Санкт-Петербургский национальный исследовательский университет

Информационных технологий, механики и оптики

Отчет по решению задач четвертой недели
По курсу «Алгоритмы и структуры данных»

на Openedu

Выполнил: Сыроватский Павел Валентинович

Группа P3218

Санкт-Петербург

2019

Стек

1.0 из 1.0 балла (оценивается)

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 109. Команда "-" означает изъятие элемента из стека. Гарантируется, что не происходит извлечения из пустого стека. Гарантируется, что размер стека в процессе выполнения команд не превысит 106 элементов.

Формат входного файла

В первой строке входного файла содержится M ($1 \le M \le 106$) — число команд. Каждая последующая строка исходного файла содержит ровно одну команду.

Формат выходного файла

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.

```
#include <fstream>
using namespace std;


int main() {
	ifstream input("input.txt");
	ofstream output("output.txt");
	char command;
	long n;
	long top = -1;
	input >> n;
	long* stack = new long[n];

	for (long i = 0; i < n; i++)
	{
		input >> command;
		if (command == '+') {
			top++;
			input >> stack[top];

		}
		else
		{
			output << stack[top] << '\n';
			top--;
		}
	}




	return 0;
}
```

Результат выполнения первой задачи

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		1.187	6385664	13389454	5693807
1	OK	0.000	2371584	33	10
2	OK	0.000	2371584	11	3
3	OK	0.000	2367488	19	6
4	OK	0.015	2371584	19	6
5	OK	0.000	2371584	19	6
6	OK	0.000	2379776	96	45
7	OK	0.000	2371584	85	56
8	OK	0.000	2367488	129	11
9	OK	0.015	2371584	131	12
10	OK	0.000	2383872	859	540
11	OK	0.000	2383872	828	573
12	OK	0.000	2371584	1340	11
13	OK	0.000	2383872	1325	12
14	OK	0.000	2387968	8292	5590
15	OK	0.015	2375680	8212	5706
16	OK	0.000	2375680	13298	111
17	OK	0.000	2371584	13354	12
18	OK	0.015	2392064	82372	56548
19	OK	0.015	2412544	82000	56993
20	OK	0.015	2412544	132796	1134
21	OK	0.031	2424832	133914	11
22	OK	0.109	2383872	819651	569557
23	OK	0.109	2572288	819689	569681
24	OK	0.125	2777088	1328670	11294
25	OK	0.125	2777088	1338543	11
26	OK	1.140	2392064	8196274	5693035
27	OK	1.093	4386816	8193816	5693807
28	OK	1.187	6307840	13286863	112020
29	OK	1.187	6385664	13389454	11
30	OK	1.187	6385664	13388564	11

Очередь

1.0 из 1.0 балла (оценивается)

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 109. Команда «-» означает изъятие элемента из очереди. Гарантируется, что размер очереди в процессе выполнения команд не превысит 106 элементов.

Формат входного файла

В первой строке содержится M ($1 \le M \le 106$) — число команд. В последующих строках содержатся команды, по одной в каждой строке.

Формат выходного файла

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.

```
#include <fstream>
using namespace std;


int main() {
	ifstream input("input.txt");
	ofstream output("output.txt");
	char command;
	long M;
	long head = 0;
	long tail = 0;
	input >> M;
	long* queue = new long[M];

	for (long i = 0; i < M; i++)
	{
		input >> command;
		if (command == '+')
		{

			input >> queue[tail];
			tail++;
		}
		else
		{
			output << queue[head] << '\n';
			head++;
		}
	}
	return 0;
}
```

