

## REPORTE

### Programa 2 - protocolo

Materia : Teoría de la Computación

Grupo : 4CM1

Alumno : Julio Cesar Hernández Reyes

Docente : Juárez Martínez Genaro

---

## 1. Introducción

En este reporte se explicara como se realizo el programa 2 - protocolo, el cual es un programa que crea 1,000,000 de cadenas binarias de 64 bits de largo, guardandolas en un archivo de texto, para luego pasar cada cadena por el AFD de imparidad de cadenas, el cual separa las cadenas totales en dos archivos .txt, el primero contiene todas las cadenas aceptadas por el autómata, el segundo contiene todas las cadenas rechazadas por el autómata. El programa empieza con un protocolo que pregunta si esta encendido o apagado, si esta prendido empieza el proceso de evaluacion de cadenas, si esta apagado el protocolo termina. Al principio del programa se tiene que tener un menu con la opción de ver el grafo del protocolo junto con el del AFD, asi como empezar el protocolo.

Resumen del programa:

Entrada: No se requiere ninguna entrada, el programa es automatico una vez empezado el protocolo

Salida: Tres archivos de texto:1->Todas las cadenas creadas, 2->Las cadenas aceptadas, 3->Las cadenas rechazadas

Otro detalle importante del programa es que el protocolo puede volver a entrar en el proceso de evaluación de las cadenas pues cada que se acaba una vuelta, el protocolo se pregunta otra vez si esta encendido; por lo tanto si se repite n veces, en los archivos de salida deben de ir las cadenas de las n vueltas que realizo el protocolo.

Para la realización del programa se utilizo el lenguaje C++, para implementar el programa se uso el IDE Dev-C++ para la codificación y ejecución del programa, el cual ya tiene compilador y ejecutador, en la misma interfaz.

Se uso un IDE en vez de compilar y ejecutar por consola, esto para una mayor facilidad a la hora de corregir errores y algunos detalles del programa.



Dev-C++  
Aplicación

## 2. Conceptos importantes

### 2.1. Autómata finito determinista

Un autómata finito determinista, que es aquel que sólo puede estar en un único estado después de leer cualquier secuencia de entradas. El término "Determinista" hace referencia al hecho de que para cada entrada sólo existe uno y sólo un estado al que el autómata puede hacer la transición a partir de su estado actual.

Un autómata finito determinista consta de:

1. Un conjunto finito de estados, a menudo designado como  $Q$
2. Un conjunto finito de símbolos de entrada, a menudo designado como  $\Sigma$
3. Una función de transición que toma como argumentos un estado y un símbolo de entrada y devuelve un estado.
4. Un estado inicial, uno de los estados de  $Q$
5. Un conjunto de estados finales o de aceptación  $F$ . El conjunto  $F$  es un subconjunto de  $Q$

### 2.2. Cómo procesa cadenas un AFD

El "lenguaje" del AFD es el conjunto de todas las cadenas que acepta. Supongamos que  $a_1, a_2, \dots, a_n$  es una secuencia de símbolos de entrada. Se comienza con el AFD en el estado inicial  $q_0$ . Se consulta la función de transición  $\delta$ , por ejemplo  $\delta(q_0, a_1) = q_1$  para hallar el estado al que pasará el AFD  $A$  después de procesar el primer símbolo de entrada  $a_1$ . A continuación se procesa el siguiente símbolo de entrada,  $a_2$ , evaluando  $\delta(q_1, a_2)$ ; supongamos que este estado es  $q_2$ . Continuamos aplicando el mismo procedimiento para hallar los estados  $q_3, q_4, \dots, q_n$  tal que  $\delta(q_{i-1}, a_i) = q_i$  para todo  $i$ . Si  $q_n$  pertenece a  $F$ , entonces la entrada  $a_1, a_2, \dots, a_n$  se acepta y, si no lo es se rechaza".

### 2.3. Diagrama de Transiciones

Un diagrama de transiciones de un AFD  $A = (Q, \Sigma, \delta, q_0, F)$  es un grafo definido como sigue:

1. Para cada estado de  $Q$ , existe un nodo.
2. Para cada estado  $q$  de  $Q$  y cada símbolo de entrada  $a$  de  $\Sigma$ , sea  $\delta(q, a) = p$ . Entonces, el diagrama de transiciones tiene un arco desde el nodo  $q$  hasta el nodo  $p$ , etiquetado como  $a$ ;
3. Existe una flecha dirigida al estado inicial  $q_0$ , etiquetada como Inicio. Esta flecha no tiene origen en ningún nodo.

