Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Пермский национальный исследовательский**

**политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЁТ**

**по лабораторной работе №18.10**

Дисциплина: «Информатика»

Тема: “ Объектно-ориентированное программирование.

Сохранение данных в файле с использованием потоком.”

Вариант 15

Выполнил:

Студент группы ИВТ-20-2Б Тедеев Алесандр Зурабович

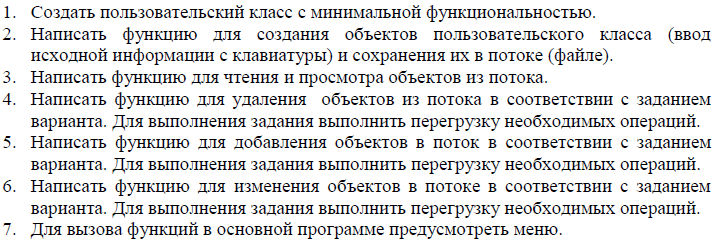
Проверила:

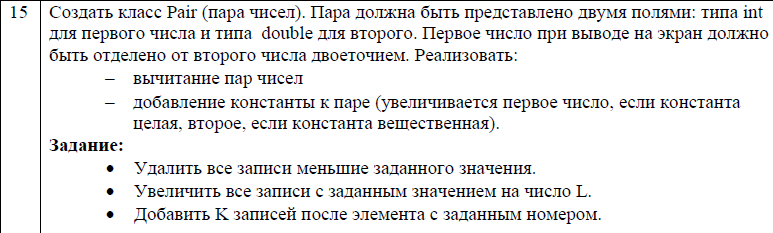
Доцент кафедры ИТАС

Полякова О.А.

Пермь, 2021

**Постановка задачи**





2

**Анализ задачи**

**1.** Для решения задачи необходимо:

**1.1.** Организовать класс Pair с полями \_first и \_second для хранения числа в целочисленном типе и дробном типе данных.

**1.2.** Организовать необходимые методы для ввода данных в поля \_first и \_second: гетторы, сетторы, конструкторы, деструкторы.

**1.3.** Организовать перегрузку оператора >> дружественной классу Pair.

**1.4.** Организовать перегрузку оператора << дружественной классу Pair.

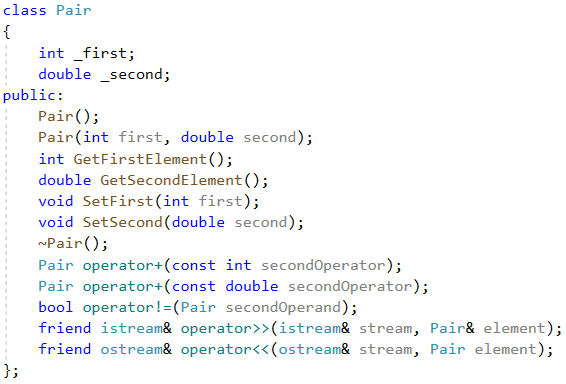
**1.5.** Организовать перегрузку оператора + класса Pair для сложения чисел типа int.

**1.6.** Организовать перегрузку оператора + класса Pair для сложения чисел типа double.

**1.7.** Организовать перегрузку оператора != класса Pair для сравнения экземпляров класса Pair.

**2.** В ходе работы были использованы следующие типы данных:

**2.1.** Организовать класс Pair с полями \_first и \_second для хранения числа в целочисленном типе и дробном типе данных.



**2.2.** Переменные типа char vvod необходима для использования меню для выбора пользователем действий.

char vvod = '5';

3

**2.2.** Массив типа char – path, необходим для хранения называния файла, в котором будут храниться пары чисел.

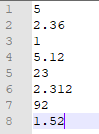
char path[] = "input.txt";

**2.3.** Вектор типа Pair – pairs, необходим для хранения всех пар чисел из файла.

vector<Pair> pairs;

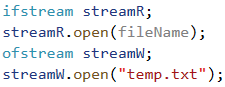
**3.** Для решения задачи данные были представлены в следующем виде:

**3.1.** Пары чисел записаны в файле input.txt.

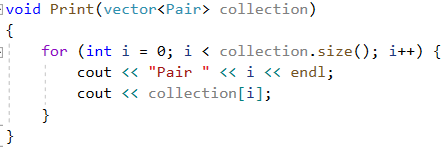


**4.** Для операций ввода и вывода использовались следующие операторы и функции:

**4.1.** Для открытия файла на чтение и запись использовать потоки ifstream и ofstream.

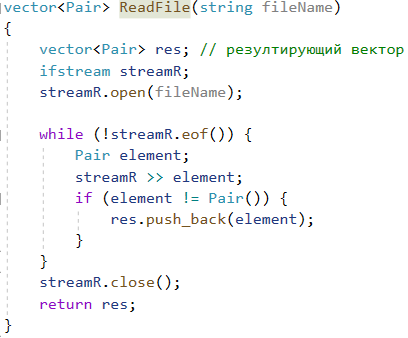


**4.2.** Для вывода пар чисел на консоль использовать функцию Print().



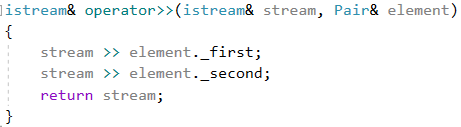
В качестве параметра функции поступает вектор Pair, который заполняется в функции ReadFile(). Открывается файл и в цикле while происходит итерирование по файлу построчно с помощью метода .eof(). Добавлена проверка на последнюю строку, чтобы не отправлять в вектор пустое значение.

4

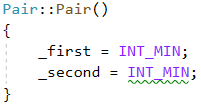


**5.** Поставленные задачи будут решены следующими действиями:

**5.1.** В функции DeleteSmallerElements() в качестве параметров передаётся название файла, и значения для сравнения с парами в самом файле. Таким образом, в цикле создаётся объект класса Pair, который инициализируется в значениями из файла через перегрузку оператора >> .

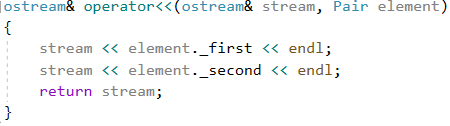


Далее происходит проверка условия удаления элемента: если итерирование дошло до конца файла, то, чтобы не заносить в поток пустую строку объект element сравнивается с объектом класса Pair, который инициализирован значениями INT\_MIN в конструкторе класса, что условно принимается как пустая строка.

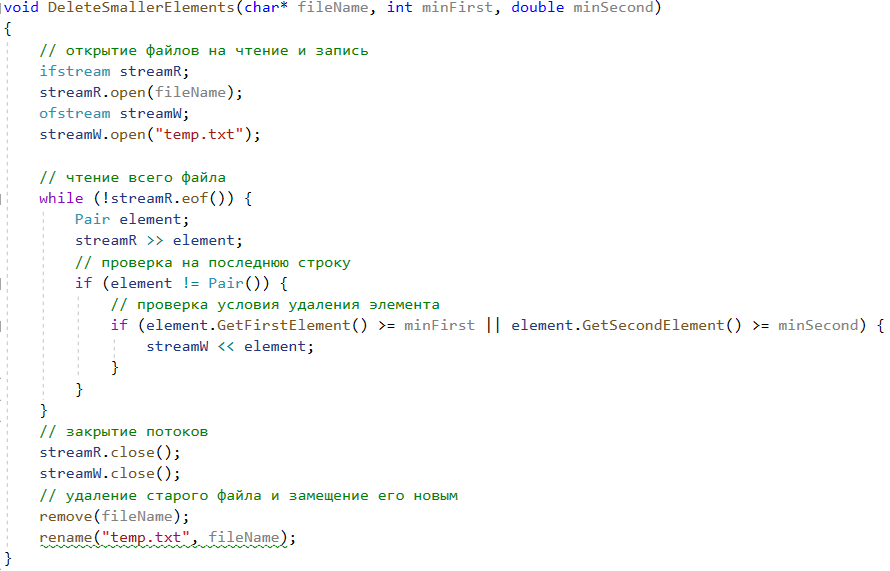


Если объект element имеет значения, не равные INT\_MIN, то этот объект подходит и можно дальше проверять его: если значения пары больше, чем переданные как параметры в функцию, то они передаются в поток через перегрузку оператора << и отправляются в запасной файл.

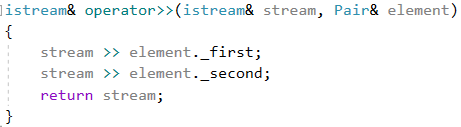
5



После цикла необходимо закрыть файлы, удалить файл изначальный, а запасному файлу дать имя изначального.



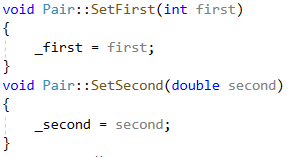
**5.2.** В функции IncreaseElement() в качестве параметров передаётся название файла, пара чисел f, s, которые пользователь хочет увеличить, а четвёртым параметром отправляется значение, на которое необходимо увеличить выбранную пользователем пару чисел. Таким образом, в цикле while() создаётся объект класса Pair, который инициализируется в значениями из файла через перегрузку оператора >> .

