Глава 34. Visual Basic

- Загрузка и выполнение примеров приложений Visual Basic
- Работа с редактором Visual Basic
- Использование объектов AutoCAD
- Свойства объектов
- Применение методов AutoCAD
- Основы Visual Basic

Язык BASIC, являющийся предшественником и основой Visual Basic, был создан в 1963 году. Предполагалось, что BASIC станет простым и понятным языком программирования, рассчитанным на широкий круг пользователей. Язык BASIC широко использовался для создания программ для микрокомпьютеров с момента их появления во второй половине 1970-х годов. Развитие BASIC продолжалось до появления операционной системы Windows. В 1991 году вышла в свет первая версия языка Visual Basic (VB), за которой вскоре последовали и другие. Последняя версия с номером 6 сохранила идеологию языка BASIC, позволяющую даже начинающему программисту создавать графические пользовательские интерфейсы. При этом сложность программирования значительно снизилась по сравнению с первыми версиями Visual Basic. Появление и постоянное совершенствование Visual Basic привлекло внимание многих программистов, использующих процедурные языки наподобие C, поскольку оказалось, что Visual Basic позволяет решать некоторые задачи гораздо быстрее и компактнее. Еще одним достоинством Visual Basic, оцененным многими инженерами старшего поколения, является возможность включения почти всех инструкций классического BASIC в модули Visual Basic (за небольшими исключениями, в основном касающимися операций ввода-вывода). Таким образом, программы, написанные 10–15 лет назад, можно использовать в качестве основы для новых разработок.

Компания Autodesk первой получила лицензию Microsoft на использование продукта Visual Basic for Applications, появившегося в 14-й версии AutoCAD. Подобному примеру последовали и другие крупные компании, специализирующиеся на разработке программного обеспечения. Средства Visual Basic for Applications идентичны или очень близки к своим аналогам из Visual Basic. Программистам, имеющим опыт создания макросов в Microsoft Office, необходимо лишь получить небольшие дополнительные сведения о специфичных функциях работы с AutoCAD.

Возможность использования языка Visual Basic означает возможность обеспечить взаимодействие с приложениями Microsoft Excel, Microsoft Word, Microsoft Access и другими посредством *средств автоматизации ActiveX*. Средства автоматизации ActiveX позволяют оперировать средствами и объектами AutoCAD при помощи

любого приложения, поддерживающего ActiveX. Возможно и обратное взаимодействие: средства и объекты, принадлежащие, например, Microsoft Excel, доступны из программы AutoCAD. Заметим, что язык AutoLISP не поддерживает подобного взаимодействия между приложениями. Для того чтобы использовать AutoCAD совместно с другими приложениями, поддерживающими ActiveX, нужно «уведомить» эти приложения о том, что AutoCAD установлен на компьютере вместе с библиотекой объектов (object library).

В настоящее время миллионы программистов используют язык Visual Basic. Еще большее число людей использовали «обычный» BASIC. При написании данной главы мы полагали, что вы уже знакомы с базовыми концепциями языка BASIC (аналогичными тем, которые описаны в этой книге для языка AutoLISP) и обладаете некоторыми вспомогательными навыками работы, такими как поиск описания и синтаксиса команд в справочной системе.

В данной версии AutoCAD можно ссылаться на макросы, находящиеся в других проектах, а также одновременно загружать несколько макросов. Кроме того, часто используемые макросы и функции можно вынести в отдельные библиотеки. Справочная система AutoCAD 2004 обладает расширенными возможностями по сравнению с предыдущими версиями AutoCAD.

Объекты

Для реализации своих возможностей Visual Basic использует *объекты*. К объектам относятся рисунки, документы, геометрические фигуры (окружности, линии и т. п.), элементы управления пользовательского интерфейса, позволяющие организовать взаимодействие пользователя с программой посредством ввода-вывода информации и т. д. К «визуальной» части Visual Basic относятся элементы пользовательского интерфейса, такие как флажки, кнопки, полосы прокрутки и т. д., которые видимы на экране. При создании интерфейса эти элементы перетаскиваются из панели инструментов на специальный объект, называемый *формой*. При добавлении к форме нового элемента код, необходимый для работы данного элемента, автоматически добавляется к объекту формы. Необходимо отметить, что значительная часть объектно-ориентированных средств программирования, присущих, например, C++, не поддерживается Visual Basic, поэтому этот язык следует относить скорее к объекто-использующим языкам, чем к объектно-ориентированным. Однако отсутствие этих объектно-ориентированных средств с лихвой восполняется инструментарием и средой разработки.

Функции, называемые в Visual Basic *методами*, описаны в библиотеке объектов AutoCAD и служат для выполнения некоторого действия над объектом, например рисования линии на чертеже. *Свойства* — это функции, которые устанавливают или возвращают параметры, описывающие состояние объекта. Так, например, линия на рисунке может быть охарактеризована цветом, типом, длиной и т. п. Это означает, что цвет, тип и длина являются свойствами линии. Пользователи AutoCAD знакомы со свойствами объектов благодаря палитре PROPERTIES редактора чертежей.

Добавление графических объектов

Для создания геометрических фигур в пространстве модели, пространстве листа или в блоке используются методы добавления графических объектов, такие как `AddCircle`, `AddLine`, `AddArc` и `AddText`. Представьте себе, что вместо рисования текущего чертежа в пространстве модели вы добавляете к нему геометрический объект. Перед названием метода указывается объект, к которому он применяется. Имя метода отделяется от имени объекта точкой; такая же нотация применяется для указания свойств объектов.

