Игорь Сафронов

bhv

Visual Basic
в задачах и примерах
2-е издание

+задачи ЕГЭ

История Бейсиков и в чем их прелесть

Первые проекты и первые радости

Алгоритмы, ветвления и циклы, процедуры и файлы, мультимедийные проекты и анимация

Как самому написать игру и всякие штучки-приколы

333 самостоятельных упражнения с примерами и решениями, изложенные на доступном для понимания русском языке!

Раздел задач, которые пригодятся при сдаче ЕГЭ

\+

Материалы на www.bhv.ru

Игорь Сафронов

Visual Basic
в задачах и примерах

2-е издание

Санкт-Петербург
«БХВ-Петербург»
2014

УДК 004.438 Visual Basic
ББК 32.973.26-018.1
С21

Сафронов И. К.

С21 Visual Basic в задачах и примерах. — 2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2014. — 400 с.: ил.

ISBN 978-5-9775-0622-9

Основу содержания книги составляют разработанные автором задачи и примеры, ярко демонстрирующие возможности языка. В занимательной и доступной форме описывается история языков семейства Basic, реализация различных видов алгоритмов средствами Visual Basic, работа с подпрограммами и файлами, мультимедийные возможности языка, написание простых игр. Большое внимание уделено интерфейсу VB. Каждая из рассматриваемых тем предваряется коротким теоретическим вступлением, поясняющим приведенные примеры и задачи. Приведены справочник по языку и решения избранных задач. Во втором издании добавлен раздел задач, направленных на подготовку к ЕГЭ, и практически треть задач заменена на новые.

Может использоваться в качестве задачника учащимися 8—11 классов, студентами первых курсов и преподавателями школ и вузов.

На сайте издательства размещены листинги программ, приводимых в книге в качестве примеров.

Для начинающих программистов

УДК 004.438 Visual Basic
ББК 32.973.26-018.1

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Людмила Еремеевская*
Зав. редакцией	*Екатерина Капалыгина*
Редактор	*Юрий Рожко*
Компьютерная верстка	*Ольги Сергиенко*
Корректор	*Зинаида Дмитриева*
Дизайн серии	*Игоря Цырульникова*
Оформление обложки	*Марины Дамбиевой*

Подписано в печать 28.06.13.
Формат 60×90$^{1}/_{16}$. Печать офсетная. Усл. печ. л. 25.
Тираж 2000 экз. Заказ №
"БХВ-Петербург", 191036, Санкт-Петербург, Гончарная ул., 20.

Первая Академическая типография "Наука"
199034, Санкт-Петербург, 9 линия, 12/28

ISBN 978-5-9775-0622-9

Оглавление

ПРЕДИСЛОВИЕ

"Лучше гор могут быть только горы…"

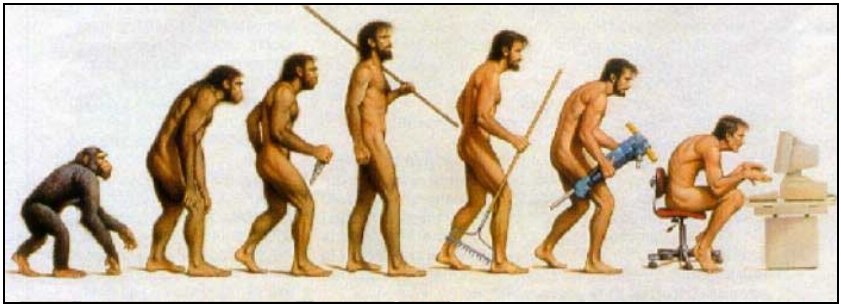
> Работа программиста и шамана имеет много общего — оба бормочут непонятные слова, совершают непонятные действия и не могут объяснить, как оно работает.
>
> *Анекдот на тему*

Для пробуждения интереса к чтению и работе с этой книгой хочу ознакомить вас с некоторыми интересными подробностями языка Visual Basic (VB) и соображениями о его полезности. Хотелось бы, чтобы эта книга ассоциировалась с преодолением себя на пути к вершинам программирования, была насыщена трудностями и радостным осознанием "Я смог!".

В качестве эпиграфа одна интересная цитата из журнала "Наука и жизнь" (№ 11, 2003 г.):

> "Способности к работе с компьютером обнаружили ученые у высших приматов, в частности, у бабуинов. В ходе исследований, проводившихся в Стэмфордской зоологической школе (США), выяснилось, что, работая с компьютером, обезьяны могут тестировать программное обеспечение и даже заниматься программированием. Правда, у них возникают трудности, когда по ходу работы приходится обращаться к меню, содержащему более двух уровней.
>
> Однако ученым удалось разрешить эту проблему. Оказалось, что если "многоходовые" действия ведут к некоей заветной картинке, самец бабуина способен осознать и запомнить до семи уровней в меню. Попутно во время работы выяснился забавный факт: как только бабуин научается нажимать нужные клавиши или пользоваться сложными меню, его социальный статус среди сородичей резко возрастает.

