**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №8

по дисциплине: «Операционные системы»

на тему: **«**Управление файловой системой»

Выполнил: студент гр. ИТИ-12

Ёвженко Ю.Д.

Принял: преподаватель-стажёр

Дашкевич Д.А.

Гомель 2022

**Цель работы**: доработать модель файловой системы.

**Задание:**

Дополнить программу, разработанную при выполнении лабораторной работы №7 вспомогательной утилитой, согласно варианта (см. табл. 1).

Таблица 1 – Вариант выполняемого задания

|  |  |
| --- | --- |
| Вариант | Дополнительная утилита |
| 4 | Журнализация |

**Ход работы**

На рисунке 1 показано меню программы.

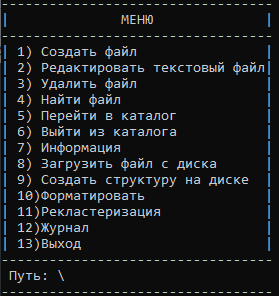


Рисунок 1 – Меню программы

На рисунке 2 показаны записи в журнале после работы с виртуальным файлом.

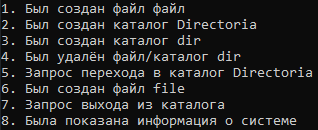


Рисунок 2 – Журнал

Журнализация осуществляет запись действий пользователя.

**Вывод:** в ходе выполнения лабораторной работы была доработана виртуальная файловая система, осуществляющая работу с текстовыми файлами и каталогами.

**ПРИЛОЖЕНИЕ А**

(обязательное)

**Текст программы**

**filesys.cpp**:

#include <stdio.h>

#include <stdlib.h>

#include <windows.h>

#include <string.h>

#include <math.h>

#include <direct.h>

struct memBlock

{

int next;

bool isFree;

bool isDir;

bool isStart;

char \*data;

char name[16];

char path[256];

};

char Jinfo[1000][100];

char Jname[1000][100];

int Jind = 0;

int j = 0;

char bufRus[256];

char\* Rus(const char\* text)

{

CharToOem(text, bufRus);

return bufRus;

}

int scanInt(const char\* dispStr)

{

int bufInt, check;

do

{

printf(Rus(dispStr));

fflush(stdin);

check = scanf("%d", &bufInt);

if ( check != 1 )

{

printf(Rus("Ошибка. Введите целое число\n"));

}

}

while ( check != 1 );

return bufInt;

}

memBlock\* freeMem(memBlock \*memory, int total\_blocks)

{

for (int i = 0; i < total\_blocks; i++)

{

delete[] memory[i].data;

}

delete[] memory;

return memory;

}

int menu(char \*path)

{

int key;

system("cls");

printf("----------------------------------\n");

printf(Rus("| МЕНЮ |\n"));

printf("----------------------------------\n");

printf(Rus("| 1) Создать файл |\n"));

printf(Rus("| 2) Редактировать текстовый файл|\n"));

printf(Rus("| 3) Удалить файл |\n"));

printf(Rus("| 4) Найти файл |\n"));

printf(Rus("| 5) Перейти в каталог |\n"));

printf(Rus("| 6) Выйти из каталога |\n"));

printf(Rus("| 7) Информация |\n"));

printf(Rus("| 8) Загрузить файл с диска |\n"));

printf(Rus("| 9) Создать структуру на диске |\n"));

printf(Rus("| 10)Форматировать |\n"));

printf(Rus("| 11)Рекластeризация |\n"));

printf(Rus("| 12)Журнал |\n"));

printf(Rus("| 13)Выход |\n"));

printf("----------------------------------\n");

printf(Rus(" Путь: %s \n"),path);

printf("----------------------------------\n");

fflush(stdin);

scanf("%d", &key);

return key;

}

memBlock\* format(memBlock \*memory, int \*total\_mem, int \*total\_blocks, int \*block\_size, int \*free\_size, char\* path)

{

int i, j;

bool flag;

system("cls");

do

{

flag = false;

\*total\_mem = scanInt("Введите количество памяти: ");

\*total\_blocks = scanInt("Введите количество блоков: ");

if ( \*total\_mem < 1 || \*total\_blocks < 1 || \*total\_mem % \*total\_blocks != 0 )

{

printf(Rus("Некорректное количество памяти/блоков.\n"));

flag = true;

}

}

while ( flag );

path[0] = '\\';

for (i = 1; i < 256; i++)

{

path[i] = '\0';

}

\*block\_size = \*total\_mem / \*total\_blocks;

