Часть 1. Приложение на VB 4.0 за пять минут

Классическим примером первого приложения на языке C является программа, которая выводит на экран приветствие "Hello, World!" ("Привет, мир!"). Попробуем реализовать эту задачу на VB (любой версии). Для начала можно предложить следующий вариант:

- запустить среду VB;
- дважды щелкнуть форму нового проекта;
- в появившемся окне кода в процедуре Form_Load ввести строку: MsgBox "Привет, мир!";
- запустить программу на выполнение.

У меня вся эта операция заняла 30 секунд, из них половина ушла на загрузку VB 4.0 (Windows 95, Pentium 90, 16 Мб ОЗУ). Конечно, от такого приложения не очень много толку. Поэтому попробуем сделать в среде VB 4.0 что-нибудь более полезное, например текстовый редактор.

В начало статьи

Создание текстового редактора — базовый вариант

Шаг 1. Загружаем VB 4.0 и, щелкнув форму, сразу меняем ее название (свойство Caption в окне Properties) на "Текстовый редактор". Далее, выбрав в меню Tools команду Menu Editor, формируем для начала стандартное меню "Файл" с командами "Создать", "Открыть", "Сохранить" и "Выход" (см. рис. 1).

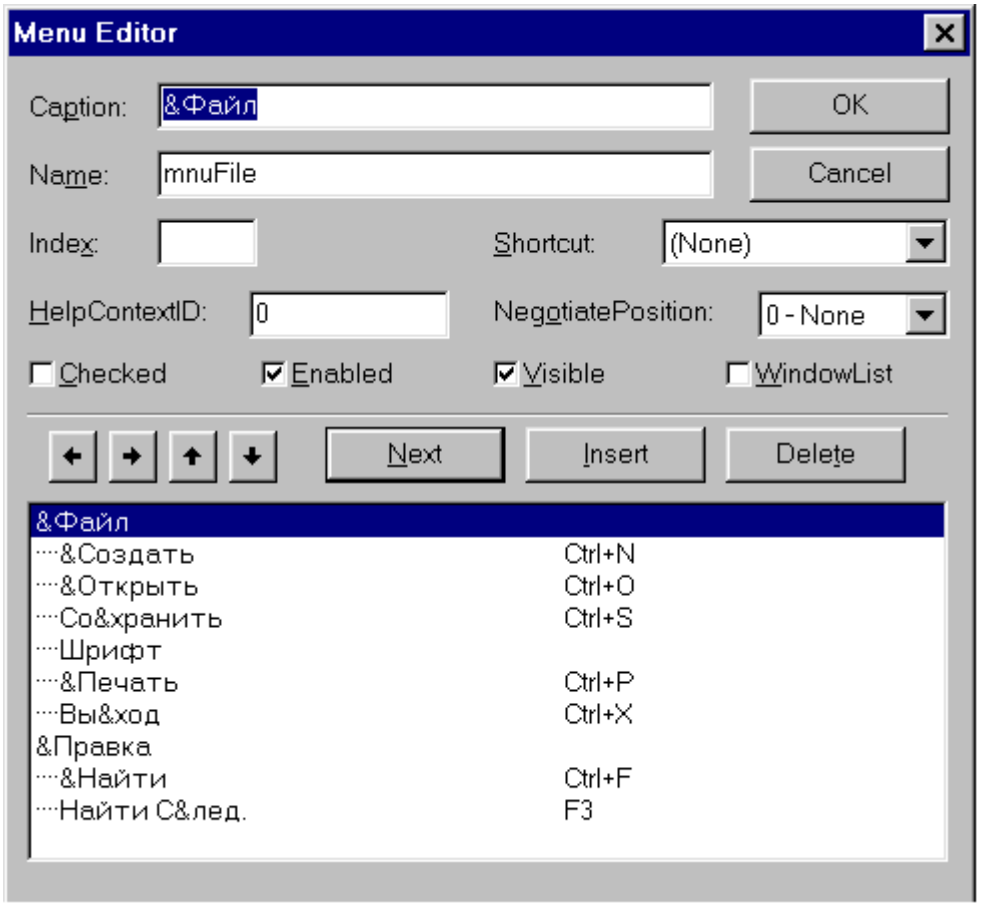

Рис. 1

Шаг 2. Размещаем на форме два элемента управления — RichTextBox (RICHTB32.OCX) и CommonDialog (COMDLG32.OCX) . Они оба являются дополнительными, то есть их надо сначала включить в панель инструментов (будем считать, что это уже сделано). Сам редактор текста будет

сделан на основе RichTextBox. В отличие от простого TextBox этот элемент управления может также работать с форматированным текстом. Кроме того, используя методы LoadFile и SaveFile, он читает и записывает не только TXT-, но и RTF-файлы. К тому же при работе с RichTextBox можно использовать загрузку данных просто с помощью метода drag-and-drop. Элемент управления CommonDialog пригодится сразу для нескольких операций в нашем приложении (см. приложение "Тестирование CommonDialog"), в частности для чтения и записи файлов.

Шаг 3. Установите для RichTextBox свойство Scrollbars = "Both".

