

ное число можно получить, обращаясь к методу `next(x)`, где `x` задает полуинтервал $[0, x)$. Команда смены цвета будет выглядеть следующим образом:

```
BackColor := Color.FromArgb(255, rnd.next(256), rnd.next(256), rnd.next(256));
```

2 Создайте проект, в котором при двойном клике мыши по форме ее размеры будут увеличиваться на 5.

1. Создайте и сохраните в новой папке проект.
2. Измените свойство `Text` формы на Упражнение 2.
3. Создайте обработчик события мыши `DbClick`.
4. Для изменения ширины и высоты формы можно воспользоваться командами:

```
Width := Width + 5;
Height := Height + 5;
```

5. Сохраните изменения в проекте.
6. Запустите проект и проверьте его работу.

3 Создайте проект, в котором цвет формы будет меняться при наведении на нее мыши, например с желтого на зеленый.

1. Измените свойство `Text` у формы на Упражнение 3.
2. Установите желтый цвет формы.
3. Создайте обработчики для двух событий мыши: `MouseEnter` и `MouseLeave`.
4. В коде события `MouseEnter` установите зеленый (`Green`) цвет формы, а коде события `MouseLeave` — желтый (`Yellow`).
5. Сохраните изменения в проекте.
6. Запустите проект и проверьте его работу.

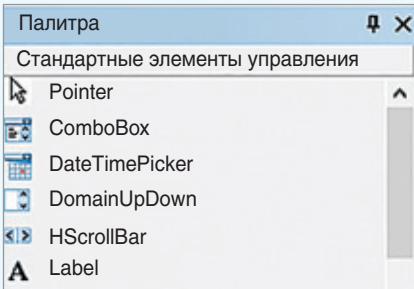
§ 3. Проектирование интерфейса оконного приложения с использованием элементов управления

3.1. Основные элементы управления

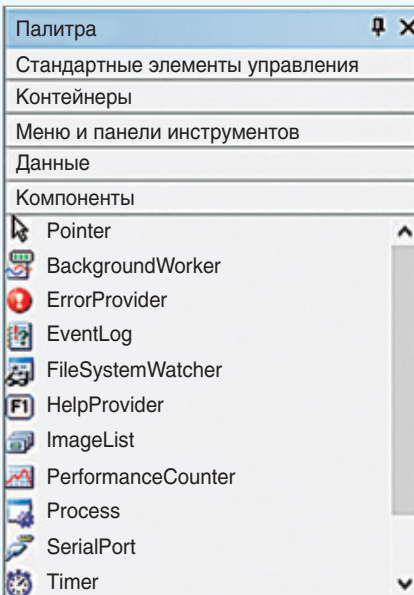
Элементами управления называются объекты, которые используются для отображения данных или организации взаимодействия между пользователем и приложением с помощью мыши или клавиатуры. Они собраны в специальные библиотеки компонентов, которые ОС использует для обеспечения единообразного интерфейса

Для элементов управления используется и другое название — виджеты. Слово употребляется примерно с 1920-х гг. в американском английском для обозначения простой, но необходимой вещи, маленького изделия. Одним из вариантов происхождения этого слова считается словослияние «`window gadget`» (букв. «оконное приспособление»), также произошедшее в начале XX в.

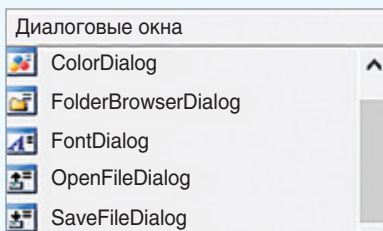
Пример 3.1. Палитра **Стандартные элементы управления**:



Пример 3.2. Палитра **Компоненты**:



Пример 3.3. Палитра **Диалоговые окна**:



прикладных программ. Наиболее распространенными элементами управления являются: кнопки, редактируемые поля, списки выбора, флажки, переключатели и т. д.

Компоненты библиотеки размещаются на различных страницах палитры компонентов. Каждая страница имеет свое название. На странице **Стандартные элементы управления** (пример 3.1) размещены наиболее употребляемые компоненты:

- Кнопка (Button)
- Надпись (Label)
- Поле для ввода текста (TextBox)
- Флажок (CheckBox)
- Радиокнопка (RadioButton)
- Список (ListBox)
- Выпадающий список (ComboBox)
- Рисунок (PictureBox)

Внешний вид этих компонентов на форме был представлен в примере 1.4.

Другие страницы палитры компонентов показаны в примерах 3.2 и 3.3.


Одним из наиболее используемых компонентов палитры **Компоненты** является компонент **Таймер** (Timer).

Палитра **Меню и панели инструментов** содержит компоненты, необходимые для создания главного меню программы или контекстных меню для различных объектов, помещенных на форму.

Палитры **Печать** и **Диалоговые окна** содержат компоненты, обеспечивающие стандартные диалоги операционной системы: открытие и сохранение файла, выбор цвета, установки параметров шрифта, настройки принтера и управление печатью.


Палитра **Данные** содержит компоненты для работы с таблицами баз данных.

3.2. Элемент управления кнопка (Button)

Компонент **кнопка** относится к элементам управления. На панели компонентов **Стандартные элементы управления** кнопка изображена в виде  Button, имя объекта — button. Кнопка, помещенная на форму, получает имя buttonN, где N — номер 1, 2, 3... (пример 3.4). При необходимости кнопку можно переместить в любое место формы. Ключевые точки позволят установить нужный размер кнопки.

Некоторые свойства кнопки перечислены в таблице (пример 3.5).


Как видно из таблицы, многие свойства кнопки совпадают по именам и назначениям со свойствами формы, поэтому в дальнейшем для компонентов будут указываться только те свойства, которые отличны от уже описанных для других компонентов.

Основным событием кнопки является Click. Для создания обработчика события Click для кнопки можно поступить так же, как и при создании аналогичного обработчика для формы: выбрать событие на вкладке **События**  и выполнить двойной щелчок в поле напротив события Click. Можно просто выполнить двойной щелчок по кнопке. (Для формы основным событием является событие Load, поэтому при двойном щелчке по форме создается обработчик события Load.)

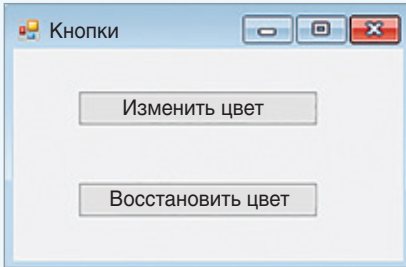
Пример 3.4. Компонент *кнопка* на форме:



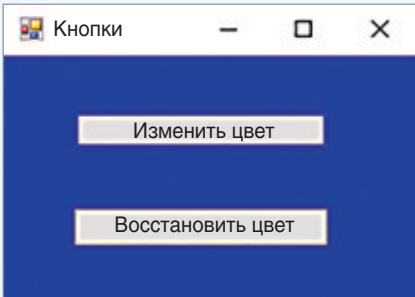
Пример 3.5. Свойства кнопки:

Некоторые свойства кнопки Button	
Свойство	Назначение
Text	Заголовок (внешнее имя) кнопки, текст, который отображается на кнопке. По умолчанию — button1
Font	Свойства шрифта для подписи заголовка. Свойство Font является сложным, о чем свидетельствует значок  , при нажатии на который раскрываются все свойства шрифта
Height	Высота кнопки
Weight	Ширина кнопки
Left	Горизонтальная координата положения верхнего левого угла кнопки на форме
Top	Вертикальная координата положения верхнего левого угла кнопки на форме
(Name)	Имя (внутреннее) кнопки. Используется в программном коде для обращения к объекту. Является идентификатором
Enabled	Значение True этого свойства обеспечивает доступность кнопки для мыши или клавиатуры
Visible	Значение True этого свойства обеспечивает видимость кнопки во время выполнения приложения


Пример 3.6. Внешний вид формы в режиме конструктора дизайна:



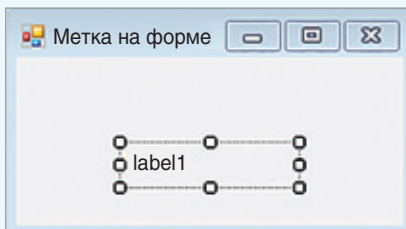
Внешний вид формы при выполнении:



Свойство кнопки `BackColor` позволяет менять ее цвет. Для обращения к этому свойству в программе используются запись: `button1.BackColor`.

Свойство кнопки `BackgroundImage` позволяет вставить на кнопку изображение, хранящееся в графическом файле. Для вставки можно использовать кнопку . Далее выбрать файл с рисунком.

Пример 3.7. Компонент *метка* на форме:



Пример 3.6. Создать проект, разместив на форме две кнопки. При нажатии на одну из них цвет формы должен измениться на синий, а при нажатии на вторую — должен восстановиться исходный цвет.

Этапы выполнения задания


1. Создать на форме две кнопки.
2. Изменить свойство `Text` у кнопки `button1` на **Изменить цвет**.
3. Изменить свойство `Text` у кнопки `button2` на **Восстановить цвет**.
4. Создать обработчик события `Click` для кнопки `button1` и изменить цвет формы. Команда

```
BackColor := Color.Blue;
```

5. Создать обработчик события `Click` для кнопки `button2` и изменить цвет формы на первоначальный (название цвета формы указано в поле `Color` инспектора объектов). Команда `BackColor := SystemColors.Control;`

6. Сохранить изменения в проекте. Название цвета `SystemColors.Control` задает не какой-то определенный цвет. Это цвет элемента управления, заданный цветовой схемой `Windows`. Поэтому он не обязательно будет серым.


3.3. Элемент управления *метка* (Label)

Компонент *метка* предназначен для отображения текста на форме. На панели компонентов **Стандартные элементы управления** метка изображена в виде , имя объекта — `label`. Кнопка, помещенная на форму, получает имя `labelN`, где `N` — номер 1, 2, 3... (пример 3.7).

Некоторые свойства метки, отличные от свойств кнопки, перечислены в таблице (пример 3.8). Основным событием для метки является Click.


Пример 3.9. Создать проект, в котором описана возможность выполнять следующие действия: после запуска программы в окне с именем «Работаем с кнопкой и меткой» при щелчке мыши по кнопке «Приветствие» появляется сообщение «Здравствуй, мир!».

Этапы выполнения задания


1. Изменить свойство Text формы на «Работаем с кнопкой и меткой».
2. Добавить на форму кнопку button1.
3. Изменить свойство Text кнопки на «Приветствие».
4. Добавить на форму метку label1.
5. Изменить свойства шрифта для компонента label1. Нажать кнопку  в поле Font (цвет шрифта — синий, размер — 20, стиль — жирный курсив).
6. Очистить поле Text у метки.
7. Установить значение true у свойства метки Autosize.

8. В обработчик события Click для кнопки button1 вписать команду `label1.Text := 'Здравствуй, мир!';`

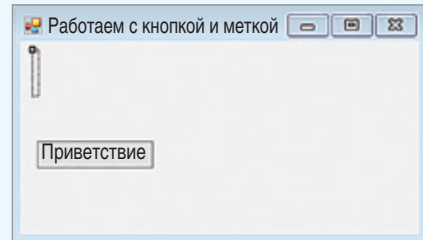
3.4. Элемент управления *текстовое поле (Edit)*

Текстовое поле — компонент, который предназначен для ввода и вывода текстовой информации. На панели компонентов **Стандартные элементы управления** текстовое поле изображено в виде  `abl TextBox`, имя объекта — `TextBox`.

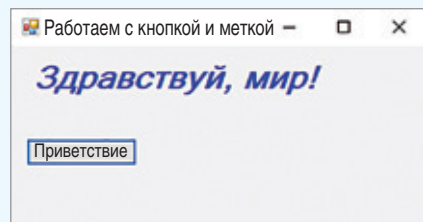
Пример 3.8. Свойства метки:

Свойство	Назначение
Text	Отображает введенный текст на форме
BackColor	Устанавливает цвет фона метки, который по умолчанию совпадает с цветом формы. Фон метки можно сделать прозрачным, задав свойству BackColor значение <code>Color.Transparent</code>
AutoSize	Значение true этого свойства приводит к автоматическому изменению размеров метки в соответствии с длиной текста
TextAlign	Выравнивание текста относительно границ метки: 

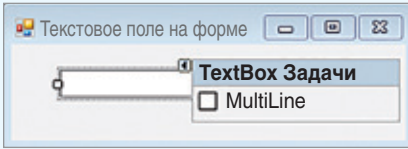
Пример 3.9. Форма на этапе конструирования:



Работающее приложение:



Пример 3.10. Компонент *текстовое поле* на форме:



Пример 3.11. Свойства текстового поля:

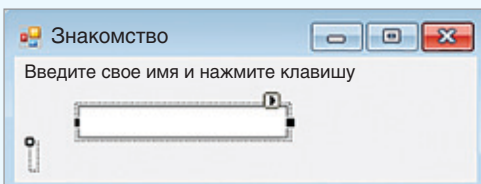
Свойство	Назначение
BorderStyle	Определяет границу вокруг текстового поля: None (нет границы), FixedSingl или Fixed3d (есть граница)
MaxLength	Ограничивает количество символов, которые можно ввести в TextBox
ReadOnly	Значение true запрещает редактирование текста, отображаемого в TextBox
Text	Содержит вводимый или выводимый текст

Текстовое поле часто называют текстовым редактором, поскольку оно снабжено такими функциями, как:

- копирование выделенного текста в буфер обмена (Ctrl+C);
- вырезание выделенного текста в буфер обмена (Ctrl+X);
- вставка текста из буфера обмена в позицию курсора (Ctrl+V);
- отмена последней команды редактирования (Ctrl+Z).

Свойство Multiline определяет, каким будет редактор — однострочным или многострочным.

Пример 3.12. Форма на этапе конструирования:



Текстовое поле, помещенное на форму, получает имя TextBoxN, где N — номер 1, 2, 3... (пример 3.10).

В отличие от ранее рассмотренных компонентов, свойство Text у текстового поля по умолчанию пусто (у других — совпадает с именем компонента). Некоторые свойства компонента TextBox приведены в таблице (пример 3.11).

Значение свойства Text компонента *текстовое поле* может изменяться программно или при вводе с клавиатуры. Основным событием для TextBox является TextChanged, которое происходит при изменении компонента. Наиболее часто программируют событие KeyPress, которое позволяет определить, какая клавиша была нажата.

Пример 3.12. Создать проект, в котором пользователя попросят ввести его имя, а затем, после нажатия клавиши Enter, будет выдано сообщение «Имя, приятно с Вами познакомиться!»

Этапы выполнения задания

1. Изменить свойство Text у формы на «Знакомство».
2. Разместить на форме две метки и текстовое поле.
3. Изменить свойство Text у label1 на «Введите свое имя и нажмите клавишу Enter».
4. Очистить поле свойства Text у Label2.
5. Написать обработчик события KeyPress для компонента Edit1, который будет проверять нажатие клавиши ввода (код клавиши Enter — 13). Если клавиша нажата, то поменять свойство Text у label2:

```
if e.KeyChar = #13 then
    label2.Text := TextBox1.Text +
    ', приятно с Вами познакомиться!';
```

Текстовое поле `TextBox` используется также для ввода и вывода чисел. При этом необходимо использовать функции для преобразования строк в числа и чисел в строки. Эти функции приведены в таблице:

Название функции	Действие
Ввод с помощью Edit	
StrToInt	Преобразование строки в целое число
StrToFloat	Преобразование строки в значение с плавающей запятой
Вывод с помощью Edit	
IntToStr	Преобразование целого числа в строку
FloatToStr	Преобразование вещественного числа в строку

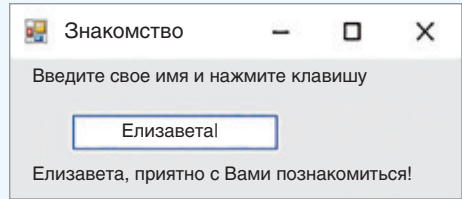
В русскоязычной версии Windows в качестве разделителя целой и дробной части числа по умолчанию используется запятая. Если при вводе чисел в текстовые поля использовать точку, то будет возникать ошибка преобразования типов.

Пример 3.13. Создать проект, в котором пользователь сможет ввести число, получить его значение в квадрате и квадратный корень из этого числа.

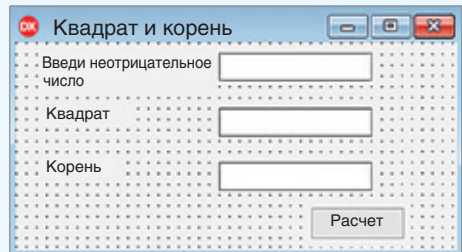
Этапы выполнения задания

1. Изменить свойство `Text` у формы на «Квадрат и корень».
2. Разместить на форме три метки, три текстовых поля и кнопку.

Пример 3.12. Продолжение.
Работающее приложение:



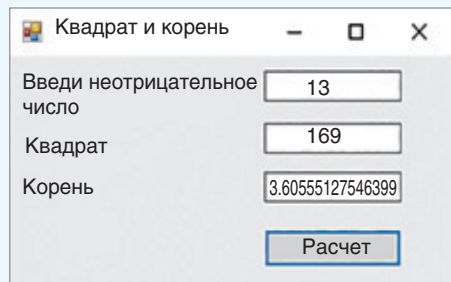
Пример 3.13. Форма на этапе конструирования:



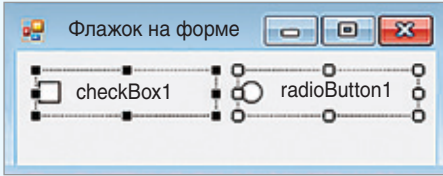
Обработчик события `OnClick` для `Button1`:

```
procedure Form1.button1_Click
(sender: Object; e: EventArgs);
var a, b: integer;
    c: real;
begin
    a := StrToInt(TextBox1.Text);
    b := a * a;
    c := sqrt(a);
    TextBox2.Text := IntToStr(b);
    TextBox3.Text := FloatToStr(c);
end;
```

Работающее приложение:



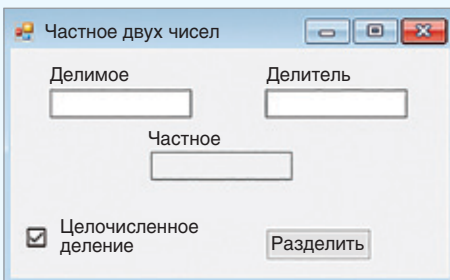
Пример 3.14. Компоненты флажок и радиокнопка на форме:



Пример 3.15. Свойства компонента флажок:

Свойство	Назначение
Checked	Значение true у этого свойства показывает, что установлена галочка — <input checked="" type="checkbox"/> , пустое окно индикатора — <input type="checkbox"/> соответствует значению false
Text	Надпись возле компонента checkBox
RightToLeft	Определяет, с какой стороны кнопки размещается надпись: Yes (надпись слева), No (надпись справа)
CheckState	Определяет состояние компонента: Unchecked — не выделен, Checked — выделен, Indeterminate (<input type="checkbox"/>) — промежуточное состояние. Первые два состояния соответствуют свойству Checked

Пример 3.16. Форма на этапе конструирования:



3. Изменить свойство Text у Label1 на «Введите неотрицательное число».
4. Изменить свойство Text у Label2 на «Квадрат».
5. Изменить свойство Text у Label3 на «Корень».
6. Изменить свойство Text у Button1 на «Расчет».
7. Написать обработчик Click для кнопки.

3.5*. Элементы управления флажок (CheckBox) и переключатель (RadioButton)

Флажок используется в приложениях для включения или выключения каких-либо опций. На панели компонентов **Стандартные элементы управления** флажок изображен в виде CheckBox, имя объекта — CheckBox. Флажок, помещенный на форму, получает имя checkBoxN, где N — номер 1, 2, 3... (пример 3.14). Некоторые свойства компонента checkBox приведены в таблице (пример 3.15).

Аналогичным образом используется компонент **Переключатель (радиокнопка)**. На панели компонентов Standard радиокнопка изображена в виде RadioButton, имя объекта — RadioButton. Переключатель, помещенный на форму, получает имя radioButtonN, где N — номер 1, 2, 3... (пример 3.14).

Обычно радиокнопки образуют группы взаимосвязанных индикаторов, позволяющих выбрать только одну из нескольких взаимоисключающих альтернатив. При размещении на форме нескольких переключате-

лей включенным должен быть только один из них (контейнер GroupBox).

Пример 3.16. Создать проект для вычисления частного от деления одного целого числа на другое. Числа задаются в текстовых полях. Результат вычисляется при нажатии на кнопку «Частное» и помещается в третье текстовое поле. Результат зависит от состояния флажка.

Этапы выполнения задания

1. Поместить на форму текстовые поля (3), надписи (3), флажок и кнопку.

2. Для компонента textBox3 установить значение true для свойства ReadOnly.

3. Изменить свойство Text у компонентов label («Делимое», «Делитель», «Частное»).

4. Изменить свойство Text компонента button1 на «Разделить».

5. Изменить свойство Text компонента checkBox1 на «Целочисленное деление».

6. Написать обработчик события Click для компонента button1.

6.1. Проверить, что поля компонентов textBox1 и textBox2 не пусты. Иначе вывести сообщение «Одно из полей не заполнено».

6.2. Проверить состояние переключателя checkBox. Если он включен, то выполнить целочисленное деление, иначе обычное деление.

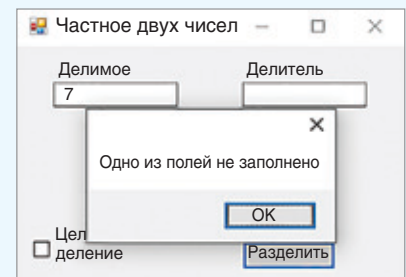
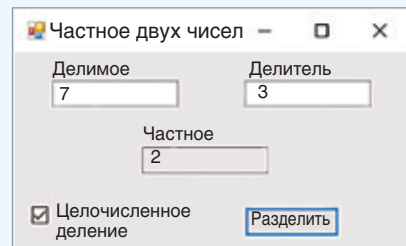
6.3. Вывести результат.

7. Выполнить программу для различных значений. Проверить работу приложения, когда одно из полей textBox1 или textBox2 (или оба поля) пусты.

Обработчик события OnClick для Button1:

```
procedure Form1.button1_Click
(sender: Object; e: EventArgs);
var a, b, c : integer;
    d : real;
begin
  if (TextBox1.Text <> '') and
    (TextBox2.Text <> '') then
  begin
    a := StrToInt(TextBox1.Text);
    b := StrToInt(TextBox2.Text);
    if CheckBox1.checked then
    begin
      c := a div b;
      TextBox3.Text := IntToStr(c);
    end
    else
    begin
      d := a / b;
      TextBox3.Text := FloatToStr(d);
    end;
  end
  else
  begin
    MessageBox.Show('Одно из
    полей не заполнено!');
  end;
end;
```

Работающее приложение:





1. Какие компоненты относят к элементам управления?
2. Как поместить компонент на форму?
3. Какие свойства компонента button вы можете назвать?
4. Какое событие является основным для компонента button?
5. Для чего предназначен компонент label?
6. В каких случаях используется компонент TextBox?
7. Для чего предназначены компоненты checkBox и radioButton?



Упражнения

1 Откройте проект из примера 3.9 и дополните его кнопкой «Очистить»¹. Кнопка «Очистить» должна очищать текст метки (Свойству Caption присвоить значение пустой строки: ""). Сделайте случайным выбор цвета и размера шрифта у метки.

2 Откройте проект из примера 3.12 и добавьте на форму три метки и две кнопки.

1. Измените свойства компонентов в соответствии с указаниями в таблице.

Компонент	Свойство	Значение свойства
button1	Text	Да
button1	Visible	False
button2	Text	Нет
button2	Visible	False
label3	Text	Вы хотите работать в ИТ?
label3	Visible	False
label4	Text	Замечательно! Успехов в изучении информатики! Она Вам понадобится!
label4	Visible	False
label5	Text	Другие профессии тоже требуют знания информатики
label5	Visible	False

2. Добавьте в обработчик события KeyPress команду, которая делает надпись Label3 и кнопки видимыми.

¹ Перед изменением скопируйте проект в новую папку

```

procedure Form1.textBox1_KeyPress(sender: Object;
e: KeyPressEventArgs);
begin
    if e.KeyChar = #13 then
        begin
            label2.Text := TextBox1.Text + ', приятно с Вами познакомиться!';
            Label3.Visible := True;
            Button1.Visible := True;
            Button2.Visible := True;
        end;
    end;
end;

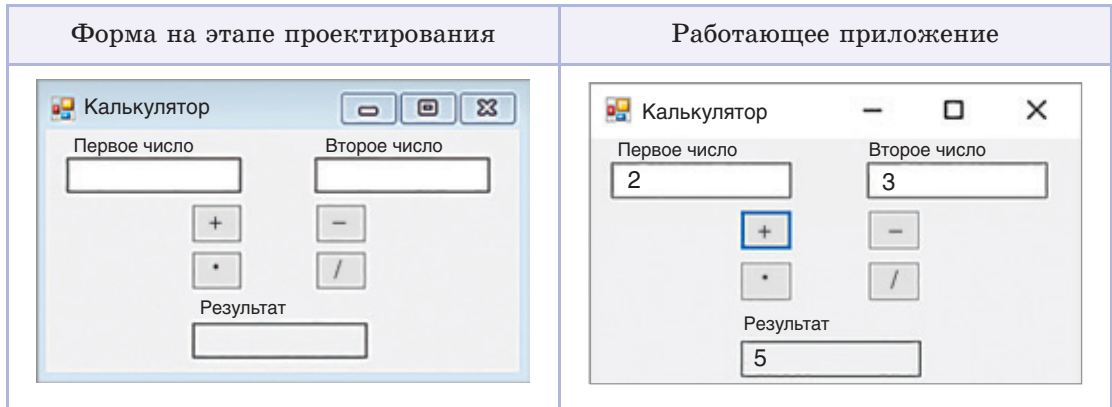
```

3. Напишите обработчики Click для кнопок Button1 и Button2. Сделайте видимыми соответствующие надписи.

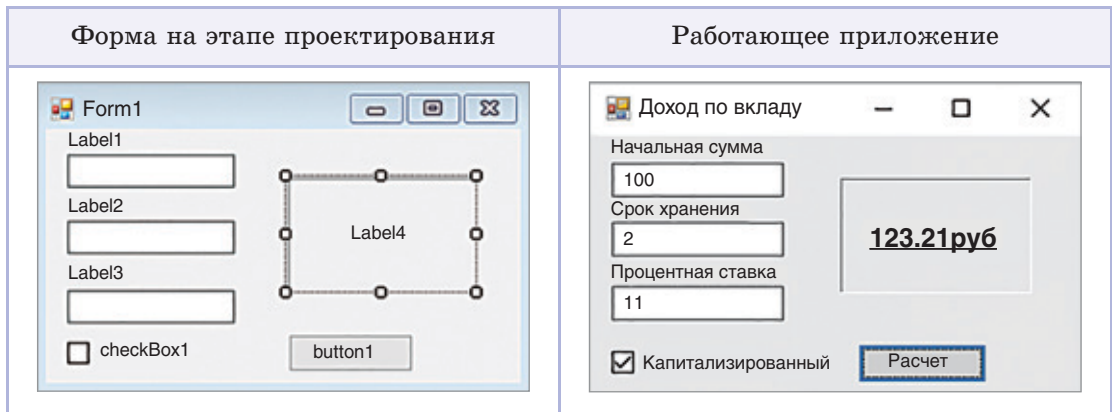
Форма на этапе проектирования	Работающее приложение после запуска
Работающее приложение до ответа на вопрос	Работающее приложение после ответа на вопрос

4*. Добавьте в приложение еще один вопрос. Форму ответа выберите самостоятельно.

3 Создайте проект **Калькулятор**. Разместите на форме три поля `TextBox` и три надписи: «Первое число», «Второе число», «Результат». Добавьте кнопки для вычисления суммы, разности, произведения и частного. Запретите редактирование в поле с ответом. *Добавьте проверку деления на нуль.



4 Создайте проект, в котором вычисляется доход по вкладу. Программа должна обеспечивать расчет денежных сумм для простых или капитализированных вкладов. Если вклад простой, то процентная ставка начисляется от исходной суммы, и каждый месяц она одинаковая, а если капитализированный, то процентная ставка начисляется каждый месяц от суммы вклада в предыдущем месяце.



Проверьте, заполнены ли поля с исходными данными. Если нет, то выведите соответствующее сообщение.

5 Реализовать «убегающую кнопку», т. е. при наведении указателя мыши на кнопку она должна случайным образом поменять место.

6 Добавить в упражнение 5 кнопку «Домой», которая должна передвинуть «убегающую» в верхний левый угол формы.