Національний лісотехнічний університет України

(повне найменування вищого навчального закладу)

Навчально-науковий інститут деревообробних та

комп'ютерних технологій і дизайну

(повне найменування інституту, назва факультету (відділення))

Кафедра інформаційних технологій

(повна назва кафедри (предметної, циклової комісії))

Пояснювальна записка

до дипломної роботи

другий (магістерський)

рівень вищої освіти

на тему: **«Розроблення інформаційної системи прогнозування лісових пожеж»**

Виконав: студент VI курсу групи КН-61(м)

спеціальності

122 "Комп'ютерні науки"

(шифр і назва напряму підготовки, спеціальності)

Віхоть О. Я.

(прізвище та ініціали)

Керівник Пірко І. Б.

(прізвище та ініціали)

Рецензент _____

(прізвище та ініціали)

Львів – 2021 р.

ННІ деревообробних та комп'ютерних технологій і дизайну

(повне найменування вищого навчального закладу)

Кафедра інформаційних технологій

Рівень вищої освіти другий (магістерський)

Спеціальність 122 «Комп'ютерні науки»

(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____Крошний І. М.

"____" _____2021 року

З А В Д А Н Н Я

НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Віхоть Олег Ярославович

(прізвище, ім'я, по батькові)

1. Тема роботи **Розроблення інформаційної системи прогнозування лісових пожеж**

керівник роботи Пірко І.Б., канд. фіз.-мат. наук, доц.

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

затверджені наказом вищого навчального закладу від "31"грудня 2020 р. № С-593

2. Термін подання студентом роботи 10 грудня 2021 р.

3. Вихідні дані до роботи _____

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

4.1. Стан проблемної області.

4.2. Інформаційне забезпечення

4.3. Математичне забезпечення

4.4. Програмне забезпечення

4.5. Розроблення стартап проекту

4.6. Висновки

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

Презентація у Microsoft Office PowerPoint

6. Дата видачі завдання 18 грудня 2020 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Огляд літературних даних та інших джерел згідно досліджуваної теми	18.12-30.01.2021	
2	Аналіз досліджуваної теми та вибір відповідних варіантів її розробки	01.02-27.02.2021	
3	Постановка задачі та її формалізація	01.03-01.04.2021	
4	Вибір та обґрунтування методів і засобів проведення дослідження	02.04-30.04.2021	
5	Розроблення концептуальної схеми реалізації завдання	01.05-01.06.2021	
6	Програмна реалізація завдання	02.06-30.10.2021	
7	Тестування програмного продукту та отриманих результатів	01.11-15.11.2021	
8	Розробка пояснювальної записки магістерської роботи	16.11-30.11.2021	
9	Корегування пояснювальної записки згідно вимог, розроблення презентації	01.12-09.12.2021	

Студент _____ **Віхоть О. Я.**
(підпис) (прізвище та ініціали)

Керівник роботи _____ **Пірко І. Б.**
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Дипломна робота містить 89 сторінок пояснювальної записки, 13 рисунків, 4 додатків, 19 джерел, 10 таблиць.

Серед багатьох важливих проблем охорони та відтворення лісових ресурсів однією з найбільш актуальних являється проблема боротьби з лісовими пожежами, а саме оцінка та прогноз пожежної небезпеки в лісі, які характеризують потенціальну загрозу виникнення лісових пожеж, їх розвиток і нанесення збитків лісовим ресурсам.

Розроблено та реалізовано систему оцінки пожежної небезпеки в лісі, яка враховуватиме погодні умови, стан лісової рослинності та можливість появи джерел вогню на території, яка охороняється. Реалізовано програмну модель прогнозування лісових пожеж, в рамках даної моделі проведено прогноз динаміки поширення лісових пожеж.

Результати роботи даної системи прогнозування можуть використовуватися при оптимізації маршрутів повітряного та наземного патрулювання. З врахуванням особливостей лісових територій дана програма може бути впроваджена в структурні підрозділи системи охорони лісових територій.

Ключові слова: лісові пожежі, пожежна небезпека, оцінка і прогноз пожежної небезпеки, Python, Tkinter, Matplotlib.

ABSTRACT

Diploma paper contains 89 pages of explanatory note, 13 pictures, 4 application, 19 used literary sources, 10 tables.

Among the many important problems of protection and reproduction of forest resources, one of the most pressing is the problem of forest fire control, namely the assessment and forecast of forest fire hazards, which characterize the potential threat of forest fires, their development and damage to forest resources.

A system for assessing the fire hazard in the forest has been developed and implemented, which will take into account weather conditions, the condition of forest vegetation and the possibility of the appearance of fire sources in the protected area. The program model of forest fire forecasting is implemented, within the framework of this model the forecast of forest fire spread dynamics is carried out.

The results of this forecasting system can be used to optimize air and ground patrol routes. Taking into account the peculiarities of forest areas, this program can be implemented in the structural units of the forest protection system.

Keywords: forest fires, fire hazard, fire hazard assessment and forecast, Python, Tkinter, Matplotlib.

ТЕХНІЧНЕ ЗАВДАННЯ

Розробити інформаційну систему для прогнозування лісових пожеж.

1. Вивчити предметну область та провести аналіз стану проблемної області.

2. Створити математичну модель поширення низових лісових пожеж.

3. Розробити засобами Tkinter мовою програмування Python застосунок користувача зі зручним інтерфейсом.

4. Протестувати роботу розробленої інформаційної системи, дослідити параметри даної системи.

.

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

- БД – база даних;
- ВМО – Всесвітня метеорологічна організація;
- ІПН – індекси пожежної небезпеки;
- ЛГМ – лісові горючі матеріали;
- ООП – об’єктно-орієнтоване програмування;
- ПЗ – програмне забезпечення;
- ПН – пожежна небезпека;
- ПР – пожежний ризик;
- ECMWF – Європейський центр середньострокових прогнозів погоди;
- FIRMS – Fire information for resource management system;
- FR (fire risk) – пожежний ризик;
- GUI – Graphical User Interface;
- NCAR – Національний центр атмосферних досліджень;
- NOAA GFS – Глобальна система прогнозування.

ЗМІСТ

ВСТУП

Актуальність роботи

Серед багатьох важливих проблем охорони та відтворення лісових ресурсів однією з найбільш актуальних являється проблема боротьби з лісовими пожежами. Тенденція зростання числа лісових пожеж, яка проявилася в останні роки в нашій країні, пов'язана з різким зниженням коштів, що виділяються на охорону лісів, що привело до істотного послаблення лісопожежних служб.

Основою ефективної їх роботи є оцінка та прогноз пожежної небезпеки (ПН) в лісі, які характеризують потенціальну загрозу виникнення лісових пожеж, їх розвиток і нанесення збитків лісовим ресурсам. На сьогоднішній день оцінка пожежної небезпеки проводиться лише по великих територіальних одиницях, що негативно впливає на точність підсумкових прогнозів. Сучасний стан вимагає об'єктивного та гнучкого підходу до охорони лісів, в тому числі до оцінки та прогнозу пожежної небезпеки.

Лісові пожежі є серйозною проблемою населення в усьому світі, так як крім прямого збитку, що включає в себе кількість людських жертв, витрати на гасіння і відновлення постраждалих територій, вартість вигорілої деревини, порушується екологічний баланс на даній території: знищуються місця проживання тварин, що призводить до їх міграції в інші райони, пошкоджується структура, хімічний склад, мікрофлора і фауна грунту, відбуваються викиди вуглекислого газу і канцерогенів в атмосферу.

Предметом дослідження є розробка інформаційної системи прогнозування лісових пожеж.

Об'єктом дослідження є лісові пожежі.

Мета роботи – розробка та реалізація системи оцінки пожежної небезпеки в лісі, яка враховуватиме погодні умови, стан лісової рослинності та можливість появи джерел вогню на території, яка охороняється.

Відповідно до мети дослідження поставлено наступні **завдання**:

- провести огляд літератури по даній тематиці, проаналізувати існуючі системи прогнозування лісових пожеж;
- розробити інформаційну систему прогнозування лісових пожеж;
- реалізувати програмну модель прогнозування лісових пожеж;
- в рамках програмної моделі провести дослідження динаміки поширення лісових пожеж.

Наукова новизна одержаних результатів

Запропоновано новий підхід для визначення пожежної небезпеки, який оснований на детальній характеристиці досліджуваної території лісових масивів. Даний підхід передбачає використання експертних оцінок, інтерполяційних методів.

Практичне значення одержаних результатів

Розроблено програмний комплекс, який представляє собою основу інформаційної системи для детальної оцінки пожежної небезпеки в лісових масивах. Результати роботи даної інформаційної системи можуть використовуватися при оптимізації маршрутів повітряного та наземного патрулювання. З врахуванням особливостей лісових територій дана програма може бути впроваджена в структурні підрозділи системи охорони лісових територій.

РОЗДІЛ 1. СТАН ПРОБЛЕМНОЇ ОБЛАСТІ

1.1. Лісові пожежі, їх класифікація

Щорічно на Землі виникають до 400 тисяч лісових пожеж, які пошкоджують близько 0,5 % загальної площі лісів і викидають в атмосферу мільйони тонн продуктів згоряння. Висока горимість лісів спостерігається в США, Канаді, Іспанії, Португалії, Франції, Росії, Україні, Австралії. Причиною лісових пожеж в 70-90 % випадків є діяльність людини.

Лісові горючі матеріали (ЛГМ) поділяються на наступні групи:

- грунтові;
- надгрунтові;
- ступінчасті;
- кронові.

Грунтові горючі матеріали – всі грунтові матеріали нижче надгрунтового покриву, які зазвичай підтримують процес горіння або тління, які пов’язані з грунтовими пожежами. Цей тип ЛГМ включає підстилку і гумус, коріння дерев і чагарників, гнилу деревину, торф і тирсу.

Надгрунтові горючі матеріали – ЛГМ, що знаходяться в безпосередньому контакті з грунтом (мохи, лишайники, гілочки, листя, хвоя).

Ступінчасті горючі матеріали – ЛГМ, які служать сходинками для переходу низової пожежі у верхову. Вони забезпечують вертикальну безперервність між шарами і дозволяють пожежі переходити від наземних ЛГМ на крони дерев або чагарників (запалення в формі факела, спалахування крон) і підтримувати безперервне поширення верхових пожеж.

Кронові (надземні) горючі матеріали – живі і відмерлі ЛГМ, які безпосередньо не контактують з землею і складаються в основному з листя, пагонів, гілок, стебел.

Залежно від характеру загоряння і складу лісу лісові пожежі поділяються на низові, верхові і грунтові.

Для низової пожежі характерна витягнута форма згарища з нерівною кромкою. Швидкість поширення низових пожеж проти вітру в 6-10 разів менша, ніж за вітром. У нічний час швидкість поширення пожежі менша, ніж вдень. При зміні напрямку вітру ускладнюється визначення форми пожежі. У таких випадках, особливо коли пожежа прийняла великі розміри, можливо оточення вогнем людей в лісі. При низовій пожежі згорає лісова підстилка, лишайники, мохи, трави, опалі на землю гілки. Швидкість руху пожежі за вітром становить 0,25-5 км/год. Висота полум'я сягає 2,5 м. Температура горіння близько 700° С.

За швидкістю поширення вогню низові пожежі поділяють на побіжні й стійкі. При низовій побіжній пожежі переважає полум'яний тип горіння, при стійкій – безполум'яний. Стійкі низові пожежі більш небезпечні, так як при них глибоко пошкоджуються підстилка і живі дерева.

Така пожежа поширюється з великою швидкістю, обходячи місця з підвищеною вологістю, тому частина площі залишається незачепленою вогнем. Побіжні пожежі в основному відбуваються навесні, коли просихає лише верхній шар дрібних горючих матеріалів. Стійкі низові пожежі поширюються повільно, при цьому повністю вигоряє надгрунтовий покрив, сильно обгоряють коріння і кора дерев, повністю згоряють підріст і підлісок. Стійкі пожежі виникають переважно з середини літа, коли просихає підстилка.

Верхова лісова пожежа охоплює листя, хвою, гілки та всю крону, може охопити трав'янисто-моховий покрив грунту і підріст. Вогонь рухається зі швидкістю, що перевищує 5-30 км/год, захоплюючи величезні площі і приносячи великі руйнування. Верхові пожежі найбільш небезпечні, боротьба з ними особливо важка. Розвиваються вони зазвичай при посушливій вітряній погоді з низової пожежі в насадженнях з низькоопущеними кронами, в різновікових насадженнях. Верхова пожежа – це зазвичай завершальна стадія пожежі.

Верхові пожежі, як і низові, можуть бути швидкими і стійкими. Ураганна пожежа виникає при сильному вітрі і поширюється зі швидкістю від 7 до 30 км/год. Вона небезпечна високою швидкістю поширення. При повальній верховій пожежі вогонь

рухається суцільною стіною від надгрунтового покрову до крон дерев зі швидкістю до 8 км/год. Ліс при цьому вигоряє повністю.

Грунтові (підземні) пожежі в лісі найчастіше пов'язані із загорянням торфу, яке стає можливим в результаті осушення боліт. Поширюються такі пожежі зі швидкістю до 1 км на добу. Вогонь часто не виходить на поверхню. Такі пожежі можуть бути малопомітні і можуть поширюватися на глибину до декількох метрів, внаслідок чого представляють додаткову небезпеку і вкрай погано піддаються гасінню (торф може горіти без доступу повітря і навіть під водою). Дерева на площі пожежі падають, створюється сильна захаращеність і посилюється загальна загроза пожежі в подальшому.

За інтенсивністю лісові пожежі поділяються на слабкі, середні і сильні (табл. 1.1). Інтенсивність горіння залежить від стану і запасу горючих матеріалів, ухилу місцевості, часу доби і особливо сили вітру.

Табл. 1.1. Показники сили пожежі

Параметри пожежі	**Значення показників сили пожежі**		
	слабкої	середньої	сильної
Низова пожежа			
швидкість поширення вогню, м/хв	до 1	1-3	більше 3
висота полум'я, м	до 0,5	0,5-1,5	більше 1,5
Верхова пожежа			
швидкість поширення вогню, м/хв	до 3	3-100	більше 10
Підземна пожежа			
глибина прогоряння, м	до 25	25-50	більше 50

1.2. Причини виникнення лісових пожеж

Щорічно в світі реєструється велика кількість лісових пожеж. Виникають вони в результаті поєднання багатьох чинників. Причини лісових пожеж можуть бути природні або антропогенні. Близько 95 % пожеж в світі відбувається внаслідок діяльності людей і тільки 5 % пожеж виникають в результаті природних причин -

екстремальних погодних умов (тривалих періодів спеки, посухи, сильного вітру), ударів блискавок. При цьому природні умови можуть лише сприяти займистості ЛГМ. Фактори ж, в результаті яких виникають лісові пожежі, є здебільшого антропогенними.

Кількість лісових пожеж протягом останніх років істотно збільшилася. Однак пік кількості лісових пожеж в світі за всю історію спостереження припав на останнє десятиліття минулого століття. Цьому сприяло розширення землекористування та сільськогосподарської діяльності, що призвело до збільшення маси ЛГМ на одиницю площі, а також до розростання кількості людських поселень в безпосередній близькості з ділянками лісу. Почастішання природних лих в усьому світі експерти пов’язують також з глобальним потеплінням і його наслідками.

1.2.1. Природні причини лісових пожеж

Основними природними причинами пожеж являються:

- блискавки;
- самозагоряння торфу.

Блискавки – це атмосферний фактор, який є найпоширенішою причиною появи природного вогню і природних пожеж. Температура в каналі блискавки досягає 30 тисяч градусів. Щодня на землі відбувається близько 44 тисяч гроз і щомиті виблискують понад 100 блискавок.

Парадоксальним є той факт, що блискавка може бути суперечливою за своїми наслідками. Вона запалює органічну речовину, але також може сприяти причиною його гасіння. Справа в тому, що електричний розряд активізує коагуляцію крапель. Тому відразу після спалаху блискавки слідує посилення дощу.

Самозаймання торфу – найхарактерніша і найбільш поширена причина лісових пожеж. При зниженні вологості торфу від 60 до 10 % завдяки діяльності грибків, що розкладають органіку, температура на глибині 0,5-1 м може підніматися до 50 градусів, що є причиною швидкого розмноження термофільних рослин. В результаті їх життєдіяльності розігрів доходить до 80 градусів, і торф починає

перетворюватися в обвуглену масу. При достатньому доступі кисню можливо його займання.

1.2.2. Антропогенні причини лісових пожеж

Основними антропогенними причинами лісових пожеж являються:

- рекреаційна діяльність;
- сільськогосподарська діяльність;
- промислова діяльність;
- залізниці;
- навмисні підпали;
- наявність в лісах великої кількості горючих матеріалів в зв'язку з лісозаготівлями або гасінням пожеж.

Лісові пожежі часто виникають в результаті рекреаційної діяльності людей (проведення відпусток, рибалка, пікніки, некомерційний збір ягід, туризм) через необережне поводження з вогнем під час відпочинку на природі. При цьому причинами пожеж бувають сигаретні недопалки, бите скло, багаття. Істотний вплив тут мають міста, численне населення яких відпочиває в зоні прилеглого лісу і не дотримується елементарних протипожежних вимог. В останні роки збільшення числа пожеж сприяє планове випалювання території для різних господарських цілей. Випалювання передбачається, наприклад, стратегією меліорації земель для скотарства. Крім того, власники дрібних фермерських господарств використовують випалювання для підготовки землі. Полум'я від такого випалювання місцевості нерідко через недогляд переходить межі передбачуваної території і, особливо в періоди посухи, поширюється на сусідні ліси.

