

Никита Культин

Microsoft®

Visual C++

В задачах и примерах

Санкт-Петербург

«БХВ-Петербург»

2010

УДК 681.3.068+800.92VisualC++
ББК 32.973.26-018.1
К90

Культин Н. Б.

К90 Microsoft Visual C++ в задачах и примерах. — СПб.:
БХВ-Петербург, 2010. — 272 с.: ил. + CD-ROM

ISBN 978-5-9775-0458-4

Книга представляет собой сборник программ и задач для самостоятельного решения. Примеры различной степени сложности — от простейших до приложений работы с графикой и базами данных Microsoft Access и Microsoft SQL Server Compact Edition — демонстрируют назначение базовых компонентов, раскрывают тонкости разработки приложений Windows Forms в Microsoft Visual C++. Справочник, входящий в книгу, содержит описание базовых компонентов, событий, исключений и наиболее часто используемых функций.

На прилагаемом компакт-диске находятся проекты, представленные в книге.

Для начинающих программистов

УДК 681.3.068+800.92VisualC++
ББК 32.973.26-018.1

Группа подготовки издания:

Главный редактор	<i>Екатерина Кондукова</i>
Зам. главного редактора	<i>Игорь Шишигин</i>
Зав. редакцией	<i>Григорий Добин</i>
Редактор	<i>Елена Кашлакова</i>
Компьютерная верстка	<i>Натальи Караваевой</i>
Корректор	<i>Виктория Пиотровская</i>
Дизайн серии	<i>Инны Тачиной</i>
Оформление обложки	<i>Елены Беляевой</i>
Зав. производством	<i>Николай Тверских</i>

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 28.09.09.

Формат 60×90¹/₁₆. Печать офсетная. Усл. печ. л. 17.

Тираж 2000 экз. Заказ №

"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.60.953.Д.005770.05.09 от 26.05.2009 г. выдано Федеральной службой
по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 978-5-9775-0458-4

© Культин Н. Б., 2009
© Оформление, издательство "БХВ-Петербург", 2009

Оглавление

Предисловие	1
Описание компакт-диска	2
ЧАСТЬ I. ПРИМЕРЫ И ЗАДАЧИ	3
Базовые компоненты	5
Общие замечания	5
Мили-километры.....	6
Фунты-килограммы	9
Конвертор	12
Фотоателье.....	15
Комплектация	18
Жалюзи.....	21
Калькулятор.....	24
Просмотр иллюстраций.....	30
Слайд-шоу.....	36
Финансовый калькулятор.....	42
ОСАГО	47
Секундомер.....	53
Таймер.....	56
Обработка исключения	60
Справочная информация	61
Операции с файлами	65
Курс	65
Котировки	69
Редактор текста	71
Графика.....	81
Общие замечания	81
Прямоугольники.....	82

Вывод текста.....	84
Диаграмма.....	87
График.....	92
Круговая диаграмма.....	97
Кисти	103
Бегущая строка.....	106
Часы.....	109
Полет	114
Базы данных	118
Общие замечания	118
Контакты.....	118
Контакты-2.....	124
Контакты-3.....	127
Ежедневник.....	138
SQL Server Compact Edition	148
Игры и другие полезные программы.....	158
Парные картинки	158
Собери картинку	170
Сапер	178
Будильник	190
Экзаменатор.....	196
Задачи для самостоятельного решения	209

ЧАСТЬ II. КРАТКИЙ СПРАВОЧНИК.....215

Форма.....	217
Компоненты	219
<i>Button</i>	219
<i>ComboBox</i>	221
<i>ContextMenuStrip</i>	222
<i>CheckBox</i>	222
<i>CheckedListBox</i>	224
<i>GroupBox</i>	225
<i>ImageList</i>	226
<i>Label</i>	227
<i>ListBox</i>	228
<i>MenuStrip</i>	229
<i>NotifyIcon</i>	230

<i>NumericUpDown</i>	230
<i>OpenFileDialog</i>	231
<i>Panel</i>	232
<i>PictureBox</i>	233
<i>RadioButton</i>	234
<i>ProgressBar</i>	236
<i>SaveFileDialog</i>	236
<i>TextBox</i>	238
<i>ToolTip</i>	239
<i>Timer</i>	240
Графика.....	240
Графические примитивы.....	240
Карандаш	243
Кисть	244
Типы данных.....	247
Целый тип.....	247
Вещественный тип.....	247
Символьный и строковый типы.....	247
Функции.....	248
Функции преобразования.....	248
Функции манипулирования строками.....	249
Функции манипулирования датами и временем.....	251
Функции манипулирования каталогами и файлами.....	253
Математические функции.....	256
События.....	257
Исключения.....	258
Предметный указатель.....	261

