**Министерство образования и науки РФ**

**Государственное образовательное учреждение**

**Высшего образования**

**«Владимирский государственный университет имени Александра Григорьевича и Николая Григорьевича Столетовых» (ВлГУ)**

**Кафедра «Физика и прикладная математика»**

Курсовая работа

По дисциплине «Прикладное программирование»

«Программный модуль агентство недвижимости»

Выполнил:

Ст. ПКсп-114

Приняла: Ст. преподаватель  
 Шишкина М. В.

Владимир 2017 г.

**Содержание.**

[**Введение.** 2](#_Toc447714027)

[**Глава1. Постановка задачи** 4](#_Toc447714028)

[**Глава 2. Выбор системы программирования** 5](#_Toc447714029)

[**Глава 3. Описание процесса проектирования приложения** 6](#_Toc447714030)

[**3.1. Взаимосвязь программных модулей** 6](#_Toc447714031)

[**3.1.1. Функция подборки рейса.** 6](#_Toc447714032)

[**3.1.2. Функция добавления рейса.** 7](#_Toc447714033)

[**3.1.3. Функция покупки билета.** 8](#_Toc447714034)

[**3.1.4. Функция очистки полей на вкладке “Добавление рейса”.** 9](#_Toc447714035)

[**3.1.5. Функции удаление рейса из listView1.** 9](#_Toc447714036)

[**3.1.5.1 Удаление первой строки.** 9](#_Toc447714037)

[**3.1.5.2 Удаление строки под номером.** 10](#_Toc447714038)

[**3.1.5.3 Удаление всех строк.** 10](#_Toc447714039)

[**3.1.6. Функция очистки полей информации о пассажире.** 11](#_Toc447714040)

[**3.2. Схема работы программы** 11](#_Toc447714041)

[**3.3. Описание интерфейса программы** 11](#_Toc447714042)

[**Глава 4. Инструкция для пользователя по работе с приложением** 16](#_Toc447714043)

[**Заключение.** 17](#_Toc447714044)

[**Список использованных информационных источников:** 18](#_Toc447714045)

[**Приложение 1** 19](#_Toc447714046)

# **Введение.**

Персональный компьютер, как известно, является универсальным устройством для обработки информации. Персональные компьютеры могут выполнять любые действия по обработке информации. Для этого необходимо составить для компьютера на понятном ему языке точную и подробную последовательность инструкций - программу, как надо обрабатывать информацию. Устанавливая программы, можно превратить компьютер в рабочее место бухгалтера или конструктора, дизайнера или ученого, писателя или агронома и т.д.

Одна из самых утомляющих работ – ручное заполнение тысяч одинаковых форм, содержащих, к примеру, информацию о недвижимости. Все крупные компании в данной сфере уже давно автоматизировали данный процесс.

В данной работе я рассмотрю все этапы создания программного модуля “Агентство недвижимости”.

# **Глава1. Постановка задачи**

В теме моей курсовой работы требуется создать приложение “Агентство недвижимости”, которое бы использовалось в риэлтерских конторах и облегчало работу риелтору. В качестве входных данных используется интересующий покупателя вид недвижимости, местоположение и кол-во комнат, персональные данные покупателя, на выходе покупатель получает чек.

# **Глава 2. Выбор системы программирования**

При выборе системы программирования я сравнивала несколько языков программирования. Основными характеристиками, позволяющими сравнивать языки программирования и выбирать наилучшие для решения той или иной задачи, являются: мощность, уровень и концептуальная целостность.

Среди объектно-ориентированных языков программирования, я выбрала C#. Причин этому несколько: интуитивно понятный интерфейс, удобство написания кода, C# для меня является самым удобным объектно-ориентированным языком программирования.

# **Глава 3. Описание процесса проектирования приложения**

# **3.1. Взаимосвязь программных модулей**

Итак, первым делом необходимо определиться с функционалом, которым будет обладать модуль. Во-первых, он должен иметь возможность добавлять недвижимость в базу данных. Во-вторых, он должен иметь возможность подбирать недвижимость, исходя из запроса покупателя. В-третьих, должна быть возможность удалять недвижимость из базы данных, чтобы внести в нее какие-либо изменения. И в-четвертых, должна реализовываться покупка недвижимости. Информация о покупателе должна содержать: Ф.И.О, серия и номер паспорта дату рождения, гражданство и срок действия предоставляемого документа. Информация о недвижимости должна содержать: вид недвижимости, местоположение, кол-во комнат, метраж, состояние, стоимость.

