|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

**ОТЧЕТ**

*к лабораторной работе №5*

*По курсу: «Моделирование»*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ***ИУ7И-76Б*** |  |  | **Нгуен Ф. С.** |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | |  | | --- | | **Рудаков И.В.** | |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

*Москва, 2021 г.*

Оглавление

[**I.** **Задача** 3](#_Toc91382625)

[**II.** **Теоретическая часть** 3](#_Toc91382626)

[**III.** **Экспериментальная часть** 4](#_Toc91382627)

[**IV.** **Код программы:** 6](#_Toc91382628)

1. Задача

В информационный центр приходят клиенты через интервал времени 10 +- 2 минуты. Если все три имеющихся оператора заняты, клиенту отказывают в обслуживании. Операторы имеют разную производительность и могут обеспечивать обслуживание среднего запроса пользователя за 20 +- 5; 40 +- 10; 40 +- 20. Клиенты стремятся занять свободного оператора с максимальной производительностью. Полученные запросы сдаются в накопитель. Откуда выбираются на обработку. На первый компьютер запросы от 1 и 2-ого операторов, на второй – запросы от 3-его. Время обработки запросов первым и 2-м компьютером равны соответственно 15 и 30 мин. Промоделировать процесс обработки 300 запросов.

1. Теоретическая часть

Необходимо создать концептуальную модель в терминах СМО, определить эндогенные и экзогенные переменные и уравнения модели.

За единицу системного времени выбрать 0,01 минуты.



В процессе взаимодействия клиентов с информационным центром возможно:

1) Режим нормального обслуживания, т.е. клиент выбирает одного из свободных операторов, отдавая предпочтение тому у которого меньше номер.

2) Режим отказа в обслуживании клиента, когда все операторы заняты

Переменные и уравнения имитационной модели.

Эндогенные переменные: время обработки задания i-ым оператором, время решения этого задания j-ым компьютером.

Экзогенные переменные: число обслуженных клиентов и число клиентов получивших отказ.

На рисунке представлена концептуальная модель в терминах СМО.



Вероятность отказа:

|  |  |
| --- | --- |
|  | (1) |

1. Экспериментальная часть

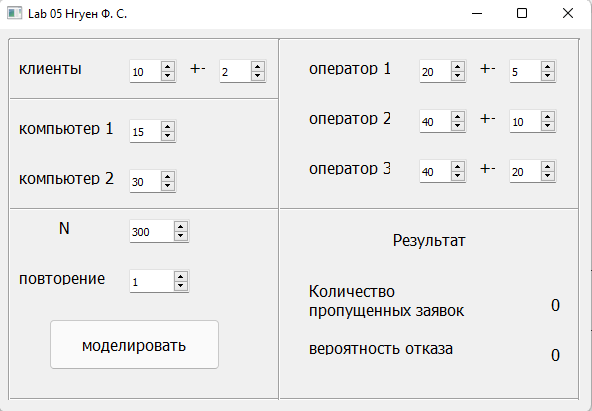


Рисунок 1. Интерфейс программы

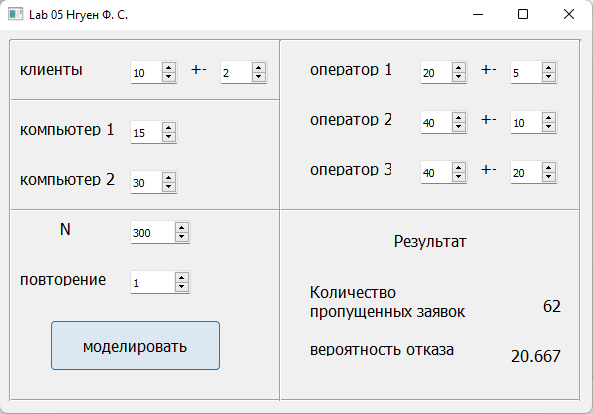


Рисунок 2. пример работы программы

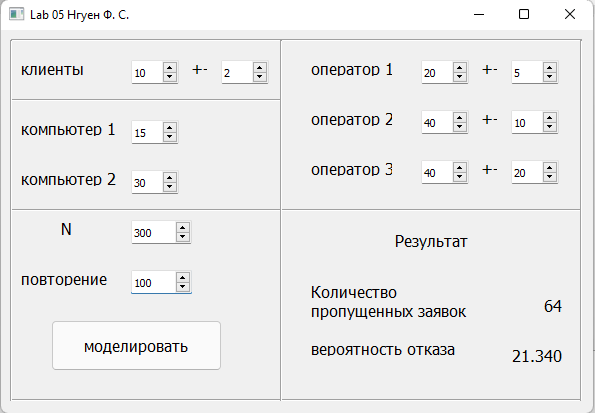


Рисунок 3. пример работы программы с повторением 100 раз

1. **Код программы:**

import numpy.random as rn

from PyQt5 import QtWidgets, uic

from PyQt5.QtWidgets import QTableWidgetItem

from PyQt5.QtGui import QPen, QColor, QImage, QPixmap, QPainter

from PyQt5.QtCore import Qt, QTime, QCoreApplication, QEventLoop, QPointF

class RandomGenerator:

def \_\_init\_\_(self, begin, delta=0):

