

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 14

Выполнила:  
Рудникова Виктория Олеговна  
К3125

Проверила:  
Артамонова В.Е.

Санкт-Петербург  
2022 г.

## Содержание отчета

<b>Содержание отчета</b>	<b>2</b>
<b>Задачи по варианту</b>	<b>3</b>
Задача №2. Очередь	3
Задача №3. Скобочная последовательность. Версия 1	5
Задача №5. Стек с максимумом	9
Задача №8. Постфиксная запись	15
<b>Дополнительные задачи</b>	<b>18</b>
Задача №1. Стек	18
Задача №6. Очередь с минимумом	21
<b>Вывод:</b>	<b>24</b>

## Задачи по варианту

### Задача №2. Очередь

Реализуйте работу очереди. Для каждой операции изъятия элемента выведите ее результат.

На вход программе подаются строки, содержащие команды. Каждая строка содержит одну команду. Команда — это либо «+ N», либо «-».

Команда «+N» означает добавление в очередь числа N, по модулю не превышающего  $10^9$ .

Команда «-» означает изъятие элемента из очереди. Гарантируется, что размер очереди в процессе выполнения команд не превысит  $10^6$  элементов.

- Формат входного файла (input.txt). В первой строке содержится M ( $1 \leq M \leq 10^6$ ) – число команд. В последующих строках содержатся команды, по одной в каждой строке.
- Формат выходного файла (output.txt). Выведите числа, которые удаляются из очереди с помощью команды «-», по одному в каждой строке. Числа нужно выводить в том порядке, в котором они были извлечены из очереди. Гарантируется, что извлечения из пустой очереди не производится.
- Ограничение по времени. 2 сек.
- Ограничение по памяти. 256 мб.

```
import time
import os
import psutil
process = psutil.Process(os.getpid())
t_start = time.perf_counter()

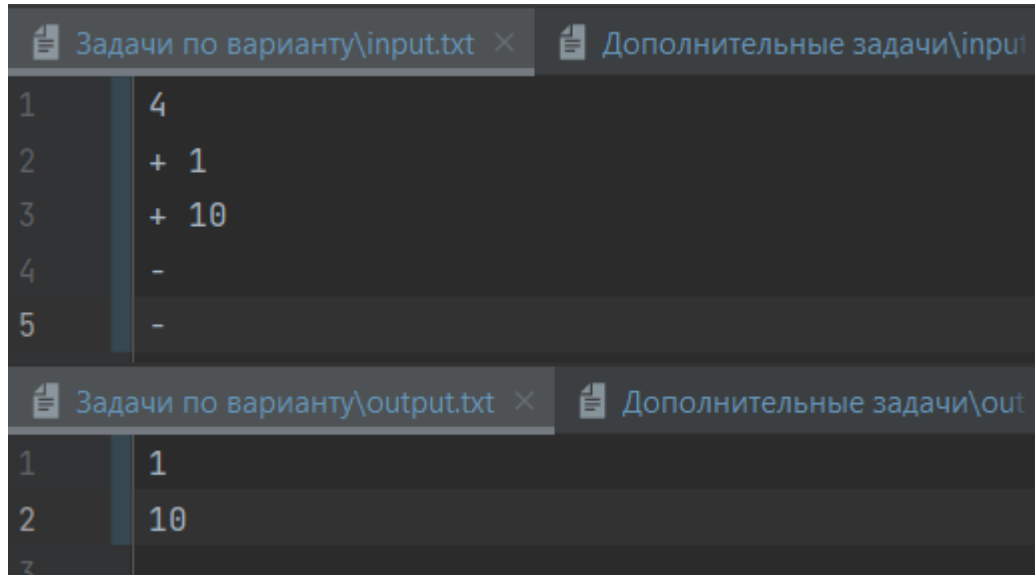
stack = []
arr = []

with open('input.txt', 'r') as f:
    for i, line in enumerate(f):
        if i > 0:
            if line[0] == "+":
                arr.append(line[2:-1])
            else:
                x = arr.pop(0)
                stack.append(x)
        print(arr)
print(stack)
with open('output.txt', 'w') as x:
    x.writelines("%s\n" % line for line in stack)
```

```
print("время работы", (time.perf_counter() - t_start), "секунд")
print('память', process.memory_info().rss / 1024 ** 2, 'mb')
```

Реализована очередь, то есть структура данных, которая хранит элементы, но элементы добавляются только в конец, а удаляются только из начала.

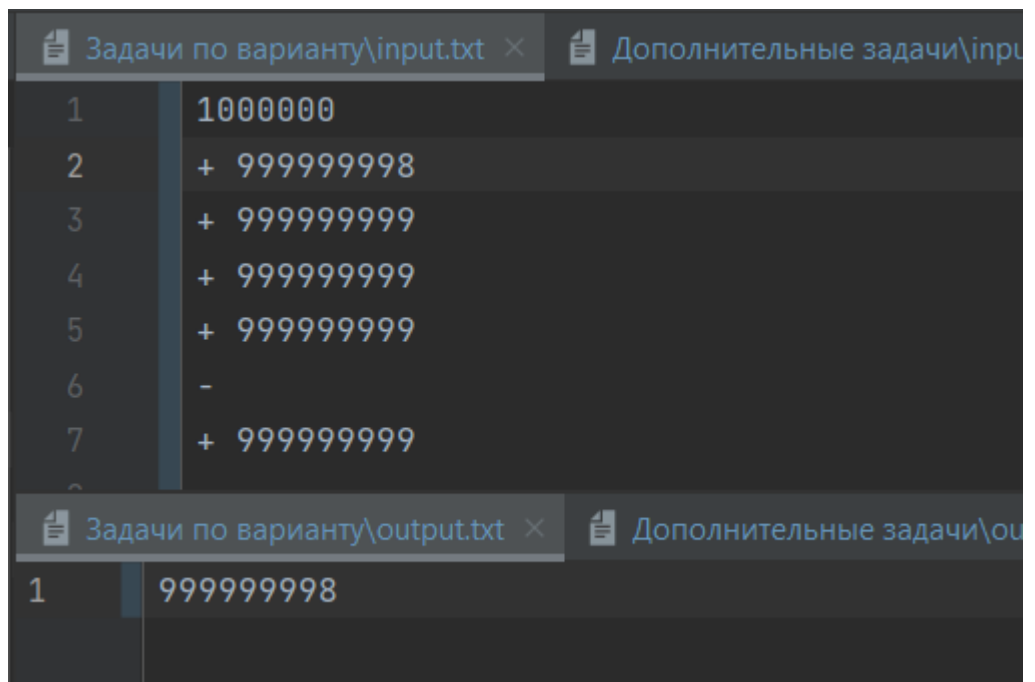
Результат работы кода на примерах из текста задачи:



The screenshot shows a code editor with two tabs: "Задачи по варианту\input.txt" and "Дополнительные задачи\input". The first tab is active and shows a list of operations: 1: 4, 2: + 1, 3: + 10, 4: -, 5: -. Below it, the output file "Задачи по варианту\output.txt" is shown with the results: 1: 1, 2: 10.

Line	Input	Output
1	4	1
2	+ 1	10
3	+ 10	
4	-	
5	-	

Результат работы кода на максимальных и минимальных значениях:



The screenshot shows a code editor with two tabs: "Задачи по варианту\input.txt" and "Дополнительные задачи\input". The first tab is active and shows a list of operations: 1: 1000000, 2: + 999999998, 3: + 999999999, 4: + 999999999, 5: + 999999999, 6: -, 7: + 999999999. Below it, the output file "Задачи по варианту\output.txt" is shown with the result: 1: 999999998.

Line	Input	Output
1	1000000	999999998
2	+ 999999998	
3	+ 999999999	
4	+ 999999999	
5	+ 999999999	
6	-	
7	+ 999999999	

```

Задачи по варианту\input.txt
1 2
2 + 1
3 -

Задачи по варианту\output.txt
1 1
2

```

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.109375 sec	17.244140625 KB
Пример из задачи	0.109375 sec	17.244140625 KB
Верхняя граница диапазона значений входных данных из текста задачи	0.125 sec	17.244140625 KB

Вывод по задаче: в задаче я реализовала очередь.

