

# Chapter 1. 프로그래밍의 개념

## Part 1. 프로그래밍이란?

# Chapter 1. 프로그래밍의 개념



- 프로그래밍이란?
- 프로그래밍 언어
- C언어의 소개
- 알고리즘이란?
- 스크래치

프로그램을  
작성하기에  
앞서서 중요한  
개념들을  
살펴봅니다..



# 컴퓨터(computer) 란?

## □ 컴퓨터(computer)의 종류

- \_\_\_\_\_/ \_\_\_\_\_/ \_\_\_\_\_/ \_\_\_\_\_ ...
- PC, 노트북
- 비행기, 자동차
- 스마트폰,
- 전자계산기 X

계산기는 정해진 기능만을 수행한다. 기능을 변경할 수 없다.

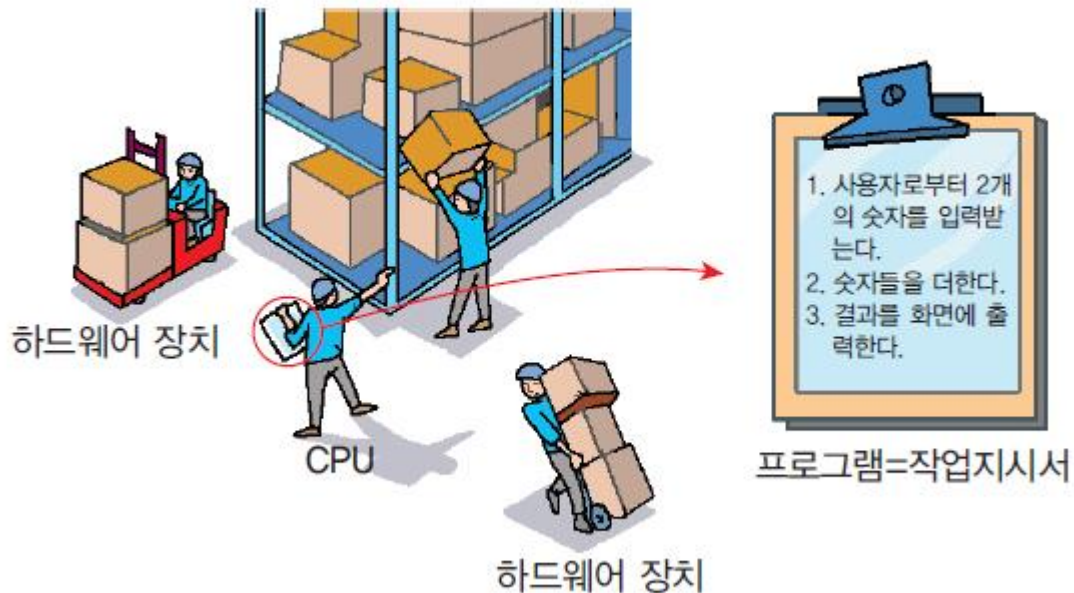


프로그램이라는 개념을 도입하여 수행하는 기능을 쉽게 변경할 수 있다.



# 컴퓨터의 정의

- 컴퓨터(computer)는 단순히 계산(compute)만 하는 기계가 아니다.
- 현대적인 의미에서의 컴퓨터는 **마이크로프로세서(CPU)**를 통해서 **프로그램**(명령어들의 리스트)에 따라 여러가지 작업을 할 수 있는 기계라고 할 수 있다



# 프로그램이란?

- 컴퓨터 = 하드웨어 + 소프트웨어(프로그램)
- 컴퓨터를 범용적(여러가지 목적으로 사용)으로 만드는 것은 바로 프로그램



동영상 재생 프로그램



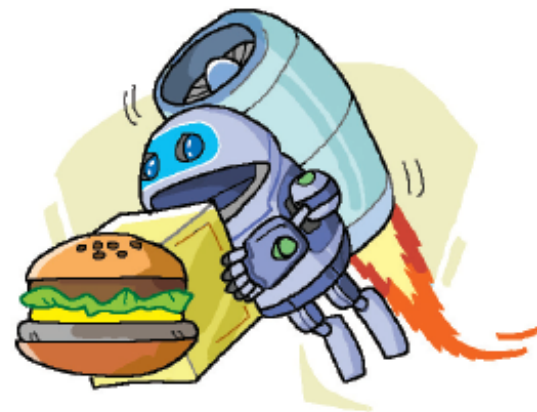
mp3 재생 프로그램



문서 편집 프로그램

# 프로그램의 예

- 로봇에게 가까운 햄버거 가게에 가서 햄버거 사오는 일을 시킨다고 하자. 이 일은 다음과 같은 지시사항들로 이루어 질 수 있다. 이 지시사항들이 바로 명령어이다.
- ▶ 500미터 직진한다.
- ▶ 교차로에서 우회전한다.
- ▶ 1,000미터 직진한다.
- ▶ 도로 왼쪽에서 햄버거 가게를 찾는다.
- ▶ 햄버거를 주문한다.
- ▶ 햄버거를 들고 출발한 위치로 다시 온다.



# 프로그램의 역사

- 프로그래밍이 가능한 최초의 기계: **Analytical Engine (해석 기관)** – 여러 종류의 계산을 하나의 기계에서 할 수 있도록 설계된 최초의 범용 컴퓨터
- 만드인: **찰스 배비지** (1791년 ~ 1871년)
- 수천 개의 기어, 바퀴, 축, 레버 등이 증기로 작동



# 최초의 프로그래머

- 프로그램을 최초로 만든 사람은 **에이다 러브레이스(Ada Lovelace)**
- 에이다는 대문호 바이런의 딸
- 배비지의 해석 기관에 매료되어 해석 기관을 위한 프로그램을 개발하였다.
- 서브루틴(subroutine), 루프(loop), 점프(jump) 등의 핵심적인 컴퓨터 프로그래밍 기본 원리를 고안





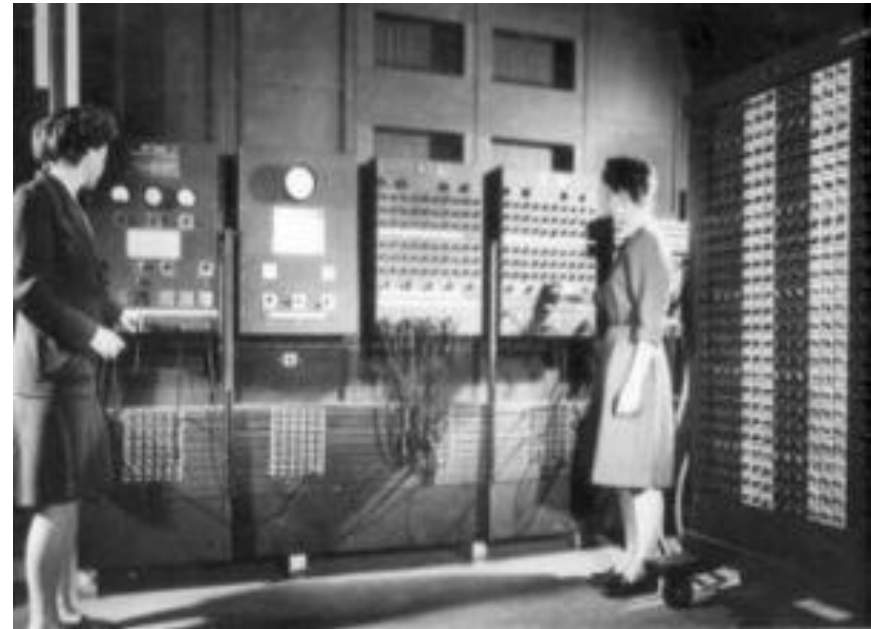
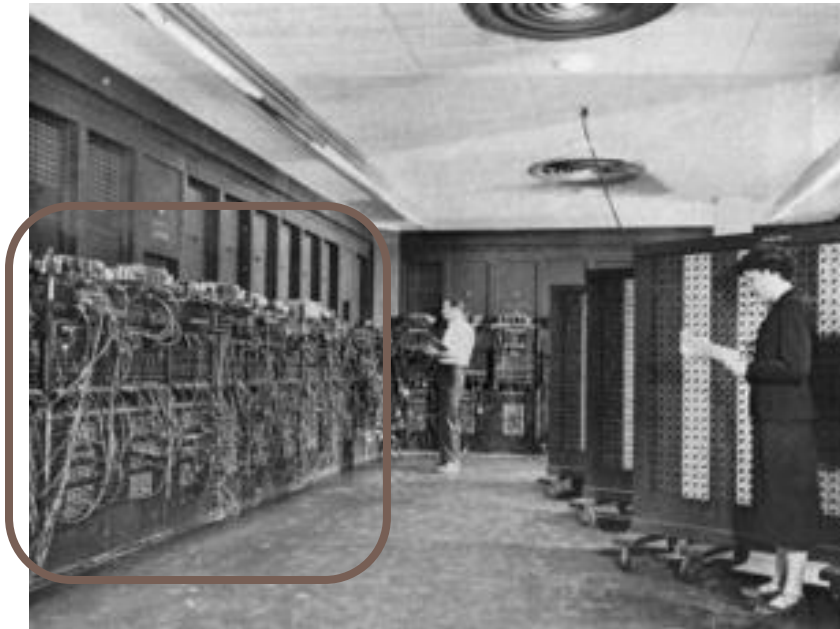
# 에이다의 프로그램

