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

Задачи четвёртой недели

по курсу «Алгоритмы и структуры данных»

на Openedu

Выполнил: студент группы P3218

Масалкин Савелий Евгеньевич

Санкт-Петербург, 2019г

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.

```
#include "edx-io.hpp"
using namespace std;


int main() {

	long N;
	io >> N;
	char action;

	long* stack = new long[N];
	long offset = -1;

	for (long i = 0; i < N; i++) {
		io >> action;
		if (action == '+') {
			offset++;
			io >> stack[offset];
		}
		else {
			io << stack[offset] << '\n';
			offset--;
		}
	}

	return 0;
}
```

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.156	19202048	13389454	5693807
1	OK	0.015	2224128	33	10
2	OK	0.000	2224128	11	3
3	OK	0.000	2224128	19	6
4	OK	0.015	2236416	19	6
5	OK	0.000	2224128	19	6
6	OK	0.000	2224128	96	45
7	OK	0.000	2224128	85	56
8	OK	0.000	2215936	129	11
9	OK	0.000	2236416	131	12
10	OK	0.015	2236416	859	540
11	OK	0.031	2236416	828	573
12	OK	0.000	2224128	1340	11
13	OK	0.000	2220032	1325	12
14	OK	0.000	2248704	8292	5590
15	OK	0.000	2228224	8212	5706
16	OK	0.000	2228224	13298	111
17	OK	0.015	2232320	13354	12
18	OK	0.000	2232320	82372	56548
19	OK	0.015	2248704	82000	56993
20	OK	0.000	2277376	132796	1134
21	OK	0.000	2265088	133914	11
22	OK	0.015	2650112	819651	569557
23	OK	0.031	2842624	819689	569681
24	OK	0.015	3543040	1328670	11294
25	OK	0.015	3551232	1338543	11
26	OK	0.156	10018816	8196274	5693035
27	OK	0.140	12013568	8193816	5693807
28	OK	0.062	19025920	13286863	112020
29	OK	0.046	19202048	13389454	11
30	OK	0.046	19202048	13388564	11

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.

```
#include "edx-io.hpp"
using namespace std;


int main() {

	long N;
	io >> N;
	char action;

	long* stack = new long[N];
	long offset = -1;
	long head = 0;

	for (long i = 0; i < N; i++) {
		io >> action;
		if (action == '+') {
			offset++;
			io >> stack[offset];
		}
		else {
			io << stack[head] << '\n';
			head++;
		}
	}

	return 0;
}
```

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.156	19202048	13389454	5693807
1	OK	0.000	2220032	20	7
2	OK	0.000	2224128	11	3
3	OK	0.000	2224128	19	6
4	OK	0.015	2224128	19	6
5	OK	0.000	2224128	96	45
6	OK	0.015	2224128	85	56
7	OK	0.015	2224128	129	12
8	OK	0.000	2224128	131	12
9	OK	0.015	2224128	859	538
10	OK	0.000	2224128	828	573
11	OK	0.000	2236416	1340	12
12	OK	0.000	2236416	1325	12
13	OK	0.000	2228224	8292	5589
14	OK	0.000	2232320	8212	5706
15	OK	0.000	2232320	13298	115
16	OK	0.000	2232320	13354	12
17	OK	0.000	2244608	82372	56552
18	OK	0.015	2248704	82000	56993
19	OK	0.000	2265088	132796	1124
20	OK	0.000	2256896	133914	12
21	OK	0.015	2846720	819651	569553
22	OK	0.015	2846720	819689	569681
23	OK	0.000	3547136	1328670	11296
24	OK	0.000	3551232	1338543	12
25	OK	0.156	12017664	8196274	5693025
26	OK	0.156	12013568	8193816	5693807
27	OK	0.062	19058688	13286863	112110
28	OK	0.046	19202048	13389454	10
29	OK	0.062	19202048	13388564	11

