МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Озерский технологический институт — филиал НИЯУ МИФИ

Кафедра прикладной математики

Вл. Пономарев

ПРАКТИКУМ

по Microsoft .NET и языку программирования C#

Учебно-методическое пособие

Часть 5. Окна и элементы управления

2018 г.

УДК 681.3.06
П 56

Вл. Пономарев. Практикум по Microsoft .NET и языку программирования C#. Учебно-методическое пособие. Часть 5. Окна и элементы управления. Озерск: ОТИ НИЯУ МИФИ, 2018. — 14 с.

В пособии предлагаются практические работы по изучению платформы Microsoft .NET и языка программирования C#.

В этой части рассматриваются окна и элементы управления.

В качестве основного материала при выполнении практических работ пособие предназначено для студентов, обучающихся по направлению подготовки 09.03.01 «Информатика и вычислительная техника» и специальности 09.05.01 «Применение и эксплуатация автоматизированных систем специального назначения».

Рецензенты:
1. Синяков В. Е., начальник УИТ ФГУП «ПО «Маяк».
2. Зубаиров А. Ф., ст. преподаватель кафедры ПМ ОТИ НИЯУ МИФИ.

УТВЕРЖДЕНО
Редакционно-издательским
Советом ОТИ НИЯУ МИФИ

Содержание

Общие цели занятий

В ходе практических работ изучаются основы использования классов в рамках методологии объектно-ориентированного программирования.

В этой части работ рассматриваются следующие темы:

- оконные приложения C#;
- пользовательские элементы управления;
- приложения WPF.

К практическим работам приписаны контрольные вопросы и упражнения. Контрольные вопросы могут быть заданы преподавателем в ходе защиты работы, однако преподаватель может задавать и другие вопросы, не указанные в списке.

Для защиты знаний и навыков, полученных в ходе выполнения практических работ студент должен иметь тетрадь (12-18 листов). Для каждой выполненной работы в тетрадь записывается отчет.

Отчет по работе начинается с заголовка:

1. Фамилия Имя Отчество
2. Группа
3. Дата начала выполнения работы
4. Код работы
5. Название работы
6. Цели работы
7. Задачи работы

При необходимости при выполнении работы в отчет записываются контрольные значения. Во время защиты тетрадь используется для вопросов преподавателя и ответов студента.

1. Работа CS-301. Введение в формы и окна

Цели:
- введение в оконные приложения C#.

Задачи:
- базовые сведения об окнах Windows;
- свойства окон в C#;
- события в окнах C#;
- взаимодействие с элементами управления.

1.1. Приложение Windowing

Создаем проект приложения C# типа Windows Forms Application.

Название проекта Windowing.

После создания увидим форму. Термином «форма» будем обозначать окно приложения в процессе его конструирования.

Конструирование окон в C# значительно отличается от того, как это происходит при программировании на C++. Конструктор окна визуальный, все элементы управления, которые можно поместить в окно, находятся на специальной панели Toolbox, их можно просто перетаскивать на форму. На форме элементы можно перетаскивать, выравнивать их, задавать равное расстояние между ними.

В C++ нет разницы между окном приложения и кнопкой. И то и другое создается одной функцией CreateWindow, и на самом деле кнопка является окном.

В C# ни окна, ни кнопки (равно как и другие элементы) не создаются программистом при помощи функций. Все элементы уже как бы созданы. Хотите еще одно окно приложения — добавьте еще одну форму. Хотите кнопку — возьмите ее на панели Toolbox и перетащите на форму. И хотя создание окон Windows происходит при помощи функции CreateWindow, детали создания скрыты в иерархии базовых классов Microsoft .NET.

Будем различать окна и элементы управления. Элементами управления являются кнопки, флажки, поля для ввода и многое другое, что является производным от класса Control.

Если стандартных элементов управления не хватает, всегда можно создать собственный элемент управления, называемый в этом случае user control. Элементы управления называются так потому, что с их помощью пользователь управляет приложением.

Прежде, чем мы приступим к исследованию элементов управления, изучим сначала собственно окна.

