**Министерство образования и науки Российской Федерации**

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ,   
МЕХАНИКИ И ОПТИКИ**

Факультет программной инженерии и компьютерной техники

Кафедра информатики и прикладной математики   
Направление подготовки 09.03.04 Программная инженерия

Дисциплина «Алгоритмы и структуры данных»

**ОТЧЁТ**

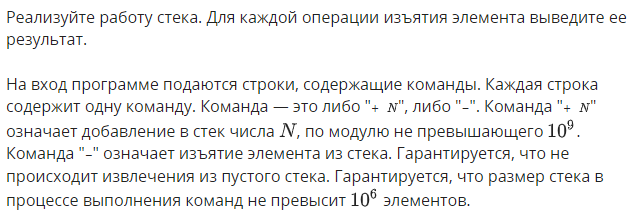
по лабораторной работе №4

Студент Кочарян С.А. группы P3218

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2019

### **Стек**



#include "edx-io.hpp"

#include <string>

using namespace std;

int main()

{

int quantity;

io >> quantity;

long\* elements = new long[quantity];

char action;

int last\_element = 0;

for (int i = 0; i < quantity; i++) {

io >> action;

if (action == '-') {

last\_element--;

io << elements[last\_element] << '\n';

}

else {

long number;

io >> number;

elements[last\_element] = number;

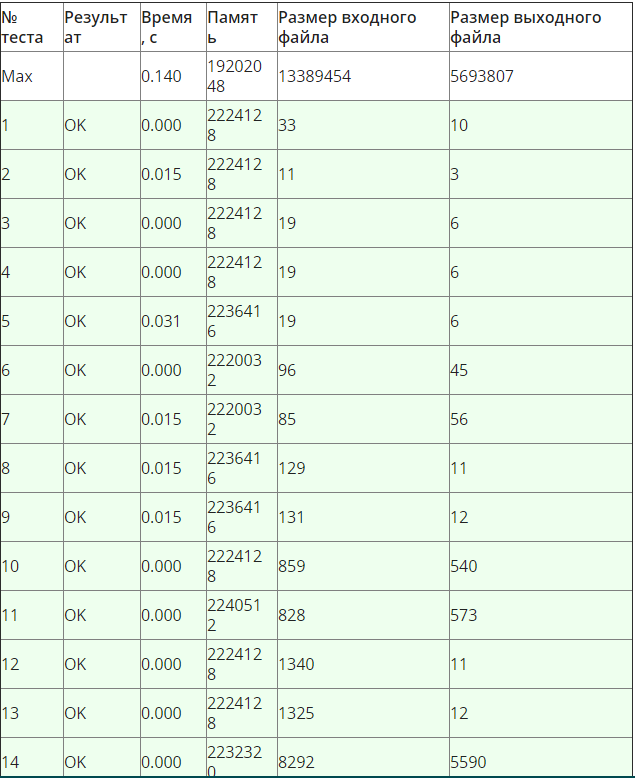
last\_element++;

}

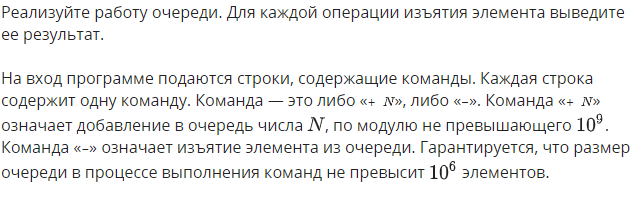
}

return 0;

}



## Очередь



#include "edx-io.hpp"

#include <string>

using namespace std;

int main()

{

int quantity;

io >> quantity;

long\* elements = new long[quantity];

char action;

int last\_element = 0;

int left = 0;

for (int i = 0; i < quantity; i++) {

io >> action;

if (action == '-') {

io << elements[left++] << '\n';

}

else {

long number;

io >> number;

elements[last\_element] = number;

