Университет ИТМО

Факультет программной инженерии и компьютерной техники

Лабораторная работа № 4

по дисциплине «Алгоритмы и структуры данных»

Openedu – неделя 4

Подготовил:

студент группы P3217

Бураков Илья Алексеевич

Преподаватели:

Романов Алексей Андреевич

Волчек Дмитрий Геннадьевич

Санкт-Петербург, 2019

Стек

Условие

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

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

В первой строке входного файла содержится M ($1 \le M \le 10^6$) — 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.

Пример

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

Решение

openedu/week4/lab4_1.cpp

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

int main() {
	auto stack = new int[1e6];

	int n;
	io >> n;

	for (int i = 0; i < n; i++) {
		// read command
		char c;
		io >> c;
```

```
            if (c == '+') {
                  // push
                  io >> *(stack++);
            } else {
                  // pop
                  io << *(--stack) << '\n';
            }
      }

      return 0;
}
```

Результаты

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.156	19202048	13389454	5693807
1	OK	0.000	2240512	33	10
2	OK	0.015	2228224	11	3
3	OK	0.015	2228224	19	6
4	OK	0.000	2228224	19	6
5	OK	0.015	2228224	19	6
6	OK	0.000	2228224	96	45
7	OK	0.015	2228224	85	56
8	OK	0.015	2228224	129	11
9	OK	0.015	2240512	131	12
10	OK	0.000	2240512	859	540
11	OK	0.000	2244608	828	573
12	OK	0.000	2224128	1340	11
13	OK	0.000	2224128	1325	12
14	OK	0.000	2236416	8292	5590
15	OK	0.000	2228224	8212	5706
16	OK	0.015	2224128	13298	111
17	OK	0.000	2240512	13354	12
18	OK	0.000	2228224	82372	56548
19	OK	0.000	2240512	82000	56993
20	OK	0.015	2281472	132796	1134
21	OK	0.000	2265088	133914	11
22	OK	0.031	2641920	819651	569557
23	OK	0.015	2838528	819689	569681
24	OK	0.015	3538944	1328670	11294
25	OK	0.015	3547136	1338543	11
26	OK	0.156	10014720	8196274	5693035
27	OK	0.140	12013568	8193816	5693807
28	OK	0.062	19021824	13286863	112020
29	OK	0.062	19202048	13389454	11

30	OK	0.046	19202048	13388564	11

Очередь

Условие

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

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

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

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.

Пример

input.txt	output.txt
4 + 1 + 10 - -	1 10

Решение

openedu/week4/lab4_2.cpp

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

using namespace std;

#define SIZE 1000000
#define CYCLE_INC(v, sz) (((v) + 1 ) % (sz))

int main() {
	auto queue = new int[SIZE];
	int head = 0, tail = 0;

	int n;
	io >> n;

	for (int i = 0; i < n; i++) {
```

```
		// read command
		char c;
		io >> c;

		if (c == '+') {
			// push
			io >> queue[tail];
			tail = CYCLE_INC(tail, SIZE);
		} else {
			// pop
			io << queue[head] << '\n';
			head = CYCLE_INC(head, SIZE);
		}
	}

	return 0;
}
```

Результаты

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.156	19202048	13389454	5693807
1	OK	0.015	2240512	20	7
2	OK	0.015	2228224	11	3
3	OK	0.000	2236416	19	6
4	OK	0.000	2228224	19	6
5	OK	0.015	2228224	96	45
6	OK	0.000	2240512	85	56
7	OK	0.000	2228224	129	12
8	OK	0.000	2228224	131	12
9	OK	0.000	2240512	859	538
10	OK	0.000	2232320	828	573
11	OK	0.015	2228224	1340	12
12	OK	0.000	2228224	1325	12
13	OK	0.015	2228224	8292	5589
14	OK	0.015	2240512	8212	5706
15	OK	0.000	2228224	13298	115
16	OK	0.000	2228224	13354	12
17	OK	0.015	2260992	82372	56552
18	OK	0.000	2244608	82000	56993
19	OK	0.000	2265088	132796	1124
20	OK	0.000	2256896	133914	12
21	OK	0.031	2838528	819651	569553
22	OK	0.015	2838528	819689	569681
23	OK	0.015	3543040	1328670	11296
24	OK	0.015	3551232	1338543	12
25	OK	0.140	12013568	8196274	5693025
26	OK	0.156	12013568	8193816	5693807

