**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В. И. Ульянова (Ленина)**

**Кафедра информационных систем**

отчет

**по курсовой работе**

**по дисциплине «Программирование»**

|  |  |  |
| --- | --- | --- |
| Студент(ка) гр. |  | Коврижных А.Н. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2023

**Цели работы:** разработать алгоритм и написать программу, которая объединяет 4 лабораторных работы.

**Основные теоретические положения:**

Для работы с файлами в языке C++ используются потоки трех видов:

· поток ввода (класс ifstream);

· поток вывода (класс ofstream);

· поток ввода-вывода (класс fstream).

Класс ifstream используется для выполнения чтения данных из файлов. Поток ofstream – для записи данных в файлы. Поток fstream – для чтения и записи данных в файлы.

Для использования этих классов потоков необходимо в программу включить заголовочный файл <fstream>.

Для выполнения любых работ с файлом необходимо:

1) создать поток одного из классов файловых потоков;

2) связать созданный поток с конкретным файлом (открыть конкретный файл для работы);

3) выполнить необходимые действия с файлом (чтение / запись данных);

4) закрыть файл.

Все инструменты для работы с системой ввода-вывода и потоками в языке С++ определены в стандартной библиотеке. Заголовочный файл iostream определяет следующие базовые типы для работы с потоками:

* istream и wistream: читают данные с потока
* ostream и wostream: записывают данные в поток
* iostream и wiostream: читают и записывают данные в поток

Для каждого типа определен его двойник, который начинается на букву w и который предназначен для поддержки данных типа wchar\_t.

Эти типы являются базовыми для других классов, управляющих потоками ввода-вывода.

Объект типа ostream получает значения различных типов, преобразует их в последовательность символов и передает их через буфер в определенное место для вывода (консоль, файл, сетевые интерфейсы и т.д.)

**Изображение выглядит как текст, линия, Шрифт, диаграмма

Автоматически созданное описание**

Поток istream получает через буфер из определенного места последовательности символов (с консоли, из файла, из сети и т.д.) и преобразует эти последовательности в значения различных типов. То есть когда мы вводим данные (с той же клавиатуры в консоли), сначала данные накапливаются в буфере и только затем передаются объекту istream.

**Изображение выглядит как текст, Шрифт, линия, белый

Автоматически созданное описание**

### Создание потока, открытие и закрытие файла

Либо последовательность из двух инструкций:

ofstream File;    - создали поток

File.open ( "E:\\test.txt" );   - связали поток с файлом (открыли файл)

Либо так:

​​​​​​​ofstream File ( "E:\\test.txt" );  - создали поток и открыли файл

После открытия файла необходимо обязательно проверить открылся ли файл.

Если файл открыть не удалось, то переменная потока (**File**) принимает значение **false**, если файл открыт – **true**. Следовательно, проверку успешного открытия файла можно выполнит так:

if  ( ! File)

     // Ошибка

Еще один способ – использовать функцию потока **is\_open ()**, которая также возвращает логическое значение в зависимости от результата операции открытия файла:

if  ( ! File.is\_open () )

     // Ошибка

Файл закрывается с помощью функции потока **close ()**:

​​​​​​​File.close ();

Подробнее об открытии файла. Каждый поток использует свой вариант функции **open**. Их прототипы выглядят так:

​​​​​​​void ifstream::open ( const char \* FileName,

                                    ios::openmode Mode = ios::in );

void ofstream::open ( const char \* FileName,

                                     ios::openmode Mode = ios::out | ios::trunc );

void fstream::open ( const char \* FileName,

                                    ios::openmode Mode = ios::in | ios::out );

Первый параметр определяет имя открываемого файла (представляет собой массив символов).

Второй параметр определяет режим открытия файла. Этот параметр имеет значение по умолчанию, то есть является необязательным. Возможные значения этого параметра:

·         **ios::app** – при открытии файла на запись (поток **ofstream**) обеспечивает добавление всех выводимых в файл данных в конец файла;

·         **ios::ate –**обеспечивает начало поиска данных в файле начиная с конца файла;

·         **ios::in –**файл открывается для чтения из него данных;

·         **ios::out –**файл открывается для записи данных в файл;

·         **ios::binary –**открытие файла в двоичном режиме (по умолчанию все файлы открываются в текстовом режиме);

·         **ios::trunc –**содержимое открываемого файла уничтожается (его длина становится равной 0).

