Юлий Кетков
Александр Кетков

СВОБОДНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

FREE PASKAL

для студентов и школьников

Санкт-Петербург
«БХВ-Петербург»
2011

УДК 681.3.068(07)+800.92FreePascal
ББК 32.973.26-018.2
К37

Кетков, Ю. Л.

К37 Свободное программное обеспечение. FREE PASCAL для студентов и школьников / Ю. Л. Кетков, А. Ю. Кетков. — СПб.: БХВ-Петербург, 2011. — 384 с.: ил. + CD-ROM — (ИиИКТ)

ISBN 978-5-9775-0604-5

Пособие предназначено для изучения компилятора Free Pascal и интегрированной среды FP IDE.

Подробно разобраны основы программирования на языке Free Pascal: история создания и развития языка Pascal, простые типы данных, строковые данные, структурированные типы данных — массивы. Рассматриваются вопросы организации типовых блоков обработки данных — процедур и функций, работа с файлами. Показаны работа с системными библиотеками и создание собственных библиотечных модулей. Книга включает информацию о возможностях двух графических систем, входящих в поставку FP IDE: модуль Graph, использующий традиционный подход, характерный для графических библиотек версий Turbo Pascal, и современный пакет OpenGL. Весь излагаемый материал ориентирован на учебный процесс, представлено большое количество примеров и программ. Прилагаемый компакт-диск содержит готовую к работе систему программирования Free Pascal, дистрибутив Free Pascal и программы, рассматриваемые в книге.

Для образовательных учреждений

УДК 681.3.068(07)+800.92FreePascal
ББК 32.973.26-018.2

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Людмила Еремеевская*
Зав. редакцией	*Григорий Добин*
Редактор	*Анна Кузьмина*
Компьютерная верстка	*Натальи Смирновой*
Корректор	*Наталия Першакова*
Дизайн серии	*Инны Тачиной*
Оформление обложки	*Елены Беляевой*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 02.09.10.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 30,96.
Тираж 1500 экз. Заказ №
"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

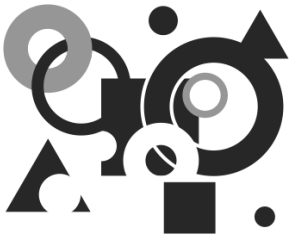
Санитарно-эпидемиологическое заключение на продукцию № 77.99.60.953.Д.005770.05.09
от 26.05.2009 г. выдано Федеральной службой по надзору
в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 978-5-9775-0604-5

Оглавление

Предисловие



Эпоха создания промышленных средств вычислительной техники и разработки программ, вдохнувших жизнь в компьютерное железо, насчитывает немногим более 60 лет. Первый этап программирования в кодах компьютера выявил самое узкое место в этой новой профессии. Выход готовой продукции за год работы достаточно хорошо подготовленного программиста составлял порядка 2—3 тыс. строк машинного кода, что соответствовало примерно 2—3 законченным программам, которые современные системы программирования позволяют создать за несколько дней.

Для преодоления этого препятствия ведущие производители компьютерной техники и научно-исследовательские организации, вкусившие первые плоды новой технологии решения задач, предприняли коллективные усилия по созданию первых алгоритмических языков — Фортрана (1954 г.) и Алгола (1958 г.). Каждая строка, написанная на алгоритмическом языке, превращалась в десятки машинных инструкций, что позволяло на порядок повысить объем создаваемой программной продукции. Серьезный вклад в развитие первого этапа автоматизации программирования в нашей стране был связан с идеей логических схем программ, предложенных А. А. Ляпуновым (1953). Реализация этих идей нашла свое воплощение в первых программирующих программах для отечественных ЭВМ БЭСМ (А. П. Ершов) и "Стрела" (М. Р. Шура-Бура, С. С. Камынин, Э. З. Любимский).

Но быстро развивающаяся сфера компьютерных приложений требовала большего. И очередным прорывом в технологии программирования стала разработка библиотек типовых алгоритмов, реализующих хорошо известные и вновь разрабатываемые численные методы. Новый подход, известный под названием БСП (библиотеки стандартных программ), потребовал консолидации усилий специалистов, работавших в разных прикладных областях. Коллективный разум был необходим и для поиска оптимальных (по быстродействию и памяти) алгоритмов, и для анализа их точности, и для выявления скрытых ошибок. Одними из первых эту проблему осознали ведущие компьютерные издания Нового и Старого света. Упомянем только некоторые из них — Communications of the ACM (Association for Computing Machinery — ассоциация по вычислительной технике, США), Mathematical Tables and other Aids to Computation (Математические таблицы и другие средства вычислений, США), "Zeitschrift fuer Angewandte Mathematik und Physik" (журнал прикладной математики и физики, Германия), журнал "Вычислительная математика и математическая физика" (СССР). В этих журналах за период 60—70-х годов прошлого столетия было опубликовано более тысячи алгоритмов, преимущественно на язы-

ках Фортран и Алгол. Многие из них были положены в основу ряда популярных и общедоступных библиотек стандартных подпрограмм.

Пользователи всемирно известных моделей серий IBM/360 и их отечественных аналогов ЕС-ЭВМ с ностальгией вспоминают библиотеку научных программ SSP (Scientific Subroutine Package). Два раздела программ из этой библиотеки в 1974 г. были переведены на русский язык и опубликованы издательством "Статистика". Наиболее яркий представитель современных систем технических вычислений, — пакет MATLAB, — начинал свое развитие с библиотек подпрограмм линейной алгебры (LINPACK) и набора алгоритмов анализа собственных значений (EISPACK), разработанных и опубликованных в 70-е годы прошлого столетия.

К числу удачных разработок некоммерческого характера того времени относятся проекты по созданию алгоритмических языков C, C++, Pascal и операционной системы UNIX. На начальной стадии в их развитии принимали участие энтузиасты из лаборатории Bell известной американской телефонной компании AT&T (Д. Ритчи, К. Томпсон, Б. Керниган, Б. Страуструп) и сотрудники технологического института в Цюрихе (Н. Вирт, К. Йенсен). Затем эти проекты ушли в самостоятельное плавание, во время которого к их совершенствованию приложили свои усилия другие энтузиасты, сотрудники разных фирм и учебных заведений.

Мы упомянули некоторые моменты в становлении технологии программирования с целью подчеркнуть вклад в мировой прогресс возможности открытого общения представителей науки и техники, далеко не всегда преследовавших коммерческие цели.

