**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информатика»

Лабораторная работа №8

по дисциплине «Архитектура вычислительных систем»

на тему: **«Реализация файловой системы»**

Исполнитель: студент гр. ИП-31

И.А. Васюков

Руководитель: ст.пр. Н.В. Самовендюк

Дата проверки: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата допуска к защите:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата защиты: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Гомель 2019

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

**Практическая часть:**

**Индивидуальные задания**

Разработать приложение, создающее виртуальный файл и позволяющее

- форматировать виртуальный файл с возможностью задания размера кластера;

- создавать каталоги в виртуальном файле;

- производить учёт свободного пространства;

- реализовывать поиск файлов и директорий;

- сохранять в виртуальный файл файлы с жёсткого диска;

- удалять файлы из виртуального файла;

- записывать на жёсткий диск файлы из виртуального файла;

- создавать в виртуальном файле текстовые файлы;

- предоставлять возможность редактировать текстовые файлы внутри виртуального файла.

Файловую систему внутри виртуального файла выбрать согласно варианта.

|  |  |  |  |
| --- | --- | --- | --- |
| **Вариант** | **Условие задачи** | **Учёт свободных блоков** | **Поиск файлов и папок** |
| 4 | Индексно-последовательная файловая система. Связанный список. | Связанный список | Хэш-таблицы |

**Код программы**

MyFile.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace OC\_LAB\_8

{

class Myfile

{

private List<int> blocksFiles = new List<int>();

public List<int> BlocksFiles { get => blocksFiles;}

public void AddBlock(int block)

{

BlocksFiles.Add(block);

}

}

}

MyDirectory.cs

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace OC\_LAB\_8

{

class MyDirectory

{

private Dictionary<string, Myfile> Files = new Dictionary<string, Myfile>();

private Dictionary<string, MyDirectory> Directories = new Dictionary<string, MyDirectory>();

public MyDirectory AddDirectory(string nameDirectory)

{

Directories.Add(nameDirectory, new MyDirectory());

return Directories[nameDirectory];

}

public void AddFile(string nameFile, string pathFile, GlobalSpace globalSpace)

{

Files.Add(nameFile, new Myfile());

Myfile myfile = Files[nameFile];

using (StreamReader sr = new StreamReader(pathFile, System.Text.Encoding.Default))

{

string text = "";

text = sr.ReadToEnd();

globalSpace.AddTextInMyFile(text, myfile);

}

}

public void AddFile(string nameFile)

{

Files.Add(nameFile, new Myfile());

}

public void Delete(string fullPath)

{

string[] splits = fullPath.Split('\\');

if (splits.Length > 1)

{

Directories[splits[0]].Delete(fullPath.Remove(0, splits[0].Length + 1));

}

else

{

if (Directories.ContainsKey(fullPath))

{

Directories.Remove(fullPath);

}

else

{

if (Files.ContainsKey(fullPath))

{

Files.Remove(fullPath);

}

}

}

}

public Myfile FindMyFile(string fullPath)

{

string[] splits = fullPath.Split('\\');

if (splits.Length > 1)

{

return Directories[splits[0]].FindMyFile(fullPath.Remove(0, splits[0].Length + 1));

} else

{

return Files[fullPath];

}

}

public MyDirectory FindMyDirectory(string fullPath)

{

string[] splits = fullPath.Split('\\');

if (splits.Length > 1)

{

return Directories[splits[0]].FindMyDirectory(fullPath.Remove(0, splits[0].Length + 1));

}

else

{

return Directories[fullPath];

}

}

}

}

Block.cs

namespace OC\_LAB\_8

{

class Block

{

public string text;

public int FreeSpace()

{

return GlobalSpace.maxBytes - text.Length;

}

}

}

Global.cs

using System.Collections.Generic;

namespace OC\_LAB\_8

{

class GlobalSpace

{

public MyDirectory root = new MyDirectory();

public static int AmountBlocks = 12;

public static List<Block> blocks = new List<Block>(AmountBlocks);

public static int maxBytes = 128;

public GlobalSpace()

{

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

{

blocks.Add(null);

}

}

public List<int> getFreeBlocks(int count)

{

List<int> indexBlocks = new List<int>();

for (int i = 0; i < AmountBlocks && count != 0; i++)

{

if (blocks[i] == null)

{

indexBlocks.Add(i);

count--;

}

}

return indexBlocks;

}

public Block GetBlock(int i)

{

if (i < AmountBlocks)

{

if (blocks[i] == null)

{

blocks[i] = new Block();

}

return blocks[i];

}

else

return null;

}

public void ClearBlock(int i)

{

blocks[i] = null;

}

public void AddTextInMyFile(string text, Myfile myfile)

{

int countBlocks = text.Length % maxBytes == 0 ? text.Length / maxBytes : (text.Length / maxBytes) + 1;

List<int> blocks = this.getFreeBlocks(countBlocks);

for (int i = 0; i < blocks.Count; i++)

{

Block block = this.GetBlock(blocks[i]);

string textInBlock = "";

if (i == blocks.Count - 1)

{

textInBlock = text.Substring(maxBytes \* i);