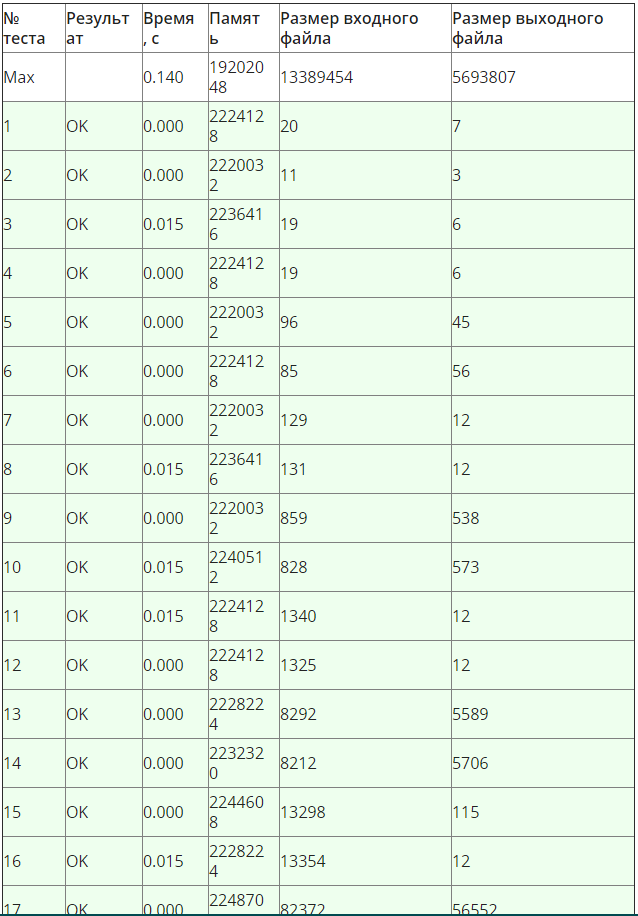
last\_element++;

}

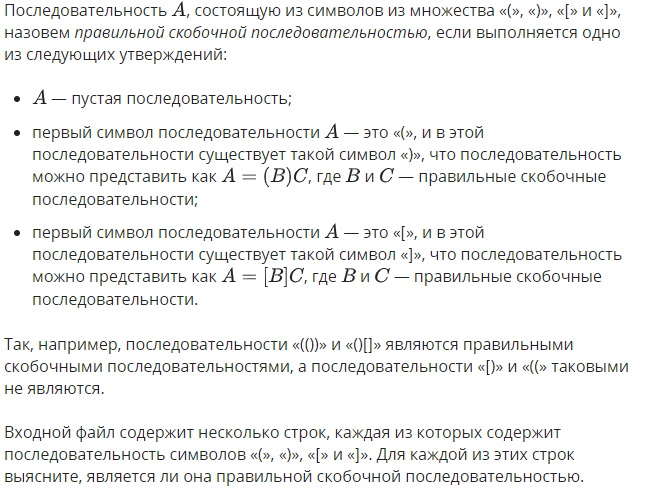
}

return 0;

}



## Скобочная последовательность



// asd.cpp : Defines the entry point for the console application.

//

#include "edx-io.hpp"

#include <string>

using namespace std;

int main()

{

int quantity;

io >> quantity;

for (int i = 0; i < quantity; i++) {

string line;

io >> line;

bool res = true;

string curr\_line = "";

for (int j = 0; j < line.length(); j++) {

char last\_el;

switch (line[j])

{

case '(':

curr\_line += "(";

break;

case '[':

curr\_line += "[";

break;

case ']':

if (curr\_line.length() == 0 || curr\_line[curr\_line.length() - 1] != '[') {

res = false;

}

curr\_line = curr\_line.substr(0, curr\_line.length() - 1);

break;

case ')':

if (curr\_line.length() == 0 || curr\_line[curr\_line.length() - 1] != '(') {

res = false;

}

curr\_line = curr\_line.substr(0, curr\_line.length() - 1);

break;

}

}

if (curr\_line.length() != 0) {

res = false;

}

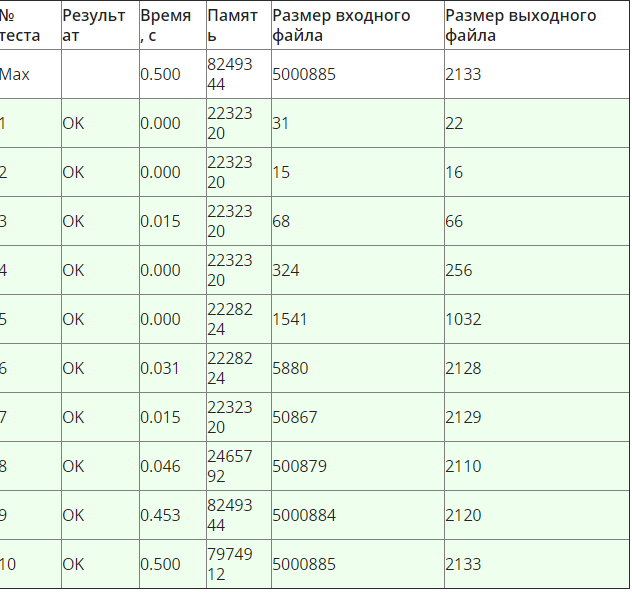
string answer = (res) ? "YES" : "NO";

io << answer << '\n';

