**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 №6**

по дисциплине «Операционные системы»

на тему: «Программирование планировщика процессов»

Выполнил: студент гр. ИТИ-11

Цветков А.А.

Принял: преподаватель Карась О.В.

Гомель 2022

**Цель работы:** разработать программу, осуществляющую моделирование режима работы с разделение времени.

**Задание 1. Не вытесняющие алгоритмы планирования процессов.**

Выполнить программирование планировщика процессов, согласно алгоритму *RR*

**Ход выполнения:**

Был реализован алгоритм планирования *RR.* В качестве языка программирования был выбран *JavaScript.* Ввод процессов и их длительность осуществляет пользователь.

Результат выполнение программы представлен на рисунках 1-2.

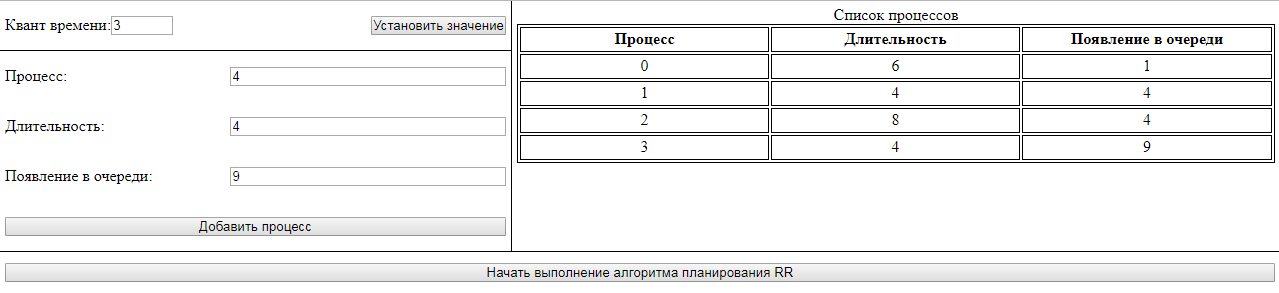


Рисунок 1 – Формирование списка процессов пользователем

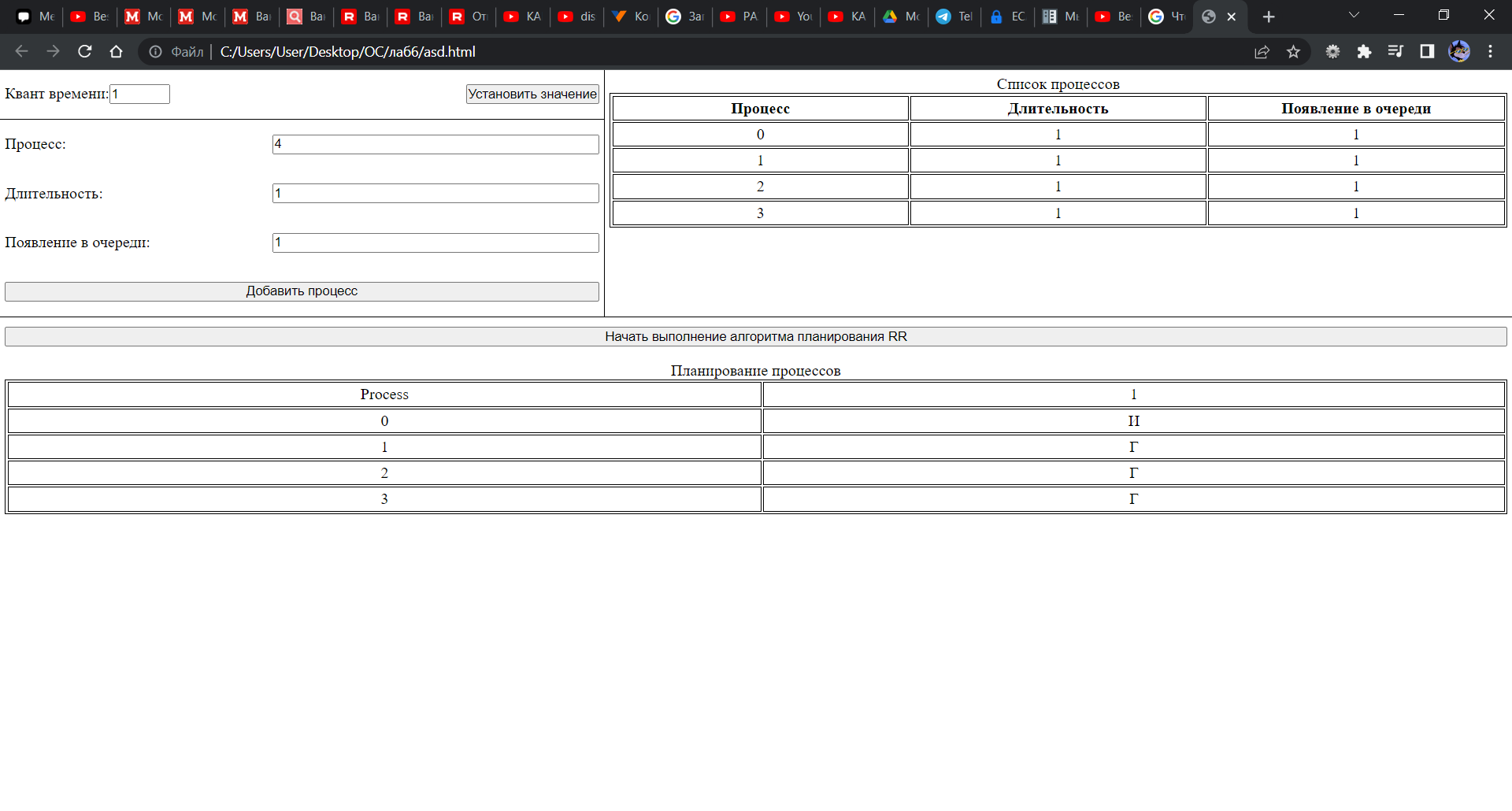


Рисунок 2 – Программный результат планирования

**Листинг программы:**

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Document</title>

<style>

\*{

box-sizing: border-box;

margin: 0;

padding: 0;

}

.menu{

border-bottom: 1px solid black;

width: 100%;

display: flex;

flex-direction: row;

}

.input{

border-right: 1px solid black;

height: 250px;

width: 40%;

display: flex;

flex-direction: column;

}

.time{

border-bottom: 1px solid black;

