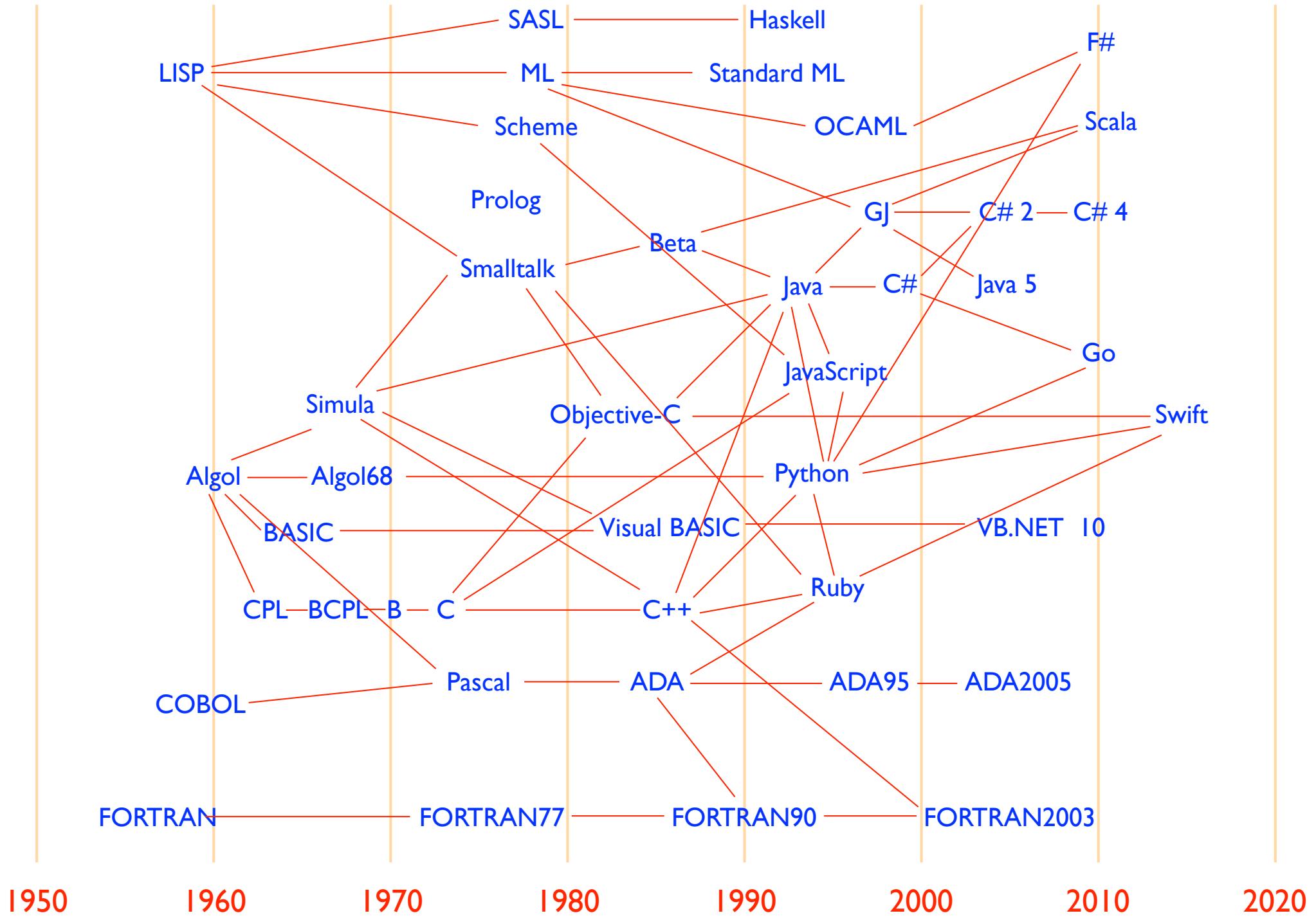




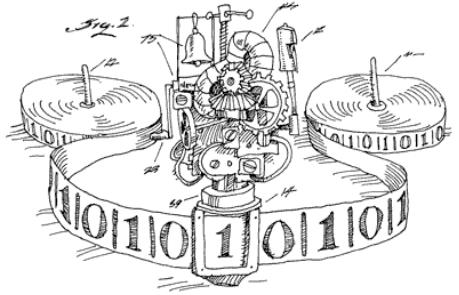
ENE4014 프로그래밍언어론

# 프로그래밍언어의 역사

소프트웨어학부  
도경구

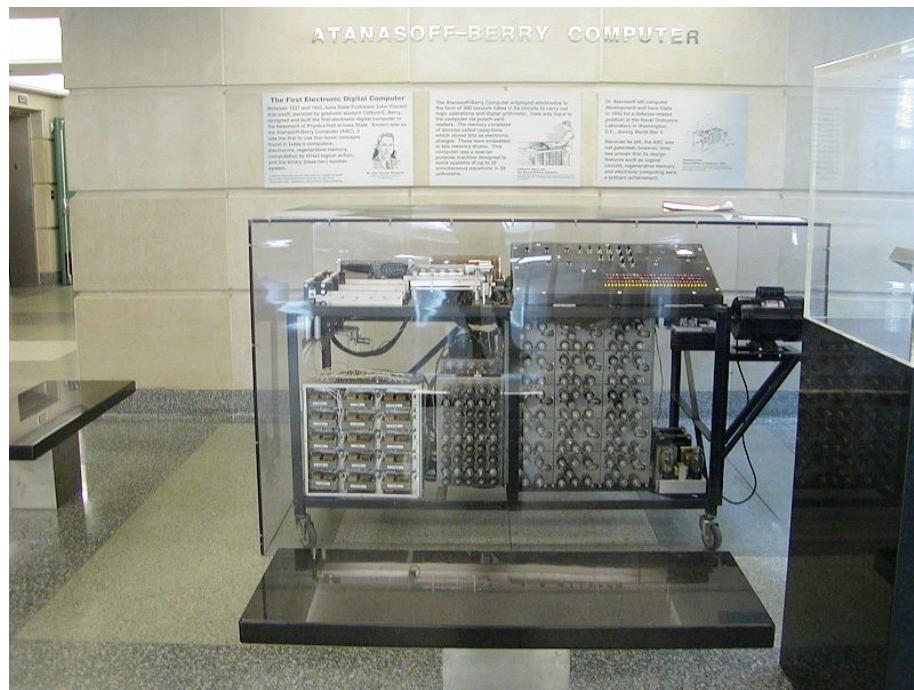


# 1950년대: 컴퓨터 출현



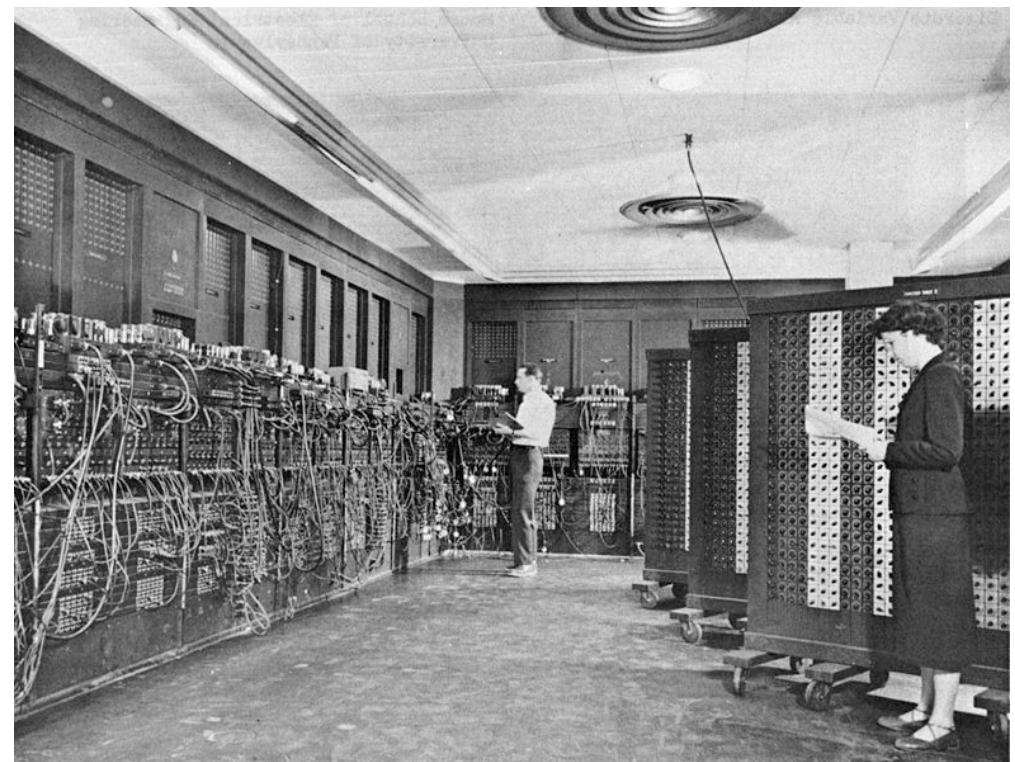
Alan Turing

튜링 머신의 혁신화

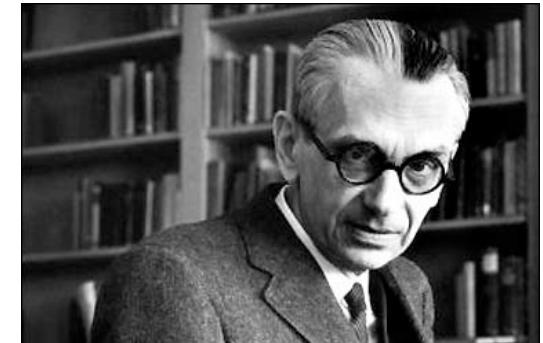
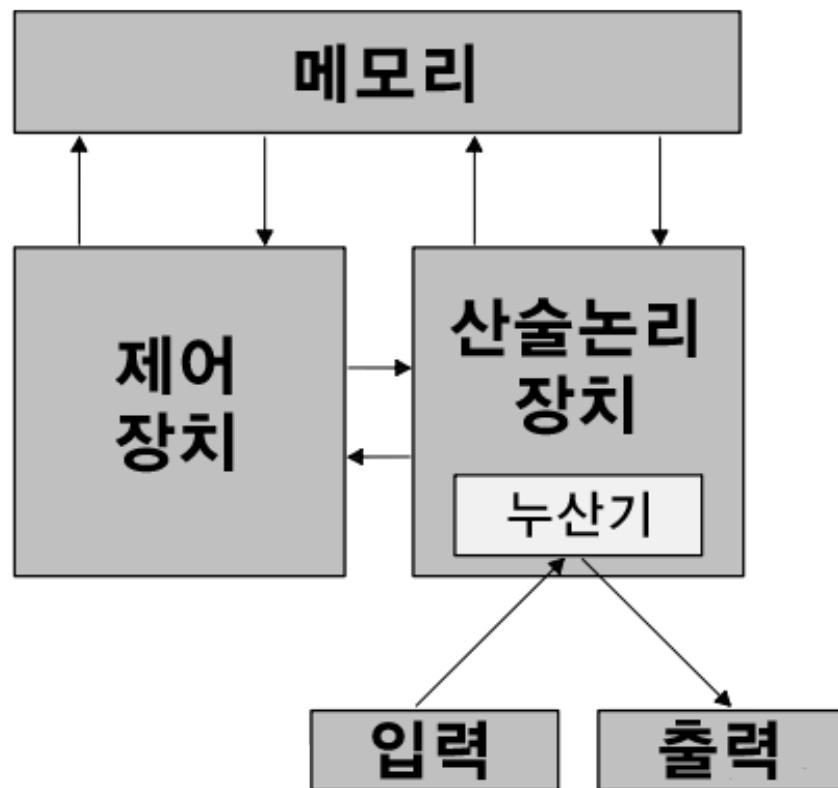


Atanasoff-Berry Computer(1937,1942)  
Iowa State University

ENIAC (1946)  
University of Pennsylvania



# 1950년대: 돈 노이만 컴퓨터



Kurt Gödel



John von Neumann

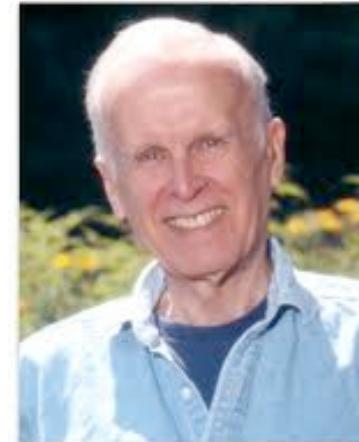
- 기계어 : 프로그램을 이진수로 표현하여 메모리에 저장 실행
- 어셈블리 언어 : 어셈블러가 기계어로 번역

# 1950년대:

## 프로그래밍 언어의 출현

- Fortran (Formula Translator)

- IBM의 존 백커스가 설계
- 과학공학 계산용
- 컴퓨터 파일 개발



John Backus

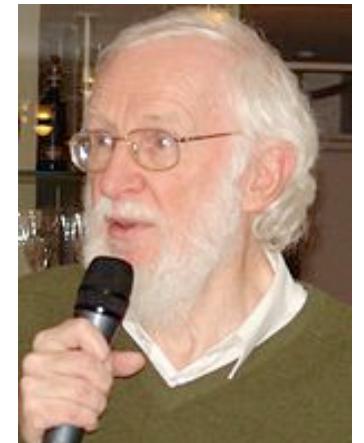
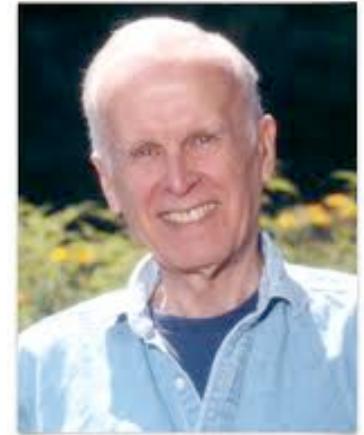
- COBOL (common Business-Oriented Language)

- 비즈니스 데이터 처리 용

# 1960년대: Algol60

John Backus

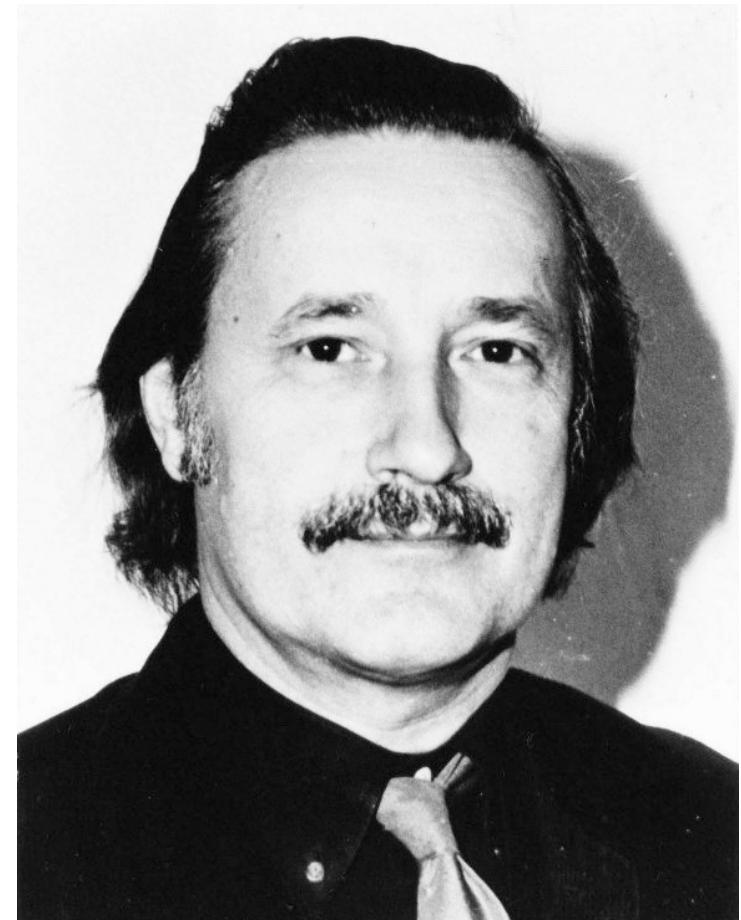
- 유럽의 컴퓨터 학자 그룹이 설계
- 구문을 문법으로 정의: BNF(Backus-Naur Form)
- 최초로 의미를 구曩적으로 설명하여 문서화: 피터 나우어
- 블록구조: 지역변수, 스택 구조를 이용한 자동 메모리 관리
- 재귀 프로시저
- 시대를 앞선 혁신적인 언어였으나  
버로우(Burroughs) 회사의 상용화 실패로 빛을 보지 못함



Peter Naur

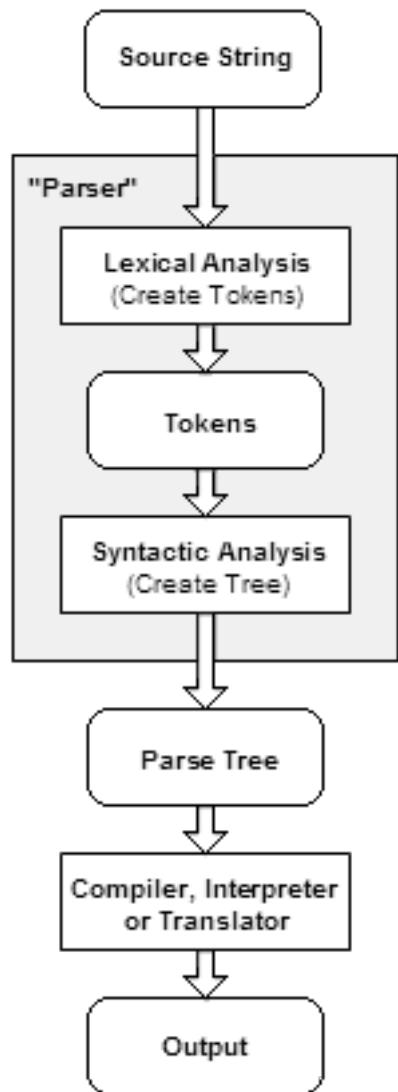
# 1960년대: 시스템 프로그래밍 언어

- 운영체제 등 시스템을 만들기 위한 언어
- CPL: Christopher Strachey
- BCPL
- B
- C



Christopher Strachey

# 1960년대: 파서, 컴파일러



- **파서:** 문법으로 구문 정의, 구문을 검증하는 파서 제작
- **인터프리터:** 실행기
- **컴파일러:** 어셈블리언어로 번역

# 1960년대: 문법

- 문법 분류 연구 활발: 도날드 크누스 주도
  - LL(k): 하향식 파싱 알고리즘
  - LR(k): 상향식 파싱 알고리즘
- LR(k) 기반 파서자동생성기
  - 프로그래밍언어의 프로토 타입 제작 가능



Donald Knuth

# 1960년대: 의미

- 의미의 구현적 정의
- 가상언어(인터프리터): 프로그램을 직접 실행
  - 최초의 가상언어: Vienna Definition Language



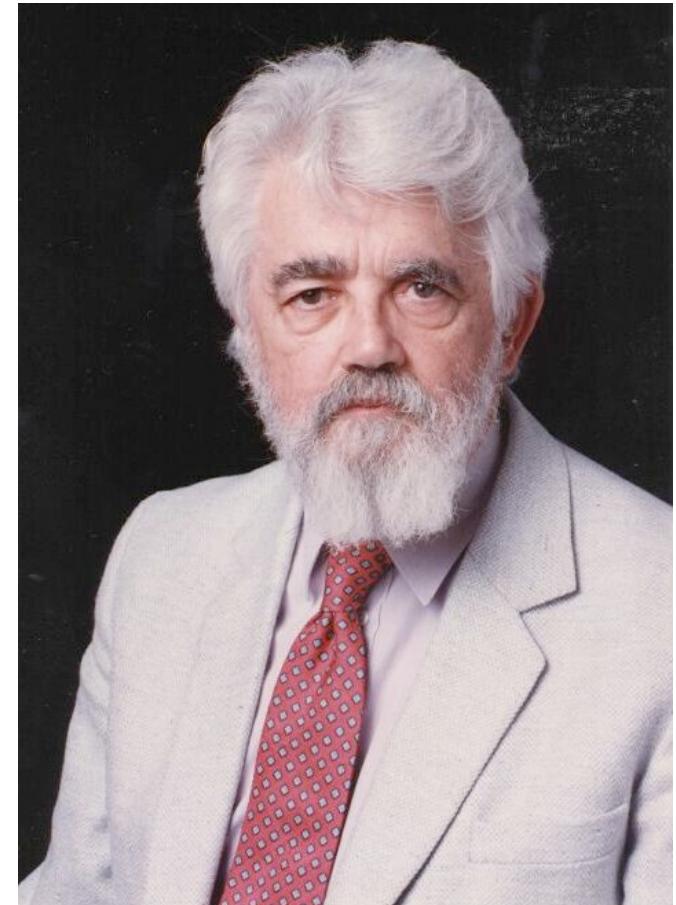
Dines Bjørner



Clifford B. Jones

# 1960년대: LISP

- LIST Processing language
- 자연어처리 용 언어로 개발
- 람다기계산법(lambda calculus) 용용
- LISP 언어 개발: 상업적으로는 실패
- 인공지능 언어로 널리 사용



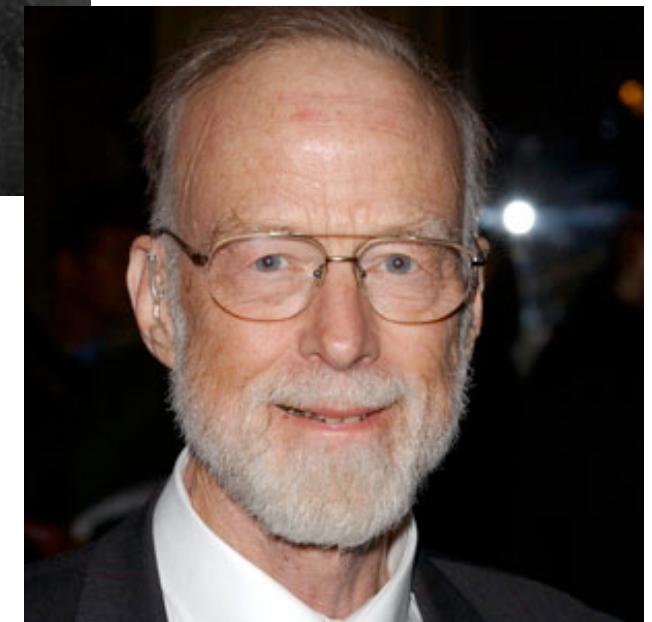
John McCarthy

# 1960년대: 푸로이드-호어 논리

- 프로그램을 지식의 변환기로  
인식
- 공리적 의미론  
axiomatic semantics
- 프로그램의 명시와 검증을  
엄밀하게 하는 틀에 마련



Robert W. Floyd

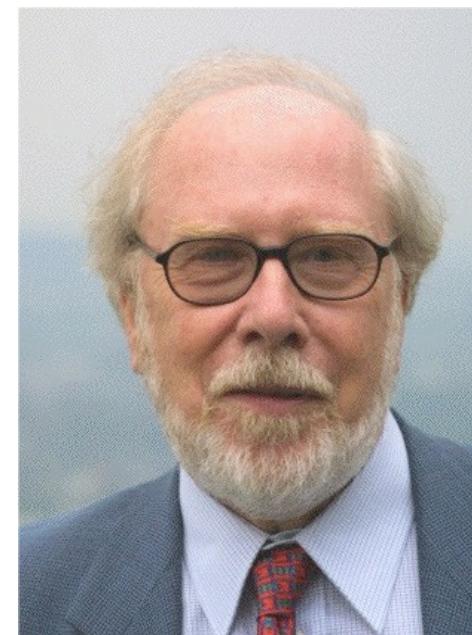
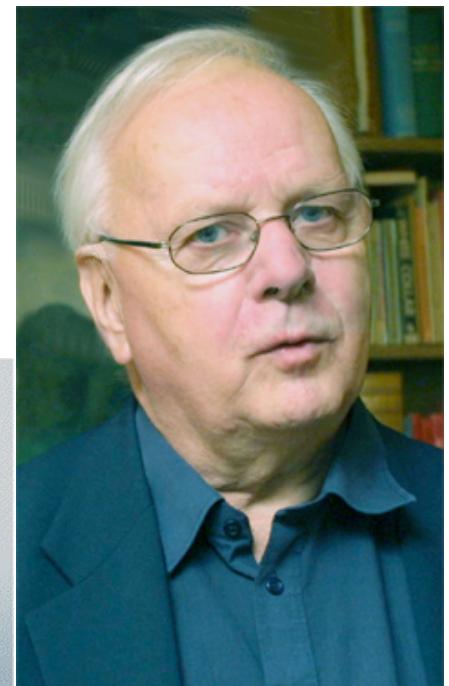


Tony Hoare

# 1970년대: 지속적 발전

- 문법
  - SLR(k), LALR(k) 발전
  - 파서 문제 완전 해결
- Simula: Kristen Nygaard
  - 최초의 객체지향언어
- Pascal, Modula: Niklaus Wirth
  - 가상머신 개념으로 컴퓨터구조 표준화 시도

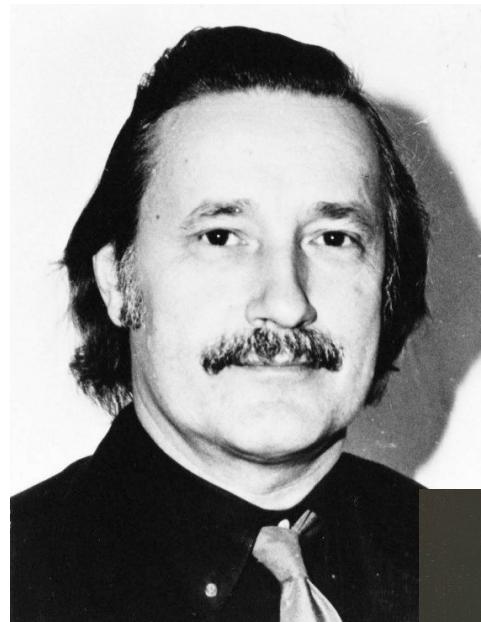
Kristen Nygaard



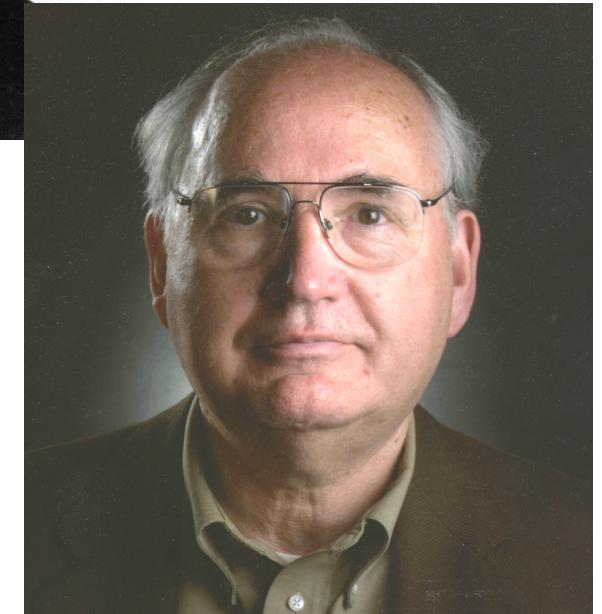
Niklaus Wirth

# 1970년대: 의미론

- 수학적 의미론  
denotational semantics
  - 수학 함수로 (함수표기법으로)  
프로그램의 의미를 표현
- 도메인 이론
  - 계산 기호의 캐릭터 구조 발전
- 함수형 언어 발전 유발
  - ML, Ocaml, Haskell, ...



