

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(ДГТУ)**

Факультет «Информатика и вычислительная техника»

Кафедра «Программное обеспечение вычислительной техники и автоматизированных систем»

**Лабораторная работа №3**

По дисциплине: «Эвристические методы и алгоритмы»

Выполнил студент группы ВПР32

Бекеров Максим

Проверил

проф. Кобак В.Г.

Ростов-на-Дону  
2024

### **Введение**

Предметом области исследования расписаний является круг задач проектирования и организационного управления в различных системах, в которых требуется найти наилучшее (оптимальное) значение выбранных критериев их функционирования с учетом имеющихся ограничений.

Программирование для многопроцессорных машинных систем связано с распараллеливанием и синхронизацией вычислений и организацией выполнения параллельных вычислительных процессов. Это выдвигает целый ряд сложных задач, среди которых весьма важными являются, расчет характеристик времени и количества операций, требующихся для выполнения параллельных программ, и построения расписаний (планов), выполнения параллельных программ на многопроцессорных и многомашинных вычислительных системах.

Модели параллельных программ и операционные характеристики процессов их выполнения служат основой для планирования параллельных вычислительных процессов, т.е. для построения расписаний указанных процессов. Расписания параллельных вычислительных процессов определяют порядок выполнения программы на вычислительной системе, включая распределение частей программы по процессам. С увеличением числа распределяемых частей программ и количества используемых процессоров сложность построения оптимальных расписаний обычно резко возрастает. Поэтому важное значение имеют простые в построении и удобные в реализации приближенные расписания параллельных вычислительных процессов, близкие к оптимальным с точки зрения времени выполнения параллельных программ.

# Постановка задачи

Имеется вычислительная система (ВС), состоящая из N несвязанных неидентичных устройств (приборов, процессоров и т.п.).

На обслуживание в ВС поступает набор из M независимых непараллельных заданий (работ)

T = {t1,1 t1,2 t1,3, … t1,M

t2,1 t2,2 t2,3, … t2,M

…

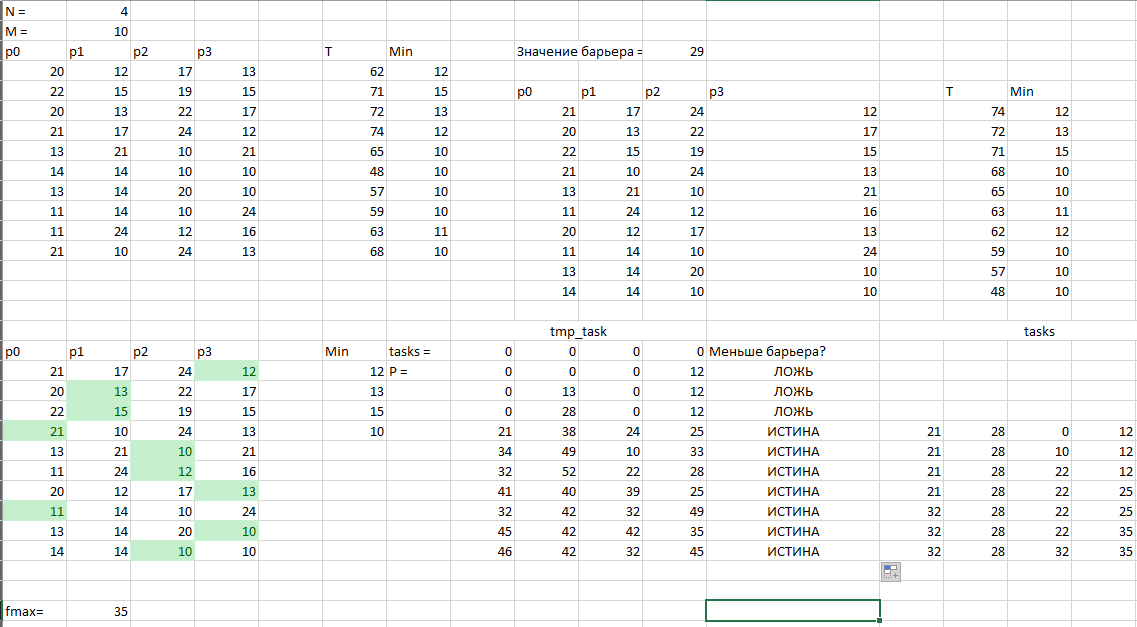
tN,1 tN,2 tN,3, … tN,M }

Известно время решения τ (ti,j) задания ti,j на любом из устройств. При этом каждое задание может выполняться на любом из устройств (процессоре). В каждый момент времени отдельный процессор обслуживает не более одного задания, и выполнение задания не прерывается для передачи на другой процессор. Требуется найти такое распределение заданий по процессорам, при котором суммарное время выполнения заданий на каждом из процессоров было бы минимальным, используя барьер. Барьер может рассчитываться по минимумам или максимумам значений времени выполнения заданий.

Под расписанием следует понимать отображение множества заданий T на множество процессоров P, то есть разбиение множества T на N непересекающихся подмножеств. Критерий, используемый для минимизации времени завершения обслуживания заданий, определяется следующим образом:

F = max(C1, C2, ..., CN), где Ci - время завершения работы процессора i.

Таким образом, цель задачи – минимизировать F путем оптимального распределения заданий между процессорами с учетом барьера.