Эти флаги можно комбинировать с помощью побитовой операции ИЛИ (|).

Если файл открывается без использования функции **open**, эти флаги тоже можно использовать:

**fstream File ( "E:\\test.txt", ios :: out | ios :: app );**- файл открывается на вывод с добавлением записываемых данных  в конец файла.

### Прямой доступ при работе с файлами

Во всех предыдущих примерах чтение и запись данных в файлы осуществлялась последовательно. Однако существуют методы, обеспечивающие возможность произвольного доступа к любому байту файла для осуществления записи или чтения данных именно в эту точку файла.

Поддержка этих методов основана на понятии указателя позиции в файле (или просто – указателя). В языке C++ поддерживается работа с двумя указателями. Один указатель (**get**-указатель) определяет место в файле, откуда должны быть прочитаны данные. Другой – **put**-указатель определяет место для записи данных. При выполнении операций чтения-записи соответствующие указатели автоматически перемещаются в файле на расстояние, равное объему прочитанных или записанных данных, тем самым подготавливая следующую операцию чтения или записи.

**Замечание.**Рассматриваемые далее методы ориентированы на использование **в двоичном режиме работы с файлами**. В текстовом режиме позиционирование указателей может оказаться не точным.

Перемещение указателей в потоках осуществляется с помощью соответствующих потоковых функций:

·         **ifstream &seekg ( off\_type offset, seekdir way );**    - перемещение **get**-указателя;

·         **ofstream &seekp ( off\_type offset, seekdir way );**   - перемещение **put**-указателя.

Перемещение соответствующего указателя осуществляется на **offset** байт (тип данных **off\_type**– целочисленный тип данных, может принимать отрицательные и положительные значения) относительно точки, задаваемой параметром **way**.

Тип данных **seekdir**параметра **way** имеет три возможных значения:

·         **ios :: beg** – начало файла;

·         **ios :: cur** – текущая позиция;

·         **ios :: end** – конец файла.

Существуют перегруженные функции позиционирования указателей, обеспечивающие абсолютную адресацию указателя в заданную позицию файла:

·         **ifstream & seekg ( pos\_type pos );**    - перемещение **get**-указателя в позицию **pos**;

·         **ofstream & seekp ( pos\_type pos );**   - перемещение **put**-указателя в позицию **pos**.

Тип данных **pos\_type**является целочисленным беззнаковым типом. Позиция 0 соответствует началу файла.

Определить текущие позиции указателей можно с помощью следующих потоковых функций:

**pos\_type tellg ();**  - возвращает текущую позицию **get**-указателя;

**pos\_type tellp ();**  - возвращает текущую позицию **put**-указателя;

**Постановка задач:**

Необходимо объединить все 4 лабораторные работы в единый проект. Нужно добавить инфраструктуру переключения между заданиями (интерактивное меню).

**Выполнение работы:**

В начале подключаем все лабораторные работы

#include "LabWork№1.cpp"

#include "LabWork№2.cpp"

#include "LabWork№3.cpp"

#include "LabWork№4.cpp"

Далее создается бесконечный цикл while(), внутри которого обращения к главным функция в каждой лабораторной работе.

case 1: system("cls"); main1(); break;

case 2: system("cls"); main2(); break;

case 3: system("cls"); main3(); break;

case 4: system("cls"); main4(); break;

В каждой лабораторной работе переименовал главной функцию. И в конце или в меню работы, создал выход из программы путем возращения.

Пример на 1 лабораторной работе:

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Рис. 1. – Меню 1 работы.

В четвертом кейсе идет возвращение в курсовую работу.

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Рис. 2. – Меню курсовой работы.

**Вывод:** в ходе курсовой работе я узнал, как работать с несколькими проектами. И закрепил материал предыдущих работ.