Diagram for the computation by the Engine of the Numbers of Bernoulli. See Note G. (page 72<sup>2</sup> *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 <sub>1</sub>	1V <sub>2</sub>	1V <sub>3</sub>	0V <sub>4</sub>	0V <sub>5</sub>	0V <sub>6</sub>	0V <sub>7</sub>	0V <sub>8</sub>	0V <sub>9</sub>	0V <sub>10</sub>	0V <sub>11</sub>	0V <sub>12</sub>	0V <sub>13</sub>	0V <sub>14</sub>	0V <sub>15</sub>	0V <sub>16</sub>	0V <sub>17</sub>	0V <sub>18</sub>	0V <sub>19</sub>	0V <sub>20</sub>	0V <sub>21</sub>	0V <sub>22</sub>	0V <sub>23</sub>	0V <sub>24</sub>			
						1	2	n																								
1	x	1V <sub>2</sub> x 1V <sub>3</sub>	1V <sub>4</sub> , 1V <sub>5</sub> , 1V <sub>6</sub>	$\begin{cases} 1V_2 = 1V_2 \\ 1V_3 = 1V_3 \\ 1V_4 = 1V_4 \\ 1V_5 = 1V_5 \\ 1V_6 = 1V_6 \end{cases}$	= 2n	...	2	n	2n	2n	2n																					
2	-	1V <sub>4</sub> - 1V <sub>1</sub>	2V <sub>4</sub>	$\begin{cases} 1V_4 = 2V_4 \\ 1V_1 = 1V_1 \end{cases}$	= 2n - 1	...	1	...	...	2n - 1																						
3	+	1V <sub>5</sub> + 1V <sub>1</sub>	2V <sub>5</sub>	$\begin{cases} 1V_5 = 2V_5 \\ 1V_1 = 1V_1 \end{cases}$	= 2n + 1	...	1	...	...	2n + 1																						
4	+	2V <sub>5</sub> + 2V <sub>4</sub>	1V <sub>11</sub>	$\begin{cases} 2V_5 = 1V_{11} \\ 2V_4 = 0V_4 \end{cases}$	$\begin{matrix} 2n - 1 \\ 2n + 1 \end{matrix}$	...	...	...	0	0																						
5	+	1V <sub>11</sub> + 1V <sub>2</sub>	2V <sub>11</sub>	$\begin{cases} 1V_{11} = 2V_{11} \\ 1V_2 = 1V_2 \end{cases}$	$\begin{matrix} 1 \cdot 2n - 1 \\ 2 \cdot 2n + 1 \end{matrix}$	...	2	...																								
6	-	0V <sub>13</sub> - 2V <sub>11</sub>	1V <sub>13</sub>	$\begin{cases} 2V_{11} = 0V_{13} \\ 0V_{13} = 1V_{13} \end{cases}$	= $-\frac{1}{2} \cdot 2n - 1 = A_0$	...	...	...																								
7	-	1V <sub>5</sub> - 1V <sub>1</sub>	1V <sub>10</sub>	$\begin{cases} 1V_5 = 1V_1 \\ 1V_1 = 1V_1 \end{cases}$	= n - 1 (= 3)	...	1	...	n																							
8	+	1V <sub>2</sub> + 0V <sub>7</sub>	1V <sub>7</sub>	$\begin{cases} 1V_2 = 1V_7 \\ 0V_7 = 1V_7 \end{cases}$	= 2 + 0 = 2	...	2	...				2																				
9	+	1V <sub>6</sub> + 1V <sub>7</sub>	3V <sub>11</sub>	$\begin{cases} 1V_6 = 1V_7 \\ 1V_7 = 3V_{11} \end{cases}$	= $\frac{2n}{2} = A_1$	...	...	...			2n	2																				
10	x	1V <sub>21</sub> x 1V <sub>11</sub>	1V <sub>12</sub>	$\begin{cases} 1V_{21} = 1V_{12} \\ 1V_{11} = 2V_{11} \end{cases}$	= $B_1 \cdot \frac{2n}{2} = B_1 A_1$	...	...	...																								
11	+	1V <sub>12</sub> + 1V <sub>13</sub>	2V <sub>13</sub>	$\begin{cases} 1V_{12} = 0V_{13} \\ 1V_{13} = 2V_{13} \end{cases}$	= $-\frac{1}{2} \cdot 2n - 1 + B_1 \cdot \frac{2n}{2}$	...	...	...																								
12	-	1V <sub>10</sub> - 1V <sub>1</sub>	2V <sub>10</sub>	$\begin{cases} 1V_{10} = 2V_{10} \\ 1V_1 = 1V_1 \end{cases}$	= n - 2 (= 2)	...	1	...																								
13	[	1V <sub>6</sub> - 1V <sub>1</sub>	2V <sub>6</sub>	$\begin{cases} 1V_6 = 2V_6 \\ 1V_1 = 1V_1 \end{cases}$	= 2n - 1	...	1	...			2n - 1																					
14		+ 1V <sub>1</sub> + 1V <sub>7</sub>	2V <sub>7</sub>	$\begin{cases} 1V_1 = 1V_7 \\ 1V_7 = 2V_7 \end{cases}$	= 2 + 1 = 3	...	1	...				3																				
15		+ 2V <sub>6</sub> + 2V <sub>7</sub>	1V <sub>8</sub>	$\begin{cases} 2V_6 = 1V_8 \\ 2V_7 = 2V_7 \end{cases}$	= $\frac{2n - 1}{3}$	...	...	...			2n - 1	3	$\frac{2n - 1}{3}$																			
16		x 1V <sub>8</sub> x 3V <sub>11</sub>	4V <sub>11</sub>	$\begin{cases} 1V_8 = 4V_{11} \\ 3V_{11} = 4V_{11} \end{cases}$	= $\frac{2n}{2} \cdot \frac{2n - 1}{3}$	...	...	...					0																			
17		- 2V <sub>6</sub> - 1V <sub>1</sub>	3V <sub>6</sub>	$\begin{cases} 2V_6 = 3V_6 \\ 1V_1 = 1V_1 \end{cases}$	= 2n - 2	...	1	...			2n - 2																					
18		+ 1V <sub>1</sub> + 2V <sub>7</sub>	3V <sub>7</sub>	$\begin{cases} 1V_1 = 3V_7 \\ 2V_7 = 3V_7 \end{cases}$	= 3 + 1 = 4	...	1	...				4																				
19		+ 3V <sub>6</sub> + 3V <sub>7</sub>	1V <sub>9</sub>	$\begin{cases} 3V_6 = 1V_9 \\ 3V_7 = 3V_7 \end{cases}$	= $\frac{2n - 2}{4}$	...	...	...			2n - 2	4	$\frac{2n - 2}{4}$																			
20		x 1V <sub>9</sub> x 4V <sub>11</sub>	0V <sub>11</sub>	$\begin{cases} 1V_9 = 0V_{11} \\ 4V_{11} = 0V_{11} \end{cases}$	= $\frac{2n}{2} \cdot \frac{2n - 1}{3} \cdot \frac{2n - 2}{4} = A_3$	...	...	...					0																			
21		x 1V <sub>22</sub> x 5V <sub>11</sub>	0V <sub>12</sub>	$\begin{cases} 1V_{22} = 1V_{20} \\ 0V_{12} = 2V_{12} \end{cases}$	= $B_3 \cdot \frac{2n}{2} \cdot \frac{2n - 1}{3} \cdot \frac{2n - 2}{3} = B_3 A_3$	...	...	...																								
22		+ 2V <sub>12</sub> + 2V <sub>13</sub>	3V <sub>13</sub>	$\begin{cases} 2V_{12} = 0V_{12} \\ 2V_{13} = 3V_{13} \end{cases}$	= $A_0 + B_1 A_1 + B_3 A_3$	...	...	...																								
23		- 2V <sub>10</sub> - 1V <sub>1</sub>	3V <sub>10</sub>	$\begin{cases} 2V_{10} = 3V_{10} \\ 1V_1 = 1V_1 \end{cases}$	= n - 3 (= 1)	...	1	...																								
Here follows a repetition of Operations thirteen to twenty-three.																																
24	+	4V <sub>13</sub> + 0V <sub>24</sub>	1V <sub>24</sub>	$\begin{cases} 4V_{13} = 0V_{24} \\ 0V_{24} = 1V_{24} \end{cases}$	= B <sub>7</sub>	...	...	...																								
25	+	1V <sub>1</sub> + 1V <sub>3</sub>	1V <sub>3</sub>	$\begin{cases} 1V_1 = 1V_3 \\ 1V_3 = 1V_3 \end{cases}$	= n + 1 = 4 + 1 = 5	...	1	...	n + 1			0	0																			
				by a Variable-card.																												
				by a Variable card.																												