}

else

{

textInBlock = text.Substring(maxBytes \* i, maxBytes);

}

block.text = textInBlock;

myfile.AddBlock(blocks[i]);

}

}

public int CountFreeSpace()

{

int freeSpace = 0;

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

{

if (blocks[i] != null)

{

freeSpace += blocks[i].FreeSpace();

}

else

{

freeSpace += maxBytes;

}

}

return freeSpace;

}

}

}

Form1.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace OC\_LAB\_8

{

public partial class Form1 : Form

{

private GlobalSpace globalSpace = new GlobalSpace();

private MyDirectory RootDirectory;

private string path = "E:\\Учёба\\3\\Ос\\lab8\\OC\_LAB\_8\\OC\_LAB\_8\\mainFolder";

private string selectedFile;

public Form1()

{

RootDirectory = globalSpace.root;

InitializeComponent();

InitialTree();

CountSpace();

}

private void InitialTree()

{

if(Directory.Exists(path))

{

string[] dirs = Directory.GetDirectories(path);

if (dirs.Length != 0)

{

for (int i = 0; i < dirs.Length; i++)

{

string directoryName = new DirectoryInfo(dirs[i]).Name;

TreeNode dirNode = new TreeNode(directoryName);

MyDirectory newMyDirectory = RootDirectory.AddDirectory(directoryName);

FillTreeNode(dirNode, dirs[i], newMyDirectory);

Root.Nodes.Add(dirNode);

}

}

string[] files = Directory.GetFiles(path);

if (files.Length != 0)

{

for (int i = 0; i < files.Length; i++)

{

string fileName = new FileInfo(files[i]).Name;

TreeNode dirNode = new TreeNode(fileName);

Root.Nodes.Add(dirNode);

RootDirectory.AddFile(fileName, files[i], globalSpace);

}

}

}

}

private void FillTreeNode(TreeNode driveNode, string path, MyDirectory myDirectory)

{

try

{

string[] dirs = Directory.GetDirectories(path);

foreach (string dir in dirs)

{

TreeNode dirNode = new TreeNode();

dirNode.Text = dir.Remove(0, dir.LastIndexOf("\\") + 1);

driveNode.Nodes.Add(dirNode);

MyDirectory newMyDirectory = myDirectory.AddDirectory(dirNode.Text);

FillTreeNode(dirNode, dir, newMyDirectory);

}

string[] files = Directory.GetFiles(path);

foreach (string file in files)

{

TreeNode dirNode = new TreeNode();

dirNode.Text = file.Remove(0, file.LastIndexOf("\\") + 1);

driveNode.Nodes.Add(dirNode);

myDirectory.AddFile(dirNode.Text, file, globalSpace);

}

}

catch (Exception)

{ }

}

private void button2\_Click(object sender, EventArgs e)

{

string fileName = Root.SelectedNode.Text;

if (fileName.Contains(".txt"))

{

string fullPath = Root.SelectedNode.FullPath;

selectedFile = fullPath;

Myfile myfile = RootDirectory.FindMyFile(fullPath);

string contentFile = "";

foreach(int index in myfile.BlocksFiles)

{

contentFile += globalSpace.GetBlock(index).text;

}

richTextBox1.Text = contentFile;

}

CountSpace();

}

private void CreateFile\_Click(object sender, EventArgs e)

{

string directoryName = "";

if (Root.SelectedNode != null)

{

directoryName = Root.SelectedNode.Text;

}

if (!directoryName.Contains(".txt"))

{

MyDirectory myDirectory;

if (!directoryName.Equals(""))

{

string fullPath = Root.SelectedNode.FullPath;

myDirectory = RootDirectory.FindMyDirectory(fullPath);

}

else

{

myDirectory = RootDirectory;

}

string fileName = textBox1.Text;

if (!fileName.Contains("\\") && !fileName.Equals(""))

{

if (!fileName.Contains(".txt"))

{

fileName += ".txt";

}

myDirectory.AddFile(fileName);

TreeNode dirNode = new TreeNode(fileName);

if (Root.SelectedNode != null)

{

Root.SelectedNode.Nodes.Add(dirNode);

}

else

{

Root.Nodes.Add(dirNode);

}

}

}

CountSpace();

}

private void textBox1\_TextChanged(object sender, EventArgs e)

{

}

private void CreateDirectory\_Click(object sender, EventArgs e)

{

string directoryName = "";

if (Root.SelectedNode != null)

{

directoryName = Root.SelectedNode.Text;

}

if (!directoryName.Contains(".txt"))

{

MyDirectory myDirectory;

if (!directoryName.Equals(""))

{

string fullPath = Root.SelectedNode.FullPath;

myDirectory = RootDirectory.FindMyDirectory(fullPath);

}

else

{

myDirectory = RootDirectory;

}

string newDirectory = textBox1.Text;

if (!newDirectory.Contains("\\") && !newDirectory.Equals("") && !newDirectory.Contains("."))

{

myDirectory.AddDirectory(newDirectory);

TreeNode dirNode = new TreeNode(newDirectory);

if (Root.SelectedNode != null)

{

Root.SelectedNode.Nodes.Add(dirNode);

}

else

{

Root.Nodes.Add(dirNode);

}

}

}

CountSpace();

}

private void button3\_Click(object sender, EventArgs e)

{

if (!selectedFile.Equals(""))

{

Myfile myfile = RootDirectory.FindMyFile(selectedFile);

string text = richTextBox1.Text;

int countBlocks = text.Length % GlobalSpace.maxBytes == 0 ? text.Length / GlobalSpace.maxBytes : (text.Length / GlobalSpace.maxBytes) + 1;

int a = text.Length;

if (countBlocks >= myfile.BlocksFiles.Count)

{

List<int> blocks = globalSpace.getFreeBlocks(countBlocks - myfile.BlocksFiles.Count);

if (blocks.Count != countBlocks - myfile.BlocksFiles.Count)

{

MessageBox.Show("Размер файла превысил ограниение, файл не сохранён");

}

else

{

for (int i = 0; i < myfile.BlocksFiles.Count; i++)

{

globalSpace.ClearBlock(myfile.BlocksFiles[i]);

}

myfile.BlocksFiles.Clear();

globalSpace.AddTextInMyFile(text, myfile);

}

}

}

CountSpace();

}

private void button4\_Click(object sender, EventArgs e)

{

if (Root.SelectedNode != null)

{

RootDirectory.Delete(Root.SelectedNode.FullPath);

Root.SelectedNode.Remove();

}

CountSpace();

}

private void button1\_Click(object sender, EventArgs e)

{

string directoryName = "";

if (Root.SelectedNode != null)

{

directoryName = Root.SelectedNode.Text;

}

if (!directoryName.Contains(".txt"))

{

MyDirectory myDirectory;

if (!directoryName.Equals(""))

{

string fullPath = Root.SelectedNode.FullPath;

myDirectory = RootDirectory.FindMyDirectory(fullPath);

}

else

{

myDirectory = RootDirectory;

}

string newFilePath = textBox1.Text;

if (newFilePath.Contains(".txt") && !newFilePath.Equals(""))

{

string fileName = newFilePath.Remove(0, newFilePath.LastIndexOf("\\") + 1);

myDirectory.AddFile(fileName, newFilePath, globalSpace);

TreeNode dirNode = new TreeNode(fileName);

if (Root.SelectedNode != null)

{

Root.SelectedNode.Nodes.Add(dirNode);

}

else

{

Root.Nodes.Add(dirNode);

}

}

}

CountSpace();

}

private void button5\_Click(object sender, EventArgs e)

{

string fileName = "";

if (Root.SelectedNode != null)

{

fileName = Root.SelectedNode.Text;

}

if (!fileName.Equals("") && fileName.Contains(".txt"))

{

Myfile myfile = RootDirectory.FindMyFile(Root.SelectedNode.FullPath);

using (StreamWriter sw = new StreamWriter(textBox1.Text + "\\" + fileName, false, System.Text.Encoding.Default))

{

string text = "";

foreach (int index in myfile.BlocksFiles)

{

text += globalSpace.GetBlock(index).text;

}

sw.Write(text);

}

}

CountSpace();

}

private void CountSpace()

{

AllSpace.Text = (GlobalSpace.AmountBlocks \* GlobalSpace.maxBytes).ToString();

FreeSpace.Text = globalSpace.CountFreeSpace().ToString();

}

private void Form1\_Load(object sender, EventArgs e)

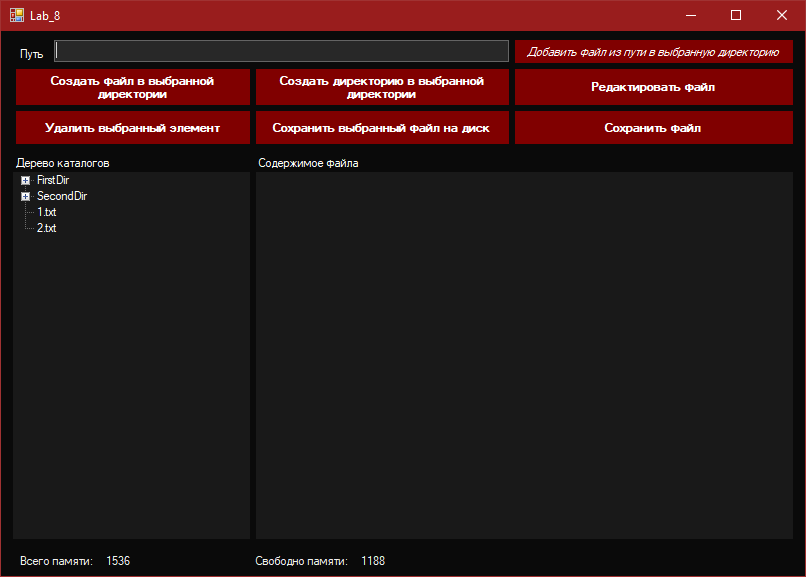
{

}

}

}

**Результат выполнения**



**Вывод:** В ходе работы была разработана модель файловой системы.