Московский государственный технический университет
имени Н.Э. Баумана
Факультет Информатика и системы управления
Кафедра Компьютерные системы и сети

«УТВЕРЖДАЮ»
Заведующий кафедрой ИУ-6
_____ Сюзев В.В.

Г.С. Иванова, Т.Н. Ничушкина, Р.С. Самарев

Создание консольных приложений в среде Turbo Delphi 2006

.

Методические указания по выполнению лабораторной работы № 1
по дисциплине Основы программирования

Москва 2012

Цель работы: изучение процесса создания и отладки программ в консольном режиме среды Turbo Delphi.

Объем работы: 2 часа.

Часть 1. Создание программ в консольном режиме

Теоретическая часть

Интегрированная среда программирования Turbo Delphi предназначена для создания 32^x разрядных приложений WINDOWS. Эта среда является частью профессиональной среды программирования Delphi Studio (2006 г.) и относится к классу визуальных, в которых разработчику предоставляется возможность прямо на экране формировать интерфейс разрабатываемого программного продукта из стандартных элементов управления.

Однако среда позволяет создавать не только приложения WINDOWS, но и консольные приложения. *Консольным приложением* в Windows называется программный продукт, который для вывода результатов использует специальное окно «Консоль». При выводе в это окно не применяются стандартные средства организации интерфейсов Windows, а потому создание консольных приложений существенно проще, чем других приложений.

Главное окно среды при запуске имеет вид, представленный на рисунке 1. Оно включает заголовок, основное меню, панель быстрого вызова, панель структуры проекта, панель инспектора объектов, Web-страницу приглашения, панель менеджера проектов и палитру инструментов.

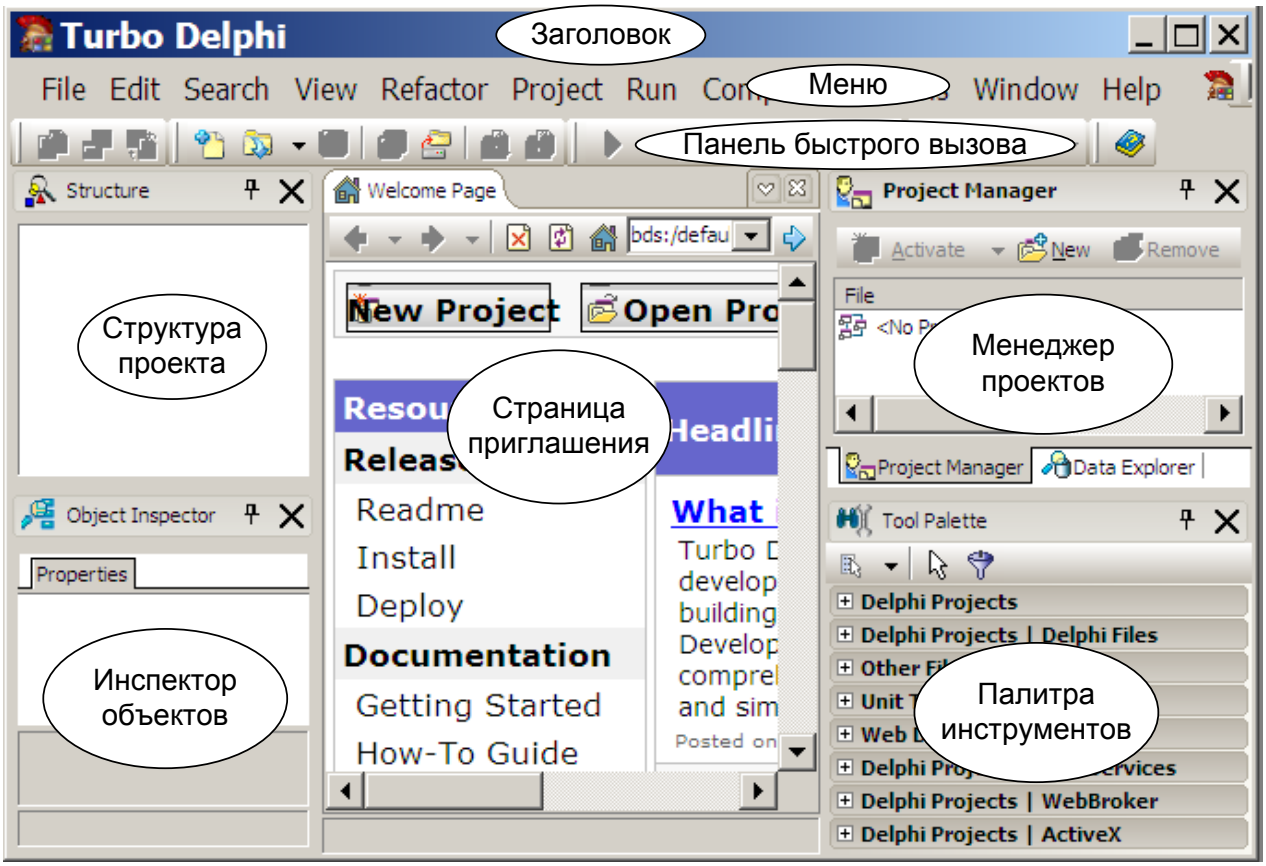

Рисунок 1 – Вид окна Turbo Delphi при входе в среду

Работа в среде осуществляется с использованием основного и вспомогательных меню, а также кнопок панели быстрого вызова. Эти кнопки применяют для упрощения доступа к часто выполняемым операциям (они дублируют соответствующие пункты меню).

