**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 №9**

по дисциплине «Операционные системы»

на тему: «Простейшие схемы управления памятью»

Выполнил: студент гр. ИТИ-11

Цветков А.А.

Принял: преподаватель-стажёр Карась О.В.

Гомель 2022

**Цель работы:** изучение алгоритмов управления памятью, разработка программы менеджера памяти.

**Задание:**

Разработать программу, реализующую заданный алгоритм выделения памяти. Менеджер памяти должен:

1. По запросу процесса выделять память, согласно заданного алгоритма (таблица). На экран должна выводиться следующая информация о состоянии памяти: объем памяти, объѐм свободной памяти, размер наибольшего свободного блока, количество запросов на выделение памяти, количество удовлетворѐнных запросов (%).

2. Для выделения памяти указывается имя процесса и размер блока. После нажатия на кнопку «ДОБАВИТЬ» память выделяется или выдаѐтся сообщение о невозможности выделения.

3. Удалять из памяти заданный блок или все блоки заданного процесса (по нажатию кнопки «УДАЛИТЬ»). Указывается номер удаляемого блока и имя процесса.

4. Реализовать возможность последовательной записи/чтения информации в/из выделенную память по логическому адресу. Вывести физического адреса ячейки памяти, в которую была осуществлена запись.

5. Организовывать циклическое выделение и освобождение памяти. При этом случайным образом задается количество выделяемых блоков и их размер.

**Ход работы**

Для разработки программы менеджера памяти был выбран язык программирования JavaScript.

На рисунке показан запуск программы, где нужно выбрать один из четырёх пунктов меню.

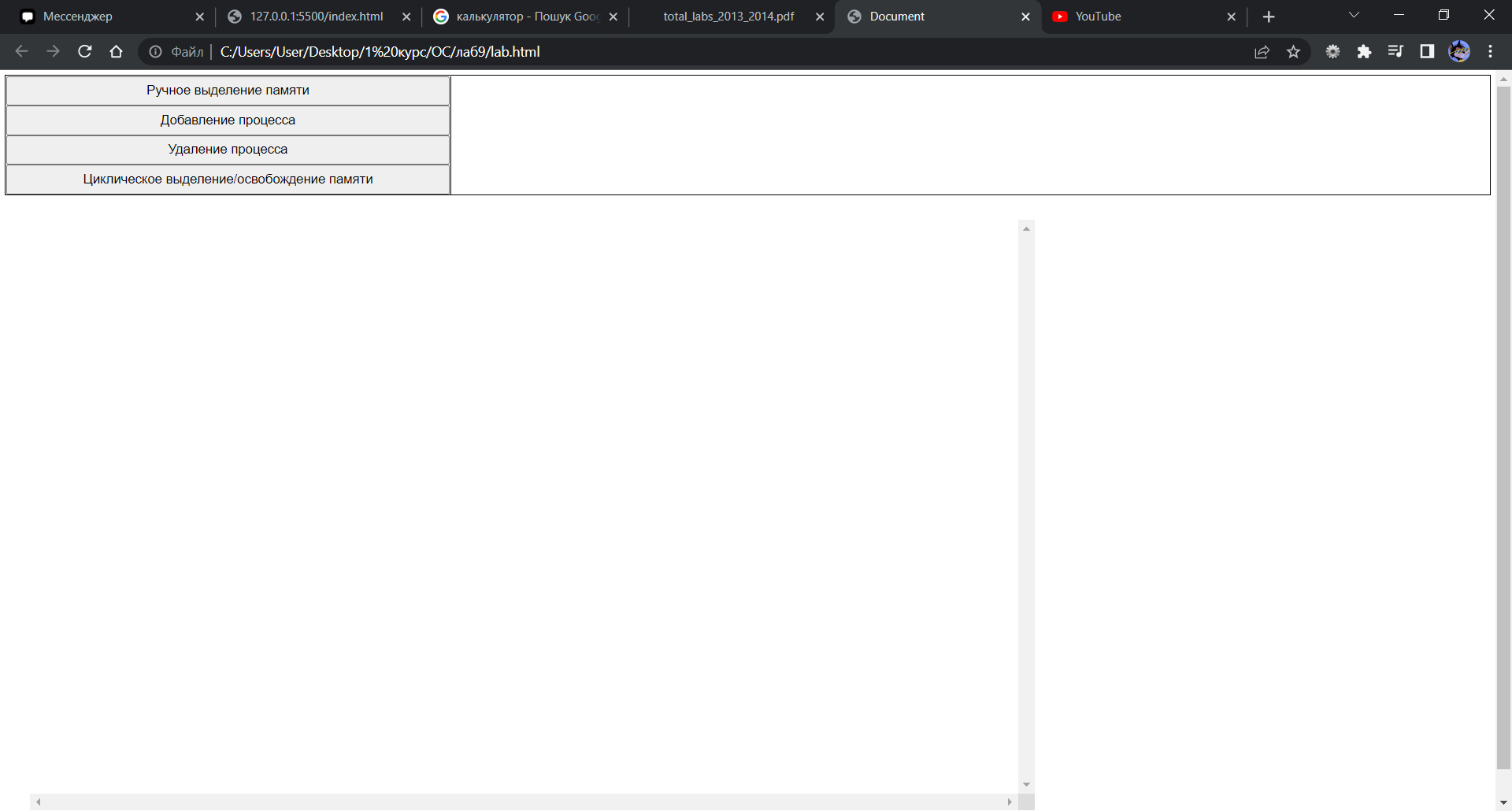


Рисунок 1 – Запуск программы

На рисунке 2 изображён процесс ручного выделения памяти.

