Міністерство освіти та науки України

Хмельницький національний університет

#### Кафедра комп’ютерних систем

##### КУРСОВИЙ ПРОЕКТ

Тема: "МОДЕЛЮВАННЯ ОДНОФАЗНОЇ СИСТЕМИ

З ПРІОРІТЕТОМ ОБСЛУГОВУВАННЯ"

**Виконав: ст.гр.КС-02-1**

**Чимишук В. І.**

**Перевірили:**

**Хмельницький 2004**

Зам.

.

Арк.

рк.

№ докум.

Підпис

Дата

Арк.

2

КРКЕ.2004.38.ПЗ

Розроб.

Чимишук В.

Перевір.

.

Реценз.

Н. Контр.

Затверд.

КУРСОВА РОБОТА

Пояснювальна записка

Літ.

Аркушів

ХНУ. гр. КС-02-1

***Зміст***

Анотація .................................................................................................3

Вступ........................................................................................................4

Умова завдання на курсову роботу.......................................................5

1.Моделювання та основні його задачі.................................................6

2.Системи масового обслуговування....................................................9

3.Імітаційне моделювання......................................................................12

4.Побудова структурної моделі..............................................................18

5.Побудова функціональної моделі........................................................18

6.Побудова концептуальної моделі........................................................19

7.Математичне та програмне забезпечення...........................................21

8.Блок-схеми.............................................................................................24

9.Технічні характеристики .....................................................................27

10.Результати моделювання....................................................................28

11.Висновки...............................................................................................33

Список використаної літератури

Додаток 1.

**Анотація**

В даній курсовій роботі змодельовано роботу системи масового обслуговування. В даній СМО є один обслуговуючий пристрій на якому відбувається обслуговування клієнтів. Поступають заявки на обслуговування, вони мають два рівня пріорітетів. Заявки з першим типом пріорітету мають більш високий пріорітет, і тому вони обслуговуються в першу чергу. Дана курсова робота моделює таку систему по обслуговуванню 1000 клієнтів.

Далі докладніше формулюється сама задача по моделюванню, приводиться текс програми на мові С++, наводяться результати моделювання, опис потрібного програмного та апаратного забезпечення необхідного для поставлення експерименту на ЕОМ. Додаються графіки що характеризують роботу системи.

***Вступ***

Аналітичні методи дослідження операцій здаються на перший погляд універсальними засобами для моделювання різноманітних явищ і процесів, які існують в житті. Але це не так, оскільки вони є надзвичайно динамічними, складнимим різноманітними. Виходом з такого положення може здатися застосування математичних моделей, але задопомогою них не завжди можна вирішувати усі завдання які постають перед нами. Математичні моделі які можуть бути реалізовані за допомогою ефективних обчислювальних методів, є надто спрощені, а отже не адекватні реальним процесам. Що ж до адекватних математичних моделей то їх здебільшого не можна реалізувати через труднощі обчислювального характеру. У такому разі є сенс застосувати машинну імітацію, що полягає в моделюванні на ЕОМ реальної виробничої чи економічної системи.

Машинна імітація дає змогу користувачеві проводити такі експеременти , які з існуючими реальними системами були б не можливі. Такий метод набуває сьогодні особливої ваги, насамперед як інструмент удосконалення управління економікою та її галузями в межах усієї України. Є всі підстави стверджувати, що машинна імітація – один з найзагальніших та найпотужніших методів прикладного системного аналізу. Адже з допомогою імітаційних моделей системи розкривається сутність відповідних явищ і процесів за умови, що натуральні досліди в реальному середовищі на реальних об’єктах виключаються.

***Умова завдання на курсову роботу:***

Опис завдання:

Усистему масового обслуговування надходять два типи клієнтів. Клієнти типу 1 мають більш високий пріорітет, тому клієнти типу 2 не обслуговуються доти, доки не будуть обслуговані всі клієнти першого типу. У той же час, обслуговування клієнта типу 2, яке вже почалося, не може бути перервано. Обслуговування клієнтів здійснюється по експотенційному закону, із середнім часом 3 і 2 хв. відповідно. Проміжки часу, між надходженнями клієнтів розподілені експотенційно із середнім часом 1.5 і 3 хв. відповідно. Система починає працювати при порожній черзі.

Завдання по моделюванню:

1. Змоделювати систему по обслуговуванню 1000 вимог з дисципліною обслуговування FIFO.

2. Обчислити середню кількість клієнтів кожного типу, що очікують, а також загальне середнє.

3. Знайти оцінку середнього часу чекання клієнтів кожного типу, а також загальне середнє значення.

4. Знайти оцінку загального середнього довжини черги в залежності від кількості вимог, що обслуговуються.

5. Визначити ймовірність простою системи.

6. Скласти можливу аналітичну модель.

7. Провести розрахунок можливих операційних характеристик з аналітичної моделі.

**1.Моделювання, основні його задачі.**

**Моделювання** – як процес полягає у відтворенні властивостей тих чи інших об’єктів , предметів і явищ за допомогою абстрактних об’єктів та описів у вигляді зображень , планів , систем рівнянь , алгоритмів або програм .

**Модель** – це умовний образ об’єкта або системи , який служить для відображення співвідношень між людськими знаннями про об’єкт і безпосередньо цим об’єктом .

У процесі моделювання відбувається заміщення об’єкта – оригінала його моделлю і вивчення властивостей об’єкта шляхом дослідження властивостей моделі Об’єктом може бути природна або штучна система . Під впливом зовнішніх дій проявляються властивості об’єкта . Характеристики У0 – кількісна міра властивості об’єкта . Кожен параметр має певну структуру і принципи , які його відображають . Параметр має підмножину параметрів .

Дослідника цікавлять тільки певні характеристики об’єкта при конкретних зовнішніх діях .Заміщення одного об’єкта іншим правомірне , якщо ті характеристики оригіналу і моделі , які цікавлять дослідника , визначають однотипними множинами параметрів і пов’язані однотипними залежностями між параметрами .

