СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ
Навчально-науковий інститут (факультет) інформаційних технологій та електроніки
Кафедра інформаційних технологій та програмування

Пояснювальна записка

до магістерської дипломної роботи
магістр
(освітньо-кваліфікаційний рівень)

на тему Інтелектуальна система підтримки рішень для управління IT-проектами

Виконав: студент 2 курсу, групи ІСТ-24дм
126 «Інформаційні системи та технології»
(шифр і назва спеціальності)
Стоянов Б. Ю.
(прізвище та ініціали)
Керівник Дьомін М.К.
(прізвище та ініціали)
Рецензент Ратов Д.В.
(прізвище та ініціали)

Київ – 2025 року

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ ДО МАГІСТЕРСЬКОЇ ДИПЛОМНОЇ РОБОТИ

СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Навчально-науковий інститут (факультет) інформаційних технологій та електроніки

Кафедра інформаційних технологій та програмування

Освітньо-кваліфікаційний рівень магістр

Спеціальність 126 «Інформаційні системи та технології»

(шифр і назва спеціальності)

ЗАТВЕРДЖУЮ
Завідувач кафедри ІТП
д.т.н., проф., Захожай О. І.
(підпис)
«____»_____2025р.

ЗАВДАННЯ
на магістерську дипломну роботу студенту
Стоянову Богдану Юрійовичу

(прізвище, ім'я, по батькові)

1. Тема роботи Інтелектуальна система підтримки рішень для управління ІТ-проектами
керівник роботи Дьомін Максим Костянтинович, к.т.н., доц.

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

затверджені наказом вищого навчального закладу
від «08» 12 2025року №241/17.03

2. Строк подання студентом роботи 15.12.2025
3. Вихідні дані до роботи: Матеріали науково-дослідної практики, науково-методична література; дані інтернет-мережі .
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
   4.1 Вступ
   4.2 Аналіз проблеми дослідження
   4.3 Застосування методів підтримки прийняття рішень на основі методу аналізу ієрархій.
   4.4 Розробка зручного десктопного програмного продукту.
   4.5 Висновки
   4.6 Перелік використаних джерел
5. Перелік графічного матеріалу (з точним значенням обов'язків креслень)

6. Консультанти розділів проекту (роботи)

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

7. Дата видачі завдання 18 жовтня 2025

КАЛЕНДАРНИЙ ПЛАН

№ з\п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Дослідження предметної галузі	25.10.25 – 28.10.25	
2	Пошук та аналіз існуючих рішень	29.10.25 – 04.11.25	
3	Застосування методів підтримки прийняття рішень на основі методу аналізу ієрархій	05.11.25 – 15.11.25	
5	Розробка зручного десктопного програмного продукту	16.11.25 – 25.11.25	
6	Тестування	25.11.25 – 02.12.25	
7	Оформлення пояснювальної записки	02.12.25 – 05.12.25	
8	Підготовка та подання магістерської роботи до захисту	06.12.25 – 06.12.25	

Студент_____ Стоянов Б.Ю.
(підпис) (прізвище та ініціали)

Керівник роботи_____ Дьомін М.К.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Магістерська дипломна робота: 62 стор., 5 рис., 7 таб., 11 джерел

Об’єкт дослідження – процеси прийняття управлінських рішень у сфері розробки та управління ІТ-проєктами.

Предмет дослідження – методи та програмні засоби підтримки прийняття рішень на основі методу аналізу ієрархій (AHP).

Мета роботи – підвищення якості та обґрунтованості управлінських рішень в ІТ-сфері шляхом створення спеціалізованої системи підтримки прийняття рішень, що реалізує повний цикл методу Сааті.

Поставлена задача: розробити зручний десктопний програмний продукт, який дозволяє за 15–20 хвилин провести багатокритеріальний аналіз будь-якої задачі (вибір технологічного стеку, постачальника, стартапу для інвестування тощо) з автоматичним розрахунком пріоритетів та контролем узгодженості суджень.

Програмний продукт розроблено мовою програмування C# з використанням технології Windows Forms. Система підтримує до 10 критеріїв та 5 альтернатив, динамічно генерує інтерфейс, забезпечує автоматичне заповнення обернених значень, розрахунок локальних і глобальних пріоритетів, перевірку узгодженості (λmax, ІС, ВС) та візуалізацію результатів.

Практичне тестування проведено на двох реальних кейсах:

- вибір технологічного стеку для корпоративного вебпорталу (переміг .NET + Angular);

- оцінка трьох ІТ-стартапів для інвестування (переміг InFortis).

Розроблена система є готовим до використання інструментом для технічних лідів, проджект-менеджерів та інвесторів, значно знижує суб’єктивізм і час прийняття складних рішень у сфері ІТ-проєктів.

Зміст

ВСТУП

Сучасна сфера розробки IT-проєктів характеризується високим рівнем невизначеності, швидкими змінами вимог, великою кількістю альтернативних рішень та необхідністю одночасного врахування десятків різнорідних критеріїв: вартості, термінів виконання, технологічної новизни, масштабованості, безпеки, доступності кадрів, ризиків тощо. У таких умовах традиційні підходи до прийняття управлінських рішень — інтуїція, досвід або просте голосування — часто призводять до субоптимальних або навіть помилкових виборів, що в підсумку збільшують бюджет, зривають терміни та знижують конкурентоспроможність продукту.

Актуальність теми зумовлена об’єктивною потребою IT-компаній, стартапів, інвесторів та технічних лідів у надійних, прозорих та науково обґрунтованих інструментах підтримки прийняття рішень. Найбільш ефективним і перевіреним часом методом вирішення багатокритеріальних задач із змішаними кількісними та якісними факторами є метод аналізу ієрархій (AHP), розроблений Томасом Сааті. Він дозволяє систематизувати складну проблему, формалізувати експертні судження та отримати чіткий рейтинг альтернатив із математично доведеною узгодженістю оцінок.

Об’єктом дослідження є процеси прийняття управлінських рішень у сфері розробки та управління IT-проєктами.

Предметом дослідження є методи та програмні засоби підтримки прийняття рішень на основі методу аналізу ієрархій.

Метою роботи є підвищення якості та обґрунтованості управлінських рішень в IT-сфері шляхом розробки спеціалізованої системи підтримки прийняття рішень, яка реалізує повний цикл методу аналізу ієрархій у зручному для кінцевого користувача вигляді.

Для досягнення поставленої мети вирішуються такі завдання:

1. Провести аналіз сучасного стану систем підтримки прийняття рішень

та обґрунтувати вибір методу аналізу ієрархій як математичної основи.

2. Розробити зручну десктопну програму на мові C# з інтерфейсом Windows Forms, яка автоматизує всі етапи AHP: побудову ієрархії, заповнення матриць парних порівнянь, обчислення пріоритетів, перевірку узгодженості та візуалізацію результатів.

3. Продемонструвати практичне застосування системи на реальних задачах IT-сфери: вибір технологічного стеку для корпоративного вебпорталу та оцінка інвестиційної привабливості стартапів.

Наукова новизна полягає у створенні спеціалізованої системи підтримки прийняття рішень, адаптованої саме до типових задач IT-проєктів, з інтуїтивно зрозумілим інтерфейсом та повною автоматизацією обчислень за методом Сааті.

Практична цінність роботи полягає у тому, що розроблена програма є готовим до використання інструментом для технічних лідів, проджект-менеджерів, архітекторів та інвесторів. Вона дозволяє за 15–20 хвилин отримати науково обґрунтовану рекомендацію замість багатогодинних дискусій, значно знижуючи суб’єктивізм та підвищуючи якість ключових рішень у розробці IT-продуктів.

Таким чином, дана дипломна робота присвячена вирішенню актуальної проблеми сучасної IT-індустрії — переходу від інтуїтивного прийняття рішень до їх формалізації та автоматизації за допомогою одного з найнадійніших математичних методів — методу аналізу ієрархій.

1 АНАЛІЗ ПРОБЛЕМИ ДОСЛІДЖЕННЯ

1.1 Прийняття рішення та їхня роль у функціонуванні організації

Розв’язання задач і прийняття рішень є фундаментальними вміннями як у повсякденному житті, так і в бізнес-середовищі. Для подолання труднощів потрібно вміло обирати варіанти дій, що стає ключовою умовою для обіймання керівних позицій у різноманітних сферах підприємництва.

Фактично, логічне й аргументоване прийняття рішень вважається основною роллю менеджменту, оскільки воно формує як структуру організації, так і її управлінські процеси. Рішення можна трактувати як послідовність кроків, свідомо відібраних із множини варіантів для реалізації організаційних або управлінських завдань. Процедура прийняття рішень є постійним і необхідним елементом керування будь-якою компанією чи комерційною діяльністю.

Менеджери на всіх рівнях щоденно ухвалюють різноманітні рішення, розв’язуючи виклики своїх установ. На сучасному етапі існує чимало підходів і технік для підвищення ефективності прийняття рішень. Надзвичайно важливо підібрати оптимальний спосіб для розв’язання конкретної управлінської задачі.

Прийняття рішень включає обрання напрямку дій серед двох чи більше потенційних опцій, аби досягти розв’язання певної проблеми. Це свідчить про те, що процедура прийняття рішень орієнтована на ціль. Цілі представляють собою заздалегідь визначені орієнтири бізнесу, місію фірми та її стратегічне бачення. На шляху до цієї цілі підприємство може стикнутися з численними бар’єрами в адміністративній, маркетинговій та операційній галузях.

На процес прийняття рішень певною мірою впливають активні колективи – об’єднання осіб, які поділяють спільні зацікавлення щодо задачі, що вимагає врегулювання.

Під час ухвалення рішень завжди враховуються інтереси залучених колективів, з огляду на їхні погляди при аналізі альтернативних варіантів. У цьому контексті значну роль відіграють фахівці, які на професійному рівні орієнтуються в усіх аспектах проблеми. Якщо спеціаліст ідеально володіє знаннями у своїй галузі, то його судження є неупередженими та мають суттєву

вагу для остаточного вибору.

Прийняття рішень являє собою процедуру відбору напрямку дій із кількох можливих опцій. Альтернатива – це комплекс заходів, що гарантує всебічний підхід до подолання задачі. Альтернативи мають демонструвати принципово відмінні методи розв'язання проблеми або різні акценти в цілях, а також пропонувати реальні можливості та варіанти для осіб, які ухвалюють рішення. Для формулювання задачі прийняття рішень потрібно мати принаймні дві альтернативи.

Альтернативи поділяються на два типи: залежні та незалежні. Залежні альтернативи – це ті, де оцінка однієї впливає на характеристики інших. Незалежні альтернативи – це опції, дії з якими не змінюють властивостей решти. Існування залежних альтернатив означає, що при одержанні даних про одну опцію особа, яка ухвалює рішення, може оцінити ефективність іншої. У такій ситуації визначення цінності інформації або дослідження трактується як задача встановлення обсягу вибірки. Форми залежності альтернатив різноманітні. Найпростішою та очевидною є пряма групова залежність: якщо вирішено аналізувати хоча б одну опцію з набору, то необхідно розглядати весь набір.

Трапляються ситуації, коли всі альтернативи вже сформовані та визначені, і потрібно лише вибрати найкращі з наявної сукупності. Наприклад, можна відшукувати найефективніше підприємство серед існуючих, визначати провідний університет чи оптимальний літак серед збудованих. Характерною рисою таких задач є закрита, не розширювана множина альтернатив.

Коли опцій надзвичайно багато, особа, яка ухвалює рішення, не завжди здатна приділити увагу кожній. У подібних обставинах зростає потреба в чітких критеріях відбору, в методиках залучення фахівців, у створенні набору норм, що дають змогу реалізовувати несуперечливу та логічну стратегію.

Сучасні техніки прийняття рішень спрямовані на врахування всіх унікальних ознак альтернатив, повноти їхнього опису, що значно наближає теоретичні моделі до реальності. Тому нині багатокритеріальний аналіз альтернатив набуває дедалі більшої популярності. Одним зі шляхів відповіді на ці вимоги є математичне формулювання проблеми прийняття рішень.

У разі, коли альтернативи не окреслені, критерії встановлюються на базі вимог до задачі відбору. При цьому вивчаються або минулі сценарії прийняття рішень, або передбачувані опції.

Критерії можуть бути незалежними чи взаємопов'язаними. Критерії вважаються залежними, коли оцінка альтернативи за одним з них визначає (детерміновано чи з високою ймовірністю) оцінку за іншим. Наявність залежних критеріїв провокує проблему «подвійного обліку», яка завищує певний аспект цілі, враховуючи двічі одні й ті самі критерії. Наприклад, можна припустити, що високоякісний автомобіль є дорогим. Залежність між критеріями призводить до формування цілісних профілів альтернатив.

