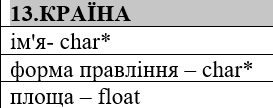
**Лабораторна робота №1**

## Тема: Класи й об'єкти в С++ та C#.

***Мета роботи:*** Одержати практичні навички реалізації класів на С++ та C#.

**Задача 1**

**Умови**



**Порядок виконання роботи.**

1. Визначено клас Country з такими полями:
   * string Name\_ — назва країни (публічне поле).
   * string FormRule\_ — форма правління (приватне поле).
   * double S\_ — площа країни (приватне поле).
2. Визначено конструктори класу:
   * Конструктор без параметрів: Country() = default;
   * Конструктор з параметрами: Country(string Name, string FormRule, double S);
   * Конструктор копіювання не реалізовано явно, але використовується конструктор копіювання за замовчуванням.
3. Визначено функції-компоненти (методи) для перегляду та встановлення полів:
   * Гетери: name(), FormRule(), S()
   * Сетери: set\_name(), set\_FormRule(), set\_S(), set\_Country()
   * Метод для виводу: Print()
4. Покажчик на компонент-функцію
5. Покажчик на екземпляр класу
6. Використання покажчика на об’єкт і покажчика на компонент-функцію

**Код:**

**Country.h**

#pragma once

#include "string";

using namespace std;

class Country

{

public:

string Name\_;

private:

string FormRule\_;

double S\_=0;

public:

Country() = default;

Country(string Name,string FormRule,double S);

const string name() const { return Name\_; }

const string FormRule() const { return FormRule\_; }

double S() const noexcept { return S\_; }

void set\_name(string name);

void set\_FormRule(string FormRule);

void set\_S(double S);

void set\_Country(string Name, string FormRule, double S);

void Print() const;

};

**Country.cpp**

#include "Country.h"

#include "iostream"

using namespace std;

Country::Country(string Name, string FormRule, double S)

:Name\_(move(Name)),

FormRule\_(move(FormRule)),

S\_(S) {

set\_name(Name\_);

set\_FormRule(FormRule\_);

set\_S(S\_);

}

void Country::set\_name(string Name) {

if (Name.empty())

{

cout << "Назва не може бути порожнію" << endl; return;

}

Name\_ = move(Name);

}

void Country::set\_FormRule(string FormRule) {

if (FormRule.empty())

{

cout << "Правління не може бути ніяким" << endl; return;

}

FormRule\_ = move(FormRule);

}

void Country::set\_S(double S) {

if (S <= 0)

{

cout << "Рлоща не може бути відємною або нульовою" << endl; return;

}

S\_ = S;

}

void Country::set\_Country(string Name, string FormRule, double S) {

if (Name.empty())

{

cout << "Назва не може бути порожнію" << endl; return;

}

Name\_ = move(Name);

if (FormRule.empty())

{

cout << "Правління не може бути ніяким" << endl; return;

}

FormRule\_ = move(FormRule);

if (S <= 0)

{

cout << "Площа не може бути відємною або нульовою" << endl; return;

}

S\_ = S;

}

void Country::Print() const{

if (Name\_.empty()) {

cout << "Нема що виводити" << endl;

return;

}

cout << "Назва країни: " << Name\_ << endl

<< "Форма правління: " << FormRule\_ << endl

<< "Площа: " << S\_ << endl;

}

**LR1.cpp**

#include "Country.h"

#include "iostream"

int main() {

string FRule, Name;

Country c;

double S;

setlocale(0, "ukr");

bool F = true;

int Choose;

while (F) {

system("cls");

cout << R"(Створення країни:

1.Ввести країну

2.Вивести країни

3.Вийти

)";

cout << "Зробіть вибір: ";

cin >> Choose;

switch (Choose) {

case 1: {

cout << "Введіть назву краЇни\n";

cin >> Name;

cout << "Введіть форма правління\n";

cin >> FRule;

cout << "Введіть площу\n";

cin >> S;

c.set\_Country(Name, FRule, S);

break;

}

case 2: {

c.Print();

system("pause");

break;

}

case 3: {

F = false;

break;

}

default: {

cout << "Такої дії не існує";

break;

}

}

}

}

РЕЗУЛЬТАТ ПРОГРАМИ

1)Створення країни:

1.Ввести країну

2.Вивести країни

3.Вийти

Зроб?ть виб?р: 1

Введ?ть назву краЇни

Ukraine

Введ?ть форма правл?ння

Dictator

Введ?ть площу

4533

Створення країни:

1.Ввести країну

2.Вивести країни

3.Вийти

Зроб?ть виб?р: 2

Назва країни: Ukraine

Форма правл?ння: Dictator

Площа: 4533

Press any key to continue . . .

Створення країни:

1.Ввести країну

2.Вивести країни

3.Вийти

Зроб?ть виб?р: 3

C:\Users\Admin\source\repos\Project1\x64\Debug\Project1.exe (process 4748) exited with code 0 (0x0).

Press any key to close this window . . .

2)

Створення країни:

1.Ввести країну

2.Вивести країни

3.Вийти

Зроб?ть виб?р: 1

Введ?ть назву краЇни

Ukr

Введ?ть форма правл?ння

Dim

Введ?ть площу

-45

Площа не може бути відємною або нульовою

Створення країни:

1.Ввести країну

2.Вивести країни

3.Вийти

Зроб?ть виб?р: 2

Назва країни: Ukr

Форма правл?ння: Dim

Площа: 0

Press any key to continue . . .

**Задача 2**

**Умови**

Створити клас Rectangle, розробивши наступні елементи класу:

* 1. Поля:
     1. int а, b;
  2. Конструктор, що дозволяє створити екземпляр класу із заданими довжинами сторін.
  3. Методи, що дозволяють:
     1. вивести довжини сторін прямокутника на екран;
     2. обчислити периметр прямокутника;
     3. обчислити площу прямокутника.
  4. Властивості:
     1. одержати-встановити довжини сторін прямокутника (доступне для читання і запису);
     2. які надають можливість встановити, чи є даний прямокутник квадратом (доступне тільки для читання).

**Порядок виконання роботи:**

 **Визначено клас Rectangle** з приватними полями:

* int a\_ — довжина сторони a;
* int b\_ — довжина сторони b.

 **Реалізовано конструктори:**

* Конструктор без параметрів: Rectangle() = default;
* Конструктор з параметрами: Rectangle(int a, int b);
* Конструктор копіювання: Rectangle(const Rectangle& other); (виводить повідомлення *"копію зробленно"*).

 **Визначено деструктор:**

* ~Rectangle(); — при знищенні об’єкта виводить повідомлення *"Видаленно масив даних"*.

 **Визначено методи доступу (гетери/сетери):**

* Гетери: a(), b() для отримання значень сторін.
* Сетери: set\_a(double a), set\_b(double b) — виконують перевірку правильності введення (сторони мають бути більші за 0).

 **Реалізовано основні методи:**

* int perimeter() const — обчислення периметра прямокутника.
* int area() const — обчислення площі.
* bool is\_square() const — перевірка, чи є прямокутник квадратом.
* void Init() — введення сторін прямокутника з клавіатури.
* void Print() const — виведення властивостей прямокутника у вигляді таблиці (один рядок: a, b, P, S, квадрат чи ні).

 **У демонстраційній програмі (main.cpp):**

* Користувач вводить кількість прямокутників N.
* Створюється динамічний масив прямокутників: Rectangle\* r1 = new Rectangle[N];
* Для кожного прямокутника викликається метод Init().

 **Передбачено меню:**

* Вивести всі прямокутники у вигляді таблиці (Print() у циклі).
* Відмовитись від виводу.
* Обробка неправильного вибору (goto + повтор).

 **Реалізовано копіювання:**

* Користувачу пропонується вибрати один з прямокутників.
* Створюється копія обраного прямокутника (r2 = r1[i];).
* Викликається конструктор копіювання, на екран виводиться повідомлення *"копію зробленно"*.
* Після цього копія також відображається у вигляді таблиці.

 **Звільнення пам’яті:**

* Після завершення роботи програми масив знищується: delete[] r1;
* Викликаються деструктори для кожного об’єкта.

