

ЗМІСТ

	с.
Вступ.....	6
1 Постановка задачі.....	7
2 Основна частина.....	8
2.1 Аналіз вхідних та вихідних даних.....	8
2.2 Методи та засоби програмування	8
2.3 Опис логічної структури програми.....	14
2.4 Опис фізичної структури програми	19
2.5 Інструкція користувача програми.....	29
2.6 Вимоги до складу та параметрів технічних та програмних засобів.....	37
2.7 Тестування програми.....	37
2.8 Результати реалізації програми.....	40
Висновки по роботі.....	41
Перелік використаних джерел.....	42
Додаток А Блок-схеми.....	43
Додаток Б Лістинг програми.....	53

					ФКЗЕ.121ООП00.КРПЗ			
Змін	Арк	№ докум	Підпис	Дата				
Разроб.		Коваленко К М			Розробити класи за паттерном «Легковаговик», що абстрактно допомагає заощадити оперативну пам'ять при відображенні на екрані множини об'єктів-дерев	Літера	Аркуш	Аркушів
Перевір.		Логвіненко В.В.					5	55
Н.Контр		Логвіненко В.В.				група ПЗ-21-1/9		
Затв.		Саприкіна І.Г.						

ВСТУП

У сучасному технологічному світі велику частину інформації займає графічна інформація. За останні роки середня кількість візуальних об'єктів в наукових матеріалах виросла майже втричі, що призвело до великої ресурсозатратності з боку технічної частини нашого життя. На жаль, не всі люди можуть забезпечити себе подібними апаратними та програмними засобами для дослідження і обробки сучасної візуальної інформації. Тому дуже важливо забезпечити гарну оптимізацію при обробці множин графічних об'єктів.

При обробці візуальної інформації найбільше страждає оперативна пам'ять. І тому для забезпечення ефективності під час подібного виду робіт треба звернути свою увагу на зменшення навантаження на ОП. для зменшення навантаження на RAM можна використовувати різні методи та інструменти.

Основними методами для забезпечення оптимізації можна вважати:

- метод лінійної ініціалізації - це метод програмування, який відкладає створення та ініціалізацію об'єкта або значення до моменту, коли вони дійсно стануть необхідними. В контексті об'єктно-орієнтованого програмування це означає, що об'єкт не буде створений або ініціалізований, поки не буде викликаний його метод або не буде здійснений доступ до його атрибуту;

- метод стиснення даних - це техніка, що використовується для зменшення розміру даних без втрати їх суттєвої інформації. Це досягається шляхом видалення надлишковості, яка присутня у вихідних даних;

- метод використання спеціалізованих структур даних - це спосіб організації даних у комп'ютерах, де використовуються конкретні типи структур даних, оптимізовані під певні завдання або алгоритми. Ці спеціалізовані структури дозволяють досягти більшої ефективності, простоти, надійності або безпеки обробки даних, порівняно з загальними структурами даних, такими як масиви, списки, словники та інші методи.

Метою курсової роботи є розробка програми з використанням класів -

					ФКЗЕ.121ООП00.КРПЗ	Арк
						6
Змін	Арк	№ докум	Підпис	Дата		

TreeType класу, похідного від нього TreeFactory , застосовуючі концепції ООП, а саме - успадкування класів та поліморфізма. Також для реалізації завдання були використані динамічні масиви даних стандартної бібліотеки шаблонів STL- контейнер map та патерн Легковаговик

					ФКЗЕ.121ООП00.КРПЗ	Арк
						7
Змін	Арк	№ докум	Підпис	Дата		

1 ПОСТАНОВКА ЗАДАЧІ

Розробити контейнерний клас для збереження та обробки масиву фабрики фрагментів вузлів дерева за паттерном «Легковаговик». Для цього розробити класи:

- графічного об'єкту – TreeType, який описує не унікальні поля об'єкта дерева;
- фабрика легковаговиків – TreeFactory, яка визначає метод створення/ використання графічного об'єкту(TreeType,);
- збереження контекстного об'єкту Tree .

Програма курсової роботи повинна зберігати та зчитувати дані контейнеру у зовнішньому файлі, мати меню для обробки контейнеру (відкриття файлу з контейнером, збереження, редагування, додання, видалення записів, сортування, пошук, друк та інше).

					ФКЗЕ.121ООП00.КРПЗ	Арк
						8
Змін	Арк	№ докум	Підпис	Дата		

2 ОСНОВНА ЧАСТИНА

2.1 Аналіз вхідних та вихідних даних

Тестова програма розробленої ієрархії класів буде зберігати масив даних кожного графічного об'єкту, а саме: Ідентифікатор типу, колір, текстура, висота, ширина, код об'єкта.

Вхідними даними розробленої тестової програми є значення елементів ієрархії класів - код об'єкта, номер пункту меню для виконання дії над елементами контейнеру - deque, введених з клавіатури або отримані з зовнішнього файлу

Вихідні дані програми є: вивід даних ієрархії класів на екран або редагування/запис файлу за вимогами користувача.

2.2 Методи та засоби програмування

Для виконання поставленого завдання курсової роботи необхідні такі розділи програмування мовою C++, як:

- класи;
- абстрактні типи даних та інкапсуляція класу;
- агрегація;
- контейнер STL – map;
- інструменти роботи з файловими потоками вводу – виводу.

Інкапсуляція – це концепція об'єктно-орієнтованого програмування, що означає поєднання властивостей (атрибутів або даних) та методів в одне ціле. Деякі автори ототожнюють інкапсуляцію з приховуванням (англ. information hiding), у той час як інші ні (відокремлюють ці визначення). Наприклад, клас, що відповідає точці на декартовій площині, можна записати так:

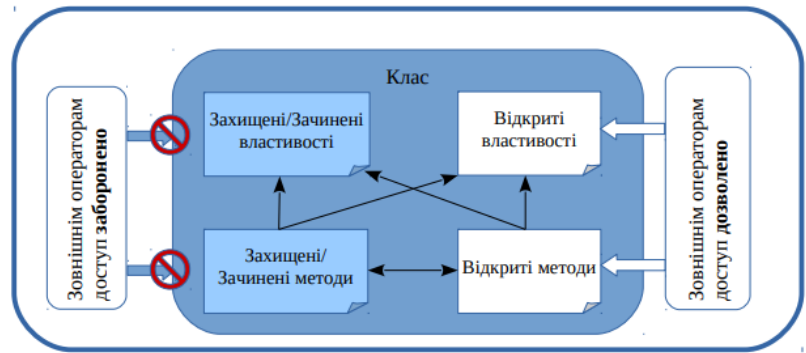


Рисунок 2.1 - Інкапсуляція

```
class CPoint
{
public:           //Специфікатор доступу
    double x;    //властивість (данні-член)
    double y;    //властивість (данні-член)
    double Distance(CPoint &p); //метод (функція-член)
};
```

Рисунок 2.2 – Специфікатор доступу

У наведеному вище прикладі з'являється нове ключове слово – `public` – яке є специфікатором доступу. Всього таких специфікаторів три, а саме: `public`, `private`, `protected`; вони дозволяють керувати доступом до членів класу (рис. 4). Розглянемо більш детально призначення кожного з них: `public` – відкритий блок класу, будь-який елемент, що знаходиться в цьому блоці, буде доступний у цьому класі, його нащадках, та зовнішнім операторам; `private` – приватний (зачинений) блок класу, будь-який елемент, що знаходиться в цьому блоці, буде доступний лише в цьому класі. Використовується за замовчуванням; `protected` – захищений блок класу, будь-який елемент, що знаходиться в цьому блоці, буде доступний в цьому



класі та його нащадках.

Рисунок 2.3 – Класи та його нащадки

У класі можуть бути присутні численні відкриті, приватні або захищені секції, які можуть розташовуватись у довільному порядку, наприклад:

```
class CPerson
{
    string fname;          //приватний атрибут
public:
    void SetName(string); //відкритий метод
    string GetName();     //відкритий метод
private:
    string name;          //приватний атрибут
    string sname;         //приватний атрибут
public:
    string ToString();    //відкритий метод
    //.....
};
```

Рисунок 2.4 – Оголошення заголовних файлів

Зазвичай більшість класів краще оголошувати у заголовних файлах, які потім можна додавати до різноманітних програмних модулів з метою спільного використання одних і тих самих класів. Зазвичай для кожного поля певного об'єкта класу виділяється своя ділянка пам'яті. Це правило не діє, якщо оголошення поля супроводжується ключовим словом `static`. Пам'ять для статичних полів класу виділяється один раз, не залежно від того, скільки об'єктів існує у програмі. Кожен об'єкт класу має вказівник `this`, який фактично посилається на поточний об'єкт.[1]

Агрегація – це відношення "має – щось" ("has – a" англ.) означає, що деяке ціле може складатися з окремих частин, причому кожна з цих частин має право на окреме життя. На діаграмі класів агрегація позначається лінією з ромбом. Ромб вказує на ціле (контейнер)



Рисунок 2.5 - Агрегація

Кожна окрема частина, яка агрегується, може продовжувати своє існування, наприклад, мишу можна відімкнути від комп'ютера і з'єднати зі смартфоном. Час життя частини може бути більше часу життя контейнера.[2]

Абстракція даних Принцип абстракції даних полягає у тому, щоб приховати деталі реалізації певного типу даних. Це досягається відокремленням інтерфейсу – набору операцій, визначених для даної структури від його реалізації. До переваг використання абстракції даних в кодї належать: - Можливість легко змінити реалізацію операції не змінюючи поведінку типу(при зміні реалізації не змінюється код, який використовує цей тип даних); - Відсутність потреби знати про конкретну реалізацію та низькорівневі операції – достатньо знати лише поведінку типу. Основним підходом до реалізації абстракції даних в програмуванні стала концепція абстрактних типів даних. Абстрактний тип даних – це клас абстрактних об'єктів, які повністю характеризуються визначеними на них 7 операціями. Абстрактний тип даних визначає набір функцій, незалежних від конкретної реалізації типу, для оперування його значеннями. Використовуючи абстрактні типи даних в своєму кодї користувач не повинен знати, як саме вони реалізовані(як зберігають-

ся в пам'яті тощо), йому достатньо знати лише поведінку(семантику, властивості) цих типів. Абстрактні типи даних у мовах програмування часто реалізуються за допомогою модулів. Модуль – це набір зв'язаних процедур разом з даними, що піддаються обробці. У мовах, що підтримують об'єктно-орієнтовану парадигму, однією з форм реалізації абстрактних типів даних є визначені користувачем типи – класи[3]

Використовується для зберігання і вилучення даних з колекції, в якій кожен елемент є парою, що володіє одночасно значенням даних і ключем сортування. Значення ключа унікальне і застосовується для автоматичного сортування даних.