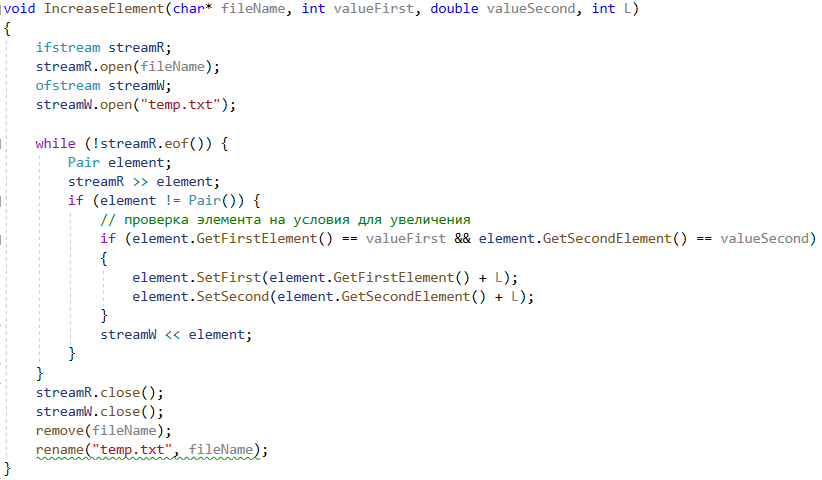


В первом ветвлении происходит проверка на пустую строку путём сравнения потока c пустым (инициализированным INT\_MIN) объектом. Во втором ветвлении происходит сравнение параметров, переданных в функцию valueFirst и valueSecond c данной парой из объекта element.

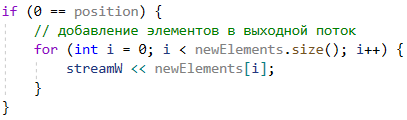
6

В случае равенства введённых элементов с парой в объекте element с помощью метода SetFirst() и SetSecond() устанавливаются значения увеличенные на L.



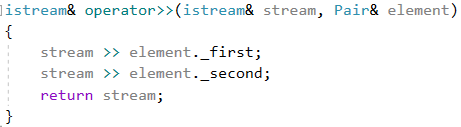


**5.3.** В функции AddElements() в качестве параметров передаётся название файла, вектор типа Pair newElements, в котором передаются две пары, которые необходимо добавить, а также передаются в переменную position место, на которое нужно вставить элемент в файл. В ветвлении прописан случай, если пользователь хочет вставить пары в начало файла в цикле for().

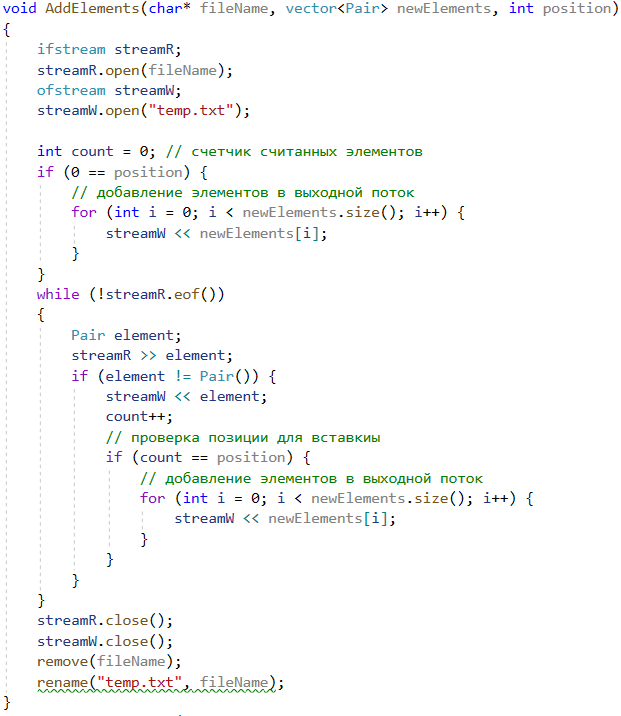


В цикле while() создаётся объект класса Pair - element, который инициализируется в значениями из файла через перегрузку оператора >> .