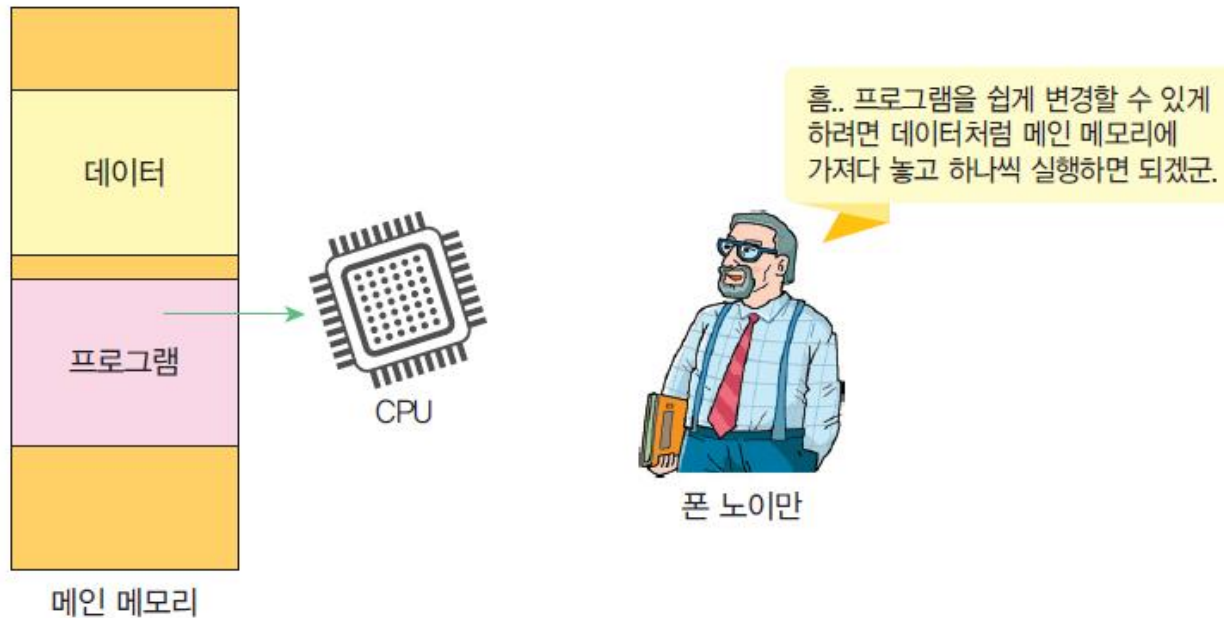
# 초기 컴퓨터의 프로그래밍

- 초기 컴퓨터인 **ENIAC**의 프로그램은 스위치에 의하여 기억되었고 프로그램을 변경할 때마다 그 많은 스위치들을 처음부터 다시 연결하여야 했다.



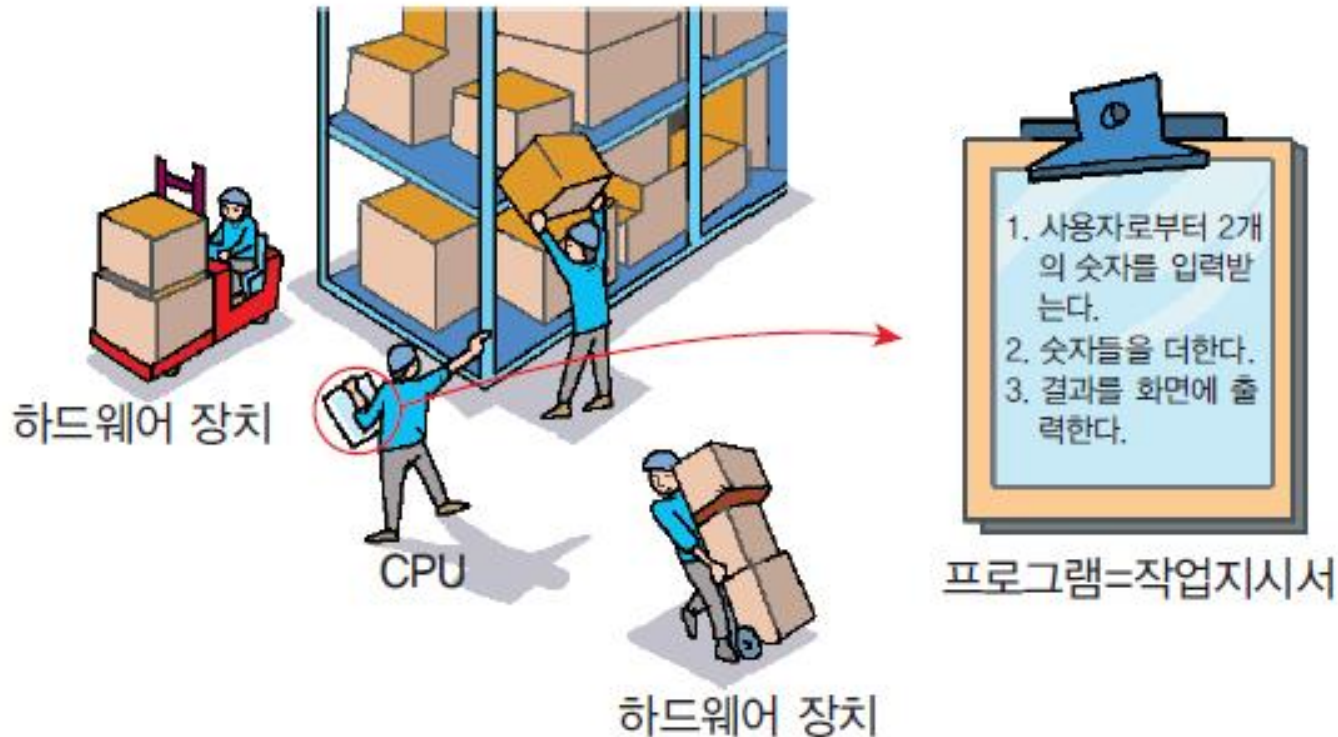
# 폰노이만 구조

- 프로그램은 메인 메모리에 저장된다.
- 메인 메모리에 저장된 프로그램에서 명령어들을 순차적으로 가져와서 실행한다.



# 프로그램 == 작업지시서

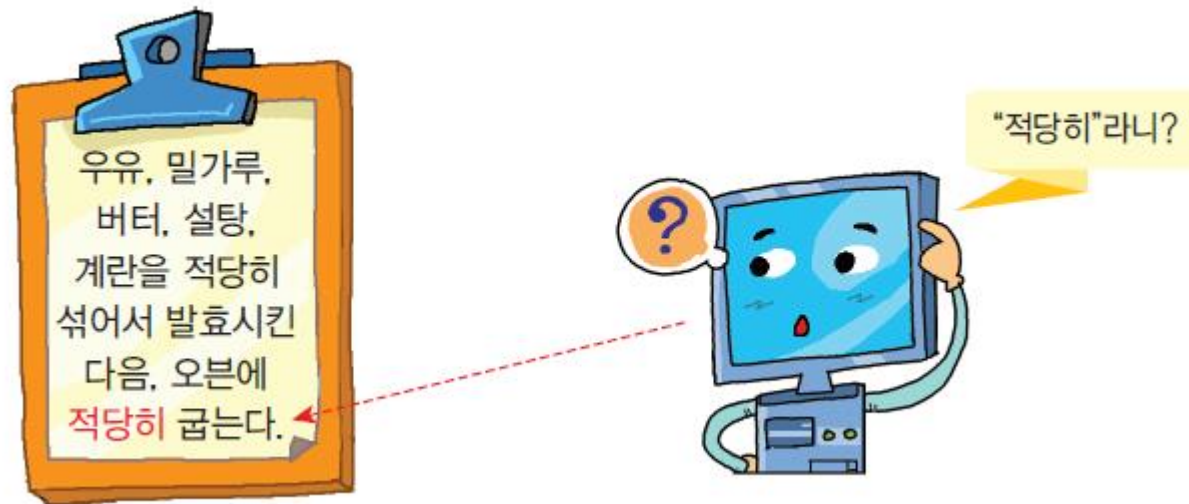
- 프로그램: 컴퓨터에게 해야 할 작업의 내용을 알려주는 문서



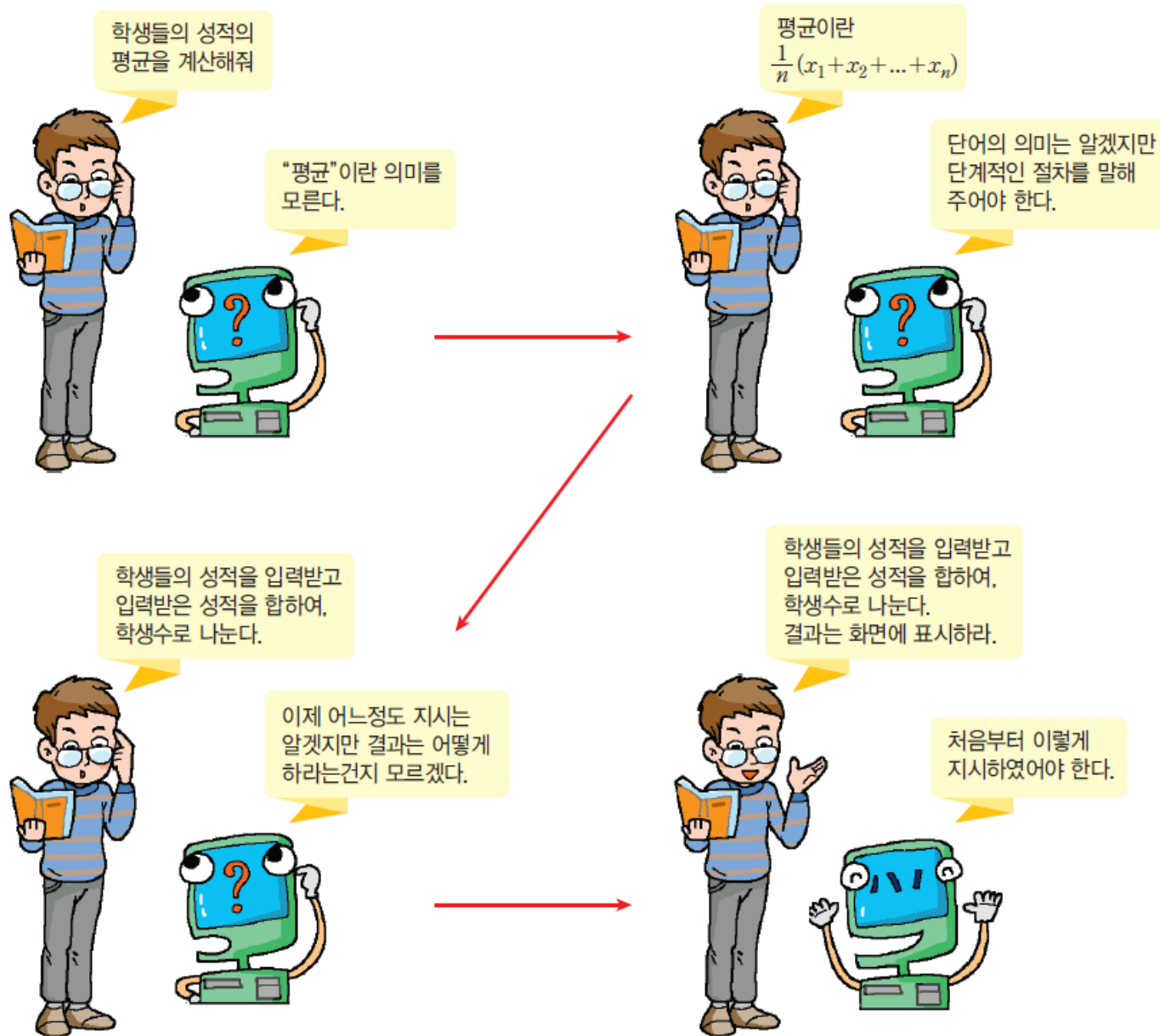
# 작업을 지시하는 방법

Q) 컴퓨터에게 어떻게 작업을 시킬 수 있을까?

A) 상식이나 지능이 없기 때문에 아주 자세하고 구체적으로 일을 지시하여야 한다.








□ [https://www.youtube.com/watch?v=cDA3\\_5982h8](https://www.youtube.com/watch?v=cDA3_5982h8)

# Chapter 1. 프로그래밍의 개념

## Part 2. 프로그래밍 언어

# 이번 장에서 학습할 내용

- 
- 프로그래밍이란?
  - 프로그래밍 언어
  - C언어의 소개
  - 알고리즘이란?
  - 스크래치

프로그램을  
작성하기에  
앞서서  
중요한  
개념들을  
살펴봅니다..

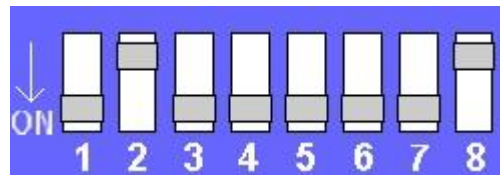




# 기계어

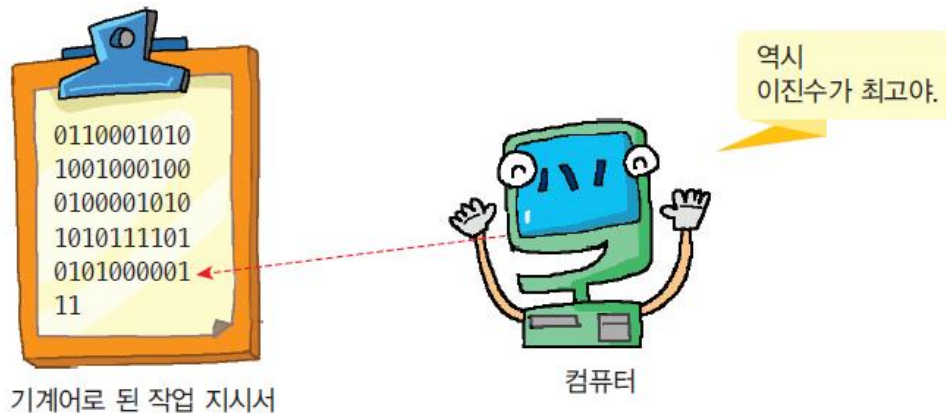
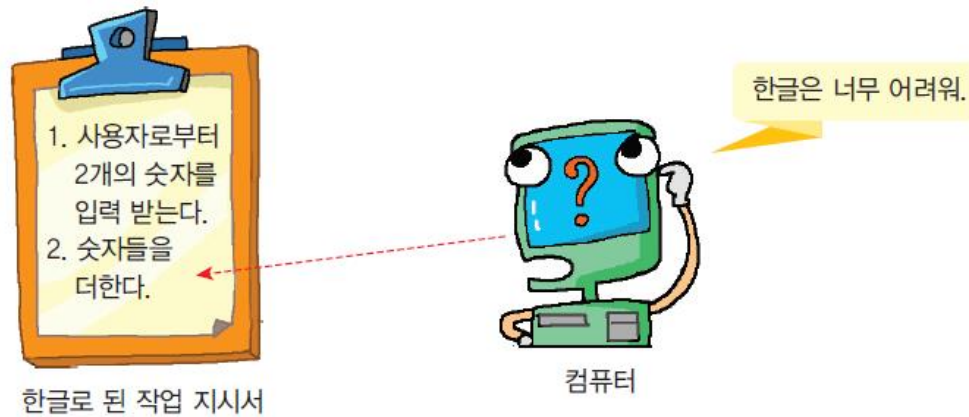
Q) 컴퓨터가 이해할 수 있는 언어는 어떤 것인가?

