

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.

Выполнил: Криличевский М. Е.

Номер группы: К3139

Проверил: Афанасьев А. В.

Санкт-Петербург

2024 г.

## Содержание

<i>Задание 1</i> .....	<b>3</b>
<i>Задание 3</i> .....	<b>6</b>
<i>Задание 5</i> .....	<b>9</b>
<i>Задание 7</i> .....	<b>12</b>
<i>Задание 9</i> .....	<b>15</b>
<i>Вывод</i> .....	<b>18</b>

# Задание 1

## 1 задача. 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.
- Пример:

input.txt	output.txt
6	10
+ 1	1234
+ 10	
-	
+ 2	
+ 1234	
-	

## Решение:

```
1. from lab_4.utils import read_input, write_output,
   check, string_column
2.
3.
4. def steck(a):
5.     m = []
6.     for i in range(len(a)-1):
7.         if a[i+1]=="-" and len(a[i+1])==1:
8.             s = "".join(x for x in a[i] if
   x.isdecimal())
9.             m.append(s)
10.    return m
11.
12. if __name__=="__main__":
13.     FILE_INPUT = "../txtf/input.txt"
14.     FILE_OUTPUT = "../txtf/output.txt"
15.     list = read_input(FILE_INPUT)
16.     check = check(FILE_INPUT)
17.     if check:
18.         result = string_column(steck(list))
19.         write_output(FILE_OUTPUT,result)
20.         print("Входные данные корректны")
21.     if not(check):
22.         write_output(FILE_OUTPUT, "ОШИБКА
   ВХОДНЫХ ДАННЫХ")
23.         raise ValueError("ОШИБКА ВХОДНЫХ
   ДАННЫХ")
```

## Объяснение решения:

Этот код анализирует список строк, извлечённый из файла, и возвращает последовательность чисел, находящихся перед строками, содержащими один символ "-". Если входные данные корректны, программа обрабатывает их по заданным правилам и записывает результат в выходной файл. В случае ошибки данных выводится сообщение об ошибке.

Входной файл:

time_memory.py		input.txt
1	6	
2	+ 1	
3	+ 10	
4	-	
5	+ 2	
6	+ 1234	
7	-	

Выходной файл:

time_memory.py	
1	10
2	1234

Тестирование времени и памяти:

```
/usr/local/bin/python3.12 /Users/max/Pycharm
ВРЕМЯ: 0.00036733399610966444
Память: 9.6 МБ
Размер списка: 88 байт

Process finished with exit code 0
```

## Задание 3

### 3 задача. 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  $10^4$  включительно. В каждой из последовательностей присутствуют только скобки указанных выше видов.

3

- **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.

- Пример:

input.txt	output.txt
5	YES
()	YES
()	NO
()	NO
()	NO
()	

## Решение:

```
1. from lab_4.utils import read_input, write_output, check_three,
   string_column
2.
3. def bracket_sequence(a):
4.     results = []
5.
6.     for s in a:
7.         stack = []
8.         is_valid = True
9.
10.        for char in s:
11.            if char in "(":
12.                stack.append(char)
13.            elif char in ")":
14.                if not stack:
15.                    is_valid = False
16.                    break
17.                top = stack.pop()
18.
19.            if (top == '(' and char != ')') or (top
== '[' and char != ']'):
20.                is_valid = False
21.                break
22.
23.            if stack:
24.                is_valid = False
25.
26.        results.append("ДА" if is_valid else "НЕТ")
27.
28.    return string_column(results)
29.
30. if __name__=="__main__":
31.     FILE_INPUT = "../txtf/input.txt"
32.     FILE_OUTPUT = "../txtf/output.txt"
33.     list = read_input(FILE_INPUT)
34.     check = check_three(FILE_INPUT)
35.     if check:
36.         write_output(FILE_OUTPUT,bracket_sequence(list))
37.         print("Входные данные корректны")
38.     else:
39.         raise ValueError("ОШИБКА ВХОДНЫХ ДАННЫХ")
```

## Объяснение решения:

Этот код проверяет корректность последовательностей скобок в списке строк, считанном из входного файла. Если последовательность скобок правильная (каждая открывающая скобка закрывается соответствующей закрывающей), результат для данной строки — "ДА". Если последовательность некорректная, результат — "НЕТ". Все результаты выводятся в виде столбца. Программа сохраняет результаты в выходной файл, если входные данные корректны.

