

개정3판

Visual
Studio
2017

쉽게 풀어쓴



언어

EXPRESS

제1장 프로그래밍의 개념





이번 장에서 학습할 내용



- 프로그래밍 개념
- 프로그래밍 언어
- 알고리즘
- 스크래치

프로그램을
작성하기에
앞서서 중요한
개념들을
살펴봅니다..





프로그램이란?

- 컴퓨터 = 하드웨어 + 소프트웨어(프로그램)
- 컴퓨터를 범용적으로 만드는 것은 바로 프로그램



동영상 재생 프로그램



mp3 재생 프로그램



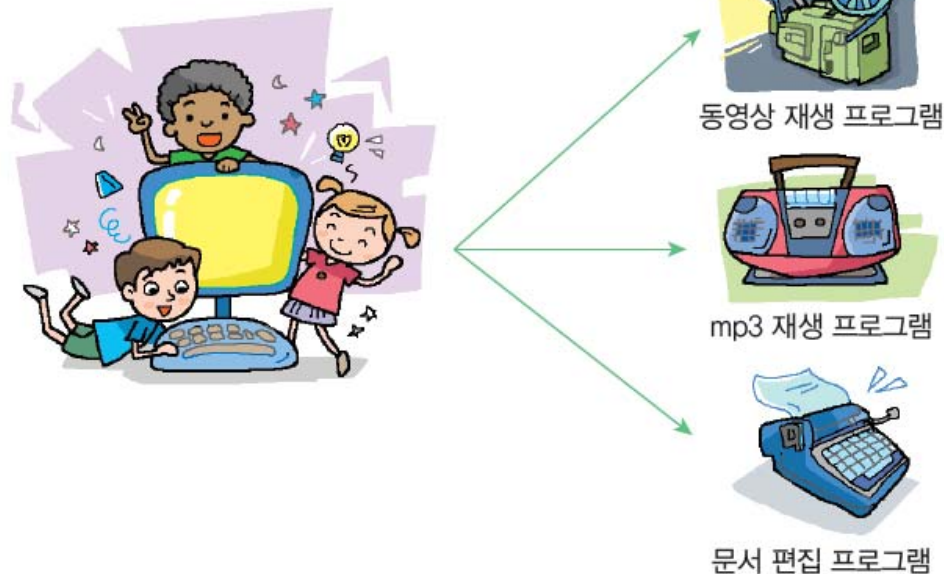
문서 편집 프로그램



프로그램

Q) 왜 컴퓨터에서는 가전제품처럼 프로그램 설치 없이 바로 동작되도록 하지 않고 불편하게 사용자가 프로그램을 설치하게 하였을까 ?

A) 컴퓨터를 범용적인 기계로 만들기 위해서이다.
컴퓨터는 :
있다.





계산기와 컴퓨터의 차이

계산기는 정해진 기능만을 수행한다. 기능을 변경할 수 없다.



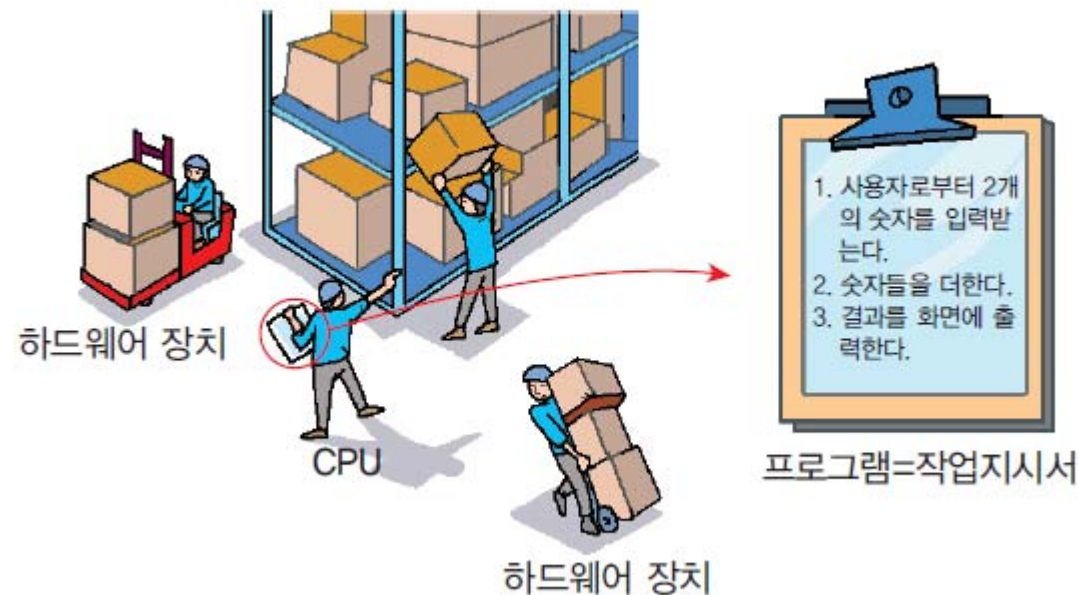
프로그램이라는 개념을 도입하여 수행하는 기능을 쉽게 변경할 수 있다.





컴퓨터의 정의

- 컴퓨터(computer)는 단순히 계산(compute)만 하는 기계가 아니다.
- 현대적인 의미에서의 컴퓨터는 프로그램(명령어들의 리스트)에 따라 데이터를 처리하는 기계라고 할 수 있다





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.





스마트폰도 컴퓨터의 일종

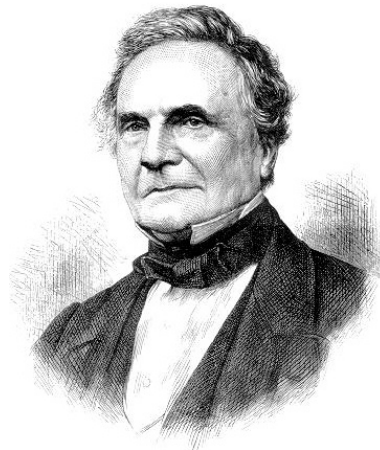
- **피쳐폰**: 미리 설정된 기능만 가능
- **스마트폰**: 애플리케이션만 변경하면 다양한 용도로 사용 가능





프로그램의 역사

- 프로그래밍이 가능한 최초의 기계: 해석 기관(Analytical Engine)
- 만든이: 찰스 배비지
- 수천 개의 기어, 바퀴, 축, 레버 등이 증기로 작동





최초의 프로그래머

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



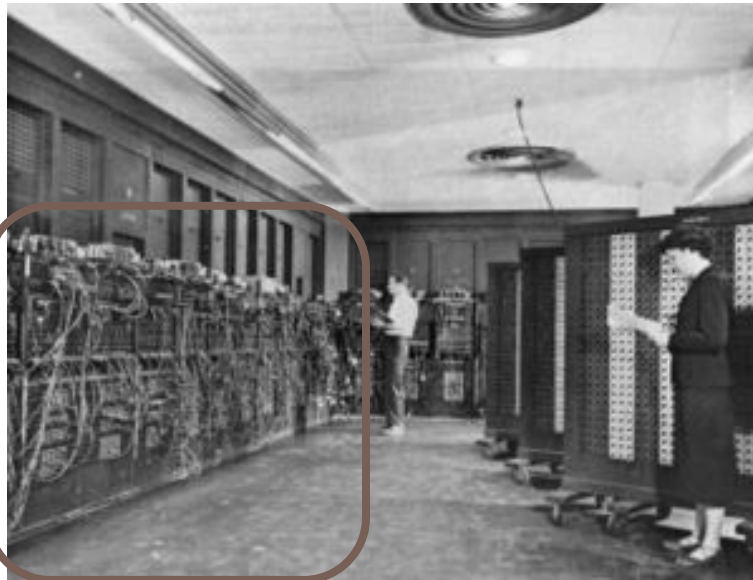


Number of Operation.	Nature of Operation.	Variables acted upon.	Variables receiving results.	Indication of change in the value on any Variable.	Statement of Results.	Data.										Working Variables.										Result Variables.			
						$1V_1$ ○ 0 0 1	$1V_2$ ○ 0 0 2	$1V_3$ ○ 0 0 4	$0V_4$ ○ 0 0 0	$0V_5$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



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 저장**된다.
- 메인 메모리에 저장된 프로그램에서 명령어들을 순차적으로 가져와서 실행한다.