- A) 컴퓨터가 알아듣는 언어는 한가지이다. 즉 0과 1로 구성되어 있는 "001101110001010..."과 같은 기계어이다.
- A) 컴퓨터는 모든 것을 0과 1로 표현하고 0과 1에 의하여 내부 스위치 회로들이 ON/OFF 상태로 변경되면서 작업을 한다.



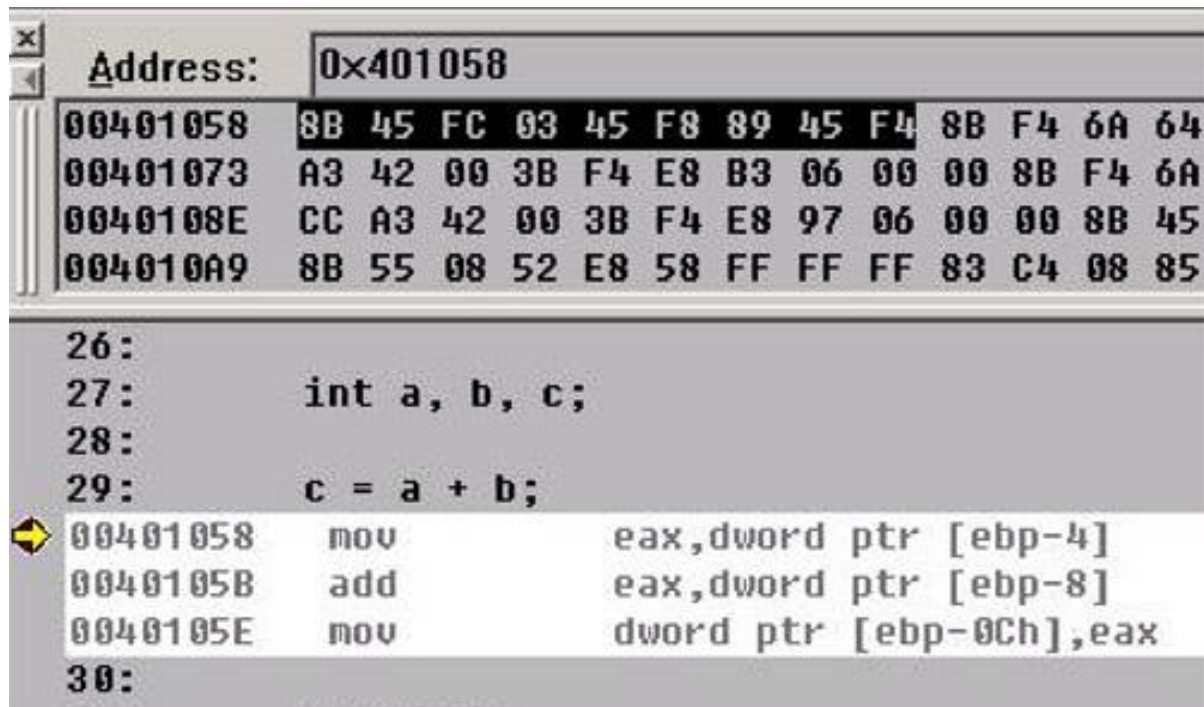
# 기계어

- 컴퓨터는 기계어를 바로 이해할 수 있다.



# 기계어

## □ 기계어의 예



The screenshot shows a debugger window with two panes. The top pane displays a memory dump starting at address 0x401058. The bottom pane shows the corresponding assembly code. The assembly code includes variable declarations and an addition operation. A yellow arrow points to the first instruction in the assembly pane.

Address	Hex Dump
00401058	8B 45 FC 03 45 F8 89 45 F4 8B F4 6A 64
00401073	A3 42 00 3B F4 E8 B3 06 00 00 8B F4 6A
0040108E	CC A3 42 00 3B F4 E8 97 06 00 00 8B 45
004010A9	8B 55 08 52 E8 58 FF FF FF 83 C4 08 85

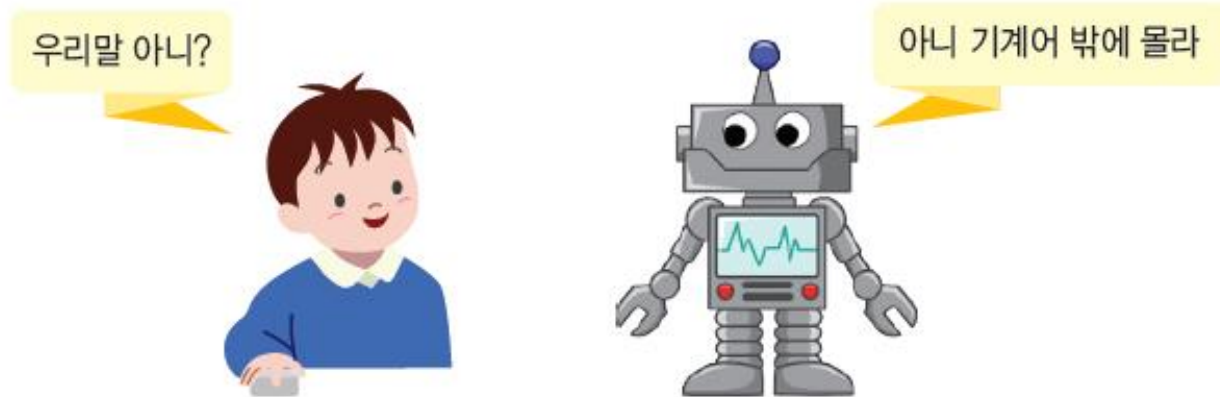
  

Address	Disassembly
26:	
27:	int a, b, c;
28:	
29:	c = a + b;
00401058	mov eax,dword ptr [ebp-4]
0040105B	add eax,dword ptr [ebp-8]
0040105E	mov dword ptr [ebp-0Ch],eax
30:	