**Ручной просчет**

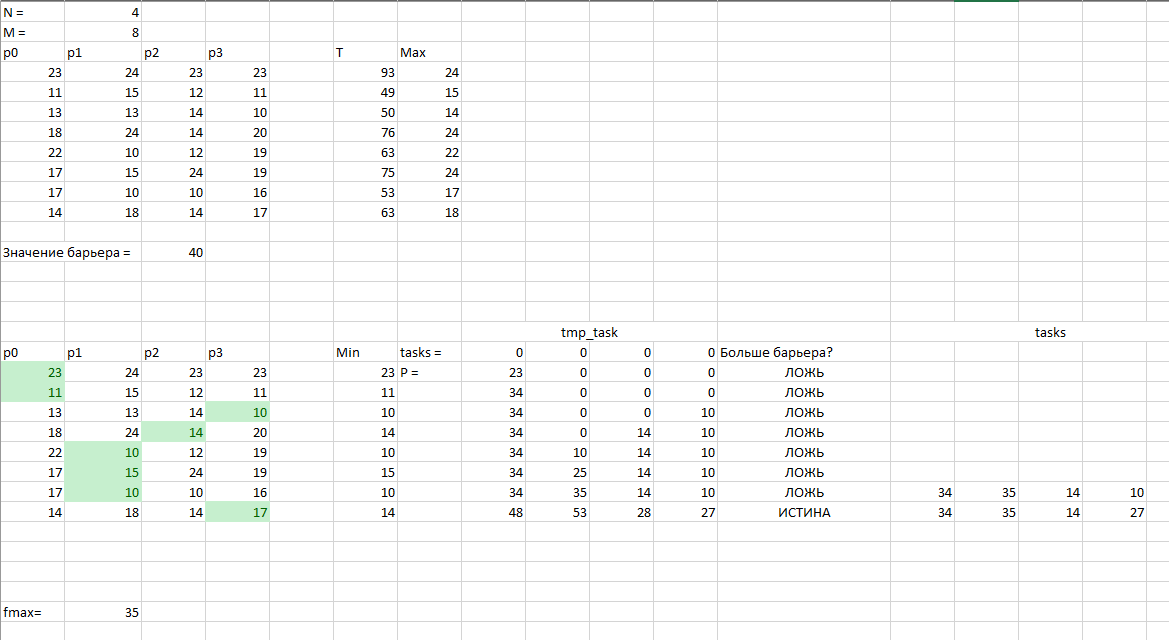
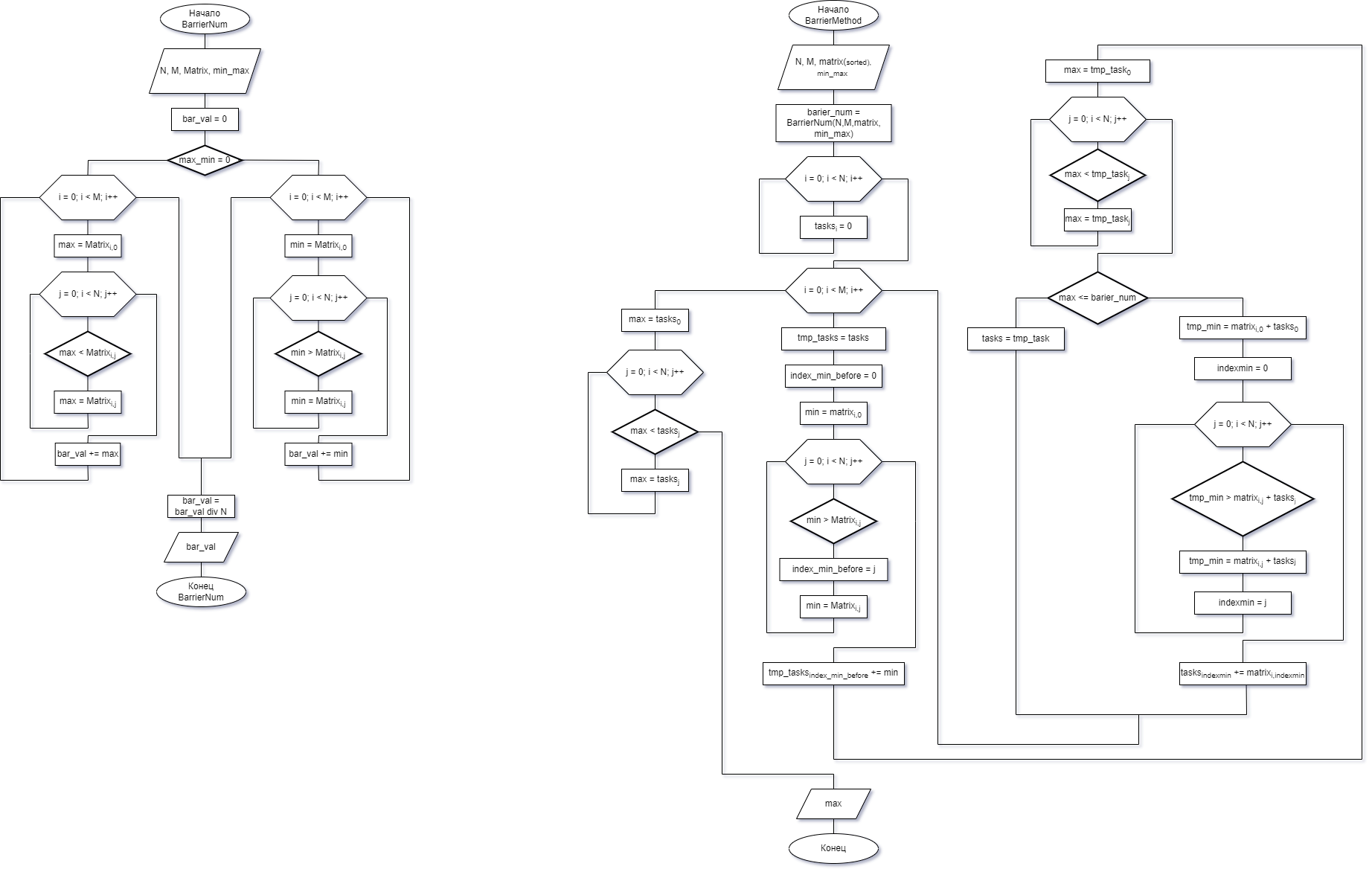
Рисунок 1 - Барьер по минимальным и сортировкой по убыванию