После простейшего курса по работе с Windows большинство "студентов" освоило язык программирования **Visual Basic 3.0**. В результате обезьяны смогли самостоятельно менять программные настройки и даже редактировать параметры атрибутов файлов. Однако освоить язык программирования Java не смог ни один из приматов".



Вперед, к Visual Basic!

Введение

Зачем я привел цитату из журнала в предисловии и, вообще, зачем я взялся за написание этой книги?

В свое время я написал книгу "Бейсик в задачах и примерах". Мне говорили, что Бейсик уже умер, он никому не нужен, но я как практикующий ☺ преподаватель информатики знал истинное положение вещей при обучении детей в школе и настоял на своем. В результате книга состоялась и несколько раз допечатывалась и выдержала два издания. Бейсик, на мой взгляд, идеальное средство для ознакомления с алгоритмизацией, и я вижу, что никакие программные пакеты или готовые игры не заменят продукта, созданного своими руками. В глазах юного программиста читается гордость за содеянное, оно демонстрируется друзьям и родственникам, а тем остается только завидовать, потому что программирование — это творчество.

А после выхода моего первого издания по Visual Basic прошло уже несколько лет. За это время в образовании прочно обосновалось понятие ЕГЭ, в заданиях которого вовсю используется Visual Basic. И это тоже подтолкнуло меня к переизданию данного задачника.

Ну, а кроме того, мне всегда хотелось написать книгу для начинающих программистов, книгу с человеческим лицом, да так, чтобы не оттолкнуть их первыми же сильно умными и непонятными словами. Поэтому принцип этого задачника будет следующий: я привожу подробно разобранный пример и прошу вас затем решить ряд однотипных с этим примером задач. В конце книги будет приведен краткий справочник по языку, а дальнейшее, конечно, будет зависеть только от вас. Для облегчения вашего труда листинги заданий из книги можно скачать по ссылке **ftp://ftp.bhv.ru/9785977506229.zip**. Ну, а если возникнут трудности, меня всегда можно будет найти по адресу **old_matros@mail.ru** (после old сто-

ит знак подчеркивания, а не пробел...). Ну а тем, кто уже знаком с любой из обычных версий языка Basic, будет намного проще — все повторяется....

Кроме того, отдельное спасибо хочу сказать всем, кто помог мне в создании этой книги: любимой жене — за то, что не мешала, сыну и дочери — за то, что они есть, и, конечно, издательству "БХВ-Петербург" — за их долготерпение и профессиональную работу.

Немного истории

Язык *Basic* был разработан преподавателями Дартмутского колледжа Джоном Кемени и Томасом Куртцом в 1964 году как средство обучения и работы начинающих программистов. (Дартмутский колледж в штате Нью-Гампшир, США, был создан в середине XVIII века, это одно из старейший высших заведений Америки.) Предназначение этого языка определено в самом его названии, которое является аббревиатурой слов Beginner's All-purpose Symbolic Instruction Code ("многоцелевой язык символических инструкций для начинающих") и при этом в дословном переводе означает "базовый". Тут создатели языка видимо провели историческую параллель с миссионерами-англичанами, которые несли христианские ценности во все новые и новые британские колонии, а средством общения сделали Basic English ("базовый английский"), включивший в себя 300 самых распространенных и простых для усвоения туземцами слов английского языка.

Замечание

Раньше языки программирования писались обязательно прописными буквами — BASIC, FORTRAN, COBOL. В 1990 году Международная организация стандартов приняла решения, что они пишутся как обычные имена собственные (прописной является только первая буква).

Однако парадокс заключается в том, что, будучи действительно весьма простым средством программирования, совершенно непригодным в те времена для решения серьезных задач, Basic представлял собой качественно новую технологию создания программ в режиме интерактивного диалога между разработчиком и компьютером. То есть представлял собой прообраз современных систем программирования. Другое дело, что решение подобной задачи на технике тех лет было возможно только за счет максимального упрощения языка программирования и использования транслятора типа "интерпретатор".

Замечание

Интерпретатор — это такой синхронный программный "переводчик" алгоритмического языка на язык машинный. Его достоинство — получение результата выполнения программы сразу же (до первой ошибки ☺). Недостаток — увы, медлительность.

Резкое развитие систем на основе языка Basic началось с появлением в начале 80-х годов XX века персональных компьютеров, производительность и популярность которых растет вот уже двадцать лет невиданными темпами.

QuickBasic против TurboBasic

В конце 80-х годов насчитывалось около десятка систем Basic различных фирм-разработчиков. Однако главная борьба шла между QuickBasic (компании Microsoft) и TurboBasic (Borland). Вообще-то конкуренция между этими двумя разработчиками средств программирования шла по целому спектру языков: Basic, Pascal и C. И результатом ее в 1989 году стало неявное мировое соглашение, когда компания Microsoft отказалась от дальнейшей поддержки Pascal, а Borland — Basic.