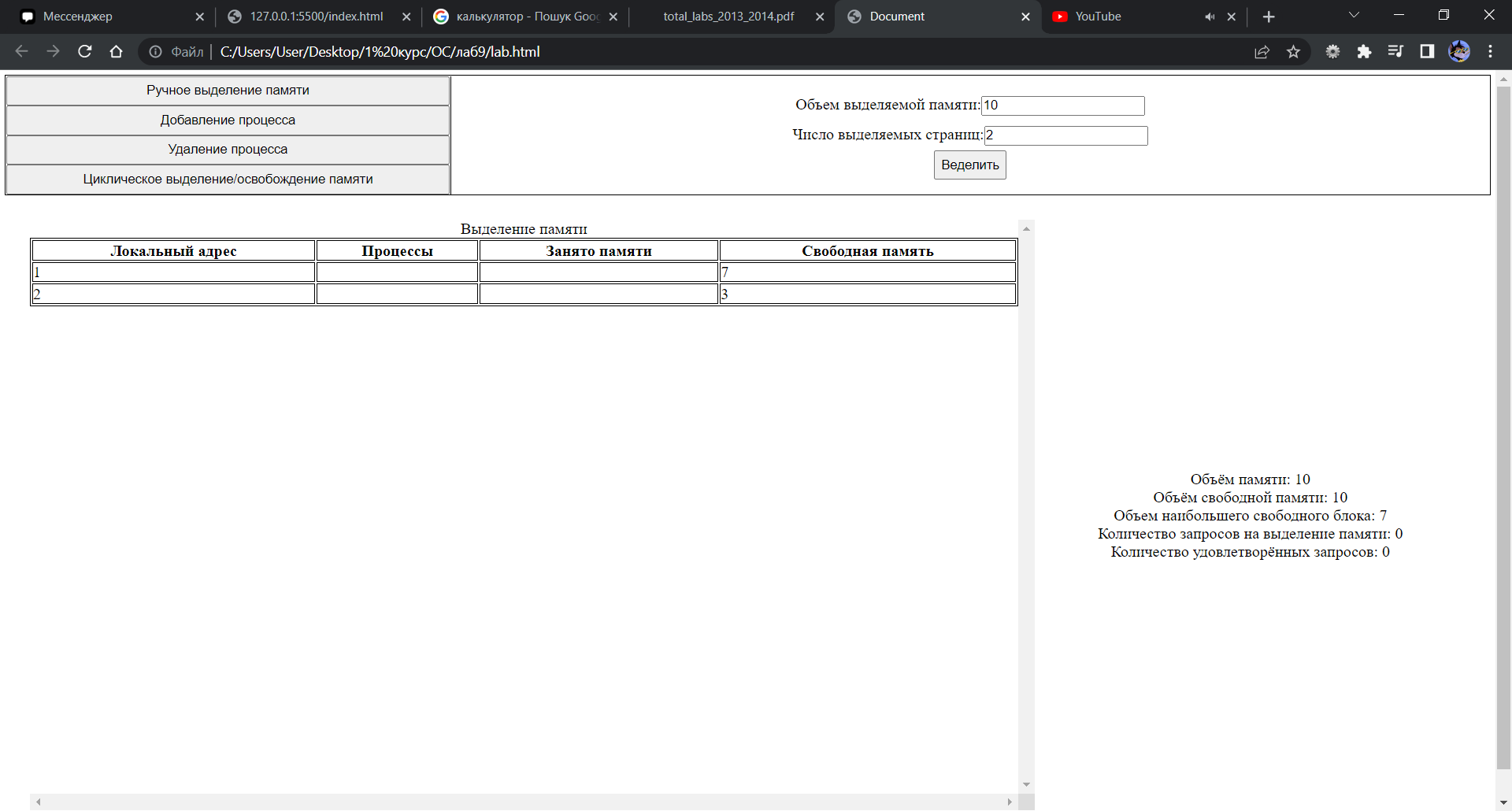


Рисунок 2 – Окно программы

На рисунке 3 показано добавление процессов.

