HACETTEPE UNIVERSITY DEPARTMENT OF

COMPUTER ENGINEERING

BBM 301 PROJECT

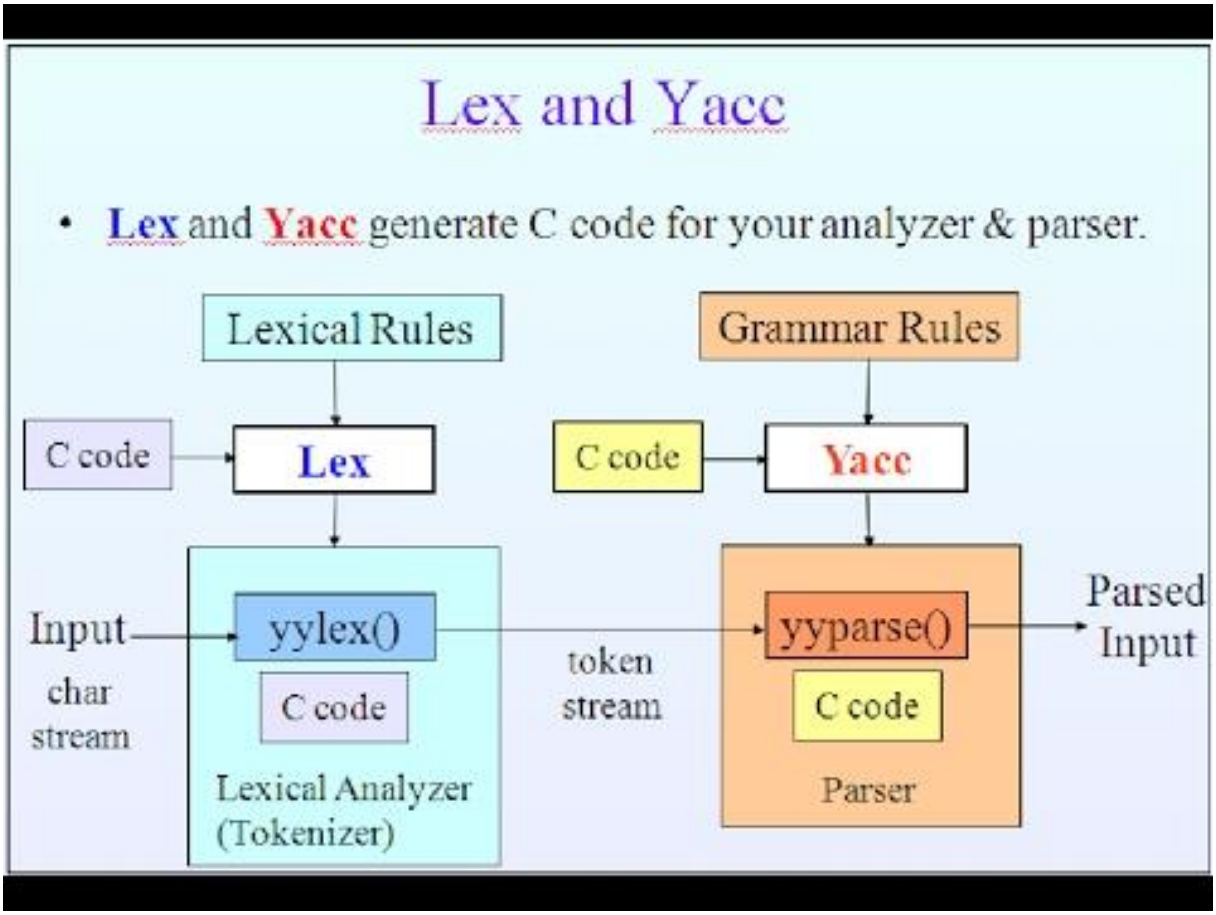
Burcu Öztaş – 21483435

Hatice ACAR – 21526599

Mehmet Taha USTA – 21527472

1-) Introduction

In this assignment:Requested from us is to designing a programming language by using lex and yacc structure to run a navigation program .This program should be able to solve basic Geographical Maps and Satellite (GPS) applications things. For instance determine the user's own location, target of address, calculating. road speed and searching to be request location at the maps.
Structure of lex & Yacc in picture;



2-) Problem Solution

We developed Lex & Yacc structure with some inputs.And with using BNF structure, rules, tokens in order to be write the this programming language.After generating this structure & test(input).txt, We check the function & rules for the project requirement.
If lex & yacc files match the input file such as function, token, data type etc. Program doesn't handle any error,If there is an error,Program will give error message.At the result, we can check this structure works true or not.

2.1-) Short descriptions of the defined tokens

 # Variable Data Types

% token INT FLOAT BOOL VOID CHAR ARRAY FILETYPE DIRTYPE

 # Boolean Data Types

% token BLN_FALSE BLN_TRUE

 # Logical Operations

% token AND_OPT OR_OPT

 # Control Flows

% token IF ELSE SWITCH CASE DEFAULT

 # Loops

% token WHILE DO BREAK CONTINUE FOR

 # Functions Decleration

% token FUNCTION RETURN

 # GPS functions

% token BLTIN_PRINT SHOW_ON_MAP SEARCH_LOCATION GET_ROAD_SPEED GET_LOCATION SHOW_TARGET GET_THE_TIME CALCULATE_THE_DISTANCE

% token TWO_LOCATION_DISTANCE FIND_THE_POPULAR_LOCATION COMMAND_ABOUT_LOCATION COLLABORATION_WITH_OTHER_USERS SHARE_SCORE SETTING

 # Conditions

% token LESSEQ_OPT GREATEREQ_OPT NEQ_OPT EQ_OPT LESS_OPT GREATER_OPT

 # Assignment Operations

%token DIVIDE_ASSIGNMENT_OPT ASSIGNMENT_OPT MULTIPLY_ASSIGNMENT_OPT MODE_ASSIGNMENT_OPT ADD_ASSIGNMENT_OPT SUB_ASSIGNMENT_OPT POW_ASSIGNMENT_OPT

 # Some Operations

%token INCREMENT_OPT DECREMENT_OPT NOT_OPT

 # Mathematical Operations

%token MULTIPLY_OPT DIVIDE_OPT MODE_OPT ADD_OPT SUB_OPT POW_OPT

 # Literals Declerations

%token INT_LTRL FLT_LTRL STR_LTRL CHR_LTRL IDNTF

 # Independent Instructions

%token SEMICOLON LEFT_BRACKET RIGHT_BRACKET COMMA COLON LEFT_PARANTHESIS RIGHT_PARANTHESIS LEFT_SQ_BRACKET RIGHT_SQ_BRACKET NEW_LINE WHITE_SPACE UNKNOWN_CHAR

2.2-) BNF Structure

Backus-Naur notation (BNF) is a formal mathematical way to describe a language. It is used to formally define the grammar of a language, so that there is no disagreement or ambiguity as to what is allowed and what is not. There can be no disagreement on what the syntax of the language is, and it makes it much easier to make compilers, because the parser for the compiler can be generated automatically with a compiler-compiler .So we used Yacc compiler for this. We define the tokens in above and according to these tokens we determine the rules of language.If all letters are lower case this mean is it is a nonterminal expression and it take value more than one.But if all letters are upper case then this is a terminal expression and it isn't parse ,take one value. The explanitions of the some rules are in below for an example:

data_type: CHAR| INT | FLOAT | BOOL| FILETYPE| DIRTYPE;

➔we can use as in input ; "c" or 2 or 5.6 or true or "input.txt" or "C:\Users\USER\Pictures\Screenshots\bnf.png''

empty: /* empty */

➔" "

factor: INT_LTRL | FLT_LTRL | STR_LTRL | CHR_LTRL;

➔we used "factor" for define a variable like : int a or float b or string word , or char c etc. We can only use int ,string,float and char types.

assignment_operator: ASSIGNMENT_OPT | MULTIPLY_ASSIGNMENT_OPT | DIVIDE_ASSIGNMENT_OPT | ADD_ASSIGNMENT_OPT | SUB_ASSIGNMENT_OPT | MODE_ASSIGNMENT_OPT| POW_ASSIGNMENT_OPT;

➔we can use only these expression "=" or "=*" or "=/" or "=%" or "=+" or "=-" or "=^"

assignment: LHS assignment_operator RHS| LHS INCREMENT_OPT | LHS DECREMENT_OPT;

➔we can use like these examples ; a=b or a++ or b--

statement: assignment SEMICOLON | declaration SEMICOLON| loop | condition | function_call SEMICOLON

| BREAK SEMICOLON | CONTINUE SEMICOLON | RETURN SEMICOLON | RETURN IDNTF SEMICOLON

| RETURN factor SEMICOLON;

➔we can use like one of these examples respectively ; a++; | int a ; | for(int i; i <5; i++) | if (a==true) | getroadspeed(12); | break; | continue; | return ; | return result; | return a;

function_call: BLTIN_PRINT LEFT_PARANTHESIS identifier_list RIGHT_PARANTHESIS

| SHOW_ON_MAP LEFT_PARANTHESIS FLT_LTRL COMMA FLT_LTRL RIGHT_PARANTHESIS

| SHOW_ON_MAP LEFT_PARANTHESIS INT_LTRL COMMA INT_LTRL RIGHT_PARANTHESIS

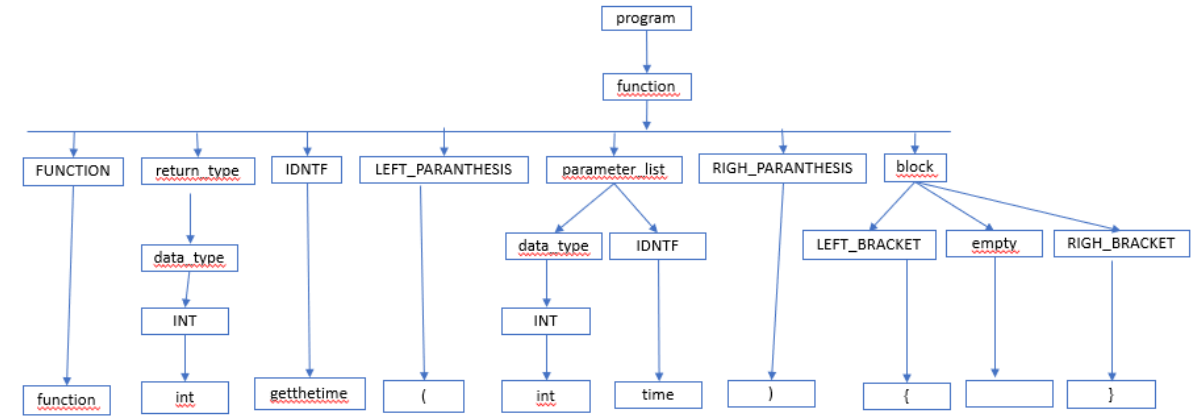
| SHOW_ON_MAP LEFT_PARANTHESIS FLT_LTRL COMMA INT_LTRL RIGHT_PARANTHESIS

| SHOW_ON_MAP LEFT_PARANTHESIS INT_LTRL COMMA FLT_LTRL RIGHT_PARANTHESIS ……. Etc.

➔we used this structure for calling the functions are in program.To give come examples ; showonmap(3,4);

searchlocation("DuisburgStreet"); getlocation("Alice");showtarget("CentralPark"); getroadspeed(12.0);calculatethedistance(3,4,5,6); getthetime(12,4); twolocationdistance("Centralpark","Hospital");

This an examle parse tree for function definition:



2.3-) Short Tutorial

 # Command Line

```
// this isa li        ne      commen      t
/*  this is Hacettepeee*/
```

 # Variable Decleration

```
int i = 2;     char c = 'c';     float a = 5.44;     bool d = true;
```

 # Variable Increment or Decrement

```
i =+ 3;      i--;
```

 # Mathmetical Operations

```
i = 3 + 5; i = 4 - 6; i = 5 * 2; i = 7 / 8;
```

```
#Control Flow
if(i < 5) {
      print("i is less than 5");
 }
```

 # Array decleration

```
array int arr1[6] = {1, 2, 3, 4, 5, 6};
```

 # Loops

```
//for loop
   for(int j = 0; j < 6; j++) {
      print("%d\n", j);
   }
//while loop
   int i = 0;
   while(i < 6) {
      print("%d\n", i);
      i++;
   }


//do while loop
```

```
   int i = 0;
   do {
      print("%d\n", i);
      i++;
   } while(i < 6);


//Switch case
int i = 0;
   while(i < 6) {
      switch(arr1[i]) {
         case 1:
            print("%d. element is 1", i);
            break;
         default:
            print("I don't know the element");
      }
      i++;
   }
```

 # Logical Operations

```
   if(i == 4 && j == 6) {
      print("i is 4 and j is 6");
   }
```

 # Functions declerations

```
function int main(void) {
          return 0;
}
```

Notice About Unique Solution Our Language;

We have the some internal ambiguity solutions of yacc is the mathematical operator precedence for the solve ambiguity of our language.We defined the operation process with order for the blocking ambiguity.And the similar function has a different parameter data type.

3-) Reference

1-) https://berthub.eu/lex-yacc/cvs/output/lexyacc.html#toc3

2-) https://github.com/konieshadow/lex-yacc-examples

3-) http://www.cs.bilkent.edu.tr/~guvenir/courses/CS315/lex-yacc/linux.html

4-) https://github.com/soham1705/lex_yacc_tutorial

5-) https://www.geeksforgeeks.org/lex-program-count-number-words/

6-) https://www.geeksforgeeks.org/yacc-program-to-recognize-string-with-grammar-anbn-n0/

7-) https://github.com/dasunpubudumal/lex-yacc

8-) https://github.com/aneeshmg/Lex-Yacc

9-) http://ashimg.tripod.com/example.html