Tecnológico Nacional de México
Campus Querétaro.

Implemente Estructura de Pilas (LIFO) en la evaluación de expresiones aritméticas.

Alumno:
Arellano Ochoa Daniel Ignacio

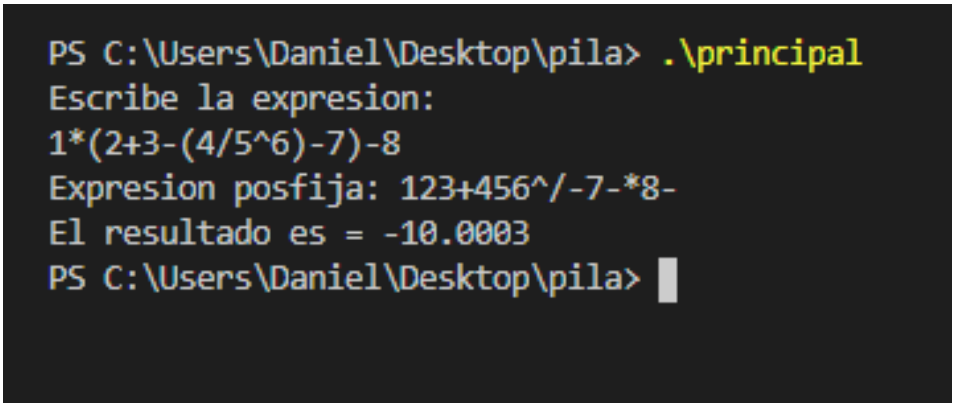
Carrera:
Ingeniera en sistemas computacionales

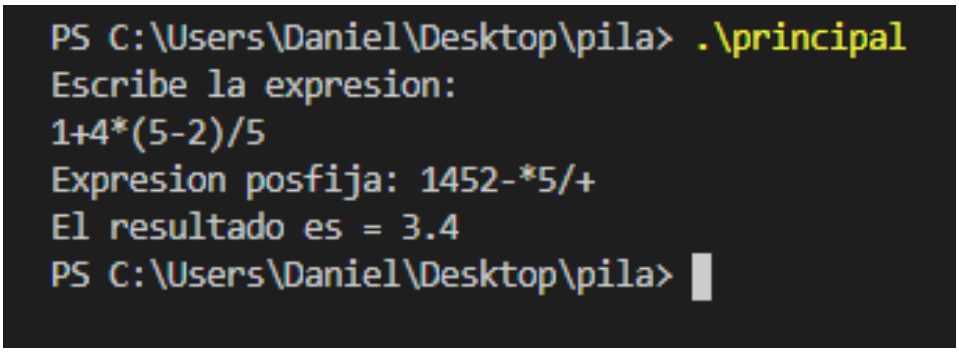
Docente:
Felipe Estrada Rojas

Fecha: 24/10/2020

Código: El archivo cpp se anexo junto con este trabajo, de igual manera el código se encuentra hasta el final de este trabajo.

Ejecución:





Conclusión:

El uso de la colas puede llegar a ser muy útil si se sabe utilizar adecuadamente, pues aunque no lo parezca las pilas se pueden encontrar implementadas en varias actividades cotidianas, en lo personal nunca habia pensado el cómo funcionaba un calculadora para evaluar las expresiones, con esta actividad me queda claro la importancia de las pilas dentro de la programación.