Christopher Strachey



Dana Scott

# 1970년대: 속성 문법

- 파스트리의 마지막에 속성  
(attribute)으로 의미를 부  
착
- 프로그램 전체의 의미는 하위  
파스트리의 의미를 조합하여  
결정



Donald Knuth

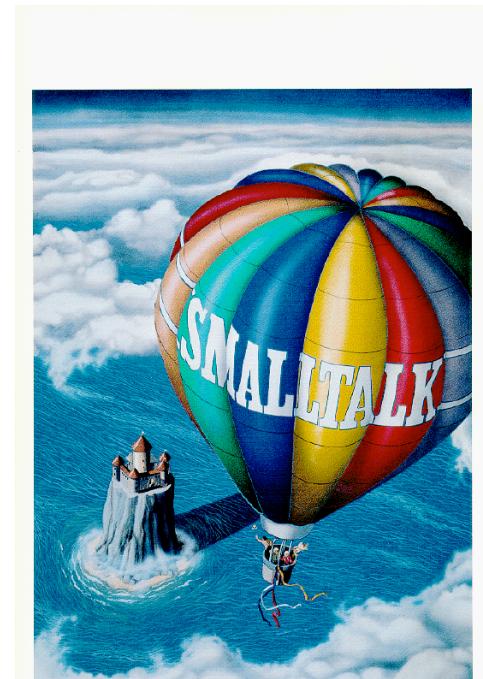
# 1980년대: 객체지향 패러다임 등장

- Simula, Ada, Euler

- 소프트웨어의 부풀화, 소프트웨어의 개발은 부풀의 조립
- 재사용과 부풀의 결합을 용이하게 하기 위한 기술 개발 활발

- Smalltalk

- 객체지향 프로그램의 시조
- 행위자(객체)끼리의 메시지 전달
- 이벤트 처리를 통하여 자연스러운 GUI 조립



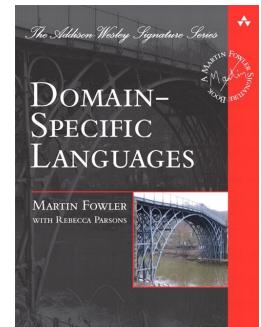
# 1980년대: 논리와 계산

- 소프트웨어 검증을 위해서 논리 사용
- 직관 논리: 명세논리식 = 타입, 증명 = 프로그램
- 논리증명시스템: NUPRL, LCF, HOL, Isabelle, coq 등
- 논리증설언어: **Prolog**
  - Horn-clause 논리식에 대한 정리증명기
  - 명세 = 프로그램, 명세의 증명 과정 = 프로그램의 실행
  - 인공지능, 데이터베이스 분야에 사용

# 1990년대: 도메인특화 언어

- 특정 용도분야 전용 언어

- GUI구축용, 웹문서작성용, 네트워킹, 수학기계산, 표만들기, ...



- Java

- 침입 강자용 프로그램 작성용으로 개발

- 가상머신에서 실행

- 웹언어로 확장: 애플리케이션에서 실행

- 범용 언어로 진화: C++의 대안



# 1990년대: 분산, 동시 계산

- 실행과정 의미론  
operational semantics
  - 큰걸을 표현방식
  - 작은걸을 표현방식
- CCS
- Object calculus
- Process Algebra

Gordon Plotkin



Gilles Kahn



Robin Milner



Luca Cardelli



Martín Abadi

## 1990년대: UML



- 소프트웨어구조를 표현  
하기 위한 그림언어

2000년대 이후:

언어의 지속적 출현과 프로그램 검증

- 새로운 언어의 지속적 출현
  - C#, F#, Scala, Swift, ...
  - Ajax, XML, Python, Ruby, JavaScript, ...
- 안전제어 애플리케이션
  - 가능성 + 보안성 검증
  - 정적의리분석 도구
  - 모델검사 도구

# 에필로그

- 개발할 소프트웨어의 목적과 환경에 잘 맞는 프로그래밍언어를 고르자.
- 맞는 게 없으면 하나 만들어야 할지도 모른다.
- 결국 양질의 안전한 소프트웨어를 개발하기 위해서는 프로그래밍언어의 원리를 제대로 이해하고 있어야 한다.