Особенностью программирования оконных приложений является их объектная модель. Любые характеристики элемента задаются при помощи свойств, либо программно, либо при помощи окна свойств Properties.

1.2. Структура окна

Окно Windows состоит из двух областей. Первая часть называется неклиентской или системной областью. Этой частью окна управляет операционная система. В неклиентскую область входит заголовок окна и рамка. Сюда же относится системное оконное меню, которое вызывается клавишами Alt+пробел. Системное меню предоставляет базовые функции управления окнами, избавляя программиста от необходимости заботиться об этом. В функции системного меню входит изменение размеров окна, его положения, и функция, которая закрывает окно. Небольшие кнопочки, расположенные в заголовке окна, — это быстрый доступ к некоторым системным функциям. Часть системных функций выполняет рамка окна, позволяя изменять размер окна перетаскиванием границ окна мышью.

Все, что не входит в системную область окна, является клиентской областью. На самом деле это прямоугольник, в пределах которого только и можно размещать в окне элементы управления.

1.3. Свойства окна

В C# с каждым окном и с каждым элементом окна связан идентификатор, обозначаемый свойством (Name). Здесь скобки являются частью названия свойства при отображении свойства в окне свойств.

Щелкнем на форму правой кнопкой мыши и в контекстном меню выберем Properties (свойства). В результате откроется, если не открыто, окно свойств. В этом окне найдите свойство (Name). Скорее всего, оно имеет значение Form1, так же, как и свойство Text, которое показывает текст, отображаемый в заголовке окна.

Как работать с окном свойств.

Во-первых, окно Properties отображает не только свойства, но связанные с событиями методы формы. Поэтому нужно быть уверенным в том, что окно сейчас показывает свойства, а не события. Во-вторых, элементы списка могут быть упорядочены либо по алфавиту, либо по группам одного назначения. Сейчас выберем порядок по алфавиту (вторая кнопка) и отображение свойств (третья кнопка):

Давайте изменим название объекта окна на frmMain.

Для этого щелкнем левой кнопкой мыши на название свойства (Name) и сразу (не делая никаких других движений ручками и мышками) набираем на клавиатуре текст frmMain, ввод которого завершаем нажатием Enter.

Если вы предполагаете изменить сразу после этого значение другого свойства, то Enter можно не нажимать, а щелкнуть на название другого свойства и ввести другое значение.

Нажимая Enter, вы сообщаете системе, что ввод свойства завершен, и система произведет необходимые изменения в тексте модуля, например, Program.cs. Откроем этот модуль и убедимся, что текст frmMain в нем встречается.

Теперь изменим свойство Text, заменив Form1 на Windowing.

Запустим программу, полюбуемся на созданное нами окно, и закроем его, используя кнопку системного меню «Закрыть».

Важнейшим свойством окна является FormBorderStyle. Это свойство определяет, каким будет окно. Сейчас свойство имеет значение Sizable, что означает, что можно изменять размер окна. В этом несложно убедиться, запустив программу, и, зацепив рамку окна мышью, изменить положение мыши.

Выберем другое значение — FixedDialog. Этот стиль предназначен для диалоговых окон, которые отличаются тем, что изменить размер окна мышью не получится. Однако в заголовке окна остались кнопочки «Свернуть» и «Развернуть», которые изменяют размер окна. Запустим программу, введем Alt+пробел, увидим, что окно можно развернуть. Закроем окно.

Наличием кнопочек и, соответственно, функций системного меню управляют свойства MaximizeBox, MinimizeBox, ControlBox, HelpButton.

Если ControlBox равно False, то никакие функции системного меню не будут доступны. Установите это значение, запустите программу, и убедитесь, что закрыть окно нельзя. Возможно, что остановить программу удастся кнопкой «Stop» на панели инструментов среды. Если нет, то тогда поможет диспетчер задач.

Вернем значение True свойству ControlBox.

Установим значение False свойствам MaximizeBox и MinimizeBox, а свойство HelpButton установим в True. Запустим программу. Закроем окно.

