Лабораторная работа №3

Работа с файловыми потоками в С++.  
Текстовые файлы

***Цель работы:*** получить практические навыки решения задач с использованием текстовых файлов на языке С++ с использованием принципов объектно-ориентированного и визуального программирования.

**Порядок выполнения лабораторной работы**

1. Изучить теоретические сведения. Разобрать представленные примеры. Запустить первые четыре примера.
2. Получить вариант индивидуального задания у преподавателя. Предполается реализация двух задач разными способами: первый способ для первой задачи – использовать стандартные потоковые классы для решения задачи, второй способ для второй задачи – перегрузить операторы << и >> для работы со стандартными потоковыми классами.
3. Реализовать полученные задания. Для выбора решения задачи пользователем реализовать меню. Любое общение с пользователем должно сопровождаться поясняющимся текстом. Также в программе должно быть реализовано минимум 2 исключительных ситуации. Исключительные ситуации должны обрабатываться в отдельном классе. Обязательна оконная (формочная) реализация программы с использованием элементов управления.
4. Показать результат работы программы преподавателю.
5. Защитить лабораторную работу.

**Требования к отчету**

Отчет должен содержать:

1. Цель работы.

2. Задание.

3. Словесное описание исключительных ситуаций.

4. UML диаграмма классов.

5. Блок-схема программы.

6. Текст программы на языке C++ с комментариями.

7. Тесты (копии выходных форм программы/сформированных файлов).

8. Выводы.

# Теоретические сведения

**Потоковый ввод/вывод дисковых файлов**

Для работы с дисковыми файлами необходимо подключение заголовочного файла <fstream>, содержащего наборы специальных классов:

* ifstream — для ввода,
* ofstream — для вывода,
* fstream — для чтения и записи данных в один и тот же файл.

Чтобы получить возможность работать с дисковым файлом, нужно открыть его с указанием режима доступа, который определяется значением константы open-mode класса ios.

|  |  |  |
| --- | --- | --- |
| **Режим доступа** | **Стандарт** | **Действие** |
| app | нет | открывает файл для дозаписи |
| ate(atend) | да | при открытии файла устанавливает файловый указатель на конец файла |
| binary(bin) | да | открыть файл в двоичном представлении |
| in | да | открыть файл для чтения (ввода) |
| nocreate | нет | если файл не существует, то новый файл не создается |
| noreplace | нет | если файл уже существует, файл не перезаписывается |
| out | да | открыть файл для записи (вывода) |
| trunc | нет | Открывает и усекает существующий файл. Новая информация замещает существующую |

**Текстовые файлы**

*Создание и запись.*

Для создания текстового файла определяют объект класса ofstream и передают конструктору класса имя дискового файла в качестве первого параметра и режим доступа в качестве второго параметра:

ofstream out\_file(“Out.txt”, ios::out);

Можно объявить константу, определяющую режим открытия файла, например:

const ios::open\_mode=ios::out | ios::app;

После того, как предпринималась попытка открыть файл, следует убедиться в том, что файл открыт и готов для записи (или перезаписи):

if (! out\_file) { cerr<<”Error: unable to write to out.txt”<<endl;

exit(1);

}

Все сказанное верно и для файлов, открываемых для чтения (или входных файлов):

ifstream in\_file(“Input.txt”, ios::in);

if (! in\_file) { cerr<<”Error: unable to open Input.txt”<<endl;

exit(1);

}

При работе с текстовыми файлами наиболее часто встречаются четыре действия:

|  |  |
| --- | --- |
| * посимвольное чтение * посимвольная запись | * построчное чтение * построчная запись |

*Посимвольное чтение текста.*

Функция get(), которая является методом istream, применяется для посимвольного чтения текстового файла.

**Пример 1.** Посимвольное чтение файла и вывод его на экран.

# include <fstream>

# include <iostream>

using namespace std;

int main()

{

setlocale(LC\_ALL, "RUS");

char sym;

ifstream in\_file("Input\_file.txt", ios::in);

if (!in\_file) {

cerr << "Error input file" << endl;

exit(1);

}

while (in\_file) {

in\_file.get(sym);

cout << sym;

}

cout << endl;

system("pause");

return 0;

}

*Посимвольная запись текста.*

Функция put(), которая является методом ostream, позволяет осуществлять посимвольную запись данных в текстовый файл.