Да, конечно разработка программной продукции требует серьезных денежных вливаний. Сложные системы изначально содержат ошибки, которые разработчик должен устранять, функции этих систем приходится развивать, чтобы они отвечали новым требованиям пользователей (вспомните Service Pack 2 и Service Pack 3 для Windows XP). Разработка, поставка и сопровождение программных продуктов — одна из наиболее прибыльных сфер современной рыночной деятельности. Свидетельством тому служит гигантская империя Microsoft, возглавляемая одним из богатейших людей планеты Билом Гейтсом. Для удержания рыночных позиций ведущие разработчики программной продукции тщательно скрывают исходные коды. Более того, в последнее время крупные фирмы всерьез задумываются уже не о продаже пакетов программ конечным пользователям, а о предоставлении им услуг по решению типовых задач на своих серверах. С одной стороны множественная "сдача в аренду" может принести большую прибыль разработчику, с другой стороны, не в накладе остаются и провайдеры компьютерных сетей. Однако высокая стоимость лицензионных услуг порождает не только массовое недовольство пользователей, но и создает питательную среду для компьютерного пиратства.

Всемирная паутина максимально сблизила пользователей разного уровня и молодых профессионалов, желающих обрести заслуженный статус. Одним из перспективных направлений для приложения их усилий является разработка свободного программного обеспечения, доступ к которому открыт для любого пользователя Интернета. Именно так появились международные проекты Free

BASIC, Free Pascal, Open BASIC, OpenOffice и многие другие. Не перевелись еще энтузиасты и филантропы.

Приобретая эту книгу по цене, едва оправдывающей расходы издательства, вы получаете еще и CD-ROM — "удочку, с помощью которой можно научиться ловить рыбу" и подготовить себя к последующему профессиональному росту. В отличие от многих книг, где подобные вложения сопровождаются лишь демо-версиями той или иной системы, на нашем диске представлена полностью готовая к работе самая свежая профессиональная версия бесплатной системы программирования.

Книга состоит из трех частей. *Часть I* составляют главы, которые знакомят читателей с основами программирования на языке Free Pascal и технологией работы в интегрированной среде FP IDE. Эта среда позволяет создавать консольные 32-разрядные приложения Windows, в которых сняты ограничения широко распространенных в нашей стране версий Turbo Pascal и Borland Pascal. В первую очередь это связано с возможностью использовать всю оперативную память, объем которой на современных ПК превышает 2—3 Гбайт. Вдобавок, компилятор Free Pascal "знаком" с расширенным набором команд современных микропроцессоров и может использовать их мощь при создании приложений. Наконец, прикладные программы получили доступ к широкому спектру услуг операционной системы и богатой коллекции статических и динамических библиотек. Достаточно важным преимуществом компилятора FPC (Free Pascal Compiler) является его универсальность. Наряду с 32-разрядными приложениями он может создавать и 64-разрядные приложения. Различные версии FPC работают под управлением разных операционных систем практически на всех средствах вычислительной техники. Существуют и так называемые кроссплатформенные версии FPC, когда вы на компьютере одного типа создаете приложение, предназначенное для работы на компьютере другого типа.

Глава 1 содержит краткий экскурс в историю создания языка Pascal и его развития. Она не является глубоким исследованием, затрагивающим влияние ряда личностей и фирм на развитие языка. Мы осведомлены о вкладе британских ученых в становление первого стандарта языка, но очень слабо информированы о попытках внесения в Pascal элементов объектно-ориентированного подхода, предпринятых в калифорнийском университете UCSD (University of California at San Diego). В нашей стране история языка ассоциируется с десятком ступенек, по которым прошли все отечественные пользователи, осваивавшие продукцию фирмы Borland.

Глава 2 знакомит читателей с набором нескольких достаточно простых по замыслу, но разнообразных по назначению программ со способами их организации и выполнения в интегрированной среде. Ее цель — продемонстрировать структуру программы и некоторые простейшие приемы программирования.

В *главе 3* достаточно подробно описывается интегрированная среда FP IDE. Перечень функциональных услуг, представляемых командами главного меню и многочисленными диалоговыми окнами, всплывающими во время работы, достаточно велик. Нельзя обойтись без их детального описания и рекомендаций по их практическому использованию. Однако к некоторым услугам мы вынуждены при-

бегать ежедневно, другие могут оказаться менее востребованными, а в отдельные команды и подкоманды главного меню пользователь может ни разу и не заглянуть. Так как довольно сложно прогнозировать уровень запросов различных пользователей, то мы включили в состав третьей главы тот набор функциональных услуг, который рано или поздно понадобится всем. А сведения по деталям настройки многочисленных параметров среды вынесли в приложение.

Глава 4 посвящена описанию простых типов данных, без которых не обходится ни одна программа. Здесь приводятся сведения о внешнем и внутреннем представлении числовых, символьных и логических данных, о способах ввода их значений и вывода результатов работы программы, об операциях, выполняемых над данными разного типа.

В *главе 5* рассматриваются строковые данные, представляющие в компьютере текстовую информацию — один из наиболее распространенных объектов массовой обработки. Наряду с традиционными для языка Pascal "короткими строками" значительное внимание уделено более современным наборам строк неограниченной длины, допускающим как однобайтовую кодировку ASCII, так и двухбайтовую Unicode. Приводятся сведения о типовых операциях, процедурах и функциях, связанных с обработкой текстовой информации, с прямыми и обратными преобразованиями числовых и символьных данных.

Глава 6 знакомит читателей с наиболее распространенным представителем структурированных типов данных — массивами. Здесь описываются преимущества, представляемые массивами при составлении программ с элементами циклической обработки. Приводятся фрагменты программ, демонстрирующие ряд операций линейной алгебры. Не забыт и раздел, связанный с использованием динамических массивов, память для которых выделяется и освобождается во время работы программы.

В *главе 7* рассматриваются объекты типа "множество". Этот тип данных, изначально появившийся в языке Pascal, не очень характерен для других алгоритмических языков. Но он немного расширил сферу приложения математических методов хотя бы в плане использования новых понятий и операций. В качестве примера построена модель "решета Эратосфена" для определения простых чисел.

Понятию *"запись"*, обобщающему наши представления о строках таблицы, посвящена *глава 8*. Этот тип данных очень востребован при обработке табличной информации, хранящейся как в оперативной памяти, так и расположенной на внешних носителях (преимущественно в файлах).

Одним из самых важных разделов *части I* является *глава 9*, в которой рассматриваются вопросы организации типовых блоков обработки данных — процедур и функций. Особое внимание уделено механизму общения по данным между автономными программными единицами. Приводится несколько способов передачи параметров в вызываемые процедуры (функции) с объяснением преимуществ и недостатков каждого из них. В качестве аргументов функций и процедур демонстрируется использование одномерных и двумерных массивов, параметров процедурного типа, параметров по умолчанию. На примере некоторых рекурсивных

функций отмечаются как положительные, так и отрицательные стороны этого приема программирования.