7

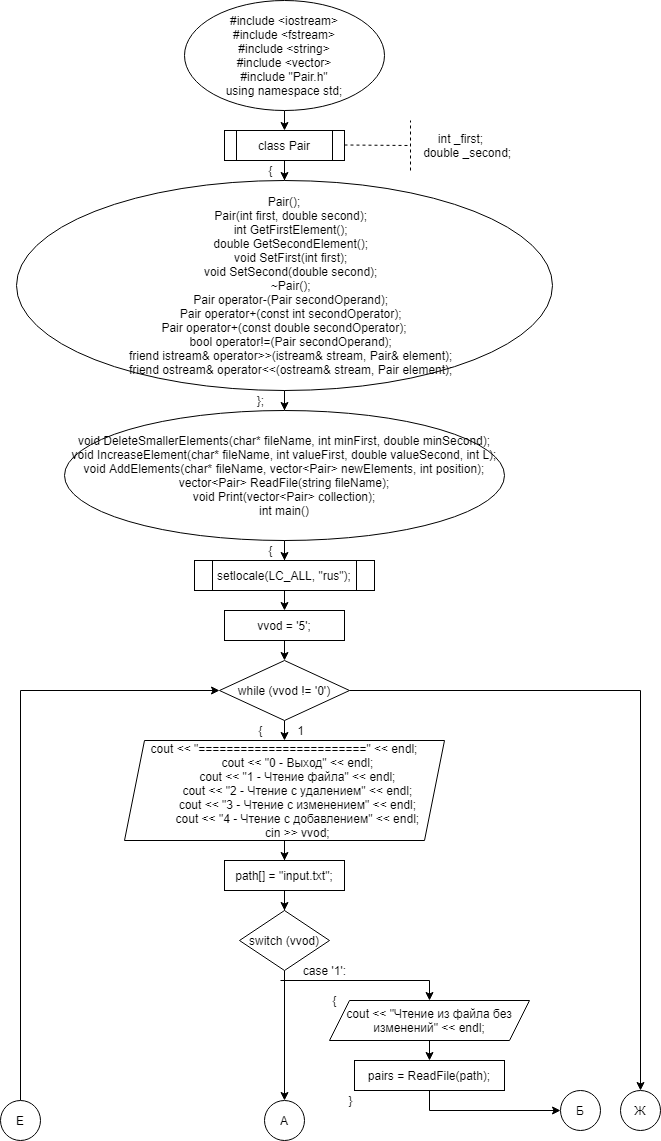


В первом ветвлении происходит проверка на пустую строку путём сравнения потока c пустым (инициализированным INT\_MIN) объектом. Далее пары из объекта element записываются в файл temp через объект streamW, далее увеличивается на 1 счётчик count, который необходим для следующего ветвления: если значение счётчика count совпадает с параметром position, то производится вставка пар в файл.