Метод AddCircle

Метод `AddCircle` добавляет объект типа «окружность» с заданными координатами центра и радиусом. Эти аргументы метода являются обязательными. Для других способов построения окружностей (например, путем задания двух или трех точек или двух касательных и радиуса) в Visual Basic не предусмотрены отдельные методы, и поэтому требуется дополнительный расчет координат центра и радиуса окружности. Формат метода `AddCircle` приведен ниже.

```
ThisDrawing.ModelSpace.AddCircle centerpoint, radius
```

или

```
Set Circle1 = ThisDrawing.ModelSpace.AddCircle(centerpoint, radius)
```

где `centerpoint` — координаты центра окружности в виде вектора чисел с двойной точностью, `radius` — радиус окружности, число с двойной точностью.

ПРИМЕЧАНИЕ

> При использовании первой из указанных синтаксических форм AddCircle после имени метода обязательно должен следовать пробел. Другая форма записи, использующая аргументы метода, заключенные в круглые скобки, предназначена для присваивания имени создаваемому объекту, что позволяет с помощью этого имени обращаться к объекту в других фрагментах программы. В примерах, приведенных ниже, point1 и point2 — это имена объектов типа «точка», описанных с помощью вектора, состоящего из двух числовых значений с двойной точностью. Возможный способ определения объектов point1 и point2 приведен в примере 1. Знак # в языке BASIC означает число с двойной точностью.

Пример применения метода `AddCircle`:

```
ThisDrawing.ModelSpace.AddCircle point 1, 3#
Set Circle2 = ThisDrawing.ModelSpace.AddCircle(point 2, 4#)
```

Метод AddLine

Метод `AddLine` добавляет объект типа «линия». Его аргументами являются векторы координат начальной и конечной точек линии. Синтаксис метода `AddLine` схож с

синтаксисом метода AddCircle:

```
ThisDrawing.ModelSpace.AddLine firstpoint, secondpoint
```

где firstpoint и secondpoint — векторы с двойной точностью.

Примеры применения метода AddCircle:

```
ThisDrawing.ModelSpace.AddLine point1, point2
Set Line1 = ThisDrawing.ModelSpace.AddLine(point1, point2)
```

ПРИМЕЧАНИЕ

Вторая форма записи, использующая круглые скобки, может быть применена к любому из методов добавления объектов. Цель такой записи всегда одна — связать объект с именем для дальнейшего использования в программе. Неудобство, связанное с применением этой формы записи, состоит в том, что определяемый объект должен быть объявлен с помощью инструкции объявления переменных Dim в секции General.

Метод AddArc

Синтаксис метода AddArc выглядит следующим образом:

```
ThisDrawing.ModelSpace.AddArc ctrpt, radius, StartAng, EndAng
```

где ctrpt — координаты центра, вектор с двойной точностью; radius — радиус, число с двойной точностью; StartAng и EndAng — соответственно начальный и конечный угол в радианах, числа с двойной точностью.

Пример применения метода AddArc:

```
ThisDrawing.ModelSpace.AddArc point1, 4#, 0#, 1.570796327
```

Метод AddText

Аргументами метода AddText являются текстовая строка, координаты вставки и высота текста. Синтаксис метода AddText выглядит следующим образом:

```
ThisDrawing.ModelSpace.AddText textString$, point1, textHeight
```

где text — выводимая текстовая строка, point1 — позиция, начиная с которой выводится текст, вектор с двойной точностью; textHeight — высота текста, положительное число с двойной точностью.

Пример применения метода AddText:

```
ThisDrawing.ModelSpace.AddArc ".063 TYP, 4 PLACES", point1, 0.25#
```

Использование справки для получения

информации об объектах и свойствах

В комплект Visual Basic входит мощное средство поддержки разработчиков, называемое VBA Integrated Development Environment, которое содержит развитую справочную систему с примерами, иллюстрирующими синтаксис и применение различных свойств и методов языка. Кроме того, имеется общая информация о методах и ключевых словах Visual Basic, а также примеры применения конструкций Visual Basic для других приложений, например для Excel или Word. Окно справочной системы AutoCAD 2004 показано на рис. 34.1.

Разделы справки по Visual Basic можно вызвать как из справочной системы графической среды AutoCAD, так и из интегрированной среды разработчика (Integrated Design Environment, IDE). В списке, показанном на рис. 34.1, следует выбрать пункт VBA And ActiveX Automation (Visual Basic и средства автоматизации ActiveX). Чтобы найти необходимый раздел, можно воспользоваться вкладками Index (Содержание) и Find (Поиск), расположенными в левой части окна. Еще одним способом вызова справки в интегрированной среде является использование средства Object Browser, доступ к которому осуществляется через меню View (Вид). В раскрывающемся списке следует выбрать пункт AcadProject. В левом окне следует выбрать объект ThisDrawing, а в правом — ModelSpace (рис. 34.2). Щелкнув на кнопке с изображением знака вопроса на панели инструментов или нажав клавишу F1, можно попасть в окно, обеспечивающее доступ к разделу справки ModelSpace Collection, содержащему описания различных свойств и методов, а также примеры их применения. Нажатие клавиши F1 приводит к выдаче справки для элемента, название которого указано в текущей строке кода.

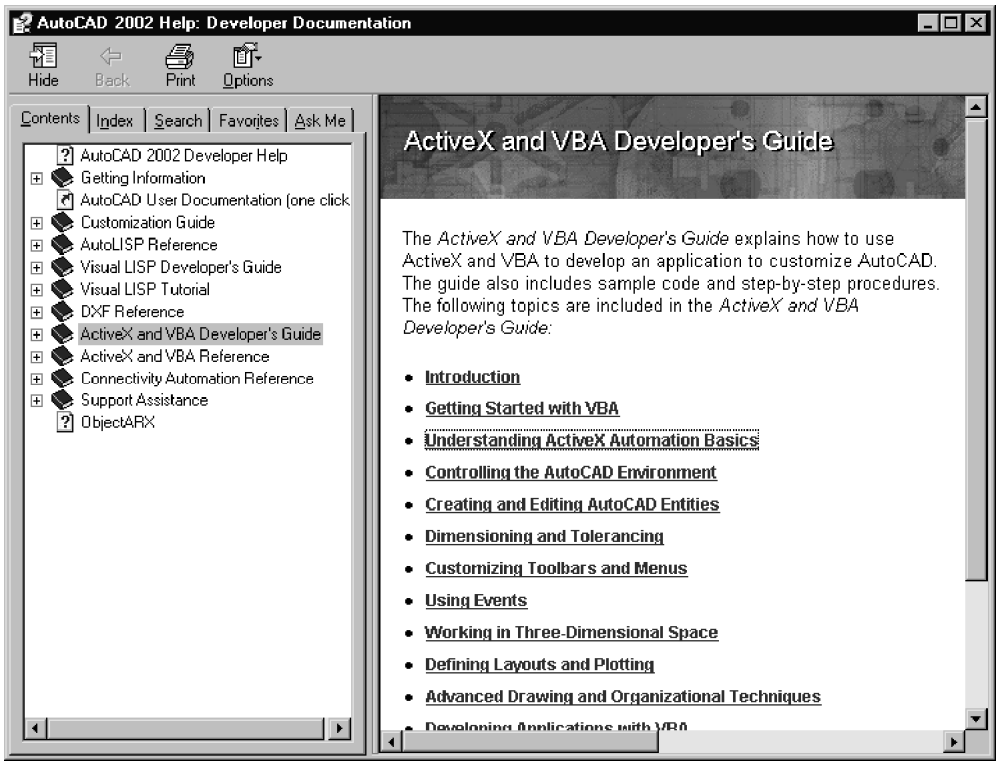


Рис. 34.1. Справочная система AutoCAD 2004

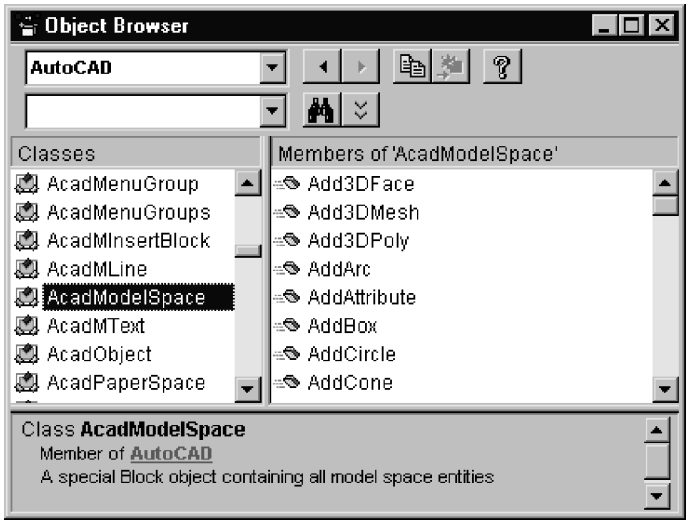


Рис. 34.2. Средство просмотра объектов Object Browser

Загрузка и сохранение проектов Visual Basic

Для того чтобы загрузить или запустить на выполнение проект в программе AutoCAD, следует выбрать команду меню Tools ▸ Macro ▸ Visual Basic (рис. 34.3).

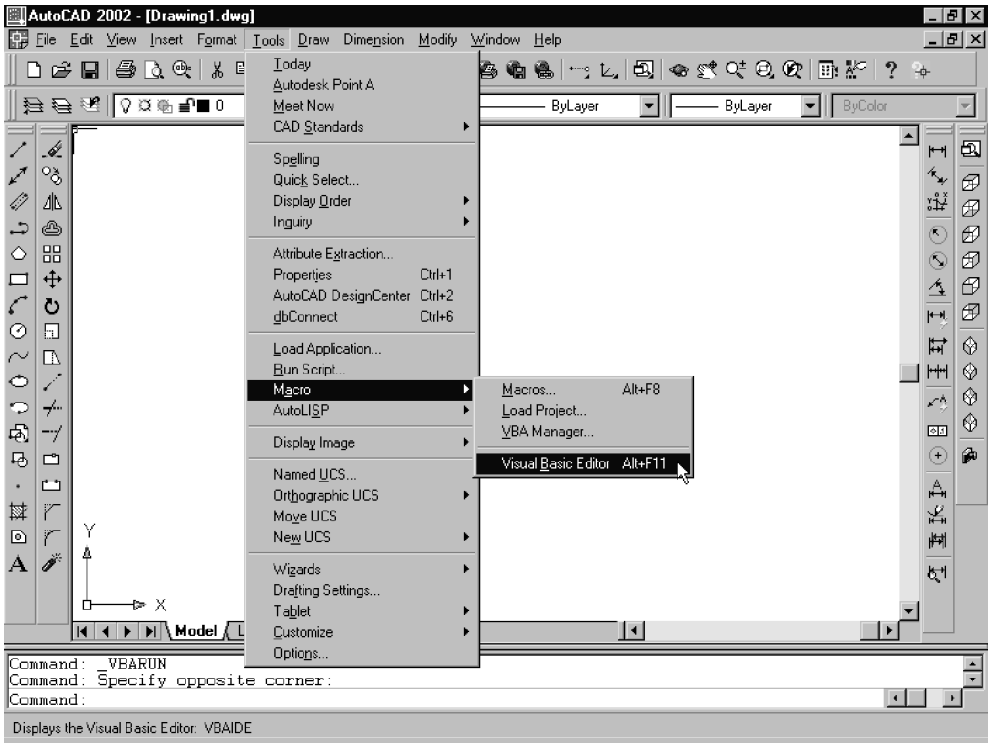


Рис. 34.3. Запуск редактора Visual Basic с помощью меню Tools

На экране появляется интегрированная среда разработки (IDE). Это приложение состоит из нескольких окон, каждым из которых можно управлять независимо от других, например изменять его размер, скрывать, отображать и т. д. На рис. 34.4 показана среда разработки с загруженным проектом примера 1. На экране отображаются шесть окон. В левом верхнем углу находится окно проекта Project Explorer, позволяющее переключаться между остальными окнами. В верхней части окна проекта расположены значки View Code, View Object, Toggle Folders. В нижнем левом углу находится окно Properties (Свойства), с помощью которого можно просматривать и изменять свойства объектов.

