Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра вычислительных технологий**

**Отчёт**

**по лабораторной работе №6**

**Дисциплина: ОСНОВЫ КОМПЬЮТЕРНОГО МОДЕЛИРОВАНИЯ**

Работу выполнила \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д. Н. Баева

Направление подготовки 02.03.02 Фундаментальная информатика и

информационные технологии

Направленность (профиль) Математическое и программное обеспечение

компьютерных технологий

Преподаватель

доцент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е. В. Кособуцкая

# **ПОСТАНОВКА ЗАДАЧИ**

Вычислительная система состоит из трех ЭВМ. С интервалом (3±1) мин. в систему поступают задания, которые с вероятностями: *P* = 0,4 идут на первую ЭВМ, с *Р* = 0,3 адресуются второй ЭВМ, а все остальные идут на обработку на третью ЭВМ. Перед каждой ЭВМ имеется очередь заданий, длина которой не ограничена. После обработки задания на первой ЭВМ оно с вероятностью *P* = 0,3 поступает в очередь ко второй ЭВМ и с вероятностью *P* = 0,7 – в очередь к третьей ЭВМ. После обработки на второй или третьей ЭВМ задание считается выполненным.

Продолжительность обработки заданий на разных ЭВМ характеризуется интервалами времени: *Т*1 = 4±1 мин, *Т*2 = 3±1 мин, *Т*3 = 5±2 мин.

Разработать программу, моделирующую процесс функционирования вычислительной системы при условии, что необходимо обработать 200 заданий.

Разработанная программа должна удовлетворять следующим требованиям:

* обеспечивать ввод исходных данных;
* поддерживать интерактивное редактирование;
* производить имитационного моделирование;
* представлять результаты моделирования в удобном виде.

# **Краткое описание разработанного алгоритма**

В рамках полученного задания рассматривается система массового обслуживания (СМО) с тремя каналами (ЭВМ) без отказов, так как длина очереди задач в ЭВМ не ограничена.

Для моделирования процесса функционирования вычислительной системы используется метод моделирования ∆t. Он заключается в том, что делается шаг по времени с определенным шагом (∆t), и на каждом шаге рассчитывается, что происходит в СМО.

Алгоритм работает, пока не выполнится последняя заявка.

В процессе работы алгоритма вычисляется enemy\_task - время, через которое произойдет одно из двух событий: одна из ЭВМ завершит выполнение текущей заявки или новая заявка поступит в систему. Далее время системы изменяется на enemy\_task. Если же заявка обрабатывается на 1-й ЭВМ, то записывается событие «ЭВМ N передана заявка M от ЭВМ 1», где N – номер машины, на которую передана заявка, а M – номер заявки. При завершении обработки заявки на 2-й или 3-й ЭВМ задание считается выполненным и заносится в журнал событий. Если поступила новая задача, то она отправляется на выполнение в одну из ЭВМ в соответствии с заданными вероятностями.

В программе предусмотрены два вида запуска:

* В качестве входных данных подаются количество заданий, вероятности и интервалы времени.
* Система начинает работу на начальных значениях из условия задачи.