Поиск должен осуществляться по любому из полей информации о недвижимости.

Форма была разделена на 2 вкладки: 1 - Поиск недвижимости. 2 – Операции над недвижимостью.

# **3.1.1. Функция подборки недвижимости.**

После нажатия кнопки управление передается обработчику события “button1\_Click”. Сперва программа проверяет, все ли поля заполнены. Далее программа ищет совпадения данных введенных пользователем с элементами в ListView, при совпадении перекрашивает элемент в красный цвет и записывает данные хранящиеся в этом элементе, в listBox1. Если данные не совпадают с элементом в ListView, программа окрашивает элемент в черный цвет.

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

{

listBox1.Items.Clear();

if (comboBox2.Text == string.Empty)

return;

if (comboBox1.Text == string.Empty)

return;

if (comboBox3.Text == string.Empty)

return;

foreach (ListViewItem item in listView1.Items)

{

if ((item.SubItems[0].Text == comboBox2.Text) && (item.SubItems[1].Text == comboBox1.Text) && (item.SubItems[2].Text == comboBox3.Text))

{

item.ForeColor = Color.Red;

listBox1.Items.Add(item.SubItems[0].Text + " " + item.SubItems[1].Text + " " + item.SubItems[2].Text + " " + item.SubItems[3].Text + " " + item.SubItems[4].Text + " " + item.SubItems[5].Text);

}

else

item.ForeColor = Color.Black;

}

# }

# **3.1.2. Функция добавления недвижимости.**

После нажатия кнопки управление передается обработчику события button2\_Click. Далее инициализируется новый объект ListViewItem, выдается сообщение, в отдельном окне, о том, что Данные добавлены!, далее программа переходит в функцию записи данных в файл.

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

{

ListViewItem ListItem = new ListViewItem(comboBox4.Text);

ListItem.SubItems.Add(comboBox5.Text);

ListItem.SubItems.Add(comboBox7.Text);

ListItem.SubItems.Add(textBox8.Text);

ListItem.SubItems.Add(comboBox6.Text);

ListItem.SubItems.Add(maskedTextBox3.Text);

listView1.Items.Add(ListItem);

MessageBox.Show("Данные добавлены!", "Агентство недвижимости");

saveToolStripMenu();

}

private void saveToolStripMenu()//записывает данные в файл.

{

isWrite = true;

StringBuilder listViewContent = new StringBuilder();

foreach (ListViewItem item in this.listView1.Items)

{

foreach (ListViewItem.ListViewSubItem subItem in item.SubItems)

{

listViewContent.Append(subItem.Text); listViewContent.Append('|');

}

listViewContent.Append(Environment.NewLine);

}

using (TextWriter tw = new StreamWriter(UsersDirectory + "\\base.txt"))

{

tw.WriteLine(listViewContent.ToString());

tw.Close();

}

}

# **3.1.3. Функция покупки недвижимости.**

При двойном клике на интересующих элемент в listBox1, управление передается обработчику события listBox1\_MouseDoubleClick. Программа создает в директории запуска программы текстовый файл “pokupka.txt” и записывает в него информацию о приобретаемой недвижимости, которая берется из соответствующих элементов на форме. После очищается буфер и файл закрывается. Далее требуется нажать кнопку “Купить”, управление передастся обработчику события button7\_Click.

private void listBox1\_MouseDoubleClick(object sender, MouseEventArgs e)//создания текстового файла и информацией о покупатели и недвижимосте.

{

StreamWriter ticket = File.CreateText(UsersDirectory + "\\pokupka.txt");

ticket.WriteLine(" Данные о покупателе ");

ticket.WriteLine("Фамилия - " + textBox3.Text);

ticket.WriteLine("Имя - " + textBox4.Text);

ticket.WriteLine("Дата рождения - " + dateTimePicker1.Value.Date.ToShortDateString());

ticket.WriteLine(" Информация о покупке");

ticket.WriteLine(listBox1.SelectedItem);

ticket.Flush();

ticket.Close();

}

riсhTextBox1 очищается, открывается файл pokupka.txt. Пока свойство EndOfStream выдает значение false, на richTextBox1 выводится строчка из файла, когда свойство EndOfStream получает значение true, файл закрывается.

private void button7\_Click(object sender, EventArgs e)

{

richTextBox1.Text = "";

StreamReader ticket = File.OpenText(UsersDirectory + "\\pokupka.txt");

while (!ticket.EndOfStream)

richTextBox1.Text += ticket.ReadLine() + "\n";

ticket.Close();

# **3.1.4. Функция очистки полей на вкладке “Добавление рейса”.**

После нажатия кнопки управление передается обработчику события button3\_Click. Программа предается всем элементам значения, заданные по умолчанию и выдается в отдельном окне сообщение о том, что поля очищены. Ниже представлен фрагмент кода:

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

{

comboBox4.Text = "Вид недвижимости";

comboBox5.Text = "Местоположения";

comboBox6.Text = "Состояние";

comboBox7.Text = "Кол-во комнат";

textBox1.Text = "";

maskedTextBox3.Clear();

MessageBox.Show("Поля очищены!", "Агентство недвижимости");

}

# **3.1.5. Функции удаление рейса из listView1.**

# **3.1.5.1 Удаление первой строки.**

После нажатия кнопки управление передается обработчику события button4\_Click. Программа проверяет есть ли элементы в ListView1, если они есть удаляет нулевой элемент, далее выдает в отдельном окне сообщение о том, что первая строка удалена, и переходит в функцию записи данных в файл. Ниже представлен фрагмент кода:

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

{

if (listView1.Items.Count > 0)

listView1.Items.Remove(listView1.Items[0]);

MessageBox.Show("Первая строка удалена! ", "Агентство недвижимости");

saveToolStripMenu();

}

# **3.1.5.2 Удаление строки под номером.**

После нажатия кнопки управление передается обработчику события button5\_Click. Программа создает переменную строкового типа m и передает в нее значение из свойства Text объекта textBox1, далее создается переменная целого типа b, в нее передается методом конвертации значение переменной m. Создается переменная целого типа r, ей присваивается значение b-1. После проверяется есть элементы в listView1, если они есть, программа удаляет строку с номером r, в отдельном окне выдается сообщение о том, что строка с номером b удалена. Программа переходит в функцию записи данных в файл. Ниже представлен фрагмент кода:

private void button4\_Click(object sender, EventArgs e)

{

string m = textBox2.Text;

int b = Convert.ToInt32(m);

int r = b - 1;

if (listView1.Items.Count > 0)

listView1.Items.Remove(listView1.Items[r]);

MessageBox.Show("Строка номер " + b + " удалена! ", "Агентство недвижимости");

saveToolStripMenu();

}

# **3.1.5.3 Удаление всех строк.**

После нажатия кнопки управление передается обработчику события button6\_Click. Программа с помощью метода Items.Clear() очищает listView1, выдает сообщение о том, что все строки удалены и переходит в функцию записи данных в файл. Ниже представлен фрагмент кода:

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

{

listView1.Items.Clear();

MessageBox.Show("Все данные удалены! ", "Агентство недвижимости");

saveToolStripMenu();

}

# **3.1.6. Функция очистки полей информации о покупателе.**

После нажатия кнопки управление передается обработчику события button8\_Click. Программа придает соответствующим элементам на форме, значения по умолчанию. Ниже представлен фрагмент кода:

private void button8\_Click(object sender, EventArgs e)//очистка полей информации о покупателе

{

textBox3.Text = "Фамилия";

textBox4.Text = "Имя";

dateTimePicker1.Value = System.DateTime.Now;

richTextBox1.Text = "";

