04
프로그래밍 언어

들어가기

OBJECTIVE

학습내용

1. 프로그래밍 언어 개요
2. 프로그램 구현
3. 고급 프로그래밍 언어 종류
4. 프로그래밍 언어와 구성 요소
5. 객체지향 프로그래밍

들어가기

학습**목표**

- ‘프로그래밍 언어’란 무엇인지 설명할 수 있다.
- 프로그래밍 언어로 프로그램을 구현할 수 있다.
- 고급 프로그래밍 언어의 종류를 나열할 수 있다.
- 프로그래밍 언어와 구성 요소를 설명할 수 있다.
- 객체지향 프로그래밍이란 무엇인지 설명할 수 있다.

복습하기

지난 주차 **복습**

03주차 학습내용. 컴퓨터 구조

1. 컴퓨터 시스템의 기본 구조
2. 기억장치
3. 중앙처리장치
4. 마이크로프로세서

복습하기

지난 주차 **복습**

컴퓨터 시스템의 기본 구조

- 저장 프로그램 방식
  - 메모리에 자료뿐만 아니라 프로그램도 저장되는 프로그램 내장 방식

기억장치

- 컴퓨터가 작동하는 동안 중앙처리장치가 해야 할 작업 내용인 프로그램 명령어와 프로그램에서 이용할 자료를 저장하고 있는 기억장치

복습하기

지난 주차 **복습**

중앙처리장치

- 메모리에 저장된 프로그램과 자료를 이용하여 실제 작업을 수행하는 회로 장치

마이크로프로세서

- 마이크로프로세서의 성능 결정 요소
  - 사이클 당 연산 수와 자료 버스의 폭
  - 레지스터의 수와 크기
  - 캐시 메모리의 크기 등

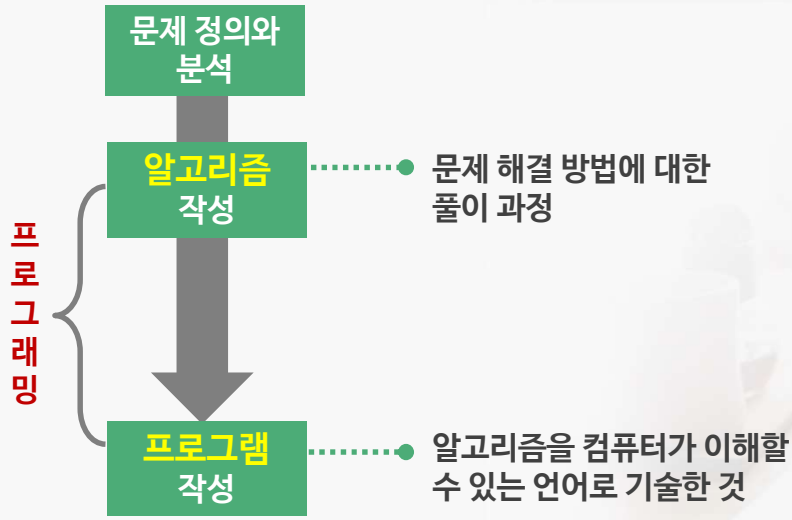
1

프로그래밍 언어 개요

1] 알고리즘

학습하기

◇ 컴퓨터를 이용한 문제 해결 과정



1] 알고리즘

학습하기

◇ 알고리즘의 중요성

1] 알고리즘

학습하기

◇ 알고리즘의 중요성

알고리즘 #1 | **알고리즘 #2**

① 100만 명의 납세액을 입력받는다.(1초)

② 대상자의 납세액을 읽어 온다.(1/100만 초)

③ 100만 명의 납세액 총액을 구한다.
(100만 * 1/100만 초 + (100만 – 1) * 1/100만 초)

④ ②의 값을 총합으로 나누어 납세 비중을 구한다.(1/100만 초)

⑤ 아직 남은 대상자가 있으면 ②~④의 과정을 반복한다.

총 소요시간 : 1 + (1/100만 + 2 + 1/100만) * 100만 = 약 200만 초

1] 알고리즘

학습하기

알고리즘의 중요성

알고리즘 #1 **알고리즘 #2**

① 100만 명의 납세액을 입력받는다.(1초)

② 100만 명의 납세액 총액을 구한다.
(100만 * 1/100만 초 + (100만 – 1) * 1/100만 초)

③ 대상자의 납세액을 읽어 온다.(1/100만 초)

④ ③의 값을 ②에서 계산한 값으로 나누어 납세 비중을 구한다.(1/100만 초)

⑤ 아직 남은 대상자가 있으면 ③~④의 과정을 반복한다.

총 소요시간 : 1 + 2 + (1/100만 + 1/100만) * 100만 = 약 5초

2] 프로그래밍 언어

학습하기

프로그래밍 언어의 필요성

- ✓ 사람과 컴퓨터가 서로 의사교환을 하기 위함
- ✓ 사람이 컴퓨터에게 지시할 명령어를 기술하기 위함
- ✓ 주어진 어떤 문제를 해결하기 위해 인간과 컴퓨터 사이에서 의사소통을 가능하게 하는 인공적인 언어를 말함

프로그래밍 언어

명령 지시

2] 프로그래밍 언어

학습하기

프로그래밍 언어를 공부해야 하는 이유

- 효율적인 알고리즘을 개발할 수 있는 능력의 향상
- 현재 사용하는 프로그래밍 언어의 능력을 향상
- 주어진 과제를 해결하는 최적의 언어를 선택
- 새로운 언어를 쉽게 배울 수 있음

3] 프로그래밍 언어의 분류

학습하기

저급 언어와 고급 언어

저급 언어

- 기계어
- 어셈블리어

컴퓨터가 이해하기 쉬운 언어

고급 언어

FORTRAN, COBOL, Lisp, BASIC, SNOBOL, Pascal, C, Ada, C++, Java, C#, 기타

인간이 이해하기 쉽고 친근한 언어

3] 프로그래밍 언어의 분류

학습하기

저급 언어(Low Level Language)