Скобочная последовательность

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.

```
#include "edx-io.hpp"
#include <string>
using namespace std;


string check(string seq, char *stack, long offset) {
	for (int j = 0; j < seq.size(); j++) {
		switch (seq[j]) {
		case '(':
			stack[++offset] = seq[j];
			continue;
		case '[':
			stack[++offset] = seq[j];
			continue;
		case ')':
			if (offset < 0 || stack[offset] != '(') {
				return "NO\n";
			}
			offset--;
			continue;
		case ']':
			if (offset < 0 || stack[offset] != '[') {
				return "NO\n";
			}
			offset--;
			continue;
		}
	}
	if (offset == -1) {
		return "YES\n";
	}
```

```
        return "NO\n";
}

int main() {

        long N;
        string seq;
        io >> N;
        char* stack = new char[10000];


        for (long i = 0; i < N; i++) {
                long offset = -1;
                io >> seq;
                io << check(seq, stack, offset);
        }

        return 0;
}
```

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.031	6848512	5000885	2133
1	OK	0.015	2457600	31	22
2	OK	0.000	2469888	15	16
3	OK	0.015	2453504	68	66
4	OK	0.000	2453504	324	256
5	OK	0.000	2449408	1541	1032
6	OK	0.000	2457600	5880	2128
7	OK	0.000	2453504	50867	2129
8	OK	0.000	2330624	500879	2110
9	OK	0.031	6848512	5000884	2120
10	OK	0.031	6848512	5000885	2133

Очередь с минимумом

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.

```
#include "edx-io.hpp"

using namespace std;
int main() {
	long N;
	io >> N;
	char action;
	//Очередь для чисел
	long* queue = new long[N];
	long offset = -1;
	long head = 0;
	//Очередь для наименьших чисел
	long *minQueue = new long[N];
	long minOffset = -1;
	long minHead = 0;

	for (long i = 0; i < N; i++) {
		io >> action;
		switch (action)
		{
		case '+':
			io >> queue[++offset];
			//Элемент вставляется в очередь для наименьших чисел таким образом,
			//чтобы все элементы спереди были меньше него
			while (minOffset - minHead >= 0 && minQueue[minOffset] >
queue[offset]) {
				minOffset--;
			}
			minQueue[++minOffset] = queue[offset];
			continue;
		case '-':
			//Если удяляется наименьший элемент (самый первый в очереди
наименьших),
			//голова очереди сдвигается к следующему
			if (queue[head] == minQueue[minHead]) {
				minHead++;
			}
			head++;
			continue;
		case '?':
			//Ввыводится первый элемент в очереди наименьших
			io << minQueue[minHead] << '\n';
			continue;
		}

	}
	return 0;
}
```

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.109	23203840	13389342	4002151
1	OK	0.000	2224128	29	10
2	OK	0.000	2224128	11	3
3	OK	0.000	2224128	22	6
4	OK	0.000	2220032	22	6
5	OK	0.000	2224128	36	9
6	OK	0.000	2224128	48	12
7	OK	0.015	2236416	76	35
8	OK	0.000	2224128	129	12
9	OK	0.000	2220032	67	48
10	OK	0.000	2224128	44	9
11	OK	0.000	2224128	45	9
12	OK	0.000	2224128	44	9
13	OK	0.000	2224128	45	9
14	OK	0.015	2220032	721	384
15	OK	0.000	2224128	1340	12
16	OK	0.000	2236416	640	407
17	OK	0.015	2236416	445	90
18	OK	0.000	2220032	456	100
19	OK	0.015	2220032	445	90
20	OK	0.000	2224128	456	100
21	OK	0.015	2232320	6616	3812
22	OK	0.000	2248704	13389	12
23	OK	0.015	2248704	6461	4008
24	OK	0.000	2236416	4896	1140
25	OK	0.000	2248704	5007	1250
26	OK	0.000	2248704	4896	1140
27	OK	0.000	2236416	5007	1250
28	OK	0.000	2265088	64907	39589
29	OK	0.000	2273280	133814	12
30	OK	0.015	2265088	64675	39996

Quack

Язык Quack — забавный язык, который фигурирует в одной из задач с Internet Problem Solving Contest. В этой задаче вам требуется написать интерпретатор языка Quack.

