# Лабораторная работа №2 – Windows Forms

**1 Цель работы**

Изучить основы построения Windows приложений на Visual Studio.NET на основе событийно-реакционной модели приложений. Обучиться основным приемам использования проекта Windows-forms.

1. **Порядок выполнения работы**

**–** Прочесть краткие теоретические сведения*.*

**–** Выполнить задания раздела.

**–** Составить отчет о лабораторной работе и защитить его у преподавателя.

**3 Теоретическая часть**

**3.1 Событийно-управляемое программирование**

В основу Windows положен принцип событийного управления. Это значит, что и сама система, и приложения после запуска ожидают действий пользователя и реагируют на них заранее заданным образом. Любое действие пользователя (нажатие клавиши на клавиатуре, щелчок кнопкой мыши, перемещение мыши) называется событием.

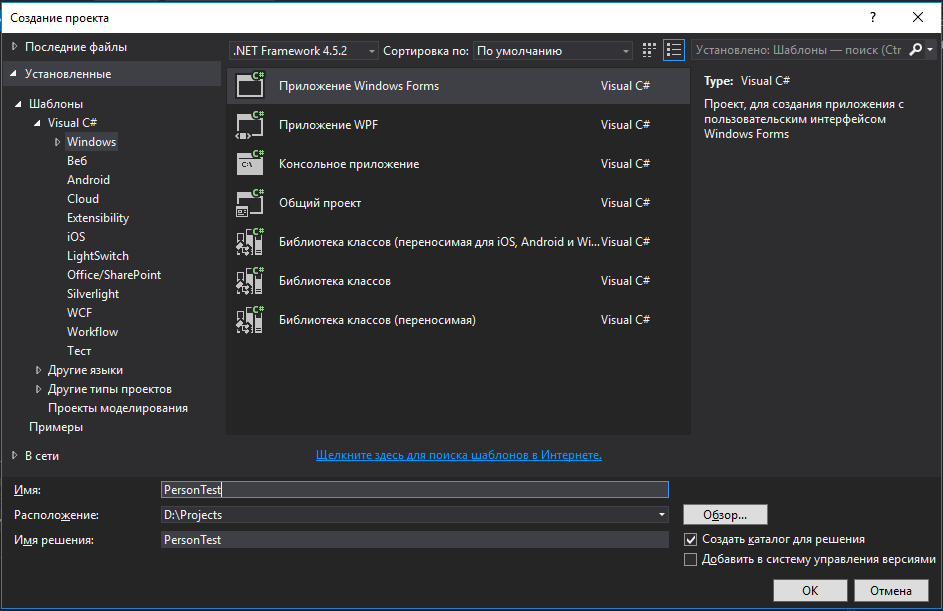
Событие воспринимается Windows и преобразуется в сообщение — запись, содержащую необходимую информацию о событии (например, какая клавиша была нажата, в каком месте экрана произошел щелчок мышью). Сообщения могут поступать не только от пользователя, но и от самой системы, а также от активного или других приложений. Определен достаточно широкий круг стан­дартных сообщений, образующий иерархию, кроме того, можно определять соб­ственные сообщения.

Сообщения поступают в общую очередь, откуда распределяются по очередям приложений. Каждое приложение содержит цикл обработки сообщений, кото­рый выбирает сообщение из очереди и через операционную систему вызывает подпрограмму, предназначенную для его обработки. Таким образом, Windows-приложение состоит из главной программы, обеспечивающей инициа­лизацию и завершение приложения, цикла обработки сообщений и набора обра­ботчиков событий.

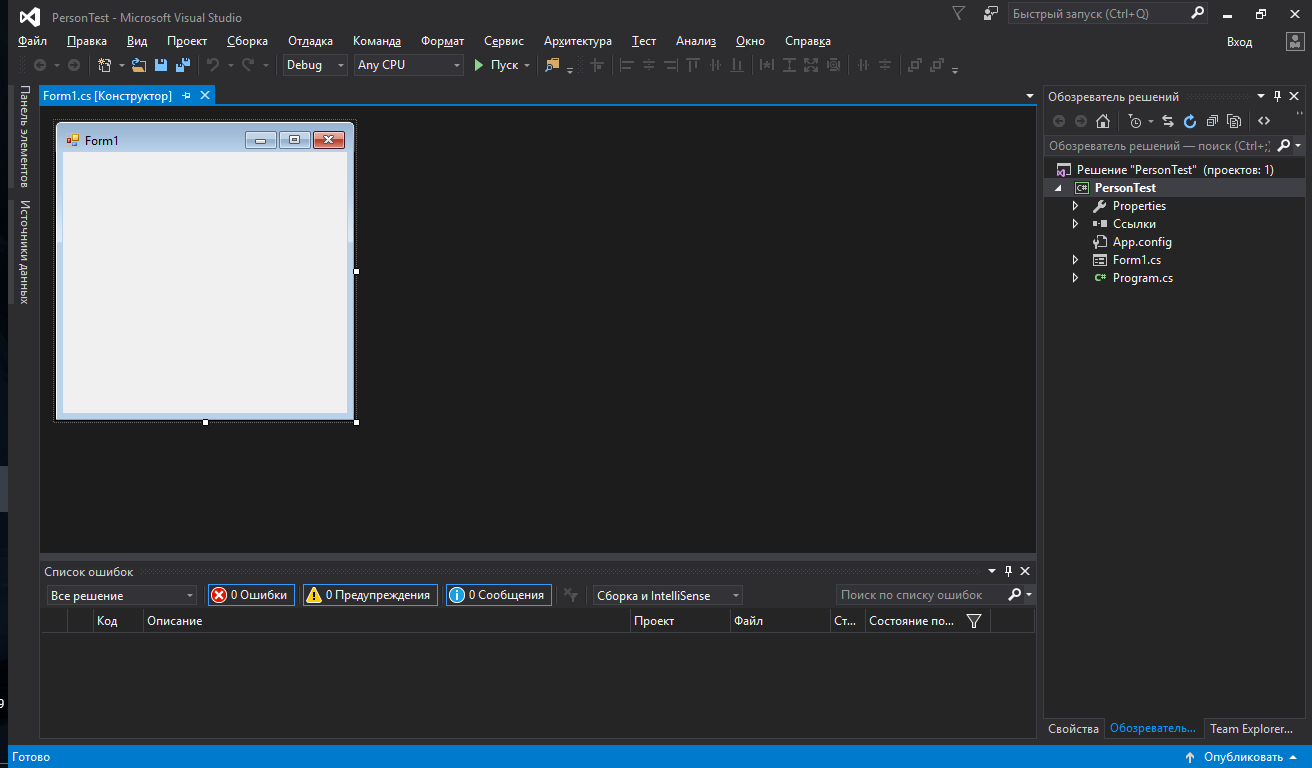
Среда Visual [Studio.NET](http://Studio.NET) содержит удобные средства разработки Windows-при­ложений, выполняющие вместо программиста рутинную работу — создание шаб­лонов приложения и форм, заготовок обработчиков событий, организацию цикла обработки сообщений.

**3.2 Создание Windows-приложения**

Создадим новый проект (Файл → Создать → Проект), выбрав шаблон ***Приложение Windows Forms*** (рисунок 1). После этого среда сформирует шаблон Windows-приложения (рисунок 2).



**Рисунок 1** – Окно выбора шаблона программы



**Рисунок 2** – Окно Windows-приложения

Среда создает не только заготовку формы, но и шаблон кода приложения. Пе­рейти к нему можно, щелкнув в окне ***Обозреватель решений*** (Вид → Обозреватель решений) правой кнопкой мыши на файле Forml.cs и выбрав в контекстном меню команду ***Перейти к коду***. При этом откроется вкладка с кодом формы, который приведен ниже:

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 PersonTest

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

}

}

Приложение начинается с директив использования пространств имен библиотеки .NET.

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;

Далее следует описание пространства имен (namespace), описание класса Form1, наследника класса Form, и описание конструктора класса Form1() с единственным оператором InitializeComponent(), осуществляющим инициализацию компонентов фармы.

