ГЛАВА 12

Создание меню

Законченное приложение, имеющее ряд компонентов, для вызова этих компонентов должно иметь собственное меню. К меню предъявляется ряд требований: оно должно быть удобным для использования, интуитивно понятным пользователю любой квалификации, иметь привлекательный дизайн. Такие меню вполне реально создать с помощью Visual FoxPro, поскольку язык содержит большое количество команд и функций создания и управления меню. Кроме того, в Visual FoxPro имеется Конструктор меню, который позволяет создавать не только обычные горизонтальные меню, но и контекстные меню, вызываемые на экран щелчком правой кнопки мыши. Но, поскольку при конструировании меню применяются некоторые специальные термины, сначала стоит разобраться с терминологией.

Меню состоит из главной линейки меню (**Bar** — меню) и набора выпадающих (раскрывающихся) меню (**Popup** — меню).

Линейка меню или **строка меню** (Menu bar) — меню верхнего уровня, которое присутствует на экране во все время выполнения приложения (рис. 12.1).

Рис. 12.1. Строка меню

Элемент главной линейки меню называется **Pad**. Выбор элемента меню либо вызовет выпадающее меню следующего уровня, либо выполнит некоторые действия.

Выпадающее меню (Menu popup) — меню следующего уровня иерархии. Представляет из себя список, появляющийся при наведении мыши на пункт строки меню (рис. 12.2). Щелчок кнопки мыши на пункте выпадающего меню, как правило, запускает какую-либо операцию.

Элемент выпадающего меню называется **Bar**.

Раскрывающееся меню или **контекстное меню** (Shortcut menu) — вертикальный список пунктов меню. Это то же самое выпадающее меню (**Popup** — меню), но при-

вязанное не к пункту меню, а к контексту: к какому-либо объекту или положению указателя мыши (рис. 12.3).

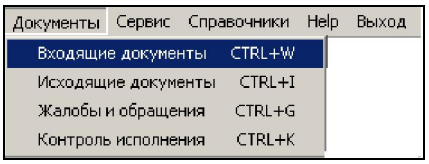

Рис. 12.2. Выпадающее меню

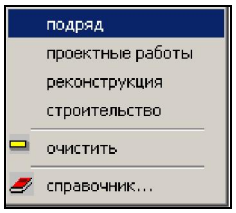

Рис. 12.3. Раскрывающееся меню

Пункт меню (Menu prompt) — это элемент списка меню. При щелчке на пункте меню может быть запущена какая-либо операция, либо открыто подменю (Submenu).

Чтобы понять идеологию работы с меню в Visual FoxPro, достаточно внимательно присмотреться к основной линейке меню самой среды VFP. Первое, что бросается в глаза, — это то, что линейка меню есть только и исключительно у основного окна FoxPro. Ни одна подчиненная форма, открывающаяся внутри этого окна, не имеет своего собственного меню, расположенного на этой подчиненной форме.

Что бы вы ни открыли: какой-либо дизайнер (формы, класса, таблицы и т. п.), утилиты (**Class Browser**, **Object Browser** и т. п.), ни одно из этих окон не имеет собственного меню.

Меню может появиться у формы, только если эта форма открывается как независимое окно. Вне основного окна FoxPro. Это видно на примере окна **Debugger**, если вы установили режим его открытия в настройках как **Debug Frame** (т. е. вне основного окна FoxPro). Если вы переключите настройку в режим **FoxPro Frame**, то меню пропадет. Точнее, в этом случае меняется интерфейс работы с отладчиком.

Как же осуществляется управление подчиненными формами? Есть несколько приемов.

Меню в FoxPro контекстно-зависимое. Это значит, что хотя пункты меню остаются одни и те же, но выполняемые ими действия зависят от того, какое именно окно активно в данный момент. Например, если вы находитесь в режиме редактирования программного кода файла PRG, то пункт меню **Save As** предложит вам сохранить именно файл PRG, а если редактируете форму, то — как форму (файл SCX).

Кроме того, если для данного окна какой-либо пункт меню не имеет смысла, то этот пункт меню становится недоступным. Например, если вы находитесь в окне проекта (файл PJX), то будет недоступно вообще все меню пункта **Edit**. Действительно, что там можно скопировать/вставить?

Далее, как правило, в момент открытия (активизации) какой-либо формы в основной линейке меню появляется дополнительный пункт, предназначенный для обслуживания именно этой формы. Например, когда вы находитесь в окне проекта (файл PJX),

то появляется пункт меню **Project** (перед пунктом меню **Windows**). Стоит вам переключиться в другое окно, например, в окно **Command**, как этот пункт меню пропадает, но появляется другой пункт меню, для обслуживания этого окна. В случае с окном **Command** — это пункт меню **Format**, появляющийся перед пунктом меню **Tools**.

В некоторых случаях могут появляться не пункты меню в основной линейке меню, а подпункты в выпадающих меню. Например, при открытии нового окна в списке пунктов меню отображающихся в пункте **Windows** появляется новый пункт с именем этого открытого окна. Аналогичная ситуация будет и при работе с Конструктором меню и пунктом меню **View**.

Однако иногда все-таки необходимы элементы управления непосредственно на форме. В этом случае на форме создается **ToolBar** или просто дополнительный набор кнопок. Но дополнительного меню на самой форме не создается! Другими словами, "узел управления" всегда находится в одном месте. Это линейка меню основного окна проекта. Пользователю не надо судорожно искать, где же у него кнопка. "Кнопка" всегда находится в одном месте. Собственно, это не столько стандарт именно FoxPro, сколько стандарт Microsoft. Более того, есть стандарт расположения некоторых пунктов меню.