Виртуальная машина, на которой исполняется программа на языке Quack, имеет внутри себя очередь, содержащую целые числа по модулю 65536 (то есть, числа от 0 до 65535, соответствующие беззнаковому 16-битному целому типу). Слово get в описании операций означает извлечение из очереди, put — добавление в очередь. Кроме того, у виртуальной машины есть 26 регистров, которые обозначаются буквами от 'a' до 'z'. Изначально все регистры хранят нулевое значение. В языке Quack существуют следующие команды (далее под α и β подразумеваются некие абстрактные временные переменные):

+	Сложение: get α, get β, put (α+β) mod 65536
-	Вычитание: get α, get β, put (α−β) mod 65536
*	Умножение: get α, get β, put (α·β) mod 65536
/	Целочисленное деление: get α, get β, put α div β (будем считать, что α div 0 = 0)
%	Взятие по модулю: get α, get β, put α mod β (будем считать, что α mod 0 = 0)
>[register]	Положить в регистр: get α, установить значение [register] в α
<[register]	Взять из регистра: put значение [register]
P	Напечатать: get α, вывести α в стандартный поток вывода и перевести строку
P[register]	Вывести значение регистра [register] в стандартный поток вывода и перевести строку
C	Вывести как символ: get α, вывести символ с ASCII-кодом α mod 256 в стандартный поток вывода
C[register]	Вывести регистр как символ: вывести символ с ASCII-кодом α mod 256 (где α — значение регистра [register]) в стандартный поток вывода
:[label]	Метка: эта строка программы имеет метку [label]
J[label]	Переход на строку с меткой [label]
Z[register][label]	Переход если 0: если значение регистра [register] равно нулю, выполнение программы продолжается с метки [label]
E[register1][register2][label]	Переход если равны: если значения регистров [register1] и [register2] равны, исполнение программы продолжается с метки [label]
G[register1][register2][label]	Переход если больше: если значение регистра [register1] больше, чем значение регистра [register2], исполнение программы продолжается с метки [label]
Q	Завершить работу программы. Работа также завершается, если выполнение доходит до конца программы
[number]	Просто число во входном файле — put это число

```cpp
#include <fstream>
#include <queue>
#include <string>
#include <map>
using namespace std;


int main() {
	ifstream input("input.txt");
	ofstream output("output.txt");

	queue<unsigned short> MainQueue = {};
	map<char, unsigned short> registers = {
		{'a', 0},
		{'b', 0},
		{'c', 0},
		{'d', 0},
		{'e', 0},
		{'f', 0},
		{'g', 0},
		{'h', 0},
		{'i', 0},
		{'j', 0},
		{'k', 0},
		{'l', 0},
		{'m', 0},
		{'n', 0},
		{'o', 0},
		{'p', 0},
		{'q', 0},
		{'r', 0},
		{'s', 0},
		{'t', 0},
		{'u', 0},
		{'v', 0},
		{'w', 0},
		{'x', 0},
		{'y', 0},
		{'z', 0}
	};
	map <string, long> labels;
	string *commands = new string[100001];
	//Получаем список команд
	long N = -1;
	while (!input.eof()) {
		input >> commands[++N];
		//Запоминаем все метки
		if (commands[N][0] == ':') {
			labels.insert(pair<string, long>(commands[N].substr(1), N));
		}
	}
	if (commands[N].empty()) {
		N--;
	}
	input.close();
	//Указатель на команду в списке
	long P = 0;
	unsigned short a, b;

	do {
		switch (commands[P][0])
		{
		case '+':
			a = MainQueue.front();
			MainQueue.pop();
```

```
		b = MainQueue.front();
		MainQueue.pop();
		MainQueue.push((a + b) % 65536);
		P++;
		break;
	case '-':
		a = MainQueue.front();
		MainQueue.pop();
		b = MainQueue.front();
		MainQueue.pop();
		MainQueue.push((a - b) % 65536);
		P++;
		break;
	case '*':
		a = MainQueue.front();
		MainQueue.pop();
		b = MainQueue.front();
		MainQueue.pop();
		MainQueue.push((a * b) % 65536);
		P++;
		break;
	case '/':
		a = MainQueue.front();
		MainQueue.pop();
		b = MainQueue.front();
		MainQueue.pop();
		if (b == 0) {
			MainQueue.push(0);
		}
		else {
			MainQueue.push(a / b);
		}
		P++;
		break;
	case '%':
		a = MainQueue.front();
		MainQueue.pop();
		b = MainQueue.front();
		MainQueue.pop();
		if (b == 0) {
			MainQueue.push(0);
		}
		else {
			MainQueue.push(a % b);
		}
		P++;
		break;
	case '>':
		a = MainQueue.front();
		MainQueue.pop();
		registers[commands[P][1]] = a;
		P++;
		break;
	case '<':
		MainQueue.push(registers[commands[P][1]]);
		P++;
		break;
	case 'P':
		if (commands[P].size() == 1) {
			output << MainQueue.front() << '\n';
			MainQueue.pop();
		}
		else {
			output << registers[commands[P][1]] << '\n';
		}
```

```
                        P++;
                        break;
                case 'C':
                        if (commands[P].size() == 1) {
                                output << (char)(MainQueue.front() % 256);
                                MainQueue.pop();
                        }
                        else {
                                output << (char)(registers[commands[P][1]] % 256);
                        }
                        P++;
                        break;
                case ':':
                        P++;
                        break;
                case 'J':
                        P = labels[commands[P].substr(1)] + 1;
                        break;
                case 'Z':
                        if (registers[commands[P][1]] == 0) {
                                P = labels[commands[P].substr(2)] + 1;
                        }
                        else {
                                P++;
                        }
                        break;
                case 'E':
                        if (registers[commands[P][1]] == registers[commands[P][2]])
                                { P = labels[commands[P].substr(3)] + 1;
                        }
                        else {
                                P++;
                        }
                        break;
                case 'G':
                        if (registers[commands[P][1]] > registers[commands[P][2]])
                                { P = labels[commands[P].substr(3)] + 1;
                        }
                        else {
                                P++;
                        }
                        break;
                case 'Q':
                        P = N;
                        P++;
                        break;
                default:
                        MainQueue.push(stoi(commands[P]));
                        P++;
                }
                //Программа работает, пока не будет выполнена последняя команда
        } while (P <= N);
        output.close();

        return 0;
}
```