Предисловие

В последнее время в общем объеме вновь создаваемого программного обеспечения различного назначения увеличивается доля .NET-приложений, программ, ориентированных на платформу .NET. Это объясняется, прежде всего, новыми возможностями, которые предоставляет платформа прикладным программам, а также тем, что технология .NET поддерживается новейшими операционными системами.

Microsoft .NET — это технология, основанная на идее универсального программного кода, который может быть выполнен любым компьютером, вне зависимости от используемой операционной системы. Универсальность программного кода обеспечивается за счет предварительной (выполняемой на этапе разработки) компиляции исходной программы в *универсальный промежуточный код* (CIL-код, Common Intermediate Language), который во время загрузки (запуска) программы транслируется в *выполняемый*. Преобразование промежуточного кода в выполняемый производит JIT-компилятор (от Just In Time — в тот же момент, "на лету"), являющийся элементом виртуальной выполняющей системы (Virtual Execution System, VES). Выполнение .NET-приложений в операционной системе Windows обеспечивает Microsoft .NET Framework.

Чтобы понять, что такое .NET, какие возможности она предоставляет программисту, необходимо опробовать ее в деле. Для этого нужно изучить среду и технологию разработки, понять назначение и возможности компонентов, их свойства и методы. И здесь хорошим подспорьем могут стать примеры — программы, разработанные другими программистами.

Среда разработки Microsoft Visual C++ является инструментом разработки приложений различного типа для Windows. В ней интегрированы удобный дизайнер форм, специализированный редактор кода, отладчик и другие полезные инструменты.

Книга, которую вы держите в руках, посвящена практике программирования в Microsoft Visual C++, разработке Windows Forms-приложений. В ней собраны разнообразные примеры, которые демонстрируют назначение базовых компонентов, технологии работы с файлами, графикой и базами данных.

Состоит книга из двух частей. Первая часть содержит примеры программ и задачи для самостоятельного решения. Примеры представлены в виде краткого описания, диалоговых окон и хорошо документированных текстов программ.

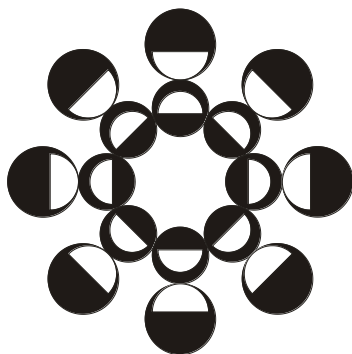
Вторая часть книги — это краткий справочник. В нем можно найти описание базовых компонентов и наиболее часто используемых функций.

Научиться программировать можно, только программируя, решая конкретные задачи. Поэтому, чтобы получить максимальную пользу от книги, вы должны работать с ней активно. Изучайте листинги, старайтесь понять, как работают программы. Не бойтесь экспериментировать — совершенствуйте программы, вносите в них изменения. Чем больше вы сделаете самостоятельно, тем большему научитесь!

Описание компакт-диска

Прилагаемый компакт-диск содержит проекты, приведенные в книге. Каждый проект находится в отдельном каталоге. Для того чтобы увидеть, как работает приложение, загрузите проект в Microsoft Visual C++, откомпилируйте его и затем запустите. Следует обратить внимание, что некоторые программы (например, работа с базами данных) требуют, чтобы на компьютере был установлен соответствующий сервер баз данных.

Для активной работы, чтобы иметь возможность вносить изменения в программы, скопируйте каталоги проектов на жесткий диск компьютера, в папку проектов Microsoft Visual Studio.



ЧАСТЬ I

Примеры и задачи

Базовые компоненты

В этом разделе приведены примеры, демонстрирующие назначение и технологию работы с базовыми компонентами.

