Министерство науки и высшего образования Российской Федерации Федеральное государственное автономное образовательное учреждение высшего образования

«Пермский национальный исследовательский политехнический университет»

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы» направление подготовки: 09.03.04 – «Программная инженерия»

Лабораторная работа №10.

" Сохранение данных в файле с использованием потоков "

15 вариант

Выполнил студент гр. РИС-24-2б

Иванова Елена Олеговна

Проверил:

Доц. Каф. ИТАС

Ольга Андреевна Полякова

(оценка) (подпись)

(дата)

г. Пермь, 2025

Цель задания:

1. Создание консольного приложения, состоящего из нескольких файлов в системе программирования Visual Studio.
2. Разработка программы, в которой данные сохраняются в файле, корректируются и выводятся из файла на печать. Работа с файлом осуществляется с использованием потоковых классов

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.

Задача:

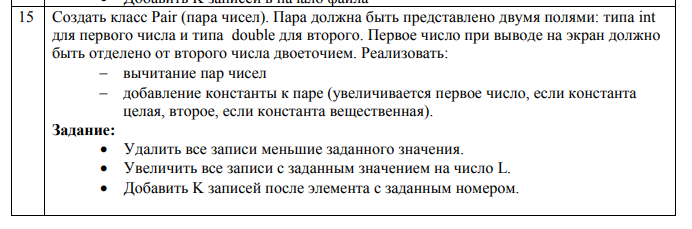
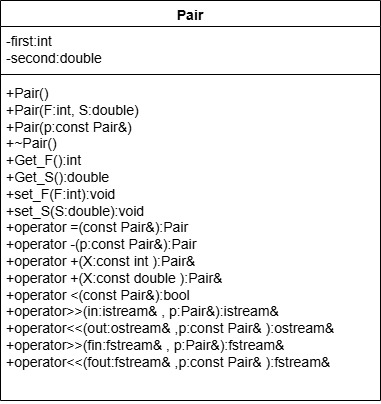
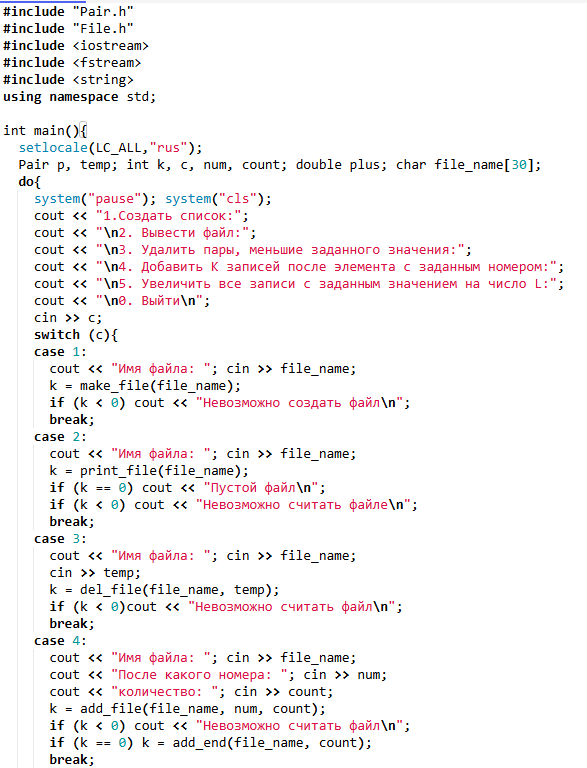
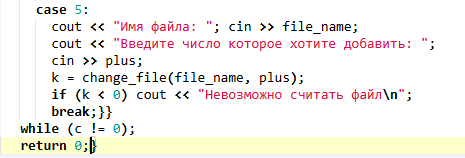


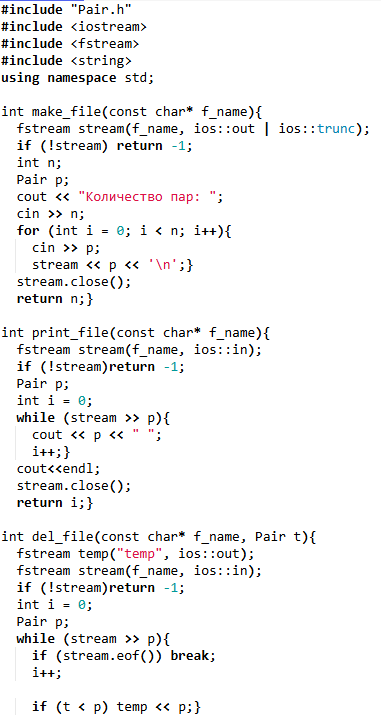
Диаграмма.