На ухвалення якісних рішень витрачається чимало часу, оскільки в управлінській сфері рішення не можна приймати поспішно. Процедура повинна дотримуватися таких етапів, як:

- визначення задачі
- збір даних та відомостей від зацікавлених сторін
- розробка та оцінка опцій
- відбір оптимального варіанта
- планування
- реалізація
- здійснення подальших заходів

Оскільки процедура прийняття рішень слідує за зазначеними послідовними етапами, у ній затрачається значний час. Це стосується кожного рішення, спрямованого на розв'язання управлінських та адміністративних задач у бізнес-контексті. Хоча весь цикл потребує багато часу, наслідки такого підходу в професійній установі є вкрай позитивними.

Коли йдеться про ухвалення зваженого та вдалого рішення, надмірна залежність від інтуїтивних реакцій, що базуються на сприйнятті, або надто сильна опора на стереотипи, коли дані надходять з усіх напрямків, може становити загрозу.

Не всі індивіди готові до ризиків. Причини зрозумілі. Одне помилкове

рішення здатне зруйнувати багато елементів, зокрема постраждати можуть репутація фірми, термін життя продукту, фінансова стабільність та імідж роботодавця.

У певних обставинах неможливо застосувати базові принципи економіки, статистики та операційного аналізу для чіткого вибору. Цю складність допоможуть подолати системи, побудовані на знаннях для бізнес-рішень. Такою є система підтримки прийняття рішень, яка в подібних випадках може накопичувати та обробляти дані, генерувати прогнози на основі вивчення наявних шаблонів. Це прискорює всю процедуру, пояснюючи, як оптимізувати процес реалізації. Крім того, система сприяє ухваленню рішень щодо планування, виробництва, керування та операцій, спираючись на доступні дані. Вона дає змогу отримати повну картину того, що могло бути пропущене, та забезпечує точні обчисленняотримання інформації, яка потрібна на прийняття рішень;

1.2 Особливості розвитку систем підтримки прийняття рішень

Система підтримки прийняття рішень (СППР) — це комп’ютеризований програмний комплекс, який збирає, впорядковує та аналізує різноманітні вихідні дані, документи, знання з фундаментальних і прикладних дисциплін, а також персональний досвід особи, що приймає рішення, з метою виявлення проблеми та формування найбільш ефективного варіанту її розв’язання. СППР допомагає долати типові бар’єри, що ускладнюють ухвалення якісних рішень, зокрема:

- недостатній практичний досвід;
- суб’єктивні упередження;
- брак часу;
- помилки в обчисленнях;
- ігнорування частини альтернатив.

Історія розвитку СППР починається наприкінці 1960-х років. Спочатку це були порівняно прості модельно-орієнтовані системи, які видавали періодичні звіти для керівництва. У 1970-х роках вони еволюціонували до

складніших інструментів, що підтримували функції планування виробництва, маркетингу, ціноутворення та логістики.

Значний прорив відбувся на початку 1980-х років: функціонал СППР суттєво розширився, а до кінця десятиліття з'явилися системи, здатні працювати з великими масивами даних у реальному часі. У 1990-ті роки завдяки розвитку клієнт-серверних архітектур, об'єктно-орієнтованого програмування та потужних систем управління базами даних (СУБД) СППР перейшли на якісно новий рівень. Масове поширення Інтернету дало поштовх до появи веб-орієнтованих рішень, технологій OLAP (Online Analytical Processing), сховищ даних та інтелектуального аналізу даних (Data Mining).

Сучасні СППР є гнучкими інтерактивними платформами, які поєднують:

- великі обсяги структурованих і неструктурованих даних;
- складні аналітичні моделі;
- засоби візуалізації;
- механізми штучного інтелекту та машинного навчання.

Еволюція СППР пройшла кілька ключових етапів:

1. 1960–1970-ті рр. — модельно-орієнтовані системи (Model-driven DSS);
2. 1980-ті рр. — системи, орієнтовані на дані (Data-driven DSS);
3. 1990-ті рр. — інтеграція з корпоративними базами даних та клієнт-серверними технологіями;
4. 2000-ті рр. — веб-орієнтовані та хмарні рішення;
5. 2010-ті – 2020-ті рр. — інтеграція штучного інтелекту, великих даних (Big Data), предиктивної та прескриптивної аналітики.

Сьогодні СППР успішно застосовуються в найрізноманітніших галузях: обороні, медицині, промисловості, фінансовому секторі, транспорті, енергетиці, державному управлінні тощо. Вони особливо цінні в ситуаціях, де потрібна висока точність, але остаточне рішення все одно залишається за людиною.

Класифікація сучасних СППР може проводитися за кількома ознаками:

За технологічною основою:

- Орієнтовані на моделі (Model-driven DSS);

- Орієнтовані на дані (Data-driven DSS);

- Орієнтовані на знання (Knowledge-driven DSS);

- Орієнтовані на документи (Document-driven DSS);

- Комунікаційно-орієнтовані (Communications-driven DSS).

За характером підтримувальних операцій:

- системи пошуку та видачі файлів;

- системи аналізу інформації;

- системи фінансово-бухгалтерського моделювання;

- системи представлення рішень та сценарного аналізу («що-якщо»);

- системи оптимізації;

- системи пропозицій (рекомендаційні системи).

Особливістю сучасного етапу розвитку є перехід від простого прискорення обробки інформації до створення інтелектуальних помічників, які не лише надають дані, а й пропонують обґрунтовані рекомендації, враховуючи нечіткість, невизначеність та динамічність зовнішнього середовища.

Таким чином, СППР перетворилися з допоміжних інструментів звітності на стратегічні платформи, що суттєво підвищують якість управлінських рішень, зменшують ризики та забезпечують конкурентні переваги організаціям у умовах швидко змінного бізнес-середовища.

1.3 Класифікація систем підтримки прийняття рішень

Системи підтримки прийняття рішень охоплюють широкий спектр застосувань – від повсякденних інструментів до складних корпоративних платформ. Залежно від використовуваних технологій, рівня автономності, характеру даних та типу підтримуваних операцій СППР прийнято класифікувати за кількома основними ознаками.

Класифікація за технологічною основою

1. Орієнтовані на моделі (Model-driven DSS)

Ґрунтуються на використанні математичних, статистичних або імітаційних моделей. Найпростіші приклади – системи планування

виробництва, фінансового моделювання, логістики. Користувач задає параметри, а система виконує розрахунки за фіксованими алгоритмами.

2. Орієнтовані на дані (Data-driven DSS)

Основний акцент – доступ до великих обсягів структурованих даних у реальному часі, їх агрегація та візуалізація. Типові представники – виконавчі інформаційні системи (EIS), OLAP-куби, дашборди та системи бізнес-аналітики (BI).

3. Орієнтовані на знання (Knowledge-driven DSS)

Містять бази знань, правила та механізми логічного виведення. Часто включають елементи експертних систем і штучного інтелекту. Застосовуються в медицині (системи діагностики), технічному обслуговуванні, кредитному скорингу.

4. Орієнтовані на документи (Document-driven DSS)

Працюють із неструктурованими або слабоструктурованими даними: тексти, звіти, контракти, електронна пошта, веб-сторінки. Використовують технології пошуку, класифікації та извлечения інформації.

5. Комунікаційно-орієнтовані (Communications-driven DSS)

Забезпечують групову роботу та спільне прийняття рішень. Включають системи групової підтримки (GDSS), відеоконференції, корпоративні портали, спільні дошки, чати з інтелектуальними ботами.

Класифікація за характером підтримувальних операцій

1. Системи видачі файлів (File drawer systems)

Найпростіший тип – надають швидкий доступ до потрібного документа або звіту за запитом.

2. Системи аналізу інформації (Data analysis systems)

Виконують статистичну обробку, побудову трендів, виявлення аномалій та формування аналітичних звітів.

3. Системи бухгалтерського та фінансового моделювання

Підтримують бюджетування, прогнозування грошових потоків, оцінку інвестиційних проектів.

4. Системи представлення рішень (Representation models)

Дозволяють проводити сценарний аналіз «що-якщо», моделювати наслідки різних управлінських дій.

5. Системи оптимізації (Optimization models)

Знаходять найкраще рішення з урахуванням обмежень (лінійне, нелінійне програмування, генетичні алгоритми тощо).

6. Рекомендаційні системи (Suggestion systems)

На основі накопичених даних та моделей машинного навчання пропонують конкретні дії або ранжують альтернативи.

Класифікація за рівнем інтелектуальності та автономності

- Пасивні СППР – лише надають інформацію, рішення повністю залишається за людиною.

- Активні СППР – пропонують варіанти рішень, пояснюють їх обґрунтування.

- Кооперативні СППР – дозволяють користувачу модифікувати, доповнювати або відхиляти запропоновані варіанти, після чого система повторно обробляє дані.

Класифікація за масштабами використання

- Персональні СППР (для одного керівника або аналітика).

- Групові/командні СППР (GDSS).

- Організаційні/корпоративні СППР (Enterprise-wide DSS).

- Міжорганізаційні СППР (використовуються партнерами, постачальниками, державними органами).

Сучасні СППР часто поєднують кілька з перелічених підходів, утворюючи гібридні системи. Наприклад, корпоративна BI-платформа може одночасно містити OLAP-куби (Data-driven), вбудовані моделі прогнозування (Model-driven), рекомендаційні механізми на основі машинного навчання (Knowledge-driven) та веб-інтерфейс для спільної роботи (Communications-driven).

Така багатогранна класифікація дає змогу підбирати або розробляти СППР, що максимально відповідає специфіці галузі, масштабам організації та

характеру задач, які необхідно розв’язувати.

1.4 Класифікація систем підтримки прийняття рішень за характером операцій

Залежно від типу задач, які система допомагає розв’язувати, та від характеру операцій, що виконуються, сучасні СППР поділяють на такі основні категорії:

1. Системи видачі файлів (File Drawer Systems)

Найпростіший клас СППР. Функціонують як «електронний архів»: за запитом користувача швидко знаходять і видають потрібний документ, звіт, довідку чи історичні дані. Не проводять аналітичної обробки, лише забезпечують зручний та швидкий доступ до інформації.

2. Системи аналізу інформації (Information Analysis Systems)

Здійснюють обробку великих масивів даних, будують агреговані звіти, діаграми, виявляють тренди, аномалії та взаємозв’язки. Типові приклади – системи бізнес-аналітики (BI), OLAP-інструменти, дашборди керівника.

3. Системи бухгалтерського та фінансового моделювання (Accounting and Financial Models)

Спеціалізовані на фінансовому плануванні, бюджетуванні, розрахунку грошових потоків, оцінці інвестиційних проектів, управлінні запасами та обліку. Автоматизують рутинні розрахунки й допомагають оцінювати фінансові наслідки різних сценаріїв.

4. Системи представлення рішень та сценарного моделювання (Representation Models / «Що-якщо» аналіз)

Дозволяють користувачу змінювати вхідні параметри й одразу бачити, як це вплине на кінцевий результат. Класичний приклад – електронні таблиці з формулами або спеціалізовані імітаційні моделі. Користувач сам формулює гіпотези та перевіряє їх наслідки.

5. Системи оптимізації (Optimization Systems)

Автоматично шукають найкраще рішення з урахуванням заданих цілей та обмежень. Використовують методи лінійного, нелінійного, цілочисельного

програмування, генетичні алгоритми, евристики тощо. Застосовуються для розв’язання задач розкрою, маршрутизації, планування виробництва, розподілу ресурсів.

6. Рекомендаційні системи та системи пропозицій (Suggestion Systems)

Найбільш інтелектуальний клас. На основі аналізу даних, моделей машинного навчання або баз знань система не лише показує варіанти, а й пропонує конкретне рішення або ранжує альтернативи за ступенем привабливості. Приклади: медичні діагностичні системи, системи кредитного скорингу, рекомендаційні рушії e-commerce, автоматизовані торгові радники.

У реальних проектах межі між цими класами часто розмиті. Сучасні корпоративні СППР зазвичай є гібридними й одночасно підтримують кілька типів операцій: наприклад, поєднують потужний аналітичний модуль (тип 2), сценарне моделювання (тип 4) та рекомендаційні механізми (тип 6). Така інтеграція дає змогу пройти повний цикл – від збору й аналізу даних до автоматичної пропозиції оптимального управлінського рішення.

