МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра прикладної математики і механіки

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«АЛГОРИТМІЗАЦІЯ ПРОЦЕСУ МОДЕЛЮВАННЯ ЗАДАЧ ПРИЙНЯТТЯ РІШЕНЬ З ВЕЛИКИМ ЧИСЛОМ АЛЬТЕРНАТИВ НА ОСНОВІ ДЕСКРИПТИВНИХ МОДЕЛЕЙ»**

Виконав: студент 2 курсу, групи 8.1138-з

спеціальності 113 прикладна математика
(шифр і назва спеціальності)

освітньої програми прикладна математика
(назва освітньої програми)

Д. І. Станкевич
(ініціали та прізвище)

Керівник доцент кафедри прикладної математики і механіки, доцент, к. ф-м. н. Кондрат'єва Н. О.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент декан математичного факультету, професор, д.т.н. Гоменюк С. І.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра кафедра прикладної математики і механіки

Рівень вищої освіти магістр

Спеціальність 113 прикладна математика

(шифр і назва)

Освітня програма прикладна математика

ЗАТВЕРДЖУЮ

Завідувач кафедри прикладної математики і механіки, д.т.н., професор

Грищак В. З.

(підпис)

«29» 05 2019 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Станкевичу Данилу Ігоровичу

(прізвище, ім'я та по-батькові)

1. Тема роботи (проекту) Алгоритмізація процесу моделювання задач прийняття рішень з великим числом альтернатив на основі дескриптивних моделей

керівник роботи (проекту) Кондрат'єва Н. О., к. ф-м. н., доцент

(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 29 » 05 2019 року № 812-с

2. Строк подання студентом роботи 26.12.2019

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Основні теоретичні відомості.
3. Вибір інструментарію розробки і планування системи.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Презентація

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 29.05.2019

КАЛЕНДАРНИЙ ПЛАН

	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	29.05.19-30.07.19	
2.	Збір вихідних даних.	01.08.19-15.08.19	
3.	Обробка методичних та теоретичних	16.08.19-01.09.19	
	джерел.		
4.	Розробка першого розділу.	01.09.19-01.10.19	
5.	Розробка другого розділу.	01.10.19-01.11.19	
6.	Оформлення та нормоконтроль	01.11.19-27.12.19	
	кваліфікаційної роботи.		
7.	Захист кваліфікаційної роботи.	09.01.2020	

Студент _____ Д. І. Станкевич
(підпис) (ініціали та прізвище)

Керівник роботи _____ Н.О. Кондрат'єва
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ В. В. Леонтьєва
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Алгоритмізація процесу моделювання задач прийняття рішень з великим числом альтернатив на основі дескриптивних моделей»: 64 с., 12 рис., 3 табл., 24 джерела, 1 додаток.

АЛЬТЕРНАТИВА, КРИТЕРІЙ, МЕТА, ОСОБА ЩО ПРИЙМАЄ РІШЕННЯ, ТЕОРІЯ ПРИЙНЯТТЯ РІШЕННЯ.

Об’єкт дослідження – процес прийняття рішення.

Мета роботи – алгоритмізація і розробка програмного продукту, що дозволяє автоматизувати процес прийняття рішень.

Метод дослідження – аналітичний.

У першому розділі було розглянуто еволюцію теорії прийняття рішень, процес прийняття рішень, схему процесу прийняття рішень, класифікацію задач прийняття рішень, класифікацію методів прийняття рішень.

Другий розділ присвячено основним теоретичним відомостям. В ньому висвітлено раціональсть рішень. Розглянуто наступні поведінкові моделі: економічної раціональності, обмеженої раціональності Саймона, соціальну модель та модель евристичних суджень і переваг. Також надано детальний опис основних критеріїв прийняття рішень.

Третій розділ описує вибір програмних засобів та мови програмування для реалізації програмного продукту. В ньому описано планування системи і розробка архітектури програми.

SUMMARY

Master qualifying paper "Algorithmization of the process of modeling decision-making problems with a large number of alternatives based on descriptive models": 64 pages, 12 figures, 3 tables., 24 references, 1 supplements.

THE ALTERNATIVE, THE CRITERIA, THE PURPOSE, THE DECISION MAKER, THE THEORY OF DECISION MAKING.

The object of the study is the decision-making process.

The purpose of the work is the algorithmization and development of a software product that allows to automate the decision-making process.

The research method is analytical.

The first section deals with the evolution of decision theory, the decision-making process, the scheme of the decision-making process, the classification of decision-making problems, the classification of decision-making methods.

The second section deals with basic theoretical information. It highlights the rationality of decisions. The following behavioral models are considered: Simon's economic rationality, Simon's limited rationality, the social model, and the model of heuristic judgments and preferences. A detailed description of the main decision-making criteria is also provided.

The third section describes the choice of software and programming language for implementing the software. It describes system design and application architecture development.

ЗМІСТ

ВСТУП

Проблема прийняття рішень, яку в широкому плані можна розглядати як проблема аналізу складних систем, займає все більше місце в сучасній науці.

У тій чи іншій мірі, системи підтримки прийняття рішень присутні у будь-якій системі керування даними. По мірі розвитку, впорядкування і налагодження зв'язків, необхідність розробки і застосування системи підтримки прийняття рішень (СППР) стає особливо актуальною[1].

Швидкість технічної модернізації сучасного світу вимагає обробки та зберігання великих обсягів інформації за короткі проміжки часу. З ровитком науки і техніки відповідно підвищується складність вирішуваних завдань. Все це потребує використання обчислювальної техніки в процесі прийняття рішень. Це стало причиною появи нового класу інформаційних систем – систем підтримки прийняття рішень[2].

Об’єкт дослідження – процес прийняття рішення.

Мета роботи – алгоритмізація і розробка програмного продукту, що дозволяє автоматизувати процес прийняття рішень.

Метод дослідження – аналітичний.

Поняття системи підтримки прийняття рішень з'явилося на початку семидесятих років минулого століття. Одним з головних питань розробки СППР є вибір математичних моделей і методів прийняття рішень, що становлять основу її функціонування.

Таким чином, можна відмітити, що СППР забезпечують наступне:

- допомагають оцінити ситуацію, здійснити вибір критеріїв;
- створюють можливі рішення;
- оцінюють рішення і вибирають краще;
- забезпечують обмін інформацією про стан рішень, що приймаються, і допомагають ухвалювати групові рішення;

- моделюють рішення;
- реалізовують динамічний комп'ютерний аналіз можливих наслідків рішень;
- виводять результати реалізації прийнятих рішень і оцінюють їх.[3]

Прийняття рішень в системі управління пов'язане зі складністю системи, її ієрархією, відсутністю визначеного стану в даний момент, необхідністю приймати до уваги велике число різних факторів і критеріїв, що характеризують варіанти рішень. Виходячи з вищесказаного, можна виявити, що при розробці СППР виникає проблема вибору математичних методів, що найбільш коректно дозволяють відображати структуру системи, для якої приймається рішення, брати до уваги оцінки експертів та якісний характер варіантів вирішення проблеми, враховувати неточність даних[1].

Математична теорія оптимізації, що бурхливо розвивалася останнім часом, створила велику кількість методів, що допомагають при комп'ютерній підтримці ефективно приймати рішення при відомих або випадкових параметрах, а також характеризують процес дослідження. Основні проблеми виникають у тому випадку, коли параметри, що суттєво впливають на очікувані результати виявляються випадковими величинами. Такі ситуації можуть виникати як при відстустності достатньої кількості знань про реалізовані процеси, для яких приймається рішення, так і через участь декількох осіб, що переслідують різні цілі, при прийнятті рішення.

Невизначеності є невід'ємною частиною процесів прийняття рішень, тому що особа, що приймає рішення (ОПР), як правило, не звикла до кількісних оцінок в процесі прийняття рішень, а також не завжди може оцінювати свої рішення на основі математичних методів за допомогою яких-небудь функцій. Ці невизначеності прийнято розділяти на три класи:

- невизначеності, пов'язані з відсутністю достатньої кількості знань про проблему;

- невизначеність, пов'язана з неможливістю чіткого уявлення про реакції довкілля на наші дії;
- невизначеність, пов'язана з неточним розумінням поставлених цілей [1, 3–5].

Прийняття рішення у більшості випадків полягає в створенні можливих альтернатив рішень, їх оцінці і виборі кращої альтернативи.

Прийняти "правильне" рішення – означає вибрати таку альтернативу з числа можливих, в якій з урахуванням усіх різноманітних чинників і суперечливих вимог буде оптимізована загальна цінність, тобто вона в максимальному ступені сприятиме досягненню поставленої мети.

При виборі альтернатив доводиться враховувати велике число вимог, що не відповідають одна одній, і, тому оцінювати варіанти рішень за багатьма критеріями. Нільс Бор помітив: "Є примітивні істини, протиріччя яким явно неправдиво, але існують також і вищі істини, такі, що постулати, що суперечать їм, також справедливі". Суперечність вимог, неоднозначність оцінки ситуацій, помилки у виборі пріоритетів сильно ускладнюють прийняття рішень.

Формалізація методів генерації рішень і їх оцінка є надзвичайно складним завданням. Це завдання стало інтенсивно вирішуватися з виникненням обчислювальної техніки. Рішення цієї задачі в різних сферах сильно залежить від характеристик доступних апаратних і програмних засобів, міри розуміння проблем, по яких приймаються рішення.[6]

1 ПОСТАНОВКА ЗАДАЧІ

Цей розділ присвячено задачі прийняття рішень (ЗПР) – одній з найпоширеніших у будь-якій предметній області. Її рішення зводиться до вибору однієї або декількох кращих альтернатив з деякого набору. Для того, щоб зробити такий вибір, необхідно чітко визначити мету і критерії, по яким проводитиметься оцінка певної вибірки альтернативних варіантів. Вибір методу рішення такої задачі залежить від кількості і якості доступної інформації. Дані, необхідні для здійснення обґрунтованого вибору, можна розділити на чотири категорії: альтернативні варіанти, критерії вибору, переваги, оточення.

1.1 Еволюція теорії прийняття рішень. ЕОМ в прийнятті рішень

У своєму розвитку теорія прийняття рішень пройшла через три стадії.

На першій стадії розвивався дескриптивний підхід до прийняття рішень. Тут зусилля учених були спрямовані на опис процесу вибору рішень людиною в цілях визначення ходу її думок під час даного процесу. В результаті проведених досліджень вони виявили, що більшість людей діють інтуїтивно, непослідовно і суперечать самі собі. Позитивним моментом проведених досліджень в області дескриптивного підходу стало те, що вдалося дати досить чітку відповідь на питання про можливості і поведінку людини при прийнятті рішення.

На другій стадії дослідники розробляли нормативний підхід до прийняття рішень. Проте, поставлене завдання не було виконане, оскільки висунуті теорії, розраховані на людину з дуже потужним над інтелектом, тому вони не знайшли практичного застосування[7].

На третій стадії був розвинений прескриптивний підхід до прийняття рішень. Він виявився найбільш плідним, оскільки пропонував, як повинна

поступати людина з нормальним інтелектом, що бажає напружено і систематизовано обмірковувати усі сторони поставленого завдання. Прескриптивний підхід не забезпечує знаходження оптимального рішення у будь-якій ситуації, але забезпечує вибір такого рішення, в якому відсутні протиріччя і непослідовність. Цей підхід висуває серйозні вимоги до людини, котра прагне засвоїти методи і прийоми теорії прийняття рішень.

Першим поштовхом для застосування ЕОМ в процесі прийняття рішень стала необхідність проведення великого обсягу обчислень з метою отримати певну оцінку після детального перегляду усіх переваг та недоліків кожної альтернативи. Для реалізації усіх поставлених цілей спеціаліст мусив мати широкі знання як в програмуванні, так і в області теорії прийняття рішень[5].

Розроблювана система повинна була мати зручний інтерфейс для того, щоб користувач міг легко змінити усі необхідні параметри обраної моделі задачі прийняття рішень та обрати конкретний алгоритм. Такі системи дозволяли отримувати усю необхідну інформацію для усебічного обґрунтування вибираних рішень.

В наш час існує безліч систем даного типу як для комерційного, так і домашнього використання. Можлтвості сучасних комп’ютерів дозволяють прогамі швидко і правильно обробити усю отриману інформацію та надати користувачеві детальний опис результатів проведених обчислень[8].

1.2 Процес прийняття рішень

Рішення приймається з метою вирішення будь-якої проблеми. Цей процес складається з трьох стадій зображених на рисунку 1.1.