}

# **3.2. Схема работы программы**

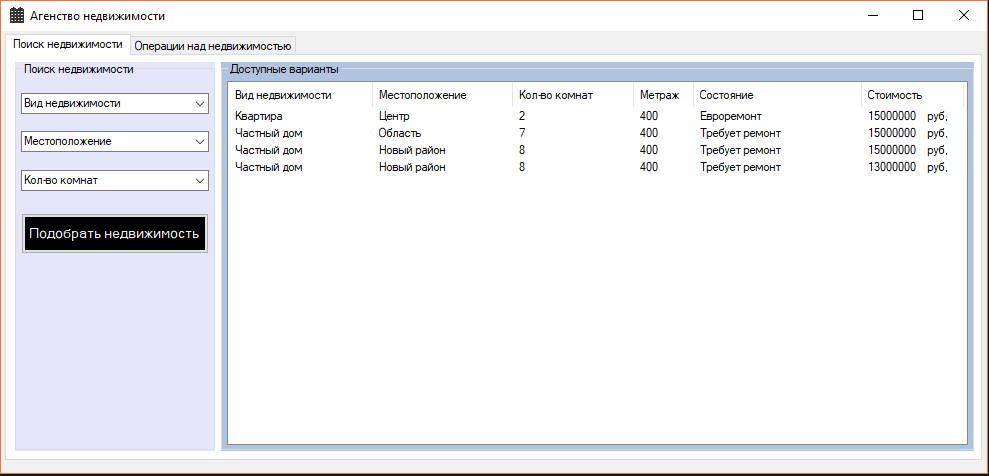
Покупатель

Риелтор

# **3.3. Описание интерфейса программы**

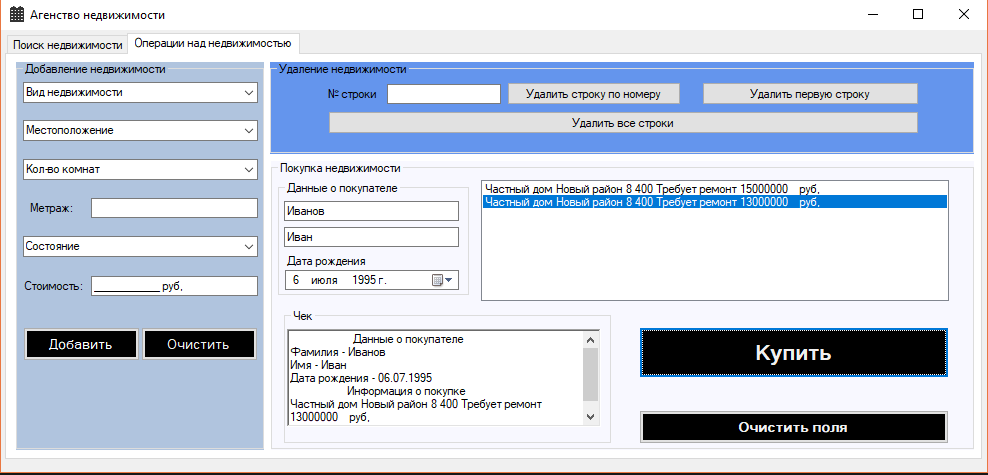
Проектирование интерфейса – не самая простая задач. Интерфейс должен быть интуитивен и понятен не только программисту, создавшему его, но и конечному пользователю, для которого собственно проект и разрабатывается. В ходе разработки интерфейс несколько раз менялся, пока не принял, на мой взгляд, достаточно приятный и понятный для стороннего пользователя вид:

На представленном ниже окне формы пользователь заполняет информацию о интересующей покупателя недвижимости и имеет возможность подобрать объекты, подходящие под критерии покупателя, а также видеть список всех доступных вариантов, если перед этим он заполнил поля, варианты, отвечающие параметрам поиска, будут подчеркнуты красным.



*Рис. 1. Интерфейс программы. Первая вкладка.*

На представленной ниже вкладке формы, пользователь имеет возможность добавить новую недвижимость в каталог, удалить из каталога недвижимость которая не является актуальной, заполнив данные о покупателе, выписать чек на покупку недвижимости.



*Рис. 2. Интерфейс программы. Вторая вкладка.*

# **Глава 4. Инструкция для пользователя по работе с приложением**

Вы открываете программу. Сначала требуется создать каталог недвижимости, для этого вам нужно перейти на вкладку “Добавление недвижимости” заполнить все поля и нажать кнопку “Добавить”, после нажатия кнопки на вкладке “Доступные варианты” появится внесенный объект. Если вы ввели какие-то данные не верно, на вкладке “Удаления недвижимости” вы можете удалить объект, введя номер строки с объектом, который требуется удалить, в специальное поле, после нажать кнопку “Удалить строку”.

