Analizador Léxico

Profesor:

Ing. Sleyter Angulo

Proyecto II

Estructura de datos II BISOFT-28

Ingeniería del software

Integrantes:

Jesé Abraham Chávez

Daniel Hernandez Sanchez

III Cuatrimestre 2021

Índice

1 Scanner

1.1 Proceso de scanning

El proceso de scanning, también conocido como análisis léxico, es la primera fase del proceso de compilación.

Éste es realizado por un Scanner o analizador léxico, que en líneas generales, se trata de un programa que recibe a modo de entrada el código fuente de otro, y a partir de las secuencias de caracteres contenidas en éste, genera una serie de tokens, cuyo ciclo de vida continua en un analizador sintáctico o Parser.

Cada token generado por el Scanner representa un componente léxico o símbolo, cuyo patrón es detectado mediante expresiones regulares. Dichas expresiones pueden ajustarse según los requerimientos del lenguaje con el que se esté tratando.

Las expresiones regulares que definen los tokens del lenguaje son revisadas por un autómata de estados finitos, encargado de procesar carácter a carácter el código fuente pasado como entrada, y retornar un token por cada estado final alcanzado.

En el caso de que existan más de dos patrones en los que una secuencia de caracteres pueda alcanzar un estado final, se toma la que mayor cantidad de coincidencias individuales posea.

El proceso de scanning implementado en este proyecto no se separa del proceso descrito anteriormente, al menos no en el orden y objetivo de cada paso a realizar.

1.1 Proceso de scanning

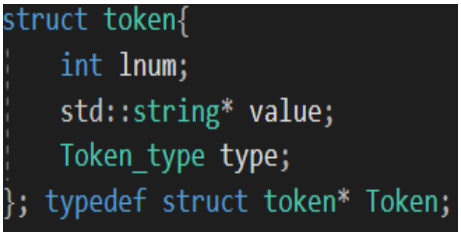
Apoyados en la herramienta Flex, - de la que se hablará con mayor detalle más adelante -, y en un conjunto de expresiones regulares que representan al lenguaje de programación Pascal, generamos un autómata de estados finitos en C++, que retorna los tokens a otro programa escrito en el mismo lenguaje, cuyo trabajo es la realización de este documento.

Ambos programas utilizan el archivo de cabecera Token.h, que contiene una colección de tipos de token, así como la forma de tratarlos al momento de ser representados en la salida.

```
#include <string>

enum token_types {
    DIGIT, IDENTIFIER, INTEGER, FLOAT, RSVWORD, OPRT, EMPTYSPC, STXSymbl, FUNTC, DTTYPE,
    COMPARATOR,STRING,CHAR,BADIDENTIFIER,BADIDENTIFIERSIZE,BADSTRSTART,BADSTREND,LINEBRK,COMMENTTOKEN
}; typedef token_types Token_type;

static const char *enum_str[] =
{ "Digit", "Identifier", "Integer", "Real", "Keyword", "Operator", "BlankSpace","Syntax Symbol", "Function", "DataType",
  "Comparator", "String", "Char","Identificador mal formado.", "Identificador demasiado grande.","Cadena mal abierta.","Cadena mal cerrada.","Line Break","COMMENT"};

static const char *enum_color[] = {
    "Black", "MoradoChogath", "LBlue", "PYellow", "DBlue", "LOrange", "Black", "Black", "Black", "LRed",
    "PPGreen","PGreen","PBlue","Black","Black","Red","Red","Black","Black"
};

static const char *enum_nbold[]{
    "NAN","BOLD","NAN","NAN","BOLD","BOLD","NAN","BOLD","NAN","BOLD",
    "BOLD","NAN","NAN","BOLD","BOLD","BOLD","BOLD","NAN","BOLD"
};

static const char *enum_emph[]{
    "NAN","NAN","NAN","NAN","NAN","NAN","NAN","NAN","NAN","NAN",
    "NAN","NAN","NAN","EMPH","EMPH","EMPH","EMPH","NAN","NAN"
};

static const char *enum_code_color[] = {
    "Black","PPGreen","LOrange","LOrange","DBlue","Black","Black","DGray","Black","MoradoChogath",
    "Black","DPumpkin", "DPumpkin","LRed","LRed","LRed","LRed","Black","MGreen"
};
```

1.1 Proceso de scanning

Los tokens son almacenados en la siguiente estructura:



Cada token detectado se almacena en una cola que servirá para recomponer el código fuente en la sección correspondiente de este documento.

1.2 Flex

Flex es un software open source desarrollado alrededor del año 1987 por Vern Paxson en el lenguaje C. Flex significa Fast Lexical analyzer generator y como el nombre lo menciona, esta herramienta nos permite realizar y generar un analizador léxico para diversos propósitos, a estos analizadores léxicos se les denomina "Scanners" o "Lexers". Para su complementar su funcionalidad, esta herramienta se puede llegar a utilizar junto con Berkeley Yacc Parsser Generator ó GNU Bison parser generator, ambos siendo software para el análisis de texto.

