

**Universidad de Guadalajara**

---

**Centro Universitario de Ciencias Exactas e Ingenierías**



**IL365 - Estructura de Datos - D01**

**Actividad de Aprendizaje #04**

**Aplicación de Pila y Cola**

**Alumna: Cervantes Araujo Maria Dolores**

**Código: 217782452**

**Fecha de Elaboración: 12 febrero de 2023**



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

## Introducción:

Para esta actividad se maneja mucho la lógica y el criterio de reglas en conversiones, para lo que fue necesario aprender el uso de las pilas y colas estáticas, comprender como era la estructura de cada una para posteriormente abordar el problema que nos daba la instrucción de realizar una conversión de infija a posfija.

Al llevarlo a cabo se crearon dos colas una en la que se estaría almacenando la operación infija que digite el usuario y la segunda será útil para ir guardando el resultado final en conversión posfija, la pila nos ayuda a almacenar los operadores que se vayan encontrando en la operación mientras itera cada valor, al final de la iteración si quedan operadores en la pila, los colocara en el resultado.



## Código Fuente:

### Stack.hpp

```
#ifndef STACK_HPP_INCLUDED
#define STACK_HPP_INCLUDED
#include <iostream>
#include "LolException.hpp"

/// DEFINICION
template<class T, int ARRAY = 300>
class Stack {

    private:
        T pila[ARRAY];
        int top;
        void copyAll(const Stack<T, ARRAY>&);

    public:
        Stack();
        Stack(const Stack<T, ARRAY>& );
        bool isEmpty();
        bool full();

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

        Stack<T, ARRAY>& operator = (const Stack<T, ARRAY>& );
};

/// IMPLEMENTACION
using namespace std;
template<class T, int ARRAY>
Stack<T, ARRAY>::Stack() {
    top = -1;
}

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

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

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



```
template<class T, int ARRAY>
void Stack<T, ARRAY>::push(const T& obj) {
    if(full()) {
        throw LolException("Desbordamiento de datos.");
    }
    pila[++top] = obj;
}

template<class T, int ARRAY>
T Stack<T, ARRAY>::pop() {
    if(isEmpty()) {
        throw LolException("Insuficiencia de datos.");
    }
    return pila[top--];
}

template<class T, int ARRAY>
T Stack<T, ARRAY>::getTop() {
    if(isEmpty()) {
        throw LolException("Insuficiencia de datos.");
    }
    return pila[top];
}

template<class T, int ARRAY>
Stack<T, ARRAY>& Stack<T, ARRAY>::operator = (const Stack<T, ARRAY>& obj) {
    copyAll(obj);
    return *this;
}

template<class T, int ARRAY>
void Stack<T, ARRAY>::copyAll(const Stack<T, ARRAY>& obj) {
    int i = 0;
    while(i <= obj.top) {
        this -> pila[i] = obj.pila[i];
        i++;
    }
    this -> top = obj.top;
}

#endif // STACK_HPP_INCLUDED
```



### Queue.hpp

```
#ifndef QUEUE_HPP_INCLUDED
#define QUEUE_HPP_INCLUDED
#include <iostream>
#include "LolException.hpp"

/// DEFINICION
template<class A, int ARRAY = 300>
class Queue {
private:
    A cola[ARRAY];
    int endPos;
    int frontPos;
    void copyAll(const Queue<A, ARRAY>& );

public:
    Queue();
    Queue(const Queue<A, ARRAY>& );

    bool isEmpty();
    bool full();
    void enqueue(const A& );
    A dequeue();
    A getFront();

    Queue<A, ARRAY>& operator = (const Queue<A, ARRAY>& );
};

/// IMPLEMENTACION
using namespace std;
template<class A, int ARRAY>
void Queue<A, ARRAY>::copyAll(const Queue<A, ARRAY>& obj) {
    int i = obj.frontPos; // i = 2
    int j = obj.endPos; // j = 2

    if(i == obj.endPos + 2) {
        while(i <= ARRAY) {
            this->cola[i] = obj.cola[i];
            i++;
        }

        while(j >= 0) {
            this->cola[j] = obj.cola[j];
            j--;
        }
    }

    if(i == 0 or i == j or i < j) {
        while(i <= j) {
            this->cola[i] = obj.cola[i];
            i++;
        }
    }
}
```



```
    }

    this -> frontPos = obj.frontPos;
    this -> endPos = obj.endPos;