Общие замечания

- ❑ Процесс создания программы состоит из двух шагов: сначала создается форма, затем — функции обработки *событий*.
- ❑ Форма создается путем помещения в нее необходимых компонентов и последующей их настройки.
- ❑ В форме практически любого приложения есть компоненты, обеспечивающие взаимодействие программы с пользователем. Такие компоненты называют базовыми.
- ❑ К базовым компонентам можно отнести:
 - Label — поле отображения информации;
 - TextBox — поле ввода-редактирования текста (данных);
 - Button — командная кнопка;
 - CheckBox — флажок;
 - RadioButton — радио-кнопка;
 - ListBox — список выбора;
 - ComboBox — поле редактирования со списком выбора.
- ❑ Вид компонента и его поведение определяют значения *свойств* (характеристик) компонента (описание свойств базовых компонентов можно найти в справочнике, во второй части книги).
- ❑ Основную работу в программе выполняют функции обработки событий (описание основных событий можно найти в справочнике, во второй части книги).

- ❑ Исходную информацию программа может получить из поля редактирования (компонент `TextBox`), списка (компонент `ListBox`), комбинированного списка (компонент `ComboBox`).
- ❑ Для ввода значений логического типа можно использовать компоненты `CheckBox` и `RadioButton`.
- ❑ Результат работы программы можно вывести в поле отображения текста (компонент `Label`), в поле редактирования или в окно сообщения (метод `MessageBox::Show()`).
- ❑ Для преобразования строки в целое число нужно использовать функцию `Convert.ToInt32()`, в дробное число — `Convert.ToDouble()`.
- ❑ Для преобразования численного значения в строку нужно использовать метод `ToString()`. В качестве параметра метода можно указать формат отображения: "c" — денежный с разделителями групп разрядов и обозначением валюты (`currency`); "N" — числовой с разделителями групп разрядов (`numeric`); "F" — числовой без разделителей групп разрядов (`fixed`).

Мили-километры

Программа **Мили-километры**, ее форма приведена на рис. 1.1, демонстрирует использование компонентов `TextBox` и `Label` для ввода исходных данных и отображения результата. Программа спроектирована таким образом, что в поле редактирования пользователь может ввести только правильные данные — дробное число. Значения свойств формы приведены в табл. 1.1.

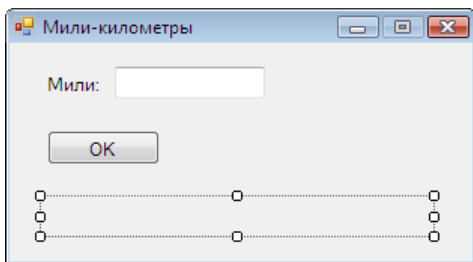


Рис. 1.1. Форма программы **Мили-километры**

Таблица 1.1. Значения свойств формы

Свойство	Значение	Комментарий
Text	Мили-километры	Текст заголовка
StartPosition	CenterScreen	Начальное положение окна — в центре экрана
FormBorderStyle	FixedSingle	Тонкая граница окна. Пользователь не сможет изменить размер окна путем перемещения его границы
MaximizeBox	False	Кнопка Развернуть окно недоступна. Пользователь не сможет развернуть окно программы на весь экран
Font	Tahoma; 9pt	Шрифт, наследуемый компонентами формы

```
// щелчок на кнопке ОК
```

```
private: System::Void button1_Click(System::Object^ sender,
                                     System::EventArgs^ e)
```

```
{
```

```
    double mile; // расстояние в милях
```

```
    double km; // расстояние в километрах
```

```
    // Если в поле редактирования нет данных,
```

```
    // то при попытке преобразовать пустую
```

```
    // строку в число возникает исключение.
```

```
    try
```

```
    {
```

```
        mile = Convert::ToDouble(textBox1->Text);
```

```
        km = mile * 1.609344;
```

```
        label2->Text = mile.ToString("n") + " miles - " +
                       km.ToString("n") + " км";
```

```

}

catch (System::FormatException^ ex )
{
    // обработка исключения:
    // - сообщение
    MessageBox::Show(
        "Надо ввести исходные данные", "Мили-километры",
        MessageBoxButtons::OK,
        MessageBoxIcon::Exclamation);

    // - установить курсор в поле редактирования
    textBox1->Focus();
}
}

```

// нажатие клавиши в поле textBox

```

private: System::Void textBox1_KeyPress(System::Object^
sender, System::Windows::Forms::KeyPressEventArgs^ e)
{

```

```

    // Правильными символами считаются цифры,
    // запятая, <Enter> и <Backspace>.
    // Будем считать правильным символом
    // также точку, но заменим ее запятой.
    // Остальные символы запрещены.
    // Чтобы запрещенный символ не отображался
    // в поле редактирования, присвоим
    // значение true свойству Handled параметра e

```

```

if ((e->KeyChar >= '0') && (e->KeyChar <= '9'))
{
    // цифра
    return;
}

```

```

if (e->KeyChar == '.')

