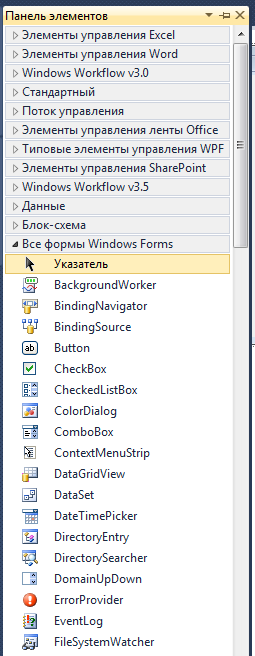
## Лабораторная работа № 4. Элементы управления (компоненты) в *C*#

Форма по своей сути является контейнером и служит для размещения на ней элементов управления (компонентов). Располагая различные элементы управления на форме, разработчик формирует визуальный интерфейс приложения, а также заранее продумывает его основные функциональные возможности. Формы определены в пространстве имен *System.Windows.Forms* (оно указано в директивах *using* в файле *Program.cs*, содержащем класс *Form*). Большинство элементов управления в каркасе *.NET* являются производными от класса *System.Windows.Forms.Control*. Этот класс определяет основные функциональные возможности элементов управления, вследствие чего многие свойства и события элементов управления идентичны. Многие из этих классов сами являются базовыми для других элементов управления.



*Рис. 4.1.* Панель элементов

Элементы управления помещают на форму с помощью панели инструментов Панель элементов (рис. 4.1) (если она не открыта, выполните команду *Вид* – *Панель элементов*).

Окно панели инструментов состоит из большого числа закладок (*Элементы управления Excel*, *Стандартные элементы управления*, *Все элементы управления WPF* и др.), которые облегчают процесс поиска нужного элемента управления. Наиболее часто употребляемой закладкой для приложений *Windows Forms* является *Все формы Windows Forms*. Здесь отображаются все элементы управления, применяемые в приложениях *Windows Forms*, в алфавитном порядке.

Для использования панели инструментов Панель элементов просто первый раз щелкните мышкой на один из элементов управления, и потом второй раз – на форме. Выбранный объект появится на форме, и им можно будет манипулировать с помощью мыши. Также можно после первого щелчка на панели компонентов, зажав на форме кнопку мыши, очертить границы компонента. Кроме того, можно, зажав иконку компонента на панели, дотащить его до нужного места на форме.

Подробнее рассмотрим некоторые наиболее часто используемые элементы управления.

Найдите компонент **Button** (кнопка****). Щелкните по нему мышью в панели Панель элементов, затем щелкните мышью на форме.

В комбинированной панели выбора объекта окна Свойства появился еще один объект – *button1*. Некоторые свойства (такие, как *Name*, *Text*, *Location* (*X*, *Y*), *Size* (*Width*, *Height*), *Cursor, BackColor*) вам уже известны, они совпадают со свойствами формы. Перечислим некоторые другие свойства компонента Button (табл. 4.1).

*Таблица 4.1*

**Некоторые свойства элемента управления Button**

|  |  |
| --- | --- |
| Свойство | Описание |
| *Enabled* | Значение *True* этого свойства обеспечивает доступность компонента для мыши, клавиатуры и событий таймера |
| *Font* | Свойство объектного типа, контролирует отображение текста на кнопке |
| *Visible* | Определяет видимость компонента во время выполнения приложения |
| *Left,*  *Top* | Используются для задания расстояния от левой/верхней границы компонента до края формы, не отображаются в окне Свойств |
| *FlatStyle* | Изменяет стиль кнопки |
| *Image* | Задает изображение (растровое, пиктограмму и т. п.), которое будет отображаться на кнопке |
| *ImageAlign* | Определяет позицию изображения на кнопке |

### Из событий компонента Button наиболее активно используется событие Click, которое происходит при щелчке мышью по кнопке, а также при нажатии клавиши *Enter* в случае, когда кнопка находится в фокусе (выделена).

Положите на форму компонент **TextBox** (окно редактирования ). Этот элемент управления используется для получения текста от пользователя и для его отображения. Некоторые свойства приведены в табл. 4.2.