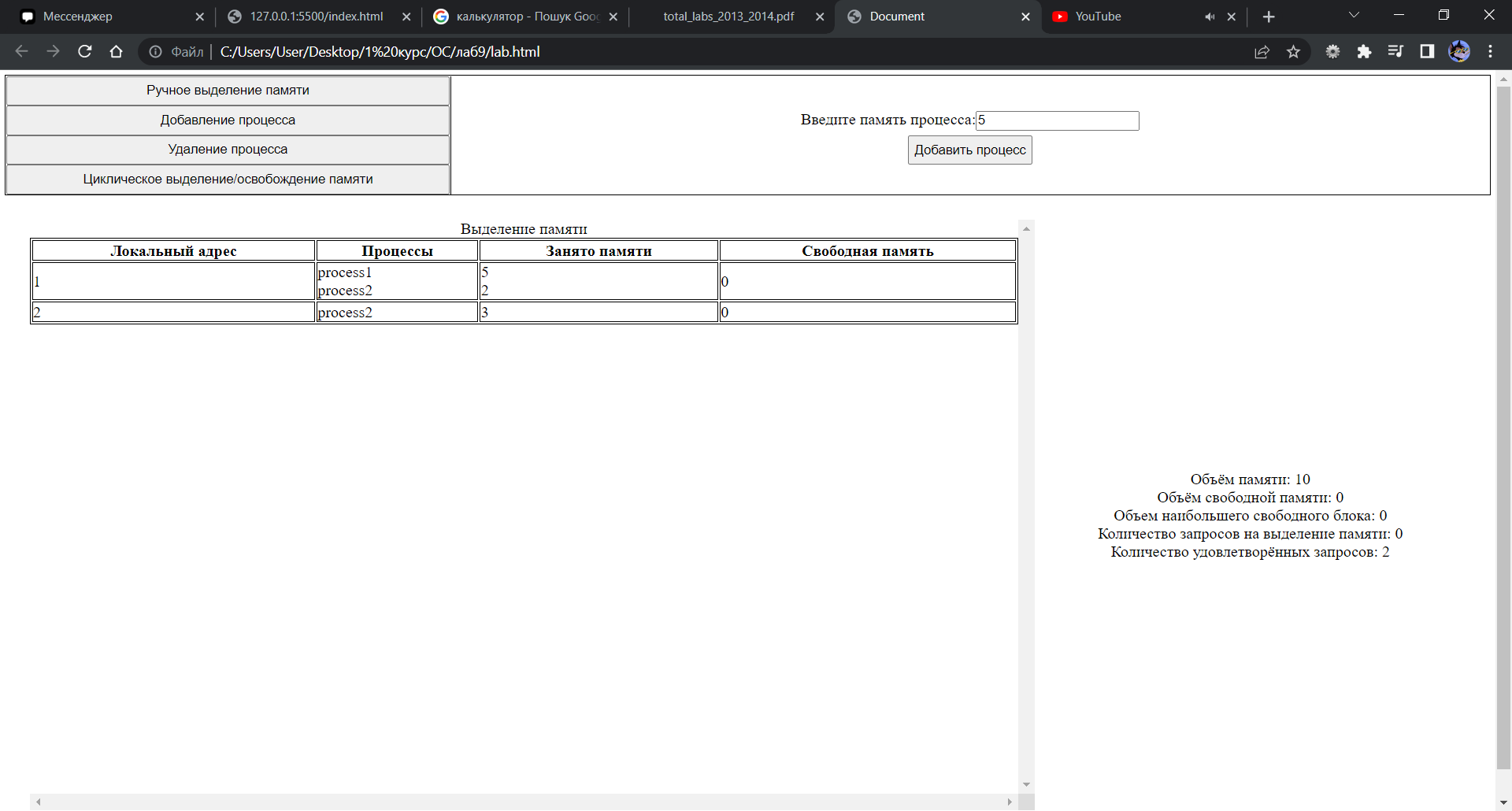


Рисунок 3 – Окно программы

На рисунке 4 показано удаление всех процессов из заданного блока памяти.

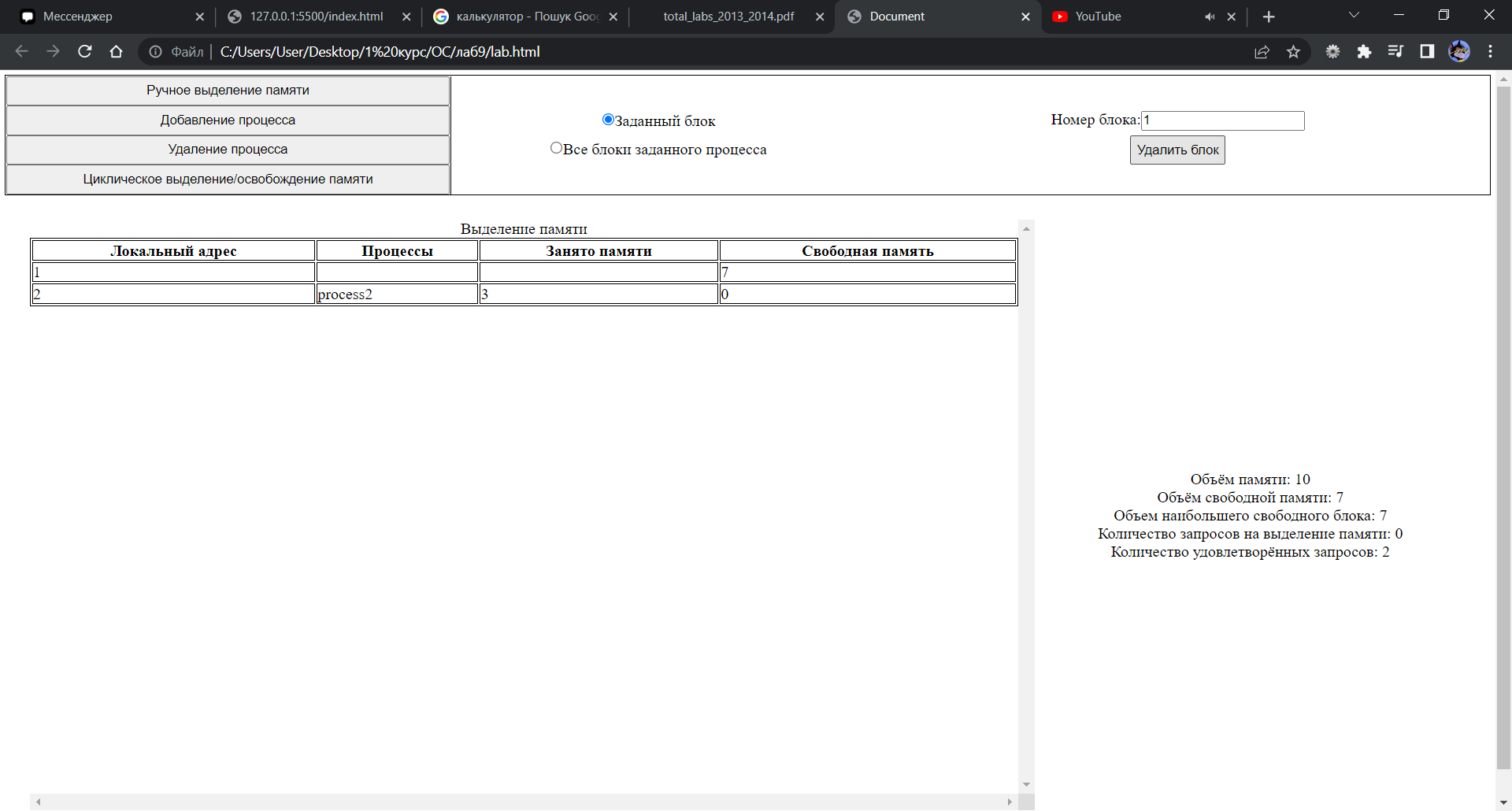


Рисунок 4 – Окно программы

На рисунке 5 показано удаление всех процесса из всех блоков памяти.

