



제01장

프로그래밍언어 개요



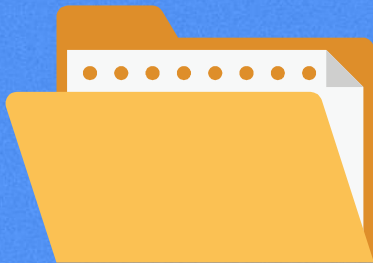


학습목표

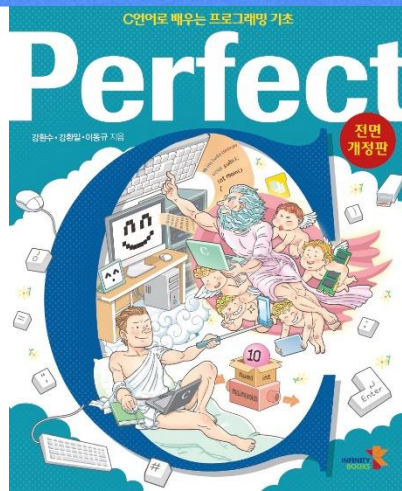
- ▶ 프로그램이 무엇이며, 컴퓨터의 소프트웨어를 이해하고 소프트웨어를 개발하는 프로그래밍 언어를 설명할 수 있다.
 - 일반 용어인 프로그램과 컴퓨터 프로그램의 유사점
 - 하드웨어와 소프트웨어
 - 기계어와 어셈블리어
 - 저급언어와 고급언어
 - 컴파일러와 어셈블러
- ▶ 프로그램이 무엇이며, C언어의 중요성과 특징을 이해하고 설명할 수 있다.
 - C 언어의 개발과 역사
 - C 언어의 특징과 중요성
- ▶ 컴퓨터의 자료표현 방법과 논리 및 문자를 이해하고 설명할 수 있다.
 - 이진수의 표현방법과 정보의 단위
 - 논리와 문자 표현
- ▶ 소프트웨어 개발과정과 알고리즘의 표현 방법을 이해하고 설명할 수 있다.
 - 알고리즘의 정의와 기술 방법
 - 소프트웨어 개발 절차
- ▶ 포트란에서부터 스크래치까지 다양한 프로그래밍 언어를 이해하고 설명할 수 있다.
 - 60~70년 초기에 개발된 프로그래밍 언어
 - 객체지향 프로그래밍 언어와 비주얼 프로그래밍 언어

학습목차

- 01.1 '프로그램'이 무엇일까?
- 01.2 언어의 계층과 번역
- 01.3 왜 C 언어를 배워야 할까?
- 01.4 프로그래밍의 자료 표현
- 01.5 소프트웨어 개발
- 01.6 다양한 '프로그래밍 언어'



01. '프로그램'이 무엇일까?



우리 주의에서 일상이 된 '프로그램'

프로그램

- 스마트폰과 노트북, 혹은 데스크탑 컴퓨터에서 특정 작업을 위해 컴퓨터를 작동시키는 것
- 특정 목적의 작업을 수행하기 위한 관련 파일의 모임



우리 주의에서 일상이 된 '프로그램'

생활에서의 프로그램

- 텔레비전의 방송에서도 예능 프로그램, 오디션 프로그램 등 프로그램이라는 용어를 자주 사용
- 연극이나 방송 따위의 진행 차례나 진행 목록
- 프로그램이란 용어의 공통적 의미
- 특정한 목적을 수행하기 위한 이미 정해놓은 순서적인 계획이나 절차를 의미



프로그램은 작업의 진행 순서를 정해놓은 목록

1. 순서가 정해져 있다.

2. 이미 만들어져 있다.

3. 특정 시점에서 취해야 할 행동이 정해져 있다.

우리 주의에서 일상이 된 '프로그램'

정보기술에서의 프로그램

- 특정 작업을 수행하기 위하여 그 처리 방법과 순서를 기술한 명령어와 자료로 구성
- 컴퓨터에게 지시할 일련의 처리 작업 내용
- 사용자의 프로그램 조작에 따라 컴퓨터에게 적절한 명령을 지시하여 프로그램이 실행



1. 알람 시각 설정: 아침 7시 30에 알람을 올려라.
2. 소리 설정: 알람 음은 듣기 좋은 '물방울 소리'로 지정하자.
3. 스누즈 기능 설정: 처음에는 작은 소리였다가 점점 소리를 크게 하라.
4. 정지 조건 설정: 버튼을 누르면 알람 정지해라.

프로그래머와 프로그래밍 언어

프로그래밍 언어

- 프로그램을 개발하기 위해 사용하는 언어
- 사람과 컴퓨터가 서로 의사 교환을 하기 위한 언어
- 사람이 컴퓨터에게 지시할 명령어를 기술하기 위하여 만들어진 언어
- 매우 다양한 언어가 개발

응용프로그램: 기상정보 앱



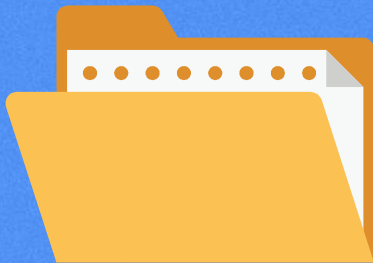
자바로 개발

C로 개발

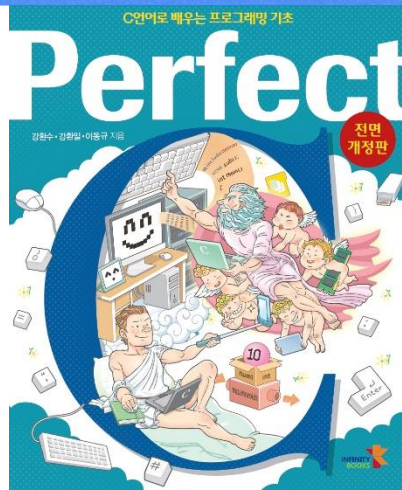


운영체제: 윈도우10





02. 언어의 계층과 번역



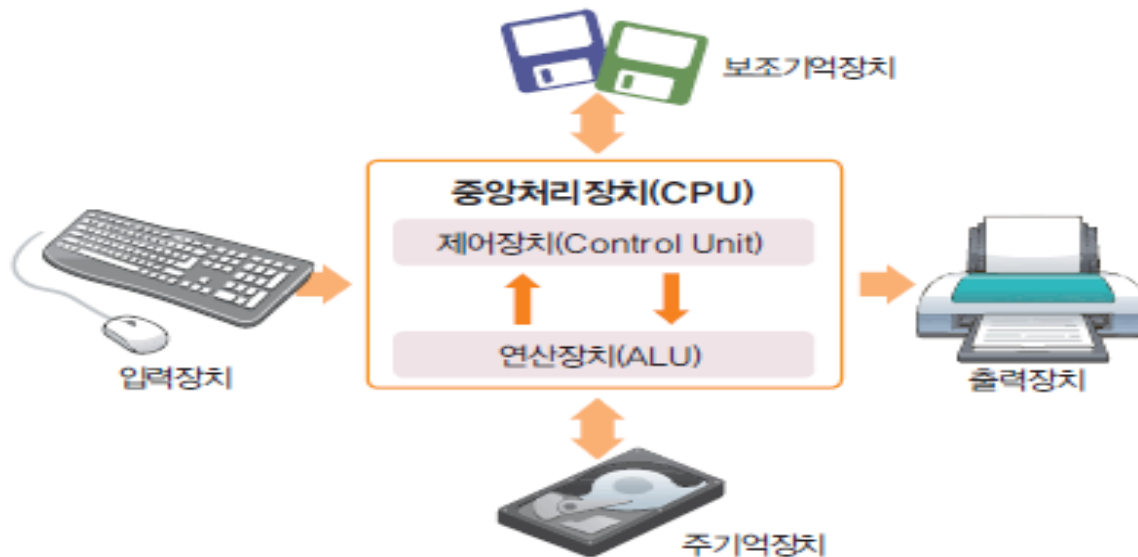
컴퓨터는 단순한 기계덩어리

컴퓨터와 하드웨어

- 영어 단어 compute와 er의 조합으로 '계산하는 기계', 즉 컴퓨터는 '전자적으로 계산을 수행하는 장치'
- 하드웨어(hardware)와 이 하드웨어를 작동하게 하는 소프트웨어(software)로 구성

중앙처리장치

- 연산을 수행하는 연산장치(ALU: Arithmetic Logic Unit)와 연산을 제어하는 제어장치(control unit)로 구성
- 중앙처리장치의 칩이 프로세서(processor)



컴퓨터는 단순한 기계덩어리

소프트웨어

- 컴퓨터가 특정 작업을 수행할 수 있도록 해주는 전자적인 명령어 집합
 - 컴퓨터의 하드웨어가 해야 할 작업 내용을 지시
- 소프트웨어는 크게 응용 소프트웨어와 시스템 소프트웨어로 나뉨
- 시스템 소프트웨어
 - 컴퓨터가 잘 작동하도록 도와주는 기본 소프트웨어
- 응용 소프트웨어
 - 문서 작성이나 인터넷검색,
 - 게임 하기, 동영상 보기 등과 같은 특정 업무에 활용되는 소프트웨어