Тогда многие комментаторы язвительно замечали, что Microsoft отказалась от Pascal в пользу Basic исключительно из-за личных пристрастий основателя и руководителя корпорации Билла Гейтса. Действительно, разработка в 1975 году интерпретатора Basic для микро-ЭВМ Altair 8800 была первым проектом двадцатилетних Билла Гейтса и Пола Аллена, только что основавших фирму Microsoft (в тот момент они были единственными сотрудниками новой компании). С тех пор президент Microsoft постоянно участвовал в стратегии разработок Basic-систем корпорации и до сих пор, перечисляя свои титулы, Билл Гейтс довольно часто добавляет "Basic-программист". Но "отцом" Visual Basic считается Алан Купер — независимый программист, который в конце 80-х годов разработал, а затем и продал фирме Microsoft прототип механизма визуального проектирования форм для Basic.

Однако победа QuickBasic определялась чисто технологическими причинами. В этой системе была удачно реализована схема смешанного использования традиционных Basic-технологий и классических методов создания сложных программных систем. Отметим, что с 1990 года усеченный вариант QuickBasic под названием QBasic был включен в состав MS-DOS.

Замечание

Многие современные пользователи ошибочно думают, что QuickBasic и QBasic — это одно и то же.

Эпоха Visual Basic

В начале 90-х годов Microsoft начала активную борьбу за продвижение в массы своей новой операционной системы Windows (против своей же, но более уже устаревающей MS-DOS). Но, как известно, пользователи работают не с операционной системой (ОС), а с программами, которые работают в ее среде. Поэтому скорость смены платформы в основном определяется темпами появления соответствующих прикладных программ.

Однако смена ОС представляет серьезную проблему и для программистов, т. к. им нужно было осваивать новую технологию разработки программ. В тот момент бытующим (и в значительной степени, совершенно справедливым) мнением было то, что Windows предъявляет более высокие требования к квалификации программиста.

В 1991 году под лозунгом "теперь и начинающие программисты могут легко создавать приложения для Windows" появилась первая версия нового инструментального средства Microsoft Visual Basic. В тот момент Microsoft достаточно скромно оценивала возможности этой системы, ориентируя ее, прежде всего, на категорию начинающих и непрофессиональных программистов. Основной задачей тогда было выпустить на рынок простой и удобный инструмент разработки в то время еще довольно новой среде Windows, программирование в которой представляло проблему и для опытных специалистов.

Действительно, Visual Basic 1.0 в тот момент был больше похож не на рабочий инструмент, а на действующий макет будущей среды разработки. Его принципиальное новшество заключалось в реализации идей событийно-управляемого и визуального программирования в среде Windows, которые весьма радикально отличались от классических схем разработки программ. По общему признанию Visual Basic стал родоначальником нового поколения инструментов, называемых сегодня *средствами быстрой разработки программ* (Rapid Application Development, RAD). Сегодня эта идеология считается привычной, но тогда она казалась совершенно необычной и создавала серьезные проблемы (в том числе чисто психологического плана) для программистов "старых времен".

Тем не менее число Visual Basic-пользователей росло, причем во многом за счет огромной популярности ее предшественника — QuickBasic. При этом Visual Basic быстро "мужал", усиливаясь за счет как развития среды программирования, так и включения профессиональных элементов языка и проблемно-ориентированных средств. И к моменту выпуска в 1995 году Visual Basic 4.0 эта система была уже признанным и одним из самых распространенных инструментов создания широкого класса приложений.

В настоящее время используется версия Visual Basic 6.0, появление версии 7.0 ожидается в ближайшем будущем.

Visual Basic for Applications

В начале 90-х годов наметилась отчетливая тенденция включения в приложения, предназначенные для конечного пользователя, средств внутреннего программирования, которые должны были решать задачи настройки и адаптации этих пакетов для конкретных условий их применения.

В конце 1993 года компания Microsoft объявила о намерении создать на основе Visual Basic новую универсальную систему программирования для прикладных программ, которая получила название *Visual Basic for Applications* (VBA, Visual Basic для приложений). Естественно, реализацию этого проекта она начала с собственных офисных пакетов.

Первый вариант VBA 1.0 появился в составе MS Office 4.0, но лишь в программах Excel 4.0 и Project 6.0. В других же приложениях — Word 6.0 и Access 2.0 — были собственные варианты Basic. Более того, VBA 1.0 довольно сильно отличался (причем имея ряд существенных преимуществ) от используемой тогда универсальной системы Visual Basic 3.0.

Качественный перелом наступил в конце 1996 года с выпуском MS Office 97, в котором была реализована единая среда программирования VBA 5.0, включенная в программы Word, Excel и PowerPoint. Более того, VBA 5.0 использовала тот же самый языковый механизм и среду разработки, что и универсальная система Visual Basic 5.0. В состав MS Office 2000 вошла соответственно версия VBA 6.0, которая используется в шести программах — Word, Excel, PowerPoint, Access, Outlook, Frontpage.

В результате последние три года Microsoft позиционирует свой пакет MS Office не просто как набор прикладных программ, а как комплекс-

ную платформу для создания бизнес-приложений, решающих широкий круг специализированных задач пользователей. Именно этим объясняется появление в его составе специального выпуска для разработчиков приложений — *Developer Edition*.

