

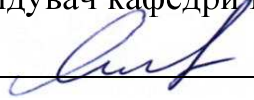
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра інформаційних управляючих систем та технологій

"Допущений до захисту"

Завідувач кафедри ІУСТ



к.т.н, доц. Михелєв І.Л.

" ____ " _____ 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти "магістр"

на тему: «Дослідження методів оптимізації маршрутів у географічних системах
диспетчерської швидкої допомоги»

Виконав: студент групи 6141 м



Пана В.Д.

Керівник роботи:

доцент кафедри ІУСТ, к.т.н.,

(посада, науковий ступінь вчене звання)



Ворона М.В.

м. Миколаїв – 2024 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами

Кафедра інформаційних управляючих систем та технологій

Спеціальність 122 "Комп'ютерні науки"

Освітня програма "Комп'ютерні науки"

"ЗАТВЕРДЖУЮ"

Гарант освітньої програми

В. Ажищев

к.т.н, доц. Ажищев В.Ф.

"___" _____ 2024 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ на здобуття ступеня вищої освіти "магістр"

Студенту Папі Валерію Дмитровичу
(прізвище, ім'я, по батькові)

1. Тема роботи: Дослідження методів оптимізації маршрутів у географічних системах диспетчерської швидкої допомоги

Керівник роботи PhD, доцент Ворона Михайло Владиславович

Затверджені наказом ректора № 1055-уч. від "08" жовтня 2024 року.

2. Термін подання роботи: 15 грудня 2024 р.

3. Вихідні дані до проекту (роботи) ДСТУ щодо обробки інформації, літературні джерела, технічна документація на існуючі аналоги систем, матеріали наукового стажування

4. Перелік питань, що належать до розробки (найменування розділів)

Дослідження методів сортування, проектування ІС, розробка ІС, техніко-економічне обґрунтування розробки, охорона праці, навколишнього охорона середовища, додатки

5. Перелік презентаційних матеріалів постановка задачі, недоліки системи, концептуальна модель, діаграма use case, логічна модель БД, діаграма розгортання, екранні форми, висновки

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	PhD, доцент Ворона М.В.	05.09.2023	22.12.2023
2	PhD, доцент Ворона М.В.	23.12.2023	15.06.2024
3	PhD, доцент Ворона М.В.	15.06.2024	25.09.2024
4	PhD, доцент Ворона М.В.	25.09.2024	14.11.2024
5	PhD, доцент Ворона М.В.	25.09.2024	14.11.2024
6	PhD, доцент Ворона М.В.	25.09.2024	14.11.2024

7. Дата видачі завдання 05 вересня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

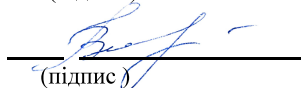
№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів роботи	Примітка
1	Дослідження та аналіз предметної галузі	22.12.2024	
2	Розробка концепції	15.03.2024	
3	Розділ з розробки проектних рішень по системі (та її частинах)	15.06.2024	
4	Реалізація проекту та розробка робочої документації	25.09.2024	
5	Обчислення економічної ефективності	14.11.2024	
6	Вирішення питань охорони праці та навколишнього середовища	14.11.2024	
7	Оформлення пояснювальної записки	14.11.2024	
8	Подання магістерської роботи на перевірку на плагіат	16.11.2024	
9	Подання магістерської роботи рецензенту	20.11.2024	
10	Подання магістерської роботи на державний захист	15.12.2024	

Студент


(підпис)

Папа М.В.
(ПІБ)

Керівник роботи


(підпис)

Ворона М.В.
(ПІБ)

Анотація

Авторами розроблено алгоритм для використання в геоінформаційних системах, який дозволяє зменшити кількість елементів у маршрутах, що виражаються масивами географічних точок. Операція виконується шляхом визначення зайвих елементів і їх відкидання, зберігаючи загальну форму лінії маршруту. Спрощення прискорює подальшу обробку даних і операції відображення.

Пояснювальна записка до дипломної роботи складається з 136 сторінок і включає вступ, п'ять розділів, висновок, список використаних джерел і літератури та чотири додатки. Робота містить 37 рисунків і 23 таблиці.

Ключові слова: ГІС; алгоритм спрощення; спрощення маршрутів.

Abstract

Authors developed an algorithm for use in geographical information systems, which allows to reduce the number of elements in routes that are expressed by arrays of geographical points. The operation is carried out by determining the redundant elements and discarding them, while maintaining the overall shape of the route line. Simplification speeds up further data processing and display operations.

The explanatory note for the thesis consists of 136 pages and includes an introduction, five sections, conclusion, a list of sources and literature and four appendices. The work contains 37 figures and 23 tables.

Keywords: GIS; simplification algorithm; routes simplification.

ЗМІСТ

Вступ.....	7
1 ДОСЛІДЖЕННЯ МЕТОДІВ СПРОЩЕННЯ МАСИВІВ МАРШРУТНИХ ТОЧОК ТА ПОСТАНОВКА ЗАДАЧІ.....	8
1.1 Географічні дані та особливості їх використання в географічних інформаційних системах.....	8
1.2 Актуальність теми.....	10
1.3 Огляд існуючих системних рішень.....	14
1.4 Постановка задачі.....	18
2 ПРОЕКТНІ РІШЕННЯ ДЛЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ	19
2.1 Ескізний проект інформаційної системи.....	19
2.1.1 Побудова моделі діаграми варіантів використання	19
2.1.2 Сценарії основних варіантів використання.....	28
2.1.3 Специфікація варіантів використання.....	34
2.1.4 Інформаційна модель системи.....	42
2.1.5 Проект інтерфейсу системи	44
2.2 Проект математичного забезпечення	46
2.2.1 Розробка алгоритма спрощення маршрутів.....	46
2.2.2 Розробка алгоритма визначення спільних ділянок маршрутів...	46
2.2.3 Розробка допоміжних алгоритмів.....	50
2.2.4 Особливості реалізація геодезійних алгоритмів.....	52
2.3 Проект інформаційного забезпечення системи.....	55
2.3.1 Статична модель програми.....	55
2.3.2 Аналіз вхідних та вихідних даних.....	57
2.4 Проект програмного забезпечення.....	60
2.4.1 Обґрунтування вибору мови програмування та засобів розробки..	60
2.4.2 Розробка діаграми компонентів системи.....	63

2.4.3 Кодування системи.....	64
2.4.4 Тестування системи.....	66
3 РЕЗУЛЬТАТИ РОЗРОБКИ ІНФОРМАЦІЙНОЇ СИСТЕМИ.....	86
3.1 Розробка інструкції користувача.....	86
3.2 Випробування системи	86
4 ТЕХНІКО-ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ РОЗРОБКИ.....	90
5 ОХОРОНА ПРАЦІ.....	104
5.1 Аналіз умов праці в приміщенні.....	104
5.2 Розрахунок освітлення.....	105
5.3 Параметри мікроклімату.....	106
5.4 Шум і вібрація	107
5.5 Випромінювання	108
5.6 Вимоги електробезпеки	109
5.7 Оцінка пожежної безпеки приміщення	110
6 ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА	111
ВИСНОВКИ.....	114
Список використаних джерел	116
Додаток А – Технічне завдання.....	119
Додаток Б – Програма та методика випробувань.....	123
Додаток В – Інструкція користувача.....	125
Додаток Г – Текст програмного забезпечення.....	132

ВСТУП

Сьогодні географічні інформаційні системи використовуються навіть у тих галузях, в яких ще 50 років тому програмні рішення практично не застосовувались. Наприклад, під час планування подорожі або виклику таксі. Найбільш затребуваними такі системи є у сільськогосподарській, транспортній, логістичній, морській, авіаційній, туристичній та інших галузях. Без геодезичних вишукувань не відбувається жоден проект з реконструкції, обладнання чи будівництва нового об'єкта для житла або підприємства. В той самий час майже кожен смартфон має встановленими декілька додатків, що тією чи іншою мірою використовують географічні системи. Простота та інтуїтивність сучасних форм географічних даних та засобів маніпуляцій ними робить такі системи дійсно всюдисущими, адже в наш час навіть неспеціаліст розуміє природу базових форм геоданих (як точки чи маршрути) та користується електронною навігацією[1].

В основі практично будь-якої географічної системи лежить створення, редагування та аналіз географічних даних. При чому, на відміну від багатьох інших сфер, програмні рішення щодо автоматизація таких операцій є за своєю природою універсальними. Наприклад, розробка застосунків для двох абсолютно різних галузей буде, однак, потребувати реалізації однотипних алгоритмів щодо визначення відстані на місцевості, скорочення або об'єднання маршрутів, вводу та виводу геоданих тощо. Результати дослідження таких алгоритмів будуть актуальними та корисними для широкого кола проектів та застосувань[2].

Саме тому, на думку авторів роботи, існує необхідність дослідження, розробки та реалізації ряду алгоритмів, а також їх подальшого об'єднання в комплексну інформаційну систему, що пропонувала б користувачам та розробникам набір функцій для обробки географічних даних, які є необхідними незалежно від сфери застосування тієї чи іншої географічної інформаційної системи.

РОЗДІЛ 1. ДОСЛІДЖЕННЯ МЕТОДІВ СПРОЩЕННЯ МАСИВІВ МАРШРУТНИХ ТОЧОК ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Географічні дані та особливості їх використання в географічних інформаційних системах

Геопросторові дані (або просто географічні дані) - це дані про предмети, події чи явища, які мають місце на поверхні землі. Місце може бути статичним у короткостроковому періоді (наприклад, на дорозі, в морі, в зоні стихійного лиха), або динамічним (наприклад, транспортний засіб або пішохід, що рухається, поширення інфекційного захворювання). Геопросторові дані поєднують інформацію про місцезнаходження (зазвичай координати на земній поверхні), інформацію про атрибути (характеристики відповідного об'єкта, події чи явища), а також часто тимчасову інформацію (часовий або життєвий період, в якому існують місцезнаходження та його атрибути).[3]

Значна частина геопросторових даних представляють загальний інтерес для широкого кола користувачів. Наприклад, дороги, місцевості, водойми та громадські зручності корисні як довідкова інформація для багатьох цілей. З цієї причини, незалежно від того, збираються вони державними чи приватними організаціями, велика кількість геопросторових даних доступна в якості відкритих. Це означає, що користувачі можуть вільно отримувати доступ до них через відкриті стандарти. Розробка та використання відкритих стандартів у геопросторовій спільноті отримали значну підтримку через широкий спектр застосувань, а також через велику кількість агентів як у світі, так і на місцях, які беруть участь у зборі таких даних.

Основною сферою застосування такого типу даних є різноманітні географічні інформаційні системи. ГІС - це система, призначена для збору, зберігання, маніпулювання, аналізу, управління та подання просторових чи географічних даних. ГІС-додатки - це інструменти, які дозволяють користувачам створювати інтерактивні запити (створені користувачем

пошуку), аналізувати просторову інформацію, редагувати дані на картах та представляти результати всіх цих операцій.[4]

Такі системи є потужним інструментом прийняття рішень для будь-якого бізнесу чи галузі, оскільки вони дозволяють аналізувати екологічні, демографічні та топографічні дані. Вони необхідні для виконання багатьох операцій і мають безліч застосувань, пов'язаних з машинобудуванням, плануванням, управлінням, транспортом та логістикою, страхуванням, телекомунікаціями та бізнесом.

ГІС використовують низку різних технологій, процесів, прийомів та методів. Але в основі більшості з них лежить принцип вираження просторової інформації за допомогою точок на поверхні землі, які визначаються за допомогою системи координат (довгота, широта, рідше також висота над рівнем моря). В подальшому за допомогою масивів таких точок можуть бути виражені лінії та криві - маршрути або треки (запис пройденого шляху), в залежності від сфери використання. А також різноманітні зони на земній поверхні - наприклад, чотири точки можуть визначити чотирикутну ділянку земної поверхні. Такі географічні зони важливі не лише завдяки інформації яку вони несуть, але і мають багато прикладних застосувань - наприклад, для забезпечення коректного відображення електронних карт. Якщо користувач бажає переглянути маршрут на мапі, ГІС має визначити, завантаження мапи якого району достатньо для відображення цього маршруту.

1.2 Актуальність теми

Одним із декількох основних типів даних, які обробляються в таких системах є маршрути (або «треки»), які в абсолютній більшості випадків виражаються за допомогою масивів маршрутних точок. Кожна точка має своє місцеположення на земній поверхні (або в іншій системі координат). Лінія,

проведена крізь всі точки (з першої до останньої), становить лінію маршруту.

З огляду на особливості створення таких маршрутів, багато з них мають надмірну кількість елементів. Наприклад, GPS пристрої традиційно записують лінію маршруту з максимальною точністю, тобто зберігаючи всі отримані системою місцеположення.[5] Так, шлях руху автомобіля по прямій трасі може бути записаний з великою кількістю проміжних точок, але ідентична (з практичної точки зору) лінія могла би бути виражена всього двома точками - початку і кінця прямої ділянки.

Типовий маршрут може містити тисячі чи навіть сотні тисяч точок. Великі об'єми зумовлюють високі вимоги до обчислювальних ресурсів для обробки таких даних. Зокрема, відображення великого маршрута в графічний спосіб на мапі часто здійснюється на межі можливостей навіть сучасних комп'ютерних систем.[6] Саме тому є важливим надання розробникам ГІС засобів для зменшення детальності маршрутів, до задовільного для наявних завдань рівня.

Наразі, для виконання такого типу завдань зазвичай використовується алгоритм Рамера-Дугласа-Пекера (також відомого як алгоритм розбиття і злиття), який дозволяє зменшити число точок кривої, апроксимованої більшою серією точок.[7] Його суттєвим недоліком є те, що коректність та ефективність роботи гарантується лише за умови неперехрещення кривою самою себе.[8] Це зумовлює суттєві труднощі при його використанні в топографічних системах.

Зазначене вище зумовлює необхідність винайдення алгоритму, придатного для використання в географічних інформаційних системах, який би дозволяв зменшувати об'єми масивів шляхом видалення надлишкових точок з маршрутів. Необхідною умовою є можливість задання користувачем точності спрощення.

Крім того, не дивлячись на різноманіття сфер застосування, традиційно-базовий набір функцій для створення, редагування та аналізу

географічних даних є доволі схожим для широкого спектру застосування. Це робить логічним створення бібліотек підпрограм, які стануть у пригоді розробникам будь-якого програмного забезпечення, яке оперує геопросторовою інформацією. Такі підпрограми повинні бути оптимізовані для виконання своїх вузьких функцій, але і водночас містити достатньо загальні алгоритми та рішення, які б не обмежували створення нових способів використання географічних даних. Використовуючи функціонал такої бібліотеки, розробники могли б зосередитись на унікальних рисах їхніх продуктів замість наново реалізувати вже існуючі алгоритми. Такі бібліотеки, в поєднанні з уніфікованими стандартами для геоданих, дозволять розробникам програмного забезпечення ефективно створювати та адаптовувати свої рішення під унікальні потреби тієї сфери, для якої вони розробляються.

Крім того, актуальність розробки напряму залежить від правильного вибору «цільової аудиторії» продукту - кінцевого користувача. Сьогодні межа між професіональними розробниками, які створюють новий комерційний продукт, і користувачами, які ними користуються, поступово стирається. Адже багато хто опановує основи написання коду для полегшення власних задач. Так на одному кінці спектра - програміст, який розробляє геосистему для служби таксі. Користуючись програмним інтерфейсом сторонньої бібліотеки він може інтегрувати її в свій продукт. На іншому кінці - мандрівник, який конвертує маршрути, щоб вони «відкрилися» в його навігаторі. Він робить за допомогою програми з графічним інтерфейсом. Але між ними є також працівник архіву (або ентузіаст), якому потрібно автоматизувати процес обробити великих масивів географічних даних. Такий користувач не проти опанувати менш інтуїтивний, але практичний інтерфейс командного рядку або навіть може мати базові навички програмування, які дозволять йому створити просту програмну оболонку або скрипт для модулів бібліотеки, яка найбільше відповідає його потребам.

Тому можна стверджувати, що такий набір функцій стане в пригоді не тільки розробникам програмного забезпечення. Доречною є розробка користувацького інтерфейсу, які би перетворили бібліотеку на повноцінну виконувану програму. Це зробить її функціонал доступним багатьом категоріям користувачів. Однак, важливо розуміти, що не останню роль у створенні такого розмаїття потенційних користувачів бібліотеки буде відігравати інтуїтивний інтерфейс, ясна структура, детальна документація до продукту, а також дотримання усталених в сфері ГІС конвенцій та стандартів. Це все допоможе зорієнтуватись користувачу з будь-яким рівнем досвіду.

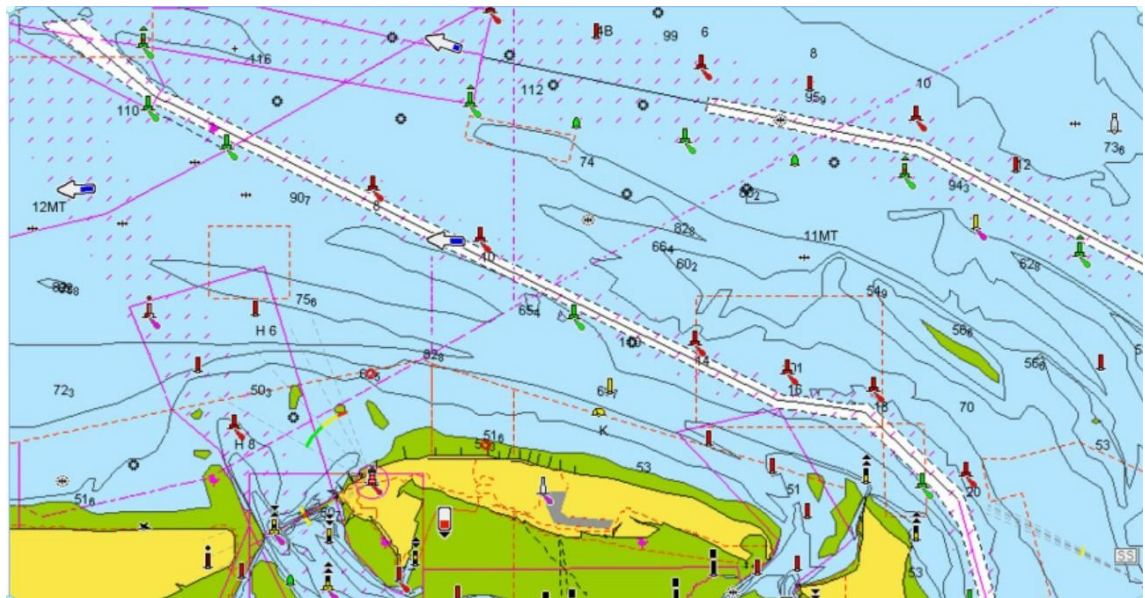


Рисунок 1.1 – Електронна навігаційна карта

Зважаючи на написане вище, постає необхідність розробки програмного забезпечення, яке дозволило би різноманітним користувачам автоматизувати процес спрощення масивів маршрутних точок, а також створення, зміни та аналізу географічних даних.

1.3 Огляд існуючих системних рішень

Через повсюдне використання географічних інформаційних систем не дивно, що на ринку присутня велика кількість програмних продуктів для автоматизації операцій з географічними даними. В цьому підрозділі ми розглянемо декілька найбільш популярних та характерних рішень.

а) GPS Babel - вільно-розповсюджуване прикладне ПЗ, що дозволяє перетворювати маршрутні точки, треки та маршрути між форматами популярними GPS-приймачами, такими як Garmin чи Magellan, та програми для картографування, такі як Google Earth чи Basecamp. Він також має потужні інструменти для маніпулювання такими даними. Наприклад фільтрування дублікатів точок або спрощення доріжок. Програма має графічну оболонку, тому є придатною для повсякденного використання.

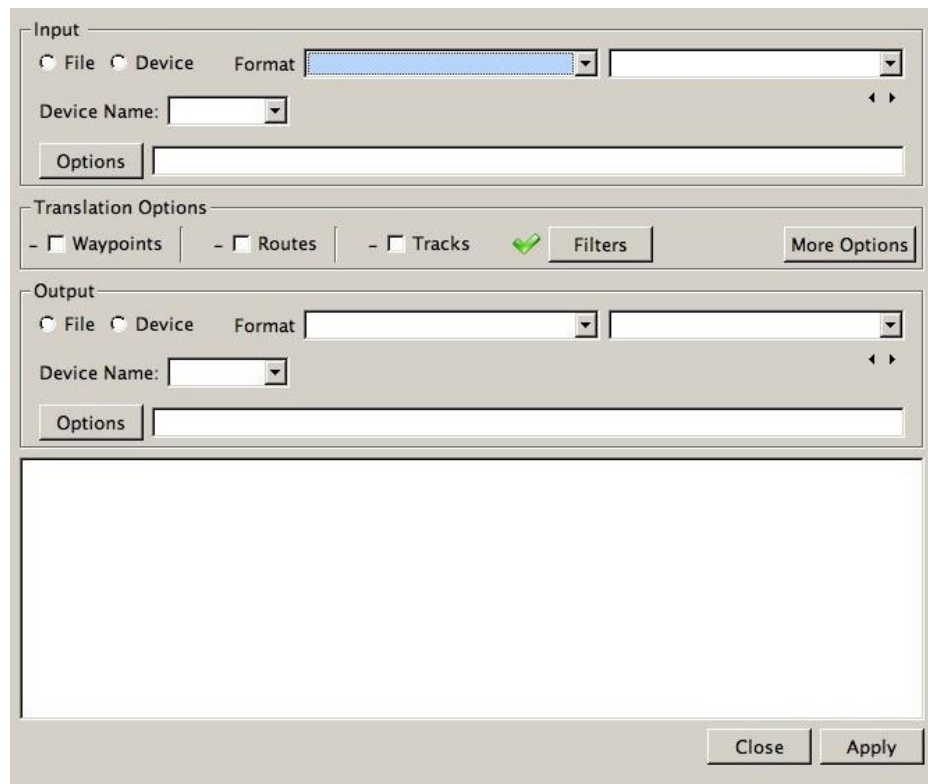


Рисунок 1.2 – Графічний інтерфейс програми GPS Babel

Однак цей продукт має очевидні обмеження:

- Вузька спеціалізація - GPS Vabel задумувався як конвертер між географічними форматами, тому більшість його, досить широкого, функціоналу розрахована саме на таке використання. Так програма не дозволяє створювати чи змінювати дані безпосередньо.

- Відсутність програмного інтерфейсу. Програма розповсюджується у вигляді виконуваного файлу для PC або Mac. Це суттєво обмежує можливість її використання у взаємодії з іншим ПЗ.

б) Google Планета Земля — це безкоштовна, вільно-завантажувана програма компанії Google, що відображає віртуальний глобус. В рамках даного проекту в мережу Інтернет було викладено аерофотознімки та сателітні знімки більшої частини Землі.

До можливостей цієї програми відноситься додання та редагування маршрутних точок і їх базовий аналіз. Так на рис. 1.3 показаний перегляд профілю висот маршруту.

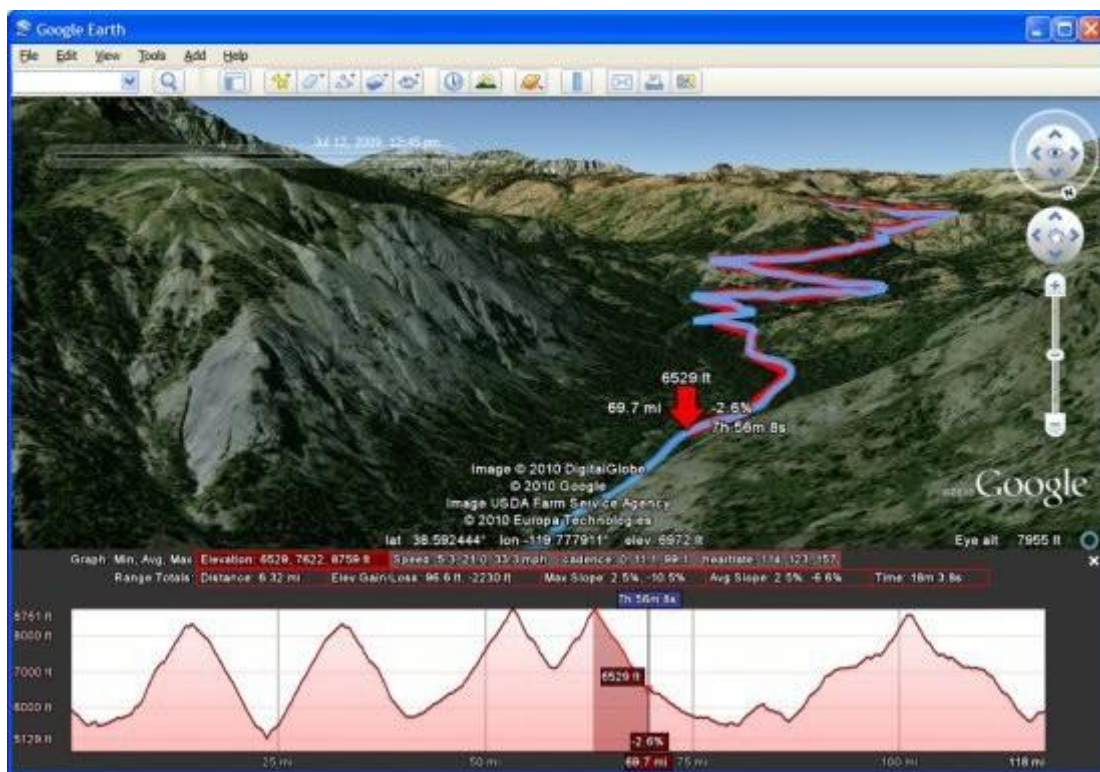


Рисунок 1.3 – Інтерфейс програми Google Earth

Програма є потужним інструментом, але має лише графічний користувацький інтерфейс, що практично виключає використання її можливостей в інших програмних продуктах. Однак популярність програми призвела до того, що багато її рішень стали офіційними та неофіційними стандартами в сфері. Наприклад формати KML та KMX, формат запису додаткової інформації про маршрутні точки, спосіб відображення інформації на картах та супутникових знімках, тощо.

в) Бібліотека Turf.js – являє собою бібліотеку JavaScript для просторового аналізу. Вона включає великий набір функції для аналізу, допоміжні функції для створення даних типу GeoJSON, а також інструменти класифікації та статистики. Turf можна додати на веб-сайт як плагін на стороні клієнта, або запустити на стороні сервера за допомогою Node.js.

Суттєвим недоліком цього ПЗ є обмеженість доступних форматів. Turf працює з GeoJSON, який є зручним та універсальним форматом, але має досить обмежений набір можливостей для професійного використання. Наприклад, на відміну від KML чи GPX географічні точки не можуть мати визначену висоту над рівнем моря.[9]

На рис. 1.4. показаний приклад використання цього ПЗ - визначений за допомогою Turf.js багатокутник відображений на мапі mapbox.



Рисунок 1.4 – Географічна зона у формі багатокутника.

Загалом, наявні рішення можна розділити на дві категорії: програми з графічним інтерфейсом та обмеженим функціоналом, призначені для повсякденного використання; «важковагові» бібліотеки, найкраще придатні для ґрунтового аналізу геоданих. Наразі в даній галузі відсутні інтуїтивні та універсальні програмні продукти, які могли би одночасно об'легшити розробку програмістам та автоматизувати інформаційні процеси користувачам. В таблиці 1.1 узагальнені спільні та відмінні риси оглянутих рішень у порівнянні з системою що має бути розроблена в рамках даної роботи.

Таблиця 1.1 – Порівняльна таблиця наявних програмних рішень з розроблюваною системою

Характеристика	GPS Babel	Google Earth	Turf.js	Розроблювана система
Наявність програмного інтерфейсу	Частково. ПЗ може використовуватись в командах файлів-сценаріїв (.bat)	Відсутня, лише графічний інтерфейс	Наявний, система являє собою бібліотеку підпрограм.	Наявний, реалізована у вигляді підсистеми
Підтримка основних форматів геоданих	Підтримка практично всіх використовуваних стандартів	Наявна підтримка основних форматів	Відсутня, бібліотека працює лише з GeoJSON	Наявна, підтримка GPX, KML, GeoJSON
Можливості щодо редагування даних	Вкрай обмежені. ПЗ є в першу чергу конвертером файлів	Широкі, за умови активного використання графічного інтерфейсу	Широкі, бібліотека містить великий набір функцій	Широкі, набір функцій щодо редагування та аналізу даних
Можливість спрощення маршрутів з заданою точністю	Функція наявна. Ступінь точності задається за школою від 0 до 10	Функція відсутня	Функція відсутня	Функція наявна. Точність задається в метрах

1.4 Постановка задачі

Проаналізувавши предметну область, можна зробити висновок, що постає необхідність спроектувати та реалізувати інформаційну систему, яка дозволить користувачам та розробникам здійснювати найчастіше затребувані операції щодо маніпуляції географічними даними, включно з функцією спрощення маршрутів і треків.

Об'єктом дослідження виступають технології розробки географічних інформаційних систем. Предметом є методи та технології розробки алгоритмів для автоматизації обробки географічних даних.

Метою дослідження є підвищення ефективності обробки масивів маршрутних точок географічними системами.

Для досягнення мети авторам необхідно вирішити наступні задачі:

- дослідити предметну галузь географічних інформаційних систем та наявні рішення;
- сформулювати вимоги до розроблюваної системи;
- розробити проект інформаційної системи;
- розробити необхідні алгоритми для їх подальшої реалізації;
- обрати засоби розробки та здійснити кодування ІС;
- провести випробування системи;
- розробити супровідну документацію до системи.

Для цього необхідно дослідити технології розробки інформаційних систем, які дозволили би реалізувати такі функції як:

- конвертація файлів, що містять географічні дані між найпоширенішими форматами,
- створення та редагування файлів що містять маршрути, точки та зони,
- аналіз маршрутів на набір висоти, довжини, тощо
- об'єднання та скорочення маршрутів,
- визначення зон, необхідних для відображення мапи що містила би маршрут чи точки,

- здійснення утилітарних функцій щодо визначення відстані, курсу та середини між точками, тощо.

Окрім дослідження технологій розробки поширених функцій в ПЗ для маніпуляції географічними даними, також необхідно дослідити який функціонал є затребуваним в розробників, але часто відсутній в таких продуктах. Так, на думку авторів, недостатньо уваги приділялось розробці рішень, що дозволили б автоматизувати операції з географічними зонами, і зокрема їх зв'язок з іншими видами даних, такі як точки і маршрути. Наприклад, практично у всіх системах, які забезпечують графічне відображення геоінформації на мапі, розробникам доводиться імплементувати алгоритми визначення необхідного масштабу та меж карти для коректного відображення користувацьких даних. Узагальнені варіанти таких алгоритмів можуть бути імплементовані одноразово в бібліотеці підпрограм.

Під час вибору засобів розробки також слід звернути особливу увагу на те, що ІС має бути доступною розробникам, які працюють на багатьох різноманітних платформах. Адже сучасні ГІС створюються для роботи на будь-яких пристроях - від мобільних телефонів до промислових машин. Саме тому багатоплатформність є неодмінною умовою актуальності розробки такої системи.

Таким чином, програмне забезпечення має бути реалізоване у вигляді підсистеми з додатковим інтерфейсом командного рядка. Вибір мови програмування та інших засобів розробки має забезпечити використання системи на багатьох платформах та середовищах.

Технічне завдання розроблюваної системи наведено у Додатку А.

РОДІЛ 2. ПРОЕКТНІ РІШЕННЯ ДЛЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Ескізний проект ІС

Ескізний проект – проектна конструкторська документація, яка містить принципові конструктивні рішення і дає загальне уявлення про будову та принцип дії виробу, а також дані, що визначають його відповідність призначенню.

Ескізний проект розробляють з метою встановлення принципових (конструктивних, схемних та ін.) рішень виробу, що дають загальне уявлення про принцип роботи і (або) будову виробу, коли це доцільно зробити до розробки технічного проекту чи робочої документації. На стадії розробки ескізного проекту розглядають варіанти виробу і (або) його складових частин.[10]

Ескізний проект “ІС для редагування географічних даних” включає в себе наступні складові:

- 1) Діаграма варіантів використання
- 2) Специфікація варіантів використання
- 3) Сценарії основних варіантів використання
- 4) Інформаційну модель

2.1.1 Побудова моделі діаграми варіантів використання

Діаграмою використання зазвичай називають варіант діаграмами поведінки, який використовуються для опису множини дій (варіантів використання), які певна система або системи (предмет) повинні або можуть виконувати у співпраці з одним або декількома зовнішніми користувачами системи (акторами).

Розробка діаграми варіантів використання переслідує наступні цілі:

- 1) Визначити спільні кордони та контекст модельованої предметної області на початкових етапах проектування системи.
- 2) Сформулювати загальні вимоги до функціонального поведінки проектованої системи.
- 3) Розробити вихідну концептуальну модель системи для її подальшої деталізації у формі логічних і фізичних моделей.
- 4) Підготувати вихідну документацію для взаємодії розробників системи з її замовниками і користувачами.

Мета визначення варіанту використання полягає в тому, щоб визначити закінчений аспект або фрагмент поведінки деякої сутності без розкриття внутрішньої структури цієї сутності. В якості такої сутності може виступати вихідна система або будь-який інший елемент моделі, який володіє власною поведінкою, подібно підсистемі або класу в моделі системи. Кожен варіант використання відповідає окремому сервісу, який надає модельовану сутність або систему по запиту користувача (актора), тобто визначає спосіб застосування цієї сутності. Сервіс, що інстанціюється по запиту користувача, представляє собою закінчену послідовність дій. Це означає, що після того, як система закінчить обробку запиту користувача, вона повинна повернутися в початковий стан, в якому готова до виконання наступних запитів.

На рис. 2.1 представлена діаграма варіантів використання для “ІС для редагування географічних даних”.

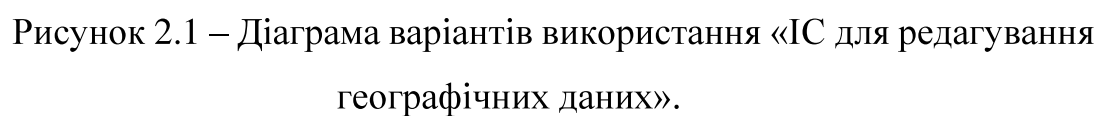


Рисунок 2.1 – Діаграма варіантів використання «ІС для редагування географічних даних».

2.1.3 Сценарії основних варіантів використання

Щодо основних варіантів використання представлені діаграми діяльності (візуальне представлення графу діяльностей).

Тобто якщо варіанти використання ставлять перед системою цілі, то діаграма діяльності показує послідовність дій, необхідних для їх досягнення. Дії це елементарні кроки, які не передбачають подальшу декомпозицію. Діяльність може містити вхідні або вихідні дуги діяльності, що показують потоки управління і потоки даних.

Діаграми активностей будуються з обмеженої кількості фігур, з'єднаних стрілочками. Найважливіші типи фігур:

- 1) скруглені прямокутники позначають дії;
- 2) ромби позначають рішення;
- 3) риски позначають початок (розподіл) чи кінець (об'єднання) паралельних активностей;
- 4) чорний кружок позначає старт (початковий стан) процесу;
- 5) чорний кружок в колі позначає кінець (кінцевий стан).

Стрілки ведуть від старту до кінця і позначають порядок в якому відбуваються активності.

Для даної роботи були створені сценарії для наступних варіантів використання:

- 1) “Визначення відстані від лінії до точки”
- 2) “Визначення найближчої до треку точки”
- 3) “Визначення спільних ділянок двох маршрутів або треків”
- 4) “Спрощення трека”
- 5) “Об'єднання двох треків чи маршрутів”
- 6) “Визначення географічних зон”

Зазначені сценарії приведені в наступному підрозділі, разом із відповідною специфікацією варіанта використання.

2.1.4 Специфікація варіантів використання

У рамках цього підрозділу розроблено специфікації варіантів використання до діаграми варіантів використання, зображеної на рис. 2.1.

Варіанти використання мають наступні специфікації:

1) Редагування геоданих.

Основна дійова особа: Користувач або основна програма.

Інші учасники прецеденту: Відсутні.

Зв'язок з іншими варіантами використання: наявний, включає зміну властивостей точок, зміну треків, редагування геоданих.

Короткий опис: Основний варіант використання, який дозволяє користувачу здійснювати створення та редагування точок, треків, маршрутів як безпосередньо (зміна місцеположення точки), так і за допомогою процесів що потребують застосування алгоритмів (спрошення треку).

Альтернативний потік: Відсутній.

Спеціальні умови: Відсутні.

Передумова: Відсутня.

Постумова: Відсутня.

2) Зміна властивостей точок.

Основна дійова особа: Користувач або основна програма.

Інші учасники прецеденту: Відсутні.

Зв'язок з іншими варіантами використання: наявний, є розширенням варіанта «редагування геоданих».

Короткий опис: Даний варіант дозволяє користувачу безпосередньо змінювати властивості точок, такі як їх місцеположення, висота на рівнем моря, опис, час (для маршрутної точки), тощо.

Альтернативний потік: Відсутній.

Спеціальні умови: Відсутні.

Передумова: Відсутня.

Постумова: Відсутня.

3) Створення геоданих.

Основна дійова особа: Користувач або основна програма.

Інші учасники прецеденту: Відсутні.

Зв'язок з іншими варіантами використання: наявний, включає створення точок та масивів точок.

Короткий опис: Даний варіант використання дозволяє користувачу за створювати нові одиниці географічних даних за допомогою включених підваріантів.

Альтернативний потік: Відсутній.

Спеціальні умови: Відсутні.

Передумова: Відсутня.

Постумова: Відсутня.

4) Створення точок.

Основна дійова особа: Користувач або основна програма.

Інші учасники прецеденту: Відсутні.

Зв'язок з іншими варіантами використання: наявний, є розширенням варіанта «створення геоданих».

Короткий опис: Даний варіант використання дозволяє користувачу створювати нові точки з заданими характеристиками.

Альтернативний потік: Відсутній.

Спеціальні умови: Відсутні.

Передумова: Відсутня.

Постумова: Відсутня.

5) Створення масивів точок.

Основна дійова особа: Користувач або основна програма.

Інші учасники прецеденту: Відсутні.

Зв'язок з іншими варіантами використання: наявний, є розширенням варіанта «Створення геоданих», використовує варіант «Створення точок».

Короткий опис: Даний варіант використання дозволяє користувачу створювати треки та маршрути, які за своєю природою є масивами точок.

Альтернативний потік: Відсутній.

Спеціальні умови: Відсутні.

Передумова: Відсутня.

Постумова: Відсутня.

6) Зміна точок і маршрутів.

Основна дійова особа: Користувач або основна програма.

Інші учасники прецеденту: Відсутні.

Зв'язок з іншими варіантами використання: наявний, є розширенням варіанту «Редагування геоданих», містить варіанти скорочення, спрощення та об'єднання треків.

Короткий опис: Даний варіант використання передбачає операції зі зміни наявних треків та маршрутів.

Альтернативний потік: Відсутній.

Спеціальні умови: Відсутні.

Передумова: Створений або завантажений в систему валідний трек.

Постумова: Відсутня.



Рисунок 2.2 – Сценарій для варіанту використання “Скорочення треку або маршруту”

7) Скорочення трека.

Основна дійова особа: Користувач, або основна програма.

Інші учасники прецеденту: Відсутні.

Зв’язок з іншими варіантами використання: наявний, використовує варіант «Визначення відстані між точками».

Короткий опис: Даний варіант використання дозволяє користувачу скоротити трек до заданої користувачем точки відліку. При чому точка відліку не обов’язково має бути на лінії маршруті. Діаграма діяльності представлена на рис. 2.2.

Базовий потік:

- 1) Визначення найбільш відповідну точці відліку частину треку.
- 2) Визначити більшу частину скорочуваного треку (до і після точки).
- 3) Зберегти частину трека яка є більшою половиною.

Альтернативний потік: Відсутній.

Спеціальні умови: Відсутні.

Передумова: Відсутня.

Постумова: Відсутня.

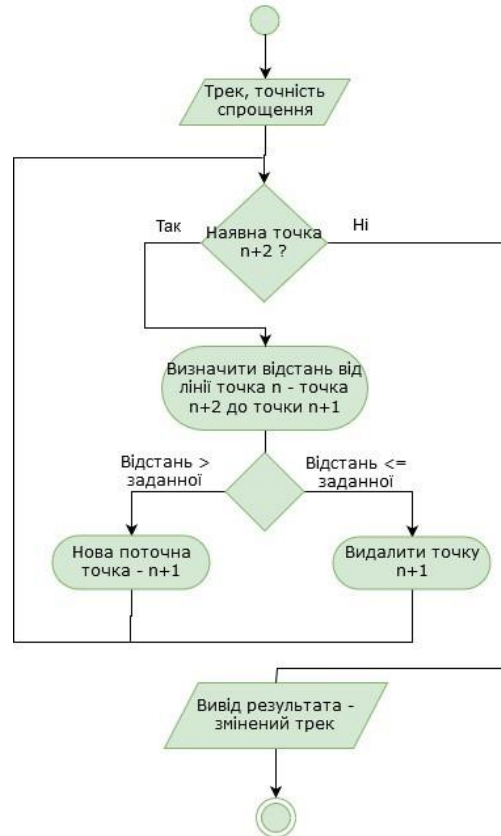


Рисунок 2.3 – Сценарій для варіанту використання “Спрощення треку або маршруту”

8) Спрощення треку або маршруту.

Основна дійова особа: Користувач або основна програма.

Інші учасники прецеденту: Відсутні.

Зв’язок з іншими варіантами використання: наявний, використовує варіант «Визначення відстані між точками» та «Визначення відстані від лінії до точки».

Короткий опис: Даний варіант використання дозволяє користувачу зменшити кількість точок в треку або маршруті, тим самим зменшивши його об’єм та прискоривши всі операції щодо його обробки і відображення.

Алгоритм забезпечує видалення лише надлишкових точок, при якому форма треку суттєво не зміниться. Діаграма діяльності представлена на рис. 2.3.

Базовий потік:

- 1) Визначити відстань кожної точки до лінії між попередньою і наступною точкою.
- 2) Визначити чи змінить суттєво видалення точки форму трека.
- 3) Видалити або лишити точку.

Альтернативний потік: Відсутній.

Спеціальні умови: Відсутні.

Передумова: Відсутня.

Постумова: Відсутня.



Рисунок 2.4 – Сценарій для варіанту використання “Об’єднання треків або маршрутів”

- 9) Об’єднання треків або маршрутів.

Основна дійова особа: Користувач або основна програма.

Інші учасники прецеденту: Відсутні.

Зв'язок з іншими варіантами використання: Наявний, використовує варіант «Знаходження спільних ділянок».

Короткий опис: Даний варіант використання дозволяє користувачу поєднати декілька треків в один. При чому ділянки треків які співпадають будуть збережені лише один раз. Діаграма діяльності представлена на рис. 2.4.

Базовий потік:

- 1) Визначення спільних ділянок.
- 2) Збереження нового треку.

Альтернативний потік: Наявний, передбачає збереження треків в якості окремих сегментів, якщо кінець першого і початок другого треку не збігаються

Спеціальні умови: Відсутні.

Передумова: Відсутня.

Постумова: Відсутня.

- 10) І/О (ввід/вивід) даних.

Основна дійова особа: Користувач, або основна програма.

Інші учасники прецеденту: Відсутні.

Зв'язок з іншими варіантами використання: наявний, включає декілька підваріантів.

Короткий опис: Даний варіант використання включає функції щодо вводу інформації в систему для подальшої обробки та її виводу і збереження, а також її адаптації для збереження даних в форматах, відмінних від формату вводу.

Альтернативний потік: Відсутній.

Спеціальні умови: Відсутні.

Передумова: Відсутня.

Постумова: Відсутня.

11) Імпорт даних.

Основна дійова особа: Користувач, основна програма.

Інші учасники прецеденту: Відсутні.

Зв'язок з іншими варіантами використання: Наявний.

Короткий опис: Даний варіант використання забезпечує ввід наявної географічної інформації в систему, шляхом считування даних з файлів GPX, KML, JSON.

Базовий потік:

- 1) Синтаксичний аналіз файлу.
- 2) Створення об'єктів і масивів з інформацією

Альтернативний потік: Відсутній.

Спеціальні умови: Відсутні.

Передумова: Відсутня.

Постумова: Відсутня.

12) Експорт даних.

Основна дійова особа: Користувач, основна програма.

Інші учасники прецеденту: Відсутні.

Зв'язок з іншими варіантами використання: Наявний.

Короткий опис: Даний варіант використання Забезпечує збереження географічних даних в файлах підтримуваних форматів.

Базовий потік:

- 1) Адаптація інформації для обраного формату.
- 2) Збереження даних в файл.

Альтернативний потік: Відсутній.

Спеціальні умови: Відсутні.

Передумова: Відсутня.

Постумова: Відсутня.

13) Конвертація файлів.

Основна дійова особа: Користувач, основна програма.

Інші учасники прецеденту: Відсутні.

Зв'язок з іншими варіантами використання: Наявний, використовує варіанти імпорту та імпорту даних.

Короткий опис: Даний варіант використання дає можливість змінити формат файлів, що містять географічні дані, шляхом імпорту та експорту даних в іншому форматі (включно з адаптацією інформації під інший стандарт).

Базовий потік:

1) Імпорт даних.

2) Експорт даних в іншому форматі.

Альтернативний потік: Відсутній.

Спеціальні умови: Відсутні.

Передумова: Відсутня.

Постумова: Відсутня.

15) Аналіз геоданих.

Основна дійова особа: Користувач, або основна програма.

Інші учасники прецеденту: Відсутні.

Зв'язок з іншими варіантами використання: наявний, включає в себе аналіз окремих точок, аналіз треків та маршрутів, визначення географічних зон.

Короткий опис: Цей варіант використання включає в себе набір функцій для здійснення операцій аналізу геоданих.

Альтернативний потік: Відсутній.

Спеціальні умови: Відсутні.

Передумова: Створений або завантажений в систему валідний трек, маршрут або точка.

Постумова: Відсутня.

16) Аналіз окремих точок.

Основна дійова особа: Користувач або основна програма.

Інші учасники прецеденту: Відсутні.

Зв'язок з іншими варіантами використання: Наявний, включає в себе декілька окремих випадків аналізу точок.

Короткий опис: Даний варіант використання передбачає таких видів аналіз, які передбачають роботу з декількома окремими точками, а не масивами точок(треків або маршрутів).

Альтернативний потік: Відсутній.

Спеціальні умови: Відсутні.

Передумова: Відсутня.

Постумова: Відсутня.

17) Аналіз маршрутів і треків.

Основна дійова особа: Користувач або основна програма.

Інші учасники прецеденту: Відсутні.

Зв'язок з іншими варіантами використання: Наявний, включає в себе декілька окремих випадків аналізу масивів точок.

Короткий опис: Даний варіант використання передбачає таких видів аналіз, які передбачають роботу з декількома масивами точок - треками або маршрутами.

Альтернативний потік: Відсутній.

Спеціальні умови: Відсутні.

Передумова: Відсутня.

Постумова: Відсутня.

18) Визначення географічних зон.

Основна дійова особа: Користувач або основна програма.

Інші учасники прецеденту: Відсутні.

Зв'язок з іншими варіантами використання: Наявний, включає в себе декілька окремих випадків.

Короткий опис: Даний варіант використання дає можливість користувачу визначити географічну зону, яка би повністю вміщувала задані елементи - точки або маршрути. такий функціонал дозволяє завчасно визначити, наприклад, мапа якої зони має бути завантажена для відображення тих чи інших даних.

Альтернативний потік: Відсутній.

Спеціальні умови: Відсутні.

Передумова: Відсутня.

Постумова: Відсутня.

19) Визначення відстані між точками.

Основна дійова особа: Користувач або основна програма.

Інші учасники прецеденту: Відсутні.

Зв'язок з іншими варіантами використання: наявний, використовується для скорочення та спрощення треків чи в маршрутів.

Короткий опис: Визначення відстані між двома точками по поверхні земної кулі.

Альтернативний потік: Відсутній.

Спеціальні умови: Відсутні.

Передумова: Відсутня.

Постумова: Відсутня.

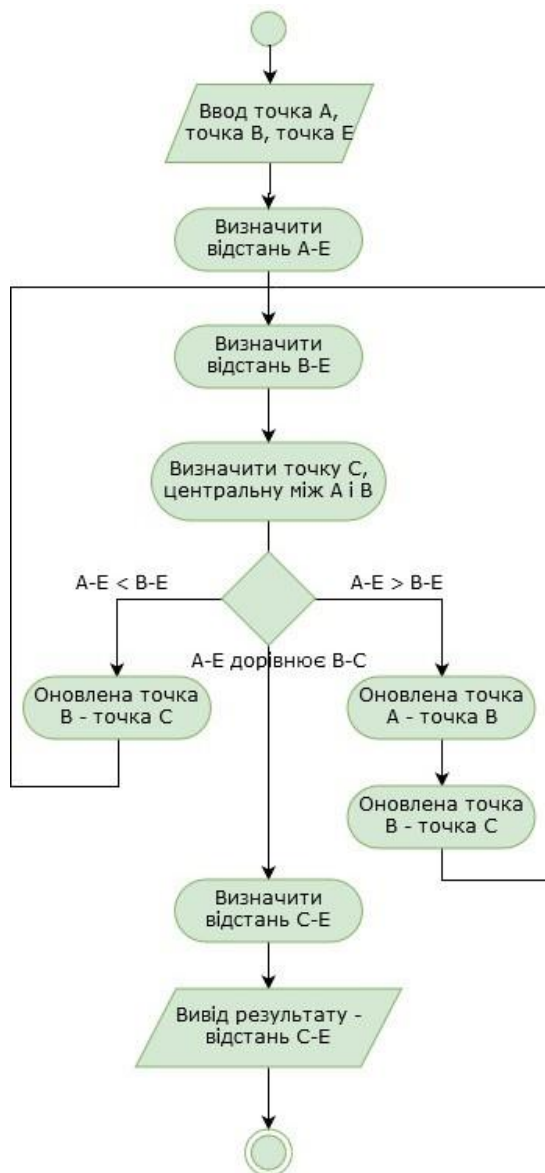


Рисунок 2.5 – Сценарій для варіанту використання “Визначення відстані від лінії до точки”

20) Визначення відстані від лінії до точки.

Основна дійова особа: Користувач або основна програма.

Інші учасники прецеденту: Відсутні.

Зв’язок з іншими варіантами використання: наявний, використовується для спрощення маршрутів.

Короткий опис: Даний варіант використання передбачає визначення найкоротшої відстані від заданої точки до лінії, вираженої двома іншими точками. Діаграма діяльності представлена на рис. 2.5.

Альтернативний потік: Відсутній.

Спеціальні умови: Відсутні.

Передумова: Відсутня.

Постумова: Відсутня.

21) Визначення напрямку між двома точками.

Основна дійова особа: Користувач або основна програма.

