

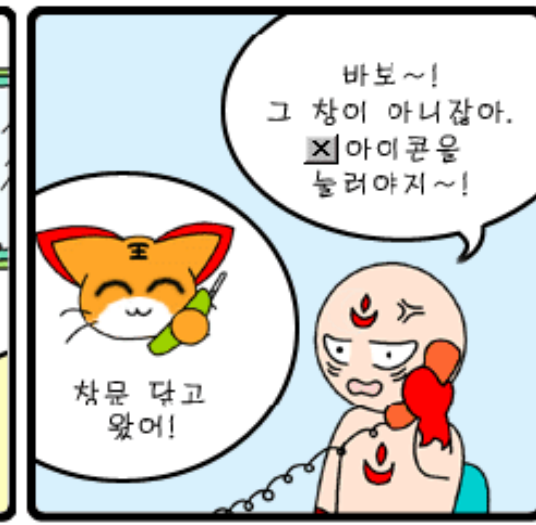
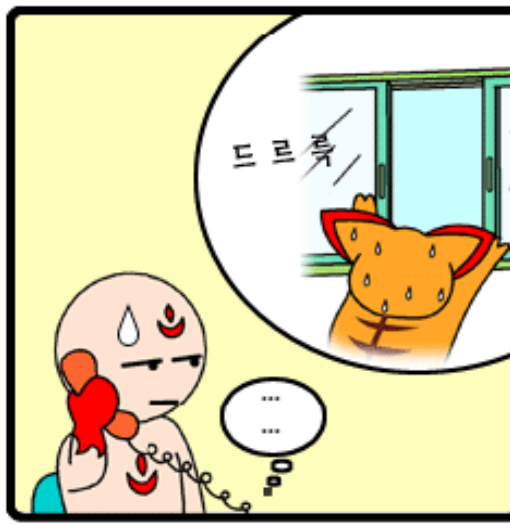
Programming 기초

경북대학교
소프트웨어융합과
배희호 교수





지식 ? 무식 = 지식





자신의 두뇌를 정복하는 방법

- 1 Slow down. 너무 많은 것을 공부하면 기억할 내용은 그만큼 적다.
- 2 연습문제 연습문제를 꼭 풀어 본다. 간단하게 메모한다.
- 3 질문하라 바보 같은 질문은 없다.
- 4 움직여라 한 자리에만 앉아서 공부하지 말고 움직여 가면서 공부하라.
- 5 잠자기 전 잠자기 전 한번 더 읽어 보라. 기억에 남게 된다.
- 6 물을 마셔라 두뇌가 잘 돌아 가려면 많은 수분이 필요
- 7 큰 소리로 크게 소리내면 기억에 도움. 남을 가르치는 건 더 효과적.
- 8 잠시 쉬기도 무조건 파고 드는 게 효과적인 건 아니다
- 9 Feel something! 몰입, 사진에 제목 달기, 그림으로 표현, 표로 줄이기 등
- 10 직접 실행 코드를 직접 타이핑하고 실행시켜 본다. 결과가 나오지 않아도 그것이 큰 경험이 된다.



TIOBE Index

- TIOBE 프로그래밍 커뮤니티 인덱스(TIOBE Programming Community index)
- Programming 언어의 인기도(popularity) 지표 : 매월 조사
 - 전세계의 숙련된 엔지니어 수, 교육 과정 또는 타사 공급 업체의 수를 기준으로 측정
- 사용 검색 엔진 : Google, Bing, Yahoo!, Wikipedia, Amazon, YouTube, Baidu
- 지표의 활용 예시
 - 개인의 Programming 기술의 변경/보완 필요성 여부 판단
 - 새 Software System을 구축할 때 채택할 Programming 언어 판단
- 비교
 - 지표값이 최고의 프로그래밍 언어를 의미하지는 않음
 - 지표값이 가장 많이 사용된 코딩에 사용된 프로그래밍 언어를 의미하지는 않음

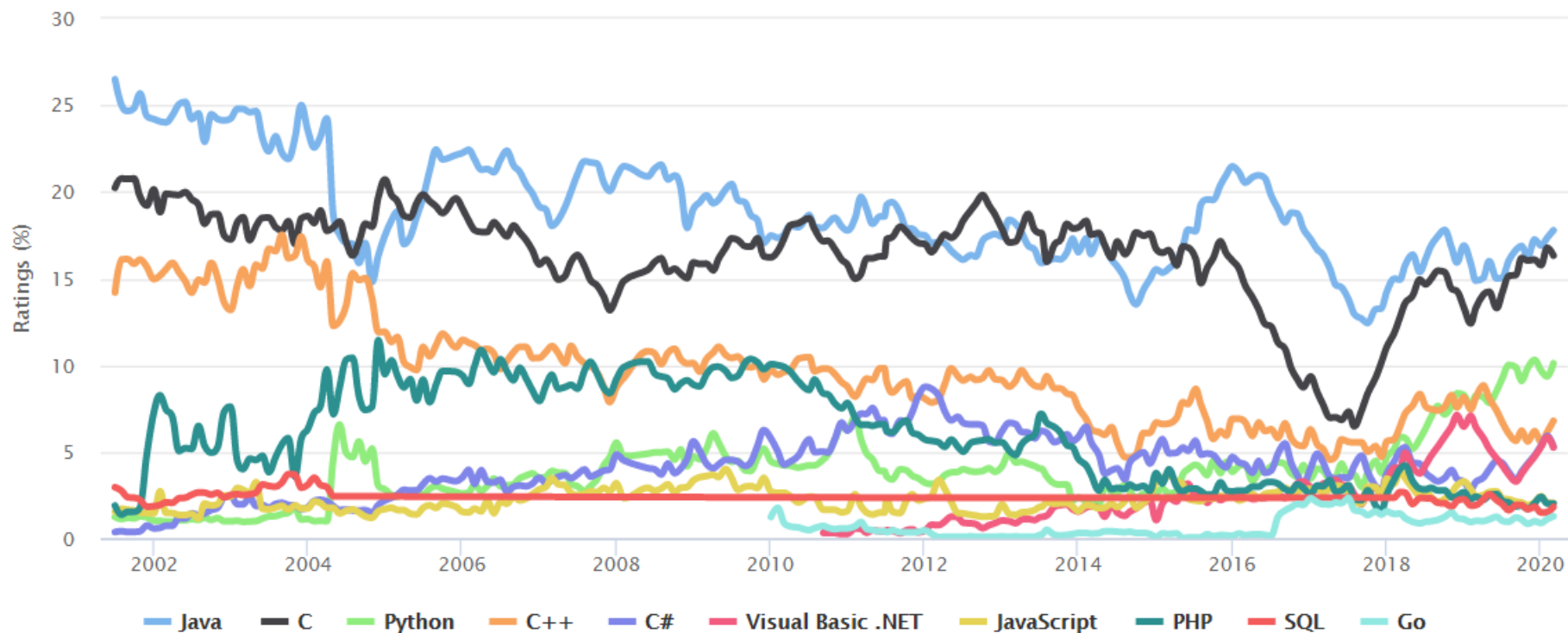


TIOBE Index













TIOBE Programming Community Index

Source: www.tiobe.com





TIOBE Index

Aug 2021	Aug 2020	Change	Programming Language		Ratings	Change
1	1			C	12.57%	-4.41%
2	3	▲		Python	11.86%	+2.17%
3	2	▼		Java	10.43%	-4.00%
4	4			C++	7.36%	+0.52%
5	5			C#	5.14%	+0.46%
6	6			Visual Basic	4.67%	+0.01%
7	7			JavaScript	2.95%	+0.07%
8	9	▲		PHP	2.19%	-0.05%
9	14	▲▲		Assembly language	2.03%	+0.99%
10	10			SQL	1.47%	+0.02%



Futuristic Innovator


京福大学校
KYUNGBOK UNIVERSITY



- Web에서 사용되는 다양한 유형의 프로그래밍 언어 및 기술에 대한 정보를 제공
- Server측 프로그래밍 언어, Client측 프로그래밍 언어와 같이 좀더 상세한 주제로 나누어 순위 정보를 제공함
- 자신이 관심 있는 분야의 트렌드를 좀더 쉽게 알아 볼 수 있다는 장점이 있음



W3Techs
Web Technology Surveys

**Free IP API** One-stop API with 45 unique IP address data points **Sign up Now!**

provided by **Q-Success**


Ad by EthicalAds


advertise here


Home Technologies Reports Sites Quality Users Blog Forum FAQ Search

advertise here

Featured products and services

 Learn to build a website, step-by-step - **WebsiteSetup**

 WordPress Tutorials & Themes - **Themeisle**

 Web Design Tips, Tutorials, and Guides - **DesignBombs**

W3Techs - World Wide Web Technology Surveys

W3Techs provides information about the usage of various types of technologies on the web.

Our **vision**

Provide the most reliable and most extensive source of information on web technology usage.

In our [web technology surveys](#) you can see the most popular technologies in these categories.

Content Management Systems

Most popular content management systems

	usage	change since 1 July 2021	market share	change since 1 July 2021
© W3Techs.com				
1. WordPress	42.4%	+0.4%	65.2%	+0.2%
2. Shopify	3.8%	+0.1%	5.8%	+0.1%

News [read all news](#)

Nginx is now the most popular web server, overtaking Apache
4 May 2021
For the first time since we started our surveys in 2009, Apache has lost the first spot as the most popular web server to Nginx.
[» more](#)

40% of the web uses WordPress
10 February 2021
The incredible success story of WordPress continues by reaching another milestone: 2 out of every 5 websites use it now.
[» more](#)

Web Technologies of the Year 2020
4 January 2021
We compiled the list of web technologies that saw the largest increase in usage in 2020.
[» more](#)



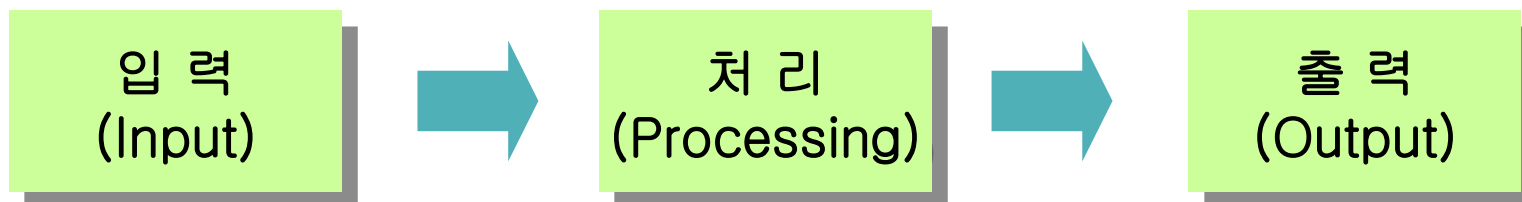
Computer System



■ System

- 여러 요소들이 상호 유기적으로 구성되어 하나의 특정한 목표를 달성하는 복합체

■ Computer System



- Computer System은 Program에 의해 Dada를 입력 받아 처리, 저장, 검색 등의 과정을 거친 후 요구되는 정보를 출력해 내는 기계
- 기계로서의 전자 장치와 처리의 주체가 되는 Program으로 구성
- 하드웨어(H/W)와 소프트웨어(S/W)로 구성



Computer System



- Computer의 구성 요소를 크게 2가지로 분류하면?
 - Computer는 기본적으로 Hardware와 Software로 구분



하드웨어



소프트웨어



Computer System



- Hardware와 Software의 분리
 - 최초의 Computer(ENIAC)에서는 분리되지 않았음



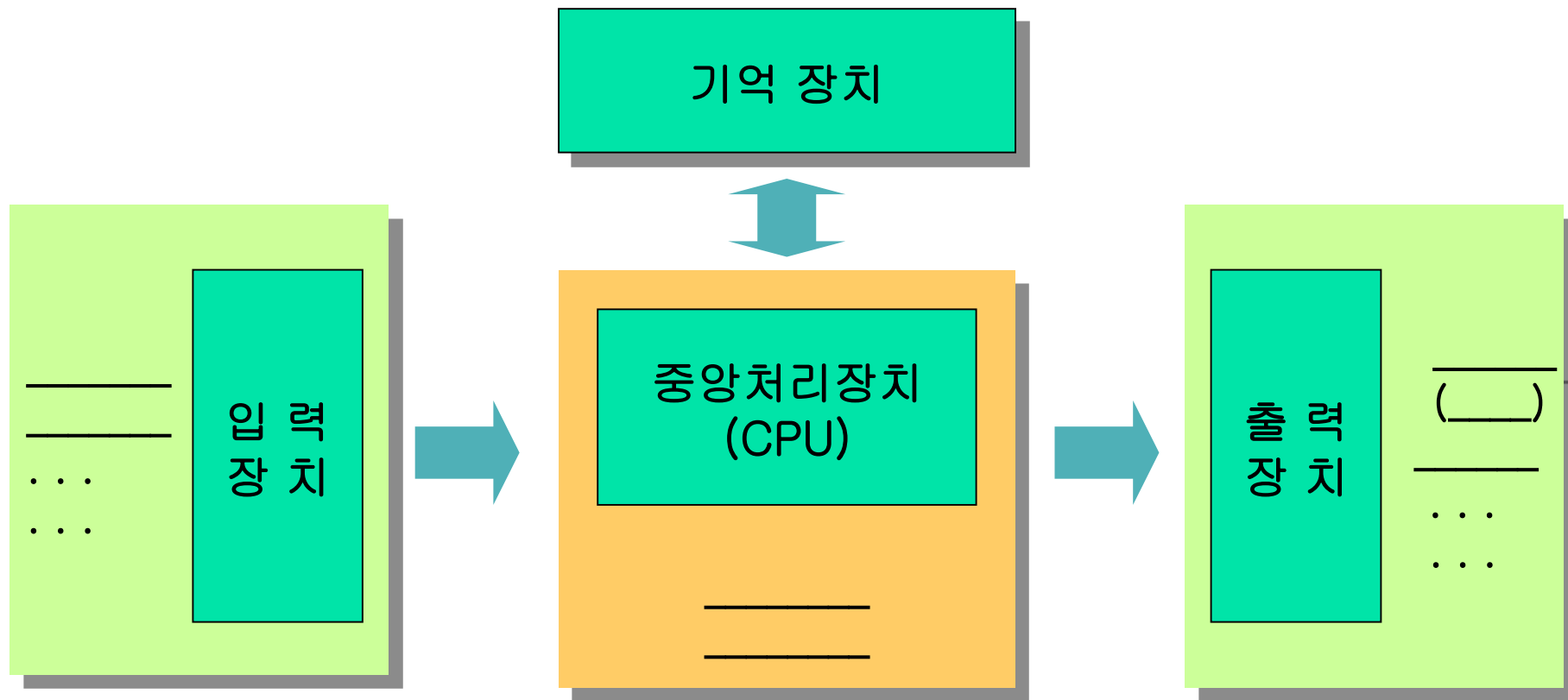
- 1950년대에 폰 노이만이 제시한 Idea
 - Program을 Memory에 저장
 - Memory에서 Program의 문장을 하나씩 꺼내와 실행



Computer System



■ Hardware





Computer System



■ Software

- 컴퓨터 하드웨어 시스템을 가동시켜 적절한 작업을 수행하도록 하는 일체의 프로그램
- 컴퓨터 언어 중 하나로 작성

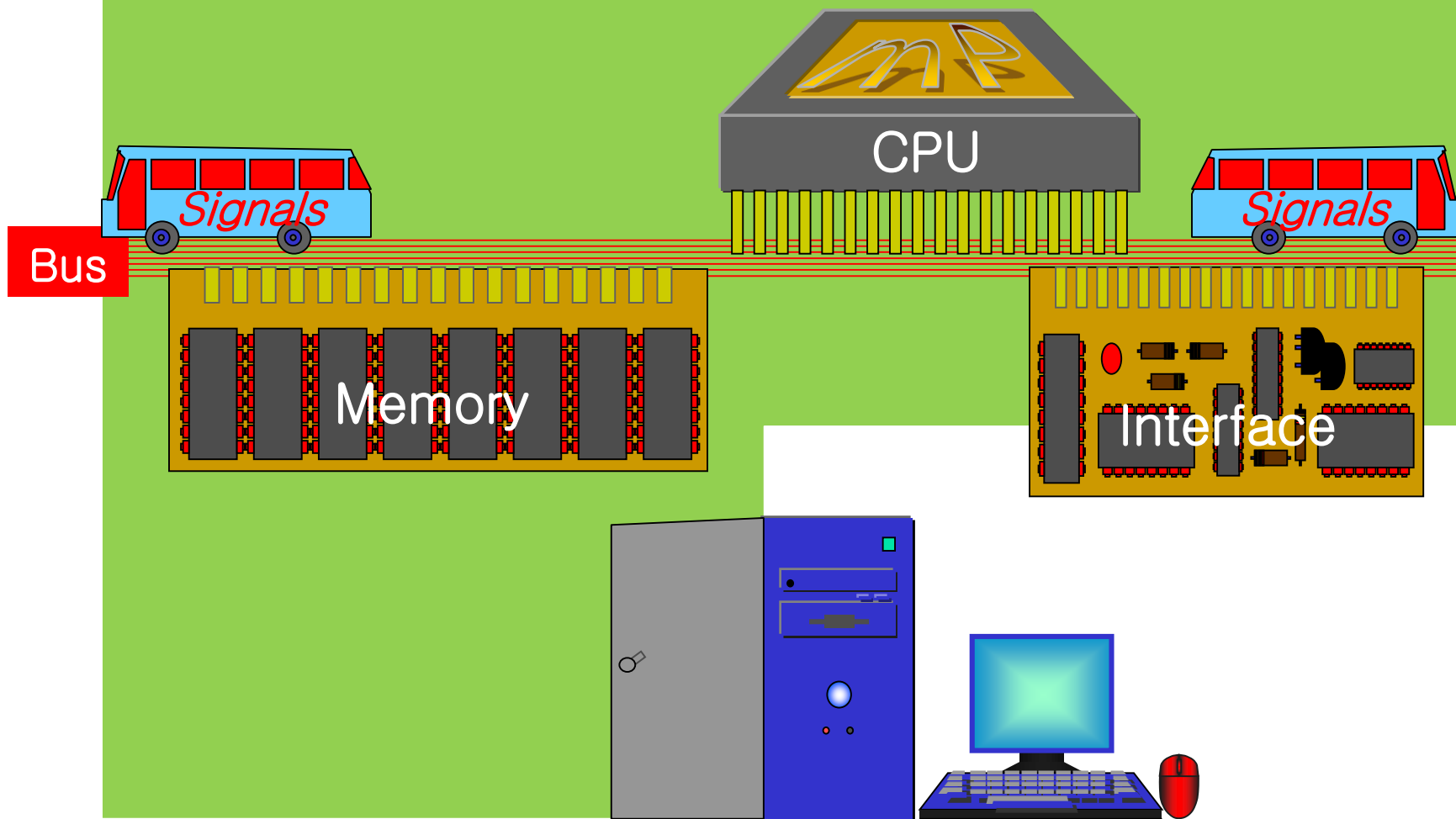
시스템 프로그램	구분	응용 프로그램
컴퓨터 생산자가 공급하는 H/W 운영에 필요한 S/W	의미	사용자의 특정 목적에 맞추어 작성된 프로그램
사용자와 H/W 자원 활용 연결	용도	과학 계산, 기업 업무 적용, 게임 등
운영체제(OS), 언어처리시스템(컴파일러, 인터프리터), 데이터베이스시스템(DBMS)	종류	패키지, 워드 프로세서, 스프레드 시트(Microsoft Excel), 프리젠테이션(Powerpoint) 등



Computer System



Mother Board



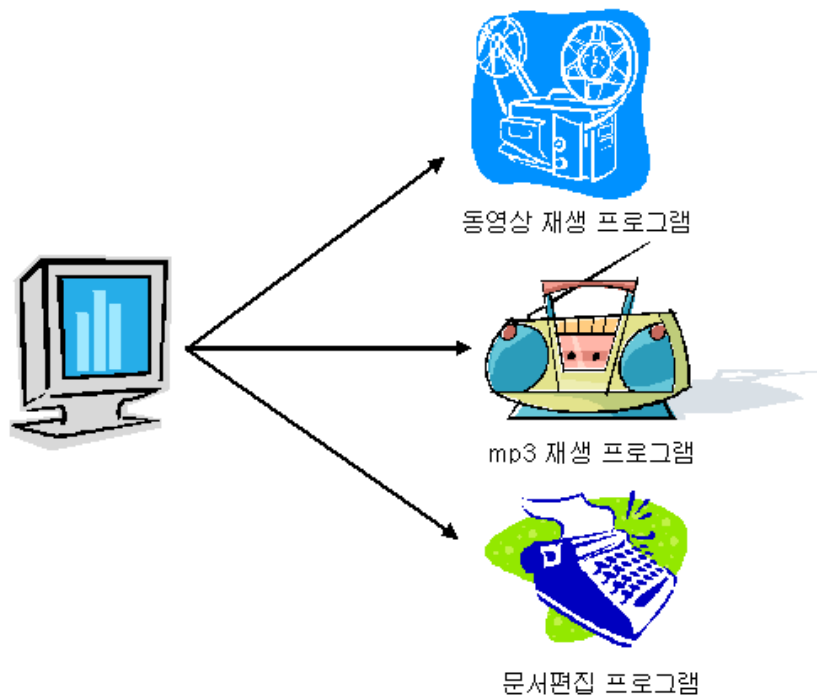
Futuristic Innovator

京福大
KYUNGBOK UNIVERSITY



Computer System

- Computer의 가장 큰 장점은 무엇일까?
 - Computer는 범용적인 기계이다
 - Program만 바꿔주면 다양한 작업이 가능하다





Software 분류



사용자1

사용자2

사용자3



사용자n

응용 소프트웨어

시스템 소프트웨어

컴파일러

어셈블러

문서편집기

라이브러리

프로세스관리
기억장치관리

운영체제

입출력관리
파일시스템관리

기억장치

처리장치

입출력장치

파일시스템

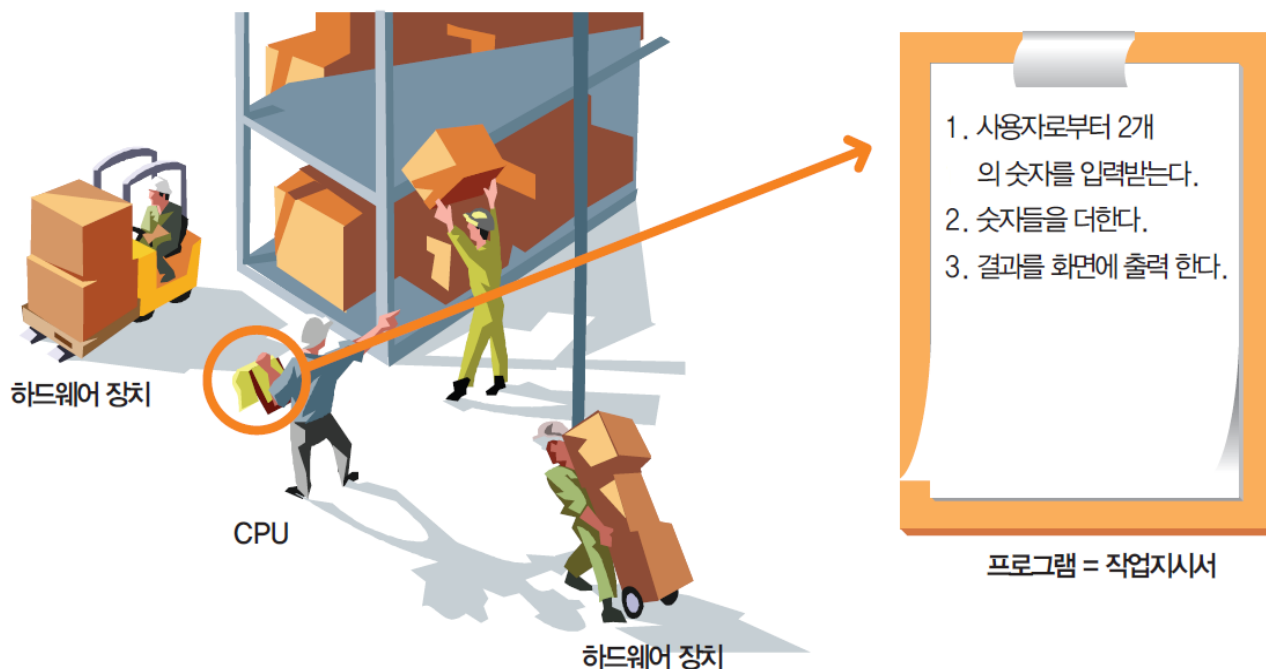
하드웨어





Program이란 ?

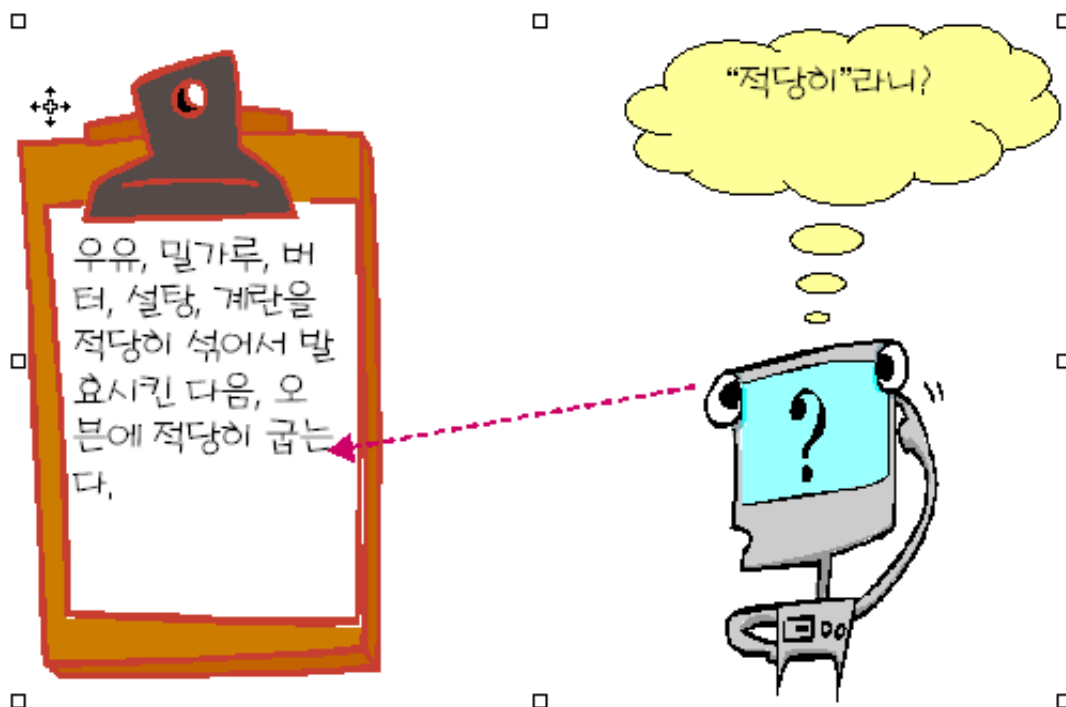
- Computer에게 무엇을 어떻게 시킬 지를 기록해 놓은 문서가 Program이다
- Program의 각 문장은 Computer에게 작업을 지시하는 명령 (instruction)으로 되어 있다





Program이란 ?

- Computer에게 적당히 작업을 시킬 수 있을까?
- 상식이나 지능이 없기 때문에 아주 자세하고 구체적으로 일을 지시하여야 한다

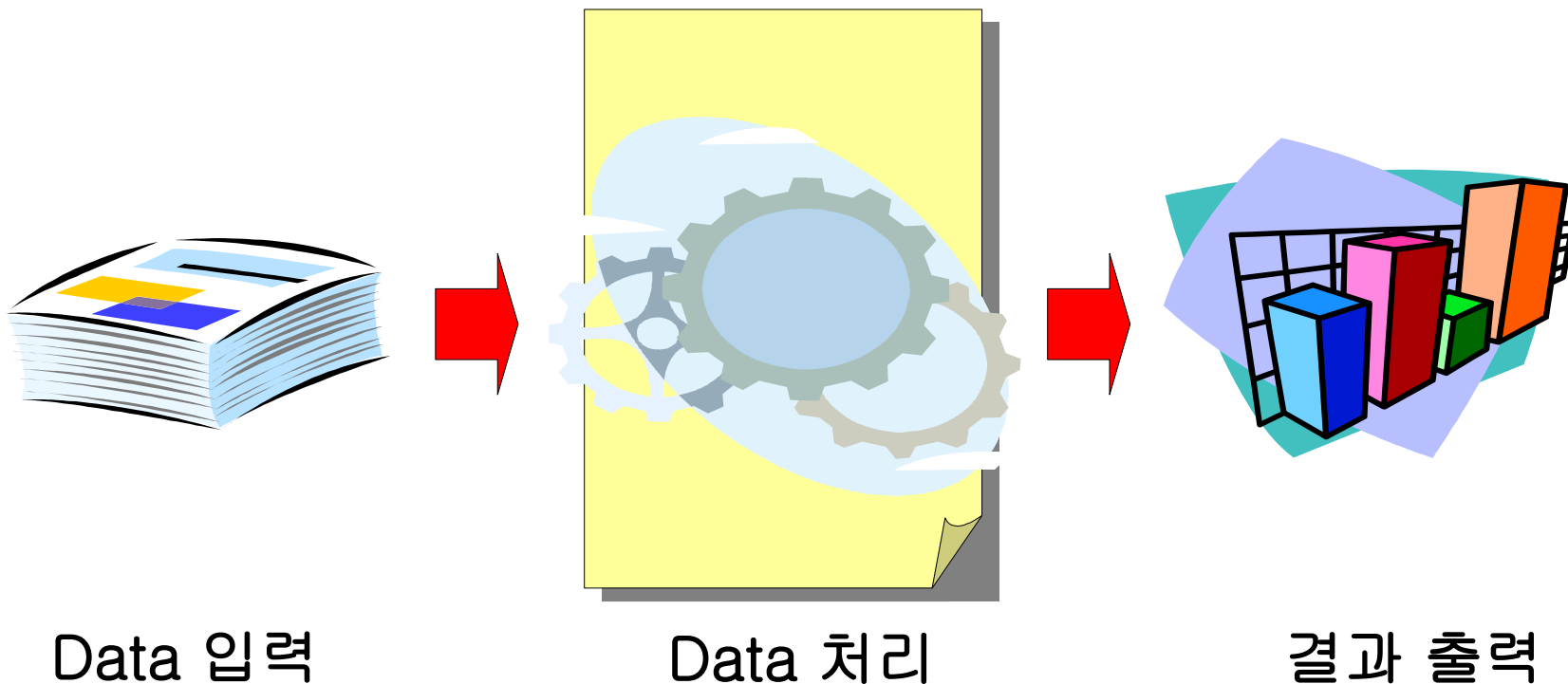




Program이란 ?

- 일반적인 Program의 형태

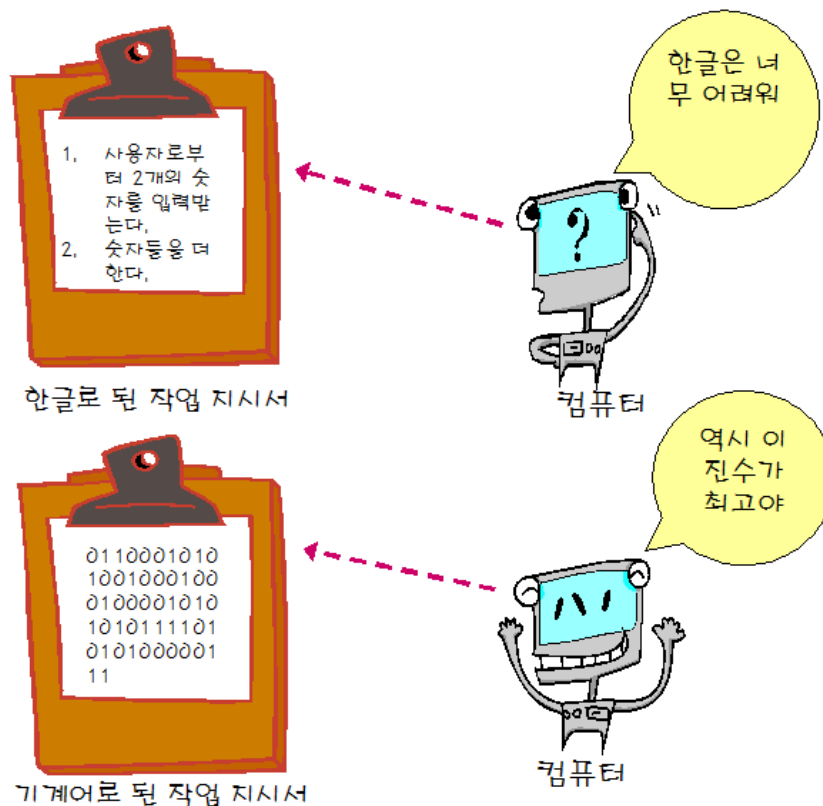
- Data를 받아서(입력단계), Data를 처리한 후에(처리단계), 결과를 화면에 출력(출력단계)한다





Program이란 ?

- Computer는 인간의 언어를 이해할 수 없다
- Computer는 이진수로 된 Machine Language만을 이해한다





Program이란 ?



프로그램
(Program)

프로그래머

프로그램 언어

```
Import java.awt.*;  
public void main(  
    String agrs[]){  
    public void paint(){  
        g.graphics();  
    }  
}
```

프로그래밍



Program이란 ?



- Program

- Computer를 이용하여 Data 처리를 할 수 있게 하기 위하여 Computer가 인식할 수 있는 Computer 명령어를 논리적 순서에 맞게 모아놓은 것

- Programming

- Program을 개발하는 작업

- Programmer

- Programming 작업을 수행하는 사람

- Programming Language



Program이란 ?



- Computer에 의한 문제 해결 : Program
- Program
 - 주어진 문제를 푸는 과정을 Computer의 명령어 세트를 이용하여 기술
- Program 작성
 - Algorithm 작성 단계
 - 문제를 푸는 방법 기술(기계 독립적)
 - Program 작성 단계
 - 기술한 Algorithm을 Computer 명령어 세트를 이용하여 Computer에서 수행할 수 있는 형태로 기술(기계 종속적)



Program이란 ?



음악	컴퓨터
악기(피아노)	
악보	
악보의 기호(음표)	
작곡가	
작곡	
연주자	
관객	



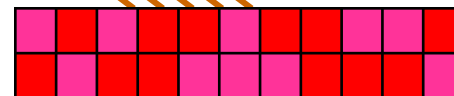
Programming이란?



■ Hard Wired 방식

1234

각 비트의 조합으로 기능을
제어



Futuristic Innovator

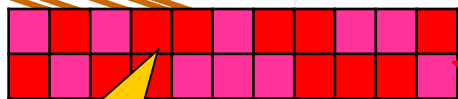
京福大學校
KYUNGBOK UNIVERSITY



Programming이란?

■ Software 방식

1234



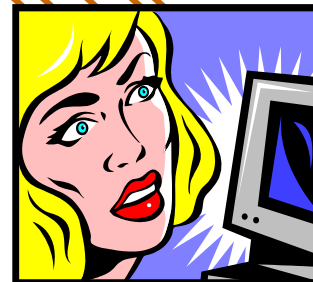
각 비트의 ON/OFF 정보를
각 회로선에 전달

프로그램 로직을 비트
의 조합으로 변환 →
10100100110
01001110001

컴파일러

언어에 정의된 코딩
방식을 이용하여 로
직을 프로그래밍

Hard Wired



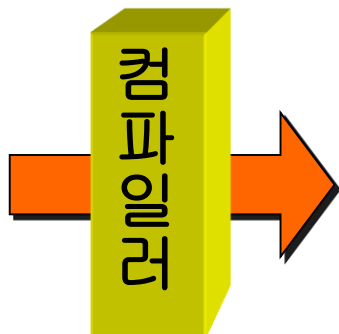


Programming이란?