В центре верхней части экрана расположено окно формы (UserForm), в котором при помощи панели инструментов создаются пользовательские интерфейсы. Под окном формы находится окно кода создаваемой формы. Код формы представляет собой набор процедур, каждая из которых запускается при наступлении какого-либо события, например, щелчка мыши. Ниже расположено окно кода модуля (Module), в котором, как правило, содержатся тексты математических процедур и функций.

Оставшиеся два окна Immediate и Watches предназначены для отладки.

Перед тем как начать отладку написанной вами программы на Visual Basic, будет не лишним сначала сохранить ее в файле. Для этого используется команда Save (Сохранить) меню File. Пункты меню можно выбирать, нажав клавишу Alt и букву, подчеркнутую в названии нужного пункта. Проекты Visual Basic в AutoCAD имеют расширение dvb.

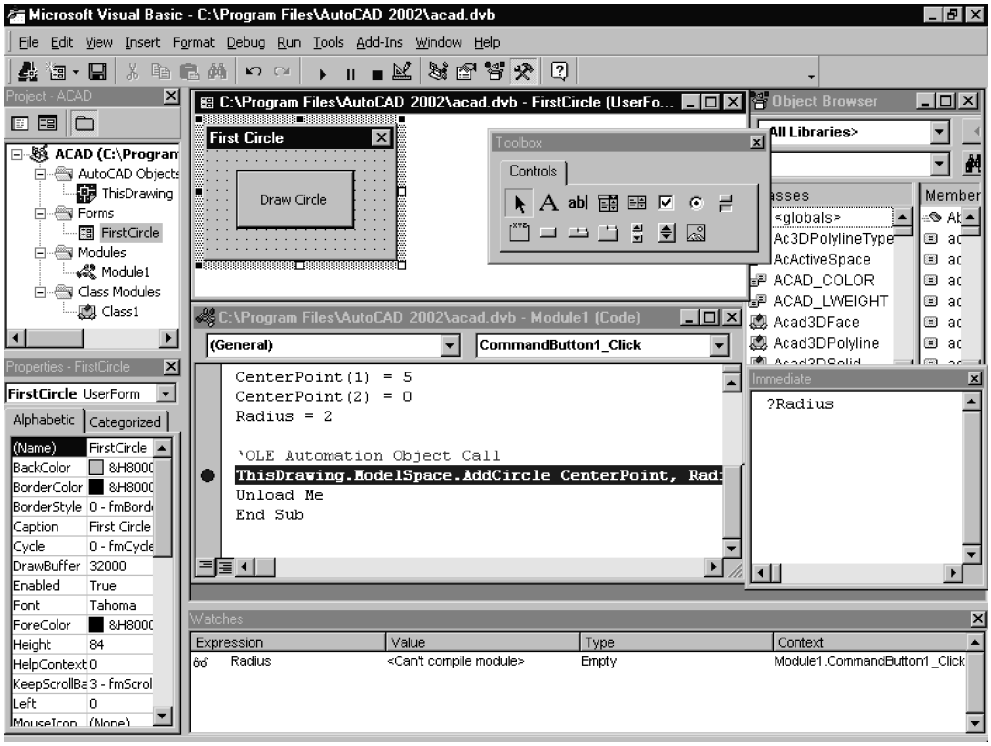


Рис. 34.4. Запуск редактора Visual Basic с помощью меню Tools

Пример 1

Напишем программу, которая рисует окружность с центром в точке (5, 5, 0) и радиусом 2 (рис. 34.5).

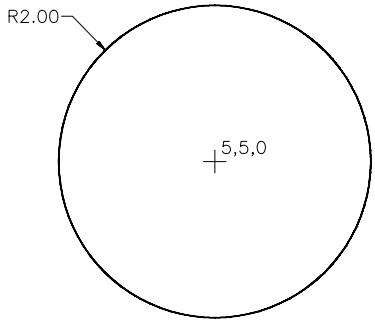


Рис. 34.5. Окружность для примера 1

Добавьте в проект модуль и форму (вид формы показан на рис. 34.6). Для этого выберите команды меню Module и UserForm соответственно. Будут созданы модуль с именем Module1 и форма с именем UserForm1. Щелкните на окне формы, чтобы оно стало активным. Теперь можно добавить в окно формы любой элемент управления, выбрав его на панели инструментов. Например, второй значок слева во втором ряду панели инструментов позволяет добавить в форму новую кнопку (рис. 34.7).

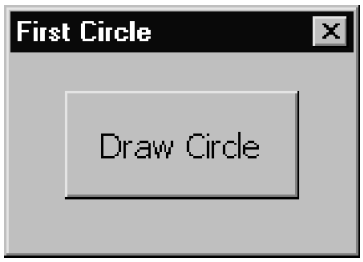


Рис. 34.6. Форма для примера 1

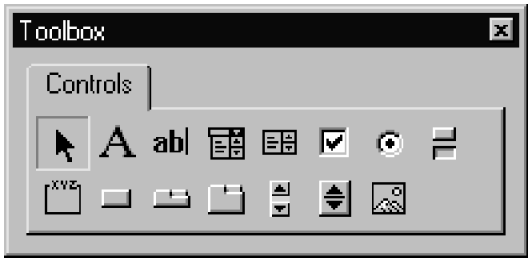


Рис. 34.7. Панель инструментов Toolbox для создания элементов интерфейса

Теперь измените размеры окон модуля и формы по своему вкусу и создайте

заголовок кнопки, назвав ее Draw Circle. Это можно сделать с помощью окна свойств, изображенного в нижнем левом углу на рис. 34.4.

