

Лабораторная работа 1. Введение в разработку форм

Цель работы:

Изучение методов построения форм Windows и добавление элементов управления. Получение навыков по настройке форм, созданию прямоугольных и наследуемых (производных) форм.

Формы Windows — это основной компонент пользовательского интерфейса. Формы предоставляют контейнер, который содержит элементы управления, меню и позволяет отображать приложение в уже привычной и единообразной модели. Формы могут реагировать на события мыши и клавиатуры, поступающие от пользователя, и выводить на экран данные для пользователя с помощью элементов управления, которые содержатся в форме.

Формы Windows содержат множество свойств, позволяющих настраивать их внешний вид и поведение. Просматривать и изменять эти свойства можно в окне Properties при разработке, а также программно во время выполнения приложения.

Основные свойства формы:

- **Name:** устанавливает имя формы - точнее имя класса, который наследуется от класса Form. Это не заголовок формы, который вы видите при запуске формы, а название формы внутри проекта, которое вы будете использовать в коде.
- **AcceptButton:** устанавливается значение кнопки, которая будет срабатывать при нажатии клавиши Enter. Для того чтобы это свойство было активным, необходимо наличие по крайней мере одной кнопки, расположенной на форме.
- **CancelButton:** устанавливается значение кнопки, которая будет срабатывать при нажатии клавиши Esc. Для того чтобы это свойство было активным, необходимо наличие по крайней мере одной кнопки, расположенной на форме.
- **BackColor:** указывает на фоновый цвет формы. Щелкнув на это свойство, мы сможем выбрать тот цвет, который нам подходит из списка предложенных цветов или цветовой палитры
- **BackgroundImage:** указывает на фоновое изображение формы
- **BackgroundImageLayout:** определяет, как изображение, заданное в свойстве BackgroundImage, будет располагаться на форме.
- **ControlBox:** указывает, отображается ли меню формы. В данном случае под меню понимается меню самого верхнего уровня, где находятся иконка приложения, заголовок формы, а также кнопки минимизации формы и крестик. Если данное свойство имеет значение false, то мы не

увидим ни иконку, ни крестика, с помощью которого обычно закрывается форма

- **Cursor:** определяет тип курсора, который используется на форме
- **DrawGrid:** устанавливается наличие либо отсутствие сетки из точек, True которая помогает форматировать элементы управления. В любом случае сетка видна только на стадии создания приложения
- **Enabled:** если данное свойство имеет значение false, то она не сможет получать ввод от пользователя, то есть мы не сможем нажать на кнопки, ввести текст в текстовые поля и т.д.
- **Font:** задает шрифт для всей формы и всех помещенных на нее элементов управления. Однако, задав у элементов формы свой шрифт, мы можем тем самым переопределить его
- **ForeColor:** цвет шрифта на форме

FormBorderStyle: указывает, как будет отображаться граница формы и строка заголовка. Возможные варианты:

- None — форма без границ и строки заголовка;
- FixedSingle — тонкие границы без возможности изменения размера пользователем;
- Fixed3D — границы без возможности изменения размера с трехмерным эффектом;
- FixedDialog — границы без возможности изменения, без иконки приложения;
- Sizable — обычные границы: пользователь может изменять размер границ;
- FixedTool Window — фиксированные границы, имеется только кнопка закрытия формы. Такой вид имеют панели инструментов в приложениях;
- SizableToolWindow — границы с возможностью изменения размеров, имеется только кнопка закрытия формы

- **HelpButton:** указывает, отображается ли кнопка справки формы
- **Icon:** задает иконку формы. Поддерживаются форматы .ico
- **Location:** определяет положение по отношению к верхнему левому углу экрана, если для свойства StartPosition установлено значение Manual
- **MaximizeBox:** указывает, будет ли доступна кнопка максимизации окна в заголовке формы
- **MinimizeBox:** указывает, будет ли доступна кнопка минимизации окна
- **MaximumSize:** задает максимальный размер формы. Форма будет принимать указанный размер при нажатии на стандартную кнопку "Развернуть"

