УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

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

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 10

по дисциплине «Операционные системы»

На тему **«**Управление виртуальной памятью.

Алгоритмы замещения страниц»

Выполнил: студент гр. ИТП-11

Александров А. Г.

Принял: преподаватель-стажер

Карась О. В.

Гомель 2022

**Цель**: изучить приемы управления виртуальной памятью, а также алгоритмы замещения страниц и реализовать программу, управляющую виртуальной памятью.

**Ход работы**

**Задание 1**. Разработать программу, реализующий заданный алгоритм замещения страниц в памяти.

Менеджер памяти должен:

* Разбивать память заданного размера на указанное количество страниц. На экран должна выводиться информация о состоянии памяти: объем памяти, число страниц, число свободных страниц (%), размер страницы;
* Размещать в памяти страницу заданного процесса с замещением занятой по заданному алгоритму (по нажатию кнопки «Добавить»). Для размещения страницы в памяти указывается имя процесса и ее номер (вводятся отдельно). После нажатия на кнопку «Добавить» страница размещается в свободной странице памяти. Если задано глобальное размещение, то выбирается любая незанятая страница. При локальном размещении страница помещается среди виртуальных страниц, выделенных этому процессу. Выделение страниц в памяти выполняется при первом ее занесении процесса в память. Алгоритм замещения выполняется только при отсутствии свободных страниц под процесс;
* Удалить из памяти заданную страницу или все страницы заданного процесса (по нажатию кнопки «Удалить»). Указывается номер удаляемой страницы в памяти.
* Организовать циклическое обращение к страницам размещенным в памяти по нажатию на кнопку, при этом случайным образом задается количество обращений к страницам. Для каждого обращения генерируется номер страницы из диапазона от 0 до количества страниц памяти. При обращении к странице в зависимости от варианта увеличивается ее внутренний счетчик обращений или устанавливается флаг обращения.

**Вариант 1**

Глобальное замещение с алгоритмом *Random*.

При открытии программы нам предложат ввести объем памяти, затем, объем страницы, и количество страниц будет равно объем пяти / объем страницы. Например, если мы введем объем памяти, равный 10, и размер страницы 2, то в и итоге мы получим 5 страниц. Результат представлен на рисунке 1 ниже:

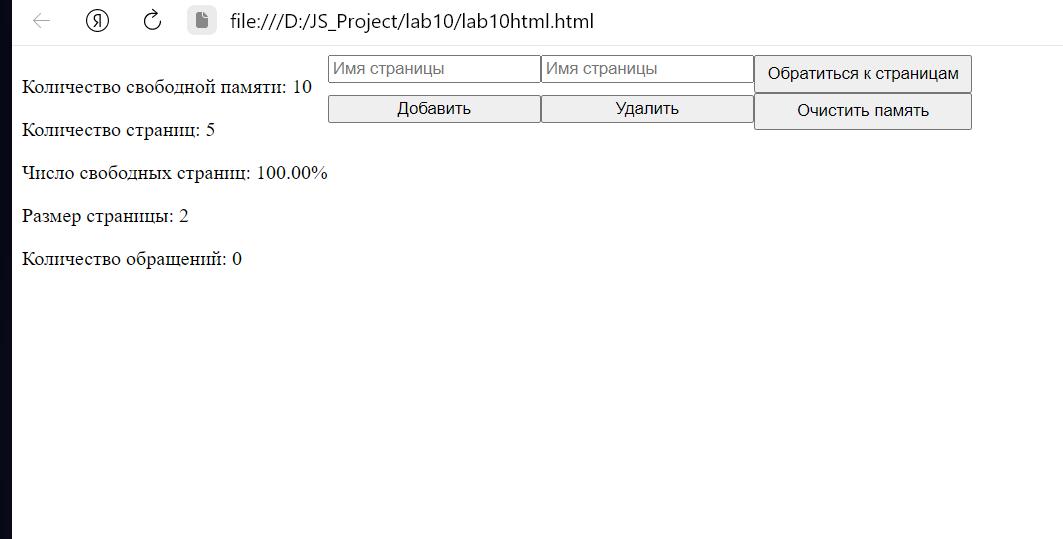


Рисунок 1 ­– Окно после ввода данных

Теперь мы можем добавить несколько страниц, чтобы проверить алгоритм замещения страниц. Сам же алгоритм состоит в том, если у нас не будет свободных страниц, то выгрузится случайная страница. Страницу можно выгрузить по нажатию на кнопку «Очистить память» или же это произойдет, если при добавлении страницы нам не хватит памяти. Окно после добавления страниц представлено на рисунке 2:

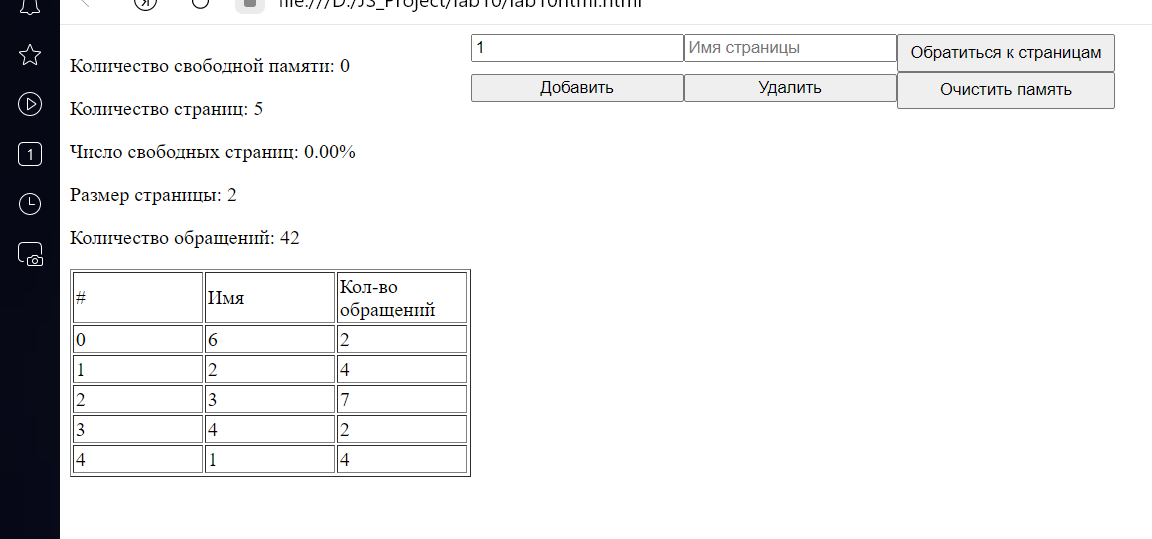


Рисунок 2 ­– Окно после добавления страниц

Мы можем сымитировать обращения к страницам, нажав на кнопку «Обратиться страницам». Кнопка сгенерирует случайно число от 1 до 10 – количество обращений, и для каждого обращения сгенерирует случайное число от 1 до количества страниц – страница, к которой обратятся. При нажатии на кнопку «удалить», мы можем выбрать нужную для нас страницу, которую мы хотим удалить.

**Вывод:** в ходе работы были изучены приемы управления виртуальной памятью, а также алгоритмы замещения страниц и была реализована программа, управляющую виртуальной памятью.

**ПРИЛОЖЕНИЕ А**

**Листинг программы**

let pagesInfo = document.getElementById("pagesInfo");

let pagesInfoTable = document.getElementById("pagesInfoTable");

let pageNameAdd = document.getElementById("pageNameAdd");

let addPageBtn = document.getElementById("addPageBtn");

let pageNameDel = document.getElementById("pageNameDel");

let deletePageBtn = document.getElementById("deletePageBtn");

let callPageBtn = document.getElementById("callPageBtn");

let freeMemoryBtn = document.getElementById("freeMemoryBtn");

let memory = {

    total: 0,

    pagesAmount: 0,

    pageSize: 0,

    pages: [],

    totalQuieries: 0

}

let total = +prompt("Введите объем памяти: ");

let pageSize = +prompt("Введите объем страницы: ");

memory.total = total;

memory.pageSize = pageSize;

memory.pagesAmount = Math.floor(total / pageSize);

showPagesInfo(memory);

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

   if (!memory.pages.length) {

       alert("Страниц нет!");

       return;

    } else

        freeMemory(memory, false)

});

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

   if (!memory.pages.length) {

       alert("Страниц нет!");

       return;

    } else

        callPage(memory)

});

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

    let name = pageNameAdd.value;

    if (!name)

    {

        alert("Вы не ввели имя страницы!");

        return;

    } else

        addPage(memory, name)

});

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

    let name = pageNameDel.value;

    if (!name)

    {

        alert("Вы не ввели имя страницы!");

        return;

    } else

        deletePage(memory, name)

});

function freeMemory(memory, newPage) {

    console.log(Math.floor((Math.random() \* memory.pages.length + 1)))

    memory.pages = del(memory.pages, Math.floor((Math.random() \* memory.pages.length )))

    showPagesTable(memory);

    showPagesInfo(memory);

}

function addPage(memory, pageName) {

    if (memory.pages.some(page => page.name == pageName)) {

        alert("Страница с таким именем уже есть!")

        return;

    }

    if (memory.pages.length >= memory.pagesAmount)

        freeMemory(memory, {

            name: pageName,

            queries: 0,

            count: 0

        });

    else

        memory.pages.push({

            name: pageName,

            queries: 0,

            count: 0

        });

    showPagesTable(memory);

    showPagesInfo(memory);

}

function callPage(memory)

{

    let amountOfQueries = Math.floor(Math.random()\*10 + 1);

    for (let i = 0; i < amountOfQueries; i++)

    {

        memory.pages[Math.floor(Math.random() \* memory.pages.length)].queries++;

        memory.totalQuieries++;

    }

     showPagesTable(memory);

     showPagesInfo(memory);

}

function deletePage(memory, pageName) {

    if (memory.pages.some(page => page.name == pageName)) {

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

            if (memory.pages[i].name == pageName)

            {

                memory.pages = del(memory.pages, i);

                showPagesTable(memory);

                showPagesInfo(memory);

            }

    } else {

        alert("Такой страницы нет!");

        return;

    }

}

function showPagesInfo(memory) {

    let str = `<p>Количество свободной памяти: ${memory.total - memory.pages.length \* memory.pageSize}</p><p>Количество страниц: ${memory.pagesAmount}</p><p>Число свободных страниц: ${(memory.pages.length ? 100 - (100 \* memory.pages.length) / memory.pagesAmount : 100).toFixed(2)}%</p><p>Размер страницы: ${memory.pageSize}</p><p>Количество обращений: ${memory.totalQuieries}</p>`;

    pagesInfo.innerHTML = str;

}

function showPagesTable(memory) {

    let str = "<table border='1'><thead><tr><td>#</td><td>Имя</td><td>Кол-во обращений</td></tr></thead><tbody>";

    memory.pages.forEach((page, ind) => {

        str += `<tr>`;

        str += `<td>${ind}</td><td>${page.name}</td><td>${page.queries}</td>`

        str += `</tr>`;

    })

    str += "</tbody></table>";

    pagesInfoTable.innerHTML = str;

}

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)];

}