**소프트웨어 기초**

**1. S.W 정의**

\* 소프트웨어와 하드웨어 함께 작동하는 프로그램 및 데이터의 집합,

- **하드웨어**: 컴퓨터를 구성하고 있는 기계 장치의 총칭. 크게 본체와 주변 장치로 나눌 수 있는데,

본체는 다시 중앙 처리 장치와 주 기억 장치로, 주변 장치는 입력 장치·출력 장치·보조 기억 장치 등으로 구분

- **소프트웨어**: 소프트웨어는 컴퓨터나 관련 장치들을 동작시키는 데 사용되는 다양한 종류의 프로그램을 부르는 일반적인 용어

\* 컴퓨터시스템의 기능을 수행하는데 필요한 명령어와 데이터 구조를 포함

- **시스템 소프트웨어**: OS Ex) Windows, Linux

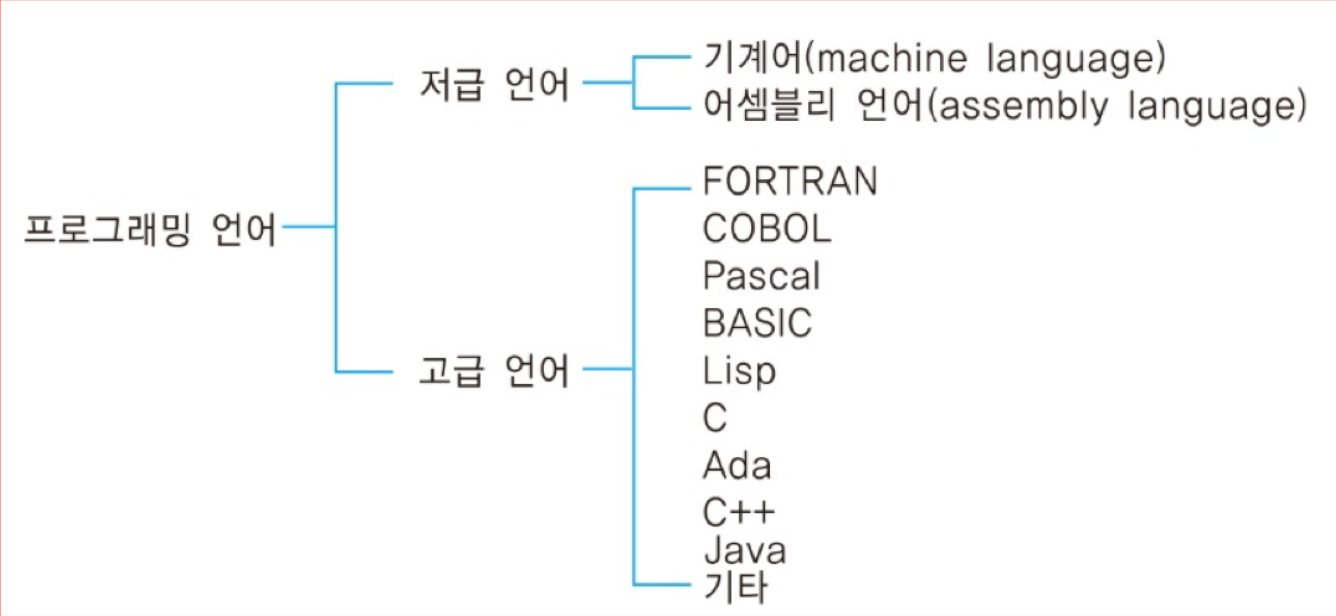
- **응용 소프트웨어**: 특정 작업을 수행하기 위해 설계된 소프트웨어, 직접 사용자에게 서비스를 제공

**2. 프로그래밍 언어**

- 소프트웨어 개발에 사용되는 언어.

- 컴퓨터에게 명령을 전달하기 위해 사용

- 프로그래밍 언어는 저급과 고급으로 나눔



- **저급 언어**: 하드웨어와 가까운 언어로 기계어와 어셈블리가 있음.

- **기계어**: 컴퓨터(CPU)가 별다른 해석(컴파일) 없이도 읽을 수 있는 프로그래밍 언어

- **어셈블리어**: 기계어에서 숫자를 의미 있는 단어로 바꿔서 사람들이 이해하기 쉽게 만드는 언어.

- **고급 언어**:

- 하드웨어 관련 지식이 없어도 프로그램 작성 가능, 생산성 수월

- 일상적인 언어, 기호 등을 사용하여 기계어보다 인간의 언어와 가깝다

- 기억장소를 임의 기호에 저장할 수 있음

- 기계어로 변환하기 위해 인터프리터나 컴파일러가 필수적으로 요구됨.



\* **프로그래밍 언어의 특성**

- **간결성(Simplicity)**: 사람이 프로그램을 쉽게 이해하고, 읽을 수 있도록 간결하게 표현할 수 있는 특성

- **직교성(Orthogonality**): 언어의 각 구성 요소가 상호 독립적이고 어떤 환경세서도 같은 요소로 사용됨.

