МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 10,11

по дисциплине **«Операционные системы»**

на тему: «Реализация файловой системы. Управление файловой системой»

Выполнил: студент гр. ИТ-11

Иванов А.А.

Принял: преподаватель
Петров И.И.

Гомель 2021

Реализация файловой системы.
Управление файловой системой.

Цель работы: разработать модель файловой системы

Задание

Разработать приложение, создающее виртуальный файл и позволяющее:

- форматировать виртуальный файл с возможностью задания размера кластера;
- создавать каталоги в виртуальном файле;
- производить учет свободного пространства;
- реализовывать поиск файлов и директорий;
- сохранять в виртуальный файл файлы с жесткого диска;
- удалять файлы из виртуального файла;
- записывать на жесткий 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	Задание
12	Последовательная файловая система. Выделение непрерывной последовательности блоков. (First Fit). Связанный

Решение:

Листинг программы:

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.IO;
using System.Windows.Forms;

namespace FileSystem
{
    public partial class Main : Form
    {
        /// <summary>
        /// Объект класса FileSystem
        /// </summary>
        public static FileSystem FileSystem = new FileSystem("1.txt");
```

```
        /// <summary>
        /// Инициализация формы
        /// </summary>
        public Main()
        {
            InitializeComponent();
        }

        /// <summary>
        /// Рекурсия для обновления дерева ФС
        /// </summary>
        /// <param name="root">Корень</param>
        /// <param name="file">Папка</param>
        private void UpdateTree(TreeNode root, FileEntry file)
        {
            var node = new TreeNode(file.Name);
            if (file.IsDir)
                foreach (var fileEntry in file.Entires)
                    UpdateTree(node, fileEntry);
            root.Nodes.Add(node);
        }

        /// <summary>
        /// Форматирование памяти
        /// </summary>
        private void форматироватьToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            var f = new Formating();
            f.ShowDialog();
            Activate();
            UpdateForm();
            editFile.Clear();
            editFile.Enabled = false;
            saveFile.Enabled = false;
        }

        /// <summary>
        /// Обновление информации
        /// </summary>
        public void UpdateForm()
        {
            infoLoadFile.Clear();
            UpdateGraf();
            UpdateInfoClasters();
            UpdateInfoDir();
            UpdateFat();

            treeFS.Nodes.Clear();
            var root = treeFS.Nodes.Add("Файловая система");
            UpdateTree(root, FileSystem.GetRootDir());
            treeFS.ExpandAll();

            Size.Text = "Объём памяти: " + FileSystem.GetSize();
            SizeClusters.Text = "Размер кластера: " +
FileSystem.GetSizeClusters();
```

```
            CountsClusters.Text = "Количество кластеров: " +
Convert.ToString(Convert.ToInt32(FileSystem.GetSize()) /
Convert.ToInt32(FileSystem.GetSizeClusters()));
            freeClusters.Text = "Свободных кластеров: " +
FileSystem.GetFreeClusters();
            freeMemory.Text = "Свободно памяти: " +
FileSystem.GetFreeMemory();
            ListFileBox.Text = "Файлы в: " + FileSystem.GetCurDir().Name;
            curDir.Text = "Текущий каталог: " + FileSystem.GetCurDir().Name;

            FileSystem.SaveFS();
        }

        private void UpdateFat()
        {
            fatTable.RowCount = 0;
            fatTable.RowCount =
Convert.ToInt32(FileSystem.GetSize())/Convert.ToInt32(FileSystem.GetSizeClust
ers());
            for (int i = 0; i < fatTable.RowCount; i++)
                fatTable[0, i].Value = i + 1;
            var root = FileSystem.GetRootDir();
            fatTable[1, root.Clusters[0]].Value = "ROOT DIR";
            fatTable[2, root.Clusters[0]].Value = root.Name;
            foreach (var file in root.Entires)
            {
                if (file.IsDir)
                {
                    fatTable[1, file.Clusters[0]].Value = "DIR";
                    fatTable[2, file.Clusters[0]].Value = file.Name;
                    UpdateFatRec(file);
                }
                else
                    UpdateFatFile(file);
            }
        }

        private void UpdateFatFile(FileEntry file)
        {
            bool flag = true;
            for (int i = 0; i < file.Clusters.Length; i++)
            {
                if ((i + 1) < file.Clusters.Length)
                {
                    int k = file.Clusters[i];
                    fatTable[1, file.Clusters[i]].Value = file.Clusters[(i +
1)] + 1;
                    if (k == file.Clusters[i] && flag)
                    {
                        fatTable[2, file.Clusters[i]].Value = ("Начало" +" "+
file.Name);
                        flag = false;
                    }
                }
                else
                {
                    fatTable[1, file.Clusters[i]].Value = "EOF";
                    fatTable[2, file.Clusters[i]].Value =  file.Name;
                }
```