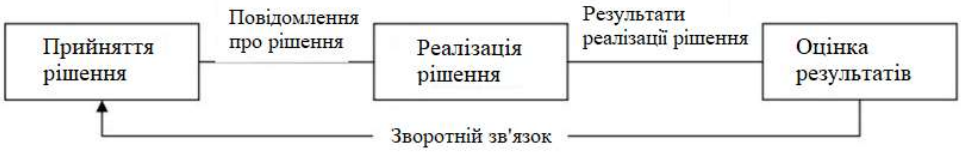

Рисунок 1.1 – Стадії процесу вирішення проблеми

Прийняття рішення можна уявити як процес, що складається з послідовності етапів і процедур, які мають між собою прямі і зворотні зв'язки. На рисунку показана послідовність виконання процедур і ітеративних циклів. Потовщеними лініями показані потоки інформації. Зліва від цих ліній позначені результати поточної процедури, праворуч – накопичені результати ЗПР. Зворотні зв'язки відображають ітеративний циклічний характер залежності між етапами і процедурами. Ітерації в виконанні елементів процесу прийняття рішень обумовлені необхідністю уточнення і коригування даних після виконання подальших процедур [9].

Процес прийняття рішення умовно можна поділити на три етапи.

Етап 1 Постановка задачі.

На даному етапі отримані вхідні дані в першу чергу подаються на виявлення і опис проблемної ситуації. Далі програмі необхідно виявити бажаний час і необхідні ресурси. Якщо отримані дані не відповідають очікуваним, цикл починається знову. Цей процес повторюється до тих пір, поки очікувані результати не стануть приблизно схожі на отримані.

Етап 2 Формування рішень.

Дана стадія починається з аналізу проблемної ситуації. Дані для аналізу алгоритм отримує з попереднього етапу, тільки в тому випадку, якщо вони співпадають очікуваним. Якщо ні, то вхідні дані відправляються на повторну обробку на етап 1, якщо так,то відбувається перехід на формування генетичних ситуацій, яке в свою чергу складається з:

- формування цілей
- визначення обмежень

- генерації рішень
- вимірюванні пріоритетів рішень.

Етап 3 Вибір рішення.

Він починається з визначення допустимих рішень. Далі необхідно сформувати критерій вибору. І в кінці з множини єефективних рішень необхідно обрати єдине.

З інформаційної точки зору в процесі прийняття рішень відбувається зменшення невизначеності. Формулювання проблемної ситуації як би породжує питання “що робити?”. Послідовне виконання процедур призводить до формування відповіді на це питання у вигляді “що і як потрібно робити?”.

Процедури прийняття рішень можуть виконуватися шляхом мислення ОПР і експертів, тобто творчо, неформальним чином, і з застосуванням формальних засобів – математичних методів і ЕОМ. Формальні процедури полягають в проведенні розрахунків за певними алгоритмами з метою аналізу варіантів рішення, оцінки необхідних ресурсів, звуження множини варіантів рішення та ін. Виконання формальних процедур здійснюється ОПР, експертами, технічним персоналом і технічними засобами [10–12].

Подання процесу прийняття рішень як логічно впорядкованої сукупності неформальних і формальних процедур є опис технологічної схеми виконання цього процесу. Такий опис дозволяє структурно упорядкувати процес прийняття рішень і визначити інформаційну модель процесу, на основі якої раціонально організовується збір, обробка та зберігання необхідної інформації.

У процесі прийняття рішень виділяють три етапи: постановка задачі, формування рішень і вибір рішення.

На етапі постановки завдання виконуються наступні процедури:

- виявлення і опис проблемної ситуації;
- визначення часу, наявного для прийняття рішення;
- визначення необхідних і наявних для прийняття рішення ресурсів.

Розглянута схема відображає реальний процес прийняття рішень в спрощеному вигляді. Насправді цей процес є більш складним і не завжди строго виконується за наведеною схемою. Наприклад, при генерації множини альтернативних рішень людина може одночасно враховувати обмеження, принаймні частина з них, і не включати в цю множину рішення, що не відповідають обмеженням[13].

1.3 Схема процесу прийняття рішень

Загальна схема процесу прийняття рішень включає наступні основні етапи.

Етап 1 Попередній аналіз проблеми. На цьому етапі визначаються:

- головні цілі;
- рівні розгляду, елементи і структура системи, типи зв'язків;
- підсистеми, використовувані ними основні ресурси і критерії якості функціонування підсистем;
- основні протиріччя, вузькі місця і обмеження.

Етап 2 Постановка завдання. Постановка конкретної ЗПР включає:

- формулювання завдання;
- визначення типу завдання;
- визначення множинаі альтернативних варіантів і основних критеріїв для вибору з них найкращих;
- вибір методу рішення ЗПР.

Етап 3 Отримання початкових даних. На цьому етапі встановлюються способи виміру альтернатив. Це або збір кількісних (статистичних) даних, або методи математичного або імітаційного моделювання, або методи експертної оцінки. У останньому випадку необхідно вирішити завдання формування групи експертів, проведення експертних опитувань, попереднього аналізу експертних оцінок[14].

Етап 4 Рішення ЗПР із залученням математичних методів і обчислювальної техніки, експертів і особи, що приймає рішення. На цьому етапі здійснюються математична обробка початкової інформації, її уточнення і модифікація у разі потреби. Обробка інформації може виявитися досить трудомісткою, при цьому може виникнути необхідність здійснення декількох ітерацій і бажання застосувати різні методи для вирішення завдання. Тому саме на цьому етапі виникає потреба в комп'ютерній підтримці процесу прийняття рішень, яка виконується за допомогою автоматизованих систем прийняття рішень.

Етап 5 Аналіз і інтерпретація отриманих результатів. Отримані результати можуть виявитися незадовільними і зажадати змін в постановці ЗПР. В цьому випадку необхідно буде повернутися на етап 2 або етап 1 і пройти заново весь шлях. Рішення ЗПР може займати досить тривалий проміжок часу, протягом якого оточення завдання може змінитися і зажадати коригувань в постановці завдання, а також у вихідних даних (наприклад, можуть з'явитися нові альтернативи, що вимагають введення нових критеріїв). Завдання прийняття рішень можна розділити на статичні і динамічні. До перших відносяться завдання, які не потребують багаторазового рішення через короткі інтервали часу. До динамічних відносяться ЗПР, які виникають досить часто. Отже, ітераційний характер процесу прийняття рішень можна вважати закономірним, що підтверджує необхідність створення і використання ефективних систем комп'ютерної підтримки. ЗПР, що вимагають одного циклу, можна швидше вважати винятком, ніж правилом [15].

1.4 Класифікація задач прийняття рішень

Завдання прийняття рішень відрізняються великим різноманіттям, класифікувати їх можна за різними ознаками, що характеризує кількість і якість доступної інформації. У загальному випадку завдання прийняття рішень можна представити таким набором інформації:

<T, A, K, X, F, G, D>,

де T – постановка задачі (наприклад, вибрати кращу альтернативу або впорядкувати весь набір);

A – множина допустимих альтернативних варіантів;

K – множина критеріїв вибору;

X – множина методів вимірювання переваг (наприклад, використання різних шкал);

F – відображення множини допустимих альтернатив в множині критеріальних оцінок (результати);

G – система переваг експерта;

D – вирішальне правило, що відбиває систему переваг.

Будь-який з елементів цього набору може служити класифікаційним ознакою прийняття рішень.

Розглянемо традиційні класифікації:

- по виду відображення F. Відображення множинаі A і K може мати детермінований характер, імовірнісний або невизначений вид, відповідно до якого завдання прийняття рішень можна розділити на завдання в умовах ризику і завдання в умовах невизначеності.
- потужність множини K. Множина критеріїв вибору може містити один елемент або декілька. Відповідно до цього завдання прийняття рішень можна розділити на завдання зі скалярним критерієм і завдання з векторним критерієм (багатокритеріальне прийняття рішень).
- тип системи G. Уподобання можуть формуватися однією особою або колективом, в залежності від цього завдання прийняття рішень можна класифікувати на завдання індивідуального прийняття рішень і завдання колективного прийняття рішень[16].

Завдання прийняття рішень в умовах визначеності. До цього класу належать задачі, для вирішення яких є достатня і достовірна кількісна інформація. В цьому

випадку з успіхом застосовуються методи математичного програмування, суть яких полягає в знаходженні оптимальних рішень на базі математичної моделі реального об'єкта. Основні умови застосовності методів математичного програмування наступні:

- завдання має бути добре формалізоване, тобто є адекватна математична модель реального об'єкта.
- існує деяка єдина цільова функція (критерій оптимізації), що дозволяє судити про якість розглянутих альтернативних варіантів.
- є можливість кількісної оцінки значень цільової функції.
- завдання має певні ступені свободи (ресурси оптимізації), тобто деякі параметри функціонування системи, які можна довільно змінювати в деяких межах з метою поліпшення значень цільової функції.

Завдання в умовах ризику. У тих випадках, коли можливі результати можна описати за допомогою деякого імовірнісного розподілу, отримуємо завдання прийняття рішень в умовах ризику. Для побудови розподілу ймовірностей необхідно або мати в розпорядженні статистичні дані, або залучати знання експертів. Зазвичай для вирішення завдань цього типу застосовуються методи теорії одновимірної або багатовимірної корисності. Ці завдання займають місце на кордоні між завданнями прийняття рішень в умовах визначеності та невизначеності. Для вирішення цих завдань залучається вся доступна інформація (кількісна та якісна).

Завдання в умовах невизначеності. Ці завдання мають місце тоді, коли інформація, необхідна для прийняття рішень, є неточною, неповною, некількісною, а формальні моделі досліджуваної системи або занадто складні, або відсутні. У таких випадках для вирішення завдання зазвичай залучаються знання експертів. На відміну від підходу, прийнятого в експертних системах, для вирішення ЗПР знання експертів зазвичай виражені у вигляді деяких кількісних даних, званих уподобаннями.

Вибір і нетривіальність завдань прийняття рішень. Слід зазначити, що однією з умов існування завдання прийняття рішень є наявність декількох допустимих альтернатив, з яких слід вибрати в деякому сенсі кращу. При наявності однієї альтернативи, що задовольняє фіксованими умовами або обмеженням, завдання прийняття рішень не має місця.

Задача прийняття рішень називається тривіальним, якщо воно характеризується виключно одним критерієм К і всім альтернативам Аі приписані конкретні числові оцінки відповідно до значень зазначеного критерію [1–4, 17].

1.5 Класифікація методів прийняття рішень

Існує множина класифікацій методів прийняття рішень, заснованих на застосуванні різних ознак. У табл. 1.1 приведена одна з можливих класифікацій, ознаками якої є зміст і тип одержуваної експертної інформації.

Використовуваний принцип класифікації дозволяє досить чітко виділити чотири великі групи методів, причому три групи відносяться до прийняття рішень в умовах визначеності, а четверта - до прийняття рішень в умовах невизначеності. З множини відомих методів і підходів до прийняття рішень найбільший інтерес представляють ті, які дають можливість враховувати багатокритеріальність і невизначеність, а також дозволяють здійснювати вибір рішень з множин альтернатив різного типу при наявності критеріїв, що мають різні типи шкал вимірювання (ці методи відносяться до четвертої групи)[18].

У свою чергу, серед методів, що утворюють четверту групу, найбільш перспективними є декомпозіційні методи теорії очікуваної корисності, методи аналізу ієрархій і теорії нечітких множин. Даний вибір визначений тим, що ці методи в найбільшій мірі задовольняють вимогам універсальності, врахування багатокритеріальності вибору в умовах невизначеності з дискретної або

безперервної безлічі альтернатив, простоти підготовки і переробки експертної інформації.

Таблиця 1.1 – Класифікація методів прийняття рішень

№ п/п	Зміст інформації	Тип інформації	Метод прийняття рішень
1	Експертна інформація не потрібна		Метод домінування Метод на основі глобальних критеріїв
2	Інформація про переваги на множині критеріїв	Якісна інформація Кількісна оцінка перевагу критеріїв Кількісна інформація про заміщення	Порівняння різниць критеріальних оцінок Методи "ефективність-вартість" Методи згортки на ієрархії критеріїв Методи порогів Методи ідеальної точки Метод кривих байдужості Методи теорії цінності
3	Інформація про перевагу альтернатив	Оцінка переваги парних порівнянь	Методи математичного програмування Лінійна і нелінійна згортка при інтерактивному способі визначення її параметрів
4	Інформація про переваги на множині критеріїв та про наслідки альтернатив	Відсутність інформації про переваги; Кількісна і / або інтервальна інформація про наслідки; Якісна інформація про переваги та кількісна про наслідки; Якісна (порядкова) інформація про переваги та наслідки; Кількісна інформація про переваги та наслідки;	Методи з дискретизацією невизначеності Методи прийняття рішень в умовах ризику і невизначеності на основі глобальних критеріїв Метод аналізу ієрархій Методи теорії нечітких множин Метод практичного прийняття рішень Методи вибору статистично ненадійних рішень Методи кривих байдужості для прийняття рішень в умовах ризику і невизначеності Методи дерев рішень

Охарактеризувати досить повно всі методи прийняття рішень, які стосуються четвертої групи, в рамках даної роботи неможливо, тому в подальшому розглядаються тільки три підходи до прийняття рішень в умовах невизначеності,

які отримали найбільш широке втілення в системах комп'ютерної підтримки, а саме: підходи, засновані на методах теорії корисності, аналізу ієрархій і теорії нечітких множин [19].

