Создание приложений под ОС Windows

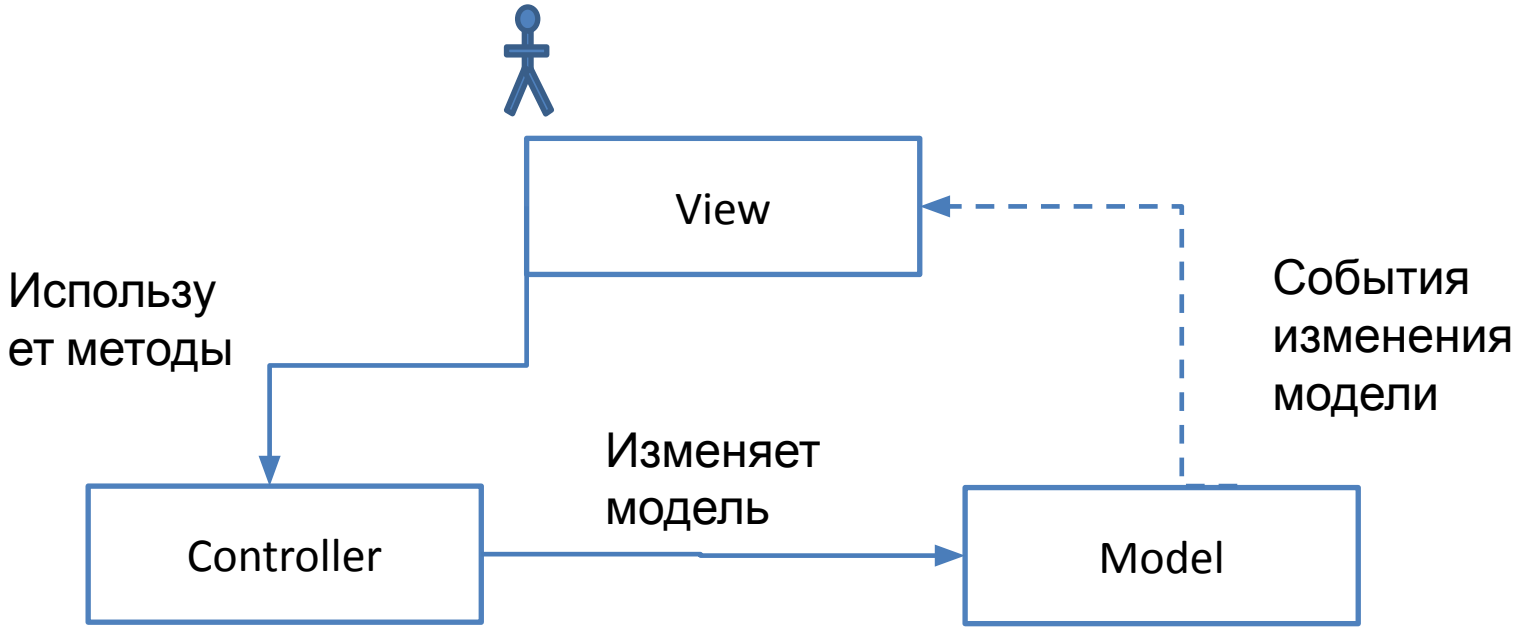
Деление на слои

- Интерфейс
- Бизнес-логика
- Данные

Паттерн Model-View-Controller (MVC)

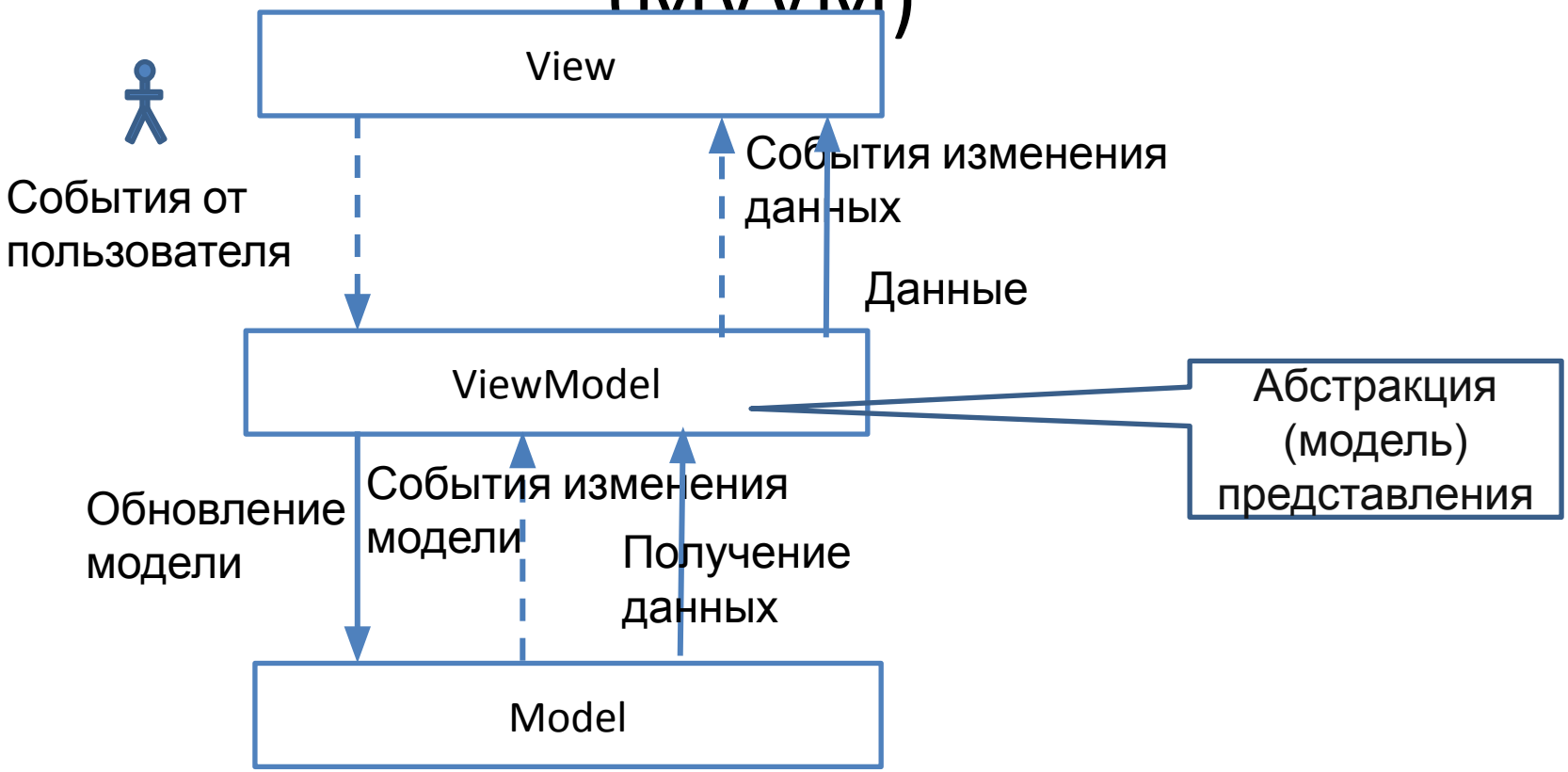
- Модель – содержит в себе функциональную бизнес-логику приложения, должна быть полностью независима от остальных частей приложения: классы, БД, файлы.
- Представление - отображает данные, полученные от Модели (read only): формы, страницы.
- Контроллер – определяет взаимодействие модели и представления.

