**Саутов Д.С. А-12-18**

**Лабораторная работа №6 «C# и SQL»**

**Вариант №16 (6)**

**Задание**

1. Создать базу данных с 1 таблицей. При необходимости можно добавить свои поля (столбцы таблицы), соответствующие общему смыслу таблицы.
2. Загрузить в таблицу небольшое количество данных в соответствие с заданием.
3. Разработать пользовательское приложение с интерфейсом, в котором реализовать

функции взаимодействия с данными.

Приложение должно содержать 7 различных функций, например:

− подсчет максимального, минимального, среднего во всей таблице;

− подсчет количества записей, удовлетворяющих условию;

− вывод строк, содержащих заданное значение;

−вывод строк, удовлетворяющих условию (например, вывести строки, где некоторое поле > n, но < m);

− удаление строк по определенному условию;

− изменение строк в соответствие с условием.

Помимо этого, в приложении должны быть реализованы базовые функции добавления, изменения и удаления данных (записей). Часть функций (2 и более)необходимо реализовать средствами C#, а часть (2 и более)–средствами SQL.

1. Обеспечить проверку корректности вводимых данных.

**Таблица «СНТ»**: номер участка, площадь участка, ФИО владельца, дата приобретения, кадастровая стоимость, величина членских взносов, наличие электричества, наличие льгот.

1. **Постановка задачи**

Создать бузц данных со следующей таблицей «**СНТ**» (Табл. 1) **:**

Табл. 1

Таблица из базы данных

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Id | NumberOfLand | Area | F | I | O | PurchaseDate | Donation | Electricity | Benefits |
| int | int | int | nvarchar(50) | nvarchar(50) | nvarchar(50) | int | int | nvarchar(50) | nvarchar(50) |

**Обязательные поля**: NumberOfLand, Area, F, I, O, PurchaseDate, Donation, Electricity, Benefits

Реализовать следующие функции взаимодействия с базой данных:

1. Подсчет максимального по площади участка (его площадь)
2. Подсчет минимального по площади участка (его площадь)
3. Подсчет средней по всей таблицы площади
4. Вывод строк с территорией больше указанного пользователем
5. Вывод строк с номером участка равному числу, указанным пользователем
6. Удаление участков, где площадь меньше указанного пользователем
7. Изменение полей Electricity и Benefits, с величиной членских взносов < указанных пользователем.
8. Добавление в таблицу
9. Изменение всех полей в строке с выбранным Id
10. Удаление строки с выбранным Id

**Методы программы:** ввод, обработка, вывод информации.

**Входные/выходные данные:** строки;

**Ограничения:** Проверка на отсутствие обязательных данных и на правильность вводимых данных.

**Среда разработки:** Визуальное оконное приложение, ООП подход, на языке C#

1. **Разработка программы**

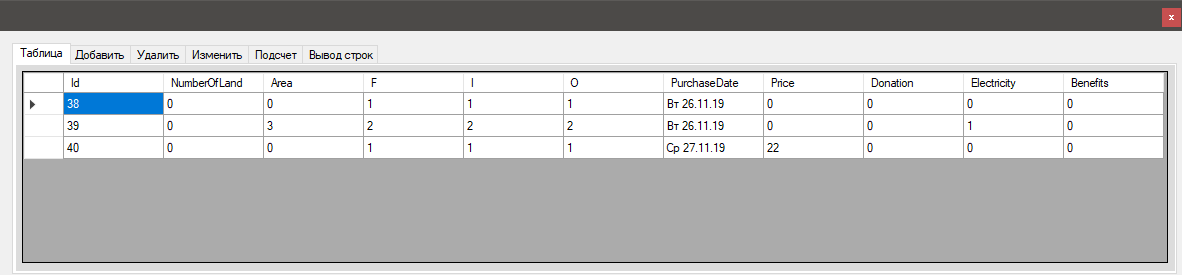
**2.1. Разработка структуры программы**

Программа будет содержать форму типа WF (Windows Forms) и вызывать ее.