Добавим с проект новый класс (клик правой кнопкой по проекту → Добавить → Класс), назовем класс Person, и в открывшемся окне редактора кода, модифицируем новый класс следующим образом:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace PersonTest

{

class Person

{

public string Name;

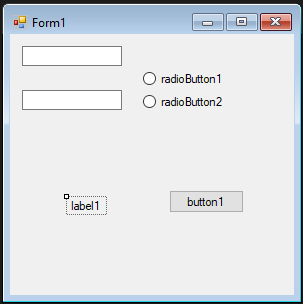
public int Age;

public char Gender;

}

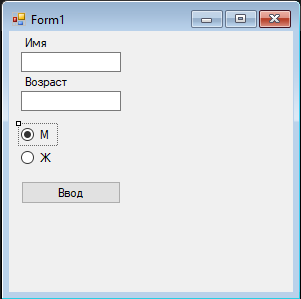
}

Затем перейдём в конструктор формы и добавим на форму новые элементы, а именно кнопку(Button), 2 поля ввода(TextBox), 2 радиокнопки и одну метку(Label)



**Рисунок 3** – Форма программы

Поскольку элементы формы являются экземплярами соответствующих классов, их свойства доступны для чтения и модификации из кода или через окно свойств (Вид → Окно свойств). Сгруппируем элементы на форме, добавим ещё метку и поменяем свойство Text у элементов при помощи окна свойств. Также выставим у одной радиокнопки свойство Checked, отвечающее за выделение, равным true.



**Рисунок 4** – Измененная форма программы

Далее модифицируем код формы следующим образом (обратите внимание на то, что обработчики событий button1\_Click генерируются средой автоматически, при двойном щелчке на соответствующие кнопки):

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace PersonTest

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

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

{

if (textBox1.Text != "" && textBox2.Text != "")

{

Person test = new Person();

test.Name = textBox1.Text;

test.Age = Convert.ToInt32(textBox2.Text);

test.Gender = radioButton1.Checked ? Convert.ToChar(radioButton1.Text) : Convert.ToChar(radioButton2.Text);

MessageBox.Show($"Имя: {test.Name} Возраст: {test.Age.ToString()} Пол: {test.Gender}");

}

else MessageBox.Show("Не все поля заполнены!");

}

}

}

Данный код проверяет, что оба поля для ввода не пусты. Если условие истинно, создаётся экземпляр класса Person и полям присваиваются значения свойств соответствующих элементов формы.

Обратите внимание:

- Команда MessageBox.Show выводит информационное сообщение, принятое в качестве текстового аргумент;

- Поскольку свойство Text элементов формы представлено текстовым типом, необходимо использовать конвертацию типов для присвоения значений полям класса Person;

- В поле ввода «Возраст» должны быть введены только цифры, в противном случае возникнет исключение;

- Для присвоения значения пола используется тернарный оператор, эквивалентный следующей записи:

if (radioButton1.Checked) test.Gender = Convert.ToChar(radioButton1.Text);

else test.Gender = Convert.ToChar(radioButton2.Text);

**3.3 Некоторые приемы работы с формой**

**3.3.1 Получение доступа к методам и свойствам формы из ее кода**

Получить доступ к методам и свойствам формы можно, используя ключевое слово ***this***. Например:

this.Text = "Я заголовок формы!";

Кроме того, можно создать новые экземпляры форм во время выполнения, описав переменную, представляющую тип формы, и создав экземпляр данной формы.

Свойства форм Windows

Внешний вид пользовательского интерфейса очень важен для приложения. Плохо разработанный пользовательский интерфейс трудно понять. Его изучение займет много времени и потребует больших затрат. Изменить внешний вид пользовательского интерфейса можно с помощью свойств форм Windows.

Формы Windows содержат множество свойств, позволяющих настраивать их вне­шний вид и поведение. Просматривать и изменять эти свойства можно в окне Pro­perties конструктора.

В таблице 1 перечислены некоторые свойства форм Windows, отвечающие за вне­шний вид и поведение приложения. Обратите внимание, что здесь представлен не полный список свойств форм Windows, а их избранная подгруппа.