}

template<class A, int ARRAY>
Queue<A, ARRAY>::Queue() {
    frontPos = 0;
    endPos = ARRAY - 1;
}

template<class A, int ARRAY>
Queue<A, ARRAY>::Queue(const Queue<A, ARRAY>& obj) {
    copyAll(obj);
}

template<class A, int ARRAY>
bool Queue<A, ARRAY>::isEmpty() {
    return (frontPos == endPos + 1) || (frontPos == 0 && endPos == ARRAY - 1);
}

template<class A, int ARRAY>
bool Queue<A, ARRAY>::full() {
    return frontPos == endPos + 2 || (frontPos == 0 && endPos == ARRAY - 2) ||
    (frontPos == 1 && endPos == ARRAY - 1);
}

template<class A, int ARRAY>
void Queue<A, ARRAY>::enqueue(const A& obj) {
    if(full()) {
        throw LolException("Desbordamiento de datos.");
    }

    if(++endPos == ARRAY) {
        endPos = 0;
    }
    cola[endPos] = obj;
}

template<class A, int ARRAY>
A Queue<A, ARRAY>::dequeue() {
    if(isEmpty()) {
        throw LolException("Insuficiencia de datos.");
    }
    A aux(cola[frontPos]);

    if(++frontPos == ARRAY) {
        frontPos = 0;
    }
    return aux;
}
```



```
    }

    template<class A, int ARRAY>
    A Queue<A, ARRAY>::getFront() {
        if(isEmpty()) {
            throw LolException("Insuficiencia de datos.");
        }

        return cola[frontPos];
    }

    template<class A, int ARRAY>
    Queue<A, ARRAY>& Queue<A, ARRAY>::operator = (const Queue<A, ARRAY>& obj) {
        copyAll(obj);
        return *this;
    }

#endif // QUEUE_HPP_INCLUDED
```

### QueuePost.hpp

```
#ifndef QUEUEPOSFIJO_HPP_INCLUDED
#define QUEUEPOSFIJO_HPP_INCLUDED
#include <iostream>
#include "LolException.hpp"

/// DEFINICION
template<class A, int ARRAY = 300>
class QueuePosfijo {
    private:
        A cola[ARRAY];
        int endPos;
        int frontPos;
        void copyAll(const QueuePosfijo<A, ARRAY>& );

    public:
        QueuePosfijo();
        QueuePosfijo(const QueuePosfijo<A, ARRAY>& );

        bool isEmpty();
        bool full();
        void enqueue(const A& );
        A dequeue();
        A getFront();

        QueuePosfijo<A, ARRAY>& operator = (const QueuePosfijo<A, ARRAY>& );
};

/// IMPLEMENTACION
using namespace std;
template<class A, int ARRAY>
void QueuePosfijo<A, ARRAY>::copyAll(const QueuePosfijo<A, ARRAY>& obj) {
```



```
int i = obj.frontPos;
int j = obj.endPos;

if(i == obj.endPos + 2) {
    while(i <= ARRAY) {
        this->cola[i] = obj.cola[i];
        i++;
    }

    while(j >= 0) {
        this->cola[j] = obj.cola[j];
        j--;
    }
}

if(i == 0 or i == j or i < j) {
    while(i <= j) {
        this->cola[i] = obj.cola[i];
        i++;
    }
}

this->frontPos = obj.frontPos;
this->endPos = obj.endPos;
}

template<class A, int ARRAY>
QueuePosfijo<A, ARRAY>::QueuePosfijo() {
    frontPos = 0;
    endPos = ARRAY - 1;
}

template<class A, int ARRAY>
QueuePosfijo<A, ARRAY>::QueuePosfijo(const QueuePosfijo<A, ARRAY>& obj) {
    copyAll(obj);
}

template<class A, int ARRAY>
bool QueuePosfijo<A, ARRAY>::isEmpty() {
    return (frontPos == endPos + 1) || (frontPos == 0 && endPos == ARRAY - 1);
}

template<class A, int ARRAY>
bool QueuePosfijo<A, ARRAY>::full() {
    return frontPos == endPos + 2 || (frontPos == 0 && endPos == ARRAY - 2) ||
    (frontPos == 1 && endPos == ARRAY - 1);
}

template<class A, int ARRAY>
void QueuePosfijo<A, ARRAY>::enqueue(const A& obj) {
    if(full()) {
        throw LolException("Desbordamiento de datos.");
    }
}
```