Інші учасники прецеденту: Відсутні.

Зв'язок з іншими варіантами використання: наявний.

Короткий опис: Дозволяє користувачу визначити напрям (курс) між двома точками на земній поверхні.

Альтернативний потік: Відсутній.

Спеціальні умови: Відсутні.

Передумова: Відсутня.

Постумова: Відсутня.

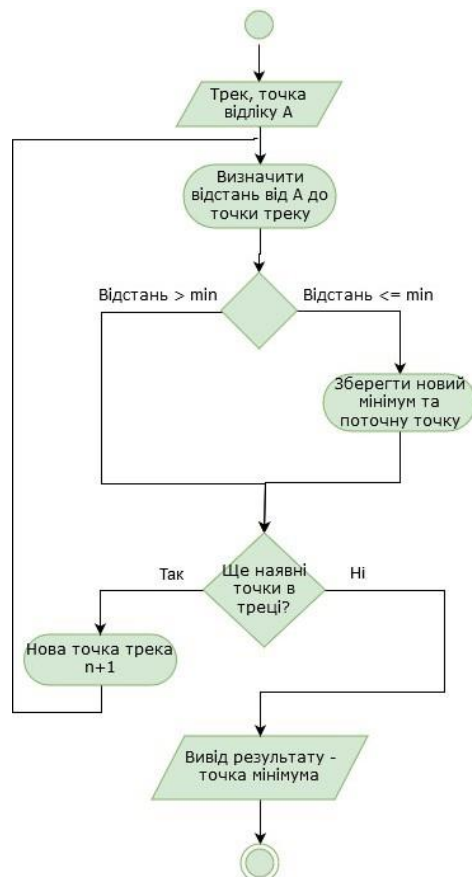


Рисунок 2.6 – Сценарій для варіанту використання “Визначення
найближчої точки треку”

22) Визначення найближчої точки треку.

Основна дійова особа: Користувач або основна програма.

Інші учасники прецеденту: Відсутні.

Зв’язок з іншими варіантами використання: наявний.

Короткий опис: Дозволяє визначити яка з точок на треку є найближчою до даної точки. Діаграма діяльності представлена на рис. 2.6.

Альтернативний потік: Відсутній.

Спеціальні умови: Відсутні.

Передумова: Відсутня.

Постумова: Відсутня.



Рисунок 2.7 – Сценарій для варіанту використання “Визначення спільних ділянок маршрутів чи треків”

23) Визначення спільних ділянок маршрутів чи треків.

Основна дійова особа: Користувач або основна програма.

Інші учасники прецеденту: Відсутні.

Зв’язок з іншими варіантами використання: наявний, використовується варіантом «Об’єднання треків і маршрутів».

Короткий опис: Даний варіант використання дозволяє користувачу отримати інформацію щодо того, які частини двох треків співпадають, тобто їх лінії проходять в межах заданої відстані одна від одної. Діаграма діяльності представлена на рис. 2.7.

Альтернативний потік: Відсутній.

Спеціальні умови: Відсутні.

Передумова: Відсутня.

Постумова: Відсутня.

24) Визначення довжини маршрутів чи треків.

Основна дійова особа: Користувач або основна програма.

Інші учасники прецеденту: Відсутні.

Зв'язок з іншими варіантами використання: наявний.

Короткий опис: Даний варіант використання дозволяє користувачу визначити загальну довжину маршруту на місцевості.

Базовий потік:

1) Визначити відстань між всіма точками маршрута.

2) Додати результати.

Альтернативний потік: Відсутній.

Спеціальні умови: Відсутні.

Передумова: Відсутня.

Постумова: Відсутня.

25) Визначення перепаду висоти на маршруті.

Основна дійова особа: Користувач або основна програма.

Інші учасники прецеденту: Відсутні.

Зв'язок з іншими варіантами використання: наявний.

Короткий опис: Даний варіант використання забезпечує можливість знаходження набору або втрати висоти в результаті руху по маршруту.

Базовий потік:

- 1) Визначити різницю висоти між всіма точками поспіль.
- 2) Визначити загальний перепад висот.

Альтернативний потік: Відсутній.

Спеціальні умови: Відсутні.

Передумова: Відсутня.

Постумова: Відсутня.



Рисунок 2.8 – Сценарій для варіанту використання “Визначення географічних зон”

26) Визначення чотирикутника що вміщає трек чи маршрут.

Основна дійова особа: Користувач або основна програма.

Інші учасники прецеденту: Відсутні.

Зв'язок з іншими варіантами використання: наявний.

Короткий опис: Даний варіант використання дозволяє визначити прямокутник, який вміщав би всі точки даного маршруту. Зона визначається двома кутовими точками. Спільна з варіантом номер 27 діаграма діяльності представлена на рис. 2.8.

Базовий потік:

- 1) Визначення крайніх значень координат (схід, захід, південь, північ)
- 2) Створення кутової точки зі східною довготою і північною широтою.
- 3) Створення кутової точки із західною довготою і північною широтою.

Альтернативний потік: Відсутній.

Спеціальні умови: Відсутні.

Передумова: Відсутня.

Постумова: Відсутня.

27) Визначення чотирикутника що вміщає декілька географічних зон.

Основна дійова особа: Користувач або основна програма.

Інші учасники прецеденту: Відсутні.

Зв'язок з іншими варіантами використання: наявний.

Короткий опис: Даний варіант використання дозволяє визначити прямокутник, який вміщав би декілька інших аналогічних прямокутних зон. Спільна з варіантом номер 26 діаграма діяльності представлена на рис. 2.8.

Базовий потік:

- 1) Визначення крайніх значень координат (схід, захід, південь, північ) для всіх точок що визначають дані прямокутники.

2) Створення кутової точки зі східною довготою і північною широтою.

3) Створення кутової точки із західною довготою і північною широтою.

Альтернативний потік: Відсутній.

Спеціальні умови: Відсутні.

Передумова: Відсутня.

Постумова: Відсутня.

2.1.4 Інформаційна модель системи

Інформаційна модель (ІМ) – це модель, яка описує інформаційні процеси або містить опис об’єкта, в якому вказано деякі його типові властивості, важливі для виконання конкретної задачі. Будь-яка ІМ містить лише суттєві відомості про об’єкт з урахуванням мети її створення.

Концептуальна модель – модель предметної області, що складається з переліку взаємопов’язаних понять, що використовуються для опису цієї області, разом з властивостями й характеристиками, класифікацією цих понять, за типами, ситуацій, ознаками в даній області і законів протікання процесів в ній.

На рис. 2.9 представлена інформаційна модель для «ІС для редагування геоданих».

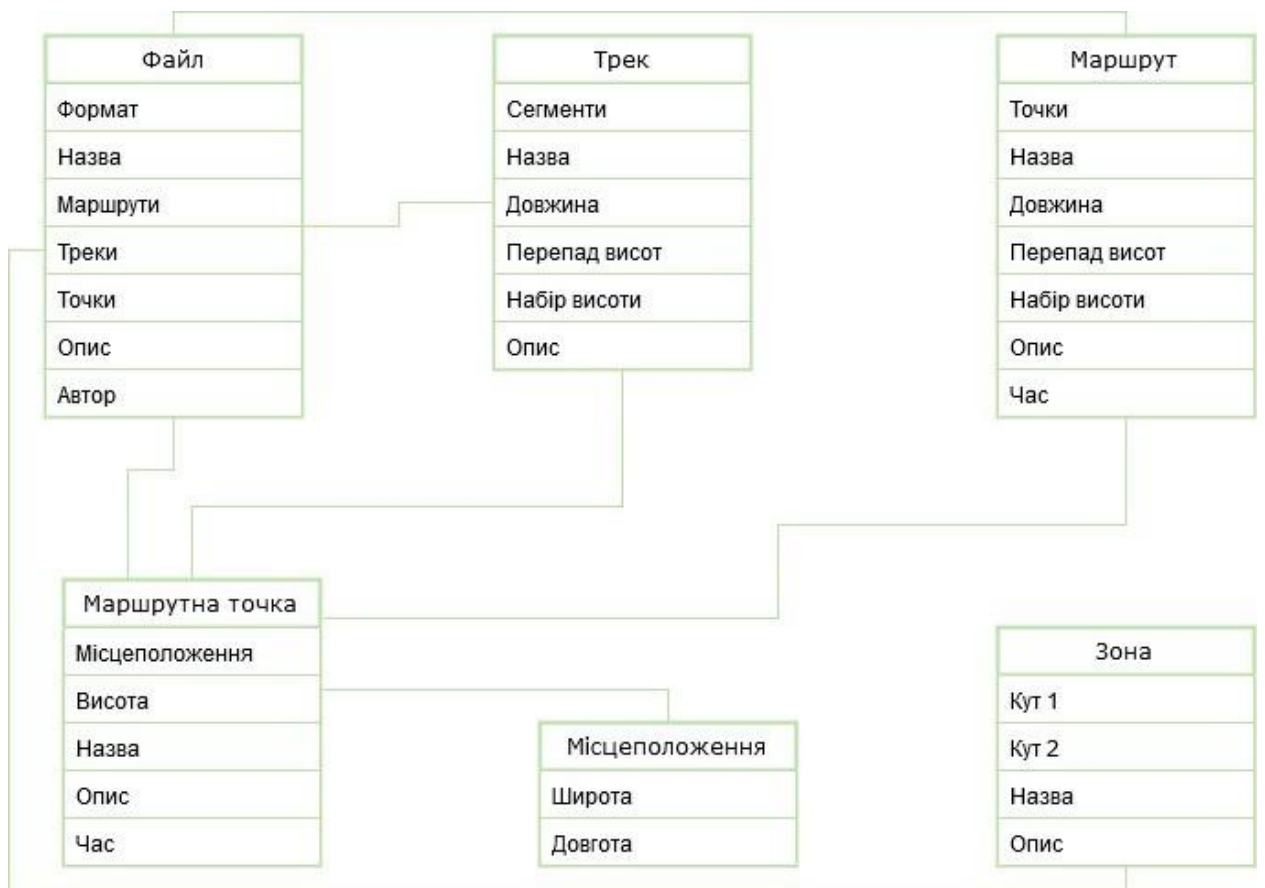


Рисунок 2.9 – Інформаційна модель для «ІС для редагування геоданих».

2.1.5 Проект інтерфейсу системи

Зважаючи, що використання розроблюваної ІС передбачається насамперед в якості підсистеми (бібліотеки підпрограм), першочерговим завданням є розробка прикладного програмного інтерфейсу (API), який би дозволив розробникам використовувати функціонал системи, одночасно абстрагувавшись від подробиць її реалізації. Специфікація такого інтерфейсу в значній мірі залежить від вибору засобів програмування, адже різним середовищам доступні різні засоби реалізації модульності коду. Втім, зважаючи на загальноприйнятні норми для такого роду інтерфейсів, вже на етапі ескізного проектування можна виділити основні вимоги до програмного інтерфейсу системи.[11]

Наприклад, більша частина функціоналу підсистеми має бути доступна через застосування функцій (процедур). Кожна така функція повинна мати чітко визначену та задокументовану сигнатуру. Найпростіший приклад - функція визначення відстані між двома точками на поверхні Землі, яка може мати приймати параметри, що визначають місцеположення двох точок, а повертати значення відстані, скажімо, в метрах:

$$\langle \text{Int} \rangle \text{ distance}(\text{lat_1}, \text{lon_1}, \text{lat_2}, \text{lon_2}) \quad (2.1)$$

Крім того, така операція має бути доступна не лише користувачу (основній системі), а й безсумнівно буде використовуватись іншими модулями підсистеми. Такий дуалізм застосування необхідно враховувати під час розробки.

Другою схемою взаємодії може бути така, що використовує об'єктно-орієнтовну парадигму програмування, включно з набором описів визначень класів, та форм поведінки, пов'язаних з цими класами. Так, наприклад, основній системі може бути доступний реалізований в підсистемі клас Point,

що виражає точку на земній поверхні, включно з її місцезорозташуванням. Такий клас матиме підпис (функцію-конструктор):

$$\langle \text{Point} \rangle \text{Point}(\text{lat_1}, \text{lon_1}) \quad (2.2)$$

В той же час екземпляр такого класу зможе мати методи, в яких реалізована поведінка, пов'язана з даним об'єктом.

Але, крім того, під час постановки завдання було прийняте рішення також розробити користувацький інтерфейс, який дозволив би користувачу використовувати основний функціонал системи безпосередньо.

Інтерфейс користувача – це сукупність програмних і апаратних засобів, що забезпечують взаємодію користувача з комп'ютером. Основу такої взаємодії складають діалоги. Під діалогом в даному випадку розуміють регламентований обмін інформацією між людиною і комп'ютером, який здійснюється в реальному масштабі часу і спрямований на спільне вирішення конкретної задачі. Кожен діалог складається з окремих процесів введення/виводу, які фізично забезпечують зв'язок користувача і комп'ютера. [12]

Визначений технічним завданням інтерфейс командного рядка являє собою різновид текстового інтерфейсу користувача й комп'ютера, в якому інструкції комп'ютеру можна дати тільки введенням із клавіатури текстових рядків (команд). Найзагальніший загальноприйнятий формат команд (у квадратні дужки поміщені необов'язкові частини):

$$\text{ім'я_команди} [\text{параметр_1} [\text{параметр_2} [\dots]]] \quad (2.3)$$

На цьому етапі розробки доречним є виділення двох особливостей реалізації такого інтерфейсу, які є доречними для даної ІС. По-перше, це файл, як базова одиниця географічних даних, з якої здійснюється імпорт

даних, відносно якої проводяться маніпуляції даними (або, скоріше, відносно об'єкту, що уособлює файл в системі), та в який зрештою здійснюється експорт даних. По-друге, це набір параметрів-селекторів, який дозволить користувачу однією командою вибрати одну чи декілька одиниць даних для здійснення операції. Форма таких параметрів може базуватись, наприклад, на типу одиниці даних (точка, маршрут, тощо) та його порядковому номері в файлі або системі. Наприклад, параметри, що вибирають маршрут номер один та точки номер один і три (скажімо для скорочення маршрута від і до цих точок) може виглядати так:

--маршрут 1 --точка 1,3 (2.4)

2.2 Проект математичного забезпечення

Географічні інформаційні системи, тобто системи, призначені для збору, зберігання, маніпулювання, аналізу, управління та подання просторових чи географічних даних, в наш час використовуються практично у всіх сферах життя та господарської діяльності.

Одним із декількох основних типів даних, які обробляються в таких системах є маршрути (або «треки»), які в абсолютній більшості випадків виражаються за допомогою масивів маршрутних точок. Кожна точка має своє місцеположення на земній поверхні (або в іншій системі координат). Лінія, проведена крізь всі точки (з першої до останньої), становить лінію маршруту. В даному підрозділі буде досліджено та розроблено ряд алгоритмі для автоматизованої маніпуляції таким типом даних, а також ряду інших алгоритмів, реалізація яких необхідна для функціонування системи.

2.2.2 Розробка алгоритма спрощення маршрутів

З огляду на особливості створення маршрутів, багато з них мають

надмірну кількість елементів. Наприклад, GPS пристрої традиційно записують лінію маршруту з максимальною точністю, тобто зберігаючи всі отримані системою місцеположення.[5] Так, шлях руху автомобіля по прямій трасі може бути записаний з великою кількістю проміжних точок, але ідентична (з практичної точки зору) лінія могла би бути виражена всього двома точками - початку і кінця прямої ділянки.

Типовий маршрут може містити тисячі чи навіть сотні тисяч точок. Великі об'єми зумовлюють високі вимоги до обчислювальних ресурсів для обробки таких даних. Зокрема, відображення великого маршрута в графічний спосіб на мапі часто здійснюється на межі можливостей навіть сучасних комп'ютерних систем.[6] Саме тому є важливим надання розробникам ГІС засобів для зменшення детальності маршрутів, до задовільного для наявних завдань рівня.

Наразі, для виконання такого типу завдань зазвичай використовується алгоритм Рамера-Дугласа-Пекера (також відомого як алгоритм розбиття і злиття), який дозволяє зменшити число точок кривої, апроксимованої більшою серією точок.[7] Його суттєвим недоліком є те, що коректність та ефективність роботи гарантується лише за умови неперехрещення кривою самою себе.[8] Це зумовлює суттєві труднощі при його використанні в топографічних системах.

Зазначене вище зумовлює необхідність винайдення алгоритму, придатного для використання в географічних інформаційних системах, який би дозволяв зменшувати об'єми масивів шляхом видалення надлишкових точок з маршрутів. Необхідною умовою є можливість задання користувачем точності спрощення.

Основним параметром, який зазначає користувач (або програма що використовує алгоритм) є гранична допустима відстань, на яку спрощена лінія шляху може віддалятися від будь-якої точки початкового маршруту. Це значення буде різнитися в залежності від застосування, тому воно має лишатися на розсуд користувача.

Отримавши це значення і масив точок для спрощення, система має виконати ряд ітерацій, в основі яких лежить оцінка того, наскільки кожна точка лежить в стороні від лінії між попередньою та наступною точками. Порівнявши це значення з заданим граничним, можна дійти висновку про доцільність збереження цієї точки в масиві. В той же час перша і остання точка (як початок і кінець маршруту) завжди лишаються незмінними.

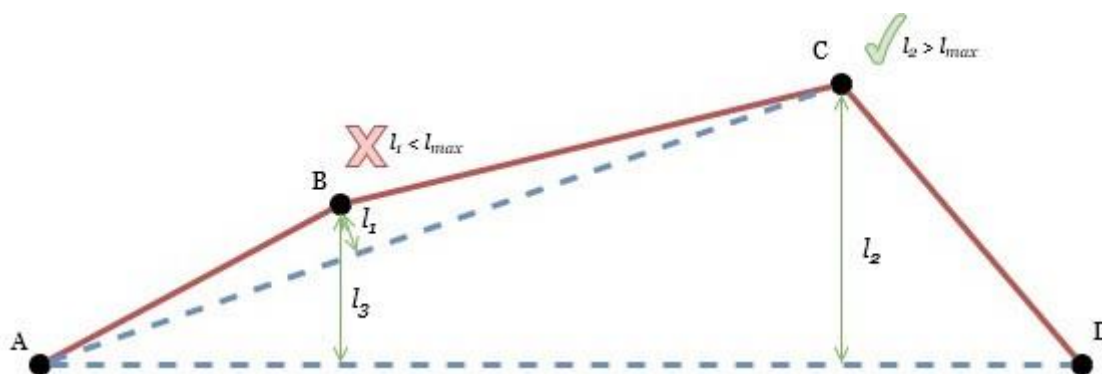


Рисунок 2.10 – Схематичне зображення дії алгоритму

На рис. 2.10 проілюстрована робота алгоритму на прикладі короткого трека з чотирьох точок - ABCD. Першим кроком визначається відстань l_1 між точкою B (другою точкою) до лінії між точками A і C (перша і третя відповідно). Вона виявляється меншою ніж задана відстань l_{max} , тому ця точка може бути відкинута. Наступним кроком визначається відстань від точки C до лінії AD (перша і четверта точки). Ця відстань є більшою ніж l_{max} , отже точка C має істотне значення для формування лінії маршруту, тому вона має бути збережена. Збереження точки означає перехід до наступного кроку, а саме - ідентичних дій починаючи з наступної збереженої точки, в цьому випадку точки C.

Також слід зазначити необхідність збереження всіх видалених точок до кінця кожної ітерації. Так, алгоритм має пересвідчитись, що видалення наступної точки не порушує умову не лише відносно поточної точки, а й тих, що були видалені на цьому відрізку раніше.

Алгоритм може бути виражений за допомогою псевдокоду наступним

чином (як показано на рис. 2.11).

```
func distance(a,b,e);
lmax;
track[];

deleted[];
i = 0;
while(i+3 <= track.length){
    l = distance(track[i],track[i+2],track[i+1])
    if(l<=lmax){
        all = true;
        for(ii = 0; ii<deleted.length && all == true; ii++){
            ll = distance(deleted[ii],track[i+2],track[i+1])
            if(ll > lmax) all = false;
        }
        if(all){
            deleted.add[track[i+1]];
            del track[i+1];
        }else{
            deleted.clear();
            i++;
        }
    }else{
        deleted.clear();
        i++;
    }
}
```

Рисунок 2.11 – Приклад реалізації алгоритма спрощення

В якості функції `distance(a,b,e)` виступає підпрограма підрахунку відстані від точки `e` до лінії `ab`. Така функція буде імплементована в різний спосіб, в залежності від особливостей застосування системи.

На рис 2.12 представлений приклад результату роботи алгоритму. В тій частині шляху, де напрям змінюється в більшій мірі, була збережена значно більша кількість точок на одиницю відстані.

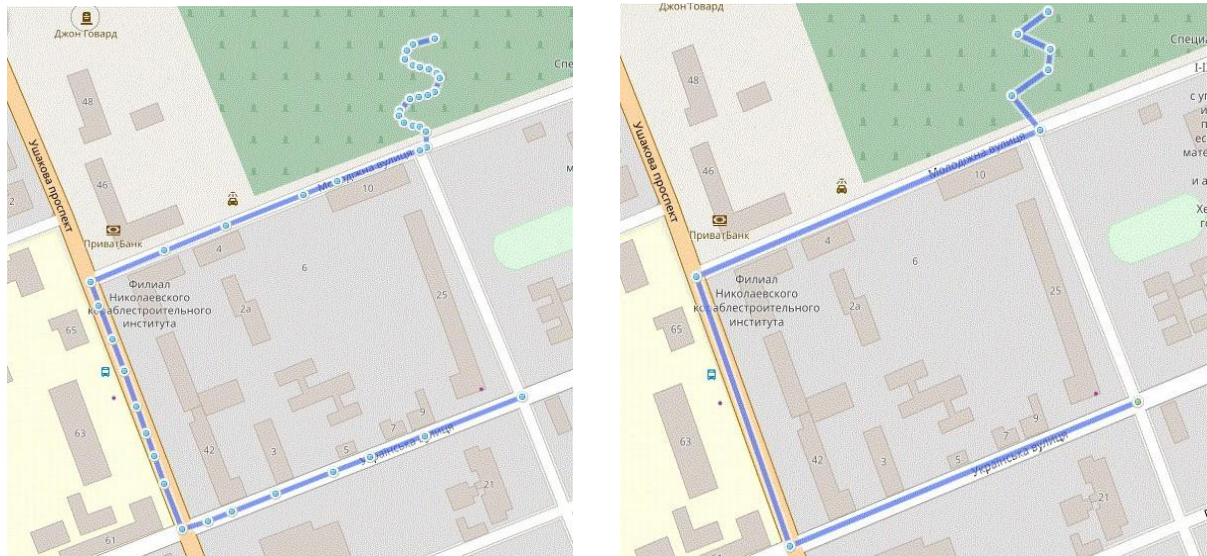


Рисунок 2.12 – Порівняння початкового маршруту (ліворуч) зі спрощеним(праворуч)

Очікувана складність алгоритму може бути визначена як $T(n) = O(n \cdot N)$, де N це кількість видалених точок, які були розташовані в масиві одна за одною. Тому в гіршому випадку (максимальна кількість видалених точок, розташованих одна за одною) складність алгоритму становить $O(n^2)$. Такі показники є подібними до показників інших рішень, наведених вище.[7]

Запропонований алгоритм є придатним для використання в географічних інформаційних система та дозволяє оптимізувати різноманітні операції пов'язані зі збереження, передачею, обробкою та відображенням даних, що містять маршрути, виражені за допомогою масивів точок. Спосіб задання точності спрощення у вигляді максимально-допустимого відхилення (відстані) апроксимації від початкової лінії є зручним та інтуїтивним, що забезпечить його коректне використання для багатьох застосунків.

2.2.3 Розробка алгоритма визначення спільних ділянок маршрутів

Ще одним алгоритмом, розробка якого необхідна для приведення ІС до відповідності технічним вимогам, є алгоритм пошуку спільних ділянок в двох масивах точок (маршрутах або треках). Це необхідно в першу чергу для

реалізації функції об'єднання декількох маршрутів, при чому їх спільні ділянки не мають бути збережені декілька раз. Крім того, отримана інформація може використовуватись користувачем (розробником основної системи) безпосередньо (наприклад, для визначення ступені збігу двох маршрутів).

Наразі не існує відкритого та загальноприйнятого в сфері алгоритма для виконання такого типу завдань.[2] Це зумовлює необхідність винайдення алгоритму, придатного для використання в географічних інформаційних системах, який би дозволяв визначити, які саме відрізки декількох маршрутів проходять достатньо недалеко один від одного щоб вважатися спільними. Необхідною умовою є можливість задання користувачем точності спрощення - значення відстані між лініями маршрутів, менше якої вони вважаються співпадаючими. Це значення буде різнитися в залежності від застосування, тому воно має лишатися на розсуд користувача.

Отримавши це значення і два масиви точок для порівняння, система має виконати ряд ітерацій, в основі яких є визначення відповідності даних відрізків двом умовам: по-перше, чи можливо, що дана ділянка маршрута (яка може являти собою криву, сформовану багатьма точкам) пересікається з аналогічною ділянкою іншого маршрута, або неможливо. Наприклад, якщо відстань між двома будь-якими (довільно обраними) точками двох частин маршрутів є більшою, ніж сума довжин кривих цих ділянок, навіть при «найгіршому» (обидва відрізка лежать на одній прямій) випадку ці відрізки не можуть досягти один-одного. Це дозволить алгоритму одразу відкинути ці відрізки та всі їх складові, адже вони розташовані занадто далеко один від одного щоб мати спільні ділянки. В зворотньому випадку (якщо ділянки в змозі пересіктися на місцевості) алгоритм має розділити одну з ділянок надвоє і рекурсивно порівняти поділені частини з другою ділянкою в аналогічний спосіб. Коли діленням навпіл будуть досягнуті найпростіші відрізки (такі, що сформовані двома точками), їх можна порівняти в другий спосіб, а саме шляхом знаходження мінімальної відстані від кожної точки,

що визначають відрізки, до лінії іншого відрізка, як проілюстровано на рис. 2.13.

