Министерство Образования Республики Беларусь

Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных Технологий

Кафедра Информационных систем и технологий

**СТРУКТУРЫ ДАННЫХ**

**Двоичные и текстовые файлы**

**Лабораторная работа № 5**

Студент:

1курса 2 группы 1 подгруппы

Марушко Тимофей Фёдорович

Проверяющий:

Белодед Николай Иванович

**ОГЛАВЛЕНИЕ**

[**Теоритические сведения 3**](#_Toc192074976)

[**1. Схема БД (Базы Данных) 3**](#_Toc192074977)

[**2. Меню программы 3**](#_Toc192074978)

[**3. Графическое меню программы 8**](#_Toc192074979)

[**Решение практических задач 11**](#_Toc192074980)

[**Решение Дополнительных задач 13**](#_Toc192074981)

Теоритические сведения

**Файлы** — это наборы данных, которые хранятся на диске или другом устройстве хранения информации. Файлы можно разделить на два типа: текстовые и двоичные.

1. Текстовый Поток

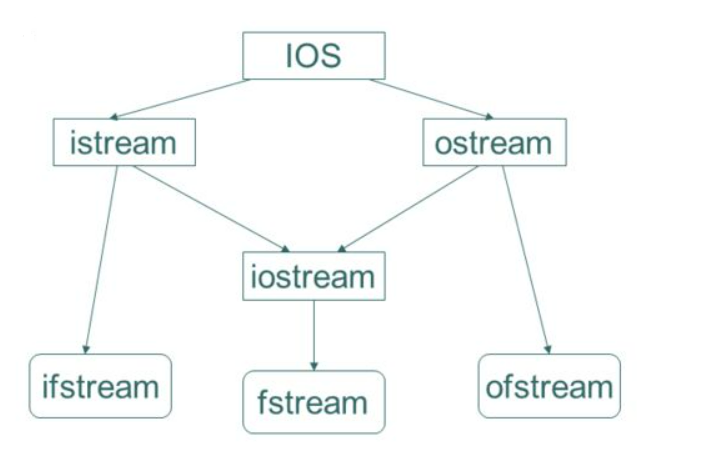
**Текстовый поток** — это последовательность символов. Они организуются по строкам, каждая из которых заканчивается символом «конца строки». Конец самого файла обозначается символом «конца файла». При записи информации в текстовый файл, просмотреть который можно с помощью любого текстового редактора, все данные преобразуются к символьному типу и хранятся в символьном виде. При передаче символов из потока на экран, часть из них не выводится (например, символ возврата каретки, перевода строки). В текстовом режиме каждый разделительный символ строки автоматически преобразуется в пару (возврат каретки – переход на новую строку).

Текстовые файлы могут быть созданы и изменены с помощью любого текстового редактора, такого как Microsoft Word, Notepad и т.д. Каждый символ в текстовом файле кодируется в соответствующий ASCII-код (American Standard Code for Information Interchange).

1. Двоичный поток

**Двоичный поток** — это последовательность байтов, которые однозначно соответствуют тому, что находится на внешнем устройстве. В двоичных файлах информация считывается и записывается в виде блоков определенного размера, в которых могут храниться данные любого вида и структуры.

Двоичные файлы могут быть созданы и изменены с помощью специальных программ, таких как бинарные редакторы. Каждый бит в двоичном файле имеет определенное значение, которое может быть интерпретировано в соответствии с определенной схемой кодирования.



1. Различия между текстовыми и двоичными файлами

Основные различия между текстовыми и двоичными файлами заключаются в способе хранения и интерпретации данных.

1)В текстовых файлах данные представлены в виде символов, в то время как в двоичных файлах данные представлены в виде битов. Текстовые файлы могут быть открыты и изменены любым текстовым редактором, в то время как для работы с двоичными файлами требуются специальные программы.

2)Текстовые файлы могут быть отформатированы и структурированы для лучшей читаемости, в то время как двоичные файлы обычно не содержат форматирования или структуры.