2 ОСНОВНІ ТЕОРЕТИЧНІ ВІДОМОСТІ

2.1 Раціональність рішення

Найбільш часто під раціональністю в прийнятті рішення розуміють спосіб досягнення поставленої мети: якщо шлях до мети є правильним, рішення вважається раціональним. Але ця найпростіша перевірка на раціональність на практиці досить складна. Адже дуже важко відрізнити шлях від мети, оскільки вона може бути початком шляху до якоїсь іншої мети. Цей взаємозв'язок часто називають ланцюжком «шлях-мета», іншими словами, ієрархією.

Г. Саймон звернув увагу на те, що ієрархія «шлях-мета» рідко являє собою повністю однозначну і взаємопов'язаний ланцюг. Часто взаємозв'язок між діяльністю організації і кінцевими цілями виявляється неясною, кінцеві цілі сформульовані в повному обсязі, існують внутрішні конфлікти і суперечності між кінцевими цілями і між засобами їх досягнення.

Варіативність поняття раціональності. Одним із способів прояснити раціональність ланцюга «шлях-мета» стало введення різних типів поняття раціональності. Об'єктивна раціональність може бути визначена як застосовна до тих рішень, які максимізують дані цінності в даній ситуації[6,20].

Суб'єктивна раціональність може мати місце в тому випадку, коли рішення максимізує досягнення щодо наявних у даного суб'єкта знань. Усвідомлена раціональність може бути застосовна до рішень, при яких орієнтація засобів досягнення і самої кінцевої мети здійснюється свідомо.

Рішення є цілеспрямовано раціональним, коли уточнення засобів досягнення мети цілеспрямовано здійснюється окремою особою або групою осіб; рішення є організаційно раціональним в тому випадку, якщо воно є ціллю групи людей; рішення є особисто раціональним, якщо воно спрямоване на досягнення особистих цілей особи, що приймає рішення.

2.2 Поведінкові моделі прийняття рішень

Існує багато моделей, які описують поведінкові аспекти прийняття рішень і намагаються визначити, наскільки раціональними є дії осіб, що приймають рішення. Ці моделі варіюють від повної раціональності, як у випадку моделі економічної раціональності, до повної ірраціональності, як у випадку соціальної моделі. Проміжне становище займають модель обмеженої раціональності Саймона і модель евристичних суджень і переваг, запозичена зі спеціальної психології[21].

2.2.1 Модель економічної раціональності

Ця класична економічна модель, в якій особа, яка приймає рішення, є повністю раціональним у всіх планах, його рішення ґрунтуються на результатах економічного аналізу. При прийнятті рішення передбачається виконання наступних умов:

- рішення повністю раціонально в сенсі ланцюга «шлях-мета»;
- є повна і несуперечлива система переваг і критеріїв вибору, яка дозволяє зробити вибір з наявних варіантів;
- всі можливі варіанти відомі;
- не існує проблеми складності розрахунків, необхідних для визначення найкращих альтернатив;
- імовірнісні розрахунки зрозумілі і не викликають побоювань.

При виконанні цих умов особи, які приймають рішення, завжди прагнуть максимізувати результати діяльності підприємства.

Ще Пітерс і Уотермен відзначали: «Кількісний, раціоналістичний підхід домінує в школах бізнесу. Для нього характерний пошук абстрактних, аналітичних обґрунтувань будь-яких рішень. Він досить точний, щоб стати небезпечно помилковим і, на думку деяких, вже призвів нас до серйозних помилок». Однак

критики раціоналістичної моделі не закликають повністю відмовитися від неї. Ця модель вносить істотний внесок в ефективне прийняття рішень. Наприклад, процвітаючі компанії, такі, як «Проктор енд Гембл», віддають перевагу раціоналістичним підходам до прийняття рішень і використання відповідних кількісних розрахунків. Суть обмеженості моделі економічної раціональності полягає в тому, що ця модель не є єдино правильною і можливою, зловживання нею може легко привести до прийняття помилкових рішень[21].

2.2.2 Модель обмеженої раціональності Саймона.

Г. Саймон запропонував більш реалістичну модель економічної раціональності:

- при виборі альтернатив менеджери прагнуть до такого варіанту, який задовільний або «досить гарний»;
- менеджери усвідомлюють, що світ, який вони сприймають, є всього лише спрощеною моделлю реального світу. Вони задоволені цією моделлю, оскільки вважають, що реальний світ неозорий;
- основним завданням менеджери вважають відповідність деяким критеріям, а не отримання максимальних результатів, тому вони можуть зробити вибір без попереднього виявлення всіх можливих альтернатив;
- оскільки вони вважають реальний світ безмежним, менеджери можуть приймати рішення на підставі відносно простих емпіричних правил, за допомогою ремісничих прийомів або в силу звички[22].

2.2.3 Соціальна модель

Повною протилежністю моделі економічної раціональності є соціально-психологічна модель. Більшість сучасних психологів вважають, що соціально-психологічні чинники роблять серйозний вплив на поведінку при прийнятті рішень.

Так, впливу з боку суспільства можуть змусити менеджерів до прийняття ірраціональних рішень. Добре відомі експерименти, які свідчать про те, що більше 30% людей схильні до конформізму, тобто здатні приймати чуже і свідомо невірна думка, погоджуватися з рішенням, яке вважають неправильним.

Крім того, встановлено, що близько 3% людей прагнуть до ризику, а багато людей, які приймають рішення, наполягають на неправильному, але прийнятому раніше, варіанті рішення[23].

2.2.4 Модель евристичних суджень і переваг

Модель евристичних суджень і переваг виникла в основному під впливом досліджень Канемана і Тверскі, теоретичних прихильників психологічного підходу до прийняття рішень. Відповідно до цієї моделі особи, які приймають рішення, у багатьох випадках покладаються насамперед на евристику - інтуїцію і минулий досвід. Метод евристичних міркувань зменшує інформаційні потреби приймають рішення і на практиці допомагає полегшити прийняття рішень. Евристичні принципи спрощують рішення і допомагають тим, хто їх приймає, але їх використання може призвести до помилок і прорахунків[23].

2.3 Критерії прийняття рішень

Критерій прийняття рішень – це функція, що виражає переваги особи, що приймає рішення (ОПР), і визначає правило, за яким вибирається прийнятний або оптимальний варіант рішення.

Будь-яке рішень в умовах неповної інформації приймається в з урахуванням кількісних характеристик ситуацій, в якій приймаються рішення. Найбільш часто приймаються наступні критерії прийняття Севіджа, критерій Гурвіца, критерій Ходжа-Лемана, критерій Гермейера, оптимістичний критерій. Відповідно до рішень: мінімаксний критерій, критерій Байеса-Лапласа, критерій множення, мінімаксний критерій Байеса-Лапласа.

Ці критерії можна використовувати по черзі, причому після обчислення їх значень серед декількох варіантів доводиться довільним чином виділяти деяке остаточне рішення. Що дозволяє, по-перше, краще проникнути в усі внутрішні зв'язки проблеми прийняття рішень і, по-друге, послабити вплив суб'єктивного фактора[24].

2.3.1 Критерій Байеса

Цей критерій використовується в припущенні, що ймовірності qj станів природи Пj відомі. В якості показника ефективності чистої стратегії Ai використовується середньозважений виграш при стратегії Ai з вагами q1, ..., qn, тобто величина

$$b_i = \sum_{j=1}^{n} a_{ij} q_j, \qquad (i = \overline{1, m})$$

Оптимальною по Байеса чистої стратегією є стратегія з максимальним показником ефективності. Ціна гри в цьому випадку визначається за формулою:

$$b = \max_{1 \le i \le m} b_i = \max_{1 \le i \le m} \sum_{j=1}^{n} a_{ij} q_j$$

Аналогічно можна знайти оптимальну стратегію по Байеса, використовуючи формулу:

$$R_i = \sum_{j=1}^{n} r_{ij} q_j, \qquad (i = \overline{1, m})$$

і матрицю ризиків. В цьому випадку середній ризик слід мінімізувати. Однак, слід зауважити, що стратегія, максимізує середній виграш, збігається зі стратегією, що мінімізує середній ризик[10].

2.3.2 Критерій Лапласа

Якщо гравець А не має об'єктивної інформації про ймовірності q_j станів природи $П_j$ і вважає в рівній мірі правдоподібними їхні капітали, то їх ймовірності вважають однаковими і рівними 1/n. цей прийом називають принципом недостатнього підстави Лапласа.

Звідси випливає і критерій Лапласа, відповідно з яким оптимальною вважається чиста стратегія, що забезпечує максимальний середній виграш гравця А за однакової кількості всіх ймовірностей. В цьому випадку показники ефективності кожної чистої стратегії розраховуються за формулою:

$$l_i = \frac{1}{n}\sum_{j=1}^{n} a_{ij}, \qquad (i = \overline{1,m})$$

А ціна гри дорівнює

$$l = \max_{1 \le i \le m} l_i = \max_{1 \le i \le m} \frac{1}{n}\sum_{j=1}^{n} a_{ij}$$

При використанні критеріїв Байєса і Лапласа передбачається, що ситуація, в якій приймається рішення, характеризується наступними обставинами:

- ймовірності появи станів Пj відомі і не залежать від часу.
- рішення реалізується (теоретично) нескінченно багато разів.
- для малого числа реалізацій рішення допускається деякий ризик.

При досить великій кількості реалізацій середнє значення поступово стабілізується. Тому при повній (нескінченної) реалізації будь-якої ризик практично виключений[10].

2.3.3 Критерій Вальда

У разі, якщо ймовірності станів природи невідомі і немає можливості отримати про них якусь статистичну інформацію, при визначенні оптимального рішення можна використовувати критерій Вальда.

Критерій Вальда є критерієм крайнього песимізму, тому що тут гравець А виходить з припущення, що природа П «діє» проти нього найгіршим чином, тобто реалізує такі стани Пj, При яких величина його виграшу приймає найменше значення.

Показники ефективності кожної чистої стратегії розраховуються за формулою:

$$w_i = \min_{1 \le j \le n} a_{ij}, \qquad (i = \overline{1,m}).$$

Оптимальною за критерієм Вальда вважається та чиста стратегія, показник ефективності якої буде максимальним, тобто забезпечується максимін

$$w = \max_{1 \le i \le m} w_i = \max_{1 \le i \le m} \min_{l \le j \le n} a_{ij}$$

Критерій Вальда часто також називають максимінним критерієм.

Обрані таким чином варіанти повністю виключають ризик. Це значає, що приймаючий рішення не може зіткнутися з гіршим результатом, ніж той, на який він орієнтується.

Застосування критерію Вальда буває виправдано, якщо ситуація, в якій приймається рішення, наступна:

- про можливість появи зовнішніх станів $П_j$ нічого не відомо;
- доводиться рахуватися з появою різних зовнішніх станів Пj;
- рішення реалізується тільки один раз;
- необхідно виключити який би то не було ризик[10].

2.3.4 Критерій Севіджа

Критерій Севіджа, як і критерій Вальда, є критерієм крайнього песимізму, бо і тут гравець А виходить з припущення, що природа реалізує найнесприятливіші для нього стани. Критерій Севіджа рекомендує вибирати в якості оптимальної ту чисту стратегію, при якій мінімізується величина максимального ризику.

Таким чином, показник ефективності визначається як величина максимального ризику:

$$s_i = \max_{1 \le j \le n} r_{ij}, \quad (i = \overline{1,\ m}).$$

А ціна гри дорівнює

$$s = \min_{1 \le i \le m} s_i = \min_{1 \le i \le m} \max_{1 \le j \le n} r_{ij}.$$

При використанні критерію Севіджа ситуація, в якій приймається рішення, повинна задовольняти тим же умовами, що і при застосуванні критерію Вальда[10].

2.3.5 Критерій Гурвіца

Зайняти більш урівноважену позицію, яка знаходиться десь між точкою зору крайнього оптимізму та крайнього песимізму, пропонує критерій Гурвіца. Його також часто називають критерієм песимізму-оптимізму.

В області чистих стратегій показник ефективності визначається з умови:

$$g_i = \gamma \min_{1 \le j \le n} a_{ij} + (1 - \gamma) \max_{1 \le j \le n} a_{ij}, \ (i = \overline{1,\ m},\ 0 \le \gamma \le 1)$$

Оптимальною по Гурвіцу вважається та чиста стратегія, показник ефективності якої приймає найбільше значення

$$g = \max_{1 \le i \le m} g_i = \max_{1 \le i \le m} \left\{ \gamma \min_{1 \le j \le n} a_{ij} + (1 - \gamma) \max_{1 \le j \le n} a_{ij} \right\}, 0 \le \gamma \le 1.$$

Параметр γ обирається з суб'єктивних міркувань, тому що на практиці дуже важко знайти кількісну характеристику для тих часток оптимізму і песимізму, які присутні при прийнятті рішення. Найчастіше γ вважають рівним 0,5.