Последняя глава *части I* посвящена работе с файлами. Как показывает опыт проведения олимпиад, в школьной информатике этому разделу уделяется недостаточное внимание. Однако большинство практических приложений извлекают свои исходные данные обычно из текстовых или двоичных файлов и сохраняют результаты своей окончательной или промежуточной работы также в файлах. Файлы являются основным средством обмена информацией во всемирной паутине.

Часть II книги посвящена использованию системных библиотек и созданию собственных библиотечных модулей. В *главе 11* описывается структура модуля и приводится список основных модулей, поставляемых в составе системы FP IDE. На примере данных типа рациональные дроби демонстрируется, как построить модуль для обработки данных нового типа. С применением элементов объектно-ориентированного подхода во вновь построенном модуле вводится понятие *объекта* и строится модель библиотеки, предоставляющей более удобные средства интерфейса.

Глава 12 знакомит читателя с возможностями программного управления дисплеем в текстовом режиме (модуль `CRT`). Здесь описываются как функции, непосредственно связанные с заданием параметров окна вывода (положение, размер, позиция курсора, цветовые атрибуты текста), так ряд вспомогательных функций, обеспечивающих редактирование текста на экране, общение с клавиатурой.

В *главе 13* приводятся сведения о функциях, которые можно условно назвать математическими. Большая их часть сосредоточена в модуле `Math`, некоторое количество содержится в модуле `System`. В главе описываются детали использования относительно мало известных функций (в частности функций округления и статистической обработки данных).

Глава 14 посвящена вопросам обработки календарных дат и интервалов времени с использованием таких компонентов операционной системы, как календарь и часы, и процедур, включенных в модуль `DateUtils`. Приводится краткая историческая справка о разных этапах измерения времени. Подробно описываются возможности полутора сотен процедур модуля, включенного в состав FP IDE.

Часть III книги включает информацию о возможностях двух графических систем, входящих в поставку FP IDE. Первая из них (модуль `Graph`) использует традиционный подход, характерный для графических библиотек версий Turbo Pascal. Здесь в полном объеме сохранен графический интерфейс, присущий стандарту BGI (Borland Graphics Interface). Это означает, что сохранен весь набор графических процедур, но учтены новые функциональные возможности современных дисплеев (разрешение, новые цветовые гаммы, пока еще не поддерживаемые в полном объеме). Вторая графическая система — современный пакет OpenGL (Open Graphics Library — открытая графическая библиотека).

Глава 15 посвящена полному описанию графических возможностей модифицированного стандарта BGI. В *главе 16* изложены основы работы с пакетом OpenGL. В рамках данной книги отсутствует возможность описания более чем 250 процедур

самого пакета OpenGL и большого количества библиотек, так или иначе поддерживающих базовое графическое ядро и его расширения.

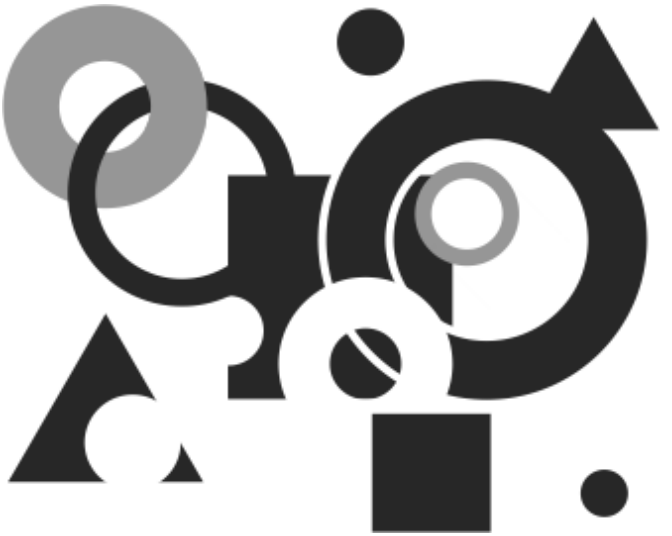
В *приложение 1* включена краткая справка по синтаксису и семантике основных объектов языка Free Pascal — данных и операторов. *Приложение 2* содержит сведения, необходимые для настройки параметров компилятора FPC и интегрированной среды. Таблица с сообщениями об ошибках во время выполнения приложения входит в состав *приложения 3*. *Приложение 4* содержит описание компакт-диска.

Кроме того, в книге представлен список литературы, которой позволит вам расширить свои знания.

Благодарности

Это пятая книга, написанная авторским коллективом Кетковых и выпускаемая издательством "БХВ-Петербург". И все эти годы авторы совместно и плодотворно работали с редакторами издательства, ощущая их благожелательное отношение и профессионализм. В этой книге особую благодарность заслуживает Анна Сергеевна Кузьмина.

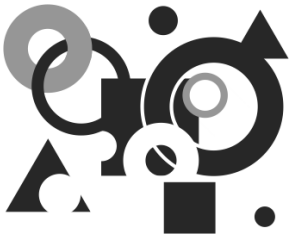
Дополнительно хотелось бы выразить признательность Александру Ивановичу Кузнецову, сотруднику НИИ ПМК, с которым авторы книги на протяжении трех лет обкатывали систему Free Pascal на студенческих и школьных олимпиадах Нижегородской области.



ЧАСТЬ I

Основы программирования в среде FP IDE

Глава 1



Введение в Free Pascal

1.1. Исторический обзор

В 1965 г. был объявлен международный конкурс на создание нового алгоритмического языка — преемника АЛГОЛА-60. В конкурсе принял участие молодой швейцарский ученый Никлаус Вирт (Nicklaus E. Wirth), работавший в то время доцентом Стэндфордского университета. В 1967 г. его проект комиссией был отвергнут — победителем оказался АЛГОЛ-68. Но Вирт продолжил работу над своим замыслом и, вернувшись в Швейцарию, вместе с коллегами из Федерального института технологии (ETH, Цюрих) разработал новую версию языка, названную в честь известного французского инженера Блеза Паскаля — создателя одного из первых механических калькуляторов. В 1970 г. под руководством Н. Вирта был разработан первый транслятор с языка Паскаль, а в 1971 г. появилась первая публикация — техническая документация для пользователей новой системы программирования. Гораздо более известная публикация — совместная книга Никлауса Вирта и его помощницы Кэтлин Йенсен появилась через четыре года. В 1982 г. она была переведена на русский язык. В 1987 г. за создание языка Паскаль Н. Вирт был удостоен медали "Пионер вычислительной техники" (Computer Pioneer) — самой престижной награды Международного компьютерного сообщества (IEEE Computer Society). В первую очередь, язык Паскаль предназначался для использования в учебных заведениях. Цели и задачи, стоявшие перед создателями Паскаля, включали:

- удовлетворение требованиям структурного программирования (программа должна состоять из небольшого количества типовых, легко заменяемых синтаксических конструкций — блоков);
- развитие набора структурированных данных и приемов их обработки (в первую очередь, записей — аналогов хорошо известных таблиц);
- написание надежных программ (за счет введения строгой типизации данных и запрета на использование различных правил умолчания);
- краткость языка (первое полное описание Паскаля занимало 30 страниц против 600 страниц руководства по языку PL/1 в документации по IBM/360).