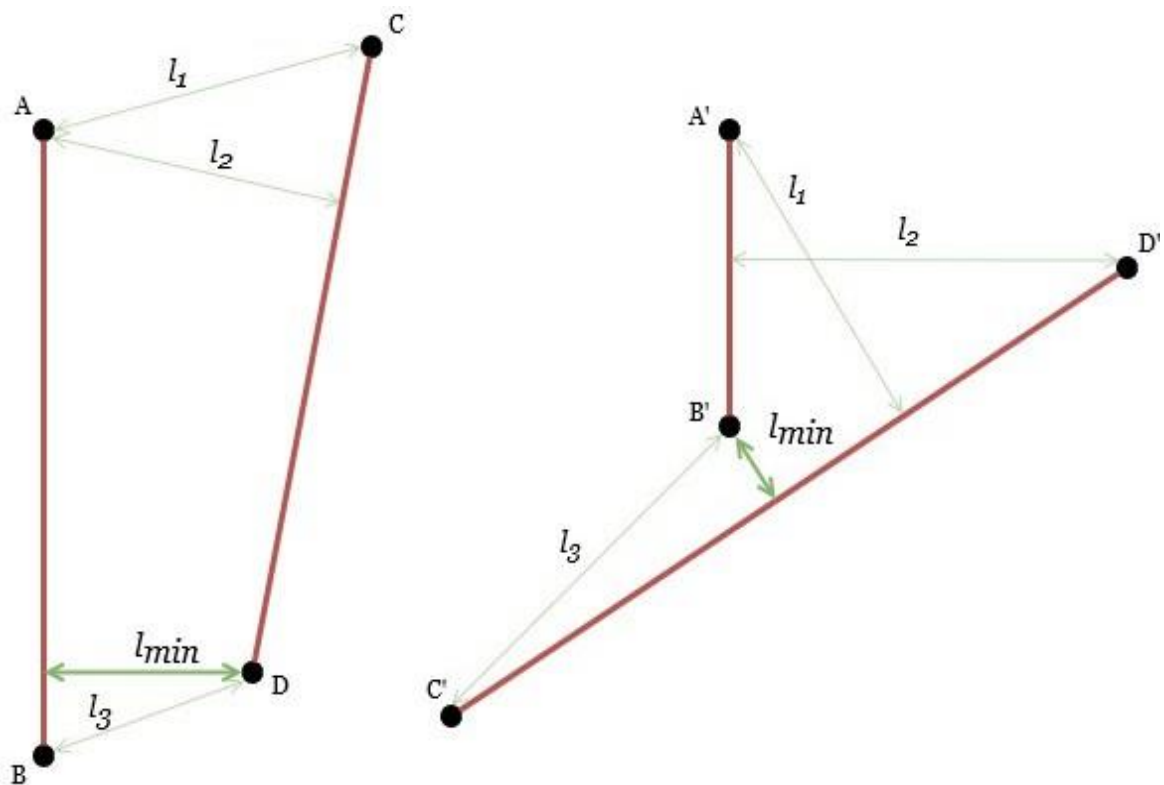


Рисунок 2.13 – Визначення мінімальної відстані між двома відрізками

Відстань l_{min} є найменшою серед чотирьох значень. Якщо вона є меншою ніж задана точність - відрізки вважаються такими, що пересікаються.

Функція має повертати результат у вигляді значень початку і кінця спільних ділянок, що потребує групування послідовних відрізків, що пересікаються.

Очікувана складність алгоритму може бути визначена як $T(n) = O(n+N)$, де N це кількість найпростіших відрізків що потребують порівняння. В найгіршому випадку складність алгоритму становить $O(n^2)$, але вхідні дані, при яких кожна ділянка кожного маршрутів розташовано достатньо близько до всіх ділянок іншого маршруту (і тому потребує порівняння з кожною) не є реалістичними.[13]

2.2.4 Розробка допоміжних алгоритмів

Окрім описаних вище, для реалізації функціоналу системи необхідно розробити ряд алгоритмів, які хоча і необхідними, але їх реалізація є достатньо очевидною. Такими алгоритмами є:

- скорочення треку(маршрута);
- розворот трека(маршрута);
- об'єднання треків(маршрутів);
- визначення довжини маршрута;
- визначення перепаду, набору та втрати висоти на маршруті.

Діаграмні діяльності даних алгоритмів приведені в попередніх підрозділах в якості ескізної документації, а їх реалізація описана в подальших підрозділах.

2.2.5 Особливості реалізація геодезійних алгоритмів

Розробка добредосліджених геодезійних (пов'язаних з визначення форми і розмірів Землі і точних вимірювань на місцевості) алгоритмів не входить в задачі даної роботи, однак перш ніж реалізовувати необхідні низькорівневі алгоритми для ІС, постала необхідність дослідити основні принципи та деякі особливості таких рішень.

Велике коло — це перетин сфери площиною, яка проходить через центр сфери. Усі великі кола сфери мають однакову довжину та слугують аналогом прямих ліній у сферичній геометрії. Іншими словами — це шлях із найменшою кривиною, кривина кола є константою і визначається за формулою:

$$\kappa = \frac{1}{R} \quad (2.5)$$

де R - радіус земної кулі.

І, таким чином, дуга великого кола є найкоротшим шляхом між двома точками на поверхні сфери, або ортодроною.[14] Ця властивість робить концепцію великого кола земної кулі центральною у вирішенні геодезичних задач. Наприклад, визначення найкоротшої відстані між двома точками на земній поверхні (і по земній поверхні), є знаходженням великого кола, що лежить на площині, на якій лежать ці точки і центр Землі, з подальшим визначенням довжини дуги сектора, обмеженого відрізками від центру кулі до точок на поверхні, як показано на рис. 2.14.

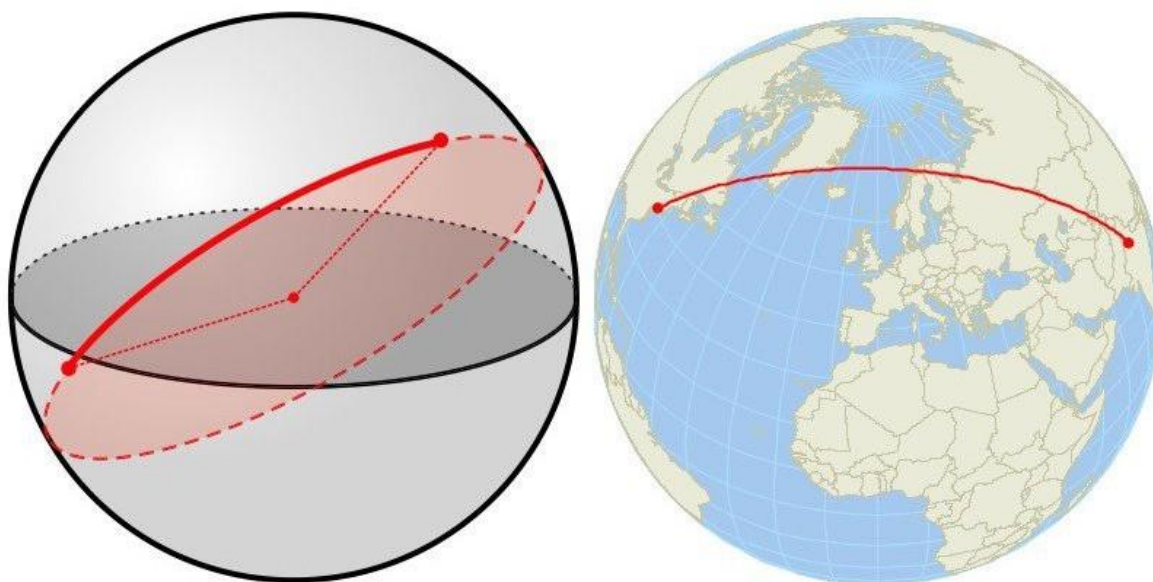


Рисунок 2.14 – Ілюстрація визначення найкоротшої відстані між двома точками по великому колі

Що є фактично вирішенням однієї з двох стандартних задач геометричній геодезії (обернена задача) - задані дві точки, необхідно визначити азимут і довжину відрізка (прямої лінії, дуги або геодезичної лінії) що сполучає ці дві точки.[15] Однак, форма Землі — не ідеальна сфера, а стиснутий біля полюсів сфероїд або еліпсоїд. Тому найкоротший шлях між двома точками на поверхні Землі (геодезична лінія) — не зовсім велике коло. Проте форма Землі мало відрізняється від кулястої, тому сферу можна

вважати першим наближенням земної поверхні. Похибки такого наближення зазвичай не перевищують 1%, для будь-яких практичних задач.[16] Однак ця особливість робить доречною реалізацію механізму, що дозволить користувачу системи підключати «plug-in» власні базові геодезійні функції, які мають бажану для цього застосування точність, і які будуть застосовуватись всіма алгоритмами системи. Можливості функціонального програмування обраного в попередньому підрозділі засобу та мови розробки (JavaScript + Node.js) зробить цю задачу легковирішувемою.[17]

Крім того, як в якості окремої, доступної користувачу функції, так і в якості елементу інших алгоритмів, необхідна реалізація визначення найкоротшої відстані до лінії відрізка (визначеного двома точками на поверхні) до іншої точки. У випадку попереднього застосування функції визначення відстані між двома точками по великому колу (ортодомі) рішення зводиться до простої тригонометрії. Для знаходження кута між відрізком і лінією від початку відрізка до точки:

$$B = \arccos\left(\frac{a^2 + c^2 - b^2}{2ac}\right)$$

де a - відстань до точки від початку відрізка; (2.6)

b - відстань від точки до кінця відрізка;

c - довжина відрізка.

Після чого знаходження добутка синуса кута B та довжини a , дасть відстань від точки до лінії.

Однак під час реалізації алгоритму необхідно звернути увагу на граничні випадки повного співпадіння двох точок, які визначають лінію, або окремої точки з однією точкою, що визначає лінію. В таку випадку точки створюють вироджений трикутник з однією стороною довжиною нуль, для якого приведена вище формула не є придатною. В першому випадку рішенням є відстань від точки до будь-якої з двох ідентичних точок що визначають відрізок, в другому - нульове значення.

2.3 Проект інформаційного забезпечення системи

2.3.1 Статична модель програми

Статистична модель – абстрактна схема відношень між величинами, що характеризують властивості реального процесу, розробка якої здійснюється неформальним шляхом. Статичні моделі називають також структурними, бо вони характеризують будову й параметри об’єкта. Статистична модель являє собою процес генерування даних, часто в значно ідеалізованій формі.[18]

В якості статичної моделі програми була розроблена у формі логічної моделі даних. Логічна модель даних, або логічна схема — модель даних конкретної предметної області, виражена незалежно від конкретного продукту керування базами даних або технології зберігання (фізична модель даних), але в термінах структур даних, таких як реляційні таблиці та колонки, об’єктно-орієнтовані класи чи теги XML.

Логічні моделі даних засновується на структурах, визначених у попередній концептуальній моделі даних (яка в даній роботі приведена в підрозділі 2.1.4), яка описує семантику організації без посилання на технологію.

Можна виділити наступні причини для побудування логічної моделі:

- 1) Допомогає загальному розумінню елементів бізнес-даних і вимог
- 2) Забезпечує основу для проектування бази даних
- 3) Сприяє уникненню надмірності даних, і таким чином запобігає неузгодженості даних і бізнес-транзакцій
- 4) Сприяє повторному використанню й обміну даними
- 5) Знижує час і вартість розробки та підтримки
- 6) Підтверджує логічну модель процесів і допомагає аналізувати вплив.

На рис. 2.15 представлена статична модель для “ІС для редагування географічних даних”.

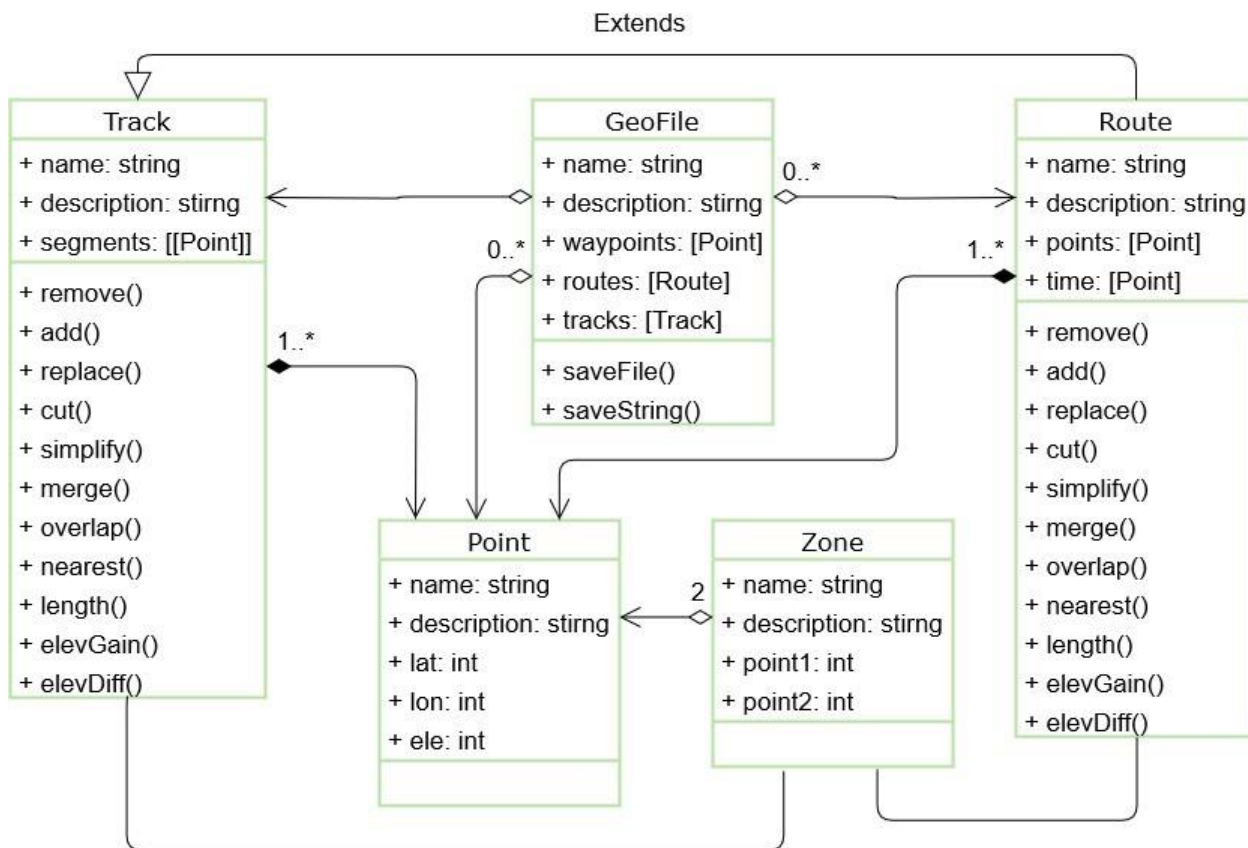


Рисунок 2.15 – Статична модель для ІС

2.3.2 Аналіз вхідних та вихідних даних

Серед великої кількості технологій передачі та зберігання географічних даних найбільш розповсюдженими стали три стандарти: GPX, KML та GeoJSON. За винятком інформації, пов'язаної з відображення географічних карт, більша частина даних кодується однією з цих технологій. Саме тому необхідність підтримки вводу і виводу даних в файли таких форматів стала очевидною ще на ранніх етапах розробки даної ГІС. Нижче буде здійснено огляд цих стандартів, їх особливостей, та основних типів одиниць даних, якими вони оперують.

GPX (GPS eXchange Format) - це текстовий формат зберігання і обміну даними GPS, заснований на XML. GPX є вільним форматом і може бути використаний без будь-яких ліцензійних відрахувань.

Формат дозволяє зберігати інформацію про орієнтири (waypoints), маршрутах (routes) і треках (tracklogs). Для кожної точки зберігаються її довгота, широта і висота над рівнем моря (якщо інформація наявна). Для точок треку зберігається також час проходження. XML-схема передбачає також зберігання довільної інформації користувачів по кожній точці, обов'язковими є тільки довгота і широта. Прикладом для користувача інформації може бути висота місцевості (ele), швидкість, частота пульсу (hr), число кроків, темп обертання педалей (каденс) (cad), температура, тощо.[19]

Основними типами даних для цього формату є маршрутна точка (waypoint), маршрут(route) і трек(track). Ці типи проілюстровані на рисунку 2.16.

Маршрутна точка позначається тегом <wpt> і означає окрему шляхову точку, яка по суті є маркером. Її координати задаються атрибутами lat та lon, висота може здаватися необов'язковим тегом <ele>. Точка може містити й інші теги, наприклад <name>(назва) або <desc> (опис).

Маршрутом є впорядкований список шляхових точок, які представляють ряд точок-поворотів, що ведуть до місця призначення. Крім власне точок повороту <rtept>(точка маршрута) може включати ім'я, опис, коментар, посилання тощо.

Трек - це послідовність точок з відмітками часу їх проходження, що позначається тегом <trk>. Містить в собі елементи <trkseg> (відрізки треку), які в свою чергу складаються з точок, що задаються тегом <trkpt>. Координати точок треку задаються аналогічно координатами для waypoint.

Традиційно у вигляді треків зберігається інформація про вже пройдені маршрути (наприклад записані дані з GPS пристроїв), в той час як маршрут зазвичай є вказівкою для руху в майбутньому.

Зазначені типи даних цього популярного формату стали де-факто стандартом для більшості ГІС, як на рівні користувача, так і всередині систем.

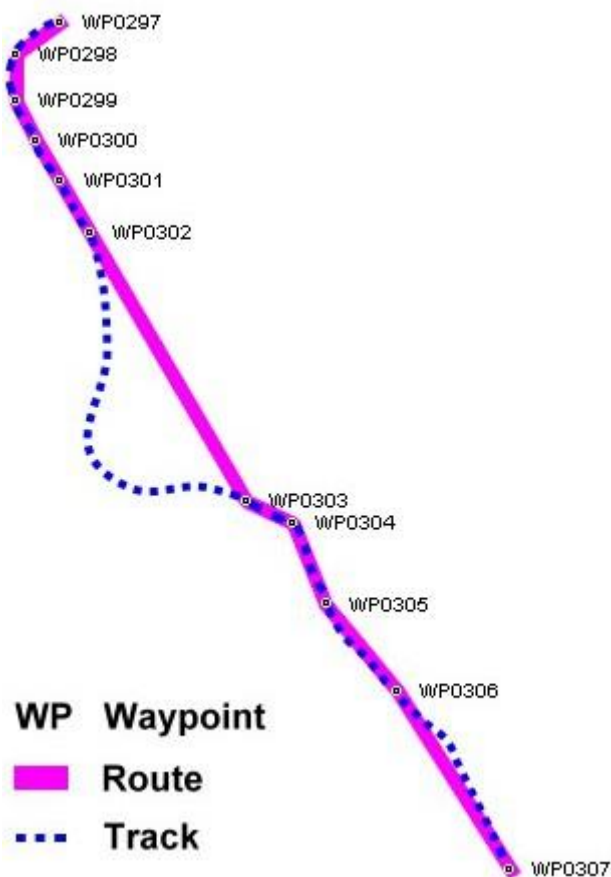


Рисунок 2.16 – Приклад маршрутних точок, трека та маршрута

Keyhole Markup Language (KML) - це ще один XML-формат для відображення географічних даних та візуалізації в двовимірних картах та тривимірних переглядачах земної поверхні. KML був розроблений для використання з Google Планета Земля. KML став міжнародним стандартом Open Geospatial Consortium у 2008 році.[20]

Файл KML визначає набір рис (позначки місць, зображення, полігони, 3D-моделі, текстові описи тощо), які можуть застосовуватись в ГІС що підтримують цей формат. Особливістю формату є його широкі можливості для збереження інформації щодо візуалізації даних. Наприклад, в файлі може бути закодовано яким кольором чи якою лінією має відібражатися той чи інший об'єкт. Хоча такі особливості роблять цей формат в першу чергу кращим вибором для використання в різноманітних переглядачах

геоінформації та електроних картах, ніж в системах що обробляють дані, формат часто використовується в якості універсального «контейнера» для збереження та передачі географічних даних. На рис. 2.17 показаний приклад вмісту KML файлу.

