НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

В.М. Пестриков
А.Н. Маслобоев
О.К. Федоров

Основы программирования в системе Borland Delphi

Учебное пособие

Санкт-Петербург
2004

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

Государственное образовательное учреждение
высшего профессионального образования

Санкт-Петербургский государственный
технологический
университет растительных полимеров

В.М. Пестриков
А.Н. Маслобоев
О.К. Федоров

Основы программирования в системе Borland Delphi

Учебное пособие

Санкт-Петербург
2004

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

ББК 32.97Я7
П 286
УДК 681.3 (075)

Пестриков В.М., Маслобоев А.Н., Федоров О.К. Основы программирования в системе Borland Delphi: Учебное пособие / СПб ГТУ РП, СПб, 2004. 107 с.: ил. 32. ISBN 5-230-14386-X

В настоящем учебном пособии описываются основные конструкции языка Object Pascal и на конкретных примерах раскрываются приемы и методы программирования в интегрированной среде Borland Delphi. Предназначена для студентов всех специальностей.

Ил.32. Библиогр.: 12 назв.

Рецензенты:
заведующий кафедрой высшей математики и математических методов в экономике Санкт-Петербургской государственной академии сервиса и экономики, профессор С.И. Никитин

кандидат технических наук С.Г. Рыбаков (СКБ «Турбина»)

Рекомендовано к изданию Редакционно-издательским советом университета в качестве учебного пособия.

ISBN 5-230-14386-X ББК 32.97Я7

© Пестриков В.М. , Маслобоев А.Н.
Федоров О.К.,

©ГОУВПО Санкт-Петербургский государственный технологический университет растительных полимеров, 2004

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

Оглавление

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

Предисловие

Настоящее пособие посвящается основам программирования в среде ***Delphi***, разработанной компанией ***Borland.*** Данная среда является одной из ведущих систем программирования, используемых для разработки современных программных продуктов, и в первую очередь приложений операционной системы ***MS Windows***. Система ***Delphi*** базируется на использовании языка программирования ***Object Pascal***, который является логическим продолжением и развитием классического языка программирования Паскаль. Поэтому, говоря о системе ***Delphi***, нельзя не сказать несколько слов о языке Паскаль, его особенностях, об истории его возникновения и развития.

Созданный в начале 70-х годов XX века признанным классиком программирования Никлаусом Виртом, язык Паскаль был назван в честь французского ученого Блеза Паскаля (1623-1662). Великий ученый, которого современники называли французским Архимедом, вошел в историю не только как автор научных трудов, охватывающих самые различные области человеческого знания – от философии до математики, но и как изобретатель арифметической машины - первого в мире механического счетного устройства.

Изобретение Блеза Паскаля положило начало тому процессу, который, в конечном счете, привел к появлению современной вычислительной техники, ставшей одним из определяющих факторов научно-технического прогресса. Без компьютеров ныне вообще немыслимо нормальное существование и развитие цивилизованного общества. Исходя из вышесказанного ясно, что название языка было выбрано не случайно. Паскаль был задуман как образцовый язык, который должен определенным образом формировать мышление программистов, помочь им почувствовать законы программирования, его красоту.

Первоначально Паскаль разрабатывался, прежде всего, как язык, предназначенный для эффективного обучения программированию, и успешно справлялся с этой задачей. Например, в 1983 году в США Паскаль был объявлен официальным языком программирования для учащихся средних школ, которые намерены специализироваться в области вычислительной техники и программирования в американских университетах. Но с течением времени язык Паскаль вышел за чисто учебные рамки и стал равноправным и популярным языком программирования.

В 80-е годы прошлого века позиции Паскаля еще более упрочились в связи с появлением версий языка, предназначенных для персональных компьютеров. Язык стал использоваться не только как средство обучения студентов и школьников, но и широко стал применяться как рабочий

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

инструмент пользователей. Возникло целое семейство языков Паскаль, и ведущее место в этом семействе занял язык Турбо Паскаль, разработанный программистами американской фирмы ***Borland International***. На протяжении ряда лет (1983-1992) фирмой ***Borland*** был создан ряд новых, более совершенных версий языка, и итогом этой работы стало создание мощной системы программирования, включающей универсальную интегрированную среду, в которую «погружен» язык. Эта среда значительно упрощала и облегчала процесс создания программ, и в то же время предоставляла пользователю ряд новых, дополнительных возможностей.

Хотя система программирования Турбо Паскаль создавалась как приложение операционной системы ***MS DOS***, она успешно работает и на компьютерах, на которых установлена операционная система ***Windows*** и продолжает широко использоваться как в учебных целях, так и для решения практических задач. Однако с течением времени возникла острая потребность в разработке системы программирования, которая бы позволяла сочетать сохранение лучших традиций программирования на Паскале с использованием всех возможностей, предоставляемых наиболее популярной сегодня для персональных компьютеров операционной системой ***Windows***.

Поэтому в 90-е годы XX века фирма ***Borland*** завершила линию, связанную с программированием для операционной системы ***MS DOS*** и представленную Турбо Паскалем, и приступила к работе над системой программирования, ориентированной на создание приложений операционной системы ***MS Windows***. В настоящее время эту деятельность продолжает компания Inprise, в состав которой влилась фирма ***Borland***. Новая система программирования получила название ***Delphi.***

Данное название, подобно названию языка Паскаль, также не является случайным. Свое название система программирования ***Delphi*** получила в честь существовавшего в древней Греции города Дельфы, где находился знаменитый храм бога Аполлона. Аполлон почитался в древнем мире как небесный покровитель наук и искусств, а в храме, построенном в его честь, находился знаменитый дельфийский оракул, у которого люди стремились не только узнать о своей личной судьбе, но и получить ответы на самые сокровенные вопросы бытия. В настоящее время система ***Delphi*** представляет собой мощное орудие, которое современные ученые и инженеры используют для познания и преобразования окружающего мира.

Систему программирования ***Delphi*** подобно системе Турбо Паскаль часто называют интегрированной средой программирования. Слово «интегрированный» (от латинского ***integrare*** – восстанавливать, восполнять) означает в данном случае, что в системе объединены в одно целое различные средства, способствующие наиболее быстрой и эффективной разработке программы.

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

К этим средствам относится, во-первых, файловый менеджер, позволяющий не покидая среду программирования создавать новые программные файлы, сохранять их там, где это необходимо и когда необходимо, а также открывать уже существующие файлы. Во-вторых, это экранный редактор, позволяющий не только набирать и корректировать текст программы, но и в ряде случаев автоматизировать этот процесс и подсказывать программисту те служебные слова, которые можно использовать в данном контексте. В третьих, – это система отладки программы, которая в большинстве случаев не ограничивается указанием характера сделанной ошибки, указывая также строку, в которой эта ошибка была допущена. В четвертых, – это разветвленная справочная система, которая содержит не только сведения теоретического характера, но и конкретные примеры разработки приложений в среде ***Delphi***. Все вышеперечисленное далеко не исчерпывает все многообразие средств, способствующих созданию приложений в данной системе.

Характеризуя среду программирования ***Delphi***, о ней также говорят как о визуальной и событийно-ориентированной. Первое означает, что пользователь визуально, т. е. наглядно может увидеть в системе те заготовки, которые в дальнейшем будут использованы для создания экранных объектов в его программе, а затем сам сконструировать ее интерфейс (внешний вид) путем переноса этих заготовок на экранную форму. Второе же означает, что программист может выбрать из имеющегося в системе программирования списка те события, на которые должны реагировать экранные объекты и запрограммировать эту реакцию нужным ему образом.

Наконец, еще одним важным достоинством системы программирования ***Delphi*** является ее универсальность. Дело в том, что многие современные языки и соответствующие системы программирования созданы для решения узкоспециальных задач. Так, язык ***Cobol*** предназначен в первую очередь для создания программ в области экономики, язык ***Fortran*** – для инженерно-технических расчетов, языки ***Lisp*** и ***Prolog*** – для работы над системами искусственного интеллекта и т.д. Система же ***Delphi*** позволяет создавать профессиональные и эффективно работающие приложения, используемые в самых различных сферах человеческой деятельности. Поэтому время, затраченное будущим специалистом на изучение данной системы программирования, будет потрачено с пользой, вне зависимости от того, какую специализацию он изберет для себя в дальнейшем. Основное предназначение настоящего учебного пособия - это облегчить процесс освоения основ программирования в системе ***Borland Delphi*** и дать необходимые базовые знания в этой области, которые в дальнейшем специалист сможет пополнять путем самообразования.

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

1. Запуск системы программирования

Как и любой сложный программный комплекс, система программирования ***Delphi*** имеет головной файл, который запускает на выполнение всю систему. Поэтому процедура запуска системы по сути дела сводится к запуску на выполнение данного головного файла.

Так как система программирования ***Delphi*** является приложением операционной системы ***Windows***, то подобно большинству действий в данной операционной системе, запуск системы программирования можно производить несколькими разными способами. Рассмотрим эти способы (конечно подразумевается, что система ***Delphi*** была правильно установлена на Ваш компьютер).

1. Открыть папку, содержащую головной файл, имеющий название ***Delphi32.exe***. Для этого на компьютере следует найти и открыть папку ***Program Files*** (эта папка, как правило, находится в корневом каталоге диска ***C***), затем найти в этой папке вложенную папку ***Borland***, открыть ее и в ней отыскать вложенную папку ***Delphi***, а в той, в свою очередь, открыть папку ***bin***. Эта папка и будет содержать файл ***Delphi32.exe***. Запуск файла производится двойным щелчком на значке файла, если для данного компьютера установлен классический стиль работы с мышью, и одиночным щелчком на значке, если на компьютере используется стиль ***Web***.

2. Открыть главное меню операционной системы нажатием клавиши «Пуск». Найти в главном меню раздел «Программы», а в этом разделе отыскать подраздел ***Borland Delphi***, в котором есть пункт ***Delphi***. Щелчок на этом пункте приводит к запуску системы программирования.

3. Если на рабочий стол ***Windows*** выведен ярлык для файла ***Delphi32***, то систему программирования можно запустить, не открывая главное меню или какие-либо папки. Для этого достаточно два раза щелкнуть на этом ярлыке для обычного стиля работы с мышью или один раз щелкнуть в том случае, если для мыши используется стиль ***Web***.

Выполнив одну из вышеуказанных операций, мы увидим, что на экране компьютера на некоторое время появляется специальная заставка, которая содержит надпись ***Borland Delphi***, а также номер устанавливаемой версии ***Delphi***. После успешного завершения процедуры запуска на экране компьютера устанавливается основной экран системы программирования ***Delphi***. Этот экран содержит все элементы, необходимые для разработки графического интерфейса программы, написания кода программы, отладки программного кода и запуска программы на выполнение. Рассмотрим элементы экрана в следующем разделе.

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

2. Элементы экрана системы Delphi

Основной экран системы программирования ***Delphi*** включает следующие элементы (рис.1):

1. Строка заголовка.
2. Главное меню системы программирования.
3. Панели инструментов.
4. Список объектов.
5. Инспектор объектов.
6. Окно формы.
7. Окно кода.

В строке заголовка содержится номер установленной на данном компьютере версии системы ***Delphi*** и название открытого в данный момент проекта. Проектом в ***Delphi*** называется программа, находящаяся в стадии разработки. Вновь открываемый проект по умолчанию называется ***Project1***. Это имя можно увидеть в строке заголовка после загрузки системы программирования.

Под строкой заголовка находится строка главного меню системы. Эта строка содержит ряд разделов, каждый из которых представляет собой ниспадающее меню. Для того, чтобы открыть ниспадающее меню, достаточно щелкнуть мышью на названии соответствующего раздела. Каждое такое ниспадающее меню содержит ряд команд. Если справа от названия команды стоит многоточие, то при выполнении этой команды открывается диалоговое окно. Если название команды закрашено серым цветом, то эта команда в данный момент времени невыполнима. Названия же тех команд, которые можно в данный момент использовать, написаны символами черного цвета. Если справа от названия команды стоит треугольная стрелка, то это означает, что данная команда содержит еще одно вложенное меню, в котором можно выбрать один из предложенных вариантов.

Например, если сразу после того, как будет установлен основной экран, щелкнуть раздел меню ***File***, содержащий команды для работы с файлами в системе программирования, то в нем все команды будут написаны черным цветом, то есть доступны для выполнения. Справа от команды ***Save as*** (сохранить как) находится многоточие, означающее, что в ходе выполнения команды появится диалоговое окно, которое позволит сохранить файл в указанной пользователем папке. Справа же от команды ***New*** (создание нового элемента проекта) находится треугольная стрелка, которая показывает, что при выполнении этой команды откроется меню, содержащее перечень тех элементов (приложение, форма, программный модуль и т. д.), которые можно добавить в программу.

Если же открыть другой раздел меню ***Edit*** (он содержит команды, используемые для редактирования текста проекта), то первоначально в этом

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

разделе ряд команд будет закрашен серым цветом. Это команды, используемые для работы с выделенным фрагментом текста. Так как изначально не один фрагмент текста не выделен, то использование этих команд лишено смысла. Когда в процессе работы над программой будет выделен какой-либо фрагмент текста, то команды ***Cut*** (удалить выделенный фрагмент текста в буфер обмена), Copy (скопировать фрагмент в буфер обмена), ***Paste*** (вставить фрагмент в то место программы, на котором стоит курсор), ***Delete*** (удалить выделенный фрагмент) станут доступными и будут закрашены черным цветом.

Под строкой меню находятся панели инструментов, которые представляют собой набор экранных кнопок. На каждой кнопке имеется пиктограмма, т. е. небольшой рисунок, который поясняет ее назначение. Кроме того, если навести на кнопку указатель мыши и задержать его там на несколько секунд, то рядом с кнопкой появляется всплывающая подсказка, которая кратко поясняет назначение кнопки в словесной форме. В нерусифицированных версиях ***Delphi*** эта подсказка выводится на английском языке. Наиболее часто в ходе работы над проектом используются кнопки панели «Компоненты». Эта панель содержит большое количество вкладок, на каждой из которой находится ряд компонентов, представляющих собой заготовки для будущих экранных объектов. Переключение между вкладками производится щелчком на корешке соответствующей вкладки.

Ниже расположено окно формы. Форма представляет собой заготовку для окна будущей программы. Именно на форме в процессе работы над проектом размещаются различные объекты, переносимые с вкладок панели компонентов. Эти объекты в совокупности составляют пользовательский интерфейс, т. е. средство взаимодействия между пользователем и программой. На форме имеется специальная координатная сетка, которая используется для более точного расположения объектов. При запуске программы на выполнение координатная сетка исчезает, и фон формы становится однородным.

Слева от формы находится окно, содержащее список всех объектов, используемых во время работы над данным проектом. Данный список представлен в форме древовидной структуры наподобие дерева файлов в программе «Проводник», являющейся стандартным приложением ОС ***Windows***. Если какой-либо объект содержит другие, вложенные в него объекты, то слева от значка объекта имеется квадратик со знаком «плюс». Для того чтобы узнать, какие это вложенные объекты, достаточно щелкнуть мышью на квадратике. После этого соответствующие объекты будут выведены в окне, а «плюс» изменится на «минус». Для того чтобы «спрятать» вложенные объекты, достаточно снова щелкнуть на квадратике.

Если щелкнуть мышью на значке самого объекта, то он будет выделен синим цветом, а в расположенном ниже окне инспектора объектов можно будет ознакомиться с его свойствами. Окно инспектора объектов (рис.2) содержит две

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

вкладки. На одной из них, имеющей название ***Properties*** (по-английски – свойства), содержится перечень свойств, присущих данному объекту. Большинство свойств объекта можно изменять непосредственно в этом окне либо путем ввода соответствующего текста или числового значения, либо выбором нужного значения из раскрывающегося списка. На другой вкладке, имеющей название ***Events*** (по-английски – события), содержится перечень событий, на которые может реагировать данный объект. Событием считается, например щелчок мышью на объекте или нажатие какой-либо клавиши. Для того чтобы запрограммировать реакцию объекта на произошедшее событие, как правило, нужно написать соответствующий программный код, отображаемый в окне кода.

Окно кода обычно не видно целиком на экране системы, так как большая его часть закрыта окном формы, и пользователь видит только нижнюю строку этого окна. Для того чтобы активизировать окно кода (т. е. вывести его на передний план), нужно щелкнуть на любом видном пользователю его фрагменте, либо, если окно закрыто целиком, использовать команды из раздела меню ***View***. Данное окно содержит исходный текст программы (называемый также исходным кодом), написанный на языке ***Object Pascal***.

Элементы экрана системы ***Delphi*** расположены таким образом, что они не закрывают основные элементы управления ОС ***Windows***. Поэтому в ходе работы всегда можно воспользоваться главным меню кнопки «Пуск» или панелью задач, например для того, чтобы переключаться между ***Delphi*** и другими открытыми пользователем во время работы за компьютером приложениями, щелкая мышью соответствующие кнопки на панели задач.

3. Первая программа на Delphi

Как это ни странно, создание первой программы в системе ***Delphi*** начнем с того, что сохраним программу, хотя, на первый взгляд, кажется, что программы как таковой еще не существует. Однако, как только пользователь войдет в систему программирования, он может убедиться, что существует не только заготовки для экранных объектов в виде кнопок панели «Компоненты», но и заготовка самой программы. Для этого достаточно активизировать окно кода так, как это было сказано в предыдущем разделе. Когда это окно выйдет на передний план, Вы увидите, что в данном окне уже имеются определенные строки, содержащие текст программы. Эти строки автоматически генерируются системой программирования. В дальнейшем пользователь сам сможет по мере необходимости добавлять в окне строки, дополняющие эту основу. При этом

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

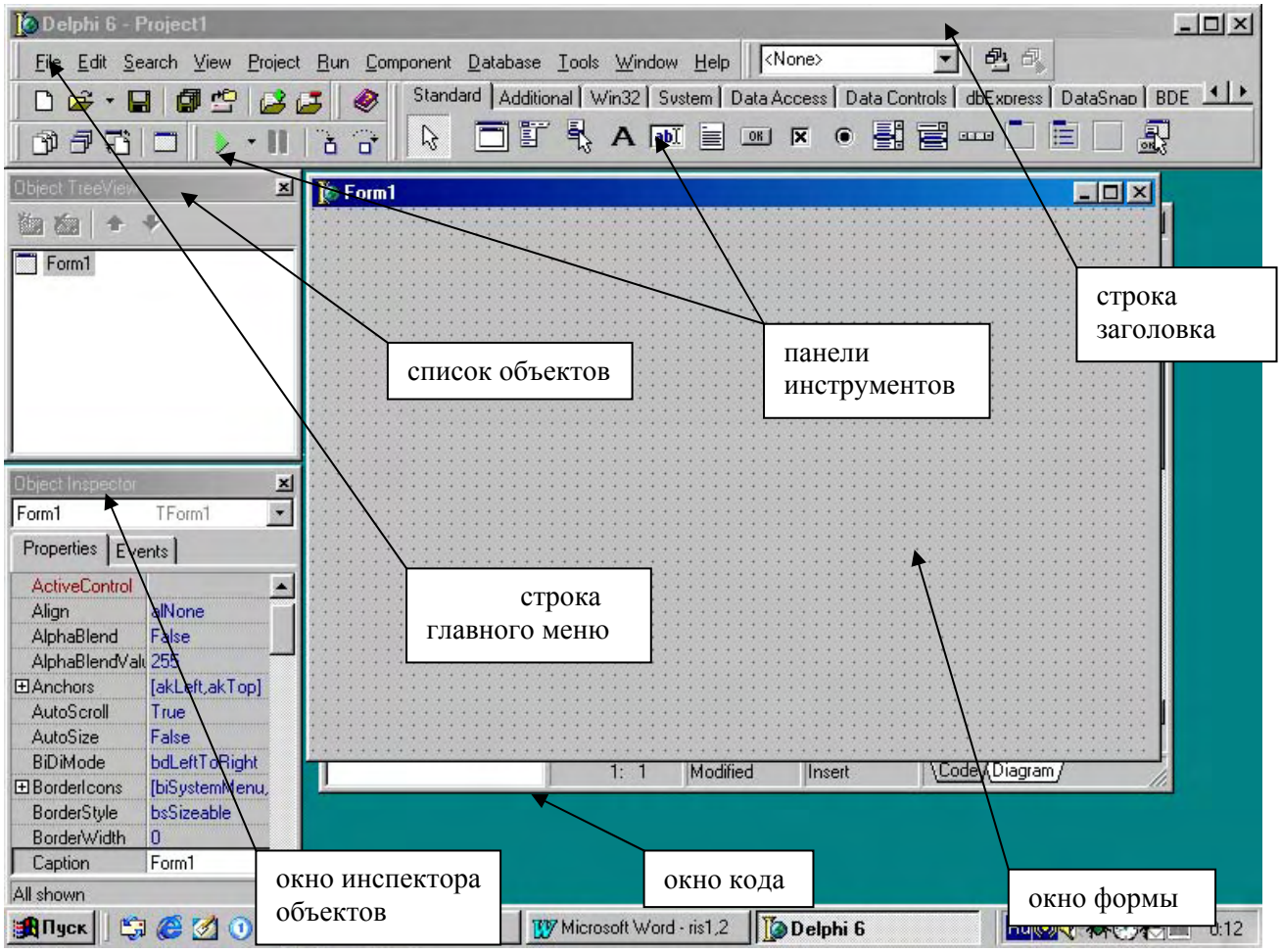

Рис. 1. Основной экран системы программирования ***Delphi***

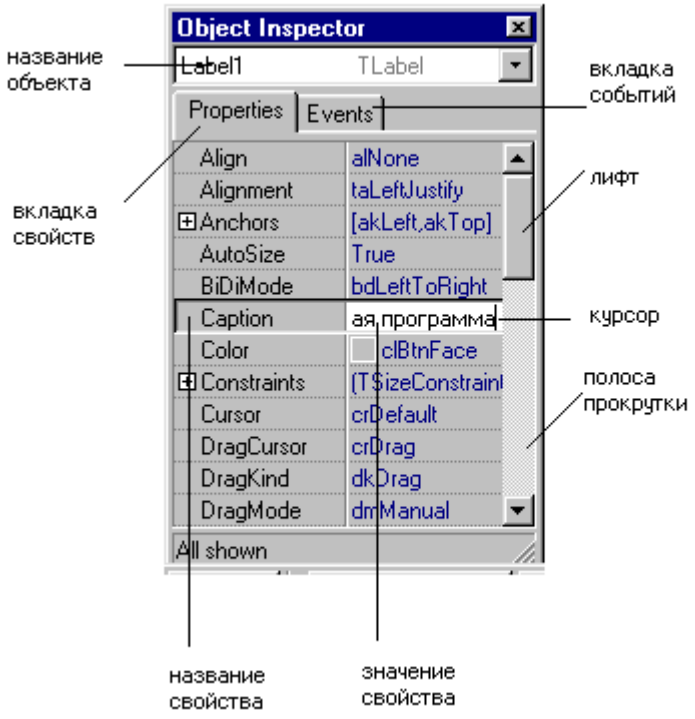

Рис. 2. Окно инспектора объектов

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

необходимо иметь в виду, что при разработке проекта система создает целую группу связанных между собой программных файлов.

Поэтому рекомендуется каждый вновь создаваемый пользователем проект сохранять в отдельной папке с тем, чтобы программист не путал файлы, относящиеся к разным проектам. В качестве места, где будут храниться все создаваемые в процессе программирования папки, лучше всего взять папку, уже имеющуюся в системе программирования. Эта папка так и называется ***Projects*** и расположена по следующему адресу (предполагается, что система программирования правильно установлени и расположена на диске ***C***): ***C\Program Files\Borland\Delphi***. В данной папке и будут создаваться вложенные папки, сохраняющие файлы разрабатываемых пользователем проектов. Рассмотрим конкретные шаги, которые необходимы для начального сохранения проекта.

Прежде всего, необходимо открыть раздел ***File*** главного меню и в нем найти команду ***Save as*** (сохранить как). При выполнении этой команды откроется диалоговое окно, которое называется ***Save Unit1 as*** (сохранение модуля 1). Внешний вид диалогового окна приведен на рис. 3. В данном окне будет отображаться содержимое папки ***Projects***. Для того чтобы создать новую папку внутри папки ***Projects,*** следует щелкнуть кнопку  на панели инструментов данного окна. Если пользователь не может сразу обнаружить эту кнопку, то следует поочередно подводить указатель мыши ко всем кнопкам, пока на одной из них не появится всплывающая подсказка «Создание новой папки». После щелчка по этой кнопке в окне появится папка, которая так и будет называться «Новая папка».

Понятно, что такое название неинформативно, так как ничего не говорит о содержании данной папки. Поэтому рекомендуется дать папке какое-либо осмысленное название. Например, папку с проектом Вашей первой программы можно назвать ***First*** (по-английски это слово и означает “первый”). Для того чтобы переименовать вновь созданную папку, следует нажать на клавиатуре клавишу ***Delete***, в результате чего исчезнет название «Новая папка» и ввести с клавиатуры подходящее название, а затем нажать клавишу ***Enter***. Затем нужно нажать экранную кнопку «Открыть», имеющуюся в диалоговом окне. При этом вновь созданная папка будет открыта и в окне будет видно, что папка пустая. Если щелкнуть на кнопке «Сохранить», то в этой папке будет сохранен файл ***Unit1***, содержащий исходный текст разрабатываемой Вами программы. Если пользователь допустил какую-либо ошибку (например открыл не ту папку), то можно закрыть окно без сохранения изменений, щелкнув кнопку «Отмена».

Следующим шагом должно стать сохранение файла ***Project1***, в котором будет содержаться откомпилированная и готовая к выполнению программа. Для этого следует вновь войти в раздел ***File*** главного меню системы и

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

использовать команду ***Save Project as***. В результате выполнения этой команды будет открыта вложенная папка (в нашем случае это папка ***First***) и для сохранения файла проекта останется только щелкнуть экранную кнопку «Сохранить». Окно сохранения проекта показано на рис.4. Остальные, связанные с проектом файлы, будут сохраняться автоматически. В дальнейшем изменения в проекте рекомендуется сохранять после добавления и настройки каждого нового объекта, но повторять каждый раз всю вышеизложенную последовательность действий нет необходимости. Достаточно дать команду ***Save all*** все из того же раздела меню ***File***, и все нужные изменения будут внесены в соответствующие файлы автоматически.

Теперь можно приступать к непосредственной разработке проекта. Создание программы включает в себя два взаимосвязанных компонента. Первым из них является создание графического интерфейса пользователя, который будет представлен в виде различных объектов и элементов управления, расположенных на экране программы. К ним относятся:

а) выводимые на экран компьютера надписи

б) экранные кнопки, щелкая мышью на которых, можно производить различные действия;

в) текстовые окна, используемые для ввода текста или чисел в ходе работы программы;

г) различные списки и меню;

д) группы флажков и переключателей, используемые для выбора одного из вариантов работы программы;

е) панели и рисунки, используемые для оформления экрана;

ж) многие другие элементы.

Вторым компонентом, который необходим для разработки приложения, является программный код, который описывает реакцию имеющихся на экране объектов и элементов управления на различные события, могущие происходитьв ходе выполнения программы. К числу событий, на которые должны реагировать объекты и элементы управления, относятся: щелчок мышью на объекте (при этом могут быть предусмотрены различные типы реакций для различных видов щелчка – одиночного, двойного, щелчка правой кнопкой мыши), перетаскивание объекта мышью, перемещение мыши, нажатие или отпускание какой-либо клавиши и другие. Реакция на событие описывается в виде соответствующей подпрограммы, входящей в основной программный модуль.

Подобно заготовкам для экранных объектов, текстовое окно, содержащее исходный код программы, также имеет заготовки для подпрограмм. В частности, по правилам языка ***Object Pascal***, каждая подпрограмма должна иметь заголовок, начинающийся со служебного слова ***procedure***, в начале основной части подпрограммы должно содержаться служебное слово ***begin***, а завершаться она должна служебным словом ***end***, после которого ставится точка

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

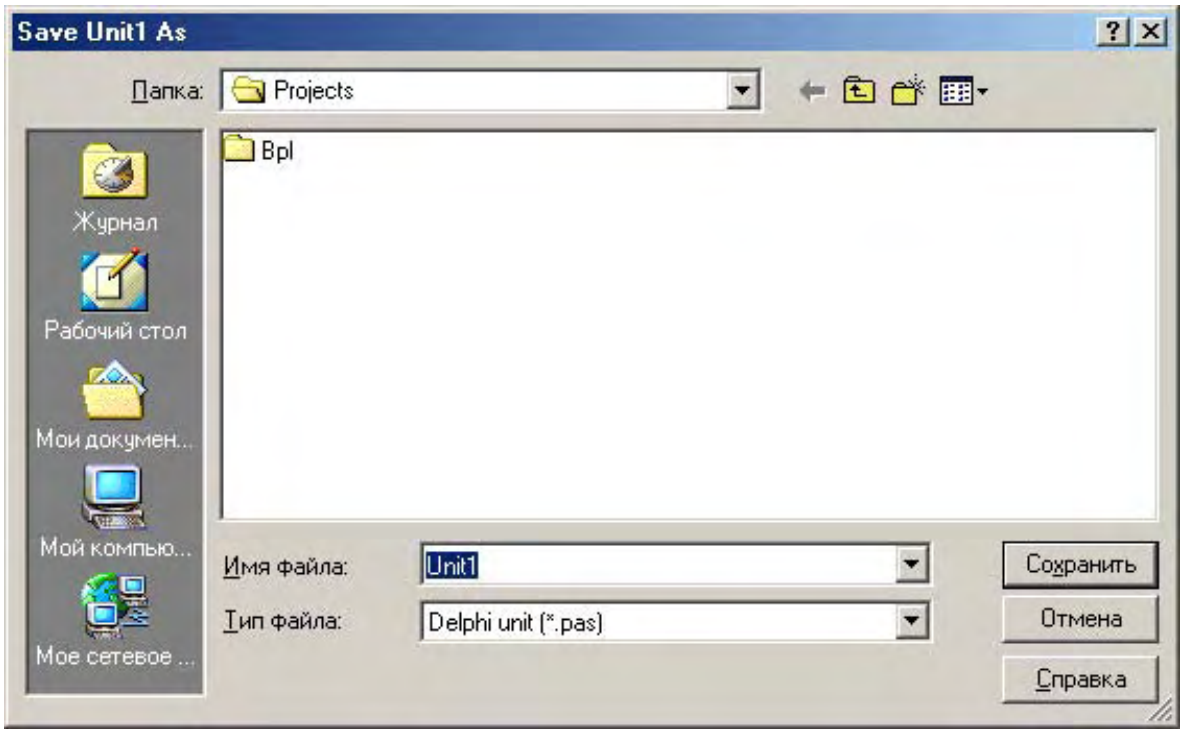

Рис. 3. Диалоговое окно сохранения файла исходного текста программы

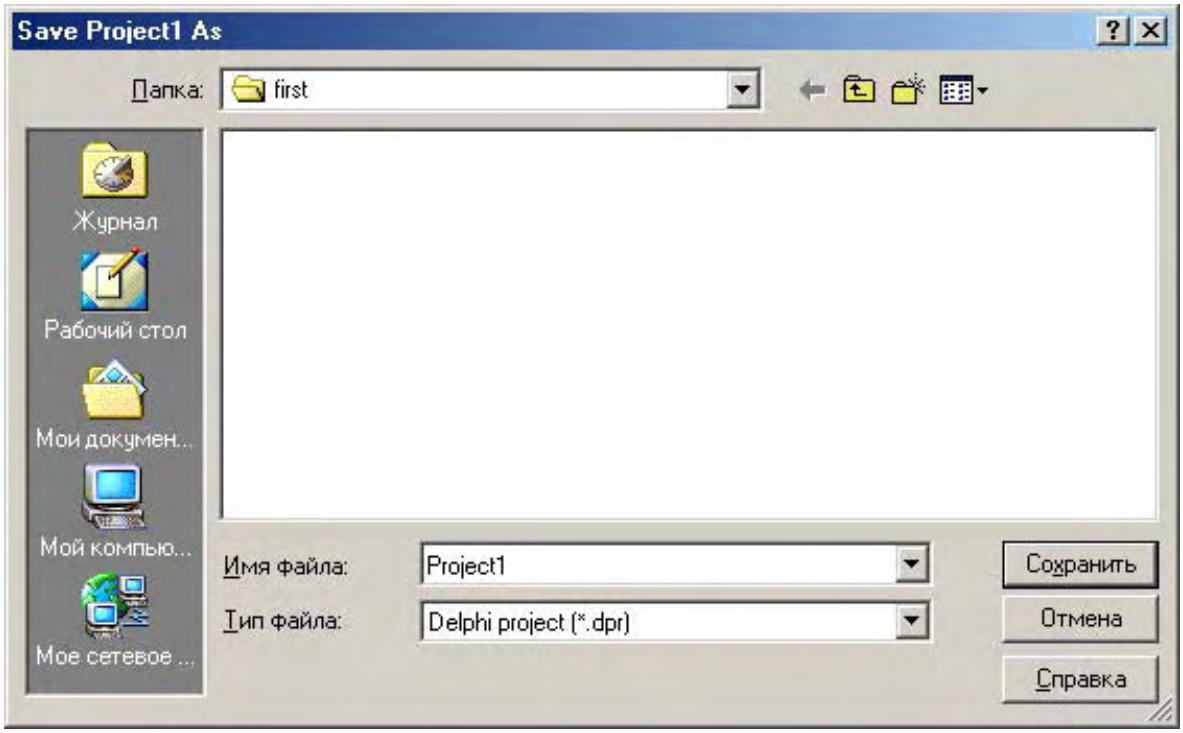

Рис. 4. Диалоговое окно сохранения файла проекта

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

с запятой. Все эти слова Вам не нужно будет вводить вручную, так как они уже имеются в автоматически создаваемой для них заготовке.

После того, как эти два компонента будут созданы, можно проверить работоспособность программы, запустив ее на выполнение. Перед запуском программы рекомендуется ее сохранить, используя для этого команду ***File→Save all.*** Если в программе обнаружатся ошибки, то текстовая строка, их содержащая, будет выделена другим цветом, что облегчит пользователю поиск ошибок и их исправление. После завершения «работы над ошибками» программист получит готовую к выполнению, работоспособную программу.

Правда при этом нужно иметь в виду, что автоматически система находит только грамматические ошибки, т. е. строки, написанные с нарушением правил языка Паскаль. Помимо них в программе могут содержаться и семантические, иначе говоря смысловые ошибки. Программа, содержащая такие ошибки, может быть внешне написана вполне корректно, но вместо сложения двух чисел вычитать их или при умножении 2 на 3 выдавать результат, равный 10. Поиск семантических ошибок составляет наиболее трудную часть работы программиста. При наличии необходимой доли упорства и терпения и эти трудности могут быть преодолены и конечным итогом работы станет программный продукт, который будет не только правильно и грамотно работать, но и будет обеспечивать для пользователя дружественный интерфейс (т. е. будет прост и удобен в использовании), а также будет удовлетворять эстетическим запросам потребителя.

Первая программа, которую мы создадим в системе ***Delphi***, будет производить следующие действия: выводить на экран компьютера надпись «Моя первая программа», при щелчке мышью на одной из экранных кнопок выводить на экран компьютера сведения об авторе программы и при щелчке на другой кнопке закрывать программу.

Создание графического интерфейса программы начнем с создания надписи. Перенос любого элемента графического интерфейса на форму можно производить двумя способами:

1. Дважды щелкнуть мышью на соответствующей кнопке на панели компонентов. В результате выбранный элемент интерфейса разместится посередине формы.

2. Один раз щелкнуть мышью на кнопке на панели компонентов и затем щелкнуть мышью в любом месте формы. В результате выбранный элемент интерфейса разместится в указанном месте. Этот способ обладает тем преимуществом, что программист может сам определять, где размещать выбранный элемент.

В данном случае элемент, который мы выбираем, находится на панели компонентов (она занимает всю правую часть отведенную на экране под панели

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

инструментов) на вкладке ***Standard***, которая расположена на данной панели первой слева. Этим элементом является надпись, которая по-английски называется ***Label***. На соответствующей кнопке имеется пиктограмма в виде большой буквы А -  .

Далее, нужно перенести надпись на форму любым из двух вышеописанных способов. В результате на форме Вы увидите надпись под названием ***Label1***. Затем пользователь должен отрегулировать положение надписи на форме (она должна находиться в верхней ее части) и ее размер. Содержанием данной надписи должна быть фраза «Моя первая программа». Как изменить содержание надписи, мы разберем чуть позже, а сейчас необходимо научиться регулировать положение и размеры надписи. Наиболее простой, хотя и не очень точный способ достижения поставленной задачи заключается в использовании мыши. Перемещать надпись по форме проще всего методом перетаскивания. Для этого нужно навести указатель мыши на центральную часть созданной надписи и нажать левую клавишу мыши. Затем, не отпуская клавишу, нужно переместить указатель мыши в нужное место формы. Вслед за указателем мыши переместится и надпись. Потом нужно отпустить клавишу мыши и надпись останется на выбранном пользователем месте.

Для изменения же размеров надписи используем следующий прием. Наведем указатель мыши на надпись и установим указатель на любую из границ надписи. Границы надписи отмечены специальными черными квадратиками, которые называются маркерами. На один из маркеров и нужно поместить указатель мыши. При этом указатель мыши изменит свою форму: он приобретет вид двунаправленной стрелки. Затем нужно нажать левую клавишу мыши и, не отпуская ее, переместить указатель мыши. Вслед за указателем мыши переместится и граница надписи. Подобный прием, часто используемый для изменения размеров экранных объектов, называется протягиванием. Размеры объекта (в данном случае объектом является надпись) можно изменять по трем направлениям – по горизонтали, по вертикали и по диагонали. В последнем случае размеры надписи будут изменяться одновременно по горизонтали и по вертикали. Для изменения вертикальных размеров нужно поместить указатель мыши на верхний или на нижний маркер, для изменения размеров по горизонтали указатель помещаем на один из боковых маркеров, а для изменения диагональных размеров используем любой из угловых маркеров.

В принципе, размеры надписи можно было бы и не изменять вручную, так как они впоследствии автоматически изменятся в соответствии с содержанием надписи, но надпись является хорошим объектом для тренировки. Поупражняйтесь с изменением размеров надписи и полученные навыки в дальнейшем пригодятся Вам при работе с другими объектами, размеры которых не изменяются автоматически в зависимости от их содержания.

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

Следующей задачей, которую необходимо решить при разработке проекта, является изменения содержания и шрифта надписи. Для этого нужно чтобы в окне инспектора объектов был выбран объект ***Label1***. Соответствующая надпись должна быть в текстовом поле, расположенном в верхней части окна инспектора объектов. В инспекторе объектов может отображаться и другой объект – сама форма, имеющая имя ***Form1***. Это возможно, например, в том случае, если пользователь, настраивая размеры надписи, случайно щелкнул мышью не на ней, а на любом свободном месте формы. Исправить такое положение очень просто – для этого достаточно один раз щелкнуть мышью на надписи, расположенной на форме.

После того, как Вы убедились, что в инспекторе объектов выделена именно надпись, выбираете в нем вкладку ***Properties***. На этой вкладке в алфавитном порядке перечислены все свойства обрабатываемого объекта. Каждое свойство в этом списке представлено в виде двух полей. В левом поле черным или коричневым цветом написано название свойства, в правом поле синим цветом указано значение, которое имеет в данный момент это свойство. Нас в данном случае интересует свойство ***Caption***, т. е. содержание надписи, называемое также ее заголовком. По умолчанию свойство ***Caption*** имеет значение ***Label1***, которое и отображается в поле, находящемся справа от названия (по умолчанию объект получает такой же заголовок, как и его имя).

Для того чтобы изменить значение этого свойства, нужно в начале удалить старое значение ***Label1***. Для этого нужно один раз щелкнуть мышью в правом поле. В результате в поле появится вертикальная черта, называемая курсором. Напомним, что для удаления символа, расположенного слева от курсора, нужно нажать на клавиатуре клавишу ***Backspace***, а для удаления символа, который находится справа от курсора, следует нажать клавишу ***Delete.*** После того, как правое поле очищено, вводим новое значение этого поля – фразу «Моя первая программа». В результате соответствующая фраза появится и в надписи, расположенной на форме.

При этом имя надписи ***Label1*** останется неизменным, так как имя надписи и ее заголовок – это разные свойства объекта. В этом нетрудно убедиться, увидев значение свойства ***Name*** (имя). Для того чтобы найти это свойство в инспекторе объектов, нужно использовать скроллер (полосу прокрутки), расположенную в правой части инспектора объектов. Прокручивать же содержимое инспектора объектов удобнее всего, перетаскивая мышью расположенный на полосе прокрутки серый прямоугольник, называемый лифтом. Найдя свойство ***Name***, убедимся, что справа от названия по-прежнему указано значение ***Label1***. На начальном этапе работы с системой имена объектов вообще лучше не изменять.

Взглянув на надпись, имеющуюся на форме, нетрудно убедиться, что выглядит она весьма неказисто. А ведь именно эта надпись будет основным элементом интерфейса нашей первой программы. Для того чтобы улучшить

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

внешний вид надписи следует использовать ее свойство, называемое ***Font*** (шрифт). Для того чтобы найти это свойство в окне инспектора объектов, мы используем уже знакомую нам полосу прокрутки с лифтом. Когда мы найдем это свойство, то увидим, что слева от названия свойства находится квадратик с плюсом. Это означает, что свойство является не простым, а составным, т. е. оно включает в себя ряд подсвойств.

Для того, чтобы ознакомиться со списком этих подсвойств, можно щелкнуть на квадратике, и ниже свойства ***Font*** в инспекторе объектов Вы увидите соответствующий список. Конечно, для того, чтобы изменить внешний вид шрифта, можно изменить значения ряда подсвойств, аналогично тому, как мы это делали со свойством ***Caption***. Но в данном случае мы воспользуемся более простым и наглядным способом настройки. Если щелкнуть один раз мышью в поле значений свойства ***Font***, то в этом поле появится кнопка, на которой изображено многоточие. Это многоточие имеет то же значение, что и соответствующий элемент в главном меню системы. Если щелкнуть на этой кнопке  , называемой также построителем, то откроется диалоговое окно, в котором можно изменять основные характеристики шрифта. Внешний вид данного окна показан на рис.5. Подобное окно хорошо знакомо пользователям, которые работали с программой ***Microsoft Word*** или другими приложениями ОС ***Windows***. Удобство работы с этим окном заключается еще и в том, что в отличие от большинства элементов интерфейса ***Delphi*** данное окно русифицировано.

В данном случае мы изменим следующие настройки. Во-первых, увеличим размер шрифта с 8 (по умолчанию) до 12. Во-вторых, изменим цвет шрифта – заменим используемый по умолчанию черный цвет на выбранный из раскрывающегося списка темно-синий. Помимо указанных настроек при желании можно изменить гарнитуру шрифта. Например, вместо используемого по умолчанию шрифта ***MS Sans Serif*** можно поставить шрифт ***Arial, Times New Roman*** или любой другой из числа установленных на Вашем компьютере. Кроме того, можно изменить начертание шрифта, сделав его наклонным (курсив) или полужирным. Наконец, в случае необходимости, щелкнув мышью соответствующий флажок, можно сделать текст подчеркнутым. В центре диалогового окна находится окошко, называемое «Образец», в котором будет отображаться внешний вид текста надписи и отражаться все сделанные пользователем изменения. В случае, если пользователь удовлетворен результатами своей работы, он может сохранить произведенные настройки, щелкнув на экранной кнопке «***Ok***». После закрытия диалогового окна все сделанные пользователем настройки вступят в силу, и можно будет увидеть, как преобразовалась надпись.

Следующим элементом интерфейса, который выведем на форму, будет закрывающая кнопка. Строго говоря, данный элемент не является для

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

программы обязательным, так как созданную нами программу, как и любое приложение ***Windows***, можно закрыть с помощью закрывающей кнопки  , которая находится в правом углу строки заголовка. Однако, для удобства пользователя принято, чтобы в программе имелся и другой способ ее закрытия (обычно с помощью экранной кнопки или пункта меню). Для того чтобы поместить на форму экранную кнопку, следует вновь воспользоваться вкладкой ***Standard*** панели компонентов. Искомый элемент имеет английское название ***Button*** и выглядит следующим образом -  . После выбора заготовки данного объекта разместим его в правой нижней части формы. По умолчанию на кнопке будет выведен заголовок ***Button1***.

Желательно, чтобы заголовок кнопки соответствовал производимому ею действию. Сделать это можно, как и в случае с объектом ***Label1,*** изменив значение свойства ***Caption*** (такое свойство есть и у экранных кнопок). Снова вводим текст надписи. В данном случае текст гласит – «Закрыть программу». Если внешний вид текста пользователя не удовлетворяет, то, как и в случае с надписью, можно отрегулировать свойство ***Font***, которое имеет точно такую же внутреннюю структуру, что и у надписи. Единственное отличие в настройке свойств этого объекта заключается в том, что кнопка не «растягивается» автоматически в зависимости от величины заголовка. Поэтому размеры экранной кнопки следует отрегулировать путем перетаскивания маркеров. Теперь экранная кнопка приобрела желаемый внешний вид, а полученный в результате вышеуказанных действий интерфейс программы приведен на рис.6. Следует отметить, что на рис.6 помимо уже описанных элементов интерфейса имеется также еще одна экранная кнопка и дополнительная надпись под кнопками, но их мы добавим позже в ходе усовершенствования программы.

Если на данном этапе работы над программой запустить программу на выполнение, то сколько мы ни будем щелкать мышью на кнопке или нажимать клавишу “***Enter***”, никакой реакции на наши действия не будет. Для того чтобы такую реакцию запрограммировать, нужно написать соответствующий программный код. Сам по себе код, закрывающий программу, крайне прост. Он состоит из одного-единственного слова “***Close***”, которое в переводе с английского и означает «закрыть». Вопрос состоит в том, что нужно знать в каком месте программы его нужно вставить. Для того чтобы облегчить себе работу, воспользуемся уже знакомым нам окном инспектора объектов.

Выделим в окне формы объект ***Button1*** и в инспекторе объектов перейдем на вкладку ***Events***, что, как читатель, конечно, помнит, означает «события». В качестве события, на которое должна реагировать наша кнопка, выберем щелчок мышью (по-английски – “***click***”). Если мы обратимся к инспектору объектов, то увидим, что там этому событию соответствует пункт ***Onclick***. Если щелкнуть два раза мышью на поле, находящемся справа от пункта ***Onclick***, то перед нами откроется окно кода с уже имеющейся заготовкой процедуры, описывающей это событие. Между служебными

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

словами процедуры ***begin*** и ***end*** вставляем требуемое слово "***close***". Внешний вид окна кода программы приведен на рис.6.

В данном окне должен быть виден следующий текст процедуры ***Button1.Click***, описывающей реакцию на щелчок мышью на кнопке «Закрыть программу»:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
close
end;
```

Как видно из вышеприведенного текста, заголовок процедуры и служебные слова система программирования вставляет автоматически. Для того чтобы убедиться, что созданная программа работоспособна, можно произвести запуск программы на выполнение. Запуск программы можно произвести тремя способами.

1. Через главное меню системы программирования. В меню нужно найти раздел ***Run.*** Открыв этот раздел, найти в нем пункт, который тоже называется ***Run***, и щелкнуть его мышью.

2. С помощью клавиатуры. Нажать на клавиатуре компьютера клавишу ***F9***.

3. С помощью панели инструментов системы программирования. Найти кнопку запуска программы в виде треугольной стрелки -  и щелкнуть ее мышью.

После запуска программы на выполнение любым из вышеуказанных способов произойдет следующее: на переднем плане экрана появится окно созданной Вами программы. О том, что это именно окно работающей программы, а не заготовка программы в виде формы, можно догадаться по отстутствию координатной сетки. Эту программу можно закрыть теперь двумя способами: либо через закрывающую кнопку в строке заголовка, либо посредством созданной нами кнопки в нижней части окна. После закрытия программы мы вновь окажемся в системе программирования ***Delphi***.

Несмотря на то, что программа уже работоспособна, работа над программой еще не закончена. Интерфейс программы должен включать в себя еще один элемент – экранную кнопку, при щелчке на которой на экран будут выводиться сведения о создателе программы. Такие сведения о себе сообщает любой уважающий себя программист. Вторая экранная кнопка создается и настраивается точно так же, как и первая, только называться она будет не Button1, а Button2. Содержимое же заголовка, помещаемого на этой кнопке, будет «Сведения об авторе». Разместим мы эту кнопку в левом нижнем углу формы. На этом создание графического интерфейса данной программы завершено. Получившийся в результате описанных действий интерфейс показан на рис.7.

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

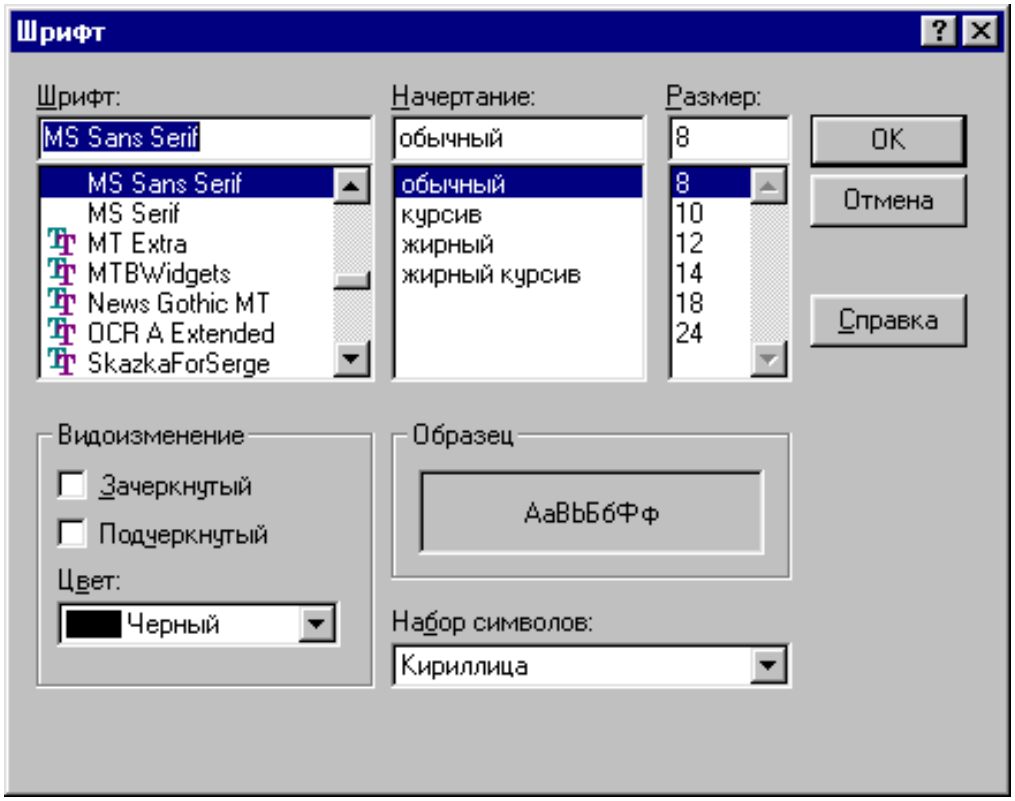

Рис. 5. Диалоговое окно настройки шрифта

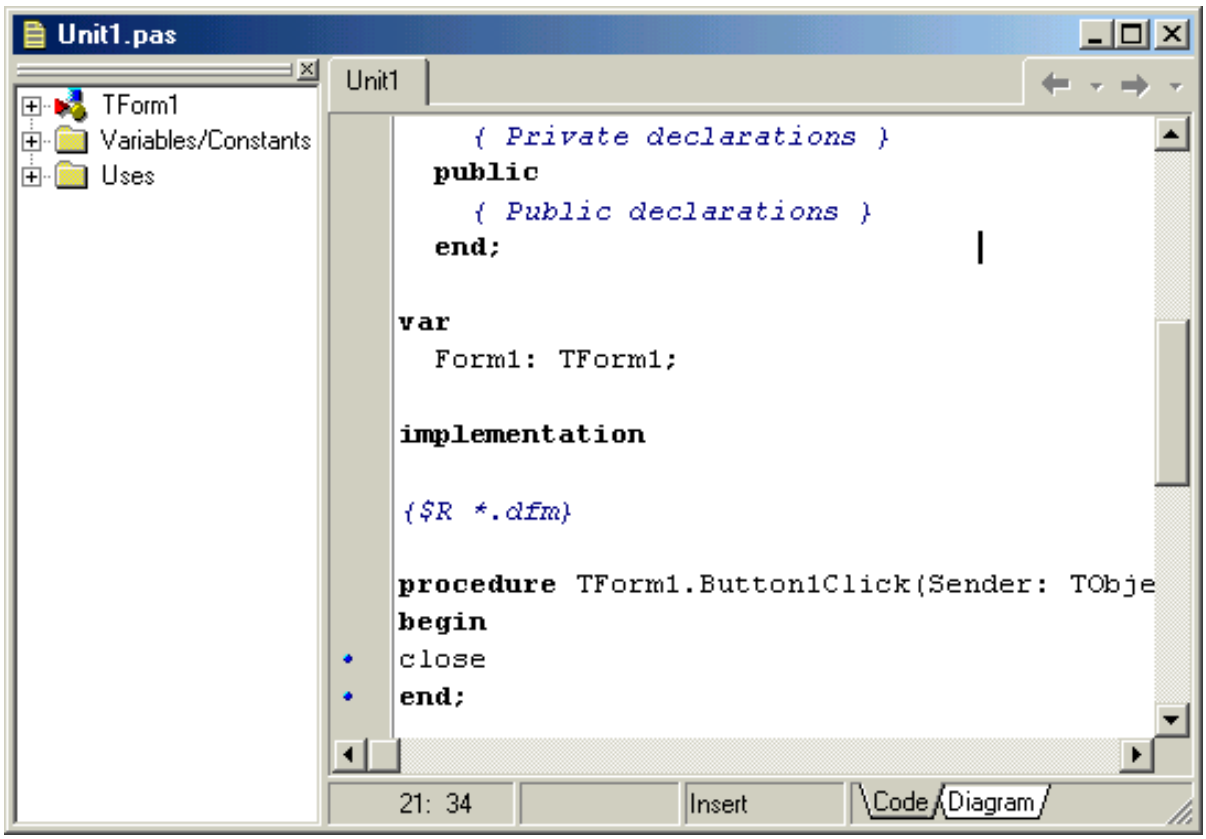

Рис. 6. Окно кода программы на ***Delphi***

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

Далее мы должны составить программный код, описывающий действие, происходящее при щелчке мышью на кнопке ***Button2***. Действие будет заключаться в следующем. При щелчке на кнопке «Сведения об авторе» на экране вместо слов «Моя первая программа» мы увидим надпись «Программу составил …» (вместо многоточия составитель программы может поставить свое имя и фамилию). Для этого при щелчке на экранной кнопке должно изменяться свойство надписи ***Caption***. Когда мы в программе ссылаемся на свойства какого-либо объекта, то эта ссылка должна состоять из двух частей: имени объекта и названия свойства. Между этими двумя частями ставится точка. Поэтому заголовок нашей надписи будет описываться как ***Label1.Caption***. Изменение свойства осуществляется путем присваивания ему нового значения. Для этого используется оператор присваивания языка Паскаль, который состоит из двоеточия и знака равенства. Если присваиваемое значение является текстом, как в нашем случае, то оно должно быть обязательно заключено в кавычки. Таким образом, код, изменяющий содержание надписи, будет выглядеть так, как показано ниже:

Label1.Caption:='Программу составил …'

Для того чтобы вставить эту строчку в нужное место программы, следует выделить объект ***Button2***, перейти в нем на вкладку ***Events*** и дважды щелкнуть поле справа от пункта ***Onclick***. В открывшуюся заготовку процедуры ***Button2.Click*** и следует ввести приведенную строку. Обратим внимание на следующее обстоятельство – как только мы вводим слово ***Label1*** и ставим после него точку, появляется список слов, которые можно ввести после этой точки. Для того чтобы выбрать какое-либо слово из списка, достаточно два раза щелкнуть его мышью. В результате данное слово появится в текстовой строке. Эта возможность автоматизации ввода текста особенно удобна в том случае, когда пользователь сомневается в правильности написания того или иного термина. Если воспользоваться указанной возможностью, то правильное с орфографической точки зрения слово подставит в текст сама система программирования. Ниже приводится полный текст программы, который должен находиться в окне кода.

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms, Dialogs, StdCtrls;

type
    TForm1 = class(TForm)
  Label1: TLabel;
```

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

```
    Button1: TButton;
    Button2: TButton;
      procedure Button1Click(Sender: TObject);
      procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
begin
close
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
Label1.Caption:='программу составил А.Н. Маслобоев'
end;

end.
```

После произведенного усовершенствования программы снова запускаем ее на выполнение одним из трех указанных способов. Напомним, что перед запуском программы на выполнение, рекомендуется сохранить все сделанные настройки и дополнения с помощью команды меню ***Save all***. Внешний вид программы во время ее работы приведен на рис.8. Убедившись в том, что программа работает так как положено, можно закрыть систему программирования. Выход из системы производится либо щелчком на закрывающей кнопке в строке заголовка системы, либо с помощью главного меню. В последнем случае нужно найти раздел меню ***File*** и в нем команду ***Exit***. Если перед выходом из системы Вы забыли сохранить изменения в проекте, то на экран компьютера автоматически выводится диалоговое окно, в котором пользователю предлагается либо сохранить изменения в проекте, либо отказаться от изменений. Если же пользователь нажимает в диалоговом окне экранную кнопку «***Cancel***», то выхода не происходит, и пользователь возвращается в систему программирования.

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

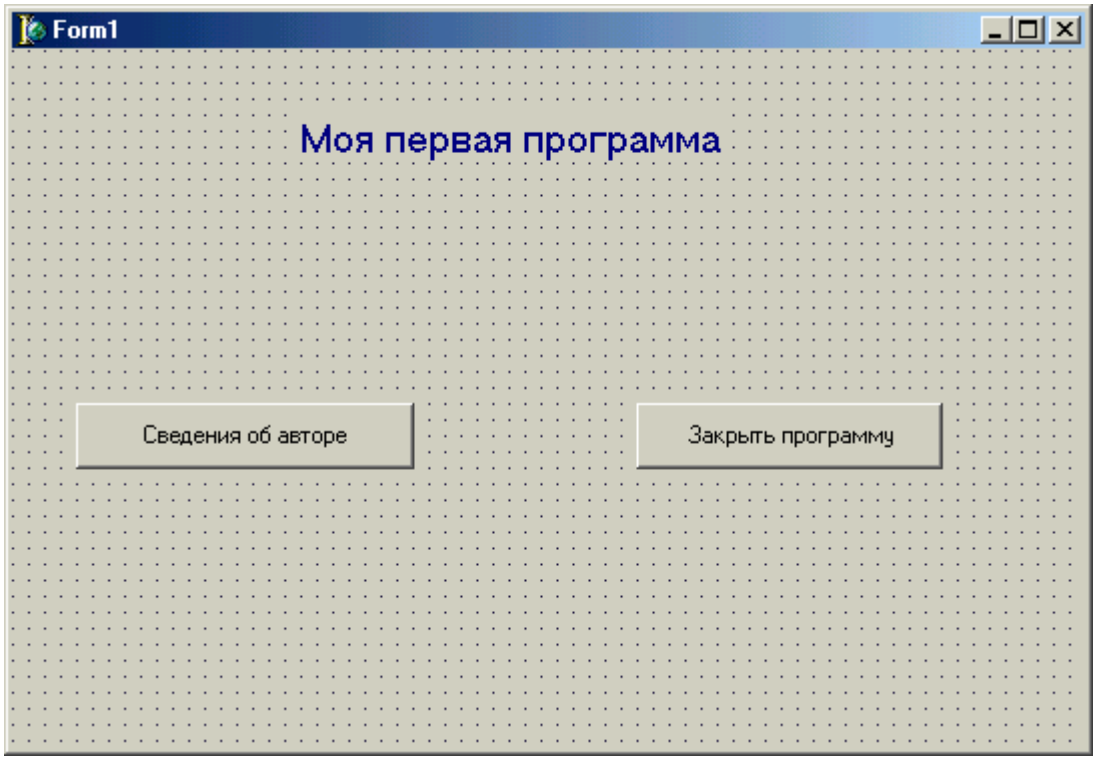

Рис. 7. Интерфейс первой программы

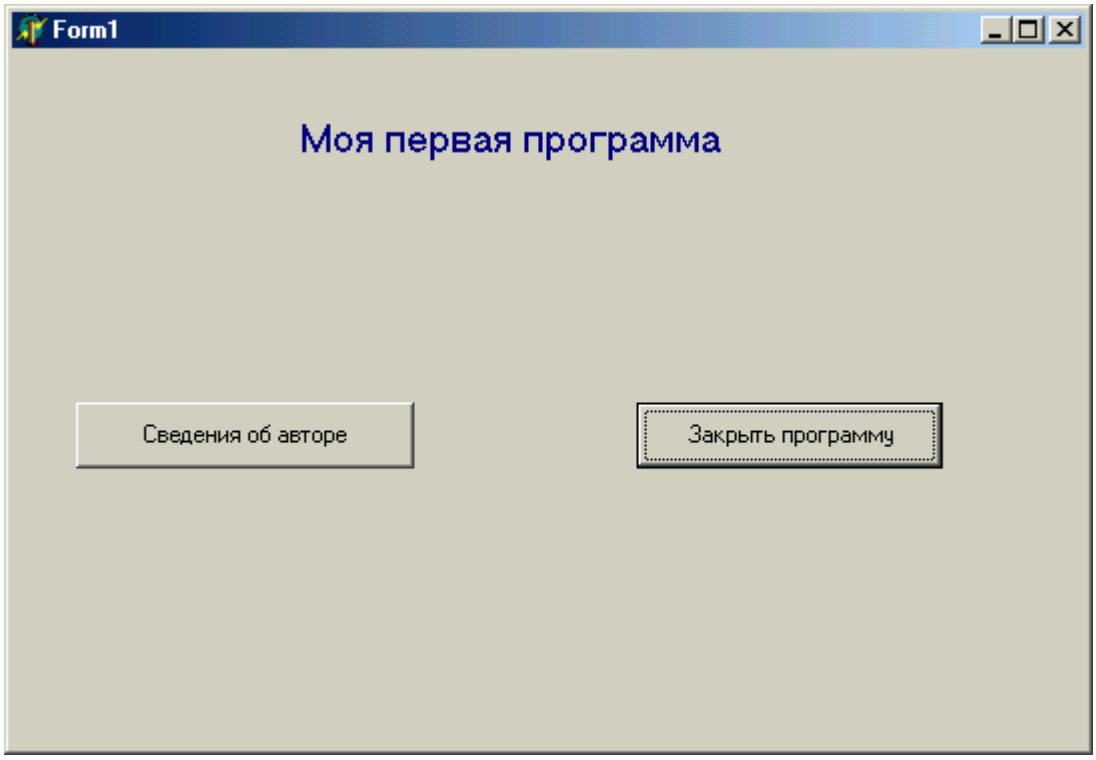

Рис. 8. Первая программа во время работы

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

Если проект был корректно сохранен, то созданный системой исполняемый файл будет работать и автономно от системы. Для того, чтобы убедиться в этом, достаточно открыть папку проекта, не запуская систему ***Delphi***, найти в ней исполняемый файл (в шестой версии ***Delphi*** значок этого файла выглядит так - ), и запустить его на выполнение двойным щелчком мыши. В результате программа запустится и будет работать должным образом.