Модель , за допомогою якої описують систему, може містити у собі тільки основні характеристики системи, інші, які не чинять суттєвого впливу на функціонування системи, відкидаються, щоб не ускладнювати процес моделювання. У більшості випадків вважається, що модель складається з трьох основних блоків: вхід, система, вихід. Для того щоб успішно моделювати роботу системи необхідно знати два з цих блоків .

Загалом, модель є абстрактним або формально описаним об’єктом або системою, відображає співвідношення між людськими знаннями про об’єкт і безпосередньо сам об’єкт. Модель , за допомогою якої описують систему, може містити у собі тільки основні характеристики системи, інші, які не чинять суттєвого впливу на функціонування системи, відкидаються, щоб не ускладнювати процес моделювання. У більшості випадків вважається, що модель складається з трьох основних блоків: вхід, система, вихід. Для того щоб успішно моделювати роботу системи необхідно знати два з цих блоків. **Для проведення моделювання можуть використовуватися різні методи:**

***Аналітичні методи*** – передбачають існування у дослідника деякої аналітичної моделі, яка дасть змогу зменшити затрати на дослідження.

***Чисельні методи*** – їх використання предбачається у тих випадках, коли є неможливим застосування аналітичних моделей. Передбачають застосуваня статистичних методів.

***Якісні методи*** – з допомогою цих методів можна оцінювати асимтотичні значення величин, які визначаються в процесі моделювання і дозволяють оцінити поведінку системи в цілому.

Найбільш ефективним із них є аналітичний метод, тому йго можна розглянути більш детально.

**Аналітичний метод моделювання** полягає у визначенні формул, які з певною точністю описують роботу системи. Для визначення необхідніх нам даних про систему відбувається підстановка значень аргументу під кожен з випадків. Даний метод може застосуватись до відносно нескладних систем. Але більшість моделей,які описують реальні системи, є занадто складними і тому за допомогою цього методу можна тільки попередньо оцінити різноманітні варіанти, попередньо значно ідеалізувавши систему.

**Існує декілька видів моделювання.** Серед них за різними показниками можна виділити наступні:

В залежності від характеру досліджуваних процесів у системі усі види моделювання можуть бути розділені на детерміновані і стохастичні, статичні і динамічні, дискретні, безперервні і дискретно-безперервні.

***Детерміноване*** моделювання відображає детерміновані процеси, тобто процеси, у яких передбачається відсутність усяких випадкових впливів;

***Стохастичне*** моделювання відображає ймовірністні процеси і події. У цьому випадку аналізується ряд реалізацій випадкового процесу й оцінюються середні характеристики, тобто набір однорідних реалізацій;

***Статичне*** моделювання служить для опису поводження об'єкта в який-небудь момент часу;

***Динамічне*** моделювання відбиває поводження об'єкта в часі;

***Дискретне*** моделювання служить для опису процесів, що передбачаються дискретними, відповідно *безперервне* моделювання дозволяє відбити безупинні процеси в системах, а *дискретно*- *безперервне* моделювання використовується для випадків, коли хочуть виділити наявність як дискретних, так і безупинних процесів.

**У залежності від форми представлення об'єкта можна виділити *уявне* і *реальн*е моделювання:**

***Уявне*** моделювання часто є єдиним способом моделювання об'єктів, які або практично не можливо реалізувати в заданому інтервалі часу, або які існують поза умовами, можливими для їхнього фізичного створення. Наприклад, на базі уявного моделювання можуть бути проаналізовані багато ситуацій мікросвіту, що не піддаються фізичному експерименту.

При ***реальному*** моделюванні на базі представлень людини про реальні об'єкти створюються різні наочні моделі, що відображають явища і процеси, що протікають в об'єкті, використовується можливість дослідження різних характеристик або на реальному об'єкті цілком, або на його частині. Такі дослідження можуть проводитися як в об'єктах, що працюють у нормальних режимах, так і при організації спеціальних режимів для оцінки характеристик, які цікавлять дослідника

**Моделі можна розділити на:**

1. ***аналогові*** – ґрунтується на застосуванні аналогій різних рівнів.
2. ***зображувальні*** можуть бути у вигляді тексту, який описує систему,блок-схем, графів,таблиць.Зображувальні моделі модуть використовуватись цу фазі попереднього аналізу або дослідження.
3. ***математичні(аналітичні)*** – формалізовуються у вигляді математичних виразів опису системи:

1). Оператори переходів і виходів;

2). Цільова функція;

3). Обмеження, що виконує опис параметрів стану системи;

4). Опис входів-виходів системи.

1. імітаційні – алгоритм опису системи у вигляді програм для ЕОМ;
2. діалогові – це людинно-машинна система, яка дозволяє дослідникові впроцесі моделювання, проведення експеременту, вносити певні значення в систему в режимі взаємодії з ЕОМ.

Адаптивні моделі передбачають присутність механізму адаптації(настройки), що використовується для надходження нової інформації в деякий момент керування.

В загальному випадку для побудови моделі використовують фізико-математичний аналог явищ,експеремент-ідентифікацію.

**Основні етапи технології моделювання :**

* постановка мети моделювання;
* розробка концептуальної моделі;
* підготовка вихідних даних;
* розробка математичної моделі;
* вибір методів моделювання;
* вибір засобів моделювання;
* перевірка адекватності і корегування моделі;
* планування експериментів з моделлю;
* аналіз результатів моделювання.

Крім того можна зустріти ще і такі етапи моделювання, як розробка програмної моделі (цей етап слідує за етапом вибір засобів моделювання ); моделювання на обчислювальній системі “комп’ютері” ( він слідує після етапу планування ).