4. Los nodos corresponden a los estados de aceptación (los que pertenecen a  $F$ ) están marcados con un doble círculo. Los estados que no pertenecen a  $F$  tienen un círculo simple.

## 2.4. Tabla de Transiciones

Una tabla de transiciones es una representación tabular convencional de una función, como ejemplo  $\delta$ , que toma dos argumentos y devuelve un valor. Las filas de la tabla corresponden a los estados y las columnas a las entradas. La entrada para la fila correspondiente al estado  $q$  y la columna correspondiente a la entrada  $a$  es el estado  $\delta(q, a)$ . El estado inicial se marca mediante una flecha y los estados de aceptación mediante un asterisco.

## 2.5. El lenguaje de un AFD

El lenguaje de un AFD  $A = (Q, \Sigma, \delta, q_0, F)$ . Este lenguaje se designa por  $L(A)$  y se define como:

$$L(A) = \{w \mid \delta(q_0, w) \text{ pertenece a } F\}$$

Es decir, el lenguaje de  $A$  es el conjunto de cadenas  $w$  que parten del estado inicial  $q_0$  y van hasta uno de los estados de aceptación. Si  $L$  es  $L(A)$  para un determinado AFD  $A$ , entonces decimos que  $L$  es un lenguaje regular.

## 3. Desarrollo

### 3.1. Código del programa

Declaración de librerías:

```
#include <iostream> // Para poder usar cin y cout
#include <stdlib.h> //Aquí están las funciones rand y srand
#include <time.h> // De aquí se usa la función time para obtener un
    indicador del tiempo actual del sistema para la semilla
#include <string.h> // Para trabajar con cadenas
#include <string>
#include <fstream> // Para poder trabajar con .txt
#include <thread> // Para dormir al programa
#include <cstdio> //para eliminar .txt

#include<graphics.h> //Para poder graficar
#include<conio.h> // Para poder usar el getch()
```

Declaración de namespace para poder usar cout y cin sin el std:: al principio

```
using namespace std;
```

Declaración de las constantes para el grafo, el tamaño de la ventana que se crea:

```
const int WIDTH = 1000;
const int HEIGHT = 800;
```

Declaración de namespace para literales de cronometro, que sirve para dormir al programa una determinada cantidad de tiempo:

```
using std::this_thread::sleep_for;
using namespace std::chrono_literals;
```

Declaración de las funciones del programa

```
void menu();
void grafo();
void protocolo();
int cincuentacincuenta();
string generarcadena();
void guardarNcadenas(int, int);
```

```

int automata(string);
void verificarcadenas(int);
void juntarcorridas(int);

```

Empieza el main() en donde solo eliminamos los archivos, si es que existen, de la ubicación en donde se encuentre el programa, y la llamada a la función menu() para desplegar el menú del programa:

```

int main() {
    remove("TodasLasCadenas.txt");
    remove("Aceptadas.txt");
    remove("Rechazadas.txt");

    menu();

    return 0;
}

```

Funcion menu() la cual despliega el menu al usuario con las opciones de:

- 1.-Mostrar el grafo del programa con el protocolo y el AFD
- 2.-Empezar el protocolo

```

void menu() {
    int opcion = 0;
    bool menu = false;
    do{
        system("cls");
        cout << "Programa que implementa un protocolo de
                seleccion de cadenas binarias de 64 bits\n" << endl;
        cout << "Opciones:"<< endl;
        cout << "1.- Ver el grafo del protocolo." << endl;
        cout << "2.- Iniciar el protocolo" << endl;
        cout << "3.- Salir." << endl;
        cin >> opcion;

        switch (opcion) {
            case 1 :
                system("cls");
                cout<< "Grafo del protocolo" << endl;
                grafo();

                break;
            case 2 :
                system("cls");

```

```

        protocolo();
        system("pause");
    break;
    case 3:  menu =  true; break;

    default:
        system("cls");
        cout << "Favor de ingresar un valor
            valido" << endl;
        system("pause");
    }
}while(menu == false);

}

```

Función del protocolo en donde se pregunta si esta encendido o no, gracias a un rand() que va de 0 a 1, si sale 0 el protocolo esta apagado, pero si sale 1 el protocolo esta encendido y empieza la creación de las cadenas y luego la evaluación con el AFD:

```

void protocolo(){
    bool fin = false;
    int contavuelta = 0;
    while( fin == false){
        cout << "El protocolo esta listo?"<<endl;
        if(cincuentacincuenta() == 1){
            contavuelta++;
            cout<< "Si" <<endl;
            cout << "Creando cadenas..." << endl;
            guardarNcadenas(1000000, contavuelta);

            cout << "Cadenas creadas..." << endl;
            //Se duerme por 1 segundos
            sleep_for(1000ms);

            cout << "Validando cadenas..." << endl;

            verificarcadenas(contavuelta);

            cout << "Vefificacion completa." <<endl;

            cout << endl;
        }
        else if(cincuentacincuenta() == 0){
            cout<< "No" <<endl;
            cout << "Fin del programa."<<endl;
        }
    }
}

