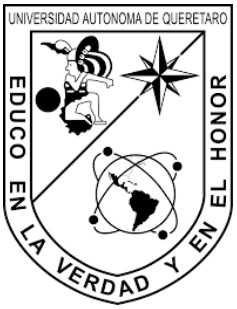
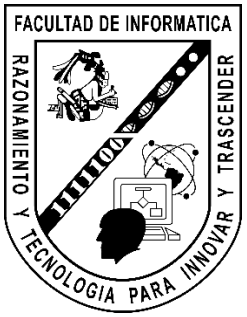


Universidad Autónoma de Querétaro

Facultad de Informática



Teoría de la Computación

Carrera:
Ingeniería de Software.

Alumno: Cano Cabrera David Emmanuel

Profesor: M. en C. Fidel González Gutiérrez

Expediente:

297140

Segunda Evaluación Parcial Práctica

29 de noviembre de 2022

```cpp
// autor: Cano Cabrera, David Emmanuel
// fecha: 2017-10-10

#include <iostream>
#include <fstream>
#include <string>

using namespace std;


// funcion que determina si un estado es un estado del AFD
bool esEstado(string estado, string estados[])
{
    for (int i = 0; i < 4; i++)
    {
        if (estado == estados[i])
        {
            return true;
        }
    }
    return false;
}

// funcion que determina si un estado es un estado final del AFD
bool esEstadoFinal(string estado, string estadosFinales[])
{
    for (int i = 0; i < 2; i++)
    {
        if (estado == estadosFinales[i])
        {
            return true;
        }
    }
    return false;
}

bool estadoSiguiente(string estadoActual, string cadena, string estados[], string alfabeto[], string
estadosFinales[], string tabla[][3])
{

    if (esEstadoFinal(estadoActual, estadosFinales) && cadena.length() == 0)
    {
        return true;
        cout << "Cadena aceptada" << endl;
    }

    // para cada simbolo de la cadena
    for (int i = 0; i < 4; i++)
```

```
    {
        // si el estado actual es igual al estado de la tabla
        if (estadoActual == estados[i])
        {

            // para cada simbolo del alfabeto
            for (int j = 0; j < 3; j++)
            {
                // si el simbolo de la cadena es igual al simbolo de la tabla
                if (cadena[0] == alfabeto[j][0])
                {
                    // si el estado siguiente es un estado del AFD
                    if (esEstado(tabla[i][j], estados))
                    {
                       // estructura de la tabla de transiciones
                        cout << "    " << estadoActual << "                "
                             << cadena[0] << "               " << tabla[i][j + 1] << endl;

                        cout << endl;
                        // si la cadena es aceptada
                        if (estadoSiguiente(tabla[i][j + 1], cadena.substr(1),
                            estados, alfabeto, estadosFinales, tabla))
                        {
                            return true;
                        }
                    }
                }
            }
        }
    }

    return false;
}

int main()
{
    string estados[4];
    string alfabeto[2];
    string estadoInicial;
    string estadosFinales[2];
    string estadosSiguientes[4][3];
    string cadena;
    string estadoActual;

    bool aceptada = false;

    // lectura y muestra de la 5-tupla del AFD
    ifstream archivo;
```