Código:

```
/*Nombre: Arellano Ochoa Daniel Ignacio
No. de control: 19141118
Programa que permite realizar la fncon basica de una calculadora (evaluar un
a expresion aritmetica)
*/
#include <string>
#include <iostream>
#include <stdlib.h>
#include <ctime>
#include <math.h>
#include "Calculadora.hpp"
using namespace std;


//Menu de inicio del programa
int main(int argc, const char * argv[]){
    Pila *pila1=new Pila();//pila creada para la calculadora
    Pila *pila2=new Pila();//pila creada para la calculadora
    Calculadora* casio=new Calculadora(pila1, pila2);//la calculadora requie
re dos pilas para su funcionamiento

    //se obtiene la expresion infija
    string expresion;
    cout<<"Escribe la expresion:"<<endl;
    getline(cin, expresion);

    //se pasa a posfija
    string pos=casio->pasarApostfija(expresion);
    cout<<"Expresion posfija: "<<pos<<endl;

    //se evalua
    double r= casio->resolver(pos);
    cout<<"El resultado es = "<<r<<endl;
}
```

```
#ifndef Nodo_hpp
#define Nodo_hpp

#include <string>

using namespace std;

class Nodo
```

```
{
    private:
        Nodo* siguiente;
        char dato;
        double num;

    public:
        Nodo(char);
        Nodo(double);
        Nodo* getSiguiente();
        char getDato();
        double getNum();
        void setSiguiente(Nodo*);

};


#endif /*Nodo_hpp*/
```

```
#include "Nodo.hpp"
#include <iostream>
#include <string>
//clase nodo que seran almacenados a una pila sin importar el dato que repre
senten
using namespace std;
//contructor de un nodo de caracter, con valores inicializados
Nodo::Nodo(char valor){
    this->dato=valor;//caracter inicializado con su parametro
    this-
>siguiente=NULL;//nodo que se encuentra adelante o arriba de este dentro de
la pila
    this-
>num=0;//al ser un nodo de caracter no es necesario darle un valor al numero
}
//constructor de un nodo de numeros reales, con valores inicializados
Nodo::Nodo(double numero){
    this->num=numero;//numero unicializado con su parametro
    this-
>siguiente=NULL;//nodo que se encuentra encima o adelante de este dentro de
una pila
    this-
>dato=0;//al ser un nodo de numero no es necesario darle un valor al caracte
r
}
```

```
//metodo que regresa el nodo siguiente de este dentro de una pila
Nodo* Nodo::getSiguiente(){
    return this->siguiente;
}
//metodo que regresa un caracter del nodo
char Nodo::getDato(){
    return this->dato;
}
//metodo que regresa el numero de un nodo
double Nodo::getNum(){
    return this->num;
}
//metodo que permite modificar el nodo adelante o encima de este dentro de u
na pila
void Nodo::setSiguiente(Nodo* n){
    this->siguiente=n;
}
```

```
#ifndef Pila_hpp
#define Pila_hpp

#include <string>
#include "Nodo.hpp"

using namespace std;
class Pila
{
    private:
        int elementos;
        Nodo* tope;
    public:
        Pila();
        void sacarTope();
        Nodo* consultarTope();
        int getElementos();
        void insertar(Nodo*);

};



#endif /*Pila_hpp*/
```

```cpp
#include "Pila.hpp"
#include "Nodo.hpp"
#include <iostream>
#include <string>
 //clase pila que que almecenara nodos, y ayudaran a la clase calculadroa a
almacenar datos sea el caso
using namespace std;
//constructor inicializando atributos
Pila::Pila(){
    this->elementos=0;//cantidad de nodos que almacena
    this->tope=NULL;//nodo que se encuentra en el top
}
//metodo que permite sacar de la pila
void Pila::sacarTope(){
    this->elementos--;
    this->tope=this->tope->getSiguiente();
}
//metodo que permite consultar el tope de la pila
Nodo* Pila::consultarTope(){
    return this->tope;
}
//metodo que permite consultar el total de nodos que hay dentro
int Pila::getElementos(){
    return this->elementos;
}
//metodo que permite insertar un nuevo nodo a la pila
void Pila::insertar(Nodo* n){
    this->elementos++;
    n->setSiguiente(this->tope);
    this->tope=n;
}
```

```cpp
#ifndef Calculadora_hpp
#define Calculadora_hpp
#include <string>