При γ = 1 критерій Гурвіца перетворюється в критерій Вальда (крайнього песимізму).

При γ = 0 – в критерій крайнього оптимізму, або критерій «азартного гравця», що робить ставку на те, що результат гри буде для нього найбільш сприятливим:

$$g = \max_{1 \le i \le m} g_i = \max_{1 \le i \le m} \max_{1 \le j \le n} a_{ij}.$$

При 0 < γ <1 виходить щось середнє між точкою зору крайнього оптимізму та крайнього песимізму.

Критерій Гурвіца застосовується в разі, коли:

- про можливості появи стану $П_j$ нічого невідомо;
- з появою стану $П_j$ необхідно рахуватися;
- реалізується тільки мала кількість рішень;
- допускається деякий ризик[10].

2.3.6 Критерій Ходжа-Лемана

Цей критерій спирається одночасно на критерій Вальда і критерій Байеса-Лапласа. За допомогою параметра ν виражається ступінь довіри до використовуваного розподілу ймовірностей, а коефіцієнт (1-ν) кількісно характеризує ступінь песимізму гравця А. Чим більше довіри гравця А цього розподілу ймовірностей станів природи, тим менше песимізму і навпаки.

Якщо довіра велика, то домінує критерій Байеса-Лапласа, в іншому випадку – критерій Вальда, тобто показник ефективності чистої стратегії A_i дорівнює:

$$h_i = \nu \sum_{j=1}^{n} a_{ij} q_i + (1 - \nu) \min_{1 \le j \le n} a_{ij}, \quad (i = \overline{1,\ m},\ 0 \le \nu \le 1).$$

Стратегія з максимальним показником ефективності є оптимальною. Ціна гри визначається за формулою:

$$h = \max_{1 \le i \le m} h_i = \max_{1 \le i \le m} \left\{ v \sum_{j=1}^{n} a_{ij} q_i + (1 - \nu) \min_{1 \le j \le n} a_{ij}, \right\}, 0 \le v \le 1.$$

При ν = 1 критерій Ходжа-Лемана переходить в критерій Байеса-Лапласа, а при ν = 0 стає критерієм Вальда.

Вибір ν суб'єктивний оскільки визначити ступінь достовірності будь-якої функції розподілу досить складно.

Для застосування критерію Ходжа-Лемана бажано, щоб ситуація в якій приймається рішення, задовольняла властивостям:

ймовірності появи стану $П_j$ невідомі, але деякі припущення про розподіл ймовірностей можливі;

прийняте рішення теоретично допускає нескінченно багато реалізацій;

при малих числах реалізації допускається деякий ризик[10].

2.3.7 Критерій множення

Критерій множення викори стовується в тих випадках, коли всі елементи платіжної матриці позитивні, тобто $a_{ij} < 0$, $i = \overline{1,m}, i = \overline{1,n}$. Якщо ця умова порушується, то можна перейти до строго позитивних значень за допомогою перетворення $a_{ij} + a$ при відповідному чином підібраному $a > 0$. Результат при цьому буде, природно, залежати від a.

При використанні цього критерію показник ефективності кожної чистої стратегії визначається за формулою:

$$p_i = \prod_{j=1}^{n} a_{ij}, \ \ (i = \overline{1,m}).$$

Оптимальною за критерієм множення буде та чиста стратегія, показник ефективності якої буде найбільшим.

$$p = \max_{1 \le i \le n} p_i = \max_{1 \le i \le n} \prod_{j=1}^{n} a_{ij}.$$

Застосування цього критерію зумовлюється такими обставинами:

- ймовірності появи стану $П_j$ невідомі;
- з появою кожного з станів $П_j$ окремо необхідно рахуватися;
- критерій застосуємо і при малому числі реалізацій рішення;
- певний ризик допускається[10].

2.3.8 Критерій Гермейера

При використанні цього критерію вихідна платіжна матриця замінюється матрицею Гермейера. Кожен елемент матриці ми домножуємо на відповідну ймовірність j стану природи.

Для матриці виграшів

$$v_{i_0} = \max_{1 \le i \le m} \min_{1 \le j \le n} \{ v_{ij} q_j \},$$

Для матриці втрат

$$v_{i_0} = \min_{1 \le i \le m} \max_{1 \le j \le n} \{ v_{ij} q_j \}.$$

Критерій Гермейера застосовують гравці не схильні до ризику, тому що кожна стратегія оцінюється з точки зору мінімуму гарантованого результату.

Стан природи утворює мінімум, а потім гравець обирає стратегію, яка принесе йому максимальний результат. Тобто він захищає себе[10].

2.3.9 Мінімаксний критерій

Мінімаксний критерій використовує оціночну функцію, що відповідає песимістичній позиції, формалізованою співвідношенням:

$$Z_{mm} = \max_{1 \le i \le m} e_{ij}$$

$$e_{ij} = \min_{1 \le j \le n} e_{ij}$$

$$E_0 = \left\{ E_{i0} \middle| E_{i0} \in E \wedge (e_{i0} = \max_{1 \le i \le m} e_{ij} \min_{1 \le j \le n} e_{ij}) \right\}$$

Правило вибору рішення відповідно до мінімаксного критерію інтерпретується наступним чином. Матриця рішень { mn } доповнюється ще одним стовпцем з найменших результатів e_i кожного рядка. При прийнятті рішення слід вибрати такі варіанти E_{i0}, рядки яких відповідають найбільшим значенням e_i цього стовпчика. Обрані таким чином варіанти повністю виключають ризик, оскільки особа, яка приймає рішення, орієнтованаа на песимістичну позицію, що не дозволяє отримати найгірший результат.

1) про можливість появи зовнішніх станів (умов) B_j нічого невідомо (наприклад, невідомі ймовірності появи станів $П_j$);

2) доводиться рахуватися з появою різних зовнішніх станів $П_р$;

3) рішення реалізується тільки один раз;

4) необхідно виключити будь-який ризик (неприпустимо отримання результату нижче значення п.2[10].

2.3.10 Оптимістичний критерій

З його допомогою визначається стратегія, що максимізує максимальні виграші для кожного стану природи. Найкращим визнається рішення, при якому досягається максимальний виграш, рівний

$$m = \max_{1 \le i \le m} \min_{1 \le j \le n} e_{ij}.$$

Цей критерій характеризується крайньою оптимістичній позицією відношнняи ЛПР до невизначеності економічного результату, тобто позицією "азартного гравця", впевненого в тому, що йому повинно пощастити, і тому схильного до найбільш ризикованих виборів. В рамках такого підходу при порівнянні альтернативних рішень за основу приймаються їх найсприятливіші результати серед можливих ситуацій для "зовнішніх" подій, які залежать від ОПР. Вибирається рішення, стосовно якого такий самий сприятливий результат (для можливих ситуацій розвитку "зовнішніх" подій) буде найбільшим.

Уявімо формальні процедури вибору рішення за цим критерієм. До матриці корисностей дописується додатковий стовпець. Його елементи визначаються як найкращі (найбільші) можливі кінцеві результати при відповідному рішенні (по рядках матриці). Потім з усіх елементів такого додаткового стовпця знаходиться найкращий (найбільший). За таким елементом і визначають оптимальне рішення: їм буде рішення відповідного рядка матриці корисностей[10].

3 ВИБІР ІНСТРУМЕНТАРІЮ РОЗРОБКИ І ПЛАНУВАННЯ СИСТЕМИ

3.1 Вимоги до розроблюваної системи

Однією з вимог, що пред'являються до розроблюваної системі, є створення та налаштування зручного для користувача інтерфейсу, що забезпечує легке сприйняття і обробку інформації, а також мінімізує власні операції.

Для забезпечення взаємодії між ЕВМ та користувачем необхідно розробити зручний інтерфейс – об'єкт, що забезпечує високу інформативність виведеної на екран інформації, організує зручність її виведення і обробки користувачем автоматизованої системи.

Таким чином, методами досягнення поставлених завдань можна вважати: створення та налаштування призначеного для користувача інтерфейсу; генерація звітних форм.

3.2 Моделювання логіки програми

З метою моделювання логіки роботи і дій, що виконуються в програмі, побудуємо модель у вигляді діаграми діяльності, наведеною на рисунку 3.1.

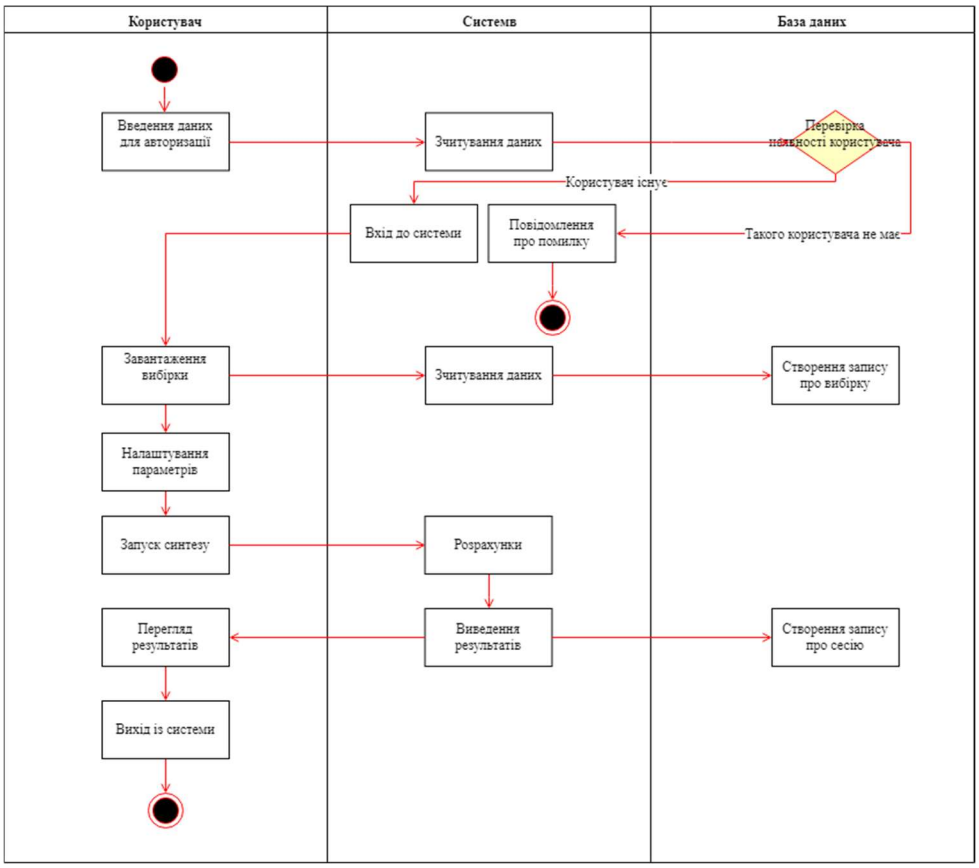

Рисунок 3.1 – Діаграма діяльності в нотації UML 2.0

Як видно з рисунку 3.1, робота із системою починається з введення облікових даних. Після успішної перевірки даних користувач може перейти до завантаження даних та підбору параметрів. Коли вхідні дані обрано користувач може запустити синтез. Після закінчення обрахунків система виводить результат та створює запис про успішну сесію. Після перегляду результату користувач може завершити роботу із системою.

3.3 Вибір мови програмування

C# (вимовляється як "сі шарп") – простий, сучасна об'єктно-орієнтована і типобезпечна мова програмування. C# відноситься до широко відомого сімейства мов C, тому здасться добре знайомою кожному, хто працював з C, C ++, Java або JavaScript. Тут представлений огляд основних компонентів мови.

C# є об'єктно-орієнтованою мовою, але підтримує також і компонентно-орієнтоване програмування. Розробка сучасних додатків все більше тяжіє до створення програмних компонентів у формі автономних і самоописуючих пакетів, що реалізують окремі функціональні можливості. Важлива особливість таких компонентів – це модель програмування на основі властивостей, методів і подій. Кожен компонент має атрибути, які надають декларативні відомості про компоненті, а також вбудовані елементи документації. C# надає мовні конструкції, безпосередньо підтримуючі таку концепцію роботи. Завдяки цьому C# відмінно підходить для створення і застосування програмних компонентів.