4. Программа «Редактор»

Следующий проект, который мы будем разрабатывать в системе ***Delphi***, отличается от предыдущего тем, что пользователь получит возможность самостоятельно изменять содержимое надписи, расположенной в окне программы. Для этого в окне формы должно быть предусмотрено специальное текстовое окно, в котором пользователь вводит любой текст. Соответствующий текст будет отображаться в виде надписи, расположенной в верхней части окна программы.

Работы над проектом, как и в прошлый раз, начнем с его сохранения. Для этого в папке ***C:\Program Files\Borland\Delphi\Projects*** создадим вложенную папку ***redaktor*** и сохраним файлы будущего проекта аналогично тому, как мы это делали в предыдущей главе.

Как обычно, составление самой программы мы начнем с создания ее интерфейса. Интерфейс данной программы будет включать в себя следующие элементы:

1. Две надписи. Одна из них – это основная надпись, значение которой можно изменять путем ввода нового текста в текстовое окно. Вторая надпись вспомогательного характера содержит пояснение к текстовому окну.

2. Текстовое окно. Введенный в него текст будет дублироваться в укрупненном виде в надписи.

3. Две экранные кнопки. Одна из них закрывает программу. Вторая выводит сведения об авторе.

4. Сама форма, на которую выводят все перечисленные компоненты.

Настройку элементов интерфейса начинаем с самой формы. Форма является таким же объектом программы, как и все прочие ее компоненты и имеет ряд свойств, которые можно изменять при ее настройке. Для того чтобы начать работу с формой достаточно щелкнуть мышью на любом ее свободном месте. При этом в окне инспектора объектов будет отображаться объект по имени ***Form1***, т. е . сама основная форма. В окне инспектора выделяем вкладку ***Properties*** и находим на этой вкладке свойство ***Caption***. В данном случае это свойство отображает ту надпись, которая выводится в строке заголовка формы. Вводим вместо имеющегося в строке ***Caption*** по умолчанию

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

слова ***Form1*** слово “Редактор”. Это же слово появится и в строке заголовка формы, выделенной синим цветом.

Следующим шагом станет изменение цвета формы. При создании формы по умолчанию она получает серый цвет. Для того чтобы сделать фон формы более ярким и красочным, нужно на вкладке ***Properties*** объекта ***Form1*** найти свойство ***Color***. Это свойство содержит раскрывающийся список, содержащий названия цветов, которые можно выбрать в качестве фона для формы. Слева от названия каждого цвета в небольшом прямоугольнике показано как выглядит этот цвет. Для нашей формы выбираем цвет ***ClSkyBlue*** (небесно-голубой цвет). Приставка ***Cl***, которую включает в себя название каждого из представленных в палитре цветов, представляет собой сокращение от английского слова ***Color*** (цвет). На этом работа с самой формой завершается.

После этого создаем надпись, которую в дальнейшем можно будет редактировать, заменяя ее своим текстом. Для создания надписи используем компонент ***Label***, а соответствующий объект получит название ***Label1***. Чтобы по содержанию надписи можно было сразу понять ее назначение, свойству ***Caption*** присваиваем значение: «Здесь отображается текст, вводимый пользователем». Затем переходим к свойству ***Font*** того же объекта и щелкаем кнопку построителя. В открывшемся диалоговом окне выбираем какой-либо цвет, контрастирующий с фоном формы (например, – ярко-красный) и увеличиваем размер шрифта до 14 кегля.

Далее с помощью панели компонентов выводим на форму еще один объект - надпись. Этот объект получит от системы наименование ***Label2***. Расположим данный объект ниже первого. Он должен находиться примерно в середине формы по вертикали и ближе к левому ее краю по горизонтали. Содержанием данной надписи будет сообщение о том, что в окно, находящееся рядом с этой надписью, нужно вводить текст, который в укрупненном виде будет отображаться выше. Поэтому свойству ***Caption*** данной надписи присваиваем значение «Вводите текст в это окно». Цвет данной надписи изменять не будем (он останется по умолчанию черным), а размер ее сделаем несколько меньшим, чем у основной надписи ***Label1*** (например 12 кегль).

Справа от надписи ***Label2*** расположим то самое текстовое окно, в которое пользователь сможет вводить текст произвольного содержания. Для создания окна используется компонент ***Edit***, который находится на той же стандартной вкладке панели компонентов, что и компонент ***Label*** (не путайте компонент ***Edit*** с разделом меню, имеющим то же название). Выглядит компонент ***Edit*** следующим образом -  . Созданный с помощью этот компонента объект получает имя ***Edit1.*** Методом протягивания расширяем данный объект почти до правого края формы. Данный объект не имеет свойства ***Caption***, но имеет сходное с ним свойство ***Text***. Сразу после создания объекта данное свойство имеет значение ***Edit1***, т. е. это значение совпадает с именем объекта. Мы

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

значение ***Edit1*** удалим, а нового значения вводить не будем. Таким образом, после запуска программы на выполнение текстовое окно останется пустым и будет свободно для ввода текста. Подобно надписи, объект «текстовое окно» имеет свойство ***Font*** с таким же построителем, открывающим диалоговое окно настройки шрифта. Цвет шрифта в данном случае изменять не будем, а размер его сделаем равным 10.

Когда в ходе работы программы будем вводить в данное окно текст, в окне будет виден курсор, имеющий форму вертикальной черты и отмечающий текущую позицию ввода текста. Текст, который будет вводиться в данное окно, можно по ходу дела править. Напомним, что для удаления отдельных символов используются клавиши ***Delete*** и ***Backspace***. Если требуется удалить сразу целый фрагмент введенного текста, то можно выделить этот фрагмент клавишей «стрелка влево» или «стрелка вправо» при нажатой клавише ***Shift*** и затем нажать клавишу ***Delete***. Для быстрого перехода в начало или конец вводимого текста достаточно нажать соответственно клавиши ***Home*** или ***End***. Все описанные действия не надо специальным образом программировать. Эту задачу берет на себя сама система ***Delphi***.

Для завершения создания интерфейса программы осталось разместить на форме еще два объекта – командные кнопки, которые будут управлять работой приложения. Для создания этих кнопок используем компонент ***Button.*** Первый из этих объектов получит по умолчанию имя ***Button1***. Мы разместим его в левой нижней части формы. Щелчок на этой кнопке будет приводить к тому, что введенная пользователем в текстовом окне фраза будет продублирована в виде надписи вверху формы. Изменим для этой кнопки свойство ***Caption.*** Вместо имеющегося по умолчанию заголовка ***Button1*** введем следующий: «Изменить текст надписи». Воспользуемся свойством ***Font*** объекта ***Button1*** для того, чтобы увеличить шрифт надписи на кнопке до 10 пунктов. Кроме того, нужно увеличить методом протягивания размер объекта таким образом, чтобы надпись была целиком видна на экранной кнопке. Если этого не сделать, то при выполнении программы от надписи на кнопке будет виден только «хвост», а начало ее будет скрыто от пользователя.

После этого создаем второй объект-кнопку, которая по умолчанию получит имя ***Button2***. Эту кнопку разместим в правой нижней части формы и свойству ***Caption*** присвоим значение «Закрыть программу». Увеличим шрифт на кнопке до 10 пунктов, а также увеличим размеры кнопки для нормального отображения на ней надписи. Интерфейс программы приведен на рис.9.

Завершающий этап составления программы заключается в написании кода, описывающего реакцию экранных кнопок при щелчке мышью по ним. Для объекта ***Button2*** выбираем на вкладке ***Events*** событие ***OnClick*** и пишем для него код, состоящий из слова ***Close.*** Эта кнопка будет закрывать программу подобно кнопке «Выход» из предыдущей программы. Вид созданной

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

процедуры также аналогичен соответствующей процедуре в предыдущем проекте.

Для того чтобы разобраться в том, какой код нужно писать для кнопки ***Button1***, нужно прежде всего понять, что это кнопка должна делать. Как мы уже знаем, эта кнопка должна изменять надпись, то есть изменять ее свойство ***Caption***. Изменение это производится путем присвоения свойству ***Label1.Caption*** нового значения, отличного от предыдущего. Как уже известно по предыдущей программе, операция присваивания состоит из знака равенства и двоеточия. Само же это новое значение содержится в объекте ***Edit1***, а точнее в его свойстве ***Text***. Поэтому запишем это свойство так: ***Edit1.Text***. Следовательно операция изменения заголовка надписи будет выглядеть следующим образом:

```
Label1.Caption:=Edit1.Text
```

Для того чтобы вставить эту строку в подпрограмму, описывающую реакцию кнопки ***Button1*** нужно выделить эту кнопку на форме, выделить в инспекторе объектов вкладку ***Events*** и найти событие ***Onclick***, для которого и указать этот код. В результате указанных действий процедура ***Button1.Click*** будет выглядеть следующим образом:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
Label1.Caption:=Edit1.Text
end;
```

Теперь программа готова к запуску ее на выполнение. Ниже приводится полный текст данной программы:

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls;

type
 TForm1 = class(TForm)
  Label1: TLabel;
 Label2: TLabel;
 Edit1: TEdit;
```

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

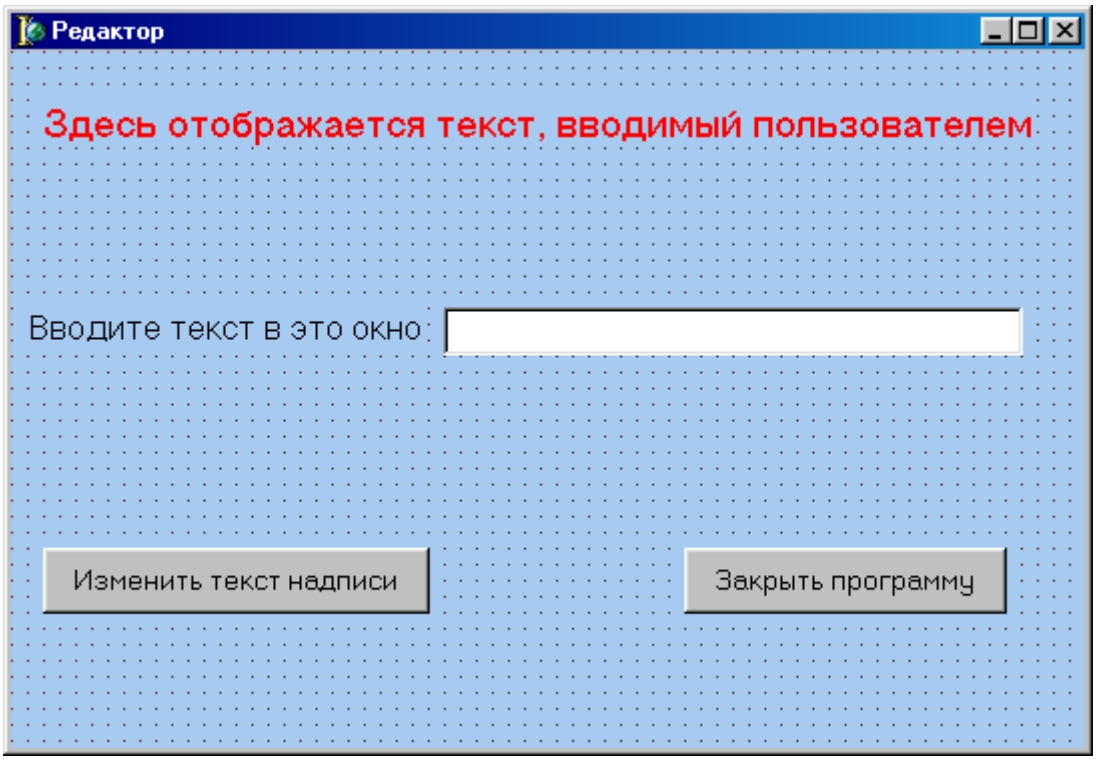

Рис. 9. Интерфейс программы «редактор»

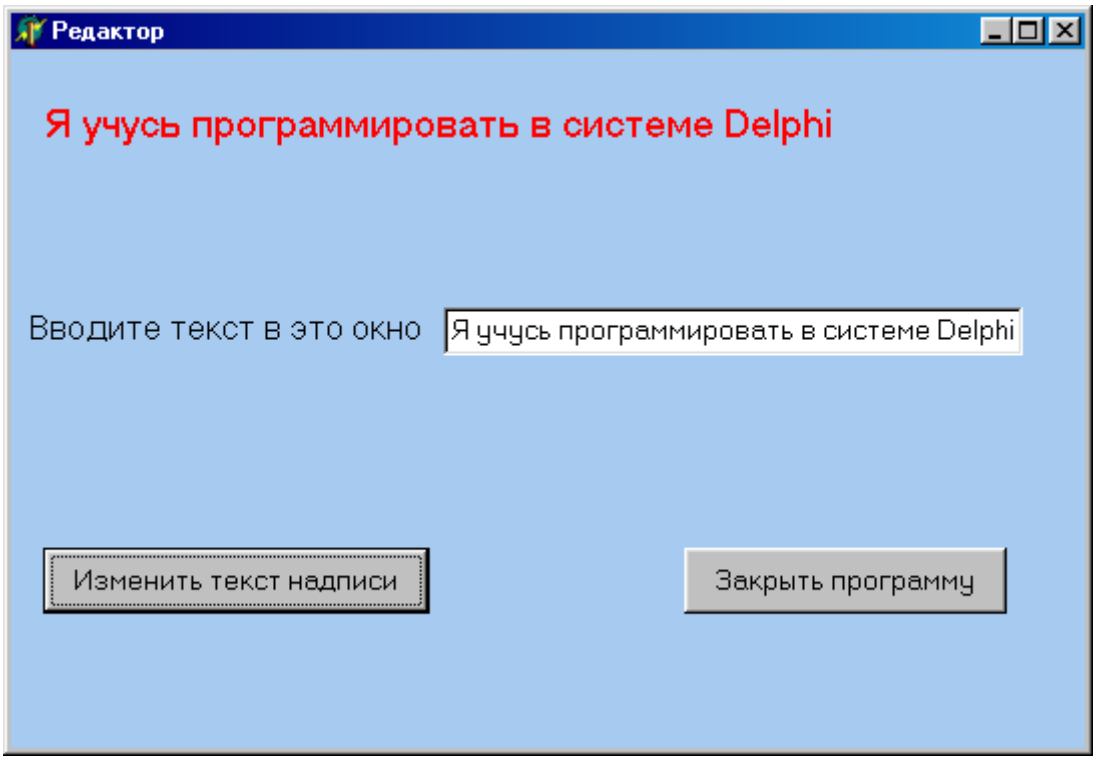

Рис. 10. Программа «редактор» во время работы

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

```
    Button1: TButton;
    Button2: TButton;
    procedure Button2Click(Sender: TObject);
    procedure Button1Click(Sender: TObject);
private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
begin
Label1.Caption:=Edit1.Text
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
close
end;

end.
```

После запуска программы увидим в окне программы чистое окошко для ввода текста с мигающим в левой его части курсором. В это текстовое окно пользователь может вводить любой понравившийся ему текст и корректировать его указанным выше способом, если в том есть необходимость. После щелчка на экранной кнопке «Изменить текст надписи» курсор в текстовом окне исчезнет, и появляется введенный текст, написанный вверху формы большими красными буквами (рис. 10).

В течение одного сеанса работы с программой содержание надписи можно изменять неоднократно. Для ввода нового текста достаточно щелкнуть мышью текстовое окно, в результате чего в нем вновь появится курсор. Тогда можно вводить в окно новый текст, который отобразится в верхней части формы после щелчка на экранной кнопке.

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

5. Использование переменных. Программа «Сложение»

Две программы, которые мы уже разработали в системе ***Delphi***, позволяли только выводить информацию на экран компьютера (программа ***First***) или вводить информацию с клавиатуры с отображением ее на экране (программа ***Redaktor***). Однако основными задачами, ради которых были разработаны компьютеры, являются задачи по обработке информации, в том числе задачи вычислительного характера (само слово «компьютер» в переводе с английского означает «вычислитель»). К составлению программы для решения простейшей вычислительной задачи мы и приступим в данном разделе пособия.

Программа, которую мы будем составлять, должна выполнять следующие действия: она должна обеспечивать ввод двух целых чисел с клавиатуры компьютера (введенные числа будут отображаться в двух текстовых окнах) и выводить в третьем окне сумму этих двух чисел. На примере такой несложной задачи мы и освоим основные приемы разработки программ вычислительного характера.

Для этого нам нужно ознакомиться с новым понятием – переменная. Переменной называется величина, которая в ходе выполнения программы может изменять свое значение. Этим она отличается от константы – величины, которая в ходе выполнения программы остается неизменной. Каждая переменная должна иметь свое имя (часто оно также называется идентификатором). Для наименования переменных существуют следующие правила: имя переменной должно состоять из букв латинского алфавита. Кроме букв в имени могут присутствовать и цифры, но начинаться имя обязательно должно с буквы.

Примеры правильных имен переменных:

x summa y1 time45

Неправильные имена переменных:

1x (имя не должно начинаться с цифры)
общ1 (в имени не должно быть букв кириллицы)

Переменная может принимать различные значения, для присваивания которых в языке Object Pascal имеется специальный оператор присваивания. Этот оператор состоит из двух частей: в левой части находится имя переменной, а в правой указывается присваиваемое значение. Левая часть от правой отделяется двоеточием со знаком равенства. Таким образом, общий вид оператора присваивания следующий:

имя:=значение;

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

Обратите внимание на то, что в конце оператора ставится точка с запятой. Этот знак должен в обязательном порядке стоять в конце оператора, если только сразу за оператором не следует служебное слово ***end.*** В простейшем случае присваиваемым значением является какое-либо число. Тогда оператор может выглядеть например так:

x:=5;

Переменной можно присваивать и значение другой переменной. Например, если в программе, написанной на языке ***Object Pascal***, мы увидим такую запись:

x:=y;

то она означает, что переменной ***x*** мы присваиваем значение переменной ***y***. При этом значение самой переменной ***y*** остается неизменным, а значение переменной ***x*** изменяется. Оно становится равным текущему значению переменной ***y***.

Можно встретить в программе и такую запись:

x:=x+1;

Если бы в математике приравняли друг к другу два числа, одно из которых больше другого на единицу, то это конечно было бы ошибкой. Но с точки зрения программирования эта запись не является ошибочной. Она означает, что переменной ***x*** присваивается новое значение, которое на единицу больше, чем старое значение этой же переменной.

Если требуется решить задачу сложения значений двух переменных и присвоения полученной суммы третьей переменной, то вышеуказанная задача решается с помощью следующей последовательности действий:

```
a:=20;
b:=50;
c:=a+b;
```

В данном примере присваиваем начальные значения переменным ***a*** и ***b***, складываем их (операции сложения и вычитания в языке ***Object Pascal*** записываются так же, как и в математике) и полученное значение присваиваем переменной ***c***. Конечно, в данной задаче искомую величину несложно подсчитать и в уме, но есть ряд соображений, по которым решение ряда задач, аналогичных вышеуказанной, лучше оформлять в виде специальной программы.

При работе с переменными необходимо иметь в виду, что каждая переменная, используемая в программе, должна быть предварительно описана. Если этого не сделать, то при запуске программы на выполнение компилятор

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

выдаст сообщение об ошибке. Пример, показывающий, как может выглядеть сообщение об ошибке, приведен ниже:

[Error] Unit1.pas(37): Undeclared identifier: 'x'
[Fatal Error] Project1.dpr(5): Could not compile used unit 'Unit1.pas'

В первой строке данного сообщения указывается номер строки в тексте программы, в которой была обнаружена ошибка (в приведенном примере строка №37), а также приводится имя той переменной, которая не была заранее описана (в примере – это переменная ***x***). Это имя заключается в одиночные кавычки, называемые апострофами. Во второй строке говорится о том, что данная ошибка является фатальной, т. е. из-за нее программа не может работать.

Для того чтобы избежать подобной ситуации, нужно в текст программы вставить раздел описания переменных. Этот раздел начинается со служебного слова ***var*** и включает в себя помимо этого слова имена переменных, используемых в данной программе, причем для каждой переменной должен быть обязательно указан ее тип. Имя переменной от его типа отделяется двоеточием. Если в программе используется несколько переменных, относящихся к одному и тому же типу, то разрешается перечислять имена этих переменных через запятую, а затем после двоеточия указывать их общий тип. Для описания целочисленных переменных, которые мы будем использовать в данной программе, применяется стандартный тип ***integer*** (слово ***integer*** в переводе с английского и означает «целый»). Следовательно, если в программе используется только одна целая переменная ***x***, то ее описание будет выглядеть следующим образом:

var x:integer;

Если же в программе присутствуют две целые переменные ***x*** и ***y***, то описание их будет выглядеть следующим образом:

var x,y:integer;

Теперь осталось выяснить, куда следует помещать данную строку. Для этого нужно знать, что все переменные, используемые в языке ***Object Pascal***, подразделяются на два основных вида. Это глобальные переменные, которые могут использоваться во всей программе, и локальные переменные, которые можно использовать только внутри определенной процедуры. Практика программирования показывает, что в программе должно быть как можно меньше глобальных переменных, так как если программист допускает ошибку, связанную с использованием такой переменной, то для выявления и устранения такой ошибки ему приходится анализировать весь текст программы, который может быть достаточно большим по объему. Соответственно и поиск ошибки

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

может занять много времени. Если же ошибка была допущена при работе с локальной переменной, то для ее устранения достаточно будет поработать только с текстом соответствующей процедуры, что займет значительно меньше времени.

При использовании локальных переменных раздел описания переменных вставляется в процедуру сразу после заголовка данной процедуры, который начинается со служебного слова procedure. Для того чтобы вставить между заголовком процедуры и последующим текстом пустую строку, в которой будет в дальнейшем приведено описание переменной, достаточно поместить курсор в конец строки заголовка нажатием клавиши ***End*** и затем нажать клавишу ***Enter***. Операторы же, которые непосредственно описывают действия, совершаемые с переменными (присваивание начальных значений, вычисление значений переменных и другие), должны располагаться в основной части программы между служебными словами ***begin*** и ***end***.

Таким образом, текст процедуры, в которой двум переменным ***x*** и ***y*** присваиваются целочисленные значения, а затем эти числа складываются и сумма присваивается третьей переменной ***z*** (естественно, что третья переменная также должна относиться к целому типу), будет выглядеть следующим образом:

```
procedure s;
var x,y,z:integer;
begin
x:=10;
y:=5;
z:=x+y
end;
```

В вышеуказанном примере значения присваивались переменным непосредственно в самой программе. При разработке же нашего проекта нам придется столкнуться с иной ситуацией. Числовые величины будут вводиться с клавиатуры в текстовые окна и затем уже использоваться для вычислений. Данные же, вводимые пользователем в текстовое окно, интерпретируются системой программирования как текст даже в том случае, если на самом деле эти данные являются числами. Поэтому для того, чтобы перевести данную текстовую информацию в числовую, необходимо использовать стандартную функцию, являющуюся частью языка ***Object Pascal***. Эту операцию выполняет функция ***StrtoInt***. В названии данной функции ***Str*** - это сокращение от английского слова ***string*** (строка), а ***Int*** - от ***integer*** (целое). Таким образом, название данной функции можно расшифровать как «преобразовать строку в целое число». Аргументом данной функции является текстовая величина, а значением соответствующая числовая величина. Функции в языке ***Object Pascal*** не являются самостоятельными операторами, а входят составной частью в

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

другие операторы, например оператор присваивания. Приведем пример оператора с использованием функции:

x:=StrtoInt(t);

В данном операторе происходит преобразование текстовой величины ***t*** в числовую, и эта числовая величина присваивается целочисленной переменной ***x.*** В программе же в качестве аргумента чаще всего будет выступать свойство какого-либо объекта, например свойство ***Text*** объекта ***Edit***. Значением данного свойства, как мы уже знаем по предыдущему проекту, является текст, введенный пользователем в текстовое окно. Поэтому операция преобразования введенного в окно ***Edit1*** текста в числовую величину может выглядеть следующим образом:

x:=StrtoInt(Edit1.Text);

Когда введенные пользователем данные будут преобразованы в числовые величины и над ними будут произведены все необходимые действия, перед программистом встанет обратная задача: как полученный численный результат преобразовать в текст, который может быть выведен на экран компьютера. Для этой цели используется другая стандартная функция ***InttoStr***. Как нетрудно догадаться по названию этой функции, она выполняет операцию, обратную операции ***StrtoInt***. Функция переводит числовую величину в соответствующую ей строковую величину. Значения этой функции могут присваиваться свойствам имеющихся в проекте графических объектов. В качестве таких свойств, может выступать, например, свойство ***Caption*** объекта ***Label*** или свойство ***Text*** объекта ***Edit.*** Если мы в нашем проекте хотим вывести полученный в программе числовой результат (например, значение переменной ***z***) в текстовом окне ***Edit2***, то оператор, выполняющий в программе это действие, будет выглядеть так:

Edit2.Text:=InttoStr(z);

Теперь, освоив основные понятия и операторы, используемые при работе с переменными, мы можем приступать к непосредственному созданию нашего проекта. Как обычно, работу над программой мы начнем с создания ее графического интерфейса. Разработку интерфейса начинаем с настройки самой формы. Мы изменим заголовок формы (свойство ***Caption*** объекта ***Form1***). Теперь окно формы будет называться “Сложение”. Для того чтобы изменить цвет формы на менее унылый, чем стандартный серый, мы откроем раскрывающийся список цветов, находящийся в свойстве ***Color***, и выберем из списка бледно-зеленый цвет, который по-английски называется ***ClMoneyGreen***, то есть цвет денег. Очевидно, имеется в виду цвет долларовых купюр старого образца.

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

Кроме основной формы, графический интерфейс данной программы будет включать в себя следующие элементы. Во-первых - это два текстовых окна, в которые будут вводиться исходные данные. Во-вторых, – это третье текстовое окно, в котором будет отображаться сумма, т. е. результат работы программы. Для этого на форме нужно создать объекты ***Edit1, Edit2*** и ***Edit3***. У всех трех объектов удаляем содержимое свойства ***Text,*** с тем, чтобы в начале работы программы текстовые окна были пустыми.

Для того чтобы пользователю было понятно, куда именно нужно вводить числа и где получится их сумма, выведем на форму три поясняющие надписи – по одной надписи к каждому текстовому окну. Это будут объекты с именами ***Label1, Label2*** и ***Label3***. Создание этих элементов интерфейса не должно вызвать затруднений у пользователя, так как аналогичные объекты мы создавали в предыдущих программах. Размер шрифта надписей увеличиваем до 10, шрифт делаем полужирным и цвет надписей изменяем на синий. Для каждого из этих объектов изменяем свойство ***Caption***. Для первой надписи заголовок будет – «Введите первое число», для второй – «Введите второе число» и для третьей – «Сумма двух чисел равна».

Завершаем создание графического интерфейса выводом на форму двух экранных кнопок. При нажатии на первую кнопку ***Button1*** будет производиться операция сложения двух чисел, а нажатие второй кнопки ***Button2*** будет закрывать программу. Как обычно для экранных кнопок мы изменяем свойство ***Caption*** (надписи на кнопках). На первой кнопке должно быть написано «Сложить», а на второй – «Выход». Графический интерфейс данной программы пользователь может увидеть на рис. 11.

Теперь мы можем приступить к написанию кода для данной программы. Введем в программу три переменные, значением каждой из которых будет числовое значение текстовой информации, отображаемой в одном из текстовых окон. Поэтому данные переменные мы назовем ***e1***,***e2*** и ***e3*** (в соответствии с окнами ***Edit1, Edit2*** и ***Edit3***). Действия с данными переменными будут производиться при нажатии экранной кнопки ***Button1.*** Поэтому описание этих переменных вставим в процедуру ***TForm1.Button1Click***. Данная процедура описывает реакцию кнопки на щелчок мышью по ней. Как читатель конечно помнит, для того, чтобы создать заготовку данной процедуры, достаточно выделить на форме объект ***Button1***, выбрать вкладку ***Events*** для данного объекта в инспекторе объектов и дважды щелкнуть мышью справа от события ***Onclick.*** Описание переменных будет выглядеть так:

```
var e1,e2,e3:integer;
```

и находиться оно будет сразу после строки заголовка процедуры ***TForm1.Button1Click.***

Переменным ***e1*** и ***e2*** будут присвоены значения первого и второго слагаемых, вводимых в текстовые окна ***Edit1*** и ***Edit2.*** Для преобразования текста, вводимого в эти окна, в числовые значения переменных ***e1*** и ***e2***

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

воспользуемся известной нам стандартной функцией языка ***Object Pascal InttoStr.*** Преобразование произведем с помощью следующих операторов:

```
e1:=StrtoInt(Edit1.text);
e2:=StrtoInt(Edit2.text);
```

Следующий оператор в программе производит сложение значений двух переменных и присваивает сумму переменной ***e3***:

```
e3:=e1+e2;
```

Наконец, осталось преобразовать полученное числовое значение в текстовое, которое будет отображаться в окне ***Edit3***. Для этого используется оператор:

```
Edit3.Text:=InttoStr(e3)
```

Поскольку данный оператор является последним перед служебным словом end, обозначающим конец процедуры, точку с запятой в конце оператора можно не ставить (хотя, если мы ее поставим, это также не будет считаться ошибкой). Таким образом, целиком процедура, описывающая сложение двух чисел при нажатии экранной кнопки «Сложить», будет выглядеть следующим образом:

```
procedure TForm1.Button1Click(Sender: TObject);
var e1,e2,e3:integer;
begin
e1:=strtoint(Edit1.text);
e2:=strtoint(Edit2.text);
e3:=e1+e2;
Edit3.Text:=InttoStr(e3)
end;
```

Строка заголовка процедуры, а также служебные слова ***begin*** и ***end*** генерируются системой программирования автоматически. Остальные строки в процедуру мы добавили сами. Для завершения работы осталось вставить строку кода в процедуру ***Button2Click***, которая выполняется при щелчке мышью на экранной кнопке ***Button2***. Эта строка состоит из одного слова – ***Close.*** Теперь программа готова к работе. Далее приводим полный текст данной программы:

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
```

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

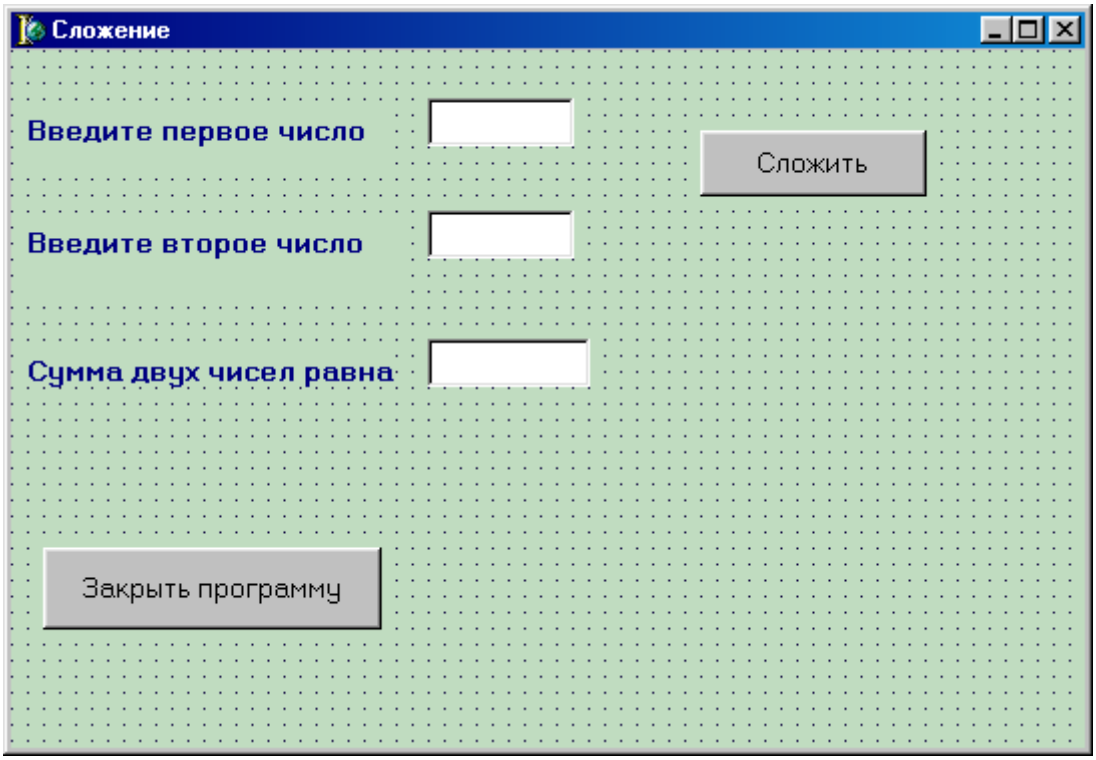

Рис. 11. Графический интерфейс программы «Сложение»

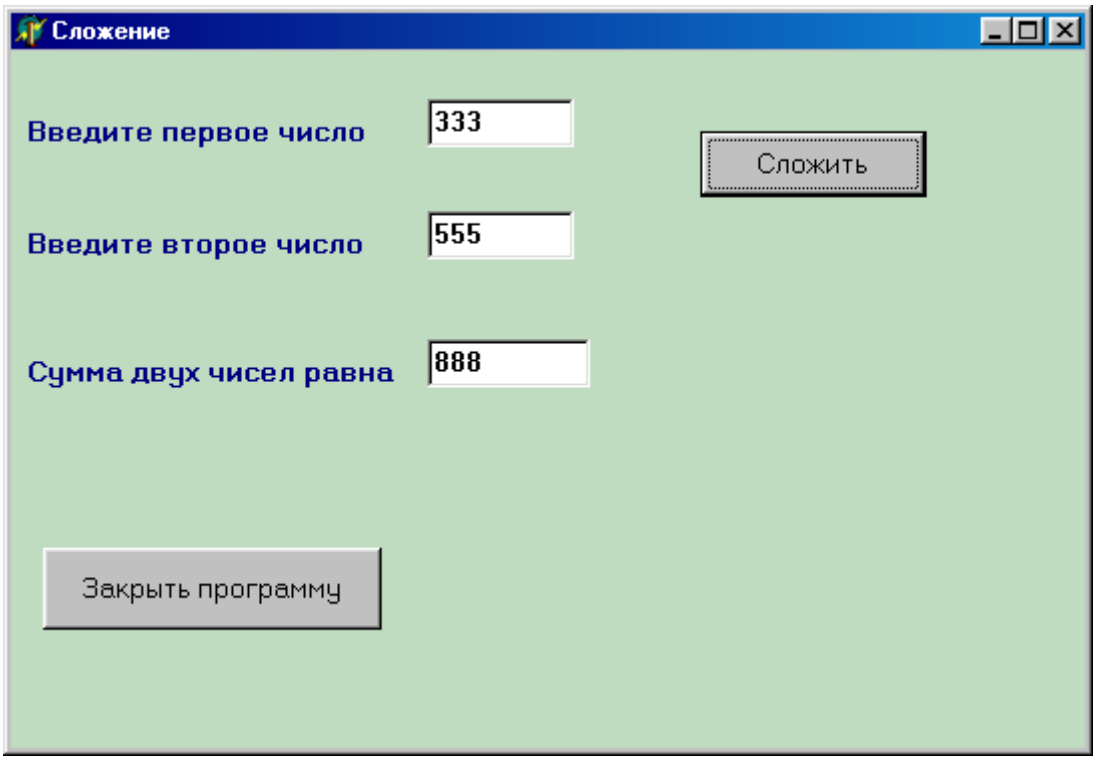

Рис. 12. Окно программы «сложение» во время работы

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

```
Dialogs, StdCtrls;

type
  TForm1 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
    Label3: TLabel;
    Edit3: TEdit;
    Button1: TButton;
    Button2: TButton;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
var e1,e2,e3:integer;
begin
e1:=strtoint(Edit1.text);
e2:=strtoint(Edit2.text);
e3:=e1+e2;
Edit3.Text:=InttoStr(e3)
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
close
end;

end.
```

В ходе работы программы исходные данные вводятся в текстовые окна, причем переход между окнами может производиться либо нажатием клавиши

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

Tab, либо щелчком мышью в соответствующем окне. Значения слагаемых в ходе работы программы можно неоднократно менять и при каждом щелчке на кнопке «Сложить» в окне результата будет видна сумма, полученная с учетом этих изменений. Окно программы во время работы показано на рис.12.

6. Работа с иллюстрациями. Программа «Параллелепипед»

Следующая задача, которую мы будем решать в системе ***Delphi***, заключается в создании программы, которая будет определять объем параллелепипеда по известным длине, ширине и высоте данной геометрической фигуры, а также вычислять общую площадь его поверхности. Как читатель помнит из школьного курса математики, первая часть задачи является очень простой, так как объем параллелепипеда определяется путем вычисления произведения его длины, ширины и высоты. Задача нахождения общей площади поверхности чуть сложнее, но и она требует знания лишь элементарной математики, так как общая площадь поверхности представляет собой сумму площадей отдельных граней параллелепипеда. Однако именно на примере такой простой задачи легче всего освоить новые элементы интерфейса, который в дальнейшем будут использоваться в других программах.

Для начала как всегда, мы настроим саму форму, изменив ее заголовок на «Параллелепипед» и цвет на бледно-зеленый. Затем поместим на форму уже знакомые нам элементы интерфейса. Это будет пять текстовых окон. В первое, второе и третье окно будут вводиться соответственно длина, ширина и высота параллелепипеда. Эти объекты имеют имена ***Edit1, Edit2*** и ***Edit3***. В окне ***Edit4*** будет отображаться вычисленный компьютером объем параллелепипеда, а в окне ***Edit5*** – подсчитанная компьютером общая площадь поверхности этой фигуры. Для того, чтобы все эти окна в начале работы программы оставались чистыми, очищаем свойство ***Text*** для всех этих объектов.

Каждому текстовому окну должна соответствовать надпись, которая поясняет его назначение. Поэтому на форме также размещаем пять надписей с именами начиная с ***Label1*** и кончая ***Label5***. Для каждой из этих надписей, как и в предыдущей программе, изменяем размер, начертание и цвет шрифта. Естественно, что у каждой надписи нужно изменить свойство ***Caption***, для того чтобы заголовок каждой надписи говорил о содержимом находящегося справа от надписи окна. Кроме того, добавим на форму две экранные кнопки ***Button1*** и ***Button2***. При нажатии на первой из этих кнопок будет происходить вычисление искомых величин. Поэтому дадим ей заголовок «Подсчитать». Вторая кнопка будет закрывать программу и получит соответствующий заголовок.

Все компоненты, используемые для создания вышеуказанных элементов интерфейса, хорошо известны по предыдущим программам. Но, помимо них,

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

для наглядности программы хорошо было бы, чтобы в окне программы был изображен тот самый геометрический объект, характеристики которого мы рассчитываем. Поэтому в окно программы желательно поместить изображение параллелепипеда. Для того чтобы картинка хорошо смотрелась, рекомендуется создать для нее соответствующий фон. Для создания такого фона используем компонент ***Panel*** (панель), находящийся на вкладке ***Standard*** панели компонентов и выглядящий следующим образом:  . Перетащив этот компонент на форму, получаем объект ***Panel1***, на котором впоследствии будет находиться картинка.

Следующим шагом станет настройка свойств объекта ***Panel1***. Во-первых, так как данный объект будет только фоном для другого – самого рисунка, то заголовок объекта ***Panel1*** не должен отображаться на экране. Поэтому на вкладке ***Properties*** данного объекта находим свойство ***Caption*** и очищаем его. Если размеры панели нас не устраивают, изменяем их как обычно методом перетаскивания.

Для создания визуального эффекта, желательно, чтобы изображение было как бы немного вдавлено в экран. Этого можно добиться, изменив характеристики фаски, которая окружает по краям панель. По умолчанию ширина этой фаски (свойство ***Bevelwidth***) равна 1 – одному пикселу. Увеличим эту ширину до двух пикселов, введя в поле справа от названия свойства соответствующее число. Кроме того, изменим сам характер этой фаски. Для этого щелкнем раскрывающийся список, содержащийся в свойстве ***BevelOuter***. Значение данного свойства определяет внешний вид фаски. Вместо имеющегося по умолчания значения ***BvNone*** мы устанавливаем значение ***BvLowered***. В результате создается впечатление, что панель «утоплена» в экран.

Теперь на панель поместим еще один объект – сам рисунок. Компонент для его создания находится на вкладке ***Additional*** панели компонентов и называется ***Image.*** Выглядит компонент ***Image*** так:  . Перетаскивая на панель этот компонент, получаем объект под названием ***Image1***. На следующем этапе нужно изменить расположение этого объекта на экране и его размеры таким образом, чтобы он закрывал большую часть панели и в то же время оставлял на виду фаски панели.

Для того чтобы это сделать, нужно прежде всего иметь в виду, что место расположения графического объекта на экране компьютера определяется положением его верхнего левого угла. Поэтому нужно совместить верхний левый угол объекта ***Image1*** с верхним левым углом фоновой панели ***Panel1.*** Выделяем в окне формы объект ***Image1*** и в инспекторе объектов просматриваем вкладку свойств данного объекта (вкладка ***Properties***). На данной вкладке находим свойство ***Top*** (в переводе с английского – верх). Данное свойство определяет положение рисунка относительно верха панели, на

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

которой он расположен. Задаем для свойства ***Top*** значение 2, т. е. верх рисунка будет на два пиксела ниже верха панели для того, чтобы не закрывать фаску. Затем находим на той же вкладке свойство ***Left*** (в переводе с английского – левый), которое определяет положение рисунка относительно левого края панели. Для свойства ***Left*** также устанавливаем значение, равное 2.

Теперь, когда рисунок «привязан» к верхнему левому углу панели, нужно определить его геометрические размеры. Размеры рисунка, как по горизонтали, так и по вертикали должны быть на 4 пиксела меньше, чем соответствующие размеры панели, для того, чтобы не закрывать фаску. Размеры же панели можно определить наведением указателя мыши на любое свободное место панели. Если задержать указатель мыши на панели на несколько секунд, то на панели появится небольшое окно с основными характеристиками объекта, включая и его размеры. В этом окне будет информация следующего вида:

Size: 294*182

слово ***Size*** по-английски означает «размер», первое число – это размер панели в пикселах по горизонтали, а второй размер – по вертикали. Естественно, что при разработке проекта числовые значения могут отличаться от выше приведенных.

Для приведенного примера размер рисунка по горизонтали должен быть равен 290, а по вертикали – 178. Снова выделяем на форме объект Image1 и задаем для свойства ***Width*** (ширина) значение 290, а для свойства ***Height*** (высота) значение 178. После определения геометрических размеров рисунка осталось сделать так, чтобы картинка, которую будет содержать рисунок, автоматически подгонялась под размеры рисунка, растягивалась соответственно рисунку. Для этого находим свойство ***Stretch*** (растянуть). Это свойство логическое и имеет только два значения – ***False*** и ***True***. Устанавливаем для данного свойства значение ***True***, и растяжка картинки будет производиться автоматически.

Теперь у пользователя может возникнуть вопрос, о том, где взять картинку, которая будет размещена на рисунке. Здесь возможны следующие варианты: Вы можете найти подходящую картинку в Интернете, скопировать ее с одного из компакт-дисков с коллекциями рисунков, поискать подходящую иллюстрацию в коллекции картинок, входящих в состав пакета ***Microsoft Office***, либо нарисовать ее самостоятельно в любом графическом редакторе, например в редакторе ***Paint,*** являющемся стандартным приложением операционной системы ***Windows***. В данном случае мы воспользуемся последним вариантом, так как картинку с изображение параллелепипеда нетрудно нарисовать самостоятельно. Созданную картинку сохраняем в той же папке, где находятся файлы нашего проекта под именем «Картинка1».

Для того чтобы «загрузить» эту картинку в объект ***Image1***, выделяем этот объект в форме и находим свойство ***Picture*** (картинка). Щелкаем мышью в поле

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

справа от этого свойства. В поле появляется уже знакомая нам кнопка-построитель. При щелчке на этой кнопке открывается диалоговое окно ***Picture Editor*** (редактор картинок), которое изображено на рис.13. В данном окне щелкаем мышью находящуюся внизу экранную кнопку ***Load...*** После щелчка на этой кнопке открывается новое диалоговое окно ***Load Picture*** (загрузка картинки), внешний вид которого приведен на рис.14. В верхней части этого окна имеется адресная строка, которая позволяет перемещаться по файловой структуре компьютера с целью поиска файла с картинкой. Обнаружив искомый файл, выделяем его и щелкаем экранную кнопку «Открыть». После этого окно ***Load Picture*** закрывается, и мы вновь видим на экране диалоговое окно ***Picture Editor***. На этот раз вместо белого поля в центре окна мы увидим уже загруженную картинку. Осталось подтвердить загрузку картинки щелчком на экранной кнопке «***Ok***».

После закрытия окна ***Load Picture*** графический интерфейс программы приобретет законченный вид, который представлен на рис.15. Остается дополнить проект кодом, описывающим реакцию программы на действие экранных кнопок.

Создание кода начинаем с того, что двойным щелчком справа от события ***Onclick*** для объекта ***Button1*** открываем процедуру ***TForm1.Button1Click***. В процедуре используются переменные ***a,b,c,*** которые содержат числовые значения длины, ширины и высоты параллелепипеда, а также ***s*** – площадь поверхности параллелепипеда и ***v*** – объем параллелепипеда. Описание переменных вставляем после строки заголовка процедуры:

```
var a,b,c,s,v:integer;
```

Затем в основной части процедуры (после слова ***begin***) вставляем группу операторов, которые преобразуют введенные в текстовые окна длину, ширину и высоту в числовые значения. Эти операторы, использующие стандартную функцию ***InttoStr***, будут выглядеть так:

```
a:=strtoint(Edit1.Text);
b:=strtoint(Edit2.Text);
c:=strtoint(Edit3.Text);
```

Следующие операторы вычисляют объем, который присваивается переменной ***v,*** и преобразуют числовое значение этой переменной в текст, выводимый в окне ***Edit4***. Обратите внимание, что для обозначения операции умножения в языке ***Object Pascal*** используется символ *******, а не принятая в математике точка.

```
v:=a*b*c;
Edit4.Text:=inttostr(v);
```

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

Теперь можно определить общую площадь поверхности параллелепипеда, которая представляет собой сумму площадей всех его шести граней. Площади граней представляют собой произведение длины и ширины, длины и высоты, ширины и высоты, причем площади передней и задней грани, верхней и нижней, левой боковой и правой боковой граней равны между собой. Поэтому достаточно найти по отдельности площади только трех граней и каждую полученную площадь умножить на два, а затем сложить полученные произведения. После вычисления значения переменной ***s*** нужно преобразовать ее в текст, выводимый в окне ***Edit5***.

Вышеуказанная последовательность действий реализуется с помощью следующих операторов:

```
s:=2*a*b+2*a*c+2*b*c;
Edit5.Text:=inttostr(s)
```

Таким образом, процедура ***TForm1.Button1Click*** в целом будет выглядеть так:

```
procedure TForm1.Button1Click(Sender: TObject);
var a,b,c,s,v:integer;
begin
a:=strtoint(Edit1.Text);
b:=strtoint(Edit2.Text);
c:=strtoint(Edit3.Text);
v:=a*b*c;
Edit4.Text:=inttostr(v);
s:=2*a*b+2*a*c+2*b*c;
Edit5.Text:=inttostr(s)
end;
```

Последнее, что осталось сделать это вставить команду ***Close,*** закрывающую окно программы в процедуру ***TForm1.Button2Click,*** описывающую реакцию на щелчок мышью на экранной кнопке ***Button2***. Ниже приводится полный текст программы:

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs, StdCtrls, ExtCtrls;
```

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

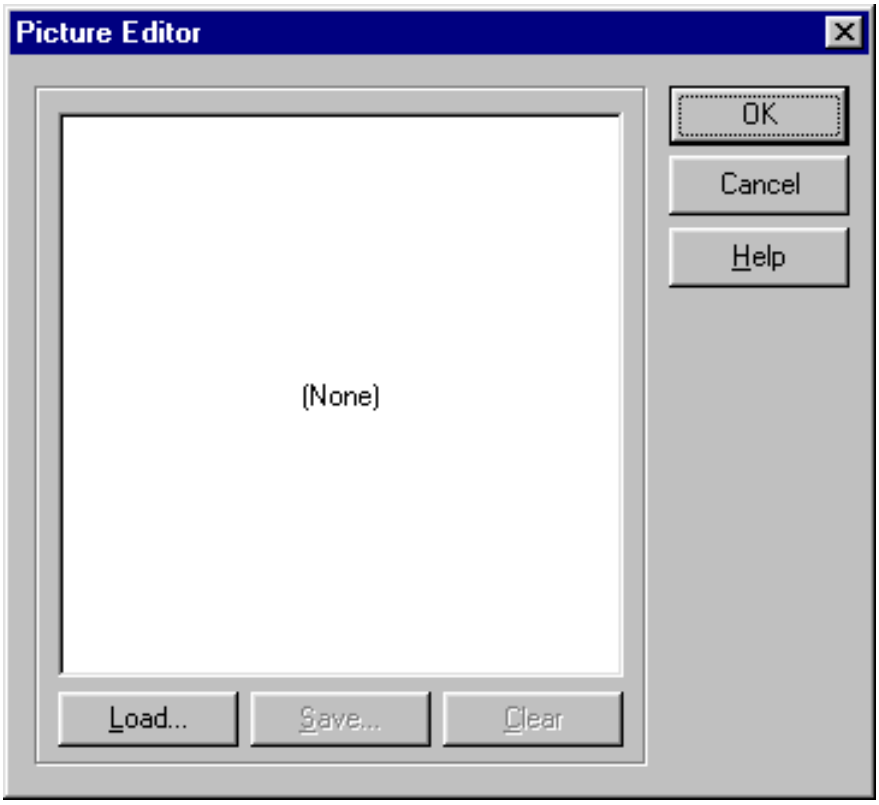

Рис. 13. Диалоговое окно редактора картинок

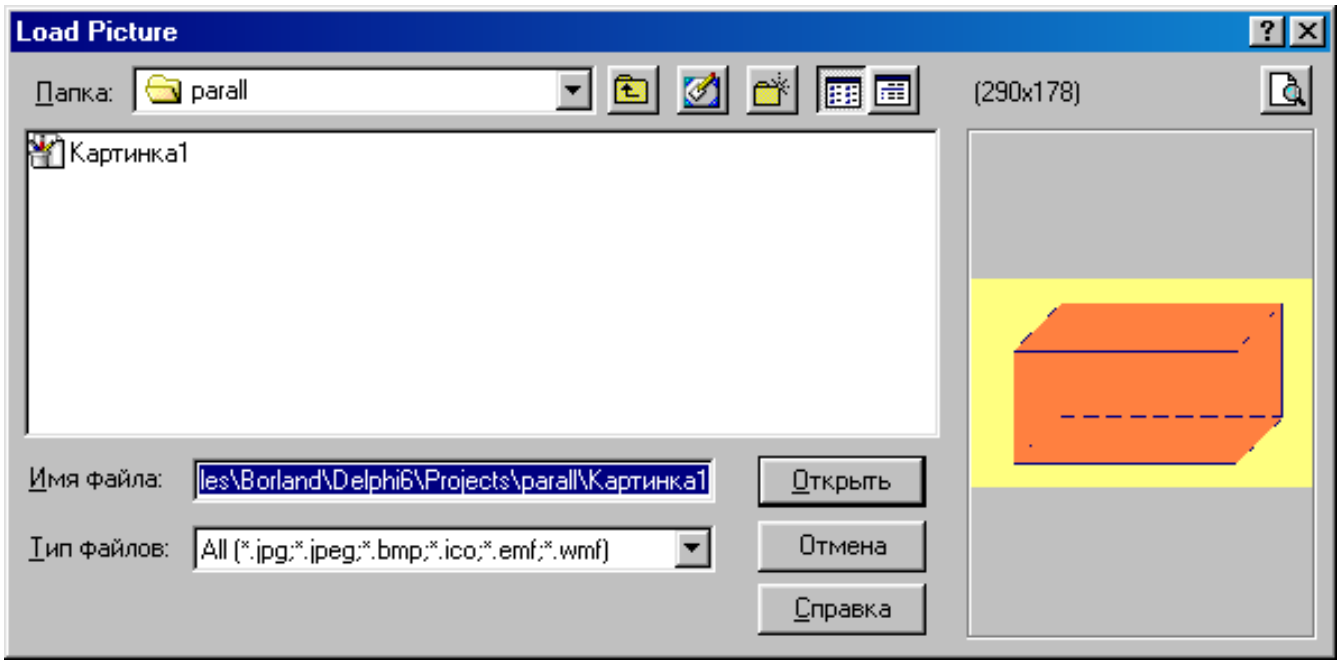

Рис. 14. Диалоговое окно «загрузка картинки»

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

```
type
  TForm1 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
    Label3: TLabel;
    Edit3: TEdit;
    Label4: TLabel;
    Edit4: TEdit;
    Label5: TLabel;
    Edit5: TEdit;
    Button1: TButton;
    Button2: TButton;
    Panel1: TPanel;
    Image1: TImage;
    procedure Button1Click(Sender: TObject);
   procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
var a,b,c,s,v:integer;
begin
a:=StrtoInt(Edit1.Text);
b:=StrtoInt(Edit2.Text);
c:=StrtoInt(Edit3.Text);
v:=a*b*c;
s:=2*(a*b+a*c+b*c);
Edit4.Text:=InttoStr(v);
Edit5.Text:=InttoStr(s);
end;

procedure TForm1.Button2Click(Sender: TObject);
```

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

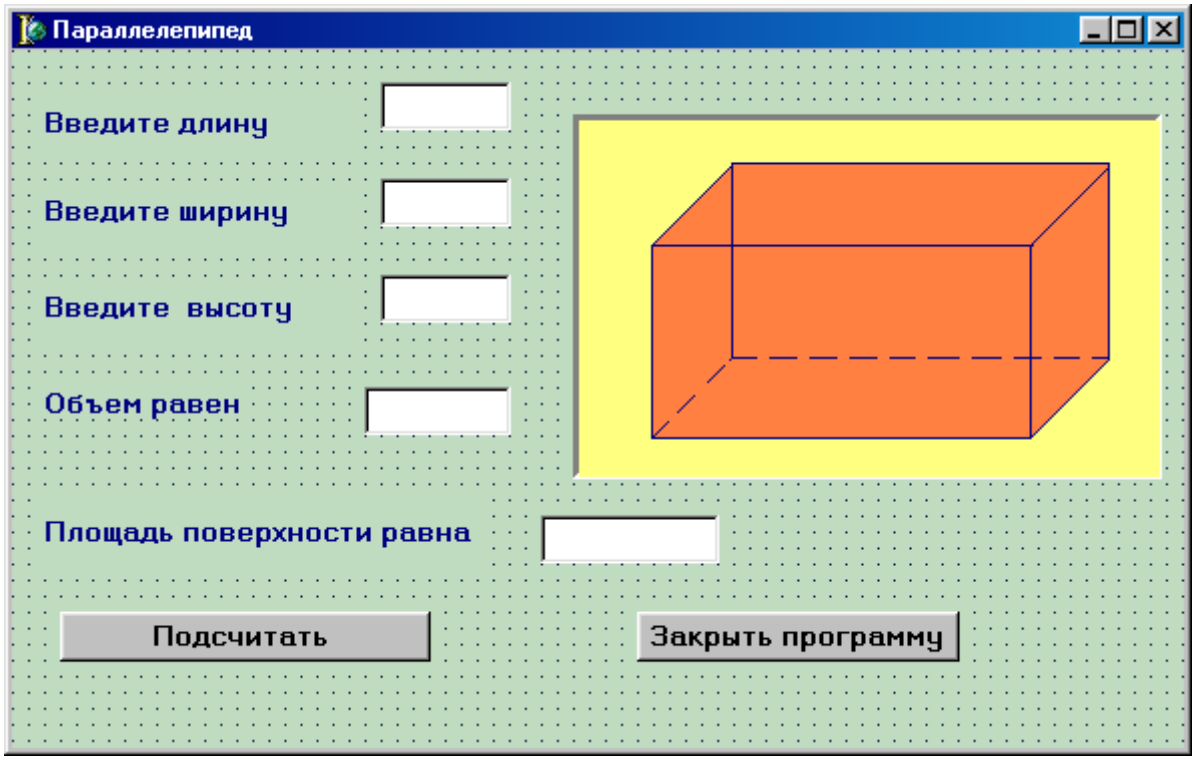

Рис. 15. Графический интерфейс программы «параллелепипед»

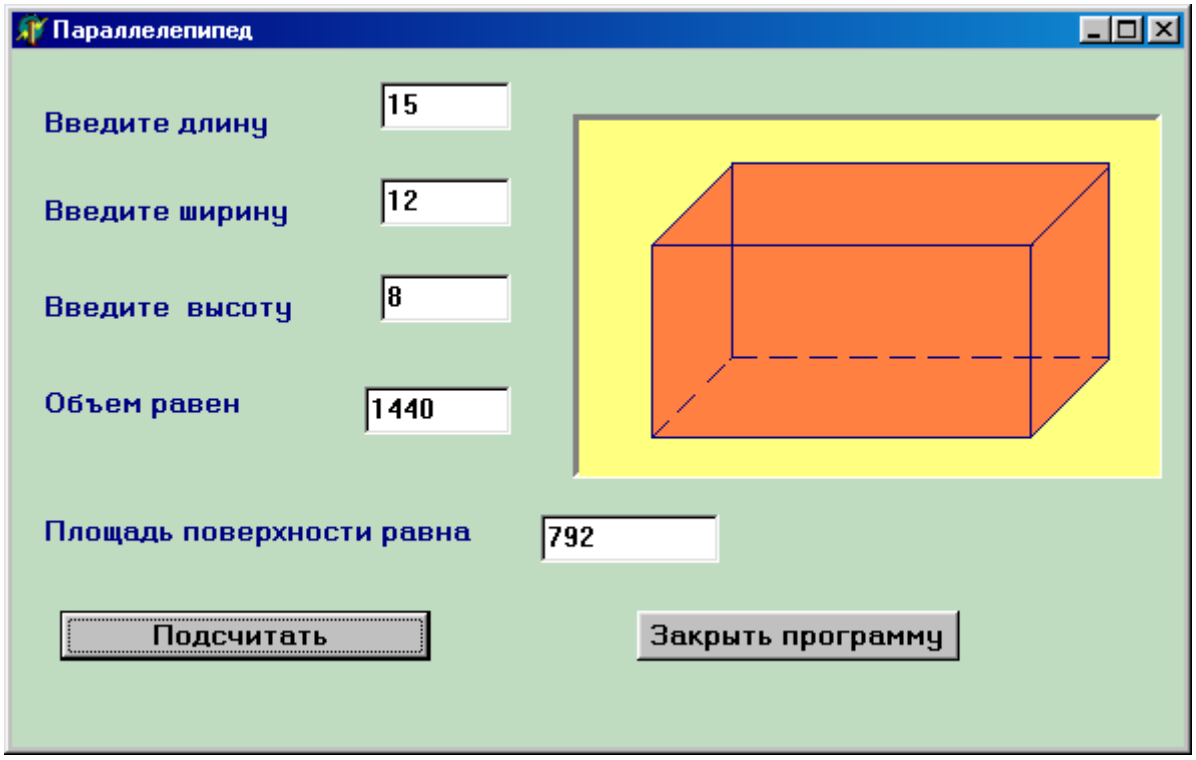

Рис. 16. Программа «параллелепипед» во время работы

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

```
begin
close
end;

end.
```

Теперь программу можно запускать на выполнение. После ввода исходных данных в текстовые окна и щелчка на экранной кнопке «Подсчитать» программа выдаст на экран сразу два результата – объем параллелепипеда и площадь его поверхности. Экран программы в процессе ее работы показан на рис. 16.

7. Операции с целыми числами. Программа «Хронометр»

Как известно, хронометром называется прибор, используемый для точного измерения времени. Проект, который мы будем разрабатывать в данном разделе, также позволит измерять время. В данном случае это будет период, прошедший от одного момента времени до другого. Как первый момент времени, так и второй, в проекте будет задаваться в часах и в минутах. Период между этими двумя моментами мы также будем определять в часах и минутах.

Для того чтобы разобраться в том, как правильно решать поставленную задачу, необходимо познакомиться с математическими действиями, используемыми в языке ***Object Pascal***. Всего в этом языке над числами можно производить шесть различных математических действий.

Четыре действия известны всем из элементарной математики – это сложение, вычитание, умножение и деление. Сложение и вычитание обозначаются теми же знаками «плюс» и «минус», что и в обычной математике и так же выполняются. Умножение тоже по существу не отличается от обычного умножения, но в записи этого действия есть некоторые отличия.

Во-первых, как читатель уже знает, между сомножителями можно ставить только символ * (звездочка). Во-вторых, знак умножения нельзя пропускать, как мы это часто делается в математике. Дело в том, что компьютер в отличие от человека понимает все буквально. Если мы напишем в программе вместо выражения ***a*b*** выражение ***ab***, то компьютер воспримет последнюю запись не как произведение двух переменных ***a*** и ***b***, а как одну переменную с именем ***ab***, что может привести к серьезным ошибкам в программе. Поэтому запомним, что в программировании «звездочку» между сомножителями нужно ставить всегда.

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

Есть некоторые особенности и в выполнении обычной операции деления (эту операцию мы называем обычным делением, так как скоро мы узнаем о существовании еще одной операции деления). Во-первых, при записи операции деления между делимым и делителем ставится не двоеточие, как в обычной математике, а символ / (косая черта). Часто в компьютерной литературе этот символ обозначается также англоязычным термином «слэш», который получил широкое распространение, вероятно, в силу своей краткости. Во-вторых, результатом операции деления одной целой величины на другую становится величина не целого, а вещественного типа. Подробно о том, что такое вещественные величины будет рассказано позднее.

Помимо этих всем известных математических операций в языке Object Pascal используются еще две – операции div и mod. Первая из этих операций называется операцией целочисленного деления. Выполняется она следующим образом: в начале производится обычная операция деления, а затем от получившегося частного берется только целая часть. Дробная часть отбрасывается. Рассмотрим выполнение этой операции на следующем примере. Нужно найти результат операции

18 div 5

Первоначально производим обычное деление – 18/5. Полученный промежуточный результат равен 3,6. Отбрасываем 0,6 и получаем окончательный результат – число 3.

Используя аналогичный метод вычисления, получим, что

22 div 4 =5
47 div 11=4

Операция же ***mod*** – это нахождение остатка при целочисленном делении. Приведем примеры выполнения операции ***mod***:

22 mod 4=2
47 mod 11=3

Как мы видим, при использовании тех же операндов, что и для операции ***div***, операция ***mod*** дает иные результаты (операндами называются те величины, над которыми выполняется операции; в операции деления присутствуют два операнда – делимое и делитель). Полученные знания пригодятся нам в дальнейшем при составлении кода программы, а работу над проектом мы начнем как обычно с создания его графического интерфейса.

В данном проекте интерфейс должен включать в себя следующие элементы:

а) Шесть текстовых окон. В первом и втором окнах будут вводиться исходные данные соответственно о первом моменте времени, в третьем и четвертом окнах – данные о втором моменте. В пятом и шестом окнах должен

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

выводиться результат работы программы – разница между двумя моментами времени.

б) Поясняющие надписи к текстовым окнам – к каждой группе из двух окон на экране должно выводиться по три надписи. Первая из надписей к группе поясняет, что за информация содержится в данной паре окон. Вторая и третья надписи в группе указывают на используемые единицы измерения времени – минуты и часы. Общее количество поясняющих надписей равно девяти.

в) Три экранные кнопки. Первая из них будет производить процесс вычисления результата. Вторая должна выводить информацию об авторе. Третья кнопка закрывает программу.

г) Панель, на которой находится рисунок, изображающий часы (данный рисунок является декоративным элементом программы).

