**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 «Основы алгоритмизации и программирования»

Синхронизация процессов

Программирование планировщиков процессов

Выполнил студент

группы ИТИ-11

*Буякевич Д.О.*

Проверил преподаватель

*Карась О.В.*

Гомель 2022

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

**Задание:** 2. 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. 3. Таблица процессов с указанием имени процессов, продолжительности, приоритета (в зависимости от задания), оставшегося времени выполнения, время появления; 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».

На рисунке 1 представлен вариант лабораторной работы



Рисунок 1 – Вариант ЛР

**Тесты:**

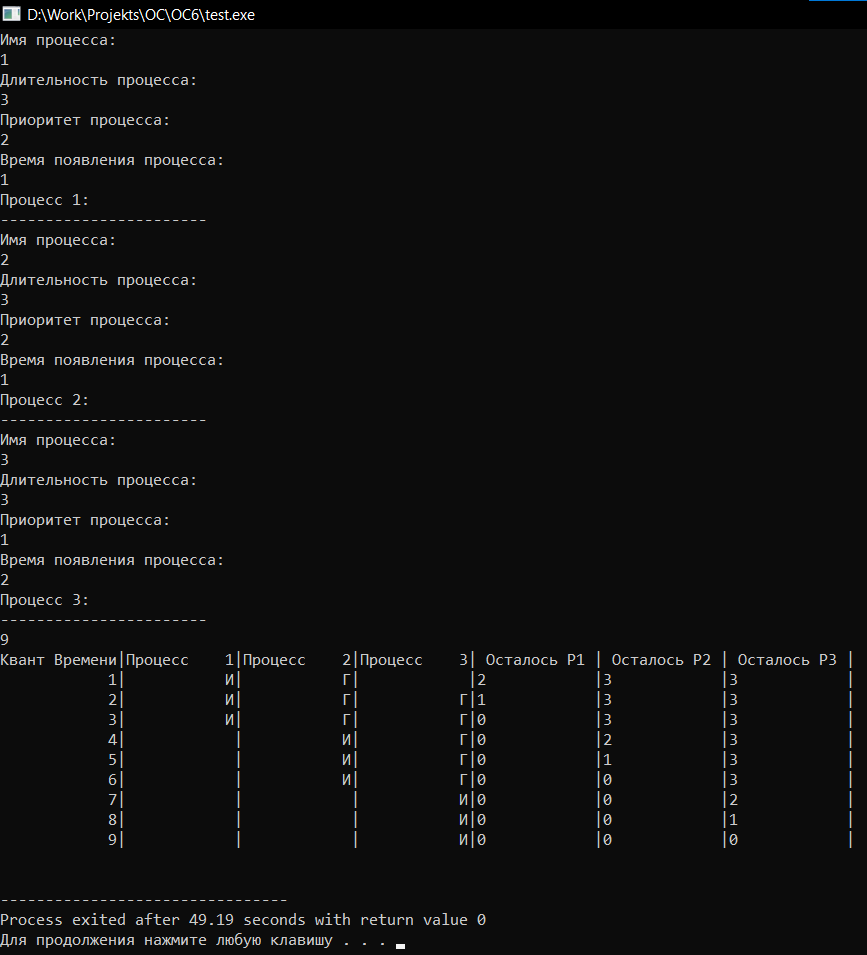
****

Рисунок 2 – Тест №1

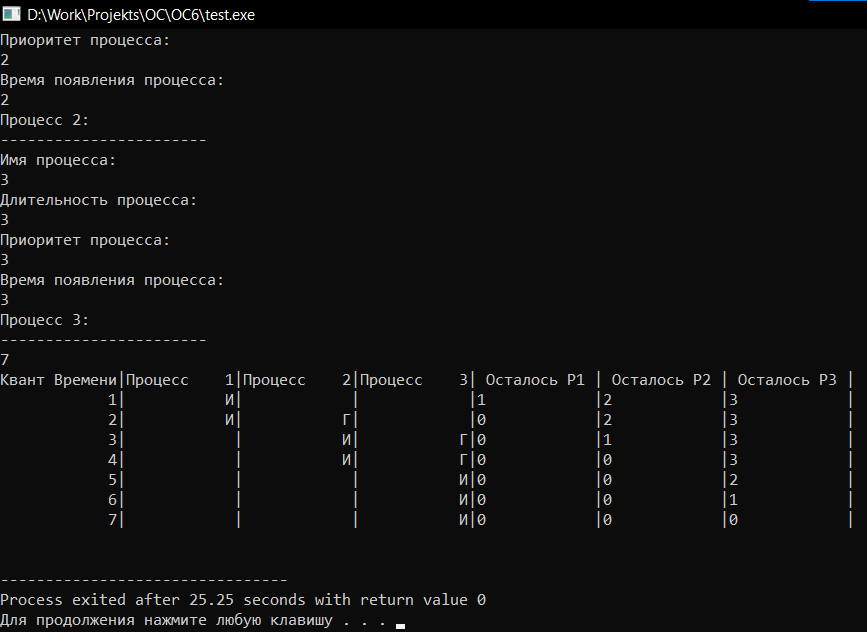


Рисунок 3 – Тест №2

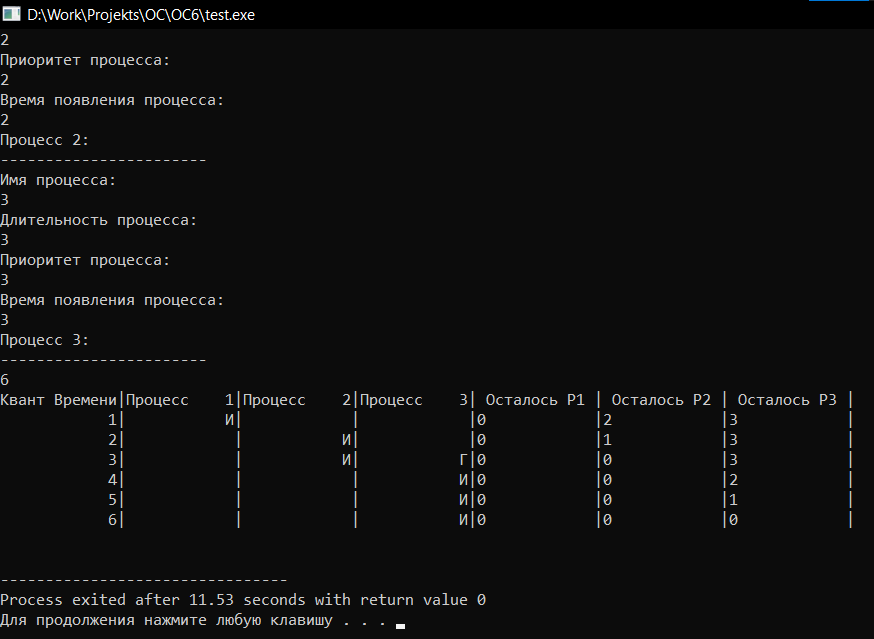


Рисунок 4 – Тест №3

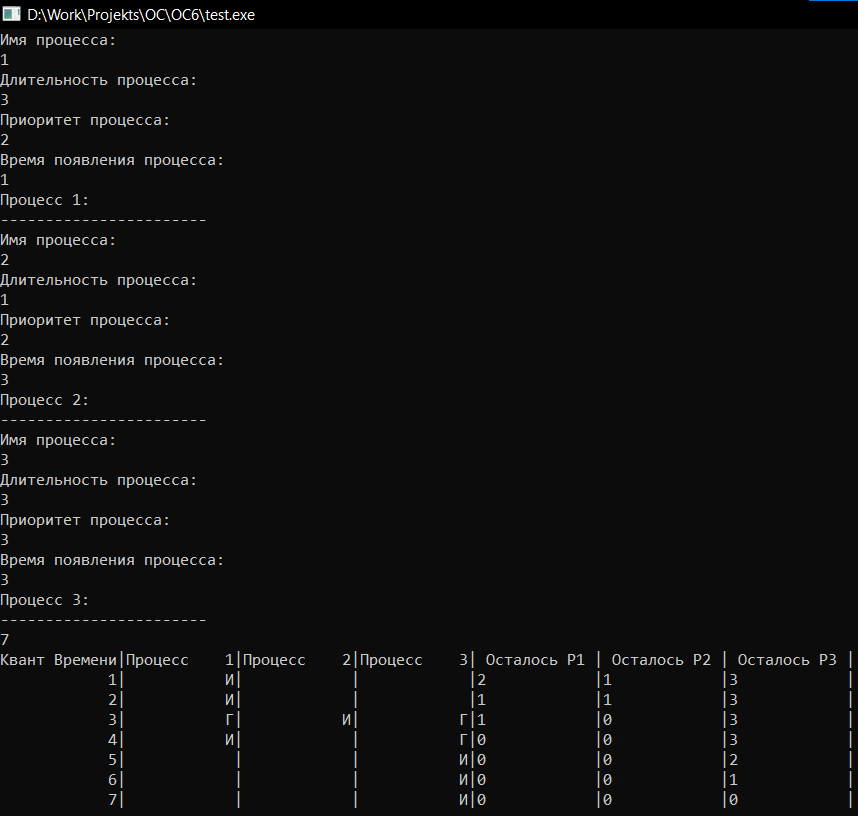


Рисунок 5 – Тест №4

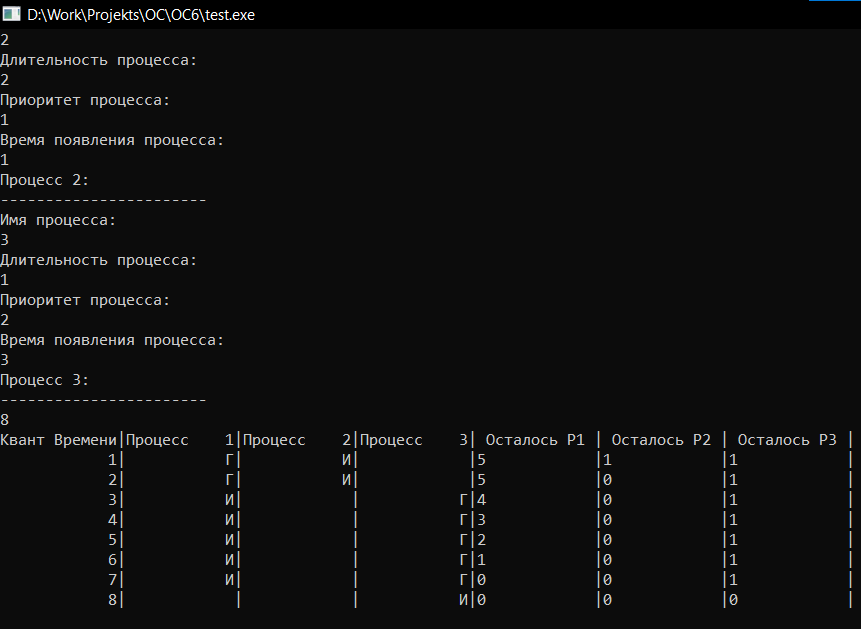


Рисунок 6 – Тест №5

Вывод: разработали программу, осуществляющую моделирование режима работы с разделение времени. Закрепили знания по поводу поведения процессов в разных условиях.