1. 컴퓨터의 주기억장치, 레지스터, 마이크로프로세서, 입출력 포트 등의 하드웨어를 직접 통제 가능함
2. 저급 언어를 사용하기 위해서는 하드웨어에 대한 충분한 지식이 요구됨
3. 고급 언어에 비하여 언어 자체가 어렵기 때문에 전문가라 하더라도 프로그램의 생산성이 낮음

3] 프로그래밍 언어의 분류

학습하기

저급 언어(Low Level Language)

기계어	• 0과 1로 표현 • 컴퓨터가 직접 이해할 수 있는 유일한 언어 • 연산 코드(Operation Code)와 피연산자(Operand)로 구성 • 프로그래밍하기가 상당히 어려움 • 컴퓨터 하드웨어에 대한 강력한 통제가 가능
어셈블리어	• 복잡한 기계어를 간략하게 기호화(Symbolize)함 • 기계어의 연산코드와 피연산자를 프로그래머가 이해하기 쉬운 기호형태로 일대일 대응시킨 언어 • 하드웨어 장치에 대한 강력한 통제 가능

3] 프로그래밍 언어의 분류

학습하기

저급 언어(Low Level Language)

- 두 정수의 합을 위한 명령어 집합

순서	기계어	어셈블리어	의미
명령어1	0101000000000100	LDA A	메모리 A의 내용을 누산 레지스터(AC)에 저장
명령어2	0111000000000110	ADD B	메모리 B의 내용과 누산 레지스터(AC)의 값을 더하여 누산 레지스터(AC)에 다시 저장
명령어3	0100000000000111	STA C	누산 레지스터(AC)의 값을 메모리 C에 저장
명령어4	0011000000000000	HLT	프로그램 종료

3] 프로그래밍 언어의 분류

학습하기

고급 언어(High Level Language)

〈1/2〉

프로그래밍 언어	연도	창시자	큰 영향을 준 선행 언어	주요 개발 목적
FORTRAN	1957	J. Backus(IBM)	-	수치 계산
COBOL	1960	위원회	FORTRAN	상업 자료 처리
Lisp	1962	J.McCarthy(MIT)	-	기호연산 리스트 처리
BASIC	1965	Kemeny (Dartmouth)	-	교육용 프로그래밍
SNOBOL	1966	T.Griswold	-	문자 처리
Pascal	1971	N.Wirth (ETH Zurich)	ALGOL68, BCPL	범용 및 교육용, 구조적 프로그래밍

3] 프로그래밍 언어의 분류

학습하기

고급 언어(High Level Language)

〈2/2〉

프로그래밍 언어	연도	창시자	큰 영향을 준 선행 언어	주요 개발 목적
C	1974	D.Ritchie (Bell Labs)	ALGOL68, BCPL	시스템 프로그래밍
Ada	1979	J.Ichbiah 등 (CII Honeywell bull)	Pascal, Simula 67, Modula	범용 및 응용 실시간 프로그래밍
C++	1983	B.Stroustrup	Simula67, ALGOL68, C	범용 및 시스템 프로그래밍
Java	1994	James Gosling	C	범용 및 인터넷 프로그램
C#	2000	Anders Helisberg 등 (Microsoft)	Java, C++, Visual Basic	객체지향적 웹 응용 프로그램

1] 프로그래밍 개요

학습하기

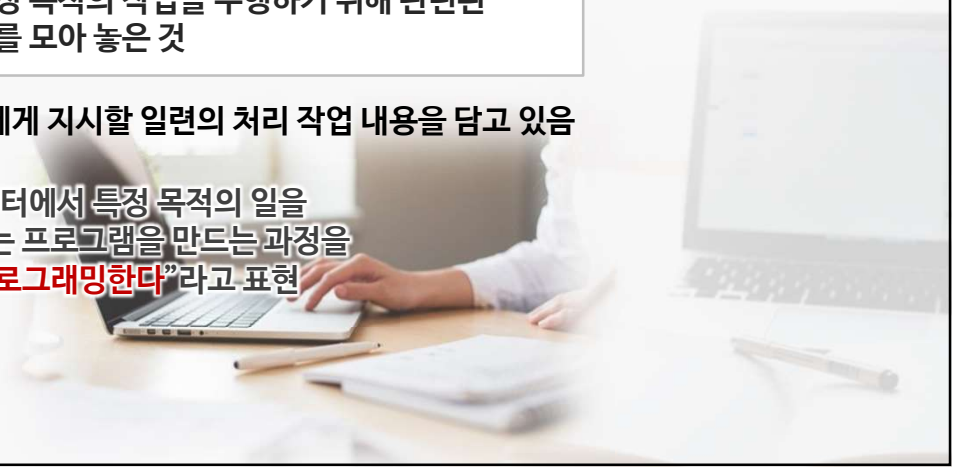
프로그램

프로그램(Program)

컴퓨터에서 특정 목적의 작업을 수행하기 위해 관련된 명령어와 자료를 모아 놓은 것

→ 컴퓨터에게 지시할 일련의 처리 작업 내용을 담고 있음

컴퓨터에서 특정 목적의 일을
수행하는 프로그램을 만드는 과정을
"프로그래밍한다"라고 표현



1] 프로그래밍 개요

학습하기

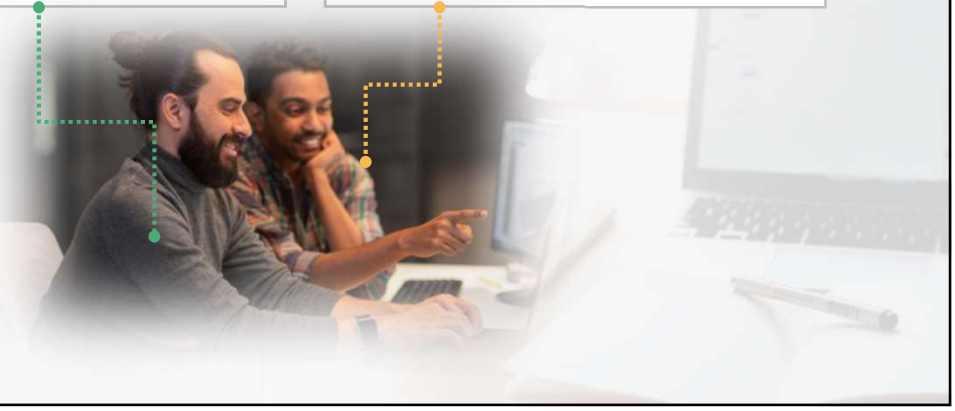
프로그래머

프로그래머(Programmer)