Таблица 1 – Некоторые свойства класса *Form*

|  |  |
| --- | --- |
| *Свойство* | *Описание* |
| ***Name*** | Задает имя классу ***Form,*** показанному в конструкторе. Данное свойство задается исключительно во время разработки |
| ***BackColor*** | Указывает цвет фона формы |
| ***Backgroundlmage*** | Указывает фоновое изображение формы |
| ***Background-Image Layout*** | Определяет, как изображение, указанное свойством ***Backgroundlmage,*** будет располагаться в форме. Если фоновое изображение не выбрано, данное свойство игнорируется |
| ***ControlBox*** | Определяет, имеет ли форма оконное меню Control/System |
| ***Cursor*** | Указывает курсор, который появляется при наведении мыши на форму |
| ***Enabled*** | Указывает, может ли форма принимать ввод от пользователя. Если свойству ***Enabled*** задано значение ***False,*** все элементы управления формы также блокируются |
| *Font* | Задает шрифту формы значение по умолчанию. Если отдельно не указать значение свойства *Font* элементов управления формы, они примут это же значение по умолчанию |
| *Fore Color* | Указывает цвет переднего плана формы, то есть цвет выводимого текста. Если отдельно не указать значение свойства *Fore Color* элементов управления формы, они примут то же значение |
| *FormBorderStyle* | Указывает вид и поведение границы и строки заголовка формы |
| *HelpButton* | Указывает, есть ли у формы кнопка Help |
| *Icon* | Указывает значок, расположенный в заголовке формы |
| *Location* | Когда свойству *StartPosition* задано значение *Manual,* это свойство указывает исходное положение формы относительно верхнего левого угла экрана |
| *MaximizeBox* | Указывает, есть ли у формы кнопка MaximizeBox |
| *MaximumSize* | Устанавливает максимальный размер формы. Если задать этому свойству размер *0; 0,* у формы не будет верхнего ограничения размера |
| *MinimizeBox* | Указывает, есть ли у формы кнопка MinimizeBox |
| *MinimumSize* | Устанавливает минимальный размер формы, который пользователь может задать |
| *Opacity* | Устанавливает уровень непрозрачности или прозрачности формы от *0* до *100%.* Форма, непрозрачность которой составляет 100%, -полностью непрозрачна, а форма, имеющая 0% непрозрачности, — наоборот, полностью прозрачна |

**3.3.2 Размещение формы поверх пользовательского интерфейса**

Порой вам нужно будет расположить одну форму поверх других форм в пользова­тельском интерфейсе. К примеру, вы разработали форму, в которой представлена важная информация о выполнении программы, и хотите, чтобы она всегда была у пользователя перед глазами. Можно сделать так, чтобы форма всегда располагалась поверх пользовательского интерфейса, задав свойству *TopMost* значение *True.* Если *TopMost* присвоить значение *True,* форма всегда будет расположена поверх всех осталь­ных форм, свойству *TopMost* которых задано значение *Fake,* являющееся значением по умолчанию. Обратите внимание, что если значение *True* свойства *TopMost* задано нескольким формам, они могут закрыть друг друга.

**3.3.3 Прозрачность и непрозрачность форм**

С помощью свойства *Opacity* можно создавать удивительные визуальные эффекты внутри формы. Свойство *Opacity* определяет степень непрозрачности формы. При настройке значения непрозрачности в окне Properties оно может варьироваться от 0 до 100 *%.* Непрозрачность в 100 *%* означает, что форма полностью непрозрачна (сплошная и видимая), а значение 0 *% —* что форма полностью прозрачна. Значения между 0 и 100 *%* делают форму частично прозрачной.

Кроме того, свойство *Opacity* можно задать в коде. При этом ему задается значение от 0 до 1, где 0 означает, что форма будет полностью прозрачной, а 1 — полностью непрозрачной.

aForm.Opacity =0.5;

Свойство *Opacity* может пригодиться, когда необходимо расположить одну форму на переднем плане, наблюдая при этом за работой формы на заднем плане, или же для создания интересных визуальных эффектов. Чаще всего элемент управления наследует прозрачность той формы, которая его содержит.

**3.3.4 Создание непрямоугольных форм Windows**

Возможно, для получения дополнительных визуальных эффектов вы захотите создать непрямоугольные формы, например овальную или форму в виде логотипа компании. Хотя создать непрямоугольную форму довольно просто, относительно окончательного вида и поведения формы существует несколько соображений.

Создать непрямоугольную форму можно, задав свойство *Region* формы в обработ­чике события *FormLoad.* Поскольку изменения в форме фактически происходят во время выполнения, вы не сможете их увидеть во время разработки. Таким образом, вам придется несколько раз запускать приложение и смотреть на форму, пока не настроите внешний вид и расположение элементов управления.