- Первый пункт меню всегда **File**. Последним элементом этого меню является пункт **Выход**.
- Второй пункт меню всегда **Edit**.
- Предпоследний пункт меню всегда **Windows**, содержащий список открытых окон.
- Последний пункт меню всегда **Help**. Последним элементом этого меню является пункт **О программе**.

Если при написании вашей программы вы будете придерживаться этих правил, то любой пользователь достаточно быстро освоится с управлением программой и не будет утверждать, что ваша программа неудобна.

В FoxPro по умолчанию приложение создается именно в основном окне FoxPro. При этом системное меню заменяется вашим собственным меню, а системный **ToolBar** вашим собственным **ToolBar**.

Создание меню в Конструкторе меню

Для создания различных видов меню в Visual FoxPro имеется специальный инструмент — Конструктор меню (Menu Designer). Для запуска **Конструктора меню** существует несколько способов:

- из главного меню Visual FoxPro (**File | New | Menu**);
- из командного окна (**CREATE MENU**);
- из окна Менеджера проектов (**Project Manager | Other | Menu | New**).

Еще до появления Конструктора меню на экране отображается окно с запросом, уточняющим, что же вы хотите создать — обычное меню-линейку или контекстное (рис. 12.4).

При нажатии кнопки **Menu** создается выпадающее меню, а при выборе **Shortcut** — контекстное. И только после этого вы увидите диалоговое окно Конструктора меню (рис. 12.5).

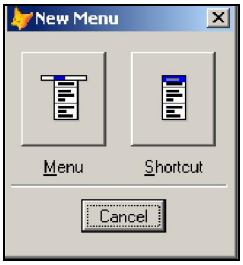

Рис. 12.4. Выбор вида меню

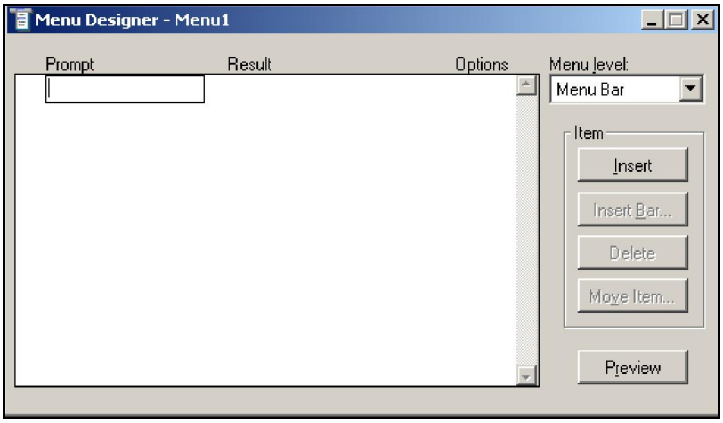

Рис. 12.5. Конструктор меню

С помощью Конструктора можно создать:

- меню, заменяющее главное меню Visual FoxPro;
- меню формы верхнего уровня;
- выпадающее меню.

Меню, заменяющее главное окно Visual FoxPro

При активизации обычные меню располагаются на месте главного меню Visual FoxPro: либо взамен его, либо вместе с ним. Главное, что необходимо предусмотреть, — это правильное распределение пунктов меню по уровням иерархии. Поэтому не поленитесь нарисовать эскиз меню и согласуйте его с заказчиком, и только потом приступайте к реализации задуманного с помощью Конструктора меню.

Комбинированный список вверху справа отображает текущий уровень меню. Конструктор имеет три колонки: **Prompt**, **Result** и **Options**. Колонка **Prompt** служит для ввода текста нового выпадающего меню, который будет отображаться как пункт меню. Если вы хотите назначить пункту меню "горячую" клавишу, то это можно сделать именно здесь, достаточно перед текстом пункта ввести символы "\<".

В этом столбце допустимо вводить специальные символы, которые сами не отображаются, но приводят к некоторым спецэффектам.

- "\<" — назначение "горячей" клавиши, как сочетание клавиши <Alt> и буквы, сразу следующей за этими спецсимволами. Такая буква будет подчеркнута при отображении.
- "\-" — вместо пункта выпадающего меню будет отображена горизонтальная линия.
- "\|" — пункт выпадающего меню будет расположен не под предыдущим пунктом, а справа, предваряемый вертикальной линией. Правда, такое расположение пунктов выпадающего меню используется достаточно редко.

Комбинированный список **Result** содержит 4 пункта:

- `Command` (Команда);
- `Pad Name` (Имя пункта);
- `Submenu` (Подменю);
- `Procedure` (Процедура);

Если выбран вариант `Command`, то в появившемся окне нужно ввести текст команды, вызывающей форму или некую программу.

Пункт `Pad Name` или `#Bar` (соответственно для меню верхнего уровня или более низкого уровня) позволяет присоединить к создаваемому пункту меню встроенных комплексных операций VFP, например, `Cut` и `Paste` (рис. 12.6).

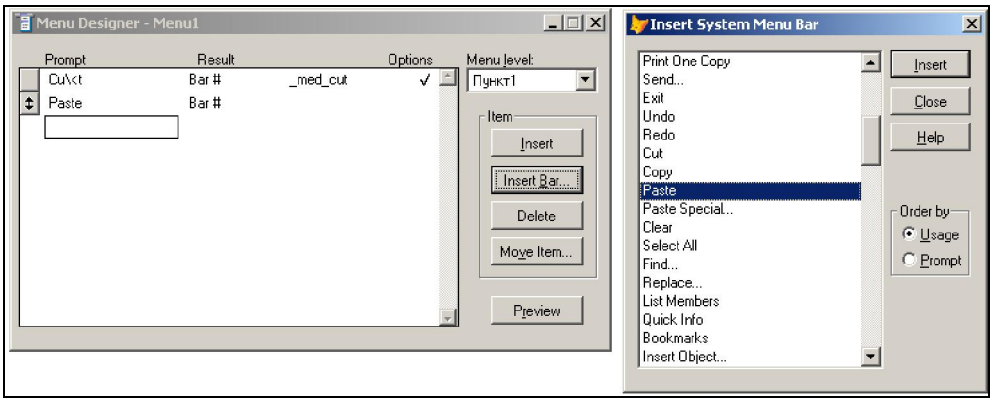

Рис. 12.6. Добавление в меню стандартных функций VFP

Для того чтобы выбрать стандартную функцию, используйте кнопку **Insert Bar** в правой части Конструктора меню.

