Никита Культин

# C++ Builder
## в задачах и примерах

Готовые решения и тексты программ

*Использование базовых компонентов*

*Программирование графики, игр, мультимедиа и баз данных*

*Справочник по компонентам и функциям*

+ CD-ROM

**Никита Культин**

# C++ Builder
## в задачах и примерах

Санкт-Петербург
«БХВ-Петербург»
2005

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

**Культин Н. Б.**

К90 С++ Builder в задачах и примерах. — СПб.: БХВ-Петербург, 2005. — 336 с.: ил.

ISBN 5-94157-631-5

Книга представляет собой сборник программ и задач для самостоятельного решения в среде разработки С++ Builder. Примеры различной сложности — от простейших до приложений работы с графикой, мультимедиа и базами данных — демонстрируют назначение компонентов и раскрывают тонкости процесса программирования в С++ Builder. Справочник содержит описания базовых компонентов и наиболее часто используемых функций. На прилагаемом компакт-диске находятся исходные тексты программ.

*Для начинающих программистов*

УДК 681.3.068+800.92С++
ББК 32.973.26-018.1

**Группа подготовки издания:**

| | |
|---|---|
| Главный редактор | *Екатерина Кондукова* |
| Зам. главного редактора | *Шишигин Игорь* |
| Зав. редакцией | *Григорий Добин* |
| Редактор | *Андрей Смышляев* |
| Компьютерная верстка | *Татьяны Олоновой* |
| Корректор | *Наталия Першакова* |
| Дизайн обложки | *Игоря Цырульникова* |
| Зав. производством | *Николай Тверских* |

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 16.06.05.
Формат 60×90 1/16. Печать офсетная. Усл. печ. л. 21.
Тираж 5000 экз. Заказ №
"БХВ-Петербург", 194354, Санкт-Петербург, ул. Есенина, 5Б.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.02.953.Д.006421.11.04 от 11.11.2004 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 5-94157-631-5

# Содержание

# Предисловие

В последнее время резко возрос интерес к программированию. Это связано с развитием и внедрением в повседневную жизнь информационных и коммуникационных технологий. Если человек имеет дело с компьютером, то, рано или поздно, у него возникает желание, а иногда и необходимость, программировать.

Бурное развитие вычислительной техники, потребность в эффективных средствах разработки программного обеспечения привели к появлению систем программирования, ориентированных на так называемую "быструю разработку". В основе систем быстрой разработки или RAD-систем (Rapid Application Development — среда быстрой разработки приложений) лежит технология визуального проектирования и событийного программирования, суть которой заключается в том, что среда разработки берет на себя большую часть рутины, оставляя программисту работу по конструированию диалоговых окон и созданию функций обработки событий. Производительность программиста при использовании RAD-систем — фантастическая!

Одной из широко используемых RAD-систем является Borland C++Builder, которая позволяет создавать различные программы: от простейших однооконных приложений до программ управления распределенными базами данных. В качестве языка программирования в среде Borland C++Builder используется C++.

Чтобы научиться программировать, надо программировать — писать программы, решать конкретные задачи. Для этого надо изучить язык программирования и среду разработки. И здесь хорошим подспорьем могут быть программы, которые демонстрируют назначение компонентов и особенности их использования.

В книге, которую вы держите в руках, собраны разнообразные примеры, которые демонстрируют технологию создания программ, возможности среды разработки, назначение компонентов,

знакомят с принципами работы с графикой, звуком, базами данных. Следует обратить внимание на то, что большинство примеров не являются учебными — это вполне работоспособные программы.

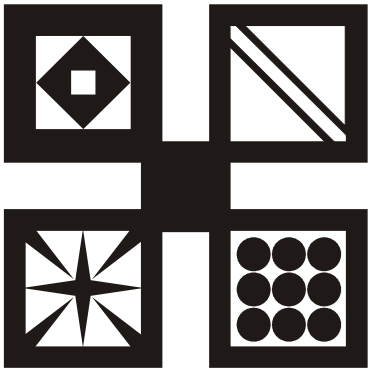
Состоит книга из двух частей.

Первая часть содержит примеры, представленные в виде краткого описания, диалоговых окон и прокомментированных текстов программ.

