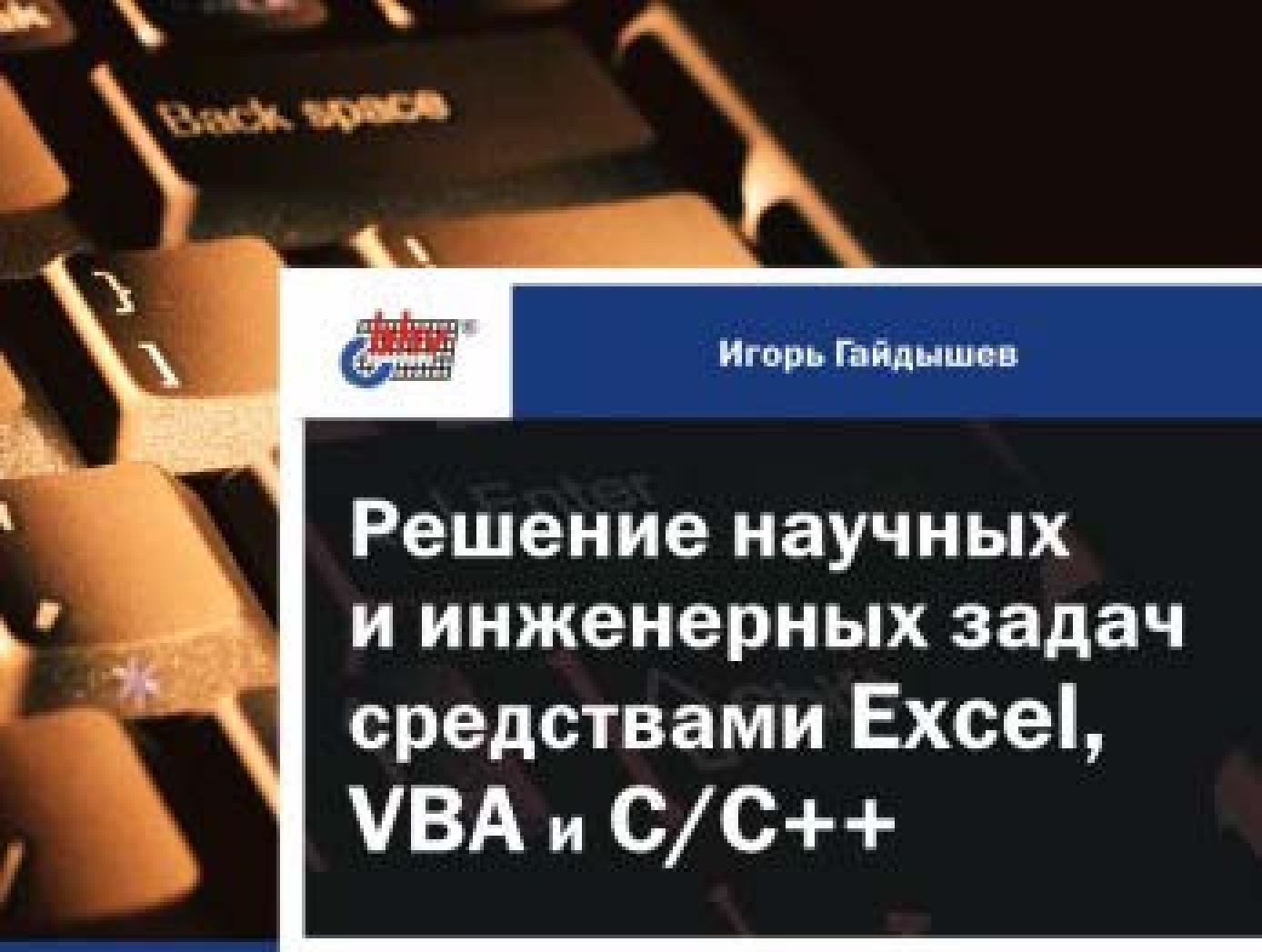


Игорь Гайдышев

Решение научных и инженерных задач средствами Excel, VBA и C/C++

+ CD-ROM

- Обработка и анализ данных
- Статистический контроль качества
- Математическое моделирование
- Распознавание образов
- Создание макросов XLM и VBA
- Разработка DLL и XLL для MS Excel

МАСТЕР_РЕШЕНИЙ_

УДК 681.3.068+800VBA,C++
ББК 32.973.26-018.1
Г14

Гайдышев И. П.

Г14 Решение научных и инженерных задач средствами Excel, VBA и C/C++. — СПб.: БХВ-Петербург, 2004. — 512 с.: ил.

ISBN 5-94157-477-0

Книга посвящена практическим вопросам программирования. Подробно рассматривается технология создания недорогого и максимально адаптированного для пользователей всех квалификаций программного обеспечения для анализа данных и математического моделирования. Изложение ведется как в применении к официальным, так и — впервые в литературе — альтернативным средствам разработки, причем последним уделяется большое внимание. На прилагаемом компакт-диске даются полные исходные тексты программ, проекты для различных средств разработки и дистрибутивы программ и дополнительных модулей для электронных таблиц. Все программы тщательно протестированы. Книга будет полезна в качестве практического пособия преподавателям и студентам, программистам, научным работникам и инженерам-исследователям.

Для программистов

УДК 681.3.068+800VBA,C++
ББК 32.973.26-018.1

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Игорь Шишигин*
Зав. редакцией	*Григорий Добин*
Редактор	*Елена Кашлакова*
Компьютерная верстка	*Татьяны Валерьяновой*
Корректор	*Евгений Камский*
Оформление серии	*Via Design*
Дизайн обложки	*Игоря Цырульникова*
Зав. производством	*Николай Тверских*

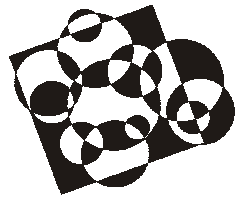
Лицензия ИД № 02429 от 24.07.00. Подписано в печать 19.03.04.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 41,28.
Тираж 3000 экз. Заказ №
"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

Гигиеническое заключение на продукцию, товар № 77.99.02.953.Д.001537.03.02
от 13.03.2002 г. выдано Департаментом ГСЭН Минздрава России.

