# Design and Development of Compiler for C- Language

과목명 : [CSE4120] 기초 컴파일러 구성
담당교수: 서강대학교 컴퓨터공학과 정 성 원
개발자: 이성준
개발기간: 2019.3.24 ~ 2019.3.27

---

## 1. 개발 목표

- C- Language로 설계된 코드로부터 flex를 이용해 lexical analyzer를 생성하고, lexical analyzer을 통해 token stream을 생성하는 프로그램을 개발한다.

## 2. 개발 범위 및 내용

가. 개발 범위

- Unix platform에서 C와 Flex를 이용하여 lexical analyzer를 구현한다. C- 문법에 따른 keyword, special symbol, identifier, comment 등을 구분시킨다.

나. 개발 내용

- 주어진 tiny compiler를 확장시켜 lexical analyzer를 구현한다.
- lexical analyzer의 세부사항
  - keyword : else if int void while
  - special symbol : + - * / < <= > >= == ; = ( ) { } [ ]
  - ID and NUM
  - white space
  - comment : /* */

## 3. 추진 일정 및 개발 방법

가. 추진일정

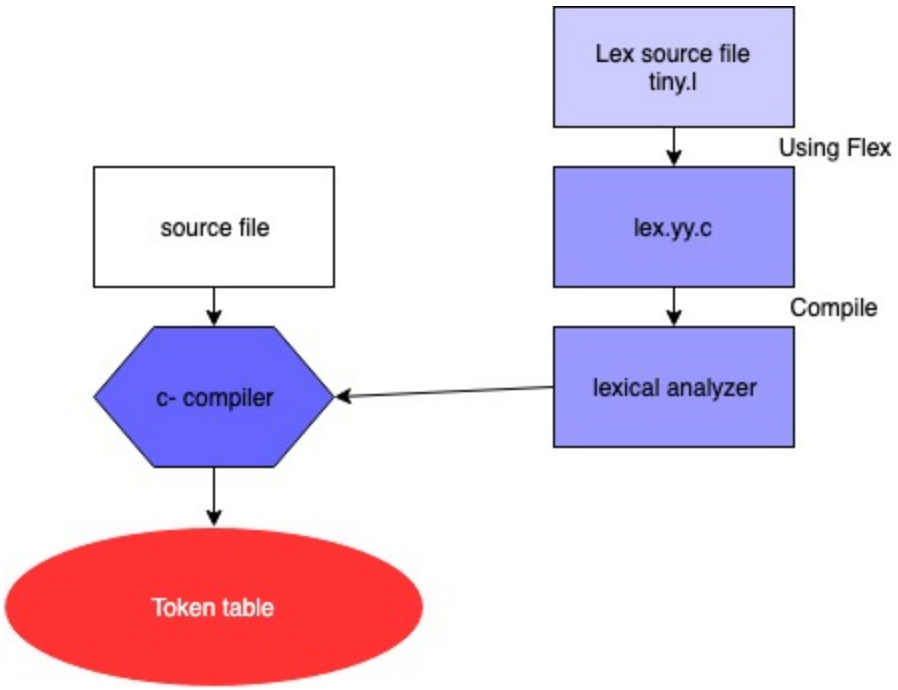
- 3.24 ~ 3.25 : flex 사용법 학습
- 3.25 ~ 3.26 : lexical analyzer 세부 구현
- 3.26 ~ 3.27 : 성능평가 및 report wjdfl

나. 개발방법

- Unix 환경에서 text editor를 통해 projectfmf tngodgksek.
- 교재의 appendix의 코드를 이해하여 lexical analyzer에서 이용되는 함수를 구현한다.

## 4. 연구 결과

1. 합성 내용




*소프트웨어 구성도*

- tiny.l : lex source file
- flex : lexical analyzer 생성기

2. 분석 내용

| Token | Regular Expression |
| --- | --- |
| | |

| | |
|---|---|
| digit | [0 - 9] |
| number | {digit}+ |
| letter | [a-zA-Z] |
| identifier | {letter}+ |
| newline | \n |
| whitespace | [ \t]+ |
| IF | if |
| ELSE | else |
| INT | int |
| WHILE | while |
| RETURN | return |
| EQ | = |
| LT | < |
| LTEQ | <= |
| EQEQ | == |
| RTEQ | >= |
| RT | > |
| PLUS | + |
| MINUS | - |
| TIMES | * |
| OVER | / |
| LBRACE | { |
| RBRACE | } |
| LBRACKET | [ |
| RBRACKET | ] |
| LPAREN | ( |