\*free\_size = \*block\_size \* \*total\_blocks;

memory = new memBlock[\*total\_blocks];

for (i = 0; i < \*total\_blocks; i++)

{

memory[i].data = new char[\*block\_size];

memory[i].isFree = true;

memory[i].next = -2;

memory[i].isDir = false;

memory[i].isStart = false;

for (j = 0; j < 16; j++)

{

memory[i].name[j] = '\0';

}

for (j = 0; j < 256; j++)

{

memory[i].path[j] = '\0';

}

}

return memory;

}

memBlock\* reclustorization(memBlock \*memory, int \*total\_mem, int \*total\_blocks, int \*block\_size, int \*free\_size, char\* path)

{

memBlock \*new\_memory;

int i, j, k, l, total\_files, new\_total\_blocks, new\_block\_size, new\_free\_size, index, new\_index, buf, max;

bool flag;

system("cls");

strcpy(Jinfo[Jind], Rus("Была проведена рекластеризация"));

Jind++;

do

{

flag = false;

new\_total\_blocks = scanInt("Введите новое количество блоков: ");

if ( new\_total\_blocks < 1 || \*total\_mem % new\_total\_blocks != 0 )

{

printf(Rus("Некорректное количество блоков.\n"));

flag = true;

}

}

while ( flag );

if ( new\_total\_blocks < \*total\_blocks )

{

for (i = 0, total\_files = 0, max = 0; i < \*total\_blocks; i++)

{

if ( memory[i].isStart )

{

total\_files++;

index = i;

buf = 1;

while ( memory[index].next >= 0 )

{

index = memory[index].next;

buf++;

}

if ( buf > max )

{

max = buf;

}

}

}

if ( max >= floor( (float)\*total\_blocks / (float)total\_files ) )

{

printf(Rus("С данным количеством блоков рекластeризацию без потери данных провести невозможно.\n"));

system("pause");

return memory;

}

}

path[0] = '\\';

for (i = 1; i < 256; i++)

{

path[i] = '\0';

}

new\_block\_size = \*total\_mem / new\_total\_blocks;

new\_free\_size = new\_block\_size \* new\_total\_blocks;

new\_memory = new memBlock[new\_total\_blocks];

for (i = 0; i < new\_total\_blocks; i++)

{

new\_memory[i].data = new char[\*block\_size];

new\_memory[i].isFree = true;

new\_memory[i].next = -2;

new\_memory[i].isDir = false;

new\_memory[i].isStart = false;

for (j = 0; j < 16; j++)

{

new\_memory[i].name[j] = '\0';

}

for (j = 0; j < 256; j++)

{

new\_memory[i].path[j] = '\0';

}

}

for (i = 0; i < \*total\_blocks; i++)

{

if ( !memory[i].isStart )

{

continue;

}

if ( memory[i].isDir )

{

for (j = 0; j < new\_total\_blocks; j++)

{

if ( new\_memory[j].isFree )

{

break;

}

}

index = j;

new\_memory[index].isFree = false;

new\_memory[index].next = -1;

new\_memory[index].isDir = true;

new\_memory[index].isStart = true;

for (j = 0; j < 16; j++)

{

new\_memory[index].name[j] = memory[i].name[j];

}

for (j = 0; j < 256; j++)

{

new\_memory[index].path[j] = memory[i].path[j];

}

continue;

}

for (new\_index = 0; new\_index < new\_total\_blocks; new\_index++)

{

if ( new\_memory[new\_index].isFree )

{

break;

}

}

new\_memory[new\_index].isFree = false;

new\_memory[new\_index].next = -1;

new\_memory[new\_index].isDir = false;

new\_memory[new\_index].isStart = true;

for (j = 0; j < 16; j++)

{

new\_memory[new\_index].name[j] = memory[i].name[j];

}

for (j = 0; j < 256; j++)

{

new\_memory[new\_index].path[j] = memory[i].path[j];

}

index = i;

k = 0;

while ( index >= 0 )

{

for (j = 0; j < \*block\_size; j++)

{

new\_memory[new\_index].data[k] = memory[index].data[j];

if ( memory[index].data[j] == '\0' )

{

break;

}

k++;

if ( k >= new\_block\_size )

{

k = 0;

buf = new\_index;

for (new\_index = 0; new\_index < new\_total\_blocks; new\_index++)

{

if ( new\_memory[new\_index].isFree )

{

break;