Результат решения задачи

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		1.234	6385664	13389454	5693807
1	OK	0.015	2588672	20	7
2	OK	0.015	2588672	11	3
3	OK	0.000	2588672	19	6
4	OK	0.000	2584576	19	6
5	OK	0.000	2605056	96	45
6	OK	0.015	2588672	85	56
7	OK	0.015	2580480	129	12
8	OK	0.015	2588672	131	12
9	OK	0.000	2588672	859	538
10	OK	0.000	2588672	828	573
11	OK	0.000	2600960	1340	12
12	OK	0.015	2588672	1325	12
13	OK	0.000	2375680	8292	5589
14	OK	0.000	2375680	8212	5706
15	OK	0.015	2387968	13298	115
16	OK	0.031	2387968	13354	12
17	OK	0.015	2396160	82372	56552
18	OK	0.015	2396160	82000	56993
19	OK	0.031	2412544	132796	1124
20	OK	0.015	2424832	133914	12
21	OK	0.109	2584576	819651	569553
22	OK	0.109	2580480	819689	569681
23	OK	0.125	2777088	1328670	11296
24	OK	0.109	2777088	1338543	12
25	OK	1.109	4386816	8196274	5693025
26	OK	1.109	4390912	8193816	5693807
27	OK	1.234	6344704	13286863	112110
28	OK	1.156	6385664	13389454	10
29	OK	1.156	6385664	13388564	11

Скобочная последовательность

1.0 из 1.0 балла (оценивается)

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

Первая строка входного файла содержит число N ($1 \le N \le 500$) - число скобочных последовательностей, которые необходимо проверить. Каждая из следующих N строк содержит скобочную последовательность длиной от 1 до 104 включительно. В каждой из последовательностей присутствуют только скобки указанных выше видов.

Формат выходного файла

Для каждой строки входного файла выведите в выходной файл «YES», если соответствующая последовательность является правильной скобочной последовательностью, или «NO», если не является.

Реализация программы на C++

```
#include <fstream>
#include <string>
using namespace std;

string volid_seq(string sequence, char* stack, long top) {
	for (int i = 0; i < sequence.size(); i++)
	{
		switch (sequence[i])
		{
		case '(':
			stack[++top] = sequence[i];
			continue;
		case '[':
			stack[++top] = sequence[i];
			continue;
		case ')':
			if (stack[top] != '(' || top < 0)
				return "NO\n";
			top--;
			continue;
		case ']':
			if (stack[top] != '[' || top < 0)
				return "NO\n";
			top--;
			continue;
		}
	}

		if (top == -1) {
			return "YES\n";
		}
		return "NO\n";




}

int main() {
	ifstream input("input.txt");
	ofstream output("output.txt");
	long N, top;
	input >> N;
	char* stack = new char[10000];
	string sequence;
	for (long i = 0; i < N; i++)
	{
		input >> sequence;
		top = -1;
		output << volid_seq(sequence, stack, top);
	}
	return 0;
}
```

Результат решения задачи

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.109	2617344	5000885	2133
1	OK	0.000	2560000	31	22
2	OK	0.000	2560000	15	16
3	OK	0.015	2568192	68	66
4	OK	0.000	2572288	324	256
5	OK	0.000	2564096	1541	1032
6	OK	0.000	2564096	5880	2128
7	OK	0.015	2576384	50867	2129
8	OK	0.015	2588672	500879	2110
9	OK	0.109	2617344	5000884	2120
10	OK	0.109	2613248	5000885	2133

Очередь с минимумом

2.0 из 2.0 баллов (оценивается)

Имя входного файла:	input.txt
Имя выходного файла:	output.txt
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мегабайт

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.

Формат входного файла

В первой строке содержится M ($1 \le M \le 10^6$) — число команд. В последующих строках содержатся команды, по одной в каждой строке.

Формат выходного файла

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 или поиска минимума для пустой очереди не производится.

Реализация программы на C++

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

using namespace std;


int main() {
	long N;
	io >> N;
	char action;
	long* queue = new long[N];
	long offset = -1;
	long head = 0;
	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];

			while (minOffset - minHead >= 0 && minQueue[minOffset] >
queue[offset]) {
				minOffset--;
			}
			minQueue[++minOffset] = queue[offset];
			continue;
		case '-':

			if (queue[head] == minQueue[minHead]) {
				minHead++;
			}
			head++;
			continue;
		case '?':
			io << minQueue[minHead] << '\n';
			continue;
		}

	}
	return 0;