Важную роль в разработке стандарта языка сыграл Британский институт стандартизации и его рабочая группа во главе с А. Эддиманом. Первый британский

стандарт BS6192 появился в 1982 г., а год спустя Международная организация по стандартам (International Organization for Standardization, ISO) приняла соответствующий документ — ISO 7185. Следует сказать, что Н. Вирт отрицательно отнесся к расширениям языка, предложенным британскими коллегами. Он считал, что нововведения нарушают основные принципы, изложенные выше. Поэтому в дальнейшем Н. Вирт отошел от Паскаля и занялся развитием новых систем программирования на базе языков Modula, Oberon, Zonnon. Наверное, он не во всем был прав: язык программирования — не только средство обучения, но и рабочий инструмент для решения практических задач.

Паскаль довольно долго оставался средством для изучения программирования в учебных заведениях, т. к. ни одна серьезная компьютерная фирма его не поддерживала. Перелом по отношению к Паскалю наметился в 1983—1984 гг., когда за его реализацию взялся молодой французский студент Филипп Канн. Предварительно он прошел длительную стажировку у Н. Вирта, после чего написал сверхскоростной компилятор Turbo Pascal и придумал удачную интегрированную среду (IDE, Integrated Development Environment — интегрированная среда разработки) для только что появившихся персональных компьютеров IBM PC. Среда объединяла редактор исходного кода, компилятор, загрузчик и не очень сложные средства отладки. Вместо многократного формирования командных строк по запуску того или иного компонента системы программирования в интегрированной среде достаточно было нажать пару кнопок. Простота и скорость работы в интегрированной среде послужили одним из главных факторов для привлечения массового пользователя.

В дальнейшем многие фирмы заимствовали идеи Филиппа Канна в своих программных продуктах. Для завоевания рынка Ф. Канн отправился в США, занял деньги у своих дальних родственников и начал продавать Turbo Pascal по смехотворной цене — $49,95 (для сравнения напомним, что незадолго до этого Билл Гейтс ухитрился назначить цену в $500 за интерпретатор Бейсика на домашнем компьютере Altair). Торговый успех (за первый месяц после серьезной рекламной подготовки было продано порядка 3000 копий) заложил фундамент для создания фирмы Borland International. Ее дешевая программная продукция, на ура воспринятая во всем мире и особенно в нашей стране, быстро заполнила вакуум инструментальных средств на IBM-совместимых ПК, где кроме ассемблера и встроенного Бейсика ничего практически не было. Первый коммерческий успех фирмы Borland был достигнут после появления версии Turbo Pascal 2.0 (середина 1984 г.). За ней последовали более развитые версии 3.0 с "черепашьей" графикой (осень 1985 г.), 4.0 (начало 1988 г.) с полноценной графической библиотекой BGI (Borland Graphics Interface), 5.0 (август 1988 г.) с развитыми средствами отладки и возможностью построения оверлейных программ. В мае 1989 г. появилась версия 5.5, в которой были заложены основы объектно-ориентированного подхода. Развитие этого направления продолжилось в версиях TP 6.0 (1990) и TP 7.0 (1992). Последняя версия была выпущена в двух модификациях — для разработки приложений только MS-DOS (TP 7.0) и приложений как MS-DOS, так и Windows (BP 7.0).

Следующий серьезный прорыв в развитии языка Паскаль был связан с преодолением препятствий, выдвигаемых операционной системой при создании 32-разрядных приложений, функционирующих под управлением Windows. Прежние 16-разрядные приложения, создававшиеся в системах программирования Turbo Pascal и Borland Pascal, упирались в серьезные ограничения как по использованию ресурсов (работа с большой оперативной и внешней памятью, новые возможности расширенной системы команд и т. п.), так и по стандартным требованиям к организации приложений, выдвигаемым операционной системой. Первый шаг в преодолении барьера Windows для непрофессиональных пользователей был предпринят в конце 1991 г. компанией Microsoft, которая выпустила на рынок хит последующих 3—4 лет — систему визуального программирования Visual Basic. Ее примеру последовала и фирма Borland, которая уже в 1995 г. разработала на базе языка Object Pascal среду визуального программирования Delphi. Объектно-ориентированный подход, реализованный в языке Object Pascal, дал пользователям Delphi большое преимущество в создании новых компонентов, реализующих не только интерфейсные функции. В системе Visual Basic для разработки VBX- и OCX-компонентов применялись системные средства, не доступные рядовому пользователю. Поэтому появление Delphi было с восторгом воспринято не только мало искушенными пользователями языка Паскаль, но и системными программистами. Благодаря этой среде за относительно короткий срок фирме Borland удалось выпустить на рынок систему визуального программирования на базе языка C++ (Borland C++Builder).

В нашей стране продукция фирмы Borland получила широкое распространение, в первую очередь, в учебных заведениях средней и высшей школы. Ее охотно используют научные и технические предприятия для разработки новых программных продуктов. Существует обширный список литературы на русском языке, посвященной программированию в средах Turbo Pascal, Borland Pascal и Delphi.

