**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**

по дисциплине «Операционные системы»

на тему: «Управление виртуальной памятью»

Выполнила: студентка гр. ИТП-11

Бородина Е. И.

Принял: преподаватель-стажёр Карась О.В.

Гомель 2022

**Цель:** изучить основные алгоритмы замещение страниц и работой с виртуальной памятью.

**Задача:**

Разработать программу, реализующую заданный алгоритм замещения страницы в памяти. Менеджер памяти должен:

1. Разбивать память заданного размера на указанное количество страниц. На экран должна выводится следующая информация о состоянии памяти: объем памяти, число страниц, число свободных страниц (%), размер страницы.
2. 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). Например: Pro 3. После нажатия на кнопку «ДОБАВИТЬ» страница размещается в свободной странице памяти. Если задано глобальное размещение (см. вариант задания), то выбирается любая не занятая страница. При локальном размещении страница размещается только 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.
3. Удалять из памяти заданную страницу или все страницы заданного процесса (по нажатию кнопки «УДАЛИТЬ»). Указывается номер удаляемой страницы в памяти.

Организовывать циклическое обращение к страницам, размещенным в памяти по нажатию на кнопку. При этом случайным образом задается количество обращений к страницам (диапазон 1...10). Для каждого обращения генерируется, случайным образом, номер страницы из диапазона [0; количество страниц памяти]. При обращении к странице в зависимости, от варианта, увеличивается ее внутренний счетчик обращений или устанавливается флаг обращения.

Вариант 4:

Алгоритм замещения – *LRU CLOCK*. Существует глобальный счетчик обращений к страницам. При каждом обращении к станице в ее внутренний регистр заносится значение глобального счетчика. Выгружается страница с наименьшим значением счетчика.

**Ход работы:**

Алгоритм Lru Clock замещает страницу, которая не использовалась в течение самого долгого промежутка времени. При запуске программы появляется меню, где пользователь может выбрать какое количество страниц памяти, и какой объем памяти может использовать пользователь. Пример меню выбора страницы и памяти представлен на рисунке 1.

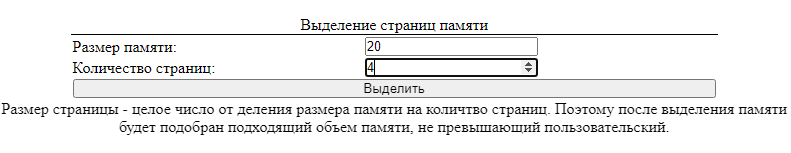


Рисунок 1 – Меню выбора страниц и памяти

На экране, в графе «INFO», показывается вся информация, которую ввел пользователь, и меню, где пользователь может добавить страницу в память, удалить страницу. После нажатия на кнопку «добавить», появляется поле для добавления процессов. Пример главного меню программы представлен на рисунке 2.

