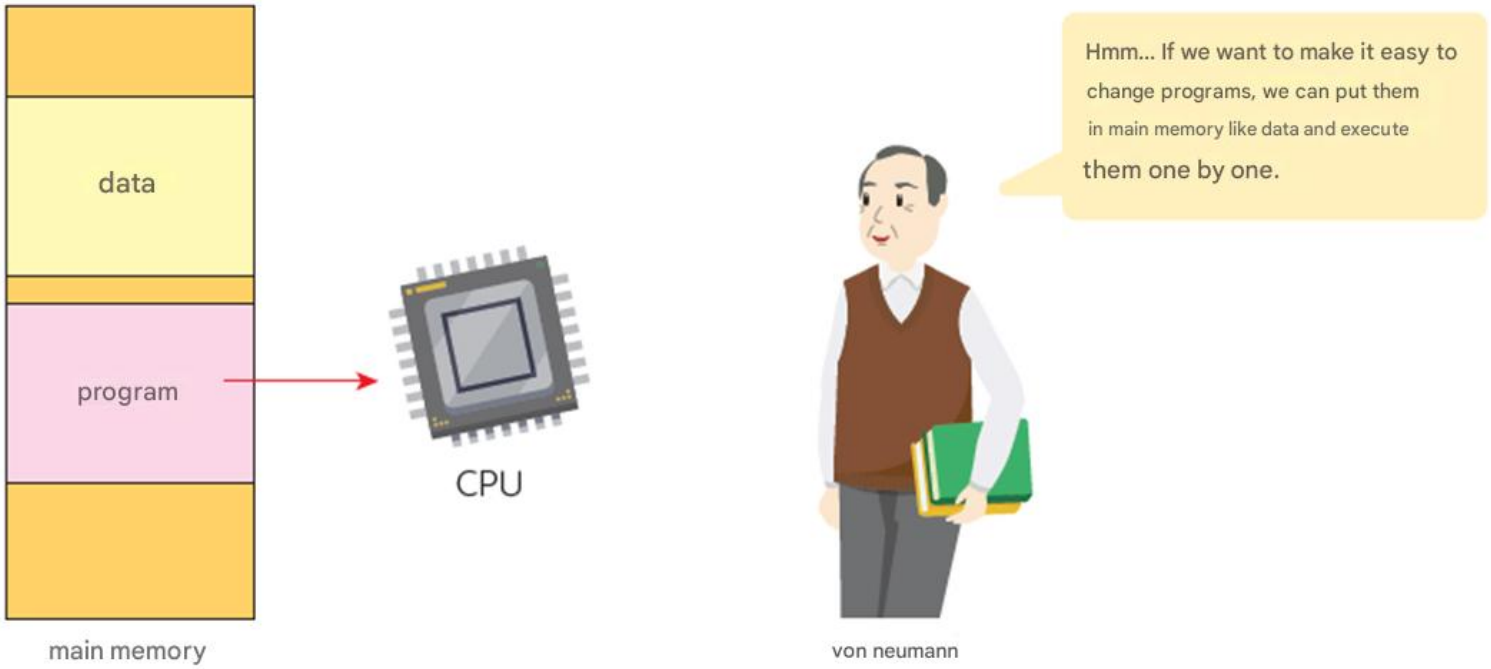
- the early computer, ENIAC , were stored in switches, and eac h time the program was changed, all the switches had to be reconnected from scratch .





Von-neumann architecture

- Programs are stored in main memory -> can be easily changed
- Sequentially retrieves and executes commands from a program s tored in main memory .



The first programmer

- The first person to create the program was Ada Lovelace .
- Ada Great writer Byron's biological daughter
- Babbage's Analytical Engine and developed a program for it .
- Invented core computer programming fundamentals such as su broutines , loops , and jumps .

Enchantress of Numbers Number)!

Ada's program

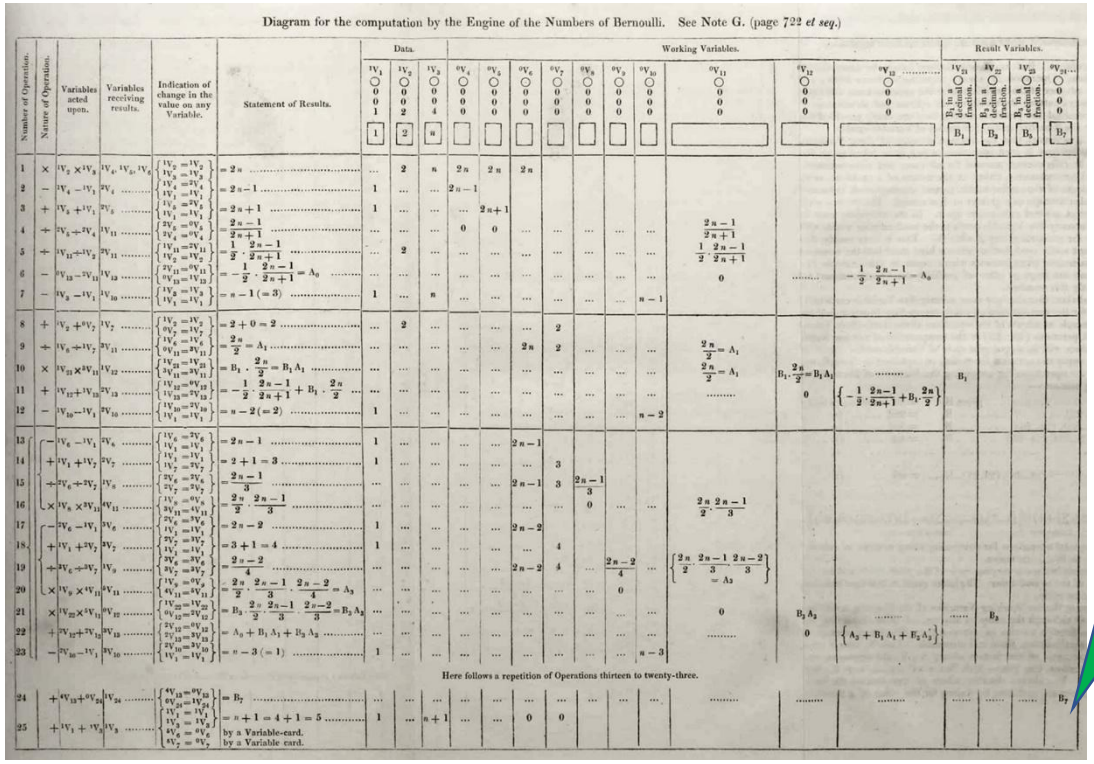


Diagram for the computation by the Engine of the Numbers of Bernoulli. See Note G. (page 722 *et seq.*)

