МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.ШУХОВА»

(БГТУ им. В.Г.Шухова)

**Лабораторная работа № 4.2**

Дисциплина: Теория надежности

Выполнил: студент группы ВТ-31

Ковалёв И. Д.

Проверил:

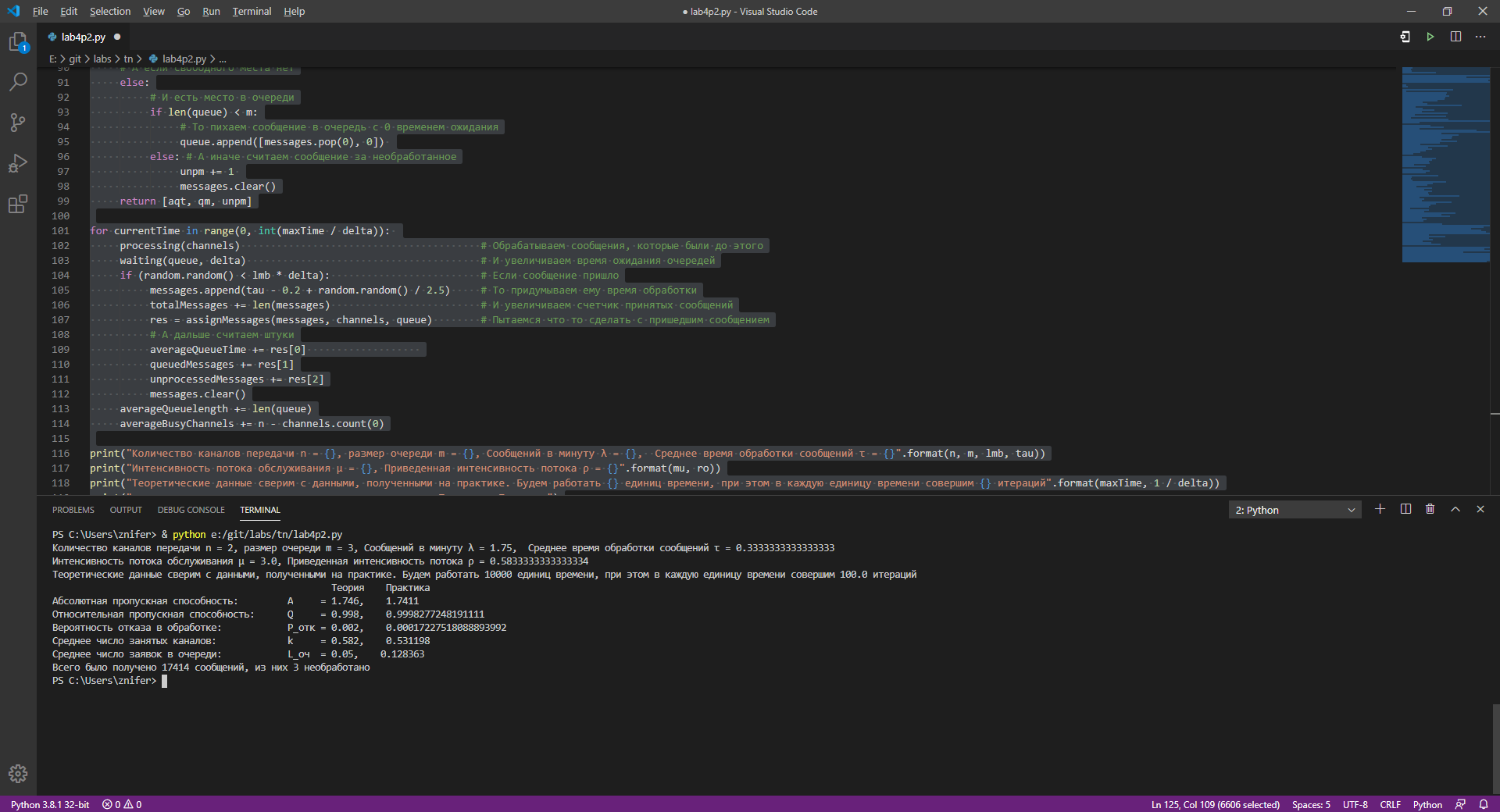
Кабалянц П. С.