### Задача №3. Скобочная последовательность. Версия 1

Последовательность  $A$ , состоящую из символов из множества «(», «)», «[» и «]», назовем правильной скобочной последовательностью, если выполняется одно из следующих утверждений:

- $A$  – пустая последовательность;
- первый символ последовательности  $A$  – это «(», и в этой последовательности существует такой символ «)», что последовательность можно представить как  $A = (B)C$ , где  $B$  и  $C$  – правильные скобочные последовательности;
- первый символ последовательности  $A$  – это «[», и в этой последовательности существует такой символ «]», что последовательность

можно представить как  $A = (B)C$ , где  $B$  и  $C$  – правильные скобочные последовательности.

Так, например, последовательности « $(( ))$ » и « $()[]$ » являются правильными скобочными последовательностями, а последовательности « $[]$ » и « $(( ($ » таковыми не являются.

Входной файл содержит несколько строк, каждая из которых содержит последовательность символов « $($ », « $)$ », « $[$ » и « $]$ ». Для каждой из этих строк выясните, является ли она правильной скобочной последовательностью.

- Формат входного файла (input.txt). Первая строка входного файла содержит число  $N$  ( $1 \leq N \leq 500$ ) – число скобочных последовательностей, которые необходимо проверить. Каждая из следующих  $N$  строк содержит скобочную последовательность длиной от 1 до  $10 \cdot 4$  включительно. В каждой из последовательностей присутствуют только скобки указанных выше видов.

- Формат выходного файла (output.txt). Для каждой строки входного файла (кроме первой, в которой записано число таких строк) выведите в выходной файл «YES», если соответствующая последовательность является правильной скобочной последовательностью, или «NO», если не является.

- Ограничение по времени. 2 сек.

- Ограничение по памяти. 256 мб.

```
from collections import deque
import time
import os
import psutil
process = psutil.Process(os.getpid())
t_start = time.perf_counter()

brackets = deque()
arr = []
stack = []
with open('input.txt', 'r') as f:
    num = int(f.readline())
    for _ in range(num):
        arr.append(f.readline())
for seq in arr:
    for i in seq:
        if i == '(' or i == '[':
            brackets.append(i)
        elif len(brackets) == 0 and (i == ')' or i == ']'):
            brackets.append('ERROR')
            break
```

```

        elif i == ')' and brackets[len(brackets) - 1] == '(' or i == ']' and
brackets[len(brackets) - 1] == '[':
            brackets.pop()
        if len(brackets) == 0:
            stack.append('YES')
        elif len(brackets) != 0 and 'ERROR' not in stack:
            stack.append('NO')
        brackets.clear()

with open('output.txt', 'w') as g:
    g.write('\n'.join(stack))

print("время работы", (time.perf_counter() - t_start), "секунд")
print('память', process.memory_info().rss / 1024 ** 2, 'mb')

```

Алгоритм проверки на правильную скобочную последовательность реализован с помощью стека. Когда в строке встречается открытая скобка, она добавляется в стек, когда встречена соответствующая ей закрытая скобка, открытая удаляется из стека.