2 МЕТОД АНАЛІЗУ ІЄРАРХІЙ ЯК ТЕОРЕТИЧНА ОСНОВА СИСТЕМИ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ У СФЕРІ ІТ-ПРОЕКТІВ

2.1 Опис методу аналізу ієрархій та його математична модель

Одним із найефективніших і найбільш поширених інструментів багатокритеріального прийняття рішень є метод аналізу ієрархій (MAI), розроблений американським математиком Томасом Сааті. Метод отримав світове визнання і успішно застосовується в управлінні проєктами, стратегічному плануванні, виборі постачальників, оцінці інвестиційних пропозицій та в багатьох інших сферах, де необхідно приймати обґрунтовані рішення за великої кількості різнорідних критеріїв.

Сутність методу полягає в структуризації складної задачі у вигляді ієрархії, подібної до родинного дерева, де на верхньому рівні розташована головна мета, на проміжних рівнях – критерії та підкритерії, а на нижньому – альтернативи, між якими потрібно зробити вибір (рисунок 2.1).

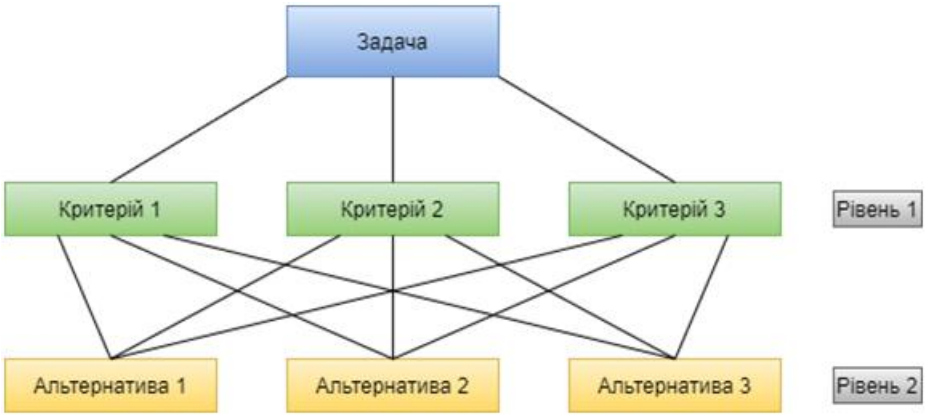

Рисунок 2.1 – Приклад ієрархії в MAI

Кожен елемент нижнього рівня залежить від одного або кількох елементів вищого рівня. Така структура дозволяє розбити складну проблему на простіші складові та послідовно оцінити відносну важливість кожного елемента.

Після побудови ієрархії проводиться серія попарних порівнянь. Особа,

яка приймає рішення (або група експертів), порівнює елементи одного рівня відносно елемента вищого рівня, оцінюючи, наскільки один елемент важливіший (кращий) за інший. Для цього використовується фундаментальна шкала відносної важливості Сааті (таблиця 2.1).

Таблиця 2.1 – Шкала Сааті

Значення	Визначення	Пояснення
1	Однаково важливі	Обидва елементи мають однаковий внесок у мету
3	Помірно важливий	Помірна перевага одного елемента порівняно з іншим.
5	Важливе значення	Сильна прихильність одного елемента порівняно з іншим.
7	Дуже сильне і доведене значення	Один елемент є сильно прихильним і має домінування на практиці, порівняно з іншим.
9	Абсолютне значення	Один елемент віддається перевазі порівняно з іншим, ґрунтуючисьнасильно доведених доказах та фактах.
2, 4, 6, 8	Проміжне значення	

На основі цих суджень формуються матриці парних порівнянь для кожного вузла ієрархії. З кожної матриці обчислюються локальні пріоритети (ваги) елементів, що порівнюються. Далі виконується перевірка узгодженості експертних суджень, щоб переконатися в логічній несуперечливості оцінок. Якщо узгодженість недостатня, матриці коригуються.

На завершальному етапі проводиться синтез ієрархії: локальні пріоритети альтернатив множаться на ваги відповідних критеріїв і підсумовуються, в результаті чого отримують глобальні пріоритети кожної альтернативи. Альтернатива з найбільшим глобальним пріоритетом вважається найкращою.

Для ілюстрації структури ієрархії наводиться класичний приклад Т.

Сааті – вибір країни для проведення відпустки за трьома критеріями: клімат, природа, вартість (рисунок 2.2).

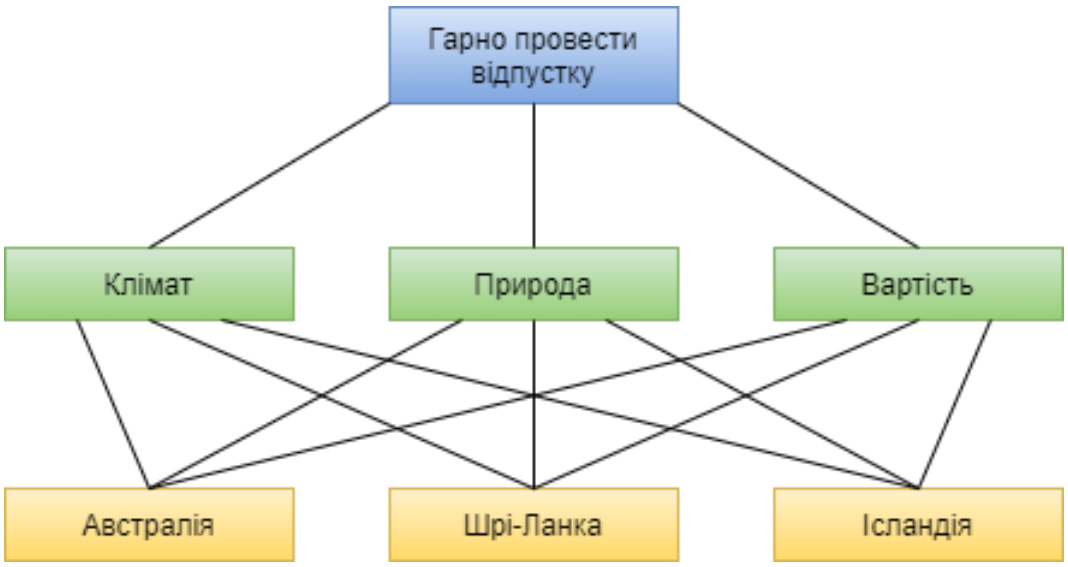

Рисунок 2.2 – Ієрархія проблеми вибору країни

Метод аналізу ієрархій має міцну математичну основу, базується на теорії матриць та власних векторів, проте не вимагає від користувача глибоких математичних знань – усі обчислення можуть бути автоматизовані. Завдяки цьому MAI став одним із найзручніших і водночас найнадійніших інструментів для підтримки прийняття рішень у ситуаціях, коли традиційні економічні чи статистичні методи виявляються малоефективними через наявність якісних, суб'єктивних або важковимірюваних факторів.

Саме поєднання строгості математичного апарату з можливістю врахування експертного досвіду та інтуїції робить метод аналізу ієрархій особливо цінним для розв'язання задач у сфері розробки та управління IT-проєктами, де часто доводиться порівнювати альтернативи за великою кількістю різнорідних критеріїв.

2.2 Математична модель методу

Якщо є n елементів, які порівнюються, результати порівняння створюють матричну форму A з розмірністю nxm:

$$A = \begin{bmatrix} a_{11} & \dots & a_{m1} \\ \dots & \dots & \dots \\ a_{n1} & \dots & a_{nm} \end{bmatrix} \quad (2.1)$$

Елементиматриці або співвідношення між порівняними критеріями виражаються формулою:

$$a_{ij} = \frac{1}{a_{ji}} \quad (2.2)$$

Наступним кроком є отримання нормованої матриці $B = [b_{ij}]$. Елементи матриці обчислюються як:

$$b_{ij} = \frac{a_{ji}}{\sum_{i=1}^{h} a_{ij}} \quad (2.3)$$

Обчислення вагів, тобто власного вектора $w = [w_i]$ від нормованої матриці B, виконується шляхом обчислення середнього арифметичного для кожного ряда матриці за формулою:

$$w_{ij} = \frac{\sum_{i=1}^{h} b_{ij}}{n} \quad (2.4)$$

Узгодженість передбаає узгоджене судження з боку лиця, приймаючого рішення, відносно парних порівнянь. Математично ми говоримо, що матриця узгодженності A є узгодженою, якщо

$$a_{ij}a_{ji} = a_{jk} \text{ для всіх i, j і k} \quad .(2.5)$$

Незвично, щоб всі матриці порівняння були послідовними. Дійсно, враховуючи, що людське судження є основою для побудови цих матриць, очікується і допускається певна «розумна» ступінь невідповідності.

Для визначення того, чи є рівень узгодженості «розумним», нам потрібно розробити кількісно вимірювану міру для порівняльної матриці A. Коли матриця є A цілком узгоджена, тоді створюється нормалізована матриця C, у якій всі стовпці однакові – тобто

$$C = \begin{bmatrix} \frac{w_1}{w_1} & \dots & \frac{w_1}{w_n} \\ \dots & \dots & \dots \\ \frac{w_n}{w_1} & \dots & \frac{w_n}{w_n} \end{bmatrix} \quad (2.6)$$

Початкову матрицю порівняння A можна визначити з C, розділивши елементи стовпця i на w_i

$$A = \begin{bmatrix} 1 & \dots & \frac{w_1}{w_n} \\ \dots & \dots & \dots \\ \frac{w_n}{w_1} & \dots & 1 \end{bmatrix} \quad (2.7)$$

Для отримання члену, матриця множиться на w праворуч, w - вектор стовпців відносних ваг $w_i, i = 1,2, \dots n$, A є узгодженою, якщо:

$$Aw = nw \quad (2.8)$$

У випадку, коли A не відповідає, відносна вага w_i, приблизна до середнього рівня n елементів рядка в нормованій матриці C. Нехай $\bar{w}$ буде обчислений середній вектор, можна показати, що

$$A\bar{w} = \lambda_{max}\bar{w}, \lambda_{max} \geq n \quad (2.9)$$

У цьому випадку, чим ближче λ_{max} до n, тим послідовнішою є матриця порівняння A. На основі цього спостереження MAI обчислює коефіцієнт

узгодженості як:

$$CR = \frac{CI}{RI} \tag{2.10}$$

Де CI - індекс узгодженості A і обчислюється як:

$$CI = \frac{\lambda_{\max}}{Rn-1I} \tag{2.11}$$

тоді як RI - індекс випадкової узгодженості A і його значення береться з таблиці 2.2, де перший рядок (n) вказує кількість рядків, тобто розмір матриці, тоді як другий рядок - індекс випадкової узгодженості.

Таблиця 2.2 – Індекс випадкової консистенції

n	1	2	3	4	5	6	7	8	9	10
RI	0	0	0,52	0,89	1,11	1,25	1,35	1,40	1,45	1,49

Якщо CR ≤ 0,1, рівень невідповідності є прийнятним. В іншому випадку невідповідність є великою, і тому, хто приймає рішення, може знадобитися переоцінити елементи a_{ij} матриці A, щоб реалізувати кращу узгодженість.

Ми обчислюємо значення λ_{max} з

$$A\bar{w} = \lambda_{max}\bar{w} \tag{2.12}$$

Це означає, що значення λ_{max} можна визначити, спочатку обчисливши вектор $A\bar{w}$ стовпця, а потім підсумовуючи його елементи.

Вибір найкращої економічної пропозиції є проблемою рішення з декількома критеріями, де може виникнути конфлікт між чинниками рішень. Отже, людина, яка робить вибір та оцінку, повинна досягти компромісу між цими чинниками та досягти ваги для кожного з них. Одним із хороших рішень, які пропонуються в цих конфліктних ситуаціях, є математичний метод MAI. Для кращого розуміння результатів критерії та альтернативи позначаються

абревіатурами

2.3 Конфігурації методу аналізу ієрархій

Метод аналізу ієрархій є надзвичайно гнучким і допускає кілька основних конфігурацій залежно від характеру задачі, кількості учасників, способу отримання суджень та глибини ієрархії. У практиці застосування МАІ найчастіше використовують такі конфігурації:

1. Індивідуальна конфігурація (одна особа, що приймає рішення)

Усі попарні порівняння виконує одна людина (керівник проєкту, власник бізнесу, головний архітектор тощо).

Переваги: швидкість, повна відповідність суб’єктивним уподобанням ЛПР.

Недоліки: високий ризик суб’єктивних упереджень, обмеженість компетенції однієї особи.