**Код:**

**Rectangle.h**

#pragma once

class Rectangle {

private:

int a\_ = 0; /

int b\_ = 0; /

public:

// Конструктори

Rectangle() = default;

Rectangle(int a, int b);

// Копія

Rectangle(const Rectangle& other) ;

// Деконструктори

~Rectangle();

// Геттери

int a() const noexcept { return a\_; }

int b() const noexcept { return b\_; }

//Сеттери

void set\_a(double a);

void set\_b(double b);

// Методи

int perimeter() const;

int area() const;

bool is\_square() const noexcept { return a\_ == b\_; }

void Init();

void Print() const;

};

**Rectangle.cpp**

#include "Rectangle.h"

#include <iostream>

#include <stdexcept>

#include <iomanip>

using namespace std;

// Конструктор з параметрами

Rectangle::Rectangle(int a, int b) : a\_(a), b\_(b) {

set\_a(a);

set\_b(b);

}

// Вивід сторін

void Rectangle::Print() const{

cout << setw(5) << "|" << setw(5) << a\_

<< setw(5) << "|" << setw(5) << b\_

<< setw(5) << "|" << setw(5) << perimeter()

<< setw(5) << "|" << setw(5) << area()

<< setw(5) << "|" << setw(5) << (is\_square() ? "Так" : "Ні") << endl;

}

Rectangle::~Rectangle() {

cout << "Видаленно масив даних"<<endl;

}

// Периметр

int Rectangle::perimeter() const {

return 2 \* (a\_ + b\_);

}

Rectangle::Rectangle(const Rectangle& other) : a\_(other.a\_), b\_(other.b\_) {

cout << "копію зробленно" << endl;

}

// Площа

int Rectangle::area() const {

return a\_ \* b\_;

}

void Rectangle::Init() {

double a,b;

n:

cout << "Введіть сторону а" << endl;

cin >> a;

cout << "Введіть сторону b" << endl;

cin >> b;

set\_a(a);

set\_b(b);

}

void Rectangle::set\_a(double a) {

if (a <= 0) throw std::invalid\_argument("Сторона a має бути > 0");

a\_ = a;

}

void Rectangle::set\_b(double b) {

if (b <= 0) throw std::invalid\_argument("Сторона b має бути > 0");

b\_ = b;

}

**Main.cpp**

#include "Rectangle.h"

#include <iostream>

#include <iomanip>

using namespace std;

int main() {

setlocale(0,"ukr");

try {

int N = 0;

cout << "Введіть склільки чотирьохкутників ви хочете ввести" << endl;

cin >> N;

Rectangle\* r1=new Rectangle[N];

for (int i = 0; i < N; i++) {

cout << i + 1<<") ";

r1[i].Init();

}

int choise;

m:

system("cls");

cout << "Вивести всі чотирикутники?" << endl

<< "1)Так" <<endl

<< "2)Ні" << endl;

cin >> choise;

switch (choise) {

case 1:{

cout << setw(5) << "№ " << setw(5) <<"|" << setw(5) << "a"

<< setw(5) << "|"<< setw(5) <<"b"

<< setw(5) << "|"<< setw(5) <<"P"

<< setw(5) << "|"<< setw(5) <<"S"

<< setw(5) << "|"<< setw(5) <<"Is squert?" << endl;

for (int i = 0; i < N; i++) {

cout << setw(5) << i + 1 ;

r1[i].Print();

}

system("pause");

system("cls");

break;

}

case 2: {

cout << "Як забажаєте" << endl;

break;

}

default: {

cout << "Такого вибору не існує.Спробуйте знову" << endl;

system("pause");

goto m;

}

}

bool f=true;

while (f) {

system("cls");

cout << "Зробити копію певного чотирихкутника?" << endl

<< "1)Так" << endl

<< "2)Ні" << endl;

cin >> choise;

switch (choise) {

case 1: {

int k;

Rectangle r2;

cout << setw(5) << "№ " << setw(5) << "|" << setw(5) << "a"

<< setw(5) << "|" << setw(5) << "b"

<< setw(5) << "|" << setw(5) << "P"

<< setw(5) << "|" << setw(5) << "S"

<< setw(5) << "|" << setw(5) << "Is squert?" << endl;

for (int i = 0; i < N; i++) {

cout << setw(5) << i + 1;

r1[i].Print();

}

n:

cout << "Введіть цифру того чотирьохкутника якого ви хочете скопіювати" << endl;

cin >> k;

if (k > N || k < 0) {

cout << "Неправильний вибір.Спробуйте ще)" << endl;

system("pause");

system("cls");

goto n;

}

for (int i = 0; i < N; i++) {

if (i + 1 == k) {

r2 = r1[i];

}

}

cout << setw(5) << "№ " << setw(5) << "|" << setw(5) << "a"

<< setw(5) << "|" << setw(5) << "b"

<< setw(5) << "|" << setw(5) << "P"

<< setw(5) << "|" << setw(5) << "S"

<< setw(5) << "|" << setw(5) << "Is squert?" << endl;

cout << setw(5) << 1;

r2.Print();

f = false;

break;

}

case 2: {

cout << "Як забажаєте" << endl;

f = false;

break;

}

default: {

cout << "Такого вибору не існує.Спробуйте знову" << endl;

system("pause");

}

}

}

delete[] r1;

}

catch (const exception& ex) {

cerr << "Помилка: " << ex.what() << "\n";

}

return 0;

}

**РЕЗУЛЬТАТ ПРОГРАМИ**

1) Введ?ть скл?льки чотирьохкутник?в ви хочете ввести

3

1) Введ?ть сторону а

5

Введ?ть сторону b

7

2) Введ?ть сторону а

4

Введ?ть сторону b

6

3) Введ?ть сторону а

5

Введ?ть сторону b

5

Вивести вс? чотирикутники?

1)Так

2)Н?

1

№ | a | b | P | S |Is squert?

1 | 5 | 7 | 24 | 35 | Н?

2 | 4 | 6 | 20 | 24 | Н?

3 | 5 | 5 | 20 | 25 | Так

Зробити коп?ю певного чотирихкутника?

1)Так

2)Н?

1

№ | a | b | P | S |Is squert?

1 | 5 | 7 | 24 | 35 | Н?

2 | 4 | 6 | 20 | 24 | Н?

3 | 5 | 5 | 20 | 25 | Так

Введ?ть цифру того чотирьохкутника якого ви хочете скоп?ювати

2

№ | a | b | P | S |Is squert?

1 | 4 | 6 | 20 | 24 | Н?

Видаленно масив даних

Видаленно масив даних

Видаленно масив даних

Видаленно масив даних

**Задача 3**

**Умови**

Зробити клас Парк Атракціонів з базовим функціоналом

**Порядок виконання роботи:**

 **Визначено клас Ticket з властивостями:**

* FirstName — ім’я відвідувача;
* LastName — прізвище;
* NameAttraction — назва атракціону;
* Age — вік;
* DateTime — дата і час відвідування.

 **Реалізовано конструктори:**

* Конструктор без параметрів — створює квиток з даними за замовчуванням;
* Конструктор з параметрами — ініціалізує всі властивості;
* Конструктор копіювання — створює новий квиток як копію іншого.

 **Реалізовано перевірку введення у формі:**

* Вік повинен бути числом від 1 до 100;
* Ім’я та прізвище містять тільки літери;
* Час вводиться у форматі HH:mm;
* Кнопка підтвердження активна тільки при правильному заповненні всіх полів.

 **У формі (Form1):**

* Користувач вводить дані квитка (ім’я, прізвище, вік, атракціон, дату та час);
* Створений квиток додається до списку tickets та виводиться у DataGridView.

 **Передбачено меню з функціями:**

* Збереження списку квитків у бінарний файл;
* Зчитування квитків із файлу у таблицю;
* Створення об’єкта за замовчуванням;
* Копіювання вибраного квитка через додаткову форму;
* Пошук квитків за ім’ям, прізвищем або атракціоном із виведенням результатів у другу таблицю.

 **Реалізовано копіювання:**

* Користувач обирає квиток зі списку;
* Створюється копія за допомогою конструктора копіювання;
* Поля копії виводяться у форму.

**Код:**

**Ticket.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LR1\_Zavd3

{

internal class Ticket

{

//--------Властивості----------

public string FirstName { get; set; }

public string LastName { get; set; }

public string NameAttraction { get; set; }

public int Age { get; set; }

public DateTime DateTime { get; set; }

//--------Конструктори----------

public Ticket() // конструктор за замовчуванням

{

FirstName = "Назар";

LastName = "Шейкін";

NameAttraction = "Freefall tower";

Age = 18;

DateTime = new DateTime(2000, 4, 1, 0, 0, 0);

}

public Ticket(string firstName, string lastName, string nameAttraction, int year, DateTime dateTime) // з параметрами

{

FirstName = firstName;

LastName = lastName;

NameAttraction = nameAttraction;

Age = year;

DateTime= dateTime;

}

public Ticket(Ticket other) // конструктор копіювання

{

FirstName = other.FirstName;

LastName = other.LastName;

NameAttraction = other.NameAttraction;

Age = other.Age;

DateTime= other.DateTime;

}

//--------Методи----------

}

}

**Form1.cs:**

using System.Diagnostics.Metrics;

using System.Globalization;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement.ScrollBar;

using Lab1\_zavd3;

namespace LR1\_Zavd3

{

public partial class Form1 : Form

{

string[] NameAttraction = new string[] { "Roller coaster", "Freefall tower", "Funnel slides", "Каруселі", "Rail-free trains" };

List<Ticket> tickets = new List<Ticket>();

int count = 0;

public Form1()

{

InitializeComponent();

comboBox1.Items.AddRange(NameAttraction);

comboBox1.SelectedIndexChanged += button\_Check;

nameBox.TextChanged += button\_Check;

last\_nameBox.TextChanged += button\_Check;

yearBox.TextChanged += button\_Check;

datePicker.ValueChanged += button\_Check;

timeBox.TextChanged += button\_Check;

}

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

}

//-----------------Перевірка-------------------

private void button\_Check(object sender, EventArgs e)

{

// Рік

if (!string.IsNullOrWhiteSpace(yearBox.Text) && !System.Text.RegularExpressions.Regex.IsMatch(yearBox.Text, @"^\d+$"))

{

MessageBox.Show("Тут повинно бути тільки цифри!");

button1.Enabled = false;

button1.BackColor = Color.Red;

return;

}

if (!string.IsNullOrWhiteSpace(yearBox.Text))

{

int year = int.Parse(yearBox.Text);

if (year < 1 || year > 100)

{

MessageBox.Show("Вік повинен бути від 1 до 100!");

button1.Enabled = false;

button1.BackColor = Color.Red;

return;