Одновременно VBA активно продвигается в качестве отраслевого стандарта для управления программируемыми приложениями, объявив о возможности его лицензирования. Сегодня уже более ста ведущих мировых фирм-разработчиков прикладных программ (среди них есть и российские) приобрели на него лицензии и включают его в состав своих программных продуктов.

Однако нужно подчеркнуть, что, несмотря на доминирование VB и VBA на рынке программных средств, этот инструмент не является единственным примером использования языка Basic. В частности, в конце 90-х годов значительное распространение получила разработка компании Sax Software — механизм *Sax Basic Engine*, своеобразный "облегченный" вариант Visual Basic 6.0, успешно используемый многими разработчиками для автоматизации своих приложений.

В настоящее время язык VBA активно используется в работе со следующими приложениями: Microsoft Word 2010, Microsoft Outlook 2010, Microsoft Access 2010, Microsoft Excel 2010, Microsoft PowerPoint 2010, Microsoft Publisher 2010

Не начать ли с "Васика"?

Из сказанного ранее можно сделать следующий вывод. Освоение механизма программирования на VBA, реализованного в офисном приложении, которое установлено на вашем компьютере, откроет вам возможность использования полученных знаний и навыков при работе с десятками и сотнями других программ, в том числе и тех, которых пока еще нет на свете. Начав с составления простейших макрокоманд, при желании можно в рамках одного инструментария стать профессионалом, разрабатывающим программные системы любой сложности. Не говоря уже о том, что после освоения технологии разработки приложений смена инструментария не будет составлять серьезных проблем.

Десять лет назад во всем мире было не более двух миллионов программистов. Сегодня их насчитывается около пятнадцати миллионов, из них не менее 70% используют в качестве хотя бы одного из инструментов VB или VBA.

ГЛАВА 1

Начинаем восхождение. Первая высота

Взойдя на первую вершину, вы узнаете о том, где взять Visual Basic и как начать работу с первым проектом, узнаете о форме и инструментах, научитесь линейно программировать и красиво оформлять ваши проекты.

1.1. Где взять и как запустить

Ну, где в нашей стране берут программы (?) — все по-разному. Одни покупают, другие берут у друзей, а некоторые так ☺.

Я лично купил в магазине на Невском проспекте. Установил на компьютер, а ярлык поместил на рабочий стол. Он выглядит вот так (рис. 1.1).

Рис. 1.1. Ярлык для Visual Basic 6.0

Теперь щелкаем по ярлыку (кто-то два раза, а кто-то и один — как настроено!), и загружается редактор языка. Я надеюсь, дорогой читатель, что у вас уже есть какие-то навыки программирования, хотя бы в обычном QBasic, и поэтому я буду меньше останавливаться на алгоритмических конструкциях (они, собственно, остались теми же самыми), а больше буду уделять времени реализации алгоритмов в редакторе

Visual Basic. Как говорил один мой хороший знакомый программист: "Язык лишь средство. Самое трудное разработать алгоритм".

Итак, окно редактора после запуска выглядит следующим образом (рис. 1.2).

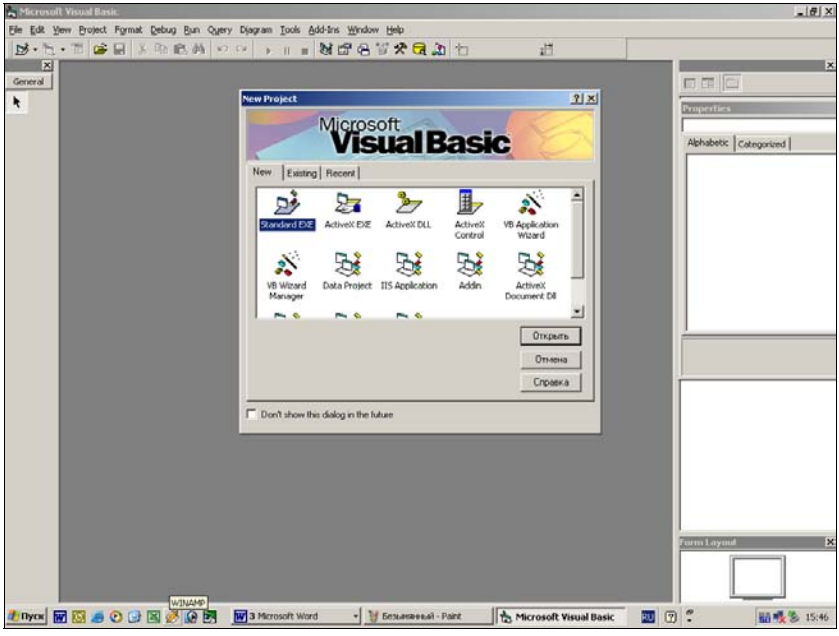


Рис. 1.2. Окно редактора

Окно как окно — все вроде бы пока стандартно для приложений Windows: заголовок, меню, панели инструментов... Все подписано по-английски, но это полезно, всякий программист должен знать несколько слов на этом языке (типа, "wait" или "game over" ☺).