ВНИМАНИЕ! *Модуль RICHTB32.OCX в последней бета-версии VB 5.0 содержал ошибку. Для вариант "Both" он устанавливал обе линейки прокрутки, но строка текста не могла выйти за правую границу окна — производился автоматический переход на новую строку. В окончательном варианте VB 5.0 эта ошибка исправлена.*

Шаг 4. Операции инициализации и завершения. Выберите на нашей форме в меню "Файл" команду "Создать". В появившемся окне кода в процедуре mnuNew_Click (названия процедур должны соответствовать именам элементов управления меню — поле Name в диалоговом окне Menu Editor) введите строку для начальной очистки текста:

```
RichTextBox1 = ""
```

Аналогичным образом задайте операцию завершения приложения ("Файл", "Выход", mnuExit_Click):

```
Unload Me: End
```

Шаг 5. Пора заняться операциями чтения-записи файлов. Щелкнув правой клавишей мыши изображение CommonDialog, выберите в меню команду Properties. В появившемся диалоговом окне во вкладе Open/Save установите два поля:

```
DefaultExt = "txt"
Filter = "Текстовые файлы|*.txt|Форматированные файлы| _
  *.rtf|Все файлы|*.*"
```

Как и раньше, щелкая мышью команды меню на нашей форме, заполните процедуры mnuOpen_Click и mnuSave_Click (далее — см. **код в листинге 1**).

СОВЕТ. *Многие строки кода нашего приложения будут очень похожи друг на друга. Их проще создавать путем копирования уже набранных, а потом исправления отдельных слов. Поэтому будет удобнее, если весь код приложения будет последовательно размещаться в одном окне Code. Для этого в меню Tools выберите команду Options, а потом во вкладке Editor установите флажок Full Module View.*

Мы уже создали первый вариант текстового редактора: он может создавать новый текст, корректировать старые файлы, а результаты работы записывать на диск. Попробуйте. При этом можно работать и с командами меню, и с "горячими" клавишами, а также читать исходные данные через механизм drag-and-drop.

В начало статьи

Поехали дальше

Шаг 6. Воспользуемся возможностью работы с форматированным текстом. Включим с помощью Menu Editor в меню Файл нашей формы еще одну команду — "Шрифт", а потом уже испробованным способом откроем окно кода и сформируем процедуру mnuFont_Click. Обратите внимание, что здесь потребуется установить константы CommonDialog, которые управляют режимами работы данного окна.

Шаг 7. Теперь аналогичным образом добавим команду "Печать" в меню "Файл" и сформируем процедуру mnuPrint_Click. Однако здесь нужно обратить внимание на один важный момент. На самом деле мы не можем непосредственно распечатать текст из RichTextBox на выбранном устройстве. Сначала нужно передать соответственно подготовленный текст объекту Printer, который потом выдаст его на печать.

ВНИМАНИЕ! *Во всех руководствах по VB4/5 в качестве примера распечатки текста приводится такой код:*

```
CommonDialog1.ShowPrinter
RichTextBox1.SelPrint CommonDialog1.hDC
```

Однако у меня ничего на печать не выдавалось. Все стало работать, когда я заменил последнюю строку на три такие:

```
Printer.Print ""
RichTextBox1.SelPrint (Printer.hDC)
Printer.EndDoc
```

Действительно, в описании VB говорится, что в любом случае объект Printer должен быть проинициализирован — например можно просто выдать пустую строку. Команда Printer.EndDoc не является обязательной — в случае ее отсутствия распечатка выполнится в момент закрытия приложения. А вот команда RichTextBox1.SelPrint (Printer.hDC) нужна всегда — именно она упоминается в описании метода SelPrint.

Шаг 8. Давайте добавим еще и функции поиска текста. В дополнение к меню "Файл" сделаем еще меню "Правка", включим в него две команды - "Поиск" и "Поиск След." — и запишем соответствующий код в процедуры mnuFind_Click и mnuFindNext_Click.

ВНИМАНИЕ! *Если у вас не была включена опция Option Explicit, то скорее всего вы увидите, что команда "Поиск След." не работает. Дело в том, что под именем sFind в обеих процедурах будут созданы две разные локальные переменные. Если Option Explicit была установлена, что при запуске будет выдано сообщение о неопределенных переменных и вы догадаетесь, что нужно определить одну глобальную переменную Public sFind в главной части модуля.*

Вот и все — мы создали довольно приличный текстовый редактор. Возможно, в первый раз вам потребовалось на это минут 10-15, но со второй-третьей попытки вы легко уложитесь в обещанные пять минут (рис. 2).