Попробуем другое значение свойства FormBorderStyle, Fixed3D, обозначающее 3D стиль. Запустим программу. Закроем окно.

Попробуем другое значение свойства FormBorderStyle, FixedSingle. Оно означает одинарную рамку окна. Запустим программу. Закроем окно.

Выберем значение FormBorderStyle, равное None. Получим окно, которое состоит только из окна. Если запустим программу, остановить ее будет проблемой, как и в случае с выключением системного меню. Такие окна нужны для создания собственных окон типа заставок или фона, самостоятельно исчезающих сообщений, окон произвольной конфигурации.

Есть два значения, в названии которых присутствует слово Toolbox. Эти стили формируют окна панелей инструментов.

Вернем значение FormBorderStyle, равное Sizable.

Установим свойство IsMdiContainer в True. Запустим программу.

Мы получили главное окно многодокументного интерфейса, когда окна документов располагаются внутри этого окна. Закроем окно. Окон документов у нас нет. Вернем значение False свойства IsMdiContainer.

Возможно, вы обратили внимание, что при запуске программы окно каждый раз появляется в разных местах экрана. Свойство StartPosition управляет начальным положением окна. Выберем значение CenterScreen. Запустим программу, убедимся, что окно в центре экрана, закроем окно.

Выберем теперь значение Manual. Установим свойство Location в значение 100; 100. Запустим программу, убедимся, что окно находится в позиции (100, 100), закроем окно.

Значение CenterParent означает центрировать окно относительно своего родителя. Родительского окна у нас нет, это значение сейчас нельзя протестировать. Вернем значение CenterScreen.

Начальным размером окна управляет свойство WindowState. Оно может быть обычным (по размеру, заданному в конструкторе), развернутым во весь экран, или свернутым в кнопку на панели задач.

Свойство Opacity задает непрозрачность. Зададим значение 50, запустим программу, закроем окно. Вернем значение 100.

Установим значение FormBorderStyle, равное FixedDialog. Значение свойства HelpButton установим в False. У окна должна быть только одна системная кнопочка «Закрыть».

1.4. Кнопки

Разместим на форме кнопку. Для этого откройте панель Toolbox, найдите в ней элемент Button и дважды щелкните на него. На форме появится кнопка. Обратим внимание на шрифт надписи на кнопке. Вероятно, это не тот шрифт, который мы хотели бы. Элемент, помещаемый на форму, изначально всегда имеет шрифт формы. Удалим кнопку Delete.

Установим шрифт формы Tahoma, 9pt, используя свойство Font.

Снова разместим на форме кнопку. Переместите ее в какое-нибудь подходящее положение.

Запомним, что первым действием с любым элементом управления, размещенным на форме, является задание программного имени, то есть свойства (Name). Зададим это свойство для кнопки равным btnClose. Затем изменим свойство Text на Закрыть.

Дважды щелкнем на кнопку.

Откроется окно кода формы, в котором появится обработчик события Click, функция с названием btnClose_Click. Событие Click является для кнопки событием по умолчанию, поэтому двойной щелчок вызывает вход в обработчик именного этого события.

Кнопка предназначена для остановки работы программы. Есть разные способы сделать это. Самое простое — закрыть окно методом Close.

Пишем метод Close в обработчике.

Запускаем программу, нажимаем кнопку.

Иногда удобно закрывать окно, просто нажав клавишу Escape. Есть несколько способов сделать это.

Если на форме есть кнопка, которая закрывает окно, то проще всего сказать форме, что эта кнопка нажимается клавишей Escape. Для этой цели у формы есть свойство CancelButton, которое нужно установить в значение btnClose (выбрать его).

Установим, запускаем программу, нажимаем Escape.

Кнопка также может нажиматься клавишей Enter, при этом кнопка называется кнопкой по умолчанию. Кнопка по умолчанию получает дополнительную рамку. Управляет этим свойство формы AcceptButton. Установим это свойство в значение btnClose. Запустим программу, нажмем Enter. Вернем значение none свойства AcceptButton.