```
    if(++endPos == ARRAY) {
        endPos = 0;
    }
    cola[endPos] = obj;
}

template<class A, int ARRAY>
A QueuePosfijo<A, ARRAY>::dequeue() {
    if(isEmpty()) {
        throw LolException("Insuficiencia de datos.");
    }
    A aux(cola[frontPos]);

    if(++frontPos == ARRAY) {
        frontPos = 0;
    }
    return aux;
}

template<class A, int ARRAY>
A QueuePosfijo<A, ARRAY>::getFront() {
    if(isEmpty()) {
        throw LolException("Insuficiencia de datos.");
    }

    return cola[frontPos];
}

template<class A, int ARRAY>
QueuePosfijo<A, ARRAY>& QueuePosfijo<A, ARRAY>::operator = (const QueuePosfijo<A,
ARRAY>& obj) {
    copyAll(obj);
    return *this;
}

#endif // QUEUEPOSFIJO_HPP_INCLUDED
```



### Menu.hpp

```
#ifndef MENU_HPP_INCLUDED
#define MENU_HPP_INCLUDED
#include "QueuePosfijo.hpp"
#include "Queue.hpp"
#include "Stack.hpp"
#include <iostream>
#include <string.h>

class Menu {
private:
    int op;
    Stack<char> Spila;
    Queue<char> colaInserc;
    QueuePosfijo<char> colaOut;
    char aux;

public:
    Menu();
    Menu(const Menu&);

    void convertinfijo_Postfijo();
    bool isOperador(const char&);
    int precedence(const char&);
    Menu& operator = (const Menu& );
};

#endif // MENU_HPP_INCLUDED
```



### Menu.cpp

```
#include "Menu.hpp"

using namespace std;
Menu::Menu() {
}

Menu::Menu(const Menu& m) {
    colaInserc = m.colaInserc;
    colaOut = m.colaOut;
    Spila = m.Spila;
}

void Menu::convertinfiijo_Postfijo() {
    char c[150];
    char aux;
    int i=0, tam, oper;

    cout<<"\n--> Digita una operacion"<<endl;
    cin.getline(c, 150, '\n');
    tam=strlen(c);
    while(i<tam) {
        colaInserc.enqueue(c[i]); ///Inserta la operacion inija
        i++;
    }
    while(!colaInserc.isEmpty()) {

        if(isOperador(colaInserc.getFront())) {
            oper=precedence(colaInserc.getFront());
            ///Si no es parentesis de apertura && parentesis menor a precedente
            apila el operador
            while(!Spila.isEmpty() && (aux=Spila.getTop()!='(') && oper<=
precedence(Spila.getTop())) {
                colaOut.enqueue(Spila.pop()); ///Se apila el digito en la cola del
resultado y se eliminan los operadores
            }
            Spila.push(colaInserc.dequeue()); ///Se recorren las posiciones de la
cola
        }
        else if(colaInserc.getFront()=='(') {
            Spila.push(colaInserc.dequeue());
        }

        ///Si parentesis de cierre:
        /// o que la pila este vacia desapilar operadores y pasarlos al resultado
        hasta enocntrar un parentesis de apertura
        else if(colaInserc.getFront()=='') {
            while((aux=Spila.getTop()!='(') {
                colaOut.enqueue(aux);
                Spila.pop();          ///Y desapilar el parentesis de apertura
            }
        }
    }
}
```



```
        colaInserc.dequeue();
        Spila.pop();
    }
    else {
        colaOut.enqueue(colaNInserc.dequeue());
    }
}

while(!Spila.isEmpty()) {
    colaOut.enqueue(Spila.pop());
}
while(!colaOut.isEmpty()) {
    cout<<colaOut.dequeue();
}
}

///Si se encuentran operadores.
///Desapilar operadores y pasarlos al resultado mientras estos sean de mayor o
igual precedencia que el que se esta leyendo
///Y no se encuentre con un parentesis de apertura o la pila este vacia
///apilar el operador leído Y Si es un operando/numero: pasarlo a la cola del
resultado

bool Menu::isOperador(const char&c) {
    char operadores[5]= {'+', '-', '*', '/', '^'};
    bool val(false);
    for(int i=0; i<5; i++) {
        if(c==operadores[i]) {
            val=true;
        }
    }
    return val;
}