```

```
{
    // точку заменим запятой
    e->KeyChar = ',';
}

if (e->KeyChar == ',')
{
    if (textBox1->Text->IndexOf(',') != -1)
    {
        // запятая уже есть в поле редактирования
        e->Handled = true;
    }
    return;
}

if ( Char::IsControl(e->KeyChar) )
{
    // <Enter>, <Backspace>, <Esc>
    if ( e->KeyChar == (char) Keys::Enter)
        // нажата клавиша <Enter>
        // установить "фокус" на кнопку ОК
        button1->Focus();
    return;
}

// остальные символы запрещены
e->Handled = true;
}
```

Фунты-килограммы

Программа **Фунты-килограммы**, ее форма приведена на рис. 1.2, показывает, как можно управлять доступностью командной кнопки в зависимости от наличия данных в поле редактирования.

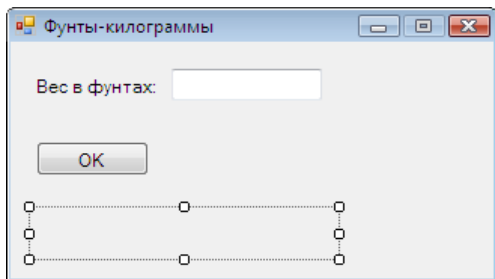


Рис. 1.2. Форма программы **Фунты-килограммы**

```
// пользователь изменил данные в поле редактирования
private: System::Void textBox1_TextChanged
    (System::Object^ sender, System::EventArgs^ e)
{
    label2->Text = ""; // очистить поле отображения
                       // результата расчета

    if (textBox1->Text->Length == 0)
        // в поле редактирования нет данных
        // сделать кнопку ОК недоступной
        button1->Enabled = false;
    else
        // сделать кнопку ОК доступной
        button1->Enabled = true;
}

// щелчок на кнопке ОК
private: System::Void button1_Click(System::Object^ sender,
System::EventArgs^ e)
{
    double funt; // вес в фунтах
    double kg;   // вес в килограммах

    funt = Convert::ToDouble(textBox1->Text);
```

```
// 1 фунт = 409,5 грамма
```

```
kg = funt * 0.4095;
```

```
label2->Text = funt.ToString("N") + " ф. = " +  
                kg.ToString("N") + " кг";
```

```
}
```

```
// нажатие клавиши в поле textBox1
```

```
private: System::Void textBox1_KeyPress(System::Object^  
sender, System::Windows::Forms::KeyPressEventArgs^ e)
```

```
{
```

```
    if ((e->KeyChar >= '0') && (e->KeyChar <= '9'))
```

```
        // цифры – правильные символы
```

```
        return;
```

```
    if (e->KeyChar == '.')
```

```
        // точку заменим на запятую
```

```
        e->KeyChar = ',';
```

```
    if (e->KeyChar == ',')
```

```
    {
```

```
        // в поле редактирования не может
```

```
        // быть больше одной запятой и запятая
```

```
        // не может быть первым символом
```

```
        if ( (textBox1->Text->IndexOf(',') != -1) ||  
            ( textBox1->Text->Length == 0) )
```

```
        {
```

```
            e->Handled = true;
```

```
        }
```

```
        return;
```

```
    }
```

```
    if ( Char::IsControl(e->KeyChar) )
```

```
    {
```

```
        // <Enter>, <Backspace>, <Esc>
```

```

if ( e->KeyChar == (char) Keys::Enter)
    // нажата клавиша <Enter>
    // установить курсор на кнопку ОК
    button1->Focus();
    return;
}

```

```

// остальные символы запрещены

```

```

e->Handled = true;

```

```

}

```

Конвертор

Программа **Конвертор**, ее форма приведена на рис. 1.3, демонстрирует обработку одной функцией событий от нескольких однотипных компонентов. Функции обработки событий `KeyPress` и `TextChanged` для компонента `textBox1` (поле **Курс**) создаются обычным образом, затем они указываются в качестве функций обработки соответствующих событий для компонента `textBox2` (поле **Цена**).