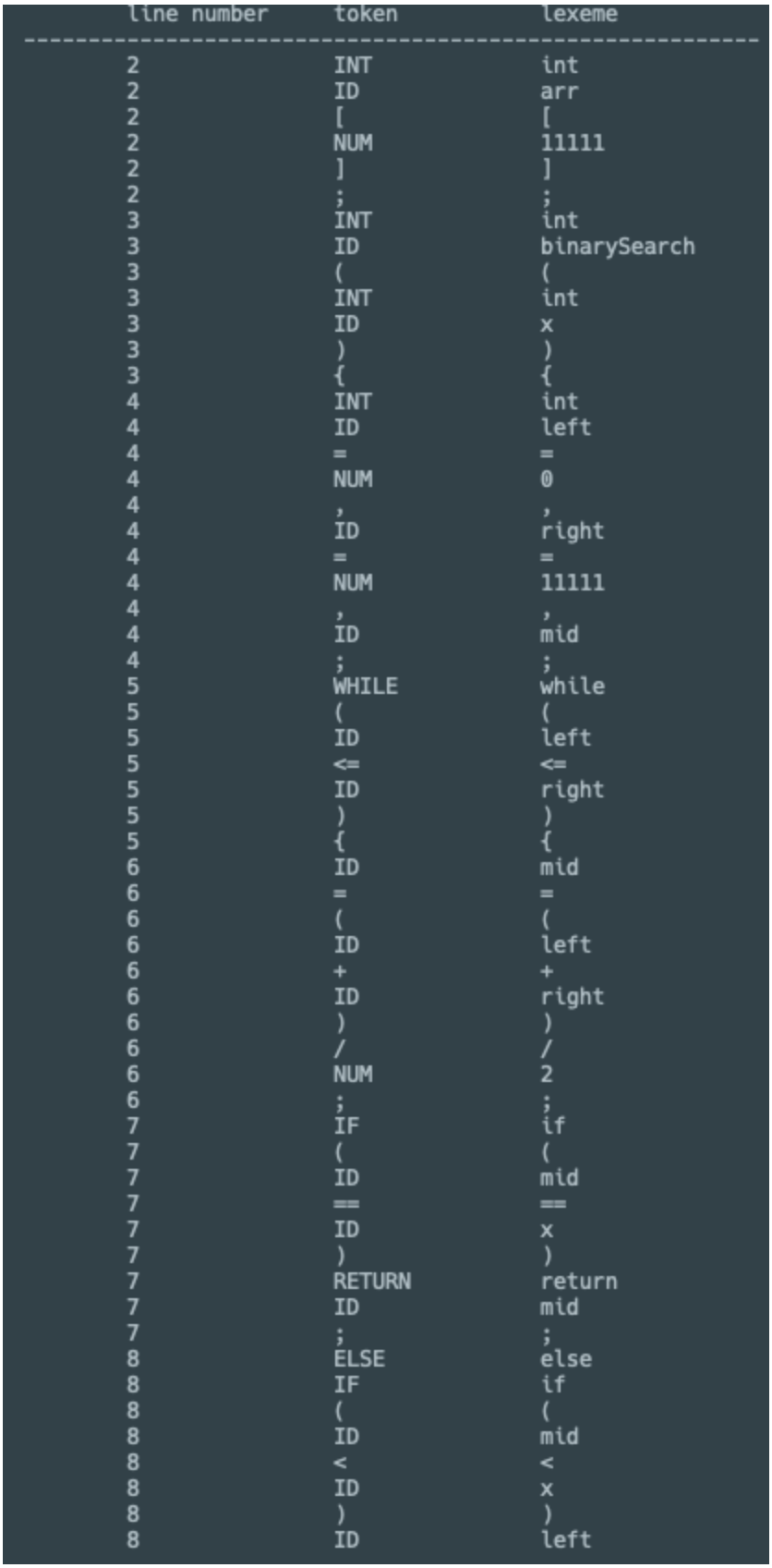
| | |
|---|---|
| RPAREN | ) |
| SEMI | ; |
| COMMA | , |

- util.c를 수정하여 알맞은 형식의 token이 출력되기 하였다.
- == 와 = 가 있을때 우선 인식 순위를 ==로 하게 하여 token 인식에 문제가 없게 한다.
- comment 처리는 /* */ 사이에 들어오게 한다. (*이 중첩된 경우는 comment로 인식되지 않는다.