Вторая часть книги — это краткий справочник, в котором можно найти описание базовых компонентов и наиболее часто используемых функций.

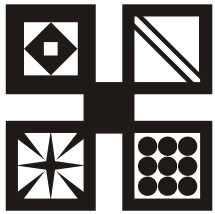
Научиться программировать можно, только решая конкретные задачи. При этом успехи, достигнутые в программировании, в значительной степени зависят от опыта. Поэтому чтобы получить максимальную пользу от книги, вы должны активно с ней работать. Изучайте листинги, старайтесь понять, как работают программы. Не бойтесь экспериментировать — вносите изменения в программы.

Если что-то не понятно, обратитесь к справочнику в конце книге, к справочной системе C++Builder или к литературе, например, к книге **Культин Н. Б. Самоучитель C++Builder. — СПб.: БХВ-Петербург, 2004**. В ней помимо описания среды разработки, компонентов, процессов создания и отладки программ вы найдете ответы на многие вопросы, в том числе: как создать базу данных и зарегистрировать ее в системе, как, при помощи Microsoft Help Workshop, создать справочную систему, как, используя InstallShield Express, создать дистрибутив (пакет для установки программы).



# ЧАСТЬ 1

# Примеры и задачи



# Базовые компоненты

В этом разделе приведены примеры, которые должны продемонстрировать назначение и технологию работы с базовыми компонентами.

## Общие замечания

- Процесс создания программы в C++Builder состоит из двух шагов: сначала нужно создать форму программы (диалоговое окно), а затем функции обработки *событий*. Форма *приложения* (так принято называть прикладные программы, работающие в Windows) создается путем добавления в нее *компонентов* и последующей их настройки.
- В форме практически любого приложения есть компоненты, которые обеспечивают интерфейс (взаимодействие) между программой и пользователем. Такие компоненты называют базовыми. К базовым компонентам можно отнести:
  - `Label` — поле вывода текста;
  - `Edit` — поле редактирования текста;
  - `Button` — командная кнопка;
  - `CheckBox` — независимая кнопка выбора;
  - `RadioButton` — зависимая кнопка выбора;
  - `ListBox` — список выбора;
  - `ComboBox` — комбинированный список выбора.
- Вид компонента, его размер и 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 полей редактирования (компонент `Edit`), списка выбора (компонент `ListBox`) или комбинированного списка (компонент `ComboBox`). Для ввода значений логического типа можно использовать компоненты `CheckBox` и `RadoiButton`.
- Результат программа может вывести в поле вывода текста (компонент `Label`) или в окно сообщения (функции `ShowMessage`, `MessageDlg`).
- Для преобразования текста, например, находящегося в поле редактирования, в целое число нужно использовать функцию `StrToInt`, а в дробное — функцию `StrToFloat`. Для преобразования целого, например, значения переменной, в строку нужно использовать функцию `IntToStr`, а для преобразования дробного — функцию `FloatToStr` или `FloatToStrF`.

## Конвертор

Программа **Конвертор** пересчитывает цену из долларов в рубли. Демонстрирует использование компонентов `TextBox` и `Label` для ввода и отображения числовых данных. Программа спроектирована таким образом, что пользователь может ввести в поля редактирования только правильные данные (число). Форма программы приведена на рис. 1.1.

