Державний вищий навчальний заклад
“Українська академія банківської справи Національного банку України”

ІМІТАЦІЙНА МОДЕЛЬ ОЦІНКИ ЕФЕКТИВНОСТІ РОБОТИ БАНКОМАТУ В ЗАЛЕЖНОСТІ ВІД ЙОГО РОЗМІЩЕННЯ

Конкурсна робота Всеукраїнського конкурсу студентських наукових робіт з природничих, технічних та гуманітарних наук у 2009/2010 навчальному році за напрямком «Кількісні методи в економіці (статистика, економіко-математичне моделювання)»

Автор — студент групи МЕК-51 спеціальності 8.050102 «Економічна кібернетика» Якименко О.В.

Науковий керівник — доцент кафедри економічної кібернетики Хайлук С.О.

2010

ЗМІСТ

ВСТУП

Сучасні банки розглядають розширення мережі банкоматів, як один з пріоритетних напрямів діяльності. Тому власне постає проблема ефективного розміщення банкоматів із оптимальними параметрами, що дозволить зекономити готівку і покращити обслуговування клієнтів, а отже і їх довіру до банку. Тож розробка імітаційної моделі оцінки ефективності роботи банкомату в залежності від розміщення дозволить більш ефективно працювати на ринку банківських послуг.

Метою даної роботи є побудова імітаційної моделі розміщення банкомату та оцінки ефективності його роботи на певній території.

Для досягнення цієї мети слід виконати такі завдання: дослідження особливостей принципів діяльності банкоматів і сфери застосування; проведення порівняльного аналізу існуючих методів оцінки ефективності роботи банкомату в залежності від його розміщення; обґрунтування доцільності використання імітаційного моделювання для оцінки ефективності роботи банкомату в залежності від місця його розташування; побудова імітаційної моделі оцінки ефективності роботи банкомата у залежності від його розташування; оцінка ефективності використання запропонованої моделі; розробка прототипу автоматизованої системи оцінки ефективності банкомату в залежності від його розміщення; проведення експериментів з моделлю та інтерпретація результатів.

Область дослідження даної роботи – теорія і практика сучасних банківських технологій та статистичне моделювання. Об’єктом дослідження є характеристики роботи банкомату в різних районах розміщення. Предмет дослідження – імітаційна модель оцінки ефективності роботи банкомату як системи масового обслуговування.

1 ТЕОРЕТИКО-МЕТОДИЧНІ ОСНОВИ ІМІТАЦІЙНОГО МОДЕЛЮВАННЯ СИСТЕМИ МАСОВОГО ОБСЛУГОВУВАННЯ

1.1 Загальні особливості процесу імітаційного моделювання

З розвитком обчислювальної техніки дедалі ширшого розвитку та використання набувають імітаційні моделі.

Імітаційна модель — це комплексна математична й алгоритмічна модель досліджуваної системи.

Імітаційні моделі можуть класифікуватись наступним чином: статична чи динамічна; детермінована чи стохастична; неперервна чи дискретна.

Імітаційне моделювання в широкому розумінні – це процес побудови моделі реальної системи та експериментування на цій моделі з метою визначити поведінки чи оцінити її реакцію на зміну певних факторів.

У вужчому розумінні імітаційне моделювання — це відтворення на комп’ютері реальної системи.

Розглянемо переваги та недоліки імітаційного моделювання (рисунок 1.1).

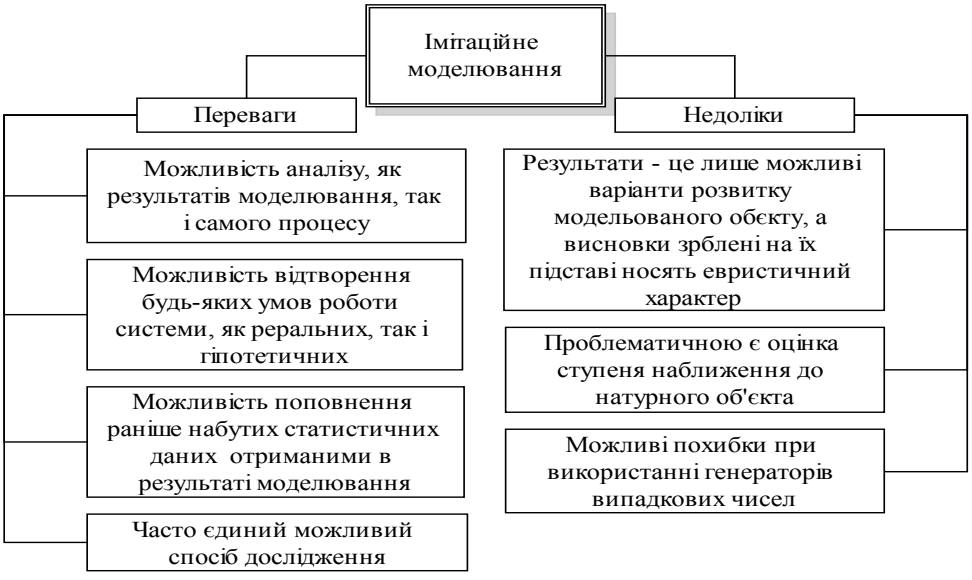

Рисунок 1.1 – Переваги та недоліки імітаційного моделювання

Можна виділити наступні етапи імітаційного моделювання: аналіз характеристик і закономірностей функціонування досліджуваного об'єкта; побудова імітаційної моделі; підготовка даних для моделі; власне програмна реалізація імітаційної моделі; оцінка адекватності моделі; проведення імітаційних експериментів.

Для вирішення проблеми перевірки адекватності моделі реальній системі часто користуються верифікацією та валідацією та акредитацію.

1.2 Об'єкт дослідження як система масового обслуговування

Розглянемо банкомат, як систему масового обслуговування.

Процес функціонування банкомату полягає у тому, що періодично до нього підходять клієнти і проводять певні операції. Клієнт, який підійшов до терміналу, що в цей момент зайнятий, може покинути його або стати в чергу.

Ефективність роботи банкомату залежить від розміру банку і його виду та місця розташування. Основним фактором буде клієнтська база банку, тобто кількість клієнтів, які обслуговуються у ньому. Але на цей показник ми впливати не можемо.

Наступним чинником, який можна регулювати, є розміщення банкомату. Саме цей фактор визначає скільки клієнтів надійде до терміналу кожної години і взагалі за день чи місяць. Тож вираженням розміщення банкомату оберемо потік клієнтів до банкомату.

Отже, маємо одноканальну систему масового обслуговування з обмеженою чергою (m+1), що характеризується інтенсивністю обслуговування μ. Вхідний потік, який характеризує місце розташування, представимо, як стаціонарний пуасонівський, що має наступні характеристики: кількість клієнтів, сума операції, час обслуговування, накопичена сума за певний час.

Час обслуговування – випадкова величина з відомим законом розподілу ймовірностей. Обслуговування клієнтів припиняється, коли у банкоматі закінчуються гроші, коли йде перезавантаження. У тому разі, якщо клієнт надійшов до банкомату, але за якихось причин його заявка залишилася не обслуженою, він покидає систему, як не обслужений запит.

1.3 Обґрунтування доцільності застосування методів імітаційного моделювання для оцінки роботи банкомату

Щодо оцінки і підвищення ефективності роботи банкомату чи його управління, то на ринку інформаційних технологій не так багато розробок для даної сфери. Одним з них «Агент автоматизації і управління систем обслуговування» (ASOMIS)[5]. Головна проблема, яку допомагає вирішити система – обробка величезного об’єму інформації, що отримується з банкоматів. Система містить модуль прогнозування та оптимізації, який може реагувати на зміни середовища і сам автоматично «навчається». Але цей проект, ще не був реалізований на практиці, і до того ж досить дорогий і складний.

Існують і деякі інші комерційні проекти, основою яких є різні статистичні методи [2]. Так тренди використовується у прогнозуванні об’єму готівки у банкоматі. Багатомірний аналіз статистичної інформації використовується, наприклад, для порівняння ефективності роботи різних банкоматів. Багатофакторний аналіз дозволяє визначити найбільш впливові і реально діючи чинники, що впливають на діяльність АТМ. Кластерний аналіз допомагає у встановленні лімітів на операції у банкоматі. Так багато статистичних методів використовуються банками для аналізу, управління і оцінки ефективності банкоматів.

Діяльність банкомату гарно описується теорією масового обслуговування. Подібні системи, внаслідок імовірнісного характеру, найзручніше досліджувати користуючись методами імітаційного моделювання. Порівняльний аналіз методів представлено у таблиці 1.1

Таблиця 1.1 – Порівняльний аналіз наявних на ринку рішень

	Вартість	Легкість впровадження і використання	Необхідний рівень знань	Ступінь охоплення проблем
Агент автоматизації і управління систем обслуговування	-	-	+/-	+
Статистичні методи	+	+	-	-
Імітаційне моделювання	+	+	+/-	+

Як бачимо імітаційне моделювання якнайкраще підходить для моделювання роботи банкомату в залежності від розміщення.

1.4 Параметри об’єкта які можуть бути досліджені засобами імітаційного моделювання

Модель, що розробляється у даній роботі, носить імовірнісний характер, оскільки має певні невизначені параметри. Серед яких виділимо: суму операції; тривалість обслуговування одного клієнта; кількість клієнтів; інтервали між надходженнями клієнтів. Оскільки такі фактори неможливо визначити аналітично, то доцільно застосувати машинну імітацію.

Розглянемо більш детально ці параметри.

Час обслуговування клієнта залежить від: складності операції; обізнаності клієнта щодо роботи банкомату; технічні проблеми з банкоматом; інші випадкові фактори.

Тож, маючи статистику із протоколів банкомату, можна визначити закон розподілу та статистичні характеристики часу обслуговування.

Сума операції є випадкова величина і залежить від: місця знаходження банкомату; контингенту клієнтів; економічних зрушень в країні; дня місяця і року; банку, який надає послуги; інші випадкові фактори.

Характеристикою ефективності розташування банкомату може служити потік клієнтів.

Основні чинники, які визначають потік клієнтів: місце розташування; година доби; розмір банку, який надає послуги.

1.5 Постановка задачі моделювання

Одним із методів збільшення прибутку є економія. Економія досягається тоді, коли наявні ресурси використовуються найбільш оптимально. На даний момент кожен банк має власні або аутсорсингові банкомати і інші термінали. Але лише деякі з них є прибутковими, інші – планово-збитковими. Вони встановлюються лише з метою збільшення кількості клієнтів і підвищення рейтингу банку.

У такому разі для досягнення ефекту економії потрібно розглянути задачу ефективного використання коштів при обслуговуванні банкоматів. Визначимо вузькі місця у цій сфері: омертвіння капіталу; не обслужені клієнти; вартість обслуговування банкомату; висока інтенсивність потоку клієнтів; розташування банкомату.

Завдання, які необхідно виконати для забезпечення ефективного функціонування банкомату:

– виконання повного комплексу робіт по обслуговуванню клієнтів карткових платіжних систем;

– забезпечення безперервного функціонування мережі банкоматів (постійний аналіз можливого попиту на грошові засоби, сервісне та технічне обслуговування і т. д.);

– своєчасне завантаження банкоматів;

– розвиток і розширення мережі клієнтів карткових платіжних систем, як приватних, так і корпоративних.

Нехай маємо банкомат встановлений у певному місці, що характеризується інтенсивністю потоку клієнтів. Відомі закони розподілу суми операції та її тривалості (із попереднього-проведеного статистичного дослідження). Необхідно побудувати модель оцінки ефективності банкомату в залежності від його розміщення, імітуючи потік клієнтів, суму та тривалість операції. Потрібно визначити кількість не обслужених клієнтів за заданих умов, а також причини відмови, визначити коефіцієнт ефективності роботи банкомату у цьому районі, а також проміжні статистики розраховані в процесі моделювання.

2 РЕАЛІЗАЦІЯ ІМІТАЦІЙНОЇ МОДЕЛІ

2.1 Загальні вимоги до моделі, вхідні та вихідні данні

Досліджувана модель призначена для оцінки ефективності роботи банкомату, залежно від місця розташування. Потенційними користувачами її будуть працівники банків, відповідальні за діяльність парку терміналів.

Контрольованими вхідними змінними таким чином є:

- сума завантаження банкомату Z_0;
- критична сума грошей в банкоматі Z_k, при досягненні якої подається сигнал про необхідність завантаження банкомату;
- частки певних видів купюр;
- середня тривалість операції по обслуговуванню і завантаженню банкомата $\bar{t}_б$;
- мінімальний інтервал часу з моменту подання сигналу про необхідність завантаження банкомату до початку операції завантаження $\bar{t}_з$;
- тривалість циклу, що моделюється, у потоці клієнтів (число робочих днів у тижні) N_c;
- число реалізацій циклу, що моделюється (чим більша кількість реалізацій тим точніші будуть результативні показники) N_r.

Основними змінними, якими буде маніпулювати користувач: початкова сума завантаження, критична сума та розподіл купюр.

Неконтрольованими змінними даної моделі є:

- середня сума операції $\overline{G}$, тобто сума, яку запитує клієнт у середньому;
- коефіцієнт варіації суми G K_G;
- середня тривалість операції по обслуговуванню клієнта $\bar{\tau}$;
- коефіцієнт варіації тривалості операції K_τ;

- кількість клієнтів за добами циклу, що моделюється N_j;
- параметри розподілу клієнтів за часом доби.

Усі ці параметри отримуються за дослідними даними, тобто шляхом збирання статистики.

Характеристики грошової суми операції були одержані в результаті статистичного аналізу вибірки з протоколів роботи банкомату за місяць у пакеті Statistica. При цьому якнайкращу згоду з дослідними даними дав гамма-розподіл (рисунок 2.1).

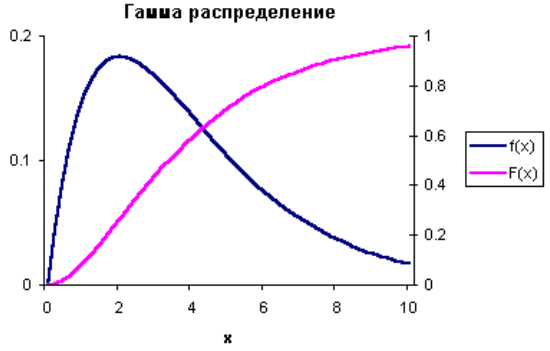

Рисунок 2.1 – Функція та щільність гамма-розподілу

Коефіцієнт варіації суми, яку запитує клієнт розраховується за наступною формулою:

$$K_G = \frac{\sigma_G}{\overline{G}}, \qquad (2.1)$$

де $\overline{G}$ – середнє значення суми операції;

σ_G – квадратичне відхилення.

Аналогічно розраховуються і коефіцієнти варіації інших вхідних параметрів.

При визначенні закону розподілу тривалості реалізації операції обслуговування клієнта в банкоматі дослідні дані згладжувалися з

використанням різних законів розподілу. Найкращі результати показав логнормальний розподіл (рисунок 2.2).

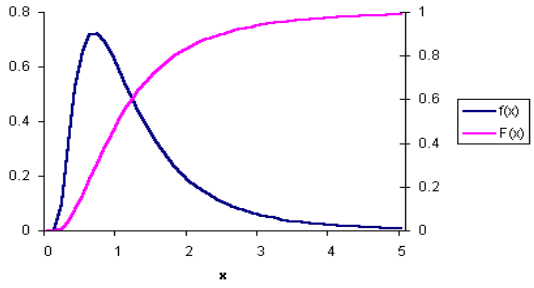

Рисунок 2.2 – Приклад функції та щільності логнормального розподілу

Також невизначеним залишається місце розташування банкомату. Як було зазначено вище його характеристикою приймемо потік клієнтів. Дослідимо стохастичні властивості потоку клієнтів. Гістограма розподілу клієнтів по дням тижня на протязі місяця зображена у додатках (рисунок Б.1).