Ось лише кілька функцій мови C#, що забезпечують надійність і стійкість додатків: прибирання сміття – автоматично звільняє пам'ять, зайняту знищеними і невикористовуваними об'єктами; обробка виключень надає структурований і розширюваний спосіб виявляти і обробляти помилки; сувора типізація мови не дозволяє звертатися до неініціалізованих змінних, виходити за межі індексованих масивів або виконувати неконтрольоване приведення типів.

У C# існує єдина система типів. Всі типи C#, включаючи типи-примітиви, такі як int і double, успадковують від одного кореневого типу object. Таким чином, всі типи використовують загальний набір операцій, і значення будь-якого типу можна зберігати, передавати і обробляти схожим чином. Крім того, C# підтримує призначені для користувача посилальні типи і типи значень, дозволяючи як динамічно виділяти пам'ять для об'єктів, так і зберігати спрощені структури в стеці.

Щоб забезпечити сумісність програм і бібліотек C# при подальшому розвитку, при розробці C# багато уваги було приділено управлінню версіями. Багато мов програмування обходять увагою це питання, і в результаті програми на цих мовах ламаються частіше, ніж хотілося б, при виході нових версій залежних бібліотек. Питання управління версіями істотно вплинули на такі аспекти розробки C#, як роздільні модифікатори virtual і override, правила вирішення перевантаження методів і підтримка явного оголошення членів інтерфейсу.

Завдяки вище перерахованим якостям C# було обрано в якості мови розробки даного проекту[7].

У таблиці 3.1 наведено порівняння обраної мови програмування із іншими найпопулярнішими мовами програмування.

Таблиця 3.1 – Порівняння мов програмування

Критерії порівняння	Мови програмування				
	C#	C++	Java	Python	Scala
Наявність літератури та простота вивчення	++	++	++	++	±
Підтримка парадигми модульного програмування	+	+	+	+	±
Стандартні бібліотеки шаблонів	+	+	+	+	±
Багатоплатформність	++	+	±	+	±
Продуктивність роботи	++	++	+	+	+
Узагальнення алгоритмів	+	+	+	+	+
Масштабованість	++	+	±	–	+
Паралельне програмування	++	++	++	+	++
Простота розроблення графічного інтерфейсу	+	±	++	+	+

Як видно з таблиці не дивлячись на те, що мова програмування C# виникла вже достатньо давно, вона і зараз достатньо популярна та активно продовжує розвиватися. Більш того саме вона найбільш вдалий вибір, якщо майбутній програмний продукт (ПП) повинен реалізовувати багатоплатформність [45].

Мова програмування C# працює на базі платформи .NET Framework 4.5.1 (актуальна версія).

Microsoft .NET (читається *дот-нет*) – програмна технологія, запропонована фірмою Microsoft як платформа для створення як звичайних програм, так і веб-застосунків. Багато в чому є продовженням ідей та принципів, покладених в технологію Java. Однією з ідей .NET є сумісність служб, написаних різними мовами. Хоча ця можливість рекламується Microsoft як перевага .NET, платформа Java має таку саму можливість.

Кожна бібліотека (збірка) в .NET має свідчення про свою версію, що дозволяє усунути можливі конфлікти між різними версіями збірок.

.NET — крос-платформова технологія, в цей час існує реалізація для платформи Microsoft Windows, FreeBSD (від Microsoft) і варіант технології для ОС Linux в проекті Mono (в рамках угоди між Microsoft з Novell), DotGNU.

Захист авторських прав відноситься до створення середовищ виконання (CLR — Common Language Runtime) для програм .NET. Компілятори для .NET випускаються багатьма фірмами для різних мов вільно.

.NET поділяється на дві основні частини – середовище виконання (по суті віртуальна машина) та інструментарій розробки.

Середовища розробки .NET-програм: Visual Studio .NET (C++, C#, J#), SharpDevelop, Borland Developer Studio (Delphi, C#) тощо. Середовище Eclipse має додаток для розробки .NET-програм. Застосовні програми також можна розроблювати в текстовому редакторі та використовувати консольний компілятор[1].

C # /. NET розробник – це програміст, який використовує в своїй роботі технології платформи .NET. Платформа Microsoft .NET Framework складається з великої кількості інструментів для розробки і технологій, використовуючи які розробник може створювати різні типи додатків, від звичайних настільних додатків і сайтів, закінчуючи рішеннями для мобільних платформ і комп'ютерними іграми. В основі платформи Microsoft .NET Framework лежить мова програмування C #.

Мова програмування C # більше десяти років займає лідируючі позиції у всіх рейтингах мов програмування. Так, як ринок праці активно розвивається, програмісти, які добре знають C # і технології .NET, є дуже затребуваними. .NET розробники здатні розвивати логічну послідовність команд для зв'язку з мережами, додатками і базами даних. Від них вимагається знання об'єктно-орієнтованого проектування та програмування з використанням систем, баз даних, а також мов програмування, які розробляють програмні додатки з .NET Framework. Сюди

входять знання і навички програмування на мові C #, XML і створення баз даних додатків, таких як Microsoft SQL Server.

Вимоги до C # /. NET розробнику:

- володіння мовою програмування C#;
- володіння ООП;
- знання технологій роботи з базами даних;
- практичний досвід роботи з MS SQL Server;
- навички використання Transact-SQL;
- знання Entity Framework;
- базовий рівень знань та досвід роботи з .NET Framework;
- знання технології WCF;
- базовий рівень знань технології ASP.NET MVC;
- знання та вміння застосовувати засоби колективної роботи, вміння читати і розуміти чужий код;
- англійська мова на рівні читання технічної документації (поглиблені знання будуть перевагою);
- знання основ командної розробки SCRUM або Agile[7].

3.4 Вибір середовища розробки

SharpDevelop є інтегрованим середовищем розробки з відкритим вихідним кодом і багатими можливостями, які ви можете використовувати для створення компонувальних блоків .NET на основі C #, VB .NET, Managed Extensions для C ++ або CIL. Крім того, що це середовище розробки абсолютно безкоштовне, слід відзначити те, що воно цілком створена на мові C#. Причому ви можете або завантажити та скомпілювати необхідні файли *.cs самостійно, або використовувати готову програму setup.exe, яка встановить SharpDevelop на вашій машині.

Після установки SharpDevelop вибір меню дозволить вказати вид (і мову проекту, який ви хочете створити. У термінах SharpDevelop combine (комбінат) позначає окрему колекцію проектів – ту, що в Visual Studio називається solution, тобто рішення.

Середовище розробки SharpDevelop пропонує різноманітні можливості підвищення продуктивності праці програміста. Ось список основних переваг SharpDevelop:

- підтримка компіляторів C # від Microsoft і Mono;
- можливості IntelliSense і розширення програмного коду;
- наявність діалогового вікна Add Reference (Додавання посилання) для посилань на зовнішні компонувальні блоки, включаючи компонувальні блоки, встановлені в GAG (Global Assembly Cache - глобальний кеш компонувальних блоків);
- наявність інструментів візуального проектування Windows Forms;
- різні вікна (в SharpDevelop вони називаються scouts - розвідники) для огляду структури проекту та його складових:
- інтегрована утиліта браузера об'єктів - Assembly Scout (розвідник компонувальних блоків);
- утиліти для роботи з базами даних;
- утиліта конвертації програмного коду C # в VB .NET (і навпаки):
- інтеграція з NUnit (утиліта тестування .NET-модулів) і NAnt (утиліта компонування .NET):
- інтеграція з документацією .NET Framework SDK[7].

Утиліта Assembly Scout пропонує огляд компонувальних блоків, на які є посилання в проекті. Це засіб пропонує інформацію в двох панелях, Ліва панель пропонує дерево перегляду, що дозволяє "увійти" всередину компоновочного блоку, щоб побачити простору імен і відповідні типи.

Права панель утиліти дозволяє побачити вміст елемента, обраного в лівій панелі. При цьому можна побачити не тільки основні характеристики елемента,

використовуючи для цього вкладку Info (Інформація), а й відповідний програмний код CIL. Можна також зберегти визначення елемента у файлі XML.

Таблиця 3.2 – Порівняння середовищ розробки

Критерії порівняння	Середовища розробки		
	SharpDeveloper	Eclipse	IntelliJ IDEA
Багатоплатформність	±	++	+
Швидкість модернізації	±	±	++
Наявність генератора сценаріїв складання	+	+	+
Підтримка репозиторіїв	++	+	++
Підтримка серверів	++	+	++
Розроблення графічного інтерфейсу користувача	++	–	+
Доступність (вимоги до розповсюдження)	++	+	–

Як видно із таблиці порівняння більша кількість балів у SharpDeveloper та популярного середовища IntelliJ IDEA, проте IntelliJ IDEA розповсюджується за окрему платню, у той час, як SharpDeveloper офіційно безкоштовний. З іншого боку SharpDeveloper працює із більш старими версіями C#, у той час, як нові версії Eclipse працюють із найактуальнішими версіями C# [7]. Проте слід відзначити достатньо велике число помилок та збоїв під час роботи із останніми версіями Eclipse, а більш старі та стабільні версії із кожним разом завантажити все важче. Більш того розробка графічного інтерфейсу користувача у середовищі Eclipse достатньо важка та проблематична. Тож можна зробити висновок, що C# – найбільш оптимальний варіант для подальшої розробки.

3.5 Вимоги до технічних засобів

Для нормальної розробки програми, повинні виконуватися наступні умови до апаратної частини:

– наявність IBM-сумісної персональної електронної обчислювальної машини (ПЕОМ), на якій буде працювати додаток;

– процесор із тактовою частотою 1,8 GHz та більше;

– оперативна пам'ять 1,5 ГБ та більше;

– вільний простір на жорсткому диску не менше, ніж 5 ГБ.

3.6 Робота програми

На рисунках 3.2 – 3.11 зображені результати роботи програми з розрахунками за різними критеріями.