Проект — это совокупность форм, элементов управления, модулей и программ на языке Visual Basic. Для того чтобы ваш проект был работоспособным, необходимо создать код, с помощью которого осуществляется управление элементами пользовательского интерфейса. Чтобы создать такой код для добавленной нами кнопки, дважды щелкните на ней мышью и введите необходимые инструкции в процедуру `CommandButton1_Click()`. Программы на языке Visual Basic являются программами, *управляемыми событиями*. Ниже приведен код, который следует ввести в качестве процедуры, выполняемой в ответ на щелчок пользователя на кнопке формы. Номера строк, расположенные справа от текста кода, не являются его частью и служат лишь для ссылок на соответствующие инструкции при пояснениях.

```
'Код UserForm1                                                    1
'Рисование окружности с центром (5, 5, 0) и радиусом 2           2
'по щелчку на кнопке Draw Circle                                  3
Private Sub CommandButton1_Click()                                4
Dim CenterPoint(0 To 2) As Double                                 5
Dim Radius As Double                                              6
'Данные                                                           7
CenterPoint(0) = 5                                                8
CenterPoint(1) = 5                                                9
CenterPoint(2) = 0                                               10
Radius = 2                                                       11
                                                                 12
'Вызов объекта автоматизации OLE                                 13
ThisDrawing.ModelSpace.AddCircle CenterPoint, Radius             14
Unload Me                                                        15
End Sub                                                          16
```

Пояснения

Строки 1–3

Три первые строки кода представляют собой комментарий, описывающий назначение программы. Наличие комментариев в программе делает ее более простой для понимания и модификации. Строка комментария должна начинаться с ключевого слова `Rem` или апострофа (`'`). При выполнении программы все строки комментария игнорируются.

Строка 4

```
Private Sub CommandButton1_Click()
```

Эта строка обозначает начало процедуры `CommandButton1_Click()`. Процедура `CommandButton1_Click()` выполняется каждый раз, когда пользователь щелкает

мышью на созданной нами кнопке.

Строки 5 и 6

```
Dim CenterPoint(0 To 2) As Double
Dim Radius As Double
```

Данные строки представляют собой объявления переменных с двойной точностью, которые мы позднее используем в качестве аргументов метода `AddCircle`. Если бы мы не указали тип `Double` явно, то по умолчанию переменным были бы присвоены типы `Variant`, что не позволило бы использовать их в качестве аргументов метода `AddCircle`.

Строки 8–11

```
CenterPoint(0) = 5
CenterPoint(1) = 5
CenterPoint(2) = 0
```

Здесь мы присваиваем значения созданным переменным.

Строка 14

```
ThisDrawing.ModelSpace.AddCircle CenterPoint, Radius
```

Здесь мы применяем метод `AddCircle` к объекту `ModelSpace`, который является частью объекта `ThisDrawing`. Обратите внимание на обязательный пробел между ключевым словом `AddCircle` и первым аргументом `CenterPoint`.

Строка 15

```
Unload Me
```

Это метод, выгружающий форму UserForm1 из памяти и переключающий фокус обратно в AutoCAD.

Строка 16

```
End Sub
```

Как и строка 4, данная инструкция генерируется автоматически.

Далее, в проект необходимо включить код модуля. В окне кода модуля введите следующий текст:

```
'Основные объявления
Sub DrawCircle()
UserForm1.Show
End Sub
```

Пояснения

Целью данной процедуры служит создание имени макроса DrawCircle, которое появится в нижнем списке диалогового окна Macro (Макрос). Созданный код относится к объекту Module1. Еще одним способом запустить проект является команда Run Sub ▸ UserForm меню Run.

Методы GetPoint, GetDistance и GetAngle

Метод GetPoint

Метод GetPoint позволяет задавать точку на плоскости посредством указания ее координат (двух или трех соответственно). Координаты могут быть заданы как вводом с клавиатуры, так и при помощи указателя мыши. Синтаксис метода GetPoint выглядит следующим образом:

```
P = ThisDrawing.Utility.GetPoint([Point], [Prompt])
```

где Point — необязательная точка, используемая для показа эластичной вспомогательной линии; Prompt — необязательное приглашение, которое будет выведено на экран.

Примеры применения метода GetPoint:

```
pnt1 = ThisDrawing.Utility.GetPoint("Введите первую точку")
Pt2 = ThisDrawing.Utility.GetPoint(Pnt1, "Введите вторую точку")
```

Метод GetDistance

Метод GetDistance позволяет ввести расстояние в командной строке (расстояние от заданной точки или двух точек) а затем возвращает расстояние в виде числа типа Double. Синтаксис метода GetDistance имеет следующий вид:

```
d = ThisDrawing.Utility.GetDistance([point], [prompt])
```

где point и prompt — те же необязательные аргументы, что используются в методе GetPoint.

Метод GetAngle

Метод GetAngle позволяет задать величину угла непосредственно с клавиатуры, либо выделив пару точек. В последнем случае положительное направление горизонтальной оси принимается в качестве одной из сторон угла, вершиной угла полагается первая из выбранных точек, а второй стороной угла считается луч, исходящий из первой точки и проходящий через вторую точку. Если точка указана в виде аргумента, то она считается первой и служит вершиной угла. Метод GetAngle возвращает величину угла в радианах, а типом значения является Double. Синтаксис метода GetAngle имеет следующий вид:

```
ang = ThisDrawing.Utility.GetAngle([point], [prompt])
```

где ang — значение угла в радианах, point и prompt — те же необязательные аргументы, что используются в методах GetPoint и GetDistance.

