# 2.5 1970 년대: 단순성, 추상화 및 연구

## Pascal 개요

- N. Wirth에 의해 설계됨
- Algol-68의 단순화된 버전
- 작고, 간단하고, 효율적인 블록 구조 언어
  (small, simple, efficient structured language)

# C 언어

- 설계 목표
  - 벨 연구소의 Dennis Ritchie에 의해 Algol의 발전된 형태로 설계됨
  - UNIX 구현을 위한 언어로 설계됨
- Pascal과 다른 형태의 단순성을 추구함
  - 타입 및 실행시간 시스템의 복잡도를 줄임
  - 하부 기계(하드웨어)에 대한 접근
- 전체적인 설계의 일관성

# Pascal과 C의 설계원리 : 단순성(Simplicity)

- 단순성은 쉬운 것 같지만 실제로는 매우 어렵다.
- 일반성, 일관성, 직교성은 단순성과 배치된다.
- Simplicity Principle by A. Einstein

  *Everything should be made as simple as possible, but not simpler*

## 70년대 중반/후반

- 실험적 연구
  - 데이터 추상화, 병행성, 프로그램 검증 등을 위한 실험적 언어 개발
- CLU by MIT
  - 추상화 메커니즘
  - 프로그램 개발의 생산성
- Euclid by Univ. of Toronto
  - 추상화 자료형
  - 프로그램 검증
- Mesa by Xerox PARC
  - 모듈, 예외처리, 병행성

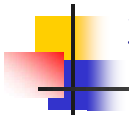
## 2.6 1980 년대: 새로운 방향 및 객체-지향

## 시대적 배경

- 프로그램의 복잡도 증가
- 추상화 메커니즘의 필요성 증가

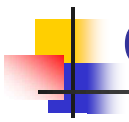
## Ada : 개요

- 미국 국방성 J. Ichbiah 팀에 의해 개발
- 추상화 자료형(Abstract data type)
  - Package
- 병행 프로그래밍(Concurrent programming)
  - Task
- 예외 처리(Exception handling)
- 질문
  - *Is Ada another PL/I ?*



## 객체-지향 프로그래밍 언어

- 객체-지향 프로그래밍에 대한 관심 증가
  - 프로그램 복잡도 증가에 대한 해결 방안
- Smalltalk-80
  - Alan Kay at XEROX PARC
  - 순수 객체-지향 프로그래밍 언어



## C++ : 개요

- 벨 연구소의 B. Stroustrup에 의해 개발됨
- 객체-지향을 포함한 C 언어의 확장
  - 객체, 클래스, 상속 등을 포함하도록 C 언어 확장
- **복잡도 제어(Complexity Control)**
  - 증가하는 프로그래밍의 복잡성(S/W 위기)
  - **통합 추상화(unit abstraction)**
    - 클래스

## 새로운 방향

- 함수형 언어(functional language)
  - Scheme, ML, Miranda, Haskell
- 논리 언어(logic language)
  - Prolog, Concurrent Prolog, ...
  - Constraint Logic Programming(CLP)

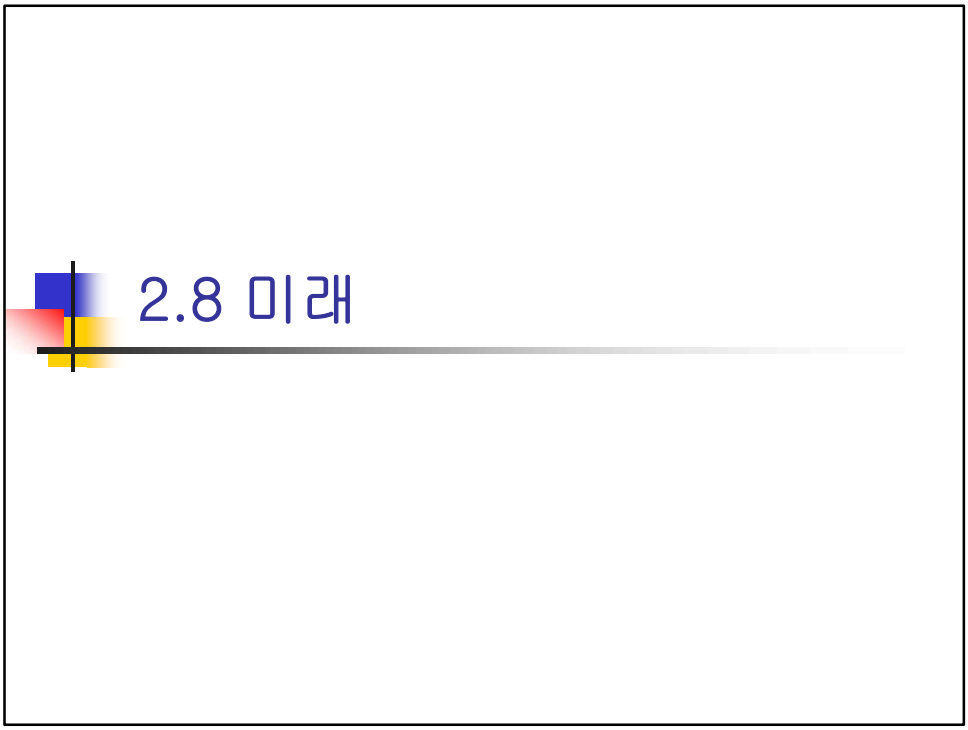
## 2.7 1990 년대: 인터넷 환경

# 인터넷 프로그래밍

- Java
  - 1995년 Sun 사 James Gosling에 의해서 개발됨
  - 프로그래밍 언어 역사에서 가장 놀라운 사건
  - 웹 및 인터넷을 위한 객체-지향 언어
    - 모바일 코드가 웹 브라우저 상에서 실행 가능
- 설계원리
  - 비교적 단순하고, 깨끗하게 설계됨
  - 이식성(portability)이 좋은 언어
    - JVM의 바이트코드로 컴파일되어 해석기로 실행된다
  - 강력한 라이브러리
    - 윈도우, 네트워크, 병행성

# 인터넷 프로그래밍

- 인터넷 환경을 위한 Script 언어
  - 클라이언트
    - AWK, Tcl, HTML, JavaScript
  - 서버
    - Perl, JSP, ASP
- 라이브러리의 중요성 증가
- 통합 사용
  - 유틸리티, 라이브러리, 운영체제 명령어


2.8 미래