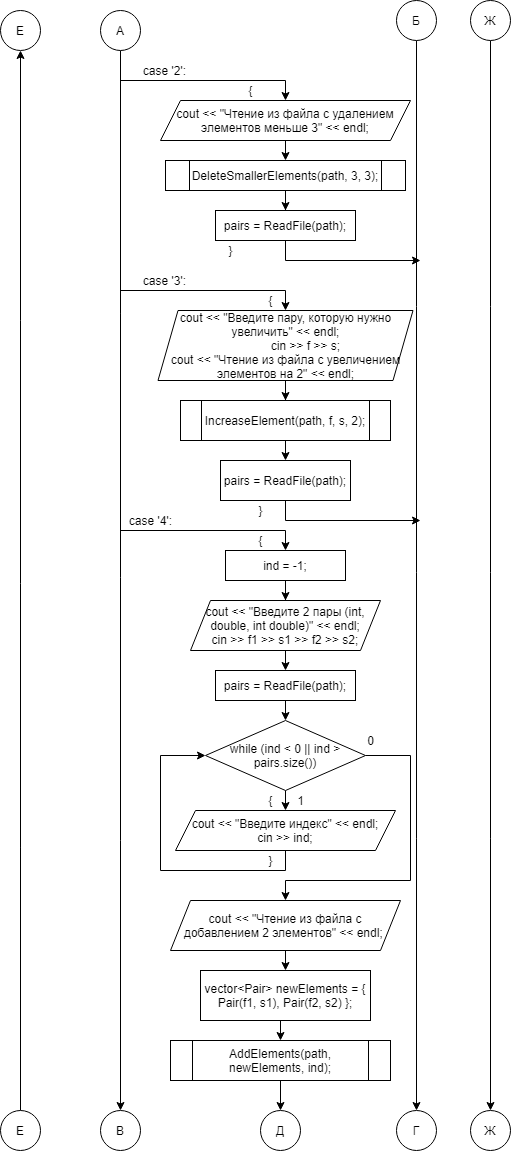


8

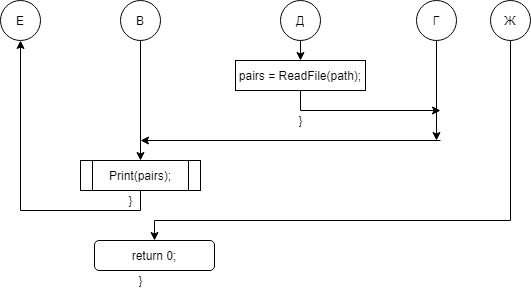
**Блок-схема**



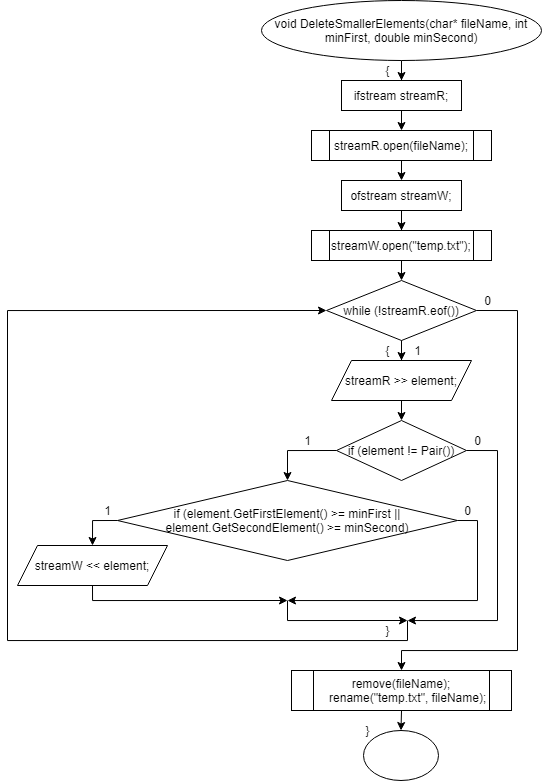
9



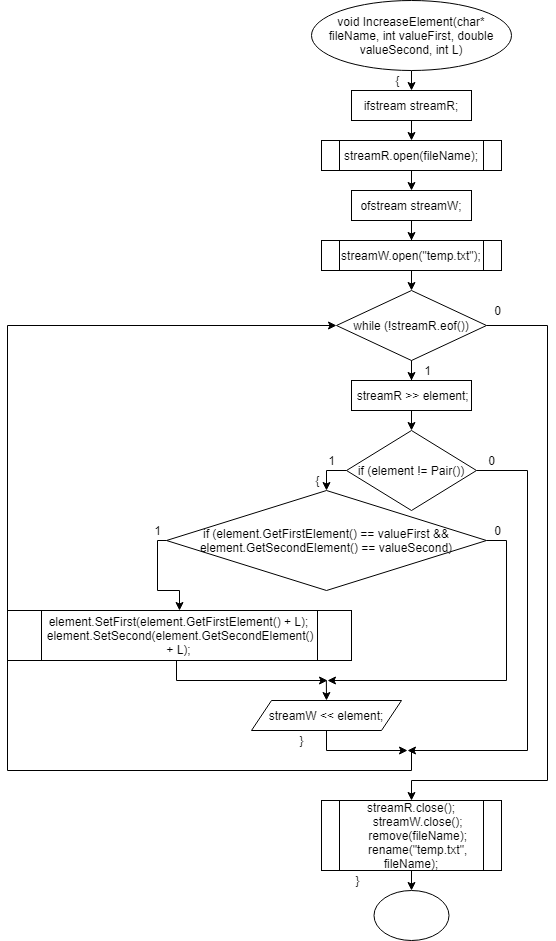
10



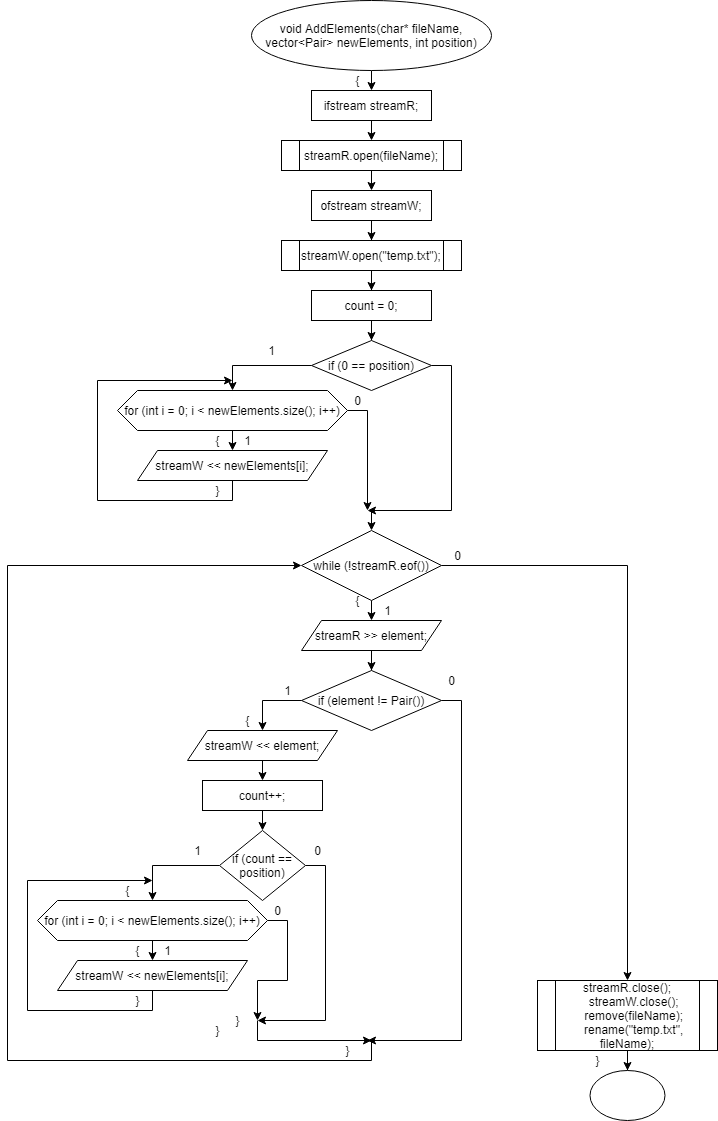
11



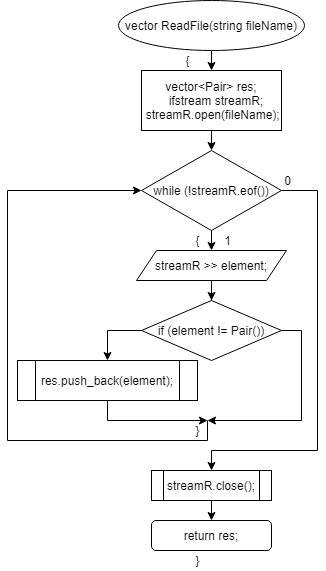
12



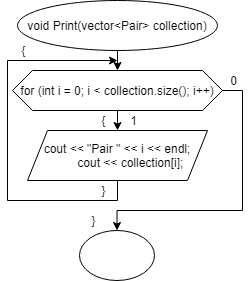
13



14



15



16

**Код**

#include <iostream>

#include <fstream>

#include <string>

#include <vector>

#include "Pair.h"

using namespace std;

void DeleteSmallerElements(char\* fileName, int minFirst, double minSecond)

{

// открытие файлов на чтение и запись

ifstream streamR;

streamR.open(fileName);

ofstream streamW;

streamW.open("temp.txt");

// чтение всего файла

while (!streamR.eof())

{

Pair element;

streamR >> element;

// проверка на последнюю строку

if (element != Pair())

{

// проверка условия удаления элемента

if (element.GetFirstElement() >= minFirst || element.GetSecondElement() >= minSecond)

{

streamW << element;

}

}

}

// закрытие потоков

streamR.close();

streamW.close();

// удаление старого файла и замещение его новым

remove(fileName);

rename("temp.txt", fileName);

}

void IncreaseElement(char\* fileName, int valueFirst, double valueSecond, int L)

{

ifstream streamR;

streamR.open(fileName);

ofstream streamW;

streamW.open("temp.txt");

while (!streamR.eof())

{

Pair element;

streamR >> element;

if (element != Pair())

{

// проверка элемента на условия для увеличения

if (element.GetFirstElement() == valueFirst && element.GetSecondElement() == valueSecond)

