УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

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

Кафедра «Информационные технологии»

Лабораторная работа №6

**«Программирование планировщика процессов»**

Выполнил: студент гр. ИТП-11

Воробьев В.А.

Принял: преподаватель

Карась О. В.

Гомель 2022

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

**Ход работы**

В соответствии с вариантом (выдается преподавателем), разработать программу на языке С# (или другом), осуществляющую моделирование работы планировщика процессов. Моделирование режима разделения времени выполняется в соответствии с заданием (таблица 2.1). На экран выводится следующая информация:

1.Номер текущего кванта времени процессора;

2. Таблица процессов с указанием имени процессов, продолжительности, приоритета (в зависимости от задания), оставшегося времени выполнения, время появления;

3. Таблица планирования процессов с отображением текущего состояния процессов.

После запуска, программа должна диалоговом режиме, ввести информацию процессах – имя, длительность, приоритет, время появления. Для алгоритмов RR число квантов времени. Выполнение должно производиться в пошаговом режиме (по нажатию на кнопку). По окончанию работы процесса на экране должно выводится сообщение о его завершении «Процесс такой-то завершен».

В соответствии с вариантом надо сделать алгоритм RR не вытесняющий приоритетный, таблица процессов приведена ниже:

|  |  |  |
| --- | --- | --- |
| Процесс | Продолжительность | Приоритет |
| P0 | 9 | 3 |
| P1 | 8 | 2 |
| P2 | 7 | 1 |

Текст программы, написанной на языке *JavaScript*, представлен в листинге **А**. Если у процессов одинаковый приоритет, то первым выполнится тот процесс, время выполнения которого меньше. По нажатию на кнопку вывода планирования выводится строка, состоящая из имён процессоров, которые по времени кванта, а затем передают очередь другому. В случае, когда время исполнения процесса меньше кванта, процесс прекращает работу и следует следующий процесс.