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
  <Document>
    <Placemark>
      <name>New York City</name>
      <description>New York City</description>
      <Point>
        <coordinates>-74.006393,40.714172,0</coordinates>
      </Point>
    </Placemark>
  </Document>
</kml>
```

Рисунок 2.17 – Точка, закодована в форматі KML

GeoJSON - відкритий формат призначений для зберігання географічних структур даних, заснований на JSON.

Формат може зберігати примітивні типи для опису географічних об'єктів, такі як: точки (адреси та місця розташування), лінії (вулиці, шосе, кордони), полігони (країни, області, ділянки землі). Також можуть зберігатися так звані мультитипи, які представляють собою об'єднання декількох примітивних типів.

Головною особливістю формату, яка робить його привабливим для використання в ГІС є його заснованість на JSON, що уможлиблює пряму інтерпретацію в об'єкти JavaScript. Ця обставина має враховуватись при виборі мови та засобів програмування.

2.4 Проект програмного забезпечення

2.4.1 Обґрунтування вибору мови програмування та засобів розробки

Зважаючи на природу застосування сучасних ГІС, багатоплатформність, тобто здатність програмного забезпечення працювати більш ніж на одній програмній (в тому числі — операційній системі) або апаратній платформі, були визначена як неодмінна умова для реалізованої системи.

Саме тому автори в першу чергу автори звернули увагу на технології веб та мобільної розробки, адже ця сфера наразі досягла найбільших успіхів в реалізації принципу «розробити програму один раз і розгорнути її де завгодно». Мова програмування JavaScript, що була розроблена як скриптова, сьогодні є сучасним та потужним засобом розробки для проектів будь-якої складності та масштабу і фактичним стандартом в багатьох сферах.

В сучасних середовищах JavaScript кросплатформність реалізована в два способи: на рівні середовища виконання (різні платформи можуть мати різні рушії); та за допомогою різноманітних інструментів, зазвичай фреймворків, що забезпечують виконання коду в багатьох середовищах.

Такі технології як Meteor, Cordova, ReactNative, Electron дозволяють розгорнути ПЗ написане на JavaScript практично на всіх сучасних платформах та операційних системах, включаючи Windows та Windows Server, Mac, Linux, Android, iOS та інші - від серверів до мобільних пристроїв.

В той самий час середовище Node.js пропонує стабільне середовище для розробки та тестування ПЗ, а також розвинену систему модульного програмування. Наразі еко-система JavaScript+Node.js має одну з найбільш потужних систем програмних модулів. Npm, менеджер пакетів для Node.js за замовчуванням, пропонує різноманітні додаткові бібліотеки та інструменти. Реєстр npm (фактично, база даних) наразі містить близько 836 000 бібліотек,

всі з яких можуть бути легко використані в коді, незалежно від того, для якої платформи він написаний.[21]

З одного боку це дозволить використовувати сторонні залежності під час реалізації ІС. Наприклад, засоби лексичного аналізу та створення XML документів - самостійна реалізація цих алгоритмів є очевидно поза межами теми роботи. А з іншого, готовий продукт може бути розгорнутий саме у вигляді прт пакета (модуля), що зробить його доступним самому широкому колу розробників.

Розвинена система модулів CommonJS, що підтримується цим середовищем, не лише дозволить розробникам в подальшому легко інтегрувати підсистему в свої проекти, але і дасть переваги під час розробки самої підсистеми для редагування геоданих. Так, розробнику не доведеться створювати велике монолітне ядро. Натомість, він зможе скористатися перевагами модульною системи і розробити набір модулів та підпрограм, кожна з яких може працювати у своєму процесі. Таким чином, процес розробки стає набагато гнучкішим і є ідеальним вибором для реалізації бібліотеки, що має містити велику кількість підпрограм які взаємодіють між собою.

Ще на етапі формування технічних вимог до ІС, була визначена необхідність підтримки формату GeoJSON, як одного з найпопулярніших та найуніверсальніших протоколів для передачі та збереження географічних даних. Хоча інші технології, такі як PHP та RubyOnRails, також підтримують обмін інформацією за допомогою формату JSON (на якому заснований GeoJSON), Node.js або інші JavaScript середовища роблять це з мінімальними перетвореннями та лексичним аналізом.[22] Адже структури даних JSON чи його різновидів є ідентичними до об'єктів та масивів JS. Це робить реалізацію вводу і виводу даних з таких форматів простою та ефективною. Таким чином, безперебійна комунікація з одним із основних стандартів передачі даних є ще однією перевагою екосистеми JavaScript.

Крім того, структури даних JavaScript є зручними для багатьох мережеских проєктів, що використовують NoSQL бази даних, наприклад, MongoDB. Особливо зважаючи на те, що розробка засобів взаємодії з базами даних не входить в завдання даної роботи.

Крім того не варто ігнорувати той факт, що JavaScript - одна з найпопулярніших і найпростіших для вивчення мов кодування в світі IT, легко вчитися розробниками-початківцями. Навіть особи, які не знають JavaScript, але мають деякі основні технічні знання, зазвичай можуть читати та розуміти код. Це є доречним, зважаючи на те, що основним варіантом використання розроблюваної системи є її інтеграція в інші програмні продукти в якості підсистеми (бібліотеки підпрограм). Використання популярної та сучасної мови програмування дозволить розширити коло потенційних користувачів - від аматорів, що хотіли би написати скрипт з використанням функцій системи, до розробників масштабних проєктів в сфері ГІС.

Не в останню чергу перевагою середовища Node.js є відкритість його коду (open-source). Продукт не лише стовідсотково безкоштовний, але й автори надають кожному право вивчати, розвивати та поширювати технології для будь-яких цілей. Це робить використання середовища в наукових та освітніх цілях дуже доречним.

2.4.2 Розробка діаграми компонентів системи

Діаграма компонентів, Component diagram – статична структурна діаграма, показує розбиття програмної системи на структурні компоненти та зв'язки (залежності) між компонентами. В якості фізичних компонентів можуть виступати файли, бібліотеки, модулі, виконувані файли, пакети.

Компоненти зв'язуються через залежності, коли з'єднується необхідний інтерфейс одного компонента з наявним інтерфейсом іншого компонента.

Таким чином ілюструються відносини клієнт-джерело між двома компонентами. Залежність показує, що один компонент надає сервіс, необхідний іншому компоненту. Залежність зображується стрілкою від інтерфейсу або порту клієнта до імпортованого інтерфейсу.

На рис. 2.18 представлена діаграма компонентів для «ІС для редагування географічних даних».

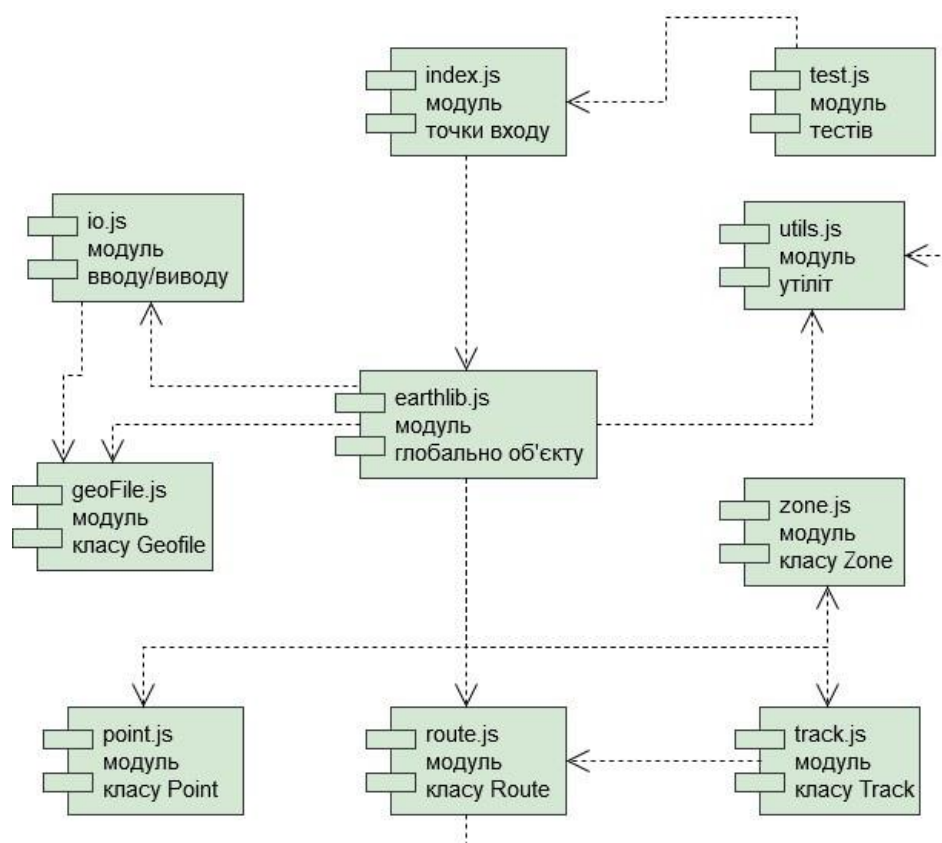


Рисунок 2.18 – Діаграма компонентів для «ІС для редагування географічних даних»

2.4.3 Кодування системи

Кодування — процес написання сирцевого коду обраною або обраними мовами програмування з метою реалізації певного алгоритму або алгоритмів.

В більш точному сенсі — процес перетворення моделей проектування в моделі реалізації обраною мовою програмування.

В цьому підрозділі описані варті уваги особливості реалізації розробленого проекту. Кодування інформаційної системи відбувалося в середовищі розробки Visual Studio Code версії 1.4, обраною мовою програмування JavaScript (ECMAScript 2018) з використанням модулів CommonJS. Тестування проводилось на платформу Node.js версії 12.14.

По-перше, варто виділити програмне рішення для забезпечення можливості додання власних геодезійних функцій в якості «плагіна», яке було можливе завдяки функціональним та модульним можливостям обраних засобів програмування. Під час імпорту підсистеми в основну систему користувач (розробник) може передати власні функції в якості параметрів. В подальшому, система буде використовувати саме ці функції під час виконання всіх алгоритмів, викликаючи їх згідно з зазначеними характеристиками (сигнатурою). поряд з асинхронними функціями та інтерфейсами Promise, описаними нижче, таке рішення є прикладом реалізації принципу передачі контролю за виконанням.

По-друге, також заслуговує уваги реалізацію паралельного виконання деяких операцій пов'язаних з запитами в файлову систему або до мережі. Для цього використовувався вбудований в Node модуль fsPromises, що являє собою спосіб виконання фонових задач через інтерфейс Promise (обіцянка). Це дозволяє уникнути «підвисання» в роботі інтерфейсу при затримках відповіді мережі або файлової системи або тривалих розрахунків при великих об'ємах імпортованих чи експортованих файлах. Підхід заключається в тому, що в результаті ініціації запиту основний процес отримує обіцянку (Promise), а той час як у фоновому процесі викликається попередньо зареєстрована функція зворотнього виклику, яка отримує в якості параметра запит. Коли дана функція повертає результат, обіцянка в основному потоці виконується цим результатом. Така схема дає можливість ініціювати декілька завдань в

одному фоновому потоці, не очікуючи завершення попередніх (наприклад для обробки декількох одночасно доданих файлів), адже забезпечується, що у відповідь на кожен запит надійде саме відповідний йому результат. На рис. 2.19 приведено фрагмент коду, який ініціює додавання даних в систему з файлу.

```
function addFile(path,type){
  type = matchFormat(path)
  return fs.readFile(path,'utf8').then(content=>addString(content,type,path), err=>{
    throw new Error(`Problems parsing ${path}. Please check the validity of your file.`)
  })
}
```

Рисунок 2.19 – Ініціація інтерфейсу fsPromises

По-третє, розвинена система модулів та пакунків реєстру npm дозволило використання бібліотеки «xml-js» в якості сторонньої залежності. Підпрограми цього модуля забезпечили як стабільно коректне здійснення синтаксичного аналізу та перевірки валідності наявних XML файлів (до яких відносяться два з трьох основних підтримуваних форматів - KML та GPX), так і створення таких файлів для експорту даних з системи.

Текст коду інформаційної системи приведено в Додатку Г.

2.4.4 Тестування системи

Тестування програмного забезпечення — це процес технічного дослідження, призначений для виявлення інформації про якість продукту відносно контексту, в якому він має використовуватись. Техніка тестування також включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою оцінки. Оцінюється: відповідність вимогам, якими керувалися проектувальники та розробники; правильна відповідь для усіх можливих вхідних даних; виконання функцій за прийнятний час; практичність; сумісність з програмним забезпеченням та операційними системами; відповідність задачам замовника.[23]

Оскільки число можливих тестів навіть для нескладних програмних компонент практично нескінченне, тому стратегія тестування полягає в тому, щоб провести всі можливі тести з урахуванням наявного часу та ресурсів. Для цього використовується підхід класів еквівалентності. Він полягає в наступному: вхідні / вихідні дані розбиваються на класи еквівалентності, за принципом, що програма веде себе однаково з кожним представником окремого класу. Таким чином, немає необхідності тестувати всі можливі вхідні дані, необхідно перевірити по окремо взятому представнику класу.

Тобто клас еквівалентності – це набір значень змінної, який вважається еквівалентним. Тестові сценарії еквівалентні, якщо:

- 1) Вони тестують одне і те ж;
- 2) Якщо один з них знаходить помилку, то й інші виявлять її;
- 3) Якщо один з них не знаходить помилку, то й інші не виявлять її.

Також варто розглянути особливості тестування методом «білої скриньки» та методом «чорної скриньки».

Біла скринька (WhiteBox) – цей метод заснований на тому, що розробник тесту має доступ до коду програм і може писати код, який пов'язаний з бібліотеками ПЗ, що тестується.

Чорна скринька (Black Box) – цей метод заснований на тому, що тестувальник має доступ до ПЗ тільки через ті ж інтерфейси, що і замовник або користувач, або зовнішні інтерфейси, що дозволяють іншому комп'ютеру або іншому процесу підключитися до системи для тестування.

Сіра скринька (Grey Box) – поєднує елементи двох попередніх підходів.

У рамках розробки «ІС для редагування географічних даних» проведено його тестування методом «чорної скриньки» та «сірої скриньки».

Під час тестування програмного забезпечення перевірялась відповідність складу даних, що вводяться їх значенням, коректність результатів роботи алгоритмів, валідність та правильний склад вихідних даних.

Нижче представлені розроблені тестові набори даних та результати їх тестування. Варто зазначити, що тестування здійснювалось при використанні системи як за допомогою інтерфейсу командного рядка, так і в якості підсистеми.

Вхідні данні для функції «I/O даних» представлені у табл. 2.1

Таблиця 2.1 – Опис вхідних даних функції «I/O даних»

Назва вхідних даних	Тип
Файл з треками та точками	GPX файл
Файл з треками та точками	KML файл
Файл з треками, точками та зонами	GeoJSON файл
Невалідний файл	XML файл

Для тестування функції було виділені класи еквівалентності вхідних та вихідних даних, які наведені в таблиці 2.2.

Таблиця 2.2 – Класи еквівалентності для функції «I/O даних»

Вхідні умови	Правильні класи еквівалентності	Неправильні класи еквівалентності
Валідні файли GPX, KML та GeoJSON, що містять географічну інформації (точки, треки, зони).	1. Валідні вхідні файли.	2. Невалідні вхідні файли.
	3. Валідні та ідентичні за змістом файли, збережені в тому ж форматі.	
	4. Валідні та ідентичні за змістом файли, збережені в іншому форматі.	

В таблицях 2.3 і 2.4 наведені параметри побудованих тестів з використанням класів еквівалентності наведених вище. Передбачається що у випадку спроби користувача імпортувати некоректний файл програма повідомить користувача чи основну систему про помилку.

Таблиця 2.3 – Тести правильних класів еквівалентності для функції «І/О даних» згідно таблиці 2.2

№ тесту	Класи еквівалентності	Вхідні дані	Очікуваний вихід
1	1	GPX, KML та GeoJSON файли з геоданими.	Інформація додана в систему коректно та в повному обсязі.
2	3		Файли збережені коректно та з інформацією в повному обсязі.
3	4		Файли збережені в іншому форматі коректно та з ідентичною інформацією.

Таблиця 2.4 – тести неправильних класів еквівалентності для функції
«I/O даних» згідно таблиці 2.2

№ тесту	Класи еквівалентності	Вхідні дані	Очікуваний вихід
1	2	Невалідний GPX, KML або GeoJSON файл.	Виведення повідомлення про помилку.

На рис. 2.20 представлений приклад результату тестування – покриття правильних класів еквівалентності за тестом №1. Як і очікувалось, вся інформація, що містилась в GPX файлі була додана в систему в повному обсязі та збережена в об'єкті класу GeoFile для подальшого використання. Аналогічний результат було досягнуто при вводі даних з файлів інших форматів - KML та GeoJSON.

```

2 <gpx xmlns="http://www.topografix.com/GPX/1/1" creator="earthlib.js" >
3   <metadata>
4     <name>Test file GPX</name>
5     <time>2019-12-21T11:24:27Z</time>
6   </metadata>
7   <wpt lat="46.64643904" lon="32.61118769">
8     <ele>48.000000</ele>
9     <name>Університет</name>
10    <desc>Х0 НУК</desc>
11  </wpt>
12  <wpt lat="46.64837216" lon="32.61056005">
13    <ele>45.000000</ele>
14    <name>Пам'ятник</name>
15    <desc>Пам'ятник Джону Говарду</desc>
16  </wpt>
17  <wpt lat="46.64505042" lon="32.60919749">
18    <ele>43.000000</ele>
19    <name>Лікарня</name>
20    <desc>Херсонська обласна лікарня</desc>
21  </wpt>
22  <trk>
23    <name>Трек n1</name>
24    <desc>Перша частина прогулянки</desc>
25    <trkseg>
26      <trkpt lat="46.64933698" lon="32.61087119">
27        <ele>45.000000</ele>
28        <time>2010-01-01T00:00:00Z</time>
29      </trkpt>
30      <trkpt lat="46.64882143" lon="32.60914385">
31        <ele>48.000000</ele>
32        <time>2010-01-01T00:00:51Z</time>
33      </trkpt>
34      <trkpt lat="46.64642775" lon="32.61056005">
35        <ele>45.000000</ele>
36        <time>2010-01-01T00:02:35Z</time>
37      </trkpt>
38      <trkpt lat="46.64741470" lon="32.61410057">
39        <ele>49.000000</ele>
40        <time>2010-01-01T00:04:20Z</time>
41      </trkpt>
42      <trkpt lat="46.64798181" lon="32.61278092">
43        <ele>44.000000</ele>
44        <time>2010-01-01T00:05:02Z</time>
45      </trkpt>
46      <trkpt lat="46.64832797" lon="32.61279165">
47        <ele>45.000000</ele>
48        <time>2010-01-01T00:05:16Z</time>
49      </trkpt>

```

Рисунок 2.21 – Вміст збереженого файлу

На рис. 2.22 – покриття неправильного класу еквівалентності за тестом №4. В результаті спроби додати в систему невалідний файл за допомогою інтерфейсу командного рядка користувачу було показане повідомлення про невідповідність файлу стандартам формату.

```
C:\work\earthlib>node cli --input test/testINV.gpx
Problems parsing test/testINV.gpx. Please check the validity of your file.
```

Рисунок 2.22 – Відображення повідомлення про помилку

Далі представлені результати тестування для функції «Створення точок та маршрутів». Вхідні дані представлені у таблиці 2.5

Таблиця 2.5 – Вхідні дані функції «Створення точок та маршрутів»

Назва вхідних даних	Тип
Координати точки	Array<float>
Невалідні координати	Array<float>
Висота над рівнем моря	float
Коментар	String

Передбачається, що при спробі створити точку з недопустимими значеннями довготи і широти система виведе повідомлення про помилку. При коректних значеннях точка буде створена для подальшого використання для визначення частини маршруту або як самостійна одиниця. В таблиці 2.6 виділені класи еквівалентності тестових випадків для цієї функції.

Таблиця 2.6 – Класи еквівалентності для функції «Створення точок та маршрутів»

Вхідні умови	Правильні класи еквівалентності	Неправильні класи еквівалентності
Завантажений або створений в системі об'єкт, що представляє файл з географічними	1. Коректні чисельні значення довготи, широти, і висоти.	2. Некоректні значення довготи або широти.
	3. Текстове значення	

даними.	коментаря та назви.
---------	---------------------

Сформовані тести для правильних та неправильних класів еквівалентності наведені в таблицях 2.7 та 2.8 відповідно.

Таблиця 2.7 – тести правильних класів еквівалентності для функції
«Створення точок та маршрутів»

№ тесту	Класи еквівалент- ності	Вхідні дані	Очікуваний вихід
1	1	Значення широти і в діапазоні -90.0 - 90.0 і -180.0 - 180.0 відповідно, чисельне значення висоти.	Створення і збереження в пам'яті точки з заданими параметрами.
2	3	Текстове значення коментаря та назви.	Додання коментаря і назви до створеної точки.

Таблиця 2.8 – тести неправильних класів еквівалентності для функції
«Створення точок та маршрутів»

№ тесту	Класи еквівалентності	Вхідні дані	Очікуваний вихід
1	2	Значення широти і довготи поза діапазоном -90.0 - 90.0 і -180.0 - 180.0 відповідно.	Виведення повідомлення про помилку.

На рис. 2.23 та 2.24 представлені результати тестування – покриття правильного класу еквівалентності за тестами № 1 та №2 відповідно. Об'єкт

класу Point створений та доступний для подальшого використання, він має задані значення місцезположення, назви та коментаря.

```
> new earthlib.Point(49,32)
Point { lat: 49, lon: 32, ele: undefined, name: '', description: '' }
>
```

Рисунок 2.23– Результат створення маршрутної точки

```
> let point = new earthlib.Point(49,32)
undefined
> point.name = 'Точка1'
'Точка1'
> point.description = 'Тестова точка'
'Тестова точка'
>
> point
Point {
  lat: 49,
  lon: 32,
  ele: undefined,
  name: 'Точка1',
  description: 'Тестова точка'
}
```

Рисунок 2.24 – Результат покриття визначення назви точки та додання коментаря

На рис. 2.25 представлені результати покриття неправильного класу еквівалентності. Під час спроби створення точки з неможливими значеннями довготи і широти програми викинула помилку класу RangeError.

```
> new earthlib.Point(999,-999)
Thrown:
RangeError: Invalid coordinates values - 999 -999
    at new Point (C:\work\earthlib\lib\point.js:4:12)
>
```

Рисунок 2.25 – Результат спроби створення точки з некоректними значеннями

Назва вхідних даних	Тип
трек_1	Array<Point>
трек_2	Array<Point>
точка	<Point>

Вхідні умови	Правильні класи еквівалентності	Неправильні класи еквівалентності
Два трека, що містять більше 10 точок кожен.	1. Валідний трек для скорочення і точка відліку.	2. Трек, що не містить надлишкових точок.
	3. Валідний трек для спрощення, який містить надлишкові точки.	4. Треки для об'єднання, що не мають спільних ділянок.
	5. Два трека для об'єднання, що мають спільну ділянку в кінці першого трека і на початку другого.	

Таблиця 2.11 – Тести правильних класів еквівалентності для функції
«Редагування треків та маршрутів»

№ тесту	Класи еквівалентності	Зміст тесту	Очікуваний вихід
1	1	Скорочення треку від точки відліку.	Збережені всі точки трека після точки, що є найближчою до точки відліку.
2	3	Спрощення трека з точністю 20м.	Спрощений трек, лінія якого не відхиляється від первинної більше ніж на 20м.
3	5	Об'єднання двох треків.	Трек, що складається з точок двох треків, при чому спільна ділянка збережена один раз.

Сформовані тести для правильних класів еквівалентності наведені в таблиці 2.11.

На рис. 2.26 та рис. 2.27 представлені результати тестування функції об'єднання треків(маршрутів). На рисунку перший трек (позначений синім) та другий (позначений) жовтим мають спільну ділянку в кінці та на початку відповідно. Функція об'єднала їх в даному місці та зберегла спільну ділянку лише один раз (об'єднаний трек позначений червоним).

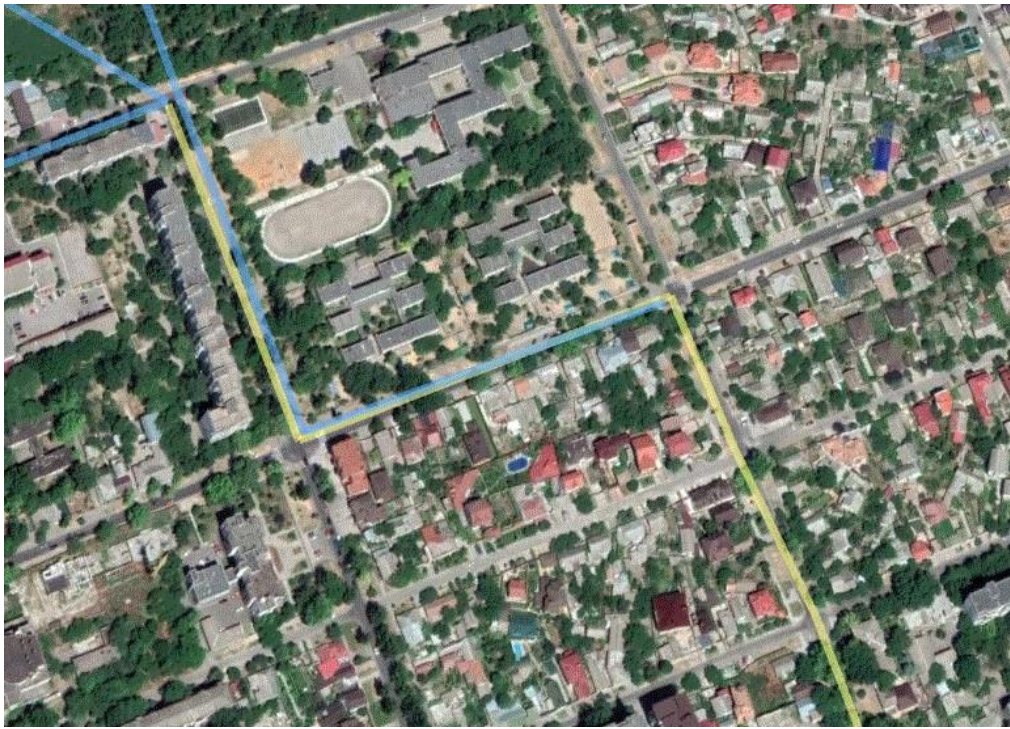


Рисунок 2.26 – Два треки для об'єднання



Рисунок 2.27 – Об'єднаний трек

Тести неправильних класів еквівалентності наведені в таблиці 2.12. Передбачається, що при об'єднання двох треків, які не мають спільних ділянок, маршрутні треки будуть збережені як два фрагмента одного трека. Це є бажаною практикою для уникнення «стрибків» між кінцем першого і початком другого треку. При збереженні трека в форматі, що не передбачає декількох фрагментів одного треку (маршруту), користувачу буде показане повідомлення про помилку як показано на рис. 2.28.

Таблиця 2.12 – Тести неправильних класів еквівалентності для функції «Створення точок та маршрутів».

№ тесту	Класи еквівалентності	Зміст тесту	Очікуваний вихід
4	4	Спрощення трека.	Незмінний трек.
5	6	Об'єднання треків що не мають спільних ділянок.	Збереження двох треків в окремих сегменти.

```
Route 1
Name: Маршрут
Number of points: 13
Length: 1706.51m
Not every point in this track(route) has valid elevation data!
Description: Маршрут без позначення висот
```

Рисунок 2.28 – Повідомлення про відсутність зони перетину в двох маршрутах

На виході функції «Спрощення» треку, який не мав надлишкових точок, було отримано ідентичний трек.

Далі представлені результати тестування для функції «Аналіз окремих точок». Тестуються функції визначення відстані між двома точками, напрям між ними, а також визначення відстані від точки до лінії. Вхідні данні для цих функцій представлені у таблиці 2.5

Таблиця 2.13 – Вхідні дані функції «Аналіз окремих точок»

Назва вхідних даних	Тип
точка_1	<Point>
точка_2	<Point>
точка_3	<Point>

Передбачається, що вказані утилітарні функції є «чистими» і тому мають давати коректний результат незалежно від стану системи. В таблиці 2.14 виділені класи еквівалентності для цієї функції.

Таблиця 2.14 – Класи еквівалентності для функції «Аналіз окремих точок»

Вхідні умови	Правильні класи еквівалентності	Неправильні класи еквівалентності
Відсутні.	1. Дві валідні точки з різним місцеположенням.	
	2. Три валідні точки з різними місцеположенням.	

Таблиця 2.15 – Тести правильних класів еквівалентності для функції «Аналіз окремих точок»

№ тесту	Класи еквівалентності	Зміст тесту	Очікуваний вихід
1	1	Знаходження відстані між двома точками по «великому	Правильна відстань між точками в метрах.

		колу».	
2	1	Знаходження напрямку між двома точками (курсу від першої до другої) .	Правильний напрям градусів.
3	3	Знаходження відстані від точки №3 до лінії, проведеною по «Великому колу» між точкою №1 та точкою №2.	Правильна відстань в метрах.

Сформовані тести наведені в таблиці 2.15. На рис. 2.29 представлені результати тестування – значення відстані та напрямку, які є в межах похибки системи координат. Наприклад, відстань між точками, розташованими на одній довготі на відстані однієї хвилини (1/60 градуса) одна від одної дорівнює 1853.2 метра, що лише на один метр більше від морської милі.