- **MinimumSize:** задает минимальный размер формы. Форма будет принимать указанный размер при изменении ее границ пользователем (если свойство `FormBorderStyle` имеет значение по умолчанию `Sizable`)
- **Opacity:** задает прозрачность формы
- **Size:** определяет начальный размер формы
- **StartPosition:** указывает на начальную позицию, с которой форма появляется на экране. Возможны следующие значения:
 - `Manual` — форма появляется в верхнем левом углу экрана;
 - `CenterScreen` — в центре экрана;
 - `WindowsDefaultLocation` — расположение формы по умолчанию.
 Если пользователь изменил размеры формы, то при последующем ее запуске она будет иметь тот же самый вид и расположение;
 - `WindowsDefaultBounds` — границы формы принимают фиксированный размер;
 - `CenterParent` — в центре родительской формы
- **Text:** определяет заголовок формы
- **TopMost:** если данное свойство имеет значение `true`, то форма всегда будет находиться поверх других окон
- **Visible:** видима ли форма, если мы хотим скрыть форму от пользователя, то можем задать данному свойству значение `false`
- **WindowState:** указывает, в каком состоянии форма будет находиться при запуске: в нормальном, максимизированном или минимизированном. Возможны следующие значения:
 - `Normal` — форма запускается с размерами, указанными в свойстве `Size`;
 - `Minimized` — форма запускается с минимальными размерами, указанными в свойстве `MinimumSize`;
 - `Maximized` — форма разворачивается на весь экран

Элементы управления - это компоненты, объединяющие графический интерфейс с предварительно разработанной функциональностью. Элементы управления представляют собой многократно используемые блоки кода, предназначенные для выполнения определенных задач. Все элементы управления являются производными базового класса **Control**, а значит, тоже используют различные свойства, задающие размер, расположение и другие основные аспекты элементов управления.

Способы добавления элемента управления на форму

Откройте форму.

1. На панели элементов щелкните элемент управления, который необходимо добавить в форму. Затем на форме щелкните место, где

должен располагаться левый верхний угол элемента управления, и перетащите его в то место, где должен располагаться правый нижний угол элемента управления. Элемент управления добавляется в форму с указанными расположением и размером.

2. Перетащите нужный элемент управления с панели элементов в форму. Так элемент управления добавляется в форму в указанном расположении с его размером по умолчанию.
3. Дважды щелкните элемент управления на **панели элементов** , чтобы добавить его в левый верхний угол формы в его размер по умолчанию.
4. Кроме того, можно динамически добавлять элементы управления в форму во время выполнения.

В методе, обрабатывающем событие кнопки `Click` в классе формы, вставьте код, аналогичный приведенному ниже, чтобы добавить ссылку на переменную элемента управления, задать элемент управления `Location` и добавить элемент управления.

```
private void button1_Click(object sender, System.EventArgs e)
{
    TextBox myText = new TextBox();
    myText.Location = new Point(25,25);
    this.Controls.Add (myText);
}
```

Упражнение 1. Настройка прямоугольной формы Windows

1. Откройте Visual Studio и создайте новый проект Windows Forms. Проект откроется с формой по умолчанию с именем **Form1** в конструкторе.
2. Выберите форму в конструкторе. Свойства формы отображаются в окне **Properties**.
3. Когда форма будет выбрана, найдите свойство **Text** в окне **Свойства**. В зависимости от того, как отсортирован список, может потребоваться прокрутить вниз. Выберите **Text**, введите **FirstForm**, а затем нажмите клавишу **ВВОД**. Теперь форма должна содержать текст «**FirstForm**» в заголовке окна.

Примечание

Свойства можно упорядочить по категориям или в алфавитном порядке. Вы можете переключаться между двумя этими представлениями с помощью кнопок в окне Свойства. В этом руководстве свойства легче находить в представлении, в котором свойства представлены в алфавитном порядке.

4. Вернитесь к конструктору **Windows Forms**. Нажмите нижний правый маркер перетаскивания формы, чтобы изменить размер формы.
5. Посмотрите в окно **Свойства** и обратите внимание, что изменилось значение свойства **Size**. Свойство **Size** меняется при каждом изменении формы. Введите значения **500, 300** непосредственно в свойстве **Size** и затем нажмите клавишу **ВВОД**.
6. Запустите приложение. Помните, что для запуска приложения можно использовать любой из описанных ниже методов.
 - Нажмите клавишу **F5**.
 - В строке меню выберите **Debug (Отладка)/ I Start Debugging (Запуск отладки)**.
 - На панели инструментов нажмите кнопку **Start**.
7. Перед переходом к следующему шагу остановите приложение, так как интегрированная среда разработки не позволяет изменять выполняющееся приложение.
8. Выберите форму и в окне **Properties** задайте свойствам значения, как указано ниже:

