Санкт-Петербургский Национальный Исследовательский Университет
Информационных Технологий, Механики и Оптики

Лабораторная работа №4

по дисциплине

«Алгоритмы и структуры данных»

Выполнил: Студент группы P3217

Сергачев Данила Дмитриевич

Преподаватели:

Романов Алексей Андреевич и

Волчек Дмитрий Геннадьевич

Санкт-Петербург

2018 г

Задание №1

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.

Решение

```
package week4;
import mooc.*;
import week2.EdxIO;

public class task1 {
        public static void main(String[] args){
            try (EdxIO io = EdxIO.create()) {
                int n = io.nextInt();
                Stack stack = new Stack(n);
                StringBuilder st = new StringBuilder();
                for(int i = 0; i < n; i++){
                    char symbol = io.nextChar();
                    if(symbol == '+'){
                        stack.addElement(io.nextInt());
                    }
                    else{
                        if(!stack.isEmpty()){
                            st.append(stack.deleteElement());
                            st.append("\n");
                        }
                    }
                }

                io.print(st);
            }
        }

        static class Stack {
            private int mSize;
            private int[] stackArray;
            private int top;

            public Stack(int m) {
                this.mSize = m;
                stackArray = new int[mSize];
                top = -1;
            }

            public void addElement(int element) {
```

```
                stackArray[++top] = element;
            }

            public int deleteElement() {
                return stackArray[top--];
            }

            public boolean isEmpty() {
                return (top == -1);
            }
        }
}
```

Результат работы

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.500	75395072	13389454	5693807
1	OK	0.125	21626880	33	10
2	OK	0.109	21565440	11	3
3	OK	0.125	21590016	19	6
4	OK	0.093	21618688	19	6
5	OK	0.109	21606400	19	6
6	OK	0.125	21614592	96	45
7	OK	0.125	21581824	85	56

Задание №2

м

Имя входного файла:	input.txt
Имя выходного файла:	output.txt
Ограничение по времени:	**2.5 секунды**
Ограничение по памяти:	256 мегабайт

Дано n строк, выведите их порядок после k фаз цифровой сортировки.

Формат входного файла

В первой строке входного файла содержатся числа n — число строк, m — их длина и k – число фаз цифровой сортировки ($1 \le n \le 10^6, 1 \le k \le m \le 10^6, n \cdot m \le 5 \cdot 10^7$). Далее находится описание строк, **но в нетривиальном формате**. Так, i-ая строка ($1 \le i \le n$) записана в i-ых символах второй, ..., $(m+1)$-ой строк входного файла. Иными словами, строки написаны по вертикали. **Это сделано специально, чтобы сортировка занимала меньше времени.**

Строки состоят из строчных латинских букв: от символа "a" до символа "z" включительно. В таблице символов ASCII все эти буквы располагаются подряд и в алфавитном порядке, код буквы "a" равен 97, код буквы "z" равен 122.

Формат выходного файла

Выведите номера строк в том порядке, в котором они будут после k фаз цифровой сортировки.

Решение

```
package week4;
import mooc.*;
import week2.EdxIO;

public class task2 {
        public static void main(String[] args){
            try (EdxIO io = EdxIO.create()) {
                int n = io.nextInt();
                boolean[] resArr = new boolean[n];

                Queue queue = new Queue(n);

                StringBuilder st = new StringBuilder();
                for(int i = 0; i < n; i++){
                    char symbol = io.nextChar();
                    if(symbol == '+'){
                        queue.addElement(io.nextInt());
                    }
                    else{
                        if(!queue.isEmpty()){
                            st.append(queue.deleteElement());
                            st.append("\n");
                        }
                    }
                }

                io.print(st);
            }
        }

        static class Queue {
            private int mSize;
            private int[] queueArray;
            private int top;

            public Queue(int m) {
                this.mSize = m;
                queueArray = new int[mSize];
                top = this.mSize;
            }

            public void addElement(int element) {
                queueArray[--top] = element;
            }

            public int deleteElement() {
                return queueArray[top++];
            }
```

```
            public boolean isEmpty() {
                return (top == this.mSize);
            }
        }
}
```

Результат работы

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.453	75476992	13389454	5693807
1	OK	0.109	21655552	20	7
2	OK	0.125	21581824	11	3
3	OK	0.125	21557248	19	6
4	OK	0.140	21602304	19	6
5	OK	0.109	21585920	96	45
6	OK	0.109	21573632	85	56
7	OK	0.156	21540864	129	12
8	OK	0.109	21598208	131	12
9	OK	0.109	21594112	859	538
10	OK	0.109	21594112	828	573
11	OK	0.109	21590016	1340	12
12	OK	0.109	21565440	1325	12

Задание №3

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.