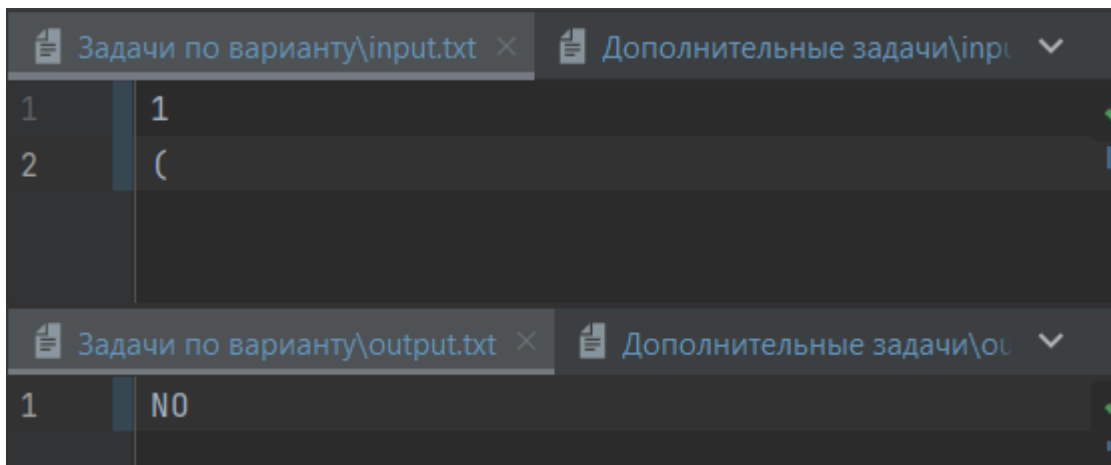
Результат работы кода на примерах из текста задачи:

Line	Input	Output
1	5	YES
2	()()	YES
3	([])	NO
4	([])	NO
5	(([])	NO
6	)()	

Результат работы кода на максимальных и минимальных значениях:







	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.109375 sec	17.244140625 KB
Пример из задачи	0.109375 sec	17.244140625 KB
Верхняя граница диапазона значений входных данных из текста задачи	0.109375 sec	19.87890625 KB

Вывод по задаче: в этой задаче я реализовала алгоритм проверки на правильную скобочную последовательность с помощью стека.

### Задача №5. Стек с максимумом

Стек - это абстрактный тип данных, поддерживающий операции Push() и Pop(). Нетрудно реализовать его таким образом, чтобы обе эти операции работали за константное время. В этой задаче ваша цель - реализовать стек, который также поддерживает поиск максимального значения и гарантирует, что все операции по-прежнему работают за константное время.

Реализуйте стек, поддерживающий операции Push(), Pop() и Max().

- Формат входного файла (input.txt). В первой строке входного файла со-

держится  $n$  ( $1 \leq n \leq 400000$ ) – число команд. Последующие  $n$  строк исходного файла содержит ровно одну команду: push  $V$ , pop или max.  $0 \leq V \leq 10^5$ .

- Формат выходного файла (output.txt). Для каждого запроса max выведите (в отдельной строке) максимальное значение стека.
- Ограничение по времени. 5 сек.
- Ограничение по памяти. 512 мб.

```
import time
import os
import psutil
process = psutil.Process(os.getpid())
t_start = time.perf_counter()

stack = []
arr = []

with open('input.txt', 'r') as f:
    for i, line in enumerate(f):
        if i > 0:
            if "push" in line:
                arr.append(int(line.replace('push ', '')[:-1]))
                print(arr)
            elif "pop" in line:
                if len(arr) == 0:
                    continue
                del arr[-1]
                print(arr)
            elif "max" in line:
                if len(arr) == 0:
                    continue
                stack.append(max(arr))

print(stack)
with open('output.txt', 'w') as x:
    x.writelines("%s\n" % line for line in stack)
print("время работы", (time.perf_counter() - t_start), "секунд")
print("память", process.memory_info().rss / 1024 ** 2, 'mb')
```

Реализован стек с операциями добавления и удаления элемента, а также поиск максимального элемента.

Результат работы кода на примерах из текста задачи:

```
Задачи по варианту\input.txt x  Дополнительные задачи\input.txt v
1      5
2      push 2
3      push 1
4      max
5      pop
6      max

Задачи по варианту\output.txt x  Дополнительные задачи\output.txt v
1      2
2      2
3
```

```
Задачи по варианту\input.txt x  Дополнительные задачи\input.txt v
1      5
2      push 1
3      push 2
4      max
5      pop
6      max

Задачи по варианту\output.txt x  Дополнительные задачи\output.txt v
1      2
2      1
3
```

```
Задачи по варианту\input.txt x  Дополнительные задачи\input.txt v
1 3
2 push 1
3 push 7
4 pop

Задачи по варианту\output.txt x  Дополнительные задачи\output.txt v
1 |
```

```
Задачи по варианту\input.txt x  Дополнительные задачи\input.txt v
1 10
2 push 2
3 push 3
4 push 9
5 push 7
6 push 2
7 max
8 max
9 max
10 pop
11 max

Задачи по варианту\output.txt x  Дополнительные задачи\output.txt v
1 9
2 9
3 9
4 9
5
```

