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 математичного та програмного забезпечення для обліку лісових насаджень»**

Виконала: студентка VI курсу групи КНз-61(м)

спеціальності

122 "Комп'ютерні науки"

(шифр і назва напряму підготовки, спеціальності)

Василів Р. Б.

(прізвище та ініціали)

Керівник Пірко І. Б.

(прізвище та ініціали)

Рецензент _____

(прізвище та ініціали)

Львів – 2022 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.

"____" _____2022 року

З А В Д А Н Н Я

НА ДИПЛОМНУ РОБОТУ СТУДЕНТЦІ

Василів Романні Богданівні

(прізвище, ім'я, по батькові)

1. Тема роботи **Розроблення математичного та програмного забезпечення для обліку лісових насаджень**

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 "4" березня 2021 р. № С-89

2. Термін подання студентом роботи 14 лютого 2022 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 26 лютого 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Огляд літературних даних та інших джерел згідно досліджуваної теми	26.02-30.03.2021	
2	Аналіз досліджуваної теми та вибір відповідних варіантів її розробки	01.04-30.04.2021	
3	Постановка задачі та її формалізація	01.05-15.05.2021	
4	Вибір та обґрунтування методів і засобів проведення дослідження	16.05-30.05.2021	
5	Розроблення концептуальної схеми реалізації завдання	01.06-30.06.2021	
6	Програмна реалізація завдання	1.07-30.09.2021	
7	Тестування програмного продукту та отриманих результатів	01.10-30.10.2021	
8	Розробка пояснювальної записки магістерської роботи	1.11-30.12.2021	
9	Корегування пояснювальної записки згідно вимог, розроблення презентації	01.01-30.01.2022	

Студентка _____ **Василів Р. Б.**
(підпис) (прізвище та ініціали)

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

РЕФЕРАТ

Дипломна робота містить 91 сторінку пояснювальної записки, 28 рисунків, 2 додатки, 16 джерел, 3 таблиці.

В роботі розглянуто особливості обліку лісових насаджень, формування їх бази даних та розробка програмного забезпечення для обліку та моделювання росту лісових масивів. Основною метою моделювання лісогосподарських процесів є створення такої моделі (програми), яка найповніше б описувала процеси зміни лісових систем загалом чи окремих її структурних частин. Кожен конкретний випадок ставить відповідне завдання: створення моделей росту та продуктивності деревостанів, моделей їх будови за обліковими показниками, моделей зміни окремих облікових та модельних показників з часом, моделей взаємозв'язків модельних показників між собою.

З врахуванням особливостей лісових територій дане програмне забезпечення може бути впроваджене в структурні підрозділи лісових господарств.

Ключові слова: облік лісових насаджень, Python, Tkinter, Matplotlib.

ABSTRACT

Diploma paper contains 91 pages of explanatory note, 28 pictures, 2 application, 16 used literary sources, 3 tables.

The peculiarities of forest plantation accounting, database formation and software development for forest plantation accounting are considered in the work. The main purpose of modeling forestry processes is to create a model (program) that would most fully describe the processes of change of forest systems in general or its individual structural parts. Each case sets the appropriate task: the creation of models of growth and productivity of stands, models of their structure on the basis of accounting indicators, models of change of individual accounting indicators over time, models of relationships between accounting indicators.

Taking into account the peculiarities of forest areas, this software can be implemented in structural units of forestry.

Keywords: forest accounting, Python, Tkinter, Matplotlib.

ТЕХНІЧНЕ ЗАВДАННЯ

Розробити інформаційну систему для обліку лісових насаджень.

1. Провести огляд літератури по даній тематиці, проаналізувати існуючі системи обліку лісових насаджень.

2. Розробити математичну модель для моделювання ходу росту лісових насаджень.

3. Реалізувати програмне забезпечення мовою Python обліку лісових насаджень сосни, смереки, дуба, граба, клена та моделювання їх росту.

4. В рамках даних моделей провести дослідження параметрів системи обліку та моделювання цих лісових насаджень на досліджуваній території.

.

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

- GUI – Graphical User Interface;
- БД – база даних;
- ПЗ – програмне забезпечення;
- ТЛ – тип лісу;
- МХР – моделі ходу росту;
- А – вік дерев (років);
- $d_{1,3}$ – діаметр стовбура дерева на висоті 1,3 м (см);
- $d_{ср}$ – середній діаметр стовбура дерева (см);
- h – висота стовбура дерева (м);
- $H_{ср}$ – середня висота лісового насадження (м);
- S – площа перерізу стовбура ($м^2$);
- V – об’єм стовбура дерева ($м^3$);

ЗМІСТ

ВСТУП

Актуальність роботи

Сучасний облік лісових насаджень потребує використання комп'ютерної техніки із застосуванням нових інформаційних технологій. Впровадження комп'ютеризації призвело до суттєвих змін в організаційній структурі, технології та методичних підходах з одержання й обробки цієї інформації.

Лісова промисловість потребує актуальних відомостей про стан лісів та лісоматеріалів, які в них заготовляються. Ця інформація є основою організації підприємств лісової промисловості, розробки проектів освоєння лісів із заготівлі деревини. Успішна практична діяльність лісозаготівельного виробництва неможлива без засвоєння практичних навичок роботи з обліковими методами, уміння виконувати кваліфіковані облікові розрахунки стосовно до різних об'єктів лісових насаджень.

Предметом дослідження дипломної роботи є розробка інформаційної системи для обліку лісових насаджень.

Об'єктом дослідження є лісові насадження.

Мета роботи – розробка та реалізація програмного забезпечення для обліку лісових насаджень.

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

- провести огляд літератури по даній тематиці, проаналізувати існуючі системи обліку лісових насаджень;
- розробити математичну модель для моделювання ходу росту лісових насаджень;
- реалізувати програмне забезпечення мовою Python для обліку та моделювання росту лісових насаджень сосни, смереки, дуба, граба, клена;
- в рамках даних моделей провести дослідження параметрів системи обліку та моделювання ходу росту цих лісових насаджень на досліджуваній території.

Наукова новизна одержаних результатів

Запропоновано новий підхід для обліку лісових насаджень, який оснований на детальній характеристиці досліджуваної території лісових масивів. Даний підхід передбачає використання експертних оцінок, апроксимаційних методів.

Практичне значення одержаних результатів

Розроблено програмне забезпечення, яке призначене для обліку та моделювання ходу росту лісових насаджень сосни, смереки, дуба, граба, берези. З врахуванням особливостей лісових територій дана програма може бути впроваджена в структурні підрозділи лісових господарств.

РОЗДІЛ 1. СТАН ПРОБЛЕМНОЇ ОБЛАСТІ

1.1. Моделювання динаміки і розвитку деревостанів

Розглянемо поняття моделей росту деревостанів та процедури моделювання їх розвитку. Статичні моделі (таблиці росту) сприяли визначенню найбільш продуктивних параметрів деревостанів, але вони не дають відповіді на питання, з яких початкових станів вони сформовані. Для побудови моделі розвитку деревостану рекомендується використовувати підхід повторних спостережень, враховуючи вихідну історію різних показників: перекриття стовбурів, коефіцієнт перекриття крони, їх загальний об’єм і відносну довжину крони. Загальні властивості насаджень, такі як константи об’єму крони, можуть бути використані як теоретична основа для розробки моделей.

Моделювання служить меті управління лісовою екосистемою, завжди починаючи з рівня деревостану і далі пов’язуючи все більше і більше агрегації. Модель завжди простіша за реальний об’єкт і повністю відображає його властивості, що цікавлять дане дослідження. Тому модель росту, динаміки та розвитку лісових деревостанів відображає основні властивості, які визначені розробником.

З першого року деревостану розміри дерев змінюються – з’являється різноманітність їх життєвого потенціалу. Причини цього процесу можуть бути різними: час появи, відмінності у віці, нерівномірне розташування, умови тощо. Відсталі дерева зникають – деревостани деградують. Кожне дерево має здатність рости і розвиватися природним шляхом, максимізуючи індивідуальні особливості та умови життя, і, успадкувавши від нього, розвиває в першу чергу кореневу систему і крону.

Деревостан, який складається з багатьох дерев з різним життєвим потенціалом, які набули загального атрибуту повноцінного використання умов для розвитку та прагнуть швидко опанувати отриманий у спадок житловий простір. У цьому полягає прагнення деревостану швидко досягти індивідуальних обмежень для всіх облікових показників. Логіка цих речей начебто цілком відповідає ідеям і законам екології, але її закони не повністю використовуються для моделювання розвитку лісових насаджень.

Ріст лісових масивів є досить різноманітний і вперше про це повідомлялося ще на початку 20 століття. Вивченням росту деревостанів займаються організації, які ведуть облік лісових масивів. Вони першими визначили типи росту, вивчили їх різноманітність і закономірності.

Табл. 1.1. Зміни класів в лісових насадженнях у різних вікових періодах, % від кількості деревостанів

поведінка класів	віковий період, років						В середньому
	21-30	31-40	41-50	51-60	61-70	71-80	
стабільна	16	16	7	10	5	3	10
нестабільна	84	84	93	90	95	97	90
в тому числі підвищення/ пониження	32/52	68/16	54/39	25/68	9/86	7/90	32/58

Під типом росту лісових масивів розуміють зміни його росту по відношенню до середніх показників у деяких деревостанах, які ростуть в однакових умовах. Причини таких типів росту були незрозумілі, тоді як типи ходу росту, представлені як системи ліній росту, відбивали вплив загальних факторів. Саме тому і у зв’язку з необхідністю вивчати ліси було вирішено розробляти таблиці ходу росту (ТХР), уніфікувати та складати їх з підбором вікового ряду, які відображали б динаміку росту насаджень.

1.2. Моделі динаміки статичних станів деревостанів

Перші дослідні таблиці запасу та приросту лісових насаджень було опублікувано у 1846 р. Варгасом де Бедемаром. Подібні таблиці розробив також М. М. Орлов у 1897 р. та назвав їх "Таблиці росту нормальних насаджень". Утворилося нове поняття, яке передбачає ріст, тобто динаміку чогось у часі.

Тюрін А. В. замість колишнього терміна нормальні використовував термін зімкнені; ТХР у нього включали 28 одиниць. У довіднику Козловського В. Б. та Павлова В. М. наведено вже 113 ТХР. Тут вони названі по-різному: більшість – зімкнутими, багато – взагалі без найменувань, деякі – нормальними, нормально

зімкнутими, модальними. Загреєв В. В. вказав, що ним зібрано дуже велику кількість ТХР: нормальних – 370 одиниць, модальних – 104.

З того часу багато дослідників почали застосовувати ТХР. Їх суть для одного типу лісу наступна: знаходження повних деревостанів у трьох опорних групах (хвойні – 50, 100, 150 років; листяні – 20, 50 та 80 років); визначення верхньої висоти деревостану по великих деревах, їх рубка та аналіз ходу росту моделей зі старших за віком деревостанів, аналітичне вирівнювання ліній росту великих моделей по висоті (вирівняна лінія вкаже лінію зміни висоти в молодшому віці). Відповідно до цієї лінії знаходять повні деревостани в молодших деревах з відповідною верхньою висотою.

Моделювання за даним методом полягає у визначенні параметрів асиметрії рядів, формул зв'язку показників з віком і т. д. В останні роки ТХР повних деревостанів намагаються покращити, аналізувати з метою знаходження густоти, що призводить до максимальної продуктивності, а також включити їх у єдину облікову систему.

Найбільш складні моделі – це такі, які базуються на виясненні статичних станів деревостанів. Вони створені з метою знаходження оптимальної їх густоти. Було вивчено площі живлення, розміри дерев, масу хвої та крони 1 тис. модельних дерев на пробних площах віком від 16 до 120 років. Пошук оптимальної густоти базувався на пошуку такої площі живлення дерев, яка забезпечувала би їм найбільший приріст.

Такі підходи отримали подальший розвиток при оцінці, моделюванні та оптимізації структури деревостанів на основі вивчення соціальних груп – штучних утворень з будь-яким випадково вибраним деревом, що поміщається в центр вибірки, для якого розраховували два показники: середню відстань до сусідів та відношення суми діаметрів крон 1-16 сусідів до діаметра крони дерева, яке вибиралося за центральне. Цей другий показник має назву напруженість конкуренції.

Загалом моделювали історію росту окремих дерев та переносили її на хід росту деревостану, а далі і на сукупність із деревостанів різного віку. Дослідники цілком покладалися на те, що модельні дерева відбивають як історію розвитку висоти свого деревостану, але вказують її й для інших, молодших деревостанів. По середній лінії росту таких моделей знаходили координати висот та віку для пошуку молодих деревостанів та встановлювали природний ряд росту деревостанів за типами лісу.

Концепцію таких ТХР можна представити як пошук статичних станів, які поєднують у собі оптимальним чином два стани деревостану: високу продуктивність (повноту) і найкращі облікові показники, що виражаються в наближенні рядів їх розподілу до нормального закону. Ці стани знаходили, закладаючи кілька пробних площ одномоментно, намагаючись рівномірно уявити їх за класами бонітету, потім апроксимували лінії тренду і отримувати з них дані для 10, 20, 30 років і далі. У таблиці відображали динаміку цих статичних станів за класами бонітету.

Але ця динаміка – не розвиток, це лише фіксація станів – граничних для нормальних і середніх для модальних ТХР. З'ясувати, якими ці стани будуть через 10 років або з яких колишніх станів вони з'явилися, укладачі таблиць не намагалися, та й такого завдання не ставили перед собою. У результаті процес розвитку деревостанів виявлявся невивченим.

1.3. Моделі розвитку лісових насаджень

Неадекватність таблиць ходу росту до реальної динаміки росту деревостанів відзначалася давно. По-перше, хід росту з них ніколи не перевірявся на реальних деревостанах повторними спостереженнями. По-друге, термін ходу росту, справедливий для дерева, було присвоєно таблицям продуктивності деревостанів у статиці.

Наразі розглядаються загальні, конкурентні моделі розвитку лісових екосистем. Ще на початку XX ст., коли почали використовувати дерева-моделі, сенс операцій з ними (вивчення ходу росту) був перенесений на статичну конструкцію, що вибудовується з повних деревостанів – таблицю показників, знятих з ліній трендів. Таблицю стали називати аналогічно – хід росту деревостанів.

Успішний розвиток деревостану визначає його приріст, який залежить від потужності фотосинтезуючого апарату. Однак цей апарат, а також об'єми крон і маса хвої раніше мало кого цікавили. Більше того, при вивченні приросту лісу так і не ставилося питання, причому приріст не пов'язували і з ходом росту деревостанів. Розроблені тоді моделі мали лаг прогнозу 5 років за точності ± 5–8 % та ймовірності 68 % (при ймовірності 95 % помилки будуть ± 10–16 %), і використовуються вони

досі. Але якщо для актуалізації даних обліку цього достатньо, то для вирощування лісу потрібен прогноз на 30-50 років. При цьому помилки зростуть до 50 %, тобто прогнози щодо таких моделей втратять сенс.

У рості лісових масивів як процесу виділяють вихідну лінію (прогрес) та спадну (регрес). Прогрес пов'язують із збільшенням приросту деревини, а регрес – зі зниженням. Причини лежать в об'ємах фотосинтезуючого апарату та масі листя, а також у сумарному об'ємі крон. Розрахунки по них складні, але результати виходять цікаві. Якщо виявлено константу (сумарний об'єм крон, маса хвої), то моделювання знаходить свою точку відліку, свій опорний експериментальний факт.

Моделі розвитку деревостанів найточніше отримують на основі стаціонарних спостережень. За такими даними проводять аналіз ходу росту культур. Виявилося, що від початкової густоти залежали всі їхні показники, і в рідких культурах вони були вищими. Потрібно визначити, як шукати природні ряди розвитку деревостанів за густотою.

Головними факторами, що враховувалися, були наступні:

- лісорослинна зона та підзона;
- тип умов росту;
- тип деревостану;
- початкова густота у віці 10 років;

Остаточні висновки щодо ідентичності умов робилися з урахуванням видової висоти деревостану. Моделювання (реконструкція) розвитку деревостанів одного природного ряду за початковою густотою складалося з наступних операцій.

1. Будувався графік $HF_{ср} = f(A)$ і проводилася вирівнювальна середня лінія за точками розташування HF. Вздовж неї проводяться дві лінії, що обмежують область нормативного відхилення: для молодняків від +15 до -15 %, для середньовікових – від +10 до -10, а для старших – від +7 до -7 %. На цей же графік наносяться значення HF для деревостанів та інших площ, попереднє визначення яких було сумнівним, і якщо HF цих деревостоїв, що перевіряються, опинялося в межах допустимих відхилень, то вони теж визнавалися, що знаходяться в одному ряду.

2. Поділ деревостанів на класи по початковій густоті є найскладнішим і здійснюється спочатку з використанням середнього діаметра деревостану $D_{ср}$. Будується графік залежності середнього діаметра від віку $D_{ср} = f(A)$ з усіх пробних площ. На цьому графіку вся площина – це область значень діаметрів, яка обмежується крайніми лініями. Вона була розділена для класів густоти на смуги однакової ширини у кожному класі віку. Кожна смуга відображає віковий ряд значень діаметру, що змінюються, що відносяться до одного з класів густоти.

3. Додатковими, а часто й обов'язковими критеріями віднесення деревостану до якого-небудь класу початкової густоти є ще чотири показники, що формуються в залежності від історії густоти деревостою (N) і важливих для прогнозу його розвитку:

- коефіцієнт К – середній збіг стовбурів: це відношення середнього діаметра на висоті 1,3 м $D_{ср}$ до середньої висоти деревостану Н;
- коефіцієнт форми стовбура q_2, який також залежить від густоти деревостану;
- середня довжина крон, яка віднесена до середньої висоти деревостану: $L_{кр}$ / Н;
- середній діаметр крон, який віднесений до середньої висоти деревостану: $D_{кр}$/Н.

У процесі поділу деревостанів за початковою густотою було створено їх класифікацію (табл. 2).

Табл. 1.2. Класифікація лісових масивів за початковою густотою

група густоти	номер класу густоти – початкова густота у віці 10 років, тис. шт/га		
дуже густі	1-172	2-62	3-32
густі	4-20	5-14	6-10
середньої густоти	7-7,9	8-6,3	9-5,1
рідкі	10-4,2	11-2,9	12-2,2
дуже рідкі	13-1,65	14-1,29	15-1,03

4. Далі для всіх площ будується область значень і підбирається тренд залежно від віку всіх необхідних для моделювання показників: $H = f(A)$, $HF = f(A)$, $D_{кр}/H = f(A)$,

$L_{кр}/H = f(A)$, $N = f(A)$. З ліній трендів беруться відліки значень показників за ступенями віку (через 5 або 10 років) і заносяться до порожніх комірок таблиці.

З перших років життя і формування в деревостанах починається диференціація дерев (якісний поділ їх по росту та розвитку), яка є наслідком конкуренції, і чим густота більша, тим інтенсивніша конкуренція, що призводить до відпаду частини рослин. Для оцінки конкуренції використовувалися два різні показники:

- коефіцієнт перекриття кронами горизонтальної поверхні (зімкненість крон $C_{кр}$);
- коефіцієнт зімкненості пологу ($C_{п}$).

Зрозуміти цю модель можна швидше, якщо уявити її утворення як процес пошуку для неї даних, починаючи з наймолодших дерев. Дана спрощена модель описує граничні статичні стани дерев при деякій їх висоті і дає уявлення про те, що у разі вивчення деревостанів у різних станах і з високою густотою з раннього віку виявляються дуже зімкнуті молодняки, де крони дерев проростають не тільки в крону сусіда, але і в крону другого-третього дерева, і площа їх проекцій може бути більшою пробної площі в 1,2-2,6 разів більше площі в 1,2-2,6 рази. Через 5–7 років виявили сильне зрідження та зниження зімкнутості, причому дерева не збільшили свій приріст, як це можна було б очікувати після відпаду їхньої частини та збільшення площі живлення. Виявлена депресія призводить до наступної гіпотези: різна початкова густота призводить до появи різних типів розвитку деревостанів, і знаходження граничних значень зімкнутості крон є ключем до пошуку цих різних типів розвитку.

Швидкість проходження цих етапів була тим вищою, чим більшою була початкова густота. На основі цих досліджень було сформульовано закон: зімкненість крон дерев будь-яких простих деревостанів зі збільшенням середньої висоти та віку підвищується по кривій лінії від мінімуму (0,1–0,2) до максимуму, що дорівнює 1,0 і більше, а потім зменшується до мінімуму, що дорівнює 0,7–0,5 та менше, по кривій лінії з інтенсивністю, що залежить від породного складу, початкової та поточної густоти, рівномірності, режиму догляду та умов росту.

За динамікою густоти деревостанів виявлено такі закономірності:

- відпад дерев відбувається із перших років життя;
- інтенсивність відпаду залежить від типів місцепроживання, але значно більше – від початкової густоти, і чим вона більша, то більше відбувалося відпаду дерев;
- інтенсивність відпаду залежить від коефіцієнта взаємного перекриття крон (зімкнутості крон), потім від зімкнутості пологу та відносної повноти, і з досягненням граничних величин цих показників відпад різко збільшується; далі зі зменшенням зімкнутості та повноти інтенсивність відпаду зменшується;
- у відпад потрапляють в основному дерева, що мають діаметри стовбура менше 0,2-0,5 від середнього і з довжиною крони менше 10-12 % від висоти стовбура;
- спочатку густі деревостани залишаються такими лише до віку початку розпаду, і перегущеність у них сягає 10-20 разів;
- природний відпад (авторегуляція густоти) довго не приводить деревостани до оптимальної густоти;
- авторегуляція густоти, хоч і діє безперервно, запізнюється, і перегущені деревостани завжди потребують розрідження. Навіть рідкі змолоду деревостани виявляються перегущеними через розростання крони.

Цей закон і наслідки з нього застосовуються для визначення оптимальної поточної густоти деревостанів і для розробки моделей лісових культур різної густоти. Таким чином, моделі мають ґрунтуватися на законах розвитку деревостанів з урахуванням тенденцій їхньої сучасної еволюції. Таких законів відомо поки три: закон морфогенезу простих деревостанів, ранговий закон зростання дерев та закон розвитку одноярусних деревостанів.

Потрібні моделі розвитку деревостанів, що описують у табличній чи іншій формі сам процес розвитку – від часу формування деревостану до початку його розпаду, з виділенням фаз прогресу та регресу. Для моделей важливо постулювати інтегральні їх властивості, наприклад константи для маси хвої та об’ємів крон.

1.4. Основні напрямки моделювання лісових насаджень

В інформаційній системі обліку лісових насаджень велике значення мають моделі, що описують процеси, які в них відбуваються. Моделювання при цьому ведеться у таких напрямках:

- моделі, які описують зміну окремих облікових показників;
- моделі ходу росту окремого дерева;
- моделі будови деревостанів по окремих показниках;
- моделі, які характеризують хід росту та продуктивність деревостану в цілому.