Отпечатано с готовых диапозитивов
в Академической типографии "Наука" РАН
199034, Санкт-Петербург, 9 линия, 12.

ISBN 5-94157-477-0

Содержание



ГЛАВА 1

Разработка элементов пользовательского интерфейса на языке VBA

«Поскольку сегодняшний пользователь ЭВМ обычно имеет профессиональные интересы, выходящие за рамки программирования, то пользовательский интерфейс должен стимулировать эти интересы, а не сдерживать их».
А. Стивенс
«У Вас есть выбор, на каком языке писать ту или иную часть приложения. Так, пользовательский интерфейс приложения Вы, скорее всего, будете создавать на Microsoft Visual Basic, но прикладную логику лучше всего реализовать на C++.»
Дж. Рихтер

Для работы с материалами главы понадобятся лицензионные средства разработки и тестирования программ:

- Microsoft Excel 97 или выше;
- Microsoft Excel Developer's Kit версии 8 или выше.

Потребуется знание VBA.

Решаемая проблема

Современная программа, предназначенная для обработки научных, финансовых, коммерческих или иных данных, предоставляет возможности для

хранения массивов числовой информации (учитывая возможность преобразования в числовой других видов информации) и программные средства для их обработки. Такой концепции наиболее полно отвечают электронные таблицы, оснащенные дополнительными аналитическими инструментами. Без потери общности, мы можем говорить о научных программах, потому что результаты наших исследований применимы для решения широкого круга задач.

Лучшие специализированные программы статистического анализа данных:

- NCSS — Number Cruncher Statistical Systems (NCSS, **www.ncss.com**),
- SPSS (SPSS Inc., **www.spss.com**),
- STADIA (НПО «Информатика и компьютеры», **statsoft.msu.ru**)
- STATISTICA (StatSoft Inc., **www.statsoft.com**),
- SYSTAT (SYSTAT Software Inc., **www.systat.com**)

и другие — построены также на концепции электронных таблиц. Однако средства манипуляции данными в перечисленных программах несравнимо более ограничены, чем в любых электронных таблицах. Это и понятно — производителям лучших программ анализа данных вряд ли по силам конкурировать в части интерфейса с производителями любых электронных таблиц. У этих программных продуктов — разные задачи и группы пользователей, пересекающиеся незначительно. Однако мы покажем, что вполне возможно создать программный продукт, сосредоточившись на реализации алгоритмов анализа данных и воспользовавшись всеми доступными средствами, предлагаемыми лучшими в мире электронными таблицами.

Остановимся кратко на истории электронных таблиц производства Microsoft, с ними нам предстоит работать на протяжении всей книги. Под руководством Ч. Саймони (Charles Simonyi), перешедшим в Microsoft из исследовательского центра Xerox PARC, в 1981 г. были разработаны электронные таблицы Multiplan для операционной системы MS–DOS, как, годом позднее, и текстовый процессор Word для MS–DOS. Программный продукт Microsoft Multiplan для Apple Macintosh был анонсирован в 1984 г. Программа Multiplan была написана на языке программирования C, чем была обеспечена ее легкая переносимость на микрокомпьютеры с различной архитектурой. Однако эти электронные таблицы оказались менее совершенными, чем продукты конкурентов.

Электронные таблицы (иначе, табличный процессор) Microsoft Excel были разработаны в 1985 г. для платформы Macintosh, и это было стратегически правильным решением. Графические возможности Apple Macintosh и Microsoft Excel обеспечили успех и тому, и другому продукту. В 1987 г. появилась версия Microsoft Excel 2 для операционной системы Microsoft Windows.

Электронные таблицы Microsoft Excel 4 были впервые оснащены специальной версией языка программирования Visual Basic Applications Edition.

Первой 32-разрядной версией стали электронные таблицы Microsoft Excel 7 (другое название Microsoft Excel 95), выпущенные в 1995 г. Версия 8 (Microsoft Excel 97) была разработана в 1997 г. и получила новый интерфейс Visual Basic for Application (VBA), единый для всех компонентов Microsoft Office 97. Благодаря Microsoft Excel и VBA, практически на каждом современном компьютере в распоряжении пользователя имеется мощная система программирования. Компонентом пакета офисных приложений Microsoft Office 2000 стала версия 9 (Microsoft Excel 2000), Microsoft Office XP — версия 10 (Microsoft Excel 2002), Microsoft Office 2003 — версия 11 (Microsoft Excel 2003). Пакет Microsoft Excel доступен и в качестве отдельного продукта для Microsoft Windows и Apple Macintosh.

Представленные в книге дополнительные модули для электронных таблиц Microsoft Excel корректно работают в локализованных версиях Microsoft Excel 97, 2000, 2002 и 2003.

Мы хотим, чтобы с помощью сведений данной главы читатель любой квалификации мог разобраться (или использовать как готовый рецепт, не вникая в подробности) в VBA и легко, как и предполагает VBA, создавать программные проекты любой сложности. Здесь представлен надежно функционирующий каркас полезного приложения, который можно модифицировать или наращивать, по своему усмотрению.

Вывод графиков средствами VBA в настоящей главе не упоминается. Он отдельно рассмотрен в *главе* 11.

Редактор Visual Basic

Разработка программы на языке VBA начинается одинаково для всех представленных в книге проектов, использующих VBA. Все примеры приводятся для русских версий Excel из комплекта офисных приложений Microsoft Office 2000 Professional, под управлением операционной системы MS Windows 98 Second Edition[1], или из комплекта офисных приложений Microsoft Office XP Standard под управлением операционной системы MS Windows XP Home Edition.

