МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА № 42

ОТЧЁТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

| старший преподаватель |  |  |  | С.Ю. Гуков |
| --- | --- | --- | --- | --- |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

| ОТЧЁТ О ЛАБОРАТОРНОЙ РАБОТЕ №4 |
| --- |
| Стек с поддержкой максимума |
| по курсу: |
| АЛГОРИТМЫ И СТРУКТУРЫ ДАННЫХ |
|  |

РАБОТУ ВЫПОЛНИЛ

| СТУДЕНТ гр. № | 4329 |  |  |  | Д.С. Шаповалова |
| --- | --- | --- | --- | --- | --- |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2024

ОГЛАВЛЕНИЕ

[Цель работы 3](#_30j0zll)

[Постановка задачи 3](#_1fob9te)

[Схема алгоритма решения 4](#_3znysh7)

[Полное описание реализованной функции 4](#_gqc1nhfmq022)

[Листинг программы 5](#_2et92p0)

[Результат выполнения программы. 5](#_tyjcwt)

[ВЫВОДЫ 6](#_3dy6vkm)

# Цель работы

Реализовать стек с поддержкой операций push, pop и max.

# Постановка задачи

Задание: Написать программ, реализующую стек с поддержкой операций push, pop и max. Необходимо использовать два любых дополнительных усложнения программы из представленных. Конкретный текст задания приведён в таблице 1.

Таблица 1. Индивидуальное задание

| № | Текст задания | Вход | Выход |
| --- | --- | --- | --- |
| 0 | Реализовать стек с поддержкой операций push, pop и max. Push v - впихивает значение в стек, pop - вытягивает последнее значение из стека, max - считывает максимальное значение в стеке.  Ограничения. 1 ≤ q ≤ 400 000, 0 ≤ v ≤ 100 000.  Два дополнительных усложнения:   1. Поддержка операций min и среднее значение (avg). Добавить операцию min, которая возвращает минимальное значение на стеке. Добавить операцию avg, которая возвращает среднее значение элементов на стеке. Эта операция также должна работать за константное время. 2. Ограничение размера стека. Ввести ограничение на максимальный размер стека. Если стек заполнен, операции push должны игнорироваться или возвращать ошибку.   (ещё могли быть удаление по значению и итерация с фильтрацией) | Первая строка содержит число запросов q. Каждая из последующих q строк задаёт запрос в одном из следующих форматов: push v, pop, max. | Для каждого запроса max выведите (в отдельной строке) текущий максимум на стеке. |

# Схема алгоритма решения

1. Создаём класс MaxStack
2. Ввод данных (размер стека задаём в коде)
3. Построчно считываем команды
4. Записываем результаты в txt файл

# Полное описание реализованной функции

**Класс MaxStack:**

Конструктор - инициализация переменных

*Метод push*:

Проверяем, не превышен ли максимальный размер стека max\_size. Если превышен, функция возвращает строку "stack overflow" и завершает выполнение. Если размер стека в рамках лимита, мы продолжаем добавление элемента. Если стек для максимумов пустой или новое значение больше либо равно текущему максимуму, добавляем его в max\_stack. Аналогично для минимумов.

Метод pop:

Проверяем, не пустой ли стек, прежде чем пытаться удалить элемент.

Используем метод pop для удаления и сохранения значения верхнего элемента value. Обновляем total\_sum, вычитая удаленное значение, и уменьшаем count.

Если удаленный элемент равен текущему максимальному значению, удаляем его из max\_stack. Аналогично, если удаленный элемент — это минимальное значение, удаляем его из min\_stack.

get\_max(): Возвращает верхний элемент из max\_stack, если он не пустой, иначе возвращает None.

get\_avg(): Возвращает среднее значение. Если в стеке нет элементов (count равен 0), возвращает None, иначе делит total\_sum на count.

get\_min(): Возвращает верхний элемент из min\_stack, если он не пуст, иначе возвращает None.

**Main:** Считываем все вводимые данные сразу, делим их на строки и убираем лишние пробелы. Инициализируем переменные - количество операций и размер стека. Создаем экземпляр MaxStack, передавая в него размер стека.

**Обработка запросов:**

Создаем список results, который будет хранить ответы на запросы.

Перебираем все запросы. Команда разбивается на части (command), чтобы определить, какая операция должна быть выполнена.

Если команда push, извлекаем значение из строки и вызываем метод push. Если вернулся overflow\_message, добавляем его в список результатов. Обрабатываем команды pop, max, avg, и min, аналогично команде push, вызывая соответствующие методы и добавляя результаты в список.