Первый пункт меню **File**, как и в других приложениях Windows, контролирует создание, открытие и чтение документов – программ. Он содержит следующие подпункты (см. рисунок 2):

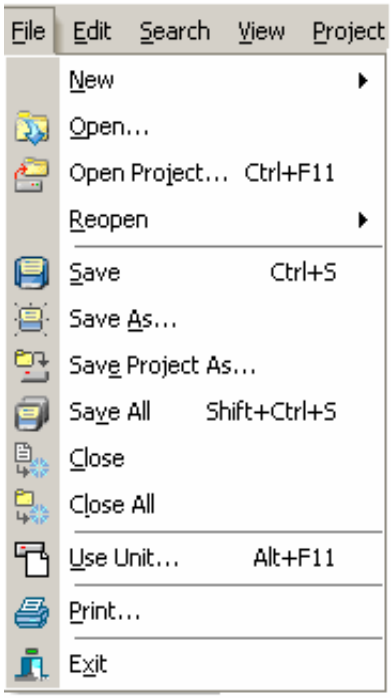

Рисунок 2 – Пункт **File**

New – создание новых проектов, файлов и т. п.;

Open – вызов уже существующих файлов;

Open Project – вызов уже существующих проектов;

Reopen – вызов проектов из списка ранее созданных;

Save – сохранение текущего файла на том же месте;

Save as… – сохранение текущего файла с новым именем или на новом месте;

Save Project as… – сохранение текущего проекта с новым именем или на новом месте;

Save All – сохранение всех файлов проекта;

Close – закрытие текущего файла;

Close All – закрытие всех открытых файлов;

Use Unit – добавление в файл ссылки на другой модуль;

Exit – выход из среды.

Пункт **Edit** позволяет выполнить основные операции с текстом программы: удаление, копирование через буфер, вставку и т. п. Пункт **Search** содержит операции поиска в текущем файле и всех файлах проекта. Через меню пункта **View** осуществляется управление внешним видом окна среды. Пункт **Project** объединяет операции работы с проектом (программой). Команды пункта **Run** осуществляют запуск и отладку программы. Остальные пункты нам пока не понадобятся.

Рассмотрим, как создаются в Turbo Delphi консольные приложения.

Примечание – Консольное приложение – простая программа и для работы с ней не понадобятся панели Инспектора объектов, Менеджера проектов и палитра инструментов. Их можно убрать, «щелкнув» мышкой по крестику правом вернем углу каждого из перечисленных окон. Окно структуры проекта используют для навигации по большим программам. К его использованию целесообразно привыкнуть. Страницу приветствия тоже можно удалить, чтобы она не отвлекала внимания.

Задание 1

Создать консольное приложение для вычисления корней квадратного уравнения.

Порядок работы:

1. Для создания заготовки консольного приложения выберите пункт **File/New/ Other…** (пункт **File** подпункт **New** и подпункт **Other…**). На экране появляется окно выбора создаваемых проектов (см. рисунок 3).

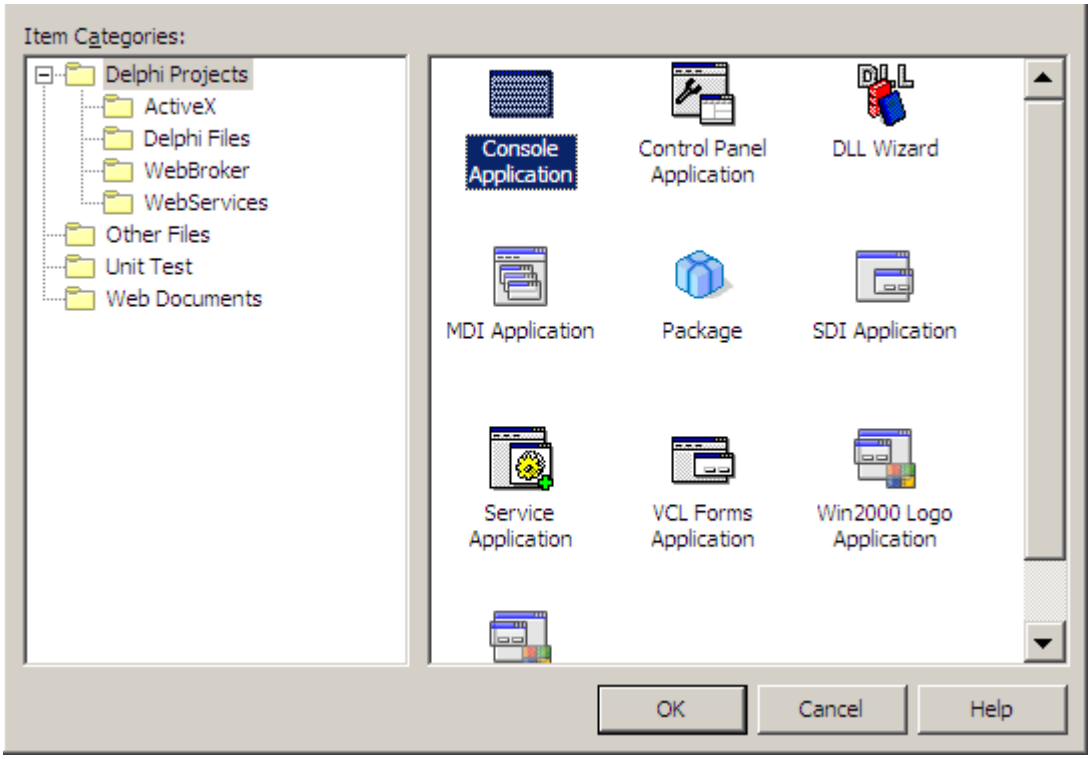

Рисунок 3 – Выбор Консольного приложения

Выберите иконку **Console Application**. После этого на экране появится заготовка консольного приложения со стандартным именем **Project1** (см. рисунок 4).