Белгород 2020

**Задание к работе:**

На сервере есть n каналов передачи сообщений. Среднее время обработки сообщений τ. На сервер поступают сообщения в среднем количестве λ сообщений в минуту. Для хранения сообщений в очереди на сервере выделено место для (n+1) сообщения. Определить основные характеристики сервера - вероятность очереди, среднее число занятых каналов, средняя длина очереди, среднее число сообщений на сервере; предполагается, что сообщение не получает отказ при занятости всех каналов и очереди длины не больше (n+1). Параметры варианта определяются по формулам: n=3+[(i+j)/8], λ=1+i/4, τ=5/(5+j). Здесь квадратные скобки означают взятие целой части, а i,j - последние цифры зачетки.

**Скриншот разработанной программы:**



**Листинг программы:**

import random

import math

import numpy

from scipy.special import factorial as fact

#На сервере есть n каналов передачи сообщений. Среднее время обработки сообщений τ. На сервер поступают сообщения в среднем количестве λ сообщений в минуту.

# Для хранения сообщений в очереди на сервере выделено место для (n+1) сообщения.

# Определить основные характеристики сервера - вероятность очереди, среднее число занятых каналов, средняя длина очереди, среднее число сообщений на сервере;

# предполагается, что сообщение не получает отказ при занятости всех каналов и очереди длины не больше (n+1).

# Параметры варианта определяются по формулам: n=3+[(i+j)/8], λ=1+i/4, τ=5/(5+j). Здесь квадратные скобки означают взятие целой части, а i,j - последние цифры зачетки.

i = 3

j = 10

#n = 3 + ((i + j) // 8)   # Количество каналов передачи сообщений

n = 2

m = n + 1                # Количество сообщений в очереди

lmb = 1 + i / 4          # Количество сообщений в минуту

tau = 5 / (5 + j)        # Среднее время обработки сообщений

mu = 1 / tau        # Интенсивность потока обслуживания

ro = lmb / mu       # Приведенная интенсивность потока

P = [0]              # Предельные вероятности

for i in range (0, n + 1):

     P[0] += (ro \*\* i) / fact(i)

P[0] += (ro \*\* (n + 1)) \* (1 - (ro / n) \*\* m) / (n \* fact(n) \* (1 - ro / n))

P[0] = P[0] \*\* -1

for i in range (1, n + 1):

     P.append((ro \*\* i) / fact(i) \* P[0])

for r in range(1, m + 1):

     P.append((ro \*\* (n + r)) / (n \*\* r \* fact(n)) \* P[0])

P\_otk = P[n + m]

Q = 1 - (((ro \*\*(n + m)) / ((n \*\* m) \* fact(n))) \* P[0])

A = lmb \* Q

k = ro \* (1 - ((ro \*\* (n + m))/ (n \*\* m \* fact(n))) \* P[0])

L\_queue = (ro \*\* (n + 1) \* P[0] \* (1 - (m + 1 - m \* (ro / n)) \* (ro / n) \*\* m)) / (n \* fact(n) \* (1 - ro / n) \*\* 2)

L\_system = L\_queue + k

messages = []            # Сообщения

channels = []            # Каналы обработки сообщений

for i in range (0, n):

     channels.append(0)  # Если 0 - канал свободен, иначе - время, которое будет обрабатываться

queue = []               # Очередь;  ервый элемент - время на обработку сообщения, второй элемент - время, которое сообщение провело в очереди

maxTime = 10000           # Сколько будем работать

delta = 0.01              # Сколько итераций обработки в единицу времени

totalMessages = 0        # Сколько всего было получено сообщений

unprocessedMessages = 0  # Сколько сообщений не было обработано

averageQueueTime = 0     # Среднее время в очереди

queuedMessages = 0       # Сколько сообщений было извлечено из очереди

averageQueuelength = 0   # Средняя длина очереди

averageBusyChannels = 0

# Возвращает номер свободного канала, если такого нет - -1

def isAnyEmptyChannels(channels):

     for i in range(0, len(channels)):

          if channels[i] == 0:

               return i

     return -1

def processing(channels):