Розробка моделей є головною метою вчення про продуктивність деревостанів. З їх допомогою здійснюється конкретизація принципів цільового лісу. Від закономірностей продуктивності деревостанів залежить вибір форм лісового господарства, порід лісових дерев. За підсумками закономірностей будуються багато нормативів лісового господарства, без знання яких неможлива реальна оцінка лісових ресурсів. Іншими словами, від закономірностей росту та продуктивності деревостанів залежить вибір раціональних способів організації та ведення обліку лісових масивів.

Ведуться роботи з вивчення ходу росту насаджень з різною початковою кількістю дерев. Таблиці росту цих насаджень мають певні переваги для прогнозування показників. Для їх побудови потрібно багато експериментального матеріалу, тому математичні методи здійснюються за допомогою комп'ютерних технологій. Фахівці багатьох країн під час побудови таблиць ходу росту поруч із математичними та комп'ютерними методами широко використовують графіки. На основі типів цих кривих за окремими показниками розробляються моделі ходу росту.

Спосіб моделювання оптимальної густоти насаджень повинен ґрунтуватися на положенні: максимальну продукцію можна отримати у разі поєднання таких умов, як максимальне поглинання сонячної енергії та максимальна зімкненість пологу, що має складатися з дерев, що знаходяться на оптимальній відстані один від одного. Розробляються нові типи моделей ходу, ріст відбивають наслідки змін довкілля. Зіставивши всі дослідження з моделювання продуктивності деревостанів, можна

виявити основні фактори, що впливають на продуктивність: біологічна особливість деревини, генетичні властивості окремих дерев, господарський режим, санітарний стан деревостану. Проте вплив цих факторів вивчений неоднаково.

Аналіз факторів, що впливають на ріст насаджень, також огляд існуючих методів моделювання їх росту дозволяють зробити такі висновки. Вивчення ходу росту насаджень слід проводити за складовими породами як біологічними сукупностями дерев, які підпорядковуються певним закономірностям.

Наявність взаємозв'язку між обліковими показниками, умовами росту та віком насаджень показує, що у будь-який час вихідний стан деревостану багато в чому визначає його подальший хід росту. Весь комплекс факторів, який впливає на ріст насаджень, традиційним методом підбору деревостанів одного природного ряду врахувати практично неможливо, оскільки потрібно складати дуже багато таблиць. У цьому випадку хід росту насаджень можна найточніше відобразити математичною моделлю, що враховує всі можливі стани деревостанів та фактори, що впливають на їх ріст.

Найбільші труднощі у вирішенні проблеми контролю та управління лісовими ресурсами є у напрямках:

- побудова математичних моделей процесів;
- перехід від імітації росту окремих дерев, насаджень до подання об'єктів у вигляді систем;
- перевірка наукових гіпотез та адекватності моделей;
- оцінка прогнозу функціонування об'єктів.

ВИСНОВКИ ДО РОЗДІЛУ 1

В першому розділі проаналізовано предметну область, засоби та технології проектування інформаційної системи обліку лісових насаджень. З допомогою розробленого програмного забезпечення можна буде моделювати динаміку росту окремого дерева та лісових масивів в цілому, проводити оцінку та моделювання параметрів цих дерев на досліджуваній лісовій території.

РОЗДІЛ 2. ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1. Структура системи обліку лісових масивів

Перехід до комп'ютерної обробки даних, автоматизованої системи управління та автоматизованої системи планових розрахунків вимагає розробки інформаційного забезпечення цих систем, що зумовлює створення відповідної системи для обліку дерев: інформації, планування та прийняття рішення. Лісова екологічна система розвивається за умов довкілля. Інформація про стан лісового фонду об'єкта збирається в лісовій екосистемі шляхом різних вимірів (вимірювальна, вибіркова, перечислювальний облік лісу, матеріали аерофотозйомки тощо) і накопичується в системі збору інформації (рис. 2.1). Дані обробляються у системі обробки інформації та надходять у систему планування, де розробляються варіанти розв'язання завдання як закінчений комплекс заходів, що проводяться в об'єкті для досягнення поставленої мети. Чим більше вироблено можливих рішень, тим краще, тому що в цьому випадку не буде втрачено цінної альтернативи для прийняття оптимального варіанту рішення.

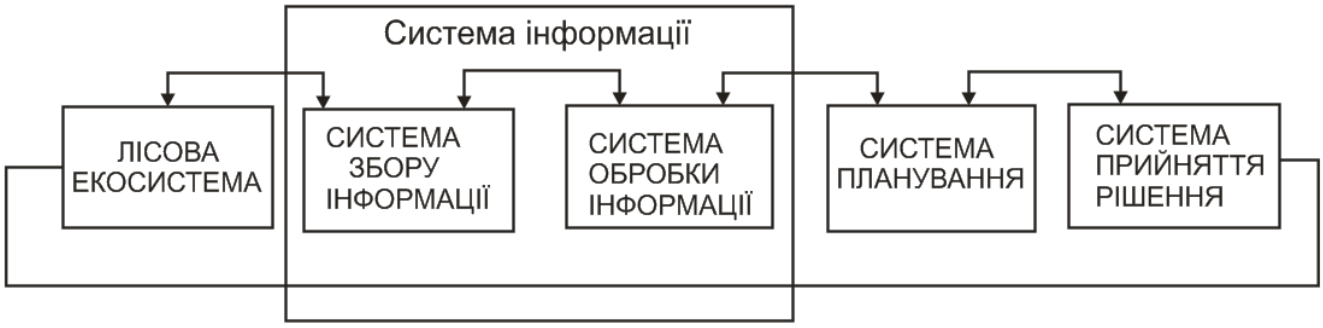


Рис. 2.1. Інформаційна система для обліку лісової інформації.

У системі прийняття рішення, виходячи з аналізів обмежень з урахуванням певного ступеня самостійності в прийнятті рішення та принципів стійкості системи, отримують допустимі альтернативи, з яких відбирають оптимальні з точки зору практичної реалізації критерії прийняття оптимальних рішень.

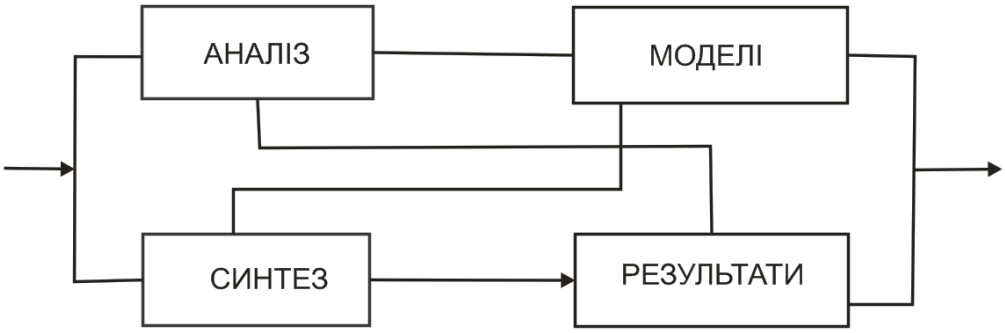


Рис. 2.2. Структура системи інформації, планування та прийняття рішення.

Комп'ютерна техніка у лісовому господарстві використовується понад 40 років. Процес інформатизації лісового господарства зачіпає всі його сфери, які пов'язані з отриманням, обробкою, аналізом та створенням на їх основі нової інформації для прийняття конкретних рішень.

У лісовій галузі накопичений певний досвід застосування автоматизованих систем та ГІС-технологій. Автоматизовані системи дозволяють вирішувати конкретні завдання: досліджують ріст деревостанів, дозволяють автоматизувати робочі місця фахівців лісового господарства.

Розглянемо можливості різних автоматизованих та геоінформаційних систем, що використовуються у лісовому господарстві. З їх допомогою проводять моделювання ходу росту деревостану лісу, автоматизацію розрахунків результатів вимірювань, імітаційне моделювання росту деревостанів, геоінформаційних систем, що застосовуються у лісовому господарстві. Це програми АРМ-Лісокористування, Sosna99, PROBA 2005, ИПК ЛесГИС.

Розроблено автоматизовані системи:

- автоматизоване робоче місце таксатора (комплекс програм АРМ-таксатор);
- автоматизована система управління лісовими ресурсами (АСУ – лісові ресурси).

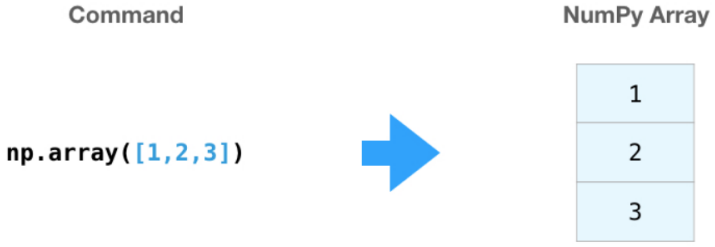
2.2. Використання бібліотеки NumPy при проектуванні системи обліку лісових насаджень

Пакет NumPy є незамінним помічником Python. Він відповідає за аналіз даних, наукові обчислення, призначений для роботи з векторами та матрицями. Робота в

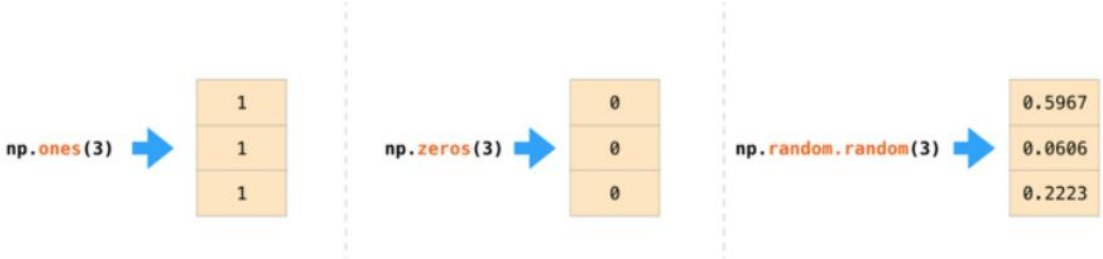
NumPy дає значну перевагу при налагодженні складніших сценаріїв бібліотек. Для роботи з цим модулем Python його потрібно імпортувати:

```
import numpy as np
```

Для створення масивів можна створити масив NumPy, він же `ndarray`, передавши йому список Python, використовуючи `np.array()`. Буде створено такий масив:

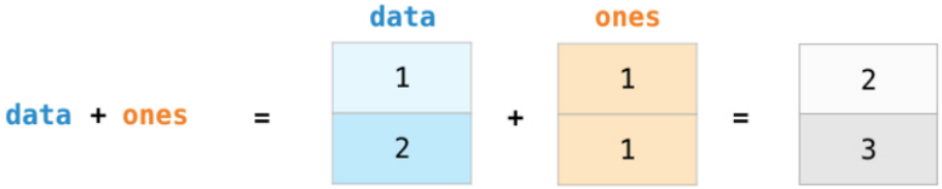


Інколи потрібно, щоб NumPy ініціалізував значення масиву. Для цього NumPy використовує такі методи, як `ones()`, `zeros()` та `random.random()`. Цим методам потрібно передати таку кількість елементів, яку потрібно згенерувати:

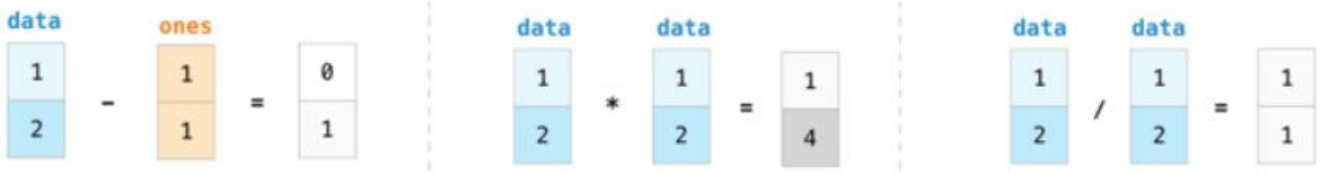


Розглянемо арифметичні операції з масивами NumPy на прикладі створення двох масивів NumPy, які мають назву `data` та `ones`:

При додаванні масивів додаються значення кожного ряду, для цього слід написати: `data + ones`:

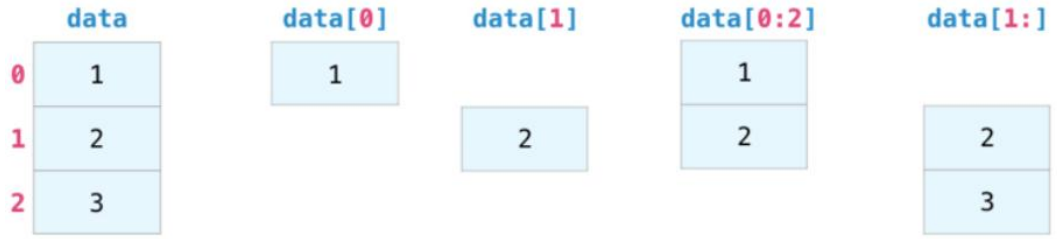


Крім додавання, також можна виконати інші операції:

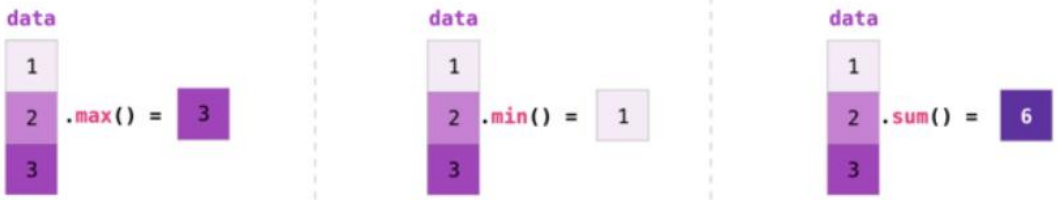


Інколи потрібно виконати якусь арифметичну операцію між масивом і числом. Її також можна назвати операцією між вектором та скалярною величиною:

Розглянемо, що таке індексація масиву NumPy. Його можна розділити на частини та присвоїти їм індекси. Принцип роботи подібний до того, як це відбувається зі списками в Python.



Додатковою перевагою NumPy є наявність у даному модулі функцій агрегування:

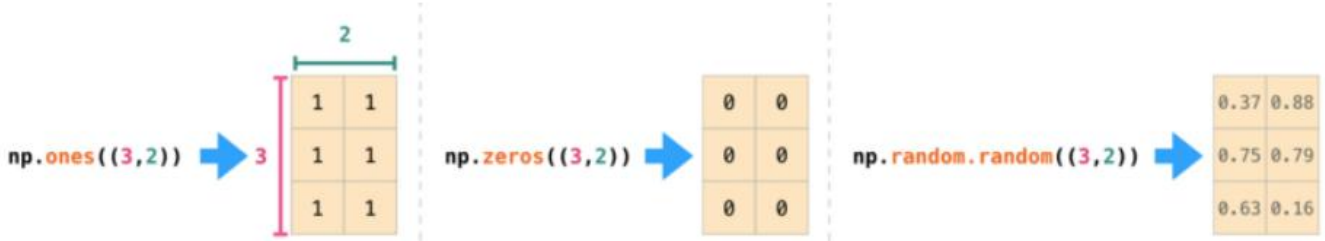


Крім функцій `min()`, `max()`, які відповідають за найбільший, найменший елемент в масиві, інші функції, наприклад `mean()` дозволяє отримати середнє арифметичне; `prod()` – результат множення всіх елементів; `std()`– дана функція необхідна для обчислення середньоквадратичного відхилення. Це лише невелика частина досить великого переліку функцій агрегування NumPy.

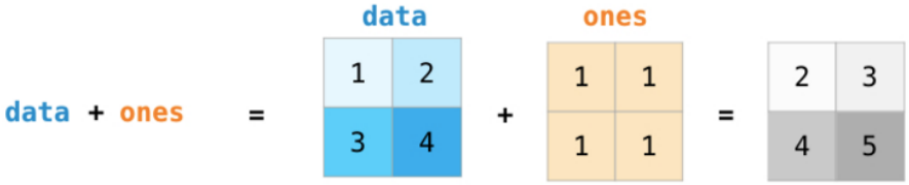
Головною перевагою NumPy є його здатність використовувати зазначені операції із будь-якою кількістю розмірностей. Створені в наступній формі списки зі списків Python можна надіслати NumPy. Він створить матрицю, яка представлятиме дані списки:

```
np.array([[1,2],[3,4]])
```

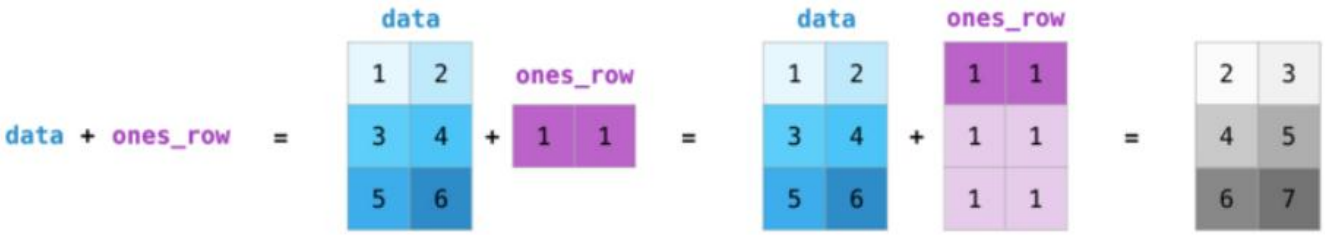
Згадані раніше методи `ones()`, `zeros()` і `random.random()` можна використовувати так довго, як цього вимагає проект. Досить лише додати їм кортеж, у якому буде вказано розмірності матриці, яку створюють.



Розглянемо арифметичні операції з матрицями NumPy. Матриці можна додавати та множити за допомогою арифметичних операторів (+ - * /). При цьому матриці повинні бути одного розміру. NumPy у цьому випадку використовує операції координат:



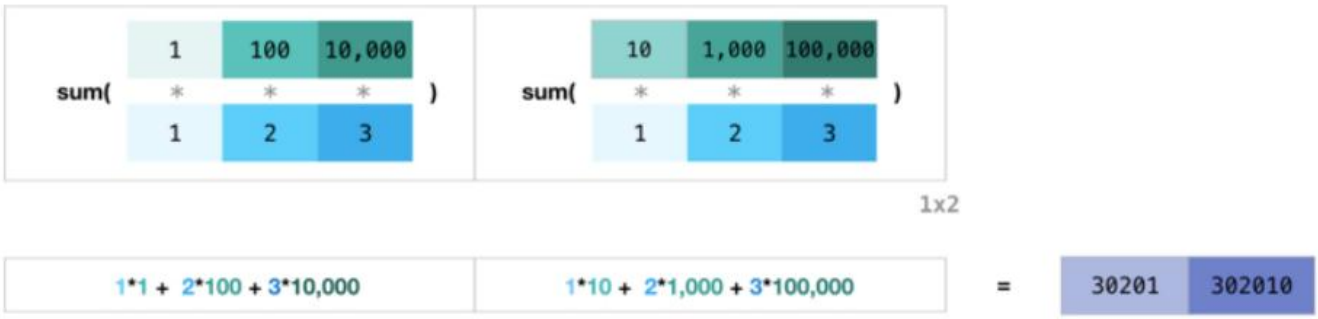
Арифметичні операції над матрицями різних розмірів можливі у випадку, якщо розмірність однієї з матриць дорівнює одиниці. Це означає, що у матриці лише один стовпець чи один ряд. У такому випадку для виконання операції NumPy використовуватиме правила трансляції:



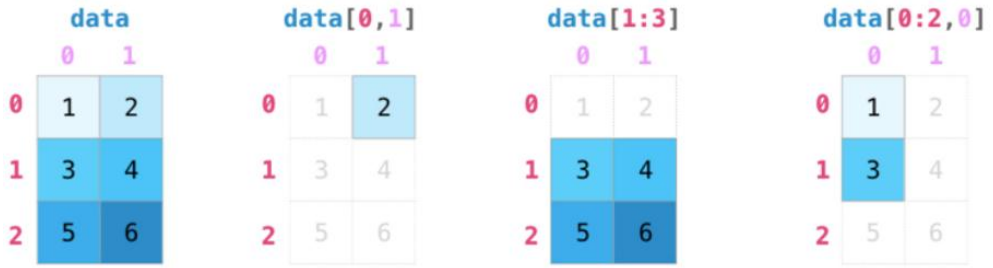
Розглянемо особливості скалярного добутку в NumPy. Головна відмінність із звичайними арифметичними операціями полягає в тому, що при множенні матриць використовується скалярний добуток. У NumPy кожна матриця може використовувати

метод `dot()`. Він застосовується для проведення скалярних операцій з розглянутими матрицями:

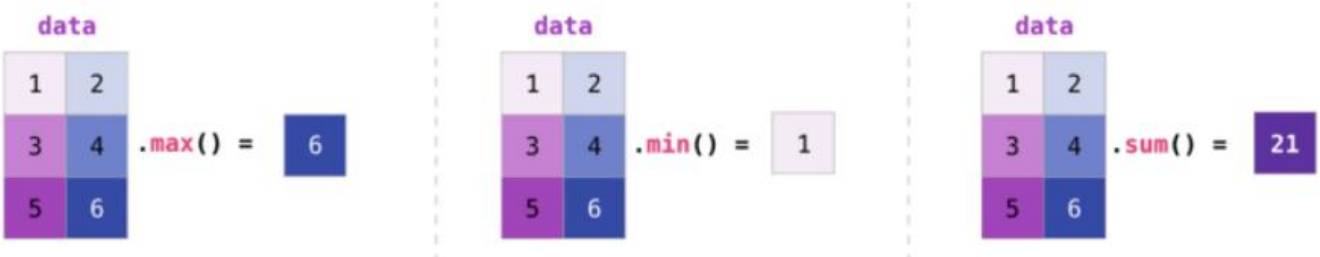
На зображенні вище під кожною фігурою вказана її розмірність. Розмірності обох матриць повинні співпадати з цієї сторони, де вони дотикаються. Візуально подати цю операцію можна так:



Операції індексації та ділення незамінимі, коли потрібно маніпулювати з матрицями:



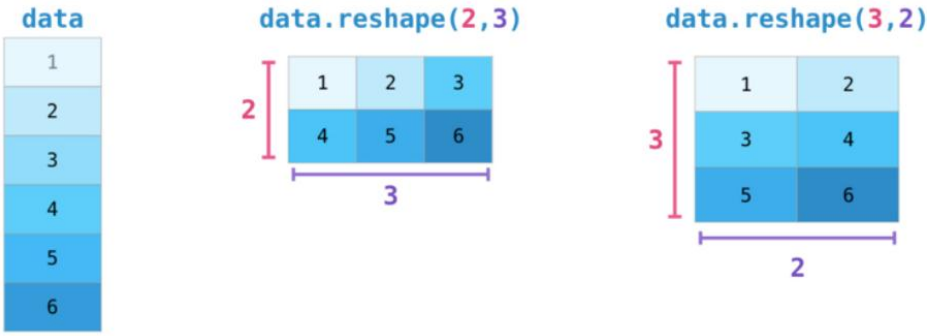
Агрегування матриць відбувається так само, як і агрегування векторів:



Використовуючи параметр `axis`, можна агрегувати як усі значення всередині матриці, а й значення за стовпцями чи рядами.

