Pila

Es una estructura de datos representado como una lista ordenada donde el acceso a sus elementos es en orden inverso a como se han agregado, siguiendo el esquema LIFO (last in first out). En todo momento, solamente se tiene acceso a la parte superior de la pila (tope). Para el manejo de los datos, las operaciones actúan sobre el tope de la pila: colocar un objeto (apilar) y su operación inversa, retirar el último elemento apilado (desapilar).

Una definición del tipo Stack es la siguiente:

```
class Stack <T>
public:
    Constructor Stack() // construye la pila vacia.
    Destructor Stack()    // destruye la pila.
    function IsEmpty() : Boolean  // indica si la pila esta vacia o no
    void Push(ref T x)  // agrega un nuevo elemento al tope de la pila , luego de Push(x), se asegura ↩
        que Top()==x
    void Pop()  // precondicion: la pila no puede estar vacia. Elimina el elemento del tope
    function Size() : Integer   // devuelve el # de elementos en la pila
    function Top() : T  // precondicion: la pila no puede estar vacia y retorna la informacion presente↩
        en el tope de la pila
end
```

Implementación con apuntadores

```
class Node <T>
public:
    T tInfo
    Node* pNext
end

class Stack <T>
private:
    Node<T>* pTop
    Integer iN

public:
    Constructor Stack()
      iN = 0
      pTop = NIL
    end

    Destructor Stack ()
      while (pTop != NIL) do
          Node<T>* pTemp = *pTop
          pTop =  *pTop.pNext
          delete pTemp
      end
    end

    function IsEmpty() : Boolean
      return iN == 0
    end

    void Push (ref T x)
      Node<T>* pNew = new Node
      *pNew.x = *x
      *pNew.pNext = pTop
      pTop = pNew
      iN = iN + 1
    end

    void Pop()
      Node<T>* pTemp = pTop
      pTop =  *pTop.pNext
      delete pTemp
      iN = iN - 1
    end

    function Size() : Integer
      return iN
    end

    function Top() : T
      return *pTop.tInfo
    end
end
```

esmitt ramírez/2014