

**Національний Технічний Університет України КПІ**

Факультет Інформатики та Обчислювальної Техніки  
Кафедра інформатики та програмної інженерії

**Практична робота №3**

З дисципліни «Моделювання систем»

ПОБУДОВА ІМІТАЦІЙНОЇ МОДЕЛІ СИСТЕМИ З ВИКОРИСТАННЯМ

ФОРМАЛІЗМУ МОДЕЛІ МАСОВОГО ОБСЛУГОВУВАННЯ

**Перевірила:**

Асистент

Бернатович Анатолій Олександрович

Оцінка:\_\_\_\_\_\_\_\_\_\_\_

**Виконав:**

Студент групи ІТ-92

Рафалюк Р.Б.

Дата: \_\_\_\_\_\_\_\_\_\_\_\_\_

**Завдання до практичної роботи**

1. Реалізувати універсальний алгоритм імітації моделі масового обслуговування з багатоканальним обслуговуванням, з вибором маршруту за пріоритетом або за заданою ймовірністю. **30 балів.**
2. Для наступного тексту задачі скласти формалізовану модель масового обслуговування та реалізувати її з використанням побудованого універсального алгоритму **(30 балів)**:

У банку для автомобілістів є два віконця, кожне з яких обслуговується одним касиром і має окрему під'їзну смугу. Обидві смуги розташовані поруч. З попередніх спостережень відомо, що інтервали часу між прибуттям клієнтів у годину пік розподілені експоненційно з математичним очікуванням, рівним 0,5 од. часу. Через те, що банк буває переобтяжений тільки в годину пік, то аналізується тільки цей період. Тривалість обслуговування в обох касирів однакова і розподілена експоненційно з математичним очікуванням, рівним 0,3 од. часу. Відомо також, що при рівній довжині черг, а також при відсутності черг, клієнти віддають перевагу першій смузі. В усіх інших випадках клієнти вибирають більш коротку чергу. Після того, як клієнт в'їхав у банк, він не може залишити його, доки не буде обслугований. Проте він може перемінити чергу, якщо стоїть останнім і різниця в довжині черг при цьому складає не менше двох автомобілів. Через обмежене місце на кожній смузі може знаходитися не більш трьох автомобілів. У банку, таким чином, не може знаходитися більш восьми автомобілів, включаючи автомобілі двох клієнтів, що обслуговуються в поточний момент касиром. Якщо місце перед банком заповнено до границі, то клієнт, що прибув, вважається втраченим, тому що він відразу ж виїжджає. Початкові умови такі: 1) обидва касири зайняті, тривалість обслуговування для кожного касира нормально розподілена з математичним очікуванням, рівним 1 од. часу, і середньоквадратичним відхиленням, рівним 0,3 од. часу; 2) прибуття першого клієнта заплановано на момент часу 0,1 од. часу; 3) у кожній черзі очікують по два автомобіля.

Визначити такі величини: 1) середнє завантаження кожного касира; 2) середнє число клієнтів у банку; 3) середній інтервал часу між від'їздами клієнтів від вікон; 4) середній час перебування клієнта в банку; 5) середнє число клієнтів у кожній черзі; 6) відсоток клієнтів, яким відмовлено в обслуговуванні; 7) число змін під'їзних смуг.

1. Для наступного тексту задачі скласти формалізовану модель масового обслуговфцування та реалізувати її з використанням побудованого універсального алгоритму **(40 балів)**:

У лікарню поступають хворі таких трьох типів: 1) хворі, що пройшли попереднє обстеження і направлені на лікування; 2) хворі, що бажають потрапити в лікарню, але не пройшли повністю попереднє обстеження; 3) хворі, які тільки що поступили на попереднє обстеження. Чисельні характеристики типів хворих наведені в таблиці:

|  |  |  |
| --- | --- | --- |
| *Тип хворого* | *Відносна частота* | *Середній час*  *реєстрації, хв* |
| 1 | 0,5 | 15 |
| 2 | 0,1 | 40 |
| 3 | 0,4 | 30 |

