ПРАКТИЧЕСКАЯ РАБОТА № 15

(4 академических часа)

Тема: Создание Windows-приложений. Создание графического калькулятора.

Цель задания:

1. Понять основы разработки графического интерфейса в приложениях Windows Forms.
2. Изучить стандартные элементы управления, обеспечивающие требуемый графический интерфейс пользователя.
3. Научиться создавать графический калькулятор на языке C# с помощью Windows Forms.

Теоретическая часть.

Процесс создания Windows-приложения состоит из двух основных этапов:

- визуальное проектирование, то есть задание внешнего облика приложения.
- определение поведения приложения.

Основным окном разрабатываемого приложения является форма. Визуальное проектирование заключается в помещении на форму элементов управления (компонентов) и задании их свойств (размер, положение на экране, цвет и пр.) и свойств самой формы.

При создании приложений Windows Forms в качестве контейнера элементов управления, составляющих графический интерфейс пользователя, используется класс, произведенный из класса Form. Дочерние объекты формы в большинстве своем являются наследниками класса Control. Класс Control является базовым классом как для подавляющего числа элементов управления, так и для самой формы.

Все элементы, которыми визуальная среда проектирования позволяет манипулировать, размещая их на форме, называются *компонентами*. Те компоненты, которые остаются видимыми и во время выполнения программы, называются *визуальными*, и это, как правило, элементы управления пользовательского интерфейса. Те компоненты, которые не видимы во время выполнения - называются *невизуальными,* или просто - компонентами.

Свойства

Каждый элемент управления обладает набором свойств. Свойства – это атрибуты (основные характеристики), которые описывают особенности объекта; например, отображают такие характеристики, как цвет, высота, ширина и положение объекта. На внешний вид объекта можно воздействовать (изменять его) во время разработки и выполнения приложения, изменяя его свойства. Практически все элементы управления реагируют на определённые события от мыши и клавиатуры.

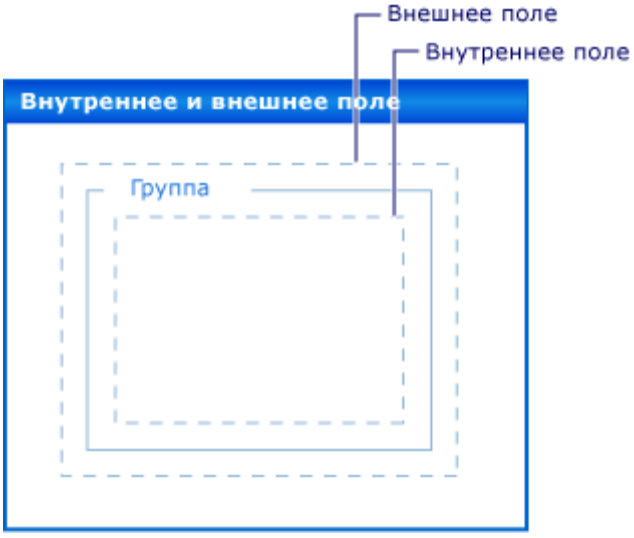
Для элементов управления, представленных в C#, существует набор свойств, которые присущи фактически всем элементам управления (табл. 1).

Таблица 1 – Общие свойства элементов управления

Свойство	Описание
Anchor	Указывает поведение элемента управления при изменении размеров его контейнера.
BackColor	Цвет фона элемента управления
BackgroundImage	Устанавливает или возвращает фоновое изображение, размещаемое внутри элемента управления

Cursor	Вид указателя мыши при позиционировании указателя на элементе управления
Dock	Пристыковывает компонент к краям его контейнера
Enabled	Установка значения свойства Enabled равным true означает, что элемент управления может принимать данные, вводимые пользователем. Установка этого значения равным false означает невозможность приема этих данных
Font	Шрифт, используемый для отображения текста на элементе управления
ForeColor	Цвет текста надписи на элементе управления
Location	Положение компонента на поверхности формы или формы на поверхности экрана. Уточняющее свойство x определяет расстояние от левой границы кнопки до левой границы формы/экрана, уточняющее свойство y — от верхней границы кнопки до верхней границы клиентской области
Name	Имя элемента управления. Это имя может использоваться для ссылки на элемент управления в коде
Size	Размер элемента управления. Уточняещее свойство width определяет ширину, свойство Heigh — высоту
TabIndex	Порядковый номер элемента управления в последовательности перехода между элементами управления внутри его контейнера по клавише табуляции
Tabstop	Указывает доступность элемента управления по клавише табуляции
Text	Содержит текст, связанный с данным элементом управления
ToolTip on…	Получает или задает содержимое подсказки в случае помещения на форму соответствующего элемента управления подсказки
Visible	Указывает видимость элемента управления во время выполнения

Существует два основных свойства, отвечающих за точное расположение компонентов на форме - Margin (определяет пространство вокруг компонента, которое обеспечивает определенное расстояние между границами этого компонента и другими компонентами) и Padding (определяет пространство внутри компонента, которое обеспечивает определенное расстояние между содержимым компонента и границами компонента).



