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 схему управления памятью»

Выполнил: студент гр. ИТП-11

Клевов А.О

Принял: преподаватель

Карась О. В.

Гомель 2022

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

**Ход работы**

**Задание 1**. Разработать программу, реализующий заданный алгоритм выделения памяти.

Менеджер памяти должен:

* По запросу выделять память согласно заданного алгоритма. На экран выводится информация о состоянии памяти: объем памяти, объем свободной памяти, размер наибольшего свободного блока, количество запросов на выделение памяти, количество удовлетворенных запросов (%).
* Для выделения памяти указывается имя процесса и размер блока. После нажатия на кнопку «Добавить» память выделяется или выдаётся сообщение о невозможности выделения.
* Удалить из памяти заданный блок или все блоки заданного процесса по нажатию кнопки удалить
* Реализовать возможность последовательной записи/чтения информации в/из выделенную память по логическому адресу. Вывести физического адреса ячейки памяти, в которую была осуществлена запись.
* Организовать циклическое выделение и освобождение памяти. При этом случайным образом задаётся количество выделяемых блоков и их размер.

Листинг программы представлен в приложении A.

Вариант 12: Схема с переменными разделами. Случайный выбор раздела.

Окно программы будет выглядеть следующим образом:

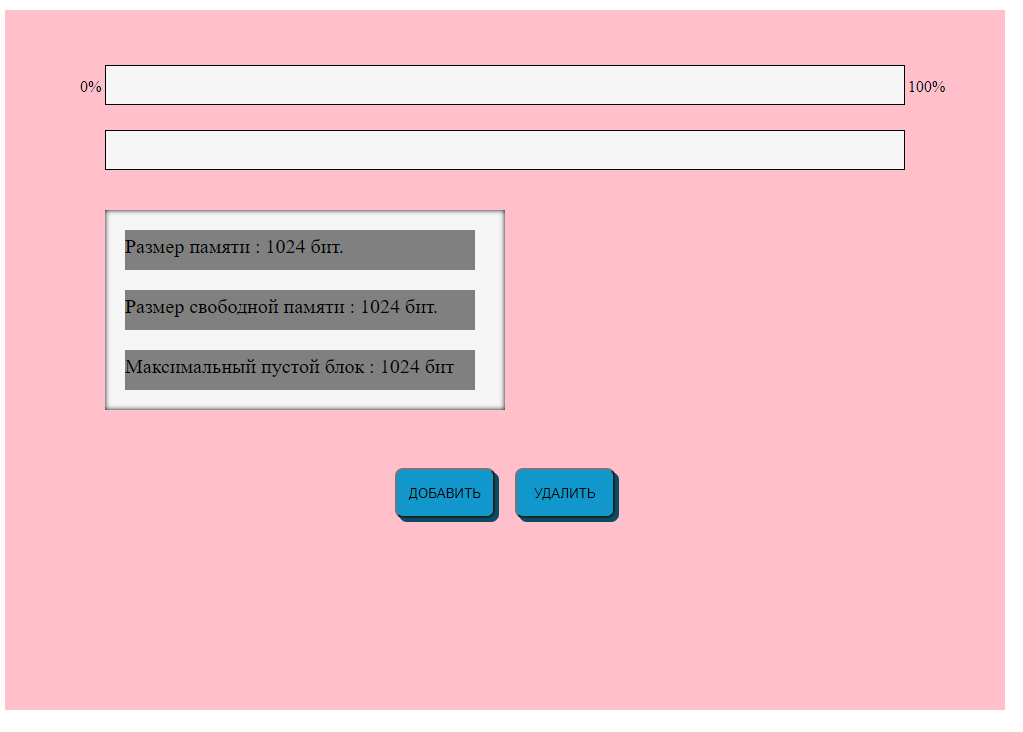


Рисунок 1 – Окно программы

Добавим несколько процессов. Цвет процентной шкалы меняется от зелёного до синего в зависимости от процентной занятости.