**ПРИЛОЖЕНИЕ А**

**Листинг программы**

#include <stdio.h>

#include <conio.h>

#include <string.h>

#include <locale.h>

#include <stdlib.h>

char name1[10],name2[10],name3[10];

int timeAlive1, timeAlive2, timeAlive3;

int prior1,prior2,prior3;

int time1, time2, time3;

struct processes

{

char name[10];

int timeAlive;

int prioritet;

int time;

char symbol[5];

int left;

};

int main()

{

int minPriority,mintimeAlive,currentInd;

setlocale(LC\_ALL,"RUS");

struct processes MainProcess[3];

for (int i=0;i<3;i++)

{

puts("Имя процесса: ");

gets(MainProcess[i].name);

puts("Длительность процесса: ");

scanf("%d",&MainProcess[i].timeAlive);

puts("Приоритет процесса: ");

scanf("%d",&MainProcess[i].prioritet);

puts("Время появления процесса: ");

scanf("%d",&MainProcess[i].time);

MainProcess[i].left=MainProcess[i].timeAlive;

printf("Процесс %d: \n-----------------------\n",i+1);

fflush(stdin);

}

int timeAll=MainProcess[0].timeAlive+MainProcess[1].timeAlive+MainProcess[2].timeAlive;

printf("%d\n",timeAll);

