**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**

1. В клас Rectangle додати:
   * Перевантаження:
     + операції ++ (--): одночасно збільшує (зменшує) значення полів а і b;
     + констант true і false: звернення до екземпляра класу дає значення true, якщо прямокутник із заданими довжинами сторін є квадратом, інакше false;
     + операції \*: одночасно домножает поля а і b на скаляр;
     + перетворення типу Rectangle в string (і навпаки).

**Порядок виконання роботи.**

**Підключення бібліотек**

* Заголовковий файл Rectangle.h описує клас Rectangle.
* У Rectangle.cpp реалізовані методи класу.
* У main.cpp — основна програма.

**Оголошення класу Rectangle**

* Приватні поля: a\_, b\_ (сторони прямокутника).
* Конструктори (за замовчуванням, з параметрами, копіювання).
* Деструктор.
* Геттери та сеттери для доступу/зміни сторін.
* Методи perimeter(), area(), Init(), Print().
* Перевантаження операторів ++, --, \*, а також явне приведення до bool (перевірка на квадрат).

**Робота конструктора та методів**

* При створенні об’єкта викликається конструктор (ініціалізація сторін).
* Init() дозволяє ввести значення сторін з клавіатури.
* set\_a() та set\_b() перевіряють коректність введених даних (сторони > 0).
* perimeter() і area() обчислюють відповідно периметр та площу.
* Print() виводить дані прямокутника у табличному форматі.
* Деструктор ~Rectangle() повідомляє про видалення об’єкта.

**Перевантажені оператори**

* ++obj та obj++ — збільшують сторони на 1.
* --obj та obj-- — зменшують сторони на 1 (з перевіркою, що б вони не були ≤ 0).
* obj \* scalar — множить сторони на число.
* (bool)obj — повертає true, якщо прямокутник є квадратом.

**Виконання основної програми (main)**

* Користувач задає кількість прямокутників N.
* Динамічно створюється масив Rectangle\* r1.
* Для кожного прямокутника викликається Init().
* Далі запускається меню з вибором дій:
* Вивести всі прямокутники (таблиця з параметрами).
* Зробити копію обраного прямокутника.
* Домножити всі прямокутники на скаляр.
* Застосувати інкремент (збільшення сторін).
* Застосувати декремент (зменшення сторін).
* Вихід з програми.
* Вибір користувача обробляється у switch.
* При завершенні програми пам’ять масиву звільняється (delete[]).

**Обробка помилок**

* Використовується try/catch для перехоплення винятків (invalid\_argument), якщо сторони ≤ 0 або неправильне введення.

**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;

void Init();

void Print(bool T) const;

// Перевантаження операторів

Rectangle operator++();

Rectangle operator++(int);

Rectangle operator--();

Rectangle operator--(int);

Rectangle operator\*(int scalar) const;

explicit operator bool() const noexcept { return a\_ == b\_; }

};

**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(bool T) 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) << (T ? "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;

}

// Перевантаження операторів

Rectangle Rectangle::operator++() {

++a\_;

++b\_;

return \*this;

}

Rectangle Rectangle::operator++(int) {

Rectangle temp = \*this;

a\_++;

b\_++;

return temp;

}

Rectangle Rectangle::operator--() {

--a\_;

--b\_;

if (a\_ <= 0 || b\_ <= 0) throw invalid\_argument("Сторони не можуть бути <= 0");

return \*this;

}

Rectangle Rectangle::operator--(int) {

Rectangle temp = \*this;

a\_--;

b\_--;

if (a\_ <= 0 || b\_ <= 0) throw invalid\_argument("Сторони не можуть бути <= 0");

return temp;

}

Rectangle Rectangle::operator\*(int scalar) const {

if (scalar <= 0) throw invalid\_argument("Сторони не можуть бути <= 0");

return Rectangle(a\_ \* scalar, b\_ \* scalar);

}

**Main.cpp**

#include "Rectangle.h"