self.begin = begin

self.d = delta

def new\_random(self):

if (self.d == 0):

return self.begin

return rn.uniform(self.begin - self.d, self.begin + self.d)

class GenerateRequest:

def \_\_init\_\_(self, generator, count):

self.random\_generator = generator

self.num\_requests = count

self.receivers = []

self.next = 0

def generate\_request(self):

self.num\_requests -= 1

for receiver in self.receivers:

if receiver.receive\_request():

return receiver

return None

def delay(self):

return self.random\_generator.new\_random()

class ProcessRequest:

def \_\_init\_\_(self, generator, max\_queue\_size=-1):

self.random\_generator = generator

self.queue, self.received, = 0, 0

self.max\_queue, self.processed = max\_queue\_size, 0

self.next = 0

def receive\_request(self):

if self.max\_queue == -1 or self.max\_queue > self.queue:

self.queue += 1

self.received += 1

return True

return False

def process\_request(self):

if self.queue > 0:

self.queue -= 1

self.processed += 1

def delay(self):

return self.random\_generator.new\_random()

class Model:

def \_\_init\_\_(self, generator, operators, computers):

self.generator = generator

self.operators = operators

self.computers = computers

def event\_mode(self):

refusals = 0

generated\_requests = self.generator.num\_requests

generator = self.generator

generator.receivers = [self.operators[0], self.operators[1], self.operators[2]]

self.operators[0].receivers = [self.computers[0]]

self.operators[1].receivers = [self.computers[0]]

self.operators[2].receivers = [self.computers[1]]

generator.next = generator.delay()

self.operators[0].next = self.operators[0].delay()

blocks = [generator,

self.operators[0],

self.operators[1],

self.operators[2],

self.computers[0],

self.computers[1]]

while generator.num\_requests >= 0:

current\_time = generator.next

for block in blocks:

if 0 < block.next < current\_time:

current\_time = block.next

for block in blocks:

if current\_time == block.next:

if not isinstance(block, ProcessRequest):

next\_generator = generator.generate\_request()

if next\_generator is not None:

next\_generator.next = current\_time + next\_generator.delay()

else:

refusals += 1

generator.next = current\_time + generator.delay()

else:

block.process\_request()

if block.queue == 0:

block.next = 0

else:

block.next = current\_time + block.delay()

return {"refusal\_percentage": refusals / generated\_requests \* 100,

"refusals": refusals}

class Window(QtWidgets.QMainWindow):

def \_\_init\_\_(self):

QtWidgets.QWidget.\_\_init\_\_(self)

uic.loadUi("window.ui", self)

self.BtnModeling.clicked.connect(lambda: startModeling(self))

self.count.setText("0")

self.percent.setText("0")

self.resLabel.setText("Результат")

def startModeling(win):

win.resLabel.setText("Processing....")

print("Procssing ....")

#Get Value From Inteface

try:

time\_clients = win.time\_client.value()

delta\_time\_clients = win.d\_time\_client.value()

#print("Get Data Form Client OK")

first\_operator = win.time\_op\_1.value()

second\_operator = win.time\_op\_2.value()

third\_operator = win.time\_op\_3.value()

delta\_first\_operators = win.d\_time\_op\_1.value()

delta\_second\_operators = win.d\_time\_op\_2.value()

delta\_third\_operators = win.d\_time\_op\_3.value()

#print("Get Data From Operators OK")

first\_computer = win.time\_comp\_1.value()

second\_computer = win.time\_comp\_2.value()

#print("Get Data From Computers OK")

clients\_number = win.n.value()

repeat\_time = win.repeat.value()

except:

print("Error While Getting Data From Interface")

res = {"refusal\_percentage": 0, "refusals": 0}

#print("Start Processing")

#Processing

for i in range(repeat\_time):

#Generators

#print("Get Generator");

generator = GenerateRequest(RandomGenerator(time\_clients, delta\_time\_clients), clients\_number)

#Operators

#print("Get Operator")

operators = [ProcessRequest(RandomGenerator(first\_operator, delta\_first\_operators), max\_queue\_size=1),

ProcessRequest(RandomGenerator(second\_operator, delta\_second\_operators), max\_queue\_size=1),

ProcessRequest(RandomGenerator(third\_operator, delta\_third\_operators), max\_queue\_size=1)]

#Computers

#print("Get Computer")

computers = [ProcessRequest(RandomGenerator(first\_computer)),

ProcessRequest(RandomGenerator(second\_computer))]

#Start Modeling

#print("Start Modeling")

model = Model(generator, operators, computers)

#print("Get Result")

result = model.event\_mode()

#print(result)

res['refusals'] += result['refusals']

res['refusal\_percentage'] += result['refusal\_percentage']

res['refusal\_percentage'] /= repeat\_time

res['refusals'] = int(res['refusals'] / repeat\_time)

win.count.setText("{}".format(res['refusals']))

win.percent.setText("{:7.3f}".format(res['refusal\_percentage']))

win.resLabel.setText("Результат")

print("Result Update OK")

if \_\_name\_\_ == '\_\_main\_\_':

import sys

app = QtWidgets.QApplication(sys.argv)

w = Window()

w.show()

sys.exit(app.exec\_())