Number of Operation.	Nature of Operation.	Variables acted upon.	Variables receiving results.	Indication of change in the value on any Variable.	Statement of Results.	Data.			Working Variables.									Result Variables.			
						$^1V_1$	$^1V_2$	$^1V_3$	$^0V_4$	$^0V_5$	$^0V_6$	$^0V_7$	$^0V_8$	$^0V_9$	$^0V_{10}$	$^0V_{11}$	$^0V_{12}$	$^0V_{13}$	$^1V_{21}$ B_1 in a decimal fraction.	$^1V_{22}$ B_3 in a decimal fraction.	$^1V_{23}$ B_5 in a decimal fraction.	$^0V_{24}$ .....
						1	2	n											B_1	B_3	B_5	B_7
1	×	$^1V_2 \times ^1V_3$	$^1V_4, ^1V_5, ^1V_6$	$\left\{\begin{matrix}^1V_2 = {}^1V_2 \\ ^1V_3 = {}^1V_3\end{matrix}\right\}$	$= 2n$	...	2	n	$2n$	$2n$	$2n$										
2	−	$^1V_4 - ^1V_1$	$^2V_4$	$\left\{\begin{matrix}^1V_4 = {}^2V_4 \\ ^1V_1 = {}^1V_1\end{matrix}\right\}$	$= 2n-1$	1	...	...	$2n-1$													
3	+	$^2V_4 + ^1V_1$	$^2V_5$	$\left\{\begin{matrix}^2V_5 = {}^2V_5 \\ ^1V_1 = {}^1V_1\end{matrix}\right\}$	$= 2n+1$	1	...	...	...	$2n+1$												
4	÷	$^2V_5 \div ^2V_4$	$^1V_{11}$	$\left\{\begin{matrix}^2V_5 = {}^0V_5 \\ ^2V_4 = {}^1V_4\end{matrix}\right\}$	$= \frac{2n-1}{2n+1}$	...	...	...	0	0	...	...	...	...	...	$\frac{2n-1}{2n+1}$ $\frac{1}{2}\cdot\frac{2n-1}{2n+1}$						
5	+	$^1V_{11} + ^1V_2$	$^2V_{11}$	$\left\{\begin{matrix}^1V_{11} = {}^2V_{11} \\ ^1V_2 = {}^1V_2\end{matrix}\right\}$	$= \frac{1}{2}\cdot\frac{2n-1}{2n+1}$	...	2	...	...	...	...	...	...	...	...							
6	−	$^2V_{13} - ^2V_{11}$	$^1V_{13}$	$\left\{\begin{matrix}^2V_{11} = {}^0V_{11} \\ ^2V_{13} = {}^1V_{13}\end{matrix}\right\}$	$= -\frac{1}{2}\cdot\frac{2n-1}{2n+1} = A_0$	...	...	...	...	...	...	...	...	...	...	0	.....	$-\frac{1}{2}\cdot\frac{2n-1}{2n+1} = A_0$				
7	−	$^1V_3 - ^1V_1$	$^1V_{10}$	$\left\{\begin{matrix}^1V_3 = {}^1V_3 \\ ^1V_1 = {}^1V_1\end{matrix}\right\}$	$= n-1\ (=3)$	1	...	n	...	...	...	...	...	...	$n-1$							
8	+	$^1V_2 + ^0V_7$	$^1V_7$	$\left\{\begin{matrix}^1V_2 = {}^1V_2 \\ ^0V_7 = {}^1V_7\end{matrix}\right\}$	$= 2+0 = 2$	...	2	...	...	...	...	2										
9	÷	$^1V_6 \div ^1V_7$	$^2V_{11}$	$\left\{\begin{matrix}^1V_6 = {}^1V_6 \\ ^0V_{11} = {}^2V_{11}\end{matrix}\right\}$	$= \frac{2n}{2} = A_1$	...	...	...	...	...	$2n$	2	...	...	...	$\frac{2n}{2} = A_1$						
10	×	$^1V_{23} \times ^2V_{11}$	$^1V_{12}$	$\left\{\begin{matrix}^1V_{23} = {}^1V_{23} \\ ^2V_{11} = {}^2V_{11}\end{matrix}\right\}$	$= B_1 \cdot \frac{2n}{2} = B_1 A_1$	...	...	...	...	...	...	...	...	...	...	$\frac{2n}{2} = A_1$	$B_1 \cdot \frac{2n}{2} = B_1 A_1$	.....	B_1			
11	+	$^1V_{12} + ^1V_{13}$	$^2V_{13}$	$\left\{\begin{matrix}^1V_{12} = {}^0V_{12} \\ ^1V_{13} = {}^2V_{13}\end{matrix}\right\}$	$= -\frac{1}{2}\cdot\frac{2n-1}{2n+1} + B_1 \cdot \frac{2n}{2}$ ...	...	...	...	...	...	...	...	...	...	...	.....	0	$\left\{-\frac{1}{2}\cdot\frac{2n-1}{2n+1} + B_1 \cdot \frac{2n}{2}\right\}$				
12	−	$^1V_{10} - ^1V_1$	$^2V_{10}$	$\left\{\begin{matrix}^1V_{10} = {}^2V_{10} \\ ^1V_1 = {}^1V_1\end{matrix}\right\}$	$= n-2\ (=2)$	1	...	...	...	...	...	...	...	...	$n-2$							
13	−	$^2V_6 - ^1V_1$	$^2V_6$	$\left\{\begin{matrix}^2V_6 = {}^2V_6 \\ ^1V_1 = {}^1V_1\end{matrix}\right\}$	$= 2n-1$	1	...	...	...	...	$2n-1$											
14	+	$^2V_7 + ^1V_2$	$^2V_7$	$\left\{\begin{matrix}^2V_7 = {}^2V_7 \\ ^1V_2 = {}^1V_2\end{matrix}\right\}$	$= 2+1 = 3$	1	...	...	...	...	...	3										
15	÷	$^2V_6 \div ^2V_7$	$^2V_8$	$\left\{\begin{matrix}^2V_6 = {}^2V_6 \\ ^2V_7 = {}^2V_7\end{matrix}\right\}$	$= \frac{2n-1}{3}$	...	...	...	...	...	$2n-1$	3	$\frac{2n-1}{3}$	...	...							
16	×	$^2V_8 \times ^2V_{11}$	$^4V_{11}$	$\left\{\begin{matrix}^2V_8 = {}^0V_8 \\ ^2V_{11} = {}^4V_{11}\end{matrix}\right\}$	$= \frac{2n}{2}\cdot\frac{2n-1}{3}$	...	...	...	...	...	...	...	0	...	...	$\frac{2n}{2}\cdot\frac{2n-1}{3}$						
17	−	$^2V_6 - ^1V_1$	$^2V_6$	$\left\{\begin{matrix}^2V_6 = {}^2V_6 \\ ^1V_1 = {}^1V_1\end{matrix}\right\}$	$= 2n-2$	1	...	...	...	...	$2n-2$											
18	+	$^1V_1 + ^2V_7$	$^2V_7$	$\left\{\begin{matrix}^2V_7 = {}^2V_7 \\ ^1V_1 = {}^1V_1\end{matrix}\right\}$	$= 3+1 = 4$	1	...	...	...	...	...	4										
19	÷	$^2V_6 \div ^2V_7$	$^2V_9$	$\left\{\begin{matrix}^2V_6 = {}^2V_6 \\ ^2V_7 = {}^2V_7\end{matrix}\right\}$	$= \frac{2n-2}{4}$	...	...	...	...	...	$2n-2$	4	...	$\frac{2n-2}{4}$	...	$\left\{\frac{2n}{2}\cdot\frac{2n-1}{3}\cdot\frac{2n-2}{4}\right\} = A_3$						
20	×	$^2V_9 \times ^4V_{11}$	$^2V_{11}$	$\left\{\begin{matrix}^2V_9 = {}^0V_9 \\ ^4V_{11} = {}^2V_{11}\end{matrix}\right\}$	$= \frac{2n}{2}\cdot\frac{2n-1}{3}\cdot\frac{2n-2}{4} = A_3$	...	...	...	...	...	...	...	...	0	...							
21	×	$^1V_{22} \times ^2V_{11}$	$^0V_{12}$	$\left\{\begin{matrix}^1V_{22} = {}^0V_{22} \\ ^0V_{12} = {}^2V_{12}\end{matrix}\right\}$	$= B_3 \cdot \frac{2n}{2}\cdot\frac{2n-1}{3}\cdot\frac{2n-2}{4} = B_3 A_3$	...	...	...	...	...	...	...	...	...	...	0	$B_3 A_3$	.....	.....	B_3		
22	+	$^2V_{12} + ^2V_{13}$	$^2V_{13}$	$\left\{\begin{matrix}^2V_{12} = {}^2V_{12} \\ ^2V_{13} = {}^2V_{13}\end{matrix}\right\}$	$= A_0 + B_1 A_1 + B_3 A_3$	...	...	...	...	...	...	...	...	...	...	.....	0	$\{A_3 + B_1 A_1 + B_3 A_3\}$				
23	−	$^2V_{10} - ^1V_1$	$^2V_{10}$	$\left\{\begin{matrix}^2V_{10} = {}^2V_{10} \\ ^1V_1 = {}^1V_1\end{matrix}\right\}$	$= n-3\ (=1)$	1	...	...	...	...	...	...	...	...	$n-3$							

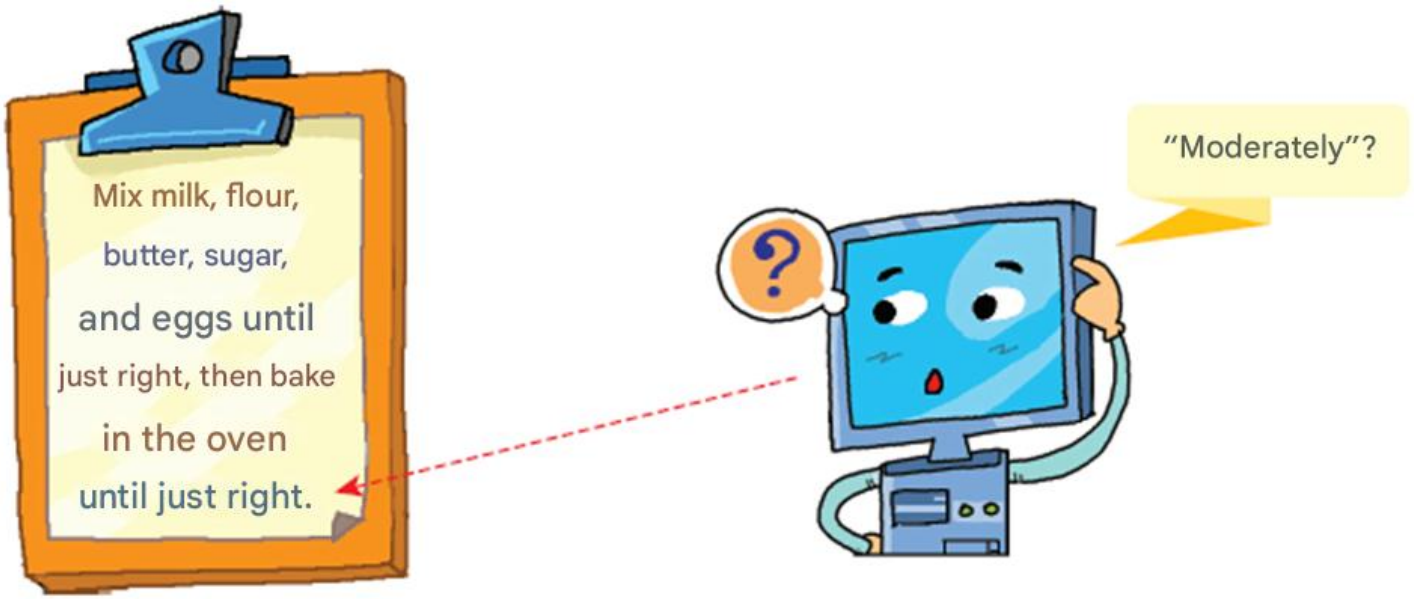
Here follows a repetition of Operations thirteen to twenty-three.

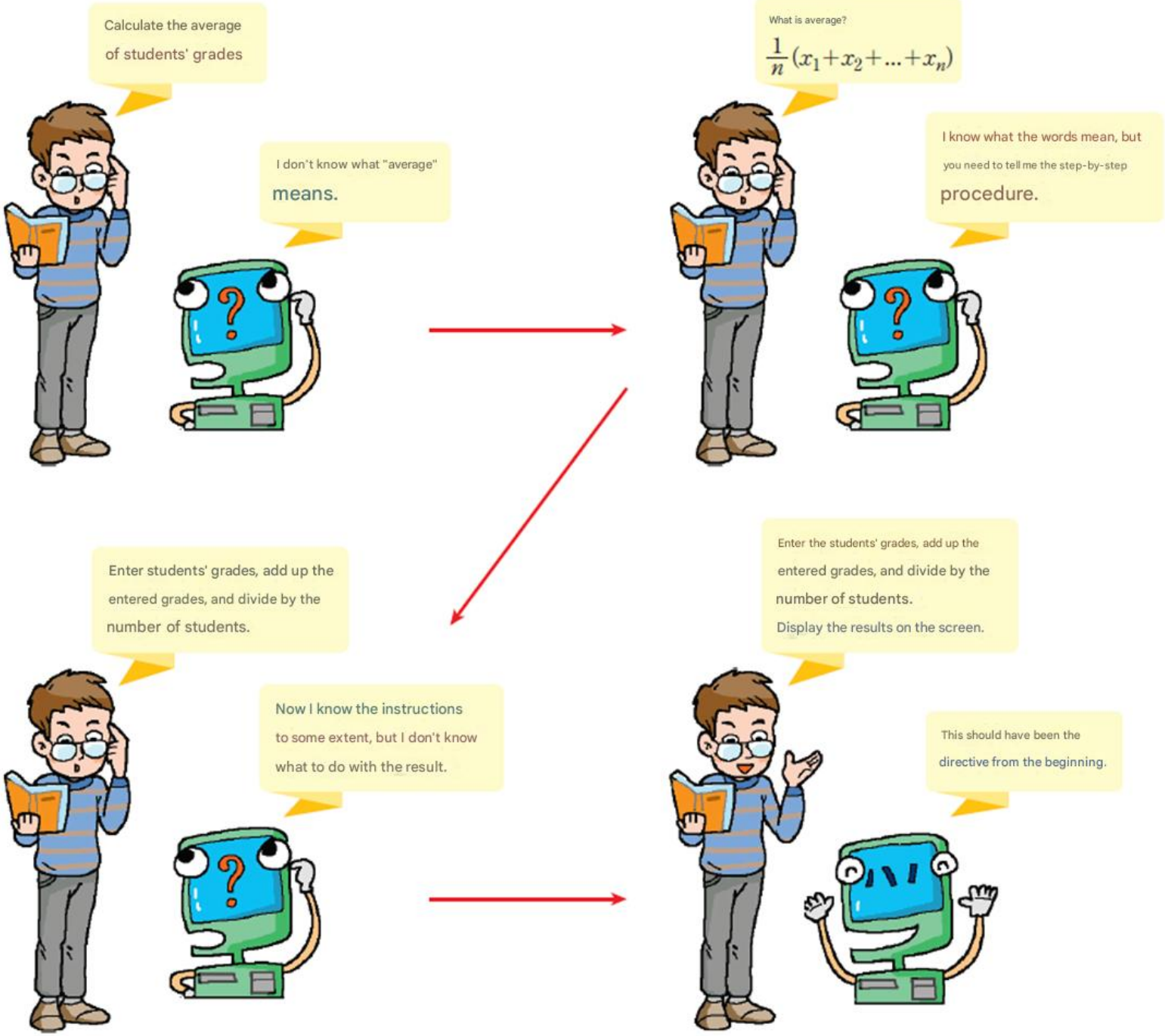
24	+	$^4V_{13} + ^0V_{24}$	$^4V_{24}$	$\left\{\begin{matrix}^4V_{13} = {}^0V_{13} \\ ^0V_{24} = {}^4V_{24}\end{matrix}\right\}$	$= B_7$	...	...	...	...	...	...	...	...	...	...	.....	.....	.....	.....	.....	.....	B_7
25	+	$^1V_1 + ^1V_2$	$^1V_3$	$\left\{\begin{matrix}^1V_1 = {}^1V_1 \\ ^1V_2 = {}^1V_2 \\ ^1V_3 = {}^1V_3 \\ ^0V_6 = {}^0V_6 \\ ^0V_7 = {}^0V_7\end{matrix}\right\}$	$= n+1 = 4+1 = 5$ ..... by a Variable-card. by a Variable card.	1	...	$n+1$	...	...	0	0											