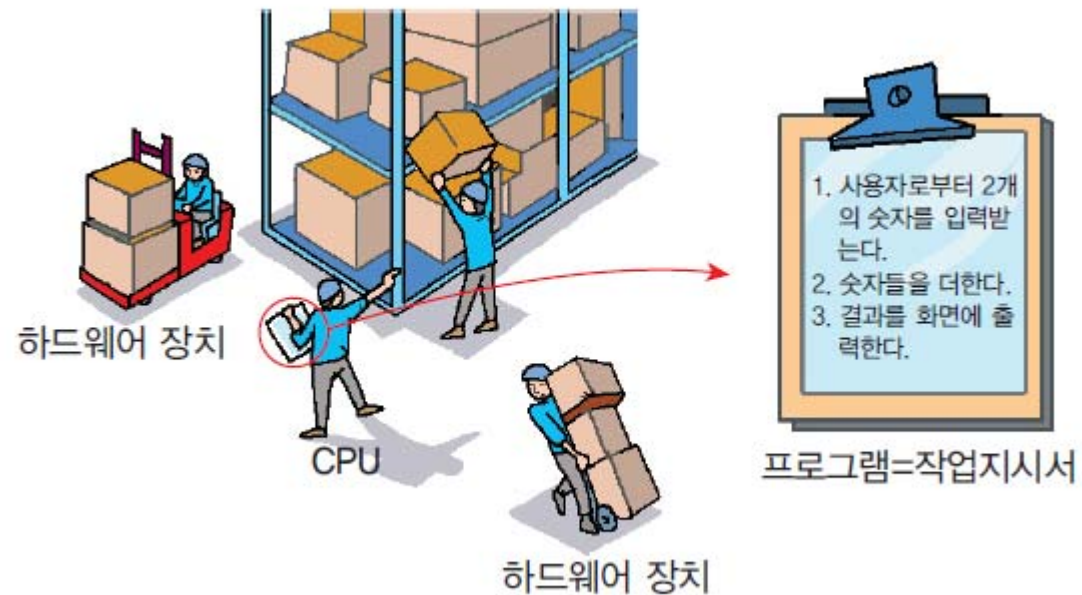


흠.. 프로그램을 쉽게 변경할 수 있게 하려면 데이터처럼 메인 메모리에 가져다 놓고 하나씩 실행하면 되겠군.