При надходженні в приймальне відділення хворий стає в чергу, якщо обидва чергових лікарі зайняті. Лікар, який звільнився, вибирає в першу чергу тих хворих, що вже пройшли попереднє обстеження. Після заповнення різноманітних форм у приймальне відділення хворі 1 типу ідуть прямо в палату, а хворі типів 2 і 3 направляються в лабораторію. Троє супровідних розводять хворих по палатах. Хворим не дозволяється направлятися в палату без супровідного. Якщо всі супровідні зайняті, хворі очікують їхнього звільнення в приймальному відділенні. Як тільки хворий доставлений у палату, він вважається таким, що завершив процес прийому до лікарні.

Хворі, що спрямовуються в лабораторію, не потребують супроводу. Після прибуття в лабораторію хворі стають у чергу в реєстратуру. Після реєстрації вони ідуть у кімнату очікування, де чекають виклику до одного з двох лаборантів. Після здачі аналізів хворі або повертаються в приймальне відділення (якщо їх приймають у лікарню), або залишають лікарню (якщо їм було призначено тільки попереднє обстеження). Після повернення в приймальне відділення хворий, що здав аналізи, розглядається як хворий типу 1.

У наступній таблиці приводяться дані по тривалості дій (хв):

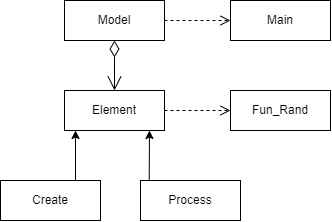
|  |  |
| --- | --- |
| *Величина* | *Розподіл* |
| Час між прибуттями в приймальне відділення | Експоненціальний з математичним сподіванням 15 |
| Час слідування в палату | Рівномірне від 3 до 8 |
| Час слідування з приймального відділення в лабораторію або з лабораторії в приймальне відділення | Рівномірне від 2 до 5 |
| Час обслуговування в реєстратуру  лабораторії | Ерланга з математичним сподіванням 4,5 і *k*=3 |
| Час проведення аналізу в лабораторії | Ерланга з математичним сподіванням 4 і *k*=2 |

Визначити час, проведений хворим у системі, тобто інтервал часу, починаючи з надходження і закінчуючи доставкою в палату (для хворих типу 1 і 2) або виходом із лабораторії (для хворих типу 3). Визначити також інтервал між прибуттями хворих у лабораторію.

**Виконання лабораторної роботи**

Для виконання даної лабораторної роботи була обрана мова програмування Python та середовище розробки Visual Studio, так як воно надає безліч різноманітних бібліотек для роботи із графіками, гістограмами та математичними функціями, що робить дані інструменти дуже зручними для виконання завдань лабораторної роботи.

Структура об’єктно-орієнтованої програми така сама як і в минулій лабораторній:



Побудуємо просту модель із використанням канальної системи, а саме із одним пристроєм, у якого є два канали. Для порівняння ефективності, ми будемо використовувати таку ж саму модель, тільки з одним каналом.

Спочатку для перевірки еффективності пристрою з 2-ма каналами, побудуємо просу модель з 1 каналом:

def simple\_model():

print('Simple model')

c1 = Create(delay\_mean=5, name='CREATOR', distribution='exp')

p1 = Process(max\_queue=3, delay\_mean=5, distribution='exp')

c1.next\_element = [p1]

elements = [c1, p1]

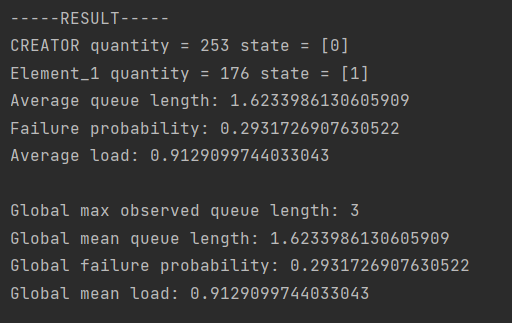
model = Model(elements)

model.simulate(1000)

