Programa 6 - Automata de Pila

Leon Tejeda 2CM5

Enero 2021

INTRODUCCION:
Que es una pila: Una pila (stack en inglés) es una lista ordinal o estructura de datos en la que el modo de acceso a sus elementos es de tipo LIFO (del inglés Last In First Out, último en entrar, primero en salir) que permite almacenar y recuperar datos.
PLANTEAMIENTO DEL PROBLEMA:
Implementar un automata de pila para reconocer el CFL (Context Free Language o Lenguaje libre de Contexto) 0*n 1*n — n¿=1
Adicionalmente, el programa debe de contar con las siguientes características: 1. La cadena puede ingresar por el usuario o automáticamente. 2. Mandar a un archivo y en pantalla la evaluación del autómata a través de descripciones instantáneas. 3. Animar el autómata de pila. 4. La longitud máxima será manejar cadenas de 100,000 caracteres.
DESARROLLO:
Para este problema se utilizo el lenguaje de C++, para que se viera calramente la implementacion de la pila. Se tuvo que implementar una libreria para que el compilador aceptara el tipo de dato PILA.
El programa solo debia aceptar cadenas de n cantidad de 0's, seguida de n cantidad de 1's, EJ: 0011, 00001111
Si por alguna razon la cantidad de 0's y 1's eran diferente, la cadena sera rechazada, al igual que si estan concatenados, EJ: 010101, 001, 101010
El funcionamiento de la pila seria que cada vez que encontrara un 0, se ingresaria una X dentro de la pila. De lo contrario se sacaria la X de la pila.
Se implemento un menu para ingresar la cadena de manera manual o aleatoria

LIBRERIA:

```
1
2 #include <iostream>
3 #include <string>
4 #include <fstream>
5 #include <stdlib.h>
6 #include <time.h>
7 #include <windows.h>
8 #include <conio.h>
9
10
11 struct Pila
12 {
13   char dato;
14   struct Pila * Siguiente;
15   struct Pila * Tope;
16   struct Pila * Fin;
17 };
18
19 struct Pila * Insertar(struct Pila * , char );
20
21 void Mostrar_Tope(struct Pila * );
22
23 struct Pila * Eliminar_Tope(struct  Pila * );
```