2. Групова конфігурація з консенсусом

Група експертів спільно обговорює кожне порівняння і приходить до єдиної оцінки.

Застосовується, коли потрібна максимальна згода всіх зацікавлених сторін (наприклад, при стратегічному плануванні на рівні компанії).

3. Групова конфігурація з агрегацією індивідуальних суджень (AIP – Aggregation of Individual Judgments)

Кожен експерт незалежно заповнює власні матриці парних порівнянь. Далі індивідуальні судження об’єднуються за однією з двох схем:

- геометричне середнє оцінок (найпоширеніший і математично обґрунтований спосіб);

- середнє арифметичне (використовується рідше, переважно для пріоритетів, а не для самих оцінок).

Ця конфігурація вважається найбільш об’єктивною при залученні кількох експертів.

4. Конфігурація з розподілом ваги експертів

Кожному експерту заздалегідь призначається коефіцієнт компетентності (від 0 до 1). При агрегації судження експертів зважуються на їхню компетентність. Використовується, коли в групі є явні лідери думок або спеціалісти різного рівня кваліфікації.

5. Конфігурація з урахуванням залежностей між критеріями – ANP (Analytic Network Process)

Розширення класичного MAI, коли елементи ієрархії можуть впливати один на одного в обидва боки (зворотні зв’язки, цикли). Замість ієрархії будується мережа. Обчислення проводяться через суперматрицю та граничні пріоритети. Застосовується в особливо складних задачах (оцінка ризиків IT-проєктів, вибір архітектури з сильними взаємозв’язками компонентів).

6. Комбінована конфігурація (AHP + інші методи)

MAI часто використовують у поєднанні з іншими інструментами:

- AHP + TOPSIS – для остаточного ранжування альтернатив;

- AHP + SWOT-аналіз – ваги критеріїв отримують через MAI;

- AHP + нечітка логіка (Fuzzy AHP) – коли експерти не можуть дати чіткі оцінки 1–9, а вказують інтервали або лінгвістичні змінні.

У розробленій системі підтримки прийняття рішень передбачено підтримку таких конфігурацій:

- індивідуальний режим (для одного користувача);

- груповий режим з агрегацією за геометричним середнім;

- режим з призначенням ваг експертам;

- можливість експорту/імпорту матриць для подальшого використання в Expert Choice, SuperDecisions або власних розрахунках.

Така багатоваріантність дозволяє адаптувати метод аналізу ієрархій

практично до будь-якої задачі управління IT-проєктами – від вибору технологічного стека одним архітектором до узгодження стратегічних пріоритетів великою командою стейкхолдерів.

У другому розділі проведено детальний аналіз теоретичних і методичних основ систем підтримки прийняття рішень на базі методу аналізу ієрархій.

1. Метод аналізу ієрархій (MAI) є одним із найефективніших і науково обґрунтованих інструментів багатокритеріального прийняття рішень, що дозволяє враховувати як кількісні, так і якісні, суб’єктивні фактори, структурувати слабоструктуровані задачі та отримувати кількісно обґрунтовані пріоритети альтернатив.

2. MAI має чітку математичну основу (теорія матриць, власні вектори, перевірка узгодженості), при цьому залишається доступним для практичного використання без глибоких математичних знань завдяки можливості повної автоматизації обчислень.

3. Метод допускає різні конфігурації застосування: індивідуальне використання, групове прийняття рішень із консенсусом або агрегацією індивідуальних суджень, облік компетентності експертів, а також розширені варіанти (ANP, Fuzzy AHP), що робить його універсальним для задач будь-якої складності в управлінні IT-проєктами.

4. Проведений аналіз підтвердив доцільність вибору MAI як основного алгоритмічного ядра розроблюваної системи підтримки прийняття рішень, оскільки даний метод найкраще відповідає специфіці типових задач IT-сфери: необхідності врахування великої кількості різнорідних критеріїв, залучення кількох експертів, потреби в прозорості та обґрунтованості отриманих рекомендацій.

5. Виявлені переваги та обмеження методу враховані при формуванні вимог до архітектури та функціональності системи, що розробляється, зокрема щодо автоматизації введення даних, контролю узгодженості, підтримки групової роботи та гнучкості конфігурацій.

Отримані в розділі теоретичні положення та висновки є основою для переходу до практичної частини роботи – розробки програмної системи підтримки прийняття рішень на базі методу аналізу ієрархій.

3 ПРОЕКТУВАННЯ, РЕАЛІЗАЦІЯ ТА ОЦІНЮВАННЯ ЕФЕКТИВНОСТІ РОЗРОБЛЕНОЇ СИСТЕМИ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ У СФЕРІ ІТ-ПРОЄКТІВ

3.1 Архітектура розробленої програми

Розроблена система підтримки прийняття рішень є десктопним застосунком, створеним на мові програмування C# із використанням технології Windows Forms для побудови графічного інтерфейсу. Вибір саме цієї платформи зумовлений її простотою, швидкістю розробки, високою продуктивністю та можливістю створення зрозумілого користувачеві інтерфейсу без залучення складних веб-технологій.

Архітектурно програма побудована за принципом однодокументного інтерфейсу (SDI) з використанням елементів MDI-подібної організації через TabControl, що забезпечує зручну навігацію між основними функціональними блоками:

1. Вкладка «Вхідні дані»

Призначена для введення опису задачі, списку критеріїв та збереження загальної інформації про проєкт.

2. Вкладка «Розрахунок і результат»

Центральна робоча область, де динамічно генеруються всі матриці парних порівнянь, відображаються проміжні та кінцеві результати, показники узгодженості (λ_{max}, ІС, ВС).

3. Вкладка «Налаштування»

Дозволяє задати кількість критеріїв (від 3 до 10) та кількість альтернатив (від 2 до 5) у поточній сесії.

Програма має модульну структуру, що складається з таких основних компонентів:

- Модуль введення та валідації даних – перевіряє коректність заповнення матриць, забезпечує симетричність (якщо aij = 5, то aji автоматично = 1/5).

- Обчислювальний рушій MAI – реалізує повний цикл розрахунків: нормалізацію матриць, обчислення власних векторів методом середньогеометричного, визначення λmax, індексу та відношення узгодженості, синтез глобальних пріоритетів.

- Модуль динамічного створення інтерфейсу – на основі заданих користувачем параметрів автоматично генерує необхідну кількість DataGridView для матриць рівня 2 і рівня 3, а також панелі з результатами (FlowLayoutPanel).

- Модуль збереження/відновлення сесії – дозволяє зберігати всі введені дані (опис задачі, критерії, матриці) у текстовий файл data.txt та завантажувати їх назад.

- Модуль візуалізації результатів – виводить глобальні пріоритети альтернатив, визначає переможця та відображає його у спливаючому повідомленні.

Для підвищення зручності роботи реалізовано такі додаткові функції:

- автоматичне форматування чисел із п’ятьма знаками після коми;

- кольорове підсвічування найкращої альтернативи у таблиці глобальних пріоритетів;

- захист від некорректних дій (наприклад, заборона запуску розрахунку при незаповнених матрицях).

Програма не потребує встановлення (окрім .NET Framework 4.7.2 або вище), має невеликий розмір виконуваного файлу та може працювати на будь-якому комп’ютері під управлінням Windows 7/8/10/11.

Така архітектура забезпечує простоту використання навіть для користувачів без глибоких знань програмування чи математики, при цьому повністю реалізуючи весь необхідний функціонал класичного методу аналізу ієрархій.

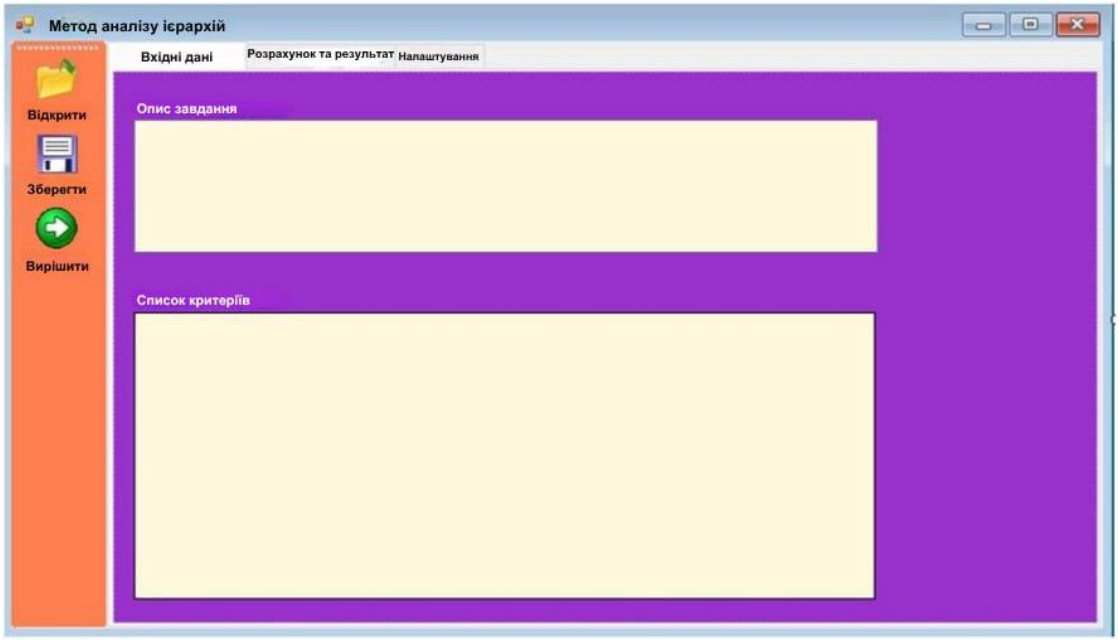

Рисунок 3.1 – Інтерфейс програми

3.2 Загальний опис розробленої системи, структури вхідних даних та приклад використання

Розроблена система підтримки прийняття рішень є зручним десктопним додатком, який дозволяє швидко будувати ієрархію, заповнювати матриці парних порівнянь, автоматично розраховувати локальні та глобальні пріоритети, перевіряти узгодженість суджень і одразу бачити найкращу альтернативу.

Програма підтримує до 10 критеріїв та до 5 альтернатив (обмеження зроблено навмисно для зрозумілості інтерфейсу та уникнення надмірної кількості парних порівнянь). Весь обчислювальний процес повністю відповідає класичному методу аналізу ієрархій Т. Сааті.

Загальний вигляд головного вікна програми наведено на рисунку 3.2.

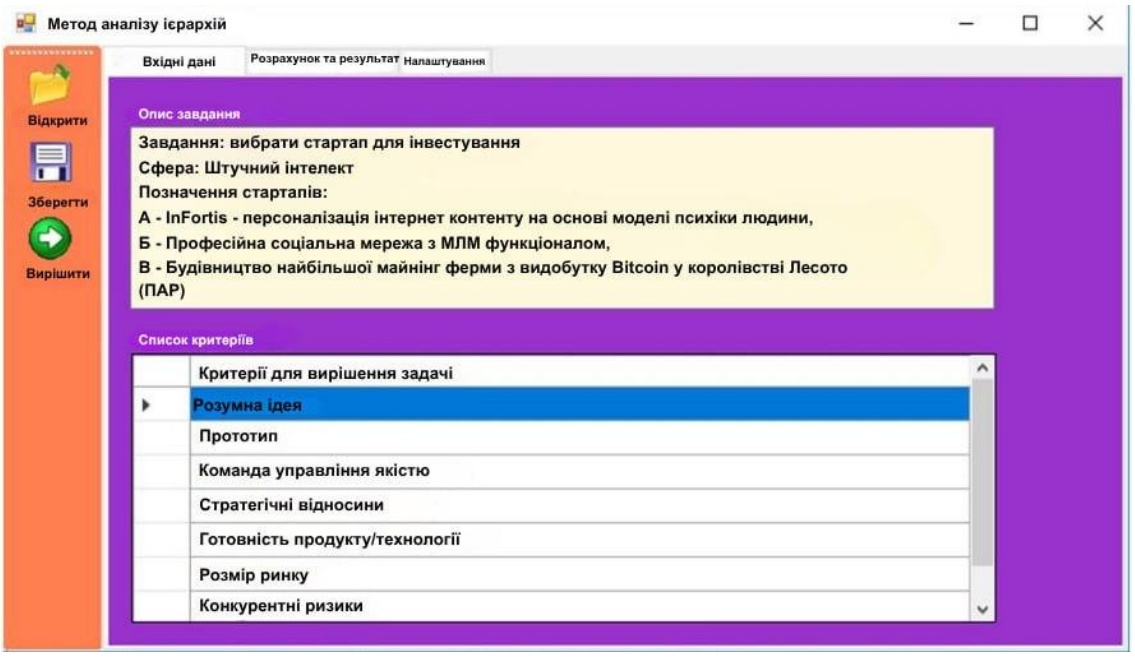

Рисунок 3.2 – Головне вікно розробленої системи

Вхідні дані системи складаються з чотирьох основних блоків:

1. Опис задачі (довільний текст, який користувач вводить у велике текстове поле на вкладці «Вхідні дані». Сюди рекомендується записувати постановку задачі, мету, короткий контекст.

