Design and Development of Compiler  
for C- Language

**Phase II**

**Design and Implementation**

**of LALR Parser**

**Project Proposal**

과목명: CSE4120 기초 컴파일러 구성

담당 교수: 서강대학교 컴퓨터공학과 정성원

개발자: 20141589 최광희

제출일: 2017. 4. 26.

1. **개발 목표**

전체 C- Compiler 설계 프로젝트 중 Phase II의 개발 목표는 LALR Parsing of C- Language로, C- Language로 쓰인 소스 코드를 Parsing할 수 있도록 한다. 구체적으로, LALR Parser을 만들어 Abstract Syntax Tree를 출력하는 것을 목표로 한다.

1. **개발 범위 및 내용**

**1. 개발 범위**

Phase II는 주어진 C- 언어 소스 코드를 Syntactic Analysis 과정을 거쳐 Stream of tokens를 Abstract syntax tree로 바꿀 수 있는 Software을 개발한다. Stream of tokens는 이전 Phase I의 Lexical Analysis를 통해 만들어낸 token과 일맥상통한다. 여기서 Parsing이란, 해당 stream of token이 grammatically well-formed인지 확인하는 것을 말한다. 이 과정에서 C- Language의 Context-free Grammar Production rule들로부터 LALR Parsing Table을 생성하는 과정은 bison이라는 Software가 대신 생성하므고, 이번 Phase에서는 bison이 이해할 수 있는 Production Rule form인 Backus-Naur form으로 소스 코드를 작성한 뒤, bison이 해당 코드를 변환시켜준 C 소스 코드를 통해 주어진 소스 코드를 Abstract Syntax Tree로 출력할 수 있도록 한다.

**2. 개발 내용**

C- 언어는 C 언어의 Subset으로써, 단순한 문법 구조를 갖고 있다. 이번 Phase에서는 먼저, 해당 언어의 분석을 통하여 BNF Grammar로 변환시키고, 해당 Grammar을 바탕으로 bison이 이해할 수 있는 Lexical Specification 소스 코드를 작성한다. 해당 BNF Grammar은 Compiler 수업의 교재에 작성되어있으므로, 해당 BNF Grammar을 Backus-Naur form으로 바꿔 cm.y 파일을 작성하면 된다.

위를 완성한 뒤에는, cm.y 파일을 bison이 변환을 한 뒤에 나오는 결과인 cm.tab.c 파일에 생성된 함수를 이용하여, C- 언어로 이뤄진 소스 코드를 Syntactically Analyze한다. 이후 Tree를 출력하기 위해 Phase I의 소스 코드를 이어서 작성하는데, 특히 printTree 함수를 수정하고, 만약 Parsing 시에 오류가 있다면 해당 오류 메세지와 Line number을 출력하는 프로그램을 만든다.

그 과정에서 필요한 C- 언어로 이뤄진 소스 코드 역시 작성한다. 정상 코드와 Syntactic Error가 있는 코드 두 개를 작성해야 한다.

1. **추진 일정 및 개발 방법**

**1. 추진 일정**

4/26 ~ 4/28: 개발해야 할 C- Language의 Production Rule의 BNF Grammar 확인

4/28 ~ 4/29: dotfiles, git @ Github 및 Makefile 작성 등의 개발환경 세팅

4/29 ~ 5/3: Syntactic Specification 설계

5/3 ~ 5/7: cm.tab.c와 printTree를 이용한 AST 출력 설계 및 예외 처리

5/7 ~ 5/12: Test case 작성 및 디버깅과 보고서 작성

**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 Syntactic 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 관리하여, 매 Commit마다 각 case들에 대해 검증한다.

1. **기타**