Третий вариант ввода — это `Submenu`. При выборе из списка **Submenu** будет создано отдельное меню, для этого рядом с названием пункта появится кнопка **Create** (или **Edit**, если подменю было создано ранее).

В случае выбора из списка пункта `Procedure` будет открыто окно для ввода процедуры. Казалось бы, это удобно — ввести текст процедуры, и в ответ на нажатие кнопки пункта меню будут выполнены определенные действия. Однако не стоит забывать о том, что меню нужно обязательно генерировать, поэтому каждое изменение процедуры приведет к новой генерации меню. Скорее всего, именно поэтому этот вариант используется редко.

При нажатии на кнопку **Options** вы можете сразу же назначить пункту меню комбинацию клавиш для быстрого вызова пункта с клавиатуры (рис. 12.7).

Одним из свойств меню является возможность помещать рядом с наименованием пункта меню графическое изображение. Для размещения графического изображения нужно воспользоваться кнопкой **Options**. В группе **Picture** выберите нужный файл. Для выбора файла можно использовать опцию **File**, чтобы выбрать из списка имеющихся у вас графических изображений подходящее, а можно использовать опцию **Resource**. В этом случае вам будет предложен список имеющихся системных графических изображений (например, Print или Send).

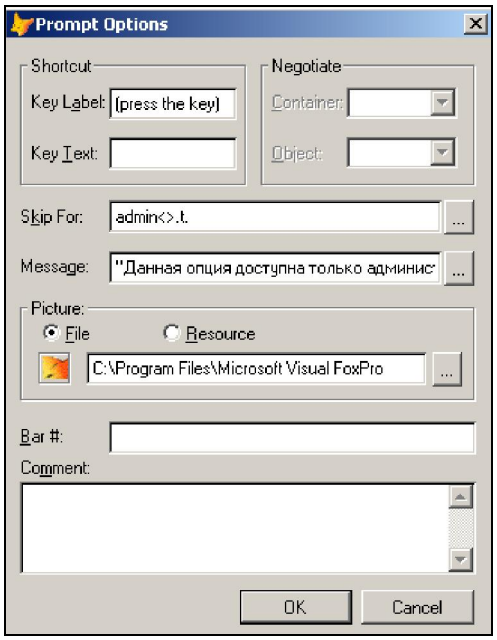

Рис. 12.7. Определение свойств меню

Размещение графического изображения возможно только для подпунктов меню.

Область **Negotiate** (Соглашение) имеет два раскрывающихся списка:

- **Container** — определяет расположение меню при редактировании по месту OLE-объектов;

- **Object** — задает расположение меню при выполнении приложения типа Active Document в Web-браузере.

Поле **Skip For** позволяет заблокировать пункт меню. Меню можно блокировать для пунктов, которые еще не готовы, чтобы не вызывать недовольства пользователей, а можно определить некоторые условия блокировки. Условия записываются в поле **Skip For**. Как только значение логического выражения, записанного вами в **Skip For**, становится истинным, пункт меню становится недоступен. В поле **Message** можно ввести сообщение, которое будет отображено в строке состояния, если курсор будет установлен на этот пункт меню.

Текстовое поле **Pad Menu** задает имя пункта меню, а **Comment** — комментарий к пункту меню. По умолчанию при генерации меню пунктам меню присваиваются уникальные имена. Они не очень читабельны (выглядят, например, так: `_0k50sa3ds`), поэтому для улучшения читабельности можно присваивать эти имена явным образом, используя для этого поле ввода **Pad Name**.

Созданное меню необходимо сгенерировать, для этого нужно выбрать пункт главного меню **Visual FoxPro Menu | Generate** (рис. 12.8).

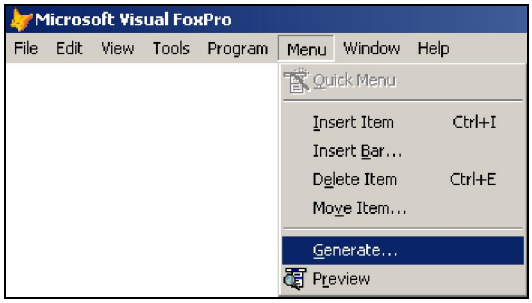

Рис. 12.8. Генерация созданного меню

Меню в FoxPro — это обычный файл PRG, но с измененным расширением. Это расширение mpr. Откомпилированное меню имеет расширение mpx.

Сгенерированное меню представляет собой 4 одноименных файла на диске:

- Mymenu.MNT — файл описания меню (memo-файл);
- Mymenu.MNX — файл описания меню;
- Mymenu.MPR — сгенерированная программа меню;
- Mymenu.MPX — скомпилированная программа меню.

Еще один путь для генерации — включение меню в состав проекта. При нажатии кнопки **Build** и установленном флажке **Rebuild All** происходит генерация всех меню, включенных в проект.

Это было создание основной части меню. Теперь надо сделать это меню контекстно-зависимым. То есть если активна какая-то конкретная форма, то либо пункт меню не активен, либо его реакция определяется собственно этой формой.

Поскольку реакция на выбор пункта меню определяется той формой, которая открыта в данный момент, то логично и сам код обработки написать собственно в форме. Тем более что, как правило, есть еще дублирующая система управления в виде ToolBar.