1842 , Ada worked with Italian engineer Luigi Menabrea on the Analytical Engine. In the process of translating the article, I added a note in the margin . Ada's note is three times longer than the article itself , and explains in detail how to compute Bernoulli numbers using an analysis engine . This note is considered by many to be the first computer program in the early history of computing , that is, an algorithm designed for a machine to perform .

How detailed should the work instructions be ?

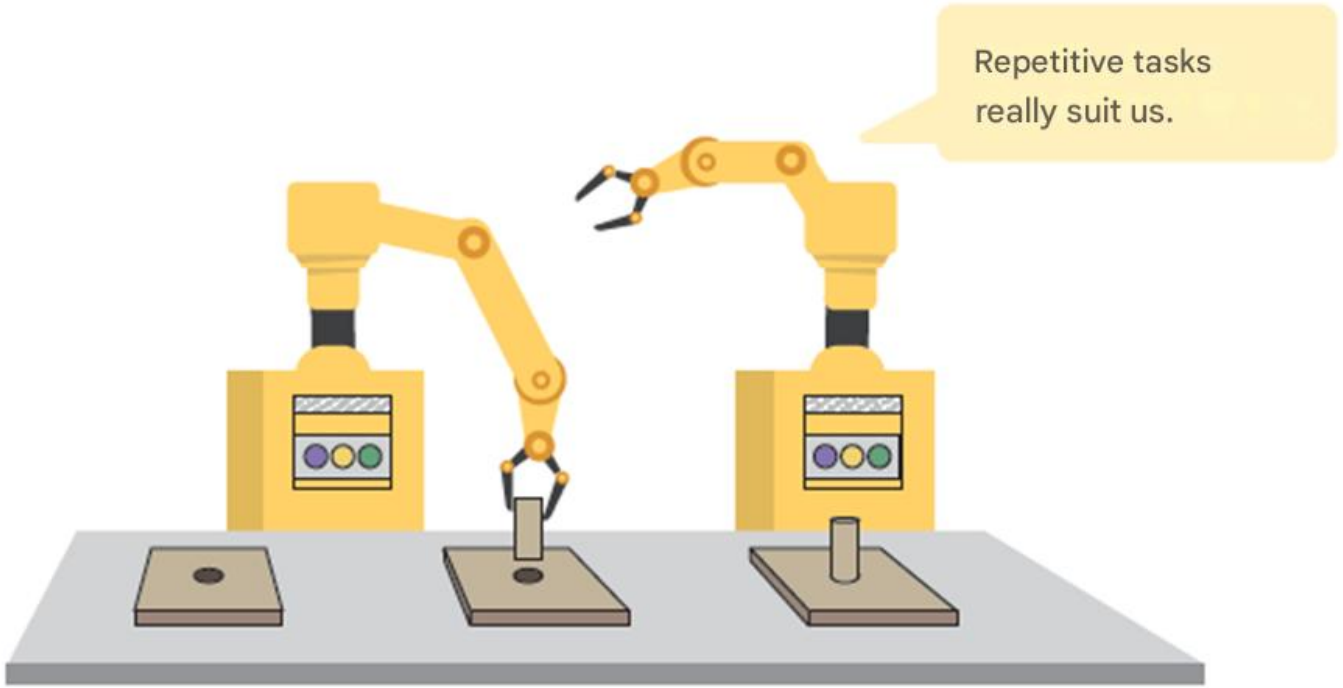
- Because they lack common sense or intelligence, they must be gi ven very detailed and specific instructions on what to do .




Calculate the average of students' grades
I don't know what "average" means.
What is average?
$\frac{1}{n}(x_1+x_2+...+x_n)$
I know what the words mean, but you need to tell me the step-by-step procedure.
Enter students' grades, add up the entered grades, and divide by the number of students.
Now I know the instructions to some extent, but I don't know what to do with the result.
Enter the students' grades, add up the entered grades, and divide by the number of students.
Display the results on the screen.
This should have been the directive from the beginning.

Advantages of Computers

- He does his work very quickly and accurately, and he never c omplains no matter how many times he is asked to do it .