```

```

        fin = true;
    }

    }
    //cout << contavuelta;
    if(contavuelta>=1){
        juntarcorridas(contavuelta);
    }
}

```

Función que regresa un numero ya sea el 0 o el 1, esta función se usa tanto en el protocolo como en la creación de las cadenas.

```

//Funcion que tiene el 50% de probabilidad de que salga 0 o 1
int cincuentacincuenta(){
    int numero;
    srand(time(NULL));
    numero = rand() % 2;
    return numero;
}

```

Función que crea las cadenas, usando la función de cincuentacincuenta() anterior, las cadenas creadas tienen largo de 64 bits de largo:

```

//Funcion para generar la cadena de 0 y 1 de 64 bits de largo
string generarcadena(){
    string cadena = "";
    int numero;
    int limite = 64;
    for(int conta = 1; conta <= limite; conta++){
        numero = rand() % 2;
        cadena = cadena + to_string(numero);
    }
    return cadena;
}

```

Función del autómata finito determinista de imparidad, al cual ingresa una cadena, de las creadas anteriormente, y va iterando por cada carácter de esta misma cadena hasta el final de ella, el retorno de esta función es un numero, 0 si la cadena fue rechazada, o 1 si la cadena fue aceptada.

```

//Funcion del automata de seleccion de imparidades
//regresa un string dependiendo si la cadena es aceptada o no
// aceptada => cadena aceptada

```

```

// rechazada => cadena rechazada
int automata(string cadena){
    int largo;
    largo = cadena.length();
    //declaracion de estados del automata
    int q0 = 0;
    int q1 = 1;
    int q2 = 2;
    int q3 = 3;
    int actual;
    actual = q0;

    bool fin = false;
    int conta = 0;
    //caracter
    char cr = ' ';
    while (fin == false){
        if(conta >= largo){
            fin == true;
            break;
        }
        //estado q0 no aceptada
        if (actual == q0) {
            if(cadena[conta] == '1'){
                actual = q1;
            }
            if(cadena[conta] == '0'){
                actual = q2;
            }
        }
        //estado q1 aceptada
        else if (actual == q1){
            if(cadena[conta] == '1'){
                actual = q0;
            }
            if(cadena[conta] == '0'){
                actual = q3;
            }
        }
        //estado q2 aceptada
        else if (actual == q2){
            if(cadena[conta] == '1'){
                actual = q3;
            }
            if(cadena[conta] == '0'){

```



```

        actual = q0;
    }
}
//estado q3 aceptada
else if (actual == q3){
    if(cadena[conta] == '1'){
        actual = q2;
    }
    if(cadena[conta] == '0'){
        actual = q1;
    }
}
conta++;
}
int estado ;

if(actual == 0){
    //rechazada
    estado = 0;
}
else if( actual == 1 || actual == 2 || actual == 3){
    //aceptada
    estado = 1;
}
return estado;
}

```

Función que guarda el millón de cadenas solicitadas anteriormente, esta función ocupa la función antes mencionadas generarcadena() la cual regresa una cadena binaria de 64 bits de largo, la cual se guarda en el archivo "TodasLasCadenas.txt". Como un requisito del programa era que si el protocolo entraba varias veces a realizar el proceso de verificación de cadenas, se crean n archivos de todas las cadenas.txt para poder evaluar cada vuelta a parte, estos archivos al final se juntan en uno solo, y se eliminan todos los archivos antes creados.

```

//Funcion para guardar las n cadenas en un .txt
void guardarNcadenas(int limite, int contavuelta){

    string nombreakchivo;
    nombreakchivo = "TodasLasCadenas"+ to_string(contavuelta)+ ".
        txt";

    ofstream todascadenas(nombreakchivo);
    string cadenagenerada;
    if(todascadenas.is_open()){

```

```

        for(int conta=0; conta < limite; conta++){
            cadenagenerada = generarcadena();
            todascadenas << cadenagenerada << endl;
        }

    }
    todascadenas.close();
}