После запуска электронных таблиц Microsoft Excel, нажмите комбинацию клавиш <Alt>+<F11> либо выберите из меню Excel последовательность пунктов **Сервис ▸ Макрос ▸ Редактор Visual Basic**. Появится окно интегрированной среды разработки приложений Visual Basic for Application, как на рис. 1.1.

---

[1] *О типах операционных систем см. приложение 4.*

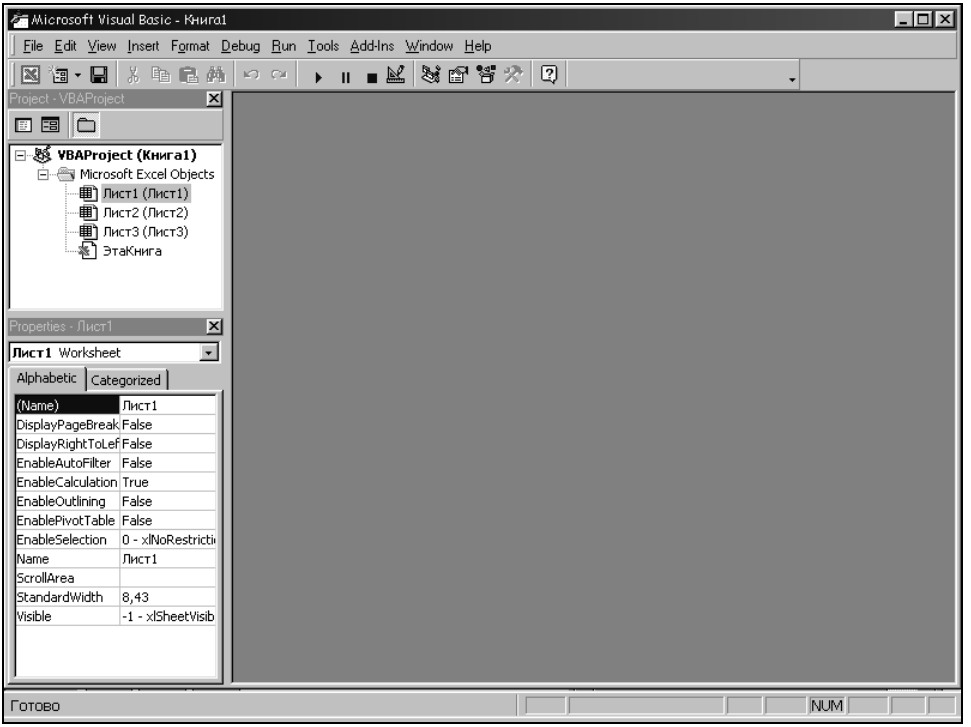


Рис. 1.1. Интегрированная среда разработки приложений Visual Basic for Application

Нажмите правую кнопку мыши в любом месте окна **Project** и выберите из появившегося меню **Insert** ▸ **Module**. К проекту будет добавлен новый модуль **Module1** (потом можно переименовать), а в появившееся окно модуля введите исходные тексты программы (макроса).

Следует предусмотреть систему меню для легкого доступа к прикладным алгоритмам и диалоговые панели (формы) для выбора или ввода различных параметров, управляющих работой этих прикладных алгоритмов. В *главах 3* и *5* дополнительно приводятся столь же удобные, хотя и функционально ограниченные, решения, позволяющие обойтись не только без отдельной системы меню, но иногда и вообще без кода VBA, воспользовавшись языком макросов Microsoft Excel.

Чтобы придать нашим разработкам вид серьезных программ, в дальнейшем, мы предполагаем, что разработанные программы будут называться AtteStat и ME.com. Разработка будет модульной, подобно настоящей программе анализа данных. Это, на самом деле, недалеко от истины — после добавления всех модулей Microsoft Excel «превратится» в настоящую программу анализа данных.

Создание меню средствами VBA

Добавление пользовательских меню, обеспечивающих удобный доступ к прикладным алгоритмам модульной программы анализа данных, работающей в среде электронных таблиц Microsoft Excel, средствами VBA не является трудной задачей. Язык визуального программирования VBA как и любой современный развитый язык программирования, предлагает несколько решений. На рис. 1.2 показано, что, примерно, мы хотим получить.