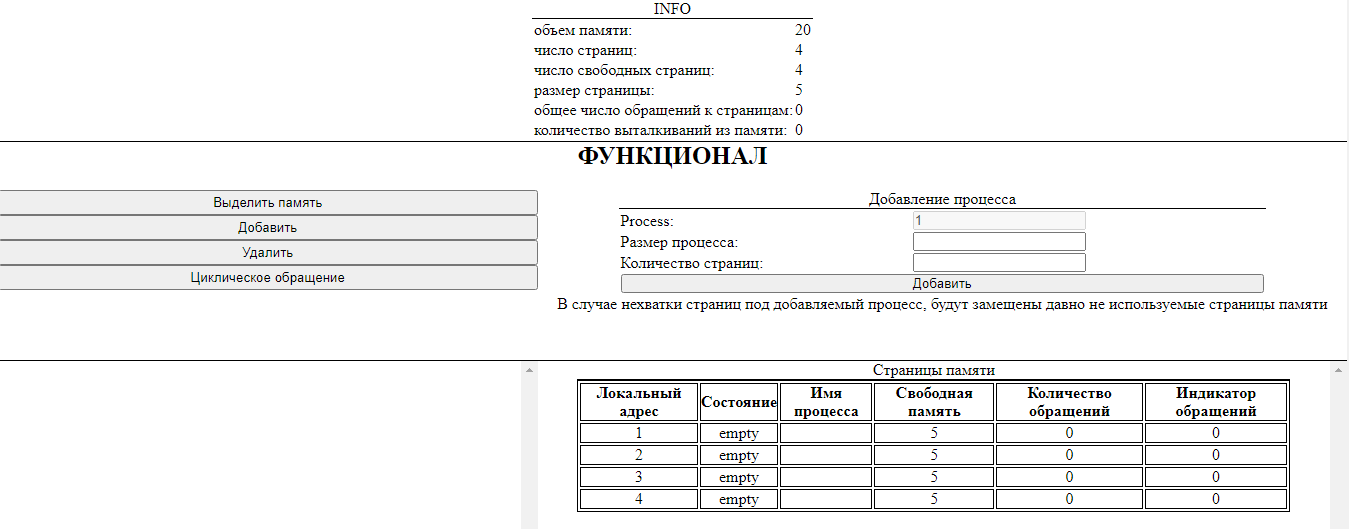


Рисунок 2 – Главное меню программы

После добавления процессов начинается выполнение алгоритма. Результат выполнения алгоритма и характеристика процессов при завершении выполнения алгоритма представлены на рисунке 3.

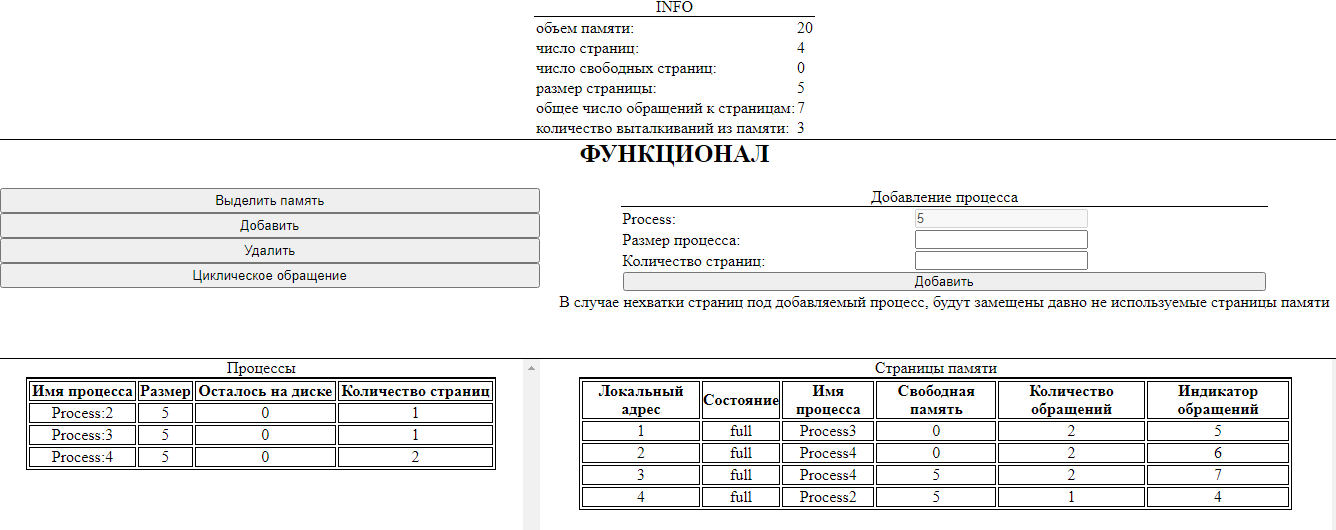


Рисунок 3 – Результат выполнения алгоритма

После его выполнения мы можем удалить некоторые процессы. Результат удаления процесса представлен на рисунке 4.