Примеры применения метода GetAngle:

```
a1 = ThisDrawing.Utility.GetAngle(, "Введите угол конуса")
ang = ThisDrawing.Utility.GetAngle(pt1) 'pt1 определена заранее
ThisDrawing.Utility.GetAngle pt1, "Введите вторую точку для задания угла"
```

ПРИМЕЧАНИЕ

Результат вычисления величины угла зависит от установок, сделанных в AutoCAD. В частности, к таким установкам относятся значения системных переменных ANGBASE и ANGDIR. По умолчанию измерение углов производится исходя из следующих посылок: положительным направлением оси абсцисс считается горизонтальное, направленное вправо (направление часовой стрелки в 3 часа); это соответствует нулевому значению переменной ANGBASE. Значение переменной ANGBASE может быть любым в пределах от 0 до 360. Положительным направлением при измерении углов по умолчанию считается направление против часовой стрелки. Это задается при помощи системной переменной ANGDIR. Существует также метод GetOrientation, сходный с методом GetAngle, но игнорирующий действительные значения переменных ANGBASE и ANGDIR и использующий значения по умолчанию.

Пример 2

Создадим программу, которая будет рисовать треугольник с заданными сторонами P1, P2 и P3 (рис. 34.8). В этой программе будут применяться методы GetPoint и AddLine. Взаимодействие программы с пользователем будет осуществляться с помощью формы, представленной на рис. 34.9.

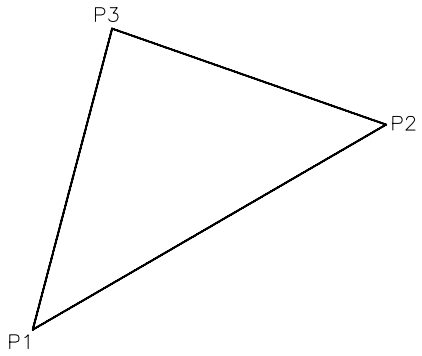


Рис. 34.8. Треугольник с вершинами, заданными пользователем

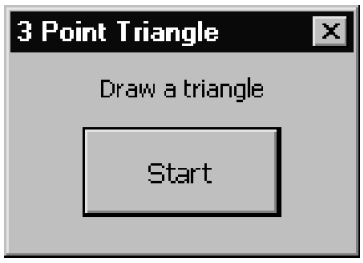


Рис. 34.9. Форма для примера 2

Выберите в меню Insert (Вставка) команды Module и UserForm, чтобы создать новый модуль и пользовательскую форму, соответственно. Для того чтобы создать форму, изображенную на рис. 34.9, перетащите в окно формы кнопку из панели инструментов. С помощью кнопки Label (Имя) вставьте нужный текст в элементы формы. Приведенный ниже код является листингом проекта для нашего примера. Номера строк, расположенные справа от текста кода, не являются его частью и служат лишь для ссылок на соответствующие инструкции при пояснениях, данных ниже.

```
'Рисование треугольника с вершинами,                                    1
'определенными пользователем                                            2
'событие наступает при щелчке мыши на кнопке с именем Start             3
'кнопка имеет имя Start                                                 4
'переменные pnt1, pnt2 и pnt3 по умолчанию имеют тип variant.           5
'функция возвращает точку в ПСК                                         6
Private Sub CommandButton1_Click()                                      7
UserForm2.Hide                                                          8
pnt1 = ThisDrawing.Utility.GetPoint(,"Введите координаты первой точки: ")  9
'возвращает вариантный вектор, т.к. метод GetPoint возвращает координаты  10
'точки в ПСК и строит эластичную линию от первой точки, если она задана   11
pnt2 = ThisDrawing.Utility.GetPoint(pnt1,"Введите координаты 2-й точки: ")12
'рисование первой стороны треугольника                                  13
ThisDrawing.ModelSpace.AddLine pnt1, pnt2                               14
pnt3 = ThisDrawing.Utility.GetPoint(pnt2,"Введите координаты 3-й точки: ")15
ThisDrawing.ModelSpace.AddLine pnt2, pnt3                               16
ThisDrawing.ModelSpace.AddLine pnt3, pnt1                               17
Unload Me                                                               18
End Sub                                                                 19
```

Приведем пояснения для данного кода.

Строки 1–6, 11 и 13

Эти строки являются комментариями, поясняющими смысл одной или нескольких инструкций, следующих за ними. Наличие комментариев в программе делает ее

более простой для понимания и модификации. Строка комментария должна начинаться с ключевого слова Rem или апострофа ('). При выполнении программы все строки комментариев игнорируются.

Строка 7

```
Private Sub CommandButton1_Click()
```

Эта строка обозначает начало процедуры CommandButton1_Click(). Процедура CommandButton1_Click() выполняется каждый раз, когда пользователь щелкает мышью на созданной нами кнопке.

Строка 8

```
UserForm1.Hide
```

Эта инструкция скрывает интерфейс пользовательской формы и переключает фокус в AutoCAD. В случае, если форма UserForm1 из предыдущего примера еще находится в памяти, следует изменить код формы текущего примера, указав инструкцию

```
UserForm2.Hide
```

поскольку по умолчанию для новой формы будет принято имя UserForm2, а не UserForm1.

Строки 9, 12 и 15

```
pnt1 = ThisDrawing.Utility.GetPoint(,"Введите координаты первой точки: ")
pnt2 = ThisDrawing.Utility.GetPoint(pnt1,"Введите координаты 2-й точки: ")
pnt3 = ThisDrawing.Utility.GetPoint(pnt2,"Введите координаты 3-й точки: ")
```

Метод GetPoint используется без указания вспомогательной точки в строке 15. Эта точка может быть указана программе путем ввода координат с клавиатуры или указания позиции мышью. Если метод применяется с дополнительной точкой, на экране возникает эластичная линия, указывающая положение курсора мыши.

Строки 14, 16 и 17

```
ThisDrawing.ModelSpace.AddLine pnt1, pnt2
ThisDrawing.ModelSpace.AddLine pnt2, pnt3
ThisDrawing.ModelSpace.AddLine pnt3, pnt1
```

Метод AddLine требует ввода двух аргументов, каждый из которых является вектором из трех значений с двойной точностью. Обратите внимание на пробел, разделяющий имя метода и аргументы.

