1. **Цель работы**

Смоделировать систему массового обслуживания для заданной предметной области средствами объектно-ориентированного языка программирования.

1. **Задание**
2. Реализовать приложение с графическим интерфейсом, моделирующее описанную в варианте систему массового обслуживания на объектно-ориентированном языке программирования (при написании программы самостоятельно можно не реализовывать только генератор равномерно распределённых псевдослучайных чисел).
3. Протестировать правильность работы разработанного приложения.
4. С помощью реализованной модели ответить на поставленные в варианте вопросы о функционировании системы массового обслуживания и сделать вывод о том, как работает смоделированная система.
5. **Вариант задания**

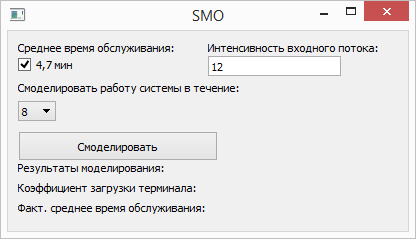
В справочной информационно-поисковой службе с одним терминалом имеет место пуассоновский входящий поток требований с интенсивностью 12 приходов в час. Обслуживание требований является экспоненциальным, но среднее время обслуживания зависит от числа требований, находящихся в очереди к терминалу. Эта зависимость представлена в таблице:

|  |  |
| --- | --- |
| Длина очереди, чел | Среднее время обслуживания, мин |
| 0 | 5.5 |
| 1 или 2 | 5 |
| 3, 4 или 5 | 4.5 |
| 6 | 4 |

Смоделировать работу системы в течение 1, 8, 40 часов. Найти фактическое среднее время обслуживания, коэффициент загрузки терминала. Повторить эти же действия при условии, что среднее время обслуживания всегда 4.7 мин. Сравнить результаты.

1. **Описание разработанного программного средства**

Программа разработана средствами Python + Qt5. Предназначена для моделирования систем массового обслуживания.



Интерфейс программного средства позволяет пользователю задавать режим работы (среднее время обслуживания – 4,7 мин либо в зависимости от очереди к терминалу), задавать интенсивность входного потока, а также выбрать время, в течение которого моделируется работа системы. Результатом моделирования будет коэффициент загрузки терминала и фактическое среднее время обслуживания.

1. **Текст программы**

import sys

from PyQt5.QtWidgets import QWidget, QLabel, QApplication, QCheckBox, QComboBox, QPushButton, QLineEdit

from PyQt5.QtCore import Qt

import queue as qq

import numpy as np

import scipy as sp

#время моделирования входного потока

def model\_time\_input(lambd, Tmax):

t = [0]

alpha = [np.random.uniform(0,1) for i in range(1,lambd \* Tmax)]

t.extend(- 1. / lambd \* np.log(alpha))

time\_input = list(filter(lambda x: x < Tmax, sp.cumsum(t)))

mean\_time\_input = (np.sum(t) / len(t)) \* 60.

return time\_input, mean\_time\_input

#время моделирования обслуживания

def model\_time\_service(q\_size, flag):

#

if flag == True:

t\_ser = 4.7 / 60. #среднее время обслуживания

else:

if q\_size == 0:

t\_ser = 5.5/ 60.

elif q\_size == 1 or q\_size == 2:

t\_ser = 5./ 60.

elif q\_size == 3 or q\_size == 4 or q\_size == 5:

t\_ser = 4.5/ 60.

elif q\_size == 6:

t\_ser = 4./ 60.

nu = 1. / t\_ser

alpha = np.random.uniform(0,1)

time\_service = - 1. / nu \* np.log(alpha)

return time\_service

def SMO(lambd, flag, Tmax):

q\_ser = False #обслуживается ли кто-то терминалом

arr\_time\_ser = []

qque = qq.Queue(maxsize = 6) #очередь

time\_input, mean\_time\_input = model\_time\_input(lambd, Tmax)

#обслуживание клиентов

tmp = 0.0

count\_ser = 0

count\_fail = 0

for el in time\_input:

time\_ser = model\_time\_service(qque.qsize(), flag)

arr\_time\_ser.append(time\_ser)

if q\_ser == False and qque.empty():

q\_ser =True

tmp = el + time\_ser

else:

if q\_ser == False and qque.qsize() !=0:

el1 = qque.get()

q\_ser =True

tmp = tmp + time\_ser

qque.put(el)

else:

if el >= tmp:

q\_ser = False

count\_ser += 1

if qque.qsize() !=0:

el1 = qque.get()

tmp = tmp + time\_ser

q\_ser = True

qque.put(el)

else:

q\_ser = True

tmp = el + time\_ser

else:

if q\_ser == True and qque.qsize() <= 5:

qque.put(el)

else:

count\_fail += 1

mean\_time\_ser = (np.sum(arr\_time\_ser)/len(arr\_time\_ser))\*60.

koef\_load = mean\_time\_input / mean\_time\_ser

return koef\_load, mean\_time\_ser

class Interface(QWidget):

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

super().\_\_init\_\_()

self.Tmax = 1 #время, в течение которого моделируется работа системы

self.flag = True

self.lambd = 12 #интенсивность входного потока в час

self.initUI()

def initUI(self):

self.lbl1 = QLabel("Среднее время обслуживания:", self)

cb = QCheckBox('4,7 мин', self)

cb.move(10, 25)

self.lbl1.move(10, 10)

cb.toggle()

cb.stateChanged.connect(self.changeTitle)

self.lbl11 = QLabel("Интенсивность входного потока:", self)

self.lbl11.move(200, 10)

self.le = QLineEdit("12",self)

self.le.move(200, 25)

self.le.textEdited[str].connect(self.onChanged)

self.lbl2 = QLabel("Смоделировать работу системы в течение:", self)

self.lbl2.move(10, 50)

combo = QComboBox(self)

combo.addItems(["1", "8", "40"])

combo.move(10, 70)

combo.activated[str].connect(self.onActivated)

btn = QPushButton('Смоделировать', self)

btn.move(10, 100)

btn.resize(200,30)

self.lbl6 = QLabel("Результаты моделирования:", self)

self.lbl6.move(10, 130)

self.lbl3 = QLabel("Коэффициент загрузки терминала:", self)

self.qle1 = QLabel(" ",self)

self.qle1.move(200, 150)

self.qle1.resize(200,15)

self.qle2 = QLabel(" ",self)

self.qle2.move(200, 170)

self.qle2.resize(200,15)

self.lbl5 = QLabel("Факт. среднее время обслуживания:", self)

self.lbl3.move(10, 150)

self.lbl5.move(10, 170)

# связывает событие нажатия на кнопку с методом

btn.clicked.connect(self.buttonClicked)

self.setGeometry(300, 300, 400, 200)

self.setWindowTitle('SMO')

self.show()

def onActivated(self, text):

self.Tmax = int(text)

def changeTitle(self, state):

if state == Qt.Checked:

self.flag = True

else:

self.flag = False

def onChanged(self, text):

if text == "":

self.lambd = 12

self.le.setText(str(self.lambd))

else:

self.lambd = int(text)

self.setFocus()

def buttonClicked(self):

koef\_load, mean\_time\_ser = SMO(self.lambd, self.flag, self.Tmax)

self.qle1.setText(str(koef\_load))

self.qle2.setText(str(mean\_time\_ser) + ' мин')

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

app = QApplication(sys.argv)

ex = Interface()

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

1. **Набор тестов**

