**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ЛАБОРАТОРНАЯ РАБОТА №11

по дисциплине: **«**[**Операционные системы**](https://www.edu.gstu.by/course/view.php?id=557)**»**

**Алгоритмы замещения страниц**

Выполнил: студент гр. ИТП-11

Горкунов А.О,

Принял: Преподаватель

Карась О.В.

Гомель 2023

**Цель работы**: разработать модель файловой системы

**Ход работы**

**Задание:** Дополнить программу, разработанную при выполнении лабораторной работы №11 вспомогательной утилитой, согласно варианта

|  |  |
| --- | --- |
| Вариант | Дополнительная утилита |
| 6 | Журнализация |

**1.** Создание файла oopfiles.cpp реализующего файловую модель с журналом действий

**2.** Тестирование работы кода

На рисунке 1 представлена файловая модель способная запоминать действия.

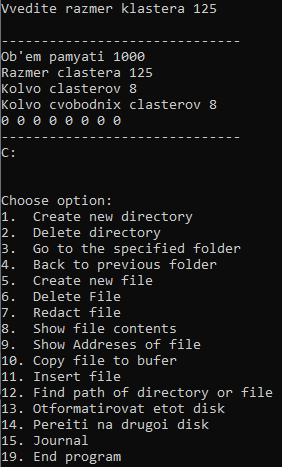


Рисунок 1 – меню файловой модели

На рисунке 2 представлено создание новой директории .

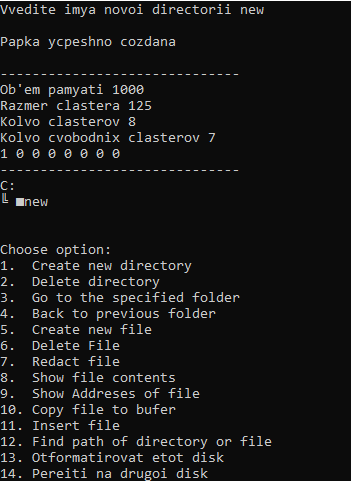


Рисунок 2 – создание и отображение новой директории

На рисунке 3 представлен журнал отображающий действие пользователя, который заполняется по мере выполнения задач поставленных пользователем.

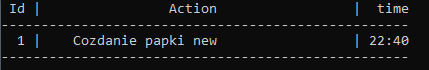


Рисунок 3 – журнал отображающий действия пользователя

**Вывод: дополнил программу утилитой журнализации.**

Листинг:

#include <iostream>

#include <string.h>

#include <vector>

#include <conio.h>

#include <iomanip>

#include <windows.h>

#include <cmath>

using namespace std;

const int FullMemorySizeDisk = 1000;

const int FullMemorySizeFlesh = 500;

class Journal

{

public:

static int id;

string fullInformation;

void formatirovanie(){

id = 0;

fullInformation = " Id | Action | time\n";

fullInformation += "---------------------------------------------------";

}

void Show(){

cout << fullInformation;

}

void AddMessage(string message){

SYSTEMTIME time;

GetLocalTime(&time);

string space(35 - message.length(), ' ');

fullInformation += "\n " + to\_string(++id) + " | " + message + space +"| "+ to\_string(time.wHour) + ":" + to\_string(time.wMinute);

fullInformation += "\n---------------------------------------------------";

}

};

int Journal::id = 0;

class ArrClust

{

public:

int ClusterSize;

int NumberOfCluster;

string bitvect;

int NumberOfFree;

ArrClust(){

}

void Formatirovanie(int clustsize, int fullmemory){

ClusterSize = clustsize;

NumberOfCluster = (int)fullmemory/ClusterSize;

NumberOfFree = NumberOfCluster;

bitvect.assign(NumberOfCluster, '0');

}

void showFullinformation(const int memory){

cout << "\n------------------------------";

cout << "\nOb'em pamyati " << memory;

cout << "\nRazmer clastera " << ClusterSize;

cout << "\nKolvo clasterov " << NumberOfCluster;

cout << "\nKolvo cvobodnix clasterov " << NumberOfFree;

cout << endl;

this->ShowBeatVect();

cout << "\n------------------------------\n";

}

int findFreeCluster(){

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

{

if(bitvect[i] == '0')