Значення елемента в зіставленні можна змінити безпосередньо. Значення ключа є константою і не може бути змінено. Замість цього значення ключів, пов'язані зі старими елементами, необхідно видалити і вставити нові значення ключів для нових елементів.

```
template <class Key,
        class Type,
        class Traits = less<Key>,
        class Allocator=allocator<pair <const Key, Type>>>
class map;
```

Рисунок 2.6 – Значення елемента

Параметри

- Key – тип даних ключа, який має зберігатися в об'єкті map;
- Type – тип даних елемента для збереження в map;
- Traits – тип, що надає об'єкт функції, який може порівняти два значення елементів у вигляді ключів сортування, щоб визначити їхній відносний порядок у .map. Цей аргумент є необов'язковим, і як значення за замовчуванням використовується бінарний предикат less<Key>.

У C++14 можна ввімкнути різнорідний пошук, вказавши `std::less<>` предикат, який не має параметрів типу.

`Allocator` - тип, що представляє збережений об'єкт розподільника, який інкапсулює відомості про виділення і звільнення пам'яті для зіставлення. Цей аргумент є необов'язковим, і значенням за замовчуванням є `allocator<pair<const Key,Type>>`.

Клас `map` у стандартній бібліотеці C++ - це:

- контейнер змінного розміру, що фактично витягує значення елементів на основі пов'язаних значень ключів;
- реверсивний, оскільки він надає двонаправлені ітератори для отримання доступу до його елементів;
- сортується, оскільки його елементи впорядковані за значеннями ключів відповідно до заданої функції порівняння;
- є унікальним, оскільки кожен його елемент повинен мати унікальний ключ;
- є контейнером асоціативної пари, оскільки його значення даних елементів відрізняються від його значень ключів. Шаблон класу, оскільки функціональність, що надається, є універсальною і незалежною від типу елемента або ключа. Типи даних, які використовуються для елементів і ключів, визначаються як параметри в шаблоні класу разом із функцією та розподільником порівняння.

Змін	Арк	№ докум	Підпис	Дата

ФКЗЕ.121ООП00.КРПЗ

Арк
14

2.3 Опис логічної структури програми

Для виконання завдання курсової роботи була спроектована та розроблена ієрархія класів для програми управління лісом. Опис логічної структури програми, що включає класи TreeType, Tree, TreeFactory, Forest, і як вони взаємодіють у системі управління лісом, наведено нижче.

Класи та їх призначення:

- TreeType – базовий клас, що містить загальну інформацію про дерево, включно з типом, кольором, текстурою, висотою та шириною. Цей клас слугує як база для специфічних типів дерев;
- Tree – клас, що наслідує TreeType, представляє конкретне дерево. Містить ідентифікатор та додаткові методи для керування індивідуальними деревами.
- TreeFactory – фабричний клас, який використовується для створення та управління об'єктами TreeType. Він забезпечує методи для створення дерев, читання, запису даних у файл, а також управління колекцією дерев;
- Forest – контейнерний клас, що включає в себе логіку для управління групою дерев (Tree). Він використовує клас TreeFactory для створення та зберігання дерев у векторі унікальних вказівників.

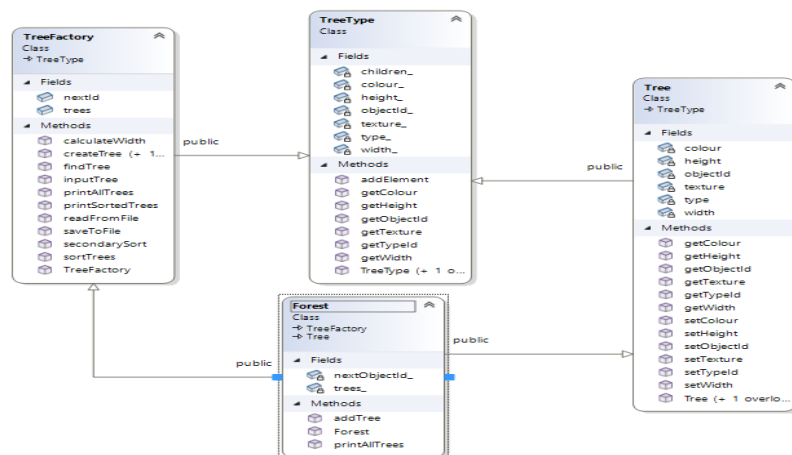


Рисунок 2.7 – Діаграма класів TreeType, Tree, TreeFactory, Forest

Структура визначення класів складається з властивостей/атрибутів та методів, які можуть мати різні рівні доступу (захисту), такі як: закриті (private), відкриті (public).

У якості атрибутів класу TreeType виступають базові типи даних:

- string type_ – символьний рядок для збереження типу дерева;
- string colour_ – символьний рядок для збереження кольору дерева;
- string texture_ – символьний рядок для збереження текстури дерева;
- double height_ – дійсна змінна для збереження висоти дерева;
- double width_ – дійсна змінна для збереження ширини дерева.

Оголошені методи класу TreeType мають відповідну сигнатуру та наступне призначення:

- TreeType() – конструктор класу без параметрів;
- TreeType(int objectId, string type, string colour, string texture, double height, double width) – конструктор класу з параметрами для ініціалізації даних-членів класу;
- геттери для доступу до даних членів класу.
- функції індивідуального доступу до даних членів класу TreeType:
- встановлення (set) значень атрибутам класу: не визначено в коді, оскільки основна робота з даними ведеться через конструктори та геттери;
- отримання (get) значень атрибутів класу: методи геттери для кожного атрибуту.
- перевантажені базові оператори для класу TreeType не визначені, але можуть включати перевантаження потокового виводу для зручності відображення об'єктів.

У якості атрибутів класу TreeFactory виступають:

- static map<int, TreeType*> trees – статичний контейнер для зберігання об'єктів дерев.

Оголошені методи класу TreeFactory мають відповідну сигнатуру та наступне

Змін	Арк	№ докум	Підпис	Дата

ФКЗЕ.121ООП00.КРПЗ

Арк
16

призначення:

- static TreeType* createTree(int objectId, string type, string colour, string texture, int height, int width) – статичний метод для створення та збереження нових дерев;

- static void saveToFile(const string& filename) – статичний метод для збереження даних у файл;

- static void readFromFile(const string& filename) – статичний метод для читання даних з файлу.

Перевантажені базові оператори для класу TreeFactory можуть включати методи для потокового вводу та виводу для зручності роботи з файлами.

Для тестування можливостей розробленої ієрархії класів, складена програма яка має виклики методів обробки контейнеру екземплярів класу Shedule та меню для виконання для їх виклику. Схема меню тестової програми наведена на рисунку 2.2.

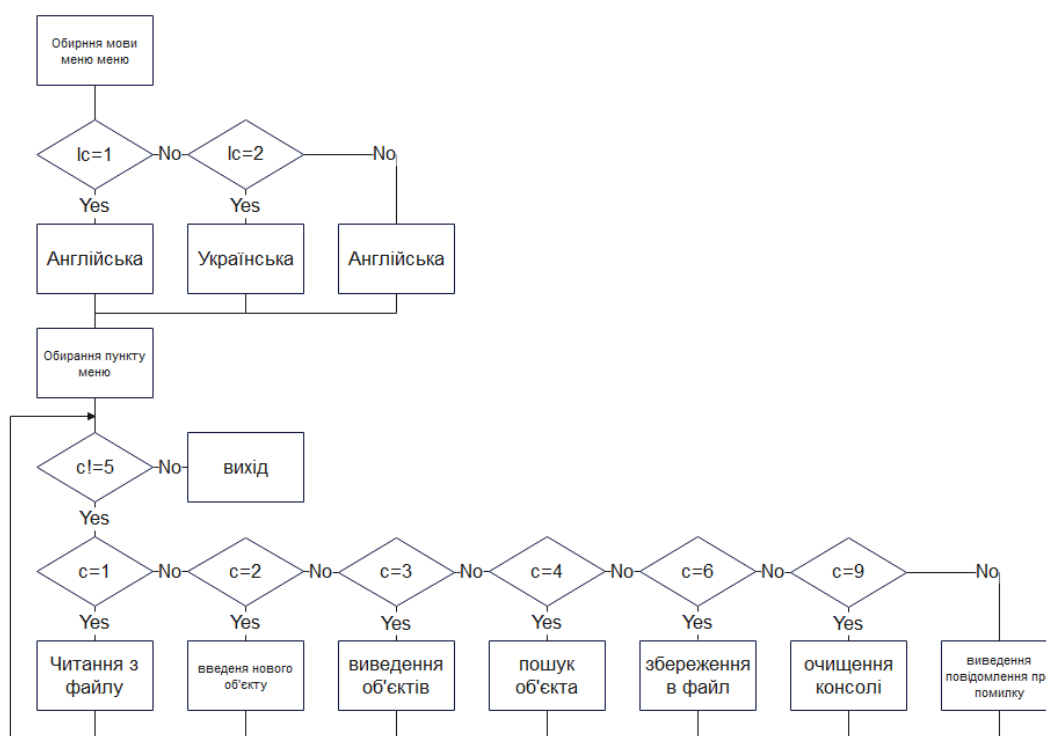


Рисунок 2.8 – Функціональна схема меню програми

2.4 Опис фізичної структури програми

Відповідно постановки задачі, діаграми класів та логічної структури розробленого головно меню програми було складено визначення методів класів: TreeType, Tree, TreeFactory, Forest і тестова програма можливостей класу Forest .

Код визначення класів та їх методів знаходяться у файлах ресурсів (*.cpp). Дерево файлів проекту наведені на рис.2.3. Скомпільований файл тестування визначених класів знаходиться у файлі Project3.exe.

Код програми складений на мові програмування C++ та складається з визначення абстрактних типів даних та функцій-членів класу. Для повноцінного функціонування методів класів та тестової програми необхідно підключення допоміжних бібліотечних файлів за допомогою директиви препроцесора #include, тобто бібліотек визначення типів, констант, вбудованих функцій і тд, які забезпечують:

- #include <iostream> - для введення та виведення стандартних потоків вводу-виводу
- #include <iomanip> - форматування виведених даних
- #include <map> - реалізує асоціативний контейнер
- #include <fstream> - дозволяє взаємодіяти з файлами
- #include <string> - надає рядкові типи даних та функції для їх опрацювання
- #include <vector> - реалізує динамічний масив
- #include <algorithm> - містить реалізації різних алгоритмів, таких як сортування, пошук тощо
- #include <conio.h> - форматування виведених даних
- #include <locale> - надає можливість роботи з різними мовними локалями
- #include <sstream> - надає можливість використання строкових потоків
- #include <Windows.h> - використовується для роботи з функціями, специфічними для операційної системи Windows, такими як SetConsoleCP та SetConsoleOutputCP

Змін	Арк	№ докум	Підпис	Дата

ФКЗЕ.12100П00.КРПЗ

Арк
18

– #include <memory> - містить різні типи смарт-вказівників, такі як unique_ptr та shared_ptr, які допомагають у керуванні пам'яттю та уникненні витоку пам'яті

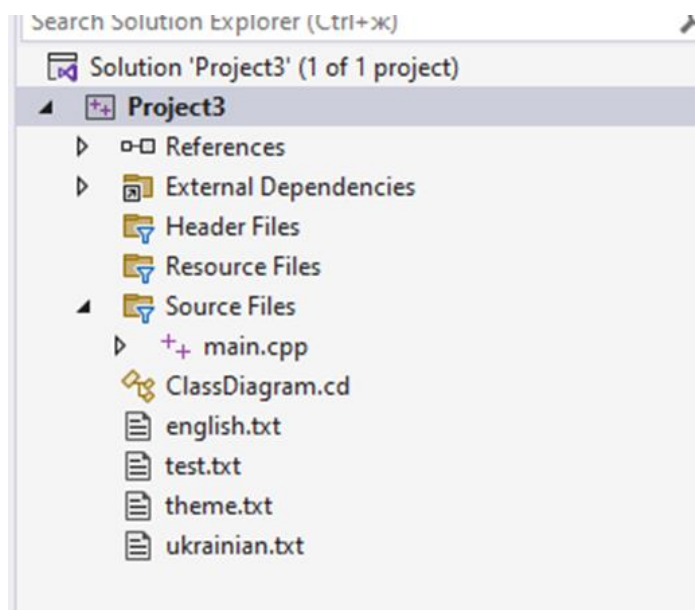


Рисунок 2.9 – Дерево файлів проекту

Після підключення бібліотечних файлів, аби спростити доступ до функцій стандартної бібліотеки STL, визначено використання простору імен глобально через директиву using namespace std;. В файлах проекту виконано оголошення та визначення класів TreeType, Tree, та TreeFactory, що є основою для управління інформацією про дерева. Опис методів та їх призначення розроблені відповідно до алгоритму виконання, який детально представлений у додатках програми.

2.4.1 Опис методів абстрактного класу TreeType

Клас TreeType використовується як базовий клас для представлення загальних характеристик дерева. В файлі TreeType.cpp знаходиться визначення конструкторів, які ініціалізують основні дані-члени класу, такі як тип, колір, текстура, висота та ширина. у табл. 2.1.

Таблиця 2.1 – Конструкторы класса TreeType

Заголовок, опис результату методу	Роль змінної	Тип	Позначення	Опис
Tree()	параметр	void	-	-
Tree(int objectId, string type, string colour, string texture, double height, double width)	Параметр	int	objectId	ID об'єкту
	Параметр	string	type	тип
	Параметр	string	colour	колір
	Параметр	string	texture	текстура
	Параметр	double	height	Висота
	Параметр	double	width	Ширина

У методах встановлення значень даних членів класу TreeType, параметр метода привласнюється відповідній властивості (даному члену) класу. Опис методів представлений у табл. 2.2.

Таблиця 2.2 – Методы установки значений атрибутов класса TreeType

Заголовок, опис результату методу	Роль змінної	Тип	Позначення	Опис
void addElement(TreeType* element)	параметр	TreeType	element	Додавання нового елемента у контейнер

Методи отримання значення властивостей класу TreeType, за допомогою оператора return, як результат роботи повертають значення відповідної властивості класу. Опис методів представлений у табл. 2.3

Таблиця 2.3 - Методы получения значений атрибутов класса TreeType

Заголовок, опис результату методу	Роль змінної	Тип	Позначення	Опис
int getObjectId()	параметр	int	-	Отримання ID

Продовження таблиці 2.3

const string& getTexture()	Змінна	const string	-	Отримання текстури
----------------------------	--------	--------------	---	--------------------

double getWidth()	параметр	double	-	Отримання Ширини
double getHeight()	параметр	double	-	Отримання висоти
const string getId()	параметр	const string	-	Отримання типу
const string& getColour()	параметр	const string	-	Отримання кольору

2.4.2 Опис методів абстрактного класу Tree

Клас TreeType використовується як базовий клас для представлення загальних характеристик дерева. В файлі Tree.cpp знаходиться визначення конструкторів, які ініціалізують основні дані-члени класу, такі як тип, колір, текстура, висота та ширина. у табл. 2.4.

Таблица 2.4 – Конструкторы класса Tree

Заголовок, опис результату методу	Роль змінної	Тип	Позначення	Опис
TreeType()	параметр	void	-	стандартний конструктор класу
TreeType(int objectId, string type, string colour, string texture, double height, double width)	Змінна	int	objectId	ID об'єкту
	Параметр	string	type	тип
	Параметр	string	colour	колір
	Параметр	string	texture	текстура
	Параметр	double	height	Висота
	Параметр	double	width	Ширина

У методах встановлення значень даних членів класу Tree, параметр метода привласнюється відповідній властивості (даному члену) класу. Опис методів представлений у табл. 2.5.

Таблица 2.5 – Методы установки значений атрибутов класса Tree

Заголовок, опис результату	Роль	Тип	Позначення	Опис
----------------------------	------	-----	------------	------

методу	змінної			
void setObjectId(int objectId)	змінна	int	objectId	Задання ID
void setTypeId(string type)	параметр	string	type	Задання типу
void setColour(string colour)	параметр	string	colour	Задання кольору
void setTexture(string texture)	параметр	string	texture	Задання текстури
void setHeight(double height)	параметр	double	height	Задання висоти
void setWidth(double width)	параметр	double	width	Задання ширини

Методи отримання значення властивостей класу Tree, за допомогою оператора return, як результат роботи повертають значення відповідної властивості класу. Опис методів представлений у табл. 2.6.

Таблица 2.6 – Методы получения значений атрибутов класса Tree

Заголовок, опис результату методу	Роль змінної	Тип	Позначення	Опис
int getId()	змінна	int	-	Отримання ID об'єкту
const string getIdType()	параметр	const string	-	Отримання типу
const string& getColour()	параметр	const string	-	Отримання кольору
const string& getTexture()	параметр	const string	-	Отримання текстури
double getHeight()	параметр	double	-	Отримання висоти
double getWidth()	параметр	double	-	Отримання Ширини

2.4.3 Опис методів абстрактного класу TreeFactory

Клас TreeFactory використовується як базовий клас для представлення загальних характеристик дерева. В файлі TreeFactory.cpp знаходиться визначення конструкторів, які ініціалізують основні дані-члени класу, такі як тип, колір, текстура, висота та ширина. у табл. 2.7.

Таблица 2.7 – Конструкторы класса TreeFactory

Заголовок, опис результату методу	Роль змінної	Тип	Позначення	Опис
static TreeType* create-Tree(const string& Type, const string& Colour, const string& Texture, int Height, int Width)	параметр	string	type	тип
	змінна	string	colour	колір
	параметр	string	texture	текстура
	параметр	double	height	Висота
	параметр	double	width	Ширина

У методах встановлення значень даних членів класу TreeFactory, параметр метода привласнюється відповідній властивості (даному члену) класу. Опис методів представлений у табл. 2.8.

Таблиця 2.8 - Методи установки значень атрибутів класу TreeFactory

Заголовок, опис результату методу	Роль змінної	Тип	Позначення	Опис
static void saveToFile(const string& filename)	Параметр	const string	filename	Назва файлу
static TreeType* createTree(int Object_id, string Type, const string& Colour, const string& Texture, int Height, int Width)	Параметр	string	type	тип
	Параметр	string	colour	колір
	Параметр	string	texture	текстура
	Змінна	double	height	Висота
	Параметр	double	width	Ширина
	Параметр	string	type	тип
static void input-Tree(std::map<int, TreeType*>& tree, std::map<std::string, std::string>& translations)	Параметр	map	tree	Контейнер для додавання нового об'єкту
	Параметр	map	translations	переклад

Методи отримання значення властивостей класу TreeFactory, за допомогою оператора return, як результат роботи повертають значення відповідної властивості класу. Опис методів представлений у табл. 2.9.

Таблиця 2.9 – Методи получения значень атрибутів класу TreeFactory

Заголовок, опис результату	Роль	Тип	Позначення	Опис
----------------------------	------	-----	------------	------

методу	змінної			
static int calculateWidth(const std::string& str)	параметр	const string	str	Вхідна строка
static void printAll-Trees(std::map<std::string, std::string> translations)	параметр	map	translations	translations використовується для виведення відсортованих дерев з врахуванням мовних перекладів
static void sort-Trees(std::vector<TreeType*> & trees, const std::string& sortBy)	параметр	vector	trees	Сортування об'єктів
	параметр	const string	sortBy	Ву визначає атрибут, за яким проводиться сортування дерев.
static void printSorted-Trees(std::map<std::string, std::string> translations)	Змінна	map	translations	translations використовується для виведення відсортованих дерев з врахуванням мовних перекладів
void secondarySort(std::vector<TreeType*> & trees, const std::string& attribute, const std::string& preferredValue)	параметр	vector	trees	trees є вектором вказівників на об'єкти TreeType, які потрібно вторинно відсортувати.
	параметр	const string	attribute	attribute визначає атрибут для вторинного сортування
	параметр	const string	preferredValue	preferredValue визначає переважне значення атрибута для вторинного сортування.
static void readFromFile(const string& filename, int langChoice)	параметр	const string	filename	filename є ім'ям файлу, з якого читаються дані.
	параметр	int	langChoice	langChoice визначає вибір мови для читання даних.
static TreeType* findTree(int Object_id)	Змінна	int	Object_id	Object_id є ідентифікатором об'єкта, за яким проводиться пошук дерева в колекції.

2.4.4 Опис методів контейнерного класу Forest

На основі розроблених класів складаємо клас Forest по обробці масиву даних за допомогою контейнерного класу map. Визначення методів класу знаходиться у файлі Forest.cpp. Клас Forest складається з властивостей та методів які наведені у підрозділі 2.10.

Таблиця 2.10 – Методы контейнерного класса Forest

Заголовок, опис результату методу	Роль змінної	Тип	Позначення	Опис
Forest()	параметр	void	-	конструктор
TreeType* addTree(const string& type, const string& colour, const string& texture, double height, double width)	Параметр	string	type	тип
	Параметр	string	colour	колір
	Змінна	string	texture	текстура
	Параметр	double	height	Висота
	Параметр	double	width	Ширина
void printAllTrees()	Параметр	void	-	Вивід списку об'єктів

У головній функції int main() оголошуємо локальні змінні:

–Forest forest - Об'єкт forest представляє ліс або колекцію дерев. Ця змінна використовується для управління та маніпуляції з деревами в рамках програми.

–int langChoice - Змінна langChoice використовується для зберігання вибору мови користувача, що може впливати на локалізацію або мовні налаштування програми.

–std::map<std::string, std::string> translations - ловник translations зберігає переклади різних термінів або фраз, які використовуються у програмі для підтримки багатомовності.

–int choice - нтифікатор Object_id використовується для пошуку або ідентифікації конкретного дерева або об'єкта в колекції.

