**Федеральное государственное автономное образовательное учреждение высшего образования   
"Национальный исследовательский университет   
"Высшая школа экономики"**

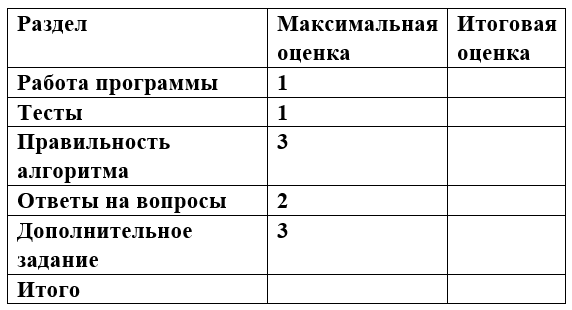
**Московский институт электроники и математики им. А. Н. Тихонова НИУ ВШЭ**

**Департамент компьютерной инженерии**

**Курс: Алгоритмизация и программирование**

**ОТЧЕТ**

**по лабораторной работе №10**

****

**Студент:   
Свистунов Андрей Сергеевич**

**Группа: БИВ234**

**Вариант: №277 (задания № (2, 2), 12)**

**Руководитель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Оценка\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Дата сдачи\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

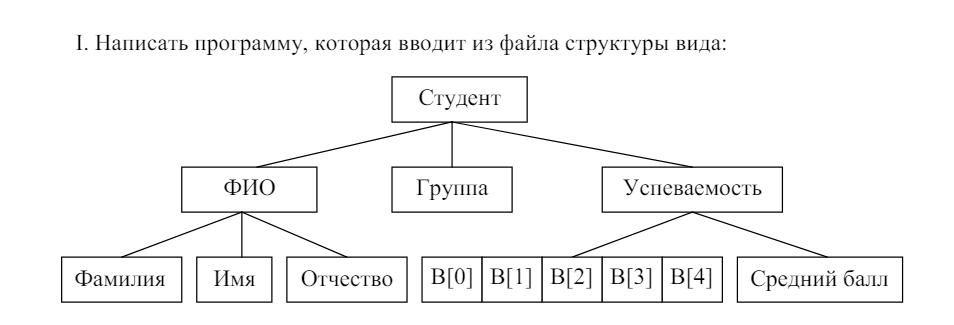
Москва 2024

# Содержание

[Содержание 2](#_Toc164641747)

[Листинг программы 4](#_Toc164641748)

[Распечатка тестов и результатов к программе 6](#_Toc164641749)

**2. Лабораторная работа 10. "Списки. Динамические переменные"**

Имя файла задает пользователь. Массив баллов B[0:4] содержит данные о результатах сдачи экзаменов, по 10-балльной шкале. Каждое поле структуры занимает в файле одну строку, а массив оценок размещается на отдельной строке. Средний балл не записан в файле, а вычисляется в процессе чтения данных. Структуры размещаются в:

Имя файла задает пользователь. Массив баллов B[0:4] содержит данные о результатах сдачи экзаменов, по 10-балльной шкале. Каждое поле структуры занимает в файле одну строку, а массив оценок размещается на отдельной строке. Средний балл не записан в файле, а вычисляется в процессе чтения данных. Структуры размещаются в: очереди;

реализованном(ой) с помощью: циклического списка

II. Для полученного списка программа выполняет следующие действия:

сформироватьновый линейный однонаправленный стек,содержащий

данные о студентах, не имеющих задолженностей, т.е. баллов, меньших

4 баллов

# Листинг программы

#include <iostream>

#include <string>

#include <fstream>

using namespace std;

struct FIO{

string name;

string surname;

string patronmic;

};

struct Progress{

float mean\_mark;

int marks[5];

};

class Student{

public:

string group;

FIO\* full\_name;

Progress\* progress;

Student(){

full\_name = new FIO();

progress = new Progress();

}

Student(FIO\* fl\_nm, string grp, Progress\* prgrss){

//full\_name = new FIO();

full\_name = fl\_nm;

//progress = new Progress();

progress = prgrss;

group = grp;

}

~Student(){

}

void display\_student(){

cout << "ФИО студнета: " << full\_name->name << ' ' << full\_name->surname << ' ' << full\_name->patronmic << endl;

cout << "Группа: " << group << endl;

cout << "Успеваемость студента: " << endl;

cout << "\tОценки студента (оценка - количество оценок): " << endl;

for (int i = 0; i < 5; i++){

cout << "\t\t" << i + 1 << " - " << progress->marks[i] << endl;