Рисунок 2 - Барьер по максимальным и с исходным порядком строк

**Блок-схема**

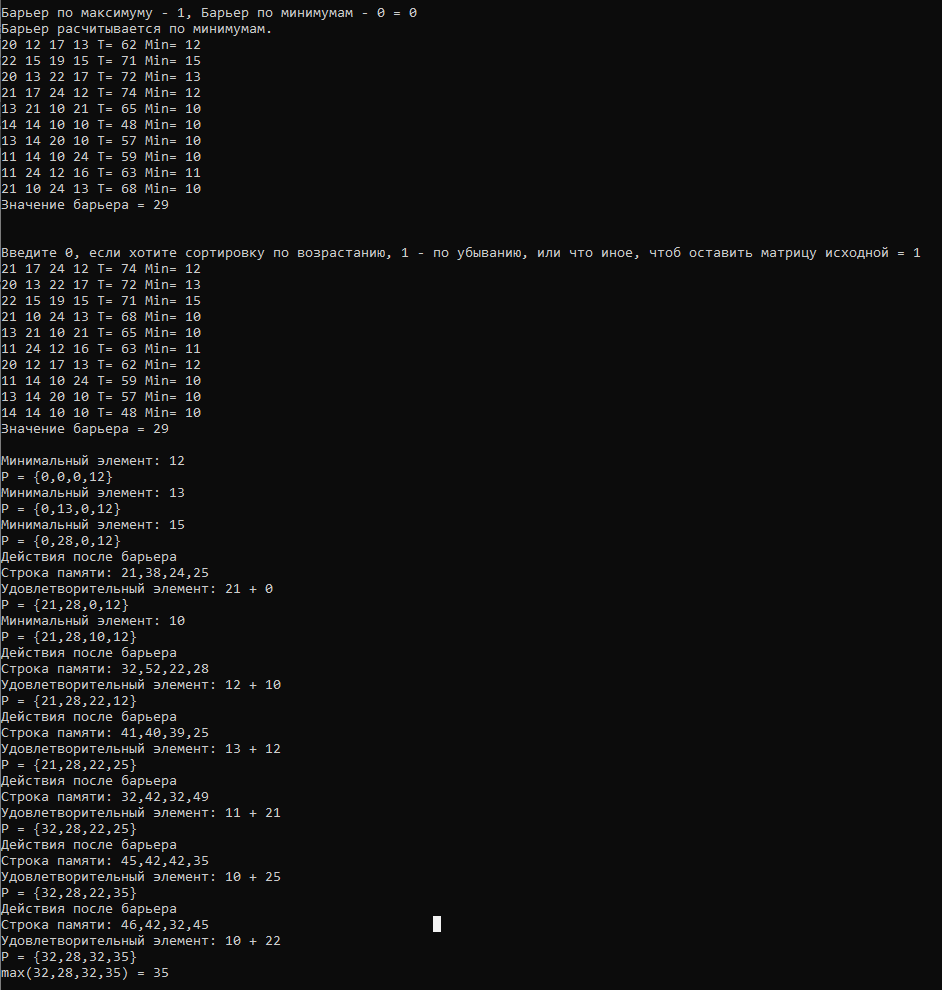
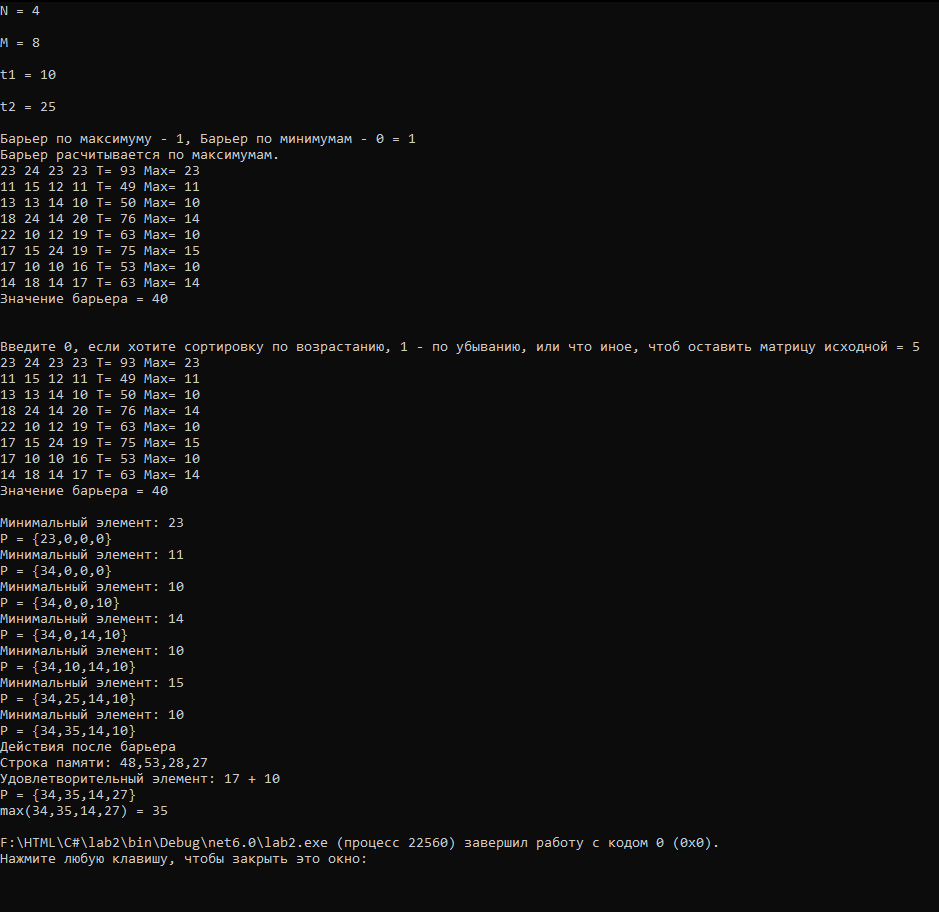
**Результат работы программы**

Рисунок 3 - Барьер по минимальным и сортировкой по убыванию

Рисунок 4 - Барьер по максимальным и с исходным порядком строк

**Вывод**

В ходе выполнения лабораторной работы была реализована программа, распределяющая набор заданий между процессорами с учетом барьера. Реализованный алгоритм позволяет находить оптимальное распределение, минимизируя максимальное время завершения работы процессора. Значительное влияние данного метода определяется выбранными методами для распределения заданий до барьера и после. Полученные результаты подтверждают эффективность предложенных методов распределения заданий.

**Список литературы**

1. Поспелов Д.А. “Введение в теорию вычислительных машин” – M.: “Советское радио”, 1972

2. Пашкеев С.Д. “Основы мультипрограммирования для специализированных вычислительных машин” – M.: “Советское радио”, 1964

3. Плотников В.Н., Зверев В.Ю. “Техническая кибернетика №3” M., 1974

4. Бондаренко А.Т., Сапатый П.С. “Техническая кибернетика №4” –Киев, 1975

**Листинг**

Листинг 1 – Метод подсчета барьера

public void BarrierNum(int max\_min)

{

int barr\_val = 0;

if (max\_min == 0)

{

Console.WriteLine("Барьер расчитывается по минимумам.");

foreach (var el in matrix)

{

barr\_val += el.Min();

}

}

else

{

Console.WriteLine("Барьер расчитывается по максимумам.");

foreach (var el in matrix)

{

barr\_val += el.Max();

}

}

barrier = (int)Math.Ceiling((double)barr\_val / N);

}

Листинг 2 – Использование барьера

public void Main()

{

foreach(var el in matrix)

{

int[] tmp\_task = tasks.ToArray();

var index\_min\_before = el.IndexOf(el.Min());

tmp\_task[index\_min\_before] += el.Min();

if (tmp\_task.Max() <= barier\_num) {

Console.WriteLine("Минимальный элемент: " + el.Min());

Console.WriteLine("P = {" + String.Join(",", tmp\_task) + "}");

tasks = tmp\_task.ToArray();

}

else

{

Console.WriteLine("Действия после барьера");

int tmp\_min = el[0] + tasks[0];

int indexmin = 0;

string answer = "";

for (int j = 0; j < N; j++)

{

answer += el[j] + tasks[j] + ",";

if (tmp\_min > (el[j] + tasks[j]))

{

tmp\_min = el[j] + tasks[j];

indexmin = j;

}

}

Console.WriteLine("Строка памяти: " + answer.Remove(answer.Length - 1));

Console.WriteLine("Удовлетворительный элемент: " + el[indexmin] + " + " + tasks[indexmin]);

tasks[indexmin] += el[indexmin];

Console.WriteLine("P = {" + String.Join(",", tasks) + "}");

}

}

Findmax();

}