height: 50px;

width: 100%;

display: flex;

justify-content: space-between;

align-items: center;

padding-left: 5px;

padding-right: 5px;

}

.data{

height: 250px;

width: 60%;

padding: 5px;

overflow: auto;

}

.processForm input{

width: 100%;

}

.processForm div{

padding-left: 5px;

padding-right: 5px;

height: 50px;

width: 100%;

display: flex;

justify-content: space-between;

align-items: center;

}

.processForm button{

width: 100%;

}

.processForm .tagName{

width: 45%;

}

.processForm .tagInput{

width: 55%;

}

table{

width: 100%;

border: 1px solid black;

}

table tr{

border: 1px solid red;

width: 100%;

height: 25px;

}

table td,th{

width: 33%;

height: 25px;

border: 1px solid black;

text-align: center;

}

.bigButton{

padding: 5px;

width: 100%;

height: 40px;

display: flex;

align-items: center;

}

.bigButton button{

width: 100%;

}

.mainBlock{

padding: 5px;

overflow: auto;

width: 100%;

}

#tablePlans td,th{

width: 50px;

}

</style>

<script type="text/javascript">

var kvant = 1;//квант времени, по умолчанию = 1;

var processIndex = 0;//количество введенных пользователем процессов

var process = new Array();//создание массива процессов

//функция изменения кванта времени

function kvantChange(){

kvant = document.getElementById('timeProcess').value;

