УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

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

Кафедра «Информационные технологии»

Лабораторная работа №6

**«Программирование планировщика процессов»**

Выполнил: студент гр. ИТП-11

Мурашко М.Д.

Принял: преподаватель

Карась О. В.

Гомель 2022

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

**Задание:** 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 SJF вытесняющий приоритетный, таблица процессов приведена ниже:

|  |  |  |  |
| --- | --- | --- | --- |
| Процесс | Продолжительность | Время появления в очереди | Приоритет |
| P0 | 8 | 0 | 2 |
| P1 | 3 | 4 | 2 |
| P2 | 2 | 7 | 3 |
| P3 | 3 | 3 | 1 |

Текст программы, написанной на языке *JavaScript*, представлен в приложении **А**. В данном алгоритме учитывается врем\ появления процессов в очереди и приоритет процессов, на рисунке 1 представлена исходная таблица данных, на рисунке 2 представлен результат моделирования работы алгоритма.

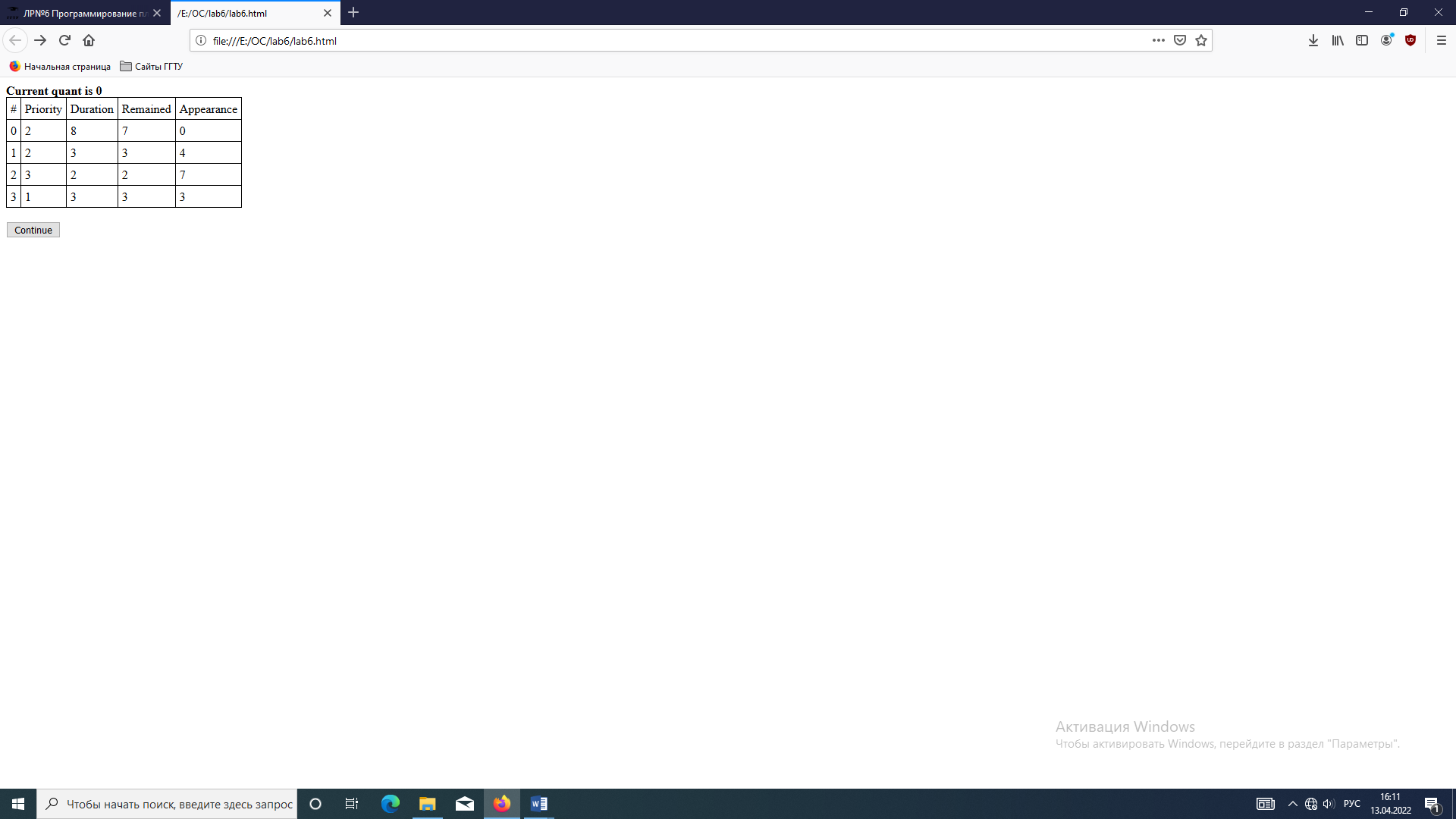


Рисунок 1 — Исходная таблица данных

После выполнения программы окончательно увидим следующее:

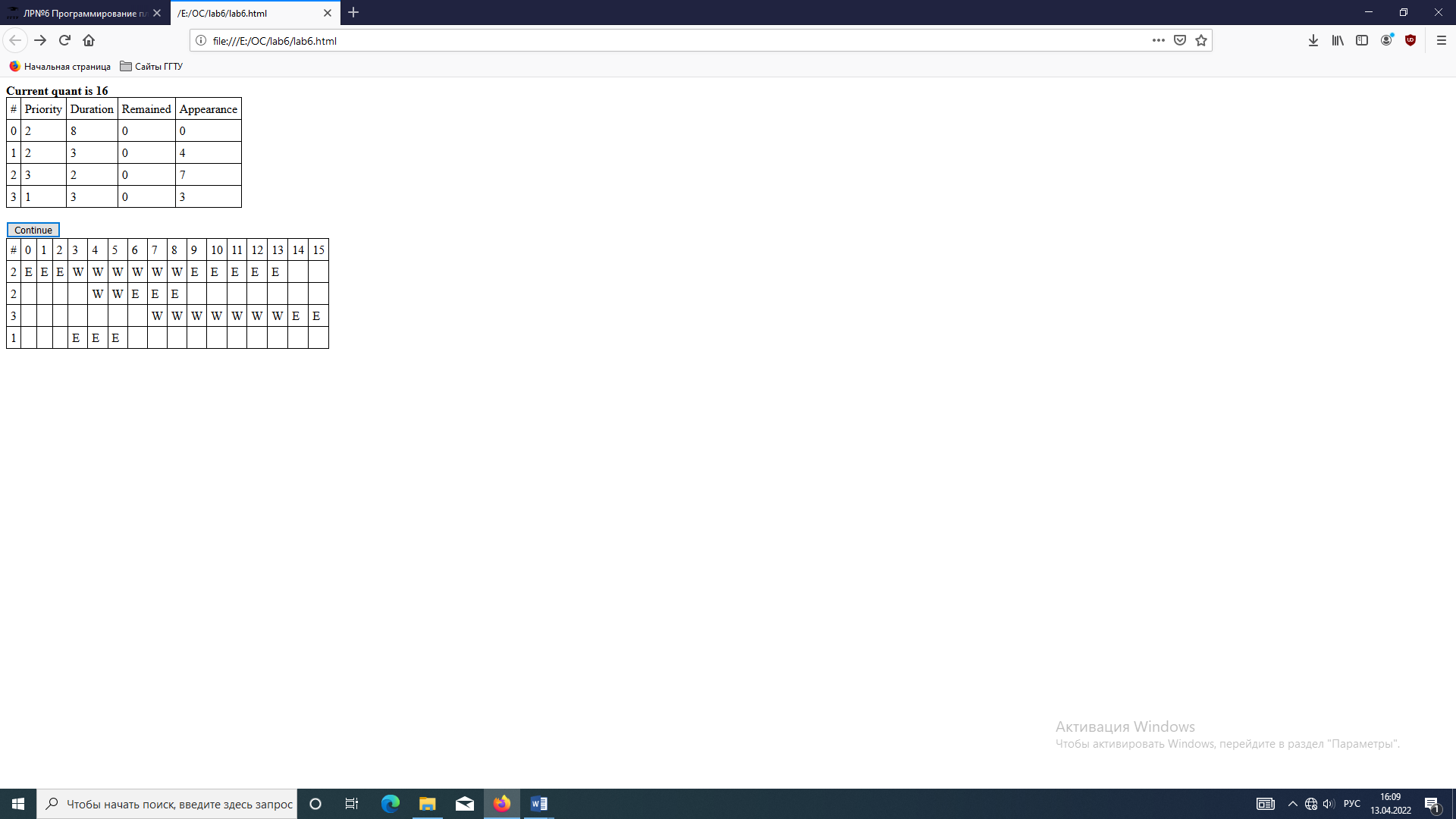


Рисунок 2 — Результат выполнения программы

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

**Приложение А**

"use strict";