**2.Системи масового обслуговування**

За останній час в самих різних областях практики виникла необхідність в рішенні різних задач вірогідності, пов’язаних з роботою так званих *систем масового обслуговування* (**СМО**). Прикладами таких систем можуть служити: телефонні станції, ремонтні майстерні, квиткові каси, стоянки таксі, перукарні і т.п.

Теорія масового обслуговування спирається на теорію вірогідності і математичну статистику.

На первинний розвиток теорії масового обслуговування зробили особливий вплив роботи Датського ученого А.К. Эрланга (1878-1929).

**Теорія масового обслуговування** – область прикладної математики, що займається аналізом процесів в системах виробництва, обслуговування, управління, в яких однорідні події повторюються багато разів, наприклад, на підприємствах побутового обслуговування; в системах прийому, переробки і передачі інформації; автоматичних лініях виробництва і ін.

**Предметом** теорії масового обслуговування є встановлення залежностей між характером потоку заявок, числом каналів обслуговування, продуктивністю окремого каналу і ефективним обслуговуванням з метою знаходження якнайкращих шляхів управління цими процесами.

**Задача теорії масового обслуговування** – встановити залежність результуючих показників роботи системи масового обслуговування (вірогідність того, що заявка буде обслужена; математичного очікування числа обслужених заявок і т.д.) від вхідних показників (кількості каналів в системі, параметрів вхідного потоку заявок і т.д.). Результуючими показниками або характеристиками СМО, що цікавлять нас, є – показники ефективності СМО, які описують чи здатна дана система справлятися з потоком заявок.

Задачі теорії масового обслуговування носять вимізацій ний характер і зрештою включають економічний аспект за визначенням такого варіанту системи, при якому буде забезпечений мінімум сумарних витрат від очікування обслуговування, втрат часу і ресурсів на обслуговування і простоїв каналів обслуговування.

Система обслуговування вважається заданою, якщо відомі:

1) потік вимог, його характер;

2) безліч обслуговуючих приладів;

3) дисципліна обслуговування (сукупність правил, задаючих процес обслуговування).

Кожна СМО складається з якогось числа обслуговуючих одиниць, які називаються каналами обслуговування. Як канали можуть фігурувати: лінії зв’язку, різні прилади, особи, що виконують ті або інші операції і т.п

Всяка СМО призначена для обслуговування якогось потоку заявок, що поступають в якісь випадкові моменти часу. Обслуговування заявок продовжується якийсь випадковий час, після чого канал звільняється і готовий до прийому наступної заявки. Випадковий характер потоку заявок і часів обслуговування призводить до того, що в якісь періоди часу на вході СМО накопичується надмірно велике число заявок (вони або стають в чергу, або покидають СМО не обслуженими); в інші ж періоди СМО працюватиме з недовантаженням або взагалі простоюватиме.

Процес роботи СМО є випадковим процесом з дискретними поляганнями і безперервним часом; полягання СМО міняється стрибком в моменти появи якихось подій ( або приходу нової заявки, або закінчення обслуговування, або моменту, коли заявка, якій набриднуло чекати, покидає чергу ).

## Перелік характеристик систем масового обслуговування можна представити таким чином:

* середній час обслуговування;
* середній час очікування в черзі;
* середній час перебування в СМО;
* середня довжина черги;
* середнє число заявок в СМО;
* кількість каналів обслуговування;
* інтенсивність вхідного потоку заявок;
* інтенсивність обслуговування;
* інтенсивність навантаження;
* коефіцієнт навантаження;
* відносна пропускна спроможність;
* абсолютна пропускна спроможність;
* частка часу простою СМО;
* частка обслужених заявок;
* частка втрачених заявок;
* середнє число зайнятих каналів;
* середнє число вільних каналів;
* коефіцієнт завантаження каналів;
* середній час простою каналів.

СМО поділяють на різні групи в залежності від складу і від часу перебування в черзі до початку обслуговування, і від дисципліни обслуговування заявок.

**По складу СМО бувають:**

***одноканальні*** *–* характеризуються одним вхідним потоком,одним обслуговуючим пристроєм;

***багатоканальн****і –* з великим числом обслуговуючих пристроїв. Багатоканальні системи можуть складатися з обслуговуючих пристроїв як однакової, так і різної продуктивності.

**За часом перебування вимог** у черзі до початку обслуговування системи поділяються на три групи:

1) з необмеженим часом очікування – чергова заявка (вимога), заставши всі пристрої зайнятими, стає в чергу й очікує обслуговування доти , поки один із пристроїв не звільниться;

2) з відмовами – вимога, яка надійшла, заставши всі пристрої зайнятими, залишає систему;

3) змішаного типу – вимога, що надійшла, заставши всі пристрої зайнятими, стає у чергу й очікує обслуговування протягом обмеженого часу. Не дочекавшись обслуговування у встановлений час, заявка залишає систему.

**3.Імітаційне моделювання**

Дослідження поведінки СМО, тобто дослідження залежностей фазових змінних від часу при подачі на входи будь-яких, необхідних відповідно до завдання, потоків заявок, називають ***імітаційним моделюванням* СМО**. Імітаційне моделювання проводиться шляхом відтворення подій, які відбуваються одночасно або послідовно в модельному часі. При цьому *подією* є факт зміни значення будь-якої фазової змінної.

Для імітаційного моделювання СМО необхідні імітаційна модель, мова для її представлення і програмна система, що реалізує цю мову.

Імітаційні моделі СМО називаються *мережними імітаційними моделями* (МІМ). Використовують декілька підходів до побудови МІМ, що відрізняються вибором структури моделі. У структурі МІМ виділяються її елементарні частини (моделі елементів), такими елементами можуть бути або події, або засоби обслуговування, або пункти маршрутів заявок. Відповідно розрізняють моделі і мови їх опису, орієнтовані на події, пристрої або процеси. Тому вибір мови моделювання визначає структуру моделі і методику її побудови.