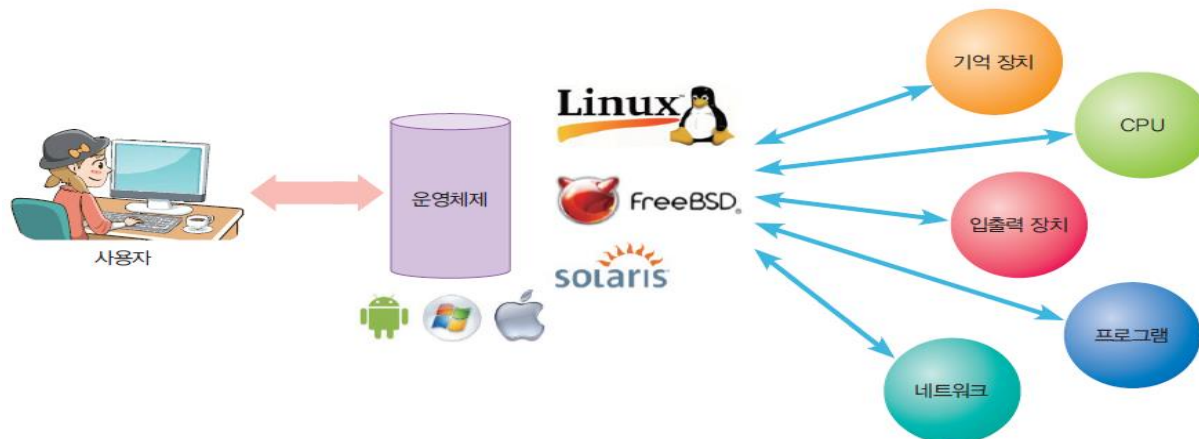


운영체제와 유틸리티 프로그램

시스템 소프트웨어

• 운영체제

- 컴퓨터 하드웨어 장치의 전반적인 작동을 제어하고 조정하며, 사용자가 최대한 컴퓨터를 효율적으로 사용할 수 있도록 돕는 시스템 프로그램
- 하드웨어와 응용프로그램간의 인터페이스 역할을 하면서 CPU, 주기억장치, 입출력장치 등의 컴퓨터 자원과 함께 다양한 프로그램과 네트워크 등을 관리
- 유닉스(Unix), 리눅스(Linux), 윈도우즈(Windows), 맥(Mac) OS X 등
- 스마트폰과 같은 스마트 기기를 위한 운영체제로
 - iOS, 안드로이드(Android), 파이어폭스(Firefox), 윈도우폰(Window Phone) 등