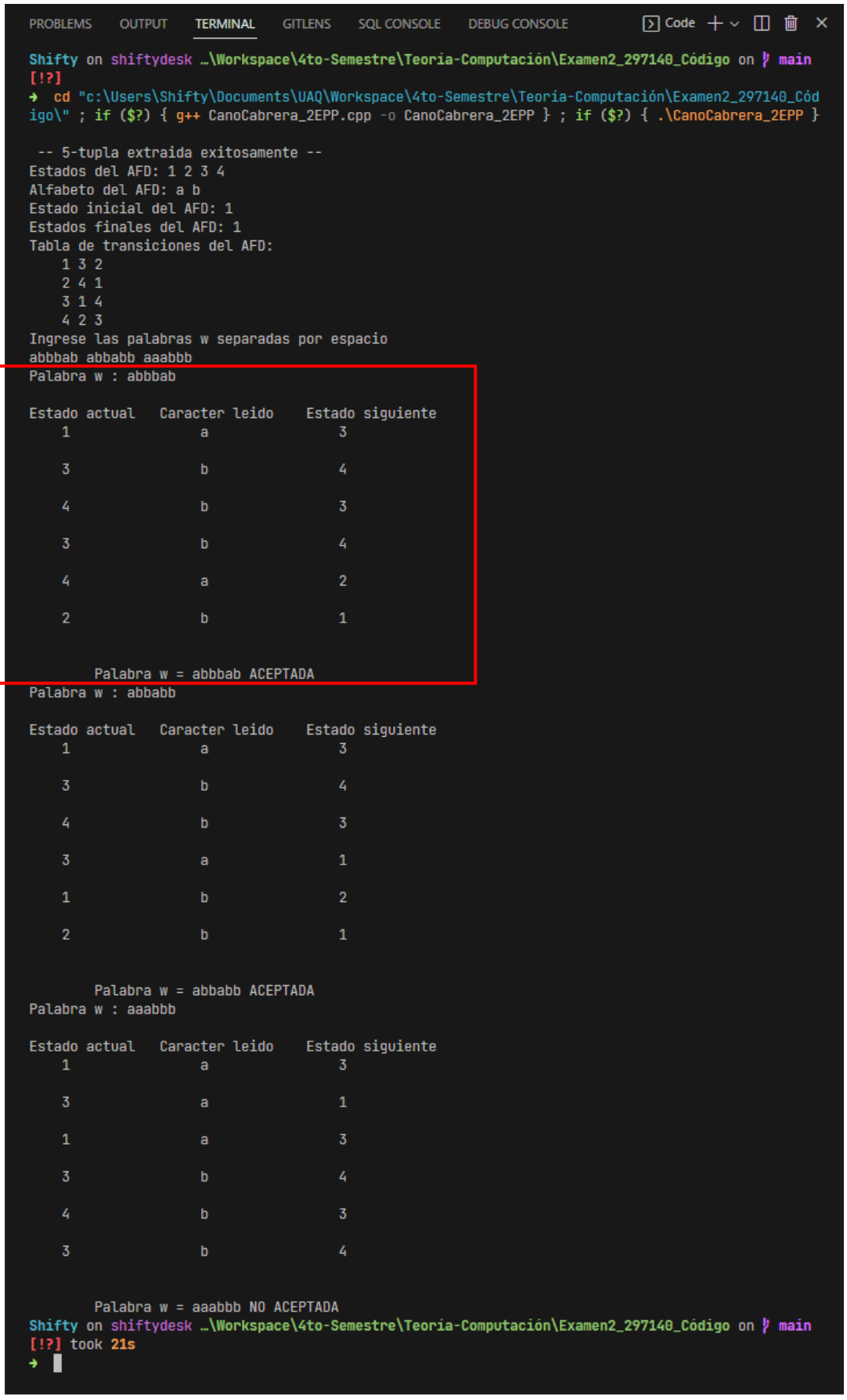
```
archivo.open("AFD.txt");
if (archivo.is_open())
{
    cout << " -- 5-tupla extraida exitosamente --" << endl;
    // lee los estados del AFD
    for (int i = 0; i < 4; i++)
    {
        archivo >> estados[i];
    }
    // imprime los estados del AFD
    cout << "Estados del AFD: ";
    for (int i = 0; i < 4; i++)
    {
        cout << estados[i] << " ";
    }

    // lee el alfabeto del AFD
    for (int i = 0; i < 2; i++)
    {
        archivo >> alfabeto[i];
    }

    // imprime el alfabeto del AFD
    cout << endl
         << "Alfabeto del AFD: ";
    for (int i = 0; i < 2; i++)
    {
        cout << alfabeto[i] << " ";
    }

    // lee el estado inicial del AFD
    archivo >> estadoInicial;

    // imprime el estado inicial del AFD
    cout << endl
         << "Estado inicial del AFD: " << estadoInicial << endl;

    // lee los estados finales del AFD
    for (int i = 0; i < 1; i++)
    {
        archivo >> estadosFinales[i];
    }

    // imprime los estados finales del AFD
    cout << "Estados finales del AFD: ";
    for (int i = 0; i < 2; i++)
    {
        cout << estadosFinales[i] << " ";
```

```
    }

    // lee la tabla de transiciones del AFD
    for (int i = 0; i < 4; i++)
    {
        for (int j = 0; j < 3; j++)
        {
            archivo >> estadosSiguientes[i][j];
        }
    }

    // imprime la tabla de transiciones del AFD
    cout << endl
         << "Tabla de transiciones del AFD: " << endl;
    for (int i = 0; i < 4; i++)
    {
        cout << "    ";
        for (int j = 0; j < 3; j++)
        {
            cout << estadosSiguientes[i][j] << " ";
        }
        cout << endl;
    }

    archivo.close();
}

// repetir mientras la cadena no sea vacia
do
{

    cout << "Ingrese las palabras w separadas por espacio" << endl;
    // leer el conjunto de palabras
    getline(cin, cadena);

    // dividir la cadena en palabras y guardarlas en un arreglo
    string palabras[100];
    int i = 0;
    string palabra = "";
    while (cadena.length() > 0)
    {
        // si el caracter es un espacio
        if (cadena[0] == ' ')
        {
            palabras[i] = palabra;
            palabra = "";
            i++;
        }
```

```
            else
            {
                palabra += cadena[0];
            }
            cadena = cadena.substr(1);
        }

        // para cada palabra
        for (int j = 0; j < i; j++)
        {
            estadoActual = estadoInicial;

            cout << "Palabra w : " << palabras[j] << endl;

            cout << "\nEstado actual   Caracter leido    Estado siguiente" << endl;
            // funcion recursiva que determina si la cadena es aceptada o no por el AFD
            aceptada = estadoSiguiente(estadoActual, palabras[j], estados,
                       alfabeto, estadosFinales, estadosSiguientes);

            if (aceptada)
            {
                cout << "\n        Palabra w = " << palabras[j] << " ACEPTADA" << endl;
            }
            else
            {
                cout << "\n        Palabra w = " << palabras[j] << " NO ACEPTADA" << endl;
            }
        }

    } while (cadena.length() > 0);

    return 0;
}
```

Como evidencia de funcionamiento se incluye el ejemplo con la **palabra w = abbbab** la cual da como resultado un estado de **ACEPTACION**. Al mismo tiempo se incluye en el input del programa un conjunto de palabras que son ACEPTADAS y NO ACEPTADAS por el autómata respectivamente.

NOTA -> El documento "AFD.txt" deberá encontrarse ubicado en el mismo directorio que el archivo .cpp para poder ser leído.


PROBLEMS
OUTPUT
TERMINAL
GITLENS
SQL CONSOLE
DEBUG CONSOLE
Code
Shifty on shiftydesk …\Workspace\4to-Semestre\Teoria-Computación\Examen2_297140_Código on main
[!?]
cd "c:\Users\Shifty\Documents\UAQ\Workspace\4to-Semestre\Teoría-Computación\Examen2_297140_Código\" ; if ($?) { g++ CanoCabrera_2EPP.cpp -o CanoCabrera_2EPP } ; if ($?) { .\CanoCabrera_2EPP }
-- 5-tupla extraida exitosamente --
Estados del AFD: 1 2 3 4
Alfabeto del AFD: a b
Estado inicial del AFD: 1
Estados finales del AFD: 1
Tabla de transiciones del AFD:
1 3 2
2 4 1
3 1 4
4 2 3
Ingrese las palabras w separadas por espacio
abbbab abbabb aaabbb
Palabra w : abbbab
Estado actual Caracter leido Estado siguiente
1 a 3
3 b 4
4 b 3
3 b 4
4 a 2
2 b 1
Palabra w = abbbab ACEPTADA
Palabra w : abbabb
Estado actual Caracter leido Estado siguiente
1 a 3
3 b 4
4 b 3
3 a 1
1 b 2
2 b 1
Palabra w = abbabb ACEPTADA
Palabra w : aaabbb
Estado actual Caracter leido Estado siguiente
1 a 3
3 a 1
1 a 3
3 b 4
4 b 3
3 b 4
Palabra w = aaabbb NO ACEPTADA
Shifty on shiftydesk …\Workspace\4to-Semestre\Teoria-Computación\Examen2_297140_Código on main
[!?] took 21s