}

cout << "\tСредний балл: " << progress->mean\_mark << endl << endl;

}

};

class Node {

public:

Student\* data;

Node\* next;

Node(Student\* data) {

this->data = data;

next = nullptr;

}

};

class Queue {

private:

Node\* head;

Node\* tail;

int size;

public:

Queue() {

head = nullptr;

tail = nullptr;

size = 0;

}

~Queue(){

}

void enqueue(Student\* data) {

Node\* newNode = new Node(data);

if (head == nullptr) {

head = newNode;

tail = newNode;

head->next = tail;

tail->next = head;

} else {

tail->next = newNode;

tail = newNode;

tail->next = head;

}

size++;

}

Student\* dequeue() {

if (head == nullptr) {

cerr << "Очередь пуста" << endl;

exit(1);

}

Node\* temp = head;

Student\* data = temp->data;

head = head->next;

tail->next = head;

delete temp;

size--;

return data;

}

void displayQueue() {

Node\* temp = head;

while (temp->next != head) {

temp->data->display\_student();

temp = temp->next;

}

temp->data->display\_student();

cout << endl;

}

int getSize() {

return size;

}

};

Queue read\_file(){

cout << "Введите имя файла (вместе с расширением)... ";

string filename;

cin >> filename;

Queue a;

ifstream file(filename);

if (!file.is\_open()){

cerr << "Ошибка открытия файла" << endl;

exit(1);

}

while (!file.eof()){

FIO\* full = new FIO();

file >> full->name;

if (file.eof()){

cout << "Неудалось заполнить данные о студенте, так как количество строк неполное" << endl;

break;

}

file >> full->surname;

if (file.eof()){

cout << "Неудалось заполнить данные о студенте, так как количество строк неполное" << endl;

break;

}

file >> full->patronmic;

if (file.eof()){

cout << "Неудалось заполнить данные о студенте, так как количество строк неполное" << endl;

break;

}

string\* group = new string();

file >> \*group;

if (file.eof()){

cout << "Неудалось заполнить данные о студенте, так как количество строк неполное" << endl;

break;

}

Progress\* progress = new Progress();

for (int i = 0; i < 5; i++){

file >> progress->marks[i];

if (i < 4 && file.eof()){

cout << "Неудалось заполнить данные о студенте, так как количество строк неполное" << endl;

break;

}

}

Student\* student = new Student(full, \*group, progress);

a.enqueue(student);

}

file.close();

return a;

}

class Stack{

private:

Node\* head;

public:

Stack(){

head = nullptr;

}

~Stack(){

}

void add\_element(Student\* data){

if (head == nullptr){

Node\* newNode = new Node(data);

head = newNode;

}

else{

Node\* newNode = new Node(data);

newNode->next = head;

head = newNode;

}

}

Student\* pop\_element(){

if (head == nullptr){

cerr << "Стэк пустой" << endl;

exit(1);

}

Node\* prev = head;

Student\* ans = head->data;

head = head->next;

delete prev;

return ans;

}

void show(){

if (head == nullptr){

cout << "Стэк пуст" << endl;

return;

}

Node\* current = head;

while (current){

current->data->display\_student();

current = current->next;

}

}

void del\_element(Node\* element){

if (head == nullptr){

cout << "Нечего удалять\nСтэк пуст" << endl << endl;

}

else if(element == nullptr){

cout << "Нечего удалять\nЕлемент указывает на ничего" << endl << endl;

}

else if (element == head){

Node\* prev = head;

head = head->next;

delete prev;

}

else if (element != head){

Node\* current = head;

while (current){

if (current->next == element){

Node\* prev = current->next;

current->next = current->next->next;

delete prev;

return;

}

current = current->next;

}

cout << "Элемент не содержится в стэке" << endl;

}

}

void filter\_students(){

cout << endl << endl << endl << "Происходит фильтрация..." << endl << endl;

Node\* current = head;

while (current){

bool flag = false;

for (int i = 0; i < 3; i++){

if (current->data->progress->marks[i] > 0){

flag = true;

break;

}

}

if (!flag)

current = current->next;

else{

Node\* prev = current;

current = current->next;

this->del\_element(prev);

}

}

cout << "Фильтрация проведена" << endl;