```

> let a = new earthlib.Point(0,0)
undefined
> let b = new earthlib.Point(1/60,0)
undefined
> let c = new earthlib.Point(0,1)
undefined
> earthlib.utils.distance(a,b)
1853.249
> earthlib.utils.bearing(a,b)
0
> earthlib.utils.bearing(a,c)
90
> earthlib.utils.distanceToLine(b,c,a)
1852.9916857009157
>

```

Рисунок 2.29– Створення тестових точок та отримані значення Далі представлені результати тестування для функції «Аналіз треків та маршрутів». В рамках цієї функції тестується визначення довжини маршрута, перепаду висоти, набору та втрати висоти під час руху по маршруту. Вхідні дані для цієї функції представлені у таблиці 2.5.

В рамках цієї функції тестується визначення довжини маршруту, перепаду, набору та втрати висоти під час руху по маршруту.

Таблиця 2.16 – Вхідні дані функції «Аналіз треків та маршрутів»

Назва вхідних даних	Тип
трек	Array<Point>
трек_noEle	Array<Point>

В таблиці 2.16 виділені класи еквівалентності для цієї функції. Слід зазначити, що наявність значення висоти точки над рівнем моря не є обов'язковим для більшості форматів геоданих, тому трек без такої інформації (як трек_noEle) лишається валідним.

Таблиця 2.17 – Класи еквівалентності для функції «Аналіз треків та маршрутів»

Вхідні умови	Правильні класи еквівалентності	Неправильні класи еквівалентності
Імпортований або створений в системі валідний трек.	1. Трек з коректно визначеними висотами точок.	2. Трек з невизначеними висотами точок.

Сформовані тести для правильних та неправильних класів еквівалентності наведені в таблицях 2.18 та 2.19 відповідно.

Таблиця 2.18 – Тести правильних класів еквівалентності для функції
«Аналіз треків та маршрутів»

№ тесту	Класи еквівалент- ності	Зміст тесту	Очікуваний вихід
1	1	Знаходження, перепаду набору та втрати висоти при русі по маршруту.	Правильні значення висоти в метрах.
2	1, 2	Знаходження загальної довжини трека, як суми всіх його відрізків.	Правильна довжина в метрах.

Таблиця 2.19 – Тести неправильних класів еквівалентності для функції
«Аналіз треків та маршрутів»

№ тесту	Класи еквівалентності	Зміст тесту	Очікуваний вихід
3	2	Спроба знаходження набору висоти на маршруті, точки якого не мають визначеної висоти на рівнем моря.	Значення NaN та попередження про помилку.

На рис. 2.30 представлені результати тестування за правильними класами еквівалентності. Було правильно визначено довжину трека, загальну різницю висоти, набір та втрату висоти. Ця інформація представлена користувачу разом з кількістю маршрутних точок, назвою та коментарем до треку.


```

Track 1
Name: Трек n1
Number of points: 16
Length: 1799.25m
Elevation difference: 6m
Elevation gain: 10m
Elevation lost: -15m
Description: Перша частина прогулянки

```

Рисунок 2.30 – Виведення результату аналізу треку

```

C:\work\earthlib>node cli --input test/test.gpx --routes 1 --analyze

Route 1
Name: Маршрут
Number of points: 13
Length: 1706.51m
  Not every point in this track(route) has valid elevation data!
Description: Маршрут без позначення висот

```

Рисунок 2.31 – Попередження про відсутність даних щодо висоти

На рис. 2.31 представлені результати покриття неправильного класу еквівалентності. Система не може визначити значення висоти за умови відсутності даних, про що вона повідомляє користувача.

Далі представлені результати тестування для функцій визначення вміщуючих зон. Вхідні данні для цієї функції представлені у таблиці 2.20.

Таблиця 2.20 – Вхідні дані функції «Визначення вміщуючих зон»

Назва вхідних даних	Тип
зона_1	Array<Point>
зона_2	Array<Point>
трек_1	Array<Point>

Вхідні умови	Правильні класи еквівалентності	Неправильні класи еквівалентності
Відсутні.	1. Валідний трек.	
	2. Дві валідні географічні зони, визначені їх кутовими точками.	

№ тесту	Класи еквівалентності	Зміст тесту	Очікуваний вихід
1	1	Знаходження прямокутної зони, що вміщає всі точки трека.	Правильна зона, визначена координатами двох її кутових точок.
2	2	Знаходження прямокутної зони, що вміщає в себе декілька інших зон.	Правильна зона, визначена координатами двох її кутових точок.

На рисунку 2.31 показаний результат знаходження прямокутника, який вміщає всі елементи маршрута. Для наглядності об'єкти накладені на супутниковий фотознімок місцевості.



Рисунок 2.31 – Зона, яка вміщає трек

РОЗДІЛ 3. РЕЗУЛЬТАТИ РОЗРОБКИ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Розробка інструкції користувача

Інструкція користувача - документ, призначення якого надати користувачам допомогу у використанні деякої системи. Документ входить до складу технічної документації на систему і, як правило, готується технічним співробітником. Використовується стиль і мова, доступні передбачуваній аудиторії, використання жаргону скорочується до мінімуму або докладно пояснюється.[24]

Крім того, зважаючи на те, що розроблена система може використовуватись як бібліотека підпрограм, необхідна розробка документації до прикладного програмного інтерфейсу (API). В цій частині інструкції буде приведено набір визначених методів для взаємодії різних компонентів для надання розробнику засобів для швидкої розробки програмного забезпечення.

Інструкція користувача для розроблюваної в даній роботі інформаційної системи представлена в Додатку В.

3.2 Випробування системи

Під час випробувань, які проводились згідно з програмою та методикою випробувань «ІС для редагування географічних даних», представлених у Додатку Б, були отримані результати, описані нижче.

Випробування проводились з метою виявлення неточностей і помилок, допущених при кодуванні і розробці алгоритму, а також для надання розробленому програмному продукту закінченого та готового вигляду, придатного до застосування.

Вимогами до програми, на основі яких проводились випробування, виступають: зручний у використанні інтерфейс командного рядка;

функціональний та послідовний програмний інтерфейс; можливість експорту та імпорту даних в підтримуваних форматах; можливість створення нових даних та зміни існуючих; здатність здійснення аналізу окремих точок, маршрутів, зон.

Для проведення випробувань була пред'явлена програмна документація у складі: технічне завдання; специфікація програми; текст програми.

Випробування проводились при наявності комп'ютера з встановленим середовищем Node.js згідно даної розробленої методики випробувань. Для реалізації всіх методів випробувань програма запускала дві рази: один раз в режимі інтерфейсу командного рядка, другий раз в якості підпрограми.

Отримані такі результати:

1) Функція імпорту даних.

Випробовується можливість додавати нові дані з файлів GPX, KML та GeoJSON з перевіркою їх валідності та подальшим відображенням основних відомостей про додані дані. Під час випробування були успішно додані дані з двох файлів. Загальна інформація про імпортовані дані коректно виводилась на екран, як показано на рис. 3.1.

```
C:\work\earthlib>node cli --input test/test.gpx
File test.gpx
Waypoints:
  1. Name: Університет   Description: ХФ НУК
  2. Name: Пам'ятник    Description: Пам'ятник Джону Говарду
  3. Name: Лікарня      Description: Херсонська обласна лікарня

Tracks:
  1. Name: Трек n1      Description: Перша частина прогулянки
  2. Name: Трек n2      Description: Друга частина прогулянки

Routes:
  1. Name: Маршрут      Description: Маршрут без позначення висот
```

Рисунок 3.1— Інформація про введені дані

2) Функція експорту даних.

Випробовується функція збереження даних в файлах форматів GPX, KML або GeoJSON з попередньою адаптацією даних для обраного формату. Під час випробування дані були збережені в двох різних форматах. На виході були отримані валідні файли GPX та GeoJSON.

3) Функція створення та зміни географічних даних.

Випробовується функція створення географічних точок, маршрутів, треків, а також зміна параметрів існуючих одиниць даних. Під час випробування були успішно змінені параметри трьох маршрутних точок та створено три нові. Програма запобігла спробі задання некоректних значень координат, і вивела повідомлення про помилку. Крім того було випробуване спрощення та скорочення маршрутів і треків. На рис. 3.2 показаний результат операції спрощення.

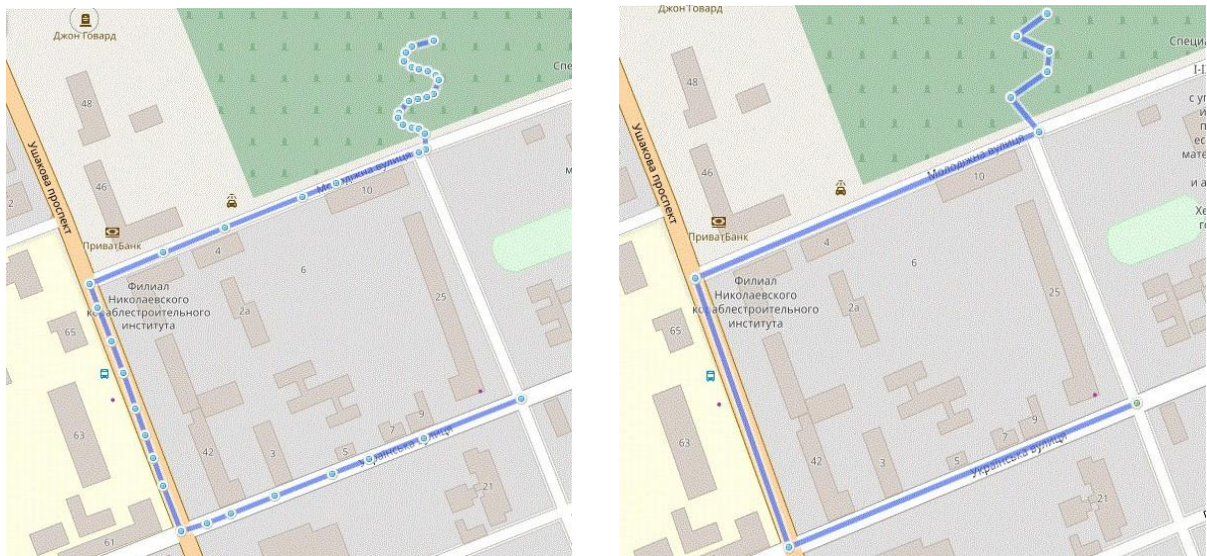


Рисунок 3.2 – Порівняння початкового маршруту (ліворуч) зі спрощеним(праворуч)

4) Функція здійснення аналізу точок, маршрутів, зон.

Було випробувано різноманітні функції аналізу географічних даних, такі як: знаходження відстані та напрямку між точками, довжини маршруту, набору та втрати висоти на маршруті, спільних ділянок

маршрутів, географічних зон, які вміщують інші зони або маршрути. Всі операції були завершені успішно та результат відповідав вимогам, зазначеним в технічному завданні.

```
C:\work\earthlib>node cli --input test/test.gpx --analyze

Route 1
Name: Маршрут
Number of points: 13
Length: 1706.51m
  Not every point in this track(route) has valid elevation data!
Description: Маршрут без позначення висот

Track 1
Name: Трек n1
Number of points: 16
Length: 1799.25m
Elevation difference: 6m
Elevation gain: 10m
Elevation lost: -15m
Description: Перша частина прогулянки

