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 Чувашев Максим

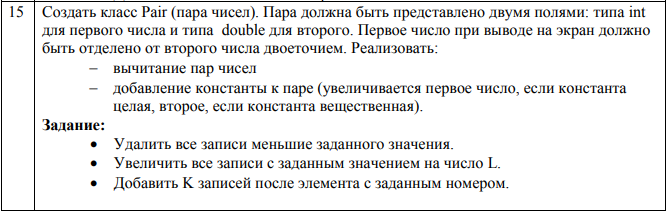
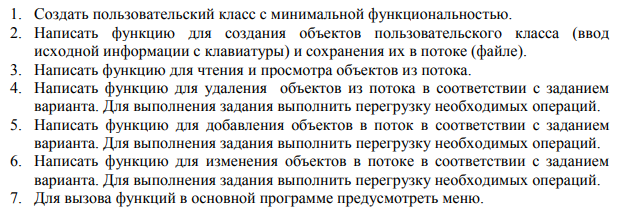
Проверила:

Доцент кафедры ИТАС

Полякова О.А.

Пермь, 2021

**Постановка задачи**



**Анализ задачи**

1. Для решения задачи необходимо:
   1. Организовать класс Pair с полями first и second для хранения числа в целочисленном типе и дробном типе данных.
   2. Организовать необходимые методы для ввода данных в поля first и second: гетторы, сетторы, конструкторы, деструкторы.
   3. Организовать перегрузку оператора >> дружественной классу Pair.
   4. Организовать перегрузку оператора << дружественной классу Pair.
   5. Организовать перегрузку оператора + класса Pair для сложения чисел типа int.
   6. Организовать перегрузку оператора + класса Pair для сложения чисел типа double.
   7. Организовать перегрузку оператора != класса Pair для сравнения экземпляров класса Pair.
2. В ходе работы были использованы следующие типы данных:
   1. Организовать класс Pair с полями first и second для хранения числа в целочисленном типе и дробном типе данных.

class Pair

{

private:

int first;

double second;

int index;

public:

Pair();

Pair(int first, double second);

int GetFirst();

double GetSecond();

int GetIndex();

void SetFirst(int first);

void SetSecond(double second);

void SetIndex(int index);

~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);

};

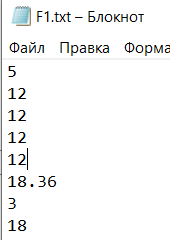
* 1. Переменные типа char tmp необходима для использования меню для выбора пользователем действий:

char tmp = '5';

* 1. Вектор типа Pair, необходим для хранения всех пар чисел из файла.

vector<Pair> res; // резултирующий вектор

1. Для решения задачи данные были представлены в следующем виде:
   1. Пары чисел записаны в файле F1.txt.



1. Для операций ввода и вывода использовались следующие операторы и функции:
   1. Для открытия файла на чтение и запись использовать потоки ifstream и ofstream.

ifstream streamR; // создаем файловый поток для чтения

streamR.open(fileName); // открываем файловый поток для чтения

ofstream streamW; // создаем файловый пооток для записи

1. Поставленные задачи будут решены следующими действиями:
   1. Функция удаления элементов из файла void DeleteElements(char\* fileName, double min)

void DeleteElements(char\* fileName, double min)

{

ifstream streamR; // создание файлового потока из файла

streamR.open(fileName); // открываем файл для чтения

ofstream streamW; // создание файлового потока в файл

streamW.open("temp.txt"); // открываем файл для записи

while (!streamR.eof()) // пока не достигнем конца файла

{

Pair element; // создаем объект класса который содержит пары элементов

streamR >> element; // с помощью перегруженного оператора вывода записываем в объект класса пары чисел

if (element != Pair()) // проверка на последнюю строку

{

// проверка условия удаления элемента

if (element.GetFirst() >= min || element.GetSecond() >= min) // если элемент больше заданного то он остается

{

streamW << element; // записываем в файл те элементы которые удовлетворяют условиям