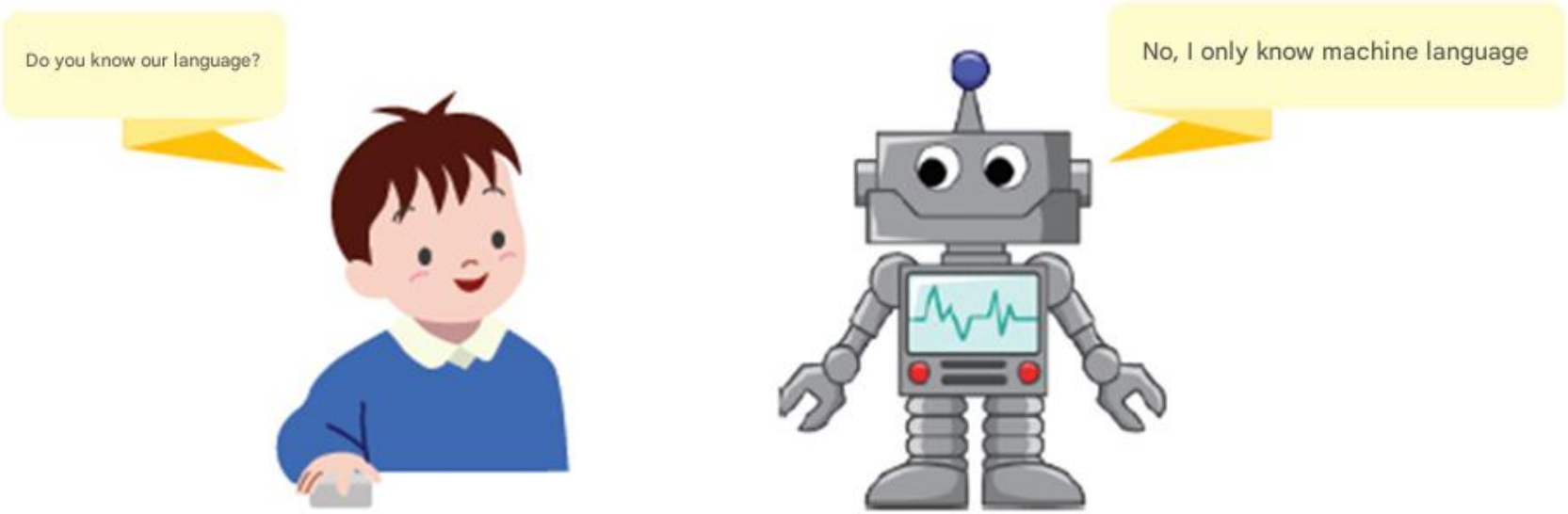


What you will learn in this chapter

- Programming Concepts
- Programming language
- Algorithm
- Scratch

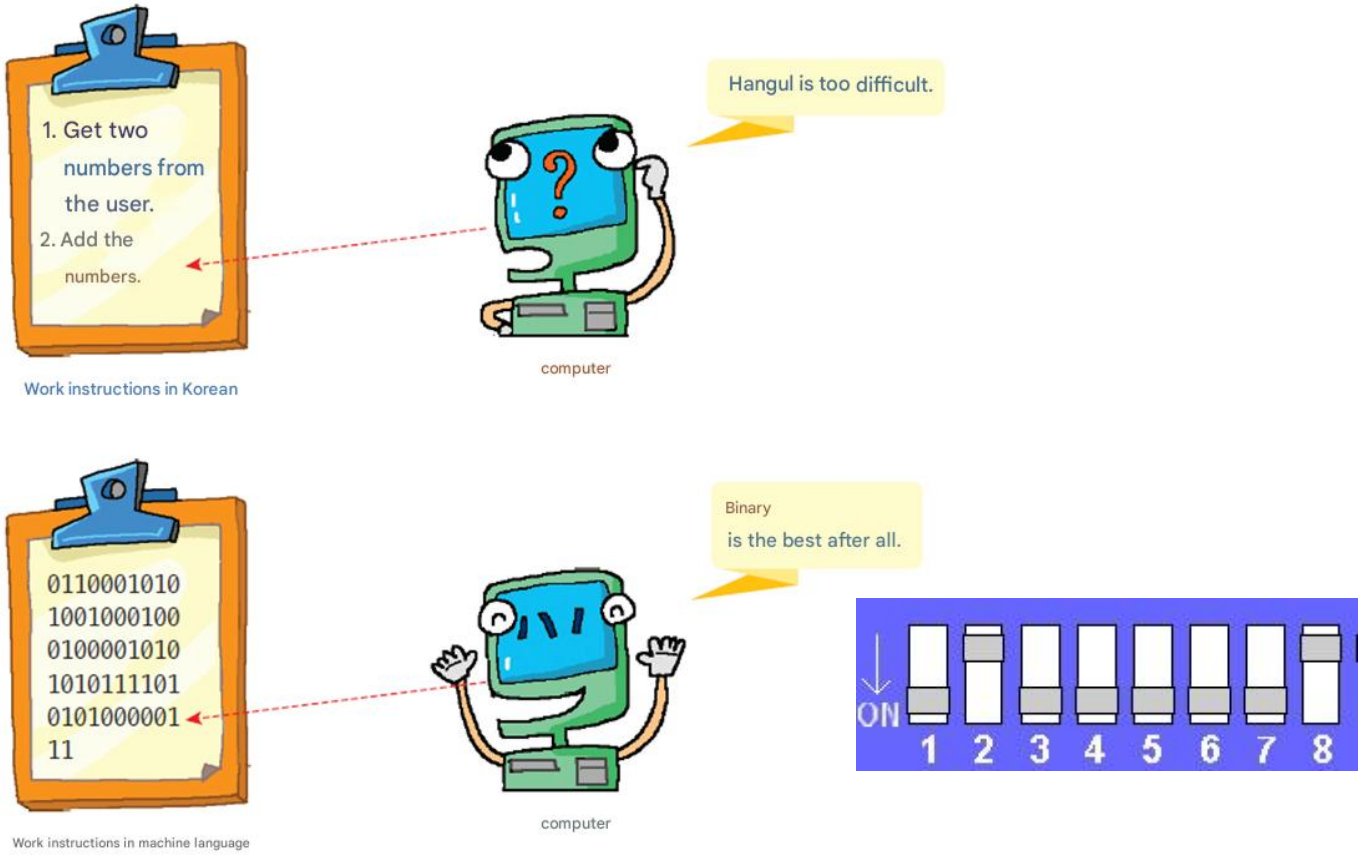
Language that computers understand

- Computers understand only one language : machine language , which consists of 0s and 1s , such as “ 001101110001010...”
- Computers express everything as 0 and 1 , and work by turnin g internal switch circuits ON/OFF based on 0 and 1 .



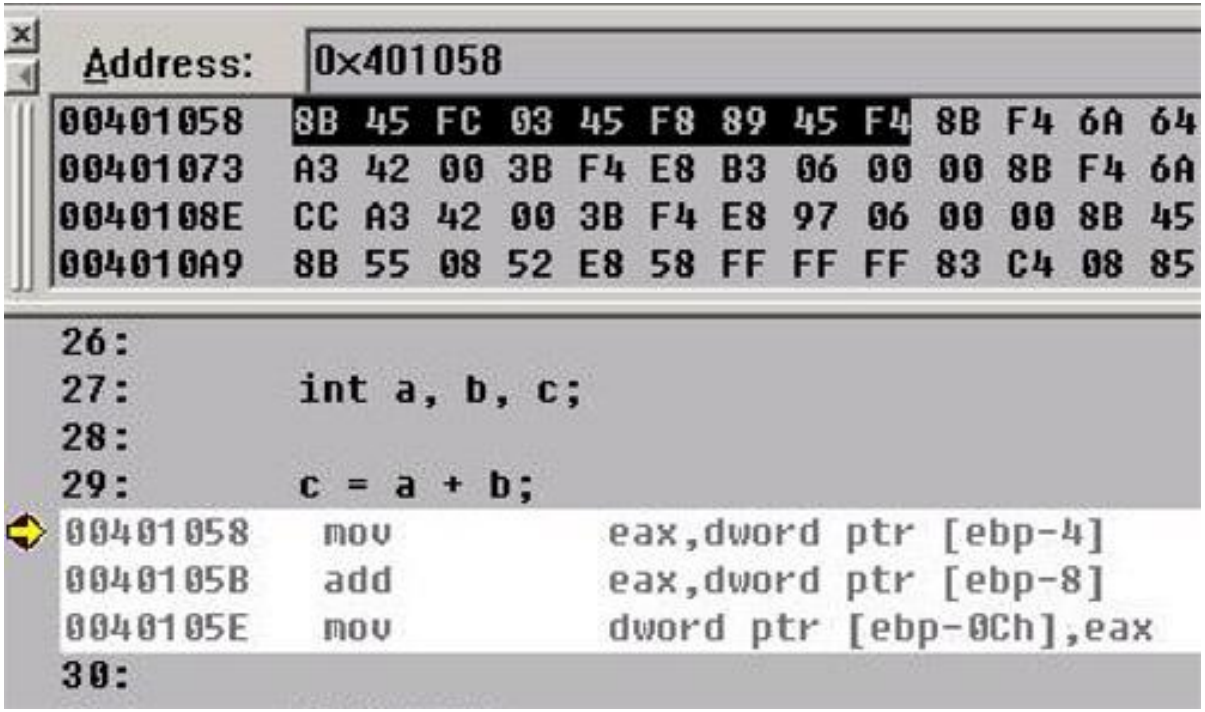
machine language

- It is a binary number such as “ 0011011100010 10 …” , which is composed of 0 and 1 .



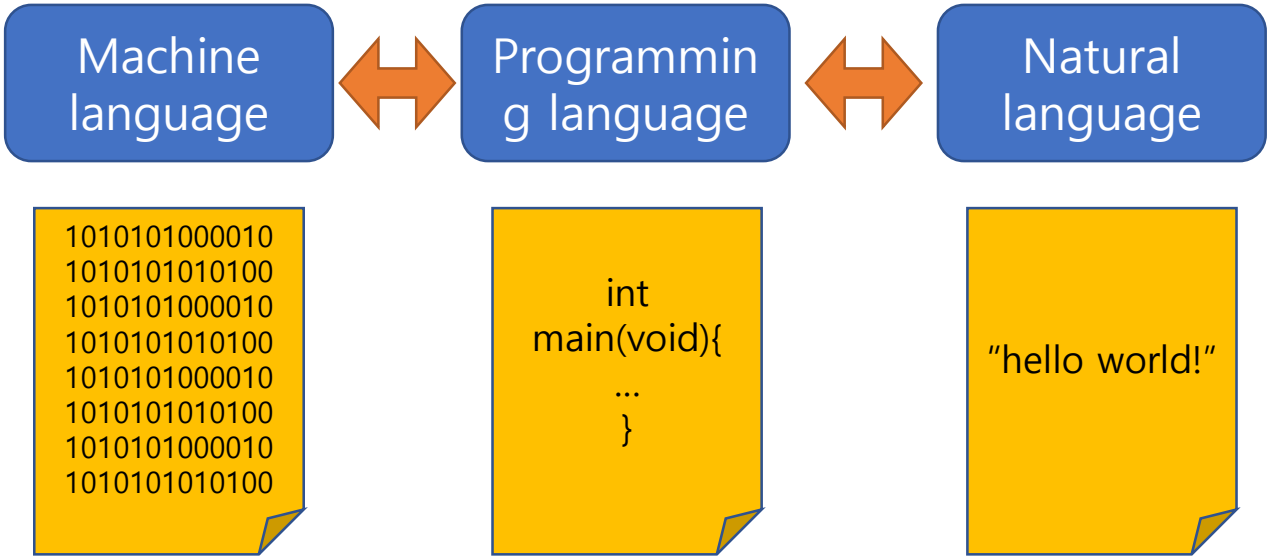
Machine language

- Example of machine language



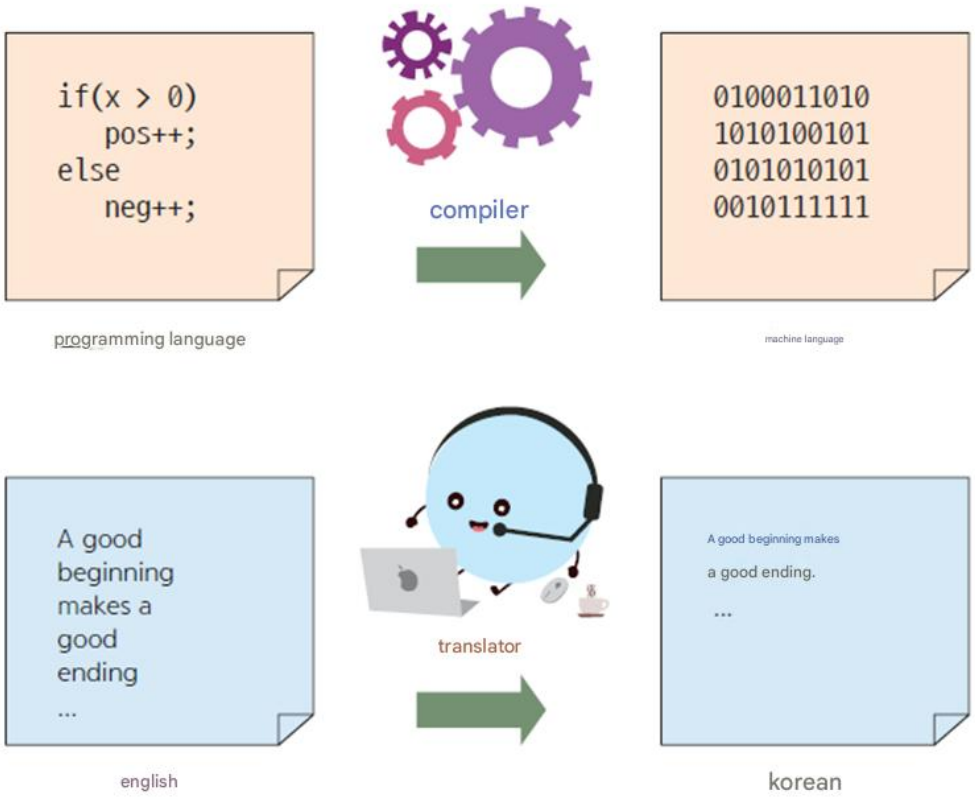
Programming language

- Although machine language can be used, it is very inconvenient because programs must be written in binary .
- Programming languages are somewhere between natural language and machine language.
- A compiler translates a programming language into machine language.