#include <iostream>

#include <iomanip>

using namespace std;

int main() {

setlocale(0, "ukr");

try {

bool T = false;

int N = 0;

cout << "Введіть скільки чотирикутників ви хочете ввести: ";

cin >> N;

Rectangle\* r1 = new Rectangle[N];

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

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

r1[i].Init();

}

int choise = -1;

while (choise != 0) {

system("cls");

cout << "Меню:" << endl

<< "1) Вивести всі чотирикутники" << endl

<< "2) Зробити копію певного чотирикутника" << endl

<< "3) Домножити всі сторони на скаляр" << endl

<< "4) Застовувати інкремент" << endl

<< "5) Застовувати декримент" << endl

<< "0) Вихід" << 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 square?" << endl;

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

T = (r1[i]) ? true:false;

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

r1[i].Print(T);

}

system("pause");

break;

}

case 2: {

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 square?" << endl;

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

T = (r1[i]) ? true : false;

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

r1[i].Print(T);

}

cout << "Введіть номер чотирикутника, який хочете скопіювати: ";

cin >> k;

if (k < 1 || k > N) {

cout << "Неправильний вибір!" << endl;

system("pause");

break;

}

r2 = r1[k - 1];

cout << "\nСкопійований чотирикутник:\n";

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 square?" << endl;

T = (r2) ? true : false;

cout << setw(5) << 1;

r2.Print(T);

system("pause");

break;

}

case 3:

cout << "Введіть на який скаляр ви хочете домножити\n";

int scalar;

cin >> scalar;

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

r1[i] = r1[i] \* scalar;

}

break;

case 4:

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

r1[i]++;

}

break;

case 5:

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

r1[i]--;

}

case 0:

cout << "Вихід з програми..." << endl;

break;

default:

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

system("pause");

break;

}

}

delete[] r1;

}

catch (const exception& ex) {

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

}

return 0;

}

РЕЗУЛЬТАТ ПРОГРАМИ

**1)** Введ?ть ск?льки чотирикутник?в ви хочете ввести: 3

1) Введ?ть сторону а

-2

Введ?ть сторону b

4

Помилка: Сторона a має бути > 0

**2)** Введ?ть ск?льки чотирикутник?в ви хочете ввести: 1

1) Введ?ть сторону а

2

Введ?ть сторону b

-4

Помилка: Сторона b має бути > 0

**3)** Введ?ть ск?льки чотирикутник?в ви хочете ввести: 3

1) Введ?ть сторону а

2

Введ?ть сторону b

4

2) Введ?ть сторону а

5

Введ?ть сторону b

5

3) Введ?ть сторону а

6

Введ?ть сторону b

8

Меню:

1) Вивести вс? чотирикутники

2) Зробити коп?ю певного чотирикутника

3) Домножити вс? сторони на скаляр

4) Застовувати ?нкремент

5) Застовувати декримент

0) Вих?д

1

№ | a | b | P | S |Is square?

1 | 2 | 4 | 12 | 8 | Н?

2 | 5 | 5 | 20 | 25 | Так

3 | 6 | 8 | 28 | 48 | Н?

Press any key to continue . . .

Меню:

1) Вивести вс? чотирикутники

2) Зробити коп?ю певного чотирикутника

3) Домножити вс? сторони на скаляр

4) Застовувати ?нкремент

5) Застовувати декримент

0) Вих?д

2

№ | a | b | P | S |Is square?

1 | 2 | 4 | 12 | 8 | Н?

2 | 5 | 5 | 20 | 25 | Так

3 | 6 | 8 | 28 | 48 | Н?

Введ?ть номер чотирикутника, який хочете скоп?ювати: 3

Скоп?йований чотирикутник:

№ | a | b | P | S |Is square?

1 | 6 | 8 | 28 | 48 | Н?

Press any key to continue . . .