}

//функция добавления процесса в таблицу процессов

function addProcess(){

var name = document.getElementById('nameProcess').value; //имя процесса(номер по порядку)

var duration = document.getElementById('durationProcess').value;//длительность процесса

var appearance = document.getElementById('appearanceProcess').value;//время вхождения в очередь

process[processIndex] = new Array();//выделение память под столбцы

process[processIndex][0] = name;//0 столбец - имя процесса

process[processIndex][1] = duration;//1 столбец - длительность процесса

process[processIndex][2] = appearance;//2 столбец - время вхождения в очередь

process[processIndex][3] = 0;//3 столбец - статус(0 - не входил в очередь, 1 - готов к выполнению, 2 - выполняется, 3 - выполнился)

process[processIndex][4] = 0;

processIndex++;

document.getElementById('nameProcess').value = processIndex; //(автоматически задаётся порядок процессов)

var tablePr = document.getElementById("tableProcess");

var row = tablePr.insertRow(-1);//добавление в конец таблицы строки с процессом

//добавление в строку ячеек с индексами и содержимым

var cell = row.insertCell(0);

cell.appendChild(document.createTextNode(name));

var cell = row.insertCell(1);

cell.appendChild(document.createTextNode(duration));

var cell = row.insertCell(2);

cell.appendChild(document.createTextNode(appearance));

}

//основная функция программы. выполнение планирования

function run(){

//создание таблицы планирования

var create = '<table cellpadding="5" id = "tablePlans">';

create += '<caption>Планирование процессов</caption>';

create += '</table>';

document.getElementById('1').innerHTML = create;

//создание строки заголовка

var table = document.getElementById("tablePlans");

var row = table.insertRow(0);

var cell = row.insertCell(0);

cell.appendChild(document.createTextNode('Process'));

//вызов функции сортировки

sort();

//создание строки в таблице под каждый процесс

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

{

row = table.insertRow(i);

cell = row.insertCell(0);

cell.appendChild(document.createTextNode(process[i-1][0]));

}

row = table.rows; //переменная row содержит строки таблицы

var cellIndex = 1;//переменная содержит индекс следующей ячейки, которая будет создана

//создание массива для работы с процессами

var process1 = new Array();

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

{

process1[i] = new Array();

for(var j = 0; j < 5; j++)

process1[i][j] = process[i][j];

}

var countProcess = processIndex; //содержит количество процессов которые должны выполнится

var time = 1; //текущее время выполнения

var countEnter = 0; //количество вошедших в очередь процессов

var index = 0; //Текущий процесс

var implementationFlag = 0;

var statusSave = "";

var statusSaveIndex;

var kvantTime = 1;

var priority = 0;

while(countProcess != 0)

{

flag = 0;

for(var i = countEnter; i < processIndex && flag == 0; i++)

{

if(process1[i][2] == time)

{

priority++; //увеличиваем значение приоритета

countEnter++; //увеличиваем количество вошедших процессов

//присваиваем приоритет

process1[i][3] = 1;

process1[i][4] = priority;

}

else

{

flag = 1; //если время вхождения приоритета не равно текущему, выходим из цикла

}

}

flag = 0;

if(process1[index][3] == 1 && implementationFlag == 0){

process1[index][3] = 2; //Если текущий процесс готов к выполнению и другой процесс не выполняется, то ему присваивается статус "выполняется"

statusSave = "И"; // статус "исполняется"

statusSaveIndex = index; //запоминаем индекс выолняемого процесса

implementationFlag = 1; //помечаем, что есть выполняющийся процесс

}

//добавление столбца с процессами на текущем кванте времени

cell = row[0].insertCell(cellIndex);

cell.appendChild(document.createTextNode(time));

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

{

if(statusSave != "" && statusSaveIndex == i-1){

status = statusSave;

statusSave = "";

statusSaveIndex = -1;

}

else

{

if(process1[i-1][3] == 0 || process1[i-1][3] == 3)

status = " ";

else

if(process1[i-1][3] == 1)

status = "Г";

else

status = "И";

}

cell = row[i].insertCell(cellIndex);

cell.appendChild(document.createTextNode(status));