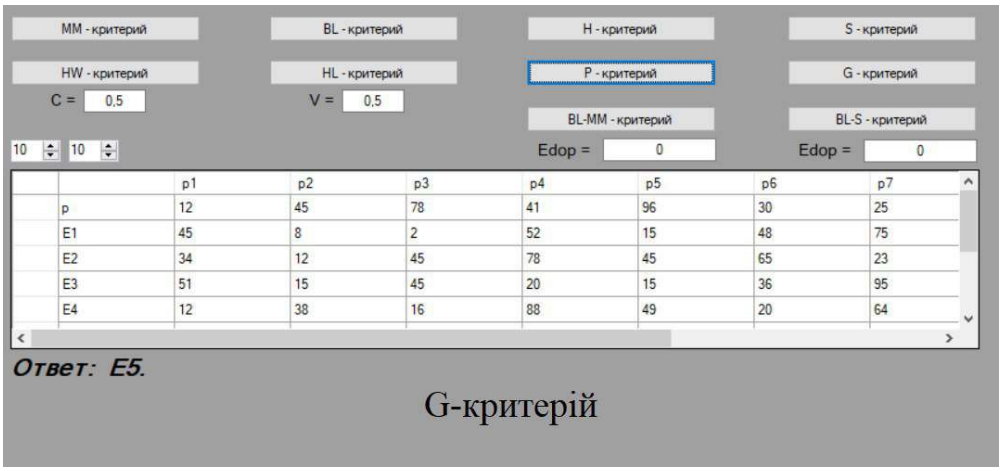

Рисунок 3.2 – Результат роботи прогрми згідно G-критерію

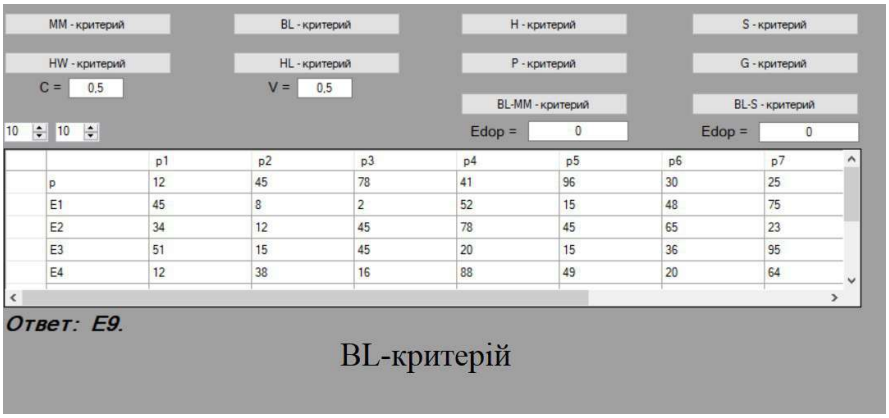

Рисунок 3.3 – Результат роботи прогрми згідно BL-критерію

Рисунок 3.4 – Результат роботи прогрми згідно P-критерію

Рисунок 3.5 – Результат роботи прогрми згідно HL-критерію

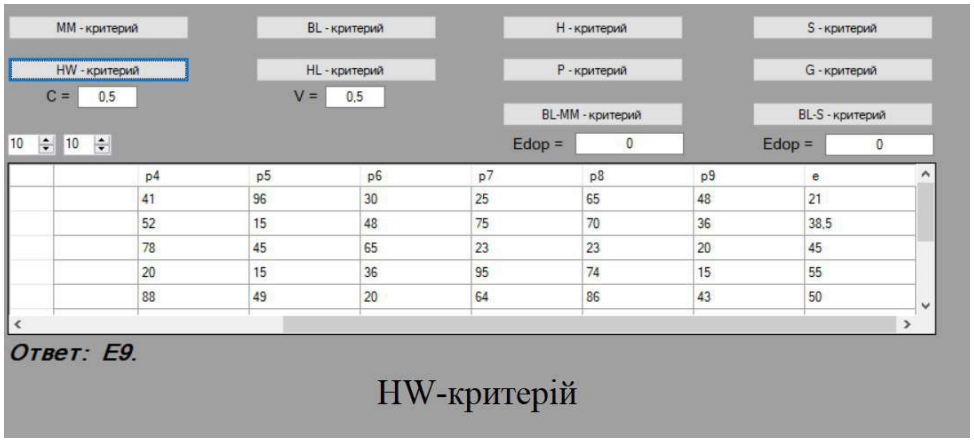

Рисунок 3.6 – Результат роботи прогрми згідно HW-критерію

Рисунок 3.7 – Результат роботи прогрми згідно H-критерію

Рисунок 3.8 – Результат роботи прогрми згідно MM-критерію

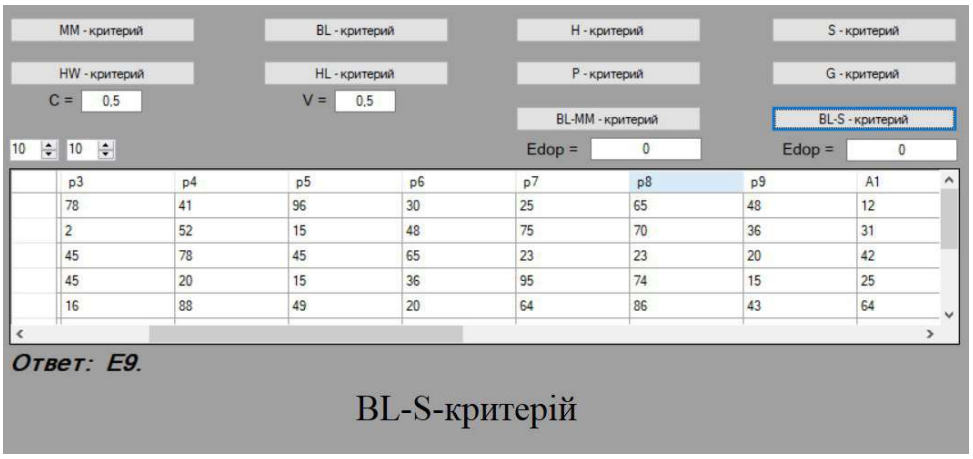

Рисунок 3.9 – Результат роботи прогрми згідно BL-S-критерію

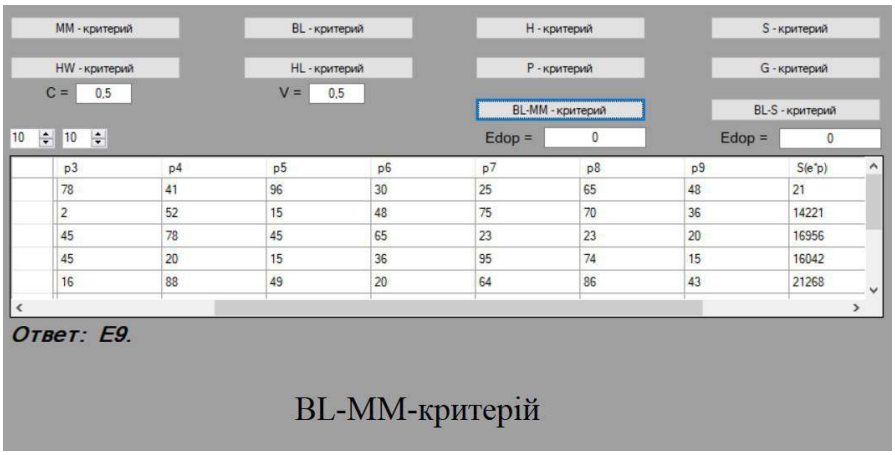

Рисунок 3.10 – Результат роботи прогрми згідно BL-MM-критерію

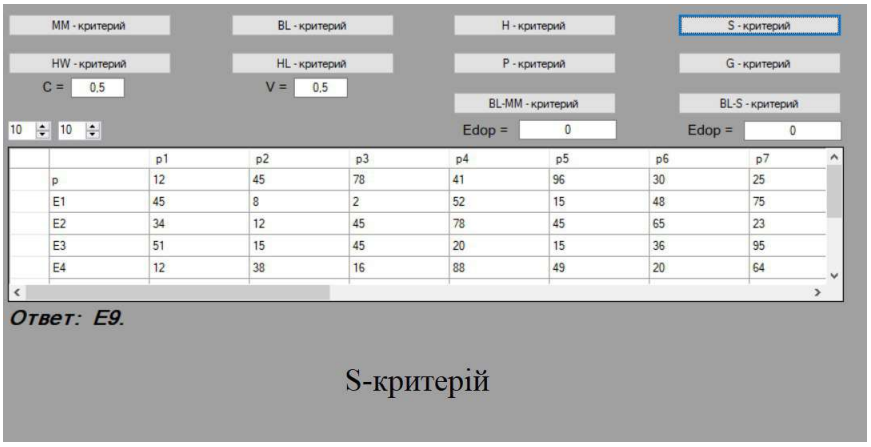

Рисунок 3.11 – Результат роботи прогрми згідно S-критерію

3.7 Висновки за розділом 3

Під час роботи над розділом було спроєктовано архітектуру майбутнього ПП. Були спроєктовані модулі майбутньої програми. Було проаналізовано найбільш популярні мови програмування та після ретельного порівняння перевага була віддана мові C#. Також проаналізувавши найбільш популярні середовища розробки, було обрано середовище SharpDeveloper.

ВИСНОВКИ

Кваліфікаційна робота присвячена алгоритмізації процесу моделювання задач прийняття рішень з великим числом альтернатив на основі дескриптивних моделей.

Було розроблено програмний комплекс, що реалізує роботу запропонованого методу.

Для цього були поставлені і вирішені такі задачі:

– ознайомлення та аналіз предметної області;

– розгляд мови CSharp та середовища розробки SharpDeveloper;

Розроблений програмний комплекс, реалізує основні функції, а саме:

– авторизація користувача;

– введення користувацьких вибірок даних;

– виведення проміжних та фінальних результатів роботи програми;

– виведення фінальних результатів роботи програми;

– збереження даних робочої сесії користувача.

Запропонована програма допомагає оптимізувати процес розрахунку критеріїв прийняття рішень. Також було наведено докладний опис кодування вхідної інформації. Було проаналізовано умови не коректного зупинення роботи методу та наведено шляхи їх уникнення.

Розроблений програмний комплекс має практичну спрямованість та готовий до використання.

Під час розробки програми та аналізу розробленого програмного комплексу з’явилися ідеї з удосконалення, такі як: використання механізму паралелізації для зменшення часу роботи та більш ефективного використання ресурсів.

Нові можливості зможуть розширити функціонал програмного комплексу та підвищити його продуктивність.

ПЕРЕЛІК ПОСИЛАНЬ

1. Балдин К.В., Брызгалов Н.А., Рукосуев А.В. Математическое программирование: ученик [Электронный ресурс]. Москва: Дашков и К. 2012. 219 с. URL: http://bibhoclub.rn/index.php?page=book&id=T12201

2. Кулик С.Д. Элементы теории принятия решений (критерии и задачи): учебное пособие [Электронный ресурс]. Москва: МИФИ, 2010. 188 с. URL: http://biblioclnb.ru/index.php?page=book&id=231904.

3. Петровский,А.Б. Теория принятия решений: учеб.: рек. УМО / Москва: Академия, 2009. 400с.: рис., табл. (Университетский учебник) .(Прикладная математика и информатика).

4. Орлов А.И. Теория принятия решений / Москва: Экзамен, 2006. 575 с.

5. Мендель А.В. Модели принятия решений: учебное пособие [Электронный ресурс] / Москва: ЮНИТИ-ДАНА, 2012. 465 с. URL: http://biblioclub.ru/index.php?page=book&id= l15173

6. Лисьев Г.А., Попова И.В. Технологии поддержки принятия решений. Учебное пособие [Электронный ресурс] / Москва: Флинта, 2011. 133 с. URL: http://biblioclub.ru/index.php?page=book&id=103806&sr= l

7. Козлов В.П. Математика и информатика: учеб .пособие: Доп.Мин.обр.РФ / Санкт-Петербург: Питер, 2004. 266 с.

8. Сорина Г.В. Основы принятия решений: учеб. пособие: рек. УМ Москва: Экономистъ, 2006. 188 с. Библиогр.: 175 с.

9. Федунец Н.И., Куприянов В.В. Теория принятия решений [Электронный ресурс]: учеб.пособие для вузов / Москва: Моск. гос. горный ун-т, 2005. 218 с. URL: http://biblioclub.ru/index.php?page=book&id=8365 4 1.

10. Шапиро Д.И. Математические методы в проблеме принятия решения / Москва: НСК, 1974. 49 с.

11. Мушик Э. Мюллер П. Методы принятия технических решений / Москва: Мир, 1990. 208 с.

12. Беляев Л.С. Решение сложных оптимизационных задач в условиях неопределенности / Новосибирск: Наука, 1978. 126 с.

13. Петросян Л.А., Зенкевич Н.А., Семина Е.А. Теория игр / Москва: Высш. шк., Книжный дом. «Университет», 1998. 304 с.

14. Розен В.В. Цель – оптимальность – решение (математические модели принятия оптимальных решений) / Москва: Радио и связь, 1982. 168 с.

15. Воробьёв Н.Н. Теория игр для экономистов-кибернетиков / Москва: Наука. Главная редакция физико-математической литературы, 1983. 272 с.

16. Вентцель Е.С. Исследование операций: задачи, принципы, методология / Москва: Наука, Главная редакция физико-математической литературы, 1980. 208 с.

17. Теория прогнозирования и принятия решений: Учеб. пособие: Под ред. С.А. Саркисяна. Москва: Высш. школа, 1977. 352 с.

18. Гуджоян О.Л. Методы принятия управленческих решений: Учебное пособие / Москва, 1997. 220 с.

19. Балабанов И.Т. Риск-менеджмент / Москва: Финансы и статистика, 1996. 192 с.

20. Дюбин Г.Н., Суздаль В.Г. Введение в прикладную теорию игр / Москва: Наука, 1981. 336 с.

21. Фишберн П. Теория полезности для принятия решений / Москва: Наука, 1978. 352 с.

22. Науман Э. Принять решение, но как? – Москва: Мир, 1987. 198 с.

23. Чернов В.А. Анализ коммерческого риска / Москва: Финансы и статистика, 1996. 128 с.

24. Вильяме Дж.Д. Совершенный стратег, или Букварь по теории стратегических игр / Москва: Советское Радио, 1960. 269 с.

ДОДАТОК А

ТЕКСТ ПРОГРАМИ

```
using System;
using System.ComponentModel;
using System.Drawing;
using System.Windows.Forms;

namespace Chouse
{
 public class Main : Form
 {
  private IContainer components = (IContainer) null;
  private NumericUpDown WidthGrid;
  private NumericUpDown HeigthGrid;
  private DataGridView dataGrid;
  private Label answer;
  private Button MmCoise;
  private Button BlChoise;
  private Button Hchoise;
  private Button Schoise;
  private Button Hwchoise;
  private Label label1;
  private TextBox C1;
  private TextBox C2;
  private Label label2;
  private Button Hlchoise;
  private Button Pchoise;
  private Button button1;
  private Button Blmmchoise;
  private Button Blschoise;
  private TextBox C3;
  private Label label3;
  private TextBox C4;
  private Label label4;

  public Main()
  {
   this.InitializeComponent();
   this.dataGrid.ColumnCount = 1;
   this.dataGrid.RowCount = 1;
  }
```

```
        private void num_ValueChanged(object sender, EventArgs e)
        {
          if (this.WidthGrid.Value > new Decimal(0))
          {
            this.dataGrid.ColumnCount = (int) this.WidthGrid.Value;
            for (int index = 1; (Decimal) index < this.WidthGrid.Value; ++index)
              this.dataGrid.Columns[index].HeaderText = "p" + (object) index;
          }
          if (!(this.HeigthGrid.Value > new Decimal(0)))
            return;
          this.dataGrid.RowCount = (int) this.HeigthGrid.Value;
          this.dataGrid[0, 0].Value = (object) "p";
          this.dataGrid[0, 0].ReadOnly = true;
          for (int index = 1; (Decimal) index < this.HeigthGrid.Value; ++index)
          {
            this.dataGrid[0, index].Value = (object) ("E" + (object) index);
            this.dataGrid[0, index].ReadOnly = true;
          }
        }

        private double[,] GetData()
        {
          double[,] numArray = new double[(int) this.WidthGrid.Value - 1, (int)
this.HeigthGrid.Value - 1];
          try
          {
            for (int index1 = 1; (Decimal) index1 < this.WidthGrid.Value; ++index1)
            {
              for (int index2 = 1; (Decimal) index2 < this.HeigthGrid.Value; ++index2)
                numArray[index1 - 1, index2 - 1] = Convert.ToDouble(this.dataGrid[index1,
index2].Value);
            }
          }
          catch (Exception ex)
          {
            this.answer.Text = "ERROR!!!Check your ask again!!!";
            return new double[0, 0];
          }
          return numArray;
        }

        private double[] GetEvent()
        {
          double[] numArray = new double[(int) this.WidthGrid.Value - 1];
          try
          {
            for (int index = 1; (Decimal) index < this.WidthGrid.Value; ++index)
              numArray[index - 1] = Convert.ToDouble(this.dataGrid[index, 0].Value);
          }
```

```
            catch (Exception ex)
            {
              this.answer.Text = "ERROR!!!Check your ask again!!!";
              return new double[0];
            }
            return numArray;
          }

          private double GetText(TextBox text)
          {
            double num = 0.0;
            try
            {
              num = Convert.ToDouble(text.Text);
            }
            catch (Exception ex)
            {
              this.answer.Text = "ERROR!!!Check your ask again!!!";
            }
            return num;
          }

          private void FindAnswer()
          {
            if (EventsChoiser.resIndexes.Length > 0)
            {
              this.answer.Text = "Ответ:";
              for (int index = 0; index < EventsChoiser.resIndexes.Length; ++index)
              {
                Label answer = this.answer;
                answer.Text = answer.Text + "  E" + (object) EventsChoiser.resIndexes[index];
              }
              this.answer.Text += ".";
            }
            else
              this.answer.Text = "Нет ответа.";
          }

          private bool Check(double[,] d, bool positiv)
          {
            int num = positiv ? 1 : -1;
            double[,] numArray = d;
            int upperBound1 = numArray.GetUpperBound(0);
            int upperBound2 = numArray.GetUpperBound(1);
            for (int lowerBound1 = numArray.GetLowerBound(0); lowerBound1 <= upperBound1;
++lowerBound1)
            {
              for (int lowerBound2 = numArray.GetLowerBound(1); lowerBound2 <=
upperBound2; ++lowerBound2)
```

```
            {
              if (numArray[lowerBound1, lowerBound2] * (double) num < 0.0)
                return false;
            }
          }
          return true;
        }

        private void MmCoise_Click(object sender, EventArgs e)
        {
          double[,] data = this.GetData();
          if (data.Length == 0)
            return;
          double[] numArray = EventsChoiser.MMchoise(data, (int) this.WidthGrid.Value - 1,
(int) this.HeigthGrid.Value - 1);
          this.dataGrid.ColumnCount = (int) this.WidthGrid.Value + 1;
          this.dataGrid.Columns[(int) this.WidthGrid.Value].HeaderText = nameof (e);
          for (int index = 0; (Decimal) index < Decimal.op_Decrement(this.HeigthGrid.Value);
++index)
            this.dataGrid[(int) this.WidthGrid.Value, index + 1].Value = (object)
Convert.ToString(numArray[index]);
          this.FindAnswer();
        }

        private void BlChoise_Click(object sender, EventArgs e)
        {
          double[,] data = this.GetData();
          double[] ev = this.GetEvent();
          if (data.Length == 0 || ev.Length == 0)
            return;
          double[] numArray = EventsChoiser.BLchoise(data, ev, (int) this.WidthGrid.Value - 1,
(int) this.HeigthGrid.Value - 1);
          this.dataGrid.ColumnCount = (int) this.WidthGrid.Value + 1;
          this.dataGrid.Columns[(int) this.WidthGrid.Value].HeaderText = nameof (e);
          for (int index = 0; (Decimal) index < Decimal.op_Decrement(this.HeigthGrid.Value);
++index)
            this.dataGrid[(int) this.WidthGrid.Value, index + 1].Value = (object)
Convert.ToString(numArray[index]);
          this.FindAnswer();
        }

        private void Hchoise_Click(object sender, EventArgs e)
        {
          double[,] data = this.GetData();
          if (data.Length == 0)
            return;
          double[] numArray = EventsChoiser.Hchoise(data, (int) this.WidthGrid.Value - 1, (int)
this.HeigthGrid.Value - 1);
          this.dataGrid.ColumnCount = (int) this.WidthGrid.Value + 1;
```

```
            this.dataGrid.Columns[(int) this.WidthGrid.Value].HeaderText = nameof (e);
            for (int index = 0; (Decimal) index < Decimal.op_Decrement(this.HeigthGrid.Value);
++index)
              this.dataGrid[(int) this.WidthGrid.Value, index + 1].Value = (object)
Convert.ToString(numArray[index]);
            this.FindAnswer();
          }

          private void Schoise_Click(object sender, EventArgs e)
          {
            double[,] data = this.GetData();
            if (data.Length == 0)
              return;
            double[,] numArray = EventsChoiser.Schoise(data, (int) this.WidthGrid.Value - 1, (int)
this.HeigthGrid.Value - 1);
            this.dataGrid.ColumnCount = (int) this.WidthGrid.Value * 2;
            for (int index1 = 0; (Decimal) index1 < Decimal.op_Decrement(this.WidthGrid.Value);
++index1)
            {
              this.dataGrid.Columns[(int) this.WidthGrid.Value + index1].HeaderText = nameof (e)
+ (object) (index1 + 1);
              for (int index2 = 0; (Decimal) index2 <
Decimal.op_Decrement(this.HeigthGrid.Value); ++index2)
                this.dataGrid[(int) this.WidthGrid.Value + index1, index2 + 1].Value = (object)
Convert.ToString(numArray[index1, index2]);
            }
            this.dataGrid.Columns[(int) this.WidthGrid.Value * 2 - 1].HeaderText = nameof (e);
            for (int index = 0; (Decimal) index < Decimal.op_Decrement(this.HeigthGrid.Value);
++index)
              this.dataGrid[(int) this.WidthGrid.Value * 2 - 1, index + 1].Value = (object)
Convert.ToString(numArray[(int) this.WidthGrid.Value - 1, index]);
            this.FindAnswer();
          }

          private void Hwchoise_Click(object sender, EventArgs e)
          {
            double text = this.GetText(this.C1);
            double[,] data = this.GetData();
            if (data.Length == 0)
              return;
            double[] numArray = EventsChoiser.HWchoise(data, (int) this.WidthGrid.Value - 1,
(int) this.HeigthGrid.Value - 1, text);
            this.dataGrid.ColumnCount = (int) this.WidthGrid.Value + 1;
            this.dataGrid.Columns[(int) this.WidthGrid.Value].HeaderText = nameof (e);
            for (int index = 0; (Decimal) index < Decimal.op_Decrement(this.HeigthGrid.Value);
++index)
              this.dataGrid[(int) this.WidthGrid.Value, index + 1].Value = (object)
Convert.ToString(numArray[index]);
            this.FindAnswer();
```

```
        }

        private void Hlchoise_Click(object sender, EventArgs e)
        {
          double text = this.GetText(this.C2);
          double[] ev = this.GetEvent();
          double[,] data = this.GetData();
          if (data.Length == 0 || ev.Length == 0)
            return;
          double[] numArray = EventsChoiser.HLchoise(data, ev, (int) this.WidthGrid.Value - 1,
(int) this.HeigthGrid.Value - 1, text);
          this.dataGrid.ColumnCount = (int) this.WidthGrid.Value + 1;
          this.dataGrid.Columns[(int) this.WidthGrid.Value].HeaderText = nameof (e);
          for (int index = 0; (Decimal) index < Decimal.op_Decrement(this.HeigthGrid.Value);
++index)
            this.dataGrid[(int) this.WidthGrid.Value, index + 1].Value = (object)
Convert.ToString(numArray[index]);
          this.FindAnswer();
        }

        private void Pchoise_Click(object sender, EventArgs e)
        {
          double[,] data = this.GetData();
          if (data.Length == 0)
            return;
          if (!this.Check(data, true))
          {
            this.answer.Text = "ERROR!!!Check your ask again!!!";
          }
          else
          {
            double[] numArray = EventsChoiser.Pchoise(data, (int) this.WidthGrid.Value - 1, (int)
this.HeigthGrid.Value - 1);
            this.dataGrid.ColumnCount = (int) this.WidthGrid.Value + 1;
            this.dataGrid.Columns[(int) this.WidthGrid.Value].HeaderText = nameof (e);
            for (int index = 0; (Decimal) index < Decimal.op_Decrement(this.HeigthGrid.Value);
++index)
              this.dataGrid[(int) this.WidthGrid.Value, index + 1].Value = (object)
Convert.ToString(numArray[index]);
            this.FindAnswer();
          }
        }

        private void Gchoise_Click(object sender, EventArgs e)
        {
          double[,] data = this.GetData();
          double[] ev = this.GetEvent();
          if (data.Length == 0 || ev.Length == 0)
            return;
```

```
            if (!this.Check(data, false))
            {
              this.answer.Text = "ERROR!!!Check your ask again!!!";
            }
            else
            {
              double[] numArray = EventsChoiser.Gchoise(data, ev, (int) this.WidthGrid.Value - 1,
(int) this.HeigthGrid.Value - 1);
              this.dataGrid.ColumnCount = (int) this.WidthGrid.Value + 1;
              this.dataGrid.Columns[(int) this.WidthGrid.Value].HeaderText = nameof (e);
              for (int index = 0; (Decimal) index < Decimal.op_Decrement(this.HeigthGrid.Value);
++index)
                this.dataGrid[(int) this.WidthGrid.Value, index + 1].Value = (object)
Convert.ToString(numArray[index]);
              this.FindAnswer();
            }
          }

          private void Blmmchoise_Click(object sender, EventArgs e)
          {
            double text = this.GetText(this.C3);
            double[,] data = this.GetData();
            double[] ev = this.GetEvent();
            if (data.Length == 0 || ev.Length == 0)
              return;
            double[,] numArray = EventsChoiser.BLMMchoise(data, ev, (int)
this.WidthGrid.Value - 1, (int) this.HeigthGrid.Value - 1, text);
            this.dataGrid.ColumnCount = (int) this.WidthGrid.Value + 3;
            this.dataGrid.Columns[(int) this.WidthGrid.Value].HeaderText = "S(e*p)";
            this.dataGrid.Columns[(int) this.WidthGrid.Value + 1].HeaderText = "Eij-minE)";
            this.dataGrid.Columns[(int) this.WidthGrid.Value + 2].HeaderText = "maxE-maxEij";
            for (int index1 = 0; index1 < 3; ++index1)
            {
              for (int index2 = 0; (Decimal) index2 <
Decimal.op_Decrement(this.HeigthGrid.Value); ++index2)
                this.dataGrid[(int) this.WidthGrid.Value + index1, index2 + 1].Value = (object)
Convert.ToString(numArray[index1, index2]);
            }
            this.FindAnswer();
          }

          private void Blschoise_Click(object sender, EventArgs e)
          {
            double text = this.GetText(this.C3);
            double[,] data = this.GetData();
            double[] ev = this.GetEvent();
            if (data.Length == 0 || ev.Length == 0)
              return;
```

```
            double[,] numArray = EventsChoiser.BLSchoise(data, ev, (int) this.WidthGrid.Value -
1, (int) this.HeigthGrid.Value - 1, text);
