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**

Семестр: 2

На тему: «ООП. Сохранение данных в файле с использованием потоков»

Выполнил студент ИВТ-22-2б:

Казанцев Антон Васильевич

(дата, подпись)

Проверила:

Полякова Ольга Андреевна

(дата, подпись)

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 15:*

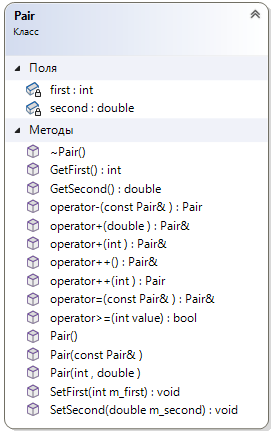
Создать класс Pair (пара чисел). Пара должна быть представлено двумя полями: типа int для первого числа и типа double для второго. Первое число при выводе на экран должно быть отделено от второго числа двоеточием. Реализовать:

* вычитание пар чисел;
* добавление константы к паре (увеличивается первое число, если константа целая, второе, если константа вещественная).

**Задание:**

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

**UML-Диаграмма**



**Программное решение**

**Pair.h**

#pragma once

#include <iostream>

using namespace std;

#include <fstream>

class Pair {

int first;

double second;

public:

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

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

Pair(int, double);

Pair();

Pair(const Pair&);

~Pair();

Pair& operator=(const Pair&);

Pair operator-(const Pair&);

Pair& operator+(int);

Pair& operator+(double);

Pair& operator++();

Pair operator++(int);

friend fstream& operator >> (fstream& fin, Pair& m\_Pair);

friend fstream& operator << (fstream& fout, const Pair& m\_Pair);

bool operator>=(int value);

void SetFirst(int m\_first);

void SetSecond(double m\_second);

int GetFirst() const;

double GetSecond() const;

};

**Pair.cpp**

#include "Pair.h"

Pair::Pair(int first, double second) {//конструктор с параметрами

this->first = first;

this->second = second;

}

istream& operator>>(istream& in, Pair& m\_pair) {//перегрузка оператора ввода

cout << "Введите данные в формате:\nцелое число вещественное число\n";

in >> m\_pair.first;

in >> m\_pair.second;

return in;

}

ostream& operator<<(ostream& out, const Pair& m\_pair) {//перегрузка оператора вывода

return (out << m\_pair.first << " : " << m\_pair.second);

}

fstream& operator>>(fstream& fin, Pair& m\_Pair) {

fin >> m\_Pair.first;

fin >> m\_Pair.second;

return fin;

}

fstream& operator<<(fstream& fout, const Pair& m\_Pair) {

fout << m\_Pair.first << ' ' << m\_Pair.second << '\n';

return fout;

}

Pair::Pair() {//конструктор без параметров

first = 0;

second = 0;

}

Pair::~Pair() {//деструктор

}

Pair::Pair(const Pair& m\_pair) {//конструктор копирования

first = m\_pair.first;

second = m\_pair.second;

}

Pair& Pair::operator=(const Pair& m\_pair) {//перегрузка оператора присваивания

if (&m\_pair != this) {

first = m\_pair.first;

second = m\_pair.second;

}

return \*this;

}

Pair Pair::operator-(const Pair& m\_pair) {//перегрузка оператора вычитания

Pair result(first - m\_pair.first, second - m\_pair.second);

return result;

}

Pair& Pair::operator+(int first) {//перегрузка оператора сложения - целый аргумент

this->first += first;

return \*this;

}

Pair& Pair::operator+(double second) {//перегрузка оператора сложения - дробный аргумент

this->second += second;

return \*this;

}

Pair& Pair::operator++() {//перегрузка оператора ++ префикс

++first;

++second;

return \*this;

}

Pair Pair::operator ++(int) {//перегрузка оператора ++ постфикс

Pair temp = \*this;

this->first++;

this->second++;

return temp;//сначала нужно вывести без изменений

}

bool Pair::operator>=(int value) {

return (this->first >= value && this->second >= value);

}

int Pair::GetFirst() const {

return first;