Track 2
Name: Трек n2
Number of points: 56
Length: 2764.27m
Elevation difference: 51m
Elevation gain: 58m
Elevation lost: -13m
Description: Друга частина прогулянки
```

Рисунок 3.3 – Результат аналізу двох треків та одного маршруту

5) Функція взаємозв'язків між формами користувацького інтерфейсу програми.

Було випробувано користувацький інтерфейс ІС. Він виявився зручним, інтуїтивним та послідовним; переходи між формами користувацького інтерфейсу були коректними та дозволяли користувачеві реалізовувати її функціонал. Програма не зазнала збоїв у роботі.

Таким чином, згідно з випробуваннями, «ІС для редагування географічних даних» відповідає поставленому технічному завданню.

РОЗДІЛ 4. ТЕХНІКО-ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ РОЗРОБКИ

Сьогодні необхідність використання інформаційних систем та інформаційних технологій в управлінні підприємством вже не викликає сумнівів. Адже дозволяє керівництву будь-якої компанії приймати правильні і своєчасні стратегічні рішення навіть у найскладніших ситуаціях, коли необхідно оперувати величезною кількістю інформації. Проте використання даних засобів інформаційного менеджменту є не завжди економічно ефективним для підприємства. Тому оцінка ефективності функціонування ІС та ІТ є необхідною процедурою будь-якого підприємства, яка їх використовує.

Поняття ефективності у контексті інформаційного менеджменту підприємства представляє собою певну результативність процесу, операції, проекту, обумовлену як співвідношення результату до витрат, що забезпечили його одержання. Економічною ефективністю є результативність виробництва, співвідношення між результатами господарської діяльності й витратами праці та капіталу на досягнення даного результату.

Іншими словами під ефективністю інформаційного менеджменту розуміється комплекс показників, що характеризують позитивний вплив експлуатації інформаційної системи, інформаційних технологій та інших технічних засобів в управлінні економічною системою підприємства.

Ефективність використання інформаційної системи та інформаційних технологій в управлінні економічними завданнями підприємства перебуває у прямій залежності від двох головних факторів: впливу на підвищення рівня управління та пов'язаного з ним поліпшення діяльності об'єкта через зниження витрат підприємства. У зв'язку з цим, розрізняють два види ефективності: прямий та непрямий.

Прямий ефект знаходить висвітлення у деяких результатних показниках діяльності та може бути виражений в натуральних, вартісних, трудових показниках (кількість вивільнених працівників управління,

економія робочого часу в людино-годинах, продуктивність праці управлінських працівників тощо).

Непрямий ефект відбивається у показниках виробничої господарської діяльності підприємства завдяки використанню більш якісної інформації, але важко піддається безпосередньому кількісному виміру.

У зв'язку з тим, що непряма ефективність проявляється у якісних факторах, а пряма виявляється безпосередньо кількісними величинами, то ці види ефекту ще називають (відповідно) якісними й кількісними.

Процеси управління інформаційною інфраструктурою підприємства, та зокрема процеси впровадження ІТ, пов'язані з серйозними вкладками капіталу на придбання техніки, розробку проектів, виконання підготовчих робіт і забезпечення персоналу. Тому у ході реалізації даної концепції необхідне здійснення економічного обґрунтування доцільності запланованих заходів.

Таким чином, завдання і зміст техніко-економічного обґрунтування автоматизованої інформаційної системи полягають у наступному:

- 1) доказ доцільності створення або розвитку АС на основі аналізу складових;
- 2) соціальна потреба в системі;
- 3) технічні, математичні, інформаційні та організаційні можливості (якісна оцінка системи);
- 4) економічна доцільність розробки АС;
- 5) доказ доцільності впровадження ІС на основі розрахунків:
- 6) одноразових витрат на впровадження;
- 7) поточних витрат на функціонування системи;
- 8) результатів, отриманих при впровадженні ІС;
- 9) показників економічної ефективності інвестиційних витрат.

4.1 Розрахунок витрат на розробку автоматизованої системи (ІС)

Витрати на розробку ІС ($K_{\text{п}}$) визначаються за формулою:

$$K_{\Pi} = K_{\text{пр}} + K_{\text{по}} + K_{\text{іо}} + K_{\text{о}}, \quad (4.1.1)$$

де $K_{\text{пр}}$ - витрати на проектування ІС;

$K_{\text{по}}$ - витрати на створення програмних виробів, що утворюють програмне забезпечення ІС;

$K_{\text{іо}}$ - витрати на підготовку інформаційного забезпечення тривалого користування, створення бази даних ІС;

$K_{\text{о}}$ - витрати на налагодження ІС

Розрахунок

$K_{\text{пр}} - 10000$ грн.

$K_{\text{по}} - 5000$ грн.

$K_{\text{іо}} - 7000$ грн.

$K_{\text{о}} - 2000$ грн.

$K_{\Pi} = 10000 + 5000 + 7000 + 2000 = 24000$ грн.

Укрупнений розрахунок витрат на розробку ІС можна виконати за формулою:

$$K_{\Pi} = \Phi_{\text{з/п}} [(1 + \beta_{\text{д}}) (1 + \beta_{\text{с}}) + \beta_{\text{н}} + \beta_{\text{пр}}] + t_{\text{ЕВМ}} C_{\text{м-ч}}, \quad (4.1.2)$$

де $\Phi_{\text{з/п}}$ - фонд основної заробітної плати розробників та інших виконавців робіт;

$\beta_{\text{д}}$ - коефіцієнт додаткової зарплати, можна приймати 0,10 ... 0,15;

$\beta_{\text{с}}$ - коефіцієнт відрахувань на соціальні потреби від основної та додаткової заробітної плати, дорівнює 0,26;

$\beta_{\text{н}}$ - коефіцієнт накладних витрат організації, що розробляє проект, можна приймати 0,6 ... 0,8;

$\beta_{\text{пр}}$ - коефіцієнт інших витрат, приймати 0,1 ... 0,2;

$t_{\text{ЕВМ}}$ - машинний час, витрачений для налагодження програмного забезпечення ІС;

$C_{\text{м-ч}}$ - вартість машино-години роботи ЕОМ.

Розрахунок

$$\Phi_{з/п} = 5000 \text{ грн}$$

$$\beta_d = 1.1$$

$$\beta_c = 0.26$$

$$\beta_n = 0.6$$

$$\beta_{пр} = 0.1$$

$$t_{ЕВМ} = 10 \text{ ч}$$

$$C_{м-ч} = 10 \text{ грн / ч}$$

$$K_{п} = 5000 * (1+1.1) * (1+0.26) + 0.6 + 0.1 + 10 * 10 = 20400 \text{ грн}$$

Укрупнений розрахунок фонду основної заробітної плати виконавців робіт по розробці ІС здійснюється за формулою:

$$\Phi_{з/п} = \sum_{ij} t_{ij} C \quad (4.1.3)$$

де $\sum_{ij} t_{ij}$ - сумарна трудомісткість робіт по розробці ІС, люд.г (люд.-дн.),

розрахунок проводиться, використовуючи дані про трудомісткість робіт;

C - тарифна ставка годинна (денна) розробників та інших виконавців робіт, грн.

Тарифна ставка розробника приймається відповідно до окладами організації, де виконується проект. Якщо розробка ведеться в організації або установі, яке фінансується з держбюджету, то використовується єдина 18-ти розрядна сітка.

Час, витрачений на налагодження програмного забезпечення на ПК $t_{ЕВМ}$, встановлюється експертним шляхом або за фактичними витратами машинного часу. Вартість машино-години роботи ПК або комплексу засобів автоматизації $C_{м-ч}$ береться в бухгалтерії тієї організації, де виконується дипломний проект. При відсутності цих даних необхідно виконати розрахунок собівартості машино-години роботи ІС з урахуванням конкретних умов. Собівартість машино-години роботи ПК або ІС визначається за формулою:

$$C_{м-г} = \frac{З_{п} + А + З + З_{р} + З_{н} + З_{м}}{\Phi}, \quad (4.1.4)$$

де $З_{п}$ - витрати на заробітну плату обслуговуючого персоналу з урахуванням всіх відрахувань;

$А$ - річна сума амортизації;

$З_{е}$ - витрати на силову електроенергію;

$З_{р}$ - витрати на ремонт і обслуговування устаткування в рік;

$З_{м}$ - витрати на матеріали в рік;

$З_{н}$ - накладні витрати;

$\Phi_{д}$ - дійсний річний фонд часу роботи ІС

Розрахунок

$А$ – 20000 грн

$З_{е}$ – 1000 грн

$З_{р}$ – 2500 грн

$З_{м}$ – 1500 грн

$З_{н}$ – 1000 грн

$\Phi_{д}$ – 2016

$$C_{м-г} = (20000 + 1000 + 2500 + 1500 + 1000) / 2016 = 12.89 \text{ грн/год}$$

Розрахунок витрат на заробітну плату обслуговуючого персоналу здійснюється за формулою:

$$З_{п} = \sum_{i=1}^n l_i * 12 * k_{д} * k_{с.н.} \quad (4.1.5)$$

де n - кількість працівників;

l_i - місячний оклад працівника;

$k_{д}$ - коефіцієнт, що враховує додаткову заробітну плату ($k_{д}$ приймається від 1,1 до 1.2);

$k_{с.н.}$ - коефіцієнт, що враховує відрахування на соціальні потреби приймається рівним 1,26 (відповідно до законодавства України відрахування

на соціальні потреби становлять 26% від основної та додаткової заробітної плати).

n - 10;

l_i - 4200;

k_d - 1.2

$k_{с.н.}$ - 1.26

$$З_{\pi} = 4200 * 12 * 1.2 * 1.26 * 10 = 762\,048 \text{ грн.}$$

Річні амортизаційні відрахування по ІС вважаються за формулою:

$$A = \frac{C_{КСА} * H_a}{100} \quad (4.1.6)$$

Де - вартість ПК та іншого обладнання, що входить в ІС, використовуваного при налагодженні ІС; $C_{КСА}$

H_a - норма амортизації, (25%). Витрати на електроенергію в рік $З_e$ визначаються наступним чином:

$$З_e = W_y + C_e + T_B \quad (4.1.7)$$

де W_y - установлена потужність, кВт;

C_e - вартість силової електроенергії, р / кВт;

T_B - час, протягом року, коли ІС споживає електроенергію, ч.

Витрати на поточні ремонти $З_p$ і на матеріали $З_m$ в рік беруться за даними бухгалтерії. При укрупненому розрахунку їх сума може бути прийнята від 4,5 до 10% від вартості ІС.

В накладні витрати включаються витрати на оплату праці АУП, витрати на амортизацію і зміст площ, витрати на опалення, освітлення та інші.

Річний фонд часу Φ_d встановлюється, виходячи з номінального фонду часу і часу профілактики обладнання і ремонтів:

$$\Phi_d = S \cdot h \cdot D - T_{пр} \quad (4.1.8)$$

де S – тривалість зміни;

h – кількість змін;

D – число робочих днів у році;

$T_{\text{пр}}$ – час ремонтів і профілактики обладнання в рік

$S = 8$

$h = 2$

$D = 252$

$T_{\text{пр}} = 50$

$\Phi_d = 8 * 2 * 252 - 50 = 3\,982$

4.2 Розрахунок-прогноз мінімальної ціни розробки ІС

Мінімальна ціна розробки ІС Z_{min} складається з повних витрат на розробку $K_{\text{п}}$ і мінімально необхідної суми прибутку Π_{min} , розмір якої дозволяв би на мінімальному рівні здійснити самофінансування організації-розробника після всіх обов'язкових платежів і виплати податків.

$$Z_{\text{min}} = K_{\text{п}} + \Pi_{\text{min}}. \quad (4.2.1)$$

Сума прибутку Π_{min} розраховується виходячи з планованого мінімального рівня рентабельності витрат організації-розробника.

$$\Pi_{\text{min}} = K_{\text{п}} (R_{\text{min}} / 100) \quad (4.2.2)$$

де R_{min} - мінімальний рівень рентабельності, (10 - 20%).
Автоматизована система може розроблятися на основі договору з замовником, або для продажу на ринку. У першому випадку ціна продажу встановлюється за згодою зацікавлених сторін (розробника і замовника) і називається договірною ціною $Z_{\text{дог}}$. Договірна ціна, як правило, встановлюється між мінімальною Z_{min} і максимальною ціною Z_{max} , тобто

$$Z_{\text{min}} \leq Z_{\text{дог}} \leq Z_{\text{max}} \quad (4.2.3)$$

де максимальна ціна - це така ціна ІС, яка визначається величиною економічного ефекту, який отримає замовник від впровадження ІС.

Якщо ІС розробляється для задоволення потреб різних організацій і планується продаж її на ринку, то студентдипломник повинен зробити прогноз цін, витрат, попиту і можливостей реалізації. З цією метою необхідно: виявити потенційних споживачів і їх вимоги до системи (технічні, інформаційні, зручність в роботі і ін.); встановити конкурентів, їх сильні і слабкі сторони, ціни продажів аналогічних систем; вибрати підхід до визначення ціни продажу; визначити можливі витрати на рекламу, супровід, доопрацювання тощо.

4.3 Оцінка беззбитковості і розрахунок доцільного обсягу продажів

Для аналізу доцільності витрат на розробку слід застосувати метод аналізу беззбитковості проекту і розрахувати доцільний обсяг продажів. Метод аналізу полягає в тому, щоб виявити точку беззбитковості (ТБ). Під нею мається на увазі точка кривої (прямий), яка б показала зростання обсягу продажів в системі двох координатних осей, в якій доходи від продажу рівні сумарних витрат (прибуток розробника дорівнює 0). Для аналізу беззбитковості необхідні наступні дані:

витрати (одноразові) на розробку ІС K_{π} ; витрати на рекламу, супровід на одну угоду S_1 ; ціна продажу Z

Обсяг продажів у вартісному вираженні Q є функцією від кількості продажів N і розраховується за формулою:

$$Q(N) = Z \cdot N. \quad (4.3.1)$$

Сумарні витрати на розробку і реалізацію проекту визначаються за формулою:

$$S(N) = K_{\pi} + S_1 \cdot N. \quad (4.3.2)$$

Точка беззбитковості ТБ знаходиться зі співвідношення:

$$Q(N_{TB}) = S(N_{TB}), \text{ або } Z \cdot N_{TB} = K_{\pi} + S_1 \cdot N_{TB} \quad (4.3.3)$$

$$N_{TB} = \frac{K_{\pi}}{Z - S_1}. \quad (4.3.4)$$

Точка беззбитковості служить розробнику хорошим орієнтиром в оцінці ризику витрат на розробку. Графік беззбитковості наведено на рисунку 4.1.

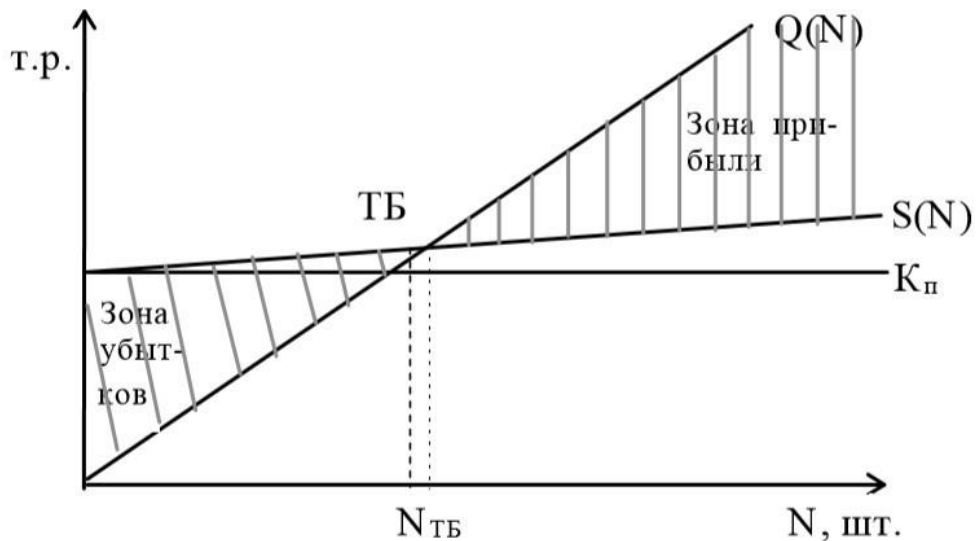


Рисунок 4.1 - Графік беззбитковості

Витрати на розробку вважаються ефективними, якщо доходи покривають усі витрати на розробку, продаж ІС і буде отримана мінімально необхідна сума прибутку Π_{min} . Тому розраховується доцільний обсяг продажів $N_{ц}$ зі співвідношення:

$$Z \cdot N_{ц} \geq (K_{\pi} + S_1 \cdot N_{ц}) + \Pi_{min} \quad (4.3.5)$$

$$N_{ц} \geq \frac{K_{\pi} + \Pi_{min}}{Z - S_1} \quad (4.3.6)$$

Графік для визначення доцільного обсягу продажів наведено на рисунку. 4.2.

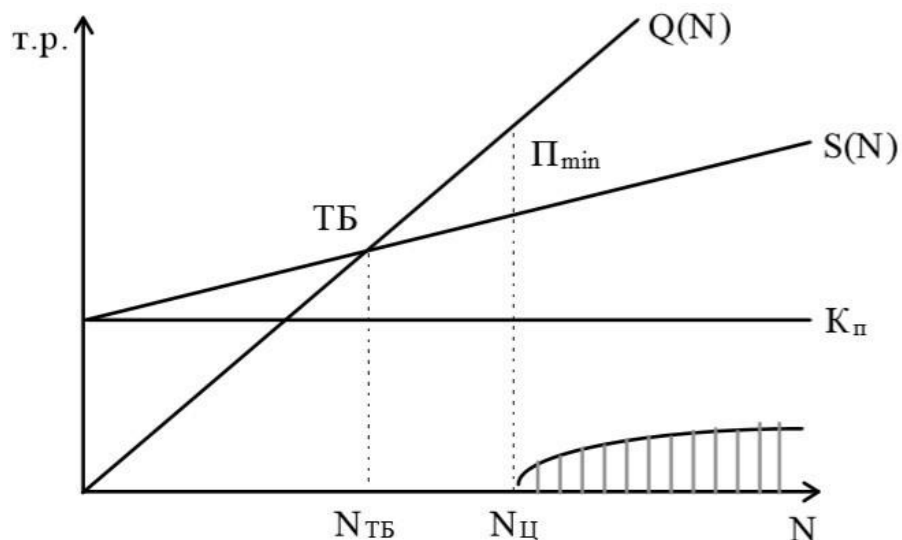


Рисунок 4.2 - Графік доцільного обсягу продажів

Рекомендується провести аналіз беззбитковості, розрахунок доцільного обсягу продажів і побудувати графіки для різних рівнів ціни продажу.

4.4 Розрахунок одноразових витрат на впровадження ІС

Одноразові витрати на впровадження ІС включають витрати на придбання проекту ІС або витрати на розробку проекту, якщо розробка ведеться фахівцями тієї ж організації (підприємства), де впроваджується ІС, капітальні витрати на комплекс технічних засобів (КТЗ), а також витрати на установку КТЗ, його монтаж і налагодження. Слід зазначити, що при розрахунку ефективності конкретної ІС величина капітальних витрат K_i визначається пропорційно частці часу використання засобів автоматизації в даній ІС δ_i . Це пояснюється тим, що один і той же комплекс засобів автоматизації може використовуватися в роботі декількох ІС. Тому одноразові витрати на впровадження i -ї системи K_i визначаються за формулою:

$$K_i = Z_{\text{пр}}^i + K_I \cdot \delta_i \quad (4.4.1)$$

де $Z_{\text{пр}}^i$ - витрати на придбання проекту ІС з урахуванням витрат на адаптацію і навчання персоналу (або все виробничі витрати, якщо ІС розробляється фахівцями даного підприємства);

K_I - величина інвестиційних (капітальних) витрат;

δ_i - коефіцієнт участі ІС.

Величина інвестиційних (капітальних) витрат визначається за формулою:

$$K_I = K_{\text{КТС}} + K_{\text{м}} + K_{\text{інв}} + K_{\text{зд}} + K_{\text{ос}} + K_{\text{тр}} + K_{\text{соп}} - K_{\text{вис}} \quad (4.4.2)$$

де $K_{\text{КТС}}$ - кошторисна вартість КТС;

$K_{\text{м}}$, $K_{\text{інв}}$, $K_{\text{зд}}$ - витрати на установку, монтаж і запуск КТС в роботу, на виробничий інвентар, на будівництво і реконструкцію будівель для розміщення КТС;

$K_{\text{ос}}$ - сума оборотних коштів;

$K_{\text{тр}}$ - транспортно-заготівельні витрати;

$K_{\text{соп}}$ - кошторисна вартість системи стандартного забезпечення застосування КТС;

$K_{\text{вис}}$ - сума вивільнених коштів в результаті введення в дію ІС, р.

Коефіцієнт участі ІС δ_i знаходиться за формулою:

$$\delta_i = \frac{t_i}{\Phi} \quad (4.4.3)$$

де t_i - час використання ІС при функціонуванні даної ІС протягом року;

Φ - дійсний фонд часу роботи ІС в рік.

Якщо комплекс засобів автоматизації використовується тільки для однієї системи, то коефіцієнт участі δ дорівнює одиниці.

Одноразові витрати можуть бути різночасними. В цьому випадку потрібно вказати терміни здійснення цих витрат, з тим, щоб їх привести до одного періоду за допомогою коефіцієнтів дисконтування.

4.5 Розрахунок поточних витрат на функціонування ІС

Розрахунок річних поточних витрат в ІС $Z_{\text{тек}}$ може виконуватися двома методами. Перший метод передбачає визначення поточних витрат, за допомогою основних складових:

$$Z_{\text{тек}} = Z_{\text{ІС}} + Z_{\text{зп}} \quad (4.5.1)$$

де $Z_{\text{ІС}}$ - річні поточні витрати на експлуатацію ІС;

$Z_{\text{зп}}$ - річні витрати на заробітну плату фахівців в умовах функціонування ІС з нарахуваннями.

Витрати $Z_{\text{ІС}}$ визначаються за формулою:

$$Z_{\text{ІС}} = Z_{\text{КТС}} \cdot \delta_i + Z_{\text{соп}} + Z_{\text{зд}} + Z_{\text{екТС}} \quad (4.5.2)$$

де $Z_{\text{КТС}}$ - річні витрати на експлуатацію КТС без урахування заробітної плати персоналу;

$Z_{\text{соп}}$ - річні витрати на підтримку і актуалізацію системи забезпечення застосування КТС (зберігання, оновлення, контроль даних, програм та інші операції);

$Z_{\text{зд}}$ - річні витрати на утримання і ремонт виробничих приміщень;

$Z_{\text{екТС}}$ - річні витрати на заробітну плату працівників групи експлуатації ІС з нарахуваннями.

Другий метод дозволяє розраховувати поточні витрати на функціонування ІС шляхом визначення сумарних витрат, що викликаються рішенням комплексу завдань (процедур) ІС і загальносистемних витрат. При цьому річні поточні витрати в ІС $Z_{\text{тек}}$ визначаються за формулою:

(4.5.3)

де $Z_{\text{сист}}$ - загальносистемні витрати за рік;
 $Z_{\text{тек}} = \sum_{i=1}^n Z_i + Z_{\text{сист}}$

Z_i - витрати, викликані рішенням i -й завдання

ІС; n - число завдань, що вирішуються

протягом року. Витрати Z_i можуть бути

розраховані за формулою:

$$Z_i = Z_{\text{ип}} + Z_{\text{м.в.}}^i \quad (4.5.4)$$

де $Z_{\text{ип}}$ - річні витрати на заробітну плату спеціалістів, які вирішують це завдання з урахуванням всіх нарахувань в умовах ІС;

$Z_{\text{м.в.}}^i$ - вартість роботи комплексу засобів автоматизації по цьому завданню.

Витрати, пов'язані з роботою ІС по цьому завданню $Z_{\text{м.в.}}^i$ можуть бути розраховані за формулою:

$$Z_{\text{м.в.}}^i = C_{\text{м-год}}^{\text{ІС}} \cdot t_i^{\text{ІС}} \quad (4.5.5)$$

де $C_{\text{м-год}}^{\text{ІС}}$ - собівартість години роботи ІС;

$t_i^{\text{ІС}}$ - час виконання завдання з використанням ІС в рік.

Вибір одного з методів обумовлюється наявністю вихідних даних для розрахунків, а також стадією розробки або функціонування ІС. На етапі розробки ІС рекомендується використовувати другий спосіб розрахунків, а в період функціонування ІС - перший.

РОЗДІЛ 5. ОХОРОНА ПРАЦІ

У даному розділі проводиться аналіз середовища, в якому проводилося дослідження програмного продукту на основі санітарних норм України. Проводилося дослідження роботи з розробки ІС для спрощення масивів маршрутних точок, напряму пов'язані з роботою на комп'ютері. Робота персонального комп'ютеру та тестувальних пристроїв (смартфон, планшет) впливають на фізичні і хімічні чинники середовища: електромагнітні випромінювання, статична електрика, температура і вологість повітря, вміст кисню і озону.

Охорона праці - це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних та лікувально-профілактичних заходів і засобів, спрямованих на збереження здоров'я та працездатності людини в процесі праці.

Законодавчими актами, що визначають основні положення про охорону праці, є загальні закони України, а також спеціальні законодавчі акти. До загальних законів, що визначають основні положення про охорону праці належать: Конституція України, Закони України «Про охорону праці», «Про охорону здоров'я», «Про пожежну безпеку», «Про використання ядерної енергії та радіаційний захист», «Про забезпечення санітарного та епідемічного благополуччя населення», «Про загальнообов'язкове державне соціальне страхування від нещасного випадку на виробництві та професійного захворювання, які спричинили втрату працездатності», Кодекс законів про працю України (КЗпП). Спеціальними законодавчими актами в галузі охорони праці є Державні нормативні акти про охорону праці, Державні стандарти Системи стандартів безпеки праці, Будівельні норми та правила, Санітарні норми, Правила технічної експлуатації електроустановок споживачів та інші нормативні документи.

5.1 Аналіз умов праці в приміщенні

Дослідження, що проводилося в даній дипломній роботі, проводилося в приміщенні, що має наступні параметри. Приміщення має довжину 3.5 м, ширину 2.5 м, висоту стелі 2,7 м. Площа – 8.75 м², об'єм – 23.625 м³. Згідно з нормами площа робочого приміщення має бути не менше 6 м², а його об'єм не менше 15 м³. Можна зробити висновок, що отримані показники відповідають існуючим нормам та вимогам.

Розглянемо тепер відповідність характеристик робочого місця нормативним. Для цього зведемо основні вимоги до організації робочого місця і відповідні фактичні значення для робочого місця, за яким виконується робота, у табл. 5.1

Таблиця 5.2 - Характеристики робочого місця

Найменування параметра	Значення	
	Фактичне	Нормативне
Висота робочої поверхні, мм	700	680 - 800
Висота простору для ніг, мм	700	не менше 600
Ширина простору для ніг, мм	700	не менше 500
Глибина простору для ніг, мм	700	не менше 650
Висота поверхні сидіння, мм	500	400 - 500
Ширина сидіння, мм	400	не менше 400
Глибина сидіння, мм	400	не менше 400
Висота поверхні спинки, мм	500	не менше 300
Ширина опорної поверхні спинки, мм	400	не менше 380
Радіус кривини спинки в горизонтальній площині, мм	400	400
Відстань від очей до екрану дисплея, мм	700	700 - 800

Крісло є підйомно-поворотним, має підлокітники і можливість регулювання за висотою і кутом нахилу спинки. Екран монітору знаходиться на відстані 0.7 м. Таким чином, за всіма параметрами робоче місце відповідає нормативним вимогам.

5.2 Розрахунок освітлення

Приміщення розташоване в 4 поясі світлового клімату. Довжина приміщення: 3.5м, глибина: 2.5м. Розрахункова точка А знаходиться від зовнішньої стіни на відстані: 2 м. Відстань від робочої поверхні до верха вікон: 1.5 м.

Нормоване значення коефіцієнту бокового природного освітлення для III розряду зорової роботи III світлового поясу $e_n^{III} = 1.5\%$. У зв'язку з тим, що приміщення знаходиться в IV світловому поясі, КПО буде визначено за формулою:

$$e_n^{IV} = e_n^{III} m c \quad (5.1)$$

де $m = 0,9$ —коефіцієнт світлового клімату;

$c = 0,7$ - коефіцієнт сонячності клімату.

$$e_n^{IV} = 1.5 \times 0.9 \times 0.7 = 0.945\% \quad (5.2)$$

Загальний коефіцієнт світло пропускання визначається за формулою:

$$\tau_0 = \tau_1 \tau_2 \tau_3 \tau_4 \tau_5 \quad (5.3)$$

де $\tau_1 = 0,8$ - засклення подвійне;

$\tau_2 = 0,65$ - Плетення подвійне дерев'яне;

$\tau_3 = 1$ - Несущі конструкції відсутні;

$\tau_4 = 1$ - Сонце захисні конструкції відсутні

$\tau_5 = 1$ - Захисних сіток немає

$$\tau_0 = 0.8 \times 0.65 \times 1 \times 1 \times 1 = 0,52 \quad (5.4)$$

Площа світлових прорізів при боковому освітленні визначається за формулою:

$$S_{\bar{\sigma}} = \frac{S_n e_n K_z \eta_{\bar{\sigma}} K_{\bar{\sigma}}}{100 r_1 \tau_0} \quad (5.5)$$

де S_n - площа підлоги приміщення,

$$S_n = 3.5 \times 2.5 = 8.75 \text{ м}^2;$$

$K_z = 1,2$ - коефіцієнт запасу;

$\eta_{\bar{\sigma}} = 13$ - світлова характеристика вікон;

$K_{\bar{\sigma}} = 1$ - коефіцієнт, що враховує затінення вікон будинками, що стоять навпроти;

$r_1 = 2,4$ - коефіцієнт, що враховує підвищення КПО при боковому освітленні завдяки світлу, відбитому від поверхонь приміщення і підстильного шару, що прилягає до будинку.

$$S_{\bar{\sigma}} = \frac{104 \times 1,26 \times 1,2 \times 13 \times 1}{100 \times 2,4 \times 0,52} = 16,38 \text{ м}^2 \quad (5.6)$$

Загальна ширина вікон знаходиться за допомогою наступної формули:

$$b_{\bar{\sigma}} = \frac{S_{\bar{\sigma}}}{h_{\bar{\sigma}}} \quad (5.7)$$

де $h_{\bar{\sigma}} = 1.5$ м - висота вікна

$$b_{\bar{\sigma}} = \frac{8,24}{1,5} = 5,49 \text{ м} \quad (5.8)$$

Фактична освітленість на робочому місці становить 363.05 лк, що відповідає вимогам до III розряду зорової роботи. Можна зробити висновок, що для роботи з дисплеєм цілком достатньо існуючих джерел світла.

5.3 Параметри мікроклімату

Суттєвий вплив на стан організму працівника, його працездатність здійснює мікроклімат у приміщенні, що визначається діючою на організм

людини сукупністю температури, вологості, руху повітря та теплового випромінювання нагрітих поверхонь.

Зазначену роботу можна віднести до категорії легка 1а. Джерелами тепла в цьому приміщенні є люди, електроустаткування, освітлювальні прилади в темний час доби і система опалювання зимою. Оператором виділяється до 120Ккал теплової енергії за годину. Оптимальні та фактичні значення параметрів мікроклімату приведені в таблиці 5.3.

Таблиця 5.3 – Значення мікроклімату

Період року	Параметр	Оптимальний	Фактичний
Теплий	Температура	23 – 25 °С	23 °С
	Вологість	40 – 60 %	40 %
	Швидкість повітря	≤ 0.1 м/с	
Холодний	Температура	22 – 24 °С	22 °С
	Вологість	40 – 60 %	50 %
	Швидкість повітря	≤ 0.1 м/с	

5.4 Шум і вібрація

Джерелом шуму в приміщенні є комп'ютер. Кулери комп'ютера є сучасними і мають низький рівень шуму за рахунок нового підшипника зі самостабілізуючимся тиском рідини (SSO2). Згідно технічній документації шум обумовлений кулером в блоці живлення складає 7-12 дБ, кулером процесора – 7-12 дБ, загальний, - 20 дБ.

Враховуючи незначний рівень шуму від персонального комп'ютера та незначний рівень фонового шуму від іншого устаткування - сумарний рівень шумового забруднення приміщення не перевищує максимально допустимий рівень коригованої звукової потужності і складає не більше 50 дБ. допустимих значень.

5.5 Випромінювання

У приміщенні відсутні інфрачервоні, ультрафіолетові та електромагнітні випромінювання, бо усі монітори ПК вироблені на основі OLED матриці, підсвітка якої здійснюється за рахунок органічних світлодіодів, що не має сильного електромагнітного випромінювання і сертифіковані в Україні.

5.6 Вимоги електробезпеки

Технічні заходи із запобігання електротравм від контакту з нормально струмовідними елементами електроустаткування

- електромережа в приміщенні розведена в спеціальних каналах стін і підлоги.

- величина напруги мережі 380х220В (міжфазна лінійна і фазна);

- всі нормально струмовідні елементів (в першу чергу електричні дроти) вкриті ізоляційними матеріалами;

- в джерелі безперебійного живлення персонального комп'ютера використовується механічне захисне блокування, що забезпечує вимикання напруги при його відкриванні;

Можна зробити висновок, що дане приміщення задовольняє нормам електробезпеки для встановлення ПК.

5.7 Оцінка пожежної безпеки приміщення

У приміщенні, що розглядається, можуть горіти: вироби з дерева, пластмас, тканини і паперу, ламінат. Горючі рідини, пил та волокна у приміщенні не використовуються і не виділяються.

Ймовірними причинами виникнення пожежі можуть бути несправна робота електрообладнання (кабелів, розеток), короткі замикання внаслідок

виходу з ладу чи експлуатації несправного електроустаткування (ПЕОМ, периферійних пристроїв), порушення правил протипожежної безпеки тощо.

Експлуатація ліній електромережі практично повністю унеможлиблює виникнення електричного джерела загоряння в наслідок короткого замикання та перевантаження проводів, а автоматичні вимикачі миттєво відключають ділянку мережі у разі перевантаження чи замикання. Застосовуються дроти з важкогорючою і негорючою ізоляцією. Блок живлення комп'ютеру оснащений автоматичним вимикачем у разі перевантаження чи стрибка напруги у мережі.

Комплекс заходів для своєчасного попередження пожеж у приміщенні та оперативного реагування у разі їх виникнення:

- заборона використання відкритого вогню у приміщенні;
- автомати на кожній ланці мережі для запобігання перевантаження та замикання;
- обов'язковий інструктаж персоналу з питань охорони праці,
- зокрема, правила пожежної безпеки у приміщеннях з ЕОМ;
- наявність системи автоматичної пожежної сигналізації з димовими пожежними оповісниками;
- ступінь вогнестійкості будівлі, у якій розташовано приміщення – II;
- наявність шляхів евакуації при виникненні пожежі;
- призначення особи, відповідальної за пожежну безпеку;
- розміщення схеми евакуації людей при пожежі і ознайомлення з нею персоналу.

Приміщення має один вихід, оскільки в ньому працює менше 25 чоловік. Будинок має два виходи – головний і запасних. Коридор між приміщеннями має два виходи на різні сходи, одні з яких ведуть до головного виходу, а другі - до спеціального евакуаційного виходу. Для гасіння пожежі кожна кімната обладнана ручними вуглекислотними вогнегасниками ВВК-1,4.

Можна зробити висновок, що шляхи евакуації з приміщення повністю відповідають нормам.

5.8 Висновки

Аналіз умов праці в розглянутому робочому приміщенні показав, що умови праці з ПЕОМ відповідають вимогам, оскільки площа та об'єм не менше нормативних значень, рівні шуму, вібрації і загазованості не перевищують нормативних обмежень.

Для підтримання параметрів мікроклімату в приміщенні встановлено радіатор центральної водяної системи опалення, що складається з 7 секцій.

Ергономіка робочого місця і режим зорової роботи задовольняють вимогам і сприяють зниженню втоми.

РОЗДІЛ 6. ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА ПРИ ОПТИМІЗАЦІЇ МАРШРУТІВ ШВИДКОЇ ДОПОМОГИ

Вступ

Оптимізація маршрутів швидкої допомоги є критично важливою для своєчасного надання медичної допомоги. Однак, при розробці алгоритмів оптимізації часто не враховується вплив транспортних засобів на навколишнє середовище. Цей розділ присвячений аналізу екологічних аспектів, пов'язаних з роботою служби швидкої допомоги, та розробці стратегій для мінімізації негативного впливу на довкілля.

Вплив роботи швидкої допомоги на навколишнє середовище

Викиди шкідливих речовин- автомобілі швидкої допомоги, як і будь-які інші транспортні засоби, є джерелом викидів парникових газів (вуглекислого газу, оксидів азоту та ін.), які сприяють зміні клімату та забрудненню повітря.

Шум - сирени та двигуни швидких викликають значний рівень шуму, що негативно впливає на здоров'я людей, особливо в житлових районах.

Споживання палива-часті виїзди та тривалі пробіги призводять до високого споживання палива, що збільшує залежність від викопних ресурсів.

Стратегії мінімізації негативного впливу

Оптимізація маршрутів з урахуванням екологічних факторів:

Вибір найкоротших маршрутів- зменшення відстані пробігу призводить до зниження споживання палива та викидів.

Урахування пробок- об'їзд заторів дозволяє скоротити час у дорозі та зменшити викиди, пов'язані з простоюми.

Вибір маршрутів з меншою кількістю зупинок- зменшення кількості розгонів і гальмувань також сприяє економії палива.

Використання екологічно чистих видів транспорту.

Електромобілі - перехід на електромобілі дозволить повністю відмовитися від викидів шкідливих речовин.

Гібридні автомобілі - комбінація електричного та бензинового двигунів дозволяє знизити споживання палива.

Велосипеди та інші екологічно чисті засоби пересування. Для невеликих відстаней та в умовах міста можливе використання велосипедів або інших екологічно чистих засобів пересування.

Оптимізація роботи диспетчерської служби.

Ефективне планування виїздів - зменшення кількості марних виїздів дозволить знизити загальний пробіг автомобілів.

Використання інтелектуальних систем диспетчеризації - сучасні системи дозволяють оптимізувати маршрути в реальному часі з урахуванням різних факторів, включаючи дорожню обстановку та наявність вільних автомобілів.

Співпраця з міською інфраструктурою - розширення мережі зарядних станцій для електромобілів. Забезпечення доступності зарядних станцій дозволить стимулювати перехід на електромобілі.

Створення виділених смуг для громадського транспорту та швидкої допомоги. Це дозволить скоротити час у дорозі та зменшити викиди.

Підвищення обізнаності персоналу.

Проведення навчань, персонал служби швидкої допомоги повинен бути ознайомлений з екологічними аспектами своєї роботи та способами зменшення негативного впливу на довкілля.

Стимулювання екологічно відповідальної поведінки. Впровадження системи заохочень за економне використання палива та дотримання екологічних стандартів.

Висновок

Оптимізація маршрутів швидкої допомоги з урахуванням екологічних факторів є важливим кроком до створення більш стійкої та екологічно чистої системи охорони здоров'я. Впровадження зазначених стратегій дозволить не тільки зменшити негативний вплив на довкілля, але й підвищити ефективність роботи служби швидкої допомоги та покращити якість життя населення.

ВИСНОВКИ

Вибухова популяризація застосувань географічних інформаційних систем зумовлює необхідність створення різноманітних програмних рішень для допомоги розробникам в розробці, а користувачам в автоматизації багатьох прикладних задач у цій сфері.

В рамках даної дипломної роботи був розроблений та реалізований ряд алгоритмів для обробки географічних даних і створено географічну інформаційну систему - комплексне рішення, призначене для спрощення і автоматизації створення, зміни, конвертації та аналізу географічних даних.

Серед розроблених алгоритмів варто виділити вдосконалений алгоритм спрощення масивів маршрутних точок, а також винайдений алгоритм пошуку спільних ділянок декількох маршрутів. Використання цих алгоритмів в географічних інформаційних системах дозволить суттєво підвищити ефективність автоматизації такого роду завдань.

Інформаційна система була реалізована у вигляді підсистеми (бібліотеки підпрограм Node.js). В сучасних реаліях зазначена технологія дозволяє інтеграцію підсистеми в системи неймовірно широкого спектру - від серверних до мобільних застосунків. Крім того, завдяки ретельно сформованим вимогам до системи, її функціонал включає найбільш затребувані та універсальні функції, необхідні для найбільш різноманітних програмних рішень.

Завдяки цьому результати даної роботи можуть бути використані як для забезпечення функціональності географічно-орієнтованого веб-сайту (веб-додатка), так і в якості елементів більш масштабних або складних проектів.

Крім того, був розроблений базовий користувацький інтерфейс для системи, що суттєво розширює коло потенційних користувачів та завдань, спрощує знайомство з функціоналом та демонструє універсальність застосування знайдених рішень.

Новизна методів розробки також полягала в комплексності рішення, адже реалізовані алгоритми значною мірою спираються на функціонал один одного. Це дозволило розробити ІС більш ефективно, а кінцевий продукт зробити компактнішим, що вигідно вирізняє її зі вже існуючих рішень.

У подальшому можливо розширення функціоналу системи пов'язаного з засобами аналізу даних, а також збільшення підтримуваних для обробки типів даних, наприклад, таких, що застосовуються в сфері візуалізації географічної інформації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

- [1] Самойленко В. (2010) Географічні інформаційні системи та технології. Київ: Ніка-Центр.
- [2] Penttinen, J., Jyrki T. (2015). The telecommunications handbook. Chichester: John Wiley.
- [3] Reynisson, J. Á. (2015). Performance of mobile GIS in conjunction with internet bandwidth in rural areas (Dissertation). Retrieved from <http://urn.kb.se/resolve?urn=urn:nbn:se:hig:diva-20021>.
- [4] Turf.js Api documentation. (2019). Retrieved September 27, 2019, from <https://turfjs.org/docs/>
- [12] Longhorn, R. and Blakemore, M. (2018). Geographic information. Boca Raton: CRC Press.
- [16] Mantyla, D. (2015). Functional programming in JavaScript. Birmingham, UK: Packt.
- [17] GPX 1.1 Schema Documentation. (2019). Retrieved August 15, 2019, from <https://www.topografix.com/gpx/1/1/>
- [18] OGC KML 2.2 Standard. (2010). Retrieved from <https://www.opengeospatial.org/standards/kml>
- [19] Heitkötter, Henning & Heierhoff, Sebastian & Majchrzak, Tim A.. (2023). Evaluating Cross-Platform Development Approaches for Mobile Applications. Lecture Notes in Business Information Processing. 140. 120-138.
- [22] D. Crockford (2016). "JSON in JavaScript". Retrieved May 15, 2019, from <https://web.archive.org/web/20160710230817/http://www.json.org/js.html>
- [23] Møller, Margrethe. (2023). Usability Testing of User Manuals. Communication & Language at Work. 2. 51. 10.7146/claw.v1i2.7892.

Технічне завдання на розробку інформаційної системи для редагування географічних даних.

Вступ

Повне найменування програмного забезпечення:

«Розробка інформаційної системи для редагування географічних даних».

Коротка характеристика області застосування:

Інформаційна система, реалізована у формі бібліотеки підпрограм Node.js (JavaScript) планується використовуватись для спрощення розробки ПЗ у всіх сферах застосування, які потребують опрацювання географічних даних.

Крім того, завдяки наявності користувацького інтерфейсу командного рядка зможе використовувати систему як окремий програмний продукт.

Робоча назва проекту: “ІС для редагування геоданих”.

1 Підстави для розробки

Підставою для розробки програмного забезпечення є наказ ректора Національного Університету Кораблебудування імені адмірала Макарова, наказ від 08.10.2025

2 Призначення розробки

2.1 Функціональне призначення

Інформаційна система повинна автоматизувати процес створення та редагування географічних даних.

2.2 Експлуатаційне призначення

Використання даної системи дозволить користувачу здійснювати різноманітні операції для редагування, створення та базового аналізу таких географічних даних як маршрути(треки), точки, географічні зони. Система може використовуватись безпосередньо за допомогою інтерфейсу командного рядка або в якості бібліотеки підпрограм.

3 Вимоги до програмного забезпечення

3.1 Вимоги до функціональних характеристик

3.1.1 Вимоги до складу виконуваних функцій

Програмне забезпечення повинне виконувати наступні функції:

- 1) Зчитувати інформацію з GPX, KML та GeoJSON файлів.
- 2) Експортувати інформацію в GPX, KML та GeoJSON файли.
- 3) Здійснювати вільну конвертацію між цими форматами.
- 4) Створювати географічні дані (маршрути, точки та зони).
- 5) Змінювати географічні дані.
- 6) Визначати відстань, серединну точку та курс між двома точками.
- 7) Визначати довжину трека чи маршрута.
- 8) Визначати набір висоти на маршруті чи треку.
- 9) Поєднувати треки та маршрути.
- 10) Скорочувати треки та маршрути до заданих користувачем меж.
- 11) Визначати спільні ділянки треків та маршрутів.
- 12) Спрощувати треки та маршрути (зменшуючи кількість маршрутних точок в них).
- 13) Визначати географічну зону, яка би повністю вміщувала трек, маршрут або набір точок.

Визначати географічну зону, яка би вміщувала декілька інших зон.

3.1.2 Перевіряти валідність GPX, KML та GeoJSON файлів Вимоги до організації вхідних та вихідних даних

Вхідною інформацією програми є треки, маршрути або точки представленні у вигляді файлів у наступних форматах:

- 1) GPX (XML) - файл, що є дійсним згідно з XML схемою «GPS Exchange Format» версії 1.1;
- 2) KML (Keyhole Markup Language) - XML файл дійсний за стандартом OGC;

- 3) GeoJSON - файл JavaScript Object Notation, що відповідає стандарту RFC 7946.

Також вхідними даними є параметри для маніпуляції даними, що визначаються користувачем за допомогою користувацького інтерфейсу або програмного інтерфейсу модулів.

Вихідними даними є:

- 1) файли в описаних вище форматах (GPX, KML, GeoJSON);
- 2) графічна інформація, що може бути відображена на растрових мапах технології за допомогою технології Leaflet.js або аналогів;
- 3) інформація, яку користувач (програма-клієнт) може отримати в результаті звернення до функцій (підпрограм).

3.1.3 Вимоги до відображення дизайну та інтерфейсу

Інтерфейс командного рядка має базуватися на усталених конвенціях для таких інтерфейсів (наприклад, POSIX).

Програмні інтерфейси бібліотеки підпрограм мають бути задокументовані згідно стандарту JSDoc 3.

3.2 Вимоги до надійності

3.2.1 Вимоги до надійного функціонування

Система повинна нормально функціонувати. При виникненні збою в роботі апаратури, відновлення нормальної роботи програми повинне виконуватися після: перезапуску програми або основної програми; повторного виконання дій, втрачених до останнього збереження інформації на пристрої. Програма повинна забезпечувати коректну обробку виняткових ситуацій.

3.2.2 Вимоги до контролю вхідної і вихідної інформації

Система повинна забезпечувати правильне введення інформації за рахунок використання шаблонів введення, процедурного блокування введення некоректної інформації, інструментів перевірки відповідності даних стандартам, тощо.

3.3 Вимоги до умов експлуатації

Використання продукта як підсистеми (бібліотеки підпрограм) потребує базових навичок програмування в середовищі Node.js або іншого Javascript середовища. Зокрема для використання програмного інтерфейсу підпрограм необхідне вміння користуватися модулями CommonJS.

3.4 Користування продуктом за допомогою користувацького інтерфейсу не має потребувати спеціальних навичок.

Вимоги до програмно-технічного забезпечення

Система має функціонувати при виконанні наступних умов:

- наявність Node.js не нижче версії 8, або іншого середовища що дозволяє використання модулів CommonJS;
- мінімальні системні вимоги не мають перевищувати 600 мб оперативної пам'яті та 1 гГц тактової частоти процесора.
- В архівному вигляді файли початкового коду програми не повинні перевищувати об'єм в 1 Мбайт.
- Об'єм виконуваних файлів не має перевищувати 10мб.
- Вимоги до програмної документації
- Документація до програмного засобу повинна містити наступні документи:
 - технічне завдання(ДЕРЖСТАНДАРТ 19.201-78);
 - текст програми (ДЕРЖСТАНДАРТ 19.401-78);
 - програма і методика випробувань (ДЕРЖСТАНДАРТ 19.301-79);
- інструкція користувача (ДЕРЖСТАНДАРТ 19.505-79).

Програма та методика випробувань інформаційної системи для
редагування географічних даних.

1 Об'єкт випробувань

Об'єктом випробувань є програмне забезпечення «ІС для редагування географічних даних».

2 Мета випробувань

Випробування проводяться з метою виявлення неточностей і помилок, допущених при кодуванні і розробці алгоритму, а також для надання розробленому програмному продукту закінченого та готового вигляду, придатного до застосування.

3 Вимоги до системи

- зручний у використанні інтерфейс командного рядка;
- функціональний та послідовний програмний інтерфейс;
- можливість експорту та імпорту даних в підтримуваних форматах ;
- можливість створення нових даних та зміни існуючих;
- здатність здійснення аналізу окремих точок, маршрутів, зон.

4 Вимоги до програмної документації

Склад програмної документації, що пред'являється на випробуваннях програмного забезпечення:

- технічне завдання;
- специфікація програми;

5 текст програми. Засоби та порядок випробувань

Випробування проводяться при наявності комп'ютера з встановленим середовищем Node.js версії 13.0.

Програма запускається один раз для реалізації всіх методу випробувань, описаних у методах випробувань.

6 Методи випробувань

1) Функція імпорту даних.

Випробовується можливість додавати нові дані з файлів GPX, KML та GeoJSON з перевіркою їх валідності та подальшим відображенням основних відомостей про додані дані.

2) Функція експорту даних.

Випробовується функція збереження даних в файлах форматів GPX, KML або GeoJSON з попередньою адаптацією даних для обраного формату.

3) Функція створення та зміни географічних даних.

Випробовується функція створення географічних точок, маршрутів, треків. А також зміна параметрів існуючих одиниць даних. Програма має унеможливити створення некоректних даних. Також випробовуються спеціальні операції щодо маршрутів та треків - їх спрощення та скорочення.

4) Функція здійснення аналізу точок, маршрутів, зон.

Випробовуються ряд функцій аналізу географічних даних, такі як: знаходження відстані та напрямку між точками, довжини маршруту, набору та втрати висоти на маршруті, спільних ділянок маршрутів, географічних зон, які вміщують інші зони або маршрути, а також спрощення та скорочення маршрутів або треків

5) Функція взаємозв'язків між формами користувацького інтерфейсу програми.

Випробовується можливість та правильність переходу між формами користувацького інтерфейсу. Система не повинна зазнавати збоїв у роботі.

Інструкція користувача

«Система для редагування географічних даних» можна використовувати в два способи: за допомогою інтерфейсу командного рядка (CLI) та в якості бібліотеки підпрограм для Node.js або іншої платформи, що підтримує модулі стандарту CommonJS (програмний інтерфейс). Саме останній спосіб дозволяє використовувати всі можливості та функції програми.

CLI Доступний користувачу за командою **node earthlib** та включає наступні опції:

-h --help

Виведення на екран довідки щодо користування програми.

-i | --input <path|URL>

Ввід даних з вказаного файлу. Аргументом є шлях до файлу або URL.

earthlib --input myfile.gpx

Така команда відобразить інформацію про вміст файлу *myfile.gpx*

-o --output [<path>]

Вивід (збереження) змінених даних. Для збереження в іншому форматі має бути вказане відповідне розширення. Якщо шлях до файлу не буде вказано, дані будуть збережені в початковий файл.

earthlib --input myfile.gpx --output myfile.kml

Команда для конвертації GPX файлу *myfile.gpx* як в форматі KML.

-c --combine <path|URL>

Об'єднання інформації що міститься в двох файлах в один.

earthlib -i file1.gpx --merge file2.gpx -o myfile.kml

Об'єднає всю інформацію з файлів *file1.gpx* та *file2.gpx* в файл *myfile.kml*

Команди селектори. Дозволяють обрати елементи в файлі (точки, треки та маршрути відповідно), над якими має виконуватись обрана операція.

-p --points [<1,2,...>]

-t --tracks [<1,2,...>]

-r --routes [<1,2,...>]

earthlib -i myfile.gpx --tracks 1,4 --points -o newFile.gpx

Створить новий GPX файл, в якому будуть збережені треки номер 1 і 4, а також всі маршрутні точки.

-c --cut <reference>

Скорочення обраного треку або маршрута. В якості точки поділу може виступати або номер точки або інша точка. В останньому випадку програма визначить найближчу до точки відліку точку на маршруті і поділить маршрут в цьому місці.

--reverse

Команда для розвороту обраного треку або маршрута в зворотньому напрямку.

-s --simplify [<precision>]

Спрощення обраних треків або маршрутів з заданою точністю в метрах (10м по замовченню). Алгоритм видалить надлишкові точки з треку або маршруту.

-m --merge

Об'єднання треків або маршрутів в порядку їх визначення. Якщо маршрути юдуть мати спільні відрізки на суміжних частинах, вони будуть збережені один раз.

-d --delete

Видалення обраного елементу.

-a --analyze

Аналіз та відображення всієї доступної інформації щодо обраних треків та маршрутів. Таких яких назва, опис, кількість точок, довжина, перепад висот, набір та втрата висоти.

Приклади застосування:

earthlib -i myfile.gpx --tracks 1 --simplify 5

Спрощення треку номер 1 з точністю 5 метрів.

earthlib -i myfile.gpx --routes 2,1 --merge

Об'єднання маршрутів номер 2 та 1.