# 프로그래밍 언어의 필요성

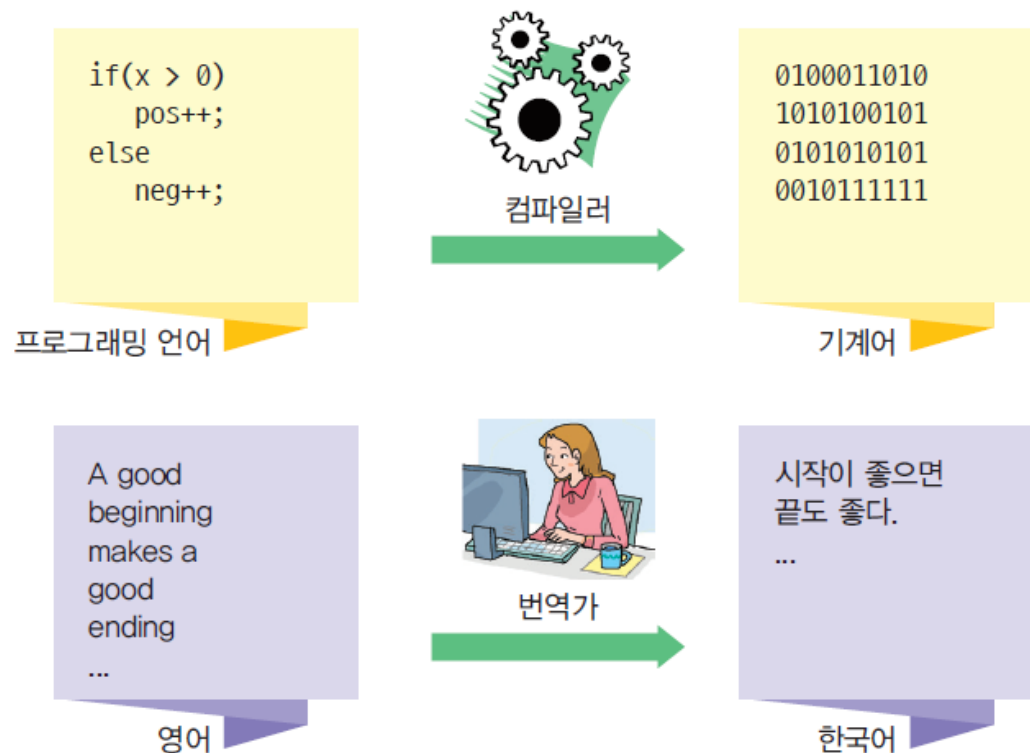
## Q) 그렇다면 인간이 기계어를 사용하면 어떤가?

- 기계어를 사용할 수는 있으나 이진수로 프로그램을 작성하여야 하기 때문에 아주 불편하다.
- 프로그래밍 언어는 자연어와 기계어 중간쯤에 위치
- 컴파일러가 프로그래밍 언어를 기계어로 번역



# 컴파일러

- 컴파일러(compiler)는 인간(프로그래밍 언어)과 컴퓨터 사이의 번역기라 할 수 있다.
- 프로그래밍 언어 종류: C, C++, Java, Python 등



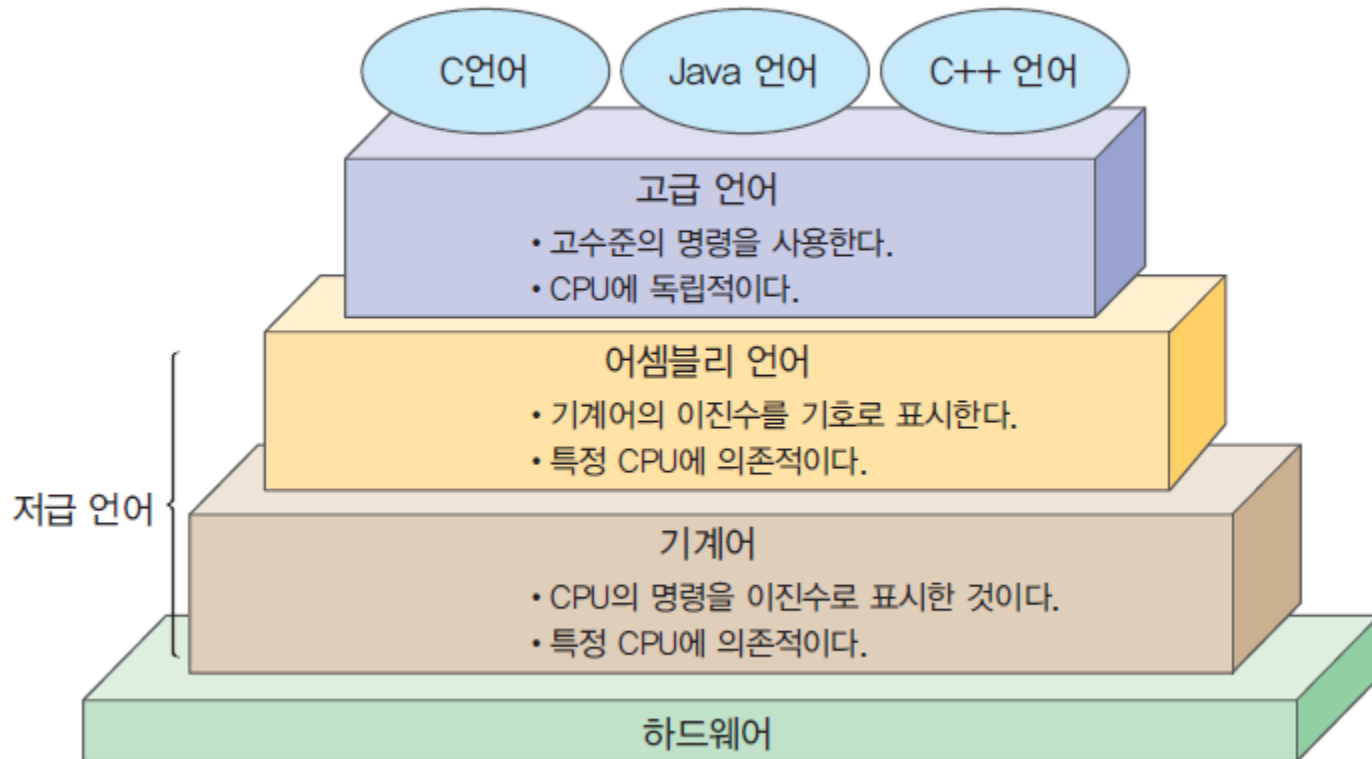
# 중간 점검

1. 왜 계산기는 컴퓨터라고 할 수 없는가?
2. 컴퓨터가 가장 쉽게 이해하는 언어는 무엇인가?
3. 컴파일러는 어떤 역할을 하는가?



# 프로그래밍 언어의 분류

- 기계어(machine language)
- 어셈블리어(assembly language)
- 고급 언어(high-level language)



# 기계어

- 특정 컴퓨터의 명령어(instruction)를 이진수로 표시한 것
- 0과 1로 구성
- 하드웨어에 종속

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



# 어셈블리어

- CPU의 명령어들을 이진수가 아닌 영어의 약자인 기호로 표기
- 기호와 CPU의 명령어가 일대일 대응
- 어셈블러(assembly): 기호를 이진수로 변환하는 프로그램

```
MOV AX, MIDSCORE  
MOV CX, FINALSORE  
ADD AX CX  
MOV TOTALSCORE, AX
```

# 고급언어

- 특정한 컴퓨터의 구조나 프로세서에 무관하게, 독립적으로 프로그램을 작성할 수 있는 언어
- C, C++, JAVA, Python, FORTRAN, PASCAL
- 컴파일러: 고급 언어 문장을 기계어로 변환하는 프로그램

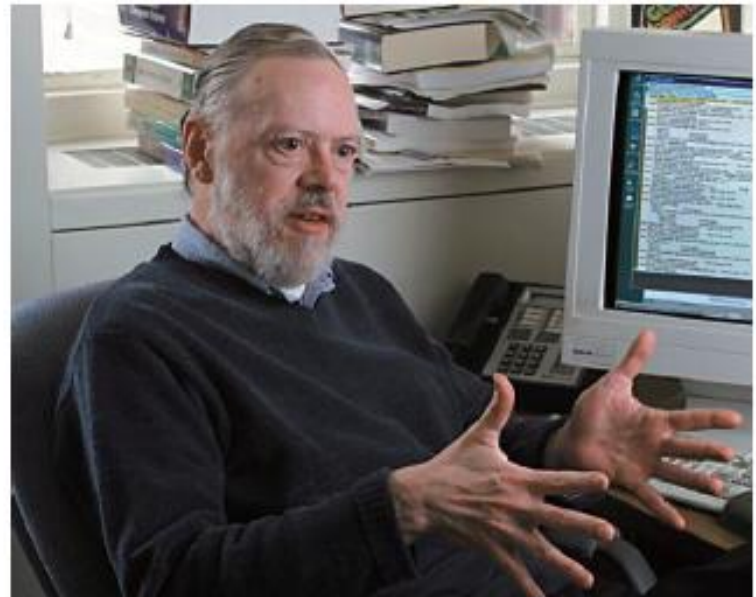
```
TotalScore = MidScore + FinalScore;
```

# Chapter 1. 프로그래밍의 개념

Part 3. C언어의 소개  
알고리즘이란?