}

void stack\_in\_file(){

cout << "Введите имя файла (вместе с расширением)...\n";

string filename;

cin >> filename;

ofstream out\_file(filename);

if (!out\_file){

cout << "Некорректный ввод файл...\n";

exit(1);

}

Node\* current = head;

while (current){

out\_file << current->data->full\_name->name << endl;

out\_file << current->data->full\_name->surname << endl;

out\_file << current->data->full\_name->patronmic << endl;

out\_file << current->data->group << endl;

for (int i = 0; i < 5; i ++)

out\_file << current->data->progress->marks[i] << endl;

current = current->next;

}

cout << "Запись завершена" << endl;

out\_file.close();

}

};

int main() {

cout << "Лабораторная работа №10\nВыполнил ученик группы БИВ234\nСвистунов Андрей\n";

cout << "Задание 1" << endl;

cout << endl;

Queue bam;

bam = read\_file();

bam.displayQueue();

cout << "\n\nЗадание 2\n";

Stack bambam;

while (bam.getSize() > 0){

bambam.add\_element(bam.dequeue());

}

bambam.filter\_students();

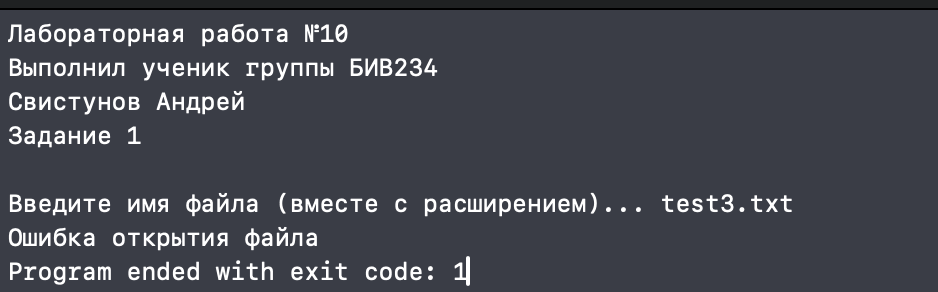
//bambam.show();

bambam.stack\_in\_file();

