ACTIVIDAD 04: **APLICACIÓN DE PILA Y COLA**

ESTUDIANTE: **EFRAIN ROBLES PULIDO**

CODIGO: **221350095**

NOMBRE DE LA MATERIA: **ESTRUCTURAS DE DATOS I**

SECCIÓN: ***D12***

CLAVE: ***I5886***

FECHA: **DOMINGO 13 DE FEBRERO DE 2022**

Tabla de autoevaluación:

Autoevaluación			
Concepto	**Sí**	**No**	**Acumulación**
Bajé el trabajo de internet o alguien me lo pasó (aunque sea de forma parcial)	-100 pts	0 pts	0 pts
Incluí el código fuente **en formato de texto** **(sólo si funciona cumpliento todos los requerimientos)**	+25 pts	0 pts	25 pts
Incluí las **impresiones de pantalla** **(sólo si funciona cumpliento todos los requerimientos)**	+25 pts	0 pts	25 pts
Incluí una **portada** que identifica mi trabajo (nombre, código, materia, fecha, título)	+25 pts	0 pts	25 pts
Incluí una **descripción y conclusiones** de mi trabajo	+25 pts	0 pts	25 pts
		Suma:	100 pts

Introducción:

Después de analizar las instrucciones y los videos de la plataforma, primero empecé haciendo los programas de la cola y la pila vistos en clase, después hice un pequeño programa en donde hice pruebas de introducción de la cadena a la pila y hacia la cola y que se mostrará, en donde más adelante lo adapté para usarlo en el programa final como el interfaz del programa en donde tendrá dialogo con el usuario para introducir las expresiones con notación infija hacia una cola, además en uno de sus métodos se inicializo un nuevo objeto con una pila, en donde se hará la conversión de la expresión. Una vez comprobado la introducción correcta de la cadena, se hizo una clase Expresión en donde se hará la conversión, teniendo una pila como atributo y también tenemos unos métodos privados en donde nos darán valores enteros si es un operador o será 0 si es un operando, y otro método que comparara la precedencia correspondiente a los operadores nuestra pila y el operador de nuestra cadena, regresando un valor entero, y como sus métodos, un método para recibir la expresión con notación infija (en donde se desencola nuestra expresión infija para apilarlo en la pila de la clase hasta que este vacía la cola de la expresión infija) y otro para convertir a una expresión con notación posfija (en donde la pila de la expresión infija ya pasada a la pila se analizara y con varias condiciones con los métodos privados que tiene la clase, regresando una cola de la expresión ya con notación posfija). Para el método de conversión de infija a posfija se siguió principalmente el algoritmo que mostro el profesor en su video en donde muestra lo que hay que hacer en cada condición que tendremos en la conversión, se utiliza las if ’s y otra de manera anidadas para cubrir cada caso para su conversión

adecuada. Finalmente, en la main solo se creó un objeto de la clase Interfaz, para después hacer el método principal de la clase Interfaz, en donde solo es una iteración do-while, y llama a otro método del interfaz, el de conversión a posfija, y un diálogo con el usuario para repetirse o no.

Código fuente:

```
#ifndef ANY_EXCEPTION_H_INCLUDED
#define ANY_EXCEPTION_H_INCLUDED

#include <string>
#include <exception>

class AnyException : public std::exception {
    private:
        std::string msg;
    public:
        explicit AnyException(const char* message): msg(message) {}
        explicit AnyException(const std::string& message): msg(message)
{}
        virtual ~AnyException() throw() {}
        virtual const char* what() const throw () {
            return msg.c_str();
            }
    };


#endif // ANY_EXCEPTION_H_INCLUDED


#ifndef STACK_H_INCLUDED
#define STACK_H_INCLUDED

#include <string>
#include "AnyException.h"

//Definicion
template <class T, int ARRAYSIZE = 1024>
class Stack {
    private:
        //Atributos
        T data[ARRAYSIZE];
        int top;

        //Metodos privados
        void copyAll(const Stack<T, ARRAYSIZE>&);

    public:
        //Constructores
        Stack();
        Stack(const Stack<T, ARRAYSIZE>&);

        //Sobrecarga de operador
        Stack<T, ARRAYSIZE>& operator = (const Stack<T, ARRAYSIZE>&);
```

```
        //Metodos para una Pila
        bool isEmpty();
        bool isFull();

        void push(const T&);
        T pop();
        T getTop();
    };

//Implementacion

using namespace std;

//Metodos privados
template <class T, int ARRAYSIZE>
void Stack<T, ARRAYSIZE>::copyAll(const Stack<T, ARRAYSIZE>& s) {
    for(int i(0); i <= s.top; i++) {
        data[i] = s.data[i];
        }
    top = s.top;
    }

//Constructores
template <class T, int ARRAYSIZE>
Stack<T, ARRAYSIZE>::Stack() : top(-1) {}

template <class T, int ARRAYSIZE>
Stack<T, ARRAYSIZE>::Stack(const Stack<T, ARRAYSIZE>& s) {
    copyAll(s);
    }

//Sobrecarga de operador
template <class T, int ARRAYSIZE>
Stack<T, ARRAYSIZE>& Stack<T, ARRAYSIZE>::operator = (const Stack<T,
ARRAYSIZE>& s) {
    copyAll(s);
    return *this;
    }

//Metodos para una Pila
template <class T, int ARRAYSIZE>
bool Stack<T, ARRAYSIZE>::isEmpty() {
    return top == -1;
    }

template <class T, int ARRAYSIZE>
bool Stack<T, ARRAYSIZE>::isFull() {
    return top == ARRAYSIZE - 1;
    }

template <class T, int ARRAYSIZE>
void Stack<T, ARRAYSIZE>::push(const T& newElement) {
    if(isFull()) {
        throw AnyException("Desbordamiento de datos, push");
        }
```

```
    data[++top] = newElement;
    }

template <class T, int ARRAYSIZE>
T Stack<T, ARRAYSIZE>::pop() {
    if(isEmpty()) {
        throw AnyException("Insuficiencia de datos, pop");
        }
    return data[top--];
    }

template <class T, int ARRAYSIZE>
T Stack<T, ARRAYSIZE>::getTop() {
    if(isEmpty()) {
        throw AnyException("Insuficiencia de datos, getTop");
        }
    return data[top];
    }

#endif // STACK_H_INCLUDED


#ifndef QUEUE_H_INCLUDED
#define QUEUE_H_INCLUDED

#include <string>
#include "AnyException.h"

//Definicion
template <class T, int ARRAYSIZE = 1024>
class Queue {
    private:
        //Atributos
        T data[ARRAYSIZE];
        int frontPos;
        int endPos;

        //Metodos privados
        void copyAll(const Queue<T, ARRAYSIZE>&);

    public:
        //Constructores
        Queue();
        Queue(const Queue<T, ARRAYSIZE>&);

        //Sobrecarga de operador
        Queue<T, ARRAYSIZE>& operator = (const Queue<T, ARRAYSIZE>&);

        //Metodos para una Cola
        bool isEmpty();
        bool isFull();

        void enqueue(const T&);
        T dequeue();
        T getFrontPos();
    };
```

```
//Implementacion

using namespace std;

//Metodos privados
template <class T, int ARRAYSIZE>
void Queue<T, ARRAYSIZE>::copyAll(const Queue<T, ARRAYSIZE>& q) {
    int i(q.frontPos);
    while(i != q.endPos + 1) {
        data[i] = q.data[i];

        if(++i == ARRAYSIZE) {
            i=0;
            }
        }

    frontPos = q.frontPos;
    endPos = q.endPos;
    }

//Constructores
template <class T, int ARRAYSIZE>
Queue<T, ARRAYSIZE>::Queue() : frontPos(0), endPos(ARRAYSIZE-1) {}

template <class T, int ARRAYSIZE>
Queue<T, ARRAYSIZE>::Queue(const Queue<T, ARRAYSIZE>& q) {
    copyAll(q);
    }

//Sobrecarga de operador
template <class T, int ARRAYSIZE>
Queue<T, ARRAYSIZE>& Queue<T, ARRAYSIZE>::operator = (const Queue<T,
ARRAYSIZE>& q) {
    copyAll(q);
    return *this;
    }

//Metodos para una Cola
template <class T, int ARRAYSIZE>
bool Queue<T, ARRAYSIZE>::isEmpty() {
    return frontPos == endPos + 1
           or (frontPos == 0 and endPos == ARRAYSIZE - 1);
    }

template <class T, int ARRAYSIZE>
bool Queue<T, ARRAYSIZE>::isFull() {
    return frontPos == endPos + 2
           or (frontPos == 0 and endPos == ARRAYSIZE - 2)
           or (frontPos == 1 and endPos == ARRAYSIZE - 1);
    }

template <class T, int ARRAYSIZE>
void Queue<T, ARRAYSIZE>::enqueue(const T& newElement) {
    if(isFull()) {
        throw AnyException("Desbordamiento de datos, enqueue");
        }
```

```
    if(++endPos == ARRAYSIZE) {
        endPos = 0;
        }
    data[endPos] = newElement;
    }

template <class T, int ARRAYSIZE>
T Queue<T, ARRAYSIZE>::dequeue() {
    if(isEmpty()) {
        throw AnyException("Insuficiencia de datos, dequeue");
        }

    T result(data[frontPos]);

    if (++frontPos == ARRAYSIZE) {
        frontPos = 0;
        }
    return result;
    }

template <class T, int ARRAYSIZE>
T Queue<T, ARRAYSIZE>::getFrontPos() {
    if(isEmpty()) {
        throw AnyException("Insuficiencia de datos, getFrontPos ");
        }
    return data[frontPos];
    }

#endif // QUEUE_H_INCLUDED


#ifndef EXPRESSION_H_INCLUDED
#define EXPRESSION_H_INCLUDED

#include "AnyException.h"