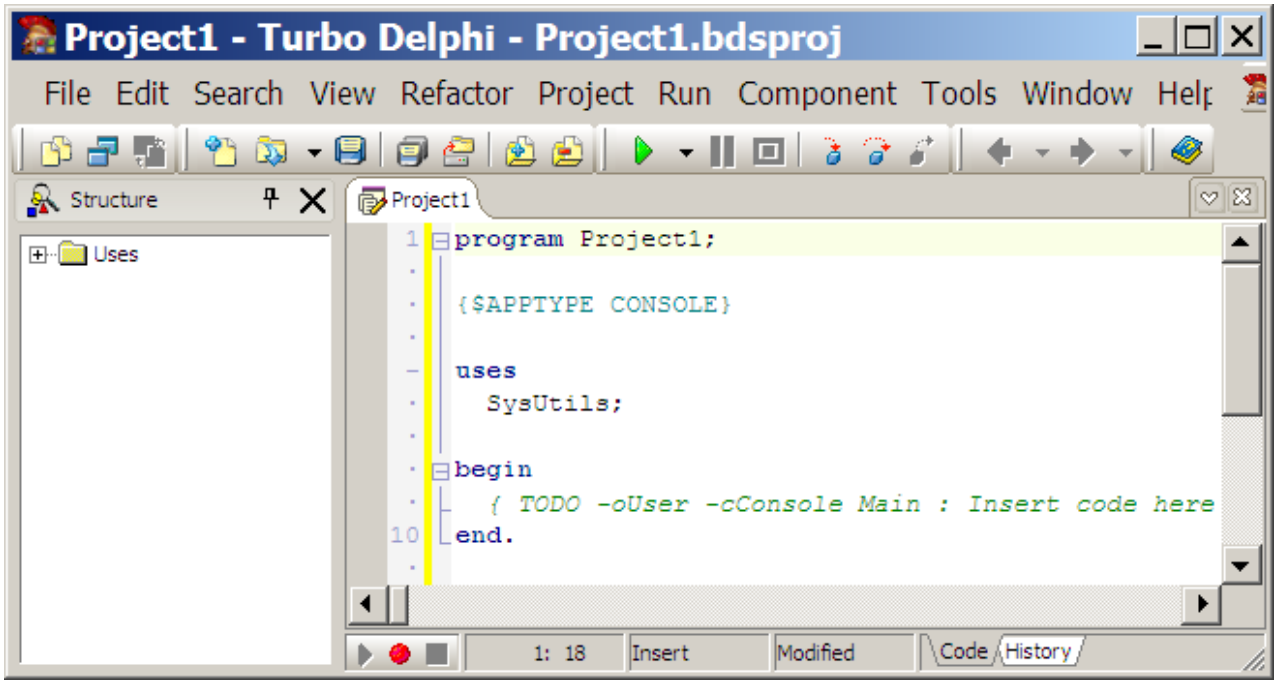

Рисунок 4 – Вид окна среды с заготовкой консольного приложения и без неиспользуемых окон

2. В рабочей области на экране теперь два окна: панель структуры и вкладка многооконного редактора текстов программ.

В окне структуры отображается дерево объектов программы: переменных функций, типов, библиотек и т. п. Его используют для быстрого перехода к описанию какого–либо объекта. Если щелкнуть мышью по плюсику у папки, то мы увидим ссылку только на библиотеку **SysUtils**. А двойной щелчок по этой ссылке установит курсор на объявление этой библиотеки в программе.

3. Созданный проект следует сохранить. Чтобы избежать дублирования имен проектов, каждый отдельный проект нужно сохранять ***в отдельной папке***. Поэтому перед сохранением создайте новую папку для этого проекта, например, *Пример1*. Затем используйте пункт меню **File/Save**. На экране появиться диалог **Save Project1 as**. В дереве файлов найдите созданную папку, задайте имя программы в окне *Имя файла*, например, *Example1* и нажмите на кнопку **Сохранить** (см. рисунок 5). После этого, изменится как имя файла проекта, так и имя на вкладке многооконного редактора программ.