**Пример 2.**

# include <fstream>

# include <iostream>

# include <string>

using namespace std;

void main()

{

setlocale(LC\_ALL, "RUS");

string guote = "Зорко одно лишь сердце.Самого главного глазами не увидишь.А.де Сент Экзюпери";

ofstream out\_file("Out\_file.txt", ios::out);

if (!out\_file)

{

cerr << "Error output file" << endl;

exit(1);

}

for (int i = 0; i<guote.size(); i++)

out\_file.put(guote[i]);

cout << "Конец записи" << endl;

system("pause");

}

*Построчное чтение файла.*

Обычно построчное чтение и запись файлов работают быстрее посимвольных действий. Для чтения строки из файла воспользуемся функцией getline(), которая является методом класса ifstream. Функция читает строку (в том числе и разделители), пока не встретит символ новой строки ‘\n’, помещая ее в буфер (первый аргумент функции). Максимальный размер буфера задается как второй аргумент функции.

**Пример 3.**

# include <fstream>

# include <iostream>

using namespace std;

void main()

{

setlocale(LC\_ALL, "RUS");

const int LEN = 80;

char BUF[LEN];

ifstream in\_file("Input\_file.txt", ios::in);

if (!in\_file)

{

cerr << "Error input file" << endl;

exit(1);

}

while (in\_file) {

in\_file.getline(BUF, LEN);

cout << BUF << endl;

}

system("pause");

}

*Построчная запись текста.*

**Пример 4.**

# include <fstream>

# include <iostream>

# include <string>

using namespace std;

void main()

{

setlocale(LC\_ALL, "RUS");

ofstream out\_file("Out\_file.txt", ios::out);

if (!out\_file) {

cerr << "Error output file" << endl;

exit(1);

}

out\_file << "Я не знаю, где встретиться\n";

out\_file << "Нам придется с тобой, \n";

out\_file << "Глобус крутится - вертится, \n";

out\_file << "Словно шар голубой\n";

cout << "Файл записан" << endl;

system("pause");

}

Записываемые строки являются не объектами класса string, а строками типа \*char, завершающимися символом ‘\n’.

**Признак конца файла**

Признак конца файла приходится искать в файлах, открытых для чтения. Этот признак устанавливается в тот момент, когда в файле не осталось больше данных, которые можно считать.

Признак конца файла анализируется в выражении вида

while (! In\_file.eof()) { … }

Для этой цели нельзя пользоваться циклом

do { … } while (! In\_file.eof()), поскольку файл может оказаться пустым.

Однако проверка на конец файла не анализирует ошибки, которые могут встретиться в процессе чтения файла.

Для проверки как конца файла, так и наличия ошибок при его чтении пользуются условием выхода из цикла:

while (In\_file.good()) { … }

Оператор цикла

while (In\_file) { … }

выполняется до тех пор, пока нет ошибок, в том числе и конца файла (EOF).

## Имена файлов и аргументы командной строки

В приведенных выше примерах были использованы файлы с фиксированными именами. Однако более удобны программы, в которых имена файлов для чтения или записи можно ввести с клавиатуры или задать в качестве аргументов командной строки.

**Пример 5.**

# include <fstream>

# include <iostream>

# include <stdlib.h>

using namespace std;

int main(int argc, char \*argv[])

{

char I;

char c;

if (argc <= 1)

{

cout << "usage: primer.exe filename1 [filename2]" << endl;

system("pause");

return 1;

}

ifstream file;

for (I = 1; I <= argc - 1; I++)

{

file.open(argv[I]);

if (!file.is\_open())

{

cout << "Bad file" << endl;

system("pause");

return 1;

};

cout << "file " << argv[I] << endl;

while (file)

{

c = file.get();

cout << c;

};

file.close();

}

cout << endl;

system("pause");

return 0;

}

Для работы с файловыми потоками любого из стандартных типов, нужно перегрузить операторы ввода и вывода под требуемый тип данных или воспользоваться шаблоном класса, задаваемым с помощью ключевого слова template.

