PRÁCTICA/LABORATORIO Nº 05

PILAS Y APLICACIONES CON PILAS

Objetivos:

- Utilizar Dev-C++.
- Implementar pilas en arreglos y listas simplemente enlazadas. (1, 2 y 3).
- Resolver problemas utilizando pilas. (4,5,6).
- Resolver problemas de tratamiento de expresiones aritméticas usando pilas (7,8).
- Resolver el problema de las torres de Hanoi usando pilas (9).

Instrucciones:

- Descargar la librería **base** del aula virtual: **pila.hpp**.
- El archivo **pila.hpp** debe dar 0 errores y alertas al **intentar compilarse** (ver imagen). Si no implementó alguna de las clases, eliminarla. Puede probar la librería en un archivo .cpp.
- Puede probar la librería **aplicaciones_pilas.hpp** en un archivo.cpp que tendrá **#include"pila.hpp"** antes de **#include"aplicaciones_pilas.hpp"** en la cabecera (header).
- NO hacer **#include"pila.hpp"** dentro de **aplicaciones_pilas.hpp**.
- Para el envío de la resolución de la práctica, adjuntar el/los archivo/s dentro de un .zip:
  - **pila.hpp (obligatorio)** y/o
  - **aplicaciones_pilas.hpp** y/o
  - **hanoi.cpp**
- Si su envío no sigue las instrucciones señaladas, se calificará con 0.

1. **Implementar, en el archivo "pila.hpp", los métodos de la clase "pila_int_array". (3 pts.)**
2. **Implementar, en el archivo "pila.hpp", los métodos de la clase "dos_pilas_int_array". (2 pts.)**
3. **Implementar, en el archivo "pila.hpp", los métodos de la clase "pila_char_list". (3 pts.)**

* El archivo pila.hpp contiene en cada clase los prototipos de las funciones que debe implementar.

pila.hpp

```
using namespace std;

struct nodo_char_pila
{
    char key;
    nodo_char_pila *prev;
    nodo_char_pila(char k)
    {
        key=k;
    }
};

class pila_int_array //pila de enteros implementada en un arreglo
{
    public:
        int *stack; //arreglo donde se almacenaran los elementos
        int top; //indice de el ultimo elemento ingresado
        int max; //tamanho del arreglo stack

        pila_int_array(int m) //constructor
        {
            this->stack=new int[m]; //arreglo dinamico tamanho m
            this->max=m; //guardamos el maximo (usar en overflow)
            this->top=-1; //sin elementos (usar en underflow)
        }

        bool empty(); //1 si vacio, O/W 0
        bool full(); //1 si lleno, O/W 0
        void push(int k); //añade k a la pila, verifica overflow con full()
        int pop(); //remueve un elemento de la pila, verifica underflow con empty()

};

class dos_pilas_int_array //dos pilas de enteros implementada en UN SOLO arreglo
{
    public:
        int *stack; //arreglo donde se almacenaran los elementos de AMBAS pilas
        int top1; //indice de el ultimo elemento ingresado en la PILA1
        int top2; //indice de el ultimo elemento ingresado en la PILA2
        int max; //tamanho del arreglo stack

        dos_pilas_int_array(int m) //constructor
        {
            this->stack=new int[m]; //arreglo dinamico tamanho m
            this->max=m; //guardamos el maximo
            this->top1=-1; //sin elementos en PILA1
            this->top2=max; //sin elementos en PILA2
        }

        bool empty1(); //1 si vacio PILA1, O/W 0
        bool empty2(); //1 si vacio PILA2, O/W 0
        bool full1(); //1 si lleno PILA1, O/W 0,
        bool full2(); //1 si lleno PILA2, O/W 0,
        void push1(int k); //añade k a la PILA1, verifica overflow con full1()
        void push2(int k); //añade k a la PILA2, verifica overflow con full2()
        int pop1(); //remueve un elemento de la PILA1, verifica underflow con empty1()
        int pop2(); //remueve un elemento de la PILA2, verifica underflow con empty2()
};

class pila_char_list //pila de caracteres implementada en una lista
{
    public:
        nodo_char_pila *top; //puntero al ultimo nodo de la lista

        pila_char_list() //constructor
        {
            this->top=nullptr; //nullptr = NULL en C++11
        }

        bool empty(); //1 si vacio, O/W 0
        void push(char k); //añade k a la pila
        char pop(); //remueve un elemento de la pila, verifica underflow con empty()
};
```

