Design and Development of Compiler  
for C- Language

**Phase III**

**Design and Implementation**

**of Semantic Analyzer**

**Project Proposal**

과목명: CSE4120 기초 컴파일러 구성

담당 교수: 서강대학교 컴퓨터공학과 정성원

개발자: 1조 20141500 권태국, 20141570 이태승, 20141589 최광희(팀장)

제출일: 2017. 5. 22.

1. **개발 목표**

전체 C- Compiler 설계 프로젝트 중 Phase III의 개발 목표는 Semantic Analysis of C- Language로, C- Language로 쓰인 소스 코드의 Semantic Analysis를 할 수 있도록 한다. 구체적으로, Symbol Table을 만들기 위하여 Variable과 Function의 Name equivalence를 이용한 Type checking을 하는 것을 목표로 한다.

1. **개발 범위 및 내용**

**1. 개발 범위**

Phase III는 Phase II에서 구현한 C- parser를 이용하여, semantic analyzer를 구현하는데, 결과적으로 Symbol table을 출력할 수 있어야 한다. 각각의 Table은 개별 Scope를 대표하며, 각 Scope에 있는 Variable과 Parameter, Function에 대한 정보를 출력해주어야 한다. 구체적으로, 각 AST Node의 이름, Scope, Stack pointer로 부터 Location Offset (Loc), Node의 종류(Variable, Parameter, Function), Array인지의 여부, Array인 경우 Size, Type(int, void), Line Numbers(등장하는 코드상의 위치) 등을 출력해야 한다.

**2.개발 내용**

C- 언어는 C 언어의 Subset으로써, 단순한 문법 구조를 갖고 있다. 이번 Phase에서는 먼저, 해당 언어의 분석을 통하여 BNF Grammar을 만족하지만 Semantic하게는 오류가 생기는 코드를 확인할 수 있어야 한다. Phase II에서 개발한 LALR Parser가 산출하는 Abstract Syntax Tree를 가지고 Semantic Analyzer를 만들고, 결과로 Symbol Table을 출력한다. Symbol Table는 각 Symbol의 Scope를 고려할 수 있게 작성한다. 구체적인 출력 내용은 위 1. 개발 범위에서 다룬 바와 같다.

전체 Semantic Analyzer의 개발은 크게 2단계로 수행된다. 첫 번째 단계는 Symbol Table을 작성하고 Scoping error를 Check 하는 단계이고, 두 번째 단계는 Name equivalence를 이용한 Type Check와 그 외 Semantic error를 Check하는 단계이다.

다음과 같은 오류 체크 사항이 있다.

* 선언되지 않은 변수나 함수는 사용 불가.
* 변수나 함수, 함수 파라미터 선언 시 중복된 이름 검사.
* 변수나 Parameter 선언 시 void type으로 선언할 수 없음.
* 변수나 Parameter 선언 시 Array 변수인지 확인해야 함.
* 변수에 값을 assign 하는 경우 type에 대한 검사.
* Array 변수의 경우 Array index가 int가 맞는지 확인.
* array가 아닌 변수를 array처럼 사용하려 하는 지 검사.
* 함수 호출 시 Parameter의 개수와 type이 일치하는지 검사.
* 함수 호출 시 호출된 것이 함수가 맞는지 (즉 변수 등이 아니었는지) 확인.
* 함수의 return 값과 return type이 일치하는지 검사
* 함수의 return type이 void일 경우 return이 있는지 확인. (없어야 함)
* main함수는 가장 마지막에 선언되어야 함.
* main함수는 반드시 void로 선언되어야 함.
* main함수는 반드시 Parameter가 없어야 함.
* loop문의 반복 조건을 표시하는 부분에 오는 statement는 int값을 가져야 한다.

그 과정에서 필요한 C- 언어로 이뤄진 예제 코드 역시 작성한다. 해당 코드는 에러가 없어야 하며, 존재하는 모든 Keyword가 포함되어야 하고, Nested scope가 3-Level 이상, 각 Scope마다 Local 변수가 선언되어야 한다.

1. **추진 일정 및 개발 방법**

**1. 추진 일정**

5/21 ~ 5/22 : 팀원간 역할 분배 및 이전 Project의 개발 내역을 합성

5/22 ~ 5/25 : Symbol Table 구현

5/26 ~ 5/29 : Name equivalence를 이용한 Type Check와 그 외 Semantic Error의 Check 구현

5/30 ~ 6/1 : 테스트 및 코드 품질 향상.

6/2 : 보고서 작성 및 제출.

**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**