**Пример 6.**

*Файл myclass.h*

#include <iostream>

using namespace std;

class myclass

{

public:

int x;

float y;

friend ostream& operator << (ostream& out, myclass& m);

friend istream& operator >> (istream& out, myclass& m);

myclass(int x\_in, float y\_in);

myclass(void);

};

*Файл myclass.cpp*

#include "myclass.h"

myclass::myclass(int x\_in, float y\_in)

{

x = x\_in;

y = y\_in;

}

myclass::myclass(void)

{

x = 0;

y = 0.0;

}

ostream& operator << (ostream& out, myclass& m)

{

out << m.x << ' ' << m.y << endl;

return out;

}

istream& operator >> (istream& in, myclass& m)

{

cout << "Введите X: ";

in >> m.x;

cout << "Введите Y: ";

in >> m.y;

return in;

}

*Главный модуль*

#include "myclass.h"

int \_tmain(int argc, \_TCHAR\* argv[])

{

setlocale(LC\_ALL, "RUS");

myclass m1(5, 6.3);

cout << "m1: " << m1;

myclass m2;

cout << "m2: " << m2;

cout << "Введиту m2:" << endl;

cin >> m2;

cout << "m2: " << m2;

myclass m3 (3, 6.9);

cout << "m3: " << m3;

cout << "Введиту m3:" << endl;

cin >> m3;

cout << "m3: " << m3;

system("pause");

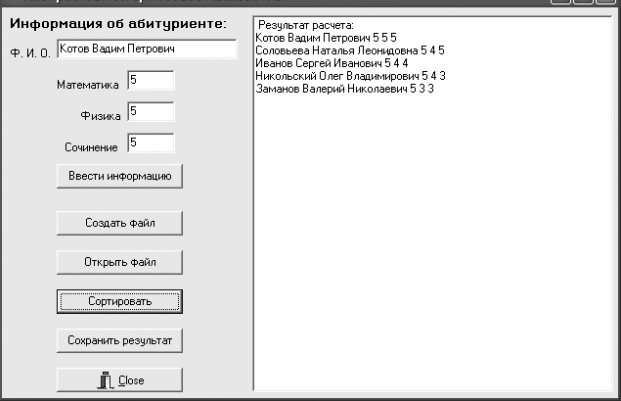
return 0;

}

**Контрольные вопросы**

1. Что такое поток?
2. Что представляет собой файловый указатель?
3. Перечислить режимы доступа к файлу.
4. Как открыть и как закрыть файл?

**Пример компоновки формы программы к лабораторной работе**



**Лабораторное задание**

**Задание первое**: реализовать свой класс и использовать стандартные потоковые классы для решения задачи.

## Варианты заданий

| **Номер варианта** | **Задание** |
| --- | --- |
| 1 | Дан файл *f*, компоненты которого являются целыми числами. Записать в файл *g*, компоненты файла *f*, исключив повторные вхождения чисел. |
| 2 | Дан файл *f*, компоненты которого являются действительными числами. Найти:   1. наибольшее из значений компонентов *f*; 2. наименьшее из значений компонентов с четными номерами; 3. наибольшее из значений модулей компонентов с нечетными номерами; 4. сумму наибольшего и наименьшего из значений компонентов файла *f*; 5. разность первой и последней компоненты файла *f*. |
| 3 | Дан символьный файл *f*. Подсчитать число вхождений в файл каждой из букв a, b, c, d, e, f.  Результат вывести в файл *g* в виде таблицы с комментариями. |
| 4 | Дан файл *f*, компоненты которого являются целыми числами. Записать в файл *g* все четные числа исходного файла, в файл *h* – все нечетные. Порядок следования чисел сохраняется. Записать в файл *g* и *h* комментарии. |
| 5 | Дан текстовый файл, содержащий программу на языке С. Проверить эту программу на соответствие числа открывающих и закрывающих фигурных скобок. |
| 6 | Дан символьный файл *f*. Найти и записать в файл *g* самое длинное слово файла *f*, снабдив его комментарием. |
| 7 | Дан файл *f*, компоненты которого являются целыми числами. Получить в файле *g* все компоненты файла *f*:   1. являющиеся четными числами; 2. делящиеся на 3 и не делящиеся на 7; 3. являющиеся точными квадратами.   Записать в файл *g* комментарий. |
| 8 | Дан файл *f*. Создать два файла, записав в первый из них все четные числа, расположив их в порядке возрастания, а во второй – все нечетные, расположив их в порядке убывания. |
| 9 | Дан текстовый файл *f*. Переформатировать исходный файл, разделяя его на строки так, чтобы каждая строка содержала столько символов, сколько содержит самая короткая строка исходного файла. |
| 10 | Дан файл *f*. Создать два файла, записав в первый из них среднее геометрическое всех четных чисел, а во второй – среднее арифметическое всех нечетных чисел. |
| 11 | Дан числовой файл *f*. Выбрать все значения, которые делятся нацело на 2 и 4, но не делятся на 6. Записать эти значения в файл *g*, а все остальные – в файл *h*. |
| 12 | Дан текстовый файл *f*. Определить, являются ли первые два символа цифрами и если да, то четно ли это число. Записать его в файл *g*, если оно четно и в *h* ,если оно нечетно. |
| 13 | Дан текстовый файл *f*. Создать новый файл *g* и переписать в него исходный файл в обратном порядке, разделив пробелами. |
| 14 | Дан текстовый файл f. Создать новый файл, включив в него только те строки исходного файла, которые содержат самый часто используемый символ исходного файла. |
| 15 | Дан текстовый файл, содержащий программу на языке С. Проверить эту программу на соответствие открывающих и закрывающих скобок разных типов - (), {}, []. |

