МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский технический университет связи и информатики»

Кафедра

«Математическая кибернетика и информационные технологии»

Лабораторная работа №4

«Реализация структур данных»

по дисциплине

«Структуры и алгоритмы обработки данных»

Выполнил:

студент гр. БСТ2004

Шадюк М.Р.

Вариант №22

Москва, 2022 г.

Оглавление

[Цель работы 3](#_Toc102096696)

[Задание 1 3](#_Toc102096697)

[КОД 3](#_Toc102096698)

[Выполнение 3](#_Toc102096699)

[Задание 2 3](#_Toc102096700)

[КОД 3](#_Toc102096701)

[Выполнение 3](#_Toc102096702)

[Задание 3 3](#_Toc102096703)

[КОД 4](#_Toc102096704)

[Выполнение 4](#_Toc102096705)

[Задание 4 4](#_Toc102096706)

[Код 4](#_Toc102096707)

[Выполнение 4](#_Toc102096708)

[Здание 5 4](#_Toc102096709)

[Код 4](#_Toc102096710)

[Выполнение 4](#_Toc102096711)

[Задание 6 4](#_Toc102096712)

[Код 4](#_Toc102096713)

[Выполнение 4](#_Toc102096714)

[Задание 7 4](#_Toc102096715)

[Код 5](#_Toc102096716)

[Выполнение 5](#_Toc102096717)

[Задание 8 5](#_Toc102096718)

[Код 5](#_Toc102096719)

[Выполнение 5](#_Toc102096720)

# Цель работы

Реализовать следующие структуры данных:  
Стек (stack)  
Операции для стека: инициализация, проверка на пустоту, добавление нового элемента в начало, извлечение элемента из начала;

Дек (двусторонняя очередь, deque):  
Операции для дека: инициализация, проверка на пустоту, добавление нового элемента в начало, добавление нового элемента в конец, извлечение элемента из начала, извлечение элемента из конца.

Разработать программу обработки данных, содержащихся в заранее подготовленном txt-файле, в соответствии с заданиями, применив указанную в задании структуру данных. Результат работы программы вывести на экран и сохранить в отдельном txt-файле.

# Задание 1

Отсортировать строки файла, содержащие названия книг, в алфавитном порядке с использованием двух деков.

## КОД

from collections import deque

def sortTitles():

    #открываем файл

    f = open('booknames.txt', 'r', encoding="utf-8")

    first = deque([])

    second = deque([])

    for line in f:

         #если первый дек уже не пустой и первая буква названия меньше чем первая буква первого элемента первого дека

        if len(first)>0 and line[0] < first[0][0]:

            first.appendleft(line[:-1])

         #если первый дек пустой либо первая буква названия больше перво буквы последнего элемента дека, то дополняем справа

        elif len(first) == 0 or line[0] > first[-1][0]:

            first.append(line[:-1])

        else: second.append(line[:-1])  #если название должно стоять между крайними элементами первого дека, добавляем во второй

    while len(second) > 0:

        insert(first, second)

    print(first)

def insert(first, second):

        if second[0][0] < first[0][0]:

            first.appendleft(second.pop())  #если первый элемент второго дека меньше первого элемента первого, то дополняем слева

        elif second[0][0] < first[-1][0]:

            second.append(first.pop())  #если последний элемент первого дека меньше первого элемента второго, то перемещаем его в конец второго дека

            insert(first, second)

        else: first.append(second.popleft())

sortTitles()

## Выполнение

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 1 –Результат работы программы

# Задание 2

Дек содержит последовательность символов для шифровки сообщений. Дан текстовый файл, содержащий зашифрованное сообщение. Пользуясь деком, расшифровать текст.

Известно, что при шифровке каждый символ сообщения заменялся следующим за ним в деке по часовой стрелке через один.

## КОД

from collections import deque

#дек символов для шифровки сообщений

key = deque(["a", "b", "c", "d", "e", "f", "g", "h", "j", "k", "l", "m", "n", "o", "p", "q", "r", "s", "t", "u", "v", "w", "x", "y", "z", " "])

def decrypt():

    f = open('zadanie2.txt', 'r', encoding="utf-8")

    res = ""

    for line in f:

        for i in line:

            if i in key: #если символ содержится в деке для шифрования сообщений

                idx = key.index(i)

                if idx == 0: #если это первый элмент, берем предпоследний

                    idx = len(key) - 2

                elif idx == 1: #если это второй элемент дека, берем последний

                    idx = len(key) - 1

                else: #в остальных случаях берем элемент через один назад

                    idx -= 2

                res += key[idx]

            else:

                res+=i

    return res

print(decrypt())

## Выполнение

**Изображение выглядит как текст

Автоматически созданное описание**

Рисунок 2 – Результат

# Задание 3

Даны три стержня и n дисков различного размера. Диски можно надевать на стержни, образуя из них башни. Перенести n дисков со стержня А на стержень С, сохранив их первоначальный порядок. При переносе дисков необходимо соблюдать следующие правила:

- на каждом шаге со стержня на стержень переносить только один диск;

- диск нельзя помещать на диск меньшего размера;

- для промежуточного хранения можно использовать стержень В.