Делается это путем создания в форме соответствующих методов с заранее фиксированными именами. Например, если надо сохранить документ, то создаем метод с именем SaveDocument во всех формах, где такая операция в принципе имеет смысл. Теперь в самом пункте меню необходимо сделать две вещи. Проверить, что у активной в данный момент формы существует метод с таким именем. И запустить его на выполнение, если он существует.

В FoxPro у основного окна **_SCREEN** существует свойство ActiveForm, которое возвращает ссылку на активную в данный момент форму. Поэтому собственно вызов метода из пункта меню будет выглядеть так:

```
_SCREEN.ActiveForm.SaveDocument()
```

Эту команду надо просто записать в столбце без имени в Конструкторе меню, в столбце **Result**, предварительно выбрав пункт **Command** Однако если никакой формы не открыто или форма не имеет такого метода, то выбор этого пункта приведет к ошибке. Чтобы этого не произошло, необходимо сделать доступным или не доступным данный пункт меню. Делается это в опции SKIP FOR, где пишется такой код

```
NOT (TYPE("_VFP.ActiveForm")="O" AND
PemStatus(_VFP.ActiveForm,"SaveDocument",5)=.T.)
```

Здесь использована системная переменная _VFP, а не _SCREEN. Но в отношении свойства ActiveForm обе они дают один и тот же результат.

Использовано отрицание, поскольку опция называется SKIP FOR, в переводе "пропустить, если". Но нам-то надо наоборот. "Разрешить, если". Поэтому и ставится общее отрицание.

В результате пункт меню **Сохранить** будет доступен только тогда, когда открыта форма, у которой есть метод с именем SaveDocument, и при выборе этого пункта меню будет выполнен именно этот метод.

Теперь как создать дополнительный пункт меню, обслуживающий данную форму. То есть пункт меню, появляющийся в основной линейке меню при активизации формы и пропадающий при переходе на другую форму. Для этого надо создать новое меню. Но это новое меню будет состоять только из одного пункта в главной линейке меню (того, который и будет вставляться). Назовем это меню, например, OneFormMenu.

Чтобы это новое меню при запуске не замещало существующее, а дополняло его, необходимо в Конструкторе меню выбрать пункт View, подпункт **General Options** и переключатель **Location** в положение **After** и в появившемся списке пунктов меню выбрать, например, пункт **Edit**.

Здесь проблема в том, что список Pad-пунктов основной линейки меню после, или перед которыми можно осуществлять вставку, ограничен только списком системных пунктов меню. Но это не представляет большой проблемы, поскольку в любой систе-

ме обязательно будут существовать четыре системных пункта: **File**, **Edit**, **Windows** и **Help**.

Причем здесь речь идет не о том тексте, который видит пользователь, а именно о внутренних именах. Надо только не забыть при конструировании основного меню присвоить соответствующим Pad-пунктам их системные имена. Это делается в окне **Prompt Options** соответствующего Pad-пункта меню в разделе **Pad Name**.

```
File            = _MSM_FILE
Edit            = _MSM_EDIT
Windows         = _MSM_WINDO
Help            = _MSM_SYSTM
```

Кроме того, чтобы из текста программы можно было обращаться к этому дополнительному элементу меню, ему надо дать имя. Таким же способом. В разделе **Pad Name**. Чтобы не путаться в именах, дайте ему то же самое имя, что и файлу меню `OneFormMenu`.

Теперь в событии формы `Activate` формируем дополнительный пункт меню командой

```
DO OneFormMenu.mpr
```

А в событии `Deactivate` уничтожаем его

```
RELEASE PAD OneFormMenu OF _MSYSMENU
```

Поскольку при подаче команды `RELEASE` или при вызове метода формы `RELEASE` событие `Deactivate` не наступает, то его надо также вызвать принудительно при уничтожении формы. Можно это выполнить в событии `UNLOAD` формы

```
ThisForm.Deactivate()
```

Кроме собственно Pad-пункта меню, надо удалить и `Popup` — меню, вызывающееся при выборе этого пункта. В этом случае необходимо дать нормальное имя этому Popup-меню в Конструкторе меню через пункт **View**, подпункт **Menu Options**. Например, если вы дадите ему то же самое имя `OneFormMenu`, то в событии `Deactivate` допишите еще такую команду:

```
RELEASE POPUPS OneFormMenu EXTENDED
```

Обычное меню, заменяющее главное меню VFP, выполняется во время работы вашего приложения. После окончания работы необходимо вернуть на место главное меню Visual FoxPro.

> ***ЗАМЕЧАНИЕ***
>
> Кстати говоря, при использовании русских названий в пункте меню и назначении им русских "горячих" клавиш появляется сообщение о синтаксической ошибке (рис. 12.9).