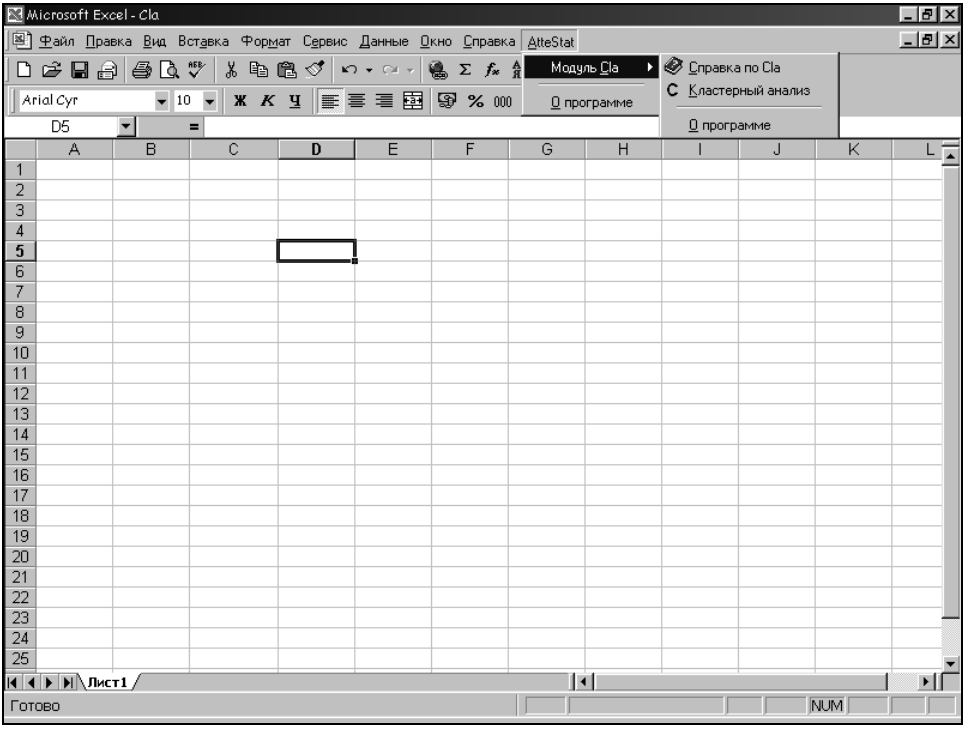


Рис. 1.2. Рабочая книга с пользовательским меню

Наша программа должна добавлять пункт меню к существующему меню Microsoft Excel. Так как программа модульная, будет хорошо, если каждый установленный модуль станет добавлять свое меню к общему меню нашей программы. Это сэкономит место на экране и обеспечит удобный доступ к функциям каждого из установленных модулей.

Программные манипуляции с пользовательскими меню удобно делать в функции `Auto_Open`. Имя функции `Auto_Open` зарезервировано Microsoft Excel в качестве стандартного имени. Это значит, что при загрузке рабочей книги или шаблона, содержащих эту функцию, Microsoft Excel автоматиче-

ски исполняет ее код. В конце главы даются рекомендации по управлению выполнением, в обход любого установленного уровня безопасности Microsoft Excel, что вызвано не пренебрежением к информационной безопасности, а обеспечением требуемой функциональности прикладной программы.

Объекты меню

На листинге показана начальная часть функции `Auto_Open` с описаниями компонентов меню, общих для всех модулей программы AtteStat.

Листинг 1.1

```
Sub Auto_Open ()
 '
 ' Загрузка меню
 '
 Dim WorksheetMenuBar As CommandBar   ' Основная панель
 Dim CstmCtrl As CommandBarControl    ' Пункт основного меню
 Dim AtteStatMenu As CommandBarPopup  ' Выпадающее меню AtteStat
 Dim AtteStat As CommandBarControl    ' Пункт меню AtteStat
 Dim MainControl As CommandBarControl ' Пункт меню модуля
 Dim Exist As Boolean                 ' Флаг существования AtteStat
```

Для построения меню используются следующие стандартные средства Microsoft Office из коллекции `CommandBarControls`:

- объект `CommandBar` — панель меню (панель команд);
- объект `CommandBarPopup` — панель выпадающего меню;
- объекты класса `CommandBarControl` — отдельные пункты меню.

Нам необходимо создать общее меню для всех модулей нашей прикладной программы, затем добавить к нему меню конкретного модуля. Идея достаточно проста: каждый модуль, если он загружается первым, создает пункт меню **AtteStat** (или **ME.com**) и добавляет свое собственное меню. Если он загружается не первым, то пункт меню **AtteStat** уже существует, его создавать не нужно, а только добавить меню модуля к существующему меню **AtteStat**.

Поэтому, перед добавлением нового меню модуля, следует решить важную задачу. Необходимо проверить, существует ли в меню Excel пункт меню **AtteStat**, который мы собираемся добавить, так как Microsoft Excel допускает существование в меню двух и более одинаковых пунктов. Для проверки существования

конкретного пункта меню как раз предназначен флаг `Exist` (см. листинг 1.1), а сама проверка выполняется, как показано на листинге 1.2.

Листинг 1.2

```
'
' Проверка наличия меню AtteStat
'
Set WorksheetMenuBar = CommandBars.ActiveMenuBar
Exist = False
For Each CstmCtrl In WorksheetMenuBar.Controls
  If CstmCtrl.Caption = "&AtteStat" Then
    Exist = True
    Set AtteStat = CstmCtrl
    Exit For
  End If
Next CstmCtrl
```

Данный фрагмент программы установит (поднимет) флаг `Exist` в положение `True`, если пункт меню `&AtteStat` уже существует (знак `&` обозначает последующий символ «горячей» клавиши, если таковые используются в меню программы; использование знака `&` в поиске пункта меню обязательно). Для нас это будет означать, что создавать новый пункт меню не нужно, а следует добавить пункт загружаемого в данный момент модуля в уже существующее меню. Если флаг `Exist` оставлен в положении по умолчанию `False`, то создадим сначала новое меню **AtteStat**. Свойства меню описаны в следующем разделе.

Как принято в современном программировании, минимальный набор пунктов меню должен содержать, помимо вызова прикладной части программы, также вызов справочной системы и информационного окна (рис. 1.3), содержащего сведения о названии программы, номере версии, обладателе авторских прав и юридических последствиях их коварного нарушения.

Рис. 1.3. Информационное окно интегратора модульной программы ME.com

Свойства меню

На листинге 1.3 показан фрагмент программы, формирующий меню модуля кластерного анализа, логика которого описана в *главе 7*, в зависимости от того, существует ли меню AtteStat на момент исполнения данного кода. Содержательные комментарии объясняют суть выполняемых операторов.