У структурі МІМ, яка орієнтована на процеси, використовуються моделі джерел вхідних потоків заявок, пристроїв, накопичувачів і вузлів.

***Джерело вхідного потоку заявок*** у моделі являє собою алгоритм, по якому обчислюються моменти появи заявок на вході. Джерела можуть бути залежними і незалежними. У залежних джерелах моменти появи заявок залежать від настання визначених подій, наприклад, від приходу іншої заявки на вхід деякого пристрою. Типовою моделлю незалежного джерела є алгоритм генерації значень випадкової величини з заданим законом розподілу.

***Пристрої*** в імітаційній моделі представляються алгоритмами генерації значень інтервалів (тривалостей) обслуговування. Частіше всього це алгоритми генерації значень випадкових величин із заданим законом розподілу. Крім того, модель пристрою відображає задану дисципліну обслуговування, оскільки в модель входить алгоритм, який управляє чергами на входах пристрою.

***Накопичувачі*** моделюються алгоритмами визначення обсягів пам’яті, які займаються заявками, що приходять на вхід накопичувача. Звичайно, обсяг пам’яті, що займається заявкою, обчислюється як значення випадкової величини, закон і (або) числові характеристики розподілу залежать від типу заявки.

***Вузли***виконують сполучні, керуючі і допоміжні функції в імітаційній моделі, наприклад, для вибору напрямків руху заявок у МІМ, зміни їхніх параметрів і пріоритету, поділу заявок на частини, об’єднання і т.п.

Звичайно кожному типу елементарної моделі, за винятком лише деяких вузлів, у програмній системі відповідає визначена процедура (підпрограма). Тоді МІМ можна представити як алгоритм, котрий складається з впорядкованих звертань до цих процедур, відображаючих поведінку системи.

В процесі моделювання відбуваються зміни модельного часу, котрий частіше всього приймається дискретним, і вимірюється в тактах. Час змінюється після того, як закінчена імітація чергової групи подій, що відносяться до поточного моменту часу tk. Імітація супроводжується накопиченням в окремому файлі статистики даних: кількості заявок, які вийшли із системи опрацьованими і неопрацьованими, сумарного часу зайнятості для кожного пристрою, середньої довжини черг і т.п. Імітація закінчується коли поточний час перевищить заданий відрізок часу або коли вхідні джерела опрацюють задане число заявок. Після цього проводиться опрацювання накопичених даних, у результаті одержуємо значення необхідних вихідних параметрів.

У програмах імітаційного моделювання СМО переважно реалізується *подійний метод* організації обчислень. Суть подійного методу полягає в відслідковуванні на моделі послідовності подій у тому ж порядку, у якому вони відбувалися б у реальній системі. Обчислення проводяться тільки для тих моментів часу і тих частин (процедур) моделі, до яких відносяться події. Даний метод може істотно прискорити моделювання в порівнянні з інкрементним методом, у якому на кожному такті аналізуються стани всіх елементів моделі.

Розглянемо можливу схему реалізації подійного методу імітаційного моделювання.

Моделювання починається з перегляду операторів генерації заявок, тобто з звертання до моделей джерел вхідних потоків. Для кожного незалежного джерела таке звертання дозволяє розрахувати момент генерації першої заявки. Цей момент разом із іменем –

Інтерпретація

СПП

СМП

Зміна стану пристрою В: його заняття

Заявкою з черги

Заявка А і її місце вСМО рух по мережі

Прогноз нових подій

вказівкою на заявку – заноситься в список майбутніх подій (СМП), а відомості про заявку, що генерується – у список заявок (СЗ). Запис у СЗ містить у собі ро’я заявки, значення її параметрів (атрибутів), місце, що займається в даний момент у МІМ. У СМП події впорядковуються по збільшенню моментів настання.

Далі зі СМП вибирається сукупність відомостей про події, що відносяться до найбільш раннього моменту часу. Ця сукупність переноситься в список поточних подій (СПП), із якого беруться вказівки на події. Звертання по вказівці до СЗ дозволяє встановити місце в МІМ заявки А, з якою пов’язана подія, що моделюється. Нехай цим місцем є пристрій В. Далі програма моделювання виконує такі дії:

1) змінює параметри стану пристрою В; наприклад, якщо заявка А звільняє В, а черга до В не була порожня, то відповідно до заданої дисципліни обслуговування з черги до В вибирається заявка С и надходить на обслуговування в В;

2) прогнозується час настання наступної події, пов’язаної з заявкою С, шляхом звертання до моделі пристрою В, у якій розраховується тривалість обслуговування заявки С; відомості про цю майбутню подію заносяться в СМП і СЗ;

3) відбувається імітація руху заявки А в МІМ по маршруту, визначеному заданою програмою моделювання, доти, поки заявка не прийде на вхід деякого ОА; тут або заявка затримується в черзі, або шляхом звертання до моделі цього ОА прогнозується настання деякої майбутньої події, пов’язаної з наступним шляхом заявки А; відомості про цю майбутню подію також заносяться в СМП і СЗ;

4) у файл статистики добавляються згадані вище дані.

Після відпрацювання всіх подій, що відносяться до моменту часу tk, відбувається збільшення модельного часу до значення, що відповідає найближчій майбутній події, і розглянутий процес імітації повторюється.

Імітаційне моделювання базується на відтворені на ЕОМ розгорнутого в часі процесу функціонування системи з обов’язковим врахуванням її взаємодії із зовнішнім середовищем. Для створення імітаційної моделі необхідно провести:

1. Побудову моделі досліджуваної системи на основі моделей підсистем, які об’єднані одним з видів взаємодій.
2. Вибір інформативних характеристик об’єкта, способів їх одержання та методів їх аналізу.
3. Побудову моделі впливу зовнішнього середовища на систему у вигляді імітаційних моделей зовнішніх факторів.
4. Вибір способу дослідження імітаційної моделі відповідно до методів планування імітаційного експерименту.