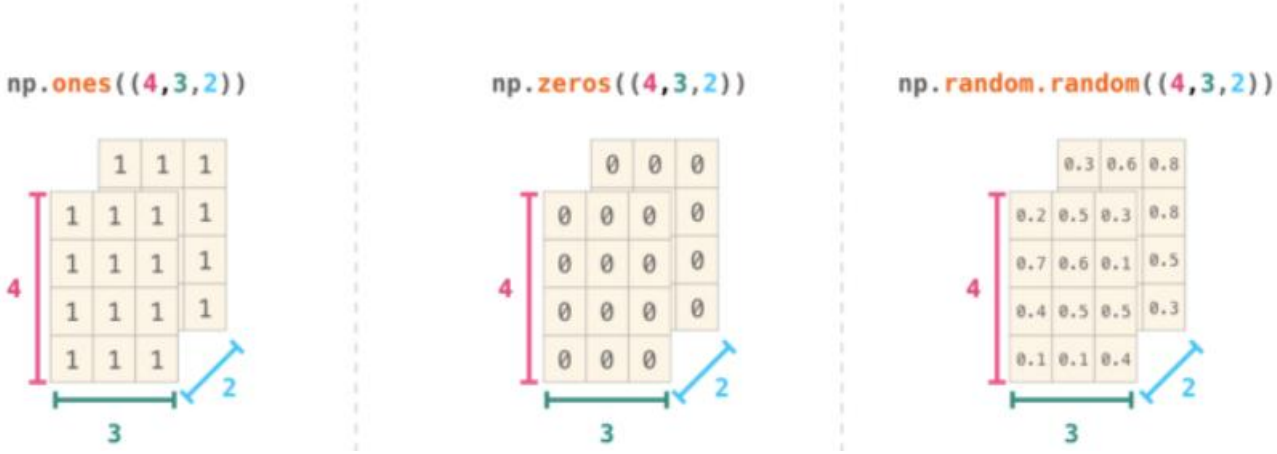
Непоодинокі випадки, коли потрібно транспонувати матрицю. Це може знадобитися при обчисленні скалярного добутку двох матриць. Тоді виникає необхідність наявності розмірностей, що збігаються. У масивів NumPy є властивість, що відповідає за транспонування матриць.

Деякі складніші ситуації вимагають можливості перемикання між розмірностями аналізованої матриці. У таких ситуаціях використовують метод `reshape()`. Потрібно лише передати нові розмірності для матриці:



NumPy може зробити всі перераховані вище операції для будь-якої кількості розмірностей. Структура даних, розташованих центрально, називається `ndarray`, або n-мірним масивом.

У більшості випадків для нової розмірності потрібно додати кому до параметрів функції NumPy:



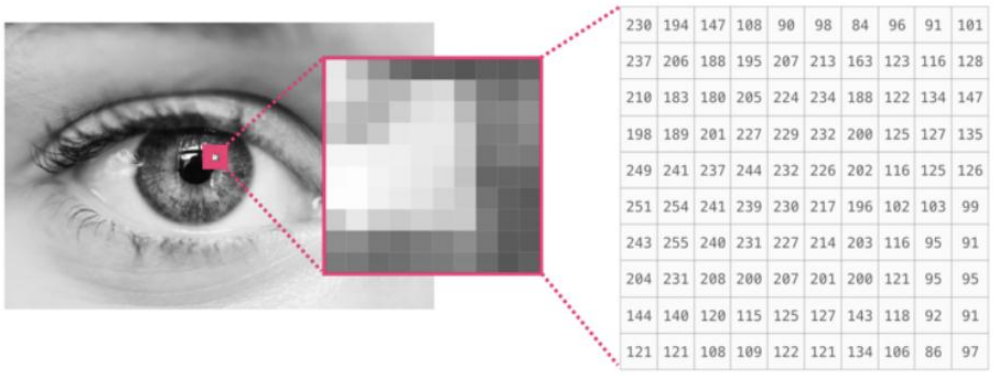
Необхідність застосування математичних формул, які будуть працювати з матрицями та векторами, є головною причиною використання NumPy. Саме тому NumPy має велику популярність серед представників науки. Розглянемо формулу середньоквадратичної помилки, яка відповідає за регресію:

$$MeanSquareError = \frac{1}{n}\sum_{i=1}^{n}(Y\_prediction_i - Y_i)^2$$

В NumPy дана формула запишеться таким чином:

```
error = (1/n) * np.sum(np.square(predictions - labels))
```

Розглянемо обробку зображень у NumPy. Зображення є матрицею пікселів за розміром (висота x ширина). Якщо зображення чорно-біле, тобто представлене у напівтонах, то кожний піксель може бути представлений як єдине число. Як правило між 0 (чорний) та 255 (білий). Для того, щоб вирізати квадрат розміром 10 x 10 пікселів з лівого верхнього кута зображення, потрібно ввести: `image[:10, :10]`. Фрагмент зображення виглядає так:



Якщо зображення кольорове, то кожен піксель представлений трьома числами. За основу береться колірна модель RGB – червоний (R), зелений (G) та синій (B). Знадобиться ще третя розмірність, оскільки кожна клітина вміщує лише одне число. Таким чином, кольорове зображення буде представлене масивом `ndarray` з розмірностями: (висота x ширина x 3).

2.3. Візуалізація результатів досліджень з допомогою бібліотеки Matplotlib

Для початку потрібно імпортувати в Python модуль Matplotlib. Для побудови графічних залежностей використовують один з його підмодулів pyplot.

```
import matplotlib.pyplot as plt
```

При побудові графіків з допомогою Matplotlib також використовується модуль Numpy. Тоді графік можна побудувати таким чином. Для початку створюється матриця x та відповідні їй матриці y1 та y2, які будуть відповідати за побудову двох ліній на графіку:

```
import numpy as np
x=np.arange(0, 5, 0.25)
y1=x**2
y2=x**3
```

Після цього можна будувати графічні залежності. За це відповідають наступні рядки коду:

```
plt.plot(x, y1)
plt.plot(x, y2)
plt.show()
```

Отримаємо такий графік:

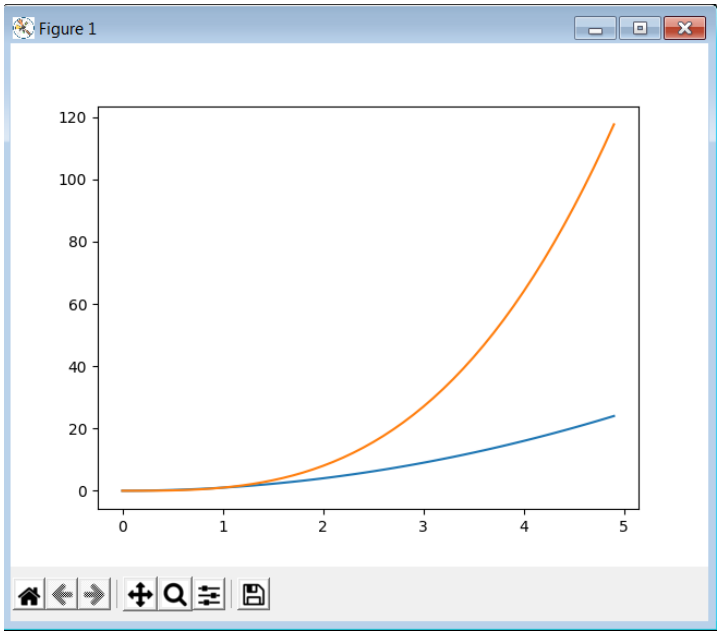


Рис. 2.3. Графік залежностей y1=*f*(x) та y2=*f*(x), побудованих у Matplotlib.

Для зміни кольору лінії графіка, додавання підписів осей по горизонталі та по вертикалі та назви графіка додатково потрібно ввести наступні рядки коду:

```
plt.plot(x, y1, "red")
plt.plot(x, y2, "green")
plt.title("Назва графіка")
plt.xlabel("Горизонтальна вісь")
plt.ylabel("Вертикальна вісь")
plt.show()
```

За зміну кольору лінії графіка на червоний (red) відповідає такий рядок коду:

```
plt.plot(x, y, "r")
```

За підписи горизонтальної та вертикальної осей відповідає такий фрагмент коду:

```
plt.xlabel("Горизонтальна вісь")
plt.ylabel("Вертикальна вісь ")
```

За назву графіка – відповідно такий рядок:

```
plt.title("Назва графіка")
```

Після цього побудований графік виглядатиме наступним чином:

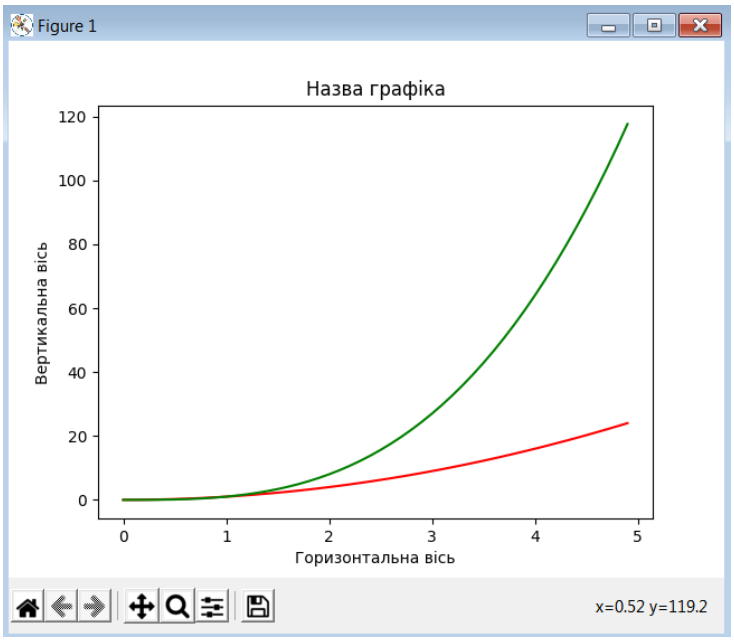


Рис. 2.4. Графік із зміненими параметрами.

Можна відобразити маркери на лініях графіку, а також добавити легенду та сітку. За це відповідають такі рядки коду:

```
plt.plot(x, y1, "r", label="y1=x^2", marker="o")
plt.plot(x, y2, "g", label="y2=x^3", marker="s")
plt.legend()
plt.grid()
```

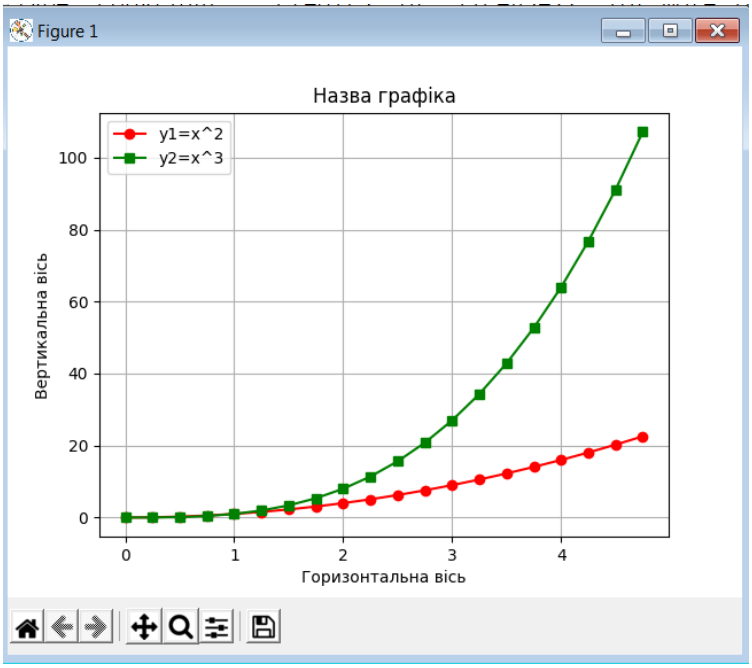


Рис. 2.5. Кінцевий графік із зміненими параметрами.

2.4. Модуль Tkinter для побудови графічного інтерфейсу програми

При проектуванні графічного інтерфейсу програми мовою Python можна використовувати такі бібліотеки:

- Tkinter;
- PyQt;
- WxPython;
- Kyvi.

В даній роботі для проектування графічного інтерфейсу програми для обліку лісових насаджень було використано бібліотеку Tkinter, з якої було використано такі віджети:

- `Label` – напис;
- `Button` – командна кнопка;
- `Entry` – текстове поле;
- `PhotoImage` – об’єкт зображення, який розміщується у інших віджетах.
- `Scrollbar` – віджет, який відповідає за вертикальнульну чи горизонтальну смуги прокрутки;
- `Menu` – віджет, який відповідає за створення меню програми;
- `Combobox` – випадаючий список;
- `Treeview` – таблиця.

Детально відомості про роботу з даними віджетами та проектування інтерфейсу програмного забезпечення для обліку та моделювання лісових насаджень приведені в розділі 4.

ВИСНОВКИ ДО РОЗДІЛУ 2

У другому розділі приведено відомості про існуючі системи обліку лісових насаджень та визначення їх параметрів. Описано технології розробки даних продуктів, а також описано принципи їх функціонування. Розглянуто бібліотеки та модулі мови Python (Tkinter, NumPy, Matplotlib), які відповідають за створення інтерфейсу програмного продукту, проведення обчислень індексів математичних залежностей, а також візуалізації результатів обчислень для програми, яка проводить облік та моделювання лісових насаджень.

РОЗДІЛ 3. МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1. Принципи математичного моделювання ходу росту лісових насаджень

Математичні методи досліджень є важливою складовою наукового методу пізнання. Сьогодні комп'ютерна техніка є однією із визначальних чинників науково-технічного прогресу. Застосування її у лісовому господарстві пов'язано з розробкою математичних моделей росту лісу, створенням обчислювальних алгоритмів і програмного забезпечення як загального, так і спеціально орієнтованого характеру. На сучасному етапі накопичення матеріалу переважає його теоретичне осмислення. Математичне моделювання привчає дослідників до логічного аналізу. Дослідник при побудові моделі повинен сформулювати чітку мету дослідження, вихідні гіпотези, виділити фактори, що впливають на процес, розглянути наслідки тощо. Математичне моделювання росту насаджень є основною складовою обліку з позицій системного підходу, сучасного напряму комп'ютерного моделювання росту лісу.

Математичне моделювання росту лісу є новим напрямком. Як у більшості нових напрямів перша увага була сконцентрована на вирішенні вузьких, спеціальних питань, а не в широкому сенсі на перспективу застосування моделей у системі управління лісовими ресурсами. Системний підхід до моделювання росту лісу на комп'ютерах пов'язаний з переглядом ідей та способів моделювання.

Моделі росту та продуктивності насаджень потрібні для різних аспектів контролю та управління лісами: оцінки насаджень, варіантів догляду за лісом, прогнозування продуктивності деревостанів, оцінки продуктивності умов росту.

Головне ж призначення математичних моделей росту насаджень – забезпечити даними для аналізу та перевірки гіпотез щодо різних варіантів ведення лісового господарства. Спільно з моделями оптимізації моделі росту насаджень дають ключову інформацію в прийнятті правильних рішень в управлінні лісами.

Практично всі моделі мають одну загальну мету: виробляти у певній точці або точках часу (віку) дані про стан насадження. Існують три основні принципи при моделюванні росту насаджень.

Перший принцип передбачає, що основною одиницею моделювання є окреме дерево. Для розробки моделі росту насадження необхідні дані обліку частин стовбура дерева, оцінки біологічної конкуренції дерев та їх просторового розміщення на площі в системі координат. Другий принцип передбачає, що основна одиниця моделювання – окреме дерево. Змінними в моделі представлені облікові ознаки дерев без урахування їх просторового розміщення і даних обліку частин деревного стовбура. Третій принцип моделювання передбачає: основною одиницею моделювання є деревостан і моделі будуються для сукупності насаджень за їхніми середніми обліковими показниками.

Моделі першого типу створюються на основі інформації про ріст окремих дерев у насадженні: індекс умов виростання, фактор конкуренції дерев, вимірювання ширини та довжини крони, відстань між деревами, аналіз ходу росту деревного стовбура, поточний приріст за діаметром та висотою вздовж стовбура, положення дерева в системі координат.

Для прогнозування ходу росту окремого дерева використано чотири основні змінні:

- SI (site index) – індекс типів умов проростання чи верхня висота деревостану;
- A – вік дерев (років);
- D – діаметр дерева на висоті 1,3 м (см);
- GSI – індекс простору, кількісна зміна біологічної конкуренції між деревами за площу живлення, світло і т.д.

Індекс GSI обчислюється шляхом оцінки кута між деревом та його конкурентами. Методичний підхід полягає у встановленні зв'язків між шириною крони та діаметром дерев, що ростуть на відкритому просторі (індекс GSI=0) та пригноблених (GSI=100), які показують відповідний кут (рис. 3.1).

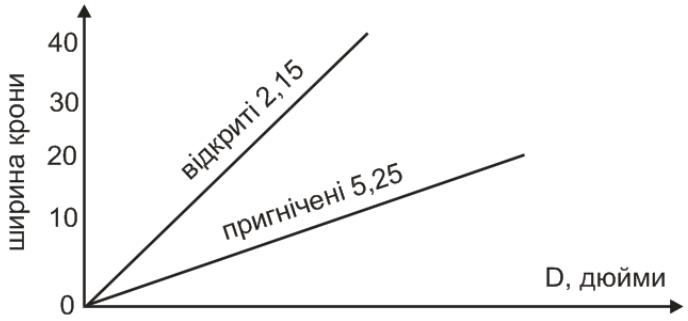


Рис. 3.1. Зв'язок між шириною крони та діаметром дерев, що ростуть на відкритому просторі.

Між прямими лініями (рис. 3.1) знаходяться межі зростаючого простору дерева. Таким чином, якщо кут між деревом та його конкурентом менший за 2,15, то конкуренція відсутня (GSI=100), якщо кут дорівнює або більший за 5,25, то конкуренція максимальна (індекс GSI=0). Індекс обчислюється на комп'ютерах за картою просторового розподілу дерев за площею насадження.

Моделі другого типу розробляються з використанням залежностей відносного приросту за висотою, діаметром та об'ємом від облікових показників дерева та насадження, факторів навколишнього середовища (середньої відстані між деревами, температури та довжини сезону росту, величини опадів тощо). У цих моделях широко використовуються функції розподілу дерев за діаметром, висотою та іншими ознаками. Моделі другого типу вимагають менше інформації і можуть бути використані під час створення системи прийняття рішень та оцінки альтернативних варіантів ведення лісового господарства. Серйозний недолік цих моделей – відсутність надійності у прогнозуванні поточного приросту деревостанів та імітації росту насаджень.

Моделі третього типу використовують як моделі ходу росту. Сучасні комп'ютери дозволяють розробляти складні регресійні моделі. В математичних моделях використовуються моделі ходу росту лісових насаджень. Вони служать базою для багатьох розрахунків при обліку лісів та в прогнозуванні. Існуючі моделі ходу росту (МХР) по призначенню діляться на:

- МХР нормальних насаджень;
- МХР модальних насаджень;

- МХР оптимальних насаджень;
- МХР з різною густотою та сумою площ перерізів.

Ці таблиці ходу росту призначені для:

- характеристики росту та розвитку деревостанів з віком;
- складання облікових даних лісових масивів.

Розроблення моделей ходу росту – це складний та тривалий вид облікових робіт. Автоматизація робіт по проектуванню моделей ходу росту усуває ці труднощі та підвищує ефективність роботи.

3.2. Математична модель ходу росту окремого дерева

Знання про закономірності зміни облікових показників окремого дерева з часом дають підстави для характеристики динаміки росту та розвитку деревостанів. Для вирішення цього завдання використовується методика аналізу ходу росту деревного стовбура. Для аналізу ходу росту деревного стовбура було проведено виміри модельних дерев. Обробка експериментальних даних здійснювалася в даній програмі обліку лісових насаджень (розділ 4). Вона є інструментом для вводу, обробки та зберігання вхідної та розрахункової інформації при проведенні аналізу ходу росту дерева. Інтерфейс програми та приклад розрахунку по модельному дереву представлені в розділі 4 (пункт 4.3). Результати обробки експериментальних даних по модельному дереву записуються в базу даних та служать для подальших статистичних розрахунків.

Закономірності зміни облікових показників лісових насаджень описуються різними математичними функціями, до яких пред’являються такі вимоги:

1) графік повинен проходити через початок координат, $f(A) = 0$ при $A = 0$;

2) функція росту повинна бути зростаючою, $f_1(A)$ більша чи рівна 0 при $A > 0$;

3) границя функції при необмеженому зростанні аргументу повинна наближатися до асимптоти: $\lim f(A) = \max F \rightarrow \infty$;

4) графік біжучого приросту повинен виходити з початку координат: $f_1(A) = 0$ при $A = 0$;

5) функція росту повинна мати точку перегину: $f_{11}(A) = 0$ при $A > 0$.

Однією з оптимальних функцій, яка задовольняє необхідним вимогам для опису загальних закономірностей росту дерев, є функція Мітчерліха:

$$T_{мод} = T_{max}\left(1 - e^{-A \cdot C_1}\right)^{C_2} \quad (3.1)$$

де $T_{мод}$ – обліковий показник, який моделюється: D (см), H (м); S (м^2); V (м^3);

T_{max} – асимптотичне значення облікового показника для даного деревостану;

e – основа натурального логарифма;

A – вік дерев, років;

C_1 – параметр росту;

C_2 – параметр форми кривої.

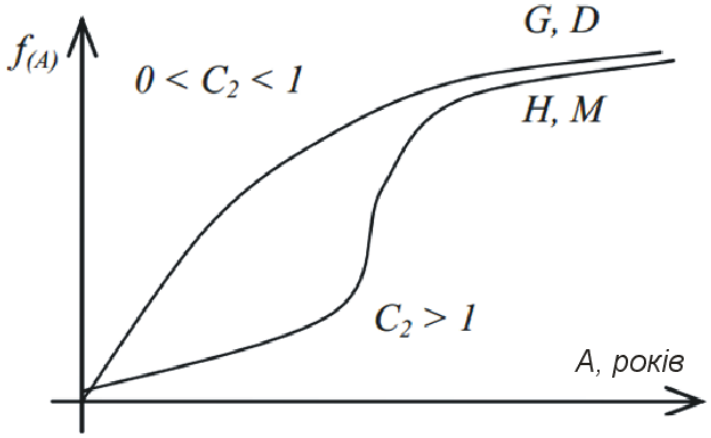


Рис. 3.2. Криві відображення взаємозв’язків.