При покупке недвижимости, поинтересуйтесь у покупателя, какой вид недвижимости он хочет приобрести, желаемое местоположение, кол-во комнат, заполните поля на вкладке “Поиск недвижимости" и нажмите кнопку подобрать недвижимость, после нажатия кнопки, перейдите на вкладку “Доступные варианты”, объекты подходящие по критериям покупателя будут выделены красным цветов, если какой-то из объектов заинтересовал покупателя, перейдите на вкладку “Покупка недвижимости”, заполните все поля исходя из данных покупателя, выберите нужный объект (в списке на вкладке) двойным щелчком и нажмите кнопку “Купить”. Чек выдается в электронном виде.

# **Заключение.**

В ходе выполнения этой работы был разработан программный модуль, предоставляющий полный функционал агентства недвижимости предоставляющий возможность покупки недвижимости и редактирования каталога объектов недвижимости. Я научилась пользоваться элементом listView. Также были закреплены мои навыки работы с файлами, строками и формами в С#.

# **Список использованных информационных источников:**

1. CLR via C#. Программирование на платформе Microsoft. NET Framework 4.5 на языке C#. 4-е изд. Автор: Дж. Рихтер
2. Язык программирования C#. Классика Computer Science. 4-е изд.

Авторы: А. Хейлсберг, М.Торгерсен, С.Вилтамут, П. Голд

1. [www.habrahabr.ru/hub/csharp/](http://www.habrahabr.ru/hub/csharp/) - Хабрахабр, информация по языку C#
2. [https://msdn.microsoft.com/](https://msdn.microsoft.com/ru-RU/library/mt472912(v=vs.110).aspx) - Microsoft Developer Network.
3. Airline Tickets – приложение для покупки авиабилетов на iOS.
4. C#. Объектно-ориентированное программирование. Учебный курс. Автор: Васильев А.Н.
5. <http://bourabai.ru/alg/oop11.htm#10> – Объектно-ориентированное программирование.

# **Приложение 1**

Код программы:

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 System.Diagnostics;

using System.IO;

using System.Xml.Linq;

using System.Globalization;

namespace agentstvonedvizh

{

public partial class Form1 : Form

{

string UsersDirectory = Environment.CurrentDirectory;

public Form1()

{

InitializeComponent();

parseItems();

}

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

{

Close();

}

bool isWrite = false;

private void saveToolStripMenu()//записывает данные в файл.

{

isWrite = true;

StringBuilder listViewContent = new StringBuilder();

//StringBuilder - Предоставляет изменяемую строку символов.

// ListView - Класс.

//Предоставляет элемент управления списка Windows, отображающий коллекцию элементов, которые могут быть выведены на экран при использовании одного из четырех различных представлений.

//ListViewItem - Инициализирует новый экземпляр класса ListViewItem значениями по умолчанию.

//ListViewItem.ListViewSubItem - Инициализирует новый экземпляр класса ListViewItem.ListViewSubItem значениями по умолчанию.

foreach (ListViewItem item in this.listView1.Items)

{

foreach (ListViewItem.ListViewSubItem subItem in item.SubItems)

{

listViewContent.Append(subItem.Text); // метод класса StringBuilder, добавляет строку к текущей строке.

listViewContent.Append('|');

}

listViewContent.Append(Environment.NewLine);

}

//TextWriter - Представляет модуль записи, который может записывать последовательные наборы символов.Это абстрактный класс.

//StreamWriter - Реализует TextWriter для записи символов в поток в определенной кодировке.

using (TextWriter tw = new StreamWriter(UsersDirectory + "\\base.txt"))

{

tw.WriteLine(listViewContent.ToString()); // Возвращает текущую строку в виде объекта System.String и записывает в файл.

tw.Close();

}

}

private void parseItems()//выводит данные из файла в таблицу ( listView ).

{

listView1.Items.Clear(); // Удаляет все составляющие элементы и столбцы.

using (StreamReader sReader = new StreamReader("base.txt")) //Реализует объект TextReader, который считывает символы из потока байтов в определенной кодировке.