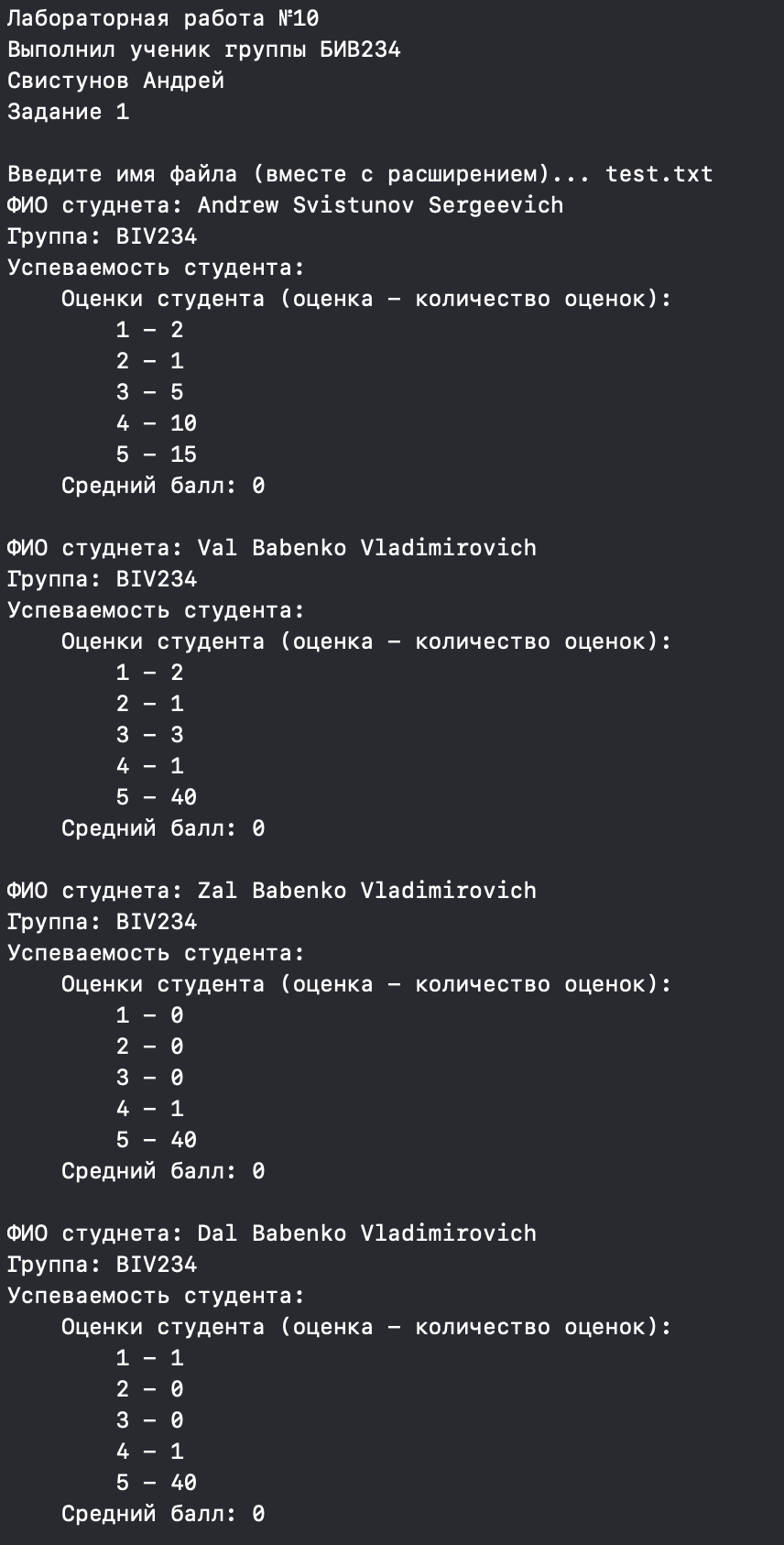
}

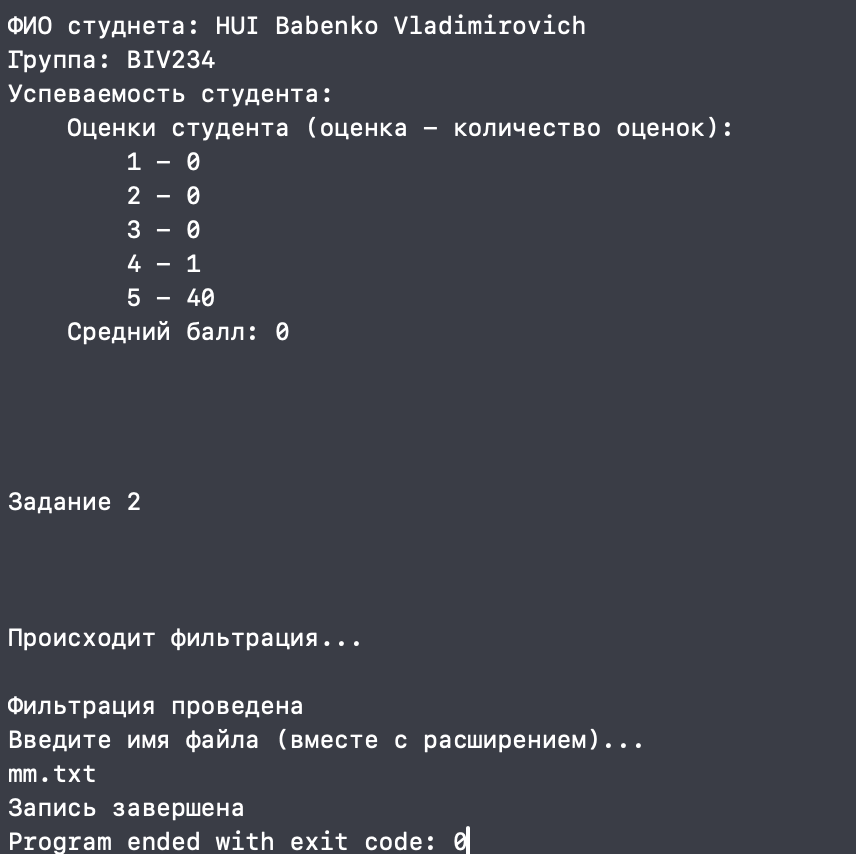
# Распечатка тестов и результатов к программе

1. Считывание с файла несуществующего файла:



1. Считывание с файла test.txt и запись в файл mm.txt (файлы приложены)





1. Считывание с файла test2.txt и запись в файл mmm.txt (файлы приложены)