Де:

* “delay\_mean” – математичне очікування;
* “distribution” – тип розподілу часу;
* “max\_queue” – максимальна черга на пристрій;
* “name” – назва пристрою;
* “c1.next\_element” – вказання на наступний пристрій у моделі (тут, наприклад, ми кажемо, що після “CREATE” буде йти “PROCESS”);
* “model.simulate” – виклик методу для початку симуляції.

Результат симуляції моделі «simple\_model»:



Де:

* “quantity” – кількість опрацьованих елементів пристроєм
* “Average queue length” – середня довжина черги пристроя;
* “Failure probability” – ймовірність не опрацювання завдання пристроєм;
* “Average load” – середнє завантаження пристроя;
* “Global max observed queue length” – найбільший розмір черги, який спостерігався у моделі;
* “Global mean queue length” – середнє значення черги у всієї моделі;
* “Global failure probability” – ймовірність не опрацювання завдань всією моделлю;
* “Global mean load” – середнє завантаження пристроїв моделі.

Тепер побудуємо таку саму модель із каналами:

def channel\_model():

print('Channel model')

c1 = Create(delay\_mean=5, name='CREATOR', distribution='exp')

p1 = Process(max\_queue=3, n\_channel=2, delay\_mean=5, distribution='exp')

c1.next\_element = [p1]

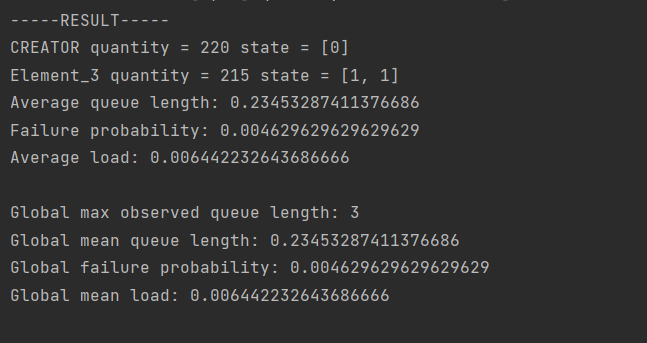
elements = [c1, p1]

model = Model(elements)

model.simulate(1000)

Де “n\_channel” – кількість каналів.

Результат симуляції моделі «channel\_model»:



Так, можна бачити, що з додаванням другого каналу, сильно знижується завантаження пристрою, ймовірність відмови та середня довжина черги. Тому, можна зробити висновок, що можна досягти кращого результату опрацювання елементів пристроєм, якщо використовувати модель з кількома каналами.

Для наступних моделей ми створимо функцію «base\_model» для більш зручої їх побудови:

def base\_model(p1):

c1 = Create(delay\_mean=5, name='CREATOR', distribution='exp')

p2 = Process(max\_queue=3, delay\_mean=5, distribution='exp')

p3 = Process(max\_queue=3, delay\_mean=5, distribution='exp')

c1.next\_element = [p1]

p1.next\_element = [p2, p3]

elements = [c1, p1, p2, p3]

model = Model(elements)

model.simulate(1000)

Тепер побудуємо модель, де маршрут буде обиратись за деякою ймовірністю:

def probability\_model():