# C

- 1970년대 초 AT&T Bell Lab. 의 Dennis Ritchie 에 의하여 개발
- **UNIX** 운영 체제 개발에 필요해서 만들어짐
- 처음부터 전문가용 언어로 출발



# C언어의 버전

## □ K&R C

- ▣ 1978년 “C Programming Language” 책 출간
- ▣ 비공식적인 명세서 역할

## □ ANSI C

- ▣ 1983년 ANSI(American National Standards Institute)는 X3J11이라는 위원회에 의한 표준

## □ C99

- ▣ 1999년에 ISO에 의한 표준
- ▣ C++에서 사용되는 특징 추가
- ▣ 점차 많은 컴파일러에서 지원

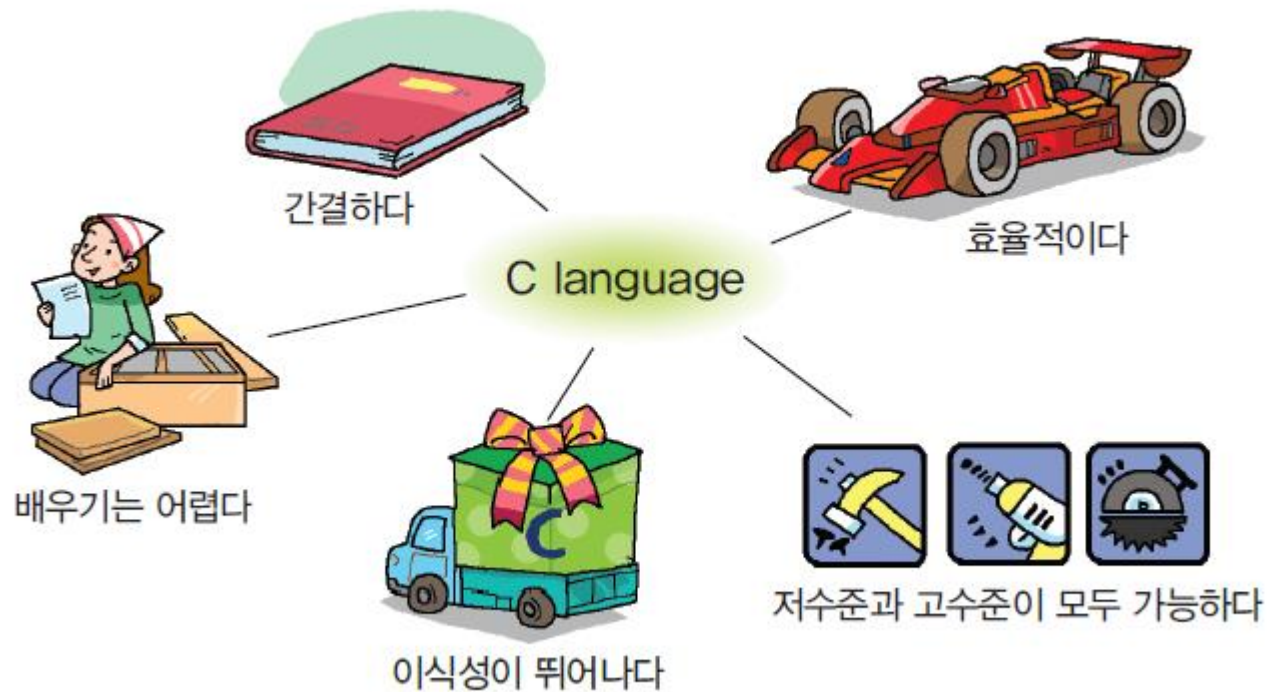
## □ C11

- ▣ ISO에 의하여 2011년 12월에 발표된 C언어 표준이다

# C언어의 특징

- 표기법이 간결하다.
- 효율적이다.
  - ▣ C언어로 만들어진 프로그램이 크기가 작고 속도가 빠르며, 메모리를 효과적으로 사용
- C 언어는 하드웨어를 직접 제어하는 하는 저수준의 프로그래밍도 가능하고 고수준의 프로그래밍도 가능하다.
- C언어는 이식성(portability)이 뛰어나다.
- 초보자가 배우기가 어렵다.

# C언어의 특징



# 이번 장에서 학습할 내용



- 프로그래밍이란?
- 프로그래밍 언어
- C언어소개
- 알고리즘
- 스크래치





# 알고리즘 (Algorithm)

Q) 오븐의 사용법만 배우고 음식 재료만 있으면 누구나 요리가 가능한가?

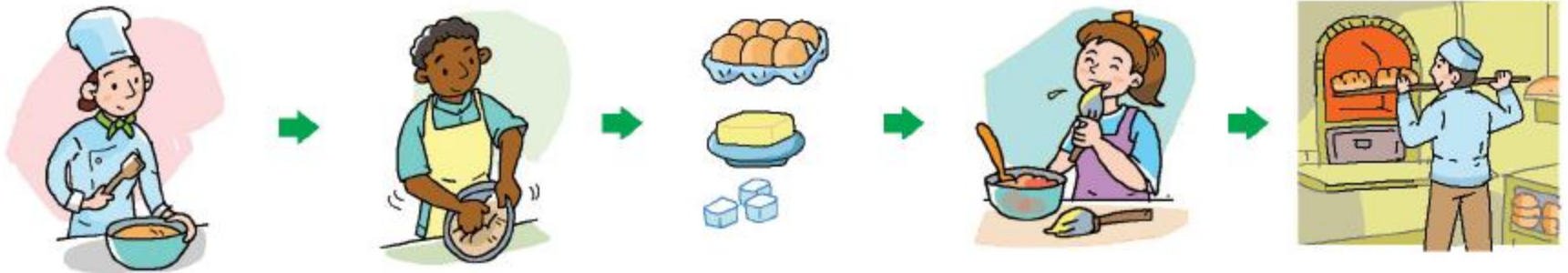
A) 요리법을 알아야 한다.

- 프로그래밍언어가 오븐이고 프로그램이 요리와 같다면, 알고리즘은 요리법에 해당한다.
- 알고리즘(algorithm): 문제를 해결하는 절차(방법)



# 빵을 만드는 알고리즘

- ① 빈 그릇을 준비한다.
- ② 이스트를 밀가루, 우유에 넣고 저어준다.
- ③ 버터, 설탕, 계란을 추가로 넣고 섞는다.
- ④ 따뜻한 곳에 놓아두어 발효시킨다
- ⑤ 170~180도의 오븐에서 굽는다



# 1부터 10까지의 합을 구하는 알고리즘

- ① 1부터 10까지의 숫자를 직접 하나씩 더한다.

$$1 + 2 + 3 + \dots + 10 = 55$$

- ② 두수의 합이 10이 되도록 숫자들을 그룹핑하여 그룹의 개수에 10을 곱하고 남은 숫자 5를 더한다.

The diagram illustrates the second algorithm. On the left, five pairs of numbers are listed, each summing to 10:  $(0 + 10) = 10$ ,  $(1 + 9) = 10$ ,  $(2 + 8) = 10$ ,  $(3 + 7) = 10$ , and  $(4 + 6) = 10$ . These are grouped by a large right-facing curly bracket. A green dot is placed at the middle of this bracket, with a green curved arrow pointing to the first box of a sequence of boxes:  $10 * 5 = 50$ ,  $+$ ,  $5$ ,  $=$ , and  $55$ . Another green dot is placed below the number 5, with a green curved arrow pointing to the box containing the number 5 in the same sequence.