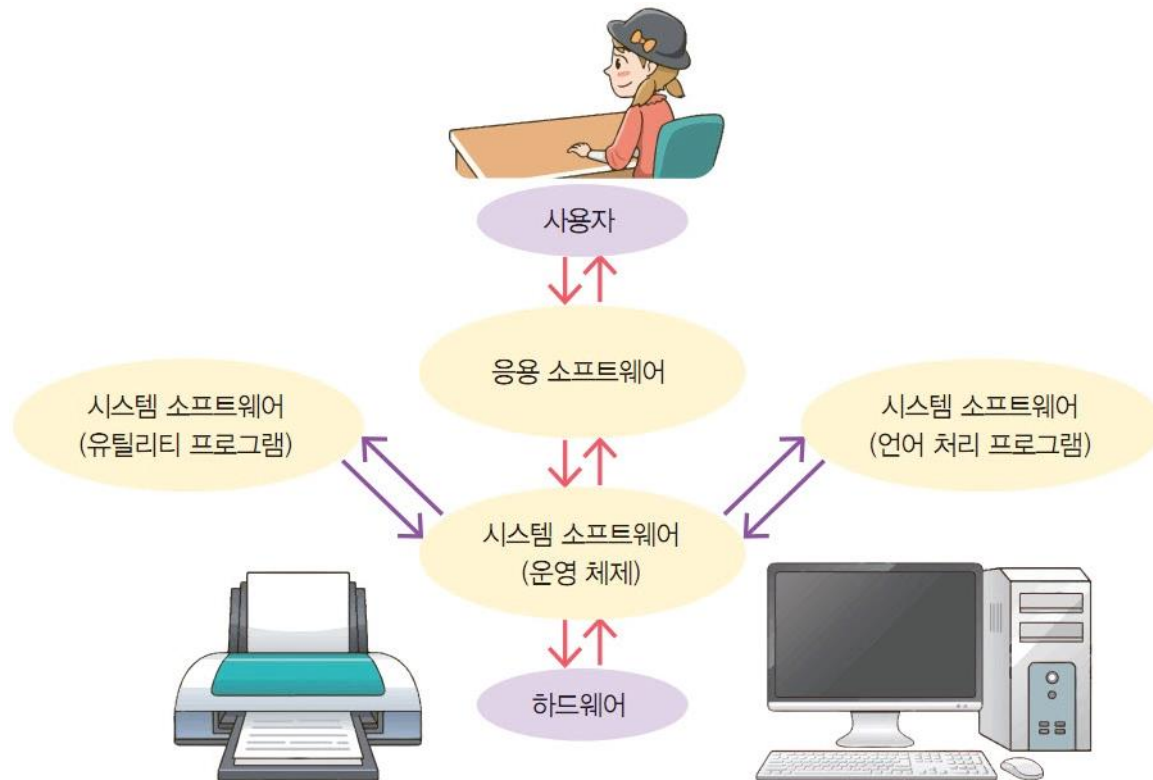


운영체제와 유틸리티 프로그램

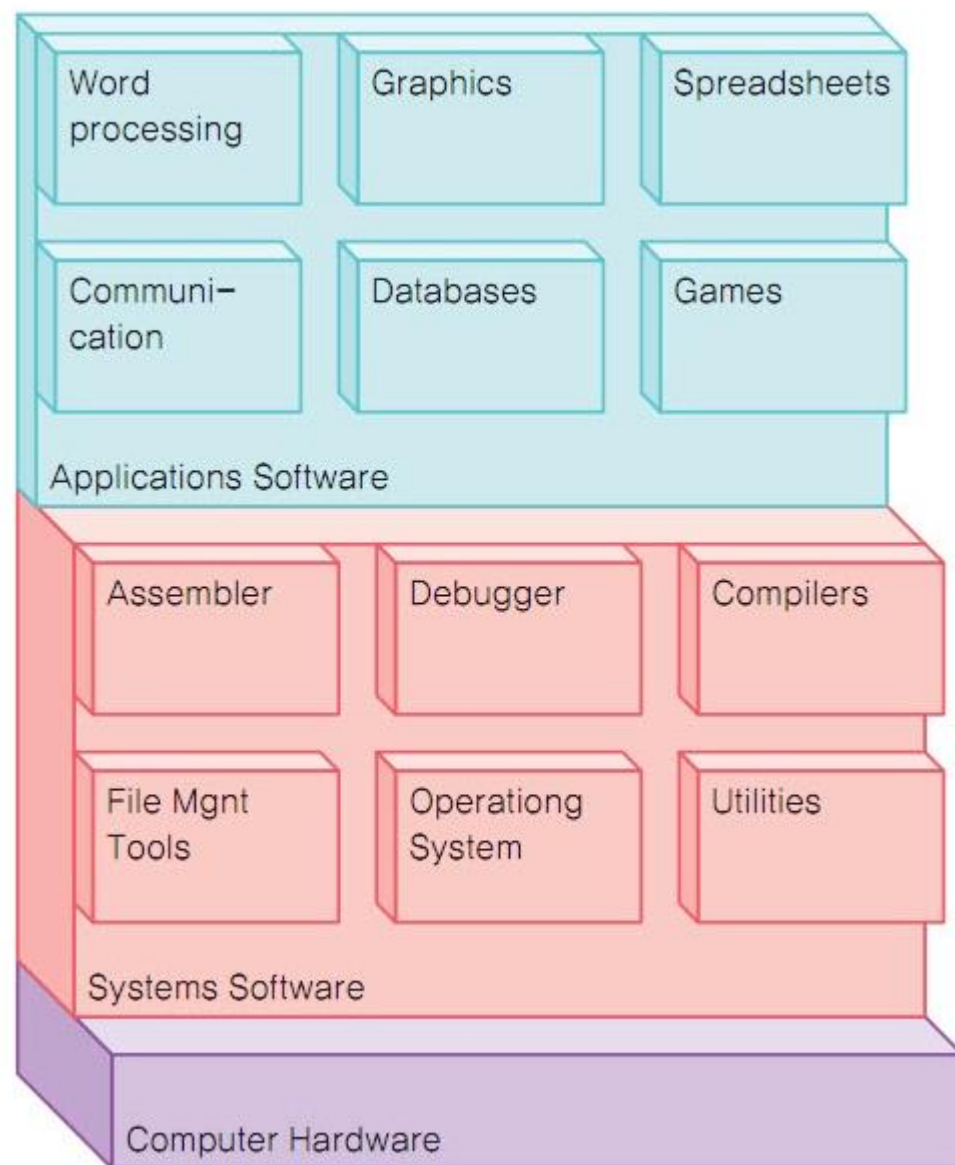
시스템 소프트웨어

- 유틸리티 프로그램

- 운영체제를 돕고 컴퓨터 시스템이 원활하게 작동하도록 도움을 줌

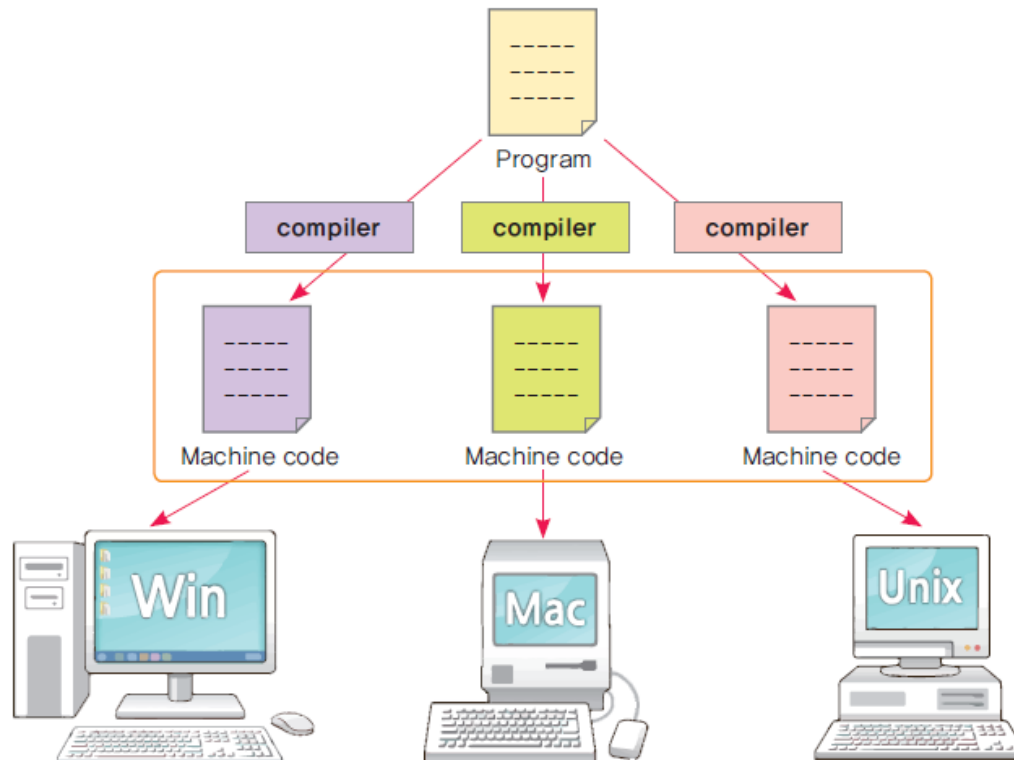


운영체제와 유틸리티 프로그램



기계어와 컴파일러

- **컴퓨터는 기계어라는 것만을 인식**
 - 즉 전기의 흐름을 표현하는 1과 흐르지 않음을 의미하는 0으로 표현되는 기계어(machine language)가 바로 컴퓨터가 유일하게 바로 인식하는 언어
- **통역사인 컴파일러(compiler)가 필요**
 - 프로그래밍 언어를 사용하는 프로그래머와 기계어를 사용하는 컴퓨터가 서로 의사 교환을 하려면 통역사와 같은 번역기가 필요



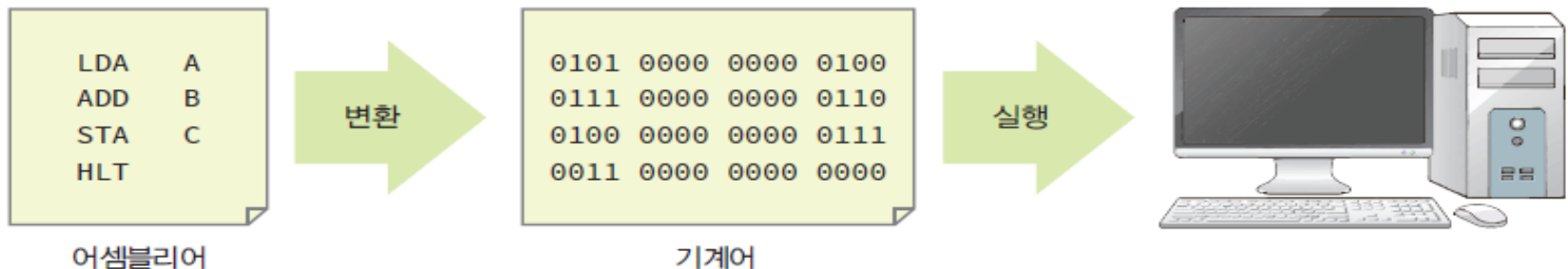
어셈블리어

- 어셈블리어(assembly language)

- 기계어를 프로그래머(programmer)인 사람이 좀 더 이해하기 쉬운 기호 형태로 일대일 대응시킨 프로그래밍 언어
 - CPU마다 제각각 다름
- 어셈블리 언어는 기계어보다는 프로그래밍이 훨씬 용이
 - 문장과 연산코드가 일대일 대응되기 때문에 기계어처럼 하드웨어 장치에 대한 강력한 통제 역시 가능하다는 장점

- 어셈블리 명령어 예

- LDA(Load Address), ADD(ADD), STA(STore Address) 등
 - 명령어를 기호화한 것을 니모닉(mnemonic)이라 함
- 연산식 $C = A + B$ 을 처리하는 프로그램을 기계어와 어셈블리어



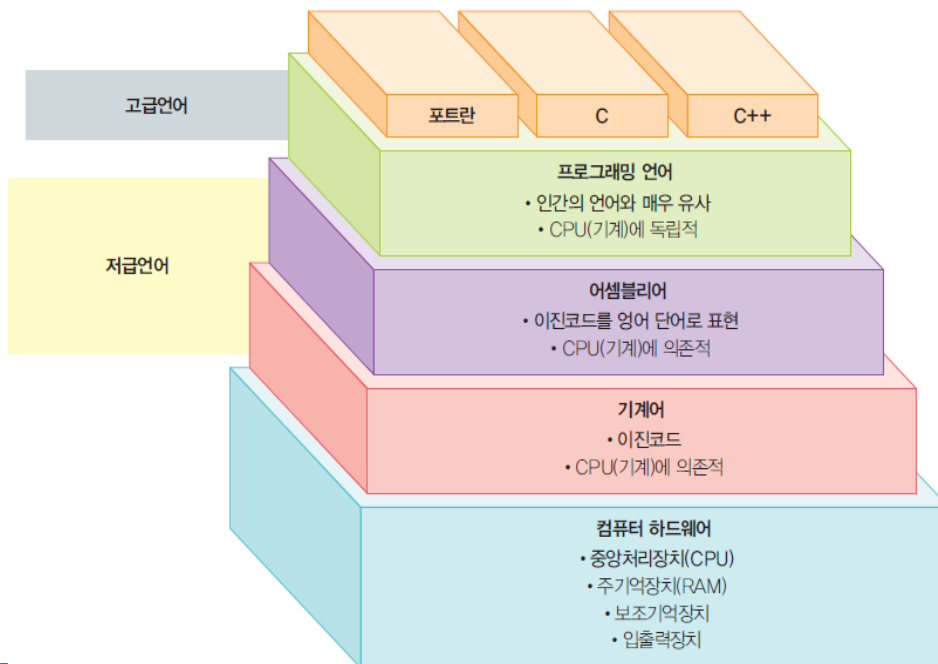
저급언어와 고급언어

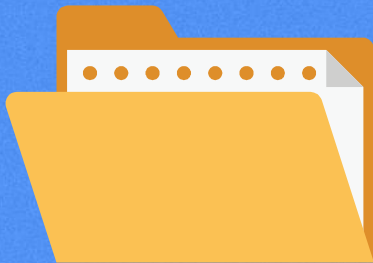
저급언어

- 컴퓨터의 중앙처리장치(CPU)에 적합하게 만든 기계어와 어셈블리 언어를 저급언어(Low Level Language)
- 저급언어는 컴퓨터의 CPU에 따라 달라지며, 특정한 CPU를 기반으로 만들어진 언어

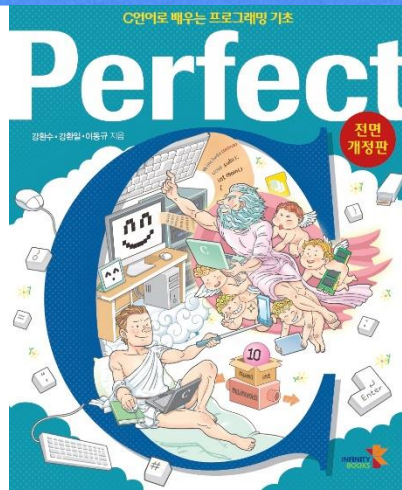
고급언어

- 컴퓨터의 CPU에 의존하지 않고 우리 사람이 보다 쉽게 이해할 수 있도록 만들어진 언어를 고급언어(High Level Language)
- 고급언어인 프로그래밍 언어로는 바로 우리가 배우고 있는 C 언어를 비롯하여 포트란, 파스칼, 베이직, C++, 자바, 파이썬 등 인간의 언어만큼이나 매우 다양





03. 왜 C 언어를 배워야 할까?



