HACETTEPE UNIVERSITY

Instructors	**Dr.Pınar Duygulu Şahin, Dr.Nazlı İkizler Cinbiş**
Research Assistant	**Feyza Nur Kılıçaslan**
Due Date	**17.12.2019**
BBM301: Programming Languages Project	
Topic of Assignment	**GIS Implementation with Lex and Yacc**

Student Name	Number
Anıl ERYILMAZ	21526973
Metin DEMİR	21526902
Mehmet Emin TUNCER	21591022

DESCRIPTION OF ASSIGNMENT

In this assignment, we tried to build a language with lex and yacc. Before starting to describe the assignment, first look up what is lex and what is yacc.

What is lex and yacc is :

Lex is a computer program that generates lexical analyzers ("scanners" or "lexers").

Lex is commonly used with the yacc parser generator. Lex, originally written by Mike Lesk and Eric Schmidt and described in 1975 is the standard lexical analyzer generator on many Unix systems, and an equivalent tool is specified as part of the POSIX standard.

Lex reads an input stream specifying the lexical analyzer and outputs source code implementing the lexer in the C programming language. In addition to C, some old versions of Lex could also generate a lexer in Ratfor.

Yacc (Yet Another Compiler-Compiler) is a computer program for the Unix operating system developed by Stephen C. Johnson.

It is a Look Ahead Left-to-Right (LALR) parser generator, generating a parser, the part of a compiler that tries to make syntactic sense of the source code, specifically a LALR parser, based on an

analytic grammar written in a notation similar to Backus–Naur Form (BNF).

Yacc is supplied as a standard utility on BSD and AT&T Unix. GNU-based Linux distributions include Bison, a forward-compatible Yacc replacement.

In this project, we are supposed to develop a programming language to use Geographical Maps and Satellite data (GPS) easily. For this purpose, we defined a lex analyzer and yacc.

Here are the lexemes of our lex analyzer :

digit [0-9]

integer [-+]?[0-9]+

float [0-9]+"."[0-9]+

string "\""{alphanumeric}*"\""

alphabetic [A-Za-z]

upperCase [A-Z]

alphanumeric ({alphabetic}|{digit})

variable {alphabetic}{alphanumeric}*

operations "+"|"-"|"*"|"/"|"%"

number {float}|{integer}|{digit}

"while" : Declaration of while loop

"continue" : It continues the process

"break" : Breaks the process

"print" : Prints variable(s)

"if" : Declaration of If statement

"else" : Declaration of Else statement

"elif" : Declaration of Else If statement

"switch" : Declaration of Switch statement

"case" : Declaration of Case statement

"default" : Declaration of default which is in switch/case

"ArrayList" : Declaration of ArrayList

"User" : Declaration of User

"CrossRoad" : Declaration of CrossRoad

"Road" : Declaration of Road

"GpsData" : Declaration of GPSData

"ShowOnMap" : It calls the ShowOnMap function

"SearchLocation" : It calls the SearchLocation function

"GetRoadSpeed" : It calls the GetRoadFunction function

"GetLocation" : It calls the GetLocation function

"ShowTarget" : It calls the ShowTarget function

"Object3D" : Declaration of 3D Object

"Graph" : Declaration of Graph structure

"TRUE" : Declaration of True expression

"FALSE" : Declaration of False expression

"return" : Declaration of return statement

"=" : Declaration of Equals assign symbol

"+" : Declaration of Plus operation

"-" : Declaration of Minus operation

"*" : Declaration of Multiply operation

"/" : Declaration of Divide operation

"%" : Declaration of Mod(%)

