각 단계별(현 개발 단계) 결과 보 고 서

/\*

본 단계별 결과 보고서 작성의 목적은 자신들의 연구 개발 내용을 정확히 기술하고 그에 대한 자체 평가를 통하여 해결책을 제시하는 과정을 수행해보는 것임.

\*/

과목명 : [CSE4120] 기초 컴파일러 구성

프로젝트 제목: Design and Development of Compiler for C- Language:

Phase 2: Design and Implementation of LALR Parser

지도 교수 : 서강대학교 컴퓨터공학과 정성원

제출일: 2017.05.12.

개발자: 20141575 임형석

I. 개발 목표

- 내용이 변경되지 않았을 경우 “단계별 제안서 I. 개발 목표 참조”와 같이 기술할 것.

- 개발 목표가 중간보고서 작성 당시의 내용과 다를 경우 그 내용 및 변경 사유를 상세히 기술할 것. 가급적 개발 목표는 변경하지 말 것.

단계별 제안서 I. 개발 목표 참조

II. 개발 범위 및 내용

- 내용이 변경되지 않았을 경우 “단계별 제안서 II. 개발 범위 및 내용 참조”와 같이 기술할 것.

- 개발 범위 및 내용이 단계별 제안서 작성 당시의 내용과 다를 경우 그 내용 및 변경 사유를 상세히 기술할 것.

가. 개발 범위

단계별 제안서 II. 개발 범위 및 내용 참조

나. 개발 내용

단계별 제안서 II. 개발 범위 및 내용 참조

III. 추진 일정 및 개발 방법

- 내용이 변경되지 않았을 경우 “단계별 제안서 III. 추진 일정 및 개발 방법 참조”와 같이 기술할 것.

- 개발 추진 일정, 개발 방법, 역할 분담 등이 중간보고서 작성 당시의 내용과 다를 경우 그 내용 및 변경 사유를 상세히 기술할 것.

가. 추진 일정

“단계별 제안서 III. 추진 일정 및 개발 방법 참조

나. 개발 방법

“단계별 제안서 III. 추진 일정 및 개발 방법 참조

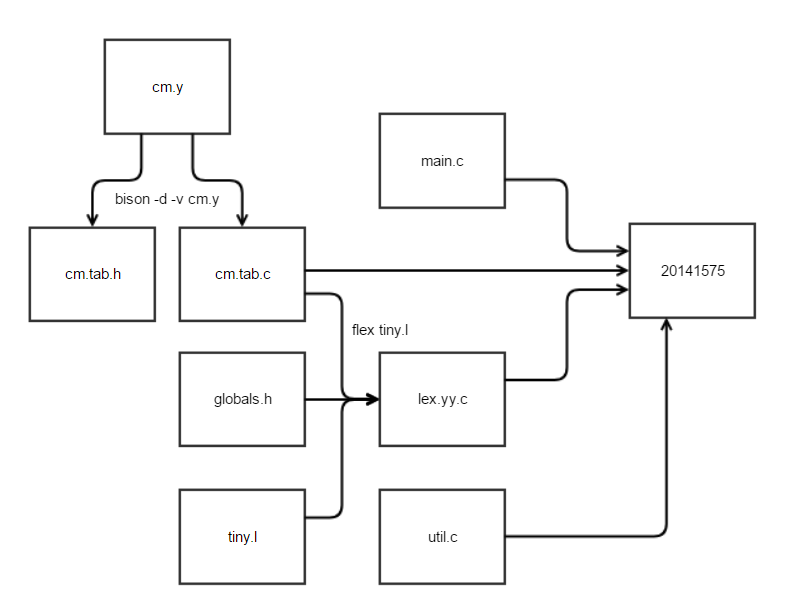
IV. 연구 결과

- 각 단계별 연구 개발 결과를 자유롭게 기술할 것.

- 기초 컴파일러 구성: 다음과 같은 내용을 포함해야함. 그 외의 내용은 자유롭게 기술할 것.

1. 합성 내용: 설계 목표 달성에 필요한 요소 기술에 대하여 조사 분석하여 그들을 바탕으로 구성한 전체 소프트웨어 구성도. 각 부분의 역할 및 사용 기술 및 툴(예: 사용한 이론적 방법, 공개 소프트웨어나 툴 또는 API 등), 구현 방법 등 관련 내용을 기술할 것. 또한 개발 소프트웨어의 사용 방법, 즉 사용자 인터페이스 규약을 명시하고 간단한 사용자 매뉴얼을 작성할 것.