Свойство *Region* является экземпляром класса *System.Drawing.Region.* Данный класс представляет собой область экрана, которая является внутренней частью гра­фической формы, определенной с помощью прямоугольников и графических путей. Самый простой способ создать непрямоугольную форму — это создать новый экзем­пляр класса *GraphicsPath,* а затем с его помощью создать новый класс *Region.* Вот простой пример кода:

System.Drawing.Drawing2D.GraphicsPath myPath = new System.Drawing.Drawing2D.GraphicsPath();

// Добавление эллипса, вписанного в прямоугольную форму // заданной ширины и высоты

myPath.AddEllipse(0, 0, this.Width, this.Height);

// Создание нового класса Region из GraphicsPath Region myRegion = new Region(myPath);

// Присвоение свойству Region формы нового региона this.Region = myRegion;

Поскольку непрямоугольные формы будут иметь ограниченные гра­ницы (если таковые вообще будут), было бы неплохо задать свойству ***FormBorderStyle*** формы значение ***None.*** Так, если какие-либо части формы пересекут исходные границы прямоугольника формы, они не изменятся и не примут нежелательную форму. Однако если задать свойству ***FormBorderStyle*** значение ***None,*** пользователь не сможет изменить размеры, передвинуть или закрыть форму, а вам придется встроить эти возможности в свой проект.

**3.4 Создание непрямоугольной формы:**

1.В окне Properties задайте свойству ***FormBorderStyle*** значение ***None.***

2. Дважды щелкните форму в конструкторе, чтобы открыть обработчик события по умолчанию ***Form\_Load.***

3. В обработчике события ***FormLoad*** создайте новый экземпляр класса ***Region,*** как показано в предыдущем примере.

4. Если нужно, добавьте в форму функциональность для закрытия, передвижения или изменения размеров, поскольку, возможно, пользователь не сможет получить доступ к границам или строке заголовка формы.

5. Задайте форму как стартовую и нажмите F5, чтобы увидеть форму Настройте по мере необходимости внешний вид и расположение элементов управления.

**4 Общее задание для выполнения на паре**

Разработать форму регистрации пользователя, которая содержит следующие элементы:

- Текстовое поле «Имя пользователя»;

- Текстовое поле «Логин пользователя»;

- Текстовое поле «Пароль пользователя»;

**-** Чекбокс (галочка) «Я принимаю условия пользовательского соглашения».

- Кнопка «Зарегистрироваться», при нажатии на которую создается экземпляр класса «User», поля класса заполняются значениями свойств элементов формы, если:

- Заполнены все поля;

- Длина пароля превышает 6 символов;

- Выбран чекбокс «Я принимаю условия пользовательского соглашения»;

В противном случае выводится информационное сообщение, содержащее текст:

- «Не все поля заполнены»;

- «Пароль недостаточной длины»;

- «Подтвердите согласие с пользовательским соглашением».

**5 Индивидуальные задания для выполнения работы**

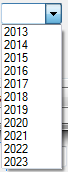
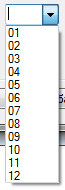
**Вариант 1**

На основе результатов предыдущей лабораторной работы разработать приложение Windows Forms, хранящее информацию о некоторых транспортных средствах, которое будет принимать значения модели(string), мощности(л/с)(int), года выпуска(int) и цвета(string) некоторого транспортного средства, после чего, по запросу, выводить эти значения в элемент RichTextBox, причем каждый с новой строки.

**Вариант 2**

На основе результатов предыдущей лабораторной работы разработать приложение Windows Forms, хранящее информацию о кредитных картах, которое будет принимать значения владелец(string); срок службы карты: месяц(int),год(int), заполнение которых реализовано при помощи элемента управления ComboBox(рис 1); а также номер карты(int). После чего, по запросу, выводить эти значения в элемент TextBox, причем каждый с новой строки.

Рисунок 1:



**Вариант 3**

На основе результатов предыдущей лабораторной работы разработать приложение Windows Forms, хранящее информацию о сотовых телефонах, которое будет принимать значения фирмы производителя(string), модели(string), количество(int) и метку о том является ли товар Б/У(bool), при помощи элемента управления CheckBox(Рис. 2), после чего, по запросу, выводить эти значения в элемент TextBox, причем каждый с новой строки.