- 프로그램을 만드는 사람

개발자

- 넓은 의미로, 개발에 참여하는 사람



2] 프로그램 개발 환경

학습하기

개발도구	▪ 편집기(Editor) ▪ 프로그램 명령어인 프로그래밍 언어의 내용을 편집하는 편집기
컴파일러	▪ 개발 도구에서 가장 중요한 것은 작성한 고급 프로그래밍 언어를 컴퓨터가 이해할 수 있는 기계어로 변환
디버거	▪ 작성된 프로그램에서 발생하는 프로그램 오류를 쉽게 찾아 수정할 수 있도록 도와주는 프로그램
링커	▪ 여러 목적 파일을 하나의 실행 파일로 만들어 주는 기능

2] 프로그램 개발 환경

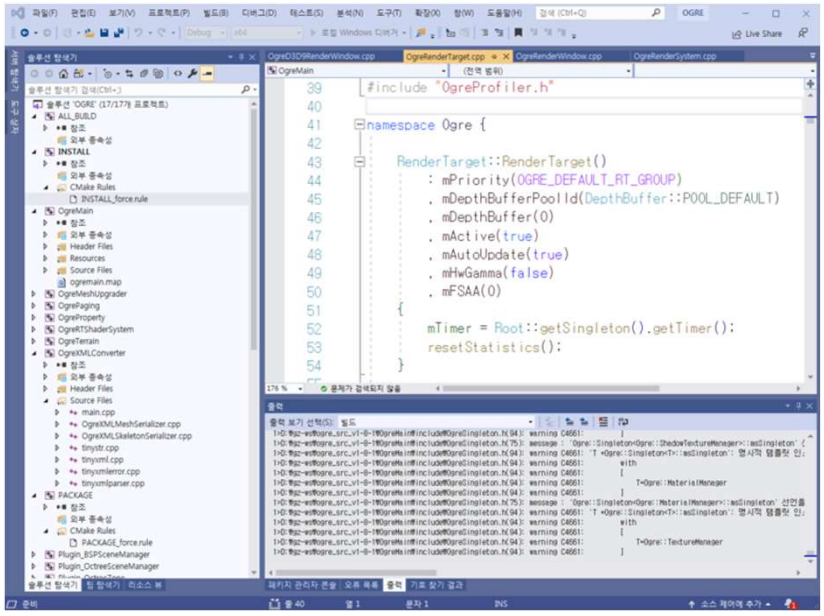
학습하기

◇ 통합 개발 환경(Integrated Development Environments)

통합 개발환경

프로그램을 개발하는데 필요한 컴파일러, 디버거, 링커, 에디터 등을 통합적으로 제공하는 개발 환경

마이크로소프트 사의 '비주얼 C++' (Visual C++)



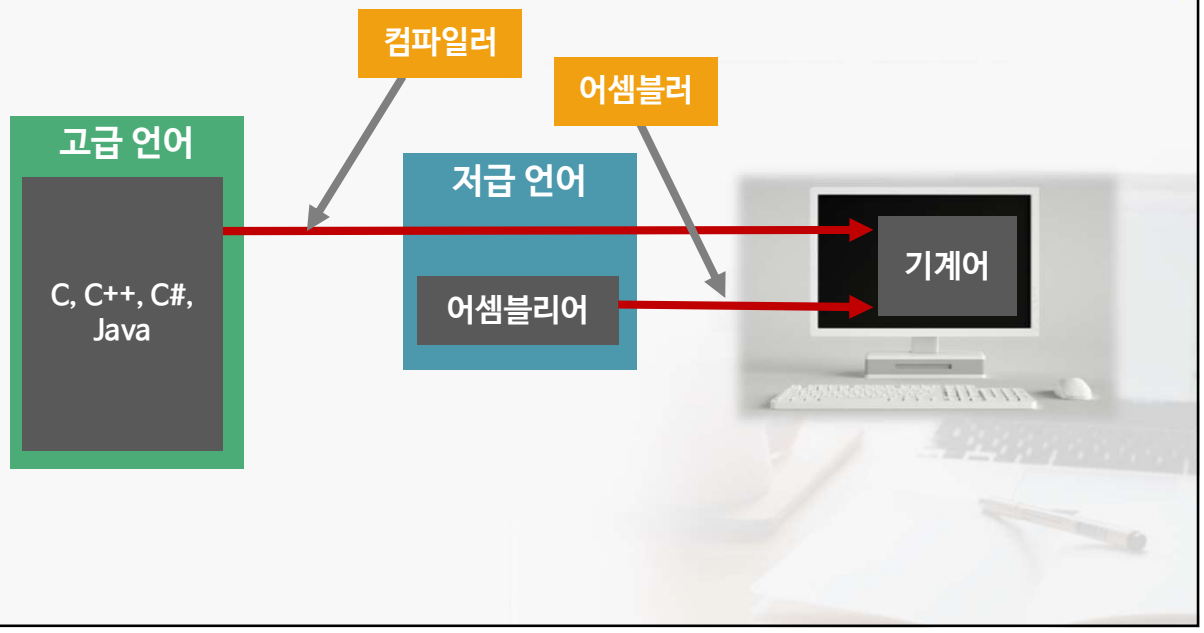
3] 프로그램 구현 과정

학습하기

- **소스작성** — 프로그램 언어를 이용하여 원하는 작업을 기술한 내용을 소스 코드(Source code) 또는 간단히 코드(Code)라 함
- **컴파일러** — 소스(Source) 파일(원시 파일)을 목적 파일(Object file)로 만들어주는 프로그램
- **어셈블러** — 어셈블리 언어의 프로그램을 기계어로 변환
- **링커** — 목적 파일을 실행 가능한 실행 파일(Eexcute File)로 만들어 주는 프로그램
- **로더** — 작성된 프로그램을 컴퓨터의 주기억장치에 로드함으로써 프로그램을 실행 가능하게 하는 역할 수행

