РОССИСКИЙ ЭКОНОМИЧЕСКИЙ УНИВЕРСИТЕТ

ИМ. Г.В. ПЛЕХАНОВА

ФАКУЛЬТЕТ МАТЕМАТИЧЕСКОЙ ЭКОНОМИКИ, СТАТИСТИКИ И ИНФОРМАТИКИ

Кафедра Автоматизированных систем

обработки информации и управления

Дисциплина: Операционные системы

**Курсовая работа по теме:**

“Организация контроля доступа к файлам”

Работу выполнил студент группы ДКТ-142Б

Харченко Антон Владимирович

Подпись\_\_\_\_\_\_\_\_\_\_\_\_\_

Работу принял Дробин С.В.

Дата\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись\_\_\_\_\_\_\_\_\_\_\_\_\_

Оглавление

[Теоретическая часть. 3](#_Toc452389837)

[Цель работы. 3](#_Toc452389838)

[Введение 3](#_Toc452389839)

[Механизм контроля доступа. 6](#_Toc452389840)

[Практическая часть. 10](#_Toc452389841)

[Этап первый. Перечень доступных действий для пользователей. 10](#_Toc452389842)

[Этап второй. Объявление переменных и создание пользователя admin. 10](#_Toc452389843)

[Этап третий. Создание функций чтения и записи файла. 12](#_Toc452389844)

[Этап четвертый. Работа под пользователем admin. 13](#_Toc452389845)

[Этап пятый. Работа пользователей. 15](#_Toc452389846)

[Полный код программы 16](#_Toc452389847)

[Список используемой литературы 21](#_Toc452389848)

# Теоретическая часть.

# Цель работы.

**Целью** работы является – контролирование доступа к файлам, путем разработки элемента программы, который отвечает за создание файлов и контролирует их в процессе работы.

В качестве **задач** можно выделить:

1. Разработка алгоритма решения поставленной цели
2. Разработка элемента программного кода, осуществляющего наиважнейшие функции контроля доступа к файлам
3. Разработка средства для отладки или средства для вывода этапов в консоль

В наши дни очень важно ограничить доступ к каким-либо ресурсам на компьютере, потому что не всегда требуется дать полный доступ к управлению и использованию личного компьютера. В связи с этим, мы ищем программы, которые смогли бы ограничить доступ к файлам/папкам и другим ресурсам, но большинство из них платные или не могут обеспечить нужный функционал. Проблема очень актуальна в наши дни, но я пошел по другому пути, который заключается в создании определенных файлов в процессе выполнения программы, что позволит полностью контролировать их.

**Методика** контролирования будет осуществляться с помощью наиважнейших функций, созданные мной для работы с программой как на уровне пользователя, так и администратора программы: создание файла, изменение файла, чтение файла. С помощь этих функций можно осуществить полный контроль над работой разных пользователей с файлами, которые будут созданы.

Введение**.**

Файлы — это частный, хотя и самый популярный, вид разделяемых ресурсов, доступ к которым операционная система должна контролировать. Существуют и другие виды ресурсов, с которыми пользователи работают в режиме совместного использования. Прежде всего это различные внешние устройства: принтеры, модемы, графопостроители и т. п. Область памяти, используемая для обмена данными между процессами, также является примером разделяемого ресурса. Да и сами процессы в некоторых случаях выступают в этой роли, например, когда пользователи ОС посылают процессам сигналы, на которые процесс должен реагировать.

Во всех этих случаях действует общая схема: пользователи пытаются выполнить с разделяемым ресурсом определенные операции, а ОС должна решать, имеют ли пользователи на это право. Пользователи являются субъектами доступа, а разделяемые ресурсы — объектами. Пользователь осуществляет доступ к объектам операционной системы не непосредственно, а с помощью прикладных процессов, которые запускаются от его имени. Для каждого типа объектов существует набор операций, которые с ними можно выполнять. Например, для файлов это операции чтения, записи, удаления, выполнения; для принтера - перезапуск, очистка очереди документов, приостановка печати документа и т. д. Система контроля доступа ОС должна предоставлять средства для задания прав пользователей по отношению к объектам дифференцированно по операциям, например, пользователю может быть разрешена операция чтения и выполнения файла, а операция удаления — запрещена.