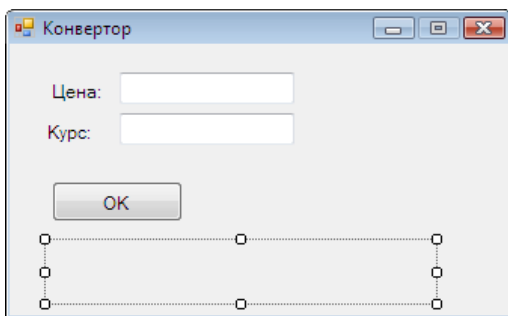


Рис. 1.3. Форма программы **Конвертор**

```

// щелчок на кнопке ОК

```

```

private: System::Void button1_Click(System::Object^ sender,
System::EventArgs^ e)

```

```

{

```



```
double usd; // цена в долларах
double k; // курс
double rub; // цена в рублях

usd = System::Convert::ToDouble(textBox1->Text);
k = System::Convert::ToDouble(textBox2->Text);

rub = usd * k;

label3->Text = usd.ToString("n") + "$ = " +
              rub.ToString("c");
}
```

```
// Эта функция обрабатывает нажатие клавиши в полях
// редактирования textBox1 (Курс) и textBox2 (Цена).
// Сначала надо обычным образом создать функцию
// обработки события KeyPress для компонента
// textBox1, затем – указать ее в качестве
// обработчика этого же события для компонента textBox2
```

```
private: System::Void textBox1_KeyPress(System::Object^
sender, System::Windows::Forms::KeyPressEventArgs^ e)
{
    if ((e->KeyChar >= '0') && (e->KeyChar <= '9'))
        return;

    if (e->KeyChar == '.') e->KeyChar = ',';

    if (e->KeyChar == ',')
    {
        if (sender->Equals(textBox1)) {
            if ((textBox1->Text->IndexOf(',') != -1) ||
                (textBox1->Text->Length == 0))
            {
                e->Handled = true;
            }
        }
    }
}
```

```
}
else {
    if ((textBox2->Text->IndexOf(',') != -1) ||
        (textBox2->Text->Length == 0))
    {
        e->Handled = true;
    }
}
return;
}

if (Char::IsControl(e->KeyChar))
{
    if (e->KeyChar == (char)Keys::Enter)
    {
        if (sender->Equals(textBox1))
            // клавиша <Enter> нажата в поле Курс
            // переместить курсор в поле Цена
            textBox2->Focus();
        else
            // клавиша <Enter> нажата в поле Цена
            // установить фокус на кнопку ОК
            button1->Focus();
    }
    return;
}

// остальные символы запрещены
e->Handled = true;
}
```

```
// Функция обрабатывает событие TextChanged (изменился
// текст в поле редактирования) обоих компонентов TextBox.
// Сначала надо обычным образом создать функцию
// обработки события TextChanged для компонента
// textBox1, затем – указать ее в качестве
```

```
// обработчика события TextChanged для компонента textBox2
private: System::Void textBox1_TextChanged(System::Object^
sender, System::EventArgs^ e)
{
    label3->Text = ""; // очистить поле отображения
                        // результата расчета

    if ((textBox1->Text->Length == 0) || (textBox2->Text-
>Length == 0))
        // в поле редактирования нет данных
        // сделать кнопку ОК недоступной
        button1->Enabled = false;
    else
        // сделать кнопку ОК доступной
        button1->Enabled = true;
}
```

Фотоателье

Программа **Фото**, ее форма приведена на рис. 1.4, позволяет рассчитать стоимость печати фотографий. Демонстрирует использование компонента `RadioButton`.

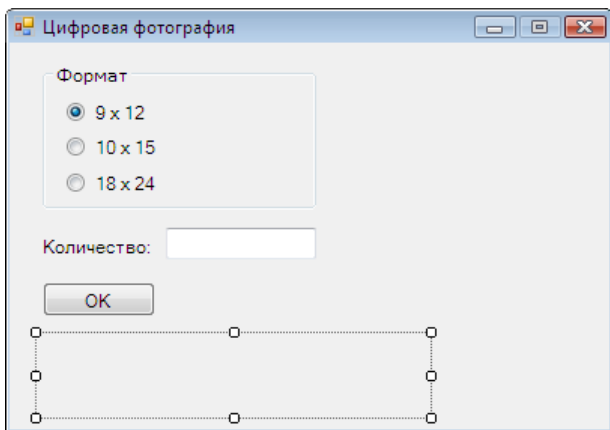


Рис. 1.4. Форма программы **Фото**

```
// щелчок на кнопке ОК
```

```
private: System::Void button1_Click(System::Object^ sender,
                                     System::EventArgs^ e)
{
    double cena = 0 ; // цена
    int n;           // кол-во фотографий
    double sum;     // сумма

    if (radioButton1->Checked)
        cena = 8.50;
    if (radioButton2->Checked)
        cena = 10;
    if (radioButton3->Checked)
        cena = 15.5;

    n = Convert::ToInt32(textBox1->Text);
    sum = n * cena;

    label2->Text = "Цена: " + cena.ToString("c") +
                  "\nКоличество: " + n.ToString() + "шт.\n" +
                  "Сумма заказа: " + sum.ToString("C");
}
```

```
// изменилось содержимое поля редактирования
```

```
private: System::Void textBox1_TextChanged(System::Object^
sender, System::EventArgs^ e)
{
    if (textBox1->Text->Length == 0)
        button1->Enabled = false;
    else
        button1->Enabled = true;

    label2->Text = "";
}
```

```
// щелчок на радиокнопке
```

```
// Функция обрабатывает событие Click компонентов
// radioButton1, radioButton2 и radioButton3
private: System::Void radioButton1_Click(System::Object^
sender, System::EventArgs^ e)
{
    label2->Text = "";

    // установить курсор в поле Количество
    textBox1->Focus();
}

// Чтобы в поле Количество можно было
// ввести только целое число
private: System::Void textBox1_KeyPress(System::Object^
sender, System::Windows::Forms::KeyPressEventArgs^ e)
{
    if ((e->KeyChar >= '0') && (e->KeyChar <= '9'))
        return;

    if (Char::IsControl(e->KeyChar))
    {
        if (e->KeyChar == (char)Keys::Enter)
        {
            // нажата клавиша <Enter>
            button1->Focus();
        }
        return;
    }

    // остальные символы запрещены
    e->Handled = true;
}
```

Комплектация

Программа **Комплектация**, ее окно приведено на рис. 1.5, позволяет посчитать стоимость автомобиля в зависимости от выбранной комплектации. Демонстрирует использование компонента `CheckBox`. Для отображения картинки используется компонент `PictureBox`. Загрузка картинки выполняется в начале работы программы. Делает это конструктор формы.

