САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

ФАКУЛЬТЕТ ИНФОКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет по лабораторной работе № 4

по курсу «Алгоритмы и структуры данных»

Тема: Стек, очередь, связанный список

Вариант 2

Выполнила:

Азизова Наида Элимизовна

К3140

Проверил(а):

Афанасьев А. В.

Санкт-Петербург

2024 г.

Содержание отчёта

[Задания 3](#_Toc185376271)

[Задача №2. Очередь 3](#_Toc185376272)

[Задача №4. Скобочная последовательность Версия 2 5](#_Toc185376273)

[Задача №6. Очередь с минимумом 7](#_Toc185376274)

[Задача №8. Постфиксная запись 9](#_Toc185376275)

[Вывод 11](#_Toc185376276)

# Задания

## Задача №2. Очередь

Изображение выглядит как текст, Шрифт, снимок экрана, алгебра

Автоматически созданное описание

Решение:

import utils  
import os  
  
def queue\_func(commands):  
 queue = []  
 front\_index = 0  
 res = []  
 for command in commands:  
 if command[0] == "+":  
 queue.append(int(command[1]))  
 elif command[0] == "-":  
 res.append(str(queue[front\_index]) + "\n")  
 front\_index += 1  
 return res  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 project\_root = os.path.abspath(os.path.join(os.path.dirname(\_\_file\_\_), '../../..'))  
 input\_file\_path = os.path.join(project\_root, 'lab4', 'task2', 'txtf', 'input.txt')  
 data = utils.read\_input(input\_file\_path)  
 res = queue\_func(data[1:])  
 utils.print\_task\_data(4, 2, data, res)  
  
 project\_root = os.path.abspath(os.path.join(os.path.dirname(\_\_file\_\_), '../../..'))  
 output\_file\_path = os.path.join(project\_root, 'lab4', 'task2', 'txtf', 'output.txt')  
 utils.write\_output(output\_file\_path, res)

Текстовое объяснение решения:

**Функция queue\_func(commands)**:

* **Входные данные**:
  + commands — список команд, каждая команда представлена строкой.
* **Алгоритм**:
  + **Очередь** queue и переменная **индекс** очереди front\_index инициализируются пустыми.
  + Для каждой команды:
    - Если команда вида + x, то число x добавляется в конец очереди.
    - Если команда вида -, то извлекается элемент из начала очереди (по индексу front\_index), и этот элемент добавляется в результат.
  + Возвращается список результатов.

Результат работы кода на примерах из текста задачи:

Ввод:

Изображение выглядит как снимок экрана, часы, число, дизайн

Автоматически созданное описание

Вывод:

Изображение выглядит как снимок экрана, черный, дизайн

Автоматически созданное описание

Вывод по задаче: Я научилась реализовывать работу очереди

## Задача №4. Скобочная последовательность Версия 2

Изображение выглядит как текст, Шрифт, снимок экрана, черно-белый

Автоматически созданное описание

Решение:

import utils  
import os  
  
def check\_brackets(data):  
 stack = []  
 bracket\_pairs = {')': '(', ']': '[', '}': '{'}  
 for i, char in enumerate(data, start=1):  
 if char in "([{":  
 stack.append((char, i))  
 elif char in ")]}":  
 if not stack or stack[-1][0] != bracket\_pairs[char]:  
 return i  
 stack.pop()  
  
 if stack:  
 return stack[0][1]  
  
 return "Success"  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 project\_root = os.path.abspath(os.path.join(os.path.dirname(\_\_file\_\_), '../../..'))  
 input\_file\_path = os.path.join(project\_root, 'lab4', 'task4', 'txtf', 'input.txt')  
 data = utils.read\_input(input\_file\_path)  
 res = check\_brackets(data[0])  
 utils.print\_task\_data(4, 4, data, res)  
  
 project\_root = os.path.abspath(os.path.join(os.path.dirname(\_\_file\_\_), '../../..'))  
 output\_file\_path = os.path.join(project\_root, 'lab4', 'task4', 'txtf', 'output.txt')  
 utils.write\_output(output\_file\_path, res)

Текстовое объяснение решения:

**Функция check\_brackets(data)**:

* **Входные данные**:
  + data — строка, содержащая скобки.
* **Алгоритм**:
  + Для каждой позиции в строке проверяется:
    - Если символ — открывающая скобка ('(', '{', '['), то она помещается в стек.
    - Если символ — закрывающая скобка (')', '}', ']'), то:
      * Проверяется, соответствует ли она последней открытой скобке в стеке.
      * Если соответствует, стек обновляется (удаляется верхний элемент).
      * Если не соответствует или стек пуст — возвращается индекс ошибки.
  + Если по завершении перебора строки в стеке остались открывающие скобки, возвращается индекс первой открывающей скобки.
  + Если все скобки корректно закрыты, возвращается "Success".

