Sebastián Rojas Salazar - A01637557
Diego Velazquez - A01632240
Pablo Blanco - A01637761

Actividad 3.2 Programando un DFA - Deterministic finite automata

Instrucciones

En la actividad se nos pide realizar un programa que al recibir de entrada un archivo de texto, el cual contenga expresiones aritméticas y comentarios, debe regresar una tabla que indica todos los tokens encontrados en el archivo y de qué tipo son.

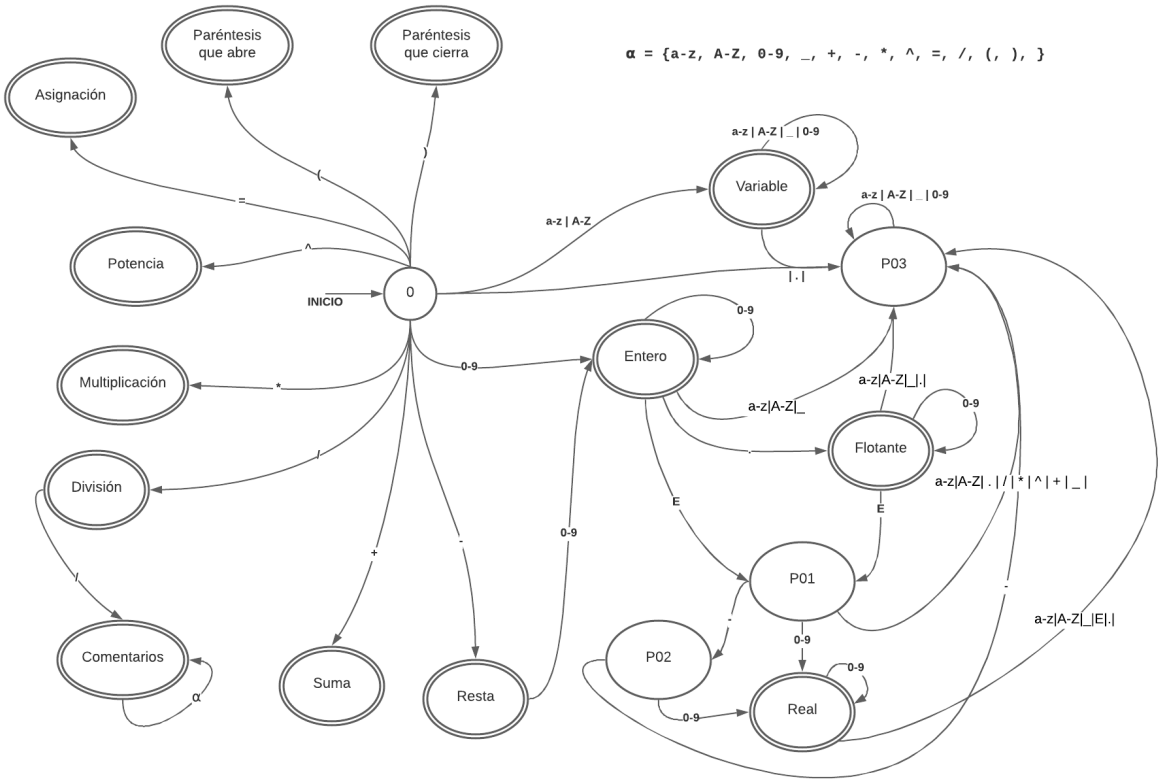
Solución de la actividad

Para realizar esta actividad utilizamos los conocimientos adquiridos durante clase sobre los autómatas finitos deterministas. Antes de describir cómo lo implementamos primero definiremos que es un autómata finito determinista.

Los autómatas finitos son parte del tema de lenguajes regulares y estos son abstracción de una máquina, describiendo sus estados y cómo estos cambian. Estos autómatas tienen un conjunto finito de estados y entradas. La parte determinista se refiere a que para cualquier entrada existe solo un estado al cual pueden transicionar desde el estado en el que se encuentra. No puede haber dos posibilidades de transición dada una entrada. Tiene que contar con un estado inicial y puede tener varios estados finales.