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

let btn = document.getElementById("btn");

let process = [];

let processN = +prompt("Enter amount of processess: ");

for (let i = 0; i < processN; i++)

{

let priority = +prompt("Enter priority of process: ");

let duration = +prompt("Enter duration of process: ");

let appearance = +prompt("Enter appearance of process: ");

process.push({priority, duration, remained: duration, appearance, isCompleted: false, states: []});

}

let table = [];

let statesN = [];

let wAmount = 0, eAmount = 0;

let isPrEx = false, f = false;

let numOfExPr = 0;

let q = 0;

btn.addEventListener("click", (e) =>

{

if (process.some(el => !el.isCompleted))

{

f = false;

if (!isPrEx)

{

let processQ = findQprocess(process);

if (processQ.length)

{

numOfExPr = findMaxProcess(processQ);

isPrEx = true;

let isF = false;

if (processQ.length >= 2) {

processQ.forEach((pr, ind) => {

if (!isF && pr.i !== numOfExPr && pr.pr.priority < process[numOfExPr].priority)

{

numOfExPr = pr.i;

isF = true;

}

});

}

process = putStates(process, processQ, numOfExPr);

process[numOfExPr].remained--;

}

else

process = putStates(process, processQ, numOfExPr)

}

else if (isPrEx && process[numOfExPr].remained)

{

let processQ = findQprocess(process);

if (processQ.length)

{

let num = findMaxProcess(processQ);

if (process[numOfExPr].priority > process[num].priority)

numOfExPr = num;

}

process = putStates(process, processQ, numOfExPr);

process[numOfExPr].remained--;

}

else if (isPrEx && !process[numOfExPr].remained)

{

process[numOfExPr].isCompleted = true;

console.log(`Process with priority ${process[numOfExPr].priority} is over!\n`);

isPrEx = false;

f = true;

}

div.innerHTML = "";

showQuantTable(div, process, q);

if (!f) {

if (isPrEx)

console.log(`Process with priority ${process[numOfExPr].priority} is executing!`);

statesN.push(q++);

}

} else

showTable();

});

btn.click();

function findMaxProcess(process) {

return process.sort((el1, el2) => el2.pr.appearance - el1.pr.appearance)[0].i;

}

function findQprocess(process) {

let arr = [];

process.forEach((pr, i) => pr.appearance <= q && !pr.isCompleted && arr.push({pr, i}));

return arr;

}

function putStates(process, processQ, numOfExPr)

{

process.forEach((el, i) =>

{

if (i != numOfExPr && !processQ.some(e => e.i == i))

el.states.push(" ");

else if (i != numOfExPr && processQ.some(e => e.i != i))

el.states.push("W");

else if (i == numOfExPr && !el.isCompleted)

el.states.push("E");

});

return process;

}

function showQuantTable(el, process, q)

{

let str1 = ["#", 'Priority', 'Duration', 'Remained', 'Appearance'];

let b = document.createElement("b");

b.innerHTML = `Current quant is ${q}`;

el.append(b);

el.append(document.createElement("br"));

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

let thead = document.createElement("thead");

let tbody = document.createElement("tbody");

table.style.borderCollapse = "collapse";

table.setAttribute("border", "1");

str1.forEach((el) => {

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

td.innerHTML = el;

thead.append(td);

})

process.forEach((el, ind) => {

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

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

td.innerHTML = ind;

tr.append(td);

Object.values(el).slice(0, 4).forEach((propName) =>

{

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

td.innerHTML = propName;

tr.appendChild(td);

});

tbody.append(tr);

})

table.append(tbody);

table.append(thead);

el.append(table);

}

function showTable()

{

table.push(["#", ...statesN]);

let t = document.createElement("table");

let thead = document.createElement("thead");

let tbody = document.createElement("tbody");

t.style.borderCollapse = "collapse";

t.setAttribute("border", "1");

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

table[0].forEach((el) => {

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

td.innerHTML = el;

tr.append(td);

})

thead.append(tr);

t.append(thead);

for (let i = 0; i < process.length; i++)

{

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

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

td.innerHTML = process[i].priority;

tr.append(td);

process[i].states.forEach((state) => {

td = document.createElement("td");

td.innerHTML = state;

tr.append(td);

});

tbody.append(tr);

}

t.append(tbody);

document.body.appendChild(t);

}