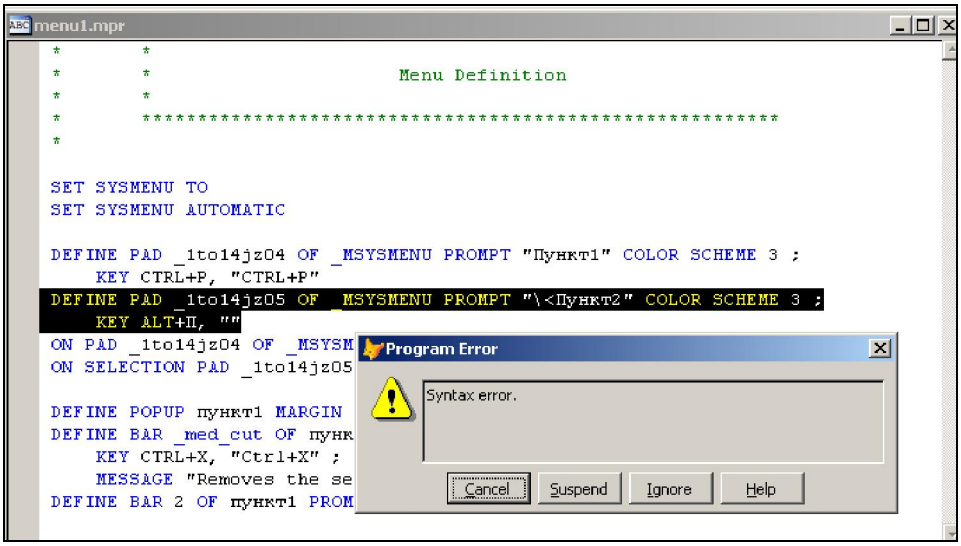

Рис. 12.9. Синтаксическая ошибка при назначении "горячей" клавиши

Чтобы избежать подобной ошибки, можно:

- начинать пункт меню с цифры;
- заменять первую букву меню соответствующей по написанию латинской;
- комбинацию клавиш быстрого вызова назначать вручную.

Меню формы верхнего уровня

Строку меню можно вставить только в два типа форм: `As-Top-Level` (форма верхнего уровня) и `Desktop`.

В первом случае можно использовать Конструктор меню, во втором придется писать меню самостоятельно в методе `Init` формы.

При создании меню в `As-Top-Level Form`:

1. Вызовите Конструктор меню.
2. В окне **New Menu** выберите **Menu**.
3. Откройте в главном меню пункт **View | General Option**; в появившемся окне установите флажок **Top-Level Form** (рис. 12.10).
4. Сформируйте в Конструкторе ваше меню и сохраните его. В `As-Top-Level Form` в методе `Init` введите код:

   ```
   DO MenuName.MPR WITH oForm, lAutoRename,
   ```

   где `oForm` — объектная ссылка на форму, в которой размещается меню;

`lAutoRename` — логическая переменная, определяет, необходима ли генерация нового уникального имени для меню. Если возможны многочисленные запуски формы, необходимо указать `.T.` в этом параметре.

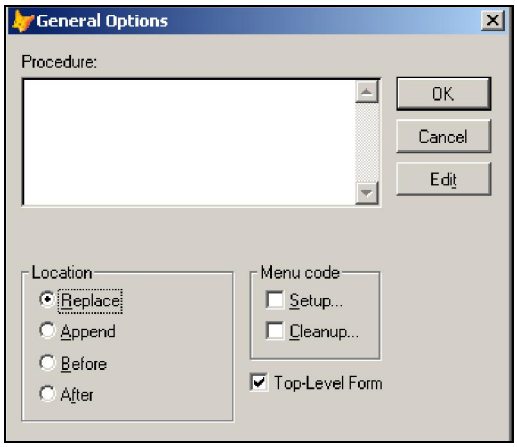

Рис. 12.10. Диалоговое окно **General Options**

В методе `Destroy` этой формы введите код:

```
RELEASE MENUS (THIS.NAME) EXTENDED
```

Пример вызова меню в `As-Top-Level`-форме:

```
DO MyMenu.mpr WITH this,.T.
```

Если вы все сделали правильно, то в форме появится меню (рис. 12.11).

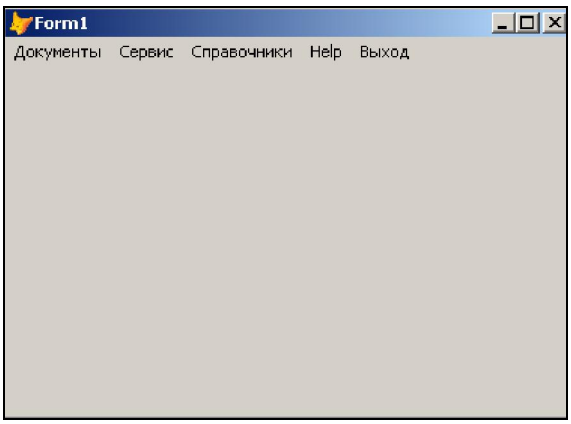

Рис. 12.11. Меню в форме верхнего уровня

Раскрывающееся (контекстное) меню

Контекстное меню обычно располагается в точке расположения указателя мыши.

При запросе вида создаваемого меню для контекстного меню необходимо указать `Shortcut` (короткое меню) и ввести в Конструкторе меню его пункты. Контекстные меню можно встраивать в элементы управления формы. Для того чтобы выполнить такое "внедрение", нужно произвести следующие действия:

1. Выбрать элемент управления, в который будет встроено меню.
2. В событии `RightClick` написать следующий код:

```
DO MyMenu.MPR
```

Меню будет вызвано на экран каждый раз после щелчка правой кнопки мыши на элементе управления.

Команды создания меню

Меню можно создать не только с помощью Конструктора меню, но и программным способом. Для этого существует целый ряд команд Visual FoxPro, которые приведены в табл. 12.1.