printf("Квант Времени|Процесс %4s|Процесс %4s|Процесс %4s| Осталось P1 | Осталось P2 | Осталось P3 |\n",MainProcess[0].name,MainProcess[1].name,MainProcess[2].name);

for (int i=1;i<=timeAll;i++)

{

minPriority=100;

mintimeAlive=timeAll;

for(int k=0;k<3;k++)

{

if (MainProcess[k].time<=i && MainProcess[k].left>0)

{

if (MainProcess[k].prioritet<=minPriority)

{

minPriority=MainProcess[k].prioritet;

if (MainProcess[k].timeAlive<mintimeAlive)

{

mintimeAlive=MainProcess[k].timeAlive;

currentInd=k;

}

}

}

}

for (int k=0;k<3;k++)

{

if(currentInd==k && MainProcess[k].left>0)

strcpy(MainProcess[k].symbol,"И");

else if(MainProcess[k].time<=i && MainProcess[k].left>0)

strcpy(MainProcess[k].symbol,"Г");

else

strcpy(MainProcess[k].symbol," ");

}

//printf("%7d|%4s|%4s|%4s|\n",i,MainProcess[0].symbol,MainProcess[1].symbol,MainProcess[2].symbol);

fflush(stdin);

MainProcess[currentInd].left--;

printf(" %7d| %4s| %4s| %4s|%-13d|%-13d|%-13d|",i,MainProcess[0].symbol,MainProcess[1].symbol,MainProcess[2].symbol,MainProcess[0].left,MainProcess[1].left,MainProcess[2].left);

// printf("\n");

getchar();

}

fflush(stdin);

getchar();

return 0;

}