На основі даних гістограм можна помітити значну нерівномірність розподілу клієнтів, наприклад різке їх збільшення на початку в кінці місяця може пояснюватись видачею заробітної плати саме в ці періоди.

Вихідними змінними моделі є:

- коефіцієнт простою через очікування клієнтів K_{H1};
- коефіцієнт простою через закінчення коштів у банкоматі K_{H2};
- коефіцієнт простою через обслуговування і завантаження банкомату K_{H3};
- коефіцієнт простою через недостатність дрібних купюр K_{H4};
- коефіцієнт використання банкомату (характеризує корисно використаний час роботи банкомату) *Ku*.

Проміжна статистика включає в себе середні залишки грошей під час перезавантаження, середня кількість обслужених і не обслужених клієнтів, середня кількість клієнтів в черзі і середній час перебування в черзі, середній

розрахований час обслуговування та сума операції, період завантаження банкомату, розрахована щільність вхідного потоку.

2.2 Математична модель оцінки ефективності роботи банкомату, залежно від місця розташування

Згідно статистики банкоматних операцій, тривалість обслуговування клієнтів τ досить добре описується логнормальним розподілом. Алгоритм моделювання випадкової величини з таким розподілом полягає у тому, що тривалість обслуговування кожного клієнта розраховується за формулою:

$$\tau = \exp\left[\rho_\tau + \alpha_\tau\left(\sum_{i-1}^{12}\gamma_i - 6\right)\right], \tag{2.2}$$

де α_τ, ρ_τ – параметри розподілу, які оцінюються за дослідними даними;

γ_i – 12 випадкових чисел, рівномірно розподілених у інтервалі 0...1.

У результаті виконання операції по обслуговуванню клієнт отримує суму грошей G, яка як випадкова величина добре описується гамма-розподілом з щільністю:

$$f(g) = \frac{1}{\rho_G \Gamma(\alpha_G)}\left(\frac{g}{\rho_G}\right)^{\alpha_{G-1}} \exp\left(-\frac{g}{\rho_G}\right), \tag{2.3}$$

де ρ_G, α_G – параметри розподілу, які також оцінюються за дослідними даними, ρ = 606,7775 грн., α = 0,8893.

Алгоритм розрахунку суми операції у додатках (рисунку Б.2).

Стан банкомату у довільний момент часу доби t характеризується наступними параметрами:

– інтервалом часу з поточного моменту до моменту приходу наступного клієнта $\Delta\tau$;

– інтервалом часу з поточного моменту до моменту закінчення операції по обслуговуванню наступного клієнта Δt;

– інтервалом часу з поточного моменту до моменту початку завантаження банкомату Δtz;

– інтервалом часу з поточного моменту при завантаженні банкомату до закінчення цієї операції Δt'z;

– індексом стану банкомату si.

Якщо запас грошей у банкоматі $Z > Z_k$, то $si = 0$. Якщо $Z \leq Z_k$, то $si = 1$. Якщо банкомат знаходиться у стані обслуговування і завантаження, то $si = 2$.

У процесі моделювання роботи банкомату збирається наступна статистика:

k – кількість клієнтів, що надійшли;

k_0 – кількість клієнтів, що було обслужено;

$\bar{k}$ – кількість клієнтів, які не отримали гроші через те, що сума, яка була запитана, менше залишку грошей у банкоматі ($G < Z$) або банкомат було заблоковано у зв'язку з його завантаженням і обслуговуванням;

k_{note} – кількість клієнтів, які не отримали грошей, через те, що у банкоматі не знайшлося дрібних купюр;

$S_{оч}$ – статистика для розрахунку середньої довжини черги;

$S_{ос}$ – статистика для розрахунку середньої величини залишку коштів при інкасації;

t_{H1} – накопичений час перебування банкомату у стані очікування клієнтів, тобто стані, коли $Оч = 0$;

t_{H2} – накопичений час перебування банкомату у стані очікування завантаження за відсутності грошей у ньому ($Z = 0$);

t_{H3} – накопичений час перебування банкомату у стані обслуговування і завантаження ($si = 2$);

N_Z – накопичена кількість завантажень банкомата;

S_z – статистика для розрахунку середнього запасу грошей в банкоматі.

Після закінчення моделювання за зібраною статистикою розраховуються підсумкові показники роботи банкомата як системи масового обслуговування.

Середня кількість клієнтів, що надійшли за день, розраховується за формулою:

$$\bar{k} = \frac{k}{N_{дн}}, \tag{2.5}$$

де $N_{дн}$ – число днів роботи банкомату в процесі моделювання, що розраховується за формулою:

$$N_{дн} = N_c N_r . \tag{2.6}$$

Середня кількість обслужених за добу клієнтів розраховується:

$$\bar{k} = \frac{k_0}{N_{дн}}. \tag{2.7}$$

Середнє число клієнтів, що отримали відмову в обслуговуванні складає:

$$\overline{k^-} = \frac{k^-}{N_{дн}}. \tag{2.8}$$

Середній період завантажень у днях:

$$\overline{T_z} = \frac{N_{дн}}{N_z}. \tag{2.9}$$

Середня сума залишку при інкасації банкомату:

$$\overline{Z_{ос}} = \frac{S_{ос}}{N_z}. \tag{2.10}$$

Середня довжина черги клієнтів до банкомату:

$$\overline{Оч} = \frac{S_{оч}}{t_ф}, \tag{2.11}$$

де $t_ф$ – тривалість інтервалу моделювання у годинах:

$$t_ф = N_{дн} 24. \tag{2.12}$$

Середній запас коштів у банкоматі складає:

$$\overline{Z} = \frac{S_z}{t_ф}. \tag{2.13}$$

Коефіцієнт простою банкомату через очікування клієнтів:

$$K_{Н1} = \frac{t_{Н1}}{t_ф}. \tag{2.14}$$

Коефіцієнт простою через закінчення коштів у банкоматі:

$$K_{Н2} = \frac{t_{Н2}}{t_ф}. \tag{2.15}$$

Коефіцієнт простою через обслуговування і завантаження банкомату:

$$K_{H3} = \frac{t_{H3}}{t_{\phi}}. \tag{2.16}$$

Коефіцієнт використання банкомату розраховується:

$$K_u = 1 - K_{H1} - K_{H2} - K_{H3}. \tag{2.17}$$

Модель передбачає оптимізацію розміру суми завантаження. В якості критерію оптимізації були використані наступні статті витрат:

- витрати на перезавантаження банкомату;
- втрати, пов’язані з омертвіння капіталу.

Формула оптимізації матиме вигляд:

$$\Theta = C_z \cdot \frac{\Phi}{\overline{T}_{\pi}} + p \cdot \overline{Z} \tag{2.18}$$

де C_Z – середні витрати на одне перезавантаження;

p –процент за кредит (вводиться користувачем, залежно від періоду експерименту);

Ф – кількість днів, що моделюється.

Представлений критерій враховує лише інтереси банка у зниженні витрат, якщо ж врахувати інтереси клієнтів, то доцільно зафіксувати прийнятний рівень якості обслуговування. Для цього візьмемо середню кількість клієнтів, які не були обслужені на протязі дня. Дня знаходження оптимальної суми завантаження потрібно мінімізувати витрати, задовольняючи умову $\bar{k} \le [\bar{k}]$, де $[\bar{k}]$ – допустиме число не обслужених клієнтів за день.

У додатках (рисунку Б.3) наведено блок-схема алгоритму статистичного моделювання функціонування банкомату.

У блоці 1 визначається інформація про вхідні дані.

У блоці 2 за раніше виведеними залежностями розраховуються параметри розподілу тривалості обслуговування ρ_τ, α_τ за , $\overline{\tau}$ і K_τ параметри розподілу суми операції ρ_G, α_G за $\overline{G}$ і K_G. Ці параметри потім використовуються для генерування випадкової тривалості операції τ і суми операції G.

У блоці 3 розраховуються накопичені значення частки клієнтів, що надходять, за годинами доби F_i. Якщо P_i – частки клієнтів за годинами доби (i=1...24), то:

$$F_i = \sum_{j-1}^{i} P_i \quad . \qquad 2.19$$

Отже $F_{24} = 1$.

У блоці 4 усі описані вище статистики прирівнюються нулю.

У блоці 5 дається початкове завантаження банкомату і визначається середня інтенсивність потоку клієнтів у перший день моделювання.

У блоці 6 Організується цикл за числом реалізацій N_r. Це число бажано брати якомога більшим.

У блоці 7 задаються початкові значення параметрів стану банкомату:

- черги $Oч = 0$;
- індексу стану $si = 0$;
- часу до закінчення операції $\Delta\tau$;
- часу до початку завантаження $\Delta t_z = \infty$.

Нескінченність означає, що дана подія не може відбутися у даній ситуації. Дійсно, якщо черга нульова, то і операція по обслуговуванню неможлива. Якщо $Z = Z_0$, що більше Z_k, то час до завантаження ще невизначено, що рівносильне заданню нескінченності цього часу.

У блоці 8 організується цикл за днями тижня.

У блоці 9 визначаються початкові значення поточного часу в межах доби, середня інтенсивність потоку клієнтів для відповідного дня тижня і годин доби.

У блоці 10 визначається час до приходу наступного клієнта.

У блоці 11 визначається час до найближчої події. При цьому відслідковуються наступні події: надходження наступного клієнта; закінчення операції по обслуговуванню клієнта; початок завантаження банкомату; закінчення завантаження банкомату.

У залежності від того, яка подія відбудеться раніше, відбувається розгалуження алгоритму.

У блоці А відбувається обробка ситуації у зв'язком з надходженням наступного клієнта. При цьому:

- число клієнтів, що надійшли k, збільшується на один;
- статистика $S_{оч}$ збільшується на $Оч\Delta\tau$;
- поточний час t збільшується на Δt.

Якщо у межах інтервалу Δt черга була рівна нулю, то статистика t_{Hl} збільшується на Δt, і клієнт одразу приймається на обслуговування, а час до закінчення його обслуговування $\Delta\tau$ визначається генеруванням випадкової довжини обслуговування, розподіленої за логнормальним законом. Якщо у черзі були клієнти, то черга збільшується на одиницю, а час до закінчення обслуговування клієнта $\Delta\tau$ зменшується на Δt.

Якщо сигнал про необхідність завантаження було подано ($si = 1$), то Δt_z також зменшується на Δt. Якщо клієнт надійшов при заблокованому банкоматі у зв'язку з його завантаженням ($si = 2$),то число k^- збільшується на одиницю, а t_{H3} збільшується на Δt. Після всіх відмічених перерахувань визначається новий інтервал часу до приходу наступного клієнта аналогічно блоку 10.

У блоці Б обробляється ситуація закінчення операції по обслуговуванню клієнта. При цьому: статистика Soч збільшується на Оч$\Delta\tau$;

черга зменшується на одиницю; інтервал Δt зменшується на Δτ; поточний час t збільшується на Δτ; розмір суми G, яку запитує клієнт, генерується у відповідності гама-розподілу.

Якщо виявляється, що G > Z, то обслуговування закінчується відмовою, статистика k^- збільшується на одиницю. Якщо у черзі є клієнти, то на обслуговування береться наступний клієнт, визначається час його обслуговування $\Delta\tau$, якщо клієнтів немає, то $\Delta\tau = \infty$. Якщо $G < Z$, то статистика k_o збільшується на одиницю, запас Z зменшується на G. Якщо клієнти є, то на обслуговування надходить наступний клієнт і визначається час до закінчення його обслуговування $\Delta\tau$, як і у попередній ситуації. Аналогічно, якщо клієнтів немає, то $\Delta\tau = \infty$. Якщо після видачі грошей запас Z стане менше або рівний Z_k, то подається сигнал про необхідність завантаження банкомату ($si = 1$) і визначається інтервал часу до завантаження Δt_z. При цьому, якщо момент завантаження приходиться на неробочий час служби, яка виконує завантаження, то цей час $\bar{t}_z$ збільшується таким чином, щоб це завантаження відбулася у робочий час, наприклад, на час з 7 до 19 години. Інтервал цього часу необхідно задавати у програмі. Якщо сигнал про необхідність завантаження було подано раніше, то Δt_z зменшується на $\Delta\tau$.

У блоці В обробляється ситуація настання моменту початку і закінчення завантаження банкомату. При цьому статистика $S_{оч}$ збільшується на $Оч\Delta t_z$, поточний час t збільшується на Δt_z, а Δt зменшується на Δt_z, $\Delta\tau = \infty$. Якщо $si = 1$, то відбувається інкасація банкомату, тобто вилучаються залишки і виконується завантаження і технічне обслуговування банкомату, а статистика t_{H2} збільшується на Δt_z. Час до закінчення цієї операції дорівнює $\Delta t'_z = \bar{t}_\delta$, індекс стану $si = 2$, статистика залишків $S_{ос}$ збільшується на Z, а поточний запас Z стає рівним нулю. Статистика k збільшується на число клієнтів у черзі, тобто на $Оч$, а якщо черга була нульовою, то статистика t_{H1} збільшується на Δt_z. Якщо $si = 2$ (завантаження закінчено), то статистика t_{H3}

збільшується на Δt_z, $Z = Z_0$, $\Delta t_z = \infty$, число завантажень N_z збільшується на одиницю.

У блоці 12 перевіряється настання кінця доби. Якщо він не настав, то виконується перехід до блоку 11 і визначається, яка подія буде наступною. Якщо кінець доби настав, то здійснюється перехід до наступної доби, при цьому поточний час t починає відраховуватися з початку наступної доби.

У блоці 16 розраховуються підсумкові показники роботи банкомата, а у блоці 17 виконується виведення результатів.

Алгоритм розподілу купюр зображений у додатках (рисунок Б.4).

У блоці 1 подаються вхідні дані:

– a – масив , що зберігає наступні значення 0, 200, 100, 50, 20, 10;

– A – масив, що містить інформацію про загальну кількість кожної купюри завантажену спочатку;

– B – масив, який містить кількість купюр кожного номіналу, які вже видані.

Також перший елемент масиву b прирівнюється до нуля.

У блоці 2 організовується цикл для кожної купюри різного номіналу.

У блоці 3 визначається кількість купюр одного номіналу, які потрібно видати клієнту і перевіряється чи не закінчились купюри певного номіналу.

У блоці 4, якщо купюри потрібного виду ще є, то вони видаються клієнту і кількість їх у банкоматі зменшується.

У блоці 5 клієнту видається та кількість певних купюр, яка залишилася у банкоматі.

На основі аналізу отриманих результатів можна, варіюючи параметри процесу обслуговування банкоматних систем і застосовуючи статистичне моделювання, здійснити оптимізацію цього процесу за обраними критеріями, що буде сприяти підвищенню ефективності функціонування банкоматів і підвищить якість банківських послуг.

3 ІМІТАЦІЙНЕ МОДЕЛЮВАННЯ РОБОТИ БАНКОМАТУ ТА АНАЛІЗ ЙОГО РЕЗУЛЬТАТІВ

3.1 Загальні відомості та кількісні характеристики натурного об’єкта

У якості натурного об’єкта візьмемо банкомат національної системи масових електронних платежів (НСМЕП), що знаходиться у власності Національного банку України. Термінал встановлений у холі центрального входу до другого корпуса Української академії банківської справи Національного банку України.

Було проведено статистичне дослідження діяльності банкомату протягом місяця (квітень 2009 року) і визначені стохастичні характеристики наступних параметрів: потік клієнтів; сума операції; тривалість обслуговування одного клієнта.

Встановлено, що у середньому в день банкомат обслуговує 677 клієнтів. Кількість клієнтів по дням тижня у додатках (таблиця Г.1).

Гістограма розподілу клієнтів по дням тижня зображено у додатках (рисунок Г.1).

Можемо помітити різке збільшення притоку клієнтів у п’ятницю та четвер, це пов’язано із закінченням робочого тижня і початком вихідних.