C 언어 역사: B언어에서 발전된 유닉스 개발언어

데니스 리치

- 1972년 미국전신전화국(AT&T)의 벨 연구소(Bell Lab)에 근무하던 데니스 리치는 시스템 PDP-11에서 운용되는 운영체제인 유닉스(Unix)를 개발하기 위해 C 언어를 개발
- 어셈블리 언어 정도의 속도를 내며, 좀 더 쉽고, 서로 다른 CPU에서도 작동되는 프로그래밍 언어가 필요

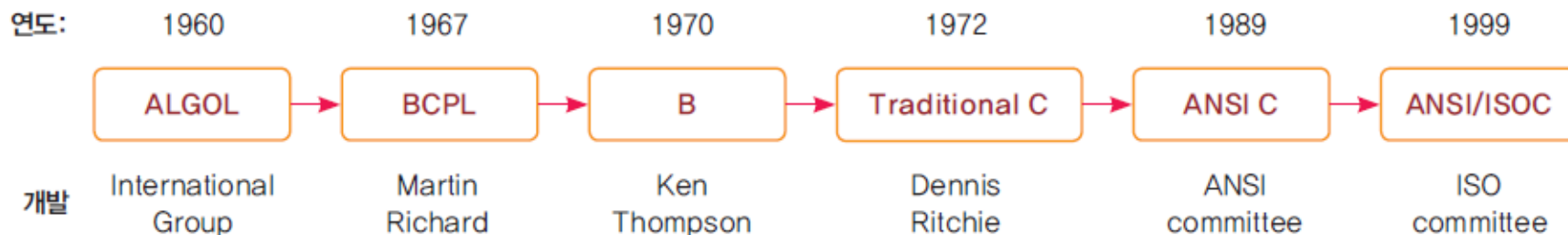
Dennis Ritchie
1941-2011



C 언어 역사: B언어에서 발전된 유닉스 개발언어

B 언어에서 발전

- 켄 톰슨이 1970년 개발한 B 언어에서 유래
- B언어 1970년 BCPL 프로그래밍 언어에 기반을 두고 개발된 언어
- BCPL(Basic Combined Programming Language)은 캠브릿지 대학의 마틴 리차드(Martin Richards)가 1969년 개발한 프로그래밍 언어로 CPL(Combined Programming Language) 언어에서 발전
- 1960년 초에 개발된 CPL은 1960년에 개발된 Algol 60으로부터 많은 영향을 받은 언어



C 언어의 특징

1.

절차지향 언어

- 함수 중심으로 구현되는 절차지향 언어(procedural language)
- 시간의 흐름에 따라 정해진 절차를 실행한다는 의미
- 문제의 해결 순서와 절차의 표현과 해결이 쉽도록 설계된 프로그램 언어

2.

간편하고 효율적인 언어

- 크기도 작으며, 메모리도 적게 효율적으로 사용하여 실행 속도가 빠르다는 장점

3.

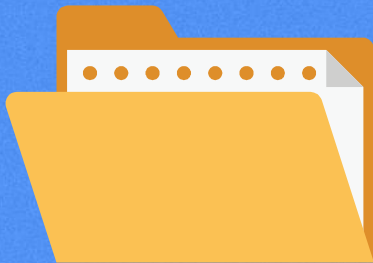
이식성이 좋은 언어

- C로 작성된 소스는 별다른 수정 없이 다양한 운영체제의 여러 플랫폼에서 제공되는 컴파일러로 번역(compile)해 실행
- 다양한 CPU와 플랫폼의 컴파일러를 지원

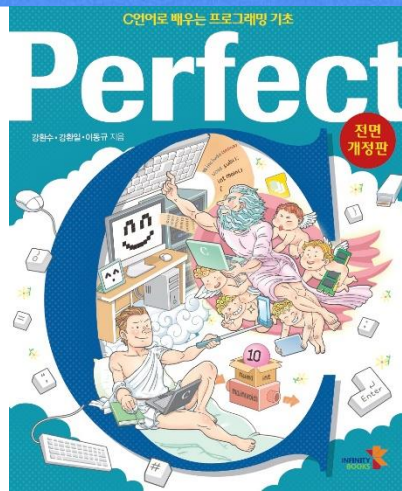
4.

다소 어렵다는 단점





04. 프로그래밍의 자료 표현



우리에게 친숙한 십진수

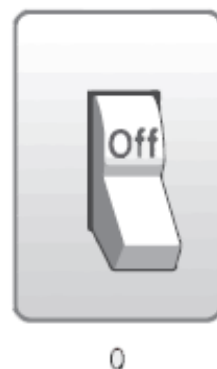
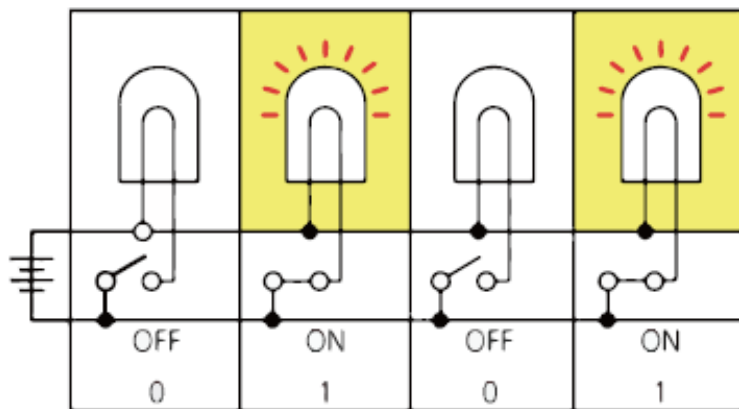
- 십이라는 것을 기수(base)

- 수에서 하나의 자릿수(digits)에 사용하는 숫자가 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 까지 열 개이므로 십진수
- 십진수 5319
 - $1000(10^3)$ 인 것이 5개, $100(10^2)$ 인 것이 3개, $10(10^1)$ 인 것이 1개, 마지막으로 $1(10^0)$ 인 것이 9개 모인 수

내부 자료표현 방법: 0과 1로 표현되는 이진수 체계

온과 오프

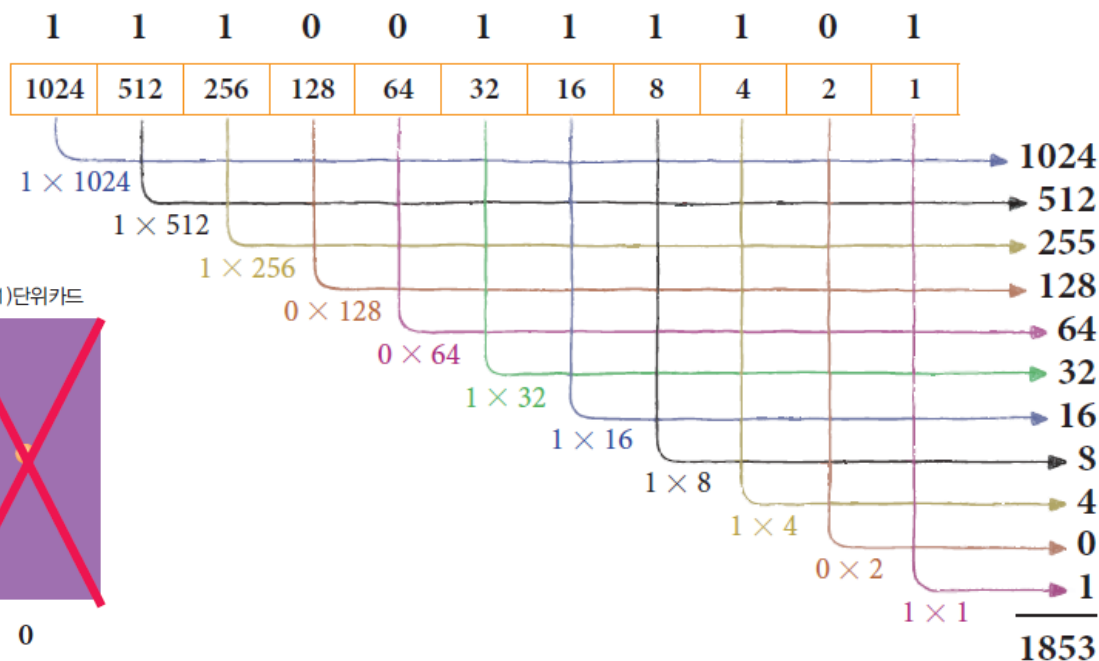
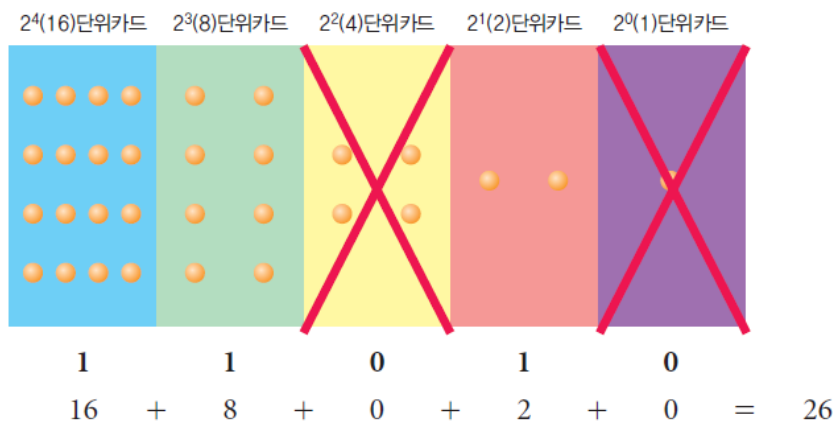
- 전기가 흐를 경우 '참'을 의미하는 '1',
흐르지 않을 경우 '거짓'의 '0'으로 표현
- 컴퓨터는 논리의 조합이 간단하고 내부에 사용되는 소자의 특성상
편리하기 때문에 이진법을 사용하는 것이 가장 합리적이고 효율적인
방식