Во многих операционных системах реализованы механизмы, которые позволяют управлять доступом к объектам различного типа с единых позиций. Так, представление устройств ввода-вывода в виде специальных файлов в операционных системах UNIX является примером такого подхода: в этом случае при доступе к устройствам используются те же атрибуты безопасности и алгоритмы, что и при доступе к обычным файлам и каталогам. Еще дальше продвинулась в этом на­правлении операционная система Windows NT. В ней используется унифицированная структура — объект безопасности, — которая создается не только для файлов и внешних устройств, но и для любых разделяемых ресурсов: секций памяти, синхронизирующих примитивов типа семафоров и мьютексов и т. п. Это позволяет использовать в Windows NT для контроля доступа к ресурсам любого вида общий модуль ядра — менеджер безопасности.

В качестве субъектов доступа могут выступать как отдельные пользователи, так и группы пользователей. Определение индивидуальных прав доступа для каждого пользователя позволяет максимально гибко задать политику расходования разделяемых ресурсов в вычислительной системе. Однако этот способ приводит в больших системах к чрезмерной загрузке администратора рутинной работой по повторению одних и тех же операций для пользователей с одинаковыми правами. Объединение таких пользователей в группу и задание прав доступа в целом

для группы является одним из основных приемов администрирования в больших системах.

У каждого объекта доступа существует владелец. Владельцем может быть как отдельный пользователь, так и группа пользователей. Владелец объекта имеет право выполнять с ним любые допустимые для данного объекта операции. Во многих операционных системах существует особый пользователь (superuser, root, administrator), который имеет все права по отношению к любым объектам системы, не обязательно являясь их владельцем. Под таким именем работает администратор системы, которому необходим полный доступ ко всем файлам и устройствам для управления политикой доступа.

Различают два основных подхода к определению прав доступа.

Избирательный доступ имеет место, когда для каждого объекта сам владелец может определить допустимые операции с объектами. Этот подход называется также произвольным (от discretionary — предоставленный на собственное усмотрение) доступом, так как позволяет администратору и владельцам объектов определить права доступа произвольным образом, по их желанию. Между пользователями и группами пользователей в системах с избирательным доступом нет жестких иерархических взаимоотношений, то есть взаимоотношений, которые определены по умолчанию и которые нельзя изменить. Исключение делается только для администратора, по умолчанию наделяемого всеми правами.

Мандатный доступ (от mandatory — обязательный, принудительный) — это такой подход к определению прав доступа, при котором система наделяет пользователя определенными правами по отношению к каждому разделяемому ресурсу (в данном случае файлу) в зависимости от того, к какой группе пользователь отнесен. От имени системы выступает администратор, а владельцы объектов лишены возможности управлять доступом к ним по своему усмотрению. Все группы пользователей в такой системе образуют строгую иерархию, причем каждая группа пользуется всеми правами группы более низкого уровня иерархии, к которым добавляются права данного уровня. Членам какой-либо группы не разрешается предоставлять свои права членам групп более низких уровней иерархии. Мандатный способ доступа близок к схемам, применяемым для доступа к секретным документам: пользователь может входить в одну из групп, отличающихся правом на доступ к документам с соответствующим грифом секретности, например «для служебного пользования», «секретно», «совершенно секретно» и «государственная тайна». При этом пользователи группы «совершенно секретно» имеют право работать с документами «секретно» и «для служебного пользования», так как эти виды доступа разрешены для более низких в иерархии групп. Однако сами пользователи не распоряжаются правами доступа — этой возможностью наделен только особый чиновник учреждения.

Мандатные системы доступа считаются более надежными, но менее гибкими, обычно они применяются в специализированных вычислительных системах с повышенными требованиями к защите информации. В универсальных системах используются, как правило, избирательные методы доступа, о которых и будет идти речь ниже.

Для определенности будем далее рассматривать механизмы контроля доступа к таким объектам, как файлы и каталоги, но необходимо понимать, что эти же механизмы могут использоваться в современных операционных системах для контроля доступа к объектам любого типа и отличия заключаются лишь в наборе операций, характерных для того или иного класса объектов.

## Механизм контроля доступа.

Каждый пользователь и каждая группа пользователей обычно имеют символьное имя, а также уникальный числовой идентификатор. При выполнении процедуры логического входа в систему пользователь сообщает свое символьное имя и пароль, а операционная система определяет соответствующие числовые идентификаторы пользователя и групп, в которые он входит. Вся идентификационные данные, в том числе имена и идентификаторы пользователей и групп, пароли пользователей, а также сведения о вхождении пользователя в группы хранятся в специальном файле (файл /etc/passwd в UNIX) или специальной [базе данных](http://pandia.ru/text/category/bazi_dannih/) (в Windows NT).