Результат работы кода на примерах задачи:

Ввод:

Изображение выглядит как текст, Шрифт, снимок экрана, Графика

Автоматически созданное описание

Вывод:

Изображение выглядит как снимок экрана, черный, дизайн

Автоматически созданное описание

Вывод по задаче: Я научилась осуществлять скобочную последовательность

## Задача №6. Очередь с минимумом

Изображение выглядит как текст, Шрифт, снимок экрана, документ

Автоматически созданное описание

Решение:

import utils  
import os  
  
def queue\_func\_min(commands):  
 queue = []  
 res = []  
 for command in commands:  
 if command[0] == "+":  
 queue.append(int(command[1]))  
 elif command[0] == "-":  
 queue.pop(0)  
 elif command[0] == '?':  
 res.append(str(min(queue)) + "\n")  
 return res  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 project\_root = os.path.abspath(os.path.join(os.path.dirname(\_\_file\_\_), '../../..'))  
 input\_file\_path = os.path.join(project\_root, 'lab4', 'task6', 'txtf', 'input.txt')  
 data = utils.read\_input(input\_file\_path)  
 res = queue\_func\_min(data[1:])  
 utils.print\_task\_data(4, 6, data, res)  
  
 project\_root = os.path.abspath(os.path.join(os.path.dirname(\_\_file\_\_), '../../..'))  
 output\_file\_path = os.path.join(project\_root, 'lab4', 'task6', 'txtf', 'output.txt')  
 utils.write\_output(output\_file\_path, res)

Текстовое объяснение решения:

**Функция queue\_func\_min(commands)**:

* Принимает список команд.
* Обрабатывает команды в цикле:
  + Для команды + x добавляется число x в конец очереди.
  + Для команды - извлекается первый элемент из очереди.
  + Для команды ? вычисляется минимальное значение в очереди с помощью min(), и это значение добавляется в список res.
* Функция возвращает список результатов запросов минимальных элементов.

Результат работы кода на примерах задачи:

Ввод:

Изображение выглядит как снимок экрана, текст, часы, дизайн

Автоматически созданное описание

Вывод:

Изображение выглядит как снимок экрана, часы, число, дизайн

Автоматически созданное описание

Вывод по задаче: Я научилась реализовывать работу очереди с минимумом

## Задача №8. Постфиксная запись

Изображение выглядит как текст, снимок экрана, Шрифт, документ

Автоматически созданное описание

Решение:

import utils  
import os  
  
def postfix(expression):  
 stack = []  
 for token in expression:  
 if token.isdigit():  
 stack.append(int(token))  
 else:  
 b = stack.pop()  
 a = stack.pop()  
 if token == '+':  
 stack.append(a + b)  
 elif token == '-':  
 stack.append(a - b)  
 elif token == '\*':  
 stack.append(a \* b)  
 return stack.pop()  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 project\_root = os.path.abspath(os.path.join(os.path.dirname(\_\_file\_\_), '../../..'))  
 input\_file\_path = os.path.join(project\_root, 'lab4', 'task8', 'txtf', 'input.txt')  
 data = utils.read\_input(input\_file\_path)  
 res = postfix(data[1])  
 utils.print\_task\_data(4, 8, data, res)  
  
 project\_root = os.path.abspath(os.path.join(os.path.dirname(\_\_file\_\_), '../../..'))  
 output\_file\_path = os.path.join(project\_root, 'lab4', 'task8', 'txtf', 'output.txt')  
 utils.write\_output(output\_file\_path, res)

Текстовое объяснение решения:

**Функция postfix(expression)**:

* Принимает список expression, который состоит из элементов, разделённых пробелами (чисел и операторов).
* Если элемент — это число (проверяется с помощью isdigit()), то оно добавляется в стек.
* Если элемент — оператор, то из стека извлекаются два числа, выполняется операция и результат снова кладется в стек.
* В конце остаётся одно число в стеке, которое и является результатом.

Результат работы кода на примерах задачи:

Ввод:

Изображение выглядит как часы, текст, снимок экрана, Шрифт

Автоматически созданное описание

Вывод:

Изображение выглядит как снимок экрана, текст, Шрифт, часы

Автоматически созданное описание

Вывод по задаче: алгоритм эффективно находит значение выражения, заданного в постфиксной форме.

# Вывод

В этой лабораторной работе я научилась основным структурам данных: стек, очередь и связанный список. Я рассмотрела их принципы работы и особенности реализации.