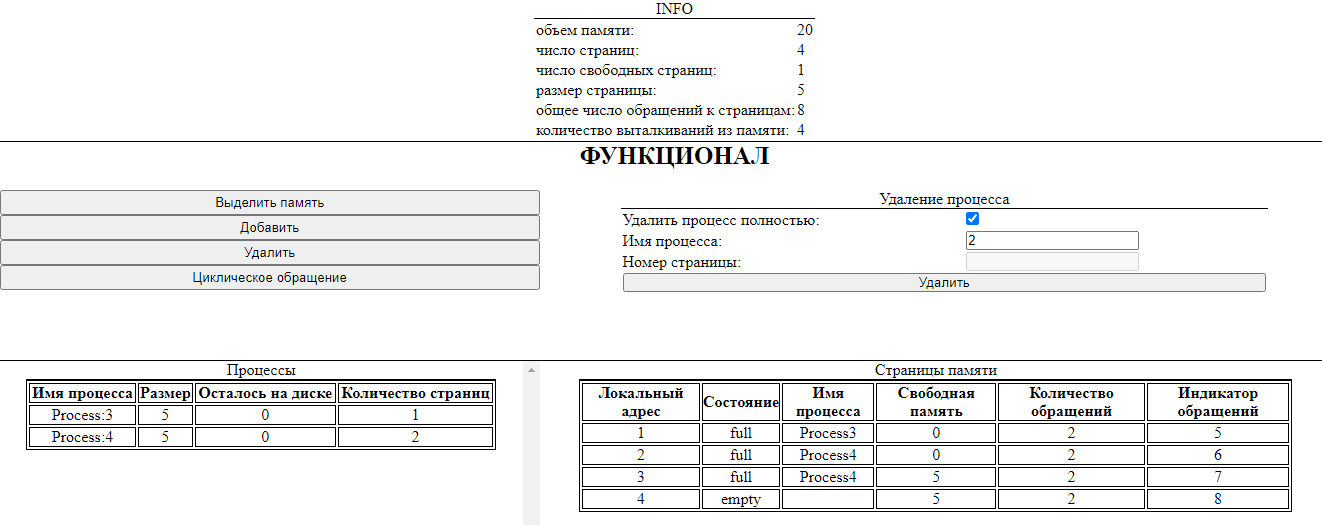


Рисунок 4 – Удаление второго процесса из программы

Вывод: изучена концепция виртуальной памяти и алгоритмы замещения страниц. Так же разработана программа, реализующая алгоритм замещения памяти «*LRU*».

Листинг программы:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Document</title>

<link rel="stylesheet" href="style.css">

<script>var memorySize = 0, pageCount = 0, freePageCount = 0, pageSize = 0, pageAccess = 0, pagePush = 0, time = 1, processIdIndex = 1;

var tableDataMain = [];

var tableProcess = [];

function TableDataMain(localAdress, size){

this.localAdress = localAdress;

this.status = "empty";

this.processId = "";

this.size = size;

this.countAcess = 0;

this.time = 0;

}

function ProcessDataInfo(id, size, count){

this.processId = id;

this.size = size;

this.residueSize = size;

this.countPage = count;

}

function giveMemoryShow(){

hide();

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

}

function giveMemory(){

if(isFinite(parseInt(document.getElementById("memory-size-input").value)) && parseInt(document.getElementById("memory-size-input").value) > 0)

if(isFinite(parseInt(document.getElementById("page-count-input").value)) && parseInt(document.getElementById("page-count-input").value) > 0)

if(parseInt(document.getElementById("memory-size-input").value) >= parseInt(document.getElementById("page-count-input").value)){

memorySize = parseInt(document.getElementById("memory-size-input").value);

pageCount = parseInt(document.getElementById("page-count-input").value);

document.getElementById("memory-size-input").value = "";

document.getElementById("page-count-input").value = "";

freePageCount = pageCount;

pageSize = Math.floor(memorySize/pageCount);

memorySize = pageSize \* pageCount;

createTableMain();

delete tableProcess;

tableProcess = [];

pageAccess = 0;

pagePush = 0;

time = 1;

processIdIndex = 1;

document.getElementById("main-info").innerHTML = "";

document.getElementById("btn1").disabled = false;

document.getElementById("btn2").disabled = false;

document.getElementById("btn3").disabled = false;

}

else alert("Размер памяти должен быть больше или равен количеству страниц");

else alert("Количество страниц должно быть целым положительным числом");

else alert("Размер памяти должен быть целым положительным числом");

}

function createTableMain(){

delete tableDataMain;

tableDataMain = [];

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

{

tableDataMain[i] = new TableDataMain(i+1,pageSize)

}

printTableMain();

}

function printTableMain(){

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

table = document.createElement('table');

table.classList.add("width90");

table.classList.add("tableCentre");

caption = document.createElement('caption');

textCaption = document.createTextNode('Страницы памяти');

caption.appendChild(textCaption);

table.appendChild(caption);

row = document.createElement('tr');

column = document.createElement('th');

columnText = document.createTextNode('Локальный адрес');

column.appendChild(columnText);

row.appendChild(column);

column = document.createElement('th');

columnText = document.createTextNode('Состояние');

column.appendChild(columnText);

row.appendChild(column);

column = document.createElement('th');

columnText = document.createTextNode('Имя процесса');

column.appendChild(columnText);

row.appendChild(column);

column = document.createElement('th');

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

column.appendChild(columnText);

row.appendChild(column);

column = document.createElement('th');

columnText = document.createTextNode('Количество обращений');

column.appendChild(columnText);

row.appendChild(column);

column = document.createElement('th');

columnText = document.createTextNode('Индикатор обращений');

column.appendChild(columnText);

row.appendChild(column);

table.appendChild(row);