Хотя, если очень хочется, то можно ☺ и русифицировать. Русификатор легко находится в Интернете по запросу "Visual Basic русификатор". Если же совсем никак — обращайтесь ко мне.

Для своего первого проекта из предложенных вариантов выберем "Standard EXE" (рис. 1.3).

Рис. 1.3. На подступах к новому проекту

1.2. Почему проект?

Раньше мы писали программы, а то, что мы будем делать здесь, будет называться *проектом*. Почему так? Когда мы переводим наш алгоритм на языки, например, QBasic, Pascal, C++ и др., то в результате получаем текст программы, которую, в свою очередь, можем запустить и полюбоваться на ее работу. В Visual Basic при воплощении вашего алгоритма в жизнь необходимо создать несколько взаимосвязанных частей — одну или несколько экранных форм, один или несколько программных кодов и один или несколько программных модулей. Поэтому эта большая работа и будет носить гордое название "Проект" и сохраняться в файле с расширением vbp (Visual Basic Project). В папке проекта также будут сохранены и все формы и модули.

1.3. Первый проект "«Зенит» — чемпион!"

Примечание

Чтобы не обижать болельщиков других команд, сразу же скажу, что вы можете делать проект "«Локомотив» — чемпион!", "«ЦСКА» — чемпион!" или даже "«Спартак» — чемпион!" ☺

Замечание

Очень рекомендую быстро, но внимательно выполнить этот проект, тогда вам станут понятны многие элементарные, но очень важные моменты создания проектов в VB.

В этом разделе мы рассмотрим следующие темы:

- создание, сохранение, запуск и остановка проекта. Создание из проекта EXE-приложения;
- инструменты **Label**, **Image**, **CommandButton**;
- свойства `Caption`, `Left`, `Top`, `Width`, `Height`, `Font`, `BackColor`, `ForeColor`, `Alignment`, `Stretch`;
- что делать, если пропала панель инструментов **Standard**;
- что делать, если пропало окно свойств **Properties**.

После запуска варианта проекта "Standard EXE" мы получаем такую интересную картинку (рис. 1.4).

В общем, те, кто хоть что-нибудь уже делал в приложениях Windows, ничего принципиально нового не увидят... Заголовок окна **Project1-Microsoft Visual Basic [design]**, ниже — строка меню (**File**, **Edit**, **View** и т. д.), еще ниже — панель инструментов **Standard** (если вы вдруг ее случайно куда-то перетащили, то восстановить ее можно из меню **View | Toolbars | Standard**), слева — набор базовых компонентов **General**, справа — окно свойств объекта **Properties-Form1** (в нашем случае они относятся к форме **Form1**), ну, а по центру — непосредственно сама форма, которую мы и будем проектировать.

Замечание

Если случайно закрыли окно свойств, то выполните последовательность команд **View | Properties Windows** или нажмите клавишу <F4>.

Кроме того, в окне свойств есть две вкладки: первая — **Alphabetic**, где свойства расположены по алфавиту, вторая — **Categorized**, где свойства сгруппированы по категориям.

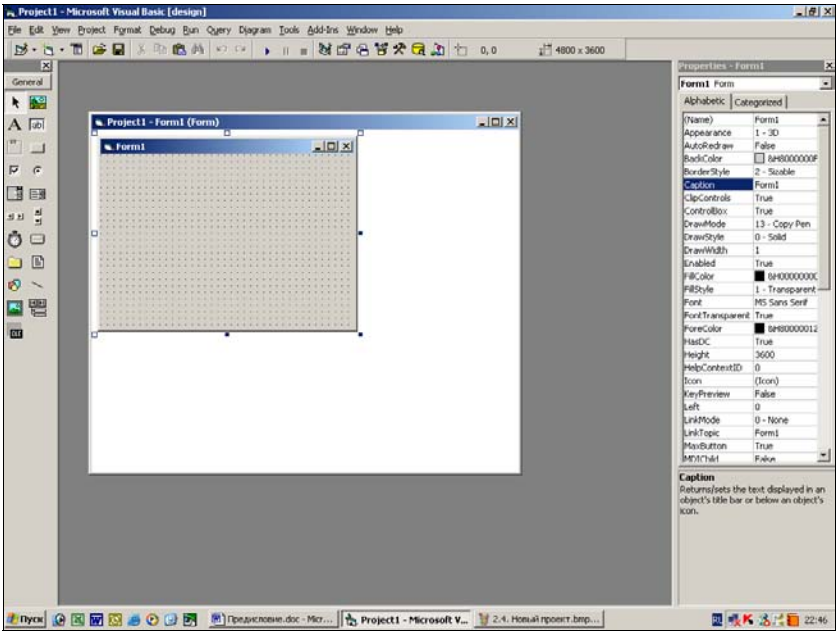


Рис. 1.4. Новый проект

Итак, первый проект до безобразия элементарен. Он будет запускать никак не оформленную форму, на которой будет находиться надпись "«Зенит» — чемпион!" (или любая другая, какая вам больше понравится). Закрываться наша форма после запуска будет как обычное окно Windows. Итак, за работу!