Меню:

1) Вивести вс? чотирикутники

2) Зробити коп?ю певного чотирикутника

3) Домножити вс? сторони на скаляр

4) Застовувати ?нкремент

5) Застовувати декримент

0) Вих?д

3

Введ?ть на який скаляр ви хочете домножити

2

Меню:

1) Вивести вс? чотирикутники

2) Зробити коп?ю певного чотирикутника

3) Домножити вс? сторони на скаляр

4) Застовувати ?нкремент

5) Застовувати декримент

0) Вих?д

1

№ | a | b | P | S |Is square?

1 | 4 | 8 | 24 | 32 | Н?

2 | 10 | 10 | 40 | 100 | Так

3 | 12 | 16 | 56 | 192 | Н?

Press any key to continue . . .

Меню:

1) Вивести вс? чотирикутники

2) Зробити коп?ю певного чотирикутника

3) Домножити вс? сторони на скаляр

4) Застовувати ?нкремент

5) Застовувати декримент

0) Вих?д

3

Введ?ть на який скаляр ви хочете домножити

-1

Помилка: Сторони не можуть бути <= 0

Меню:

1) Вивести вс? чотирикутники

2) Зробити коп?ю певного чотирикутника

3) Домножити вс? сторони на скаляр

4) Застовувати ?нкремент

5) Застовувати декримент

0) Вих?д

1

№ | a | b | P | S |Is square?

1 | 11 | 31 | 84 | 341 | Н?

2 | 26 | 26 | 104 | 676 | Так

3 | 21 | 46 | 134 | 966 | Н?

Press any key to continue . . .

Меню:

1) Вивести вс? чотирикутники

2) Зробити коп?ю певного чотирикутника

3) Домножити вс? сторони на скаляр

4) Застовувати ?нкремент

5) Застовувати декримент

0) Вих?д

4

Меню:

1) Вивести вс? чотирикутники

2) Зробити коп?ю певного чотирикутника

3) Домножити вс? сторони на скаляр

4) Застовувати ?нкремент

5) Застовувати декримент

0) Вих?д

1

№ | a | b | P | S |Is square?

1 | 12 | 32 | 88 | 384 | Н?

2 | 27 | 27 | 108 | 729 | Так

3 | 22 | 47 | 138 | 1034 | Н?

Press any key to continue . . .

Меню:

1) Вивести вс? чотирикутники

2) Зробити коп?ю певного чотирикутника

3) Домножити вс? сторони на скаляр

4) Застовувати ?нкремент

5) Застовувати декримент

0) Вих?д

5

Меню:

1) Вивести вс? чотирикутники

2) Зробити коп?ю певного чотирикутника

3) Домножити вс? сторони на скаляр

4) Застовувати ?нкремент

5) Застовувати декримент

0) Вих?д

1

№ | a | b | P | S |Is square?

1 | 11 | 31 | 84 | 341 | Н?

2 | 26 | 26 | 104 | 676 | Так

3 | 21 | 46 | 134 | 966 | Н?

Press any key to continue . . .

Меню:

1) Вивести вс? чотирикутники

2) Зробити коп?ю певного чотирикутника

3) Домножити вс? сторони на скаляр

4) Застовувати ?нкремент

5) Застовувати декримент

0) Вих?д

0

Вих?д з програми...

Видаленно масив даних

Видаленно масив даних

Видаленно масив даних

D:\VisualStudio\ООП\LR2\LR2\_zavd1\x64\Debug\LR2\_zavd1.exe (process 32824) exited with code 0 (0x0).

Press any key to close this window . . .

**Задача 2**

**Умови**

Добавити перевантаження в 3 задачу перщої лабороторної та реалізувати їх. Переконатись, що перевантаження працюють вірно та виконують свою функцію.