**Запись в txt:**

Открываем файл output.txt в режиме записи ('w').

Записываем все результаты, которые были собраны в списке results, в файл. Используем метод join для создания строки с разделителем в виде новой строки.

# Листинг программы

class MaxStack:

def \_\_init\_\_(self, max\_size):

self.stack = []

self.max\_stack = []

self.total\_sum = 0

self.count = 0

self.min\_stack = []

self.max\_size = max\_size # Максимальный размер стека

def push(self, value):

if self.count >= self.max\_size:

return "stack overflow" # Не добавляем, если превышен размер стека

self.stack.append(value)

self.total\_sum += value

self.count += 1

if not self.max\_stack or value >= self.max\_stack[-1]:

self.max\_stack.append(value)

if not self.min\_stack or value <= self.min\_stack[-1]:

self.min\_stack.append(value)

def pop(self):

if self.stack:

value = self.stack.pop()

self.total\_sum -= value

self.count -= 1

if value == self.max\_stack[-1]:

self.max\_stack.pop()

if value == self.min\_stack[-1]:

self.min\_stack.pop()

def get\_max(self):

return self.max\_stack[-1] if self.max\_stack else None

def get\_avg(self):

if self.count == 0:

return None

return self.total\_sum / self.count

def get\_min(self):

return self.min\_stack[-1] if self.min\_stack else None

def main():

import sys

input = sys.stdin.read

data = input().strip().splitlines()

q = int(data[0])

max\_size = 3 # Размер стека

max\_stack = MaxStack(max\_size)

results = []

for i in range(1, q + 1):

command = data[i].split()

if command[0] == 'push':

value = int(command[1])

overflow\_message = max\_stack.push(value)

if overflow\_message:

results.append(overflow\_message) # Добавляем сообщение о переполнении

elif command[0] == 'pop':

max\_stack.pop()

elif command[0] == 'max':

current\_max = max\_stack.get\_max()

if current\_max is not None:

results.append(str(current\_max))

elif command[0] == 'avg':

current\_avg = max\_stack.get\_avg()

if current\_avg is not None:

results.append(f"{current\_avg:.6f}") # Форматирование до 6 знаков после запятой

elif command[0] == 'min':

current\_min = max\_stack.get\_min()

if current\_min is not None:

results.append(str(current\_min))

# Запись результатов в файл

with open('output.txt', 'w') as f:

f.write('\n'.join(results) + '\n')

if \_\_name\_\_ == '\_\_main\_\_':

main()

# 

# Результат выполнения программы.

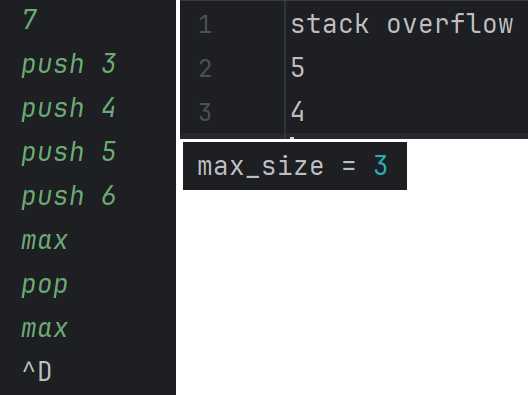


Рисунок 1 - 1-ый результат работы программы

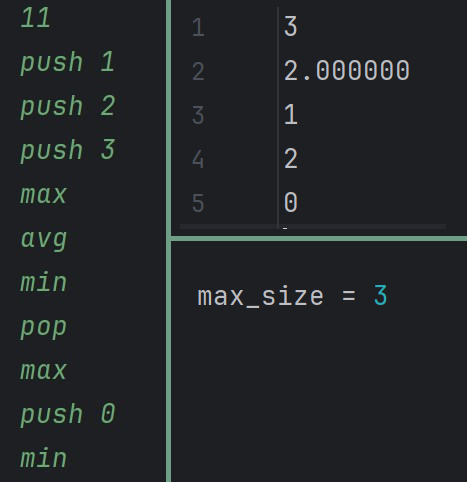


Рисунок 2 - 2-рой результат работы программы

# 

# Выводы

В ходе выполнения лабораторной работы мной были освоены и изучены: понятие стек, операции: push, pop, max; операции: min, avg; размер стека и переполнение. Написанная программа была протестирована, полученный результат соответствует значению в примере.