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

{

row = document.createElement('tr');

column = document.createElement('td');

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

column.appendChild(columnText);

row.appendChild(column);

column = document.createElement('td');

columnText = document.createTextNode(tableDataMain[i].status);

column.appendChild(columnText);

row.appendChild(column);

column = document.createElement('td');

if(tableDataMain[i].status === 'empty') columnText = document.createTextNode(tableDataMain[i].processId);

else columnText = document.createTextNode('Process'+tableDataMain[i].processId);

column.appendChild(columnText);

row.appendChild(column);

column = document.createElement('td');

columnText = document.createTextNode(tableDataMain[i].size);

column.appendChild(columnText);

row.appendChild(column);

column = document.createElement('td');

columnText = document.createTextNode(tableDataMain[i].countAcess);

column.appendChild(columnText);

row.appendChild(column);

column = document.createElement('td');

columnText = document.createTextNode(tableDataMain[i].time);

column.appendChild(columnText);

row.appendChild(column);

table.appendChild(row);

}

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

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

updateInfo();

}

function addProcessShow(){

hide();

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

}

function addProcess(){

if(isFinite(parseInt(document.getElementById("add-process-size-input").value)) && parseInt(document.getElementById("add-process-size-input").value) > 0)

if(isFinite(parseInt(document.getElementById("add-page-count-input").value)) && parseInt(document.getElementById("add-page-count-input").value) > 0 && parseInt(document.getElementById("add-page-count-input").value) <= pageCount) {

var processId = parseInt(document.getElementById("add-process-name-input").value);

processIdIndex++;

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

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

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

var processCountPage = parseInt(document.getElementById("add-page-count-input").value);

document.getElementById("add-page-count-input").value = "";

createTableMainInfo(processId, processSize, processCountPage);

}

else alert("Размер процесса должен быть целым положительным числом");

else alert("Количество выделяемых страниц должно быть целым положительным числом и не должно превышать общее количество страниц");

}

function createTableMainInfo(processId, processSize, processCountPage){

tableProcess[tableProcess.length] = new ProcessDataInfo(processId,processSize,processCountPage);

var whileIndex = processCountPage, indexPage, min;

while(whileIndex)

{

indexPage = -1;

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

if(tableDataMain[i].status === "empty"){

indexPage = i;

break;

}

if(indexPage == -1){

min = tableDataMain[0].time;

indexPage = 0;

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

if(min > tableDataMain[i].time){

min = tableDataMain[i].time;

indexPage = i;

}

}

if(indexPage != -1){

if(tableDataMain[indexPage].status == "full")

{

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

{

if(parseInt(tableDataMain[indexPage].processId) == parseInt(tableProcess[i].processId))

{

tableProcess[i].countPage -= 1;

if(tableProcess[i].countPage == 0)

tableProcess.splice(i,1);

tableDataMain[indexPage].size = pageSize;

freePageCount++

pagePush++;

break;

}

}

}

tableDataMain[indexPage].status = "full";

tableDataMain[indexPage].processId = tableProcess[tableProcess.length-1].processId;

tableDataMain[indexPage].countAcess++;

tableDataMain[indexPage].time = time++;

pageAccess++;

freePageCount--;

if(tableProcess[tableProcess.length-1].residueSize)

if(tableDataMain[indexPage].size >= tableProcess[tableProcess.length-1].residueSize) {

tableDataMain[indexPage].size -= tableProcess[tableProcess.length-1].residueSize;

tableProcess[tableProcess.length-1].residueSize -= tableProcess[tableProcess.length-1].residueSize

}

else {

tableProcess[tableProcess.length-1].residueSize -= tableDataMain[indexPage].size

tableDataMain[indexPage].size -= tableDataMain[indexPage].size;

}

whileIndex--;

}

}

updateInfo();

printTableMain();

printTableMainInfo();

}

function printTableMainInfo(){

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

table = document.createElement('table');

table.classList.add("width90");

table.classList.add("tableCentre");

caption = document.createElement('caption');

textCaption = document.createTextNode('Процессы');

caption.appendChild(textCaption);

table.appendChild(caption);

row = document.createElement('tr');

column = document.createElement('th');

columnText = document.createTextNode('Имя процесса');

column.appendChild(columnText);

row.appendChild(column);

column = document.createElement('th');

columnText = document.createTextNode('Размер');

column.appendChild(columnText);

row.appendChild(column);

column = document.createElement('th');

columnText = document.createTextNode('Осталось на диске');

column.appendChild(columnText);

row.appendChild(column);

column = document.createElement('th');

columnText = document.createTextNode('Количество страниц');

column.appendChild(columnText);

row.appendChild(column);

table.appendChild(row);