Частки клієнтів кожної години протягом доби зображено у додатках (рисунок Г.2).

На протязі доби були виявлені сплески активності, вони відбуваються в години обіду, а в інтервалі 17:00 до 19:00, саме в цей час люди йдуть з роботи.

Сума операції досліджуваного банкомату має гамма-розподіл з параметрами: $\rho = 606{,}7775$ і $\alpha = 0{,}8893$. Математичне очікування суми однієї операції 540 грн., а $K_G = 1.285$ Можемо відмітити дуже великий розкид значень суми операції: мінімальне значення 20 грн., а максимальне 5000.

Гістограма розподілу суми операції представлена у додатках (рисунок Г.3).

Тривалість обслуговування одного клієнта розподілена за логнормальний законом розподілу, математичне очікування тривалості операції 1,587, коваріація дорівнює 0,988.

3.2 Імітаційні дослідження ефективності роботи натурного об'єкта

Проаналізувавши результати моделювання (рисунок 3.5), можна зробити висновок, що загалом корисний час користування банкоматом досить непоганий – близько 77%. Тобто 77% робочого часу у банкоматі обслуговувались клієнти. А найбільші втрати часу були через очікування клієнтів – 17%., хоча з іншого боку найбільше не обслужених клієнтів саме через обмеження довжини черги, тобто спостерігається неоднорідність потоку клієнтів по годинам доби.

У середньому на день обслуговується 80 особи із 117 тих, що надійшли до банкомату, тобто близько 68%. Основна причина відмови у обслуговуванні – надто довгі черги. За час моделювання рідко виконується перезавантаження банкомату – не більше разу за весь період, і виконується досить швидко, тож нестача грошей у АТМ не суттєва проблема, як і простій під час сервісного обслуговування.

Внаслідок того, що середня сума операції становить 532 грн., то проблем з розподілом готівки не спостерігається.

Якщо розподілити накопичену кількість клієнтів у черзі на весь час моделювання, то маємо, що у середньому у черзі стояли 3-х клієнтів, а середній час перебування в черзі приблизно 3,054 хвилини.

Середній час обслуговування 1,588 хвилини.

Середній залишок у загальному випадку близько 240 гривень. Це означає, що загалом клієнтам вистачає коштів. У годину банкомат здатен

обслужити близько 3,317 клієнтів, тобто 68% від усього потоку, щільність якого близько 5 чоловік на годину.

Тож за даними умовами банкомат працює нормально. Тільки через нерівномірний розподіл клієнтів у різні періоди циклу моделювання (створюються великі черги і водночас 17% робочого часу банкомат чекає клієнтів) близько 35% клієнтів залишаються незадоволеними. Але це проблема організації всієї мережі.

Дослідимо банкомат, що розташований у приміському районі. Потік клієнтів не значний, отже простоювати банкомат буде довше. Коефіцієнт використання банкомату становить 0,45, що не є ефективним показником. Середня сума операції досить велика порядку 317 гривень, але загалом витрати на утримання терміналу будуть більші за дохід від нього.

Перевіривши реакцію моделі на зміну критичної суми, початкової суми, купюрної структури (додатки (таблиці Г.2 – Г.4)) можемо зробити певні висновки.

Можна помітити ріст кількості обслужених клієнтів із збільшенням критичної суми, і зменшення не обслужених. Це пояснюється тим, що під час достатньо невеликої критичної суми у банкоматі довгий час знаходиться невелика сума. Із ростом критичної суми зменшується період завантаження, оскільки частіше потрібно проводити сервісне обслуговування. Також зростає середній залишок у банкоматі, що є очевидним. Інші показники залишилися на тому ж рівні. Із ростом початкової суми завантаження зменшується кількість не обслужених клієнтів через недостатність грошей. Взагалі із зростанням початкової суми кількість обслугованих клієнтів зростає, і зменшується, відповідно, кількість не обслужених. Але такі умови збільшують мертвий капітал, тож потрібно обирати «золоту середину». Щодо інших показників, то як наслідок збільшення початкової суми збільшується період завантаження, пропускна здатність банкомату і інтенсивність обслуговування клієнтів. Проаналізувавши результати експериментів можна

помітити значне збільшення кількості не обслужених клієнтів внаслідок недостачі дрібних купюр.

Виконаємо порівняльний аналіз двох гіпотетичних банкоматів, які мають однакові параметри, але знаходяться у різних районах. Результати порівняльного аналізу представлено у додатках (таблиця Г.5).

3.3 Оцінка ефективності впровадження моделі та її адекватності натурному об’єкту

Визначимо мінімальну кількість реалізацій для оцінки середнього значення випадкових величин (суми операції, тривалості операції) з ймовірністю 95% (t_a=1.96, ε=0.05).

Припустимо час обслуговування має математичне очікування А та дисперсію σ^2.

$$\overline{t_p} = \frac{1}{N}\sum_{i=1}^{N} x_i \qquad (3.1)$$

Тоді $A = \overline{t_p}$ = 1.584., а σ = 1,496

У відповідність з центральною граничною теоремою при великих значеннях вибірки середнє арифметичне $\overline{t_p}$ буде нормально розподілене з математичним очікуванням A і дисперсію $\sigma^2/(N\text{-}1)$, тоді:

$$\varepsilon = t_a \cdot \frac{\sigma}{\sqrt{N-1}}. \qquad (3.2)$$

Звідси:

$$N = \frac{t_a^2 \cdot \sigma^2}{\varepsilon^2} + 1. \qquad (3.3)$$

Отже мінімальна кількість прогонів 3440.

Дослідимо вхідні дані на однорідність дисперсій.

Статистичні оцінки s_j^2 дисперсій $\sigma^2\{y_j\}$ для кожної j-ї спроби обчислюються за формулою

$$s_j^2 = \frac{1}{k-1}\sum_{s=1}^{k}\left(y_{js} - y_j\right)^2, \qquad (3.4)$$

де k — число повторень (дублювань) експерименту (це число далі беруть одне й те саме для всіх спроб); y_j — значення функції відгуку в j-й спробі.

Очевидно, що в результаті дії випадкових факторів при обчисленнях значень функції відгуку в кожній спробі не доводиться сподіватися на рівність оцінок дисперсій S^2_j. Тому перевірка на однорідність практично полягає в перевірці гіпотези щодо належності N вибіркових дисперсій S^2_j (j= 1, 2,..., N) до однієї генеральної сукупності. Оскільки $N > 2$, то для перевірки цієї гіпотези використовується критерій Кохрена.

Гіпотезу про однорідність вибіркових дисперсій за критерієм Кохрена перевіряють за такою схемою:

– Серед обчислених за формулою (3.4) оцінок дисперсій S^2_j знаходять найбільшу S^2_{jmax}.

– Обчислюють відношення найбільшої оцінки до суми оцінок усіх дисперсій

$$G = \frac{s_{j\max}^2}{\sum_{j=1}^{N} s_j^2}. \tag{3.5}$$

– Визначають число ступенів вільності f_1 і f_2:

$$\begin{aligned} f_1 &= k - 1, \\ f_2 &= N. \end{aligned} \tag{3.6}$$

– Обирають рівень значущості q (часто беруть q = 0,05).

– За даними q, $f1$ і $f2$ у спеціальній таблиці знаходять величину критичного відношення *Gкр*.

– Порівнюють величини *G* і *Gкр.*

Провели декілька експериментів постійно змінюючи критичну суму та суму завантаження. Перевіривши отримані оцінки дисперсій за допомогою критерію Кохрена отримали наступні результати:

$G_{кр}$=0,5157; G_G=0,2773; G_t=0,1549.

Оскільки усі відношення максимальної дисперсії до суми всіх дисперсій менше G_G , то можна можна говорити, що гіпотеза про однорідність дисперсій підтвердилася.

Провівши оптимізацію досліджуваного натурного банкомату (додатки (рисунок Г.4)), визначили його оптимальні параметри:

Витрати будуть становити 1801.2 гривні за період моделювання, а в початкових умовах (сума завантаження дорівнює 100000 грн.; критична сума дорівнює 1000 грн.) витрати становили 3795,6.Тож чиста економія складає 1994,4 грн. на період моделювання, тобто п’ять днів.

ВИСНОВКИ

У даній роботі були розглянуті теоретичні відомості, щодо принципів і механізму роботи банкомату, існуючи методи оцінки ефективності діяльності банкомату, їх недоліки та переваги, можливість використання банками України у сучасних умовах, розглянуті проблеми банків у цій сфері і їх можливі рішення. За основу була взята модель оцінки ефективності роботи банкомата запропонована Васіним М.С. Був розроблений програмний додаток у середовищі Borland Delphi 7, що автоматизує виконання оцінки на основі статистичних даних і декількох показників.

Використання статистичних методів для оцінки діяльності банкоматів і подальшої їх оптимізації дозволить більш точно і обґрунтовано створювати парк АТМ, організовувати їх роботу із максимальною ефективністю і мінімальними витратами. Гарно організована банкоматна мережа сприяє збільшенню репутації банка серед клієнтів, що призведе до збільшення останніх, а отже і доходів.

Можливі шляхи вдосконалення моделі:

- додати блок обробки статистичної інформації із протоколів банкоматної мережі, таким чином спростивши роботу з програмним додатком;
- додати блок оптимізації розподілу купюр різного номіналу;
- розробити методи, які потребуватимуть менше часового ресурсу для виконання оцінки ефективності роботи банкомату в залежності від його розміщення;
- пристосувати програмний додаток для оцінки ефективності роботи мережі банкоматів.

Отже, дана модель досить адекватно реагує на зміну факторів і дозволяє достатньо точно оцінити ефективність роботи банкомату і може використовуватися у сучасних умовах для підвищення ефективності банкоматної мережі банків.

ПЕРЕЛІК ПОСИЛАНЬ

1. Черемисинов Д., Черемисинова Л. Подход к програмированию агентов в мультиагентных системах [Електронний ресурс] – Режим доступу : http://www.foibg.com/ibs-04/IBS-04-p22.pdf

2. Авербух О. Качество работы банкоматной сети: как и чем его измерить [Електронний ресурс] – Режим доступу : //http://www.bitec.spb.ru/article/atm_quality.php

3. Франко А. Банокматы на аутсорсинг [Електронний ресурс] – Режим доступу : www.companion.ua/Articles/Content/?Id=20304&Callback=70

4. Васин М. С. Анализ и прогнозирование движения денег в банкоматных системах [Електронний ресурс] – Режим доступу : www.ostu.ru/upk/avtoreferat/vasin.doc

5. Буслович Д. Новаторское решение позволит снизить затраты на обслуживание банкоматов [Електронний ресурс] – Режим доступу : www.kurier.lt/?r=16&a=1209

6. Форум [Електронний ресурс] – Режим доступу : http://dom.bankir.ru/forumdisplay.php?s=3bb9b23f999a1042b02582c83759673e&f=142

7. [Електронний ресурс] – Режим доступу : http://www.plusworld.ru/daily/page1_3508.phphttp://www.kursiv.kz

8. Банкоматы. Историческая справка. [Електронний ресурс] – Режим доступу : //http://uabanks.info/content/bankomaty_istoricheskaya_spravka.html

9. Моделирование систем : практикум / Б. Я. Советов, С. А. Яковлев : М. : Высшая школа, 1999. – 230 с.

10. Імітаційне моделювання [Текст] : навчальний посібник / В. Ф. Ситник, Н. С. Орленко : Мін-во освіти України КНЕУ. – К. : КНЕУ, 1998. – 232 с.

11. Программирование в Delphi 7 [Текст] : справоное пособие / А.Я. Архангельський. – М. : БИНОМ, 2004. – 1024 с.

12. Имитационное моделирование. Классика CS. 3-е изд. [Текст] : учебное пособие / В. Кельтон, А. Лоу. – СПб.: Питер; Киев: Издательская группа BHV, 2004. – 847 с.

13. Имитационное моделирование в среде GPSS. [Текст] : учебное пособие / В. Томашевский., Е. Жданов. – М.: Бестселлер, 2003. – 416 с.

14. Імітаційне моделювання. [Текст] : навчально-методичний посібник / В.Ф. Ситник, Н.С. Орленко. – К.: КНЕУ, 1999. – 219 с.

15. Гамма-распределение. [Електронний ресурс] – Режим доступу: http://ru.wikipedia.org

16. Логнормальное распределение. . [Електронний ресурс] – Режим доступу: http://algolist.manual.ru/maths/matstat/lognormal/index.php

17. Прес-служба НБУ інформує. [Електронний ресурс] – Режим доступу: http://pr.bank.gov.ua/ukr/titul_news_detail.php

ДОДАТКИ

Додаток А – Код реалізації програми

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ExtCtrls, StdCtrls, Math, Grids, Unit2, Menus, ComCtrls, ComObj;

type
  PClient=^TClient;

  TWorld=class
    private
      clients:array of PClient;
      clients_count:integer;
      accumulate_time:int64;
      ro:real;
      alpha:real;
      eta:real;
      teta:real;
      procedure CreateClients;
      procedure StartTimers;
    public
      accumulate_sum_notes:array[0..5] of integer;
      inf_sum_notes: array [0..4] of integer;
      inf_notes: array[0..5] of integer;
      served_clients:integer;
      constructor Create;
      procedure Start;
      Calc.welcome_timer Calc.freq_timer frmMain.free_all_timer
      function LogNormal:real;
      function Gamma:real;
  end;

  TClient=class
    private
      id:integer;
      me:PClient;
      term_id:integer;
      wait_timer_was:cardinal;
      served:boolean;
      procedure GoToBank(Sender: TObject);
      procedure WaitTime(Sender: TObject);
      procedure FreeTerm(Sender: TObject);
    public
      cash_out: integer;
      count_notes: array[0..5] of integer;
      //operation_sum_notes:array of integer;
      be_in_queue:boolean;
      mu_time:cardinal;
      wait_time:cardinal;
      operation_sum:integer;
      wait_timer:TTimer;
      day:integer;
      hour:integer;
      constructor Create;
```

```
end;

TBank=class
  private
    terminals:array of boolean;
    term_money_amount:array of integer;
    procedure Recharge(Sender: TObject);
    procedure AfterRecharge(Sender: TObject);
  public
    //term_notes: array of real;
    recharge_timer:TTimer;
    critical_amount:integer;
    recharging:boolean;
    pre_charging:boolean;
    constructor Create;
end;

TQueue=class
  private
    Clients:array of PClient;
    array_length:integer;
  public
    clients_count:integer;
    procedure ReleaseClient(term_id:integer);
    procedure InsertClient(client:PClient);
    procedure StopQueue;
    procedure ReleaseQueue;
    constructor Create;
end;

TCalc=class
  private
    deadline:cardinal;
    step_calc:cardinal;
    avg_serve_time:real;
    avgOperSum:real;
    queue_time:array of cardinal;
    queue_time_length:cardinal;
    avg_queue_time:real;
    queue_quan:array of cardinal;
    queue_quan_length:integer;
    avg_queue_quan:real;
    procedure Step(Sender: TObject);
  public
    avgHourSum: real;
    freq_timer:TTimer;
    welcome_timer:TTimer;
    rest_recharge:array of cardinal;
    count_recharge:byte;
    avg_rest_recharge:real;
    n_serv_qtb:integer;
    n_serv_rech:integer;
    n_serv_nem:integer;
    n_serv_note:integer;
    n_serv_rq:integer;
    welcome_count:integer;
    empty_count:integer;
    //nonotes_count:integer;
    q,A:real;
    procedure AvgOperationSum;
    procedure AvgServeTime;
    procedure AvgQueueTime;
```

```
    procedure AvgQueueQuan;
    procedure RelAbsTraf;
    procedure WelcomeCheck(Sender: TObject);
    procedure AvgRestRecharge(Sender: TObject);
    constructor Create;
  end;