}

}

// Ім'я

if (!string.IsNullOrWhiteSpace(nameBox.Text) && !System.Text.RegularExpressions.Regex.IsMatch(nameBox.Text, @"^[a-zA-Zа-яА-ЯїЇєЄіІґҐ]+$"))

{

MessageBox.Show("Тут повинно бути тільки літери!");

button1.Enabled = false;

button1.BackColor = Color.Red;

return;

}

// Прізвище

if (!string.IsNullOrWhiteSpace(last\_nameBox.Text) && !System.Text.RegularExpressions.Regex.IsMatch(last\_nameBox.Text, @"^[a-zA-Zа-яА-ЯїЇєЄіІґҐ]+$"))

{

MessageBox.Show("Тут повинно бути тільки літери!");

button1.Enabled = false;

button1.BackColor = Color.Red;

return;

}

// Перевірка часу

bool isTimeValid = DateTime.TryParseExact(timeBox.Text, "HH:mm",

CultureInfo.InvariantCulture, DateTimeStyles.None, out DateTime time);

// Кнопка активна тільки якщо всі поля заповнені + валідний час

bool allFilled = comboBox1.SelectedIndex != -1

&& !string.IsNullOrWhiteSpace(nameBox.Text)

&& !string.IsNullOrWhiteSpace(last\_nameBox.Text)

&& !string.IsNullOrWhiteSpace(yearBox.Text)

&& isTimeValid;

button1.Enabled = allFilled;

button1.BackColor = allFilled ? Color.Green : Color.Red;

}

private void button1\_Click(object sender, EventArgs e)

{

var ticket = new Ticket(

nameBox.Text,

last\_nameBox.Text,

comboBox1.SelectedItem.ToString(),

int.Parse(yearBox.Text),

datePicker.Value.Date + TimeSpan.Parse(timeBox.Text)

);

tickets.Add(ticket);

WriteInDataGrid(ticket);

}

private void WriteInDataGrid(Ticket ticket)

{

int rowIndex = dataGridView1.Rows.Add();

DataGridViewRow row = dataGridView1.Rows[rowIndex];

row.Cells[0].Value = count + 1;

row.Cells[1].Value = ticket.FirstName;

row.Cells[2].Value = ticket.LastName;

row.Cells[3].Value = ticket.Age;

row.Cells[4].Value = ticket.DateTime;

row.Cells[5].Value = ticket.NameAttraction;

count++;

}

private void nameBox\_TextChanged(object sender, EventArgs e)

{

}

private void зберегтиToolStripMenuItem\_Click(object sender, EventArgs e)

{

}

private void зберегтиУФайлToolStripMenuItem\_Click(object sender, EventArgs e)

{

using (SaveFileDialog sfd = new SaveFileDialog())

{

sfd.Filter = "BIN files (\*.bin)|\*.bin|All files (\*.\*)|\*.\*";

if (sfd.ShowDialog() == DialogResult.OK)

{

using (BinaryWriter writer = new BinaryWriter(File.Open(sfd.FileName, FileMode.Create)))

{

foreach (var ticket in tickets)

{

writer.Write(ticket.FirstName);

writer.Write(ticket.LastName);

writer.Write(ticket.Age);

writer.Write(ticket.DateTime.ToBinary()); // зберігаємо DateTime у бінарному вигляді

writer.Write(ticket.NameAttraction);

}

}

MessageBox.Show("Дані збережено у бінарний файл!");

}

}

}

private void зчитатиЗФайлуToolStripMenuItem\_Click(object sender, EventArgs e)

{

using (OpenFileDialog ofd = new OpenFileDialog())

{

ofd.Filter = "BIN files (\*.bin)|\*.bin|All files (\*.\*)|\*.\*";

if (ofd.ShowDialog() == DialogResult.OK)

{

tickets.Clear();

dataGridView1.Rows.Clear();

count = 0;

using (BinaryReader reader = new BinaryReader(File.Open(ofd.FileName, FileMode.Open)))

{

while (reader.BaseStream.Position < reader.BaseStream.Length)

{

string firstName = reader.ReadString();

string lastName = reader.ReadString();

int age = reader.ReadInt32();

DateTime dateTime = DateTime.FromBinary(reader.ReadInt64());

string attraction = reader.ReadString();

var ticket = new Ticket(firstName, lastName, attraction, age, dateTime);

tickets.Add(ticket);

int rowIndex = dataGridView1.Rows.Add();

DataGridViewRow row = dataGridView1.Rows[rowIndex];

row.Cells[0].Value = ++count;

row.Cells[1].Value = ticket.FirstName;

row.Cells[2].Value = ticket.LastName;

row.Cells[3].Value = ticket.Age;

row.Cells[4].Value = ticket.DateTime;

row.Cells[5].Value = ticket.NameAttraction;

}

}

MessageBox.Show("Дані завантажено з бінарного файлу!");

}

}

}

private void копіюОбєктуToolStripMenuItem\_Click(object sender, EventArgs e)

{

if (tickets.Count == 0)

{

MessageBox.Show("Немає квитків для копіювання!");

return;

}

using (var copyForm = new CopyTicketForm(tickets.Count))

{

if (copyForm.ShowDialog() == DialogResult.OK)

{

int index = copyForm.SelectedIndex;

var original = tickets[index];

nameBox.Text = original.FirstName;

last\_nameBox.Text = original.LastName;

yearBox.Text = original.Age.ToString();

comboBox1.Text = original.NameAttraction;

timeBox.Text = original.DateTime.ToString("HH:mm");

datePicker.Value = original.DateTime;

}

}

}

private void поДефолтуToolStripMenuItem\_Click(object sender, EventArgs e)

{

Ticket defolt = new Ticket();

nameBox.Text = defolt.FirstName;

last\_nameBox.Text = defolt.LastName;

yearBox.Text = defolt.Age.ToString();

comboBox1.Text = defolt.NameAttraction;

timeBox.Text = defolt.DateTime.ToString("HH:mm");

datePicker.Value = defolt.DateTime;

}

private void поІменіToolStripMenuItem\_Click(object sender, EventArgs e)

{

if (tickets.Count == 0)

{

MessageBox.Show("Список квитків порожній!");

return;

}

// Створюємо список всіх імен

List<string> allNames = tickets.Select(t => t.FirstName).ToList();

using (var searchForm = new NameSearchForm(allNames))

{

if (searchForm.ShowDialog() == DialogResult.OK)

{

string name = searchForm.NameToSearch;

dataGridView2.Rows.Clear();

int tempCount = 0;

foreach (var ticket in tickets)

{

if (ticket.FirstName.Equals(name, StringComparison.OrdinalIgnoreCase))

{

int rowIndex = dataGridView2.Rows.Add();

DataGridViewRow row = dataGridView2.Rows[rowIndex];

row.Cells[0].Value = ++tempCount;

row.Cells[1].Value = ticket.FirstName;

row.Cells[2].Value = ticket.LastName;

row.Cells[3].Value = ticket.Age;

row.Cells[4].Value = ticket.DateTime;

row.Cells[5].Value = ticket.NameAttraction;

}

}

if (tempCount == 0)

MessageBox.Show("Квитки з таким ім'ям не знайдено!");

}

}

}

private void поПрізвещіToolStripMenuItem\_Click(object sender, EventArgs e)

{

if (tickets.Count == 0)

{

MessageBox.Show("Список квитків порожній!");

return;

}

// Створюємо список всіх імен

List<string> allNames = tickets.Select(t => t.LastName).ToList();

using (var searchForm = new NameSearchForm(allNames))

{

if (searchForm.ShowDialog() == DialogResult.OK)

{

string name = searchForm.NameToSearch;

dataGridView2.Rows.Clear();

int tempCount = 0;

foreach (var ticket in tickets)

{

if (ticket.LastName.Equals(name, StringComparison.OrdinalIgnoreCase))

{

int rowIndex = dataGridView2.Rows.Add();

DataGridViewRow row = dataGridView2.Rows[rowIndex];

row.Cells[0].Value = ++tempCount;

row.Cells[1].Value = ticket.FirstName;

row.Cells[2].Value = ticket.LastName;

row.Cells[3].Value = ticket.Age;

row.Cells[4].Value = ticket.DateTime;

row.Cells[5].Value = ticket.NameAttraction;

}

}

if (tempCount == 0)

MessageBox.Show("Квитки з таким ім'ям не знайдено!");

}

}

}

private void назвіАтракціонуToolStripMenuItem\_Click(object sender, EventArgs e)

{

if (tickets.Count == 0)

{

MessageBox.Show("Список квитків порожній!");

return;

}

// Створюємо список всіх імен

List<string> allNames = tickets.Select(t => t.NameAttraction).ToList();

using (var searchForm = new NameSearchForm(allNames))

{

if (searchForm.ShowDialog() == DialogResult.OK)

{

string name = searchForm.NameToSearch;

dataGridView2.Rows.Clear();

int tempCount = 0;

foreach (var ticket in tickets)

{

if (ticket.NameAttraction.Equals(name, StringComparison.OrdinalIgnoreCase))

{

int rowIndex = dataGridView2.Rows.Add();

DataGridViewRow row = dataGridView2.Rows[rowIndex];

row.Cells[0].Value = ++tempCount;

row.Cells[1].Value = ticket.FirstName;

row.Cells[2].Value = ticket.LastName;

row.Cells[3].Value = ticket.Age;

row.Cells[4].Value = ticket.DateTime;

row.Cells[5].Value = ticket.NameAttraction;

}

}

if (tempCount == 0)

MessageBox.Show("Квитки з таким ім'ям не знайдено!");

}

}

}

}

}

**NameSearchForm.cs:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace Lab1\_zavd3

{

public partial class NameSearchForm : Form

{

public string NameToSearch { get; private set; } = "";

public NameSearchForm(List<string> names)

{

InitializeComponent();

comboBox1.Items.AddRange(names.Distinct().ToArray());

if (comboBox1.Items.Count > 0) comboBox1.SelectedIndex = 0;

}

private void button1\_Click(object sender, EventArgs e)

{

if (comboBox1.SelectedItem != null)

{

NameToSearch = comboBox1.SelectedItem.ToString();

this.DialogResult = DialogResult.OK;

this.Close();

}

}

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

button1.Enabled = true;

button1.BackColor = Color.Green;

}

}

}

**CopyTicketForm.cs:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Lab1\_zavd3

{

public partial class CopyTicketForm : Form

{

public int SelectedIndex { get; private set; } = -1;

public CopyTicketForm(int maxIndex)

{

InitializeComponent();

numericUpDown1.Minimum = 1;

numericUpDown1.Maximum = maxIndex;

}

private void button1\_Click(object sender, EventArgs e)

{

SelectedIndex = (int)numericUpDown1.Value - 1; // Зсув на індекс списку

this.DialogResult = DialogResult.OK;

this.Close();

}

private void numericUpDown1\_ValueChanged(object sender, EventArgs e)

{

button1.Enabled = true;

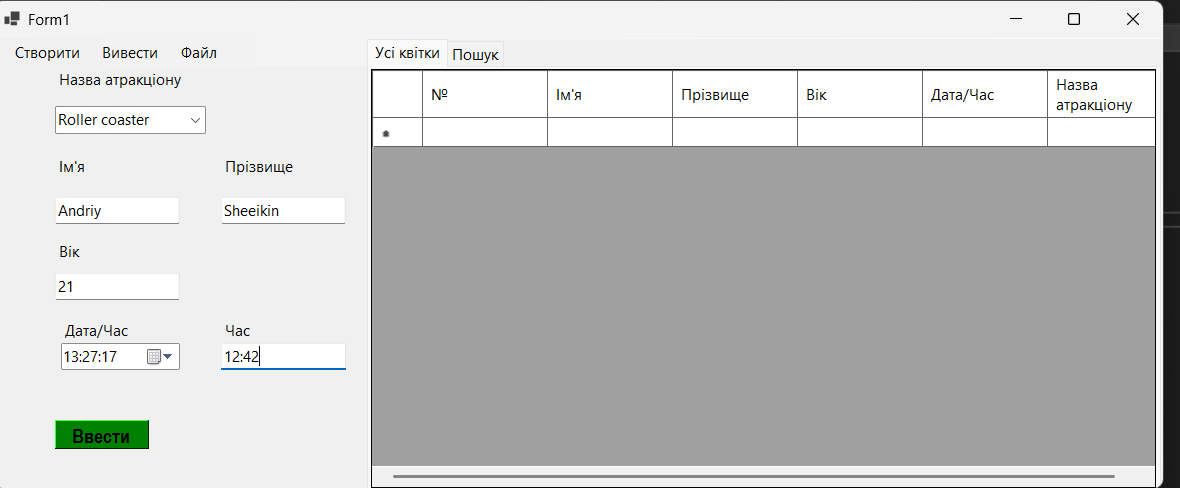
button1.BackColor = Color.Green;

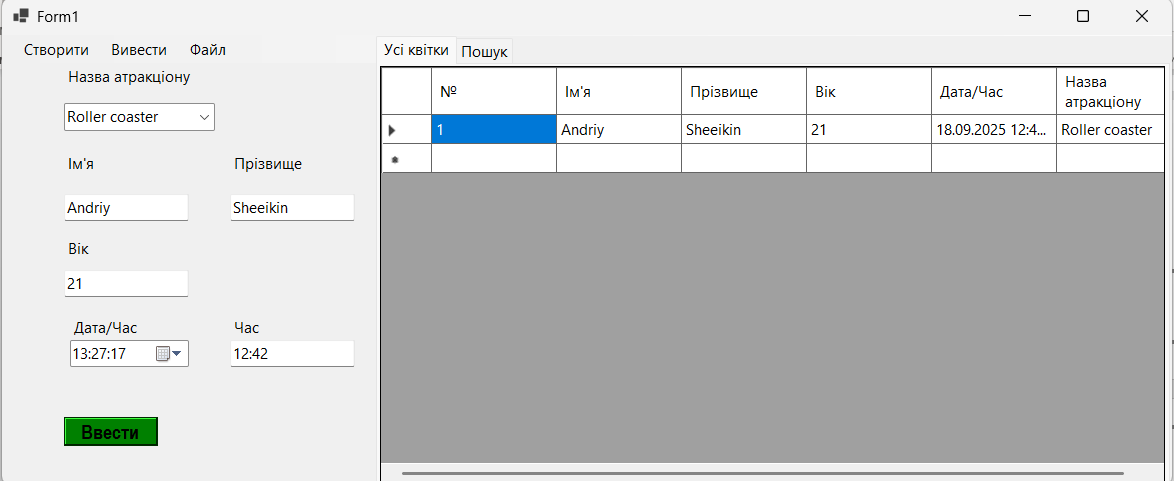
}

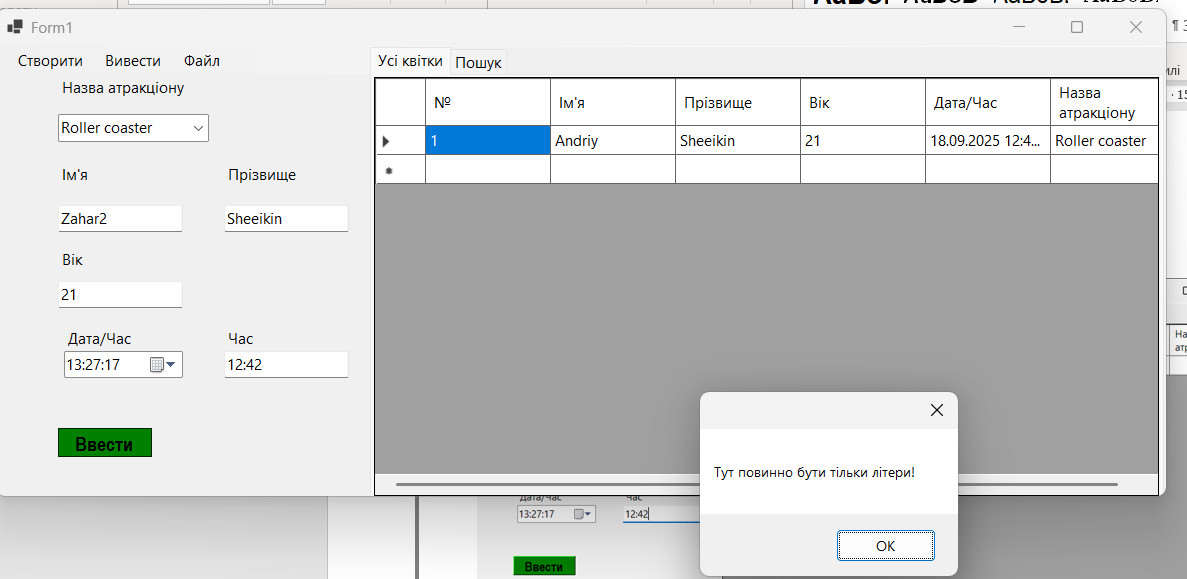
}

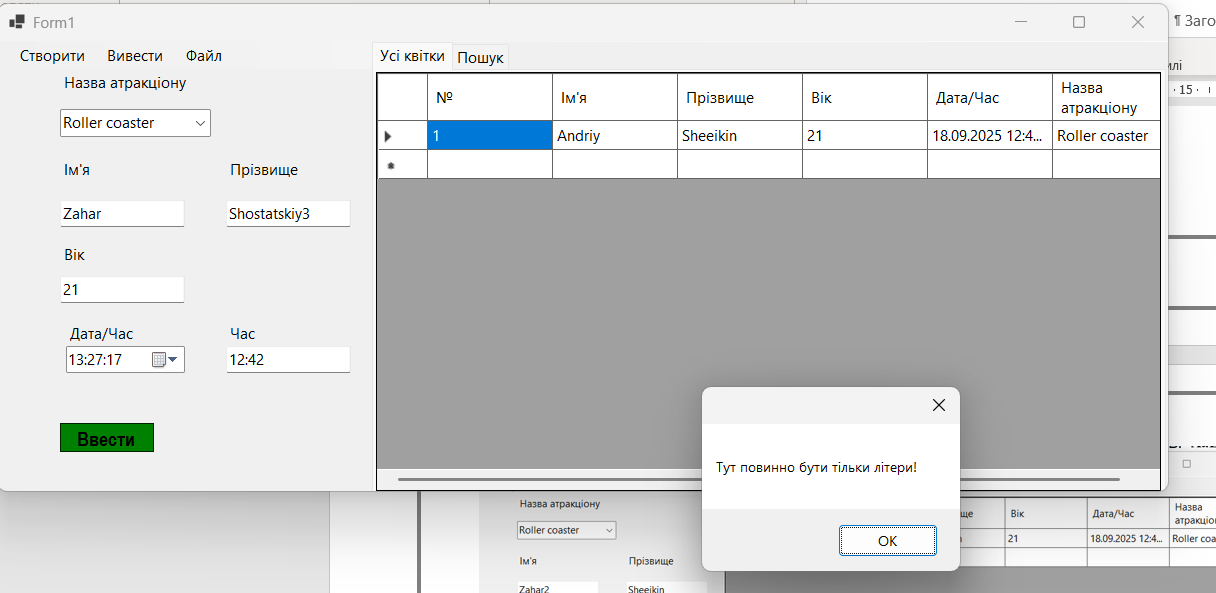
}

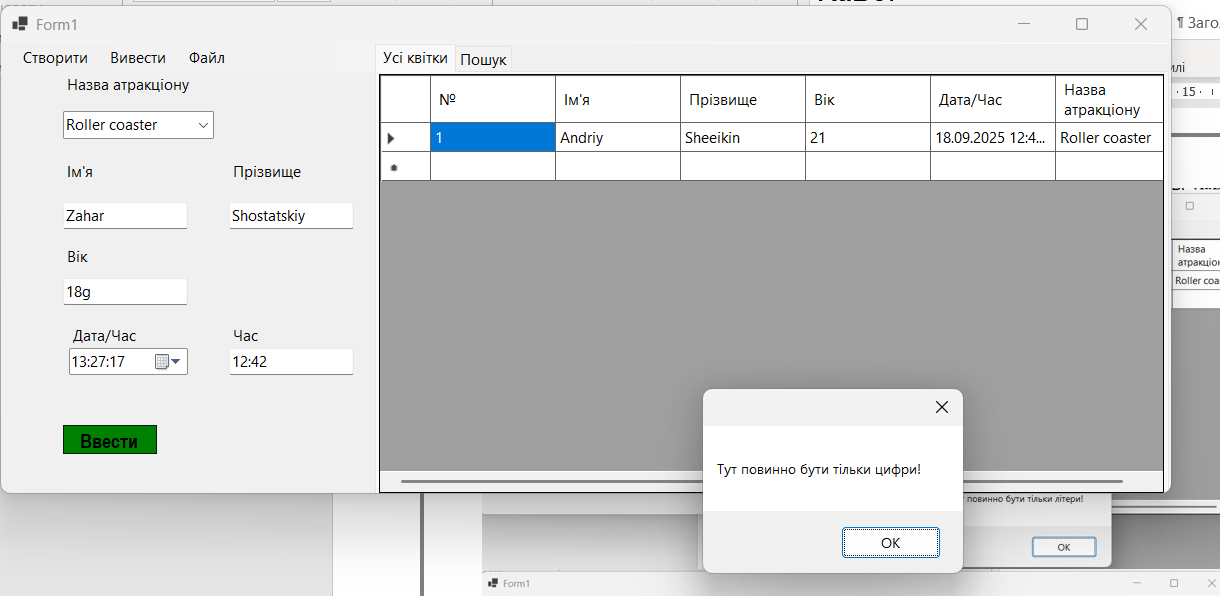
**РЕЗУЛЬТАТ ПРОГРАМИ**

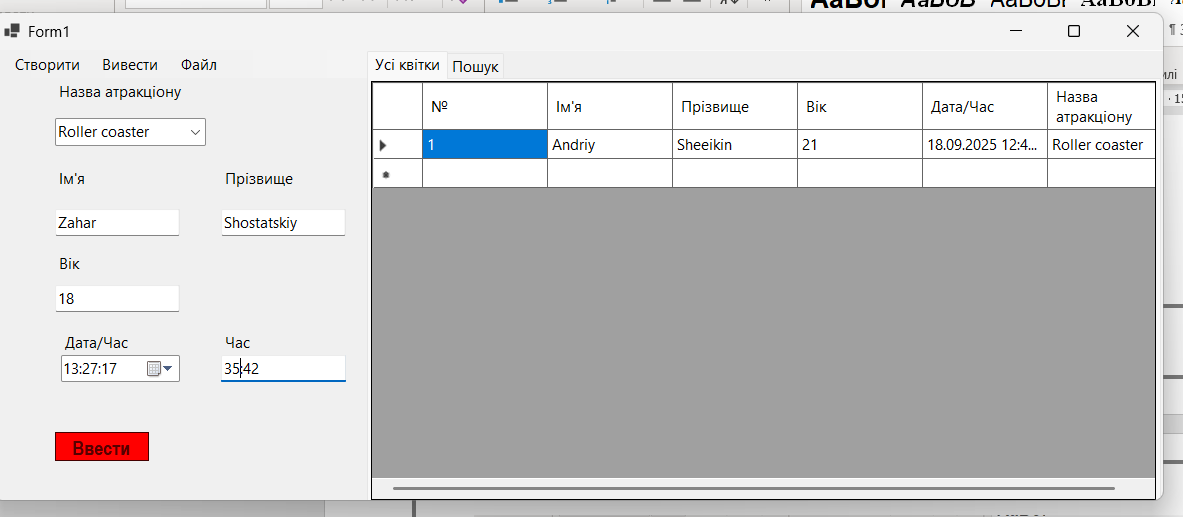
****

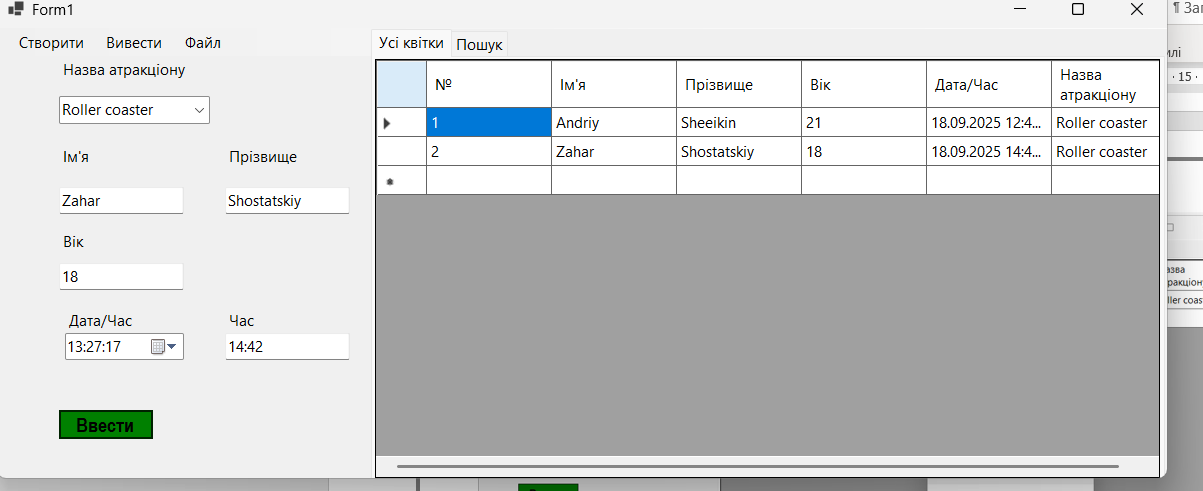
****

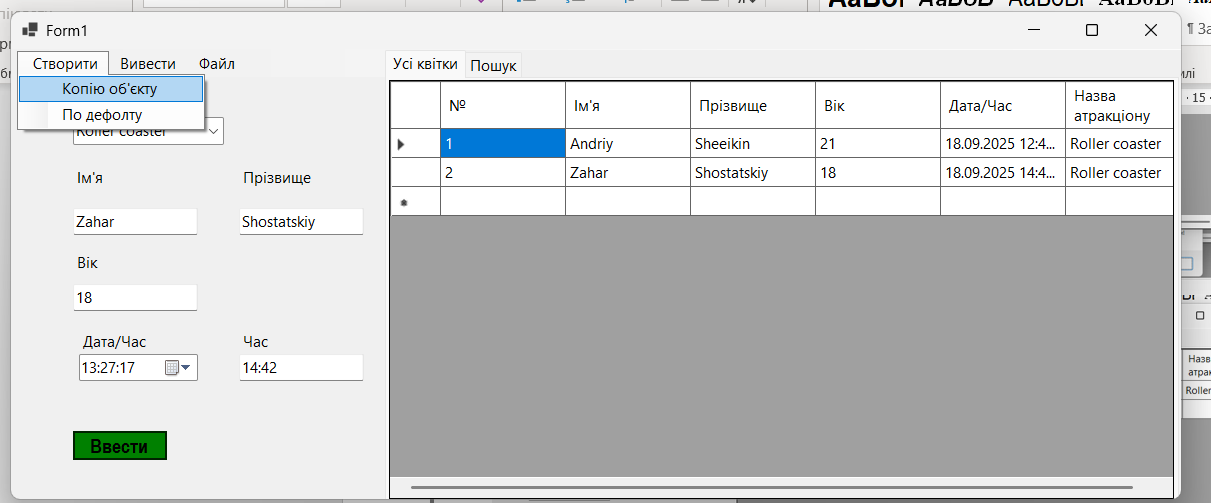
****

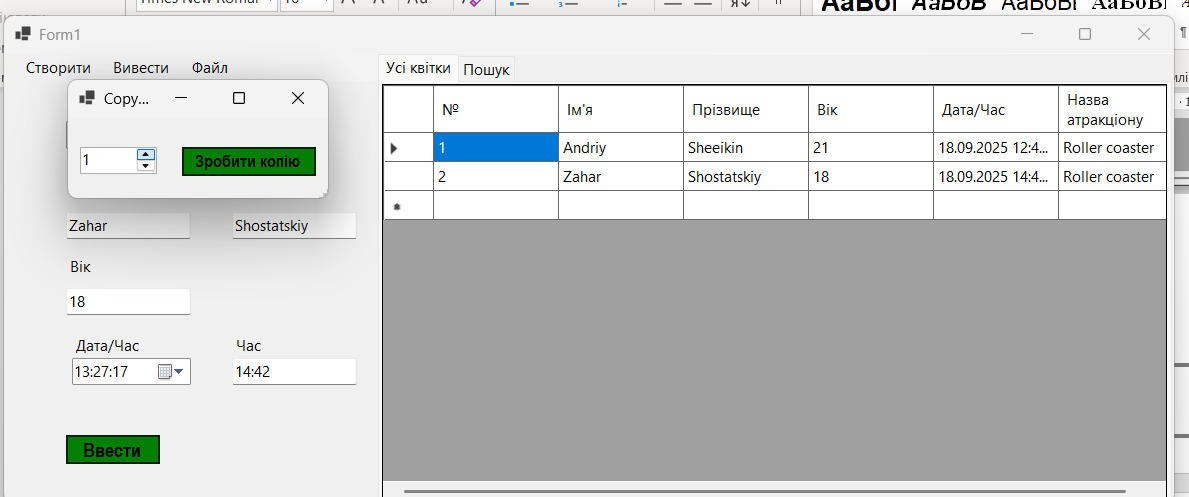
****

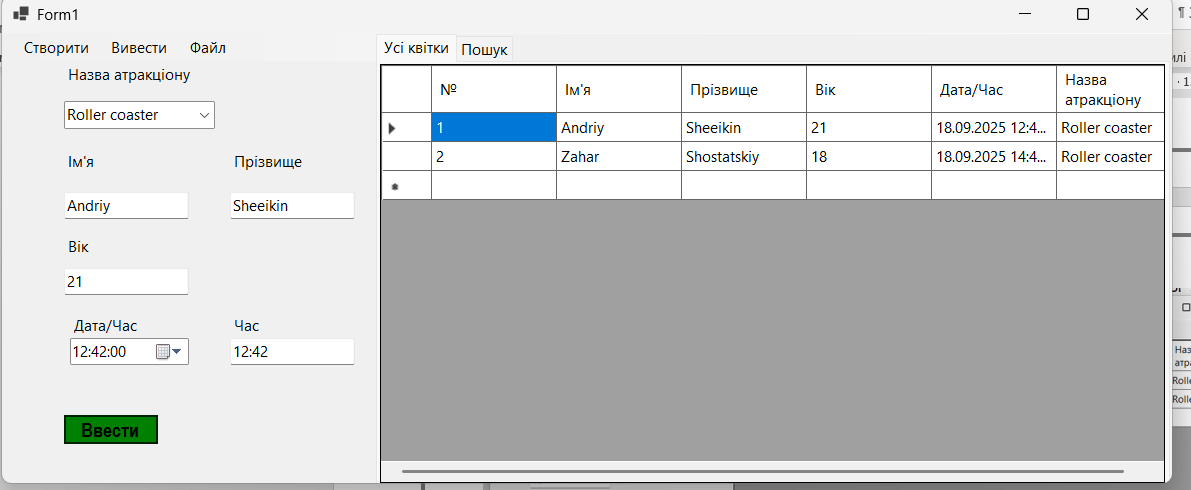
****

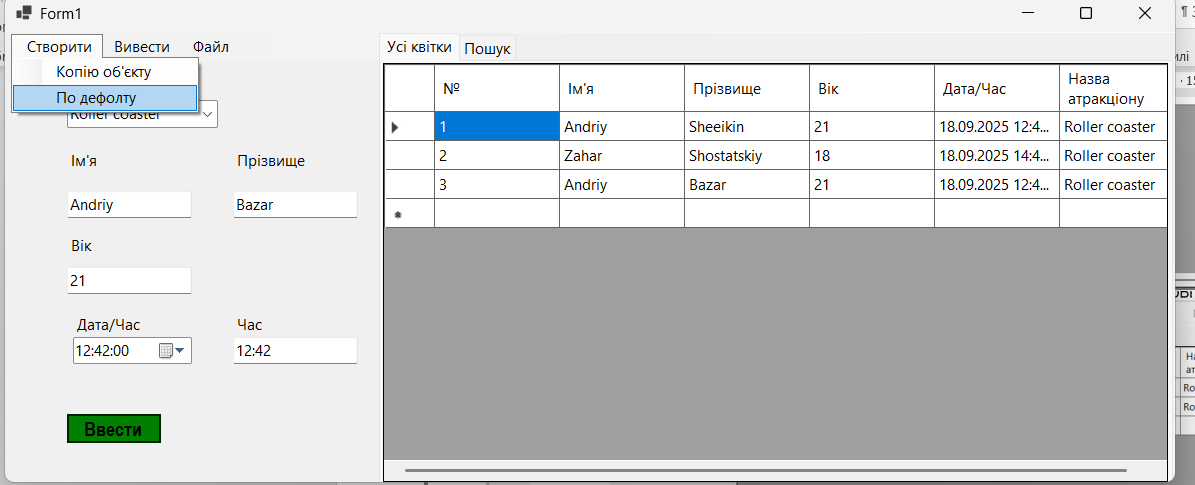
****

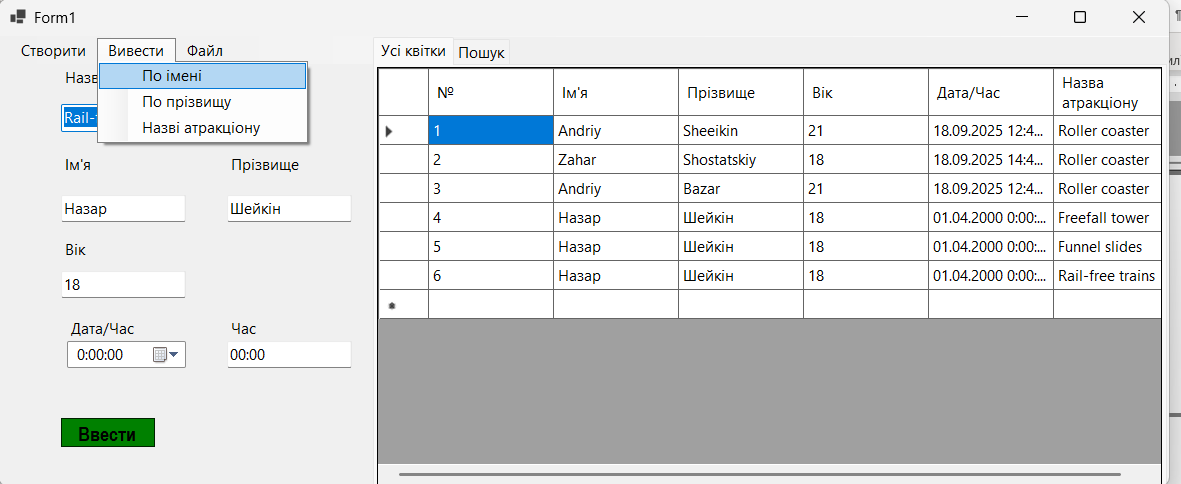
****

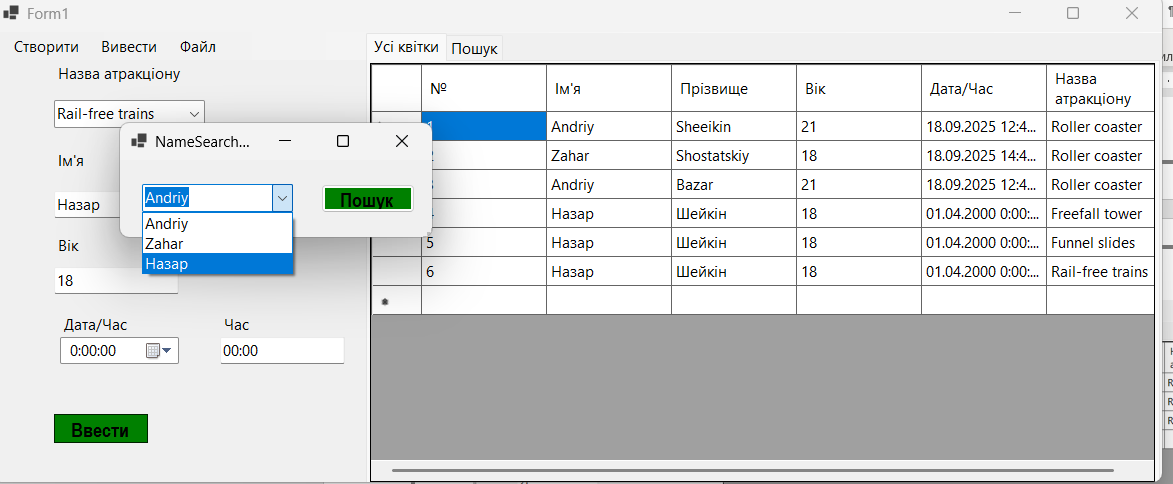
****

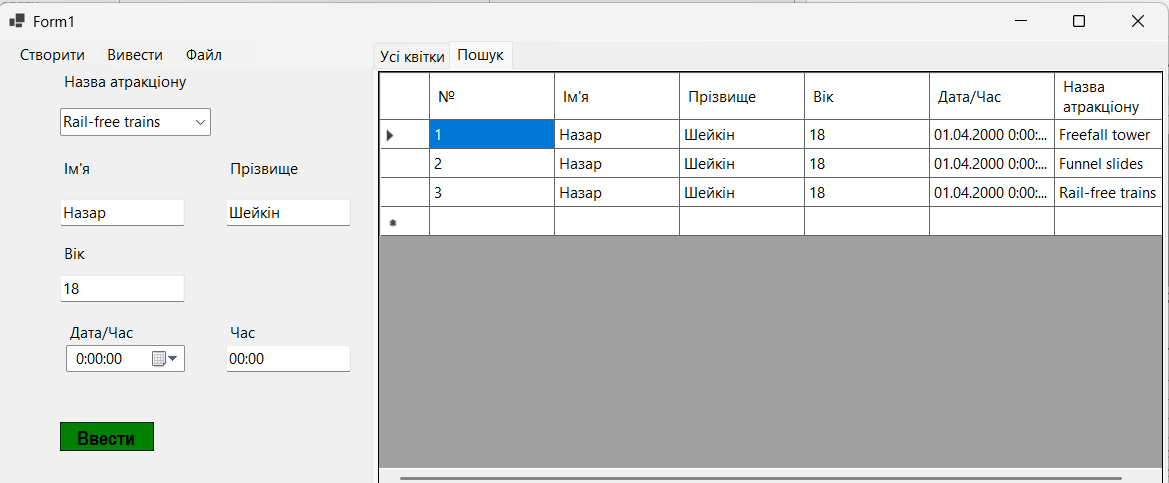
****

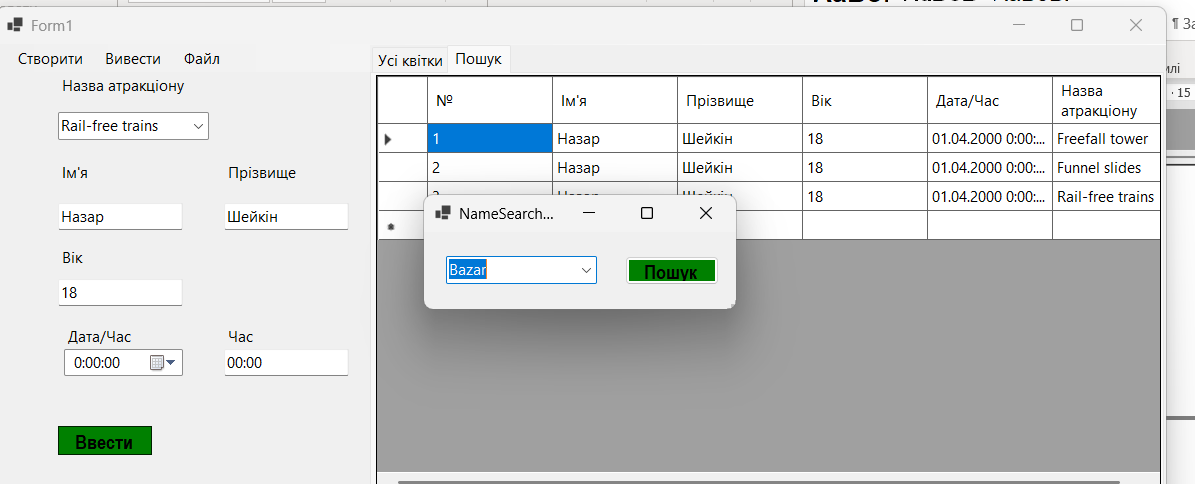
****

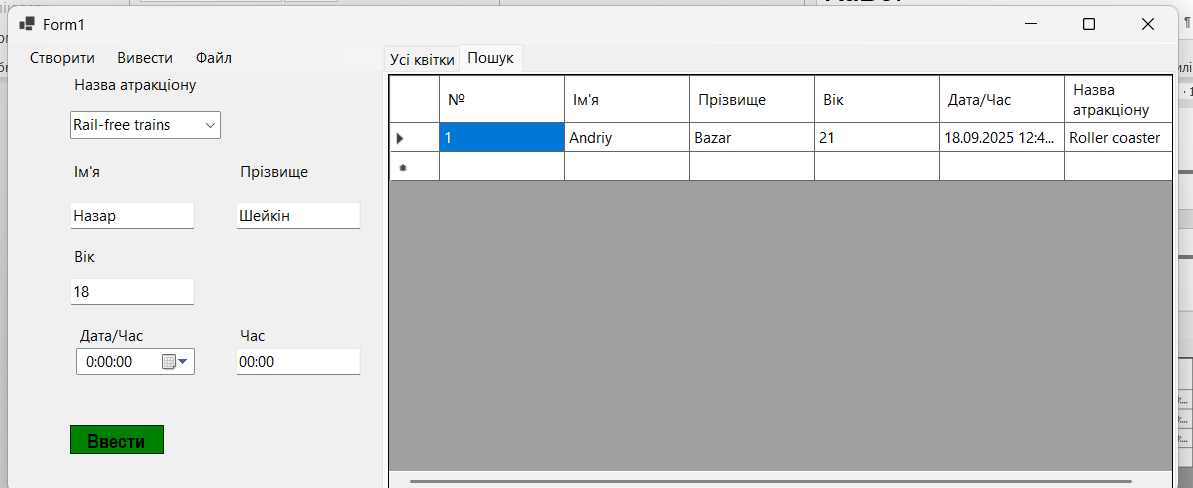
****

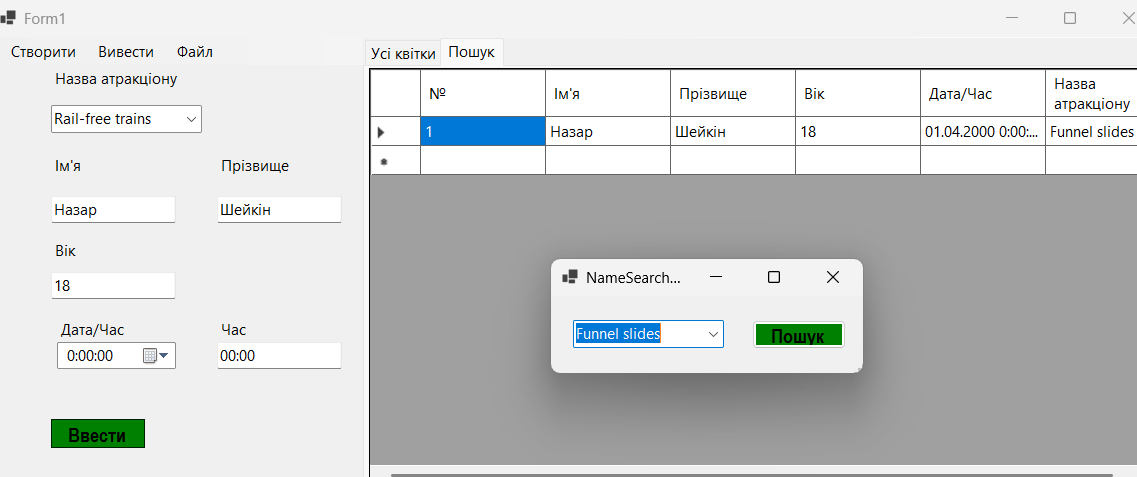
****

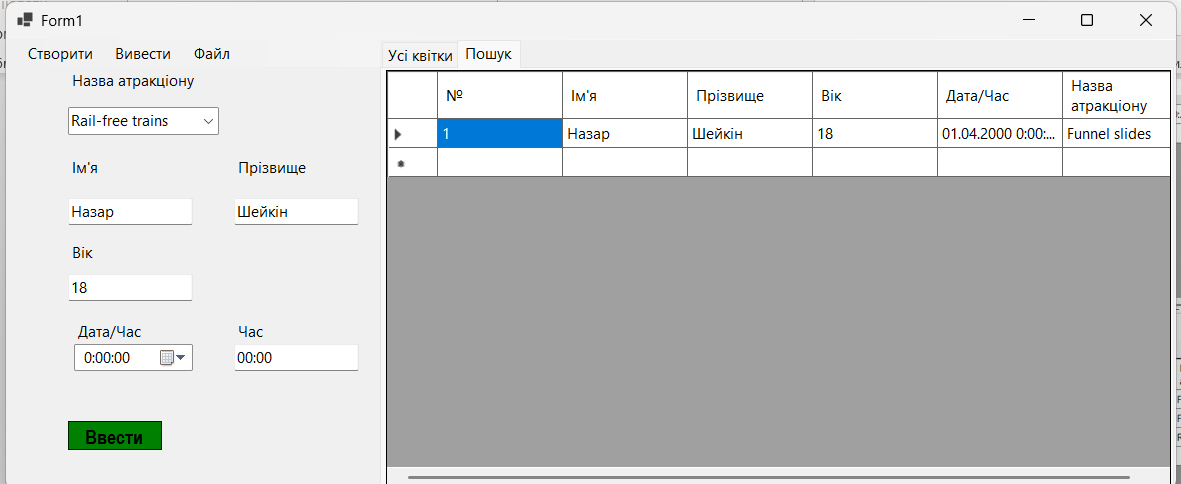
****

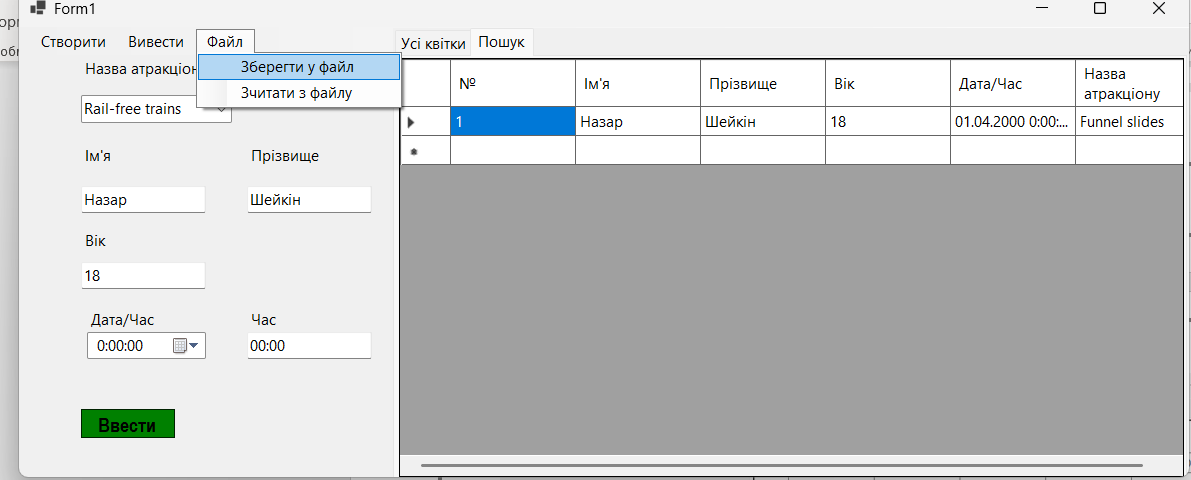
****

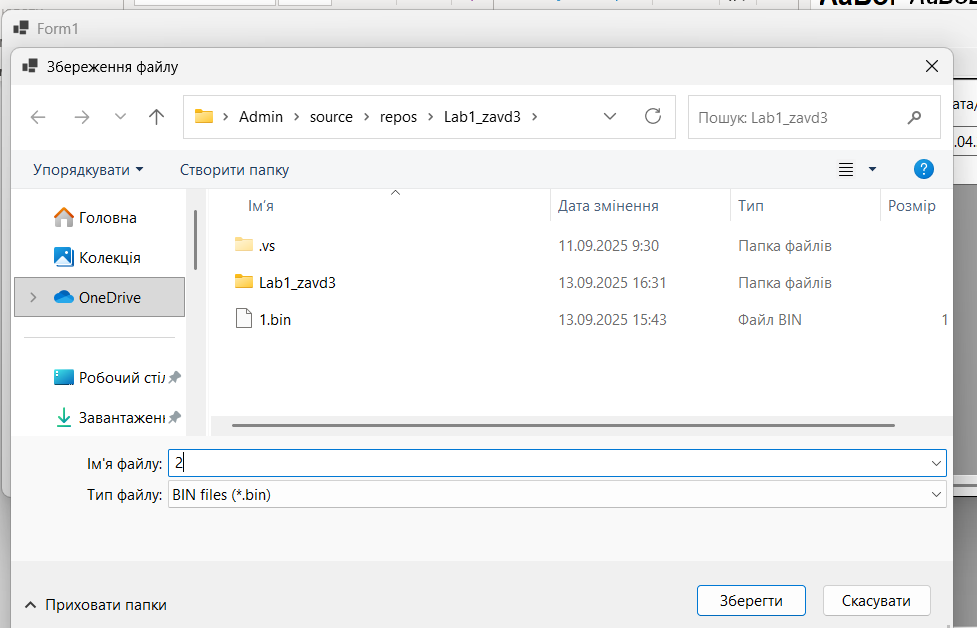
****

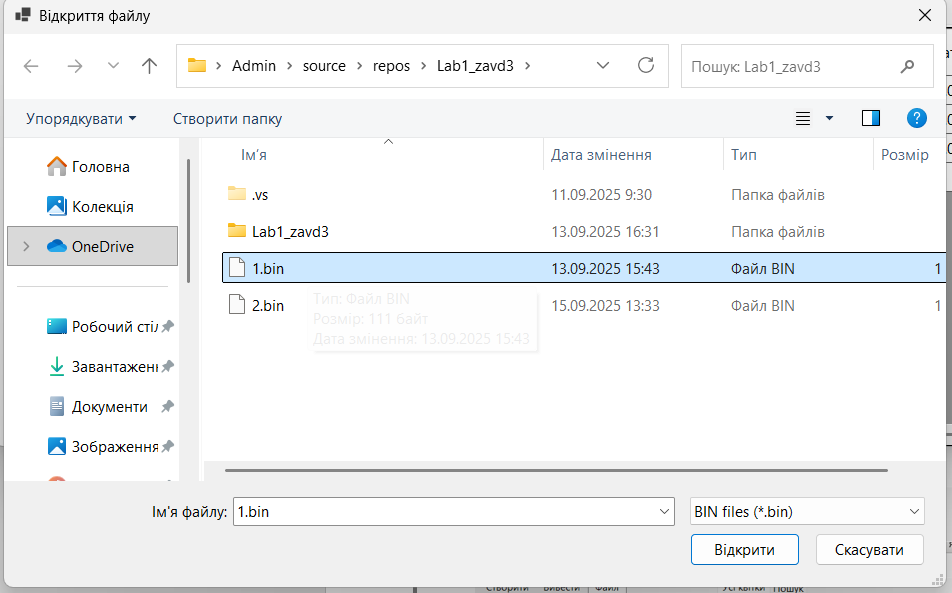
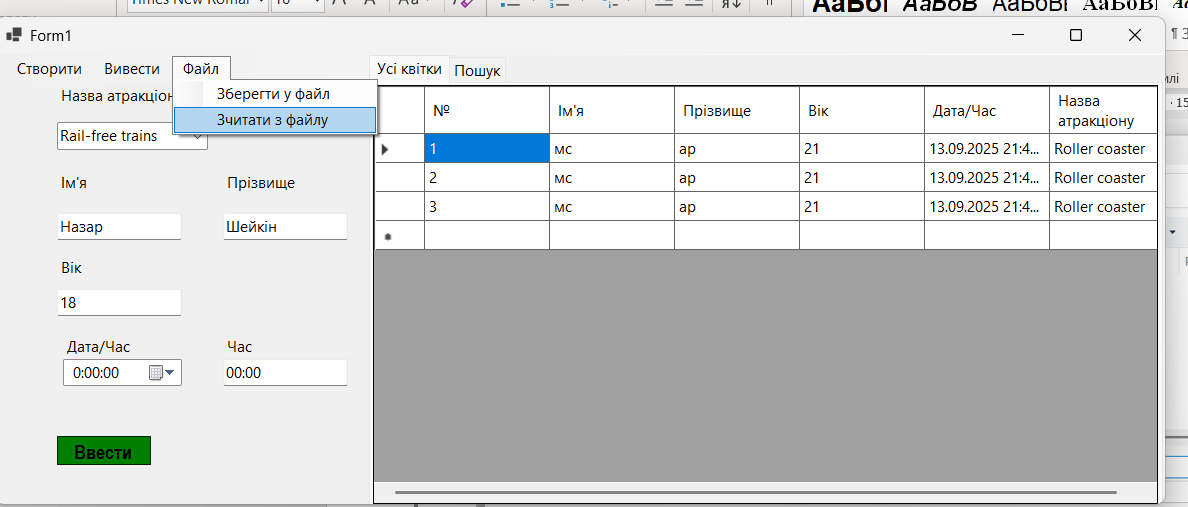
****

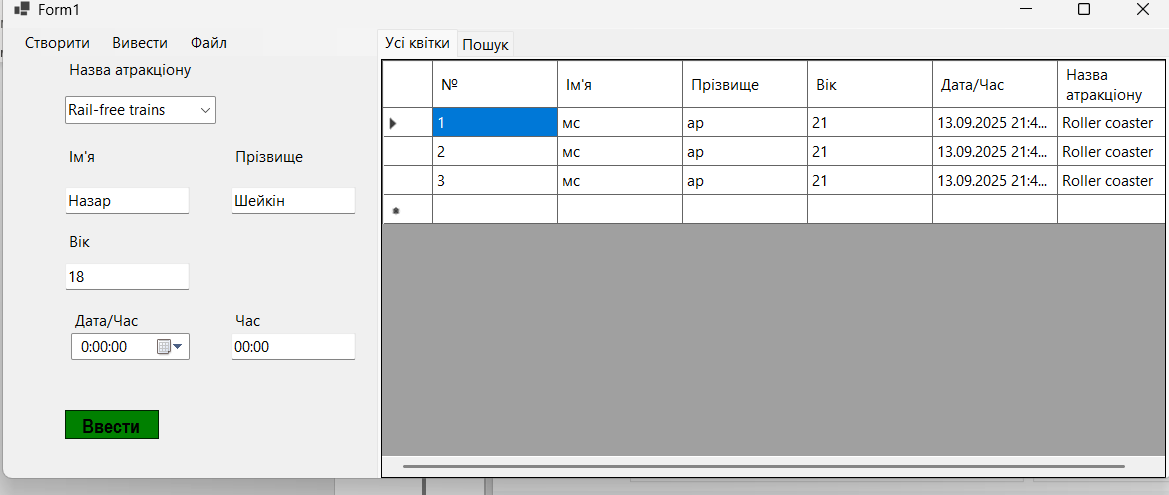
****

****

****

****

****

****

**Висновки:** Виконавши лабораторну роботу, я набув практичних навиків програмування, тестування програм з використанням класів та об’єктів на мовах С++ та C#.