УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

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

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 11

по дисциплине «Операционные системы»

На тему **«**Управление файловой системой»

Выполнил: студент гр. ИТП-11

Александров А. Г.

Принял: преподаватель-стажер

Карась О. В.

Гомель 2022

**Цель**: реализовать вспомогательную утилиту для файловой системы.

**Ход работы**

**Задание 1.** Реализовать вспомогательную утилиту для управления файловой системой.

В соответствии с вариантом необходимо реализовать утилиту «Дефрагментация».

Дефрагментация – процесс объединения нескольких разрозненных фрагментов (частей) одного файла в один большой для повышения скорости считывания данных с жесткого диска компьютера.

При запуске программы мы увидим окно, в котором нам предложат ввести размер памяти, например, введем 10 и добавим несколько файлов, результат данных действий представлен на рисунке 1 ниже:

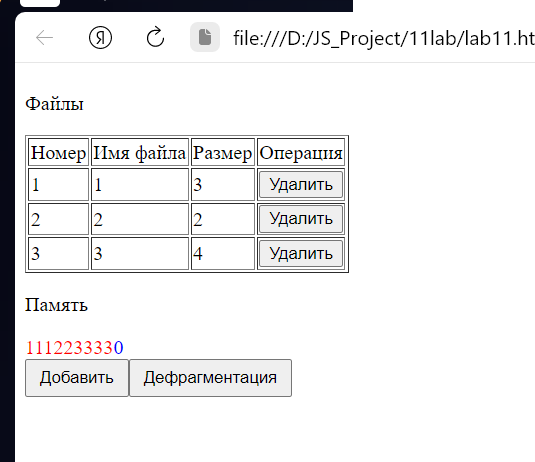


Рисунок 1 ­– Окно программы

При удалении 2 файла, образуется пустое место, которое при нажатии на кнопку «фрагментировать» будет перезаписано в один фрагмент, расположенный впритык для быстрого чтения файлов. Ниже представлен рисунок после удаления и до фрагментации:

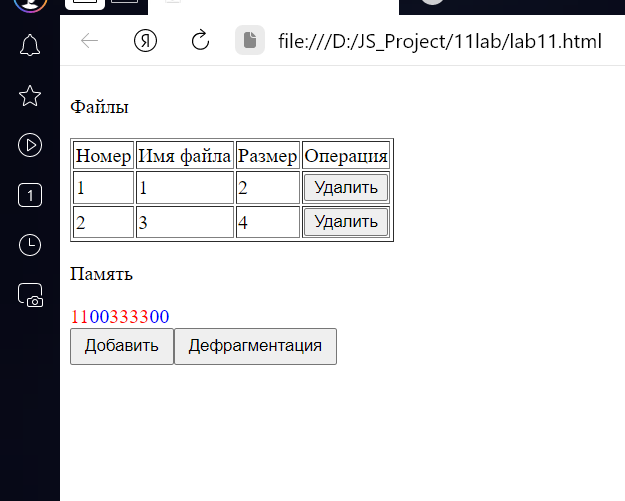


Рисунок 2 ­– Окно программы после удаления 2 файла

Ниже представлен рисунок после фрагментации «диска», на котором теперь файлы располагаются в порядке чтения:

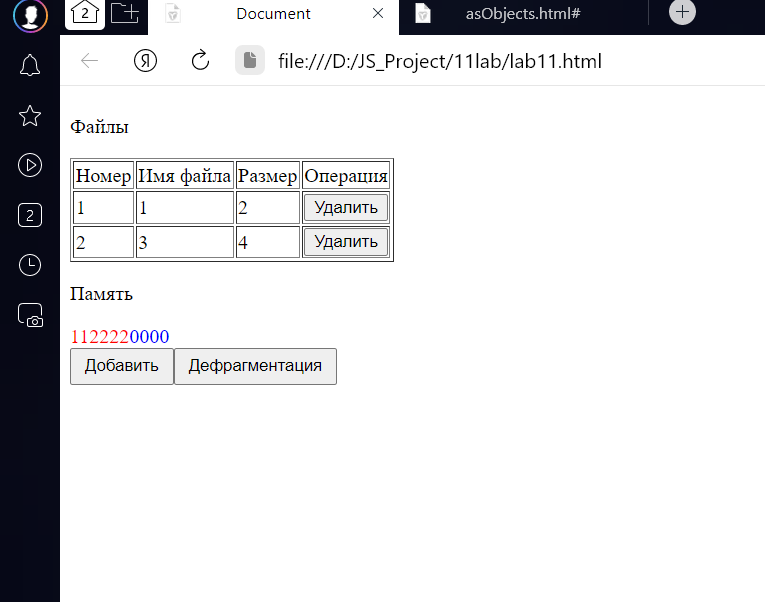


Рисунок 3 ­– Окно программы после фрагментации

**Вывод**: в ходе работы была реализована вспомогательная утилита «Дефрагментация».

**ПРИЛОЖЕНИЕ А**

**Листинг программы**

**Код утилиты «Дефрагментация»**

let addFileBtn = document.getElementById("addFile");

let defragmentBtn = document.getElementById("defragment");

let filesInfo = document.querySelector(".info\_\_files");

let memoryInfo = document.querySelector(".info\_\_memory");

let memory = {

    blocksAmount: 0,

    blocks: [],

    files: [],

}

memory.blocksAmount = +prompt("Введите размер памяти");

for (let i = 0; i < memory.blocksAmount; i++)

    memory.blocks[i] = 0;

defragmentBtn.addEventListener("click", (e) => {

    defragment();

    showFilesTable(memory, true);

    showMemory(memory, true);

})

addFileBtn.addEventListener("click", (e) => {

    let name = prompt("Имя файла: ");

    let size = +prompt("Размер файла:");

    addFile(name, size);

    showFilesTable(memory, true);

    showMemory(memory, true);

});

function getBackMemoryTo(number) {

    memory = journal[number].memory;

    showFilesTable(memory, true);

    showMemory(memory, true);

}

function defragment() {

    let files = memory.files;

    for (let i = 0; i < memory.blocksAmount; i++)

        memory.blocks[i] = 0;

    memory.files = [];

    for (let i = 0; i < files.length; i++)

        addFile(files[i].name, files[i].size);

}

function addFile(name, size) {

    let startPos = 0;

    if (memory.files.some(file => file.name == name))

    {

        alert("Файл с таким именем уже есть!");

        return;

    }

    if (!memory.files.length) {

        startPos = 0;

    } else

        startPos = memory.files[memory.files.length - 1].start +  memory.files[memory.files.length - 1].size;

    if (memory.blocks.length - startPos + 1 < size) {

        alert("Не достаточно памяти!");

        return;

    }

    let len = 0;

    for (let i = startPos; i < memory.blocks.length && len != size; i++, len++)

        memory.blocks[i] = memory.files.length + 1;

    let file = {

        name: name,

        size: size,

        start: startPos,

    };

    memory.files.push(file);

}

function deleteFile(name) {

    if (!memory.files.some(file => file.name == name)) {

        alert("Такого файла нет!");

        return;

    }

    let fileI = 0;

    memory.files.forEach((file, ind) => file.name == name && (fileI = ind));

    for (let i = memory.files[fileI].start; i < memory.files[fileI].start + memory.files[fileI].size; i++)

        memory.blocks[i] = 0;

    memory.files = del(memory.files, fileI);

    showFilesTable(memory, true);

    showMemory(memory, true);

    showJournalTable(true);

}

function del(arr, pos) {

    if (pos + 1 == arr.length)

        return [...arr.slice(0, pos)]

    else if (pos == 0)

        return [...arr.slice(1)]

    else

        return [...arr.slice(0, pos), ...arr.slice(pos + 1, arr.length)];

}

function showFilesTable(memory, shouldShow) {

    let str = "";

    str += `<table border='1'><thead><tr><td>Номер</td><td>Имя файла</td><td>Размер</td><td>Операция</td></tr></thead><tbody>`;

    memory.files.forEach((file, ind) => {

        str += "<tr>";

        str += `<td>${ind + 1}</td><td>${file.name}</td><td>${file.size}</td><td><input type='button' value='Удалить' onclick='deleteFile("${file.name}")'></td></tr>`;

    });

    str += "</tbody></table>";

    if (shouldShow)

        filesInfo.innerHTML = str;

    return str;

}

function showMemory(memory, shouldShow) {

    let str = "";

    let c = 0;

    memoryInfo.innerHTML = "";

    memory.blocks.forEach(block => {

        str += `<span ${block !== 0 ? "style='color:red;'" : "style='color:blue;'"}>${block}</span>`

    });

    if (shouldShow)

        memoryInfo.innerHTML += str + "<br>";

    return str;

}