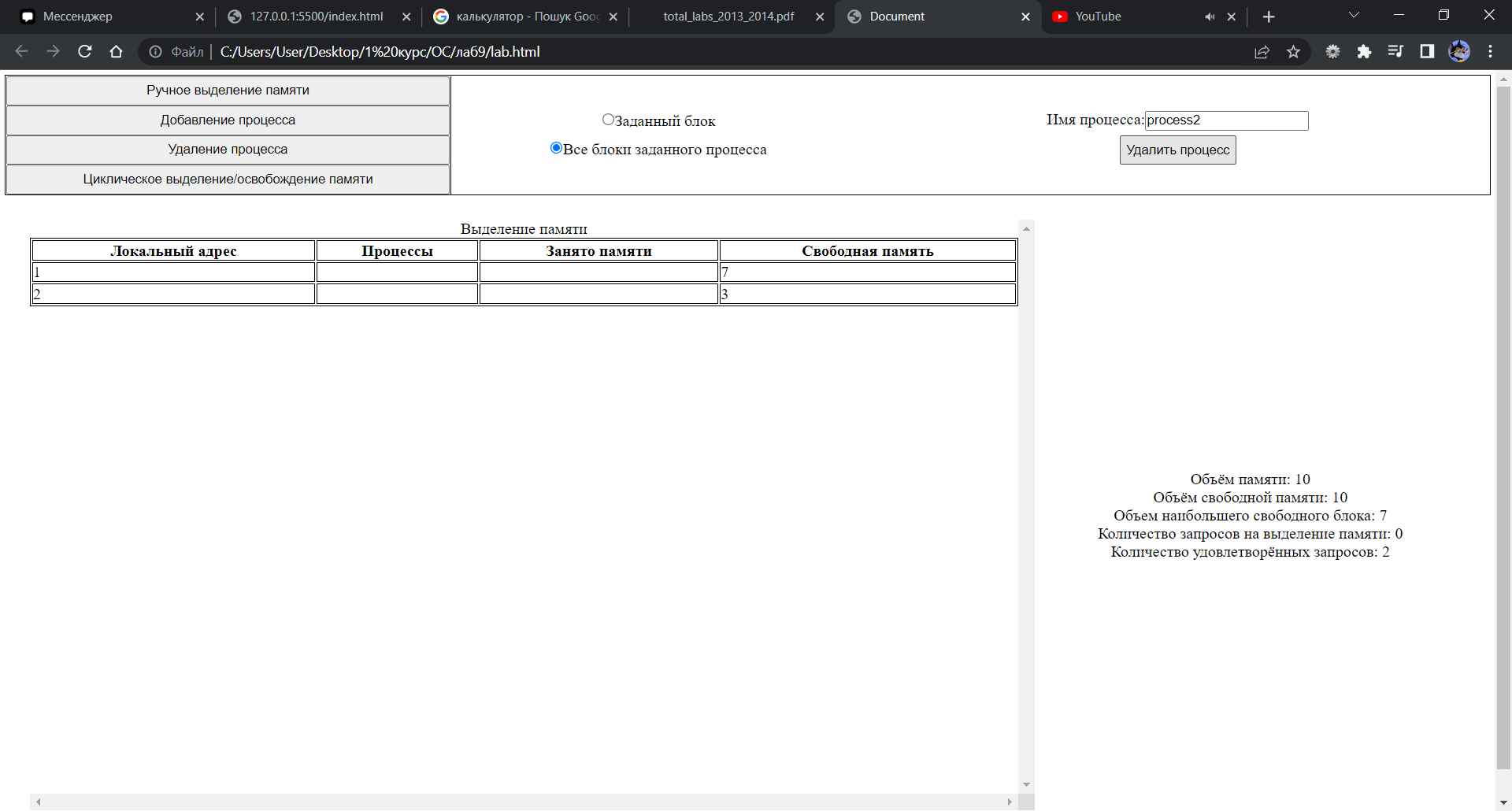


Рисунок 5 – Окно программы

На рисунке 6 показано циклическое выделение памяти.