{

bitvect[i] = '1';

NumberOfFree--;

return i;

}

}

}

void ClearAddToFile(vector<int>& vect){

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

{

bitvect[vect[i]] = '0';

NumberOfFree++;

}

vect.clear();

}

void ClearOneAddres(int i){

bitvect[i] = '0';

NumberOfFree++;

}

void ShowBeatVect() const{

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

cout << bitvect[i] << " ";

}

// i dlya failov nyzhen

bool IsFree(int i){

if(NumberOfFree + i != 0)

return 1;

else

return 0;

}

};

class File

{

public:

string data;

int Size;

vector<int> AddresCluster;

string name;

File(string a): name(a), data("\0"), Size(1){

}

void Clear(){

data = "\0";

Size = 1;

AddresCluster.clear();

name = "\0";

}

void show() const{

system("cls");

cout << "Soderzhimoe faila " << name << ":\n" << data;

}

void ShowAddres() const{

system("cls");

cout << "Addres faila " << this->name << ": ";

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

cout << AddresCluster[i];

}

void RedactFile(ArrClust \*Clust){

char symbol;

int sizeOfSymbol = 1;

int count\_symbols = data.length();

do{

system("cls");

cout << "Backspace to delete symbol";

cout << "\nCtrl+z to stop";

cout << "\n` dlya perexoda na novyu stroku\n";

cout << endl << data << endl;

symbol = getche();

if(symbol == char(26))

break;

if(symbol == '`')

data += "\n";

else

{

if(symbol == '\b' && count\_symbols != 0)

{

data.erase(count\_symbols - 1);

count\_symbols--;

Size -= sizeOfSymbol;

if(Size % Clust->ClusterSize == 0)

{

Clust->ClearOneAddres(AddresCluster.back());

AddresCluster.pop\_back();

}

}

else

{

data += symbol;

count\_symbols++;

Size += sizeOfSymbol;

if(Size % Clust->ClusterSize == 0)

AddresCluster.push\_back(Clust->findFreeCluster());

}

}

}while(Clust->IsFree(1));

cout <<"\nFile ycpeshno coxranen\n";

}

};

class Directory

{

public:

vector<File> files;

vector<Directory> dir;

string name;

Directory \*parentDir;

string path;

int addres;

//dlya cozdaniya diska

Directory(string a): name(a), parentDir(nullptr), path(a){

}

//dlya papok

Directory(string a, Directory \*parent, string b, ArrClust \*Clust): name(a), parentDir(parent), path(b){

addres = Clust->findFreeCluster();

}

void showDir() const{

int i=0;

while (i<dir.size())

cout << endl << i << ". " << char(254)<< dir[i++].name;

int j=0;

while (j < files.size())

cout << endl << i++ << ". " << char(45)<<files[j++].name;

}

//Kracivo

void showDir2(int j) const{

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

{

for(int k = 0; k<j; k++)

cout << char(186) <<"\t";

if(i == dir.size() - 1 && files.size() == 0)

cout << char(200) << " " << char(254) << dir[i].name << endl;

else

cout << char(204) << " " << char(254) << dir[i].name << endl;

if(dir[i].dir.size() != 0 || dir[i].files.size() != 0)

{

dir[i].showDir2(++j);

j--;

}

}

for(int b=0;b < files.size(); b++)

{

for(int k = 0; k<j; k++)

cout << char(186) <<"\t";

if(b != files.size() - 1)

cout << char(204) << " " << char(45) << files[b].name << endl;

else

cout << char(200) << " " << char(45) << files[b].name << endl;

}

}

void findPathOfObject(string imya, string \*way){

for(int j=0; j<dir.size();j++)

{

if(dir[j].name == imya)

\*way = \*way + path + "\n";

if(dir[j].dir.size() + dir[j].files.size() > 0)

dir[j].findPathOfObject(imya, way);

}

for(int k=0; k<files.size(); k++)

if(files[k].name == imya)

\*way = \*way + path + "\n";

}

bool findSimilarName(string imya){

for(int j=0; j<dir.size();j++)

{

if(dir[j].name == imya)

return 1;

}

for(int k=0; k<files.size(); k++)

if(files[k].name == imya)

return 1;

return 0;

}

void createDir(ArrClust \*Clust, Journal \*journal){

if(Clust->IsFree(0))

{

string name1;

cout << "Vvedite imya novoi directorii ";

cin >> name1;

if(!findSimilarName(name1))

{

journal->AddMessage("Cozdanie papki " + name1);

Directory newdir(name1, this, (path + char(92) + name1), Clust);

dir.push\_back(newdir);

cout << "\nPapka ycpeshno cozdana\n";

}

else

{

cout << "V dannoi directorii uzhe est ob'ekt c takim imenem\n";

createDir(Clust, journal);

}

}

else

cout << "\nNet mecta na diske\n";

}

void createFile(ArrClust \*Clust, Journal \*journal){

if(Clust->IsFree(0))

{

string name1;

cout << "Vvedite imya faila ";

cin >> name1;

if(!findSimilarName(name1))

{

journal->AddMessage("Cozdanie faila " + name1);

File newFile(name1);

newFile.AddresCluster.push\_back(Clust->findFreeCluster());

files.push\_back(newFile);

}

else

{

cout << "V dannoi directorii uzhe est ob'ekt c takim imenem\n";

createFile(Clust, journal);

}

}

else

cout << "\nNet mecta ";

}

void showFile() const{

int n;

cout << "\nChoose file ";

cin >> n;

if((n > dir.size() + files.size() - 1) || (n < dir.size()))

cout << "\nWrong number, choose again ";

else

files[n - dir.size()].show();

}

void deleteFile(int i, ArrClust \*Clust, Journal \*journal){

int position = i - dir.size();

journal->AddMessage("Ydalenie faila " + files[position].name);

Clust->ClearAddToFile(files[position].AddresCluster);

files[position].name = '\0';

files[position].data = '\0';

files.erase(files.begin() + position);

}

void deleteDir(int solution, ArrClust \*Clust, Journal \*journal){

journal->AddMessage("Ydalenie papki " + dir[solution].name);

while(dir[solution].dir.size() + dir[solution].files.size() > 0)

{

if(dir[solution].dir.size() != 0)

dir[solution].deleteDir(0, Clust, journal);

else

dir[solution].deleteFile(0, Clust, journal);

}

Clust->bitvect[dir[solution].addres] = '0'; //4istim addres papki

Clust->NumberOfFree++;

dir.erase(dir.begin() + solution);

}

void Formatirovanie(ArrClust \*Clust, int fullmemory, Journal \*journal){

int ClSize;

cout << "Vvedite razmer klastera ";

cin >> ClSize;

if(ClSize < 1 || ClSize > fullmemory)

{

cout << "\nNevozmozhnii razmer klastera\n";

this->Formatirovanie(Clust, fullmemory, journal);

}

else

{

while(dir.size() > 0)

this->deleteDir(0, Clust, journal);

while(files.size() > 0)

this->deleteFile(0, Clust, journal);

Clust->Formatirovanie(ClSize, fullmemory);

}

}

void insertFile(File \*bufer, ArrClust \*Clust, Journal \*journal){

if(bufer->name != "\0")

{

int sizeee = bufer->AddresCluster.size();

if(sizeee <= Clust->NumberOfFree)

{

journal->AddMessage("Cozdanie kopii faila " + bufer->name);

if(findSimilarName(bufer->name))

bufer->name += "1";

bufer->AddresCluster.clear();

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

bufer->AddresCluster.push\_back(Clust->findFreeCluster());

files.push\_back(\*bufer);

bufer->Clear();

}

else

{

cout << "\nNet mecta";

system("PAUSE");

}

}

else

{

cout << "\nBufer pystoi";

system("PAUSE");

}

}

};

int main()

{

Journal journal;

journal.formatirovanie();

ArrClust Clust\_Disk;

Directory disk("C:");

disk.Formatirovanie(&Clust\_Disk, FullMemorySizeDisk, &journal);

ArrClust Clust\_flesh;

Directory flesh("E:");

flesh.Formatirovanie(&Clust\_flesh, FullMemorySizeFlesh, &journal);

Directory \*location = &disk;

ArrClust \*Clust\_location = &Clust\_Disk;

int tempMemory = FullMemorySizeDisk;

File bufer("\0");

string way = "\0", name2;

int solution;

while(true)

{

Clust\_location->showFullinformation(tempMemory);

cout << location->path << endl;

location->showDir2(0);

cout << "\n\nChoose option:\n";

cout << "1. Create new directory\n";

cout << "2. Delete directory\n";

cout << "3. Go to the specified folder\n";

cout << "4. Back to previous folder\n";

cout << "5. Create new file\n";

cout << "6. Delete File\n";

cout << "7. Redact file\n";

cout << "8. Show file contents\n";

cout << "9. Show Addreses of file\n";

cout << "10. Copy file to bufer\n";

cout << "11. Insert file\n";

cout << "12. Find path of directory or file\n";

cout << "13. Otformatirovat etot disk\n";

cout << "14. Pereiti na drugoi disk\n";

if(tempMemory == FullMemorySizeDisk)

cout << "15. Journal\n";

cout << "19. End program\n";

cin >> solution;

switch(solution)

{

case 1:

system("cls");

location->createDir(Clust\_location, &journal);

break;

case 2:

cout << "\nChoose directory ";

cin >> solution;

if((solution >= location->dir.size()) || (solution < 0))

cout << "\nWrong number ";

else

location->deleteDir(solution, Clust\_location, &journal);

system("cls");

break;

case 3:

cout << "\nChoose direction ";

cin >> solution;

if((solution >= location->dir.size()) || (solution < 0) || (solution == 0 && location->dir.size() == 0))

cout << "\nnet takoi papki";

else

location = &(location->dir[solution]);

system("cls");

break;

case 4:

if(location->parentDir != nullptr)

location = location->parentDir;

system("cls");

break;

case 5:

system("cls");

location->createFile(Clust\_location, &journal);

break;

case 6:

cout << "\nChoose the file ";

cin >> solution;

if(solution >= location->dir.size() && solution < location->files.size() + location->dir.size())

location->deleteFile(solution, Clust\_location, &journal);

else

cout << "\nNet takogo faila\n";

break;

case 7:

cout << "\nChoose file ";

cin >> solution;

if(solution >= location->dir.size() && solution < location->files.size() + location->dir.size())

location->files[solution-location->dir.size()].RedactFile(Clust\_location);

else

cout << "\nNet takogo faila\n";

break;

case 8:

location->showFile();

break;

case 9:

cout << "\nChoose file ";

cin >> solution;

if(solution >= location->dir.size() && solution < location->files.size() + location->dir.size())

location->files[solution-location->dir.size()].ShowAddres();

else

cout << "\nNet takogo faila\n";

break;

case 10:

cout << "\nChoose file ";

cin >> solution;

if(solution >= location->dir.size() && solution < location->files.size() + location->dir.size())

bufer = location->files[solution - location->dir.size()];

else

cout << "\nNet takogo faila\n";

break;

case 11:

location->insertFile(&bufer, Clust\_location, &journal);

break;

case 12:

system("cls");

cout << "Vvedite imya faila/directorii ";

cin >> name2;

while(location->parentDir != nullptr)

location = location->parentDir;

location->findPathOfObject(name2, &way);

cout << "\nDannii ob'ect mozhno naiti po etim adrecam: \n" << way;

way = "\0";

break;

case 13:

system("cls");

while(location->parentDir != nullptr)

location = location->parentDir;

location->Formatirovanie(Clust\_location, tempMemory, &journal);

journal.formatirovanie();

break;

case 14:

system("cls");

cout << "Choose disk to go \n";

cout << "1. " << disk.name << endl;

cout << "2. " << flesh.name << endl;

cin >> solution;

switch(solution)

{

case 1:

location = &disk;

Clust\_location = &Clust\_Disk;

tempMemory = FullMemorySizeDisk;

break;

case 2:

location = &flesh;

Clust\_location = &Clust\_flesh;

tempMemory = FullMemorySizeFlesh;

break;

default:

cout << "\nVi nepravilno ukazali cifru";

}

system("cls");

break;

case 15:

if(tempMemory == FullMemorySizeDisk)

{

system("cls");

journal.Show();

fflush(stdin);

getchar();

system("cls");

}

break;

case 19:

return 0;

break;

default:

cout << "\nVi nepravilno ukazali cifru, vvedite esche raz ";

}

}

}