{

element.SetFirst(element.GetFirstElement() + L);

element.SetSecond(element.GetSecondElement() + L);

}

streamW << element;

17

}

}

streamR.close();

streamW.close();

remove(fileName);

rename("temp.txt", fileName);

}

void AddElements(char\* fileName, vector<Pair> newElements, int position)

{

ifstream streamR;

streamR.open(fileName);

ofstream streamW;

streamW.open("temp.txt");

int count = 0; // счетчик считанных элементов

if (0 == position)

{

// добавление элементов в выходной поток

for (int i = 0; i < newElements.size(); i++)

{

streamW << newElements[i];

}

}

while (!streamR.eof())

{

Pair element;

streamR >> element;

if (element != Pair())

{

streamW << element;

count++;

// проверка позиции для вставкиы

if (count == position)

{

// добавление элементов в выходной поток

for (int i = 0; i < newElements.size(); i++)

{

streamW << newElements[i];

}

}

}

}

streamR.close();

streamW.close();

remove(fileName);

rename("temp.txt", fileName);

}

vector<Pair> ReadFile(string fileName)

{

vector<Pair> res; // резултирующий вектор

ifstream streamR;

streamR.open(fileName);

while (!streamR.eof())

{

Pair element;

streamR >> element;

if (element != Pair())

18

{

res.push\_back(element);

}

}

streamR.close();

return res;

}

void Print(vector<Pair> collection)

{

for (int i = 0; i < collection.size(); i++)

{

cout << "Pair " << i << endl;

cout << collection[i];

}

}

int main()

{

setlocale(LC\_ALL, "rus");

char vvod = '5';

while (vvod != '0')

