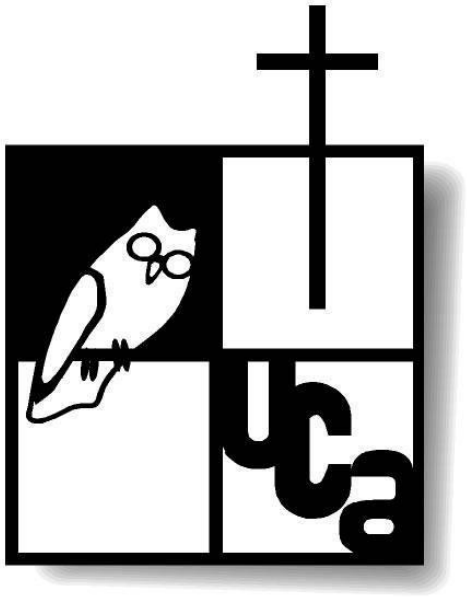

UNIVERSIDAD CENTROAMERICANA

Analizador sintáctico y manejador de errores

ALUMNOS:

José Alejandro Olmedo Araniva 00097017

Raúl Fabricio Calderón Jiménez 00004017

Alexis Javier Gómez Coto 00122717

André Vladimir González Hernández 00103017

Karina Daniela Mina Martínez 00012217

ANTIGUO CUSCATLÁN,

26 de noviembre del 2021

- Parser / analizador sintáctico
  Se encarga de revisar el texto de entrada con base a una sintaxis definida y utilizando métodos sistemáticos basados en gramáticas libres del contexto. También se encarga de funciones del analizador semántico como la revisión de tipos y ámbitos de las variables.

  - Tipo de datos:
    - int
    - char
    - float
  - Variables
  - Instrucciones condicionales
    - if
    - else
  - Declaración y definición de funciones
  - Manejo de comentarios
  - Palabras claves/reservadas
    - char
    - int
    - float
    - return
    - void
  - Operadores aritméticos
    -
  - Operadores lógicos

  1. Instrucciones de iteración
  2. Directivas de inclusión y definición de macros
  3. Funciones para leer desde teclado

- Diagramas de sintaxis: notación BNF
  La notación de Backus-Naur es un metalenguaje usado para expresar gramáticas libres de contexto, es decir, una manera formal de describir lenguajes formales.

- Árbol sintáctico: análisis descendente con una leída de izquierda a derecha y las derivaciones en el mismo sentido o recursividad
- Herramienta que ayudan a la generación del parser: YACC
- Implementado en Python
- Notifica al usuario sobre los errores sintácticos encontrados, sin detenerse al encontrar un error, sino que lo notifica y continua la revisión

- Almacena la tabla de símbolo que ayuda en las comprobaciones semánticas y facilita la generación del código objeto
    - Tipo de variable
    - Valor
    - Número de línea
    - Ámbito del identificador

Lista de tokens

"NUMBER",

"LETTER",

"PLUS",

"MINUS",

"TIMES",

"DIVIDE",

"MODULUS",

"AND",

"OR",

"NOT",

"EQUALS",

"LESS",

"GREATER",

"LPAREN",

"RPAREN",

"HEADER",

"ID",

"COMMA",

"SEMICOLON",

"APOST",

"QUOTE",

"LBRACE",

"RBRACE",

"LBRACKET",

“RBRACKET",

"POUND",

"COMMENT",

"COMMENTBLOCK",

"ASSIGN",

"EOF",

"INT",

"CHAR",

"FLOAT",

"IF",

"ELSE",

"DO",

"WHILE",

"RETURN",

"VOID",

"DEFINE",

"INCLUDE",

Análisis descendente subset de C

El *subset* que definimos para nuestra gramática de análisis descendente y que seguidamente construimos su respectiva tabla de parseo, contiene definición de variables con su respectivo tipo, definición de varias variables en una línea, expresiones que soporten operaciones aritméticas y lógicos, la sentencia condicional *if* con sus respectiva expresión a evaluar y cuerpo del *if*, así como también el bucle *while*.

