2025-1

Tarea 4
LISTA DOBLEMENTE LIGADA

Estructura de Datos
Universidad Nacional Autónoma de México
Facultad de Estudios Superiores Aragón

REALIZADO POR:

Enrique Emiliano Cano García

22/08/2024

Clase Nodo Doble

```python
class NodoDoble:  4 usages  kno4
    def __init__(self, dato, anterior = None, siguiente = None):  kno4
        self.dato = dato
        self.anterior = None
        self.siguiente = None

    def get_dato(self):  3 usages (3 dynamic)  kno4
        return self.dato

    def get_anterior(self):  kno4
        return self.anterior

    def get_siguiente(self):  17 usages (17 dynamic)  kno4
        return self.siguiente

    def set_dato(self, dato):  2 usages (2 dynamic)  kno4
        self.dato = dato

    def set_anterior(self, anterior):  kno4
        self.anterior = anterior

    def set_siguiente(self, siguiente):  3 usages (3 dynamic)  kno4
        self.siguiente = siguiente

    def __str__(self):  kno4
        print(f"dato: {self.dato}, anterior: {self.anterior}, siguiente: {self.siguiente}")
```

Clase DoubleLinkedList

```python
from Tareas.Tarea5.NodoDoble import NodoDoble


class DoubleLinkedList:  2 usages  kno4
    def __init__(self):  kno4
        self.head = None
        self.tail = None
        self.tamanio = 0

    def esta_vacia(self):  5 usages  kno4
        return self.head is None

    def get_tamanio(self):  kno4
        return self.tamanio

    def agregar_al_inicio(self, dato):  1 usage  kno4
        nuevo_nodo = NodoDoble(dato, None, None)
        if self.esta_vacia():
            self.head = self.tail = nuevo_nodo
        else:
            nuevo_nodo.siguiente = self.head
            self.head.anterior = nuevo_nodo
            self.tail = nuevo_nodo
        self.tamanio += 1
```

```python
    def agregar_al_final(self, dato):  5 usages  kno4
        nuevo_nodo = NodoDoble(dato)
        if self.esta_vacia():
            self.head = self.tail = nuevo_nodo
        else:
            nuevo_nodo.anterior = self.tail
            self.tail.siguiente = nuevo_nodo
            self.tail = nuevo_nodo
        self.tamanio += 1

    def agregar_despues_de(self, referencia, dato):  kno4
        if self.esta_vacia():
            raise Exception("La lista esta vacia")
        actual = self.head
        while actual and actual.dato != referencia:
            actual = actual.siguiente
        if actual is None:
            raise Exception("Referencia no encontrada")
        nuevo_nodo = NodoDoble(dato)
        nuevo_nodo.siguiente = actual.siguiente
        nuevo_nodo.anterior = actual
        if actual.siguiente:
            actual.siguiente.anterior = nuevo_nodo
        actual.siguiente = nuevo_nodo
        if actual == self.tail:
            self.tail = nuevo_nodo
        self.tamanio += 1

    def obtener(self, posicion):  kno4
        if posicion < 0 or posicion >= self.tamanio:
            raise IndexError("Índice fuera de rango")
        actual = self.head
        for _ in range(posicion):
            actual = actual.siguiente
        return actual.valor

    def eliminar_primero(self):  1 usage  kno4
        if self.esta_vacia():
            raise Exception("La lista esta vacia")
        if self.head == self.tail:
            self.head = self.tail = None
        else:
            self.head = self.head.siguiente
            self.head.anterior = None
        self.tamanio -= 1

    def eliminar_ultimo(self):  1 usage  kno4
        if self.esta_vacia():
            raise Exception("La lista esta vacia")
        if self.head == self.tail:
```

```
74              raise Exception("La lista esta vacia")
75          if self.head == self.tail:
76              self.tail = self.head = None
77          else:
78              self.tail = self.tail.anterior
79              self.tail.siguiente = None
80          self.tamanio -= 1
81
82      def eliminar(self, posicion):  1 usage  kno4
83          if posicion < 0 or posicion >= self.tamanio:
84              raise IndexError("La posicion no es valida")
85          if posicion == 0:
86              self.eliminar_primero()
87              return
88          if posicion == self.tamanio -1:
89              self.eliminar_ultimo()
90              return
91          actual = self.head
92          for _ in range(posicion):
93              actual = actual.siguiente
94          actual.anterior.siguiente = actual.siguiente
95          actual.siguiente.anterior = actual.anterior
96          self.tamanio -= 1
98      def buscar(self, valor):  1 usage  kno4
99          actual = self.head
100         posicion = 0
101         while actual:
102             if actual.dato == valor:
103                 return posicion
104             actual = actual.siguiente
105             posicion += 1
106         print("El valor que busca no se encuentra en la lista")
107         return -1
108
109     def actualizar(self, posicion, nuevo_dato):  1 usage  kno4
110         if posicion < 0 or posicion >= self.tamanio:
111             print("Posición no válida")
112             return False
113         aux = self.head
114         for _ in range(posicion):
115             aux = aux.get_siguiente()
116         aux.set_dato(nuevo_dato)
117         return True
    def transversal(self, direccion='izquierda'):  3 usages  kno4
120         elementos =[]
121         if direccion == 'izquierda':
122             actual = self.head
123             while actual:
124                 elementos.append(actual.dato)
125                 actual = actual.siguiente
126         else:
127             actual = self.tail
128             while actual:
129                 elementos.append(actual.dato)
130                 actual = actual.anterior
131         return elementos
```

Main

```python
def main():  1 usage  kno4
    lista = DoubleLinkedList()

    lista.agregar_al_inicio(50)
    lista.agregar_al_final(60)
    lista.agregar_al_final(65)
    lista.agregar_al_final(70)
    lista.agregar_al_final(80)
    lista.agregar_al_final(90)

    print("Contenido de la lista:", lista.transversal())

    lista.eliminar(2)
    print("Después de eliminar el elemento en la posición 2:", lista.transversal())

    lista.actualizar(3, 88)
    print("Después de actualizar el cuarto elemento a 88:", lista.transversal())

    posicion = lista.buscar(80)
    if posicion != -1:
        print(f"El valor 80 se encuentra en la posición: {posicion}")
    else:
        print("El valor 80 no se encuentra en la lista")


if __name__ == "__main__":
    main()
```

Ejecución

```
Contenido de la lista: [50, 60, 65, 70, 80, 90]
Después de eliminar el elemento en la posición 2: [50, 60, 70, 80, 90]
Después de actualizar el cuarto elemento a 88: [50, 60, 70, 88, 90]
El valor que busca no se encuentra en la lista
El valor 80 no se encuentra en la lista

Process finished with exit code 0
```