Строки 18 и 19

```
Unload Me
End Sub
```

Эти инструкции выгружают форму из памяти, возвращают фокус в AutoCAD и завершают процедуру.

Код модуля 2 (Module 2)

В окне кода модуля необходимо ввести следующие инструкции:

```
Option Explicit   1
Sub Triangle()    2
UserForm1.Show    3
End Sub   4
```

Пояснения

Строка 1

```
Option Explicit
```

Данная инструкция требует, чтобы типы всех объявляемых переменных были указаны явно. Это позволяет избежать ошибок, связанных с неправильным использованием типов переменных, а также с опечатками при вводе инструкций. Переменным, тип которых не указан, будет присвоен тип `Variant`.

Строки 2–4

```
Sub Triangle()
UserForm1.Show
End Sub
```

Целью данной процедуры служит создание макроса с именем `Triangle`, которое будет отображено в нижнем списке диалогового окна Macro. Данный код относится к созданному нами модулю. В случае, если форма из предыдущего примера все еще загружена, то по умолчанию текущей форме присваивается имя UserForm2. В этом случае необходимо изменить приведенный код так, чтобы вместо имени UserForm1 использовалось имя UserForm2.

Методы PolarPoint и AngleFromXAxis

Метод PolarPoint

Метод `PolarPoint` определяет точку, находящуюся на заданном расстоянии и под определенным углом от указанной точки. Синтаксис метода `PolarPoint` имеет следующий вид:

```
P = ThisDrawing.Utility.PolarPoint(Point, Angle, Distance)
```

где `Point` — необязательная точка, используемая для показа эластичной вспомогательной линии; `Angle` — угол в радианах, число с двойной точностью; `Distance` — расстояние от заданной точки, число с двойной точностью.

Метод AngleFromXAxis

Метод `AngleFromXAxis` подсчитывает (в радианах) угол между линией, проходящей через две данные точки, и осью абсцисс. Синтаксис метода `AngleFromXAxis` выглядит следующим образом:

```
Ang = ThisDrawing.Utility.AngleFromXAxis(point1,point2)
```

где `point1` и `point2` — точки, задающие соответственно начало и конец линии.

Упражнение 1

Напишите программу на языке Visual Basic, которая будет рисовать линию между двумя заданными точками (рис. 34.10). Пользователь должен иметь возможность как задавать координаты точек с клавиатуры, так и выделять требуемые точки мышью на экране. На графическом экране должна появляться эластичная вспомогательная линия. Для взаимодействия пользователя с программой используйте форму, схожую с изображенной на рис. 34.11.

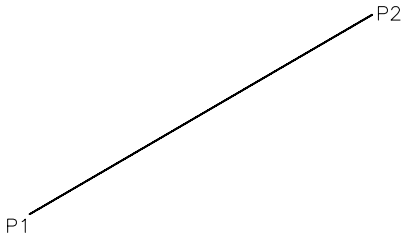


Рис. 34.10. Линия, концы которой заданы пользователем

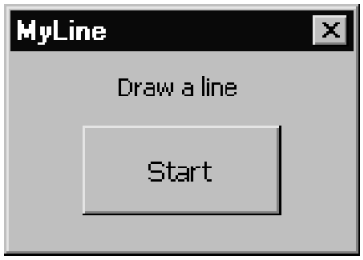


Рис. 34.11. Форма к упражнению 1

Пример 3

Создадим программу, которая будет рисовать треугольник, у которого задана одна сторона, длина другой стороны и угол между двумя сторонами (рис. 34.12).

Создаваемая программа будет использовать интерфейс, включающий два текстовых поля (рис. 34.13).

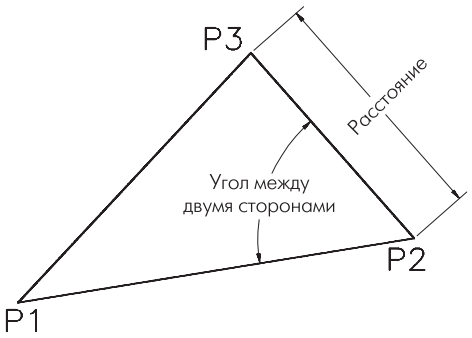


Рис. 34.12. Треугольник с заданными двумя сторонами и углом между ними

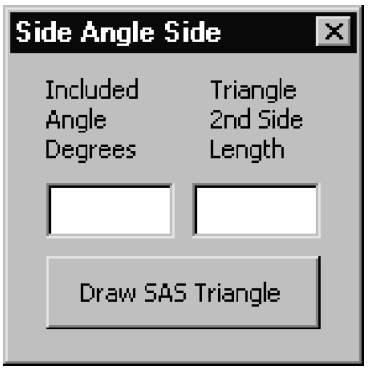


Рис. 34.13. Форма для примера 3

Чтобы создать текстовое поле, необходимо перетащить значок Textbox из панели инструментов в окно формы. Также следует создать кнопку и ввести для созданных элементов заголовки, по аналогии с предыдущими примерами. Кроме того, нам понадобится новый модуль, который следует создать с помощью команды Insert ▸ Module (Вставка ▸ Модуль). Если модули двух предыдущих примеров находятся в памяти, то по умолчанию новому модулю будет присвоено имя Module3.

