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 ПО ЛАБОРАТОРНОЙ РАБОТЕ №10**

Дисциплина: «Основы алгоритмизации и программирования»

Тема : "Сохранение данных в файле с использованием потоков"

Выполнил работу:

Студент группы РИС-22-2Б

Бехтольт Д.А.

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).

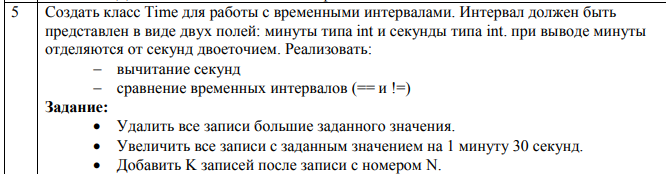
3. Написать функцию для чтения и просмотра объектов из потока.

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

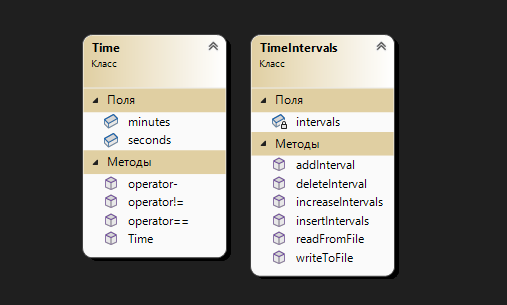
5. Написать функцию для добавления объектов в поток в соответствии с заданием варианта. Для выполнения задания выполнить перегрузку необходимых операций.

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.



**Диаграмма классов**

****

**Код**

#include <iostream>

#include <fstream>

#include <sstream>

#include <vector>

using namespace std;

class Time {

public:

Time(int m, int s) : minutes(m), seconds(s) {}

int minutes;

int seconds;

Time operator-(int s) {

int newSeconds = seconds - s;

if (newSeconds < 0) {

minutes -= 1;

newSeconds += 60;

}

return Time(minutes, newSeconds);

}

bool operator==(const Time& other) {

return minutes == other.minutes && seconds == other.seconds;

}

bool operator!=(const Time& other) {

return !(\*this == other);

}

friend ostream& operator<<(ostream& os, const Time& t) {

os << t.minutes << ":" << t.seconds;

return os;

}

};

class TimeIntervals {

public:

void addInterval(Time t) {

intervals.push\_back(t);

}

void deleteInterval(Time t) {

for (auto it = intervals.begin(); it != intervals.end(); ) {

if (\*it == t) {

it = intervals.erase(it);

}

else {

++it;

}

}

}

void increaseIntervals(Time t) {

for (auto& interval : intervals) {

if (interval == t) {

interval = interval - 90;

}

}

}

void insertIntervals(int n, int k) {

Time t(0, 0);

auto it = intervals.begin();

advance(it, n);

intervals.insert(it, k, t);

}

void writeToFile(string fileName) {

ofstream outFile(fileName);

if (!outFile) {

cerr << "Error: Could not open file " << fileName << endl;

return;

}

for (auto& interval : intervals) {

outFile << interval.minutes << "," << interval.seconds << endl;

}

outFile.close();

}

void readFromFile(string fileName) {

ifstream inFile(fileName);

if (!inFile) {

cerr << "Error: Could not open file " << fileName << endl;

return;

}

intervals.clear();

string line;

while (getline(inFile, line)) {

istringstream iss(line);

int m, s;

char comma;

if (iss >> m >> comma >> s) {

intervals.push\_back(Time(m, s));

}

}

inFile.close();

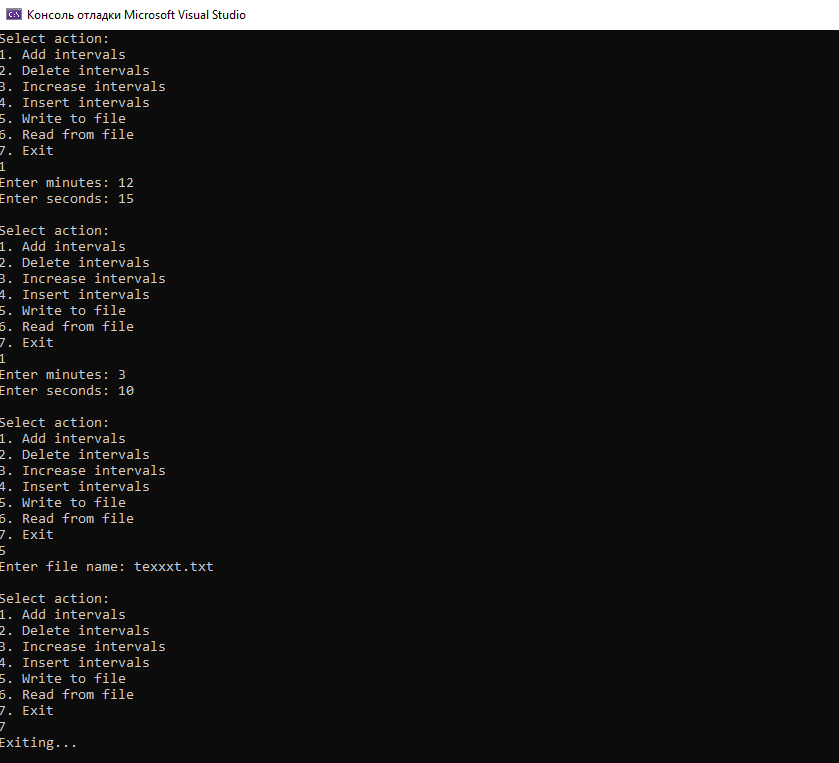
}

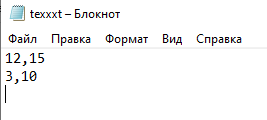
private:

vector<Time> intervals;

};

**Результаты работы программы**

****

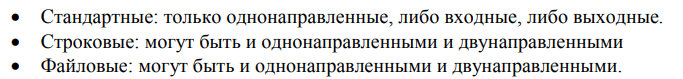
****

**Ответы на контрольные вопросы**

**1. Что такое поток?**

Поток - это абстрактный объект, который представляет собой последовательность байтов, которые могут быть считаны из устройства ввода или записаны в устройство вывода. Потоки представляют собой абстракцию для ввода и вывода данных и используются для чтения или записи данных в файлы, сетевые соединения, память и т.д.

**2. Какие типы потоков существуют?**



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

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

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

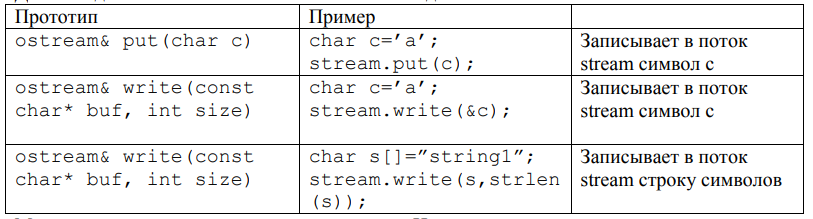
**6. Какая операция используется при выводе в форматированный поток?**

<< (бинарный оператор вставки) 

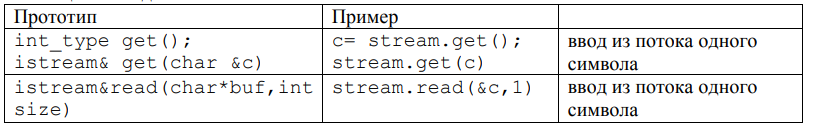
**7. Какая операция используется при вводе из форматированных потоков?**

операция извлечения из потока >> 

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

operator<< - оператор вставки, который позволяет добавить значение в поток. Он перегружен для разных типов данных и может быть использован для вывода числовых значений, строк, символов и других типов данных.

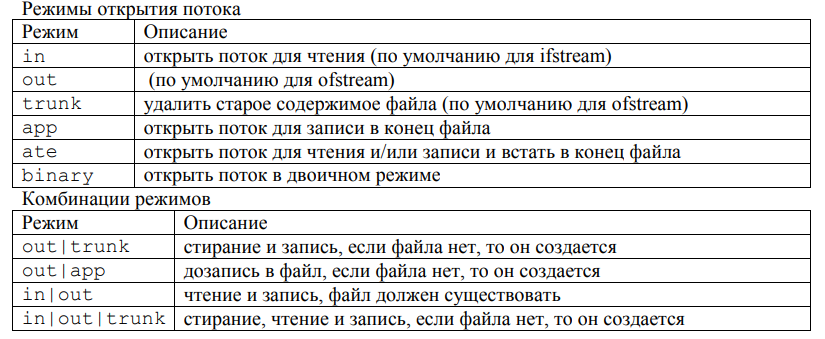
**9. Какие методы используется при вводе из форматированного потока?**



- operator>>: основной метод, используемый для чтения из потока. Он позволяет считывать данные из потока и сохранять их в переменной или объекте.

- getline: метод, используемый для чтения строки из потока. Он позволяет считать всю строку, включая пробелы и символы новой строки.

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



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

out

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

**12. Какой режим (комбинация режимов) используется в конструкторе**

**ifstream file("f.txt")?**

ios::in

**13. Какой режим (комбинация режимов) используется в конструкторе**

**fstream file("f.txt")?**

ios::in | ios::out

**14. Какой режим (комбинация режимов) используется в конструкторе**

**ofstream file("f.txt")?**

ios::out

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

ios::out указывает на то, что файл будет открыт для записи, а ios::app гарантирует, что при каждой записи данные будут добавляться в конец файла. Если файл не существует, он будет создан. Если файл уже существует, то данные будут добавляться в конец файла, не перезаписывая его содержимое.

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

ios::out - файл будет открыт в режиме вывода и при этом, если файл уже существует, его содержимое будет удалено (ios::trunc). Если же файл не существует, он будет создан.

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

ios::out указывает на то, что файл будет использоваться для записи, ios::in указывает на то, что файл будет использоваться для чтения, а ios::trunc указывает на то, что содержимое файла будет очищено перед записью в него.

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



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



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

- добавляет новое содержимое в конец файла, не удаляя предыдущее содержимое



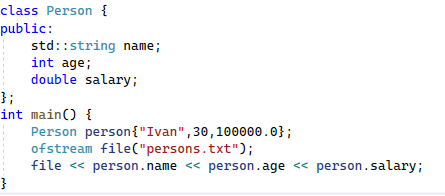
**-** открыт для чтения и записи



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



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



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

1) открыть файл в режиме чтения

2) открыть вспомогательный файл в режиме записи

3) пока не конец файла, копировать записи из файла в вспомогательный, пропуская те, которые нужно удалить

4) удалить изначальный файл

5) переименовать вспомогательный файл в изначальный

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

1) открыть файл в режиме чтения

2) открыть вспомогательный файл в режиме записи

3) пока не конец файла, копировать записи из файла в вспомогательный, если встретится запись, после которой нужно добавить новую , остановиться, начать добавлять новые, затем продолжить копирование старых записей из изначального файла

4) удалить изначальный файл

5) переименовать вспомогательный файл в изначальный

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

1) открыть файл в режиме чтения

2) открыть вспомогательный файл в режиме записи

3) пока не конец файла, копировать записи из файла в вспомогательный, пропуская те, которые нужно удалить, если встретится запись, после которой нужно добавить новую , остановиться, начать добавлять новые, затем продолжить копирование старых записей из изначального файла

4) удалить изначальный файл

5) переименовать вспомогательный файл в изначальный