Задачи по варианту\input.txt		Дополнительные задачи\input	
1	6		
2	push 7		
3	push 1		
4	push 7		
5	max		
6	pop		
7	max		
Задачи по варианту\output.txt		Дополнительные задачи\ou	
1	7		
2	7		

Результат работы кода на максимальных и минимальных значениях:

Задачи по варианту\input.txt

Дополнительные задачи\input

1

400000

2

push 99999

3

push 99998

4

push 99997

5

push 99998

6

push 99999

7

push 99994

8

push 99999

9

max

Задачи по варианту\output.txt

Дополнительные задачи\ou

1

99999

2

99999

3

99999

4

99999

5

99999

6

99999

7

99999

8

99999

Задачи по варианту\input.txt

Дополнительные задачи\input

1

2

2

push 1

3

max

Задачи по варианту\output.txt

Дополнительные задачи\ou

1

1

2

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.109375 sec	17.244140625 KB

Пример из задачи	0.109375 sec	17.244140625 KB
Пример из задачи	0.109375 sec	17.244140625 KB
Пример из задачи	0.109375 sec	17.244140625 KB
Пример из задачи	0.109375 sec	17.244140625 KB
Пример из задачи	0.109375 sec	17.244140625 KB
Верхняя граница диапазона значений входных данных из текста задачи	0.109375 sec	17.244140625 KB

Вывод по задаче: в задаче я реализовала стек с поддержкой максимума.

### Задача №8. Постфиксная запись

В постфиксной записи (или обратной польской записи) операция записывается после двух операндов. Например, сумма двух чисел  $A$  и  $B$  записывается как  $A B +$ . Запись  $B C + D *$  означает привычное нам  $(B + C) * D$ , а запись  $A B C + D * +$  означает  $A + (B + C) * D$ . Достоинство постфиксной записи в том, что она не требует скобок и дополнительных соглашений о приоритете операторов для своего чтения.

Дано выражение в обратной польской записи. Определите его значение.

- Формат входного файла (input.txt). В первой строке входного файла дано число  $N$  ( $1 \leq n \leq 10^6$ ) – число элементов выражения. Во второй строке содержится выражение в постфиксной записи, состоящее из  $N$  элементов. В выражении могут содержаться неотрицательные однозначные числа и операции  $+$ ,  $-$ ,  $*$ . Каждые два соседних элемента выражения разделены ровно одним пробелом.
- Формат выходного файла (output.txt). Необходимо вывести значение записанного выражения. Гарантируется, что результат выражения, а также результаты всех промежуточных вычислений, по модулю будут меньше, чем  $2^{31}$ .
- Ограничение по времени. 2 сек.
- Ограничение по памяти. 256 мб.

```
import time
import os
```

```

import psutil

process = psutil.Process(os.getpid())
t_start = time.perf_counter()

stack = []
c = 0

with open('input.txt', 'r') as f:
    for i in range(1):
        f.readline()
        stack = f.readline()
        stack = (list(map(str, stack.split()))))

    while len(stack) != 1:
        if stack[c] == '+':
            summ = int(stack[c - 2]) + int(stack[c - 1])
            del stack[c - 2:c + 1]
            stack.insert(c - 2, str(summ))
            c = 0
        if stack[c] == '-':
            diff = int(stack[c - 2]) - int(stack[c - 1])
            del stack[c - 2:c + 1]
            stack.insert(c - 2, str(diff))
            c = 0
        if stack[c] == '*':
            prod = int(stack[c - 2]) * int(stack[c - 1])
            del stack[c - 2:c + 1]
            stack.insert(c - 2, str(prod))
            c = 0
        print(c)
        print(stack)
        c += 1

with open('output.txt', 'w') as x:
    x.writelines("%s\n" % line for line in stack)

print("время работы", (time.perf_counter() - t_start), "секунд")
print('память', process.memory_info().rss / 1024 ** 2, 'mb')

```

Реализован “польский калькулятор” на основе стека. Если встречено число, оно добавляется в стек. Если же операция, то из стека удаляются два числа, а затем добавляется результат операции.