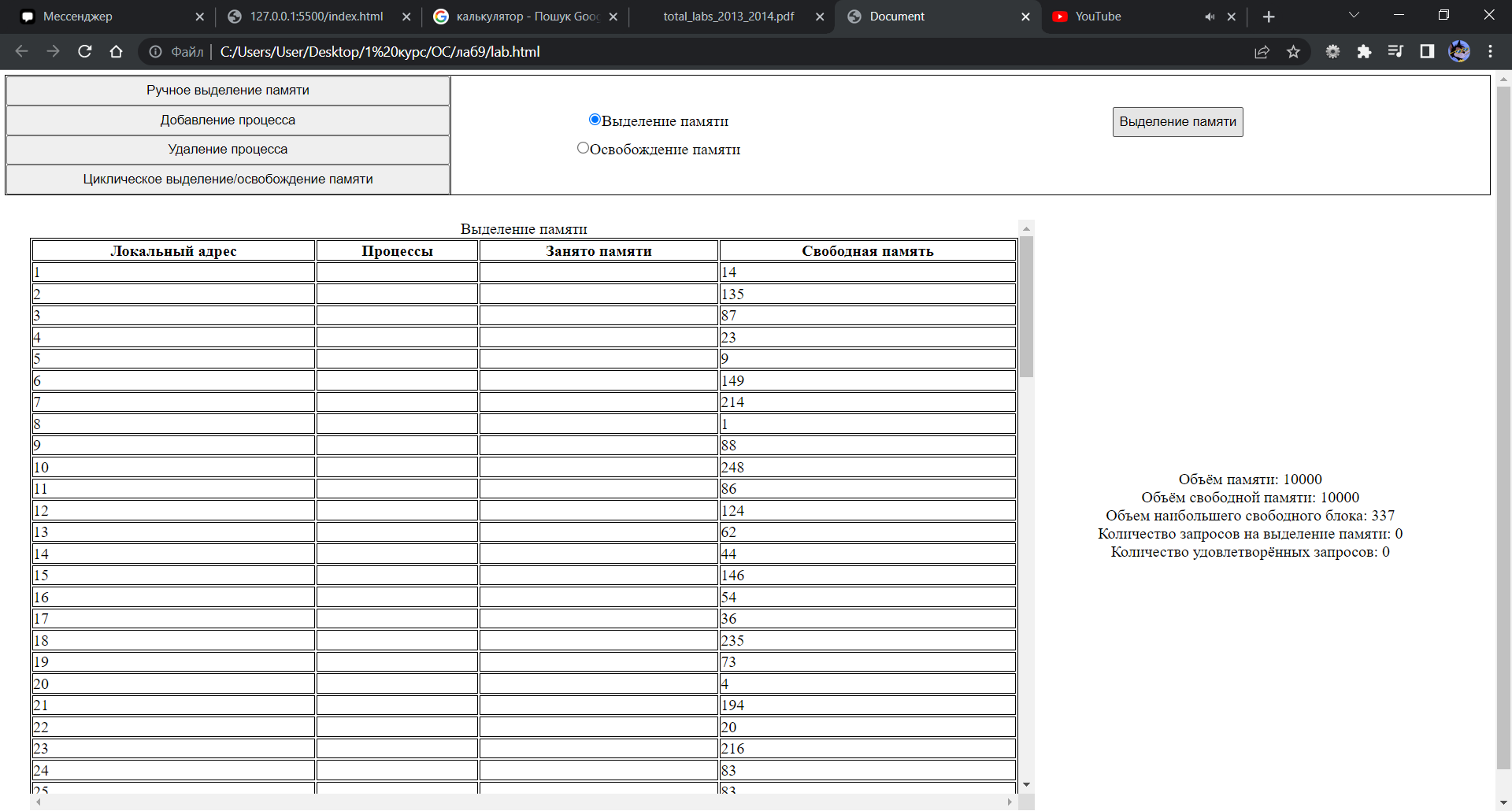


Рисунок 6 – Окно программы

На рисунке 7 показано циклическое освобождение памяти.