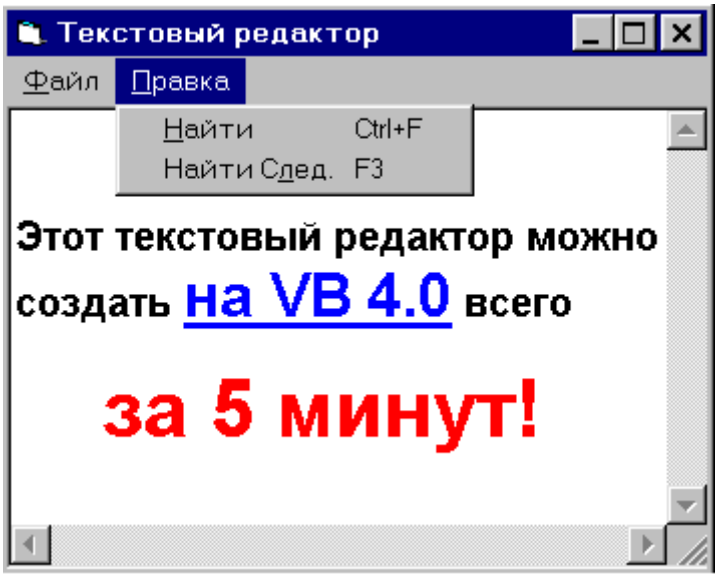

Рис. 2

В начало статьи

Приложения к Части 1

В начало статьи

Листинг 1. Код приложения — текстового редактора

Из пяти минут работы большая часть времени ушла именно на ввод кода (строки Sub/End Sub формируются автоматически).

```
Option Explicit
Public sFind As String

Private Sub mnuExit_Click()
    ' Завершение работы приложения
    Unload Me: End
End Sub

Private Sub mnuFind_Click()
    ' Запрос на поиск
    sFind = InputBox("Какое слово искать?", , sFind)
    ' Поиск слова по всему тексту
    RichTextBox1.Find sFind
End Sub

Private Sub mnuFindNext_Click()
    ' Передвигаем текущую позицию курсора
    RichTextBox1.SelStart = RichTextBox1.SelStart _
            + RichTextBox1.SelLength + 1
    ' Поиск с текущей позиции
    RichTextBox1.Find sFind, , Len(RichTextBox1)
End Sub

Private Sub mnuFont_Click()
    ' Установка шрифтов
    '
    ' Установка флагов для диалогового окна Font
    ' cdlCFBoth — шрифты принтера и экрана
    ' cdlCFEffects — полный набор свойств шрифтов
    CommonDialog1.Flags = cdlCFBoth + cdlCFEffects
    ' Вывод окна Font — выбор шрифтов
    CommonDialog1.ShowFont
    ' Установка в редакторе параметров шрифтов
    ' в соответствии с выбранными в окне Font
    With RichTextBox1
        .SelFontName = CommonDialog1.FontName
        .SelFontSize = CommonDialog1.FontSize
        .SelBold = CommonDialog1.FontBold
        .SelItalic = CommonDialog1.FontItalic
        .SelStrikethru = CommonDialog1.FontStrikethru
        .SelUnderline = CommonDialog1.FontUnderline
        .SelColor = CommonDialog1.Color
    End With
End Sub

Private Sub mnuNew_Click()
    ' Начальная очистка тестового окна
    RichTextBox1.Text = ""
End Sub

Private Sub mnuOpen_Click()
```

```
    ' Режим открытия файла — окно Open
    CommonDialog1.ShowOpen
    ' Прочитать открытый файл в окно редактора
    RichTextBox1.LoadFile (CommonDialog1.filename)
End Sub

Private Sub mnuPrint_Click()
    ' Вывод на печать
    '
    ' Установка флагов для вывода на печать
    CommonDialog1.Flags = cdlPDReturnDC + _
            CdlPDNoPageNums
    If RichTextBox1.SelLength = 0 Then
       ' нет выделенного текста
       CommonDialog1.Flags = CommonDialog1.Flags _
           + cdlPDAllPages
    Else
       ' есть выделенный текст
       CommonDialog1.Flags = CommonDialog1.Flags _
           + cdlPDSelection
    End If
    ' Вывод диалогового окна Print
    CommonDialog1.ShowPrinter
    ' Вывод на печать
    Printer.Print ""
    'RichTextBox1.SelPrint CommonDialog1.hDC
    RichTextBox1.SelPrint (Printer.hDC)
    Printer.EndDoc
End Sub

Private Sub mnuSave_Click()
    ' Режим сохранения файла — окно Save
    CommonDialog1.ShowSave
    ' Сохранить созданный файл
    RichTextBox1.SaveFile (CommonDialog1.filename)
End Sub
```

В начало статьи

Тестирование CommonDialog

Элемент управления CommonDialog является весьма многофункциональным. С помощью шести методов обращения к нему — ShowOpen, ShowSave, ShowPriner, ShowFont, ShowColor, ShowFont — можно получить доступ к различным очень полезным операциям в виде диалоговых окон. При этом на форме достаточно установить всего один элемент управления (он не виден при выполнении приложения).

Свойствами CommonDialog можно управлять программным образом или в среде VB. В последнем случае достаточно щелкнуть правой кнопкой мыши его изображение и выбрать команду Properties. Кроме того, режимы работы конкретных окон задаются с помощью набора флагов, устанавливаемых в коде программы. Их описание приведено в Справочной системе VB.

Для освоения CommonDialog может быть полезно следующее небольшое тестовое приложение. Разместите на форме три элемента управления - Command, Option, CommonDialog. Для Option установите свойство Index равным 0. В процедуры Form_Load и Command_Click запишите код, представленный **в листинге 2**. После этого, устанавливая поочередно разные переключатели и щелкая мышью командную кнопку, выводите на экран разные диалоговые окна (см. рис. 3). Попробуйте поуправлять их режимами, задавая различные значения свойств и флагов CommonDialog.