27	OK	0.062	19062784	13286863	112110
28	OK	0.062	19202048	13389454	10
29	OK	0.046	19202048	13388564	11

Скобочная последовательность

Условие

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

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 10^4 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.

Пример

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

Решение

openedu/week4/lab4_3.cpp

```
#include "edx-io.hpp"
#include <stack>
#include <string>

using namespace std;

char invert_brace(char brace) {
	switch (brace) {
		case ')': return '(';
		case ']': return '[';
	}
}

bool check_braces(string sequence) {
	auto st = stack<char>();
	for (auto c : sequence) {
		if (c == '(' || c == '[') {
			st.push(c);
		} else {
			if (!st.empty() && st.top() == invert_brace(c)) {
				st.pop();
			} else {
				return false;
			}
		}
	}
	return st.size() == 0;
}

int main() {
	auto stack = new int[1e6];

	int n;
	io >> n;

	for (int i = 0; i < n; i++) {
		auto sequence = string();
		io >> sequence;

		io << (check_braces(sequence) ? "YES" : "NO") << '\n';
	}

	return 0;
}
```

Результаты

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.062	7016448	5000885	2133
1	OK	0.000	2244608	31	22
2	OK	0.015	2232320	15	16
3	OK	0.000	2236416	68	66
4	OK	0.000	2248704	324	256
5	OK	0.000	2236416	1541	1032
6	OK	0.000	2248704	5880	2128

7	OK	0.000	2236416	50867	2129
8	OK	0.015	2326528	500879	2110
9	OK	0.062	7016448	5000884	2120
10	OK	0.062	7016448	5000885	2133

Очередь с минимумом

Условие

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.

Пример

input.txt	output.txt
7 + 1 ? + 10 ? - ? -	1 1 10

Решение

openedu/week4/lab4_4.cpp

```
#include "edx-io.hpp"
#include <queue>
```

```
#include <set>
#include <algorithm>

using namespace std;

#define SIZE 1000000
#define CYCLE_INC(v, sz) (((v) + 1 ) % (sz))


int main() {
	auto queue = new int[SIZE];
	int head = 0, tail = 0;
	auto pq = multiset<int>();

	int n;
	io >> n;

	for (int i = 0; i < n; i++) {
		// read command
		char c;
		io >> c;

		if (c == '+') {
			// push
			io >> queue[tail];
			pq.insert(queue[tail]);
			tail = CYCLE_INC(tail, SIZE);
		} else if (c == '-') {
			// pop
			//io << queue[head] << '\n';
			pq.erase(pq.find(queue[head]));
			head = CYCLE_INC(head, SIZE);
		} else {
			io << *pq.begin() << '\n';
		}
	}

	return 0;
}
```

Результаты

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.968	67559424	13389342	4002151
1	OK	0.000	2248704	29	10
2	OK	0.000	2244608	11	3
3	OK	0.000	2232320	22	6
4	OK	0.000	2248704	22	6
5	OK	0.000	2248704	36	9
6	OK	0.000	2236416	48	12
7	OK	0.000	2236416	76	35
8	OK	0.015	2236416	129	12
9	OK	0.000	2236416	67	48
10	OK	0.000	2232320	44	9
11	OK	0.000	2248704	45	9