La gramática es la siguiente:

S -> D S //Inicio

S -> ' '

D -> R A ; //Tipo de dato y variables

D -> A ; //variable

D -> I //IF STATEMENT

D -> W //IF STATEMENT

I -> if (E) { I1 } I2 // If

I1 -> D I1 // Cuerpo del if

I1 -> ' '

I2 -> else { I1 } //Else opcional

I2 -> ' '

W -> while (E) { I1 } //Bucle while

R -> int //Tipos de datos

R -> float

R -> char

R -> void

A -> id = E A1 //Asignacion

A1 -> , A // Asignacion inline

A1 -> ' '

E -> T E1 //Expresiónes

E1 -> + T E1 //Soporte de suma y resta

E1 -> - T E1

E1 -> ' '

T -> F T1 //Multiplicación y división separadas por el orden de las operaciones

T1 -> * F T1

T1 -> / F T1

T1 -> L F T1 //Para operadores lógicos

T1 -> ' '

F -> (E) //Encuentra la Expresion

F -> C

C -> num //Terminal para las expresiones

C -> letter

C -> id

L -> == //Operadores Logicos

L -> &&

L -> ||

L -> >

L -> <

Tabla de parseo

Una vez construida la gramática, se procede a construir la tabla de parseo:

https://drive.google.com/file/d/1GmnHCTlIwmP20y2ZPZGiVz_vY3EkQz8i/view?usp=sharing

Gramática utilizada con YACC

```
def p_program(p):
    """
    program : function program
            | external-declaration program
            | empty
    """
```

Declaraciones y asignaciones

```
def p_external_declaration(p):
    """
    external-declaration : type assignment SEMICOLON
                         | array_usage SEMICOLON
                         | type array_usage SEMICOLON
                         | macro_definition
                         | file_inclusion
    """



def p_declaration(p):
    """
    declaration : type assignment SEMICOLON
                | assignment SEMICOLON
                | function_call SEMICOLON
```

```
                | array_usage SEMICOLON
                | type array_usage SEMICOLON
    """



def p_assignment(p):
    """
    assignment : ID ASSIGN assignment
               | ID ASSIGN function_call
               | ID ASSIGN array_usage
               | array_usage ASSIGN assignment
               | ID COMMA assignment
               | NUMBER COMMA assignment
               | ID PLUS assignment
               | ID MINUS assignment
               | ID TIMES assignment
               | ID DIVIDE assignment
               | ID MODULUS assignment
               | NUMBER PLUS assignment
               | NUMBER MINUS assignment
               | NUMBER TIMES assignment
               | NUMBER DIVIDE assignment
               | NUMBER MODULUS assignment
               | APOST assignment APOST
               | LPAREN assignment RPAREN
               | MINUS assignment
               | NUMBER PLUS PLUS
               | ID PLUS PLUS
               | array_usage PLUS PLUS
               | NUMBER MINUS MINUS
               | ID MINUS MINUS
               | array_usage MINUS MINUS
```

```
                | NUMBER
                | ID
                | LETTER
    """
```

Bloques de código, expresiones y funciones

```
def p_function_call(p):
    """
    function_call : ID LPAREN RPAREN
                  | ID LPAREN assignment RPAREN
    """


def p_array_usage(p):
    """
    array_usage : ID LBRACKET assignment RBRACKET
    """


def p_function(p):
    """
    function : type ID LPAREN argument_list_option RPAREN
compound_statement

    argument_list_option : argument_list
                         | empty

    argument_list : argument_list COMMA argument
                  | argument

    argument : type ID
```

```
    compound_statement : LBRACE statement_list RBRACE


    statement_list : statement_list statement
                   | empty


    statement : iteration_statement
              | declaration
              | selection_statement
              | return-statement
    """



def p_type(p):
    """
    type : INT
         | FLOAT
         | CHAR
         | VOID
    """



def p_iteration_statement(p):
    """
    iteration_statement : WHILE LPAREN expression RPAREN
compound_statement
                        | WHILE LPAREN expression RPAREN statement
                        | DO compound_statement WHILE LPAREN expression
RPAREN SEMICOLON
                        | DO statement WHILE LPAREN expression RPAREN
SEMICOLON
    """
```

```
def p_selection_statement(p):
    """
    selection_statement : IF LPAREN expression RPAREN compound_statement
                        | IF LPAREN expression RPAREN statement
                        | IF LPAREN expression RPAREN compound_statement
ELSE compound_statement
                        | IF LPAREN expression RPAREN compound_statement
ELSE statement
                        | IF LPAREN expression RPAREN statement ELSE
compound_statement
                        | IF LPAREN expression RPAREN statement ELSE
statement
    """



def p_return_statement(p):
    """
    return-statement : RETURN SEMICOLON
                     | RETURN expression SEMICOLON
    """



def p_expression(p):
    """
    expression : expression EQUALS expression
               | expression LESS expression
               | expression GREATER expression
               | expression AND expression
               | expression OR expression
               | NOT expression
               | assignment
               | array_usage
    """
```

```
def p_macro_definition(p):
    """
    macro_definition : POUND DEFINE ID assignment
    """
def p_file_inclusion(p):
    """
    file_inclusion : POUND INCLUDE LESS HEADER GREATER
                   | POUND INCLUDE QUOTE HEADER QUOTE
    """
```

Link al repositorio:

https://github.com/alejandroolmed59/Proyecto_final_teo