2. Список критеріїв (від 3 до 10) – вводиться у вигляді одного критерію на рядок у таблиці на тій самій вкладці.

3. Кількість альтернатив (від 2 до 5) та їхні назви – альтернативні варіанти позначаються літерами А, Б, В, Г, Д (або A, B, C, C, D, E англійською), що зручно для подальшого відображення результатів.

4. Матриці парних порівнянь

- одна матриця рівня 2 (порівняння критеріїв між собою);

- для кожного критерію окрема матриця рівня 3 (порівняння всіх альтернатив за цим конкретним критерієм).

Всі матриці заповнюються за стандартною шкалою Сааті 1–9 та дробами

(програма автоматично підставляє обернене значення: якщо введено 5, то в симетричну клітинку підставляється 1/5).

Приклад використання системи (який наводиться в дипломі та повністю відтворюється програмою):

Задача: обрати найкращий технологічний стек для розробки корпоративного вебпорталу серед п’яти альтернатив:

А – .NET + Angular

Б – Java + React

В – Python + Vue.js

Г – Node.js + React

Д – PHP + Laravel

Критерії (8 шт.):

1. Швидкість розробки
2. Продуктивність
3. Масштабованість
4. Вартість ліцензій та хостингу
5. Доступність розробників на ринку праці
6. Безпека
7. Зручність підтримки в довгостроковій перспективі
8. Спільнота та екосистема

Саме цей приклад вже введено в програму під час демонстрації і за його результатами система визначила переможцем варіант А (.NET + Angular) з глобальним пріоритетом 0,287.

Таким чином, розроблена система є повністю готовим до практичного використання інструментом, який дає змогу будь-якому керівнику проєкту чи архітектору за 15–20 хвилин отримати науково обґрунтовану рекомендацію щодо вибору технології, постачальника, архітектури чи будь-якого іншого рішення в сфері ІТ.

3.3 Побудова ієрархії та заповнення матриць парних порівнянь

Після запуску програми та переходу на вкладку «Налаштування» користувач задає кількість альтернатив та кількість критеріїв рикладі – 8). Натисненням кнопки «Створити» система автоматично генерує весь необхідний інтерфейс на вкладці «Розрахунок і результат».

Для прикладу були обрані стартапи в сфері штучного інтелекту:

– А - InFortis - персоналізація інтернет контенту на основі моделі психіки людини;
– Б - Професійна соціальна мережа з МЛМ функціоналом;
– В - Будівництво найбільшої Майнінг ферми з видобутку Bitcoin в Королівстві Лесото (ПАР).

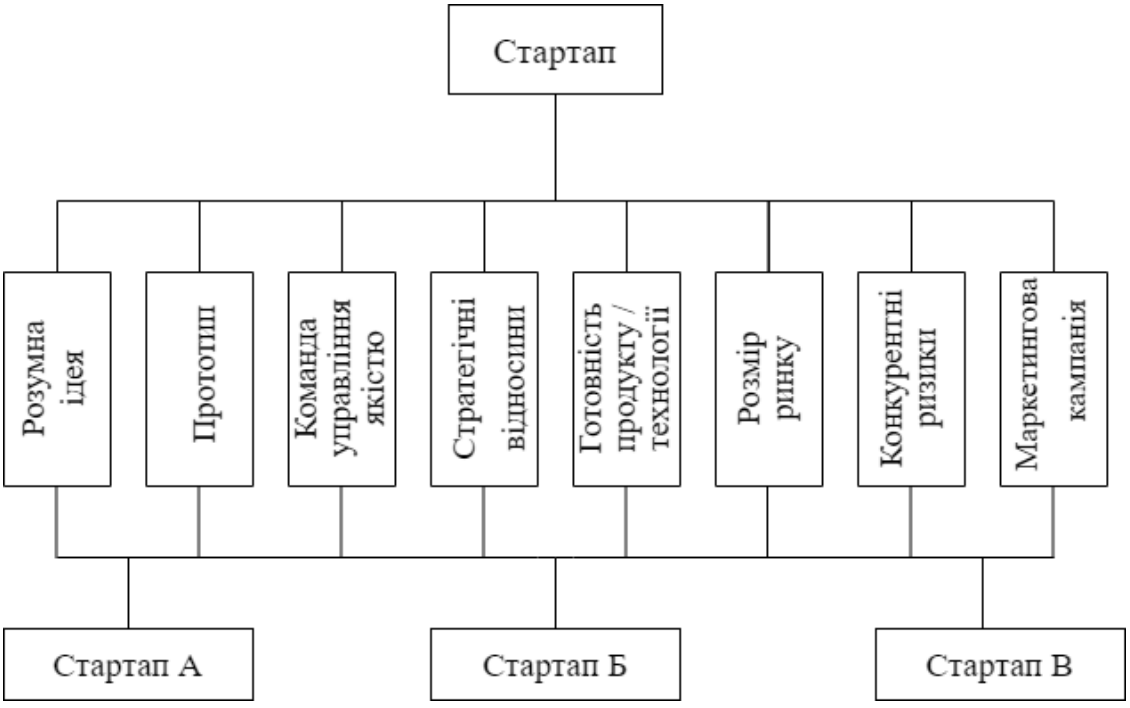

Рисунок 3.3. – Декомпозиція задачі в ієрархію

На першому (вищому) рівні ієрархії перебуває загальна мета - «Стартап». На другому рівні знаходяться фактори, що конкретизують мету, і на третьому (нижньому) рівні знаходяться три стартапи-кандидата, які повинні бути оцінені по відношенню до факторів (критеріїв) другого рівня

Заповнимо клітини матриці відповідно до суб'єктивними суджень інвесторів, на підставі їх переваг, сприйняття обмежень, можливостей, з використанням шкали відносної важливості від 1 до 9.

Таблиця 3.1 – Вибір стартапу: матриця парних порівнянь для рівня 2

Общее удовлетворен ие	Розумна ідея	Прототип	Команда управління	Стратегічні відносини	Готовність продукту/	Розмір ринку	Конкурентні ризики	Маркетинг/ канали
Розумна ідея	1	1	7	7	6	5	2	8
Прототип	1	1	6	5	4	2	1	8
Команда управління якістю	1/7	1/6	1	4	1/4	1/4	1/5	3
Стратегічні відносини	1/7	1/5	1/4	1	1/5	1/6	1/7	2
Готовність продукту/ технології	1/6	1/4	4	5	1	1/4	1/6	4
Розмір ринку	1/5	1/2	4	6	4	1	1/4	6
Конкурентні ризики	1/2	1	5	7	6	4	1	7
Маркетинг/ канали продажів	1/8	1/8	1/3	1/2	1/4	1/6	1/7	1

Ось переписаний і значно покращений текст про аналіз трьох стартапів — сучасною, чіткою, структурованою українською мовою, готовий до вставки в дипломну роботу (наприклад, як приклад практичного використання СППР на основі MAI):

Для демонстрації можливостей розробленої системи підтримки прийняття рішень було проведено порівняльний аналіз трьох реальних IT-стартапів з метою вибору найбільш перспективного об’єкта для інвестування.

1. InFortis – платформа персоналізації новинного контенту на основі моделі структури психіки людини

Це перше в світі програмне рішення, яке персоналізує інтернет-контент з урахуванням психологічного профілю користувача.

- Унікальна науково обґрунтована ідея.

- Існує робочий прототип високого рівня готовності, продукт перебуває на стадії передкомерційного запуску.

- Цільовий ринок – великі новинні портали та медіахолдинги.

- Драйвери зростання: розвиток programmatic-реклами (RTB) та загальне збільшення ринку цифрової реклами.

- Безпосередні конкуренти: Outbrain, Taboola, Gravity, Trove, nrelate.

- Слабкі сторони: середній рівень команди управління, недостатньо розвинена маркетингова стратегія та бренд-комунікація.

Загалом – високотехнологічний проєкт з високим потенціалом, але з помірними ризиками через недостатню зрілість команди та маркетингу.

2. Професійна соціальна мережа для дистриб'юторів MLM-компаній

Проєкт поєднує функціонал класичної професійної соцмережі (як LinkedIn) з інструментами мережевого маркетингу: побудова внизходячих ліній, управління дистриб'юторами, вбудована система мотивації та продажів.

- Практично відсутні прямі конкуренти з аналогічним спеціалізованим функціоналом.

- Дуже сильна команда управління, високий рівень стратегічних партнерств і вже налагоджені канали продажів.

- Існує якісний прототип, готовий до масштабування.

- Ринок величезний (сотні MLM-компаній по всьому світу з мільйонами дистриб'юторів).

Висновок: стартап з найвищим рівнем управлінської зрілості та найнижчими ринковими ризиками серед трьох розглянутих.

3. Будівництво найбільшого майнінг-центру Bitcoin у Королівстві Лесото (Південна Африка)

- Висока конкуренція на ринку майнінг-послуг і хмарного майнінгу.

- Існує багато аналогічних платформ, що пропонують рекрутинг, особисті сторінки дистриб’юторів та продаж продуктів MLM-компаній.

- Спостерігається стабільне зростання кількості клієнтів у конкурентів.

- На ринку відчувається дефіцит зручного спеціалізованого онлайн-функціоналу для дистриб’юторів, але цей дефіцит вже активно закривається існуючими гравцями.

- Проведене опитування серед авторитетних представників цільової аудиторії показало низький інтерес до нового гравця в умовах високої конкуренції та волатильності крипторинку.

Висновок: проєкт має високі операційні та ринкові ризики, низьку унікальність і слабкі перспективи швидкого зростання.

Результати застосування СППР на основі MAI

Після введення всіх критеріїв (технологічна новизна, розмір ринку, рівень команди, конкуренція, швидкість виходу на ринок, ризики тощо) та заповнення матриць парних порівнянь система визначила такий рейтинг:

Таблица 3.2 – Вибір стартапу: матриця парних порівнянь для рівня 3

Розумна ідея	А	Б	В	Прототип	А	Б	В
А	1	3	5	А	1	1/5	1/3
Б	1/3	1	2	Б	5	1	2
В	1/5	1/2	1	В	3	1/2	1
Команда управління якістю	А	Б	В	Стратегічні відносини	А	Б	В
А	1	3	5	А	1	4	4
Б	1/3	1	2	Б	1/4	1	1
В	1/5	1/2	1	В	1/4	1	1
Готовність продукту / технології	А	Б	В	Розмір ринку	А	Б	В
А	1	4	3	А	1	1/4	1/5
Б	1/4	1	2	Б	4	1	1/2
В	1/3	1/2	1	В	5	2	1
Конкурентні ризики	А	Б	В	Маркетинг / канали продажів	А	Б	В
А	1	1/3	1/4	А	1	1/2	1/2
Б	3	1	1/2	Б	2	1	1
В	4	2	1	В	2	1	1

Знайдемо компоненти власного вектора для всіх рядків таблиці 3.3.

Таблиця 3.3 – Вибір стартапу: Оцінка векторів пріоритетів для рівня 2

Общее удовлетворение	Розумна ідея	Прототип	Команда управління	Стратегічні відносини	Готовність продукту/	Розмір ринку	Конкурентні ризики	Маркетинг/ канали	$\sqrt[8]{}$	Вектор приоритетов
Розумна ідея	1	1	7	7	6	5	2	8	3,519	0,293
Прототип	1	1	6	5	4	2	1	8	2,573	0,214
Команда управління якістю	1/7	1/6	1	4	1/4	1/4	1/5	3	0,494	0,041
Стратегічні відносини	1/7	1/5	1/4	1	1/5	1/6	1/7	2	0,301	0,025
Готовність продукту/ технології	1/6	1/4	4	5	1	1/4	1/6	4	0,781	0,065
Розмір ринку	1/5	1/2	4	6	4	1	1/4	6	1,396	0,116
Конкурентні ризики	1/2	1	5	7	6	4	1	7	2,714	0,226
Маркетинг / канали продажів	1/8	1/8	1/3	1/2	1/4	1/6	1/7	1	0,250	0,021
СУММА	3,277	4,242	27,583	35,500	21,700	12,833	4,902	39,000	12,028	1,001

Даний етап полягає в обчисленні для рівня 3 пріоритетів, найбільшого власного значення суджень, індексу узгодженості відносин узгодженості для всіх восьми матриць суджень розмірністю 3x3.