12	OK	0.000	2236416	44	9
13	OK	0.015	2232320	45	9
14	OK	0.000	2248704	721	384
15	OK	0.000	2248704	1340	12
16	OK	0.000	2256896	640	407
17	OK	0.000	2244608	445	90
18	OK	0.000	2244608	456	100
19	OK	0.000	2248704	445	90
20	OK	0.015	2248704	456	100
21	OK	0.000	2240512	6616	3812
22	OK	0.000	2293760	13389	12
23	OK	0.000	2269184	6461	4008
24	OK	0.000	2269184	4896	1140
25	OK	0.000	2281472	5007	1250
26	OK	0.015	2269184	4896	1140
27	OK	0.000	2265088	5007	1250
28	OK	0.015	2265088	64907	39589
29	OK	0.015	2555904	133814	12
30	OK	0.000	2363392	64675	39996
31	OK	0.015	2367488	53897	13890
32	OK	0.000	2428928	55008	15000
33	OK	0.000	2367488	53897	13890
34	OK	0.000	2437120	55008	15000
35	OK	0.015	2646016	645271	404305
36	OK	0.062	8466432	1338956	12
37	OK	0.031	4145152	646300	400008
38	OK	0.031	4976640	588898	163890
39	OK	0.031	4972544	600009	175000
40	OK	0.031	4972544	588898	163890
41	OK	0.031	4976640	600009	175000
42	OK	0.203	9781248	6465010	4002151
43	OK	0.968	67555328	13389342	12
44	OK	0.468	23945216	6462989	4000004
45	OK	0.203	33193984	6388899	1888890
46	OK	0.218	33292288	6500010	2000000
47	OK	0.218	33193984	6388899	1888890
48	OK	0.218	33292288	6500010	2000000
49	OK	0.968	67551232	13388086	12
50	OK	0.015	2236416	55	16
51	OK	0.000	2248704	705	225
52	OK	0.000	2252800	6506	2000
53	OK	0.015	2355200	65007	20000

54	OK	0.015	3944448	650008	200000
55	OK	0.218	22663168	6675213	2000000
56	OK	0.015	2252800	117	12
57	OK	0.000	2260992	1327	12
58	OK	0.000	2289664	13417	12
59	OK	0.015	2560000	133845	12
60	OK	0.031	8466432	1339319	12
61	OK	0.296	67559424	13388955	12

Quack

Условие

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

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 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

Формат входного файла

Входной файл содержит синтаксически корректную программу на языке Quack. Известно, что программа завершает работу не более чем за 10^5 шагов. Программа содержит не менее одной и не более 10^5 инструкций. **Метки имеют длину от 1 до 10 и состоят из цифр и латинских букв.**

Формат выходного файла

Выведите содержимое стандартного потока вывода виртуальной машины в выходной файл.