Паттерн Model-View-Controller (MVC)



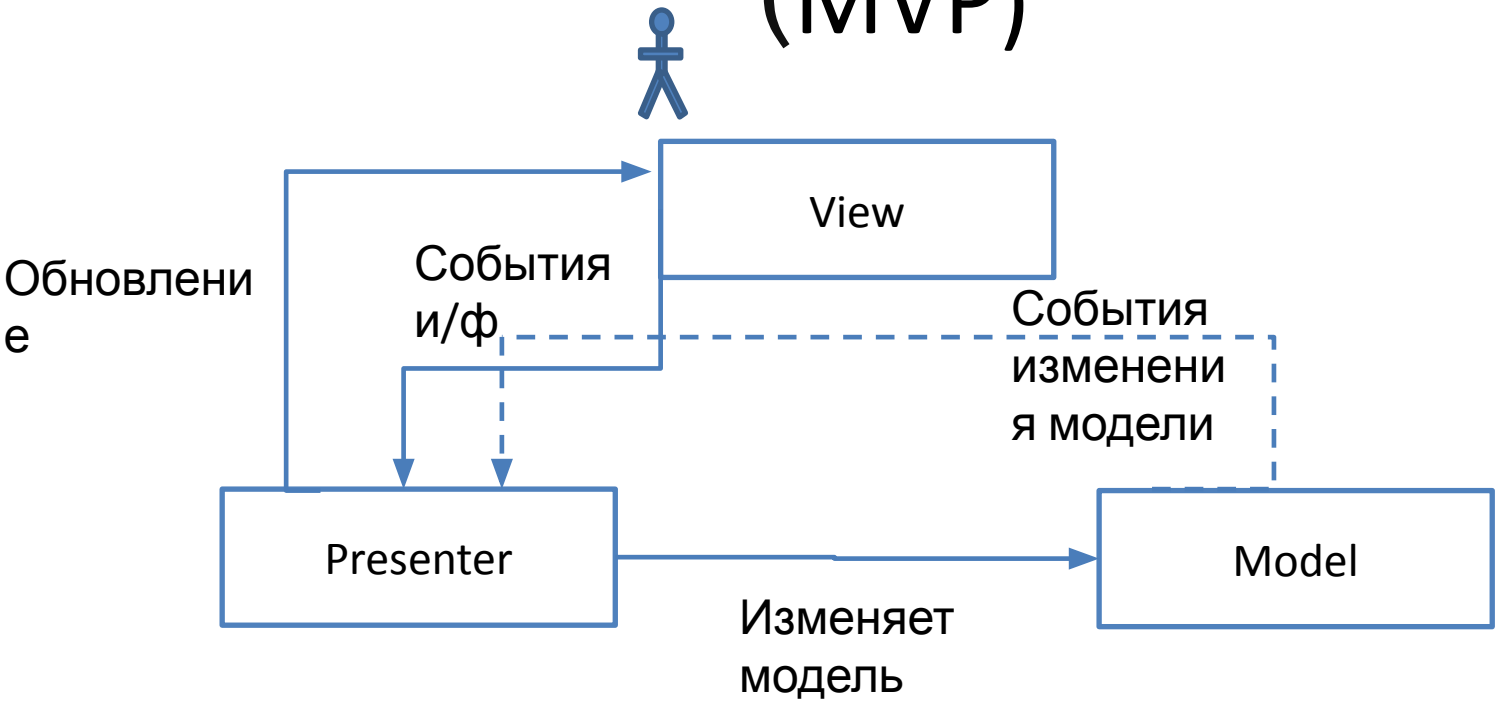
- Контроллер перехватывает событие извне и в соответствии с заложенной в него логикой, реагирует на это событие, изменяя Модель, посредством вызова соответствующего метода.
- После изменения Модель использует событие о том что она изменилась, и все подписанные на это события Представления, получив его, обращаются к Модели за обновленными данными, после чего их и отображают.

Паттерн Model-View-ViewModel (MVVM)



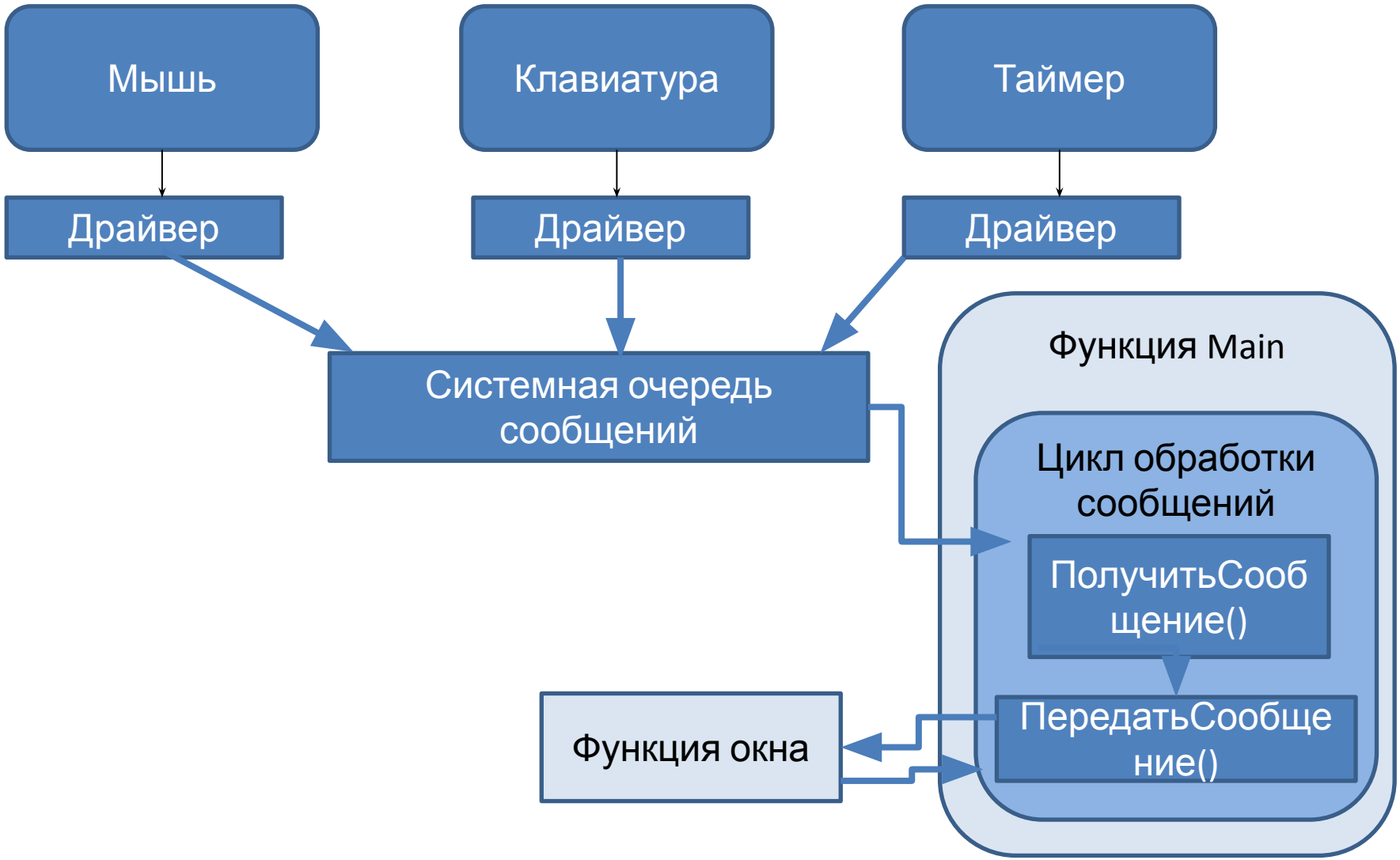
- Представление должно иметь ссылку на источник данных (DataContex), которым в данном случае является View-модель.
- Элементы представления связаны с соответствующими свойствами и событиями View-модели.
- В свою очередь, View-модель реализует специальный интерфейс, который используется для автоматического обновления элементов представления.

Паттерн Model-View-Presenter (MVP)



Каждое представление должно реализовывать соответствующий интерфейс. Интерфейс представления определяет набор функций и событий, необходимых для взаимодействия с пользователем.
Логика представления должна иметь ссылку на экземпляр презентера.
Все события представления передаются для обработки в презентер и практически никогда не обрабатываются логикой представления (в том числе создание других представлений).

Процесс обработки сообщений в Windows Forms



Сообщения

1. **Создание сообщений :** Большинство сообщений создают драйверы устройств ввода и вывода(клавиатура, мышь или таймер) при поступлении аппаратных прерываний.
2. **Очередь сообщений:** Сообщения попадают в системную очередь сообщений Microsoft Windows, из системной очереди сообщения распределяются в очереди сообщений отдельных приложений, для каждого приложения создается своя собственная очередь сообщений. Любое приложение может послать сообщение любому другому приложению, в том числе и само себе.

Сообщения

3. **Обработка сообщений:** Приложение в цикле опрашивает свою очередь сообщений. Обнаружив сообщение, приложение с помощью специальной функции из программного интерфейса Windows (Win32 API) передает его нужному программному модулю (функция окна). Эта функция и выполняет обработку сообщения

Структура приложения Microsoft Windows

- инициализирующая часть запускает цикл обработки сообщений;
- реализация функциональности ложится на программные модули (функции), вызываемые внутри цикла обработки сообщений.