Листинг 1.3

```
'
' Формирование меню модуля в составе AtteStat
'
Dim ClaMenuBar As CommandBar        ' Панель меню модуля
Dim ClaMenu As CommandBarPopup      ' Выпадающее меню модуля
Dim ClaControl As CommandBarControl ' Пункт меню модуля
If Not Exist Then
  '
  ' Формирование нового меню AtteStat и добавление к нему меню модуля
  '
  Set AtteStatMenu =
WorksheetMenuBar.Controls.Add(Type:=msoControlPopup, _
                    Temporary:=True)
  AtteStatMenu.Caption = "&AtteStat"
  Set ClaMenu = AtteStatMenu.Controls.Add(Type:=msoControlPopup, _
                Temporary:=True)

  Set MainControl = AtteStatMenu.Controls.Add(Type:=msoControlButton)
  MainControl.BeginGroup = True
  MainControl.Caption = "&О программе"
  MainControl.OnAction = "Cla.MakeAboutMain"
Else
  '
  ' Добавление меню модуля к существующему меню AtteStat
  '
  Set ClaMenu = AtteStat.Controls.Add(Type:=msoControlPopup, _
                Temporary:=True, Before:=1)
End If
ClaMenu.Caption = "Модуль &Cla"
```

```
'
' Формирование меню модуля
'
Set ClaControl = ClaMenu.Controls.Add(Type:=msoControlButton)
ClaControl.FaceId = 983
ClaControl.Caption = "&Справка по Cla"
ClaControl.OnAction = "Cla.MakeHelp"

Set ClaControl = ClaMenu.Controls.Add(Type:=msoControlButton)
ClaControl.FaceId = 82
ClaControl.Caption = "&Кластерный анализ"
ClaControl.OnAction = "Cla.RunSolution"

Set ClaControl = ClaMenu.Controls.Add(Type:=msoControlButton)
ClaControl.BeginGroup = True
ClaControl.Caption = "&О программе"
ClaControl.OnAction = "Cla.MakeAbout"
End Sub
```

Подробные описания методов и свойств даны во встроенной документации VBA и в указанных в конце главы источниках. В данном фрагменте используются свойства:

- `FaceId` — номер изображения, появляющегося перед пунктом меню;
- `Caption` — заголовок пункта меню;
- `OnAction` — метод, исполняемый при выборе пункта меню;
- `BeginGroup` — разделитель.

Сделаем подробные пояснения относительно свойства `FaceId`. Полный список изображений и соответствующих им номеров не является одинаковым для каждого компьютера, а зависит от установленных на нем программ. Соответствие номера конкретному изображению поможет получить показанная на листинге 1.4 компактная утилита (вспомогательная программа) в коде VBA. Макрос создает два пункта меню (можно изменить их количество, по своему усмотрению), которые в листинге 1.4. мы назвали **1–300** и **301–600** (в примере на компакт-диске пунктов меню немного больше). В каждом пункте меню выводится по 300 изображений с номерами, которые им соответствуют. Программисту остается выбрать понравившееся изображение и указать его номер в свойстве `FaceId`.

Листинг 1.4

```
Sub Auto_Open()
 Dim TestIDMenuBar As Object
 Dim TestIDMenu As Object
 Dim TestIDControl As Object

 Set TestIDMenuBar = CommandBars.ActiveMenuBar
 Set TestIDMenu = TestIDMenuBar.Controls.Add(Type:=msoControlPopup, _
                  Temporary:=True)
 TestIDMenu.Caption = "1-300"
 For I = 1 To 300
   Set TestIDControl = TestIDMenu.Controls.Add(Type:=msoControlButton)
   TestIDControl.FaceId = I
   TestIDControl.Caption = Str(I)
 Next I

 Set TestIDMenuBar = CommandBars.ActiveMenuBar
 Set TestIDMenu = TestIDMenuBar.Controls.Add(Type:=msoControlPopup, _
                  Temporary:=True)
 TestIDMenu.Caption = "301-600"
 For I = 301 To 600
   Set TestIDControl = TestIDMenu.Controls.Add(Type:=msoControlButton)
   TestIDControl.FaceId = I
   TestIDControl.Caption = Str(I)
 Next I
End Sub
```

Рабочая книга, содержащая данный макрос, может медленно загружаться на компьютерах с низкой мощностью процессора. В этом случае, можно уменьшить значения счетчика I и создать, например, не 2 пункта меню: от 1 до 300 и от 301 до 600, как в показанном листинге, а один — от 1 до 100. Затем можно проверить, опять однократно, вариант от 101 до 200, и т. д.

Методы меню

Свойства `OnAction` в листинге 1.3 указывают имена методов, которые должны быть исполнены при выборе пользователем того или иного пункта меню. Это могут быть стандартные методы класса `Application` или специально составленные пользовательские функции, как показано на листинге 1.5.

Листинг 1.5

```
Private Sub MakeHelp()
 Application.Help "Cla.hlp"
End Sub
Private Sub MakeAboutMain()
 MainUserForm.Show
End Sub
Private Sub RunSolution()
 ClaUserForm1.Show
End Sub
Private Sub MakeAbout()
 ClaUserForm2.Show
End Sub
```

Метод `Application.Help` заданием параметров вызывает соответсвующую справочную систему, с указанием полного пути к тому каталогу, где находится файл справки. Программисту нужно заранее позаботиться о том, чтобы файл справки уже находился в указанном месте (или в каталоге по умолчанию) до того, как пользователь решит им воспользоваться. Метод одинаково хорошо работает и со справочной системой в формате HLP, и в формате HTML. О создании справочных систем см. *главу 12*. Там же подробно описано местонахождение файлов различных типов, применяемых в наших разработках.

Метод `MainUserForm.Show` вызывает форму (об их проектировании см. следующий раздел) с информацией о проекте AtteStat. Данная форма — общая для всех модулей, составляющих одну программу.

Метод `ClaUserForm1.Show` вызывает диалоговую панель, предназначенную для выбора параметров расчета и запуска прикладных алгоритмов (рис. 7.2). В нашем проекте, помимо общей формы с информацией о программном обеспечении AtteStat, каждый модуль имеет свою собственную информационную форму.