3] 프로그램 구현 과정

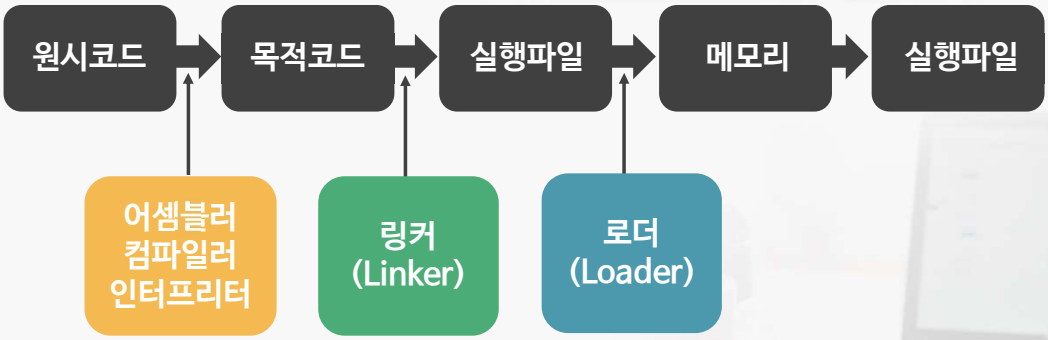
학습하기



3] 프로그램 구현 과정

학습하기

◇ 원시 코드에서 실행파일이 되어 실행되는 과정



3] 프로그램 구현 과정

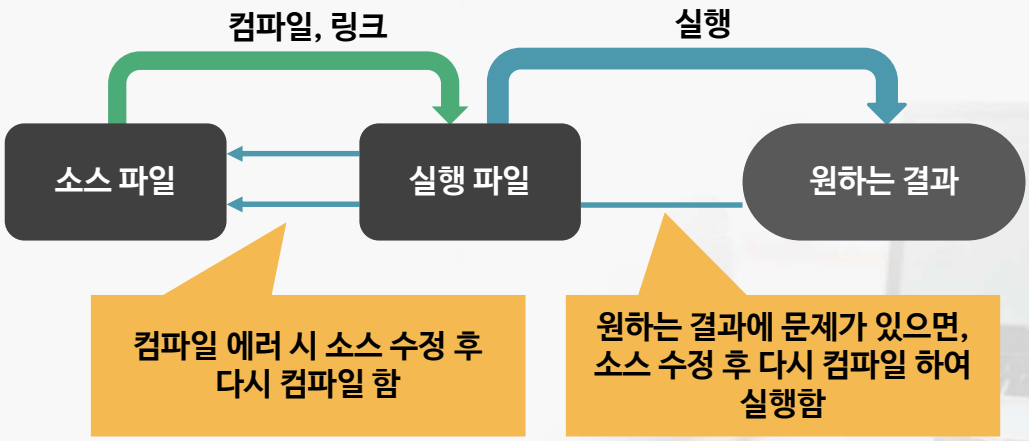
학습하기

디버거 (Debugger)	프로그램의 명령을 수행함에 있어 컴퓨터의 상태를 보여주거나 오류(또는 에러) 발생시 오류를 쉽게 찾을 수 있도록 도와주는 프로그램
에러 또는 오류	컴파일(시간) 에러, 실행(시간) 에러
디버깅 (Debugging)	컴파일 에러나 실행 에러를 수정하는 과정

3] 프로그램 구현 과정

학습하기

◇ 자바 응용 프로그램의 실행 과정



4] 컴파일러와 인터프리터

학습하기

인터프리터(Interpreter)

- 고급 언어를 기계어로 번역해주는 역할을 수행
- **원시 코드를 한 줄씩 읽어 들여 목적 코드로 바꾸어 줌**
- 컴파일러에 비해 번역 속도가 느릴 수 밖에 없지만, 프로그램을 작성할 때보다 융통성을 가질 수 있음

컴파일러(Complier)

- **원시 코드 전체를 읽은 다음 이를 기계어로 번역**
- 컴파일러는 한 번 컴파일한 후에는 수정이 없다면 매번 컴파일 할 필요 없이 빠른 시간 내에 프로그램 실행이 가능

4] 컴파일러와 인터프리터

학습하기

인터프리터	특징	컴파일러
실행되는 줄(라인) 단위 번역	번역 방법	프로그램 전체 번역
번역 과정이 비교적 간단하고 대화형 언어에 편리함	장점	한 번 컴파일 한 후에 매번 빠른 시간 내 전체 실행 가능
실행할 때마다 매번 기계어로 바꾸는 과정을 다시 수행해야 하기에 항상 인터프리터가 필요함	단점	프로그램의 일부를 수정하는 경우에도 전체 프로그램을 다시 컴파일 해야 함
즉시 실행	출력물	목적 코드
BASIC 등	언어 종류	FORTRAN, COBOL, C 등

