

Задание

Вариант 10

Имеется пешеходный переход, оснащённый кнопкой.

После нажатия на кнопку, через 10 с движение автомобилей запрещается и возможен переход в течение 10 с.

Пешеходы, прибывшие менее, чем за 3 с до окончания сигнала, разрешающего переход, остаются на месте. Считается, что они нажмут на кнопку в момент возобновления автомобильного движения.

После возобновления движения автомобилей в течение первых 10 с средняя интенсивность проезда составляет 10 машин в минуту (все распределения в данной задаче носят экспоненциальный характер), затем она увеличивается до 50.

Интенсивность движения автомобилей – 25 машин в минуту.

Интенсивность прихода пешеходов – 2 человека в минуту.

1. Промоделировать работу перехода в течение часа.
2. Проанализировать пропускную способность движения автомобилей в районе перехода, оценить:
 - величину очередей ожидающих проезда автомобилей
 - число ожидающих пешеходов и время их ожидания.
3. Исследовать модификацию системы, в которой остановка движения автомобилей допускается не чаще, чем раз в 40 с.

Отчет по GPSS

```
; GPSS/PC Traffic Light Simulation

GENERATE ,,,1
ADVANCE 3600
TERMINATE 1

;GREENTIME EQU      10
;REDTIME EQU        10

PASS_TIME EQU 7
GAP_TIME EQU 3
BTN_PRESSSS_DELAY EQU 10

CAR_INTENSE equ 60/25
MAN_INTENSE equ 60/2

wait_time VARIABLE C1 - X$car_start_time
CarsPassSpeed FUNCTION wait_time,D2
10,10/11,50

GENERATE 0,0,0,1
LOGIC R man_sem
LOGIC R btn
MARK_HANDLER LOGIC S car_sem
SAVEVALUE car_start_time,C1

GATE LS btn
ADVANCE BTN_PRESSSS_DELAY
LOGIC R car_sem
LOGIC S man_sem
LOGIC R btn
ADVANCE PASS_TIME
LOGIC R man_sem
ADVANCE GAP_TIME
TRANSFER ,MARK_HANDLER

GENERATE          MAN_INTENSE;Create people queue
QUEUE q_people

    GATE LS car_sem,CARS_NOT_MOVE
GATE LR btn,CARS_NOT_MOVE
LOGIC S btn ; # PUSH_THE_BUTTON

CARS_NOT_MOVE GATE LS man_sem
PASS_THE_ROAD DEPART q_people
TERMINATE
```

```

GENERATE      CAR_INTENSE      ;Create next automobile.
QUEUE q_cars
SEIZE cross
ADVANCE (Exponential(2,0,60/FN%CarsPassSpeed))
GATE LS car_sem
DEPART q_cars
RELEASE cross

TERMINATE; INCOMING CARS

START 1

```

```

06/02/19 01:39:17 Model Translation Begun.
06/02/19 01:39:17 Ready.
06/02/19 01:39:17 Simulation in Progress.
06/02/19 01:39:17 The Simulation has ended. Clock is 3600.000000.
06/02/19 01:39:17 Reporting in ModelText.1.1 - REPORT Window.

```

| Начальное время | Конечное время | Кол-во блоков | Кол-во устройств | Кол-во мн.канал. устройств |
|-----------------|----------------|---------------|------------------|-------------------------------|
| 0 | 3600 | 34 | 0 | 0 |

Рисунок 1: Общая информация

| Имя | Значение |
|------------------|-----------|
| BTN | 10008.000 |
| BTN_PRESSS_DELAY | 10.000 |
| CARSPASSSPEED | 10006.000 |
| CARS_NOT_MOVE | 23.000 |
| CAR_INTENSE | 2.400 |
| CAR_SEM | 10009.000 |
| CAR_START_TIME | 10010.000 |
| GAP_TIME | 3.000 |
| MAN_INTENSE | 30.000 |
| MAN_SEM | 10007.000 |
| MARK_HANDLER | 7.000 |
| MARK_TO | 34.000 |
| PASS_THE_ROAD | 24.000 |
| PASS_TIME | 7.000 |
| Q_CARS | 10011.000 |
| Q_PEOPLE | 10012.000 |
| WAIT_TIME | 10005.000 |