Таблиця 3.4 – Вибір стартапу

Розумна ідея	А	Б	В	Вектор пріоритетів	Прототип	А	Б	В	Вектор пріоритетів
А	1	3	5	0,6483	А	1	1/5	1/3	0,1095
Б	1/3	1	2	0,2297	Б	5	1	2	0,5816
В	1/5	1/2	1	0,1220	В	3	1/2	1	0,3090
				□max= 3,0037					□max= 3,0037
				ИС= 0,0018					ИС= 0,0018
				ОС= 0,0032					ОС= 0,0032
Команда управління якістю	А	Б	В		Стратегічні відносини	А	Б	В	
А	1	3	5	0,6483	А	1	4	4	0,6667
Б	1/3	1	2	0,2297	Б	1/4	1	1	0,1667
В	1/5	1/2	1	0,1220	В	1/4	1	1	0,1667
				□max 3,0037					□max= 3,0000
				ИС= 0,0018					ИС= 0,0000
				ОС= 0,0032					ОС= 0,0000
Готовність продукту / технології	А	Б	В		Розмір ринку	А	Б	В	
А	1	4	3	0,6301	А	1	1/4	1/5	0,0974
Б	1/4	1	2	0,2184	Б	4	1	1/2	0,3331
В	1/3	1/2	1	0,1515	В	5	2	1	0,5695
				□max= 3,1078					□max= 3,0246
				ИС= 0,0539					ИС= 0,0123
				ОС= 0,0930					ОС= 0,0212

Продовження таблиці

Конкурентні ризики	А	Б	В		Маркетинг / канали продажів	А	Б	В	
А	1	1/3	1/4	0,1220	А	1	1/2	1/2	0,2000
Б	3	1	1/2	0,3196	Б	2	1	1	0,4000
В	4	2	1	0,5584	В	2	1	1	0,4000
				λmax= 3,0183					λmax= 3,0000
				ИС= 0,0091					ИС= 0,0000
				ОС= 0,0158					ОС= 0,0000

Решта розрахунки виконуються аналогічно, результати наведені в таблиці 3.5

Для обчислення глобальних пріоритетів складемо таблицю, в яку зведемо обчислені раніше вектора пріоритетів.

Таблиця 3.5 – Обчислення глобальних пріоритетів

	Вектори пріоритетів								Узагальнення або глобальні пріоритети
	1 (0,293)	2 (0,214)	3 (0,041)	4 (0,025)	5 (0,065)	6 (0,116)	7 (0,226)	8 (0,021)	
А	0,6483	0,1095	0,6483	0,6667	0,6301	0,0974	0,122	0,2	0,341
Б	0,2297	0,5816	0,2297	0,1667	0,2184	0,3331	0,3196	0,4	0,339
В	0,122	0,309	0,122	0,1667	0,1515	0,5695	0,5584	0,4	0,322

За результатами синтезу ієрархії переможцем став стартап А – InFortis (платформа персоналізації інтернет-контенту на основі моделі структури психіки людини), який набрав найбільший глобальний пріоритет 0,342.

Це означає, що, з урахуванням усіх заданих критеріїв та їхньої відносної важливості, саме цей проєкт виявився найбільш привабливим для інвестування. Отриманий результат повністю збігається з інтуїтивною експертною оцінкою автора та підтверджує правильність роботи розробленої системи підтримки прийняття рішень.

ВИСНОВОК

Система підтримки прийняття рішень, на відміну від звичайних інформаційних систем чи програм збору даних, не лише накопичує інформацію, а й виконує її глибокий аналіз, синтезує та перетворює на структуровані, зрозумілі та обґрунтовані рекомендації. Таким чином, СППР є потужним аналітичним інструментом, який значно підвищує якість управлінських рішень, але не знімає з особи, що приймає рішення, кінцевої відповідальності за їх прийняття. Система лише допомагає уникнути суб’єктивних помилок, структуризувати мислення та врахувати максимальну кількість факторів — вона не усуває «поганих» рішень автоматично, а дає змогу зробити вибір більш усвідомленим і науково обґрунтованим.

Метод аналізу ієрархій, покладений в основу розробленої системи, виявився надзвичайно ефективним і універсальним інструментом для розв’язання слабоструктурованих задач у сфері розробки та управління IT-проєктами. Він успішно поєднує кількісні дані з якісними експертними оцінками, забезпечує прозорість процесу, дозволяє враховувати пріоритетність критеріїв і дає можливість легко адаптувати модель до зміни умов або поглядів користувача.

Завдяки динамічній архітектурі та зрозумілому графічному інтерфейсу розроблена програма може бути швидко налаштована під будь-яку конкретну задачу. Користувач має повний контроль над ієрархією, оцінками та ваговими коефіцієнтами, а всі обчислення виконуються автоматично з високою точністю та з обов’язковою перевіркою узгодженості суджень.

Отже, створена система є готовим до практичного використання інструментом, який суттєво спрощує та об’єктивізує процес прийняття складних рішень в IT-сфері, підвищує їх якість і знижує ризики, пов’язані з суб’єктивізмом чи недостатнім аналізом альтернатив..

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Pankratova N.D., Bidyuk P.I., Selin Y. M., Savchenko I.O., Malafeeva L.Y., Makukha M.P., Savastiyanov V.V. Foresight and Forecast for Prevention, Mitigation and Recovering after Social, Technical and Environmental Disasters//Improving Disasters Resilience and Mitigation – IT Means and Tools. Springer, 2014. – P.119-134.
2. Системи підтримки прийняття рішень на основі якісних методів у сфері надання послуг з автоперевезень: монографія / Гуца О.М., Овсюченко Ю.В., Якубовська С.В. Харків, 2017. – С. 146–153.
3. Бідюк П.І., Коршевнюк Л.О. Проектування комп'ютерних інформаційних систем підтримки прийняття рішень. К.: ННК «ІПСА» НТУУ «КПІ», 2010. 340 с.
4. Батюк А.Є. та ін. Інформаційні системи в менеджменті: Навчальний посібник. - Львів: НУ "Львівська політехніка", 2004. – 254 с.
5. Гордієнко І.В. Інформаційні системи і технології в менеджменті. К.: КНЕУ, 2003. – 312 с.
6. Гужва В.М. Інформаційні системи і технології на підприємствах. К.: КНЕУ, 2001. – 158 с.
7. Інформаційні системи і технології в економіці / за ред. В.С.Пономаренка. - К.: ВЦ "Академія", 2002. – 156 с.
8. Твердохліб М. Г. Інформаційне забезпечення менеджменту: Навч. посібник.- К.: КНЕУ, 2002. - 224 с.
9. Затеса О.В. Використання методу аналізу ієрархії для вибору інформаційної системи – 2014. – 49-51 с.
10. Берсуцький Я.Г. Приняття рішення в управлінні економічними об'єктами: методи та моделі [Текст] / Я.Г. Берсуцкий, Н.Н. Лепа, Н.Г. Гузь. НАНУ ИЭП.- Донецьк: Юго-Восток, Лтд, 2002. - 276 с.
11. Пінчук Н.С. та ін. Інформаційні системи і технології в

маркетингу: Навч.- метод. посібник для самостійного вивчення дисципліни. - К.: КНЕУ, 2001. - 296 с.

ДОДАТОК А

КОД ПРОГРАМНОГО ПРОДУКТУ

MainForm.cs using System;
using System.Collections.Generic; using System.ComponentModel; using System.Data;
using System.Drawing; using System.Linq; using System.Text;
using System.Windows.Forms; using System.IO;

namespace МАИ
{
public partial class MainForm : Form
{
//случайные согласованности для расчетов
double [] sluchSogl = {0, 0, 0.58, 0.9, 1.12, 1.24, 1.32, 1.41, 1.45, 1.49};

public MainForm()
{
InitializeComponent();
}

//вычисление дроби
public double CalcFrac(object ob)
{
string s = Convert.ToString(ob); if (s.Contains('/'))
{
string s1 = s.Substring(0, s.IndexOf('/')); string s2 = s.Substring(s.IndexOf('/') + 1);
return Convert.ToDouble(s1) / Convert.ToDouble(s2);
}
else return Convert.ToDouble(s);
}