Умовно імітаційну модель представляють у вигляді діючих програмно чи апаратно реалізованих блоків:

БОР

БМО

БКІМ

БІЗВ

*Структура імітаційної моделі*

• Блок імітації зовнішніх впливів (БІЗВ) – формує реалізації випадкових чи детермінованих процесів, які імітують впливи зовнішнього середовища на об’єкти.

• Блок опрацювання результатів (БОР) – дає змогу одержати інформативні характеристики об’єкта. Для цього він використовує інформацію, що надходить від блоку математичної моделі об’єкта.

• Блок математизації об’єкта (БМО).

• Блок керування імітаційною моделлю (БКІМ) – реалізовує способи дослідження моделі та автоматизовує процес проведення імітаційного експерименту.

Метою імітаційного моделювання є конструювання або побудова імітаційної моделі та проведення імітаційного експерименту з цією моделлю для вивчення функціювання та поведінки досліджуваної системи обо об’єкта.

До переваг імітаційного моделювання відносяться:

1) можливість проведення експериментів над моделями системи, для яких натурні експерименти неможливі з етичних та небезпечних для життя причин.

2) вирішення задач, до яких не застосовні аналітичні методи за причини їх складності або відсутності.

3) можливість аналізу загальносистемних функцій для складних систем.

4) скорочення термінів для прийняття проектних рішень та проведення оцінки їх ефективності.

5) можливість проведення аналізу варіантів структури великих систем та їх поведінки під дією тих чи інших впливів.

**4.Побудова структурної моделі**

Структурна модель показує, з яких компонентів складається система і які зв’язки існують між ними:



**5.Побудова функціональної моделі**



**6.Побудова концептуальної моделі**

***Концептуальною моделлю*** об’єкта називають сукупність характеристик , властивостей та особливостей функціонування об’єкта і припущення щодо ступеня впливу на вихідні потоки . Це абстрактна модель, що визначає склад і структуру системи, властивості елементів та причинно-наслідкові зв’язки, притаманні досліджуваній системі і необхідні для досягнення цілі моделювання.

Основними поняттями концептуальної моделі є: дані, дані користувача, керуючі дані, пакет, вузол комутації, канал зв’язку і мережа передачі даних.

Дані – це факти описані у формалізованому вигляді. Виділяють дані користувача(вводяться ни сисему) і керуючі дані(служать для керування роботи системи).

Мережа передачі даних – сукупність пристроїв, що служить для передачі даних.

Вузол комутації – виконує функцію комутації даних. Їх перерозподілу.

Канал зв’язку – сукупність засобів, що забезпечують доставку даних в необхідне місце призначення.

Пакет – дані, що мають заголовок і обмежену довжину.

Концептуальна модель є основою для розробки математичної моделі. Створенням математичної моделі досягається дві мети:

– для однозначного розуміння процесу функціонування системи дається її формалізований опис;

– дає можливість представити опис системи у такому вигляді, який дозволяє проведеня аналітичного дослідженя.



Концептуальна модель зображена вище складається з вхідного потоку заявок, черги, та обслуговуючого присрою. Заявки першого типу надходять із середнім інтервалом 1.5 хв. , а другого -- 3 хв. Відхилення генерується підпрограмою генератором за допомогою експотенційного закону розподілу. В залежності від типу заявки вона обслуговується три або дві хвилини. Процес моделювання триває доти, доки система не опрацює 1000 заявок.

**7.Математичне та програмне забезпечення**

При виконанні данного завдання були використані формули теорії ймовірності. Це формули для отримання псевдовипадкових чисел.

Всі необхідні величини генеруються за експотенційним законом розподілу. Розглянемо закони що використовуються в даній роботі.

**Генератор випадкових чисел(рівномірний закон розподілу)**

Найбільш поширений алгоритм для одержання псевдо випадкових чисел був запропонований Д. Лемером.

У алгоритмі Лемера беруться два цілих числа: множник  і модуль .

Послідовність випадкових чисел обчислюється в такий спосіб: число , відоме з попереднього кроку, обчислюється добуток , число  ділиться на . Одержуємо ціле число  і цілочисельний залишок , що можна представити у вигляді:

 (2)

так як  - число між 0 і , то потрібно його ще розділити на , щоб одержати число між 0 і 1:

 (3)

Співвідношення (2) записується звичайно у вигляді

 (4)

**Експоненційний закон розподілу.**

Щільність розподілу:

; ; ; (2 )

 (3)

де λ - параметр розподілу, .

Потрібно знайти формулу для моделювання випадкової величини *X* за допомогою рівномірно розподіленої випадкової величини .

Знаходимо обернену стосовно *F* функцію. Маємо :

 (4)

Так, як 1-R має той же розподіл, що і *R*, то зручніше при знаходженні значень випадкової величини *X* користуватися формулою :

 (5)

Випадкове число з експонентним розподілом обчислюється по формулі



Розглянемо, як використовуючи метод оберненої функції, можна змоделювати випадкову величину, розподілену за експоненційним законом. Нехай λ=1. Виконаємо апроксимацію експоненційного розподілу лінійними ділянками, щоб можна було використовувати її для моделювання методом оберненої функції. Для апроксимації достатньо 24-х точок.

У разі необхідності моделювання випадкової величини Х, роходіленої за експоненційним розподілом з λ≠1, яка використовується як затримка у часі (наприклад, для моделювання пуасонівського потоку надходження заявок), виконується наступним чином:

1) генерується значення випадкової величини, розподіленої за експоненційним розподілом з λ=1;

2) знаходиться добуток цього значення та математичне сподівання випадкової величини Х; у результаті отримаємо шукану послідовність значень шуканої величини Х.