- **가독성(Readability**) 누구나 쉽게 프로그램을 읽을 수 있는 특성:

- **정확성(Preciseness**): 엄밀하게 정의된 문법에 따라 작성된 프로그램은 정확성을 보장하며 예측 가능한 번역

- **기계 독립성(Machine Independence**): 서로 다른 컴퓨터 상에서도 운영이 가능해야 되며, 똑같은 결과 출력

\* **컴파일러**

- 고급언어를 기계어로 번역해주는 도구, 논리적 장치

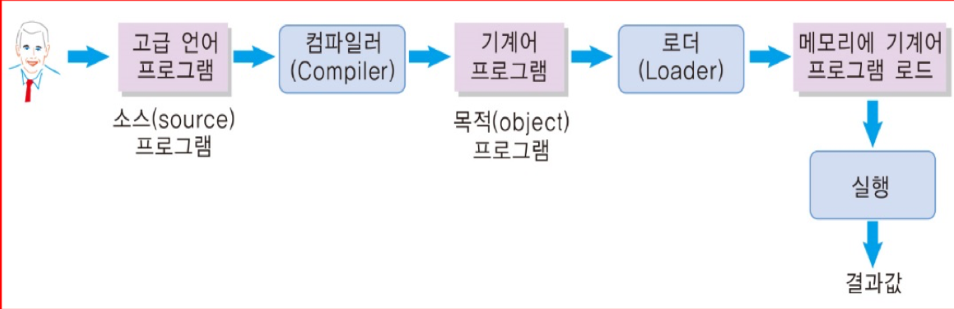
- 고급언어로 작성된 프로그램을 기계어로 번역하는 것.

- 개발자가 고급언어로 프로그래밍한 코드를 CPU가 이해할 수 있도록 변환 해주는 것

\* 컴파일과 빌드의 차이점

- 컴파일은 단순히 소스코드를 기계어로 번역해주는 것

- 빌드는 실행 가능한 파일로 만들어주는 것



**3. 알고리즘**

- 문제를 해결하기 위한 단계적 절차나 방법을 의미

- 특정 입력을 받아들여 결과를 생성하는 일련의 명령어

- 알고리즘의 효율은 시간 복잡도와 공간 복잡도로 평가

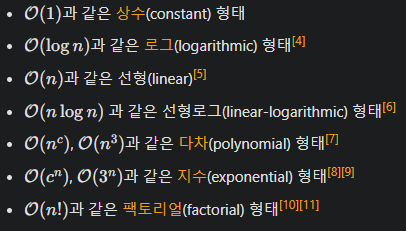
**\* 복잡도**

**-** 알고리즘의 성능,효율성 나타내는 척도

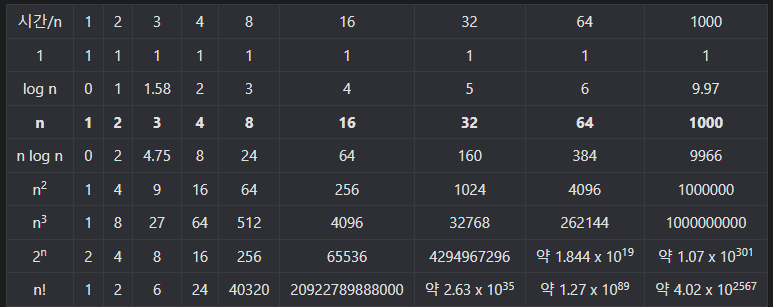
- 입력 값을 기준으로 연산에 사용되는 공간이 얼마나 되는지 객관적으로 비교

**\* 시간 복잡도:**

- 프로그램의 입력 값과 연산 수행 시간의 상관관계를 나타내는 척도.



- 복잡도는 아래로 내려갈수록 효율성이 떨어지며 상단에 있는 것일수록 효율이 올라간다.



**\* 공간 복잡도**

- 프로그램이 실행되고 얼마 많은 공간(메모리를)가 필요한지 나타냄.

\* 공간

- 고정 공간: 코드가 저장되는 공간, 알고리즘 실행을 위해 시스템이 필요한 공간.

- 가변 공간: 문제 해결위해 알고리즘이 필요한 공간.

**4. 데이터 구조**

**-** 데이터를 저장하고 조작하는 방법을 정의

- 데이터 구조는 문제 해결에 필요한 데이터를 구성하는 방식

1. 선형 자료구조

- 배열: 동일한 데이터 타입을 연속적으로 저장

- 리스트: 동적으로 크기가 변할 수 있는 데이터 집합

- 스택: LIFO방식의 데이터를 저장

- 큐: FIFO방식으로 데이터를 저장

- Deque: 양쪽 끝에서 데이터를 추가/반환/삭제 방식으로 데이터를 저장

2. 비선형 자료구조

- 트리: 계층적 구조로 데이터를 저장

DFS: 깊이 우선 탐색:

BFS: 너비 우선 탐색

- 그래프

**5. 소프트우에 개발 생명주기(SDLC)**

- 소프트웨어의 계획, 개발, 테스트, 배포, 유지보수 등 단계적 접근방식.

- 요구사항 분석: 사용자 요구사항을 수집하고 분석.

- 설계: 시스템 아키텍처와 데이터 구조를 설계.

- 구현: 실제 코드를 작성하여 소프트웨어를 개발.

- 테스트: 소프트웨어의 오류를 찾고 수정.

- 배포: 소프트웨어를 사용자에게 배포.

- 유지보수: 소프트웨어의 문제를 해결하고 업데이트.