Головною причиною лісових пожеж в світі вважаються підпали з метою подальшого використання територій для будівництва або сільського господарства. Найбільшого поширення вони отримали в Бразилії, Індонезії, Іспанії, Португалії. Навмисні підпали мають місце в районах, де недостатньо землі для сільськогосподарського виробництва, або виникають конфлікти через права володіння

ресурсами або доступу до них. Чутливість лісу до пожеж і до їх наслідків безпосередньо залежить від ступеня порушень лісу до пожежі. Зазвичай інтенсивність пожеж і шкоди, яку вони завдають, набагато вища в лісах, які зазнали рубок, ніж в природних лісах.

1.2.3. Причини виникнення лісових пожеж в Україні

Основна причина лісових пожеж в Україні – дія антропогенних факторів. Необережна або недобросовісна діяльність населення, неорганізованих відпочиваючих і туристів на тлі відповідних природних умов (пожежонебезпечні типи лісу, які представлені хвойними породами, суха спекотна погода і т.д.) є причинами переважної більшості лісових пожеж. Пожежі виникають через непогашені під час пікніків багать, кинутих в траву непогашених недопалків та сірників, а також через безконтрольне випалювання залишків рослинності на сільськогосподарських угіддях і придорожніх смугах, вогонь від яких поширюється на лісову територію.

Ще однією причиною лісових пожеж в Україні є велика кількість відходів лісопромисловості. У зв’язку з тим, що в Україні немає комплексної програми по роботі з відходами лісопромисловості, підвищується ймовірність виникнення і поширення пожеж. Як правило, деревна тріска, гілки та інші відходи, які залишаються після санітарної очистки лісу, викидаються і залишаються гнити в ярах. Особливо небезпечними щодо пожежі є суха тирса і деревний пил. Ці відходи могли б бути використані при виробництві альтернативного твердого палива (паливних брикетів і гранул), як це робиться в європейських країнах. З усього вищесказаного можна зробити висновок, що в Україні, як і в усьому світі, причини лісових пожеж пов’язані в основному з антропогенною діяльністю. Лісові пожежі є наслідком меркантильного ставлення людини до лісу. Адже людина в першу чергу думає про своє матеріальне благополуччя, а природні ресурси, до яких відноситься ліс, розглядає як засіб для свого збагачення, а не як дар природи, який необхідно зберігати.

1.3. Наслідки лісових пожеж

Лісові пожежі завдають величезної шкоди навколишньому середовищу, якості життя людини, ефективності його діяльності, а також в цілому економіці країни.

Еколого-економічні збитки від лісової пожежі включають: вартість втрат деревини; збиток від пошкодження молодняків природного і штучного походження; збиток від пошкодження ресурсів побічного лісокористування; витрати на гасіння лісових пожеж; вартість згорілих об'єктів і готової продукції в лісі (зниження вартості об'єктів і готової продукції, пошкоджених пожежею); витрати на додаткові санітарні рубки в насадженнях, пошкоджених лісовими пожежами; збиток від зниження ґрунтозахисних, санітарно-гігієнічних, водоохоронних та інших функцій лісу; збиток від забруднення повітряного середовища продуктами горіння; збиток від загибелі тварин і рослин.

На рис. 1.1 відображена структура наслідків лісових пожеж.

Рис. 1.1. Наслідки лісових пожеж.

1.4. Пожежна небезпека, лісопожежний ризик

Визначення лісопожежних термінів є спробою пояснення лісової пожежі як природного явища і засновані на його сприйнятті людиною.

Особливо важливо розуміти різницю термінів fire hazard і fire danger. Hazard означає небезпеку, ризик, шанс, а danger – небезпека, загроза. Поняття fire hazard має більш вузький зміст, ніж fire danger. В українській мові аналоги термінів fire hazard та fire risk не використовуються, а використовується термін fire danger, який перекладається як пожежна небезпека (пожежонебезпека).

В даний час існує декілька визначень поняття fire danger (пожежна небезпека ПН). Згідно з визначенням Продовольчої і сільськогосподарської організації ООН (Food and Agricultural Organisation of the United Nations) пожежна небезпека – це результуюча (часто виражена у вигляді індексу) постійних і змінних факторів, що впливають на виникнення, поширення і складність управління лісовими пожежами, а також збиток, який вони завдають.

Згідно з визначенням Канадського комітету з управління лісовими пожежами (Canadian committee on Forest Fire Management) пожежна небезпека – це загальний термін, який використовується для вираження постійних і змінних факторів, що впливають на лісову пожежу, які визначають легкість виникнення, швидкість поширення, складність контролю, а також оцінку наслідків лісових пожеж.

У словнику лісопожежної термінології, який виданий Національною координаційною групою по природних пожежах (США) термін пожежна небезпека визначається як сума постійних і змінних факторів пожежної навколишнього середовища, що впливають на виникнення, поширення пожежі, а також складність її гасіння.

Всі наведені вище визначення пожежної небезпеки описують набір характеристик, що впливають на лісові пожежі. Ці визначення об’єднюють фактори пожежної небезпеки, що мають різну природу. Однак було б доцільно розглядати кожен фактор окремо з метою визначення його відносного впливу на лісопожежний ризик.

Пожежна небезпека поділяється на постійну та змінну.

Постійна пожежна небезпека – це результуюча всіх факторів пожежної небезпеки, які відносно незмінні на даній площі, наприклад, цінності лісу, який зазнає ризику, географічних і геометричних особливостей місцевості, типу ЛГМ, впливу переважаючого вітру.

Мінлива пожежна небезпека – це результуюча дія всіх факторів пожежної небезпеки, які змінюються щодня, щомісяця або щорічно (погода, вологовміст ЛГМ, стан рослинності, змінюється показник ймовірності виникнення пожеж). Fire danger перекладається як пожежна небезпека і визначається як термін, який використовується для вираження ймовірності виникнення пожежі на певній території та можливої шкоди від пожежі.

Виділяють три типи пожежної небезпеки в залежності від задач управління лісовими пожежами:

1) природна (типова) пожежна небезпека – визначається для:

– площі 100 га;

– періоду часу 10 років.

2) денна пожежна небезпека – визначається в залежності від погодніх умов для:

– площі 100 га;

– періоду часу 1 день;

– виявлення пожежі.

3) пожежна зрілість – оцінка готовності рослинності до загоряння в даний час для:

– площі 1-10 га;

– періоду декілька годин;

– прогнозування поведінки пожежі.

Термін fire hazard перекладається як небезпека виникнення пожежі або як ступінь пожежної небезпеки.

Згідно з першим визначенням fire hazard означає небезпеку виникнення пожежі та визначається як комплекс ЛГМ, що характеризується обсягом, типом, станом,

структурою і місцезнаходженням, який визначає легкість займання і складність гасіння вогню.

Степінь пожежної небезпеки залежить тільки від стану ЛГМ і не залежить від погодних умов, а також характеристик навколишнього середовища, в якій знаходяться ЛГМ. Однак для виникнення і підтримання пожежі необхідні три основні чинники: паливо, тепло і кисень, а поведінка пожежі залежить від палива, погоди і топографії. Іншими словами, паливо – це лише один з кількох компонентів, необхідних для виникнення пожежі та визначають його поведінку, отже, для прогнозування пожежі недостатньо володіти інформацією тільки про fire hazard.

Визначення терміну fire risk (пожежний ризик ПР) особливо неоднозначне. Почнемо з опису поняття ризик. Термін ризик використовується самими різними групами людей в різних ситуаціях. Відповідно, існує дуже широкий спектр понять, які цей термін може означати.

Виходячи з інженерно-технічного визначення, зміст терміну ризик дуальний і складається з двох компонентів: ймовірність події і його результат. Більшість сучасних словників лісопожежних термінів визначають пожежний ризик вужче, без урахування другого компонента, пов’язуючи пожежний ризик тільки з ймовірністю займання. Таке визначення фокусується на причину пожежі (ймовірності займання) і не враховує його наслідків.

Лісопожежний ризик – це ймовірність або можливість початку пожежі, що визначається наявністю і дією причинних факторів (тобто потенційних джерел займання).

В американській системі NFDRS під лісопожежним ризиком розуміють ймовірність виникнення пожежі, яка поширюється. При цьому в NFDRS виділяються два джерела лісопожежного ризику: розряди блискавок і антропогенне навантаження. Розраховується два індекси лісопожежного ризику, кожен з яких представляє собою число від 0 до 100. Потім на їх основі визначається загальний показник лісопожежного ризику (також в інтервалі від 0 до 100). У лісопожежній літературі можна знайти лише кілька джерел, які при визначенні ризику приймають до уваги обидва аспекти: ймовірність загоряння і результат пожежі.

Таким чином, аналіз пожежного ризику містить дві основні задачі: визначення ймовірності виникнення пожежі та визначення його наслідків. Ці завдання є ключовими для трьох основних областей дослідження лісової пожежі (визначення наслідків пожежі, опис поведінки пожежі та визначення ймовірності виникнення пожежі). У цьому сенсі аналіз ризику є міждисциплінарним завданням (рис. 1.2). Кількісна оцінка ризику об'єднює всі три основних напрямки дослідження лісових пожеж.

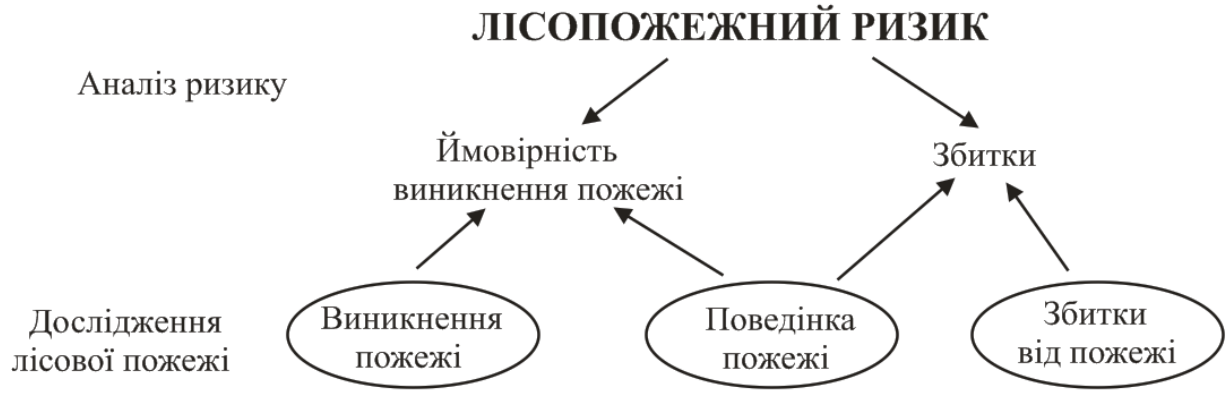

Рис. 1.2. Аналіз ризику та дослідження лісової пожежі.

ВИСНОВКИ ДО РОЗДІЛУ 1

В першому розділі проаналізовано предметну область, засоби та технології проектування інформаційної системи прогнозування виникнення пожеж в лісових масивах.

Представлено класифікацію лісових пожеж, а також лісових горючих матеріалів. Залежно від характеру загоряння і складу лісу лісові пожежі поділяються на низові, верхові і грунтові. За інтенсивністю лісові пожежі поділяються на слабкі, середні і сильні. Інтенсивність горіння залежить від стану і запасу горючих матеріалів, ухилу місцевості, часу доби, сили вітру. Розглянуто причини виникнення лісових пожеж, а також вказано наслідки, до яких можуть привести лісові пожежі.

З допомогою цієї інформаційної системи можна буде моделювати динаміку поширення пожеж в лісових масивах, проводити оцінку наслідків лісових пожеж.

РОЗДІЛ 2. ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1. Аналіз існуючих інформаційних систем прогнозування лісових пожеж

В даний час завдання збереження лісів є одним із пріоритетних для суспільства. Найбільш сильного збитку лісовим масивам приносять природні пожежі. Щорічно на території багатьох держав в пожежнонебезпечний період виникають безліч вогнищ загорянь, через що знищуються мільйони гектарів лісу.

Для боротьби з лісовими пожежами існують два типи систем. Перший тип – це системи моніторингу. За допомогою камер операторам передаються зображення про стан лісу. Багато систем цього типу має алгоритм розпізнавання пожежі на даних, які передаються. Таким чином, система зменшує навантаження на оператора.

Також існують декілька недоліків даного типу систем.

Головним недоліком є необхідність постійного контролю оператора за роботою системи. Відповідно, для роботи системи необхідні великі витрати людських ресурсів. Виходячи з цього, підвищується ризик виникнення людської помилки в процесі експлуатації системи.

Іншим недоліком систем є помилкове розпізнавання пожеж. Як пожежа може бути розпізнано будь-яке інше теплове випромінювання або дим від інших джерел.

Деякі системи використовують для моніторингу не камери, а супутники. У цих випадках недоліком є затримка інформації про зафіксовані пожежі. Відповідно, час реагування і гасіння підвищується, що призводить до великих втрат.

До систем цього типу відносяться:

- Forest Fire Detection;
- Лесной Дозор;
- FIRMS;
- Система моніторингу пожежної та екологічної безпеки.

Приклад інтерфейсу системи моніторингу представлено на рис. 2.1.

Рис. 2.1. Інтерфейс системи FIRMS.

Для кожного типу систем було виявлено ряд критеріїв оцінки. У табл. 2.1 показано порівняння систем моніторингу за цими критеріями.

Табл. 2.1. Порівняння систем моніторингу

	Автоматичне розпізнавання пожежі	Вироблення рекомендацій по гасінню	Визначення характеристик пожежі	Отримання інформації про оцінку площ, які вигоріли	Результат
Forest Fire Detection	5	5	5	3	4,6
Лесной дозор	4	2	4	2	3
FIRMS	2	1	3	5	2,4
Система моніторингу пожежної та екологічної безпеки	1	3	1	1	1,6

В ході порівняння систем моніторингу було виявлено, що найкращою є Forest Fire Detection. Вона поступається тільки системі FIRMS в отриманні інформації про оцінку вигорілих площ.

Іншим типом систем для боротьби з лісовими пожежами є системи прогнозування. У цих системах відбувається не розпізнавання вже наявної пожежі, а складання прогнозу її виникнення.

До цього типу відносяться:

- система моніторингу пожежної небезпеки та прогнозування надзвичайних лісопожежних ситуацій;
- система моніторингу лісових пожеж;
- ArcGIS;
- Антистихія.

Результат роботи однієї з цих систем представлено на рис. 2.2.

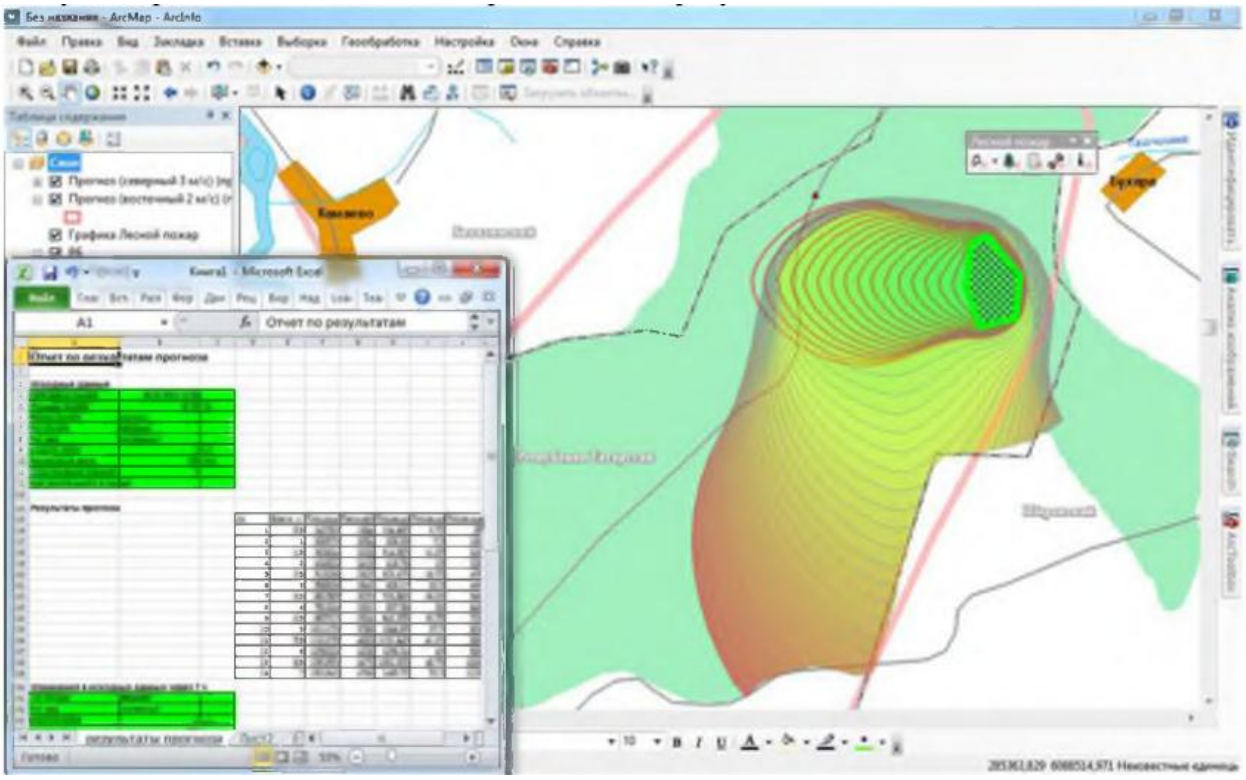

Рис. 2.2. Результати моделювання лісової пожежі в ArcGIS.

У цього типу систем також існують недоліки.

Основний недолік полягає в тому, що результати прогнозу не є точною інформацією. Неможливо точно визначити час і місце виникнення пожежі. Також пожежа може і не відбутися, і тоді всі заходи з підготовки до гасіння будуть марними.