**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:**

**Tickets.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace LR1\_Zavd3

{

internal class Ticket

{

//Test2

//--------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;

}

//--------Перевантаження----------

// == і != (порівняння по всіх полях)

public static bool operator ==(Ticket t1, Ticket t2)

{

if (ReferenceEquals(t1, t2)) return true;

if ((object)t1 == null || (object)t2 == null) return false;

return t1.FirstName == t2.FirstName &&

t1.LastName == t2.LastName &&

t1.NameAttraction == t2.NameAttraction &&

t1.Age == t2.Age &&

t1.DateTime == t2.DateTime;

}

public static bool operator !=(Ticket t1, Ticket t2) => !(t1 == t2);

// > і < (порівняння за віком)

public static bool operator >(Ticket t1, Ticket t2) => t1.Age > t2.Age;

public static bool operator <(Ticket t1, Ticket t2) => t1.Age < t2.Age;

// Додавання (зміщення дати на кілька днів)

public static Ticket operator +(Ticket t, int days)

{

return new Ticket(t.FirstName, t.LastName, t.NameAttraction, t.Age, t.DateTime.AddDays(days));

}

// ToString() для гарного виводу

public override string ToString()

{

return $"{FirstName} {LastName}, {Age} років, Атракціон: {NameAttraction}, Дата: {DateTime}";

}

// Щоб не сварився компілятор, треба перевизначити Equals і GetHashCode

public override bool Equals(object obj)

{

if (obj is Ticket other) return this == other;

return false;

}

public override int GetHashCode()

{

return HashCode.Combine(FirstName, LastName, NameAttraction, Age, DateTime);

}

}

}

**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!");

}

}

}

private void UpdateDataGrid()

{

dataGridView1.Rows.Clear();

count = 0;

foreach (var ticket in tickets)

{

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;

}

}

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

{

tickets = tickets.OrderBy(t => t.Age).ToList();

UpdateDataGrid();

}

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

{

tickets = tickets.OrderByDescending(t => t.Age).ToList();

UpdateDataGrid();

}

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

{

tickets = tickets.OrderBy(t => t.FirstName).ToList();

UpdateDataGrid();

}

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

{

tickets = tickets.OrderByDescending(t => t.FirstName).ToList();

UpdateDataGrid();

}

private void відАДоЯToolStripMenuItem1\_Click(object sender, EventArgs e)

{

tickets = tickets.OrderBy(t => t.LastName).ToList();

UpdateDataGrid();

}

private void відЯДоАToolStripMenuItem1\_Click(object sender, EventArgs e)

{

tickets = tickets.OrderByDescending(t => t.LastName).ToList();

UpdateDataGrid();

}

}

}

**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**

namespace Lab1\_zavd3

{

partial class CopyTicketForm

{

/// <summary>

/// Required designer variable.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Clean up any resources being used.

/// </summary>

/// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

private void InitializeComponent()

{

button1 = new Button();

numericUpDown1 = new NumericUpDown();

((System.ComponentModel.ISupportInitialize)numericUpDown1).BeginInit();

SuspendLayout();

//

// button1

//

button1.BackColor = Color.Red;

button1.Enabled = false;

button1.FlatStyle = FlatStyle.Popup;

button1.Font = new Font("Arial Narrow", 10.2F, FontStyle.Bold, GraphicsUnit.Point, 204);

button1.Location = new Point(114, 30);

button1.Name = "button1";

button1.Size = new Size(134, 29);

button1.TabIndex = 0;

button1.Text = "Зробити копію";

button1.UseVisualStyleBackColor = false;

button1.Click += button1\_Click;

//

// numericUpDown1

//

numericUpDown1.Location = new Point(12, 30);

numericUpDown1.Name = "numericUpDown1";

numericUpDown1.Size = new Size(77, 27);

numericUpDown1.TabIndex = 2;

numericUpDown1.ValueChanged += numericUpDown1\_ValueChanged;

//

// CopyTicketForm

//

AutoScaleDimensions = new SizeF(8F, 20F);

AutoScaleMode = AutoScaleMode.Font;

ClientSize = new Size(260, 81);

Controls.Add(numericUpDown1);

Controls.Add(button1);

Name = "CopyTicketForm";