#include "Nodo.hpp"
#include "Pila.hpp"

using namespace std;

class Calculadora
{
    private:
        Pila* pilaOperadores;
```

```
        Pila* pilaNumeros;
    public:
        Calculadora(Pila*, Pila*);
        string pasarApostfija(string);
        double resolver(string);
        bool validadJerarquia(Nodo*);

};
#endif /* Calculadora_hpp*/
```

```
#include "Calculadora.hpp"
#include "Pila.hpp"
#include "Nodo.hpp"
#include <string>
#include <iostream>
#include <stdlib.h>
#include <math.h>
//clase de calculadora que realizara los procesos basicos de una calculadora
using namespace std;
//constructor con parametros de dos pilas, las cuales serviran para realizar
 las funciones de la calculadora
Calculadora::Calculadora(Pila* pila1, Pila* pila2){

    this-
>pilaNumeros=pila1;//pila de numeros que ayudara a la evaluacion de las expr
esion posfijas
    this-
>pilaOperadores=pila2;//pila de caracteres que ayudadra a realizar la trnafo
rmacion a posfija
}
//Metodo que permitira pasar de una expresion infija (parametro) a una posfi
ja, utilizando la pila de caracteres que se creo en el constructor de la cal
culadora
string Calculadora::pasarApostfija(string infija){
    char* aux;//el caractere auxiliar que ira tomando el valor de cada carac
ter de la expresion infija para poder analizarla
    string posfija="";//expresion posfija incializada en blanco
    Nodo* nodo;

    for (size_t i = 0; i < infija.length(); i++)
    {
        *aux=infija.at(i);
        if(*aux>=48 && *aux<=57){//identifica si es un numero si es el caso
la pasa a la expresion posfija
```

```
            posfija=posfija + *aux;
        }else if(*aux=='(' || *aux=='+' || *aux=='-
' || *aux=='*' || *aux=='/' || *aux=='^'){//en caso de ser un operador o par
entesis izquierdo se iran almacenando en la pila segun sea el caso

            nodo =new Nodo(*aux);
            bool con=true;
            do//todo el codigo dentro del do-
while, se puede colocar en una funcion recursiva, pero creo que sera mas com
prensible el codigo si se deja de manera iterativa
            {
                if(this->pilaOperadores->getElementos()==0){//caso base--
>en caso de estar vacia la pila se intrducira el operador
                    this->pilaOperadores->insertar(nodo);
                    con=false;
                }else if(*aux=='('){//caso base--
>en caso de ser parentesis izquierdo ocurrira lo mismo de introducirlo dentr
o de la pila sin evaluar su jerarquia
                    this->pilaOperadores->insertar(nodo);
                    con=false;
                }else{//caso base--
>en caso contrario de los dos anteriores, el operador sera evaluado segun su
 jerarquia
                    if(this-
>validadJerarquia(nodo)){//si su jerarquia es mayor que la del operador que
se encuentra en la cima de la pila, este sera insertado en la pila tomando l
ugar como el top
                        this->pilaOperadores->insertar(nodo);
                        con=false;
                    }else{//caso recursivo--
>en caso de tener una menor o igual jerarquia al top, el top sera sacado de
la pila e introducido a la expresion posfija y se volvera a repetir la compa
racion de jerarqui con el nuevo top
                        posfija=posfija + this->pilaOperadores-
>consultarTope()->getDato();
                        this->pilaOperadores->sacarTope();
                    }
                }
            } while (con);
        }else if(*aux==')'){//en caso de ser un parentesis derecho todos los
 operador dentro de la pila seran sacados uno a uno y colocados en la expres
ion posfija hasta que se encuentre un parentesis izquierdo el cual en vez de
 ser colocado a la expresion sera slo eliminado de la pila
            bool con2=true;
            do
```

```
            {
                posfija=posfija + this->pilaOperadores->consultarTope()-
>getDato();
                this->pilaOperadores->sacarTope();
                if(this->pilaOperadores->consultarTope()->getDato()=='('){