}

}

new\_memory[new\_index].isFree = false;

new\_memory[new\_index].next = -1;

new\_memory[new\_index].isDir = false;

new\_memory[new\_index].isStart = false;

for (l = 0; l < 16; l++)

{

new\_memory[new\_index].name[l] = memory[i].name[l];

}

for (l = 0; l < 256; l++)

{

new\_memory[new\_index].path[l] = memory[i].path[l];

}

new\_memory[buf].next = new\_index;

}

}

index = memory[index].next;

}

}

for (i = 0, new\_free\_size = 0; i < new\_total\_blocks; i++)

{

if ( new\_memory[i].isFree )

{

new\_free\_size += new\_block\_size;

}

}

memory = freeMem(memory, \*total\_blocks);

\*total\_blocks = new\_total\_blocks;

\*block\_size = new\_block\_size;

\*free\_size = new\_free\_size;

return new\_memory;

}

void delBlock(memBlock \*memory, int index, int \*free\_size, int block\_size)

{

int i;

\*free\_size = \*free\_size + block\_size;

memory[index].isDir = false;

memory[index].isStart = false;

memory[index].isFree = true;

memory[index].next = -2;

for (i = 0; i < 16; i++)

{

memory[index].name[i] = '\0';

}

for (i = 0; i < 256; i++)

{

memory[index].path[i] = '\0';

}

}

int deleteFile(memBlock \*memory, int total\_blocks, char \*path, int \*free\_size, int block\_size)

{

int i, j, index;

char name[16];

system("cls");

printf(Rus("Введите имя файла: "));

fflush(stdin);

gets(name);

for (i = 0, index = -2; i < total\_blocks && index == -2; i++)

{

if ( strcmp(memory[i].path, path) == 0 && strcmp(memory[i].name, name) == 0 && memory[i].isStart )

{

index = i;

strcpy(Jinfo[Jind],Rus("Был удалён файл/каталог"));

strcpy(Jname[Jind], name);

Jind++;

}

}

if ( index == -2 )

{

printf(Rus("Файл с таким именем в директории не найден.\n"));

strcpy(Jinfo[Jind],Rus("Попытка удаления несуществующего файла"));

strcpy(Jname[Jind],name);

Jind++;

system("pause");

return 1;

}

while ( index >= 0 )

{

i = index;

index = memory[i].next;

delBlock(memory, i, free\_size, block\_size);

}

return 0;

}

int createFile(memBlock \*memory, int total\_blocks, int block\_size, int \*free\_size, char \*path)

{

int i, index;

char check;

char name[16];

bool flag;

system("cls");

for (i = 0, flag = true; i < total\_blocks && flag; i++)

{

if ( memory[i].isFree )

{

index = i;

flag = false;

}

}

if ( flag )

{

printf(Rus("Для создания файла нет свободных блоков памяти.\n"));

system("pause");

return 1;

}

printf(Rus("Введите тип файла (d - каталог / другое - текстовый файл): "));

fflush(stdin);

scanf("%c", &check);

printf(Rus("Введите имя файла (до 16 символов): "));

fflush(stdin);

gets(name);

i = 0;

while ( name[i] != '\0' && i < 16)

{

if ( name[i] == '\\' || name[i] == '/' || name[i] == ':' || name[i] == '\*' || name[i] == '?' || name[i] == '.' || name[i] == '\"' || name[i] == '<' || name[i] == '>' || name[i] == '|' )

{

printf(Rus("В имени файла нельзя использовать символы \\, /, :, \*, ?, ., \", <, >, |.\n"));

system("pause");

return 1;

}

i++;

}

for (i = 0, flag = true; i < total\_blocks && flag; i++)

{

if ( strcmp(memory[i].path, path) == 0 && strcmp(memory[i].name, name) == 0 && memory[i].isStart )

{

flag = false;

}

}

if ( !flag )

{

printf(Rus("Файл с таким именем уже существует.\n"));

system("pause");

return 1;

}

\*free\_size = \*free\_size - block\_size;

for (i = 0; i < 16; i++)

{

memory[index].name[i] = name[i];

}

for (i = 0; i < 256; i++)

{

memory[index].path[i] = path[i];

}

for (i = 0; i < block\_size; i++)

{

memory[index].data[i] = '\0';

}

memory[index].isDir = ( check == 'd' ) ? true : false;

memory[index].next = -1;

memory[index].isFree = false;

memory[index].isStart = true;

if( memory[index].isDir ){

strcpy(Jinfo[Jind],Rus("Был создан каталог"));

strcpy(Jname[Jind], name);

Jind++;

