Лабораторна робота № 4

Основні структури даних: зв’язаний список, стек, черга

Для спрощення виконання завдань лабораторної роботи, приведемо вихідний код класів, які реалізують частину методів класу зв’язаний список.

```
class Node:
    def __init__(self, cargo=None, next=None):
        self.cargo = cargo
        self.next = next

    def __str__(self):
        return str(self.cargo)

class LinkedList:
    def __init__(self):
        self.length = 0
        self.head = None
        self.tail = None

    def addLast(self, e):
        node = Node(e)
        if self.length == 0:
            self.head = node
            self.tail = node
        else:
            self.tail.next = node
            self.tail = node
        self.length += 1

    def addFirst(self, e):
        node = Node(e)
        if self.length == 0:
            self.head = node
            self.tail = node
        else:
            node.next = self.head
            self.head = node
        self.length += 1

    def add(self, index, e):
        if index < 0 or index > self.length: return
        if self.length == 0:
            self.addLast(e)
            return
        if index == 0:
            self.addFirst(e)
            return
        if index == self.length:
            self.addLast(e)
            return
        node = self.head
        i = 0
        while i < index-1:
            node = node.next
            i += 1
```

```
            newNode = Node(e, node.next)
            node.next = newNode

    def __str__(self):
            node = self.head
            string = ''
            while node:
                string += str(node) + ' '
                node = node.next
            return string

L = LinkedList()
L.addLast(1)
L.addLast(2)
L.addLast(5)
L.addFirst(0)
L.add(0, -1)
L.add(5, 6)
L.add(4, 3)
print L
```

Завдання

1. Реалізувати клас однонаправлений зв'язаний список, який містив би наступні методи:
   addFirst(e) – додає елемент у початок списку
   addLast(e) – додає елемент у кінець списку
   add(index, e) – додає елемент за вказаним індексом
   takeFirst() – видаляє та повертає перший елемент зі списку
   takeLast() – видаляє та повертає останній елемент зі списку
   remove(index) – метод видаляє елемент за індексом та повертає його
   set(index, e) – заміняє значення елемент у списку за вказаним індексом
   get(index) – повертає значення елементу за вказаним індексом
   sort() – виконує сортування списку алгоритмом сортування вставками

2. Реалізувати клас двонаправлений зв'язаний список, який містив би наступні методи:
   addFirst(e) – додає елемент у початок списку
   addLast(e) – додає елемент у кінець списку
   add(index, e) – додає елемент за вказаним індексом
   takeFirst() – видаляє та повертає перший елемент зі списку
   takeLast() – видаляє та повертає останній елемент зі списку
   remove(index) – метод видаляє елемент за індексом та повертає його
   set(index, e) – заміняє значення елемент у списку за вказаним індексом
   get(index) – повертає значення елементу за вказаним індексом
   printReverse() – друкує елементи списку у зворотному порядку

3. Реалізувати клас однонаправлений зв'язаний список, який містив би наступні методи:
   addLast(e) – додає елемент у кінець списку
   takeLast() – видаляє та повертає останній елемент зі списку

   І на основі цього класу реалізувати клас стек (Stack), у середині якого буде використовуватись зв'язаний список

4. Реалізувати клас однонаправлений зв’язаний список, який містив би наступні методи:
   addFirst(e) – додає елемент у початок списку
   takeLast() – видаляє та повертає останній елемент зі списку

   І на основі цього класу реалізувати клас черга (Queue), у середині якого буде використовуватись зв’язаний список

5. На основі двох екземплярів об’єктів класу стек (Stack) реалізувати клас черга (Queue).

6. Використовуючи стек написати перевірку правильності розстановки дужок ({, (, [ ), у деякому рядку. Наприклад “ab(cd{ef}gh[j])”

7. Для математичного виразу представленого як рядок, у вигляді постфіксної нотації (зворотній польський запис), реалізуйте метод для обчислення значення цього виразу з використанням стеку. Операції, які повинні оброблятись: +, -, *, /
   Приклад запису виразу "5 + ((1 + 2) * 4) − 3" у польській нотації: 5 1 2 + 4 * + 3 –

8. На основі приведеного за посиланням алгоритму, реалізуйте перетворення вхідного виразу в постфіксну нотацію(зворотній польський запис)
   http://natalia.appmat.ru/c&c++/postfisso.html
   Приклад запису вхідного виразу "5 + ((1 + 2) * 4) − 3". Результат роботи програми перетворення у постфіксну нотацію: “5 1 2 + 4 * + 3 –“

9. Реалізуйте клас чергу з пріоритетом (PriorityQueue). Пріоритет визначається виходячи з результату порівняння даного елемента з іншими елементами. Наприклад, для чисел пріоритет буде дорівнювати значенню числа

10. На основі зв’язаного списку реалізуйте клас кільцевий буфер (RingBuffer) (http://ru.wikipedia.org/wiki/Кольцевой_буфер). Методи:
    __init__(self, size=1)- задає розмір буферу
    add(e) – додає елемент, якщо в буфері є місце і повертає True, інакше повертає False
    take() – повертає елемент з буфера, якщо він є, інакше повертає None