Реализовать алгоритм, используя три стека вместо стержней А, В, С. Информация о дисках хранится в исходном файле.

## КОД

class Node:

    def \_\_init\_\_(self, value):

        self.value = value

        self.next = None

class Stack:

    # конструктор для стэка

    def \_\_init\_\_(self, name):

        self.head = Node("head")

        self.size = 0

        self.name = name

    def \_\_str\_\_(self):

        cur = self.head.next

        out = ""

        while cur:

            out += str(cur.value) + " "

            cur = cur.next

        return out[:-1]

    # получить текущий размер стека

    def getSize(self):

        return self.size

    # проверить стек на пустоту

    def isEmpty(self):

        return self.size == 0

    # достать последний элемент

    def peek(self):

        if self.isEmpty():

            raise Exception("Peeking from an empty stack")

        return self.head.next.value

    # положить элемент наверх

    def push(self, value):

        node = Node(value)

        node.next = self.head.next

        self.head.next = node

        self.size += 1

    # вернуть значение последнего элемента и удалить этот элемент

    def pop(self):

        if self.isEmpty():

            raise Exception("Popping from an empty stack")

        remove = self.head.next

        self.head.next = self.head.next.next

        self.size -= 1

        return remove.value

def TowerOfHanoi(n , source, destination, auxiliary):

    if n==1: #если операция над верхним диском, он перемещается на стержень С

        print ("Перемещен диск 1 со стержня", source.name, "на стержень", destination.name)

        destination.push(source.pop())

        return

    TowerOfHanoi(n - 1, source, auxiliary, destination) #дальше операции над следующими дисками, стержень B как назначение, С как вспомогательный

    print ("Перемещен диск", n, "со стержня", source.name, "на стержень", destination.name)

    destination.push(source.pop())

    TowerOfHanoi(n - 1, auxiliary, destination, source) #когда диски на стержне А кончились, запускаем тот же процесс с стержнем B на стержень С

n = int(input("Сколько дисков должно быть на стержне?"))

base = "\*"

rodA, rodB, rodC = Stack("A"),Stack("B"),Stack("C")

#заполнение стека дисками, где кол-во \* обозначает размер диска

for i in range (n, 0, -1):

    symbol = ""

    j = i

    while j > 0:

        symbol+=base

        j -= 1

    rodA.push(symbol)

TowerOfHanoi(n, rodA, rodC, rodB)

#проверка - достаем элементы из стержня C

for \_ in range(1, n+1):

    remove = rodC.pop()

    print(f"Pop: {remove}")

## Выполнение

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 3 – Результат

# Задание 4

Дан текстовый файл с программой на алгоритмическом языке. За один просмотр файла проверить баланс круглых скобок в тексте, используя стек.

## Код

class Node:

    def \_\_init\_\_(self, value):

        self.value = value

        self.next = None

class Stack:

    # конструктор

    def \_\_init\_\_(self, name):

        self.head = Node("head")

        self.size = 0

        self.name = name

    def \_\_str\_\_(self):

        cur = self.head.next

        out = ""

        while cur:

            out += str(cur.value) + " "

            cur = cur.next

        return out[:-1]

    # получить размер стека

    def getSize(self):

        return self.size

    # проверка на пустоту

    def isEmpty(self):

        return self.size == 0

    # получить последнее значение

    def peek(self):

        if self.isEmpty():

            raise Exception("Peeking from an empty parentheses")

        return self.head.next.value

    # поместить значение в стек

    def push(self, value):

        node = Node(value)

        node.next = self.head.next

        self.head.next = node

        self.size += 1

    # Вернуть последнее знаечние в стеке и удалить его

    def pop(self):

        if self.isEmpty():

            raise Exception("Popping from an empty parentheses")

        remove = self.head.next

        self.head.next = self.head.next.next

        self.size -= 1

        return remove.value

#массивы для скобок

open\_list = ["[","{","("]

close\_list = ["]","}",")"]

# Функция для проверки баланса скобок

def check():

    parentheses = Stack("parentheses")

    f = open('zadanie4.txt', 'r', encoding="utf-8")

    for line in f:

        for i in line:

            if i in open\_list: #если символ содержится в массиве открытых скобок, то добавляем в стек

                parentheses.push(i)

            elif i in close\_list: #если символ содержится в массиве закрытых скобок, то находим, какого типа эта скобка

                pos = close\_list.index(i)

                if ((parentheses.size > 0) and (open\_list[pos] == parentheses.peek())): #и если в стэке уже есть открывающая скобка и она соответствует по индексу закрывающей

                    parentheses.pop() #то удаляем

                else:

                    return "Баланс скобок нарушен"

    if parentheses.isEmpty():

        return "Скобки в порядке"

    else:

        return "Баланс скобок нарушен"

check()

## Выполнение

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 4 – Результат работы программы

# Здание 5

Дан текстовый файл с программой на алгоритмическом языке. За один просмотр файла проверить баланс квадратных скобок в тексте, используя дек.