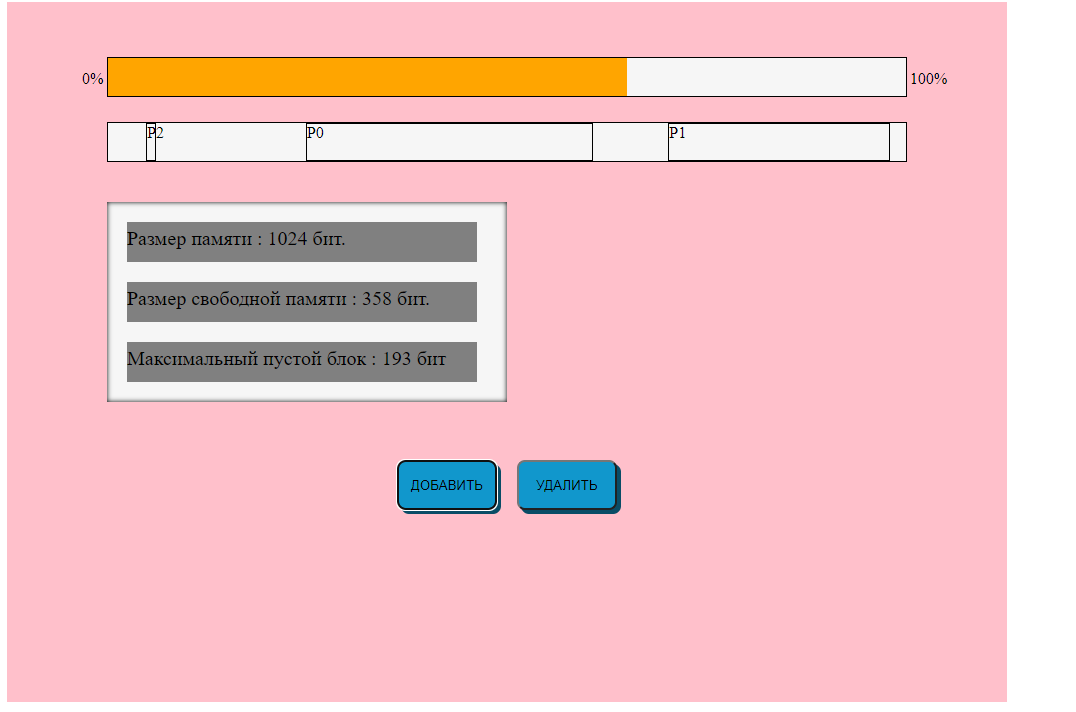


Рисунок 2 – Окно программы после

добавления процессов

Если памяти не хватит программа выдаст сообщение об ошибке

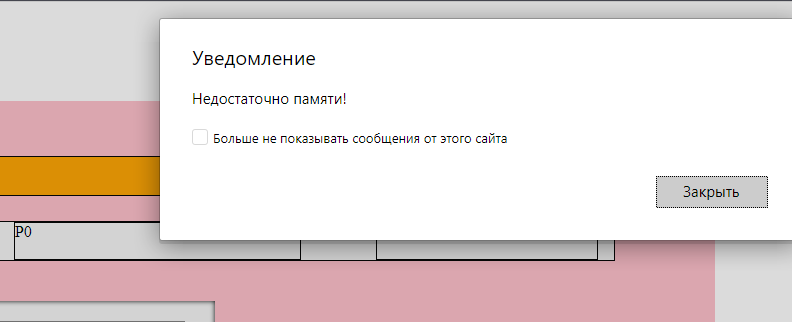


Рисунок 3 – Сообщение о нехватке памяти для заданного процесса

В программе предусмотрен вывод информации о блоках: размер памяти, размер свободной памяти, размер максимального свободного блока.

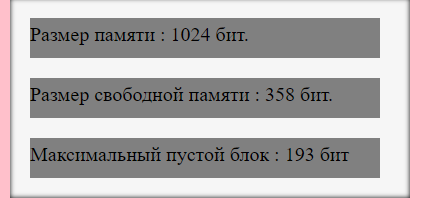


Рисунок 4 – Результат чтения из блока

В интерфейсе присутствую две кнопки добавить и удалить. При нажатии на кнопку добавить спрашивается размер блока, далее он добавляется в случайную позицию памяти, которая представлена массивом из *n* элементов, блоку присваивается уникальное *id* имя и его размер. По кнопке удаления блок удаляется через *id*.

**Вывод**: в ходе работы были изучены алгоритмы управления памятью, разработана программа менеджера памятью.

**ПРИЛОЖЕНИЕ А**

**Html, js**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>LAB9</title>

<link rel="stylesheet" type="text/css" href="LAB9.css">

</head>

<body>

<script type="text/javascript">

//var n = parseInt(prompt("Введите размер памяти",""));

var n=1024;

</script>

<div class="container">

<div class="lab9">

<div class="memcont">

<div class="memory proc">

<div id="full"></div>

</div>

<p id="leftInf">0%</p><p id="rightInf">100%</p>

</div>

<div class="bloks memory" id="blocks"></div>

<div class="inform">

<div id="volMem" class="inf"></div>

<div id="volFrMem" class="inf"></div>

<div id="maxBlock" class="inf"></div>

</div>

<div class="buttons">

<button id="addBlock" onclick="addBlock()">ДОБАВИТЬ</button>

<button id="removeBlock" onclick="removeBlock()"> УДАЛИТЬ</button>

</div>

</div>

</div>

<script type="text/javascript">

document.getElementById('volMem').innerHTML = 'Размер памяти : ' + n + ' бит.';

let arr = new Array(n);

let blockList = new Array(100);

let poss=[];

let possMax = new Array();

let numOfblocks=0;

const blocks = document.getElementById('blocks');

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

blockList[i] = new Array(3);

}

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

arr[i]=0;

}

function chekСompl() {

let k=0;