  TfrmMain = class(TForm)
    Button1: TButton;
    txtTerminals: TEdit;
    txtCountdown: TEdit;
    Label6: TLabel;
    txtCl_count: TEdit;
    Label7: TLabel;
    Label4: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    Label10: TLabel;
    Label13: TLabel;
    Label14: TLabel;
    Label15: TLabel;
    Label18: TLabel;
    Label23: TLabel;
    Label16: TLabel;
    Label17: TLabel;
    StringGrid1: TStringGrid;
    StringGrid2: TStringGrid;
    txtTermMoney: TEdit;
    txtCritical: TEdit;
    txtQueueLimit: TEdit;
    txtEta: TEdit;
    txtTeta: TEdit;
    txtRo: TEdit;
    txtAlpha: TEdit;
    Label2: TLabel;
    txtTime: TEdit;
    Button2: TButton;
    Button3: TButton;
    txtRealiseQuant: TEdit;
    Label1: TLabel;
    txtNumRealis: TEdit;
    Label3: TLabel;
    Bevel1: TBevel;
    Bevel3: TBevel;
    PageControl1: TPageControl;
    TabSheet1: TTabSheet;
    TabSheet2: TTabSheet;
    Bevel2: TBevel;
    TabSheet3: TTabSheet;
    Bevel4: TBevel;
    Label5: TLabel;
    Label11: TLabel;
    Label12: TLabel;
    Label19: TLabel;
    Label20: TLabel;
    txt200: TEdit;
    txt100: TEdit;
    txt50: TEdit;
    txt20: TEdit;
    txt10: TEdit;
    Label21: TLabel;
    txt200part: TEdit;
```

```
    txt100part: TEdit;
    txt50part: TEdit;
    txt20part: TEdit;
    txt10part: TEdit;
    Label22: TLabel;
    Label24: TLabel;
    Label25: TLabel;
    Button4: TButton;
    TabSheet4: TTabSheet;
    txtProcent: TEdit;
    txtCostRecharge: TEdit;
    Label26: TLabel;
    Label27: TLabel;
    txtPossibleNeobsl: TEdit;
    Label28: TLabel;
    txtMaxZ: TEdit;
    txtMinZ: TEdit;
    txtMaxCritical: TEdit;
    txtMinCritical: TEdit;
    Label29: TLabel;
    Label30: TLabel;
    Label31: TLabel;
    Label32: TLabel;
    Label33: TLabel;
    Label34: TLabel;
    Button5: TButton;
    procedure Optimisation; // (Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure FreeAll(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Button5Click(Sender: TObject);
  private
    { Private declarations }
  public
    tempCrit: real;
    count: integer;
    optim: boolean;
    TempBankMoney, TempBankCritical:real;
    lock: boolean;
    free_all_timer:TTimer;
    free_all_flag:boolean;
    global_iter_count:integer;
    welcome:array of real;
    empty:array of real;
    recharge:array of real;
    //notes: array of real;
    numR:integer;
    { Public declarations }
  end;

var
  frmMain: TfrmMain;
  World:TWorld;
  Bank:TBank;
  Queue:TQueue;
  Calc:TCalc;

implementation
```

```
{$R *.dfm}

constructor TWorld.Create;
begin
  inherited Create;
end;

function TWorld.LogNormal:real;
var sum1_12, logN: real;
    l: integer;
begin
  sum1_12:=0;
  for l:=1 to 12 do
    begin
      logN:=random;
      sum1_12:=sum1_12+logN;
    end;
  LogNormal:=exp(ln(sqr(ro)/sqrt(sqr(ro)+sqr(alpha)))+sqrt(ln(1+(sqr(alpha)/sqr(ro))))*(sum1_12-6));
end;

function TWorld.Gamma:real;
var sum1_eta, Rnd1, Rnd2, Rez, Rezult:real;
begin
  Rezult:=0;
  sum1_eta:=(exp(1)+(1/sqr(teta)))/exp(1);
  While (Rezult=0) do
    begin
    Rnd1:=random;
    if (sum1_eta*Rnd1 >= 1) then
    begin
      Rez:= -ln((sum1_eta-sum1_eta*Rnd1)/(1/sqr(teta)));
      Rnd2:=random;
      if (Rnd2<=power(Rez,(1/sqr(teta)-1))) then
        Rezult:= Rez;
    end
    else
    begin
      Rez:=power(sum1_eta*Rnd1,sqr(teta));
      Rnd2:=random;
      if Rnd2<=exp(-Rez) then
        Rezult:=Rez;
    end;
   end;
   Result:=eta*sqr(teta)*Rezult;
end;


procedure TWorld.Start;
begin
  Bank:=TBank.Create;
  Queue:=TQueue.Create;
  Calc:=TCalc.Create;
  ro:=strtofloat(frmMain.txtRo.Text);
  alpha:=strtofloat(frmMain.txtAlpha.Text);
  eta:=strtofloat(frmMain.txtEta.Text);
  teta:=strtofloat(frmMain.txtTeta.Text);
  accumulate_time:=0;
  clients_count:=0;
  served_clients:=0;
  randomize;
  CreateClients;
  Calc.welcome_timer.Enabled:=true;
```

```
  StartTimers;
  Calc.freq_timer.Enabled:=true;
  frmMain.free_all_timer.Enabled:=true;

end;

procedure TWorld.CreateClients;
var i,j,k,l,g:integer;
    intens_per_hour, random_gener,exp_gener,exp_gener_mu:real;
    exp_gener_int,exp_gener_int_mu:int64;
begin
  i:=0;
  l:=0;
  inf_notes[0]:=0;
  inf_notes[1]:=200;
  inf_notes[2]:=100;
  inf_notes[3]:=50;
  inf_notes[4]:=20;
  inf_notes[5]:=10;
  inf_sum_notes[0]:= strtoint(frmMain.txt200.text);
  inf_sum_notes[1]:= strtoint(frmMain.txt100.text);
  inf_sum_notes[2]:= strtoint(frmMain.txt50.text);
  inf_sum_notes[3]:= strtoint(frmMain.txt20.text);
  inf_sum_notes[4]:= strtoint(frmMain.txt10.text);
  g:=0;
  for k:=0 to 4 do
  begin
    for j:=0 to 23 do
    begin
      intens_per_hour:=strtofloat(frmMain.StringGrid1.Cells[0,k])*strtofloat(frmMain.StringGrid2.Cells[0,j]);
      l:=l+100;
      while accumulate_time<l do
      begin
        setlength(clients,i+1);
        new(clients[i]);
        clients[i]^:=TClient.Create;
        clients[i].me:=clients[i];
        clients[i].id:=i+1;
        clients[i].served:=false;
        random_gener:=random;
        if random_gener=0 then
          random_gener:=0.01;
        exp_gener:=(-1/intens_per_hour)*ln(random_gener);
        exp_gener:=exp_gener*100;
        exp_gener_mu:=LogNormal;
        exp_gener_mu:=exp_gener_mu*(100/60);
        exp_gener_int:=round(exp_gener);
        exp_gener_int_mu:=round(exp_gener_mu);
        if exp_gener_int_mu=0 then
          exp_gener_int_mu:=1;
        clients[i].wait_timer.Interval:=accumulate_time+exp_gener_int;
        clients[i].mu_time:=exp_gener_int_mu;
        clients[i].operation_sum:=(round(Gamma) div 10)*10;
        if clients[i].operation_sum > 5000 then
          clients[i].operation_sum:= 5000;
        clients[i].count_notes[0]:=0;
        clients[i].hour:=j+1;
        clients[i].day:=k+1;
        i:=i+1;
        clients_count:=clients_count+1;
        accumulate_time:=accumulate_time+exp_gener_int;
        frmResult.StringGrid3.Cells[0,g]:=floattostr(exp_gener_int_mu/(100/60));
```

```
      inc(g);
    end;
  end;
 end;
 frmMain.txtCl_count.Text:=inttostr(clients_count-1);
 setlength(Calc.queue_time,clients_count-1);
 setlength(Calc.queue_quan,clients_count-1);
end;

procedure TWorld.StartTimers;
var i:integer;
begin
 for i:=0 to clients_count-2 do
  clients[i].wait_timer.Enabled:=true;
end;

constructor TClient.Create;
begin
 inherited Create;
 wait_time:=0;
 wait_timer_was:=0;
 be_in_queue:=false;
 wait_timer:=TTimer.Create(nil);
 wait_timer.Enabled:=false;
 wait_timer.OnTimer:=GoToBank;
end;

procedure TClient.GoToBank(Sender: TObject);
var i,n,f:integer;
  temp, temp2:integer;
begin
 temp2:=operation_sum;
 if Bank.recharging=true then
 begin
  wait_timer.Enabled:=false;
  Calc.n_serv_rech:=Calc.n_serv_rech+1;
 end;
 if Queue.clients_count >= strtoint(frmMain.txtQueueLimit.Text) then
 begin
  wait_timer.Enabled:=false;
  Calc.n_serv_qtb:=Calc.n_serv_qtb+1;
  exit;
 end;
 if (Bank.term_money_amount[0]<>0)and(Bank.term_money_amount[0]<operation_sum) then
  operation_sum:=Bank.term_money_amount[0];
 if Bank.term_money_amount[0] = 0 then
 begin
  wait_timer.Enabled:=false;
  Calc.n_serv_nem:=Calc.n_serv_nem+1;
  exit;
 end;
 temp:=0;
 for n:=0 to 4 do
  begin
   temp:=temp + count_notes[n]*World.inf_notes[n];
   count_notes[n+1]:= (operation_sum - temp)div World.inf_notes[n+1];
   if (World.accumulate_sum_notes[n] + count_notes[n+1] <= World.inf_sum_notes[n]) then
    World.accumulate_sum_notes[n]:=World.accumulate_sum_notes[n]+ count_notes[n+1]
   else
    count_notes[n+1]:=World.inf_sum_notes[n]-World.accumulate_sum_notes[n];
  end;
 cash_out:= 0;
```

```
  for f:=0 to 4 do
  begin
    cash_out:= cash_out + (count_notes[f+1]* World.inf_notes[f+1]);
  end;
  if (cash_out < operation_sum)and(operation_sum<>0) then
    operation_sum:= cash_out;
    if (Bank.term_money_amount[0] > 0)and(cash_out = 0)and(temp2<>0) then
    begin
      wait_timer.Enabled:=false;
      Calc.n_serv_note:=Calc.n_serv_note+1;
    end;
    if (Queue.clients_count=0)and(be_in_queue=false) then
      for i:=0 to strtoint(frmMain.txtTerminals.Text)-1 do
      begin
        if Bank.terminals[i]=false then
        begin
          wait_timer.Enabled:=false;
          wait_timer_was:=wait_timer.Interval;
          term_id:=i;
          wait_timer.Interval:=mu_time;
          wait_timer.OnTimer:=FreeTerm;
      Bank.terminals[i]:=true;
      if (Bank.critical_amount >= Bank.term_money_amount[i]-operation_sum)and(Bank.pre_charging=false) then
      begin
        Bank.pre_charging:=true;
        Bank.recharge_timer.Enabled:=true;
      end;
      Bank.term_money_amount[i]:=Bank.term_money_amount[i]-operation_sum;
      wait_timer.Enabled:=true;
      exit;
     end;
    end;
  wait_timer.Enabled:=false;
  wait_timer_was:=wait_timer.Interval;
  wait_timer.Interval:=1;
  wait_timer.OnTimer:=WaitTime;
  Queue.InsertClient(me);
  wait_timer.Enabled:=true;
end;

procedure TClient.WaitTime(Sender: TObject);
begin
  wait_time:=wait_time+1;
end;

procedure TClient.FreeTerm(Sender: TObject);
begin
  wait_timer.Enabled:=false;
  Bank.terminals[term_id]:=false;
  if cash_out<>0 then
    served:=true;
  if Queue.clients_count<>0 then
    Queue.ReleaseClient(term_id);
  If cash_out<> 0 then
    inc(World.served_clients);
end;

constructor TBank.Create;
var i:integer;
begin
  inherited Create;
  setlength(terminals,strtoint(frmMain.txtTerminals.Text));
  setlength(term_money_amount,strtoint(frmMain.txtTerminals.Text));
```

```
   for i:=0 to length(terminals)-1 do
  begin
    terminals[i]:=false;
    term_money_amount[i]:=strtoint(frmMain.txtTermMoney.Text);
  end;
  recharge_timer:=TTimer.Create(nil);
  recharge_timer.Enabled:=false;
  recharge_timer.OnTimer:=Recharge;
  recharge_timer.Interval:=round(10*(100/60));
  critical_amount:=strtoint(frmMain.txtCritical.Text);
  recharging:=false;
  pre_charging:=false;
end;

procedure TBank.Recharge;
begin
  recharging:=true;
  recharge_timer.Enabled:=false;
  Queue.ReleaseQueue;
  recharge_timer.Interval:=round(20*(100/60));
  recharge_timer.OnTimer:=AfterRecharge;
  recharge_timer.Enabled:=true;
  setlength(Calc.rest_recharge,length(Calc.rest_recharge)+1);
  Calc.rest_recharge[length(Calc.rest_recharge)-1]:=term_money_amount[0];
  inc(Calc.count_recharge);
end;

procedure TBank.AfterRecharge;
var n:integer;
begin
  recharge_timer.Enabled:=false;
  recharge_timer.OnTimer:=Recharge;
  recharge_timer.Interval:=round(10*(100/60));
  term_money_amount[0]:=strtoint(frmMain.txtTermMoney.Text);
  for n:=0 to 4 do
  begin
    World.accumulate_sum_notes[n]:=0;
  end;
  recharging:=false;
  pre_charging:=false;
end;

constructor TQueue.Create;
begin
  inherited Create;
  clients_count:=0;
  array_length:=1;
end;

procedure TQueue.InsertClient(client:PClient);
begin
  clients_count:=clients_count+1;
  array_length:=array_length+1;
  setlength(clients,array_length);
  clients[array_length-1]:=client;
  client.be_in_queue:=true;
end;

procedure TQueue.ReleaseQueue;
var i:integer;
begin
  Calc.n_serv_rq:=Queue.clients_count;
```

