**Завдання до практичної роботи**

1. Реалізувати алгоритм імітації простої моделі обслуговування одним пристроєм з використанням об’єктно-орієнтованого підходу. **5 балів.**
2. Модифікувати алгоритм, додавши обчислення середнього завантаження пристрою. **5 балів.**
3. Створити модель за схемою, представленою на Рисунку 2.1. **30 балів.**
4. Виконати верифікацію моделі, змінюючи значення вхідних змінних та параметрів моделі. Навести результати верифікації у таблиці. **10 балів.**

Изображение выглядит как диаграмма, План, линия, Прямоугольник

Автоматически созданное описание

Рисунок 2.1 – Схема моделі

1. Модифікувати клас PROCESS, щоб можна було його використовувати для моделювання процесу обслуговування кількома ідентичними пристроями. **20 балів.**
2. Модифікувати клас PROCESS, щоб можна було організовувати вихід в два і більше наступних блоків, в тому числі з поверненням у попередні блоки. **30 балів.**

**Хід роботи**

Комп’ютерний практикум був реалізований універсальною мовою програмування Python. Ця мова є найбільш зручною, бо надає достатню кількість бібліотек для роботи з обробкою даних в форматі Excel, із математичними функціями, графіками, гістограмами. Для виконання математичних операцій використовувалась бібліотека Numpy.

Крім того, було вирішено використовувати PyCharm, сумісний із цією мовою програмування, що дозволяє нам створювати і відображати код, результати обчислень і графіки в одному інтерактивному середовищі, і є більш зручним середовищем для масивних проектів, аніж Jupyter Notebook.

Алгоритм для імітації простої моделі обслуговування одним пристроєм, використовуючи об'єктно-орієнтований підхід, створюється, враховуючи приклад, який був представлений на лекції, і розраховується середнє завантаження пристрою.

Вимоги на обслуговування генеруються елементом CREATE і відправляються на обслуговування до елементу PROCESS, який здійснює обслуговування з часовою затримкою, заданою випадковим числом, та обмеження на довжину черги, заданим невід’ємним числом. Побудова алгоритму представлена на Рисунку 2.2

Изображение выглядит как диаграмма, текст, Шрифт, линия

Автоматически созданное описание

Рисунок 2.2 – Структура моделі

Структуру об’єктно-орієнтованої програми представимо діаграмою класів (рис. 2.3). Модель складається з елементів, які є нащадками одного універсального типу Element. Цей клас містить основні поля та методи елементу моделі такі, як tcurr (поточний момент часу), tnext (момент часу наступної події), delayMean (середеє значення часової затримки), delayDev(середнє квадратичне відхилення часової затримки), getDelay() (розрахунок часової затримками), inAct() (вхід в елемент), outAct() (вихід з елементу). Використання універсального класу надає можливість уніфікувати використання різних елементів в імітаційній програмі.

Важливим для з’єднання елементів в єдину модель є поле nextElement, що вказує на наступний (в маршруті слідування вимоги) елемент моделі.

Клас Model містить метод simulate(double time), що здійснює імітацію на інтервалі часу time. Імітація здійснюється за відомим з попередньої теми принципом: визначається момент найближчої події, просувається час в момент найближчої події та здійснюється відповідна подія. Щоб зменшити обсяг обчислень, введемо також здійснення відповідної події для всіх елементів, час наступної події яких співпадає з поточним моментом часу. Оскільки ми не розглядаємо можливість присвоєння пріоритету елементам моделі, то ця дія цілком допустима.

Изображение выглядит как текст, диаграмма, линия, снимок экрана

Автоматически созданное описание

Рисунок 2.3 – Діаграма класів

Delay для CREATE та PROCESS встановлено 5, максимальне значення черги – 10, інтервал імітації(час моделювання) - 1000 одиниць. Для розрахунку, коли відбудиться наступна подія використовується експоненційний розподіл, що і провокує чергу.

from create import Create  
from process import Process  
from model import Model  
  
  
c = Create(5)  
  
p1 = Process(5)  
  
p1.max\_queue = 10  
  
c.distribution = 'exp'  
p1.distribution = 'exp'  
  
c.name = 'CREATE'  
p1.name = 'PROCESS-1'  
  
c.next\_element = [p1]  
  
elements = [c, p1]  
model = Model(elements)  
res = model.simulate(1000)

Приклад роботи:

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

“Quantity” – кількість опрацьованих елементів процесами CREATE(209) ТА PROCESS(192), “mean length of queue = 3.1818023354299694” – середня довжина черги для PROCESS-1, “failure probability = 0.036458333333333336” – ймовірність не опрацювання завдання для PROCESS-1.

Також для практичного практикуму було додано реалізацію середнього завантаження пристрою, що залежить від DELAY, стану та загального часу. Для класу PROCESS було додано можливість для моделювання процесу обслуговування кількома ідентичними пристроями, використовуючи систему каналів. Тобто у кожного об'єкта класу Process може бути декілька каналів, які виконуватимуть корисну роботу. Якщо один із каналів зайнятий, інший буде продовжувати виконувати роботу, що призведе до зменшення кількості елементів у черзі та ймовірність відмов.

c = Create(5)  
  
p1 = Process(5, 2)  
  
p1.max\_queue = 10  
  
c.distribution = 'exp'  
p1.distribution = 'exp'  
  
c.name = 'CREATE'  
p1.name = 'PROCESS-1'  
  
c.next\_element = [p1]  
  
elements = [c, p1]  
model = Model(elements)  
res = model.simulate(1000)

У даному випадку відображено роботу із Delay для CREATE та PROCESS - 5, максимальне значення черги – 10, інтервал імітації(час моделювання) - 1000 одиниць, до того ж процес є двоканальним, шо має зменшити навантаження, середнє значення черги та ймовірність відмов для PROCESS.