-전체 소프트웨어 구성도



프로그램 실행을 하여 입력을 받으면 yyparse는 yylex를 호출해서 token을 구분해 yyparse로 다시 보내준다. yyparse는 이 token들이 syntax에 맞는 지 확인하고 맞다면 savedTree를 PrintTree로 보내 parse tree를 출력해주고, 맞지 않다면 parse tree를 출력하지 않고 error가 났다는 글과 함께 어느 라인에서 error가 났는지 알려주고 그 때의 token을 출력해준다.

- 각 부분의 역할 및 사용 기술 및 툴

main.c : 전 프로젝트에서 사용하였던 출력 양식은 지우고 parse tree만을 출력시켜준다.

globals.h ; 전에 정의하였던 enum type TokenType을 지워주고 tree를 구성하는 노드들의 종류 즉 Nodekind를 설정해주고 treeNode를 정의해준다.

util.c : 이 프로젝트의 결과물인 parse tree를 출력해주는 printTree를 구현해주어야 하고 tree node를 생성해주는 함수가 있다.

tiny.l : 전 프로젝트와 내용이 대부분 비슷하지만 token이 ID나 NUM일 경우 그 때이 이름을 저장하여 cm.y에서 사용할 수 있도록 해준다.

cm.y : tree를 구성해주는 grammar rule들을 모아준 곳으로 grammar들이 정의되어 있고 이를 통하여 code를 parse tree로 바꾸어준다.

**-프로그램 실행 방법**

1. **linux에 압축파일을 넣은 후 압축을 풀기 위하여 tar -cvf project2\_20141575.tar 라는 명렁어를 입력해주면 project2\_20141575 라는 폴더가 생긴다.**
2. **cd project2\_20141575 라는 명령어로 폴더에 들어간 후 make를 입력하면 20141575 라는 이름을 가진 실행 파일이 생긴다.**
3. **이 20141575라는 파일을 실행시키기 위하여 ‘./20141575 [파일이름] 을 입력하면된다.   
   이 폴더안에서는 ‘./20141575 error.tny’ 와 ‘./20141575 errorfree.tny’를 입력하면 각각 parsing error가 있는 것과 없는 것의 결과가 출력된다.**

2. 분석 내용: 각 구성 요소를 상세히 분석하여 어떠한 방식으로 개발하였는지 그 내용을 기술할 것.

main.c : 이번 프로젝트에서는 parse를 할 것이니 TraceParse 값을 False에서 True로 변경하고, NO\_PARSE 값을 True 에서 False로 변경한다. 그리고 전 프로젝트에서 사용하였던 출력양식을 지워주고, parse()를 호출하여 parse tree를 그려준다.

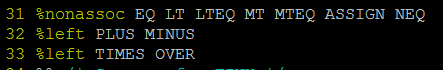
globals.h : tree의 Nodekind를 정의해주기 위해 enum type 6가지(NodeKind,StmtKind,ExpKind, DeclKind,TypeKind,ParaKind,CallKind)를 선언해준다. 그리고 6가지 nodekind들에 대해 각각 element들을 추가해준다. 또한 treenode struct에도 위의 nodekind들을 정의해준다.

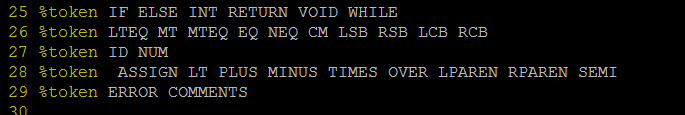
util.c : printToken()을 각 Node의 종류에 따라서 출력해준다. 6가지 nodekind에 대해서 경우를 나누어주고 또 각 nodeKind의 element에 맞추어서 출력해주는데 assign과 array의 경우 각각의 경우에 flag를 설정해주어 assign만 있을 경우 id의 출력을 해주지 않고 바로 값을 출력해주고, array이면서 assign일 경우에는 assign to : Array 라고 출력하고 그 밑에 id : value 이런식으로 출력하게 해주었다. 그리고 tree의 node들을 정의해주는 함수들을 구현하였다.

tiny.l : token이 ID나 NUM일 경우 그 때이 이름을 저장하여 cm.y에서 사용할 수 있도록 해줄 수 있게 구현하였다.

cm.y : 컴파일러 책의 appendix에 소개된 grammar rule들을 입력해 syntax에 맞게 token들이 처리할 수 있게 해주었으며, token들의 우선 순위를 지정해주고 conflict를 해결해줄 수 있도록 구현하였다.

