Министерство высшего образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

**«Пермский национальный исследовательский политехнический университет» (ПНИПУ)**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

ОТЧЁТ

по лабораторной работе №10

Тема: «Сохранение данных в файле с использованием потоков»

Выполнил

Студент группы РИС-22-2б

Чиртулов М.В.

Проверил доц. 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 14:

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:

 операции сравнения (==,!=).

 вычитание константы из пары (уменьшается первое число, если константа

целая, второе, если константа вещественная).

**Задание:**

 Удалить все записи из интервала от k1 до k2, где k1 и k2 переменные типа Pair.

 Увеличить все записи с заданным значением в два раза.

 Добавить K записей в начало файла

# 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.

1. Какие типы потоков существуют?

Потоки бывают

* Стандартные: только однонаправленные, либо входные, либо выходные.
* Строковые: могут быть и однонаправленными и двунаправленными
* Файловые: могут быть и однонаправленными и двунаправленными.

1. 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.

1. 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;

1. 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;

1. Какая операция используется при выводе в форматированный поток?

Для форматируемых потоков вывод, как правило, осуществляется перегруженной операцией сдвига влево operator<<.

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

1. Какая операция используется при вводе из форматированных потоков?

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

stream>>a>>b;

Ввод в переменную завершается, если очередной символ в потоке не соответствует типу вводимого значения. Стандартными разделителями в потоке являются: ‘ ‘(пробел), ‘\n’, ‘\t’.

1. Какие методы используются при выводе в форматированный поток?

Для вывода символов можно использовать методы:



1. Какие методы используется при вводе из форматированного потока?



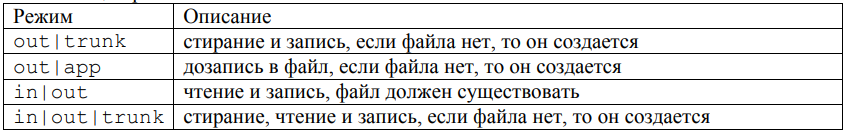


1. Какие режимы для открытия файловых потоков существуют?

Режимы открытия потока:

****

Комбинации режимов:



1. 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.

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

Используется режим in - открыть поток для чтения (по умолчанию для ifstream).

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

Используется комбинация режимов in | out - чтение и запись, файл должен существовать.

1. Какой режим (комбинация режимов) используется в конструкторе ofstream file(“f.txt”)?

Используется режим out - открыть поток для записи в файл (по умолчанию для ofstream).

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

Поток открывается для дозапись в файл, если файла нет, то он создаётся.

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

Поток открывается для стирания и записи, если файла нет, то он создаётся.

1. 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.

1. Каким образом можно открыть файл для чтения?

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

1. Каким образом можно открыть файл для записи?

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

1. 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);

1. 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;

1. 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;

1. Сформулировать алгоритм удаления записей из файла.

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. Сформулировать алгоритм добавления записей в файл.

1. Открыть файл для записи с помощью функции fstream::open().

2. Проверить, что файл успешно открыт с помощью функции fstream::is\_open().

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 fstream::write().

6. Проверить, что запись прошла успешно с помощью функции.

7. Повторять шаги 3-6 для каждой новой записи.

8. Закрыть файл с помощью функции fstream::close().

1. 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.

**UML-диаграмма**

