Formal Languages and Compiler Design

Lab 9 – Documentation

Ariadna Vilasó Escarp

Link to the repository:

https://github.com/UO270119/Formal-Languages-and-Compiler-Design

Problem Statement:

Use yacc. You may use any version (yacc or bison)

1. Write a specification file containing the production rules corresponding to the language specification (use syntax rules from lab1).
2. Then, use the parser generator (no errors)
   Deliverables: lang.y (yacc specification file)

BONUS: modify lex to return tokens and use yacc to return string of productions

To solve this, I created the *lang.y* file following the example given in Moodle. I added my own reserved words and production rules according to what I had in my *syntax.in* and *token.in* file developed in previous sessions.

```
%{
#include <stdio.h>
#include <stdlib.h>

#define YYDEBUG 1
%}

%token regular
%token read_n
%token check
%token entonces
%token checkif
%token not
%token loop
%token var
%token rango_a_b
%token input
%token show
%token end
%token y
%token o
%token matriz
%token haz
%token mientras
%token caracter
%token constante
%token programa
%token empieza
%token acaba
%token de
%token BOOLEAN
%token CHAR
%token INTEGER
%token REAL
%token IDENTIFIER
```

```
%token CONSTANT
%token DOS_PUNTOS
%token PUNTO_Y_COMA
%token COMA
%token PUNTO
%token MAS
%token MENOS
%token POR
%token ENTRE
%token DIVISION
%token MODULO
%token LEFT_PARENTESIS
%token RIGHT_PARENTESIS
%token LEFT_CORCHETE
%token RIGHT_CORCHETE
%token LEFT_LLAVE
%token RIGHT_LLAVE
%token INTERROGANTE
%token LESS_THAN
%token LESS_OR_EQUAL_THAN
%token EQUAL
%token NOT_EQUAL
%token GREATER_OR_EQUAL_THAN
%token GREATER_THAN
%token ASSIGNMENT
%token AND_OPERATOR
%token OR_OPERATOR

%start program

%%

program : var decllist PUNTO_Y_COMA cmpdstmt PUNTO ;
decllist : declaration PUNTO_Y_COMA decllist | declaration;
declaration : IDENTIFIER DOS_PUNTOS type ;
type : type1 | arraydecl ;
type1 : BOOLEAN | CHAR | INTEGER | REAL ;
arraydecl : matriz LEFT_CORCHETE CONSTANT RIGHT_CORCHETE de type1 ;
cmpdstmt : empieza stmtlist acaba ;
stmtlist : stmt | stmt PUNTO_Y_COMA stmtlist ;
stmt : simplstmt | structstmt ;
simplstmt : assignstmt | iostmt ;
assignstmt : IDENTIFIER ASSIGNMENT expression ;
expression : expression MAS term | term ;
term : term POR factor | factor ;
factor : LEFT_PARENTESIS expression RIGHT_PARENTESIS | IDENTIFIER ;
iostmt : input | show LEFT_PARENTESIS IDENTIFIER RIGHT_PARENTESIS ;
structstmt : cmpdstmt | ifstmt | whilestmt ;
ifstmt : check condition entonces stmt LEFT_CORCHETE not stmt
RIGHT_CORCHETE ;
whilestmt : mientras condition haz stmt ;
condition : expression RELATION expression ;
RELATION : LESS_THAN | LESS_OR_EQUAL_THAN | EQUAL | NOT_EQUAL |
GREATER_OR_EQUAL_THAN | GREATER_THAN ;

%%

yyerror(char *s)
{
  printf("%s\n", s);
}
```

```
extern FILE *yyin;

main(int argc, char **argv)

{

  if (argc > 1)

    yyin = fopen(argv[1], "r");

  if ( (argc > 2) && ( !strcmp(argv[2], "-d") ) )

    yydebug = 1;

  if ( !yyparse() )

    fprintf(stderr,"\t No errors! \n");

}
```

Then, I compiled the program and executed it using p1.txt with the following commands:

flex lang.lxi

bison -d lang.y

gcc lex.yy.c lang.tab.c -o result

result.exe p1.txt