-**Precedence Directives with explanations**

**grammar rule을 따라 parse tree가 만들어 지는 도중에 precedence가 지정되지 않는 다면 원하지 않는 연산을 계산하도록 parse tree가 만들어 줄 수 있기 때문에 다음과 같이 cm.y에서 연산에 대한 우선 순위를 지정해준다.**



**PLUS와 MINUS가 TIMES OVER보다 우선순위가 낮으므로 더 위에 입력해준다.**

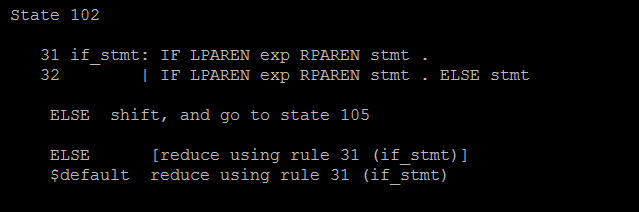
**또한 비교연산자들은 계산 결과가 다 나오고 난 후의 값들을 비교해주어야 하므로 계산연산자들 보다 우선 순위가 더 낮도록 설정해주었다. 또한 token들 간의 우선순위도 위 그림과 같이 설정 해주어 parsing할 때 conflict가 생겨 parse tree가 잘못 나오지 않도록 해주었다.**

**-List of all conflicts with the explanations of how each conflict are resolved.**

**책에 있는 grammar rule에 맞추어 코드를 짜고 나서 프로그램을 실행하였더니**

**cm.output에서 다음과 같은 conflict가 발생하였다.**

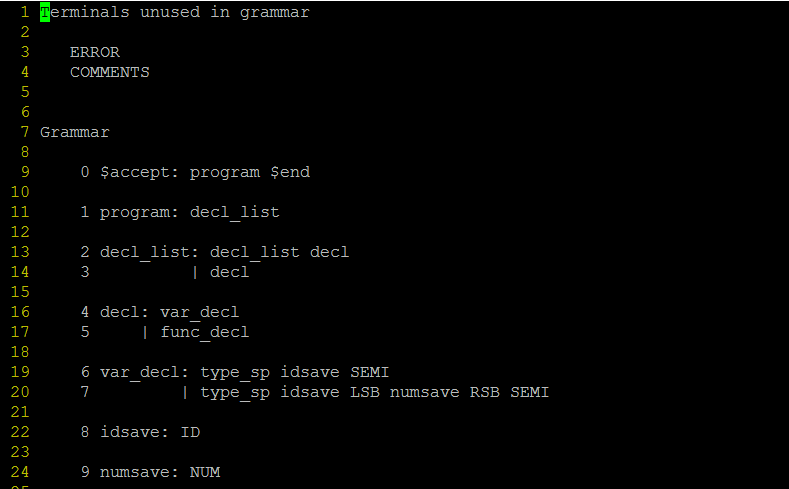




**이 state는 If문을 입력으로 받고 나서 If만 있는 것으로 생각해 reduce를 할지 아니면 else문을 받기 위해 shift를 해야할지에 대하여 conflict가 발생한다. 이를 해결하기 위해서 다음과 같이 코드를 추가해주었다.**



**%nonassoc를 활용해서 ELSE문을 받기 위해 shift해주는 것을 먼저 할 수 있도록 우선권을 부여해주어서 conflict를 해결해주었다. 아래의 cm.output에서 그 결과로 Conflict가 일어나지 않는 것을 알 수 있다.**



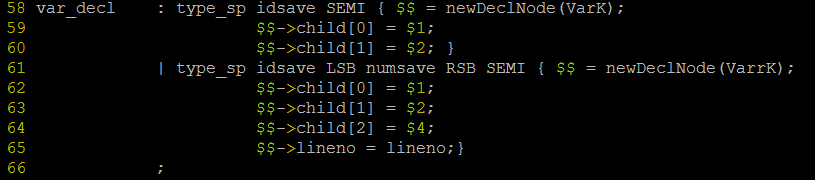
**-Explanation on the implementation including semantic action part**

**semantic action part부분은 cm.y에서 구현되어 있다. 이 때 grammar rule은 교재의 appendix에 적혀있는 29가지의 rule들이다.**

**grammar rule을 구현한 방법을 설명하자면 예를 들어 이런 rule이 있을 때,**



**다음과 같이 구현해 주었다.**



**여기서 var\_decl : type\_sp idsave SEMI 이렇게 되어 있는데 $$는 var\_decl이고, $1은 type\_sp, $2는 idsave가 될 것이다. var\_decl에 대해 일단 어느 노드 종류인지 선언하고 정의주어서 그 노드종류(이 경우에는 variable)를 생성해주고 ($$ = newDeclNode(varK); 부분), 그 노드의 child를 가지 게 해준다. ($$->child[0] = $1; $$child[1] = $2;부분)**