Таблица 12.1. *Команды создания меню*

Команда или функция	Описание
`ACTIVATE MENU | POPUP`	Отображает и активизирует горизонтальное | вертикальное меню
`CLEAR MENUS | POPUPS`	Удаляет из памяти определения всех горизонтальных | вертикальных меню
`CREATE MENU`	Открывает Конструктор меню
`DEACTIVATE MENU | POPUP`	Делает меню неактивным и невидимым, оставляя его в памяти
`DEFINE MENU | POPUP`	Создает горизонтальное | вертикальное меню
`DEFINE PAD | BAR`	Создает пункт в горизонтальном | вертикальном меню, созданном командой **DEFINE MENU | POPUP**
`EXTERNAL MENU`	Объявляет меню внешним
`HIDE MENU | POPUP`	Скрывает горизонтальное | вертикальное меню
`MODIFY MENU`	Открывает Конструктор меню для создания нового или редактирования имеющегося меню
`MOVE POPUP`	Перемещает вертикальное меню
`ON PAD | BAR`	Задает горизонтальное | вертикальное меню, которое активируется при выборе пункта горизонтального | вертикального меню

`ON SELECTION PAD | BAR`	Задает команду, выполняемую при выборе любого пункта заданного или любого горизонтального | вертикального меню
`ON SELECTION MENU | POPUP`	Задает команду, выполняемую при выборе любого пункта заданного или любого горизонтального | вертикального меню
`POP MENU | POPUP`	Возвращает определение меню из стека, помещенного туда командой **PUSH MENU | POPUP**
`PUSH MENU | POPUP`	Помещает в стек определение горизонтального | вертикального меню
`RELEASE PAD | BAR`	Удаляет из памяти заданный пункт или все пункты меню
`RELEASE MENUS | POPUPS`	Удаляет из памяти меню пользователя
`SET MARK OF MENU`	Задает или отменяет задание указателя меню
`SET SKIP OF`	Активирует или деактивирует меню или один из его пунктов
`SET SYSMENU`	Активирует или деактивирует системное меню VFP
`SHOW MENU | POPUP`	Отображает пользовательские меню
`SIZE POPUP`	Изменяет размер вертикального меню

Для примера создадим программным способом контекстное меню (рис. 12.12, листинг 12.1).

Рис. 12.12. Внешний вид меню, созданного программным способом

```
sel = SELECT()
IF ! USED("tipd")
   USE data\tipd IN 0 ORDER naim
   SELECT tipd
```

```
   SET ORDER TO naim
ELSE
   SELECT tipd
   SET ORDER TO naim
endif
GO TOP
STORE SPACE(100) TO m.simv
STORE SPACE(4) TO m.simv1
i = 1
DEFINE POPUP shortcut SHORTCUT RELATIVE  SCROLL FROM 1, 20
IF !EMPTY(m.simv)
   DEFINE BAR i   OF shortcut PROMPT " " + SUBSTR(ALLTRIM(m.simv),1,20) STYLE
"B" Picture "help.bmp" INVERT
   DEFINE BAR i+1 OF shortcut PROMPT "\-"
   i = i + 2
ENDIF

DO WHILE !EOF()
   IF ALLTRIM(m.simv) # ALLTRIM(tipd.naim)
      DEFINE BAR i       OF shortcut PROMPT "  " +
STR(tipd.kod,10)+"|"+SUBSTR(ALLTRIM(tipd.naim),1,30)
      ON SELECTION BAR i OF shortcut m.simv = PROMPT()
      i = i + 1
   EndIf

   IF !EOF()
      SKIP
   ELSE
      EXIT
   EndIf
EndDo

IF i # 3
   DEFINE BAR i + 1 OF shortcut PROMPT "\-"
EndIf
DEFINE BAR i + 2 OF shortcut PROMPT "  Новый тип докум." Picture "next.bmp"
ON SELECTION BAR i + 2 OF shortcut do form little_win2

ACTIVATE POPUP shortcut
m.simv1=SUBSTR(m.simv,1,AT("|",m.simv)-1)
m.simv= SUBSTR(m.simv,AT('|',m.simv)+1)
m.naim_doc=m.simv
SELECT (sel)
```

Меню в SDI- и MDI-формах

Чтобы разместить меню в SDI-форме, нужно произвести следующие действия:

1. Определить для формы свойство `ShowWindow=2`.
2. В свойстве `Init` формы добавить следующий код, вызывающий меню:

   ```
   DO SDIMenu.MPR WITH this, .T.
   ```

3. В методе `Destroy` ввести код

   ```
   RELEASE MENUS (THIS.NAME) EXTENDED
   ```

Разместить меню в немодальной MDI-форме, свойство `MDIFORM` которой установлено в "истину", возможно только в случае, если `DeskTop=.T.`

```
DEFINE MENU
IN [WINDOW] WindowName
```

размещает меню в окнах, определенных пользователем. Определите имя окна, в котором вы хотите разместить меню, как `WindowName`. Если вы пропустите опцию `IN WINDOW`, меню будет размещено в главном окне Visual FoxPro. Если есть активное определенное пользователем окно, то меню будет размещено в активном окне. Эта опция поддерживается только для `In-Top-Level` форм, у которых установлено `ShowWindow=2` или `Desktop=.T.`

Вот маленький пример (листинг 12.2).

```
Public o
o=CreateObject('mdimenu')
o.show()
Define Class mdimenu as Form
      MDIForm=.T.
      Desktop=.t.
      Procedure Init
      Define Menu menu1 Bar in Window (this.name)
      Define Pad mmm of menu1 prompt 'Exit'
      Define popup mmm margin relative
      On Pad  of mmm activate popup mmm
      Define Bar 1 of menu1 PROMPT 'exit'
      Activate Menu menu1 nowait
EndDefine
```

Кроме использования обычных и контекстных меню, можно создавать меню, используя и другие компоненты Visual FoxPro, например, `CommandGroup`, `TreeView` или `ListView`.

Примеры создания различных меню находятся на компакт-диске: CHAR12\menu.

Вызов меню

Сгенерированное меню обычно располагается в подкаталоге MENU проекта. Управление им производится, как правило, из главного файла. Для вызова меню используется команда

```
DO Namemenu.MPR
READ EVENTS
```

Команда READ EVENTS необходима для начала обработки событий. Указание на расширение файла mpr обязательно.

Вызов из меню методов формы

В отличие от других компонентов Visual FoxPro, меню не является объектом в том виде, в каком это понимается при объектно-ориентированном программировании. Тем не менее объектно-ориентированный подход применим и к меню. Каким же образом? Например, при необходимости обратиться к свойствам формы из меню используется свойство ActiveForm объекта _SCREEN.