3)Текстовые файлы занимают меньше места на диске, чем двоичные файлы, потому что каждый символ занимает только один байт, в то время как каждый бит в двоичном файле занимает один бит.

| **Константа** | **Описание** |
| --- | --- |
| **ios\_base::in** | открыть файл для чтения |
| **ios\_base::out** | открыть файл для записи |
| **ios\_base::ate** | при открытии переместить указатель в конец файла |
| **ios\_base::app** | открыть файл для записи в конец файла |
| **ios\_base::trunc** | удалить содержимое файла, если он существует |
| **ios\_base::binary** | открытие файла в двоичном режиме |

4)Наконец, текстовые файлы могут быть легко читаемы и интерпретируемы человеком, в то время как двоичные файлы обычно требуют специальных программ и знаний для их интерпретации.

Обратите внимание на то, что флаги ate и app по описанию очень похожи, они оба перемещают указатель в конец файла, но флаг app позволяет производить запись, только в конец файла, а флаг ate просто переставляет флаг в конец файла и не ограничивает места записи.

Режимы открытия файлов можно устанавливать непосредственно при создании объекта или при вызове функции open().

ofstream fout("study.txt", ios\_base::app);

*// открываем файл для добавления информации к концу файла*

fout.open("study.txt", ios\_base::app);

*// открываем файл для добавления информации к концу файла*

Режимы открытия файлов можно комбинировать с помощью поразрядной логической операции **или |**, например: ios\_base::out | ios\_base::trunc — открытие файла для записи, предварительно очистив его.

Объекты класса ofstream, при связке с файлами по умолчанию содержат режимы открытия файлов  ios\_base::out | ios\_base::trunc**.**

**ofstream и ifstream - это классы стандартной библиотеки С++, которые используются для работы с файлами.**

**ofstream - это класс, который используется для записи данных в файл. Он наследуется от класса ostream, который определяет операторы << и >> для вывода и ввода данных. Для создания файла и записи данных в него необходимо создать объект класса ofstream. При создании объекта можно указать имя файла и режим открытия файла.**

**Вот пример кода, демонстрирующего использование класса ofstream :**

*#include* <iostream>

*#include* <fstream>

using namespace std;

int main() {

    ofstream file("example.txt");

*if* (file.is\_open()) {

        file << "Hello, world!" << endl;

        file.close();

    }

*else* {

        cout << "Ошибка открытия файла" << endl;

    }

*return* 0;

}

**В данном примере мы создаем объект класса ofstream с именем "example.txt" и записываем в него строку "Hello, world!" с помощью оператора <<. Затем мы закрываем файл с помощью функции close().**

**ifstream - это класс, который используется для чтения данных из файла. Он наследуется от класса istream, который также определяет операторы >> и << для ввода и вывода данных. Для чтения данных из файла необходимо создать объект класса ifstream. При создании объекта можно указать имя файла и режим открытия файла.**

**Вот пример кода, демонстрирующего использование класса ifstream:**

*#include* <iostream>

*#include* <fstream>

*#include* <string>

using namespace std;

int main() {

    ifstream file("example.txt");

    string line;

*if* (file.is\_open()) {

*while* (getline(file, line)) {

            cout << line << endl;

        }

        file.close();

    } *else* {

        cout << "Ошибка открытия файла" << endl;

    }

*return* 0;

}

**В данном примере мы создаем объект класса ifstream с именем "example.txt" и читаем данные из файла с помощью функции getline(). Затем мы выводим прочитанные данные на экран с помощью функции cout. В конце мы закрываем файл с помощью функции close().**

Решение практических задач

**Условие:**

Создание и запись данных в файл, название которого вводится с клавиатуры;

**Решение:**

void task1() {

    string filename = "data.txt";

    ofstream file(filename);

*if* (!file) {

        cerr << "Failed to open the file!" << endl;

*return*;

    }

    string data;

    cout << "Enter the data to write to the file: ";

    cin.ignore();

    getline(cin, data);

    file << data;

    file.close();

}

**Условие:**

Чтение данных из файла (название файла не нужно вводить с клавиатуры);

**Решение:**

void task2() {

    string filename = "data.txt";

    ifstream file(filename);

*if* (!file) {

        cerr << "Failed to open the file!" << endl;

*return*;

    }

    string data;

*while* (getline(file, data)) {

        cout << data << endl;

    }

    file.close();

}

**Условие:**

Удаление содержимого из файла;

**Решение:**

void task3() {

    string filename = "task1.txt";

    ofstream file(filename);

*if* (!file) {

        cerr << "Failed to open the file!" << endl;

*return*;

    }

    file.close();

}

**Условие:**

Удаление файла.

**Решение:**

void task4() {

    string filename = "task1.txt";

*if* (remove(filename.c\_str()) != 0) {

        cerr << "Failed to delete the file!" << endl;

    }

*else* {

        cout << "File successfully deleted!" << endl;

    }

}

Решение Дополнительных задач

**Условие:**

Напишите программу, которая считывает список файлов из указанной директории и выводит на экран их размеры в байтах. (3 балла)

**Решение:**

void AdditionalTask1() {

    string dir;

    cout << "Enter the directory path: ";

    cin >> dir;

*for* (const auto& entry : filesystem::directory\_iterator(dir)) {

        cout << entry.path().filename() << " : " << filesystem::file\_size(entry.path()) << " bytes" << endl;

    }

}

**Условие:**

Напишите программу, которая открывает двоичный файл, ищет в нем все вхождения некоторой последовательности байтов и заменяет их на другую последовательность байтов. (3 балла)

**Решение:**

void AdditionalTask2() {

    string filename = "task1.bin";

    vector<char> search = { 'a', 'b', 'c' }; *// Что ищем*

    vector<char> replace = { 'x', 'y', 'z' }; *// На что заменяем*

*// 1. Открытие файла для чтения*

    ifstream file(filename);

*if* (!file) {

        cerr << "Failed to open the file!" << endl;

*return*;

    }

*// 2. Чтение всего содержимого файла в вектор*

    vector<char> content((istreambuf\_iterator<char>(file)), istreambuf\_iterator<char>());

    file.close();

*// 3. Поиск и замена всех вхождений*

    size\_t pos = 0;

*while* (true) {

*// Поиск подпоследовательности начиная с позиции pos*

        auto it = search(content.begin() + pos, content.end(), search.begin(), search.end());

*// Если не нашли - выходим из цикла*

*if* (it == content.end()) *break*;

*// Вычисляем позицию найденной последовательности*

        pos = it - content.begin();

*// Заменяем найденную последовательность*

        copy(replace.begin(), replace.end(), content.begin() + pos);

*// Перемещаем позицию поиска вперед на длину замененной последовательности*

        pos += replace.size();

    }

*// 4. Перезапись файла с измененным содержимым*

    ofstream outFile(filename);

    outFile.write(content.data(), content.size());

    outFile.close();

}

**Условие:**

Напишите программу, которая создает двоичный файл, записывает в него некоторые данные (например, числа) и затем читает эти данные из файла. (3 балла)

**Решение:**

void AdditionalTask3() {

    string filename = "task1.bin";

*// 1. Запись в файл*

    ofstream file(filename);

*if* (!file) {

        cerr << "Failed to open the file!" << endl;

*return*;

    }

    int number = 42; *// Данные для записи*

*// Преобразуем число в последовательность байтов и записываем*

    file.write(reinterpret\_cast<char\*>(&number), sizeof(number));

    file.close();

*// 2. Чтение из файла*

    ifstream inFile(filename);

*if* (!inFile) {

        cerr << "Failed to open the file!" << endl;

*return*;

    }

    int readNumber;

*// Читаем байты из файла обратно в переменную*

    inFile.read(reinterpret\_cast<char\*>(&readNumber), sizeof(readNumber));

    cout << "Data read from the binary file: " << readNumber << endl;

    inFile.close();

}