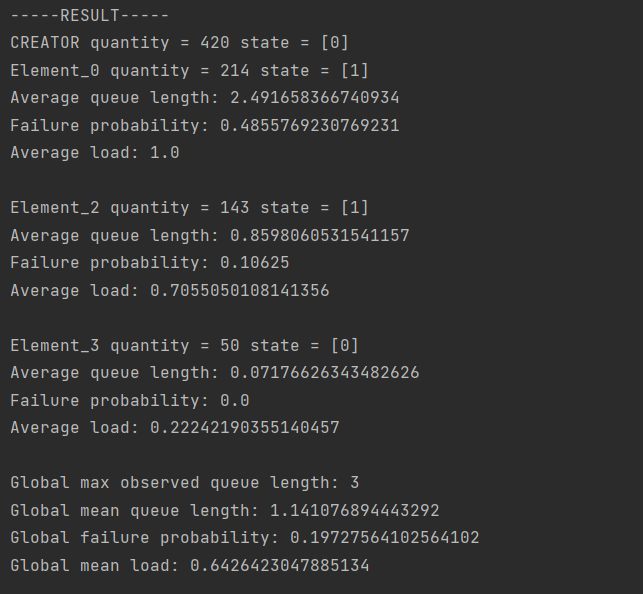
print('Probability model')

p1 = Process(max\_queue=3, delay\_mean=5, distribution='exp')

p1.probability = [0.8, 0.2]

base\_model(p1)

Тут p1.probability розприділяє ймовірність вибору пристроїв(р2 = 0.8 а р3 = 0.2). Запустимо програму та подивимося на результат:



Як бачимо, другий елемент має набагато більше опрацьованих елементів ніж третій, який мав менший шанс бути обраним для опрацювання події.

Також побудуємо модель, де ми будемо обирати маршрут за пріоритетом:

def priority\_model():

print('Priority model')

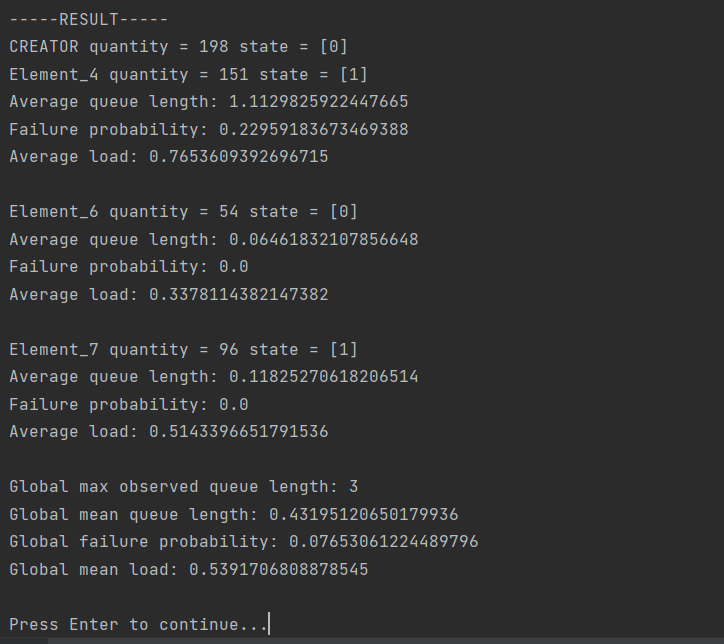
p1 = Process(max\_queue=3, delay\_mean=5, distribution='exp')

p1.priority = [2, 1]

base\_model(p1)

Тут p1.priority задає пріоритет для p2 – 2, а для p3 – 1, тобто p3 має обиратися частіше.

Виконаємо програму та подивимося на результат:



Як можна бачити, третій пристрій обирався частіше ніж другий, так як мав більший пріоритет.

Тепер побудуємо модель автомобільного банку, для чого ми модифікували універсальний алгоритм, який ми реалізували у минулому завданні.

def bank\_model():

print('Bank model')

c1 = CreateBank(delay\_mean=0.5, name='CREATOR', distribution='exp')

p1 = ProcessBank(max\_queue=3, delay\_mean=0.3, name='CASHIER\_1', distribution='exp')

p2 = ProcessBank(max\_queue=3, delay\_mean=0.3, name='CASHIER\_2', distribution='exp')

c1.next\_element = [p1, p2]

p1.state[0] = 1

p2.state[0] = 1

p1.t\_next[0] = fun.norm(1, 0.3)

p2.t\_next[0] = fun.norm(1, 0.3)

c1.t\_next[0] = 0.1

p1.queue = 2

p2.queue = 2

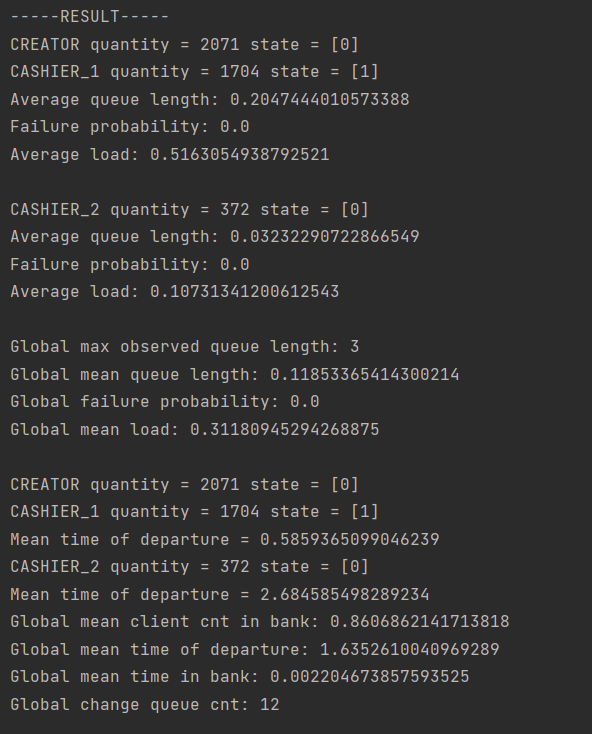
element\_list = [c1, p1, p2]

bank = ModelBank(element\_list, balancing=[p1, p2])

bank.simulate(1000)

Де “fun.norm” – функція визначення нормально розподілених значень в залежності від параметрів. “CASHIER\_1” та “CASHIER\_2” – відповідні касири за номерами. Моделювання проходить на проміжку часу в 1000 часових одиниць.

Запустимо програму та подивимося на результати:



Із результатів ми можемо побачити, що оскільки віддавалась перевага першій смузі, то і касир один опрацював більше подій, ніж другий.

Також ми отримали наступні дані:

* Завантаженість касира 1 – 0.516, касира 2 – 0.107
* Середнє число клієнтів у банку – 0.86
* Середній інтервал часу між від’їздами клієнтів від вікон – 1.635
* Середній час перебування клієнта у банку – 0.002
* Середнє число клієнтів у черзі до першого касира – 0.2, до другого – 0.03
* Всіх клієнтів вдалося обслугувати так як відсоток тих, кому відмовлено – 0%
* Число змін під’їзних смуг – 12.

Також, схожим чином цпобудуємо формалізовану модель масового обслуговування лікарні, як наведено у завданні. Для цього, як і у попередньому завданні, ми модифікували базовий алгоритм. Тут ми додали такий елемент як DisposeHospital. Він потрібний для індикації завершення обслуговування а також для збору інформації по кількості обслугованих хворих.

def hospital\_model():

print('Hospital model')

c1 = CreateHospital(delay\_mean=15.0, name='CREATOR\_1', distribution='exp')

p1 = ProcessHospital(max\_queue=100, n\_channel=2, name='RECEPTION', distribution='exp')

p2 = ProcessHospital(max\_queue=100, delay\_mean=3.0, delay\_dev=8, n\_channel=3, name='FOLLOWING\_TO\_THE\_WARD',

distribution='unif')

p3 = ProcessHospital(max\_queue=0, delay\_mean=2.0, delay\_dev=5, n\_channel=10, name='FOLLOWING\_TO\_THE\_LAB\_RECEPTION',

