Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

Пермский национальный исследовательский политехнический университет

Электротехнический факультет

Кафедра информационных технологий и автоматизированных систем

**ОТЧЕТ**

**о работе по информатике**

**Тема:** Лабораторная работа № 10 по ООП

Семестр: 2

Выполнил студент ИВТ-22-2б:

Солодов Александр Андреевич

Проверил доцент кафедры ИТАС:

Полякова Ольга Андреевна

Пермь 2023

**Постановка задачи**

1. Создать пользовательский класс с минимальной функциональностью.

2. Написать функцию для создания объектов пользовательского класса (ввод исходной информации с клавиатуры) и сохранения их в потоке (файле).

3. Написать функцию для чтения и просмотра объектов из потока.

4. Написать функцию для удаления объектов из потока в соответствии с заданием варианта. Для выполнения задания выполнить перегрузку необходимых операций.

5. Написать функцию для добавления объектов в поток в соответствии с заданием варианта. Для выполнения задания выполнить перегрузку необходимых операций.

6. Написать функцию для изменения объектов в потоке в соответствии с заданием варианта. Для выполнения задания выполнить перегрузку необходимых операций.

7. Для вызова функций в основной программе предусмотреть меню.

*Вариант 12*

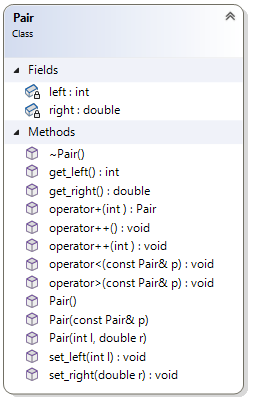
Создать класс Pair (пара чисел). Пара должна быть представлено двумя полями: типа int для первого числа и типа double для второго. Первое число при выводе на экран должно быть отделено от второго числа двоеточием. Реализовать:

* операции сравнения (<, >).
* операция ++, которая работает следующим образом: если форма операции префиксная, то увеличивается первое число, если форма операции постфиксная, то увеличивается второе число.

**Задание:**

* Удалить все записи большие заданного значения.
* Увеличить все записи с заданным значением на число L.
* Добавить K записей после записи с номером N.

**UML-диаграмма**



**Программное решение**

Pair.h

#pragma once

#include <iostream>

#include <fstream>

using namespace std;

class Pair {

private:

int left;

double right;

public:

//конструкторы

Pair() {

left = 0;

right = 0.0;

}

Pair(int l, double r) {

left = l;

right = r;

}

Pair(const Pair& p) {

left = p.left;

right = p.right;

}

~Pair() {}

//геттеры

int get\_left() {

return left;

}

void set\_left(int l) {

left = l;

}

//сеттеры

double get\_right() {

return right;

}

void set\_right(double r) {

right = r;

}

//перегрузки операторов сравнения

void operator < (const Pair& p);

void operator > (const Pair& p);

//перегрузка префиксного инкремента

void operator ++ () { ++left; }

//перегрузка постфиксного инкремента

void operator ++ (int) { right += 1.0; }

//перегрузка оператора сложения

Pair operator + (int);

//перегрузки операций ввода-вывода

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

friend ostream& operator << (ostream&, const Pair&);

//перегрузки для работы с файловым потоком

friend fstream& operator >> (fstream&, Pair&);

friend fstream& operator << (fstream&, const Pair&);

};

Pair.cpp

#include "Pair.h"

using namespace std;

//перегрузка операций сравнения

void Pair::operator < (const Pair& p) {

if ((this->left < p.left) || (this->left == p.left && this->right < p.right)) {

cout << "Первая пара чисел меньше, чем вторая пара" << endl;

}

else {

cout << "Первая пара чисел НЕ меньше, чем вторая пара" << endl;

}

}

void Pair::operator > (const Pair& p) {

if ((this->left > p.left) || (this->left == p.left && this->right > p.right)) {

cout << "Первая пара чисел больше, чем вторая пара" << endl;

}

else {

cout << "Первая пара чисел НЕ больше, чем вторая пара" << endl;

}

}

//перегрузка оператора сложения

Pair Pair::operator + (int add) {

return Pair(left + add, right + add);