```
earthlib -i myfile.gpx --tracks --analyze
```

Аналіз всіх наявних в файлі треків.

```
-v --version
```

Відображення версії програми.

Для використання ПЗ в якості бібліотеки підпрограм вона може буде встановлена командою **npm install earthlib** (для менеджера пакетів npm). Бібліотека експортує глобальний об'єкт, що містить модулі та класи.

Приклад використання в системі CommonJS:

```
const earthlib = require('earthlib');  
const utils = earthlib.utils;  
const Track = earthlib.Track;
```

Бібліотека має наступний інтерфейс (API):

Модуль "io" містить інструменти, необхідні для імпорту та експорту даних в та з системи. Для додання файлу з геоданими в систему використовуйте одну з наступних функцій:

addFile(path)

Функція додання файлу. Параметр path є шляхом до файлу або URL.

Функція повертає об'єкт типу **geoFile**.

addString(string, type)

Функція для додання даних безпосередньо з перемінної String, яка містить його вміст файлу. Параметр type визначає формат файлу (gpx, kml або jeojson).

Якщо формат файл не буде відповідати очікуваному, буде викинуто виняток. Наприклад:

```
addFile("/myfile.gpx")
```

```
addString(fileContent, 'gpx')
```

Функція addString() негайно повертає об'єкт типу **geoFile**. В інших випадках буде повернено обіцянку (Promise), яка поверне такий об'єкт щойно операцію зчитування або завантаження буде закінчення.

new geoFile([path, [author]])

Створює новий об'єкт, що представляє файл з географічними даними. Об'єкти класу мають наступні атрибути, які можна змінювати для редагування його вмісту:

- ***routes :[Route]***
- ***tracks :[Track]***
- ***waypoints :[Point]***
- ***path:String***
- ***author :string***
- ***name :string***

description :string

В подальшому для збереження інформації в файл використовуйте звернення до наступних методів:

file.saveFile(path)

Для збереження інформації в файл обраного формату в файловій системі пристрою. Функція повертає обіцянку Promise.

file.saveString(type)

Для збереження змісту файла обраного типу в переміну типу String, яку повертає функція.

Основними видами одиниць даних для обробки географічної інформації в системі є маршрутні точки, треки та маршрути. Їм відповідають такі класи:

new Point (lat,lon,[ele])

Створює новий об'єкт, що представляє маршрутну точку. Параметри довготи і широти є обов'язковими. Параметр **ele** визначає висоту місця над рівнем моря. Об'єкти класу Point мають наступні атрибути:

- ***name :string***
- ***lat :float***
- ***lon :float***
- ***ele :float***
- ***description :[string]***
- ***time :[int]***

new Track ([points])

Створює новий об'єкт, що представляє трек. Об'єкти класу мають наступні атрибути:

name :string description

:string segments :[[Point]]

Атрибут **segments** містить масив масивів точок, що відповідають сегментам треку.

Для редагування та аналізу трека, клас має наступний набір методів:

track.remove(pointN)

Метод дозволяє видалити точку за індексом pointN з треку.

track.add(point[,pointN])

Використовуйте для додання нової точки point до кінця треку. У випадку зазначення другого параметру точка буде додана на індекс pointN.

track.replace(point,pointN)

Метод для заміни однієї точки (за індексом pointN) в треку на іншу (point).

Методи для скорочення треку:

track.cut(pointN[,saveStart])

Метод скоротить трек від або до заданої точки (якщо точка-орієнтир не лежить на маршруті, то буде визначена найближча ділянку треку до цієї точки), залежно від того, яка частина виявиться більшою. Опціональний параметр saveStart дозволяє чітко визначити яка частина треку має бути збережена - true для початку і false для кінця.

track.simplify(presc=10)

Дозволяє скоротити об'єм треку шляхом видалення надмірних точок. Параметр presc дозволяє визначити міру спрощення, а саме - максимальну відстань в метрах, на яку може розходитись лінія первинного і спрощеного треків.

track.merge(track)

Використовуйте для з'єднання двох треків.

track.nearest(point)

Визначає індекс точки трека, яка є найближчою до точки point.

track.overlap(track[,presc=10])

Дозволяє визначити спільні ділянки двох треків. Параметр presc визначає відстань між двома точками, при якій їх положення вважається ідентичним.

Результат повертається у вигляді двох масивів (для кожного треку), які містять значення початок і кінець спільної ділянки. Наприклад: [

```
[[track1_point1, track1_point2],...],  
[[track2_point1, track2_point2],...]  
]
```

Тут показано результат з однією спільною ділянкою - від track1_point1 до track1_point2 на першому треку і від track2_point1 до track2_point2 на другому треку.

track.length([slope=true])

Використовуйте щоб визначити довжину трека на місцевості в метрах. Параметр slope визначає чи необхідно враховувати кут схилу ділянок (у випадку визначеної висоти точок над рівнем моря).

track.elevGain()

Визначає загальну висоту підйому та спуску в метрах під час проходження треку від початку до кінця (у випадку визначеної висоти точок над рівнем моря). Результата повертається у вигляді масива з двох точок - значення набору і втрати висоти в метрах.

track.elevDiff()

Визначає загальну різницю в метрах між максимальною і мінімальною висотою на маршруті (у випадку визначеної висоти точок над рівнем моря).

new Route([Point])

Створює новий об'єкт класу Route з зазначеними маршрутними точками.

Об'єкти класу мають наступні атрибути:

- **name :string**
- **description :string**
- **points :[Point]**

Клас Route є дочірнім до класу Track, і має аналогічні методи для маніпуляції даними. Наприклад:

route.simplify(presc=10)

Метод що спростить масив точок маршрута.

new Zone(point1,point2)

Створює об'єкт класу Zone, який представляє прямокутну зону на земній поверхні, яка визначена двома кутовими точками. Об'єкти мають наступні параметри.

- **name :string**
- **description :string**
- **p1 :Point**
- **p2 :Point**

Модуль **utils** містить ряд утилітарних функцій для обробки даних.

distance(point1,point2)

Визначає відстань в метрах по земній поверхні між двома точками.

distanceToLine(point,point1,point2)

Визначає відстань в метрах від точки point до лінії, яка визначена точками point1 та point2.

bearing(point1,point2)

Функція для визначення напрямку (курсу) від точки point1 до точки point2 в градусах.

enclZone([tracks])

Функція визначає зону, яка вміщає всі треки [tracks]. Повертає об'єкт класу **Zone**.

enclZone([zones])

Аналогічна попередній, але в якості параметра виступає одна чи декілька інших зон [zones].

Текст програмного забезпечення

1. ./index.js

```
module.exports = require("./lib/earthlib.js")
```

2. ./package.json

```
{
  "name": "earthlib",
  "version": "1.0.0",
  "description": "Collection of tools for GIS operations.",
  "main": "index.js",
  "scripts": {
    "test": "node test.js"
  },
  "keywords": [
    "GIS"
  ],
  "author": "Pavlo Deshko",
  "license": "MIT",
  "dependencies": {
    "@tmcw/togeojson": "^3.0.1",
    "commander": "^4.0.1",
    "tokml": "^0.4.0",
    "xml-js": "^1.6.11",
    "xmldom": "^0.1.27"
  }
}
```

3. ./cli.js

```
const p = require('commander')
const path = require('path')
const io = require('./lib/io.js')
const utils = require('./lib/utils.js')
const Route = require('./lib/route.js')
const NO_TRACKS = "No tracks or routes were selected!"

p.version('0.0.1')
) p
  .option('-i, --input <path>', 'Import current file from given path')
  .option('-o, --output <path>', 'Export current file to given path and
format')
  .option('-c, --combine <filePath>', 'Merge current file with specified
file')

  .option('-p, --points <numList>', 'Specify selected waypoints by their
numbers e.i. "-p 1,2,4,10"')
  .option('-t, --tracks <numList>', 'Specify selected tracks by their
numbers e.i. "-t 1,2,4,10"')
  .option('-r, --routes <numList>', 'Specify selected routes by their
numbers e.i. "-r 1,2,4,10"')

  .option('-c, --cut [pointNum]', 'Shorten selected route or track to
section before point number or selected reference point')
  .option('--reverse', 'Reverse selected track or route')
  .option('-s, --simplify [prescition]', 'Simplify selected track or route
with given prescition')
  .option('-m, --merge', 'Merge two selected tracks or routes together')
.option('-d, --delete', 'Delete selected information')

  .option('-a, --analyze', 'Get all possible information on selected
objects')
```

```

p.parse(process.argv)
if(p.input){
  io.addFile(p.input).then(file=>{
    let selected = {
      waypoints: p.points?
p.points.split(',').map(d=>parseInt(d,10)).map(i=>(file.waypoints[i-1])): [],
      tracks: p.tracks?
p.tracks.split(',').map(d=>parseInt(d,10)).map(i=>(file.tracks[i-1])): [],
      routes: p.routes?
p.routes.split(',').map(d=>parseInt(d,10)).map(i=>(file.routes[i-1])): []
    }

    if(p.combine){
      io.addFile(p.combine).then(file2=>{
        Object.keys(selected).forEach(type=>{

file[type].splice(file[type].length,0,...file2[type])
          })
          file.saveFile(p.output)
        })
      }
      else if(p.reverse){
        let toRev =
selected.routes.map(r=>r.points).concat(selected.tracks.map(t=>t.segments[0])
)
        if (toRev.length ==0) throw new Error(NO_TRACKS)
        toRev.forEach(arr=>arr.reverse())
      }
      else if(p.cut){
        let toCut = selected.routes.concat(selected.tracks)
        if (toCut.length ==0) throw new Error(NO_TRACKS)
        let ref = typeof p.cut == "boolean" ? selected.waypoints[0]
: p.cut-1
        if (!ref) throw new Error("Please provide reference point or
point number!")
        toCut.forEach(obj=>obj.cut(ref))
      }
      else if(p.merge){
        if(selected.tracks.length>1)

selected.tracks.slice(1).forEach(t=>selected.tracks[0].merge(t))
        if(selected.routes.length>1)

selected.routes.slice(1).forEach(t=>selected.routes[0].merge(t))
        else throw new Error(NO_TRACKS)
      }
      else if(p.simplify){
        let toSim = selected.routes.concat(selected.tracks)
        if (toSim.length ==0) throw new Error(NO_TRACKS)
        toSim.forEach(obj=>obj.simplify( typeof p.simplify !=
"boolean" ? p.simplify : undefined))
      }
      else if(p.delete){
        Object.keys(selected).forEach(type=>{

selected[type].forEach(obj=>file[type].splice(file[type].indexOf(obj),1))
          })
        }
      }
    else if(p.analyze){
      let pool = selected.tracks.length+selected.routes.length > 0
      ? selected : file
      pool.routes.concat(pool.tracks).forEach((t,i)=>{
        let ele = utils.checkEle(t.points||t.segments[0]) ?
`Elevation difference: ${t.elevDiff()}m
Elevation gain: ${t.elevGain()[0]}m
Elevation lost: ${t.elevGain()[1]}m`

```



```

        : '    Not every point in this track(route) has valid
elevation data!'

```

```

        let full = `
${t instanceof Route ? 'Route ':'Track'} ${file[t instanceof Route ?
'routes':'tracks'].indexOf(t)+1}
${t.name.length>0?`Name: ${t.name}`:''}
Number of points: ${t.points||t.segments[0]}.length}
Length: ${utils.roundTo(t.length(),2)+'m'}
${ele}
${t.description.length>0?`Description: ${t.description}`:''}
`

        console.log(full)
    })
}
if(p.output && !p.combine){
    file.saveFile(p.output)
    console.log(`File saved at
    ${p.output}`)
}else{
    let points = file.waypoints.map((p,i)=>` ${i+1}. Name:
    ${p.name.length>0?p.name: '(none)'}    Description:
    ${p.description.length>0?p.description: '(none)'}\n`).join('')
    let tracks = file.tracks.map((t,i)=>` ${i+1}. Name:
    ${t.name.length>0?t.name: '(none)'}    Description:
    ${t.description.length>0?t.description: '(none)'}\n`).join('')
    let routes = file.routes.map((r,i)=>` ${i+1}. Name:
    ${r.name.length>0?r.name: '(none)'}    Description:
    ${r.description.length>0?r.description: '(none)'}\n`).join('')
    console.log(
    `File ${path.basename(p.input)} ${file.name || ''}
Waypoints:
${points}
Tracks:
${tracks}
Routes:
${routes}

    `)
}

    }).catch(er=>console.log(er.message))
}else{
    p.help()
}

```

4. ../lib/earthlib.js

```

module.exports = {
    io: require('./io.js'),
    GeoFile:
    require('./geoFile.js'), Track:
    require('./track.js'), Route:
    require('./route.js'), Point:
    require('./point.js'), Zone:
    require('./zone.js'), utils:
    require('./utils.js')
}

```