Цикл обработки сообщений

```
MSG msg;
//выборка сообщения из очереди
//приложения
  while(ПолучитьСообщение(msg))
  {
   //передача выбранного из очереди
  //сообщения нужной функции окна
  ПередатьСообщение(msg);
  }
```

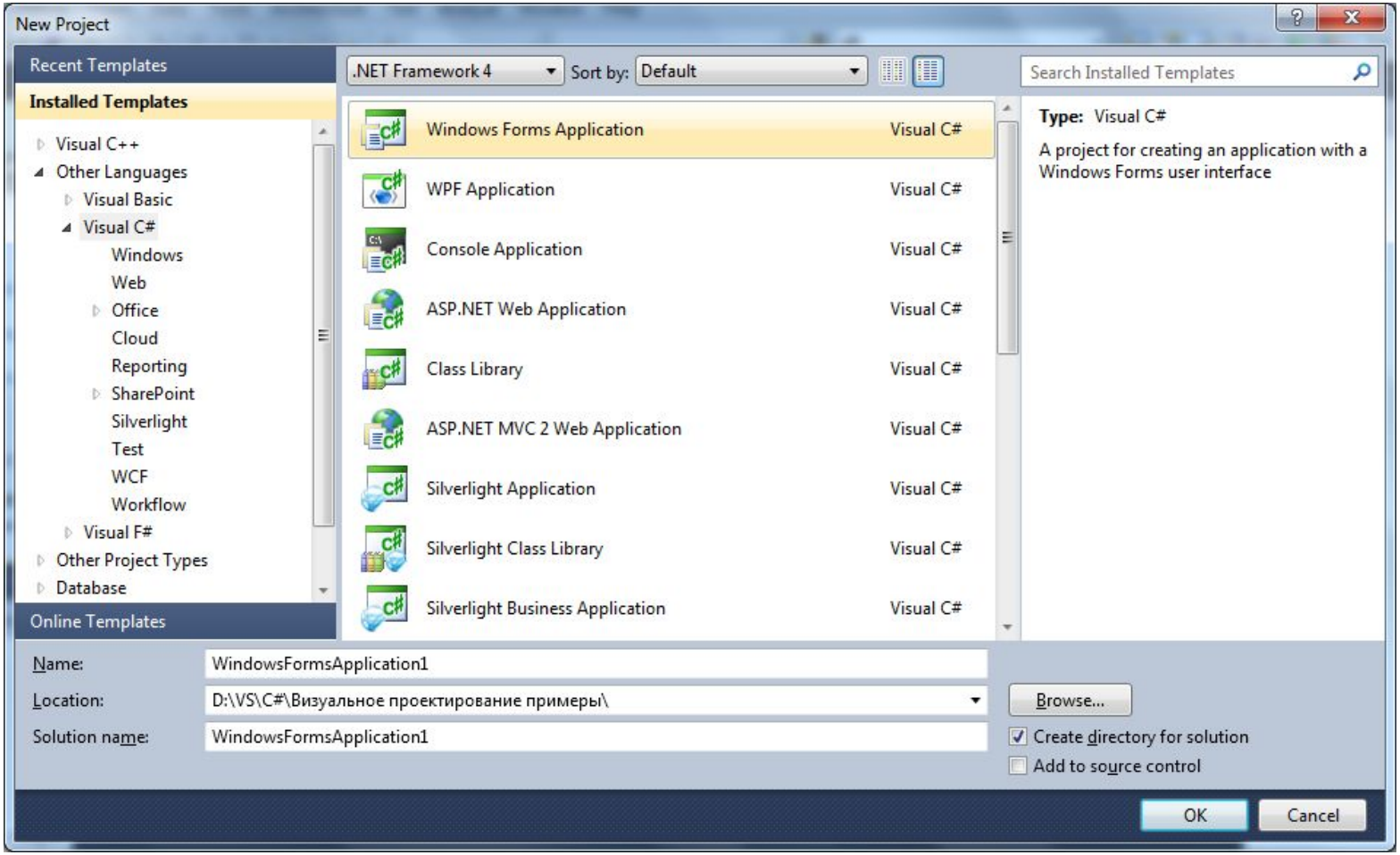
Структура приложения с обработкой сообщений

```
static class Program
  {
     /// <summary>
     /// The main entry point for the application.
     /// </summary>
     [STAThread]
     static void Main()
     {
        Application.EnableVisualStyles();
        Application.SetCompatibleTextRenderingDefault(false);
        Application.Run(new Form1());
     }
  }
```

Процесс создания Windows-приложения состоит из двух основных этапов:

1. Визуальное проектирование, то есть создание внешнего облика приложения.
2. Определение поведения приложения путем написания процедур обработки событий.

Создание Windows-приложения



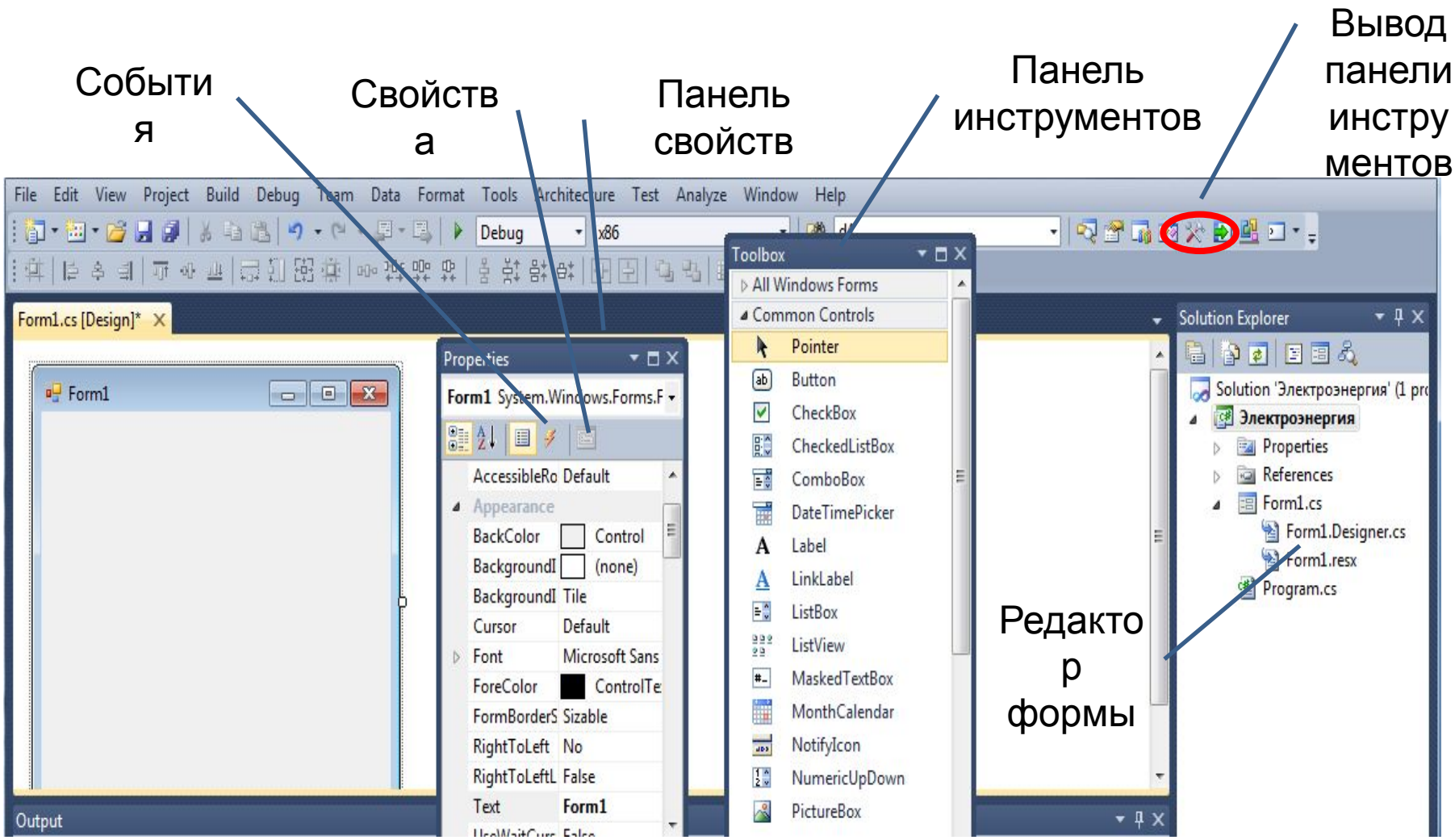
Базовые элементы

- Процесс создания программы состоит из:
  - создания формы программы (диалогового окна), создается путем добавления управляющих элементов на форму;
  - функций обработки событий, выполняют настройку требуемого поведения управляющих элементов.
- Вид элемента и его поведение определяют значения свойств.
- Основную работу в программе выполняют функции обработки событий.

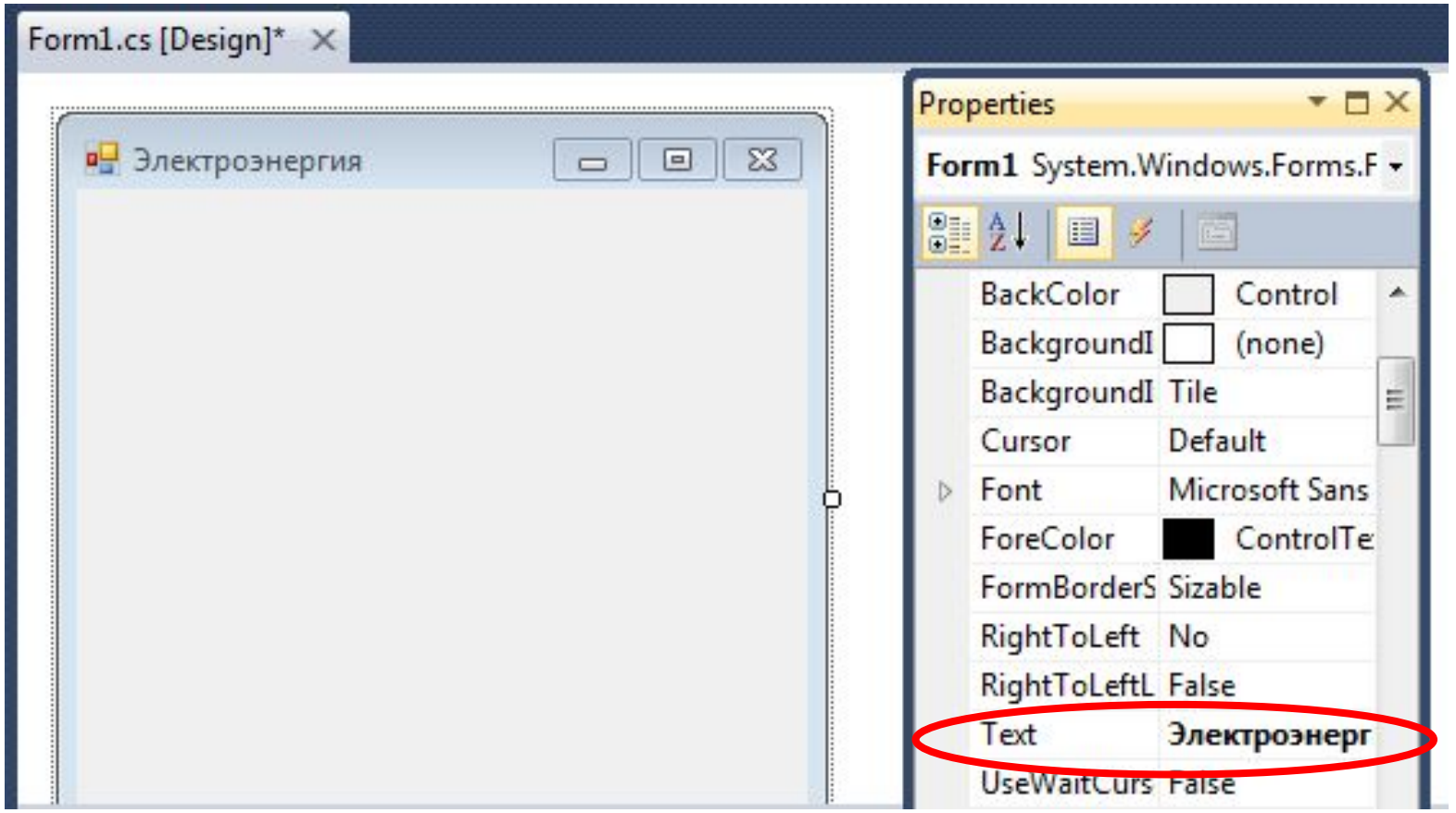
Базовые элементы

- В форме приложения должны быть компоненты, обеспечивающие взаимодействие с пользователем (базовые):
  - Label – поле вывода текста;
  - TextBox – поле редактирования текста;
  - Button – командная кнопка;
  - CheckBox – независимая кнопка выбора;
  - RadioBox – зависимая кнопка выбора;
  - ListBox – список выбора;
  - ComboBox – комбинированный список выбора;
  - MainMenu – главное меню программы;
  - ContextMenu – контекстное меню программы.