}

//перегрузка операции ввода

istream& operator >> (istream& in, Pair& p) {

cout << "Введите левое число: ";

in >> p.left;

cout << "Введите правое число: ";

in >> p.right;

return in;

}

//перегрузка операции вывода

ostream& operator << (ostream& out, const Pair& p) {

return (out << p.left << ":" << p.right) << endl;

}

//операции файлового потока

fstream& operator >> (fstream& fin, Pair& p) {

fin >> p.left;

fin >> p.right;

return fin;

}

fstream& operator << (std::fstream& fout, const Pair& p) {

fout << p.left << '\n';

fout << p.right << '\n';

return fout;

}

file\_work.h

#include "Pair.h"

#include <string>

//функция для записи в файл

int make\_file(const char\* f\_name) {

fstream stream(f\_name, ios::out | ios::trunc);

if (!stream)

return -1;

int n;

Pair p;

cout << "Введите количество экземпляров: ";

cin >> n;

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

{

cin >> p;

stream << p << "\n";

}

stream.close();

return n;

}

//функция для печати файла

int print\_file(const char\* f\_name) {

fstream stream(f\_name, ios::in);

if (!stream)

return -1;

Pair temp;

int i = 0;

while (stream >> temp)

{

cout << temp << "\n";

i++;

}

stream.close();

return i;

}

//удалить все записи большие заданного значения

int deletePairs(const char\* f\_name, int value) {

fstream fromStream(f\_name, ios::in);

fstream inStream("temp", ios::out | ios::trunc);

int i = 0;

Pair p;

while (fromStream >> p) {

if (p.get\_left() <= value && !fromStream.eof()) {

inStream << p;

i++;

}

}

fromStream.close();

inStream.close();

remove(f\_name);

rename("temp", f\_name);

return i;

}

//увеличить все записи с заданным значением на число L

int increasePairs(const char\* f\_name, int currentValue, int addValue) {

fstream fromStream(f\_name, ios::in);

fstream inStream("temp", ios::out | ios::trunc);

int i = 0;

Pair p;

while (fromStream >> p) {

if (p.get\_left() == currentValue && !fromStream.eof()) {

p = p + addValue;

i++;

}

inStream << p;

}

fromStream.close();

inStream.close();

remove(f\_name);

rename("temp", f\_name);

return i;

}

//добавить K записей после записи с номером N

int add\_file(const char\* f\_name, int k, Pair newPair) {

fstream temp("temp", ios::out);

fstream stream(f\_name, ios::in);

if (!stream)

return -1;

Pair p;

int i = 0, l = 0;

while (stream >> p)

{

if (stream.eof())

break;

i++;

temp << p;

if (i == k)

{

temp << newPair;

l++;

}

}

stream.close();

temp.close();

remove(f\_name);

rename("temp", f\_name);

return l;

}

main.cpp

/\* Вариант 12.

Создать класс Pair (пара чисел). Пара должна быть представлено двумя полями:

типа int для первого числа и типа double для второго.

Первое число при выводе на экран должно быть отделено от второго числа двоеточием.

Реализовать:

- операции сравнения (<, >).

- операция ++, которая работает следующим образом:

если форма операции префиксная, то увеличивается первое число,

если форма операции постфиксная, то увеличивается второе число.

Задание:

• Удалить все записи большие заданного значения.

• Увеличить все записи с заданным значением на число L.

• Добавить K записей после записи с номером N. \*/

#include "Pair.h"

#include "file\_work.h"

int main() {

setlocale(LC\_ALL, "RU");

Pair p;

int k, c;

char file\_name[30];

do {

cout << "1.Создать файл";

cout << "\n2.Вывести содержимое файла";

cout << "\n3.Удалить записи из файла больше заданного значения";

cout << "\n4.Увеличить все записи с заданным значением на число L";

cout << "\n5.Добавить K записей после элемента с номером N";

cout << "\n0.Выход\n";

cout << "Ваш выбор: ";

cin >> c;

switch (c)