–string filename - Змінна filename використовується для зберігання імені файлу, з якого потрібно читати або в який потрібно записувати дані.

–int Object_id - Ідентифікатор Object_id використовується для пошуку або ідентифікації конкретного дерева або об'єкта в колекції.

–TreeType* tree - Вказівник tree на тип TreeType використовується для роботи з окремим деревом, що може включати його створення, модифікацію або вивід інформації.

–std::vector<std::string> lines - Вектор lines зберігає набір рядків або тексту, які можуть бути отримані з файлу або введені користувачем для подальшої обробки в програмі.

У головній функції main() оголошено змінну app, яка є екземпляром класу Application. Це використовується для тестування методів розробленого контейнерного класу. Для керування логікою програми використовується метод createUI, в рамках якого реалізована структура керування do...while та switch.

```
cout << translations["read_data"] << "\n";
cout << translations["add_item"] << "\n";
cout << translations["display_sorted"] << "\n";
cout << translations["object_search"] << "\n";
cout << translations["exit"] << "\n";
cout << translations["save_changes"] << "\n";
cout << translations["clear_console"] << "\n";
cout << translations["enter_choice"] << "\n";
cin >> choice;
```

2.5 Тестування програми

Запуск коду

Для запуску програми користувач повинен запустити файл Project1.exe

Початок роботи з програмою

Після запуску, користувач побачить консольне вікно (рис.2.10):

Змін	Арк	№ докум	Підпис	Дата

ФКЗЕ.121ООП00.КРПЗ

Арк
26

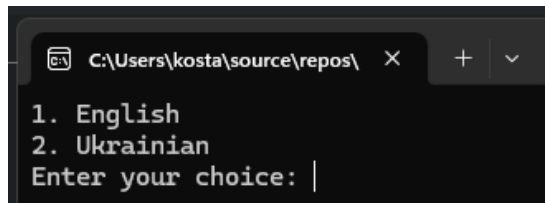


Рисунок 2.10 – Початкове меню вибору мови

В надане поле потрібно ввести число, що вказує на мову інтерфейсу, якою буде користуватися програмою, та натиснути Enter (рис.2.11)

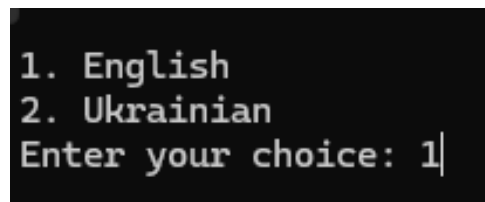


Рисунок 2.11 – Вибір мови

Робота в меню

Після вибору мови користувач потрапляє до головного меню (Рис.2.12)

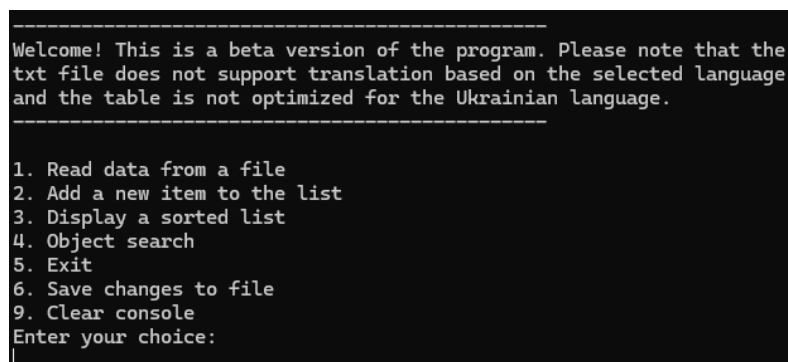


Рисунок 2.12 – Основне меню програми

Дане меню надає наступні функції:

- Прочитати дані з файлу;

- Додати новий елемент до списку;
- Відобразити відсортований список;
- Пошук об'єкта;
- Вихід;
- Зберегти зміни в файл;
- Очистити консоль;

Кожен з цих пунктів меню виконує свою функцію, описану нижче.

Вивід даних з файлу у табличному вигляді

Функція зчитування та відображення інформації про дерева з файлу у форматі табличного вигляду. Користувач може ввести ім'я файлу, з якого буде здійснено читання даних. Дані представлені у таблиці з такими колонками: ID об'єкта, тип, колір, текстура, висота, ширина (рис.2.13)

```
Enter choice: 1
```

Object ID	Figur	Colour	Texture	Height	Width
1	triangle	Black	Metallic	4	44
2	triangle	Black	Metallic	4	44
4	rectangle	Black	Metallic	4	44
5	rectangle	Blue	Metallic	44	44
6	circle	White	Matte	2	22
3	triangle	orange	Metallic	44	22

Рисунок 2.13 – Функція 1

Вивід відсортованої таблиці клієнтів

Ця опція дозволяє користувачу відсортувати та відобразити дані дерев у зчитаній таблиці за вказаним критерієм, таким як колір, тип або текстура. Вивід відсортованих даних здійснюється у табличному форматі з можливістю вибору критерію сортування (рис.2.14)


```
Enter choice: 2
```

Object ID	Figur	Colour	Texture	Height	Width
6	circle	White	Matte	2	22
4	rectangle	Black	Metallic	4	44
5	rectangle	Blue	Metallic	44	44
1	triangle	Black	Metallic	4	44
2	triangle	Black	Metallic	4	44
3	triangle	orange	Metallic	44	22

Рисунок 2.14 – Функція 3

Додати нового об'єкта

Функція дозволяє користувачу ввести дані для створення нового дерева. Дані включають тип, колір, текстуру, висоту та ширину дерева. Вводяться дані через консоль з валідацією введення. У разі помилки введення, програма запропонує ввести дані ще раз (рис.2.15-16)

```
Enter your choice:
2
Select Colour
1. Red 2. Green 3. Blue 4. Black 5. White 6. Gray 7. Purple 8. orange 9. Pink 10. Brown
3
Select Texture
1. Smooth 2. Rough 3. Matte 4. Glossy 5. Metallic
2
Enter Height: 44
Enter Width: 44
select_figure
1. circle 2. rectangle 3. triangle
2
```

```
Enter choice: 2
```

Object ID	Figur	Colour	Texture	Height	Width
6	circle	White	Matte	2	22
4	rectangle	Black	Metallic	4	44
5	rectangle	Blue	Metallic	44	44
7	rectangle	Blue	Rough	44	44
1	triangle	Black	Metallic	4	44
2	triangle	Black	Metallic	4	44
3	triangle	orange	Metallic	44	22

Рисунки 2.15-16 – Функція 2

Пошук по ID

Цей пункт допомагає зробити пошук по ID(рис.2.17)

```
Enter Object ID: 5
Object found
Object ID      |Figur      |Colour    |Texture    |Height    |Width
-----|-----|-----|-----|-----|-----
5              |rectangle  |Blue      |Metallic   |44        |44
```

Рисунок 2.17 – Функція 4

Вихід з програми

Цей пункт виходить з програми (рис 2.18)

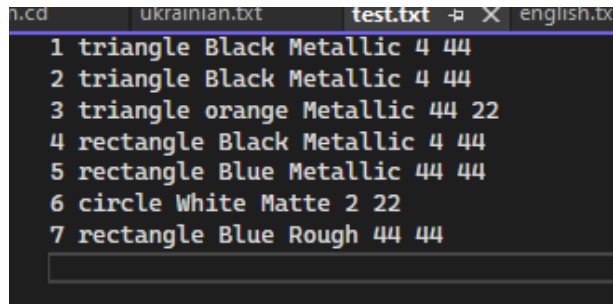
```
1. Read data from a file
2. Add a new item to the list
3. Display a sorted list
4. Object search
5. Exit
6. Save changes to file
9. Clear console
Enter your choice:
5
```

Рисунок 2.18 – Функція 5

Збереження нової інформації

Цей пункт дає змогу зберегти те, що було додано в ваш файл (рис.2.19-20)

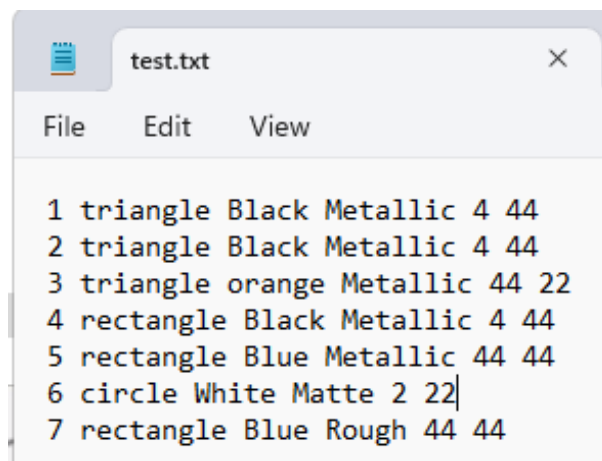
```
1. Read data from a file
2. Add a new item to the list
3. Display a sorted list
4. Object search
5. Exit
6. Save changes to file
9. Clear console
Enter your choice:
6
```



```
n.cd    ukrainian.txt    test.txt  + X  english.txt
1 triangle Black Metallic 4 44
2 triangle Black Metallic 4 44
3 triangle orange Metallic 44 22
4 rectangle Black Metallic 4 44
5 rectangle Blue Metallic 44 44
6 circle White Matte 2 22
7 rectangle Blue Rough 44 44
```

Рисунки 2.19-20 – Функція 6

Зовнішній файл повинен містити в собі ID, назву фігури, колір, текстуру, висоту та ширину як наведено на рис. 2.21



```
test.txt
File Edit View
1 triangle Black Metallic 4 44
2 triangle Black Metallic 4 44
3 triangle orange Metallic 44 22
4 rectangle Black Metallic 4 44
5 rectangle Blue Metallic 44 44
6 circle White Matte 2 22
7 rectangle Blue Rough 44 44
```

Рисунок 2.21 – Зовнішній файл

2.6 Вимоги до складу та параметрів технічних та програмних засобів

Розроблена програма за завданням курсової роботи призначена для запуску на операційній системі Windows та вимагає мінімальних додаткових програмних і апаратних засобів такі як:

- операційна система: Windows 10 або 11;

- процесор: 2 GHz або вище;
- оперативна пам'ять: мінімум 8 GB ОЗУ;
- місце на диску: мінімум 3 GB;
- відеокарта: NVIDIA GeForce GTX 660 з 2000 MB відеопам'яті.

Тип монітора, миші та клавіатури не є обов'язковими для компіляції коду. Ви можете використовувати будь-який із цих пристроїв, підключених до вашого комп'ютера. Але зазвичай рекомендується мати монітор з діагоналлю не менше 13-15 дюймів для комфортної роботи з інтерфейсом редактора коду та відображенням консольних вікон.

Якщо ви плануєте виконувати цей код на іншій платформі або в іншому середовищі розроблення, наприклад, у Linux або macOS, вам, можливо, знадобиться відповідним чином налаштувати середовище і внести зміни до коду з урахуванням можливостей цієї платформи, оскільки програма використовує функції бібліотеки <windows.h>.