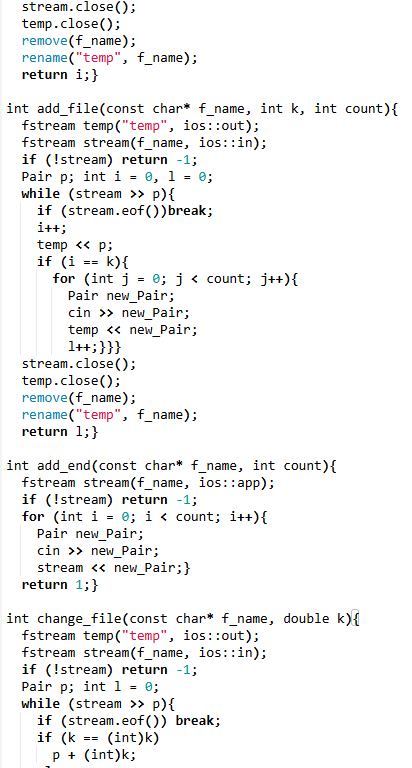


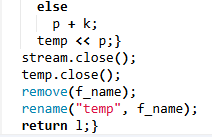
Код.

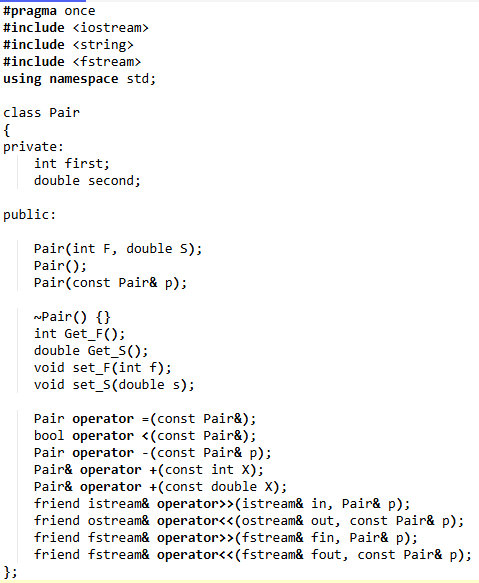


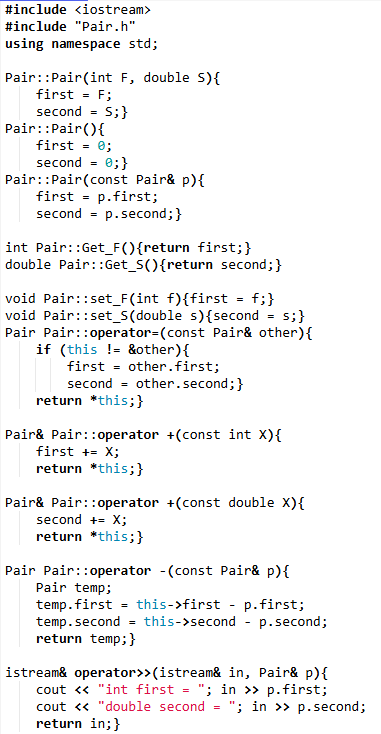


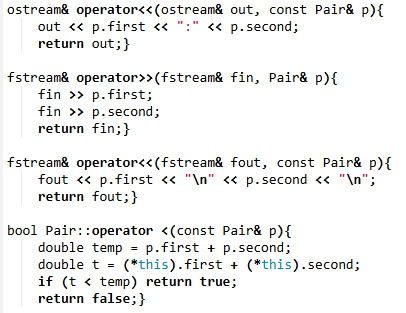




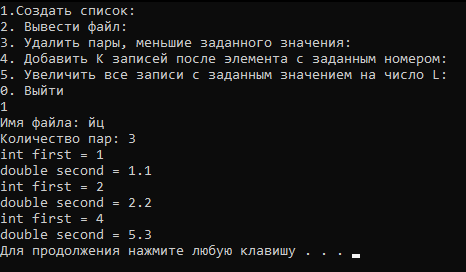


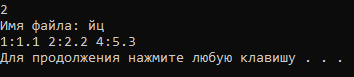


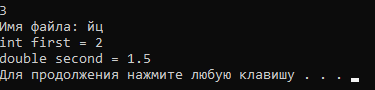


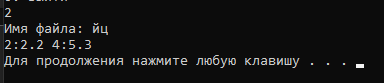


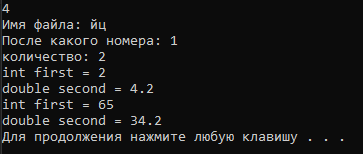
Вывод.

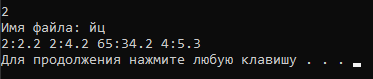


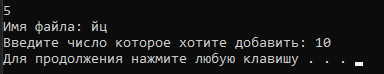


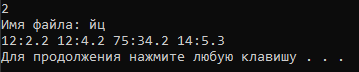




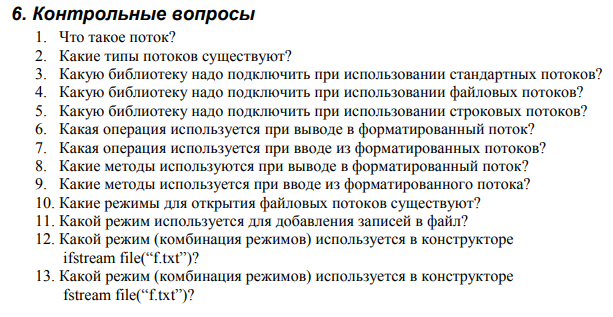


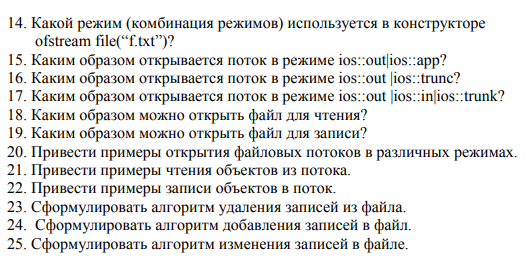






Контрольные вопросы.





1. 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. Потоки бывают

⦁ Стандартные: только однонаправленные, либо входные, либо выходные.

⦁ Строковые: могут быть и однонаправленными и двунаправленными

⦁ Файловые: могут быть и однонаправленными и двунаправленными.

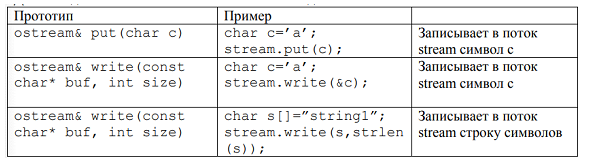
1. #include <iostream>
2. #include <fstream>
3. #include <sstream>
4. Для форматируемых потоков вывод, как правило, осуществляется перегруженной операцией сдвига влево operator<<.

stream<<3.4;stream<<’\n’;

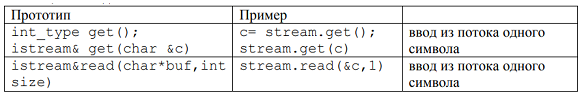
1. Для форматируемых потоков ввод перегруженной операцией сдвига вправо operator>>.

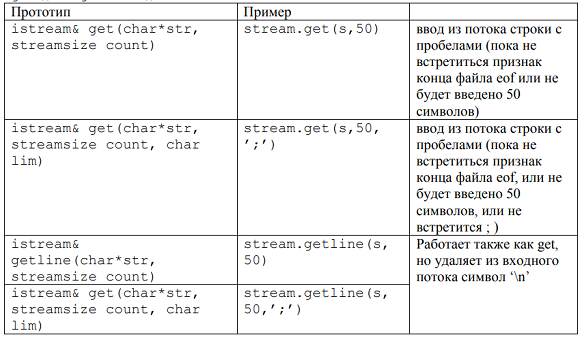
stream>>a>>b;

1. Для вывода символов можно использовать методы:

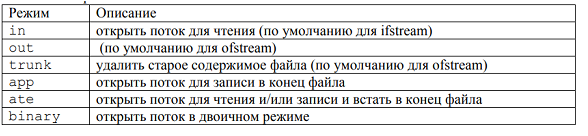


1. Поместить в символьную переменную любой символ (в том числе и символ разделитель) можно с помощью методов:

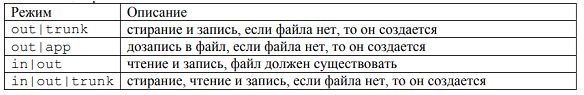




1. Режимы открытия потока:



Комбинации режимов:



1. Для добавления записи в конец файла используется режим app.
2. Используется режим in - открыть поток для чтения (по умолчанию для ifstream).
3. Используется комбинация режимов in | out - чтение и запись, файл должен существовать.
4. Используется режим out - открыть поток для записи в файл (по умолчанию для ofstream).
5. Поток открывается для дозапись в файл, если файла нет, то он создаётся.
6. Поток открывается для стирания и записи, если файла нет, то он создаётся.
7. Поток открывается для стирания, чтения и записи, если файла нет, то он создаётся.
8. Файл может быть открыт либо явно методом open(), либо неявно – конструктором при создании потока.

Операции обмена между файлом и программой зависят от типа связываемого с файлом 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.

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

Операции обмена между файлом и программой зависят от типа связываемого с файлом потока.

Для открытия файла для записи можно использовать различные режимы:

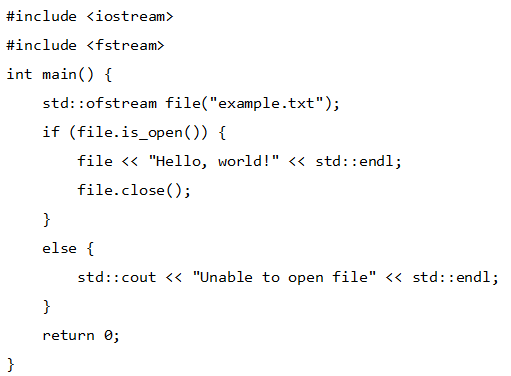
out - открыть поток для записи в файл (по умолчанию для ofstream);

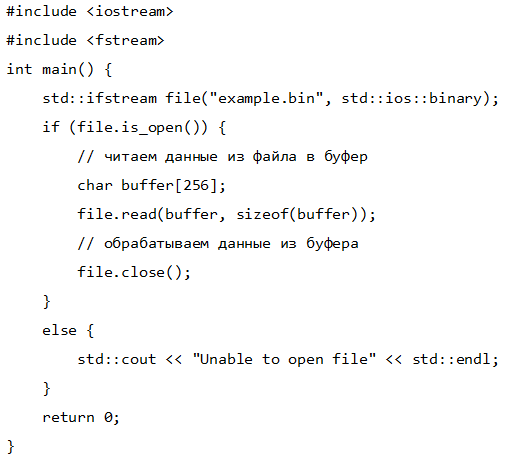
trunk - удалить старое содержимое файла (по умолчанию для оfstream);

app - открыть поток для записи в конец файла

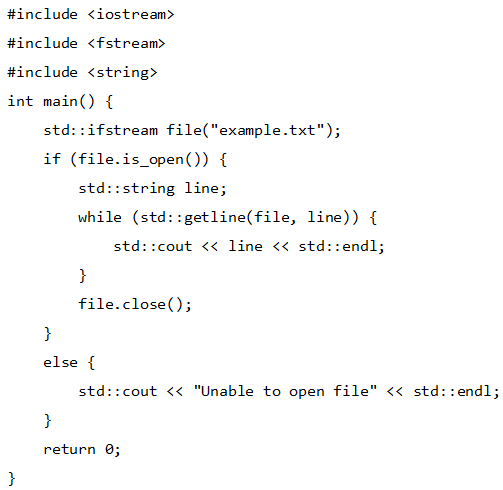
ate - открыть поток для чтения и/или записи и встать в конец файла

