**Лабораторна робота №2. Перевантаження операцій.**

**Мета.** Одержати практичні навички створення абстрактних типів даних і перевантаження операцій у мові С++.

**Постановка задачі:**Розширити функціональність класу, реалізовану в попередній роботі, згідно тогож варіанту.

**Варіанти завдань для виконання роботи.**

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 домножає поля а і b на скаляр;
     + перетворення типу Rectangle в string (і навпаки).

**Код програми:**

**Rectangle.h**

#pragma once

#include <iostream>

#include <string>

using namespace std;

class Rectangle {

private:

int a, b;

public:

// Конструктори

Rectangle();

Rectangle(int s1, int s2);

Rectangle(const Rectangle& c);

~Rectangle();

// Методи

void printSides();

int calculatePerimeter();

int calculateArea();

// Властивості

int getA();

int getB();

void setA(int s);

void setB(int s);

bool isSquare();

// Оператори

Rectangle& operator++();

Rectangle operator++(int);

Rectangle& operator--();

Rectangle operator--(int);

explicit operator bool() const;

Rectangle operator\*(int scalar) const;

operator string() const;

};

Rectangle operator"" \_rect(const char\* str, size\_t);

**Rectangle.cpp**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include "Rectangle.h"

// Конструктор за замовчуванням

Rectangle::Rectangle() : a(0), b(0) {

cout << "Створено прямокутник за замовчуванням: " << a << " x " << b << " (" << this << ")" << endl;

}

// Конструктор з параметрами

Rectangle::Rectangle(int s1, int s2) : a(s1), b(s2) {

cout << "Створено прямокутник: " << a << " x " << b << " (" << this << ")" << endl;

}

// Конструктор копіювання

Rectangle::Rectangle(const Rectangle& c) : a(c.a), b(c.b) {

cout << "Викликано конструктор копiювання: " << a << " x " << b << " (" << this << ")" << endl;

}

// Деструктор

Rectangle::~Rectangle() {

cout << "Викликано деструктор для прямокутника: " << a << " x " << b << " (" << this << ")" << endl;

}

// Методи

void Rectangle::printSides() {

cout << "Сторони прямокутника: a = " << a << ", b = " << b << " (" << this << ")" << endl;

}

int Rectangle::calculatePerimeter() {

return 2 \* (a + b);

}

int Rectangle::calculateArea() {

return a \* b;

}

// Властивості

int Rectangle::getA() { return a; }

int Rectangle::getB() { return b; }

void Rectangle::setA(int s) { a = s; }

void Rectangle::setB(int s) { b = s; }

bool Rectangle::isSquare() { return a == b; }

// ++ префіксний

Rectangle& Rectangle::operator++() {

++a; ++b;

return \*this;

}

// ++ постфіксний

Rectangle Rectangle::operator++(int) {

Rectangle temp(\*this);

++(\*this);

return temp;

}

// -- префіксний

Rectangle& Rectangle::operator--() {

--a; --b;

return \*this;

}

// -- постфіксний

Rectangle Rectangle::operator--(int) {

Rectangle temp(\*this);

--(\*this);

return temp;

}

// "true" / "false"

Rectangle::operator bool() const {

return a == b;

}

// множення на скаляр

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

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

}

// перетворення в string

Rectangle::operator string() const {

return "Прямокутник: " + to\_string(a) + " x " + to\_string(b);

}

// перетворення зі string (літерал)

Rectangle operator"" \_rect(const char\* str, size\_t) {

int s1 = 0, s2 = 0;

// підтримка "7 4"

if (sscanf(str, "%d %d", &s1, &s2) == 2) {

return Rectangle(s1, s2);

}

// підтримка "7x4"

if (sscanf(str, "%dx%d", &s1, &s2) == 2) {

return Rectangle(s1, s2);

}

// підтримка "7,4"

if (sscanf(str, "%d,%d", &s1, &s2) == 2) {

return Rectangle(s1, s2);

}

// одне число -> квадрат

if (sscanf(str, "%d", &s1) == 1) {

return Rectangle(s1, s1);

}

return Rectangle();

}

**Main.cpp**

#include "Rectangle.h"

#include <iostream>

using namespace std;

int main() {

setlocale(0, "ukr");

// 1. Конструктор за замовчуванням

Rectangle r1;

r1.printSides();

// 2. Конструктор з параметрами

Rectangle r2(10, 10);

r2.printSides();

// 3. Конструктор копіювання

Rectangle r3 = r2;

r3.printSides();

Rectangle rectArr[3] = {

Rectangle(),

Rectangle(8, 2),

Rectangle(r2)};

for (int i = 0; i < 3; i++) {

cout << "\nПрямокутник " << i + 1 << ":\n";

rectArr[i].printSides();

cout << "Периметр = " << rectArr[i].calculatePerimeter() << endl;

cout << "Площа = " << rectArr[i].calculateArea() << endl;

cout << (rectArr[i].isSquare() ? "Це квадрат" : "Не квадрат") << endl;

}

// 5. Покажчик на компонент-функцію

typedef void (Rectangle::\* FuncPointer)();

FuncPointer fp = &Rectangle::printSides;

// 6. Покажчик на об'єкт

Rectangle\* pRect = &r2;

// 8. Виклик методу через обидва покажчики

(pRect->\*fp)();

cout << "\nТест операторів:\n";

cout << "До ++: "; r1.printSides();

++r1;

cout << "Пiсля ++: "; r1.printSides();

cout << "До --: "; r1.printSides();

r1--;

cout << "Пiсля --: "; r1.printSides();

cout << (r2 ? "r2 - квадрат\n" : "r2 - не квадрат\n");

Rectangle r4 = r2 \* 3;

cout << "Множення r2\*3: "; r4.printSides();

string s = (string)r4;

cout << "У string: " << s << endl;

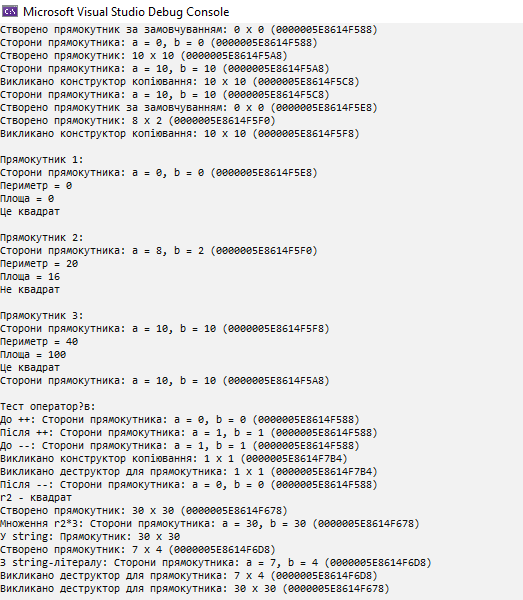
Rectangle r5 = "7 4"\_rect;

cout << "З string-лiтералу: "; r5.printSides();

return 0;

}

**Результат програми:**

****

**Задача №2**

**Умова задачі:**Перевантаження операторів потоку та присвоєння

**Код програми:**

**Rectangle.h**

#pragma once

#include <iostream>

#include <iomanip>

#include <string>

using namespace std;

class Rectangle {

private:

int a, b;

public:

// Конструктори

Rectangle();

Rectangle(int s1, int s2);

Rectangle(const Rectangle& c);

~Rectangle();

// Методи

void printSides();

int calculatePerimeter() const;

int calculateArea() const;

// Властивості

int getA() const;

int getB() const;

void setA(int s);

void setB(int s);

bool isSquare() const;

// Оператори

Rectangle& operator=(const Rectangle& other); // оператор присвоєння

Rectangle& operator++();

Rectangle operator++(int);

Rectangle& operator--();

Rectangle operator--(int);

explicit operator bool() const;

Rectangle operator\*(int scalar) const;

operator string() const;

// Друзі

friend ostream& operator<<(ostream& os, const Rectangle& r);

friend istream& operator>>(istream& is, Rectangle& r);

};

Rectangle operator"" \_rect(const char\* str, size\_t);

**Rectangle.cpp**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include "Rectangle.h"

// Конструктори

Rectangle::Rectangle() : a(0), b(0) {

cout << "Створено прямокутник за замовчуванням: " << a << " x " << b << " (" << this << ")" << endl;

}

Rectangle::Rectangle(int s1, int s2) : a(s1), b(s2) {

cout << "Створено прямокутник: " << a << " x " << b << " (" << this << ")" << endl;

}

Rectangle::Rectangle(const Rectangle& c) : a(c.a), b(c.b) {

cout << "Викликано конструктор копiювання: " << a << " x " << b << " (" << this << ")" << endl;

}

Rectangle::~Rectangle() {

cout << "Викликано деструктор для прямокутника: " << a << " x " << b << " (" << this << ")" << endl;

}

// Методи

void Rectangle::printSides() {

cout << "Сторони прямокутника: a = " << a << ", b = " << b << " (" << this << ")" << endl;

}

int Rectangle::calculatePerimeter() const {

return 2 \* (a + b);

}

int Rectangle::calculateArea() const {

return a \* b;

}

// Властивості

int Rectangle::getA() const { return a; }

int Rectangle::getB() const { return b; }

void Rectangle::setA(int s) { a = s; }

void Rectangle::setB(int s) { b = s; }

bool Rectangle::isSquare() const { return a == b; }

// Оператор присвоєння

Rectangle& Rectangle::operator=(const Rectangle& other) {

if (this != &other) {

a = other.a;

b = other.b;

}

return \*this;

}

// ++ префіксний

Rectangle& Rectangle::operator++() {

++a; ++b;

return \*this;

}

// ++ постфіксний

Rectangle Rectangle::operator++(int) {

Rectangle temp(\*this);

++(\*this);

return temp;

}

// -- префіксний

Rectangle& Rectangle::operator--() {

--a; --b;

return \*this;

}

// -- постфіксний

Rectangle Rectangle::operator--(int) {

Rectangle temp(\*this);

--(\*this);

return temp;

}

// "true" / "false"

Rectangle::operator bool() const {

return a == b;

}

// множення на скаляр

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

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

}

// перетворення в string

Rectangle::operator string() const {

return "Прямокутник: " + to\_string(a) + " x " + to\_string(b);

}

// Потоки

ostream& operator<<(ostream& os, const Rectangle& r) {

os << left

<< setw(12) << (to\_string(r.a) + " x " + to\_string(r.b))

<< setw(10) << r.calculatePerimeter()

<< setw(10) << r.calculateArea()

<< (r.isSquare() ? "Квадрат" : "Прямокутник");

return os;

}

istream& operator>>(istream& is, Rectangle& r) {

cout << "Введiть сторону a: ";

is >> r.a;

cout << "Введiть сторону b: ";

is >> r.b;

return is;

}

// перетворення зі string (літерал)

Rectangle operator"" \_rect(const char\* str, size\_t) {

int s1 = 0, s2 = 0;

if (sscanf(str, "%d %d", &s1, &s2) == 2) return Rectangle(s1, s2);

if (sscanf(str, "%dx%d", &s1, &s2) == 2) return Rectangle(s1, s2);

if (sscanf(str, "%d,%d", &s1, &s2) == 2) return Rectangle(s1, s2);

if (sscanf(str, "%d", &s1) == 1) return Rectangle(s1, s1);

return Rectangle();

}

**Main.cpp**

#include "Rectangle.h"

#include <iostream>

#include <iomanip>

using namespace std;

int main() {

setlocale(0, "ukr");

// 1. Створення прямокутників

Rectangle r1; // За замовчуванням

Rectangle r2(10, 10); // З параметрами

Rectangle r3 = r2; // Копіювання

// 2. Оператори

Rectangle r4 = r1; r4++; // Постфіксний ++

Rectangle r5 = r1; ++r5; // Префіксний ++

Rectangle r6 = r2 \* 3; // Множення на скаляр

Rectangle r7 = r2; r7--; // Постфіксний --

Rectangle r8 = r2; --r8; // Префіксний --

// 3. Ввід користувача

Rectangle rUser;

cin >> rUser;

// 4. Вивід у таблицю

cout << "\nТаблиця прямокутникiв:\n";

cout << "№ " << left << setw(12) << "Сторони"<< setw(10) << "Периметр"<< setw(10) << "Площа"<< "Тип" << endl;

cout << "---------------------------------------------------\n";

cout << 1 << " " << r1 << " (r1)\n";

cout << 2 << " " << r2 << " (r2)\n";

cout << 3 << " " << r3 << " (r3 = r2)\n";

cout << 4 << " " << r4 << " (r1++)\n";

cout << 5 << " " << r5 << " (++r1)\n";

cout << 6 << " " << r6 << " (r2 \* 3)\n";

cout << 7 << " " << r7 << " (r2--)\n";

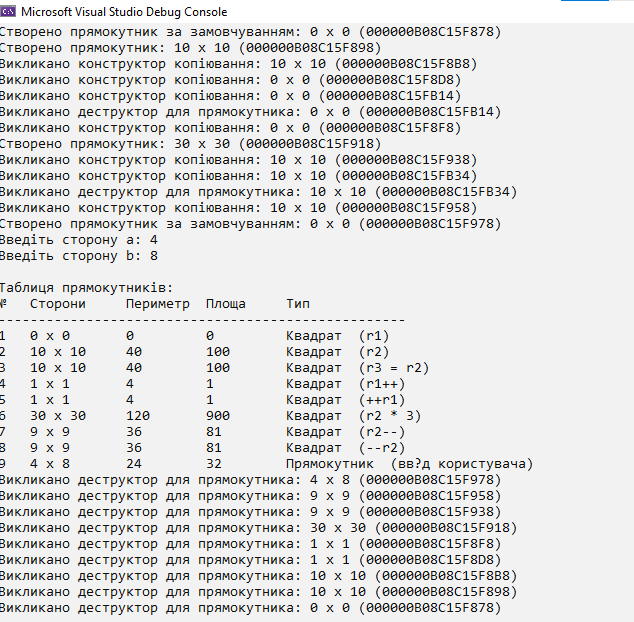
cout << 8 << " " << r8 << " (--r2)\n";

cout << 9 << " " << rUser << " (ввід користувача)\n";

return 0;

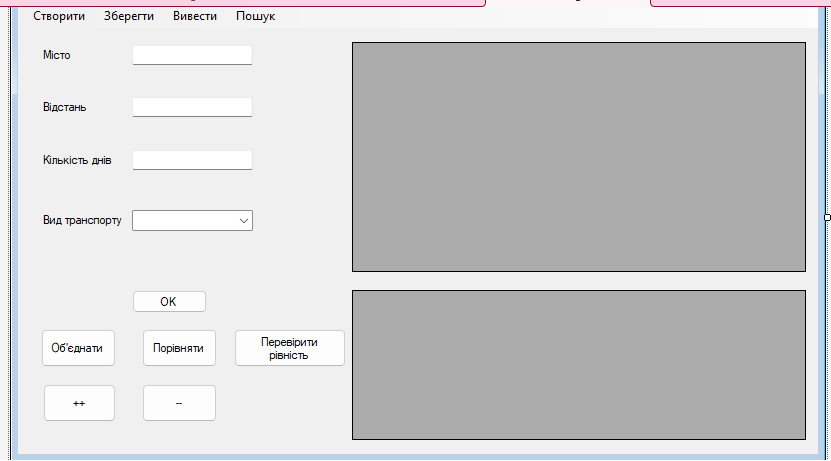
}

**Результат програми:**

****

**Задача №3**

**Умова задачі:** Перевантажити оператори в С#



Форма складається з 3 textBox, 1 comboBox, 6 button, 2 datagridview

**Код програми:**

**Class1.cs**

using System;

namespace LR1\_3

{

public class Podorozh

{

public string Misto { get; set; }

public double Vidstan { get; set; }

public int Dni { get; set; }

public string Transport { get; set; }

public Podorozh()

{

Misto = "Львів";

Vidstan = 150;

Dni = 1;

Transport = "Автобус";

}

public Podorozh(string misto, double vidstan, int dni, string transport)

{

Misto = misto;

Vidstan = vidstan;

Dni = dni;

Transport = transport;

}

public double RozrakhVartist()

{

double tsinaZaKm = 5;

double tsinaZaDen = 200;

return (Vidstan \* tsinaZaKm) + (Dni \* tsinaZaDen);

}

public double SerednyaTsinaKm()

{

return RozrakhVartist() / Vidstan;

}

public string Info()

{

return $"Місто: {Misto}, Відстань: {Vidstan} км, Днів: {Dni}, Транспорт: {Transport}, Вартість: {RozrakhVartist()} грн";

}

// Додавання двох подорожей -> нова подорож з сумою відстаней та днів

public static Podorozh operator +(Podorozh a, Podorozh b)

{

return new Podorozh(

a.Misto + "-" + b.Misto, // нова назва (об’єднані міста)

a.Vidstan + b.Vidstan, // сумарна відстань

a.Dni + b.Dni, // сумарні дні

a.Transport // лишаємо транспорт від першої (можна інакше)

);

}

// Порівняння за вартістю (більше/менше)

public static bool operator >(Podorozh a, Podorozh b)

{

return a.RozrakhVartist() > b.RozrakhVartist();

}

public static bool operator <(Podorozh a, Podorozh b)

{

return a.RozrakhVartist() < b.RozrakhVartist();

}

public static bool operator ==(Podorozh a, Podorozh b)

{

// Якщо обидва null — вони рівні

if (ReferenceEquals(a, null) && ReferenceEquals(b, null))

return true;

// Якщо один з них null — не рівні

if (ReferenceEquals(a, null) || ReferenceEquals(b, null))

return false;

// Тепер можна безпечно порівнювати властивості

return a.Misto == b.Misto && a.Transport == b.Transport;

}

public static bool operator !=(Podorozh a, Podorozh b)

{

return !(a == b);

}

// Щоб уникнути попередження при == і !=, треба перевизначити Equals та GetHashCode

public override bool Equals(object obj)

{

if (obj is Podorozh other)

return this == other;

return false;

}

public static Podorozh operator ++(Podorozh a)

{

if (a == null) return a;

a.Dni += 1;

return a;

}

public static Podorozh operator --(Podorozh a)

{

if (a == null) return a;

a.Dni = a.Dni > 1 ? a.Dni - 1 : 1;

return a;

}

public override int GetHashCode()

{

return (Misto + Transport).GetHashCode();

}

}

}

**Form1.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 LR1\_3

{

public partial class Form1 : Form

{

private List<Podorozh> podorozhi = new List<Podorozh>();

public Form1()

{

InitializeComponent();

dataGridView1.AutoGenerateColumns = false;

// Додаємо колонки

dataGridView1.Columns.Add("Misto", "Місто");

dataGridView1.Columns.Add("Vidstan", "Відстань (км)");

dataGridView1.Columns.Add("Dni", "Днів");

dataGridView1.Columns.Add("Transport", "Транспорт"); // нова колонка

dataGridView1.Columns.Add("Vartist", "Вартість");

dataGridView1.Columns.Add("Serednya", "Середня ціна/км");

dataGridView2.AutoGenerateColumns = false;

dataGridView2.Columns.Add("Misto", "Місто");

dataGridView2.Columns.Add("Vidstan", "Відстань (км)");

dataGridView2.Columns.Add("Dni", "Днів");

dataGridView2.Columns.Add("Transport", "Транспорт");

dataGridView2.Columns.Add("Vartist", "Вартість");

dataGridView2.Columns.Add("Serednya", "Середня ціна/км");

// Заповнюємо ComboBox транспортом

comboBox1.Items.AddRange(new string[] { "Автобус", "Поїзд", "Літак", "Авто" });

comboBox1.SelectedIndex = 0; // за замовчуванням "Автобус"

textBox1.KeyPress += TextBox1\_KeyPress;

textBox2.KeyPress += TextBox2\_KeyPress;

textBox3.KeyPress += TextBox3\_KeyPress;

}

// Місто — лише літери та пробіл

private void TextBox1\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!char.IsControl(e.KeyChar) && !char.IsLetter(e.KeyChar) && e.KeyChar != ' ')

{

e.Handled = true; // забороняємо введення

MessageBox.Show("Тільки літери та пробіли дозволені!");

}

}

// Відстань — лише цифри і крапка

private void TextBox2\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar) && e.KeyChar != '.')

{

e.Handled = true;

MessageBox.Show("Тільки цифри та крапка дозволені!");

}

// Не більше однієї крапки

if (e.KeyChar == '.' && (sender as TextBox).Text.Contains("."))

{

e.Handled = true;

}

}

// Дні — тільки цифри

private void TextBox3\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar))

{

e.Handled = true;

MessageBox.Show("Тільки цифри дозволені!");}

}

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

{

try

{

string misto = textBox1.Text;

double vidstan = double.Parse(textBox2.Text);

int dni = int.Parse(textBox3.Text);

string transport = comboBox1.SelectedItem.ToString(); // додаємо транспорт

Podorozh p = new Podorozh(misto, vidstan, dni, transport); // 4 аргументи

podorozhi.Add(p);

dataGridView1.Rows.Add(

p.Misto,

p.Vidstan,

p.Dni,

p.Transport, // нова колонка

p.RozrakhVartist(),

p.SerednyaTsinaKm().ToString("F2")

);

textBox1.Clear();

textBox2.Clear();

textBox3.Clear();

}

catch

{

MessageBox.Show("Будь ласка, введіть коректні дані!");

}

}

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

{

podorozhi.Clear();

dataGridView1.Rows.Clear();

MessageBox.Show("Створено новий список подорожей!");

}

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

{

SaveFileDialog saveFileDialog = new SaveFileDialog();

saveFileDialog.Filter = "Текстові файли|\*.txt";

saveFileDialog.Title = "Збереження даних";

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

{

using (System.IO.StreamWriter sw = new System.IO.StreamWriter(saveFileDialog.FileName))

{

foreach (var p in podorozhi)

{

sw.WriteLine($"{p.Misto};{p.Vidstan};{p.Dni};{p.Transport};{p.RozrakhVartist()};{p.SerednyaTsinaKm():F2}");

}

}

MessageBox.Show("Дані збережено!");

}

}

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

{

OpenFileDialog openFileDialog = new OpenFileDialog();

openFileDialog.Filter = "Текстові файли|\*.txt";

openFileDialog.Title = "Відкрити файл з даними";

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

{

podorozhi.Clear();

dataGridView1.Rows.Clear();

using (System.IO.StreamReader sr = new System.IO.StreamReader(openFileDialog.FileName))

{

string line;

while ((line = sr.ReadLine()) != null)

{

string[] parts = line.Split(';');

// Перевіряємо, що в рядку є хоча б 3 колонки (misto, vidstan, dni)

if (parts.Length >= 3)

{

string misto = parts[0];

double vidstan = double.Parse(parts[1]);

int dni = int.Parse(parts[2]);

// Якщо transport є у файлі, беремо його, інакше дефолт

string transport = parts.Length > 3 ? parts[3] : "Не вказано";

Podorozh p = new Podorozh(misto, vidstan, dni, transport);

podorozhi.Add(p);

// Додаємо в DataGridView, не забуваємо колонку Transport

dataGridView1.Rows.Add(

p.Misto,

p.Vidstan,

p.Dni,

p.Transport,

p.RozrakhVartist(),

p.SerednyaTsinaKm().ToString("F2")

);

}

}

}

MessageBox.Show("Дані завантажено!");

}

}

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

{

Form2 formSearch = new Form2(podorozhi);

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

{

dataGridView2.Rows.Clear();

foreach (var p in formSearch.SearchResults)

{

dataGridView2.Rows.Add(

p.Misto,

p.Vidstan,

p.Dni,

p.Transport,

p.RozrakhVartist(),

p.SerednyaTsinaKm().ToString("F2")

);

}

}

}

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

{

if (dataGridView1.SelectedRows.Count == 2)

{

// Беремо 2 вибрані рядки

int index1 = dataGridView1.SelectedRows[0].Index;

int index2 = dataGridView1.SelectedRows[1].Index;

Podorozh p1 = podorozhi[index1];

Podorozh p2 = podorozhi[index2];

// Використовуємо оператор +

Podorozh combined = p1 + p2;

// Додаємо нову подорож у список і таблицю

podorozhi.Add(combined);

dataGridView1.Rows.Add(

combined.Misto,

combined.Vidstan,

combined.Dni,

combined.Transport,

combined.RozrakhVartist(),

combined.SerednyaTsinaKm().ToString("F2")

);

MessageBox.Show("Подорожі успішно об’єднано!");

}

else

{

MessageBox.Show("Виберіть рівно 2 подорожі для об’єднання!");

}

}

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

{

if (dataGridView1.SelectedRows.Count == 2)

{

int index1 = dataGridView1.SelectedRows[0].Index;

int index2 = dataGridView1.SelectedRows[1].Index;

Podorozh p1 = podorozhi[index1];

Podorozh p2 = podorozhi[index2];

// Використовуємо оператори > і <

if (p1 > p2)

MessageBox.Show($"{p1.Misto} дорожча за {p2.Misto}");

else if (p1 < p2)

MessageBox.Show($"{p2.Misto} дорожча за {p1.Misto}");

else

MessageBox.Show("Подорожі мають однакову вартість!");

}

else

{

MessageBox.Show("Виберіть рівно 2 подорожі для порівняння!");

}

}

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

{

if (dataGridView1.SelectedRows.Count == 2)

{

int index1 = dataGridView1.SelectedRows[0].Index;

int index2 = dataGridView1.SelectedRows[1].Index;

Podorozh p1 = podorozhi[index1];

Podorozh p2 = podorozhi[index2];

// Використовуємо оператор == і !=

if (p1 == p2)

MessageBox.Show("Подорожі однакові (місто + транспорт)");

else

MessageBox.Show("Подорожі різні");

}

else

{

MessageBox.Show("Виберіть рівно 2 подорожі для перевірки!");

}

}

private void button5\_Click(object sender, EventArgs e)

{

int index = -1;

if (dataGridView1.SelectedRows.Count > 0)

index = dataGridView1.SelectedRows[0].Index;

else if (dataGridView1.SelectedCells.Count > 0)

index = dataGridView1.SelectedCells[0].RowIndex;

if (index >= 0 && index < podorozhi.Count)

{

podorozhi[index]++; // мутуємо об’єкт

RefreshRow(index); // оновлюємо таблицю

}

else

{

MessageBox.Show("Виберіть одну подорож!");

}

}

private void button6\_Click(object sender, EventArgs e)

{

int index = -1;

if (dataGridView1.SelectedRows.Count > 0)

index = dataGridView1.SelectedRows[0].Index;

else if (dataGridView1.SelectedCells.Count > 0)

index = dataGridView1.SelectedCells[0].RowIndex;

if (index >= 0 && index < podorozhi.Count)

{

podorozhi[index]--; // мутуємо об’єкт

RefreshRow(index); // оновлюємо таблицю

}

else

{

MessageBox.Show("Виберіть одну подорож!");

}

}

private void RefreshRow(int index)

{

if (index < 0 || index >= podorozhi.Count) return;

Podorozh p = podorozhi[index];

dataGridView1.Rows[index].Cells[0].Value = p.Misto;

dataGridView1.Rows[index].Cells[1].Value = p.Vidstan;

dataGridView1.Rows[index].Cells[2].Value = p.Dni;

dataGridView1.Rows[index].Cells[3].Value = p.Transport;

dataGridView1.Rows[index].Cells[4].Value = p.RozrakhVartist();

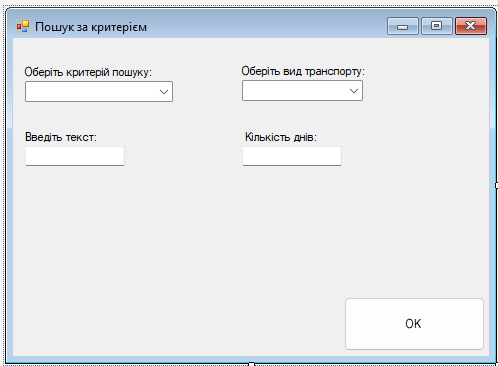
dataGridView1.Rows[index].Cells[5].Value = p.SerednyaTsinaKm().ToString("F2");

}

}

}

Form2 (Пошук)



Форма складається з: 2 comboBox, 2 textBox, 1 button

**Form2.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Windows.Forms;

namespace LR1\_3

{

public partial class Form2 : Form

{

private List<Podorozh> \_list;

public List<Podorozh> SearchResults { get; private set; }

public Form2(List<Podorozh> list)

{

InitializeComponent();

\_list = list;

// Заповнення combobox1 критеріями пошуку

comboBox1.Items.Clear();

comboBox1.Items.Add("Місто");

comboBox1.Items.Add("Транспорт");

comboBox1.Items.Add("Дні");

comboBox1.Items.Add("Місто + Транспорт + Дні"); // для комбінованого пошуку

comboBox1.SelectedIndex = 0; // за замовчуванням перший елемент

// Заповнення combobox2 транспортом

comboBox2.Items.Clear();

comboBox2.Items.Add("Будь-який"); // для всіх

foreach (var t in \_list.Select(p => p.Transport).Distinct())

{

comboBox2.Items.Add(t);

}

comboBox2.SelectedIndex = 0; // за замовчуванням “Будь-який”

}

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

{

string crit = comboBox1.SelectedItem?.ToString(); // критерій пошуку

string mistoText = textBox1.Text.Trim();

string transportText = comboBox2.SelectedItem?.ToString();

string dniText = textBox2.Text.Trim();

var results = \_list.AsEnumerable();

if (!string.IsNullOrEmpty(crit))

{

switch (crit)

{

case "Місто":

if (string.IsNullOrEmpty(mistoText))

{

MessageBox.Show("Будь ласка, введіть місто для пошуку!");

return;

}

results = results.Where(p => p.Misto.IndexOf(mistoText, StringComparison.OrdinalIgnoreCase) >= 0);

break;

case "Транспорт":

if (string.IsNullOrEmpty(mistoText))

{

MessageBox.Show("Будь ласка, введіть транспорт для пошуку!");

return;

}

results = results.Where(p => p.Transport.IndexOf(mistoText, StringComparison.OrdinalIgnoreCase) >= 0);

break;

case "Дні":

if (string.IsNullOrEmpty(dniText))

{

MessageBox.Show("Будь ласка, введіть кількість днів для пошуку!");

return;

}

if (!int.TryParse(dniText, out int dni))

{

MessageBox.Show("Кількість днів повинна бути числом!");

return;

}

results = results.Where(p => p.Dni == dni);

break;

case "Місто + Транспорт + Дні":

if (string.IsNullOrEmpty(mistoText) || string.IsNullOrEmpty(transportText) || string.IsNullOrEmpty(dniText))

{

MessageBox.Show("Будь ласка, заповніть усі поля для комбінованого пошуку!");

return;

}

if (!int.TryParse(dniText, out int dniCombined))

{

MessageBox.Show("Кількість днів повинна бути числом!");

return;

}

results = results.Where(p =>

p.Misto.IndexOf(mistoText, StringComparison.OrdinalIgnoreCase) >= 0 &&

p.Transport.Equals(transportText, StringComparison.OrdinalIgnoreCase) &&

p.Dni == dniCombined);

break;

}

}

var finalResults = results.ToList();

if (finalResults.Count == 0)

{

MessageBox.Show("Нічого не знайдено!");

return;

}

double totalCost = finalResults.Sum(p => p.RozrakhVartist());

double avgDays = finalResults.Average(p => p.Dni);

double minDistance = finalResults.Min(p => p.Vidstan);

double maxDistance = finalResults.Max(p => p.Vidstan);

MessageBox.Show(

$"Знайдено {finalResults.Count} подорожей\n" +

$"Сумарна вартість: {totalCost}\n" +

$"Середня кількість днів: {avgDays:F1}\n" +

$"Відстань: від {minDistance} до {maxDistance} км"

);

SearchResults = finalResults;

this.DialogResult = DialogResult.OK;

this.Close();

}

}

}

**Результат програми:**

|  |  |
| --- | --- |
|  | Для початку тестування, заповнимо таблицю даними. |
|  | Натискаємо кнопку “Об’єднати” і бачимо що програма видає повідомлення. Аналогічно з іншими кнопками. |
|  | Виділили два міста і натискаємо “Oб’єднати”. Програма видала що об’єднання пройшло успішно. |
|  | Виділимо два міста та натиснемо “Порівняти”. Програма видасть що якесь місто дорожче за інше. |
|  | Виділимо два міста та натиснемо “Перевірити рівність”. Програма видасть що подорожі абсолютно різні. |
|  | Тепер використаємо оператор ++ (Додає дні). Як бачимо в нас зараз 8 днів, натиснемо “++” та перевіримо результат.  Програма дійсно додає дні до вибраної поїздки. |
|  | Тепер застосуємо оператор “--”(Віднімає дні), та перевіримо результат роботи програми. Як бачимо у нас зараз 54 дні. Натиснемо – та перевіримо результат.  Як бачимо програма відняла день.  На цьому тестування програми завершено. |

**Висновок:** Під час виконання лабораторної роботи, я отримав практичні навички створення абстрактних типів даних і перевантаження операторів і умов в С++ та С#