■ 정보의 변환과정

```
if (a>b)
  display("1234");
else
  display("0000");
```

프로그램 편집



1	0	1	0	0	1	0	0
1	1	0	0	1	.	.	.

메모리



제어기

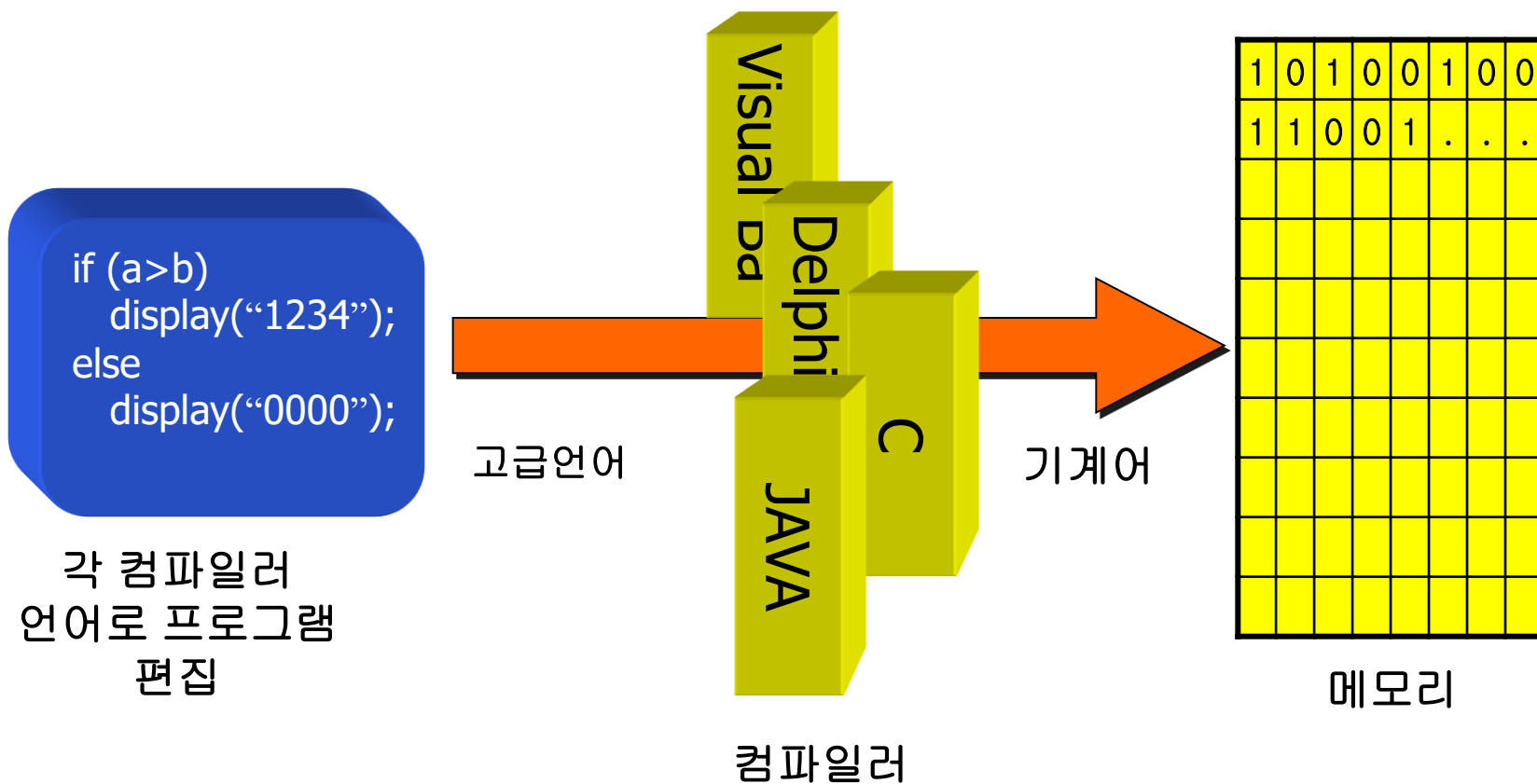
1234

Hard Wired



Programming이란?

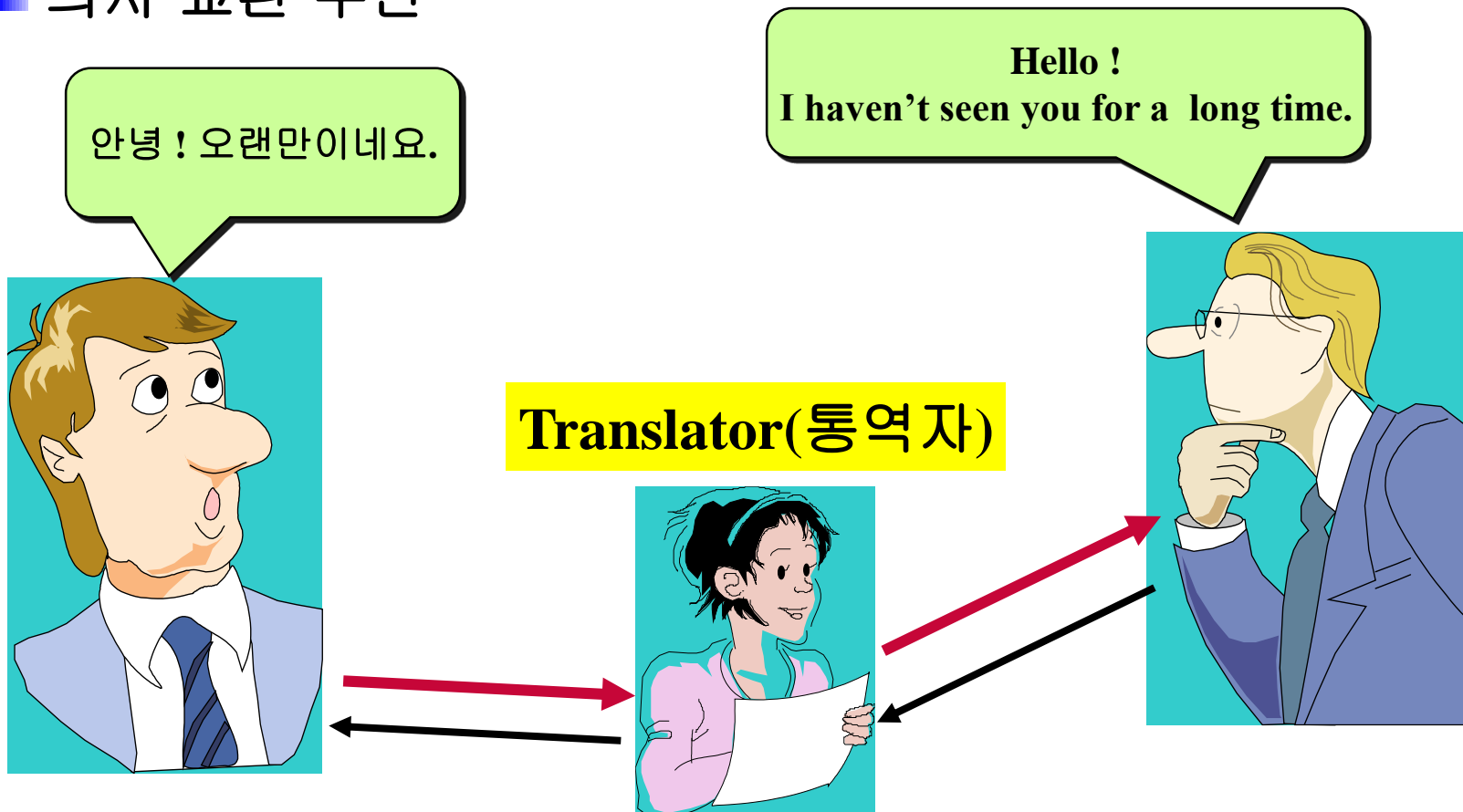
■ Software 변환 과정





Programming 언어란?

- 언어(言語)란 ?
 - 의사 교환 수단





Programming 언어란?



```
include <graphics.h>
include <conio.h>
include <dos.h>
void main()
```

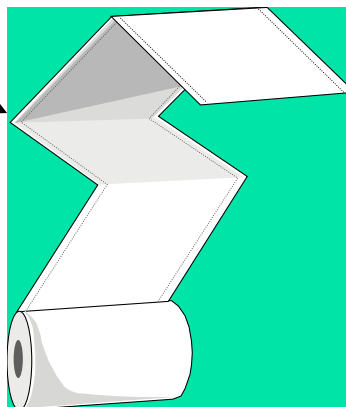
```
int gd=DETECT,gm;
int i;
initgraph(&gd,&gm,"");
settextjustify(1,0);
rectangle(0,0,639,399);
```

000100110110110 11010101011 !



request

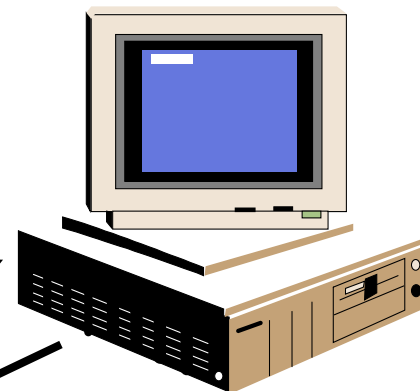
result



Compiler or
Interpreter

request

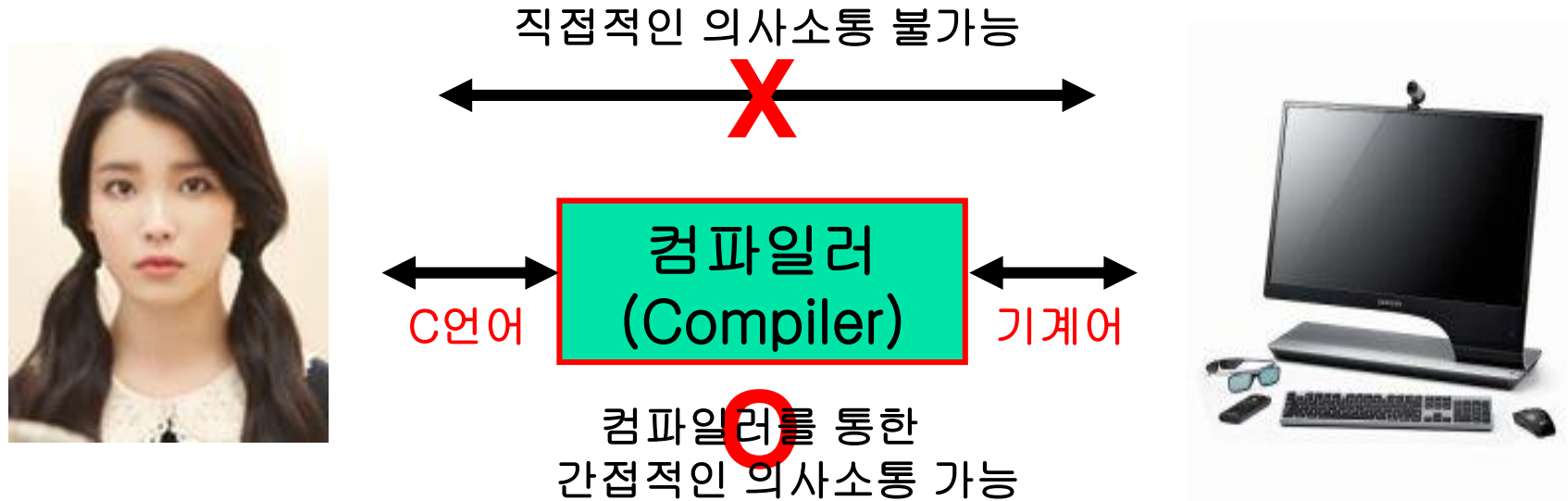
response





Programming 언어란?

- Computer와 의사소통을 할 수 있는 Program의 표현 방식

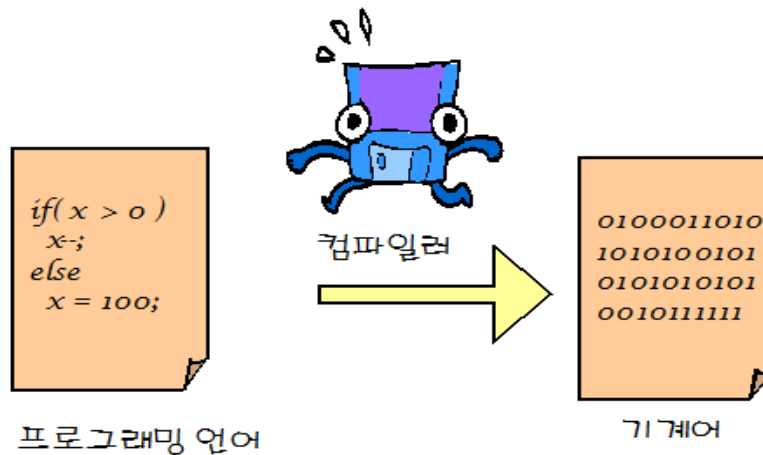


컴퓨터와 대화의 수단은 JAVA 언어!!



Programming 언어란 ?

- Programming 언어는 기계어와 자연어의 중간 위치
- Compiler는 Programming 언어를 기계어로 변환





Programming 언어란?

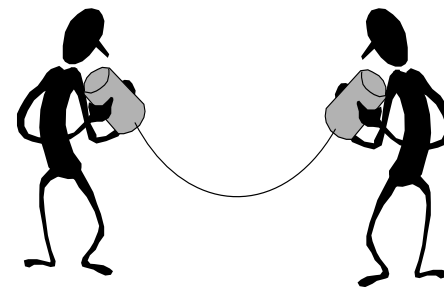
- Computer를 사용하여 어떤 문제를 풀기 위해서 일련의 과정을 기술하는데 사용되는 단어와 기호들의 명세를 Programming 언어라 하고 이들을 조합하여 Program 문장 작성
- 그러면 Computer 이를 이해하여 작동



Programming 언어란?



- 언어의 종류
 - 자연언어(natural language)
 - 형식언어(formal language)
- 형식 언어의 종류



Type 0 : Recursively enumerable languages
recognized by Turing machines

Type 1 : Context-Sensitive languages recognized
by linear-bounded automata

Type 2 : Context-Free languages recognized
by push-down automata

Type 3 : Regular languages recognized
by finite automata



Programming 언어란 ?

■ Natural Language

- 사람과 사람 사이에서 사상, 감정, 의사 등 심적 내용을 서로 전달하기 위한 음성에 의한 기호와 그 사회적인 조직

■ Programming Language

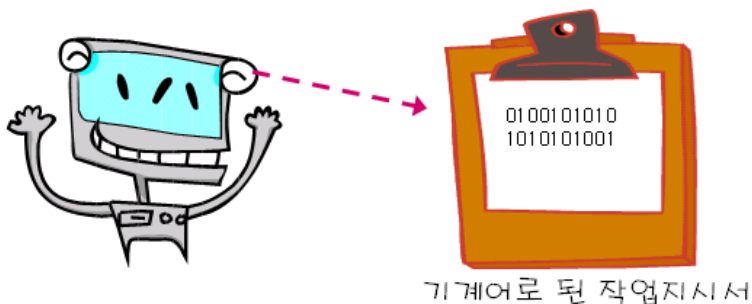
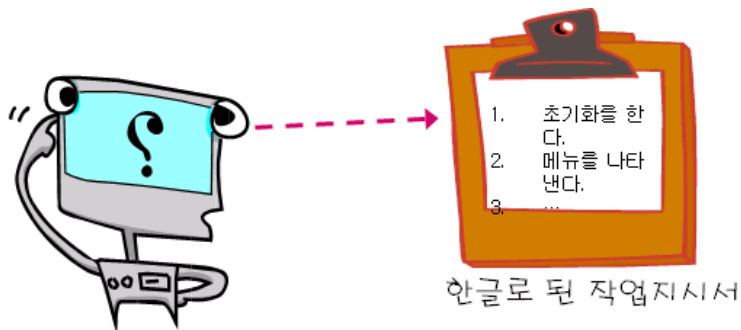
- Program을 작성하는 데 사용되는 언어(言語)
- 기계 판독가능 및 인간 판독 가능한 형태로서 계산을 서술하기 위한 표기 체계
- 컴퓨터(시스템)와 사용자(인간)사이에 정보교환을 위한 표현 체계(Notational System)



Programming 언어란 ?

■ 컴퓨터가 이해할 수 있는 언어는 어떤 것인가?

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



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



Programming 언어란 ?

■ 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

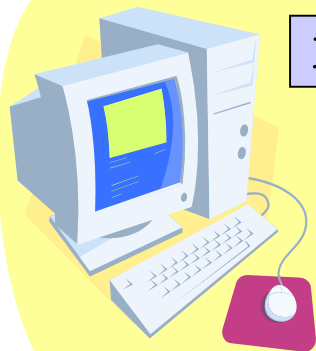




Programming 언어란 ?