```

Función que hace el guardado de las cadenas en los archivos correspondientes, dependiendo si es una cadena aceptada se guarda en un archivo ".Aceptadas.txt", pero si es rechazada se guarda en un archivo Rechazadas.txt", de igual forma con el archivo de todas las cadenas, esta validación se hace las n veces que el protocolo haya iniciado.

```

void verificarcadenas(int contavuelta){

    string nombreamchivo;
    nombreamchivo = "TodasLasCadenas"+ to_string(contavuelta)+ ".
        txt";
    string archivomceptadas;
    archivomceptadas = "Aceptadas"+ to_string(contavuelta)+ ".txt";
    string archivomrechazadas;
    archivomrechazadas = "Rechazadas"+ to_string(contavuelta)+ ".txt
        ";
    string linea;
    ifstream Cadenas(nombreamchivo);

    ofstream mceptadas(archivomceptadas);
    ofstream mrechazadas(archivomrechazadas);
    if(Cadenas.is_open()){

        while(getline(Cadenas,linea)){

            //Si es aceptada se guarda en Aceptadas.txt
            if(automata(linea) == 1){
                if(mceptadas.is_open()){
                    mceptadas << linea << endl;
                }
            }
            if(automata(linea) == 0){

                if(mrechazadas.is_open()){
                    mrechazadas << linea << endl;
                }
            }
        }
    }
}

```

```

        }

    }

    }

    }
    rechazadas.close();
    aceptadas.close();
    Cadenas.close();
}

```

Esta es la función que junta todos los archivos de texto creados del mismo tipo en uno solo, copiando el contenido de los n archivos de cada tipo en un solo archivo, el cual sera el final. Por lo que si el protocolo inicio n veces, los n archivos creados de cada tipo solo se queden los 3 finales: "TodasLasCadenas.txt", "Aceptadas.txt", "Rechazadas.txt".

```

void juntarcorridas(int contavuelta){

    string linea;

    for(int i = 1; i <= contavuelta; i++){

        string nombreamchivo = "TodasLasCadenas"+
            to_string(i)+ ".txt";
        ifstream cadenas(nombreamchivo);
        ofstream cadenasfinal("TodasLasCadenas.txt", ios
            ::app);
        if(cadenas.is_open()){
            if(cadenasfinal.is_open()){
                cadenasfinal << "Vuelta: "<< i
                    << endl;
                while(getline(cadenas, linea)){
                    cadenasfinal << linea
                        << endl;
                }
                cadenasfinal << endl;
            }
            cadenasfinal.close();
        }
        cadenas.close();
        char * archivotexto1 = new char[nombreamchivo.
            length() + 1];
    }
}

```

```

strcpy(archivotexto1, nombearchivo.c_str());

remove(archivotexto1);

string archivoaceptadas = "Aceptadas"+
    to_string(i)+ ".txt";
ifstream aceptadas(archivoaceptadas);
fstream aceptadasfinal("Aceptadas.txt",ios::app
    );
if(aceptadas.is_open()){
    if(aceptadasfinal.is_open()){
        aceptadasfinal << "Vuelta: "<<
            i << endl;
        while(getline(aceptadas, linea))
        {
            aceptadasfinal << linea
                << endl;
        }
        aceptadasfinal << endl;
    }
    aceptadasfinal.close();
}
aceptadas.close();
char * archivotexto2 = new char[
    archivoaceptadas.length() + 1];
strcpy(archivotexto2, archivoaceptadas.c_str())
    ;
remove(archivotexto2);

string archivorechazadas = "Rechazadas"+
    to_string(i)+ ".txt";
ifstream rechazadas(archivorechazadas);
fstream rechazadasfinal("Rechazadas.txt",ios::
    app);
if(rechazadas.is_open()){
    if(rechazadasfinal.is_open()){
        rechazadasfinal << "Vuelta: "<<
            i << endl;
        while(getline(rechazadas, linea)
            ){
            rechazadasfinal <<
                linea << endl;
        }
        rechazadasfinal << endl;
    }
}

```

```

        }
        rechazadasfinal.close();
    }
    rechazadas.close();
    char * archivotexto3 = new char[
        archivorechazadas.length() + 1];
    strcpy(archivotexto3, archivorechazadas.c_str()
    );
    remove(archivotexto3);
}

}