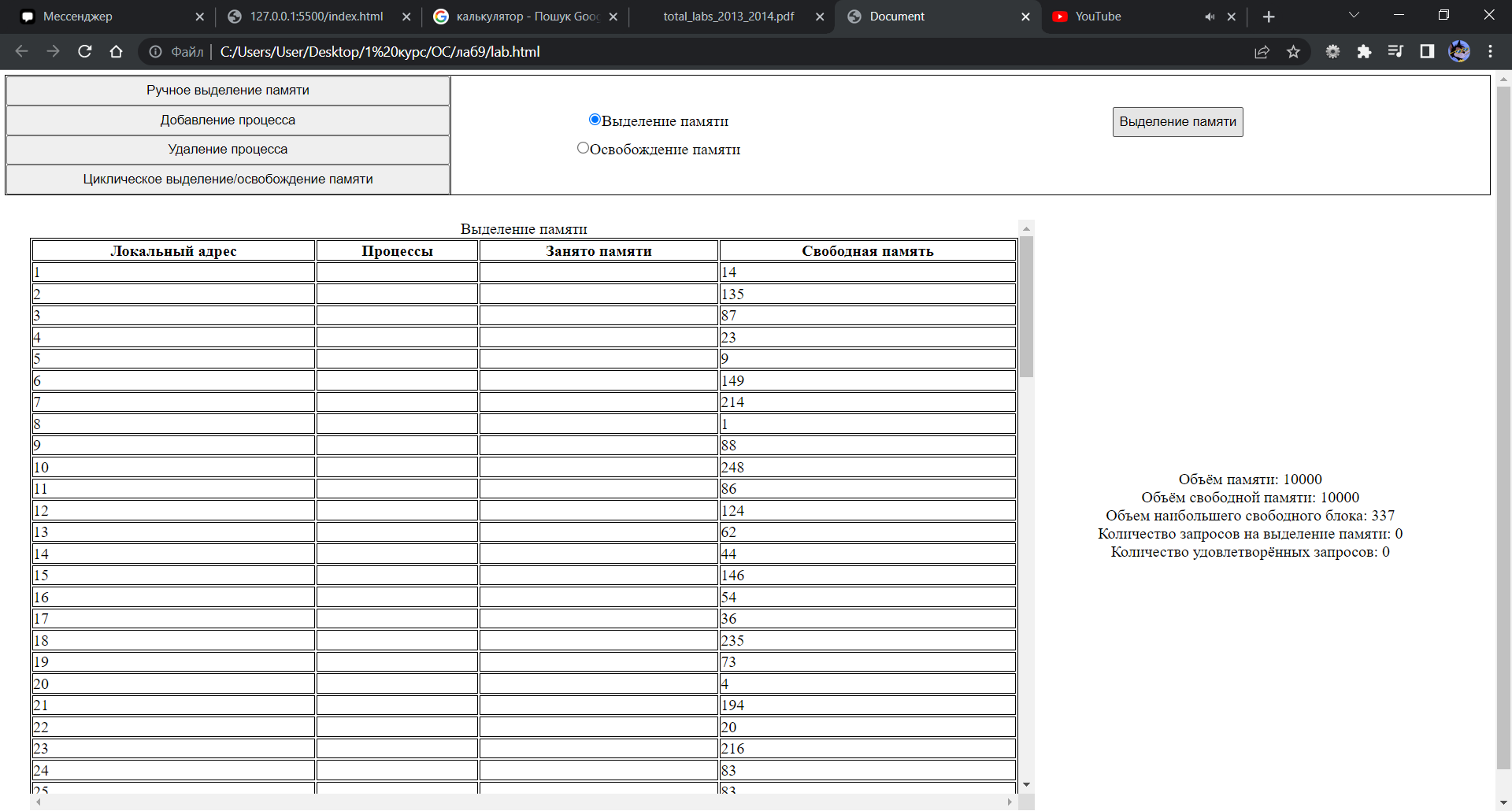


Рисунок 7 – Окно программы

**Вывод:** в ходе данной лабораторной работы были изучены алгоритмы управления памятью и разработана программа менеджера памяти.

**ПРИЛОЖЕНИЕ А**

var flagAction, memoryAndProcessData, localAdress, memorySize, pageCount, idProcess, processStack, countTrue, countFalse, maxBlockSize, totalMemory;

idProcess = 0;

countFalse = countTrue = 0;

processStack = [];

//cтруктура данных памяти

function TableData(localAdress, freeMemory){

    this.localAdress = localAdress;

    this.processName = [];

    this.processSize = [];

    this.freeMemory  = freeMemory;

}

//пользовательское выделение памяти и страниц

function userCustomSelection(){

    memorySize = parseInt(document.getElementById("user-memory").value); //размер памяти выделяемой пользователем

    pageCount  = parseInt(document.getElementById("user-pages").value);  //количество страниц выделяемых пользователем

    if(!isNaN(memorySize) && !isNaN(pageCount) && memorySize >= pageCount && pageCount > 0){

        memorySize = Math.floor(memorySize);

        pageCount  = Math.floor(pageCount);

        CustomSelection(memorySize, pageCount);

    }

    else{

        alert("Проверьте ввод данных(Размер выделяемой памяти должен быть > выделяемых страниц");

    }

}

//системное динамическое выделение памяти и страниц

function systemCustomSelection(){

    memorySize = 10000;

    pageCount  = 100;

    CustomSelection(memorySize,pageCount);

}

//метод выделения динамических страниц

function CustomSelection(memorySize, pageCount){

    var freeMemory, baseSize, reserve;

    totalMemory = 0;

    countTrue = 0;

    countFalse = 0;

    baseSize    = parseInt(Math.floor(memorySize/pageCount)); //базовый размер ячейки

    reserve     = 0; //остаток памяти после выделения памяти ячейки

    totalMemory = 0; //общая выделенная память

    localAdress = 1; //локальный адрес ячеек начинается с 1

    delete memoryAndProcessData; //очищение массива

    memoryAndProcessData = []; //объявление нового массива

    for(var i = 0; i < pageCount; i++)

    {

        freeMemory = Math.floor(Math.random() \* (baseSize + reserve)+1); //случайный размер памяти ячейки от 1 до (базовый размер + резервная память)

        reserve += baseSize - freeMemory; //расчёт резервной памяти

        memoryAndProcessData[i] = new TableData(localAdress,freeMemory); //создание структуры ячейки

        localAdress++; //увеличение локального адреса

        totalMemory += freeMemory; //подсчёт выделенной памяти

    }

    //сравнение выделенной памяти с заданной пользователем и распределение оставшейся случайному процессу

    if(parseInt(totalMemory) < parseInt(memorySize)){

        var randomProcess = Math.floor(Math.random() \* (pageCount));

        memoryAndProcessData[randomProcess].freeMemory += memorySize - totalMemory;

        totalMemory += memorySize - totalMemory;

    }

    createTable(); //создание визуальной таблицы

}

//метод создания визуальной таблицы

function createTable(){

    var row = column = columnP = columnText = table = caption = textCaption = "";

    table       = document.createElement('table'); //создание таблицы

    caption     = document.createElement('caption'); //создание заголовка таблицы

    textCaption = document.createTextNode('Выделение памяти'); //создание текстового узла заголовка

    caption.appendChild(textCaption);//добавление текстового узла заголовка в конец заголовка

    table.appendChild(caption);//добавление заголовка в конец таблицы

    row        = document.createElement('tr'); //создание строки с заголовками

    column     = document.createElement('th'); //создание ячейки заголовка локального адреса

    columnText = document.createTextNode('Локальный адрес'); //создание текстового узла заголовка локального адреса

    column.appendChild(columnText); //добавление текстового узла заголовка локального ареса в ячейку заголовка локального адреса

    row.appendChild(column); //добавление ячейки в строку

    column     = document.createElement('th');

    columnText = document.createTextNode('Процессы');

    column.appendChild(columnText);

    row.appendChild(column);

    column     = document.createElement('th');

    columnText = document.createTextNode('Занято памяти');

    column.appendChild(columnText);

    row.appendChild(column);

    column     = document.createElement('th');

    columnText = document.createTextNode('Свободная память');

    column.appendChild(columnText);

    row.appendChild(column);

    table.appendChild(row); // добавление строки в таблицу

    for(var i = 0; i < pageCount; i++)