Іншим недоліком є надмірність інформації в деяких системах. Існують системи, які прогнозують не тільки природні пожежі, а й інші надзвичайні ситуації.

В табл. 2.2 показано порівняння систем прогнозування за виявленими критеріями.

Табл. 2.2. Порівняння систем прогнозування

	Збереження інформації про виявлені пожежі	Прогнозування наслідків	Розроблення пропозицій по підвищенню пожежної безпеки	Створення прогнозів на різні періоди
Система моніторингу пожежної безпеки і прогнозування лісопожежних ситуацій	2	1	5	3
Система монтіорингу лісових пожеж Інсістем	4	3	2	2
ArcGIS	3	5	3	4
Антистихія	5	2	1	5

В ході порівняння систем прогнозування було виявлено, що кращими системами є ArcGIS і Антистіхія. Кращою системою моніторингу було виявлено FIRMS, а системами прогнозування ArcGIS та Антистіхія.

Система повинна бути обрана виходячи з потреб організації.

Якщо підприємству необхідно отримувати оперативну інформацію про пожежі і воно має досить людських ресурсів для підтримки оптимальної роботи системи, то слід купувати систему моніторингу.

Якщо ж підприємство хоче передбачати появу пожеж і проводити їх оперативну ліквідацію, то кращим вибором буде система прогнозування.

У будь-якому випадку систему необхідно буде модернізувати для конкретної організації.

Наявність достовірного прогнозу поширення та розвитку лісової пожежі дозволяє оцінити загрозу природному середовищу, об'єктам економіки і населеним пунктам, вжити необхідних заходів щодо запобігання збиткам, спланувати роботу протипожежних сил.

В умовах лісових пожеж (ЛП) для прийняття правильних управлінських рішень щодо їх ліквідації необхідний оперативний і точний прогноз динаміки ЛП на основі даних розвідки. У зв'язку з цим актуальним бачиться впровадження в роботу пожежних аварійно-рятувальних підрозділів комп'ютерних програм за оцінкою динаміки ЛП.

На сучасному етапі вченими різних країн розроблений ряд математичних моделей поширення ЛП. Вони істотно відрізняються по набору вихідних даних і алгоритмах їх обробки. Наприклад, в простих методиках, заснованих на емпіричних залежностях, використовуються тільки метеорологічні дані і дані про надгрунтовий покрив, а результатом є швидкості поширення фронту, флангів і тилу пожежі. Більш складні моделі додатково враховують рельєф місцевості, склад і стан лісового горючого матеріалу, природні та штучні перепони і з використанням багатовимірних рівнянь газової динаміки визначають області згорілих і палаючих ділянок лісу в тривимірному просторі. На основі цих моделей розроблено різні програмні комплекси.

У Республіканському центрі управління та реагування на надзвичайні ситуації МНС Республіки Білорусь використовується ПК "Розрахунок і візуалізація динаміки лісової пожежі", що дозволяє в реальному часі розраховувати конфігурацію контуру ЛП, його периметр, площу лісу, який вигорів, візуалізувати результати на електронній карті, прогнозувати стан фронту ЛП.

Перед початком роботи з програмою необхідно задати основний тип лісового горючого матеріалу в області моделювання, а також підобласті довільної форми зі своїми типами лісового горючого матеріалу або їх відсутності.

Також існує програма "PSModel", яка розроблена в Сибірському державному технологічному університеті і призначена для прогнозування поширення ЛП, а також підготовки документів з результатами для подальшого аналізу фахівцем. Для розрахунку в даній програмі використовується ймовірнісно-множинна модель поширення ЛП, яка враховує випадковий характер її поширення.

2.2. Формат GRIB

GRIB (англ. GRId in Binary – сітка в двійковому вигляді) – є математичним форматом з даних. Він широко застосовується в метеорології для зберігання прогнозованих даних про погоду. Він був стандартизований комісією по основних системах Всесвітньої метеорологічної організації (ВМО).

Файл формату GRIB включає в себе інформацію про погоду в точках сітки, опис самих точок сітки, а також метадані для декодування файлу і інтерпретації метеорологічної інформації. Інтерпретація цих метаданих вимагає використання зовнішніх таблиць.

Національні або міжнародні метеорологічні центри зазвичай передають свої прогнозні дані в файлах формату GRIB. Вони знаходяться на порталі загальнодоступних даних Метео-Франса в форматі GRIB 2, в інформаційній системі ВМО через портал OpenWIS МетеоФранс, на сайті Європейського центру середньострокових прогнозів погоди (ECMWF), на сайті NOAA (Національне управління океанічних і атмосферних досліджень) і NCAR (Національний центр атмосферних досліджень).

Деякі метеорологічні організації також пропонують доступ до кліматологічних даних. Таким чином, архіви NOAA GFS (Глобальна система прогнозування) доступні онлайн.

Для роботи з файлами даного формату існують спеціальні програмні інтерфейси. Одним з них є GRIB-API від Європейського центру середньострокових прогнозів погоди (ECMWF). ECMWF GRIB-API – це програмний інтерфейс, доступний з програм C і Python, розроблений для кодування і декодування повідомлень GRIB.

Наступні інструменти командного рядка призначені для допомоги користувачам у всій інтерактивній та пакетній обробці даних GRIB:

- grib_compare – порівняти grib-повідомлення, які знаходяться в двох файлах;
- grib_copy – копіювати вміст файлів GRIB для друку значень деяких ключів;
- grib_count – друкувати загальну кількість повідомлень GRIB в даних файлах;
- grib_dump – дамп вмісту файлу GRIB в різних форматах;
- grib_get – отримати значення деяких ключів з файлу GRIB;
- grib_get_data – роздрукувати список значень широти, довготи, даних;
- grib_index_build – створити індексний файл для набору вхідних файлів GRIB;
- grib_info – роздрукувати інформацію про версії API GRIB, визначеннях по замовчуванню та зразках;
- grib_ls – список вмісту файлів GRIB для друку значень деяких ключій;
- grib_set – встановити пари ключ / значення у вхідному файлі GRIB і записувати кожне повідомлення в файл output_grib_file.

Для спрощення роботи з файлами формату GRIB в мові програмування Python, який використовувався при написанні програм розрахунку індексів і візуалізації результатів, є спеціальний модуль Pygrib. Він дає можливість проводити читання і запис файлів даного формату.

2.3. Мова програмування Python

Для реалізації програм розрахунку індексів і візуалізації результатів була обрана високорівнева мова програмування Python. Система модулів дозволяє логічно організувати код на Python. Групування коду в модулі значно полегшує процес написання та розуміння програми. Модуль Python – це просто файл, що містить код Python. Кожен модуль у Python може містити змінні, класи та функції. Крім того, в модулі може знаходитися виконуваний код.

Кожна програма може імпортувати модуль та отримати доступ до його класів, функцій та об’єктів.

Підключити модуль можна за допомогою інструкції import. Після ключового слова import вказується назва модуля.

```
>>> import os
```

Після імпортування модуля його назва стає змінною, через яку можна отримати доступ до атрибутів модуля. Наприклад, можна звернутися до константи e, яка розташована в модулі math:

```
>>> import math
>>> math.e
2.718281828459045
```

За допомогою інструкції from можна підключити певні атрибути модуля.

```
from <Назва модуля> import *
```

Коли імпортують модуль, інтерпретатор Python шукає цей модуль у таких місцях:

1. директорії, де знаходиться файл, в якому викликається команда імпорту.

2. якщо модуль не знайдено, Python шукає у кожній директорії, визначеній у консольній змінній PYTHONPATH.

3. якщо і там модуль не знайдено, Python перевіряє шлях, заданий за замовчуванням.

Шлях пошуку модулів збережений у системному модулі sys у змінній path. Змінна sys.path містить усі три вищеописані місця пошуку модулів. Для того, щоб отримати список усіх модулів, достатньо виконати команду:

```
help("modules")
```

Код Python може бути організований таким чином:

- перший рівень – це звичайні команди на Python.
- команди на Python можуть бути зібрані у функції.
- функції можуть бути частиною класу.
- класи можна визначити всередині модулів.
- модулі можуть складатися в пакети модулів.

Окремі файли-модулі з кодом Python можуть об’єднуватися в пакети модулів. Пакет – це директорія (папка), що містить кілька окремих файлів-скриптів.

2.4. Бібліотека NumPy

При розрахунку індексів пожежонебезпеки необхідно обробляти великі масиви даних і використовувати їх для розрахунків. Для цього використовується бібліотека мови Python – NumPy. NumPy – це розширення мови Python, що додає підтримку великих багатовимірних масивів та матриць, разом із великою бібліотекою високорівневих математичних функцій для операцій із цими масивами.

NumPy – це бібліотека мови Python, що додає підтримку великих багатовимірних масивів і матриць, разом із великою бібліотекою високорівневих математичних функцій для операцій із цими масивами.

Основним об'єктом NumPy є однорідний багатовимірний масив (numpy.ndarray). Це багатовимірний масив елементів (зазвичай чисел), одного типу. Найважливіші атрибути об’єктів ndarray:

ndarray.ndim – число вимірів (осі) масиву.

ndarray.shape – розміри масиву, його форма. Це кортеж натуральних чисел, що показує довжину масиву кожної осі. Для матриці з n рядків та m стовпів, shape буде (n,m). Число елементів кортежу shape дорівнює ndim.

ndarray.size – кількість елементів масиву, дорівнює добутку всіх елементів атрибута shape.

ndarray.dtype – об’єкт, що описує тип елементів масиву. Можна визначити dtype за допомогою стандартних типів даних Python. NumPy тут надає цілий букет можливостей як вбудованих, наприклад: bool, character, int8, int16, int32, int64, float8, float16, float32, float64, complex64, object, так і можливість визначити власні типи даних.

ndarray.itemsize – розмір кожного елемента масиву в байтах.

ndarray.data – буфер, що містить фактичні елементи масиву. Зазвичай не потрібно використовувати цей атрибут, тому що звертатися до елементів масиву найпростіше за допомогою індексів.

Один з найпростіших – створити масив зі звичайних списків або кортежів Python. Для цього використовується функція numpy.array(). array - функція, що створює об’єкт типу ndarray:

```
>>> import numpy as np
>>> a = np.array([1, 2, 3])
```

Функція array() трансформує вкладені послідовності багатомірні масиви. Тип елементів масиву залежить від типу елементів вихідної послідовності.

2.5. Бібліотека Matplotlib

Matplotlib – набір додаткових модулів (бібліотек) мови Python. Надає засоби для побудови найрізноманітніших 2D-графіків та діаграм даних. Переваги цієї бібліотеки – простота використання. Для побудови дуже складних діаграм досить декількох рядків коду. При цьому якість зображень більш ніж достатня для їх опублікування. Робота цього модуля зазвичай має на увазі так само використання модуля NumPy.

Для установки Matplotlib та NumPy використовують pip3 – систему керування пакетами, яка використовується для встановлення та керування програмними пакетами, написаними на Python.

```
pip3 install  numpy
pip3 install  matplotlib
```

Після установки потрібно перевірити працездатність. Для цього в консолі Python потібно ввести:

```
>>> import matplotlib as mpl
>>> print ('Current version on matplotlib library is', mpl.__version__)
import matplotlib.pyplot as plt
plt.plot([1,2,3])
plt.title('Пряма лінія')
<matplotlib.text.Text object at 0x03062430>
plt.show()
```

З пакету matplotlib імпортований модуль pyplot під назвою plt. Модуль pyplot містить функції створення діаграм та зміни властивостей їх елементів. Функція plot() будує прямокутні двовимірні діаграми (графіки) в координатах X - Y.

Якщо функції plot передано один аргумент (список [1, 2, 3]), вона розглядає його як сукупність значень, які відкладаються по осі Y, тоді по осі X їм відповідатиме автоматично згенерований набір чисел 0, 1, 2, …, N – 1, де N – число елементів у переданому списку.

Функція title() визначає заголовок діаграми, а функція show() виводить інтерактивне вікно діаграми.

Одне з основних завдань, яке виконує matplotlib – надання набору функцій та інструментів для представлення та управління Figure (основний об’єкт) разом із усіма внутрішніми об’єктами, з яких він складається. Але в matplotlib є інструменти для обробки подій, наприклад анімації. Завдяки їм ця бібліотека здатна створювати інтерактивні графіки на основі подій.

Архітектура matplotlib логічно розділена на три шари, розташовані на трьох рівнях. Кожен шар може взаємодіяти тільки з тим, що розташований під ним. Це шари:

- шар сценарію;
- художній шар;
- шар бекенду.

Шар Backend нижній на діаграмі з архітектурою всієї бібліотеки. Він містить всі API та набір класів, що відповідають за реалізацію графічних елементів на низькому рівні.

Середнім шаром є художній (artist). Усі елементи, що становлять графік, такі як назва, мітки осей, маркери тощо, є екземплярами цього об’єкта. Кожен їх грає свою роль в ієрархічній структурі. На цьому рівні часто доводиться мати справу з об’єктами, що займають високе становище у ієрархії: графік, система координат, осі. Тому важливо розуміти, яку роль вони відіграють.

Художні класи та пов’язані з ними функції (API matplotlib) підходять усім розробникам, особливо тим, хто працює із серверами веб-додатків або розробляє

графічні інтерфейси. Але для обчислень, зокрема для аналізу та візуалізації даних, найкраще підходить шар сценарію. Він включає інтерфейс pyplot.

Існують дві бібліотеки: pylab та pyplot. Pylab – це модуль, який встановлюється разом з matplotlib, а pyplot – внутрішній модуль matplotlib.

```
from pylab import *
import matplotlib.pyplot as plt
import numpy as np
```

Pylab поєднує функціональність pyplot із можливостями NumPy в одному просторі імен, тому окремо імпортувати NumPy не потрібно. Більш того, при імпорті pylab функції з pyplot та NumPy можна викликати без посилання на модуль (простір імен).

```
plot(x,y)
array([1,2,3,4])
plt.plot()
np.array([1,2,3,4]
```

Пакет pyplot пропонує класичний інтерфейс Python для програмування, має власний простір має та потребує окремого імпорту NumPy.

2.6. Файли CSV

Програмісти часто зіштовхуються із завданням обробки великих об’ємів структурованих даних. Python має вбудовану бібліотеку CSV, за допомогою якої програміст може працювати зі спеціальними CSV файлами. Це свого роду електронні таблиці.

Кожен рядок у файлі csv представляє собою окремий запис або рядок, що складається з окремих стовпців, розділених комами. Саме тому формат і називається Comma Separated Values. Але хоча формат CSV – це формат текстових файлів, Python для спрощення роботи з ним надає спеціальний вбудований модуль CSV.

CSV – це особливий вид файлу, який дозволяє структурувати великі об’єми даних. По суті, він є звичайним текстовим файлом, однак кожен новий елемент відокремлений від попереднього комою або іншим роздільником. Зазвичай кожен

запис починається з нового рядка. Дані CSV можна легко експортувати до електронних таблиць або баз даних.

У першому рядку вказується, яка інформація буде знаходитися в кожному стовпці. Крім того, після останнього елемента рядка кома не ставиться, інтерпретатор визначає кінець рядка за символом перенесення. Замість коми можна використовувати будь-який інший роздільник, тому під час читання CSV файлу потрібно заздалегідь знати, який символ використовується. CSV – це звичайний текстовий файл, який не підтримує символи в кодуваннях, що відрізняються від ASCII або Unicode.

Бібліотека csv є вбудованою, тому її не потрібно завантажувати, достатньо використовувати звичайний імпорт:

```
import csv
```

2.7. Створення графічного інтерфейсу на Python з допомогою бібліотеки Tkinter

У багатьох програмах виділяють додатки з графічним інтерфейсом користувача (GUI). При створенні таких програм стають важливими не лише алгоритми обробки даних, але й розробка для користувача програми зручного інтерфейсу, взаємодіючи з яким він визначатиме поведінку програми.

Користувач взаємодіє з програмою за допомогою різних кнопок, меню, вводячи інформацію в спеціальні поля, вибираючи певні значення у списках тощо (від англ. widget – "штучка"). Для мови програмування Python такі віджети включені до спеціальної бібліотеки tkinter. Якщо її імпортувати в програму (скрипт), можна користуватися її компонентами, створюючи графічний інтерфейс.

Послідовність кроків під час створення графічного додатка має особливості. Програма повинна виконувати своє основне призначення, бути зручною для користувача, реагувати на його дії. Потрібно пройти такі етапи при програмуванні, щоб отримати програму з GUI.

- імпорт бібліотеки;
- створення головного вікна;
- створення віджетів;

- встановлення їх властивостей;
- визначення подій;
- визначення обробників подій;
- розташування віджетів на головному вікні;
- відображення головного вікна.

Як і будь-який модуль, tkinter у Python можна імпортувати таким способом:

```
from tkinter import *.
```

У версії Python 3 ім’я модуля пишеться з малої літери (tkinter), хоча в попередніх версіях використовувалася велика (Tkinter).