4] 컴파일러와 인터프리터

학습하기

◇ 컴파일러와 인터프리터의 특징을 모두 갖는 방식

자바 언어와 C# 언어

- 컴파일러가 존재하여 컴파일 과정이 필요함
- 컴파일된 실행 파일을 실행할 때는 인터프리터 방식과 같이 인터프리터가 필요함
- 모든 시스템에서 독립적인 프로그램 언어를 개발하기 위함

4] 컴파일러와 인터프리터

학습하기

◇ 시스템에 독립적인 자바 프로그램의 실행

3

고급 프로그래밍 언어 종류

1] 프로그램 언어의 발전 과정

학습하기

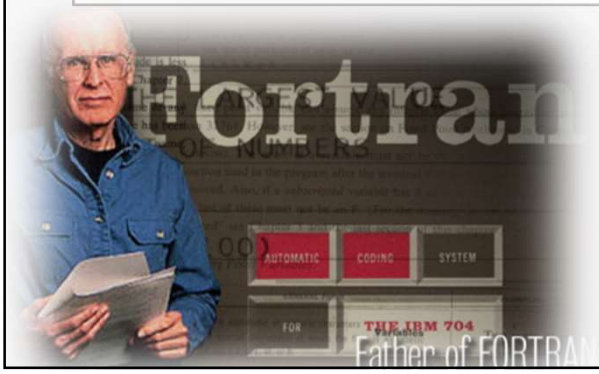
- 2000년대 이후
  - 4세대 언어
  - 5세대 언어
- 1990년대
  - C++
  - JAVA
  - Visual Basic
- 1980년대
  - BASIC
- 1970년대
  - C
  - PASCAL
- 1960년대
  - COBOL
  - PL/I
- 1950년대
  - 어셈블리언어
  - FORTRAN

2] 포트란

학습하기

포트란(FORTRAN)

- FORmula TRANslating system
- 과학과 공학 및 수학적 문제들을 해결하기 위해 1950년대 중반에 IBM 704 컴퓨터 시스템에 이용할 목적으로 IBM의 존 배커스(John Backus)에 의해 고안된 제3세대 프로그래밍 언어



3] 코볼

학습하기

코볼(COBOL)

- COmmon Business Oriented Language
- 코볼은 포트란에 이어 두 번째로 개발된 고급언어

3] 코볼

학습하기

코볼의 장점과 단점

장점

- 컴퓨터의 내부적인 특성과 별개로 설계되어 COBOL 컴파일러만 있으면 컴퓨터 기종에 관계없이 사용 가능
- 타 프로그래밍 언어에 비해 파일의 순차 및 비순차 처리 기능이 강력함
- 작성이 쉽고 이해하기 쉬움

단점

- 컴파일러에 너무 많은 항목을 포함하고 있어, 이를 수용하려면 주기억장치의 용량이 커짐
- 프로그램 작성 양이 많고 길어서 전체적으로 간결하지 못함

3] 코볼

학습하기

프로그램의 구성

디비전	설명	기술 내용
IDENTIFICATION	프로그램의 내용을 파악하는 식별 디비전	프로그램 이름, 작성자, 작성 일자 등
ENVIRONMENT	프로그램의 처리에 관계되는 환경 디비전	컴퓨터 종류, 입출력 파일 및 장치
DATA	데이터 처리를 위한 기억 장소 디비전	기억 장소 형식, 성격과 크기, 내용 등
PROCEDURE	처리할 명령에 관한 구체적 기술 디비전	처리 순서에 따른 명령문 실행을 기술

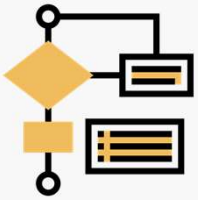
4] 파스칼

학습하기

파스칼

1971년 스위스의 나클라우스 워스(Nicholas Wirth) 교수에 의해 개발된 프로그래밍 언어

✓ 교육용으로 제작된 프로그래밍 언어이기 때문에 모든 명령어가 갖추어져 있음



알고리즘 실험

프로그램 연습

4] 파스칼

학습하기

✓ 제어 구조가 있어 구조적 프로그래밍에 적합

```
procedure squareroots (input,output)
var
x : real;
begin
repeat
read(x);
if x ≥ 0
then write(sqrt(x))
else write ('argument error')
until x = 0
end
```

① 복합문 begin-end

② 조건문 if-then-else

③ 반복문 while-do

5] 베이직

학습하기

◆ BASIC : Beginner's All-purpose Symbolic Instruction Code

1. 초보자도 쉽게 배울 수 있도록 만들어진 대화형 프로그래밍 언어로 1963년에 개발
2. 대화형의 영어 단어를 바탕으로 **약 200여 개의 명령어들로 구성된 가장 쉬운 대화형 프로그래밍 언어**
3. 1980년대에 개인용 컴퓨터의 출현과 함께 베이직은 기본 개발 언어로 탑재되어 범용적인 언어로 널리 사용
4. 마이크로소프트는 이 베이직을 기본으로 비주얼베이직(Visual Basic)이라는 프로그램 언어를 개발

6] 비주얼 베이직

학습하기

장점

- 초보자나 학생이 교육용으로 사용하기 좋음
- 한글 지원이 우수함
- 마이크로소프트사의 각종 툴을 편하게 이용할 수 있음

단점

- 객체 지향 기능이 C#, JAVA 등에 비해 상대적으로 약함

7] C와 C++

학습하기

1972년 C언어

- 켄 톰슨(Ken Tompson)이 개발한 B 언어에서 발전된 언어
- 시스템 PDP-11에서 운용되는 운영체제 유닉스(Unix)를 개발하기 위한 시스템 프로그래밍 언어로 미국전신전화국인 AT&T의 벨 연구소의 데니스 리치(Dennis Ritchie)가 개발

1983년 C++언어

- C 언어에서 발전된 언어
- 객체지향 프로그래밍(OOP Object-Oriented Programming)을 지원하기 위해 C언어가 가지는 장점을 그대로 계승하면서 객체의 상속성(inheritance) 등의 개념을 추가한 효과적인 언어