**Задание второе**: реализовать свой класс и перегрузить в нём операторы << и >> для работы со стандартными потоковыми классами. Использовать этот класс для решения задания вместе со стандартными потоковыми классами.

| **Номер варианта** | **Задание** |
| --- | --- |
| 1 | Сформировать массив на диске, содержащий сведения о пациентах глазной клиники. Структурный тип содержит поля: фамилия пациента, пол, возраст, место проживания (город), диагноз.  Написать программу, которая выбирает необходимую информацию с диска и выводит на экран:  - количество иногородних, прибывших в поликлинику;  - список пациентов старше Х лет с диагнозом J. |
| 2 | Сформировать массив на диске, содержащий сведения о сотрудниках института. Структурный тип содержит поля: фамилия работающего, название отдела, год рождения, стаж работы, должность, оклад.  Написать программу, которая выбирает необходимую информацию с диска и выводит на экран:  - список сотрудников пенсионного возраста на сегодняшний день с указанием стажа работы;  - средний стаж, работающих в отделе Х. |
| 3 | Сформировать массив на диске, содержащий сведения об отправлении поездов дальнего следования с Казанского вокзала. Структурный тип содержит поля: номер поезда, станция назначения, время отправления, время в пути, наличие билетов.  Написать программу, которая выбирает необходимую информацию с диска и выводит на экран:  - время отравления поездов в город Х во временном интервале от А до В часов;  - наличие билетов на поезд с номером ХХХ. |
| 4 | Сформировать массив на диске, содержащий сведения о том, какие из пяти предлагаемых дисциплин по выбору желает изучать студент. Структурный тип содержит поля: фамилия студента, индекс группы, пять дисциплин, средний балл успеваемости. Выбираемая дисциплина отмечается символом 1, иначе – пробелом.  Написать программу, которая выбирает необходимую информацию с диска и выводит на экран список студентов, желающих прослушать дисциплину X. Если число желающих превышает 4 человека, то отобрать студентов, имеющих более высокий средний балл успеваемости. |
| 5 | Сформировать массив на диске, содержащий сведения о нападающих команды «Спартак». Структурный тип содержит поля: имена нападающих, число заброшенных ими шайб, число сделанных голевых передач, заработанное штрафное время.  Написать программу, которая выбирает необходимую информацию с диска и выводит на экран четырех лучших игроков по сумме очков (голы + передачи). |
| 6 | Сформировать массивна диске, содержащий сведения об ассортименте обуви в магазине фирмы. Структурный тип содержит поля: артикул, наименование, количество, стоимость одной пары. Артикул начинается с буквы Д – для дамской обуви, М – для мужской, П – для детской.  Написать программу, которая выбирает необходимую информацию с диска и выводит на экран:  - сведения о наличии и стоимости обуви артикула X;  - ассортиментный список дамской обуви с указанием наименования и имеющегося в наличии числа пар каждой модели. |
| 7 | Сформировать массив на диске, содержащий сведения о наличии билетов на рейсы аэрофлота. Структурный тип содержит поля: номер рейса, пункт назначения, время вылета, время прибытия, количество свободных мест в салоне.  Написать программу, которая выбирает необходимую информацию с диска и выводит на экран:  - время вылета самолетов в город X;  - наличие свободных мест на рейс в город X с временем отправления Y. |
| 8 | Сформировать массив на диске, содержащий сведения о личной коллекции книголюба. Структурный тип содержит поля: шифр книги, автор, название, год издания, местоположение (номер стеллажа).  Написать программу, которая выбирает необходимую информацию с диска и выводит на экран:  - местоположение книги, автора X названия Y;  - список книг автора Z, находящихся в коллекции;  - число книг издания XX года, имеющихся в библиотеке. |