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.093	9293824	1349803	250850
1	OK	0.015	5574656	69	6
2	OK	0.000	5591040	232	232
3	OK	0.000	5570560	3	0
4	OK	0.015	5591040	100	19
5	OK	0.031	5578752	56	58890
6	OK	0.015	5591040	67	30000
7	OK	0.015	5591040	67	30000
8	OK	0.015	5578752	55	30000
9	OK	0.000	5586944	461	62
10	OK	0.000	5586944	11235	21
11	OK	0.000	5574656	23748	42
12	OK	0.015	5582848	66906	9136
13	OK	0.015	5574656	7332	993
14	OK	0.000	5582848	4611	632
15	OK	0.000	5586944	37968	7332
16	OK	0.015	5586944	14	3
17	OK	0.000	5591040	70	14
18	OK	0.000	5582848	350	70
19	OK	0.000	5586944	1750	350
20	OK	0.015	5586944	8750	1750
21	OK	0.015	5586944	43750	8750
22	OK	0.015	5586944	218750	43750
23	OK	0.015	5586944	34606	4867
24	OK	0.046	5787648	683180	7
25	OK	0.062	5771264	683102	0
26	OK	0.093	9293824	1349803	0
27	OK	0.078	5591040	491572	247791
28	OK	0.062	5595136	491488	249618
29	OK	0.078	5591040	491600	249600
30	OK	0.078	5591040	491502	250850
31	OK	0.078	5591040	491416	249477
32	OK	0.078	5591040	491520	250262
33	OK	0.078	5591040	491317	246859
34	OK	0.078	5591040	491514	248199
35	OK	0.078	5586944	491557	249601