CODIGO:

```
#include "pila.h"

using namespace std;


struct Pila * Insertar(struct Pila * Mi_Pila, char dato)
{
  struct Pila * Aux;
  Aux=Mi_Pila;
  struct Pila * Nuevo;
  Nuevo=(Pila *)malloc(sizeof(struct Pila));
  Nuevo->dato=dato;
  Nuevo->Siguiente=Mi_Pila;
  Mi_Pila=Nuevo;

  if(Aux == NULL)
  {
    Mi_Pila->Fin=Nuevo;
  }

  Mi_Pila->Tope=Nuevo;

return Mi_Pila;
}

void Mostrar_Tope(struct Pila * Mi_Pila)
{
  if(Mi_Pila == NULL)
  {
    //cout << "La Pila esta Vacia" << endl;
  }
  else
  {
    cout << Mi_Pila->Tope->dato;
  }
}

struct Pila * Eliminar_Tope(struct  Pila * Mi_Pila)
{
  struct Pila * Aux;
  Aux=Mi_Pila;

  if(Mi_Pila == NULL)
  {
    //cout << "La Pila esta Vacia, No se pueden eliminar datos" <<
    endl;
  }

  else
  {
    Mi_Pila->Tope=Aux->Siguiente;
    Mi_Pila=Mi_Pila->Siguiente;
    Aux->Siguiente=NULL;
  }

return Mi_Pila;
}

void descripciones(char estado, char evaluacion, string cadena, int
     aux)
{
```

```
60   ofstream documento;
61
62   if (aux == 0)
63   {
64     documento.open("Programa6_Descripciones.txt", ios::out);
65
66     if (evaluacion == 'e')
67     {
68       documento << "d(q, 0, Z0) = {(q, XZ0)}" << endl;
69
70       cout << "d(q, 0, Z0) = {(q, XZ0)}" << endl;
71     }
72
73     //Cadena Rechazada
74     else
75     {
76       documento << "d(p, 1, X) = {(p, e)}" << endl;
77       documento << "--Cadena Invalida--" << endl;
78
79       cout << "d(p, 1, X) = {(p, e)}" << endl;
80       cout << "--Cadena Invalida--" << endl;
81     }
82   }
83
84   else
85   {
86     documento.open("Programa6_Descripciones.txt", ios::app);
87
88     if (evaluacion != 'e')
89     {
90       //Cadena Aceptada
91       if (evaluacion == 'a')
92       {
93         documento << endl;
94         documento << "d(p, e, Z0) = {(F, Z0)}" << endl;
95         documento << "--Cadena Aceptada--" << endl;
96
97         cout << endl;
98         cout << "d(p, e, Z0) = {(F, Z0)}" << endl;
99         cout << "--Cadena Aceptada--" << endl;
100      }
101
102      //Cadena Rechazada
103      else
104      {
105        if (estado == 'q')
106        {
107          documento << "d(q, 0, X) = {(q, " << cadena << ")}" <<
     endl;
108          documento << "--Cadena Invalida--" << endl;
109
110          cout << "d(q, 0, X) = {(q, " << cadena << ")}" << endl;
111          cout << "--Cadena Invalida--" << endl;
112        }
113
114        else if (aux != -1)
115        {
116          documento << "d(p, 1, X) = {(p, " << cadena << ")}" <<
     endl;
117          documento << "--Cadena Invalida--" << endl;
118
119          cout << "d(p, 1, X) = {(p, " << cadena << ")}" << endl;
```

```
120           cout << "--Cadena Invalida--" << endl;
121         }
122
123         else
124         {
125           documento << "--Cadena Invalida--" << endl;
126
127           cout << "--Cadena Invalida--" << endl;
128         }
129       }
130     }
131
132     //Cadena en espera de evaluacion
133     else
134     {
135       if (estado == 'q')
136       {
137         documento << "d(q, 0, X) = {(q, " << cadena << ")}" << endl
      ;
138
139         cout << "d(q, 0, X) = {(q, " << cadena << ")}" << endl;
140       }
141
142       else
143       {
144         documento << "d(p, 1, X) = {(p, " << cadena << ")}" << endl
      ;
145
146         cout << "d(p, 1, X) = {(p, " << cadena << ")}" << endl;
147       }
148     }
149   }
150
151   documento.close();
152 }
153
154 void proceso(string cadena)
155 {
156   struct Pila * automata;
157   automata=NULL;
158
159   string cadena_aux;
160   cadena_aux = "e";
161
162   int aux;
163
164   for (int i = 0; i < cadena.length(); ++i)
165   {
166     //Filtro para cadenas del tipo: 01010101, 011111
167     if(i > 0 && automata == NULL)
168     {
169       if (cadena[i] == '1')
170       {
171         descripciones('p', 'r', cadena_aux, i);
172       }
173
174       else
175       {
176         descripciones('q', 'r', cadena_aux, i);
177       }
178       return;
179     }
```

```
180
181     else if (cadena[i] == '0')
182     {
183       automata = Insertar(automata, 'X');
184
185       if (cadena_aux == "e")
186       {
187         cadena_aux = 'X';
188       }
189
190       else
191       {
192         cadena_aux = cadena_aux + 'X';
193       }
194       descripciones('q', 'e', cadena_aux, i);
195     }
196
197     //Filtro para cadenas del tipo: 1010101, 1100000
198     else if (cadena[i] == '1' && automata != NULL)
199     {
200       automata = Eliminar_Tope(automata);
201
202       aux = cadena_aux.length()-1;
203       cadena_aux = "e";
204       for (int j=0; j < aux; j++)
205       {
206         if (cadena_aux == "e")
207         {
208           cadena_aux[j] = 'X';
209         }
210
211         else
212         {
213           cadena_aux = cadena_aux + "X";
214         }
215       }
216       descripciones('p', 'e', cadena_aux, i);
217     }
218
219     else
220     {
221       descripciones('p', 'r', cadena_aux, i);
222       return;
223     }
224   }
225
226   //Filtro para cadenas del tipo: 00001
227   if (automata != NULL)
228   {
229     descripciones('p', 'r', cadena_aux, -1);
230     return;
231   }
232
233   //Acepta Cadena
234   descripciones('p', 'a', cadena_aux, 2);
235
236 }
237
238 void gotoxy (int x, int y)
239 {
240   HANDLE hcon = GetStdHandle(STD_OUTPUT_HANDLE);
241   COORD Pos;
```

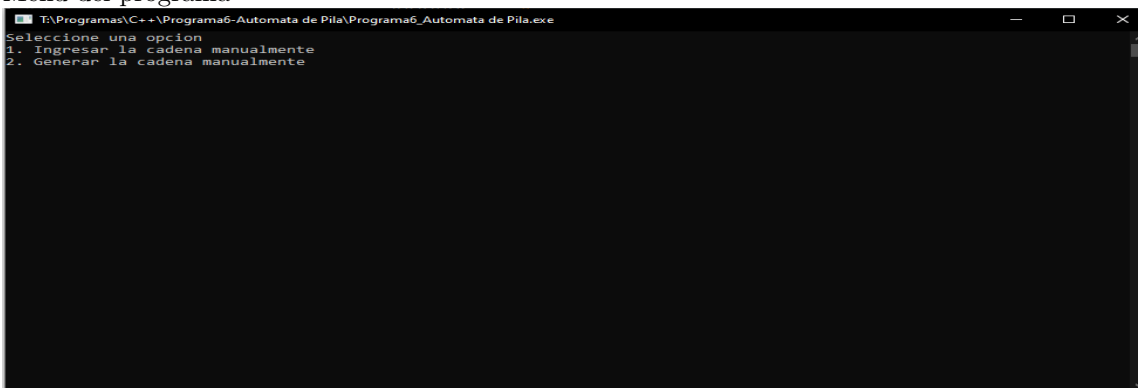
```
242   Pos.X = x;
243   Pos.Y = y;
244   SetConsoleCursorPosition(hcon, Pos);
245 }
246
247 void color(int x)
248 {
249   WORD n;
250   n=x;
251   SetConsoleTextAttribute(GetStdHandle(STD_OUTPUT_HANDLE),n);
252 }
253
254
255 void animacion(string cadena)
256 {
257   int posiciony = 20;
258   //auxiliar para guardar posicion de borrar
259   int aux;
260
261   system("cls");
262
263   cout << "Cadena: " << cadena << endl;
264
265   for (int i = 0; i< cadena.length(); i++)
266   {
267     Sleep(2000);
268
269     if (i< 10 && cadena[i] == '0')
270     {
271       color(i+1);
272       gotoxy(10, posiciony);
273       cout << "0";
274       aux = posiciony;
275       posiciony = posiciony-2;
276
277     }
278
279     else
280     {
281       color(0);
282       gotoxy(10, aux);
283       cout << "0";
284       aux = aux +2;
285     }
286   }
287
288   color(7);
289   gotoxy(0, 24);
290
291   cout << "Fin de animacion";
292   Sleep(2000);
293   system("cls");
294 }
295
296 int main (void)
297 {
298   bool ciclo = true;
299   srand (time(NULL));
300   char repetir;
301   int cantidad_de_datos;
302   char anim;
303
```

```
304   while (ciclo == true)
305   {
306     int opcion;
307     string cadena;
308     cadena = "e";
309
310     cout << "Seleccione una opcion" << endl;
311     cout << "1. Ingresar la cadena manualmente" << endl;
312     cout << "2. Generar la cadena manualmente" << endl;
313     cin >> opcion;
314
315     if (opcion == 1)
316     {
317       cout << "Favor de ingresar la cadena. EJ: 00001111" << endl;
318       cin >> cadena;
319       proceso(cadena);
320
321       cout << "Desea ver la animacion?: (y)si, (cualquier tecla)no"
       << endl;
322       cin >> anim;
323       if(anim == 'y' || anim == 'Y')
324       {
325         animacion(cadena);
326       }
327     }
328
329     else if (opcion == 2)
330     {
331       cout << "Generando cadena aleatoria" << endl;
332       Sleep (2000);
333
334       cantidad_de_datos = rand() % 10 + 1;
335
336       for(int i=0; i<cantidad_de_datos*2; i++)
337       {
338         if (i < cantidad_de_datos)
339         {
340           if (cadena == "e")
341           {
342             cadena = "0";
343           }
344
345           else
346           {
347             cadena = cadena + "0";
348           }
349         }
350
351         else
352         {
353           cadena = cadena + "1";
354         }
355       }
356
357       cout << "Usando la siguiente cadena: " << cadena << endl;
358       proceso(cadena);
359
360       cout << "Desea ver la animacion?: (y)si, (cualquier tecla)no"
       << endl;
361       cin >> anim;
362       if(anim == 'y' || anim == 'Y')
363       {
```

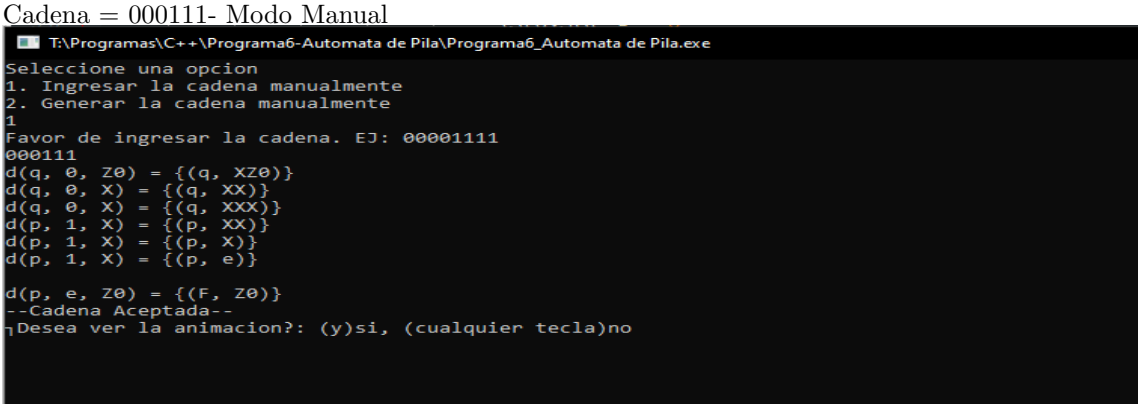
```
364         animacion(cadena);
365       }
366     }
367
368     else
369     {
370       cout << "Favor de ingresar una opcion valida" << endl;
371     }
372
373     cout << "Desea volver a correr el programa (y)si, (cualquier
        tecla)no?" << endl;
374     cin >> repetir;
375
376     if (repetir != 'y' && repetir != 'Y')
377     {
378       ciclo = false;
379     }
380     system("cls");
381
382   }
383
384 return 0;
385 }
```

FUNCIONAMIENTO:

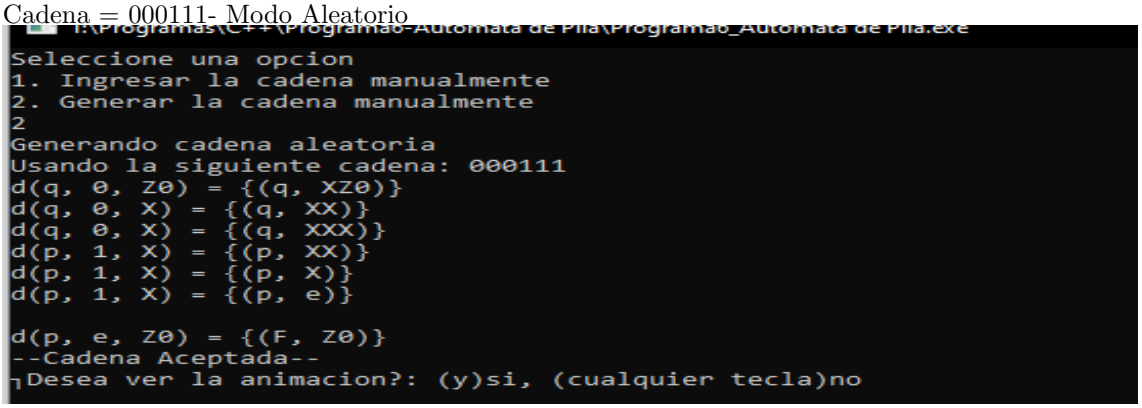
Menu del programa



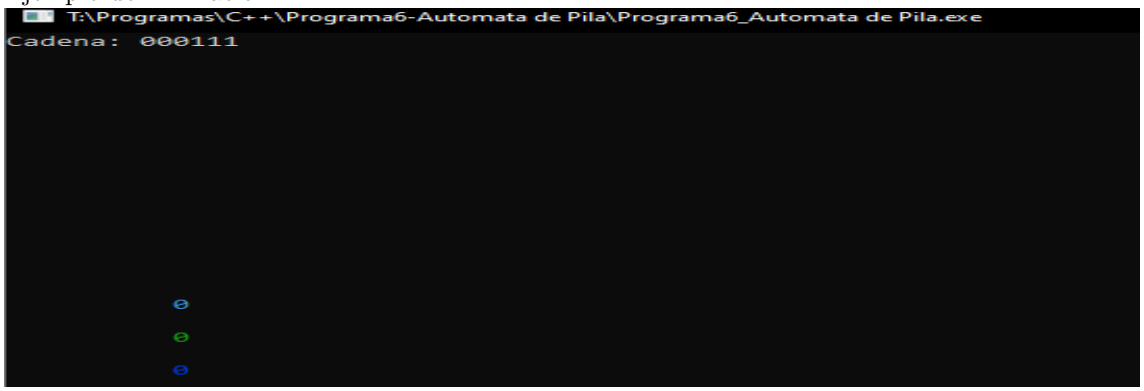
Cadena = 000111- Modo Manual



Cadena = 000111- Modo Aleatorio



Ejemplo de Animacion



Final