Министерство образования и молодежной политики Свердловской области



ГАПОУ СО «Екатеринбургский колледж транспортного строительства»

**Отчёт по программе «Практическая работа №3. Индивидуальное задание.»**

Выполнила: Мезенцева Полина Сергеевна

Группа: ПР-31

Преподаватель: Мирошниченко Г.В

2024

**Задание 3:**

Используя методы Linq и коллекции (использовать не менее 2-х видов коллекций)

Описать базовый класс с указанными в задании полями и методами:

* Конструктор;
* Добавить 2 своих поля;
* Функция, которая определяет качество объекта – *Q* по заданной формуле;
* Вывод информации об объекте.
* Реализуйте и проверьте методы для добавления и удаления объекта базового класса (сделать 3 перегрузки!);

Построить класс потомок, который содержит:

* Дополнительное поле *Р;*
* Добавить еще 3 свойства
* Функция, которая определяет «качество» объекта класса потомка

– *Qp*, которая перекрывает функцию качества базового класса, выполняя вычисления по новой формуле.

* Добавить свою функцию для класса потомка

Создать проект для демонстрации работы: ввод-вывод информации об объектах базового класса и класса потомка.

Разработать модульные тесты к реализованным методам классов

|  |  |  |
| --- | --- | --- |
| №  Варианта | Поля и функция «качества» Q базового  класса | Поля и функция «качества» Qp  класса потомка |
| 2. | Дорожные работы:   * ширина дороги (м) * длина (м) * масса дорожного покрытия на 1 кв.м (кг) * Q=ширина \* длину \*масса/1000 | P: коэффициент прочности в зависимости от погодных условий  Qp= Q\*1,1 (если P = от 5 до 8)  Если P=3, 4, 9, 10 то Qp=Q\*1,6  иначе Qp=1,9\*Q |

**Входные данные**

double width – ширина дороги(м),

double length – длина дороги(м),

double roadMass – масса дорожного покрытия(м),

string location – местоположение,

string type – тип дорожного покрытия,

double p – коэффициент прочности,

string year – год окончания постройки,

double traffic – траффик движения,

string serviceLife – срок эксплуатации.

**Выходные данные**

dataGridViewRoadWorks – вывод в таблицу, берется из коллекции List <CoefOfSafety>; Сообщения в MessageBox.

**Описание разработанных функций и методов:**

Методы базового класса:

|  |  |
| --- | --- |
| AddList() | Предназначен для добавления новой дороги в коллекцию. Метод получает объект класса, который нужно занести в коллекцию, и затем добавляет его в List |
| RemoveRoadList(3 перегрузки) | Предназначен для удаления записи о дороге в List по разным критериям: по экземпляру класса, по индексу и по совпадениям между входными данными и данными в List |

Функции базового класса:

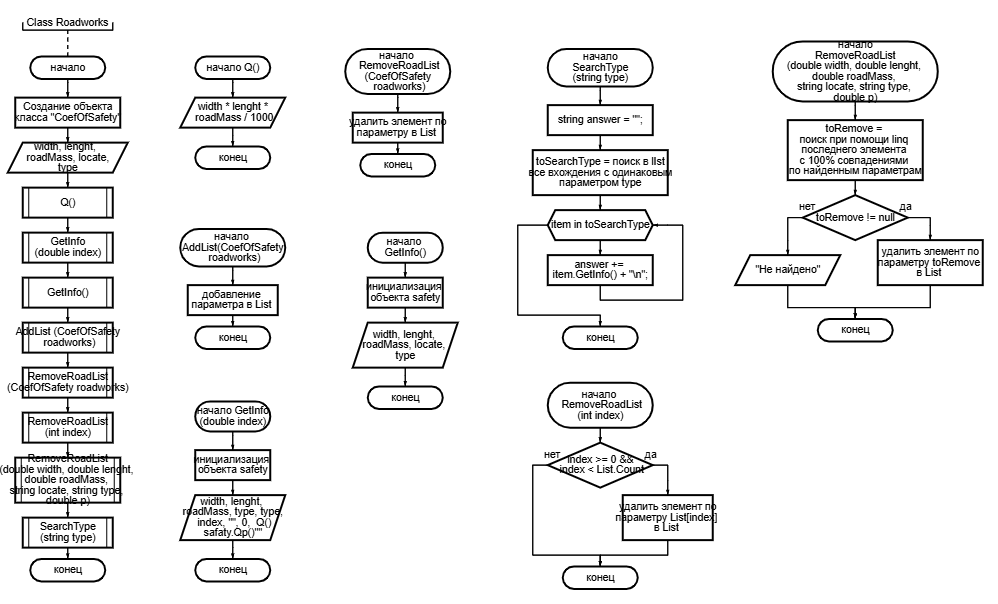
|  |  |
| --- | --- |
| Q() | Функция, которая определяет качество объекта Q по заданной формуле. Возвращает значение double. |
| GetInfo(2 перегрузки) | Функции, предназначенные для вывода информации об объекте. Функции возвращают строку string. |
| SearchType() | Функция, предназначенная для вывода информации об конкретных объектах по выборке. Возвращает строку string. |

Функции дочернего класса:

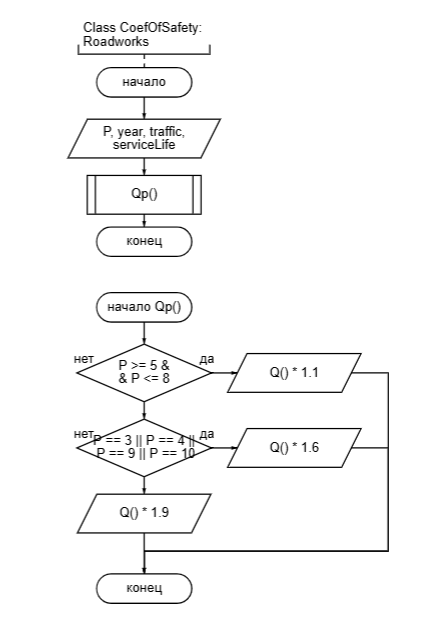
|  |  |
| --- | --- |
| Qp | Функция, которая определяет «качество» объекта класса-потомка Qp, выполняет вычисления по новой формуле. |

**Блок-схема**

**Roadworks.cs:**



**CoefOfSafety.cs:**



**Листинг программы**

Form1.cs

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

comboBoxP.DropDownStyle = ComboBoxStyle.DropDownList;

dataGridViewRoadWorks.AutoGenerateColumns = true; // Позволяем автоматическую генерацию столбцов

}

private void UpdateDataGridView()

{

dataGridViewRoadWorks.DataSource = null; // Сбрасываем источник данных

dataGridViewRoadWorks.DataSource = CoefOfSafety.roadList; // Устанавливаем новый источник данных

}

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

{

if (!(textBoxWidth.Text == "" || textBoxLength.Text == "" || textBoxRoadMass.Text == "" ||

textBoxLocation.Text == "" || comboBoxType.Text == "" || comboBoxP.Text == ""))

{

if (double.TryParse(textBoxWidth.Text, out double width) && double.TryParse(textBoxLength.Text, out double length) &&

double.TryParse(textBoxRoadMass.Text, out double roadMass) && double.TryParse(comboBoxP.Text, out double b4) &&

int.TryParse(textBoxYear.Text, out int p))

{

string location = textBoxLocation.Text;

string type = comboBoxType.Text;

string year = textBoxYear.Text;

string serviceLife = textBoxServiceLife.Text;

double traffic = double.Parse(textBoxTraffic.Text);

// Создание экземпляра CoefOfSafety и добавление в список

CoefOfSafety roadwork = new CoefOfSafety(width, length, roadMass, location, type, p, year, traffic, serviceLife);

roadwork.AddList(roadwork);

// Обновление DataGridView

UpdateDataGridView();

}

else

{

MessageBox.Show("Проверьте правильность написания информации.");

}

}

else MessageBox.Show("Заполните все поля");

}

private void removeButton1\_Click(object sender, EventArgs e)//1

{

if (dataGridViewRoadWorks.SelectedRows.Count > 0)

{

var selectedRoadwork = (CoefOfSafety)dataGridViewRoadWorks.SelectedRows[0].DataBoundItem;

Roadworks.RemoveRoadList(selectedRoadwork);

UpdateDataGridView();

}

else MessageBox.Show("В базе данных не найдено информации");

}

private void removeButton2\_Click(object sender, EventArgs e)//2

{

if (dataGridViewRoadWorks.SelectedRows.Count > 0)

{

if (int.TryParse(textBoxIndex.Text, out int index))

{

// Удаляем объект по индексу

Roadworks.RemoveRoadList(index);

// Обновляем DataGridView

UpdateDataGridView();

}

else

{

MessageBox.Show("Пожалуйста, введите корректный индекс.");

}

}

else MessageBox.Show("В базе данных не найдено информации");

}

private void removeButton3\_Click(object sender, EventArgs e)//3

{

if (!(textBoxWidth.Text == "" || textBoxLength.Text == "" || textBoxRoadMass.Text == "" ||

textBoxLocation.Text == "" || comboBoxType.Text == "" || comboBoxP.Text == ""))

{

if (double.TryParse(textBoxWidth.Text, out double b1) && double.TryParse(textBoxLength.Text, out double b2) &&

double.TryParse(textBoxRoadMass.Text, out double b3) && double.TryParse(comboBoxP.Text, out double b4))

{

double width = b1;

double length = b2;

double roadMass = b3;

double p = b4;

string location = textBoxLocation.Text;

string type = comboBoxType.Text;

Roadworks.RemoveRoadList(width, length, roadMass, location, type, p);

UpdateDataGridView();

}

else

{

MessageBox.Show("Проверьте правильность написания информации.");

}

}

else MessageBox.Show("Заполните все поля");

}

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

{

if (dataGridViewRoadWorks.SelectedRows.Count > 0)

{

if (dataGridViewRoadWorks.SelectedRows.Count > 0)

{

// Получаем выбранный объект

var selectedRoadwork = (Roadworks)dataGridViewRoadWorks.SelectedRows[0].DataBoundItem;

// Получаем первую выделенную строку

DataGridViewRow selectedRow = dataGridViewRoadWorks.SelectedRows[0];

// Получаем значение из нужного столбца

double cellValue = (double)selectedRow.Cells[0].Value;

// Вызываем метод GetInfo для получения информации

string info = selectedRoadwork.GetInfo(cellValue);

// Выводим информацию в MessageBox

MessageBox.Show(info, "Информация о дорожных работах");

}

}

else MessageBox.Show("В базе данных не найдено информации");

}

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

{

MessageBox.Show(Roadworks.SearchType(comboBoxSearchType.Text));

}

}

}

Class Readworks.cs

class Roadworks

{

CoefOfSafety safety;

public static List<CoefOfSafety> roadList = new List<CoefOfSafety>();

public double width { get; set; }

public double lenght { get; set; }

public double roadMass { get; set; }

public string locate { get; set; } //свое поле 1

public string type { get; set; } // свое поле 2

//конструктор

public Roadworks(double width, double lenght, double roadMass, string locate, string type)

{

this.width = width;

this.lenght = lenght;

this.roadMass = roadMass;

this.locate = locate;

this.type = type;

}

//вычисляем Q

public double Q()

{

return width \* lenght \* roadMass / 1000;

}

public string GetInfo(double index)//вывод информации

{

safety = new CoefOfSafety(width, lenght, roadMass, type, type, index, "", 0, "");

return $"\nШирина дороги: {width}," +

$"\nДлина: {lenght}," +

$"\nМасса дорожного покрытия на 1 кв.м.: {roadMass}" +

$"\nМестоположение:{locate}," +

$"\nТип дорожного покрытия:{type};" +

$"\nQ = {Q()}," +

$"\nQp = {safety.Qp()}.";

