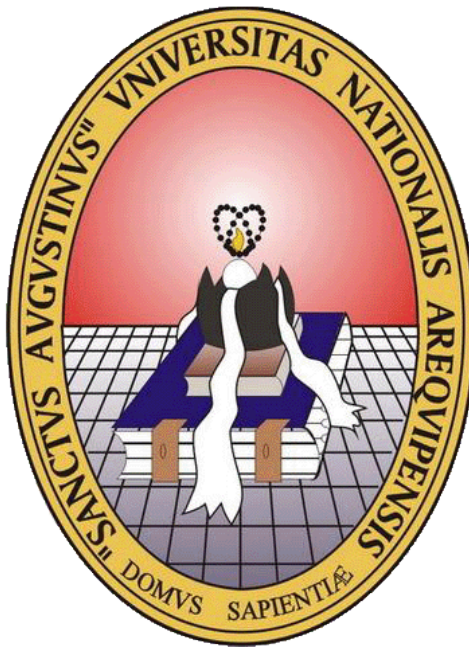


UNIVERSIDAD NACIONAL DE SAN AGUSTÍN DE AREQUIPA
FACULTAD DE PRODUCCIÓN Y SERVICIOS
ESCUELA DE CIENCIAS DE LA COMPUTACIÓN



PRÁCTICA DE LABORATORIO 11
CURSO DE CIENCIAS DE LA COMPUTACIÓN II

ESTUDIANTE:
RUIZ MAMANI, EDUARDO GERMÁN

EMAIL: eruizm@unsa.edu.pe

CUI: 20193061

TURNO:

C

AREQUIPA- PERÚ

2021

LINK DEL REPOSITORIO: https://github.com/EGRM23/CCII_20193061.git

1. EJERCICIO 1, 2 y 3

1. Defina una Pila que permita insertar elementos utilizando clases.
2. Sobre el ejercicio anterior, adecue el programa para eliminar elementos de una Pila.
3. Implemente un algoritmo para buscar elementos de la Pila.

- **CÓDIGO**

- **Item.h**

```
#ifndef ITEM_H
#define ITEM_H
#include <iostream>
using namespace std;

//EDUARDO GERMAN RUIZ MAMANI
//CUI: 20193061

template <typename R> class Stack;
template <typename R> class Item {
public:
    Item(R val, Item* = NULL);
    ~Item();
    friend class Stack<R>;
private:
    Item* debajo;
    R valor;
};

template <typename R>
Item<R> :: Item(R val, Item* d) {
    valor = val;
    debajo = d;
}

template <typename R>
Item<R> :: ~Item() {}

#endif
```

- **Item.cpp**

```
#include "Item.h"

//EDUARDO GERMAN RUIZ MAMANI
//CUI: 20193061
```

- **Stack.h**

```
#ifndef STACK_H
#define STACK_H
#include <iostream>
#include "Item.h"
using namespace std;
```

```
//EDUARDO GERMAN RUIZ MAMANI  
//CUI: 20193061
```

```
template <typename R> class Item;  
template <typename R> class Stack {  
public:  
    Stack(char);  
    ~Stack();  
    void push(const R val);  
    void pop();  
    R& top();  
    bool find(const R val);  
    void mostrarpila();  
private:  
    Item<R>* cima;  
    int cant;  
    char nom;  
};  
  
template <typename R>  
Stack<R> :: Stack(char n) {  
    cima = NULL;  
    cant = 0;  
    nom = n;  
}  
  
template <typename R>  
Stack<R> :: ~Stack() {  
    Item<R> *temp = cima;  
    Item<R> *borrar;  
    while(temp!=NULL){  
        borrar = temp;  
        temp = temp->debajo;  
        delete borrar;  
    }  
}  
  
template <typename R>  
void Stack<R> :: push(const R val) {  
    Item<R>* nuevo = new Item<R>(val);  
  
    if (cima == NULL)  
        nuevo->debajo = NULL;  
    else  
        nuevo->debajo = cima;  
  
    cima = nuevo;  
    cant++;  
}  
  
template <typename R>  
void Stack<R> :: pop() {  
    Item<R>* temp = cima;  
    cima = cima->debajo;  
    delete temp;
```

```

    }

    template <typename R>
    R& Stack<R> :: top() {
        return cima->valor;
    }

    template <typename R>
    bool Stack<R> :: find(const R val) {
        Item<R>* temp = cima;

        while(temp!=NULL){
            if (temp->valor == val)
                return true;
            else
                temp = temp->debajo;
        }

        return false;
    }

    template <typename R>
    void Stack<R> :: mostrarpila() {
        cout << "\nPILA " << nom << ": \n";
        Item<R>* temp = cima;
        while (temp != NULL) {
            cout << temp->valor << endl;
            temp = temp->debajo;
        }
    }
}

#endif