**type\_sp의 경우에 다음 rule과 같은데**



**다음과 같이 간단하게 구현할 수 있다. INT일 경우 integer형의 typenode를 가지게 해주면 되고 VOID일 경우 void형의 typenode를 가지게 해주면 된다.**

**idsave의 경우에는 다음과 같이 구현되어 있는데,** 

**node가 expression종류이고 id이니 IdK로 node를 생성해주었다. 그리고 tiny.l에서 저장한 savedName으로 id의 이름을 설정해주었다.**

**그러면 parse tree에 다음과 같이 출력 된다.**

**Variable Declaration**

**Type : type\_sp**

**Id : idsave**

**또한 var\_decl : type\_sp idsave LSB numsave RSB SEMI 에서는 $$는 var\_decl가 되고 $1은 type\_sp $2는 idsave $4는 numsave가 된다. 위 식은 배열을 선언하는 것이므로 var\_decl의 노드종류를 배열로 해주고 자식들을 각각 type\_sp, idsave, numsave로 해준다.**

**그러면 parse tree에 다음과 같이 출력된다.**

**Array Declaration**

**Type : type\_sp**

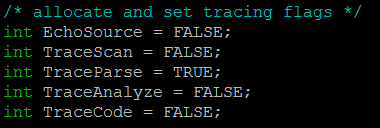
**Id : idsave**

**Const : numsave**

**나머지 grammar rule에 대해서도 다음과 같이 코드를 구성하였다.**

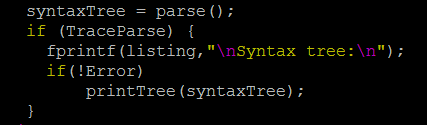
3. 제작 내용: 개발 결과.

-main.c





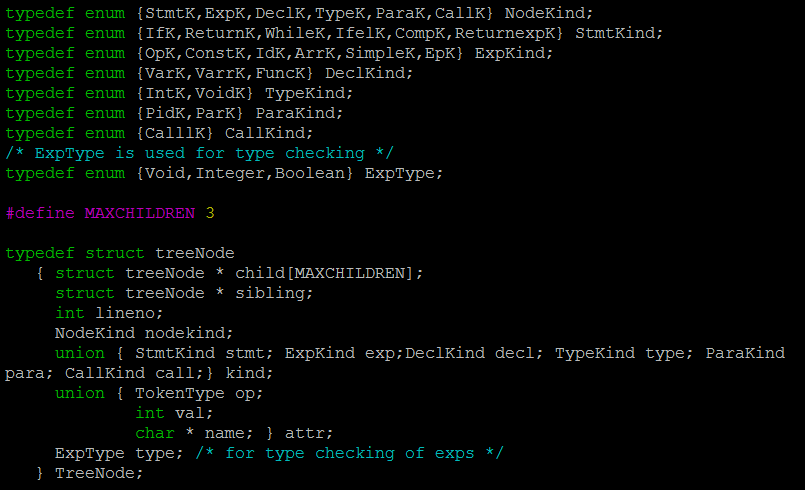
#define 과 flag들을 parse를 사용할 수 있게 수정해주었다.



parse()함수를 불러 parsetree를 그릴 수 있게 해주고, 에러가 없을 때만 tree를 print할 수

있게 해 주었다.

-globals.h



NodeKind : 노드의 종류를 나타낸다.

StmtK : statement 종류

1. IfK = if statement
2. ReturnK = return statement
3. WhileK = While statement
4. IfelK = if else statement
5. CompK = Compound statement
6. ReturnexpK = expression을 return하는 statement

ExpK : expression 종류

1. OpK = operator
2. ConstK = constant ( 숫자)
3. IdK = name (이름)
4. ArrK = Array(배열)
5. SimpleK = 간단한 expression
6. EpK = expression

DeclK : Declaration 종류

1. VarK = variable declaration
2. VarrK = array declaration
3. FuncK = function declaration

TypeK : Type 종류

1. IntK : integer type
2. VoidK : void type

ParaK : parameter 종류

1. PidK : parameter variable
2. ParK : parameter array variable

Callk : call 종류

1. calllk : function call

그리고 treenode structure에 node들의 종류를 구분할 수 있도록 kind라는 union을 만들어 위의 6가지 종류의 kind들을 나타낼 수 있게하였다.



그리고 assign과 array에 대한 flag를 만들어 주어 parsetree를 출력할 때 형식을 더 잘 맞출수 있게 해주었다.



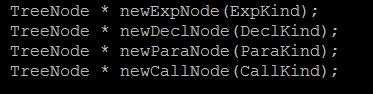
tiny.l에서 number와 identifier를 저장할 수 있게 변수를 선언하였다.

-tiny.l



위에서 선언해준 savedName을 이용해서 number와 identifer를 저장하고, 이 값을 cm.y에서 사용할 수 있게 하여 grammar rule을 완성하였다.