*Таблица 4.2*

**Некоторые свойства элемента управления TextBox**

|  |  |
| --- | --- |
| Свойство | Описание |
| *BorderStyle* | Определяет вид границы между внутренней частью компонента и клиентской областью приложения. Имеет три возможных значения: *None* (нет границы), *Fixed3D* (объемное) и *FixedSingle* (плоское) |
| *CharacterCasing* | Определяет регистр символов, отображаемых в компоненте |
| *BackColor* | Цвет фона компонента |
| *MaxLength* | Ограничивает количество символов, которые можно ввести |
| *PasswordChar* | Используется для скрытия вводимой информации. Если значение свойства равно #0, то текст отображается нормально. Любое другое значение этого свойства определяет отображаемые в окне ввода символы |
| *ReadOnly* | Значение свойства, равное *True*, запрещает редактирование текста, отображаемого компонентом |
| *Text* | Свойство используется для чтения текста или определения нового значения текста |
| *TextAlign* | Получает или задает выравнивание текста в элементе управления |

При изменении содержимого компонента TextBox происходит событие TextChange.

Положите на форму компонент **Label** (метка ). Компонент Label – это графический элемент управления, предназначенный для отображения текста на форме. Некоторые его свойства приведены в табл. 4.3.

*Таблица 4.3*

**Некоторые свойства элемента управления Label**

|  |  |
| --- | --- |
| **Свойство** | **Описание** |
| *AutoSize* | Значение свойства, равное *True*, приводит к автоматическому изменению размеров метки в соответствии с длиной текста |
| *BorderStyle* | Задает стиль рамки вокруг надписи, по умолчанию рамка отсутствует |
| *FlatStyle* | Определяет способ отображения элемента управления |
| *Image* | Задает одиночное изображение, которое будет отображаться в надписи |
| *ImageAlign* | Указывает выравнивание изображения в метке |
| *Text* | Определяет строку, отображаемую в компоненте |
| *TextAlign* | Задает выравнивание текста в метке |

Проследите за изменением состояния компонентов Label, TextBox и Button при изменении свойств, перечисленных выше. При необходимости узнать подробнее о свойствах, методах и событиях любого компонента можно воспользоваться справочной системой.

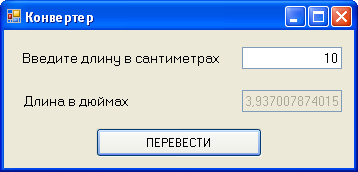
**Упражнение 4.1.** Напишите программу «Конвертер» для перевода сантиметров в дюймы.

**Решение**

Создайте новый проект *Ex1* в новом решении *Lab4*.

**1‑й этап.** Создадим визуальный интерфейс приложения (рис. 4.2).

*Рис. 6.2.* Форма из упражнения 6.1



Измените значения свойств формы следующим образом:

|  |  |
| --- | --- |
| Свойство | Значение |
| *Name* | *MainForm* |
| *Text* | Конвертер |

Поместите на форму компонент Label, измените значения свойств следующим образом:

|  |  |
| --- | --- |
| Свойство | Значение |
| *Name* | *CentiMLabel* |
| *Text* | Введите длину в сантиметрах |
| *Font*.*Size* | *10* |

Поместите на форму компонент TextBox, измените значения свойств следующим образом:

|  |  |
| --- | --- |
| Свойство | Значение |
| *Name* | *CentiMTextBox* |
| *Text* | 0 |
| *Font*.*Size* | 10 |
| *TextAlign* | *Right* |

Поместите на форму компонент Button, измените значения свойств следующим образом:

|  |  |
| --- | --- |
| Свойство | Значение |
| *Name* | *ExecuteButton* |
| *Font*.*Size* | 10 |
| *Text* | Перевести |

Поместите на форму компонент Label, измените значения свойств следующим образом:

|  |  |
| --- | --- |
| Свойство | Значение |
| *Name* | *InchLabel* |
| *Font*.*Size* | 10 |
| *Text* | Длина в дюймах |

Поместите на форму компонент TextBox, измените значения свойств следующим образом:

|  |  |
| --- | --- |
| Свойство | Значение |
| *Name* | *InchTextBox* |
| *Font*.*Size* | 10 |
| *Enabled* | *False* |
| *Text* | 0 |
| *TextAlign* | *Right* |

Измените размеры формы в соответствии с расположенными компонентами (рис. 4.2).