Метод `ClaUserForm2.Show` вызывает форму с информацией о модуле (рис. 1.3). Данная форма — специфическая для каждого разработанного в рамках нашего проекта модуля.

Все представленные в данном модуле функции описаны как `Private`, чтобы исключить их случайный вызов из листа рабочей книги или других модулей и исключить конфликты имен в случае, если в других модулях окажутся функции с такими же именами. Если функции специально предназначены для вызова за пределами содержащего их модуля, они должны быть описаны как `Public`.

Создание диалоговых панелей

Для обеспечения удобства пользователей и упрощения работы программиста, в VBA предусмотрена возможность быстрого создания диалоговых панелей (форм). На рис. 1.4 показано окно редактора Visual Basic в режиме проектирования формы. Неопытным в программировании читателям не следует пугаться слишком сложной картины. Во-первых, редактор весьма удобен. Во-вторых, сложность формы зависит от сложности решаемой задачи. В-третьих, на прилагаемом к книге компакт-диске содержатся готовые коды всех рассмотренных модулей, которые можно использовать в качестве основы для собственных разработок.

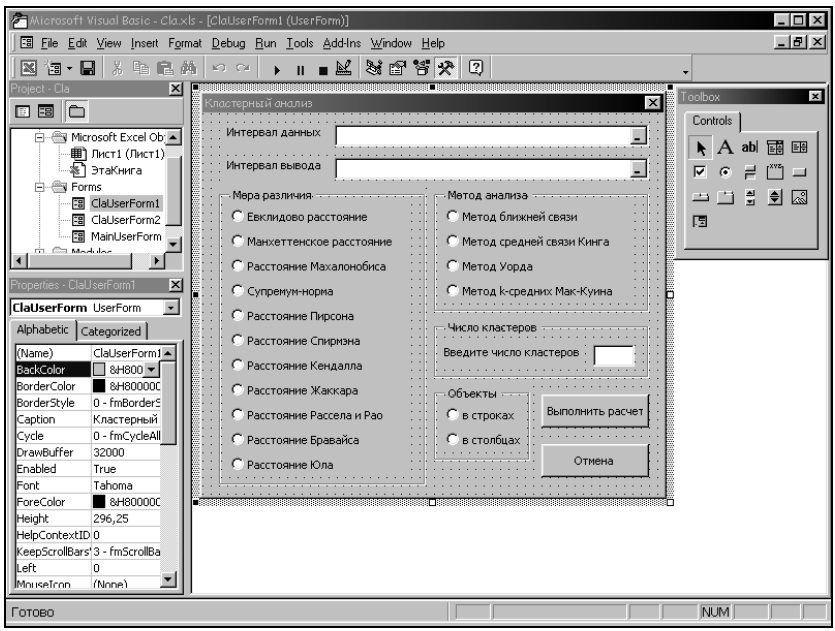


Рис. 1.4. Окно редактора Visual Basic в режиме проектирования формы

Основные элементы управления VBA

Для создания новой формы нажмите правую кнопку мыши в любом месте окна **Project** и выберите из появившегося меню **Insert ▸ UserForm**. В проект будет добавлена новая форма UserForm1. Суть визуального проектирования формы заключается в помещении на поле формы путем перенесения мышью элементов управления из палитры элементов управления **Toolbox** редактора VBA (рис. 1.5), определении свойств этих элементов (в специальном окне) и привязке к ним кода VBA, для обработки событий, возникающих при выборе пользователем тех или иных элементов управления.

Элементы управления подробно описаны во встроенной справочной системе и в литературе. Чтобы не пересказывать содержание справочной системы, которая всегда под рукой у VBA-программиста, отметим только те элементы управления, которые используются в наших расчетных программах, (в алфавитном порядке):

- `CheckBox` — флажок (кнопка независимого выбора);
- `ComboBox` — элемент управления, отображающий список величин и позволяющий выбрать одну их них;
- `CommandButton` — командная кнопка;
- `Frame` — рамка;
- `Image` — изображение;
- `Label` — метка (надпись, статический текст);
- `MultiPage` — многостраничный элемент управления;
- `OptionButton` — переключатель (кнопка зависимого выбора, радиокнопка);
- `RefEdit` — поле ссылки на ячейки листа рабочей книги электронной таблицы;
- `TextBox` — поле ввода (окно редактирования, «текстовое» поле).

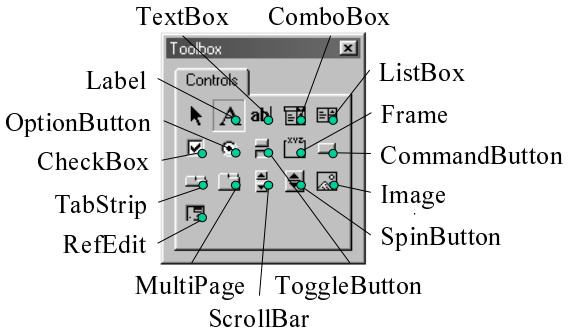


Рис. 1.5. Палитра элементов управления Toolbox редактора VBA электронных таблиц Microsoft Excel

Флажок `CheckBox` позволяет пользователю выбрать ту или иную опцию, связанную с данным флажком. Он имеет два состояния: `False` — не установлен (это значение по умолчанию), `True` — флажок установлен. Состояние флажка передается в пользовательскую программу через свойство `Value`.

Элемент управления `ComboBox` отображает список величин заданного типа и позволяет пользователю выбрать только одну из этих величин. Дальнейшая обработка выбранного варианта зависит от логики программы. Пример применения `ComboBox` см. в *главе 3*.

Командная кнопка `CommandButton` служит для запуска той или иной процедуры, например, расчетного модуля программы. К командной кнопке можно привязать и другие события, например, установку всех флажков формы в значение `True` или `False` (см. *главу 4*). Можно привязать к кнопке вызов справочной системы или возможность выхода из формы.