```
    }
}

private void UpdateFatRec(FileEntry dir)
{
    foreach (var file in dir.Entires)
    {
        if (file.IsDir)
        {
            fatTable[1, file.Clusters[0]].Value = "DIR";
            fatTable[2, file.Clusters[0]].Value = (file.Name);
            UpdateFatRec(file);
        }
        else
            UpdateFatFile(file);
    }
}

/// <summary>
/// Обновление Claster Info
/// </summary>
private void UpdateInfoClasters()
{
    string s = "";
    Cluster[] clusters = FileSystem.GetClusters();
    for (int i = 0; i < clusters.Length; i++)
    {
        var cluster = clusters[i];
        if (cluster.Data == null)
            s += (i + 1) + ". null";
        else
            s += (i + 1) + ". " + cluster.Data;
        s += "\r\n";
    }
    infoClastersData.Text = s;
}

/// <summary>
/// Обновление таблицы кластеров
/// </summary>
private void UpdateGraf()
{
    var tmp = FileSystem.GetBitBock();
    var bmp = new Bitmap(panel1.Width, panel1.Height);
    panel1.BackgroundImage = bmp;
    using (Graphics g = Graphics.FromImage(bmp))
    {
        Brush free = new SolidBrush(Color.Chartreuse);
        Brush file = new SolidBrush(Color.CornflowerBlue);
        Brush dir = new SolidBrush(Color.Orange);
        int x = 0;
        int y = 0;
        int number = 0;
        foreach (var bit in tmp)
        {
            switch (bit)
            {
                case 0:
                    g.FillRectangle(free, x, y, 10, 10);
```

```
                        break;
                    case 1:
                        g.FillRectangle(dir, x, y, 10, 10);
                        break;
                    case 2:
                        g.FillRectangle(file, x, y, 10, 10);
                        break;
                }
                x += 12;
                number += 1;
                if (number != 107) continue;
                number = 0;
                x = 0;
                y += 12;
            }
            free.Dispose();
            file.Dispose();
            dir.Dispose();
        }
    }

    /// <summary>
    /// Обновление окна файлов
    /// </summary>
    private void UpdateInfoDir()
    {
        listFile.Items.Clear();
        listFile.Clear();
        var dir = FileSystem.GetCurDir();
        List<FileEntry> list = dir.Entires;
        if (dir.Parent != null)
        {
            var backDir = listFile.Items.Add("Назад", 0);
            backDir.Tag = dir.Parent;
        }
        foreach (FileEntry file in list)
        {
            if (file.IsDir)
            {
                var lvi = listFile.Items.Add(Convert.ToString(file.Name),
1);
                lvi.Tag = file;
            }
            else
            {
                var lvi = listFile.Items.Add(Convert.ToString(file.Name),
2);
                lvi.Tag = file;
            }
        }
    }

    /// <summary>
    /// Запуск файла
    /// </summary>
    private void Main_Load(object sender, EventArgs e)
    {
        FileSystem.Loaging();
        UpdateForm();
```

```
            editFile.Enabled = false;
            saveFile.Enabled = false;
        }

        /// <summary>
        /// Выбор файла если папка обновление окна, если нет загрузка
содержимого в editFile
        /// </summary>
        private void listFile_DoubleClick_1(object sender, EventArgs e)
        {
            var item = listFile.SelectedItems[0];
            var fileEntry = (FileEntry)item.Tag;
            if (fileEntry.IsDir)
            {
                editFile.Enabled = false;
                editFile.Clear();
                saveFile.Enabled = false;
                FileSystem.SetCurDir(fileEntry);
                UpdateForm();
            }
            else
            {
                editFile.Enabled = true;
                saveFile.Enabled = true;
                var data = "";
                var numberCluster = "";
                foreach (var clusterNum in fileEntry.Clusters)
                {
                    numberCluster += Convert.ToString(clusterNum + 1) + " ";
                    data += FileSystem.GetClusterData(clusterNum);
                }
                editFile.Text = data;
                infoLoadFile.Text = numberCluster;
            }
        }

        /// <summary>
        /// Сохранение введенного в editFile текста в выделенный файл
        /// </summary>
        private void saveFile_Click(object sender, EventArgs e)
        {
            var file = listFile.SelectedItems[0];
            var fe = (FileEntry)file.Tag;
            if (fe.IsDir)
            {
                MessageBox.Show("Нельзя выполнить запись данных в
папку!\nВыберите файл для сохранения!", "ERROR", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                return;
            }
            GetTextFromClustersAndWriteInFs(fe);
            UpdateForm();
            saveFile.Enabled = false;
        }

        /// <summary>
        /// Чтение из editFile разбитие на длинну кластера с запись в файл
        /// </summary>
        /// <param name="file">Входной файл для записи</param>
```

```
        private void GetTextFromClustersAndWriteInFs(FileEntry file)
        {
            editFile.Enabled = true;

            var clusterLength =
Convert.ToInt32(FileSystem.GetSizeClusters());
            var bitMask = FileSystem.GetBitBock();
            var clusters = FileSystem.GetClusters();
            var clusterNumber = 0;

            ClearFile(file, bitMask, clusters, editFile.Text.Length,
clusterLength);

            for (int index = 0; index < editFile.Text.Length; index++)
            {
                string tmpText = "";
                if (index + clusterLength <= editFile.Text.Length)
                {
                    tmpText += editFile.Text.Substring(index, clusterLength);
                    index += clusterLength - 1;
                }
                else
                {
                    tmpText += editFile.Text.Substring(index,
editFile.Text.Length - index);
                    index += editFile.Text.Length - index;
                }
                WriteFile(file, bitMask, clusters, tmpText, clusterNumber);
                clusterNumber += 1;
            }

            foreach (var fileEntry in FileSystem.GetCurDir().Entires)
                if (fileEntry.Name == file.Name)
                    fileEntry.Clusters = file.Clusters;

            editFile.Clear();
            editFile.Enabled = false;
        }

        /// <summary>
        /// Очистка файла перед записью и возвращение начального кластера
        /// </summary>
        /// <param name="file">Входной файл для обнуления содержимого</param>
        /// <param name="bitMask">Битовый вектор кластеров</param>
        /// <param name="clusters">Кластер</param>
        /// <param name="text">Записываемый текст</param>
        /// <param name="clusterLength">Длинна кластера</param>
        private void ClearFile(FileEntry file, int[] bitMask, Cluster[]
clusters, int text, int clusterLength)
        {
            var tmpNumClu = new int[file.Clusters.Length];
            Array.Copy(file.Clusters, tmpNumClu, file.Clusters.Length);
            if (tmpNumClu.Length != 0)
            {
                foreach (var bit in tmpNumClu)
                {
                    bitMask[bit] = 0;
                    clusters[bit].Data = null;
                }
```

```
            }
            var lenght = 0;
            var tmp = text / clusterLength;
            if (text % clusterLength != 0)
                tmp += 1;
            tmpNumClu = new int[tmp];
            for (var i = 0; i < bitMask.Length && lenght < tmp; i++)
            {
                if (bitMask[i] != 0) continue;
                tmpNumClu[lenght] = i;
                lenght += 1;
            }
            Array.Resize(ref file.Clusters, tmpNumClu.Length);
            Array.Copy(tmpNumClu, file.Clusters, tmpNumClu.Length);
            UpdateForm();
        }

        /// <summary>
        /// Запись в кластера
        /// </summary>
        /// <param name="file">Входной файл для записи</param>
        /// <param name="bitMask">Битовый вектор кластеров</param>
        /// <param name="clusters">Кластер</param>
        /// <param name="s">Входная строка равная длинне кластера</param>
        /// <param name="clusterNumber">Номер кластера для записи</param>
        private void WriteFile(FileEntry file, int[] bitMask, Cluster[]
clusters, string s, int clusterNumber)
        {
            var tmpNumClu = new int[file.Clusters.Length];
            Array.Copy(file.Clusters, tmpNumClu, file.Clusters.Length);
            clusters[tmpNumClu[clusterNumber]].Data = s;
            bitMask[tmpNumClu[clusterNumber]] = 2;
        }

        /// <summary>
        /// Очистка формы editFile
        /// </summary>
        private void clear_Click(object sender, EventArgs e)
        {
            editFile.Clear();
            saveFile.Enabled = false;
            editFile.Enabled = false;
            infoLoadFile.Clear();
        }

        /// <summary>
        /// Удаление объекта
        /// </summary>
        /// <param name="file">Объект</param>
        /// <param name="dir">Текущая папка</param>
        private void DeletingItem(FileEntry file, FileEntry dir)
        {
            if (file.IsDir)
            {
                var countDir = file.Entires.Count;
                for (int i = 0; i < countDir; i++)
                {
                    if (file.Entires[i].IsDir)
                    {
```

```
                DeletingItem(file.Entires[i], file.Parent);
                countDir = file.Entires.Count;
            }
            else
            {
                DelFile(file.Entires[i]);
                file.Entires.Remove(file.Entires[i]);
                countDir = file.Entires.Count;
            }
            i--;
        }
        DelDir(file);
    }
    else
    {
        DelFile(file);
        dir.Entires.Remove(file);
    }
}

/// <summary>
/// Удаление файла
/// </summary>
/// <param name="file">Файл</param>
private static void DelFile(FileEntry file)
{
    var bitMask = FileSystem.GetBitBock();
    var clusters = FileSystem.GetClusters();
    var tmpNumClu = new int[file.Clusters.Length];
    Array.Copy(file.Clusters, tmpNumClu, file.Clusters.Length);
    foreach (var bit in tmpNumClu)
    {
        bitMask[bit] = 0;
        clusters[bit].Data = null;
    }
}

/// <summary>
/// Удаление папки
/// </summary>
/// <param name="dir">Папка</param>
private static void DelDir(FileEntry dir)
{
    var bitMask = FileSystem.GetBitBock();
    var clusters = FileSystem.GetClusters();
    for (int i = 0; i < dir.Parent.Entires.Count; i++)
    {
        if (dir.Parent.Entires[i].Name == dir.Name)
        {
            var tmpCluster = dir.Clusters[0];
            clusters[tmpCluster].Data = null;
            bitMask[tmpCluster] = 0;
            dir.Parent.Entires.Remove(dir.Parent.Entires[i]);
        }
    }
}

/// <summary>
/// Выгрузка из файловой системы
```

```
        /// </summary>
        private void сохранитьВToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            if (saveFileOutFS.ShowDialog() == DialogResult.OK)
            {
                var writer = new StreamWriter(saveFileOutFS.FileName);
                var file = listFile.SelectedItems[0];
                var fe = (FileEntry)file.Tag;
                var dir = FileSystem.GetCurDir();
                string data = "";
                foreach (var fileEntry in dir.Entires)
                {
                    if (fileEntry.Name == fe.Name)
                    {
                        foreach (var clusterNum in fileEntry.Clusters)
                            data += FileSystem.GetClusterData(clusterNum);
                        writer.WriteLine(data);
                        break;
                    }
                }
                writer.Close();
                UpdateForm();
            }
        }

        /// <summary>
        /// Загрузка файла в файловую систему
        /// </summary>
        private void загрузитьToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            if (LoagingFileInFS.ShowDialog() != DialogResult.OK) return;
            editFile.Enabled = true;
            editFile.Clear();
            var reader = new StreamReader(LoagingFileInFS.FileName);
            while (true)
            {
                var s = reader.ReadLine();
                if (s == null)
                    break;
                editFile.Text += s;
            }
            FileSystem.CreateFile(LoagingFileInFS.SafeFileName);
            var dir = FileSystem.GetCurDir();
            foreach (var fileEntry in dir.Entires)
            {
                if (fileEntry.Name == LoagingFileInFS.SafeFileName)
                {
                    GetTextFromClustersAndWriteInFs(fileEntry);
                    break;
                }
            }
            reader.Close();
            UpdateForm();
        }

        /// <summary>
        /// Кнопка "Найти"
```

```
        /// </summary>
        private void seach_Click(object sender, EventArgs e)
        {
            SeachFile();
        }

        /// <summary>
        /// Поиск файла, переход в его папку и выделение найденного файлаы
        /// </summary>
        private void SeachFile()
        {
            var seachFile = seachBox.Text;
            FileSystem.Seach(FileSystem.GetRootDir(), seachFile);
            UpdateForm();
            var findColection = listFile.Items;
            for (int i = 0; i < findColection.Count; i++)
            {
                var textItem = findColection[i].Text;
                if (textItem == seachFile)
                {
                    MessageBox.Show("Файл: " + seachFile + " Найден!",
"Завершено!", MessageBoxButtons.OK,
                        MessageBoxIcon.Information);
                    listFile.Select();
                    findColection[i].Selected = true;
                    listFile.EnsureVisible(i);
                    seachBox.Clear();
                    return;
                }
            }
            MessageBox.Show("Файл: " + seachFile + " отсутствует!", "Ошибка",
MessageBoxButtons.OK,
                    MessageBoxIcon.Error);
        }

        /// <summary>
        /// Удаление выделеного файла или папки
        /// </summary>
        private void удалитToolStripMenuItem_Click(object sender, EventArgs
e)
        {
            if (listFile.SelectedItems.Count == 0)
                MessageBox.Show("Нет выбранного файла!", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            else
            {
                for (int i = 0; i < listFile.SelectedItems.Count; i++)
                {
                    var selectedFile = listFile.SelectedItems[i];
                    var file = (FileEntry)selectedFile.Tag;
                    var dir = FileSystem.GetCurDir();
                    DeletingItem(file, dir);
                }
            }
            UpdateForm();
        }

        /// <summary>
        /// Кнопка добавление файла в диалоговом окне listFile
```

```
        /// </summary>
        private void создатьФайлToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            editFile.Enabled = false;
            editFile.Clear();
            var rand = new Random();
            int tmp = rand.Next(1, 1000);
            string filename = "Файл " + Convert.ToString(tmp);
            FileSystem.CreateFile(filename);
            UpdateForm();
        }

        /// <summary>
        /// Кнопка создание каталога в диалоговом окне listFile
        /// </summary>
        private void создатьПапкуToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            var rand = new Random();
            int tmp = rand.Next(1, 1000);
            string dirname = "Папка " + Convert.ToString(tmp);
            FileSystem.CreateDir(dirname);
            UpdateForm();
        }

        /// <summary>
        /// Запуск изменения размера кластера
        /// </summary>
        private void рекластизироватьToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            var f = new Recluster();
            f.ShowDialog();
            Activate();
            UpdateForm();
        }

        private void Main_menu_ItemClicked(object sender,
ToolStripItemClickedEventArgs e)
        {

        }

    }

  }
```

Результат

Результат выполнения программы представлен на рисунке 1.

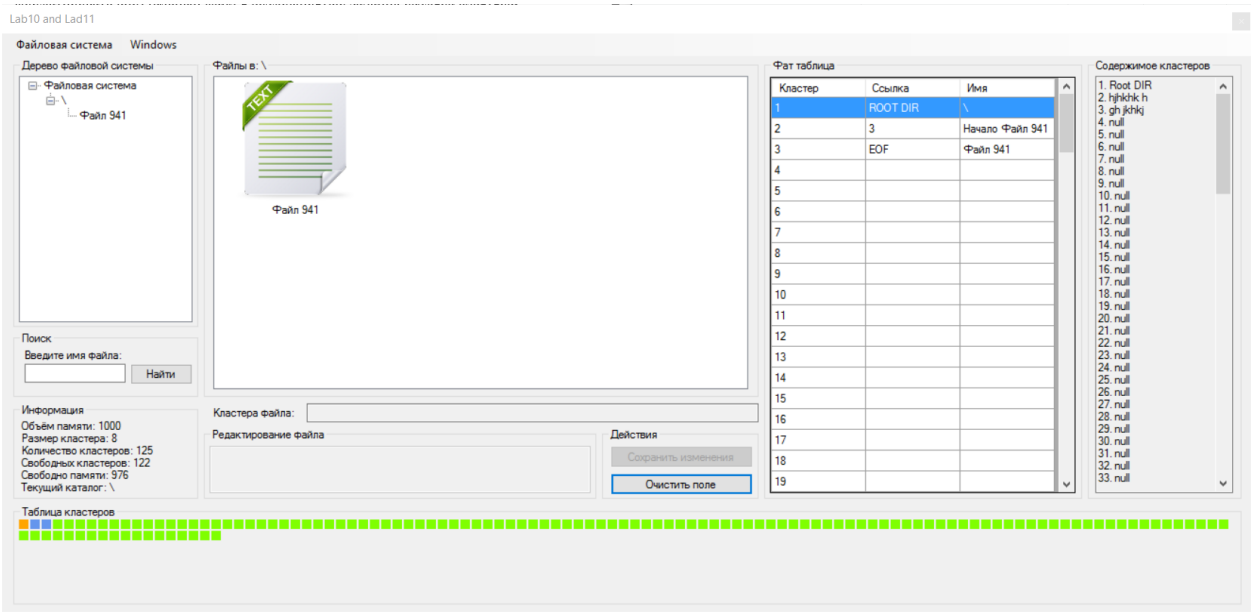

Рисунок 1 – Результат выполнения программы

Вывод: В ходе выполнения данной лабораторной работы была разработана модель файловой системы.