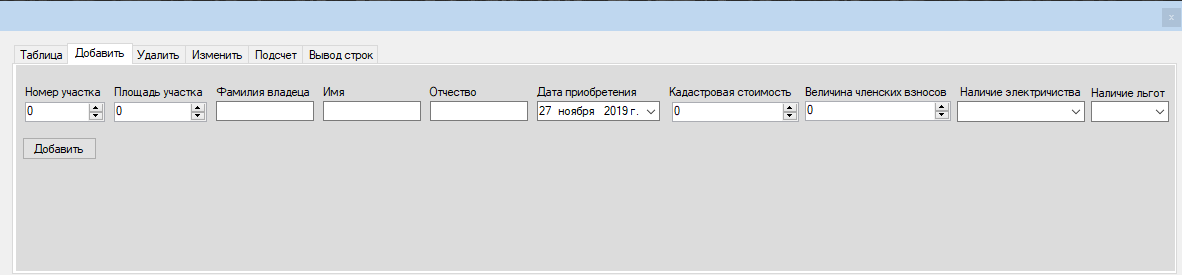
Для того, чтобы в поля, которые подразумевают только числовой ответ, а не строчку из всевозможных символов, будем использовать NumericUpDown.

**2.2. Разработка пользовательского интерфейса**

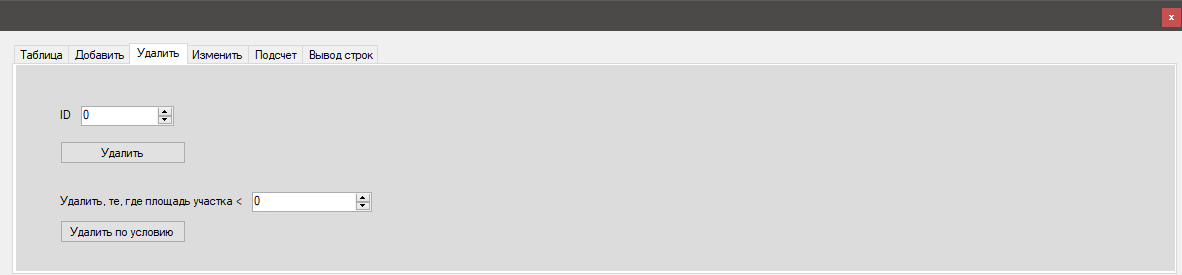
Программа предназначен для любого пользователя. Вид формы представлен на Рис. 1, используемые элементы управления в форме приведены в табл. 2.



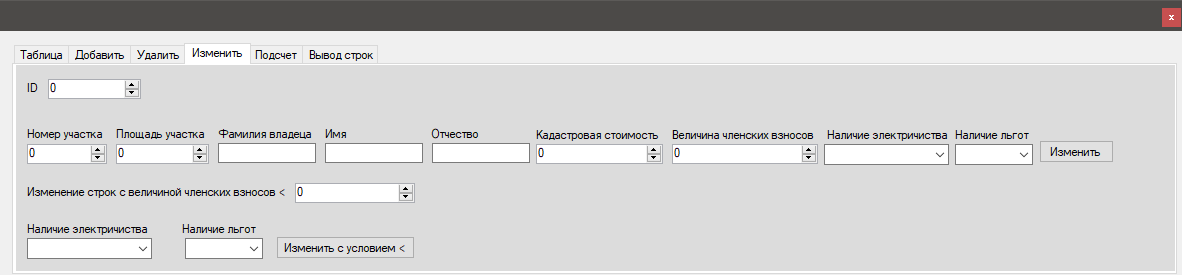
а)



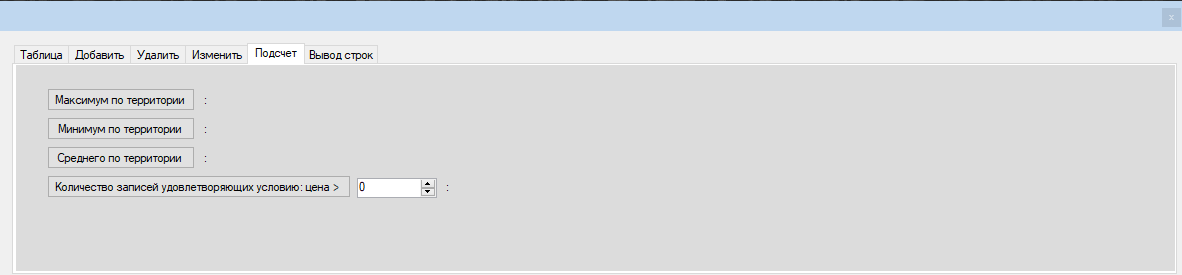
б)