■ Programming 언어의 분류

- Machine Language
- Assembly Language
- High-Level Language : 자바, C++, C언어



컴퓨터

기계어

어셈블리어

고급언어

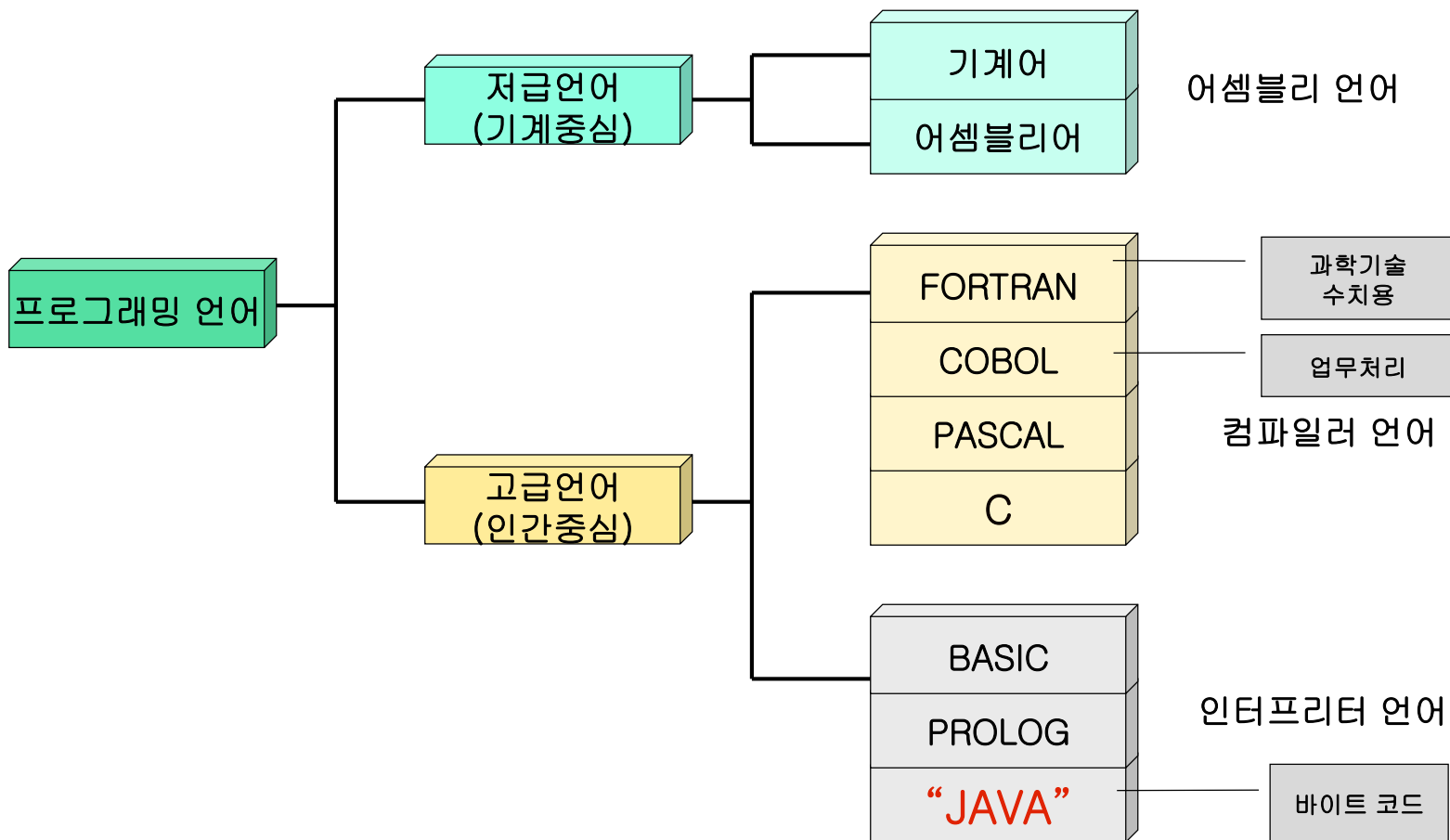


인간



Programming 언어란 ?

■ Programmer가 사용 → 작업 지시서(프로그램) 작성





Programming 언어란 ?

■ Low Level Language

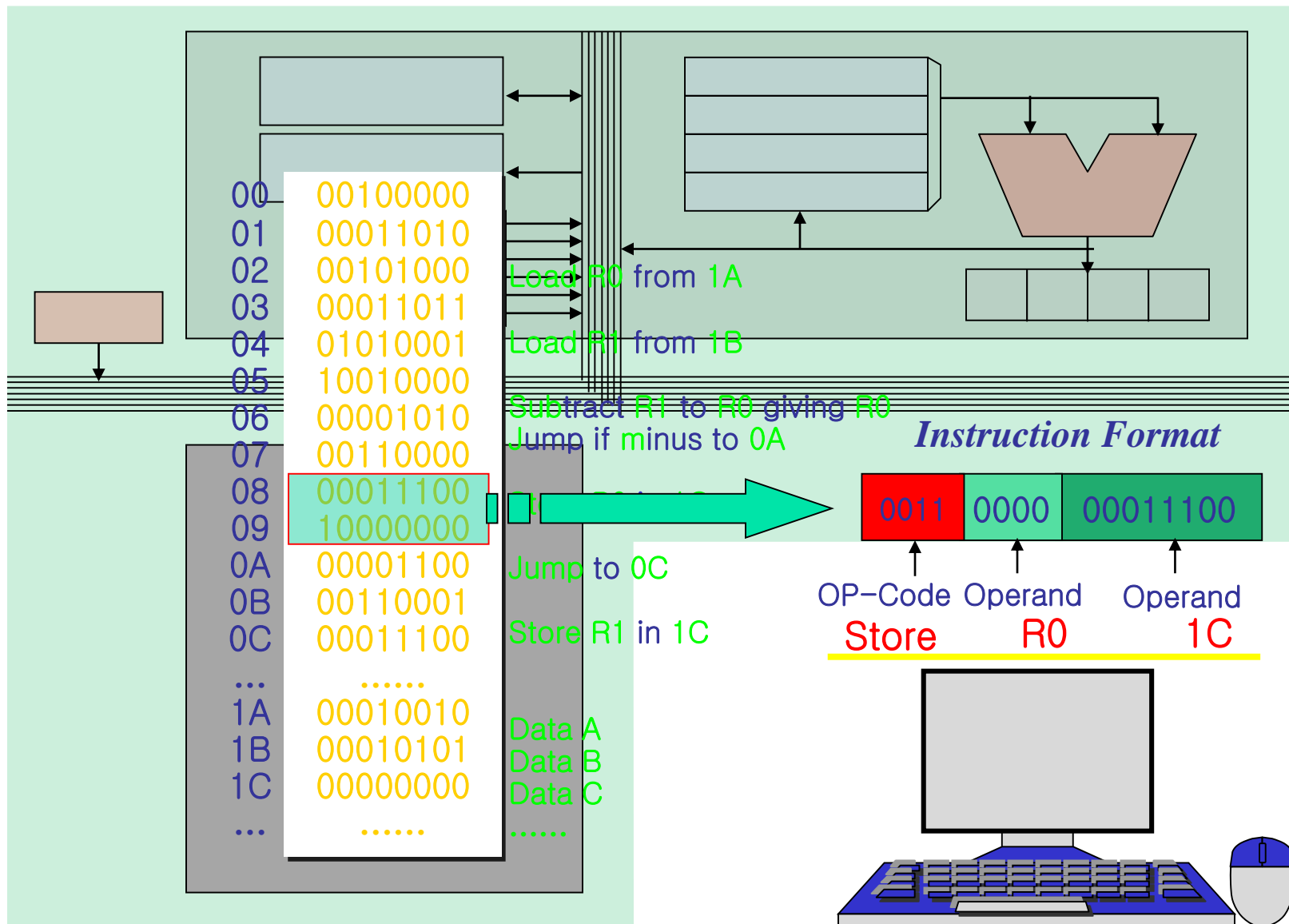
- 기계어와 어셈블리 언어를 의미
- 하드웨어에 관련된 직접제어 가능
- 프로그램 작성시 상당한 지식과 노력이 필요

〈표 9.1〉 어셈블리 언어로 표현된 명령어들의 예

연산 코드 기호	의미
LOAD X	$CON(X) \rightarrow R$
STORE X	$R \rightarrow CON(X)$
CLEAR X	$0 \rightarrow CON(X)$
ADD X	$R + CON(X) \rightarrow R$
INCREMENT X	$CON(X) + 1 \rightarrow R$
SUBTRACT X	$R - CON(X) \rightarrow R$
DECREMENT X	$CON(X) - 1 \rightarrow CON(X)$
COMPARE X	if $CON(X) > R$ then $GT = 1(on)$



Machine Language





Machine Language



- ✓ '0'과 '1'로만 구성(코드(code)라고 함)
- ✓ 가장 저수준의 언어
- ✓ 효율성이 가장 높음(기계적 측면)
- ✓ 기계(컴퓨터) 내부 사항을 자세히 알아야 함
- ✓ 배우기 어려움(인간적 측면)
- ✓ 사용하기 어려움 – 프로그램작업이 지루함
- ✓ 컴퓨터 종류마다 다름 – machine dependent
- ✓ 에러 발생이 빈발



Machine Language



- 기계어로 고급 언어를 택할 경우의 문제점
 - 기계의 복잡성 증가(가격 상승의 원인)
 - 고급 언어에 비해 적응성 결여
 - 처리 속도가 느리다
 - 디지털 컴퓨터와 직접 응답하는 표기법
 - 사용자는 이해하기 어렵다
 - 실행시간(Run Time) 효율성 고려



Assembly Language

```
LOAD R0 A ; Load R0 from A
;
LOAD R1 B ; Load R1 from B
;
SUBT R0 R1 ; Subtract R1 to R0
JPIM L0 ; Jump if minus to L0
;
STOR R0 C ; Store R0 in C
;
JUMP L1 ; Jump to L1
;
L0: STOR R1 C ; Store R1 in C
L1: .....
A: DB 18
B: DB 21
C: DB 0
.....
```

assemble

```
00 00100000
01 00011010
02 00101000
03 00011011
04 01010001
05 10010000
06 00001010
07 00110000
08 00011100
09 10000000
0A 00001100
0B 00110001
0C 00011100

... .....
1A 00010010
1B 00010101
1C 00000000

... .....
```



Assembly Language



- ✓ 기계 명령어와 1:1 의 기호나 상징 단어로 대치
- ✓ 효율적 실행(기계적 측면)
- ✓ mnemonic code – 기계어 코드보다 배우기 쉬움
- ✓ 의사 코드(Pseudo Operation) 개발
- ✓ Assembler라는 변환기 사용(Assembler 개발)
- ✓ 컴퓨터 내부 사항을 자세히 알아야 함
- ✓ 컴퓨터 기종마다 다름 – machine dependent
- ✓ 에러 발생이 쉬움



High Level Language

- 하드웨어에 관련된 지식 없이도 프로그램 작성 가능(기계와 무관)
- 사용자의 명령을 컴파일러가 해석, 기계어보다 낮은 효율성
- 일상 적인 언어, 기호 등을 그대로 이용하므로 사용자의 편리성 증대(문장과 같은 판독력)
- 기억장소를 임의의 기호(symbol)에 저장하여 사용
- 하나의 명령으로 다수의 동작 가능
 - 예) $A = B + C * D$



High Level Language

C

```
if ( a > b )  
    c = a;  
else  
    c = b;  
c = ( a > b ) ? a : b;
```

COBOL

```
IF A IS GREATER THAN B  
    MOVE A TO C  
ELSE MOVE B TO C.
```

FORTRAN

```
IF(A.GT.B)  
    C = A  
ELSE  
    C = B  
ENDIF
```

compile

00	00100000
01	00011010
02	00101000
03	00011011
04	01010001
05	10010000
06	00001010
07	00110000
08	00011100
09	10000000
0A	00001100
0B	00110001
0C	00011100
...
1A	00010010
1B	00010101
1C	00000000
...



High Level Language



- ✓ 고급 언어 – 생산성이 높음
- ✓ 기계어보다 낮은 효율성
- ✓ 자연언어나 수식과 유사함 (배우기 쉽다)
- ✓ 기억장소를 임의의 기호(symbol)에 저장하여 사용
- ✓ 컴퓨터 기종에 불변 (기계 독립적) – 문제 지향적
- ✓ 컴퓨터 내부 사항을 잘 몰라도 됨
- ✓ 표준화가 잘 됨 – 프로그램 이식성이 좋음
- ✓ 컴파일러(compiler)라는 변환기를 사용해야 함



최고 수준 언어

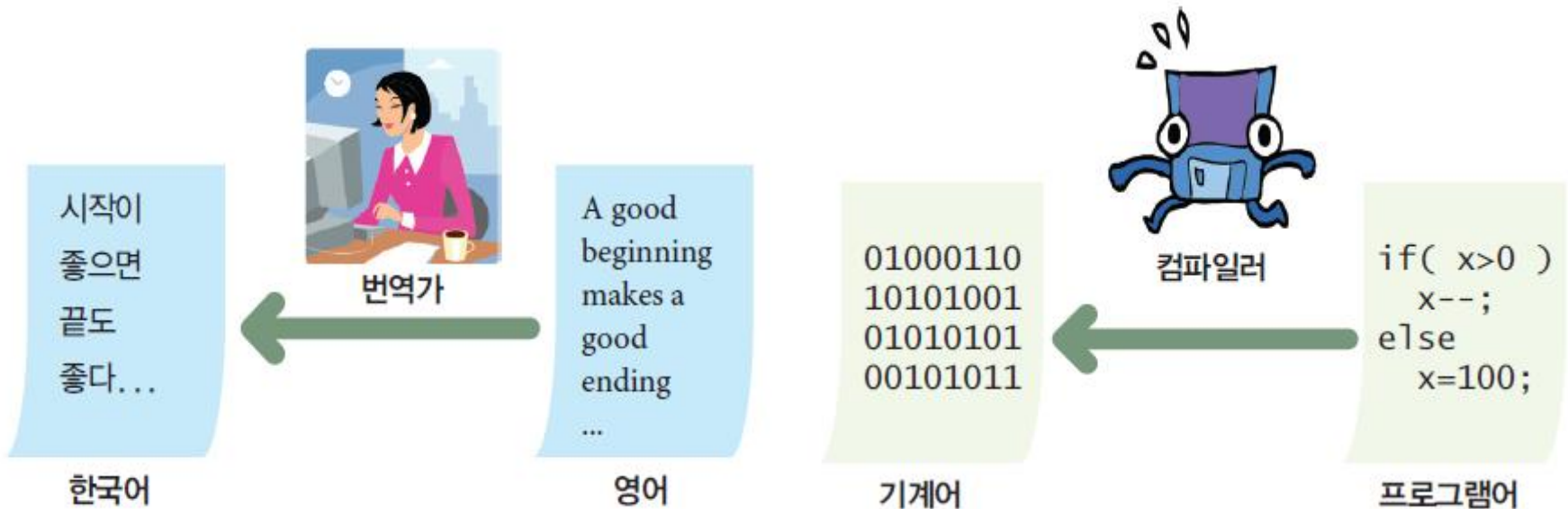


- 프로덕션지향 언어(production oriented language)
 - 제4세대 언어
 - 컴퓨터 전문가의 프로그래밍과 대형 컴퓨터를 이용한 정보 시스템 개발을 위해 설계
 - 컴퓨터 제작 회사의 데이터베이스 관리 시스템 소프트웨어와 함께 구성
- 사용자 지향 언어(User oriented language)
 - 제4세대 언어
 - 최종 사용자들을 위해 설계
 - 데이터베이스로부터 정보를 얻어내고 기능 중심의 정보를 만들기 위하여 프로그램을 작성



Program의 번역과 실행

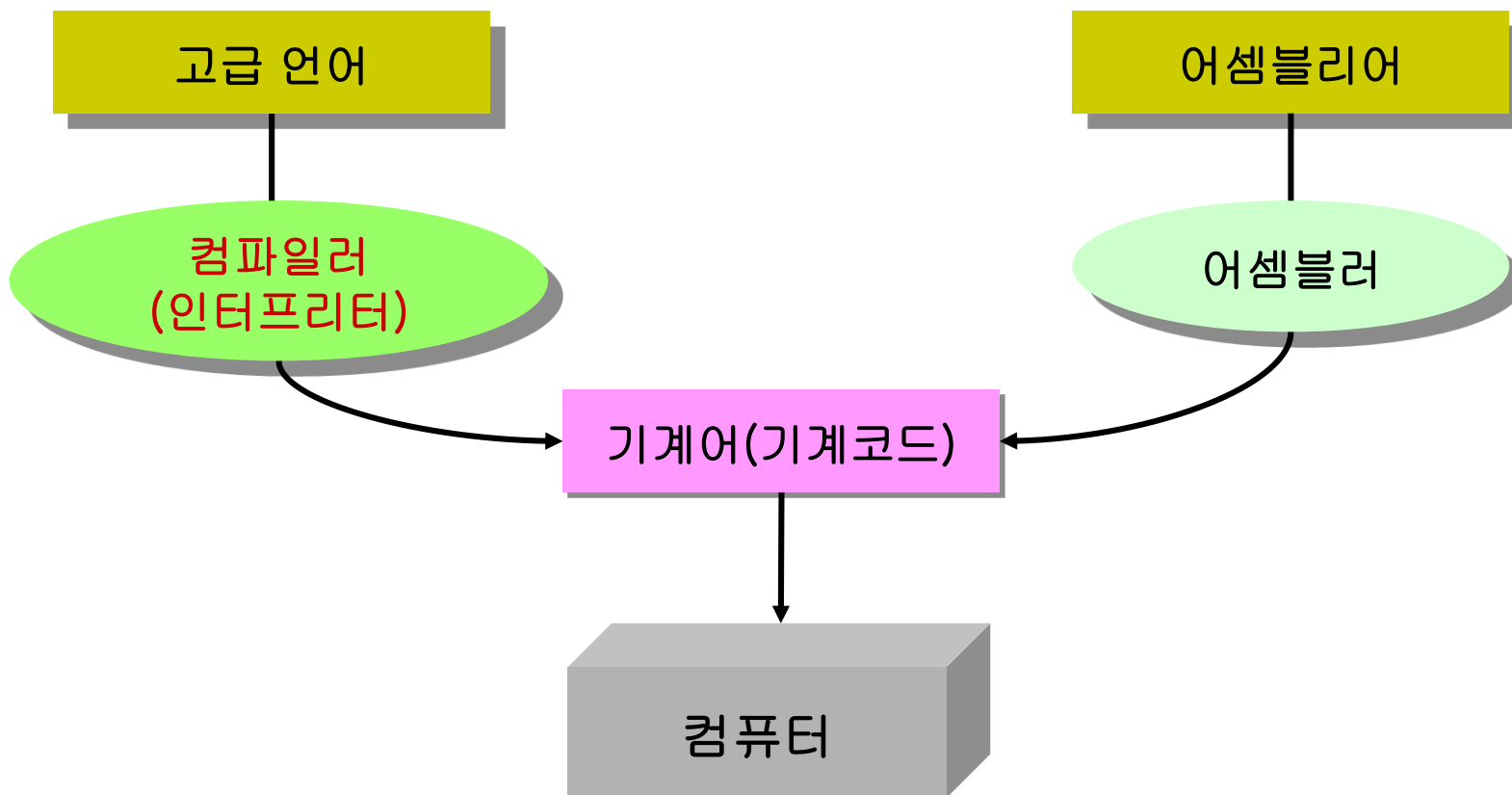
- Compiler는 Programming 언어로 작성된 Program을 Machine Language로 변환