Рисунок 2: Имена

| Метка | Позиция блока | Тип блока | Кол-во тран. вошедших в блок | Кол-во тран. в блоке в конце моделирования | Кол-во тран., ожидающих выполнения спец. условия |
|---------------|---------------|-----------|---------------------------------|-----------------------------------------------|--------------------------------------------------------|
| | 1 | GENERATE | 1 | 0 | 0 |
| | 2 | ADVANCE | 1 | 0 | 0 |
| | 3 | TERMINATE | 1 | 0 | 0 |
| | 4 | GENERATE | 1 | 0 | 0 |
| | 5 | LOGIC | 1 | 0 | 0 |
| | 6 | LOGIC | 1 | 0 | 0 |
| MARK_HANDLER | 7 | LOGIC | 120 | 0 | 0 |
| | 8 | SAVEVALUE | 120 | 1 | 0 |
| | 9 | GATE | 119 | 0 | 0 |
| | 10 | ADVANCE | 119 | 0 | 0 |
| | 11 | LOGIC | 119 | 0 | 0 |
| | 12 | LOGIC | 119 | 0 | 0 |
| | 13 | LOGIC | 119 | 0 | 0 |
| | 14 | ADVANCE | 119 | 0 | 0 |
| | 15 | LOGIC | 119 | 0 | 0 |
| | 16 | ADVANCE | 119 | 0 | 0 |
| | 17 | TRANSFER | 119 | 0 | 0 |
| | 18 | GENERATE | 120 | 0 | 0 |
| | 19 | QUEUE | 120 | 0 | 0 |
| | 20 | GATE | 120 | 0 | 0 |
| | 21 | GATE | 120 | 0 | 0 |
| | 22 | LOGIC | 120 | 1 | 0 |
| CARS_NOT_MOVE | 23 | GATE | 119 | 0 | 0 |
| PASS_THE_ROAD | 24 | DEPART | 119 | 0 | 0 |
| | 25 | TERMINATE | 119 | 0 | 0 |
| | 26 | GENERATE | 1500 | 0 | 0 |
| | 27 | QUEUE | 1500 | 0 | 0 |
| | 28 | TERMINATE | 1500 | 0 | 0 |
| | 29 | GENERATE | 3000 | 0 | 0 |
| | 30 | TEST | 3000 | 0 | 0 |
| | 31 | GATE | 2390 | 0 | 0 |
| | 32 | DEPART | 1499 | 0 | 0 |
| | 33 | TERMINATE | 1499 | 0 | 0 |
| MARK_TO | 34 | TERMINATE | 1501 | 0 | 0 |

Рисунок 3: Блоки

| Имя / номер | Макс. содержимое очереди за период моделирования | Текущее содержимое очереди | Общее кол-во входов тран. в очередь | Общее кол-во входов тран. в очередь с нулевым временем ожидания | Ср. значение содержимого очереди | Ср. время пребывания одного транзакта в очереди | Ср. время пребывания одного транзакта в очереди без учета 'нулевых' входов | Кол-во тран., ожидающих выполнения спец. условия |
|-------------|--------------------------------------------------------|----------------------------------|-------------------------------------------|-----------------------------------------------------------------------------|----------------------------------------|-------------------------------------------------------|----------------------------------------------------------------------------------------|-----------------------------------------------------------|
| Q_CARS | 6 | 1 | 1500 | 1499 | 1,541 | 3,698 | 5546,4 | 0 |
| Q_PEOPLE | 1 | 1 | 120 | 0 | 0,331 | 9,917 | 9,917 | 0 |

Рисунок 4: Очереди

| Имя / номер | Значение ключа в конце моделирования | Кол-во тран., ожидающих выполнения спец. условия |
|-------------|-----------------------------------------|-----------------------------------------------------|
| MAN_SEM | 0 | 1 |
| BTN | 1 | 1 |
| CAR_SEM | 1 | 0 |

Рисунок 5: Логические переключатели

Отчет по программе-симулятору

Программа построена на принципе последовательного исполнения действий, помещаемых по ходу действия программы в цепь будущих событий.

На каждом этапе исполнения программы из массива хранимых событий извлекается событие с минимальным временем, модельное время обновляется до времени извлеченного события и исполняется действие, ассоциированное с этим событием.

Отдельно хранятся состояния элементов модели, которые могут меняться при исполнении программы и влияют на ее поведение.

В программе объявлен класс `semaphore_w_button` методы которого являются основными действиями, меняющими состояние модели, поток входящих автомобилей и пешеходов реализуется функциями `car_generator` и `people_generator`.