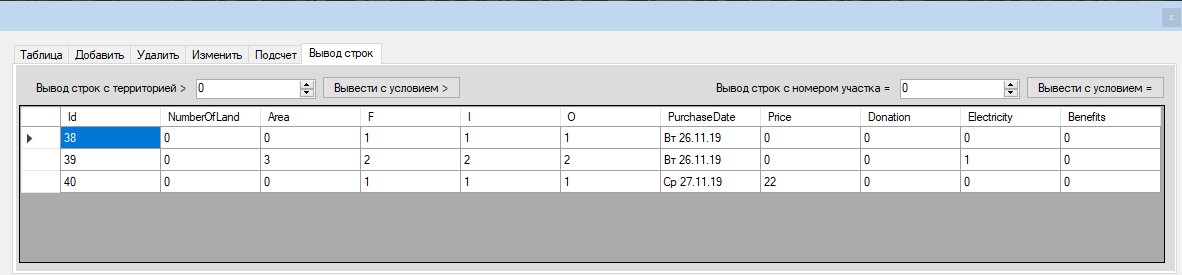
в)



г)



д)



е)

*Рис. 1. Форма разбитая на разделы, где а)раздел с таблицей б) раздел добавления в) раздел удаления, г) раздел удаления, д) раздел подсчета, е)раздел вывода строк*

Табл. 2

*используемые элементы управления в форме*

|  |  |  |
| --- | --- | --- |
| Элемент управления | класс | Описание |
| label1-31 | Label | Навигационная строка |
| numericUpDown1-15 | NumericUpDown | Элемент для ввода чисел |
| comboBox1-6 | ComboBox | Элемент для выбора определенных ответов |
| textBox3-8 | TextBox | Элемент для записи текста |
| button1-11 | Button | Кнопки выполнения определенных функций |
| tabControl1 | TabControl | Позволяет разделить форму на разделы |
| dataGridView1 | DataGridView | Отображает таблицу для вывода строк |

1. **Реализация и тестирование**

***Реализация***

Программа разработана на языке C# и T-SQL. Методы 1-3 реализованы средствами С#, методы 4-10 реализованы средствами T-SQL. Реализация кода программы будет расписана ниже в пункте «**Листинг кода**».

**Описание компьютера:**

Операционная система: Windows 10 (x64)

ОЗУ: 16 ГБ

Процессор: Intel Core i5-2500

Видеокарта: Radeon R7 200 Series

***Тестирование***

**Объект тестирования:** Lab5.exe

**Цель тестирования:** Проверка общей работоспособности программы.

**Метод тестирование:** функциональное тестирование.

Тестирование будет проводиться со следующими входными данными, изображенными на Рис. 2.