У сучасних операційних системах будь-який додаток користувача укладено у вікно, яке можна назвати головним, у ньому розташовуються решта віджетів. Об’єкт вікна верхнього рівня створюється при зверненні до класу Tk модуля tkinter. Змінну, пов’язану з об’єктом-вікном, прийнято називати root. Другий рядок коду:

```
root = Tk()
```

Нехай у вікні потрібно розмістити кнопку. Кнопка створюється при зверненні до класу Button. Об’єкт кнопка зв’язується з якоюсь змінною. У класу Button є обов’язковий параметр – об’єкт, якому кнопка належить. Це є єдине вікно (root), воно і буде аргументом, що передається до класу при створенні об’єкта-кнопки:

```
but = Button(root)
```

Кнопка має багато властивостей: розмір, колір фону, написи та ін. Поки Для встановлення напису на кнопці потрібно використати властивість текст напису (text):

```
but["text"] = "Обчислити"
```

Дії (алгоритм), що відбуваються у тій чи іншій події, можуть бути досить складними. Тому часто їх оформляють як функції, а потім викликають, коли вони знадобляться. Вивід повідомлення на екран буде оформлений у вигляді функції printer:

```
def printer(event):
     print ("OK")
```

Функцію потрібно розміщувати на початку коду. Параметр event – це подія. Подія натискання лівою кнопкою миші виглядає так: <Button-1>. Потрібно зв’язати цю

подію з обробником (функцією printer). Для зв'язку призначено метод bind. Синтаксис зв'язування події з обробником виглядає так:

```
but.bind("<Button-1>",printer)
```

В будь-якому додатку віджети не розташовані у вікні абияк, а добре організовані, інтерфейс продуманий до дрібниць і зазвичай підпорядкований певним стандартам. Далі потрібно кнопку якось відобразити у вікні. Найпростіший спосіб – це використання методу pack.

```
but.pack()
```

Якщо не вставити цей рядок коду, то кнопка у вікні не появиться, хоча вона є в програмі. Головне вікно теж не з'явиться, доки не буде викликаний спеціальний метод mainloop. Цей рядок коду повинен бути завжди в кінці скрипта:

```
root.mainloop()
```

Коли створюють графічний інтерфейс програми, визначають, які віджети будуть використовувати і як вони будуть розташовані в додатку. Щоб організувати віджети у додатку, використовуються спеціальні невидимі об'єкти – менеджери розмітки. Існує два види віджетів:

- контейнери;
- дочірні віджети.

Контейнери поєднують віджети для формування розмітки. У Tkinter є три вбудовані менеджери розмітки: pack, grid і place.

- place – це менеджер геометрії, який розміщує віджети за допомогою абсолютного позиціонування;
- pack – це менеджер геометрії, який розміщує віджети по горизонталі та вертикалі;
- grid – це менеджер геометрії, який розміщує віджети у двомірній сітці.

ВИСНОВКИ ДО РОЗДІЛУ 2

У другому розділі приведено відомості про існуючі системи прогнозування та визначення параметрів поширення пожеж в лісових масивах. Описано принципи їх функціонування, технології їх розробки. Розглянуто засоби, з допомогою яких проектується дана інформаційна система. Приведено відомості про особливості проектування програмного продукту засобами Python, проектування графічного інтерфейсу засобами бібліотеки Tkinter, особливості візуалізації результатів з допомогою бібліотек NumPy та Matplotlib.

РОЗДІЛ 3. МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1. Математична модель визначення комплексних показників пожежної небезпеки

У службах лісового господарства України клас пожежної небезпеки визначається за класичним і модифікованим індексами пожежної небезпеки (ІПН).

Щоб обчислити класичний індекс пожежної небезпеки, потрібно мати інформацію про наступні метеорологічні параметри: температуру повітря (°С) і точку роси (°С) в 12-15 год за місцевим часом або в найближчий до нього термін синхронних метеорологічних спостережень, кількості опадів (мм), що випали за попередню добу; опади до 2,5 мм на добу не враховуються).

Обчислення індексу проводиться для окремого пункту і конкретного часу за такими формулами:

$$G = \sum_{i=1}^{n} t_i d_i; \tag{3.1}$$

$$d_i = T_i - r_i, \tag{3.2}$$

де G – класичний індекс для певного дня, T – температура повітря (°С) в 12–15 годинах дня по місцевому часу; r – точка роси в 12–15 годинах дня по місцевому часу (°С); d – дефіцит точки роси; n – число днів після останнього дощу.

Розрахунки за формулами (3.1)-(3.2) проводяться в пожежонебезпечний сезон (від сходу снігового покриву навесні до встановлення його восени) в дні без опадів, причому опади менше 2,5 мм за добу не враховуються.

Обчислення показника G ведеться на кожен день, поки в один з них не випаде сума опадів за добу понад 2,5 мм. Тоді накопичена за період сума стає рівною 0 і обчислення відновлюється з наступного дня без дощу.

Обчислення показника слід проводити в дні без опадів, поки їх сума за добу не перевищить 3 мм. Наприклад, в деяких регіонах через місцеві особливості при розрахунку за формулою (3.1) не враховуються опади менші за 3 мм.

Істотним недоліком класичного індексу є його різке скидання до 0 в тому випадку, якщо за добу випало опадів 2,5 або 3 мм. Показник стає вкрай неінформативним, так як в розрахунку ніяк не враховуються відомості про можливу посушливу погоду в період до опадів і наявності лісових горючих матеріалів.

Пропонується не обнуляти показник *G* у формулі (3.1) після кожного дня з опадами за добу 2,5 мм або 3 мм, а вести облік опадів за допомогою безрозмірного коефіцієнта *K*. Він залежить від суми опадів за поточний день:

$$g_i = K(R_i)\, g_i - 1 + T_i d_i. \qquad (3.3)$$

Залежно від суми опадів за добу коефіцієнт приймає відповідне значення, яке зазначено в табл. 3.1.

Табл. 3.1. Коефіцієнт для розрахунку модифікованого індекса

Опади *R*, мм	0	0,1-0,9	1-2	3-5	6-14	15-19	≥ 20
K(*R*)	1	0,8	0,6	0,4	0,2	0,1	0

Індекс, який вираховується за формулою (3.3), називається модифікованим індексом.

За чисельним значенням індексів (класичного і модифікованого) у відповідності зі шкалою (табл. 3.2) визначається клас пожежної небезпеки для певної території. Від класу пожежної небезпеки залежить те, які рішення приймуть фахівці в службах лісового господарства.

Табл. 3.2. Шкала пожежної небезпеки в лісі за умовами погоди

Клас пожежної небезпеки	**Величина показника пожежної небезпеки**	**Степінь пожежної небезпеки**
I	0-300	відсутня
II	301-1000	мала
III	1001-4000	середня
IV	4001-10 000	висока
V	>10 000	надзвичайна

Для кожного з класів пожежної небезпеки можуть бути введені поправки, які затверджені для даної місцевості і періоду пожежо-небезпечного сезону (весна, літо, осінь).

3.2. Математична модель прогнозування поширення низової лісової пожежі

Представлено математичну модель прогнозу поширення лісових пожеж, яка адаптована з врахуванням умов, переважаючим породам і характерним масштабам лісових масивів, особливостям клімату. Описано класифікацію та формування вхідних даних. Запропонована двовимірна модель є узагальненням одновимірної напівемпіричної моделі Ротермела, яка широко застосовується.

З багатьох природних і антропогенних факторів негативного впливу на стан і динаміку лісових екосистем домінуючими є пожежі, які завдають значної матеріальної та екологічної шкоди. Лісові пожежі поділяються на підземні (торф'яні), низові і верхові в залежності від того, які яруси лісу, ділянки території залучені в процес поширення вогню. Низові пожежі – найпоширеніші в природі. Більшість лісових пожеж починаються як низові, які можуть потім перейти в підземні або верхові в залежності від типу лісових горючих матеріалів (ЛГМ) і кліматичних умов.

Для науково обгрунтованих та успішних дій, спрямованих на попередження, прогноз розвитку та ліквідацію лісових пожеж, потрібна розробка відповідних математичних та комп'ютерних моделей, їх включення до складу систем підтримки прийняття рішень щодо запобігання надзвичайних ситуацій в лісах.

Прийнята така класифікація моделей лісових пожеж:

- теоретичні (математичні);
- емпіричні (статистичні);
- напівемпіричні.

Теоретичні математичні моделі базуються на законах газової динаміки, тепломасопереносу та інших фундаментальних законах фізики, хімії і записуються у вигляді системи диференціальних рівнянь. При верифікації таких моделей виникають певні складнощі, однак тільки вони, описуючи розвиток лісової пожежі на основі

загальних законів і з врахуванням великої кількості факторів, дозволяють відповідати на досить широке коло питань.

В емпіричних (статистичних) моделях систематизується ряд даних про швидкість розповсюдження лісової пожежі при зміні обраної кількості параметрів, визначаються коефіцієнти кореляції по кожній незалежній змінній. При такому підході не описується механізм явища, отримані співвідношення, строго кажучи, не можуть бути поширені за межі застосовності використаних статистичних даних, в рамках таких моделей дають прогноз з певною ймовірністю.

У напівемпіричних моделях для визначення швидкості поширення фронту пожежі залучаються загальні закони (збереження енергії, маси і кількості руху), які записуються у вигляді спрощених залежностей, а відповідні коефіцієнти підбираються шляхом узагальнення експериментальної інформації. Напівемпіричні моделі застосовні в ситуаціях, схожих на ті, при яких були зібрані і узагальнені експериментальні дані. Такі моделі значно простіші в верифікації в порівнянні з теоретичними, при цьому вони більш адекватні в порівнянні з емпіричними (статистичними) моделями.

Моделі згаданих класів створювалися, починаючи з другої половини минулого століття для прогнозу виникнення і розвитку лісових пожеж всіх типів, але найбільше уваги приділяли проблемам прогнозу динаміки низових лісових пожеж. Побудовано ряд моделей, що складаються з систем рівнянь, що включають параметри навколишнього середовища, рельєфу місцевості і кліматичних умов, лісового горючого матеріалу і дозволяють оцінити швидкість поширення пожежі, інтенсивність тепловиділення в зоні фронту горіння, геометрію вигорілої площі.

Найбільш поширена на сьогоднішній день напівемпірична модель поширення низових лісових пожеж була створена Ротермелом. Недолік даної математичної моделі полягає у відсутності явного обліку законів збереження маси і кількості руху, в результаті чого доводиться залучати різні емпіричні співвідношення, що знижує точність прогнозу швидкості розповсюдження лісової пожежі. Зокрема, в рамках цієї моделі не вдається визначати швидкість поширення верхових пожеж. Однак завдяки

простоті методики її програмні реалізації успішно впроваджені в більшості лісових служб Північної Америки та ряду європейських країн.

Дана математична модель на основі формул відповідає на питання, яка прогнозована швидкість поширення низової лісової пожежі в залежності від:

- швидкості вітру, типу росинності;
- вологовмісту рослинності;
- рельєфу місцевості.

Вхідні параметри для даної математичної моделі:

- ω_0, кг/м^2 – запас лісових горючих матеріалов (ЛГМ) на місцевості в абсолютно сухому стані;
- δ, м – глибина шару ЛГМ;
- σ, м$^{-1}$ – питома поверхня ЛГМ;
- h, Дж/кг – теплотворна здатність сухого горючого;
- ρ_p, кг/м^3 – густина горючого матеріалу в абсолютно сухому стані;
- M_f – вологовміст ЛГМ;
- S_T – масова частка всіх мінеральних речовин в ЛГМ;
- S_e – масова частка ефективних мінеральних речовин;
- U, м/с – швидкість вітру на середині висоти полум'я;
- tgφ – тангенс кута нахилу рельєфу;
- M_x – критичний вологовміст (мінімальне значення вологовмісту ЛГМ, при досягненні якого горіння припиняється).

В даній математичній моделі розраховується швидкість поширення пожежі (м/хв).

Фахівці з надзвичайних ситуацій обмежені в можливостях оперувати, а тим більше визначати на місцевості специфічні величини, такі як критичний вологовміст, питома поверхня шару горючих матеріалів і т.п. У їх розпорядженні є карти лісової ділянки, можливість візуального спостереження на місцевості, дані від метеослужб.

Існує зв'язок між поточним вологовмістом M_f конкретного типу ЛГМ і комплексним показником горимості лісу:

$$M_f = a \cdot \frac{1}{\Gamma} + b, \tag{3.4}$$

де a та b – емпіричні константи, які визначаються для конкретних типів ЛГМ; a – швидкість висушування; b – мінімальне значення вологовмісту даного типу ЛГМ в природніх умовах; Γ – індекс горимості, який визначається за формулою:

$$\Gamma = \sum_{j=1}^{n} T_j \left(T_j - T_{dj} \right). \tag{3.5}$$

Тут T_j – температура повітря в 12 годині в градусах Цельсія; T_{dj} – точка роси в 12 годин в градусах Цельсія; n – число сухих діб (діб з опадами менше 3 мм). В ті доби, коли температура повітря T_j чи різниця $\left(T_j - T_{dj} \right)$ від'ємна, наростання показника приймається рівним нулю.

В залежності від розрахованого значення Γ виділяють наступні класи пожежної небезпеки:

- I клас (Г до 300) – відсутність небезпеки;
- II клас (Г від 301 до 1000) – мала пожежна небезпека;
- III клас (Г від 1001 до 4000) – середня пожежна небезпека;
- IV клас (Г від 4001 до 10 000) – висока пожежна небезпека;
- V клас (Г більше 10 000) – надзвичайна небезпека.

Таким чином, можна обійтися без необхідності кожного разу на місцевості експериментально визначати M_f. Його можна виразити для конкретного переважаючого типу рослинності через відомий клас пожежної небезпеки.

Як і вологовміст M_f, більшість інших параметрів математичної моделі відносяться безпосередньо до характеристик ЛГМ. Якщо їх числові значення для типів рослинності, які найбільш часто зустрічаються в конкретному регіоні експериментально визначені, класифіковані, зведені в таблиці, збережені в базі даних, тоді можна оперативно використовувати готові набори в розрахунках і прогнозувати швидкість розповсюдження лісової пожежі в конкретних умовах.

Приймаючи такий підхід за основу, для розрахунку прогнозних положень фронту пожежі необхідно вибрати лише наступні параметри:

- тип рослинності (вибір варіанту з бази даних);
- клас пожежної небезпеки (I, II, III, IV, V);
- швидкість вітру та його напрям;
- тангенс кута схилу (врахування особливостей рельєфу, для рівнинної місцевості не враховується).

Вибір з бази даних типів рослинності здійснюється таким чином:

- 303 – лишайники, рідкий сосновий ліс, дерева молоді та середньовікові;
- 406 – зелені мохи, густий ялиновий ліс;
- 507 – опад хвої, рідкий ялиновий ліс, дерева молоді та середньовікові;
- 710 – чагарники, густий листяний ліс;
- 802 – відходи лісозаготівель, густий сосновий ліс;
- 901 – мінералізована смуга, ділянка без рослинності;
- 903 – водяна перешкода (ріка, озеро і т.п.).

Назва (опис) типу рослинності включає в себе переважаючий тип горючих матеріалів нижнього ярусу лісу (мохи, чагарники та ін.), переважаючу породу дерев, їх вік і степінь зімкнутості крон.

При моделюванні низової лісової пожежі потрібно вказувати тип верхнього ярусу лісу. Це необхідно з двох причин.

По-перше, в математичній моделі необхідно задавати швидкість вітру U на висоті середини полум’я. Як правило, відомі дані з метеостанції розраховуються на певній висоті (10-15 м) над рівнинною поверхнею. Яка буде швидкість вітру в нижньому ярусі лісу, залежить від густоти крон дерев, сортів дерев (чи є просвіт в нижньому ярусі лісу чи ні). Для оцінки швидкості вітру в математичній моделі враховуються два числових параметри:

- Z_d, м – середня висота деревостану;
- f, кг/м^3 – об’ємна щільність полога лісу.

По-друге, знання типу лісу і його віку дає додаткову інформацію по нижньому ярусі лісу. У стиглих і перестійних хвойних лісах в нижньому ярусі лісу неминуче накопичується шар певної товщини осаду голок; в листяних лісах – опади листя. Залежно від кліматичної смуги і степені зімкнутості крон можна класифікувати їх стан: вологий перегній (слабо горить) або добре висушений опад.

Класифікація типів рослинності і визначення їх характеристик – трудомістка процедура, однак реальна. Доказом цього служить альбом типів рослинності (з фотографіями) регіону. В математичній моделі для різних типів рослинності використані довідкові дані, вказано характерні значення.

Табл. 3.3. Перелік вхідних даних математичної моделі для деяких типів рослинності

Код	ω_0, кг/м^3	σ, м$^{-1}$	h, Дж/кг	ρ, кг/м^3	δ, м	M_x	S_T	Z^d, м	f, кг/м^3
303	1,7	2000	17991200	300	0,12	0,3	0,02	15	0,07
406	1,0	2500	19664800	300	0,10	0,5	0,02	15	0,24
507	0,3	6000	18828000	512	0,10	0,3	0,02	10	0,09
710	0,225	6560	18409600	512	0,60	0,2	0,02	20	0,08
802	0,9	4920	18409600	512	0,70	0,2	0,02	20	0,08
901	0	0	0	0	0	0	0	0	0

Для типу 901 в таблиці вказані нульові значення характеристик ЛГМ, що відповідає відсутності горючих речовин, пожежа безпосередньо по них не поширюється.

Ще однією особливістю математичної моделі є питання, яким чином розраховувати швидкості фронту пожежі в двовимірному випадку. Оригінальна модель є одновимірною, а результатом її застосування є число – швидкість поширення фронту пожежі в напрямку вітру ω_n.

У запропонованій моделі поширення фронту пожежі прогнозується на площі, розрахунки проводяться згідно з наведеними нижче рівняннями. Обгрунтуванням

такого алгоритму служать спостереження, що низова пожежа в однорідному середовищі поширюється еліпсом (рис. 3.1).

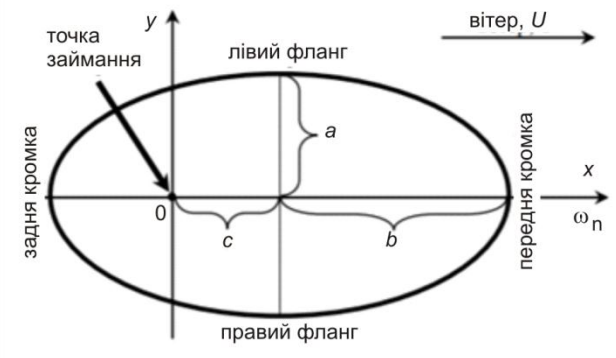

Рис. 3.1. Еліптична форма контуру низової лісової пожежі.

$$a = \frac{b}{LB}, \qquad b = \frac{\omega_n}{2}\ \frac{1 + HB}{HB}, \qquad c = b - \frac{\omega_n}{HB}. \tag{3.6}$$

$$LB = 0,936\exp(0,2566U) + 0,461\exp(-0,1548U) - 0,397,$$
$$HB = \frac{LB + \sqrt{LB^2 - 1}}{LB - \sqrt{LB^2 - 1}}. \tag{3.7}$$

де ω_n – швидкість поширення фронту пожара в напрямку вітру U.

ВИСНОВКИ ДО РОЗДІЛУ 3

Розроблена математична модель низової пожежі складає основу майбутньої інформаційної системи для прогнозу лісових пожеж, яка складатиметься з програмних модулів забезпечення інтерактивної роботи, прийому вихідних даних від метеостанцій, розрахунку прогнозних положень фронту пожежі, а також невід'ємною частиною є база даних наборів характеристик для всіх типових лісових горючих матеріалів.

Проведені дослідження підтверджують адекватність розробленої математичної моделі інформаційної системи та можливість для прогнозування ймовірності виникнення лісової пожежі та визначення параметрів поширення пожеж в лісових масивах.

РОЗДІЛ 4. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

4.1. Розробка алгоритму роботи програми для прогнозування динаміки лісових пожеж

Для створення інформаційної системи по моделюванню динаміки поширення лісової пожежі розроблено загальний алгоритм її роботи, а також алгоритм для визначення можливості переходу фронту пожежі через протипожежний бар'єр (рис. 4.1-4.2).

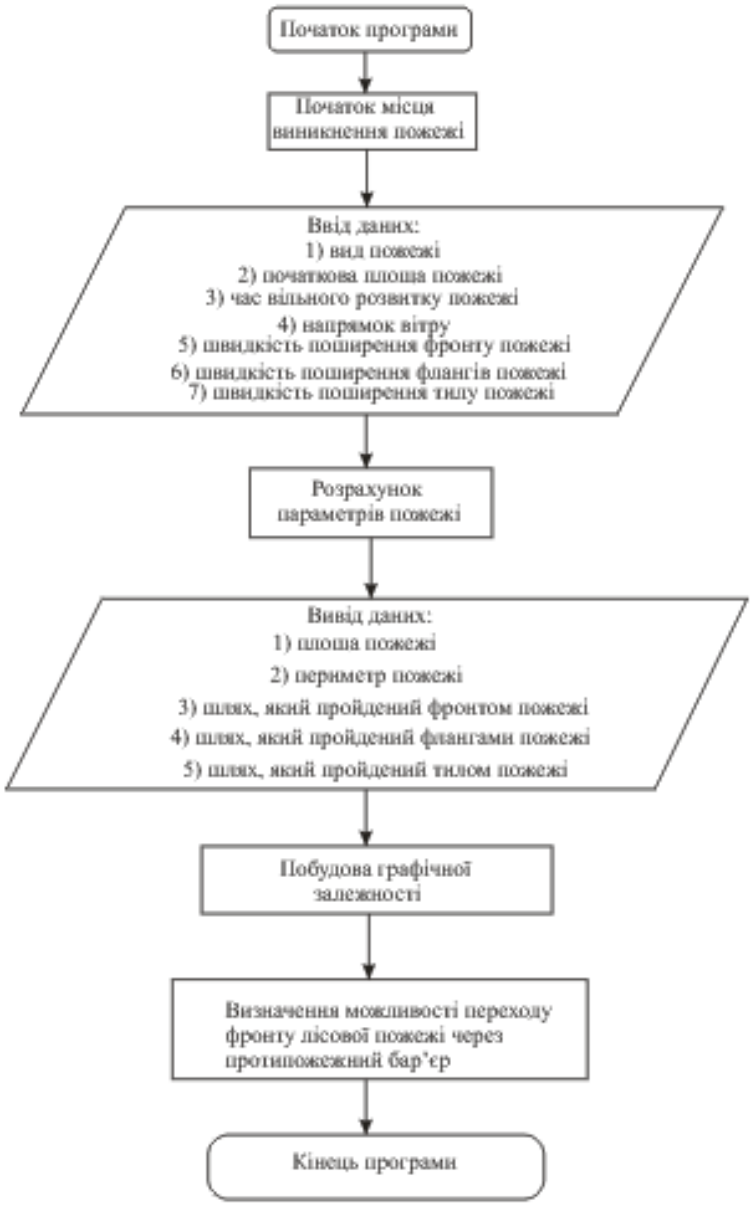

Рис. 4.1. Блок-схема алгоритму динаміки лісової пожежі.

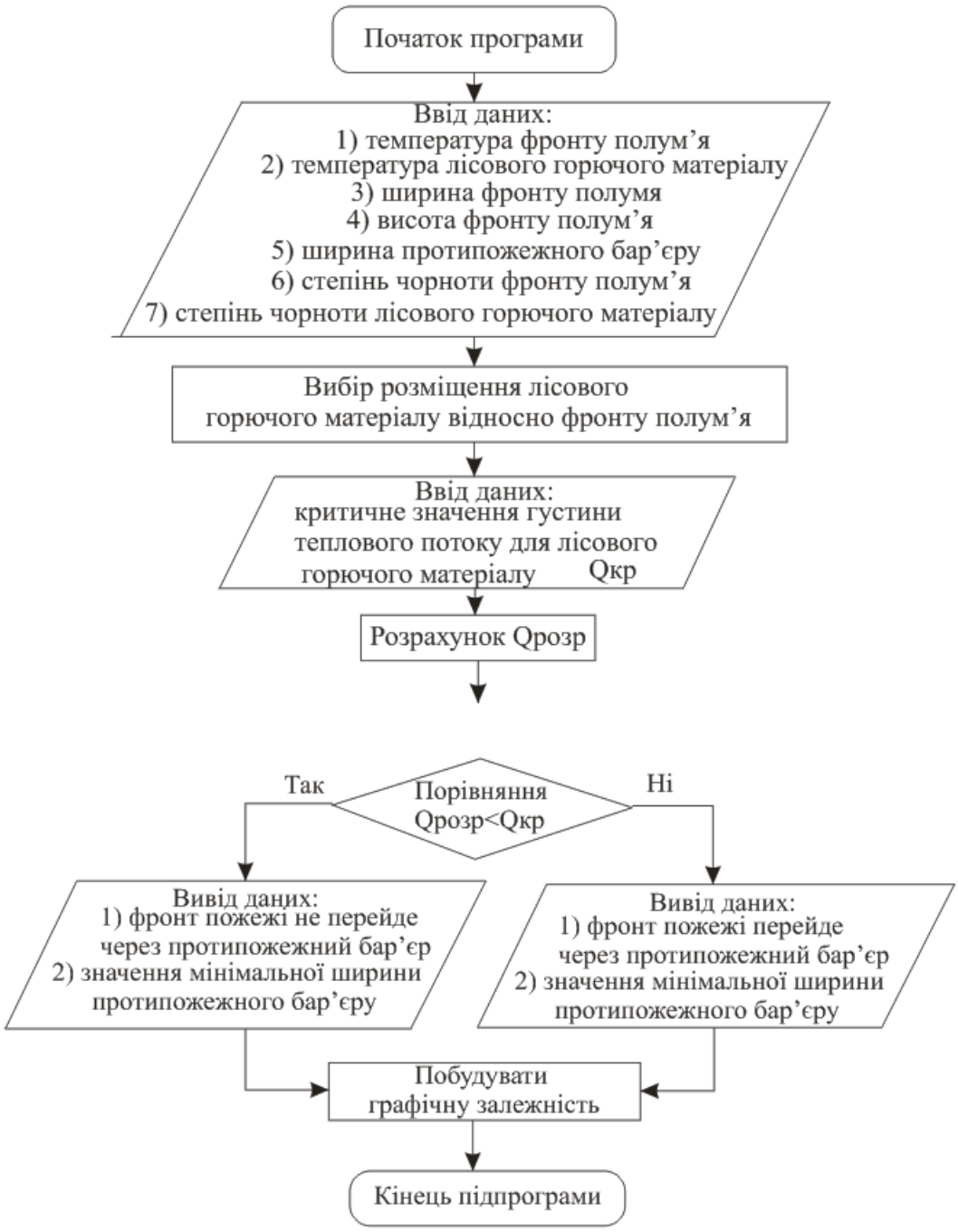

Рис. 4.2. Блок-схема алгоритму визначення можливості переходу фронту лісової пожежі через протипожежний бар'єр.

4.2. Розробка програми для прогнозування динаміки лісових пожеж

Розроблена по приведеному вище алгоритму засобами Python програма "Прогнозування динаміки лісових пожеж" дозволяє спрогнозувати:

- швидкість поширення фронту, флангів і тилу лісової пожежі;
- площу та периметр лісових пожеж;
- можливість переходу фронту полум'я лісової пожежі через протипожежний бар'єр.

Вхідними даними в інформаційній системі є:

- вид пожежі;
- напрямок вітру;
- температура повітря;
- точка роси;
- день спостереження;
- число днів після дощу.

4.3. Проектування інтерфейсу програми засобами Tkinter

Створимо файл fire.py і збережемо його. Для початку потрібно імпортувати модуль Tkinter. З його допомогою будемо проектувати графічний інтерфейс інформаційної системи прогнозування лісових пожеж:

```
from tkinter import *
```

Зірочка означає, що будуть імпортовані всі компоненти з модуля Tkinter. Після цього створюється вікно програми. Назвемо цю змінну Window1. Це буде об'єкт класу Tk():

```
Window1 = Tk()
```

Це основний клас, який створює всі об'єкти, які необхідні для графічного інтерфейсу програми. Як тільки ми об'єкт вікно створений, для нього потрібно задати деякі параметри. Щоб задати назву програми, потрібно для об'єкта Window1 використати метод title().

```
Window1.title("Прогнозування лісових пожеж")
```

Як аргумент для даного методу потрібно ввести назву програми.

Далі задають ширину і висоту вікна, а також відступ віл лівого верхнього кута екрану. За це відповідає метод geometry():

```
Window1.geometry("800x600+200+200")
```

Першим параметром йде ширина вікна, другим – його висота.

Заборонимо можливість зміни розміру вікна. За це відповідає метод rezizable(). Перший аргумент даного методу – це можливість зміни розміру вікна по ширині, другий – по висоті. Якщо задати обидва ці параметри True, то розміри вікна можна змінювати як по ширині, так і по висоті. Щоб користувач не міг змінювати розміри вікна, задають для цих параметрів значення False:

```
Window1.resizable(False, False)
```

Щоб добавити піктограму до програми, потрібно для об’єкта вікно потрібно викликати метод iconbitmap(). Як аргумент цього методу потрібно ввести назву файлу (піктограми):

```
Window1.iconbitmap("Пожежа.ico")
```

Піктограма має бути в форматі .ico та знаходитися у тій же папці, що й файл зі скриптом Fire.py. Якщо потрібен абсолютний шлях до піктограми, тоді слід ввести:

```
Window1.iconbitmap("C:\Пожежа.ico")
```

Щоб вікно програми появилося на екрані, потрібно запустити головний цикл графічної програми. Це є безконечний цикл:

```
Window1.mainloop()
```

Дана конструкція має знаходитися на останньому рядку скрипта Fire.py.

Ця функція викликає безконечний цикл вікна, тому вікно буде чекати любої взаємодії з користувачем, доки не буде закрите.

Щоб застосувати певний колір фону, для вікна виконується наступна конструкція:

```
Window1["bg"] = "green"
```

Для вікна задано зелений колір фону. Після створення головного вікна програми можна наповнювати його потрібними елементами інтерфейсу – віджетами.

Віджет Label необхідний для відображення текстової інформації. Для цього потрібно створити змінну lbl1 і використати клас Label. Перший аргумент служить, де даний віджет потрібно розмістити. Це головне вікно Window1. Наступний атрибут text: інформація, яка буде відображена екрані:

```
label1 = Label(Window1, text=" Оцінка стану пожежної небезпеки")
```

Після створення цього віджета його потрібно розмістити у вікні. За це відповідають спецальні методи. Це так звані менеджери геометрії. Для цього потрібно для змінної label1 викликати метод pack():

```
label1.pack()
```

Крім цього методу існують ще інші методи розміщення віджетів у вікні програми (gride, place).

```
label1 = Label(Window1, text="Оцінка стану пожежної небезпеки",
bg="red", fg="blue")
```

За колір напису відповідає атрибут bg, за колір тексту – fg.

Для атрибуту font кортежом потрібно передати три значення: шрифт, розмір шрифта, стиль шрифта:

```
label1 = Label(Window1, text="Оцінка стану пожежної небезпеки",
bg="red", fg="blue", font="Arial", 20, "bold")
```

Можна впливати на розміри цього віджета. Для цього задамо йому відступи по горизонталі та по вертикалі, а також задати ширину та висоту:

```
padx=10, pady=20.
```

```
width=20, height=10
```

В даному параметрі anchor потрібно вказати сторону світу: n (верх), s (низ), w (ліва), e (права), nw (лівий верхній кут віджета), ne (правий верхній), sw (лівий нижній), se (правий нижній). По замовчуванню стоїть параметр center:

```
anchor="n"
```

Для створення кнопки використовується клас Button. Перший атрибут – це об’єкт, в якому він буде знаходитися (Window1). Другий атрибут text – це напис на кнопці.

Третій атрибут command – це подія, яка має відбуватися при натисканні на кнопці. Буде запускатися обробка функції `start_window_1`.

```
    button1 = Button(Window1, text = "Обчислити",
command = start_window_1)
    button1.place(x=200, y=200)
```

Кнопка буде розмішена з допомогою методу place() на віддалі в 200 пікселів відносно лівого верхнього краю головного вікна Window1.

Перед тим потрібно цю функцію `start_window_1` створити і розмістити вгорі скрипта:

```
    def start_window_1():
        new_window_1 = Toplevel(root)
        new_window_1.title(Вікно 2)
        new_window_1.resizable(0, 0)
```

Віджет текстове поле призначений для вводу чи виводу текстової інформації чи результатів обчислень:

```
    text1 = Entry(Window1, width = 30, fg="blue",
font=("Verdana", 9), justify = CENTER, relief=SUNKEN, bd=3)
    text1.pack()
    text2 = Entry(Window1, width=20, bg="violet"
    text2.pack()
```

Для того, щоб ефективно використовувати поле вводу, потрібно розуміти, як з нього отримувати дані.

```
    e = text1.get()
```

text1 – це ім’я поля вводу, get – команда отримати. Тобто інформація, яку ввели у поле вводу з іменем text1, присвоюється змінній e.

```
    but=Button(Window1, text="Обчислити", command=mycom)
    but.pack()
    def mycom():
        e = text1.get()
```

```
t1.config(text=e)
```

Всі елементи інтерфейсу (написи, текстові поля, командні кнопки, випадаючі списки, радіоперемикачі) розміщені у вікні програми за допомогою метода геометрії place(). Цей метод дозволяє розміщувати віджети в абсолютних значеннях (у пікселях, за координатами X та Y) та у відносних значеннях (у процентах від розміру вікна, в якому його розміщують).

Метод place() дозволяє точніше налаштувати параметри позиціонування. Він приймає такі параметри:

- x та y: встановлюють зміщення елемента по горизонталі та вертикалі в пікселях відповідно щодо верхнього лівого кута контейнера;
- height і width: встановлюють відповідно висоту та ширину елемента в пікселях;
- bordermode: визначає формат межі елемента, приймає значення INSIDE (за замовчуванням) та OUTSIDE;
- anchor: визначає опції розтягування елемента. Може приймати такі значення: n, e, s, w, ne, nw, se, sw, c, які є скороченнями від Noth (північ – вгору), South (південь – низ), East (схід – права сторона), West (захід) – ліва сторона) та Center (по центру). Наприклад, значення nw вказує на те, що даний віджет буде розміщений у верхньому лівому куті екрану:

```
btn.place(anchor="nw", height=30, width=130, bordermode=OUTSIDE)
```

4.4. Результати роботи програми прогнозування параметрів лісової пожежі.

На рис.4.3-4.5 приведений інтерфейс інформаційної системи для прогнозування параметрів низової лісової пожежі. Програмний код приведено відповідно у додатках.

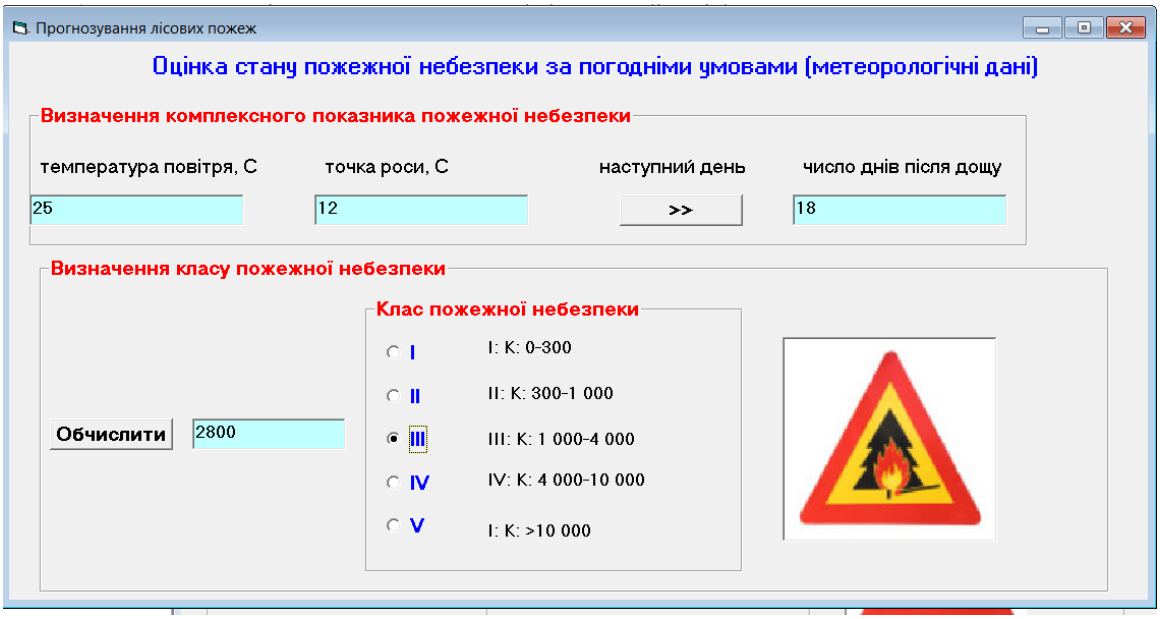

Рис. 4.3. Оцінка стану пожежної небезпеки за погодніми умовами (метеорологічні дані).