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

{

row = document.createElement('tr');

column = document.createElement('td');

columnText = document.createTextNode("Process:"+tableProcess[i].processId);

column.appendChild(columnText);

row.appendChild(column);

column = document.createElement('td');

columnText = document.createTextNode(tableProcess[i].size);

column.appendChild(columnText);

row.appendChild(column);

column = document.createElement('td');

columnText = document.createTextNode(tableProcess[i].residueSize);

column.appendChild(columnText);

row.appendChild(column);

column = document.createElement('td');

columnText = document.createTextNode(tableProcess[i].countPage);

column.appendChild(columnText);

row.appendChild(column);

table.appendChild(row);

}

document.getElementById("main-info").innerHTML = ""; //очистка поля вывода таблицы

document.getElementById("main-info").appendChild(table);

updateInfo();

}

function deleteProcessShow(){

hide();

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

}

function deleteProcess(){

if(isFinite(parseInt(document.getElementById("delete-name-process-input").value)) && parseInt(document.getElementById("delete-name-process-input").value) > 0){

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

if(document.getElementById("delete-all-page-checkbox").checked){

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

{

if(tableDataMain[i].processId == name){

tableDataMain[i].size = pageSize;

tableDataMain[i].countAcess++;

tableDataMain[i].time = time++;

tableDataMain[i].processId = "";

tableDataMain[i].status = "empty";

freePageCount++;

pageAccess++;

pagePush++;

}

}

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

if(tableProcess[i].processId == name){

tableProcess.splice(i,1);

break;

}

}

updateInfo();

printTableMain();

printTableMainInfo();

}

else {

if(isFinite(parseInt(document.getElementById("delete-page-number-input").value)) && parseInt(document.getElementById("delete-page-number-input").value) > 0 && parseInt(document.getElementById("delete-page-number-input").value) <= pageCount){

var num = parseInt(document.getElementById("delete-page-number-input").value);

if(tableDataMain[num-1].processId == name) {

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

{

if(tableProcess[i].processId == name){

tableDataMain[num-1].size = pageSize;

tableDataMain[num-1].countAcess++;

tableDataMain[num-1].time = time++;

pageAccess++;

pagePush++;

if(tableProcess[i].residueSize){

if(tableDataMain[num-1].size >= tableProcess[i].residueSize) {

tableDataMain[num-1].size -= tableProcess[i].residueSize;

tableProcess[i].residueSize -= tableProcess[i].residueSize

}

else {

tableProcess[i].residueSize -= tableDataMain[num-1].size

tableDataMain[num-1].size -= tableDataMain[num-1].size;

}

}

else{

tableDataMain[num-1].processId = "";

tableDataMain[num-1].status = "empty";

tableProcess[i].countPage--;

freePageCount++;

if(tableProcess[i].countPage == 0)

tableProcess.splice(i,1);

}

updateInfo();

printTableMain();

printTableMainInfo();

}

}

}

else alert("Процесса с таким именем на данной странице не обнаружено");

}

else alert("Проверьте номер страницы");

}

}

else alert("Id процесса должно быть целым положительным числом");

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

document.getElementById("delete-name-process-input").value

}

function checkboxChange(){

if(document.getElementById("delete-all-page-checkbox").checked)

document.getElementById("delete-page-number-input").disabled = true;

else

document.getElementById("delete-page-number-input").disabled = false;

}

function updateInfo(){

document.getElementById("memory-size").innerText = memorySize;

document.getElementById("page-count").innerText = pageCount;

document.getElementById("page-free-count").innerText = freePageCount;

document.getElementById("page-size").innerText = pageSize;

document.getElementById("request-count").innerText = pageAccess;

document.getElementById("push-count").innerText = pagePush;

}

function access(){

var index;

for(var i = 0; i < Math.floor(Math.random()\*10 + 1); i++){

index = Math.floor(Math.random()\*pageCount);

tableDataMain[index].countAcess++;

tableDataMain[index].time = time++;

pageAccess++;

updateInfo();

printTableMain();

}

}

function clear(){

document.getElementById("memory-size-input").value = "";

document.getElementById("page-count-input").value = "";

