**Definiciones**

Caracteres={a-z}

Numeros={0-9}

Operadores={+, -, /, \*, %, (, ), &, {, }, ;, :, [, ], |, “,”, =, <, >}

Tipos\_datos={"int", "char", "string", "float", "long", "double", "bool", "void"}

Reservados\_cpp={"for", "while", "if", "do", "switch", "else", "case" ,"return", "true", "false", "using", "namespace", "cout", "cin", "endl"}

Comentario\_simple\_cpp={//(operador|” “|numero|carácter|∊)\*/n}

Comentario\_cpp={/\*( operador|” “|numero|carácter|/n|∊)\*\*/}

Librerias\_cpp={"#include"}

Texto={“( operador|” “|numero|carácter|∊)\*”}

Elemento\_cpp={operador| Reservados\_cpp | Librerias\_cpp |” “|numero|carácter |texto}

Párrafo\_cpp={ Elemento\_cpp\* (Comentario\_simple\_cpp|/n| Comentario\_cpp)}

Reservados\_python={"def", "while", "if", "return", "for", "input", "true", "false", "else", "elif", "print", "in", "len"}

Librerias\_python={"import", "from"}

Comentario\_ simple\_python={#(operador|” “|numero|carácter|∊)\*/n }

Comentario\_python={“””( operador|” “|numero|carácter|/n|∊)\*”””}}

Elemento\_python={operador| Reservados\_python | Librerias\_python |” “|numero|carácter }

Párrafo\_python={ Elemento\_python\*(Comentario\_simple\_ python |/n| Comentario\_ python)}

Txt\_cpp={párrafo\_cpp\*}

Txt\_ Python={párrafo\_ python \*}

Fin\_Texto={EOF}

Txt={( Txt\_cpp | Txt\_ Python)Fin\_Texto}

**Autómata Finito Determinista**

Diagrama, Esquemático

Descripción generada automáticamente

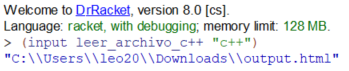
**Tabla de transición**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **A** | **B** | **C** | **D** | **E** | **F** | **G** | **H** | **I** |
| TXT | B | ᴓ | ᴓ | ᴓ | ᴓ | ᴓ | ᴓ | ᴓ | ᴓ |
| Txt\_cpp | ᴓ | C | ᴓ | ᴓ | ᴓ | ᴓ | ᴓ | ᴓ | ᴓ |
| Txt\_python | ᴓ | G | ᴓ | ᴓ | ᴓ | ᴓ | ᴓ | ᴓ | ᴓ |
| Párrafo\_cpp | ᴓ | ᴓ | D | ᴓ | ᴓ | ᴓ | ᴓ | ᴓ | ᴓ |
| Párrafo\_python | ᴓ | ᴓ | ᴓ | ᴓ | ᴓ | ᴓ | F | ᴓ | ᴓ |
| “ “ | ᴓ | ᴓ | ᴓ | ᴓ | D | ᴓ | ᴓ | F | ᴓ |
| numero | ᴓ | ᴓ | ᴓ | ᴓ | D | ᴓ | ᴓ | F | ᴓ |
| caracter | ᴓ | ᴓ | ᴓ | ᴓ | D | ᴓ | ᴓ | F | ᴓ |
| /n | ᴓ | ᴓ | ᴓ | ᴓ | C | ᴓ | ᴓ | G | ᴓ |
| operador | ᴓ | ᴓ | ᴓ | ᴓ | D | ᴓ | ᴓ | F | ᴓ |
| texto | ᴓ | ᴓ | ᴓ | ᴓ | D | ᴓ | ᴓ | ᴓ | ᴓ |
| Librería\_cpp | ᴓ | ᴓ | ᴓ | ᴓ | D | ᴓ | ᴓ | ᴓ | ᴓ |
| Reservado\_cpp | ᴓ | ᴓ | ᴓ | ᴓ | D | ᴓ | ᴓ | ᴓ | ᴓ |
| Elemento\_cpp | ᴓ | ᴓ | ᴓ | E | ᴓ | ᴓ | ᴓ | ᴓ | ᴓ |
| Comentario\_cpp | ᴓ | ᴓ | ᴓ | ᴓ | C | ᴓ | ᴓ | ᴓ | ᴓ |
| Comentario\_simple\_cpp | ᴓ | ᴓ | ᴓ | ᴓ | C | ᴓ | ᴓ | ᴓ | ᴓ |
| Librería\_python | ᴓ | ᴓ | ᴓ | ᴓ | ᴓ | ᴓ | ᴓ | F | ᴓ |
| Reservado\_ python | ᴓ | ᴓ | ᴓ | ᴓ | ᴓ | ᴓ | ᴓ | F | ᴓ |
| Elemento\_ python | ᴓ | ᴓ | ᴓ | ᴓ | ᴓ | H | ᴓ | ᴓ | ᴓ |
| Comentario\_ python | ᴓ | ᴓ | ᴓ | ᴓ | ᴓ | ᴓ | ᴓ | G | ᴓ |
| Comentario\_simple\_python | ᴓ | ᴓ | ᴓ | ᴓ | ᴓ | ᴓ | ᴓ | G | ᴓ |
| Fin\_Texto | ᴓ | I | ᴓ | ᴓ | ᴓ | ᴓ | ᴓ | ᴓ | ᴓ |