Program의 번역과 실행

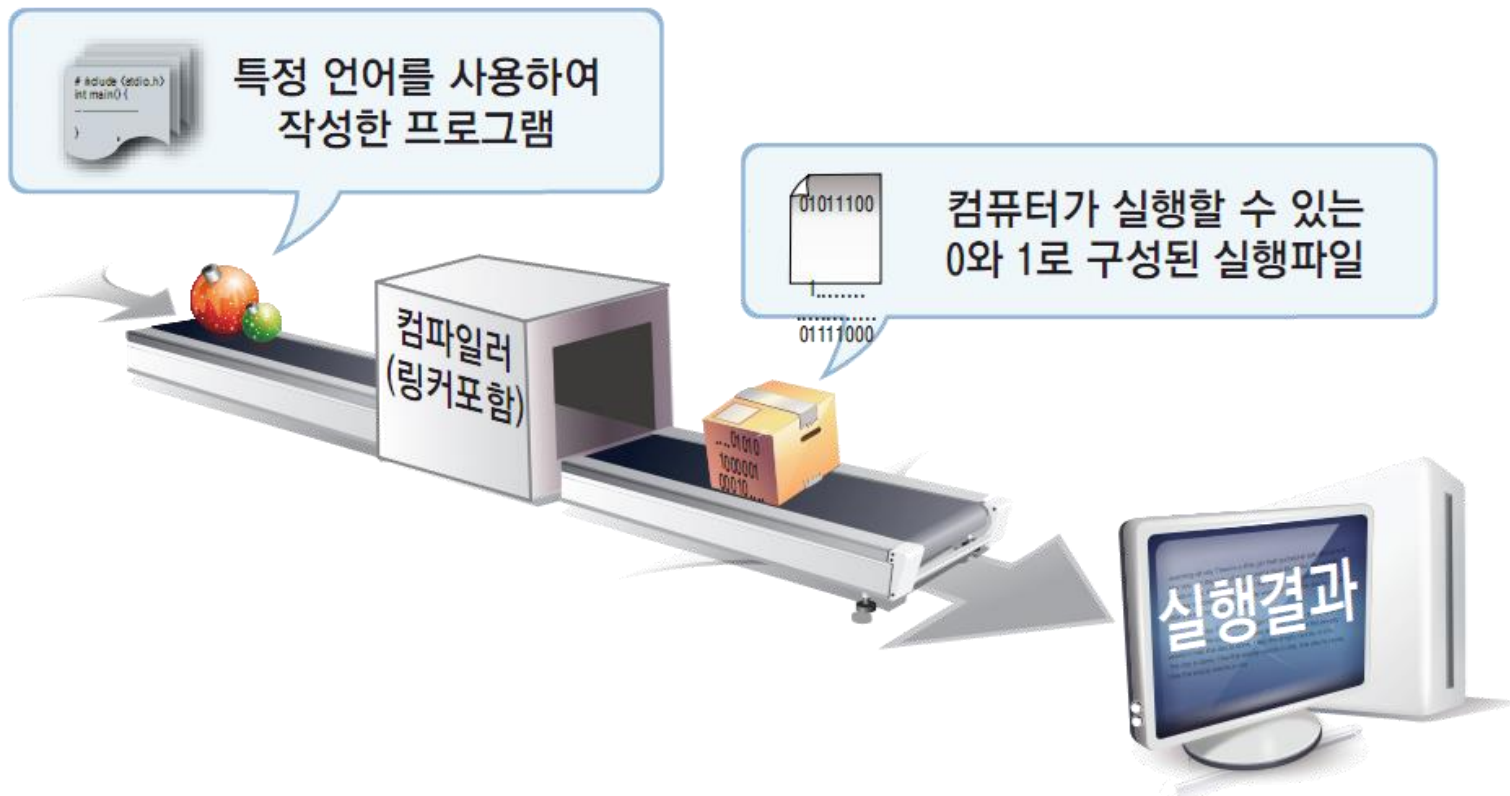
■ 기계어로의 변환





Compile 기법

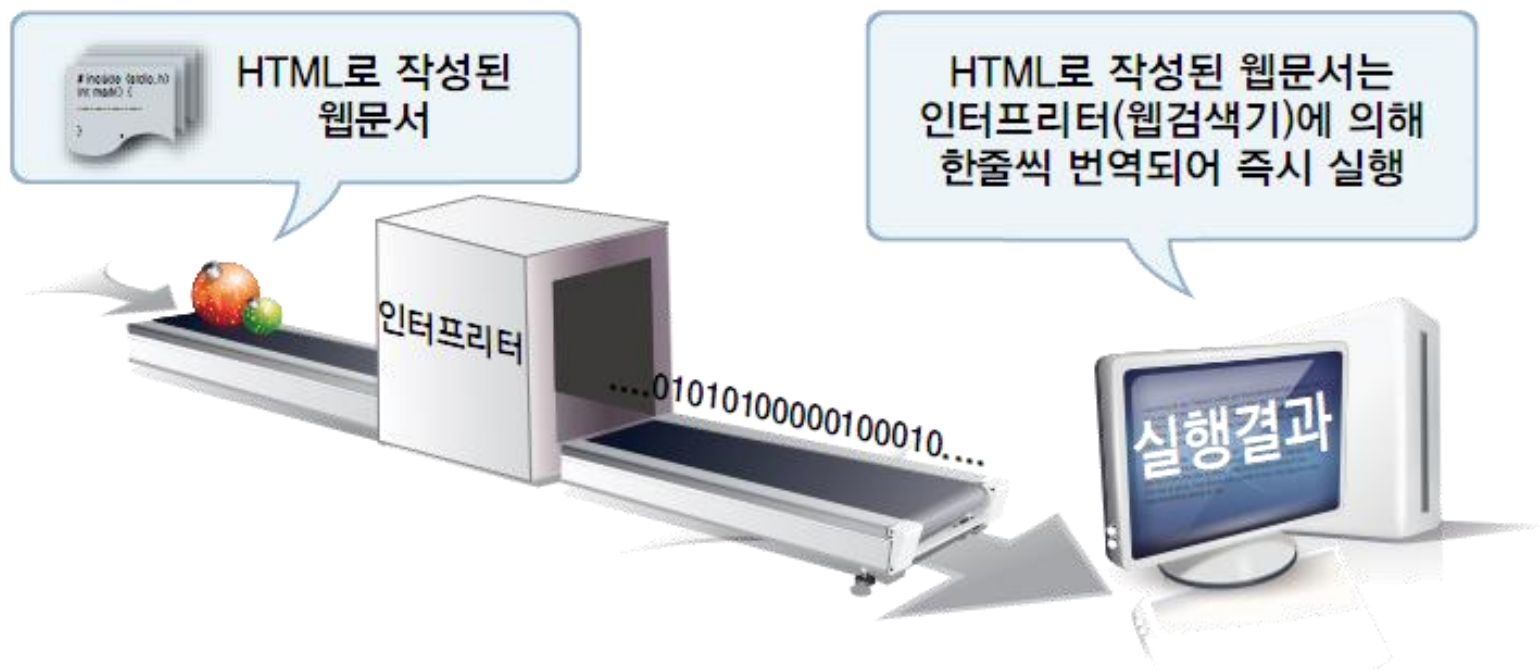
- Program이 Compiler에 의해 0과 1로 구성된 Binary File(0과 1로 구성된 파일)로 번역된 다음, 번역된 File이 Computer에서 실행되는 기법





Interprete 기법

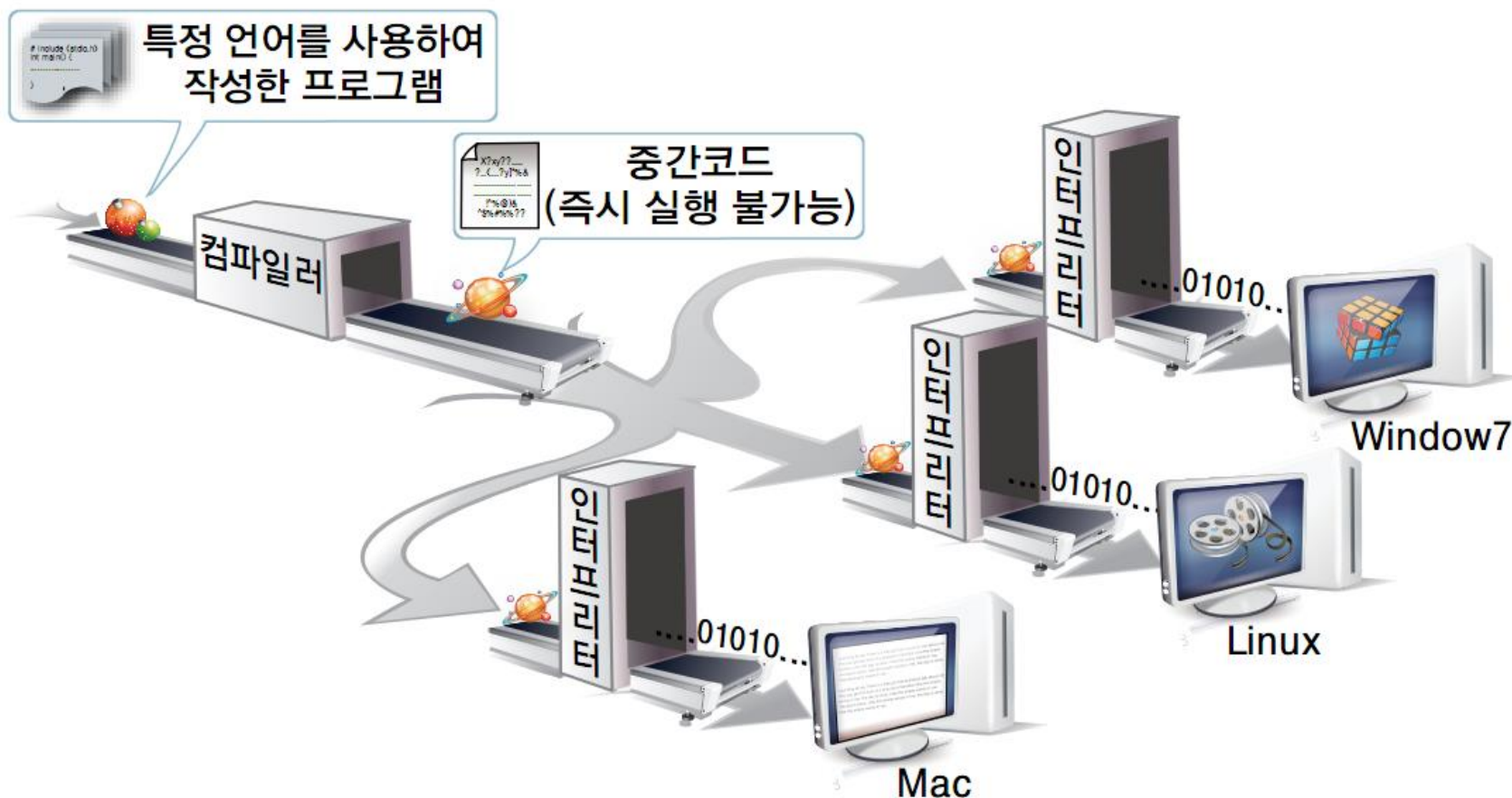
- Interprete(해석) 기법은 인터프리터(interpreter)에 Program 을 실행시키는 방법





Hybrid 기법

- Compile 기법과 Interpret 기법을 모두 사용하는 방식

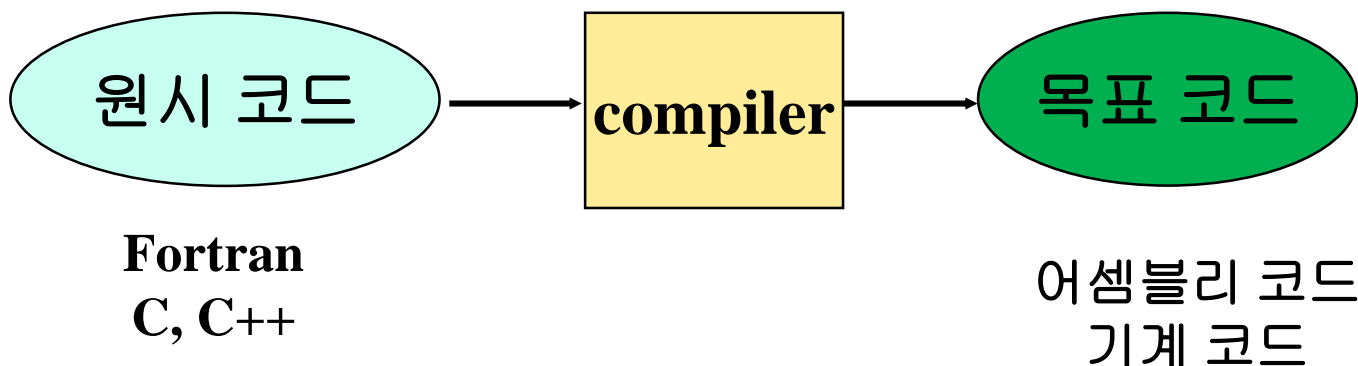




Compiler



- 고급 언어로 작성된 프로그램을 기계어나 어셈블리어로 번역
- 원시 코드를 목표 코드로 번역하는 소프트웨어 도구



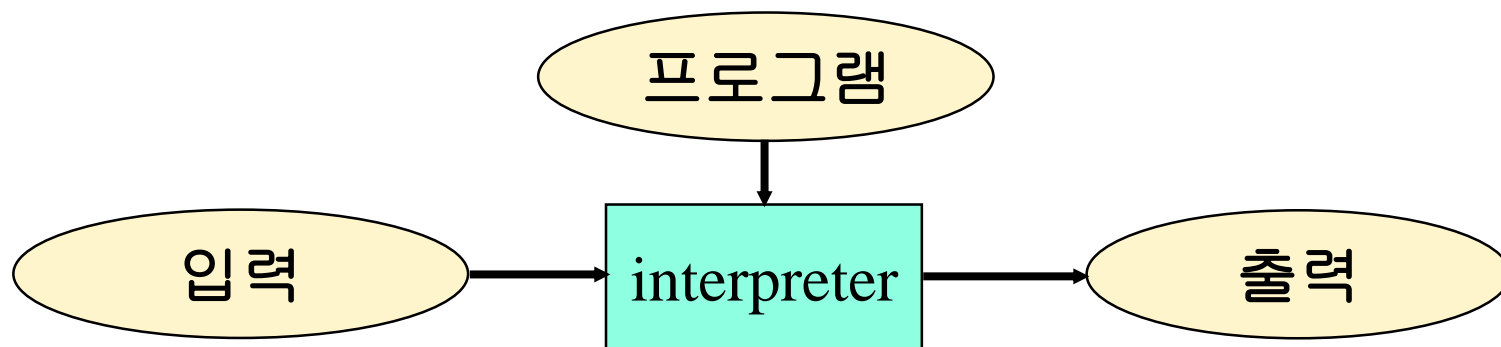
- 모든 내용을 한번에 기계어로 번역
 - 실행은 기계어로 번역된 프로그램으로
- 일단 번역이 완료되면, 매우 빠르게 실행 가능
 - FORTRAN, Pascal, C 등



Interpreter




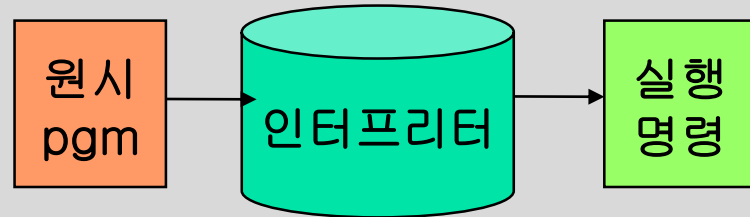
- 프로그램이 번역 과정 없이 해석되면서 실행
 - 한 고급 명령어 씩 읽고 해석, 해석된 것 실행(이 과정을 반복)
- 예) Basic, Prolog, LISP, APL, JavaScript ... 등
- 인터프리터는 고급언어 프로그램을 처리하는 기계의 소프트웨어 모의실험으로 동작
- 언어에 대한 가상 기계 제공
- 문장 해석에 따른 실행 속도가 느리다





Compiler와 Interpreter



특징	컴파일러	인터프리터
번역 방법	컴파일러들은 원시 프로그램을 바로 목적 프로그램으로 번역한다	원시 프로그램을 기계어로 된 목적 프로그램으로 변환하지 않고 중간어로 변환해 놓고 그것을 해석하여 실행한다
		