У кнопок, как и у других элементов управления, есть также множество других событий.

Выберем кнопку на форме. Нажмем в окне Properties кнопку Events. Увидим перечень всех возможных событий. Их много, это не значит, что все они нужны. Как правило, для кнопки требуется только событие Click, означающее, что кнопка нажата и отпущена.

Ради интереса добавим событие опускания мыши на кнопку. Выбираем в списке событие MouseDown и вписываем название функции обработчика btnClose_MouseDown. Нажимаем Enter и попадаем в обработчик.

Вписываем в обработчик вызов сообщения:

```
private void btnClose_MouseDown(object sender, . . . ) {
    MessageBox.Show(this, "MouseDown", "btnClose",
        MessageBoxButtons.OK, MessageBoxIcon.Information);
}
```

Запускаем программу, щелкаем на кнопку, видим сообщение.

Закроем сообщение.

Сотрем значение события MouseDown кнопки. Событие возникать будет, но оно уйдет в никуда. Функция события останется невостребованной.

1.5. Вторичные окна

Добавим в проект еще одну форму.

Выбираем в меню Add Windows Form…, далее раздел Windows Forms, форму Window Form, название модуля формы frmShow.cs. Эта форма послужит сначала окном, образующим задний фон программы.

Зададим форме свойство BackColor, цвет формы любой. Другие свойства: WindowState = Maximized, FormBorderStyle = None.

Модуль Form1.cs.

Описываем переменную showForm, создаем окно frmShow.

```
public partial class frmMain : Form {
    private frmShow showForm = new frmShow();
    . . .
}
```

Дважды щелкнем в свободную поверхность формы frmMain.

Появится обработчик события загрузки формы в память.

Записываем в нем вывод формы на экран и относительное положение окон форм друг относительно друга:

```
showForm.Show();
showForm.BringToFront();
BringToFront();
```

Перейдем в окно Properties, выберем события формы frmMain, найдем событие FormClosing, вписываем название обработчика frmMain_Closing, нажимаем Enter. Вписываем код, закрывающий окно формы frmShow, то есть вызов метода Close этой формы.

Запустим программу.

Окно программы должно быть впереди окна формы frmShow.

Закроем окно программы Escape.

1.5.1. Исчезающее сообщение

Теперь будем использовать форму frmShow в ином качестве, как сообщение, которое само закрывается по истечении некоторого времени.

Код обработчиков формы frmMain удалим (закомментируем). Установим свойства frmShow: WindowState = Normal, StartPosition = CenterOwner, выберем шрифт Tahoma, 28pt, Size = 300; 100.

Модуль frmShow.cs.

В классе формы frmShow опишем переменную msg типа string. Затем опишем метод для установки значения переменной, метод void, один параметр text типа string, название метода SetMessage.

Выберем форму frmShow. В панели Toolbox найдем элемент Timer и дважды щелкнем на него. Таймер появится в нижней части окна формы. Установим свойства: (Name) = tmrClose, Enabled = False, Interval = 1200.

Дважды щелкнем на таймер, чтобы войти в его обработчик.

В обработчике устанавливаем свойство таймера Enabled в значение false, закрываем окно методом Close.

Откроем форму frmShow. Выберем в окне Properties события формы.

В событие Paint вписываем название обработчика frmShow_Paint, нажимаем Enter. В код обработчика вписываем вывод сообщения при помощи графических методов и запуск таймера:

```
TextFormatFlags flags = TextFormatFlags.HorizontalCenter
    | TextFormatFlags.VerticalCenter;
TextRenderer.DrawText(e.Graphics, msg, Font,
    new Rectangle(0, 0, Width, Height),
    SystemColors.ControlText, flags);
tmrClose.Enabled = true;
```

Более простым решением было бы размещение на форме элемента типа Label и использование его свойства Text. Но это просто.

Модуль кода формы frmMain Form1.cs.