내부 자료표현 방법: 0과 1로 표현되는 이진수 체계

이진수의 이해

- 수의 자릿수에 사용할 수 있는 숫자가 0과 1, 2개 이므로 이진수
- 11010은 1, 2, 4, 8, 16 자릿수의 카드로 표현
- 이진수 11100111101은 1853



정보의 표현, 비트와 바이트

비트

- 가장 작은 기본 정보 단위
- 즉 전기의 흐름 상태인 온(on)과 오프(off)를 표현하는 단위가 비트로 1과 0인 이진수로 표현이 가능
- 비트(bit)는 Binary digit의 합성어

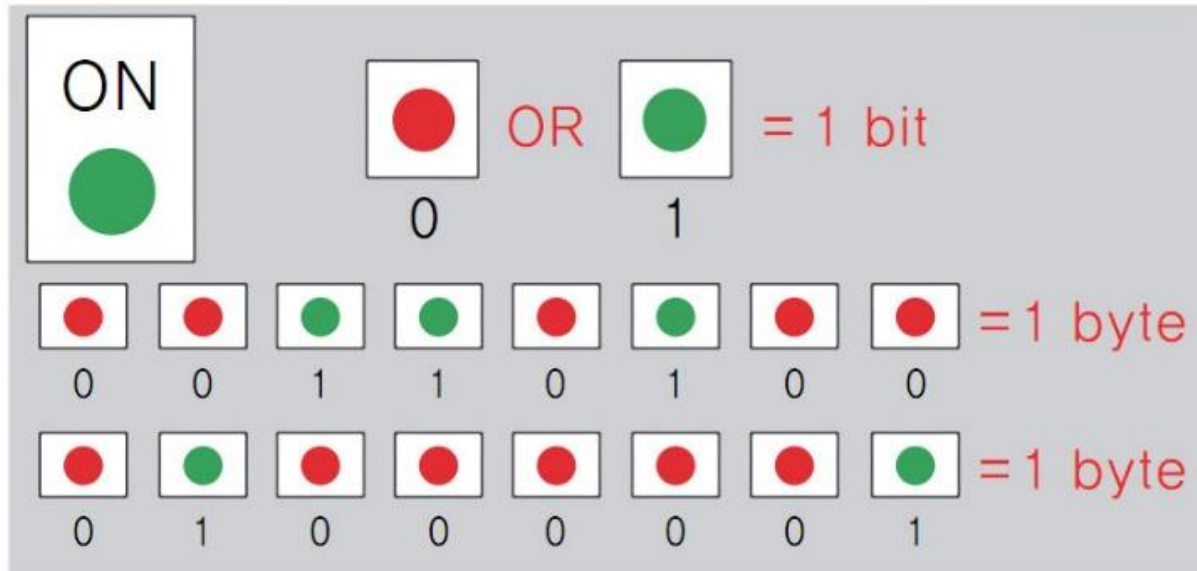


그림1-36 비트와 바이트의 정보인 00110100과 01000001의 이해

정보의 표현, 비트와 바이트

바이트

- 바이트가 정보 용량의 단위, 킬로, 메가, 기가, 테라 등은 그 크기를 표현
- 킬로(Kilo)는 2^{10} 을 의미하며 1024

단위	약자	표현	바이트(byte)	근접
1 bit	1 b (lower case)	0 또는 1		
1 Nibble		4 bits	½ byte	
1 Byte	1B (upper case)	8 bits 또는 2 Nibbles 또는 2^3 bits	1 byte	
1 Kilobyte	1 KB	2^{10} bytes	1,024 byte	1 thousand bytes
1 Megabyte	1 MB	$(2^{10})^2$ bytes	1,048,576 byte	1 million bytes
1 Gigabyte	1 GB	$(2^{10})^3$ bytes	1,073,741,824 byte	1 trillion bytes
1 Terabyte	1 TB	$(2^{10})^4$ bytes	1,099,511,627,776 byte	1 quadrillion bytes
1 Petabyte	1 PB	$(2^{10})^5$ bytes	1,125,899,906,842,624 byte	1 quintillion bytes
1 Exabyte	1 EB	$(2^{10})^6$ bytes	1,152,921,504,606,846,976 byte	1 sextillion bytes
1 Zeatbyte	1 ZB	$(2^{10})^7$ bytes	1,180,591,620,717,411,303,424 bytes	1 septillion bytes

그림 1-38 저장 용량 단위

정보의 표현, 비트와 바이트

바이트와 워드

- 8개의 비트가 모인 정보 단위를 바이트(byte)
- 비트는 총 $2^8=256$ 가지의 정보 종류를 저장
- 일반적으로 바이트가 4개, 8개 모이면 워드(word)
- 시스템마다 그 크기는 다를 수 있음

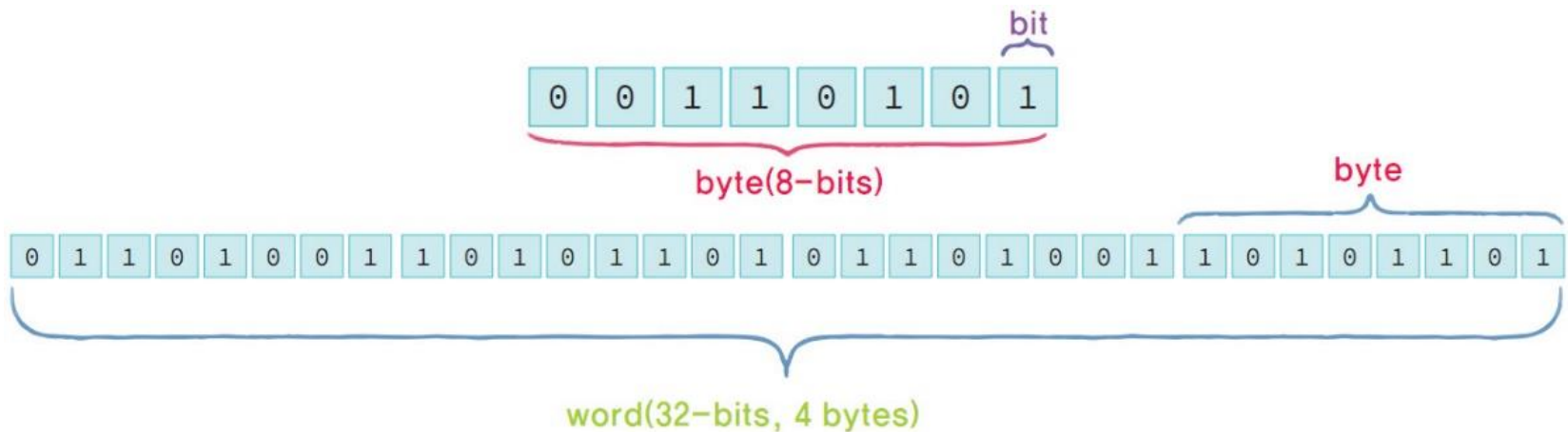


그림 1-37 비트와 바이트, 워드

논리 표현과 연산

- 논리값

- 0과 1을 각각 거짓과 참으로 표현 가능
- 부울 대수(Boolean Algebra)
 - 영국의 수학자 조지 부울(George Boole)이 제창한 기호 논리학으로 컴퓨터가 정보를 처리하는 방식에 대하여 이론적인 배경을 제공