Якщо C_2 знаходиться в межах від 0 до 1 ($0 < C_2 < 1$), то функція не має точки перегину. Ця крива добре описує значення сум площ перерізів та діаметрів.

Крива росту має S-подібний вид при $C_2 > 1$ з точкою перегину $A_{mn} = \ln C_2 / C_1$, S-подібний вид кривої описує значення висот і запасів.

Функція Мітчерліха використовується для опису динаміки зміни облікових показників як окремого дерева, так і лісових масивів (рис. 3.2). Аналіз відомих функцій росту показує, що хід росту живих організмів можна описати множиною

функцій росту з різною точністю. Для моделювання динаміки облікових показників дерев добрі результати дає функція Мітчерліха.

Узагальнююча модель по діаметру модельних дерев залежно від віку по всій вибірковій сукупності характеризується високими коефіцієнтами кореляції:

$$d = 56,43 \cdot (1 - \exp(-0,0123 \cdot A))^{1,3015}. \quad (3.2)$$

Аналогічні моделі за висотою модельних дерев залежно від віку та діаметра по всій вибірковій сукупності оцінюються високими коефіцієнтами кореляції.

$$h = 17,79 \cdot (1 - \exp(-0,0303 \cdot A))^{1,2032}, \quad (3.3)$$

$$d = 23,64 \cdot (1 - \exp(-0,0296 \cdot d))^{0,7541}. \quad (3.4)$$

Моделювання облікових показників за результатами вимірювань за методикою повного аналізу ходу росту окремого дерева підтверджує правильність вибору функції Мітчерліха для передбачення співвідношень висот та діаметрів залежно від віку та висот від діаметрів.

3.3. Математична модель ходу росту лісових насаджень

При моделюванні закономірностей росту лісових масивів використовуються функції росту: степенева, експоненціальна, функція Мітчерліха. Існує кілька видів моделей ходу росту, що поділяються за призначенням: для нормальних насаджень, модальних насаджень, оптимальних насаджень різного класу бонітету та різних сум площ перерізів. Облікові дані ходу росту лісових масивів призначені для характеристики та прогнозування їх росту , виявлення закономірностей їх розвитку.

Зміна суми площ перерізів лісового масиву на 1 га:

$$S = 11,25 \cdot (1 - \exp(-0,0147 \cdot h))^{-0,2325}. \quad (3.5)$$

Вплив класу бонітету n, віку a та абсолютної повноти g на значення середніх висот h та середній діаметрів d лісового масиву мають вигляд:

$$h = 1,20 - 12,28 / g + 0,228 \cdot a + 1129,3 / n. \quad (3.6)$$

$$d = 8,25 - 32,10 / g + 0,0320 \cdot a + 6377,6 / n. \quad (3.7)$$

Показники суми площ перерізів лісового масиву за кожною групою класу бонітету:

$$\sum S_{рідкі} = \sum g_{ср} - t_{0,95} \cdot \frac{\sum g_{ср} \cdot \left[11,25 \cdot (1 - \exp(-0,0147 \cdot h))^{-0,2325}\right]}{100} \quad (3.8)$$

$$\sum S_{середні} = \sum g_{ср} \,. \quad (3.9)$$

$$\sum S_{густі} = \sum g_{ср} + t_{0,95} \cdot \frac{\sum g_{ср} \cdot \left[11,25 \cdot (1 - \exp(-0,0147 \cdot h))^{-0,2325}\right]}{100} \quad (3.10)$$

Систему регресійних рівнянь таблиць ходу росту по класах бонітету для породи сосна наведено у табл. 3.1.

Табл. 3.1. Математичні моделі ходу росту лісових культур сосни

I бонітет
$h = 21,506 \cdot (1 - \exp(-a \cdot (0,0023 + 0,0003 \cdot a)))^{(0,9761+0,0014 \cdot a+0,000001 \cdot a^2)}$
$d = 29,55 \cdot (1 - \exp(-a \cdot (0,0004 + 0,0004 \cdot a)))^{(1,0453+0,0006 \cdot a+0,000003 \cdot a^2)}$
$S = 174,174 \cdot (1 - \exp(-a \cdot (0,0004 + 0,0003 \cdot a)))^{(0,9909+0,0016 \cdot a)}$
$V = S / (1,07641 + 0,40049 \cdot h)$
II бонітет
$h = 20,961 \cdot (1 - \exp(-a \cdot (0,0031 + 0,0003 \cdot a)))^{(1+0,00001 \cdot a)}$
$d = 28,956 \cdot (1 - \exp(-a \cdot (0,0186 + 0,0002 \cdot a)))^{(2,002+0,0001 \cdot a)}$
$S = 195,409 \cdot (1 - \exp(-a \cdot (0,0026 + 0,0003 \cdot a)))^{(0,9991+(-0,0000 \cdot a))}$
$V = S / (1,09950 + 0,39970 \cdot h)$

III бонітет
$h = 21,320 \cdot (1 - \exp(-a \cdot (0,0027 + 0,0003 \cdot a)))^{(0,9087+0,00001 \cdot a)}$
$d = 27,45 \cdot (1 - \exp(-a \cdot (0,0186 + 0,0003 \cdot a)))^{(1,9823+0,0007 \cdot a+0,000001 \cdot a^2)}$
$S = 228,314 \cdot (1 - \exp(-a \cdot (0,0248 + 0,0001 \cdot a)))^{(2,0155+(-0,0006 \cdot a))}$
$V = S / (1,001976 + 0,41138 \cdot h)$
IV бонітет
$h = 22,404 \cdot (1 - \exp(-a \cdot (0,0016 + 0,0003 \cdot a)))^{(0,8926+0,0009 \cdot a+0,000001 \cdot a^2)}$
$d = 24,32 \cdot (1 - \exp(-a \cdot (0,0003 + 0,0004 \cdot a)))^{(0,8653+0,0005 \cdot a+0,000002 \cdot a^2)}$
$S = 165,211 \cdot (1 - \exp(-a \cdot (0,0003 + 0,0002 \cdot a)))^{(0,8764+0,0013 \cdot a)}$
$V = S / (1,05541 + 0,38163 \cdot h)$
V бонітет
$h = 23,167 \cdot (1 - \exp(-a \cdot (0,0014 + 0,0002 \cdot a)))^{(0,7654+0,0006 \cdot a+0,000001 \cdot a^2)}$
$d = 22,14 \cdot (1 - \exp(-a \cdot (0,0002 + 0,0003 \cdot a)))^{(0,8312+0,0003 \cdot a+0,000001 \cdot a^2)}$
$S = 154,183 \cdot (1 - \exp(-a \cdot (0,0002 + 0,0001 \cdot a)))^{(0,8296+0,0011 \cdot a)}$
$V = S / (1,02354 + 0,26378 \cdot h)$

Отримана система регресійних рівнянь характеризує хід росту в динаміці по класах бонітету для сосни.

Розробка облікових даних сум площ перерізів полягає в тому, що запас лісового масиву породи є функцією сум площ перерізів та середніх видових чисел, а при однакових середніх висотах запаси мають однакові значення.

Коефіцієнти зміни сум площ перерізів лісових масивів розраховувалися за класами бонітету за таким співвідношенням:

$$S = 100 \cdot \sigma / \sum g_{cp}, \quad (3.11)$$

а модельні значення зміни сум площ перерізів:

$$S_{\sum g} = 11,25 \cdot (1 - \exp(-0,0147 \cdot h))^{-0,2325}, \quad (3.12)$$

де σ – середньоквадратичне відхилення.

Для обчислення максимального значення сум площ перерізів використано таке співвідношення:

$$\sum S_{max\ i} = \sum g_{cp} + t_{0,95} \cdot \frac{\sum g_{cp} \cdot \left[11,25 \cdot (1 - \exp(-0,0147 \cdot h))^{-0,2325}\right]}{100} \quad (3.13)$$

де $\sum S_{max\ i}$ – максимальне значення сум площ перерізів лісового масиву, м^2.

Для опису закономірностей зміни сум площ перерізів в залежності від висоти лісового масиву використовували функцію Мітчерліха, що має вигляд:

$$\sum S_{cp} = 31,61 \cdot (1 - \exp(-0,0894 \cdot H))^{0,7072}. \quad (3.15)$$

Значення запасу лісового масиву визначає співвідношення:

$$V = \sum S_{max\ i} \cdot H \cdot F. \quad (3.16)$$

В результаті розрахунків отримали математичну модель росту лісових насаджень, яка потрібна для знаходження модельних облікових даних ходу росту дерев.

ВИСНОВКИ ДО РОЗДІЛУ 3

У третьому розділі розроблено математичну модель для моделювання росту лісових насаджень, яка є базою для проектування програмного забезпечення, яке складається з програмних модулів для забезпечення інтерактивної роботи, прийому вхідних даних, оцінки параметрів лісових деревостанів, моделювання ходу росту лісових масивів.

Моделювання облікових показників за результатами вимірювань за методикою повного аналізу ходу росту окремого дерева підтверджує правильність вибору функції Мітчерліха для передбачення співвідношень висот та діаметрів в залежності від віку.

РОЗДІЛ 4. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

4.1. Алгоритм розробки облікових даних ходу росту лісових масивів

Для розробки програмного забезпечення для обліку даних ходу росту лісових масивів використано математичну модель (розділ 3), яка базується на функції росту Мітчерліха з параметрами, які змінюються в динаміці. Спираючись на аналіз функцій росту для проектування моделей та отримання облікових даних моделювання росту лісових масивів різного класу бонітету, було розроблено наступний алгоритм.

1. На основі облікових даних формується база даних лісових масивів для досліджуваної ділянки лісу. Основні показники: порода, вік (років), висота (м), діаметр (см).

2. За обліковими даними (вимірами) виявляють закономірність росту лісових масивів висот, діаметрів, суми площ перерізів, запасу деревини. На наступному етапі по згрупованих за класом бонітету лісових масивах отримують дані облікові показники.

3. Виявляють взаємозв’язок зміни суми площ перерізів та запасів лісових масивів з віком.

4. Виявляють вплив класу бонітету на ріст дерев по висоті та діаметру.

5. Для кожного класу бонітету формується база облікових та розрахункових показників: вік A (років), висота H (м), діаметр D (см), сума площ перерізів S ($м^2$), запас V ($м^3$).

6. Проводять моделювання ходу росту досліджуваного лісового масиву в програмі обліку лісових насаджень.

7. Проводять аналіз та оцінку результатів моделювання ходу росту лісового масиву на досліджуваній ділянці лісу.

4.2. Проектування програми моделювання ходу росту лісових масивів

Дана програма призначена для моделювання ходу росту лісових насаджень, в якій використовується математична модель їх ходу росту. В ній можна провести моделювання динаміки облікових показників даних лісових насаджень. При проектуванні програми спочатку потрібно проаналізувати вхідні дані. Після цього на основі математичної моделі ходу росту лісових насаджень отримати ці моделі в аналітичному вигляді по висоті, діаметру, запасу та сум площ перерізів.

Вхідними даними програми є показники деревостанів, які отримані на досліджуваних лісових ділянках обліковими способами. Ці лісові ділянки приводяться до одного природного ряду (класу бонітету), для якого потрібно отримати модельні дані ходу росту.

З допомогою даної програми отримують ріст лісових масивів за математичною моделлю, в якій можна проводити аналіз росту лісових насаджень по отриманих параметрах та графічних залежностях. Показниками по кожному деревостану для отримання облікових даних росту є вік дерев (років), їх висота (м), діаметр (см), сума площ перерізів ($м^2$/га), запас ($м^3$/га),. Результатом виконання програми є графіки зміни висоти, діаметра, запасу та суми площ перерізів стовбурів, а також розраховані дані по об’єкту побудови моделей ходу росту.

4.2.1. Проектування вікна програми

Проектування програми мовою Python починається зі створення графічного інтерфейсу. Для цього в Python потрібно імпортувати модуль Tkinter, який відповідає за побудову графічного інтерфейсу програми:

```
from tkinter import *
```

Для створення головного вікна програми відповідають наступні рядки коду:

```
root = Tk()
root.title("Облік лісових масивів")
root.resizable(False, False)
root.geometry("1350x650")
root.iconbitmap("Forest.ico")
```

Головне вікно буде мати назву `root`. З допомогою методу `title` задають назву програми: Облік лісових масивів. З допомогою методів `geometry` задають розміри вікна по ширині та по висоті, а з допомогою методу `resizable` забороняють змінювати розміри вікна програми. З допомогою методу `iconbitmap` задають піктограму програми, яка буде відображатися у вікні вгорі зліва біля її назви. Для цього файл, який буде відповідати за піктограму програми, повинен мати розширення `.ico` і знаходитися в тій же папці, що й головний скрипт програми `Oblik.py`.

Щоб програму було видно на екрані, останній рядок скрипта має бути таким:

```
root.mainloop()
```

4.2.2. Проектування меню програми

Для створення ієрархічного меню в Tkinter використовують віджет Menu. Меню може містити багато елементів, причому ці елементи також можуть представляти собою меню та містити інші елементи. В залежності від того, який тип елементів потрібно добавити в меню, буде відрізнятися метод, який використовується для їх добавлення.

Доступні наступні методи:

- `add_command(options)`: добавляє елемент меню через параметр options;
- `add_cascade(options)`: добавляє елемент меню, який в свою чергу може представляти собою підменю.

```
main_menu = Menu()
main_menu.add_cascade(label="Файл")
main_menu.add_cascade(label="Переглянути")
main_menu.add_cascade(label="Редактувати")
main_menu.add_cascade(label="Друк")
main_menu.add_cascade(label="Вихід")
root.config(menu=main_menu)
```

Для додавання пунктів меню у об’єкта Menu викликається метод `add_cascade()`. В цей метод передаються параметри пункту меню, в даному випадку вони представлені текстовою міткою, яка встановлюється через параметр `label`.

Але просто створити меню ще недостатньо. Його потрібно встановити для поточного вікна за допомогою параметра menu в методі `config()`. Після створення головного меню можна створити підменю:

```
main_menu = Menu()
file_menu = Menu()
file_menu.add_command(label="Створити")
file_menu.add_command(label="Відкрити")
file_menu.add_command(label="Зберегти")
file_menu.add_command(label="Вихід")
main_menu.add_cascade(label="File", menu=file_menu)
root.config(menu=main_menu)
```

Тут визначається підменю `file_menu`, яке добавляється в перший пункт основного меню завдяки установці опції `menu=file_menu`:

```
main_menu.add_cascade(label="File", menu=file_menu)
main_menu = Menu()
file_menu = Menu(font=("Verdana", 13, "bold"), tearoff=0)
file_menu.add_command(label="Новий")
file_menu.add_command(label="Зберегти")
file_menu.add_command(label="Відкрити")
file_menu.add_command(label="Вихід")
main_menu.add_cascade(label="File", menu=file_menu)
main_menu.add_cascade(label="Edit")
main_menu.add_cascade(label="View")
```

Основною особливістю елементів меню є здатність реагувати на дії користувача. Для цього у кожного елемента меню потрібно встановити параметр `command`, який встановлює посилання на функцію, що виконується при виборі елемента меню.

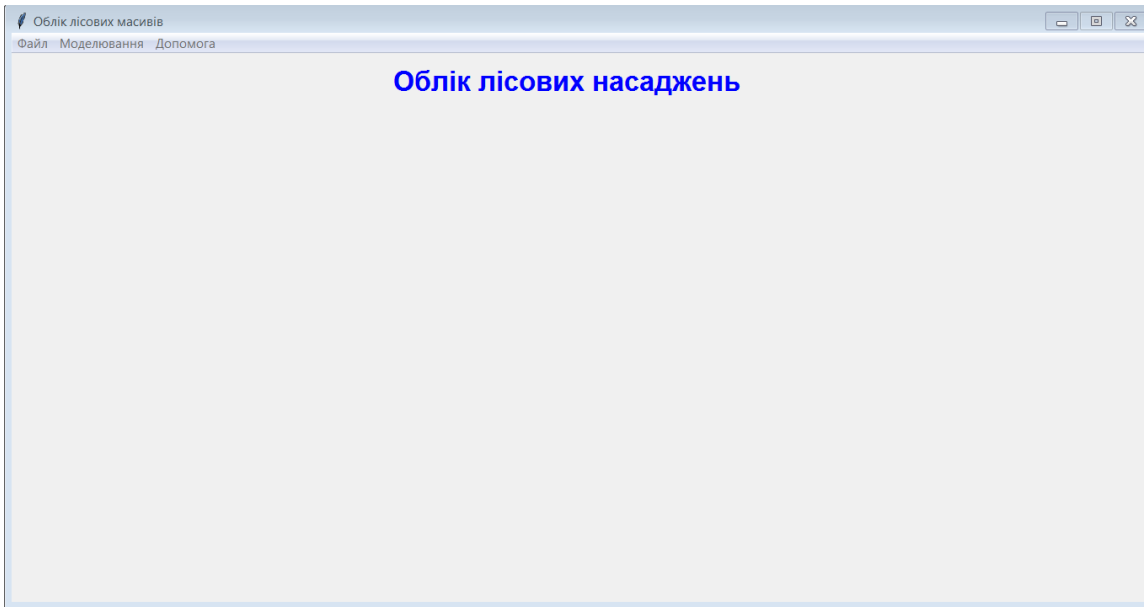


Рис. 4.1. Стартове вікно програми для обліку лісових насаджень.

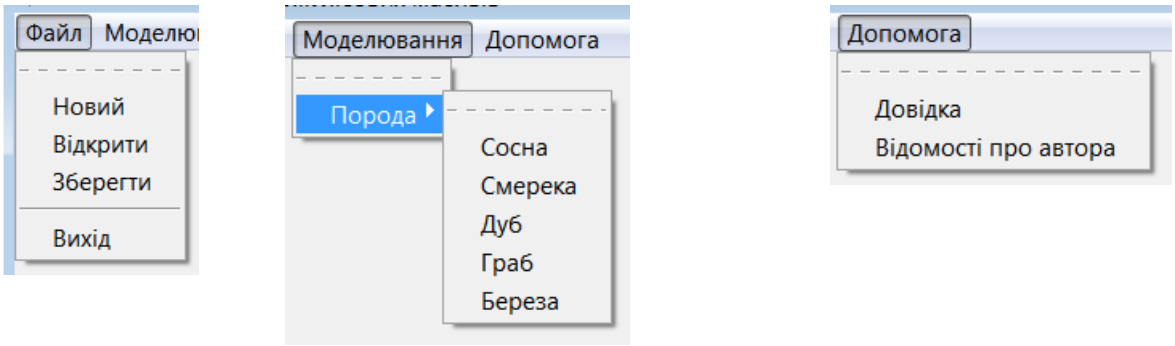


Рис. 4.2. Структура меню та підменю головного вікна.

4.2.3. Розміщення віджетів у вікні програми

Віджети у вікні програми можуть бути розміщені з допомогою трьох менеджерів геометрії: `pack()`, `grid()`, `place()`. В даній роботі використано метод розміщення віджетів `place()`, який їх розміщує по координатах. Використання даного менеджеру використовується в Tkinter через метод `place` віджетів.

Цим методом віджету вказується його положення в абсолютних значеннях (в пікселях). Також абсолютно можна задавати розмір самого віджета. Основними параметрами методу `place` є:

`x`, `y` – абсолютна позиція в пікселях. Значення по замовчуванню прирівнюються до нуля.

width, height – абсолютний розмір віджета в пікселях. Значення по замовчуванню (коли даних опцій немає) прирівнюються до природного розміру віджета, тобто до такого, який визначається при його створенні.

anchor (якір) – визначає частину віджета, для якого задаються координати. Може приймати значення: N, NE, E, SE, SW, W, NW, CENTER. По замовчуванню використовується NW (верхній лівий кут).

При проектуванні інтерфейсу програми для обліку лісових насаджень було використано наступні віджети:

- Button – командна кнопка;
- Entry – однорядкове текстове поле;
- Label – напис;
- Treeview – таблиця;
- Combobox – випадаючий список;

Віджет Entry призначений для вводу в програму та виводу з неї числової та символьної інформації. За це відповідають методи get() та insert() відповідно для даного віджету. Для очистки текстового поля використовують метод delete(). Приклад створення текстового поля показано нижче. В даному прикладі текстове поле із назвою txt1 буде розміщене в батьківському віджеті (вікні з назвою Window1, ширина поля 30 символів, колір заднього плану (фону) білий, колір переднього плану (шрифта) синій. Розміщений даний віджет буде на відстані 20 пікселів від лівого краю вікна та на 150 пікселів від його верхнього краю.

```
    txt1 = Entry(root, width = 30, bg="#ffffff" fg="blue",
font=("Arial 9 bold"))
    txt1.place(x=20, y=150)
```

Віджет Label використовується для різного роду статичних надписів в інтерфейсі програми:

```
    lbl1 = Label(root, text="Розрахунки день", bg="gray", fg="black",
    font=("Arial 20 bold"), width=20, height=10)
    lbl1.place(x=20, y=100)
```

Віджет `Button` використовується для здійснення розрахунків в програмі, побудові графіків, побудові нових вікон та побудові таблиць, вводу та виводу інформації з бази даних.

```
btn1 = Button(root, text = "Обчислити", command = sosna)
btn1.place(x=20, y=200)
```

Крім того, у даного віджета є параметр `command`. При натисненні на кнопку відбувається подія (викликається функція із назвою `sosna`). Цю кнопку потрібно прив'язати до обробника подій, який потрібно перед тим створити. В даному прикладі використано функцію з іменем `sosna()`.

```
def sosna():
    new_window2 = Toplevel(root)
    new_window2.title("Сосна")
    new_window2.resizable(0, 0)
    new_window2.mainloop()
```

При натисненні на цю кнопку буде побудоване нове вікно поверх існуючого.

Віджет `Treeview` використовується, коли потрібно в програмі представити дані табличного типу. Віджет `Combobox` використовується, коли потрібно в програмі вибрати дані з певного переліку (списку).

4.2.4. Проектування бази даних обліку лісових насаджень

При виборі в стартовому вікні програми пункту меню Файл/ Новий появиться нове вікно: База даних для обліку лісових масивів (рис. 4.3). Воно має чотири поля: номер дерева, висота, клас бонітету, діаметр. Для вводу облікових даних лісових масивів потрібно натиснути кнопку Добавити новий запис. При цьому появиться вікно, в яке можна вводити дані по кожному дереву при обліку масивів (рис. 4.4). В нього вводиться висота дерева, клас бонітету, діаметр дерева. Після вводу облікових даних для даного дерева потрібно натиснути кнопку Добавити.