3. 제작 내용

- tiny.l : 교제의 appendix에서 추가로 본 프로젝트에 맞게 lexical specification을 작성, comment 처리, getToken() 함수 구현을 하였다
- lex.yy.c : flex로 인해 tiny.l 파일이 c로 변환된 형태
- main.c : C- compiler의 main 함수이며 lexical analysis만 수행하기 위해 parse나 analyzing 기능은 false 처리 하였다.
- util.c : 각 토큰에 대한 lineno, token, lexeme을 출력하는 함수를 담고있다.
- global.h : 프로그램의 전역 변수를 포함하는 헤더파일이다. 본 프로젝트에 맞게 TokenType을 수정하였다.
- 20121622 : lexical analyzing 기능을 수행하는 실행파일이다.

4. 시험 내용



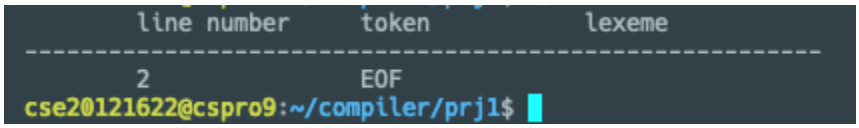
| line number | token | lexeme |
|---|---|---|
| 2 | INT | int |
| 2 | ID | arr |
| 2 | [ | [ |
| 2 | NUM | 11111 |
| 2 | ] | ] |
| 2 | ; | ; |
| 3 | INT | int |
| 3 | ID | binarySearch |
| 3 | ( | ( |
| 3 | INT | int |
| 3 | ID | x |
| 3 | ) | ) |
| 3 | { | { |
| 4 | INT | int |
| 4 | ID | left |
| 4 | = | = |
| 4 | NUM | 0 |
| 4 | , | , |
| 4 | ID | right |
| 4 | = | = |
| 4 | NUM | 11111 |
| 4 | , | , |
| 4 | ID | mid |
| 4 | ; | ; |
| 5 | WHILE | while |
| 5 | ( | ( |
| 5 | ID | left |
| 5 | <= | <= |
| 5 | ID | right |
| 5 | ) | ) |
| 5 | { | { |
| 6 | ID | mid |
| 6 | = | = |
| 6 | ( | ( |
| 6 | ID | left |
| 6 | + | + |
| 6 | ID | right |
| 6 | ) | ) |
| 6 | / | / |
| 6 | NUM | 2 |
| 6 | ; | ; |
| 7 | IF | if |
| 7 | ( | ( |
| 7 | ID | mid |
| 7 | == | == |
| 7 | ID | x |
| 7 | ) | ) |
| 7 | RETURN | return |
| 7 | ID | mid |
| 7 | ; | ; |
| 8 | ELSE | else |
| 8 | IF | if |
| 8 | ( | ( |
| 8 | ID | mid |
| 8 | < | < |
| 8 | ID | x |
| 8 | ) | ) |
| 8 | ID | left |

*test1.c 에 대한 실행 내용*

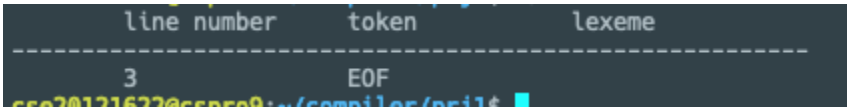
test1.c에 대해서 위와같이 프로그램이 정상 작동하는 것을 알 수 있다.

또한 본 프로그램에서 요구한 comment 요구조건도 다음과 같이 만족한다.

1. /*  comment  */




2. /* comment
   comment  */




3. /*  comment


4. /* comment /*


5. 평가내용

- 교재의 appendix를 참고해 lexical analyzer를 제작하였다. C- language 의 특성상 주요한 symbol들에 관해서만 reserver word로 처리하였다. 프로젝트를 진행하면서 큰 문제는 없었다. 본 프로젝트는 flex를 이용하여 regular expression으로만 analyze를 수행하기 때문에 ID를 오류없이 골라내었고 프로그램의 신뢰성 및 안정성을 확인할 수 있었다.

# 4. 기타

1. 자체평가

- 정규표현식을 이용해 flex를 만들고 이를 이용해 lexical analyzer를 구현할 수 있었다. 또한 프로그램의 요구사항에 맞게 comment에 대한 예외 처리도 하였다. 또한 token의 우선순위에 맞게 token의 순서를 올바르게 배열하였다. ( == > =) 이로인해 어떠한 예외 사항에서도 본 규칙에 해당하면 문제가 없게 제작하였다.

2. 기타 본 설계 프로젝트를 수행하면서 느낀 점

- 처음 프로젝트를 진행할때 어떻게 진행해야할지 어려움이 있었다. 우선 flex의 사용법을 찾고 교재의 appendix를 코드화하면서 문제를 차근히 해결할 수 있었다.