program.py

```
import os
import sys
import random
from math import log

import matplotlib.pyplot as plt

from sim import Color, semaphore
from sim import Sim as simulation
from sim import queue

simulation_time = 3600

simul = simulation(simulation_time)

car_queue = queue()
man_queue = queue()

def exp_generator(mean):
    return -log(1 - random.random()) * mean

class semaphore_w_button():
    def __init__(self):
        self.button_active = True

        self.car_sem = semaphore(start_color=Color.RED)
        self.people_sem = semaphore(start_color=Color.RED)
        self.press_button_event = None
        self.change_state_time = 0
```

```

self.btn_delay_time = 0

def press_button(self):
    if sem.button_active == True:
        sem.button_active = False
        simul.new_event(self.allow_people_pass, 10)

def button_activate(self):
    self.button_active = True

def deprecate_car_pass(self):
    self.car_sem.value = Color.RED
    simul.new_event(self.button_activate, self.btn_delay_time)

def allow_people_pass(self):
    self.deprecate_car_pass()
    self.people_sem.value = Color.GREEN

    global man_queue
    man_queue.free(simul.time)

    simul.new_event(self.deprecate_people_pass, 7)
    simul.new_event(self.allow_car_pass, 10)

def deprecate_people_pass(self):
    self.people_sem.value = Color.RED

def car_generator_pass(self):
    if self.car_sem.value != Color.GREEN:
        return

    global car_queue
    car_queue.sub(simul.time)

    if simul.time - self.change_state_time > 10:
        mean = 60. / 50.
    else:
        mean = 60. / 10.
    simul.new_event(self.car_generator_pass, exp_generator(mean))

def allow_car_pass(self):
    if self.people_sem.value == Color.GREEN:
        raise BaseException
    self.car_sem.value = Color.GREEN

    self.change_state_time = simul.time

    simul.new_event(self.car_generator_pass, 0)

sem = semaphore_w_button()

```

```

def car_generator():
    simul.new_event(car_generator, exp_generator(60. / 25.))
    global car_queue
    car_queue.add(simul.time)

def people_generator():
    if sem.people_sem.value == Color.RED:
        sem.press_button()

    global man_queue
    man_queue.add(simul.time)

    simul.new_event(people_generator, exp_generator(60. / 2.))

def plot_results():
    fig, axes = plt.subplots(2, 1, figsize=(12, 9), sharex=True)
    fig.set_dpi(100)
    axes[0].plot(car_queue.times, car_queue.vals, linewidth=2.0, linestyle="-")
    axes[0].set_xlabel("time", fontsize=16)
    axes[0].set_ylabel("car queue size", fontsize=16)
    axes[0].grid()

    axes[1].plot(man_queue.times, man_queue.vals, linewidth=2.0, linestyle="-")
    axes[1].set_xlabel("time", fontsize=16)
    axes[1].set_ylabel("man queue", fontsize=16)
    axes[1].grid()

    # axes[0].plot([0, car_queue.time], [car_queue.get_mean_val(),
    #                                     car_queue.get_mean_val()], linewidth=1.0,
    #               linestyle="--")
    # axes[1].plot([0, man_queue.time], [man_queue.get_mean_val(),
    #                                     man_queue.get_mean_val()], linewidth=1.0,
    #               linestyle="--")

    plt.tight_layout()
    plt.show()

def main():
    sem.btn_delay_time = 100

    car_generator()
    people_generator()

    sem.allow_people_pass()

    while simul.pop_event() != None:
        if simul.time > simul.sim_time:
            break

```

```

        # print("time:", simul.time)
        simul.event.action()

    print("mean queue len: ", car_queue.get_mean_val(), man_queue.get_mean_val())

    plot_results()

if __name__ == "__main__":
    main()

```

sim.py

```

import os
import sys
import math
import numpy as np

from enum import IntEnum, auto
from itertools import cycle
from operator import attrgetter

class Color(IntEnum):
    _order_ = 'GREEN RED'
    GREEN = 1
    RED = 2

class queue():
    def __init__(self):
        self.size = 0
        self.times = []
        self.vals = []
        self.time = 0
        self.sum = 0

    def plot_update(self, time, next_val):
        self.vals.append(self.size)
        self.times.append(time)
        self.sum += self.size * (time - self.time)

        self.vals.append(next_val)
        self.times.append(time)

        self.time = time
        self.size = next_val

    def add(self, time):
        self.plot_update(time, self.size + 1)

    def sub(self, time):

```



```

        if self.size > 0:
            self.plot_update(time, self.size - 1)

    def free(self, time):
        self.plot_update(time, 0)

    def get_mean_val(self):
        return self.sum / self.time

class semaphore():

    colors = ['GREEN', 'RED']

    def __init__(self, start_color=Color.RED):
        self.sem = cycle(enumerate(semaphore.colors, start=start_color))
        self.value = next(self.sem)

    def turn_ligth(self):
        self.value = next(self.sem)
        pass

class sim_event():

    def __init__(self, action, sys_time, time):
        self.time = sys_time + time
        self.action = action

class Sim(sim_event):
    next_evets_queue = []
    time = 0

    def finished(self):
        print("sim done")

    def new_event(self, action, time):
        event = sim_event(action, self.time, time)
        self.next_evets_queue.append(event)

    def __init__(self, sim_time):
        self.sim_time = sim_time
        self.event = sim_event(None, 0, 0)
        self.new_event(self.finished, sim_time)

    def pop_event(self):
        """
        pops event with min time from queue self.event
        """
        if len(self.next_evets_queue) == 0:

```

```

        return None
    min_val = min(self.next_evets_queue, key=attrgetter('time'))
    minpos = self.next_evets_queue.index(min_val)
    self.event = self.next_evets_queue[minpos]
    self.time = min_val.time

    self.next_evets_queue.remove(self.event)
    return 0

if __name__ == "__main__":
    pass

```

Результаты работы программы

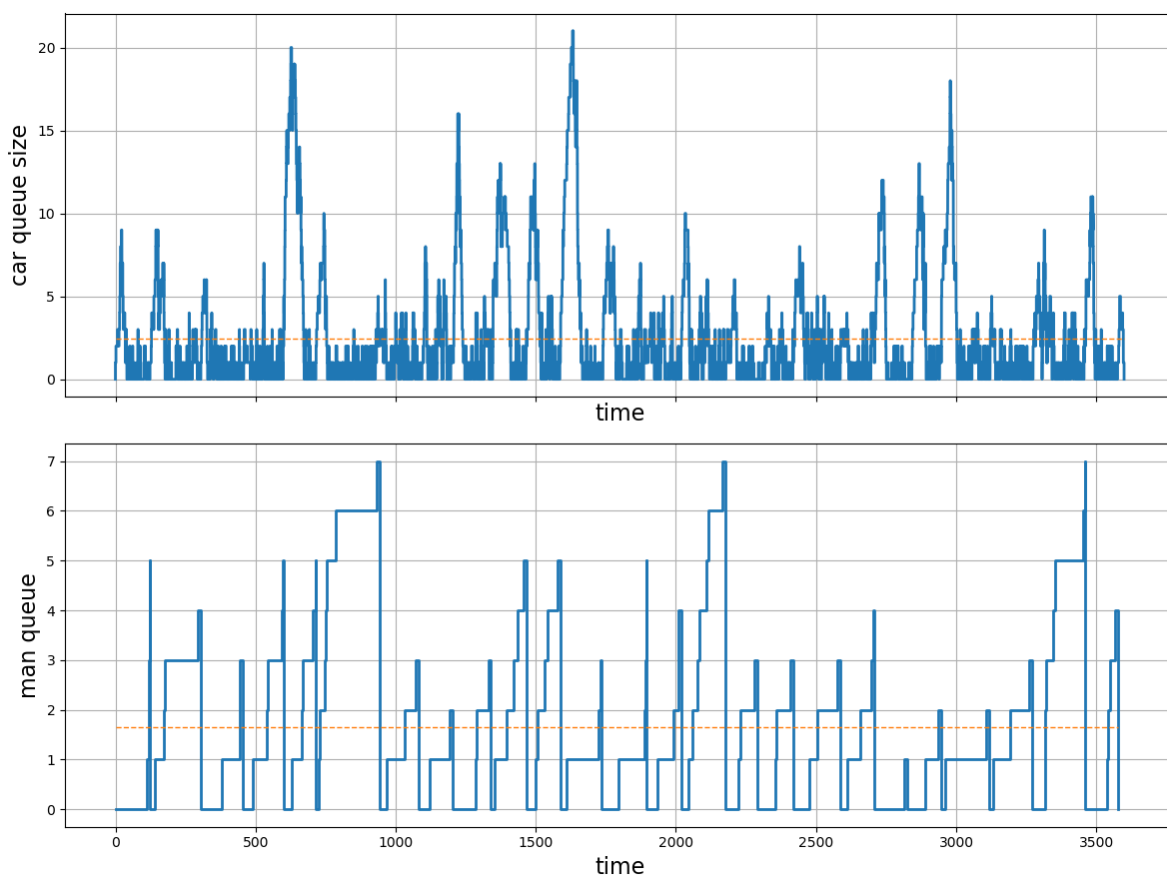


Рисунок 6: величины очередей машин и пешеходов

На рисунке 6 пунктиром показана средняя средняя длина очереди.

Исследовать модификацию системы, в которой остановка движения автомобилей допускается не чаще, чем раз в 40 с.

| минимальное время между остановками движения | средняя длина очереди автомобилей | средняя длина очереди пешеходов |
|----------------------------------------------|-----------------------------------|---------------------------------|
| 0 | 12.210879739656688 | 0.23339623867578385 |
| 100 | 3.3536987574151538 | 1.6355950451017411 |