```

Función que muestra el grafo del protocolo con el del autómata finito determinista, el cual por usar c++ tuve que utilizar la librería graphics.h la cual permite hacer gráficos en ventanas con muchas posibilidades, pero la creación es solo con coordenadas entonces la creación del grafo tomo bastante tiempo, pero al final quedo de una forma aceptable.

```

void grafo() {
    initwindow(WIDTH, HEIGHT, "Grafo de protocolo");
    setbkcolor(WHITE);
    cleardevice();
    setcolor(BLACK);
    setlinestyle(SOLID_LINE, 0, 3);

    setfillstyle(SOLID_FILL, BLACK);
    //Lineas
    //q0--q2
    line(100, 150, 100, 630);
    circle(100, 640, 8);
    floodfill(102, 640, BLACK);
    //q0--q1
    line(160, 110, 850, 110);
    circle(158, 110, 8);
    floodfill(158, 110, BLACK);
    //q1--q3
    line(900, 170, 900, 650);
    circle(900, 160, 8);
    floodfill(900, 160, BLACK);
    //q3--q2
    line(150, 690, 835, 690);
    circle(843, 690, 8);
    floodfill(843, 690, BLACK);

    settextstyle(8, 0, 6);
}

```

```

//Q
circle(500,400,50);
outtextxy(480,370,"Q");
//->q
setfillstyle(SOLID_FILL,BLACK);
circle(500,460,8);
floodfill(500,460,BLACK);
//Inicio--Q
line(500,470,500,500);
//Inicio
settextstyle(8, 0, 4);
outtextxy(450,505,"Inicio");

// 0
outtextxy(520,310,"0");

//OFF
circle(500,250,50);
outtextxy(470,235,"OFF");
//->OFF
setfillstyle(SOLID_FILL,BLACK);
circle(500,310,8);
floodfill(500,310,BLACK);
//Q--OFF
line(500,310,500,350);

settextstyle(8, 0, 5);
//q0
circle(100,100,50);
outtextxy(75,80,"q0");
//q1
circle(900,100,40);
circle(900,100,50);
outtextxy(875,80,"q1");
//q2
circle(100,700,40);
circle(100,700,50);
outtextxy(75,680,"q2");
//q3
circle(900,700,40);
circle(900,700,50);
outtextxy(875,680,"q3");

//q -- q0

```

```

line(145,145,462,362);
//->q0
setfillstyle(SOLID_FILL,BLACK);
circle(142,140,8);
floodfill(143,142,BLACK);

settextstyle(8, 0, 4);
//1
outtextxy(350,250,"1");

//arco q0 -> q1
arc(500,800,65,116,790);
circle(842,90,8);
floodfill(844,92,BLACK);

//arco q2 -> q3
arc(500,0,245,296,790);
circle(158,711,8);
floodfill(158,711,BLACK);

//arco q0 -> q2
arc(800,400,161,200,790);
circle(60,140,8);
floodfill(60,140,BLACK);

//arco q1 -> q3
arc(200,400,341,20,790);
circle(945,660,8);
floodfill(945,660,BLACK);

// 0 y 1 externos
settextstyle(8, 0, 4);
outtextxy(150,40,"1");

outtextxy(970,150,"0");

outtextxy(800,750,"1");

outtextxy(20,640,"1");

// 0 y 1 internos
outtextxy(70,200,"0");

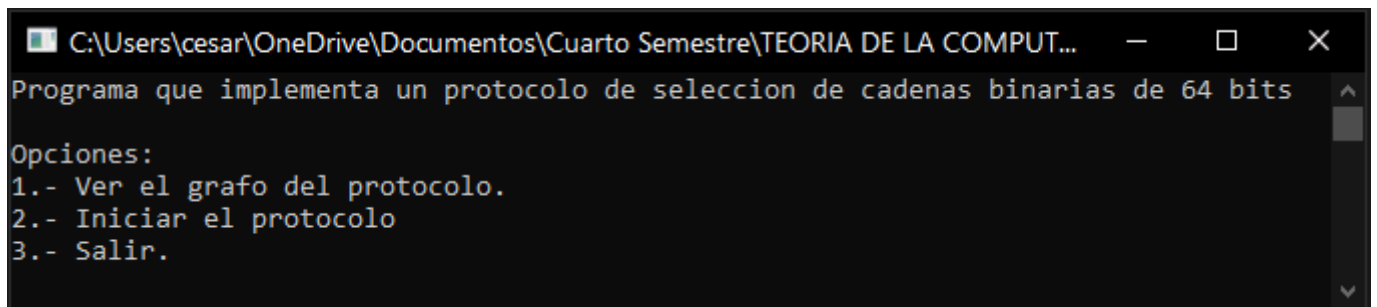
outtextxy(300,700,"1");

```

```
    outtextxy(920, 550, "0");  
  
    outtextxy(700, 70, "1");  
  
    system("pause");  
    closegraph();  
}
```