Примеры

input.txt	output.txt
100 0 :start >a Zaend <a <a 1 + - >b <b Jstart :end P	5050

Решение

openedu/week4/lab4_5.cpp

```
#include "edx-io.hpp"
#include <queue>
#include <vector>
#include <string>
#include <map>
#include <tuple>
#include <fstream>
#include <assert.h>

using namespace std;

typedef uint16_t val_t;

// registers
vector<val_t> reg('z' - 'a' + 1, 0);
```

```
// (machine) queue
auto mqueue = queue<val_t>();

// loaded program (instructions reside here)
auto prog = vector<string>(100000);

// instruction pointer (index of "prog")
int ip = 0;

// mapping of labels to their target ip (index of "prog")
auto labels = map<string, int>();

// terminator flag
bool is_terminating = false;


tuple<val_t, val_t> pop_two() {
	val_t a = mqueue.front();
	mqueue.pop();
	val_t b = mqueue.front();
	mqueue.pop();

	return { a, b };
}

val_t pop_one() {
	val_t a = mqueue.front();
	mqueue.pop();
	return a;
}

#define cmd_reg(pos) (reg[instr[pos] - 'a'])
#define cmd_label_ip(pos) (labels[instr.substr(pos)])

void exec(string instr) {
	// if whole instruction is a number
	if (find_if(instr.begin(), instr.end(), [](char c) {return !isdigit(c); })
== instr.end()) {
		mqueue.push(stoi(instr));
		return;
	}

	// process general instructions
	val_t a, b;
	switch (instr[0]) {
		case '+': {
			tie(a, b) = pop_two();
			mqueue.push(a + b);
		} break;
		case '-': {
			tie(a, b) = pop_two();
			mqueue.push(a - b);
		} break;
		case '*': {
			tie(a, b) = pop_two();
			mqueue.push(a * b);
		} break;
		case '/': {
			tie(a, b) = pop_two();
			val_t result = (b == 0) ? 0 : a / b;
			mqueue.push(result);
		} break;
		case '%': {
			tie(a, b) = pop_two();
			val_t result = (b == 0) ? 0 : a % b;
			mqueue.push(result);
```

```
                } break;
                case '>': {
                        cmd_reg(1) = pop_one();
                } break;
                case '<': {
                        mqueue.push(cmd_reg(1));
                } break;
                case 'P': {
                        val_t printed = (instr.size() == 1) ? pop_one() : cmd_reg(1);
                        io << printed << '\n';
                } break;
                case 'C': {
                        char printed = (instr.size() == 1) ? pop_one() : cmd_reg(1);
                        io << printed;
                } break;
                case ':': {
                        // pass (no need to execute a label definition)
                } break;
                case 'J': {
                        ip = cmd_label_ip(1);
                } break;
                case 'Z': {
                        if (cmd_reg(1) == 0) {
                                ip = cmd_label_ip(2);
                        }
                } break;
                case 'E': {
                        if (cmd_reg(1) == cmd_reg(2)) {
                                ip = cmd_label_ip(3);
                        }
                } break;
                case 'G': {
                        if (cmd_reg(1) > cmd_reg(2)) {
                                ip = cmd_label_ip(3);
                        }
                } break;
                case 'Q': {
                        is_terminating = true;
                } break;
        }
}

int main() {
        auto cin = ifstream("input.txt");

        // load program and mark labels (first pass)
        string instr;
        int i = 0;
        while (cin >> instr) {
                if (instr[0] == ':') {
                        labels[instr.substr(1)] = i + 1;
                }
                prog[i++] = instr;
        }

        // i is now size of the program
        // start executing it
        while (!is_terminating) {
                instr = prog[ip++];
                exec(instr);
                if (ip == i) {
                        // reached end of program
                        break;
                }
        }

        cin.close();
```

```
	return 0;
}
```

Результаты

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.093	9256960	1349803	250850
1	OK	0.015	5120000	69	6
2	OK	0.000	5124096	232	232
3	OK	0.015	5111808	3	0
4	OK	0.015	5124096	100	19
5	OK	0.015	5124096	56	58890
6	OK	0.015	5120000	67	30000
7	OK	0.015	5124096	67	30000
8	OK	0.015	5124096	55	30000
9	OK	0.015	5124096	461	62
10	OK	0.000	5120000	11235	21
11	OK	0.000	5120000	23748	42
12	OK	0.015	5124096	66906	9136
13	OK	0.015	5124096	7332	993
14	OK	0.015	5124096	4611	632
15	OK	0.015	5124096	37968	7332
16	OK	0.000	5124096	14	3
17	OK	0.000	5124096	70	14
18	OK	0.015	5124096	350	70
19	OK	0.015	5124096	1750	350
20	OK	0.000	5124096	8750	1750
21	OK	0.015	5120000	43750	8750
22	OK	0.031	5120000	218750	43750
23	OK	0.015	5120000	34606	4867
24	OK	0.046	5685248	683180	7
25	OK	0.062	5677056	683102	0
26	OK	0.093	9256960	1349803	0
27	OK	0.062	5120000	491572	247791
28	OK	0.062	5124096	491488	249618
29	OK	0.062	5124096	491600	249600
30	OK	0.062	5124096	491502	250850
31	OK	0.062	5124096	491416	249477
32	OK	0.062	5124096	491520	250262
33	OK	0.046	5124096	491317	246859
34	OK	0.062	5120000	491514	248199
35	OK	0.062	5120000	491557	249601