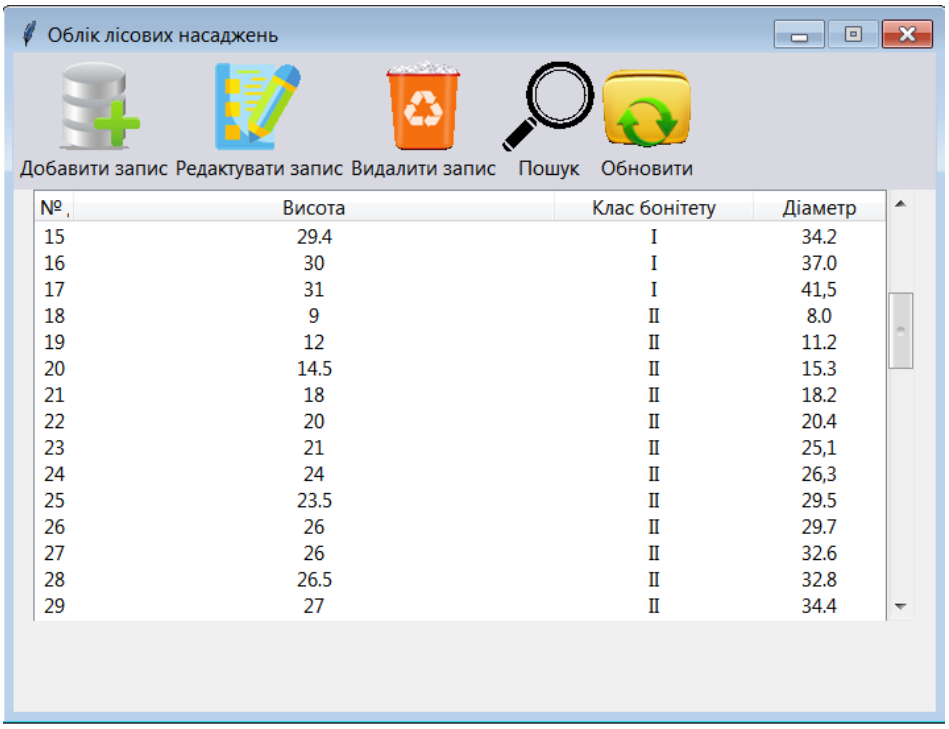


Рис. 4.3. База даних обліку лісових насаджень.

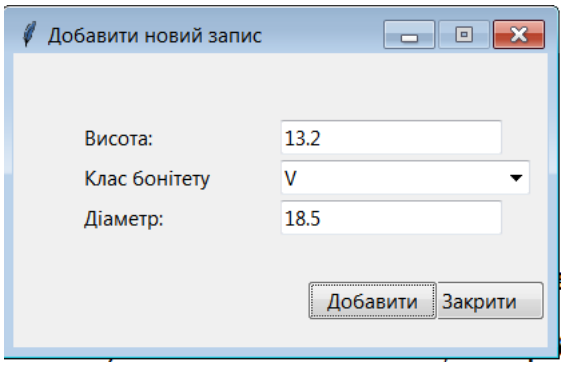


Рис. 4.4. Діалогове вікно для вводу інформації в базу даних.

Натиснувши кнопку Редактувати запис, появиться діалогове вікно, в якому можна змінювати дані для введених записів (рис. 4.5). Для цього потрібно попередньо виділити запис, який потрібно відредактувати. Після проведених змін запису натиснути кнопку Редактувати.

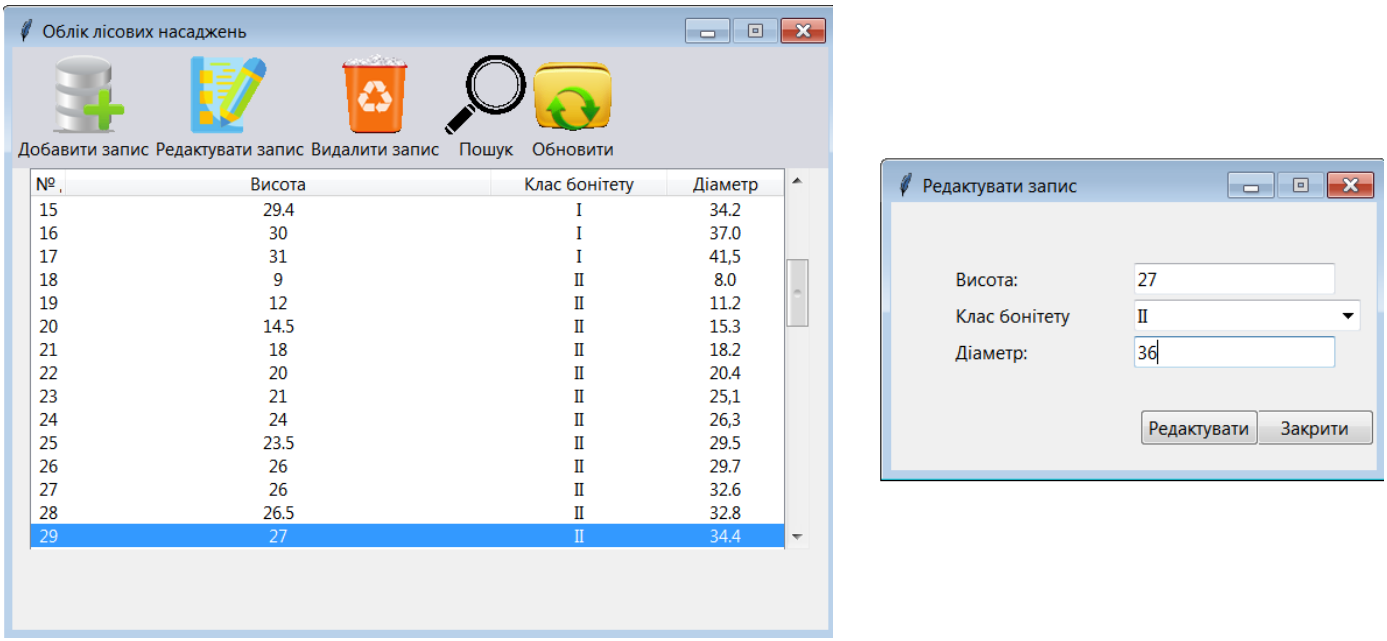


Рис. 4.5. Діалогове вікно для редактування записів.

При натисненні кнопки Видалити запис можна видалити непотрібний запис, який попередньо треба виділити, із бази даних. Натиснувши кнопку Пошук, можна знайти потрібний запис по певних параметрах полів (рис. 4.6).

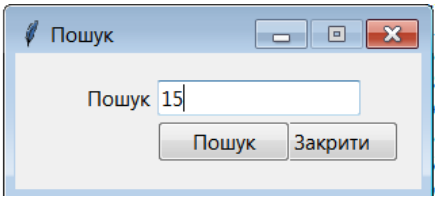


Рис. 4.6. Діалогове вікно Пошук.

Ввівши всі потрібні облікові записи дерев, потрібно натиснути кнопку Обновити. При цьому всі записи будуть збережені у базі даних SQLite у файлі lis.db у цій же папці, де знаходиться основний файл скрипта Oblik.py.

4.3. Порядок роботи з програмою обліку лісових насаджень

Для моделювання ходу росту лісових масивів в меню потрібно вибрати пункт Моделювання/ Порода/ Сосна. Появиться нове вікно із назвою Сосна, інтерфейс якого представлений на рис. 4.7. Дане вікно містить чотири вкладки: Висота, Діаметр, Площа, Об’єм, на яких отримаємо відповідні результати моделювання та облікові записи по деревах досліджуваної ділянки.

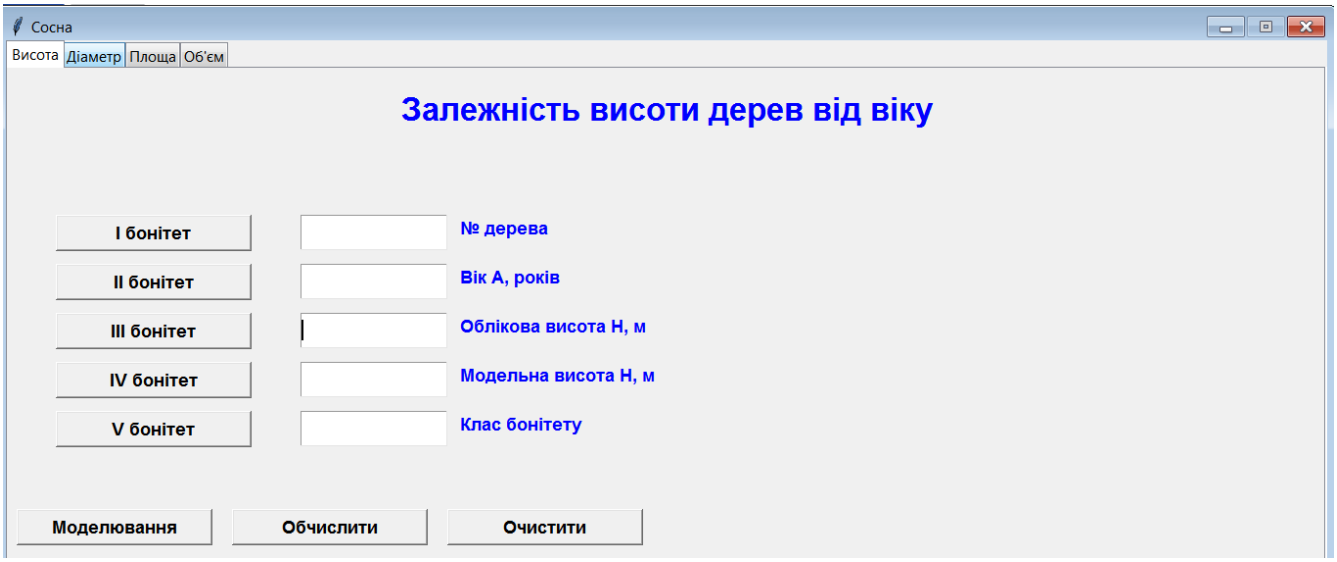


Рис. 4.7. Інтерфейс програми для моделювання ходу росту дерев у висоту.

Натиснувши кнопку Моделювання, отримаємо графік залежності висоти дерев сосни в залежності від віку для п’ятьох класів бонітету (з I по V):

Рис. 4.8. Залежність висоти сосни від віку для п’ятьох класів бонітету.

Для обробки вхідних даних та вводу та виводу інформації з бази даних потрібно в текстове поле №дерева ввести номер модельного дерева та натиснути кнопку Обчислити. При цьому текстові поля заповняться відповідними значеннями (рис. 4.9). При натисненні кнопки I бонітет появиться графічна залежність для облікових та модельних даних для даного класу бонітету (рис. 4.11).

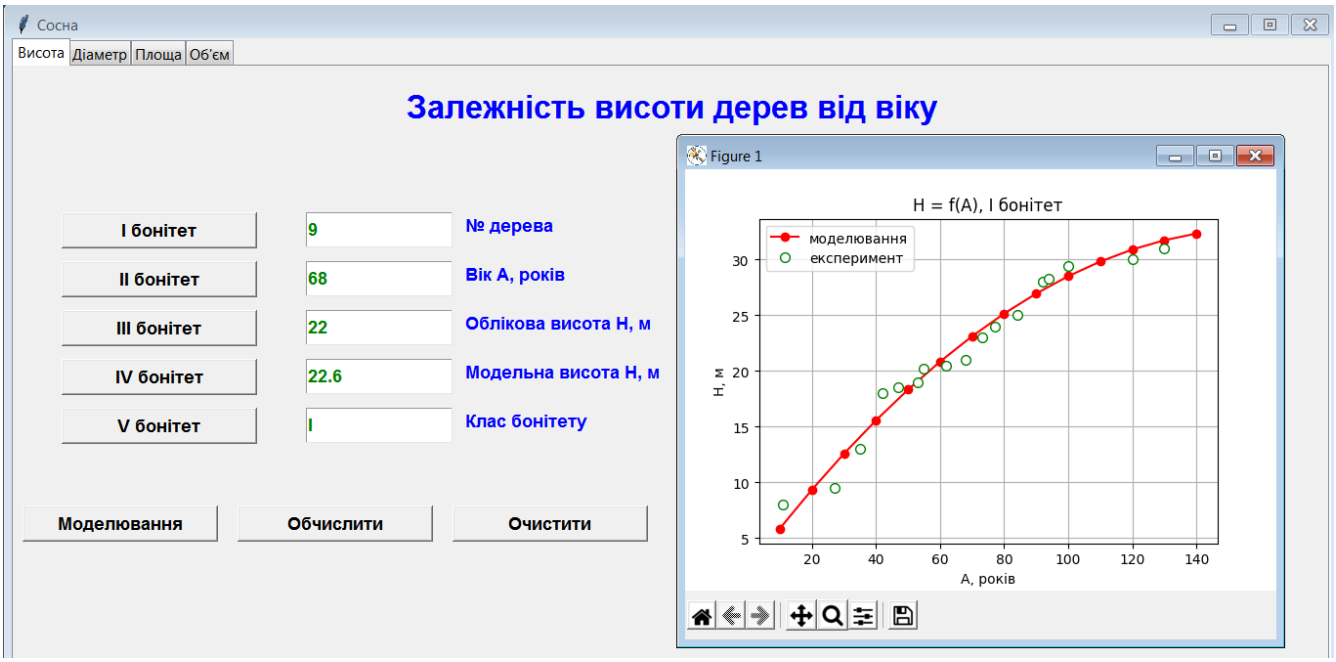


Рис. 4.9. Робоче вікно програми з побудованим графіком висот.

Для перегляду результатів моделювання та графіків зміни висоти, діаметру, суми площ перерізів та запасу у межах від віку необхідно вибрати потрібну вкладку:

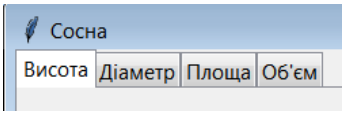


Рис. 4.10. Вкладки для різних параметрів моделювання ходу росту сосни (вибрана вкладка Висота).

Для збереження графіків слід у графічному вікні із графіками натиснути крайню праву кнопку Save the figure. Появиться діалогове вікно, в якому треба вибрати диск та папку, куди потрібно зберегти графік, і ввести ім’я файлу (наприклад: diametr). Також необхідно вибрати тип файлу png – portable network graphics.

4.4. Результати роботи програми

В аналізі результатів слід відобразити дані про висоту H, діаметр D, суми площ перерізів S та запаси деревини V на досліджуваній ділянці лісу. За отриманими графіками зміни фактичних (експериментальних) та модельних показників в залежності від віку дерев можна порівняти зміни цих облікових ознак. Для найбільш характерних змін можна вказати їхні вікові межі.

Для моделювання часового ряду ходу росту діаметрів та висот окремих дерев сосни розраховуються потрібні параметри з використанням функції росту Мітчерліха. При розрахунку параметрів цієї функції як аргументи використовують вік і діаметр дерева на висоті 1,3 м.

Вибирають вкладку Висота. У даному вікні розміщені п’ять кнопок, кожна з яких відповідає за свій клас бонітету облікових дерев сосни. Натиснувши відповідні кнопки, отримаємо графічні залежності модельних та облікових даних для даного класу бонітету.

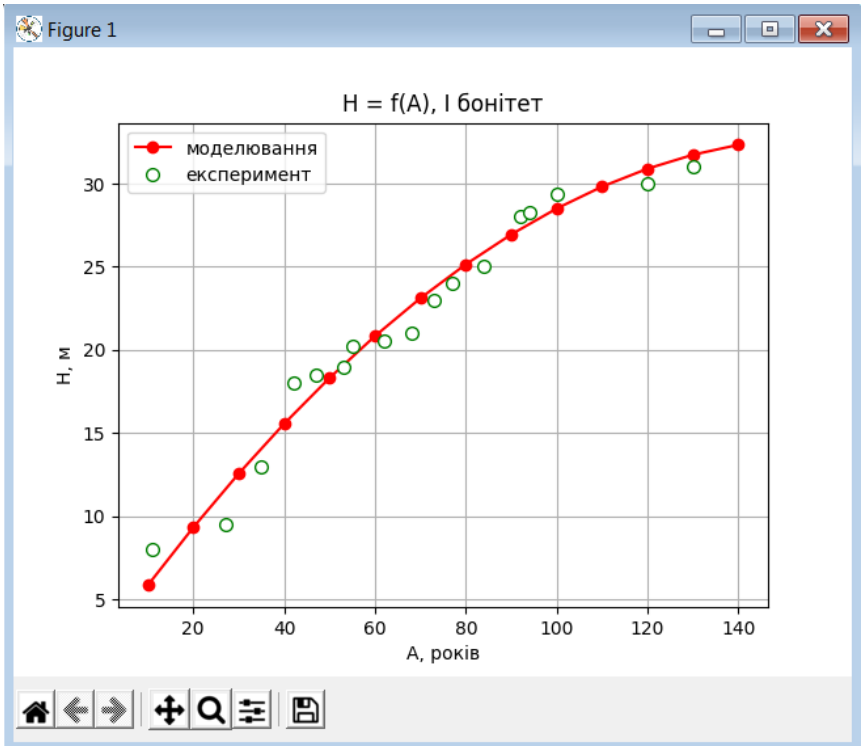


Рис. 4.11. Експериментальні та модельні значення ходу росту по висоті дерев сосни для I класу бонітету.

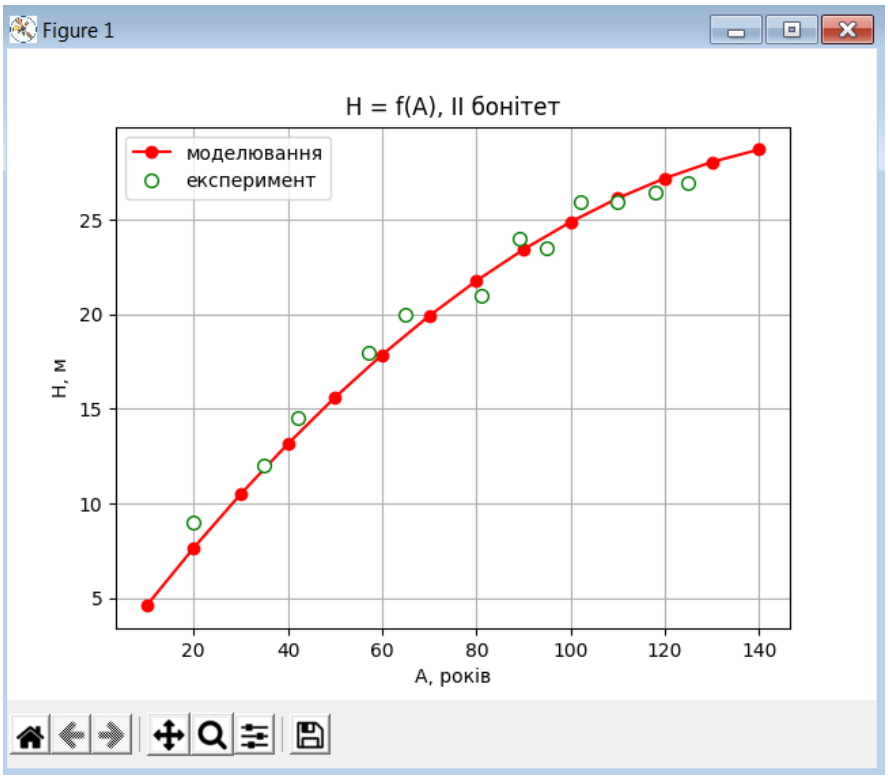


Рис. 4.12. Експериментальні та модельні значення ходу росту по висоті дерев сосни для II класу бонітету.

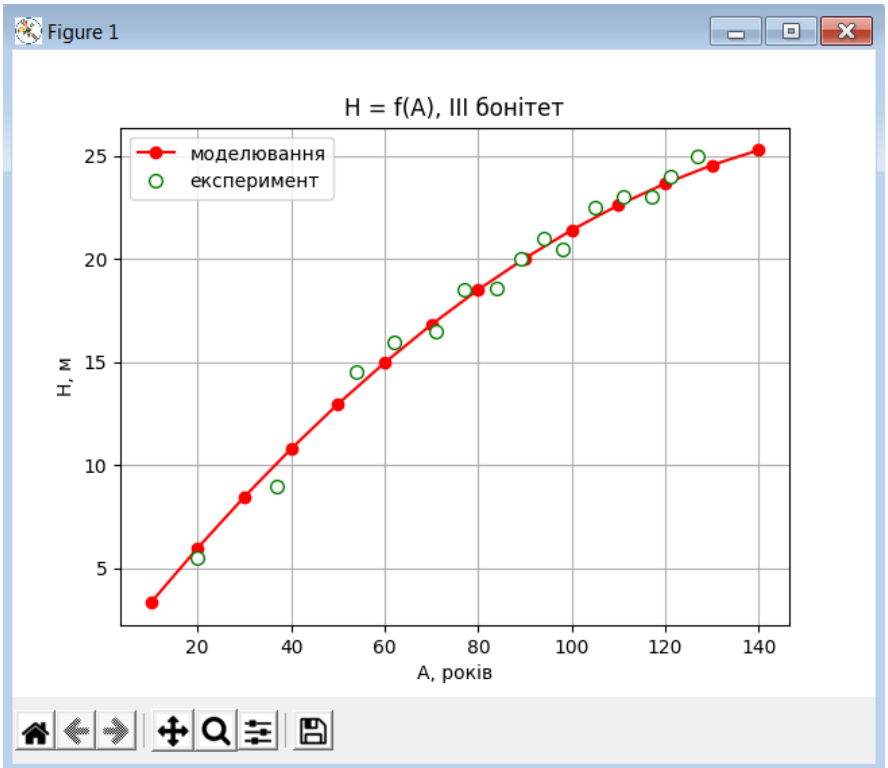


Рис. 4.13. Експериментальні та модельні значення ходу росту по висоті дерев сосни для III класу бонітету.