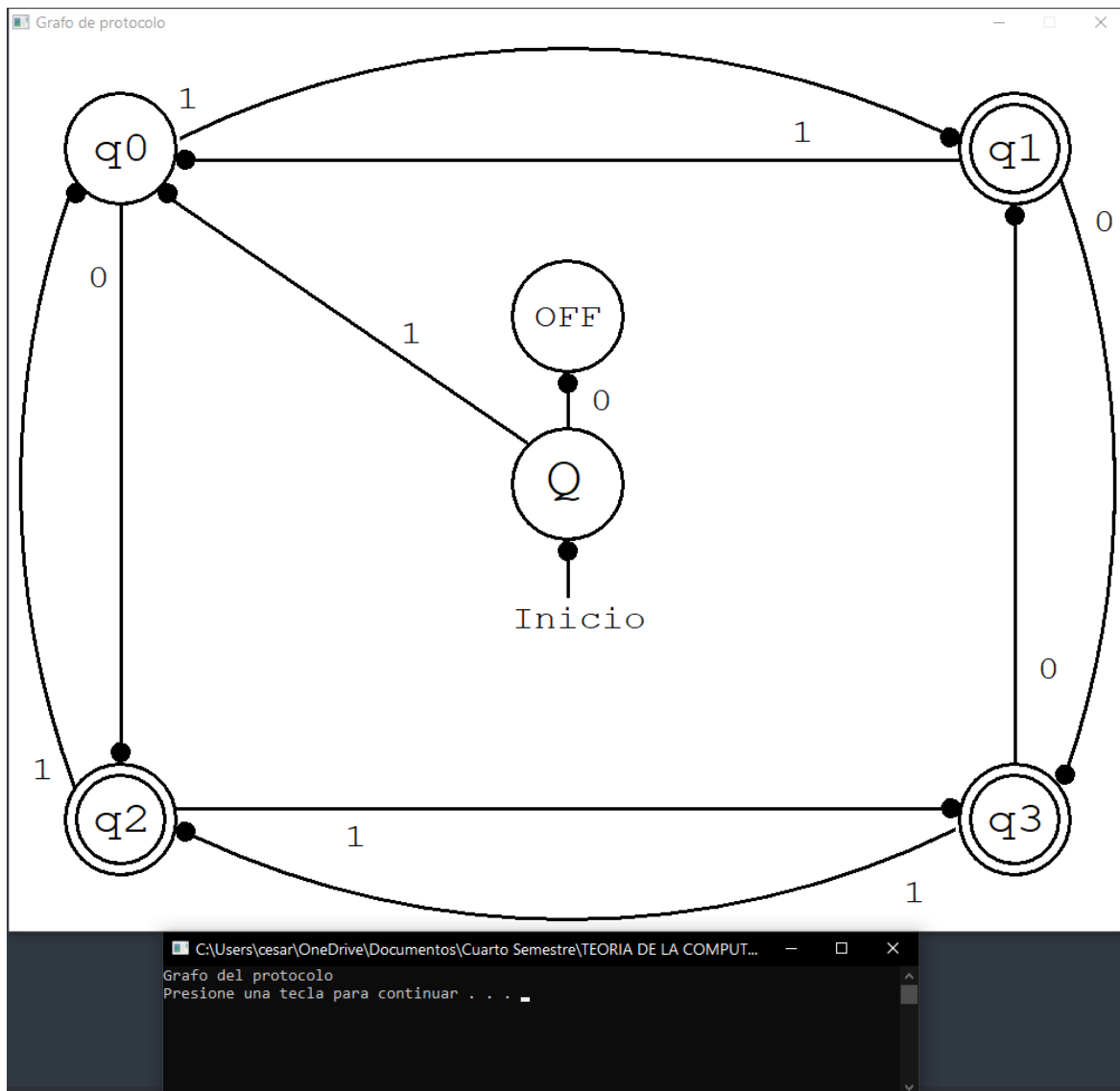
### 3.2. Ejecución del programa, en Dev-C++

Primera parte, el menú: En donde el usuario elige ver el grafo o iniciar el protocolo.