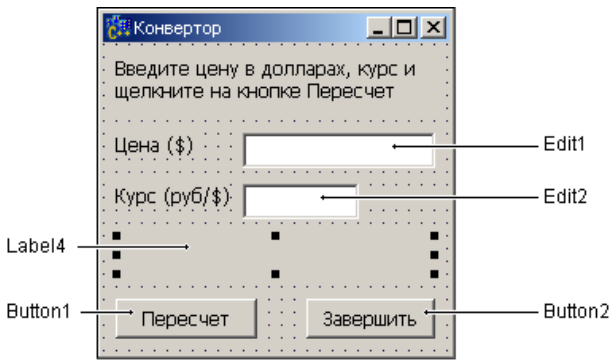



**Рис. 1.1.** Форма программы **Конвертор**

```
// нажатие клавиши в поле Цена
void __fastcall TForm1::Edit1KeyPress(TObject *Sender, char
&Key)
{
     // код запрещенного символа заменим нулем, в результате
    // символ в поле редактирования не появится

    // Key - код нажатой клавиши
    // проверим, является ли символ допустимым
    if ((Key >= '0') && (Key <= '9')) //цифра
        return;

    // глобальная переменная DecimalSeparator
    // содержит символ, используемый в качестве разделителя
    // при записи дробных чисел
    if (Key == DecimalSeparator)
    {
       if ((Edit1->Text).Pos(DecimalSeparator) != 0)
           Key = 0; // разделитель уже введен
       return;
    }

    if (Key == VK_BACK) // клавиша <Backspace>
        return;

    if (Key == VK_RETURN) // клавиша <Enter>
    {
        Edit2->SetFocus();
        return;
    }

    // остальные клавиши запрещены
    Key = 0; // не отображать символ
}
```

```
// нажатие клавиши в поле Курс
void __fastcall TForm1::Edit2KeyPress(TObject *Sender,
                                      char &Key)
{
   if ((Key >= '0') && (Key <= '9')) //цифра
       return;

    if (Key == DecimalSeparator)
    {
      if ((Edit2->Text).Pos(DecimalSeparator) != 0)
          Key = 0; // разделитель уже введен
      return;
    }

    if (Key == VK_BACK) // клавиша <Backspace>
       return;

    if (Key == VK_RETURN) // клавиша <Enter>
    {
       Button1->SetFocus();// переход к кнопке Вычислить
       // повторное нажатие клавиши <Enter>
       // активизирует процесс вычисления денег
        return;
    };

    // остальные клавиши запрещены
    Key = 0; // не отображать символ
}

// щелчок на кнопке Пересчет
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    float usd; // цена в долларах
```

```
    float k;   // курс
    float rub; // цена в рублях

    // проверим, введены ли данные в поля Цена и Курс
    if (((Edit1->Text).Length() == 0) ||
              ((Edit2->Text).Length() == 0))
    {
       MessageDlg("Надо ввсети цену и курс",
           mtInformation, TMsgDlgButtons() << mbOK, 0);
        if ((Edit1->Text).Length() == 0)
           Edit1->SetFocus(); // курсор в поле Цена
        else
           Edit2->SetFocus(); // курсор в поле Курс
        return;
    };

    // ввод исходных данных
    usd = StrToFloat(Edit1->Text);
    k = StrToFloat(Edit2->Text);

    // вычисление
    rub = usd * k;

    // вывод результата
    Label4->Caption = FloatToStrF(usd,ffGeneral,7,2) +
            "$ = "+FloatToStrF(rub,ffGeneral,7,2) + " руб." ;
}

//щелчок на кнопке Завершить
void __fastcall TForm1::Button2Click(TObject *Sender)
{
    Form1->Close(); // закрыть форму приложения
}
```

# Фунты-килограммы

Программа **Фунты-килограммы**, форма которой приведена на рис. 1.2, позволяет пересчитать вес из фунтов в килограммы. Программа спроектирована таким образом, что кнопка **Пересчет** доступна только в том случае, если пользователь ввел исходные данные.

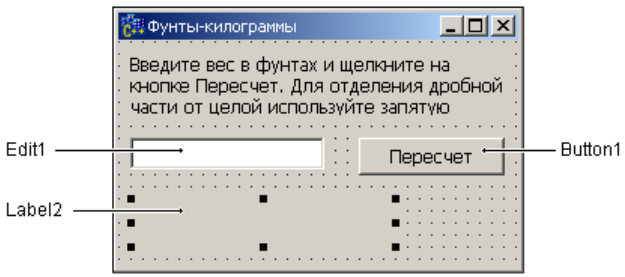



**Рис. 1.2.** Кнопка **Пересчет** доступна только тогда, когда в поле редактирования есть данные

```
__fastcall TForm1::TForm1(TComponent* Owner) : TForm(Owner)
{
    /* так как поле Edit1 пустое (пользователь
    еще не ввел исходные данные), то
    сделаем кнопку Пересчет недоступной */
    Button1->Enabled  =  False;
}

// нажатие клавиши в поле Edit1
void __fastcall TForm1::Edit1KeyPress(TObject *Sender, char
&Key)
{
    // код запрещенного символа заменим нулем, в результате
    // символ в поле редактирования не появится

    // Key - код нажатой клавиши
    // проверим, является ли символ допустимым
    if ((Key >= '0') && (Key <= '9')) //цифра
        return;
```

```
    // глобальная переменная DecimalSeparator
    // содержит символ, используемый в качестве разделителя
    // при записи дробных чисел
    if (Key == DecimalSeparator)
    {
        if ((Edit1->Text).Pos(DecimalSeparator) != 0)
            Key = 0; // разделитель уже введен
        return;
    }

    if (Key == VK_BACK) // клавиша <Backspace>
        return;

    if (Key == VK_RETURN) // клавиша <Enter>
    {
        Button1->SetFocus();
        return;
    }

    // остальные клавиши запрещены
    Key = 0; // не отображать символ
}

// Содержимое поля Edit1 изменилось
void __fastcall TForm1::Edit1Change(TObject *Sender)
{
    // проверим, есть ли в поле Edit1 исходные данные
    if ( (Edit1->Text).Length() == 0)
        Button1->Enabled  =  False; // кн. Пересчет недоступна
    else Button1->Enabled  =  True; // кн. Пересчет доступна

   Label2->Caption = "";
}
```

```
// щелчок на кнопке Пересчет
void __fastcall TForm1::Button1Click(TObject *Sender)
{
   double funt; // вес в фунтах
   double kg;   // вес в килограммах

   // кнопка Пересчет доступна только в том случае,
   // если в поле Edit1 есть данные. Поэтому,
   // наличие в поле информации можно не проверять.
   funt  =  StrToFloat(Edit1->Text);
   kg  =  funt * 0.4995;
   Label2->Caption  =  FloatToStrF(funt,ffGeneral,5,2) +
                       " ф. - это " +
                       FloatToStrF(kg,ffGeneral,5,2) + " кг";
}
```

## Сила тока

Программа **Сила тока** демонстрирует использование компонентов TextBox и Label, а также обработку исключения "деление на ноль". Форма программы показана на рис. 1.3.

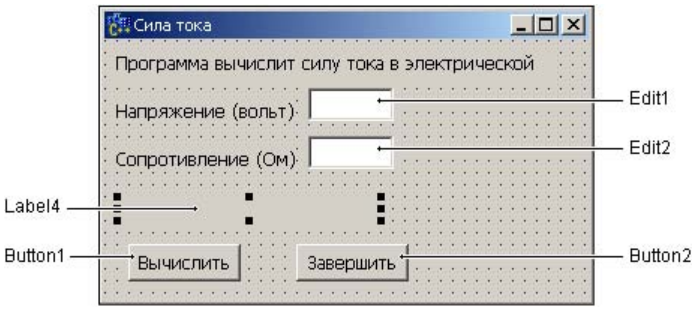



**Рис. 1.3.** Форма программы **Сила тока**

```
// щелчок на кнопке Вычислить
void __fastcall TForm1::Button1Click(TObject *Sender)
```

```
{
    float u; // напряжение
    float r; // сопротивление
    float i; // ток

    // проверим, введены ли данные в поля Напряжение и
    // Сопротивление
    if ( ((Edit1->Text).Length() == 0) ||
         ((Edit2->Text).Length() == 0))
    {
        MessageDlg("Надо ввести напряжение и сопротивление",
                 mtInformation, TMsgDlgButtons() << mbOK, 0);
        if ((Edit1->Text).Length() == 0)
            Edit1->SetFocus(); // курсор в поле Напряжение
        else
            Edit2->SetFocus(); // курсор в поле Сопротивление
        return;
    };

    // получить данные из полей ввода
    u = StrToFloat(Edit1->Text);
    r = StrToFloat(Edit2->Text);

    // вычислить ток
    try
    {
       i = u/r;
    }
    catch (EZeroDivide &e)
    {
        ShowMessage("Величина сопротивления не должна быть"
                    "равна нулю");
        Edit2->SetFocus();   // курсор в поле Сопротивление
        return;
    }
```

```
    // вывести результат в поле Label4
   Label4->Caption = "Ток : " +
                     FloatToStrF(i,ffGeneral,7,2) + " A";
}

// нажатие клавиши в поле Напряжение
// коды запрещенных клавиш заменим нулем, в результате
// символы этих клавиш в поле редактирования не появятся
void __fastcall TForm1::Edit1KeyPress(TObject *Sender,
                                      char &Key)
{
    // Key - код нажатой клавиши
    // проверим, является ли символ допустимым

    if ( ( Key >= '0') && ( Key <= '9' ) ) // цифра
        return;

    // Глобальная переменная Decimalseparator
    // содержит символ, используемый в качестве разделителя
    // при записи дробных чисел
    if ( Key == DecimalSeparator)
    {
       if ( (Edit1->Text).Pos(DecimalSeparator) != 0 )
           Key = 0; // разделитель уже введен
    return;
    }

    if (Key == VK_BACK) // клавиша <Backspace>
        return;

    if ( Key == VK_RETURN) // клавиша <Enter>
    {
       Edit2->SetFocus();
       return;
    };
```

```
    // остальные клавиши запрещены
    Key = 0; // не отображать символ
}

// нажатие клавиши в поле Сопротивление
void __fastcall TForm1::Edit2KeyDown(TObject *Sender,
                          WORD &Key,TShiftState Shift)
{
    if ( ( Key >= '0') && ( Key <= '9' ) ) // цифра
        return;

    if ( Key == DecimalSeparator) {
       if ( (Edit2->Text).Pos(DecimalSeparator) != 0 )
           Key = 0; // разделитель уже введен
       return;
    }

    if (Key == VK_BACK) // клавиша <Backspace>
        return;

    if ( Key == VK_RETURN) // клавиша <Enter>
    {
       Button1->SetFocus(); // переход к кнопке Вычислить
                    // повторное нажатие клавиши <Enter>
                    // активизирует процесс вычисления тока
                    // см. Button1Click
       return;
    };

    // остальные клавиши запрещены
    Key = 0; // не отображать символ
}

// щелчок на кнопке Завершить
void __fastcall TForm1::Button2Click(TObject *Sender)
```

```
{
    Form1->Close(); // закрыть форму приложения
}

/* Процедура Edit1Change обрабатывает событие Change
   как поля Edit1, так и поля Edit2.
   Сначала надо создать процедуру обработки события Change
   для поля Edit1, затем - в строке события Change компонента
   Edit2 щелкнуть на значке раскрывающегося списка и выбрать
   Edit1Change. */
void __fastcall TForm1::EditChange(TObject *Sender)
{
    Label4->Caption = "";
}
```

## Сопротивление

Программа **Сопротивление**, ее форма приведена на рис. 1.4, вычисляет сопротивление электрической цепи, состоящей из двух резисторов, которые могут быть соединены последовательно или параллельно. Демонстрирует использование компонента `RadioButton`.

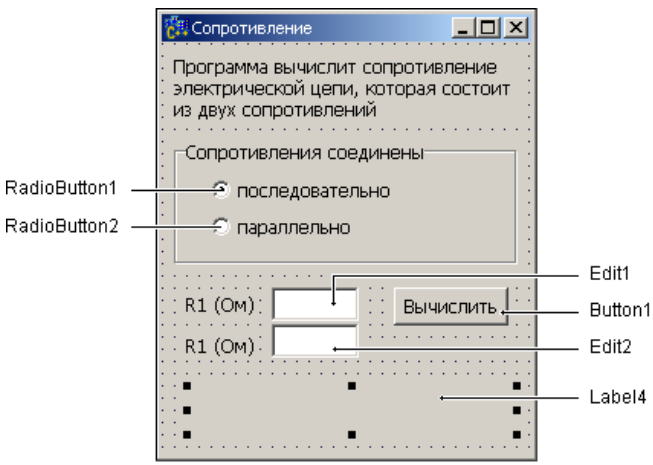



**Рис. 1.4.** Форма программы **Сопротивление**

```
// щелчок на кнопке Вычислить
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    float r1,r2,r;

    r1 = StrToFloat(Edit1->Text);
    r2 = StrToFloat(Edit2->Text);

   /* Переключатели RadioButton1 и RadioButton2
      зависимые, поэтому о типе соединения можно
      судить по состоянию одного из них */
    if ( RadioButton1->Checked )
    {
        // выбран переключатель "последовательно"
        r = r1 + r2;
    }
    else
    {
        // выбран переключатель "параллельно"
        // при вычислении сопротивления возможно
        // исключение EInvalidOp
        try
        {
            r = (r1 * r2) / (r1 + r2);
        }
        catch ( EInvalidOp &e)
        {
            ShowMessage("Необходимо задать величину"
                        "сопротивлений");
            return;
        }
    }
    Label4->Caption = FloatToStrF(r,ffGeneral,6,2) + " Ом";
}
```

```
// щелчок на переключателе "последовательно"
void __fastcall TForm1::RadioButton1Click(TObject *Sender)
{
    Label4->Caption = "";
}

// щелчок на переключателе "параллельно"
void __fastcall TForm1::RadioButton2Click(TObject *Sender)
{
    Label4->Caption = "";
}
```

## Кафе

Программа **Кафе**, ее форма приведена на рис. 1.5, демонстрирует использование компонента `CheckBox`.

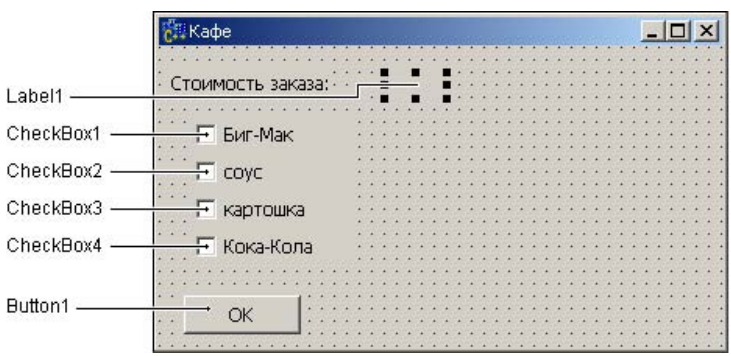



**Рис. 1.5.** Форма программы **Кафе**

```
float summ; // сумма заказа

// конструктор формы
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
```

```
{
    // сделать недоступным переключатель "соус"
    CheckBox2->Enabled = false;
}

// щелчок на переключателе "Биг-Мак"
void __fastcall TForm1::CheckBox1Click(TObject *Sender)
{
    if ( CheckBox1->Checked )
    {
        /* переключатель был сброшен,
           пользователь установил его */
        summ += 54;

        // сделать доступным переключатель "соус"
        CheckBox2->Enabled = true;
    }
    else
    {
        /* переключатель был установлен,
           пользователь сбросил его */

       summ -= 54;

       // сбросить и сделать недоступным переключатель "соус"
       if (CheckBox2->Checked)
            CheckBox2->Checked = false;
       CheckBox2->Enabled = false;
    }

    // отобразить измененную сумму в поле
    Label1->Caption = FloatToStrF(summ,ffCurrency,6,2);
}
```

```
// щелчок на переключателе "соус"
void __fastcall TForm1::CheckBox2Click(TObject *Sender)
{
    if ( CheckBox2->Checked)
        summ += 10.5;
    else
        summ -= 10.5;

    Label1->Caption = FloatToStrF(summ,ffCurrency,6,2);
}

// щелчок на переключателе "картошка"
void __fastcall TForm1::CheckBox3Click(TObject *Sender)
{
    if ( CheckBox3->Checked)
        summ += 18.5;
    else
        summ -= 18.5;

    Label1->Caption = FloatToStrF(summ,ffCurrency,6,2);
}

// щелчок на переключателе "Кока-Кола"
void __fastcall TForm1::CheckBox4Click(TObject *Sender)
{
    if ( CheckBox4->Checked)
        summ += 14;
    else
        summ -= 14;
    Label1->Caption = FloatToStrF(summ,ffCurrency,6,2);
}

// щелчок на кнопке OK
```