}
```

Результат решения задачи

№ 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	0.015	2224128	11	3
3	OK	0.000	2236416	22	6
4	OK	0.015	2236416	22	6
5	OK	0.000	2236416	36	9
6	OK	0.000	2232320	48	12
7	OK	0.000	2236416	76	35
8	OK	0.015	2224128	129	12
9	OK	0.000	2236416	67	48
10	OK	0.015	2224128	44	9
11	OK	0.015	2224128	45	9
12	OK	0.000	2224128	44	9
13	OK	0.000	2224128	45	9
42	OK	0.109	9625600	6465010	4002151
43	OK	0.062	19206144	13389342	12
44	OK	0.109	9617408	6462989	4000004
45	OK	0.078	12210176	6388899	1888890
46	OK	0.062	10321920	6500010	2000000
47	OK	0.078	12206080	6388899	1888890
48	OK	0.062	10321920	6500010	2000000
49	OK	0.062	19202048	13388086	12
50	OK	0.015	2236416	55	16
51	OK	0.000	2224128	705	225
52	OK	0.000	2236416	6506	2000
53	OK	0.015	2269184	65007	20000
54	OK	0.015	2871296	650008	200000
55	OK	0.078	12496896	6675213	2000000
56	OK	0.000	2224128	117	12
57	OK	0.000	2224128	1327	12
58	OK	0.015	2244608	13417	12
59	OK	0.031	2306048	133845	12
60	OK	0.000	3952640	1339319	12
61	OK	0.046	23203840	13388955	12

Quack

2.0 из 2.0 баллов (оценивается)

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 $(\alpha+\beta) \bmod 65536$
-	Вычитание: get α, get β, put $(\alpha-\beta) \bmod 65536$
*	Умножение: get α, get β, put $(\alpha\cdot\beta) \bmod 65536$
/	Целочисленное деление: get α, get β, put $\alpha \operatorname{div} \beta$ (будем считать, что $\alpha \operatorname{div} 0 = 0$)
%	Взятие по модулю: get α, get β, put $\alpha \bmod \beta$ (будем считать, что $\alpha \bmod 0 = 0$)
>[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 $\alpha \bmod 256$ в стандартный поток вывода
C[register]	Вывести регистр как символ: вывести символ с ASCII-кодом $\alpha \bmod 256$ (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

Реализация программы на C++

```
int main() {
	ifstream input("input.txt");
	ofstream output("output.txt");
	//создаем очередь
	queue<unsigned short> MainQueue = {};
	// создаем массив и с помощью цикла заполняем нулями
	map<char, unsigned short> registers;
	for (char i = 'a'; i <= 'z'; i++)
	{
		registers.insert(pair<char, unsigned short>(i, 0));
	}
	// фиксируем метки
	map <string, long> labels;
	//считываем команды
	string *commands = new string[100001];
	long N = -1;
	while (!input.eof()) {
		input >> commands[++N];
		if (commands[N][0] == ':') {
			labels.insert(pair<string, long>(commands[N].substr(1), N));
		}
	}
	if (commands[N].empty()) {
		N--;
	}
	input.close();
	long Number_of_command = 0;
	unsigned short a, b;
```

Парсер команд

```
do {
		switch (commands[Number_of_command][0])
		{
		case '+':
			a = MainQueue.front();
			MainQueue.pop();
			b = MainQueue.front();
			MainQueue.pop();
			MainQueue.push((a + b) % 65536);
			Number_of_command++;
			break;
		case '-':
			a = MainQueue.front();
			MainQueue.pop();
```