} else {

strcpy(Jinfo[Jind],Rus("Был создан файл"));

strcpy(Jname[Jind], name);

Jind++;

}

return 0;

}

void searchFile(memBlock \*memory, int total\_blocks)

{

int i;

char name[16];

system("cls");

printf(Rus("Введите имя файла: "));

fflush(stdin);

gets(name);

strcpy(Jinfo[Jind],Rus("Поиск файла"));

strcpy(Jname[Jind], name);

Jind++;

printf(Rus("Список найденных файлов:\n"));

for (i = 0; i < total\_blocks; i++)

{

if ( memory[i].isStart && strcmp(memory[i].name, name) == 0 )

{

printf("%s%s\n", memory[i].path, memory[i].name);

}

}

system("pause");

}

int editFile(memBlock \*memory, int total\_blocks, int block\_size, int \*free\_size, char \*path)

{

int i, j, k, l, index;

char name[16];

char str[256];

system("cls");

printf(Rus("Введите имя файла: "));

fflush(stdin);

gets(name);

strcpy(Jinfo[Jind],Rus("Запрос на редактирование файла"));

strcpy(Jname[Jind], name);

Jind++;

for (i = 0, index = -2; i < total\_blocks && index == -2; i++)

{

if ( strcmp(memory[i].path, path) == 0 && strcmp(memory[i].name, name) == 0 && memory[i].isStart && !memory[i].isDir )

{

index = i;

}

}

if ( index == -2 )

{

printf(Rus("Файл с таким именем в директории не найден.\n"));

system("pause");

return 1;

}

printf(Rus("Старый текст файла:\n%s"), strncpy(str, memory[index].data, block\_size));

i = memory[index].next;

memory[index].next = -1;

while ( i >= 0 )

{

j = i;

printf("%s", strncpy(str, memory[j].data, block\_size));

i = memory[j].next;

delBlock(memory, j, free\_size, block\_size);

}

printf(Rus("\nВведите новый текст файла:\n"));

fflush(stdin);

gets(str);

l = strlen(str) + 1;

if ( l > \*free\_size + block\_size )

{

printf(Rus("Ошибка. Недостаточно свободного пространства для записи строки.\n"));

system("pause");

return 1;

}

for (i = 0, l = ceil((float) l / (float) block\_size); i < l; i++)

{

strncpy(memory[index].data, &(str[i\*block\_size]), block\_size);

for (j = 0; j < total\_blocks; j++)

{

if ( memory[j].isFree )

{

break;

}

}

if ( i + 1 == l )

{

break;

}

\*free\_size = \*free\_size - block\_size;

for (k = 0; k < 16; k++)

{

memory[j].name[k] = memory[index].name[k];

}

for (k = 0; k < 256; k++)

{

memory[j].path[k] = memory[index].path[k];

}

for (k = 0; k < block\_size; k++)

{

memory[j].data[k] = '\0';

}

memory[j].next = -1;

memory[j].isDir = false;

memory[j].isFree = false;

memory[j].isStart = false;

memory[index].next = j;

index = j;

}

return 0;

}

int toDir(memBlock \*memory, int total\_blocks, char \*path)

{

int i, index;

char name[16];

system("cls");

printf(Rus("Введите имя директории: "));

fflush(stdin);

gets(name);

strcpy(Jinfo[Jind],Rus("Запрос перехода в каталог"));

strcpy(Jname[Jind], name);

Jind++;

for (i = 0, index = -2; i < total\_blocks && index == -2; i++)

{

if ( strcmp(memory[i].path, path) == 0 && strcmp(memory[i].name, name) == 0 && memory[i].isStart && memory[i].isDir )

{

index = i;

}

}

if ( index == -2 )

{

printf(Rus("Директория с таким именем не найдена.\n"));

system("pause");

return 1;

}

strcpy(&(path[strlen(path)]), memory[index].name);

path[strlen(path)] = '\\';

return 0;

}

void exitDir(char \*path)

{

int i;

strcpy(Jinfo[Jind],Rus("Запрос выхода из каталога"));

Jind++;

i = strlen(path) - 1;

if ( i > 0 )

{

path[i--] = '\0';

while( path[i] != '\\' )

{

path[i--] = '\0';

}

}

}

void showInfo(memBlock \*memory, int total\_mem, int total\_blocks, int block\_size, int free\_size)

{

int i;

char str[256];

char buf[5];

strcpy(Jinfo[Jind],Rus("Была показана информация о системе"));