1.3.1. Инструмент *Label*

Возьмем на панели **General** инструмент **Label** (выглядит он так A), который позволяет выводить на форме любой текст, и растянем на форме прямоугольник. Должно получиться так, как на рис. 1.5.

Теперь в окне свойств **Properties-Label1** (следите только, чтобы нужный объект на форме был выделен) найдем свойство `Caption` и изменим его, написав вместо "Label1" — "«Зенит» — чемпион!".

Теперь форма должна выглядеть так, как на рис. 1.6.

Все! Первый проект готов. Аплодисменты в студию! Но, чтобы не пропал наш скорбный труд, надо обязательно перед запуском проекта его сохранить.

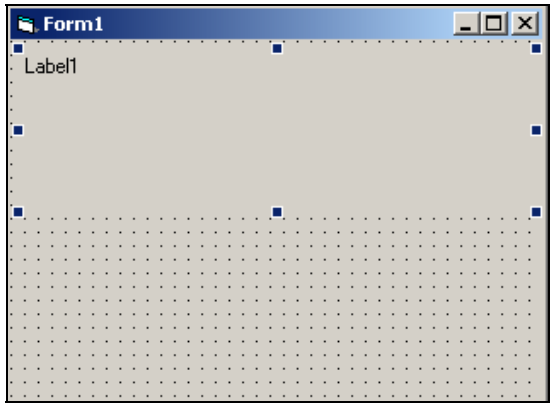


Рис. 1.5. Метка на форме

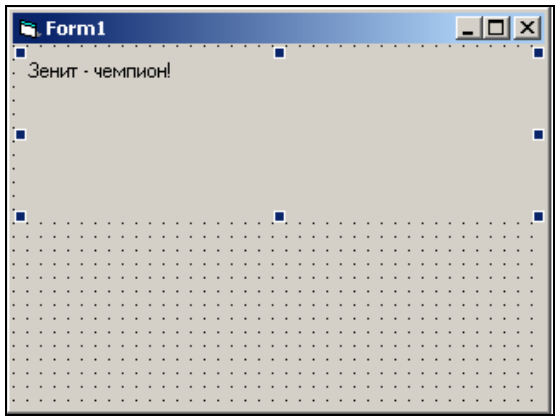


Рис. 1.6. Форма с надписью "Зенит — чемпион!"

1.3.2. Сохранение проекта

В меню **File** выбираем команду **Save project** или нажимаем на стандартной панели инструментов кнопку — **Save**. В появившемся окне выбираем диск и папку, создаем в ней папку для нашего первого проекта (так давайте и назовем ее — "Первый проект") и сохраним туда нашу форму (Form1.frm) и проект (Project1.vbp).

Теперь проект можно запускать.

1.3.3. Запуск и остановка проекта

Запускается проект клавишей <F5> или командой из меню **Run | Start** или нажатием на стандартной панели инструментов кнопки ▶ (правда, очень похоже, как на плеере ☺!).

И вот наш первый проект в работе. Ничего, что он пока такой серенький и невзрачный, как гадкий утенок. Пройдет совсем немного времени, и он будет красив и достоин тех слов, которые на нем красуются!

Остановим выполнение проекта. Лучше это делать через меню **Run | End** или нажатием на стандартной панели инструментов кнопки ■.

1.3.4. Создание EXE-приложения

Прежде чем мы займемся красотой нашего проекта, предлагаю ознакомиться с возможностью создания из него полноценного EXE-приложения, которое сможет запускаться безо всякого участия оболочки Visual Basic.

Выберем в меню команду **File | Make Project1.exe...** В появившемся окне сохранения выберем нужную папку (в нашем случае пусть будет все та же папка Первый проект) и дадим имя файлу (пусть будет zenit), после чего нажмем кнопку **OK**.

1.3.5. Запуск EXE-приложения

Теперь давайте закроем Visual Basic, найдем и откроем папку Первый проект и двойным щелчком запустим файл zenit.exe. Вы видите, что наша форма загружается и работает безо всякого внешнего воздействия ☺.

Теперь закроем форму и снова загрузим Visual Basic. Откроем уже существующий проект. Для этого выполним команду из меню **File | Open Project...** или нажмем на стандартной панели инструментов кнопку 📂 — **Open**, выберем там нужную папку и файл Project1.vbp.

Замечание

Если вдруг стало отсутствовать окно обзора Проекта, то выберите в меню **View | Project Explorer** или нажмите комбинацию клавиш <Ctrl>+<R>.

Займемся свойствами формы. Выделите ее. Почему у нее в заголовке написано безликое Form1? Непорядок! Изменим свойство `Caption` с Form1 на "Футбол". Да и имя объекта Form1 (свойство `Name`) можно

изменить на Football (данная версия VB поддерживает и русские имена объектов, хотя я все же предпочитаю по старинке давать имена на английском). Уже лучше.

Теперь займемся размерами объектов.

1.3.6. Установка положения и размеров объекта

Размеры объекта задаются свойствами `Width` (Ширина) и `Height` (Высота). По умолчанию эти величины задаются в специальных единицах — твипах. *Один твип* — это 1/1440 логического дюйма. *Логический дюйм* — это такое расстояние на экранной форме, которое при печати на принтере будет равным 1 дюйму (1 дюйм = 2,54 см). Вы можете видеть на экранной форме сетку — ряды точек. Сетка помогает точно размещать на форме объекты. По умолчанию расстояние между соседними точками сетки составляет 120 твипов.

Положение формы и объектов на ней тоже определяется в твипах. На стандартной панели инструментов справа вы можете видеть две пары чисел, которые называются *индикатором положения и размеров* выделенного объекта (рис. 1.7).

Рис. 1.7. Индикатор положения и размеров

Первая пара чисел указывает в данном случае положение формы относительно левого верхнего угла экрана (по горизонтали форма после запуска будет отстоять на 0 твипов, по вертикали от того же угла тоже на 0 твипов).

Вторая пара чисел задает размеры формы в твипах по горизонтали (в нашем случае 4800) и вертикали (3600).

Таким образом, положение формы на экране после запуска можно регулировать свойствами `Left` и `Top`, а размеры, соответственно, `Width` и `Height`.

Для приблизительного, на глазок, изменения положения формы можно воспользоваться окном, вызываемым через меню **View | Form Layout Window**, которое появится обычно ниже **Properties**. На нем вы будете видеть экран и схематичное положение формы, которое мышью сможете подвинуть туда, куда хотите. Числа на индикаторе положения тут же изменятся.

Размеры формы и объектов на ней также легко меняются мышью, как, впрочем, и размеры любого окна Windows.

Теперь об объектах, расположенных на форме. Когда они выделены, то индикатор положения показывает первой парой чисел положение объекта относительно верхнего левого угла формы, второй парой — размеры объекта.

Точно выставить эти позиции можно через те же свойства, что и для формы.

Давайте вернемся к нашему проекту и установим следующие свойства для формы и объекта Label (табл. 1.1).

Таблица 1.1. *Свойства объектов Form и Label*

Свойство, объект	`Left`	`Top`	`Width`	`Height`
Экранная форма	3000	3000	6000	4000
Label	240	120	5415	1215

Замечание

Если вдруг захотелось измерять размеры объектов не в твипах, а в чем-либо еще, то находите для формы свойство `ScaleMode` и выбирайте из вариантов, представленных на рис. 1.8 (числа на индикаторе положения и размеров будут представлены в этой же размерности).

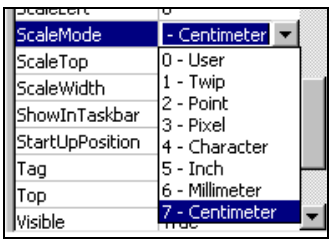


Рис. 1.8. Изменение свойства `ScaleMode`

1.3.7. Цвета и шрифты объектов

Как-то все в серых тонах, для чемпиона не пафосно! Используем свойство `BackColor`. Воспользуйтесь им, выбрав закладку **System** (Системные цвета) или **Palette** (Выбор из палитры). Выберите подходящие на ваш взгляд цвета для формы и объекта Label.

Цвета объектов можно подбирать и самим — на закладке **Palette** щелкнуть правой кнопкой мыши на любом белом квадратике — выбирайте!

Шрифт для нашего гордого текста в объекте Label — это, соответственно, свойства `Font` (шрифт, начертание, размер) и `ForeColor` (цвет шрифта). Установите для нашего проекта: цвет шрифта — белый, шрифт — MS Sans Serif, начертание — жирный, размер — 24.

Осталось выравнивание текста внутри нашего объекта. Это будет свойство `Alignment` — выберите значение Center.

Запустим проект и увидим, как хорошо он расположился на экране и как он прекрасно выглядит ☺ (рис. 1.9)!

Сохраним его под тем же именем.

Замечание

К сожалению, не все шрифты VB поддерживают кириллицу, поэтому опытным путем проверьте, какие шрифты можно использовать для написания на нашем великом и могучем ☺.

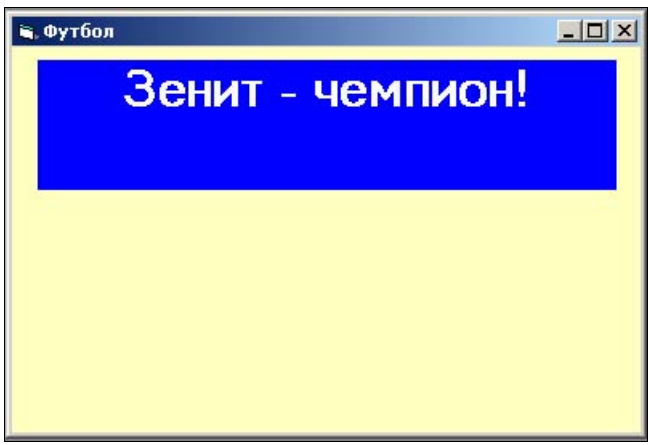


Рис. 1.9. Слегка оформленный объект

1.3.8. Картинки на форме

Продолжаем наводить красоту. Хочется приукрасить все это дело картинкой. Сразу вопрос: "Где взять?" Можно из библиотеки стандартных картинок или с дисков с ClipArt'ами, можно из Интернета, а можно и самим нарисовать. Важно, чтобы рисунок был у вас на диске в виде файла.