Удалим код создания формы frmShow, оставив объявление переменной showForm в начале класса формы frmMain:

```
public partial class frmMain : Form {
    private frmShow showForm;
```

Добавим на форму еще одну кнопку. Программное имя btnShow, текст кнопки «Показать». Дважды щелкнем на кнопку.

Описываем действия кнопки:

```
showForm = new frmShow();
showForm.SetMessage("Message");
showForm.ShowDialog(this);
```

Запускаем программу, нажимаем кнопку «Показать», наблюдаем модальный диалог, который закрывается примерно через секунду. Закроем основное окно.

Чтобы показать немодальный диалог, следует использовать метод Show вместо ShowDialog. При этом наблюдается проблема с владельцем диалога. Окно диалога не устанавливается в центре окна владельца. В этом случае нужно вычислять положение окна владельца и задавать положение окна диалога. Сначала установим свойство StartPosition в значение Manual для формы frmShow. В обработчике кнопки frnShow пишем другой код, показывающий немодальный диалог:

```
showForm = new frmShow();
showForm.Left = Left + (Width - showForm.Width) / 2;
showForm.Top = Top + (Height - showForm.Height) / 2;
showForm.SetMessage("Message");
showForm.Show();
```

Запускаем программу, нажимаем кнопку «Показать». Чтобы убедиться в том, что диалог немодальный, достаточно щелкнуть в основное окно, при этом оно выйдет на передний план. Закроем основное окно.

Теперь сделаем так, чтобы окно постепенно исчезало, используя таймер и свойство Opacity. Код формы frmShow, обработчик таймера:

```
if (Opacity < 0.01) {
    tmrClose.Enabled = false;
    Close();
}
Opacity -= 0.02;
if (tmrClose.Interval > 1000) {
    tmrClose.Interval = 10;
}
```

Запускаем программу, нажимаем кнопку «Показать», наблюдаем, как окно постепенно исчезает.

Закрываем основное окно после того, как оно получит фокус.

1.5.2. Перемещение окна

Внесем в форму frmShow еще пару изменений.

Во-первых, дадим возможность закрыть окно клавишей Escape.

Откроем окно frmShow.

В окне Properties выберем события.

В событие KeyPress впишем название обработчика frmShow_KeyPress, нажмем Enter. Вписываем в обработчик код, закрывающий окно:

```
if (e.KeyChar == 27) {
    tmrClose.Enabled = false;
    Close();
}
```

У формы frmShow нужно установить свойство KeyPreview в значение True, это нужно для того, чтобы форма сначала проверяла ввод с клавиатуры, а затем направляла этот ввод своим элементам.

Запустим программу, нажмем кнопку «Показать», когда появится окно сообщений, нажмем Escape. Закроем основное окно.

Теперь добавим возможность перемещать окно мышью.

Есть много разных способов. Воспользуемся стандартным сообщением Windows WM_SYSCOMMAND с параметром SC_MOVE. Для этого необходимо подключиться к модулю операционной системы при помощи атрибутов.

Добавим инструкцию using System.Runtime.InteropServices.

Код формы frmShow:

```
public partial class frmShow : Form {
    [DllImport("user32", CharSet = CharSet.Auto)]
    internal extern static bool PostMessage(
        IntPtr hWnd, uint Msg, uint WParam, uint LParam);
    const uint WM_SYSCOMMAND = 0x0112;
    const uint SC_MOVE = 0xF012;
```

Теперь мы можем вызывать функции операционной системы. При щелчке в форму frmShow нужно отпустить захват событий мыши и послать команду перемещения окна.

Откроем форму frmShow.

В окне Properties выберем события, найдем событие MouseDown, впишем название обработчика frmShow_MouseDown, нажмем Enter.

В обработчике описываем следующие действия:

```
if (e.Button == MouseButtons.Left) {
    Capture = false;
    PostMessage(Handle, WM_SYSCOMMAND, SC_MOVE, 0);
}
```

Запускаем программу, нажимаем кнопку «Показать», хватаем окно сообщения мышью и таскаем по экрану. После того, как окно исчезнет, закроем основное окно.

