Министерство образования и науки 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 комплексной автоматизации

КАФЕДРА Системы автоматизированного проектирования (РК-6)

ОТЧЕТ О ВЫПОЛНЕНИИ ДОМАШНЕГО ЗАДАНИЯ ПО КУРСУ«АНАЛИТИЧЕСКИЕ МОДЕЛИ И ИМИТАЦИОННОЕ МОДЕЛИРОВАНИЕНА СИСТЕМНОМ УРОВНЕ»

Студент Никатов Владислав Алексеевич

Группа РК6-84б

Тип задания Домашнее задание

Вариант 7

Студент _____ **Никатов В.А.**
подпись, дата *фамилия, и.о.*

Преподаватель _____ **Берчун Ю.В.**
подпись, дата *фамилия, и.о.*

Оценка _____

Москва, 2020 г.

Оглавление

1 Задание

Для многоканальной системы с неограниченной очередью, описанной в предыдущей лабораторной работе, построить таблицу отражающую состояния цепи текущих событий (ЦТС) и цепи будущих событий (ЦБС) по модельному времени. Решение задачи должно быть представлено в виде программного кода с использованием любого языка программирования, по усмотрению студента. В программе требуется реализовать алгоритм дискретно-событийного моделирования.

2 Решение

При решении задачи был реализован объект типа Transaction, хранящий в себе следующие значения:

- номер транзакта;
- время движения;
- номер текущего блока;
- уровень приоритета;
- номер следующего блока.

Под блоками понимаются объекты, которые в определенный момент времени содержат в себе транзакты, а так же выполняют над ними какие-либо действия. К объектам данного типа относятся:

1. **Generate.** При прохождении транзактом данного объекта, при входе транзакт обновляет значения текущего и следующего блоков на блок Generate и следующий за ним блок соответственно, а так же время движения на КМР (Как можно раньше). При выходе транзакта из блока создается новый транзакт в ЦБС, который попадет в блок через заданное функцией-генератором количество времени. Используется для генерации заявок на обслуживание, а так же транзакта, завершающего моделирование. В первом случае функция-генератор возвращает число, заданное экспоненциальным распределением со средним значением 39, во втором – время моделирования.

2. **Queue.** Объект может вмещать в себя ограниченное или неограниченное число объектов. При входе транзакта в очередь значение счетчиков объектов внутри увеличивается, при выходе – уменьшается.

3. **Storage.** Объект, исполняющий задачу многоканального обслуживания. Может иметь ограничение на число каналов. Транзакт может войти в объект только если счетчик внутри объекта меньше максимального заданного размера. При входе транзакта в объект значение счетчика увеличивается на 1, а сам транзакт перемещается в ЦБС, будущим событием является выход из объекта Storage, а время, через которое это произойдет, задается генератором, возвращающим числа, заданные экспоненциальным распределением со средним значением 229. Через указанный промежуток времени, транзакт вернется в ЦТС и будет готов выйти из объекта Storage. При выходе, счетчик объектов уменьшается.

4. **Terminate.** При входе транзакта в объект, значение счетчика модели уменьшается на заданное заранее число, а сам транзакт удаляется из модели.

Основным объектом является Model, в котором используются экземпляры перечисленных выше классов, хранятся ЦТС, ЦБС и ТМБ, а так же реализованы фазы ввода, коррекции таймера и просмотра. Используемые для моделирования объекты представлены и пронумерованы в порядке их вызова в таблице 2.1.

№	Объект
0	Generate
1	Queue
2	Storage
3	Terminate
4	Generate
5	Terminate

Состояния ЦТС и ЦБС сразу после стадии ввода, коррекции таймера и просмотра, а так же значение таймера модельного времени (ТМВ) отражены в таблице 2.2.2.

Таблица 2.2.2 а. Состояния ТМВ, ЦТС и ЦБС после каждой стадии.

ТМВ	Стадия	ЦТС	ЦБС
0.0	до ввода	Пусто	Пусто
0.0	посла ввода	Пусто	[1,1,нет,0,0], [2,240,нет,0,4]
1.0	после кор. тайм.	[1,КМР,нет,0,0]	[2,240,нет,0,4]
1.0	после просмотра	Пусто	[3,25.1387,нет,0,0], [1,190.2625,2,0,3], [2,240,нет,0,4]
25.1387	после кор. тайм.	[3,КМР,нет,0,0]	[1,190.2625,2,0,3], [2,240,нет,0,4]
25.1387	после просмотра	Пусто	[4,103.5584,нет,0,0], [3,154.9645,2,0,3], [1,190.2625,2,0,3], [2,240,нет,0,4]
103.5584	после кор. тайм.	[4,КМР,нет,0,0]	[3,154.9645,2,0,3], [1,190.2625,2,0,3], [2,240,нет,0,4]
103.5584	после просмотра	[4,КМР,1,0,2]	[5,133.3050,нет,0,0], [3,154.9645,2,0,3], [1,190.2625,2,0,3], [2,240,нет,0,4]
133.3050	после кор. тайм.	[4,КМР,1,0,2], [5,КМР,нет,0,0]	[3,154.9645,2,0,3], [1,190.2625,2,0,3], [2,240,нет,0,4]
133.3050	после просмотра	[4,КМР,1,0,2], [5,КМР,1,0,2]	[3,154.9645,2,0,3], [6,181.6222,нет,0,0], [1,190.2625,2,0,3], [2,240,нет,0,4]
154.9645	после кор. тайм.	[4,КМР,1,0,2], [5,КМР,1,0,2], [3,КМР,2,0,3]	[6,181.6222,нет,0,0], [1,190.2625,2,0,3], [2,240,нет,0,4]
154.9645	после просмотра	[5,КМР,1,0,2]	[6,181.6222,нет,0,0], [1,190.2625,2,0,3], [2,240,нет,0,4], [4,768.6171,2,0,3]
181.6222	после кор. тайм.	[5,КМР,1,0,2], [6,КМР,нет,0,0]	[1,190.2625,2,0,3], [2,240,нет,0,4], [4,768.6171,2,0,3]
181.6222	после просмотра	[5,КМР,1,0,2], [6,КМР,1,0,2]	[1,190.2625,2,0,3], [7,201.0180,нет,0,0], [2,240,нет,0,4], [4,768.6171,2,0,3]

Таблица 2.2.2 б. Состояния ТМВ, ЦТС и ЦБС после каждой стадии.

ТМВ	Стадия	ЦТС	ЦБС
190.2625	после кор. тайм.	[5,КМР,1,0,2], [6,КМР,1,0,2], [1,КМР,2,0,3]	[7,201.0180,нет,0,0], [2,240,нет,0,4], [4,768.6171,2,0,3]
190.2625	после просмотра	[6,КМР,1,0,2]	[7,201.0180,нет,0,0], [2,240,нет,0,4], [5,319.7421,2,0,3], [4,768.6171,2,0,3]
201.0180	после кор. тайм.	[6,КМР,1,0,2], [7,КМР,нет,0,0]	[2,240,нет,0,4], [5,319.7421,2,0,3], [4,768.6171,2,0,3]
201.0180	после просмотра	[6,КМР,1,0,2], [7,КМР,1,0,2]	[8,217.2184,нет,0,0], [2,240,нет,0,4], [5,319.7421,2,0,3], [4,768.6171,2,0,3]
217.2184	после кор. тайм.	[6,КМР,1,0,2], [7,КМР,1,0,2], [8,КМР,нет,0,0]	[2,240,нет,0,4], [5,319.7421,2,0,3], [4,768.6171,2,0,3]
217.2184	после просмотра	[6,КМР,1,0,2], [7,КМР,1,0,2], [8,КМР,1,0,2]	[9,226.2480,нет,0,0], [2,240,нет,0,4], [5,319.7421,2,0,3], [4,768.6171,2,0,3]
226.2480	после кор. тайм.	[6,КМР,1,0,2], [7,КМР,1,0,2], [8,КМР,1,0,2], [9,КМР,нет,0,0]	[2,240,нет,0,4], [5,319.7421,2,0,3], [4,768.6171,2,0,3]
226.2480	после просмотра	[6,КМР,1,0,2], [7,КМР,1,0,2], [8,КМР,1,0,2], [9,КМР,1,0,2]	[10,238.9538,нет,0,0], [2,240,нет,0,4], [5,319.7421,2,0,3], [4,768.6171,2,0,3]
238.9538	после кор. тайм.	[6,КМР,1,0,2], [7,КМР,1,0,2], [8,КМР,1,0,2], [9,КМР,1,0,2], [10,КМР,нет,0,0]	[2,240,нет,0,4], [5,319.7421,2,0,3], [4,768.6171,2,0,3]
238.9538	после просмотра	[6,КМР,1,0,2], [7,КМР,1,0,2], [8,КМР,1,0,2], [9,КМР,1,0,2], [10,КМР,1,0,2]	[2,240,нет,0,4], [11,249.9901,нет,0,0], [5,319.7421,2,0,3], [4,768.6171,2,0,3]
240.0	после кор. тайм.	[6,КМР,1,0,2], [7,КМР,1,0,2], [8,КМР,1,0,2], [9,КМР,1,0,2], [10,КМР,1,0,2], [2,КМР,нет,0,4]	[11,249.9901,нет,0,0], [5,319.7421,2,0,3], [4,768.6171,2,0,3]
240.0	после просмотра	[6,КМР,1,0,2], [7,КМР,1,0,2], [8,КМР,1,0,2], [9,КМР,1,0,2], [10,КМР,1,0,2]	[11,249.9901,нет,0,0], [5,319.7421,2,0,3], [12,480,нет,0,4], [4,768.6171,2,0,3]

Приложение 1

main.py

```
from Model import Model


if __name__ == '__main__':
    model = Model()
    model.init()
    model.run()
```

Model.py

```
from Generate import Generate
from Terminate import Terminate
from Queue import Queue
from Storage import Storage
import random


def time_generator(avg, first_one=False):
    if first_one:
        yield 1
    while True:
        yield random.expovariate(1./avg)


def const_generator(number):
    while True:
        yield number


class Model:
    def __init__(self):
        self._cec = []  # Current Events Chain (цепь текущих событий)
        self._fec = []  # Future  Events Chain (цепь будущих событий)
        self._mtc = [0]  # Model Time Counter (счетчик времени моделирования)
        self._count = [1]
        self._blocks = []

    def init(self):
        self._blocks.append(Generate(fec=self._fec,
                                     mtc=self._mtc,
```

```
                        time_generator=time_generator(39, first_one=True)))
        self._blocks.append(Queue())
        self._blocks.append(Storage(cec=self._cec,
                        fec=self._fec,
                        mtc=self._mtc,
                        time_generator=time_generator(229),
                        max_len=2))
        self._blocks.append(Terminate(cec=self._cec,
                         counter=self._count))
        self._blocks.append(Generate(fec=self._fec,
                         mtc=self._mtc,
                         time_generator=const_generator(240)))
        self._blocks.append(Terminate(cec=self._cec,
                         counter=self._count,
                         decrement=1))

    def run(self):
        self.print_info('До стадии ввода')
        self._insert()
        self.print_info('После стадии ввода')
        while self._count[0] != 0:
            self._timer_correction()
            self.print_info('После стадии коррекции таймера')
            self._view()
            self.print_info('После стадии просмотра')

    def _insert(self):
        for i, block in enumerate(self._blocks):
            if isinstance(block, Generate):
                block.generate(cur_block=i)

    def _timer_correction(self):
        if len(self._fec) == 0:
            exit('Произошла ошибка. Цепь будущих событий опустела раньше
окончания программы.')
        t = self._fec[0]
        self._mtc[0] = t.transition_time
        self._fec.remove(t)
        t.transition_time = -1
        self._cec.append(t)
        while len(self._fec) != 0 and self._fec[0].transition_time == self._mtc[0]:
            t = self._fec[0]
            self._fec.remove(t)
            t.transition_time = -1
            self._cec.append(t)
```

```
    def _view(self):
        f_change = True
        while f_change:
            f_change = False
            for t in self._cec:
                while self._blocks[t.current_block].can_exit() and
self._blocks[t.next_block].can_enter():
                    f_change = True
                    # print('Переход транзакта', t.num, 'из блока', t.current_block, 'в блок',
t.next_block)
                    if t.current_block != -1:
                        self._blocks[t.current_block].exit()
                    self._blocks[t.next_block].enter(t)
                    if len(self._cec) == 0:
                        return
                    t = self._cec[0]

    def print_info(self, stage: str):
        print(stage + ':')
        print('ТМВ:', self._mtc[0])
        print('ЦТС:', self._cec)
        print('ЦБС:', self._fec)
        print()
```

Generate.py

```
from Transaction import Transaction
from operator import attrgetter


class Generate:
    _counter = 1

    def __init__(self,
                 fec: list,
                 mtc: list,
                 time_generator,
                 priority: int = 0):
        self._fec = fec
        self._mtc = mtc
        self._priority = priority
        self._time_generator = time_generator
        self.cur_transaction = None
```

```
  def generate(self, cur_block):
    self._fec.append(Transaction(num=Generate._counter,
                                 transition_time=self._mtc[0] + next(self._time_generator),
                                 current_block=-1,
                                 priority=self._priority,
                                 next_block=cur_block))
    Generate._counter += 1
    self._fec.sort(key=attrgetter('transition_time'))

  def can_enter(self):
    return True

  def can_exit(self):
    return True

  def enter(self, t):
    if self.cur_transaction is not None:
      print('Потеря транзакта', t)
      return
    t.current_block = t.next_block
    t.next_block = t.next_block + 1
    t.transition_time = -1
    self.cur_transaction = t

  def exit(self):
    t = self.cur_transaction
    self.generate(t.current_block)
    self.cur_transaction = None
    return t
```

Storage.py

```
from operator import attrgetter


class Storage:
  def __init__(self,
               cec: list,
               fec: list,
               mtc: list,
               time_generator,
               max_len: int = None):
```

```
        self._cec = cec
        self._fec = fec
        self._mtc = mtc
        self._time_generator = time_generator
        self._m_len = max_len
        self._s = 0

    def can_enter(self):
        if self._m_len is None:
            return True
        return self._s < self._m_len

    def can_exit(self):
        return True

    def enter(self, t):
        self._s += 1
        t.current_block = t.next_block
        t.next_block = t.next_block + 1
        t.transition_time = self._mtc[0] + next(self._time_generator)
        self._fec.append(t)
        self._cec.remove(t)
        self._fec.sort(key=attrgetter('transition_time'))

    def exit(self):
        self._s -= 1
```

Queue.py

```
from collections import deque


class Queue:
    def __init__(self, max_len: int = None):
        self._m_len = max_len
        self._q = deque()

    def can_enter(self):
        if self._m_len is None:
            return True
        return len(self._q) < self._m_len

    def can_exit(self):
```

```
        return True

    def enter(self, t):
        t.current_block = t.next_block
        t.next_block = t.next_block + 1
        t.transition_time = -1
        self._q.appendleft(t)

    def exit(self):
        return self._q.pop()
```

Terminate.py

```
class Terminate:
    def __init__(self,
                 counter: list,
                 cec: list,
                 decrement: int = 0):
        self._cec = cec
        self._counter = counter
        self._decrement = decrement

    def can_enter(self):
        return True

    def can_exit(self):
        return True

    def enter(self, t):
        self._cec.remove(t)
        self._counter[0] -= self._decrement

    def exit(self, t):
        pass
```

Transaction.py

```
class Transaction:
    def __init__(self,
```

```
            num: int,
            transition_time: float,
            current_block: int,
            priority: int,
            next_block: int):
        self.num = num
        self.transition_time = transition_time
        self.current_block = current_block
        self.priority = priority
        self.next_block = next_block

    def __str__(self):
        string = '[' + str(self.num) + ','
        if self.transition_time == -1:
            string += 'КМР'
        else:
            string += str(self.transition_time)
        string += ','
        if self.current_block == -1:
            string += 'нет'
        else:
            string += str(self.current_block)

        string += ',' + str(self.priority) + ',' + str(self.next_block) + ']'
        return string

    def __repr__(self):
        return str(self)
```