Рамка `Frame` обычно применяется для визуального объединения некоторых элементов управления в логически связанную группу. Помимо графического оформления, рамка может выполнять еще одну задачу. Так, элементы управления `OptionButton`, визуально объединенные в рамку, не требуют обязательного составления программистом функций `OptionButton_Click` обработки события типа `Click` для каждого элемента управления `OptionButton`. Достаточно только инициализации формы. Обработка (установка значения `True` только для выбранной кнопки и `False` для всех остальных) будет выполнена автоматически. Если форма имеет несколько рамок, и выбор `OptionButton` должен производиться только для одного элемента из всех рамок, данное преимущество теряется, и программисту придется составлять подпрограммы обработки для каждого элемента.

Элемент управления `Image` служит для включения в форму прямоугольного графического изображения. Данный элемент управления может служить как для улучшения внешнего вида формы, так и для отображения на экране различной графической информации.

Метка `Label` служит для включения в форму надписей (статического текста), служащих для различных целей. Например, с помощью метки `Label` можно ввести в форму различные пояснения относительно элементов управления формы, а также информацию о владельце авторских прав на программу.

Элемент управления `MultiPage` в наших проектах применяется в *главе 7* при создании модуля распознавания образов с обучением. Данный элемент позволяет отобразить в компактной форме большое число элементов управления. В упомянутой главе элемент `MultiPage` позволяет в удобной и компактной форме отобразить как обучающие, так и распознающие алгоритмы.

Переключатель `OptionButton` обеспечивает зависимый выбор (обычно одного из нескольких возможных). Например, переключатель можно применять для выбора одного из нескольких методов расчета либо для выбора техники расчета (см. проекты, представленные в *главах 6–9* и *11*).

Основным элементом управления, обеспечивающем передачу данных из листа рабочей книги электронных таблиц в программу VBA, является поле `RefEdit`. Использование данного поля открывает доступ к широким возможностям Microsoft Excel. С помощью поля `RefEdit` пользователь получает возможность ввести или указать методом протаскивания курсора интервал ячеек Microsoft Excel, содержащих данные для расчета. Указанный в поле `RefEdit` интервал ячеек может находиться не обязательно в активном в данным момент рабочем листе, а в любом листе рабочей книги. Данная приятная особенность делает ненужным дополнительное указание листа (как сделано в стандартном пакете анализа Microsoft Excel), если, например, вывод результатов расчета предполагается в листе, отличном от листа, содержащего исходные данные, или исходные данные берутся из разных листов рабочей книги.

В большинстве известных программ статистического анализа, вместо конструкции, подобной `RefEdit`, для выбора параметров или переменных анализа применяется совершенно другой прием. На рис. 1.6 показано окно выбора параметров программы статистического анализа данных SYSTAT. Пользователю предлагается из списка доступных выбрать переменные для анализа, причем обоснование данного выбора может вызвать у пользователя массу вопросов.

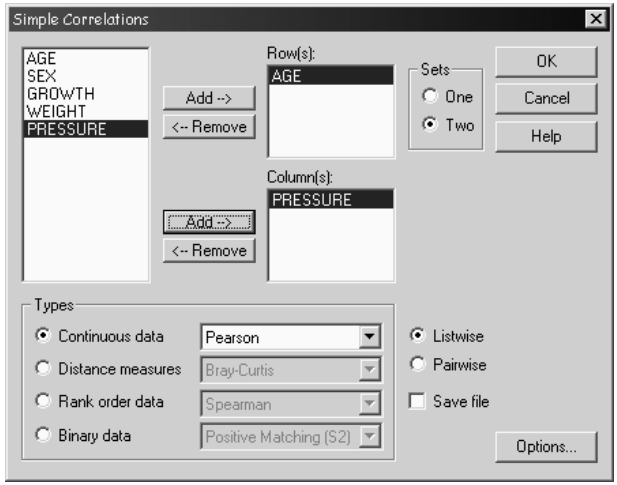


Рис. 1.6. Выбор переменных для расчета коэффициента корреляции в программе SYSTAT

Удобнее ли такой способ, чем типичный для электронных таблиц метод протаскивания курсора — решать пользователю. Отметим только, что способ протаскивания, хотя и малоэффективный в некоторых случаях (например, при большом объеме данных интервалы проще ввести с клавиатуры), отличается гораздо большей гибкостью. А недостатки типичного для программ статистического анализа данных подхода очевидны. Это, прежде всего, выбор только столбца как области исходных данных. Во вторых, столбец берется для расчета целиком — нельзя выделить даже его часть. Примитивность и ограниченность решений выбора переменных для анализа в большинстве стандартных программ статистического анализа, по сравнению с удобством и гибкостью Microsoft Excel и других электронных таблиц, у нас сомнений не вызывает.

В поле ввода `TextBox`, вопреки его названию, вводятся не только данные текстового типа, но и числовые данные различных типов. Это эффективно и удобно. Любые данные вводятся как символьные, а преобразование этих данных в нужный тип производится функцией их обработки. Одновременно можно выполнить и верификацию введенных в данное поле данных.

Свойства каждого элемента управления, помещенного в форму, можно изменять, по усмотрению программиста, с помощью панели свойств, активизируемой при выборе того или иного элемента управления, находящейся, по умолчанию, в левом нижнем углу. В число изменяемых свойств входят наименование, координаты относительно левого верхнего угла формы, абсолютные размеры, цвет, шрифт, индекс пункта контекстно-зависимой помощи и т. д. Для изучения свойств элементов управления проще всего варьировать значения параметров, оценивая результат.

Существуют и другие способы передачи массива данных в модуль VBA, однако они, по нашему мнению, не так удобны и делают решения пользовательской программы менее похожими на стандартные решения Microsoft Excel.