Однако не следует забывать, что, во-первых, все упомянутые системы программирования являются коммерческими продуктами, затраты на легальное приобретение которых не всегда по карману многим отечественным пользователям. В этом плане особенно страдают от нехватки средств высшие и средние учебные заведения. Во-вторых, кроме операционных систем типа Windows на IBM-совместимых персональных компьютерах расширяется использование существенно более надежных и более компактных операционных систем типа Linux с сопутствующим программным обеспечением, разрабатываемым и распространяемым на некоммерческой основе в соответствии с правилами FSF (Free Software Foundation). Основное соглашение этого фонда известно под аббревиатурой GPL (General Public License) или, как неологизм, *copyleft* (в противовес известному символу авторского права *copyright*). Это соглашение-лицензия является обязательным для всех разработчиков и пользователей свободно распространяемого программного обеспечения (ПО). Смысл его довольно простой: получив бесплатно дистрибутив ПО вместе со всеми исходными текстами программ, вы можете доработать любой компонент этого ПО и устранить замеченные ошибки. Полученный или модифицированный таким образом дистрибутив вы не имеете права включать в состав любого коммер-

чески создаваемого продукта. Более того, продавая или передавая созданную вами модификацию третьим лицам, вы обязаны передать им дополнительно все исходные тексты своих изменений и пополнить список авторов, включая всех своих предшественников. Такого рода программные продукты, содержащие элементы творчества многих профессионалов, не обязательно являющихся членами первоначального коллектива разработчиков, составляют категорию проектов GNU. К числу наиболее известных GNU-проектов относятся различные версии операционной системы Linux, компиляторы с языков ассемблера (GA.exe, GAS.exe), C и C++ (GCC.exe), Фортран (GFortran.exe), универсальный редактор Emacs, отладчик (GDb.exe), библиотека научных программ (GSL) и др.

В 1993 г. в Интернете стартовал очередной GNU-проект Free Pascal Compiler (сокращенно — FPC), который поначалу ставил целью разработку 32-разрядного компилятора входного языка Turbo Pascal для нескольких операционных систем (DOS, Windows, Linux, ...) и персональных компьютеров разных производителей (*x*86, AMD, Power PC, ...). Инициатором этого проекта был немецкий программист Флориан Клэмпфель (Florian Klämpfl, **florian@freepascal.org**). Следует отметить, что основная часть компилятора FPC писалась на Паскале, и уже к началу 1995 г. появилась версия, которая могла компилировать собственный текст, переводя его код на язык ассемблера IBM PC. В настоящее время на сайте разработчиков (**http://www.freepascal.org**) доступна версия FPC-2.2.4, которая датируется апрелем 2009 г. Говорят, что аппетит приходит во время еды. Поэтому авторов проекта FPC потянуло на большее — они решили создать систему программирования, которая бы не только понимала последнюю версию языка Turbo Pascal, но и могла поддерживать языковые возможности системы Delphi.

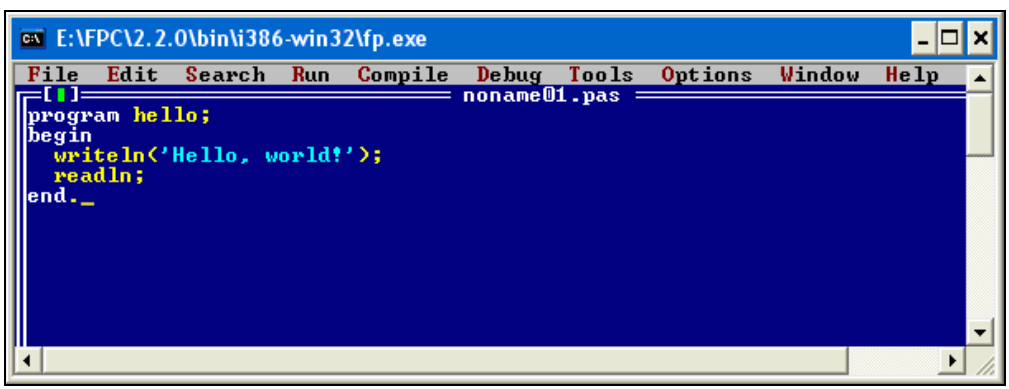


Рис. 1.1. Внешний вид интегрированной среды Free Pascal IDE for Win32 for i386

Для этой цели пришлось усовершенствовать не только компилятор fpc.exe, но и разработать несколько вариантов интегрированных сред, которые бы автоматически формировали запуск таких автономных GNU-утилит, как компилятор с ассемблера ga.exe, библиотекарь (архиватор) ar.exe, загрузчик ld.exe и отладчик gdb.exe.

Первая такая среда (Free Pascal IDE for Win32 for i386) почти один в один напоминала интегрированные среды ранних версий фирмы Borland (рис. 1.1). Ее разработка была начата в 1998 г. и продолжается по сию пору. Последняя версия среды (ver 1.0.10) датируется апрелем 2009 г. Ее авторами являются венгр Бржи Габор (Bbrczi Gabor), француз Пьер Мюллер (Pierre Muller, **muller@janus.u-strasbg.fr**) и голландец Петер Времан (Peter Vreman, **peter@freepascal.org**).

Годом позднее стартовал более амбициозный проект Lazarus. Интерфейс Lazarus IDE и возможности этой среды напоминают систему визуального программирования Delphi. Приведенная на рис. 1.2 бета-версия IDE датируется мартом 2009 года. К ее разработке и наполнению визуальными компонентами причастен довольно большой коллектив программистов из разных стран.

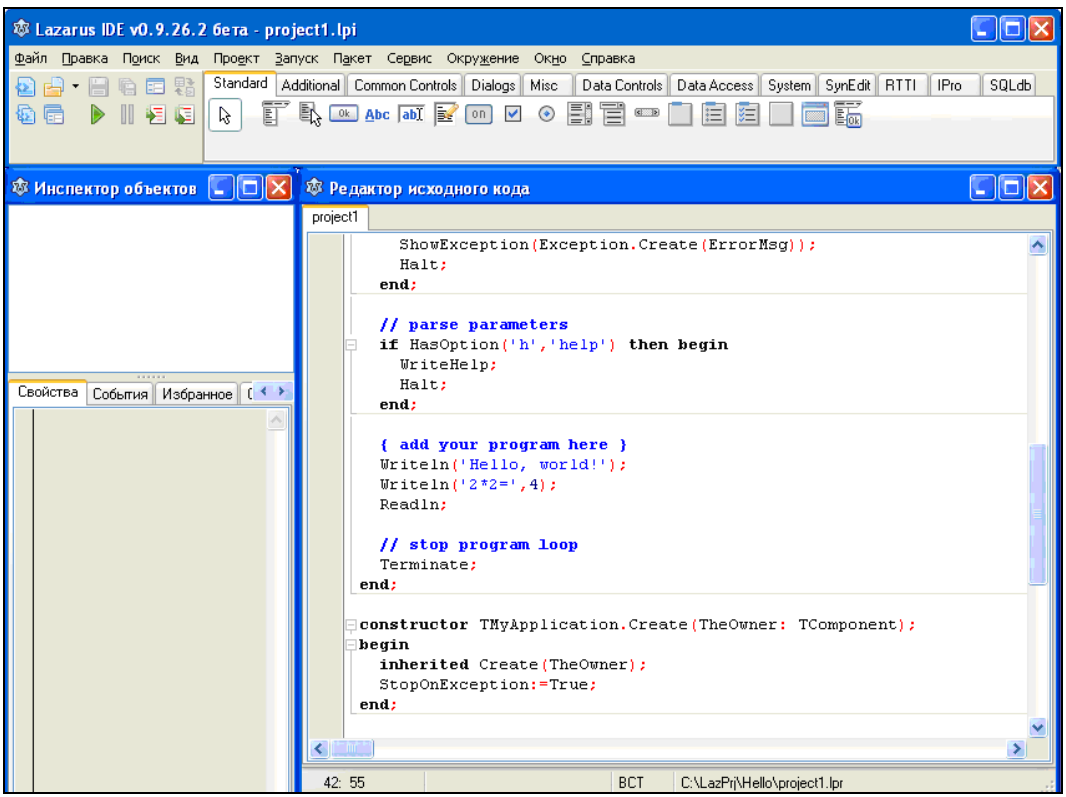


Рис. 1.2. Интегрированная среда Lazarus

В справке о среде содержится следующий текст: "Lazarus — это IDE для создания (графических и консольных) приложений при помощи компилятора Free Pascal. Free Pascal — это компилятор языков Pascal и Object Pascal, распространяемый под лицензией (L)GPL и работающий под Windows, Linux, Mac OS X, FreeBSD, и не только.

Lazarus — это недостающий элемент, который позволит вам разрабатывать программы для всех вышеперечисленных платформ в Delphi-подобном окружении. Эта IDE является инструментом RAD (Rapid Application Development — быстрая разработка приложений), включающим в себя дизайнер форм".

Добавим к этому новую возможность создания 64-разрядных приложений для постоянно возрастающего количества операционных систем (DOS, Win32, OS/2, Linux, FreeBSD, NetBSD, AmigaOS, BeOS, Mac OS и др.) и разнообразных технических платформ (*x*86, *x*86_64, AMD64, ARM, Motorola, PowerPC, Sparc и т. д.).

1.2. Структура программы на языке Free Pascal

Структура программы на языке Free Pascal по форме мало чем отличается от установившихся правил оформления программ в системах Turbo Pascal и в консольных приложениях Delphi (листинг 1.1).

Листинг 1.1. Структура программы

```
program prog_name;
uses Unit1,Unit2,...;        {список используемых модулей}
{-------------------------------------------------------}
const
  {раздел объявления глобальных констант}
type
  {раздел описания глобальных типов}
var
  {раздел объявления глобальных переменных}
label
  {раздел объявления меток в теле основной программы}
procedure
  {раздел описаний пользовательских процедур}
function
  {раздел описаний пользовательских функций}
//-------------------------------------------------------}
begin
  {тело основной программы}
end.
```

В приведенной схеме жирным шрифтом выделены наиболее существенные служебные слова языка программирования, которые предшествуют тому или иному фрагменту программы. Штриховые линии делят программу на три части.

Оператор `program`, с которого начинается первая часть, носит название *заголовка программы*. Он не является обязательным, но его рекомендуется включать в текст программы для указания ее имени (`prog_name`). В первой части программы с помощью служебного слова `uses` (от англ. *uses* — использует) перечисляются имена модулей, которые компилятор должен подключить к исполняемой программе. Модуль (по терминологии Паскаля — `Unit`) играет роль библиотеки подпрограмм (функций и процедур). Он может быть системным или пользовательским. Для обеспечения работы вашей программы используемые модули должны быть предварительно протранслированы и находиться на диске в виде файлов с расширениями tpu (от Turbo Pascal Unit). Если программа использует наиболее востребованный системный модуль `System`, то он подключается автоматически.

Вторую часть программы составляют разделы *объявлений* и *описаний*. Все версии систем программирования на базе языка Паскаль, следуя традициям фирмы Borland, разрешают перечислять приведенные выше разделы в произвольном порядке и даже фрагментировать содержимое любого раздела. При этом должно соблюдаться единственное правило. Если описание/объявление `S1` используется в описании/объявлении `S2`, то `S1` должно быть описано первым. Термином *"глобальный"* мы хотели подчеркнуть, что соответствующие объекты (константы, типы, переменные) можно использовать не только в теле основной программы, но и в любой процедуре/функции, входящей в соответствующий раздел ваших описаний. Любой из разделов второй части может отсутствовать.

В теле основной программы должно быть описано хотя бы одно разумное действие, оправдывающее использование компьютера. Например, практически все руководства по различным алгоритмическим языкам не обходятся без программы, приветствующей мир (листинг 1.2).

Листинг 1.2. Программа `Hello`

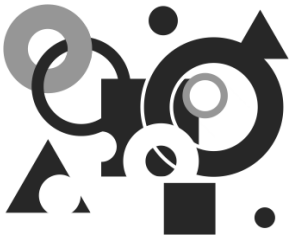
```
Program Hello;
begin
  writeln('Hello, world!');
end.
```

В оформлении описания любой функции или процедуры, включенной в текст вашей программы, присутствуют почти все те же элементы, которые имеют место в основной программе.

Есть только две характерные особенности:

- *заголовок функции* начинается с обязательного оператора `function`.
  Например:
  ```
  function rasst(x1,x2:double):double;
  ```
- *заголовок процедуры* начинается с обязательного оператора `procedure`.
  Например:
  ```
  procedure sum(a,b:double; var c:double);
  ```
- завершающий `end` заканчивается не точкой, а *точкой с запятой*.

Таким образом, программу на языке Free Pascal можно представлять в виде большой матрешки (*основной* или *головной* программы), в которую *вложены* описания используемых данных и подпрограмм. В свою очередь, каждая подпрограмма (процедура или функция) имеет вид *вложенной матрешки*. Этим Паскаль принципиально отличается от языков Basic, C, Fortran, где вложения подобного рода запрещены. Глубина вложения каким-то конкретным числом не ограничена, но на практике довольно редко можно встретить программы с уровнем вложения более 4—5.



Глава 2

Знакомство с простыми программами

Эта глава предназначена для начинающих изучать Паскаль, и ее без ущерба могут пропустить читатели, изучавшие Паскаль в школе или в институте. Хотя повторение — мать учения. Задачи, представленные в этой главе, знакомят читателя с видом программы и некоторыми приемами программирования.

Задача 2.1

Необходимо написать программу, которая вычислит и выведет на экран результаты в виде таблицы (табл. 2.1).

Таблица 2.1

x	sin(x)	cos(x)
0.0	0.000000	1.000000
0.1	0.099833	0.995004
0.2	0.198669	0.980067
0.3	0,295520	0.955336
0.4	0.389418	0.921061
0.5	0.479426	0.877583
0.6	0.564642	0.825336
0.7	0.644218	0.764842
0.8	0.717356	0.696708
0.9	0.783327	0.621610

Обратите внимание на то, что в таблице аргумент `x` задается в радианах. Хотелось бы написать более универсальную программу, которая выводит `k` строк подобной таблицы, начиная с заданного значения `x0`. Приведенный выше фрагмент таблицы должен получиться при `k=10` и `x0=0`.

Самый простой (и одновременно самый тупой) вариант тела программы может состоять из строк, представленных в листинге 2.1.

Листинг 2.1. Программа `table1`

```
program table1;
var
  x:double;
begin
  writeln('x','    sin(x)','    cos(x)');
  x:=0.0;
  writeln(x,sin(x),cos(x));
  x:=x+0.1;
  writeln(x,sin(x),cos(x));
  x:=x+0.1;
  writeln(x,sin(x),cos(x));
  x:=x+0.1;
  writeln(x,sin(x),cos(x));
  x:=x+0.1;
  writeln(x,sin(x),cos(x));
  x:=x+0.1;
  writeln(x,sin(x),cos(x));
  x:=x+0.1;
  writeln(x,sin(x),cos(x));
  x:=x+0.1;
  writeln(x,sin(x),cos(x));
  x:=x+0.1;
  writeln(x,sin(x),cos(x));
  x:=x+0.1;
  writeln(x,sin(x),cos(x));
end.
```

В программе `table1` использована единственная переменная с именем `x`, которая может принимать вещественные значения с удвоенной точностью (тип `double`). Первый оператор `writeln` предназначен для вывода заголовка таблицы. Его три аргумента представлены строковыми константами, которые, в принципе, можно было бы объединить в одну строку. Затем переменной `x` присваивается начальное значение. Операция присваивания, образованная двумя символами `:=`, унаследована Паскалем от своего предшественника — языка АЛГОЛ-60. После этого 10 раз повторяется пара операторов, которая выводит на экран значения `x`, `sin(x)` и `cos(x)` и увеличивает значение переменной `x` на `0.1`.

В поле редактора FPC IDE эта программа выглядела бы следующим образом (рис. 2.1).

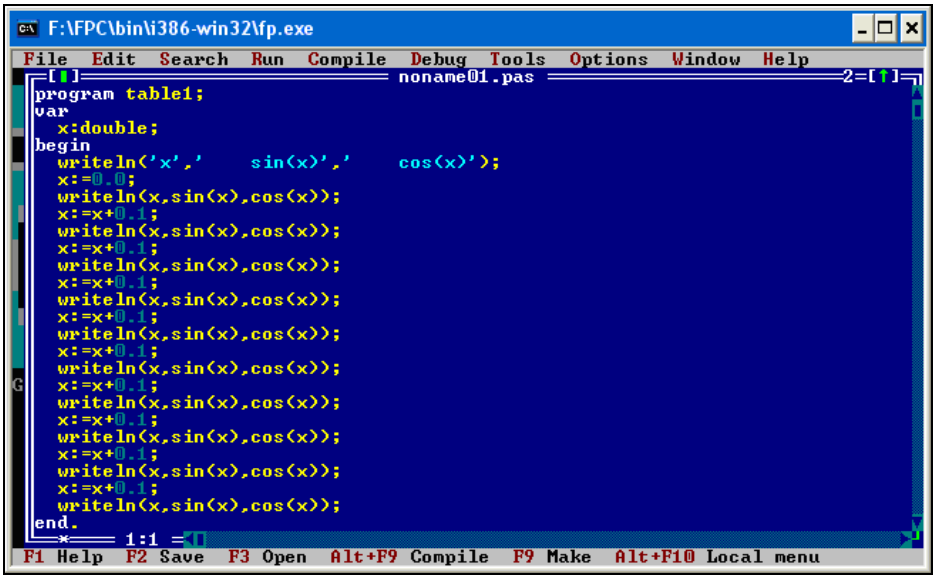


Рис. 2.1. Внешний вид первого варианта программы

Попытка запустить эту программу (команда **Run → Run**) будет приостановлена средой, предлагающей сначала сохранить текст, набранный в поле редактора (рис. 2.2). Такое сохранение вновь набранного или только что модифицированного текста надо обязательно делать перед запуском программы, т. к. во время ее выполнения могут произойти непредвиденные события, и текст программы может пропасть.

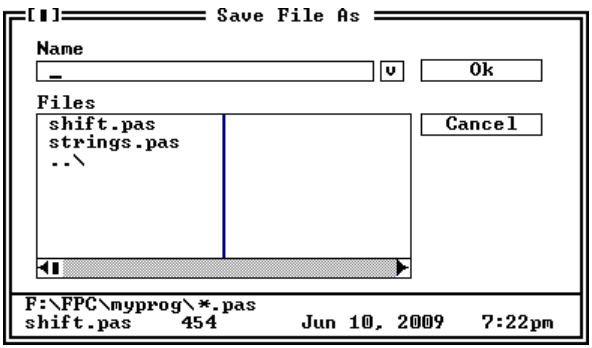


Рис. 2.2. Окно для сохранения текста программы

Сохранив программу под именем table.pas (расширение среда добавляет сама) и снова запустив ее, мы не заметим на экране никаких изменений. Дело в том, что после выполнения программы управление автоматически возвращается среде, а за то ничтожное время, которое программа работала, результаты промелькнули на экране слишком быстро. *Увидеть их можно, нажав комбинацию клавиш*

<Alt>+<F5>. Однако, чаще всего перед заключительным `end` вставляют оператор `readln`, который приостановит работу программы до нажатия клавиши <Enter>.

Результаты, выданные программой и приведенные на рис. 2.3, очень слабо напоминают вид таблицы, указанной в задании. Во-первых, по умолчанию использован так называемый "научный формат" записи вещественных чисел. Символ "E" означает, что число, расположенное слева, надо умножить на 10 в степени, расположенной справа. Во-вторых, нам выдано слишком много значащих цифр.

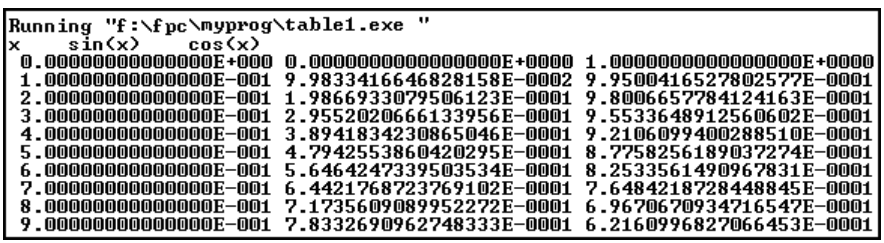


Рис. 2.3. Выдача результатов программы table1.pas

С этим легко побороться — достаточно в операторе `writeln` после каждого выводимого значения указать его общую длину и количество цифр в дробной части:

```
writeln(x:4:1, sin(x):10:6, cos(x):10:6);
```

Однако самая большая нелепость в первом варианте программы заключается в десятикратном повторении однотипных операций. Любой алгоритмический язык, и Free Pascal в том числе, предусматривает возможность циклического повторения таких действий. Этих способов не один, мы воспользуемся оператором цикла `for`. В результате второй вариант нашей программы будет выглядеть так, как представлено в листинге 2.2.

Листинг 2.2. Программа `table2`

```
program table2;
var
  x:double;
  k:integer;          {целое, счетчик циклов}
begin
  writeln('x','    sin(x)','    cos(x)');
  x:=0;
  for k:=1 to 10 do  {тело цикла повторяется 10 раз}
    begin             {начало тела цикла}
      writeln(x:4:1,sin(x):10:6,cos(x):10:6);
      x:=x+0.1;
    end;              {конец тела цикла}
  readln;             {задержка для просмотра результатов}
end.
```

Второй вариант программы выводит результаты в соответствии с заданием. Единственная небрежность — заголовок таблицы немного смещен относительно столбцов соответствующих значений (рис. 2.4).

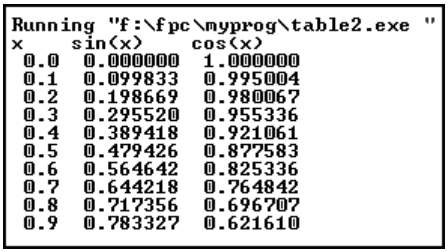


Рис. 2.4. Усовершенствованный вывод

Это легко исправить, добавив к оператору вывода заголовков указание о требуемой длине выводимого текста или нужное количество пробелов в соответствующих текстовых константах:

```
writeln('  x','    sin(x)','    cos(x)');
```

Завершая работу над первой программой, мы можем включить в нее ввод таких данных, как начальное значение `x` и количество `k` выводимых строк таблицы (листинг 2.3).

Листинг 2.3. Программа `table3`

```
program table3;
var
  x:double;
  j,k:integer;        {j - счетчик циклов}
begin
  readln(k,x);        {ввод исходных данных}
  writeln('  x','    sin(x)','    cos(x)');
  for j:=1 to k do    {тело цикла повторяется k раз}
    begin
      writeln(x:4:1,sin(x):10:6,cos(x):10:6);
      x:=x+0.1;
    end;
  readln;
end.
```

Обратите внимание на ввод числовых данных по запросу программы — в нашем примере программа остановит свою работу на первом же операторе тела программы и будет ждать завершения ввода значений `k` и `x` с клавиатуры. Между набираемыми значениями должен быть хотя бы один пробел. Внутри чисел пробелы

недопустимы, ибо они рассматриваются как разделители числовых данных. Признаком завершения набора является нажатие клавиши <Enter>.

Задача 2.2

Натуральное число *N* называется *простым*, если оно имеет только два разных делителя — 1 и само себя. Примеры простых чисел — 2, 3, 5, 7, 11, 13, ... Некоторые математики считают, что 1 не является простым числом, т. к. имеет единственный делитель. Считать 1 составным числом тоже нет никакого резона. Задачей нашей следующей программы является запрос у пользователя числа *N* и анализ его на простоту.

В математике известен достаточно сложный алгоритм такого анализа, требующий минимального числа операций. Но мы воспользуемся следующим алгоритмом, более затратным по времени, но более простым по реализации. Если число N — четное и не равно 2, то оно составное. Если N — нечетное, то будем делить его на последовательные нечетные числа (3, 5, 7, 9, 11, ..., M, где M не превосходит $\sqrt{N}$) и анализировать остатки от деления. Первый же нулевой остаток свидетельствует о том, что N — составное число. Если все остатки будут отличны от 0, то N — простое.

Первый вариант программы, реализующий описанный алгоритм, может выглядеть так, как представлено в листинге 2.4.

Листинг 2.4. Программа `prime1`

```
program prime1;
const
  msg1=' is prime';
  msg2=' is not prime';
label
  m1;
var
  d,N: word;                          {целое, без знака}
  msg: string;                        {строка сообщения}
begin
  write('N = ');                      {вывод приглашения}
  readln(N);                          {ввод N}
  msg := msg1;                        {на случай, если N простое}
  if N<4 then goto m1;                {при N<4 - простое}
  msg := msg2;                        {на случай, если N составное}
  if not odd(N) then goto m1;         {при N четном - составное}
  d:=3;                               {первый делитель - 3}
  while d*d<=N do                     {пока делитель меньше корня из N}
    begin
      if (N mod d)=0 then goto m1;    {если остаток = 0}
```

```
      d:=d+2;                    {увеличение делителя на 2}
    end;
  msg:=msg1;
m1:
  writeln(N,msg);                {вывод сообщения }
  readln;
end.
```

В этой программе появились новые объекты и операторы. Во-первых, в разделе констант мы заготовили два сообщения — `msg1` и `msg2`, одно из которых по результатам анализа будет присвоено переменной строкового типа `msg`. Во-вторых, у нас появился раздел меток, в котором описана метка `m1`. Метка располагается перед оператором, на который планируется переход с помощью оператора `goto`, и отделяется от него двоеточием. В-третьих, мы воспользовались операцией `mod`, которая находит *остаток* от деления двух целочисленных операндов. Еще одна новая функция `odd` принимает значение "истина" (`true`), если ее аргумент *нечетный*. Наконец, в нашей программе активно используется условный оператор `if`, который в случае истинности заданного условия выполняет оператор, расположенный вслед за служебным словом `then`. Если заданное условие не выполнено, то срабатывает оператор, следующий за `if`.

Предположим, что для решения каких-то других задач нам потребуется не один раз заниматься подобным анализом. В этом случае было бы полезно выделить фрагмент проверки на простоту в отдельную функцию, например, с именем `prime`, которая бы возвращала логическое значение `true` (истина), если ее аргумент — простое число. Тогда второй вариант программы мог бы выглядеть так, как представлено в листинге 2.5.

Листинг 2.5. Программа `prime2`

```
prog prime2;
var
  N: Word;
function prime(N:Word):boolean;
var
  d: Word;
begin
  Result:=true;
  if N<=3 then exit;
  Result:=false;
  if not odd(N) then exit;
  d:=3;
  while d*d<=N do
```