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

if (arr[i]==1) {k++;}

return k/n;

}

function fiendMax() {

let counter=0,max=0;

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

if (arr[i]==0) {counter++;}

if (max<counter) {max=counter;}

else if (arr[i]==1) {counter=0;}

}

return max;

}

function changeInf() {

document.getElementById('volMem').innerHTML = 'Размер памяти : ' + n + ' бит.';

document.getElementById('volFrMem').innerHTML = 'Размер свободной памяти : ' + (n-n\*chekСompl()) + ' бит.';

document.getElementById('maxBlock').innerHTML = 'Максимальный пустой блок : ' + fiendMax() + ' бит';

let k=chekСompl()\*100;

const full = document.getElementById('full');

if (k<34) {full.style.backgroundColor= 'green';}

else if (k<67) {full.style.backgroundColor= 'orange';}

else if (k<101) {full.style.backgroundColor= 'red';}

k+='%';

full.style.width = k;

}

function randInt(max) {

return Math.floor(Math.random() \* max);

}

function addBlock() {

let size = parseInt(prompt("Введите размер блока",""));

poss.splice(0,poss.length);

let counter=0;

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

if (arr[i]==0) {counter++;}

if (counter>=size) {

let sizeBlock=i-size;

poss.push(sizeBlock);

}

if (arr[i]==1) {counter=0;}

}

if (poss.length!=0) {

let p = poss[randInt(poss.length)];

let wid = size/n\*100;

wid+='%';

let left = p/n\*100;

left+='%';

for (var i = p ; i < p+size; i++) {

arr[i]=1;

}

let div = document.createElement('div');

div.id ='P'+numOfblocks;

div.className = 'block';

div.innerHTML +='P'+numOfblocks;

div.style.width= wid;

div.style.marginLeft = left;

blocks.appendChild(div);

blockList[numOfblocks][0]='P'+numOfblocks;

blockList[numOfblocks][1]=size;

blockList[numOfblocks][2]=p;

numOfblocks++;

changeInf();

}

else{alert('Недостаточно памяти!');}

}

function removeBlock() {

let num = parseInt(prompt("Введите номер процесса (если p1 введите 1)",""));

let size = blockList[num][1];

let p = blockList[num][2];

let idProc = 'P' + num;

for (var i = p ; i < p+size; i++) {

arr[i]=0;

}

document.getElementById(idProc).remove();

blockList[num][0]=0;

blockList[num][1]=0;

blockList[num][2]=0;

changeInf();

}

changeInf();

</script>

</body>

</html>

CSS

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

}

div.container {

width: 100%;

height:auto;

background-color: black;

position: relative;

}

div.lab9{

**position: absolute;**

**left: 50%;**

**width: 1000px;**

**margin-left: -500px;**

**margin-top: 100px;**

**background-color: pink;**

**height: 700px;**

**}**

**div.memory {**

**position: absolute;**

**left: 50%;**

**width: 800px;**

**margin-left: -400px;**

**height: 40px;**

**background-color: #f6f6f6;**

**border: 1px solid;**

**}**

**div.memcont {**

**position: absolute;**

**left: 50%;**

**width: 900px;**

**margin-left: -450px;**

**margin-top: 50px;**

**height: 70px;**

**}**

**#leftInf {**

**float: left;**

**margin: 18px 0 0 25px;**

**}**

**#rightInf {**

**float: right;**

**margin: 18px 10px 0 0;**

**}**

**.proc {**

**margin-top: 5px;**

**}**

**.bloks {**

**margin-top: 120px;**

**}**

**#full {**

**background-color: green;**

**height: 100%;**

**width: 10%;**

**}**

**.inform {**

**position: absolute;**

**left: 100px;**

**width: 400px;**

**margin-top: 200px;**

**background-color: blue;**

**height: 200px;**

**box-shadow: 0 0 5px black inset;**

**background-color: #f6f6f6;**

**}**

**.inf {**

**margin: 20px;**

**background-color: gray;**

**width: 350px;**

**height: 40px;**

**font-size: 20px;**

**padding-top: 5px;**

**}**

**.buttons {**

**position: absolute;**

**left: 50%;**

**width: 900px;**

**margin-left: -450px;**

**margin-top: 450px;**

**height: 70px;**

**}**

**.buttons button {**

**position: absolute;**

**left: 50%;**

**width: 100px;**

**height: 50px;**

**margin-top: 8px;**

**background-color: #1197cc;**

**box-shadow: #094c66 4px 4px 0px;**

**border-radius: 8px;**

**transition: transform 200ms, box-shadow 200ms;**

**}**

**.buttons button:active {**

**transform: translateY(4px) translateX(4px);**

**box-shadow: #094c66 0px 0px 0px;**

**}**

**#removeBlock {**

**margin-left: 10px;**

**}**

**#addBlock {**

**margin-left: -110px;**

**}**

**.block {**

**height: 100%;**

**border: 1px solid;**

**position: absolute;**

**}**