{

case 1: cout << "Введите название файла: ";

cin >> file\_name;

k = make\_file(file\_name); //вызов функции для записи в файл

if (k < 0)

cout << "Не удалось создать файл!";

break;

case 2: cout << "Введите название файла: ";

cin >> file\_name;

k = print\_file(file\_name); //функция для печати файла

if (k == 0)

cout << "Файл пустой!\n";

if (k < 0)

cout << "Не удалось открыть файл!\n";

break;

case 3: cout << "Введите название файла: ";

cin >> file\_name;

int valueDelete; //удалить все записи большие заданного значения

cout << "Введите значение, записи больше него будут удалены: ";

cin >> valueDelete;

k = deletePairs(file\_name, valueDelete);

if (k == 0)

cout << "Файл пустой!\n";

if (k < 0)

cout << "Не удалось открыть файл!\n";

break;

case 4: cout << "Введите название файла: ";

cin >> file\_name;

int valueIncrease, L;

cout << "Введите значение, записи с которым будут увеличены: ";

cin >> valueIncrease;

cout << "Введите число L: ";

cin >> L;

k = increasePairs(file\_name, valueIncrease, L); //увеличить все записи с заданным значением на число L

if (k == 0)

cout << "Файл пустой!\n";

if (k < 0)

cout << "Не удалось открыть файл!\n";

break;

case 5: cout << "Введите название файла: ";

cin >> file\_name;

int N, K;

Pair newPair;

cout << "Введите номер N: ";

cin >> N;

cout << "Введите К - количество записей для добавления: ";

cin >> K;

cout << "Новая пара чисел:" << endl;

cin >> newPair;

k = add\_file(file\_name, N, newPair); //добавить K записей после записи с номером N

if (k > 0) {

for (int i = 1; i < K; i++) {

cout << "Новая пара чисел:" << endl;

cin >> newPair;

k = add\_file(file\_name, N + 1, newPair);

N++;

}

}

if (k == 0)

cout << "Файл пустой!\n";

if (k < 0)

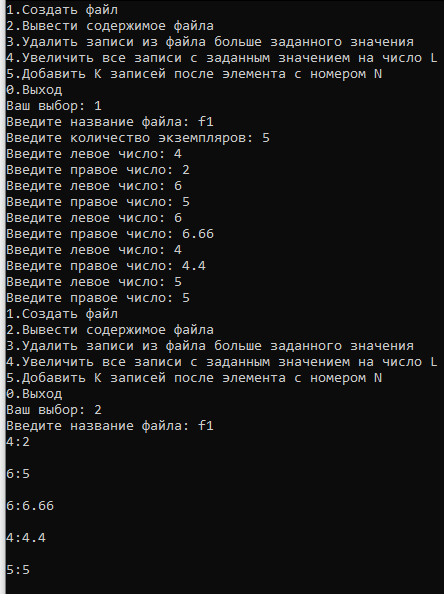
cout << "Не удалось открыть файл!\n";

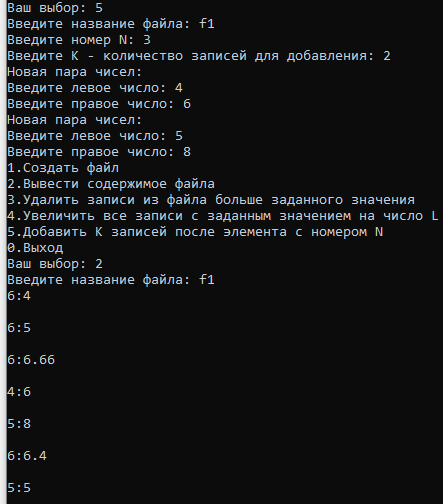
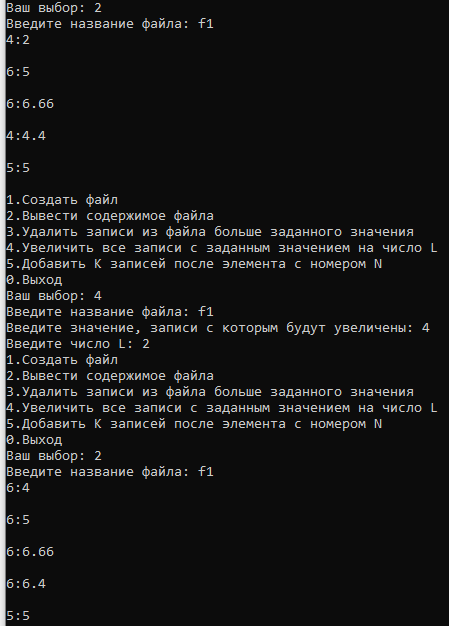
break;

}

} while (c != 0);

return 0;}

**Результат работы программы**



**Контрольные вопросы**

*1. Что такое поток?*

Поток - определяется как последовательность байтов и не зависит от конкретного устройства, с которым производится обмен (оперативная память, файл на диске, клавиатура или принтер). Обмен с потоком для увеличения скорости передачи данных производится, как правило, через специальную область оперативной памяти — буфер. Буфер накапливает байты, и фактическая передача данных выполняется после заполнения буфера. При вводе это дает возможность исправить ошибки, если данные из буфера еще не отправлены в программу.

*2. Какие типы потоков существуют?*

* Стандартные: только однонаправленные, либо входные, либо выходные.
  + Привязаны к стандартным устройствам: клавиатуре и экрану.
* Строковые: могут быть и однонаправленными и двунаправленными.
* Файловые: могут быть и однонаправленными и двунаправленными.
  + Переменная файлового потока связывается со стандартным файлом на диске.

*3. Какую библиотеку надо подключить при использовании стандартных потоков?*

Для использования стандартных потоков надо задать в программе директиву:

#include <iostream>

В заголовочном файле iostream содержатся описания классов ввода/вывода и четыре стандартных системных объекта:

* cin – объект класса istream, по умолчанию связан с клавиатурой;
* cout - объект класса ostream, по умолчанию связан с экраном;
* clog - объект класса ostream, соответствующий стандартному выводу для ошибок, по умолчанию связан с экраном;
* cerr- объект класса ostream, соответствующий стандартному выводу для ошибок, по умолчанию связан с экраном.

*4. Какую библиотеку надо подключить при использовании файловых потоков?*

Для использования файловых потоков надо задать в программе директиву: #include <fstream>

После этого в программе можно объявлять объекты – файловые потоки трех типов:

* входной ifstream;
* выходной ofstream;
* двунаправленный fstream.

*5. Какую библиотеку надо подключить при использовании строковых потоков?*

Для использования строковых потоков надо задать в программе директиву #include <sstream>

После этого в программе можно объявлять объекты – строковые потоки трех типов:

* входной istringstream;
* выходной ostringstream;
* двунаправленный stringstream.

*6. Какая операция используется при выводе в форматированный поток?*

Для форматируемых потоков вывод осуществляется перегруженной операцией сдвига влево operator<<

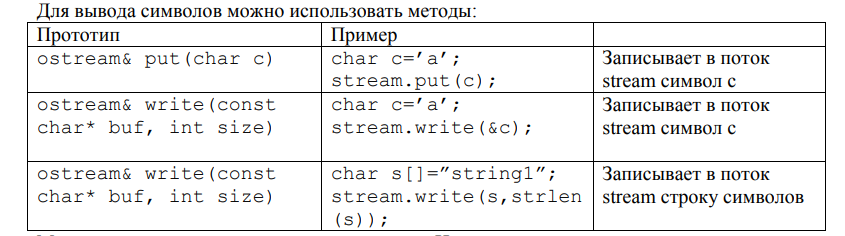
stream << 3.4 << ’\n’; //Вывод константы