2.7 Інструкція для користувача

На рисунку 2.22 обирається мова, якою буде працювати програма (якщо обрати інше число, то програма не буде працювати).

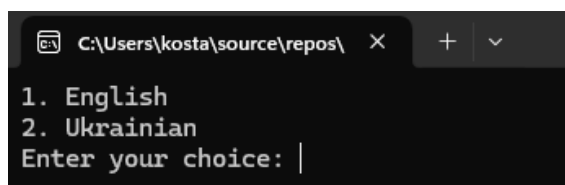


Рисунок 2.22 - Вибір мови

На рисунку 2.23 обирається пункт 1 для підключення файлу. Також на рисунку 2.24 можна побачити, що буде, якщо обрати неіснуючий файл (програма не буде зберігати та створювати файл)

```

-----
Welcome! This is a beta version of the program. Please note that the
txt file does not support translation based on the selected language
and the table is not optimized for the Ukrainian language.
-----

1. Read data from a file
2. Add a new item to the list
3. Display a sorted list
4. Object search
5. Exit
6. Save changes to file
9. Clear console
Enter your choice:
1
Enter a file name: test.txt

```

Рисунок 2.23 - Коректно обраний файл

```

1
Enter a file name: ыцц

-----
Welcome! This is a beta version of the program. Please note that the
txt file does not support translation based on the selected language
and the table is not optimized for the Ukrainian language.
-----

1. Read data from a file
2. Add a new item to the list
3. Display a sorted list
4. Object search
5. Exit
6. Save changes to file
9. Clear console
Enter your choice:
|

```

Рисунок 2.24 - Некоректно обраний файл

На рисунку 2.25 обирається пункт 2, де додаються предмети до списку. На рис. 2.25 можна побачити кольори, які можна обрати (10 кольорів). На рис. 2.26 можна побачити вибір текстури. На рис. 2.27 можна додати висоту та ширину. На рис 2.28 можна обрати фігури. На рис. 2.29 можна побачити, що буде, якщо вве-

сти некоректні дані (якщо обрати невірну відповідь, то програма видасть повідомлення про невірну відповідь і запропонує зробити вибір з запропонованих чисел або ввести тільки цифри).

```
Enter your choice:
2
Select Colour
1. Red 2. Green 3. Blue 4. Black 5. White 6. Gray 7. Purple 8. orange 9. Pink 10. Brown
|
```

Рисунок 2.25 - Вибір кольору

```
2
Select Texture
1. Smooth 2. Rough 3. Matte 4. Glossy 5. Metallic
|
```

Рисунок 2.26 - Обираємо текстуру

```
5
Enter Height: 44
Enter Width: 55
|
```

Рисунок 2.27 - Обираємо ширину та висоту

```
Enter Width: 55
select_type
1. circle 2. rectangle 3. triangle
|
```

Рисунок 2.28 - Обираємо фігуру

```

Enter your choice:
2
Select Colour
1. Red 2. Green 3. Blue 4. Black 5. White 6. Gray 7. Purple 8. orange 9. Pink 10. Brown
11
Invalid input. Please enter a number between 1 and 10.
Select Colour
1. Red 2. Green 3. Blue 4. Black 5. White 6. Gray 7. Purple 8. orange 9. Pink 10. Brown
8
Select Texture
1. Smooth 2. Rough 3. Matte 4. Glossy 5. Metallic
6
Invalid input. Please enter a number between 1 and 5.
Select Texture
1. Smooth 2. Rough 3. Matte 4. Glossy 5. Metallic
5
Enter Height: swe
Invalid input. Please enter a number: 44
Enter Width: ffr
Invalid input. Please enter a number: 22
select_type
1. circle 2. rectangle 3. triangle
5
Invalid input. Please enter a number between 1 and 3.
select_type
1. circle 2. rectangle 3. triangle
3

```

Рисунок 2.29 - обір некоректних чисел

На рисунку 2.30 обирається пункт 3, де можна зробити сортування за кольором, фігурою та текстурою від А-Я, А-Z (якщо обрати невірну цифру, то програма буде сортувати за замовчуванням за ID).

```

1. Read data from a file
2. Add a new item to the list
3. Display a sorted list
4. Object search
5. Exit
6. Save changes to file
9. Clear console
Enter your choice:
3
Select sort option:
1. Colour
2. Type
3. Texture
Enter choice: |

```

Рисунок 2.30 - Вибираємо пункт сортування за типом

```

Enter choice: 4
Invalid choice, sorting by Type by default.

```

Змін	Арк	№ докум	Підпис	Дата

ФКЗЕ.121ООП00.КРПЗ

Арк

35

Рисунок 2.31 - Ми обираємо некоректну відповідь

5. Texture
Enter choice: 1

Object ID	Figur	Colour	Texture	Height	Width
1	triangle	Black	Metallic	4	44
2	triangle	Black	Metallic	4	44
4	rectangle	Black	Metallic	4	44
5	rectangle	Blue	Metallic	44	44
6	circle	White	Matte	2	22
3	triangle	orange	Metallic	44	22

Рисунок 2.32 - Вибираємо сортування за кольором

На рисунку 2.33 обирається пункт 4 - пошук за ID об'єкта (якщо введено некоректний ID або букву, то програма нічого не знайде та пошук буде відмінено. Якщо введене ID буде у файлі, то пошук виведе його).

Object found

Object ID	Figur	Colour	Texture	Height	Width
3	triangle	orange	Metallic	44	22

Welcome! This is a beta version of the program. Please note that the txt file does not support translation based on the selected language and the table is not optimized for the Ukrainian language.

1. Read data from a file
2. Add a new item to the list
3. Display a sorted list
4. Object search
5. Exit
6. Save changes to file
9. Clear console

Enter your choice:
4
Enter Object ID: s
Object not found

Рисунок 2.34 - Коректний та не коректний пошук по ID

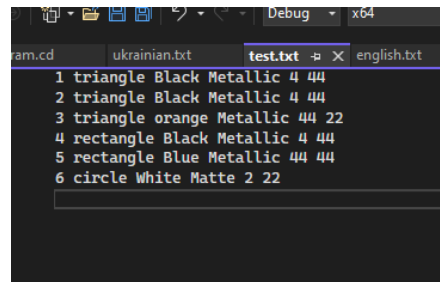
На рисунку 2.35 обирається операція 6, де зберігаються всі дії з файлом. На рисунку 2.36 можна побачити, що вони були збережені у файл (якщо файлу немає, то дані не будуть збережені).


```

1. Read data from a file
2. Add a new item to the list
3. Display a sorted list
4. Object search
5. Exit
6. Save changes to file
9. Clear console
Enter your choice:
6

```

Рисунок 2.35 - Вибір функції 6



```

1 triangle Black Metallic 4 44
2 triangle Black Metallic 4 44
3 triangle orange Metallic 44 22
4 rectangle Black Metallic 4 44
5 rectangle Blue Metallic 44 44
6 circle White Matte 2 22

```

Рисунок 2.36 - збереження наших дій

На рисунку 2.37 показані дії 9 та 5. Дія 9 очищує консоль від усього тексту в ній, дія 5 виходить з програми (якщо обрати невірну цифру, то програма не виконає ніякої дії).

```

-----
Welcome! This is a beta version of the program. Please note that the
txt file does not support translation based on the selected language
and the table is not optimized for the Ukrainian language.
-----

1. Read data from a file
2. Add a new item to the list
3. Display a sorted list
4. Object search
5. Exit
6. Save changes to file
9. Clear console
Enter your choice:

```

Рисунок 2.37 - Дія 9 та 5

2.8 Результати реалізації програми

Поставлені цілі:

Створення швидкої, простої в використанні та ефективної програми для обробки даних про дерева.

Можливість легкої інтеграції з мультилінгвальними системами для забезпечення використання у різних культурних контекстах.

Забезпечення можливості сортування та пошуку даних за різними параметрами для зручності користувачів.

Досягнуті цілі:

Реалізована програма забезпечує базові функціональності по введенню, сортуванню, пошуку та збереженню даних про дерева, але може бути не зовсім простою для використання через велику кількість параметрів вводу.

Інколи в програмі спостерігається неефективне використання оперативної пам'яті, особливо при обробці великої кількості даних, що може призвести до зниження швидкості виконання.

Рекомендації до вдосконалення:

Оптимізувати алгоритми сортування та пошуку для підвищення продуктивності, особливо при великому обсязі даних.

Покращити інтерфейс користувача, зробивши його більш інтуїтивно зрозумілим і зручним для користувачів без технічного фону.

Розробити додаткові опції для керування пам'яттю, зокрема за допомогою використання більш ефективних структур даних або методів кешування.

ВИСНОВКИ ПО РОБОТІ

У цій курсовій роботі була розроблена система управління даними про дерева за допомогою ієрархії класів Tree, TreeType, і TreeFactory. Головною метою було створення структурованої системи, яка забезпечує ефективне управління, пошук, сортування та зберігання інформації про дерева в різноманітних атрибутах, таких як колір, тип, текстура, висота і ширина.

Протягом виконання роботи було активно застосовано принципи об'єктно-орієнтованого програмування. Було реалізовано наслідування та поліморфізм через базовий клас TreeType та його нащадки, що дозволило забезпечити гнучкість та розширюваність системи. Також велика увага приділялася роботі з файлами і мультязичною підтримкою, що реалізувалося через зчитування та запис у файлові потоки, а також використання структури даних `std::map` для зберігання перекладів інтерфейсу.

В ході роботи було демонстровано ефективне використання контейнерів STL, таких як `std::vector` і `std::map`, для управління колекціями об'єктів. Застосування функцій стандартної бібліотеки, таких як `std::sort`, для сортування даних за різними критеріями, забезпечило високу продуктивність при обробці даних.

Розроблена програма демонструє глибоке розуміння теми та технічних аспектів роботи з даними і об'єктно-орієнтованим програмуванням, має значний потенціал для подальшої адаптації та використання в реальних додатках, зокрема в галузях лісового господарства та ботаніки. Ця система може бути розширена з додаванням нових функцій та оптимізації для специфічних завдань управління даними про дерева

Змін	Арк	№ докум	Підпис	Дата

ФКЗЕ.121ООП00.КРПЗ

Арк
39

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

- 1 Клас map. Microsoft Learn: Build skills that open doors in your career.
URL: <https://learn.microsoft.com/ru-ru/cpp/standard-library/map-class?view=msvc-170> (дата звернення: 14.05.2024).
- 2 Патерни/шаблони проектування. Refactoring and Design Patterns.
URL: <https://refactoring.guru/uk/design-patterns> (дата звернення: 09.05.2024).
- 3 C++ Standard Library including C++ 14 & C++ 17 - AI-Powered Learning for Developers. Educative. URL: <https://www.educative.io/courses/cpp-standard-library-including-cpp-14-and-cpp-17> (date of access: 09.05.2024).
- 4 Shalloway A., Trott J. Design Patterns Explained: A New Perspective on Object-Oriented Design. 2nd ed. Addison-Wesley, 2004. P. 468.
- 5 Stroustrup B. Programming: Principles and Practice Using C++. 2nd ed. Addison-Wesley, 2014. P. 2339.
- 6 Schildt G. C++: A Beginner's Guide. McGrawHil, 2012. 541 p.

					ФКЗЕ.121ООП00.КРПЗ	Арк
						40
Змін	Арк	№ докум	Підпис	Дата		

ДОДАТОК А

БЛОК-СХЕМИ

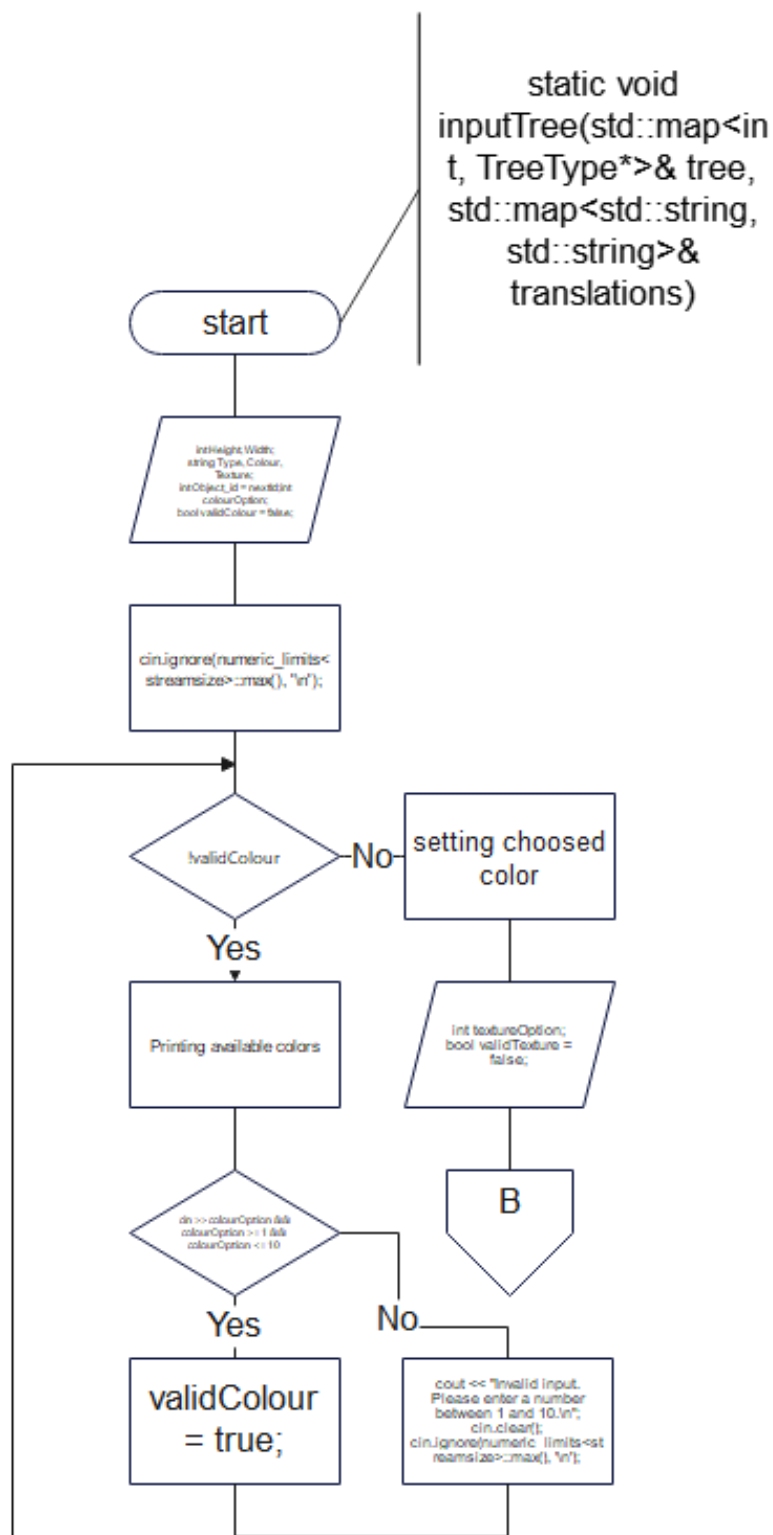
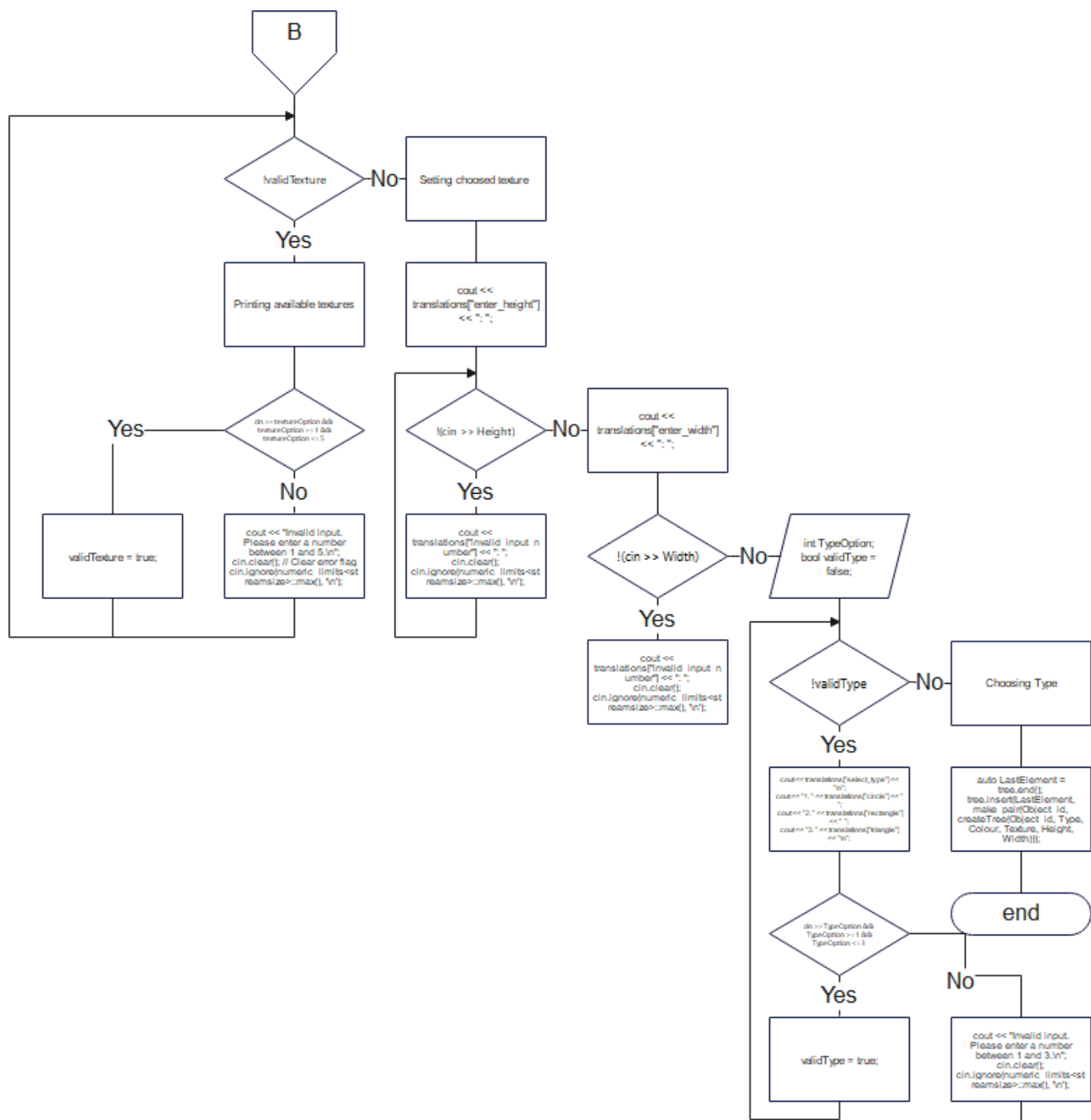


Рисунок А.1 – Функція inputTree()



Продовження рисунку А1

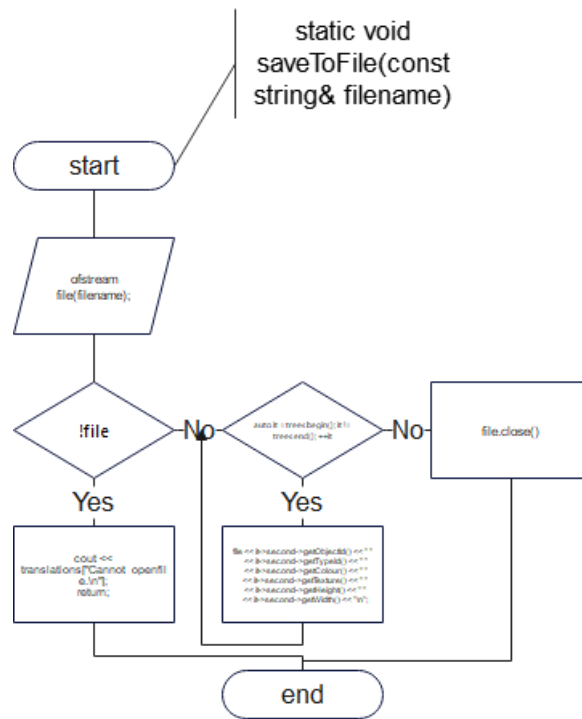


Рисунок А.2 – Функція saveToFile()

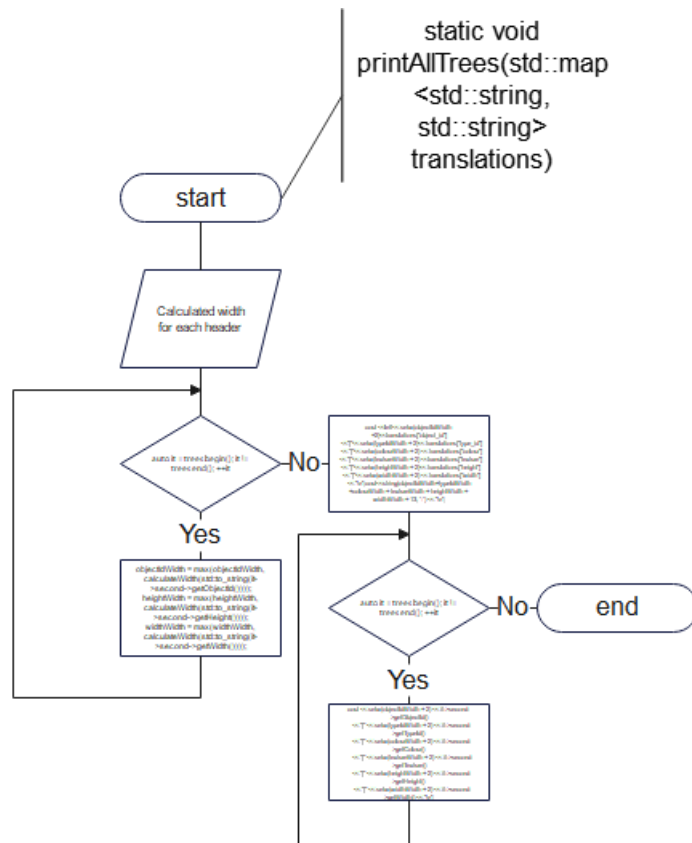


Рисунок А.3 – Функція printAllTrees()

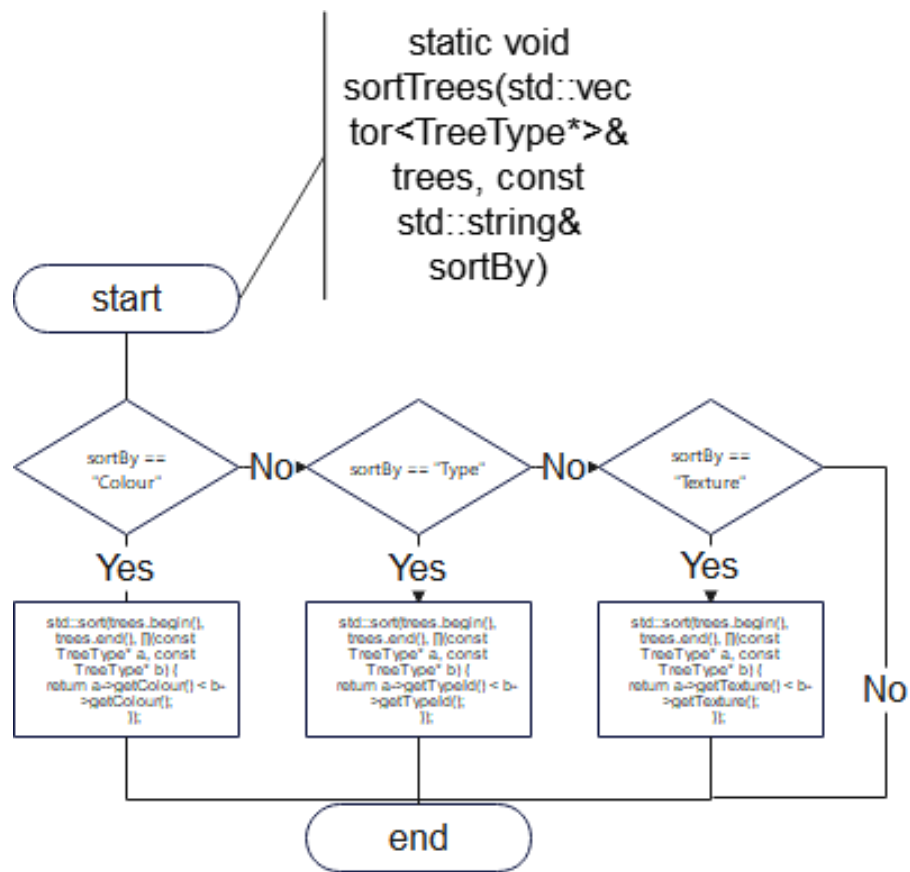


Рисунок А.5– Функція sortTrees()

ДОДАТОК Б

ЛІСТИНГ ПРОГРАМИ

```
#include <windows.h>
#include <iomanip>
#include <iostream>
#include <fstream>
#include <string>
#include <map>
#include <conio.h>
using namespace std;

struct Date {
    int day;
    int month;
    int year;
};

class Device {
protected:
    string Name;
    string Producer;
    string model;
    Date date;
    int volume;
public:
    Device() : Name("Empty"), Producer("Empty"), model("Empty"), date{ 0,0,0 },
    volume(0) {};
    Device(string Name, string Producer, string model, int day, int month, int year,
    int volume)
        : Name(Name), Producer(Producer), model(model), date{ day, month, year },
    volume(volume) {};
    virtual void show() const = 0;
    virtual void set() = 0;
    string GetName() { return Name; }
    string Getmodel() { return model; }

    void GetAllstr(string& Name, string& Producer, string& model) {
        Name = this->Name, Producer = this->Producer, model = this->model;
    }
    void GetAllint(int& day, int& month, int& year, int& volume) {
        day = date.day, month = date.month, year = date.year, volume = this->volume;
    }

    void setVolume(int num) { volume = num; }
    void LockVolume(int num) { volume *= num; }
};

string Func(string text, int n, int type) {
    n -= text.size();
    string rez;
    for (int i = 0; i <= n - 2; i++) rez = rez + ' ';
    if (type == 1) rez += " | ";
    if (type == 2) rez += " |";
    return rez;
}

class TV : public Device {
public:
```

```

TV(string Name, string Producer, string model, int day, int month, int year, int
volume)
    : Device(Name, Producer, model, day, month, year, volume) {}
virtual void show() const {
    string space;
    if (Name.size() > 10) {
        cout << Name.substr(0, 10);
        cout << "...";
        space = Func(Name.substr(0, 10), 10, 1);
    }
    else {
        cout << Name;
        space = Func(Name, 14, 1);
    }
    cout << space;
    if (Producer.size() > 10) {
        cout << Producer.substr(0, 10);
        cout << "...";
        space = Func(Producer.substr(0, 10), 11, 1);
    }
    else {
        cout << Producer;
        space = Func(Producer, 14, 1);
    }
    cout << space;
    cout << model;
    space = Func(model, 6, 1);
    cout << space;

    string text;
    text = text + to_string(date.day) + "." + to_string(date.month) + "." +
to_string(date.year);
    space = Func(text, 11, 1);
    cout << text << space;

    text = "";
    if (volume > 0) text = text + "Sound is available ( ";
    else text = text + "No sound ( ";
    text = text + to_string(volume) + " )";
    space = Func(text, 26, 2);
    cout << text << space << endl;
}
virtual void set() {
    cout << "Enter Name: "; cin >> Name;
    cout << "Enter Producer: "; cin >> Producer;
    cout << "Enter model: "; cin >> model;
    cout << "Enter date (day, month, year): "; cin >> date.day >> date.month >>
date.year;
    cout << "Enter volume: "; cin >> volume;
}
};

class Radio : public Device {
public:
    Radio(string Name, string Producer, string model, int day, int month, int year, int
volume)
        : Device(Name, Producer, model, day, month, year, volume) {}
    virtual void show() const {
        string space;
        if (Name.size() > 10) {
            cout << Name.substr(0, 10);
            cout << "...";

```

```

        space = Func(Name.substr(0, 10), 10, 1);
    }
    else {
        cout << Name;
        space = Func(Name, 14, 1);
    }
    cout << space;
    if (Producer.size() > 10) {
        cout << Producer.substr(0, 10);
        cout << "...";
        space = Func(Producer.substr(0, 10), 11, 1);
    }
    else {
        cout << Producer;
        space = Func(Producer, 14, 1);
    }
    cout << space;
    cout << model;
    space = Func(model, 6, 1);
    cout << space;

    string text;
    text = text + to_string(date.day) + "." + to_string(date.month) + "." +
to_string(date.year);
    space = Func(text, 11, 1);
    cout << text << space;

    text = "";
    if (volume > 0) text = text + "Sound is available ( ";
    else text = text + "No sound ( ";
    text = text + to_string(volume) + " )";
    space = Func(text, 26, 2);
    cout << text << space << endl;
}
virtual void set() {
    cout << "Enter Name: "; cin >> Name;
    cout << "Enter Producer: "; cin >> Producer;
    cout << "Enter model: "; cin >> model;
    cout << "Enter date (day, month, year): "; cin >> date.day >> date.month >>
date.year;
    cout << "Enter volume: "; cin >> volume;
}
};

class Remote {
protected:
    TV* obj_1 = NULL;
    Radio* obj_2 = NULL;
public:
    Remote(TV& obj) : obj_1(&obj) {}
    Remote(Radio& obj) : obj_2(&obj) {}
    void ChangeVolume() {
        int num = -20;
        while (num <= 0 || num > 100) {
            cout << "Enter new number of volume(1-100): "; cin >> num;
        }
        if (obj_1 != NULL) obj_1->setVolume(num);
        else obj_2->setVolume(num);
    }
};

class AdvacedRemote : public Remote {

```

```

public:
    AdvacedRemote(TV& obj) : Remote(obj) {};
    AdvacedRemote(Radio& obj) : Remote(obj) {};
    void Un_LockVolume() {
        if (obj_1 != NULL)
            obj_1->LockVolume(-1);
        else
            obj_2->LockVolume(-1);
        cout << "Un_LockVolume Successfully!!!" << endl << endl;
    }
};

class Devices {
private:
    map<string, pair<AdvacedRemote*, Device*>> dev;
    map<string, pair<AdvacedRemote*, Device*>, greater<>> tmp_dev;
    map<string, pair<AdvacedRemote*, Device*>>::iterator iter;
    map<string, pair<AdvacedRemote*, Device*>>::iterator iter_end;
    int typeRemote;
    string filename = "save.txt";
public:
    void print();
    void save();
    void load();
    void sorting();
    void add(string, string, string, int, int, int, int, int);
    void edit();
    void del();
    void find();
};

//ok and check
void Devices::print() {
    if (dev.empty() && tmp_dev.empty()) { cout << "Контейнер порожній\n"; return; }

    if (tmp_dev.empty()) {
        iter = dev.begin();
        iter_end = dev.end();
    }
    else {
        iter = tmp_dev.begin();
        iter_end = tmp_dev.end();
    }

    int i = 0;
    cout << " |-----|
-----| " << endl;
    cout << " | N |          Name          |      Producer      |   Type   |      Date      |
Volume | " << endl;
    for (iter; iter != iter_end; i++) {
        cout << " |---|-----|-----|-----|-----|
-----| ";
        cout << "\n| " << i + 1 << " |   ";
        iter->second->show(); iter++;
    }
    cout << " |-----|
-----| " << endl << endl;
}

//ok and check
void Devices::save() {
    if (dev.empty() && tmp_dev.empty()) { cout << "Контейнер порожній\n"; return; }
    if (tmp_dev.empty()) {
        iter = dev.begin();