Свойство	Значение
FormBorderStyle	Fixed3D
StartPosition	Manual
Location	100; 200
Opacity	75%

Примечание

*Если окно **Свойства** не открывается, остановите приложение, нажав квадратную кнопку **Остановить отладку** на панели инструментов, или просто закройте окно. Если приложение остановлено, но окно **Свойства** все равно не отображается, в строке меню выберите **Вид > Окно свойств**.*

9. Перетащите три кнопки из **Toolbox** в форму и разместите их так, как вам будет удобно.

10. Поочередно выберите каждую кнопку и в окне **Properties** задайте свойству кнопок **Text** значения **Border Style**, **Resize** и **Opacity**.

11. Для кнопки **Border Style** задайте свойство **Anchor - Top, Left**. Для кнопки **Opacity** задайте свойство **Anchor - Bottom, Right**.

Реализация обработчиков событий

12. В конструкторе дважды щелкните кнопку **Border Style**, чтобы открыть окно с кодом обработчика события **Button1 Click**. Добавьте в этот метод следующую строку кода:

```
this.FormBorderStyle = FormBorderStyle.Sizable;
```

13. Возвратитесь в окно конструктора, дважды щелкните кнопку **Resize** и добавьте следующую строку:

```
this.Size = new Size(300, 500);
```

14. Возвратитесь в окно конструктора, дважды щелкните кнопку **Opacity** и добавьте следующую строку:

```
this.Opacity = 1;
```

15. Запустите приложение еще раз.

16. Щелкайте каждую кнопку и наблюдайте, как изменяется вид формы. Измените поочередно расположение границ формы и сравните поведение кнопок внутри формы. Обратите внимание, что расстояние до левой и верхней границ от кнопки **Border Style** остается постоянным. Почему?

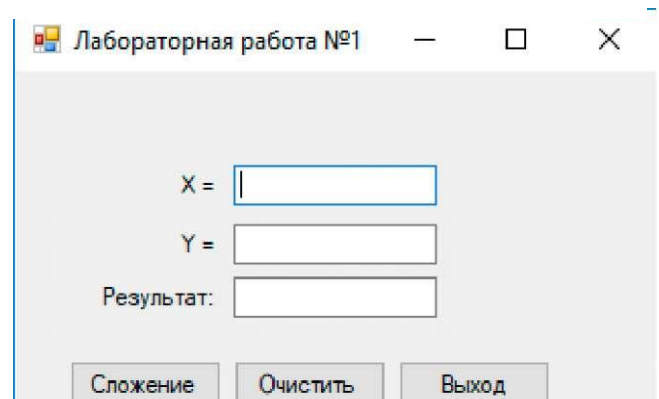
Упражнение 2. Разработка приложения с GUI-интерфейсом.

Задание: разработать программу, которая производит арифметические действия с двумя введенными числами. По нажатию кнопки «очистить» поля для ввода и результат удаляются.

1. Создайте новый проект Windows Forms.

2. Задайте заголовок окна «Лабораторная работа 1»

3. Разместите на форме три компонента **Label**, три компонента **TextBox** и три компонента **Button**. Поменяйте для них свойства **Text** и положение так, чтобы получился внешний вид формы, аналогичный примеру.



4. В зависимости от варианта, выданного преподавателем, ваша программа будет выполнять сложение / вычитание / умножение / деление / возведение в степень. Соответствующее название дайте первой кнопке.

5. С кнопкой *Button1* свяжите обработчик события *Click*. В теле обработчика события объявите три переменные. Две переменные инициализируются числами, введенными в первые два текстовых поля, а в третьей хранится результат арифметического действия и выводится в третье текстовое поле.

Пример преобразования текста из TextBox1 в вещественное число:

```
x = double.Parse(TextBox1.Text);
```

Пример вывода значения переменной в TextBox1:

```
TextBox1.Text = x.ToString();
```

6. По нажатии кнопки «очистить» обработчик события должен всем трем текстовым полям присваивать пустые строки.

7. По нажатии кнопки «выход» обработчик события должен закрывать форму. Для этого существует функция `Close()`.

8. Протестируйте свое приложение.

9. Теперь добавьте в код решения задачи проверку введенных данных. В случаях сложения и умножения числа не должны быть слишком большими. При вычитании – первое число должно быть больше второго, при делении – делитель не должен быть равен 0. При возведении в степень, степень должна быть целым числом. В случае неверных исходных данных выдается сообщение об ошибке и поля очищаются.

Для вывода сообщений можно использовать метод `Show` класса `MessageBox` из пространства имен `System.Windows.Forms`.

```
MessageBox.Show("This is a test", "Title", MessageBoxButtons.OK);
```

10. Протестируйте приложение.

Лабораторная работа 2. Работа с элементами управления

Цель работы:

Изучение способов использования элементов управления и получение навыков по обработке событий.

Кнопка окна свойств `Events` (События) переключает окно `Properties` в режим управления обработчиками различных событий (например, мыши, клавиатуры) и одновременно выводит список всех событий компонента.

Двойной щелчок мыши в поле значения события генерирует обработчик для него и переключает в режим кода.

Некоторые события контролов:

Click	Происходит при щелчке элемента управления.
DoubleClick	Происходит при двойном щелчке элемента управления.
DragDrop	Вызывается при завершении операции перетаскивания.
DragEnter	Происходит, когда объект перетаскивается в границы элемента управления.
DragLeave	Вызывается, когда объект перетаскивается за пределы элемента управления.
DragOver	Происходит, когда объект перетаскивается через границу элемента управления.
GotFocus	Вызывается при получении фокуса элементом управления.
KeyDown	Происходит при нажатии клавиши, если элемент управления имеет фокус.
KeyPress	Происходит при нажатии клавиши с буквой, пробела или клавиши BACKSPACE, если фокус находится в элементе управления.
KeyUp	Происходит, когда отпускается клавиша, если элемент управления имеет фокус.
Leave	Происходит, когда фокус ввода покидает элемент управления.
LostFocus	Происходит при потере фокуса элементом управления.
MouseClick	Вызывается при щелчке мышью элемента управления.
MouseDoubleClick	Вызывается при двойном щелчке мышью элемента управления.
MouseDown	Происходит при нажатии кнопки мыши, если указатель мыши находится на элементе управления.
MouseEnter	Происходит, когда указатель мыши оказывается на элементе управления.
MouseHover	Происходит, когда указатель мыши задерживается на элементе управления.
MouseLeave	Происходит, когда указатель мыши покидает элемент управления.
MouseMove	Происходит при перемещении указателя мыши по элементу управления.

MouseUp	Происходит при отпускании кнопки мыши, когда указатель мыши находится на элементе управления.
MouseWheel	Происходит при прокручивании колеса мыши, если данный элемент управления находится в фокусе.
Move	Происходит при перемещении элемента управления.
TextChanged	Происходит при изменении значения свойства Text .

Упражнение 1. Обработка событий Click и MouseMove

Создайте простое приложение, отслеживающее события мыши, которые происходят у конкретного элемента управления.

Размещение на форме элементов управления

1. Создайте новое Windows приложение. Назовите его WinQuestion.
2. Расположите на форме две кнопки **Button** и надпись **Label**, разместите их по-своему усмотрению.
3. Установите следующие свойства элементов управления и формы:

Объект	Свойство	Значение
Form1	FormB orderStyle	Fixed3D
	Size	350; 200
	Text	Насущный вопрос
label1	Text	Вы довольны своей зарплатой?
Button1	Name	btnyes
	Text	Да
Button2	Name	btnno
	Text	Нет

4. Щелкните дважды по кнопке "Да". В обработчике этой кнопки добавьте следующий код:

```
MessageBox.Show("Мы и не сомневались, что Вы так думаете!");
```

5. Выделите кнопку "Нет". В окне **Properties** переключитесь в окно событий и дважды щелкните в поле MouseMove.

6. В обработчике этого события добавьте код для связывания движения мыши с координатами кнопки и указания координат, куда кнопка будет перемещаться:

```
btnno.Top -= e.Y; btnno.Left += e.X;
if (btnno.Top < -10 || btnno.Top > 100) btnno.Top = 60;
if (btnno.Left < -80 || btnno.Left > 250) btnno.Left = 120;
```

7. Запустите приложение и нажмите на каждую из кнопок.
8. Что означает "e" в коде, приведенном выше?