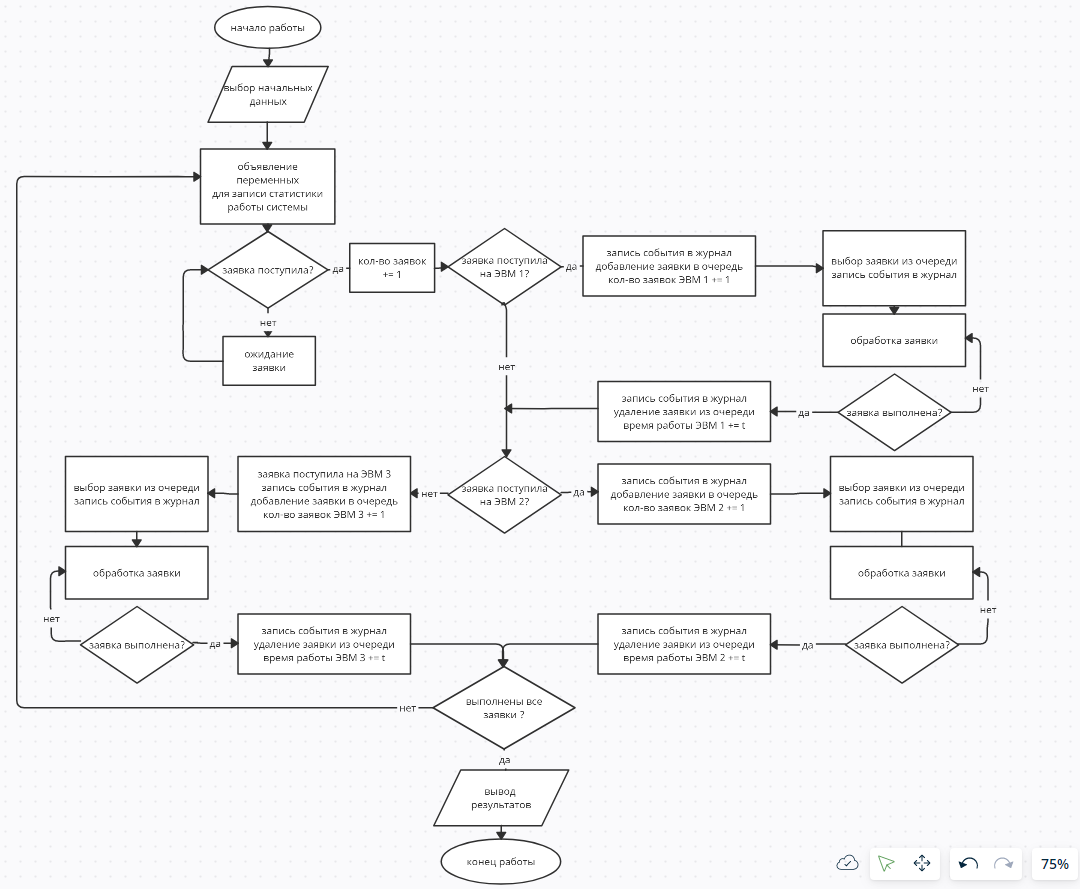
На рисунке 1 представлен полный алгоритм работы СМО.

Рисунок 1 – Алгоритм работы системы массового обслуживания

# **ТЕСТОВЫЕ ПРИМЕРЫ И РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ ПРОГРАММЫ**

Тестовые примеры и результаты тестирования программы для поиска оптимизированного распределения представлены в таблице 1 и таблице 2 соответственно.

Таблица 1 – Результаты работы программы на исходных данных из условия задачи

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Исходные данные:**  Количество заявок: 200  Интервал поступления заявок: 3±1минут  Продолжительность обработки заявки на ЭМВ №1: 4±1 минут  Продолжительность обработки заявки на ЭМВ №2: 3±1 минут  Продолжительность обработки заявки на ЭМВ №3: 5±2 минут  Вероятность поступления заявки на ЭВМ №1: 0.4  Вероятность поступления заявки на ЭВМ №2: 0.3  Вероятность поступления заявки на ЭВМ №3: 0.3  Вероятность поступления заявки с ЭВМ №1 на ЭВМ №2: 0.3  Вероятность поступления заявки с ЭВМ №1 на ЭВМ №3: 0.7  **Полученные результаты:** | | | | | |
|  | ЭВМ №1 | ЭВМ №2 | ЭВМ №3 | Общая статистика работы системы | |
| Кол-во заявок, шт. | 62 | 96 | 104 | Время работы системы, мин. | 751.03 |
| Общее время работы, мин. | 746.18 | 754.84 | 753.74 | Среднее число каналов в обслуживании, шт. | 1.04 |
| Простой ЭВМ, % | 79.81 | 74.81 | 52.49 | Среднее число заявок в очереди, шт. | 0 |
| Коэффициент занятости ЭВМ, % | 20 | 25 | 48 | Среднее число заявок в системе, шт. | 1 |
| Среднее время ожидания в очереди, мин | 0.00 | 0.02 | 0.36 | Среднее время пребывания заявки в системе, мин. | 5.94 |
|  |  |  |  | Среднее время ожидания в очереди, мин. | 0.13 |