```

```

        iter_end = dev.end();
    }
    else {
        iter = tmp_dev.begin();
        iter_end = tmp_dev.end();
    }
    int num;
    cout << "Куди ви хочете зберегти: \n \t0) В свій файл \n \t1) Авто\nВведіть цифру: "; cin >> num;
    if (num == 0) {
        cout << "Введіть назву файлу(без вказання типу файлу): "; cin >> filename;
        filename += ".txt";
    }
    ofstream Myfile(filename);
    Myfile << ' ';
    for (iter; iter != iter_end; iter++) {
        string Name, Producer, model;
        int day, month, year, volume;
        iter->second.second->GetAllstr(Name, Producer, model);
        iter->second.second->GetAllint(day, month, year, volume);
        Myfile << Name << ';' << Producer << ';' << model
            << ';' << day << ' ' << month << ' ' << year << ' ' << volume << ' '
    << typeRemote << endl;
    }

}
//ok and check
void Devices::load() {
    if (tmp_dev.empty()) {
        iter = dev.begin();
        iter_end = dev.end();
    }
    else {
        iter = tmp_dev.begin();
        iter_end = tmp_dev.end();
    }
    int num;
    cout << "Звідки ви хочете скопіювати: \n \t0) Зі свого файлу \n \t1) Авто\nВведіть цифру: "; cin >> num;
    if (num == 0) {
        cout << "Введіть назву файлу(без вказання типу файлу): "; cin >> filename;
        filename += ".txt";
    }
    ifstream Myfile(filename);
    string Name, Producer, model;
    int day, month, year, volume;
    if (Myfile.is_open()) {
        cout << "true" << endl;
        while (!Myfile.eof()) {
            Myfile.ignore();
            getline(Myfile, Name, ';');
            getline(Myfile, Producer, ';');
            getline(Myfile, model, ';');
            Myfile >> day >> month >> year >> volume >> typeRemote;
            add(Name, Producer, model, day, month, year, volume, typeRemote);
        }
    }
    else cout << "Файлу не існує!" << endl;
    filename = "save.txt";
}
//ok and check
void Devices::sorting() {

```

```

    if (dev.empty() && tmp_dev.empty()) {
        cout << "Контейнер порожній\n";
        return;
    }
    int num;
    cout << "За алфавітом (0) / за спаданням (1): "; cin >> num;
    if (num == 0) {
        if (tmp_dev.empty()) return;
        for (iter = tmp_dev.begin(); iter != tmp_dev.end(); iter++)
            dev.insert(*iter);
        tmp_dev.clear();
        cout << "Сортування успішне!" << endl;
    }
    if (num == 1) {
        if (dev.empty()) return;
        for (iter = dev.begin(); iter != dev.end(); iter++)
            tmp_dev.insert(*iter);
        dev.clear();
        cout << "Сортування успішне!" << endl;
    }
}

bool check_date(int day, int month, int year) {
    // Масив, що містить кількість днів у місяцях
    int days_in_month[] = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };

    // Перевіряємо чи введений місяць є в діапазоні
    if (month < 1 || month > 12)
        return false;

    // Перевіряємо чи введений день не перевищує кількість днів у місяці
    if (day < 1 || day > days_in_month[month - 1])
        return false;

    // Якщо високосний рік і лютий, перевіряємо чи день не більше 29
    if (month == 2 && day == 29 && (year % 4 == 0 && (year % 100 != 0 || year % 400 ==
0)))
        return true;

    return true;
}
//ok and check
void Devices::add(string Name = "0", string Producer = "0", string model = "0",
    int day = 0, int month = 0, int year = 0, int volume = 0, int type = 0) {

    if (Name == "0") {
        int N;
        cin.ignore();
        cout << "Введіть назву пристрою: "; getline(cin, Name, ';');
        cin.ignore();
        cout << "Введіть виробника: "; getline(cin, Producer, ';');
        cin.ignore();
    ModuleAgain:
        cout << "Виберіть модель: \n\t1) TV \n\t2) Radio \n<: "; cin >> N;
        cin.ignore();
        if (N == 1) model = "TV";
        else if (N == 2) model = "Radio";
        else goto ModuleAgain;
    DateAgain:
        cout << "Введіть дату (день, місяць, рік): "; cin >> day >> month >> year;
        cin.ignore();
        if (check_date(day, month, year) == 0) goto DateAgain;

```

```

        do {
            cout << "Введіть гучність (1 - 100): "; cin >> volume;
            cin.ignore();
        } while (volume <= 0 || volume > 100);
        do {
            cout << "Введіть тип пульта керування: Remote(0) або AdvacedRemote(1)
- "; cin >> typeRemote;
            cin.ignore();
        } while (typeRemote != 0 && typeRemote != 1);
    }
    if (Name != "0") typeRemote = type;
    if (tmp_dev.empty()) {
        if (model == "TV") {
            TV* obj = new TV(Name, Producer, model, day, month, year, volume);
            AdvacedRemote* remote = new AdvacedRemote(*obj);
            dev.insert(make_pair(Name, make_pair(remote, obj)));
        }
        if (model == "Radio") {
            Radio* obj = new Radio(Name, Producer, model, day, month, year,
volume);
            AdvacedRemote* remote = new AdvacedRemote(*obj);
            dev.insert(make_pair(Name, make_pair(remote, obj)));
        }
    }
    else {
        if (model == "TV") {
            TV* obj = new TV(Name, Producer, model, day, month, year, volume);
            AdvacedRemote* remote = new AdvacedRemote(*obj);
            tmp_dev.insert(make_pair(Name, make_pair(remote, obj)));
        }
        if (model == "Radio") {
            Radio* obj = new Radio(Name, Producer, model, day, month, year,
volume);
            AdvacedRemote* remote = new AdvacedRemote(*obj);
            tmp_dev.insert(make_pair(Name, make_pair(remote, obj)));
        }
    }
}
//ok and check
void Devices::edit() {

    if (dev.empty() && tmp_dev.empty()) { cout << "Контейнер порожній\n"; return; }
    int num, N;

    cout << "Введіть num для редагування: "; cin >> num;
    if (tmp_dev.empty()) {
        if (num > dev.size()) { cout << "Значення перевищує розмір контейнеру\n";
return; }

        iter = dev.begin();
    }
    else {
        if (num > tmp_dev.size()) { cout << "Значення перевищує розмір контейнеру\n";
return; }

        iter = tmp_dev.begin();
    }
    advance(iter, num - 1);
    iter->second.second->show();
    cout << "Введіть N для редагування: якщо всі значення(0), гучність(1): "; cin >> N;
    if (N == 0) {

```



```

        iter->second.second->set();
    }
    else {
        if (N == 1 && typeRemote == 1) {
            cout << "Введіть N для редагування гучності: змінити(0), виключити(1):";
            cin >> N;

            if (N == 0) {
                iter->second.first->ChangeVolume();
            }
            else iter->second.first->Un_LockVolume();
        }
        else if(N == 1 && typeRemote != 1) iter->second.first->ChangeVolume();
    }
}

//ok and check
void Devices::del() {
    if (dev.empty() && tmp_dev.empty()) { cout << "Контейнер порожній\n"; return; }
    int num;

    cout << "Введіть num для видалення: "; cin >> num;
    if (tmp_dev.empty()) {
        if (num > dev.size()) { cout << "Значення перевищує розмір контейнеру\n";
return; }
        iter = dev.begin();
        advance(iter, num - 1);
        iter = dev.erase(iter);
    }
    else {
        if (num > tmp_dev.size()) { cout << "Значення перевищує розмір контейнеру\n";
return; }
        iter = tmp_dev.begin();
        advance(iter, num - 1);
        iter = tmp_dev.erase(iter);
    }
    cout << "Елемент видалено\n";
}

//ok and check
void Devices::find() {
    if (dev.empty() && tmp_dev.empty()) { cout << "Контейнер порожній\n"; return; }
    string Name;

    cout << "Введіть назву прилада: "; cin >> Name;

    if (tmp_dev.empty()) {
        iter = dev.begin();
        iter_end = dev.end();
    }
    else {
        iter = tmp_dev.begin();
        iter_end = tmp_dev.end();
    }
    for (iter; iter != iter_end; iter++) {
        if (iter->second.second->GetName().find(Name) != string::npos) {
            iter->second.second->show();
        }
    }
}

//ok and check
char Menu() {
    char c = 1;
    system("cls");
}

```

```

        cout << "Програма для обробки приладів та їхніх пультів керування:\n" << " TV, Ра-
діо, звич_пульт, спец_пульт\n";
        cout << "-----\n";
        cout << "1) Додати прилад \n2) Редагувати параметри \n3) Видалити прилад \n4) Вивід
на екран"
        << "\n5) Сортунання контейнеру \n6) Фільтрувати за критерієм \n7) Зберегти у
файл \n8) Завантажити з файлу \n0) Вихід\n";

        while ((c < '0' || c > '8') && c != 27) {
            c = _getch();
        }
        return c;
    }
    int main()
    {
        SetConsoleCP(1251);
        SetConsoleOutputCP(1251);
        Devices devices;
        char num; char N;
        system("cls");
        while (((num = Menu()) != '0') && num != 27) {
            system("cls");
            switch (num) {
                case '1': {
                    do {
                        cout << "Підтвердження дій: \n\t1) Продовжити \n\t2) Вихід\n<:";
N = _getch();
                        cout << endl;
                        if (N == '1') devices.add();
                    } while (N != '2');
                    break;
                }
                case '2': {
                    do {
                        cout << "Підтвердження дій: \n\t1) Продовжити \n\t2) Вихід\n<:";
N = _getch();
                        cout << endl;
                        if (N == '1') { devices.print(); devices.edit(); }
                    } while (N != '2');
                    break;
                }
                case '3': {
                    do {
                        cout << "Підтвердження дій: \n\t1) Продовжити \n\t2) Вихід\n<:";
N = _getch();
                        cout << endl;
                        if (N == '1') { devices.print(); devices.del(); }
                    } while (N != '2');
                    break;
                }
                case '4': {
                    do {
                        cout << "Підтвердження дій: \n\t1) Продовжити \n\t2) Вихід\n<:";
N = _getch();
                        cout << endl;
                        if (N == '1') devices.print();
                    } while (N != '2');
                    break;
                }
                case '5': {
                    do {
                        cout << "Підтвердження дій: \n\t1) Продовжити \n\t2) Вихід\n<:";

```

```

N = _getch();
        cout << endl;
        if (N == '1') devices.sorting();
    } while (N != '2');
    break;
}
case '6': {
    do {
        cout << "Підтвердження дій: \n\t1) Продовжити \n\t2) Вихід\n<:";
N = _getch();
        cout << endl;
        if (N == '1') devices.find();
    } while (N != '2');
    break;
}
case '7': {
    do {
        cout << "Підтвердження дій: \n\t1) Продовжити \n\t2) Вихід\n<:";
N = _getch();
        cout << endl;
        if (N == '1') devices.save();
    } while (N != '2');
    break;
}
case '8': {
    do {
        cout << "Підтвердження дій: \n\t1) Продовжити \n\t2) Вихід\n<:";
N = _getch();
        cout << endl;
        if (N == '1') devices.load();
    } while (N != '2');
    break;
}
}
system("pause");
}

return 0;
}

```