д) Надпись, содержащая сведения об авторе программы.

Создание интерфейса начинаем как обычно с настройки основной формы. В начале изменяем заголовок формы. Для этого справа от свойства ***Caption*** ставим его значение – Хронометр. Затем изменяем фоновый цвет формы. Для этого щелкаем раскрывающийся список справа от свойства ***Color*** и выбираем из списка ярко-голубой цвет – ***ClAqua.***

Затем размещаем на форме первую группу текстовых окон с надписями. Это будут объекты ***Edit1*** – в это окно будет введено количество часов для первого момента времени и ***Edit2*** – количество минут для первого момента времени. Для того чтобы лучше смотрелась вводимая в окна информация, мы изменим характеристики шрифта, для чего работаем со свойством ***Font*** данных объектов. Щелкнув кнопку-построитель, открываем диалоговое окно для настройки шрифта и увеличиваем размер шрифта до 10 кегля, а начертание делаем полужирным.

Над вышеописанными текстовыми окнами помещаем надпись ***Label1***, заголовок которой говорит о том, что в эти окна вводятся данные о первом моменте времени. Справа от каждого из окон размещаем надписи ***Label2*** и ***Label3***, заголовки которых содержат информацию о единицах измерения времени, используемых в каждом из окон. Заголовок ***Label2*** – «час», а ***Label3*** – «мин». Для всех надписей мы изменяем характеристики шрифта – увеличиваем его размер до 12 кегля и делаем его полужирным.

Под первой группой текстовых окон и надписей размещаем вторую группу для ввода данных о втором моменте времени. Эта группа содержит окна ***Edit3*** и ***Edit4*** и надписи ***Label4***, ***Label5*** и ***Label6***. Настройки этих окон и надписей аналогичны соответствующим настройкам для первой группы. Под второй группой оставляем немного свободного места для размещения экранной кнопки.

Третья группа окон и надписей размещается под второй и содержит два текстовых окна ***Edit5*** и ***Edit6***, служащие для вывода результатов работы

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

программы. Над окнами должна находиться поясняющая надпись следующего содержания: «Разница между двумя моментами времени составляет». Однако эта надпись слишком длинна для того, чтобы разместиться на форме в одну строку. Поэтому в данном случае придется воспользоваться следующим приемом: разбить длинную надпись на две более коротких и вывести их одна под одной. Таким образом, пользователь программы увидит на экране текст, состоящий из двух строк, которым будут соответствовать надписи ***Label7*** и ***Label8***. Кроме того, справа от текстовых окон будут располагаться еще две вспомогательные надписи ***Label9*** и ***Label10***, аналогичные таким же надписям в предыдущих двух группах. Настройки окон и надписей также производятся аналогично соответствующим настройкам в предыдущих группах.

В правой верхней части формы расположим панель с находящимся на ней рисунком. Для этого создаем на форме объект ***Panel1***, на котором размещаем объект ***Image1***. Настройка данных объектов сводится к подгонке расположения и размеров рисунка (объект ***Image1***) к расположению и размерам панели (объект ***Panel1***). Производим эту операцию аналогично тому как мы это делали при работе над проектом «Параллелепипед».

Единственное существенное отличие заключается в следующем: мы не будем рисовать загружаемую картинку вручную, а воспользуемся уже готовой. Картинку возьмем из коллекции, входящей в состав программного пакета ***Microsoft Office***, установленного в настоящее время практически на любом компьютере. Для этого мы выделяем на форме объект ***Image1*** и на вкладке свойств объекта находим уже знакомое нам свойство ***Picture***. Щелкнув в поле справа от свойства и затем щелкнув на кнопке-построителе, открываем окно редактора картинок. Щелкнув мышью в открывшемся окне на кнопке ***Load***, открываем окно загрузки картинок.

Навигацию (перемещение) по файловой структуре компьютера в данном окне можно осуществлять двумя способами. Первый из них заключается в использовании кнопки  (переход на один уровень вверх). Щелкая несколько раз по этой кнопке, можно подняться вверх по файловой структуре компьютера, пока не будет достигнут уровень «Мой компьютер». Текущий уровень отображается в небольшом окне, расположенном слева от панели с кнопками. В основном же окне, находящемся внутри окна загрузки, отображаются диски и папки и файлы. Для того, чтобы открыть диск или папку, нужно дважды щелкнуть мышью на соответствующем значке. Открываем диск, на котором находятся файлы пакета ***Microsoft Office*** (при установке данного пакета по умолчанию эти файлы записываются на диск ***C***). Затем находим на этом диске папку ***Program Files***, а внутри нее открываем папку ***Microsoft Office***.

Дальнейшая последовательность действий зависит от той версии ***Microsoft Office***, которая установлена на данном компьютере. Для версии ***Microsoft Office 97*** последовательность действий будет следующей. В папке ***Microsoft Office*** находим вложенную папку ***Clipart***. Внутри папки ***Clipart***

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

содержится папка ***Popular***, в которой и находятся файлы с картинками. Это графические файлы, созданные средствами векторной графики и имеющие расширение wmf. Когда выбираем в окне один из этих файлов, то справа от основного окна появляется изображение, находящееся в данном файле. Сверху от изображения указывается его размер в пикселах (эти сведения могут пригодиться при размещении файла внутри формы). Если изображение слишком мелкое и пользователь хочет рассмотреть его более подробно, то для этого нужно щелкнуть находящуюся также сверху кнопку  (предварительный просмотр). После щелчка на этой кнопке появляется еще одно окно, в котором изображение представлено в натуральную величину. Закрыть это окно можно обычным способом – щелчком на закрывающей кнопке. Для нашего проекта выбираем изображение ***Clock.wmf***. Для загрузки данного изображения щелкаем экранную кнопку “Открыть” и подтверждаем сделанный выбор щелчком на кнопке “***Ok***” в окне редактора картинок. В результате выбранное изображение ***Clock.wmf*** будет размещено на объекте ***Image1.***

Второй способ навигации заключается в следующем: справа от окна, в котором показан текущий уровень, щелкнуть раскрывающую кнопку  . После щелчка открывается вся файловая структура компьютера, представленная в виде дерева файлов. Далее можно перемещаться по дереву файлов, открывая нужные папки. В остальном этот способ навигации не отличается от описанного выше.

Если на компьютере установлена другая версия пакета ***Microsoft Office***, то адрес папки, содержащей изображения, которые можно вставить в проект, может быть иным. Например, для версии ***Microsoft Office XP***, нужно в папке ***Microsoft Office*** открыть вложенную папку ***media***, а внутри папки ***media*** открыть папку ***cagcat10***, которая и содержит коллекцию рисунков. Другими могут быть и сами рисунки. Однако вышеописанные методы загрузки файлов с помощью окна редактирования картинок и окна загрузки остаются при этом неизменными.

Последней группой элементов интерфейса, которую необходимо разместить на форме, являются экранные кнопки. Первая из этих кнопок ***Button1*** отвечает за выполнение расчетов по определению искомой разницы. Вторая кнопка ***Button2*** должна выводить на экран формы сведения об авторе программы. Сведения будут выводиться на экран в виде надписи. Так как данная надпись вновь оказывается слишком длинной для вывода ее на экран в одну строку, то мы компонуем эту надпись из двух объектов ***Label11*** и ***Label12,*** расположенных друг под другом. Так как непосредственно после запуска программы на выполнение надпись со сведениями об авторе не должна быть видна на экране, то производим настройку объектов ***Label11*** и ***Label12***, которая заключается в том, что для свойства ***Visible*** каждого из этих объектов устанавливаем значение ***False***. В результате подобной настройки по умолчанию

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

эти объекты не будут видны. Завершаем создание интерфейса помещением на форму объекта ***Button3*** – кнопки, производящей закрытие программы. Интерфейс данной программы представлен на рис. 17.

Теперь приступаем к написанию кода для данной программы. Начнем этот процесс с описания действий, производимых при нажатии кнопки ***Button1.*** Вначале необходимо разобрать ход производимых вычислений, что позволит нам определить, какие переменные понадобятся при написании кода. Как читатель помнит, все используемые в процедуре переменные необходимо описать после служебного слова ***var***. Во-первых, нужно описать переменные для исходных величин. Количество часов для первого момента времени обозначим как ***c1***, а количество минут для первого момента времени обозначим ***m1***. Количество часов и минут для второго момента времени назовем соответственно ***c2*** и ***m2***. Этих величин в готовом виде у нас нет, но можно воспользоваться стандартной функцией ***StrtoInt***, которая преобразует текстовые величины в числовые. Тогда операторы, преобразующие значения, введенные пользователем в текстовые окна ***Edit1, Edit2, Edit3*** и ***Edit4***, в числовые переменные будут выглядеть так:

```
c1:=StrtoInt(Edit1.Text);
m1:=StrtoInt(Edit2.Text);
c2:=StrtoInt(Edit3.Text);
m2:=StrtoInt(Edit4.Text);
```

Дальнейший ход решения задачи будет выглядеть следующим образом. Переведем первый момент времени из часов и минут только в минуты. Для этого нужно количество часов для первого момента умножить на 60 и к нему прибавить количество минут. Величину первого момента времени, выраженную только в минутах, обозначим ***t1***. Теперь таким же образом преобразуем второй момент времени, а соответствующую величину обозначим ***t2***. В программе эти преобразования будут выражены следующими операторами:

```
t1:=c1*60+m1;
t2:=c2*60+m2;
```

Для того, чтобы найти величину ***t3*** - разницу между этими моментами времени, достаточно вычесть из второго момента первый, что делает оператор:

```
t3:=t2-t1;
```

В результате мы получили искомый результат, но выраженный только в минутах. Желательно этот результат представить, так же как и исходные данные в часах и минутах. Получить количество целых часов, которые содержатся в этой разнице, можно использовав целочисленное деление, т. е.

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

разделив величину ***t3*** на 60 с помощью операции ***div.*** Обозначим найденную величину через ***c3***. Для определения количества оставшихся минут ***m3*** мы прибегнем к операции ***mod***, которая, как известно, находит остаток при целочисленном делении. В качестве делителя будет снова выступать число 60. Таким образом, операторы, вычисляющие количество часов и минут в конечном результате, будут выглядеть так:

```
c3:=t3 div 60;
m3:=t3 mod 60;
```

Искомые величины определены, но необходимо их преобразовать в текст, который будет отображаться в текстовых окнах ***Edit5*** и ***Edit6.*** Это действие, обратное тому, что мы производили в начале процедуры можно реализовать с помощью функции ***InttoStr.*** Операторы, осуществляющие преобразование, будут выглядеть так:

```
Edit5.Text:=InttoStr(c3);
Edit6.Text:=InttoStr(m3);
```

Теперь работа над процедурой ***TForm1.ButtonClick***, описывающей реакцию на нажатие кнопки «Подсчитать разницу», завершена и целиком эта процедура будет выглядеть следующим образом:

```
procedure TForm1.Button1Click(Sender: TObject);
var c1,m1,c2,m2,c3,m3,t1,t2,t3:integer;
begin
c1:=StrtoInt(Edit1.Text);
m1:=StrtoInt(Edit2.Text);
c2:=StrtoInt(Edit3.Text);
m2:=StrtoInt(Edit4.Text);
t1:=c1*60+m1;
t2:=c2*60+m2;
t3:=t2-t1;
c3:=t3 div 60;
m3:=t3 mod 60;
Edit5.Text:=InttoStr(c3);
Edit6.Text:=InttoStr(m3);
end;
```

Следующим шагом в работе над программой должно стать создание процедуры, которая описывает реакцию на нажатие кнопки ***Button2*** – «сведения об авторе». Действие, описываемое в данной процедуре, должно заключаться в том, что невидимая по умолчанию надпись со сведениями об авторе должна становиться видимой. Напомним, что эта надпись

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

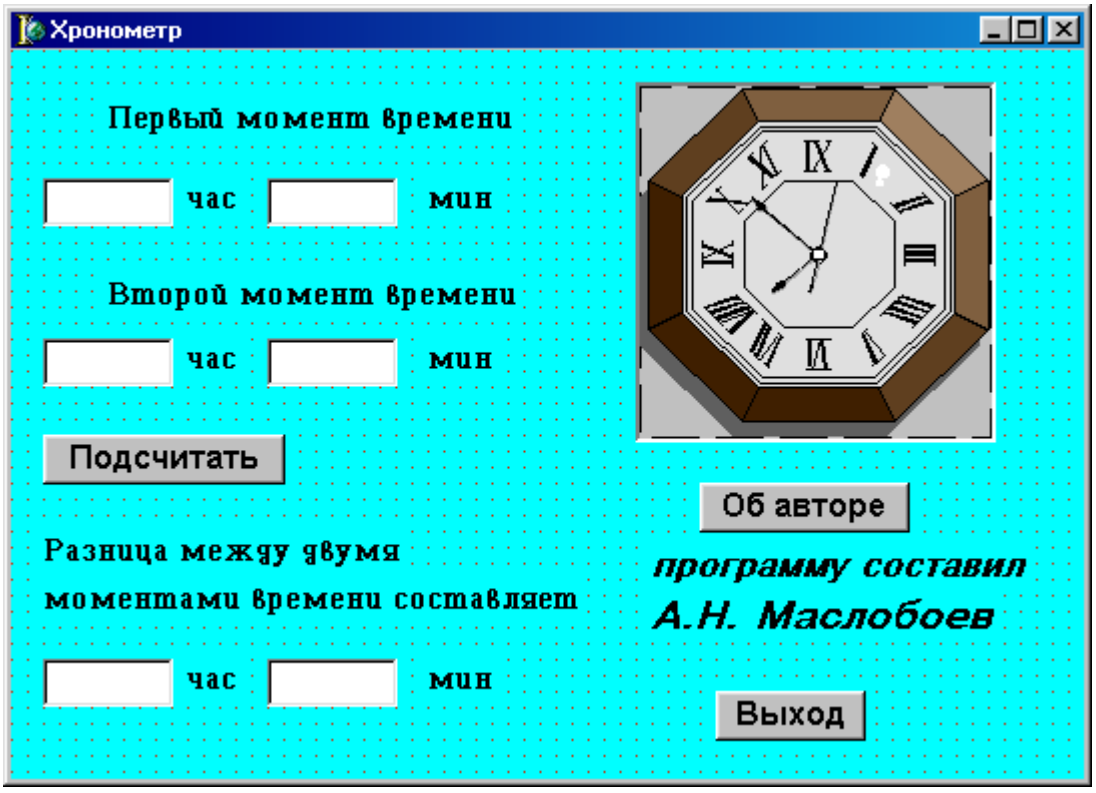

Рис. 17. Графический интерфейс программы «хронометр»

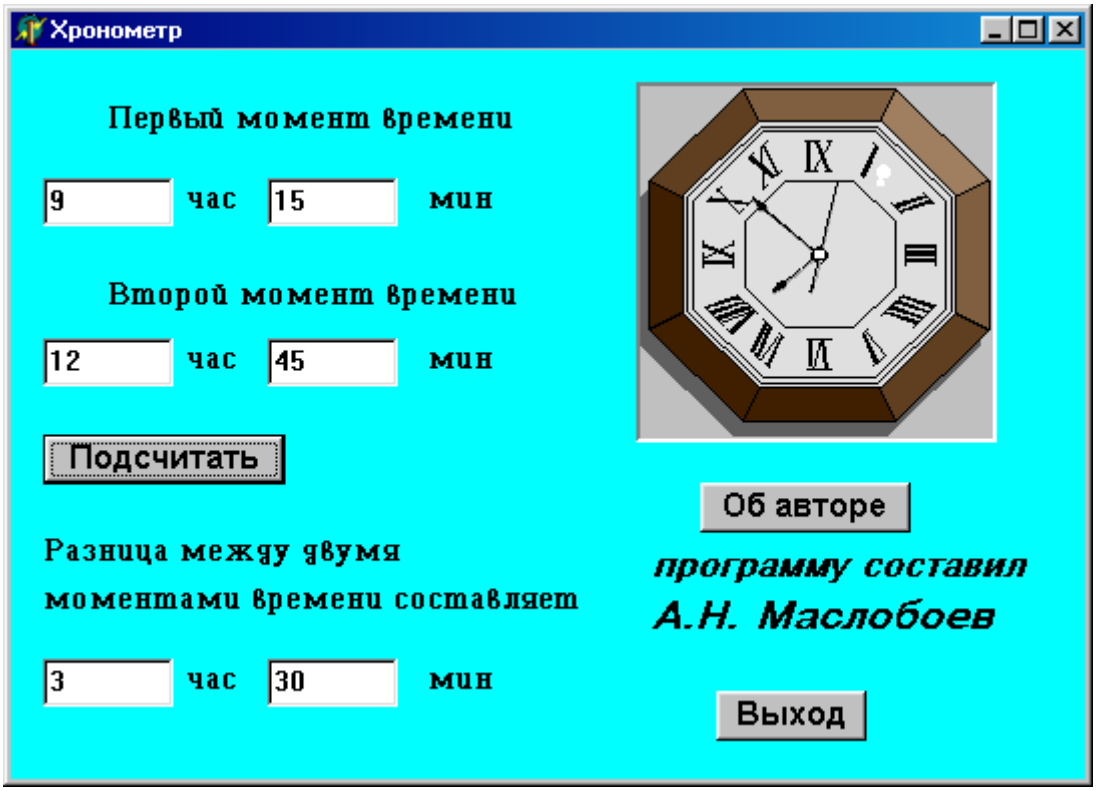

Рис. 18 . Программа «хронометр» во время работы

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

фактически состоит из двух с именами ***Label11*** и ***Label12***. Превращение невидимого объекта в видимый, как уже известно читателю, производится путем присваиванию свойству ***Visible*** данного объекта значения ***True*** (вместо имевшегося по умолчания значения ***False***). Следовательно, процедура, описывающая эти действия, будет выглядеть так:

```
procedure TForm1.Button2Click(Sender: TObject);
begin
Label11.Visible:=True;
Label12.Visible:=True
end;
```

Последнее, что осталось сделать – это записать команду закрытия программы в процедуру, описывающую реакцию на нажатие кнопки «Выход». Процедура эта будет выглядеть так:

```
procedure TForm1.Button3Click(Sender: TObject);
begin
Close
end;
```

Целиком же вся программа должна выглядеть следующим образом:

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls;

type
  TForm1 = class(TForm)
    Edit1: TEdit;
    Edit2: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Edit3: TEdit;
    Edit4: TEdit;
    Label5: TLabel;
    Label6: TLabel;
```

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

```
            Label7: TLabel;
Label8: TLabel;
        Edit5: TEdit;
        Edit6: TEdit;
        Label9: TLabel;
        Label10: TLabel;
        Button1: TButton;
        Panel1: TPanel;
        Image1: TImage;
        Button2: TButton;
        Label11: TLabel;
        Label12: TLabel;
        Button3: TButton;
        procedure Button1Click(Sender: TObject);
        procedure Button2Click(Sender: TObject);
        procedure Button3Click(Sender: TObject);
      private
        { Private declarations }
      public
        { Public declarations }
      end;

    var
      Form1: TForm1;

    implementation

    {$R *.dfm}

    procedure TForm1.Button1Click(Sender: TObject);
    var c1,m1,c2,m2,c3,m3,t1,t2,t3:integer;
    begin
    c1:=StrtoInt(Edit1.Text);
    m1:=StrtoInt(Edit2.Text);
    c2:=StrtoInt(Edit3.Text);
    m2:=StrtoInt(Edit4.Text);
    t1:=c1*60+m1;
    t2:=c2*60+m2;
    t3:=t2-t1;
    c3:=t3 div 60;
    m3:=t3 mod 60;
    Edit5.Text:=InttoStr(c3);
    Edit6.Text:=InttoStr(m3);
    end;
```

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

```
        procedure TForm1.Button2Click(Sender: TObject);
begin
    Label11.Visible:=True;
    Label12.Visible:=True
    end;

    procedure TForm1.Button3Click(Sender: TObject);
    begin
    Close
    end;

    end.
```

Обратите внимание на то, что список использованных в программе элементов графического интерфейса, а также список использованных в программе процедур система программирования создает автоматически.

После запуска готовой программы на выполнение будут производиться все необходимые для расчетов действия, причем нажатие экранных кнопок можно производить в произвольном порядке и реакция на них будет правильной. Пример того, как может выглядеть экран программы во время работы, показан на рис.18.

8. Вещественные переменные. Программа «Цилиндр»

Во всех рассмотренных нами до сих пор задачах мы работали только с целочисленными переменными, значениями которых могут быть лишь целые числа. На практике же чаще всего полученные при расчетах значения выражаются дробными, вещественными числами. Вещественные результаты часто получаются и тогда, когда исходными данными при вычислениях являются целые числа. Пример подобной задачи – это вычисление объема и площади поверхности цилиндра по его высоте и радиусу основания. Даже в том случае, если высота и радиус основания являются целыми числами, полученные результаты заведомо будут вещественными, так как для их вычисления используется число «пи», которое, как известно, является бесконечной дробью.

На примере этой задачи освоим работу с вещественными переменными, а также научимся более рационально разрабатывать графический интерфейс программы. С создания интерфейса и начнем работу над программой. Интерфейс данной программы будет включать следующие элементы:

а) два текстовых окна для ввода исходных данных (высоты и радиуса цилиндра) и две поясняющие надписи к этим окнам;

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

б) два текстовых окна для вывода полученных результатов (объема и площади поверхности цилиндра) и две надписи с информацией о содержимом этих окон;

в) панель с рисунком, на котором должен быть изображен геометрический объект, параметры которого вычисляются в программе;

д) три экранные кнопки. Щелчок на первой из этих кнопок приводит к вычислению искомых величин. Щелчок на второй кнопке будет выводить на экран компьютера информацию об авторе программы. Щелчок на третьей кнопке закрывает программу;

е) дополнительное окно с информацией об авторе программы и кнопкой, которая закрывает это окно.

Как видно из перечисления элементов интерфейса, в целях экономии места на основной форме мы вводим дополнительную форму с информацией об авторе, которая может выводиться на экран и закрываться при отсутствии надобности в ней. Как создавать дополнительную форму, мы разберем позже, а сейчас рассмотрим процесс создания графического интерфейса основной формы.

Основная форма получит название «Цилиндр», для чего изменим свойство ***Caption*** объекта ***Form1***, цвет же формы в данном случае оставим без изменения. На форме помещаем текстовые окна ***Edit1*** и ***Edit2***, в которые будут вводиться данные соответственно о высоте цилиндра и радиусе его основания. Объекты ***Label1*** и ***Label2*** будут содержать поясняющие надписи к содержанию данных окон. Далее на форме размещаем объекты ***Edit3*** и ***Edit4***, которые содержат результаты вычислений. В окне ***Edit3*** будет выводиться информация о вычисленном объеме цилиндра, а в окне ***Edit4*** данные о найденной площади его поверхности. Объекты ***Label3*** и ***Label4,*** подобно предыдущим двум надписям, служат пояснениями к окнам с соответствующими номерами.

В нижней части экрана расположим три кнопки ***Button1, Button2*** и ***Button3***. При нажатии на кнопку ***Button1*** производятся вычисления объема и площади поверхности параллелепипеда по исходным данным, введенным в окна ***Edit1*** и ***Edit2***, а полученные результаты отображаются в окнах ***Edit3*** и ***Edit4***. При нажатии на вторую кнопку ***Button2*** на экран выводится дополнительное окно, содержащее сведения об авторах программы. При нажатии на кнопку ***Button3*** основная форма закрывается, и работа программы на этом завершается.

В правой части экрана программы должен находиться компонент ***Panel1***, на котором размещается объект ***Image1***, т. е. иллюстрация к данной программе, изображающая цилиндр. В начале с помощью графического редактора ***Paint*** создадим рисунок, котороый изображает цилиндр. Затем разместим этот рисунок в папке, содержащей проект «Цилиндр», а потом под размеры созданного рисунка будем подгонять размеры панели и иллюстрации. Для того чтобы разобраться в том, почему поступаем именно так, а не наоборот (как читатель конечно помнит, при разработке проекта «Параллелепипед» сначала

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

выводили на форму объекты ***Panel1*** и ***Image1***, а затем уже подгоняли рисунок под размеры этих объектов), нужно более детально разобраться в принципах создания компьютерной графики.

Существует два основных класса графических редакторов (программ, используемых для создания изображений на компьютерах). Первый класс – это растровые редакторы. Создаваемое с их помощью изображение представляет собой растр, т. е. совокупность пикселов – точек, являющихся минимальными элементами изображения. Такие изображения плохо поддаются масштабированию. При изменении геометрических размеров изображения оно может сильно искажаться. Особенно заметно искажение изображений бывает в том случае, если исходное изображение содержит кривые линии (как в нашем случае с изображением цилиндра). К растровым редакторам относится и графический редактор ***Paint***, в котором мы создаем вставляемый в текст рисунок.

Вторым основным классом графических редакторов являются векторные редакторы. Изображение, создаваемое в таких редакторах, состоит из отдельных элементов (как правило, – это различные кривые), каждый из которых задается определенной математической формулой. Изображения, созданные векторными редакторами, как правило, масштабируются без особых проблем. Рисунки, находящиеся в коллекции, входящей в состав пакета ***Microsoft Office*** как раз и являются векторными. Именно поэтому подгонка рисунка из коллекции ***Microsoft Office*** под объект ***Image1***, которая была произведена в проекте «Хронометр» и не вызвала у нас затруднений.

В проекте «Параллелепипед» было создано растровое изображение, которое подгоняли под размеры объектов ***Panel1*** и ***Image1***. Но так как сам по себе рисунок был достаточно простым и содержал только прямые линии, то существенных искажений его при масштабировании не происходило. В данном же проекте создаваемый посредством ***Paint*** растровый рисунок заведомо будет содержать кривые линии, что приведет к его искажению при масштабировании. Поэтому в данном случае более рациональным представляется следующий путь.

В начале в графическом редакторе создается изображение цилиндра (это изображение будет содержать файл «Рисунок», находящийся в папке проекта). Затем определяются размеры созданного изображения в пикселах. Для этого нужно открыть файл «Рисунок», в главном меню найти раздел «Рисунок» и в нем выполнить команду «Атрибуты». После выполнения этой команды откроется диалоговое окно, в котором можно увидеть ширину и высоту рисунка в пикселах. Для того рисунка, который приведен в книге, размеры равны 225 пикселов по горизонтали и 250 пикселов по вертикали. Такие же размеры будет иметь и объект ***Image1***. Что же касается объекта ***Panel1***, то нужно помнить, что на нем помимо иллюстрации должна размещаться фаска, которая занимает по 2 пиксела как сверху и снизу от иллюстрации, так и слева и справа от нее.

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

Поэтому величина объекта ***Panel1*** должна быть на 4 пиксела больше, чем у объекта ***Image1*** как по вертикали, так и по горизонтали. В данном случае эти размеры будут равны 229 по горизонтали и 254 по вертикали. Теперь, когда размеры объекта ***Panel*** уже известны, можно создать данный объект на форме, как это делалалось в предыдущих проектах. Затем на объект ***Panel1*** поместим объект ***Image1***, для которого укажем соответствующий рисунок, находящийся в файле.

На этом разработку графического интерфейса основной формы проекта можно считать завершенной. Графический интерфейс проекта приведен на рис. 19. Далее необходимо перейти к созданию кода, описывающего действия различных объектов формы. Однако, делая это, необходимо помнить, что величины, используемые в данной программе, являются вещественными. Вещественные же величины имеют определенные особенности как в их описании, так и в их использовании в программе.

Для описания вещественных величин используется специальный тип ***Real.*** Следовательно, описание вещественной переменной ***y*** в программе будет выглядеть следующим образом:

```
var y: real;
```

если в программе имеется несколько вещественных величин, то они описываются по тому же принципу, что и целые:

```
var y1,y2,y3:real;
```

если же в программе используются как целые, так и вещественные переменные, то служебное слово ***var*** употребляется только один раз, но разделы описания целых и вещественных переменных должны быть отделены друг от друга точкой с запятой. Пример описания подобного рода приведен ниже:

```
var a,b,c:integer;
x,y:real;
```

Подобно преобразованию строковых величин в целые и обратно нам придется производить подобные действия и с вещественными числами. Для преобразования строковой величины в вещественную используется стандартная функция языка ***Object Pascal StrtoFloat***. Для обратного преобразования используется стандартная функция ***FloattoStr***. Теперь мы готовы к написанию кода для данной программы.

Начнем написание программы с описания реакции на нажатие кнопки ***Button1***. Эта кнопка позволяет рассчитать объем и площадь поверхности цилиндра по имеющимся исходным данным. Обозначим высоту цилиндра буквой ***h,*** а радиус его основания буквой ***r***. Искомые величины обозначим следующим образом: объем цилиндра буквой ***v***, а общую площадь его

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

поверхности буквой ***s***. Все эти величины должны быть описаны в разделе описаний подпрограммы ***Tform1.Button1Click*** таким образом:

var h,r:integer; v,s:real;

то есть исходные данные описываются как величины целого типа, а результаты как вещественные величины.

Для того чтобы исходные данные были представлены в виде числовых величин, необходимо преобразовать в числовой (в данном случае – целый) вид, те данные, которые пользователь вводит в окно ***Edit1*** и ***Edit2*** в строковом виде. Это преобразование будет выполнено с помощью таких операторов:

h:=StrtoInt(Edit1.Text);
r:=StrtoInt(Edit2.Text);

Следующим шагом станет вычисление значений искомых величин. Как известно из стереометрии, объем цилиндра определяется как произведение высоты цилиндра на площадь его основания, которая, в свою очередь, вычисляется как площадь круга с радиусом ***r.*** Площадь же круга, как известно, представляет собой квадрат его радиуса, умноженный на число “пи”.

Для возведения радиуса в квадрат воспользуемся стандартной функцией ***sqr,*** имеющейся в языке ***Object Pascal***. Эта функция как раз и вычисляет квадрат аргумента, который должен быть заключен в скобки и указываться после названия функции. Таким образом, возведение величины ***r*** в квадрат можно записать следующим образом – ***sqr(r)***. Величину числа “пи” можно конечно записать цифрами. Чем больше цифр после запятой в данном числе укажет пользователь, тем точнее будет вычисленный результат. Но более рациональным является использование стандартной функции ***pi***, также входящей в состав языка ***Object Pascal***. Эта функция в отличие от предыдущей не имеет аргумента, а значением ее является определяемое компьютером с высокой степенью точности число «пи». Следовательно, площадь основания можно представить следущим образом – ***pi*sqr(r)***. Исходя из всего вышесказанного, на языке ***Object Pascal*** вычисление объема цилиндра можно записать следующим образом:

v:=pi*sqr(r)*h;

Следующая величина, которую нам необходимо определить – это площадь поверхности цилиндра. Общая площадь складывается из трех площадей – площади боковой поверхности цилиндра, площади его верхнего основания и площади нижнего основания, причем последние две величины равны между собой. Площадь одного основания, запишем как ***pi*sqr(r)***, а сумму площадей - как ***2*pi*sqr(r).*** Можно ту же сумму записать и по-другому – ***2*pi*r*r***. Площадь же боковой поверхности цилиндра вычисляется как

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

произведение длины окружности основания на высоту цилиндра. Длину окружности можно записать как ***2*pi*r***, а боковая поверхность цилиндра тогда определится по формуле ***2*pi*r*h***. Таким образом, общая площадь поверхности цилиндра будет равна ***2*pi*r*r+2*pi*r*h***. Вынося общие множители за скобки, получим следующее выражение: ***2*pi*r*(h+r).*** Теперь осталось присвоить значение этого выражения искомой величине ***s:***

```
s:=2*pi*r*(h+r);
```

Для того, чтобы преобразовать полученные вещественные величины в строковые, которые будут отображаться в текстовых окнах ***Edit3*** и ***Edit4,*** следует воспользоваться стандартной функцией ***FloattoStr.*** Операции преобразования будут записаны в программе следующим образом:

```
Edit3.Text:=FloattoStr(v);
Edit4.Text:=FloattoStr(s)
```

Тогда, целиком подпрограмма, описывающая действия, осуществляемые при нажатии кнопки ***Button1***, будет выглядеть следующим образом:

```
procedure TForm1.Button1Click(Sender: TObject);
var h,r:integer; v,s:real;
begin
h:=StrtoInt(Edit1.Text);
r:=StrtoInt(Edit2.Text);
v:=pi*sqr(r)*h;
s:=2*pi*r*(h+r);
Edit3.Text:=FloattoStr(v);
Edit4.Text:=FloattoStr(s)
end;
```

Следующая подпрограмма, которую мы создадим, должна описывать реакцию на нажатие кнопки ***Button2***. Результатом такого нажатия должен стать вывод дополнительного окна, которое содержит сведения об авторе или авторах данной программы. Для того, чтобы это правильно сделать, нужно знать последовательность действий, выполняемых при подключении к проекту дополнительного окна.

Создание и работа с дополнительным окном включает в себя следующие этапы:

1. Создание заготовки для нового окна. Такая заготовка представляет собой дополнительную форму. Для подключения к проекту новой формы необходимо в главном меню системы программирования открыть раздел “***File***”, в разделе “***File***” найти команду “***New***”. Если задержать указатель мыши на несколько секунд на данной команде, то справа от нее появится список

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

элементов, которые можно добавить в проект. Выберем из списка вариант "***Form***". После этого в проекте появится дополнительная форма. В нашем случае это будет "***Form2***".

2. В основном программном модуле (этот модуль как правило называется ***Unit1***) в подпрограмму, описывающую работу кнопки или другого элемента, открывающего дополнительное окно, нужно вставить оператор следующего вида:

FormN.Show;

где ***N*** – порядковый номер открываемого дополнительного окна. В нашем случае этот оператор будет выглядеть так:

Form2.Show;

целиком же подпрограмма, описывающая работу второй кнопки в нашем проекте, будет выглядеть следующим образом:

```
procedure TForm1.Button2Click(Sender: TObject);
begin
Form2.Show;
end;
```

3. Создание графического интерфейса подключаемого окна. Для того, чтобы вывести на экран необходимую форму, поступаем следующим образом: входим в главное меню системы программирования и выбираем в нем раздел "***View***". В этом разделе находим команду "***Forms…"*** После выполнения данной команду на экран компьютера выводится диалоговое окно, которое содержит список всех форм, используемых в данном проекте. В нашем случае поиск нужной формы не составит никаких затруднений, так как в проекте всего две формы. Выбираем вторую из них, после чего данная форма будет выведена на передний план. Сам же процесс создания графического интерфейса аналогичен соответствующему процессу для основной формы. По завершении работы над графическим интерфейсом дополнительное окно можно закрыть, щелкнув стандартную закрывающую кнопку, расположенную в правом верхнем углу окна.

4. Разработка кода, описывающего действия, производимые пользователем в дополнительном окне. Данный код будет содержаться в дополнительном модуле ***Unit2***. Для того, чтобы вывести на экран текст данного модуля, нужно открыть раздел "***View"*** главного меню системы программирования, в данном разделе найти команду "***Units…"*** После щелчка на данной команде открывается диалоговое окно со списком модулей, в котором выбираем нужный нам (в данном случае – это ***Unit2***).

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

Сам по себе процесс написания кода аналогичен соответствущему процессу для основного окна, но нужно учесть, что модуль ***Unit2***, должен быть подключен к основному модулю ***Unit1***. Данное подключение производится так: в основной части программы после заголовка указывается список используемых в данной программе дополнительных модулей. Этот список начинается со служебного слова ***Uses***, после которого через запятую перечисляются модули. В этот список следует добавить модуль ***Unit2***. Тогда список будет выглядеть следующим образом:

```
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ExtCtrls, StdCtrls, Menus, Unit2;
```

После выполнения всех вышеуказанных операций дополнительное окно становится полноценной частью проекта.

Следующая кнопка ***Button3***, которую используем в основной форме, закрывает эту форму. Данная операция производится с помощью команды «***Close***», а текст подпрограммы, описывающей это действие, будет выглядеть так:

```
procedure TForm1.Button3Click(Sender: TObject);
begin
Close
end;
```

На этом завершаем написание программного модуля ***Unit1*** для основной формы. Целиком данный модуль будет выглядеть следующим образом:

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ExtCtrls, StdCtrls, Menus, Unit2;

type
  TForm1 = class(TForm)
    Label1: TLabel;
    Edit1: TEdit;
    Label2: TLabel;
    Edit2: TEdit;
    Label3: TLabel;
```

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

```
    Edit3: TEdit;
    Label4: TLabel;
    Edit4: TEdit;
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    Panel1: TPanel;
    Image1: TImage;
    procedure Button1Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
var h,r:integer; v,s:real;
begin
h:=StrtoInt(Edit1.Text);
r:=StrtoInt(Edit2.Text);
v:=pi*sqr(r)*h;
s:=2*pi*r*(h+r);
Edit3.Text:=FloattoStr(v);
Edit4.Text:=FloattoStr(s)
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
Form2.Show;
end;

procedure TForm1.Button3Click(Sender: TObject);
begin
Close
end;
 end.
```

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

Теперь осталось только создать графический интерфейс и написать код для вспомогательного окна со сведениями об авторе проекта. В начале переименуем саму форму ***Form2***, изменив значение ее свойства ***Caption*** на «Сведения об авторе». Цвет формы мы изменять не будем, на форме же создадим следующие объекты: две надписи ***Label1*** и ***Label2*** со сведениями об авторе и закрывающую кнопку ***Button1***. Единственный объект, выполняющий действия в данном окне – это кнопка ***Button1***. Для нее мы и напишем код, состоящий из одного оператора «***Close***». Целиком же программный модуль ***Unit2*** будет выглядеть следующим образом:

```
unit Unit2;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs, StdCtrls;

type
  TForm2 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Button1: TButton;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form2: TForm2;

implementation

{$R *.dfm}

procedure TForm2.Button1Click(Sender: TObject);
begin
Close
end;

end.
```

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

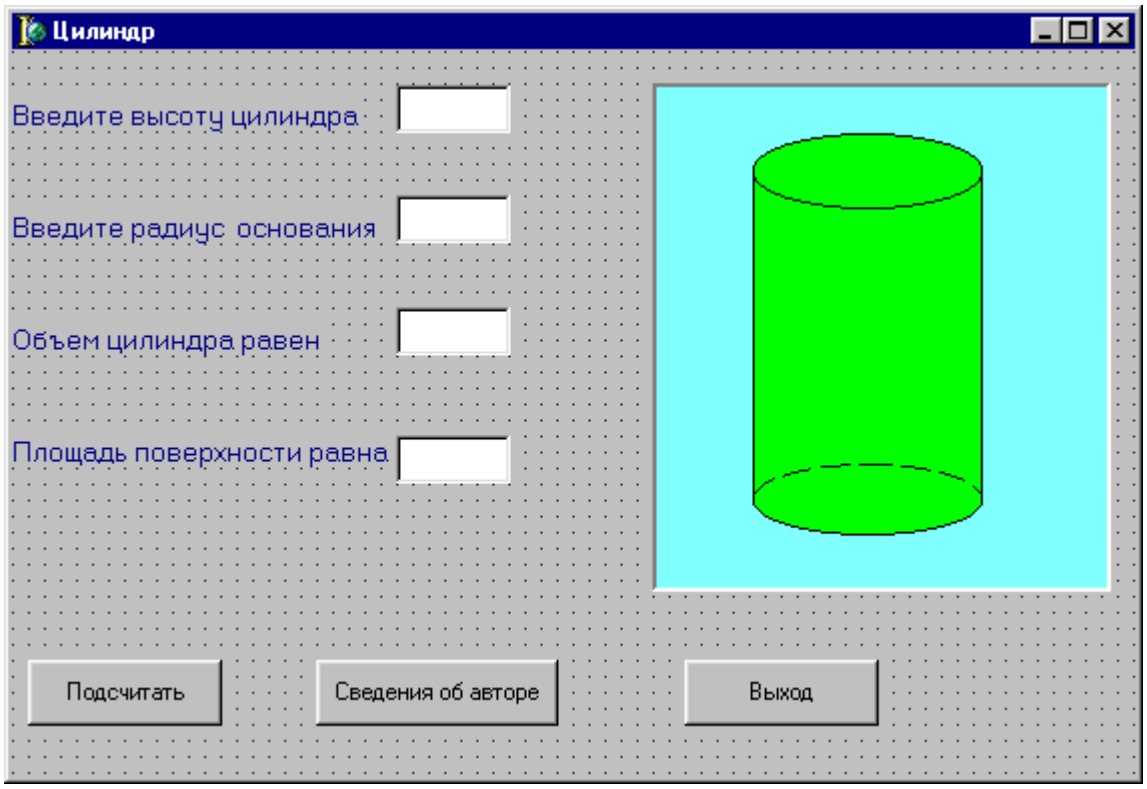

Рис. 19. Графический интерфейс проекта «Цилиндр»

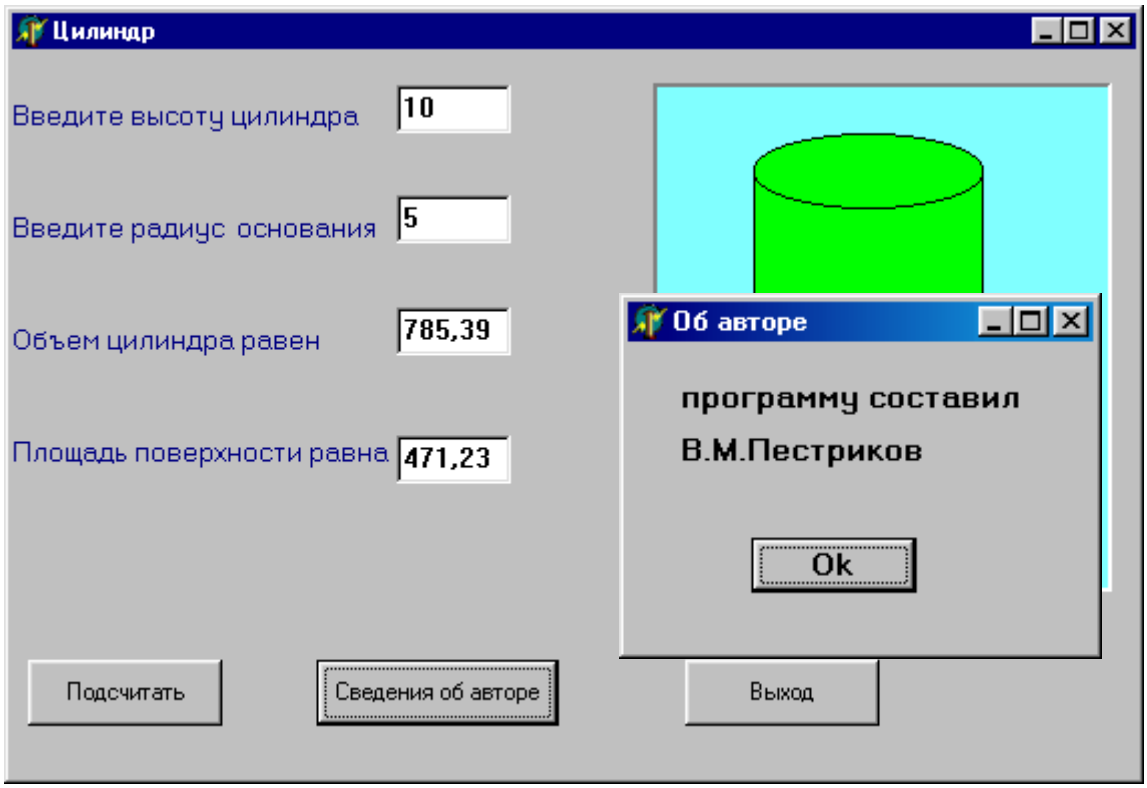

Рис. 20. Программа «Цилиндр» во время работы

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

После написания кода для второго окна работу над проектом завершена. Результаты работы программы можно увидеть на рис. 20.

9. Условный оператор. Программа «Тест»

Все программы, которые до сих пор были разработаны в среде ***Delphi,*** состояли из операторов, которые последовательно выполняются друг за другом. Такие программы называются программами линейной структуры, так как ход решения задачи представляет собой как бы прямую линию, ведущую от постановки задачи и ввода исходных данных к получению конечного результата.

Однако, это не единственно возможный вид программы. При решении задачи, поставленной перед программистом, нередко возникает ситуация, когда нужно выбирать один из двух или нескольких возможных вариантов дальнейшего хода программы. В этом случае в программе используется структура, называемая ветвлением. В программах с ветвлением в зависимости от значений вводимых исходных данных может работать одна из двух или нескольких ветвей (вариантов) программы. Такая структура реализуется в программе посредством условных операторов. В заголовке условного оператора содержится некоторое условие, выполнение или невыполнение которого и определяет дальнейшую работу программы. Так как программист заранее не знает, какой из вариантов будет выполняться, выбор должен производить сам компьютер.

В качестве условий, истинность или ложность которых проверяется, чаще всего используются знакомые читателю по школьному курсу математики операции сравнения различных величин. В языке ***Object Pascal*** используется 6 таких операций:

< - меньше > - больше = - равно

< = - меньше или равно > = - больше или равно < > - не равно

Написание последних 3 операций отличается от принятого в математике, где соответственно используются обозначения $\leq$, $\geq$ и $\neq$, но так как клавиш с подобными обозначениями нет на клавиатуре компьютера, то используются вышеуказанные сочетания из двух символов.

Основным условным оператором, используемым в языке ***Object Pascal***, является условный оператор ***if.*** Общий вид оператора ***if*** следующий:

if <условие >
then <вариант 1 >
else <вариант 2 >;

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

служебные слова ***if***, ***then*** и ***else*** в переводе с английского означают соответственно: если, то, иначе. Оператор действует следующим образом: сначала проверяется, выполняется ли условие, находящееся после слова ***if***. Если это условие выполняется (иначе говоря, является истинным), то осуществляется первый вариант действий, в противном случае, если условие не выполняется (является ложным) – второй, записанный после служебного слова else. В самом простом случае действие, осуществляемое в каждом из вариантов, состоит из одного оператора.

В качестве примера использования оператора ***if*** рассмотрим фрагмент программы на языке ***Object Pascal***, в котором определяется значение какой из двух переменных ***a*** или ***b*** является максимальным.

```
if a>b
then max:=a
else max:=b;
```

Значение максимального из двух переменных должно быть присвоено третьей переменной с именем ***max***. Если указанное в заголовке условие истинно, то выполняется тот вариант, который указан после ***then***, то есть переменной ***max*** присваивается значение, равное значению переменной ***a***. Если же условие ложно, т. е. на самом деле ***b*** больше, чем ***a***, то работает тот вариант, который находится после ***else***, иначе говоря, переменной ***max*** присваивается значение ***b***. В любом варианте переменной ***max*** будет присвоено значение той из двух переменных, которое действительно является максимальным.

Тот вид условного оператора, который был описан нами выше, представляет собой полную форму условного оператора, но такая форма его записи не является единственно возможной. Наряду с ней в языке Паскаль используется и сокращенный условный оператор. Такой оператор имеет следующий общий вид:

```
if <условие > then <действие >;
```

Сокращенный условный оператор работает следующим образом. Если условие, содержащееся после служебного слова ***if***, истинно, то выполняется действие, записанное после ***then***, а если условие ложно, то в условном операторе не выполняется никаких действий, и программа переходит к выполнению следующего оператора, расположенного вслед за данным условным оператором.

В качестве примера применения сокращенного условного операторa рассмотрим фрагмент программы, которая определяет максимальное из трех чисел ***a,b*** и ***c***. Найденное максимальное в данной тройке число присваивается переменной с именем ***m3***. Данный фрагмент программы выглядит следующим образом:

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

```
 m3:=a;
if b>m3 then m3:=b;
if c>m3 then m3:=c;
```

В приведенном примере переменной ***m3*** вначале присваивается значение переменной ***a.*** Затем значение присвоенное переменной значение сравнивается со значением переменной ***b***. Если значение переменной ***b*** больше чем значение ***m3*** (и соответственно ***a***), то ***m3*** присваивается значение ***b*** – большей из двух переменных. Если же ***b*** меньше, чем ***m3*** (и соответственно меньше, чем ***a***) или равно ***m3***, то условный оператор ***if*** не производит никаких действий. В любом случае после выполнения условного оператора переменная *m3* содержит значение большей из двух переменных ***a*** и ***b.***

Второй условный оператор производит сравнение значение ***m3*** со значением переменной ***c.*** Если выясняется, что ***c*** больше, чем ***m3,*** то ***m3*** присваивается новое значение – ***c*** (такое присваивание будет иметь место в том случае, если ***c*** будет больше чем ***a*** и больше, чем ***b***, т. е. является максимальным из 3 переменных). Если же значение ***c*** будет меньшим, чем ***m3***, то никакого присваивания не происходит и значение ***m3*** остается неизменным. В результате описанных действий переменная ***m3*** действительно будет содержать значение максимального из трех переменных.

Условный оператор ***if*** удобно использовать для составления различных программ-тестов, проверяющих знания пользователя. В качестве примера рассмотрим составление программы, проверяющей, знает ли пользователь дату запуска первого искусственного спутника Земли. В случае правильного ответа на экран выводится сообщение «Вы ответили правильно», в противном случае сообщение «Вы ошиблись».

Графический интерфейс основной формы данной программы будет включать в себя следующие компоненты: во-первых, окно комментария, содержащее вопрос, на который должен дать ответ тестируемый. Во-вторых, – текстовое окно, куда тестируемый вводит свой вариант ответа. Слева от текстового окна должно находиться еще одно, вспомогательное окно комментария, поясняющее, куда нужно вводить ответ. Справа от текстового окна поместим иллюстрацию к тесту, представляющую собой фотографию первого искусственного спутника Земли. Под текстовым окном должна располагаться экранная кнопка, нажатием на которую вводится ответ тестируемого пользователя. Ниже экранной кнопки должна быть расположена еще одна надпись, которая сообщает пользователю о том, правильно или неправильно ответил он на заданный вопрос.

Помимо кнопки «Ввод» на основной форме будут находиться еще две экранные кнопки – кнопка, закрывающая окно программы и кнопка, нажатие на которую выводит на экран компьютера дополнительное окно со сведениями об авторе. Это дополнительное окно, в свою очередь, должно содержать окно

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

комментария со сведениями об авторах или авторе программы и экранную кнопку, закрывающую это дополнительное окно.

Разработку графического интерфейса мы начнем как обычно с работы над самой формой. Изменяем заголовок формы – она будет называться «Тест». Цвет формы изменяем на небесно-голубой. Размеры формы оставляем без изменений. В левой верхней части формы располагаем объект ***Memo1*** – окно комментария с текстом задаваемого вопроса. Текст вопроса «В каком году был запущен первый искусственный спутник Земли» вводим в поле комментария в три строки, а размер шрифта (используя свойство ***Font***) увеличиваем до 12.

Под окном комментария размещаем текстовое окно ***Edit1***. В него будет вводиться текст ответа. Для удобства контроля пользователя за вводимой им информацией размер текста в окне также увеличиваем до 12. Слева от объекта ***Edit1*** помещаем вспомогательное окно комментария ***Memo2***, содержащее пояснение о том, куда нужно вводить ответ. В этом окне информацию также вводим в три строчки. Для того чтобы текст поместился в указанном месте размер шрифта принимаем равным 9. Цвет текста в этом окне изменяем на темно-синий.

Ниже окна ***Edit1*** помещаем экранную кнопку ***Button1***. Заголовок этой кнопки изменяем на слово «Ввод», а размер шрифта делаем равным 12. В правом нижнем углу основной формы будет расположена надпись ***Label1***, информирующая пользователя о результатах теста. Эта надпись будет выводиться в одну строку шрифтом красного цвета. Начертание шрифта сделаем полужирным, а размер также примем равным 12. Сразу после начала работы программы эта надпись вообще не должна быть видна, так как было бы абсурдно, если бы пользователь узнал о результатах теста еще до его начала. Можно было бы найти свойство ***Visible*** объекта и придать ему значение ***False***. Но такой путь усложнил бы написание кода. Поэтому поступим иначе: найдем свойство ***Caption*** объекта ***Label1*** и очистим его. Тогда надпись после запуска программы не будет видна, хотя формально она будет считаться видимой.

Следующим объектом должна стать иллюстрация к тексту. Как мы это уже неоднократно делали раньше, в начале размещаем на форме объект ***Panel1***, а уже затем поверх него располагаем ***Image1***, т. е. непосредственно саму иллюстрацию. Иллюстрацию к данной программе, представляющую собой черно-белую фотографию первого спутника, авторы книги взяли из Интернета. Данная фотография находилась на Web-странице в графическом формате ***jpeg***. Графические изображения, созданные в данном формате, достаточно хорошо поддаются масштабированию, поэтому особых проблем, связанных с взаимной подгонкой размеров панели и самой иллюстрации, здесь возникать не должно.

Нам осталось разобраться еще с двумя элементами графического интерфейса – экранными кнопками ***Button2*** и ***Button3***. Основные настройки для этих кнопок – такие же, как и для кнопки ***Button1*.** Отличаться будут только

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

заголовки кнопок – «Об авторах» для ***Button2*** и «Выход» для ***Button3***. Нажатие кнопки ***Button2*** должно приводить к появлению на экране компьютера дополнительной формы с именем ***Form2***. Последовательность создания такой формы и подключения ее к основному проекту была описана в предыдущей главе и поэтому не должна вызывать сложностей у пользователя.

Настройка дополнительной формы будет заключаться в изменении ее заголовка на «Сведения об авторах», а также уменьшении ее размеров, так как на ней будут расположены всего два объекта. Уменьшение размеров формы можно производить как путем перетаскивания мышью маркеров формы, так и посредством задания соответствующих величин для свойств ***Height*** и ***Width.*** Графические объекты, которые будут расположены на вспомогательной форме – это окно комментария ***Memo1***, которое непосредственно и будет содержать информацию об авторах программы, а также кнопка ***Button1***, закрывающая вспомогательное окно. Текст комментария будет содержать надпись в две строки, имеющую размер шрифта 10, а закрывающая кнопка должна иметь заголовок «***Ok***» и иметь размер шрифта 12. Как должен выглядеть созданный нами графический интерфейс можно увидеть на рис. 21.

Далее переходим к написанию кода программы. Начнем с написания кода для основной программной формы. В первую очередь оформим процедуру, которая описывает реакцию на щелчок на кнопке ***Button1***. После щелчка на этой кнопке должна производиться проверка данных, введенных пользователем. Строковая информация преобразуется в числовую с помощью стандартной функции ***StrtoInt.*** Введенная в окно ***Edit1*** строковая информация присваивается целочисленной переменной ***k***:

```
k:=StrtoInt(Edit1.Text);
```

Теперь ответ пользователя, содержащийся в переменной ***k***, нужно проверить на совпадение с правильным ответом – числом 1957 (год запуска первого спутника). При совпадении двух этих ответов на экран выводится одна информация, при их несовпадении другая - об ошибке. Такая проверка выполняется с помощью условного оператора ***if***:

```
if k=1957
then Label1.Caption:='Вы ответили правильно'
else Label1.Caption:='Вы ошиблись';
```

Таким образом, целиком процедура, отслеживающая реакцию на нажатие кнопки «Ввод» после ввода ответа в текстовое окно, должна выглядеть так:

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

Рис. 21. Графический интерфейс программы «Тест»

Рис. 22. Программа «Тест» во время работы

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

```
 procedure TForm1.Button1Click(Sender: TObject);
var k:integer;
begin
k:=StrtoInt(Edit1.Text);
if k=1957
then Label1.Caption:='Вы ответили правильно'
else Label1.Caption:='Вы ошиблись'
end;
```

Следующая кнопка, для нажатия которой мы пишем код – это кнопка ***Button2***. В процедуру, описывающую реакцию на нажатие этой кнопки, вставляем всего одну команду:

```
 Form2.Show;
```

При этом нужно не забыть, что в раздел ***Uses***, находящийся в начале данного программного модуля, следует добавить ссылку на второй программный модуль, обслуживающий дополнительную форму ***Form2***:

```
uses Unit2;
```

Наконец, в процедуру, обслуживающую кнопку ***Button3***, нужно вставить всего одно слово:

```
Close;
```

Теперь можно целиком рассмотреть получившийся программный модуль ***Unit1***, содержащий вышеуказанные элементы программного кода:

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs, StdCtrls, jpeg, ExtCtrls;

type
  TForm1 = class(TForm)
    Memo1: TMemo;
    Memo2: TMemo;
    Edit1: TEdit;
    Button1: TButton;
    Label1: TLabel;
    Panel1: TPanel;
    Image1: TImage;
    Button2: TButton;
```

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

```
    Button3: TButton;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

uses Unit2;

{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
var k:integer;
begin
k:=StrtoInt(Edit1.Text);
if k=1957
then Label1.Caption:='Вы ответили правильно'
else Label1.Caption:='Вы ошиблись'
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
Form2.Show
end;

procedure TForm1.Button3Click(Sender: TObject);
begin
Close
end;

end.
```

Последнее, что осталось сделать в плане написания кода - это описать реакцию на нажатие кнопки ***Button1***, расположенной на вспомогательной форме ***Form2***. Для этого нужно в списке программных модулей выделить модуль ***Unit2*** и в процедуру ***TForm2.Button1Click*** добавить команду ***Close***. Полный текст модуля ***Unit2*** приводить не будем, так как он аналогичен такому

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

же модулю в предыдущем проекте. Внешний вид программы «Тест» во время работы приведен на рис. 22.

10. Программа «Квадратное уравнение»

В примерах, приведенных в предыдущей главе, каждая из ветвей условного оператора ***if*** содержала только один оператор. Но часто возникает ситуация, когда при выполнении или невыполнении некоторого условия нужно произвести не одно какое-либо действие, а целый ряд действий. Соответственно внутри ветви условного оператора нужно поместить не один, а несколько операторов. Такая группа операторов называется составным оператором. Составной оператор, находящийся внутри одной из ветвей условного оператора, заключается в операторные скобки, роль которых выполняют служебные слова ***begin*** и ***end.*** Внутри составного оператора простые операторы отделяются друг от друга точками с запятой. Приведем фрагмент программы, где внутри условного оператора используется составной оператор.

В этом фрагменте производятся следующие действия: проверяется является ли значение переменной ***r*** неотрицательным числом. Переменная ***r*** интерпретируется в данном случае как радиус некоторой окружности, для которой требуется определить ее длину, а также вычислить площадь круга, ограниченного данной окружностью. Естественно, что вычисление длины окружности и площади круга будет иметь смысл только в том случае, если радиус окружности - неотрицательная величина. Поэтому в данной программе мы используем сокращенный условный оператор, который в случае, если условие $r \geq 0$ выполняется, вычисляет площадь круга и длину окружности, а в том случае, если это условие не выполняется, не производит никаких действий. Выглядеть этот фрагмент программы будет следующим образом:

```
if r>=0
then
        begin
        l:=2*pi*r;
        s:=pi*sqr(r)
        end;
```

В данном фрагменте программы используются еще 2 переменные: ***l*** – длина окружности и ***s*** – площадь круга. В случае выполнения условия, находящегося в заголовке условного оператора, выполняются сразу два действия, которые оформлены в виде составного оператора. Этот составной оператор находится после служебного слова ***then*** и состоит из двух простых операторов. В первом из этих операторов производится вычисление длины

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

окружности для заданного радиуса, а во втором – вычисление площади круга. После первого оператора ставится точка с запятой для отделения его от второго. В операторах используются уже знакомые читателю стандартные функции языка Паскаль – ***pi*** и ***sqr***. Служебные слова ***begin*** и ***end***, начинающие и завершающие составной оператор, сами отдельными операторами не являются. Так как в том случае, если радиус отрицателен, дальнейшие вычисления лишены смысла, то в данном условном операторе ветвь, начинающаяся со слова ***else***, отсутствует.

В общем же случае составной оператор может находиться и после служебного слова ***then*** и после служебного слова ***else*** и после каждого из этих служебных слов. Следующей задачей как раз и будет разработка проекта, в котором реализуется как раз последний вариант условного оператора. Результатом разработки должна стать программа решения квадратного уравнения вида:

$$ax^2+bx+c=0.$$

По введенным с клавиатуры исходным данным программа должна либо находить корни квадратного уравнения, либо выдавать сообщение о том, что данное уравнение не имеет решения. Напомним читателю алгоритм решения данного уравнения. В начале следует вычислить дискриминант данного уравнения, который определяется по формуле:

$$d=b^2-4ac\ .$$

Затем в зависимости от величины дискриминанта определяется возможность нахождения решения данного уравнения. Если дискриминант положителен или равен нулю, то уравнение имеет решение и можно найти два его корня ***x1*** и ***x2*** (в случае равенства дискриминанта нулю два корня будут равны между собой). Вычисление корней производится по следующим формулам:

$$x1=\frac{-b-\sqrt{d}}{2a}, \qquad x2=\frac{-b+\sqrt{d}}{2a}$$

В случае же, если найденный дискриминант отрицателен, уравнение вообще не имеет решения. Как видно из вышесказанного, данный алгоритм как раз и подходит для реализации с помощью условного оператора ***if***.

Как всегда, работу над проектом начинаем с разработки интерфейса программы. Основное окно программы должно содержать три текстовых окна для ввода исходных данных – коэффициентов ***a,b*** и ***c*** квадратного уравнения. Слева от каждого из этих окон должна находиться надпись, которая поясняет, какой именно коэффициент вводится в то или иное окно. Таким образом,

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

количество надписей в этой части основного окна также должно быть равно трем. Ниже текстовых окон для ввода исходных данных должны находиться два текстовых окна для вывода решения (конечно, в том случае, если такое решение имеется). Над этими текстовыми окнами также должна располагаться поясняющая надпись.

Между окнами с исходными данными и окнами с результатами на основной форме будут расположены две экранные кнопки. Слева должна находиться кнопка, нажатие на которую приводит к решению уравнения. Назовем эту кнопку «Найти решение». Правее будет расположена кнопка «Сброс», щелчок на которой очищает как текстовые окна с исходными данными, так и текстовые окна с результатами. Эта кнопка имеет двоякое предназначение. Во-первых, она может понадобиться в том случае, если пользователь допустил ошибку при вводе исходных данных. Пользователь может исправить ошибку, не закрывая окно программы, а только щелкнув мышью экранную кнопку. Во-вторых, если пользователь уже решил квадратное уравнение и хочет решить аналогичное уравнение, но уже при других значениях коэффициентов, ему также нет необходимости закрывать окно программы и запускать ее заново. Достаточно нажать кнопку «Сброс» и ввести затем эти новые коэффициенты.

Еще одним элементом основного окна программы, используемым для того, чтобы немного оживить его, станет иллюстрация юмористического характера, расположенная в правой части окна. Данная иллюстрация изображает персонажа, задумавшегося над решением сложной задачи.

Все перечисленные выше элементы уже неоднократно использовались нами при составленни различных проектов. Но наряду с ними в этой программе будут использованы и новые элементы. Необходимость их введения объясняется следующим: для закрытия окна и для вывода справочной информации мы ранее использовали специальные экранные кнопки. Это допустимо в том случае, если число элементов интерфейса в окне невелико и, кроме того, в окне имеется достаточное количество свободного места для размещения кнопок. Если же в основном окне программы находится большое количество элементов интерфейса и к тому же ощущается дефицит свободного места в окне, то целесообразно поступить по-другому. В таком случае лучше организовать в верхней части окна программы специальное меню, с помощью пунктов которого можно производить различные операции с файлами (включая закрытие программы), получать необходимую информацию справочного характера и решать многие другие задачи. В системе программирования ***Delphi*** имеются все необходимые средства для создания такого меню.

Как обычно, непосредственную работу по созданию интерфейса программы начнем с работы над самой формой. Изменим заголовок основной формы. Вместо ***Form1*** она будет называться «Решение квадратного уравнения». Кроме того, изменим цвет формы, сделав его бледно-зеленым.

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

Затем приступим к созданию текстовых окон для ввода исходных данных с сопровождающими их надписями. Текстовые окна для ввода данных получат наименования ***Edit1***, ***Edit2*** и ***Edit3***. Соответствующие надписи получат имена ***Label1***, ***Label2*** и ***Label3***. В окно ***Edit1*** будут вводиться данные о коэффициенте ***a***. Поле справа от свойства ***Text*** данного объекта предварительно очистим, так как в начале работы программы текстовое окно должно быть пустым. Шрифт же для данного объекта (свойство ***Font***) мы изменим, сделав шрифт полужирным и увеличив его размер до 10 кегля. Аналогичным образом настроим окна ***Edit2*** и ***Edit3***, которые используются для ввода коэффициентов ***b*** и ***c***.

Настройка надписи ***Label1*** включает в себя следующие действия. Во-первых, изменим заголовок надписи (свойство ***Caption***) на следующий: «Введите коэффициент ***a***». Во-вторых, изменим шрифт надписи. Подобно шрифту текстового окна сделаем шрифт полужирным и равным 10 кеглю. Подобные же настройки произведем и для объектов ***Label2*** и ***Label3.*** При этом содержание заголовка для ***Label2*** будет «Введите коэффициент ***b***», а для ***Label3*** соответственно «Введите коэффициент ***c***».

Следующими элементами интерфейса программы станут текстовые окна ***Edit4*** и ***Edit5***, которые используются для вывода корней уравнения ***x1*** и ***x2***. Настройка этих объектов большей частью аналогична настройке предыдущих текстовых окон, т. е. мы очищаем свойство ***Text***, а шрифт для обоих текстовых окон делаем полужирным и равным по величине 10 пунктам.

Есть, однако, еще момент, который надо учитывать при настройке этих окон. Дело в том, что уравнение, как мы уже знаем, может и не иметь решения. В таком случае наличие в окне программы текстовых окон с корнями уравнения лишено какого–либо смысла. Поэтому поступим с этими окнами следующим образом: изначально сделаем их невидимыми, а в том случае, если решение уравнения будет найдено, эти окна проявятся и мы увидим в них найденный результат. Для того чтобы настроить окна соответствующим образом, мы найдем для каждого из окон ***Edit4*** и ***Edit5*** свойство ***Visible*** и из раскрывающегося списка значений этого свойства выберем значение ***False*** (ложно), которое скроет их от глаз пользователя. На этом настройку текстовых окон можно считать завершенной.

Надпись ***Label4*** должна содержать информацию об итогах решения уравнения (как при наличии корней, так и при их отсутствии). Поскольку эта надпись в любом случае должна появляться только после ввода исходных данных и щелчке на кнопке «Найти решение», сделаем ее изначально невидимой, для чего свойству ***Visible*** данного объекта дадим значение ***False***. Заголовок данной надписи изменим на «Корни уравнения», а шрифт заголовка сделаем полужирным и равным по величине 10 пунктам. При отсутствии решения заголовок надписи будет иным, но соответствующее изменение заголовка мы обеспечим при написании кода.

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

Между верхней и нижней группой надписей расположим две экранные кнопки ***Button1*** и ***Button2.*** Для кнопки ***Button1*** мы изменим заголовок, который будет теперь следующим: «Найти решение». Размеры кнопки увеличим таким образом, чтобы данный заголовок был полностью виден. Шрифт заголовка сделаем полужирным и равным 12 пунктам. Аналогичные настройки шрифта произведем и для расположенной правее кнопки ***Button2***. Заголовок же этой кнопки будет состоять из одного слова: «Сброс».

В правой части окна поместим юмористическую иллюстрацию, которая изображает радость мультипликационного персонажа, нашедшего решение поставленной перед ним задачи. Для размещения иллюстрации мы используем, как обычно, объект ***Panel1***, на котором будет помещен объект ***Image1***, то есть сама иллюстрация. Картинку мы возьмем из коллекции рисунков, входящих в состав пакета ***Microsoft Office*** 97. К данной коллекции мы уже неоднократно обращались за сюжетами для иллюстраций. Искомая иллюстрация находится, как читатель уже знает, по адресу ***Program Files\Clipart\ Popular***. Сам же файл с данной иллюстрацией называется ***Amconfus*** и имеет расширение ***wmf***. Иллюстрация, содержащаяся в таком файле, хорошо поддается масштабированию и, поэтому не требуется производить особо тщательную взаимную подгонку размеров панели и соответствующей ей иллюстрации.

Последним объектом, который осталось разместить на основной форме проекта, является главное меню программы. Этот новый для нас объект имеет более сложную структуру, нежели уже известные нам объекты. Меню может включать в свой состав ряд разделов, каждый из которых в свою очередь может состоять из одного или нескольких пунктов. Каждый из разделов и пунктов меню является особым объектом интерфейса, входящим в состав другого объекта – самого меню.

Если Вы посмотрите на список объектов, входящих в состав проекта (он находится в окне ***Object TreeView***), то увидите, что рядом с некоторыми объектами находится значок в виде плюса. Такой значок говорит о том, что данный объект является составным и этот значок можно увидеть, в частности, рядом с объектом ***MainMenu1***(главное меню 1). Если щелкнуть по значку «плюс», то можно увидеть список объектов, содержащихся в составном объекте. При этом значок «плюс» изменяется на «минус». Если щелкнуть на значке «минус», то составной объект снова свернется и «минус» при этом сменится на «плюс». Объекты, входящие в составной объект, сами в свою очередь, тоже могут быть составными. На рис. 23 показан пример того, как может выглядеть список объектов проекта, где некоторые объекты являются составными.

Порядок создания главного меню программы следующий: в начале на панели компонентов на вкладке ***Standard*** нужно найти компонент ***MainMenu***, который выглядит таким образом -  . Выделив данный компонент, щелкаем мышью в верхней части формы, в результате чего такой же значок появляется

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

на форме, т. е. получаем заготовку для создания будущего меню. Для заполнения меню разделами и пунктами необходимо дважды щелкнуть значок  (расположенный на форме), в результате чего на экране компьютера появляется диалоговое окно ***Form1.MainMenu1***. Это диалоговое окно используется для создания разделов и пунктов будущего меню (рис. 24).

Для того, чтобы создать первый раздел меню, достаточно щелкнуть в окне инспектора объектов справа от свойства ***Caption***, затем ввести название раздела (например слово «Файл») и подтвердить ввод нажатием клавиши ***Enter***. Если теперь посмотреть на содержимое диалогового окна ***Form1.MainMenu1***, то в нем можно увидеть появившийся пункт с именем «Файл». Если же посмотреть теперь список объектов, то можно убедиться в том, что в проекте появился новый объект с именем ***N1***. Это и есть созданный нами раздел меню. Справа от раздела «Файл» появилась заготовка для создания следующего раздела меню. Выделяем эту заготовку, затем снова щелкаем свойство ***Caption***, вводим название нового раздела (на этот раз название раздела – «Справка») и получаем еще один раздел меню с именем ***N2***. Теперь необходимо дополнить существующие разделы соответствующими пунктами.

Для того чтобы добавить пункт в раздел «Файл», нужно выделить этот раздел и затем щелкнуть мышью заготовку, расположенную снизу от слова «Файл». Затем снова щелкаем то же свойство Caption, вводим имя раздела – «Выход» и получаем в проекте еще один объект с именем ***N3***. Как видно, нумерация объектов продолжается в порядке их создания. Это связано с тем, что система программирования воспринимает и разделы меню и содержащиеся в разделах пункты как объекты одного класса. Аналогичным образом в разделе меню «Справка» создаем пункт «Об авторах». Этот пункт становится объектом по имени ***N4***. Закрываем диалоговое окно ***Form1.MainMenu1*** и на этом создание графического интерфейса основной формы программы «Решение квадратного уравнения» завершается. Создание вспомогательной формы, содержащей сведения об авторах проекта, производится аналогично тому, как мы это делали в предыдущих проектах. Получившийся в результате нашей работы графический интерфейс основной формы изображен на рис. 25.

Написание кода для этой программы начинаем с описания реакции на нажатие экранной кнопки «Найти решение» (кнопка ***Button1***). Эта реакция записывается в процедуре ***TForm1.Button1Click***. Работу над процедурой начнем с того, что опишем используемые в ней переменные. Для решения поставленной задачи будут необходимы переменные для коэффициентов ***a, b*** и ***c***, для дискриминанта – ***d,*** и для хранения полученных корней уравнения – ***x1*** и ***x2***. Как исходные данные, так и результаты могут быть вещественными величинами, поэтому для их описания используем тип ***real***:

```
var a,b,c,d,x1,x2:real;
```

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

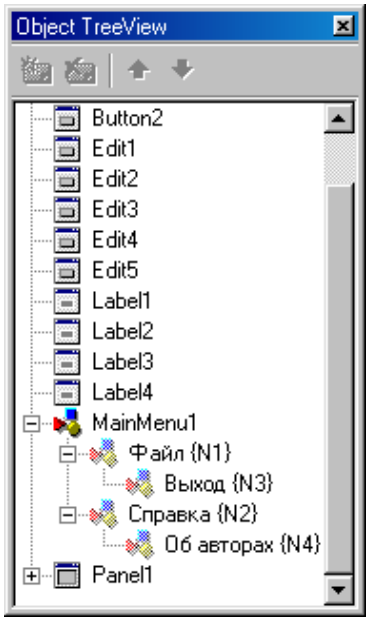

Рис. 23. Список объектов проекта, содержащий составные объекты MainMenu1 и Panel1 (составной объект MainMenu1 представлен в развернутом виде)

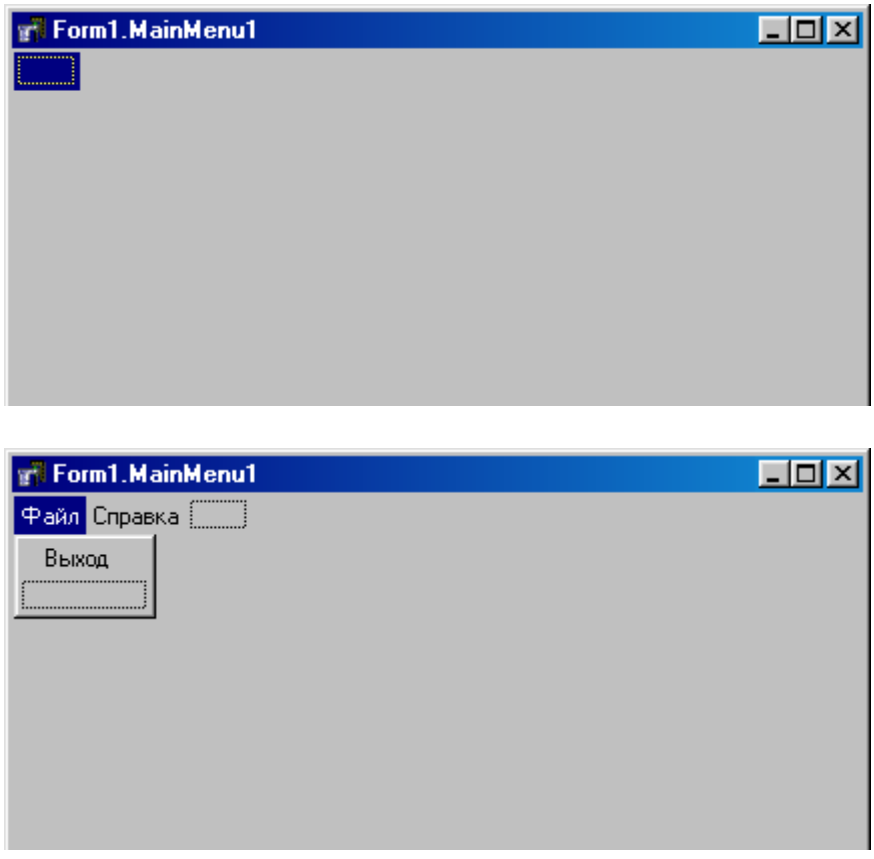

Рис. 24. Диалоговое окно создания главного меню (вверху представлено диалоговое окно сразу после его открытия, внизу – окно по окончании разработки меню)

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

Далее в основной части процедуры необходимо произвести преобразование вводимых в текстовые окна исходных данных – строковых величин в соответствующие им числовые величины вещественного типа. Это преобразование производится с помощью знакомой нам стандартной функции ***StrtoFloat***:

```
a:=StrtoFloat(Edit1.Text);
b:=StrtoFloat(Edit2.Text);
c:=StrtoFloat(Edit3.Text);
```

Следующим шагом в решении задачи является вычисление дискриминанта с помощью полученных в результате преобразования вещественных величин:

```
d:=sqr(b)-4*a*c;
```

Дальнейший ход решения задачи зависит от вычисленного значения дискриминанта. Если дискриминант неотрицателен, то необходимо произвести следующие действия:

а) вычислить по известным формулам корни уравнения;

б) сделать видимыми текстовые окна ***Edit4*** и ***Edit5***, в которых будут выведены вычисленные корни;

в) преобразовать полученные вещественные корни с помощью стандартной функции ***FloattoStr*** в текстовые величины, годные для вывода в текстовых окнах;

г) вывести надпись, поясняющую полученные результаты, предварительно сделав ее видимой.

Вся перечисленная последовательность действий может быть реализована в программе с помощью следующих операторов:

```
x1:=(-b-sqrt(d))/(2*a);
x2:=(-b+sqrt(d))/(2*a);
Edit4.Text:= FloattoStr (x1);
Edit4.Visible:=True;
Edit5.Text:=FloattoStr(x2);
Edit5.Visible:=True;
Label4.Visible:=True;
Label4.Caption:='Корни уравнения'
```

В том случае, если найденный дискриминант отрицателен, требуемые действия сводятся к тому, чтобы:

а) вывести на экран надпись, говорящую об отсутствии решения уравнения, сделав надпись перед этим видимой;

б) сделать невидимыми текстовые окна для вывода результатов, так как в этом случае необходимость в них отсутствует .

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

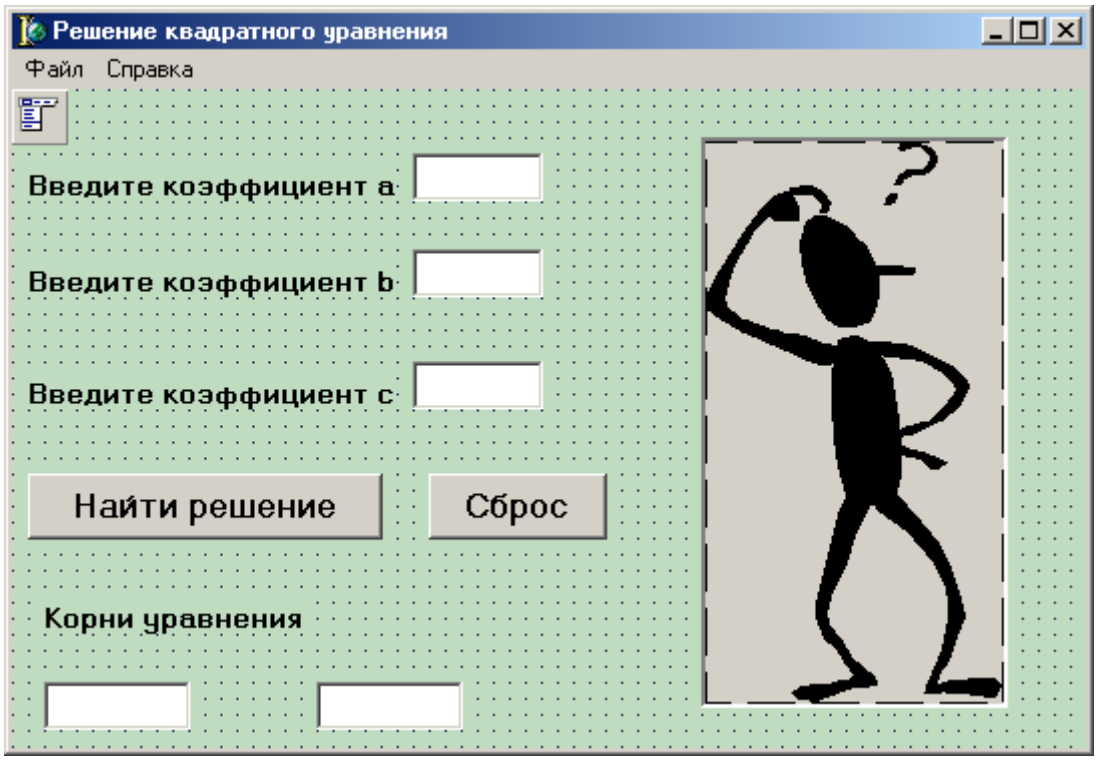

Рис. 25. Графический интерфейс программы решения квадратного уравнения

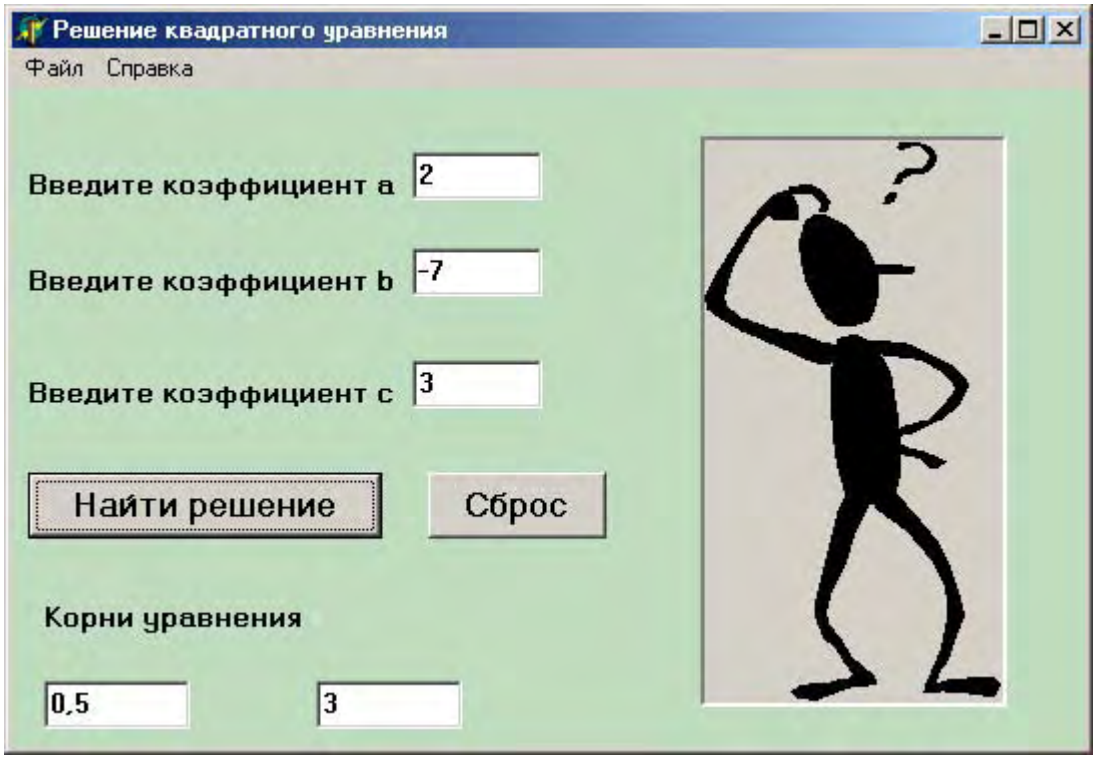

Рис. 26. Пример работы программы решения квадратного уравнения при положительном значении дискриминанта

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

Перечисленные действия реализуются с помощью операторов:

```
Label4.Visible:=True;
Label4.Caption:='Уравнение не имеет решения';
Edit4.Visible:=False;
Edit5.Visible:=False
```

И в том и в другом случае для описания требуемых действий нам понадобилась последовательность из нескольких операторов. Читатель, конечно, помнит, что такая последовательность называется в программе составным оператором и должна быть заключена в операторные скобки ***begin*** и ***end.*** Сами же эти составные операторы являются ветвями условного оператора ***if***, в заголовке которого указано условие в зависимости от выполнения или невыполнения которого работает та или иная ветвь оператора. В данном случае в заголовке условного оператора значение переменной ***d*** (дискриминант) сравнивается с нулем, что и определяет дальнейший ход работы программы. В целом условный оператор будет выглядеть следущим образом:

```
if d>=0
then
    begin
    x1:=(-b-sqrt(d))/(2*a);
    x2:=(-b+sqrt(d))/(2*a);
    Edit4.Text:=FloattoStr(x1);
    Edit4.Visible:=True;
    Edit5.Text:=FloattoStr(x2);
    Edit5.Visible:=True;
    Label4.Visible:=True;
    Label4.Caption:='Корни уравнения'
    end
else
    begin
    Label4.Visible:=True;
    Label4.Caption:='Уравнение не имеет решения';
    Edit4.Visible:=False;
    Edit5.Visible:=False
    end
```

Вся же разобранная выше процедура в целом будет выглядеть так:

```
procedure TForm1.Button1Click(Sender: TObject);
var a,b,c,d,x1,x2:real;
begin
a:=StrtoFloat(Edit1.Text);
```

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

```
 b:=StrtoFloat(Edit2.Text);
c:=StrtoFloat(Edit3.Text);
d:=sqr(b)-4*a*c;
if d>=0
then
      begin
      x1:=(-b-sqrt(d))/(2*a);
      x2:=(-b+sqrt(d))/(2*a);
      Edit4.Text:=FloattoStr(x1);
      Edit4.Visible:=True;
      Edit5.Text:=FloattoStr(x2);
      Edit5.Visible:=True;
      Label4.Visible:=True;
      Label4.Caption:='Корни уравнения'
      end
else
      begin
      Label4.Visible:=True;
      Label4.Caption:='Уравнение не имеет решения';
      Edit4.Visible:=False;
      Edit5.Visible:=False
      end
end;
```

Следующая процедура, которую рассматриваем, описывает реакцию на нажатие экранной кнопки «Сброс» (кнопка ***Button2***). Эта процедура называется ***TForm1.Button2Click*** и в ней должны выполняться следующие действия:

а) очистка текстовых окон, в которые вводятся исходные данные;

б) превращение окон с результатами и поясняющей надписи в невидимые окна.

Первая операция осуществляется путем присвоения свойству ***Text*** каждого из окон пустой строки, которая записывается в виде пары апострофов. Вторая операция производится путем присвоения свойству ***Visible*** каждого из указанных объектов значения ***False***. В итоге процедура будет выглядеть следующим образом:

```
procedure TForm1.Button2Click(Sender: TObject);
begin
Edit1.Text:='';
Edit2.Text:='';
Edit3.Text:='';
Edit4.Visible:=False;
Edit5.Visible:=False;
Label4.Visible:=False
```

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

end;

Для того чтобы завершить работу над основной формой проекта, осталось запрограммировать пункты меню. Написание кода для пунктов меню производится таким же образом, как и для прочих объектов, т. е. вначале мы выделяем объект (например, пункт меню ***N3***, с помощью которого программа закрывается), затем выбираем вкладку ***Events*** в инспекторе объектов и на вкладке находим событие ***Onclick***. Дважды щелкнув мышью в поле справа от названия события, автоматически открываем соответствующую процедуру, которая для пункта ***N3*** будет называться ***TForm1.N3Click***. Для закрытия программы в процедуру нужно вставить единственную команду ***Close***, а процедура целиком будет выглядеть так:

```
procedure TForm1.N3Click(Sender: TObject);
begin
close
end;
```

Пункт меню, представленный объектом ***N4,*** должен вызывать дополнительное окно со сведениями об авторах. Данное вспомогательное окно должно содержать сведения об авторе и кнопку, которая его закрывает. Для включения окна в состав проекта нужно его создать, а затем дополнить его необходимыми элементами интерфейса. Эта задача не вызовет у читателя никаких сложностей, так как она выполняется точно так же, как и в предыдущих проектах. Не должно вызвать проблем и написание соответствующей подпрограммы. Подпрограмма, описывающая реакцию на выбор пункта меню «Об авторах» называется ***TForm1.N4Click*** и должна выглядеть следующим образом:

```
procedure TForm1.N4Click(Sender: TObject);
begin
Form2.Show
end;
```

В целом программный модуль ***Unit1*** для основной формы проекта имеет следующий вид:

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs, StdCtrls, ExtCtrls, Menus;
```

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

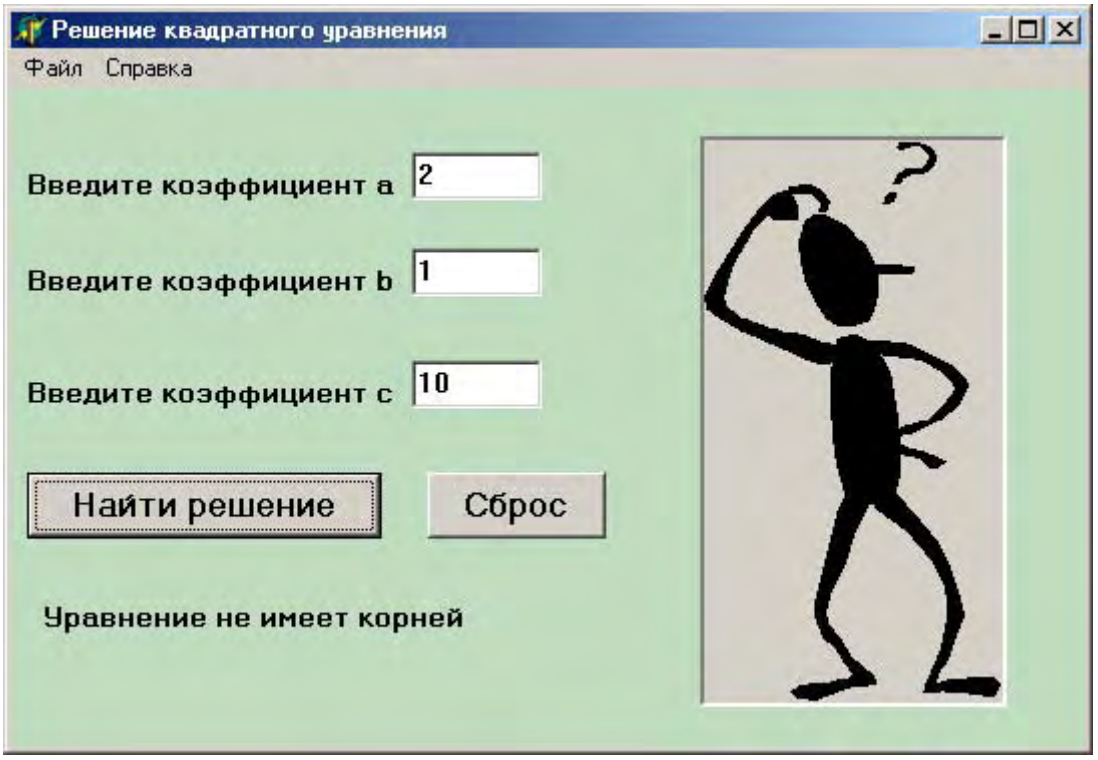

Рис. 27. Пример работы программы решения квадратного уравнения при отрицательном значении дискриминанта

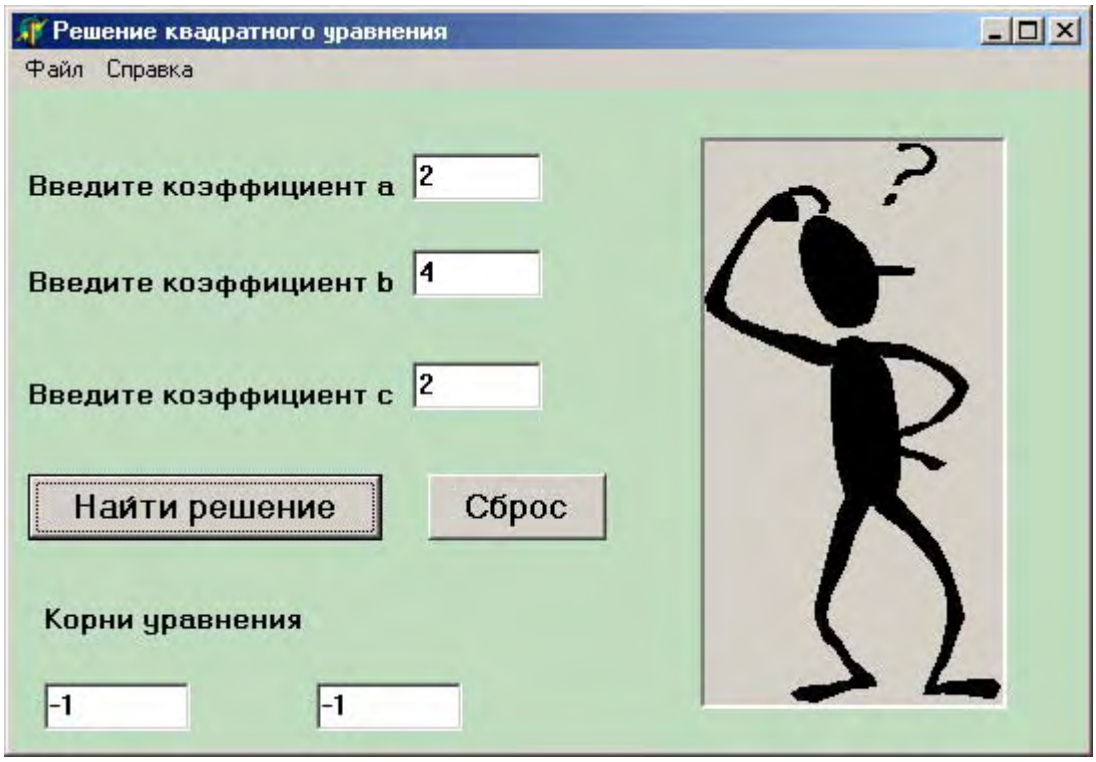

Рис. 28. Пример работы программы решения квадратного уравнения при нулевом значении дискриминанта.

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

```
        type
  TForm1 = class(TForm)
    Label1: TLabel;
    Edit1: TEdit;
    Label2: TLabel;
    Label3: TLabel;
    Edit2: TEdit;
    Edit3: TEdit;
    Panel1: TPanel;
    Image1: TImage;
    Button1: TButton;
    Label4: TLabel;
    Edit4: TEdit;
    Edit5: TEdit;
    Button2: TButton;
    MainMenu1: TMainMenu;
    N1: TMenuItem;
    N2: TMenuItem;
    N3: TMenuItem;
    N4: TMenuItem;
    procedure Button1Click(Sender: TObject);
    procedure N3Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure N4Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation
 uses Unit2;
{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
var a,b,c,d,x1,x2:real;
begin
a:=StrtoFloat(Edit1.Text);
b:=StrtoFloat(Edit2.Text);
c:=StrtoFloat(Edit3.Text);
d:=sqr(b)-4*a*c;
```

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

```
        if d>=0
then
    begin
    x1:=(-b-sqrt(d))/(2*a);
    x2:=(-b+sqrt(d))/(2*a);
    Edit4.Text:=FloattoStr(x1);
    Edit4.Visible:=True;
    Edit5.Text:=FloattoStr(x2);
    Edit5.Visible:=True;
    Label4.Visible:=True;
    Label4.Caption:='Корни уравнения'
    end
else
    begin
    Label4.Visible:=True;
    Label4.Caption:='Уравнение не имеет решения';
    Edit4.Visible:=False;
    Edit5.Visible:=False
    end
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
Edit1.Text:='';
Edit2.Text:='';
Edit3.Text:='';
Edit4.Visible:=False;
Edit5.Visible:=False;
Label4.Visible:=False
end;

procedure TForm1.N3Click(Sender: TObject);
begin
close
end;

procedure TForm1.N4Click(Sender: TObject);
begin
Form2.Show
end;

end.
```

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

Для полного завершения работы над проектом осталось написать программный код для модуля Unit2, который описывает реакцию объектов входящих в состав второй, вспомогательной формы. Этот код аналогичен соответствующему коду из предыдущих проектов, написание его также не вызовет у читателя затруднений. После написания кода для вспомогательной формы работа над проектом завершена и его можно запускать на выполнение.

На рис. 26-28 приведены примеры, демонстрирующие, как будет выглядеть основное окно программы при различных исходных данных, дающих различные значения дискриминанта. На рис. 26 приведен пример для положительного дискриминанта, на рис. 27 – для отрицательного дискриминанта, а на рис. 28 – для дискриминанта, равного нулю. Кроме того, на рис. 28 показано, как будет выглядеть вызываемое через меню дополнительное окно со справочной информацией.

11. Оператор цикла. Программа «Факториал»

При выполнении различных научно-технических расчетов часто возникает ситуация, когда необходимо много раз подряд выполнять такие действия, в которых будут изменяться только значения операндов, над которыми эти действия производятся, а сам характер действий будет оставаться неизменным.

Примером такой многократно повторяющейся последовательности действий является возведение числа ***a*** в некоторую степень. Обозначим искомую степень числа ***st***, а показатель степени - ***n.*** Как известно из курса элементарной математики, нулевая степень любого числа равна единице, т. е. при ***n***=0 значение ***st*** будет равно единице. Если нужно возвести некоторое число a в первую степень, то необходимо умножить начальное значение ***st*** на ***a***. На языке ***Object Pascal*** эти действия можно записать следующим образом:

```
st:=1;
st:=st*a;
```

Если нужно возвести число ***a*** в квадрат, то необходимо полученное значение ***st*** (т. е. ***a*** в первой степени) еще раз умножить на основание ***a***. Тогда запись на ***Object Pascal*** будет выглядеть так:

```
st:=1;
st:=st*a;
st:=st*a;
```

Для возведения того же числа ***a*** в куб понадобится следующая последовательность операторов:

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

```
 st:=1;
st:=st*a;
st:=st*a;
st:=st*a;
```

Из приведенного примера видно, что при возведении числа в ***n***-ю степень мы многократно выполняем один и тот же оператор, в котором производится одна и та же операция умножения, один из операндов ***a*** остается неизменным, а значение другого ***st*** в процессе вычислений изменяется. Понятно, что для возведения числа в десятую степень понадобится повторить одно и то же действие 10 раз, для возведения в двадцатую степень – 20 раз и так далее.

Естественно, что много раз подряд записывать один и тот же оператор (или группу операторов) в ходе работы над программой является крайне нерациональным. Для многократного выполнения одного и того же действия (или последовательности действий) в программировании существует специальная структура, называемая циклом.

В общем виде цикл представляет собой последовательность операторов, которая состоит из состоит из двух основных блоков:

1. Заголовок цикла. В заголовке цикла содержится некоторое условие, которое определяет число повторений цикла.

2. Тело цикла. Телом цикла называется простой или составной оператор (как читатель, конечно, помнит, составным оператором называется группа операторов, заключенная в операторные скобки ***begin*** и ***end***), который может многократно повторяться в ходе работы цикла. В теле цикла могут содержаться операторы различных типов. Это могут быть операторы ввода и вывода, операторы присваивания, условные операторы, а также другие операторы цикла. Оператор цикла, находящийся в теле другого циклического оператора, называется вложенным.

Циклы могут использоваться для решения самых разнообразных задач и соответственно структура самих циклов может несколько отличаться. Всего в языке Паскаль используется 3 разновидности циклов:

1. Цикл с заранее заданным числом повторений. Его также называют циклом со счетчиком. (Цикл ***for..to***).
2. Цикл с предусловием (Цикл ***while***).
3. Цикл с постусловием (Цикл ***repeat..until***).

Каждый из этих видов цикла имеет свои особенности и область применения, которые мы и рассмотрим ниже. Общим для всех этих операторов является то, что они включают в себя управляющие конструкции (в операторах ***for..to*** и ***while*** **–** это строка заголовка, а в операторе ***repeat..until*** **–** строка заголовка и завершающая строка) и тело цикла. Если количество повторений тела цикла заранее известно, то рекомендуется использовать оператор цикла ***for..to***. В другом случае нужно использовать оператор ***repeat..until*** (если для решения поставленной задачи требуется, чтобы тело цикла выполнялось хотя

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

бы один раз), либо оператор ***while*** (в этом случае перед выполнением тела цикла проверяется, есть ли вообще необходимость в его выполнении). В настоящем пособии мы более подробно рассмотрим цикл с заранее известным числом повторений. Общий вид данного оператора следующий:

for i:=n1 to n2 do
< тело цикла >;

где ***i –*** управляющая переменная цикла, называемая также счетчиком цикла; ***n1*** – начальное значение счетчика цикла; ***n2*** – конечное значение счетчика цикла; ***for, to*** и ***do*** – служебные слова (***for*** в данном случае означает «для», ***to*** – «до», ***do*** – «делать, выполнять»). Весь текст заголовка можно расшифровать таким образом: для ***i***, изменяющего значение от ***n1*** до ***n2*** выполнить. Имеется в виду выполнение тела цикла. При этом конечное значение счетчика цикла должно быть больше, чем начальное. В ходе работы данного оператора значение счетчика при каждом выполнении тела цикла увеличивается на единицу и таким образом принимает все целочисленные значения от ***n1*** до ***n2***, а тело цикла всего выполняется ***n2-n1+1*** раз. Следует обратить внимание на то, что между заголовком цикла и телом цикла не ставится точка с запятой.

В качестве примера использования циклов в системе ***Delphi*** рассмотрим задачу вычисления факториала некоторого натурального числа. Факториалом натурального числа ***n*** называется произведение всех целых чисел от 1 до ***n*** включительно. Например факториал числа 5 равен:

$$1 * 2 * 3 * 4 * 5 = 120.$$

Перед началом работы над проектом напишем фрагмент программы на языке программирования ***Object Pascal***, в котором вычисляется значение факториала. Исходную величину обозначим ***n***, а искомый результат ***f***. Помимо указанных переменных в данном фрагменте используем еще одну вспомогательную переменную ***i***, которая выступает в роли переменной цикла. Перед началом работы цикла, в котором производятся вычисления, переменной ***f*** необходимо присвоить значение 1, чтобы впоследствии его умножать на текущее значение переменной цикла. Данный фрагмент программы будет выглядеть следующим образом:

```
f:=1;
for i:=1 to n do
f:=f*i;
```

В итоге работы данного фрагмента находим искомое значение факториала ***f***. При работе над проектом необходимо будет дополнить данный фрагмент операторами, которые обеспечивают ввод исходных данных с клавиатуры и вывод полученного результата на экран компьютера.

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

Теперь приступим к реализации данной задачи в среде ***Delphi***. Для этого разработаем новый проект, который назовем ***faktor.*** Как всегда работу над проектом мы начинаем с его сохранения и создания графического интерфейса программы. Интерфейс будет включать в себя следующие основные компоненты:

1. Строка меню, содержащая разделы «Файл» и «Справка».
2. Текстовое окно для ввода исходных данных, слева от которого будет расположено поле комментария, который поясняет содержание данного окна.
3. Кнопка «Ввод», нажатие которой должно приводить к вычислению искомого результата.
4. Текстовое окно с результатом, дополненное поясняющей надписью.
5. Экранная кнопка «Сброс», используемая для очистки текстовых окон и кнопка «Выход», используемая для закрытия окна программы.
6. Панель с рисунком юмористического содержания, изображающая человека, излучающего радость оттого, что ему удалось добиться решения сложной задачи.

Работу над интерфейсом начнем с создания строки меню. Сам процесс создания меню не должен вызывать у пользователя затруднений, так как этот процесс был подробно описан в предыдущем разделе данной книги. Отметим только, что каждый из разделов меню содержит вложенный пункт. Для меню «Файл» (объект ***N1***) таким пунктом является «Выход» (объект ***N3***), а для раздела «Правка» (объект ***N2***) – соответственно пункт «Об авторе» (объект ***N4***). Выбор этого пункта меню приведет к появлению на экране компьютера дополнительного второго окна, содержащего информацию об авторах програмы. Это окно содержит поясняющую надпись и закрывается при нажатии экранной кнопки «***Ok***».

Теперь приступаем к настройке основной формы программы. Присваиваем основной форме новый заголовок – факториал и изменяем цвет формы на светло-зеленый. Следующим компонентом основной формы является текстовое окно ***Edit1***. Слева от него расположим окно комментария ***Memo1***. Для того чтобы данное окно не выделялось на общем фоне формы, поступаем так же, как и в предыдущем проекте. Цвет окна делаем светло-зеленым, как и у самой формы, а свойству ***BorderStyle*** присваиваем значение ***None***, т. е. границу делаем невидимой. Под текстовым окном располагаем экранную кнопку (объект ***Button1***), которой присваиваем заголовок «Ввод».

Под кнопкой «Ввод» располагаем второе текстовое окно для вывода результатов (объект ***Edit2***), а слева от него надпись, говорящую о содержимом данного окна (объект ***Label1***). Внизу экрана располагаем две экранные кнопки. Это кнопка «Сброс» (объект ***Button2***) и кнопка «Выход» (объект ***Button3***). Назначение этих кнопок понятно из их заголовков.

В правой части формы размещаем объект ***Panel1***, а на этом объекте разместим иллюстрацию (объект ***Image1***). Картинку для данной иллюстрации

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

найдем в коллекции ***Clipart***, входящей в состав пакета ***Microsoft Office 97***. Картинка содержится в файле под названием ***Amidea***. Такие изображения хорошо поддаются масштабированию, поэтому производить точную подгонку объектов под размеры картинки нет необходимости. Затем к основной форме подключаем вспомогательную форму со сведениями об авторе программы и создаем ее интерфейс. На этом создание графического интерфейса программы завершается (рис. 29).

Следующим этапом работы над программой является написание кода для пунктов меню и экранных кнопок. Пункт меню ***N3*** закрывает окно формы, поэтому соотвествующий код будет содержать только оператор close. Для пункта меню ***N4*** необходимо написать оператор ***Form2.Show***, который выводит на экран компьютера дополнительную форму со сведениями об авторе. При этом нужно предварительно создать новую форму с помощью раздела меню системы программирования ***File→New***, а также описать подключение дополнительного модуля для этой формы ***Unit2*** в разделе основного модуля ***Unit1***, который начинается со служебного слова ***Uses***.

Далее нужно написать код, который описывает реакцию на нажатие экранной кнопки «Ввод», которое должно приводить к вычислению факториала с отображением его в соответствующем текстовом окне. Подпрограмма, описывающая данную реакцию, должна содержать описание используемых переменных (выше мы рассматривали какие переменные здесь необходимы), а также, как уже говорилось, средства ввода данных и вывода результатов. Получившаяся подпрограмма должна выглядеть следующим образом:

```
procedure TForm1.Button1Click(Sender: TObject);
var i,f,n:integer;
begin
n:=StrtoInt(Edit1.Text);
f:=1;
for i:=1 to n do
f:=f*i;
Edit2.Text:=InttoStr(f)
end;
```

Теперь осталось запрограммировать действия кнопки «Сброс», очищающей текстовые окна и кнопки «Выход», закрывающей основную форму. Написание соответствующих кодов не должно вызвать у пользователя каких-либо затруднений, так как аналогичные коды мы уже писали в предыдущих проектах. Ниже приводится исходный код основного модуля данного проекта.

```
unit Unit1;
```

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

```
        interface

    uses
      Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
      Dialogs, Menus,Unit2, StdCtrls, ExtCtrls;

    type
      TForm1 = class(TForm)
        MainMenu1: TMainMenu;
        N1: TMenuItem;
        N2: TMenuItem;
        N3: TMenuItem;
        N4: TMenuItem;
        Memo1: TMemo;
        Edit1: TEdit;
        Button1: TButton;
        Label1: TLabel;
        Edit2: TEdit;
        Panel1: TPanel;
        Image1: TImage;
    Button2: TButton;
        Button3: TButton;
        procedure N3Click(Sender: TObject);
        procedure N4Click(Sender: TObject);
        procedure Button1Click(Sender: TObject);
        procedure Button2Click(Sender: TObject);
        procedure Button3Click(Sender: TObject);
    private
        { Private declarations }
      public
        { Public declarations }
      end;

    var
      Form1: TForm1;

    implementation

    {$R *.dfm}

    procedure TForm1.N3Click(Sender: TObject);
    begin
    close
    end;
```

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

```
        procedure TForm1.N4Click(Sender: TObject);
begin
Form2.Show
end;

procedure TForm1.Button1Click(Sender: TObject);
var i,f,n:integer;
begin
n:=StrtoInt(Edit1.Text);
f:=1;
for i:=1 to n do
f:=f*i;
Edit2.Text:=InttoStr(f)
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
Edit1.Text:='';
Edit2.Text:=''
end;

procedure TForm1.Button3Click(Sender: TObject);
begin
Close
end;

end.
```

Написание дополнительного модуля Unit2, описывающего объекты вспомогательной формы, аналогично написанию таких же модулей в предыдущих проектах и не должно вызвать у читателя затруднений. После написания этого модуля программа готова к работе. Внешний вид программы после запуска ее на выполнение приведен на рис. 30. При работе с данной программой следует обратить внимание на то, что в качестве исходных данных следует вводить натуральные числа не больше 15, так как факториалы даже сравнительно небольших по модулю чисел являются большими величинами. Например факториал числа 12 равен 479001600.

12. Программа «Интеграл»

Интегрированная среда ***Delphi*** может быть использована не только для решения задач элементарной математики. Данная среда позволяет создавать проекты для удобного и наглядного решения задач из области высшей

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

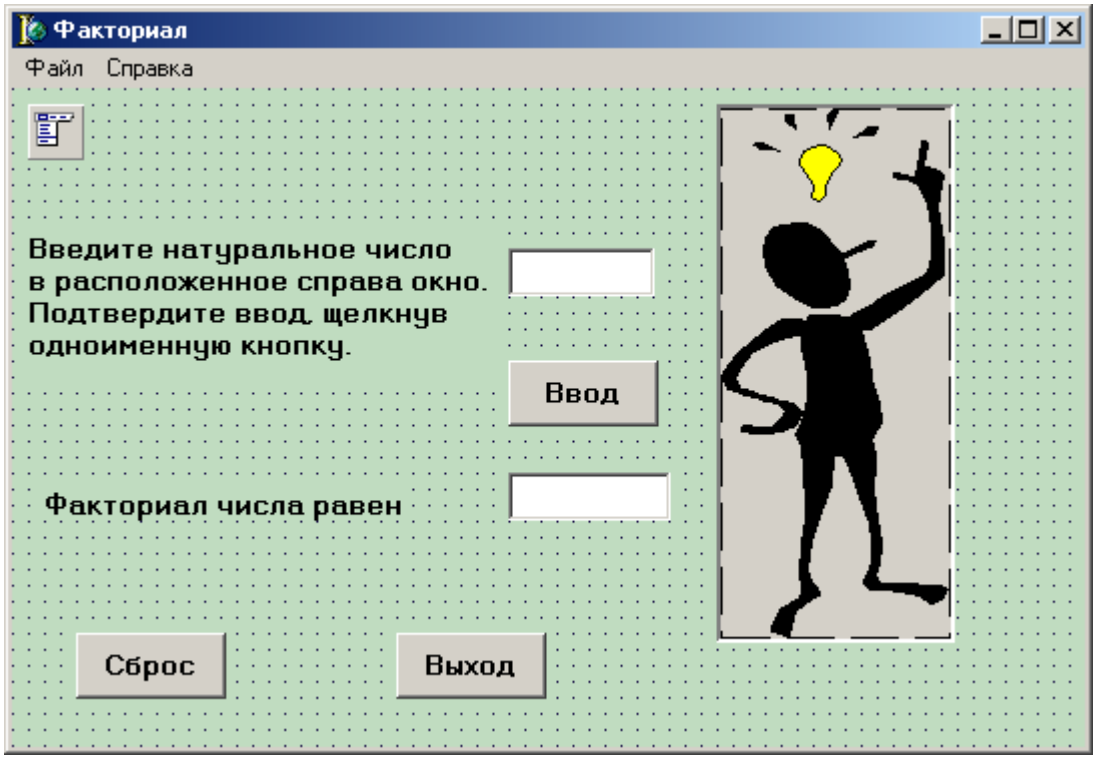

Рис. 29. Графический интерфейс программы «факториал»

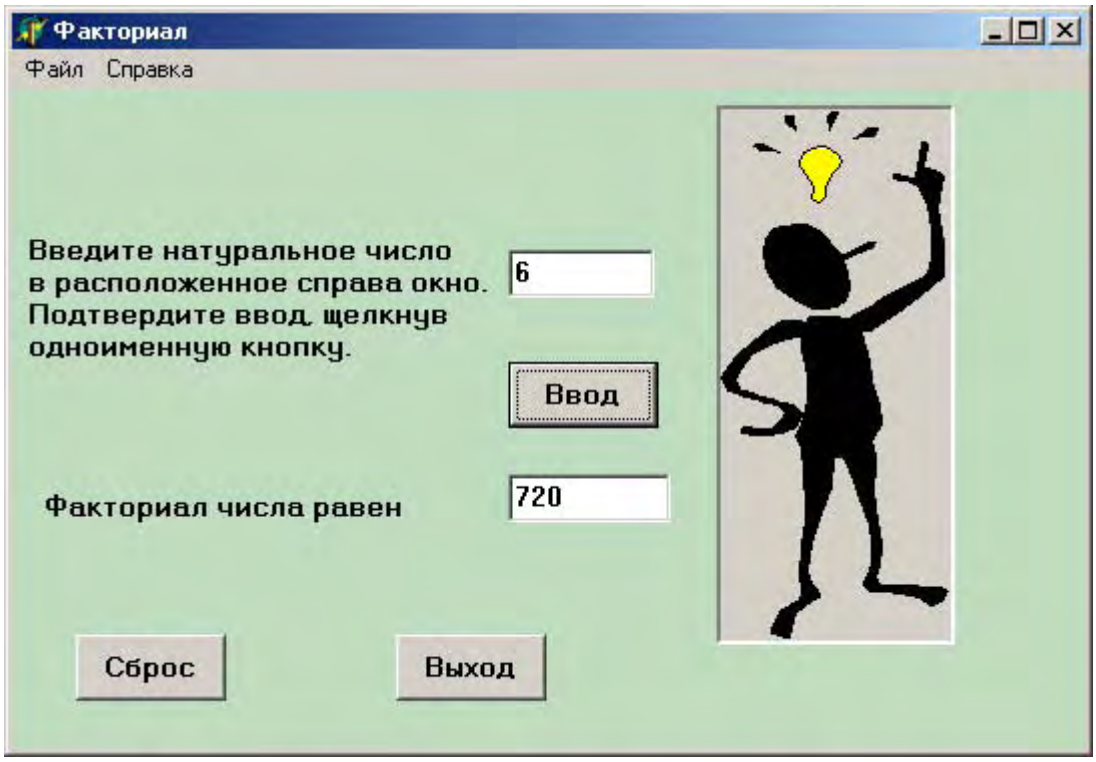

Рис. 30. Программа «факториал» во время работы

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

математики. Примером такой задачи является вычисление определенного интеграла некоторой функции.

Вычисление определенного интеграла геометрически можно интерпретировать как вычисление площади фигуры, которая ограничена линией графика подинтегральной функции, осью абсцисс и двумя вертикальными линиями, соответствующими верхнему и нижнему пределам интегрирования (рис. 32). Такую фигуру можно разбить на ряд участков, каждый из которых по своей форме близок к прямоугольной трапеции. Площадь каждой такой трапеции можно вычислить, а затем сложить получившиеся площади и найти таким образом общую площадь фигуры, т. е. искомый интеграл. На этом и основан способ приближенного вычисления определенного интеграла, который так и называется методом трапеций.

Вычисление значения определенного интеграла $\int_a^b f(x)dx$ по методу трапеций производится следующим образом: участок от ***a*** до ***b*** разбивается на ***n*** равных по величине отрезков. Длина каждого отрезка **h** = ***(b-a)/n***. Значения аргумента в точках разбиения будут соответственно равны: $x_0=a$, $x_1=a+h$, …, $x_i=x_0+ih$, …, $x_n=b$. Значения функции ***f(x)*** в точках x_i будут соответственно обозначаться y_i. Тогда, согласно формуле трапеций, определенный интеграл можно будет вычислить следующим образом:

$$\int_a^b f(x)dx \approx h\left(\frac{y_0 + y_n}{2} + y1 + \ldots y_i + \ldots + y_n\right)$$

Перед тем, как приступить к разработке проекта «Интеграл» составим фрагмент программы на языке ***Object Pascal***, который реализует вышеуказанный метод на компьютере. В качестве подинтегральной функции мы используем тригонометрическую функцию ***y=sin(x).*** В данном фрагменте будут использованы следующие обозначения:

n – количество отрезков, на которые разбивается участок интегрирования;

i – вспомогательная переменная цикла;

a – начальный предел интегрирования;

b – конечный предел интегрирования;

h – длина отрезка интегрирования;

yn – значение подинтегральной функции в начальной точке (точка a);

yk – значение подинтегральной функции в конечной точке (точка b);

yp – одно из промежуточных значений подинтегральной функци;

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

s – искомое значение определенного интеграла.

Первые две переменные должны относиться к целочисленному типу, а все остальные – к вещественному. Тогда описание переменных должно выглядеть следующим образом:

```
var i,n:integer;
a,b,s,h,x,yp,yn,yk:real;
```

Сама формула трапеции реализована на языке ***Object Pascal*** в следующем фрагменте программы, в котором для расчетов использован цикл с заранее известным числом повторений:

```
h:=(b-a)/n;
yp:=0;
x:=a;
for i:=1 to n-1 do
begin
x:=x+h;
yp:=yp+sin(x)
end;
yn:=sin(a);
yk:=sin(b);
s:=((yk+yn)/2+yp)*h;
```

Далее мы приступаем к разработке графического интерфейса для данной программы. Интерфейс программы должен включать в себя следующие элементы (рис.31):

1. Строка меню, содержащего разделы «Файл» и «Справка». Первый раздел включает в себя пункт «Выход», а второй пункт «Об авторе». Выполнение данного пункта выводит на экран компьютера дополнительную форму со справочной информацией и закрывающей кнопкой.
2. Текстовое окно для ввода верхнего предела интегрирования, над которым находится поле комментария, поясняющее, что вводится в текстовое окно.
3. Текстовое окно для ввода нижнего предела интегрирования, над которым находится соответствующее поле комментария.
4. Текстовое окно для ввода количества участков, на которые разбивается отрезок интегрирования.
5. Кнопка «Вычислить». Щелчком на этой кнопке должно производиться вычисление определенного интеграла.
6. Кнопка «Сброс». Щелчок на данной кнопке очищает окна, содержащие исходные данные и результат вычислений.
7. Текстовое окно для вывода полученного значения интеграла с расположенным над ним окном комментария.
8. Панель с иллюстрацией, поясняющей суть процесса вычисления

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

определенного интеграла.

Создание интерфейса начинаем с работы над основной формой проекта. В начале переименовываем объект, который получает название «Вычисление определенного интеграла функции ***y=sin(x)***». Затем изменяем цвет формы на небесно-голубой.

Далее создаем главное меню, которое имеет разделы «Файл» (объект ***N1***) и «Справка» (объект ***N2***). Эти разделы имеют соответственно пункты «Выход» (объект ***N3***) и «Об авторе» (объект ***N4***). Создание и настройка меню производится аналогично тому, как это было сделано в предыдущих проектах. Выполнение пункта меню об авторе приводит к выводу на экран дополнительной формы справочного характера. Создание и настройка данного окна также производится аналогично предыдущим проектам.

Следующим этапом работы над проектом является создание 3 групп похожих объектов, используемых для ввода исходных данных. Это текстовое окно ***Edit1*** для ввода верхнего предела интегрирования и окно комментария ***Memo1***, текстовое окно ***Edit2*** для ввода нижнего предела интегрирования и окно комментария ***Memo2***, текстовое окно ***Edit3*** для ввода количества участков, на которые разбивается отрезок интегрирования, и окно комментария ***Memo3.***

Под этими объектами создаем две экранные кнопки. Это кнопка «Вычислить» (объект ***Button1***) и кнопка «Сброс» (объект ***Button2***). Первая кнопка подсчитывает значения определенного интеграла функции ***sin(x)***, а вторая очищает все текстовые окна, включая окно результата. Под экранными кнопками располагаем текстовое окно ***Edit4*** для вывода полученного значения интеграла и поясняющую надпись ***Label1.***

Последнюю группу объектов данной программы создаем в правой части формы. Это панель (объект ***Panel1***), содержащее иллюстрацию к программе, поясняющую смысл вышеуказанного метода вычисления (объект ***Image1***). На создании графического объекта остановимся подробнее. Картинка, которая использована в данной иллюстрации, содержится в справочном материале программного комплекса ***StudyWorks***. ***StudyWorks*** – это программный комплекс, разработанный фирмой ***Mathsoft*** для обучения основам математики. Картинка не является отдельным графическим объектом как картинки из коллекции ***Clipart***, входящей в состав пакета ***Microsoft Office***, а интегрирована в страницу со справочной информацией.

Для того чтобы этот рисунок можно было вставить в создаваемый проект, используем следующую технологию. Открываем страницу программы ***StudyWorks*** со справочной информацией. Создаем копию окна программы командой ***Alt+PrintScreen*** (содержимое окна при этом копируется в буфер обмена).

Далее запускаем стандартное приложение ***Windows*** – программу ***Paint.*** Командой «Правка»→ «Вставить» вставляем в открытый документ ***Paint***

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

копию окна из буфера. Затем с помощью кнопки  (выделение) в окне программы ***Paint*** выделяем фрагмент, который содержит рисунок. Командой меню «Правка»→»Копировать» копируем в буфер обмена выделенный фрагмент. Затем командой «Рисунок»→«Очистить» очищаем окно программы Paint и командой «Правка»→ «Вставить» вставляем в окно Paint фрагмент с рисунком. Получившийся документ ***Paint*** с помощью команды «Файл»→«Сохранить как…» сохраняем в папке проекта ***Delphi*** для дальнейшего использования.

Обратите внимание на то, что созданная в ***Paint*** картинка является растровым объектом, и плохо поддается масштабированию. Поэтому следует определить размеры данного рисунка в пикселах с помощью команды меню «Рисунок» → «Атрибуты», а затем, исходя из уже известных размеров, подобрать размеры объектов ***Panel1*** и ***Image1*** в проекте. Объект ***Image1*** может иметь те же размеры, что и рисунок, а объект ***Panel*** по вертикали и по горизонтали больше на 4 пиксела. Если установить вышеуказанные размеры объектов, то внешний вид рисунка не будет искажен. На этом создание интерфейса программы завершено (рис.31).

Далее приступаем к созданию программного кода. Подпрограммы, соответствующие пунктам программного меню ***N3*** и ***N4***, аналогичны соотвествующим пунктам в предыдущих проектах. В качестве основы для процедуры ***Button1Click***, описывающей реакцию на нажатие кнопки «Вычислить», берем фрагмент программы приведенный в начале данной главы и дополняем его операторами, описывающими преобразование исходных данных из строковой формы в числовую и оператором, преобразующим полученный результат из числовой формы в строковую. Подпрограмма ***Button2Click***, описывающая реакцию на щелчок на кнопке «Сброс», аналогична соответствующим подпрограммам в предыдущих проектах. Ниже приводится полный текст программного кода для данного проекта.

```
unit Unit1;

  interface

  uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
    Dialogs, Menus, StdCtrls, ExtCtrls,Unit2;

  type
    TForm1 = class(TForm)
      MainMenu1: TMainMenu;
      N1: TMenuItem;
      N2: TMenuItem;
```

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

```
      N3: TMenuItem;
    N4: TMenuItem;
    Memo1: TMemo;
    Edit1: TEdit;
    Memo2: TMemo;
    Edit2: TEdit;
    Memo3: TMemo;
    Edit3: TEdit;
    Button1: TButton;
    Button2: TButton;
    Label1: TLabel;
    Edit4: TEdit;
    Panel1: TPanel;
    Image1: TImage;
    procedure N3Click(Sender: TObject);
    procedure N4Click(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.N3Click(Sender: TObject);
begin
close
end;
procedure TForm1.N4Click(Sender: TObject);
begin
Form2.Show
end;

procedure TForm1.Button1Click(Sender: TObject);
var a,b,s,h,x,yp,yn,yk:real; i,n:integer;
begin
a:=StrtoFloat(Edit1.Text);
b:=StrtoFloat(Edit2.Text);
```

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

Рис. 31. Графический интерфейс программы «вычисление определенного интеграла»

Рис. 32. Программа «вычисление определенного интеграла» во время работы

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

```
n:=StrtoInt(Edit3.Text);
h:=(b-a)/n;
yp:=0;
x:=a;
for i:=1 to n-1 do
begin
x:=x+h;
yp:=yp+sin(x)
end;
yn:=sin(a);
yk:=sin(b);
s:=((yk+yn)/2+yp)*h;
Edit4.Text:=FloattoStr(s)
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
Edit1.Text:='';
Edit2.Text:='';
Edit3.Text:='';
Edit4.Text:=''
end;
end.
```

Завершающим этапом работы над программой является, как обычно, написание кода для вспомогательной формы. После этого программа готова к запуску на выполнение.

Внешний вид программы после ее запуска на выполнение приведен на рис. 32. При работе с программой следует обратить внимание на то, что для правильной работы программы исходные данные (верхний и нижний пределы интегрирования) следует вводить не в градусах, а в радианах.

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

Библиографический список

Дмитриева С.И., Турсунов Б. С. Численные методы и программирование на ЭВМ. Учебное пособие/ ЛТА. Л.,1986.

Жуков А. В. Изучаем Delphi. СПб: Питер, 2000.

Культин Н.Б. Программирование в Turbo Pascal 7.0 и Delphi. СПб: БХВ, 1999.

Малышев О.В. Основы Delphi, 1999 (электронное издание).

Пестриков В.М., Маслобоев А.Н., Федоров О.К. Программирование в системе Турбо Паскаль 7.0. Учебное пособие. СПбГТУРП , СПб., 2002.

Пестриков В. М., Маслобоев А.Н. Turbo Pascal 7.0. Изучаем на примерах. СПб.: Наука и техника, 2003.

Симонович С.В. и др. Информатика. Базовый курс. СПб: Питер, 2001.

Слабун Т. Интерактивный курс изучения Borland Delphi 6, 2001 (электронное издание).

Borland Delphi 6 for Windows. Developers Guide. Borland Software Corporation. Scotts Valley, CA, 2001.

Borland Delphi 6 for Windows. Object Pascal Language Guide. Borland Software Corporation. Scotts Valley, CA, 2001.

Borland Delphi 6 for Windows. Quick Start. Borland Software Corporation. Scotts Valley, CA, 2001.

Kent Reisdorph. Teach Yourself Borland Delphi 4 in 21 Days. Macmillan Publishing, Indianapolis, IN,1999 .

НАУЧНО-ИНФОРМАЦИОННЫЙ ЦЕНТР САНКТ-ПЕТЕРБУРГСКОГО ГОСУДАРСТВЕННОГО ТЕХНОЛОГИЧЕСКОГО УНИВЕРСИТЕТА РАСТИТЕЛЬНЫХ ПОЛИМЕРОВ

Виктор Михайлович Пестриков
Артур Николаевич Маслобоев
Олег Константинович Федоров

Основы программирования в системе Borland Delphi

Учебное пособие

Редактор и корректор М.А. Полторак
Техн. редактор Л. Я. Титова Темплан 2004 г., поз. *19*

Подп. к печати 22.03.2004.
Формат бумаги 60 х 84/16. Бумага тип. №1. Печать офсетная.
Уч.-изд. л. 6,75. Усл. печ. л. 6,27. Усл. кр.-отт. 6,42.
Тираж 100 экз. Изд. 19. Цена «С» 19 Заказ

Ризограф ГОУВПО Санкт-Петербургского государственного технологического университета растительных полимеров, 198095, Санкт-Петербург, ул. Ивана Черных, 4