**Рекомендации по улучшению:**

1. Изменить интервал поступления заявок в систему;
2. Увеличить продолжительность обработки заявки на каждой ЭВМ;
3. Направить больше заявок на ЭВМ\_2 и ЭВМ\_3, так как ЭВМ\_1 работает, но все равно отдает заявки на ЭВМ\_2 и ЭВМ\_3;
4. Изменить распределение из ЭВМ\_1 в ЭВМ\_2 и ЭВМ\_3 на [0.8:0.2], так как ЭВМ\_2 работает быстрее ЭВМ\_3;

Таблица 2 – Результаты работы программы на изменённых данных, с учётом рекомендаций, указанных в таблице 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Измененные данные:**  Количество заявок: 200  **Изменяем интервал поступления заявок, на 1 минуту быстрее:**  Интервал поступления заявок: 2±2минут  **Улучшаем машины, каждая из ЭВМ будет работать на 1 минуту быстрее:**  Продолжительность обработки заявки на ЭВM №1: 3±1 минут  Продолжительность обработки заявки на ЭВМ №2: 2±1 минут  Продолжительность обработки заявки на ЭВМ №3: 4±1 минут  **Изменяем вероятность попадания на ЭВМ:**  Вероятность поступления заявки на ЭВМ №1: 0.3  Вероятность поступления заявки на ЭВМ №2: 0.3  Вероятность поступления заявки на ЭВМ №3: 0.4  **Увеличиваем вероятность поступления заявки из ЭВМ №1 в ЭМВ №2:**  Вероятность поступления заявки с ЭВМ №1 на ЭВМ №2: 0.7  Вероятность поступления заявки с ЭВМ №1 на ЭВМ №3: 0.3  **Полученные результаты:** | | | | | |
|  | ЭВМ №1 | ЭВМ №2 | ЭВМ №3 | Общая статистика работы системы | |
| Кол-во заявок, шт | 60 | 105 | 95 | Время работы системы, мин | 523.6 |
| Общее время работы, мин. | 509.11 | 522.5 | 525.6 | Среднее число каналов в обслуживании, шт. | 1.08 |
| Простой ЭВМ, % | 75.82 | 69.8 | 52.51 | Среднее число заявок в очереди, шт. | 0 |
| Коэффициент занятости ЭВМ, % | 24 | 30 | 47 | Среднее число заявок в системе, шт. | 1 |
| Среднее время ожидания в очереди, мин | 0.0 | 0.0 | 0.69 | Среднее время пребывания заявки в системе, мин. | 3.95 |
|  |  |  |  | Среднее время ожидания в очереди, мин. | 0.23 |

**Результаты изменений:**

1. Уменьшилось время работы системы на 227,43 минут
2. Уменьшилось время работы каждой ЭВМ на 237.07 мин., 232.34 мин., 231,14 мин. соответственно;
3. Уменьшился процент простоя каждой ЭВМ и соответственно вырос коэффициент занятости каждой ЭВМ;
4. Уменьшилось среднее время пребывания заявки в системе на 1,99 мин.

**Дополнительные рекомендации по улучшению:**

1. Улучшить ЭВМ или купить новые;
2. Отказаться от системы двойных обработок заявок (с ЭВМ №1 на ЭВМ №2 и ЭВМ №3).

# **ЗАКЛЮЧЕНИЕ**

Результаты моделирования при проведении тестирования программы подтвердили следующую гипотезу для базовой точки эксперимента – после уменьшения времени обработки заявок, изменения начального распределения по ЭВМ, изменения распределения при передаче заявок из ЭВМ №1 производительность системы значительно увеличилась.