                    this->pilaOperadores->sacarTope();
                    con2=false;
                }else{
                    con2=true;
                }
            } while (con2);
        }
    }
    while (this->pilaOperadores-
>getElementos()>0)//finalmente, todos los operadores restantes dentro de la
pila seran sacados y colocados en la expresion posfija
    {
        posfija=posfija + this->pilaOperadores->consultarTope()->getDato();
        this->pilaOperadores->sacarTope();
    }

    return posfija;//se regresa la expresion posfija
}
//metodo que resuelve la operacion resiviendo como parametro una expresion p
osfija
double Calculadora::resolver(string expresion){
    char aux;//caracter auxiliar que ira tomando los valores de cada caracte
r dentro de la expresion
    double resultado;
    Nodo* nodo;
    Nodo* nodoR;
    for (size_t i = 0; i < expresion.length(); i++)
    {
        aux=expresion.at(i);
        if(aux>=48 && aux<=57){//en caso de ser un numero se colocara dentro
 de la pila de numeros
            double z=aux-48;
            nodo=new Nodo(z);
            this->pilaNumeros->insertar(nodo);
        }else{//en caso de ser un operador, se secaran los dos valores del t
op de la pila
            double b=this->pilaNumeros->consultarTope()-
>getNum();//b es el top
```

```
            this->pilaNumeros-
>sacarTope();//b es sacado, por lo tanto el numero debajo de la pila sera el
 nuevo top, representado como 'a'
            double a=this->pilaNumeros->consultarTope()-
>getNum();//a es el nuevo top
            this->pilaNumeros-
>sacarTope();//a es sacado por lo tanto el valor de bajo tomara el nuevo top
            double c=0;
            switch (aux)//segun sea el caso se realizara la operacion y alme
ceanra el resultado en c, de manera "c= a & b"
            {
                case '+':
                {
                    c=a+b;
                }
                break;
                case '-':
                {
                    c=a-b;
                }
                break;
                case '*':
                {
                    c=a*b;
                }
                break;
                case '/':
                {
                    c=a/b;
                }
                break;
                case '^':
                {
                    c=pow(a,b);
                }
                break;

            }
            nodoR = new Nodo(c);
            this->pilaNumeros-
>insertar(nodoR);//el resultado sera almacenado en la pila
        }

    }
```

```
    resultado=this->pilaNumeros->consultarTope()-
>getNum();//Al final solamente debera quedar un numero dentro de la pila, el
 cual es el resultado de la expresion
    this->pilaNumeros->sacarTope();

    return resultado;


}
//regresa true si el dato almacenado en el nodo nuevo (parametro) tiene mayo
r prioridad que el del tope en caso contrario regresara un false
bool Calculadora::validadJerarquia(Nodo* nuevo){
    int aux;//se dan valores numericos segun sea la importnacia en la jerarq
uia de operaciones
    if(nuevo->getDato()=='('
        aux=0;
    else if(nuevo->getDato()=='+' || nuevo->getDato()=='-')
        aux=1;
    else if(nuevo->getDato()=='*' || nuevo->getDato()=='/')
        aux=2;
    else if(nuevo->getDato()=='^')
        aux=3;

    int auxT;//se da valor numericos al operador del top
    if(this->pilaOperadores->consultarTope()->getDato()=='('
        auxT=0;
    else if(this->pilaOperadores->consultarTope()->getDato()=='+' || this-
>pilaOperadores->consultarTope()->getDato()=='-')
        auxT=1;
    else if(this->pilaOperadores->consultarTope()->getDato()=='*' || this-
>pilaOperadores->consultarTope()->getDato()=='/')
        auxT=2;
    else if(this->pilaOperadores->consultarTope()->getDato()=='^')
        auxT=3;

    if(aux>auxT)//se compran ambas si es mayor la del parametro regresara un
 true
        return true;
    else
        return false;
}
```