Compiler와 Interpreter

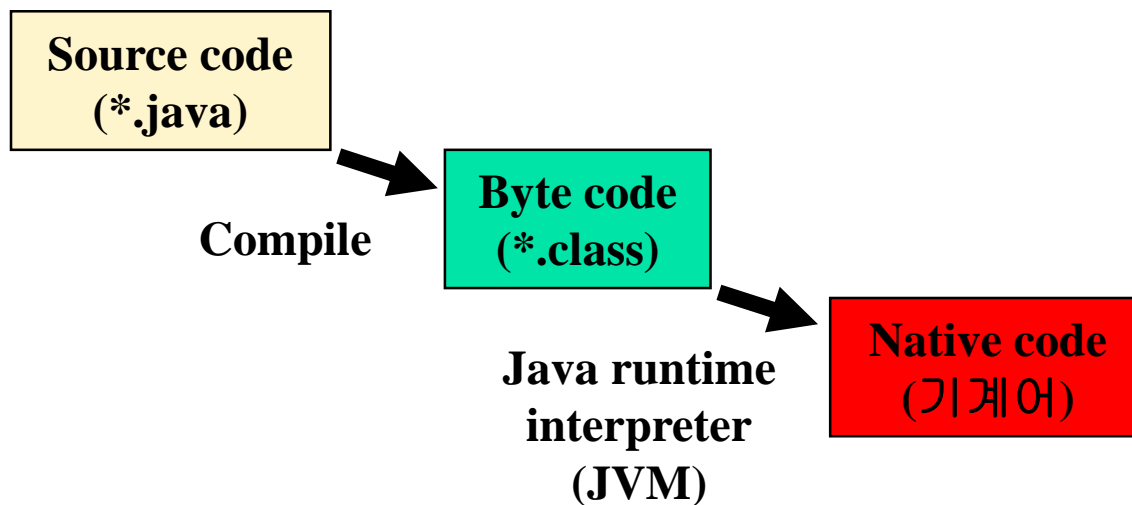


특징	Compiler	Interpreter
장점	처리 속도가 빠르다	Program 실행 시간 동안 고치고 개선하기가 용이하다
	매번 번역할 필요는 없다	번역기 Program의 크기가 작다
단점	번역기 Program의 크기가 크다	처리 속도가 느리다
	Program 전부를 실행 전에 한꺼번에 번역한다	Program을 한 문장씩 기계어로 번역한다
결과	목적 언어로 된 프로그램	실행의 결과
적용 언어	포트란, 코볼, 파스칼, C	베이식, 리스프, 프롤로그, 로고



Hybrid 방식

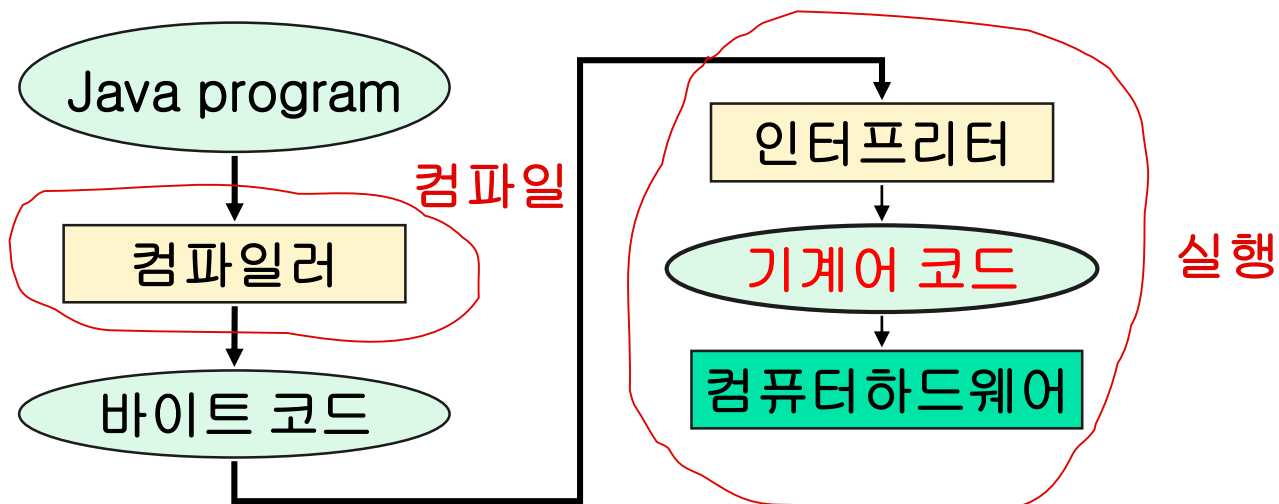
- Source Code를 중간 언어로 번역(Compiler)
- 번역된 중간 코드를 한 명령어 씩 실행(Interpreter)
- 예) Lisp, Snobol4, APL, Prolog, JAVA 등
- Program의 호환성 및 Compiler 개발 효율 높일 수 있음
 - JAVA의 방법
 - Byte Code 생성
 - Byte Code 실행



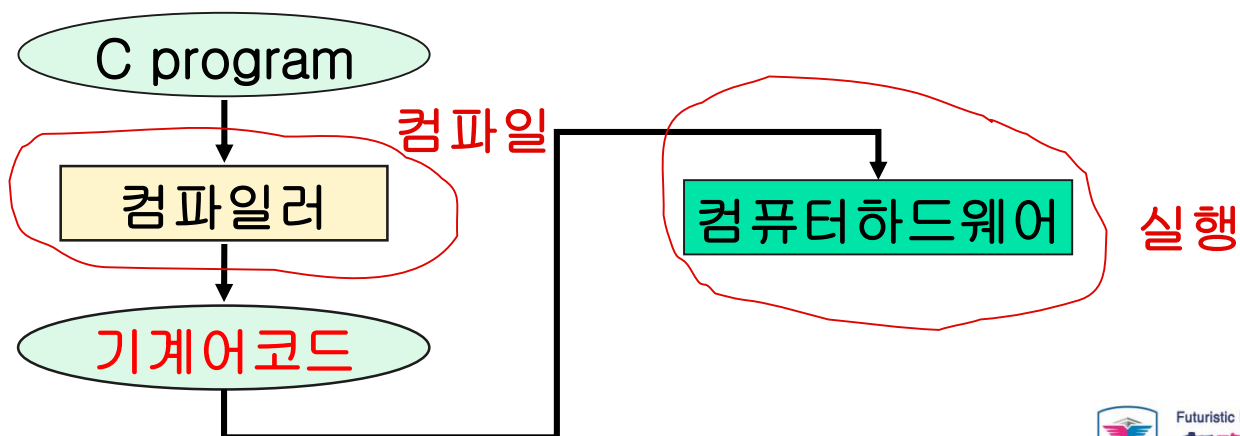


Hybrid 방식

■ JAVA



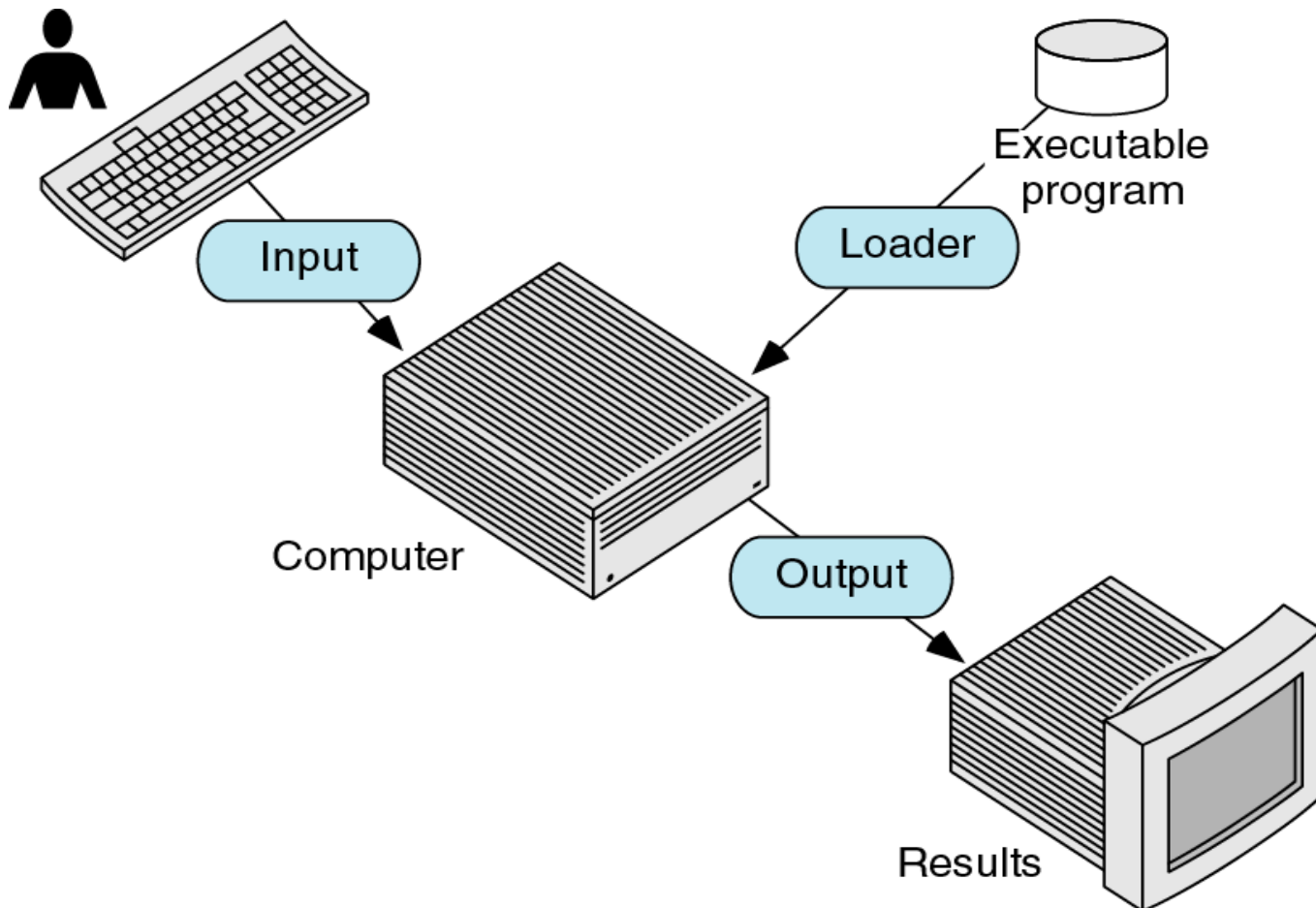
■ C/C++





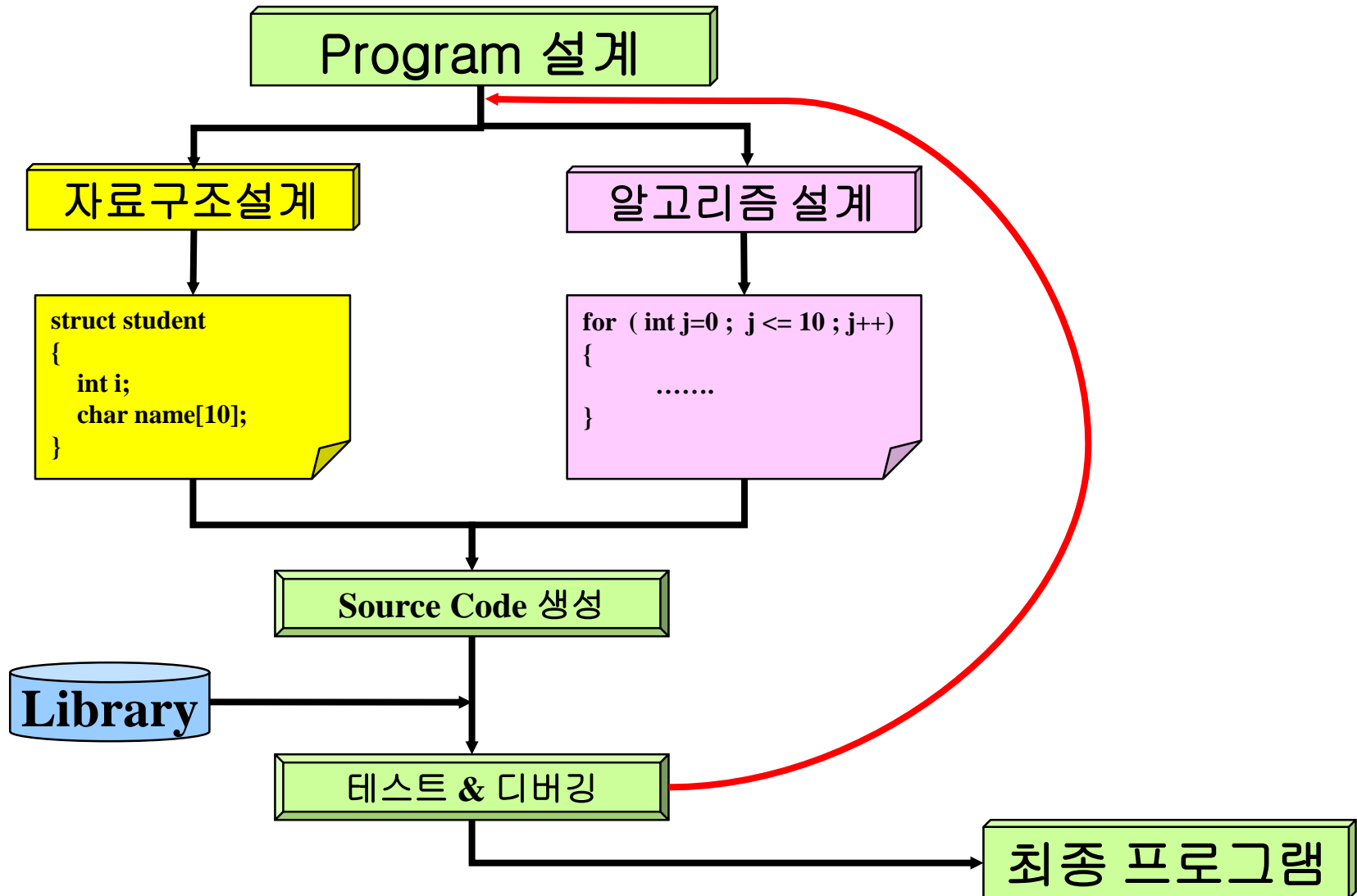
Program 작성 과정

■ Program 실행



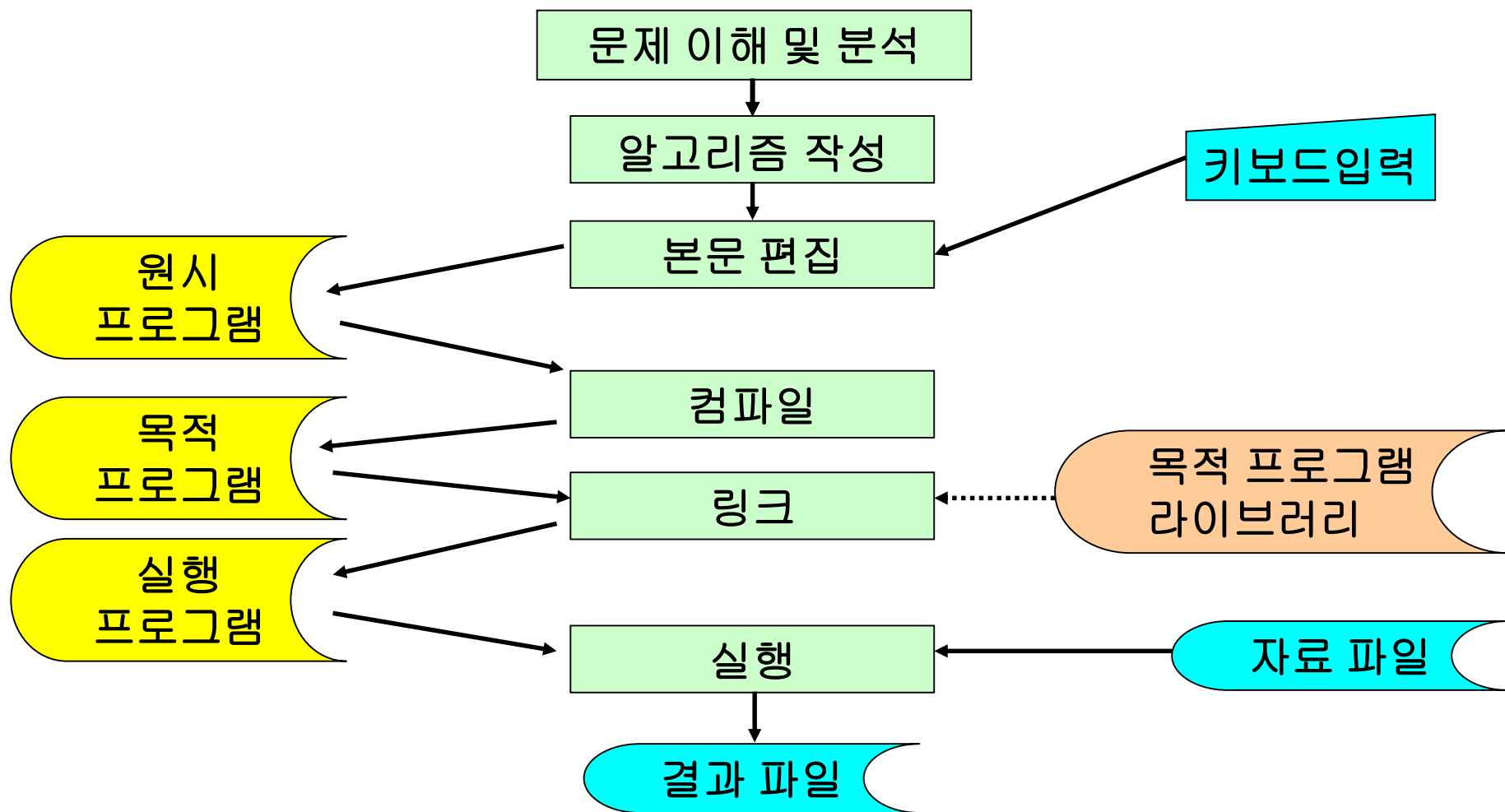


Program 작성 과정





Program 작성 과정





Program 작성 과정



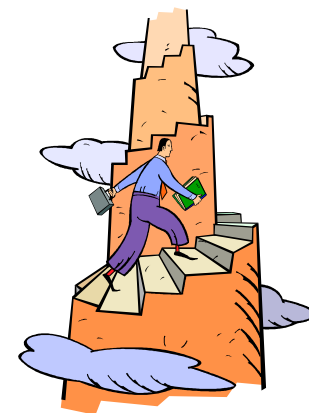
- 문제 이해 및 분석
 - 도메인(Domain) 지식
 - Data 분석 -> 공통점 찾기
 - 작성하려는 Program이 어떠한 역할을 수행해야 하는지를 명확하게 분석
 - 그 Program이 수행해야 하는 순서와 행위를 설정
 - 작성하려는 Program을 시간 순서적으로 정리
 - 작성하려는 Program을 처리 단위로 Group화 정리
 - 특별한 사건(Event) 시에 처리 절차를 정리



Program 작성 과정



- Algorithm 작성
 - 수행해야 할 일의 종류와 순서를 정함
 - 처리 대상 파악
 - 처리 순서의 정립
 - 도식화(Flow-Chart 작성) : Symbol 사용
 - 문서 작성





Programming 언어 배우기

(프로그래밍) 언어 배우기

— “이렇게 쓰는 거야” : 문법 구조

겉모양 → Syntax

— “이게 이런 뜻이란다” : 의미 구조

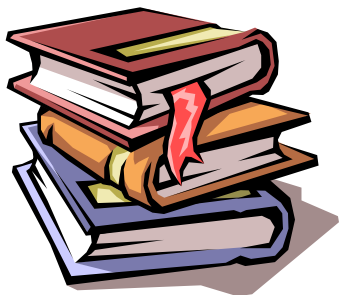
속내용 → Semantics

문법과 의미 (겉과 속) 이 과연 완전히
구분될 수는 없는 것이지만 ...