{

cout << "========================" << endl;

cout << "0 - Выход" << endl;

cout << "1 - Чтение файла" << endl;

cout << "2 - Чтение с удалением" << endl;

cout << "3 - Чтение с изменением" << endl;

cout << "4 - Чтение с добавлением" << endl;

cin >> vvod;

char path[] = "input.txt";

vector<Pair> pairs;

switch (vvod)

{

case '1':

{

// ===reading===

cout << "Чтение из файла без изменений" << endl;

pairs = ReadFile(path);

}; break;

case '2':

{

// ===deleting===

cout << "Чтение из файла с удалением элементов меньше 3" << endl;

DeleteSmallerElements(path, 3, 3);

pairs = ReadFile(path);

}; break;

case '3':

{

// ===changing===

int f;

double s;

cout << "Введите пару, которую нужно увеличить" << endl;

cin >> f >> s;

cout << "Чтение из файла с увеличением элементов на 2" << endl;

IncreaseElement(path, f, s, 2);

pairs = ReadFile(path);

19

}; break;

case '4':

{

// ===appending===

int f1, f2, ind = -1;

double s1, s2;

cout << "Введите 2 пары (int, double, int double)" << endl;

cin >> f1 >> s1 >> f2 >> s2;

pairs = ReadFile(path);

while (ind < 0 || ind > pairs.size())

{

cout << "Введите индекс" << endl;

cin >> ind;

}

cout << "Чтение из файла с добавлением 2 элементов" << endl;

vector<Pair> newElements = { Pair(f1, s1), Pair(f2, s2) };

AddElements(path, newElements, ind);

pairs = ReadFile(path);

}; break;

}

Print(pairs);

}

}

20

#pragma once

using namespace std;

class Pair

{

private:

int \_first;

double \_second;

public:

Pair();

Pair(int first, double second);

int GetFirstElement();

double GetSecondElement();

void SetFirst(int first);

void SetSecond(double second);

~Pair();

Pair operator-(Pair secondOperand);

Pair operator+(const int secondOperator);

Pair operator+(const double secondOperator);

bool operator!=(Pair secondOperand);

friend istream& operator>>(istream& stream, Pair& element);

friend ostream& operator<<(ostream& stream, Pair element);

};

21

#include <iostream>

#include "Pair.h"

using namespace std;

Pair::Pair()

{

\_first = INT\_MIN;

\_second = INT\_MIN;

}

Pair::Pair(int first, double second)

{

\_first = first;

\_second = second;

}

int Pair::GetFirstElement()

{

return \_first;

}

double Pair::GetSecondElement()

{

return \_second;

}

void Pair::SetFirst(int first)

{

\_first = first;

}

void Pair::SetSecond(double second)

{

\_second = second;

}

Pair::~Pair()

{

}

Pair Pair::operator-(Pair secondOperand)

{

Pair res;

res.\_first = this->\_first - secondOperand.\_first;

res.\_second = this->\_second - secondOperand.\_second;

return res;

}

Pair Pair::operator+(const int secondOperator)

{

Pair res;

res.\_first = this->\_first + secondOperator;

res.\_second = this->\_second;

return res;

}

Pair Pair::operator+(const double secondOperator)

{

Pair res;

res.\_first = this->\_first;

res.\_second = this->\_second + secondOperator;

return res;

}

22

bool Pair::operator!=(Pair secondOperand)

{

return \_first != secondOperand.\_first && \_second != secondOperand.\_second;

}

istream& operator>>(istream& stream, Pair& element)

{

stream >> element.\_first;

stream >> element.\_second;

return stream;

}

ostream& operator<<(ostream& stream, Pair element)

{

stream << element.\_first << endl;