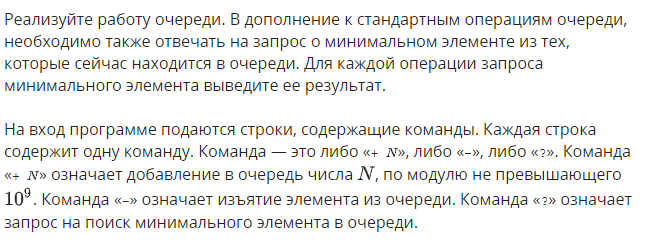
}

return 0;

}



## Очередь с минимумом



#include "edx-io.hpp"

//#include "stdafx.h"

#include <string>

#include <iostream>

using namespace std;

int main()

{

int quantity;

io >> quantity;

char command;

long\* elements = new long[quantity];

int last = -1;

int left = 0;

long \*mins = new long[quantity];

int min\_last = -1;

int min\_left = 0;

for (long i = 0; i < quantity; i++) {

io >> command;

if (command == '+') {

io >> elements[++last];

while (min\_last - min\_left >= 0 && mins[min\_last] > elements[last]) {

min\_last--;

}

mins[++min\_last] = elements[last];

}

else if (command == '-') {

if (elements[left] == mins[min\_left]) {

min\_left++;

}

left++;

}

else{

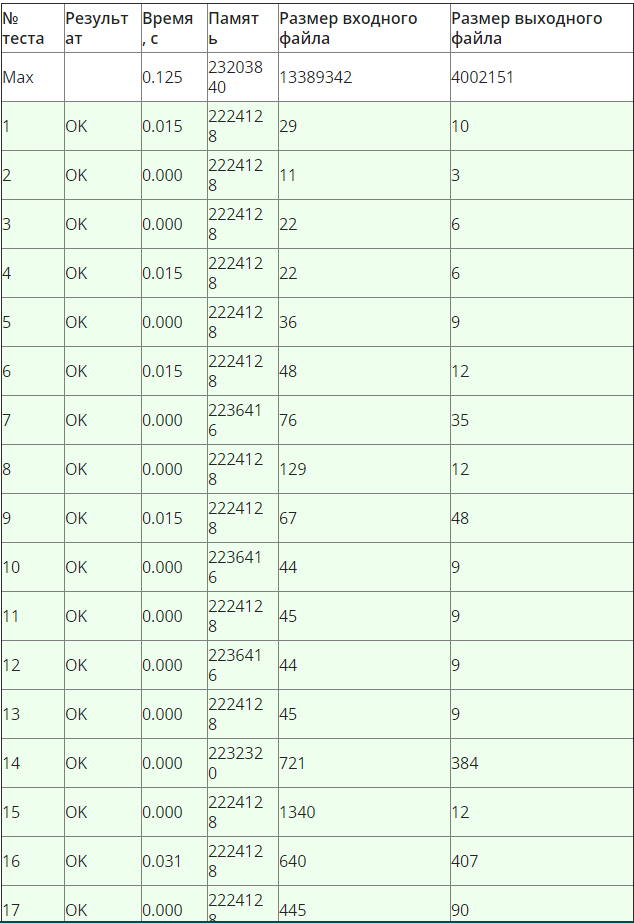
io << mins[min\_left] << '\n';