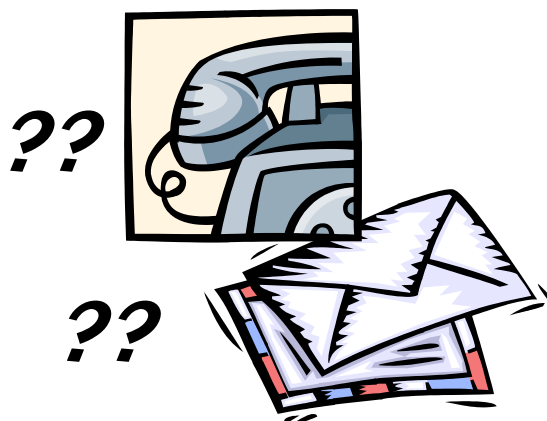


Algorithm

- 어떤 일을 해결하기 위한 순서
- 어떤 특정한 해결하기 위한 명령어들의 집합
- 외부에서 입력되는 자료가 존재
- 적어도 한가지의 결과를 생성
- 각 명령들은 모호하지 않고 명확
- 어떤 경우에도 한정된 수의 단계 뒤에는 반드시 종료
- 원칙적으로 모든 명령들은 종이와 연필만으로 수행될 수 있는 기본적인 것



??





Algorithm

[질문] 오븐의 사용법만 배우고 음식 재료만 있으면 누구나 요리가 가능한가?

A) 요리법을 알아야 한다

프로그램이 요리와 같다면
알고리즘은 요리법에 해당한다





Algorithm

■ 빵 만들기

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



->



->



->



->





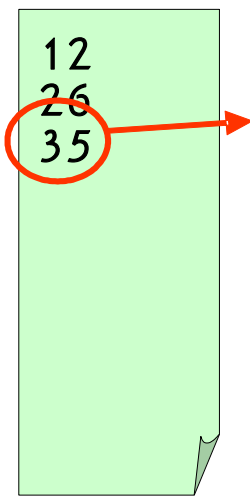
Algorithm

■ Algorithm 작성

- 영어와 국어와 같은 자연어
- 의사 코드(pseudo-code)
- 순서도(flowchart)

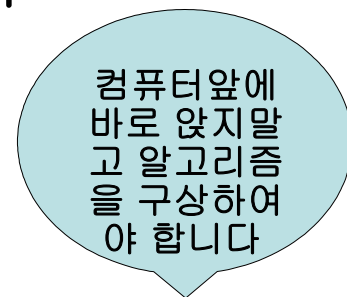
■ Algorithm을 도형 등의 그림으로 표현한 것

■ 예제: 숫자들의 리스트에서 최대값을 구하는 문제



노트

최대값





Algorithm



- 사람의 입장이 아니라 Computer(Programming 언어)의 입장으로 생각할 것
- 예) 앞의 삼각형 넓이 구하기
 - 입력
 - 삼각형 높이, 넓이 \rightarrow 데이터 형은 \rightarrow 정수? 실수?
 - 처리
 - 계산 값을 지정하기 위한 변수?
 - 공식 : $1/2 * (\text{높이}) = \text{넓이}$
 - 출력
 - 계산 결과를 출력 : 넓이



Algorithm

- 1에서 100까지의 합을 구하라

<방법1> 1 부터 100까지의 숫자를 열로 정렬한 후 더한다.

$$\begin{array}{r} 1 \\ 2 \\ 3 \\ : \\ : \\ 98 \\ 99 \\ +100 \\ \hline 5050 \end{array}$$

컴퓨터(프로그
램 언어)의 입장
에서 코딩 방법
생각

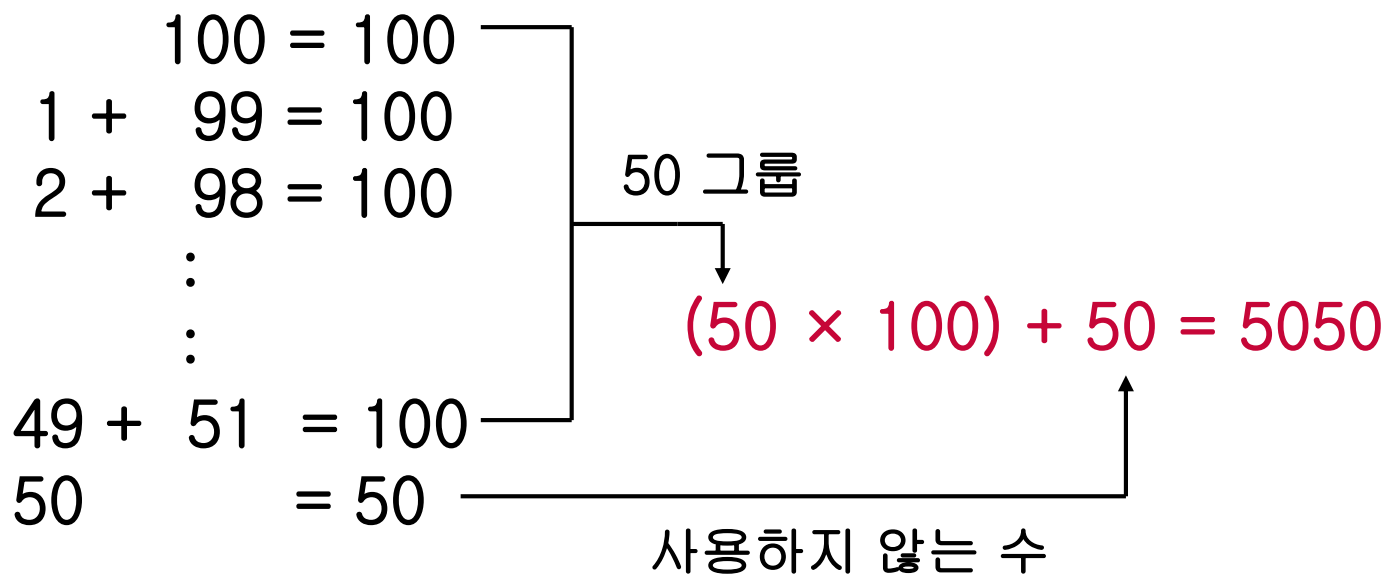


Algorithm



■ 1에서 100까지의 합을 구하라

<방법2> 두 수의 합이 100이 되도록 숫자들을 그룹으로 정렬한다. 그룹의 개수에 100을 곱한 후에 사용되지 않는 수를 앞에서 구한 총합에 더한다.





Algorithm



■ 1에서 100까지의 합을 구하라

<방법3> 공식을 사용하는 방법

$$\text{sum} = \frac{\text{item} (\text{first} + \text{last})}{2}$$

item = 더하는 항의 수 (100 개)

first = 더하는 처음 숫자 (1)

last = 더하는 수의 마지막 숫자 (100)

$$\text{sum} = \frac{100 (1 + 100)}{2} = 5050$$



Futuristic Innovator

京福大學校
KYUNGBOK UNIVERSITY



Program 작성 과정



- Source Program 작성
 - Editor를 이용하여 원하는 작업의 내용(Algorithm)을 Programming 언어의 문법에 맞게 명령어로 바꾸어서 문서를 작성하여 Source Code(File) 작성
 - Source File
 - Source Code가 들어 있는 Text File
 - 작성된 문서를 보조 기억 장치에 저장
 - ASCII Code 형태로 “Test.java”처럼 확장자는 “*.java”로 하여 File로 저장



Program 작성 과정



- Object Code(Program) 생성
 - Source Program을 Computer가 이해할 수 있는 형태의 Code(Machine Language)로 번역
 - Compile
 - Object File(목적 파일)
 - Machine Language로 변환된 File
 - DOS : * .obj 확장자
 - UNIX : *.o 확장자
 - 예) test.obj
 - JAVA 언어는 Byte Code를 생성 : *.class



Program 작성 과정



- Executable File 생성
 - Object File들을 실행에 필요한 Library File 또는 Sub Program들과 연결하여 하나의 Executable File 생성
 - Link 시킨다고 함
 - 실행이 가능한 파일
 - 예) test.exe



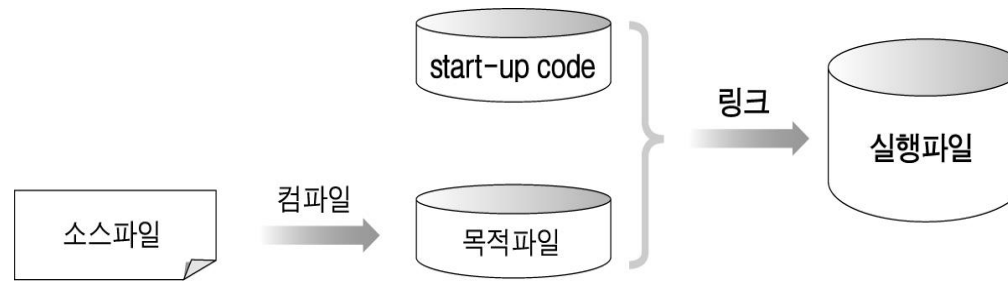
Program 작성 과정

■ Link의 목적

- 분할 Compile 된 Object File들을 연결시킴
- 하나의 Program을 여러 개의 Source File로 작성하고 개별적으로 Compile 한 후에 하나의 Executable File로 만들 수 있음



- Executable File이 될 수 있는 자격을 갖추는 과정
- Object File에 start-up code를 붙여서 운영체제가 실행시킬 수 있는 Executable File을 만듦





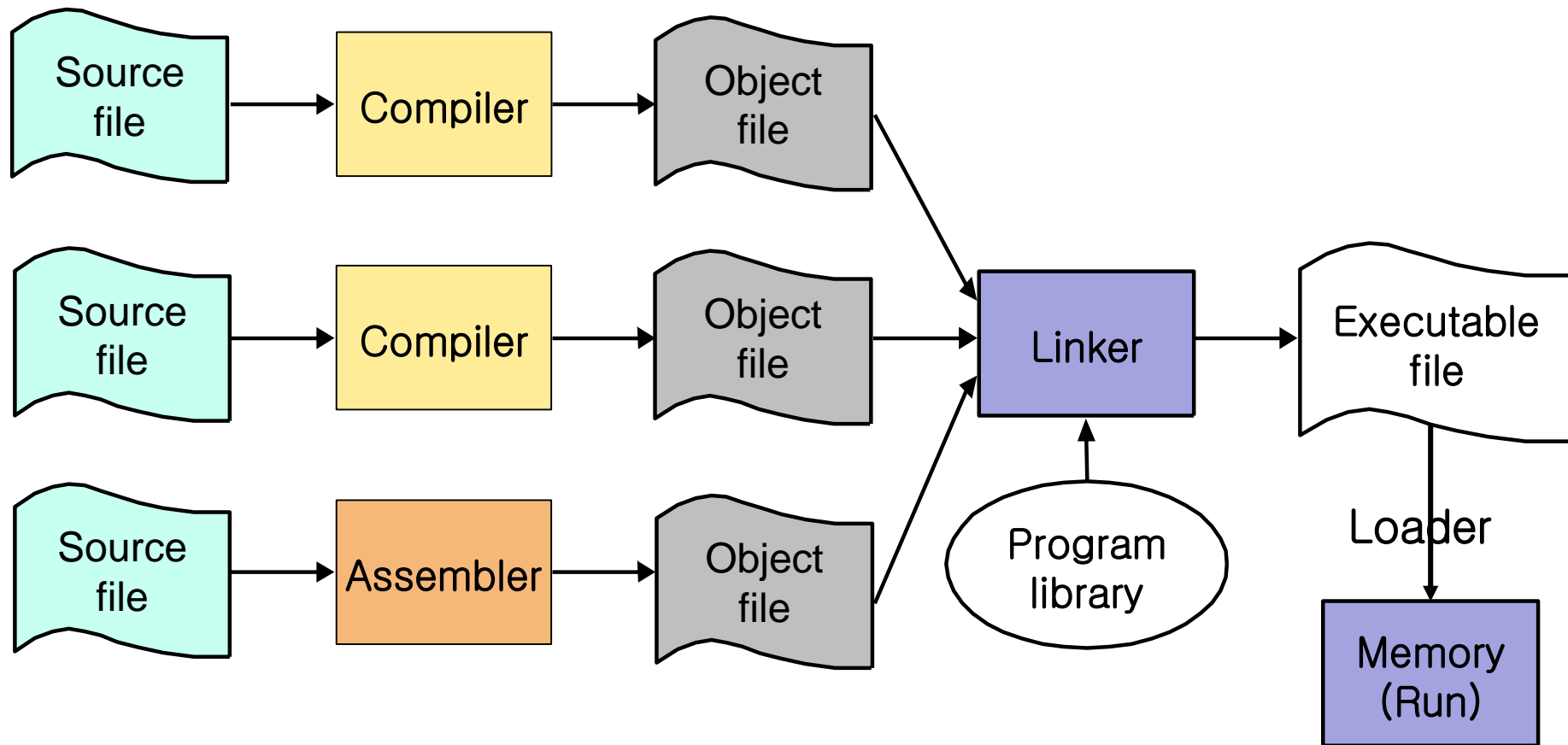
Program 작성 과정



- Testing 및 실행
 - Executable File을 작동시켜서 확인
 - 기대하는 결과 값과 다르면 그 원인을 파악하여 Source Code를 수정 후 Program 개발 과정을 반복하여 작업



Compile 작업





Compile 작업



■ Compile이란?

- 사람이 이해하는 High Level Programming Language를 Machine Language Code로 변환한 후, 최종적으로 운영 체제에서 실행할 수 있는 형태의 File로 만드는 작업을 말함
- 요약해서 Compile이라고 표현하나, 그 내부 과정에서는 Compiler, Assembler, Linker 등의 Program이 동작하여 이루어짐
- 최근의 Compiler들은 3가지 Program을 모두 내장하고 있어서 한번에 모든 작업을 처리할 수 있음



Compile 작업

■ Source, Assembly, Machine Language Code 비교

```
void main()
```

```
{
```

```
int n,s;
```

```
printf("Enter upper limit: ");
```

```
scanf("%d",&n);
```

```
s = count(n);
```

```
printf("Sum of i from 1 to %d =  
%d\n",n,s);
```

```
}
```

```
.LCFI1:
```

```
sd $28,32($sp)
```

```
.LCFI2:
```

```
move $fp,$sp
```

```
.LCFI3:
```

```
.set noat
```

```
lui $1,%hi(%neg(%gp_rel(count)))
```

```
addiu $1,$1,%lo(%neg(%gp_rel(count)))
```

```
daddu $gp,$1,$25
```

```
.set at
```

```
sw $4,16($fp)
```

```
sw $0,24($fp)
```

```
li $2,1 # 0x1
```

```
sw $2,20($fp)
```

```
.L3:
```

```
lw $2,20($fp)
```

```
lw $3,16($fp)
```

```
slt $2,$3,$2
```

```
beq $2,$0,.L6
```

```
b .L4
```

```
00000000 7f45 4c46 0102 0100 0000 0000 0000 0000  
00000200 0002 0008 0000 0001 1000 1060 0000 0034  
00000400 0000 6c94 2000 0024 0034 0020 0007 0028  
00000600 0023 0022 0000 0006 0000 0034 1000 0034  
00001000 1000 0034 0000 00e0 0000 00e0 0000 0004  
00001200 0000 0004 0000 0003 0000 0114 1000 0114  
00001400 1000 0114 0000 0015 0000 0015 0000 0004  
00001600 0000 0001 7000 0002 0000 0130 1000 0130  
00002000 1000 0130 0000 0080 0000 0080 0000 0004  
00002200 0000 0008 7000 0000 0000 01b0 1000 01b0  
00002400 1000 01b0 0000 0018 0000 0018 0000 0004  
00002600 0000 0004 0000 0002 0000 01c8 1000 01c8  
00003000 1000 01c8 0000 0108 0000 0108 0000 0004  
00003200 0000 0004 0000 0001 0000 0000 1000 0000  
00003400 1000 0000 0000 3000 0000 3000 0000 0005  
.....
```

C 언어 소스

Assembly 소스

Compiler

Assembler

기계어 코드



Futuristic Innovator

京福大學校

KYUNGBOK UNIVERSITY



Preprocessor



- Source Program을 번역하기 전에 미리 언어의 기능을 확장한 Source Program을 생성하는 System Program
- 예) JAVA의 import 문



Compiler



■ Input

- High-Level Language Code
- 예) C, Java 등

■ Output

- Assembly Language Code
- 예) Intel x86, MIPS 등

■ 수행 작업

- High-Level Language Code를 Pseudo Instruction으로 변환
- Pseudo Instruction은 Assembler는 이해하나 Machine는 이해하지 못하는 명령을 말함
- 예) `mov $s1, $s2 = or $s1, $s2, $zero`