///PRIORIDAD DE OPERADORES
int Menu::precedence(const char&c) {
    return(c=='+' || c=='-')? 1:
        (c=='*' || c=='/')? 2:
        (c=='^')? 3:0;
}

Menu& Menu::operator = (const Menu& m) {
    Spila = m.Spila;
    colaInserc = m.colaNInserc;
    colaOut = m.colaNOut;

    return *this;
}
```



### Main.cpp

```
#include <iostream>
#include "Menu.hpp"
using namespace std;

int main() {
    Menu contenedor;
    contenedor.convertirfijo_Postfijo();
    return 0;
}
```

### LolException.hpp

```
#ifndef LOLEXCEPTION_HPP_INCLUDED
#define LOLEXCEPTION_HPP_INCLUDED
#include <exception>
#include <iostream>
class LolException: public std::exception {

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

#endif // LOLEXCEPTION_HPP_INCLUDED
```



## Impresiones de Pantalla:

"C:\Users\cerva\Escritorio\F.Prog\POO C++\Actividad4\_v2\bin\Debug\Actividad4\_v2.exe"

```
--> Digita una operacion
(A-B)+(C/(D-E^F))/G*H
AB-CDEF^-/G/H*+
Process returned 0 (0x0)   execution time : 42.739 s
Press any key to continue.
```

"C:\Users\cerva\Escritorio\F.Prog\POO C++\Actividad4\_v2\bin\Debug\Actividad4\_v2.exe"

```
--> Digita una operacion
((4+5)/7)+8^B-2
45+7/8B^+2-
Process returned 0 (0x0)   execution time : 25.547 s
Press any key to continue.
```

"C:\Users\cerva\Escritorio\F.Prog\POO C++\Actividad4\_v2\bin\Debug\Actividad4\_v2.exe"

```
--> Digita una operacion
(((6+7-5)^G)-U/R)/4*1
67+5-G^UR/-4/1*
Process returned 0 (0x0)   execution time : 39.040 s
Press any key to continue.
```

"C:\Users\cerva\Escritorio\F.Prog\POO C++\Actividad4\_v2\bin\Debug\Actividad4\_v2.exe"

```
--> Digita una operacion
((((AE+45/U)*E)^3*7)-Y8)*(^9)/G+T)-6
AE45U/+E*3^7*Y8-9^*G/T+6-
Process returned 0 (0x0)   execution time : 99.198 s
Press any key to continue.
```



"C:\Users\cerva\Escritorio\F.Prog\POO C++\Actividad4\_v2\bin\Debug\Actividad4\_v2.exe"

```
--> Digita una operacion
((((87*4)/8+G)^2)/I-6)*R
874*8/G+2^I/6-R*
Process returned 0 (0x0)   execution time : 34.240 s
Press any key to continue.
```

"C:\Users\cerva\Escritorio\F.Prog\POO C++\Actividad4\_v2\bin\Debug\Actividad4\_v2.exe"

```
--> Digita una operacion
7+8(U*R(Y/Q)^3D-MA)+2
78URYQ/3D^*MA--+2+
Process returned 0 (0x0)   execution time : 46.758 s
Press any key to continue.
```

"C:\Users\cerva\Escritorio\F.Prog\POO C++\Actividad4\_v2\bin\Debug\Actividad4\_v2.exe"

```
--> Digita una operacion
8+9(((E*R/7)-U)^2+5)*T
89ER*7/U-2^5+T*+
Process returned 0 (0x0)   execution time : 33.335 s
Press any key to continue.
```



## Resumen Personal

Al término de esta actividad, puedo concluir que es mejor para la definición de los métodos, utilizar *template* para crear una plantilla la cual me permite colocar cualquier tipo de dato para el manejo de los métodos, porque de lo contrario no nos permite modificaciones durante la depuración del código, dado que solo nos permitirá cambios cuando creamos una copia de la pila o cola; al ser una implementación estática lo hace de cierta manera más factible para el manejo de caracteres entre pilas y colas.

Me genero conflicto la conversión porque al momento de ingresar la operación la pantalla solo se quedaba congelada y no retornaba ningún valor, ni mucho menos entraba a la función *toString()*, por lo que tuve que cambiar un poco la estructura del programa, y revisar en más de una ocasión la lógica en la función de infija a posfija y como es que me estaba almacenando los caracteres, percatándome que el fallo estaba en la iteración y la asignación de los valores entre las pilas y colas; una vez hechos los cambios fue posible lograr la conversión a una operación posfija.