Задание. Написать программу с визуальным интерфейсом, согласно варианта. Для генерирования случайного цвета можно использовать следующий код:

```
Color.FromArgb(rnd.Next(256), rnd.Next(256), rnd.Next(256));
```

1. Добавьте на экран кнопки (красный, синий, зеленый, желтый, белый), по нажатию на которые будет меняться фоновый цвет экрана.
2. Программа загадывает число от 1 до 3, а пользователь пытается его угадать, нажимая на нужную кнопку с цифрой.
3. Добавьте на экран три кнопки и текстовое поле. По нажатию на каждую из кнопок текст, написанный на кнопке, попадает в текстовое поле. Добавьте кнопку для очистки текстового поля.
4. Добавьте на экран текстовое поле и две кнопки. Пусть в текстовом поле отображается количество нажатий на первую кнопку, а вторая сбрасывает подсчет.
5. Добавьте текстовое поле и две кнопки «Next» и «Back», по нажатию на которые, текстовое поле циклично выводит сообщения.
6. Добавьте на экран две кнопки. Нажатие на первую кнопку меняет цвет второй кнопки на случайный, нажатие на вторую, делает тоже самое с первой кнопкой.
7. Добавьте на экран три кнопки. Нажатие на одну из них меняет тексты на двух других кнопок между собой.
8. Добавьте на экран кнопки (красный, синий, зеленый) и текстовое поле, по нажатию на кнопки будет меняться цвет текста.
9. Программа загадывает цвет, а пользователь пытается его угадать, нажимая на кнопку нужного цвета. Достаточно четырех цветов.
10. Сделайте «клавиатуру». Добавьте текстовое поле и несколько кнопок с буквами и пробелом. В текстовом поле будет отображаться набранный текст.

Лабораторная работа 3. Создание приложения с несколькими формами.

Цель работы:

Изучение способов создания нескольких форм, передачи параметров и переход между ними.

Как правило, полноценное windows-приложение должно содержать несколько форм (главную форму, формы для ввода данных, формы для различных дополнительных действий (возможностей) и т.д.). Чтобы создать дополнительную форму необходимо проделать следующие действия:

1. Выбрать закладку projects, на имени проекта вызвать контекстное меню (правой клавишей мыши) и выбрать пункт меню Add->New Item.
2. В открывшемся диалоге выбрать добавление формы и ввести ее имя.
3. После этого в проекте (projects) появляется новая форма
4. Для того, чтобы переключиться на новую форму - нужно выбрать ее на закладке projects.

Новая форма создается следующим образом:

```
<ИмяКлассаФормы> <ИмяФормы>= new <ИмяКлассаФормы>();
```

Формы обладают многими методами, наиболее нужные - Show(), ShowDialog() и Close(). Методы Show(), ShowDialog() служат для отображения формы на экране (разница между методами заключается в том, что метод ShowDialog() блокирует переход на основную форму, а метод Show() - нет). Метод Close() закрывает текущую форму. Если этот метод вызвать в классе основной формы - закроется всё приложение.

Для того, чтобы отобразить новую форму на экране - нужно ее создать и вызвать метод Show() или ShowDialog().

Формы могут обмениваться между собой информацией, т.е. можно, например, на одной форме разместить модуль расчета, а на второй - модуль ввода данных. Для того, чтобы передать введенные данные с формы ввода данных, существует несколько вариантов

1. Создать в классе формы расчета несколько статических переменных, которые нужно будет проинициализировать в форме ввода. Данный метод передачи данных не самый правильный с точки зрения ООП, но самый простой.
2. Создать отдельный класс, лучше статический, в основном namespace, т.е. например, в файле Program.cs. Его открытые свойства/методы доступны из любой формы.
3. Передать данные в конструктор второй формы.
4. Создавать методы-обертки для скрытых полей.
5. Использовать свойство 'родитель':

При создании второй формы устанавливаем владельца

```
Form2 f = new Form2();  
f.Owner = this;  
f.ShowDialog();
```

Во второй форме определяем владельца

```
Form1 main = this.Owner as Form1;  
if (main != null)  
{  
    string s = main.textBox1.Text;  
    main.textBox1.Text = "OK";  
}
```

Задание. В приложении, созданном в первой лабораторной работе, добавьте функционал, чтобы по нажатию на кнопку расчета открывалось новое окно, в котором бы отображался результат вычислений. В этом окне создайте кнопку «Вернуться», по нажатию на которую диалоговое окно закрывается.