**8.Блок-схеми**

**Загальна блок-схема програми.**

Початок

Ініціалізація та запуск потоків лієнтів

Ініціалізація та запуск потоку обслуговування

Виконання основного потоку

Закриття усіх відкритих потоків та дискрипторів

Кінець

**Блок-схема потоку обслуговування.**

Початок

Кількість опрацьованих заявок = 0

Ініціалізація локальних даних та дискрипторів

1

1

НІ

Кількість опрацьованих заявок = 0

ТАК

Генерація часу опрацювання заявки

Опрацювання заявки

Запис результатів у файл

Збільшення лічильника опрацьованих заявок на одиницю

Закритя локальних дискрипторів

Кінець

**Блок-схема гнератора випадкових чисел.**

X0:=p0

X:=a\*X0

Q:=trunc(x/m)

p:=x-q\*m;

Rand:=p/m

Генератор випадкових чисел з експоненційним законом розподілу(в програмі використовується з попередньо згенерованим рівномірно розподіленим випадковим числом) .

exponent:=(-1/λ)\*ln(r);

Де: r – генерується на попередньому генераторі.

**9.Технічні характеристики**

1. Intel – сумісний процесор з тактовою частотою >50 Мгц

2. IBM – сумісний монітор

3. IBM-сумісний відео адаптер

4. Операційна система Windows 95 і вище.

5. Пакет розробки програмного забезпечення Microsoft Visual Studio 6.0

6. 64 Мb RAM

7.IBM-сумісна клавіатура

**10. Результати моделювання**

**Дослід 1.**

Середня довжина черги клієнтів типу 1: 954

Середня довжина черги клієнтів типу 2: 513

Загальна середня довжина черги: 733

Середній час очікування заявок типу 1: 8,475 сек.

Середній час очікування заявок типу 2: -

Загальний середній час очікування: -

**Дослід 2.**

Середня довжина черги клієнтів типу 1: 952

Середня довжина черги клієнтів типу 2: 509

Загальна середня довжина черги: 730

Середній час очікування заявок типу 1: 8,615 сек.

Середній час очікування заявок типу 2: -

Загальний середній час очікування: -

**Дослід 3.**

Середня довжина черги клієнтів типу 1: 954

Середня довжина черги клієнтів типу 2: 504

Загальна середня довжина черги: 729

Середній час очікування заявок типу 1: 8,987 сек.

Середній час очікування заявок типу 2: -

Загальний середній час очікування: -

**Дослід 4.**

Середня довжина черги клієнтів типу 1: 957

Середня довжина черги клієнтів типу 2: 510

Загальна середня довжина черги: 733

Середній час очікування заявок типу 1: 8,991 сек.

Середній час очікування заявок типу 2: -

Загальний середній час очікування: -

**Дослід 5.**

Середня довжина черги клієнтів типу 1: 958

Середня довжина черги клієнтів типу 2: 509

Загальна середня довжина черги: 733

Середній час очікування заявок типу 1: 9,049 сек.

Середній час очікування заявок типу 2: -

Загальний середній час очікування: -

**11.Висновки**

Після проведення експериментів можна зробити такі висновки:

Система працює стабільно, але не ефективно. Дуже швидко зростає довжина черги, це пов”язане із тим, що система не справляється із поступаючим потоком заявок. Існує також проблема того, що клієнти типу 2 взагалі не обслуговуються. Ймовірність простою системи рівна нулю, оскільли заявки надходять на багато швидше ніж система може їх опрацювати.

Отже, можна дати такі рекомендації щодо покращення системи:

1. Зменшення часу обслуговування (особливо клієнтів типу 1).
2. Збільшення періоду між надходженням заявок (особливо клієнтів типу 1).
3. Введення в систему додаткового обслуговуючого пристрою.

Особисто на мій погляд, найкращим є третій варйант. Він дає можливість розвантажити систему, але без зменшення її продуктивності.

**Список використаної літератури**

1. Саати Т. Л. Элементы теории обслуживания. М.: Советское радио , 1971.
2. Гнеденко Б. В., Коваленко И. Н. Введение в теорию массового обслуживания , 2-е изд. М.:Наука , 1987.
3. Советов Б.Я., Яковлев С.А. “Моделирование систем” Учебник для ВУЗов- М.: Высшая школа , 1985.
4. Рябов В.Ф.,Советов Б.Я.Яковлев С.А. Машинное моделирование при проектировании больших систем .Л., 1978
5. Сочнев А.В.,Яковлев С.А. Методические указания к выполнению курсовой работы по дисциплине “Моделирование систем” Л.,1986.
6. Якимов И.М.,Мосунов В.Е.,Яхина З.Т. Имитационное моделирование сложных систем. Казань,1984.
7. Ивахненко А.Г.,Юрачковский Ю.П. Моделирование сложных систем по экспериментальным данным.М., 1987.
8. Имитационное моделирование в задачах синтеза структуры сложных систем /А.Д.Цвиркун и др.М.,1985**.**

**Додаток 1.**

**Лістинг програми на мові С++**

Файл functions.h

#include <math.h>

#define a 16807

#define m 2147483647;

#define LAMBDAi 2;

double p = (int)GetTickCount();

//генератор нормально розподілених випадкових чисел

double Rand(double p0)

{

double t, x, q;

x = a \* p0;

t = x / m;

q = (int)t;

p = x - q \* m;

t = p / m

return t;

}

// генератор випадкових чисел, розподілених по експоненційному закону

double Exponent(UINT lambda)

{

double r;

r = Rand(p);

return ( (-1 / lambda) \* log(r) );

}

//return -1000 ... +1000

int Generator()

{

double d1;

int res;

static bool b = TRUE;

d1 = Exponent(900000);

res = (int)d1;

if(res<0) res \*= -1;

while(res>1000)

{

res = (int)res / 2;

}

if(b) res \*= -1;

b = !b;

return res;

}

//////////////////////////////////////////////////////////////////////////////

void IntToStr(char \*lpResultBuff, int iDigit)

{

// PARAMETRS:

// lpResult -- buffer, wich will get string

// iDigit -- integer digit, wich will translated to string

wsprintf(lpResultBuff, "%d", iDigit);

return;

}

#define MESSAGE(szText) MessageBox(0, szText, "Kursova", 0);

Файл types.h

typedef struct \_CLIENT

{

UINT uIn;

UINT uOut;

UINT uNumber;

} CLIENT;

Файл Kursova.cpp

#include "stdafx.h"

#include <shlwapi.h>

#include <fstream.h>

#include "resource.h"

#include "functions.h"

#include "types.h"

////////// all times in miliseconds

#define CLIENT1\_SERVTIME 3000

#define CLIENT2\_SERVTIME 2000

#define CLIENT1\_COMETIME 1500

#define CLIENT2\_COMETIME 3000

#define TIME\_RATIO 1 //1 - 1min == 1sec

//0.5 - 1min == 0.5sec

// prototipes ///////////////////////////////////////////////////////////////////////

LRESULT CALLBACK WindowProc(HWND,UINT,WPARAM,LPARAM);

DWORD WINAPI ServerThread(HWND);

DWORD WINAPI Client1Thread(HWND);

DWORD WINAPI Client2Thread(HWND);

VOID CALLBACK TimerProc(HWND, UINT, UINT, DWORD);

void Serving(CLIENT\*, UINT, UINT);

/////////////////////////////////////////////////////////////////////////////////////

// global variables /////////////////////////////////////////////////////////////////

char szResults[150];

UINT uWorkingTime = 0;

HANDLE hServerThread, hClient1Thread, hClient2Thread;

DWORD dwServerID, dwTread1ID, dwTread2ID;

HWND hMainWindow;

HANDLE hSemaphore;

UINT uTimerID;

//controls

UINT uTurn1 = 0;

UINT uTurn2 = 0;

UINT uTurn = 0;

//turns

bool bCMOWork = TRUE;

CLIENT Turn1[5000] = {0, 0, 0};

UINT uAmount1 = 0; //кількість

CLIENT Turn2[5000] = {0, 0, 0};

UINT uAmount2 = 0;

// code /////////////////////////////////////////////////////////////////////////////

int APIENTRY WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)

{

//create semaphore

hSemaphore = CreateSemaphore(NULL, 1, 1, "JustASemaphore");

if(hSemaphore==NULL)

{

MessageBox(NULL, "Failed to create semaphore.", "Kursova", 0);

return 0;

}

DialogBoxParam(hInstance, (LPSTR)IDD\_DIALOG1, NULL, (DLGPROC)WindowProc, 0);

return 0;

}

//\*\*\*\*\*

LRESULT CALLBACK WindowProc(HWND hWnd,UINT message,WPARAM wParam,LPARAM lParam)

{

switch(message)

{

case WM\_COMMAND:

if(LOWORD(wParam) == IDC\_QUIT)

SendMessage(hWnd, WM\_CLOSE, 0, 0);

case WM\_INITDIALOG:

//create clients generators

hClient1Thread = CreateThread(0, 0, (LPTHREAD\_START\_ROUTINE)Client1Thread,

hWnd, 0, &dwTread1ID);

hClient2Thread = CreateThread(0, 0, (LPTHREAD\_START\_ROUTINE)Client2Thread,

hWnd, 0, &dwTread2ID);

if(hClient1Thread==NULL || hClient2Thread==NULL)

{

MessageBox(NULL, "Can't create client threads", "Kursova", 0);

PostMessage(hWnd, WM\_CLOSE, 0, 0);

}

//create server thread

hServerThread = CreateThread(0, 0, (LPTHREAD\_START\_ROUTINE)ServerThread,

hWnd, 0, &dwServerID);

if(hServerThread == NULL)

{

MessageBox(NULL, "Can't create server thread", "Kursova", 0);

PostMessage(hWnd, WM\_CLOSE, 0, 0);

}

//create timer

uTimerID = SetTimer(0, 0, 1000, TimerProc);

if(uTimerID==NULL)

{

MessageBox(NULL, "Can't create timer", "Kursova", 0);

PostMessage(hWnd, WM\_CLOSE, 0, 0);

}

hMainWindow = hWnd;

break;

case WM\_CLOSE:

TerminateThread(hServerThread, 0);

TerminateThread(hClient1Thread, 0);

TerminateThread(hClient2Thread, 0);

CloseHandle(hSemaphore);

KillTimer(0, uTimerID);

EndDialog(hWnd, 0);

break;

}

return FALSE;

}

/// main thread, wich is serv clients ///////////////////////////////////////////////

DWORD WINAPI ServerThread(HWND hWnd)

{

UINT uServing1 = 0;

UINT uServing2 = 0;

UINT uCount;

bool bWait;

CLIENT Client;

UINT ClientType;

UINT uServingTime;

char cBuff[100]; //!!!!!!!! 100 --> 10

UINT TurnLength;

ofstream File;

File.open("Results.txt");

for(uCount=0; uCount<=1000; uCount++)

{

do

{

WaitForSingleObject(hSemaphore, INFINITE);

if(uServing1 >= uAmount1)

{

if(uServing2 >= uAmount2) bWait = TRUE;

else

{

Client = Turn2[uServing2];

ClientType = 2;

uServing2++;

bWait = FALSE;

}

}

else

{

Client = Turn1[uServing1];

ClientType = 1;

uServing1++;

bWait = FALSE;

}

ReleaseSemaphore(hSemaphore, 1, NULL);

Sleep(0);

}

while(bWait==TRUE);

//update main window

if(ClientType == 1)

{

uTurn1--;

IntToStr(cBuff, uTurn1);

SendDlgItemMessage(hWnd, IDC\_TURN1, WM\_SETTEXT, 0, (long)cBuff);

}

else

{

uTurn2--;

IntToStr(cBuff, uTurn2);

SendDlgItemMessage(hWnd, IDC\_TURN2, WM\_SETTEXT, 0, (long)cBuff);