     # Обрабатываем уже имеющиеся сообщения

     for i in range(0, len(channels)):

          if channels[i] >= delta:

               channels[i] -= delta

          else:

               channels[i] = 0

def waiting(queue, delta):

     for i in range(0, len(queue)):

          queue[i][1] += delta

def assignMessages(messages, channels, queue):

     aqt = 0

     qm = 0

     unpm = 0

     # Проверим, есть ли свободное место в каналах связи

     pos = isAnyEmptyChannels(channels)

     # Если свободное место есть

     if pos != -1:

          # И есть сообщение в очереди

          if len(queue) > 0:

               # Достаем это сообщение

               tmp = queue.pop(0)

               newMessage = tmp[0]

               # И учитываем, сколько времени сообщение провело в очереди

               qm += 1

               aqt += tmp[1]

          else: # А иначе

               # Достаем сообщение, полученное на этой итерации

               newMessage = messages.pop(0)

          channels[pos] = newMessage

     # А если свободного места нет

     else:

          # И есть место в очереди

          if len(queue) < m:

               # То пихаем сообщение в очередь с 0 временем ожидания

               queue.append([messages.pop(0), 0])

          else: # А иначе cчитаем сообщение за необработанное

               unpm += 1

               messages.clear()

     return [aqt, qm, unpm]

for currentTime in range(0, int(maxTime / delta)):

     processing(channels)                                        # Обрабатываем сообщения, которые были до этого

     waiting(queue, delta)                                       # И увеличиваем время ожидания очередей

     if (random.random() < lmb \* delta):                         # Если сообщение пришло

          messages.append(tau - 0.2 + random.random() / 2.5)     # То придумываем ему время обработки

          totalMessages += len(messages)                         # И увеличиваем счетчик принятых сообщений

          res = assignMessages(messages, channels, queue)        # Пытаемся что то сделать с пришедшим сообщением

          # А дальше считаем штуки

          averageQueueTime += res[0]

          queuedMessages += res[1]

          unprocessedMessages += res[2]

          messages.clear()

     averageQueuelength += len(queue)

     averageBusyChannels += n - channels.count(0)

print("Количество каналов передачи n = {}, размер очереди m = {}, Сообщений в минуту λ = {},  Среднее время обработки сообщений τ = {}".format(n, m, lmb, tau))

print("Интенсивность потока обслуживания μ = {}, Приведенная интенсивность потока ρ = {}".format(mu, ro))

print("Теоретические данные сверим с данными, полученными на практике. Будем работать {} единиц времени, при этом в каждую единицу времени совершим {} итераций".format(maxTime, 1 / delta))

print("                                                   Теория    Практика")

print("Абсолютная пропускная способность:         A     = {},    {}".format(round(A, 3), (totalMessages - unprocessedMessages) / maxTime))

print("Относительная пропускная способность:      Q     = {},    {}".format(round(Q, 3), (totalMessages - unprocessedMessages) / totalMessages))

print("Вероятность отказа в обработке:            P\_отк = {},    {}".format(round(P\_otk, 3), unprocessedMessages / totalMessages))

print("Среднее число занятых каналов:             k     = {},    {}".format(round(k, 3), averageBusyChannels / (maxTime / delta)))

print("Среднее число заявок в очереди:            L\_оч  = {},    {}".format(round(L\_queue, 3), averageQueuelength / (maxTime / delta)))

print("Всего было получено {} сообщений, из них {} необработано".format(totalMessages, unprocessedMessages))