*7. Какая операция используется при вводе из форматированных потоков?*

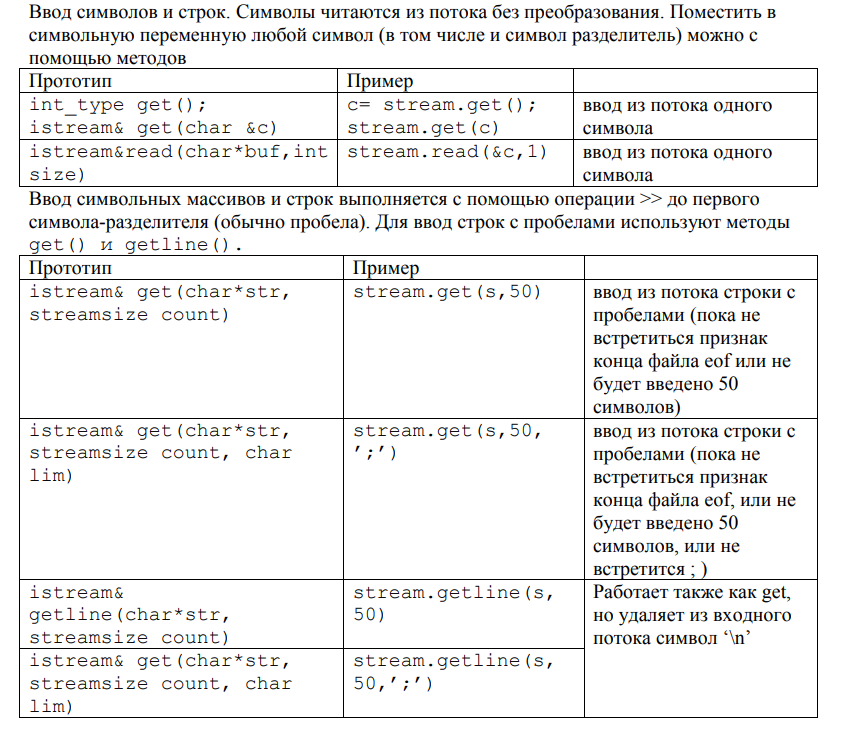
Для форматируемых потоков ввод осуществляется перегруженной операцией сдвига вправо operator>>

stream >> a; //Вывод переменной a

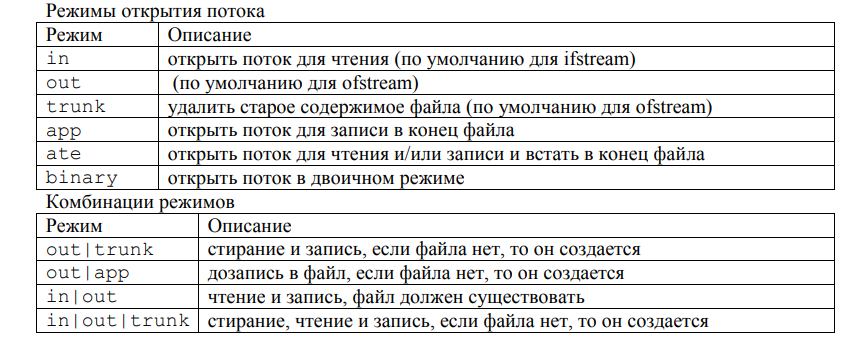
*8. Какие методы используются при выводе в форматированный поток?*



*9. Какие методы используется при вводе из форматированного потока?*



*10. Какие режимы для открытия файловых потоков существуют?*



*11. Какой режим используется для добавления записей в файл?*

Для добавления записи в конец файла используется режим app.

Также, используется режим ate для чтения и/или записи в конец файла.

ofstream stream("number.txt", std::ios::app); //создаем выходной поток, открываем файл для дозаписи в конец файла и связываем его с потоком

*12. Какой режим (комбинация режимов) используется в конструкторе ifstream file(“f.txt”)?*

Используется режим in - открыть поток для чтения (по умолчанию для ifstream).

*13. Какой режим (комбинация режимов) используется в конструкторе fstream file(“f.txt”)?*

Используется комбинация режимов in | out - чтение и запись, файл должен существовать.

*14. Какой режим (комбинация режимов) используется в конструкторе ofstream file(“f.txt”)?*

Используется режим out - открыть поток для записи в файл (по умолчанию для ofstream).

*15. Каким образом открывается поток в режиме ios::out|ios::app?*

Поток открывается для дозаписи в файл, если файла нет, то он создаётся.

*16. Каким образом открывается поток в режиме ios::out |ios::trunc?*

Поток открывается для стирания и записи, если файла нет, то он создаётся.