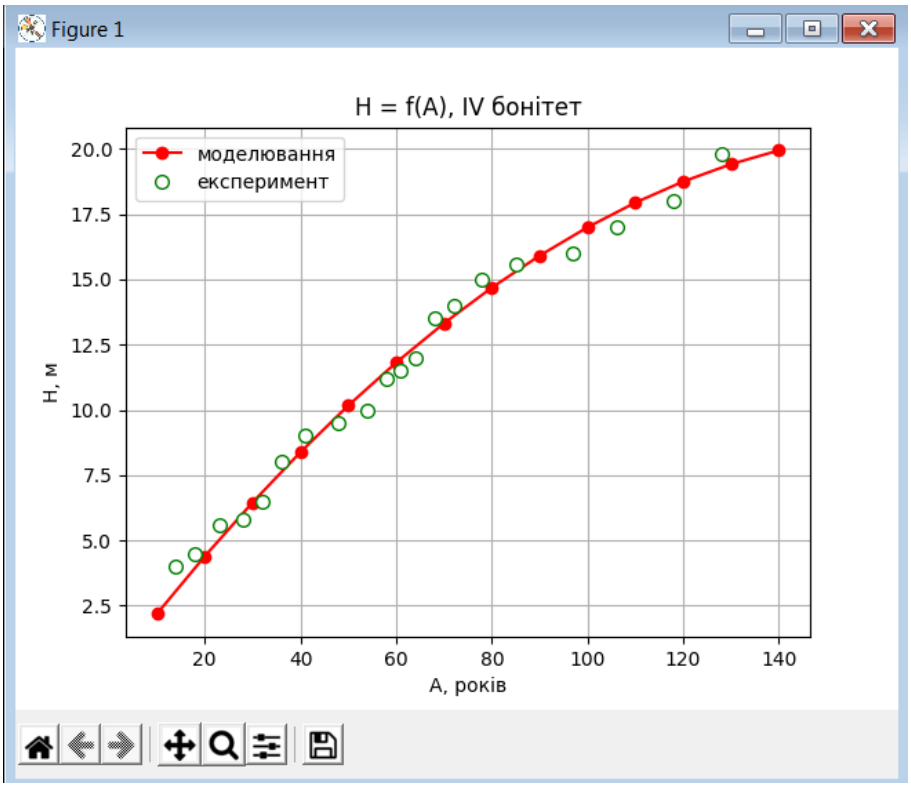


Рис. 4.14. Експериментальні та модельні значення ходу росту по висоті дерев сосни для IV класу бонітету.

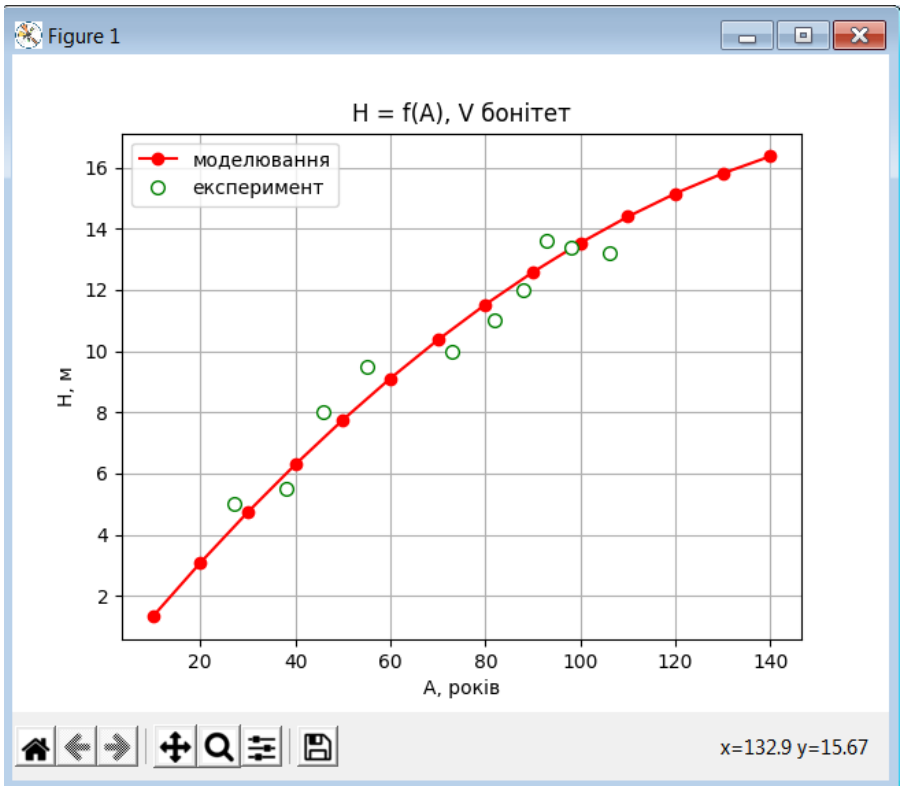


Рис. 4.15. Експериментальні та модельні значення ходу росту по висоті дерев сосни для V класу бонітету.

Як видно з рис. 4.11-4.15, крива ходу росту по висоті з використанням функції Мітчерліха з високою точністю співпадає з експериментальними значеннями обліку висоти Н.

На вкладці Діаметр представлено результати моделювання по діаметру дерев сосни в залежності від їхнього віку по всій вибірковій їхній сукупності.

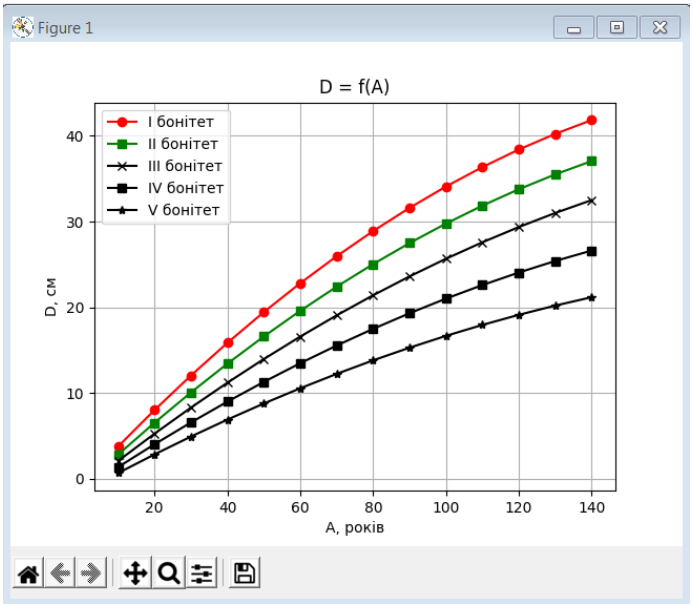


Рис. 4.16. Залежність діаметра стовбурів сосни від віку для п’ятьох класів бонітету.

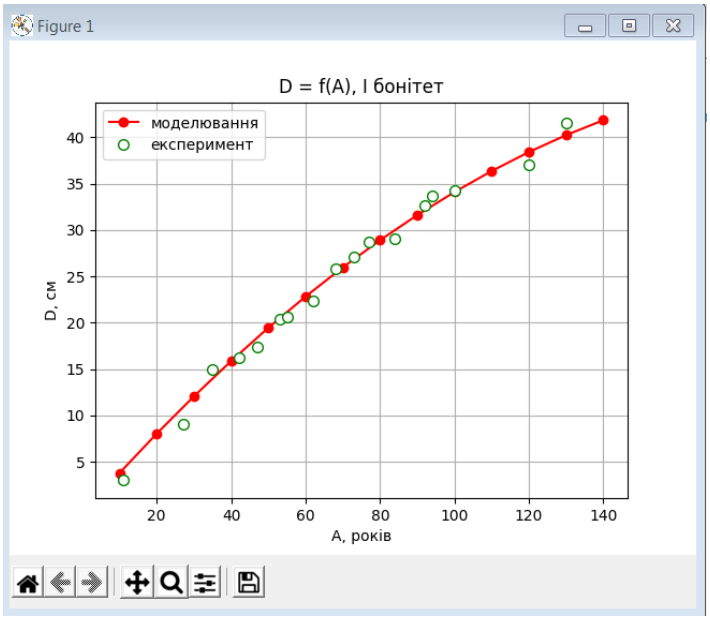


Рис. 4.17. Експериментальні та модельні значення ходу росту по діаметру стовбурів сосни I класу бонітету.

На вкладці Площа представлено результати моделювання по середній площі поперечного перерізу дерев сосни в залежності від їхнього віку по всій їхній вибірковій сукупності.

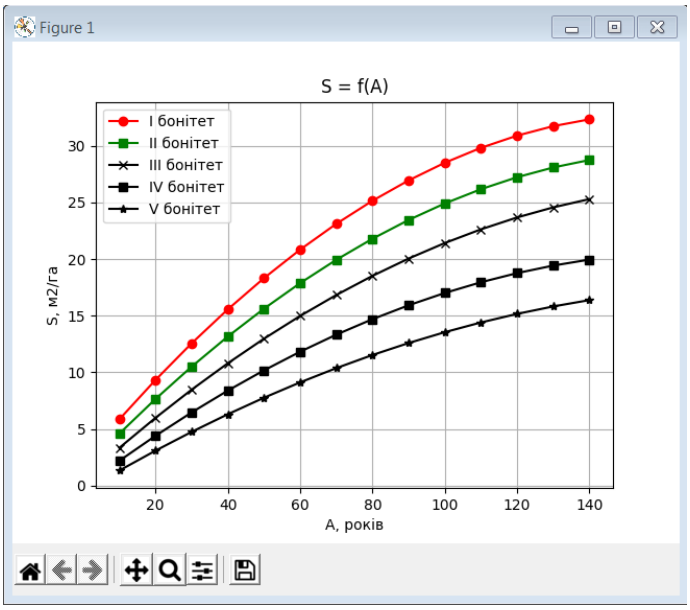


Рис. 4.18. Залежність середньої площі поперечного перерізу стовбурів сосни від віку для п’ятьох класів бонітету.

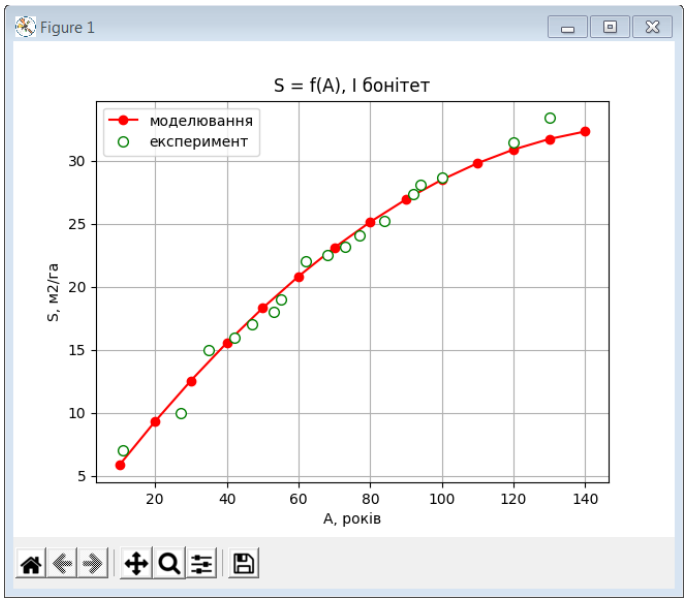


Рис. 4.19. Експериментальні та модельні значення ходу росту середньої площі поперечного перерізу стовбурів сосни I класу бонітету.

На вкладці Об’єм представлено результати моделювання по середньому об’єму запасу дерев сосни в залежності від їхього віку по всій їхній вибірковій сукупності.

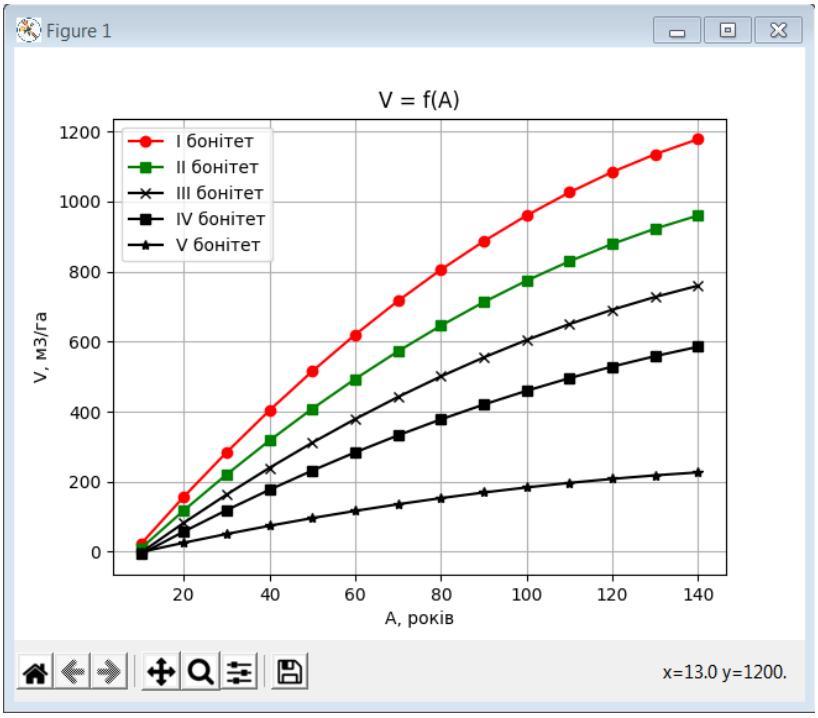


Рис. 4.20. Залежність середнього об’єму запасу стовбурів сосни від віку для п’ятьох класів бонітету.

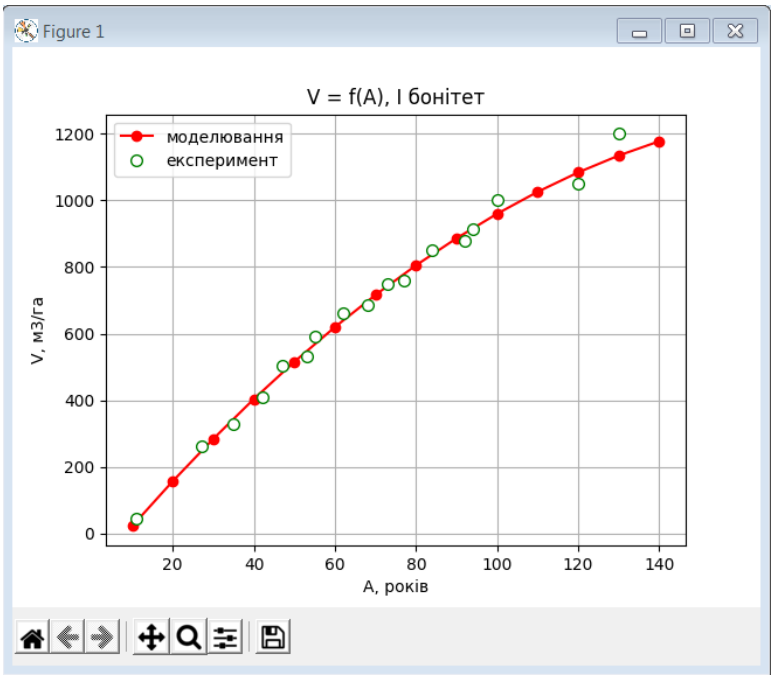


Рис. 4.21. Експериментальні та модельні значення ходу росту середнього об’єму запасу стовбурів сосни I класу бонітету.

Крім сосни, в даній програмі можна проводити моделювання ще для чотирьох порід дерев:

- смереки;
- дуба:
- граба;
- берези.

Для цього потрібно в меню вибрати пункт Моделювання/ Порода/, далі вибрати з наявного переліку потрібну породу дерева.

ВИСНОВКИ ДО РОЗДІЛУ 4

У четвертому розділі розроблено програмне забезпечення для моделювання та оцінки динаміки облікових показників лісових масивів. Отримано модель ходу росту лісових масивів в залежності від віку. На її основі отримано дані для сум площ перерізів та запасів даної породи. Запропоновано алгоритм моделювання ходу росту лісових масивів різного класу бонітету. На основі математичної моделі отримано модельні дані для висот, діаметрів, сум площ перерізів, запасів деревини для п’ятьох лісових порід: сосни, смереки, дуба, граба та берези.

РОЗДІЛ 5. РОЗРОБЛЕННЯ СТАРТАП ПРОЕКТУ

5.1. Інформаційна карта програмного продукту

В інформаційній карті проекту відображають його вартість та характеристики, що передбачає видатки на заробітню платню, оренду приміщення та обладнання, витрати на повний цикл розробки програмного продукту. Дана карта представлена в табл. 5.1.

Табл. 5.1. Інформаційна карта програмного продукту

Назва номінації	Python програма
Назва проекту	Розроблення математичного та програмного забезпечення для обліку лісових насаджень
Назва ВНЗ, факультету, спеціальності	НЛТУ, кафедра інформаційних технологій, 122 «Комп'ютерні науки»
Прізвище, ім'я, по-батькові	Василів Романна Богданівна
Цілі і задачі проекту	Ціль проекту – розробка та реалізація інформаційної системи для обліку лісових насаджень. Задачі проекту: • проаналізувати системи, які існують для обліку лісових насаджень; • розробити інформаційну систему для обліку лісових насаджень; • реалізувати програмне забезпечення для обліку лісових насаджень; • в рамках програми провести дослідження параметрів росту

	поодинокого дерева та лісових масивів на досліджуваній території лісу.
Короткий зміст проекту	Розроблено та реалізовано програмне забезпечення для обліку лісових насаджень, яке враховуватиме стан лісової рослинності на досліджуваній території та динаміку розвитку лісових деревостанів на території, яка досліджується. Спроектовано програмне забезпечення для обліку лісових масивів, в рамках даної програми представлено прогноз динаміки росту лісових масивів. Результати роботи даної системи прогнозування росту лісових деревостанів можуть бути використані працівниками лісового господарства.
Терміни виконання проекту	8 місяців
Бюджет проекту	60 000 грн.

5.2. Стратегія проекту

При виконані проекту потрібно буде зібрати велику кількість даних із різних джерел, на основі цих даних оцінити правильність обраного курсу, як приймати правильні рішення та впевнено йти до мети.

Існують багато факторів, які визначають популярність продукту на ринку. На землі живуть різні люди – з різним світоглядом, характерами, способом життя. Ці люди так само створюють продукти – орієнтуючись на свої уподобання і наголошуючи на власних інтересах. Тому у всіх фахівців бачення ідеального продукту, а також способи його створення завжди відрізнятимуться.

Одні роблять ставку на дизайн, інші розвивають технічні особливості продукту та широко використовують хмарні технології. До речі, ці технології дозволяють програмістам користуватися ресурсами потужних комп'ютерів, об'єднаних в ще більш потужну машину. Хмарні технології вирішують багато технічних проблем. Крім того, з їх допомогою користувач може ефективно працювати з програмами, що вимагають високої продуктивності, та величезними базами даних. Ці технології з недавніх пір активно використовують компанії та держструктури. Інша команда віддасть перевагу технології «клієнт-сервер». Вона дозволяє виконувати більшість функцій програмування на одній машині.

Тому що навіть великі корпорації, що виділяють великі кошти на створення сучасних технологій, не можуть гарантувати успіх своїм продуктам. Великі корпорації можуть собі дозволити просто дочекатися появи потрібного продукту на ринку, не витрачаючи при цьому своїй ресурсів на його розробку.

Якщо технологія є дуже важливою, корпорація самостійно займеться її розвитком. Але при цьому вона має тримати руку на пульсі та стежити за тим, що відбувається на ринку у цій сфері. Звичайно, якщо інша компанія представляє найкращий продукт, то тут нічого не вдієш. Але якщо це стартап, то його можна придбати. Крім перерахованих причин, через які скуповуються компанії, є й інші. Іноді угоду проводять для того, щоб стартапи своєю активністю не заважали розвиватися великим підприємствам. Або щоб вони не дісталися конкурентам.

Коли компанія вирішує освоїти нову сферу діяльності, вона стикається з багатьма питаннями. Одне із найголовніших — нова команда. Сформувати її – непросте завдання і займає багато часу. Створити команду непросто не лише тому, що потрібен певний рівень знань, кваліфікації, але ще й тому, що ніколи не знаєш, чи всі ці люди зможуть працювати разом. Порозуміння у колективі – це важливий аспект успіху всього підприємства.

Таким чином, не лише непросто заробити на створенні нового продукту, а й повернути витрачені гроші. У традиційному бізнесі ринок зростає повільно. Його учасники можуть спостерігати за всіма змінами та поступово на основі висновків робити прогнози на майбутнє. В інтернет-бізнесі все відбувається стрімко та хаотично. Через це передбачити майбутні події досить складно. Ніхто не може сказати з упевненістю, чи стане новий продукт феноменом і сенсацією або вже через місяць про нього все забудуть, тому що на ринку з'являться більш цікаві продукти.

Насправді більшість інвесторів вкладають гроші у надію. Наприклад, досі немає такої бізнес-моделі, яку можна було б підігнати під новітні технології та цим виправдати інвестиції. В даному випадку інвестори просто не знають, як розвиватиметься новий продукт і чи буде він популярним серед споживачів. Найчастіше залишається тільки сподівається і чекати, що товар завоює ринок і на цьому можна буде заробити. Іноді доводиться просто довіряти інтуїції власника підприємства та вірити у його справу. Ця віра в інвесторів посилюється, якщо в історіє підприємства була серія успішних ідей. Швидкість росту компанії – це один із орієнтирів, за яким можна передбачити, як розвиватиметься бізнес.

Динаміка виручки – той показник, який дозволяє передбачити успіх. Інший важливий показник – кількість користувачів.

Будь-який проект має певні ризики, які пов’язані з його діяльністю. Підприємницький ризик очевидний через невизначеність. Це економічні, соціальні та політичні умови, у яких фірма здійснює свою діяльність. Невизначеність залежить від багатьох змінних, контрагентів та осіб, дії яких не завжди можна передбачити з необхідною точністю. Підприємницький ризик – ризик, який пов’язаний із певним бізнесом у його ринковій ніші.

Ефективний спосіб зниження ризиків проекту – вибрати одну з моделей та адаптувати її під стартап проект.

Основні задачі проекту включають:

- розробка критеріїв для дослідження моделей;
- аналіз існуючих моделей і вибір найбільш оптимальної моделі для проекту;
- опис бізнес моделі стартапу;
- розробка алгоритму використання моделі в виді рекомендацій;
- економічне обгрунтування проекту.

Стартап – це назва для фірми або проекту, які існують короткий проміжок часу, зазвичай він становить від кількох тижнів до кілька місяців. Після цього періоду проект перестає бути стартапом, оскільки або отримує визнання, або закривається як неконкурентоспроможний.

Визначення стартапу вперше виникло в інформаційній сфері. Кожна компанія намагалася знайти для клієнта індивідуальний підхід та запропонувати щось нове.

При відборі моделей для аналізу були використані два критерії: мінімізація витрат та вплив моделі на ризик запуску стартап проекту. Рівень витрат складається з:

- часових ресурсів;
- грошових ресурсів.

Організаційна структура проекту складається з керівника проекту та команди проекту. У проекті функціональний розподіл праці виглядає наступним чином: керівник проекту відповідальний за спілкування з постачальником техніки та орендодавцем, решта завдань рівномірно лягають на плечі всіх учасників команди, у тому числі і керівника проекту. Кожен із учасників команди здатний брати участь у виконанні будь-якого із завдань проекту.

Проектні ризики діляться на три категорії:

- технічні;
- зовнішні;
- організаційні.

Технічні ризики представлені в табл. 5.3.

Табл. 5.3. Технічні ризики

Види ризиків	Від'ємний вплив	Способи захисту	Спосіб запобігання
ненадійність техніки	збільшення затрат на ремонт і покупку обладнання	резервування грошових засобів	пошук альтернативних апаратів

Організаційні ризики представлені в табл. 5.4.