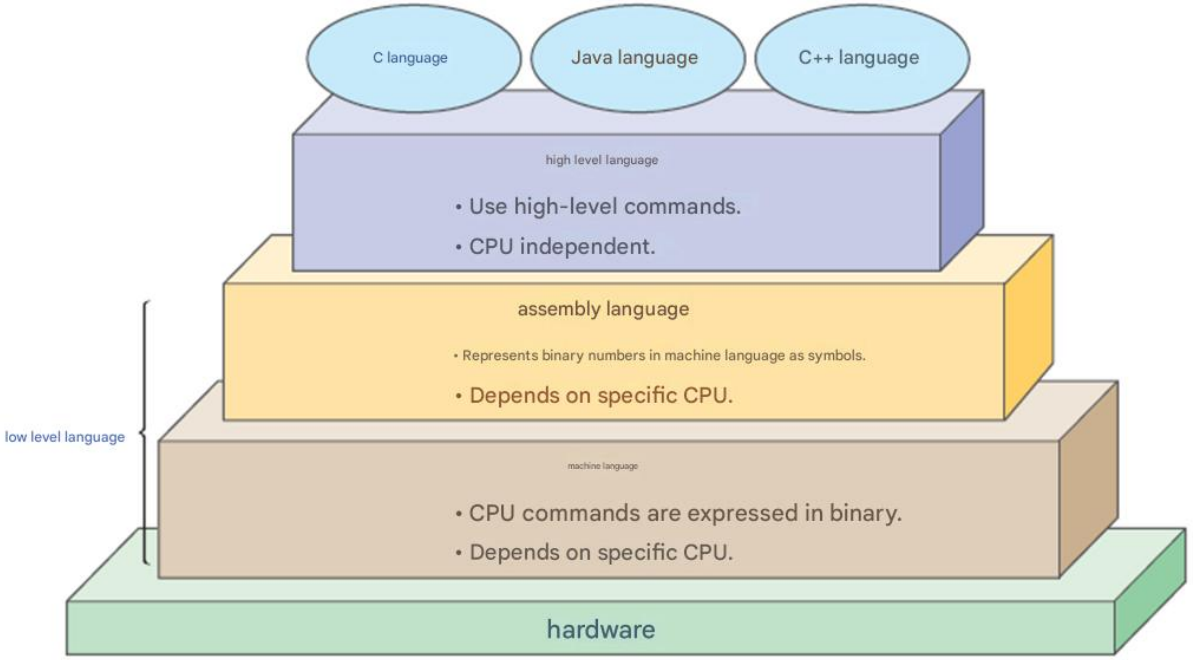
compiler

- A compiler can be considered an interpreter between humans and computers .



Classification of programming languages

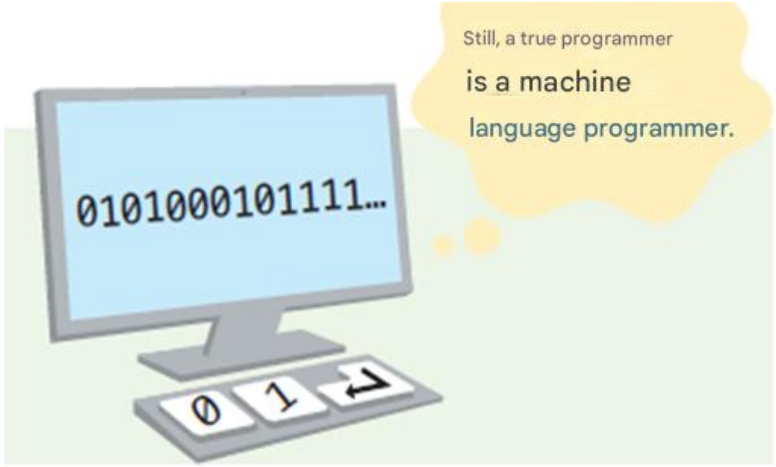
- machine language
- assembly language
- high-level



Machine language

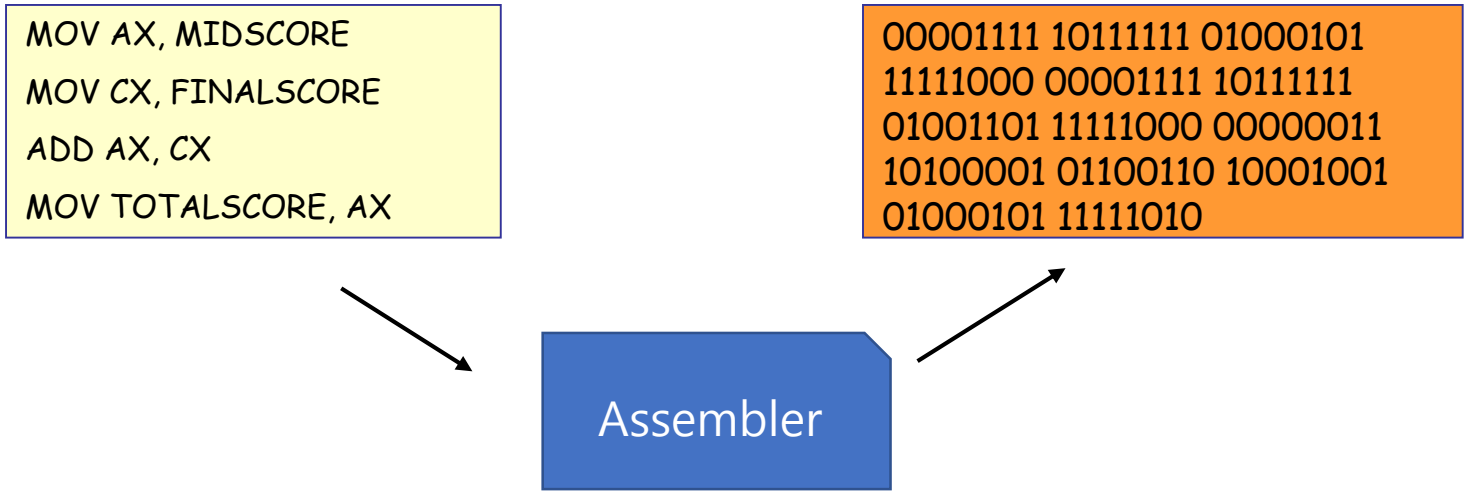
- A binary representation of the instructions of a specific comp uter .
- Consists of 0 and 1
- Hardware dependent

```
00001111 10111111 01000101
11111000 00001111 10111111
01001101 11111000 00000011
10100001 01100110 10001001
01000101 11111010
```



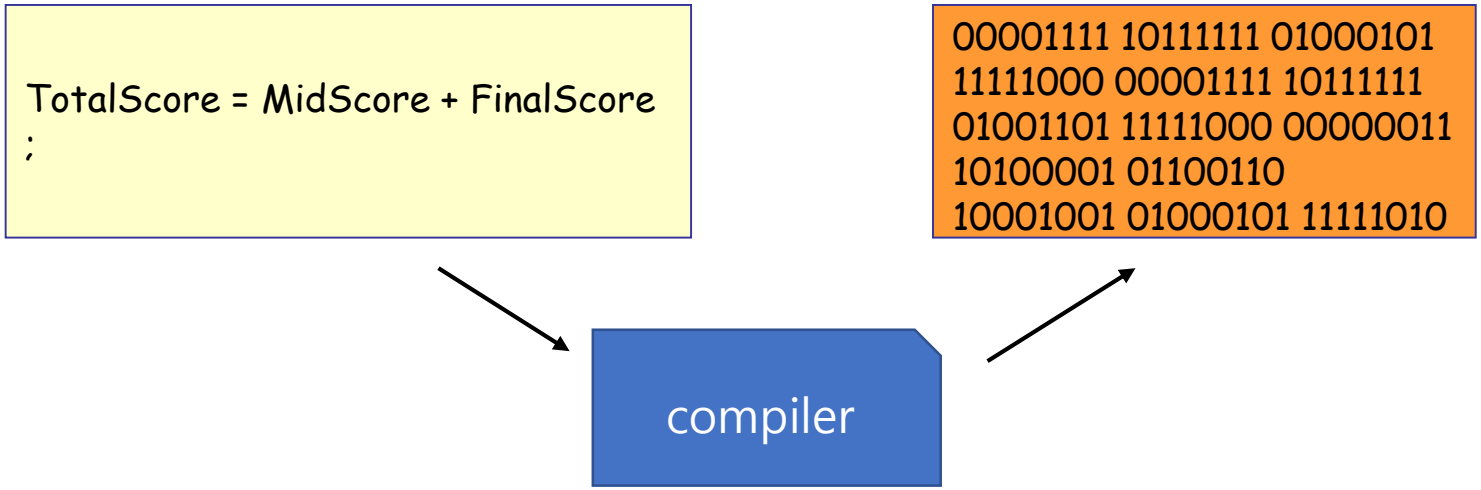
Assembler

- CPU commands are expressed in symbols that are abbreviations of English letters rather than binary numbers.
- It is possible to write programs at a higher level than machine language between symbols and CPU commands
- Assembler : A program that converts symbols into binary.



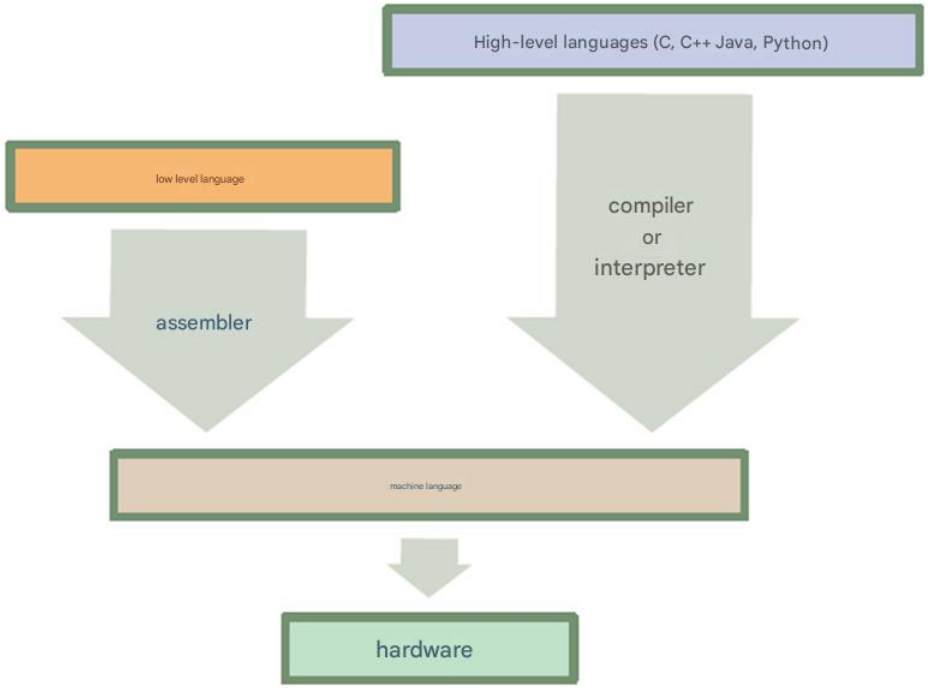
High-level language

- A language that allows you to write programs independently , regardless of the architecture or processor of a specific comp uter.
- C, C++, JAVA, FORTRAN, PASCAL
- Compiler : A program that converts high-level language state ments into machine language.



Low-level and high-level languages

- The reason why they are called high-level languages is becau se their structure is closer to human language than to machi ne language . In contrast, assembly language and the like are classified as low-level languages because they are closer to machine language .