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

document.getElementById("add-page-count-input").value = "";

document.getElementById("delete-all-page-checkbox").checked = false;

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

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

}

function hide(){

clear();

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

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

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

}

</script>

</head>

<body>

<div id = "info" class = "width100 flex-display flex-centre">

<table id = "info-table">

<caption>INFO</caption>

<tr>

<td>объем памяти:</td><td id = "memory-size">10</td>

</tr>

<tr>

<td>число страниц:</td><td id = "page-count">100</td>

</tr>

<tr>

<td>число свободных страниц:</td><td id = "page-free-count">1000</td>

</tr>

<tr>

<td>размер страницы:</td><td id = "page-size">10000000000000000</td>

</tr>

<tr>

<td>общее число обращений к страницам:</td><td id = "request-count">10000</td>

</tr>

<tr>

<td>количество выталкиваний из памяти:</td><td id = "push-count"> 100000</td>

</tr>

</table>

</div>

<h1>ФУНКЦИОНАЛ</h1>

<div id = "menu" class = "width100 flex-display flex-row">

<div id = "menu-block" class = "width40 flex-display flex-column">

<button class = "button button100 button-menu" onclick="giveMemoryShow()">Выделить память</button>

<button class = "button button100 button-menu" onclick="addProcessShow()" id = "btn1" disabled = "true">Добавить</button>

<button class = "button button100 button-menu" onclick="deleteProcessShow()" id = "btn2" disabled = "true">Удалить</button>

<button class = "button button100 button-menu" onclick="access()" id = "btn3" disabled = "true">Циклическое обращение</button>

</div>

<div id = "menu-block-show" class = "width60 flex-display flex-column">

<div id = "give-memory" class = "width100 flex-display flex-column flex-centre flex-item-centre">

<table class = "width80">

<caption>Выделение страниц памяти</caption>

<tr>

<td>Размер памяти:</td><td><input type = "number" min = "1" pattern="[0-9]" id = "memory-size-input"></td>

</tr>

<tr>

<td>Количество страниц:</td><td><input type = "number" min = "1" pattern="[0-9]" id = "page-count-input"></td>

</tr>

<tr>

<td colspan="2"><button id = "give-memory-button" class = "button button100" onclick="giveMemory()">Выделить</button></td>

</tr>

</table>

<p>Размер страницы - целое число от деления размера памяти на количтво страниц. Поэтому после выделения памяти будет подобран подходящий объем памяти, не превышающий пользовательский.</p>

</div>

<div id = "add-process" class = "width100 flex-display flex-column flex-centre flex-item-centre">

<table class = "width80">

<caption>Добавление процесса</caption>

<tr>

<td>Process:</td><td><input type = "number" min = "1" pattern="[0-9]" disabled = "true" value = "1" id = "add-process-name-input"></td>

</tr>

<tr>

<td>Размер процесса:</td><td><input type = "number" min = "1" pattern="[0-9]" id = "add-process-size-input"></td>

</tr>

<tr>

<td>Количество страниц:</td><td><input type = "number" min = "1" pattern="[0-9]" id = "add-page-count-input"></td>

</tr>

<tr>

<td colspan="2"><button id = "add-process-button" class = "button button100" onclick = "addProcess()">Добавить</button></td>

</tr>

</table>

<p>В случае нехватки страниц под добавляемый процесс, будут замещены давно не используемые страницы памяти</p>

</div>

<div id = "delete-process" class = "width100 flex-display flex-column flex-centre flex-item-centre">

<table class = "width80">

<caption>Удаление процесса</caption>

<tr>

<td>Удалить процесс полностью:</td><td><input type = "checkbox" id = "delete-all-page-checkbox" onchange="checkboxChange()"></td>

</tr>

<tr>

<td>Имя процесса:</td><td><input type = "number" min = "1" pattern="[0-9]" id = "delete-name-process-input"></td>

</tr>

<tr>

<td>Номер страницы:</td><td><input type = "number" min = "1" pattern="[0-9]" id = "delete-page-number-input"></td>

</tr>

<tr>

<td colspan="2"><button id = "delete-process-button" class = "button button100" onclick="deleteProcess()">Удалить</button></td>

</tr>

</table>

</div>

</div>

</div>

<div id = "main" class = "width100 flex-display flex-row">

<div id="main-info" class = "width40 flex-display flex-column flex-item-centre">

</div>

<div id = "main-table" class = "width60 flex-display flex-column flex-item-centre">

</div>

</div>

</body>

</html>