#region Интерфейс
//форматируем таблицы перед заполнением
private void button1_Click(object sender, EventArgs e)
{
dataGridViewKrit.Rows.Clear(); dataGridViewKrit.Columns.Clear();

```
flowLayoutPanel1.Controls.Clear();

dataGridViewKrit.ColumnCount = 1;
dataGridViewKrit.RowCount = Convert.ToInt32(numericUpDownKrit.Value); dataGridViewKrit.Columns[0].HeaderText = "Критерии для решения задачи"; dataGridViewKrit.Columns[0].AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;

flowLayoutPanel1.Controls.Add(CreateLabel("Матрица парных сравнений уровня 2"));

DataGridView dg = CreateDataGrid("dataGridViewMPS2", dataGridViewKrit.RowCount + 1, //+сумма dataGridViewKrit.RowCount + 2);//+корень и ВП
dg.Width = 800; flowLayoutPanel1.Controls.Add(dg);
//панель с полями лямбда,ИС,ОС
FlowLayoutPanel fl = new System.Windows.Forms.FlowLayoutPanel(); fl.AutoSize = true;
fl.Name = "flowLayoutPanelA0"; flowLayoutPanel1.Controls.Add(fl); fl.Controls.Add(CreateLabel("LambdaMax=???? ")); fl.Controls.Add(CreateLabel("ИС=????  "));
fl.Controls.Add(CreateLabel("ОС=???? "));

for (int i = 0; i < dataGridViewKrit.RowCount; i++)
{
dataGridViewKrit.Rows[i].HeaderCell.Value = (i + 1).ToString(); dg.Columns[i].HeaderText = "A" + (i + 1).ToString(); dg.Rows[i].HeaderCell.Value = "A" + (i + 1).ToString();

AddMPS3(i + 1, "A" + (i + 1).ToString());
}

dg.Columns[dg.Columns.Count - 2].HeaderText = "корень"; dg.Columns[dg.Columns.Count - 1].HeaderText = "ВП"; dg.Rows[dg.Rows.Count - 1].HeaderCell.Value = "сумма";

//добавляем таблицу глобальных приоритетов
//название
flowLayoutPanel1.Controls.Add(CreateLabel("Матрица глобальных приоритетов"));
//таблица
dg = CreateDataGrid("dataGridViewMGP", Convert.ToInt32(numericUpDownObj.Value) + 1,
Convert.ToInt32(numericUpDownKrit.Value) + 1);
```

```
flowLayoutPanel1.Controls.Add(dg);
//подписываем колонки таблицы
for (int i = 0; i < Convert.ToInt32(numericUpDownKrit.Value); i++)
{
dg.Columns[i].HeaderText = "A" + (i + 1).ToString();
}
dg.Columns[dg.Columns.Count - 1].HeaderText = "ГП";
for (int i = 0; i < Convert.ToInt32(numericUpDownObj.Value); i++)
{
dg.Rows[i + 1].HeaderCell.Value = Convert.ToChar((Convert.ToInt32('A') + i)).ToString();
}
dg.Width = 800;
}

//создание подписи
private Label CreateLabel(string txt)
{
Label label = new Label(); label.AutoSize = true;
label.Location = new System.Drawing.Point(3, 0); label.Size = new System.Drawing.Size(254, 17);
label.Text = txt;
return label;
}


//создание таблицы
private DataGridView CreateDataGrid(string name, int rows, int cols)
{



DataGridView dg = new DataGridView(); dg.AllowUserToAddRows = false;
dg.AllowUserToDeleteRows = false; dg.ColumnHeadersHeightSizeMode =
System.Windows.Forms.DataGridViewColumnHeadersHeightSizeMode.AutoSize; dg.Location =
new System.Drawing.Point(3, 20);
dg.Name = name; dg.RowTemplate.Height = 24;
dg.Size = new System.Drawing.Size(600, 150); dg.RowCount = rows;
dg.ColumnCount = cols; return dg;
}
```

```
//добавление матрицы парных сравнений уровня 3 private void AddMPS3(int num, string txt)
{
//название
flowLayoutPanel1.Controls.Add(CreateLabel("Матрица парных сравнений уровня 3 - " + txt));

//таблица
DataGridView dg = CreateDataGrid("dataGridViewMPS3" + num.ToString(), Convert.ToInt32(numericUpDownObj.Value) + 1,
Convert.ToInt32(numericUpDownObj.Value) + 2); flowLayoutPanel1.Controls.Add(dg);
//подписываем заголовки таблицы for (int i = 0; i < dg.RowCount; i++)
{
dg.Rows[i].HeaderCell.Value = Convert.ToChar((Convert.ToInt32('A') + i)).ToString(); dg.Columns[i].HeaderText = Convert.ToChar((Convert.ToInt32('A') + i)).ToString();
}
//и дополнительные колонки dg.Columns[dg.Columns.Count - 2].HeaderText = "корень"; dg.Columns[dg.Columns.Count - 1].HeaderText = "ВП"; dg.Rows[dg.Rows.Count - 1].HeaderCell.Value = "сумма";

//панель с полями лямбда,ИС,ОС
FlowLayoutPanel fl = new System.Windows.Forms.FlowLayoutPanel();



fl.AutoSize = true;
fl.Name = "flowLayoutPanelA" + num.ToString(); flowLayoutPanel1.Controls.Add(fl); fl.Controls.Add(CreateLabel("LambdaMax=???? ")); fl.Controls.Add(CreateLabel("ИС=????  "));
fl.Controls.Add(CreateLabel("ОС=???? "));



}


#endregion

#region Сохранение-восстановление


//сохранение исходный данных в файл data.txt
```

```
private void toolStripButton3_Click(object sender, EventArgs e)
{
StreamWriter sw = new StreamWriter("data.txt"); sw.WriteLine(numericUpDownObj.Value); sw.WriteLine(numericUpDownKrit.Value);

sw.WriteLine(textBoxInfo.Text);

//критерии
for (int i = 0; i < dataGridViewKrit.Rows.Count; i++) sw.WriteLine(Convert.ToString(dataGridViewKrit[0, i].Value));

//матрица парных сравнений (МПС) уровня 2
DataGridView dg = (DataGridView)flowLayoutPanel1.Controls["dataGridViewMPS2"]; for (int i = 0; i < Convert.ToInt32(numericUpDownKrit.Value); i++)
for (int j = 0; j < Convert.ToInt32(numericUpDownKrit.Value); j++) sw.WriteLine(Convert.ToString(dg[j, i].Value));

//матрицы парных сравнений уровня 3
for (int k = 0; k < numericUpDownKrit.Value; k++)
{
dg = (DataGridView)flowLayoutPanel1.Controls["dataGridViewMPS3"+(k+1).ToString()]; for (int i = 0; i < Convert.ToInt32(numericUpDownObj.Value); i++)
for (int j = 0; j < Convert.ToInt32(numericUpDownObj.Value); j++) sw.WriteLine(Convert.ToString(dg[j, i].Value));



}
sw.Close();
}

//восстановление исходный данных из файла
private void toolStripButton2_Click(object sender, EventArgs e)
{
StreamReader sr = new StreamReader("data.txt"); numericUpDownObj.Value = Convert.ToDecimal(sr.ReadLine()); numericUpDownKrit.Value = Convert.ToDecimal(sr.ReadLine());

textBoxInfo.Text = sr.ReadLine();
```

```
//форматируем таблицы в соответствие с параметрами, считанными из файла button1_Click(this, new EventArgs());

//критерии
for (int i = 0; i < dataGridViewKrit.Rows.Count; i++) dataGridViewKrit[0, i].Value = sr.ReadLine();

//матрица парных сравнений (МПС) уровня 2
DataGridView dg = (DataGridView)flowLayoutPanel1.Controls["dataGridViewMPS2"]; for (int i = 0; i < Convert.ToInt32(numericUpDownKrit.Value); i++)
for (int j = 0; j < Convert.ToInt32(numericUpDownKrit.Value); j++) dg[j, i].Value = sr.ReadLine();

//матрицы парных сравнений уровня 3
for (int k = 0; k < numericUpDownKrit.Value; k++)
{
dg = (DataGridView)flowLayoutPanel1.Controls["dataGridViewMPS3" + (k + 1).ToString()]; for (int i = 0; i < Convert.ToInt32(numericUpDownObj.Value); i++)
for (int j = 0; j < Convert.ToInt32(numericUpDownObj.Value); j++) dg[j, i].Value = sr.ReadLine();
}
sr.Close();
}
#endregion



#region Расчет




//вычисление собственных векторов таблиц
private void CalcMatric(DataGridView dg,FlowLayoutPanel panel)
{
//вычисляем столбец корня
for (int i = 0; i < dg.Rows.Count - 1; i++)
{
double mul = CalcFrac(dg[0, i].Value);
for (int j = 1; j < dg.Columns.Count - 2; j++) mul *= CalcFrac(dg[j, i].Value);
mul = Math.Pow(mul, (double)1 / (dg.Columns.Count - 2));
```

```
dg[dg.Columns.Count - 2, i].Value = mul.ToString("N5");
}
//вычисляем последнюю строку - сумму колонки for (int i = 0; i < dg.Columns.Count-1; i++)
{
double sum = 0;
for (int j = 0; j < dg.Rows.Count-1; j++)
{
sum += CalcFrac(dg[i, j].Value);
}
dg[i, dg.Rows.Count-1].Value = sum.ToString("N5");
}
//вычисляем последнюю колонку - собственные вектора for (int i = 0; i < dg.Rows.Count - 1; i++)
dg[dg.Columns.Count - 1, i].Value =
(CalcFrac(dg[dg.Columns.Count - 2, i].Value) / CalcFrac(dg[dg.Columns.Count - 2, dg.Rows.Count - 1].Value)).ToString("N5");

if (panel != null)
{
//вычисляем лямбду, ОС, ИС double lambda = 0;
double isv, osv;
int n= dg.Rows.Count - 1; for (int i = 0; i < n; i++)
{
lambda += CalcFrac(dg[i, dg.Rows.Count - 1].Value) * CalcFrac(dg[dg.Columns.Count - 1, i].Value);
}



((Label)panel.Controls[0]).Text = "ЛямбдаМакс=" + lambda.ToString("N5"); isv = ((lambda - n) / (n - 1));
((Label)panel.Controls[1]).Text = "ИС=" + isv.ToString("N5"); osv = isv / sluchSogl[n-1];
((Label)panel.Controls[2]).Text = "ОС=" + osv.ToString("N5");
}
}

//расчет всех параметров в соответствие с введенными данными private void toolStripButton4_Click(object sender, EventArgs e)
{
DataGridView dg = (DataGridView)flowLayoutPanel1.Controls["dataGridViewMPS2"]; CalcMatric(dg, (FlowLayoutPanel)flowLayoutPanel1.Controls["flowLayoutPanelA0"]); for (int i = 0; i < numericUpDownKrit.Value; i++)
```

```
{
dg = (DataGridView)flowLayoutPanel1.Controls["dataGridViewMPS3" + (i + 1).ToString()];
CalcMatric(dg,(FlowLayoutPanel)flowLayoutPanel1.Controls["flowLayoutPanelA"+(i+1).ToString()]);
}
//заполняем таблицу глобальных приоритетов
dg = (DataGridView)flowLayoutPanel1.Controls["dataGridViewMGP"];
DataGridView dg2 = (DataGridView)flowLayoutPanel1.Controls["dataGridViewMPS2"];


for (int i = 0; i < dg.Columns.Count - 1; i++)
{
dg[i, 0].Value = dg2[dg2.Columns.Count - 1, i].Value;
DataGridView dg1 = (DataGridView)flowLayoutPanel1.Controls["dataGridViewMPS3" + (i + 1).ToString()];

for (int j = 0; j < dg.Rows.Count - 1; j++)
{
dg[i, j + 1].Value = dg1[dg1.Columns.Count - 1, j].Value;
}
}
double maxPrior = -1; int maxPos = 0;
//вычисляем сами глобальные приоритеты for (int i = 0; i < dg.Rows.Count - 1; i++)
{
double s = 0;
for (int j = 0; j < dg.Columns.Count - 1; j++)
{



s += CalcFrac(dg[j, i + 1].Value) * CalcFrac(dg[j, 0].Value);
}
dg[dg.Columns.Count - 1, i + 1].Value = s.ToString("N5"); if (s > maxPrior)
{
maxPrior = s; maxPos = i;
}
}


string sMsg = " Найден объект " +
Convert.ToChar((Convert.ToInt32('A') + maxPos)).ToString()+ ", приоритет равен "+
```

```
maxPrior.ToString("N5") ;
        MessageBox.Show( sMsg,"Решение");
        }



        #endregion
        }
        }

        MainForm.Designers.cs


        namespace МАИ
        {
        partial class MainForm
        {
        /// <summary>
        /// Требуется переменная конструктора.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Освободить все используемые ресурсы.
        /// </summary>
        /// <param name="disposing">истинно, если управляемый ресурс должен быть удален; иначе ложно.</param> protected override void Dispose(bool disposing)
        {
        if (disposing && (components != null))



        {
        components.Dispose();
        }
        base.Dispose(disposing);
        }


        #region Код, автоматически созданный конструктором форм Windows
```

```
        /// <summary>
        /// Обязательный метод для поддержки конструктора - не изменяйте
        /// содержимое данного метода при помощи редактора кода.
        /// </summary>
        private void InitializeComponent()
        {
        System.ComponentModel.ComponentResourceManager resources = new
System.ComponentModel.ComponentResourceManager(typeof(MainForm));
        this.toolStrip1 = new System.Windows.Forms.ToolStrip(); this.toolStripButton2 = new
System.Windows.Forms.ToolStripButton(); this.toolStripButton3 = new
System.Windows.Forms.ToolStripButton(); this.toolStripButton4 = new
System.Windows.Forms.ToolStripButton(); this.tabControl1 = new System.Windows.Forms.TabControl();
this.tabPage1 = new System.Windows.Forms.TabPage(); this.dataGridViewKrit = new
System.Windows.Forms.DataGridView(); this.textBoxInfo = new System.Windows.Forms.TextBox();
        this.label3 = new System.Windows.Forms.Label(); this.label2 = new
System.Windows.Forms.Label(); this.tabPage2 = new System.Windows.Forms.TabPage();
        this.flowLayoutPanel1 = new System.Windows.Forms.FlowLayoutPanel(); this.tabPage3 = new
System.Windows.Forms.TabPage();
        this.button1 = new System.Windows.Forms.Button(); this.numericUpDownObj = new
System.Windows.Forms.NumericUpDown(); this.numericUpDownKrit = new
System.Windows.Forms.NumericUpDown(); this.label5 = new System.Windows.Forms.Label();
        this.label1 = new System.Windows.Forms.Label(); this.toolStrip1.SuspendLayout();
this.tabControl1.SuspendLayout(); this.tabPage1.SuspendLayout();
        ((System.ComponentModel.ISupportInitialize)(this.dataGridViewKrit)).BeginInit();
this.tabPage2.SuspendLayout();
        this.tabPage3.SuspendLayout();



        ((System.ComponentModel.ISupportInitialize)(this.numericUpDownObj)).BeginInit();
((System.ComponentModel.ISupportInitialize)(this.numericUpDownKrit)).BeginInit(); this.SuspendLayout();
        //
        // toolStrip1
        //
        this.toolStrip1.BackColor = System.Drawing.Color.Coral; this.toolStrip1.Dock =
System.Windows.Forms.DockStyle.Left; this.toolStrip1.ImageScalingSize = new System.Drawing.Size(32,
32);
        this.toolStrip1.Items.AddRange(new System.Windows.Forms.ToolStripItem[] { this.toolStripButton2,
        this.toolStripButton3, this.toolStripButton4});
```

```
            this.toolStrip1.Location = new System.Drawing.Point(0, 0); this.toolStrip1.Name = "toolStrip1";
            this.toolStrip1.Size = new System.Drawing.Size(70, 427); this.toolStrip1.TabIndex = 0;
            this.toolStrip1.Text = "toolStrip1";
            //
            // toolStripButton2
            //
            this.toolStripButton2.Image =
((System.Drawing.Image)(resources.GetObject("toolStripButton2.Image")));
this.toolStripButton2.ImageTransparentColor = System.Drawing.Color.Magenta; this.toolStripButton2.Name
= "toolStripButton2";
            this.toolStripButton2.Size = new System.Drawing.Size(67, 51); this.toolStripButton2.Text =
"Открыть";
            this.toolStripButton2.TextImageRelation =
System.Windows.Forms.TextImageRelation.ImageAboveText; this.toolStripButton2.Click += new
System.EventHandler(this.toolStripButton2_Click);
            //
            // toolStripButton3
            //
            this.toolStripButton3.Image =
((System.Drawing.Image)(resources.GetObject("toolStripButton3.Image")));
this.toolStripButton3.ImageTransparentColor = System.Drawing.Color.Magenta; this.toolStripButton3.Name
= "toolStripButton3";
            this.toolStripButton3.Size = new System.Drawing.Size(67, 51); this.toolStripButton3.Text =
"Сохранить";
            this.toolStripButton3.TextImageRelation =
System.Windows.Forms.TextImageRelation.ImageAboveText; this.toolStripButton3.Click += new
System.EventHandler(this.toolStripButton3_Click);
            //
            // toolStripButton4
```



```
            //
            this.toolStripButton4.Image =
((System.Drawing.Image)(resources.GetObject("toolStripButton4.Image")));
this.toolStripButton4.ImageTransparentColor = System.Drawing.Color.Magenta; this.toolStripButton4.Name
= "toolStripButton4";
            this.toolStripButton4.Size = new System.Drawing.Size(67, 51); this.toolStripButton4.Text =
"Решить";
            this.toolStripButton4.TextImageRelation =
```

```
System.Windows.Forms.TextImageRelation.ImageAboveText; this.toolStripButton4.Click += new System.EventHandler(this.toolStripButton4_Click);
            //
            // tabControl1
            // this.tabControl1.Controls.Add(this.tabPage1); this.tabControl1.Controls.Add(this.tabPage2); this.tabControl1.Controls.Add(this.tabPage3);
            this.tabControl1.Dock = System.Windows.Forms.DockStyle.Fill; this.tabControl1.Location = new System.Drawing.Point(70, 0); this.tabControl1.Margin = new System.Windows.Forms.Padding(2); this.tabControl1.Name = "tabControl1"; this.tabControl1.SelectedIndex = 0;
            this.tabControl1.Size = new System.Drawing.Size(722, 427); this.tabControl1.TabIndex = 1;
            //
            // tabPage1
            //
            this.tabPage1.BackColor = System.Drawing.Color.DarkOrchid; this.tabPage1.Controls.Add(this.dataGridViewKrit); this.tabPage1.Controls.Add(this.textBoxInfo); this.tabPage1.Controls.Add(this.label3); this.tabPage1.Controls.Add(this.label2);
            this.tabPage1.Font = new System.Drawing.Font("Comic Sans MS", 9F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.tabPage1.Location = new System.Drawing.Point(4, 22); this.tabPage1.Margin = new System.Windows.Forms.Padding(2); this.tabPage1.Name = "tabPage1";
            this.tabPage1.Padding = new System.Windows.Forms.Padding(2); this.tabPage1.Size = new System.Drawing.Size(714, 401); this.tabPage1.TabIndex = 0;
            this.tabPage1.Text = "Входные данные";
            //
            // dataGridViewKrit



            //
            this.dataGridViewKrit.AllowUserToAddRows = false; this.dataGridViewKrit.AllowUserToDeleteRows = false; this.dataGridViewKrit.BackgroundColor = System.Drawing.Color.Cornsilk; this.dataGridViewKrit.GridColor = System.Drawing.Color.DarkGray; this.dataGridViewKrit.Location = new System.Drawing.Point(16, 195); this.dataGridViewKrit.Margin = new System.Windows.Forms.Padding(2); this.dataGridViewKrit.Name = "dataGridViewKrit"; this.dataGridViewKrit.RowTemplate.Height = 24; this.dataGridViewKrit.Size = new System.Drawing.Size(609, 190); this.dataGridViewKrit.TabIndex = 2;
            //
            // textBoxInfo
            //
            this.textBoxInfo.BackColor = System.Drawing.Color.Cornsilk; this.textBoxInfo.Location = new
```