Вход пользователя в систему порождает процесс-оболочку, который поддерживает диалог с пользователем и запускает для него другие процессы. Процесс-оболочка получает от пользователя символьное имя и пароль и находит по ним числовые идентификаторы пользователя и его групп. Эти идентификаторы связываются с каждым процессом, запущенным оболочкой для данного пользователя. Говорят, что процесс выступает от имени данного пользователя и данных групп пользователей. В наиболее типичном случае любой порождаемый процесс наследует идентификаторы пользователя и групп от процесса родителя.

Определить права доступа к ресурсу — значит определить для каждого пользователя набор операций, которые ему разрешено применять к данному ресурсу. В разных операционных системах для одних и тех же типов ресурсов может быть определен свой список дифференцируемых операций доступа. Для файловых объектов этот список может включать следующие операции:

* создание файла;
* уничтожение файла;
* открытие файла;
* закрытие файла;
* чтение файла;
* запись в файл;
* дополнение файла;
* поиск в файле;
* получение атрибутов файла;
* установка новых значений атрибутов;
* переименование;
* выполнение файла;
* чтение каталога;
* смена владельца;
* изменение прав доступа.

Набор файловых операций ОС может состоять из большого количества элементарных операций, а может включать всего несколько укрупненных операций. Приведенный выше список является примером первого подхода, который позволяет весьма тонко управлять правами доступа пользователей, но создает значительную нагрузку на администратора. Пример укрупненного подхода демонстрируют операционные системы семейства UNIX, в которых существуют всего три операции с файлами и каталогами: читать (read, r), писать (write, w) и выполнить (execute, x). Хотя в UNIX для операций используется всего три названия, в действительности им соответствует гораздо больше операций. Например, содержание операции выполнить зависит от того, к какому объекту она применяется. Если операция выполнить файл интуитивно понятна, то операция выполнить каталог интерпретируется как поиск в каталоге определенной записи. Поэтому администратор UNIX, по сути, располагает большим списком операций, чем это кажется на первый взгляд.

В ОС Windows NT разработчики применили гибкий подход — они реализовали возможность работы с операциями над файлами на двух уровнях: по умолчанию администратор работает на укрупненном уровне (уровень стандартных операций), а при желании может перейти на элементарный уровень (уровень индивидуальных операций).

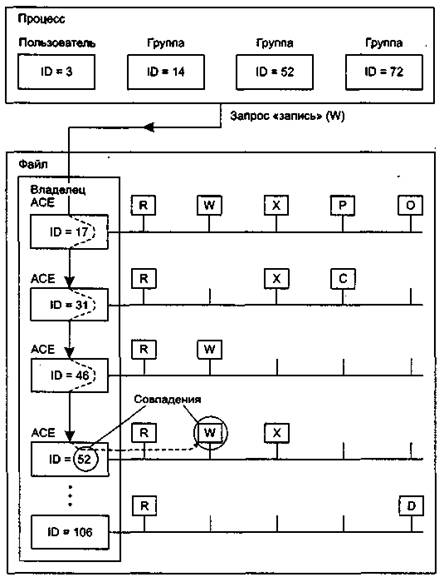
В самом общем случае права доступа могут быть описаны матрицей прав доступа, в которой столбцы соответствуют всем файлам системы, строки — всем пользователям, а на пересечении строк и столбцов указываются разрешенные операции (рис. 7.28).

**

Рис. 7.28. Матрица прав доступа

Практически во всех операционных системах матрица прав доступа хранится «по частям», то есть для каждого файла или каталога создается так называемый список управления доступом (Access Control List, ACL), в котором описываются права на выполнение операций пользователей и групп пользователей по отношению к этому файлу или каталогу. Список управления доступа является частые характеристик файла или каталога и хранится на диске в соответствующей области, например в индексном дескрипторе inode файловой системы ufs. He все файловые системы поддерживают списки управления доступом, например, его не поддерживает файловая система FAT, так как она разрабатывалась для однопользовательской однопрограммной операционной системы MS-DOS, для которой задача защиты от несанкционированного доступа не актуальна.