            this.dataGrid.ColumnCount = (int) this.WidthGrid.Value * 2 + 2;
            for (int index1 = 0; index1 < (int) this.WidthGrid.Value + 2; ++index1)
            {
             if (index1 < (int) this.WidthGrid.Value - 1)
             {
              this.dataGrid.Columns[(int) this.WidthGrid.Value + index1].HeaderText = "A" +
(object) (index1 + 1);
              this.dataGrid[(int) this.WidthGrid.Value + index1, 0].Value = this.dataGrid[index1 +
1, 0].Value;
             }
             for (int index2 = 0; (Decimal) index2 <
Decimal.op_Decrement(this.HeigthGrid.Value); ++index2)
              this.dataGrid[(int) this.WidthGrid.Value + index1, index2 + 1].Value = (object)
Convert.ToString(numArray[index1, index2]);
            }
            this.dataGrid.Columns[(int) this.WidthGrid.Value * 2 - 1].HeaderText = "S(a*p)";
            this.dataGrid.Columns[(int) this.WidthGrid.Value * 2].HeaderText = "maxA-Aij";
            this.dataGrid.Columns[(int) this.WidthGrid.Value * 2 + 1].HeaderText = "minAij-
minA";
            this.FindAnswer();
           }

           protected override void Dispose(bool disposing)
           {
            if (disposing && this.components != null)
              this.components.Dispose();
            base.Dispose(disposing);
           }

           private void InitializeComponent()
           {
            this.WidthGrid = new NumericUpDown();
            this.HeigthGrid = new NumericUpDown();
            this.dataGrid = new DataGridView();
            this.answer = new Label();
            this.MmCoise = new Button();
            this.BlChoise = new Button();
            this.Hchoise = new Button();
            this.Schoise = new Button();
            this.Hwchoise = new Button();
            this.label1 = new Label();
            this.C1 = new TextBox();
            this.C2 = new TextBox();
            this.label2 = new Label();
            this.Hlchoise = new Button();
            this.Pchoise = new Button();
            this.button1 = new Button();
```

```
this.Blmmchoise = new Button();
this.Blschoise = new Button();
this.C3 = new TextBox();
this.label3 = new Label();
this.C4 = new TextBox();
this.label4 = new Label();
this.WidthGrid.BeginInit();
this.HeigthGrid.BeginInit();
((ISupportInitialize) this.dataGrid).BeginInit();
this.SuspendLayout();
this.WidthGrid.Location = new Point(24, 119);
this.WidthGrid.Minimum = new Decimal(new int[4]
{
 1,
 0,
 0,
 0
});
this.WidthGrid.Name = "WidthGrid";
this.WidthGrid.Size = new Size(44, 20);
this.WidthGrid.TabIndex = 0;
this.WidthGrid.Value = new Decimal(new int[4]
{
 1,
 0,
 0,
 0
});
this.WidthGrid.ValueChanged += new EventHandler(this.num_ValueChanged);
this.HeigthGrid.Location = new Point(74, 119);
this.HeigthGrid.Minimum = new Decimal(new int[4]
{
 1,
 0,
 0,
 0
});
this.HeigthGrid.Name = "HeigthGrid";
this.HeigthGrid.Size = new Size(44, 20);
this.HeigthGrid.TabIndex = 1;
this.HeigthGrid.Value = new Decimal(new int[4]
{
 1,
 0,
 0,
 0
});
this.HeigthGrid.ValueChanged += new EventHandler(this.num_ValueChanged);
this.dataGrid.AllowUserToAddRows = false;
```

```
            this.dataGrid.AllowUserToDeleteRows = false;
            this.dataGrid.ColumnHeadersHeightSizeMode                                    =
DataGridViewColumnHeadersHeightSizeMode.AutoSize;
            this.dataGrid.Location = new Point(24, 145);
            this.dataGrid.Name = "dataGrid";
            this.dataGrid.Size = new Size(836, 155);
            this.dataGrid.TabIndex = 2;
            this.answer.AutoSize = true;
            this.answer.Font = new Font("Microsoft Sans Serif", 15f, FontStyle.Bold |
FontStyle.Italic, GraphicsUnit.Point, (byte) 204);
            this.answer.Location = new Point(19, 303);
            this.answer.Name = "answer";
            this.answer.Size = new Size(0, 25);
            this.answer.TabIndex = 3;
            this.MmCoise.AccessibleDescription = "";
            this.MmCoise.Location = new Point(24, 12);
            this.MmCoise.Name = "MmCoise";
            this.MmCoise.Size = new Size(164, 23);
            this.MmCoise.TabIndex = 4;
            this.MmCoise.Tag = (object) "";
            this.MmCoise.Text = "MM - критерий";
            this.MmCoise.UseVisualStyleBackColor = true;
            this.MmCoise.Click += new EventHandler(this.MmCoise_Click);
            this.BlChoise.AccessibleDescription = "";
            this.BlChoise.Location = new Point(247, 12);
            this.BlChoise.Name = "BlChoise";
            this.BlChoise.Size = new Size(164, 23);
            this.BlChoise.TabIndex = 5;
            this.BlChoise.Tag = (object) "";
            this.BlChoise.Text = "BL - критерий";
            this.BlChoise.UseVisualStyleBackColor = true;
            this.BlChoise.Click += new EventHandler(this.BlChoise_Click);
            this.Hchoise.AccessibleDescription = "";
            this.Hchoise.Location = new Point(469, 12);
            this.Hchoise.Name = "Hchoise";
            this.Hchoise.Size = new Size(164, 23);
            this.Hchoise.TabIndex = 6;
            this.Hchoise.Tag = (object) "";
            this.Hchoise.Text = "H - критерий";
            this.Hchoise.UseVisualStyleBackColor = true;
            this.Hchoise.Click += new EventHandler(this.Hchoise_Click);
            this.Schoise.AccessibleDescription = "";
            this.Schoise.Location = new Point(694, 12);
            this.Schoise.Name = "Schoise";
            this.Schoise.Size = new Size(164, 23);
            this.Schoise.TabIndex = 7;
            this.Schoise.Tag = (object) "";
            this.Schoise.Text = "S - критерий";
            this.Schoise.UseVisualStyleBackColor = true;
```

```
            this.Schoise.Click += new EventHandler(this.Schoise_Click);
            this.Hwchoise.AccessibleDescription = "";
            this.Hwchoise.Location = new Point(24, 50);
            this.Hwchoise.Name = "Hwchoise";
            this.Hwchoise.Size = new Size(164, 23);
            this.Hwchoise.TabIndex = 8;
            this.Hwchoise.Tag = (object) "";
            this.Hwchoise.Text = "HW - критерий";
            this.Hwchoise.UseVisualStyleBackColor = true;
            this.Hwchoise.Click += new EventHandler(this.Hwchoise_Click);
            this.label1.AutoSize = true;
            this.label1.Font = new Font("Microsoft Sans Serif", 10f, FontStyle.Regular,
GraphicsUnit.Point, (byte) 204);
            this.label1.Location = new Point(56, 76);
            this.label1.Name = "label1";
            this.label1.Size = new Size(29, 17);
            this.label1.TabIndex = 9;
            this.label1.Text = "C =";
            this.label1.TextAlign = ContentAlignment.MiddleCenter;
            this.C1.Location = new Point(87, 76);
            this.C1.Name = "C1";
            this.C1.Size = new Size(54, 20);
            this.C1.TabIndex = 10;
            this.C1.Text = "0,5";
            this.C1.TextAlign = HorizontalAlignment.Center;
            this.C2.Location = new Point(310, 76);
            this.C2.Name = "C2";
            this.C2.Size = new Size(54, 20);
            this.C2.TabIndex = 13;
            this.C2.Text = "0,5";
            this.C2.TextAlign = HorizontalAlignment.Center;
            this.label2.AutoSize = true;
            this.label2.Font = new Font("Microsoft Sans Serif", 10f, FontStyle.Regular,
GraphicsUnit.Point, (byte) 204);
            this.label2.Location = new Point(279, 76);
            this.label2.Name = "label2";
            this.label2.Size = new Size(29, 17);
            this.label2.TabIndex = 12;
            this.label2.Text = "V =";
            this.label2.TextAlign = ContentAlignment.MiddleCenter;
            this.Hlchoise.AccessibleDescription = "";
            this.Hlchoise.Location = new Point(247, 50);
            this.Hlchoise.Name = "Hlchoise";
            this.Hlchoise.Size = new Size(164, 23);
            this.Hlchoise.TabIndex = 11;
            this.Hlchoise.Tag = (object) "";
            this.Hlchoise.Text = "HL - критерий";
            this.Hlchoise.UseVisualStyleBackColor = true;
            this.Hlchoise.Click += new EventHandler(this.Hlchoise_Click);
```

```
            this.Pchoise.AccessibleDescription = "";
            this.Pchoise.Location = new Point(469, 50);
            this.Pchoise.Name = "Pchoise";
            this.Pchoise.Size = new Size(164, 23);
            this.Pchoise.TabIndex = 14;
            this.Pchoise.Tag = (object) "";
            this.Pchoise.Text = "P - критерий";
            this.Pchoise.UseVisualStyleBackColor = true;
            this.Pchoise.Click += new EventHandler(this.Pchoise_Click);
            this.button1.AccessibleDescription = "";
            this.button1.Location = new Point(694, 50);
            this.button1.Name = "button1";
            this.button1.Size = new Size(164, 23);
            this.button1.TabIndex = 15;
            this.button1.Tag = (object) "";
            this.button1.Text = "G - критерий";
            this.button1.UseVisualStyleBackColor = true;
            this.button1.Click += new EventHandler(this.Gchoise_Click);
            this.Blmmchoise.AccessibleDescription = "";
            this.Blmmchoise.Location = new Point(469, 90);
            this.Blmmchoise.Name = "Blmmchoise";
            this.Blmmchoise.Size = new Size(164, 23);
            this.Blmmchoise.TabIndex = 16;
            this.Blmmchoise.Tag = (object) "";
            this.Blmmchoise.Text = "BL-MM - критерий";
            this.Blmmchoise.UseVisualStyleBackColor = true;
            this.Blmmchoise.Click += new EventHandler(this.Blmmchoise_Click);
            this.Blschoise.AccessibleDescription = "";
            this.Blschoise.Location = new Point(694, 90);
            this.Blschoise.Name = "Blschoise";
            this.Blschoise.Size = new Size(164, 23);
            this.Blschoise.TabIndex = 17;
            this.Blschoise.Tag = (object) "";
            this.Blschoise.Text = "BL-S - критерий";
            this.Blschoise.UseVisualStyleBackColor = true;
            this.Blschoise.Click += new EventHandler(this.Blschoise_Click);
            this.C3.Location = new Point(535, 118);
            this.C3.Name = "C3";
            this.C3.Size = new Size(98, 20);
            this.C3.TabIndex = 19;
            this.C3.Text = "0";
            this.C3.TextAlign = HorizontalAlignment.Center;
            this.label3.AutoSize = true;
            this.label3.Font = new Font("Microsoft Sans Serif", 10f, FontStyle.Regular,
GraphicsUnit.Point, (byte) 204);
            this.label3.Location = new Point(476, 119);
            this.label3.Name = "label3";
            this.label3.Size = new Size(53, 17);
            this.label3.TabIndex = 18;
```

```
            this.label3.Text = "Edop =";
            this.label3.TextAlign = ContentAlignment.MiddleCenter;
            this.C4.Location = new Point(760, 119);
            this.C4.Name = "C4";
            this.C4.Size = new Size(98, 20);
            this.C4.TabIndex = 21;
            this.C4.Text = "0";
            this.C4.TextAlign = HorizontalAlignment.Center;
            this.label4.AutoSize = true;
            this.label4.Font = new Font("Microsoft Sans Serif", 10f, FontStyle.Regular,
GraphicsUnit.Point, (byte) 204);
            this.label4.Location = new Point(701, 119);
            this.label4.Name = "label4";
            this.label4.Size = new Size(53, 17);
            this.label4.TabIndex = 20;
            this.label4.Text = "Edop =";
            this.label4.TextAlign = ContentAlignment.MiddleCenter;
            this.AutoScaleDimensions = new SizeF(6f, 13f);
            this.AutoScaleMode = AutoScaleMode.Font;
            this.BackColor = SystemColors.ControlDark;
            this.ClientSize = new Size(876, 331);
            this.Controls.Add((Control) this.C4);
            this.Controls.Add((Control) this.label4);
            this.Controls.Add((Control) this.C3);
            this.Controls.Add((Control) this.label3);
            this.Controls.Add((Control) this.Blschoise);
            this.Controls.Add((Control) this.Blmmchoise);
            this.Controls.Add((Control) this.button1);
            this.Controls.Add((Control) this.Pchoise);
            this.Controls.Add((Control) this.C2);
            this.Controls.Add((Control) this.label2);
            this.Controls.Add((Control) this.Hlchoise);
            this.Controls.Add((Control) this.C1);
            this.Controls.Add((Control) this.label1);
            this.Controls.Add((Control) this.Hwchoise);
            this.Controls.Add((Control) this.Schoise);
            this.Controls.Add((Control) this.Hchoise);
            this.Controls.Add((Control) this.BlChoise);
            this.Controls.Add((Control) this.MmCoise);
            this.Controls.Add((Control) this.answer);
            this.Controls.Add((Control) this.dataGrid);
            this.Controls.Add((Control) this.HeigthGrid);
            this.Controls.Add((Control) this.WidthGrid);
            this.Name = nameof (Main);
            this.Text = "Принятие решения";
            this.WidthGrid.EndInit();
            this.HeigthGrid.EndInit();
            ((ISupportInitialize) this.dataGrid).EndInit();
            this.ResumeLayout(false); this.PerformLayout();}}}
```