    {

        row = document.createElement('tr');

        column     = document.createElement('td');

        columnP    = document.createElement('p');

        columnText = document.createTextNode(memoryAndProcessData[i].localAdress);

        columnP.appendChild(columnText);

        column.appendChild(columnP);

        row.appendChild(column);

        column = document.createElement('td');

        //вывод имени всех процессов содержащихся в ячейке памяти

        for(var j = 0; j < memoryAndProcessData[i].processName.length; j++)

        {

            columnP    = document.createElement('p');

            columnText = document.createTextNode(memoryAndProcessData[i].processName[j]);

            columnP.appendChild(columnText);

            column.appendChild(columnP);

        }

        row.appendChild(column);

        column = document.createElement('td');

        //вывод размера всех процессов содержащихся в ячейке памяти

        for(var j = 0; j < memoryAndProcessData[i].processSize.length; j++)

        {

            columnP    = document.createElement('p');

            columnText = document.createTextNode(memoryAndProcessData[i].processSize[j]);

            columnP.appendChild(columnText);

            column.appendChild(columnP);

        }

        row.appendChild(column);

        column     = document.createElement('td');

        columnP    = document.createElement('p');

        columnText = document.createTextNode(memoryAndProcessData[i].freeMemory);

        columnP.appendChild(columnText);

        column.appendChild(columnP);

        row.appendChild(column);

        table.appendChild(row);

    }

    document.getElementById("table-block").innerHTML = ""; //очистка поля вывода таблицы

    document.getElementById("table-block").appendChild(table);

    info();

}

function info(){

    maxSize = memoryAndProcessData[0].freeMemory;

    for(var i = 1; i < pageCount; i++)

    {

        if(memoryAndProcessData[i].freeMemory > maxSize)

            maxSize = memoryAndProcessData[i].freeMemory;

    }

    document.getElementById("size").textContent = "Объём памяти: "                                   + memorySize;

    document.getElementById("free-size").textContent = "Объём свободной памяти: "                    + totalMemory;

    document.getElementById("max-size").textContent = "Объем наибольшего свободного блока: "         + maxSize;

    document.getElementById("count-false").textContent = "Количество запросов на выделение памяти: " + countFalse;

    document.getElementById("count-true").textContent = "Количество удовлетворённых запросов: "      + countTrue;

}

function addProcess(){

    var processSize = parseInt(document.getElementById("process-add").value);

    if(!isNaN(processSize) && processSize > 0){

        idProcess++;

        processName = "process"+idProcess;

        if (totalMemory >= processSize){

            addTrueBlock(processName, processSize);

        }

        else{

            var index = processStack.length;

            processStack[index] = new Array();

            processStack[index][0] = processName;

            processStack[index][1] = processSize;

            countFalse++;

        }

        createTable();

    }

    else

        alert("Проверьте размерность процесса");

}

function addTrueBlock(processName, processSize){

    for(var i = 0; i < pageCount && processSize > 0; i++)

            {

                var index = memoryAndProcessData[i].processName.length;

                if(processSize >= memoryAndProcessData[i].freeMemory && memoryAndProcessData[i].freeMemory !== 0)

                {

                    memoryAndProcessData[i].processName[index] = processName;

                    memoryAndProcessData[i].processSize[index] = memoryAndProcessData[i].freeMemory;

                    processSize -= memoryAndProcessData[i].freeMemory;

                    totalMemory -= memoryAndProcessData[i].freeMemory;

                    memoryAndProcessData[i].freeMemory = 0;

                }

                else

                {

                    if(memoryAndProcessData[i].freeMemory !== 0)

                    {

                        memoryAndProcessData[i].processName[index] = processName;

                        memoryAndProcessData[i].processSize[index] = processSize;

                        memoryAndProcessData[i].freeMemory -= processSize;

                        totalMemory -= processSize;

                        processSize = 0;

                    }

                }

            }

            countTrue++;

}

function deleteBlock(){

    var idBlock =  parseInt(document.getElementById("block-number-delete").value);

    if(idBlock > 0 && idBlock <= memoryAndProcessData.length)

    {

        var plusMemory = 0;

        delete memoryAndProcessData[idBlock-1].processName;

        memoryAndProcessData[idBlock-1].processName = [];

        while(memoryAndProcessData[idBlock-1].processSize.length != 0)

        {

            plusMemory += memoryAndProcessData[idBlock-1].processSize[0];

            memoryAndProcessData[idBlock-1].processSize.splice(0,1);

        }

        memoryAndProcessData[idBlock-1].freeMemory += plusMemory;

        totalMemory += plusMemory;

        stackCheck();

        createTable();

    }

    else

        alert("Такого блока не существует")

}

function deleteProcess()

{

    var plusMemory,j;

    var nameProcess = document.getElementById("name-process-delete").value

    for(var i = 0; i < memoryAndProcessData.length; i++)

    {

        plusMemory = 0;

        j = 0;

        while(j < memoryAndProcessData[i].processName.length )

        {

            if(nameProcess == memoryAndProcessData[i].processName[j])

            {

                memoryAndProcessData[i].processName.splice(j,1);

                plusMemory += memoryAndProcessData[i].processSize[j];

                memoryAndProcessData[i].processSize.splice(j,1);

                j--;

            }

            j++

        }

        memoryAndProcessData[i].freeMemory += plusMemory;

        totalMemory += plusMemory;

    }

    stackCheck();

    createTable();

}

function stackCheck()

{

    var i = 0;

    while(i < processStack.length)

    {

        if(totalMemory >= processStack[i][1]){

            addTrueBlock(processStack[i][0], processStack[i][1]);

            processStack.splice(i,1)

            countFalse--;

            i--;

        }

        i++;