7] C와 C++

학습하기

C언어 계열의 장점과 단점

장점

- 어셈블리어 같은 저급 언어와 유사한 기능을 포함함
- 구조적 프로그래밍 기능이 있어 프로그램 읽기와 작성이 쉬움
- 프로그램의 융통성과 이식성이 상대적으로 뛰어남
- 기존 C언어로 개발된 프로그램들을 거의 수정 없이 C++언어로 확장할 수 있으므로 대부분의 운영 체제에서 사용할 수 있음
- 전 세계의 수많은 C 프로그래머들이 자연스럽게 C++ 프로그래머로 전환이 가능하여 전문 인력 부족 문제를 해결 가능

7] C와 C++

학습하기

C언어 계열의 장점과 단점

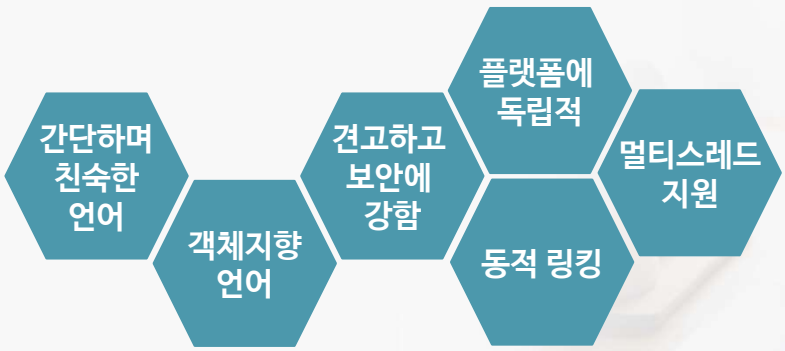
단점

- C는 객체 지향 개념이 없음
- C++는 너무 방대하고 복잡하여 안정성이 떨어지며, C언어와의 호환성이 강조됨에 따라 새로운 기능을 추가시키는데 제한이 따름
- C#은 사용자 저변이 JAVA에 비해 아직 활성화되기에 미흡함

8] JAVA

학습하기

1. C++의 강력함을 제공하면서도 규모는 더 작고 안전성은 강화된 언어
2. 웹 환경에 적합하며, 월드 와이드 웹의 보급 확대와 보조를 맞춰 발전



4

프로그래밍 언어와 구성 요소

1] 주석과 문장

학습하기

주석

주석

프로그램 언어의 문법에 관련 없이 프로그램 내부에 기술되는 부분으로, 프로그램을 설명하는 내용이나 기타 프로그래머가 기술하고 싶은 내용

주석 표현 방법

행 주석 (//)

블록 주석 (/* */)

```
//================================================
//                HelloComments.java
//================================================
/*
   main 메소드는 자바 응용 프로그램을 실행하는 경우,
   제일 먼저 실행되는 모듈입니다.
*/
```