En la actividad implementamos el concepto de autómatas finitos deterministas al realizar el diagrama de cómo cambiarían sus estados dado el archivo de texto. Después de leer el archivo de texto se analiza línea por línea los caracteres y se toman como tokens. Por cada carácter leído el estado cambia dependiendo de si el mismo se encuentra en el vocabulario del autómata. Al reconocer el carácter o conjunto de caracteres se clasifica el token. Seguido de esto se escribe en el archivo de resultado cada token leído y su tipo de dato.

Diagrama del funcionamiento del autómata



[Automata: Lucidchart]

Requerimientos para ejecutar el programa

- Para usuario de Macos : Ingresar el siguiente comando en tu terminal “curl https://raw.githubusercontent.com/pablo-blancoc/TC2037/master/DFA/install-mac.sh > install.sh && bash install.sh”

- Se crea una nueva carpeta “Brucelords” con todos los archivos necesarios, se debe entrar a ella para ejecutar el programa con el comando: cd Brucelords
- Para correr el programa recuerda activar el ambiente virtual usando el comando: source venv/bin/activate
- Para correr el programa recuerda tener tu archivo para leer en esa carpeta
- Para correr el programa solo debes usar el comando: python automata.py
- Para desactivar el ambiente virtual usa el comando: deactivate
- El resultado lo entrega en un nuevo archivo llamado result.txt

- Para usuarios de Windows: Abrir el siguiente link:

  https://tc2037.herokuapp.com/ que te dirigirá a la página web y después subir un archivo .txt al Autómata finito determinístico dentro de la página web.

- Una vez subido el archivo la misma página web redirige a una pestaña con el resultado.

Ejemplo de ejecución del programa con archivo de texto ejemplo

Entrada:

```
1a.a1=1-1a
#include <iostream>
using namespace std;

int main()
{
   char line[150];
   int vowels, consonants, digits, spaces;

   vowels =  consonants = digits = spaces = 0;

   cout << "Enter a line of string: ";
   cin.getline(line, 150);
   for(int i = 0; line[i]!='\0'; ++i)
   {
      if(line[i]=='a' || line[i]=='e' || line[i]=='i' ||
         line[i]=='o' || line[i]=='u' || line[i]=='A' ||
         line[i]=='E' || line[i]=='I' || line[i]=='O' ||
         line[i]=='U')
      {
         ++vowels;
      }
      else if((line[i]>='a'&& line[i]<='z') || (line[i]>='A'&& line[i]<='Z'))
      {
         ++consonants;
      }
      else if(line[i]>='0' && line[i]<='9')
      {
         ++digits;
      }
      else if (line[i]==' ')
      {
         ++spaces;
      }
   }

   cout << "Vowels: " << vowels << endl;
```

```
    cout << "Consonants: " << consonants << endl;
    cout << "Digits: " << digits << endl;
    cout << "White spaces: " << spaces << endl;

    return 0;
}
```

Salida:

```
1a.a1                   Error
=                       Asignación
1                       Entero
-1a                     Error
#include                ERROR: Invalid character
<iostream>              ERROR: Invalid character
using                   Variable
namespace               Variable
std;                    ERROR: Invalid character
int                     Variable
main                    Variable
(                       Paréntesis que abre
)                       Paréntesis que cierra
{                       ERROR: Invalid character
char                    Variable
line[150];              ERROR: Invalid character
int                     Variable
vowels,                 ERROR: Invalid character
consonants,             ERROR: Invalid character
digits,                 ERROR: Invalid character
spaces;                 ERROR: Invalid character
vowels                  Variable
=                       Asignación
consonants              Variable
=                       Asignación
digits                  Variable
=                       Asignación
spaces                  Variable
=                       Asignación
0;                      ERROR: Invalid character
cout                    Variable
```