```
  if Queue.clients_count>0 then
    for i:=array_length downto 1 do
      if clients[i-1]<>nil then
      begin
        clients[i-1].wait_timer.Enabled:=false;
        clients[i-1]:=nil;
      end;
  clients_count:=0;
end;

procedure TQueue.ReleaseClient(term_id:integer);
var  i, temp2:integer;
begin
  for i:=0 to length(clients)-1 do
    if clients[i]<>nil then
    begin
      clients[i].wait_timer.Enabled:=false;
      temp2:=clients[i].operation_sum;
      Calc.queue_time[Calc.queue_time_length]:=clients[i].wait_time;
      inc(Calc.queue_time_length);
      if (Bank.term_money_amount[0]<>0)and(Bank.term_money_amount[0]<clients[i].operation_sum) then
        clients[i].operation_sum:=Bank.term_money_amount[0];
      if (clients[i].cash_out < clients[i].operation_sum)and(clients[i].operation_sum<>0) then
        clients[i].operation_sum:=clients[i].cash_out;
      if Bank.term_money_amount[0] = 0 then
      begin
        clients[i].wait_timer.Enabled:=false;
        Calc.n_serv_nem:=Calc.n_serv_nem+1;
        frmResult.txtNServe_nem.Text:=inttostr(Calc.n_serv_nem);
        exit;
      end;
      if (Bank.term_money_amount[0] <> 0)and(clients[i].cash_out = 0)and(temp2<>0) then
      begin
        clients[i].wait_timer.Enabled:=false;
        Calc.n_serv_note:=Calc.n_serv_note+1;
      end;
      clients[i].wait_timer.Interval:=clients[i].mu_time;
      clients[i].wait_timer.OnTimer:=clients[i].FreeTerm;
      clients[i].term_id:=term_id;
      Bank.terminals[term_id]:=true;
      if (Bank.critical_amount >= Bank.term_money_amount[0]-
clients[i].operation_sum)and(Bank.pre_charging=false) then
      begin
        Bank.pre_charging:=true;
        Bank.recharge_timer.Enabled:=true;
      end;
      Bank.term_money_amount[term_id]:=Bank.term_money_amount[term_id]-clients[i].operation_sum;
      clients[i].wait_timer.Enabled:=true;
      clients[i]:=nil;
      clients_count:=clients_count-1;
      exit;
    end;
end;

procedure TQueue.StopQueue;
var i:integer;
begin
  for i:=0 to length(World.clients)-1 do
    world.clients[i].wait_timer.Enabled:=false;
end;

constructor TCalc.Create;
```

```
begin
  inherited Create;
  freq_timer:=TTimer.Create(nil);
  freq_timer.Enabled:=false;
  freq_timer.Interval:=100;
  freq_timer.OnTimer:=Step;
  deadline:=strtoint(frmMain.txtTime.Text)*100;
  step_calc:=0;
  avgHourSum:=0;
  queue_time_length:=0;
  queue_quan_length:=0;
  count_recharge:=0;
  n_serv_qtb:=0;
  n_serv_rech:=0;
  n_serv_nem:=0;
  n_serv_note:=0;
  n_serv_rq:=0;
  avgOperSum:=0;
  welcome_timer:=TTimer.Create(nil);
  welcome_timer.Enabled:=false;
  welcome_timer.Interval:=1;
  welcome_timer.OnTimer:=WelcomeCheck;
  welcome_count:=0;
  empty_count:=0;
end;

procedure TCalc.Step(Sender: TObject);
begin
  step_calc:=step_calc+100;
  frmMain.txtCountdown.Text:=floattostr(step_calc/100);
  queue_quan[queue_quan_length]:=Queue.clients_count;
  inc(queue_quan_length);
  avgHourSum:=avgHourSum + Bank.term_money_amount[0];
  if step_calc=deadline then
  begin
    freq_timer.Enabled:=false;
    Queue.StopQueue;
    welcome_timer.Enabled:=false;
    AvgServeTime;
    AvgQueueTime;
    AvgQueueQuan;
    AvgOperationSum;
    RelAbsTraf;
    AvgRestRecharge(Sender);
    frmMain.free_all_flag:=true;
  end;
end;

procedure TCalc.AvgServeTime;
var i:integer;
    accum_time:cardinal;
begin
  accum_time:=0;
  for i:=0 to World.clients_count-2 do
    accum_time:=accum_time+World.clients[i].mu_time;
  avg_serve_time:=accum_time/(World.clients_count-1);
end;

procedure TCalc.AvgQueueTime;
var i:integer;
    accum_time:cardinal;
begin
```

```
  accum_time:=0;
  for i:=0 to queue_time_length-1 do
    accum_time:=accum_time+queue_time[i];
  if queue_time_length<>0 then
    avg_queue_time:=accum_time/queue_time_length
  else
    avg_queue_time:=0;
end;

procedure TCalc.AvgQueueQuan;
var i:integer;
    accum_quan:cardinal;
begin
  accum_quan:=0;
  for i:=0 to World.clients_count-2 do
    if World.clients[i].be_in_queue=true then inc(accum_quan);
  avg_queue_quan:=accum_quan/(queue_quan_length);
  avg_queue_quan:=roundto(avg_queue_quan,-2);
end;

procedure TCalc.RelAbsTraf;
var p_otk:real;
begin
  p_otk:=(World.clients_count-World.served_clients)/strtoint(frmMain.txtCl_count.Text);
  q:=1-p_otk;
  A:=((World.clients_count-1)/strtoint(frmMain.txtTime.Text))*q;
  q:=roundto(q,-3);
  A:=roundto(A,-3);
end;

procedure TCalc.AvgOperationSum;
var i, n: integer;
    accum_operSum: real;
begin
  accum_operSum:=0;
  n:=0;
  for i:=0 to World.clients_count-2 do
    if World.clients[i] <> nil then
    begin
      accum_operSum:=accum_operSum+World.clients[i].operation_sum;
      inc(n);
    end;
  avgOperSum:=accum_operSum /n;
end;

procedure TCalc.WelcomeCheck(Sender: TObject);
begin
  if (Bank.recharging=false)and(Bank.term_money_amount[0]=0) then
  begin
    inc(empty_count);
    exit;
  end;
  if (Bank.terminals[0]=false) then
    inc(welcome_count);
end;

procedure TCalc.AvgRestRecharge;
var u, avg:cardinal;
begin
  avg:=0;
  if Calc.count_recharge<>0 then
  begin
```

```
    for u:=0 to Calc.count_recharge-1 do
      avg:=avg+Calc.rest_recharge[u];
    Calc.avg_rest_recharge:=avg/Calc.count_recharge;
  end
  else
    Calc.avg_rest_recharge:=0;
end;

procedure TfrmMain.Button1Click(Sender: TObject);
var r: integer;
    temp1, temp2, temp3, temp4, temp5, temp6, temp7, temp8, temp9, temp10, temp11, temp12, temp13, temp14,
temp15, temp16, temp17, temp18, temp19, tempK1, tempK2, tempK3:real;
begin
  if lock=true then
    begin
      if (TempBankMoney<=strtoint(txtMaxZ.Text))then
      begin
        if TempBankCritical>=strtoint(txtMaxCritical.Text) then
        begin
          TempBankMoney:=TempBankMoney+5000;
          frmMain.txtTermMoney.Text:=floattostr(TempBankMoney);
          tempCrit:=strtoint(txtMinCritical.Text);
          frmMain.txtCritical.Text:=floattostr(tempCrit);
        end;
        if TempBankCritical<strtoint(txtMaxCritical.Text) then
        begin
          tempCrit:=TempBankCritical+500;
          frmMain.txtCritical.Text:=floattostr(tempCrit);
        end;
        TempBankCritical:=tempCrit;
        if frmMain.global_iter_count>0 then
          dec(frmMain.global_iter_count);
        inc(count);
      end;
      if (TempBankMoney>strtoint(txtMaxZ.Text)) or (TempBankCritical>strtoint(txtMaxCritical.Text)) then
      begin
        lock:=false;
        frmMain.numR:=0;
        frmMain.global_iter_count:=0;
        showmessage('Моделювання закінчено!');
        Exit;
      end;
  end;

frmMain.txt200.Text:=floattostr(roundto((strtofloat(frmMain.txt200part.Text)*strtoint(frmMain.txtTermMoney.Tex
t)/200),0));

frmMain.txt100.Text:=floattostr(roundto((strtofloat(frmMain.txt100part.Text)*strtoint(frmMain.txtTermMoney.Tex
t)/100),0));

frmMain.txt50.Text:=floattostr(roundto((strtofloat(frmMain.txt50part.Text)*strtoint(frmMain.txtTermMoney.Text)/
50),0));

frmMain.txt20.Text:=floattostr(roundto((strtofloat(frmMain.txt20part.Text)*strtoint(frmMain.txtTermMoney.Text)/
20),0));
  frmMain.txt10.Text:=floattostr((strtofloat(frmMain.txtTermMoney.Text)-strtofloat(frmMain.txt200.Text)*200-
strtofloat(frmMain.txt100.Text)*100-strtofloat(frmMain.txt50.Text)*50-strtofloat(frmMain.txt20.Text)*20)/10);
  if ((strtofloat(frmMain.txt200part.Text) + strtofloat(frmMain.txt100part.Text) + strtofloat(frmMain.txt50part.Text)
+ strtofloat(frmMain.txt20part.Text) + strtofloat(frmMain.txt10part.Text))<>1) then
  begin
    showmessage('Перевірте рорзподіл купюр');
    exit;
```

```
end;
setlength(frmMain.welcome,strtoint(frmMain.txtRealiseQuant.Text));
setlength(frmMain.empty,strtoint(frmMain.txtRealiseQuant.Text));
setlength(frmMain.recharge,strtoint(frmMain.txtRealiseQuant.Text));
if frmMain.global_iter_count<strtoint(frmMain.txtRealiseQuant.Text) then
begin
  frmMain.free_all_flag:=false;
  World:=TWorld.Create;
  World.Start;
  inc(frmMain.numR);
  frmMain.txtNumRealis.Text:=inttostr(numR);
end
else
if frmMain.global_iter_count=strtoint(frmMain.txtRealiseQuant.Text) then
begin
  showmessage('Моделювання закінчено!');
  frmMain.numR:=0;
  frmMain.global_iter_count:=0;
  if optim=false then
  begin
  temp1:=0;
  temp2:=0;
  temp3:=0;
  temp4:=0;
  temp5:=0;
  temp6:=0;
  temp7:=0;
  temp8:=0;
  temp9:=0;
  temp10:=0;
  temp11:=0;
  temp12:=0;
  temp13:=0;
  temp14:=0;
  temp15:=0;
  temp16:=0;
  temp17:=0;
  temp18:=0;
  temp19:=0;
  tempK1:=0;
  tempK2:=0;
  tempK3:=0;
  for r:=0 to strtoint(frmMain.txtRealiseQuant.Text)-1 do
  begin
    temp1:=temp1+strtofloat(frmResult.StringGrid1.Cells[r,1]);
    temp2:=temp2+strtofloat(frmResult.StringGrid1.Cells[r,2]);
    temp3:=temp3+strtofloat(frmResult.StringGrid1.Cells[r,3]);
    temp4:=temp4+strtofloat(frmResult.StringGrid1.Cells[r,4]);
    temp5:=temp5+strtofloat(frmResult.StringGrid1.Cells[r,5]);
    temp6:=temp6+strtofloat(frmResult.StringGrid1.Cells[r,6]);
    temp7:=temp7+strtofloat(frmResult.StringGrid1.Cells[r,7]);
    temp8:=temp8+strtofloat(frmResult.StringGrid1.Cells[r,8]);
    temp9:=temp9+strtofloat(frmResult.StringGrid1.Cells[r,9]);
    temp10:=temp10+strtofloat(frmResult.StringGrid1.Cells[r,10]);
    temp11:=temp11+strtofloat(frmResult.StringGrid1.Cells[r,11]);
    temp12:=temp12+strtofloat(frmResult.StringGrid1.Cells[r,12]);
    temp13:=temp13+strtofloat(frmResult.StringGrid1.Cells[r,13]);
    temp14:=temp14+strtofloat(frmResult.StringGrid1.Cells[r,14]);
    temp15:=temp15+strtofloat(frmResult.StringGrid1.Cells[r,15]);
    temp16:=temp16+strtofloat(frmResult.StringGrid1.Cells[r,16]);
    temp17:=temp17+strtofloat(frmResult.StringGrid1.Cells[r,17]);
    temp18:=temp18+strtofloat(frmResult.StringGrid1.Cells[r,18]);
```

```
      temp19:=temp19+strtofloat(frmResult.StringGrid1.Cells[r,19]);
      tempK1:=tempK1+welcome[r];
      tempK2:=tempK2+empty[r];
      tempK3:=tempK3+recharge[r];
    end;
    frmResult.txtClCount.Text:=floattostr(roundto(temp1/strtoint(frmMain.txtRealiseQuant.Text),-3));
    frmResult.txtServe.Text:=floattostr(roundto(temp2/strtoint(frmMain.txtRealiseQuant.Text),-3));
    frmResult.txtNoServe.Text:=floattostr(roundto(temp3/strtoint(frmMain.txtRealiseQuant.Text),-3));
    frmResult.txtNServe_nem.Text:=floattostr(roundto(temp4/strtoint(frmMain.txtRealiseQuant.Text),-3));
    frmResult.txtNServ_rq.Text:=floattostr(roundto(temp5/strtoint(frmMain.txtRealiseQuant.Text),-3));
    frmResult.txtNServ_rech.Text:=floattostr(roundto(temp6/strtoint(frmMain.txtRealiseQuant.Text),-3));
    frmResult.txtNServ_qtb.Text:=floattostr(roundto(temp7/strtoint(frmMain.txtRealiseQuant.Text),-3));
    frmResult.txtNServe_note.Text:=floattostr(roundto(temp8/strtoint(frmMain.txtRealiseQuant.Text),-3));
    frmResult.txtAvgQueueQuan.Text:=floattostr(roundto(temp9/strtoint(frmMain.txtRealiseQuant.Text),-3));
    frmResult.txtAvgQueueTime.Text:=floattostr(roundto(temp10/strtoint(frmMain.txtRealiseQuant.Text),-3));
    frmResult.txtAvgServeTime.Text:=floattostr(roundto(temp11/strtoint(frmMain.txtRealiseQuant.Text),-3));
    frmResult.txtRechargePeriod.Text:=floattostr(roundto(temp12/strtoint(frmMain.txtRealiseQuant.Text),-3));
    frmResult.txtRestRecharge.Text:=floattostr(roundto(temp13/strtoint(frmMain.txtRealiseQuant.Text),-3));
    frmResult.txtAbsTraf.Text:=floattostr(roundto(temp14/strtoint(frmMain.txtRealiseQuant.Text),-3));
    frmResult.txtRelTraf.Text:=floattostr(roundto(temp15/strtoint(frmMain.txtRealiseQuant.Text),-3));
    frmResult.txtLambdaEmp.Text:=floattostr(roundto(temp16/strtoint(frmMain.txtRealiseQuant.Text),-3));
    frmResult.txtMuEmp.Text:=floattostr(roundto(temp17/strtoint(frmMain.txtRealiseQuant.Text),-3));
    frmResult.txtAvgsum.Text:=floattostr(roundto(temp18/strtoint(frmMain.txtRealiseQuant.Text),-2));
    frmResult.txtAvgHourSum.Text:=floattostr(roundto(temp19/strtoint(frmMain.txtRealiseQuant.Text),-2));
    frmResult.txtKWait.Text:=floattostr(roundto((tempK1/(24*strtoint(frmMain.txtRealiseQuant.Text))),-3));
    frmResult.txtKEmpty.Text:=floattostr(roundto((tempK2/(24*strtoint(frmMain.txtRealiseQuant.Text))),-3));
    frmResult.txtKRecharge.Text:=floattostr(roundto((tempK3/(24*strtoint(frmMain.txtRealiseQuant.Text))),-3));
    frmResult.txtKUse.Text:=floattostr(roundto((1-strtofloat(frmResult.txtKWait.Text)-
strtofloat(frmResult.txtKEmpty.Text)- strtofloat(frmResult.txtKRecharge.Text)),-3));
    if strtofloat(frmResult.txtKUse.Text)>0.6 then
      frmResult.Label52.Caption:='Ефективно';
    if (strtofloat(frmResult.txtKUse.Text)>0.4) and (strtofloat(frmResult.txtKUse.Text)<0.6) then
      frmResult.Label52.Caption:='Не достатньо ефективно';
    if  strtofloat(frmResult.txtKUse.Text)<0.4 then
      frmResult.Label52.Caption:='Не ефективно';
    end;
    if optim=true then
    begin
      count:=0;
    end;
  end;
end;

procedure TfrmMain.FormCreate(Sender: TObject);
begin
  frmMain.numR:=0;
  free_all_flag:=false;
  global_iter_count:=0;
  optim:=false;
  count:=0;
  StringGrid1.Cells[0,0]:='100,5';
  StringGrid1.Cells[0,1]:='119';
  StringGrid1.Cells[0,2]:='100,75';
  StringGrid1.Cells[0,3]:='193,4';
  StringGrid1.Cells[0,4]:='173,5';
  StringGrid2.Cells[0,0]:='0,005';
  StringGrid2.Cells[0,1]:='0,005';
  StringGrid2.Cells[0,2]:='0,005';
  StringGrid2.Cells[0,3]:='0,005';
  StringGrid2.Cells[0,4]:='0,005';
  StringGrid2.Cells[0,5]:='0,005';
  StringGrid2.Cells[0,6]:='0,005';
```

```
  StringGrid2.Cells[0,7]:='0,027';
  StringGrid2.Cells[0,8]:='0,093';
  StringGrid2.Cells[0,9]:='0,053';
  StringGrid2.Cells[0,10]:='0,055';
  StringGrid2.Cells[0,11]:='0,055';
  StringGrid2.Cells[0,12]:='0,037';
  StringGrid2.Cells[0,13]:='0,110';
  StringGrid2.Cells[0,14]:='0,065';
  StringGrid2.Cells[0,15]:='0,079';
  StringGrid2.Cells[0,16]:='0,097';
  StringGrid2.Cells[0,17]:='0,161';
  StringGrid2.Cells[0,18]:='0,077';
  StringGrid2.Cells[0,19]:='0,027';
  StringGrid2.Cells[0,20]:='0,014';
  StringGrid2.Cells[0,21]:='0,005';
  StringGrid2.Cells[0,22]:='0,005';
  StringGrid2.Cells[0,23]:='0,005';
  free_all_timer:=TTimer.Create(nil);
  free_all_timer.Interval:=100;
  free_all_timer.OnTimer:=FreeAll;
end;

procedure TfrmMain.FreeAll(Sender: TObject);
var h:integer;
    CRech:string;
begin
  if frmMain.free_all_flag=true then
  begin
    frmMain.free_all_timer.Enabled:=false;
    for h:=0 to World.clients_count-1 do
    begin
      world.clients[h].wait_timer.Free;
      world.clients[h]^.Free;
      dispose(world.clients[h]);
    end;
    Queue.Free;
    Bank.recharge_timer.Free;
    Bank.Free;
    Calc.freq_timer.Free;
    Calc.welcome_timer.Free;
    if optim=false then
    begin
    frmResult.StringGrid1.Cells[frmMain.global_iter_count,0]:=inttostr(frmMain.numR);
    frmResult.StringGrid1.Cells[frmMain.global_iter_count,1]:=floattostr(strtoint(frmMain.txtCl_count.Text)/5);
    frmResult.StringGrid1.Cells[frmMain.global_iter_count,2]:=floattostr(World.served_clients/5);
    frmResult.StringGrid1.Cells[frmMain.global_iter_count,3]:=floattostr((strtoint(frmMain.txtCl_count.Text)-
World.served_clients)/5);
    frmResult.StringGrid1.Cells[frmMain.global_iter_count,4]:=floattostr(Calc.n_serv_nem/5);
    frmResult.StringGrid1.Cells[frmMain.global_iter_count,5]:=floattostr(Calc.n_serv_rq/5);
    frmResult.StringGrid1.Cells[frmMain.global_iter_count,6]:=floattostr(Calc.n_serv_rech/5);
    frmResult.StringGrid1.Cells[frmMain.global_iter_count,7]:=floattostr(Calc.n_serv_qtb/5);
    frmResult.StringGrid1.Cells[frmMain.global_iter_count,8]:=floattostr(Calc.n_serv_note/5);
    frmResult.StringGrid1.Cells[frmMain.global_iter_count,9]:=floattostr(Calc.avg_queue_quan);
    frmResult.StringGrid1.Cells[frmMain.global_iter_count,10]:=floattostr(roundto(Calc.avg_queue_time/(100/60),-
3));
    frmResult.StringGrid1.Cells[frmMain.global_iter_count,11]:=floattostr(roundto(Calc.avg_serve_time/(100/60),-
3));
    if Calc.count_recharge<>0 then
      frmResult.StringGrid1.Cells[frmMain.global_iter_count,12]:=floattostr(120/Calc.count_recharge)
    else
      frmResult.StringGrid1.Cells[frmMain.global_iter_count,12]:='120';
    frmResult.StringGrid1.Cells[frmMain.global_iter_count,13]:=floattostr(Calc.avg_rest_recharge);
```

```
   frmResult.StringGrid1.Cells[frmMain.global_iter_count,14]:=floattostr(Calc.A);
   frmResult.StringGrid1.Cells[frmMain.global_iter_count,15]:=floattostr(Calc.q);
   frmResult.StringGrid1.Cells[frmMain.global_iter_count,16]:=floattostr(roundto((World.clients_count-
1)/strtoint(frmMain.txtTime.Text),-2));

frmResult.StringGrid1.Cells[frmMain.global_iter_count,17]:=floattostr(roundto(World.served_clients/strtoint(frmM
ain.txtTime.Text),-2));
   frmResult.StringGrid1.Cells[frmMain.global_iter_count,18]:=floattostr(roundto(Calc.avgOperSum,-3));

frmResult.StringGrid1.Cells[frmMain.global_iter_count,19]:=floattostr(roundto(Calc.avgHourSum/strtoint(frmMain
.txtTime.Text),-2));
   end;
   frmMain.welcome[global_iter_count]:=roundto((Calc.welcome_count/(100/60))/60,-3);
   frmMain.empty[global_iter_count]:=roundto((Calc.empty_count/(100/60))/60,-3);
   frmMain.recharge[global_iter_count]:=roundto((Calc.count_recharge*20/60),-3);
   if optim=true then
   begin
     if Calc.count_recharge<>0 then
       CRech:=floattostr(120/Calc.count_recharge)
     else
       CRech:='120';
     frmResult.StringGrid2.Cells[0,0]:='Витрати';
     frmResult.StringGrid2.Cells[0,count]:=floattostr(roundto((                ((Calc.avgHourSum                *
strtofloat(frmMain.txtProcent.text))                                /(strtoint(frmMain.txtTime.Text)*24*100))+
(strtofloat(frmMain.txtCostRecharge.Text)*strtofloat(frmMain.txtTime.text)/strtofloat(CRech))),-2));
     frmResult.StringGrid2.Cells[1,0]:='Необслужені клієнти';
     frmResult.StringGrid2.Cells[1,count]:=floattostr(roundto(((strtoint(frmMain.txtCl_count.Text)-
World.served_clients-Calc.n_serv_qtb)/5)/(World.served_clients/5), -2));
     frmResult.StringGrid2.Cells[2,0]:='Коеф. використання';
     frmResult.StringGrid2.Cells[2,count]:=          floattostr(roundto(1-((Calc.welcome_count/(100/60))/60)/24-
((Calc.empty_count/(100/60))/60)/24- (Calc.count_recharge*20/60)/24,-2));
     frmResult.StringGrid2.Cells[3,0]:='Сума завантаження';
     frmResult.StringGrid2.Cells[4,0]:='Критична сума';
     frmResult.StringGrid2.Cells[3,count]:=frmMain.txtTermMoney.Text;
     frmResult.StringGrid2.Cells[4,count]:=frmMain.txtCritical.Text;
   end;
   Calc.Free;
   World.Free;
   inc(frmMain.global_iter_count);
   frmMain.Button1.Click;
  end;
end;

procedure TfrmMain.Optimisation;
var min1, min2: real;
   n,g1:integer;
begin
       g1:=1;
       min2:=strtofloat(frmResult.StringGrid2.cells[0,1]);
       min1:=9999999;
       for n:=1 to round(frmResult.StringGrid2.RowCount) do
       begin
         min1:=min(min1,strtofloat(frmResult.StringGrid2.cells[0,n]));
         if min1<=min2 then
         begin
           if (strtofloat(frmResult.StringGrid2.cells[1,n])<=strtofloat(frmMain.txtPossibleNeobsl.Text)/100) then
           begin
             if strtofloat(frmResult.StringGrid2.cells[2,n])>=0.6 then
             begin
               if strtofloat(frmResult.StringGrid2.cells[0,n])<strtofloat(frmResult.StringGrid2.cells[0,g1]) then
                 g1:=n
             end
```

```
            else
              min1:=9999999;
            end
            else
              min1:=9999999;
          end;
          min2:=min1;
        end;
        frmResult.txtOptZ.Text:=frmResult.StringGrid2.cells[3,g1];
        frmResult.txtOptCritical.Text:=frmResult.StringGrid2.cells[4,g1];
        frmResult.txtCost.Text:=frmResult.StringGrid2.cells[0,g1];
        frmResult.txtUse.Text:=frmResult.StringGrid2.cells[2,g1];
        frmResult.txtn_obsl.Text:=frmResult.StringGrid2.cells[1,g1];
        showmessage('Розрахунок закінчено!');
        Exit;
end;

procedure TfrmMain.Button2Click(Sender: TObject);
begin
  frmResult.Show;
end;

procedure TfrmMain.Button3Click(Sender: TObject);
begin
  frmMain.Close;
  frmResult.Close;
end;

procedure TfrmMain.Button4Click(Sender: TObject);
var row: real;
begin
  row:=(((strtoint(frmMain.txtMaxZ.Text)-
strtoint(frmMain.txtMinZ.Text))/5000)+1)*(((strtoint(frmMain.txtMaxCritical.Text)-
strtoint(frmMain.txtMinCritical.Text))/500)+1);
  frmResult.StringGrid2.RowCount:=round(row)+1;
  optim:=true;
  TempBankMoney:= strtoint(txtMinZ.Text);
  TempBankCritical:=strtoint(txtMinCritical.Text)-500;
  lock:=true;
  frmMain.txtTermMoney.Text:=floattostr(TempBankMoney);
  frmMain.txtCritical.Text:=floattostr(TempBankCritical);
  frmMain.Button1.Click;
end;

procedure TfrmMain.Button5Click(Sender: TObject);
begin
  Optimisation;
end;

end.
```

Додаток Б

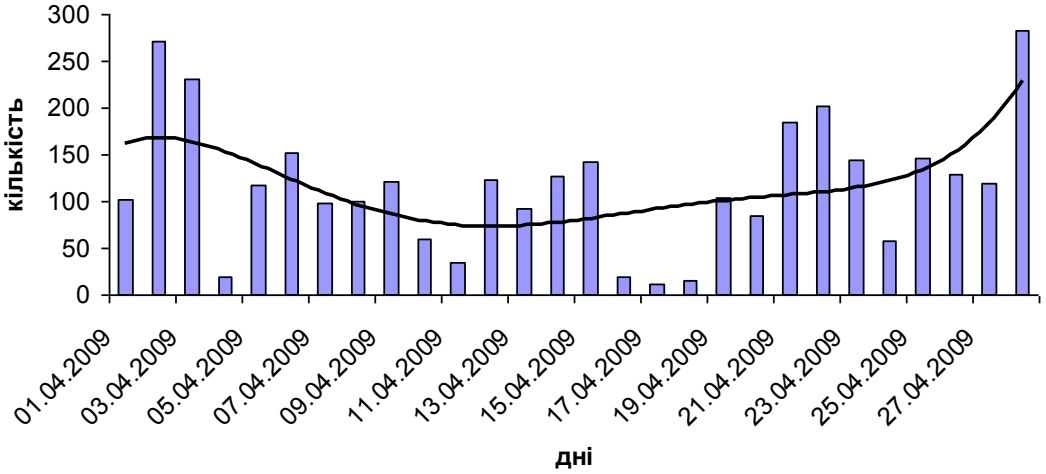

Рисунок Б.1 – Гістограма розподілу клієнтів по дням

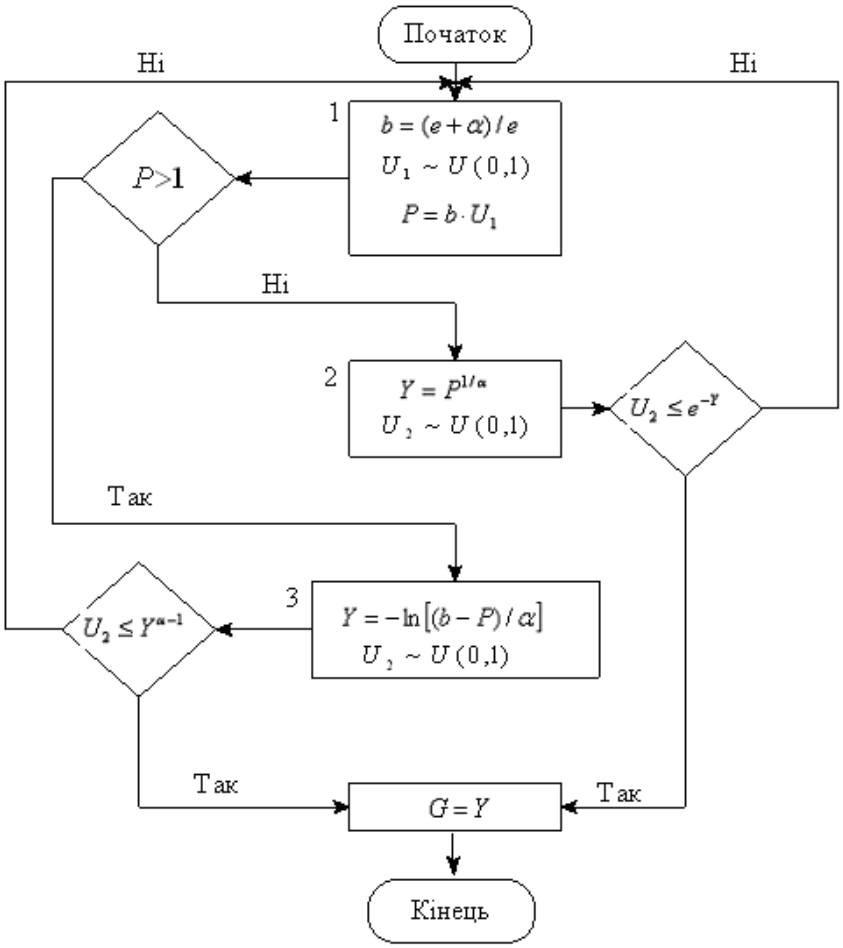

Рисунок Б.2 – Алгоритм генерування суми операції

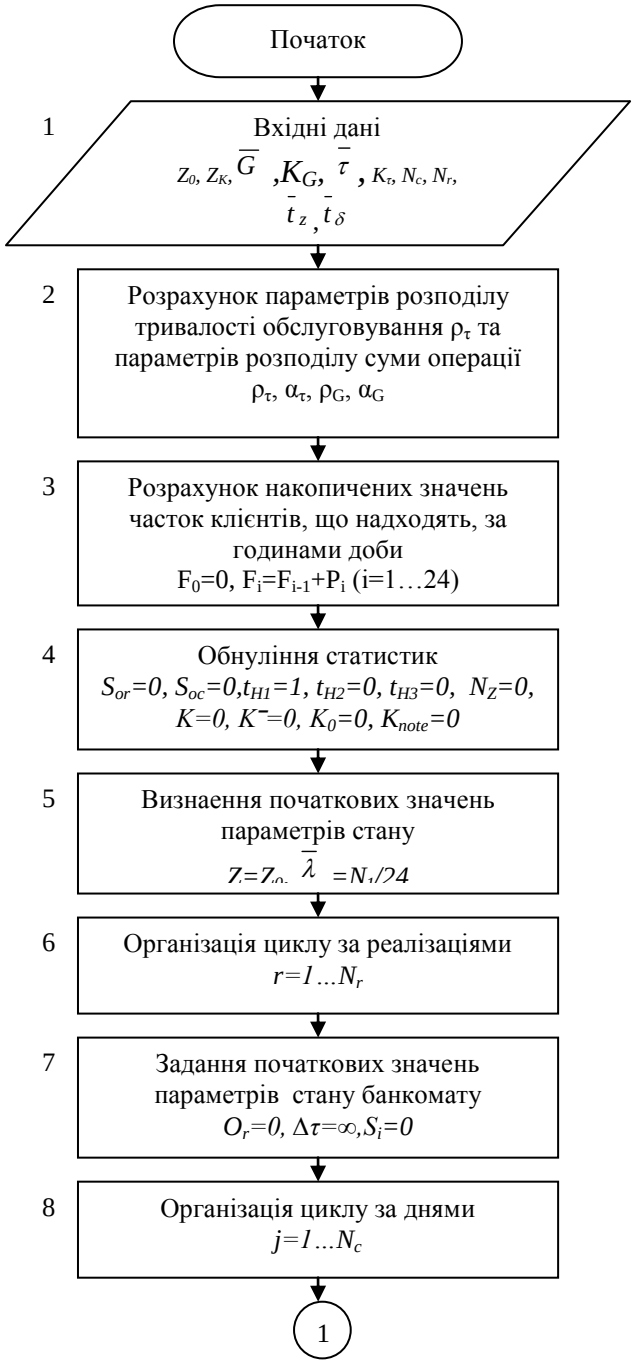

Рисунок Б.3 – Блок-схема алгоритму статистичного моделювання функціонування банкомату

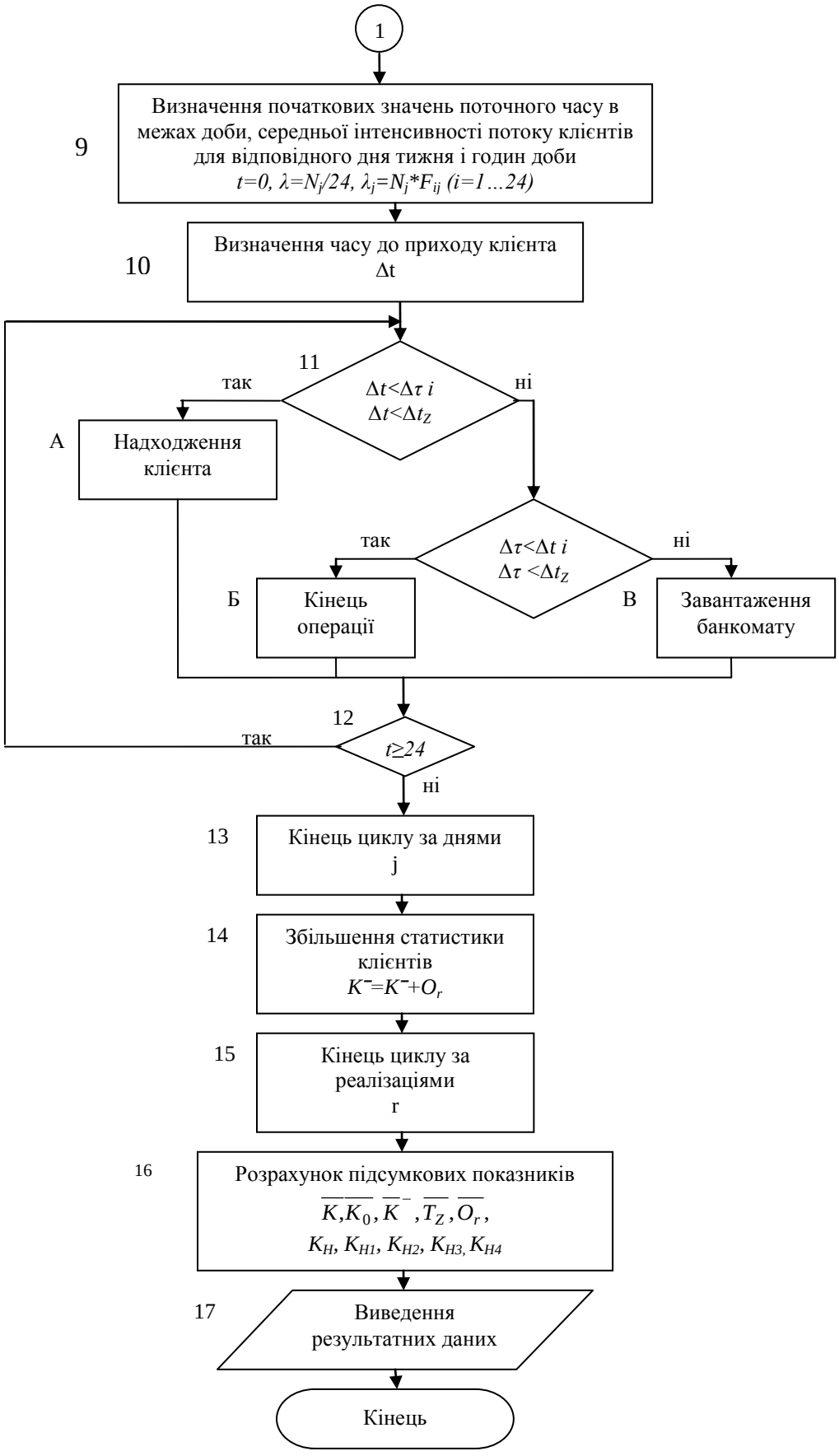

Рисунок Б.3, аркуш 2

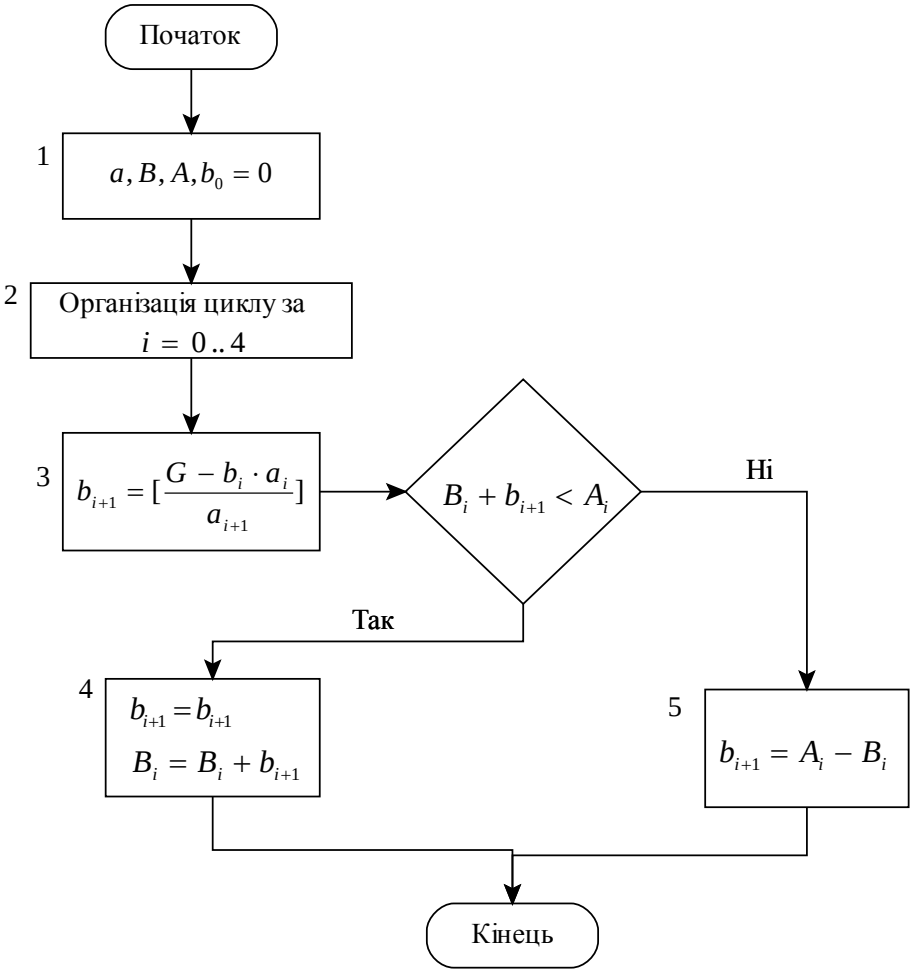

Рисунок Б.4 – Алгоритм розподілу купюр

Додаток В

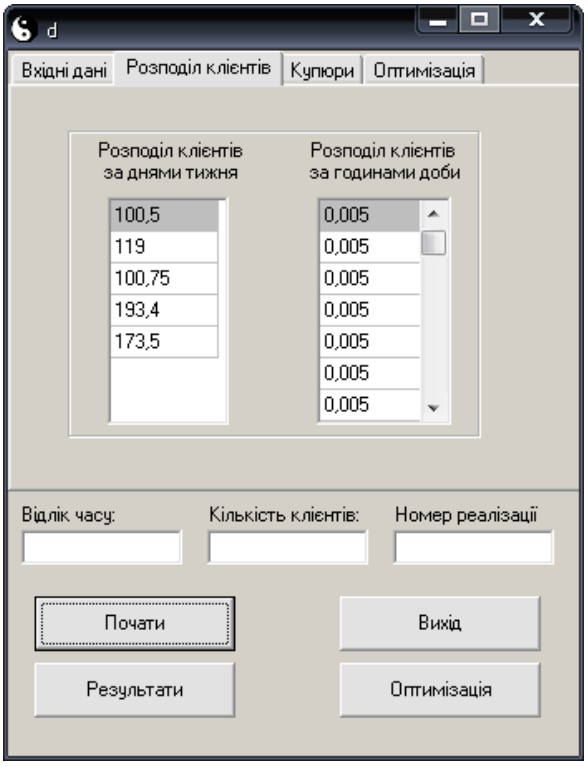

Рисунок В.1 – Інтерфейс користувача (вкладка «Розподіл клієнтів»)

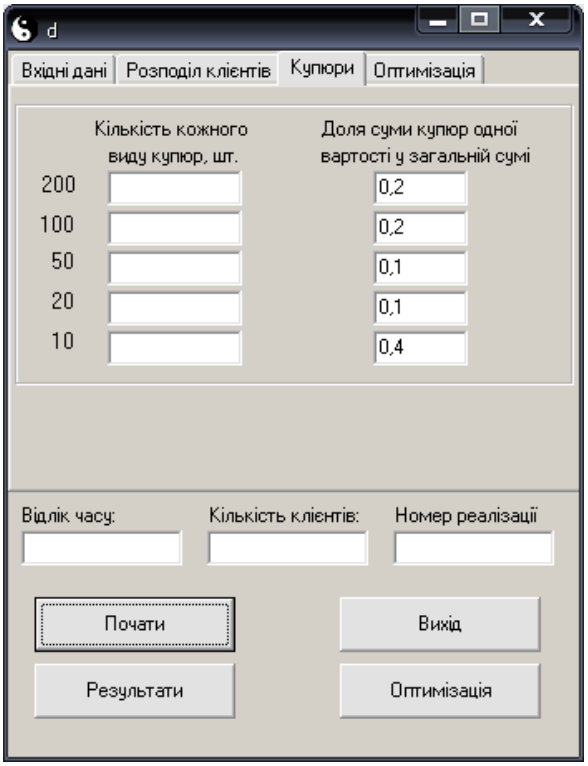

Рисунок В.2 – Інтерфейс користувача (вкладка «Купюри»)

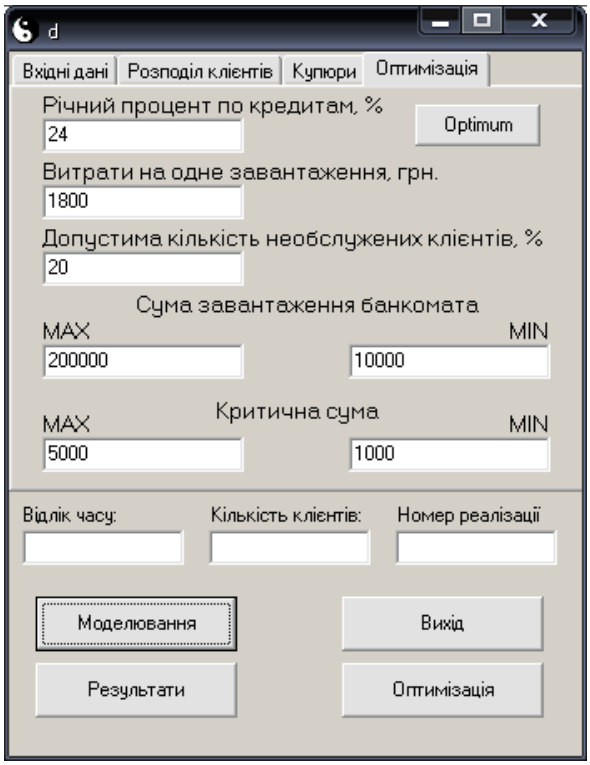

Рисунок В.3 – Інтерфейс користувача (вкладка «Оптимізація»)

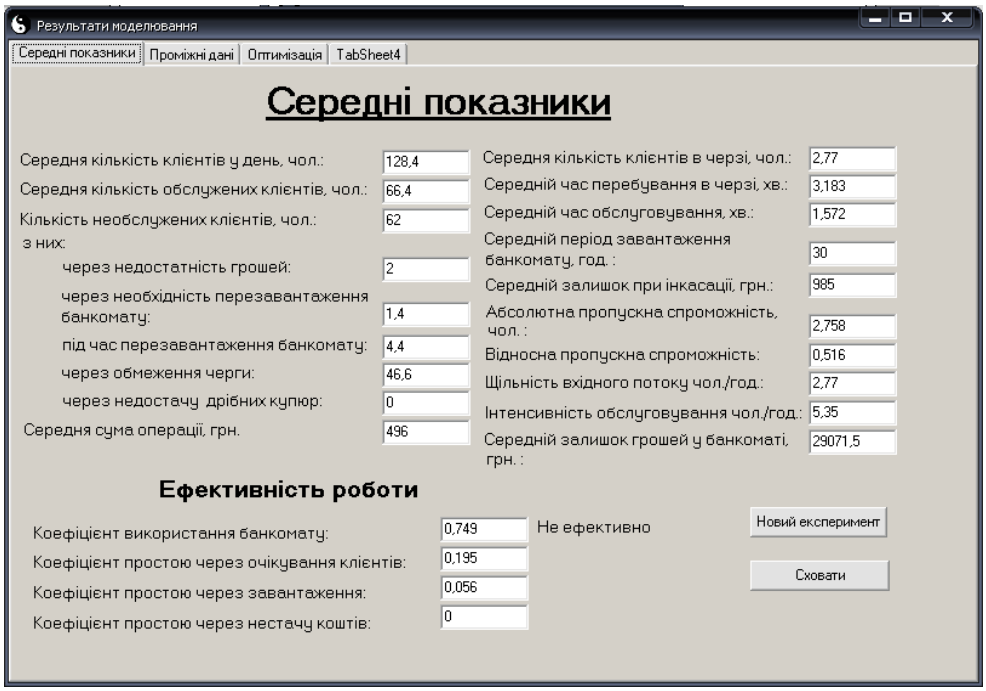

Рисунок В.4 – Результати моделювання (вкладка «Середні показники»)

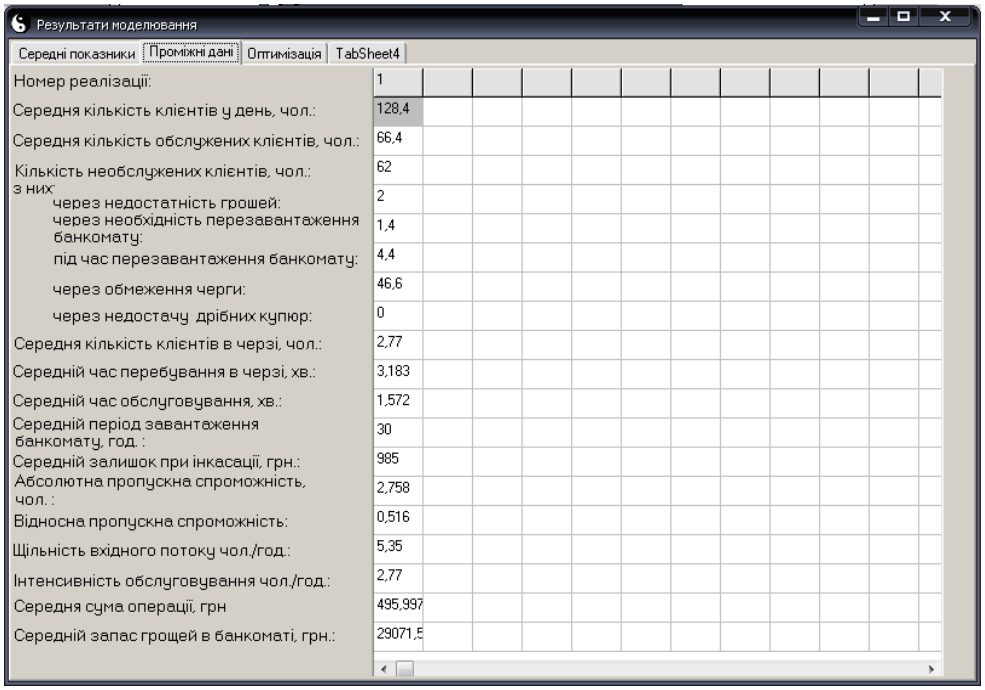

Рисунок В.5 – Результати моделювання (вкладка «Проміжні дані»)

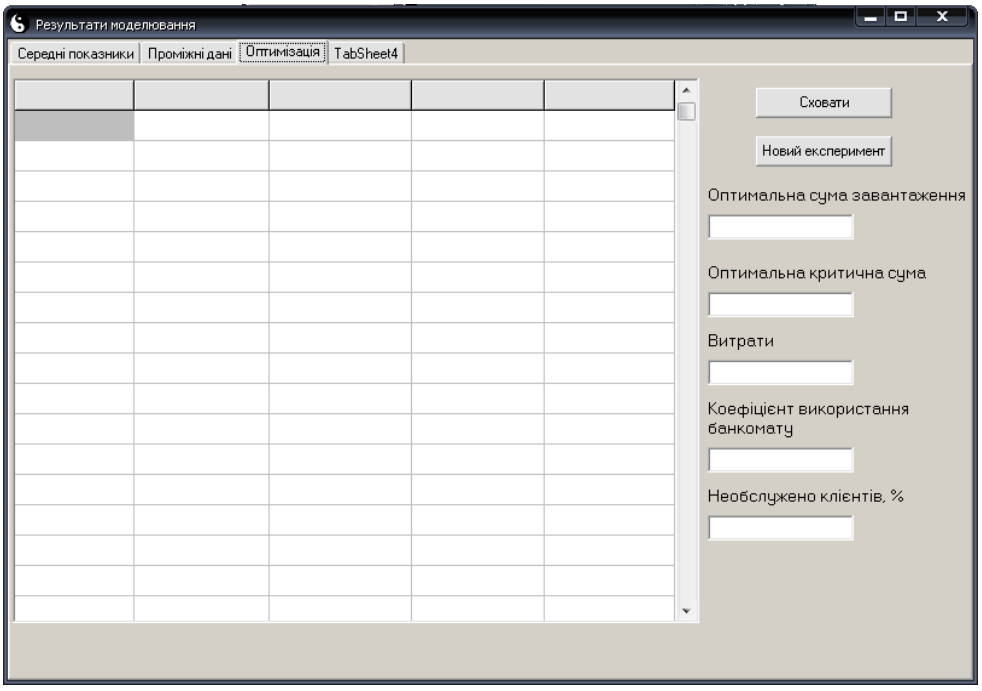

Рисунок В.6 – Результати моделювання (вкладка «Оптимізація»)

Додаток Г

Таблиця Г.1 – Кількість клієнтів по дням тижня

	Понеділок	Вівторок	Середа	Четвер	П’ятниця	Субота	Неділя
Кількість			100	272	230	19	33
	117	151	98	100	121	59	34
	123	93	99	127	142	20	12
	16	103	85	185	201	145	57
	146	129	120	283			
Всього	402	476	403	967	694	243	103
Середнє	100,5	119	100,75	193,4	173,5	60,75	34,33

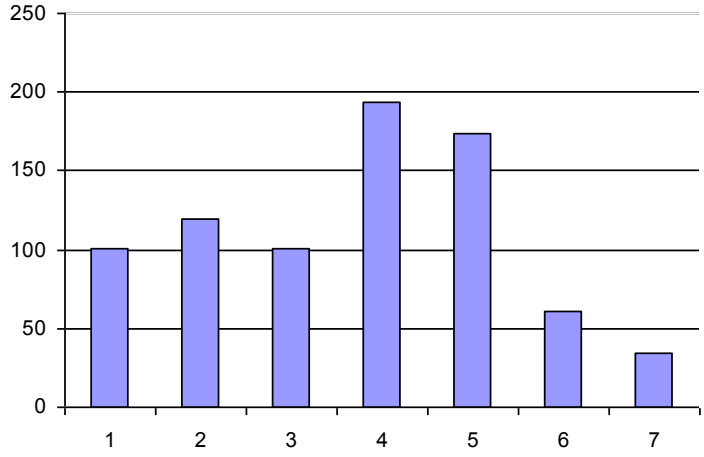

Рисунок Г.1 – Розподіл клієнтів банкомату по дням тижня

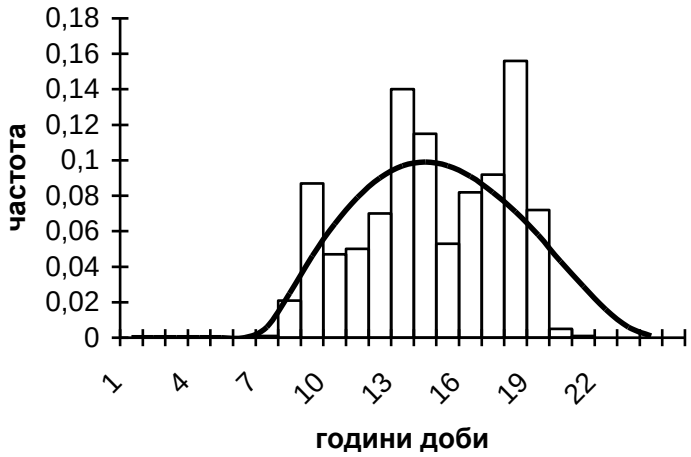

Рисунок Г.2 – розподіл клієнтів за годинами доби

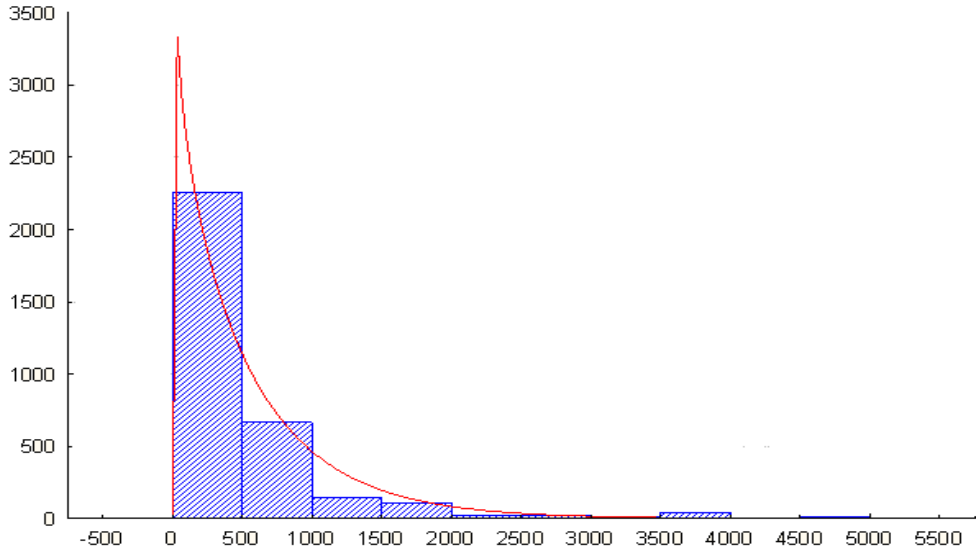

Рисунок Г.3 – Гістограма розподілу суми операції

Таблиця Г.2 – Реакція моделі на зміну критичної суми

	10 грн.	100 грн.	1000 грн.	10000 грн.
Середня кількість клієнтів у день, чол.	124,56	123,72	123,4	124,8
Середня кількість обслужених клієнтів у день, чол.	81,08	82,28	83,48	85,92
Кількість не обслужених клієнтів, чол.	43,48	41,44	39,92	38,88
Кількість не обслужених клієнтів через недостатність грошей, чол.	1,28	1,08	0	0
Кількість не обслужених клієнтів через необхідність перезавантаження банкомату, чол.	0,88	0,88	0,76	1,16
Кількість не обслужених клієнтів під час завантаження банкомату, чол.	0,8	0,64	0,24	1,2
Кількість не обслужених клієнтів через обмеження черги, чол.	41,24	39,64	39,16	36,84
Кількість не обслужених клієнтів через недостачу дрібних купюр, чол..	0,2	0	0	0,1
Середня кількість клієнтів в черзі, чол.	2,82	2,854	2,842	3,03
Середній час перебування в черзі, хв.	3,26	3,347	3,458	3,35
Середній час обслуговування, хв.	2,02	2,028	2,002	1,996
Середній період завантаження, год.	108	108	120	72
Середній залишок при інкасації, грн.	0	0	506,8	8501,2
Абсолютна пропускна спроможність, чол.	3,37	3,42	3,47	3,572
Відносна пропускна спроможність	0,651	0,663	0,676	0,686
Щільність вхідного потоку, чол./год.	5,19	5,154	5,142	5,198
Інтенсивність обслуговування, чол./год.	3,38	3,426	3,478	3,582
Кількість реалізацій	5	5	5	5

Таблиця Г.3 – Реакція моделі на зміну початкової суми

	10000 грн.	30000 грн.	50000 грн.	100000 грн.
Середня кількість клієнтів у день, чол.	123,28	122,64	121,08	120,04
Середня кількість обслужених клієнтів у день, чол.	57,04	63,4	81,68	82,24
Кількість не обслужених клієнтів, чол.	66,24	59,24	39,4	37,8