Я бы сделал так: взял бы инструмент **Image**  с панели **General** и растянул бы в правом нижнем углу формы прямоугольник для вставки подходящей картинки. Установил бы свойства: `Left` — 240; `Top` — 1440; `Width` — 1455; `Height` — 2055.

И новое для нас свойство — `Stretch`. Установите для него значение True. В этом случае картинка растянется по отведенному ей месту.

Замечание

Правда, в таком случае могут быть нарушены ее пропорции!

После установки свойств можно войти в свойство `Picture` и найти там файл с вашей картинкой. У меня получилось вот так, как на рис. 1.10.

Рис. 1.10. Проект с картинкой

1.3.9. Командные кнопки на форме и в проекте

Теперь пора сделать проект хоть немножко управляемым. Таким образом, от оформления формы мы переходим непосредственно к ее про-

граммированию. И вот тут-то и пригодится знание старого доброго Quick или любого другого Basic'a.

Сейчас наша форма закрывается как обычное окно Windows, но мы будем закрывать ее с помощью специальной командной кнопки.

Итак, разместим на форме CommandButton командную кнопку со свойствами положения и размеров: `Left` — 2040; `Top` — 2880; `Width` — 1815; `Height` — 615.

Выберем теперь какой-нибудь подходящий цвет, выбранный с помощью свойства `BackColor`. Ой, цвет кнопки не изменился! Паника в проекте! Это все потому, что не установлено свойство `Style` — Graphical. Это касается только командных кнопок. Теперь все нормально, надеюсь?

Примечание

Я долго искал и в справке, и во всевозможной литературе, и в Интернете, и у своих коллег информацию об изменении цвета шрифта на командной кнопке. Не нашел ответа. Может быть, кто-то из вас найдет, сообщите, пожалуйста, а то ведь автору плохо спится ☺.

Установите размер шрифта — 12 (Font), поменяйте Caption на слово "Закрыть".

Вот так у вас получилось (рис. 1.11)?

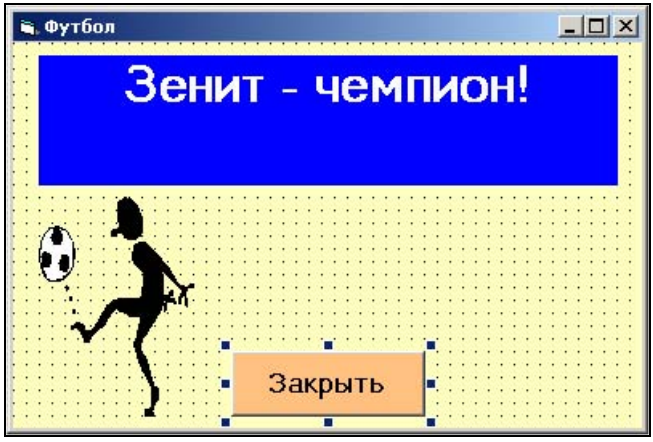


Рис. 1.11. Форма с командной кнопкой

А теперь напишем для этой кнопки наш первый командный код. Дважды щелкнув по кнопке "Закрыть", мы попадаем в окно **Project1 — Form1 (Code)** (рис. 1.12).

Рис. 1.12. Окно программного кода

Как вы видите, все наши описания событий, происходящих в проекте, будут представляться в виде процедур, начало и окончание которых нам любезно предоставляется. В данном случае нам надо описать, что будет происходить с формой после ее запуска по щелчку мышью по кнопке "Закрыть". Мы хотим, чтобы в этом случае форма просто закрывалась. Это описывается очень короткой, но емкой командой `End`, которую мы и впишем в предоставленное нам место (рис. 1.13).

Рис. 1.13. Процедура закрытия формы

Запустим проект и щелкнем по кнопке "Закрыть". Ой, закрылось! И все это сделано вашими замечательными руками и головой!

Разместим на форме еще одну командную кнопку (повыше предыдущей) со свойствами: `Left` — 2040; `Top` — 2800; `Width` — 1815; `Height` — 615; `Caption` — Показать; `Style` — Graphical и каким-нибудь значением цвета для свойства `BackColor` (рис. 1.14).

Мы хотим теперь, чтобы после запуска проекта на форме были бы только командные кнопки и все, а потом, по нажатию кнопки "Показать", увидели бы нашу надпись и картинку (необычайно красивые!).

Во-первых, чтобы текст и картинка изначально не были видны, предварительно выделив их, установите для них свойство `Visible` — False.