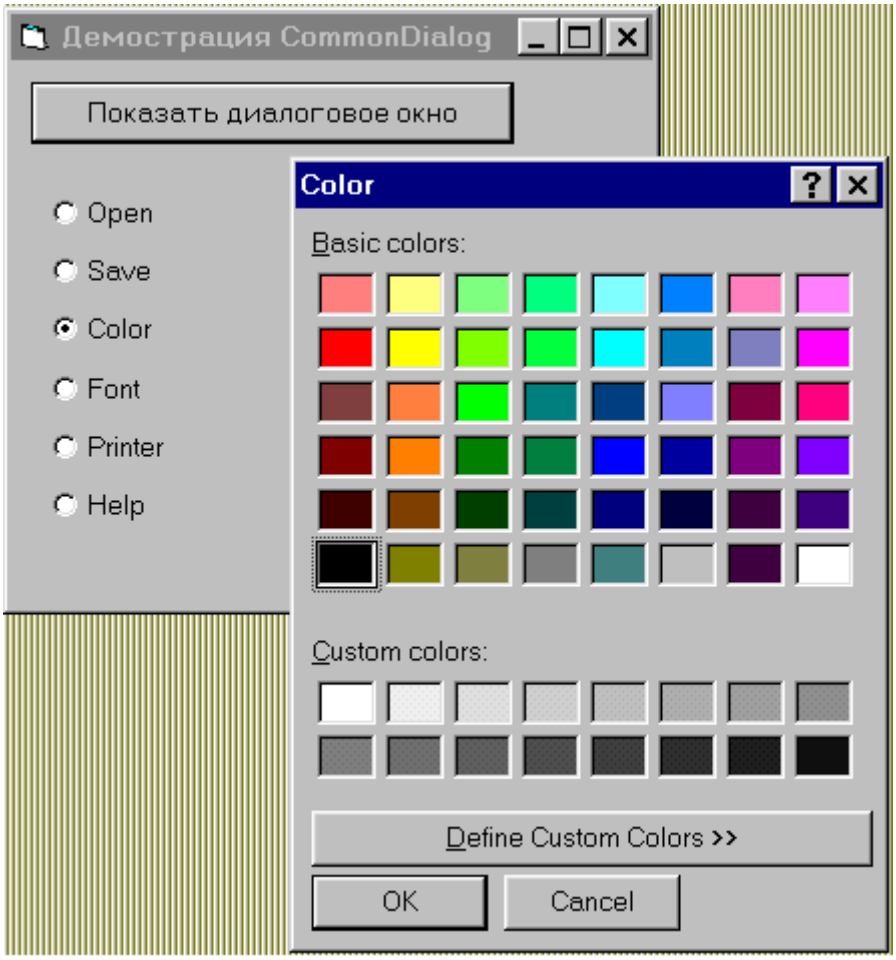

Рис. 3

Листинг 2. Код теста для изучения элемента управления CommonDialog

```
Option Explicit

Private Sub Form_Load()
  Static FlagFormPainted As Integer
  Dim i As Integer
  ' При первом запуске — сделать разметку формы
  If FlagFormPainted <> True Then
    For i = 1 To 5  ' установить массив переключателей
      Load Option1(i)
      Option1(i).Top = Option1(i - 1).Top + 350
      Option1(i).Visible = True
    Next I
    ' Сделать надписи на  каждом переключателе
    Option1(0).Caption =  "Open"
    Option1(1).Caption =  "Save"
    Option1(2).Caption =  "Color"
    Option1(3).Caption =  "Font"
    Option1(4).Caption =  "Printer"
    Option1(5).Caption =  "Help"
    Command1.Caption = "Показать диалоговое окно"
  End If
  FlagFormPainted = True
End Sub

Private Sub Command1_Click()
```

```
  If Option1(0).Value Then commonDialog1.ShowOpen
  If Option1(1).Value Then commonDialog1.ShowSave
  If Option1(2).Value Then commonDialog1.ShowColor
  If Option1(3).Value Then
    commonDialog1.Flags = cdlCFBoth
    commonDialog1.ShowFont
  End If
  If Option1(4).Value Then commonDialog1.ShowPrinter
  If Option1(5).Value Then
    commonDialog1.HelpFile = "VB4.HLP"
    commonDialog1.HelpCommand = cdlHelpContents
    commonDialog1.ShowHelp
  End If
End Sub
```

В начало статьи

Часть 2. Приложение на VB 5.0 за три минуты

Теперь попробуем решить примерно ту же задачу — создать собственный текстовый редактор с помощью VB 5.0. Одним из достоинств новой версии является большой набор мастеров и утилит для создания различных заготовок. В меню Add-Ins есть команда Add-In Manager, с помощью которой можно познакомиться со списком этих средств, а также подключить их к среде разработки.

Скорее всего, одним из наиболее применяемых мастеров будет Application Wizard, который помогает создать законченное приложение, включающее главную форму, меню, панель инструментов, строку состояния и несколько вспомогательных форм, а также набросок кода нового приложения. Именно поэтому значок Application Wizard включен по умолчанию в окно создания нового проекта. Итак, начинаем — засекайте время работы.

В начало статьи

Создание заготовки приложения

Шаг 1. Запускаем VB 5.0 и в окне New Project (рис. 4) дважды щелкаем VB Application Wizard, после чего появляется его начальное окно. Для полного использования мастера нужно последовательно нажимать кнопку Next и в появляющихся окнах устанавливать разные параметры будущего приложения. (По умолчанию создается приложение с интерфейсом типа Multiple Document Interface (MDI) — это как раз то, что нам нужно для данной задачи.)

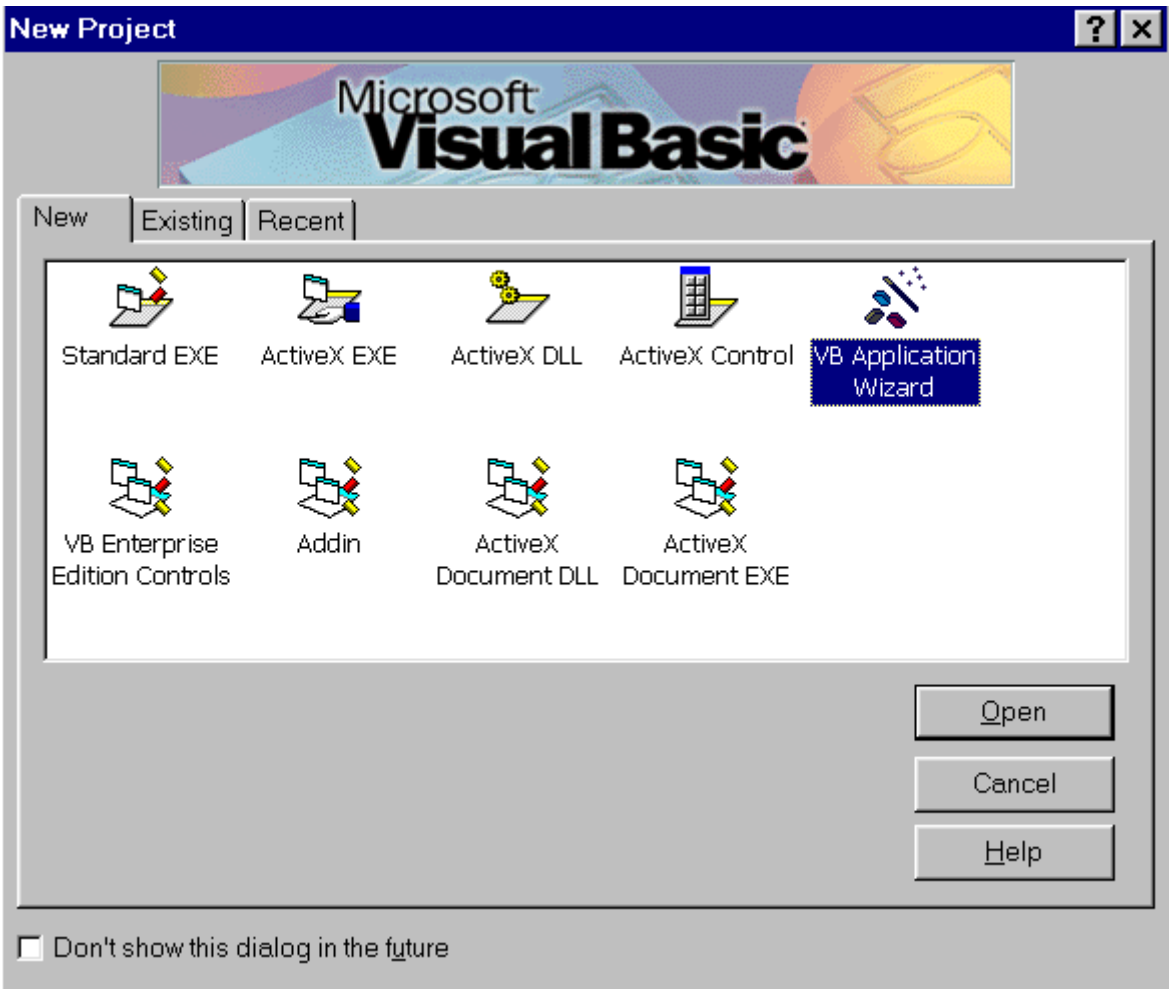

Рис. 4

Но детальное изучение возможностей Application Wizard сейчас не входит в нашу задачу, поэтому сразу нажмите кнопку Finish - мастер начнет создавать заготовку приложения с параметрами, заданными по умолчанию. Эта операция займет около 10 секунд и закончится выдачей окна с отчетом о проделанной работе, который вы можете при желании сохранить в виде файла. В результате всех этих операций вы создали проект с именем Project1 (это имя лучше сразу поменять на что-то более конкретное, например на TextEdit5), который имеет две формы (frmMain и frmDocument) и один модуль (на этот раз он нам не пригодится).

Шаг 2. Нажав F5, запустите проект на выполнение, и вы увидите симпатичное приложение с полным меню, панелью инструментов и строкой состояния, на которой показаны дата и время, а также MDI-дочерним окном с заголовком "Document 1".

С этим приложением уже можно поработать. Например, ввести текст в окне документа. Или создать новый документ, выбрав к меню File команду New - появится еще одно окно "Document 2". Это происходит потому, что окно frmDocument является MDI-дочерней формой, которую можно использовать для создания дополнительных документов. Мастер Application Wizard уже добавил нужный код для команды File/New в виде процедуры LoadNewDoc. В вашем распоряжении также команды меню Windows. С их помощью можно, например, устанавливать различные типы многооконного интерфейса - каскад, вертикальное и горизонтальное деление экрана.

Шаг 3. Но все же толку от такого приложения не очень много. Если вы попробуете сохранить свой документ, выбрав команду Save в меню File, то на экран выведется сообщение: "Save Code goes here!" ("Здесь должен быть код для команды Save!"). Что ж, Application Wizard — не волшебник. Он создает приложение, но в большинстве случаев не записывает код процедур, которые соответствуют тем или иным командам, помечая эти места в коде формы комментариями, начинающимися со слов "To Do" ("Сделать"). Наличие такой строки означает, что мастер

предлагает вам заполнить эти места своим кодом. Обычно подобные места сопровождаются строкой сообщения типа MsgBox "Save Code goes here!" для предупреждения пользователя в явном виде.

В данном случае весь код приложения связан с операциями, выполняемыми в форме frmMain. Выберем ее в окне Project и щелкнем View Code. Теперь нажмите клавиши Ctrl-F и введите "To Do". Щелкая кнопку Find Next, вы увидите все элементы "To Do", которые вам предлагается заполнить кодом, — в нашем проекте их должно быть 25. Но в данном примере мы используем только некоторые из них.

Прежде чем начать доводку нашего приложения до рабочего состояния, рассмотрим повнимательнее две формы полученного проекта (рис. 5). Форма frmMain имеет элементы управления Toolbar и ImageList, а также StatusBar и CommonDialog. В ней мы ничего менять не будем, кроме ее имени, — нажмите F4 и в окне Properties измените свойство Name на "Текстовый редактор — VB5". frmDocument содержит один элемент управления Textbox, который мы заменим на RichTextBox (о его преимуществах мы уже говорили в первой части статьи).

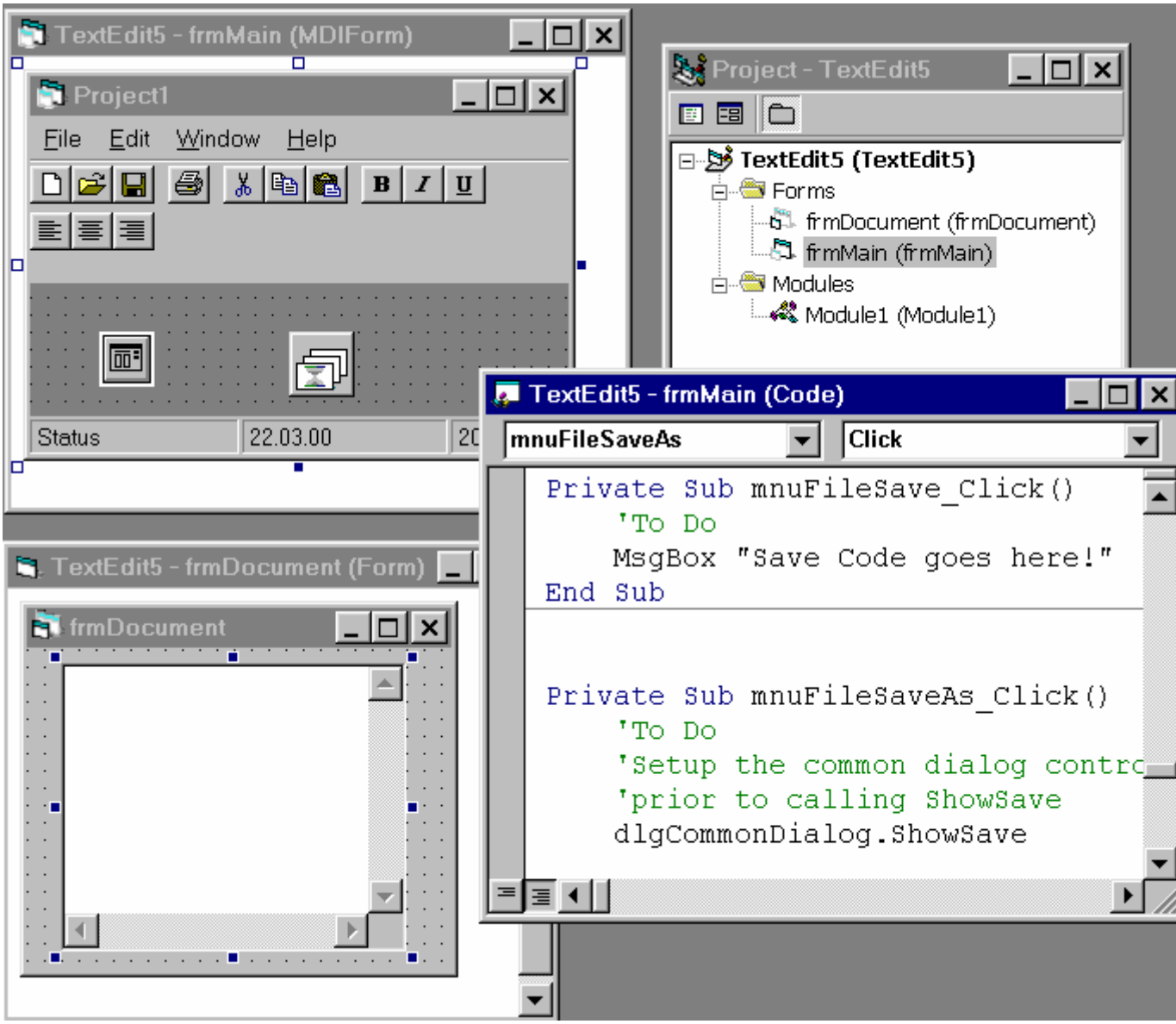

Рис. 5

Если этот дополнительный элемент управления еще не установлен в вашем Toolbox, то щелкните правой кнопкой эту панель инструментов. В появившемся диалоговом окне Components найдите

Microsoft RichTextBox Control 5.0 и отметьте его, чтобы добавить к панели Toolbox. Затем удалите элемент управления Textbox с формы frmDocument (выделите щелчком мыши и нажмите Del) и установите на его место RichTextBox. Измените значение его свойства Name на txtText (это нужно сделать для совместимости с уже готовым кодом) и установите свойство ScrollBars равным rtfBoth.

В начало статьи

Написание кода для меню

Шаг 4. Практически весь код вашего проекта основывается на форме frmMain, где располагаются меню и панель инструментов. Выведите ее на экран, щелкнув правой кнопкой мыши название frmMain в окне Project, а затем щелкните View Object.

Теперь выберите команду Open в меню File, и вы увидите в окне кода следующий текст:

```
Private Sub mnuFileOpen_Click()
    Dim sFile As String
    With dlgCommonDialog
        ' To Do
        ' Установка флажков и атрибутов
        ' элемента управления CommonDialog
        .Filter = "All Files (*.*)|*.*"
        .ShowOpen
        If Len(.filename) = 0 Then
                Exit Sub
        End If
        sFile = .filename
    End With
    ' To Do
    ' Обработка открытого файла
End Sub
```

Вместо последних строк комментария введите следующий код, который производит загрузку выбранного файла в элемент управления RichTextBox активной формы документа и меняет название документа на полное имя файла:

```
ActiveForm.txtText.LoadFile (sFile)
ActiveForm.Caption = sFile
```

Шаг 5. Аналогичным образом запишем код для команды Save:

```
If Left(ActiveForm.Caption, 8) = "Document" Then
    mnuFileSaveAs_Click
Else
    ActiveForm.txtText.SaveFile (ActiveForm.Caption)
End If
```

Конструкция If нужна для проверки заголовка документа: если он начинается с "Document", то, наверное, пользователь еще не присвоил файлу никакого имени. Строку с сообщением MsgBox... из процедуры mnuFileSave_Click нужно убрать.

Шаг 6. Для команды Save As запишем в процедуре mnuFileSaveAs_Click следующий код:

```
With dlgCommonDialog
    .Filter = "All Files (*.*)|*.*"
    .ShowSave
    If Len(.filename) = 0 Then Exit Sub
    ActiveForm.Caption = .filename
End With
mnuFileSave_Click
```

СОВЕТ. *Фрагмент этого кода можно скопировать из mnuFileOpen_Click, а потом откорректировать его.*

Шаг 7. Написание кода для Save All можно оставить пока на потом, для домашнего задания. Для его выполнения воспользуйтесь коллекцией Forms для просмотра всех форм frmDocument.

Активизация кнопок панели инструментов

Шаг 8. Откройте окно кода, щелкнув дважды панель инструментов на форме вашего приложения, и найдите там процедуру tbToolBar_ButtonClick. Чтобы вводить в ней поменьше текста, сразу заключите весь ее код в скобки конструкции With ActiveForm.txtText ... End With.

Шаг 9. Элемент управления RichTextBox имеет несколько свойств, которые начинаются с "Sel..." и управляют форматированием выделенного текста. Например, свойство SelBold возвращает значение "Истина" (True), если шрифт выделенного текста является полужирным (Bold).

Для активизации кнопок форматирования добавьте в каждый оператор Case события tbToolBar_ButtonClick следующий код:

```
Case "Bold"
    .SelBold = Not .SelBold
Case "Italic"
    .SelItalic = Not .SelItalic
Case "Underline"
    .SelUnderline = Not .SelUnderline
```

Аналогичную операцию сделайте для кнопок выравнивания:

```
Case "Left"
    .SelAlignment = rtfLeft
Case "Center"
    .SelAlignment = rtfCenter
Case "Right"
    .SelAlignment = rtfRight
```

В начало статьи

Команды копирования

Шаг 10. Свойство SelText используется также для копирования фрагментов текста в сочетании с методами GetText и SetText объекта Clipboard. В нашем приложении для этого предусмотрены команды Cut, Copy, Paste в меню Edit. Функция Copy просто помещает выделенный текст в буфер обмена. Cut вначале вызывает функцию Copy, чтобы поместить текст в буфер обмена, а затем превращает выделенный текст в пустую строку. Наконец, функция Paste вставляет текст из буфера обмена в элемент управления RichTextBox в точку, где установлен курсор, заменяя там любой выделенный текст. Аналогичные кнопки на панели инструментов используют обращение к тем же событийным процедурам.

Для выполнения этих операций нужно ввести такой код:

```
Private Sub mnuEditCopy_Click()
    Clipboard.SetText ActiveForm.txtText.SelText
End Sub

Private Sub mnuEditCut_Click()
    mnuEditCopy_Click
    ActiveForm.txtText.SelText = ""
End Sub
```

```
Private Sub mnuEditPaste_Click()
    ActiveForm.txtText.SelText = Clipboard.GetText
End Sub
```

Как и в первом примере, на освоение приведенных здесь пояснений у вас наверняка ушло больше обещанных трех минут, но уже со второго раза вы вполне сможете уложиться в это время.

Данный вариант редактора по ряду функций уступает тому, который мы создали средствами VB 4.0, но при этом он выполняет ряд новых операций. Но самое главное — он создает настоящий многооконный редактор (рис. 6)! Конечно, для доводки его до состояния законченного приложения предстоит еще многое сделать: убрать ненужные команды, добавить необходимые, перевести на русский язык стандартные названия команд и сообщений... Это потребует дополнительного времени — может быть, даже целого часа.

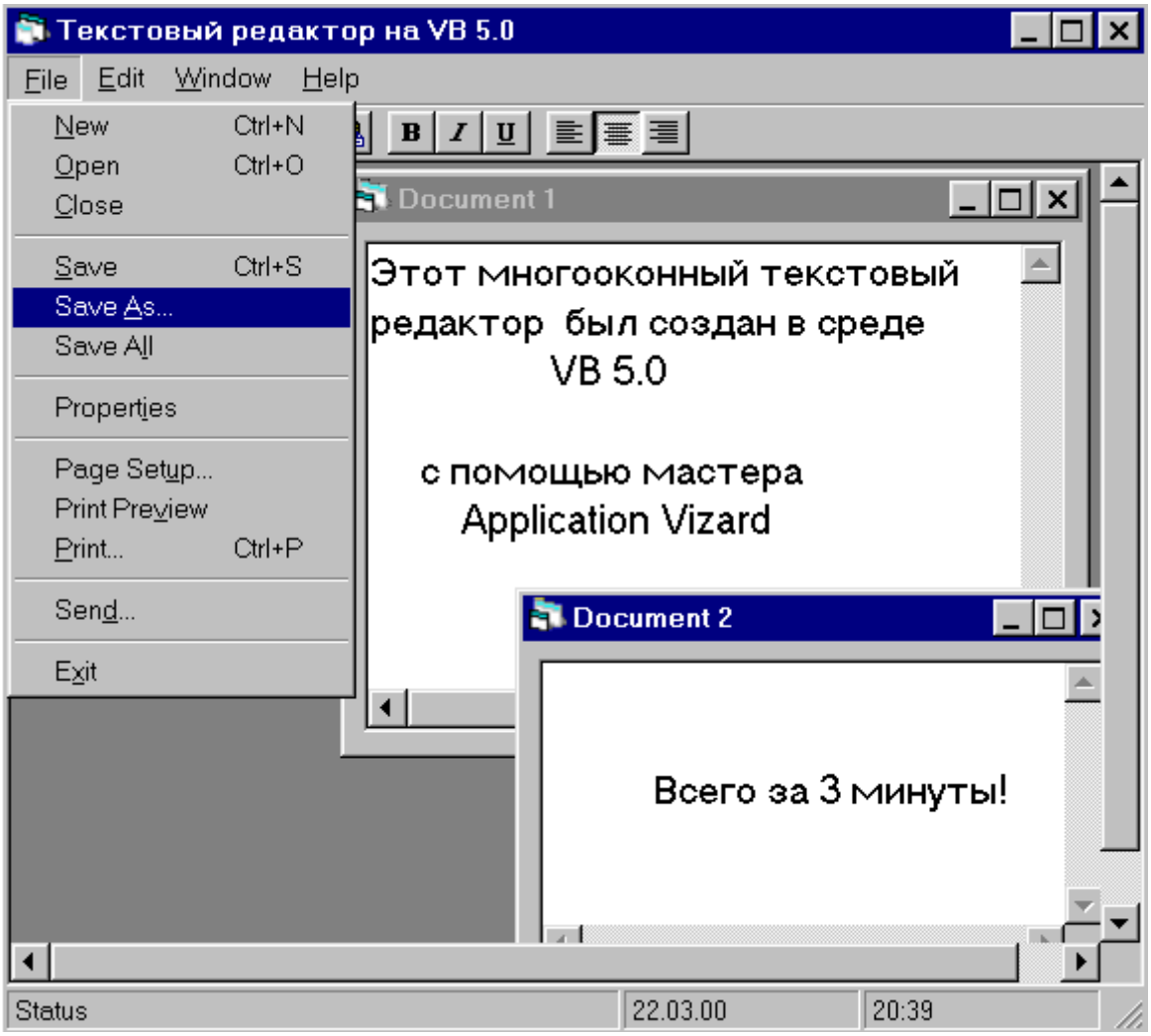

Рис. 6

Если же вы собираетесь пользоваться мастером Application Wizard достаточно часто, то можно реализовать такую идею — написать утилиту для автоматической замены английских названий на русские: Edit — "Правка", "Save Code goes here!" — "Здесь нужно записать Код команды Сохранить" и т.п. Кроме того, можно отредактировать список необходимых команд в исходном тексте.