Результат работы кода на примерах из текста задачи:



```

Задачи по варианту\input.txt x  Дополнительные задачи
1 7
2 8 9 + 1 7 - *

Задачи по варианту\output.txt x  Дополнительные задачи
1 -102

```

Результат работы кода на максимальных и минимальных значениях:

```

Задачи по варианту\input.txt x  Дополнительные задачи\input.txt v
1 1000000
2 1 1 + 8 9 * + 6 2 + * 1 0 * * 4 5 * * 7 3 + * 8

Задачи по варианту\output.txt x  Дополнительные задачи\output.txt v
1 0

```

```

Задачи по варианту\input.txt x  Дополнительные задачи\input.txt v
1 3
2 2 7 +

Задачи по варианту\output.txt x  Дополнительные задачи\output.txt v
1 9

```

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.109375 sec	17.244140625 KB
Пример из задачи	0.109375 sec	17.244140625 KB
Верхняя граница диапазона значений входных данных из	0.109375 sec	17.244140625 KB

текста задачи		
---------------	--	--

Вывод по задаче: в задаче я реализовала польский калькулятор на основе стека.

## Дополнительные задачи

### Задача №1. Стек

Реализуйте работу стека. Для каждой операции изъятия элемента выведите ее результат.

На вход программе подаются строки, содержащие команды. Каждая строка содержит одну команду. Команда — это либо “+ N”, либо “-”. Команда “+ N” означает добавление в стек числа N, по модулю не превышающего  $10^9$ . Команда “-” означает изъятие элемента из стека. Гарантируется, что не происходит извлечения из пустого стека. Гарантируется, что размер стека в процессе выполнения команд не превысит  $10^6$  элементов.

- Формат входного файла (input.txt). В первой строке входного файла содержится M ( $1 \leq M \leq 10^6$ ) — число команд. Каждая последующая строка исходного файла содержит ровно одну команду.
- Формат выходного файла (output.txt). Выведите числа, которые удаляются из стека с помощью команды “-”, по одному в каждой строке. Числа нужно выводить в том порядке, в котором они были извлечены из стека. Гарантируется, что изъятий из пустого стека не производится.

- Ограничение по времени. 2 сек.
- Ограничение по памяти. 256 мб.

```
import time
import os
import psutil
process = psutil.Process(os.getpid())
t_start = time.perf_counter()

stack = []
arr = []

with open('input.txt', 'r') as f:
    for i, line in enumerate(f):
        if i > 0:
            if line[0] == "+":
```

```

        arr.append(line[2:-1])
    else:
        x = arr.pop(-1)
        stack.append(x)
    print(arr)
print(stack)
with open('output.txt', 'w') as x:
    x.writelines("%s\n" % line for line in stack)
print("время работы", (time.perf_counter() - t_start), "секунд")
print('память', process.memory_info().rss / 1024 ** 2, 'mb')

```

Реализованы операции стека - добавление элемента в конец и удаление из конца.

Результат работы кода на примерах из текста задачи:

The screenshot shows a code editor with two tabs: 'Задачи по варианту\input.txt' and 'Дополнительные задачи\input.txt'. The first tab is active, showing a list of operations: 1: 6, 2: + 1, 3: + 10, 4: -, 5: + 2, 6: + 1234, 7: -. Below this, another set of tabs shows 'Задачи по варианту\output.txt' and 'Дополнительные задачи\output.txt'. The second tab is active, showing the output: 1: 10, 2: 1234, 3: (empty).