*17. Каким образом открывается поток в режиме ios::out |ios::in|ios::trunk?*

Поток открывается для стирания, чтения и записи, если файла нет, то он создаётся.

*18. Каким образом можно открыть файл для чтения?*

Файл может быть открыт либо явно методом open(), либо неявно – конструктором при создании потока.

Для открытия файла для чтения можно использовать различные режимы:

* in - открыть поток для чтения (по умолчанию для ifstream);
* ate - открыть поток для чтения и/или записи и встать в конец файла.

fstream file1;

file1.open("file.txt", ios::in); //Открытие для чтения с помощью метода open через fstream и in

ifstream file2("file.txt"); //Открытие через конструктор; in по умолчанию, т.к. ifstream

ifstream file3("file.txt", ios::ate); //Открытие через конструктор; режим ate через ifstream

*19. Каким образом можно открыть файл для записи?*

Файл может быть открыт либо явно методом open(), либо неявно – конструктором при создании потока.

Для открытия файла для чтения можно использовать различные режимы:

* out - открыть поток для записи в файл (по умолчанию для ofstream);
* trunk - удалить старое содержимое файла (по умолчанию для оfstream);
* app - открыть поток для записи в конец файла;
* ate - открыть поток для чтения и/или записи и встать в конец файла.

fstream file1;

file1.open("file.txt", ios::out); //Открытие для записи с помощью метода open через fstream и out

ofstream file2("file.txt"); //Открытие через конструктор; out по умолчанию, т.к. ofstream

ofstream file3;

file3.open("file.txt", ios::ate); //Открытие через метод open режимом ate через ofstream

*20. Привести примеры открытия файловых потоков в различных режимах.*

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

int main() {

ofstream file("MyFile.txt"); //Открытие файла для записи

if (!file.is\_open()) cout << "Error" << endl;

else {

file << "Hello!" << endl; //Запись данных в файл

file.close();

}

ifstream file1("input.txt"); //Открытие файла для чтения

if (!file1.is\_open()) cout << "Error" << endl;

else {

//Чтение данные из файла

string line;

while (getline(file1, line)) {

cout << line << std::endl;

}

file1.close();

}

}

*21. Привести примеры чтения объектов из потока.*

ifstream file1;

file1.open("input.txt", ios::ate); //Открытие файла для чтения с режимом ate

if (!file1.is\_open()) cout << "Error" << endl;

else {

//Чтение данные из файла

string line;

while (getline(file1, line)) {

cout << line << std::endl;

}

file1.close();

}

*22. Привести примеры записи объектов в поток.*

ofstream file("MyFile.txt", ios::app); //Открытие файла для записи с режимом app

if (!file.is\_open()) cout << "Error" << endl;

else {

file << "Hello!" << endl; //Запись данных в файл

file.close();

}

*23. Сформулировать алгоритм удаления записей из файла.*

1. Открыть файл для чтения и записи.
2. Проверить, что файл успешно открыт
3. Создать временный файл.
4. Прочитать записи из исходного файла и проверять каждую запись на соответствие критерию удаления.
5. Записать нужные записи во временный файл.
6. Закрыть исходный файл и временный файл.
7. Удалить исходный файл с помощью функции.
8. Переименовать временный файл в имя исходного файла.

*24. Сформулировать алгоритм добавления записей в файл.*

1. Открыть файл для записи с помощью функции.
2. Проверить, что файл успешно открыт с помощью функции.
3. Создать объект записи, который будет содержать данные для записи в файл.
4. Заполнить объект записи данными.
5. Записать объект записи в файл с помощью оператора или функции.
6. Проверить, что запись прошла успешно с помощью функции.
7. Повторять шаги 3-6 для каждой новой записи.
8. Закрыть файл с помощью функции.

*25. Сформулировать алгоритм изменения записей в файле.*

1. Открыть файл для чтения и записи с помощью функции std::fstream.
2. Проверить, что файл успешно открыт.
3. Считать данные из файла в структуру или массив данных.
4. Изменить необходимые данные в структуре или массиве.
5. Перезаписать измененные данные в файл.
6. Закрыть файл.