1.5.3. Окно произвольной формы

Сначала нужно иметь рисунок окна.

Откроем MSPaint (графический редактор).

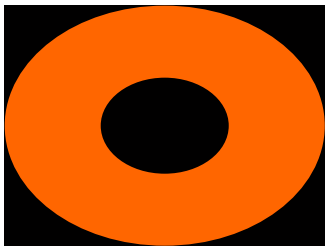
Установим размеры рисунка 400 пикселей ширины, 300 пикселей высоты (клавиши Ctrl+E).

Создадим цвет, равный 0 синего, 0 зеленого, 1 красного. Это практически черный цвет, но не черный. Закрасим этим цветом весь рисунок.

Этот цвет будет прозрачным в конечном итоге.

Нарисуем при помощи кисти или любым другим способом, например, бублик красного, синего или иного цвета.

Нужно следить за тем, чтобы у рисунка не было черных границ, всю площадь рисунка нужно использовать, чтобы видимое окно имело размер рисунка. Пример (сильно уменьшено):



Сохраним рисунок в каталоге C:\Windowing, название файла bublik, тип файла — 24-битный рисунок BMP.

Откроем форму frmShow. Установим свойство Size = 400; 300.

Выберем свойство BackgroundImage.

В появившемся диалоге Select Resource нажмем кнопку Import, найдем и выберем рисунок bublik, нажмем кнопку OK.

Установим свойство TransparencyKey = 1;0;0.

Запустим программу, нажмем кнопку «Показать».

Как видим, при изменении свойства Opacity наш черный цвет проявляется. В этом можно также убедиться, если установить начальное значение свойства Opacity равным 50%. Поэтому либо изменяющаяся прозрачность, либо окно произвольной формы.

Закомментируем код, запускающий таймер.

1.5.4. Результат диалога

Рассмотрим, как получить результат диалога.

Форма frmShow.

Установим свойство StartPosition = CenterParent.

Разместим на форме:

- кнопку отмены btnCancel с надписью Отмена;
- кнопку подтверждения btnOk с надписью Ok;
- поле для ввода текста, название txtText.

Для кнопки Отмена установим свойство DialogResult = Cancel.

Для кнопки Ok установим DialogResult = OK.

Выберем обе кнопки и поле для ввода.

Установим им шрифт Tahoma, 9pt.

Код кнопки Отмена закрывает окно. Код кнопки Ok записывает в переменную msg значение свойства Text поля txtText.

В коде формы опишем метод для получения значения переменной msg, метод возвращает string, параметров нет, название GetMessage.

Форма frmMain.

Разместим на форме элемент Label для вывода принятого из диалога текста. Название элемента lblText.

Перепишем обработчик кнопки btnShow.

Здесь мы вызываем диалог, анализируем результат:

```
showForm = new frmShow();
if (showForm.ShowDialog(this) == DialogResult.OK) {
    lblText.Text = showForm.GetMessage();
} else {
    lblText.Text = "Отменено пользователем";
}
```

Запустим программу, нажмем кнопку «Показать».

Введем в поле строку символов, нажмем кнопку Ok.

Убедимся, что введенная строка появляется в основном окне.

1.5.5. Управление переходом фокуса

Откроем форму frmShow. Выберем поле.

Установим свойство TabIndex = 0.

Для кнопки Ok установим TabIndex = 1.

Для кнопки Отмена установим TabIndex = 2.

Свойство TabIndex устнавливает порядок, в котором перемещается фокус при нажатии клавиши Tab. Этот порядок следует задавать сверху вниз и справа налево.

Свойство TabStop указывает, получает элемент фокус или нет.

1.6. Контрольные вопросы и упражнения

1. Опишите свойство формы FormBorderStyle.
2. Опишите свойство формы StartPosition.
3. Опишите свойство формы WindowState.
4. Опишите свойство формы ControlBox.
5. Опишите свойство формы KeyPreview.
6. Опишите свойства формы TabIndex и TabStop.
7. Поясните разницу между методами форм Show и ShowDialog.