Рисунок 2: 

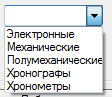
**Вариант 4**

На основе результатов предыдущей лабораторной работы разработать приложение Windows Forms, хранящее информацию о результатах футбольных матчей, которое будет принимать значения название команды встречающих(string), название команды гостей(string), счет сыгранного матча(int):(int), после чего, по запросу, выводить эти значения в элемент RichTextBox, причем каждый с новой строки.

**Вариант 5**

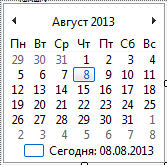
На основе результатов предыдущей лабораторной работы разработать приложение Windows Forms, хранящее информацию о часах, которое будет принимать значения тип часов(string) заполнение которого реализовано при помощи элемента управления ComboBox(рис 3), срок гарантии(int), цена(int). После чего, по запросу, выводить эти значения в элемент RichTextBox, причем каждый с новой строки.

Рисунок 3:

****

**Вариант 6**

На основе результатов предыдущей лабораторной работы разработать приложение Windows Forms, хранящее информацию о продуктах на складе овощей, которое будет принимать значения название(string); дата завоза заполнение которой реализовано при помощи элемента управления monthCalendar (рис 4) и его свойсв SelectionEnd.Date; а также цена за кг(int). После чего, по запросу, выводить эти значения в элемент TextBox, причем каждый с новой строки.

Рисунок 4: 

**Вариант 7**

На основе результатов предыдущей лабораторной работы разработать приложение Windows Forms, хранящее информацию о телепрограмме на день, которое будет принимать значения название телепередачи(string), жанр(string), время показа(int):(int), после чего, по запросу, выводить эти значения в элемент TextBox, причем каждый с новой строки.

**Вариант 8**

На основе результатов предыдущей лабораторной работы разработать приложение Windows Forms, хранящее информацию о складе бытовой техники, которое будет принимать значения срок вид устройства, заполнение которых реализовано при помощи элемента управления ComboBox(рис 5); название фирмы производителя(string); модель(string), метку о том является ли товар Б/У(bool), при помощи элемента управления CheckBox(Рис. 6). После чего, по запросу, выводить эти значения в элемент TextBox, причем каждый с новой строки.

Рисунок 5:

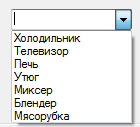


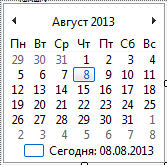
Рисунок 6: 

**Вариант 9**

На основе результатов предыдущей лабораторной работы разработать приложение Windows Forms, хранящее информацию о транспортных перевозках, которое будет принимать значения город отправки(string), город назначения(string), расстояние(int), срок доставки(int), после чего, по запросу, выводить эти значения в элемент RichTextBox, причем каждый с новой строки.

**Вариант 10**

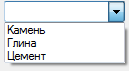
На основе результатов предыдущей лабораторной работы разработать приложение Windows Forms, хранящее информацию о зернохранилище, которое будет принимать значения сорт пшеницы(string); дата сбора урожая заполнение которой реализовано при помощи элемента управления monthCalendar (рис 7) и его свойсв SelectionEnd.Date; а также объем в тоннах(int). После чего, по запросу, выводить эти значения в элемент TextBox, причем каждый с новой строки.

Рисунок 7: 

**Вариант 11**

На основе результатов предыдущей лабораторной работы разработать приложение Windows Forms, хранящее информацию о продукции завода по производству плитки, которое будет принимать значения тип плитки(string); материал(string), заполнение которого реализовано при помощи элемента управления ComboBox(рис 8); цена за штуку(int). После чего, по запросу, выводить эти значения в элемент RichTextBox, причем каждый с новой строки.

Рисунок 8:

****

**Вариант 12**

На основе результатов предыдущей лабораторной работы разработать приложение Windows Forms, хранящее информацию о исторических постройках, которое будет принимать значения название постройки(string), примерный год создания(int), местоположение(срана)(string) некоторого транспортного средства, после чего, по запросу, выводить эти значения в элемент RichTextBox, причем каждый с новой строки.

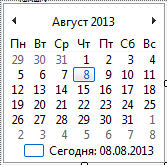
**Вариант 13**

На основе результатов предыдущей лабораторной работы разработать приложение Windows Forms, хранящее информацию о компьютерных запчастях, которое будет принимать значения фирмы производителя(string), наименования(string), модель(string), цена за шт(int) и метку о том является ли товар Б/У(bool), при помощи элемента управления CheckBox(Рис. 9), после чего, по запросу, выводить эти значения в элемент TextBox, причем каждый с новой строки.