1] 주석과 문장

학습하기

문장

문장

문장이 모여서 하나의 프로그램이 만들어지므로, 프로그램 언어에서 일을 수행하는 문법상의 최소 단위

문장 구분

C, JAVA	문장의 끝을 ;(세미콜론)으로 표시
BASIC	한 줄에 하나의 문장을 기술

블록

- 여러 문장의 모임

2] 변수와 자료유형

학습하기

변수

변수

프로그램에서 임시로 자료 값을 저장할 수 있는 저장 장소

변수의 선언

```
//변수 선언 구조
자료유형                    변수이름;

//실제 예
int myAge = 19;
```

- 대부분의 언어에서 변수는 반드시 사용하기 이전에 먼저 선언을 해야 함
- 이 선언은 시스템에게 적당한 공간을 메모리에 확보하라는 의미함

- C 언어에서 선언된 변수에 저장 값을 수정하려면 "=" 기호의 대입 연산자(assignment operator)를 이용함

3] 제어 구조

학습하기

제어 구조

프로그램 언어에서 프로그램의 실행 순서를 결정하는 주요 구문의 구조

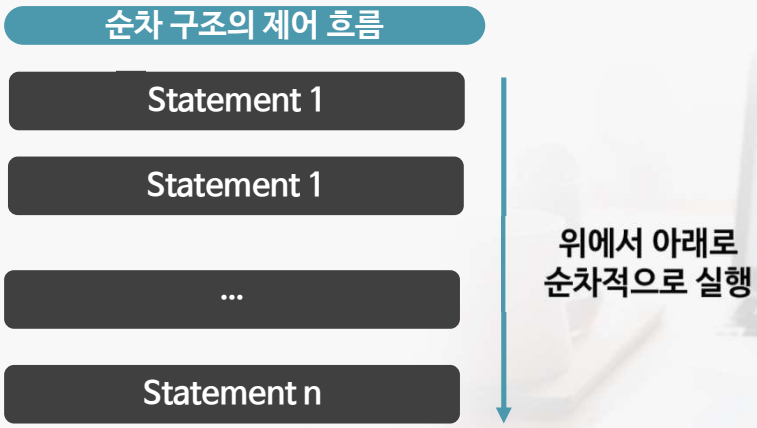
◇ 제어 구조의 종류

3] 제어 구조

학습하기

◇ 순차 구조

- 위에서 아래 순서로 문장을 실행하는 구조



3] 제어 구조

학습하기

선택 구조

- 주어진 조건이 만족되는 경우와 그렇지 않은 경우의 처리 내용을 달리 처리하고자 할 때 사용

if 문장

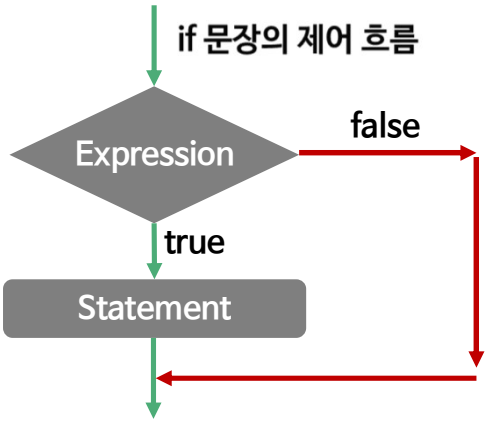
switch 문장

if 문장

if 문장 구조

```
//if (expression) statement;

if (expression)
    statement;
```

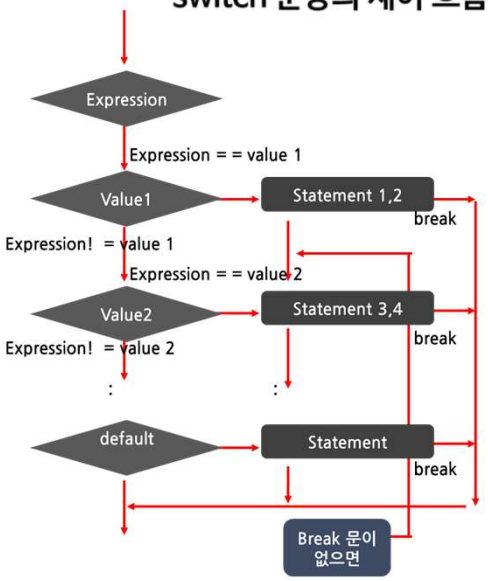


switch 문장

switch 문장 구조

```
switch(expression){
        case value 1:
                statements 1;
                statements 2;
                ...
                break;
          case value 2:
                statements 3;
                statements 4;
                ...
                break;
          ...
          case value N:
          ...
          ...
          default:
                statements;
                ...
                break;
```

switch 문장의 제어 흐름



3] 제어 구조

학습하기

반복 구조

- 어떤 일을 반복적으로 수행할 때 이용되는 구문의 구조

종류	구문	특징

for	for(초기화; 조건검사; 증감연산) {
	  for문 몸체(body);
	}
		▪ 일정한 반복 횟수를 이용하는 반복문에 적합함

while	while(조건검사) {
	  while문 몸체(body);
	}
		▪ 구문이 간단함
		▪ 검사 부분이 처음에 있어, 몸체를 1번도 실행하지 않을 수 있음

do while	do { do while문 몸체(body); } while(조건검사);	▪ 검사 부분이 뒤에 있어 반복 몸체를 적어도 1번은 실행함

5

객체지향 프로그래밍

1] 객체지향 개요

학습하기

- 1960년대 말에 시뮬라(SIMUAL)라는 프로그램 언어에서 처음 소개됨
- 객체지향은 많은 분야에서 발전됨

1] 객체지향 개요

학습하기

객체

객체

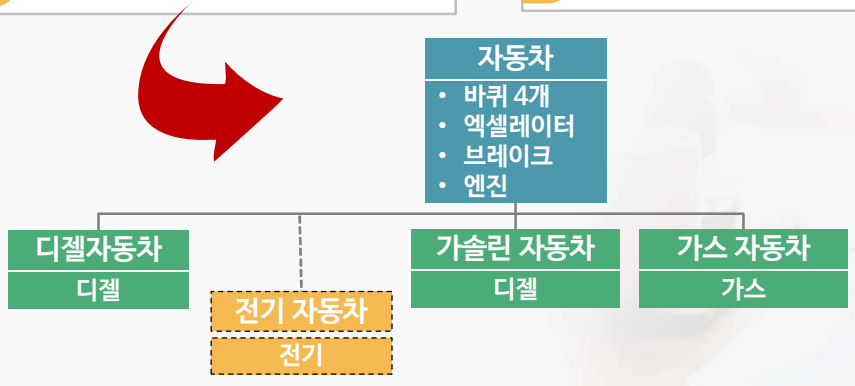
현실 세계의 사물이나 개념을 시스템에서 이용하기 위해 현실 세계를 자연스럽게 표현하여 손쉽게 이용할 수 있도록 만든 소프트웨어 모델

1] 객체지향 개요

학습하기

객체지향의 특징

1. 추상화(Abstraction)
2. 상속성(Inheritance)
3. 캡슐화(Encapsulation)
4. 다형성(Polymorphism)



1] 객체지향 개요

학습하기

객체의 구성

속성 (Attributes, Properties)	객체의 특성을 표현하는 정적인 성질
행동 (Messages, Behaviors)	객체 내부의 일을 처리하거나 객체들 간의 서로 영향을 주고받는 동적인 일을 처리하는 단위

1] 객체지향 개요

학습하기

객체의 구성

- 객체 자동차 모델링의 예

현실 세계의 자동차

모델링

객체 자동차

색상 차종 제조연월일 변속장치	객체 상태의 특징인 **속성 정보**로서 프로그램 언어로 구현할 때는 **소속 변수**가 됨
시동걸기 기어변속하기 속도증가하기 속도감소하기 정지하기	객체의 동적인 **행위 정보**를 나타내며 프로그램 언어로 구현할 때는 **소속 메소드**가 됨

2] 절차지향과 객체지향

학습하기

프로그램 방식에 따른 언어의 분류

절차지향 (Procedural) 언어	▪ 전체 과정을 나누어 처리하는 단위를 함수(Function)라 함 ▪ 문제를 여러 개의 작은 함수로 나누어 그 문제를 해결함 ▪ 절차지향이 동사 중심의 프로그래밍 방식
객체지향 (Object oriented) 언어	▪ 명사 중심의 프로그래밍 방식 ▪ 문제를 구성하는 객체를 만들어 이 객체들 간의 메시지 교환으로 문제를 해결 ▪ 객체는 자료와 일련의 처리 명령을 하나로 묶어 놓은 메소드로 구성되는 프로그램 단위로 함수보다 높은 수준의 모듈화 방법이라 할 수 있음