}

double Pair::GetSecond() const {

return second;

}

void Pair::SetFirst(int m\_first) {

first = m\_first;

}

void Pair::SetSecond(double m\_second) {

second = m\_second;

}

**File\_work.h**

#pragma once

#include "Pair.h"

#include <iostream>

using namespace std;

char f\_name[30];

int value;

void check\_k(int k) {

if (k < 0) {

cout << "Невозможно прочесть файл!\n";

}

}

int make\_file() {

cout << "Введите название файла: ";

cin >> f\_name;

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

if (!stream) {

return -1;

}

int n;

Pair m\_Pair;

cout << "Введите количество элиментов: ";

cin >> n;

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

cin >> m\_Pair;

stream << m\_Pair << '\n';

}

stream.close();

return n;

}

int print\_file() {

cout << "Введите название файла: ";

cin >> f\_name;

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

if (!stream) {

return -1;

}

Pair m\_Pair;

int i = 0;

while (stream >> m\_Pair) {

cout << m\_Pair << '\n';

i++;

}

if (i == 0) {

cout << "Файл пуст!\n";

}

stream.close();

return i;

}

int del\_file() {

cout << "Введите название файла: ";

cin >> f\_name;

cout << "Введите значение: ";

cin >> value;

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

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

if (!stream) {

return -1;

}

int i = 0;

Pair m\_Pair;

while (stream >> m\_Pair) {

if (stream.eof()) {

break;

}

i++;

if (m\_Pair >= value) {

temp << m\_Pair;

}

}

stream.close();

temp.close();

remove(f\_name);

rename("temp", f\_name);

return i;

}

int add\_file() {

int number;

cout << "Введите название файла: ";

cin >> f\_name;

cout << "Введите номер: ";

cin >> number;

cout << "Введите количество записей: ";

cin >> value;

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

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

int i = 1;

if (!stream) {

return -1;

}

Pair pair;

while (i != number && stream >> pair) {

if (stream.eof()) {

break;

}

i++;

temp << pair;

}

if (i == number) {

for (int j = 0; j < value; j++) {

Pair m\_Pair;

cin >> m\_Pair;

temp << m\_Pair;

}

}

while (stream >> pair) {

if (stream.eof()) {

break;

}

temp << pair;

}

temp.close();

stream.close();

remove(f\_name);

rename("temp", f\_name);

return value;

}

int change\_file() {

cout << "Введите название файла: ";

cin >> f\_name;

cout << "Введите значение: ";

cin >> value;

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

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

if (!stream) {

return -1;

}

Pair pair;

int i = 0;

while (stream >> pair) {

temp << pair + value;

}

temp.close();

stream.close();

remove(f\_name);

rename("temp", f\_name);

return i;

}

**LabaOOP10.cpp**

#include <iostream>

using namespace std;

#include "file\_work.h"

using namespace std;

#include <string>

#include <functional>

int main() {

system("chcp 1251 > NULL");

int choose = 6;

function<int()> f[5] = {make\_file, print\_file, del\_file, change\_file, add\_file};

do {

if (choose != 6) {

check\_k(f[choose - 1]());

}

cout << "Введите номер команды:\n1. Создать файл\n2. Печать файла\n3. Удалить все записи меньше заданного значения\n4. Увеличить все записи с заданным значением на число L\n5. Добавить K записей перед элиментом с заданным номером\n6. Выход\n";

cin >> choose;

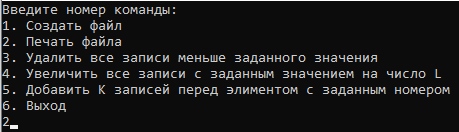
system("cls");

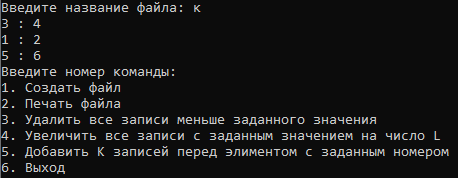
} while (choose != 6);

return 0;

}

**Пример 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<<

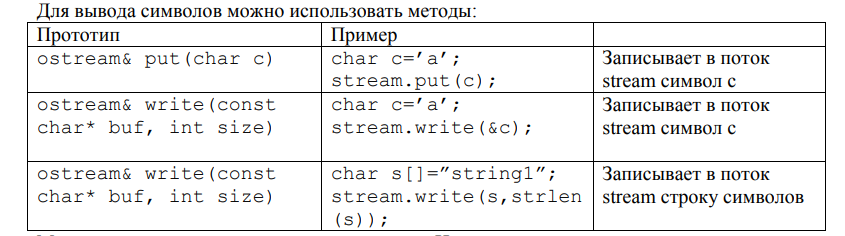
stream << 3.4 << ’\n’; //Вывод константы

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

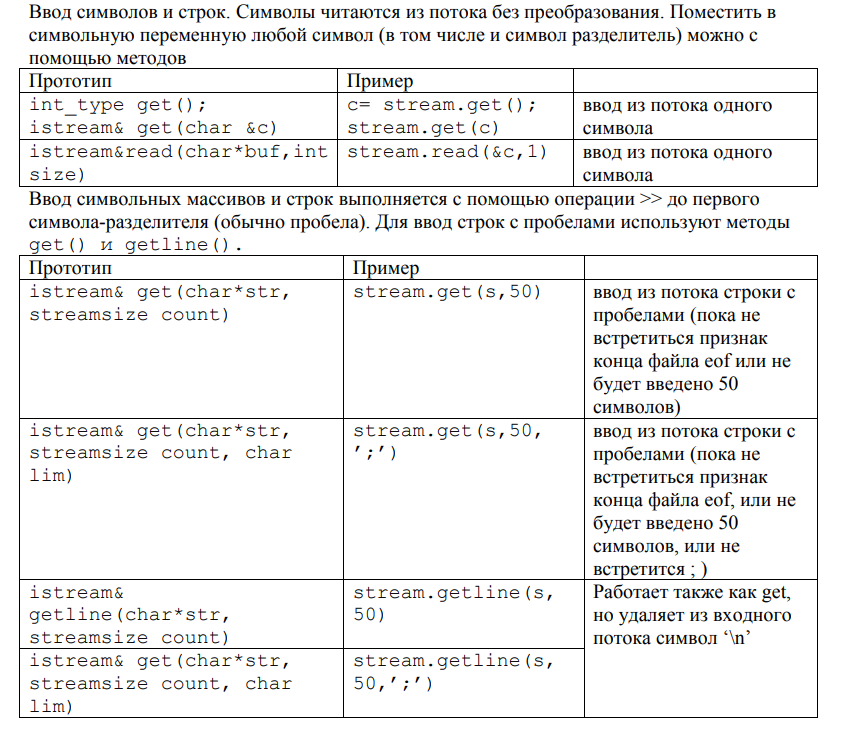
Для форматируемых потоков ввод осуществляется перегруженной операцией сдвига вправо operator>>

stream >> a; //Вывод переменной a

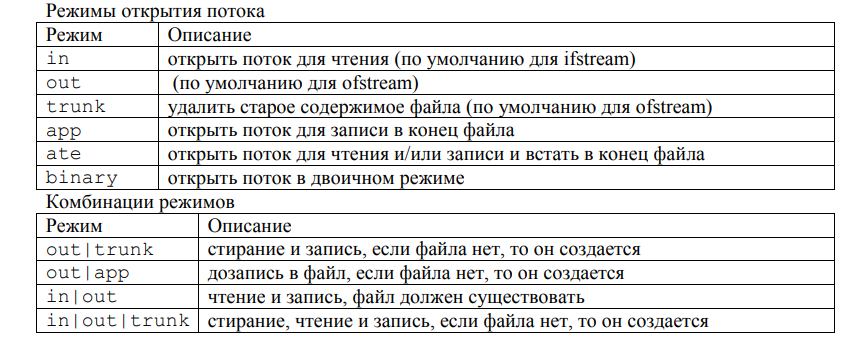
*8. Какие методы используются при выводе в форматированный поток?*



*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.