Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«**Пермский национальный исследовательский политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЕТ**

Дисциплина: «Основы алгоритмизации и программирования»

# Лабораторная работа №10

# «Сохранение данных в файле с использованием потоков»

# Семестр 2

Выполнил работу

Студент группы РИС-22-1Б

Зырянов Ростислав

Проверил

Доцент кафедры ИТАС

Полякова О.А.

Г. 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

**Ответы на контрольные вопросы**

**1. Что такое поток?**

Поток (stream) – это последовательность данных, передающихся между источником и приемником.

**2. Какие типы потоков существуют?**

Существует три типа потоков: ввода (input), вывода (output) и ввода-вывода (input/output).

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

Для использования стандартных потоков необходимо подключить библиотеку iostream.

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

Для работы с файловыми потоками необходимо подключить библиотеку fstream.

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

Для работы со строковыми потоками необходимо подключить библиотеку sstream.

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

Для вывода в форматированный поток используется оператор вставки <<.

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

Для ввода из форматированных потоков используется оператор извлечения >>.

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

Для вывода в форматированный поток используются методы width(), precision(), fill(), и т.д.

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

Для ввода из форматированного потока используются методы getline(), ignore(), peek(), и т.д.

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

Существуют три режима для открытия файловых потоков: ввод (ios::in), вывод (ios::out) и добавление (ios::app).

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

Для добавления записей в файл используется режим ios::app.

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

В конструкторе ifstream file(“f.txt”) используется режим ios::in, который открывает файл для чтения.

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

В конструкторе fstream file(“f.txt”) используется режим ios::in|ios::out, который открывает файл для чтения и записи.

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

В конструкторе ofstream file("f.txt") используется режим ios::out | ios::trunc, который открывает файл для вывода данных и усекает его до нулевой длины, если файл уже существует.

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

При открытии потока в режиме ios::out|ios::app данные будут добавляться в конец файла, если он уже существует. Если файл не существует, он будет создан. Для открытия потока в этом режиме можно использовать конструктор ofstream с указанием имени файла и режима открытия:

ofstream myfile("example.txt", ios::out | ios::app);

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

При открытии потока в режиме ios::out|ios::trunc, если файл уже существует, то его содержимое будет усечено до нулевой длины, а если файл не существует, то он будет создан. Для открытия потока в этом режиме можно использовать конструктор ofstream с указанием имени файла и режима открытия:

ofstream myfile("example.txt", ios::out | ios::trunc);

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

Поток в режиме ios::out|ios::in|ios::trunc не может быть открыт, так как этот режим сочетает в себе возможность вывода и чтения данных из файла, а усечение файла до нулевой длины при этом приведет к потере всех записанных в него данных. Если требуется каким-либо образом изменять содержимое файла, следует использовать другие режимы открытия потока.

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

Для открытия файла для чтения в C++ используется поток ifstream с режимом ios::in. Для этого можно использовать конструктор ifstream с указанием имени файла и режима открытия:

ifstream myfile("example.txt", ios::in);

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

Для открытия файла для записи в C++ используется поток ofstream с режимом ios::out. Для этого можно использовать конструктор ofstream с указанием имени файла и режима открытия:

ofstream myfile("example.txt", ios::out);

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

Ниже приведены примеры открытия файловых потоков в различных режимах:

Открытие файла для записи: ofstream outputFile("example.txt");

Открытие файла для чтения: ifstream inputFile("example.txt");

Открытие файла для добавления записей в конец: ofstream appendFile("example.txt", ios::app);

Открытие файла для чтения и записи: fstream inOutFile("example.txt", ios::in | ios::out);

Открытие файла для чтения и записи, обрезание файла до нулевой длины: fstream truncFile("example.txt", ios::trunc | ios::in | ios::out);

**21. Привести примеры чтения объектов из потока.**

Чтение объектов из потока в C++ может быть выполнено с помощью операторов извлечения (>>). Ниже приведены примеры чтения различных типов данных из потока:

Чтение целого числа: int number; inputFile >> number;

Чтение строки: string str; getline(inputFile, str);

Чтение символа: char ch; inputFile >> ch;

Чтение числа с плавающей точкой: double value; inputFile >> value;

**22. Привести примеры записи объектов в поток.**

Запись объектов в поток в C++ может быть выполнена с помощью операторов вставки (<<). Ниже приведены примеры записи различных типов данных в поток:

Запись целого числа: int number = 10; outputFile << number;

Запись строки: string str = "example"; outputFile << str;

Запись символа: char ch = 'a'; outputFile << ch;

Запись числа с плавающей точкой: double value = 3.14; outputFile << value;

**23. Сформулировать алгоритм удаления записей из файла.**

Алгоритм удаления записей из файла может быть следующим:

* Открыть файл в режиме чтения и записи.
* Создать новый файл для сохранения обновленных данных.
* Прочитать первую запись в файле.
* Если запись должна быть удалена, пропустить ее и перейти к следующей записи.
* Если запись не должна быть удалена, записать ее в новый файл.
* Повторять шаги 3-5 для каждой записи в файле.
* Закрыть и удалить старый файл.
* Переименовать новый файл в имя старого файла.

**24. Сформулировать алгоритм добавления записей в файл.**

Алгоритм добавления записей в файл может быть следующим:

* Открыть файл в режиме чтения и записи.
* Перейти в конец файла.
* Записать новую запись в файл.
* Закрыть файл.

**25. Сформулировать алгоритм изменения записей в файле.**

Алгоритм изменения записей в файле может быть следующим:

* Открыть файл в режиме чтения и записи.
* Создать новый файл для сохранения обновленных данных.
* Прочитать первую запись в файле.
* Если запись должна быть изменена, изменить ее и записать в новый файл.
* Если запись не должна быть изменена, записать ее в новый файл без изменений.
* Повторять шаги 3-5 для каждой записи в файле.
* Закрыть и удалить старый файл.
* Переименовать новый файл в имя старого файла.

**Диаграмма класса**

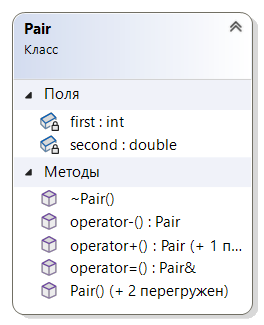


Рисунок 1 – диаграмма класс Pair

**Описание класса**

**Класс Pair**

#pragma once

#include <iostream>

#include <iomanip>

#include <conio.h>

#include <fstream>

#include <string>

using namespace std;

class Pair

{

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

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

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

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

friend void print\_file(const string& path, const int& count);

friend int del\_from\_file(const string& path, const int& count);

friend void add\_number(const string& path, const int& count);

private:

int first;

double second;

public:

Pair();

Pair(int first, double second);

Pair(const Pair& ob);

Pair operator-(Pair& pair) const;

Pair operator+(const int data) const;

Pair operator+(const double data) const;

Pair& operator=(const Pair& pair);

~Pair();

};

**Определение методов**

**Класс Pair**

#include "Pair.h"

Pair::Pair()

{

first = 0;

second = 0;

}

Pair::Pair(int first, double second)

{

this->first = first;

this->second = second;

}

Pair::Pair(const Pair& ob)

{

this->first = ob.first;

this->second = ob.second;

}

Pair Pair::operator-(Pair& pair) const

{

Pair temp = \*this;

temp.first -= pair.first;

temp.second -= pair.second;

return temp;

}

Pair Pair::operator+(int data) const

{

Pair temp = \*this;

temp.first += data;

return temp;

}

Pair Pair::operator+(double data) const

{

Pair temp = \*this;

temp.second += data;

return temp;

}

Pair& Pair::operator=(const Pair& pair)

{

if (this == &pair)

return \*this;

first = pair.first;

second = pair.second;

return \*this;

}

ostream& operator<<(ostream& out, const Pair& ob)

{

out << "(" << ob.first << ":" << ob.second << ")";

return out;

}

istream& operator>>(istream& in, Pair& ob)

{

cout << "first(int): "; in >> ob.first;

cout << "second(double): "; in >> ob.second;

return in;

}

fstream& operator>>(fstream& fin, Pair& ob)

{

fin >> ob.first;

fin >> ob.second;

return fin;

}

fstream& operator<<(fstream& fout, const Pair& ob)

{

fout << ob.first << ' ' << ob.second << '\n';

return fout;

}

Pair::~Pair()

{

}

**Результаты работы программы**

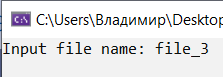


Рисунок 2 – ввод имени файла

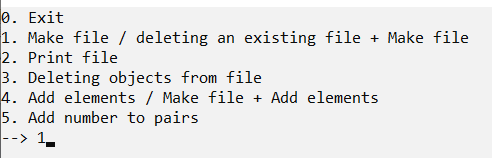


Рисунок 3 – создание файла с введенным именем

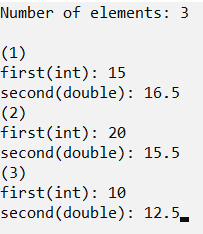


Рисунок 4 – заполнение файла

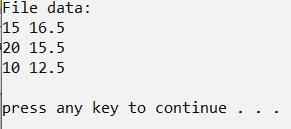


Рисунок 5 – вывод данных из файла

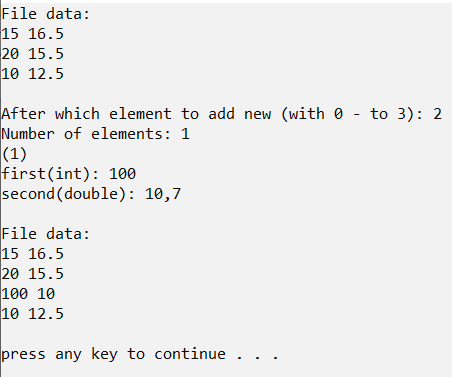


Рисунок 6 – добавление данных в файл после 2 элемента