Входной файл:

```
time_memory.py  input.txt x
1      5
2      ()()
3      ([])
4      ([)]
5      (([]
6      )(
```

Выходной файл:

```
time_memory.py  input.txt
1      ДА
2      ДА
3      НЕТ
4      НЕТ
5      НЕТ
```

Тестирование времени и памяти:

```
/usr/local/bin/python3.12 /Users/max/Pycharm
ВРЕМЯ: 0.00032295800338033587
Память: 9.7 МБ
Размер списка: 94 байт

Process finished with exit code 0
```



# Задание 5

## 5 задача. 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 содержится  $n$  ( $1 \leq n \leq 400000$ ) – 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.
- Пример:

input.txt	output.txt	input.txt	output.txt	input.txt	output.txt
5	2	5	2	3	
push 2	2	push 1	1	push 1	
push 1		push 2		push 7	
max		max		pop	
pop		pop			
max		max			

5

input.txt	output.txt	input.txt	output.txt
10	9	6	7
push 2	9	push 7	7
push 3	9	push 1	
push 9	9	push 7	
push 7		max	
push 2		pop	
max		max	
max			
max			
pop			
max			

## Решение:

```
1. from lab_4.utils import read_input, check_five, write_output,
   string_column
2.
3. def process_stack(commands):
4.     stack = []
5.     max_stack = []
6.     results = []
7.
8.     for command in commands:
9.         if command.startswith("push"):
10.             _, value = command.split()
11.             value = int(value)
12.             stack.append(value)
13.
14.             if max_stack:
15.                 max_stack.append(max(value, max_stack[-
16. 1]))
17.             else:
18.                 max_stack.append(value)
19.         elif command.startswith("pop"):
20.             if stack:
21.                 stack.pop()
22.                 max_stack.pop()
23.             elif command.startswith("max"):
24.                 if max_stack:
25.                     results.append(str(max_stack[-1]))
26.
27.     return results
28.
29. if __name__=="__main__":
30.     FILE_INPUT = "../txtf/input.txt"
31.     FILE_OUTPUT = "../txtf/output.txt"
32.     list = read_input(FILE_INPUT)
33.     check = check_five(FILE_INPUT)
34.     if check:
35.         result = process_stack(list)
36.         write_output(FILE_OUTPUT, string_column(result))
37.         print("Входные данные корректны")
38.     else:
39.         write_output(FILE_OUTPUT, "ОШИБКА ВХОДНЫХ
40.     raise ValueError("ОШИБКА ВХОДНЫХ ДАННЫХ")
    ДАННЫХ")
```

## Объяснение решения:

Этот код реализует обработку команд работы со стеком, включая команды добавления числа, удаления верхнего элемента и получения максимального элемента в текущем состоянии стека. Помимо основного стека, используется дополнительный стек для отслеживания максимального значения, что позволяет выполнять команду `max`. Если входные данные корректны, результат записывается в выходной файл. При некорректных входных данных выводится сообщение об ошибке.

Входной файл:

time_memory.py		input.txt
1	5	
2	push 2	
3	push 1	
4	max	
5	pop	
6	max	

Выходной файл:

time_memory.py	
1	2
2	2

Тестирование времени и памяти:

```
/usr/local/bin/python3.12 /Users/max/PycharmI
ВРЕМЯ: 0.0003386669995961711
Память: 9.5 МБ
Размер списка: 88 байт

Process finished with exit code 0
```

## Задание 7

### 7 задача. Максимум в движущейся последовательности

Задан массив из  $n$  целых чисел -  $a_1, \dots, a_n$  и число  $m < n$ , нужно найти максимум среди последовательности ("окна")  $\{a_i, \dots, a_{i+m-1}\}$  для каждого значения  $1 \leq i \leq n - m + 1$ . Простой алгоритм решения этой задачи за  $O(nm)$  сканирует каждое "окно" отдельно.

Ваша цель - алгоритм за  $O(n)$ .

- **Формат входного файла (input.txt).** В первой строке содержится целое число  $n$  ( $1 \leq n \leq 10^5$ ) – количество чисел в исходном массиве, вторая строка содержит  $n$  целых чисел  $a_1, \dots, a_n$  этого массива, разделенных пробелом ( $0 \leq a_i \leq 10^5$ ). В третьей строке - целое число  $m$  - ширина "окна" ( $1 \leq m \leq n$ ).
- **Формат выходного файла (output.txt).** Нужно вывести  $\max a_i, \dots, a_{i+m-1}$  для каждого  $1 \leq i \leq n - m + 1$ .
- Ограничение по времени. 5 сек.
- Ограничение по памяти. 512 мб.
- Пример:

input.txt	output.txt
8	7 7 5 6 6
2 7 3 1 5 2 6 2	
4	

- Есть несколько решений этой задачи. Например:
  - использование очереди на основе двух стеков;
  - использование Dequeue.

## Решение:

```
1. from collections import deque
2. from lab_4.utils import string, read_input_seven, check_seven,
   write_output
3.
4. def max_dynamic_sequence(n,a,m):
5.     decart_queue = deque()
6.     result = []
7.
8.     for i in range(n):
9.         if decart_queue and decart_queue[0] < i - m + 1:
10.             decart_queue.popleft()
11.
12.         while decart_queue and a[decart_queue[-1]] <=
a[i]:
13.             decart_queue.pop()
14.
15.             decart_queue.append(i)
16.
17.             if i >= m - 1:
18.                 result.append(a[decart_queue[0]])
19.
20.     return result
21.
22. if __name__=="__main__":
23.     FILE_INPUT = "../txtf/input.txt"
24.     FILE_OUTPUT = "../txtf/output.txt"
25.     check = check_seven(FILE_INPUT)
26.     if check:
27.         argues = read_input_seven(FILE_INPUT)
28.         n, a, m = argues[0], argues[1], argues[2]
29.         result = max_dynamic_sequence(n, a, m)
30.         write_output(FILE_OUTPUT,string(result))
31.         print("Входные данные корректны")
32.
33.     else:
34.         write_output(FILE_OUTPUT, "ОШИБКА ВХОДНЫХ
ДАННЫХ")
35.         raise ValueError("ОШИБКА ВХОДНЫХ ДАННЫХ")
```