Jind++;

system("cls");

printf(Rus("Всего памяти: %d\nСвободно памяти: %d\nВсего блоков: %d\nСвободно блоков: %d\nПамяти в блоке: %d\n"), total\_mem, free\_size, total\_blocks, free\_size/block\_size, block\_size);

printf("\xC9\xC4\xC4\xC4\xC4\xC4\xCB\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xCB\xC4\xC4\xC4\xC4\xC4\xBB\n\xB3 N \xB3 ");

printf(Rus("Имя"));

printf(" \xB3");

printf(Rus(" Тип "));

printf("\xB3\n");

for (i = 0; i < total\_blocks; i++)

{

strcpy(str, memory[i].path);

strcpy(&(str[strlen(str)]), memory[i].name);

printf("\xCC\xC4\xC4\xC4\xC4\xC4\xCE\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xCE\xC4\xC4\xC4\xC4\xC4\xB9\n\xB3%5d\xB3%-25s\xB3 %c \xB3\n", i+1, str, memory[i].isFree ? ' ' : memory[i].isDir ? 'D' : 'F');

}

printf("\xC8\xC4\xC4\xC4\xC4\xC4\xCA\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xC4\xCA\xC4\xC4\xC4\xC4\xC4\xBC\n");

system("pause");

}

int createStruct(memBlock \*memory, int total\_blocks, int block\_size)

{

int i, j, index, dlen, errs;

char\*\* dirs;

char path[32], fpath[304], str[256];

FILE\* f;

system("cls");

printf(Rus("Введите путь и название директории, где будет создана структура (D:\\test): "));

fflush(stdin);

gets(path);

strcpy(Jinfo[Jind],Rus("Запрос на создание структуры по адресу:"));

strcpy(Jname[Jind], path);

Jind++;

if ( path[0] == '\0' )

{

strcpy(path, "D:\\test");

}

if ( mkdir(path) )

{

printf(Rus("Не удалось создать каталог %s\n"), path);

system("pause");

return 1;

}

for (i = 0, dlen = 0; i < total\_blocks; i++)

{

if ( memory[i].isDir && memory[i].isStart )

{

dlen++;

}

}

dirs = new char\*[dlen];

for (i = 0; i < dlen; i++)

{

dirs[i] = new char[304];

}

for (i = 0, dlen = 0; i < total\_blocks; i++)

{

if ( memory[i].isDir && memory[i].isStart )

{

strcpy(dirs[dlen], path);

strcpy(&(dirs[dlen][strlen(dirs[dlen])]), memory[i].path);

strcpy(&(dirs[dlen][strlen(dirs[dlen])]), memory[i].name);

dlen++;

}

}

do

{

for (i = 0, errs = 0; i < dlen; i++)

{

errs += mkdir(dirs[i]);

}

}

while ( dlen + errs );

for (i = 0; i < total\_blocks; i++)

{

if ( !memory[i].isDir && memory[i].isStart )

{

strcpy(fpath, path);

strcpy(&(fpath[strlen(fpath)]), memory[i].path);

strcpy(&(fpath[strlen(fpath)]), memory[i].name);

strcpy(&(fpath[strlen(fpath)]), ".txt");

f = fopen(fpath, "w");

index = i;

str[block\_size] = '\0';

while ( index >= 0 )

{

fprintf(f, "%s", strncpy(str, memory[index].data, block\_size));

index = memory[index].next;

}

fclose(f);

}

}

system("pause");

for (i = 0; i < dlen; i++)

{

delete[] dirs[i];

}

delete[] dirs;

return 0;

}

int loadFromDisc(memBlock \*memory, int total\_blocks, int block\_size, int \*free\_size, char \*path)

{

FILE\* f;

char fpath[32];

char name[16];

char \*data;

int i, j, k, fsize, index;

bool flag;

system("cls");

printf(Rus("Введите путь к текстовому файлу, который нужно загрузить (D:\\test.txt): "));

fflush(stdin);

gets(fpath);

strcpy(Jinfo[Jind],Rus("Запрос на загрузку файла с диска"));

strcpy(Jname[Jind], fpath);

Jind++;

if ( fpath[0] == '\0' )

{

strcpy(fpath, "D:\\test.txt");

}

f = fopen(fpath, "r");

if ( !f )

{

printf(Rus("Не удалось открыть файл %s.\n"), fpath);

system("pause");

return 1;

}

fseek(f, 0, SEEK\_END);

fsize = ftell(f) + 1;

fseek(f, 0, SEEK\_SET);

if ( fsize > \*free\_size )