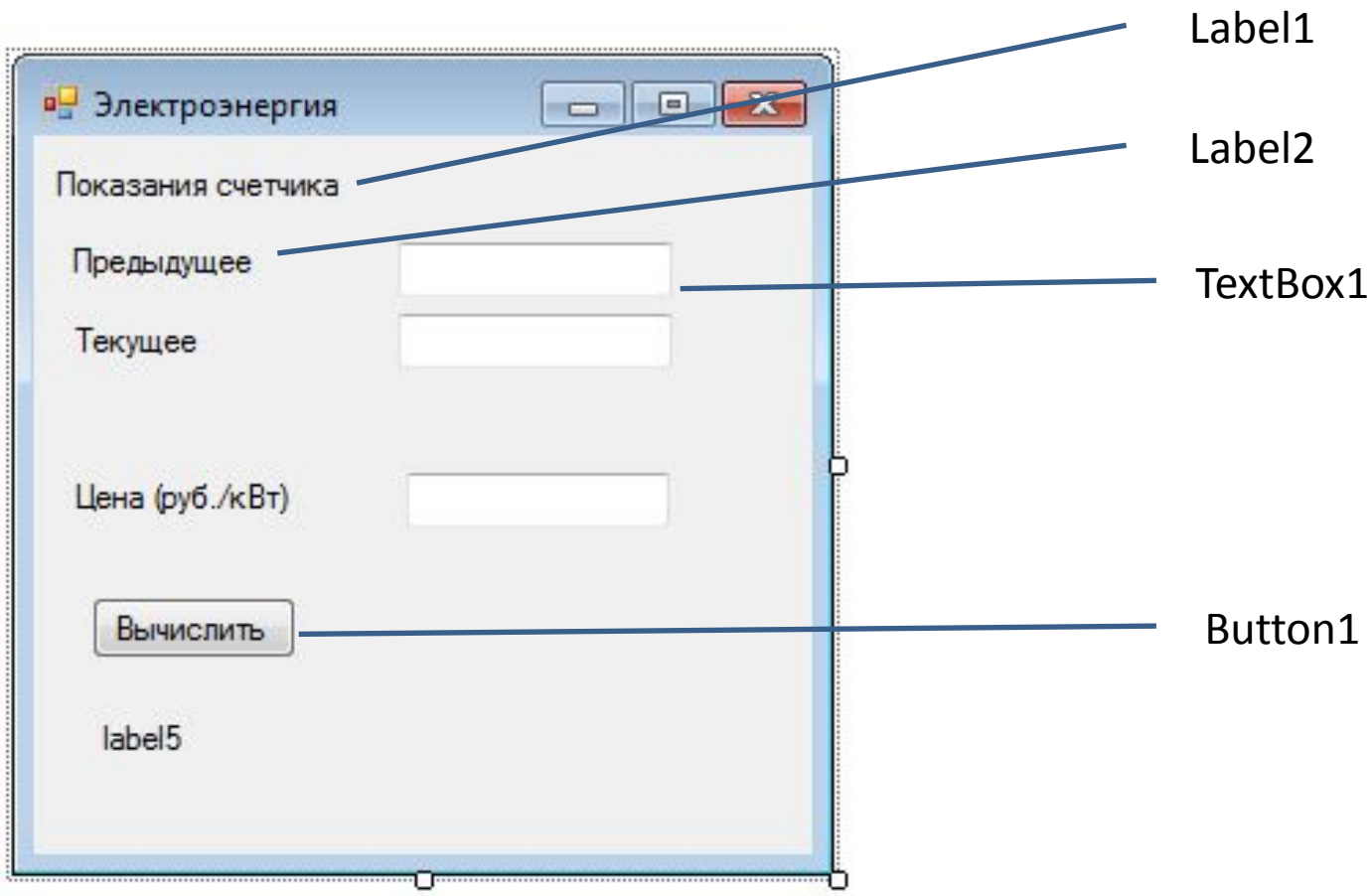
Визуальное проектирование



Настройка свойств формы



Визуальное проектирование



Определение поведения приложения

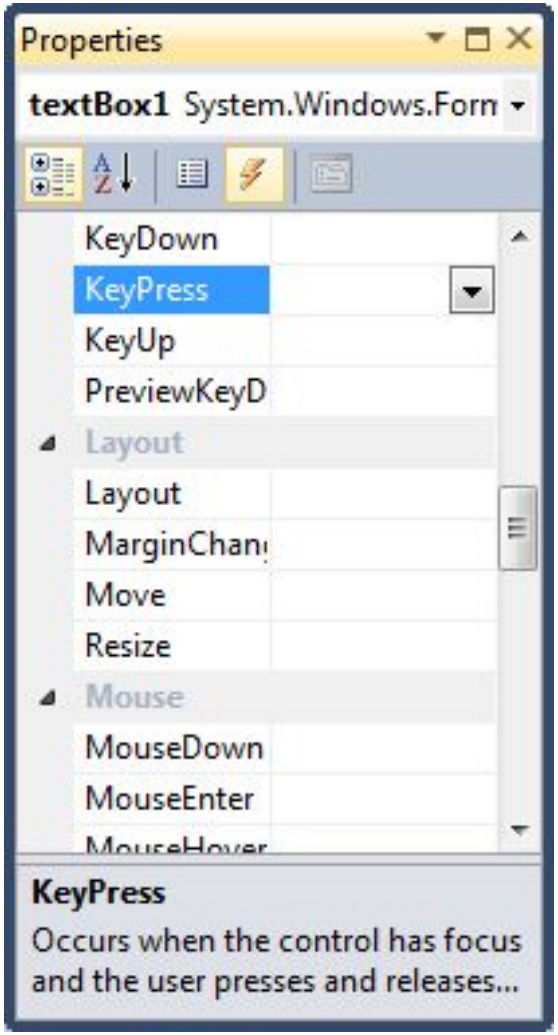
```
//Program.cs
using System;
. . . . . .
using System.Windows.Forms;
namespace Электроэнергия
{
   static class Program
   {
      /// <summary>
      /// The main entry point for the application.
      /// </summary>
      static void Main()
      {
         Application.EnableVisualStyles();
         Application.SetCompatibleTextRenderingDefault(false);
         Application.Run(new Form1());
      }
   }
}
```

```
//Form1.cs
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace Электроэнергия
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }

      private void Form1_Load(object sender, EventArgs e)
      {
        button1.Enabled = false;//блокируем кнопку
      }
   }
}
```

Задаем поведение управляющих элементов



Обработка события для текстовой строки:

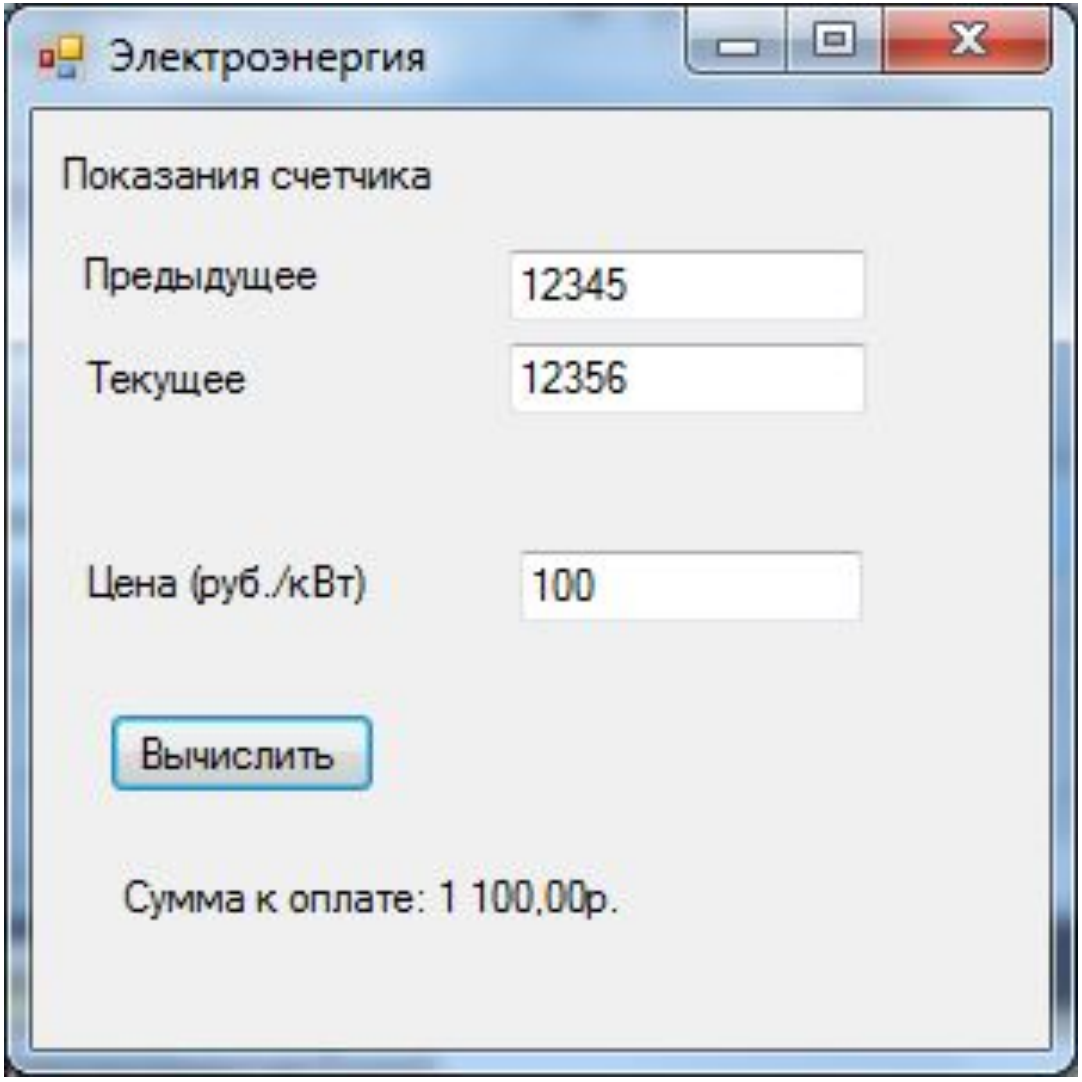
```
private void textBox1_KeyPress(object sender,
KeyPressEventArgs e)
      {
         if (!Char.IsDigit(e.KeyChar) &&
!(e.KeyChar.ToString() == "," &&
textBox1.Text.IndexOf(',') == -1))
/*сообщение о нажатии клавиши не должно
передаваться элементу управления*/
            e.Handled = true;
if(e.KeyChar.Equals((char)13))textBox2.Focus(); //Enter
      }
```

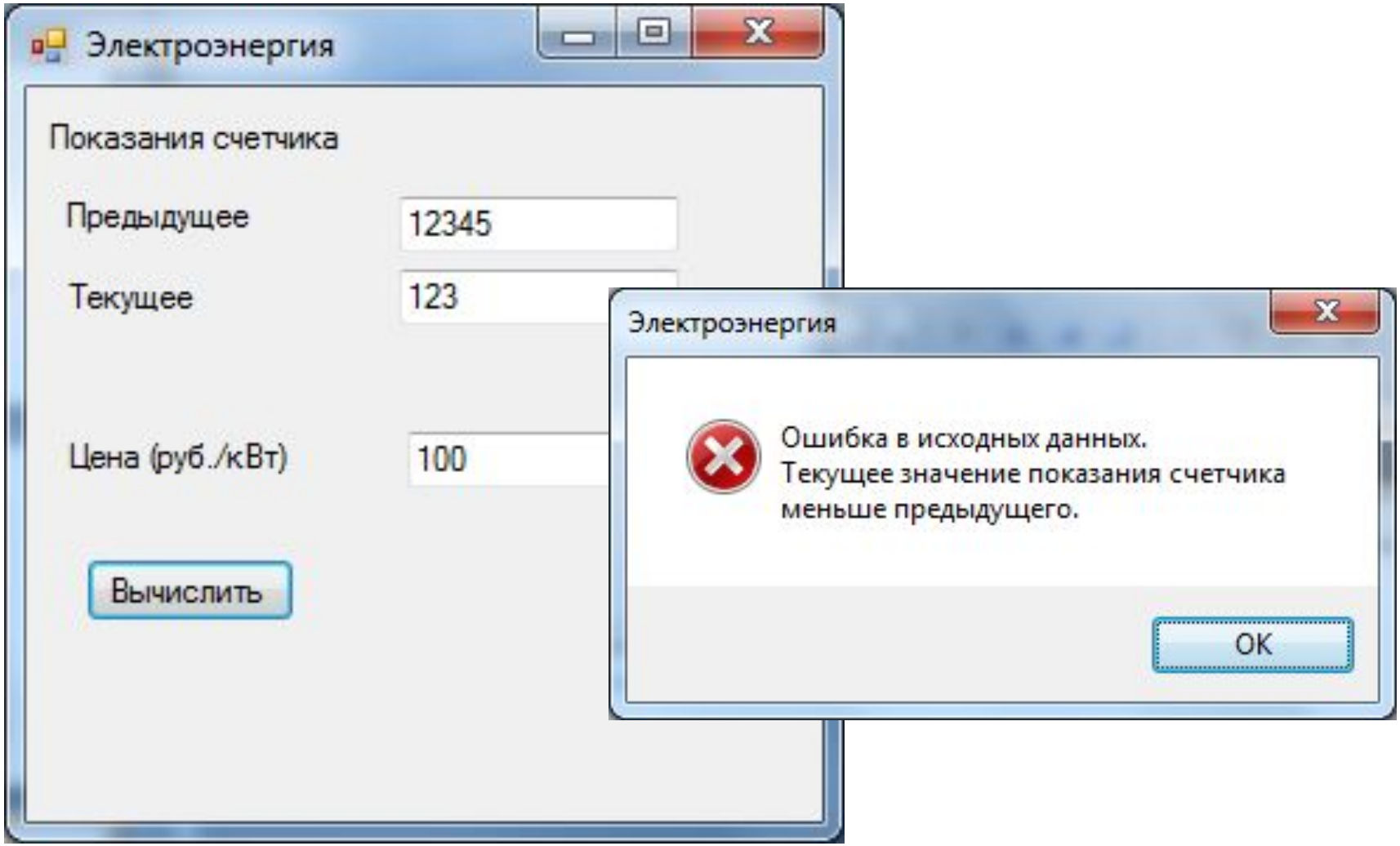
```
//проверка заполнения текстовых полей
private void textBox3_KeyUp(object sender, KeyEventArgs e)
      {
         if ((textBox1.Text.Length > 0) && (textBox2.Text.Length > 0)
              && (textBox3.Text.Length > 0))
        button1.Enabled = true;
        else
       button1.Enabled = false;
      }
```

```
private void button1_Click(object sender, EventArgs e)
        {
            double curr, prev, price, all;
            label5.Text = "";
            try
            {
                prev = Convert.ToDouble(textBox1.Text);
                curr = Convert.ToDouble(textBox2.Text);
                price = Convert.ToDouble(textBox3.Text);
                if (curr >= prev)
                {
                    all = (curr - prev) * price;
                    label5.Text = "Сумма к оплате: " +
    all.ToString("C");
                }
```

```
else
            {
                MessageBox.Show("Ошибка в исходных данных.\n"+
          "Текущее значение показания счетчика\n"+
           "меньше предыдущего. ", "Электроэнергия",
                MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
           }//try
           catch(Exception exc)
           {
              MessageBox.Show("Ошибка в исходных данных.\n"+
              "Исходные данные имеют неверный формат."+
               exc.Message, "Электроэнергия", MessageBoxButtons.OK,
               MessageBoxIcon.Error);
           }

      }
```


Электроэнергия
Показания счетчика
Предыдущее
12345
Текущее
12356
Цена (руб./кВт)
100
Вычислить
Сумма к оплате: 1 100,00р.


Электроэнергия
Показания счетчика
Предыдущее
12345
Текущее
123
Цена (руб./кВт)
100
Вычислить
Электроэнергия
Ошибка в исходных данных.
Текущее значение показания счетчика
меньше предыдущего.
OK

Основные типы Windows.Forms

Класс	Назначение
ColorDialog, FileDialog, FontDialog, PrintPreviewDialog	Примеры стандартных диалоговых окон для выбора цветов, файлов, шрифтов, окно предварительного просмотра
Menu, MainMenu, MenuItem, ContextMenu	Классы выпадающих и контекстных меню
Clipboard, Help, Timer, Screen, ToolTip, Cursors	Вспомогательные типы для организации графических интерфейсов: буфер обмена, помощь, таймер, экран, подсказка, указатели мыши
StatusBar, Splitter, ToolBar, ScrollBar	Примеры дополнительных элементов управления, размещаемых на форме: строка состояния, разделитель, панель инструментов и т. д.

Наиболее часто используемые события:

- Activated — получение формой фокуса ввода;
- Click, Doubleclick — одинарный и двойной щелчки мышью;
- Closed — закрытие формы;
- Load — загрузка формы;
- KeyDown, KeyUp — нажатие и отпускание любой клавиши и их сочетаний;
- Keypress — нажатие клавиши, имеющей ASCII-код;
- MouseDown, MouseUp — нажатие и отпускание кнопки мыши;
- MouseMove — перемещение мыши;
- Paint — возникает при необходимости прорисовки формы.

Класс Control

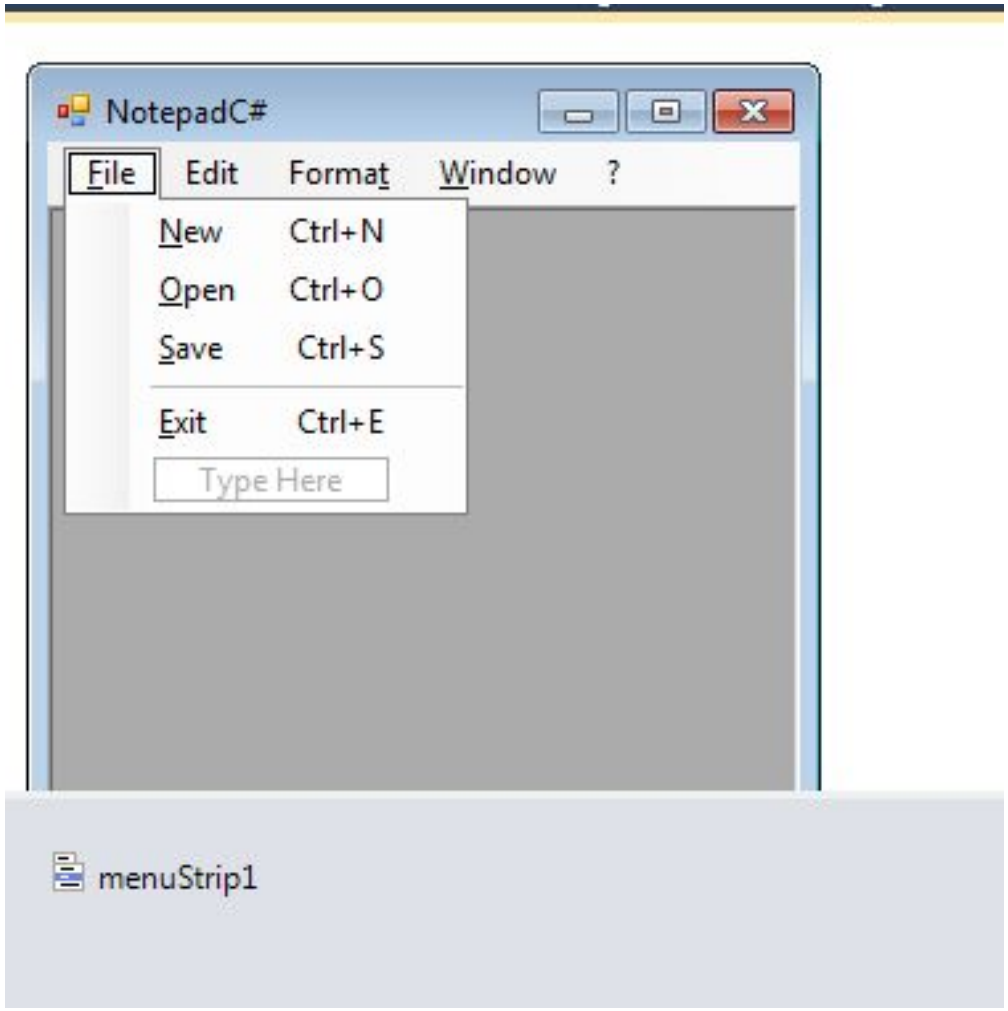
- **Класс Control** является базовым для всех отображаемых элементов.
- Он реализует базовую функциональность интерфейсных элементов: методы обработки ввода пользователя с помощью мыши и клавиатуры, определяет размер, положение, цвет фона и другие характеристики элемента.

