Usac

Nombre: Pablo Isai Matusalen Cutzal Mazariegos carné: 202209622

Introducción a la programación 2 tarea de **Enunciados TDA**

1. Implementar TDA Lista Ordenada en pseudocódigo utilizando memoria dinámica:

```
class TElem:
    def __init__(self, clave, dato):
        self.clave = clave
        self.dato = dato

class Nodo:
    def __init__(self, elemento):
        self.elemento = elemento
        self.siguiente = None

class ListaOrdenada:
    def __init__(self):
        self.cabeza = None

    def compara_elem(self, A, B):
        if A.clave < B.clave:
            return -1
        elif A.clave == B.clave:
            return 0
        else:
            return 1

    def insertar(self, elem):
        nuevo_nodo = Nodo(elem)
        if self.cabeza is None:
            self.cabeza = nuevo_nodo
        else:
            actual = self.cabeza
            anterior = None
            while actual is not None and self.compara_elem(actual.elemento, elem) < 0:
                anterior = actual
                actual = actual.siguiente
            if anterior is None:
                nuevo_nodo.siguiente = self.cabeza
                self.cabeza = nuevo_nodo
            else:
                nuevo_nodo.siguiente = anterior.siguiente
                anterior.siguiente = nuevo_nodo
```

2. Algoritmo para extraer un nodo de una lista doblemente encadenada en pseudocódigo:

```
class NodoDoble:
    def __init__(self, info):
        self.info = info
        self.anterior = None
        self.siguiente = None

class ListaDoble:
    def __init__(self):
        self.cabeza = None
        self.cola = None

    def insertar(self, info):
        nuevo_nodo = NodoDoble(info)
        if self.cabeza is None:
            self.cabeza = nuevo_nodo
            self.cola = nuevo_nodo
        else:
            nuevo_nodo.anterior = self.cola
            self.cola.siguiente = nuevo_nodo
            self.cola = nuevo_nodo

    def extraer(self, valor):
        actual = self.cabeza
        while actual is not None:
            if actual.info == valor:
                if actual.anterior is not None:
                    actual.anterior.siguiente = actual.siguiente
                else:
                    self.cabeza = actual.siguiente

                if actual.siguiente is not None:
                    actual.siguiente.anterior = actual.anterior
                else:
                    self.cola = actual.anterior

                del actual
                return True
            actual = actual.siguiente
        return False
```

Algoritmo en pseudocódigo utilizando memoria dinámica:

```
Función CrearPila(): Pila
    pila = Crear(Pila)
    pila.cima = NULL
    devolver pila
Fin Función

Función Apilar(p: Pila, c: caracter)
    nuevoNodo = Crear(Nodo)
    nuevoNodo.dato = c
    nuevoNodo.siguiente = p.cima
    p.cima = nuevoNodo
Fin Función

Función Desapilar(p: Pila): caracter
    Si p.cima ≠ NULL entonces
        c = p.cima.dato
        nodoAEliminar = p.cima
        p.cima = p.cima.siguiente
        Liberar(nodoAEliminar)
        devolver c
    Sino
        devolver NULL  // Indicador de pila vacía
    Fin Si
Fin Función

Función EsPilaVacia(p: Pila): booleano
    devolver p.cima = NULL
Fin Función

Función ParentesisEquilibrados(s: cadena): booleano
    pila = CrearPila()
    Para cada caracter c en s hacer
        Si c = '(' entonces
            Apilar(pila, c)
        Sino si c = ')' entonces
            Si EsPilaVacia(pila) entonces
                devolver Falso  // Hay más paréntesis de cierre que de apertura
            Fin Si
            Desapilar(pila)
        Fin Si
    Fin Para
    Si EsPilaVacia(pila) entonces
        devolver Verdadero  // Todos los paréntesis están equilibrados
    Sino
        devolver Falso  // Hay paréntesis de apertura sin cerrar
    Fin Si
Fin Función
```