2. 프로그래밍 언어의 발전사

---

2. 프로그래밍 언어의 발전사

> 핵심 개념
>
> - 고급 프로그래밍 언어의 시대별 발전 과정
> - 각 시대별 대표 언어와 특징
> - 추상화의 개념과 종류
> - 데이터 추상화와 제어 추상화

1. 고급 프로그래밍 언어의 발전 과정

1.1 1950년대: 고급 프로그래밍 언어의 시작

> 최초의 고급 언어들
>
> - FORTRAN: 최초의 고급 언어
> - COBOL: 최초의 비즈니스용 언어
> - LISP: 최초의 함수형 언어

FORTRAN (1957)

- **목적**: 과학 계산용 언어
- **주요 특징**:
  - 최초의 고급 프로그래밍 언어
  - 효율적인 부동소수점 연산
  - 배열, FOR 반복문, IF-문 등 기본 제어 구조 도입

COBOL (1959-1960)

- **목적**: 비즈니스 처리용 언어
- **주요 특징**:
  - 영어와 유사한 구문
  - 레코드 구조와 데이터 구조 분리
  - 다양한 출력 기능

LISP (1958)

- **목적**: 인공지능 분야
- **주요 특징**:
  - 리스트 자료구조 기반
  - 재귀호출 지원
  - 함수형 프로그래밍 패러다임

1.2 1960년대: 프로그래밍 언어의 다양화

> 구조적 프로그래밍의 시작
>
> - Algol60/68: 구조적 프로그래밍의 기초
> - BASIC: 교육용 언어의 시작
> - Simula-67: 객체지향의 시작

Algol60/68

- **주요 특징**:
  - 구조적 문장 (begin-end 블록)
  - 자유 양식 (free format)
  - 재귀 호출 지원
  - 값 전달 매개변수

BASIC

- **주요 특징**:
  - 대화형 프로그래밍 지원
  - 교육용으로 널리 사용
  - PC 시대의 대표 언어

1.3 1970년대: 단순성과 새로운 패러다임

> **시스템 프로그래밍과 논리 프로그래밍**
>
> - C: 시스템 프로그래밍의 혁신
> - Prolog: 논리 프로그래밍의 시작
> - Pascal: 교육용 구조적 언어

C 언어

- **주요 특징**:
  - 시스템 프로그래밍 언어
  - 중급 언어 (middle-level)
  - 효율적인 메모리 관리
  - 모듈화와 함수 기반 프로그래밍

Prolog

- **주요 특징**:
  - 논리 프로그래밍 언어
  - 사실(Fact), 규칙(Rule), 질의(Query) 기반
  - 선언적 프로그래밍

1.4 1980년대: 객체지향의 시대

> **객체지향 프로그래밍의 발전**
>
> - Smalltalk: 순수 객체지향 언어
> - C++: C의 객체지향 확장
> - Ada: 임베디드 시스템용 언어

Smalltalk

- **주요 특징**:
  - 순수 객체지향 언어
  - GUI 지원
  - 메시지 전달 기반 프로그래밍

C++

- **주요 특징**:
  - C 언어의 객체지향 확장
  - 하위 호환성 유지
  - 클래스 기반 객체지향

1.5 1990년대 이후: 인터넷 시대

현대 프로그래밍 언어의 특징

- Java: 플랫폼 독립성
- Python: 다목적 스크립트 언어
- JavaScript: 웹 프로그래밍

2. 추상화(Abstraction)

2.1 추상화의 개념

추상화의 정의

실제적이고 구체적인 개념들을 요약하여 보다 높은 수준의 개념을 유도하는 과정

2.2 데이터 추상화

데이터 추상화의 종류

1. 기본 추상화
   - 변수: 메모리 위치의 추상화
   - 자료형: 값의 종류 추상화
2. 구조적 추상화
   - 배열: 연속된 데이터 추상화
   - 레코드: 관련 데이터 추상화

기본 추상화

- **변수**
  - 메모리 위치의 추상화
  - 예: 메모리 120번지 → 변수 x
- **자료형**
  - 값들의 종류에 대한 추상화
  - 예: int, float, double 등

구조적 추상화

- **배열**
  - 같은 타입의 연속된 데이터 추상화
  - 인덱스 기반 접근
- **레코드(구조체)**
  - 서로 다른 타입의 관련 데이터 추상화
  - 필드 기반 접근

2.3 제어 추상화

제어 추상화의 종류

1. 기본 제어 추상화
   - 대입문
   - goto 문
2. 구조적 제어 추상화
   - if-문, switch-문
   - 반복문 (for, while)
   - 프로시저(함수)

기본 제어 추상화

- **대입문**
  - 여러 기계어 명령어의 추상화
  - 예: x = x + 3
- **goto 문**
  - jump 명령어의 추상화

구조적 제어 추상화

- **조건문**
  - if-문, switch-문
  - 중첩된 조건 처리
- **반복문**
  - for-문, while-문
  - 반복 작업의 추상화
- **프로시저(함수)**
  - 일련의 계산 과정을 이름으로 추상화
  - 재사용 가능한 코드 블록

2.4 추상 자료형(ADT)

추상 자료형의 정의

데이터와 관련된 연산들을 캡슐화하여 정의한 자료형

추상 자료형의 구현

- **Modula-2**: 모듈
- **Ada**: 패키지
- **C++/Java**: 클래스

핵심 정리

1. 프로그래밍 언어의 시대별 발전 과정 이해
2. 각 시대별 대표 언어의 특징과 목적 파악
3. 추상화의 개념과 종류 이해
4. 데이터 추상화와 제어 추상화의 차이점
5. 추상 자료형의 개념과 구현 방식