컴파일러 기말과제

소프트웨어학부, 20163091 김민주

□ 목표

- kotlin언어를 인식할 수 있는 mini-kotlin 어휘분석기를 작성
- kotlin예제를 분석하여 예상한 결과와 일치하는지 확인

□ 코틀린 예제

예제	내용
class.mc	class선언 후 class 변수와 함수를 선언하는 예제 class, var, val, fun() 등
while.mc	while문과 true, false를 추가한 예제
multi_class.mc	다중 class로 한 파일 안에서 여러개의 class를 사용하였다. 기존의 class.mc에서 알아볼 수 있는 정보에서 if문을 추가하였다.

- class.mc

```
data class minjoo1412{
    val name = minjoo
    val type = System
    var num = 24

    fun get(){
        minjoo = name
    }

    fun set(){
        name = minjoo
        type = System
        num = 24
    }
}
```

- while.mc

```
data class minjoo{
    var minjoo = false
    fun main(){
        var minjoo = true
        while(minjoo == true){
            var minjoo = false
        }
    }
}
```

- multi_class.mc

```
data class hi{
    var hello = 36
    fun jelly(){
        if(true){
            hello = 21
        }
    }
}

data class nice{
    var hello = 12
    fun mogi(){
        if(false){
            hello = 21
        }else{
            hello = 64
        }
    }
}
```

□ 코틀린 어휘분석기 작성

BNF를 기준으로 lex에서 syntax를 선언했으며, 이번 예제를 분석하기 위해
class, fun 등 을 추가하여 class문, while문, if문, 함수를 인식하도록 설계하였습니다.
또한 val, var, true, false 등 kotlin예제에서 사용되는 문법을 추가하였습니다.

- mini_kotlin.l

```
%{

#include "y.tab.h"
%}

letter  [a-zA-Z_]
digit [0-9]

%%
"=="  return(TYES);
"=" return(TASSIGN);
"+" return(TPLUS);
"*" return(TMUL);
"(" return(TLPAREN);
")" return(TRPAREN);
"[" return(TLBRACKET);
"]" return(TRBRACKET);
"{" return(TBEGIN);
"}" return(TEND);
":" return(TCOLON);

class return(CLASSSYM);
fun return(FUNSYM);
data  return(DATASYM);
var return(VARSYM);
val return(VALSYM);
while return(WHILESYM);
if  return(IFSYM);
true  return(TRUESYM);
false return(FALSESYM);


{digit}+  return(TNUMBER);
{letter}({letter}|{digit})* return(TIDENT);
\"  return(DDA);
[ \t\r\n] ;
. return(TERROR);
%%
int yywrap()
{
  return 1;
}
```

- class앞에서 사용할 data와 kotlin에서 변수선언 시 필요한 var, val 등을 추가하였다.

- mini_kotlin.y

```
%{
#include <stdio.h>
extern int yylex();
void yyerror();
%}

%token  TIDENT TNUMBER TASSIGN TDOT TBEGIN TEND TERROR
%token  TLPAREN TRPAREN TLBRACKET TRBRACKET TPLUS TMINUS TMUL TDIV TMOD
%token  CLASSSYM DDA FUNSYM IFSYM TCOLON TRUESYM TYES VALSYM VARSYM WHILESYM
%token  DATASYM FALSESYM
%%
class_stmts:    class_stmt  { puts("0-1"); }
        | class_stmts class_stmt    { puts("0-2"); }
        ;
class_stmt: DATASYM CLASSSYM TIDENT class_compound_stmt {puts("1"); }
        ;
class_compound_stmt:    TBEGIN class_statement2 TEND    { puts("2"); }
        ;
class_statement2: class_statement   { puts("3-1"); }
        | class_statement2 class_statement  { puts("3-2"); }
        ;
class_statement: assign_stmt    { puts("4-1"); }
        | fun_stmt {puts("4-2");}
        ;
fun_stmt: FUNSYM TIDENT TLPAREN TRPAREN fun_compound_stmt   {puts("11");}
        ;
fun_compound_stmt:  TBEGIN fun_statement2 TEND  { puts("12"); }
        ;
fun_statement2: fun_statement   { puts("13-1"); }
        | fun_statement2 fun_statement  { puts("13-2"); }
        ;
fun_statement:  statement   { puts("14-1"); }
        |fun_compound_stmt  { puts("14-2"); }
        ;
statement:  assign_stmt { puts("15-1"); }
        |while_stmt { puts("15-2"); }
        |if_stmt    { puts("15-3"); }
        ;

assign_stmt: lhs TASSIGN exp    { puts("5"); }
        ;
while_stmt: WHILESYM TLPAREN variable TYES exp TRPAREN fun_statement    {puts("16");}
        ;
if_stmt: IFSYM TLPAREN exp TRPAREN fun_statement    { puts("17-1"); }
        |  ELSESYM fun_statement    { puts("17-2"); }
        ;
lhs:    variable    { puts("6-1"); }
        |VARSYM variable    { puts("6-2"); }
        |VALSYM variable    { puts("6-3"); }
```

```
        ;
exp:    exp TPLUS term  { puts("7-1"); }
        | exp TMINUS term   { puts("7-2"); }
        | term  { puts("7-3"); }
        ;
term:   term TMUL factor    { puts("8-1"); }
        | term TDIV factor  { puts("8-2"); }
        | term TMOD factor  { puts("8-3"); }
        | factor    { puts("8-4"); }
        ;
factor: TMINUS factor   { puts("9-1"); }
        | variable  { puts("9-2"); }
        | TNUMBER   { puts("9-3"); }
        | TLPAREN exp TRPAREN   { puts("9-4"); }
        | TRUESYM   {puts("9-5");}
        | FALSESYM  {puts("9-6");}
        ;
variable:   TIDENT  { puts("10-1"); }
        | TIDENT TLBRACKET exp TRBRACKET    { puts("10-2"); }
        | TIDENT TCOLON { puts("10-3"); }
        ;
%%

int main()
{
    yyparse();
    return 0;
}

void yyerror(char *s)
{
    fprintf(stderr, "%s\n", s);
    return;
}
```

- class_stmts에서 1개 이상의 class를 인식하도록 하였고, class안에서 fun(함수) 및 while문, if문, 변수 선언등을 사용할 수 있게 만들었습니다.
- statement에서 if문과 while문을 인식하도록 설계하였습니다.
- 변수 선언시 필요한 val과 var을 lhs에 추가하였고, while문과 if문에 사용될 true, false를 factor에 추가함으로써 어휘분석기가 이를 인식하도록 설계하였다.

□ 과제3 변경점

1. 출력을 숫자만이 아닌 설명을 더해 가시성 확보

2. else문 statement 추가 (기존의 else문은 if문 없이도 인식되었는데 이를 수정) (fun_compound_stmt를 통해 begin과 end인식하는 것을 활용하여 기존의 syntax error 제거)

□ 변경된 mini_kotlin.y

```
%{
#include <stdio.h>
extern int yylex();
void yyerror();
%}

%token  TIDENT TNUMBER TASSIGN TDOT TBEGIN TEND TERROR
%token  TLPAREN TRPAREN TLBRACKET TRBRACKET TPLUS TMINUS TMUL TDIV TMOD
%token  CLASSSYM DDA FUNSYM IFSYM TCOLON TRUESYM TYES VALSYM VARSYM WHILESYM ELSESYM
%token  DATASYM FALSESYM
%%
class_stmts:    class_stmt  { puts("0-1, 클래스 인식"); }
        | class_stmts class_stmt    { puts("0-2, 클래스 인식(여러개)"); }
        ;
class_stmt: DATASYM CLASSSYM TIDENT class_compound_stmt {puts("1, 클래스 분석"); }
        ;
class_compound_stmt:    TBEGIN class_statement2 TEND    { puts("2, class{ }"); }
        ;
class_statement2: class_statement   { puts("3-1"); }
        | class_statement2 class_statement  { puts("3-2"); }
        ;
class_statement: assign_stmt    { puts("4-1"); }
        | fun_stmt {puts("4-2, 클래스 내 함수 발견");}
        ;
fun_stmt: FUNSYM TIDENT TLPAREN TRPAREN fun_compound_stmt   {puts("11, 함수 분석");}
        ;
fun_compound_stmt:  TBEGIN fun_statement2 TEND  { puts("12, fun{ }"); }
        ;
fun_statement2: fun_statement   { puts("13-1, 함수 인식"); }
        | fun_statement2 fun_statement  { puts("13-2, 함수 인식(여러개)"); }
        ;
fun_statement:  statement   { puts("14-1"); }
        |fun_compound_stmt  { puts("14-2"); }
        ;
statement:  assign_stmt { puts("15-1, assign문 인식"); }
        |while_stmt { puts("15-2, while문 인식"); }
        |if_stmt    { puts("15-3, if문 인식"); }
        ;

assign_stmt: lhs TASSIGN exp    { puts("5, assign문"); }
        ;
while_stmt: WHILESYM TLPAREN variable TYES exp TRPAREN fun_statement    {puts("16, while문 분석");}
        ;
if_stmt: IFSYM TLPAREN exp TRPAREN fun_compound_stmt ELSESYM else_stmt  { puts("17-1,if문, else문 분석"); }
        |IFSYM TLPAREN exp TRPAREN fun_statement    { puts("17-2, if문 분석"); }
        ;
else_stmt: fun_compound_stmt    { puts("17-3, else문 확인"); }
        ;
lhs:    variable    { puts("6-1"); }
```

```
        |VARSYM variable    { puts("6-2, var 변수"); }
        |VALSYM variable    { puts("6-3, val 변수"); }
        ;
exp:    exp TPLUS term  { puts("7-1"); }
        | exp TMINUS term   { puts("7-2"); }
        | term  { puts("7-3"); }
        ;
term:   term TMUL factor    { puts("8-1"); }
        | term TDIV factor  { puts("8-2"); }
        | term TMOD factor  { puts("8-3"); }
        | factor    { puts("8-4"); }
        ;
factor: TMINUS factor   { puts("9-1"); }
        | variable  { puts("9-2, 변수"); }
        | TNUMBER   { puts("9-3, 숫자"); }
        | TLPAREN exp TRPAREN   { puts("9-4"); }
        | TRUESYM   {puts("9-5, TRUE");}
        | FALSESYM  {puts("9-6, FALSE");}
        ;
variable:   TIDENT  { puts("10-1"); }
        | TIDENT TLBRACKET exp TRBRACKET    { puts("10-2"); }
        | TIDENT TCOLON { puts("10-3"); }
        ;
%%

int main()
{
    yyparse();
    return 0;
}

void yyerror(char *s)
{
    fprintf(stderr, "%s\n", s);
    return;
}
```

□ 코틀린 어휘분석기 실행

- class.mc

```
File  Edit  View  Search  Terminal  Help
kmucs@kmucs-ThinkPad-T440:~/Downloads/compiler/compiler_final$ ./a.out < class.mc
10-1
6-3, val 변수
10-1
9-2, 변수
8-4
7-3
5, assign문
4-1
3-1
10-1
6-3, val 변수
10-1
9-2, 변수
8-4
7-3
5, assign문
4-1
3-2
10-1
6-2, var 변수
9-3, 숫자
8-4
7-3
5, assign문
4-1
3-2
10-1
6-1
10-1
9-2, 변수
8-4
7-3
5, assign문
15-1, assign문 인식
14-1
13-1, 함수 인식
12, fun{ }
11, 함수 분석
4-2, 클래스 내 함수 발견
3-2
10-1
6-1
10-1
9-2, 변수
```

```
8-4
7-3
5, assign문
15-1, assign문 인식
14-1
13-1, 함수 인식
10-1
6-1
10-1
9-2, 변수
8-4
7-3
5, assign문
15-1, assign문 인식
14-1
13-2, 함수 인식(여러개)
10-1
6-1
9-3, 숫자
8-4
7-3
5, assign문
15-1, assign문 인식
14-1
13-2, 함수 인식(여러개)
12, fun{ }
11, 함수 분석
4-2, 클래스 내 함수 발견
3-2
2, class{ }
1, 클래스 분석
0-1, 클래스 인식
```

- 어휘분석기가 class하나를 인식하여(0-1) 변수와 함수를 선언한 것을 확인할 수 있다.

- while.mc

```
kmucs@kmucs-ThinkPad-T440:~/Downloads/compiler/compiler_final$ ./a.out < while.mc
10-1
6-2, var 변수
9-6, FALSE
8-4
7-3
5, assign문
4-1
3-1
10-1
6-2, var 변수
9-5, TRUE
8-4
7-3
5, assign문
15-1, assign문 인식
14-1
13-1, 함수 인식
10-1
9-5, TRUE
8-4
7-3
10-1
6-2, var 변수
9-6, FALSE
8-4
7-3
5, assign문
15-1, assign문 인식
14-1
13-1, 함수 인식
12, fun{ }
14-2
16, while문 분석
15-2, while문 인식
14-1
13-2, 함수 인식(여러개)
12, fun{ }
11, 함수 분석
4-2, 클래스 내 함수 발견
3-2
2, class{ }
1, 클래스 분석
0-1, 클래스 인식
```

- while문(16)이 정상적으로 인식되고, true, false 또한 class내부, while문 조건 내부에서 정상적으로 인식된 것을 확인할 수 있다.

- multi_class.mc

```
kmucs@kmucs-ThinkPad-T440:~/Downloads/compiler/compiler_final$ ./a.out < multi_clas
s.mc
10-1
6-2, var 변수
9-3, 숫자
8-4
7-3
5, assign문
4-1
3-1
9-5, TRUE
8-4
7-3
10-1
6-1
9-3, 숫자
8-4
7-3
5, assign문
15-1, assign문 인식
14-1
13-1, 함수 인식
12, fun{ }
14-2
17-2, if문 분석
15-3, if문 인식
14-1
13-1, 함수 인식
12, fun{ }
11, 함수 분석
4-2, 클래스 내 함수 발견
3-2
2, class{ }
1, 클래스 분석
0-1, 클래스 인식
10-1
6-2, var 변수
9-3, 숫자
8-4
7-3
5, assign문
4-1
3-1
9-6, FALSE
8-4
```

```
7-3
10-1
6-1
9-3, 숫자
8-4
7-3
5, assign문
15-1, assign문 인식
14-1
13-1, 함수 인식
12, fun{ }
10-1
6-1
9-3, 숫자
8-4
7-3
5, assign문
15-1, assign문 인식
14-1
13-1, 함수 인식
12, fun{ }
17-3, else문 확인
17-1,if문, else문 분석
15-3, if문 인식
14-1
13-1, 함수 인식
12, fun{ }
11, 함수 분석
4-2, 클래스 내 함수 발견
3-2
2, class{ }
1, 클래스 분석
0-2, 클래스 인식(여러개)
```

- 2개이상의 class가 확인되어(0-2) 2개의 class를 선언해주고, if문(17-1)과 else문(17-3)도 정상적으로 인식된 것을 확인할 수 있다.

□ 문제점 및 해결방안

- if문과 while문 구현 시 조건에서 TRUESYM(true)을 인식하도록 구현하였지만 true를 인식하지 못해 conflict가 발생

  while과 if문의 조건에서 true를 TRUESYM로 인식하는 것이 아닌 exp로 인식하게 한 후 exp -> term -> factor -> TRUESYM으로 인식하도록 설계

- Class를 여러개 선언하였을 때 1개의 클래스만 인식하고 종료되는 것을 확인

  class_stmts(0)를 선언함으로써 1개 이상의 클래스를 인식할 수 있도록 설계

- if문에서 ELSE를 인식 못함

  기존의 ELSE문은 if문과 연결되어 선언하였었음,
  if문과 분리하여 따로 분리하여 ELSE만 읽을 수 있도록 선언

```
if_stmt: IFSYM TLPAREN exp TRPAREN fun_statement { puts("17-1"); }
        | IFSYM TLPAREN exp TRPAREN fun_statement ELSESYM fun_statement
                { puts("17-2"); }
```

```
if_stmt: IFSYM TLPAREN exp TRPAREN fun_statement   { puts("17-1"); }
        |  ELSESYM fun_statement       { puts("17-2"); }
        ;
```

- if문와 else문을 분리시키지 못함
  기존의 else문은 if문이 없어도 인식가능하였지만 이를 수정하여
  if문에서 ELSESYM을 인식하면 else문을 읽을 수 있도록 수정
  괄호인 TBEGIN과 TEND를 사용하는 fun_compound_stmt를 사용하여 if문 없이
  else를 실행할 수 없도록 함

```
if_stmt: IFSYM TLPAREN exp TRPAREN fun_compound_stmt ELSESYM else_stmt   {
        puts("17-1,if문, else문 분석"); }
        |IFSYM TLPAREN exp TRPAREN fun_statement { puts("17-2, if문 분석"); }
        ;
else_stmt: fun_compound_stmt     { puts("17-3, else문 확인"); }
        ;
```

yacc에서 보시면, 0번에서 1개 이상의 클래스를 인식한 후
1번에서 클래스를 인식하면 tbegin과 tend라는 중괄호로 범위를 줍니다.
3번에서는 구문들을 각각 인식할 수 있도록 하며, 4번에서 함수 및 여러 구문들을 인식할 수 있도록 합니다.
함수는 11번으로 클래스와 비슷하게 설계되어있으며 statement로 이어져 while문과 if문을 사용할 수 있습니다.