Решение

```
package week4;

import mooc.*;
import week2.EdxIO;

public class task3 {
```

```
public static void main(String[] args){
    try (EdxIO io = EdxIO.create()) {
        int n = io.nextInt();

        Stack stack = new Stack(n);

        StringBuilder st = new StringBuilder();

        for(int i = 0; i < n; i++){
            char[] line = io.next().toCharArray();
            boolean isCorrect = true;

            for(int j = 0; j < line.length; j++){
                switch(line[j]){
                    case '(': {
                        stack.addElement(line[j]);
                        break;
                    }
                    case '[': {
                        stack.addElement(line[j]);
                        break;
                    }
                    case ')': {
                        if(stack.isEmpty() || stack.readTop() != '('){
                            isCorrect = false;
                        } else {
                            stack.deleteElement();
                        }
                        break;
                    }
                    default :{
                        if(stack.isEmpty() || stack.readTop() != '['){
                            isCorrect = false;
                        } else {
                            stack.deleteElement();
                        }
                    }
                }
            }

            st.append(isCorrect && stack.isEmpty() ? "YES\n" : "NO\n");

            stack.refresh();
        }

        io.print(st);
    }
}

static class Stack {
    private int mSize;
```

```
            private char[] stackArray;
            private int top;

            public Stack(int m) {
                this.mSize = m;
                stackArray = new char[mSize];
                top = -1;
            }

            public void addElement(char element) {
                stackArray[++top] = element;
            }

            public char deleteElement() {
                return stackArray[top--];
            }

            public boolean isEmpty() {
                return (top == -1);
            }

            public void refresh(){
                top = -1;
            }

            public char readTop() {
                return stackArray[top];
            }
        }
}
```

Задание №4

Реализуйте работу очереди. В дополнение к стандартным операциям очереди, необходимо также отвечать на запрос о минимальном элементе из тех, которые сейчас находится 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.

Решение

```
package week4;
import mooc.*;

public class task4 {
        public static void main(String[] args){
            try (EdxIO io = EdxIO.create()) {
                int n = io.nextInt();
```

```
        Queue queue = new Queue(n);
        Queue queueMin = new Queue(n);

        StringBuilder st = new StringBuilder();
        for(int i = 0; i < n; i++){
            char symbol = io.nextChar();
            switch(symbol){
                case '+':{
                    int item = io.nextInt();
                    queue.addElement(item);

                    if(queueMin.isEmpty()){
                        queueMin.addElement(item);
                    } else {
                        queueMin.insertMin(item);
                    }

                    break;
                }
                case '-':{
                    if(!queue.isEmpty()){
                        if(queueMin.getFront() == queue.deleteElement()){
                            queueMin.deleteElement();
                        }
                    }

                    break;
                }
                default: {
                    st.append(queueMin.getFront());
                    st.append('\n');
                }
            }
        }

        io.print(st);
    }
}

static class Queue{
    private int[] queue;
    private int maxSize; // максимальное количество элементов в очереди
    private int nElem;  // текущее количество элементов в очереди
    private int front;
    private int rear;


public Queue(int maxSize) {
    this.maxSize = maxSize;
    queue = new int[maxSize];
    rear = -1;
```

```
            front = 0;
            nElem = 0;
        }

        public void addElement(int elem) {
            if (rear == maxSize - 1) {  // циклический перенос
                rear = -1;
            }

            queue[++rear] = elem;  //увеличение Rear и вставка
            nElem++;  // увеличение количества элементов в очереди
        }

        public int deleteElement() {
            int temp = queue[front++]; // получаем первый элемент из очереди

                if (front == maxSize) { // циклический перенос
                    front = 0;
                }
                nElem--; // уменьшаем количество элементов в очереди
                return temp;

        }

        public int getFront() {
            return queue[front];
        }

        public int getRear() {
            return queue[rear];
        }

        public boolean isEmpty() {
            return (nElem == 0);
        }

        public void insertMin(int elem){
            if(nElem > 1 && queue[rear] > elem){
                 queue[front] = elem;
            }
            else{
                addElement(elem);
            }
        }
    }
}
```

Результат работы

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.125	29454336	13389342	4002151
1	OK	0.000	2236416	29	10
2	OK	0.031	2248704	11	3
3	OK	0.000	2236416	22	6
4	OK	0.000	2248704	22	6
5	OK	0.000	2236416	36	9
6	OK	0.015	2248704	48	12
7	OK	0.000	2232320	76	35
8	OK	0.000	2248704	129	12