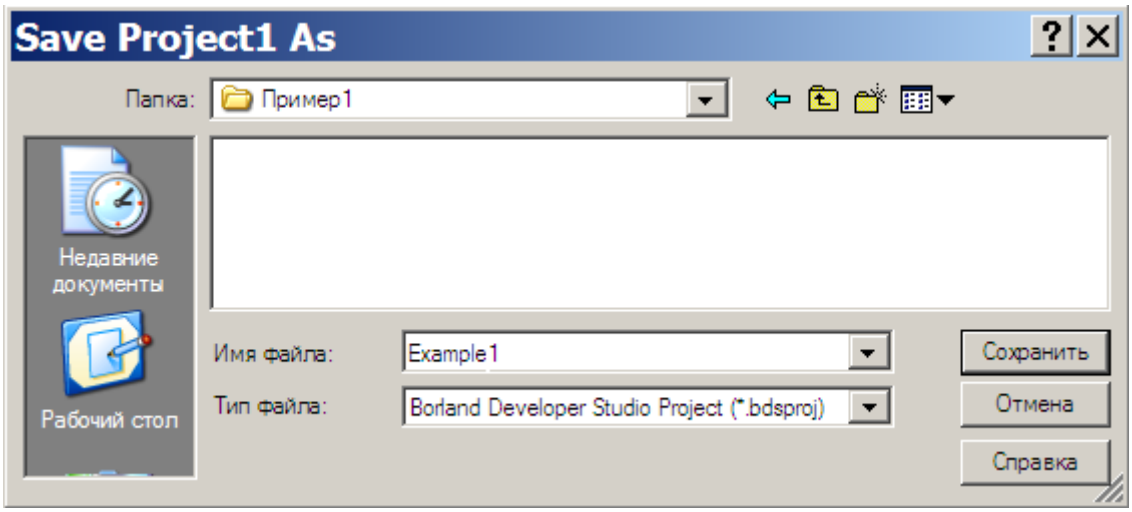

Рисунок 5 – Внешний вид диалога сохранения проекта

4. На переименованной вкладке Example1 многооконного редактора программ высвечивается заготовка консольного приложения. В этом окне набирают текст программы. При необходимости можно в других окнах открыть файлы с другими программами или данными с помощью пункта **File/Open**, но при этом проект может быть открыт только один.

Введите в открытое окно редактора программ выделенный текст:

```
program Example1;
{$APPTYPE CONSOLE}
uses  SysUtils;
Var A,B,C,D,E,X1,X2:Single;
Begin WriteLn('Input A, B, C:');
      ReadLn(A,B,C);
      D:= Sqr(B)- 4*A*C;
      if D>=0 then
           begin E:=2*A;
                 X1:= (-B+Sqrt(D))/E;X2:= (-B-Sqrt(D))/E;
```

```
                    WriteLn('X1=', X1:6:1, '  X2=', X2:6:1)
            end
        else WriteLn('No result');
        ReadLn;
    end.
```

5. Для выполнения программы следует использовать кнопку быстрого вызова **Run** (зеленый треугольник на панели инструментов), пункт меню **Run/Run** или клавишу **F9**.

Если при вводе программы вы допустили синтаксические ошибки, то компилятор выдаст на экране в окне вывода **Messages** (в нижней части окна среды) на вкладке **Build** соответствующие сообщения. При этом курсор будет установлен в тексте на место первой ошибки (см. рисунок 6).

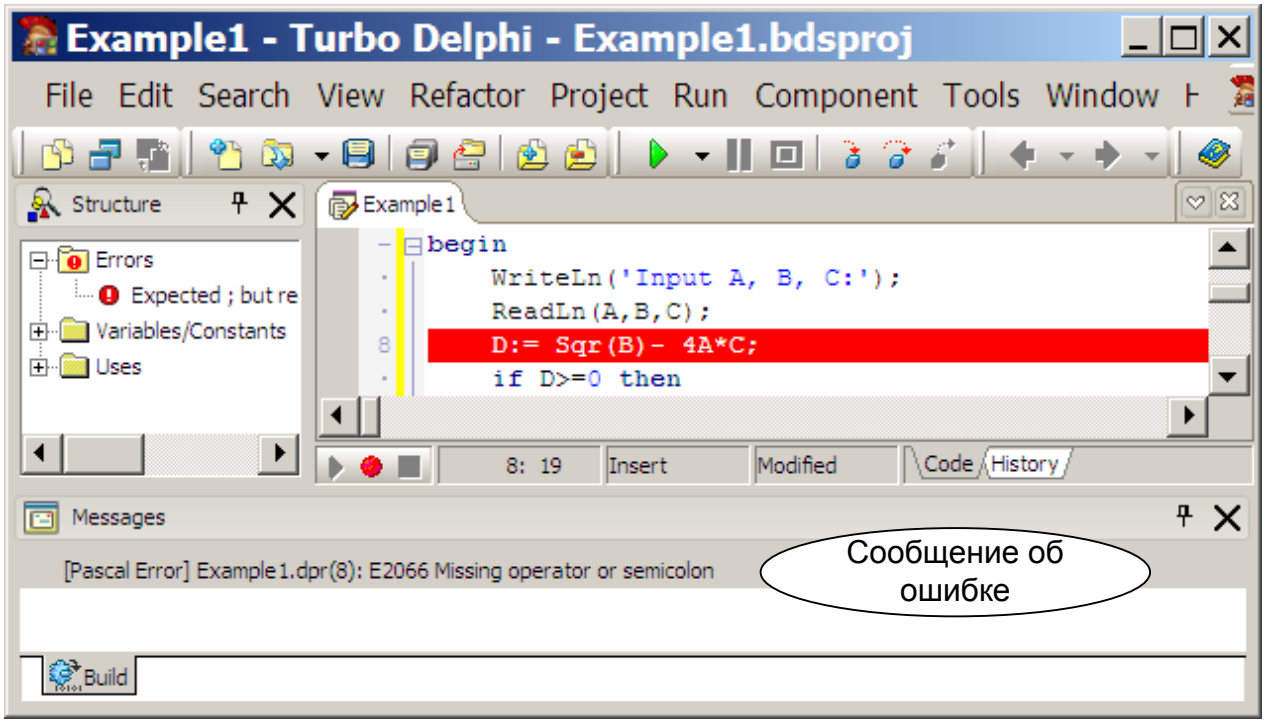

Рисунок 6 – Вид окна при обнаружении ошибки компиляции

Для перехода на строку, содержащую следующую ошибку, необходимо дважды щелкнуть мышью по соответствующему сообщению на вкладке **Build.**