При обращении к какому-либо элементу, находящемуся в контейнере, требуется указывать полный путь до него, состоящий из имени контейнера, символа "точка" и имени самого элемента. При этом контейнер может быть частью другого контейнера, и путь до него формируется по такому же правилу. В качестве контейнера самого высокого уровня может служить основное окно Visual FoxPro, к которому можно обратиться с помощью системной переменной _Screen.

Кроме того, существует возможность адресоваться к активному в данный момент элементу с помощью имени ActiveControl, а к активной форме — ActiveForm. Например, для того чтобы записать какую-то команду для элемента text1, расположенного в форме, с которой в настоящий момент происходит работа, можно использовать следующее имя: _Screen.ActiveForm.Text1.

В Visual FoxPro допустима адресация с помощью так называемых "относительных ссылок". При этом текущий объект можно назвать This, текущую форму — ThisForm, текущий набор форм — ThisFormSet. При работе с каким-либо из элементов, входящим в контейнер, можно сослаться на контейнер с помощью выражения This.Parent. Для записи команды, изменяющей какое-либо свойство объекта или выполняющей какой-либо метод, используется следующее правило:

```
<Имя объекта>.=[значение],
<Имя объекта>.<Метод>,
<Имя объекта>.<Событие>.
```

ActiveForm ссылается на активный объект формы.

```
Object.ActiveForm.Property [ = Setting]
```

или

```
Object.ActiveForm.Method
```

Таким образом можно определить любое свойство активной формы, например, `Caption`.

Ниже приводится пример кода, позволяющий из меню обращаться к свойствам формы (листинг 12.3).

```
SET SYSMENU TO
SET SYSMENU AUTOMATIC

DEFINE PAD m_file OF _MSYSMENU PROMPT "\<Файл" COLOR SCHEME 3 ;
  KEY ALT+F, ""
DEFINE PAD m_Edit OF _MSYSMENU PROMPT "\<Правка" COLOR SCHEME 3 ;
  KEY ALT+E, ""

ON PAD m_file OF _MSYSMENU ACTIVATE POPUP p_File
ON PAD m_Edit OF _MSYSMENU ACTIVATE POPUP p_Edit

DEFINE POPUP p_file MARGIN RELATIVE SHADOW COLOR SCHEME 4
DEFINE POPUP p_edit MARGIN RELATIVE SHADOW COLOR SCHEME 4

DEFINE POPUP p_file MARGIN RELATIVE SHADOW COLOR SCHEME 4
DEFINE BAR 1 OF p_file PROMPT "\<Закрыть форму" ;
  KEY CTRL+F4, "Ctrl+F4" ;
  SKIP FOR  !_SCREEN.ActiveForm.Closable ;
  PICTURE "pict\close.bmp"
DEFINE BAR 2 OF p_file PROMPT "\-"
DEFINE BAR _MFI_SAVE OF p_file PROMPT "\<Сохранить" ;
  PICTRES _MFI_SAVE
DEFINE BAR _MFI_SAVAS OF p_file PROMPT "\<Сохранить как..."
DEFINE BAR 5 OF p_file PROMPT "\-"
DEFINE BAR 6 OF p_file PROMPT "\<Настройка принтера" ;
  PICTURE "pict\print.bmp"
DEFINE BAR _MFI_PGSET OF p_file PROMPT "\<Формат страницы" ;
  PICTRES _MFI_PGSET
DEFINE BAR 8 OF p_file PROMPT "\<Печать" ;
  SKIP FOR  (TYPE("_SCREEN.ActiveForm.Print") ==   "U") ;
  OR !_SCREEN.ActiveForm.Print ;
  PICTRES _MFI_SYSPRINT
DEFINE BAR 9 OF p_file PROMPT "\-"
DEFINE BAR 10 OF p_file PROMPT "Завершить \<работу" ;
  KEY ALT+F4, "Alt+F4"

ON SELECTION BAR 1 OF p_file _SCREEN.ActiveForm.Release()
ON SELECTION BAR 6 OF p_file oApp.SetUp_Printer()
```

```
ON SELECTION BAR 8 OF p_file _SCREEN.ActiveForm.PReport()
ON SELECTION BAR 10 OF p_file OnShutdown()

DEFINE POPUP p_Edit MARGIN RELATIVE SHADOW COLOR SCHEME 4
DEFINE BAR _med_undo OF p_Edit PROMPT "\<Отказ" ;
  KEY CTRL+Z, "Ctrl+Z" ;
  PICTRES _med_undo
  DEFINE BAR _med_redo OF p_Edit PROMPT "\<Повтор" ;
  KEY CTRL+R, "Ctrl+R" ;
  PICTRES _med_redo
DEFINE BAR 3 OF p_Edit PROMPT "\-"
DEFINE BAR _med_cut OF p_Edit PROMPT "\<Вырезать" ;
  KEY CTRL+X, "Ctrl+X" ;
  PICTRES _med_cut
DEFINE BAR _med_copy OF p_Edit PROMPT "\<Скопировать" ;
  KEY CTRL+C, "Ctrl+C" ;
  PICTRES _med_copy
DEFINE BAR _med_paste OF p_Edit PROMPT "Вс\<тавить" ;
  KEY CTRL+V, "Ctrl+V" ;
  PICTRES _med_paste
DEFINE BAR 7 OF p_Edit PROMPT "\-"
DEFINE BAR _med_slcta OF p_Edit PROMPT "Вы\<делить все" ;
  KEY CTRL+A, "Ctrl+A"
```

Свойства меню

Формируется в дизайнере меню — это не окончательное меню, а лишь некий макет меню. Из этого макета впоследствии формируется окончательное меню, когда вы выбираете пункт системного меню **Menu** | **Generate**. При этом формируется файл MPR, который является ничем иным, как программным кодом созданного вами меню. Этим можно воспользоваться. Для чего?