#include "Stack.h"
#include "Queue.h"

class Expression {
    private:
        //Atributo
        Stack<char> *myStack;

        //Metodo privado
        int getPrecedence(const char&) const; //Pasamos los operadores
como valores, como la precedencia

        //Determina la precedencia, de la pila a utilizar y la de la cola
infija
        int comparePrecedence(Stack<char>, char)const;

    public:
        //Contructor
        Expression();
        Expression(Stack<char>&);

        //Metodos
```

```
        void getInfix(Queue<char>&);
        Queue<char> convertPosfix() ;
    };


#endif // EXPRESSION_H_INCLUDED


#include "Expression.h"
using namespace std;

int Expression::getPrecedence(const char& operators)const {
    switch (operators) {
        case '+':
        case '-':
            return 1;

        case '/':
        case '*':
            return 2;

        case '^':
            return 3;

        case '(':
            return 4;

        case ')':
            return 5;

        default:
            return 0;
        }
    }

int Expression::comparePrecedence(Stack<char> stackOp, char
operatorOfString)const {
    char stackSymbol;
    int i(0), opPos(0);

    while(!stackOp.isEmpty()) {
        i++;
        stackSymbol = stackOp.pop();

        if(getPrecedence(stackSymbol) == 4) { //Si el operador de la pila
operador es (
            break;
            }
        if(getPrecedence(operatorOfString) <= getPrecedence(stackSymbol))
{ //Si encuentra de mayor o igual presedencia
            opPos = i;
            }
        }
    if(opPos != 0) {
        return opPos;
        }
    else {
```

```
        return -1;
        }
    }

Expression::Expression() {}
Expression::Expression(Stack<char>& s):myStack(&s) {}

void Expression::getInfix(Queue<char>& q) {
    while(!q.isEmpty()) {
        myStack->push(q.dequeue());
        }
    }

Queue<char> Expression::convertPosfix() {
    Queue<char> queueConvert;
    Stack<char> stackOperator;
    int operatorPos(0);

    while(!myStack->isEmpty()) {

        if(getPrecedence(myStack->getTop()) == 0) {//Si un operando
            queueConvert.enqueue(myStack->pop());

            }
        else {//Si NO es un operando

            if(stackOperator.isEmpty()) { //Si la pila del operador esta
vacia
                stackOperator.push(myStack->pop());
                }
            else {
                if(getPrecedence(myStack->getTop()) == 5) { //Si el
operador es un )
                    while(getPrecedence(stackOperator.getTop()) != 4) {
                        queueConvert.enqueue(stackOperator.pop());
                        }

                    stackOperator.pop();//Se elimina el )
                    myStack->pop();


                    }
                else if(getPrecedence(myStack->getTop()) == 4) {//Si es(,
se apila
                    stackOperator.push(myStack->pop());


                    }
                else if((operatorPos = comparePrecedence(stackOperator,
myStack->getTop())) != -1) { //Si es un operador mayor o igual
precedencia
                    while(operatorPos > 0) {
                        queueConvert.enqueue(stackOperator.pop());
                        operatorPos--;
                        }
                    stackOperator.push(myStack->pop());//Se apila el
operador comparada
```

```
                    }
                else {
                    stackOperator.push(myStack->pop());//Se apila el
operador
                    }
                }
            }
        }

    while(!stackOperator.isEmpty()) {//Vacia los operadores de la pila
        queueConvert.enqueue(stackOperator.pop());
        }

    return queueConvert;
    }


#ifndef INTERFACE_H_INCLUDED
#define INTERFACE_H_INCLUDED

#include "Stack.h"
#include "Queue.h"
#include "Expression.h"
#include <iostream>

class Interface {
    private:
        //Atributos
        string expres;
        Queue<char> firstQueue, secondQueue;
        Stack<char> myStack;

    public:
        //Constructor
        Interface();
        Interface(Interface&);
        Interface& operator = (Interface&);

        //Metodos
        void convertToPosFix();
        void mainInterface();

    };
#endif // INTERFACE_H_INCLUDED


#include "Interface.h"

using namespace std;

Interface::Interface() {}

Interface::Interface(Interface& i): expres(i.expres),
firstQueue(i.firstQueue), secondQueue(i.secondQueue), myStack(i.myStack)
{}

Interface& Interface::operator = (Interface& i) {
```

```
    expres = i.expres;
    firstQueue = i.firstQueue;
    secondQueue = i.secondQueue;
    myStack = i.myStack;
    }


void Interface::convertToPosFix() {

    cout << "Ingrese expresion con notacion infija: ";
    cin>> expres;

    int i(expres.size());
    while(i >= 0) {
        firstQueue.enqueue(expres[i]);
        i--;
        }

    Expression myExpression(myStack);
    myExpression.getInfix(firstQueue);
    secondQueue = myExpression.convertPosfix();

    cout << "Conversion a notacion posfija: ";
    while(!secondQueue.isEmpty()) {
        cout << secondQueue.dequeue();
        }
    cout << endl << endl;
    }

void Interface::mainInterface() {
    char opc;

    cout<< "\tConvertidor de Notacion de Infija a Posfija"<<endl<<endl;
    do {
        convertToPosFix();

        cout << "Desea convertir otra expresion [S]i/[N]o): ";
        cin >> opc;
        opc = toupper(opc);

        cout << endl;

        }
    while(opc != 'N');
    }



//Efrain Robles Pulido

#include "Interface.h"

int main() {
    Interface myInterface;
    myInterface.mainInterface();
    return 0;
    }
```

