|  |  |
| --- | --- |
| 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 *Робототехники и комплексная автоматизации*

КАФЕДРА *Системы автоматизированного проектирования (РК-6)*

**ОТЧЕТ О ВЫПОЛНЕНИИ УЧЕБНОЙ ПРАКТИКИ №3**

по дисциплине: «Учебно-технологическая практика на С++»

Студент: **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Новокшанов Е.А.**

*подпись, дата фамилия, и.о.*

Группа:  **РК6-36Б**

Преподаватель: **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Берчун Ю.В.**

*подпись, дата фамилия, и.о.*

*Москва, 2021 г*

# Оглавление

[Оглавление 2](#_Toc93061962)

[Постановка задачи 3](#_Toc93061963)

[Входные и выходные данные 4](#_Toc93061964)

[Алгоритмы 5](#_Toc93061965)

[Приложение 1. Содержание файлов исходного кода 6](#_Toc93061966)

# Постановка задачи

Требуется разработать программу, реализующую *дискретно-событийное моделирование* системы, рассмотренной в задании 2 домашнего задания №4:

* Смоделируем поведение покупателя в магазине, в котором работают 2 кассы, причём к каждой из них выстраивается отдельная очередь, а квалификация сотрудников немного отличается, поэтому время обслуживания распределено с разными параметрами. Каждая касса будет представлена одноканальным устройством, обращение к которым будем осуществлять по номерам. Очереди также будут идентифицироваться номерами, без введения символьных имён. Моделирование будем проводить в течение 1 часа, в качестве единицы времени будем выбирать секунду.
* Время между приходом покупателей распределено на отрезке [0; R1+G1+B1] ([0; 24]). Время обслуживания на первой кассе распределено на отрезке [R1; R1+G1+B1] ([10;24]). Время обслуживания на второй кассе распределено на отрезке [G1; R1+G1+B1] ([9;24]).
* При принятии решения покупатель сперва проверяет, есть ли свободная касса, и, если есть, направляется к ней.
* Если же обе кассы заняты, то выбирает кассу, очередь к которой в данный момент короче.
* Если же свободны обе кассы, или очередь к ним одинакова, то выбирается первая касса.

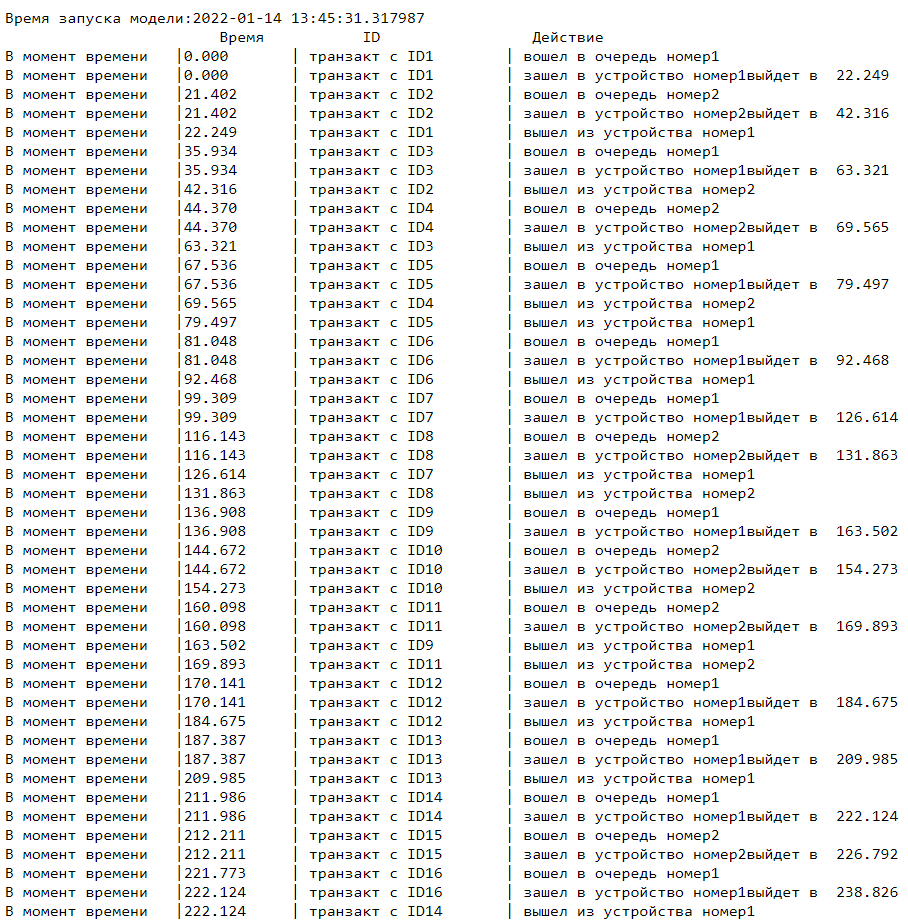
Обратите внимание, что все интервалы времени подчиняются законам распределений, носящим непрерывный характер. Поэтому категорически неверными является выбор целочисленных типов данных для моментов и интервалов времени, и тем более инкремент модельного времени с единичным шагом. Нужно реализовать именно 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».