Обобщенно формат списка управления доступом можно представить в виде набора идентификаторов пользователей и групп пользователей, в котором для каждого идентификатора указывается набор разрешенных операций над объектом (рис. 7.29). Говорят, что список ACL состоит из элементов управления доступом (Access Control Element, АСЕ), при этом каждый элемент соответствует одному идентификатору. Список ACL с добавленным к нему идентификатором владель­ца называют характеристиками безопасности.

**

**Рис. 7.29.** Проверка прав доступа

В приведенном на рисунке примере процесс, который выступает от имени пользователя с идентификатором 3 и групп с идентификаторами 14, 52 и 72, пытается выполнить операцию записи (W) в файл. Файлом владеет пользователь с идентификатором 17. Операционная система, получив запрос на запись, находит характеристики безопасности файла (на диске или в [буферной](http://pandia.ru/text/category/bufer/) системной области) и последовательно сравнивает все идентификаторы процесса с идентификатором владельца файла и идентификаторами пользователей и групп в элементах АСЕ. В данном примере один из идентификаторов группы, от имени которой выступает процесс, а именно 52, совпадает с идентификатором одного из элементов АСЕ. Так как пользователю с идентификатором 52 разрешена операция чтения (признак W имеется в наборе операций этого элемента), то ОС разрешает процессу выполнение операции.

Описанная обобщенная схема хранения информации о правах доступа и процедуры проверки имеет в каждой операционной системе свои особенности, которые рассматриваются далее на примере операционных систем UNIX и Windows NT.

# Практическая часть.

## Этап первый. Перечень доступных действий для пользователей.

В моей работе мы можем создавать неограниченное количество пользователей.

Возможные действия, которые мы присваиваем пользователям, приведены ниже:

* чтение;
* запись;
* чтение и запись.

## Этап второй. Объявление переменных и создание пользователя admin.

Сначала мы объявляем все переменные и создадим пользователя admin,который будет обладать всеми правами, такими как: запись, чтение, создание пользователей и добавление прав, созданным пользователям.

int main() //главная функция

{

setlocale(LC\_ALL, "Rus"); // локализация русского языка

system("color 2");

int s = 1; //объявление переменных

char input\_log[21]; // вводимый пароль

string admin = "admin"; // логин администратора

char str\_array[20][6]; // массив, где будут храниться логины пользователей

int otv = 1, n = -1, v = 0;

Мы создали переменные и далее вводим имя первого пользователя.

cout << "Введите логин: ";

cin >> input\_log; // присвоение переменной вводимого значения

Далее мы проверяем, соответствует ли введенный логин пользователя логину администратора, если нет, то программа проверяет соответствует ли введенный логин логинам пользователей, которые создал admin.

if (input\_log == admin)

{

adminpass();

…

}

else

{

int x=0,j;

while (x<21)

{

if (strcmp(str\_array[x], input\_log) == 0)

{

…

}

else

{

x++;

}

}

goto u;

}

Данная строка проверяет правильность вводимого пароля admina, и вызывает функцию, которая в свою очередь еще вызывает функцию.

adminpass();

Функции:

bool passwordadmin\_is\_valid(string password) //функция проверки пароля администратора

{

string valid\_pass = "admin"; //правильный пароль

if (valid\_pass == password) //сравнение с правильным паролем

return true;

else

return false;

}

void adminpass() // функция ввода пароля администратора и его проверки

{

string user\_pass; // объявление переменной

cout << "Введите пароль: ";

cin >> user\_pass; // присвоение переменной вводимого значения

system("cls"); // очистка экрана

if (passwordadmin\_is\_valid(user\_pass)) // проверка пароля

{

cout << "Доступ разрешен." << endl;

}

else

{

cout << "Неверный пароль!" << endl;

adminpass();

}

}

## Этап третий. Создание функций чтения и записи файла.

Создаем функции записи и чтения файла, которые позже мы будем вызывать у каждого пользователя.

void reed() // функция чтения из файла

{

char buff[50];//буфер промежуточного хранения считываемого из файла текста

ifstream fin("C:\\Курсовая\\test1.txt");//открыли файл для чтения

fin.getline(buff, 50); // считали строку из файла

fin.close(); // закрываем файл

cout << buff << endl; // напечатали эту строку

}

void write() // функция записи в файл

{

char a[255]; //размер вводимой строки

cout << "Введите строку:" << endl; //вывод на экран надписи

cin.getline(a, sizeof(a)); //присвоение "а" введенной строки

ofstream out("C:\\Курсовая\\test1.txt");//открыли файл для записи

out << a << "\n"; //ввод строки

out.close(); //закрываем файл

memset(a, NULL, 255); //обнуляем переменную

}

Так же мы указываем здесь какой файл мы будем использовать, если он существует, а если нет такого файла он создастся.