}

}

}

streamR.close(); // закрывается файловый поток открытый для чтения

streamW.close(); // закрывается файловый поток открытый для записи

// удаление старого файла и замещение его новым

remove(fileName); // удаление старого файла

rename("temp.txt", fileName);

}

* 1. Функция добавления константы ко всем парам из файла

void IncreaseElement(char\* fileName, int valueFirst, double valueSecond, int L) // функция которая увеличивает поля элемента класса

{

ifstream streamR; // создаем файловый поток для чтения

streamR.open(fileName); // открываем файловый поток для чтения

ofstream streamW; // создаем файловый пооток для записи

streamW.open("temp.txt"); // открываем файловый поток для записи

while (!streamR.eof()) // пока не файл не кончится

{

Pair element; // создаем элемент нашего класса

streamR >> element; // выводим из файлового потока и записываем в поля нашего класса

if (element != Pair()) // проверка на последнюю строку файла

{

// проверка элемента на условия для увеличения

if (element.GetFirst() == valueFirst && element.GetSecond() == valueSecond)

{

element.SetFirst(element.GetFirst() + L); // меняем значение поля с помощью сеттера, увеличиваем элементы на заданную константу

element.SetSecond(element.GetSecond() + L); // меняем значение поля с помощью сеттара, увеличиваем элемента на заданную константу

}

streamW << element; // записываем в файл

}

}

streamR.close(); // закрываем файловый поток для чтения

streamW.close(); // закрываем файлоый поток для записи

remove(fileName); // удаление файла

rename("temp.txt", fileName);

}

* 1. Функция добавления пары в файл после определенной позиции

void AddElements(char\* fileName, vector<Pair> newElements, int ind) // добавление элементов в файл

{

ifstream streamR; // создаем поток для чтения

streamR.open(fileName); // открываем поток для чтения

ofstream streamW; // создаем поток для записи

streamW.open("temp.txt"); // открываем поток для записи

int count = 0; // счетчик

if (ind == 0)

{

// добавление элементов в выходной поток

for (int i = 0; i < newElements.size(); i++) // проходимся по всему вектору

{

streamW << newElements[i]; // записываем с помощью перегруженного оператора вывода новые элементы в файл

newElements[i].SetIndex(i);

}

}

while (!streamR.eof()) // пока не достигли конца файла

{

Pair element; // создаем элемент класса Pair

streamR >> element; // записываем с помощью потока для чтения из файла и записываем в наш новый элемент

if (element != Pair()) // проверяем на последнюю строку

{

streamW << element; // записываем новый элемент в файл с помощью перегруженного оператора вывода

count++; // увеличиваем счетчик

if (ind == count) // если текущая позиция равна требуемой позиции то

{

for (int i = 0; i < newElements.size(); i++) // проходимся по всему размеру вектора

{

streamW << newElements[i]; // добавляем с помощью перегруженного оператора вывода новый элемент в файл

}

}

}

}

streamR.close(); // закрываем файловый поток открытый для чтения

streamW.close(); // закрываем файловый поток открытый для записи

remove(fileName); // удаляем файл

rename("temp.txt", fileName);

}

* 1. Функция чтения данных из файла

vector<Pair> ReadFile(string fileName) // чтение пар чисел из файла

{

vector<Pair> res; // резултирующий вектор

ifstream streamR; // поток который считывает данные из файла

streamR.open(fileName); // открываем файл

int count = 0; // счетчик считанных элементов

while (!streamR.eof()) // проходимя по всему файлу пока не достигнем конца файла

{

Pair element; // объект класса Pair

streamR >> element; // заполняем поля объекта класса с помощью оператора вывода

if (element != Pair()) // если пара чисел существует, то

{

element.SetIndex(count);

res.push\_back(element); // добавляем объект класса в конец вектора

}

}

streamR.close(); // закрываем файловый поток

return res; // возвращаем указатель на вектор

}

**Блок-схема**















**Код**

#include <iostream>

#include <fstream>

#include <string>

#include <vector>

#include "Pair.h"

using namespace std;

void DeleteElements(char\* fileName, double min)