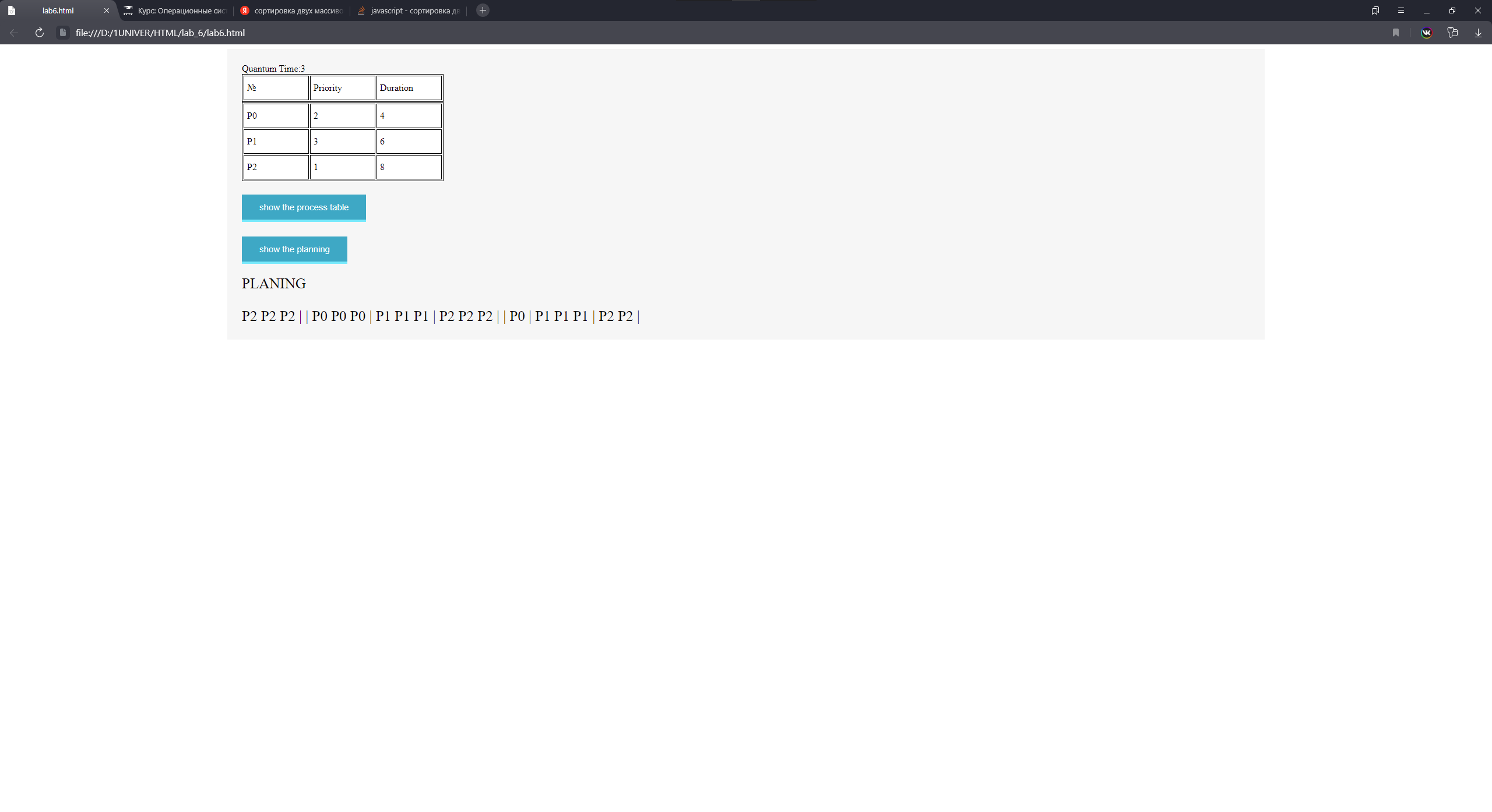


Рисунок 1 — Пример состояния на

текущий квант времени

После выполнения программы окончательно увидим следующее:

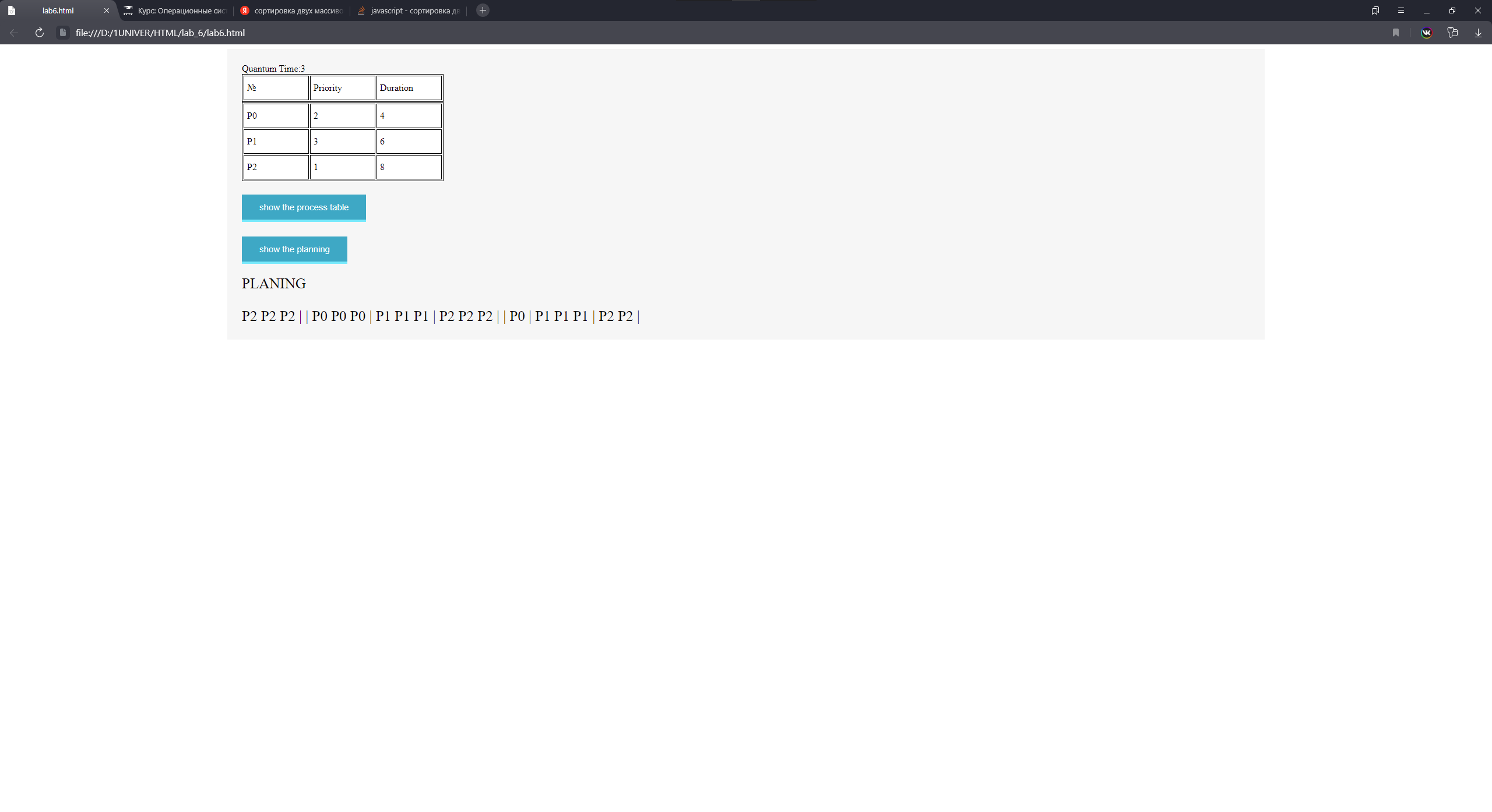


Рисунок 2 — Результат выполнения программы

**Вывод**: в ходе работы была разработана программа, осуществляющая моделирование режима работы с разделением времени.

**ЛИСТИНГ А**

**КОД ПРОГРАММЫ, МОДЕЛИРУЮЩЕЙ**

**РАБОТУ ПЛАНИРОВЩИК ЗАДАЧ**

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<title></title>

<link rel="stylesheet" type="text/css" href="lab6.css">

</head>

<body>

<script type="text/javascript">

"use strict";

let quantum = parseInt(prompt("Enter a time quantum: "));

let priority = [], duration = [];

let N = prompt("Enter the number of processes: ");

let i, j, burntime = 0, min, k =quantum,c = 0, difference;

for ( i = 0; i < N; i++) {

priority.push(prompt("Enter priority of process: "));

duration.push(prompt("Enter duration of process: "));

}

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

burntime += parseInt(duration[i]);

}

function ShowProcessTable() {

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

for ( i = 0; i < N; i++) {

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

for ( j = 0; j < 3; j++) {

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

switch(j) {

case 0:td.innerHTML = 'P' + i;break;

case 1:td.innerHTML = priority[i];break;

case 2:td.innerHTML = duration[i] ;break;

}

tr.appendChild(td);

}

table.appendChild(tr);

}

document.getElementById("tableDiv").appendChild(table);

document.getElementById("quantum").innerHTML = 'Quantum Time:' + quantum;

}

function FindMin() {

min = priority[0];

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

if (priority[i+1] < priority[i]) {min = priority[i+1];}

}

return priority.indexOf(min);

}

c= FindMin();

function Show() {

let div = document.getElementById("Div");

div.innerHTML += '<br>';

for ( i = 0; i < burntime; ) {

if (c==N) {c=0;div.innerHTML += ' |';}

if (duration[c] >= quantum ) {

for (j = 0; j < quantum; j++) {

div.innerHTML += ' P' + c;

duration[c]--;

i++;

}

++c;

div.innerHTML += ' |';

}

if (duration[c] < quantum) {

for (j = 0; j < duration[c]; j++) {

div.innerHTML += ' P' + c;

i++;

}

++c;

div.innerHTML += ' |';

}

}

}

</script>

<div class="container">

<div id="tableDiv">

<div id="quantum"></div>

<table>

<tr>

<td>№</td>

<td>Priority</td>

<td>Duration</td>

</tr>

</table>

</div>

<br><input type="button" value="show the process table" class="dws" onclick="ShowProcessTable()">

<br><input type="button" value="show the planning" class="dws" onclick="Show()">

<div id="Div" > PLANING</b><br>

</div>

</div>

</body>

</html>

.dws{

padding: 13px 30px;

margin: 5px 0 20px 0;

font-size: 15px;

color: #fff;

background-color: #2ca8c6;

border: none;

border-bottom: 4px solid #6ee9fd;

cursor: pointer;

}

.dws:hover{

transition: all 0.5s;

background: #fff;

color: #2c536c;

}

.container{

margin: 0 15% 0 15%;

background: #f6f6f6;

padding: 1%;

}

table{

background: white;

border: 1px solid;

}

td{

padding: 5px;

border: 1px solid;

width: 100px;

height: 30px;

}

tr{

padding: 5px;

border: 1px solid;

}

#Div{

font-size: 25px;

}