Cuando se elige la opción 1 se muestra el grafo del protocolo junto con el del autómata finito determinista, el cual fue echo con código como se pidió, en vez de una imagen:



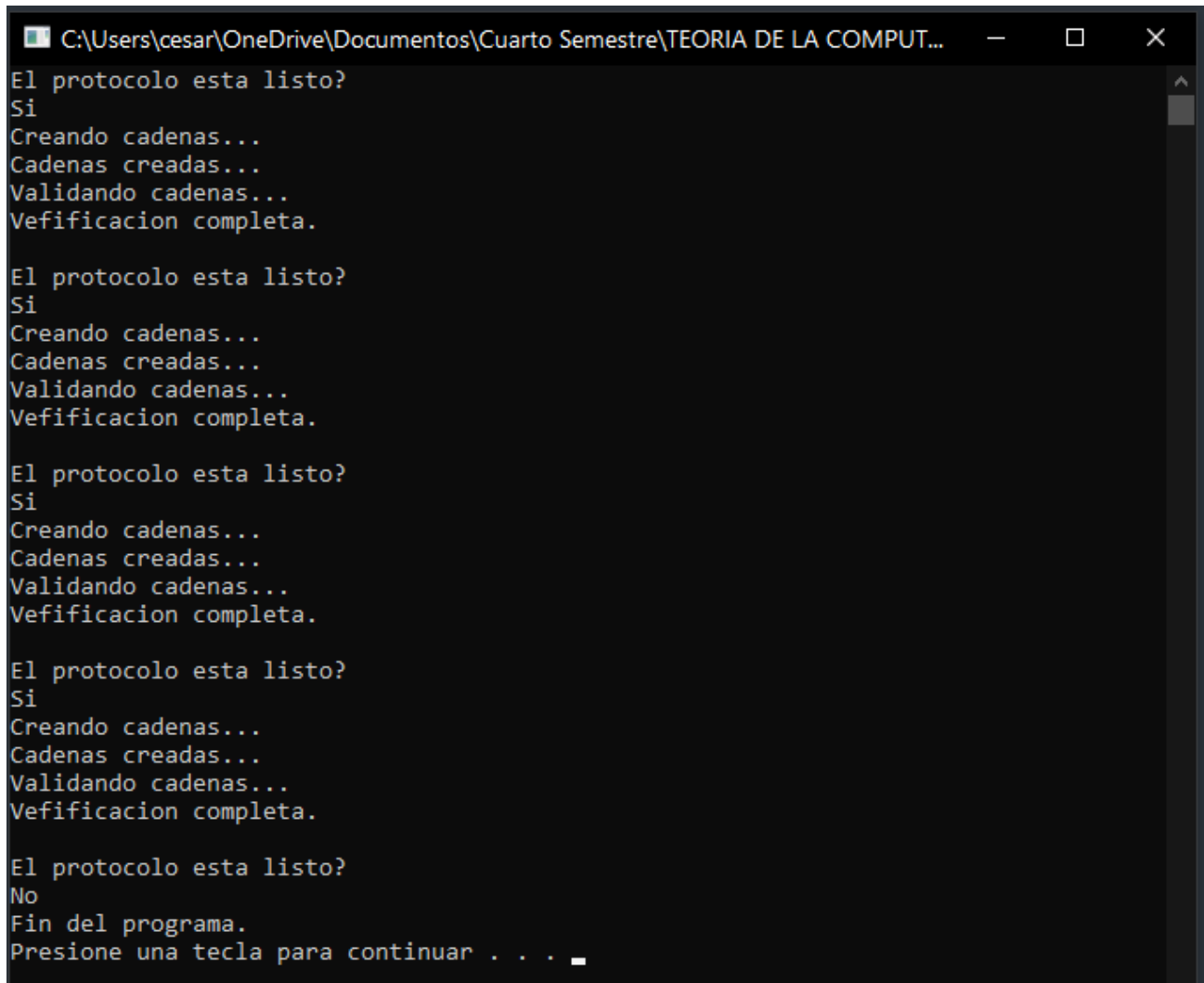


Ahora cuando se da enter después de ver el grafo y se elige la segunda opción, el protocolo empieza, preguntándose si esta encendido o no:

```
C:\Users\cesar\OneDrive\Documentos\Cuarto Semestre\TEORIA DE LA COMPUT...
El protocolo esta listo?
No
Fin del programa.
Presione una tecla para continuar . . .
```

Imagen cuando el protocolo esta apagado

La otra posibilidad es que el protocolo este encendido y entonces empieza el proceso de la creación y validación de las cadenas:



```
C:\Users\cesar\OneDrive\Documentos\Cuarto Semestre\TEORIA DE LA COMPUT...
El protocolo esta listo?
Si
Creando cadenas...
Cadenas creadas...
Validando cadenas...
Vefificacion completa.

El protocolo esta listo?
Si
Creando cadenas...
Cadenas creadas...
Validando cadenas...
Vefificacion completa.

El protocolo esta listo?
Si
Creando cadenas...
Cadenas creadas...
Validando cadenas...
Vefificacion completa.

El protocolo esta listo?
Si
Creando cadenas...
Cadenas creadas...
Validando cadenas...
Vefificacion completa.

El protocolo esta listo?
No
Fin del programa.
Presione una tecla para continuar . . . _
```