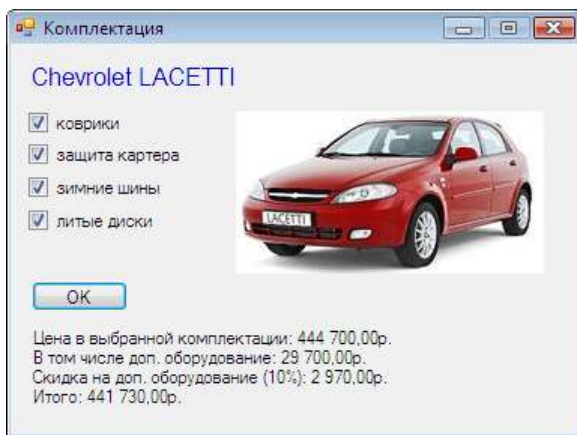


Рис. 1.5. Окно программы **Комплектация**

```
// конструктор
```

```
Form1(void)
```

```
{
```

```
InitializeComponent();
```

```
this->pictureBox1->Image =
```

```
Image::FromFile(
```

```
Application::StartupPath+"\\Lacetti_r.jpg");
```

```
/* получить имя папки Изображения можно так:
```

```
System::Environment::GetFolderPath(System::Environment::
SpecialFolder::MyPictures)
```

```
*/
    }

// щелчок на кнопке ОК
private: System::Void button1_Click(System::Object^ sender,
                                     System::EventArgs^ e)
{
    double cena;      // цена в базовой комплектации
    double dop;       // сумма за доп. оборудование
    double discount;  // скидка
    double total;     // общая сумма

    cena = 415000;
    dop = 0;

    if (checkBox1->Checked)
    {
        // коврики
        dop += 1200;
    }

    if (checkBox2->Checked)
    {
        // защита картера
        dop += 4500;
    }

    if (checkBox3->Checked)
    {
        // зимние шины
        dop += 12000;
    }

    if (checkBox4->Checked)
    {
```

```
// ЛИТЫЕ ДИСКИ
dop += 12000;
}

total = cena + dop;

System::String ^st;
st = "Цена в выбранной комплектации: " +
    total.ToString("C");
if ( dop != 0)
{
    st += "\nВ том числе доп. оборудование: " +
        dop.ToString("C");
}

if ((checkBox1->Checked) && (checkBox2->Checked) &&
    (checkBox3->Checked) && (checkBox4->Checked))
{
    // скидка предоставляется, если
    // выбраны все опции
    discount = dop * 0.1;
    total = total - discount;
    st += "\nСкидка на доп. оборудование (10%): " +
        discount.ToString("C") +
        "\nИтого: " + total.ToString("C");
}

label2->Text = st;
}
```

```
// щелчок на компоненте CheckBox.
// Функция обрабатывает событие Click на
// компонентах checkBox1-checkBox4
```



```
private: System::Void checkBox1_Click(System::Object^ sender,
                                     System::EventArgs^ e)
{
    label2->Text = "";
}
```

Жалюзи

Программа **Жалюзи**, ее форма приведена на рис. 1.6, демонстрирует использование компонента `ComboBox`. Компонент используется для выбора материала (пластик, алюминий, соломка, текстиль). Следует обратить внимание на то, что настройку компонента `ComboBox` выполняет конструктор формы.

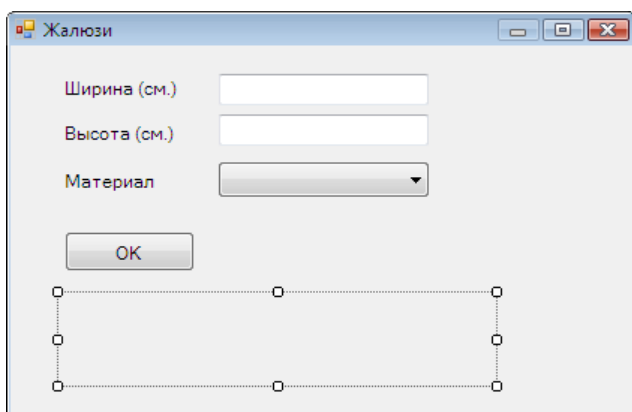


Рис. 1.6. Форма программы Жалюзи

```
// конструктор
Form1(void)
{
    InitializeComponent();

    // настройка компонентов
    comboBox1->DropDownStyle = ComboBoxStyle::DropDownList;
```

```

// сформировать список
comboBox1->Items->Add("пластик");
comboBox1->Items->Add("алюминий");
comboBox1->Items->Add("бамбук");
comboBox1->Items->Add("соломка");
comboBox1->Items->Add("текстиль");

// Элементы списка нумеруются с нуля.
// По умолчанию выбран первый элемент
comboBox1->SelectedIndex = 0;
}

```

// щелчок на кнопке ОК

```

private: System::Void button1_Click(System::Object^ sender,
                                     System::EventArgs^ e)
{
    double w;
    double h;
    double cena = 0; // цена за 1 кв.м.
    double sum;

    w = Convert::ToDouble(textBox1->Text);
    h = Convert::ToDouble(textBox2->Text);

    switch (comboBox1->SelectedIndex)
    {
        case 0: cena = 100; break; // пластик
        case 1: cena = 250; break; // алюминий
        case 2: cena = 170; break; // бамбук
        case 3: cena = 170; break; // соломка
        case 4: cena = 120; break; // текстиль
    }

    sum = (w * h) / 10000 * cena;
    label4->Text =
        "Размер: " + w + " x " + h + " см.\n" +
        "Цена (р./м.кв.): " + cena.ToString("c") +

```

```
        "\nСумма: " + sum.ToString("c");
    }

// Нажатие клавиши в поле редактирования.
// Функция обрабатывает событие KeyPress
// компонентов textBox1 и textBox2
private: System::Void textBox1_KeyPress(System::Object^
sender, System::Windows::Forms::KeyPressEventArgs^ e)
{
    if ((e->KeyChar >= '0') && (e->KeyChar <= '9'))
        return;

    if (Char::IsControl(e->KeyChar))
    {
        if (e->KeyChar == (char)Keys::Enter)
        {
            if (sender->Equals(textBox1))
                // клавиша <Enter> нажата в поле Ширина
                // переместить курсор в поле Высота
                textBox2->Focus();
            else
                // клавиша <Enter> нажата в поле Высота
                // переместить фокус на comboBox1
                button1->Focus();
        }
        return;
    }

    // остальные символы запрещены
    e->Handled = true;
}

// в списке Материал пользователь
// выбрал другой элемент
private: System::Void comboBox1_SelectedIndexChanged
(System::Object^ sender, System::EventArgs^
```

```
e)
{
    label4->Text = "";
}

// пользователь изменил размер (содержимое
// textBox1 или textBox2)
private: System::Void textBox1_TextChanged(System::Object^
sender, System::EventArgs^ e)
{
    label4->Text = "";

    if ((textBox1->Text->Length == 0) ||
        (textBox2->Text->Length == 0))

        button1->Enabled = false;
    else
        button1->Enabled = true;
}
```

Калькулятор

Программа **Калькулятор** демонстрирует создание компонентов в коде. Кнопки калькулятора, объединенные в массив `btn` компоненты `Button`, создает и настраивает конструктор формы. Он же назначает кнопкам функции обработки события `Click`. Форма и окно программы приведены на рис. 1.7.

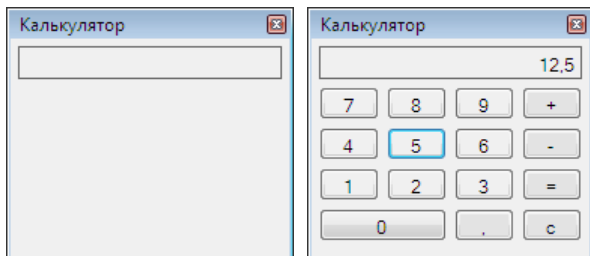


Рис. 1.7. Форма и окно программы **Калькулятор**

```
// переменные класса
```

```
private:
```

```
static int bw = 40, bh = 22; // размер кнопки
```

```
// расстояние между кнопками по X и Y
```

```
static int dx = 5, dy = 5;
```

```
// кнопки
```

```
array<System::Windows::Forms::Button^> ^btn;
```

```
// текст на кнопках
```

```
static array<String^> ^btnText = {"7","8","9","+",  
                                "4","5","6","-",  
                                "1","2","3","=",  
                                "0","","","c"};
```

```
// при нажатии кнопку будем идентифицировать
```

```
// по значению свойства Tag
```

```
static array<int^> ^btnTag =  
    {7,8,9,-3,  
     4,5,6,-4,  
     1,2,3,-2,  
     0,-1,-5};
```

```
double ac; // аккумулятор
```

```
int op; // код операции
```

```
bool fd;
```

```
// fd == true - ждем первую цифру числа,
```

```
// например, после нажатия кнопки "+"
```

```
// fd == false - ждем следующую цифру
```

```
// конструктор формы
```

```
public:
```

```
Form1(void)
```

```
{
    InitializeComponent();

    // установить размер формы (клиентской области)
    int cw = 4*bw + 5*dx;
    int ch = 5*bh + 7*dy;
    this->ClientSize = System::Drawing::Size(cw,ch);

    // задать размер и положение индикатора
    labell->SetBounds(dx, dy, 4*bw + 3*dx, bh);
    labell->Text = "0";

    int x, y; // координаты кнопки

    y = labell->Bottom + dy ;

    // кнопки калькулятора объединены в массив
    btn = gcnew array<System::Windows::Forms::Button^>(15);

    int k =0;    // номер настраиваемой кнопки

    for ( int i=0; i < 4; i++)
    {
        x = dx;
        for (int j = 0; j < 4; j++)
        {
            if ( !((i == 3 ) && (j == 0)))
            {
                // создать кнопку
                btn[k] = gcnew
                    System::Windows::Forms::Button;

                // настроить кнопку
                btn[k]->SetBounds(x, y, bw, bh);
                btn[k]->Tag = btnTag[k];
            }
        }
    }
}
```