```
                b = MainQueue.front();
                MainQueue.pop();
                MainQueue.push((a - b) % 65536);
                Number_of_command++;
                break;
        case '*':
                a = MainQueue.front();
                MainQueue.pop();
                b = MainQueue.front();
                MainQueue.pop();
                MainQueue.push((a * b) % 65536);
                Number_of_command++;
                break;
        case '/':
                a = MainQueue.front();
                MainQueue.pop();
                b = MainQueue.front();
                MainQueue.pop();
                if (b == 0) {
                        MainQueue.push(0);
                }
                else {
                        MainQueue.push(a / b);
                }
                Number_of_command++;
                break;
        case '%':
                a = MainQueue.front();
                MainQueue.pop();
                b = MainQueue.front();
                MainQueue.pop();
                if (b == 0) {
                        MainQueue.push(0);
                }
                else {
                        MainQueue.push(a % b);
                }
                Number_of_command++;
                break;
        case '>':
                a = MainQueue.front();
                MainQueue.pop();
                registers[commands[Number_of_command][1]] = a;
                Number_of_command++;
                break;
        case '<':
                MainQueue.push(registers[commands[Number_of_command][1]]);
                Number_of_command++;
                break;
        case 'P':
                if (commands[Number_of_command].size() == 1) {
                        output << MainQueue.front() << '\n';
                        MainQueue.pop();
                }
                else {
                        output << registers[commands[Number_of_command][1]] << '\n';
                }
                Number_of_command++;
                break;
        case 'C':
                if (commands[Number_of_command].size() == 1) {
                        output << (char)(MainQueue.front() % 256);
                        MainQueue.pop();
                }
                else {
```

```
                        output << (char)(registers[commands[Number_of_command][1]] %
256);
                }
                Number_of_command++;
                break;
            case ':':
                Number_of_command++;
                break;
            case 'J':
                Number_of_command = labels[commands[Number_of_command].substr(1)] +
1;
                break;
            case 'Z':
                if (registers[commands[Number_of_command][1]] == 0) {
                    Number_of_command =
labels[commands[Number_of_command].substr(2)] + 1;
                }
                else {
                    Number_of_command++;
                }
                break;
            case 'E':
                if (registers[commands[Number_of_command][1]] ==
registers[commands[Number_of_command][2]]) {
                    Number_of_command =
labels[commands[Number_of_command].substr(3)] + 1;
                }
                else {
                    Number_of_command++;
                }
                break;
            case 'G':
                if (registers[commands[Number_of_command][1]] >
registers[commands[Number_of_command][2]]) {
                    Number_of_command =
labels[commands[Number_of_command].substr(3)] + 1;
                }
                else {
                    Number_of_command++;
                }
                break;
            case 'Q':
                Number_of_command = N;
                Number_of_command++;
                break;
            default:
                MainQueue.push(stoi(commands[Number_of_command]));
                Number_of_command++;
            }
        } while (Number_of_command <= N);
        output.close();
```

Результат решения задачи

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.078	9289728	1349803	250850
1	OK	0.015	5795840	69	6
2	OK	0.015	5791744	232	232
3	OK	0.000	5783552	3	0
4	OK	0.015	5791744	100	19
5	OK	0.015	5804032	56	58890
6	OK	0.015	5795840	67	30000
7	OK	0.046	5586944	67	30000
8	OK	0.015	5791744	55	30000
9	OK	0.015	5582848	461	62
10	OK	0.015	5787648	11235	21
11	OK	0.015	5582848	23748	42
12	OK	0.015	5582848	66906	9136
13	OK	0.015	5787648	7332	993
14	OK	0.015	5570560	4611	632
15	OK	0.000	5582848	37968	7332
16	OK	0.015	5787648	14	3
17	OK	0.015	5586944	70	14
18	OK	0.015	5566464	350	70
19	OK	0.015	5578752	1750	350
20	OK	0.000	5578752	8750	1750
21	OK	0.015	5582848	43750	8750
22	OK	0.031	5578752	218750	43750
23	OK	0.015	5582848	34606	4867
24	OK	0.046	5783552	683180	7
25	OK	0.046	5767168	683102	0
26	OK	0.078	9289728	1349803	0
27	OK	0.078	5586944	491572	247791
28	OK	0.062	5582848	491488	249618
29	OK	0.062	5586944	491600	249600
30	OK	0.078	5586944	491502	250850
31	OK	0.062	5582848	491416	249477
32	OK	0.078	5586944	491520	250262
33	OK	0.062	5586944	491317	246859
34	OK	0.078	5586944	491514	248199
35	OK	0.062	5586944	491557	249601