Министерство образования и науки Российской Федерации Московский государственный технический университет имени Н.Э.Баумана

Факультет «Робототехника и комплексная автоматизация» Кафедра «Системы автоматизированного проектирования»

**Отчет по практической работе №3**

По курсу «Теория вероятности и математическая статистика»

**Вариант 6**

Выполнил: студент группы: РК6-32Б Журавлев Н.В.

Проверил: Берчун Ю.В

Москва, 2020 г.

**Задача**

Требуется разработать программу, реализующую дискретно-событийное моделирование системы, рассмотренной в задании 2 домашнего задания №4. Обратите внимание, что все интервалы времени подчиняются законам распределений, носящим непрерывный характер. Поэтому категорически неверными является выбор целочисленных типов данных для моментов и интервалов времени, и тем более инкремент модельного времени с единичным шагом. Нужно реализовать именно переход от события к событию, как это сделано в GPSS и других проблемно-ориентированных системах. Для упрощения можно ограничиться использованием единственного потока случайных чисел для генерации всех необходимых случайных величин. Результатом работы программы должен быть лог-файл, содержащий записи типа: «В момент времени 12.345 транзакт с идентификатором 1 вошёл в модель», «В момент времени 123.456 транзакт с идентификатором 123 встал в очередь 1», «В момент времени 234.567 транзакт с идентификатором 234 занял устройство 2», «В момент времени 345.678 транзакт с идентификатором 345 освободил устройство 1», «В момент времени 456.789 транзакт с идентификатором 456 вышел из модели».

**Код программы**

from random import uniform

R1 = 6

B1 = 5

G1 = 8

N1 = 19

END\_TIME = 3600

def main():

def lenght\_queve(index\_task, number\_channel):

lenght = 0

i = index\_task

while i > 0:

if time\_system["channel"][i - 1] == number\_channel:

if time\_system["time\_generate"][index\_task] < time\_system["end\_work"][i - 1]:

lenght += 1

i -= 1

return lenght

def copy():

result = {"time\_generate": [],

"start\_work": [],

"end\_work": [],

"channel": [],

"index\_task": []}

for j in range(i):

result["time\_generate"].append(time\_system["time\_generate"][j])

result["start\_work"].append(time\_system["start\_work"][j])

result["end\_work"].append(time\_system["end\_work"][j])

result["channel"].append(time\_system["channel"][j])

result["index\_task"].append(time\_system["index\_task"][j])

return result

def expectation(index\_task, number\_channel):

temp\_index = index\_task - 1

while time\_system["channel"][temp\_index] != number\_channel and temp\_index > 0:

temp\_index -= 1

time\_wait = time\_system["end\_work"][temp\_index]

return time\_wait

def output():

select\_words = ["time\_generate", "start\_work", "end\_work"]

copy\_time = copy()

while True:

min\_time = [min(copy\_time["time\_generate"]), min(copy\_time["start\_work"]), min(copy\_time["end\_work"])]

min\_index = min\_time.index(min(min\_time))

min\_value = copy\_time[select\_words[min\_index]].index(min(copy\_time[select\_words[min\_index]]))

check\_time = copy\_time[select\_words[min\_index]][min\_value]

if check\_time > now\_time:

break

else:

if check\_time >= prelast\_time:

write\_result(min\_index, min\_value, check\_time)

copy\_time[select\_words[min\_index]][min\_value] = END\_TIME + 1

def write\_result(select, index, time):

result = "Время " + str(time) + " транзакт с идентификатором " + str(time\_system["index\_task"][index])

part\_result = result

if select == 0:

result += " вошел в модель " + '\n'

result += part\_result + " встал в очередь " + str(time\_system["channel"][index])

else:

if select == 1:

result += " занял устройство " + str(time\_system["channel"][index])

else:

result += " освободил устройство " + str(time\_system["channel"][index]) + '\n'

result += part\_result + " вышел из модели"

f.write(result + '\n')

f = open('log\_file.txt', 'w')

time\_system = {"time\_generate": [uniform(0, N1)],

"start\_work": [uniform(0, N1)],

"end\_work": [uniform(0, N1) + uniform(R1, N1)],

"channel": [1],

"index\_task": [1]

}

i = 1

now\_time = 0

prelast\_time = 0

while time\_system["time\_generate"][i - 1] < END\_TIME:

time\_system["time\_generate"].append(uniform(0, N1) + time\_system["time\_generate"][i - 1])

time\_system["index\_task"].append(time\_system["index\_task"][i - 1] + 1)

min\_queue = min([lenght\_queve(i, 1), lenght\_queve(i, 2)])

number\_channel = [lenght\_queve(i, 1), lenght\_queve(i, 2)].index(min([lenght\_queve(i, 1), lenght\_queve(i, 2)]))

time\_system["channel"].append(number\_channel + 1)

if number\_channel == 0:

time\_serve = uniform(R1, N1)

else:

time\_serve = uniform(G1, N1)

if min\_queue == 0:

time\_system["start\_work"].append(time\_system["time\_generate"][i])

else:

time\_system["start\_work"].append(expectation(i, number\_channel + 1))

time\_system["end\_work"].append(time\_system["start\_work"][i] + time\_serve)

if time\_system["end\_work"][0] <= time\_system["time\_generate"][i] <= END\_TIME:

now\_time = time\_system["time\_generate"][i]

output()

while time\_system["end\_work"][0] <= time\_system["time\_generate"][i] <= END\_TIME:

del time\_system["time\_generate"][0]

del time\_system["start\_work"][0]

del time\_system["end\_work"][0]

del time\_system["channel"][0]

del time\_system["index\_task"][0]

i -= 1

i += 1

prelast\_time = now\_time

now\_time = END\_TIME

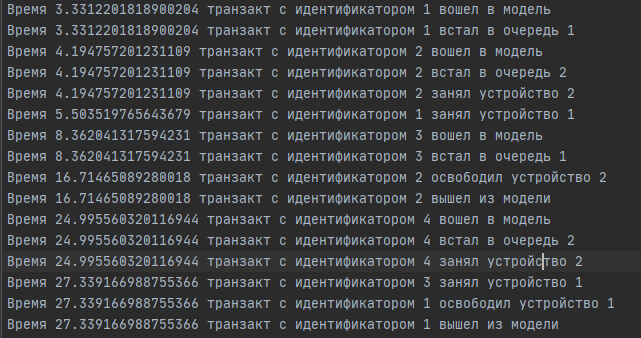
output()

f.close()

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

main()

# **Результаты:**



**Список литературы**

1.http://bigor.bmstu.ru/?cnt/?doc=VU/base.cou

2. Лекции и семинары по курсу теория вероятности и математическая статистика