Imagen cuando después de que el protocolo hiciera 4 vueltas

Como en este caso el protocolo entro 4 veces, la siguiente imagen tomada del archivo "TodasLas-Cadenas.txt" demuestra que se crearon 4 archivos de cada tipo pero que al final todos se juntaron en uno solo

```
TodasLasCadenas.txt
3999996 0011011100100001111101110010011100111100001010001100011100000100
3999997 0011001111111011011011100111111000010111110011000000010000010101
3999998 111100111110101111000010100110111101101111111110101011011010110
3999999 010101001100101101010011011101110110011001100111001111110111110
4000000 000000000100110011011111110111001100101010001000000000100111110
4000001 1101011100011001100011110111100001011111010000110001010010100000
4000002 010001001111111011011101101000010011110111111100000101011101100
4000003 0101100000101110110101000011001101000010010011001100010101111001
4000004 1111001001011111100000101110001000000110100110000111010101110100
4000005 1101101010110101000011011011110101101000001011001001110101101101
4000006 1111011000000100001000010101101011101000011011101110110011100101
4000007 1001001010000100001101100111011001101011101000110100110010001100
```

Imagen que muestra que el archivo llego hasta las 4 millones de lineas, osea tiene 4 millones de cadenas binarias de 64 bits de largo.

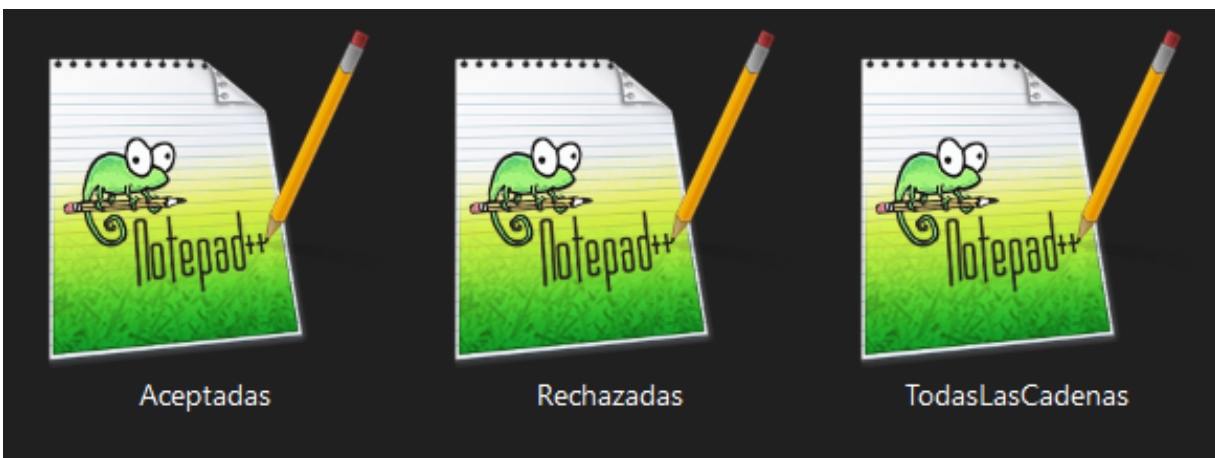


Imagen de la creación de los tres archivos finales.

```
C:\Users\cesar\OneDrive\Documentos\Cuarto Semestre\TEORIA DE LA COMPUT...
Programa que implementa un protocolo de seleccion de cadenas binarias de 64 bits

Opciones:
1.- Ver el grafo del protocolo.
2.- Iniciar el protocolo
3.- Salir.
3

-----
Process exited after 427.8 seconds with return value 0
Presione una tecla para continuar . . .
```

Imagen del menú al elegir la opción 3 y finalizar el programa.

## 4. Conclusiones

El programa del protocolo fue muy largo y laborioso, en especial la parte de hacer el grafo con la librería `graphics.h` pues todo se tenia que hacer con coordenadas, lo cual hace que el proceso se extienda mucho si quieres que se vea bien el grafo. En la parte de la gestión de archivos fue la que mas me gusto hacer porque aprendí como hacer  $n$  archivos para después juntarlos todos en uno solo, borrando los demás, haciendo que todo quede mejor ordenado, aunque el código quedo un poco extenso para solo 3 tipos de archivos. Espero poder optimizar mi código mejor para los próximos programas y así no tardarme tanto en cosas tan sencillas como el manejo de archivos.