Text = "CopyTicketForm";

((System.ComponentModel.ISupportInitialize)numericUpDown1).EndInit();

ResumeLayout(false);

}

#endregion

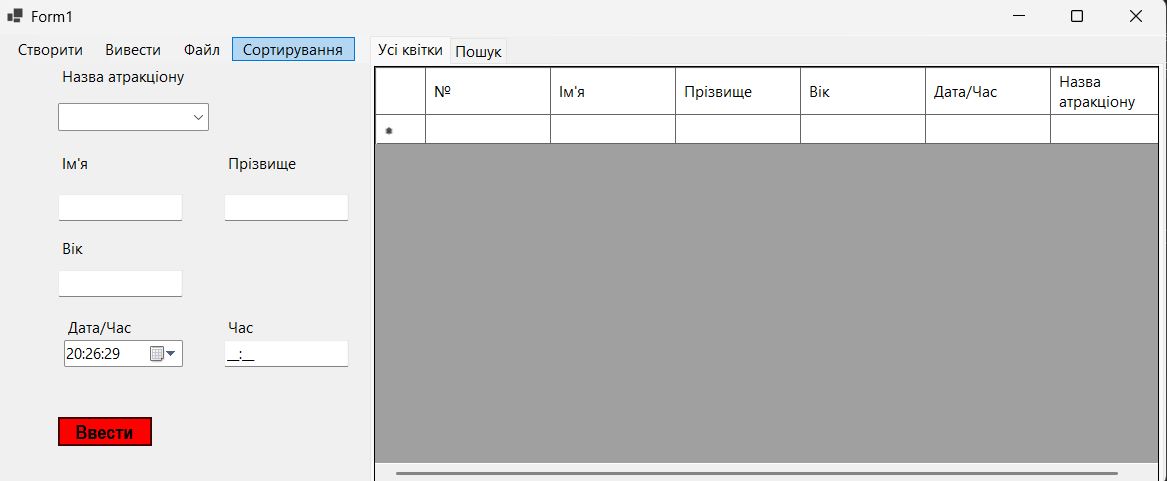
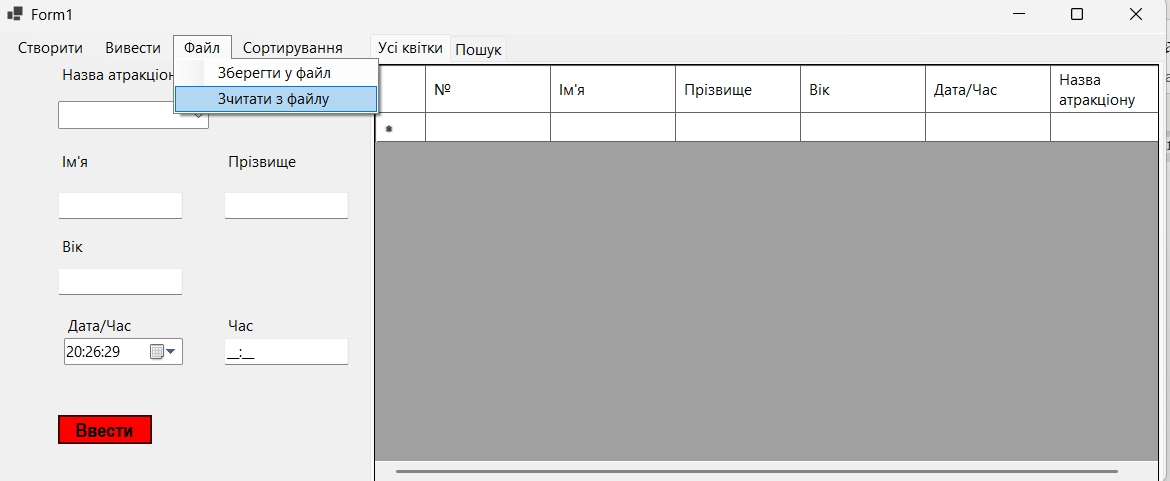
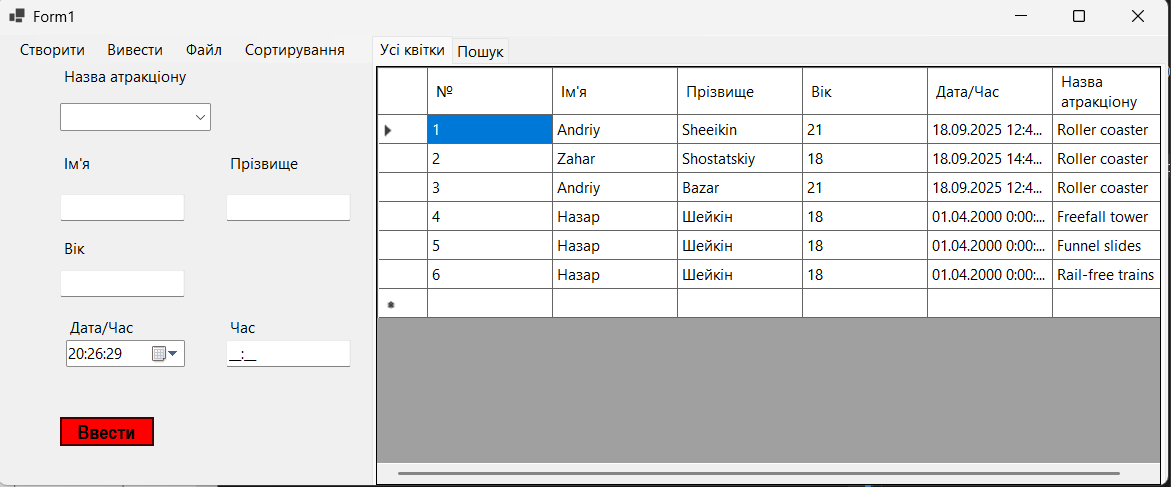
private Button button1;