}

cellIndex++;

time++;

if(process1[index][3] == 2)

process1[index][1] -= 1;//У исполняемого процесса убираем еденицу времени выполнения

if(kvantTime >= kvant)//Проверка на длительность кванта

flag = 1;

else

kvantTime++;

if(process1[index][1] <= 0 && process1[index][3] != 3)//Если дительность процесса завершилась и процесс не имеет статус "выполнен", то уменьшается количество выполняемых процессов, а процессу присваивается статус "выполнен"

{

countProcess--;

flag = 1;

process1[index][3] = 3;

}

if((flag == 1 || implementationFlag == 0) && countProcess > 0)

{

implementationFlag = 0; //выполняемых процессов нет

if(process1[index][3] != 3) //если выполняемый процесс не выполнился, то присваеваем ему статус "готов к выполнению"

process1[index][3] = 1;

priority++;

process1[index][4] = priority; //увеличиваем значение приоритета процесса

kvantTime = 1; //устанавливаем значение кванта по умолчанию

//нахождение индекса следующего готового к выполнению процесса, с наименьшим числовым приоритетом

var minPriority = process1[index][4]; //значение приоритета следующего процесса, должно быть меньше нового приоритета текущего процесса

//поиск готового процесса с наименьшим числовым приоритетом

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

{

if (i != index)

{

if(minPriority > process1[i][4] && process1[i][3] == 1)

{

index = i;

minPriority = process1[i][4];

}

}

}

}

}

}

//сортировка процессов по времени вхождения в очередь. От меньшего к большему

function sort(){

var trade0,trade1,trade2,trade3,trade4;

for(var i = 0; i < processIndex-1; i++)

{

for(var j = i+1; j < processIndex; j++)

{

if(parseInt(process[i][2],10) > parseInt(process[j][2],10))

{

trade0 = process[i][0];

trade1 = process[i][1];

trade2 = process[i][2];

trade3 = process[i][3];

trade4 = process[i][4];

process[i][0] = process[j][0];

process[i][1] = process[j][1];

process[i][2] = process[j][2];

process[i][3] = process[j][3];

process[i][4] = process[j][4];

process[j][0] = trade0;

process[j][1] = trade1;

process[j][2] = trade2;

process[j][3] = trade3;

process[j][4] = trade4;

}

}

}

}

</script>

</head>

<body>

<div class = "menu">

<div class = "input">

<div class = "time">

<p>

<label>Квант времени:<input type="number" value="1" min="1" max="20" step="1" id = "timeProcess" onkeypress="return false" width="35%"></label>

</p>

<button type="button" onclick="kvantChange()">

Установить значение

</button>

</div>

<div class = "processForm">

<div>

<p class = "tagName">Процесс:</p>

<p class = "tagInput">

<input type="text" id = "nameProcess" onkeypress="return false" value = "0">

</p>

</div>

<div>

<p class = "tagName">Длительность:</p>

<p class = "tagInput">

<input type="number" value="1" min="1" max="10" step="1" id = "durationProcess" onkeypress="return false">

</p>

</div>

<div>

<p class = "tagName">Появление в очереди:</p>

<p class = "tagInput">

<input type="number" value="1" min="1" max="" step="1" id = "appearanceProcess" onkeypress="return false">

</p>

</div>

<div>

<button type = "button" onclick="addProcess()">Добавить процесс</button>

</div>

</div>

</div>

<div class = "data">

<table cellpadding="5" id = "tableProcess">

<caption>

Список процессов

</caption>

<tr>

<th>Процесс</th><th>Длительность</th><th>Появление в очереди</th>

</tr>

</table>

</div>

</div>

<div class = "bigButton">

<button onclick="run()">

Начать выполнение алгоритма планирования RR

</button>

</div>

<div class = "mainBlock" id = "1">

</div>

</body>

</html>

**Вывод:** в результате выполнения работы, был запрограммирован планировщик, выполняющий алгоритм согласно варианту.