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    if ( (CheckBox1->Checked)&& (CheckBox2->Checked)&&
          (CheckBox3->Checked)&&(CheckBox4->Checked) )
    {
        /* пользователь заказал полный набор
           предоставить скидку 5% */
        summ = summ * 0.95;
        ShowMessage("Вам предоставляется скидка 5%.\n"
        "Сумма заказа: " + FloatToStrF(summ,ffCurrency,6,2)+
        " руб.");
    }
    else
        if ( (CheckBox1->Checked)||
             (CheckBox3->Checked)||
             (CheckBox4->Checked))
               ShowMessage("Сумма заказа: " +
                 FloatToStrF(summ,ffGeneral,6,2)+ " руб.");
        else   ShowMessage("Вы ничего не заказали");
}
```

## Любимый напиток

Программа **Любимый напиток**, ее форма приведена на рис. 1.6, демонстрирует использование компонента `ComboBox`.

Списки компонентов `ComboBox2` и `ComboBox3` формируются во время работы программы (делает это конструктор формы). Пользователь может добавить элементы в списки компонентов `ComboBox2` и `ComboBox3`, однако элемент в список компонента `ComboBox3` добавляется только в том случае, если такого элемента в списке нет.

Значения свойств компонентов `ComboBox` приведены в табл. 1.1.

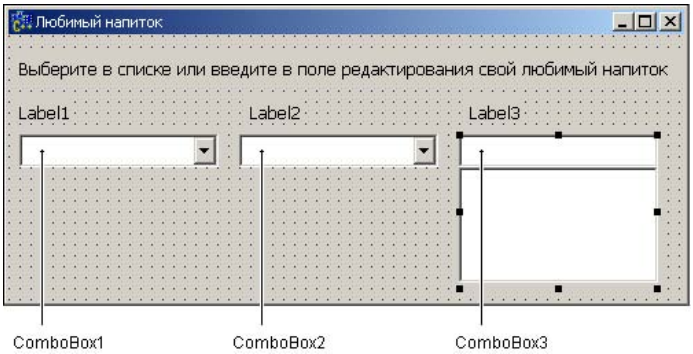



**Рис. 1.6.** Программа **Любимый напиток**

***Таблица 1.1.*** *Значения свойств компонентов* `ComboBox`

| Свойство | Значение | Комментарий |
|---|---|---|
| `ComboBox1.Style` | `csDropDownList` | Раскрывающийся список (добавить элемент в список нельзя) |
| `ComboBox2.Style` | `csDropDown` | Раскрывающийся комбинированный список. Пользователь может ввести текст в поле редактирования |
| `ComboBox3.Style` | `csSimple` | Поле редактирования и список |

```
// конструктор формы
__fastcall TForm1::TForm1(TComponent* Owner) : TForm(Owner)
{
    // сформировать список компонента ComboBox3
    ComboBox2->Sorted = true; // список упорядочен
    ComboBox2->Items->Add("Кока-Кола");
    ComboBox2->Items->Add("Меринда");
    ComboBox2->Items->Add("Пепси-Кола");
    ComboBox2->Items->Add("Спрайт");
    ComboBox2->Items->Add("Фанта");
```

```
// сформировать список компонента ComboBox3
    ComboBox2->Sorted = true; // список упорядочен
    ComboBox3->Items->Add("Чай");
    ComboBox3->Items->Add("Чай с лимоном");
    ComboBox3->Items->Add("Кофе черный");
    ComboBox3->Items->Add("Кофе со сливками");
    ComboBox3->Items->Add("Какао");
}

// выбор элемента в списке ComboBox1
void __fastcall TForm1::ComboBox1Click(TObject *Sender)
{
    Label1->Caption = ComboBox1->Text;
}

// щелчок на элементе списка компонента ComboBox2
void __fastcall TForm1::ComboBox2Click(TObject *Sender)
{
    Label2->Caption = ComboBox2->Items->
                            Strings[ComboBox2->ItemIndex];
}

// щелчок на элементе списка компонента ComboBox3
void __fastcall TForm1::ComboBox3Click(TObject *Sender)
{
    Label3->Caption = ComboBox3->Items->
                           Strings[ComboBox3->ItemIndex];
}

// нажатие клавиши в поле редактирования компонента ComboBox2
void __fastcall TForm1::ComboBox2KeyPress(TObject *Sender,
char &Key)
{
    if  (Key == VK_RETURN)
    {
```