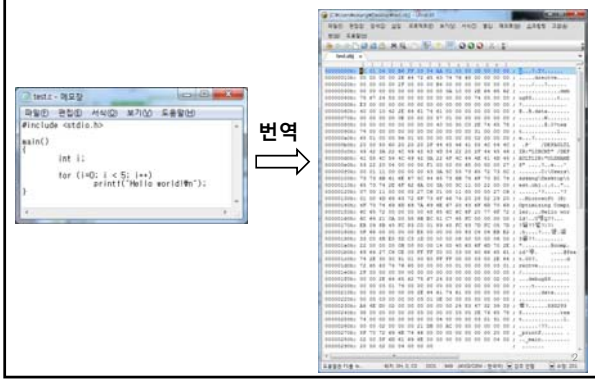
컴파일러 3학년2학기

- 강승식: sskang@kookmin.ac.kr
- 연구실: 7호관 719호

- 교재
  - 컴파일러와 오토마타 -이론 및 실습-, 국민대학교 출판부
- 강의자료
  - http://cafe.daum.net/sskang-compiler

1

컴파일러의 역할



2

컴파일러 과목에서 배우는 내용

- 컴파일러 개발 이론 및 실습
  - 오토마타 이론
    - Finite Automata, Pushdown Automata
- Parsing 방법과 문법이론
- 컴파일러 작성 도구
  - Lex, Yacc

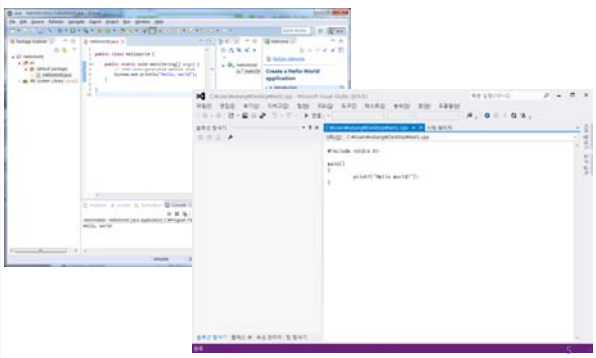
3

컴파일러: 1장

국민대학교 컴퓨터공학부
강 승 식

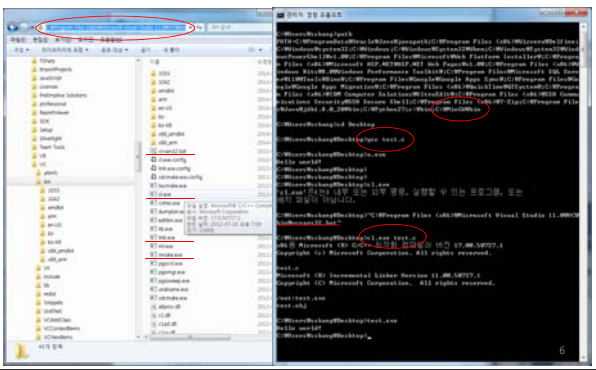
4

컴파일 환경? IDE



5

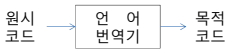
gcc와 cl.exe



6

제1장 컴파일러 소개

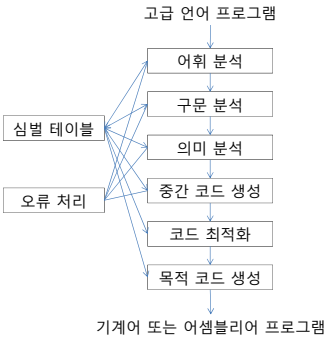
- 언어 번역기(language translator)
  - 원시 프로그램(source code)
    - 고급언어 프로그램 소스
  - 목적 프로그램(target code)
    - 저급언어(기계어, 어셈블리어) 프로그램
  - 언어 번역기
    - 컴파일러, 인터프리터, 프리프로세서,
    - 어셈블러, 교차 컴파일러(cross compiler) 등