Con la herramienta Flex, podemos generar un analizador léxico para cualquier lenguaje en corto tiempo, ya que a este solo se le deben de indicar los patrones a buscar, las reglas al reconocer los patrones y las acciones que se deben tomar una vez encontrados. Flex genera un archivo .l el cual contiene todas las reglas definidas por el programador. Dentro de los archivos generados, se crea una función yylex(), la cual es la responsable por ejecutar el archivo .l para comenzar a leer el documento pasado por parámetros y comenzar a reconocer y devolver los tokens encontrados.

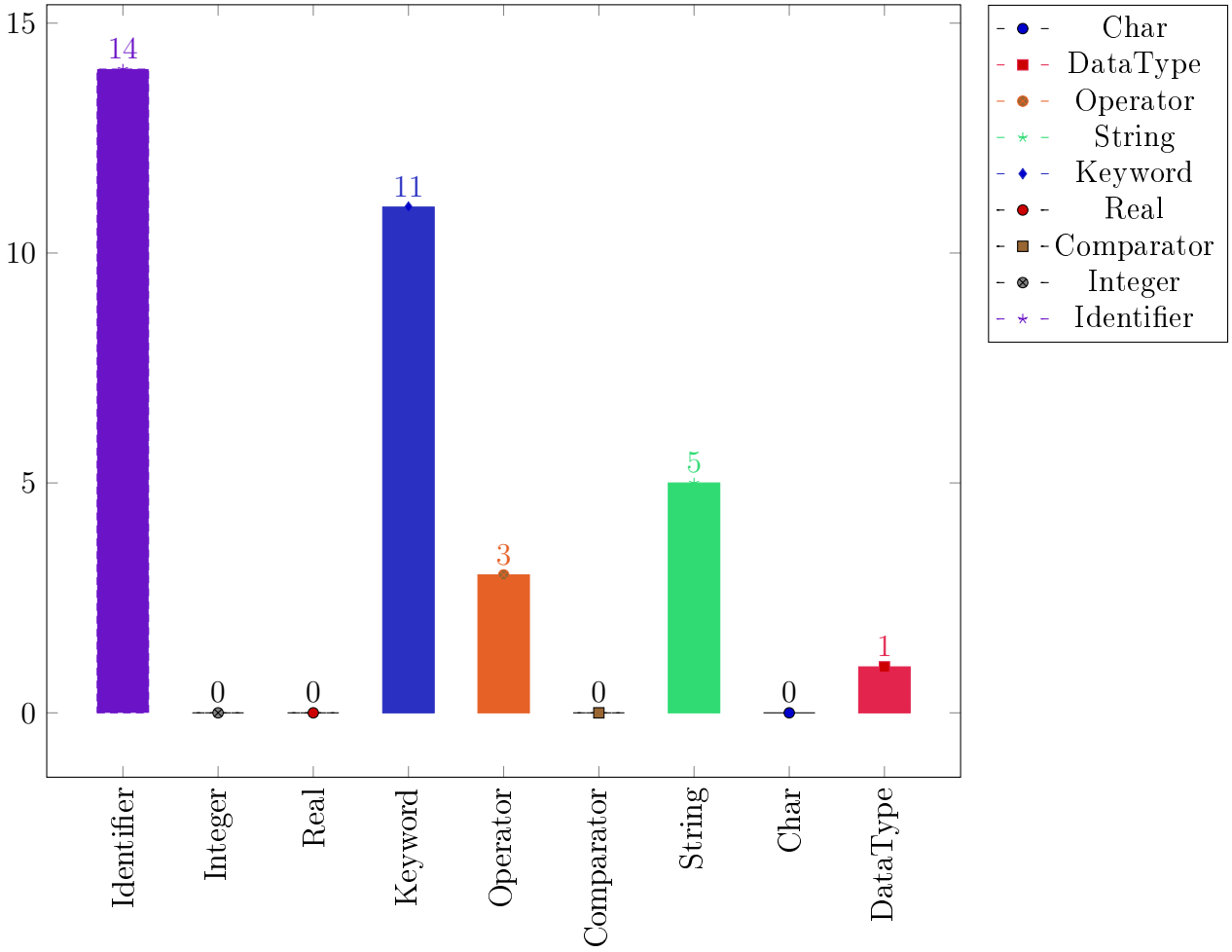
2 Resultados

2.1 Código escaneado

```
Program ejb;
uses crt;
        Var
                R,Num1,num2:integer;

Begin
clrscr;
        Writeln('Ingrese Un numero');
        Readln(Num1);
        Writeln('Ingrese Un numero');
        Readln(Num2);
        R:=Num1-Num2;
        Writeln('Al restar: ', num1,' - ',num2,' se obtiene: ',R);
readln;
ENd
```

Errores encontrados:

2.2 Histograma



2.3 Diagrama circular