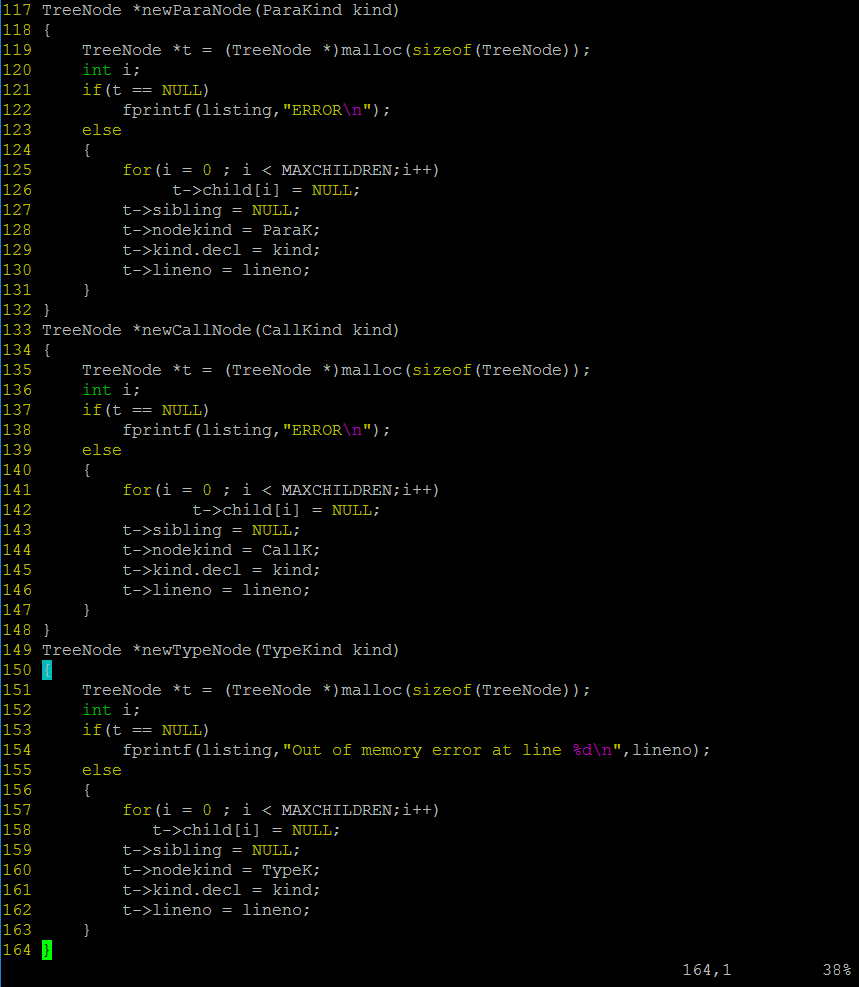
-util.h



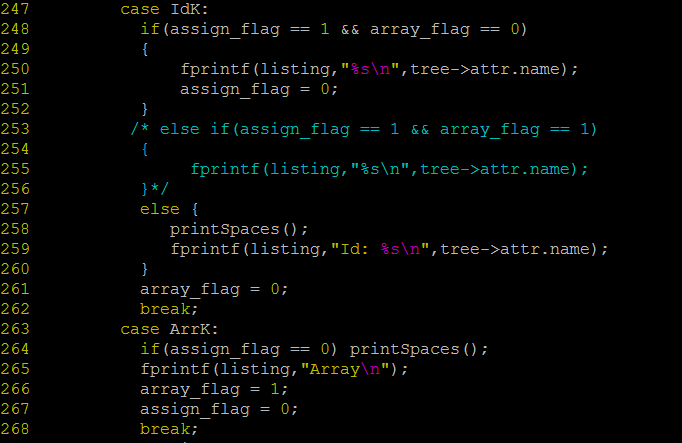
util.c에서 여러 종류들의 노드들을 정의하고 선언해주기 위해 함수들을 정의해놓았다.

-util.c

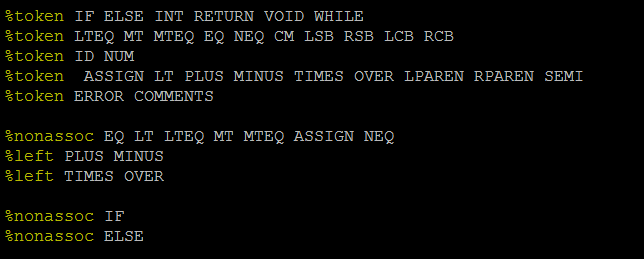
여러 종류들의 노드를 정의해주고 선언해주기 위해 추가한 함수들



Printtree()함수 내에서 assign\_flag와 array\_flag를 사용해서 parse tree형식에 잘 맞게 print해준 코드 부분