System.Drawing.Point(15, 35); this.textBoxInfo.Margin = new System.Windows.Forms.Padding(2); this.textBoxInfo.Multiline = true;

this.textBoxInfo.Name = "textBoxInfo"; this.textBoxInfo.Size = new System.Drawing.Size(609, 129); this.textBoxInfo.TabIndex = 1;

//

// label3

//

this.label3.AutoSize = true;

this.label3.ForeColor = System.Drawing.Color.Transparent; this.label3.Location = new System.Drawing.Point(14, 178); this.label3.Margin = new System.Windows.Forms.Padding(2, 0, 2, 0); this.label3.Name = "label3";

this.label3.Size = new System.Drawing.Size(115, 16); this.label3.TabIndex = 0;

this.label3.Text = "Список критериев";

//

// label2

//

this.label2.AutoSize = true;

this.label2.ForeColor = System.Drawing.Color.Transparent; this.label2.Location = new System.Drawing.Point(13, 19); this.label2.Margin = new System.Windows.Forms.Padding(2, 0, 2, 0); this.label2.Name = "label2";

this.label2.Size = new System.Drawing.Size(114, 16); this.label2.TabIndex = 0;

this.label2.Text = "Описание задачи";

//

// tabPage2

// this.tabPage2.Controls.Add(this.flowLayoutPanel1);

this.tabPage2.Location = new System.Drawing.Point(4, 22); this.tabPage2.Margin = new System.Windows.Forms.Padding(2); this.tabPage2.Name = "tabPage2";

this.tabPage2.Padding = new System.Windows.Forms.Padding(2); this.tabPage2.Size = new System.Drawing.Size(714, 401); this.tabPage2.TabIndex = 1;

this.tabPage2.Text = "Расчет и результат"; this.tabPage2.UseVisualStyleBackColor = true;

//

// flowLayoutPanel1

//

this.flowLayoutPanel1.AutoScroll = true; this.flowLayoutPanel1.BackColor = System.Drawing.Color.Cornsilk; this.flowLayoutPanel1.Dock = System.Windows.Forms.DockStyle.Fill;

this.flowLayoutPanel1.FlowDirection = System.Windows.Forms.FlowDirection.TopDown; this.flowLayoutPanel1.Font = new System.Drawing.Font("Sitka Text", 8.25F,

```
System.Drawing.FontStyle.Regular,
        System.Drawing.GraphicsUnit.Point, ((byte)(204))); this.flowLayoutPanel1.Location = new System.Drawing.Point(2, 2); this.flowLayoutPanel1.Margin = new System.Windows.Forms.Padding(2); this.flowLayoutPanel1.Name = "flowLayoutPanel1"; this.flowLayoutPanel1.Size = new System.Drawing.Size(710, 397); this.flowLayoutPanel1.TabIndex = 0; this.flowLayoutPanel1.WrapContents = false;
        //
        // tabPage3
        //
        this.tabPage3.BackColor = System.Drawing.Color.DarkOrange; this.tabPage3.Controls.Add(this.button1); this.tabPage3.Controls.Add(this.numericUpDownObj); this.tabPage3.Controls.Add(this.numericUpDownKrit); this.tabPage3.Controls.Add(this.label5); this.tabPage3.Controls.Add(this.label1);



        this.tabPage3.Font = new System.Drawing.Font("Comic Sans MS", 9F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
        this.tabPage3.Location = new System.Drawing.Point(4, 22); this.tabPage3.Margin = new System.Windows.Forms.Padding(2); this.tabPage3.Name = "tabPage3";
        this.tabPage3.Padding = new System.Windows.Forms.Padding(2); this.tabPage3.Size = new System.Drawing.Size(714, 401); this.tabPage3.TabIndex = 2;
        this.tabPage3.Text = "Настройка";
        //
        // button1
        //
        this.button1.Location = new System.Drawing.Point(229, 150); this.button1.Margin = new System.Windows.Forms.Padding(2); this.button1.Name = "button1";
        this.button1.Size = new System.Drawing.Size(113, 29); this.button1.TabIndex = 3;
        this.button1.Text = "Создать"; this.button1.UseVisualStyleBackColor = true;
        this.button1.Click += new System.EventHandler(this.button1_Click);
        //
        // numericUpDownObj
        //
        this.numericUpDownObj.Location = new System.Drawing.Point(200, 76); this.numericUpDownObj.Margin = new System.Windows.Forms.Padding(2); this.numericUpDownObj.Maximum = new decimal(new int[] {
        5,
        0,
        0,
```

```
0});
this.numericUpDownObj.Minimum = new decimal(new int[] { 2,
0,
0,
0});
this.numericUpDownObj.Name = "numericUpDownObj"; this.numericUpDownObj.Size = new System.Drawing.Size(54, 24); this.numericUpDownObj.TabIndex = 2; this.numericUpDownObj.Value = new decimal(new int[] {
3,




0,
0,
0});
//
// numericUpDownKrit
//
this.numericUpDownKrit.Location = new System.Drawing.Point(200, 31); this.numericUpDownKrit.Margin = new System.Windows.Forms.Padding(2); this.numericUpDownKrit.Maximum = new decimal(new int[] {
10,
0,
0,
0});
this.numericUpDownKrit.Minimum = new decimal(new int[] { 3,
0,
0,
0});
this.numericUpDownKrit.Name = "numericUpDownKrit"; this.numericUpDownKrit.Size = new System.Drawing.Size(54, 24); this.numericUpDownKrit.TabIndex = 2; this.numericUpDownKrit.Value = new decimal(new int[] {
8,
0,
0,
0});
//
// label5
//
this.label5.AutoSize = true;
```

```
this.label5.ForeColor = System.Drawing.SystemColors.ControlText; this.label5.Location = new System.Drawing.Point(13, 78); this.label5.Margin = new System.Windows.Forms.Padding(2, 0, 2, 0); this.label5.Name = "label5";
this.label5.Size = new System.Drawing.Size(134, 16); this.label5.TabIndex = 1;
this.label5.Text = "Количество объектов";
//
// label1
//



this.label1.AutoSize = true;
this.label1.Location = new System.Drawing.Point(13, 33); this.label1.Margin = new System.Windows.Forms.Padding(2, 0, 2, 0); this.label1.Name = "label1";
this.label1.Size = new System.Drawing.Size(183, 16); this.label1.TabIndex = 1;
this.label1.Text = "Количество критериев (3-10)";
//
// MainForm
//
this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F); this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font; this.ClientSize = new System.Drawing.Size(792, 427); this.Controls.Add(this.tabControl1); this.Controls.Add(this.toolStrip1);
this.Margin = new System.Windows.Forms.Padding(2); this.Name = "MainForm";
this.Text = "Метод анализа иерархий"; this.toolStrip1.ResumeLayout(false); this.toolStrip1.PerformLayout(); this.tabControl1.ResumeLayout(false); this.tabPage1.ResumeLayout(false); this.tabPage1.PerformLayout();
((System.ComponentModel.ISupportInitialize)(this.dataGridViewKrit)).EndInit();
this.tabPage2.ResumeLayout(false);
this.tabPage3.ResumeLayout(false); this.tabPage3.PerformLayout();
((System.ComponentModel.ISupportInitialize)(this.numericUpDownObj)).EndInit();
((System.ComponentModel.ISupportInitialize)(this.numericUpDownKrit)).EndInit();
this.ResumeLayout(false);
this.PerformLayout();



}

#endregion
```

```
private System.Windows.Forms.ToolStrip toolStrip1;
private System.Windows.Forms.ToolStripButton toolStripButton2; private System.Windows.Forms.ToolStripButton toolStripButton3; private System.Windows.Forms.ToolStripButton toolStripButton4;

private System.Windows.Forms.TabControl tabControl1; private System.Windows.Forms.TabPage tabPage1; private System.Windows.Forms.TabPage tabPage2; private System.Windows.Forms.TabPage tabPage3; private System.Windows.Forms.Button button1;
private System.Windows.Forms.NumericUpDown numericUpDownKrit; private System.Windows.Forms.Label label1;
private System.Windows.Forms.DataGridView dataGridViewKrit; private System.Windows.Forms.TextBox textBoxInfo;
private System.Windows.Forms.Label label3; private System.Windows.Forms.Label label2;
private System.Windows.Forms.NumericUpDown numericUpDownObj; private System.Windows.Forms.Label label5;
private System.Windows.Forms.FlowLayoutPanel flowLayoutPanel1;
}
}
```

Program.cs

```
using System;
using System.Collections.Generic; using System.Linq;
using System.Windows.Forms;

namespace МАИ
{
static class Program
{
/// <summary>
/// Главная точка входа для приложения.
/// </summary> [STAThread]
static void Main()
{
Application.EnableVisualStyles(); Application.SetCompatibleTextRenderingDefault(false); Application.Run(new MainForm());
}
}
}
```