Производительность можно увеличить еще больше, если уменьшить время поступления заявок и отказаться от двойной обработки – т. е. чтобы ЭВМ №1 сразу завершала работу и не передавала заявки другим ЭВМ, либо отказаться от ЭВМ №1 вовсе, заменив ее на копию другой.

# ПРИЛОЖЕНИЕ А

**Листинг программы с комментариями**

Язык программирования – Python. Версия – 3.10

import random

import numpy as np

class Event:

def \_\_init\_\_(self, name, time):

self.name = name

self.time = time

def show(self):

print(self.name, '--', self.time)

f = int(input("Выберите функцию: \n 1. Ввести с клавиатуры; \n 2. Запустить стандартные значения. \n"))

if f == 1:

print('Введите последовательно следующую информацию:')

print()

tasks = int(input('Количество заданий: '))

# стандартное значение = 200

weight\_1, weight\_2, weight\_3 = map(float, input('Веса распределения на три ЭВМ, через пробел: ').split())

# стандартные значения = 0.4 0.3 0.3

weight\_evm\_1\_to\_2, weight\_evm\_1\_to\_3 = map(float,

input(

'Веса распределения из ЭВМ\_1 в ЭВМ\_2 и ЭВМ\_3, через пробел: ').split())

# стандартные значения = 0.3 0.7

a1, b1 = map(int, input('Интервалы для ЭВМ\_1, через пробел: ').split())

# стандартные значения = 3 5

a2, b2 = map(int, input('Интервалы для ЭВМ\_2, через пробел: ').split())

# стандартные значения = 2 4

a3, b3 = map(int, input('Интервалы для ЭВМ\_3, через пробел: ').split())

# стандартные значения = 3 7

elif f == 2:

# количество заданий

tasks = 200

# вероятность поступления заданий

weight\_1 = 0.4

weight\_2 = 0.3

weight\_3 = 0.3

# вероятность перехода заданий на другую эвм

weight\_evm\_1\_to\_2 = 0.3

weight\_evm\_1\_to\_3 = 0.7

# интервалы времени

a1 = 4

a2 = 3

a3 = 5

b1 = 1

b2 = 1

b3 = 2

else:

print("Ошибка! Некорректно введены данные.")

events = []

now\_time = 0 # текущее время

# глобальные переменные для ЭВМ 1

evm\_1\_total\_tasks = 0 # общее время выполнения заданий

evm\_1\_total\_time = 0 # общее время работы

evm\_1\_last = 0 # время обработки последней заявки

evm\_1\_relax = 0 # время простоя

evm\_1\_total\_relax = 0 # общее время простоя ЭВМ

evm\_1\_total\_queue = 0 # общее время простоя ЭВМ в очереди

evm\_1\_total = 0

# глобальные переменные для ЭВМ 2

evm\_2\_total\_tasks = 0 # общее время выполнения заданий

evm\_2\_total\_time = 0 # общее время работы

evm\_2\_last = 0 # время обработки последней заявки

evm\_2\_relax = 0 # время простоя

evm\_2\_total\_relax = 0 # общее время простоя ЭВМ

evm\_2\_total = 0

evm\_2\_total\_queue = 0 # общее время простоя ЭВМ в очереди

flag\_2 = 0

# глобальные переменные для ЭВМ 3

evm\_3\_total\_tasks = 0 # общее время выполнения заданий

evm\_3\_total\_time = 0 # общее время работы

evm\_3\_last = 0 # время обработки последней заявки

evm\_3\_relax = 0 # время простоя

evm\_3\_total\_relax = 0 # общее время простоя ЭВМ

evm\_3\_total = 0

evm\_3\_total\_queue = 0 # общее время простоя ЭВМ в очереди

flag\_3 = 0

evm\_1\_w = 0

evm\_2\_w = 0

evm\_3\_w = 0

def evm\_1(task\_number):

global now\_time, evm\_1\_relax, evm\_1\_last, evm\_1\_total\_time, evm\_1\_total\_relax, evm\_1\_total\_tasks, evm\_1\_total\_queue