Types of high-level languages

Made by FREE-VECTORS.NET

language	characteristic	example

FORTRAN	Fortran is a language that was first created in the 1950s and is suitable for numerical and scientific computing.	`PROGRAM HELLO`
		`  PRINT '(A)', 'Hello World'`
		`  STOP`
		`END`

COBOL	COBOL is a business processing language created in 1959. The language was designed to be written in a colloquial sentence format.	`IDENTIFICATION DIVISION.`
		`  PROGRAM-ID. HELLO-WORLD.`
		`  PROCEDURE DIVISION.`
		`  DISPLAY 'Hello World'.`
		`  STOP RUN.`

Python	Python is an interpreted language developed by Guido van Rossum in 1991. It is easy for beginners to learn and is widely used in the fields of artificial intelligence and data science.	`print("Hello World")`

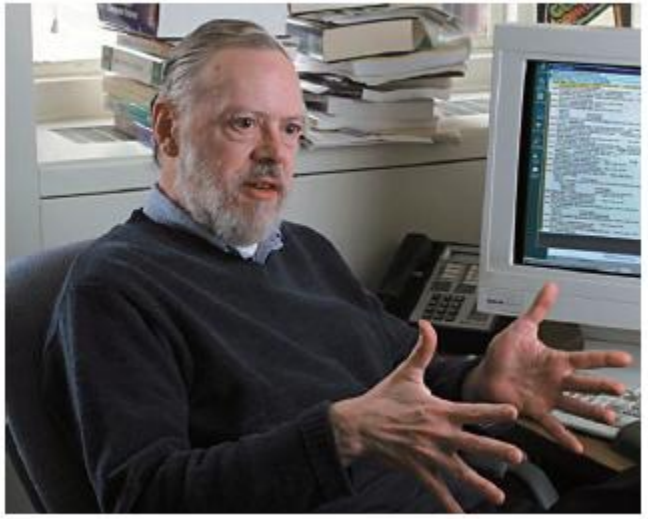
C	The C language was created in the early 1970s by Dennis Ritchie, who was working at AT&T's Bell Labs for the UNIX operating system.	`int main(void)`
		`{`
		`  printf("Hello World\n");`
		`  return 0;`
		`}`

C++	C++ is a language developed by Stroustrup at Bell Labs in 1983, and is a language that adds several object-oriented features, including the concept of classes, to the C language.	`int main()`
		`{`
		`  cout << "Hello World" << endl;`
		`  return 0;`
		`}`

Java	Java is an object-oriented language developed by James Gosling of Sun Microsystems in 1995.	`public class Hello {`
		`  public static void main(String[] args)`
		`  {`
		`  System.out.println("Hello World");`
		`  }`
		`}`

Introduction of C Language

- Developed by Dennis Ritchie of AT&T in the early 1970s.
- Language B -> Language C
- Created for the development of UNIX operating systems.
- Starting from scratch with professional language



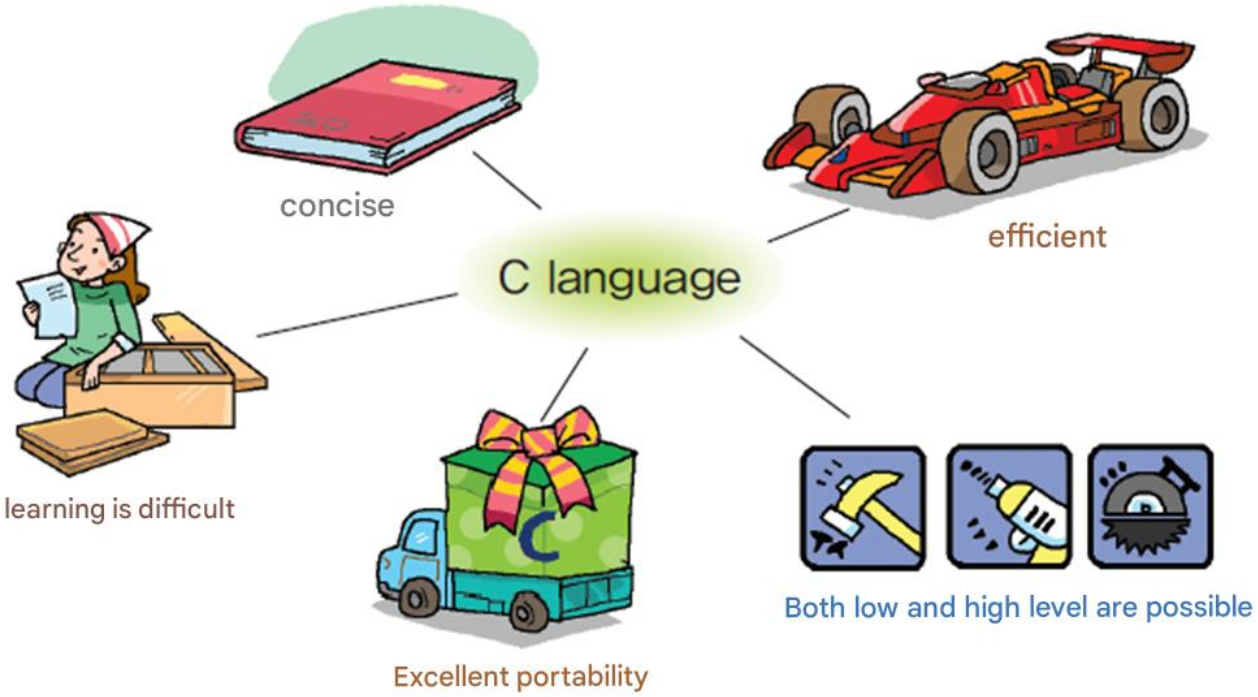
C language

- K & RC
  - 1978 “ C Programming Language ” Book publication
  - Informal specification role
- ANSI C
  - In 1983 , the American National Standards Institute (ANSI) published a s tandard by a committee called X3J11.
- C99
  - Standardized by ISO in 1999
  - Add features used in C++
  - Supported by a growing number of compilers
- C11
  - C language standard published by ISO in December 2011 .
- C17, C18
  - C17 , published as ISO/IEC 9899:2018 in June 2018 , is the current stand ard . It adds no new language features and only fixes technical defects f rom the C11 version .

Features of the C language

- It's concise .
- It is efficient .
- The C language allows for both low-level programming that directly controls hardware , as well as high-level programming.
- C language is highly portable .
- C language will help you understand how computer hardware works .

Features of the C language



Disadvantages of the C language

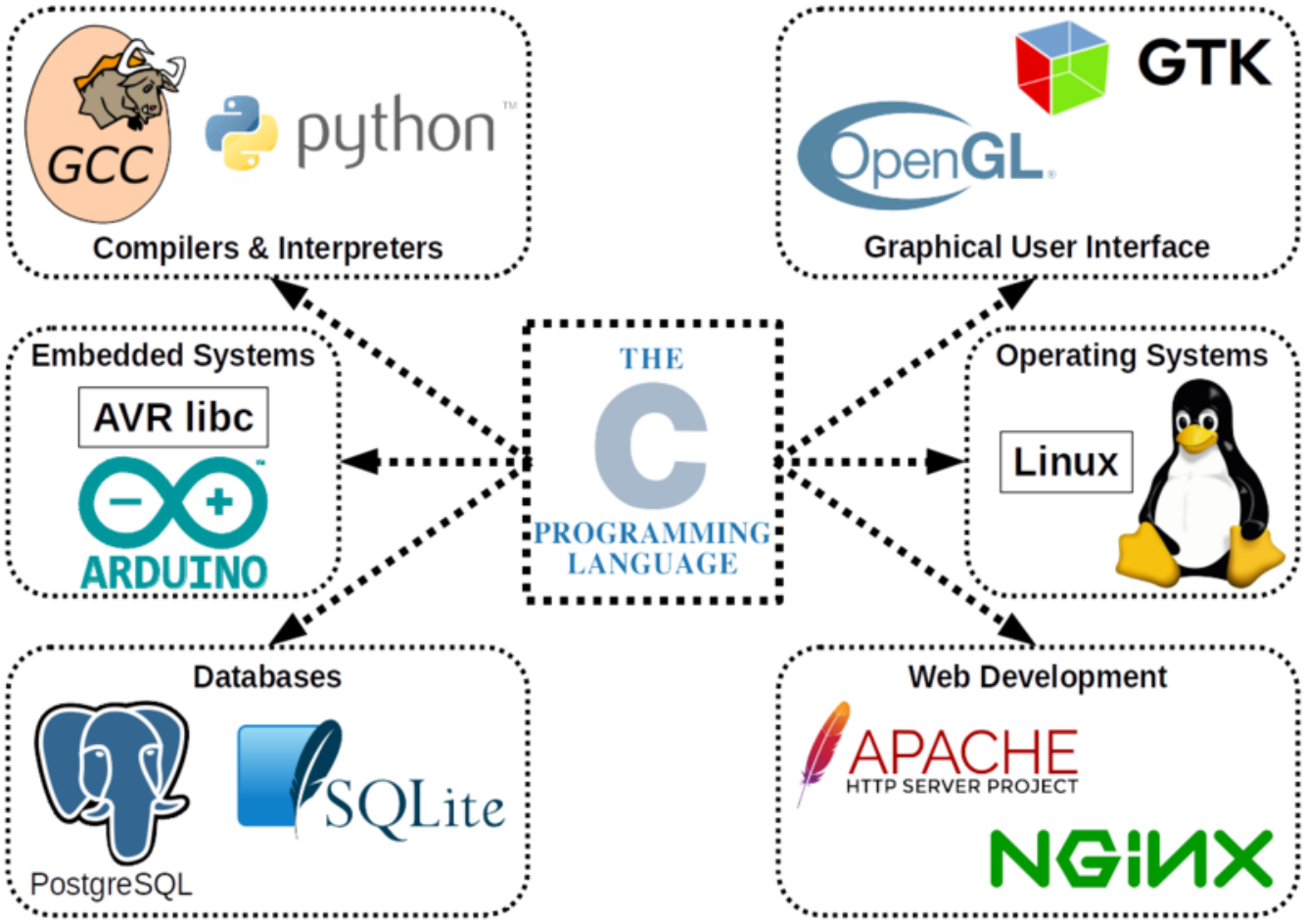
- It is difficult for beginners to learn .
- There are many cases where pointers, which are essential elements for controllin g hardware, are misused .

C language still be used in the future ?

- C language is a common part of C++ and JAVA .
- Programs where execution speed is important are implemented in the C language .
- C language is widely used in

Embedded System : An embedded system is a special purpose system in which a computer is embedded in a device , such as an MP3 player or cell phone .