}

public string GetInfo()//вывод информации для метода поиска по типу покрытия

{

return $"\nШирина дороги: {width}," +

$"\nДлина: {lenght}," +

$"\nМасса дорожного покрытия на 1 кв.м.: {roadMass}" +

$"\nМестоположение:{locate}," +

$"\nТип дорожного покрытия:{type}." +

"\n";

}

//добавление

public void AddList(CoefOfSafety roadworks)

{

roadList.Add(roadworks);

}

//удаление по экземляру класса

public static void RemoveRoadList(CoefOfSafety roadworks)

{

roadList.Remove(roadworks);

}

//удаление по индексу

public static void RemoveRoadList(int index)

{

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

{

roadList.RemoveAt(index);

}

}

//удаление по полям

public static void RemoveRoadList(double width, double lenght, double roadMass, string locate, string type, double p)

{

var toRemove = roadList.FindLast(r => r.width == width && r.lenght == lenght && r.roadMass == roadMass && r.locate == locate && r.type == type && r.P == p);

if (toRemove != null)

roadList.Remove(toRemove);

else MessageBox.Show("не найдено");

}

//новый метод,поиск по типу покрытия

public static string SearchType(string type)

{

string answer = "";

var toSearchType = roadList.FindAll(s => s.type == type);

foreach (var item in toSearchType)

{

answer += item.GetInfo() + "\n";

}

return answer;

}

}

Class CoefOfSafety.cs

class CoefOfSafety : Roadworks

{

public double P { get; set; } // Коэффициент прочности

// Дополнительные свойства

public string year { get; set; } // Год постройки

public double traffic { get; set; } //Трафик движения

public string serviceLife { get; set; } //Срок эксплуатации

//конструктор

public CoefOfSafety(double width, double length, double roadMass,

string location, string type, double p, string year, double traffic, string serviceLife)

: base(width, length, roadMass, location, type)

{

P = p; // Инициализация P

this.year = year;

this.traffic = traffic;

this.serviceLife = serviceLife;

}

// Метод для расчета качества с учетом коэффициента прочности

public double Qp()

{

if (P >= 5 && P <= 8) return Q() \* 1.1;

else if (P == 3 || P == 4 || P == 9 || P == 10) return Q() \* 1.6;

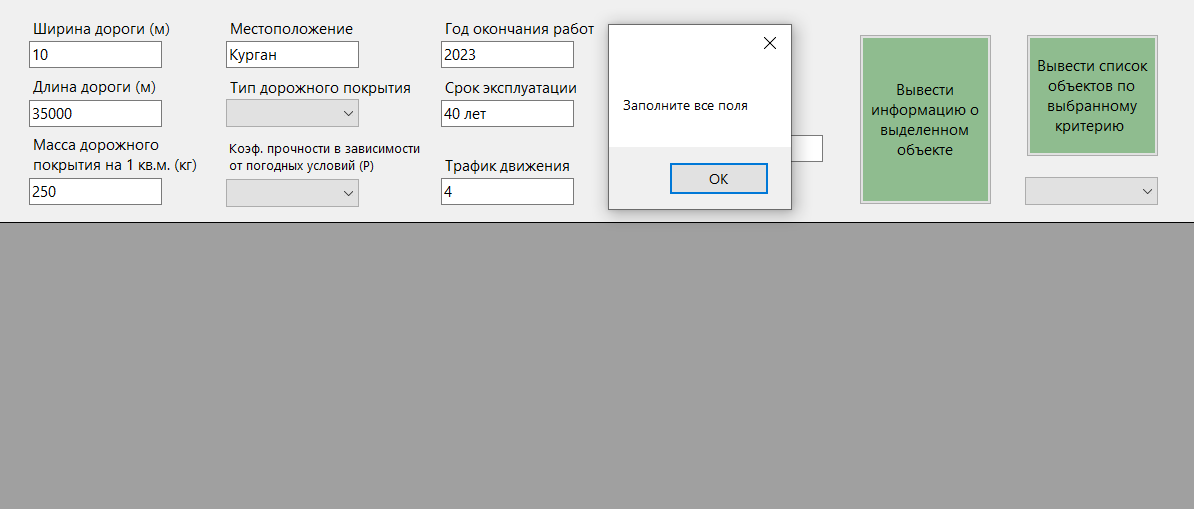
else return Q() \* 1.9;

}

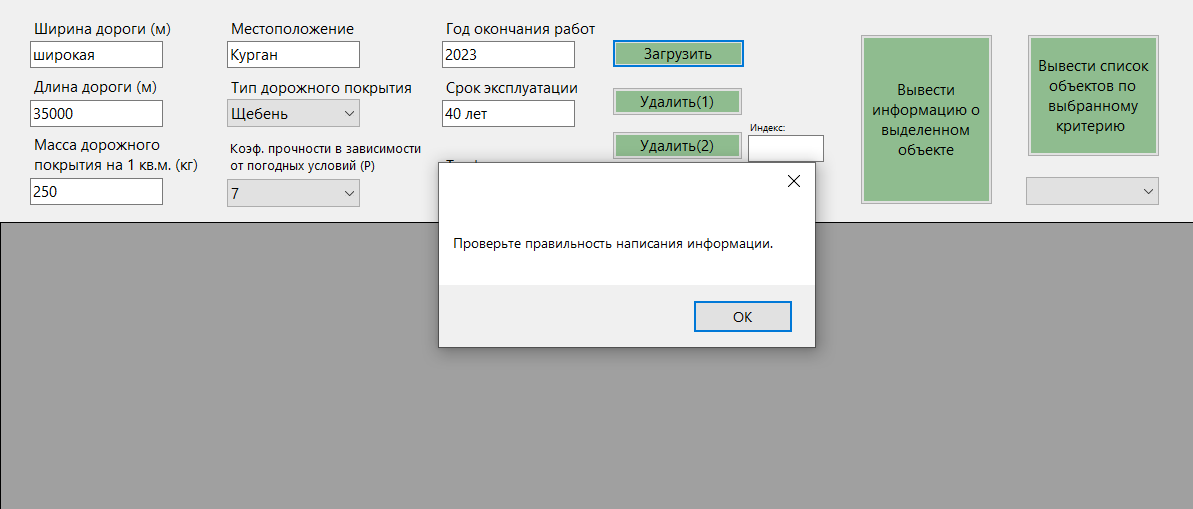
}

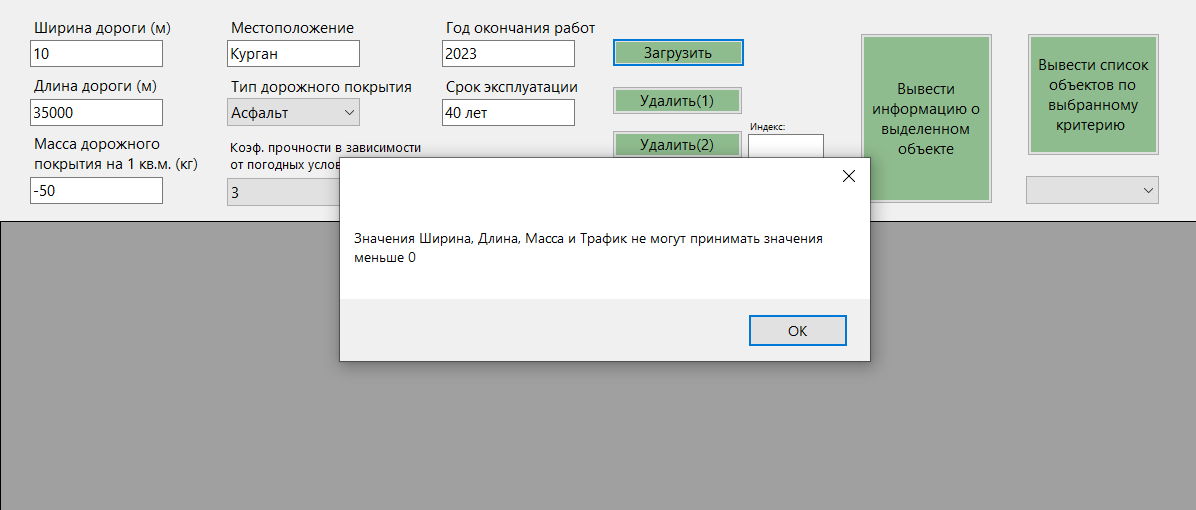
**Тестовые ситуации**

Пользователь не ввёл данные в какое-либо поле для ввода и нажмет кнопку, то ему будет показано сообщение о том, что он ещё не закончил ввод данных:

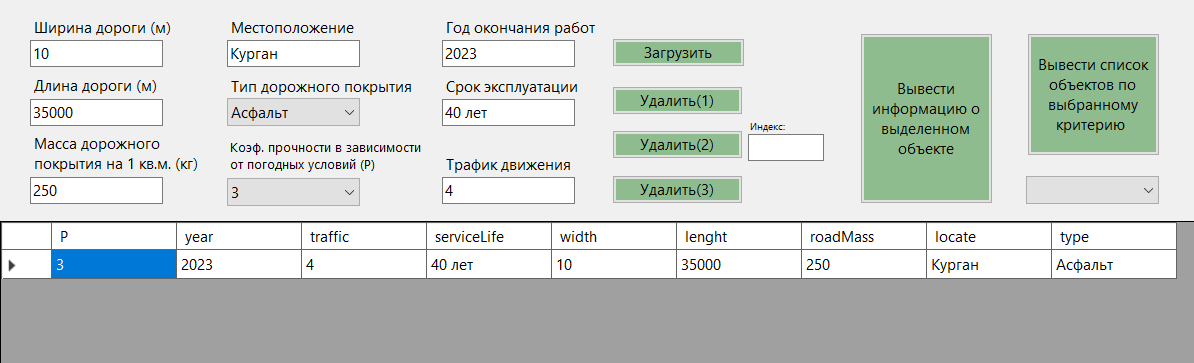


Если пользователь в поле, предназначенное для плавающих или целочисленных чисел введёт не число или отрицательное число, то ему будет показано сообщение об ошибке:

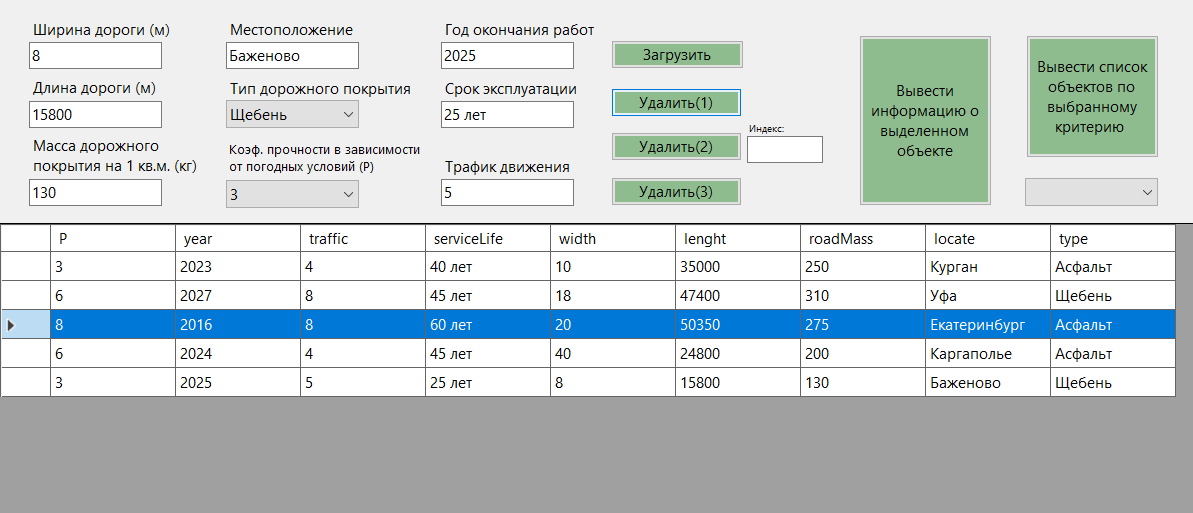
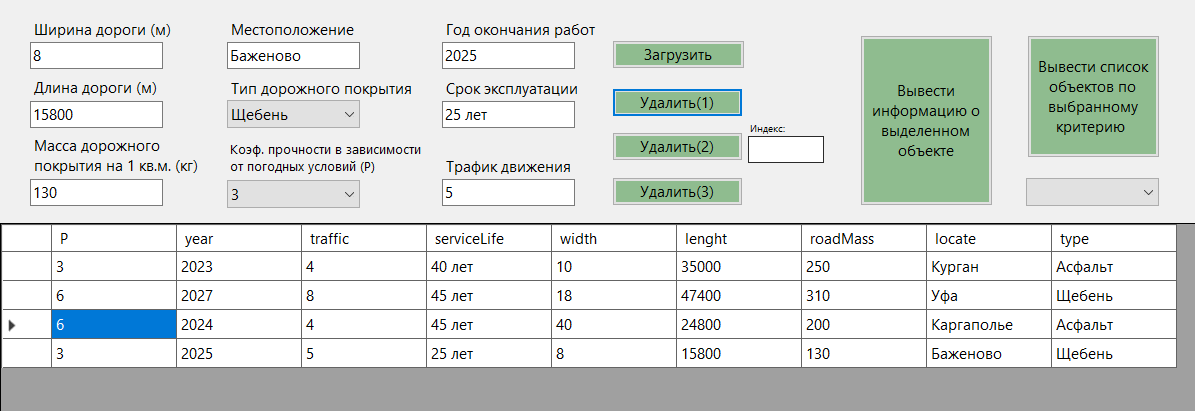


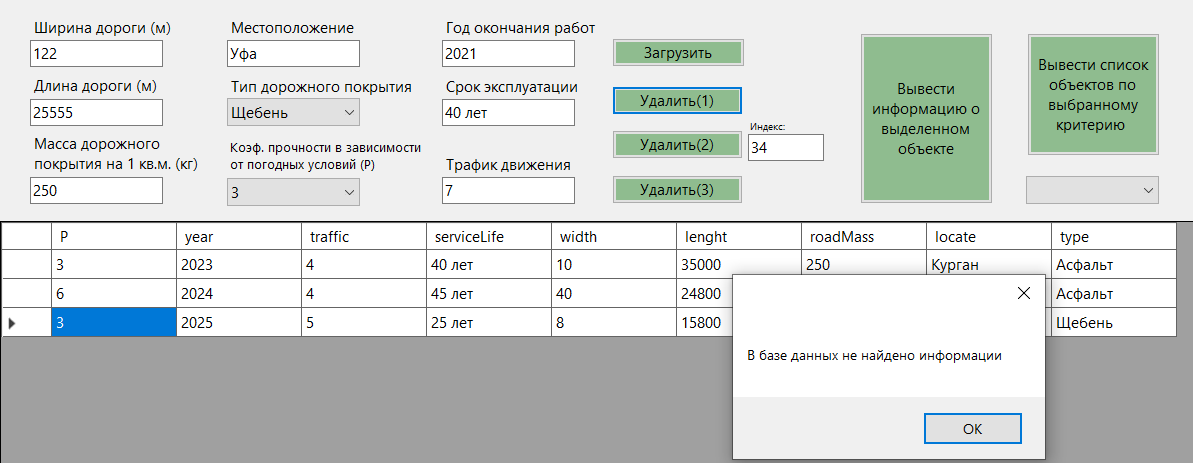


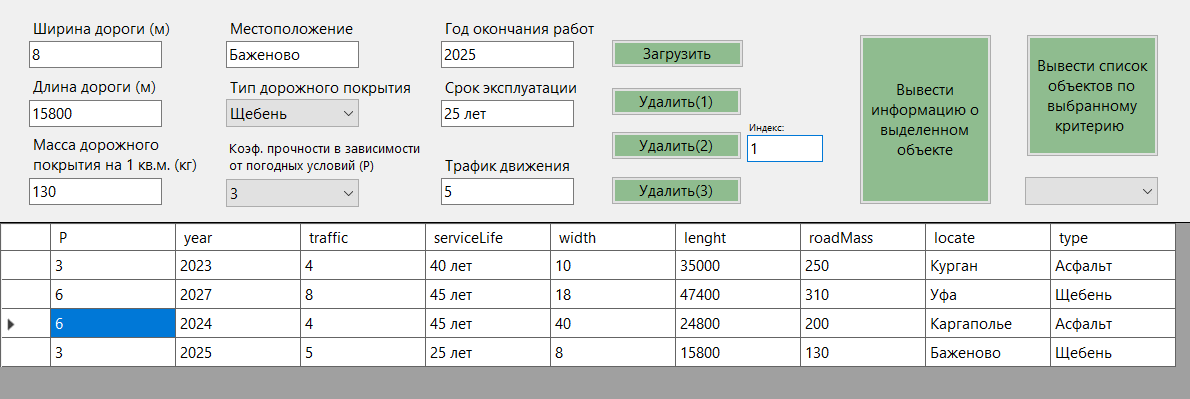
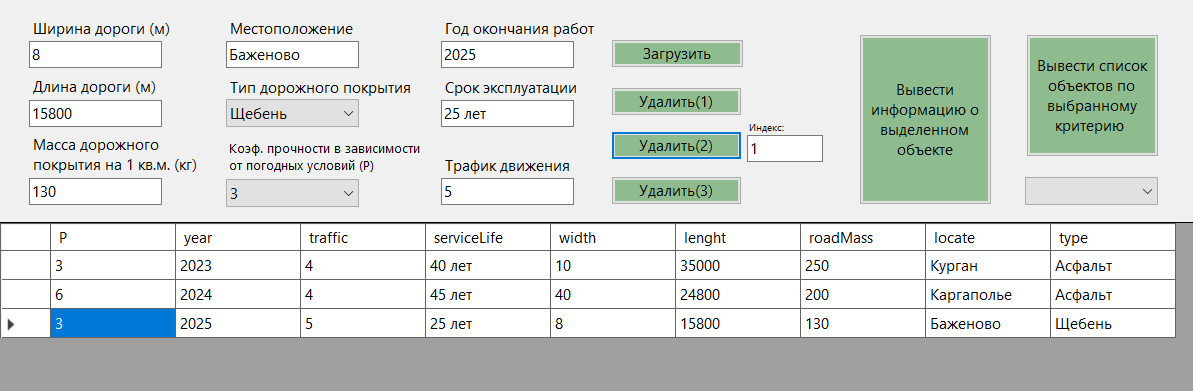
Как только пользователь вводит все данные о магазине корректно, они отображаются в таблице:



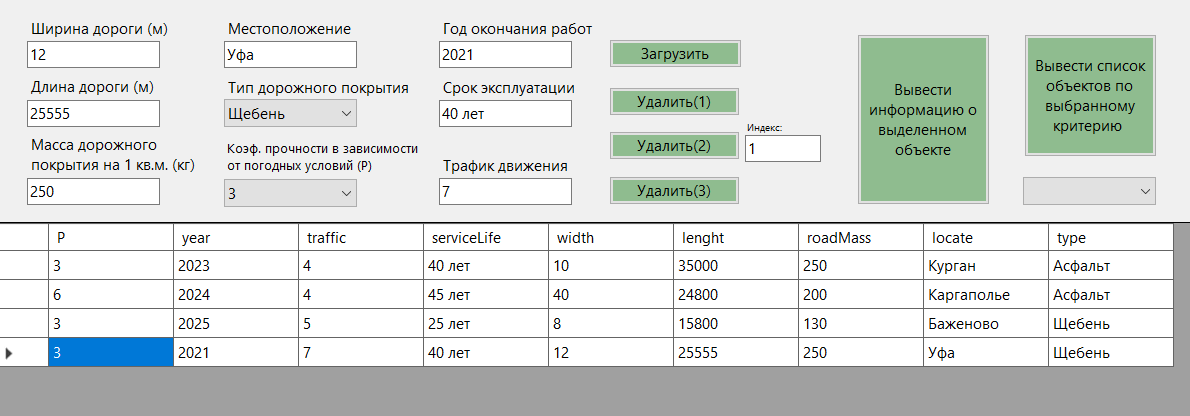
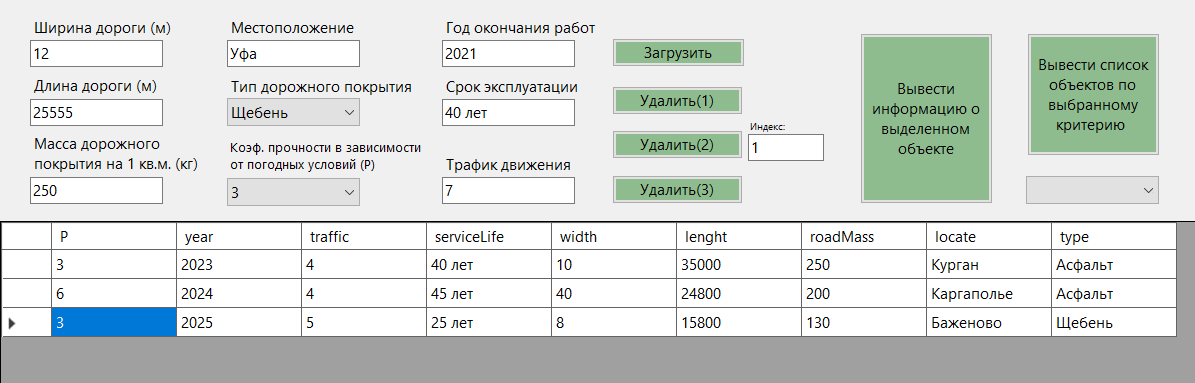
В программе существует три метода удаления- на одну кнопку по одной перегрузки метода удаления:

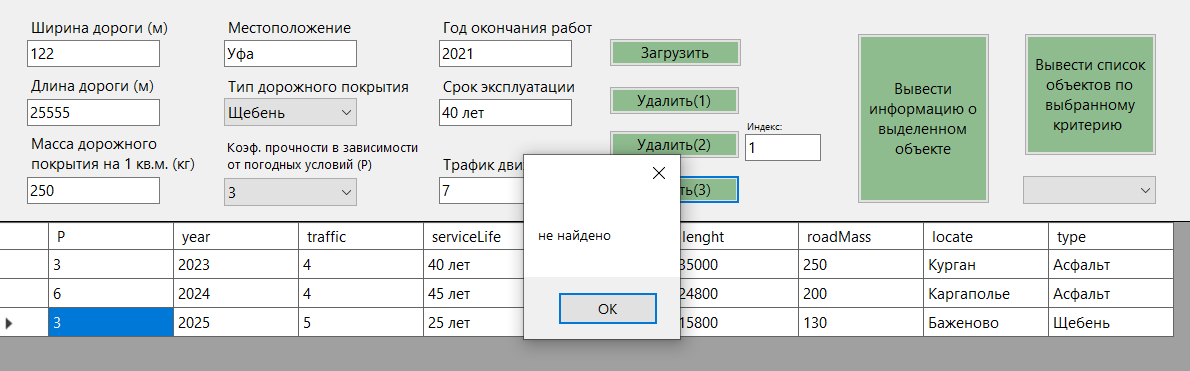
1. **По выделению нужной строки для удаления:** 

**1.1 Не выделена строка:** 

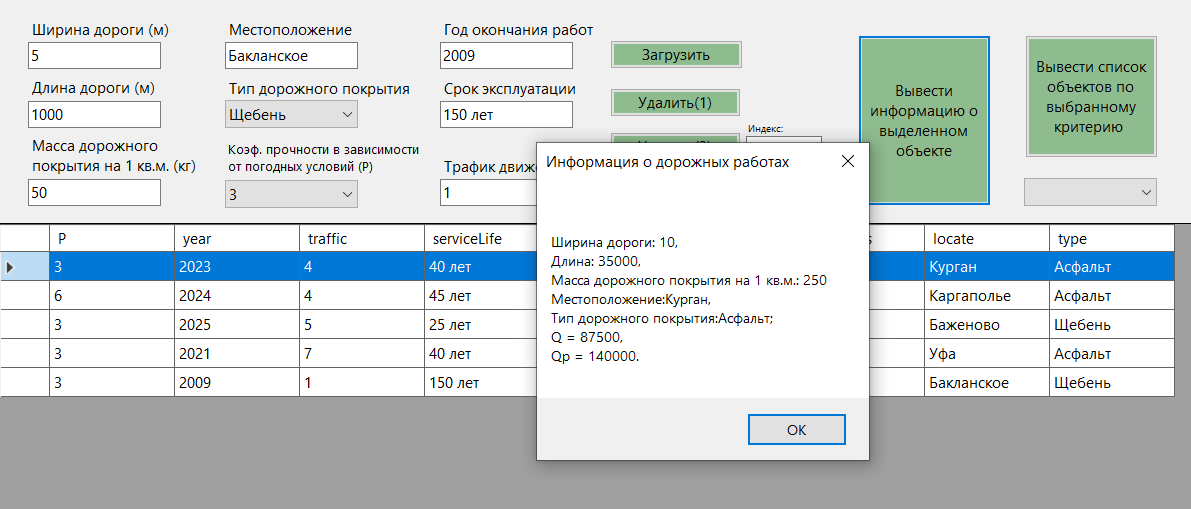
1. **По индексу:**  

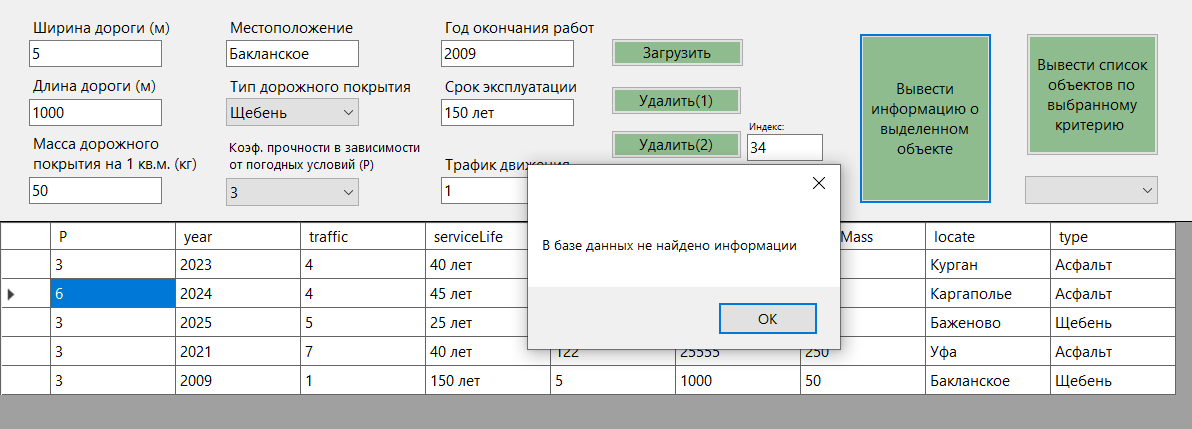
**2.1. Если не нашел- ничего не происходит**

1. **По первому совпадению в полях и таблице:**  

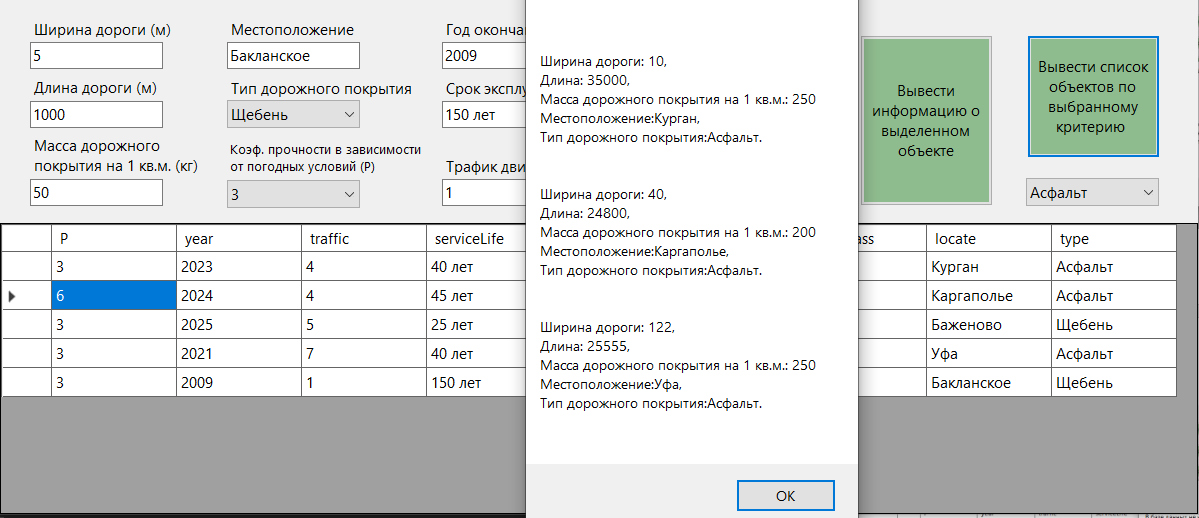
**3.1. Не найдено:** 

Метод вывода информации о выделенном объекте считает качество объекта:



Строка не выделена: 

Дополнительная функция. Поиск записей работ по типу дорожного покрытия:



**Модульные тесты:**

**Листинг тестов:**

using ClassLibrary1;

namespace TestProject

{

[TestClass]

public class UnitTest1

{

Roadworks roadworks;

CoefOfSafety safety;

[TestInitialize]

public void Setup()

{

// Инициализация объектов перед каждым тестом

roadworks = new Roadworks(10, 35000, 250, "Курган", "Асфальт");

safety = new CoefOfSafety(10, 35000, 250, "Курган", "Асфальт", 6, "2016", 3, "25 лет");

}

[TestMethod]

public void Add()

{

Roadworks.roadList.Clear(); // Очистка списка перед тестом

// Добавление объекта в список

safety.AddList(safety);

// Проверка, что объект был добавлен в список

Assert.AreEqual(safety, Roadworks.roadList[0]);

}

[TestMethod]

public void TestCalculateQ()

{

// Проверка расчета Q

double expectedQ = roadworks.width \* roadworks.lenght \* roadworks.roadMass / 1000; // Q = 10 \* 35000 \* 250 / 1000

Assert.AreEqual(expectedQ, roadworks.Q());

}

///

[TestMethod]

public void TestCalculateQp1()

{

// Проверка расчета Qp1

double expectedQp1 = Math.Round(roadworks.Q() \* 1.1); // P = 6, поэтому Qp = Q \* 1.1

Assert.AreEqual(expectedQp1, safety.Qp());

}

[TestMethod]

public void TestCalculateQp2()

{

safety = new CoefOfSafety(10, 35000, 250, "Курган", "Асфальт", 9, "2016", 3, "25 лет");

// Проверка расчета Qp2

double expectedQp2 = Math.Round(roadworks.Q() \* 1.6); // P = 9, поэтому Qp = Q \* 1.6

Assert.AreEqual(expectedQp2, safety.Qp());

}

[TestMethod]

public void TestCalculateQp3()

{

safety = new CoefOfSafety(10, 35000, 250, "Курган", "Асфальт", 1, "2016", 3, "25 лет");

// Проверка расчета Qp3

double expectedQp3 = Math.Round(roadworks.Q() \* 1.9); // P = 1, поэтому Qp = Q \* 1.9

Assert.AreEqual(expectedQp3, safety.Qp());

}

///

[TestMethod]

public void TestGetInfo()

{

// Проверка вывода информации

string expectedInfo = "\nШирина дороги: 10," +

"\nДлина: 35000," +

"\nМасса дорожного покрытия на 1 кв.м.: 250," +

"\nМестоположение: Курган," +

"\nТип дорожного покрытия: Асфальт;" +

"\nQ = 87500," +

"\nQp = 96250.";

Assert.AreEqual(expectedInfo, roadworks.GetInfo(6));

}

[TestMethod]

public void TestRemoveRoadListByObject()

{

// Проверка удаления объекта из списка

Roadworks.roadList.Clear(); // Очистка списка перед тестом

safety.AddList(safety);

Roadworks.RemoveRoadList(safety);

Assert.AreEqual(0, Roadworks.roadList.Count);

}

[TestMethod]

public void TestRemoveRoadListByIndex()

{

// Проверка удаления по индексу

Roadworks.roadList.Clear(); // Очистка списка перед тестом

safety.AddList(safety);

Roadworks.RemoveRoadList(0);

Assert.AreEqual(0, Roadworks.roadList.Count);

}

[TestMethod]

public void TestRemoveRoadListByAll()

{

// Проверка удаления по совпадениям в полях

Roadworks.roadList.Clear(); // Очистка списка перед тестом

safety.AddList(safety);

Roadworks.RemoveRoadList(10, 35000, 250, "Курган", "Асфальт", 6);

Assert.AreEqual(0, Roadworks.roadList.Count);

}

[TestMethod]

public void TestSearchType()

{

// Проверка поиска по типу

Roadworks.roadList.Clear(); // Очистка списка перед тестом

safety.AddList(safety);

string result = Roadworks.SearchType("Асфальт");

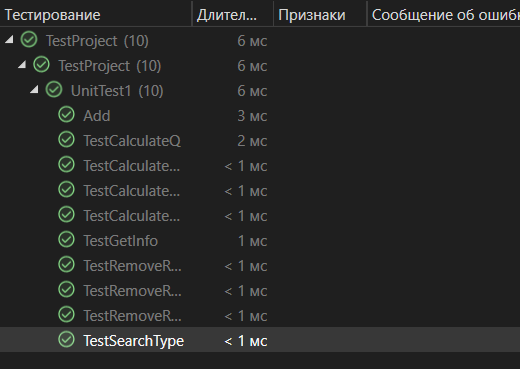
Assert.IsTrue(result.Contains("Ширина дороги: 10"));

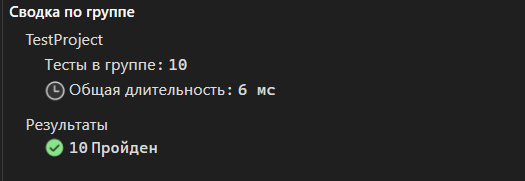
}

}

}

**Результаты прохождения тестов:**





**Используемые библиотеки:**

|  |  |
| --- | --- |
| System | Содержит основные классы и базовые классы, определяющие часто используемые значения и ссылочные типы данных, события и обработчики событий, интерфейсы, атрибуты и исключения обработки. |
| System.Collections.Generic | Содержит интерфейсы и классы, определяющие универсальные коллекции, которые позволяют пользователям создавать строго типизированные коллекции, обеспечивающие лучшую безопасность типов и производительность, чем не универсальные строго типизированные коллекции. |
| System.Windows.Forms | Используется для создания многофункциональных настольных приложений с графическими пользовательскими интерфейсами (GUI) в Windows. Оно включает в себя широкий спектр элементов управления (например, кнопки, текстовые поля и таблицы данных) и функций для обработки пользовательского ввода. |

**Используемые инструменты:** WindowsForms C#.