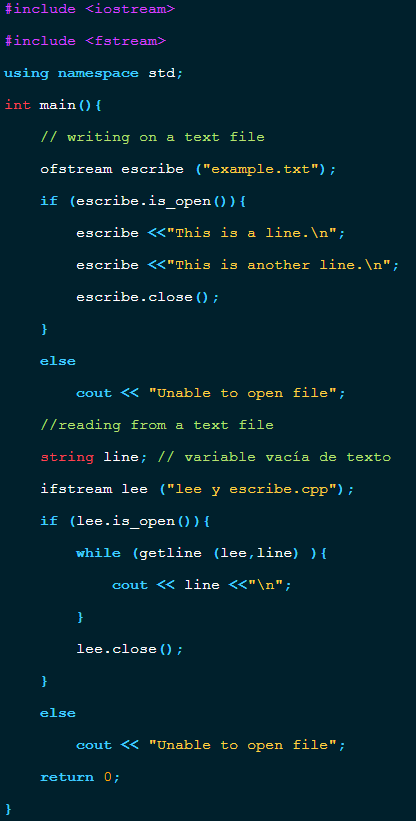
**Input ejemplo cpp**

|  |
| --- |
| #include <iostream>  #include <fstream>  using namespace std;  int main(){  // writing on a text file  ofstream escribe (“example.txt”);  if (escribe.is\_open()){  escribe <<”This is a line.\n”;  escribe <<”This is another line.\n”;  escribe.close();  }  else  cout << “Unable to open file”;  //eading from a text file  string line; // variable vacía de texto  ifstream lee (“lee y escribe.cpp”);  if (lee.is\_open()){  while (getline (lee,line) ){  cout << line <<”\n”;  }  lee.close();  }  else  cout << “Unable to open file”;  return 0;  } |

**Output racket ejemplo cpp**



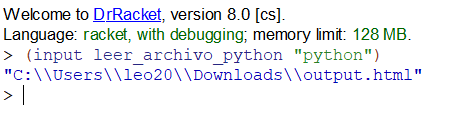
**Output ejemplo html**



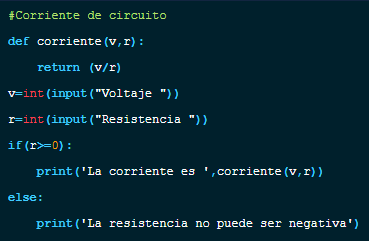
**Input ejemplo python**

|  |
| --- |
| #Corriente de circuito  def corriente(v,r):  return (v/r)  v=int(input("Voltaje "))  r=int(input("Resistencia "))  if(r>=0):  print('La corriente es ',corriente(v,r))  else:  print('La resistencia no puede ser negativa') |

**Output racket ejemplo python**



**Output ejemplo html**



**Reflexión**

En este programa lo que hacemos es obtener el contenido de un archivo de texto, carácter por carácter, para después poder comparar lo recibido con listas predefinidas según el lenguaje en el que estemos trabajando, en nuestro caso c++ y Python. Primero se comparan los caracteres con varias cadenas de texto, en caso de que no coincida, este se agrupa con otros previos, para poder compararla ahora con palabras de más de un carácter, este proceso se repite hasta terminar de leer el txt, y lo que se hace cada que se identifica el tipo de carácter que estamos leyendo, ya sea un tipo de dato, operador, entero, entre otras, se escribe un html separando por categorías para así tener un css que le dé formato.

Para obtener la complejidad de nuestro programa, primero nos damos cuenta de que estamos leyendo un txt carácter por carácter, por lo que nuestro documento de inicio ya os da un O(n), pero después en nuestras comparaciones para conocer el tipo de dato que recibimos, contamos con una función la cual va creando palabras con los elementos dados, por lo que nuevamente se recorre el arreglo haciendo así un O(n 2 ), pero después cada que se agrega una letra a la palabra formada, buscamos relación entre esta palabra y una lista con palabras definidas, por lo que contamos con una triple anidación dándonos así un O(). Por este orden, podemos decir que nuestro programa no es muy eficiente, ya que, si se introduce un txt muy extenso y nuestras listas definidas dentro del programa estuvieran completas con todas las palabras reservadas del lenguaje, podría ser un poco tardada la compilación.

Estos programas pueden ser usados como buscadores, dando coincidencias sobre lo que el usuario este buscando con información con la que ya se cuenta en una base de datos, dando así un mejor servicio al cliente o usuario.

En cuanto a la parte ética, este tipo de programas se podría usar para notar falsificaciones, teniendo un formato con el cual comparar lo recibido y así poder ver las diferencias entre estos y poder determinar si es falso lo que se recibe.