Assembler



- Input

- Assembly Language Code

- Output

- 거의 완벽한 형태의 실행 코드(Nearly-complete image of executable code)

- 수행 작업

- 어셈블리 명령을 16진수의 기계어 코드로 변환
(Binary encoding of each instruction)
 - 서로 다른 바이너리 파일 간의 주소 참조(reference)가
확정되지 않은 상태(Missing linkages between code in
different files)



Linker



■ Input

- 하나 이상의 바이너리 파일

■ Output

- 실행 가능한 프로그램 파일

■ 수행 작업

- 서로 다른 바이너리 파일 간의 참조 주소(포인터)를 계산 및 설정 (Resolves references between files)
- 정적인 런타임 라이브러리를 결합 (Combines with static run-time libraries, e.g., code for malloc, printf)
- 일부 라이브러리는 동적으로 결합 (Some libraries are dynamically linked, linking occurs when program begins execution)



Loader



- Executable File은 보조 기억장치인 Disk에 저장되어 있음
- Executable File이 실행되면 Loader는 Executable File을 Memory로 적재(load)한 후 실행을 시작
- 실질적으로 Loader는 운영체제의 기능 중 하나임



Loader



■ Loader 작업 순서

- Executable File의 헤더(header)를 읽어 필요한 Memory 용량을 계산
- Memory를 할당(allocate) 한 후, Program을 Memory에 적재
- main() 메소드에서 사용할 수 있도록 Program 실행 인자를 스택(stack)에 복사
- Program 시작 전에 CPU Register를 초기화 함
- Program의 main() 메소드를 시작
- Program이 종료하면 main() 메소드의 반환 값을 운영체제에 전달

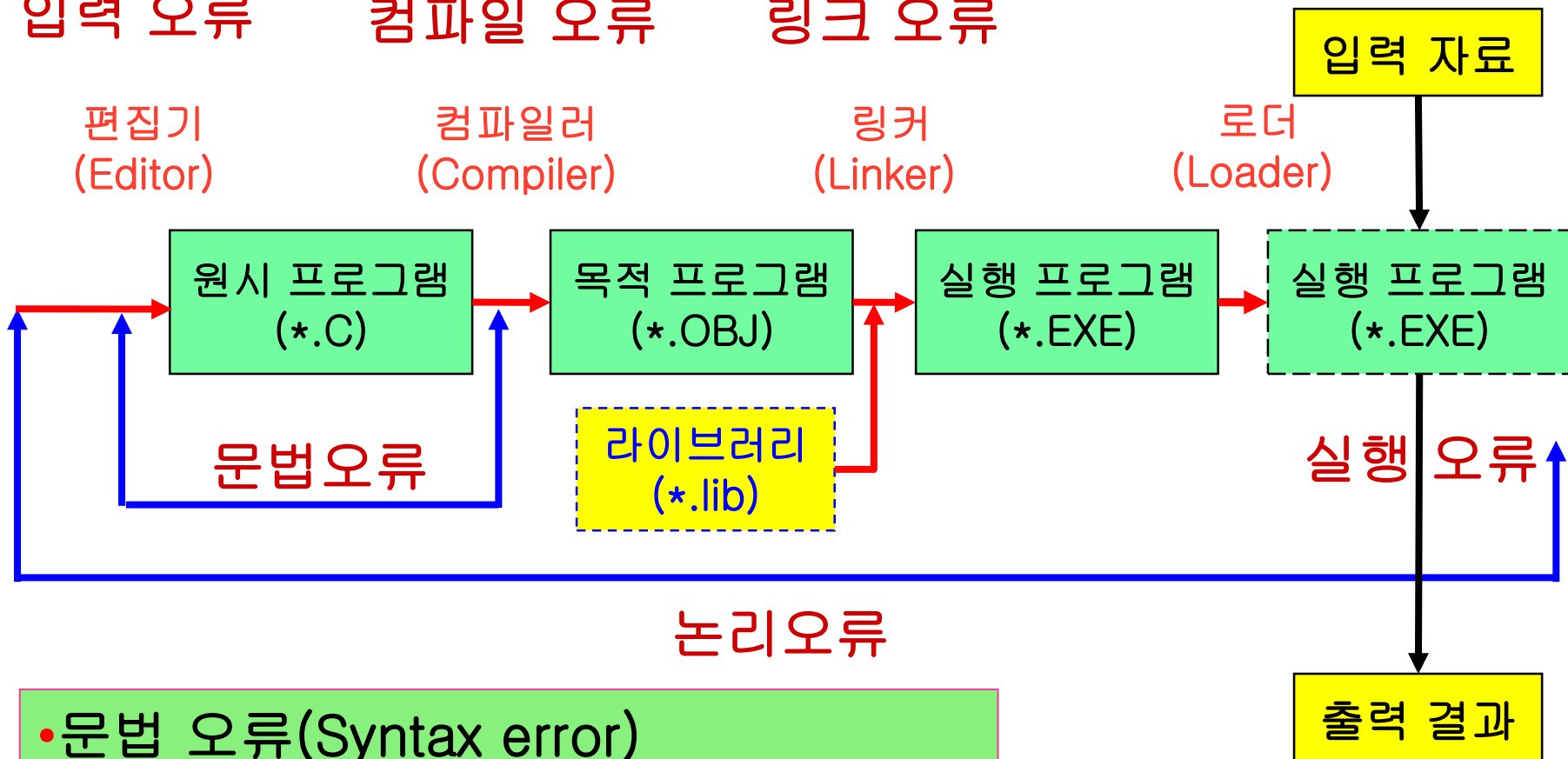


Syntax과 Semantic Error

입력 오류

컴파일 오류

링크 오류



- 문법 오류 (Syntax error)
 - 컴파일러의 오류 체크
- 논리 오류 (Semantic error)



Program Error



■ Compile Error

- Compile 시에 발견되는 Error

- Syntax Error

- 문법 규칙에 따르지 않아 발생

- Compiler가 Compile 시 Error 정보 출력

```
HelloWorld.java:13: ';' expected
      Console.println("Hello world")
                        ^
```

1 error

```
HelloWorld.java:13: cannot resolve symbol
symbol : method println (java.lang.String)
location: class com.otherwise.jurtle.Console
      Console.println("Hello world");
                        ^
```

1 error



Program Error



- Compile Error

- Semantic Error

- 문법 규칙에 모두 맞지만 계산식에서 호환이 되지 않는 서로 다른 타입의 Data가 포함된 계산이나 초기화되지 않은 변수 값을 출력하려고 할 때 발생

```
public void runTurtle() {  
    int j;  
    Console.println(j);  
}
```

Test.java:12: variable j might not have been initialized
 Console.println(j);
 ^



1 error



Program Error

■ Compile Error

■ Cascading Errors

- Syntax 또는 Semantic Error가 발생하는 데 있어서 Compiler가 확실하게 하나의 Error로서 판단하지 못하고 애매모호한 경우 여러 가능성 있는 Error를 모두 리스트 함

```
fo ( int i = 0; i < 4; i++ ) {  
    forward( 60 );  
    right( 90 );  
}
```

실제 Error는 for 키워드를
fo로 잘못 typing한 것

```
ASimpleSquare.java:24: '.class' expected  
fo ( int i = 0; i < 4; i++ )  
  ^  
  
ASimpleSquare.java:24: ')' expected  
fo ( int i = 0; i < 4; i++ )  
  ^  
  
ASimpleSquare.java:24: not a statement  
fo ( int i = 0; i < 4; i++ )  
  ^  
  
ASimpleSquare.java:24: ';' expected  
fo ( int i = 0; i < 4; i++ )  
  ^  
  
ASimpleSquare.java:24: unexpected type  
required: value  
found   : class  
fo ( int i = 0; i < 4; i++ )  
  ^  
  
ASimpleSquare.java:24: cannot resolve symbol  
symbol  : variable i  
location: class ASimpleSquare  
fo ( int i = 0; i < 4; i++ )  
  ^  
  
6 errors
```



Program Error



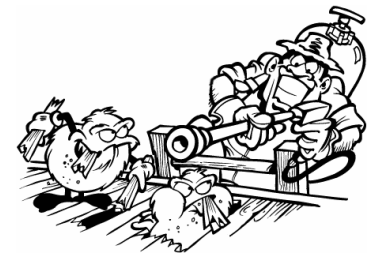
- Runtime Error

- Runtime Error는 Program이 실행되는 도중에 발생하는 Error

- 예) 0으로 나누려고 시도

- JAVA에서는 많은 Runtime Error들을 Exception을 사용하여 처리

- 예) Program상에서 만약 0으로 나누는 시도가 발견되면 관련된 Exception 객체를 생성하여 던져 버림





Program Error

■ Logical Error

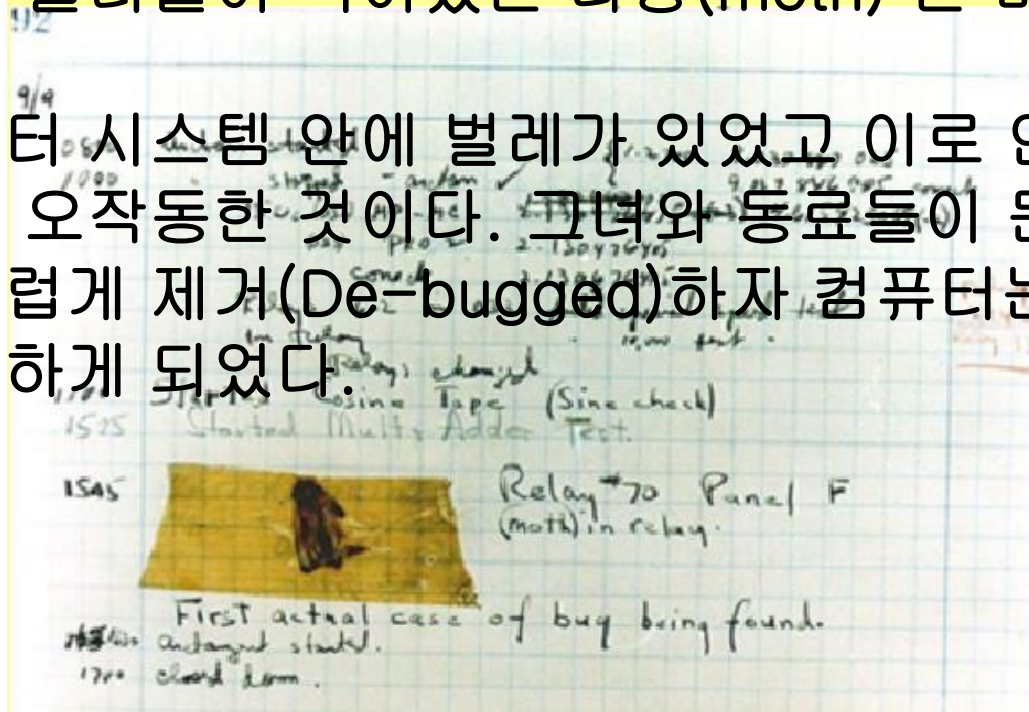
- Compile되고 실행도 잘 되지만 의도하지 않는 결과를 발생하는 경우
 - 예) 값이 예상과는 다르게 잘못 계산되는 경우
- Debugging : Logical Error를 찾고 고치는 과정





Debugging

- 1947년 여름, 당시 그레이스 호퍼는 하버드 대학에서 마크 II 컴퓨터를 이용해 연구 중이었는데 컴퓨터가 자주 고장을 일으켰다고 한다. 호퍼와 동료들은 고장의 원인을 찾기 위해 컴퓨터 내부를 들여다보며 조사하던 중, 릴레이(relay)의 접점 사이에 끼어 들러붙어 죽어있는 나방(moth) 한 마리를 발견하게 된다.
- 실제로 컴퓨터 시스템 안에 벌레가 있었고 이로 인한 누전으로 컴퓨터가 오작동한 것이다. 그녀와 동료들이 문제가 된 나방을 조심스럽게 제거(De-bugged)하자 컴퓨터는 다시 정상적으로 작동하게 되었다.





Debugging 방법

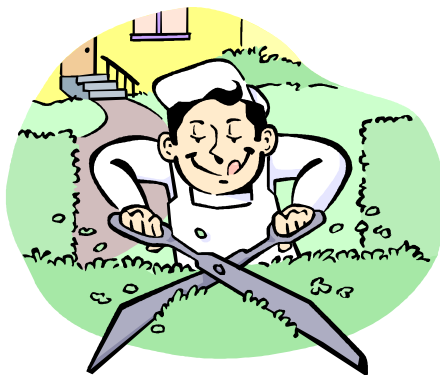


1. Error Message에 지정된 행(line)에서부터 그 이전 행으로 이동하면서 Error의 원인을 찾자
2. 많은 Error Message 중에 제일 처음의 Error를 우선적으로 수정하여 다시 Compile 하면 2차로 발생한 Error Message들은 해결할 수 있다
3. Warning Message도 반드시 수정



Software 유지 보수

- Software 유지 보수가 필요한 이유
 - Debugging 후에도 Bug가 남아 있을 수 있기 때문
 - Software가 개발된 다음에 사용자의 요구가 추가될 수 있기 때문
- 유지 보수 비용이 전체 비용의 50% 이상을 차지





Program 연습



■ 잘하는 방법

- 항상 먼저 Algorithm을 생각할 것
- 정답을 보지 않고 끝까지 스스로...
(밤을 새워서라도 끝까지.....)
- 여러 가지 예제를 많이, 계속해서 풀어본다
- 문제를 수정해서 다시 작성해본다
- 스스로 푼 방법을 책의 예제 또는 친구의 방법과 비교해 본다
- 사람과 Computer의 차이점을 이해할 것!

■ 잘못하는 방법

- 책에 나오는 Program을 Typing만 열심히...
- 눈(eye)과 머리(head)로만 Programming?
- 안되면 쉽게 단념





Program 설계 시 격언



- 최소 당황 법칙
 - 사용자를 당황하게 하는 일을 최소화
- 사용자 인터페이스의 일관성
 - 가능한 간단하고 일관성 있게 만듦
- 사용자에게 도움말을 많이 제공
- 에러가 발생하면 “에러”라는 명확한 정보 제공
- 주석! 주석! 주석!
- “KISS” 원칙(Keep It Simple, Stupid)



Program을 작성할 때 꼭 필요한 내용들

- Program을 작성할 때는 합리적 순서를 따라야 한다
- 관례에 따라 Coding 형식을 지켜야 한다
- Comment은 선택이 아니라 필수다
- 식별자는 의미를 부여하여 만든다

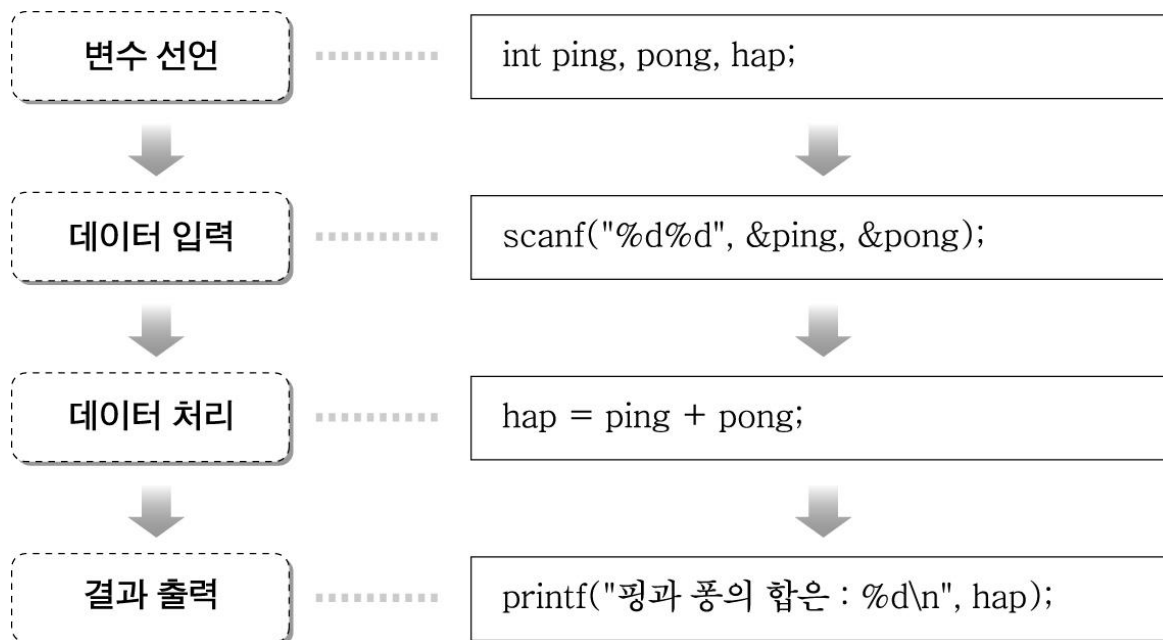


Program을 작성할 때 꼭 필요한 내용들

■ Program을 작성하는 순서

- Data를 입력하기 전에 반드시 입력할 Data를 저장할 기억공간이 있어야 한다. 즉, **변수선언이 입력문 전에 있어야 한다!!**

■ 일반적인 Program의 작성 순서





Program을 작성할 때 꼭 필요한 내용들

■ Coding 형식

■ Program의 문장을 구분하는 것은 세미콜론(;)

■ 한 줄에 여러 문장을 작성해도 되고 한 문장을 여러 줄에 작성해도 된다(free format)

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

같은 프로그램?

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

■ Program은 알아보기 쉽게 관례에 따라 형식을 지킨다!



중간 점검



1. 컴퓨터가 직접 이해할 수 있는 단 하나의 언어는 기계어 이다.
2. 프로그래밍 언어를 기계어로 변환시켜주는 프로그램을 컴파일러 한다.
3. 우리는 왜 기계어를 사용해서 프로그램하지 않는가?
힘들어서



시작하면서



“코딩(Coding)하기 전에 계획(Plan)하고 설계(Design)”

상당한 시간 절약
문제점이 없는 프로그램
소프트웨어 개발을 가능하게 함



Futuristic Innovator

京福大學校
KYUNGBOK UNIVERSITY