- ③ 공식을 이용하여 계산할 수도 있다.

$$10(1+10)/2=55$$

# 알고리즘의 기술

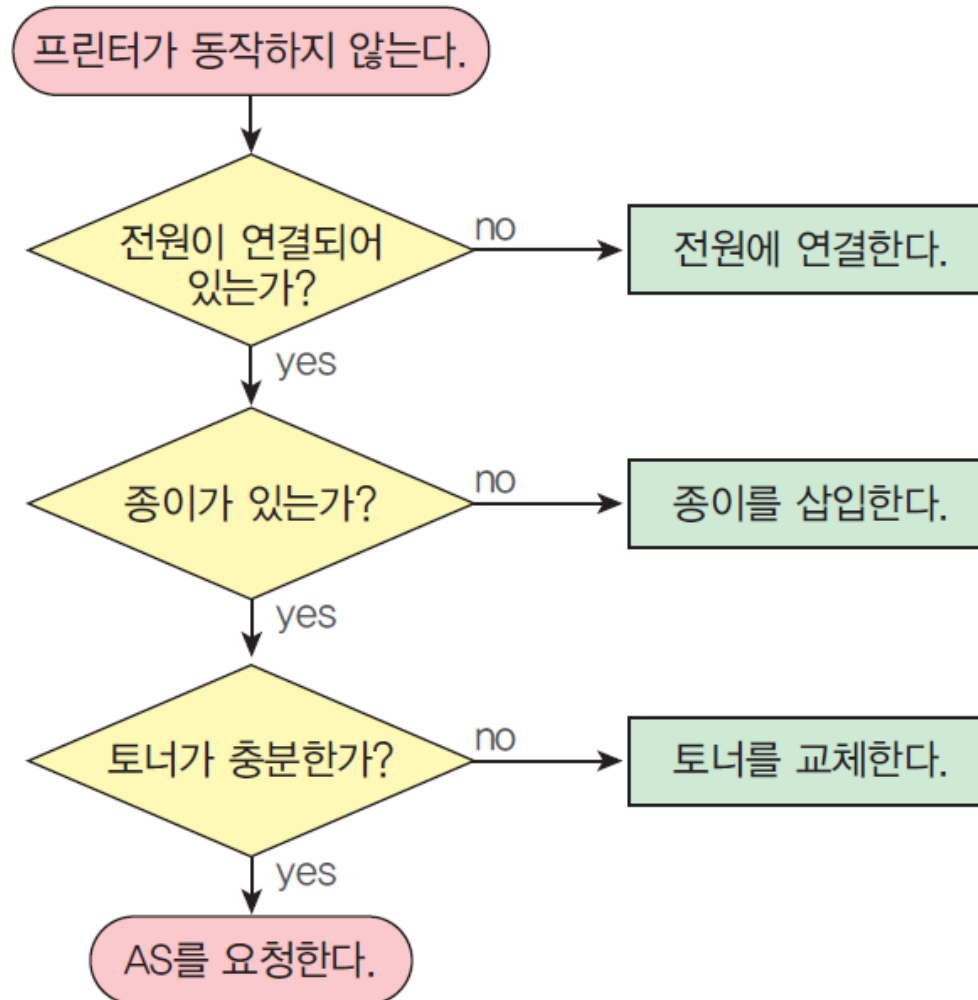
- 자연어(natural language)
- 순서도(flowchart)
- 의사 코드(pseudo-code)

# 알고리즘의 기술

- **순서도(flow chart):** 프로그램에서의 논리 순서 또는 작업 순서를 그림으로 표현하는 방법



# 알고리즘의 예: 프린터 고장 처리



# Pseudocode

- **Pseudocode(의사 코드):** 자연어보다는 더 체계적이고 프로그래밍 언어보다는 덜 엄격한 언어로서 알고리즘의 표현에 주로 사용되는 코드

알고리즘 GetLargest

입력: 숫자들의 리스트 L.

출력: 리스트에서 가장 큰 값

```
largest ← L[0]
for each n in L do
    if n > largest then
        largest ← n
return largest
```

# 알고리즘을 만드는 방법

1. 문제를 한 번에 해결하려고 하지 말고 더 작은 크기의 문제들로 분해한다.
2. 문제가 충분히 작아질 때까지 계속해서 분해한다.

① 방을 청소한다.  
② 거실을 청소한다.  
③ 부엌을 청소한다.

① 환기를 시킨다.  
② 물건들을 정리한다.  
③ 진공 청소기를 돌린다.  
④ 걸레질을 한다.



환기



물건정리



진공청소기



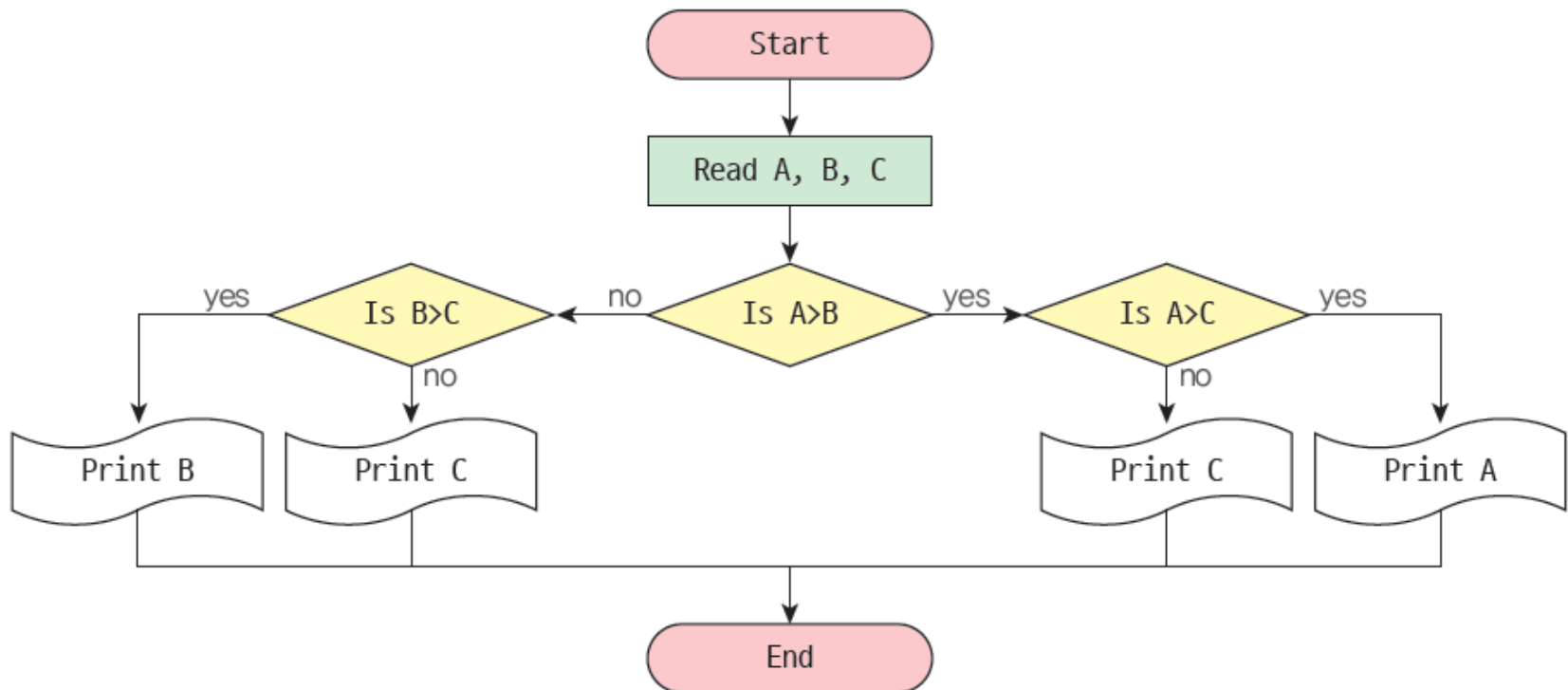
걸레질



# 3개의 수 중에서 최대값 찾기

- 이번에는 사용자로부터 받은 3개의 수 중에서 최대값을 찾는 알고리즘을 순서도로 작성해보자.
- 1. 사용자로부터 받은 3개의 수를 **A, B, C**라고 하자. 먼저 **A**와 **B**를 비교한다.
- 2. **A**가 **B**보다 크면 **A**와 **C**를 비교해서 큰 수를 출력하면 된다. 만약 **B**가 **A**보다 크다면 **B**와 **C**를 비교하여서 큰 수를 출력하면 된다.

# 알고리즘



# 도전 문제

- 3개의 수 중에서 최소값을 찾는 알고리즘을 생각할 수 있는가?



# 이번 장에서 학습할 내용



- 프로그래밍이란?
- 프로그래밍 언어
- C언어소개
- 알고리즘
- 스크래치