{

ifstream streamR; // создание файлового потока из файла

streamR.open(fileName); // открываем файл для чтения

ofstream streamW; // создание файлового потока в файл

streamW.open("temp.txt"); // открываем файл для записи

while (!streamR.eof()) // пока не достигнем конца файла

{

Pair element; // создаем объект класса который содержит пары элементов

streamR >> element; // с помощью перегруженного оператора вывода записываем в объект класса пары чисел

if (element != Pair()) // проверка на последнюю строку

{

// проверка условия удаления элемента

if (element.GetFirst() >= min || element.GetSecond() >= min) // если элемент больше заданного то он остается

{

streamW << element; // записываем в файл те элементы которые удовлетворяют условиям

}

}

}

streamR.close(); // закрывается файловый поток открытый для чтения

streamW.close(); // закрывается файловый поток открытый для записи

// удаление старого файла и замещение его новым

remove(fileName); // удаление старого файла

rename("temp.txt", fileName);

}

void IncreaseElement(char\* fileName, int valueFirst, double valueSecond, int L) // функция которая увеличивает поля элемента класса

{

ifstream streamR; // создаем файловый поток для чтения

streamR.open(fileName); // открываем файловый поток для чтения

ofstream streamW; // создаем файловый пооток для записи

streamW.open("temp.txt"); // открываем файловый поток для записи

while (!streamR.eof()) // пока не файл не кончится

{

Pair element; // создаем элемент нашего класса

streamR >> element; // выводим из файлового потока и записываем в поля нашего класса

if (element != Pair()) // проверка на последнюю строку файла

{

// проверка элемента на условия для увеличения

if (element.GetFirst() == valueFirst && element.GetSecond() == valueSecond)

{

element.SetFirst(element.GetFirst() + L); // меняем значение поля с помощью сеттера, увеличиваем элементы на заданную константу

element.SetSecond(element.GetSecond() + L); // меняем значение поля с помощью сеттара, увеличиваем элемента на заданную константу

}

streamW << element; // записываем в файл

}

}

streamR.close(); // закрываем файловый поток для чтения

streamW.close(); // закрываем файлоый поток для записи

remove(fileName); // удаление файла

rename("temp.txt", fileName);

}

void AddElements(char\* fileName, vector<Pair> newElements, int ind) // добавление элементов в файл

{

ifstream streamR; // создаем поток для чтения

streamR.open(fileName); // открываем поток для чтения

ofstream streamW; // создаем поток для записи

streamW.open("temp.txt"); // открываем поток для записи

int count = 0; // счетчик

if (ind == 0)

{

// добавление элементов в выходной поток

for (int i = 0; i < newElements.size(); i++) // проходимся по всему вектору

{

streamW << newElements[i]; // записываем с помощью перегруженного оператора вывода новые элементы в файл

newElements[i].SetIndex(i);

}

}

while (!streamR.eof()) // пока не достигли конца файла

{

Pair element; // создаем элемент класса Pair

streamR >> element; // записываем с помощью потока для чтения из файла и записываем в наш новый элемент

if (element != Pair()) // проверяем на последнюю строку

{

streamW << element; // записываем новый элемент в файл с помощью перегруженного оператора вывода

count++; // увеличиваем счетчик

if (ind == count) // если текущая позиция равна требуемой позиции то

{

for (int i = 0; i < newElements.size(); i++) // проходимся по всему размеру вектора

{

streamW << newElements[i]; // добавляем с помощью перегруженного оператора вывода новый элемент в файл

}

}

}

}

streamR.close(); // закрываем файловый поток открытый для чтения

streamW.close(); // закрываем файловый поток открытый для записи

remove(fileName); // удаляем файл

rename("temp.txt", fileName);

