Design and Development of Compiler  
for C- Language

**Phase I**

**Design and Implementation**

**of Lexical Analyzer**

**Project Proposal**

과목명: CSE4120 기초 컴파일러 구성

담당 교수: 서강대학교 컴퓨터공학과 정성원

개발자: 20141589 최광희

제출일: 2017. 3. 20.

1. **개발 목표**

전체 C- Compiler 설계 프로젝트 중 Phase I의 개발 목표는 Lexical Analyzing of C- Language로, C- Language로 쓰인 소스 코드를 Token 단위로 쪼개 분석할 수 있도록 한다.

1. **개발 범위 및 내용**

**1. 개발 범위**

Lexical Analysis는 주어진 C- 언어 소스 코드에 대해 각 Token별로 쪼개기 위해, 각 Token을 나타내는 Regular Expression을 통해 생성한 Automation을 모조리 묶어 하나의 Deterministic Finite State Automation을 만드는 것을 말한다. 이 과정에서 Regular Expression으로부터 Automation을 생성하는 과정, Nondeterministic Finite State Automation을 Deterministic Finite State Automation으로 변환하는 것 등의 과정은 flex라는 Software가 대신 생성하고, 이번 Phase에서는 오직 flex가 이해할 수 있는 Lexical Specification 소스 코드를 작성한 뒤, flex가 해당 코드를 변환시켜준 lex.yy.c를 통해 주어진 소스 코드를 Token 단위로 쪼개 출력할 수 있도록 한다.

**2. 개발 내용**

C- 언어는 C 언어의 Subset으로써, 단순한 문법 구조를 갖고 있다. Keyword는 총 6개, else / if / int / return / void / while 이 존재한다. 특수 Symbol들의 경우 + - \* / < <= > >= == != = , . ( ) [ ] { } /\* \*/ 가 있다. ID와 NUM이라는 Token이 존재하는데, ID는 오직 Alphabet, NUM은 오직 0~9 사이의 숫자로만 되어 있다. White space는 seperation 이외에는 의미를 갖지 않고, 주석의 경우 /\* \*/ 형태로, 흔히 C에서 사용하는 형태로 되어 있다.

이번 Phase에서는 먼저, 해당 언어의 분석을 통하여 BNF Grammar로 변환시키고, 해당 Grammar을 바탕으로 flex가 이해할 수 있는 Lexical Specification 소스 코드를 작성한다.

위를 완성한 뒤에는, flex가 변환을 한 뒤에 나오는 결과인 lex.yy.c 파일에 생성된 yylex() 함수를 이용하여, C- 언어로 이뤄진 소스 코드를 Lexically Analyze한다. 이 과정에서 교재에 주어진 main.c , global.h , util.h , util.c 를 적절히 작성하여 Lexical Analyzer을 설계한다.

그 과정에서 필요한 C- 언어로 이뤄진 소스 코드 역시 작성한다.

1. **추진 일정 및 개발 방법**

**1. 추진 일정**

3/18 ~ 3/20: 개발해야 할 C- Language에 대해 분석 및 BNF Grammar 확인

3/20 ~ 3/21: dotfiles, git @ Github + Travis CI 및 Makefile 작성 등의 개발환경 세팅

3/21 ~ 3/23: Lexical Specification 설계

3/23 ~ 3/25: lex.yy.c의 yylex() 함수를 이용한 Lexical Analyzer 설계

3/25 ~ 3/27: Test case 작성 및 Travis CI와의 연동

**2. 개발 방법**

개발 환경은 cspro9와 cspro10을 사용하여, Ubuntu 16.04.2 LTS 기준으로 작업한다. vi editor을 이용하여 편집을 하며, Indentation은 GNU Standard를 따른다. 주 개발 언어는 C이나, 이번 Phase의 경우 Lexical Specification 소스 역시 작성한다. 컴파일은 gcc 5.4.0 을 통해 하며, Makefile을 이용하여 빌드 및 Testing의 자동화를 한다. git을 이용하여 Version Control을 하며, Github에서 remote branch를 관리함과 동시에 Travis CI와의 연동을 하여, 매 Commit마다 각 case들에 대해 검증한다.

1. **기타**