## Объяснение решения:

Этот код решает задачу поиска максимума в подмассивах фиксированной длины  $m$  для массива  $a$  длины  $n$ . Алгоритм эффективно использует двухстороннюю очередь ('deque') для поддержки скользящего окна, обеспечивая временную сложность. Результаты записываются в выходной файл, если входные данные корректны.

Входной файл:

time_memory.py	
1	8
2	2 7 3 1 5 2 6 2
3	4

Выходной файл:

time_memory.py		input.txt
1	7 7 5 6 6	
2		
3		

Тестирование времени и памяти:

```
/usr/local/bin/python3.12 /Users/max/PycharmI  
ВРЕМЯ: 0.0003032500098925084  
Память: 9.9 МБ  
Размер списка: 120 байт  
  
Process finished with exit code 0
```

## Задание 9

### 9 задача. Поликлиника

Очередь в поликлинике работает по сложным правилам. Обычные пациенты при посещении должны вставать в конец очереди. Пациенты, которым "только справку забрать" встают ровно в ее середину, причем при нечетной длине очереди они встают сразу за центром. Напишите программу, которая отслеживает порядок пациентов в очереди.

- **Формат входного файла (input.txt).** В первой строке записано одно целое число  $n$  ( $1 \leq n \leq 10^5$ ) - число запросов к вашей программе. В следующих  $n$  строках заданы описания запросов в следующем формате:

- «+ i» – к очереди присоединяется пациент  $i$  ( $1 \leq i \leq N$ ) и встает в ее конец;
- «\* i» – пациент  $i$  встает в середину очереди ( $1 \leq i \leq N$ );
- «-» – первый пациент в очереди заходит к врачу. Гарантируется, что на момент каждого такого запроса очередь будет не пуста.

8

- **Формат выходного файла (output.txt).** Для каждого запроса третьего типа в отдельной строке выведите номер пациента, который должен зайти к шаманам.
- Ограничение по времени. Оцените время работы и используемую память при заданных максимальных значениях.
- Пример:

input.txt	output.txt	input.txt	output.txt
7	1	10	1
+ 1	2	+ 1	3
+ 2	3	+ 2	2
-		* 3	5
+ 3		-	4
+ 4		+ 4	
-		* 5	
-		-	
		-	
		-	
		-	

## Решение:

```
1. from collections import deque
2. from lab_4.utils import read_input, check_nine, write_output,
   string_column
3.
4. def process_requests(n, requests):
5.     queue = deque()
6.     results = []
7.
8.     for request in requests:
9.         if request[0] == '+':
10.
11.             _, i = request.split()
12.             queue.append(i)
13.         elif request[0] == '*':
14.
15.             _, i = request.split()
16.             mid = (len(queue) + 1) // 2
17.             queue.insert(mid, i)
18.         elif request[0] == '-':
19.
20.             results.append(queue.popleft())
21.
22.     return results
23.
24. if __name__=="__main__":
25.     FILE_INPUT = "../txtf/input.txt"
26.     FILE_OUTPUT = "../txtf/output.txt"
27.     check = check_nine(FILE_INPUT)
28.     if check:
29.         list = read_input(FILE_INPUT)
30.
31.         write_output(FILE_OUTPUT, string_column(process_requests(list[1
32.             :], list)))
33.         print("Входные данные корректны")
34.     else:
35.         write_output(FILE_OUTPUT, "ОШИБКА ВХОДНЫХ
        ДАННЫХ")
36.         raise ValueError("ОШИБКА ВХОДНЫХ ДАННЫХ")
```

## Объяснение решения:

Этот код реализует обработку последовательности запросов на двустороннюю очередь deque. Входные запросы могут добавлять элемент в конец, в середину или удалять первый элемент из очереди. После выполнения запросов выводится список результатов операций удаления. Если входные данные корректны, результат записывается в выходной файл, иначе программа выводит сообщение об ошибке.



Входной файл:

time_memory.py		input.txt
1	7	
2	+ 1	
3	+ 2	
4	-	
5	+ 3	
6	+ 4	
7	-	
8	-	

Выходной файл:

time_memory.py	
1	1
2	2
3	3

Тестирование времени и памяти:

```
/usr/local/bin/python3.12 /Users/max/PycharmI
ВРЕМЯ: 0.00025029199605342
Память: 10.5 МБ
Размер списка: 88 байт

Process finished with exit code 0
```

## Вывод

В данной лабораторной работе познакомился и поработал с элементарными структурами данных: стеком, очередью и связанным списком.