6. Если программа введена верно, то на экране появится окно Консоль, в которое будет выведен запрос на ввод чисел. Числа вводят через пробел или каждое число в своей строке, нажимая после ввода числа клавишу Enter. После ввода чисел в то же окно будет выведен результат (см. рисунок 7).

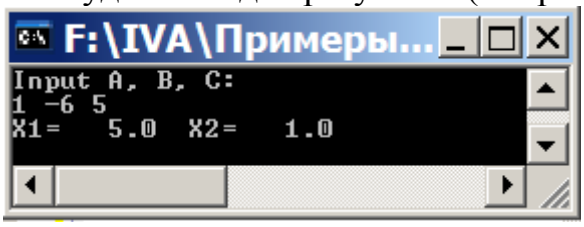

Рисунок 7 – Окно Консоли с выведенными результатами

При этом вид окна среды за окном консоли изменится, на нем появятся панели режима отладки программы: Call Stack – стек вызовов, Watch List –

список отслеживаемых значений переменных, Local Variables – значения локальных переменных, Event Log – перечень событий. Особенности работы с этими окнами будут рассмотрены далее.

Задание 2

Изучить диагностические сообщения Turbo Delphi.

Поочередно внося ошибки в программу, фиксируйте сообщения об ошибках в отчете в специальной таблице, представленной ниже. Классифицируйте ошибку, расшифруйте сообщение системы и определите этап выполнения программы (компиляция, компоновка или выполнение), на котором была обнаружена данная ошибка.

Таблица – Диагностические сообщения

	Ошибка	Физический смысл ошибки	Проявление ошибки	Расшифровка сообщения	Этап
1.	*Var AB, C, X1, X2, D, E;*	Вместо переменных A и B описана переменная AB	Получено сообщение: Undeclared identifier: 'A'	Не объявлена переменная A	Компиляция
2.	Readln(A,B,C)				
3.	E=2*A;				
4.	*D:=spr(B)-4*A*C;*	Вызвана несуществующая функция			
5.	Исходные данные: 0 1 3				
6.	Исходные данные: 1 1 3				
7.	{ *E:=2*A;* }	Пропущена строка вычислений			
8.	*Read(A,C);*	Не определено (не введено) значение B			

Примечание – В процессе работы в среде Turbo Delphi создаются следующие файлы:

<Имя проекта>**.bdsproj –** файл настроек Borland Developer Studio Project File;

<Имя проекта>**.bdsproj.local –** файл настроек Delphi

<Имя проекта>**.dpr** – исходный файл;

<Имя проекта>**.cfg** – файл конфигурации – опции компилятора и компоновщика;

<Имя проекта>**.exe** – исполняемый файл.

Кроме этого будет создана скрытая папка **__history**, которая будет хранить копии измененных файлов.

Если в процессе работы создаются библиотечные файлы (Unit), то они будут иметь расширение <Имя файла>**. pas**. Все файлы будут созданы в той папке, где был сохранен и сам проект (см. пункт 3 выше)

Для переноса программы с машины на машину достаточно переписать файлы с расширениями **dpr** и **pas** (если они создавались), остальные будут пересозданы в процессе загрузки и запуска приложения**.**

Средства отладки Delphi

Компилятор и компоновщик находят не все ошибки программы. После них в программе могут остаться логические ошибки, ошибки в формулах и т.д. Для локализации этих ошибок обычно осуществляют пошаговое выполнение программы с начала или с заданной точки с одновременным контролем значений всех или выбранных переменных.

Управление пошаговым выполнением программы. Для входа в режим пошагового выполнения необходимо запустить программу, используя пункт меню **Run/Step Over** (**F8**) или **Run/Trace Info** (**F7**). Первый означает, что необходимо выполнить шаг, не заходя в подпрограммы, вызываемые на данном шаге. Второй предполагает, что в подпрограмму надо зайти. Если на данном шаге подпрограммы не вызываются, то соответствующие режимы неразличимы. В дальнейшем используется тот пункт меню, который вызывает необходимую операцию. Кроме того, в состоянии отладки можно:

Run/Run Until Return (**Shift+F8**) – завершить выполнение под-программы;

Run/Run (**F9**) – выполнить программу до конца;

Run/Program Reset (**Ctrl+F2**) – прекратить процесс отладки.

Для входа в пошаговый режим можно также использовать пункт меню **Run To Cursor** (**F4**), при выборе которого программа выполняется до строки, в которой установлен курсор.

Просмотр значений переменных в окне Watch. В режиме пошагового выполнения существует возможность просмотра значений переменных. Для этого следует перейти на панель **Watch List** (см. рисунок 8).