Элементы управления

- **Метка Label** предназначена для размещения текста на форме.
- **Кнопка Button.** Основное событие, обрабатываемое кнопкой, — щелчок мышью (Click).
- **Поле ввода TextBox** позволяет пользователю вводить и редактировать текст, который запоминается в свойстве Text.

Элементы управления

- **Главное меню** размещается на форме таким же образом, как и другие компоненты. При этом значок располагается под заготовкой формы, а среда переходит в режим редактирования пунктов меню. Каждый пункт меню представляет собой объект типа MenuItem, и при вводе пункта меню мы задаем его свойство Text.
- **Контекстное меню ContextMenu** — это меню, которые вызывается во время выполнения программы по нажатию правой кнопки мыши на форме или элементе управления.


NotepadC#
File
Edit
Format
Window
?
New Ctrl+N
Open Ctrl+O
Save Ctrl+S
Exit Ctrl+E
Type Here
menuStrip1

Элементы управления

- **Флажок CheckBox** используется для включения-выключения пользователем какого-либо режима.
- **Переключатель RadioButton** позволяет пользователю выбрать один из нескольких предложенных вариантов, поэтому переключатели обычно объединяют в группы.
- **Панель GroupBox** служит для группировки элементов на форме, например для того, чтобы дать общее название и визуально выделить несколько переключателей или флажков, обеспечивающих выбор связанных между собой режимов.

Элементы управления

- **Список ListBox** служит для представления перечня элементов, в которых пользователь может выбрать одно (свойство SelectionMode равно One) или несколько значений (свойство SelectionMode равно MultiSimple или MultiExtended).

Диалоговые окна

- Класс Form представляет собой заготовку формы, от которой наследуются классы форм приложения.
- Диалоговое окно характеризуется:
  - неизменяемыми размерами (FormBorderStyle = FixedDialog);
  - отсутствием кнопок восстановления и свертывания в правом верхнем углу заголовка формы (MaximizeBox = False, MinimizedBox = False);
  - наличием кнопок наподобие ОК, подтверждающей введенную информацию, и Cancel, отменяющей ввод пользователя, при нажатии которых окно закрывается (AcceptButton = имя_кнопки_ОК, Cancel Button = имя_кнопки_ Cancel);
  - установленным значением свойства DialogResult для кнопок, при нажатии которых окно закрывается.

Диалоговые окна

Добавление сотрудника

ФИО

Возраст

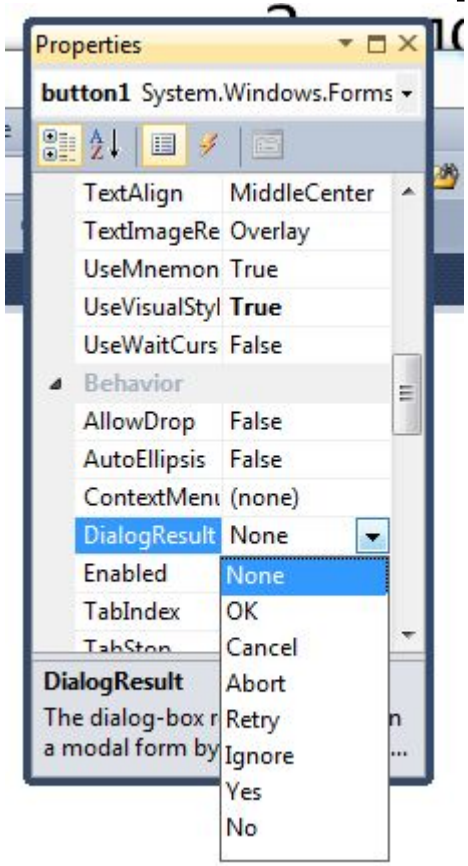
Должность

Оклад

OK

Cancel

Обработчик для вызова диалогового окна



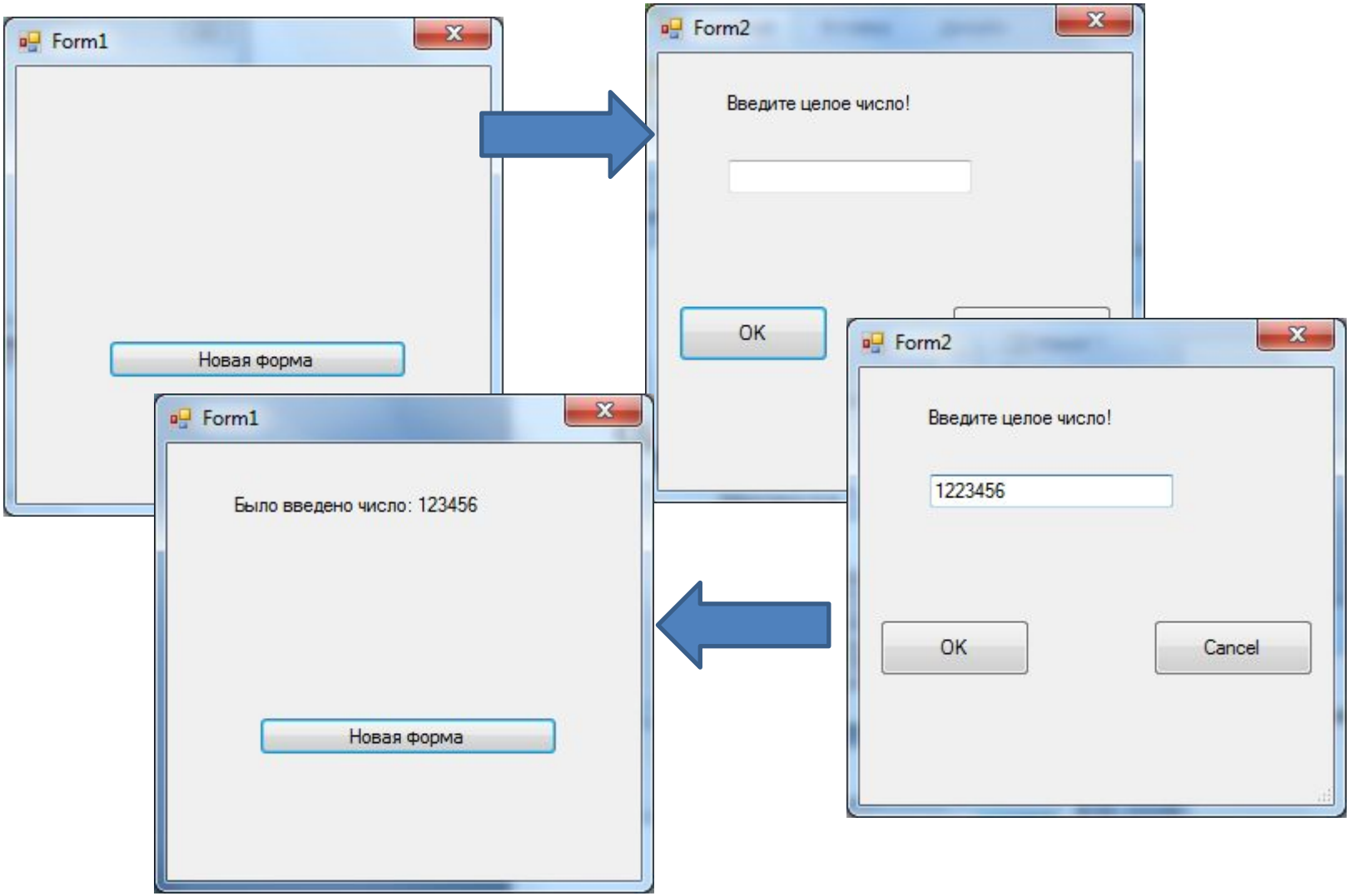

```
private void cmdAdd_Click(object sender, EventArgs e)
    {
       Employee empl=new Employee();
       AddForm dlg = new AddForm(); // создать диалоговое
окно
       dlg.ShowDialog(); //вызов
       if(dlg.DialogResult==DialogResult.OK)
          {
     . . . . . .
          }
          else
          {
             MessageBox.Show("Введите данные", "Employee
Administration", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);
             return;
          }

    }
```