}

uTurn--;

IntToStr(cBuff, uTurn);

SendDlgItemMessage(hWnd, IDC\_TURN, WM\_SETTEXT, 0, (long)cBuff);

//serving client

//generate serving time and sleep for this time

uServingTime = (unsigned)(Generator() \* TIME\_RATIO);

if(ClientType==1) uServingTime += (unsigned)(CLIENT1\_SERVTIME \* TIME\_RATIO);

else uServingTime += (unsigned)(CLIENT2\_SERVTIME \* TIME\_RATIO);

Serving(&Client, ClientType, uServingTime);

//save results in Results.txt

File << "Клієнт номер: " << uCount << endl;

File << " тип: " << ClientType << endl;

File << " порядковий номер у групі: " << uCount << endl;

File << " поступив у чергу: " << Client.uIn << endl;

File << " обслугований: " << Client.uOut << endl;

File << " час роботи системи: " << uWorkingTime << endl;

File << " черга клієнтів 1 : " << uAmount1 << endl;

File << " черга клієнтів 2 : " << uAmount2 << endl;

//update main window

if(ClientType == 1)

{

IntToStr(cBuff, uServing1);

SendDlgItemMessage(hWnd, IDC\_SERVED1, WM\_SETTEXT, 0, (long)cBuff);

}

else

{

IntToStr(cBuff, uServing2);

SendDlgItemMessage(hWnd, IDC\_SERVED2, WM\_SETTEXT, 0, (long)cBuff);

}

IntToStr(cBuff, uServing1+uServing2);

SendDlgItemMessage(hWnd, IDC\_SERVED, WM\_SETTEXT, 0, (long)cBuff);

} // while(TRUE) Sleep(10000);

SendMessage(hWnd, WM\_CLOSE, 0, 0);

File.close();

return 0;

}

/////////////////////////////////////////////////////////////////////////////////////

/// client treads realization ///////////////////////////////////////////////////////

DWORD WINAPI Client1Thread(HWND hWnd)

{

UINT uCount;

UINT uTime;

char cBuff[10];

UINT TurnLength;

for(uCount=1; uCount<=5000; uCount++)

{

//generate to next client uTime --- in miliseconds

//uTime = CLIENT1\_COMETIME + Generator();

uTime = (unsigned)(CLIENT1\_COMETIME \* TIME\_RATIO);

uTime += (unsigned)(Generator() \* TIME\_RATIO);

Sleep(uTime);

//is CMO working

if(bCMOWork==FALSE) return 0;

//add client1 to the first turn

WaitForSingleObject(hSemaphore, INFINITE);

Turn1[uCount].uNumber = uCount;

Turn1[uCount].uIn = GetTickCount();

Turn1[uCount].uOut = 0;

uAmount1++;

//update main window

uTurn1++;

IntToStr(cBuff, uTurn1);

SendDlgItemMessage(hWnd, IDC\_TURN1, WM\_SETTEXT, 0, (long)cBuff);

uTurn++;

IntToStr(cBuff, uTurn);

SendDlgItemMessage(hWnd, IDC\_TURN, WM\_SETTEXT, 0, (long)cBuff);

ReleaseSemaphore(hSemaphore, 1, NULL);

}

return 0;

}

DWORD WINAPI Client2Thread(HWND hWnd)

{

UINT uCount;

UINT uTime;

char cBuff[10];

UINT TurnLength;

for(uCount=1; uCount<=5000; uCount++)

{

//generate to next client uTime --- in miliseconds

//uTime = CLIENT2\_COMETIME + Generator();

uTime = (unsigned)(CLIENT2\_COMETIME \* TIME\_RATIO);

uTime += (unsigned)(Generator() \* TIME\_RATIO);

Sleep(uTime);

//is CMO working

if(bCMOWork==FALSE) return 0;

//add client2 to the first turn

WaitForSingleObject(hSemaphore, INFINITE);

Turn2[uCount].uNumber = uCount;

Turn2[uCount].uIn = GetTickCount();

uAmount2++;

//update main window

uTurn2++;

IntToStr(cBuff, uTurn2);

SendDlgItemMessage(hWnd, IDC\_TURN2, WM\_SETTEXT, 0, (long)cBuff);

uTurn++;

IntToStr(cBuff, uTurn);

SendDlgItemMessage(hWnd, IDC\_TURN, WM\_SETTEXT, 0, (long)cBuff);

ReleaseSemaphore(hSemaphore, 1, NULL);

}

return 0;

}

/////////////////////////////////////////////////////////////////////////////////////

////////// timer procedure //////////////////////////////////////////////////////////

void Serving(CLIENT \*lpClient, UINT uClientType, UINT uServingTime)

{

char cBuff[10];

UINT uQuitTime;

UINT uStart;

//calculate quit time

uStart = GetTickCount();

uQuitTime = GetTickCount() + uServingTime;

//update main window

IntToStr(cBuff, uClientType);

SendDlgItemMessage(hMainWindow, IDC\_CLIENTTYPE, WM\_SETTEXT, 0, (long)cBuff);

do

{

IntToStr(cBuff, GetTickCount() - uStart);

SendDlgItemMessage(hMainWindow, IDC\_PASSED, WM\_SETTEXT, 0, (long)cBuff);

}while(uQuitTime >= GetTickCount());

lpClient->uOut = GetTickCount();

return;

}

/////////////////////////////////////////////////////////////////////////////////////

////////// timer procedure //////////////////////////////////////////////////////////

VOID CALLBACK TimerProc(HWND hwnd, UINT uMsg, UINT idEvent, DWORD dwTime)

{

char cBuff[10];

uWorkingTime++;

IntToStr(cBuff, uWorkingTime);

SendDlgItemMessage(hMainWindow, IDC\_WORKINGTIME, WM\_SETTEXT, 0, (long)cBuff);

return;

}