# количество заявок

evm\_1\_total\_tasks += 1

# время простоя

evm\_1\_relax = now\_time - evm\_1\_last

if evm\_1\_relax < 0:

evm\_1\_relax = 0

# если нужно время ожидания между заявками, то выводим каждый раз evm\_1\_relax

#print('Интервалы между заявками для ЭВМ1:',round(evm\_1\_relax,2))

# общее время простоя

evm\_1\_total\_relax += evm\_1\_relax

# поступление заявки

task\_start = Event('ЭВМ 1 поступила заявка {}'.format(task\_number), now\_time)

start\_queue = now\_time

# время выполнения заявки

evm\_1\_work = np.random.uniform(a1, b1)

global evm\_1\_w

evm\_1\_w = evm\_1\_work

# время конца работы

# если заявка есть

if evm\_1\_last > now\_time:

evm\_1\_last = evm\_1\_last + evm\_1\_work

# если заявки нет

else:

evm\_1\_last = now\_time + evm\_1\_work

# выполнение заявки

task\_end = Event('ЭВМ 1 выполнена заявка {}'.format(task\_number), evm\_1\_last)

end\_queue = evm\_1\_last

evm\_1\_total\_queue += end\_queue - start\_queue

# общее время работы = время всего простоя + время всей работы

evm\_1\_total\_time = evm\_1\_total\_time + evm\_1\_relax + evm\_1\_work

# заносим события в список

events.append(task\_start)

events.append(task\_end)

# передаем задачу другой ЭВМ

next\_evm\_number = random.choices((2, 3), weights=[weight\_evm\_1\_to\_2, weight\_evm\_1\_to\_3])

# ЭВМ 2

if next\_evm\_number == [2]:

evm\_2(i, 1, evm\_1\_last)

# ЭВМ 3

elif next\_evm\_number == [3]:

evm\_3(i, 1, evm\_1\_last)

def evm\_2(task\_number, enemy\_task, enemy\_time):

global now\_time, evm\_2\_relax, evm\_2\_last, evm\_2\_total\_time, evm\_2\_total\_relax, evm\_2\_total\_tasks, evm\_2\_total\_queue

# кол-во заявок

evm\_2\_total\_tasks += 1

# поступила задача от ЭВМ 1

if enemy\_task == 1:

now\_time = enemy\_time

# время простоя

evm\_2\_relax = now\_time - evm\_2\_last

if evm\_2\_relax < 0:

evm\_2\_relax = 0

# если нужно время ожидания между заявками, то выводим каждый раз evm\_2\_relax

#print('Интервалы между заявками для ЭВМ1:',round(evm\_2\_relax,2))

# общее время простоя

evm\_2\_total\_relax += evm\_2\_relax

# поступление заявки

if enemy\_task == 0:

task\_start = Event('ЭВМ 2 поступила заявка {}'.format(task\_number), now\_time)

else:

task\_start = Event('ЭВМ 2 передана заявка {} от ЭВМ 1'.format(task\_number), now\_time)

start\_queue = now\_time

# время выполнения заявки

evm\_2\_work = np.random.uniform(a2, b2)

# evm\_2\_work = 10

global evm\_2\_w

evm\_2\_w = evm\_2\_work

# время конца работы

# если заявка есть

if evm\_2\_last > now\_time:

evm\_2\_last = evm\_2\_last + evm\_2\_work

# если заявки нет

else:

evm\_2\_last = now\_time + evm\_2\_work

# выполнение заявки

if enemy\_task == 0:

task\_end = Event('ЭВМ 2 выполнена заявка {}'.format(task\_number), evm\_2\_last)

else:

task\_end = Event('ЭВМ 2 выполнена переданная заявка {} от ЭВМ 1'.format(task\_number), evm\_2\_last)

end\_queue = evm\_2\_last

evm\_2\_total\_queue += end\_queue - start\_queue

# общее время работы = время всего простоя + время всей работы

evm\_2\_total\_time = evm\_2\_total\_time + evm\_2\_relax + evm\_2\_work

# заносим события в список

events.append(task\_start)

events.append(task\_end)

def evm\_3(task\_number, enemy\_task, enemy\_time):

global now\_time, evm\_3\_relax, evm\_3\_last, evm\_3\_total\_time, evm\_3\_total\_relax, evm\_3\_total\_tasks, evm\_3\_total\_queue

# количество заявок

evm\_3\_total\_tasks += 1

# поступила задача от ЭВМ 1

if enemy\_task == 1:

now\_time = enemy\_time

# время простоя

evm\_3\_relax = now\_time - evm\_3\_last

if evm\_3\_relax < 0:

evm\_3\_relax = 0

# если нужно время ожидания между заявками, то выводим каждый раз evm\_3\_relax

#print('Интервалы между заявками для ЭВМ1:',round(evm\_3\_relax,2))

# общее время простоя

evm\_3\_total\_relax += evm\_3\_relax

# поступление заявки

if enemy\_task == 0:

task\_start = Event('ЭВМ 3 поступила заявка {}'.format(task\_number), now\_time)

else:

task\_start = Event('ЭВМ 3 передана заявка {} от ЭВМ 1'.format(task\_number), now\_time)

start\_queue = now\_time

# время выполнения заявки

evm\_3\_work = np.random.uniform(a3, b3)

global evm\_3\_w

evm\_3\_w = evm\_3\_work

# время конца работы

# если заявка есть

if evm\_3\_last > now\_time:

evm\_3\_last = evm\_3\_last + evm\_3\_work

# если заявки нет

else:

evm\_3\_last = now\_time + evm\_3\_work

# выполнение заявки

if enemy\_task == 0:

task\_end = Event('ЭВМ 3 выполнена заявка {}'.format(task\_number), evm\_3\_last)

else:

task\_end = Event('ЭВМ 3 выполнена переданная заявка {} от ЭВМ 1'.format(task\_number), evm\_3\_last)

end\_queue = evm\_3\_last

evm\_3\_total\_queue += end\_queue - start\_queue

# общее время работы = время всего простоя + время всей работы

evm\_3\_total\_time = evm\_3\_total\_time + evm\_3\_relax + evm\_3\_work

# заносим события в список

events.append(task\_start)

events.append(task\_end)

for i in range(1, tasks + 1):

task\_interval = np.random.uniform(2, 4)

global tsk\_interval

tsk\_interval = task\_interval

evm\_number = random.choices((1, 2, 3), weights=[weight\_1, weight\_2, weight\_3])

# добавляем время выдачи задачи

now\_time += task\_interval

# ЭВМ 1

if evm\_number == [1]:

evm\_1(i)

# ЭВМ 2

elif evm\_number == [2]:

evm\_2(i, 0, 'str')

# ЭВМ 3

elif evm\_number == [3]:

evm\_3(i, 0, 'str')

events\_sorted = sorted(events, key=lambda x: x.time, reverse=True)

for i in events\_sorted:

i.show()

l1 = weight\_1/a1

l2 = weight\_2/a2

l3 = weight\_3/a3

# суммарная интенсивность потока заявок для всех эвм

l = l1 + l2 + l3

# интенсивности обслуживания на каждой ЭВМ

fi1 = 1/a1

fi2 = 1/a2

fi3 = 1/a3

# средние загрузки каждой из ЭВМ

ro = l/(fi1+fi2+fi3)

ro1 = l1/fi1

ro2 = l2/fi2

ro3 = l3/fi3

# среднее количество каналов в обслуживании

served\_chanel = l\*(weight\_1\*a1+weight\_2\*a2+weight\_3\*a3)