Для записи: ofstream out("C:\\Курсовая\\test1.txt")

Для чтения: ifstream fin("C:\\Курсовая\\test1.txt")

## Этап четвертый. Работа под пользователем admin.

Разберем все права, которыми обладает admin.

Первое что мы рассмотрим это создание новых пользователей.

if (input\_log == admin)

{

adminpass(); //вызов функции

while (s != 0)

{

p:

cout << "1-Создание нового пользователя." << endl << "2-Добавление прав." << endl << "3-Запись." << endl << "4-Чтение." << endl << "5-Выход." << endl;

scanf("%d", &s);

switch (s)

{

case 1:

{

n++;

cout << "Введите имя нового пользователя: ";

cin >> (str\_array[n]); // ввод в n-ую строку массива

Когда мы выбрали в меню «Создание нового пользователя»,программа предлагает ввести имя нового пользователя в n-ую строку массива str-array.

while (otv != 0)

{

cout << "Создать еще одного пользователя?" << endl << "1.Да." << endl << "2.Нет, вывести список пользователей." << endl << "3.Вернуться назад." << endl;

scanf("%d", &otv);

system("cls");

switch (otv)

{

case 1:

n++;

cout << "Введите имя нового пользователя: ";

cin >> (str\_array[n]);

system("cls");

break;

case 2:

for (int i = 0; i <= n; i++)

{

cout << str\_array[i] << endl;

}

break;

case 3:

goto p;

break;

}

}

break;

}

После этого программа предлагает создать еще одного пользователя, или вывести весь список пользователей, или вернуться назад в меню.

Вернувшись назад в главное меню мы можем выбрать действие присвоения прав пользователям.

case 2:

{

cout << "Выберите пользователя: " << endl;

for (int i = 0; i <= n; i++)

{

cout << i + 1 << "- " << str\_array[i] << endl;

}

cin >>v;

v--;

cout << "a)разрешить запись;" << endl << "b)разрешить чтение;" << endl << "c)разрешить запись и чтение;"<<endl;

cin >> str\_array[v][21]; // ввод буквы в строку v на место 21 элемента строки

break;

}

Выбрав пользователя и права, которые присваиваются ему, он записывает букву, в 21 элемент строки, выбранного пользователя, с помощью которой мы позже будет определять какие права ему доступны.

Далее идут команды записи и чтения для admina, и выход в меню, где мы вводим имя пользователя. Это происходит с помощью команды goto.

case 3:

{

write();

system("cls");

break;

}

case 4:

{

system("cls");

read();

break;

}

case 5:

{

system("cls");

goto u;

break;

}

}

}

}

## Этап пятый. Работа пользователей.

Далее, введя имя пользователя, программа определяет доступные права пользователя по букве, которая записана в в 21 элементе строки пользователя.

else

{

int x=0,j;

while (x<21)

{

if (strcmp(str\_array[x], input\_log) == 0)

{

if (str\_array[x][21] == 'a' )

{

system("cls");

cout << "1-Запись." << endl;

cin >> j;

if (j == 1)

{

write();

}

}

if (str\_array[x][21] == 'b')

{

system("cls");

cout << "1-Чтение." << endl;

cin >> j;

if (j == 1)

{

read();

}

}

if (str\_array[x][21] == 'c')

{

system("cls");

cout << "1-Запись." << endl << "2-Чтение." << endl;

cin >> j;

if (j == 1)

{

write();

}

if (j == 2)

{

read();

}

}

x++;

}

else

{

x++;

}

}

goto u;

}

И далее программа завершается:

system("pause");

return 0;

}

## Полный код программы

#include <iostream> // подключаем заголовочный файл с классами, функциями и переменными для организации ввода-вывода

#include <fstream> // набор классов, методов и функций, которые предоставляют интерфейс для чтения/записи данных из/в файл