Uses of the C Language



What you will learn in this chapter

- Understanding the program
- Programming language
- Algorithm
- Scratch

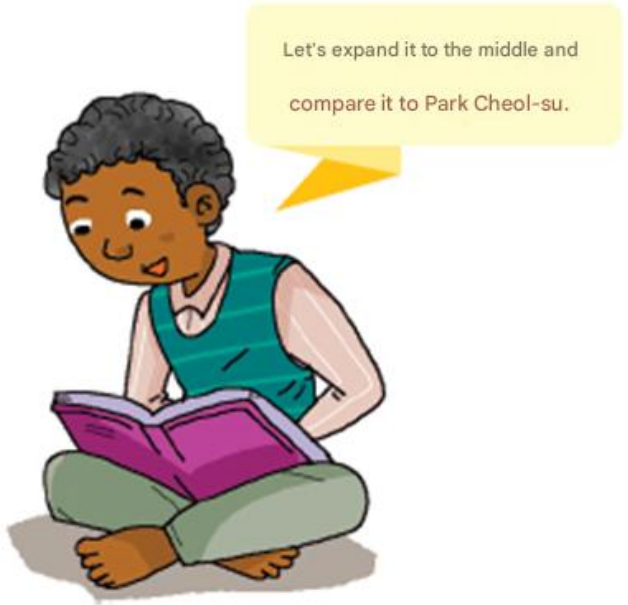
Algorithm

Q) Can anyone cook if they just learn how to use an oven and have the ingredients ?

A) You need to know how to cook .

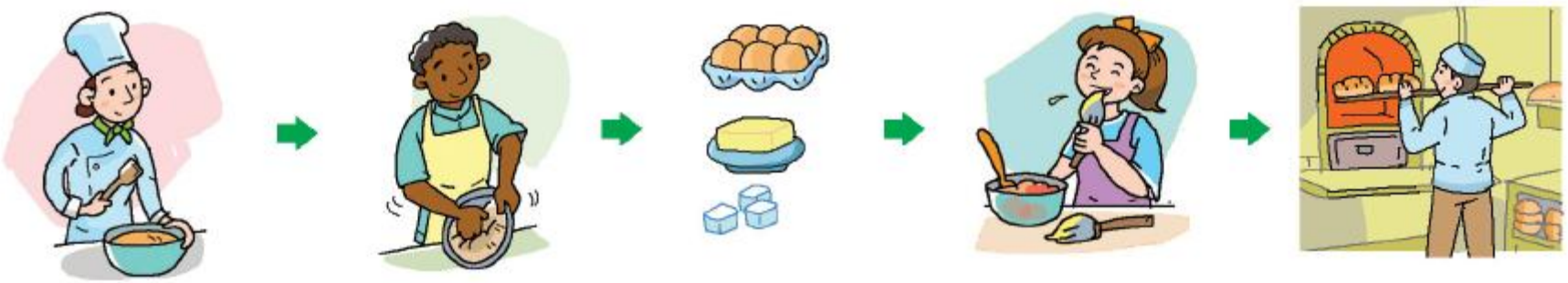
What is an algorithm ?

- Algorithm : A description of the step-by-step procedure that a computer mus t perform to solve a problem
- (Example) Let's consider the problem of finding the phone number of a spe cific person (let's say Park Cheol-su) in a phone book .



Algorithm for making bread

① Prepare an empty bowl .
② Add yeast to flour and milk and stir .
③ Add butter , sugar , and eggs and mix .
④ Leave in a warm place to ferment.
⑤ Bake in an oven at

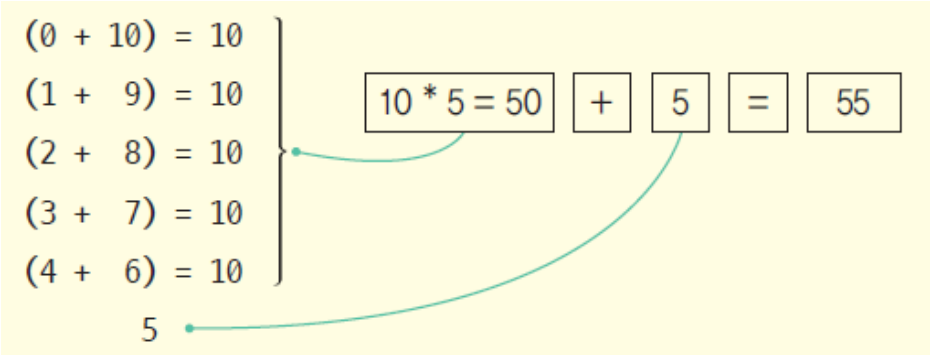


Algorithm to find the sum of numbers f rom 1 to 10

① Add the numbers from 1 to 10 one by one .

1 + 2 + 3 + ... + 10 = 55

② Group the numbers so that the sum of the two numbers is 10 , multiply the number of groups by 10, and add the remaining number 5 .



③ You can also calculate it using the formula .

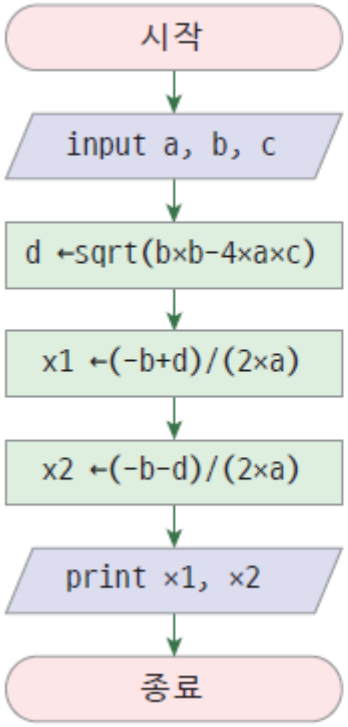
10(1+10)/2=55

The Art of Algorithms

- Natural language (natural language)
- Flowchart
- pseudo-code

Example of an algorithm

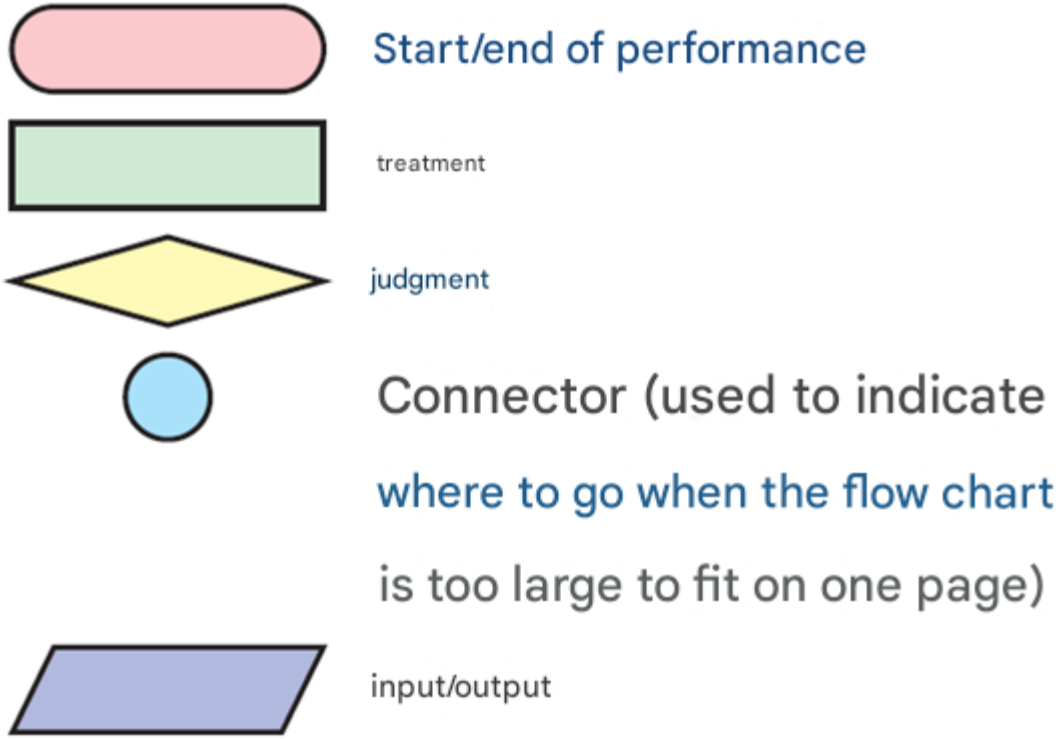
- Algorithm for finding roots of quadratic equations



- Step 1: input a, b, c
- Step 2: d ←sqrt(b×b-4×a×c)
- Step 3: x1 ←(-b+d)/(2×a)
- Step 4: x2 ←(-b-d)/(2×a)
- Step 5: print ×1, ×2

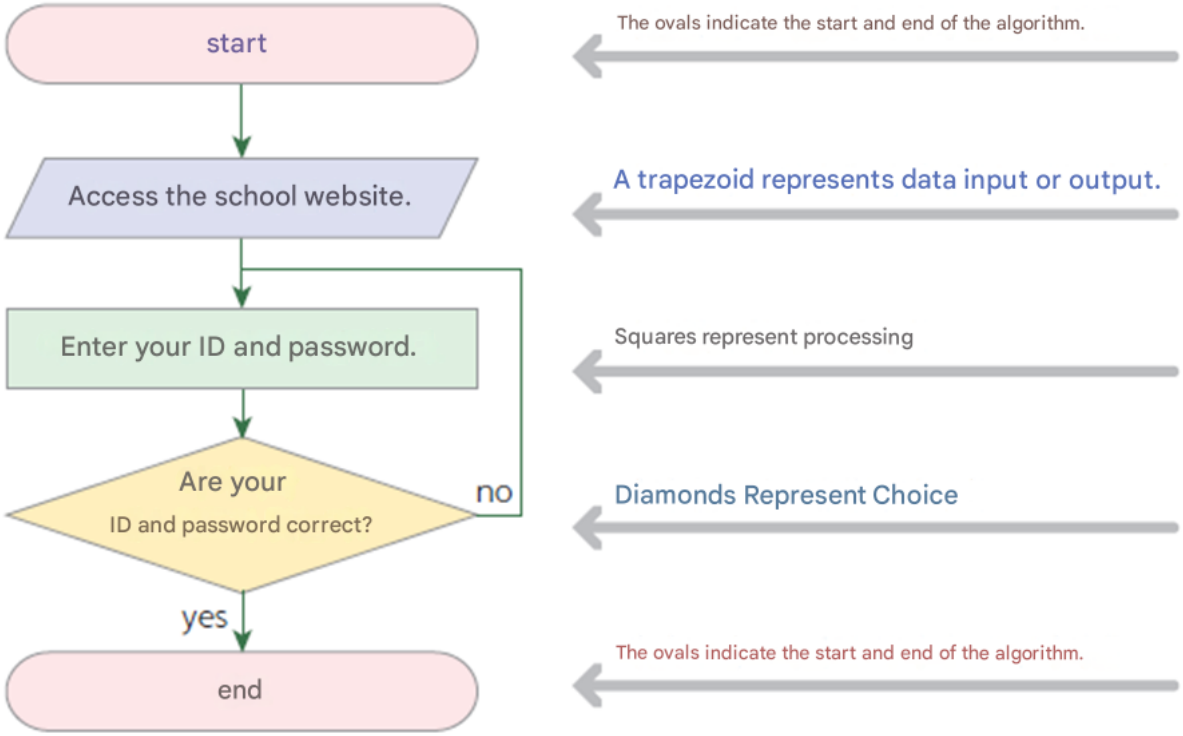
The Art of Algorithms

- Flow chart : A method of graphically representing the logical sequence or sequence of operations in a program.



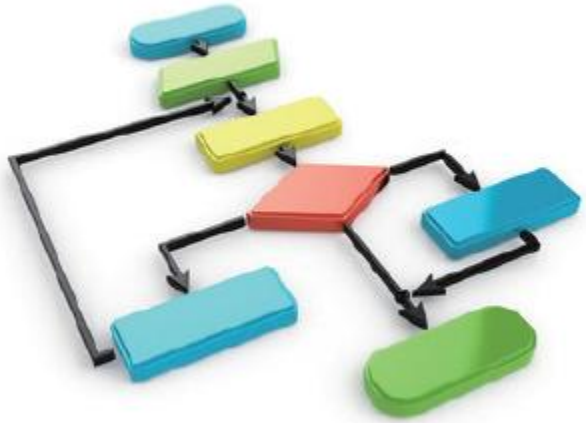
Example of an algorithm

- Let's show the algorithm for logging into the school homepa ge in a flowchart .



Pseudocode

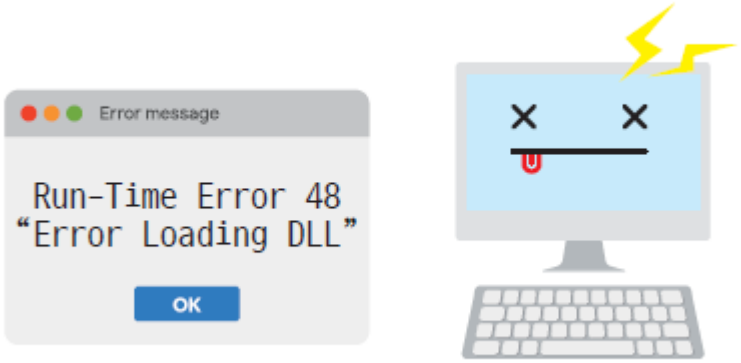
- Pseudo code is a code that is more systematic than natural la nguage and less rigorous than a programming language and is mainly used to express algorithms.
- For example, the algorithm that takes the grades of 10 students and calculates th



```
total ← 0
counter ← 1
while counter <= 10
       input grade
       grade ← grade + total
       counter ← counter + 1
average ← total / 10
print average
```

The importance of algorithms

- If we give computers bad algorithms, we are bound to get b ad results .
- our smartphones, home appliances , and cars have caused so far .
- If there are no errors in the algorithm, the computer program will also operate w ithout logical errors .

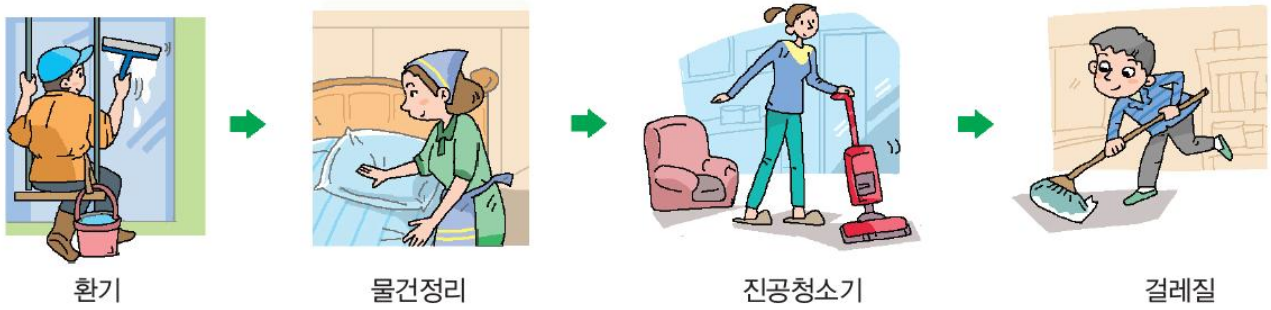


How to create an algorithm

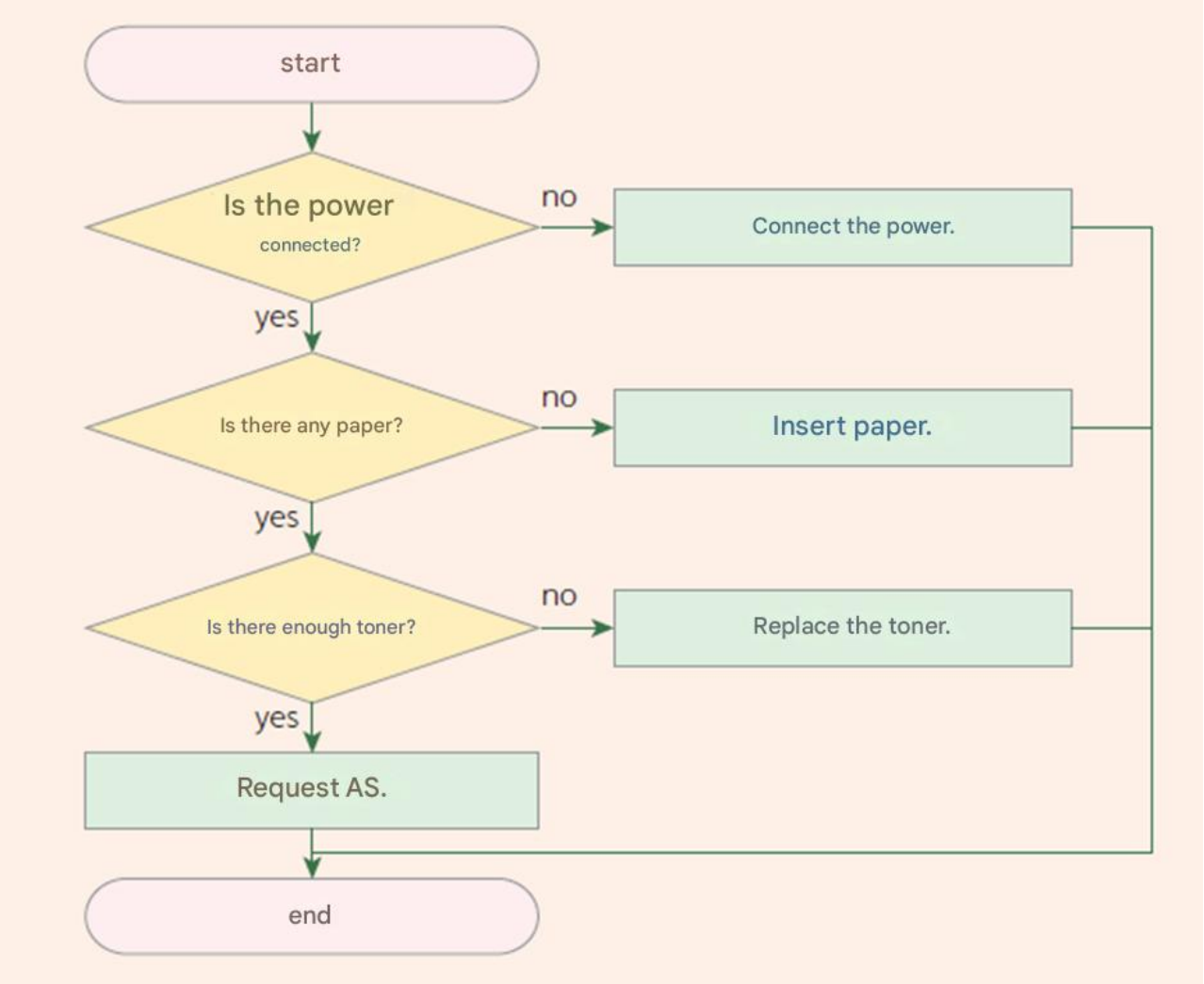
1. a problem all at once, break it down into smaller problem s .
2. Keep breaking the problem down until it is small enough .

① Clean the room .
② Clean the living room .
③ Clean the kitchen .

① Ventilate .
② Organize your things .
③ Turn on the vacuum cleaner .
④ Mopping .

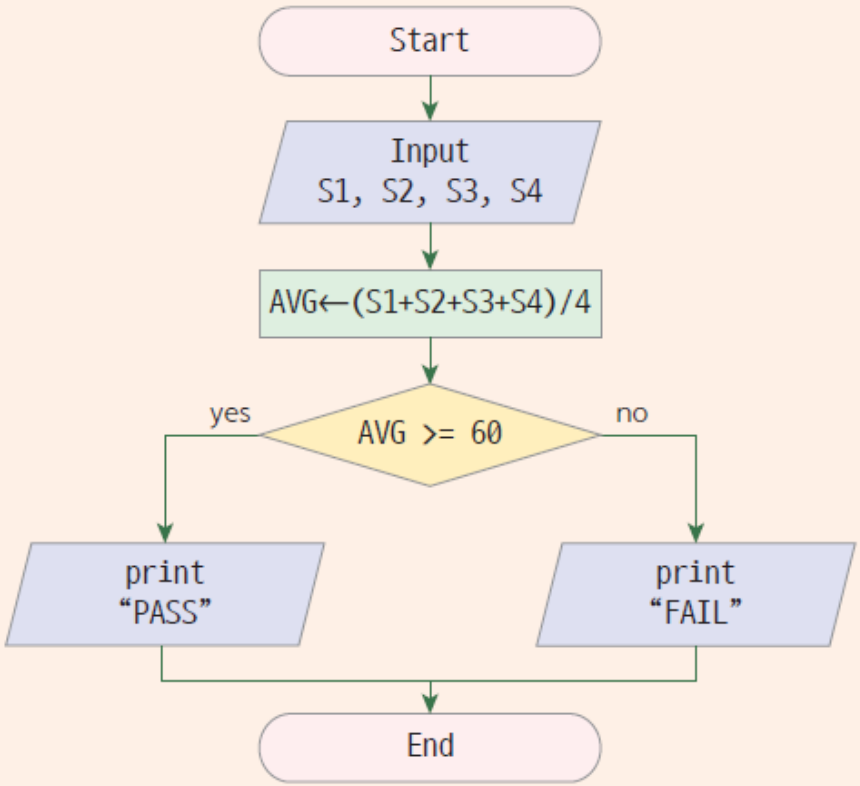


Lab: Algorithm for handling printer failures



Lab: Algorithm for determining pass or fail

Solution



Q & A