{

printf(Rus("Файл слишком большой.\n"));

system("pause");

return 1;

}

printf(Rus("Придумайте, как будет называться файл в виртуальном файле: "));

fflush(stdin);

gets(name);

i = 0;

while ( name[i] != '\0' && i < 16 )

{

if ( name[i] == '\\' || name[i] == '/' || name[i] == ':' || name[i] == '\*' || name[i] == '?' || name[i] == '.' || name[i] == '\"' || name[i] == '<' || name[i] == '>' || name[i] == '|' )

{

printf(Rus("В имени файла нельзя использовать символы \\, /, :, \*, ?, ., \", <, >, |.\n"));

system("pause");

return 1;

}

i++;

}

for (i = 0, flag = true; i < total\_blocks && flag; i++)

{

if ( strcmp(memory[i].path, path) == 0 && strcmp(memory[i].name, name) == 0 && memory[i].isStart )

{

flag = false;

}

}

if ( !flag )

{

printf(Rus("Файл с таким именем уже существует.\n"));

system("pause");

return 1;

}

data = new char[fsize];

fread(data, sizeof(char), fsize-1, f);

data[fsize-1] = '\0';

for (index = 0; index < total\_blocks; index++)

{

if ( memory[index].isFree )

{

break;

}

}

\*free\_size = \*free\_size - block\_size;

for (k = 0; k < 16; k++)

{

memory[index].name[k] = name[k];

}

for (k = 0; k < 256; k++)

{

memory[index].path[k] = path[k];

}

for (k = 0; k < block\_size; k++)

{

memory[index].data[k] = '\0';

}

memory[index].next = -1;

memory[index].isDir = false;

memory[index].isFree = false;

memory[index].isStart = true;

for (i = 0, fsize = ceil((float) fsize / (float) block\_size); i < fsize; i++)

{

strncpy(memory[index].data, &(data[i\*block\_size]), block\_size);

for (j = 0; j < total\_blocks; j++)

{

if ( memory[j].isFree )

{

break;

}

}

if ( i + 1 == fsize )

{

break;

}

\*free\_size = \*free\_size - block\_size;

for (k = 0; k < 16; k++)

{

memory[j].name[k] = name[k];

}

for (k = 0; k < 256; k++)

{

memory[j].path[k] = path[k];

}

for (k = 0; k < block\_size; k++)

{

memory[j].data[k] = '\0';

}

memory[j].next = -1;

memory[j].isDir = false;

memory[j].isFree = false;

memory[j].isStart = false;

memory[index].next = j;

index = j;

}

fclose(f);

delete[] data;

return 0;

}

int main()

{

memBlock \*memory;

int key, total\_mem, total\_blocks, block\_size, free\_size;

char path[256];

memory = format(memory, &total\_mem, &total\_blocks, &block\_size, &free\_size, path);

do

{

key = menu(path);

switch (key)

{

case 1:

// Создать файл, запретить ввод неподходящих символов

createFile(memory, total\_blocks, block\_size, &free\_size, path);

break;

case 2:

// Редактировать текстовый файл

editFile(memory, total\_blocks, block\_size, &free\_size, path);

break;

case 3:

// Удалить файл

deleteFile(memory, total\_blocks, path, &free\_size, block\_size);

break;

case 4:

// Найти файл

searchFile(memory, total\_blocks);

break;

case 5:

// Перейти в каталог

toDir(memory, total\_blocks, path);

break;

case 6:

// Выйти из каталога

exitDir(path);

break;

case 7:

// Информация

showInfo(memory, total\_mem, total\_blocks, block\_size, free\_size);

break;

case 8:

// Загрузить файл с диска

loadFromDisc(memory, total\_blocks, block\_size, &free\_size, path);

break;

case 9:

// Создать структуру на диске

createStruct(memory, total\_blocks, block\_size);

break;

case 10:

// Форматировать

memory = freeMem(memory, total\_blocks);

memory = format(memory, &total\_mem, &total\_blocks, &block\_size, &free\_size, path);

Jind = 0;

break;

case 11:

// Рекласторизация

memory = reclustorization(memory, &total\_mem, &total\_blocks, &block\_size, &free\_size, path);

break;

case 12:

// Вывод журнала

system("cls");

for(j = 0; j < Jind; j++){

printf(Rus("%d. %s %s\n"),j+1, Jinfo[j], Jname[j]);

}

system("pause");

break;

}

}

while ( key != 13 );

freeMem(memory, total\_blocks);

return 0;

}