Рисунок 9: 

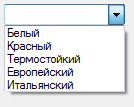
**Вариант 14**

На основе результатов предыдущей лабораторной работы разработать приложение Windows Forms, хранящее информацию о продуктах на складе фруктов, которое будет принимать значения название(string); дата завоза заполнение которой реализовано при помощи элемента управления monthCalendar (рис 10) и его свойсв SelectionEnd.Date; а также цена за кг(int). После чего, по запросу, выводить эти значения в элемент TextBox, причем каждый с новой строки.

Рисунок 10: 

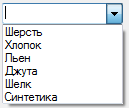
**Вариант 15**

На основе результатов предыдущей лабораторной работы разработать приложение Windows Forms, хранящее информацию о продукции завода по производству кирпича, которое будет принимать значения тип кирпича(string), заполнение которого реализовано при помощи элемента управления ComboBox(рис. 11); материал(string), цена за штуку(int). После чего, по запросу, выводить эти значения в элемент RichTextBox, причем каждый с новой строки.

Рисунок 11:

**Вариант 16**

На основе результатов предыдущей лабораторной работы разработать приложение Windows Forms, хранящее информацию о продукции цеха по производству ковров, которое будет принимать значения тип материала(string), заполнение которого реализовано при помощи элемента управления ComboBox(рис 12); тип рисунка(string), страна производитель(string). После чего, по запросу, выводить эти значения в элемент RichTextBox, причем каждый с новой строки.

Рисунок 12: 

**Вариант 17**

На основе результатов предыдущей лабораторной работы разработать приложение Windows Forms, хранящее информацию о работе столярного цеха, которое будет принимать значения наименования изделия(string), наименования(string), материал(string), сложность работы(string), и метку о том находиться ли предмет на реставрации(bool), при помощи элемента управления CheckBox(Рис. 13), после чего, по запросу, выводить эти значения в элемент TextBox, причем каждый с новой строки.

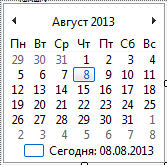
Рисунок 13:

**Вариант 18**

На основе результатов предыдущей лабораторной работы разработать приложение Windows Forms, хранящее информацию о предоставляемом перечне каналов компанией спутникового ТВ, которое будет принимать значения название канала (string), тип канала (string), длительность эфира(int), после чего, по запросу, выводить эти значения в элемент RichTextBox, причем каждый с новой строки.

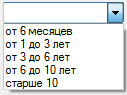
**Вариант 19**

На основе результатов предыдущей лабораторной работы разработать приложение Windows Forms, хранящее информацию о товаре цветочного магазина, которое будет принимать сорт цветов(string); цвет(string); дата завоза заполнение которой реализовано при помощи элемента управления monthCalendar (рис 14) и его свойсв SelectionEnd.Date; а также цена за шт.(int). После чего, по запросу, выводить эти значения в элемент TextBox, причем каждый с новой строки.

Рисунок 14: 

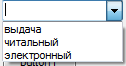
**Вариант 20**

На основе результатов предыдущей лабораторной работы разработать приложение Windows Forms, хранящее информацию о товарах магазина детских игрушек, которое будет принимать значения возраст ребенка которому подходит данная игрушка(string), заполнение которого реализовано при помощи элемента управления ComboBox(рис 15); тип игрушки(string), страна производитель(string). После чего, по запросу, выводить эти значения в элемент RichTextBox, причем каждый с новой строки.

Рисунок 15: 

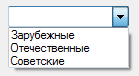
**Вариант 21\***

На основе результатов предыдущей лабораторной работы разработать приложение Windows Forms, реализующее базу библиотеки, внутренняя структура представляется из 4 классов(без наследования), 1-ый Книги выдаваемые на руки, 2-ой книги для читального зала, 3-ий книги в электронном формате, четвертый класс должен содержать перегруженный конструктор сохраняющий каждый новый элемент класса в отдельном списке(List<>), так же должны быть реализованы методы возвращающие элементы хранящиеся в списке(для этого в каждом классе должны быть прописаны свойства для переменных). Все классы принимают одинаковые значения: название книги(string), издание(string), год издания(int), к какому классу книг принадлежит(реализовано через ComboBox(рис. 16) ). Так же должен быть реализован метод удаления элемента из списка по индексу. Визуальное отображения списков на форме должно быть выполнено в элементе ListBox, причем для каждой категории книг он свой.