Для добавления имени переменной в окно просмотра можно вызвать контекстное меню, щелкнув правой кнопкой мыши в области окна **Watch List**, и выбрать в нем пункт **Add Watch**. На экране появляется окно **Watch Properties**. В этом окне вводим имя переменной, например, «D» (см. рисунок 9). Тип переменной среда определяет автоматически, однако, если переменную необходимо показывать в каком-либо особом виде, то тип следует определить, поставив точку напротив выбранного типа среди указанных на выделенной панели окна.

Теперь при остановке в любой точке программы в окне **Watch List** можно видеть значение переменной D в этот момент.

В процессе отладки значения переменных просчитывают вручную и сравнивают со значениями, получаемыми в процессе работы программы. Контекстное меню позволяет не только добавить переменную в окно наблюдения, но и скорректировать ее имя (**Edit Watch**) или указать способ отображения, а также удалить ее (**Delete Watch**).

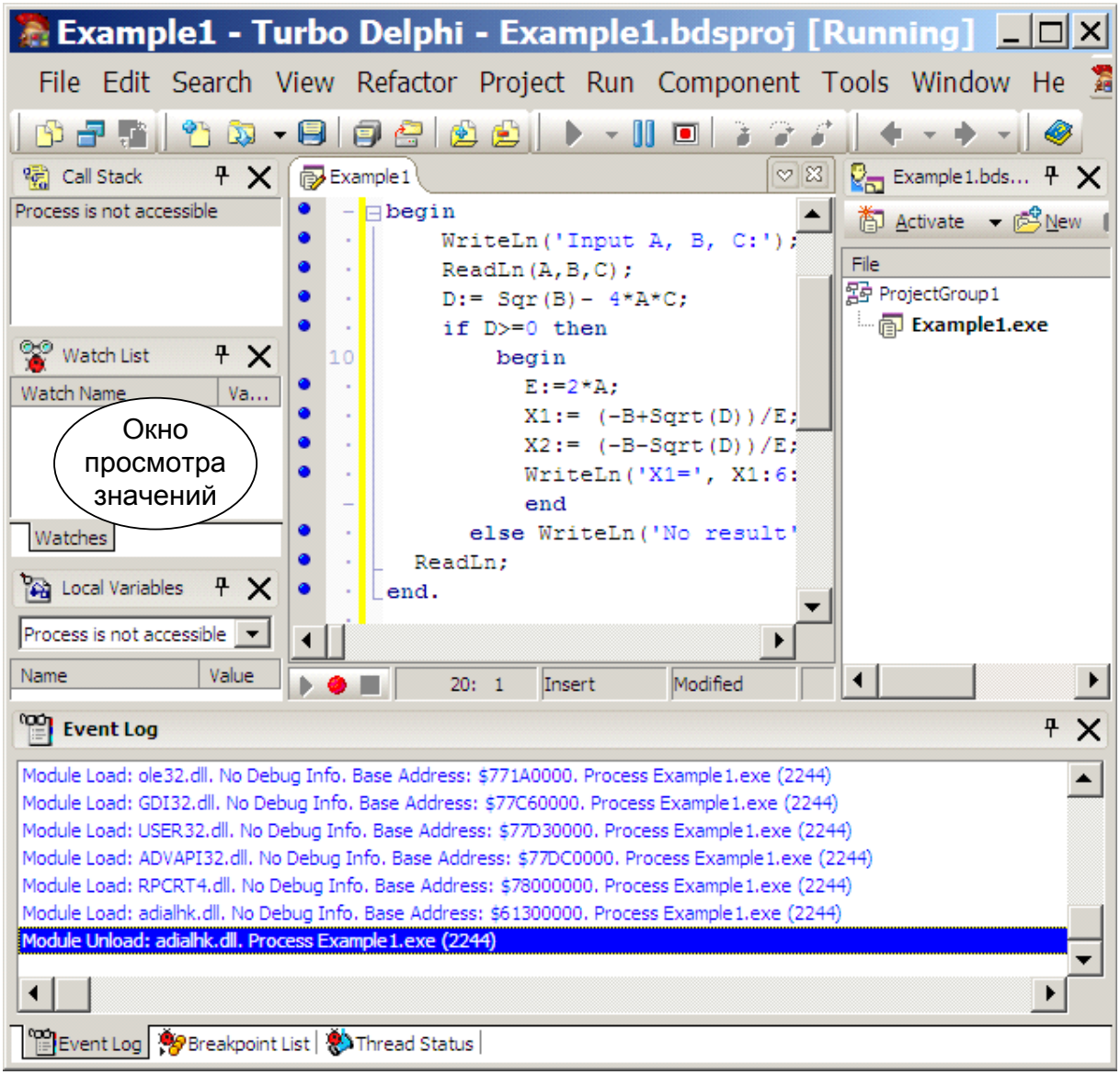