**2‑й этап.** Создание программного кода.

Создайте обработчик события Click кнопки *ExecuteButton*. Для этого достаточно выполнить двойной щелчок мышью по кнопке. При нажатии на кнопку «*Перевести*» требуется информацию, введенную пользователем в компонент *CentiMTextBox*, преобразовать в соответствии с математической формулой: *дюймы = сантиметры / 2,54*, – и отобразить результат в компоненте *InchTextBox*. Итак, в обработчике события Click кнопки *ExecuteButton* введем следующие операторы:

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

{

try

{

InchTextBox.Text = (double.Parse(CentiMTextBox.Text)

/2.54).ToString();

}

catch

{

MessageBox.Show("Некорректный ввод данных", "Ошибка", MessageBoxButtons.OK , MessageBoxIcon.Error);

CentiMTextBox.Text = "0";

InchTextBox.Text = "0";

}

}

Отметим, что информация, введенная пользователем в компонент *CentiMTextBox*, хранится в свойстве *Text* в строковом формате, а значение длины в сантиметрах – это вещественное число. Для преобразования строковой величины в вещественное число используется метод *Parse*() класса *double*, для обратного преобразования используется метод *ToString*(). Кроме того, при попытке перевести строку в число может возникнуть ошибка. Оператор *try* позволяет обработать эту исключительную ситуацию программным путем и вывести сообщение об ошибке.

Синтаксис оператора:

try

{

… //контролируемый блок

}

catch //один или несколько блоков

{

… //блок обработки исключений

}

finally

{

… //блок завершения

}

Программные инструкции, которые нужно проконтролировать на предмет исключений, помещаются в блок *try*. Если исключение возникает в этом блоке, оно дает знать о себе выбросом определенного рода информации. Выброшенная информация может быть перехвачена и обработана соответствующим образом с помощью блока *catch*. Весь код, который должен быть обязательно выполнен при выходе из блока *try*, в любом случае, произошла ошибка или нет, помещается в блок *finally*.

Изучению способов использования обработчиков исключительных ситуаций посвящена лекция 4.

Сохраните изменения, внесенные в проект. Запустите приложение.

Создайте обработчик события KeyDown компонента *CentiMTextBox*. Укажите в нем следующее:

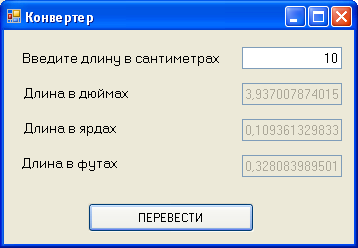
if (e.KeyCode == Keys.Enter) ExecuteButton\_Click(sender,e);

Теперь подсчет результата будет выполняться не только при нажатии на кнопку *ExecuteButton*, но и при нажатии на клавишу *Enter*. Убедитесь в этом, запустив приложение.

**Упражнение 4.2.** Дополним приложение «Конвертер» возможностью перевода сантиметров не только в дюймы, но еще и в футы и ярды.

**Решение**

**1‑й этап.** Дополним визуальный интерфейс приложения (рис. 4.3).



*Рис. 4.3.* Форма из упражнения 4.2

Поместите на форму еще два компонента Label и два компонента TextBox и измените их свойства в соответствии с таблицей:

|  |  |  |
| --- | --- | --- |
| Компонент | Свойство | Значение |
| *Label1* | *Name* | *YardLabel* |
| *Font*.*Size* | 10 |
| *Text* | Длина в ярдах |
| *Label2* | *Name* | *FootLabel* |
| *Font*.*Size* | 10 |
| *Text* | Длина в футах |
| *TextBox1* | *Name* | *YardTextBox* |
| *Font*.*Size* | 10 |
| *Enabled* | *False* |
| *Text* | 0 |
| *TextAlign* | *Right* |
| *TextBox2* | *Name* | *FootTextBox* |
| *Font*.*Size* | 10 |
| *Enabled* | *False* |
| *Text* | 0 |
| *TextAlign* | *Right* |

**2‑й этап.** Создание программного кода.

Добавьте в обработчик события Click кнопки *ExecuteButton* определение значения длины в ярдах и футах, используя следующие формулы: *1 фут = 12 дюймов, 1 ярд = 3 фута*.

Запустите и протестируйте приложение, убедитесь в правильности его работы.

Компонент **RadioButton**

Найдите элемент управления RadioButton(зависимый переключатель ****). Этот компонент чаще всего используется в группе и позволяет пользователю выбрать единственный вариант из нескольких предложенных. Щелчок по переключателю приводит к автоматическому освобождению ранее выбранного переключателя в группе. Для объединения компонентов RadioButton в группу используется компонент GroupBox ().

Элемент управления RadioButton может отображать текст, объекты *Image* или оба типа содержимого. Некоторые свойства компонента RadioButton перечислены в табл. 4.4.

*Таблица 4.4*

**Некоторые свойства элемента управления RadioButton**

|  |  |
| --- | --- |
| Свойство | Описание |
| *Appearance* | Получает или задает значение, определяющее внешний вид переключателя |
| *AutoCheck* | Получает или задает значение, показывающее, будет ли автоматически изменяться значение *Checked* и внешний вид элемента управления, когда он выбирается щелчком |
| *CheckAlign* | Получает или задает расположение выбранной части элемента управления RadioButton |
| *Checked* | Логическое свойство, показывающее, выбран ли данный элемент управления |
| *Image* | Задает изображение (растровое, пиктограмму и т. п.), которое будет отображаться в RadioButton |
| *Text* | Определяет строковую константу, отображаемую в компоненте |
| *TextAlign* | Получает или задает выравнивание текста в элементе управления |

При работе с элементом управления RadioButton чаще всего используются события:

* Click – происходит при щелчке мышью по кнопке, а также при нажатии клавиши *Enter* в случае, когда кнопка находится в фокусе;
* CheckedChanged – происходит при изменении значения свойства *Checked*.

**Упражнение 4.3.** Изменимвизуальный интерфейс приложения «Конвертер» таким образом, чтобы пользователь мог выбрать нужную ему единицу измерения. Для этого воспользуемся компонентом RadioButton.

**Решение**

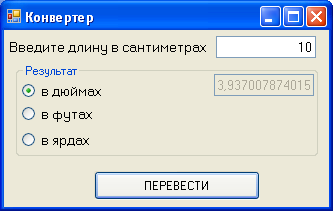
Создайте новый проект *Ex4* в решении *Lab4*.

**1‑й этап.** Создадим визуальный интерфейс приложения (рис. 4.4).

Измените значения свойств формы следующим образом:

|  |  |
| --- | --- |
| Свойство | Значение |
| *Name* | *MainForm* |
| *Text* | Конвертер |

Скопируйте из упражнения 4.2 (*Ex1*) с формы компоненты *CentiMLa­bel,* *CentiMTextBox* и *ExecuteButton* и вставьте их на новую форму.



*Рис. 4.4.* Форма из упражнения 4.3

Поместите на форму компонент GroupBox. Измените значение его свойств:

|  |  |
| --- | --- |
| Свойство | Значение |
| *Name* | *ResultGroupBox* |
| *Text* | Результат |

На компонент *ResultGroupBox* поместите окно редактирования Text­Box и три переключателя RadioButton. Измените их свойства в соответствии со следующей таблицей:

|  |  |  |
| --- | --- | --- |
| Компонент | Свойство | Значение |
| *TextBox* | *Name* | *ResultTextBox* |
| *Font*.*Size* | 10 |
| *Enabled* | *False* |
| *Text* | 0 |
| *TextAlign* | *Right* |
| *RadioButton1* | *Name* | *InchRadioButton* |
| *Font*.*Size* | 10 |
| *Text* | в дюймах |
| *RadioButton2* | *Name* | *FootRadioButton* |
| *Font*.*Size* | 10 |
| *Text* | в футах |
| *RadioButton3* | *Name* | *YardRadioButton* |
| *Font*.*Size* | 10 |
| *Text* | в ярдах |

**2‑й этап.** Создание программного кода.

Создайте обработчик события Click кнопки *ExecuteButton* и поместите туда следующий код:

try

{

if (InchRadioButton.Checked)

ResultTextBox.Text =

(double.Parse(CentiMTextBox.Text) / 2.54).ToString();

if (YardRadioButton.Checked)

ResultTextBox.Text =

(double.Parse(CentiMTextBox.Text) / 91.44).ToString();

if (FootRadioButton.Checked)

ResultTextBox.Text =

(double.Parse(CentiMTextBox.Text) / 30.48).ToString();

}

catch

{

MessageBox.Show("Некорректный ввод исходных данных", "Ошибка",

MessageBoxButtons.OK, MessageBoxIcon.Error);

CentiMTextBox.Text = "0";

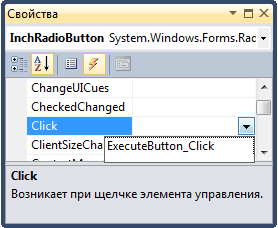
ResultTextBox.Text = "0";

}

Здесь результат преобразования определяется в зависимости от того, какой переключатель (единицу измерения) выбрал пользователь.

Запустите и протестируйте приложение, убедитесь в правильности его работы.

Модифицируем приложение так, чтобы при выборе единицы измерения результат преобразования выводился сразу, а не после щелчка по кнопке. Для этого в окне Свойства на вкладке событий для каждого из компонентов RadioButton в правом столбце напротив события CheckedChanged выберите в раскрывающемся списке обработчик события *ExecuteButton\_Click* (рис. 4.5).



*Рис. 4.5.* Окно Cвойств

Запустите и протестируйте приложение.

### Компонент Timer

На Панели элементов найдите компонент Timer (таймер). Щелк­ните по нему мышью, затем щелкните мышью на форме. Заметьте, что значок с именем этого компонента отображается на вкладке *Form1.cs[Конструктор]* не на самой форме, а на специальной панели, расположенной ниже формы, предназначенной для вывода некоторых компонентов, не имеющих графического отображения или имеющих сложную составную структуру.

Timer – это невизуальный компонент, позволяющий задавать в программе определенные интервалы времени. Он достаточно часто используется, например, для реализации эффекта движения, а также для отслеживания времени, потраченного пользователем на выполнение некоторых операций.

Для управления таймером используются два свойства:

* *Enabled* – получает или задает признак активности таймера;
* *Interval –*получает или задает время в миллисекундах до вызова события Tick относительно момента, когда событие Tick произошло последний раз.

### Единственное событие компонента Timer – событие Tick, которое происходит по истечении интервала времени, заданного в свойстве *Interval* таймера, при условии, что таймер активен.

**Упражнение 4.4.** Напишите приложение «Секундомер».

**Решение**

Создайте новый проект *Ex3* в решении *Lab4*.

**1‑й этап.** Создадим визуальный интерфейс приложения.

Измените значения свойств формы следующим образом:

|  |  |
| --- | --- |
| Свойство | Значение |
| *Name* | *MainForm* |
| *Text* | Секундомер |

Поместите на форму компонент Label и две кнопки Button. Измените их свойства следующим образом:

|  |  |  |
| --- | --- | --- |
| Компонент | Свойство | Значение |
| *Label1* | *Name* | *TimeLabel* |
| *AutoSize* | *false* |
| *Dock* | *top* |
| *Font*.*Size* | 36 |
| *Text* |  |
| *TextAlign* | *MiddleCenter* |
| *Button1* | *Name* | *StartButton* |
| *Font*.*Size* | 10 |
| *Text* | Старт |
| *Button2* | *Name* | *StopButton* |
| *Font*.*Size* | 10 |
| *Text* | Стоп |
| *Enabled* | *false* |

Поместите на форму компонент Timer. Установите значение свойства *Interval* 1000, а *Enabled* – *false*.

**2‑й этап.** Создание программного кода.

В классе формы опишите целочисленную переменную:

int Seconds = 0;

При нажатии на кнопку «Старт» на форме отображается количество секунд, прошедших с момента запуска секундомера. Создайте обработчик события Click кнопки *StartButton* и поместите туда следующие операторы:

//изменяем активность кнопок

StopButton.Enabled = true;

StartButton.Enabled = false;

Seconds = 0; //обнуляем счетчик секунд

timer1.Enabled = true; //запускаем Timer

//выводим значение секундомера в метке

TimeLabel.Text = Seconds.ToString();

Заметьте, что после нажатия кнопка «Старт» становится неактивной, в то время как кнопка «Стоп», наоборот, становится активной.

Создайте обработчик события Tick компонента Timer.

private void timer1\_Tick(object sender, EventArgs e)

{

Seconds++; //увеличиваем счетчик секунд

//отображаем изменения в метке

TimeLabel.Text = Seconds.ToString();

}

Создайте обработчик события Click кнопки *StopButton*.

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

{

//изменяем активность кнопок

StartButton.Enabled = true;

StopButton.Enabled = false;

timer1.Enabled = false; //останавливаем Timer

}

Запустите и протестируйте приложение, убедитесь в правильности его работы. Какие недостатки в работе приложения вы заметили?

**Упражнение 4.5.** Модифицируем приложение «Секундомер» таким об­разом, чтобы результат работы секундомера выводился в формате «часы:ми­нуты:секунды».

**Решение**

Далее будет отмечено, что нужно изменить в предыдущем упражнении. Если вы не хотите удалить первый вариант решения (упражнение 4.4), то при указании заменить операторы старые операторы не удаляйте, а заносите в комментарии.

В классе формы замените тип переменной *Seconds* на *DateTime* (эта структура была изучена в лабораторной работе № 2 данной части практикума).

В обработчике события Click кнопки *StartButton* оператор обнуления счетчика секунд замените на следующий оператор:

Seconds = new DateTime();

а оператор вывода значения секундомера замените на такой:

TimeLabel.Text = Seconds.ToLongTimeString();

Здесь мы создаем новый экземпляр структуры *DateTime* с помощью конструктора и выводим его значение в метке, используя метод *ToLongTimeString*().

Операторы в обработчике события Tick компонента *timer1* замените на следующие операторы:

//увеличиваем значение секундомера на 1 секунду

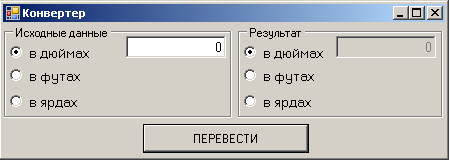
Seconds = Seconds.AddSeconds(1);

TimeLabel.Text = Seconds.ToLongTimeString(); //отображаем изменения

Запустите приложение и убедитесь в правильности его работы.

### Задания для самостоятельного выполнения

* 1. Модифицируйте приложение из упражнения 4.3 таким образом, чтобы пользователь имел возможность задать единицу измерения не только результата, но и исходных данных (рис. 4.6). Осуществите проверку корректности ввода. В любой момент работы программы в окно ответа должен выводиться верный ответ, или (в случае невозможности вычисления ответа) оно должно оставаться пустым. Добавьте следующие меры длины: сантиметр, метр.



*Рис. 4.6.* Форма из задания 4.1

* 1. Напишите программу для подсчета количества полных дней, оставшихся до конца года, если сейчас год *Y*, месяц *M*, день *D* (*Y*, *M* и *D* вводятся пользователем). Кнопка «Подсчитать» должна быть расположена в правом нижнем углу формы на расстоянии 10 пикселей от ее края, даже если размер формы изменяется. Осуществите проверку корректности ввода.
  2. Напишите приложение для определения идеального веса по следующей формуле:

*Вес*= *k*× (50 + 0,75 × (*h* – 150) + 0,25 × (*a* – 20)),

где

*k* – коэффициент (для мужчин *k* = 1, для женщин *k* = 0,9);

*h –*рост в см (корректными данными считаем 120<*h*<200);

*a* – возраст в годах (корректными данными считаем 15<*a*<80).

Для ввода *k* использовать RadioButton, остальные данные вводить в TextBox. При вводе некорректных данных вывести соответствующее сообщение.

* 1. Модифицируйте упражнение 3.2 «Клавиатурный тренажер» из лабораторной работы № 3 таким образом, чтобы пользователь мог задать уровень сложности тренажера (максимальное время отображения очередного символа строки задания на форме) – простой, средний или сложный. Каждый раз выдается по 10 символов. После окончания тестирования должно выводиться затраченное время, количество допущенных ошибок и количество пропущенных букв.
  2. Напишите программу нахождения корней квадратного уравнения *a × x*2 + *b*× *x*+ *c*= 0. Осуществите проверку корректности ввода. Учитите, что *a* может быть равно нулю.
  3. Создайте приложение с заставкой. Приложение должно состоять из двух форм. На первой форме появляется сообщение о создателе приложения и о том, что при двойном щелчке по форме появится заставка, а также пользователю предлагается задать некоторые параметры заставки:
* вид отображаемой фигуры (прямоугольники / эллипсы / прямоуголь­ни­ки и эллипсы в произвольном порядке),
* количество одновременно отображаемых фигур (одна / две),
* скорость смены картинки на заставке (медленная / средняя / быстрая).

Окно заставки темно‑синего цвета, без заголовка, занимает весь экран (изучите свойство формы *FormBorderStyle*, а также используйте свойство *WindowState*). В окне заставки в произвольном месте формы должны появляться фигуры, заданные пользователем в параметрах заставки. Приложение (обе формы) должно закрываться при нажатии любой клавиши клавиатуры, предварительно выдав сообщение, уточняющее, точно ли пользователь хочет закрыть приложение.

* 1. Напишите игровое приложение «Мышь за сыром». При запуске приложения в левом верхнем углу формы расположен сыр*,* а в правом нижнем – мышь (рис. 7.1). Через заданные интервалы времени сыр меняет свое местоположение произвольным образом. Если сыр выпадает на мышь – он автоматически съедается и перемещается на новую позицию. Пользователь с помощью клавиш‑стрелок на клавиатуре передвигает мышь, пытаясь поймать сыр. На игру отводится ровно одна минута. По истечении этого времени выводится диалоговое окно, в котором отображается информация о количестве кусочков сыра, которое пользователю удалось собрать, и предлагается сыграть еще раз. При нажатии на кнопку “*No*” форма закрывается, при нажатии на кнопку “*Yes*” игра возобновляется.

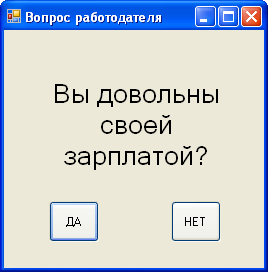


*Рис. 4.7.* Форма из задания 4.7

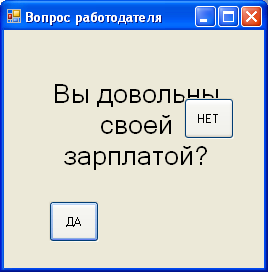
*Указания к решению:* для отображения графического объекта на форме используйте компонент PictureBox. Исследуйте его свойства и события с помощью справочной системы. Для простоты реализации установите одинаковые размеры компонентов PictureBox.

* 1. Напишите приложение «Вопрос работодателя», которое выполняет следующие действия:
* первоначально форма имеет следующий вид (рис. 4.8);
* кнопка «Да» имеет подсказку «Нажми сюда». При нажатии на кнопку «Да» приложение закрывается, выводя сообщение «Мы так и думали!»;
* при попытке нажатия на кнопку «Нет» эта кнопка перемещается по форме в произвольном направлении, не уходя за границы формы (рис. 4.9).

*Указания к решению*: для отображения всплывающей подсказки воспользуйтесь компонентом ToolTip и свойством *ToolTip on toolTip1* у соответствующего компонента Button.



*Рис. 4.8.* Первоначальный вид



*Рис. 4.9.* При движении мыши

* 1. Напишите приложение, изображающее произвольный треугольник ABC на форме (при нажатии на кнопку «Изобразить новый треугольник»). Далее пользователь задает произвольную точку внутри треугольника (кликом мыши) и вводит в TextBox количество шагов. После этого при нажатии на кнопку «Начать» программа выполняет определенное количество следующих шагов:

1. определяется случайное число от 1 до 3; 1 соответствует A, 2 – B, 3 – С;
2. на форме отображается половина отрезка от текущей точки до соответствующей вершины;

После выполнения определенного количества шагов можно снова задать количество шагов и продолжить такие же шаги от текущего положения точки.

При нажатии на кнопку «Изобразить новый треугольник» изображение очищается и на форме отображается новый произвольный треугольник.

***Баллы***: задачи 1–3, 5, 6, 8, 9 оцениваются по одному баллу, задачи 4 и 7 оцениваются в два балла.