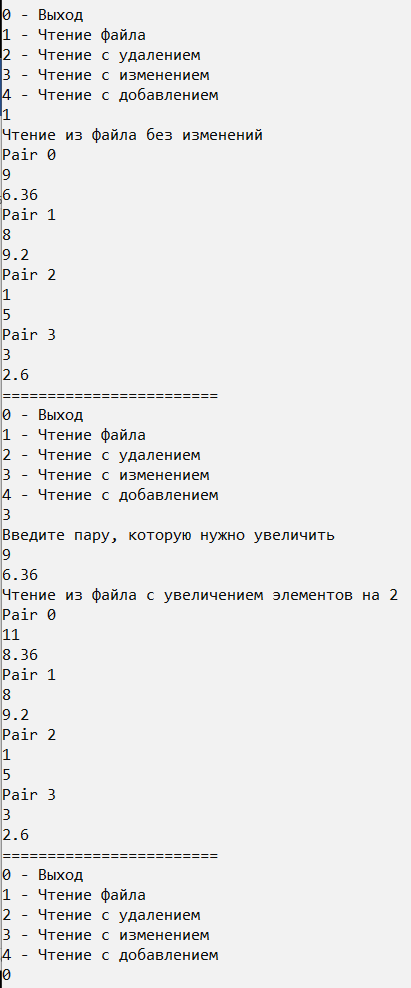
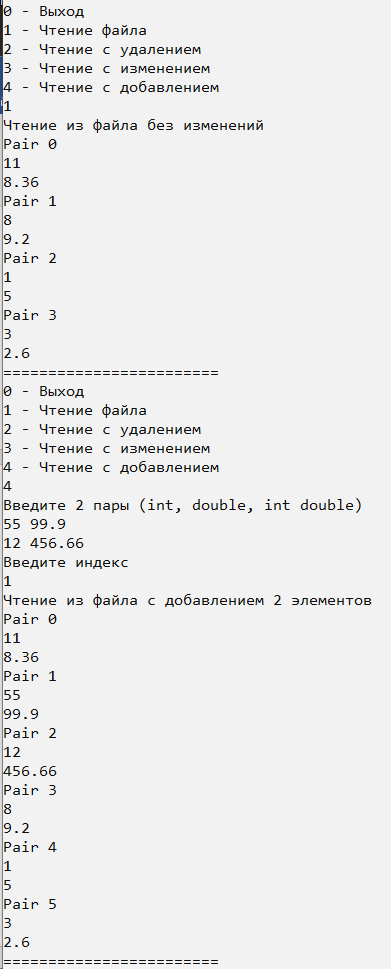
stream << element.\_second << endl;

return stream;

}

23

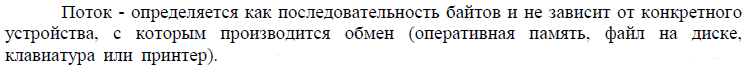
**Скриншоты**

24

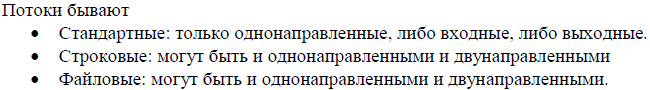
**Контрольные вопросы**



1. 



2.





3. 



4. 



5. 

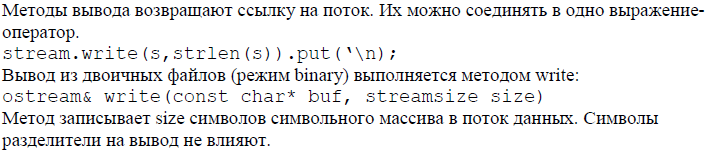


6. 



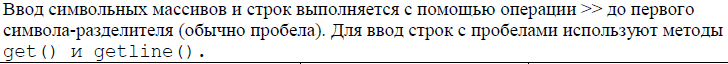
7. operator >>

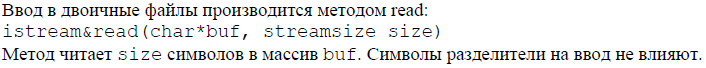


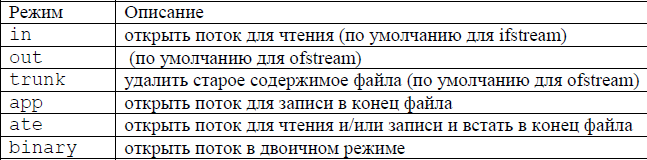
8. 

25



9.



10. 



11. 

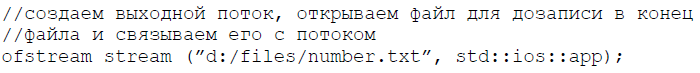


12. 



13. fstream file("f.txt", ios\_base::in | ios\_base::out)



14. 



15. 

26



16. 



17. 

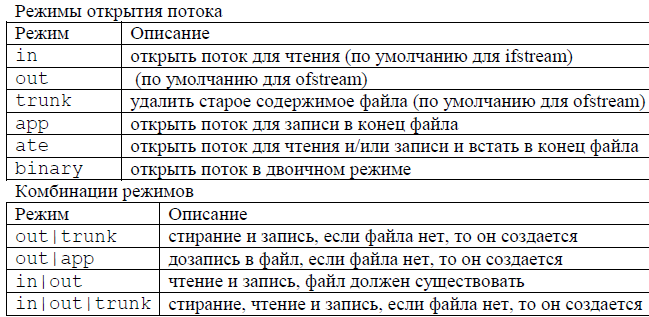


18. 



19. 



20. 



21. Перегрузка оператора в классе.



22. 



23. Перенесение в файл элементов, удовлетворяющих условию.

27



24. Запись в файл из массива элементов между записью элементов из входного файла.



25. Перезаписывание всех элементов в выходной файл с исправлением тех, что удовлетворяют условию.

28