Приклад роботи:

Изображение выглядит как текст, Шрифт, снимок экрана, алгебра

Автоматически созданное описание

Де mean load – середнє завантаження пристрою.

Як бачимо, дійсно необхідні показники впали і досягли майже нульового значення.

За прикладом таблиці верифікації моделі з лекції було створено таблицю з ідентичними показниками.

Изображение выглядит как текст, Параллельный, число, черно-белый

Автоматически созданное описание

Таблиця 1 – Верифікація моделі масового обслуговування (лекція)

Для операцій з електронними таблицями, було використано бібліотеку openpyxl, що надає зручний і потужний інтерфейс для взаємодії з файлами Excel, дозволяючи нам легко маніпулювати даними у форматі, зрозумілому для багатьох користувачів.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| delay\_create | channels\_process1 | max\_queue1 | delay\_process1 | channels\_process2 | max\_queue2 | delay\_process2 | channels\_process3 | max\_queue3 | delay\_process3 |
| 5 | 1 | 5 | 5 | 1 | 5 | 5 | 1 | 5 | 5 |
| 10 | 1 | 5 | 5 | 1 | 5 | 5 | 1 | 1 | 5 |
| 5 | 3 | 5 | 5 | 3 | 5 | 5 | 3 | 5 | 5 |
| 5 | 1 | 5 | 5 | 3 | 5 | 10 | 2 | 5 | 5 |
| 5 | 1 | 5 | 5 | 1 | 5 | 5 | 3 | 5 | 10 |
| 5 | 2 | 10 | 5 | 1 | 5 | 5 | 1 | 5 | 5 |
| 1 | 3 | 5 | 5 | 2 | 10 | 5 | 1 | 5 | 5 |
| 5 | 1 | 5 | 1 | 3 | 5 | 5 | 2 | 10 | 5 |
| 5 | 1 | 5 | 5 | 1 | 5 | 1 | 3 | 5 | 5 |
| 5 | 1 | 1 | 5 | 1 | 5 | 5 | 1 | 5 | 1 |

Таблиця 2.0 – вхідні дані для верифікації моделі масового обслуговування

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| quantity\_create | quantity\_p1 | fail1 | queue1 | load1 | quantity\_p2 | fail2 | queue2 | load2 | quantity\_p3 | fail3 | queue3 | load3 |
| 192 | 178 | 0,1 | 1,8978 | 0,82 | 168 | 0 | 1,3215 | 0,79 | 155 | 0,1 | 1,9654 | 0,79 |
| 97 | 96 | 0 | 0,3716 | 0,53 | 96 | 0 | 0,5466 | 0,55 | 85 | 0,1 | 0,0984 | 0,4 |
| 206 | 204 | 0 | 0,0145 | 0 | 202 | 0 | 0,0928 | 0 | 200 | 0 | 0,0232 | 0 |
| 191 | 156 | 0,2 | 2,3313 | 0,85 | 155 | 0 | 0,0505 | 0 | 155 | 0 | 0,0891 | 0 |
| 215 | 171 | 0,2 | 2,5619 | 0,93 | 155 | 0,1 | 1,6526 | 0,71 | 152 | 0 | 0,0779 | 0 |
| 187 | 187 | 0 | 0,4136 | 0 | 155 | 0,2 | 2,1933 | 0,84 | 152 | 0 | 1,2033 | 0,7 |
| 982 | 639 | 0,5 | 3,3942 | 0 | 418 | 0,5 | 8,5339 | 0 | 193 | 1,2 | 4,2239 | 0,99 |
| 195 | 194 | 0 | 0,0399 | 0,2 | 192 | 0 | 0,0577 | 0 | 188 | 0 | 0,4648 | 0 |
| 213 | 184 | 0,1 | 2,3383 | 0,88 | 183 | 0 | 0,0399 | 0,18 | 182 | 0 | 0,0164 | 0 |
| 206 | 133 | 0,5 | 0,3796 | 0,74 | 131 | 0 | 0,5474 | 0,6 | 131 | 0 | 0,0094 | 0,14 |

Таблиця 2.1 – вихідні дані для верифікації моделі масового обслуговування

З таблиці видно, що змінюючи параметри моделі, ми спостерігаємо очікувані зміни в результатах. Наприклад, підвищення значення черги призводить до зменшення кількості необроблених елементів, і навпаки, зменшення значення черги призводить до збільшення кількості необроблених елементів. Також, підвищення затримок призводить до високої кількості необроблених елементів на поточному пристрої і низької на наступних пристроях. Зменшення затримок на пристрої зменшує кількість необроблених елементів на ньому. З цього можна зробити висновок, що модель працює коректно.

Насотанок було реалізовано входження Process1 у кілька пристроїв одночасно з ймовірністю 0.7 та 0.3.

c = Create(5)  
  
p1 = Process(5)  
p2 = Process(5)  
p3 = Process(5)  
  
c.next\_element = [p1]  
p1.next\_element = [p2, p3]  
p1.probability = ([0.7, 0.3])  
  
p1.max\_queue = 5  
p2.max\_queue = 5  
p3.max\_queue = 5  
  
c.distribution = 'exp'  
p1.distribution = 'exp'  
p2.distribution = 'exp'  
p3.distribution = 'exp'  
  
c.name = 'CREATE'  
p1.name = 'PROCESS-1'  
p2.name = 'PROCESS-2'  
p3.name = 'PROCESS-3'  
  
elements = [c, p1, p2, p3]  
model = Model(elements)  
res = model.simulate(1000)

Приклад роботи:

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание