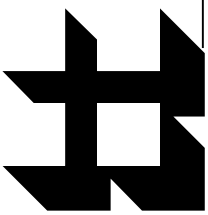

ГЛАВА 4

Формы

Форма как основной компонент приложения

Приложения Windows, использующие графический интерфейс пользователя (GUI), для отображения информации используют окна. Окном называется прямоугольная область на экране монитора, связанная с приложением. События (Events), такие как перемещение или изменение размеров окна, щелчок мышью на области экрана, занимаемой окном, трансформируются в сообщения, посылаемые Windows этому окну. Если окно активно, то ему также направляются сообщения о событиях, происходящих при нажатии клавиш на клавиатуре.

Программируя на VFP, мы не создаем окна непосредственно; мы используем объекты, которые и выполняют всю необходимую работу по созданию окон и обработке получаемых ими сообщений Windows. В VFP такие объекты создаются из базового класса `Form`, они и называются формами.

Форма — объект, предназначенный для ввода данных, отображения их на экране или управления работой приложения. Объекты `Form`, представляющие экономические, бухгалтерские документы, на экране могут выглядеть как стандартные бланки этих документов.

Формы как объекты предоставляют разработчику множество свойств, позволяющих управлять внешним видом окна, например, шириной (Width), высотой (Height) и т. п. Кроме того, формы включают в себя методы (обработчики некоторых сообщений Windows), например, сообщения о событии `Paint`, происходящем при создании или перерисовке окна, о событиях `Click`, `DblClick` и `RightClick`, вызванных щелчком мыши на форме. Объект `Form` предоставляет так же ряд методов, непосредственно не связанных с окном и расширяющих его собственную функциональность, например, методы `Cls`, `Move`, `Draw`, `Release`. Но самое важное — это то, что объект `Form` является контейнером, позволяющим на созданном им окне размещать различные элементы управления, что позволяет организовать интерактивный диалог с пользователем. Это и делает форму основным компонентом приложения. Объектами формы могут яв-

ляться сетка (grid), флажок (checkbox), кнопка (Command Button), текстовое поле (Text Box), редактируемое поле (Edit Box), наборы кнопок и переключателей (Command Group, Option Group), список (List Box), выпадающий список (ComboBox), счетчик (Spinner), список (ListView), дерево (TreeView), набор вкладок (PageFrame) и т. д. В форме могут располагаться графические объекты и поля типа General.

Любое приложение Windows, поддерживающее GUI, имеет главное окно и, возможно, ряд дополнительных окон, являющихся дочерними по отношению к главному окну. В VFP все дочерние окна также создаются объектами Form. Как наследие долгого жизненного цикла, в VFP существуют объекты FormSet (наборы форм). Formset — это объект-контейнер, поэтому ряд свойств контейнера распространяются на вложенные объекты. В частности, такими свойствами являются:

- свойство WindowType. Если Formset является модальным, то модальными будут и все входящие в его состав формы;
- объект DataEnvironment. Он является общим для всех форм, входящих в набор форм.

Мы не будем пока касаться объекта DataEnvironment, поскольку он будет описан в *главе 9*.

VFP, как интегрированная интерактивная среда разработки приложений, позволяет формам вашего приложения выполняться в собственном главном окне. Основной недостаток такого способа построения приложения — необходимость постоянно "убирать" при запуске вашего приложения всякие компоненты среды разработки: окно Менеджера проектов, панели инструментов, иные служебные окна, а так же заменять стандартное меню VFP на меню вашего приложения. Современный стиль программирования заключается в том, что среда разработки должна использоваться по прямому назначению — для разработки и отладки приложений, а само приложение должно выполняться в собственном главном окне, а главное окно VFP просто "убирается", т. е. делается невидимым (хотя на самом деле оно продолжает существовать).

Для объявления окна формы главным вы должны присвоить свойству ShowWindow значение 2 — As Top-Level form (форма верхнего уровня). Такая форма не имеет родительского окна. Самое интересное — это то, что VFP позволяет в одном приложении иметь произвольное количество форм, объявленных как формы верхнего уровня. С одной стороны, это хорошо, но злоупотреблять этим не следует.

Особенностью формы верхнего уровня является также возможность включения в нее строки меню. В принципе, можно встроить меню в любую форму, но корректно оно будет работать только в формах верхнего уровня и формах типа DeskTop, о которых речь пойдет ниже.

Для форм, выполняющихся в главном окне VFP, значение свойства ShowWindow равно нулю. Имейте в виду, что такая форма, будучи запущена из формы верхнего уровня, будет отображаться в окне VFP — т. е. если это окно не скрыто, то форма верхнего уровня "свернется", и вы увидите главное окно VFP с запущенной формой. В случае, когда главное окно VFP невидимо, такая форма так же не сможет быть нарисована на

экране — она будет создана, но у Windows не будет возможности нарисовать ее, т. к. ее родительское окно скрыто. Поэтому все формы, запускаемые из формы верхнего уровня, должны иметь значение свойства `ShowWindow`, равное 1 (`In top level form` — выполняется в форме верхнего уровня).

Каждая форма VFP имеет еще три важных свойства — `Desktop`, `AlwaysOnTop` и `AlwaysOnBottom`. Если обычные формы видимы только в пределах главного окна, то установка в истину свойства `Desktop` позволяет дочерней форме "выходить за рамки" родительского окна. Кроме того, такие формы могут содержать строку меню. Установка в истину свойства `AlwaysOnTop` заставляет форму держаться "наверху", т. е. она будет перекрывать все другие формы, для которых это свойство не установлено. Соответственно, установка в истину свойства `AlwaysOnBottom` заставляет форму располагаться под всеми другими окнами, не перекрывая их.

Модальные и немодальные формы. Форма верхнего уровня

Модальность

Все формы VFP, кроме форм верхнего уровня, обладают свойством, которое называется модальностью (для форм верхнего уровня значение этого свойства игнорируется).

Немодальные формы работают независимо друг от друга; вы можете запустить несколько немодальных форм и переходить с одной из них на другую, делая такую форму активной простым щелчком мыши на ней (или на одном из элементов управления, расположенном на такой форме). Только одна из нескольких одновременно выполняющихся немодальных форм может быть активной!

Модальные формы ведут себя иначе: они "перехватывают" все сообщения Windows и не дают возможности перейти на любую другую форму приложения (включая главную форму), пока такая форма не будет закрыта. Обычно такие формы называют диалогами или диалоговыми окнами; их цель — запросить от пользователя некие действия (например, ввод данных) и дождаться, пока пользователь не выполнит эти действия и не закроет форму. Управляет модальностью свойство `WindowType`; его установка в 1 заставляет форму стать диалогом. VFP позволяет одну и ту же форму запускать как модально, так и немодально — управлять модальностью можно при помощи метода `Show` объекта `Form`.

Типы интерфейса. SDI или MDI?

В настоящее время для приложений, разрабатываемых в среде Windows при помощи Visual FoxPro, используется три типа интерфейса: однодокументный *SDl (Single-Document Interface)*, многодокументный *MDI (Multiple-Document Interface)* и интерфейс типа проводник *(Explorer).* В этой главе основное внимание будет уделено пер-

вым двум типам интерфейса, поскольку они наиболее часто применяются для разработки пользовательских приложений.

Однодокументный интерфейс — это тип интерфейса, в котором предоставляется возможность работы только с одним документом в одном окне. Примером может служить редактор Microsoft WordPad или Paint. Эти приложения позволяют редактировать только один документ. Для работы с несколькими документами в таком интерфейсе необходимо многократно запускать приложение. Для каждого типа данных и документов требуется своя форма и, соответственно, свое приложение с интерфейсом типа SDI. В принципе это тоже один из возможных вариантов, но он подходит только для работы с небольшим количеством форм документов. При загрузке большого количества SDI-приложений начинает переполняться оперативная память компьютера, и приложения работают очень медленно. Каждый раз при запуске SDI-приложения в память загружаются одни и те же данные (меню, панель и элементы управления), выполняющие одинаковые действия, что приводит к неэффективной и медленной работе запускаемых приложений.

Однако полностью отказываться от интерфейса типа SDI не стоит, поскольку он вполне годится для работы с одним или двумя документами (например, для копирования из одного документа в другой).

Многодокументный интерфейс (MDI) — это стандарт интерфейса для приложений Windows, которые позволяют пользователю одновременно работать с несколькими открытыми документами. Документ, в этом смысле, это обычно связанная с файлом задача, например, редактирование текстового файла или работа с файлом электронной таблицы. В приложениях MDI пользователь может, например, иметь несколько открытых файлов в одном приложении. Возможно, что вы уже использовали приложения MDI: Microsoft Excel, администратор программ Windows, администратор файлов Windows.

Обычно MDI-приложения состоят минимум из двух форм — родительской и дочерней. Родительская форма служит контейнером, содержащим дочерние формы, которые заключены в клиентскую область и могут перемещаться, изменять размеры, минимизироваться или максимизироваться. В вашем приложении могут быть дочерние формы разных типов, например, одна — для обработки изображений, а другая — для работы с текстом. Важной особенностью окон MDI-приложения является их возможность "распахиваться" на всю главную форму; при этом заголовок такого окна "исчезает", и если окно имело строку меню, то его пункты добавляются в главное меню приложения, а в правой части строки меню появляются кнопки "свернуть", "развернуть" и "закрыть" для этого окна.

VFP в явном виде не поддерживает концепции SDI и MDI — все окна в нем создаются только объектом `Form`; тем не менее он позволяет "имитировать" рассмотренные концепции построения приложения. Если вы хотите создать на VFP приложение, напоминающее приложение MDI, то для тех форм, которые будут эмулировать работу с документами, нужно установить в истину свойство `MDIForm`; естественно, такие формы должны быть немодальными.

Диалоги

Если ваше приложение не отвечает концепции документ-представление, то оно относится к семейству диалогов. Примерами таких приложений являются Калькулятор или Проводник. Фактически все формы VFP также являются диалогами, потому что для работы с документами (таблицами) они используют элементы управления — а наличие в окне элементов управления (текстовых полей ввода, кнопок и т. д.) и определяет его как диалоговое окно. Отметим, что в классическом программировании только диалоговым окнам присуща модальность.

Как уже говорилось, модальная диалоговая форма "перехватывает" все сообщения Windows. Если ваше приложение использует в качестве главного окна форму верхнего уровня, и вы запускаете модальную форму, свойство `WindowType` которой равно нулю (т. е. форма может выполняться только в главном окне VFP), то, если главное окно VFP невидимо, это повлечет зависание приложения — Windows не удастся нарисовать такую форму, в то же время невидимая модальная форма не сможет быть закрыта, потому что закрыть ее может только пользователь, а как нажать на невидимую кнопку?

Свойства, события и методы экранных форм

Форма, как уже было показано выше, представляет собой контейнер, в который "складываются" другие объекты. Она отличается своими особыми "правилами поведения", т. е. свойствами, событиями и методами. Свойства, события и методы характерны не только для форм, но и для всех объектов Visual FoxPro.

Список наиболее важных и часто используемых событий формы приведен в *приложении 4*.

Свойства, методы и события формы сгруппированы в окне **Properties**, которое можно вызвать из контекстного меню (рис. 4.1), которое, в свою очередь, появляется при нажатии на свободном месте формы правой кнопки мыши.

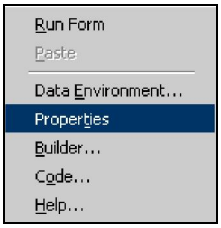

Рис. 4.1. Контекстное меню

Если дважды щелкнуть левой кнопкой мыши на тех значениях свойств, которые обозначены как `[Default]`, появляется окно ввода программного кода. Это значит, что при выполнении определенного события может быть выполнена написанная в окне программа. Именно поэтому для программиста является важным порядок выполне-

ния событий. Порядок выполнения событий для формы, имеющей открытые таблицы в Data Environment, приведен в табл. 4.1.

В процессе от момента создания и до закрытия объект Form генерирует события, сообщения о которых направляются Windows. Получив сообщение об этом событии,

Таблица 4.1. *Порядок выполнения событий в форме*

Порядок	Событие	Описание
1	BeforeOpenTable Data Environment	Самым первым выполняется событие BeforeOpenTable (то, что должно произойти перед открытием таблиц) для Data Environment
2	Form Load	Загружается форма
3	Init Data Environment	Инициализация объекта "Data Environment" осуществляется после загрузки формы, но перед ее инициализацией и перед инициализацией любого из объектов управления формы. Это необходимо для того, чтобы была возможность связывания данных из таблиц с элементами управления в форме
4	Controls Init	Перед инициализацией формы инициализируются все элементы управления, которые в ней содержатся. Если один из элементов в процессе своей инициализации вернул значение FALSE (.F.) из своего события INIT(), оставшиеся элементы управления и сама форма не инициализируются, а вызывается событие "Destroy"
5	Form Init ()	Наступает инициализация формы. В ходе этого события можно использовать данные из таблиц. Можно устанавливать указатель записи таблиц, открытых в "Data Environment" или "Load Event" формы. Если событие INIT() вернуло значение FALSE (.F.), форма не активизируется и вызывается событие "Destroy"
6	Form Activate()	Объект становится активным
7	When() и Focus() для элементов управления формы	После завершения активизации формы выполняется событие WHEN() для первого элемента управления в форме. Если оно удовлетворяется, т. е. в нем нет кода или оно вернуло значение TRUE (.T.), форма передает фокус этому элементу. Если событие WHEN() для первого элемента не выполнилось, т. е. код возврата FALSE (.F.), Visual FoxPro проверяет событие WHEN() для следующего элемента управления. Если ни один из элементов не получил фокус, сама форма не может получить фокус

8	Другие события формы (KeyPress(), MouseDown() и др.)	После того как элемент управления получил фокус, могут наступить другие события, например, "KeyPress" или события обработки мыши. Событие VALID() вызывается перед событием LostFocus(). Только когда VALID() вернет .T. или есть код, который позволит пользователю покинуть поле, наступает событие LostFocus(). Щелчок мыши вызывает события MouseDown() и MouseUp() перед самим событием Click(). И затем только после этих трех событий наступает событие Valid(). (Если щелчок пришелся вне поля, на котором только что было управление)

Таблица 4.1 *(окончание)*

Порядок	Событие	Описание
	Закрытие формы:	
9	QueryUnload()	Происходит перед выгрузкой объекта формы
10	Destroy() формы	Объект освобождается из памяти
11	Destroy() для элементов управления формы	Освобождается память от объектов-элементов управления
12	UnLoad() формы	Если все события Destroy() элементов управления формы завершились успешно, наступает событие UnLoad() самой формы
13	CloseTables Data Environment	Закрываются таблицы
14	AfterCloseTables Data Environment	Выполняется код после закрытия таблиц и файлов
15	Destroy() Data Environment	Освобождается память от таблиц

Windows создает и регистрирует в реестре окно формы. Затем на форме размещаются необходимые элементы управления. В VFP базовые элементы управления не являются самостоятельными окнами, хотя каждый такой элемент является самостоятельным объектом; за их отображение и обработку событий отвечает объект Form. Поскольку форма является контейнером, т. е. позволяет размещать внутри себя другие элементы (они тоже могут являться контейнерами), то *событие Init всех элементов управления происходит раньше, чем событие Init формы.* Только когда все компоненты формы "рассажены по местам", объект Form генерирует событие Init. Если событие Init вернет логическую ложь (.F.), то объект вообще не может быть создан. Это касается не только самой формы, но и входящих в ее состав объектов; при этом событие Init() элементов управления происходит раньше, чем Init() самой формы. Windows, получив сообщение о событии Init, рисует окно формы на экране; после того как оно будет нарисовано, Windows посылает этому окну сообщение об активации. Если все успешно и окно становится активным, вызывается метод обработки события Activate объекта Form; если на форме размещены элементы управления, то одному из этих

элементов передается фокус ввода, после чего форма переходит в режим ожидания следующих сообщений о событиях.

При закрытии формы объект `Form` генерирует новые события, оповещая Windows о предстоящем разрушении всех размещенных на форме объектов — управляющих элементов (Destroy) и затем, после того как все объекты будут уничтожены, событие, уведомляющее Windows о необходимости закрытия окна (Unload) — по получении сообщения об этом событии Windows уничтожает окно и удаляет из реестра информацию об окне, после чего объект `Form` удаляется из памяти.

Описанная последовательность событий характерна для случая, когда формы создаются и закрываются последовательно; но может возникнуть ситуация, когда происходит закрытие одной формы и открытие другой формы в методе самой закрываемой формы. В этом случае в последовательность генерируемых событий для первой формы "вклиниваются" события второй формы:

- закрываемая немодальная форма: событие `Destroy`;
- создаваемая немодальная форма: событие `Load`;
- создаваемая немодальная форма: событие `Init`;
- закрываемая немодальная форма: событие `Unload`.

В случае, если создаваемая форма — немодальная, то, скорее всего, это не вызовет никаких проблем, если только в методах `Load` или `Init` создаваемой формы вы не открываете таблицы базы данных, а в методе `Unload` закрываемой формы не выдаете команду типа `CLOSE TABLES ALL` — т. к. метод `Unload` закрываемой формы выполняется позже метода `Load (Init)` создаваемой формы, то создаваемая форма "не обнаружит" открытых таблиц.

Но если создаваемая форма — модальная, то закрываемая форма "зависнет" в памяти (и, возможно, на экране) до тех пор, пока не будет завершена модальная форма — а последовательность событий создания и разрушения форм будет такой:

1. Немодальная форма: событие `Destroy`.
2. Модальная форма: событие `Load`.
3. Модальная форма: событие `Init`.
4. Прочие события и ожидание закрытия.
5. Модальная форма: событие `Destroy`.
6. Модальная форма: событие `Unload`.
7. Немодальная форма: событие `Unload`.

Средства отслеживания событий

В Visual FoxPro существует возможность "перехвата" сообщений, посылаемых Windows окну любой формы приложения, при помощи функции `EventHandler`. Функция в числе параметров принимает номер сообщения (а в Windows все сообщения

имеют жестко определенные номера) и имя метода объекта (это может быть форма или любой иной объект), который должен обработать это сообщение. Другая возможность, реализуемая при помощи функции `BindEvent`, — это "перехват" вызова метода-обработчика события любого объекта и трансляция этого вызова для обработки указанному методу другого объекта. Подробнее эти возможности рассматриваются в *главе 20*.

Создание экранной формы. Конструктор форм

VFP предлагает несколько способов создания форм: программно, посредством Конструктора классов и при помощи Конструктора форм. Первые два способа подробно рассматриваются в *главе 11*, здесь мы познакомимся с назначением и возможностями **Конструктора форм**.

Настройка параметров Конструктора форм

Прежде чем мы обратим свое внимание на формы, необходимо настроить вкладку **Forms** (Формы) в меню **Tools** | **Options** (рис. 4.2).

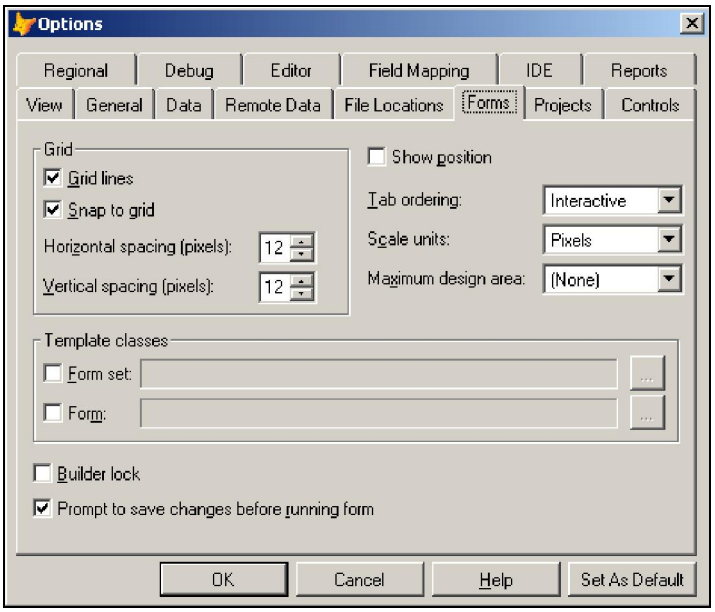

Рис. 4.2. Вкладка **Forms** меню **Options**

Вкладка предназначена для настройки работы с формами.

На форме может присутствовать координатная сетка, которая облегчает задание размеров и выравнивание объектов на форме интерактивно, с помощью мыши.

Флаг **Grid lines** определяет, будет ли координатная сетка видимой. Это позволяет точнее располагать объекты на форме.

Флаг **Snap to grid** включает выравнивание по координатной сетке объектов формы.

Если этот флаг установлен, то координаты объектов (Top, Left) при их перемещении мышью автоматически устанавливаются в узлы координатной сетки. Размеры объектов тоже автоматически "подгоняются" под ближайшую линию сетки.

Счетчик **Horizontal spacing (pixels)** определяет расстояние в пикселах между горизонтальными линиями координатной сетки, а второй счетчик **Vertical spacing (pixels)** устанавливает расстояние между вертикальными линиями сетки.

При установке флага **Show position** координаты объекта выводятся в строке состояния.

Список **Tab ordering** определяет способ задания порядка обхода объектов в форме — интерактивно или по списку. Это способ задания свойства `TabIndex` объектов одного и того же уровня вложенности.

Список **Scale units** позволяет выбрать единицу измерения координатной сетки — пикселы или фокселы. Один фоксел — это средний размер одной буквы текущего шрифта. Пиксел — наименьшая единица измерения в растровой графике или устройствах вывода изображений, чаще всего имеет форму квадрата. От количества пикселов и фокселов зависит детальность изображения.

Список **Maximum design area** разрешает выбрать максимальную область дизайна формы при проектировании. Это тот размер, за границы которого нежелательно выходить при проектировании формы.

Группа флагов **Template classes** устанавливает класс, необходимый для построения формы или набора форм. Если таковые не указаны, Visual FoxPro использует для построения базовые классы.

Флаг **Builder lock**, если он установлен, запускает автоматически построитель при добавлении объекта.

Флажок **Prompt to save changes before running form**, если он установлен, предлагает сохранить изменения перед запуском формы.

Запуск Конструктора форм

Существует несколько вариантов запуска Конструктора форм:

- из окна Менеджера проектов;
- из главного меню VFP;
- из панели инструментов VFP;
- из окна **Command**.

Чтобы запустить Конструктор форм из окна Менеджера проектов, необходимо открыть вкладку **Documents** и перейти на вложенный уровень `Forms`. После нажатия

кнопки **New** вы увидите на экране диалоговое окно, предлагающее вам произвести выбор: будете ли вы создавать форму с помощью мастера или посредством конструктора (рис. 4.3).

Рис. 4.3. Создание новой формы

Нажав на кнопку **New Form**, вы увидите на экране новую форму.

Если вы захотите воспользоваться вместо Менеджера проектов главным меню Visual FoxPro, то, последовательно выбирая из него **File | New | Form | New File**, вы получите тот же самый результат, т. е. появление Конструктора форм на экране.

Нажав на панели инструментов кнопку **New**  , мы также перейдем к Конструктору форм. Можно просто набрать в командном окне команду `CREATE FORM` и, таким образом, тоже вызвать Конструктор форм (рис. 4.4).

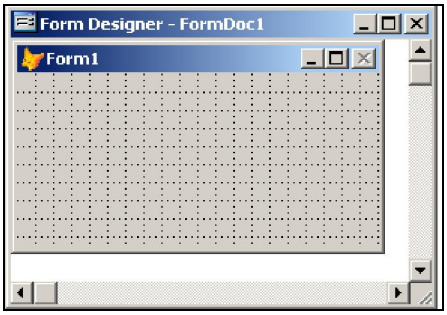

Рис. 4.4. Конструктор форм

Использование окна свойств

Все объекты Visual FoxPro, в том числе и формы, характеризуются свойствами, которые вы можете увидеть в окне свойств и настроить в соответствии со своими требованиями. Кроме свойств, для объектов существуют методы, которые выполняются при наступлении связанных с ними событий. Для просмотра свойств и методов нажмите на форме правую кнопку мыши и выберите из контекстного меню пункт **Property**. Вы увидите на экране таблицу со свойствами и методами формы (рис. 4.5).

Раскрывающийся список в верхней части окна свойств позволяет выбрать объект, для которого просматриваются или устанавливаются свойства. При выборе определенного свойства его значение появится в поле ввода, которое будет видно над списком свойств. Для корректировки свойства нужно изменить его значение в поле ввода. Откорректированные значения выводятся полужирным шрифтом.

При необходимости получить информацию о свойстве, методе или событии нужно установить курсор на свойство, метод, событие и нажать клавишу <F1>. При этом на экране появляется контекстно-зависимая справка по указанному свойству или методу.

Возможны следующие типы значений определяемого свойства (табл. 4.2).

Левее поля ввода находятся четыре кнопки, которые предназначены для выполнения следующих функций (табл. 4.3).

Построитель выражений позволяет включать в значение свойства строковые, математические и логические функции и операции с датами (рис. 4.8).

Таблица 4.2. *Типы значений определяемого свойства*

Тип свойства	**Состояние поля ввода значения свойства**
Свойство доступно только для чтения	Поле ввода свойства не активно, и вы не можете перейти в это поле для ввода значения свойства. Свойство выделяется курсивом
Свойство доступно для редактирования	Поле ввода активно, и вы можете ввести в него требуемое значение
Возможны два или более различных вариантов значений свойства	Рядом с полем коррекции свойства появляется кнопка раскрытия списка (рис. 4.6)
Возможен выбор свойств с помощью окна настройки	Рядом с полем коррекции свойства появляется кнопка открытия окна настройки (рис. 4.7)

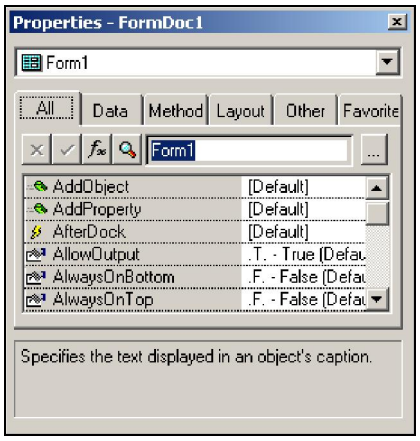

Рис. 4.5. Свойства и методы формы

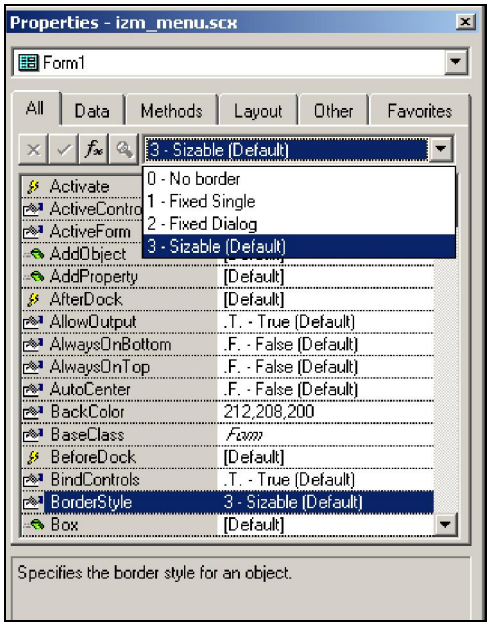

Рис. 4.6. Выбор значения свойства из списка

Таблица 4.3. *Кнопки окна* ***Properties***

Кнопка	Назначение
	Кнопка используется для отказа от введенного в поле значения
	Кнопка используется для подтверждения ввода свойства
	Вызывает построитель выражений для определения значения свойства
	Открывает диалоговое окно **Zoom** (Крупный план) для редактирования свойства

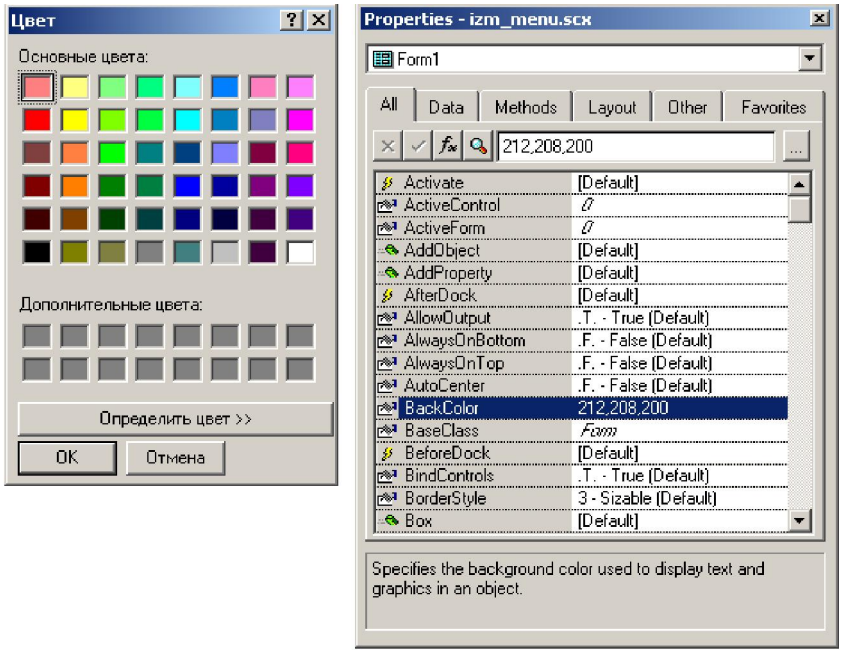

Рис. 4.7. Выбор значения свойства **BackColor** в диалоговом окне **Цвет**

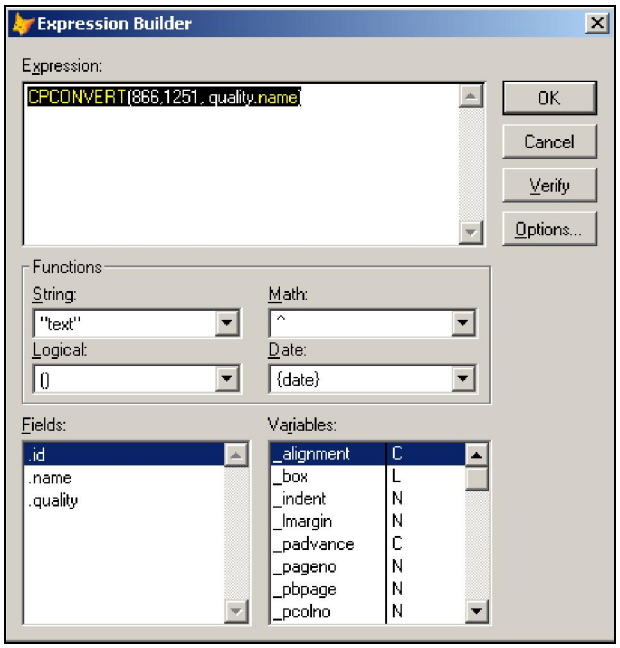

Рис. 4.8. Вызов построителя выражений

Если в окне выбора объекта нажать правую кнопку мыши, на экране появляется контекстное меню. Выбрав в этом меню пункт **Non-Default Properties Only**, мы увидим на экране только откорректированные значения свойств (рис. 4.9).

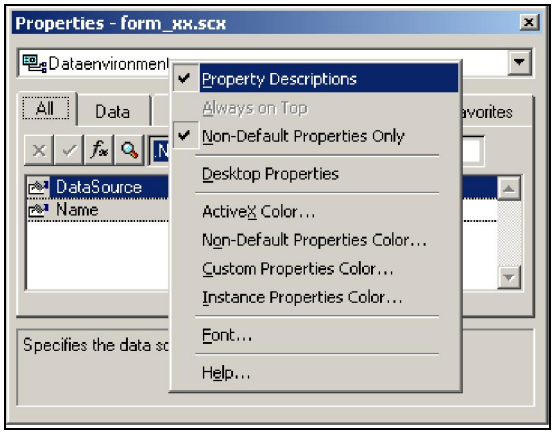

Рис. 4.9. Выбор откорректированных свойств в окне **Property**

При двойном щелчке левой кнопкой мыши на методе откроется окно редактора кода для выбранного метода (рис. 4.10).

Что означает `ThisForm` в записанном коде? Это ссылка на объект — форму.

Общие правила именования объектов приведены в табл. 4.4.

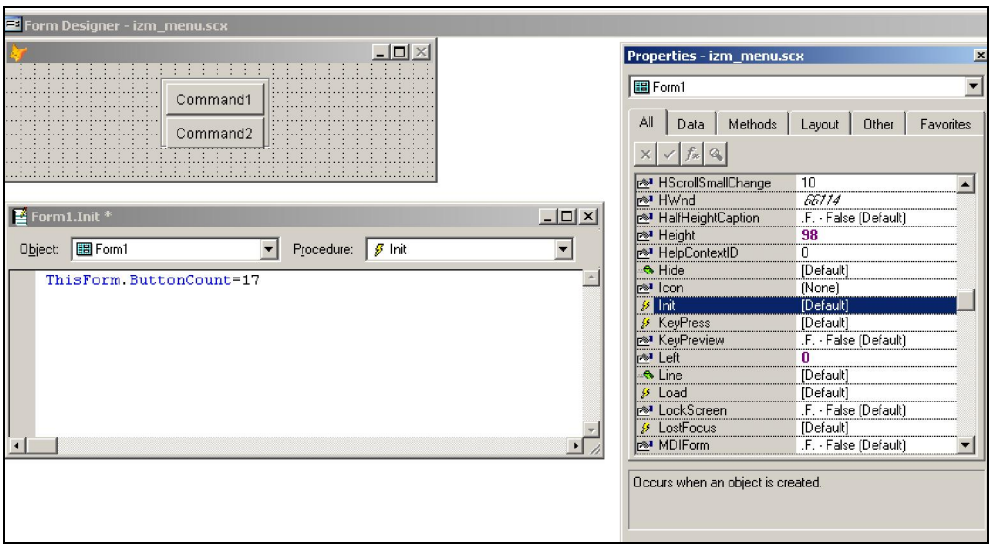

Рис. 4.10. Редактор кода методов

Таблица 4.4. *Правила именования объектов*

Ключевое слово	Ссылки
`Parent`	Определяет ссылку на объект-контейнер элемента управления
`This`	Ссылка на текущий объект для события, процедуры, свойства
`ThisForm`	Определяет ссылку на форму, которая содержит текущий объект
`ThisFormSet`	Определяет ссылку на набор форм, который содержит текущий объект

Но при обращении к объектам снаружи формы ссылки не используются. В этом случае указывают полное имя:

```
NameForm.NameObj.Caption="Выполнить"
```

Если объект является частью родительского объекта или контейнера, синтаксис используемой команды будет таким:

`<Контейнер>. <Объект>. <Свойство>` или `<Метод>`

Например:

```
Screen.FormSet1.Form1.Grid1.Column1.Header1="Номер"
```

Окно **Property** содержит 6 вкладок, краткое описание которых приведено в табл. 4.5.

Таблица 4.5. *Вкладки окна* ***Property***

Вкладка	Описание
All	Включает все свойства и методы в алфавитном порядке
Data	Содержит свойства, связанные с источником данных
Methods	Содержит список методов в алфавитном порядке
Layout	Содержит список свойств, связанных с внешним оформлением
Other	Включает все свойства, не вошедшие во вкладки **Data** и **Layout**
Favorites	Позволяет сформировать список наиболее часто используемых свойств. Чтобы создать такой список, нужно найти нужное свойство на других вкладках, нажать правую кнопку мыши, в контекстном меню выбрать **Add to Favorites**

Панели инструментов Конструктора форм

Конструктор форм содержит несколько панелей инструментов, позволяющих управлять созданием и модификацией формы.

На рис. 4.11 показано окно Конструктора форм, содержащее панели инструментов **Form Designer** (Конструктор форм), **Color Palette** (Цветовая палитра), **Layout** (Расположение объектов формы) и **Form Controls** (Объекты формы). Если эти панели отсутствуют на вашем экране, вы можете установить флажки в соответствующих оп-

циях меню **View** | **Toolbars**. Каждую из панелей можно пристыковать к панели инструментов Visual FoxPro (рис. 4.12).

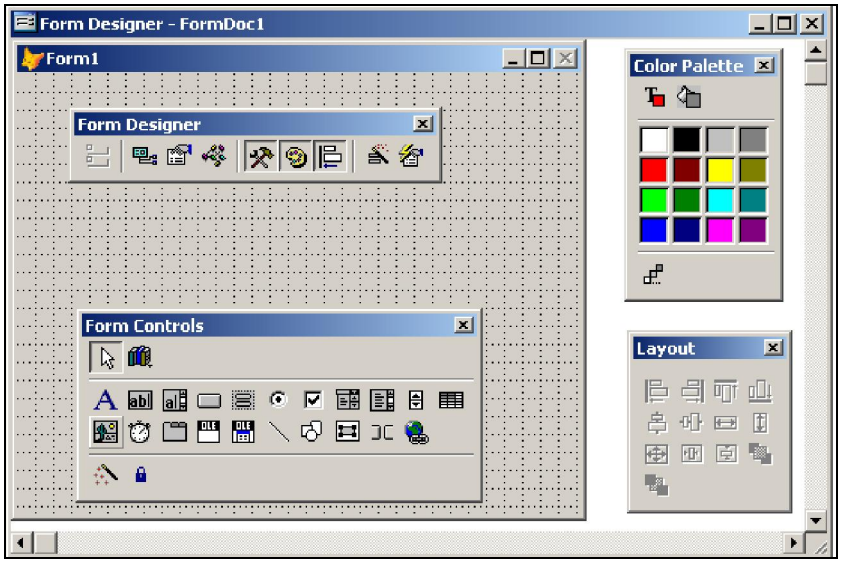

Рис. 4.11. Окно Конструктора форм с панелями инструментов

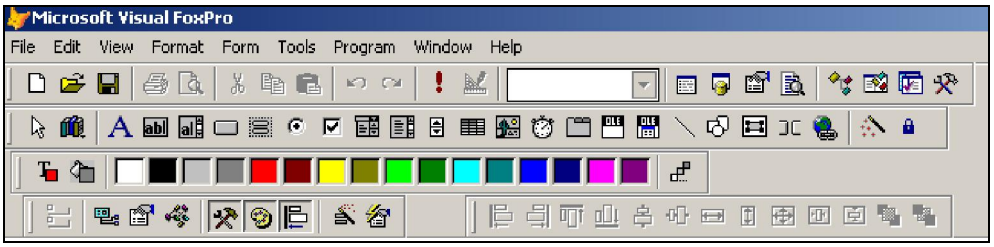

Рис. 4.12. Панель инструментов Visual FoxPro с пристыкованными к ней панелями

Практически все иконки **Toolbar** продублированы соответствующими пунктами меню. Например, панель **Layout** дублируется пунктом меню **Format** и его подпунктами.

Действие кнопки **Button Lock** можно продублировать, если дважды щелкнуть левой кнопкой мыши по иконке объекта в **Form Controls**.

Панель инструментов *Form Designer*

Панель инструментов **Form Designer** содержит кнопки вызова панелей инструментов **Form Controls**, **Color Palette**, **Layout**, а также выполняет некоторые дополнительные

функции управления формой. Краткое описание кнопок данной панели приведено в табл. 4.6.

Таблица 4.6. *Описание кнопок панели* ***Form Designer***

Кнопка	Наименование	Описание
	**Set Tab Order**	Переключает Конструктор форм в режим установки порядка обхода объектов формы
	**Data Environment**	Открывает окно определения среды окружения
	**Properties Window**	Открывает окно свойств
	**Code Window**	Открывает окно редактора кода
	**Form Controls Toolbar**	Вызывает на экран панель инструментов **Form Controls**
	**Color Palette Toolbar**	Вызывает на экран панель инструментов **Color Palette**
	**Layout Toolbar**	Вызывает на экран панель инструментов **Layout**
	**Form Builder**	Вызывает построитель экрана
	**Auto Format**	Вызывает построитель автоформата для выбранных объектов формы

Панель инструментов *Form Controls*

Панель инструментов **Form Controls** состоит из набора кнопок (табл. 4.7).

Таблица 4.7. *Типы элементов управления формы*

Кнопка	Наименование элемента	Описание
	**Select Objects** (Выбор объектов)	Указатель выделения. Кнопка имеет два состояния — **Включено** и **Выключено**. Если кнопка включена, это значит, что мышь является указателем объекта на поле формы. Если кнопка выключена, это значит, что указатель мыши имеет информацию о некоем выбранном объекте. Если вы передумали помещать выбранный элемент на поле формы, достаточно щелкнуть по кнопке **Select Objects**, чтобы отменить размещение объекта

	**View Classes** (Просмотр классов)	Позволяет выбрать класс для создаваемых в форме объектов
	**Label** (Этикетка)	Создает на экране текстовый объект

Таблица 4.7 *(продолжение)*

Кнопка	Наименование элемента	Описание
	**Text Box** (Текстовое поле)	Создает на экране поле для ввода символьной информации, которая может быть интерпретирована как число, дата или текст
	**Edit Box** (Редактируемое поле)	Создает поле, в которое можно вводить длинный текст и его редактировать
	**Command Button** (Кнопка)	Создает кнопку, на которой можно щелкнуть мышью. При этом выполняется событие `Click()`
	**Command Group** (Группа кнопок)	Размещает в форме группу связанных кнопок. Можно, например, использовать как меню
	**Option Group** (Группа переключателей)	Создает группу переключателей. Выбирается один переключатель из группы
	**Check Box** (Флажок)	Создает флажок, который имеет два состояния — включено и выключено (On/Off). Используется для логических значений
	**Combo Box** (Комбинированный список)	В обычном состоянии список показывает текущий элемент списка. При раскрытии списка можно выбрать другой элемент
	**List Box** (Список)	Создает в форме список. В списке возможен поиск
	**Spinner** (Счетчик)	Создает комбинированный элемент, представляющий собой поле для ввода числового параметра и две стрелки для изменения этого параметра
	**Grid** (Сетка)	Создает объект в виде многоколоночной таблицы
	**Image** (Графический объект)	Размещает в форме рисунок, фотографию или другой графический объект
	**Timer** (Таймер)	Создает элемент управления, который периодически запускает на выполнение команду с определенным временным интервалом
	**Page Frame** (Набор вкладок)	Создает элемент управления в виде набора вкладок. Щелкнув на вкладке мышью, можно выбрать одну из страниц. Каждая страница — это отдельная форма со своими элементами управления
	**ActiveX Control** (OLEControl) (OLE-объект)	Создает OLE-объект
	**ActiveX Bound Control** (ActiveX-объект)	Отображает содержимое OLE-объекта, хранящегося в поле типа `General`

	Line (Линия)	Создает графический элемент — линию
	Shape (Фигура)	Создает графический контур
	Container (Контейнер)	Создает в форме контейнер для размещения других объектов

Таблица 4.7 *(окончание)*

Кнопка	Наименование элемента	Описание
	Separator (Разделитель)	Размещает на панели инструментов разделитель кнопок
	HyperLink (Гиперссылка)	Создает объект, содержащий ссылку на страницу в Интернете
	Builder Lock (Закрепитель построителя)	Позволяет автоматически вызвать построитель при размещении объектов в форме
	Button Lock (Закрепитель кнопки)	Закрепляет выбранную кнопку на панели инструментов

Кнопка **View Classes** позволяет выбрать библиотеку классов элементов управления.

В текущий момент доступны классы библиотеки базовых классов Visual FoxPro, но можно переключаться на другие библиотеки. После щелчка на кнопке появляется контекстное меню, состоящее из трех пунктов (рис. 4.13).

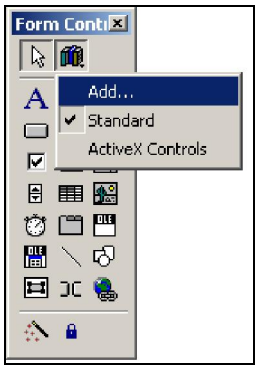

Рис. 4.13. Кнопка **View Classes**

Рис. 4.14. Добавление класса

Опция **Add** позволяет включать любую другую библиотеку (рис. 4.14). Кроме этого способа добавить новую библиотеку, существует еще один: можно добавлять библиотеки с помощью вкладки **Controls** диалогового окна **Options**. Опция **Standard** включает библиотеку базовых классов Visual FoxPro. Опция **ActiveX Controls** включает

набор классов `ActiveX Controls`, которые Visual FoxPro включает по умолчанию. Добавленные библиотеки становятся очередным пунктом меню.

Кнопки, расположенные в центральной части панели управления **From Controls**, являются доступными элементами управления, которые можно размещать в форме. О них речь пойдет ниже. А кнопки, расположенные в нижней части панели — **Builder Lock** и **Button Lock** — это служебные кнопки. Первая из них предназначена для автоматического вызова построителя элемента. Разумеется, построитель можно вызвать и вручную, но можно настроить панель управления на автоматический вызов и не тратить время на нажатие большого числа кнопок. Вторая кнопка представляет собой "автоматический повторитель", т. е. позволяет вставлять однотипные элементы в форму. Для этого кнопка должна находиться во включенном, т. е. "утопленном" состоянии.

Панель инструментов *Layout*

Панель **Layout** предназначена для выравнивания объектов формы. Краткое описание кнопок панели приведено в табл. 4.8.

Таблица 4.8. *Описание кнопок панели* ***Layout***

Кнопка	Наименование	Описание
	**Align Left Sides**	Выравнивает выбранные объекты по левому краю самого левого объекта
	**Align Top Edges**	Выравнивает выбранные объекты по верхнему краю самого верхнего объекта
	**Align Right Sides**	Выравнивает выбранные объекты по правому краю самого правого объекта
	**Align Bottom Edges**	Выравнивает выбранные объекты по нижнему краю самого нижнего объекта
	**Align Vertical Centers**	Выравнивает выбранные объекты по вертикальной оси
	**Align Horizontal Centers**	Выравнивает выбранные объекты по горизонтальной оси
	**Center Vertically**	Центрирует выбранный объект относительно вертикального центра формы
	**Center Horizontally**	Центрирует выбранный объект относительно горизонтального центра формы
	**Same Width**	Устанавливает одинаковую ширину для выбранных объектов формы
	**Same Size**	Устанавливает одинаковую ширину и высоту для выбранных объектов формы

	**Same Height**	Устанавливает одинаковую высоту для выбранных объектов формы
	**Send to Back**	Направляет выбранный объект на самый нижний слой формы
	**Bring to Front**	Направляет выбранный объект на самый верхний слой формы

Пункты меню *Form* и *Format*

После загрузки Конструктора форм в главном меню VFP появляются новые пункты — **Form** и **Format**. Обычно меню **Format** состоит из команд, управляющих характеристиками шрифта, отступами и интервалами. Однако при использовании Конструктора форм становятся доступными и некоторые другие команды. Список команд меню **Format** смотрите в табл. 4.9.

Таблица 4.9. *Команды меню* ***Format***

Команда	**Доступны после загрузки Конструктора форм**	**Описание**
Font (Шрифт)		Предназначена для выбора шрифта
Enlarge Font (Увеличить шрифт)		Увеличивает размер шрифта текущего окна
Reduce Font (Уменьшить шрифт)		Уменьшает шрифт текущего окна
Single Space (Одинарный интервал)		Использование одинарного межстрочного интервала для текста в текущем окне
1 ½ Space (Полуторный интервал)		Использование полуторного интервала для текста в текущем окне
Double Space (Двойной интервал)		Использование двойного межстрочного интервала для текста в текущем окне
Indent (Отступ)		Делает отступ текущей или выделенных строк в активном окне
Unindent (Отмена отступа)		Отменяет отступ текущей или выделенных строк в активном окне
Comment (Комментарий)		Делает выделенные строки программного кода комментариями
Uncomment (Отмена комментария)		Удаляет символы комментария из выделенных строк программы
Align (Выравнивание)	Да	Открывает окно с перечнем команд выравнивания объектов в текущем окне

Size (Размер)	Да	Привязка размера объектов
Horizontal Spacing (Горизонтальная настройка)	Да	Открывает окно с перечнем команд настройки горизонтального интервала между выделенными объектами
Vertical Spacing (Вертикальная настройка)	Да	Открывает окно с перечнем команд настройки вертикального интервала между выделенными объектами
Bring to Front (Поверх всех)	Да	Помещает выделенный объект перед всеми объектами формы

Таблица 4.9 *(окончание)*

Команда	Доступны после загрузки Конструктора форм	Описание
Send to Front (Позади всех)	Да	Помещает выделенный объект позади остальных объектов формы
Group (Сгруппировать)		Связывает выделенные в отчетах объекты в группу для совместной обработки
Ungroup (Отменить группировку)		Разделяет группу на отдельные объекты
Snap to Grid (Привязка к сетке)	Да	Совмещение левого верхнего угла перемещаемого объекта с ближайшей точкой пересечения линий сетки
Set Grid Scale (Установка размера сетки)	Да	Установка размера сетки в горизонтальном или вертикальном направлении
Text Alignment (Выравнивание текста)		Выравнивание текста внутри выделенного объекта
Fill (Заливка)		Определение орнамента заливки для фигур
Pen (Перо)		Определяет параметры пера для фигур
Mode (Режим)		Определяет прозрачность для заднего плана объекта
Make Uppercase (Отображать прописными)		Отображает текст прописными буквами
Make Lovercase (Отображать строчными)		Отображает текст строчными буквами
View White Space (Замена пробелов)		Заменяет пробелы на точки в тексте программы
Toggle Word Wrap (Флажок переключения на многострочный текст)		Разрешает размещение "длинного" текста в несколько строк

При активном Конструкторе форм в главном меню, кроме меню **Format**, появляется еще один пункт меню — **Form** (рис. 4.15).

Описание пунктов меню приведено в табл. 4.10.

Таблица 4.10. *Описание пунктов меню*

Наименование	Описание
New Property	Позволяет добавить новое свойство
New Method	Позволяет добавить новый метод
Edit Property/Method	Позволяет редактировать свойство или метод
MemberData Editor	Позволяет редактировать **MemberData**
Include File	Позволяет включать файл директив препроцессора
Create Form Set	Позволяет создать набор форм
Remove Form Set	Удаляет набор форм
Add new Form	Позволяет добавить новую форму
Remove Form	Удаляет форму
Quick Form	Вызывает построитель форм
Run Form	Запускает форму на выполнение

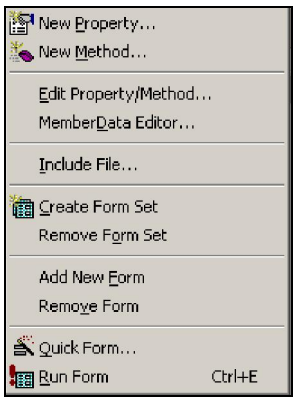

Рис. 4.15. Меню **Form** главного меню

Размещение на форме элементов управления

Основные и наиболее часто используемые элементы управления расположены на панели **Form Controls**: **Label** (этикетка), **Text Box** (текстовое поле), **Edit Box** (редактируемое поле), **Command Button** (кнопка), **Command Group** (группа кнопок), **Option Group** (группа переключателей), **Check Box** (флажок), **Combo Box** (комбинированный список), **List Box** (список), **Spinner** (счетчик), **Grid** (сетка), **Image** (графический объект), **Timer** (таймер), **PageFrame** (набор вкладок).

Для размещения объекта на форме нужно выполнить следующие действия:

1. Выбрать объект на панели **Form Controls**. Если панель отсутствует на форме и не пристыкована к панели инструментов VFP, выполните **View | Form Controls Toolbar**.
2. Установить курсор на место предполагаемого расположения элемента управления. Отрегулировать размер получившегося прямоугольника, передвигая мышь с нажатой правой кнопкой по горизонтали и вертикали. Если мышью не получается, можно воспользоваться клавиатурой. Стрелка вправо или влево изменяет местоположение элемента управления на форме, а <Shift>+<Стрелка> — размер.
3. Откройте окно **Property** и установите свойства объекта.

В процессе создания формы над ее объектами можно производить разнообразные действия: перемещать их, изменять их размеры, удалять, изменять их свойства.

Выделение объектов

Прежде всего, перед выполнением каких-либо действий объект нужно выделить. Для выделения объекта нужно установить указатель мыши на объект и нажать левую кнопку мыши. Для выделения группы объектов нужно нажать клавишу <Shift> и, удерживая ее в нажатом состоянии, выделить мышью нужную группу объектов. Второй способ выделения состоит в том, что вы нажимаете кнопку **Select Objects** на панели **Form Controls**, а затем мышью выделяете группу объектов, после этого можно управлять группой объектов как единым объектом.

Если необходимо получить справочную информацию о размещаемом элементе управления, нужно выделить его и нажать клавишу <F1>.

Отмена выделения

Для отмены выделения нужно установить указатель мыши на любое свободное от элементов место формы и нажать левую кнопку мыши. В этом случае происходит отмена выделения как одного объекта, так и отмеченной группы.

Изменение размеров объекта

Выделенный объект маркируется маленькими черными квадратиками по углам и сторонам. Для изменения размеров объекта необходимо "зацепить" мышью один из квадратиков и тянуть в нужную сторону. При этом объект изменит свои размеры. Если вы

еще не приобрели нужные навыки, можно воспользоваться окном свойств **Property**, найти свойства `Height` и `Width` и установить нужную высоту и ширину объекта. Еще один возможный вариант — выделить объект и, нажав клавишу <Shift>, нажимать стрелки влево-вправо и вверх-вниз. При этом также изменяются размеры объекта.

Перемещение объекта

Для перемещения объекта нужно выделить его, а затем перетащить мышью в нужное место. Можно воспользоваться также стрелками. При пользовании стрелками объект перемещается на один пиксел при каждом нажатии клавиши, иногда это критично для достижения необходимой точности расположения объекта.

Удаление объекта

Удаляемый объект необходимо выделить. Удалить его можно клавишами <Backspace> или <Delete>, а также через меню **Edit | Cut**.

Выравнивание и изменение размеров объектов

Для выравнивания объектов относительно сетки формы и для установки одинаковых размеров для однотипных объектов можно использовать меню **Format** окна Конструктора форм и панель инструментов **Layout**. На панели расположены кнопки, выполняющие наиболее распространенные действия с объектами. Меню содержит полный набор действий над объектами (рис. 4.16).

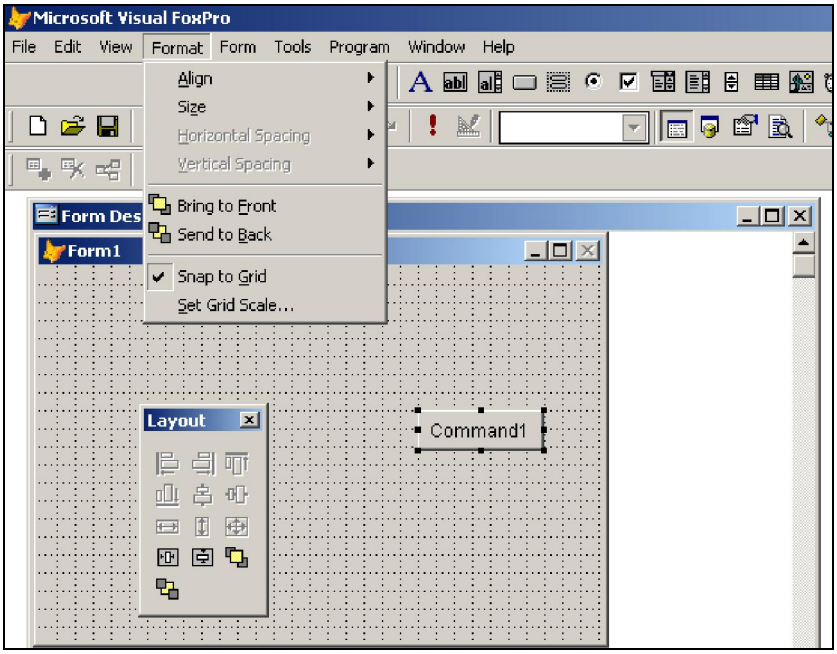

Рис. 4.16. Инструменты для выравнивания объектов формы

Использование окна свойств для работы с элементами управления

При выделении на форме элемента управления в окне свойств высвечиваются его свойства, методы и события (PEM — сокращение от `Properties`, `Events`, `Methods`).

Добавление в форму пользовательских свойств и методов

Диалоговое окно **Property** состоит из двух частей: в левой части отображается список свойств, в правой — их значения. Изменить значение свойства достаточно просто — нужно найти свойство и ввести в текстовое поле под списком нужное значение. Существует ряд свойств, которые имеют только определенный диапазон или тип данных значений. Например, свойство `Visible` может принимать значение только `.T.` или `.F.` Значения таких свойств выбираются из комбинированного списка, который появляется в таких случаях вместо текстового поля.

В проектируемой форме можно не только менять значения свойств, но и добавлять новые. Это можно делать в меню **Form | New Property**, в этом случае появится окно,

в котором можно ввести название свойства и его описание, а затем щелкнуть по кнопке **Add** (рис. 4.17).

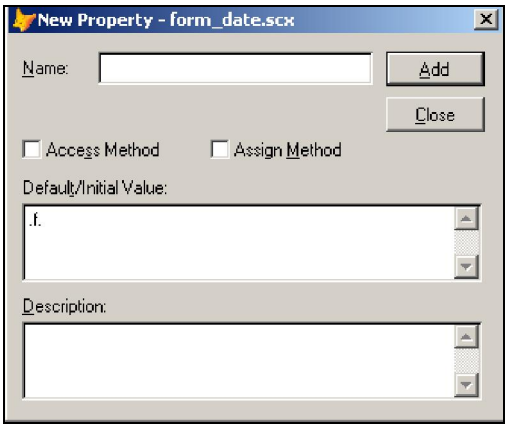

Рис. 4.17. Окно ввода нового свойства

По умолчанию новое свойство получает значение `.F.`.

Для добавления нового свойства (рис. 4.18) можно использовать альтернативный метод: правой кнопкой мыши щелкнуть по списку свойств в окне **Property**, затем в контекстном меню выбрать пункт **Add to Favorites**, в этом случае в конце списка свойств добавится строка с названием свойства по умолчанию `_memberdata`, которую потом можно отредактировать по вашему желанию, выбрав в контекстном меню пункт **Edit Property/Method**.

Довольно часто в виде свойства формы используется массив (рис. 4.19). Он, в частности, применяется для использования `ComboBox` в `Grid` как источник данных для комбинированного списка. Чтобы создать массив, нужно указать его имя и размерность. Если размер неизвестен, можно указать минимальный, в процессе работы с ним он будет расширен. Заполнение массива значениями происходит обычно в методе `Init` экранной формы. Например, так:

```
SELECT naim, code FROM us1 INTO ARRAY ThisForm.aCombo
SELECT naim, code FROM us2 INTO ARRAY ThisForm.aCombo1
SELECT naim, code FROM us3 INTO ARRAY ThisForm.aCombo2
```

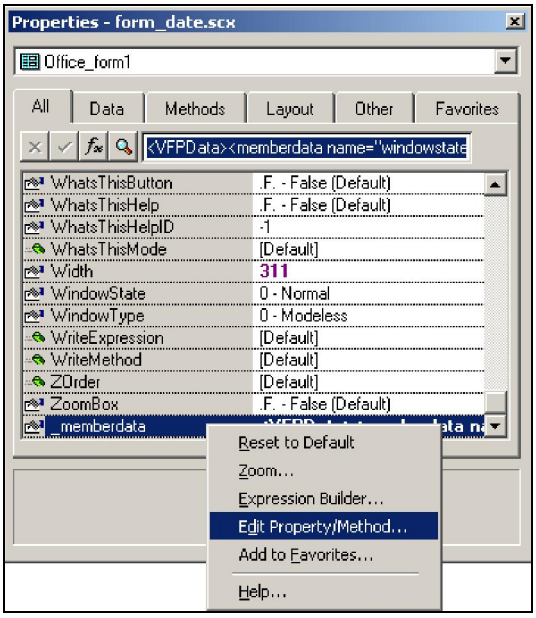

Рис. 4.18. Добавление нового свойства

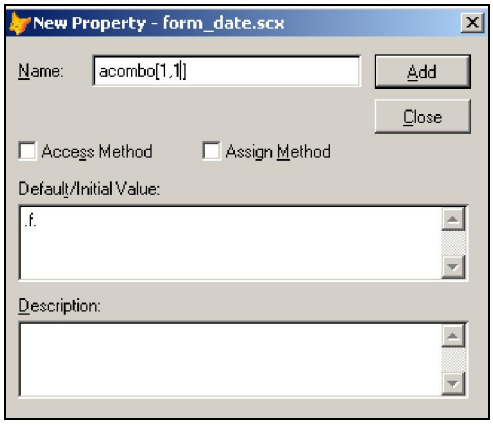

Рис. 4.19. Ввод массива в качестве нового свойства

Добавление и редактирование методов

Методы можно добавлять в форму точно так же, как свойства. Для этого нужно выбрать в главном меню **Form | New Method**. Для нового метода по умолчанию устанавливается пустая процедура, которая состоит из операторов `PROCEDURE` и `ENDPROC` и не содержит между ними никакого программного кода.

Текст кода должен написать программист.

Сохранение формы

Для сохранения формы можно использовать несколько способов:

- в главном меню выбрать пункт **File** | **Save (Save As)**;
- на панели инструментов нажать кнопку **Save**;
- нажать комбинацию клавиш <Ctrl>+<W> на клавиатуре;
- нажать на "крестик" в правом верхнем углу VFP.

В *главе 2* уже говорилось о необходимости хранить все файлы проекта в специально выделенных для этого каталогах. Важно сохранить форму в строго определенном месте. Этим местом, как правило, является подкаталог `FORMS` вашего проекта.

Управление формами

Форму или набор форм можно запустить на выполнение командой

```
DO FORM <имя формы> | ? [NAME <имя переменной> [LINKED]]
[WITH  <список параметров>] [TO переменная] [NOSHOW]
```

Рассмотрим далее опции и параметры.

`NAME <имя переменной> [LINKED]` — задает имя переменной или элемента существующего массива, используя которое можно ссылаться на форму или на набор форм. Тип переменной — объект. Если опция `NAME` опущена, то VFP создает переменную типа `Object` с тем же именем, что и имя файла формы. Если включена опция `LINKED`, то форма присоединяется к переменной, ассоциированной с формой. Тогда если ассоциированная переменная оказывается вне области видимости или освобождается, то освобождается и форма. Например, задайте следующие команды в командном окне:

```
SET DEFAULT TO D:\CHAR4\KNOPKA
DO FORM Form1 NAME ln LINKED
```

На экране появится форма с кнопкой **Выход**.

А теперь дайте команду

```
RELEASE ln
```

Форма закроется

`WITH <список параметров>` — список передаваемых форме параметров.

Параметры передаются обработчику события `Init`, если свойство формы `WindowType=0 (Modeless)` или `1 (Modal)`. Если же свойство `WindowType=2 (Read)` или `3 (ReadModal)`, то параметры передаются обработчику события `Load`. При этом значения `2 (Read)` и `3 (Read Modal)` включены для совместимости с предыдущими версиями и доступны только для форм, полученных в результате конвертации из предыдущих версий.

`TO <переменная>` — имя переменной, содержащей результат, возвращаемый формой или набором форм. При этом команда `RETURN <значение>` задается в обработчике со-

бытия UNLOAD формы. Если такая команда отсутствует, то переменная получает значение .T. Опция To используется для модальных форм (свойство WindowType=1).

Если обработчик события Load вернет .F., то форма не будет открыта, а обработчик ее события Unload не вернет никакого значения переменной. А если команду Return .F. задать в событии Init формы, то такая форма даже не будет загружена. Это даже более важно, чем Return в Load, поскольку решение о загрузке или незагрузке формы принимается на основании переданных форме параметров, а параметры передаются именно в Init формы.

NOSHOW — метод Show не будет вызван при запуске формы (т. е. форма будет не видна). Для отображения формы необходимо установить свойство формы

Visible = .T. и выполнить метод Show. Если NOSHOW опущена, то DO FORM выполнит метод Show формы.

Запущенная форма по умолчанию принимает имя файла SCX, в котором она хранится, а внутри приложения к ней можно обратиться как к переменной _SCREEN. Но форму можно запускать и с другими именами, например, используя следующую команду:

```
DO FORM Form1 NAME Form2.
```

А если использовать команду еще раз, но уже с другим именем, например

```
DO FORM Form1 NAME Form3,
```

мы получим еще один экземпляр запущенной формы.

Если мы используем в команде ключевое слово LINKED, например,

```
DO FORM Form1 NAME Form2 LINKED,
```

то мы получим возможность обращаться к методам и свойствам формы Form1 как к Form2:

```
Form2.AutoCenter=.T.
```

Если форма требует передачи параметра, то он может быть передан в форму таким образом (например, передадим в форму пункт меню, из которого она вызвана):

```
DO FORM Form1 with BAR()
```

В этом случае в методе Init() формы мы должны будем принять параметр по команде LPARAMETERS:

```
*Метод Init() формы
LPARAMETERS nbar
If ! Empty(nbar)
     ThisForm.Caption="Участок № "+STR(nbar,2)
endif
```

Иногда требуется запустить форму, но до ее появления на экране выполнить некоторые действия, например, изменить значения определенных свойств. Тогда до момента окончания действий форма должна быть скрыта. Это достигается запуском с опцией NOSHOW:

```
DO FORM Form1 NOSHOW
```

Часто программистам приходится бороться с повторным запуском одной и той же формы. Для исключения повторного запуска рекомендуют в меню, из которого запускается форма, включить такой код:

```
SKIP FOR WEXIST("Form1").
```

В этом случае пункт меню блокируется (становится недоступным) до тех пор, пока работающая форма не будет закрыта.

Если вы не желаете, чтобы форма была загружена (например, вы проверили права пользователя, и эта форма не должна быть ему доступна), то, поместив в событии `Load()` формы команду `RETURN .F.`, вы добьетесь того, что форма для этого пользователя на экране так и не появится.