#include <string> // для организации работы со строками

using namespace std; // пространство имен

void read()// функция чтения из файла

{

char buff[50]; // буфер промежуточного хранения считываемого из файла текста

ifstream fin("C:\\Курсовая\\test1.txt"); // открыли файл для чтения

fin.getline(buff, 50); // считали строку из файла

fin.close(); // закрываем файл

cout << buff << endl; // напечатали эту строку

}

void write() // функция записи в файл

{

char a[255];//

cin.ignore();

cout << "Введите строку:" << endl; //вывод на экран надписи

cin.getline(a, sizeof(a));//присвоение "а" введенной строки

ofstream out("C:\\Курсовая\\test1.txt"); //открыли файл для записи

out << a << "\n"; //ввод строки

out.close(); //закрываем файл

memset(a, NULL, 255); //обнуляем переменную

}

bool passwordadmin\_is\_valid(string password) //функция проверки пароля для пользователя 1

{

string valid\_pass = "admin"; //правильный пароль

if (valid\_pass == password) //сравнение с правильным паролем

return true;

else

return false;

}

void adminpass() // функция ввода пароля 1 и его проверки

{

string user\_pass; // объявление переменной

cout << "Введите пароль: ";

cin >> user\_pass; // присвоение переменной вводимого значения

system("cls"); // очистка экрана

if (passwordadmin\_is\_valid(user\_pass)) // проверка пароля

{

cout << "Доступ разрешен." << endl;

}

else

{

cout << "Неверный пароль!" << endl;

adminpass();

}

}

int main() //главная функция

{

setlocale(LC\_ALL, "Rus"); // локализация русского языка

system("color 2");

int s = 1; //объявление переменных

char input\_log[21]; // вводимый пароль

string admin = "admin"; // логин администратора

char str\_array[20][6]; // массив,где будут храниться логины пользователей

int otv = 1, n = -1, v = 0;

u:

cout << "Введите логин: ";

cin >> input\_log; // присвоение переменной вводимого значения

if (input\_log == admin)

{

adminpass(); //вызов функции

while (s != 0)

{

p:

cout << "1-Создание нового пользователя." << endl << "2-Добавление прав." << endl << "3-Запись." << endl << "4-Чтение." << endl << "5-Выход." << endl;

scanf("%d", &s);

switch (s)

{

case 1:

{

n++;

cout << "Введите имя нового пользователя: ";

cin >> (str\_array[n]); // ввод в n-ую строку массива

while (otv != 0)

{

cout << "Создать еще одного пользователя?" << endl << "1.Да." << endl << "2.Нет, вывести список пользователей." << endl << "3.Вернуться назад." << endl;

scanf("%d", &otv);

system("cls");

switch (otv)

{

case 1:

n++;

cout << "Введите имя нового пользователя: ";

cin >> (str\_array[n]);

system("cls");

break;

case 2:

for (int i = 0; i <= n; i++)

{

cout << str\_array[i] << endl;

}

break;

case 3:

goto p;

break;

}

}

break;

}

case 2:

{

cout << "Выберите пользователя: " << endl;

for (int i = 0; i <= n; i++)

{

cout << i + 1 << "- " << str\_array[i] << endl;

}

cin >>v;

v--;

cout << "a)разрешить запись;" << endl << "b)разрешить чтение;" << endl << "c)разрешить запись и чтение;"<<endl;

cin >> str\_array[v][21]; // ввод буквы в строку v на место 21 элемента строки

break;

}

case 3:

{

write();

system("cls");

break;

}

case 4:

{

system("cls");

read();

break;

}

case 5:

{

system("cls");

goto u;

break;

}

}

}

}

else

{

int x=0,j;

while (x<21)

{

if (strcmp(str\_array[x], input\_log) == 0)

{

if (str\_array[x][21] == 'a' )

{

system("cls");

cout << "1-Запись." << endl;

cin >> j;

if (j == 1)

{

write();

}

}

if (str\_array[x][21] == 'b')

{

system("cls");

cout << "1-Чтение." << endl;

cin >> j;

if (j == 1)

{

read();

}

}

if (str\_array[x][21] == 'c')

{

system("cls");

cout << "1-Запись." << endl << "2-Чтение." << endl;

cin >> j;

if (j == 1)

{

write();

}

if (j == 2)

{

read();

}

}

x++;

}

else

{

x++;

}

}

goto u;

}

system("pause");

return 0;

}

# Список используемой литературы

1. <http://polpoz.ru/umot/lekciya-7-fajlovie-sistemi/>
2. <http://pandia.ru/text/78/337/1288.php>
3. Особенности объектно-ориентированного программирования на C++/CLI, C# и Java, Медведев В. И.
4. Основные концепции и механизмы объектно-ориентированного программирования, Пышкин Е.В.
5. C++. Объектно-ориентированное программирование, Лаптев В.В.