## Код

from collections import deque

def checkParentheses():

    queue = deque([])

    f = open('zadanie5.txt', 'r', encoding="utf-8")

    for line in f:

        for i in line:

            if str(i) == "[": #если найдена открывающая скобка, дополняем дек

                queue.append(str(i))

            elif str(i) == "]" and len(queue)>0: #если найдена закрывающая скобка и в деке есть открывающая

                queue.popleft()

            elif str(i) == "]" and len(queue) == 0: # если встречена закрывающая при пустом деке, то сразу возвращаем нарушенный баланс

                return "Баланс скобок нарушен"

    if len(queue) > 0:

        return "Баланс скобок нарушен"

    else: return "Скобки в порядке"

checkParentheses()

## Выполнение

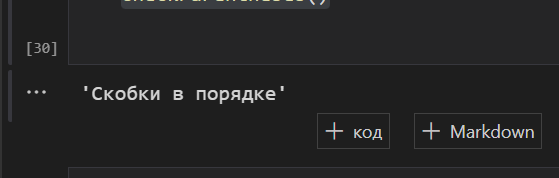


Рисунок 5 – Результат работы программы

# Задание 6

Дан файл из символов. Используя стек, за один просмотр файла напечатать сначала все цифры, затем все буквы, и, наконец, все остальные символы, сохраняя исходный порядок в каждой группе символов.

## Код

class Node:

    def \_\_init\_\_(self, value):

        self.value = value

        self.next = None

class Stack:

    # конструктор

    def \_\_init\_\_(self, name):

        self.head = Node("head")

        self.size = 0

        self.name = name

    def \_\_str\_\_(self): #с помощью второго стека формируем строку в правильном порядке

        cur = Stack("reversed")

        out = ""

        ur = self.head

        while cur.size < self.size:

            cur.push(ur.value)

            ur = ur.next

        while cur.size >1:

            out+=str(cur.pop())

        return out

    # получить размер стека

    def getSize(self):

        return self.size

    # проверка на пустоту

    def isEmpty(self):

        return self.size == 0

    # получить последнее значение

    def peek(self):

        if self.isEmpty():

            raise Exception("Peeking from an empty stack")

        return self.head.next.value

    # поместить значение в стек

    def push(self, value):

        node = Node(value)

        node.next = self.head.next

        self.head.next = node

        self.size += 1

    # Вернуть последнее знаечние в стеке и удалить его

    def pop(self):

        if self.isEmpty():

            raise Exception("Popping from an empty stack")

        remove = self.head.next

        self.head.next = self.head.next.next

        self.size -= 1

        return remove.value

#инициализируем три стека для соответствующих символов

letters, digits, other, = Stack("Letters"), Stack("Digits"), Stack("Others")

def readfile(letters, digits, others):

    f = open('zadanie6.txt', 'r', encoding="utf-8")

    for line in f:

        for i in line:

            if i.isnumeric(): #если символ цифровой, то добавляем его в соответствующий стек

                digits.push(i)

                continue

            if i.isspace() or i.isalpha(): #если символ пробел или буква, то добавляем его в соответствующий стек

                letters.push(i)

                continue

            others.push(i) #если первые два условия не сработали, то добавляем в третий стек

    print(letters)

    print(digits)

    print(others)

readfile(letters, digits, other)

## Выполнение

Изображение выглядит как текст, снимок экрана, электроника

Автоматически созданное описание

Рисунок 6 – Результат работы программы

# Задание 7

Дан файл из целых чисел. Используя дек, за один просмотр файла напечатать сначала все отрицательные числа, затем все положительные числа, сохраняя исходный порядок в каждой группе.

## Код

from collections import deque

def groupNums():

    f = open('zadanie7.txt', 'r', encoding="utf-8")

    queuePos = deque([])

    queueNeg = deque([])

    for x in next(f).split(";"):

         #если положительные, то дополняем соответствующий дек

        queuePos.append(int(x)) if int(x) > 0 else queueNeg.append(int(x))

    queueNeg.extend(queuePos) #склеиваем два дека

    print(queueNeg)

groupNums()

## Выполнение

Изображение выглядит как текст, устройство, датчик

Автоматически созданное описание

Рисунок 7 -- Результат

# Задание 8

Дан текстовый файл. Используя стек, сформировать новый текстовый файл, содержащий строки исходного файла, записанные в обратном порядке: первая строка становится последней, вторая – предпоследней и т.д.

## Код

## Выполнение