Рисунок 8 – Окно просмотра значений

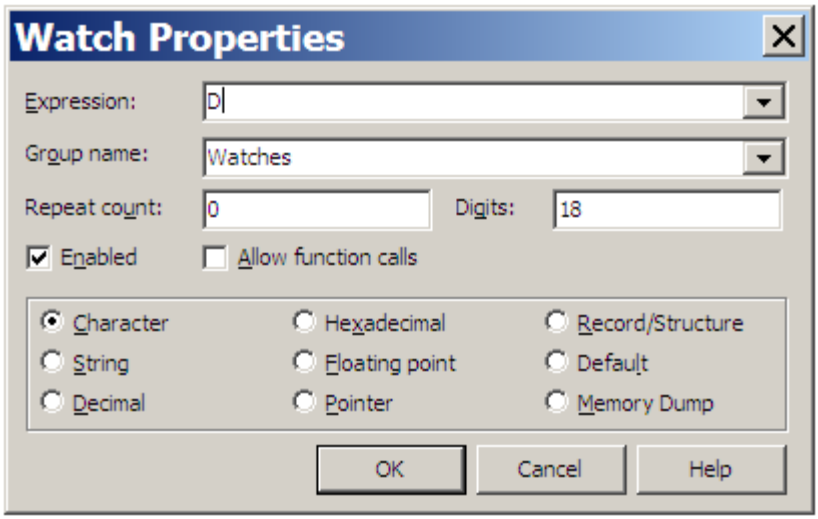

Рисунок 9 – Определение имени просматриваемой переменной

Установка контрольных точек. Для установки контрольных точек необходимо щелкнуть мышью по левому полю окна программы напротив оператора, перед выполнением которого необходимо остановить процесс вычислений. В этом место появляется точка, а строка оператора выделяется розовым (см. рисунок 10).

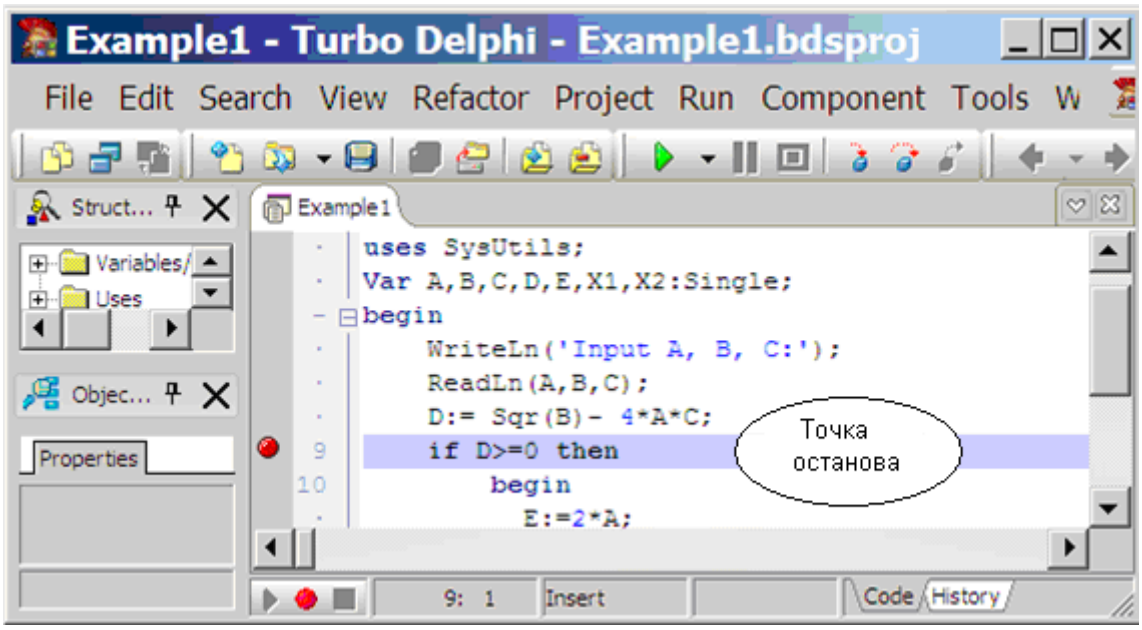

Рисунок 10 – Добавление/удаление точек останова

Теперь, если запустить программу, то ее выполнение остановится в указанной точке. Далее для поиска ошибки используют управление пошаговым выполнением, описанное в предыдущем пункте. Для отмены точки останова необходимо повторно щелкнуть по той же точке.

Просмотр текущих значений переменных. Текущие значения переменных во время выполнения программы можно просматривать, указывая во время остановки на соответствующую переменную курсором (см. рисунок 11).