"E:\Documentos PC\UDG Materias\ESTRUCTURA\A4 - Pila\Stack - Queue\bin\Debug\Stack - Queue.exe"

```
        Convertidor de Notacion de Infija a Posfija

Ingrese expresion con notacion infija: A/B^(C+D)-E*F/G^H
Conversion a notacion posfija: ABCD+^/EF*GH ^/-

Desea convertir otra expresion [S]i/[N]o): s

Ingrese expresion con notacion infija: (A-B)+(C/(D-E^F))/G*H
Conversion a notacion posfija: AB-CDEF^-/G/H *+

Desea convertir otra expresion [S]i/[N]o): s

Ingrese expresion con notacion infija: ((((((A+B)*C)-D)^E)/F)+G)*H
Conversion a notacion posfija: AB+C*D-E^F/G+H *

Desea convertir otra expresion [S]i/[N]o): s

Ingrese expresion con notacion infija: (A+B)/C(A^B+C)-E^A
Conversion a notacion posfija: AB+CAB^C+/EA ^-

Desea convertir otra expresion [S]i/[N]o): s

Ingrese expresion con notacion infija: ((A+E)^C-E)/(D*A-C)
Conversion a notacion posfija: AE+C^E-DA*C- /

Desea convertir otra expresion [S]i/[N]o): s

Ingrese expresion con notacion infija: A*F+G(E/T^B+U)*E
Conversion a notacion posfija: AF*GETB^/U+E *+

Desea convertir otra expresion [S]i/[N]o): s

Ingrese expresion con notacion infija: (((A*B)^E-B)^B+C*E)*A
Conversion a notacion posfija: AB*E^B-B^CE*+A *

Desea convertir otra expresion [S]i/[N]o): s

Ingrese expresion con notacion infija: A/B(((C-E*Q)+E*Q)^D)*C
Conversion a notacion posfija: ABCEQ*-EQ*+D^/C *

Desea convertir otra expresion [S]i/[N]o): N


Process returned 0 (0x0)   execution time : 5.558 s
Press any key to continue.
```

Conclusión:

En esta práctica comenzó fácil porque la teoría de la pila y de la cola esta más fácil de entender que el de las listas, pero cuando tuve que implementarlo al problema de actividad, fue cuando se volvió difícil porque tuve que hacer mucha analogía de como introducir la cadena o expresión hacia una cola, después pasarlo a una pila y regresarlo como una cola, así que fue mucho prueba y error para que funcionara correctamente, tuve que investigar como seria la analogía para determinar la precedencia de los operadores para el programa, también de cómo utilizar los switch con varias condiciones, pero no encontré mucho así que opte mejor por las if anidadas solo para el algoritmo de conversión. Me encontraba muchos errores de insuficiencia de datos debido a que quería meter elementos a las colas o la pila que estaban vacías, porque en las condiciones directamente hacia la acción de introducir elementos si que estuvieran inicializados. También tenia errores de que no se ordenaban correctamente debido a que usaba algunas veces un if en donde tuve que cambiar a un while para que analizara o vaciara toda la pila o la cola. Finalmente, con mucha investigación y pruebas y errores se pudo obtener el programa.