В дополнение к настраиваемым с помощью Конструктора меню функциям можно отдельно настраивать некоторые опции, например, появление символа указателя возле пункта меню. Для отображения или очистки этого символа служит команда `SET MARK OF`, которая имеет несколько версий синтаксиса.

```
SET MARK OF MENU MenuBarName TO lExpression1
SET MARK OF POPUP MenuName1 TO lExpression2
SET MARK OF BAR nMenuItemNumber OF MenuName2 TO lExpression3
```

Если логическое выражение, указанное в предложении `TO`, оценивается в "истину" (`.T.`), то символ отметки отображается около каждого названия меню. Если `lExpression1` оценивается в "ложь" (`.F.`), символ отметки очищается у каждого пункта меню.

Считается, что тип и размер шрифта в меню невозможно изменить, поскольку он берется из системных настроек Windows. Однако есть возможность обойти это ограничение, правда, эта возможность не касается основной линейки меню.

Выберите нужный вам пункт меню (исключая пункт основной линейки меню) и нажмите кнопку в столбце **Options**.

В разделе **SKIP FOR** напишите примерно следующее:

```
.F. FONT "Arial Cyr",20 STYLE "B"
```

После генерации меню получим в итоговом файле MPR что-то вроде

```
DEFINE BAR 1 OF Primer PROMPT "Пример" ;
SKIP FOR .F. "Arial Cyr",20 STYLE "B"
```

В режиме `Preview` в самом дизайнере меню вы не увидите результатов этого "хакерского" трюка. Результат будет виден только при запуске самого меню. Если по каким-либо причинам вы не хотите использовать раздел **SKIP FOR** для указания шрифта, то примерно то же самое можно сделать в разделе **Message**. Только там код будет несколько другим

```
'' FONT "Arial Cyr",20 STYLE "B"
```

После генерации меню получим в итоговом файле MPR что-то вроде

```
DEFINE BAR 1 OF пример PROMPT "Пример" ;
MESSAGE '' "Arial Cyr",20 STYLE "B"
```

Разумеется, для задания шрифта следует использовать только один раздел. То есть либо раздел `SKIP FOR`, либо раздел **Message**. Не надо указывать шрифт в обоих разделах. Это вызовет синтаксическую ошибку в итоговом меню. Будет одновременно две опции `FONT` у одного пункта.

Начиная с версии Visual FoxPro 9, в синтаксисе команды `DEFINE BAR` у опции `FONT` появился третий необязательный параметр — `nFontCharSet`. То есть можно сделать шрифт меню относительно независимым от региональных настроек системы. Например, для русского языка в разделе **SKIP FOR** это будет выглядеть примерно так:

```
.F. FONT "Arial",20,204 STYLE "B"
```

Более того, генератор меню версии Visual FoxPro 9 стал более "интеллектуальным". Если у него получается конструкция вида `SKIP FOR .F.`, то она просто исключается, и меню будет иметь следующий вид:

```
DEFINE BAR 1 OF Primer PROMPT "Пример" ;
"Arial",20,204 STYLE "B"
```

Для строки меню можно определить процедуры, которые будут выполняться перед запуском меню, а также после выхода из меню. Для того чтобы определить процедуры, выполните команду главного меню **View | General Options**. На экране появится окно **General Options** (рис. 12.12).

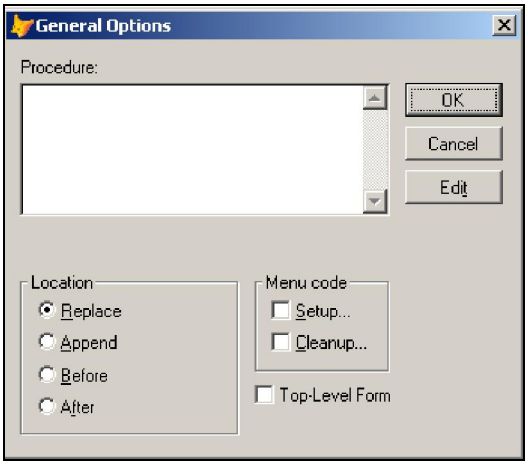

Рис. 12.13. Диалоговое окно **General Options**

Группа переключателей **Location** позволяет определить варианты расположения строки меню:

- **Replace** — созданное меню заменяет главное меню Visual FoxPro;
- **Append** — созданное меню добавляется к главному меню VFP;
- **Before** — созданное меню будет вставлено перед указанным пунктом меню VFP;
- **After** — созданное меню будет размещено за указанным пунктом меню VFP.

В области **Procedure** можно написать код, который будет выполняться при активизации строки меню.

Флажки **Setup** и **Cleanup** используются для открытия окна редактирования процедуры, выполняемой соответственно перед запуском сгенерированного меню, и после завершения его работы.

Восстановление главного меню Visual FoxPro

Существует такой основополагающий принцип программирования, который называется "Не наследи!" Этот принцип актуален не только для Visual FoxPro, но и для любых других языков программирования и заключается он в том, что перед окончанием работы программы программист должен вернуть все на свое место: закрыть и удалить временные файлы и таблицы, закрыть таблицы и базы данных, убрать свое меню и восстановить главное меню. Как это сделать?

Команда **DEACTIVATE MENU | POPUP** удаляет ненужные более меню с экрана. Чтобы удалить ненужные меню из памяти, следует использовать команды **RELEASE** или **CLEAR MENUS | POPUP**.

Для восстановления главного меню VFP служит команда **SET SYSMENU TO DEFAULT**.