{

string line;

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

{

string[] values = line.Split('|'); // Возвращает строковый массив, содержащий подстроки данного экземпляра, разделенные элементами заданной строки или массива знаков Юникода.

listView1.Items.Add(new ListViewItem(values));

}

}

}

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

{

listBox1.Items.Clear();

if (comboBox1.Text == string.Empty)

return;

if (comboBox2.Text == string.Empty)

return;

if (comboBox3.Text == string.Empty)

return;

foreach (ListViewItem item in listView1.Items)

{

if ((item.SubItems[0].Text == comboBox2.Text) && (item.SubItems[1].Text == comboBox1.Text) && (item.SubItems[2].Text == comboBox3.Text))

{

item.ForeColor = Color.Red;

listBox1.Items.Add(item.SubItems[0].Text + " " + item.SubItems[1].Text + " " + item.SubItems[2].Text + " " + item.SubItems[3].Text + " " + item.SubItems[4].Text + " " + item.SubItems[5].Text);

}

else

item.ForeColor = Color.Black;

}

}

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

{

ListViewItem ListItem = new ListViewItem(comboBox4.Text);

ListItem.SubItems.Add(comboBox5.Text);

ListItem.SubItems.Add(comboBox6.Text);

ListItem.SubItems.Add(textBox1.Text);

ListItem.SubItems.Add(comboBox7.Text);

ListItem.SubItems.Add(maskedTextBox1.Text);

listView1.Items.Add(ListItem);

MessageBox.Show("Данные добавлены!", "Агентство недвижимости");

saveToolStripMenu();

}

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

{

comboBox4.Text = "Вид недвижимости";

comboBox5.Text = "Местоположения";

comboBox6.Text = "Состояние";

comboBox7.Text = "Кол-во комнат";

textBox1.Text = "";

maskedTextBox1.Clear();

MessageBox.Show("Поля очищены!", "Агентство недвижимости");

}

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

{

listView1.Items.Clear();

MessageBox.Show("Все данные удалены! ", "Агентство недвижимости");

saveToolStripMenu();

}

private void button4\_Click(object sender, EventArgs e)

{

string m = textBox2.Text;

int b = Convert.ToInt32(m);

int r = b - 1;

if (listView1.Items.Count > 0)

listView1.Items.Remove(listView1.Items[r]);

MessageBox.Show("Строка номер " + b + " удалена! ", "Агентство недвижимости");

saveToolStripMenu();

}

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

{

if (listView1.Items.Count > 0)

listView1.Items.Remove(listView1.Items[0]);

MessageBox.Show("Первая строка удалена! ", "Агентство недвижимости");

saveToolStripMenu();

}

private void textBox3\_MouseDown(object sender, MouseEventArgs e)

{

textBox3.Text = "";

}

private void textBox4\_MouseDown(object sender, MouseEventArgs e)

{

textBox4.Text = "";

}

private void button7\_Click(object sender, EventArgs e)

{

richTextBox1.Text = "";

StreamReader ticket = File.OpenText(UsersDirectory + "\\pokupka.txt");

while (!ticket.EndOfStream)

richTextBox1.Text += ticket.ReadLine() + "\n";

ticket.Close();

}

private void listBox1\_MouseDoubleClick(object sender, MouseEventArgs e)//создания текстового файла и информацией о покупатели и недвижимосте.

{

StreamWriter ticket = File.CreateText(UsersDirectory + "\\pokupka.txt");

ticket.WriteLine(" Данные о покупателе ");

ticket.WriteLine("Фамилия - " + textBox3.Text);

ticket.WriteLine("Имя - " + textBox4.Text);

ticket.WriteLine("Дата рождения - " + dateTimePicker1.Value.Date.ToShortDateString());

ticket.WriteLine(" Информация о покупке");

ticket.WriteLine(listBox1.SelectedItem);

ticket.Flush();

ticket.Close();

}

private void button8\_Click(object sender, EventArgs e)//очистка полей информации о покупателе

{

textBox3.Text = "Фамилия";

textBox4.Text = "Имя";

dateTimePicker1.Value = System.DateTime.Now;

richTextBox1.Text = "";

}

private void Form1\_FormClosing(object sender, FormClosingEventArgs e)

{

if (isWrite)

{

StreamReader sReader = new StreamReader("base.txt");

string str = sReader.ReadToEnd();

sReader.Close();

TextWriter tw = new StreamWriter(UsersDirectory + "\\base.txt");

tw.Write(str.Remove(str.Length - 4, 4));

tw.Close();

}

}

}

}