- 논리 연산

- 기본적으로 AND, OR 연산과 NOT 연산 제공

A	B	A and B
0	0	0
0	1	0
1	0	0
1	1	1

AND

A	B	A or B
0	0	0
0	1	1
1	0	1
1	1	1

OR

A	not B
0	1
1	0

NOT

A	B	A nand B
0	0	1
0	1	1
1	0	1
1	1	0

NAND

A	B	A nor B
0	0	1
0	1	0
1	0	0
1	1	0

NOR

A	B	A xor B
0	0	0
0	1	1
1	0	1
1	1	0

XOR

그림 1-39 논리 연산 AND, OR, NOT

문자 인코딩 방식: 아스키코드와 유니코드

- 문자를 하나의 숫자인 코드로 지정하여 처리하는 방식
 - 문자 'A': 0100 0001

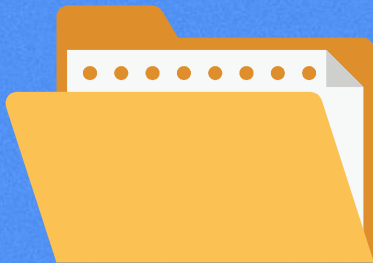
아스키코드

- 아스키(ASCII: American Standard Code for Information Interchange) 코드는 1967년에 표준으로 제정
- 1986년에 마지막으로 개정
- 아스키는 초창기에 7비트 인코딩(8비트중 7비트만 데이터 비트로 사용)
 - 33개의 출력 불가능한 제어문자들과 공백을 비롯한 95개의 출력 가능한 문자들로 이루어져 총 128개의 코드로 구성
- 8비트 인코딩을 사용하도록 확장
 - 아스키코드값이 주로 그래픽에 관련된 문자와 선 그리기에 관련된 문자가 추가되어 256개로 확장

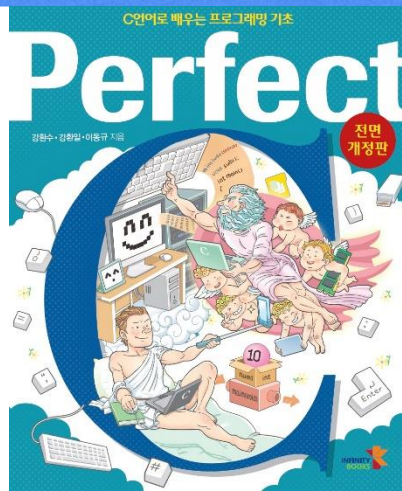
문자 인코딩 방식: 아스키코드와 유니코드

유니코드

- 유니코드는 전 세계의 모든 문자를 컴퓨터에서 일관되게 표현하고 다룰 수 있도록 설계된 산업 표준
- 유니코드 협회(Unicode Consortium)가 제정하여 1991년 버전 1.0이 발표
- 동양권의 컴퓨터 관련 시장을 쉽게 접근하기 위해서도 미국 등의 유수의 S/W, H/W업체에게 문자 코드 문제는 가장 시급하게 해결되어야 할 걸림돌
- 전 세계의 문자를 모두 표현하기 위해 2바이트 즉, 16비트로 확장된 코드 체계가 유니코드
- 1996년 65,536자의 코드영역을 언어학적으로 분류
- 한글 완성자 11,172자의 한글과 중국, 일본을 포함해 세계 유수의 언어 문자를 배열해 만든 유니코드는 국제표준화기구(ISO: International Organization for Standardization)에 상정 확정
- 현재 계속 수정, 보완



05. 소프트웨어 개발



알고리즘

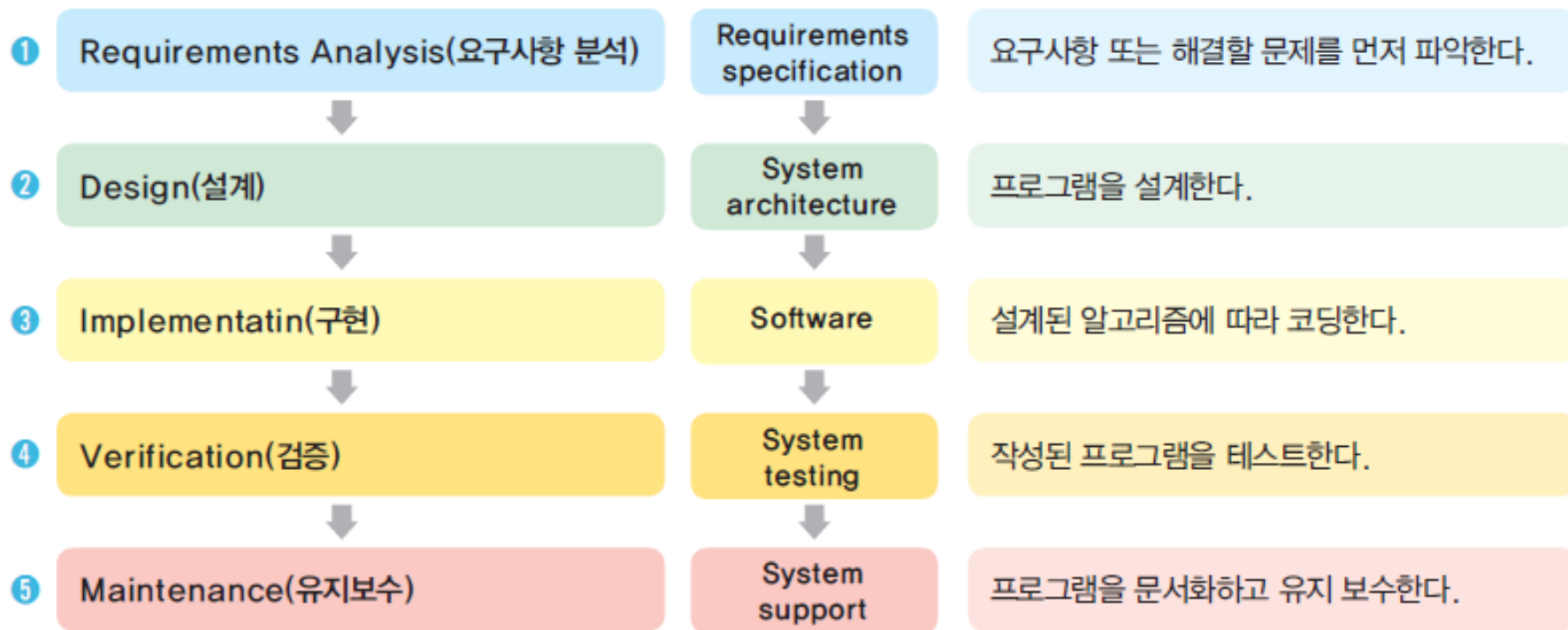
- 어떠한 문제를 해결하기 위한 절차나 방법으로 명확히 정의된 (well-defined) 유한 개의 규칙과 절차의 모임
- 프로그램에서 문제를 해결하기 위한 여러 명령을 단계적으로 알려주어야 하는데, 이것이 바로 '알고리즘'
- 컴퓨터 프로그램
 - 특정한 업무를 수행하기 위한 정교한 알고리즘들의 집합

소프트웨어 개발 방법을 연구하는 소프트웨어 공학

소프트웨어 공학(software engineering)

- 소프트웨어 공학이란 공학적 원리에 의하여 소프트웨어를 개발하는 학문
- 소프트웨어 개발과정인 분석, 설계, 개발, 검증, 유지보수 등 개발수명주기 전반에 걸친 계획·개발·검사·보수·관리, 방법론 등을 연구하는 분야

소프트웨어 개발 방법을 연구하는 소프트웨어 공학



설계 단계에서의 알고리즘 기술 방법

- 문제를 해결하기 위한 절차나 방법의 모임인 알고리즘
 - 우리가 사용하는 자연어 또는 흐름도(flow chart)나 의사코드(pseudo code) 등을 사용하여 표현











의사코드

- 의사코드는 슈도코드라고도 하며
- 특정 프로그래밍 언어의 문법을 따르지 않고 간결한 특정 언어로 코드를 흉내 내어 알고리즘을 써놓은 코드
- 의사코드는 다양한 스타일의 언어가 존재

설계 단계에서의 알고리즘 기술 방법

흐름도

- 표준화된 기호 및 도형으로 도식화하여 데이터의 흐름과 수행되는 연산들의 순서를 표현하는 방법

기호	기능	기호	기능
	터미널 순서도의 시작과 끝을 표시		처리 각종 연산, 데이터 이동 등을 처리
	판단 여러 가지 경로 중 하나의 경로 선택을 표시		입·출력 데이터의 입력 및 출력 표시
	흐름선 처리간의 연결 기능을 표시		연결자 흐름이 다른 곳과 연결되는 입출구를 나타냄
	서류 서류를 매체로하는 입출력 표시		준비 기억장소, 초기값 등 작업의 준비 과정을 나타냄
	수동입력 콘솔에 의한 입력		천공카드 천공카드의 입출력

설계 단계에서의 알고리즘 기술 방법

흐름도

시작

- ① 기차표 예매사이트에 접속한다.
 - ② 원하는 시각과 역에 정차하는 좌석이 있는가?
 - ③ 있으면 4로 간다.
 - ④ 없으면 다시 2로 가서 다른 좌석을 검색한다.
 - ⑤ 결제한다.
 - ⑥ 구매 완료한 기차표를 출력한다.
- 종료

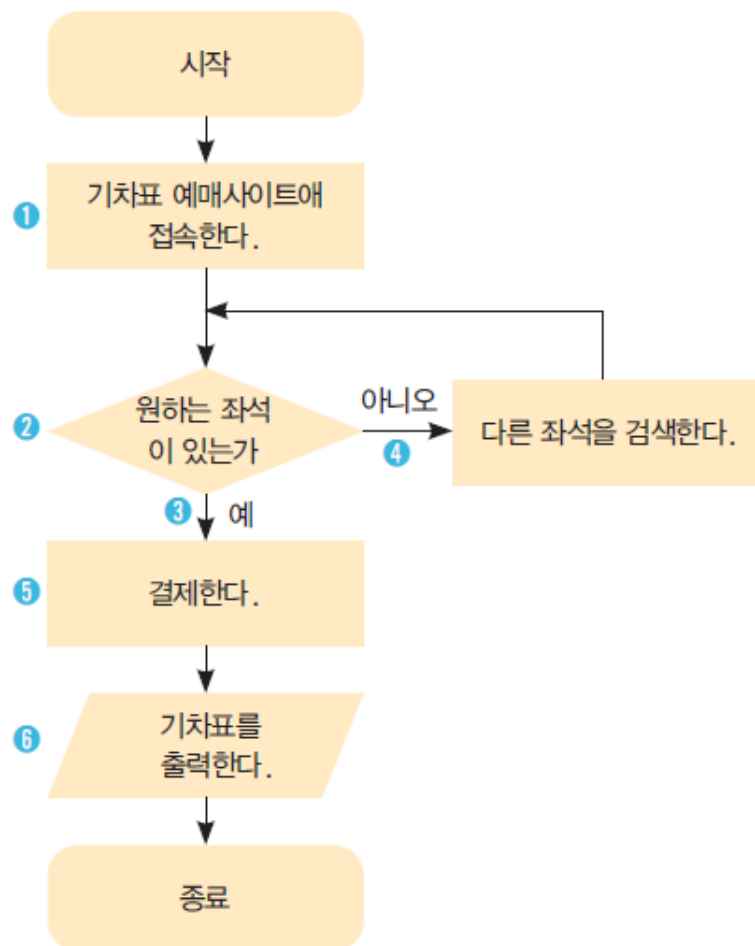
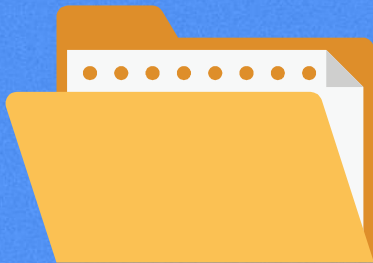
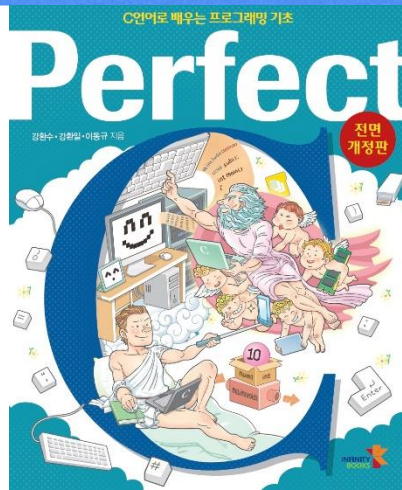


그림 1-51 '인터넷에서 기차표를 예매하고 출력'하는 과정의 자연어 한글 기술과 흐름도 표현



06. 다양한 ' 프로그래밍 언어 '



50 ~ 60년대에 개발된 프로그래밍 언어(1)

포트란

- 과학과 공학 및 수학적 문제들을 해결하기 위해 고안된 프로그래밍 언어
- 널리 보급된 최초의 고급 언어
- FORMula TRANslating system(수식 번역 시스템)의 약자
- 수학에서 사용하는 $+$, $-$, $*$, $/$ 와 같은 사칙연산 기호와 \sin , \cos , \tan 의 삼각함수를 비롯하여 \log , abs 등 다양한 수학함수들을 프로그램 내에서 그대로 사용
- 어셈블리 언어에 익숙해져 있던 1957년경, 포트란은 IBM에서 존 배커스(John Backus) 등의 전문가가 개발한 프로그래밍 언어
- FORTRAN은 가장 오래된 언어지만 언어 구조가 단순해 놀라운 생명력을 갖추고 있어 지금도 과학 계산 분야 등에서는 많이 사용

50 ~ 60년대에 개발된 프로그래밍 언어(2)

코볼

- 학계 위원과 허니웰, GE, 뷰로우, RCA, IBM 등의 업체가 참여한 협회 CODASYL이 1960년, 사무처리에 적합한 프로그래밍 언어로 개발한 것이 코볼(COMmon Business Oriented Language)
- 포트란에 이어 두 번째로 개발된 고급 언어
- 대량의 데이터를 효율적으로 입력, 출력 및 처리
- 코볼 컴파일러만 있으면 어떠한 컴퓨터 기종이라도 코볼 프로그램을 작성하여 실행 가능
- 코볼은 사무처리에 목적이 있으므로 다른 프로그래밍 언어에 비하여 파일이나 데이터베이스에서 데이터를 쉽게 읽고 쓰며, 또한 양식을 가진 보고서를 쉽게 만들 수 있는 등 사무처리에 효율적
- FORTRAN이 수식과 비슷한 반면, 코볼은 일상 영어회화와 비슷한 구어체 문장 형태를 갖고 있으므로 쉽게 이해할 수 있도록 프로그램 작성이 가능

50 ~ 60년대에 개발된 프로그래밍 언어(3)

알골

- 알고리즘(ALGOritm)을 표현하기 위한 언어로 ALGOritmic Language를 줄여서 만든 이름
- 포트란이 미국을 중심으로 사용했다면 알골은 유럽을 중심으로 과학기술 계산용 프로그래밍 언어로 사용
- 알고리즘의 연구 개발에 적합한 언어로 개발
- 절차적 언어로서 구조화 프로그래밍에 적합하고, 최초로 재귀호출이 가능한 프로그래밍 언어
- 알골은 파스칼, C 언어 등 이후 언어의 발전에 큰 영향을 주었으나, IBM이 주로 포트란을 사용하여 더 이상 대중화에 성공하지 못함

50 ~ 60년대에 개발된 프로그래밍 언어(4)

베이직

- 1964년에 미국 다트머스(Dartmouth) 대학의 켄니(John Kemeny) 교수와 커쯔(Thomas Kurtz) 교수가 개발
- 베이직(BASIC)은 'Beginner's All-purpose Symbolic Instruction Code'(초보자의 다목적용이고 부호를 사용하는 명령어 코드)의 약어
- 초보자도 쉽게 배울 수 있도록 만들어진 대화형 프로그래밍 언어
- 대화형의 영어 단어를 바탕으로 약 200여 개의 명령어들로 구성된 가장 쉬운 대화형 프로그래밍 언어
- 문장의 종류가 많지 않고 문법이 간단하며, 배우고 쓰기가 간단하고 쉬움
- 컴파일 없이 바로 작동되나 인터프리터를 거쳐야 하므로 실행 속도가 느리다는 단점

50 ~ 60년대에 개발된 프로그래밍 언어(5)

베이직의 발전

- 1980년대에 개인용 컴퓨터의 출현과 함께 베이직은 기본 개발 언어로 탑재되어 범용적인 언어로 널리 사용
- 마이크로소프트는 이 베이직을 기본으로 비주얼 베이직(Visual Basic)이라는 프로그램 언어를 개발
- 비주얼 베이직은 표준 베이직에 객체지향 특성과 그래픽 사용자 인터페이스를 추가한 프로그램 언어이자 통합개발환경
- 웹 프로그래밍 기술에서 많이 사용하는 언어 중 하나인 ASP(Active Server Page)도 VBScript를 사용하는데, VBScript도 베이직 문법을 그대로 사용

70년대 이후 개발된 주요 프로그래밍 언어(1)

파스칼

- 파스칼은 프랑스의 수학자인 파스칼(Pascal)의 이름에서 따온 언어
- 프로그램을 작성하는 방법인 알고리즘 학습에 적합하도록 1971년 스위스 취리히 공과대학교의 니콜라우스 비르트(Nicholas Wirth) 교수에 의해 개발된 프로그래밍 언어
- 파스칼은 알골(Algol)을 모체로 해서 초보자들이 프로그램의 문법적 에러를 줄일 수 있도록 매우 엄격한 문법을 가진 프로그래밍 언어
- 구조적인 프로그래밍(structured programming)이 가능하도록 begin~end의 블록 구조가 설계
- 1980년에서 1990년대까지 대부분의 대학에서 프로그래밍 언어의 교과과정으로 파스칼을 채택
- 애플 사는 1980년 초, 파스칼 문법에 객체지향 기능을 추가시킨 오브젝트 파스칼(Object Pascal) 개발
- 1980년대에는 볼랜드 사에서 파스칼을 발전시켜 터보 파스칼(Turbo Pascal)이라는 제품으로 상용화
- 볼랜드 사는 1990년 중반에 마이크로소프트 사의 비주얼 베이직과 유사한 오브젝트 파스칼 언어를 기반으로 하는 델파이(Delphi)를 출시

70년대 이후 개발된 주요 프로그래밍 언어(2)

C++

- 1972년에 개발된 C 언어는 1983년에 프로그램 언어 C++로 발전
- C++는 객체지향 프로그래밍(OOP: Object Oriented Programming)을 지원하기 위해 C 언어가 가지는 장점을 그대로 계승하면서 객체의 상속성(inheritance) 등의 개념을 추가한 효과적인 언어
- AT&T의 얀 스트로스트럽(Bjarne Stroustrup)이 개발
- C++언어가 C언어와 유사한 문법을 사용함으로써 C언어에 익숙한 프로그래머들이 C++언어를 쉽게 배울 수 있다는 장점



70년대 이후 개발된 주요 프로그래밍 언어(3)

파이썬

- 현재 미국의 대학에서 컴퓨터 기초 과목으로 가장 많이 가르치는 프로그래밍 언어 중 하나
- 1991년 네덜란드의 귀도 반 로섬(Guido van Rossum)이 개발한 객체지향 프로그래밍 언어
- 비영리의 파이썬 소프트웨어 재단이 관리하는 개방형, 공동체 기반 개발 모델
- C#으로 구현된 닷넷프레임워크 위에서 동작하는 IronPython, Java로 구현되어 JVM위에서 돌아가는 Jython, 파이썬 자체로 구현된 PyPy 등 다양한 언어로 만들어진 버전 등 다양
- C언어로 구현된 C 파이썬(cpython) 구현이 사실상의 표준
- 프로그래밍 언어 파이썬이 대학의 컴퓨터기초 교육에 많이 활용
- 파이썬이 무료이며, 간단하면서 효과적으로 객체지향을 적용할 수 있는 강력한 프로그래밍 언어
- 베이직과 같은 인터프리터 언어로 간단한 문법구조를 가진 대화형 언어
- 쉽고 빠르게 개발할 수 있어, 개발기간이 매우 단축되는 것이 장점

70년대 이후 개발된 주요 프로그래밍 언어(4)

자바

- 1992년 미국의 SUN사에서 가전제품들을 제어하기 위해 고안한 언어에서 부터 시작
- 1995년 5월에 SunWorld 95에서 공식 발표되었으며 C++를 기반으로 한 객체지향 프로그래밍 언어
 - 월드와이드웹(World Wide Web) 이용에도 적합하도록 자바를 발전
 - 범용적인 프로그래밍 언어 자바의 개발도구인 JDK(Java Development Kit)를 발표
- 선 마이크로시스템즈 사는 오라클(oracle)사에 합병되어 현재 자바는 오라클 기술
- 자바개발환경인 JDK는 현재까지 계속 발표

70년대 이후 개발된 주요 프로그래밍 언어(4)

자바개발 배경

- 1990년 양방향 TV를 만드는 제어 박스의 개발을 위한 그린 프로젝트(Green Project)를 시작
 - 초기에는 객체지향 언어로 광범위하게 이용되고 있는 C++ 언어를 이용
 - 다양한 하드웨어를 지원하는 분산 네트워크 시스템 개발에 부족함을 느낀 개발팀은 C++ 언어를 기반으로 오크(Oak, 떡갈나무)라는 언어를 직접 개발
- 제임스 고슬링(James Gosling)은 이 오크라는 언어를 발전시켜 자바라는 범용적인 프로그래밍 언어를 개발

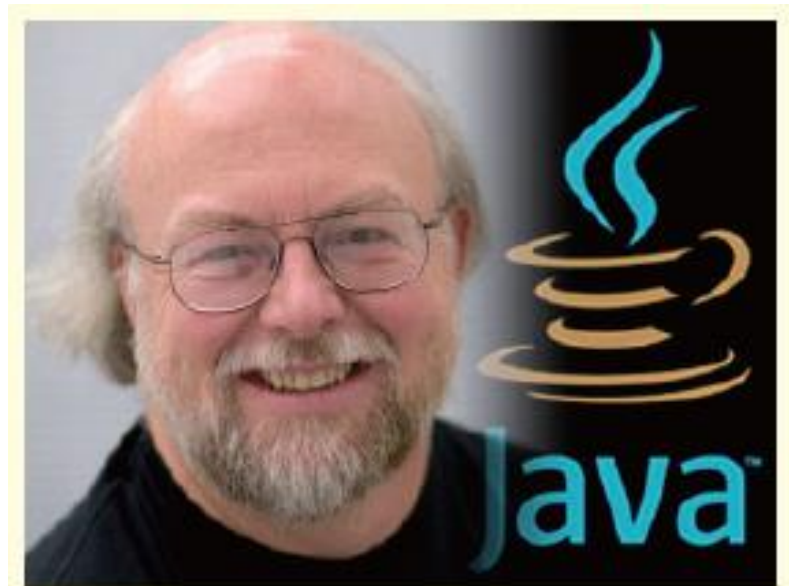


그림 1-55 자바 로고와 제임스 고슬링

프로그래밍 초보를 위한 비주얼 프로그래밍 언어

스크래치

- 2007년 MIT 대학의 미디어랩(Media Lab)에서 개발한 비주얼 프로그래밍(visual programming) 개발 도구
- 브라우저에서 직접 개발하는 환경으로 커뮤니티 기반 웹 인터페이스로 구성
- 일반인과 청소년 또는 프로그래밍 초보자를 학생들을 대상으로 컴퓨터 프로그래밍의 개념을 이해할 수 있도록 도와주는 교육용 프로그래밍 언어(educational programming language)
- 다양한 이미지나 사운드를 제공하여 쉽게 사용 가능
- 직관적으로 누구나 쉽게 이해할 수 있는 블록을 끼워 맞춰 프로그램을 작성

프로그래밍 초보를 위한 비주얼 프로그래밍 언어

스크래치



The image shows the Scratch website homepage. At the top, there's a navigation bar with links like '만들기' (Create), '답변하기' (Answer), '토론하기' (Discuss), '도움말' (Help), and a search bar. The main heading says '이야기와 게임, 애니메이션을 만들고 전세계 친구들과 공유하기' (Create stories and games, animations and share them with friends all over the world). Below this, there are three cartoon characters: a cat, a blue creature, and a yellow creature, each with a button to '바로 시작하기' (Start now). A code block preview shows a sequence of actions: 'when clicked', 'repeat 10', 'move 10 steps', 'change color by 25%', 'play drum 4* for 0.2 beats', and 'say Welcome to Scratch! for 2 secs'. The text 'A creative learning community with 9,650,401 projects shared' is displayed. Below this, there's a section for '최근 프로젝트' (Recent projects) with five thumbnails: 'How To Draw Dragons', 'How I recover from...', 'Better Pen Thing', 'A Guide Through La...', and 'Bench Breakd'. Another section for '최근 스튜디오' (Recent studios) shows four thumbnails: 'APis', 'Media Library!', 'Virtual Architecture', and 'Food Puns!'. At the bottom, there's a link to a '규제된 프로젝트 by Malik44'.

Scratch 만들기 답변하기 토론하기 도움말 검색 스크래치 게임 로그인

이야기와 게임, 애니메이션을 만들고
전세계 친구들과 공유하기

바로 시작하기 예제 보기 스크래치 게임 (부록!)

A creative learning community with 9,650,401 projects shared

스크래치 소개 | 교육자 | 부모

최근 프로젝트

How To Draw Dragons
by happygeek432

How I recover from...
by cinnabungr113

Better Pen Thing
by PulkJosh

A Guide Through La...
by XxKawaii_GirlXx

Bench Breakd
by Fairyflrks

최근 스튜디오

APis

Media Library!

Virtual Architecture

Food Puns!

규제된 프로젝트 by Malik44
<https://scratch.mit.edu/projects/602788747>

더 알아보기

프로그래밍 초보를 위한 비주얼 프로그래밍 언어

앨리스

- 카네기 멜론 대학교에서 '마지막 강의'로 알려진 랜디 포시(Randy Pausch) 교수가 주도하여 개발한 프로그래밍 교육 도구
- 쉽게 삼차원 인터랙티브 그래픽 콘텐츠를 제작하면서 객체지향 프로그래밍 기법을 학습할 수 있는 교육용 소프트웨어
- 앨리스는 컴퓨터 전공학과와 저학년 학생들이 프로그래밍을 어렵게 생각하고 포기하지 않도록 프로그래밍 개념과 실습을 학습할 수 있도록 지원하는 프로그래밍 소프트웨어
- 특히 프로그래밍 언어 자바, C++, C#과 같은 객체지향 프로그래밍 기법을 학습하는데 유용한 개발 도구
- 다양한 캐릭터나 객체를 끌어다 놓는(Drag & Drop) 방식으로 프로그래밍
- 비주얼 프로그래밍 기법은 프로그래밍언어 문법학습에 대한 학습자들의 부담을 덜어 주고 쉽게 프로그래밍이 가능
- 클래스와 객체로 콘텐츠 제작에 사용할 수 있는 다양한 캐릭터를 제공하고, 사용법이 쉬우므로 일반인도 쉽게 삼차원 애니메이션이나 게임을 제작 가능



Thank you