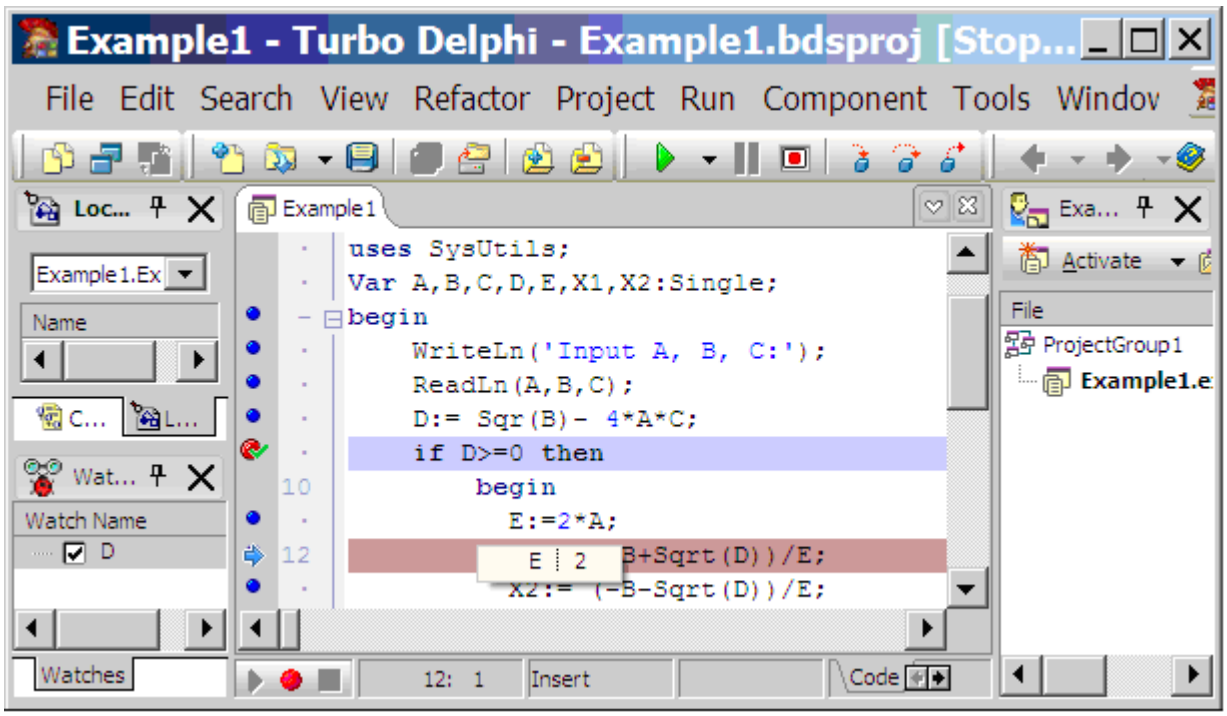

Рисунок 11 – Просмотр значений переменных, указанных курсором

Задание 3

Изучить средства отладки программ в среде Delphi.

Порядок работы:

1. Внесите в программу ошибку 7. Выполните пошаговую трассировку программы, отслеживая значения переменных A, B, C, D, E, x1, x2 в окне Watch. Какое значение имеет переменная E в момент вычисления корней уравнения? Почему?
2. Внесите в программу ошибку 6. Выполните пошаговую трассировку программы, отслеживая значение переменных A, B, C, D, E, x1, x2. Какое значение имеет переменная D? В какой момент обнаруживается ошибка?
3. Установите точку останова перед вычислением дискриминанта. Выполните программу до точки останова. Просмотрите значения переменных, подводя к ним курсор мыши.
3. Ответы на вопросы и выводы занести в отчет.

Часть 2. Создание схем алгоритмов средствами Microsoft Visio и OpenOffice Draw

Теоретический материал и задания по второй части лабораторной работы см. в методических указаниях Р.С. Самарев. Создание схем алгоритмов средствами Microsoft Visio и OpenOffice Draw.

Контрольные вопросы

Защита лабораторной работы предполагает ответы на указанные вопросы.

1. Что такое среда программирования, для чего она предназначена?

2. Какая информация высвечивается на экране при входе в среду Turbo Delphi.

3. Как создать новое консольное приложение? Ввести программу и запустить ее на выполнение?

4. Назовите файлы, которые создает среда при вводе программы, и поясните их назначение. Какие файлы содержат текст программы?

5. Перечислите возможности отладчика, встроенного в среду программирования. Как их использовать?

6. Что такое схема алгоритма и для чего она строится?

7. Какие средства могут использоваться для рисования схем алгоритма?

8. Как вставить схему алгоритма в текстовый документ?

Требования к отчету

Задание по лабораторной работе считается выполненным, если преподаватель принял программу и отчет.

Отчет должен выполняться на любой бумаге формата А4 или А5 в том числе в тетрадях или на тетрадных листах. Если отчет выполняется на отдельных тетрадных листах, то они должны быть аккуратно обрезаны по линии подшивки и скреплены. Неаккуратно выполненные, оборванные или грязные отчеты не принимаются.

Все записи в отчете должны быть либо напечатаны на принтере, либо разборчиво выполнены от руки синей или черной ручкой (карандаш – не допускается). Схемы также должны быть напечатаны при помощи компьютера или нарисованы с использованием чертежных инструментов.

Каждый отчет должен иметь титульный лист на котором указывается:

а) наименование факультета и кафедры;

б) название дисциплины;

в) номер и тема лабораторной работы;

г) фамилия преподавателя, ведущего занятия;

д) фамилия, имя и номер группы студента;

е) номер варианта задания.

Отчет по лабораторной работе № 1 должен включать следующую информацию:

1) номер и текст задания;

2) текст вводимой программы;

последовательность действий по созданию консольного приложения;

3) таблицу диагностических сообщений и пояснения к ней;

4) последовательность действий при отладке программы и результаты использования средств отладки.

5) схему алгоритма, нарисованную в одном из предлагаемых графических пакетов и вставленную в текст отчета.

6) выводы по работе.

Приложение А

Правила поведения во время выполнения лабораторных работ по дисциплине «Основы программирования»

1. Посещение лабораторных работ является обязательным. В компьютерный зал не допускаются студенты в верхней одежде, с едой и напитками.
2. Не разрешается в учебное время ходить по залу, громко разговаривать и играть в компьютерные игры.
3. Студенты, не выполняющие настоящие правила, от занятий отстраняются и будут допущены до следующих занятий только при наличии объяснительной записки с визой заместителя декана I курса.