Табл. 5.4. Організаційні ризики

Види ризиків	Від'ємний вплив	Способи захисту	Спосіб запобігання
Зменшення мотивації учасників проекту. Відсутність резервних можливостей	Зниження об'ємів продаж.	Мотивація і стимулювання. Пошук додаткових можливостей	Постійна мотивація. Мати запасний варіант.

5.3. Розробка програми стартап проекту

Табл. 5.5. Основні переваги та концепції інформаційної системи

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами
1	моделювання обліку лісових насаджень	створення власних функціональних одиниць	гнучкість та свобода для кінцевого споживача

2	визначення параметрів обліку лісових насаджень	можливість підключення різноманітних вимірювальних пристроїв та датчиків	використання даних з безпілотних літальних апаратів
3	зручний інтерфейс	інтерфейс, який містить потрібні дані для обліку лісових насаджень	можливість прогнозування росту окремого дерева та лісових масивів в динаміці

Табл. 5.6. Опис трьох рівнів інформаційної системи обліку лісових насаджень

рівні товару	**сутність та її складові**
1. Програмний продукт за задумом	моделювання для обліку лісових насаджень
2. Програмний продукт, який має бути реально виконаний	програмний продукт для обліку лісових насаджень
3. Підкріплення	служба для підтримки системи прийняття рішень в лісогосподарській діяльності за допомогою даної інформаційної системи

Табл. 5.7. Ціна 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	250 $+	250/125 $

ВИСНОВКИ ДО РОЗДІЛУ 5

У п’ятому розділі спроектовано стартап проекту, дана методика дозволяє досліджувати лісові ділянки, вести їх облік та спрогнозувати їх ріст. Розроблено та реалізовано стартап-проект для обліку лісових насаджень. Він може бути реалізований на практиці. Його перевагою є те, що альтернативних програмних продуктів на ринку є мало. При використанні реклами та оголошень для просування даного стартапу програмного продукту можна досягти успіху та отримати пристойний дохід.

ВИСНОВКИ

В роботі розглянуто особливості обліку лісових насаджень, формування бази даних та розробку програмного забезпечення для обліку лісових насаджень. Основною метою моделювання лісогосподарських процесів є створення такої моделі, яка найповніше описує процеси зміни лісових систем загалом чи окремих її структурних частин. Кожен конкретний випадок ставить відповідне завдання: створення моделей росту та продуктивності деревостанів, моделей їх будови за обліковими показниками, моделей зміни окремих облікових показників з часом, моделей взаємозв'язків облікових показників між собою.

В першому розділі проаналізовано предметну область, засоби та технології проектування системи обліку лісових насаджень. З допомогою створеного програмного забезпечення можна моделювати динаміку росту окремого дерева та лісових масивів в цілому, проводити оцінку параметрів цих дерев на досліджуваній лісовій території.

У другому розділі приведено відомості про існуючі системи обліку лісових насаджень та визначення їх параметрів. Описано технології розробки даних продуктів, а також описано принципи їх функціонування. Розглянуто бібліотеки та модулі мови Python, які відповідають за створення інтерфейсу програмного продукту, проведення обчислень індексів математичних залежностей, а також візуалізації результатів.

У третьому розділі розроблено математичну модель для моделювання росту лісових насаджень, яка є базою для проектування програмного забезпечення, яке складається з програмних модулів для забезпечення інтерактивної роботи, прийому вхідних даних, оцінки параметрів лісових деревостанів, моделювання ходу росту лісових масивів. Моделювання облікових показників за результатами вимірювань за методикою повного аналізу ходу росту окремого дерева підтверджує правильність вибору функції Мітчерліха для передбачення співвідношень висот та діаметрів залежно від віку.

У четвертому розділі розроблено програмне забезпечення для моделювання та оцінки динаміки облікових показників лісових масивів. Отримано моделі ходу росту

лісових масивів в залежності від віку. На їх основі отримано дані для сум площ перерізів та запасів для конкретної лісової породи. Запропоновано алгоритм моделювання ходу росту лісових масивів різного класу бонітету. На основі математичної моделі отримано модельні дані для висот, діаметрів, сум площ перерізів, запасів деревини для п’ятьох лісових порід: сосни, смереки, дуба, граба та берези.

У п’ятому розділі спроектовано стартап проекту, дана методика дозволяє досліджувати лісові ділянки, вести їх облік та спрогнозувати їх ріст. Розроблено та реалізовано стартап-проект для обліку лісових насаджень. Він може бути реалізований на практиці. Його перевагою є те, що альтернативних програмних продуктів на ринку є мало. При використанні реклами та оголошень для просування даного стартапу програмного продукту можна досягти успіху та отримати пристойний дохід.

Проведені дослідження програмного забезпечення підтверджують адекватність даної моделі по обліку та моделювнню параметрів росту окремого дерева та лісових насаджень. Дане програмне забезпечення може допомогти працівникам лісового господарства у їхній роботі.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Шоу Зед. Легкий способ выучить Python. Москва : Эксмо, 2017. – 352 с.

2. Чан Уэсли Дж. Python: создание приложений. 3-е издание. – М.: Вильямс, 2015. – 816 с.

3. Хеллман Даг. Стандартная библиотека Python 3. Справочник с примерами. – СПб.: Диалектика, 2019. – 1376 с.

4. Хайбрахманов С.А. Основы научных расчётов на языке программирования Python. – Челябинский государственный университет, 2019. – 96 с.

5. Сузи Р.А. Язык программирования Python. – М.: "Интуит", 2016. – 351 с.

6. Седер Наоми. Python. Экспресс-курс. 3-е изд. – СПб.: Питер, 2019. – 479 с.

7. Северенс Ч. Введение в программирование на Python. М.: Интуит, 2016. – 232 с.

8. Саммерфилд Марк. Python на практике. – ДМК-Пресс, 2014. – 338 с.

9. Рамальо Лучано. Python. К вершинам мастерства. – ДМК Пресс, 2016. – 768 с.

10. Прохоренок Н.А., Дронов В.А. Python 3. Самое необходимое. – Спб.; БХВ-Петербург, 2019. – 608 с.

11. Антанайтис В.В., Загреев В.В. Прирост леса. 2-е изд., перераб. – М.: Лесная промышленность, 1981. – 200 с.

12. Воропанов П.В. Определение текущего древесного прироста. М.: Л.: Гослесбумиздат, 1961. – 135 с.

13. Кузьмичев В.В. Закономерности роста древостоев. Новосибирск: Наука, 1977. – 160 с.

14. Макаренко А.А. Строение древостоев. Алма-Ата: Кайнар, 1982. – 68 с.

15. Рогозин М.В. Структура древостоев: конкуренция или партнерство? – Пермь: Пермский государственный национальный исследовательский университет, 2019. – 223 с.

16. Сидаренко П.В., Веселов О.О. Таксация леса. Новочеркасск: Новочеркасская государственная мелиоративная академия, 2013. – 50 с.

ДОДАТКИ

ДОДАТОК А.

Oblik.py

```
from tkinter import *
from tkinter import ttk
from tkinter.filedialog import *
import matplotlib.pyplot as plt
import numpy as np

def open_file():
    open_file = askopenfilename()
def save_file():
    save_file = asksaveasfilename()
def a1():
    root.destroy()
def sosnah():
    x=np.arange(10, 150, 10)
    y1=-0.0012*x**2+0.3838*x+2.1222
    y2=-0.001*x**2+0.3359*x+1.3182
    y3=-0.0008*x**2+0.289*x+0.5123
    y4=-0.0007*x**2+0.2417*x-0.168
    y5=-0.0005*x**2+0.1908*x-0.5406
    plt.plot(x, y1, "-ro", label="I бонітет")
    plt.plot(x, y2, "-gs", label="II бонітет")
    plt.plot(x, y3, "-kx", label="III бонітет")
    plt.plot(x, y4, "-ks", label="IV бонітет")
    plt.plot(x, y5, "-k*", label="V бонітет")
    plt.title("H = f(A)")
    plt.xlabel("A, років")
    plt.ylabel("H, м")
    plt.legend()
    plt.grid()
    plt.show()
```

```
def sosnah1():
    x=np.arange(10, 150, 10)
    y1=-0.0012*x**2+0.3838*x+2.1222
    x2=[11, 27, 35, 42, 47, 53, 55, 62, 68, 73, 77, 84,  92, 94, 100, 120, 130]
    y2=[8, 9.5, 13, 18, 18.5, 19, 20.2, 20.5, 21, 23, 24, 25, 28, 28.3, 29.4, 30, 31]
    plt.plot(x, y1, "-ro", label="моделювання")
    plt.plot(x2, y2, "go", markersize=7, markerfacecolor='#ffffff',
    label="експеримент")
    plt.title("H = f(A), I бонітет")
    plt.xlabel("A, років")
    plt.ylabel("H, м")
    plt.legend()
    plt.grid()
    plt.show()
def sosnah2():
    x=np.arange(10, 150, 10)
    y1=-0.001*x**2+0.3359*x+1.3182
    x2=[20, 35, 42, 57, 65, 81, 89, 95, 102, 110, 118, 125]
    y2=[9, 12, 14.5, 18, 20, 21, 24, 23.5, 26, 26, 26.5, 27]
    plt.plot(x, y1, "-ro", label="моделювання")
    plt.plot(x2, y2, "go", markersize=7, markerfacecolor='#ffffff',
    label="експеримент")
    plt.title("H = f(A), II бонітет")
    plt.xlabel("A, років")
    plt.ylabel("H, м")
    plt.legend()
    plt.grid()
    plt.show()
def sosnah3():
    x=np.arange(10, 150, 10)
    y1=-0.0008*x**2+0.289*x+0.5123
    x2=[20, 37, 54, 62, 71, 77, 84, 89, 94, 98, 105, 111, 117, 121, 127]
    y2=[5.5, 9, 14.5, 16, 16.5, 18.5, 18.6, 20, 21, 20.5, 22.5, 23, 23, 24, 25]
    plt.plot(x, y1, "-ro", label="моделювання")
    plt.plot(x2,      y2,      "go",      markersize=7,      markerfacecolor='#ffffff',
    label="експеримент")
    plt.title("H = f(A), III бонітет")
    plt.xlabel("A, років")
    plt.ylabel("H, м")
    plt.legend()
    plt.grid()
    plt.show()
```

```
def sosnah4():
    x=np.arange(10, 150, 10)
    y1=-0.0007*x**2+0.2417*x-0.168
    x2=[14, 18, 23, 28, 32, 36, 41, 48, 54, 58, 61, 64, 68, 72, 78, 85, 97, 106, 118,
    128]
    y2=[4, 4.5, 5.6, 5.8, 6.5, 8, 9, 9.5, 10, 11.2, 11.5, 12, 13.5, 14, 15, 15.6, 16,
    17, 18, 19.8]
    plt.plot(x, y1, "-ro", label="моделювання")
    plt.plot(x2,      y2,      "go",      markersize=7,      markerfacecolor='#ffffff',
    label="експеримент")
    plt.title("H = f(A), IV бонітет")
    plt.xlabel("A, років")
    plt.ylabel("H, м")
    plt.legend()
    plt.grid()
    plt.show()
def sosnah5():
    x=np.arange(10, 150, 10)
    y1=-0.0005*x**2+0.1908*x-0.5406
    x2=[27, 38, 46, 55, 73, 82, 88, 93, 98, 106]
    y2=[5, 5.5, 8, 9.5, 10, 11, 12, 13.6, 13.4, 13.2]
    plt.plot(x, y1, "-ro", label="моделювання")
    plt.plot(x2,      y2,      "go",      markersize=7,      markerfacecolor='#ffffff',
    label="експеримент")
    plt.title("H = f(A), V бонітет")
    plt.xlabel("A, років")
    plt.ylabel("H, м")
    plt.legend()
    plt.grid()
    plt.show()
def sosnad():
    x=np.arange(10, 150, 10)
    y1=-0.0011*x**2+0.4577*x-0.6848
    y2=-0.0009*x**2+0.3982*x-1.0546
    y3=-0.0007*x**2+0.3389*x-1.2234
    y4=-0.0006*x**2+0.2842*x-1.402
    y5=-0.0005*x**2+0.2327*x-1.593
    plt.plot(x, y1, "-ro", label="I бонітет")
    plt.plot(x, y2, "-gs", label="II бонітет")
    plt.plot(x, y3, "-kx", label="III бонітет")
    plt.plot(x, y4, "-ks", label="IV бонітет")
    plt.plot(x, y5, "-k*", label="V бонітет")
```

```
    plt.title("D = f(A)")
    plt.xlabel("A, років")
    plt.ylabel("D, см")
    plt.legend()
    plt.grid()
    plt.show()
def sosnad1():
    x=np.arange(10, 150, 10)
    y1=-0.0011*x**2+0.4577*x-0.6848
    x2=[11, 27, 35, 42, 47, 53, 55, 62, 68, 73, 77, 84, 92, 94, 100, 120, 130]
    y2=[3, 9, 15, 16.2, 17.4, 20.4, 20.6, 22.3, 25.8, 27.1, 28.7, 29.1, 32.6, 33.7,
    34.2, 37, 41.5]
    plt.plot(x, y1, "-ro", label="моделювання")
    plt.plot(x2, y2, "go", markersize=7, markerfacecolor='#ffffff',
    label="експеримент")
    plt.title("D = f(A), I бонітет")
    plt.xlabel("A, років")
    plt.ylabel("D, см")
    plt.legend()
    plt.grid()
    plt.show()
def sosnad2():
    x=np.arange(10, 150, 10)
    y1=-0.0009*x**2+0.3982*x-1.0546
    x2=[20, 35, 42, 57, 65, 81, 89, 95, 102, 110, 118, 125]
    y2=[8, 11.2, 15.3, 18.2, 20.4, 25.1, 26.3, 29.5, 29.7, 32.6, 32.4, 33]
    plt.plot(x, y1, "-ro", label="моделювання")
    plt.plot(x2, y2, "go", markersize=7, markerfacecolor='#ffffff',
    label="експеримент")
    plt.title("D = f(A), II бонітет")
    plt.xlabel("A, років")
    plt.ylabel("D, см")
    plt.legend()
    plt.grid()
    plt.show()
def sosnad3():
    x=np.arange(10, 150, 10)
    y1=-0.0007*x**2+0.3389*x-1.2234
    x2=[20, 37, 54, 62, 71, 77, 84, 89, 94, 98, 105, 111, 117, 121, 127]
    y2=[6, 9.1, 16.3, 17.1, 18, 21.6, 22.8, 23.3, 23.7, 25.6, 27.2, 27.8, 29.1, 29.4,
        32.3]
    plt.plot(x, y1, "-ro", label="моделювання")
```

```
    plt.plot(x2, y2, "go", markersize=7, markerfacecolor='#ffffff',
    label="експеримент")
    plt.title("D = f(A), III бонітет")
    plt.xlabel("A, років")
    plt.ylabel("D, см")
    plt.legend()
    plt.grid()
    plt.show()

def sosnad4():
    x=np.arange(10, 150, 10)
    y1=-0.0006*x**2+0.2842*x-1.402
    x2=[14, 18, 23, 28, 32, 36, 41, 48, 54, 58, 61, 64, 68, 72, 78, 85, 97, 106, 118,
    128]
    y2=[3.2, 3.3, 5.7, 6.5, 7.8, 8.5, 8.7, 9.4, 12.5, 13.2, 14, 14.1, 14.4, 15.1, 16,
    17.1, 18.3, 23, 24.2, 26.3]
    plt.plot(x, y1, "-ro", label="моделювання")
    plt.plot(x2, y2, "go", markersize=7, markerfacecolor='#ffffff',
    label="експеримент")
    plt.title("D = f(A), IV бонітет")
    plt.xlabel("A, років")
    plt.ylabel("D, см")
    plt.legend()
    plt.grid()
    plt.show()
def sosnad5():
    x=np.arange(10, 150, 10)
    y1=-0.0005*x**2+0.2327*x-1.593
    x2=[27, 38, 46, 55, 73, 82, 88, 93, 98, 106]
    y2=[5, 7, 8.4, 9, 13.5, 14, 14.2, 15, 17, 18.5]
    plt.plot(x, y1, "-ro", label="моделювання")
    plt.plot(x2, y2, "go", markersize=7, markerfacecolor='#ffffff',
    label="експеримент")
    plt.title("D = f(A), V бонітет")
    plt.xlabel("A, років")
    plt.ylabel("D, см")
    plt.legend()
    plt.grid()
    plt.show()
def sosnaS():
    x=np.arange(10, 150, 10)
    y1=-0.0012*x**2+0.3838*x+2.1222
```

```
    y2=-0.001*x**2+0.3359*x+1.3182
    y3=-0.0008*x**2+0.289*x+0.5123
    y4=-0.0007*x**2+0.2417*x-0.168
    y5=-0.0005*x**2+0.1908*x-0.5406
    plt.plot(x, y1, "-ro", label="I бонітет")
    plt.plot(x, y2, "-gs", label="II бонітет")
    plt.plot(x, y3, "-kx", label="III бонітет")
    plt.plot(x, y4, "-ks", label="IV бонітет")
    plt.plot(x, y5, "-k*", label="V бонітет")
    plt.title("S = f(A)")
    plt.xlabel("A, років")
    plt.ylabel("S, м2/га")
    plt.legend()
    plt.grid()
    plt.show()
def sosnaS1():
    x=np.arange(10, 150, 10)
    y1=-0.0012*x**2+0.3838*x+2.1222
    x2=[11, 27, 35, 42, 47, 53, 55, 62, 68, 73, 77, 84, 92, 94, 100, 120, 130]
    y2=[7, 10, 15, 16, 17, 18, 19, 22, 22.5, 23.2, 24.1, 25.2, 27.4, 28.1, 28.7, 31.5,
    33.4]
    plt.plot(x, y1, "-ro", label="моделювання")
    plt.plot(x2, y2, "go", markersize=7, markerfacecolor='#ffffff',
    label="експеримент")
    plt.title("S = f(A), I бонітет")
    plt.xlabel("A, років")
    plt.ylabel("S, м2/га")
    plt.legend()
    plt.grid()
    plt.show()
def sosnaS2():
    x=np.arange(10, 150, 10)
    y1=-0.001*x**2+0.3359*x+1.3182
    x2=[20, 35, 42, 57, 65, 81, 89, 95, 102, 110, 118, 125]
    y2=[9, 12.2, 13.6, 18, 18.1, 24, 25, 25.5, 26, 28, 27, 26]
    plt.plot(x, y1, "-ro", label="моделювання")
    plt.plot(x2, y2, "go", markersize=7, markerfacecolor='#ffffff',
    label="експеримент")
    plt.title("S = f(A), II бонітет")
    plt.xlabel("A, років")
    plt.ylabel("S, м2/га")
    plt.legend()
```

```
    plt.grid()
    plt.show()
def sosnaS3():
    x=np.arange(10, 150, 10)
    y1=-0.0008*x**2+0.289*x+0.5123
    x2=[20, 37, 54, 62, 71, 77, 84, 89, 94, 98, 105, 111, 117, 121, 127]
    y2=[7, 12, 13, 14.4, 16, 17.2, 19, 20, 21, 21.5, 22.2, 23, 23.5, 24, 23.5]
    plt.plot(x, y1, "-ro", label="моделювання")
    plt.plot(x2, y2, "go", markersize=7, markerfacecolor='#ffffff',
    label="експеримент")
    plt.title("S = f(A), III бонітет")
    plt.xlabel("A, років")
    plt.ylabel("S, м2/га")
    plt.legend()
    plt.grid()
    plt.show()
def sosnaS4():
    x=np.arange(10, 150, 10)
    y1=-0.0007*x**2+0.2417*x-0.168
    x2=[14, 18, 23, 28, 32, 36, 41, 48, 54, 58, 61, 64, 68, 72, 78, 85, 97, 106, 118,
    128]
    y2=[3, 4.2, 4.6, 5.3, 5.9, 8, 9, 10.4, 11.1, 11.7, 12, 12.1, 12.6, 14, 14.1, 14.6,
    17.2, 18, 19.4, 18.7]
    plt.plot(x, y1, "-ro", label="моделювання")
    plt.plot(x2, y2, "go", markersize=7, markerfacecolor='#ffffff',
    label="експеримент")
    plt.title("S = f(A), IV бонітет")
    plt.xlabel("A, років")
    plt.ylabel("S, м2/га")
    plt.legend()
    plt.grid()
    plt.show()
def sosnaS5():
    x=np.arange(10, 150, 10)
    y1=-0.0005*x**2+0.1908*x-0.5406
    x2=[27, 38, 46, 55, 73, 82, 88, 93, 98, 106]
    y2=[5, 5.5, 5.8, 9.5, 10.2, 11.4, 12, 12.4, 12.5, 13]
    plt.plot(x, y1, "-ro", label="моделювання")
    plt.plot(x2, y2, "go", markersize=7, markerfacecolor='#ffffff',
    label="експеримент")
    plt.title("S = f(A), V бонітет")
    plt.xlabel("A, років")
```

```
    plt.ylabel("S, м2/га")
    plt.legend()
    plt.grid()
    plt.show()
def sosnaV():
    x=np.arange(10, 150, 10)
    y1=-0.0383*x**2+14.638*x-120.77
    y2=-0.0296*x**2+11.754*x-105.47
    y3=-0.0221*x**2+9.1791*x-92.352
    y4=-0.0157*x**2+6.9096*x-74.433
    y5=-0.0075*x**2+2.8746*x-28.88
    plt.plot(x, y1, "-ro", label="I бонітет")
    plt.plot(x, y2, "-gs", label="II бонітет")
    plt.plot(x, y3, "-kx", label="III бонітет")
    plt.plot(x, y4, "-ks", label="IV бонітет")
    plt.plot(x, y5, "-k*", label="V бонітет")
    plt.title("V = f(A)")
    plt.xlabel("A, років")
    plt.ylabel("V, м3/га")
    plt.legend()
    plt.grid()
    plt.show()
def sosnaV1():
    x=np.arange(10, 150, 10)
    y1=-0.0383*x**2+14.638*x-120.77
    x2=[11, 27, 35, 42, 47, 53, 55, 62, 68, 73, 77, 84, 92, 94, 100, 120, 130]
    y2=[45, 260, 330, 410, 505, 530, 590, 660, 685, 750, 760, 850, 880, 915, 1000,
    1050, 1200]
    plt.plot(x, y1, "-ro", label="моделювання")
    plt.plot(x2, y2, "go", markersize=7, markerfacecolor='#ffffff',
label="експеримент")
    plt.title("V = f(A), I бонітет")
    plt.xlabel("A, років")
    plt.ylabel("V, м3/га")
    plt.legend()
    plt.grid()
    plt.show()
def sosnaV2():
    x=np.arange(10, 150, 10)
    y1=-0.0296*x**2+11.754*x-105.47
    x2=[20, 35, 42, 57, 65, 81, 89, 95, 102, 110, 118, 125]
    y2=[115, 250, 360, 440, 550, 630, 725, 735, 800, 840, 850, 920]
```

```
    plt.plot(x, y1, "-ro", label="моделювання")
    plt.plot(x2, y2, "go", markersize=7, markerfacecolor='#ffffff',
    label="експеримент")
    plt.title("V = f(A), II бонітет")
    plt.xlabel("A, років")
    plt.ylabel("V, м3/га")
    plt.legend()
    plt.grid()
    plt.show()
def sosnaV3():
    x=np.arange(10, 150, 10)
    y1=-0.0221*x**2+9.1791*x-92.352
    x2=[20, 37, 54, 62, 71, 77, 84, 89, 94, 98, 105, 111, 117, 121, 127]
    y2=[80, 200, 350, 375, 460, 470, 530, 540, 585, 600, 615, 635, 650, 710, 725]
    plt.plot(x, y1, "-ro", label="моделювання")
    plt.plot(x2, y2, "go", markersize=7, markerfacecolor='#ffffff',
    label="експеримент")
    plt.title("V = f(A), III бонітет")
    plt.xlabel("A, років")
    plt.ylabel("V, м3/га")
    plt.legend()
    plt.grid()
    plt.show()
def sosnaV4():
    x=np.arange(10, 150, 10)
    y1=-0.0157*x**2+6.9096*x-74.433
    x2=[14, 18, 23, 28, 32, 36, 41, 48, 54, 58, 61, 64, 68, 72, 78, 85, 97, 106, 118,
    128]
    y2=[15, 50, 60, 120, 125, 170, 180, 240, 250, 260, 270, 315, 335, 355, 360, 420,
    460, 470, 500, 565]
    plt.plot(x, y1, "-ro", label="моделювання")
    plt.plot(x2, y2, "go", markersize=7, markerfacecolor='#ffffff',
    label="експеримент")
    plt.title("V = f(A), IV бонітет")
    plt.xlabel("A, років")
    plt.ylabel("V, м3/га")
    plt.legend()
    plt.grid()
    plt.show()
def sosnaV5():
    x=np.arange(10, 150, 10)
    y1=-0.0075*x**2+2.8746*x-28.88
```

```
    x2=[27, 38, 46, 55, 73, 82, 88, 93, 98, 106]
    y2=[40, 60, 95, 120, 130, 142, 155, 164, 187, 200]
    plt.plot(x, y1, "-ro", label="моделювання")
    plt.plot(x2, y2, "go", markersize=7, markerfacecolor='#ffffff',
    label="експеримент")
    plt.title("V = f(A), V бонітет")
    plt.xlabel("A, років")
    plt.ylabel("V, м3/га")
    plt.legend()
    plt.grid()
    plt.show()
def osh():
    t1.delete(0, END)
    t2.delete(0, END)
    t3.delete(0, END)
    t4.delete(0, END)
def rozh1():
    r=int(t2.get())
    b=-0.0012*r**2+0.3838*r+2.1322
    t4.insert(0, str(b))
def sosna():
    win2 = Tk()
    win2.title("Сосна")
    win2.geometry("1350x650")
    win2.resizable(False, False)
    tab_control=ttk.Notebook(win2)
    tab1=ttk.Frame(tab_control)
    tab2=ttk.Frame(tab_control)
    tab3=ttk.Frame(tab_control)
    tab4=ttk.Frame(tab_control)
    tab_control.add(tab1, text="Висота")
    tab_control.add(tab2, text="Діаметр")
    tab_control.add(tab3, text="Площа")
    tab_control.add(tab4, text="Об'єм")
    tab_control.pack(expand=1, fill="both")
    btn1=Button(tab1, text="Математична модель", font=("Arial 11 bold"),
    command=sosnah)
    btn1.place(x=10, y=450, width=200)
    btn2=Button(tab1, text="I бонітет", font=("Arial 11 bold"), command=sosnah1)
    btn2.place(x=50, y=150, width=200)
    btn3=Button(tab1, text="II бонітет", font=("Arial 11 bold"), command=sosnah2)
    btn3.place(x=50, y=200, width=200)
```

```
btn4=Button(tab1, text="III бонітет", font=("Arial 11 bold"), command=sosnah3)
btn4.place(x=50, y=250, width=200)
btn5=Button(tab1, text="IV бонітет", font=("Arial 11 bold"), command=sosnah4)
btn5.place(x=50, y=300, width=200)
btn6=Button(tab1, text="V бонітет", font=("Arial 11 bold"), command=sosnah5)
btn6.place(x=50, y=350, width=200)
btn7=Button(tab1, text="Обчислити", font=("Arial 11 bold"), command=rozh1)
btn7.place(x=230, y=450, width=200)
btn8=Button(tab1, text="Очистити", font=("Arial 11 bold"), command=osh)
btn8.place(x=450, y=450, width=200)
t1 = Entry(tab1, bg="yellow", fg="green",font=("Arial 11 bold"))
t1.place(x=300, y=150, width=150, height=36)
t2 = Entry(tab1, bg="yellow", fg="green",font=("Arial 11 bold"))
t2.place(x=300, y=200, width=150, height=36)
t3 = Entry(tab1, bg="yellow", fg="green",font=("Arial 11 bold"))
t3.place(x=300, y=250, width=150, height=36)
t4 = Entry(tab1, bg="yellow", fg="green",font=("Arial 11 bold"))
t4.place(x=300, y=350, width=150, height=36)
lbl1=Label(tab1, text="№ дерева", fg="blue", font="Arial 11 bold")
lbl1.place(x=460, y=150)
lbl2=Label(tab1, text="Вік A, років", fg="blue", font="Arial 11 bold")
lbl2.place(x=460, y=200)
lbl3=Label(tab1, text="Облікова висота H, м", fg="blue", font="Arial 11 bold")
lbl3.place(x=460, y=250)
lbl4=Label(tab1, text="Модельна висота H, м", fg="blue", font="Arial 11 bold")
lbl4.place(x=460, y=350)
lbl5=Label(tab1, text="Залежність висоти дерев від віку", fg="blue", font="Arial 20
bold")
lbl5.place(x=400, y=20)
btn1=Button(tab2, text="Математична модель", font=("Arial 11 bold"),
command=sosnah)
btn1.place(x=10, y=450, width=200)
btn2=Button(tab2, text="I бонітет", font=("Arial 11 bold"), command=sosnah1)
btn2.place(x=50, y=150, width=200)
btn3=Button(tab2, text="II бонітет", font=("Arial 11 bold"), command=sosnah2)
btn3.place(x=50, y=200, width=200)
btn4=Button(tab2, text="III бонітет", font=("Arial 11 bold"), command=sosnah3)
btn4.place(x=50, y=250, width=200)
btn5=Button(tab2, text="IV бонітет", font=("Arial 11 bold"), command=sosnah4)
btn5.place(x=50, y=300, width=200)
btn6=Button(tab2, text="V бонітет", font=("Arial 11 bold"), command=sosnah5)
btn6.place(x=50, y=350, width=200)
```

```
    btn7=Button(tab2, text="Обчислити", font=("Arial 11 bold"), command=rozh1)
    btn7.place(x=230, y=450, width=200)
    btn8=Button(tab2, text="Очистити", font=("Arial 11 bold"), command=osh)
    btn8.place(x=450, y=450, width=200)
    t1 = Entry(tab2, bg="yellow", fg="green",font=("Arial 11 bold"))
    t1.place(x=300, y=150, width=150, height=36)
    t2 = Entry(tab2, bg="yellow", fg="green",font=("Arial 11 bold"))
    t2.place(x=300, y=200, width=150, height=36)
    t3 = Entry(tab2, bg="yellow", fg="green",font=("Arial 11 bold"))
    t3.place(x=300, y=250, width=150, height=36)
    t4 = Entry(tab2, bg="yellow", fg="green",font=("Arial 11 bold"))
    t4.place(x=300, y=350, width=150, height=36)
    lbl1=Label(tab2, text="№ дерева", fg="blue", font="Arial 11 bold")
    lbl1.place(x=460, y=150)
    lbl2=Label(tab2, text="Вік A, років", fg="blue", font="Arial 11 bold")
    lbl2.place(x=460, y=200)
    lbl3=Label(tab2, text="Обліковий діаметр D, см", fg="blue", font="Arial 11 bold")
    lbl3.place(x=460, y=250)
    lbl4=Label(tab2, text="Модельний діаметр D, см", fg="blue", font="Arial 11 bold")
    lbl4.place(x=460, y=350)
    lbl5=Label(tab2, text="Залежність діаметра дерев від віку", fg="blue", font="Arial
20 bold")
    lbl5.place(x=400, y=20)
    btn1=Button(tab3,    text="Математична    модель",    font=("Arial    11    bold"),
command=sosnah)
    btn1.place(x=10, y=450, width=200)
    btn2=Button(tab3, text="I бонітет", font=("Arial 11 bold"), command=sosnah1)
    btn2.place(x=50, y=150, width=200)
    btn3=Button(tab3, text="II бонітет", font=("Arial 11 bold"), command=sosnah2)
    btn3.place(x=50, y=200, width=200)
    btn4=Button(tab3, text="III бонітет", font=("Arial 11 bold"), command=sosnah3)
    btn4.place(x=50, y=250, width=200)
    btn5=Button(tab3, text="IV бонітет", font=("Arial 11 bold"), command=sosnah4)
    btn5.place(x=50, y=300, width=200)
    btn6=Button(tab3, text="V бонітет", font=("Arial 11 bold"), command=sosnah5)
    btn6.place(x=50, y=350, width=200)
    btn7=Button(tab3, text="Обчислити", font=("Arial 11 bold"), command=rozh1)
    btn7.place(x=230, y=450, width=200)
    btn8=Button(tab3, text="Очистити", font=("Arial 11 bold"), command=osh)
    btn8.place(x=450, y=450, width=200)
    t1 = Entry(tab3, bg="yellow", fg="green",font=("Arial 11 bold"))
    t1.place(x=300, y=150, width=150, height=36)
```

```
t2 = Entry(tab3, bg="yellow", fg="green",font=("Arial 11 bold"))
t2.place(x=300, y=200, width=150, height=36)
t3 = Entry(tab3, bg="yellow", fg="green",font=("Arial 11 bold"))
t3.place(x=300, y=250, width=150, height=36)
t4 = Entry(tab3, bg="yellow", fg="green",font=("Arial 11 bold"))
t4.place(x=300, y=350, width=150, height=36)
lbl1=Label(tab3, text="Кількість дерев", fg="blue", font="Arial 11 bold")
lbl1.place(x=460, y=150)
lbl2=Label(tab3, text="Вік А, років", fg="blue", font="Arial 11 bold")
lbl2.place(x=460, y=200)
lbl3=Label(tab3, text="Облікова середня площа S, м2/га", fg="blue",
font="Arial 11 bold")
lbl3.place(x=460, y=250)
lbl4=Label(tab3, text="Модельна середня площа S, м2/га", fg="blue",
font="Arial 11 bold")
lbl4.place(x=460, y=350)
lbl5=Label(tab3, text="Залежність середнього поперечного перерізу від віку",
fg="blue", font="Arial 20 bold")
lbl5.place(x=400, y=20)
btn1=Button(tab4, text="Математична модель", font=("Arial 11 bold"),
command=sosnah)
btn1.place(x=10, y=450, width=200)
btn2=Button(tab4, text="I бонітет", font=("Arial 11 bold"), command=sosnah1)
btn2.place(x=50, y=150, width=200)
btn3=Button(tab4, text="II бонітет", font=("Arial 11 bold"), command=sosnah2)
btn3.place(x=50, y=200, width=200)
btn4=Button(tab4, text="III бонітет", font=("Arial 11 bold"), command=sosnah3)
btn4.place(x=50, y=250, width=200)
btn5=Button(tab4, text="IV бонітет", font=("Arial 11 bold"), command=sosnah4)
btn5.place(x=50, y=300, width=200)
btn6=Button(tab4, text="V бонітет", font=("Arial 11 bold"), command=sosnah5)
btn6.place(x=50, y=350, width=200)
btn7=Button(tab4, text="Обчислити", font=("Arial 11 bold"), command=rozh1)
btn7.place(x=230, y=450, width=200)
btn8=Button(tab4, text="Очистити", font=("Arial 11 bold"), command=osh)
btn8.place(x=450, y=450, width=200)
t1 = Entry(tab4, bg="yellow", fg="green",font=("Arial 11 bold"))
t1.place(x=300, y=150, width=150, height=36)
t2 = Entry(tab4, bg="yellow", fg="green",font=("Arial 11 bold"))
t2.place(x=300, y=200, width=150, height=36)
t3 = Entry(tab4, bg="yellow", fg="green",font=("Arial 11 bold"))
t3.place(x=300, y=250, width=150, height=36)
```

```
    t4 = Entry(tab4, bg="yellow", fg="green",font=("Arial 11 bold"))
    t4.place(x=300, y=350, width=150, height=36)
    lbl1=Label(tab4, text="Кількість дерев", fg="blue", font="Arial 11 bold")
    lbl1.place(x=460, y=150)
    lbl2=Label(tab4, text="Вік А, років", fg="blue", font="Arial 11 bold")
    lbl2.place(x=460, y=200)
    lbl3=Label(tab4, text="Обліковий середній об'єм V, м3/га", fg="blue", font="Arial
 11 bold")
    lbl3.place(x=460, y=250)
    lbl4=Label(tab4, text="Модельний середній об'єм V, м3/га", fg="blue", font="Arial
    11 bold")
    lbl4.place(x=460, y=350)
    lbl5=Label(tab4, text="Залежність середнього об'єму дерев від віку", fg="blue",
    font="Arial 20 bold")
    lbl5.place(x=400, y=20)
    win2.mainloop()

root = Tk()
root.title("Облік лісових масивів")
root.resizable(False, False)
root.geometry("1350x650")
golowneMenu = Menu()
root.config(menu=golowneMenu)
fileMenu = Menu(golowneMenu)
golowneMenu.add_cascade(label="Файл", menu=fileMenu)
fileMenu.add_command(label="Новий", command=sosnah)
fileMenu.add_command(label="Відкрити", command=open_file)
fileMenu.add_command(label="Зберегти", command=save_file)
fileMenu.add_separator()
fileMenu.add_command(label="Вихід", command=a1)
modelMenu = Menu(golowneMenu)
golowneMenu.add_cascade(label="Моделювання", menu=modelMenu)
podMenu = Menu(modelMenu)
modelMenu.add_cascade(label="Порода", menu=podMenu)
podMenu.add_command(label="Сосна", command=sosna)
podMenu.add_command(label="Смерека", command=smereka)
podMenu.add_command(label="Дуб", command=dub)
podMenu.add_command(label="Граб", command=grab)
podMenu.add_command(label="Береза", command=bereza)
helpMenu = Menu(golowneMenu)
golowneMenu.add_cascade(label="Допомога", menu=helpMenu)
```

```
helpMenu.add_command(label="Довідка")
helpMenu.add_command(label="Відомості про автора")
lbl1 = Label(root, text="Облік лісових насаджень", fg="blue", font="Arial 20 bold")
lbl1.place(x=450, y=10)

root.mainloop()
```

