Н. Культин

bhv

Microsoft®

Visual C++

в задачах и примерах

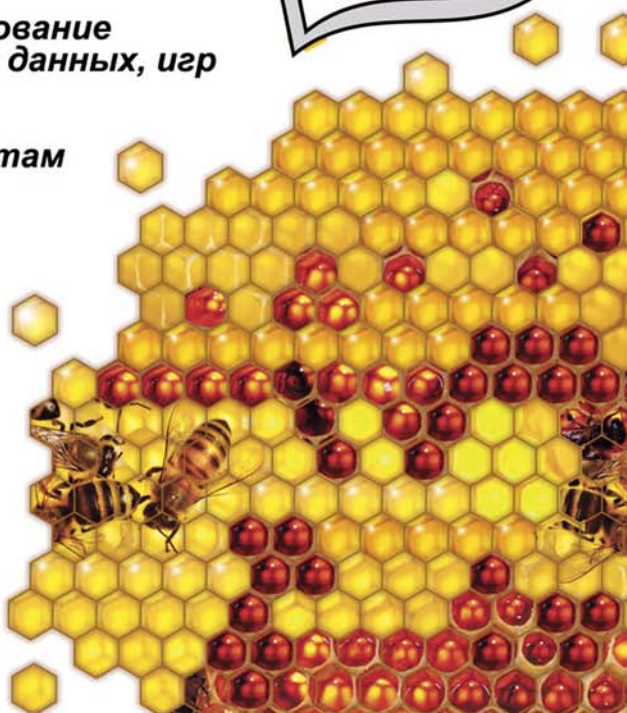
2-е издание

В ногу со временем!

Базовые компоненты

Программирование графики, баз данных, игр

Справочник по компонентам и функциям



+

Материалы на www.bhv.ru

Никита Культин

Microsoft® Visual C++ в задачах и примерах

2-е издание, исправленное

Санкт-Петербург
«БХВ-Петербург»
2014

УДК 004.438 Visual C++
ББК 32.973.26-018.1
К90

Культин Н. Б.

К90 Microsoft® Visual C++ в задачах и примерах. — 2-е изд., исправл. — СПб.: БХВ-Петербург, 2014. — 272 с.: ил.

ISBN 978-5-9775-3321-8

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 FTP-сервере издательства находятся коды примеров из книги.

Для начинающих программистов

УДК 004.438 Visual C++
ББК 32.973.26-018.1

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Евгений Рыбаков*
Зав. редакцией	*Екатерина Капалыгина*
Компьютерная верстка	*Ольги Сергиенко*
Корректор	*Зинаида Дмитриева*
Дизайн серии	*Инны Тачиной*
Оформление обложки	*Марины Дамбиевой*

Подписано в печать 28.02.14.
Формат 60×90$^{1}/_{16}$. Печать офсетная. Усл. печ. л. 17.
Тираж 1500 экз. Заказ №
"БХВ-Петербург", 191036, Санкт-Петербург, Гончарная ул., 20.

Первая Академическая типография "Наука"
199034, Санкт-Петербург, 9 линия, 12/28

ISBN 978-5-9775-3321-8

Оглавление

Предисловие

В последнее время в общем объеме вновь создаваемых программ 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 Jast In Time — в тот же момент, "на лету"), являющийся элементом виртуальной выполняющей системы (Virtual Execution System, VES). Выполнение .NET-приложений в операционной системе Microsoft Windows обеспечивает платформа Microsoft .NET Framework.

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

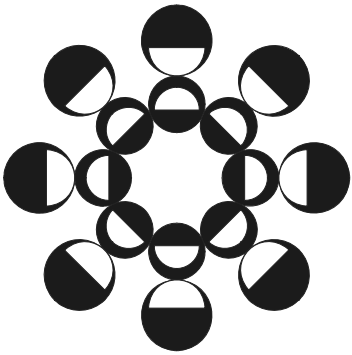
Среда программирования Microsoft Visual C++ является инструментом разработки приложений различного типа для операционной системы Microsoft Windows. В ней интегрированы удобный конструктор форм, специализированный редактор кода, отладчик и другие полезные инструменты.

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!



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` — радиокнопка;
  - `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.

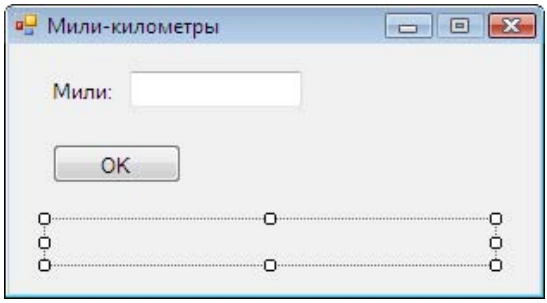


Рис. 1.1. Форма программы **Мили-километры**

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 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.

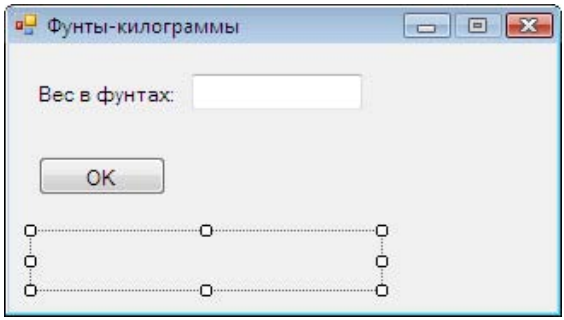


Рис. 1.2. Форма программы **Фунты-килограммы**

```
// 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**).

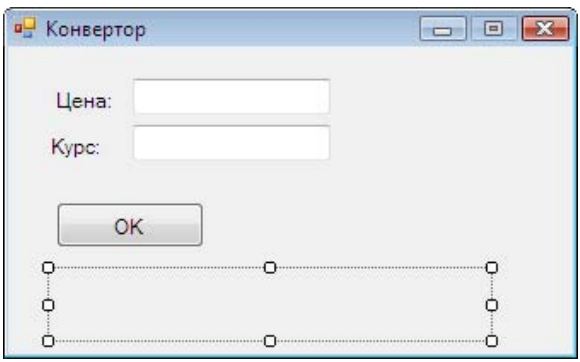


Рис. 1.3. Форма программы **Конвертор**

```
// 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 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 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`.

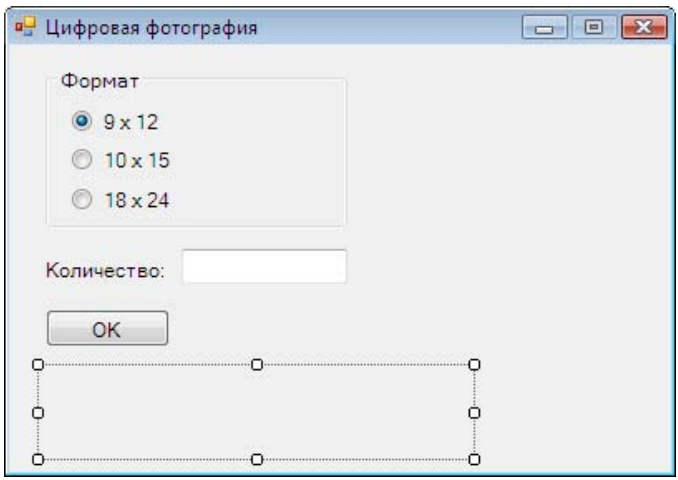


Рис. 1.4. Форма программы **Фото**

```
// 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.

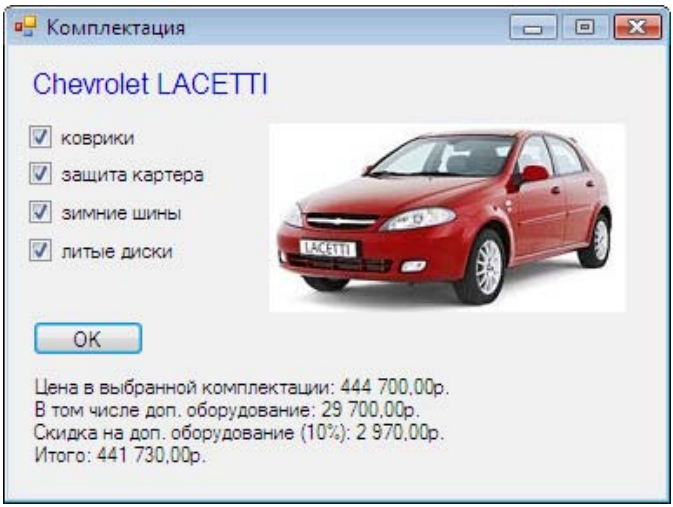


Рис. 1.5. Окно программы **Комплектация**

```
// 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)
    {
```