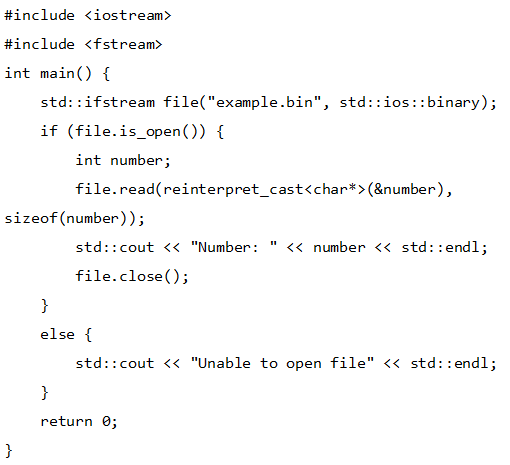
1. Привести примеры открытия файловых потоков в различных режимах:



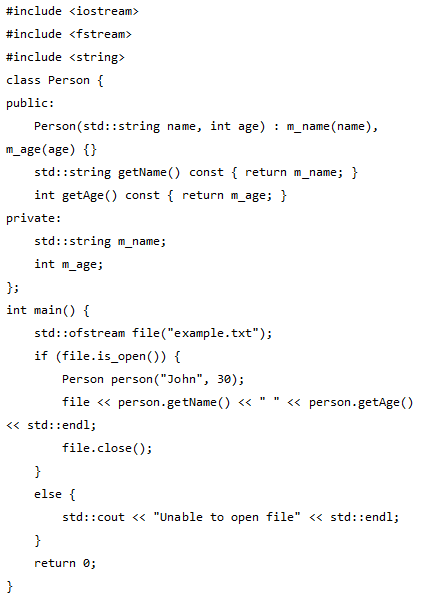


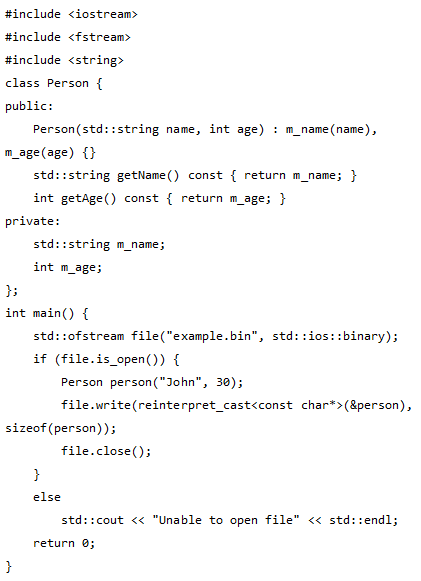
1. Привести примеры чтения объектов из потока.





1. Привести примеры записи объектов в поток.





1. Алгоритм удаления записей из файла в С++ может быть следующим:

1. Открыть файл для чтения и записи.

2. Создать временный файл.

3. Прочитать записи из исходного файла и проверять каждую запись на соответствие критерию удаления.

4. Записать нужные записи во временный файл.

5. Закрыть исходный файл и временный файл.

6. Удалить исходный файл с помощью функции.

7. Переименовать временный файл в имя исходного файла.

1. Алгоритм добавления записей в файл в С++ может быть следующим:
2. Открыть файл для записи с помощью функции fstream::open().
3. Проверить, что файл успешно открыт с помощью функции fstream::is\_open().
4. Создать объект записи, который будет содержать данные для записи в файл.
5. Заполнить объект записи данными.
6. Записать объект записи в файл с помощью оператора << или функции fstream::write().
7. Проверить, что запись прошла успешно с помощью функции.
8. Повторять шаги 3-6 для каждой новой записи.
9. Закрыть файл с помощью функции fstream::close().
10. Алгоритм изменения записей в файле в С++ может быть следующим:
11. Открыть файл для чтения и записи с помощью функции std::fstream.
12. Проверить, что файл успешно открыт.
13. Считать данные из файла в структуру или массив данных.
14. Изменить необходимые данные в структуре или массиве.
15. Перезаписать измененные данные в файл.
16. Закрыть файл.