Для обоих свойств можно задать отдельные значения как для всех границ (All), для правой (Right), левой (Left), верхненей (Top), нижней (Bottom).

Для элементов управления можно выделить следующие типы свойств:
1. *Числовой.* Свойства такого типа принимают числовые значения (как правило целые). Например, размер объекта Width (ширина) и Height (высота). Свойства данного типа изменяются непосредственно в окне свойства или при изменении размеров с помощью мыши.
2. *Строковый.* В окне свойства необходимо набрать текст. Например, Text (заголовок формы) или Name (имя объекта.).
3. *Логический.* Свойства такого типа могут принимать только два значения «True» или «False». Например, свойство «Visible» (видимость). Изменение свойства логического типа возможно или выбором значения из предлагаемых вариантов или двойной щелчок мыши в поле свойства изменяет его значение на противоположное.
4. *Фиксированный набор значений.* Требуемое значение свойства выбирается из предлагаемого списка, либо двойной щелчок мыши в окне свойства перебирает все варианты. Например, свойство «WindowState» для формы может принимать одно из значений:
- Normal
- Minimized
- Maximized
5. *Файловый* содержит имя файла, из которого берется значение свойства. Например: свойство BackGroundImage для формы отображает значок, рисунок, который записан в выбранном файле.
6. *Константы определения цвета.* Например: Backcolor (цвет фона), Forecolor (текста) и т.п. Значения этих свойств устанавливается выбором на палитре цветов.
7. *Константы привязки* (свойство Dock)
8. *Константы выравнивания* (свойство Anchor)
Большая часть свойств задается в режиме конструирования, хотя большинство из них может изменяться и во время выполнения приложения. Для этого необходимо обратиться к элементу управления и указать название свойства через точку:

[имя_формы.]имя_ элемента_управления.свойство

Например, button1.BackColor = Color.Green – изменение цвета кнопки на зеленый.

События

Взаимодействие приложения с пользователем описывается *серией событий*, которые генерирует элемент управления и на которые он реагирует. В технологии .NET на события нужно подписываться (subscribe). Под подпиской на событие подразумевается предоставление кода, который должен выполняться при генерации данного события, в виде обработчика событий (event handler). На событие можно подписывать несколько обработчиков, которые все будут вызываться при генерации этого события.

Обработчики событий представляют собой функции. Единственным ограничением для такой функции является то, что ее возвращаемый тип и параметры должны обязательно соответствовать тем, которых требует событие. Поэтому, строго не рекомендуется создание в редакторе программного кода заголовков обработчиков событий «вручную».
События, которые присущи всем элементам управления, описаны в табл. 2.

Таблица 2 События, присущие всем элементам управления

Событие	Описание
Click	Происходит при щелчке на элементе управления. В некоторых случаях это событие происходит также при нажатии пользователем клавиши <Enter>
Doubleclick	Происходит при двойном щелчке на элементе управления. Иногда, обработка события Click для некоторых элементов управления, полностью исключает возможность вызова события Doubleclick

DragDrop	Происходит по завершении операции перетаскивания и оставления — иначе говоря, при перетаскивании объекта поверх элемента управления и освобождении кнопки мыши пользователем
DragEnter	Происходит, когда перетаскиваемый объект перемещается внутрь границ элемента управления
DragLeave	Происходит, когда перетаскиваемый объект покидает границы элемента управления
DragOver	Происходит, когда объект перетаскивается поверх элемента управления
KeyDown	Происходит при нажатии клавиши во время нахождения элемента управления в фокусе. Это событие всегда происходит прежде событий KeyPress и KeyUp
Keypress	Происходит при нажатии клавиши, в то время как элемент управления находится в фокусе. Это событие всегда происходит после события KeyDown и перед событием KeyUp. Различие между событиями KeyDown и KeyPress состоит в том, что KeyDown передает код нажатой клавиши, а KeyPress — соответствующее значение char клавиши.
KeyUp	Происходит при освобождении клавиши, в то время как элемент управления находится в фокусе. Это событие всегда происходит после событий KeyDown и KeyPress
Enter	Происходит, когда элемент управления получает фокус. Это событие не следует использовать для выполнения проверки допустимости элементов управления. В этом случае вместо него следует применять события Validating и Validated
Leave	Происходит, когда элемент управления теряет фокус. Это событие не следует использовать для выполнения проверки допустимости элементов управления. В этом случае вместо него следует применять события validating и validated
MouseDown	Происходит при помещении указателя мыши над элементом управления и нажатии кнопки мыши. Это событие не эквивалентно событию Click, поскольку MouseDown происходит сразу после нажатия кнопки мыши и перед ее освобождением
MouseMove	Происходит непрерывно в процессе перемещения указателя мыши над элементом управления
Mouseup	Происходит при помещении указателя мыши над элементом управления и освобождении кнопки мыши
Paint	Происходит при прорисовке элемента управления
Resize	Происходит при изменении размеров элемента управления
Validated	Запускается, когда элемент управления, свойство CausesValidation которого установлено равным true, готов принять фокус. Это событие запускается по завершении события validating, и оно указывает на завершение проверки
Validating	Запускается, когда элемент управления, свойство CausesValidation которого установлено равным true, готов принять фокус. Проверяемым компонентом является тот, который теряет фокус, а не тот, который его получает

В ***C#*** заголовки обработчиков событий выбранного компонента формируются автоматически в окне программного кода. Синтаксис обработчика события:

private void имя_компонента_событие(object sender, список_параметров)
{
Тело обработчика

}

где ***private*** **–** ключевое слово, которое определяет область действия обработчика событий (обычно в пределах формы);
object sender – элемент управления, который генерирует событие;
список_параметров – список аргументов, которые передаются в процедуру и несут информацию о событии.

Методы

Методы представляют собой подпрограммы, которые манипулируют данными, определенными в классе, а во многих случаях они предоставляют доступ к этим данным. Методы могут вызываться так же, как и любые другие функции, и тем же самым образом использовать возвращаемые значения и параметры. Методы применяются для обеспечения доступа к функциональным возможностям объекта. Они нередко подразумевают использование состояния объекта в своих операциях
Т.о., метод представляет собой законченный фрагмент кода, к которому можно обратиться по имени. Он описывается один раз, а вызываться может столько раз, сколько необходимо. Один и тот же метод может обрабатывать различные данные, переданные ему в качестве аргументов.
Синтаксис метода:

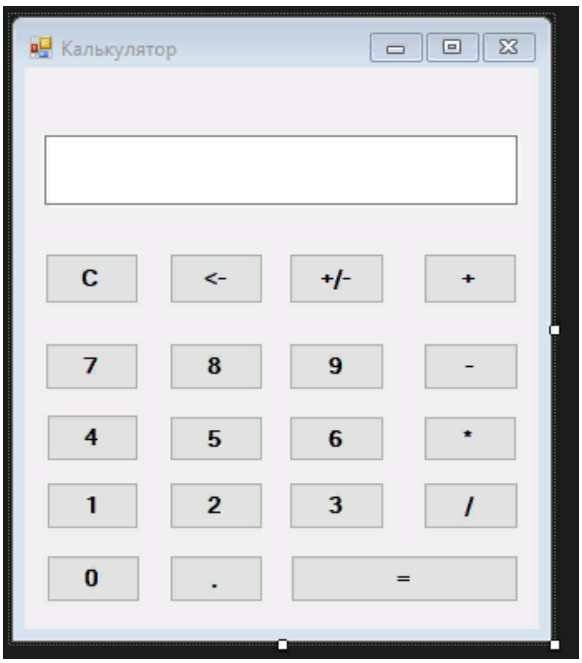
[атрибуты] [спецификаторы] тип имя_метода ([параметры])
тело метода

Здесь первая строка представляет собой заголовок метода. Тело метода, задающее действия, выполняемые методом, чаще всего представляет собой блок — последовательность операторов в фигурных скобках.
Синтаксис вызова метода:

имя_компонента.имя_метода ([параметры])

Задание

Создать графический калькулятор на языке C# с помощью Windows Forms.



Технология работы

1. Разработку калькулятора необходимо начать с разработки формы (Windows Forms) приложения:
   – определить размеры и положение формы;

- задать заголовок формы;
- определить цветовую гамму.

2. На форму нужно поместить кнопки, текстовое поле, а также соответствующие надписи для кнопок.
3. Выполнить настройку свойств элементов управления.
4. Написать обработчики событий для кнопок. Для всех кнопок – цифр и знаков арифметических действий должна вызываться одна процедура обработки нажатия на кнопку.
5. Выполнить тестирование приложения.

Пояснения:

В каждую процедуру обработки события передается ссылка на объект, который вызвал это событие - *sender*.

```
public void OnClick (object  sender,  EventArgs a)
 {
   // обработка события
}
```

Если в приложении есть много объектов, то можно использовать один и тот же обработчик событий.

Например:

```
…
        string s;
       s=(sender  as TextBox).text; // в переменную s будет записан текст из поля ввода
…
```

или

```
        string s;
                    s=(sender as Button).text; ; // в переменную s будет записан текст на кнопке
```

Контрольные вопросы:

1. Какое свойство определяет положение компонента относительно клиентской области родителя?
2. Перечислите этапы разработки Windows-приложения.
3. Что такое событийная процедура?
4. К какому классу относятся элементы управления?
5. Какие способы работы со свойствами компонента Вы знаете?
6. Какой тип данных имеет информация, которая вводится и выводится с помощью компонента «текстовое поле»?