"+=" : Declaration of Plus+Equal

"-=" : Declaration of Minus+Equal

"*=" : Declaration of Multipyle+Equal

"/=" : Declaration of Divide+Equal

"%=" : Declaration of Mod+Equal

"." : Declaration of Dot

";" : Declaration of Semicolon

"," : Declaration of Comma

":" : Declaration of Colon

"(" : Declaration of Left Paranthesis

")" : Declaration of Right Paranthesis

"[" : Declaration of Left Bracket

"]" : Declaration of Right Bracket

"{" : Declaration of Left Curly Bracket

"}" : Declaration of Right Curly Bracket

".||." : Declaration of Comment Line

"&&" : Declaration of AND

"||" : Declaration of OR

"!" : Declaration of Not Equals

">=" : Declaration of Greater than or Equals

"<=" : Declaration of Less than or Equals

">" : Declaration of Greater sign

"<" : Declaration of Less Than sign

And here, this is our BNF grammar :

```
program
        : user_functions
        | program user_functions
        ;
user_functions
                : VARIABLE L_PARANTS params R_PARANTS function_block
                ;
function_block
        : LC_BRACKET statements RC_BRACKET
        ;
params
        :
        | VARIABLE
        | params COMMA VARIABLE
        ;
statements
        : stmt SEMI_COLON
        | statements stmt SEMI_COLON
```

```
        ;

stmt
        : assignment_stmt
        | control_stmt
        | loop_stmt
        | function_stmt
        | return_stmt
        | print_stmt
        | array_declaration
        | variable_declaration
        | func_declarations
        | object_declarations
        ;


func_declarations
        : showonMap
        | searchLocation
        | getRoadSpeed
        | getLocation
        | showTarget
```

```
        ;
variable_declaration
        : number_declaration
        | string_declaration
        ;
object_declarations
        : graph_declaration
        | crossroad_declaration
        | road_declaration
        | GpsData_declaration
        | Object3D_declaration
        | user_Declaration ;


string_declaration
        : STR VARIABLE EQUAL_SIGN STRING
        | STR VARIABLE EQUAL_SIGN VARIABLE
        | STR VARIABLE
        ;
number_declaration
        : datatype VARIABLE
```

```
        | datatype VARIABLE EQUAL_SIGN NUMBER
        | datatype VARIABLE EQUAL_SIGN VARIABLE
        ;
datatype
        : FLOAT|INT
        ;
user_Declaration
        : USER VARIABLE EQUAL_SIGN USER L_PARANTS VARIABLE R_PARANTS
        | USER VARIABLE EQUAL_SIGN USER L_PARANTS STRING R_PARANTS;
searchLocation
        : SEARCHLOCATION L_PARANTS VARIABLE R_PARANTS
        | SEARCHLOCATION L_PARANTS STRING R_PARANTS
        ;
getLocation
        : GETLOCATION L_PARANTS VARIABLE R_PARANTS
        | GETLOCATION L_PARANTS STRING R_PARANTS
        ;
showTarget
        : SHOWTARGET L_PARANTS VARIABLE R_PARANTS
        | SHOWTARGET L_PARANTS STRING R_PARANTS
```

;

getRoadSpeed

: GETROADSPEED L_PARANTS VARIABLE R_PARANTS

| GETROADSPEED L_PARANTS STRING R_PARANTS

showonMap

: SHOWONMAP L_PARANTS NUMBER COMMA NUMBER R_PARANTS

| SHOWONMAP L_PARANTS VARIABLE COMMA VARIABLE R_PARANTS

;

Object3D_declaration

: OBJECT3D VARIABLE EQUAL_SIGN OBJECT3D L_PARANTS NUMBER COMMA NUMBER COMMA NUMBER R_PARANTS

;

GpsData_declaration

: GPSDATA VARIABLE EQUAL_SIGN GPSDATA L_PARANTS VARIABLE R_PARANTS

| GPSDATA VARIABLE EQUAL_SIGN GPSDATA L_PARANTS STRING R_PARANTS

;

road_declaration

: ROAD VARIABLE EQUAL_SIGN ROAD L_PARANTS R_PARANTS

;

graph_declaration

: GRAPH VARIABLE EQUAL_SIGN GRAPH L_PARANTS NUMBER COMMA NUMBER R_PARANTS

| GRAPH VARIABLE EQUAL_SIGN GRAPH L_PARANTS VARIABLE COMMA VARIABLE R_PARANTS

;

crossroad_declaration

: CROSSROAD VARIABLE EQUAL_SIGN CROSSROAD L_PARANTS VARIABLE R_PARANTS

| CROSSROAD VARIABLE EQUAL_SIGN CROSSROAD L_PARANTS STRING R_PARANTS

;

assignment_stmt

: VARIABLE EQUAL_SIGN right_assignment

| VARIABLE shortcut_operations right_assignment

|

;

print_stmt

: PRINT VARIABLE

operations

: ADDITION_SIGN

| SUBTRACTION_SIGN

| MULTIPLY

```
        | DIVIDE
        | MOD;
shortcut_operations
        : ADDITION_SIGN_EQUALS
        | SUBTRACTION_SIGN_EQUALS
        | MULTIPLY_EQUALS
        | DIVIDE_EQUALS
        | MOD_EQUALS
        ;
right_assignment
        : VARIABLE
        | VARIABLE operations variable_list
        | STRING
        | NUMBER
        | STRING operations variable_list
        | NUMBER operations variable_list
        ;
variable_list
        : VARIABLE
        | VARIABLE operations right_assignment
```

```
        | STRING
        | NUMBER
        | STRING operations right_assignment
        | NUMBER operations right_assignment
        ;
control_stmt
        : if_stmt
        | switch_stmt
        ;
if_stmt
        : IF L_PARANTS boolean_expression R_PARANTS function_block
        | if_stmt ELSE_IF L_PARANTS boolean_expression R_PARANTS function_block
        | if_stmt ELSE function_block
        ;
switch_stmt
        : SWITCH L_PARANTS VARIABLE R_PARANTS function_block
        | SWITCH L_PARANTS VARIABLE R_PARANTS LC_BRACKET switch_cases
RC_BRACKET
        ;
switch_cases
        : CASE boolean_expression COLON statements
```

```
        | CASE boolean_expression COLON statements BREAK
        | CASE boolean_expression COLON statements switch_cases
        | CASE boolean_expression COLON statements BREAK switch_cases
        | default
        ;
default
        : DEFAULT COLON statements BREAK
        | DEFAULT COLON statements
        ;
loop_stmt
        : while_loop
        | forLoop
        ;
loop_block
        : LC_BRACKET statements RC_BRACKET
        ;
while_loop
        : WHILE_LOOP L_PARANTS boolean_expression R_PARANTS loop_block
        ;
forLoop
```

: FOR_LOOP L_PARANTS VARIABLE SEMI_COLON boolean_expression SEMI_COLON VARIABLE shortcut_operations right_assignment R_PARANTS loop_block

;

function_stmt

: VARIABLE L_PARANTS params_list R_PARANTS

;

params_list

: empty

| right_assignment

| right_assignment COMMA params_list

;

return_stmt

: RETURN right_assignment

;

boolean_expression

: TRUE

| FALSE

| VARIABLE

| logical_expression

| NUMBER

;

```
logical_expression
        : boolean_expression boolean_check boolean_expression
        | boolean_expression relational_check boolean_expression
        ;
boolean_check
        : AND
        | OR
        ;
relational_check
        : LESS_THAN
        | LESS_OR_EQ
        | GREATER
        | GREAT_OR_EQ
        | IS_EQUAL
        ;
array_declaration
        : ARRAY VARIABLE EQUAL_SIGN ARRAY L_PARANTS R_PARANTS
        ;
empty
        : ;
```

And this is our test file :

```
main(){
        int x;
        int y=5;
        int z=y;
        int t;
        x=y;
        x+=3;
        z=x-z;
        y*=2;
        t=x+y;
        z=t/5;
        x=x+y*3-t/3+z*y;
        float i;
        float j= 3.1;
        float k=j;
        float m;
        i=j;
        i+=3.5;
        k=i+j;
```

```
j*=2.3;

m=i+j;

k=m/2.3;

i=i+j*3.7+m/5.235-k*j;

str string1;

str string2="This is a string";

str string3="I am at Holiday.

                                    How are you????                                    ";

string1= " ajkhsdk bnfhsuka fısSW+T^+RGHGsajdkwhudhas";

string2="string1 was a RANDOM String :)";

string1= string3;

string2= "4565645 daa24-59udaıscy8gzvxq213*94131 3d/asdavq      1";

string1= "I am changed, now string 2 is a RANDOM string :P";

User user1= User("Metin Demir");

str username2;

username2= "Anil Eryilmaz";

User user2= User(username2);

str username3="mc_Memin_06_ANKARA_yaraLi";

User user3 = User(username3);

str username4;
```

```
int userID;

userID=1;

if(userID==1){

        username4="I am 4th user";

};

GetLocation("Metin Demir");

GetLocation(username2);

GetLocation(username3);

GetLocation(username4);

userID=3;

if(userID==3){

        if(username2 == "Anil Eryilmaz"){

                print username2;

        };

};

Graph graph1= Graph(1,2);

int edge1= 5;

int vertex1;

vertex1=8;

Graph graph2 = Graph(edge1,vertex1);
```

```
CrossRoad crossraod1 = CrossRoad("Taksim Square");

str crossroadname= "KIZILAY";

CrossRoad crossraod2 = CrossRoad(crossroadname);

Object3D object1 = Object3D(1,2,3);

int xvalue=1;

int yvalue=2;

int zvalue=3;

Object3D object2 = Object3D(xvalue,yvalue,zvalue);

GpsData gpsdata = GpsData("Sincan Lisesi");

gpsLocation = "Hacettepe";

GpsData gpsdata1 = GpsData(gpsLocation);

ShowOnMap(12,4);

int xloc=15;

int yloc=75;

ShowOnMap(xloc,yloc);

str loc= "ADANA";

SearchLocation(loc);

SearchLocation("Ankara Sincan");

ShowTarget(loc);

ShowTarget("Hacettepe Universitesi");
```

```
str road = "Istiklal Caddesi";

GetRoadSpeed(road);

GetRoadSpeed("Ankara Bulvarı");

username1="alican";

GetLocation(username1);

GetLocation("mehmet emin");

x=0;

for(x; x<5; x+=1){

        if(x==6){

                y=3;

        };

};

x=10;

while(x>1){

        for(x; x<5; x+=1){

                y+=3;

        };

};

getME();

Array arraylist = Array();
```

```
        return 0;

}

getME(){

        str myname="whoIam";

        print myname;

        int go=10;

        while(go>1){

                for(go; go<5; go+=1){

                        go+=3;

                };

        };

}
```

Challenges While Developing This Programming Language:

There are few challenges occurred while we were developing our BNF. First of all, we designed our BNF as ambigious. After that, we wrote almost the whole grammar rules, than we wanted to test our grammar rules, and wrote a test.txt. After we tested our BNF grammar, we faced some errors. Hence, we noticed that BNF should have been unambigious. That's why, we spent almost 2 days to fix it.

The second and the last one was the warning messages when we compile a program. Early of the development, we was not able to give a warning message, at least which line the error occurred in. It was only saying : "error in compilation". Then we figured out a way to write a warning message with line number. Now it prints first line of the error with syntax error in it. But we could not print error messages with the type of the error,column and row number with it like : "text.txt:17:12: makes pointer from integer without a cast ". And also we could not do match more than one syntax error in program.