Рисунок 16:

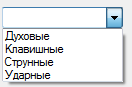
**Вариант 22\***

На основе результатов предыдущей лабораторной работы разработать приложение Windows Forms, реализующее базу фильмов, внутренняя структура представляется из 4 классов(без наследования), 1-ый Зарубежные фильмы, 2-ой Отечественные фильмы, 3-ий советские фильмы, четвертый класс должен содержать перегруженный конструктор сохраняющий каждый новый элемент класса в отдельном массиве(ArrayList), так же должны быть реализованы методы возвращающие элементы хранящиеся в массиве(для этого в каждом классе должны быть прописаны свойства для переменных). Все классы принимают одинаковые значения: название фильма(string), компания снявшая фильм(string), год выпуска в прокат(int), к какой категории фильмов принадлежит(реализовано через ComboBox(рис. 17)). Так же должен быть реализован метод изменения элемента из массива по индексу. Визуальное отображения списков на форме должно быть выполнено в элементе DataGridView, причем для каждой категории фильмов он свой.

Рисунок 17: 

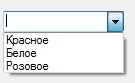
**Вариант 23\***

На основе результатов предыдущей лабораторной работы разработать приложение Windows Forms, реализующее базу музыкальных инструментов, внутренняя структура представляется из 5 классов(без наследования), 1-ый Духовые инструменты, 2-ой клавишные инструменты, 3-ий струнные инстменты,4-ый ударные инструменты, пятый класс должен содержать перегруженный конструктор сохраняющий каждый новый элемент класса в отдельном списке(List<>), так же должны быть реализованы методы возвращающие элементы хранящиеся в списке(для этого в каждом классе должны быть прописаны свойства для переменных). Все классы принимают одинаковые значения: фирма производитель(string), страна производитель(string), материал из которого изготовлен инструмент(string), к какому классу инструментов относится (реализовано через ComboBox(рис. 18) ). Так же должен быть реализован метод удаления элемента из списка по индексу. Визуальное отображения списков на форме должно быть выполнено в элементе ListBox, причем для каждой категории инструментов он свой.

Рисунок 18: 

**Вариант 24\***

На основе результатов предыдущей лабораторной работы разработать приложение Windows Forms, реализующее базу винной фабрики, внутренняя структура представляется из 4 классов(без наследования), 1-ый красное вино, 2-ой белое вино, 3-ий розовые вино, четвертый класс должен содержать перегруженный конструктор сохраняющий каждый новый элемент класса в отдельном массиве(List<>), так же должны быть реализованы методы возвращающие элементы хранящиеся в массиве(для этого в каждом классе должны быть прописаны свойства для переменных). Все классы принимают одинаковые значения: сорт винограда(string), количесво(int), год(int), к какому типу вина относиться(реализовано через ComboBox(рис. 19)). Так же должен быть реализован метод изменения элемента из массива по индексу. Визуальное отображения списков на форме должно быть выполнено в элементе ListBox, причем для каждой категории вина он свой.

Рисунок 19:

**Вариант 25\***

На основе результатов предыдущей лабораторной работы разработать приложение Windows Forms, реализующее базу завода по производству лампочек, внутренняя структура представляется из 4 классов(без наследования), 1-ый люминесцентные лампы , 2-ой лампа накаливания 3-ий энергосберегающие лампы, четвертый класс должен содержать перегруженный конструктор сохраняющий каждый новый элемент класса в отдельном массиве(ArrayList), так же должны быть реализованы методы возвращающие элементы хранящиеся в массиве(для этого в каждом классе должны быть прописаны свойства для переменных). Все классы принимают одинаковые значения: цветовая температура(int), срок службы(int), световая отдача(int), к какой категории ламп принадлежит(реализовано через ComboBox(рис. 20)). Так же должен быть реализован метод удаления элемента из массива по индексу. Визуальное отображения списков на форме должно быть выполнено в элементе DataGridView, причем для каждой категории фильмов он свой.

Рисунок 20: