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: Сохранение данных в файле с использованием потоков.

Выполнил студент ИВТ-22-2б:

Бобриков Михаил Александрович

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 3.

Создать класс Time для работы с временными интервалами. Интервал должен быть представлен в виде двух полей: минуты типа int и секунды типа int. При выводе минуты отделяются от секунд двоеточием. Реализовать:

* добавление секунд (учесть, что в минуте не может быть более 60 секунд)
* вычитание секунд

**Задание:**

* Удалить все записи равные заданному значению.
* Уменьшить все записи с заданным значением на 1 минуту 30 секунд. Значение интервала не должно быть меньше 0 минут 0 секунд.
* Добавить K записей после элемента с заданным значением.

**Описание класса**

Класс Time. Поля типа private: min: int, sec: int.

Методы типа public: 3 вида конструкторов и деструктор; селекторы и модификаторы для полей; перегруженные операторы присваивания operator=(const Time &), вычитания operator-(const Time &), сложения operator+( const Time &), перегруженные операторы ввода-вывода. Дружественные функции для ввода/вывода данных для потока и для файла.

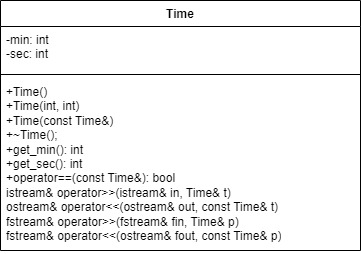
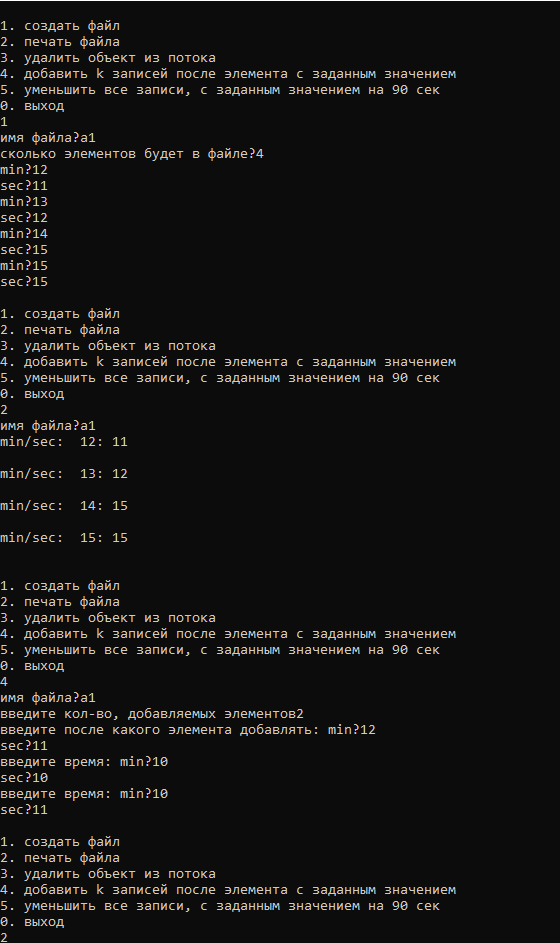


Рис. 1. UML-диаграмма классов

**Вывод 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<<

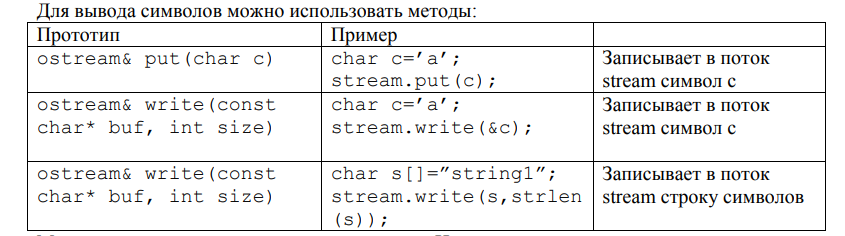
stream << 3.4 << ’\n’; //Вывод константы

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

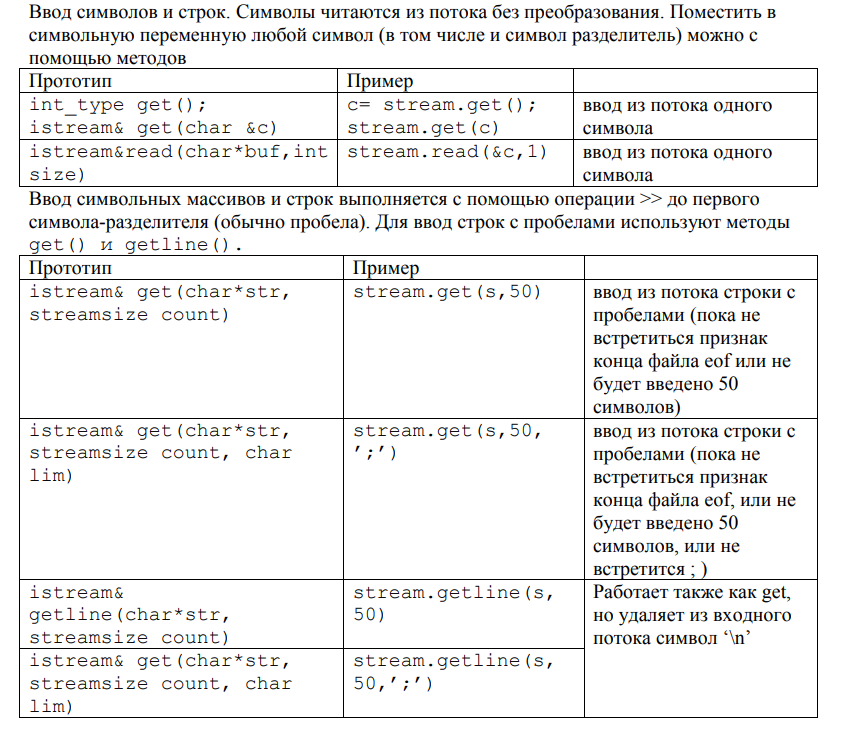
Для форматируемых потоков ввод осуществляется перегруженной операцией сдвига вправо operator>>

stream >> a; //Вывод переменной a

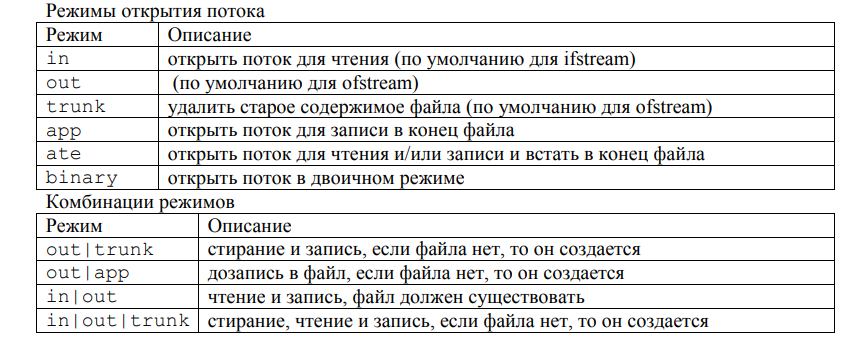
*8. Какие методы используются при выводе в форматированный поток?*



*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.