distribution='unif')

p4 = ProcessHospital(max\_queue=100, delay\_mean=4.5, delay\_dev=3, n\_channel=1, name='LAB\_REGISTRY',

distribution='erlang')

p5 = ProcessHospital(max\_queue=100, delay\_mean=4.0, delay\_dev=2, n\_channel=1, name='EXAMINATION',

distribution='erlang')

p6 = ProcessHospital(max\_queue=0, delay\_mean=2.0, delay\_dev=5, n\_channel=10, name='FOLLOWING\_TO\_THE\_RECEPTION',

distribution='unif')

d1 = DisposeHospital(name='EXIT1')

d2 = DisposeHospital(name='EXIT2')

c1.next\_element = [p1]

p1.next\_element = [p2, p3]

p2.next\_element = [d1]

p3.next\_element = [p4]

p4.next\_element = [p5]

p5.next\_element = [d2, p6]

p6.next\_element = [p1]

p1.prior\_types = [1]

p1.required\_path = [[1], [2, 3]]

p5.required\_path = [[3], [2]]

elements = [c1, p1, p2, p3, p4, p5, p6, d1, d2]

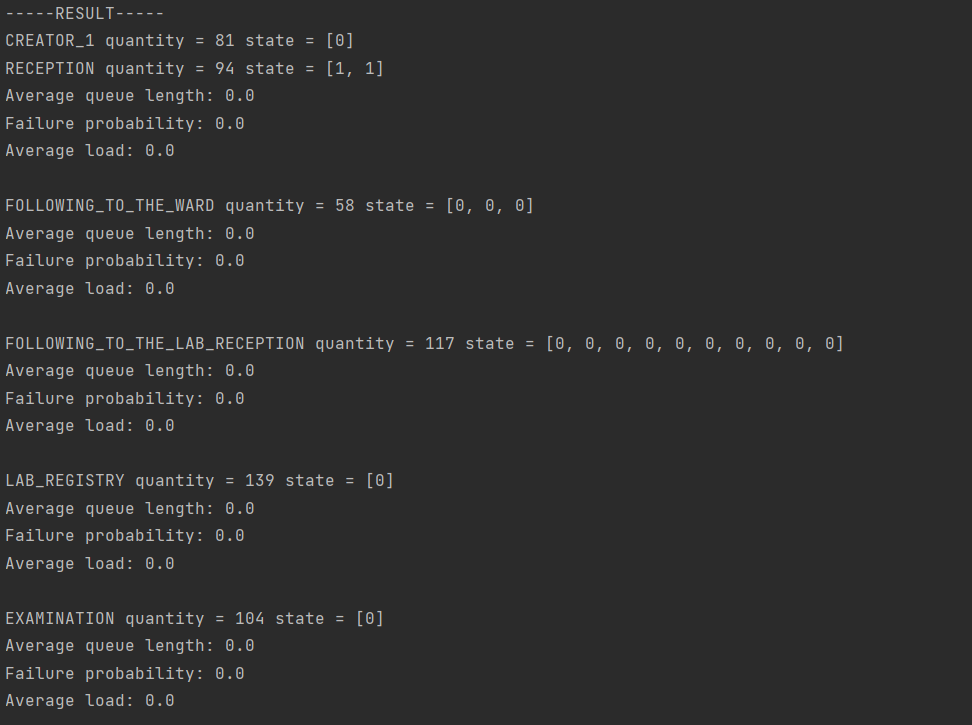
model = ModelHospital(elements)