4. **Implementar, en el archivo "aplicaciones_pilas.hpp" una función `int maximo(pila_int_array x)`, que usando los métodos pop y push de la clase "pila_int_array" para manejar la pila, retorne el máximo elemento de x (puedes usar una pila auxiliar). Retornar 0 si la pila está vacía. (2 puntos)**

5. **Implementar en el archivo "aplicaciones_pilas.hpp" la función `void dosmil48(pila_int_array &x)` que, usando los métodos pop y push de la clase "pila_int_array" para manejar la pila, combine los elementos de igual valor de x empezando desde el top. (2 puntos)**
   Ejemplo: 64←8←4←2←2←top = 64←16←top

6. **Implementar en el archivo "aplicaciones_pilas.hpp" la función `bool palindromo(string x)` que, usando los métodos pop y push de la clase "pila_char_list", determine si x es palíndromo. (2 puntos)**

7. **Implementar en el archivo "aplicaciones_pilas.hpp" la función `bool balanceado(string x)` que, usando los métodos pop y push de la clase "pila_char_list", determine si x es una expresión aritmética con paréntesis, corchetes y llaves balanceados. (2 puntos)**

8. **Implementar en el archivo "aplicaciones_pilas.hpp" la función `int eval_posfija(string x)` que, usando los métodos pop y push de la clase "pila_int_array", evalue la expresión posfija. La expresión posfija x solo contiene [0-9] como operandos y a */+-^ como operadores. (2 puntos)**
   https://scanftree.com/Data_Structure/prefix-postfix-infix-online-converter

* Para todas las funciones se debe utilizar, como única estructura para almacenar datos, una pila a la cual deberá acceder solo a través de los métodos implementados en pila.hpp.

9. Torre de Hanoi (archivo hanoi.cpp) (2 puntos)

//si desea puede usar una pila de pilas.hpp o definir dentro del .cpp su propia pila

https://www.youtube.com/watch?v=k2nLziNzCDE

Se te ha dado el trabajo de resolver el problema de la Torre de Hanoi con **N** discos de diferentes tamaños.

Las torres se llaman **A**, **B** y **C**. Inicialmente, todos los discos están ubicados en la torre **S** ordenados de mayor a menor desde abajo hasta arriba (es decir, el más pequeño está en la cima de la torre) y quieres llevarlos a otra torre **D**. Solo puedes mover un disco a la vez y no puedes colocar un disco encima de otro que tenga menor tamaño.

Debes describir la secuencia de movimientos más corta posible para completar tu trabajo.

Entrada

La primera línea de entrada contiene un entero **T**, la cantidad de casos de prueba.

Las siguientes **T** líneas contienen un entero **N** y dos caracteres **S** y **D**, la cantidad de discos del **i**-ésimo caso de prueba, la torre en la que están ubicados los discos y la torre a la que quieres llevarlos, respectivamente.

Salida

Para cada caso de prueba, imprime la secuencia de movimientos más corta posible para completar tu trabajo. Cada movimiento debe ser descrito de la forma:

`Mueve el disco de X a Y`

Donde **X** y **Y** son los nombres de las torres que intervienen en el movimiento.

Luego de toda la secuencia, imprime `Listo!`

Ejemplo:

Entrada	Salida
2 2 A B 3 C B	Mueve el disco de A a C. Mueve el disco de A a B. Mueve el disco de C a B. Listo! Mueve el disco de C a B. Mueve el disco de C a A. Mueve el disco de B a A. Mueve el disco de C a B. Mueve el disco de A a C. Mueve el disco de A a B. Mueve el disco de C a B. Listo!