}

vector<Pair> ReadFile(string fileName) // чтение пар чисел из файла

{

vector<Pair> res; // резултирующий вектор

ifstream streamR; // поток который считывает данные из файла

streamR.open(fileName); // открываем файл

int count = 0; // счетчик считанных элементов

while (!streamR.eof()) // проходимя по всему файлу пока не достигнем конца файла

{

Pair element; // объект класса Pair

streamR >> element; // заполняем поля объекта класса с помощью оператора вывода

if (element != Pair()) // если пара чисел существует, то

{

element.SetIndex(count);

res.push\_back(element); // добавляем объект класса в конец вектора

}

}

streamR.close(); // закрываем файловый поток

return res; // возвращаем указатель на вектор

}

void Print(vector<Pair> collection) // функция вывода вектора который хранит пары чисел в консоль

{

for (int i = 0; i < collection.size(); i++) // проходимся по всему вектору

{

cout << " [" << i << "]" << endl; // номер объекта

cout << collection[i] << endl;; // выводим с помощью перегруженного оператора вывода пару чисел в консоль

cout << "\_\_\_\_\_\_\_\_\_\_" << endl << endl;

}

}

int main()

{

setlocale(LC\_ALL, "rus");

char tmp = '5';

while (tmp != '0')

{

tmp = '5';

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

cout << "0:\tВыход" << endl;

cout << "1:\tЧтение файла" << endl;

cout << "2:\tЧтение с удалением" << endl;

cout << "3:\tЧтение с изменением" << endl;

cout << "4:\tЧтение с добавлением" << endl;

cin >> tmp;

char path[] = "F1.txt";

vector<Pair> pairs;

switch (tmp)

{

case '1':

{

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

pairs = ReadFile(path);

};

break;

case '2':

{

cout << "Введите минимально возможное значение для пар: \t";

double min;

cin >> min;

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

DeleteElements(path, min);

pairs = ReadFile(path);

};

break;

case '3':

{

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);

};

break;

case '4':

{

int first1;

int first2;

int ind = -1;

double second1, second2;

cout << "Введите 2 пары" << endl;

cout << "Введите первое число типа int:\t";

cin >> first1;

cout << "Введите 2 число типа double:\t";

cin >> second1;

cout << "Введите первое число типа int:\t";

cin >> first2;

cout << "Введите 2 число типа double:\t";

cin >> second2;

pairs = ReadFile(path);

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

{

cout << "На какое место вставить эти элементы:\t";

cin >> ind;

}

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

vector<Pair> newElements = { Pair(first1, second1), Pair(first2, second2) };

AddElements(path, newElements, ind);

pairs = ReadFile(path);

};

break;

}

Print(pairs);

}

}

#include <iostream>

#include "Pair.h"

using namespace std;

Pair::Pair()

{

first = NULL;

second = NULL;

index = NULL;

}

Pair::Pair(int first, double second)

{

this->first = first;

this->second = second;

this->index = NULL;

}

int Pair::GetFirst()

{

return first;

}

double Pair::GetSecond()

{

return second;

}

int Pair::GetIndex()

{

return index;

}

void Pair::SetFirst(int first)

{

this->first = first;

}

void Pair::SetSecond(double second)

{

this->second = second;

}

void Pair::SetIndex(int index)

{

this->index = index;

}

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;

}

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;

}

#pragma once

using namespace std;

class Pair

{

private:

int first;

double second;

int index;

public:

Pair();

Pair(int first, double second);

int GetFirst();

double GetSecond();

int GetIndex();

void SetFirst(int first);

void SetSecond(double second);

void SetIndex(int index);

~Pair();

Pair operator-(Pair secondOperand);

Pair operator+(const int secondOperator);

Pair operator+(const double secondOperator);

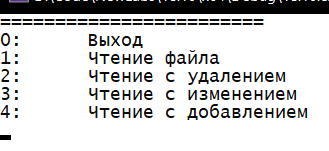
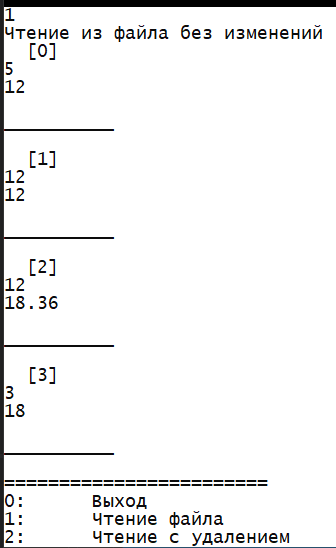
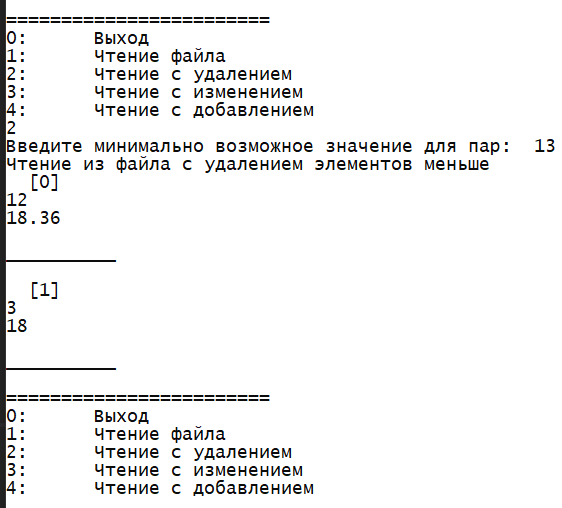
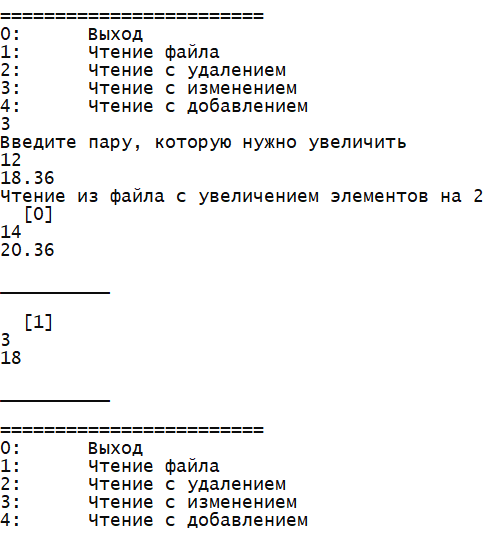
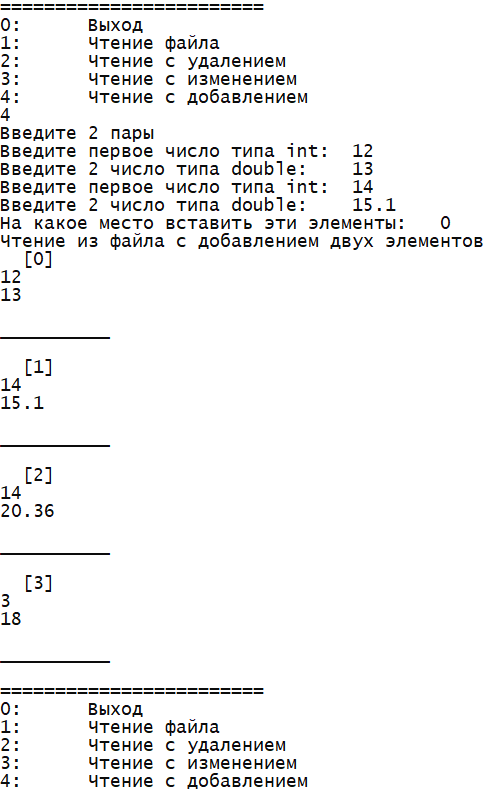
bool operator!=(Pair secondOperand);

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

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

};

**Скриншоты**

1. 
2. 
3. 
4. 
5. 

**Ответы на вопросы**