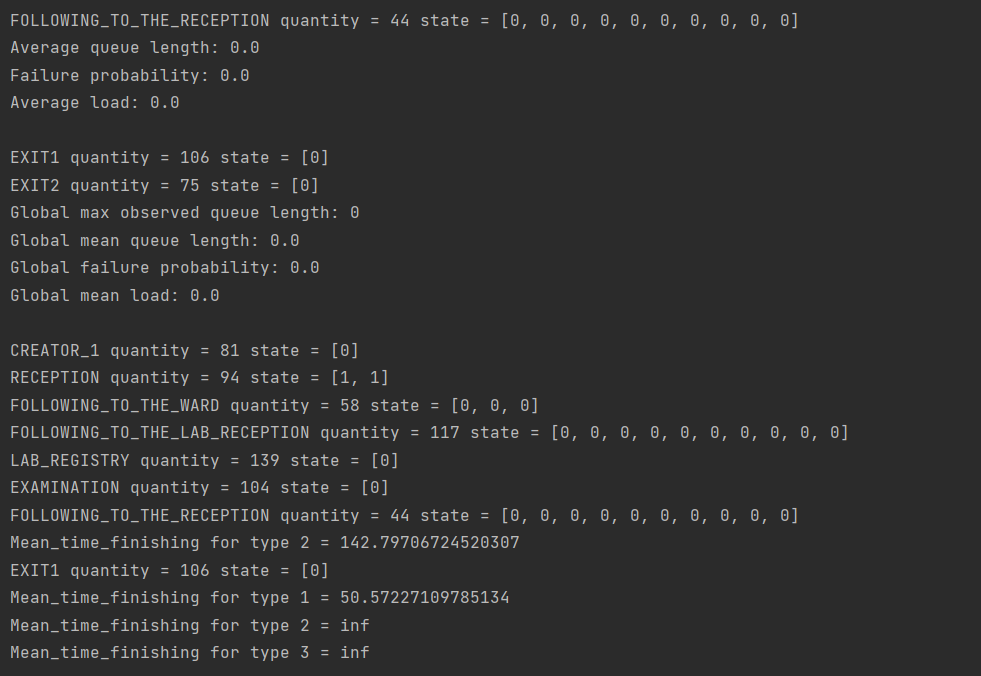
model.simulate(1000)

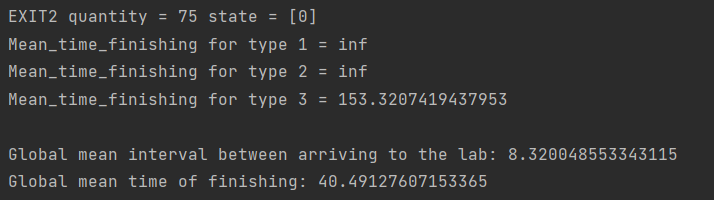
Де

* “CREATOR\_1” та “RECEPTION” – прибуття в приймальне відділення,
* “FOLLOWING\_TO\_THE\_WARD” – слідування в палату,
* “FOLLOWING\_TO\_THE\_LAB\_RECEPTION” – слідування у лабораторію,
* “'LAB\_REGISTRY” – обслуговування у реєстратурі лабораторії,
* “EXAMINATION” – проведення аналізу,
* “FOLLOWING\_TO\_THE\_RECEPTION” – перехід у приймальне відділення.

Виконаємо програму та подивимося на результати:







Маємо такі дані:

1. Інтервал часу, починаючи з надходження і закінчуючи доставкою в палату – 50.57 (оскільки після здачі аналізів хворий другого типу буде вважатися як хворий першого типу, а тому значення одне) Інтервал часу виходу із лабораторії – 153.
2. Інтервал часу між прибуттям хворих у лабораторію – 8.32.

**Висновок**

Під час виконання даної лабораторної роботи ми реалізували універсальний алгоритм імітації моделі масового обслуговування з багатоканальним обслуговуванням, з вибором маршруту за пріоритетом або за заданою ймовірністю.

Також було побудувано формалізовану модель масового обслуговування для задачі із банком для автомобілістів та лікарні та реалізовано їх з використанням побудованого універсального алгоритму.

Посилання на гітхаб: <https://github.com/RomBik17/modeling.git>