# среднее время, которое заявка проводит в очереди

Wq = (ro1\*\*2+ro2\*\*2+ro3\*\*2)/(2\*(1-ro1)\*l)

# среднее число заявок в очереди

task\_in\_queue = l\*Wq

lis = l\*served\_chanel

# среднее число заявок в системе

task\_in\_system = task\_in\_queue + lis

print('')

print('Статистика ЭВМ 1:')

print('Количество заявок:', evm\_1\_total\_tasks)

print('Общее время работы:', round(evm\_1\_total\_time,2),'мин.')

print('Простой ЭВМ:', round((round(evm\_1\_total\_relax,2)/round(evm\_1\_total\_time,2))\*100,2),'%')

print('Коэффициент занятости ЭВМ:', round(( (evm\_1\_total\_time - evm\_1\_total\_relax) / evm\_1\_total\_time),2)\*100, '%')

# мы больше ждем заявок, чем они ждут в очереди, это можно увидеть во времени простоя

queue\_1 = float(abs(evm\_1\_total\_time - evm\_1\_total\_relax - evm\_1\_total\_queue) / evm\_1\_total\_tasks)

print('Среднее время ожидания в очереди', round(queue\_1,3),'мин.')

print('')

print('Статистика ЭВМ 2:')

print('Количество заявок:', evm\_2\_total\_tasks)

print('Общее время работы:', round(evm\_2\_total\_time,2),'мин.')

print('Простой ЭВМ:', round((round(evm\_2\_total\_relax,2)/round(evm\_2\_total\_time,2))\*100,2),'%')

print('Коэффициент занятости ЭВМ:', round(((evm\_2\_total\_time - evm\_2\_total\_relax) / evm\_2\_total\_time),2)\*100, '%')

# мы больше ждем заявок, чем они ждут в очереди, это можно увидеть во времени простоя

queue\_2 = float(abs(evm\_2\_total\_time - evm\_2\_total\_relax - evm\_2\_total\_queue) / evm\_2\_total\_tasks)

print('Среднее время ожидания в очереди', round(queue\_2,2),'мин.')

print('')

print('Статистика ЭВМ 3:')

print('Количество заявок:', evm\_3\_total\_tasks)

print('Общее время работы:', round(evm\_3\_total\_time,2),'мин.')

print('Простой ЭВМ:', round((round(evm\_3\_total\_relax,2)/round(evm\_3\_total\_time,2))\*100,2),'%')

print('Коэффициент занятости ЭВМ:', round(((evm\_3\_total\_time - evm\_3\_total\_relax) / evm\_3\_total\_time),2)\*100, '%')

# мы больше ждем заявок, чем они ждут в очереди, это можно увидеть во времени простоя

queue\_3 = float(abs(evm\_3\_total\_time - evm\_3\_total\_relax - evm\_3\_total\_queue) / evm\_3\_total\_tasks)

print('Среднее время ожидания в очереди', round(queue\_3,2),'мин.')

print('')

print("Общая статистика работы системы:")

print('Время работы системы:', round(events\_sorted[0].time - events\_sorted[len(events\_sorted) - 1].time,2),'мин.')

print('Среднее число каналов в обслуживании:', round(served\_chanel,2))

print('Среднее число заявок в очереди:', round(task\_in\_queue))

print('Среднее число заявок в системе:',round(task\_in\_system))

print('Среднее время пребывания заявки в системе:',round( (evm\_1\_total\_relax + evm\_2\_total\_relax + evm\_3\_total\_relax) / (

evm\_1\_total\_tasks + evm\_2\_total\_tasks + evm\_3\_total\_tasks),2),'мин.')

print('Среднее время ожидания в очереди', round (((queue\_1 + queue\_2 + queue\_3) / 3.0),2),'мин.')

#print('Эффективность', tasks / (events\_sorted[0].time - events\_sorted[len(events\_sorted) - 1].time))