| 9 | Сформировать массив на диске, содержащий сведения о сдаче студентами сессии. Структурный тип содержит поля: индекс группы, фамилия студента, оценки по пяти экзаменам.  Написать программу, которая выбирает необходимую информацию с диска и выводит на экран:   * фамилии неуспевающих студентов с указанием индексов групп и количества задолженностей; * средний балл, полученный каждым студентом группы X, и всей группы в целом. |
| 10 | Сформировать массив на диске, содержащий сведения об ассортименте игрушек в магазине. Структурный тип содержит поля: название игрушки, цена, количество, возрастные границы (2 - 5).  Написать программу, которая выбирает необходимую информацию с диска и выводит на экран:   * название игрушек, которые подходят детям от 1 до 3 лет; * стоимость самой дорогой игрушки и ее название; * название игрушки, которая по стоимости не превышает X руб. и подходит ребенку в возрасте от A до B лет.   Значения A, B, X вводятся с клавиатуры. |
| 11 | Сформировать массив на диске, содержащий сведения о телефонах абонентов. Структурный тип содержит поля: фамилия абонента, место жительства (название улицы, номер дома), год установки телефона.  Написать программу, которая выбирает необходимую информацию с диска и выводит на экран:   * номер телефона по вводимой с клавиатуры фамилии абонента; * количество установленных телефонов с XXXX года; * список номеров телефонов, принадлежащих жильцам определенного дома и улицы.   Номер года, название улицы и номер дома вводятся с клавиатуры. |
| 12 | Сформировать массив на диске, содержащий сведения о количестве изделий категорий A, B, C, собранных рабочим за месяц. Структурный тип содержит поля: фамилия сборщика, наименование цеха, количество изделий по категориям, собранных рабочим за месяц.  Считая заданными значения расценок SA, SB, SC за выполненную работу по сборке единицы изделия категорий A, B, C, выбрать необходимую информацию с диска и вывести на экран:   * общее количество изделий категорий A, B, C, собранных рабочим цеха; * средний размер заработной платы рабочих цеха X. |
| 13 | Сформировать массив на диске, содержащий сведения о количестве изделий, собранных сборщиками цеха за неделю. Структурный тип содержит поля: фамилия сборщика, количество изделий, собранных им ежедневно в течение шестидневной недели, т.е. раздельно в понедельник, вторник и т.д. Написать программу, которая выбирает необходимую информацию с диска и выводит на экран:   * фамилию сборщика и общее количество деталей, собранных им за неделю; * фамилию сборщика собравшего наибольшее количество изделий, и день, когда он достиг наивысшей производительности труда. |
| 14 | Сформировать массив на диске, содержащий сведения о личной видеоколлекции. Класс содержит поля: шифр по IMDB, название студии, название фильма, год издания, ка­тегория, жанр.  Написать программу, которая выбирает необходимую информацию с диска и выводит на экран:   * статистика по жанрам для студии Y; * список фильмов категории Z, находящихся в коллекции; * число фильмов XX года, имеющихся в коллекции. |
| 15 | Сформировать массив на диске, содержащий сведения о клиентах предприятия. Класс содержит поля: фамилия, имя, отчество, дата рождения, число покупок, потраченная сумма.  Написать программу, которая выбирает необходимую информацию с диска и выводит на экран:   * список клиентов у которых на этой неделе день рождения; * список клиентов у которых покупок более чем на Х; * средний размер покупки для клиентов пенсионного возраста (старше 60). |