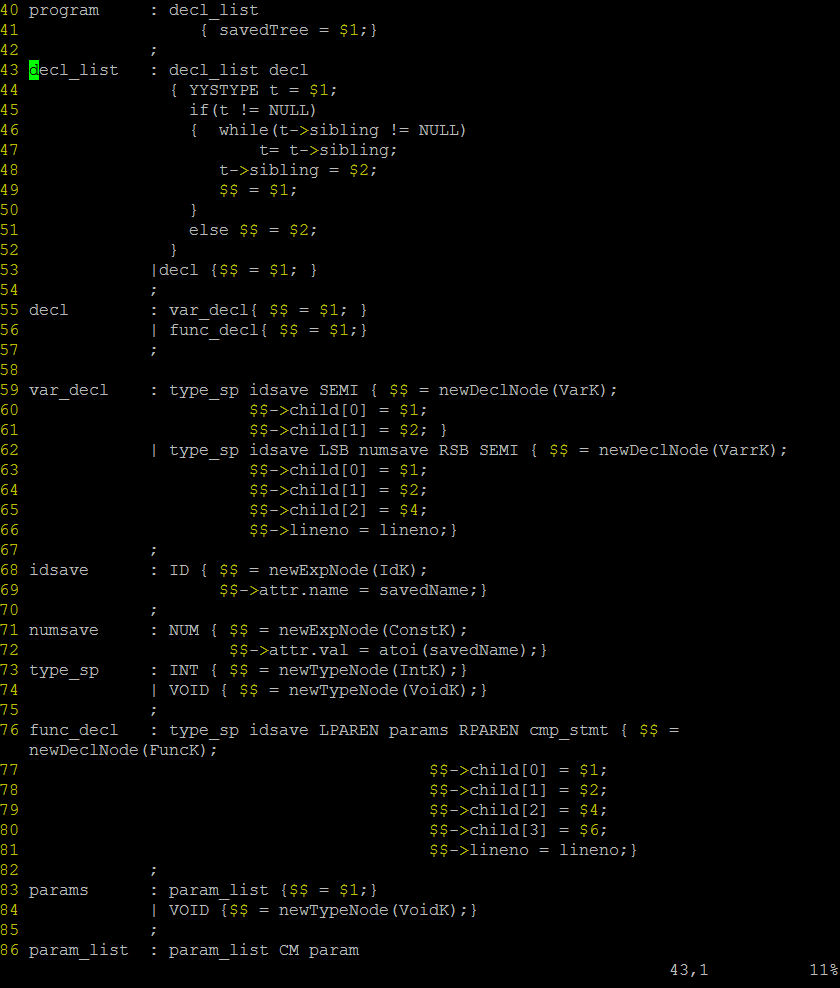


-cm.y



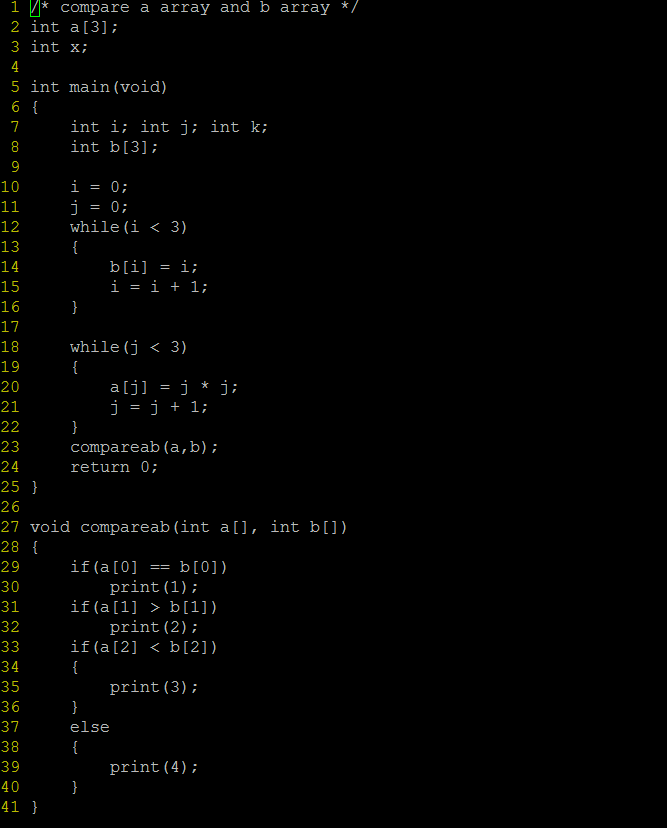
token들의 precedence와 conflict를 해결해주기 위해 우선순위 설정해준 부분

29가지의 Grammar Rule들을 정의해준 코드 부분



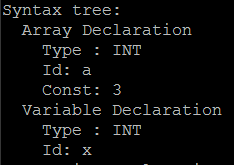
4. 시험 내용: 자신이 개발한 소프트웨어를 자체 평가하기 위한 기준 및 수행한 평가 방법, 그리고 그 것에 기반을 둔 평가 결과를 상세히 기술하라. (반드시 평가 점수가 높은 것이 좋은 것이 아니라 자신이 설정한 방법이 문제점을 찾는데 도움이 되었다는 것을 보이는 것이 더 중요함.

-errorfree.tny

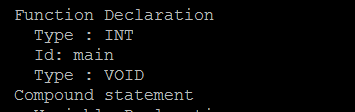


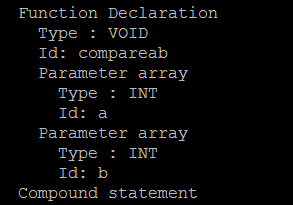
위 errorfree code는 parse tree가 잘 출력되는 지 확인하기 위해 작성 한 코드이다.

변수와 배열의 선언, 함수의 선언을 볼 수 있고 while문, if문 , if else문을 볼 수 있다. 이에 대해 출력된 parse tree로 분석해보면

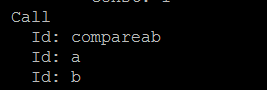


처음에 global variable로 배열과 일반 변수를 선언 해보았다. 각 변수의 type과 id가 잘 출력된 것을 볼 수 있고, 배열의 경우 몇 번째 index인지를 나타내는 const도 제대로 나왔다.

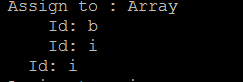
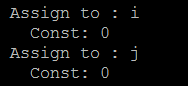




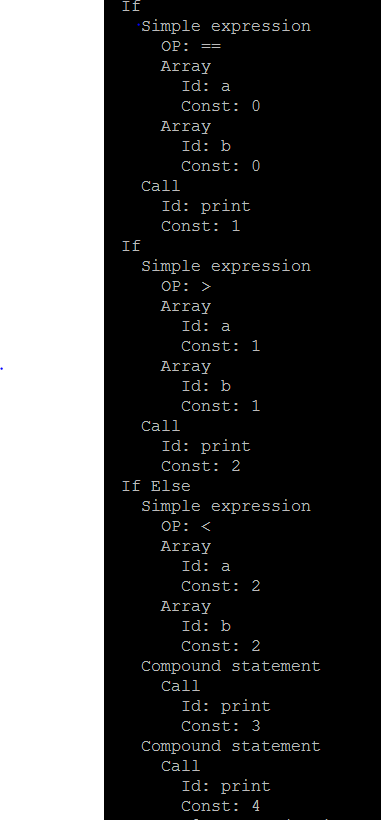
main함수와 user-define 함수인 compareab가 선언 된 부분의 parse tree이다. 두 함수 모두 Type에 맞게 나왔으며, id도 같고, compareab의 경우에는 array parameter에 대해서도 올바르게 출력되었다.



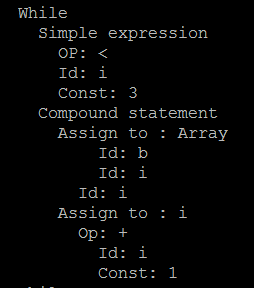
함수를 call한 부분인데, 함수의 이름인 compareab가 제대로 나왔고 그 뒤에 parameter인 a,b 모두 정상적으로 추력되었다.



변수와 array 변수에 값을 assign해준 부분이다.

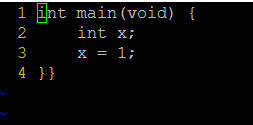


if문과 if-else문을 사용한 부분이다. if문 안에서는 == , < , > 과 같은 비교 연산자를 이용해 두 배열의 element를 비교하였고 if문에 해당된다면 print라는 함수를 call하였다. 마지막 부분의 if-else문도 비교연산자를 이용해 if문에 해당된다면 print(3)을 call하였고 else에 해당된다면 print(4)를 call해주었다.

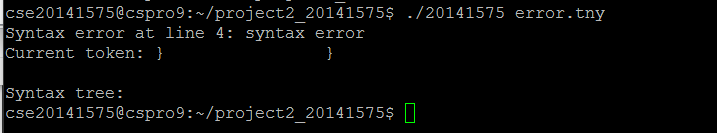


while문으로 조건에 비교연산자를 사용하였고 array의 index에 하나씩 하나씩 값을 assign해주었다.

-error.tny



마지막 4번째 줄에 }}와 같이 잘못된 syntax를 가진 testcase를 만들어보았고 그 결과는



4번 째 라인에서 error 라는 출력이 뜨고 현재 token이 뜨고 syntax tree를 출력되지 않게 하였다.

※ 본 항목에서의 시험은 테스트케이스를 의미하는 것으로, 자신이 추측한 ‘오류’가 발생할 수 있는 테스트 케이스를 작성하고, 이를 수행한 내용을 작성한다. ex) A라는 문제가 발생할 것으로 예상되어, test case a를 작성하였고, 예상한 데로 A의 문제가 발생하였다. 의 형식이 된다.

5. 평가 내용: 상기 내용을 바탕으로 자신이 개발한 소프트웨어의 장단점을 정확히 분석하고 그 내용을 기술할 것. 특히 개발 도중 예상치 않았던 문제가 발생하였다면 상세히 기술하고, 추후 어떠한 방법으로 해결할 수 있을지에 대하여 해결책을 기술할 것. (이 부분에서의 창의성이 중요하며, 반드시 자체 평가 내용이 좋을 필요는 없고, 스스로 문제점을 찾아 향후 해결책을 제시하는 것이 중요함.) 또한 공학 외적인 부분을 포함하여 경제 요건, 안정성, 신뢰성, 윤리성, 사회적 영향, 실행 가능성 등에 측면에 대해서도 연구 결과와 관련지어 논하라. 본 설계 프로젝트의 경우 안정성과 신뢰성에 대하여 반드시 기술할 것.

이 프로그램은 교재의 appendix에서 제공한 29개의 grammar rule를 정확히 잘 따르고 conflict가 발생하지 않도록 precedence를 잘 정해주었다. 또한 위에서 많은 결과를 고려한 testcase를 이용하여 while, if , else if, assign, array assign, function call, function define, compound expression, simple expression, return 등 많은 경우에 대해 올바른 결과가 나온 것을 확인 했으며, error 가 났을 경우에도 어디에서 에러가 났는지 확인하고, 그 때의 token을 출력 및 syntax tree가 나오지 않게 잘 하였다. 단점은 num1과 같은 variable을 선언했을 때 num까지 밖에 id가 인식되지 않아 그런 부분은 잘 되지 않았고, 이런 부분은 stack을 이용해 구현하였으면 충분히 구현할 수 있었다고 생각한다. 또한 몇몇 복잡한 grammar가 들어가면 제대로 안될 때도 있는데 그럴 때는 flag를 더 설정하여 print해주면 해결될 것이라 생각한다.그리고 testcase를 위에 2개 만이 아닌 약 10가지의 다른 여러가지 의 testcase를 만들어서 실험하고 확인하여 이 결과물의 신뢰성이 충분히 보장되고 그런 testcase 하나에 대해서 여러 번 실행해본 결과 항상 같은 결과물이 확인 되는 것으로 보아 안정성도 충분히 보장된다.

V. 기타

1. 자체 평가: 본 프로젝트의 수행에 있어 왜 자신이 다른 사람보다 더 좋은 점수를 받아야 하는지를 기술하라. 위에서의 (5. 평가 내용)은 개발한 소프트웨어에 대한 자체 기준을 통한 평가이고, 여기서의 자체 평가는 프로젝트 수행 전반에 걸쳐 자신이 얼마나 창의적으로 목표 및 방법을 설정하고, 수행 중 발생한 문제를 얼마나 현명하게 해결 했는지, 얼마나 독창적인 아이디어가 들어가 있는지, 자신의 결과가 얼마나 우수한 지에 대한 평가임.

위에서 평가한 신뢰성과 안정성처럼 본 프로그램은 주어진 input에 대한 결과물의 신뢰도가 높고, 또한 여러 번 실행해도 다른 요인에 의해 결과물이 바뀌지 않음으로 좋은 프로그램이라고 볼 수 있다. 이번 프로젝트를 하면서 tokenizing과 parsing만으로 syntax tree와 syntax error를 잡아내었고 교재에 명시된 29가지의 grammar rule에 대해 cm.y에 잘 명시되었다. 또한 많은 정보를 알 수 있는 input파일을 넣어 제대로 된 정보만을 추력 하였으므로 우수한 결과라고 볼 수 있다.

2. 기타 본 설계 프로젝트를 수행하면서 느낀 점을 요약하여 기술하라. 그 내용은 어떤 것이든 상관이 없으며, 본 프로젝트에 대한 문제점 제시 및 제안을 포함하여 자유롭게 기술할 것.

이론으로만 접한 syntax tree는 매우 어려웠는데 주어진 코드와 tool(bison,flex)들을 이용하고 교재에 있는 grammar rule을 참고하니 생각보다 이해가 잘가고 parse tree를 이해하는데 큰 도움이 된 것 같다. 처음에 grammar rule을 작성하는데 조금 해매고, tree를 print할 때 사소한 문제와 conflict에 대한 문제 때문에 고생을 했지만 이런 것들을 해결해가면서 많은 것을 얻었다고 생각한다.