Кількість не обслужених клієнтів через недостатність грошей, чол.	2,44	0,72	0,44	0,12
Кількість не обслужених клієнтів через необхідність перезавантаження банкомату, чол.	1,12	1	1,08	0,08
Кількість не обслужених клієнтів під час завантаження банкомату, чол.	4,44	1,48	0,6	0,08
Кількість не обслужених клієнтів через обмеження черги, чол.	58,16	55,92	37,88	37,64
Кількість не обслужених клієнтів через обмеження черги, чол.	0,2	0,2	0,2	0
Середня кількість клієнтів в черзі, чол.	2,028	2,284	2,746	2,782
Середній час перебування в черзі, хв.	2,322	3,126	3,274	3,369
Середній час обслуговування, хв.	2,028	1,984	1,954	2,01
Середній період завантаження, год.	21,095	68	108	120
Середній залишок при інкасації, грн.	326,688	526,733	426,8	0
Абсолютна пропускна спроможність, чол.	2,368	2,633	3,395	3,418
Відносна пропускна спроможність	0,461	0,516	0,672	0,684
Щільність вхідного потоку, чол./год.	5,138	5,11	5,044	5,002
Інтенсивність обслуговування, чол./год.	2,378	2,644	3,402	3,428
Кількість реалізацій	5	5	5	5

Таблиця Г.4 – Реакція моделі на зміну купюрної структури

	100%-200	100%- 10	20% - 200 20% - 100 10% - 50 10% - 20 40% - 10
Середня кількість клієнтів у день, чол.	121,24	111,2	123,24
Середня кількість обслужених клієнтів у день, чол.	79,88	74	79,44
Кількість не обслужених клієнтів, чол.	41,36	37,2	43,8
Кількість не обслужених клієнтів через недостатність грошей, чол.	3,52	2	2,24
Кількість не обслужених клієнтів через необхідність перезавантаження банкомату, чол.	1,72	1,8	1,16
Кількість не обслужених клієнтів під час завантаження банкомату, чол.	3,56	2	2,28
Кількість не обслужених клієнтів через обмеження черги, чол.	34,28	33,4	37
Кількість не обслужених клієнтів через обмеження черги, чол.	3,56	0,6	0
Середня кількість клієнтів в черзі, чол.	2,702	2,36	2,874
Середній час перебування в черзі, хв.	3,2	3,234	3,481
Середній час обслуговування, хв.	2,702	2,305	2,596
Середній період завантаження, год.	40	40	30
Середній залишок при інкасації, грн.	160,667	356,667	92,5
Абсолютна пропускна спроможність, чол.	3,32	3,075	3,302
Відносна пропускна спроможність	0,658	0,664	0,643
Щільність вхідного потоку, чол./год.	5,054	4,63	5,132
Інтенсивність обслуговування, чол./год.	3,33	3,08	3,308
Кількість реалізацій	5	5	5

Таблиця Г.5 – Порівняльний аналіз банкоматів, розміщених у різних районах

	Приміський район	Центр міста
Середня кількість клієнтів у день, чол.	8,8	137,6
Середня кількість обслужених клієнтів у день, чол.	7,8	85,2
Кількість не обслужених клієнтів, чол.	1	52,4
Середня кількість клієнтів в черзі, чол.	0,02	3,46
Середній час перебування в черзі, хв.	0,6	3,61
Середній час обслуговування, хв.	1,923	1,584
Середній період завантаження, год.	120	60
Середній залишок при інкасації, грн.	0	0
Абсолютна пропускна спроможність, чол.	0,317	3,542
Відносна пропускна спроможність	0,864	0,618
Щільність вхідного потоку, чол./год.	0,33	3,55
Інтенсивність обслуговування, чол./год.	0,7	5,73
Середня сума операції, грн.	668,864	483,84
Середній запас грошей в банкоматі, грн.	86445,7	54064
Коефіцієнт простою через нестачу коштів	0	0,001
Коефіцієнт простою через очікування клієнтів	0,632	0,16
Коефіцієнт простою через завантаження	0	0,028
Коефіцієнт використання	0,368	0,811

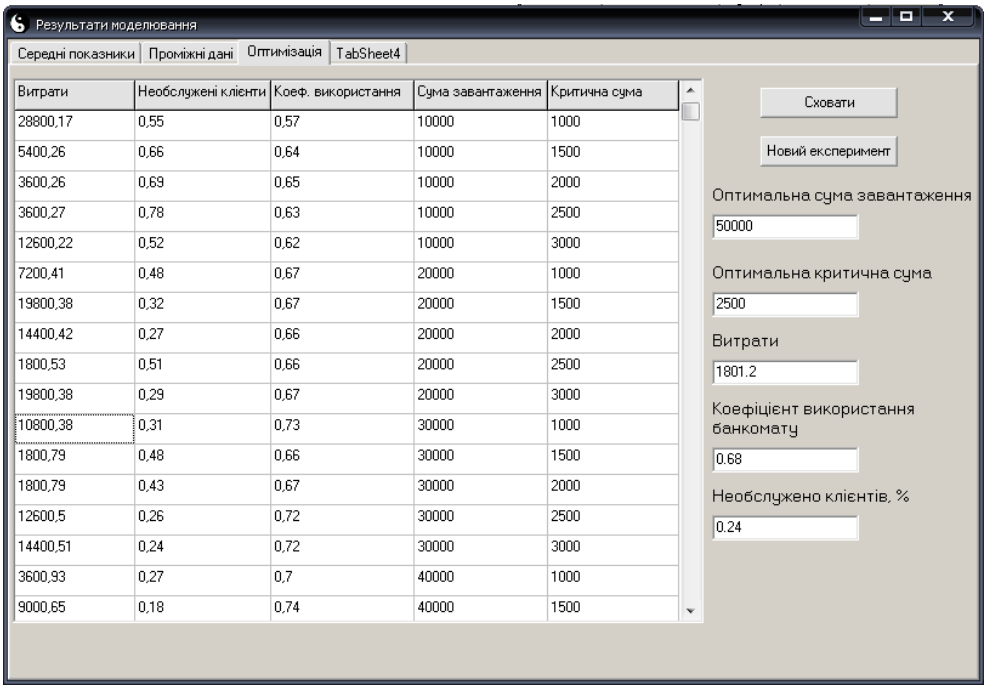

Рисунок Г.4 – результати оптимізації реально-діючого банкомату