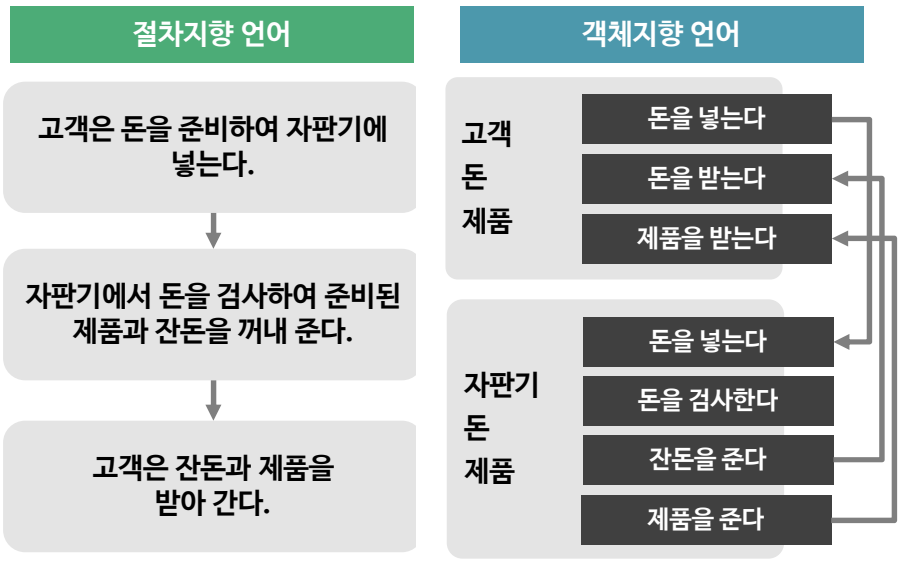
2] 절차지향과 객체지향

학습하기

프로그램 방식에 따른 언어의 분류

구분	절차지향	객체지향
프로그래밍 방식	동사	명사
모듈 단위	함수 또는 프로시저	속성과 행동을 표현한 객체
언어	FORTRAN, BASIC, COBOL, PASCAL, C 등	Object Pascal, Visual Basic, C#, C++, JAVA 등

자판기 분석



3] 객체지향 언어

학습하기

- 자바(JAVA)
  - 1992년 미국의 SUN 사에서 가전제품들을 제어하기 위한 언어에서부터 비롯됨
  - 1995년 자바는 공식 발표되었음
  - 프로그래밍 언어 C++를 기반으로 한 객체지향 프로그래밍 언어임

3] 객체지향 언어

학습하기

자바(JAVA)

자바의 역사

1. 썬 마이크로시스템즈 사는 1990년 양방향 TV를 만드는 제어박스의 개발을 위한 그린 프로젝트(Green Project)를 시작함
2. 개발의 책임자인 James Gosling은 이 Oak라는 언어를 발전시켜 자바라는 범용적인 프로그래밍 언어를 개발
3. 썬 마이크로시스템즈 사는 자바를 개발하면서 1990년 초부터 세계적으로 이용 범위가 폭발적으로 늘어나는 월드 와이드 웹(WWW) 이용에 적합하도록 자바를 발전시키게 됨
4. 썬 마이크로시스템즈 사는 1995년 5월에 SunWorld 95에서 자바를 공식 발표함

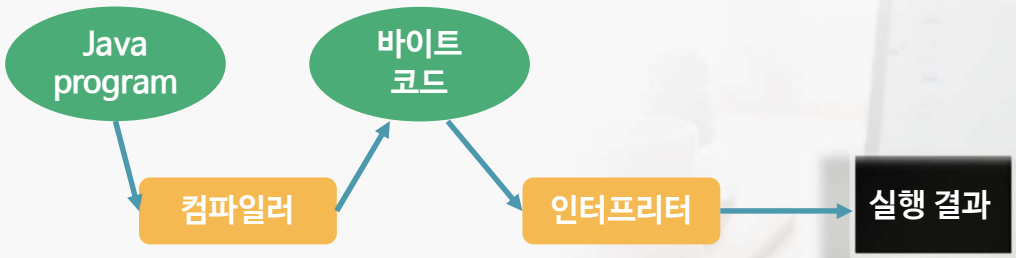
3] 객체지향 언어

학습하기

자바(JAVA)

자바는 시스템에 독립적인 언어

✓ 자바 가상 기계는 바이트코드가 실행될 수 있도록 돕는 가상 컴퓨터



SUMMARY

정리하기

정리하기

프로그래밍 언어 개요

- 기계어
  - 0과 1로 표현되는 프로그래밍 언어
- 저급 언어와 고급 언어
  - 저급 언어 : 컴퓨터가 이해하기 쉬운 언어
  - 고급 언어 : 사람이 이해하기 쉽고 친근한 언어

SUMMARY

정리하기

정리하기

프로그램 구현

인터프리터	특징	컴파일러
실행되는 줄(라인) 단위 번역	번역 방법	프로그램 전체 번역
번역 과정이 비교적 간단하고 대화형 언어에 편리함	장점	한 번 컴파일 한 후에 매번 빠른 시간 내 전체 실행 가능
실행할 때마다 매번 기계어로 바꾸는 과정을 다시 수행해야 하기에 항상 인터프리터가 필요함	단점	프로그램의 일부를 수정하는 경우에도 전체 프로그램을 다시 컴파일 해야 함
즉시 실행	출력물	목적 코드
BASIC 등	언어 종류	FORTRAN, COBOL, C 등

SUMMARY

정리하기

정리하기

고급 프로그래밍 언어 종류

- 베이직, 파스칼, C, C++, 자바 등

프로그래밍 언어와 구성 요소

- 주석, 문장, 변수, 제어구조(순차, 선택, 반복 구조)

객체지향 프로그래밍

- 객체지향 언어 : 자바

ANNOUNCEMENT

차시예고

정리하기

1주차 2주차 3주차 4주차 **5주차** 6주차 중간고사

운영체제

- 수고하셨습니다.