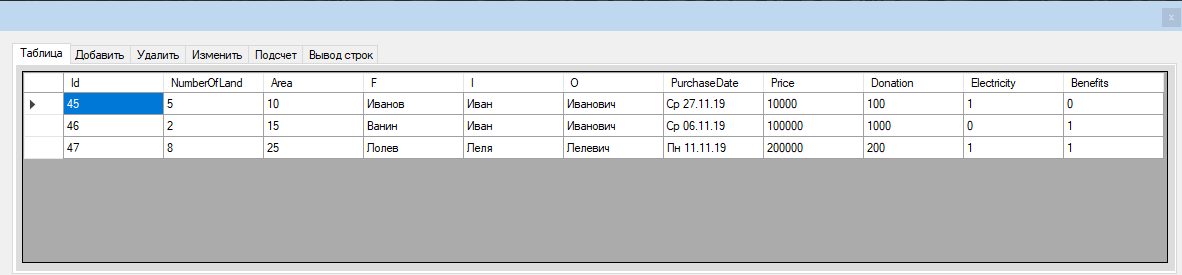


Рис. 2. Таблица данных в начале перед тестами

Тестирование методов 1-10 изображено на Рис. 3-

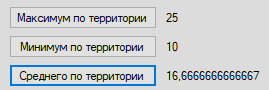


Рис. 3 Тестирование методов 1-3

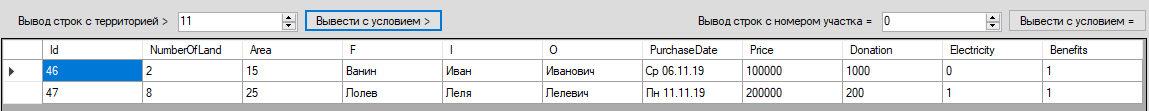


Рис. 4. Тестирование метода 4

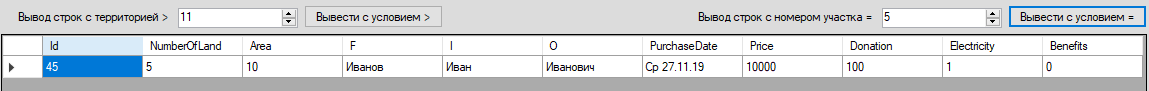


Рис. 5. Тестирование метода 5

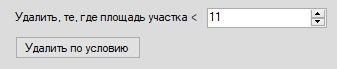


Рис. 6. Форма для метода 6

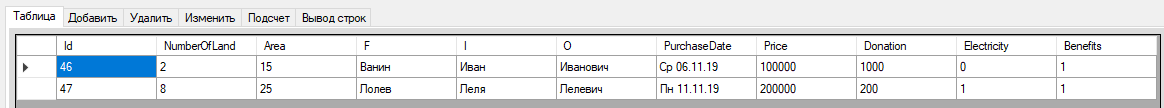


Рис. 7. Изменение в таблице после вызова метода 6

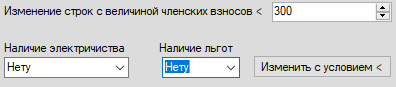


Рис. 8. Форма для метода 7

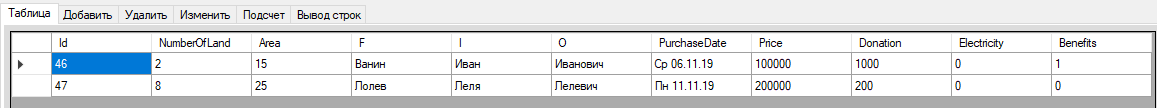


Рис. 9. Изменение в таблице после вызова метода 7

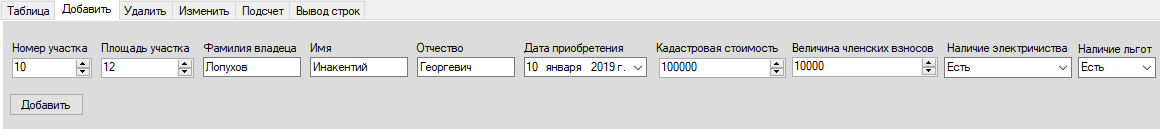


Рис. 10. Форма для метода 8

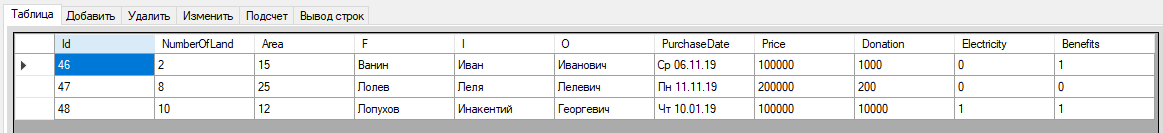


Рис. 11. Изменение в таблице после вызова метода 8



Рис. 12. Форма для метода 9

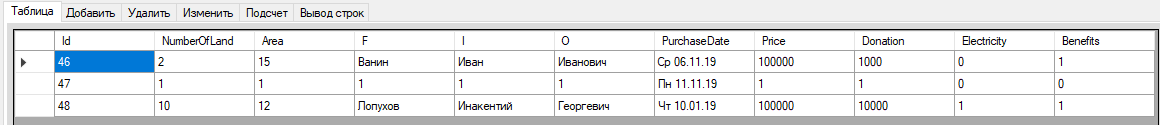
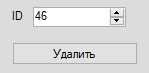
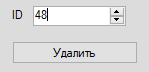


Рис. 13. Изменение в таблице после вызова метода 9

а) б)

Рис. 14. Форма для метода 10, где а) Первый вызов метода, б) второй вызов метода

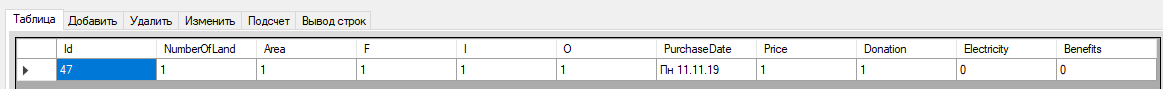


Рис. 15. Изменение в таблице после вызова метода 10 a) и б)

**Листинг кода**

**Файл «Form1.cs»**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WindowsFormsApp3

{

public partial class Form1 : Form

{

SqlConnection sqlConnection = new SqlConnection(@"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=D:\SAUTOV'S PRIDE\Рабочий стол\source\repos\WindowsFormsApp3\WindowsFormsApp3\Database.mdf;Integrated Security=True");

public Form1()

{

InitializeComponent();

}

void displayData() {//Команды для обновления данных на форме

SqlCommand cmd = sqlConnection.CreateCommand();

cmd.CommandType = CommandType.Text;

cmd.CommandText = "SELECT \* FROM [Table]";

cmd.ExecuteNonQuery();

DataTable dt = new DataTable();

SqlDataAdapter da = new SqlDataAdapter(cmd);

//da.Fill(databaseDataSet.Table);

//tableDataGridView.DataSource = databaseDataSet.Table;

da.Fill(dt);

tableDataGridView.DataSource = dt;

}

private void Form1\_Load(object sender, EventArgs e)//При загрузке формы подгрузить данные из таблицы и открыть соединение

{

sqlConnection.Open();

displayData();

}

private void Form1\_FormClosing(object sender, FormClosingEventArgs e)//при закрытии формы

{

if (sqlConnection != null && sqlConnection.State != System.Data.ConnectionState.Closed)

sqlConnection.Close();

}

private void button2\_Click(object sender, EventArgs e)//добавление

{//Проверка на ненулевые значения у обязательных полей

if (!string.IsNullOrEmpty(numericUpDown1.Value.ToString()) &&

!string.IsNullOrEmpty(numericUpDown2.Value.ToString()) &&

!string.IsNullOrEmpty(textBox3.Text) &&

!string.IsNullOrEmpty(textBox4.Text) &&

!string.IsNullOrEmpty(textBox5.Text) &&

!string.IsNullOrEmpty(dateTimePicker1.Value.ToString()) &&

!string.IsNullOrEmpty(numericUpDown3.Value.ToString()) &&

!string.IsNullOrEmpty(numericUpDown4.Value.ToString()) &&

comboBox1.SelectedIndex >= 0 &&

comboBox2.SelectedIndex >= 0)

{

SqlCommand command = new SqlCommand("INSERT INTO [Table] (NumberOfLand, Area, F, I, O, PurchaseDate, Price, Donation, Electricity, Benefits)VALUES(@NumberOfLand, @Area, @F, @I, @O, @PurchaseDate, @Price, @Donation, @Electricity, @Benefits)", sqlConnection);

SqlCommand cmd = sqlConnection.CreateCommand();

cmd.CommandType = CommandType.Text;

command.Parameters.AddWithValue("NumberOfLand", numericUpDown1.Value);

command.Parameters.AddWithValue("Area", numericUpDown2.Value);

command.Parameters.AddWithValue("F", textBox3.Text);

command.Parameters.AddWithValue("I", textBox4.Text);

command.Parameters.AddWithValue("O", textBox5.Text);

command.Parameters.AddWithValue("PurchaseDate", dateTimePicker1.Value);

command.Parameters.AddWithValue("Price", numericUpDown3.Value);

command.Parameters.AddWithValue("Donation", numericUpDown4.Value);

command.Parameters.AddWithValue("Electricity", comboBox1.SelectedIndex);

command.Parameters.AddWithValue("Benefits", comboBox2.SelectedIndex);

command.ExecuteNonQuery();

displayData();

numericUpDown1.Value = 0;

numericUpDown2.Value = 0;

textBox3.ResetText();

textBox4.ResetText();

textBox5.ResetText();

dateTimePicker1.ResetText();

numericUpDown3.Value = 0;

numericUpDown4.Value = 0;

comboBox1.ResetText();

comboBox2.ResetText();

}

else

{

MessageBox.Show("Не все компоненты заполены", "Error!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