ДОДАТОК Б.

Lis.py

```
import tkinter as tk
from tkinter import ttk
import sqlite3

class Main(tk.Frame):
    def __init__(self, root):
        super().__init__(root)
        self.init_main()
        self.db = db
        self.view_records()
    def init_main(self):
        toolbar = tk.Frame(bg='f0f0f0')
        toolbar.pack(side=tk.TOP, fill=tk.X)
        self.plus_img = tk.PhotoImage(file='plus.gif')
        btn_open_dialog = tk.Button(toolbar, text='Добавити запис',
         command=self.open_dialog, bg='#d7d8e0', bd=0, compound=tk.TOP,
         image=self.plus_img)
        btn_open_dialog.pack(side=tk.LEFT)
        self.upd_img = tk.PhotoImage(file='upd.gif')
        btn_edit_dialog = tk.Button(toolbar, text='Редактувати запис',
         bg='#f0f0f0', bd=0, image=self.upd_img, compound=tk.TOP,
         command=self.open_upd_dialog)
        btn_edit_dialog.pack(side=tk.LEFT)
        self.del_img = tk.PhotoImage(file='del.gif')
        btn_delete = tk.Button(toolbar, text='Видалити запис', bg='#f0f0f0', bd=0,
         image=self.delete_img, compound=tk.TOP, command=self.del_records)
        btn_delete.pack(side=tk.LEFT)
        self.sear_img = tk.PhotoImage(file='sear.gif')
        btn_search = tk.Button(toolbar, text='Пошук', bg='#f0f0f0', bd=0,
         image=self.sear_img, compound=tk.TOP, command=self.open_sear_dialog)
        btn_search.pack(side=tk.LEFT)
        self.ref_img = tk.PhotoImage(file='ref.gif')
        btn_refresh = tk.Button(toolbar, text='Обновити', bg='#f0f0f0', bd=0,
         image=self.ref_img, compound=tk.TOP, command=self.view_records)
        btn_refresh.pack(side=tk.LEFT)
        self.tree = ttk.Treeview(self, columns=('ID', 'height', 'bonitet',
         'diameter'), height=15, show='headings')
```

```
        self.tree.column('ID', width=30, anchor=tk.CENTER)
        self.tree.column('height', width=365, anchor=tk.CENTER)
        self.tree.column('bonitet', width=150, anchor=tk.CENTER)
        self.tree.column('diameter', width=100, anchor=tk.CENTER)
        self.tree.heading('ID', text='№ дерева')
        self.tree.heading('height', text='Висота')
        self.tree.heading('bonitet', text='Клас бонітету')
        self.tree.heading('diameter', text='Діаметр')
        self.tree.pack(side=tk.LEFT)
        scroll = tk.Scrollbar(self, command=self.tree.yview)
        scroll.pack(side=tk.LEFT, fill=tk.Y)
        self.tree.configure(yscrollcommand=scroll.set)
    def records(self, height, bonitet, diameter):
        self.db.insert_data(height, bonitet, diameter)
        self.view_records()
    def update_record(self, height, bonitet, diameter):
        self.db.c.execute('''UPDATE lis SET height=?, bonitet =?, diameter =?
        WHERE ID=?''', (height, bonitet, diameter,
        self.tree.set(self.tree.selection()[0], '#1')))
        self.db.conn.commit()
        self.view_records()
    def view_records(self):
        self.db.c.execute('''SELECT * FROM lis''')
        [self.tree.delete(i) for i in self.tree.get_children()]
        [self.tree.insert('', 'end', values=row) for row in self.db.c.fetchall()]
    def delete_records(self):
        for selection_item in self.tree.selection():
            self.db.c.execute('''DELETE FROM lis WHERE id=?''',
           (self.tree.set(selection_item, '#1'),))
        self.db.conn.commit()
        self.view_records()
    def search_records(self, height):
        description = ('%' + height + '%',)
        self.db.c.execute('''SELECT * FROM lis WHERE height LIKE ?''',
        height)
        [self.tree.delete(i) for i in self.tree.get_children()]
        [self.tree.insert('', 'end', values=row) for row in self.db.c.fetchall()]
    def open_dialog(self):
        Child()
    def open_update_dialog(self):
        Update()
```

```
        def open_search_dialog(self):
            Search()
    class Child(tk.Toplevel):
        def __init__(self):
            super().__init__(root)
            self.init_child()
            self.view = app
        def init_child(self):
            self.title('Добавити новий запис')
            self.geometry('400x220+400+300')
            self.resizable(False, False)
            label_description = tk.Label(self, text='Висота:')
            label_description.place(x=50, y=50)
            label_select = tk.Label(self, text='Клас бонітету')
            label_select.place(x=50, y=80)
            label_sum = tk.Label(self, text='Діаметр:')
            label_sum.place(x=50, y=110)
            self.entry_description = ttk.Entry(self)
            self.entry_description.place(x=200, y=50)
            self.entry_money = ttk.Entry(self)
            self.entry_money.place(x=200, y=110)
            self.combobox = ttk.Combobox(self, values=[u'I', u'II', u'III', u'IV',
            u'V'])
            self.combobox.current(0)
            self.combobox.place(x=200, y=80)
            btn_cancel = ttk.Button(self, text='Закрити', command=self.destroy)
            btn_cancel.place(x=300, y=170)
            self.btn_ok = ttk.Button(self, text='Добавити')
            self.btn_ok.place(x=220, y=170)
            self.btn_ok.bind('<Button-1>', lambda event:
             self.view.records(self.entry_description.get())

self.combobox.get(),

self.entry_money.get()))

            self.grab_set()
            self.focus_set()
    class Update(Child):
        def __init__(self):
            super().__init__()
            self.init_edit()
```

```
            self.view = app
            self.db = db
            self.default_data()
        def init_edit(self):
            self.title('Редактувати запис')
            btn_edit = ttk.Button(self, text='Редактувати')
            btn_edit.place(x=205, y=170)
            btn_edit.bind('<Button-1>', lambda event:
             self.view.update_record(self.entry_height.get(),

self.combobox.get(),

self.entry_money.get()))
            self.btn_ok.destroy()
        def default_data(self):
            self.db.c.execute('''SELECT * FROM lis WHERE id=?''',
                              (self.view.tree.set(self.view.tree.selection()[0],
'#1'),))
            row = self.db.c.fetchone()
            self.entry_description.insert(0, row[1])
            if row[2] != 'Діаметр':
                self.combobox.current(1)
            self.entry_money.insert(0, row[3])
    class Search(tk.Toplevel):
        def __init__(self):
            super().__init__()
            self.init_search()
            self.view = app
        def init_search(self):
            self.title('Пошук')
            self.geometry('300x100+400+300')
            self.resizable(False, False)
            label_search = tk.Label(self, text='Пошук')
            label_search.place(x=50, y=20)
            self.entry_search = ttk.Entry(self)
            self.entry_search.place(x=105, y=20, width=150)
            btn_cancel = ttk.Button(self, text='Закрити', command=self.destroy)
            btn_cancel.place(x=185, y=50)
            btn_search = ttk.Button(self, text='Пошук')
            btn_search.place(x=105, y=50)
            btn_search.bind('<Button-1>', lambda event:
             self.view.search_records(self.entry_search.get()))
```

```
        btn_search.bind('<Button-1>', lambda event: self.destroy(), add='+')
class DB:
    def __init__(self):
        self.conn = sqlite3.connect('lis.db')
        self.c = self.conn.cursor()
        self.c.execute(
            '''CREATE TABLE IF NOT EXISTS lis (id integer primary key, height
             real, bonitet text, diameter real)''')
        self.conn.commit()
    def insert_data(self, height, bonitet, diameter):
        self.c.execute('''INSERT INTO lis(height, bonitet, diameter) VALUES (?, ?,
                    ?)''', (height, bonitet, diameter))
        self.conn.commit()
if __name__ == "__main__":
    root = tk.Tk()
    db = DB()
    app = Main(root)
    app.pack()
    root.title("Облік лісових насаджень")
    root.geometry("700x500+300+200")
    root.resizable(False, False)
root.mainloop()
```