Н. Культин

Microsoft®

Visual C++

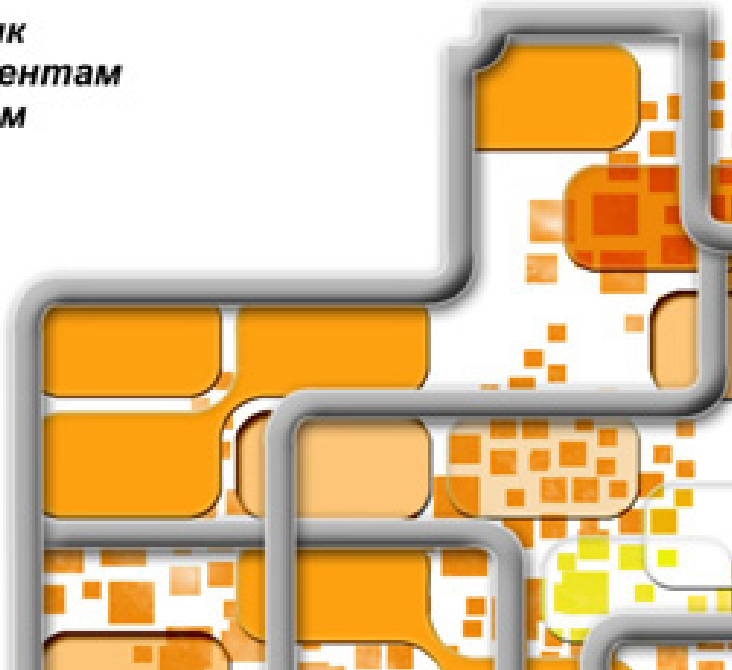
в задачах и примерах

В ногу со временем!

Базовые компоненты

Программирование графики и баз данных

Справочник по компонентам и функциям



+ CD

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

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 28.09.09.
Формат $60\times90^{1}/_{16}$. 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

Оглавление

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 `RadoiButton`.
- 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.

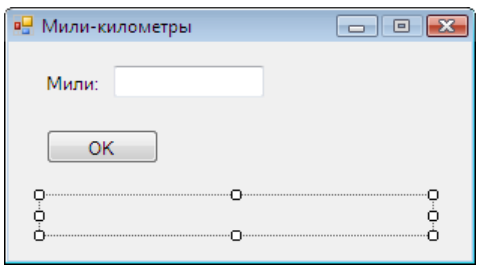


Рис. 1.1. Форма программы **Мили-километры**

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 OK
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 OK
            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.

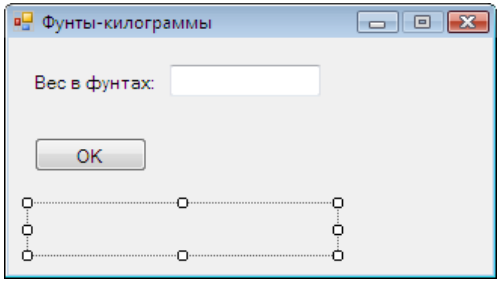


Рис. 1.2. Форма программы **Фунты-килограммы**

```
// 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 OK недоступной
        button1->Enabled = false;
    else
        // сделать кнопку OK доступной
        button1->Enabled = true;
}

// щелчок на кнопке OK
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 OK
            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**).

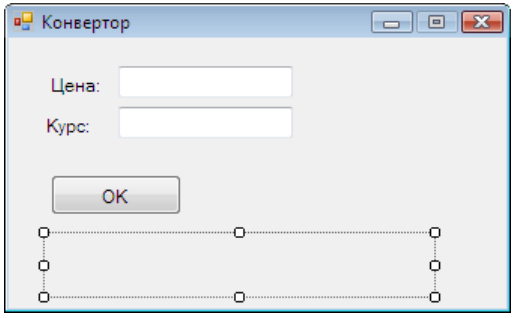


Рис. 1.3. Форма программы **Конвертор**

```
// щелчок на кнопке OK
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 OK
               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 компонета 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 OK недоступной
        button1->Enabled = false;
    else
        // сделать кнопку OK доступной
        button1->Enabled = true;
}
```

Фотоателье

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

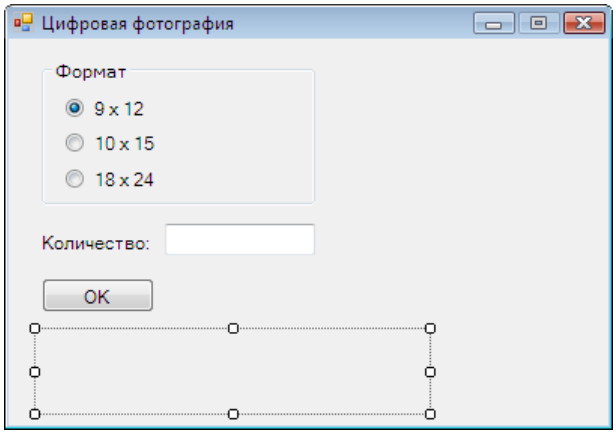


Рис. 1.4. Форма программы **Фото**

```
// щелчок на кнопке OK
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.

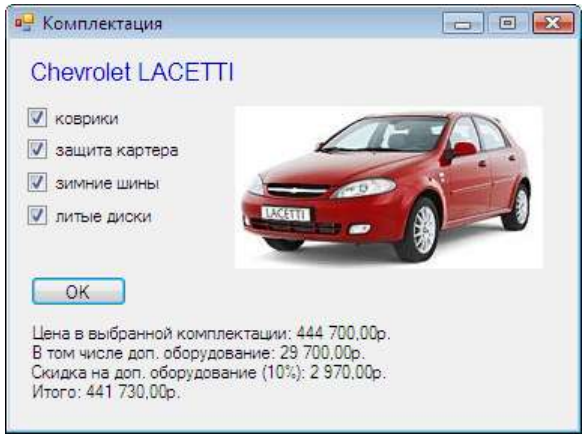


Рис. 1.5. Окно программы **Комплектация**

```
// 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 OK
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)
    {
```