프로그램 == 작업지시서

- 프로그램: 컴퓨터에게 해야 할 작업의 내용을 알려주는 문서

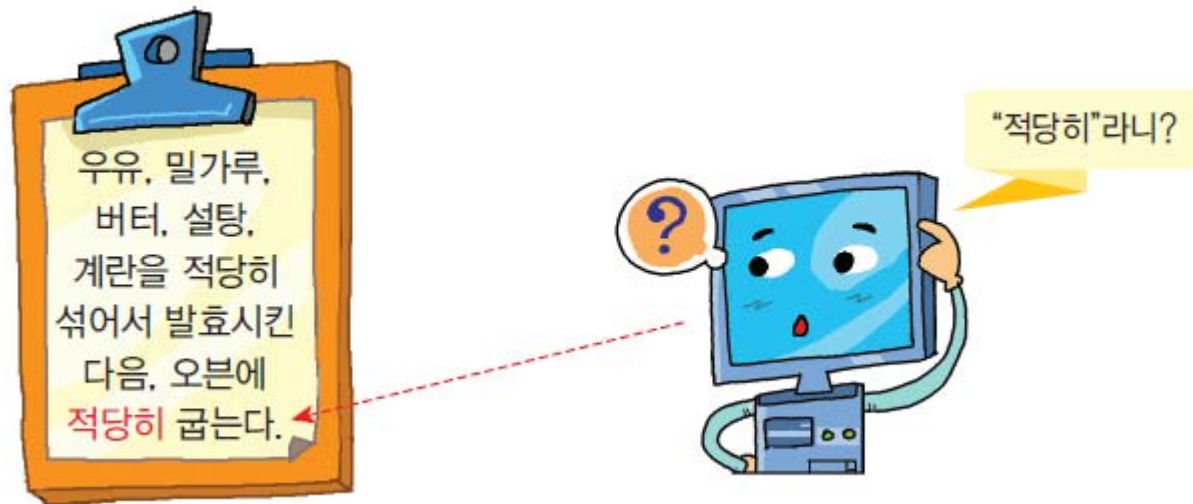


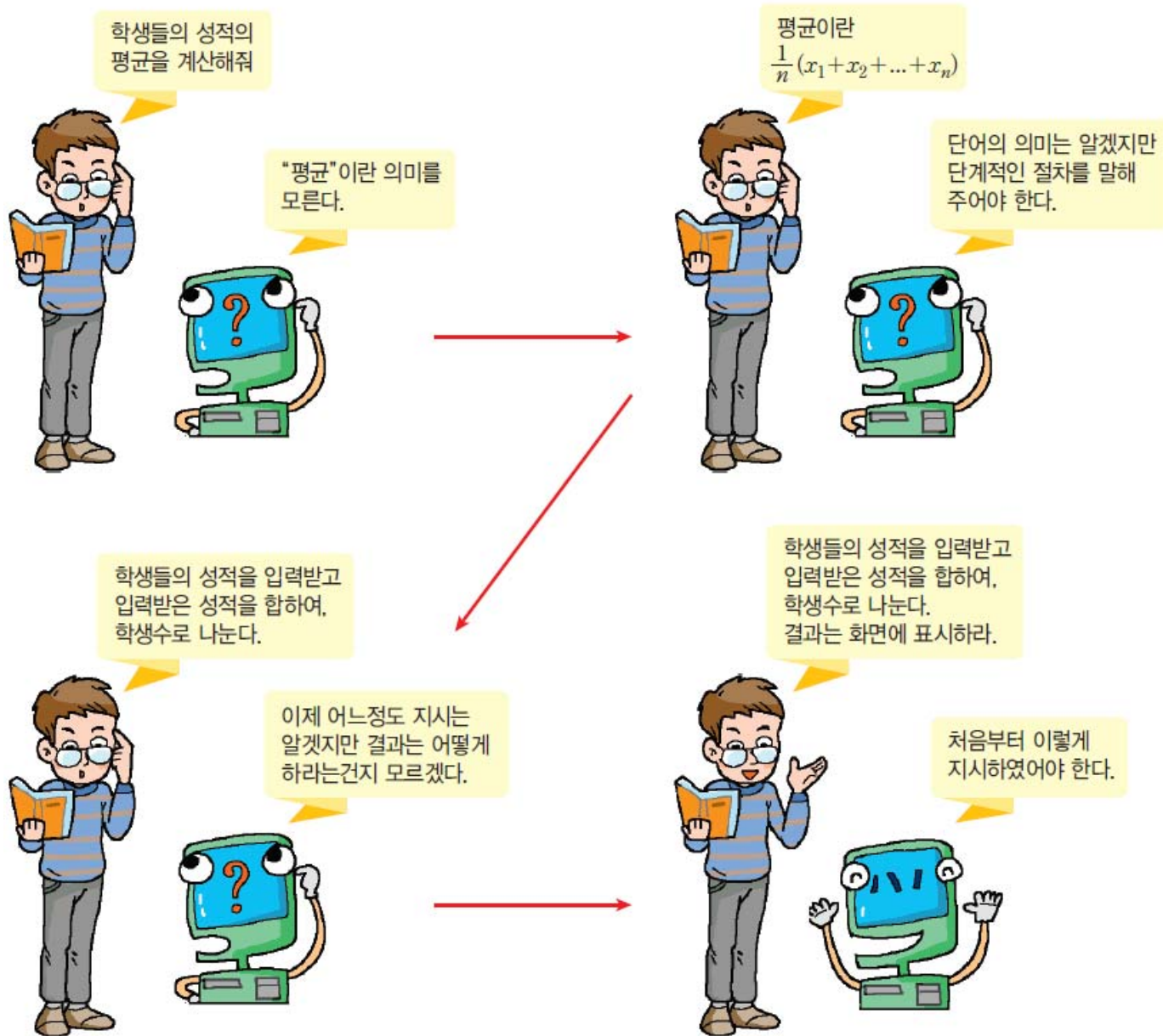


작업을 지시하는 방법

Q) 컴퓨터에게 어떻게 작업을 시킬 수 있을까?

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







이번 장에서 학습할 내용



- 프로그래밍 개념
- 프로그래밍 언어
- 알고리즘
- 스크래치

프로그램을 작성하기에 앞서서
중요한 개념들을 살펴봅니다 ..



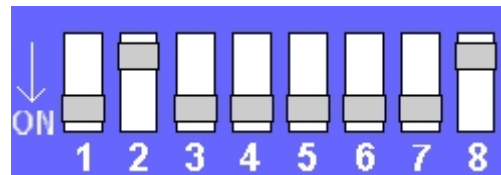


기계어

Q) 컴퓨터가 이해할 수 있는 언어는 어떤 것인가?

A) 컴퓨터가 알아듣는 언어는 한가지이다. 즉 0과 1로 구성되어 있는 “001101110001010...”과 같은 기계어이다.

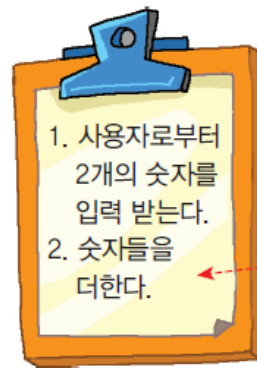
A) 컴퓨터는 모든 것을 0과 1로 표현하고 0과 1에 의하여 내부 스위치 회로들이 ON/OFF 상태로 변경되면서 작업을 한다.





기계어

- 컴퓨터는 기계어를 바로 이해할 수 있다.

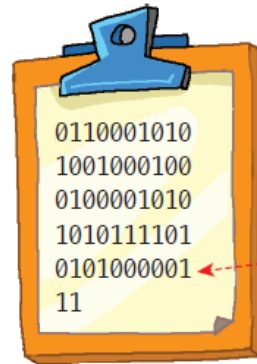


한글로 된 작업 지시서

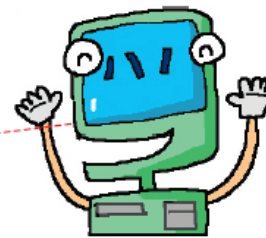


컴퓨터

한글은 너무 어려워.



기계어로 된 작업 지시서



컴퓨터

역시
이진수가 최고야.



기계어

□ 기계어의 예

Address:		0x401058															
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															

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:		



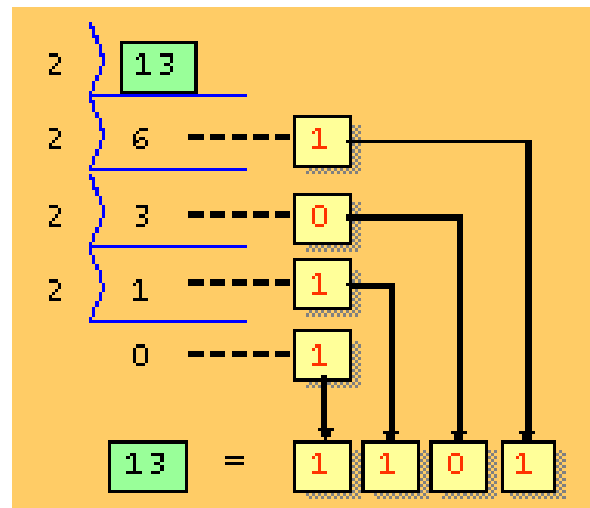
이진수

Q) 이진수는 십진수와 무엇이 다른가?

A) 이진수는 0과 1로만 구성되어 있다.

Q) 십진수를 이진수로 바꾸려면?

A) 십진수를 이진수로 바꾸려면 십진수를 2로 나누고 나머지를 기록하는 작업을 몫이 0이 될 때까지 되풀이하면 된다.





프로그래밍 언어의 필요성

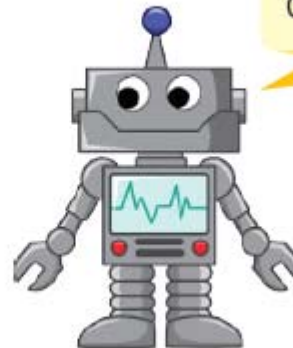
Q) 그렇다면 인간이 기계어를 사용하면 어떤가?

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

우리말 아니?



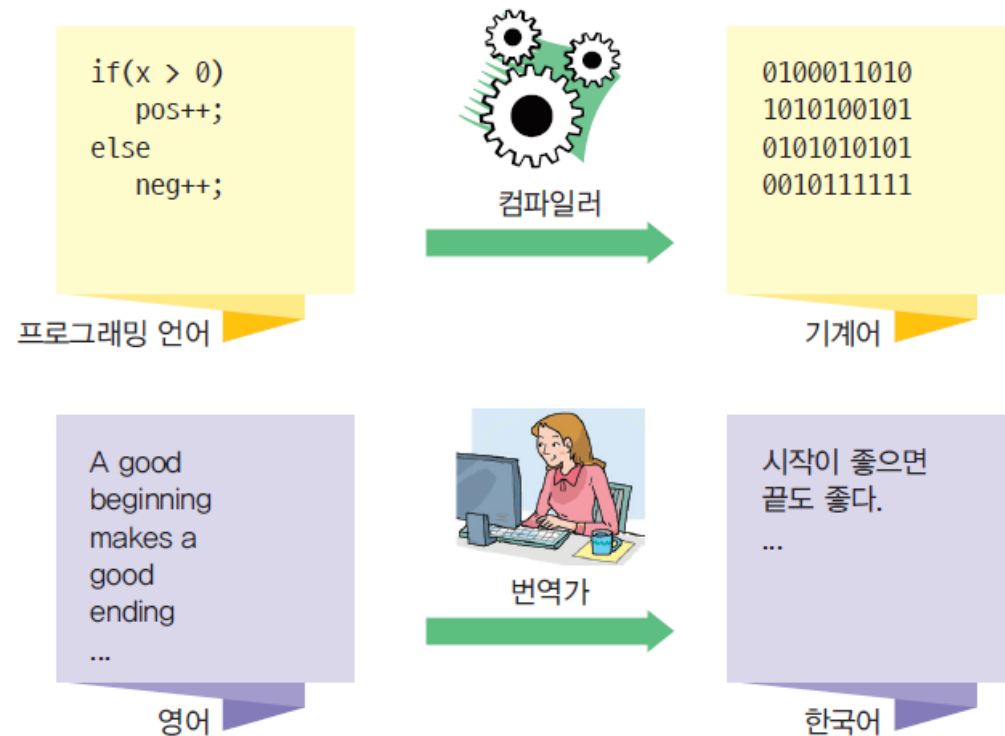
아니 기계어 밖에 몰라





컴파일러

- 컴파일러(compiler)는 인간과 컴퓨터 사이의 통역이라 할 수 있다.

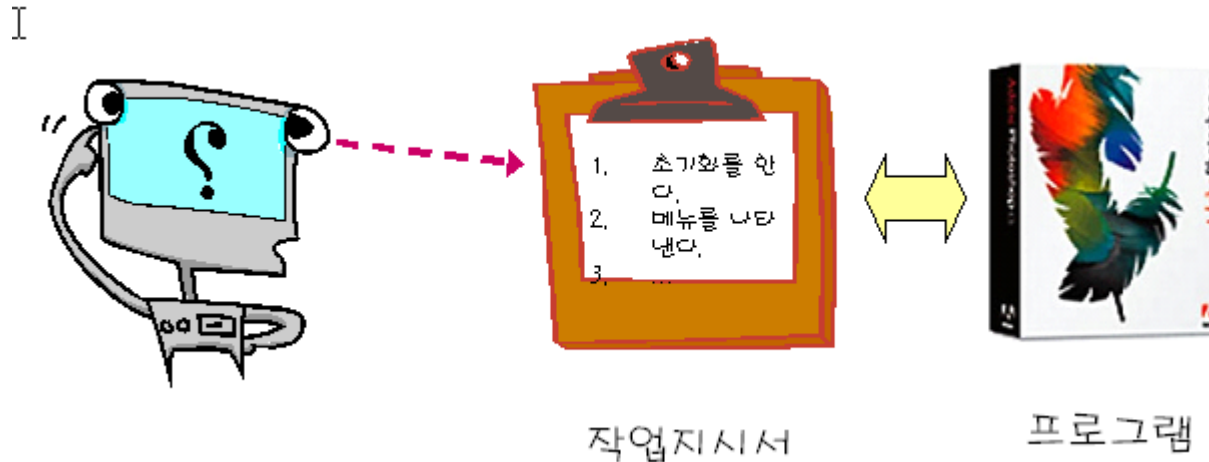




프로그램의 역할

Q) 컴퓨터에서 프로그램이 하는 일은 무엇인가?

A) 프로그램이란 우리가 하고자 하는 작업을 컴퓨터에게 전달하여 주는 역할을 한다.





장간 점검

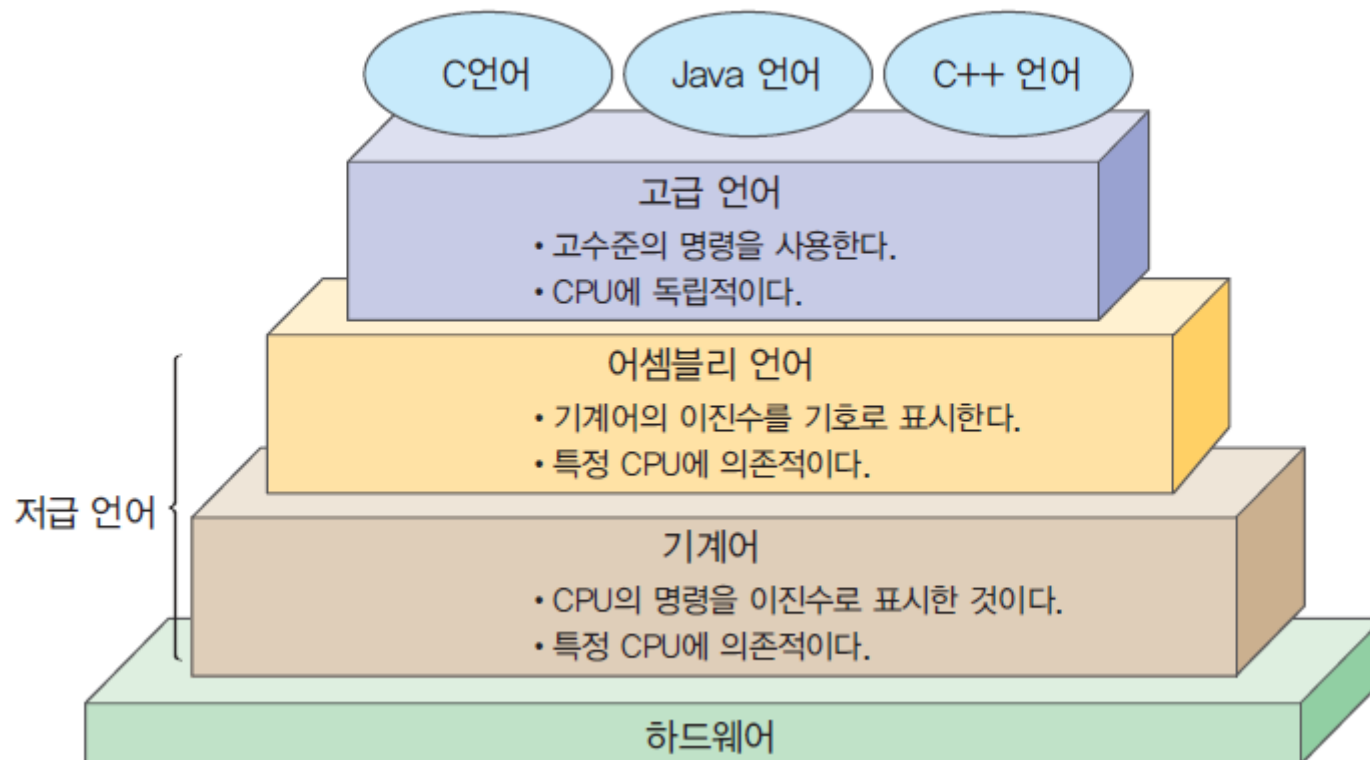
1. 왜 계산기는 컴퓨터라고 할 수 없는가?
2. 컴퓨터가 가장 쉽게 이해하는 언어는 무엇인가?
3. 컴파일러는 어떤 역할을 하는가?





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
- 기계어보다는 더 높은 수준에서 프로그램을 작성하는 것을 가능
- 기호와 CPU의 명령어가 일대일 대응
- 어셈블러(assembly): 기호를 이진수로 변환하는 프로그램

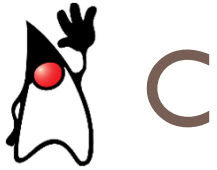
```
MOV AX, MIDSORE
MOV CX, FINALSORE
ADD AX CX
MOV TOTALSCORE, AX
```



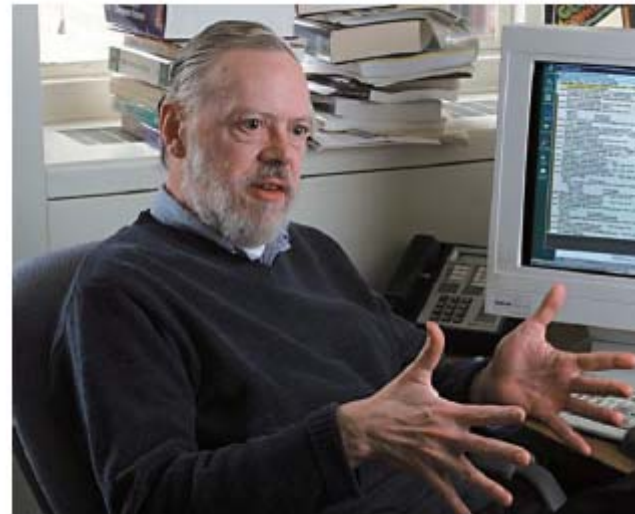
고급언어

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

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



- 1970년대 초 AT&T의 Dennis Ritchie 에 의하여 개발
- B언어->C언어
- 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

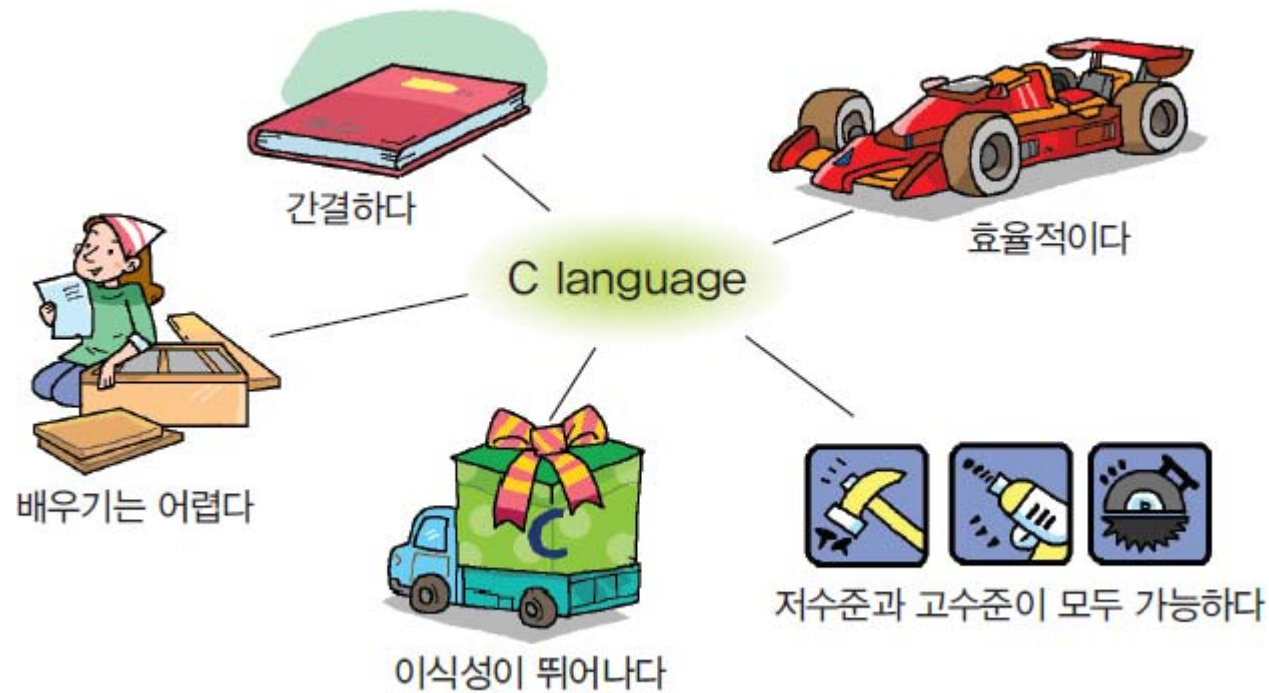


C언어의 특징

- 간결하다.
- 효율적이다.
- C 언어는 하드웨어를 직접 제어하는 하는 저수준의 프로그래밍도 가능하고 고수준의 프로그래밍도 가능하다.
- C언어는 이식성이 뛰어나다.
- 초보자가 배우기가 어렵다.



C언어의 특징





C언어의 미래

Q) 앞으로도 C언어는 사용될 것인가?

- C언어는 C++와 JAVA의 공통적인 부분이다.
- 임베디드 시스템에서는 C언어가 많이 사용된다.

임베디드 시스템: 임베디드 시스템이란 특수 목적의 시스템으로 컴퓨터가 장치 안에 MP3 플레이어, 핸드폰등이 여기에 속한다.



스마트폰도 CPU와 플래시 메모리 등이 들어가 있는 임베디드 시스템이다.



장점 점검

1. 임베디드 시스템이란 무엇인가?
2. C언어의 장점과 단점을 정리하여 보자.





이번 장에서 학습할 내용



- 프로그램의 이해
- 프로그래밍 언어
- **알고리즘**
- 스크래치

프로그램을
작성하기에
앞서서 중요한
개념들을
살펴봅니다..





알고리즘

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

A) 요리법을 알아야 한다.

- 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.

$$\left. \begin{array}{l} (0 + 10) = 10 \\ (1 + 9) = 10 \\ (2 + 8) = 10 \\ (3 + 7) = 10 \\ (4 + 6) = 10 \end{array} \right\}$$

5

$$10 * 5 = 50 \quad + \quad 5 \quad = \quad 55$$

- ③ 공식을 이용하여 계산할 수도 있다.

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



알고리즘의 기술

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



자연어

- 자연어 (natural language)는 인간이 사용하는 언어
- 단어들을 명백하게 정의해야 한다.

1. 리스트의 첫 번째 숫자가 가장 크다고 가정하자.
2. 리스트의 남아있는 숫자들이 하나씩 조사하여 현재의 최대값보다 크면 노트에 적는다.
3. 모든 숫자들을 전부 조사된 후에 노트에 가장 나중에 적힌 숫자가 최대값이 된다.



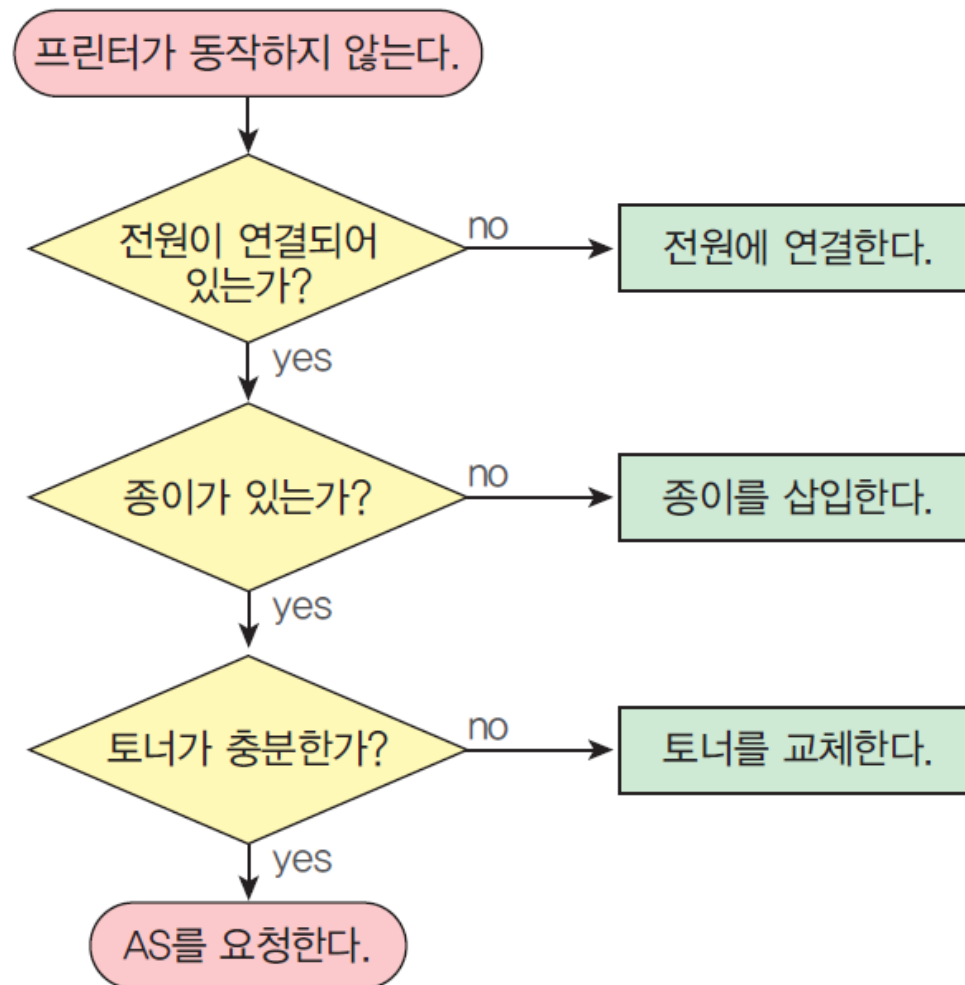
알고리즘의 기술

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





알고리즘의 예





의사 코드

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

알고리즘 **GetLargest**

입력: 숫자들의 리스트 **L**.

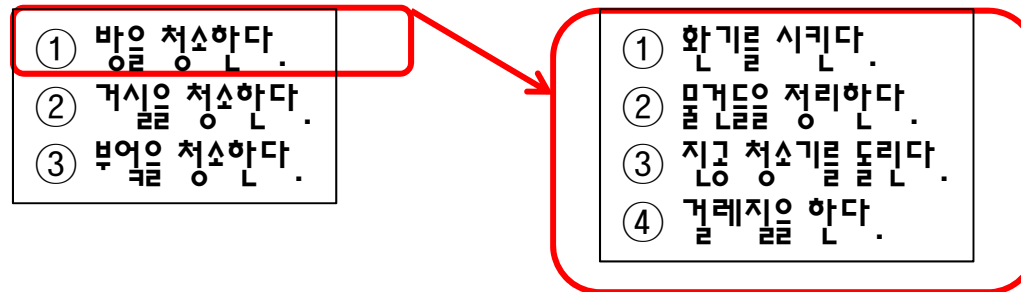
출력: 리스트에서 가장 큰 값

```
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.



환기



물건정리



진공청소기



걸레질



장간 점검

- 친구에게 전화를 거는 알고리즘을 만들어보자.
- 세탁기를 이용하여서 세탁을 하는 알고리즘을 만들어보자.
- 햄버거 가게에서 햄버거를 주문하는 알고리즘을 순서도로 작성해보자.





이번 장에서 학습할 내용



- 프로그래밍의 개념
- 프로그래밍 언어
- 알고리즘
- 스크래치

프로그램을
작성하기에
앞서서 중요한
개념들을
살펴봅니다.





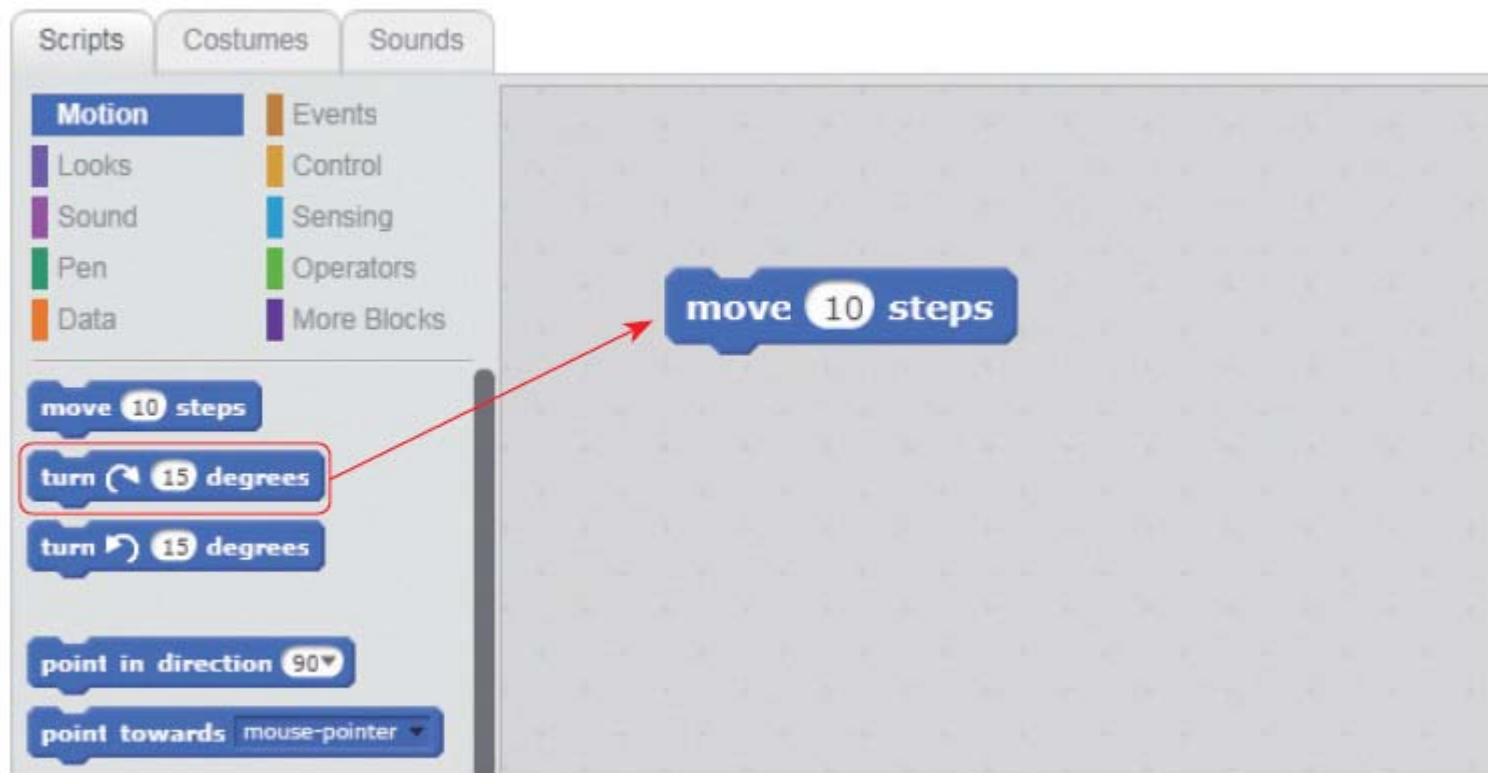
스크래치

- 스크래치(scratch): MIT에서 개발된 도구로서 프로그래밍 개념을 학습하기 위한 환경





고양이 댄스 예제 (1 / 4)





고양이 댄스 예제 (2/4)

① 클릭

Scripts | Costumes | Sounds

- Motion
- Looks
- Sound**
- Pen
- Data
- Events
- Control
- Sensing
- Operators
- More Blocks

play sound meow

play sound meow until done

stop all sounds

play drum 1▼ for 0.25 beats

rest for 0.25 beats

move 10 steps

play drum 1▼ for 0.25 beats

② 드래그



고양이 댄스 예제 (3/4)

The image shows the Scratch script editor interface. The 'Scripts' tab is selected, and the 'Motion' category is highlighted in the left sidebar. A sequence of three blocks is being assembled in the workspace:

- Block 1:** 'move 10 steps' (blue block).
- Block 2:** 'play drum 1 for 0.25 beats' (purple block).
- Block 3:** 'move -10 steps' (blue block).

Annotations in Korean provide instructions for editing the script:

- ① 클릭 (Click): Points to the 'move 10 steps' block in the sidebar.
- ② 드래그 (Drag): Points to the 'move -10 steps' block in the workspace, with a red arrow indicating the block is being moved from the sidebar.
- ③ 이동 거리를 -10으로 변경한다. (Change the movement distance to -10): Points to the '-10' value in the 'move -10 steps' block.



고양이 댄스 예제 (4/4)

Scripts Costumes Sounds

Motion Looks **Sound** Pen Data Events Control Sensing Operators More Blocks

play sound meow
play sound meow until done
stop all sounds
play drum 1 for 0.25 beats
rest for 0.25 beats

① 클릭

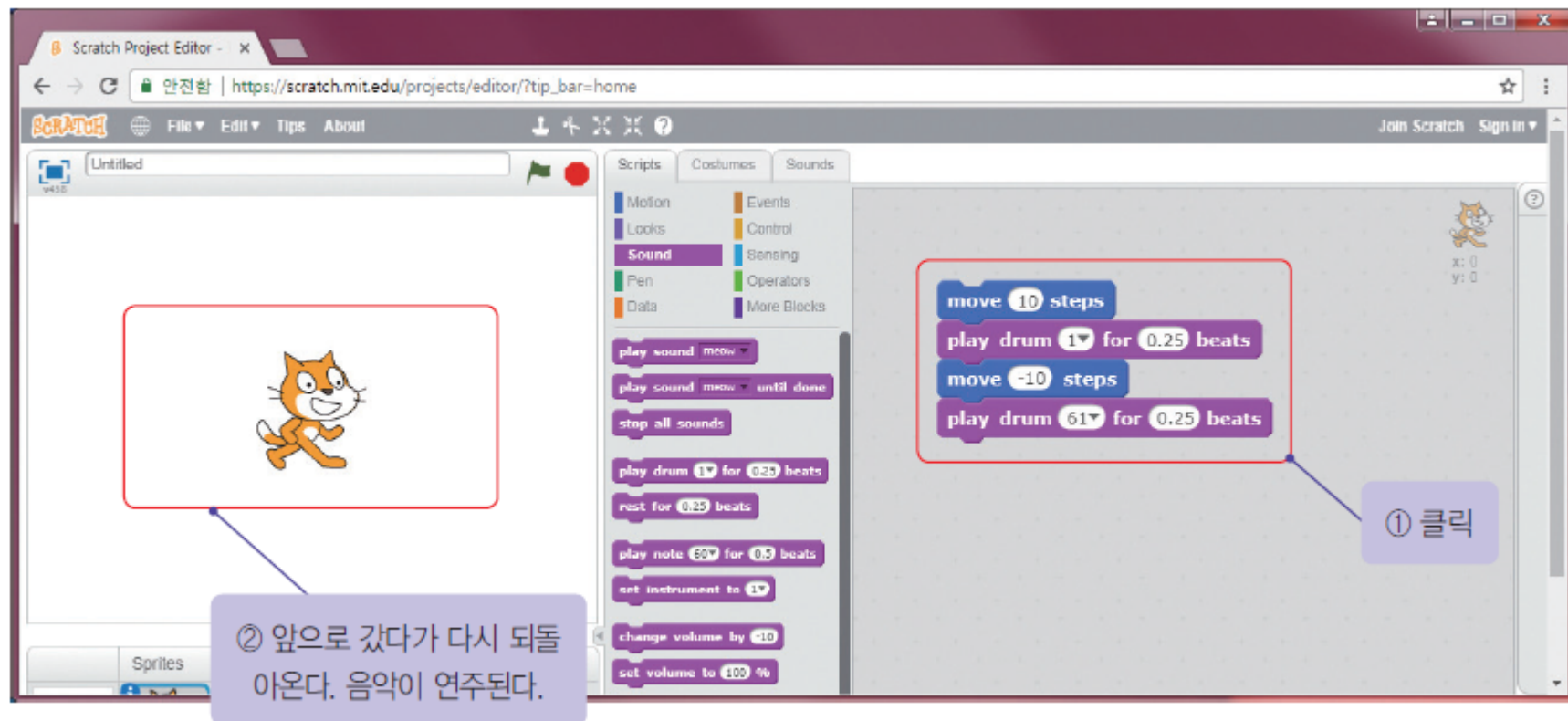
move 10 steps
play drum 1 for 0.25 beats
move -10 steps
play drum 61 for 0.25 beats

② 드래그

③ 드럼 번호를 61로 변경한다.

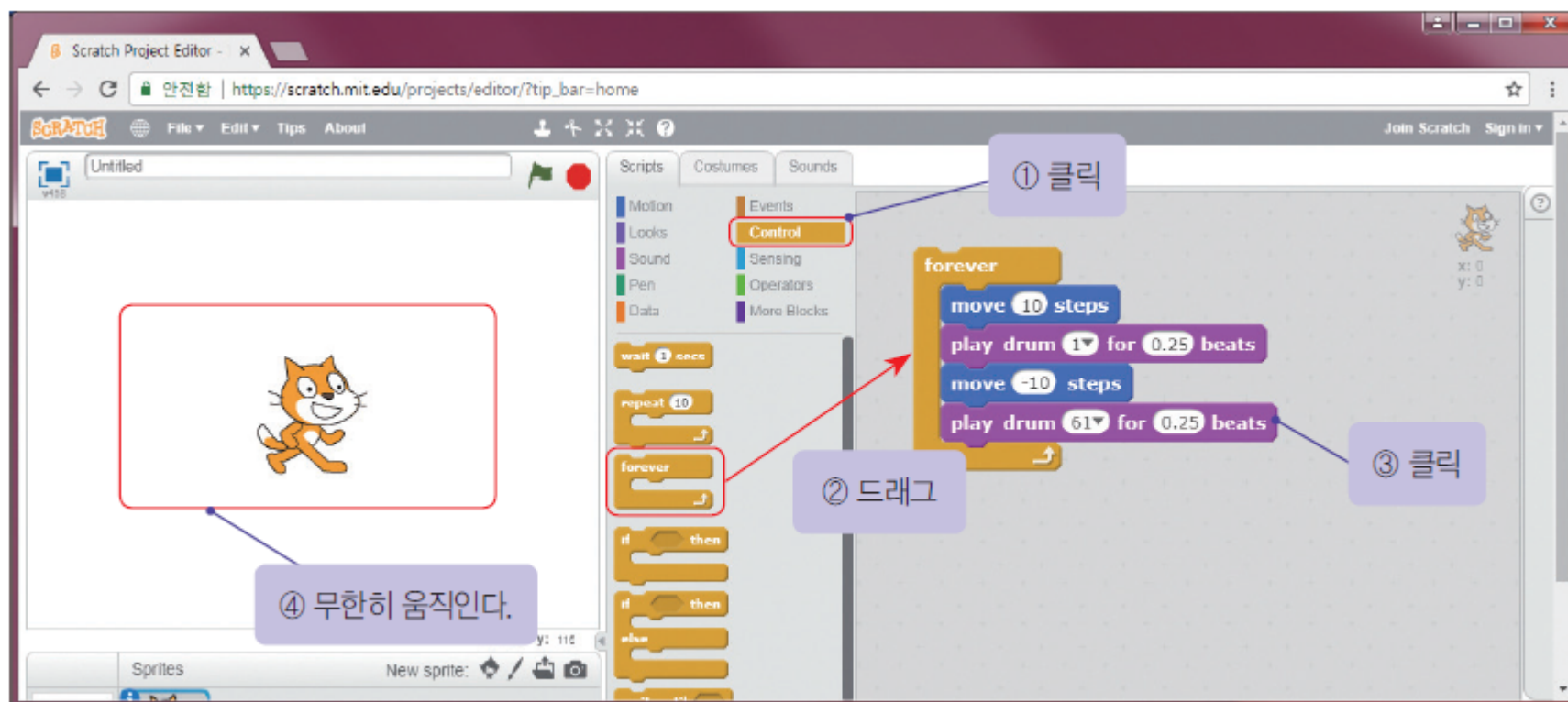


실행 결과



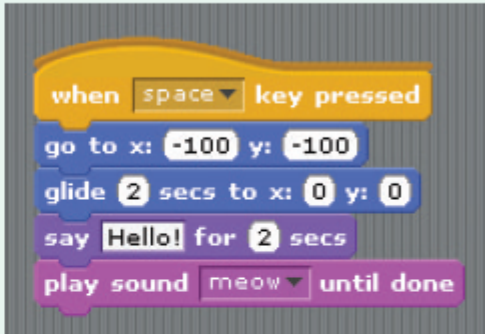


무한 반복

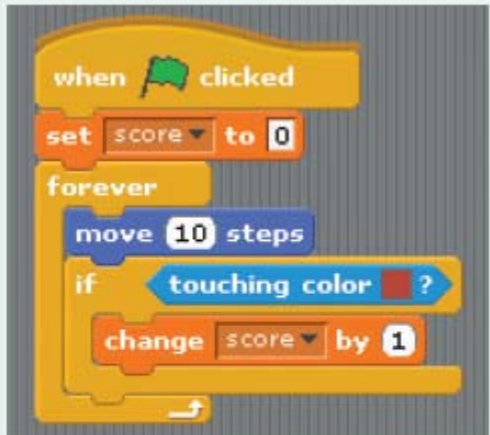




스크래치와 프로그래밍

개념	설명	예
순차실행(sequence)	서로 붙어 있는 블록들은 위에서 아래로 순차적으로 실행된다.	
반복실행(iteration)	forever와 repeat 블록은 블록들의 반복 실행을 위하여 사용될 수 있다.	



조건부 실행	if와 if-else 블록은 조건을 검사하는데 사용된다.	
변수(variable)	variable 블록은 변수를 생성하고 프로그램에서 사용할 수 있게 한다. 변수는 숫자나 문자열을 저장하는 공간이다. 변수는 <code>Make a variable</code> 블록으로 생성할 수 있다.	



장간 점검

- 고양이를 무한히 회전시키는 스크립트를 작성하고 실행하여 보자.
- 표 1-2에 나오는 스크립트들을 작성하여서 실행하여 보고 어떤 동작을 하는지를 조사하여 보자. 필요하다면 변수도 생성한다.



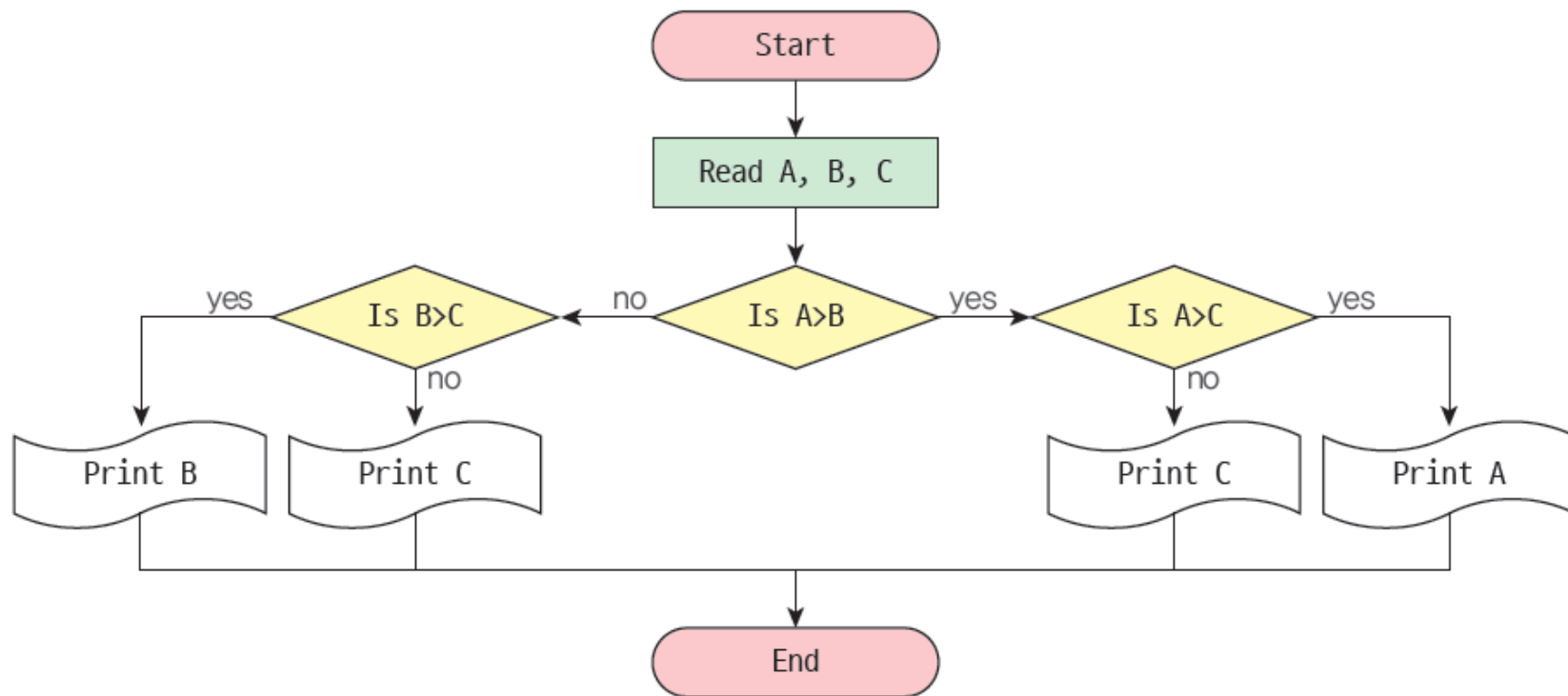


mini project: 3수 중에서 최대값 찾기

- 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

- 3수 중에서 최소값을 찾는 알고리즘을 생각할 수 있는가?





Q & A