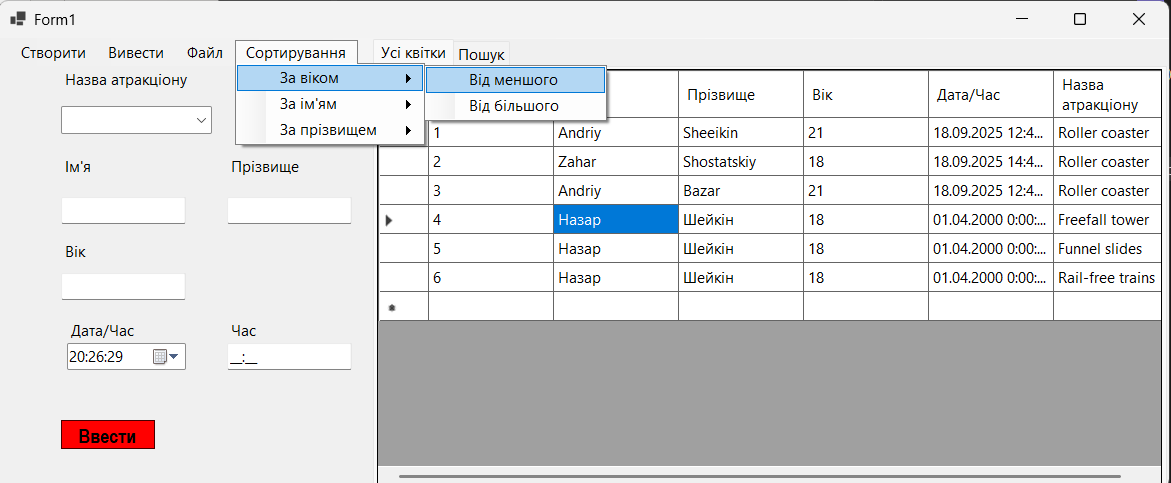
private NumericUpDown numericUpDown1;

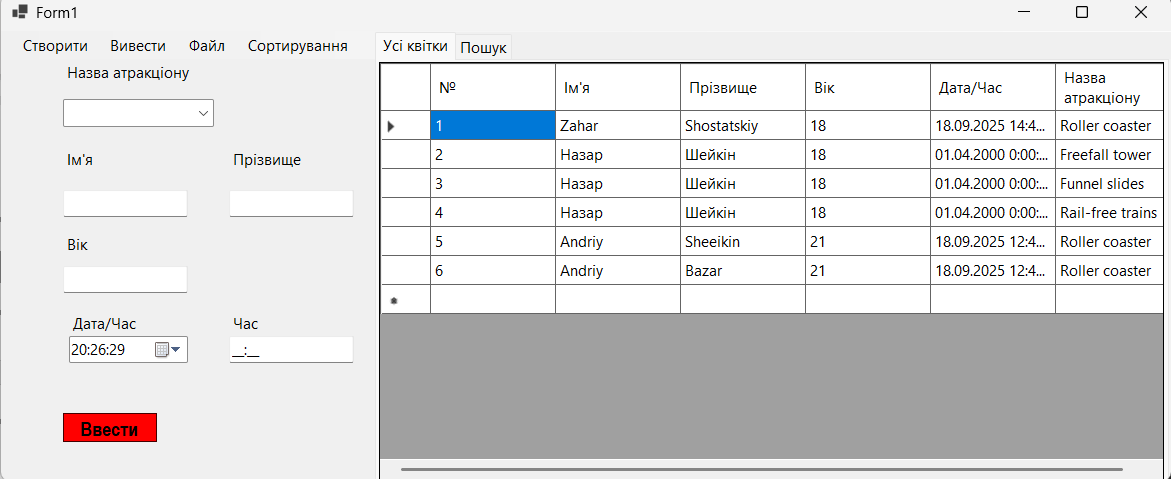
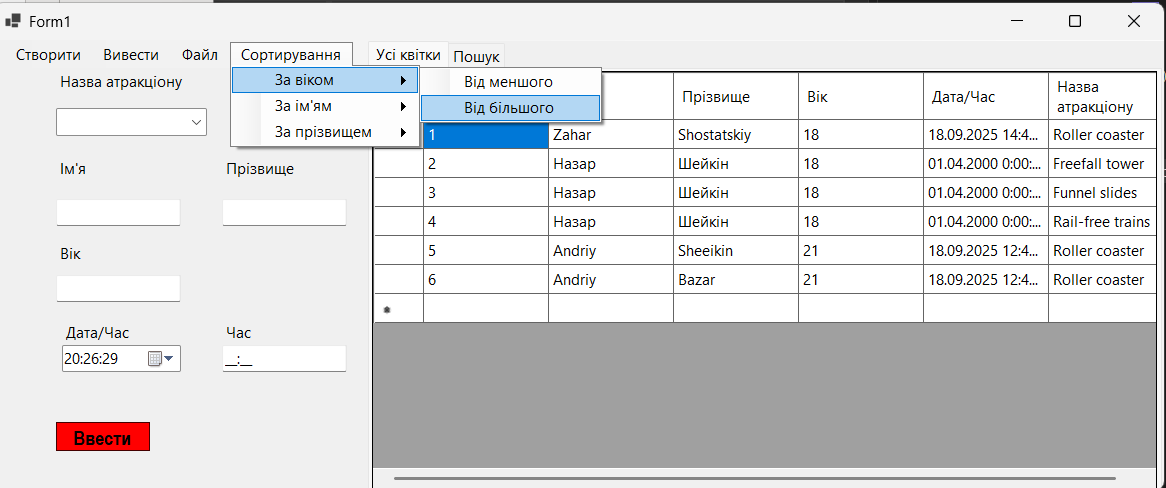
}

}

**РЕЗУЛЬТАТ ПРОГРАМИ**

**** **** 

****

**** **** 