```

- **Stack.cpp**

```

#include "Stack.h"

//EDUARDO GERMAN RUIZ MAMANI
//CUI: 20193061

```

- **main.cpp**

```

#include<iostream>
#include "Item.h"
#include "Stack.h"
using namespace std;

//EDUARDO GERMAN RUIZ MAMANI
//CUI: 20193061

int main (int argc, char *argv[]) {

    Stack<int> p1('1');

    p1.push(3);
    p1.push(5);
    p1.push(7);
}

```

```

p1.mostrarpila();

p1.pop();
p1.mostrarpila();

p1.push(9);
p1.mostrarpila();

int n = 5;

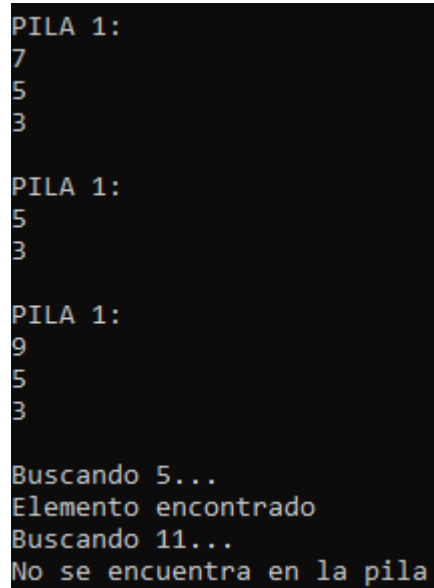
cout << "\nBuscando " << n << "... \n";
if (p1.find(n) == true)
    cout << "Elemento encontrado\n";
else
    cout << "No se encuentra en la pila\n";

n = 11;
cout << "Buscando " << n << "... \n";
if (p1.find(n) == true)
    cout << "Elemento encontrado\n";
else
    cout << "No se encuentra en la pila\n";

return 0;
}

```

- **CAPTURAS**



```

PILA 1:
7
5
3

PILA 1:
5
3

PILA 1:
9
5
3

Buscando 5...
Elemento encontrado
Buscando 11...
No se encuentra en la pila

```

2. EJERCICIO 4

4. Escribir un programa que dé la solución al problema de las Torres de Hanoi para N discos, utilizando pilas, las cuales representen cada uno de los postes:



- **CÓDIGO (función HANOI y parte del main)**

```
#include<iostream>
#include "Item.h"
#include "Stack.h"
using namespace std;

//EDUARDO GERMAN RUIZ MAMANI
//CUI: 20193061

template <typename R>
void Hanoi (int n, Stack<R>* origen, Stack<R>* aux, Stack<R>*
destino) {
    int disco;
    if (n == 1) {
        disco = origen->top();
        origen->pop();
        destino->push(disco);

        origen->mostrarpila();
        aux->mostrarpila();
        destino->mostrarpila();
        cout << "-----";
    } else {
        Hanoi<R>(n-1, origen, destino, aux);

        disco = origen->top();
        origen->pop();
        destino->push(disco);

        origen->mostrarpila();
        aux->mostrarpila();
        destino->mostrarpila();
        cout << "-----";

        Hanoi<R>(n-1, aux, origen, destino);
    }
}

int main (int argc, char *argv[]) {
    cout << "-----"
    -----";

    cout << endl << "\nPROBLEMA HANOI\n";
```

```

Stack<int> A('A');
Stack<int> B('B');
Stack<int> C('C');

int cantdisc;

cout << "Ingrese la cantidad de discos: ";
cin >> cantdisc;

for (int i = cantdisc; i > 0; i--) {
    A.push(i);
}

A.mostrarpila();
B.mostrarpila();
C.mostrarpila();
cout << "-----";

Hanoi<int>(cantdisc, &A, &B, &C);

return 0;
}

```

- **CAPTURAS**

```

PROBLEMA HANOI
Ingrese la cantidad de discos: 3

PILA A:
1
2
3

PILA B:

PILA C:
-----
PILA A:
2
3

PILA B:

PILA C:
1
-----
PILA A:
3

PILA C:
1

PILA B:
2
-----

```

PILA C:

PILA A:

3

PILA B:

1

2

PILA A:

PILA B:

1

2

PILA C:

3

PILA B:

2

PILA C:

3

PILA A:

1

PILA B:

PILA A:

1

PILA C:

2

3

PILA A:

PILA B:

PILA C:

1

2

3

<< El programa ha finalizado: codigo de salida: 0 >>

<< Presione enter para cerrar esta ventana >>