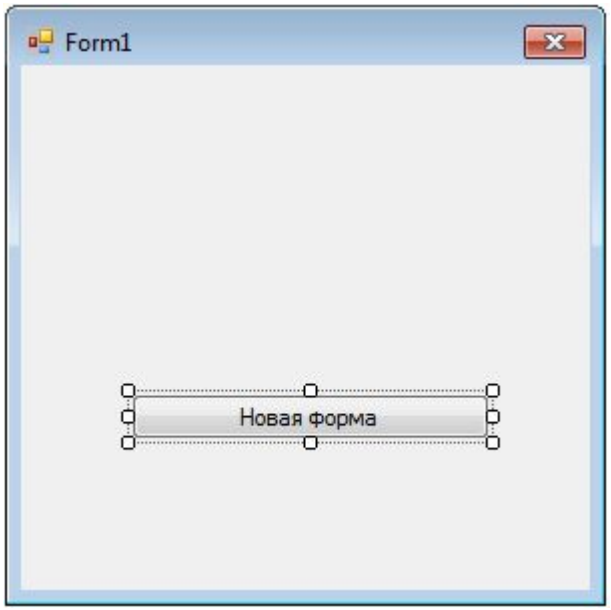
Задача 1

- Пример передачи информации из диалогового окна Form2 в диалоговое окно Form1:
- Form1 вызывает Form2;
- пользователь вводит в Form2 число;
- Form1 получает число, введенное в Form2.

Пример передачи информации из одного диалогового окна в другое



Проектируем формы



Данные для Формы 2

```
//обработчик для ввода цифр
 public partial class Form2 : Form
    {
     public int number;//число
     public bool ok;//признак правильного
    ввода
```

Методы для Формы 2

```
//обработчик для ввода цифр
private void textNumber_KeyPress(object sender,
    KeyPressEventArgs e)
        {
            if (e.KeyChar >= '0' && e.KeyChar <= '9')
            return;
            if (e.KeyChar.Equals((char)13))
            button1.Focus();
            e.Handled = true;
        }
```

Методы для Формы 2

```
//обработчик кнопки
private void button1_Click(object sender, EventArgs e)
      {
         if (textNumber.Text == "")
         {
             ok = false;
         }
         else
         {
            number = Convert.ToInt32(textNumber.Text);
            ok = true;
         }
         Close();
      }
```

Методы для Формы 1

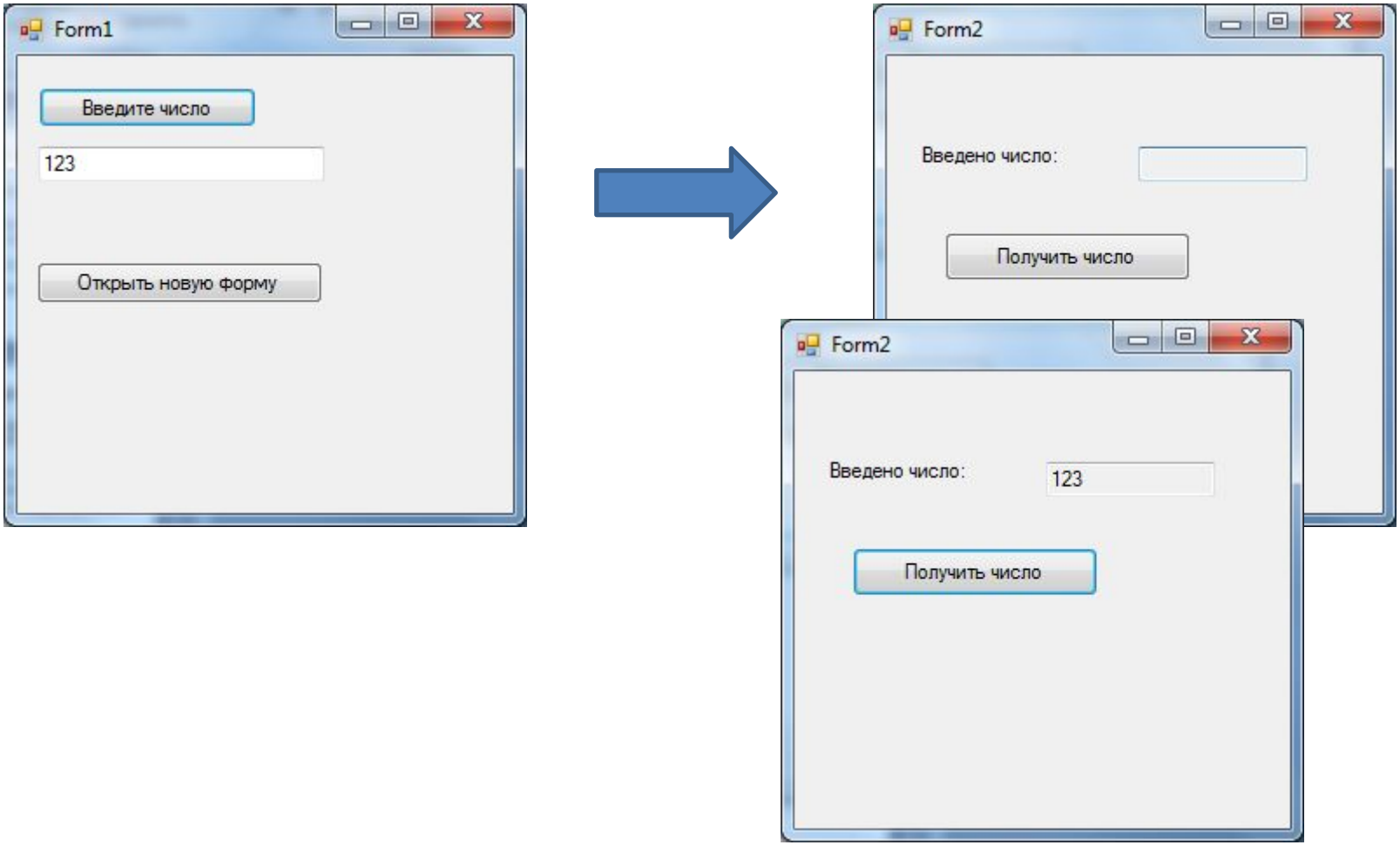
```
//обработчик кнопки
 private void button1_Click(object sender, EventArgs e)
      {
         Form2 form2 = new Form2();
         form2.ShowDialog();
         if (form2.DialogResult == DialogResult.OK)
         {
            if (form2.ok == true) label1.Text ="Было введено число: "+
     form2.number.ToString();
            else
               label1.Text = "Число не было введено!";
         }
         else
            label1.Text = "Число не было введено!";

      }
```

Задача 2

- Пример передачи информации из диалогового окна Form1 в диалоговое окно Form2:
- пользователь вводит в Form1 число;
- Form1 вызывает Form2;
- Form2 получает число, введенное в Form1.

Пример передачи информации из одного диалогового окна в другое



```
public partial class Form1 : Form
    {
        public int number; //число
        public Form1()
        {
            InitializeComponent();
             OpenForm.Enabled =
    false;//блокировка
        }
```

```
//обработчик текстового поля
private void EnterNumber_KeyPress(object
    sender, KeyPressEventArgs e)
        {
            if (e.KeyChar >= '0' && e.KeyChar <= '9')
                return;
            if (e.KeyChar.Equals((char)13))
                button1.Focus();
            e.Handled = true;
        }
```

```
//обработчик кнопки для ввода числа
private void EnterNumber_Click(object sender, EventArgs e)
        {
            try
            {
                number = Convert.ToInt32(textBox1.Text);
                OpenForm.Enabled = true;
            }
            catch (FormatException)
            {
                number = 0;
                MessageBox.Show("Ошибка в исходных
    данных.\n" + "Исходные данные имеют неверный
    формат.", "Передача данных", MessageBoxButtons.OK,
    MessageBoxIcon.Error);
            }
        }
```

```
//обработчик кнопки Открыть форму
private void OpenForm_Click(object sender,
    EventArgs e)
        {
            Form2 f2 = new Form2();
            f2.Owner = this;//запомнить
    владельца
            f2.Show();
        }
```

```
//обработчик кнопки Получить число
private void GetNumber_Click(object sender,
    EventArgs e)
        {
     //получить владельца
            Form1 f1 = (Form1)this.Owner;
            textBox1.Text = f1.number.ToString();
        }
```