SqlCommand cmd1 = sqlConnection.CreateCommand();

cmd1.CommandType = CommandType.Text;

cmd1.CommandText = "SELECT COUNT(\*) FROM [Table] WHERE Id=" + Convert.ToString(numericUpDown9.Value);

int id = Convert.ToInt32(cmd1.ExecuteScalar());

if (Convert.ToBoolean(id))//Проверка на наличие Id в таблице, впринципе программа и без нее работала, но пользователь не узнавал о том, что ввел некорректный Id

{

SqlCommand cmd = sqlConnection.CreateCommand();

cmd.CommandType = CommandType.Text;

cmd.CommandText = "DELETE FROM [Table] WHERE Id=" + Convert.ToString(numericUpDown9.Value);

cmd.ExecuteNonQuery();

}

else

{

MessageBox.Show("Отсутствует элемент с данным Id", "Error!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

numericUpDown9.Value = 0;

displayData();

}

private void button3\_Click(object sender, EventArgs e)//Изменение

{

SqlCommand cmd1 = sqlConnection.CreateCommand();

cmd1.CommandType = CommandType.Text;

cmd1.CommandText = "SELECT COUNT(\*) FROM [Table] WHERE Id=" + Convert.ToString(numericUpDown10.Value);

int id = Convert.ToInt32(cmd1.ExecuteScalar());

if (Convert.ToBoolean(id))//Проверка на наличие Id в таблице, впринципе программа и без нее работала, но пользователь не узнавал о том, что ввел некорректный Id

{

//Проверка на ненулевые значения у обязательных полей

if (!string.IsNullOrEmpty(numericUpDown8.Value.ToString()) &&

!string.IsNullOrEmpty(numericUpDown7.Value.ToString()) &&

!string.IsNullOrEmpty(textBox8.Text) &&

!string.IsNullOrEmpty(textBox7.Text) &&

!string.IsNullOrEmpty(textBox6.Text) &&

!string.IsNullOrEmpty(numericUpDown6.Value.ToString()) &&

!string.IsNullOrEmpty(numericUpDown5.Value.ToString()) &&

comboBox4.SelectedIndex>=0 &&

comboBox3.SelectedIndex>=0)

{

SqlCommand cmd = sqlConnection.CreateCommand();

cmd.CommandType = CommandType.Text;

cmd.CommandText = "UPDATE [Table] SET NumberOfLand = " + numericUpDown8.Value.ToString() + ", Area = " + numericUpDown7.Value.ToString() + ", F = '" + textBox8.Text + "', I = '" + textBox7.Text + "', O = '" + textBox6.Text + "', Price = " + numericUpDown6.Value.ToString() + ", Donation = " + numericUpDown5.Value.ToString() + ", Electricity = " + comboBox4.SelectedIndex + ", Benefits = " + comboBox3.SelectedIndex + " WHERE Id = " + numericUpDown10.Value.ToString();

cmd.ExecuteNonQuery();//Само изменение полей

displayData();

numericUpDown8.Value = 0;//Обнуление полей, для дальнейшего использования

numericUpDown7.Value = 0;

textBox8.ResetText();

textBox7.ResetText();

textBox6.ResetText();

numericUpDown6.Value = 0;

numericUpDown5.Value = 0;

comboBox4.ResetText();

comboBox3.ResetText();

displayData();

}

else

{

MessageBox.Show("Не все компоненты заполены", "Error!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

else

{

MessageBox.Show("Отсутствует элемент с данным Id", "Error!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

numericUpDown10.Value = 0;

}

private void button4\_Click(object sender, EventArgs e) //удаление по условию

{

SqlCommand cmd = sqlConnection.CreateCommand();

cmd.CommandType = CommandType.Text;

cmd.CommandText = "DELETE FROM [Table] WHERE Area<" + Convert.ToString(numericUpDown11.Value);

cmd.ExecuteNonQuery();

displayData();

numericUpDown11.Value = 0;

}

private void button8\_Click(object sender, EventArgs e) //изменение по условию

{

if (comboBox6.SelectedIndex>=0 && comboBox5.SelectedIndex>=0)

{

SqlCommand cmd = sqlConnection.CreateCommand();

cmd.CommandType = CommandType.Text;

cmd.CommandText = "UPDATE [Table] SET Electricity=" + comboBox6.SelectedIndex + ", Benefits=" + comboBox5.SelectedIndex + " WHERE Donation < " + numericUpDown14.Value.ToString();

cmd.ExecuteNonQuery();

displayData();

comboBox6.ResetText();

comboBox5.ResetText();