Результат работы кода на максимальных и минимальных значениях:

```

1 1000000
2 + 999999999
3 + 999999999
4 + 999999999
5 + 999999998
6 -
7 + 999999999

```

```

1 999999998

```

```

1 2
2 + 1
3 -

```

```

1 1

```

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.109375 sec	14.179140625 KB
Пример из задачи	0.109375 sec	17.244140625 KB
Пример из задачи	0.109375 sec	17.244140625 KB
Верхняя граница диапазона значений	0.109375 sec	196.244140625 KB

ВХОДНЫХ ДАННЫХ ИЗ ТЕКСТА ЗАДАЧИ		
------------------------------------	--	--

Вывод по задаче: в задаче я реализовала стек.

### Задача №6. Очередь с минимумом

Реализуйте работу очереди. В дополнение к стандартным операциям очереди, необходимо также отвечать на запрос о минимальном элементе из тех, которые сейчас находятся в очереди. Для каждой операции запроса минимального элемента выведите ее результат.

На вход программе подаются строки, содержащие команды. Каждая строка содержит одну команду. Команда – это либо «+ N», либо «-», либо «?». Команда «+ N» означает добавление в очередь числа N, по модулю не превышающего  $10^9$ . Команда «-» означает изъятие элемента из очереди. Команда «?» означает запрос на поиск минимального элемента в очереди.

- Формат входного файла (input.txt). В первой строке содержится M ( $1 \leq M \leq 10^6$ ) – число команд. В последующих строках содержатся команды, по одной в каждой строке.
- Формат выходного файла (output.txt). Для каждой операции поиска минимума в очереди выведите её результат. Результаты должны быть выведены в том порядке, в котором эти операции встречаются во входном файле. Гарантируется, что операций извлечения и поиска минимума для пустой очереди не производится.
- Ограничение по времени. 2 сек.
- Ограничение по памяти. 256 мб.

```
import time
import os
import psutil
process = psutil.Process(os.getpid())
t_start = time.perf_counter()

stack = []
arr = []

with open('input.txt', 'r') as f:
    for i, line in enumerate(f):
        if i > 0:
            if line[0] == "+":
                arr.append(line[2:-1])
            elif line[0] == "-":
```

```

        del arr[0]
    else:
        stack.append(min(arr))

print(stack)
with open('../Задачи по варианту/output.txt', 'w') as x:
    x.writelines("%s\n" % line for line in stack)
print("время работы", (time.perf_counter() - t_start), "секунд")
print('память', process.memory_info().rss / 1024 ** 2, 'mb')

```

Реализована очередь с добавлением и удалением элемента, а также операция поиска минимума.

Результат работы кода на примерах из текста задачи:

Линия	Ввод	Выход
1	7	1
2	+ 1	1
3	?	10
4	+ 10	
5	?	
6	-	
7	?	
8	-	

Результат работы кода на максимальных и минимальных значениях:

```

1 1000000
2 + 999999999
3 + 999999999
4 + 999999999
5 + 999999999
6 + 999999999
7 + 999999999
8 + 999999999
9 -
10 0

1 по варианту\output.txt x
Дополнительные задачи\output.txt x
1 999999999
2 999999999
3 999999999
4 999999999

```

```

1 2
2 + 13
3 ?

1 по варианту\output.txt x
Дополнительные задачи\output.txt x
1 13

```

	Время выполнения	Затраты памяти
Нижняя граница диапазона значений входных данных из текста задачи	0.1875 sec	17.5947265625 KB

Пример из задачи	0.125 sec	17.7724609375 KB
Пример из задачи	0.125 sec	17.7724609375 KB
Верхняя граница диапазона значений входных данных из текста задачи	0.15625 sec	156.7783203125 KB

Вывод по задаче: в задаче я реализовала очередь с поддержкой минимума.

### **Вывод:**

В работе я вспомнила и реализовала основные структуры данных (стек и очередь), а также применила их в некоторых практических задачах (таких как проверка на правильную скобочную последовательность и польский калькулятор).