}

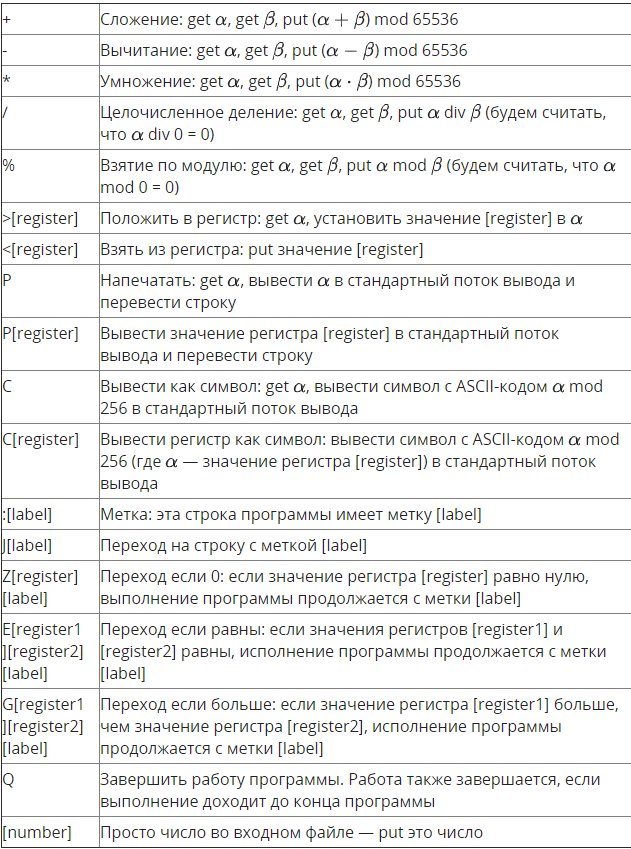
}

return 0;

}



## Quack



//#include "stdafx.h"

#include <fstream>

#include <string>

#include <iostream>

#include <queue>

#include <map>

using namespace std;

int main()

{

ifstream input("input.txt");

ofstream output("output.txt");

map<char, unsigned short> regs = {

{'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}

};

queue<unsigned short> queue = {};

map<string, long> labels;

string\* actions = new string[100001];

long ind = -1;

while (!input.eof()) {

input >> actions[++ind];

if (actions[ind][0] == ':') {

labels.insert(pair<string, long>(actions[ind].substr(1), ind));

}

}

if (actions[ind].empty()) {

ind--;

}

input.close();

long P = 0;

unsigned short num1, num2;

do {

switch (actions[P][0])

{

case '+':

num1 = queue.front();

queue.pop();

num2 = queue.front();

queue.pop();

queue.push((num1 + num2) % 65536);

P++;

break;

case '-':

num1 = queue.front();

queue.pop();

num2 = queue.front();

queue.pop();

queue.push((num1 - num2) % 65536);

P++;

break;

case '\*':

num1 = queue.front();

queue.pop();

num2 = queue.front();

queue.pop();

queue.push((num1 \* num2) % 65536);

P++;

break;

case '/':

num1 = queue.front();

queue.pop();

num2 = queue.front();

queue.pop();

if (num2 == 0) {

queue.push(0);

}

else {

queue.push(num1 / num2);

}

P++;

break;

case '%':

num1 = queue.front();

queue.pop();

num2 = queue.front();

queue.pop();

if (num2 == 0) {

queue.push(0);

}

else {

queue.push(num1 % num2);

}

P++;

break;

case '>':

num1 = queue.front();

queue.pop();

regs[actions[P][1]] = num1;

P++;

break;

case '<':

queue.push(regs[actions[P][1]]);

P++;

break;

case 'P':

if (actions[P].size() == 1) {

output << queue.front() << '\n';

queue.pop();

}

else {

output << regs[actions[P][1]] << '\n';

}

P++;

break;

case 'C':

if (actions[P].size() == 1) {

output << (char)(queue.front() % 256);

queue.pop();

}

else {

output << (char)(regs[actions[P][1]] % 256);

}

P++;

break;

case ':':

P++;

break;

case 'J':

P = labels[actions[P].substr(1)] + 1;

break;

case 'Z':

if (regs[actions[P][1]] == 0) {

P = labels[actions[P].substr(2)] + 1;

}

else {

P++;

}

break;

case 'E':

if (regs[actions[P][1]] == regs[actions[P][2]]) {

P = labels[actions[P].substr(3)] + 1;

}

else {

P++;

}

break;

case 'G':

if (regs[actions[P][1]] > regs[actions[P][2]]) {

P = labels[actions[P].substr(3)] + 1;

}

else {

P++;

}

break;

case 'Q':

P = ind;

P++;

break;

default:

queue.push(stoi(actions[P]));

P++;

}

} while (P <= ind);

output.close();

return 0;

}