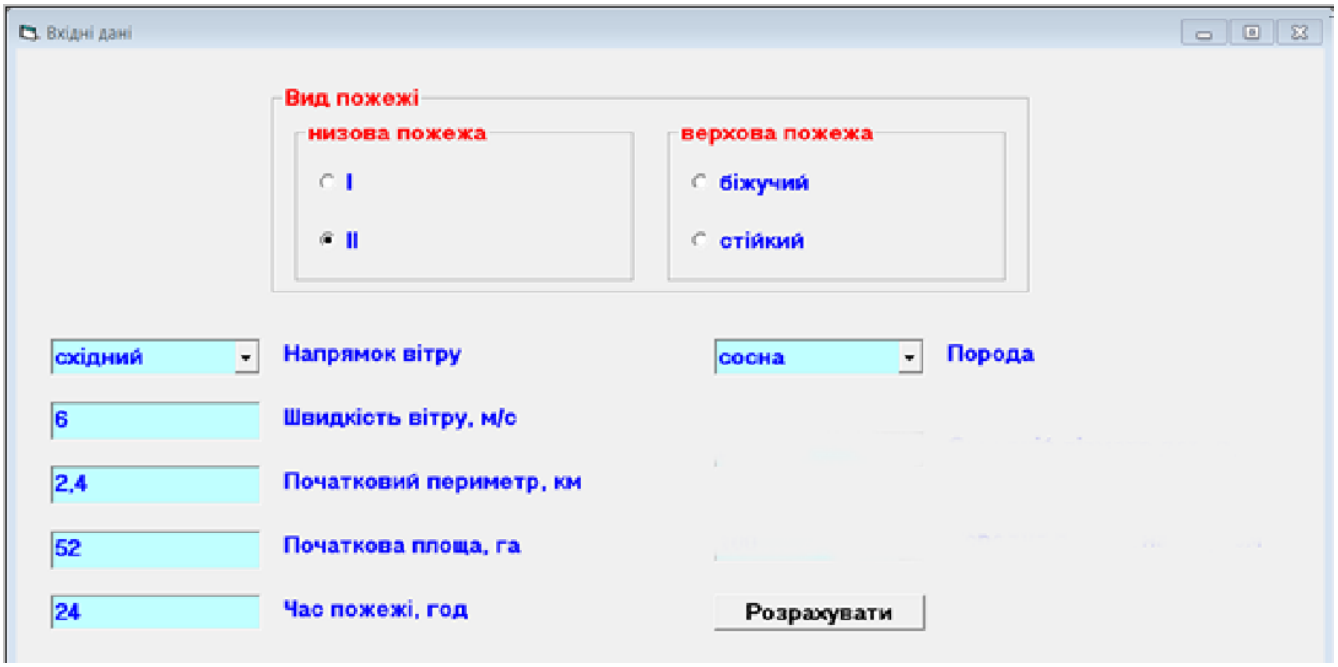

Рис. 4.4. Вхідні дані додатку для прогнозування параметрів лісових пожеж.

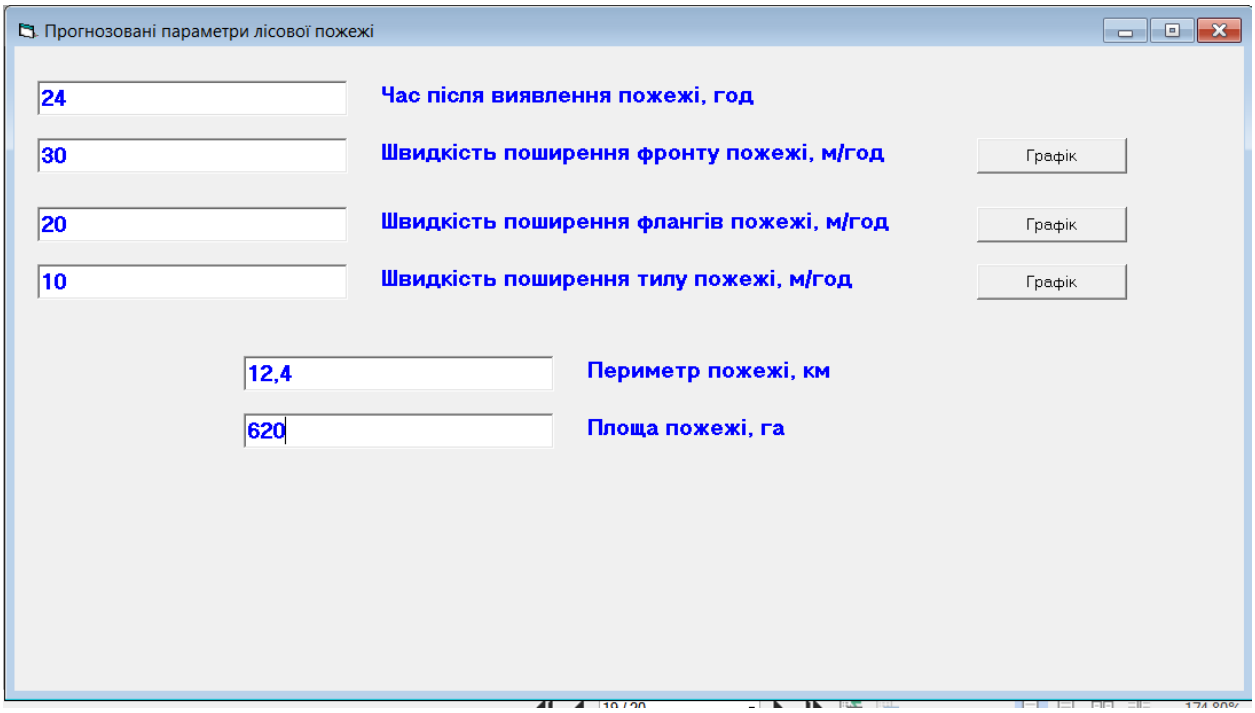

Рис. 4.5. Результати прогнозування параметрів лісових пожеж.

Результати роботи програми представлені на рис. 4.6-4.8.

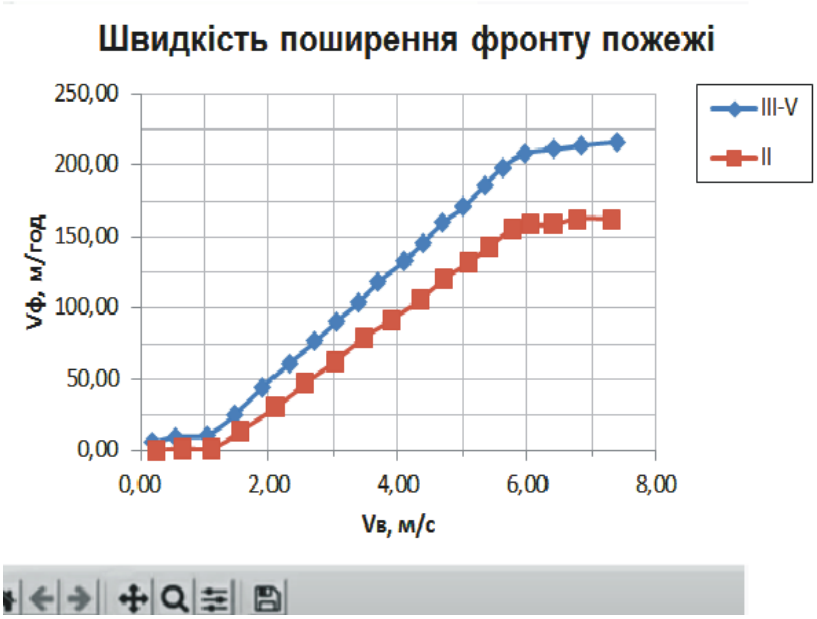

Рис. 4.6. Залежність швидкості поширення фронту $V_ф$ низової пожежі від швидкості вітру $V_в$.

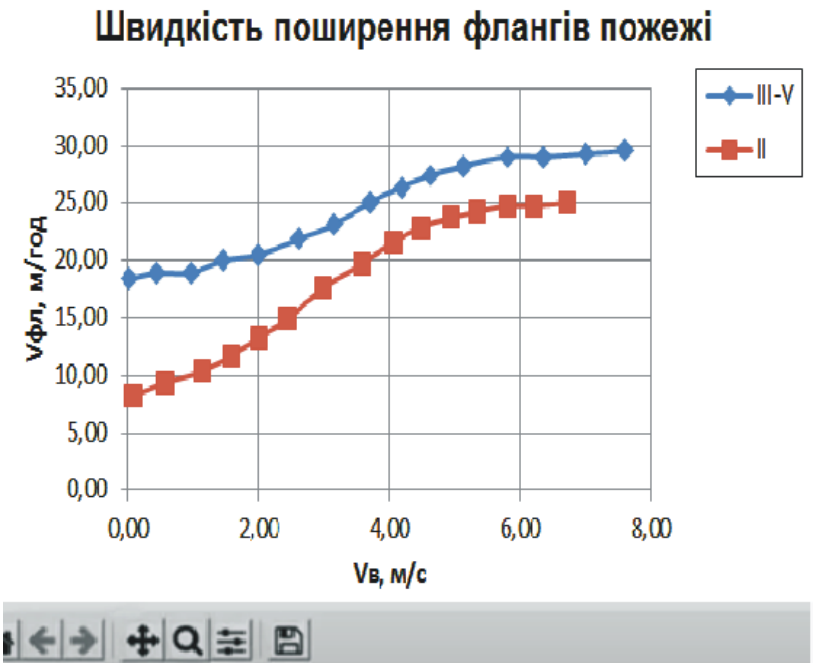

Рис. 4.7. Залежність швидкості поширення флангів $V_{фл}$ низової пожежі від швидкості вітру $V_в$.

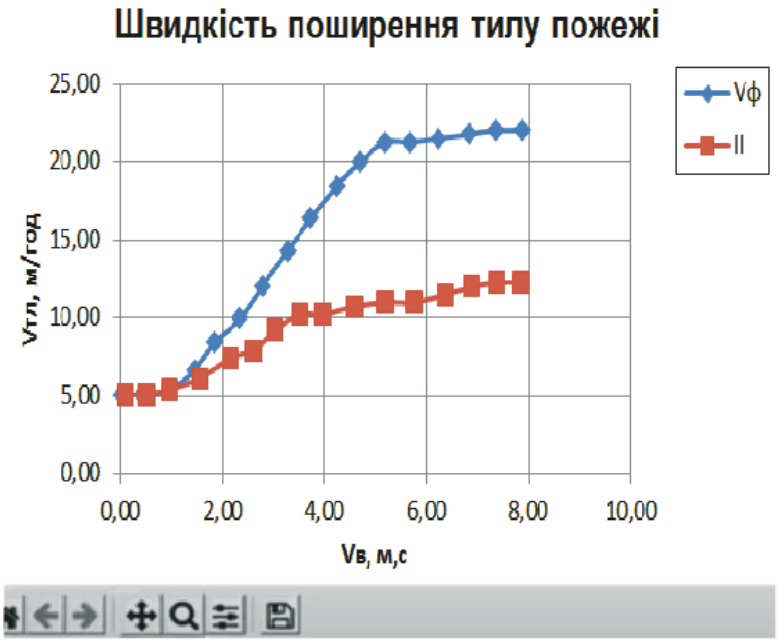

Рис.4.8. Залежність швидкості поширення тилу $V_{тл}$ низової пожежі від швидкості вітру $V_в$.

ВИСНОВКИ ДО РОЗДІЛУ 4

Розроблено та реалізовано інформаційну систему оцінки пожежної небезпеки в лісі, в якій враховуються погодні умови, стан лісової рослинності та можливість прогнозування появи пожежі на території, яка охороняється. Реалізовано програмну модель прогнозування лісових пожеж, в рамках даної моделі проведено прогноз динаміки поширення лісових пожеж.

Результати роботи даної системи прогнозування лісових пожеж можуть використовуватися при оптимізації маршрутів повітряного та наземного патрулювання. З врахуванням особливостей лісових територій дана програма може бути впроваджена в структурні підрозділи системи охорони лісових територій.

РОЗДІЛ 5. РОЗРОБЛЕННЯ СТАРТАП ПРОЕКТУ

5.1. Опис проекту інформаційної системи

Перш ніж почати просувати свій проект як стартап, ви можете зробити кілька кроків. Перший – створити інформаційну карту проекту. На цій карті показані характеристики проекту та приблизний бюджет. Сюди входять зарплати розробника, які розраховані на певні цикли розробки, та оренда обладнання. Інформаційна карта наведена в таблиці. 5.1.

Табл. 5.1. Карта інформаційної системи

Назва номінації	Python програма
Назва проекту	Розроблення інформаційної системи прогнозування лісових пожеж
Назва ВНЗ, факультету, спеціальності	НЛТУ, кафедра інформаційних технологій, 122 «Комп'ютерні науки»
Прізвище, ім'я, по-батькові	Віхоть Олег Ярославович
Цілі і задачі проекту	Ціль проекту – розробка та реалізація інформаційної системи для оцінки пожежної небезпеки в лісі, в якій будуть враховуватися погодні умови, стан лісової рослинності та можливість появи джерел вогню на території, яка охороняється. Задачі проекту: • проаналізувати існуючі системи прогнозування лісових пожеж; • розробити інформаційну систему прогнозування лісових пожеж; • реалізувати програмну модель

	прогнозування лісових пожеж; • в рамках програмної моделі провести дослідження динаміки поширення лісових пожеж.
Короткий зміст проекту	Розроблено та реалізовано систему оцінки пожежної небезпеки в лісі, яка враховуватиме погодні умови, стан лісової рослинності та можливість появи джерел вогню на території, яка охороняється. Реалізовано програмну модель прогнозування лісових пожеж, в рамках даної моделі проведено прогноз динаміки поширення лісових пожеж. Результати роботи даної системи прогнозування можуть використовуватися при оптимізації маршрутів повітряного та наземного патрулювання. З врахуванням особливостей лісових територій дана програма може бути впроваджена в структурні підрозділи системи охорони лісових територій.
Терміни виконання проекту	10 місяців
Бюджет проекту	45 000 грн.

5.2. Стратегія проекту

У технічній сфері можна використовувати такі методи:

1. Стратегія лідерства. Це дозволяє отримати переваги на ринку, маючи передові технології. Ця стратегія пов’язана з високою вартістю досліджень, розробок і робіт.

2. Стратегія лідерства. Якщо інші не впровадять інновації, компанія не буде впроваджувати інновації. Цей метод дозволяє врахувати помилки керівника і уникнути їх.

3. Стратегія моделювання. Компанія використовує існуючі технології. Така стратегія знижує витрати на дослідження та рекламу, тому ціна товару буде дуже низькою.

Вибрано стратегію моделювання в дипломній роботі. Вона заснована на технології, яка відповідає вимогам, властивим проекту. Компанії, що впроваджують стратегії моделювання, не будуть нести витрати на дослідження, тому вони можуть значно знизити витрати та збільшити дохід від продажів. Основним напрямком використання цієї стратегії є швидкий розвиток технологій та введення в експлуатацію товарів.

Стратегія моделювання вимагає швидкозростаючого ринку. Імітатори досягають своїх цілей тим, що забирають клієнтів у авторів нового винаходу чи послуги своїм обслуговуванням. Творча імітація використовує вже існуючий попит, а не створює новий. Проте дана стратегія також не вільна від ризику, причому значного. Загрозою може бути невірна оцінка ситуації та реалізація того, що не має перспектив з точки зору ринкових можливостей.

Доступні ринкові стратегії наступні:

1. Стратегія лідера. Відповідно до ступеня сформованості ринку та характеру конкуренції провідні компанії обирають одну з наступних стратегій:

• розширити первинний попит;

• наступальна або оборонна стратегія;

• диверсифіковані або демаркетингові програми.

2. Стратегія виклику лідера – цю стратегію часто обирають компанії, які не є лідерами, мають невелику частку ринку, але хочуть розширити сферу свого впливу. Ці компанії мають можливість прийняти одне з двох стратегічних рішень: стати на лідера в боротьбі за ринок або слідувати за лідером. Вирішити протистояти лідеру досить ризиковано. Його реалізація вимагає чималих фінансових витрат. Якщо від цієї стратегії відмовитися, компанія може надовго опинитися в позиції аутсайдерів.

3. Стратегія імітації лідерства. Лідерські організації – це компанії з невеликою часткою ринку, які обирають адаптивну поведінку, усвідомлюють своє місце в ній і слідують за лідерами. Основною перевагою цієї стратегії є економія фінансових резервів, що пов’язано з витратами на розширення ринку, інновації та збереження домінуючого становища.

4. Стратегії отримання конкурентної переваги. Під час використання компанія вибирає один або кілька сегментів ринку як цільовий ринок. Основна відмінність – менший розмір сегмента. Цей вид конкурентної стратегії походить від базової стратегії - концентрації компанії.

Найкращою стратегією для цього проекту буде отримання конкурентної переваги, оскільки дана інформаційна система містить функції, подібні до продуктів конкурентів, але з істотними відмінностями.

5.3. Розробка програми стартап проекту

Табл. 5.2. Основні переваги та концепції інформаційної системи

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами
1	моделювання поширення лісових пожеж	створення власних функціональних одиниць	гнучкість та свобода для кінцевого споживача
2	визначення параметрів поширення лісових пожарів	можливість підключення датчиків оповіщення про початок пожежі	можливість використання даних з безпілотних літальних апаратів
3	зручний інтерфейс	інтерфейс, який містить потрібні дані для можливості прогнозування виникнення лісової пожежі	можливість прогнозування ймовірності виникнення пожежі в лісі

Табл. 5.3. Опис трьох рівнів інформаційної системи моделювання поширення лісових пожеж

рівні товару	**сутність та її складові**
1. Програмний продукт за задумом	моделювання для прогнозування ймоірності виникнення лісової пожежі
2. Програмний продукт, який має бути реально виконаний	програмний продукт для системи попереднього оповіщення про небезпеку виникнення пожежі
3. Підкріплення	служба для підтримки системи прийняття рішень в надзвичайній ситуації за допомогою даної інформаційної системи

Захист від несанкціонованого копіювання може бути здійснений шляхом вводу серійного номену при інсталяції даної інформаційної системи.

Табл. 5.4. Визначення меж для встановлення ціни на інформаційну систему

№ п/п	рівень цін на товари замінники	рівень цін на товари-аналоги	рівень доходів цільової групи споживачів	верхня та нижня межі встановлення ціни на товар/послугу
1	наперед не задано	наперед не задано	200$+	200/100 $

ВИСНОВКИ ДО РОЗДІЛУ 5

Серед багатьох важливих проблем охорони та відтворення лісових ресурсів є проблема боротьби з лісовими пожежами, а саме оцінка та прогноз пожежної небезпеки в лісі, які характеризують потенціальну загрозу лісовим масивам, їх розвиток і нанесення збитків лісовим ресурсам.

Розроблено та реалізовано інформаційну систему для оцінки пожежної небезпеки в лісі, в якій враховуються погодні умови, стан лісової рослинності та можливість появи джерел вогню на території, яка охороняється. Ця інформаційна система по прогнозуванню ймовірності виникнення пожежі в лісі може бути реалізована на практиці. Її перевагою є те, що альтернативних програмних продуктів немає, а якщо є, то дуже мало. При використанні реклами та оголошень для просування програмного продукту можна досягти успіху та отримати пристойний дохід.

Дана методика дозволяє отримати і спрогнозувати ймовірність виникнення і визначити параметри поширення лісової пожежі. А також визначати збитки, які завдала лісова пожежа лісовим масивам у грошовому еквіваленті. Колектив, який приймає рішення по попередженню поширення пожежі, виходить із рекомендацій, що випливають з існуючих математичних моделей поширення лісових пожеж.

ВИСНОВКИ

В дипломній роботі спроектовано інформаційну систему прогнозування лісових пожеж. Розроблено методику розрахунку розподілу джерел вогню, яка базується на статистиці лісових пожеж. Дана методика оцінки пожежної небезпеки дозволяє диференційовано враховувати особливості погоди, насаджень та розподіл джерел вогню на території, яка охороняється.

В першому розділі проаналізовано предметну область, засоби та технології проектування інформаційної системи прогнозування виникнення пожеж в лісових масивах. Представлено класифікацію лісових пожеж, а також лісових горючих матеріалів. Розглянуто причини виникнення лісових пожеж, а також вказано наслідки, до яких вони можуть призвести. З допомогою цієї інформаційної системи можна буде моделювати динаміку поширення пожеж в лісових масивах, а також проводити оцінку їх наслідків.

У другому розділі приведено відомості про існуючі системи прогнозування та визначення параметрів поширення лісових пожеж. Описано принципи їх функціонування та технології розробки. Розглянуто засоби, з допомогою яких проектується дана інформаційна система. Приведено відомості про особливості проектування програмного продукту засобами Python, проектування графічного інтерфейсу засобами бібліотеки Tkinter, особливості візуалізації результатів з допомогою бібліотек NumPy та Matplotlib.

У третьому розділі розроблена математична модель низової пожежі, яка складає основу майбутньої інформаційної системи прогнозу лісових пожеж. Вона складатиметься з програмних модулів забезпечення інтерактивної роботи, прийому вихідних даних від метеостанцій, розрахунку прогнозних положень фронту пожежі, а також невід'ємною частиною є база даних наборів характеристик для всіх типових лісових горючих матеріалів.

В четвертому розділі розроблено та реалізовано систему оцінки пожежної небезпеки в лісі, в якій враховуються погодні умови, стан лісової рослинності та можливість прогнозування появи пожежі на території, яка охороняється. Реалізовано

програмну модель прогнозування лісових пожеж, в рамках даної моделі проведено прогноз динаміки поширення лісових пожеж.

В п’ятому розділі розроблено стартап проекту, який дозволяє отримати і спрогнозувати ймовірність виникнення і визначити параметри поширення лісової пожежі. А також визначати збитки, які завдала лісова пожежа лісовим масивам у грошовому еквіваленті. Колектив, який приймає рішення по попередженню поширення пожежі, виходить із рекомендацій, що випливають з існуючих математичних моделей поширення лісових пожеж.

Отримані результати підтверджують адекватність розробленої методики, обгрунтовують можливість їх використання при оптимізації протипожежних заходів, зокрема при проектуванні маршрутів повітряного та наземного патрулювання. З врахуванням особливостей лісових територій дана програма може бути впроваджена в структурні підрозділи системи охорони лісових територій.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Софронов М.А., Вакуров А.Д. Огонь в лесу. Новосибирск: Наука. 1981. – 128 С.

2. Кузнецов Г.В., Барановский Н.В. Прогноз возникновения лесных пожаров и их экологических последствий. Новосибирск: Изд-во СО РАН, 2009. – 301 С.

3. Нестеров В.Г. Горимость леса и методы ее определения. М.; Л.: Гослесбумиздат, 1949. – 76 С.

4. Барановский Н.В. Оценка вероятности возникновения лесных пожаров с учетом метеоусловий, антропогенной нагрузки и грозовой активности // Пожарная безопасность. 2009. № 1. – С. 93-99.

5. Барановский Н.В. Модель прогноза и мониторинга лесной пожарной опасности // Экология и промышленность России. 2008, № 9. – С. 59-61.

6. Курбатский Н. П. Методические указания для опытной разработки местных шкал пожарной опасности. Л.: ЦНИИЛХ, 1954. – 33 с.

7. Матвеев П. М., Матвеев А. М. Лесная пирология. Красноярск: СибГТУ, 2002. – 317 с.

8. Наставление по определению пожарной опасности погоды и горимости лесов. М.; Л.: Гос. лесотехн. изд-во, 1946. – 20 с.

9. Нестеров В. Г. Горимость леса и методы ее определения. М.: Гослесбумиздат, 1949. – 76 с.

10. Софронов М. А., Гольдаммер И. А., Волокитина А. В., Софронова Т. М. Пожарная опасность в природных условиях. Красноярск: ИЛ СО РАН, 2005. – 330 с.

11. Софронова Т. М., Волокитина А. В., Першин К. С. Автоматизированное составление усовершенствованных местных шкал пожарной опасности // Вестник КрасГАУ. 2013. № 3. – С. 157–163.

12. Телицын Г. П. Определение пожарной опасности на лесной территории: методические рекомендации. Хабаровск: ДальНИИЛХ, 1989. – 24 с.

13. Кац А.Л., Гусев В.Л., Шабунина Т.А. Методические указания по прогнозированию пожарной опасности в лесах по условиям погоды. – М.: Гидрометеоиздат, 1975. – 16 с.

14. Гришин А.М. Физика лесных пожаров. Томск: Изд-во Томского ун-та, 1994. – 218 с.

15. Гришин А.М., Барановский Н.В., Лобода Е.Л., Фильков А.И. Проведение работ по программно-информационному обеспечению прогноза лесопожарной опасности для Томской области: Отчет ТГУ, Центр образования и исследований по механике реагирующих сред и экологии (ЦОИМЭК ТГУ). Томск, 2001. – 221 с.

16. Сверлова Л.И. Метод оценки пожарной опасности в лесах по условиям погоды с учетом поясов атмосферной засушливости и сезонов года. Хабаровск. 2000. 47 с.

17. Python [Електронний ресурс] URL:
https://ru.wikipedia.org/wiki/Python

18. Matplotlib [Електронний ресурс] URL:
https://ru.wikipedia.org/wiki/Matplotlib

19. Наукова графіка в Python [Електронний ресурс] URL:
https://nbviewer.jupyter.org/github/whitehorn/Scientific_graphics_in_python/blob/master/P1%20Chapter%201%20Pyplot.ipynb

ДОДАТКИ

ДОДАТОК А. Прогнозування лісових пожеж

Fire.py

```
from tkinter import *
def start_window_1():
    new_window_1 = Toplevel(root)
    new_window_1.title("Вхідні дані")
    new_window_1.resizable(0, 0)
def start_window_2():
    new_window_2 = Toplevel(root)
    new_window_2.title("Прогнозовані параметри лісової
    пожежі")
    new_window_2.resizable(0, 0)
Window1 = Tk()
Window1.title("Прогнозування лісових пожеж")
Window1.geometry("800x600")
Window1.resizable(False, False)
Window1.iconbitmap("Пожежа.ico")
Window1["bg"] = "green"
    lbl1 = Label(Window1, text="Оцінка стану пожежної небезпеки за
погодніми умовами (метеорологічні дані)", bg="gray", fg="blue",
width=20, height=10, font="Arial", 20,
"bold")
    lbl1.place(x=50, y=10)
     lbl2 = Label(Window1, text="Визначення комплексного показника
пожежної небезпеки", bg="gray", fg="red", font="Arial", 20, "bold",
width=20, height=10)
    lbl2.place(x=20, y=50)
    lbl3 = Label(Window1, text="температура повітря, С", bg="gray",
fg="black", font="Arial", 20, "bold", width=20, height=10)
    lbl3.place(x=20, y=80)
    lbl4 = Label(Window1, text="точка роси, С", bg="gray", fg="black",
font="Arial", 20, "bold" padx=10, pady=20, width=20, height=10)
    lbl4.place(x=100, y=80)
```

```
    lbl5 = Label(Window1, text="наступний день", bg="gray", fg="black",
font="Arial", 20, "bold" padx=10, pady=20, width=20, height=10)
    lbl5.place(x=200, y=80)
    lbl6 = Label(Window1, text="число днів після дощу", bg="gray",
fg="black", font="Arial", 20, "bold" padx=10, pady=20, width=20,
height=10)
    lbl6.place(x=300, y=80)
    txt1 = Entry(Window1, width = 30, bg="#00E0EE" fg="blue",
font=("Verdana", 9), justify = CENTER, relief=SUNKEN, bd=3)
    txt1.place(x=20, y=150)
    txt2 = Entry(Window1, width = 30, bg="#00E0EE" fg="blue",
font=("Verdana", 9), justify = CENTER, relief=SUNKEN, bd=3)
    txt2.place(x=100, y=150)
    btn1 = Button(Window1, text = "<<", command = start_window_2)
    btn1.place(x=200, y=150)
    txt3 = Entry(Window1, width = 30, bg="#00E0EE", fg="blue",
font=("Verdana", 9), justify = CENTER, relief=SUNKEN, bd=3)
    txt3.place(x=300, y=150)
    lbl17 = Label(Window1, text="Визначення класу пожежної небезпеки",
bg="gray", fg="black", font="Arial", 20, "bold" padx=10, pady=20,
width=20, height=10)
    lbl17.place(x=50, y=200)
    btn2 = Button(Window1, text = "Обчислити", command = start_window_2)
    btn2.place(x=50, y=400)
    txt4 = Entry(Window1, width = 30, bg="#00E0EE", fg="blue",
font=("Verdana", 9), justify = CENTER, relief=SUNKEN, bd=3)
    txt4.place(x=100, y=400)
    lbl18 = Label(Window1, text="Клас пожежної небезпеки", bg="gray",
fg="black", font="Arial", 20, "bold" padx=10, pady=20, width=20,
height=10)
    lbl18.place(x=300, y=250)
    var = IntVar()
    var.set(0)
    r1 = Radiobutton(text="I", variable=var, value=0)
    r1.place(x=320, y=300)
```

```
    r2 = Radiobutton(text="II",variable=var, value=1)
    r2.place(x=320, y=350)
    r3 = Radiobutton(text="III",variable=var, value=2)
    r3.place(x=420, y=400)
    r4 = Radiobutton(text="III",variable=var, value=3)
    r4.place(x=320, y=450)
    r5 = Radiobutton(text="III",variable=var, value=4)
    r5.place(x=320, y=500)
    lbl19 = Label(Window1, text="I: K:0-300", bg="gray", fg="black",
font="Arial", 20, "bold" padx=10, pady=20, width=20, height=10)
    lbl19.place(x=400, y=300)
    lbl20 = Label(Window1, text="II: K:300-1000", bg="gray", fg="black",
font="Arial", 20, "bold" padx=10, pady=20, width=20, height=10)
    lbl20.place(x=400, y=350)
    lbl21 = Label(Window1, text="III: K:1000-4000", bg="gray",
fg="black", font="Arial", 20, "bold" padx=10, pady=20, width=20,
height=10)
    lbl21.place(x=400, y=350)
    lbl22= Label(Window1, text="IV: K:4000-10000", bg="gray",
fg="black", font="Arial", 20, "bold" padx=10, pady=20, width=20,
height=10)
    lbl22.place(x=400, y=400)
    lbl23 = Label(Window1, text="V: K:>10000", bg="gray", fg="black",
font="Arial", 20, "bold" padx=10, pady=20, width=20, height=10)
    lbl23.place(x=400, y=450)
    btn3 = Button(Window1, text = "Fire.png",
command = start_window_1)
    btn3.place(x=500, y=300)
    Window1.mainloop()
```

ДОДАТОК Б. Вхідні дані

```
    from tkinter import *
    def start_window_1():
        new_window_1 = Toplevel(root)
        new_window_1.title("Вхідні дані")
        new_window_1.resizable(0, 0)
    Window2 = Tk()
    Window2.title("Вхідні дані")
    Window2.geometry("800x600")
    Window2.resizable(False, False)
    Window2.iconbitmap("Пожежа2.ico")
    Window1["bg"] = "green"
    lbl1 = Label(Window2, text="Вид пожежі", bg="gray", fg="red",
font="Arial", 20, "bold" padx=10, pady=20, width=20, height=10)
    lbl1.place(x=50, y=25)
    lbl2 = Label(Window2, text="низова пожежа", bg="gray", fg="red",
font="Arial", 20, "bold" padx=10, pady=20, width=20, height=10)
    lbl2.place(x=55, y=40)
    lbl3 = Label(Window2, text="верхова пожежа", bg="gray", fg="red",
font="Arial", 20, "bold" padx=10, pady=20, width=20, height=10)
    lbl3.place(x=200, y=40)
    var = IntVar()
    var.set(0)
    r6 = Radiobutton(Window2, text="I", variable=var, value=0)
    r6.place(x=60, y=70)
    r7 = Radiobutton(Window2, text="II",variable=var, value=1)
    r7.place(x=60, y=100)
    r8 = Radiobutton(Window2, text="біжучий",variable=var, value=2)
    r6.place(x=60, y=130)
    r9 = Radiobutton(Window2, text="стійкий",variable=var, value=3)
    r9.place(x=60, y=160)
    combo1 = Combobox(Window2, selectmode=EXTENDED, value=[східний])
    combo1.place(x=20, y=300)
    lbl4 = Label(Window2, text="Напрямок вітру", bg="gray", fg="black",
font="Arial", 20, "bold" padx=10, pady=20, width=20, height=10)
```

```
    lbl4.place(x=150, y=300)
    lbl5 = Label(Window2, text="Швидкість вітру, м/с", bg="gray",
fg="black", font="Arial", 20, "bold" padx=10, pady=20, width=20,
height=10)
    lbl5.place(x=150, y=350)
    lbl6 = Label(Window2, text="Початковий периметр, км", bg="gray",
fg="black", font="Arial", 20, "bold" padx=10, pady=20, width=20,
height=10)
    lbl6.place(x=150, y=400)
    lbl7 = Label(Window2, text="Початкова площа, га", bg="gray",
fg="black", font="Arial", 20, "bold" padx=10, pady=20, width=20,
height=10)
    lbl7.place(x=150, y=450)
    lbl8 = Label(Window2, text="Час пожежі, год", bg="gray", fg="black",
font="Arial", 20, "bold" padx=10, pady=20, width=20, height=10)
    lbl8.place(x=150, y=500)
    combo2 = Combobox(Window2, selectmode=EXTENDED, value=[сосна])
    combo2.place(x=300, y=300)
    lbl9 = Label(Window2, text="Порода", bg="gray", fg="black",
font="Arial", 20, "bold" padx=10, pady=20, width=20, height=10)
    lbl9.place(x=450, y=300)
    combo3 = Combobox(Window2, selectmode=EXTENDED, value=[25])
    combo3.place(x=300, y=350)
    lbl10 = Label(Window2, text="Середній діаметр дерев", bg="gray",
fg="black", font="Arial", 20, "bold" padx=10, pady=20, width=20,
height=10)
    lbl10.place(x=450, y=350)
    combo4 = Combobox(Window2, selectmode=EXTENDED, value=[100])
    combo4.place(x=300, y=300)
    lbl11 = Label(Window2, text="Середня висота нагару, см", bg="gray",
fg="black", font="Arial", 20, "bold" padx=10, pady=20, width=20,
height=10)
    lbl11.place(x=450, y=400)
    btn1 = Button(Window2, text = "Розрахувати",
command = start_window_2)
```

```
btn1.place(x=300, y=500)
Window2.mainloop()
```

ДОДАТОК В. Прогнозовані параметри лісової пожежі

Fire3.py

```
from tkinter import *
import numpy as np
import matplotlib.pyplot as plt
matplotlib inline
def graf_1():
    x1=np.array([0.18,   0.56,   1.04,   1.47,   1.90,   2.30,   2.71,
    3.04,   3.39,   3.70,   4.10,   4.38,   4.68,   5.01,   5.34,   5.62,
    5.97, 6.41, 6.84, 7.39])
    y1=np.array([5.56,    9.72,    11.11,    25.00,    44.44,    61.11,
    76.39,   90.28,   104.17,   118.06,   133.33,   145.83,   159.72,
    170.83,     186.11,     198.61,     208.33,     211.11,     213.89
    216.67])
    x2=np.array([0.25,    0.66,  1.11,    1.57,  2.10,  2.56,  3.01,
    3.47,   3.90,   4.33,   4.71,   5.09,   5.42,   5.77,   6.05,   6.41,
    6.76, 7.32])
    y2=np.array([0.00    1.39    1.39,    13.89,    30.56,    47.22,
    62.50,   79.17,   91.67,   105.56,   120.83,   131.94,   143.06,
    155.56, 158.33, 158.33, 162.50, 162.50])
    plt.plot(x1, y1)
    plt.plot(x2, y2)
    plt.xlabel('Vв, м/с')
    plt.ylabel('Vф, м/год')
    plt.title('Швидкість поширення фронту пожежі')
    plt.plot(x,y,color='green',linewidth=5,
    linestyle='dotted')
    plt.legend(loc='upper right', shadow=True,
    fontsize='large')
    plt.grid()
    plt.show()
```

```
def graf _2():
    x3=np.array([0.03,  0.43,  0.96,  1.44,  2.00,  2.61,  3.16,
    3.70, 4.18, 4.63, 5.14, 5.80, 6.35, 7.01, 7.59])
    y3=np.array([18.40,  18.93,  18.93,  20.00,  20.53,  21.87,
    23.20, 25.07, 26.40, 27.47, 28.27, 29.07, 29.33, 29.60])
    x4=np.array([0.08,  0.56,  1.14,  1.57,  2.00,  2.43,  2.99,
    3.57, 4.05, 4.48, 4.94, 5.34, 5.80, 6.20, 6.71])
    y4=np.array([8.27,  9.33,  10.40,  11.73,  13.33,  14.93,
    17.60,  19.73,  21.60,  22.93,  23.73,  24.27,  24.80,  24.80,
    25.07])
    plt.plot(x3, y3)
    plt.plot(x4, y4)
    plt.xlabel('Vв, м/с')
    plt.ylabel('Vфл, м/год')
    plt.title('Швидкість поширення флангів пожежі')
    plt.plot(x,y,color='green',linewidth=5,
    linestyle='dotted')
    plt.legend(loc='upper right', shadow=True,
    fontsize='large')
    plt.grid()
    plt.show()
def graf_3():
    x5=np.array([0.03,  0.45,  0.95,  1.45,  1.86,  2.36,  2.78,
    3.29,  3.74,  4.24,  4.69,  5.19,  5.69,  6.24,  6.85,  7.35,
    7.87])
    y5=np.array([5.13,  5.13,  5.38,  6.67,  8.46,  10.00,  12.,05,
    14.36,  16.41,  18.46,  20.00,  21.28,  21.28,  21.54,  21.79,
    22.05, 22.05])
    x6=np.array([0.10,  0.53,  0.95,  1.55,  2.16,  2.61,  3.03,
    3.51, 3.96, 4.59, 5.19, 5.77, 6.37, 6.90, 7.37, 7.85])
    y6=np.array([5.13,  5.13,  5.38,  6.15,  7.44,  7.95,  9.23,
    10.26,  10.26,  10.77,  11.03,  11.03,  11.54,  12.05,  12.31,
    12.31])
    plt.plot(x5, y5)
    plt.plot(x6, y6)
```

```
        plt.xlabel('Vв, м/с')
        plt.ylabel('Vтл, м/год')
        plt.title('Швидкість поширення тилу пожежі')
        plt.plot(x,y,color='green',linewidth=5,
        linestyle='dotted')
        plt.legend(loc='upper right', shadow=True,
        fontsize='large')
        plt.grid()
        plt.show()
    Window3 = Tk()
    Window3.title("Вхідні дані")
    Window3.geometry("800x600")
    Window3.resizable(False, False)
    Window3["bg"] = "green"
    lbl1 = Label(Window3, text="Час після виявлення пожежі", bg="gray",
fg="red", font="Arial", 20, "bold" padx=10, pady=20, width=20,
height=10)
    lbl1.place(x=200, y=20)
    lbl2 = Label(Window3, text="Швидкість поширення фронту пожежі,
м/год", bg="gray", fg="red", font="Arial", 20, "bold" padx=10, pady=20,
width=20, height=10)
    lbl2.place(x=200, y=40)
    lbl3 = Label(Window3, text="Швидкість поширення флангів пожежі",
bg="gray", fg="red", font="Arial", 20, "bold" padx=10, pady=20,
width=20, height=10)
    lbl3.place(x=200, y=60)
    lbl4 = Label(Window3, text="Швидкість поширення тилу пожеж, м/год",
bg="gray", fg="red", font="Arial", 20, "bold" padx=10, pady=20,
width=20, height=10)
    lbl4.place(x=200, y=80)
    lbl5 = Label(Window3, text="Периметр пожежі, км", bg="gray",
fg="red", font="Arial", 20, "bold" padx=10, pady=20, width=20,
height=10)
    lbl5.place(x=200, y=100)
```

```
    lbl6 = Label(Window3, text="Площа пожежі, га", bg="gray", fg="red",
font="Arial", 20, "bold" padx=10, pady=20, width=20, height=10)
    lbl6.place(x=200, y=120)
    txt1 = Entry(Window3, width = 30, bg="#00E0EE" fg="blue",
font=("Verdana", 9), justify = CENTER, relief=SUNKEN, bd=3)
    txt1.place(x=10, y=20)
    txt2 = Entry(Window3, width = 30, bg="#00E0EE" fg="blue",
font=("Verdana", 9), justify = CENTER, relief=SUNKEN, bd=3)
    txt2.place(x=10, y=40)
    txt3 = Entry(Window3, width = 30, bg="#00E0EE" fg="blue",
font=("Verdana", 9), justify = CENTER, relief=SUNKEN, bd=3)
    txt3.place(x= x=10, y=60)
    txt4= Entry(Window3, width = 30, bg="#00E0EE" fg="blue",
font=("Verdana", 9), justify = CENTER, relief=SUNKEN, bd=3)
    txt4.place(x=10, y=80)
    txt5 = Entry(Window3, width = 30, bg="#00E0EE" fg="blue",
font=("Verdana", 9), justify = CENTER, relief=SUNKEN, bd=3)
    txt5.place(x=10, y=100)
    txt6 = Entry(Window3, width = 30, bg="#00E0EE" fg="blue",
font=("Verdana", 9), justify = CENTER, relief=SUNKEN, bd=3)
    txt6.place(x=10, y=120)
    btn1 = Button(Window3, text = "Графік",
command = graf_1)
    btn1.place(x=400, y=100)
    btn2 = Button(Window3, text = "Графік",
command = graf_2)
    btn2.place(x=400, y=150)
    btn3 = Button(Window3, text = "Графік",command = graf_3)
    btn3.place(x=400, y=200)
    Window3.mainloop()
```

ДОДАТОК Г

datatable.py

```
import tkinter as tk
from tkinter import ttk
import pandas as pd
class DataTable(ttk.Treeview):
    def __init__(self, parent, axis=None):
        super().__init__(parent)
        self.stored_dataframe = pd.DataFrame()
        if axis == "both":
            scroll_Y = tk.Scrollbar(self, orient="vertical",
command=self.yview)
            scroll_X = tk.Scrollbar(self, orient="horizontal",
command=self.xview)
            self.configure(yscrollcommand=scroll_Y.set,
xscrollcommand=scroll_X.set)
            scroll_Y.pack(side="right", fill="y")
            scroll_X.pack(side="bottom", fill="x")
        elif axis == "x":
            scroll_X = tk.Scrollbar(self, orient="horizontal",
command=self.xview)
            self.configure(xscrollcommand=scroll_X.set)
            scroll_X.pack(side="bottom", fill="x")
        elif axis == "y":
            scroll_Y = tk.Scrollbar(self, orient="vertical",
command=self.yview)
            self.configure(yscrollcommand=scroll_Y.set)
            scroll_Y.pack(side="right", fill="y")
        else:
            pass
    def set_datatable_from_dataframe(self, dataframe):
        self.stored_dataframe = dataframe
        self._draw_table(dataframe)
```

```
def set_datatable_from_object(self, objects):
    if not isinstance(objects, list):
        objects = [objects]
    df = pd.DataFrame([vars(header) for header in objects])
    self.stored_dataframe = df
    self._draw_table(df)
def _draw_table(self, dataframe):
    self.delete(*self.get_children())
    columns = list(dataframe.columns)
    self.__setitem__("column", columns)
    self.__setitem__("show", "headings")
    for col in columns:
        self.heading(col, text=col)

    df_rows = dataframe.to_numpy().tolist()
    for row in df_rows:
        self.insert("", "end", values=row)
    return None
def find_value(self, pairs):
    # pairs is a dictionary
    new_df = self.stored_dataframe
    for col, value in pairs.items():
        query_string = f"{col}.str.contains('{value}')"
        new_df = new_df.query(query_string, engine="python")
    self._draw_table(new_df)
def reset_table(self):
    self._draw_table(self.stored_dataframe)
```

Database.py

```
import sqlite3
from pathlib import Path
from collections import namedtuple
class DBContext(object):
    def __init__(self, db_file=""):
        self.db_file = db_file
        self.connection_status = Path(db_file).exists()
    def __enter__(self):
        self.conn = sqlite3.connect(self.db_file)
        self.conn.row_factory = sqlite3.Row
        self.cur = self.conn.cursor()
        return self.cur
    def __exit__(self, ext_type, exc_value, traceback):
        self.cur.close()

        if isinstance(exc_value, Exception):
self.conn.rollback()
            self.conn.commit()
        self.conn.close()
class PRMS_Database(object):
    def __init__(self):
        _DATABASE_NAME = "fire.db"
        _PARENT_PATH = Path(__file__).parent.parent
        self.DB_PATH = str(Path.joinpath(_PARENT_PATH, _DATABASE_NAME))
        self.context = DBContext(self.DB_PATH)
        if not self.context.connection_status:
            self._setup_database_tables()
            self.context.connection_status = True
        self.is_connected = self.context.connection_status
    def get_db_instruments(self):
        with self.context as db:
            db.execute(
            )
```

```
            instrument_table = db.fetchall()
        instrument_pairs = {row["name"]: row["displayName"] for row in
instrument_table}
        return instrument_pairs
    def store_instruments(self, oanda_instruments):
        db_instruments = self.get_db_instruments()
        db_names = set(db_instruments.keys())
        count = 0
        insert_query = "INSERT INTO Instruments VALUES (:name,
:displayName);"
        with self.context as db:
            for name, display_name in oanda_instruments.items():
                if name not in db_names:
                    db.execute(
                        insert_query, {"name": name, "displayName":
display_name}
                    )
                    count += 1
        return count
    def record_trade(self, fill):
        if not isinstance(fill, str):
            if "orderFillTransaction" in fill:
                id = fill["orderFillTransaction"]["orderID"]
                instrument = fill["orderFillTransaction"]["instrument"]
                units = fill["orderFillTransaction"]["units"]
                price = fill["orderFillTransaction"]["price"]
                profit = fill["orderFillTransaction"]["pl"]
                cancelled = 0
                with self.context as db:
                    values = {
                        "id": id,
                        "name": instrument,
                        "quantity": units,
                        "price": price,
                        "profit": profit,
```

```
                    "cancelled": cancelled,
                }
                insert_query = "INSERT INTO All_Transactions VALUES
(:id, :name, :quantity, :price, :profit, :cancelled);"
                db.execute(insert_query, values)
    def _setup_database_tables(self):
        with self.context as db:
            db.execute(
            )
            db.execute(
            )
    def get_transactions(self):
        query = "SELECT * FROM All_Transactions;"
        with self.context as db:
            db.execute(query)
            result = db.fetchall()
        transactions = []
        for row in result:
            t = DBTransaction(
                id=row["id"],
                name=row["name"],
                quantity=row["quantity"],
                price=row["price"],
                pnl=row["pnl"],
                cancelled=row["cancelled"],
            )
            transactions.append(t)
        return transactions
    def cancelled_toggle(self, id, toggle):
        with self.context as db:
            db.execute(
                (toggle, id),
            )
            result = db.rowcount
        if result == 0:
```

```
            return "Order ID does not exist in the database."
        else:
            return f"Changes have been made to Order ID {id}."
    def get_credentials(self):
        with self.context as db:
            db.execute(
            )
            credentials = db.fetchall()
        return credentials
    def _generateId(self):
        with self.context as db:
            db.execute()
            last_id = db.fetchone()[0]
        if last_id is None:
            new_id = "C{:04n}".format(1)
        else:
            new_id = "C{:04n}".format(int(last_id) + 1)
        return new_id
    def save_transaction(self, name, units, price):
        id = self._generateId()
        placeholders = {
            "id": id,
            "name": name,
            "quantity": units,
            "price": price,
            "pnl": 0,
            "cancelled": 0,
        }
        with self.context as db:
            db.execute(placeholders)
            )
        return f"Order ID {id} stored to the database"

    def get_db_positions(self):
        query = """SELECT
```

```
                    Instrument as 'name',
                    SUM([Units]) as 'prms_units',
                    AVG([Price]) as 'prms_avg_price'
                FROM
                (
                    SELECT
                        a.name as 'Instrument',
                        CASE WHEN a.quantity < 0 then SUM(a.quantity)
else 0 end as 'Units',
                        CASE WHEN a.quantity < 0 then
SUM(a.quantity*a.price)/SUM(a.quantity) else 0 end as 'Price'
                        from All_Transactions a
                    WHERE a.cancelled = 0 AND a.quantity < 0
                    GROUP BY a.name
                    HAVING 'Value' > 0
                    UNION ALL
                    SELECT
                        b.name as 'Instrument',
                        CASE WHEN b.quantity > 0 then SUM(b.quantity)
else 0 end as 'Units',
                        CASE WHEN b.quantity > 0 then
SUM(b.quantity*b.price)/SUM(b.quantity) else 0 end as 'Price'
                        FROM All_Transactions b
                    WHERE b.cancelled = 0 AND b.quantity > 0
                    GROUP BY b.name
                    HAVING 'Price' > 0)
        with self.context as db:
            db.execute(query)
            db_positions = db.fetchall()
        prms_positions = []
        for position in db_positions:
            p = DBPosition(
                position["name"], position["prms_units"],
position["prms_avg_price"]
            )
```

```
            prms_positions.append(p)
        return prms_positions
    def get_largest_positions(self):
        query = """SELECT name, SUM(quantity*price) as 'MarketVal'
                    FROM All_Transactions
                    WHERE cancelled = 0
                    GROUP BY name
                    ORDER BY ABS(SUM(quantity*price))DESC
                    LIMIT 5;"""
        with self.context as db:
            db.execute(query)
            db_positions = db.fetchall()
        positions = [
            DBPieChartData(row["name"], row["Fire"]) for row in
db_positions
        ]
        return positions

    def validate_registration(self, username):
        with self.context as db:
            query = "SELECT Username FROM LoginInfo WHERE Username=?"
            db.execute(query, (username,))
            user = db.fetchone()
        is_valid = user is None
        return is_valid

    def store_user_credentials(
        self, username, hash, oanda_account, oanda_api, news_api,
alpha_vantage_api
    ):
        with self.context as db:
            query = "INSERT INTO LoginInfo VALUES (:username, :key,
:salt, :oanda_api, :oanda_account, :news_api, :alphaVantage_api)"
            values = {
            db.execute(query, values)
```

```
    def get_user(self, username):
        with self.context as db:
            query = "SELECT * FROM LoginInfo WHERE username=?"
            db.execute(query, (username,))
            db_user = db.fetchone()
        if db_user is None:
            return None
        else:
            user = DBUser(
                db_user["username"],
                db_user["key"],
                db_user["salt"],
                db_user["oanda_account"],
                db_user["oanda_api"],
                db_user["news_api"],
                db_user["alphaVantage_api"],
            )
            return user
class DBObject(object):
    pass
class DBTransaction(DBObject):
    def __init__(self, id, name, quantity, price, pnl, cancelled):
        self.id = id
        self.name = name
        self.quantity = quantity
        self.price = price
        self.pnl = pnl
        self.cancelled = cancelled
class DBPosition(DBObject):
    def __init__(self, name, prms_units, prms_avg_price):
        self.name = name
        self.prms_units = round(prms_units, 2)
        self.prms_avg_price = round(prms_avg_price, 2)
class DBPieChartData(DBObject):
    def __init__(self, name, Fire):
```

```
        self.name = name
        self.MarketVal = round(Fire, 2)
class DBUser(DBObject):
    def __init__(
        self, username, key, salt, oanda_account, oanda_api, news_api,
alpha_vantage_api
    ):
        self.username = username
        self.key = key
        self.salt = salt
        self.oanda_account = oanda_account
        self.oanda_api = oanda_api
        self.news_api = news_api
        self.alpha_vantage_api = alpha_vantage_api
```