Рекомендации разработчика Microsoft Excel

Очень важно при разработке дополнительных модулей Microsoft Excel создать хороший код, тщательно протестировать его и обеспечить поддержку эффективных методов кодирования. VBA не ограничен применением стандартных разработок, но использование накопленного разработчиками опыта может сделать более легкой разработку и поддержку программ.

Данные рекомендации не имеют целью ограничить творчество программиста. Microsoft Excel — хорошая программа, в смысле удобного обращения и качества реализации представленных в ней методов расчета. Она оставляет программисту место и для творчества, благодаря документированному API

(Application Programming Interface — интерфейс прикладного программирования) и явно недостаточному количеству стандартных методов анализа.

Очень важно, чтобы проект легко интегрировался в Microsoft Excel. Основная рекомендация, при проектировании формы, заключается в требовании как можно более походить на стандартные формы Microsoft Excel, в функциональности и стиле оформления (размер, цвет, шрифт, взаимное расположение элементов управления, содержание надписей и т. п.). Меню дополнительного модуля должны подражать встроенным меню Microsoft Excel. Панели инструментов должны выглядеть так, как будто их разработала корпорация Microsoft. Также должен быть обеспечен доступ к функциям программы с помощью горячих клавиш (называемых еще акселераторами) в диалоговых панелях.

Интерфейс проекта должен выглядеть так, как будто это — встроенная часть Microsoft Excel. Для обеспечения данной рекомендации, предлагается найти готовую диалоговую панель, чтобы использовать ее как модель. Например, кнопки **OK** и **Cancel** в программах Microsoft Office принято располагать в правом нижнем углу формы. Следует убедиться, что управляющие элементы проекта как можно больше походят на встроенные управляющие элементы.

Эффективность программного кода оценивается только через пользовательский интерфейс. Пользователь должен воспринимать внешний вид формы (как и программы в целом) естественно. Все вопросы (если они действительно необходимы) должны быть краткими, но информативными, а ответы на них, в отличие от законодательных актов известного государства, трактоваться однозначно и не нуждаться в пространных комментариях. К тому же, четкое решение составляет хорошую основу для дальнейшего развития проекта.

Обработка событий формы средствами VBA

Теперь рассмотрим код VBA, ответственный за обработку событий, возникающих во время работы пользователя с формой. После двойного щелчка левой кнопкой мыши на имени формы в окне редактора появится окно **Code** с текстом программы VBA, предназначенной для обработки данной формы. Иначе доступ к коду можно получить после нажатия правой кнопки мыши на имени формы с последующим выбором из контекстного меню **View Code**.

Код VBA, прежде всего, содержит функцию `UserForm_Initialize`, исполняемую при вызове формы из меню или из другой формы. В данной функции обычно присваиваются начальные значения свойствам `Value` всех элементов управления формы, в зависимости от их типа. Листинг 1.6 содержит код инициализации следующих объектов:

- 11 кнопок `LinkButton` (их имена введены в соответствующем окне свойств в левом нижнем углу окна редактора) типа `OptionButton`, предназначенных для выбора меры различия для кластерного анализа (*глава 7*);
- 3 кнопок `MethodButton` также типа `OptionButton`, ответственных за выбор метода решения;
- 2 кнопок `ObjectButton` типа `OptionButton`, управляющих порядком ввода исходных данных из таблицы.

Листинг 1.6

```
Private Sub UserForm_Initialize()
 LinkButton1.Value = True
 LinkButton2.Value = False
 LinkButton3.Value = False
 LinkButton4.Value = False
 LinkButton5.Value = False
 LinkButton6.Value = False
 LinkButton7.Value = False
 LinkButton8.Value = False
 LinkButton9.Value = False
 LinkButton10.Value = False
 LinkButton11.Value = False
 MethodButton1.Value = True
 MethodButton2.Value = False
 MethodButton3.Value = False
 MethodButton4.Value = False
 ObjectButton1.Value = True
 ObjectButton2.Value = False
End Sub
```

Структуры функций обработки нажатий перечисленных кнопок одинаковы. Пример такой функции показан на листинге 1.7.

Листинг 1.7

```
Private Sub MethodButton2_Click()
 MethodButton1.Value = False
 MethodButton2.Value = True
 MethodButton3.Value = False
 MethodButton4.Value = False
End Sub
```

Если для логического объединения нескольких кнопок зависимого выбора применяется элемент управления Frame, то, как было указано выше, в применении функций обработки нажатия кнопок, подобной функции, показанной на листинге 1.7, необходимость отпадает. Данные события будут обработаны автоматически.

Более подробного рассмотрения заслуживает функция обработки нажатия командной кнопки, ответственной за запуск программы расчета. Задачами данной функции являются:

- формирование набора параметров расчета, в соответствии с выбором пользователя,
- верификация введенных или выбранных пользователем параметров расчета (не всегда),
- вызов функции, выполняющей верификацию исходных данных (если это не было сделано заранее), производящей расчет и осуществляющей вывод результатов.

В итоге получается то, что во времена перфокарт на электронно-вычислительных машинах (ЭВМ) называлось пакетной обработкой данных, а колода перфокарт для расчета конструкции методом конечных элементов была высотой от плоскости стола до потолка.

Код одной из возможных реализаций функции показан на листинге 1.8.

Листинг 1.8

```
Private Sub CommandButton1_Click()
 Dim Link As Integer
 Dim Met As Integer
 Dim Obj As Integer
 If LinkButton1.Value = True Then
   Link = 0
 ElseIf LinkButton2.Value = True Then
   Link = 1
 ElseIf LinkButton3.Value = True Then
   Link = 2
 ElseIf LinkButton4.Value = True Then
   Link = 3
 ElseIf LinkButton5.Value = True Then
   Link = 4
 ElseIf LinkButton6.Value = True Then
   Link = 5
```