<<	ERROR: Invalid character
"Enter	ERROR: Invalid character
a	Variable
line	Variable
of	Variable
string:	ERROR: Invalid character
";	ERROR: Invalid character
cin.getline	Error
(	Paréntesis que abre
line,	ERROR: Invalid character
150	Entero
)	Paréntesis que cierra
;	ERROR: Invalid character
for	Variable
(	Paréntesis que abre
int	Variable
i	Variable
=	Asignación
0;	ERROR: Invalid character
line[i]!	ERROR: Invalid character
=	Estado inicial
'\0';	ERROR: Invalid character
+	Suma
+	Suma
i	Variable
)	Paréntesis que cierra
{	ERROR: Invalid character
if	Variable
(	Paréntesis que abre
line[i]	ERROR: Invalid character
=='a'	ERROR: Invalid character
||	ERROR: Invalid character
line[i]	ERROR: Invalid character
=='e'	ERROR: Invalid character
||	ERROR: Invalid character
line[i]	ERROR: Invalid character
=='i'	ERROR: Invalid character
||	ERROR: Invalid character
line[i]	ERROR: Invalid character
=='o'	ERROR: Invalid character

|| ERROR: Invalid character
line[i] ERROR: Invalid character
=='u' ERROR: Invalid character
|| ERROR: Invalid character
line[i] ERROR: Invalid character
=='A' ERROR: Invalid character
|| ERROR: Invalid character
line[i] ERROR: Invalid character
=='E' ERROR: Invalid character
|| ERROR: Invalid character
line[i] ERROR: Invalid character
=='I' ERROR: Invalid character
|| ERROR: Invalid character
line[i] ERROR: Invalid character
=='O' ERROR: Invalid character
|| ERROR: Invalid character
line[i] ERROR: Invalid character
=='U' ERROR: Invalid character
) Estado inicial
{ ERROR: Invalid character
+ Suma
+ Suma
vowels; ERROR: Invalid character
} ERROR: Invalid character
else Variable
if Variable
(Paréntesis que abre
(Paréntesis que abre
line[i]> ERROR: Invalid character
= Estado inicial
'a'&& ERROR: Invalid character
line[i]< ERROR: Invalid character
= Estado inicial
'z' ERROR: Invalid character
|| ERROR: Invalid character
(Paréntesis que abre
line[i]> ERROR: Invalid character
= Estado inicial
'A'&& ERROR: Invalid character
line[i]< ERROR: Invalid character

=	Estado inicial
'Z'	ERROR: Invalid character
))	Paréntesis que cierra
{	ERROR: Invalid character
+	Suma
+	Suma
consonants;	ERROR: Invalid character
}	ERROR: Invalid character
else	Variable
if	Variable
(	Paréntesis que abre
line[i]>	ERROR: Invalid character
=	Estado inicial
'0'	ERROR: Invalid character
&&	ERROR: Invalid character
line[i]<	ERROR: Invalid character
=	Estado inicial
'9'	ERROR: Invalid character
)	Estado inicial
{	ERROR: Invalid character
+	Suma
+	Suma
digits;	ERROR: Invalid character
}	ERROR: Invalid character
else	Variable
if	Variable
(	Paréntesis que abre
line[i]	ERROR: Invalid character
=='	ERROR: Invalid character
'	ERROR: Invalid character
)	Estado inicial
{	ERROR: Invalid character
+	Suma
+	Suma
spaces;	ERROR: Invalid character
}	ERROR: Invalid character
}	ERROR: Invalid character
cout	Variable
<<	ERROR: Invalid character
"Vowels:	ERROR: Invalid character

" ERROR: Invalid character
<< ERROR: Invalid character
vowels Variable
<< ERROR: Invalid character
endl; ERROR: Invalid character
cout Variable
<< ERROR: Invalid character
"Consonants: ERROR: Invalid character
" ERROR: Invalid character
<< ERROR: Invalid character
consonants Variable
<< ERROR: Invalid character
endl; ERROR: Invalid character
cout Variable
<< ERROR: Invalid character
"Digits: ERROR: Invalid character
" ERROR: Invalid character
<< ERROR: Invalid character
digits Variable
<< ERROR: Invalid character
endl; ERROR: Invalid character
cout Variable
<< ERROR: Invalid character
"White ERROR: Invalid character
spaces: ERROR: Invalid character
" ERROR: Invalid character
<< ERROR: Invalid character
spaces Variable
<< ERROR: Invalid character
endl; ERROR: Invalid character
return Variable
0; ERROR: Invalid character
} ERROR: Invalid character