    }

}

function freeMemory(){

    idProcess = 0;

    countTrue = 0;

    countFalse = 0;

    delete processStack;

    processStack = [];

    for(var i = 0; i < memoryAndProcessData.length; i++)

    {

        plusMemory = 0;

        delete memoryAndProcessData[i].processName;

        memoryAndProcessData[i].processName = [];

        while(memoryAndProcessData[i].processSize.length != 0)

        {

            plusMemory += memoryAndProcessData[i].processSize[0];

            memoryAndProcessData[i].processSize.splice(0,1);

        }

        memoryAndProcessData[i].freeMemory += plusMemory;

        totalMemory += plusMemory;

    }

    createTable();

}

//скрытие и очистка всех робочих областей и полей

function hide(){

    document.getElementById("custom-selection").style.display = "none";

    document.getElementById("user-memory").value = "";

    document.getElementById("user-pages").value = "";

    document.getElementById("add-process").style.display = "none";

    document.getElementById("process-add").value = "";

    document.getElementById("delete-process").style.display = "none";

    document.getElementById("specified-block").style.display = "none";

    document.getElementById("block-number-delete").value = "";

    document.getElementById("all-blocks-process").style.display = "none";

    document.getElementById("name-process-delete").value = "";

    document.getElementById("specified-block-radio").checked = false;

    document.getElementById("all-blocks-process-radio").checked = false;

    document.getElementById("cyclic-selection").style.display = "none";

    document.getElementById("memory-allocation").style.display = "none";

    document.getElementById("memory-free").style.display = "none";

    document.getElementById("memory-allocation-radio").checked = false;

    document.getElementById("memory-free-radio").checked = false;

}

//появление первого пункта меню

function showCustomSelection(){

    document.getElementById("custom-selection").style.display = "flex";

}

//появление второго пункта меню

function showAddProcess(){

    document.getElementById("add-process").style.display = "flex";

}

//появление третьего пункта меню

function showDeleteProcess(){

    document.getElementById("delete-process").style.display = "flex";

}

//появление четвертого пункта меню

function showCyclicSelection(){

    document.getElementById("cyclic-selection").style.display = "flex";

}

//появление блока с удалением в зависимости от выбора пользователя

function showDeleteBlocks(){

    if(document.getElementById("specified-block-radio").checked){

        document.getElementById("specified-block").style.display = "flex";

        document.getElementById("all-blocks-process").style.display = "none";

    }

    if(document.getElementById("all-blocks-process-radio").checked){

        document.getElementById("all-blocks-process").style.display = "flex";

        document.getElementById("specified-block").style.display = "none";

    }

}

//появление блока с линамическим выделением/освобождением

function showCyclicSelectionBlocks(){

    if(document.getElementById("memory-allocation-radio").checked){

        document.getElementById("memory-allocation").style.display = "flex";

        document.getElementById("memory-free").style.display = "none";

    }

    if(document.getElementById("memory-free-radio").checked){

        document.getElementById("memory-free").style.display = "flex";

        document.getElementById("memory-allocation").style.display = "none";

    }

}

//присваивание событий кнопке первого пункта меню

var menuButton1 = document.getElementById("menu-button-1");

if(menuButton1){

    menuButton1.addEventListener('click',hide,false);

    menuButton1.addEventListener('click',showCustomSelection,false);

}

//присваивание событий кнопке второго пункта меню

var menuButton2 = document.getElementById("menu-button-2");

if(menuButton2){

    menuButton2.addEventListener('click',hide,false);

    menuButton2.addEventListener('click',showAddProcess,false);

}

//присваивание событий кнопке третьего пункта меню

var menuButton3 = document.getElementById("menu-button-3");

if(menuButton3){

    menuButton3.addEventListener('click',hide,false);

    menuButton3.addEventListener('click',showDeleteProcess,false);

}

//присваивание событий кнопке четвертого пункта меню

var menuButton4 = document.getElementById("menu-button-4");

if(menuButton4){

    menuButton4.addEventListener('click',hide,false);

    menuButton4.addEventListener('click',showCyclicSelection,false);

}

//присваивание событий переключателям пункта удаления

document.getElementById("specified-block-radio").addEventListener('change',showDeleteBlocks,false);

document.getElementById("all-blocks-process-radio").addEventListener('change',showDeleteBlocks,false);

//присваивание событий переключателям пункта динаимеского выделения/освобождения

document.getElementById("memory-allocation-radio").addEventListener('change',showCyclicSelectionBlocks,false);

document.getElementById("memory-free-radio").addEventListener('change',showCyclicSelectionBlocks,false);

//присваивание событий кнопке пользовательского распределения

var buttonUserGive = document.getElementById("button-user-give");

if(buttonUserGive){

    buttonUserGive.addEventListener('click',userCustomSelection,false);

}

//присваивание событий кнопке динамического распределения

var buttonMemoryAllocation = document.getElementById("button-memory-allocation");

if(buttonMemoryAllocation){

    buttonMemoryAllocation.addEventListener('click',systemCustomSelection,false);

}

var buttonProcessAdd = document.getElementById("button-process-add");

if(buttonProcessAdd){

    buttonProcessAdd.addEventListener('click',addProcess,false);

}

var buttonDeleteBlock = document.getElementById("button-delete-block");

if(buttonDeleteBlock){

    buttonDeleteBlock.addEventListener('click',deleteBlock,false);

}

var buttonDeleteProcess = document.getElementById("button-delete-process");

if(buttonDeleteProcess){

    buttonDeleteProcess.addEventListener('click',deleteProcess,false);

}

var buttonMemoryFree = document.getElementById("button-memory-free");

if(buttonMemoryFree){

    buttonMemoryFree.addEventListener('click',freeMemory,false);

}