- 컴파일러/인터프리터의 역할
  - 입력: 고급언어 프로그램
    (원시 코드, source code)
  - 출력: 기계어/어셈블리어 프로그램
    (목적 코드, target code)
- 컴파일러와 인터프리터의 차이점/장단점은?
  - 둘 다 제공하는 언어도 있음

컴파일 과정

- 컴파일 과정
  - 전단부(front-end): 분석
  - 후단부(back-end): 생성
- 전단부(분석 과정)
  - 어휘 분석(lexical analysis): 토큰 단위로 구분
  - 구문 분석(syntactic analysis, parsing): parse tree 생성
  - 의미 분석(semantic analysis)
- 후단부(생성 과정)
  - 중간코드 생성: machine independent
  - 코드 최적화(code optimization)
  - 목적 코드 생성



어휘 분석(lexical analysis)

- 어휘분석기(lexical analyzer) 또는 스캐너 (scanner)
  - 입력: 원시 코드(source code)
  - 출력: 토큰 열(token sequence)
- 토큰(token): 의미가 있는 최소 단위
- 예) my_array[index+1] = x + 100;

구문 분석(syntax analysis)

- 구문분석(syntax analysys) 또는 파싱(parsing)
- 구문분석기(syntax analyzer) 또는 파서(parser)
- 구문 분석이란?
  - 문장의 구조 분석, 트리 형태로 출력
- 입력
  - 토큰 열(token sequence): 어휘 분석 결과
- 출력
  - 파스 트리(parse tree) 또는 구문 트리(syntax tree)

소프트웨어 개발 환경

- 소프트웨어 개발 환경
  - 컴파일러/인터프리터, 전처리기(pre-processor)
  - 어셈블러, 링커(linker), 로더(loader),
  - 편집기(editor), 디버거(debugger), 프로파일러(profiler),
  - 예) 윈도 Visual Studio, Eclipse, GNU Dev CPP
- 프로젝트 관리기(make)
- 컴파일러-컴파일러 시스템(compiler-compiler system)
  - 컴파일러 자동화 도구
  - LEX(어휘분석기 생성기), YACC(파서 생성기)

13

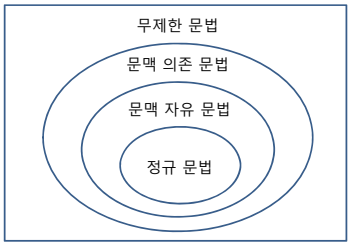
컴파일러 이론: 문법과 오토마타

- 문법(grammar)
  - 촘스키(Noam Chomsky) 분류법: 복잡도에 따라
  - Type 3: 정규 문법(RG: Regular Grammar)
  - Type 2: 문맥자유 문법(CFG: Context-Free Grammar)
  - Type 1: 문맥의존 문법(CSG: Context-Sensitive Grammar)
  - Type 0: 무제한 문법(UG: Unrestricted Grammar)

14

문법의 포함 관계

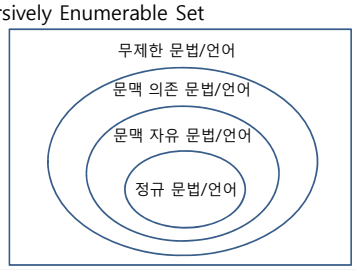
- RG ⊂ CFG ⊂ CSG ⊂ UG



15

언어

- 정규 언어(regular language)
- 문맥자유 언어(context-free language)
- 문맥의존 언어(context-sensitive language)
- Recursively Enumerable Set



16

오토마타(automata)

- 유한 오토마타(Finite Automata): NFA, DFA
  - 정규문법
- 푸시다운 오토마타(Pushdown Automata)
  - 문맥 자유 문법
- 선형제한 오토마타(Linear Bounded Automata)
  - 문맥 의존 문법
- 튜링 기계(Turing Machine)
  - 무제한 문법

17

기타: 컴파일러 자료구조 등

- 심볼 테이블(symbol table), 리터럴 테이블(literal table)
- 오류 처리(error message) 기법

18