ГУАП

КАФЕДРА № 42

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

доцент		С. А. Чернышев
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ № 4

СТЕК С ПОДДЕРЖКОЙ МАКСИМУМА

по курсу:

АЛГОРИТМЫ И СТРУКТУРЫ ДАННЫХ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ гр. № 4326		Г. С. Томчук
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2024

СОДЕРЖАНИЕ

1 Цель работы 3
2 Задание 3
3 Краткое описание хода разработки 3
4 Исходный код программы 4
5 Реализация основного и двух любых дополнительных заданий из списка 5
6 Результаты работы программы с примерами 6
7 Выводы 7

1 Цель работы

Целью данной лабораторной работы является реализовать стек с поддержкой операций push, pop, max и двух дополнительных операций: min и avg.

2 Задание

- Реализовать стек, поддерживающий операции:
  - push v — добавление элемента со значением v;
  - pop — удаление верхнего элемента;
  - max — получение максимального элемента.
- Добавить поддержку операций min (получение минимального значения) и avg (среднего значения всех элементов).
- Ввести ограничение на максимальный размер стека. Если стек заполнен, операция push должна выдавать ошибку.

3 Краткое описание хода разработки

Разработка началась с определения требований к стеку и операций, которые он должен поддерживать. Основная задача заключалась в обеспечении быстродействия операций push, pop, max, min и avg, каждая из которых должна выполняться за константное время O(1).

Для этого был разработан класс, в котором хранились:

1. Основной стек — для хранения всех добавленных элементов.
2. Вспомогательные стеки:
   - max_stack для отслеживания текущего максимального элемента;
   - min_stack для отслеживания текущего минимального элемента.
3. Сумма элементов для быстрой реализации операции avg.

Сначала реализованы базовые операции:

1. push добавляет элемент в основной стек и при необходимости обновляет вспомогательные стеки и сумму.
2. pop удаляет элемент с вершины основного стека и корректирует связанные значения.
3. max, min и avg возвращают соответствующие значения из

вспомогательных структур.

После этого стек был дополнен ограничением на максимальный размер. Для этого перед выполнением операции push добавлена проверка количества элементов. Если стек заполнен, операция возвращает сообщение об ошибке.

На завершающем этапе программа была протестирована на нескольких наборах входных данных для проверки корректности всех операций.

4 Исходный код программы

```
class MaxStack:
    def __init__(self, max_size=None):
        self.stack = []
        self.max_stack = []
        self.min_stack = []
        self.sum = 0
        self.max_size = max_size

    def push(self, value):
        if self.max_size is not None and len(self.stack) >= self.max_size:
            print("Стек переполнен!")
            return

        self.stack.append(value)
        self.sum += value

        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 not self.stack:
            print("Стек пуст!")
            return

        value = self.stack.pop()
        self.sum -= value

        if value == self.max_stack[-1]:
            self.max_stack.pop()

        if value == self.min_stack[-1]:
            self.min_stack.pop()

    def max(self):
        if not self.max_stack:
            print("Стек пуст!")
            return None
        return self.max_stack[-1]

    def min(self):
        if not self.min_stack:
            print("Стек пуст!")
```

```
            return None
        return self.min_stack[-1]

    def avg(self):
        if not self.stack:
            print("Стек пуст!")
            return None
        return self.sum / len(self.stack)


def process_requests(requests, max_stack):
    for request in requests:
        parts = request.split()
        command = parts[0]

        if command == "push":
            value = int(parts[1])
            max_stack.push(value)
        elif command == "pop":
            max_stack.pop()
        elif command == "max":
            result = max_stack.max()
            if result is not None:
                print(result)
        elif command == "min":
            result = max_stack.min()
            if result is not None:
                print(result)
        elif command == "avg":
            result = max_stack.avg()
            if result is not None:
                print(result)


if __name__ == "__main__":
    q = int(input(""))
    requests = []

    for _ in range(q):
        requests.append(input())

    max_stack = MaxStack(max_size=10)
    process_requests(requests, max_stack)
```

5 Реализация основного и двух любых дополнительных заданий из списка

Для работы с max был создан вспомогательный стек max_stack, который обновляется при каждой операции push или pop. Это позволяет быстро и эффективно определять максимальное значение.

Дополнительное задание 1: Поддержка операций min и avg. Добавлены две дополнительные операции:

- min возвращает минимальное значение на стеке. Для её реализации

используется вспомогательный стек min_stack, который обновляется аналогично max_stack.

- avg возвращает среднее значение всех элементов. Чтобы обеспечить O(1) для этой операции, введена переменная для хранения суммы всех элементов. Эта сумма обновляется при каждом добавлении или удалении элемента.

Дополнительное задание 2: Ограничение размера стека. Введено ограничение на максимальный размер стека. Если стек заполнен, то при выполнении операции push генерируется сообщение об ошибке, а элемент не добавляется. Это позволяет избежать переполнения структуры и некорректного поведения программы.

6 Результаты работы программы с примерами

На рис. 1, 2, 3 изображено тестирование написанной программы с различными входными данными.

```
5
push 2
push 1
max
pop
max
2
2

Process finished with exit code 0
```

Рисунок 1

```
3
push 1
push 7
pop

Process finished with exit code 0
```

Рисунок 2

```
7
push 7
push 2
push 10
max
avg
min
push 23
10
6.333333333333333
2
Стек переполнен!

Process finished with exit code 0
```

Рисунок 3

7 Выводы

1. Цель работы достигнута. Реализован стек с поддержкой всех указанных операций, включая push, pop, max, а также двух дополнительных: min и avg.
2. Производительность. Все операции, включая дополнительные, выполняются за константное время O(1), что соответствует требованиям.
3. Гибкость и расширяемость. Введение ограничения на размер стека добавляет надёжности, а вспомогательные структуры позволяют легко добавить новые операции, такие как отслеживание медианы

или частоты элементов.

4. Применимость. Реализованный стек может быть использован в различных задачах, где требуется не только стандартное поведение стека, но и доступ к максимальным, минимальным или средним значениям за минимальное время.
5. Тестирование. Программа успешно протестирована на различных наборах входных данных, включая граничные случаи (например, пустой стек, переполнение). Все результаты подтвердили корректность работы.