# Входные и выходные данные

По итогам выполнение программы формируется файл, название которого может быть задано первым аргументом командной строки или изменено в коде программы, содержащий в себе информацию о поведение транзактов в моделе.

Выходные данные:



*Рисунок 1. Лог-файл*

# Алгоритмы

В программе предусмотрены 2 цепи: цепь текущих событий (CEC) и цепь будущих событий (FEC), которые будут заполняться и изменяться в будущем. Изначально в цепи будущих событий будут находиться транзакты, имеющие только время генерации. Программа последовательно выполняет 3 этапа (или фазы):

* 1ый этап – фаза ввода: здесь транзакты добавляются в FEC и ожидают своей очереди
* 2ой этап – фаза распределения: здесь транзакты с минимальным временем перемещаются из FEC в CEC и указывается соответствующая очередь.
* 3ый этап – фаза просмотра: здесь транзакты либо меняют свою позицию в модели (перемещается по очереди, переходит на обслуживающее устройство), либо выходит из модели.

# Приложение 1. Содержание файлов исходного кода

**Pract3.py**

from Transact import \*  
#from prt import \*  
  
from Queue\_h import \*  
from Device import \*  
from copy import \*  
from datetime import datetime  
def funcSort(x):  
 #M = [x.get\_t\_e]  
 return int(1000\*max(x.get\_T\_e(), x.get\_T\_g()))  
def main():  
 print("Ввещдите время моделирования:")  
  
 osCommandString = "notepad.exe file.txt"  
 current\_datetime = datetime.now()  
 f.write("Время запуска модели:" + str(current\_datetime) + "\n")  
 f.write("\t\t\tВремя" + "\t\tID" + "\t\t Действие\n")  
 #log = mylogger("res.txt", True)  
 #print = log.printml()  
 global\_time = 0.0  
 t\_end = int(input())  
 t = 0.0  
 i = 1  
 FEC = []  
 while(t < t\_end):  
  
 FEC.append(Transact(i, t, -1.0, 0))  
 i+=1  
 t += Rand(0, 28)  
  
 for j in range(0,len(FEC)):  
 (FEC[j].get\_T\_g())  
  
 j = 0  
 D1 = Device(1, 10, 28)  
 D2 = Device(2, 9, 28)  
 Q1 = Queue\_to\_device(1, D1)  
 Q2 = Queue\_to\_device(2, D2)  
 while(j < len(FEC)):  
  
 if(FEC[j].get\_stat() == 0):  
 global\_time = FEC[j].get\_T\_g()  
 if (global\_time > t\_end):  
 f.write("End model")  
 return 0  
 if(Q1.get\_lenght()+D1.get\_stat() <= Q2.get\_lenght() + D2.get\_stat()):  
 FEC[j].set\_stat(1)  
 FEC[j].set\_Q\_number(1)  
 out = "В момент времени |" + str("%.3lf"%global\_time) + "\t| транзакт с ID" + str(FEC[j].get\_id()) + "\t| вошел в очередь номер" + "1" + "\n"  
 f.write(out)  
 Q1.add\_transact\_to\_Q(FEC[j])  
 #print(Q1.get\_lenght())  
 if(D1.get\_stat() == 0 and Q1.get\_lenght() == 1):  
 D1.set\_T\_g(global\_time)  
 Q1.go\_to\_device()  
  
 #D1.add\_transact\_to\_D(FEC[j])  
 Tr\_in\_D = copy(D1.go\_out\_of\_D())  
 #k = 0  
 # while(Tr\_in\_D.get\_T\_e() > FEC[j].get\_T\_g() or (Tr\_in\_D.get\_T\_e() > FEC[j].get\_T\_e() and FEC[j].get\_stat()==2)):  
 # k+=1  
 FEC.append(Tr\_in\_D)  
 FEC = sorted(FEC, key = lambda trans: funcSort(trans))  
 #print(1)  
 #FEC.insert(k-1, Tr\_in\_D)  
  
 else:  
 FEC[j].set\_stat(1)  
 FEC[j].set\_Q\_number(2)  
 out = "В момент времени |" + str("%.3lf"%global\_time) + "\t| транзакт с ID" + str(FEC[j].get\_id()) + "\t| вошел в очередь номер" + '2' + "\n"  
 f.write(out)  
 Q2.add\_transact\_to\_Q(FEC[j])  
 #print(Q2.get\_lenght())  
 if (D2.get\_stat() == 0 and Q2.get\_lenght() == 1):  
 D2.set\_T\_g(global\_time)  
 Q2.go\_to\_device()  
  
 #D2.add\_transact\_to\_D(FEC[j])  
 Tr\_in\_D = copy(D2.go\_out\_of\_D())  
 FEC.append(Tr\_in\_D)  
 FEC = sorted(FEC, key=lambda trans: funcSort(trans))  
 elif(FEC[j].get\_stat() == 1):  
 global\_time = FEC[j].get\_T\_g()  
 if (global\_time > t\_end):  
 f.write("End model")  
 return 0  
 if(FEC[j].set\_Q\_number()==1):  
 if (D1.get\_stat() == 0 and Q1.get\_lenght() == 1):  
 D1.set\_T\_g(global\_time)  
 Q1.go\_to\_device()  
  
 #D1.add\_transact\_to\_D(FEC[j])  
 Tr\_in\_D = copy(D1.go\_out\_of\_D())  
 FEC.append(Tr\_in\_D)  
 FEC = sorted(FEC, key=lambda trans: funcSort(trans))  
 else:  
 if (D2.get\_stat() == 0 and Q2.get\_lenght() == 1):  
 D2.set\_T\_g(global\_time)  
 Q2.go\_to\_device()  
  
 #D2.add\_transact\_to\_D(FEC[j])  
 Tr\_in\_D = copy(D2.go\_out\_of\_D())  
 FEC.append(Tr\_in\_D)  
 FEC = sorted(FEC, key=lambda trans: funcSort(trans))  
 elif(FEC[j].get\_stat() == 2):  
 global\_time = FEC[j].get\_T\_e()  
 if (global\_time > t\_end):  
 f.write("End model")  
 return 0  
 if(FEC[j].get\_Q\_num() == 1):  
 D1.set\_stat(0)  
 D1.set\_T\_g(global\_time)  
 if(Q1.get\_lenght()>0):  
 D1.set\_T\_g(global\_time)  
 Q1.go\_to\_device()  
  
 # D2.add\_transact\_to\_D(FEC[j])  
 Tr\_in\_D = D1.go\_out\_of\_D()  
 FEC.append(Tr\_in\_D)  
 FEC = sorted(FEC, key=lambda trans: funcSort(trans))  
 elif(FEC[j].get\_Q\_num() == 2):  
 D2.set\_stat(0)  
 D2.set\_T\_g(global\_time)  
 if (Q2.get\_lenght() > 0):  
 D2.set\_T\_g(global\_time)  
 Q2.go\_to\_device()  
  
 # D2.add\_transact\_to\_D(FEC[j])  
 Tr\_in\_D = D2.go\_out\_of\_D()  
 FEC.append(Tr\_in\_D)  
 FEC = sorted(FEC, key=lambda trans: funcSort(trans))  
 FEC[j].end()  
 j += 1  
 f.close()  
 #os.startfile("Результат\_моделирования.txt")  
 #os.system(osCommandString)  
 #subprocess.call(['notepad.exe', "Результат\_моделирования.txt"])  
 return 0  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

**Queue\_h.**

from Device import \*  
from copy import \*  
class Queue\_to\_device:  
 def \_\_init\_\_(self, i, D):  
 self.Q = []  
 self.Id = i  
 self.D = D  
 def add\_transact\_to\_Q(self, new\_Tr):  
 self.Q.append(new\_Tr)  
 def go\_to\_device(self):  
 self.D.add\_transact\_to\_D(self.Q[0])  
 self.Q.pop(0)  
 def get\_lenght(self):  
 return len(self.Q)  
 def get\_ID(self):  
 return self.Id

**Device.py**

from Transact import \*  
from copy import \*  
class Device:  
 def \_\_init\_\_(self, id, t1, t2):  
 self.Tr = Transact(0, 0, 0, 0)  
 self.Id = id  
 self.t1 = t1  
 self.t2 = t2  
 self.T\_g = 0.0  
 self.stat = 0  
 def add\_transact\_to\_D(self, transact):  
  
 self.Tr = copy(transact)  
 self.stat = 1  
 self.Tr.set\_stat(2)  
 out = "В момент времени |" + str("%.3lf"%self.T\_g) + "\t| транзакт с ID" + str(self.Tr.get\_id()) + "\t| зашел в устройство номер" + str(self.Id)  
 f.write(out)  
 def set\_T\_g(self, t):  
 self.T\_g = t  
 def go\_out\_of\_D(self):  
 #self.stat = 0  
 self.Tr.set\_T\_e(self.T\_g + Rand(self.t1, self.t2))  
 out = "выйдет в " + str("%.3lf"%self.Tr.get\_T\_e()) + "\n"  
 f.write(out)  
  
 #self.Tr.set\_stat()  
 return self.Tr  
 def get\_stat(self):  
 return self.stat  
 def set\_stat(self, s):  
 self.stat = s

**Transact.py**

from random import \*  
from copy import \*  
def Rand(t1, t2):  
 return uniform(t1, t2)  
f = open('Результат\_моделирования.txt', 'w')  
class Transact:  
 def \_\_init\_\_(self, id, T\_g, T\_e, stat):  
 self.id = id  
 self.T\_g = T\_g  
 self.T\_e = T\_e  
 self.stat = stat  
 self.Q\_number = 0  
 def set\_Q\_number(self, n):  
 self.Q\_number = n  
 def set\_id(self, i):  
 self.id = i  
 def set\_T\_g(self, t):  
 self.T\_g = t  
 def set\_T\_e(self, t):  
 self.T\_e = t  
 def set\_stat(self, s):  
 self.stat = s  
 def get\_id(self):  
 return self.id  
 def get\_T\_g(self):  
 return self.T\_g  
 def get\_T\_e(self):  
 return self.T\_e  
 def get\_stat(self):  
 return self.stat  
 def get\_Q\_num(self):  
 return self.Q\_number  
 def end(self):  
 out = "В момент времени |" + str("%.3lf"%self.T\_e) + "\t| транзакт с ID" + str(self.id) + "\t| вышел из устройства номер" + str(self.Q\_number) + "\n"  
 f.write(out)

Полный листинг программы можно увидеть по ссылке: <https://github.com/NovokshanovE/NewRepo/tree/master>