}

numericUpDown14.Value = 0;

}

private void button5\_Click(object sender, EventArgs e)//Вывод строк удовлетворяющих условию

{

SqlCommand cmd = sqlConnection.CreateCommand();

cmd.CommandType = CommandType.Text;

cmd.CommandText = "SELECT \* FROM [Table] where Area > " + numericUpDown12.Value.ToString();

cmd.ExecuteNonQuery();

DataTable dt = new DataTable();

SqlDataAdapter da = new SqlDataAdapter(cmd);

da.Fill(dt);

dataGridView1.DataSource = dt;

}

private void button6\_Click(object sender, EventArgs e)//Вывод строк удовлетворяющих условию

{

SqlCommand cmd = sqlConnection.CreateCommand();

cmd.CommandType = CommandType.Text;

cmd.CommandText = "SELECT \* FROM [Table] where NumberOfLand = " + numericUpDown13.Value.ToString();

cmd.ExecuteNonQuery();

DataTable dt = new DataTable();

SqlDataAdapter da = new SqlDataAdapter(cmd);

da.Fill(dt);

dataGridView1.DataSource = dt;

}

private void button11\_Click(object sender, EventArgs e)//Подсчет кол-ва записей, удовлетворяющих условию. Функция C#

{

int buff = 0;

label31.ResetText();

for (int i = 0; i < databaseDataSet.Table.Rows.Count; i++)

{

if (Convert.ToInt32(databaseDataSet.Table[i][7]) > numericUpDown15.Value)

{

buff++;

}

}

label31.Text += buff.ToString();

}

private void button7\_Click\_1(object sender, EventArgs e) //Подсчет максимума. Функция C#

{

if (databaseDataSet.Table.Rows.Count > 0)

{

label29.ResetText();

int buff = Convert.ToInt32(databaseDataSet.Table[0][2]);

for (int i = 1; i < databaseDataSet.Table.Rows.Count; i++)

{

if (Convert.ToInt32(databaseDataSet.Table[i][2]) > buff)

{

buff = Convert.ToInt32(databaseDataSet.Table[i][2]);

}

}

label29.Text += buff.ToString();

}

}

private void button9\_Click\_1(object sender, EventArgs e)//Подсчет минимального. Функция C#

{

if (databaseDataSet.Table.Rows.Count > 0)

{

label28.ResetText();

int buff = Convert.ToInt32(databaseDataSet.Table[0][2]);

for (int i = 1; i < databaseDataSet.Table.Rows.Count; i++)

{

if (Convert.ToInt32(databaseDataSet.Table[i][2]) < buff)

{

buff = Convert.ToInt32(databaseDataSet.Table[i][2]);

}

}

label28.Text += buff.ToString();

}

}

private void button10\_Click\_1(object sender, EventArgs e)//Подсчет среднего по таблице. Функция C#

{

if (databaseDataSet.Table.Rows.Count > 0)

{

label30.ResetText();

double buff = Convert.ToDouble(databaseDataSet.Table[0][2]);

for (int i = 1; i < databaseDataSet.Table.Rows.Count; i++)

{

buff += Convert.ToDouble(databaseDataSet.Table[i][2]);

}

buff = buff / Convert.ToDouble(databaseDataSet.Table.Rows.Count);

label30.Text += buff.ToString();

}

}

}

}

**Документ «Program.cs»**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WindowsFormsApp3

{

static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

}

}

**Определение таблицы:**

CREATE TABLE [dbo].[Table] (

[Id] INT IDENTITY (1, 1) NOT NULL,

[NumberOfLand] INT NOT NULL,

[Area] INT NOT NULL,

[F] NVARCHAR(50) NOT NULL,

[I] NVARCHAR(50) NOT NULL,

[O] NVARCHAR(50) NOT NULL,

[PurchaseDate] DATE NOT NULL,

[Price] INT NOT NULL,

[Donation] INT NOT NULL,

[Electricity] NVARCHAR (50) NOT NULL,

[Benefits] NVARCHAR (50) NOT NULL,

PRIMARY KEY CLUSTERED ([Id] ASC)

);