Далее мы приведем листинг программы. Номера строк, расположенные справа от текста кода, не являются его частью и служат лишь для ссылок на соответствующие инструкции при пояснениях.

```
'объявления общих переменных                                    1
Const PI = 3.141592654                                          2
Public IncludedAngle As Double 'вводимый угол                   3
Public Angle As Double 'заданный угол                           4
Public Dist As Double 'длина второй стороны                     5
```

```
                                                                        6
'Рисование треугольника по двум сторонам и углу между ними             7
'Событие наступает при щелчке мыши на кнопке с именем Draw SAS Triangle 8
'Дополнительная возможность: ввод угла и длины второй стороны          9
'осуществляется с помощью текстовых полей.                            10
'Вводимый угол — это угол между заданной и вводимой стороной          11
Private Sub CommandButton1_Click()                                     12
UserForm3.Hide                                                         13
'p1 — это первая вершина, по умолчанию тип variant                    14
'p2 — это вторая вершина, по умолчанию тип variant                    15
p1 = ThisDrawing.Utility.GetPoint(,"Введите или выберите первую вершину:")16
p2 = ThisDrawing.Utility.GetPoint(,"Введите или выберите вторую вершину:")17
ThisDrawing.ModelSpace.AddLine p1,p2 'рисование стороны                18
If TextBox1.Text = "" Then                                             19
Angle = ThisDrawing.Utility.GetAngle(p2,"[Key ang. from+…]")           20
Else                                                                   21
Angle = ThisDrawing.Utility.AngleFromXAxis(p1,p2) + Included Angle     22
End If                                                                 23
If TextBox2.Text = "" Then                                             24
Dist=ThisDrawing.Utility.GetDistance(p2, "Введите расст. от [base point]")25
End If                                                                 26
p3 = ThisDrawing.Utility.PolarPoint(p2, Angle, Dist)                   27
ThisDrawing.ModelSpace.AddLine p2, p3                                  28
ThisDrawing.ModelSpace.AddLine p1, p3                                  29
Unload Me                                                              30
End Sub                                                                31
                                                                       32
Private Sub textBox1_Change()                                          33
IncludedAngle = PI - Val(TextBox1.Text)*PI / 180                       34
End Sub                                                                35
                                                                       36
Private Sub textBox2_Change()                                          37
Dist = Val(TextBox2.Text)                                              38
End Sub                                                                39
```

Пояснения

Строка 2

```
Const PI = 3.141592654
```

Объявление константы `PI`. Демонстрирует способ объявления констант.

Строки 3–5

```
Public IncludedAngle As Double 'вводимый угол
```

```
Public Angle As Double 'заданный угол
Public Dist As Double  'длина второй стороны
```

Эти переменные объявлены как открытые. Это означает, что любая функция из любого модуля может получить к ним доступ без указания переменной в качестве аргумента функции. Тип указанных переменных — `Double`, поскольку мы собираемся использовать эти переменные в качестве аргументов методов AutoCAD.

Строки 7–12

```
'Рисование треугольника по двум сторонам и углу между ними
'Событие наступает при щелчке мыши на кнопке с именем Draw SAS Triangle
'Дополнительная возможность: ввод угла и длины второй стороны
'осуществляется с помощью текстовых полей.
'Вводимый угол – это угол между заданной и вводимой стороной
Private Sub CommandButton1_Click()
```

В данном комментарии указывается назначение процедуры `CommandButton1_Click()`, а также событие, при котором происходит ее вызов. Подобные комментарии рекомендуется делать для всех процедур, предназначенных для обработки событий.

Строка 13

```
UserForm3.Hide
```

Данная инструкция скрывает форму и переключает фокус в AutoCAD. Если бы этой инструкции не было в программе, то для возвращения в окно AutoCAD пользователю приходилось бы каждый раз закрывать форму вручную.

Строки 14–17

```
'p1 – это первая вершина, по умолчанию тип variant
'p2 – это вторая вершина, по умолчанию тип variant
p1 = ThisDrawing.Utility.GetPoint(,"Введите или выберите первую вершину:")
p2 = ThisDrawing.Utility.GetPoint(,"Введите или выберите вторую вершину:")
```

Здесь происходит ввод данных, описывающих сторону треугольника, с клавиатуры или при помощи указателя мыши.

Строки 19–23 и 33–35

```
If TextBox1.Text = "" Then
Angle = ThisDrawing.Utility.GetAngle(p2,"[Key ang. from+…]")
Else
Angle = ThisDrawing.Utility.AngleFromXAxis(p1,p2) + Included Angle
End If
Private Sub textBox1_Change()
IncludedAngle = PI - Val(TextBox1.Text)*PI / 180
End Sub
```

Конструкция `If … Then … Else` анализирует содержимое текстового поля TextBox1. Если в форму не вводилась величина угла, то применяется метод `GetAngle`. Если же величина угла была введена, то с помощью метода `textBox1_Change()` она переводится из градусной меры в радианную. Этот угол в сумме с углом, полученным в результате выполнения метода `AngleFromXAxis`, дает полярный угол третьей вершины треугольника.

Строки 24–26

```
If TextBox2.Text = "" Then
Dist = ThisDrawing.Utility.GetDistance(p2, "Введите расстояние от [base
point]")
End If
```

Если в форму была введена длина второй стороны, то производится ее преобразование из текстового формата в числовой.

Строки 18, 28 и 29

```
ThisDrawing.ModelSpace.AddLine p1, p2
ThisDrawing.ModelSpace.AddLine p2, p3
ThisDrawing.ModelSpace.AddLine p1, p3
```

Каждая из этих инструкций рисует соответствующую сторону треугольника.

Код модуля (Module 3)

В окне модуля необходимо ввести следующий текст:

```
Option Explicit   1
Sub SAS()   2
UserForm3.Show   3
End Sub   4
```

Упражнение 2

Напишите программу, которая будет рисовать треугольник с заданными вершинами и прилежащими к ней углами (рис. 34.14). Создайте форму, сходную с представленной на рис. 34.15.

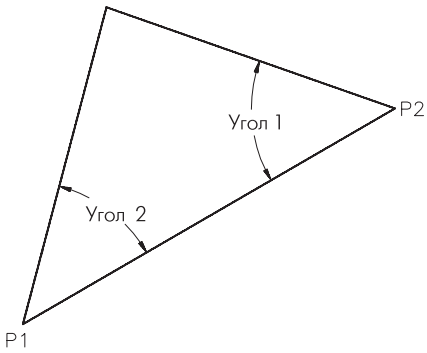


Рис. 34.14. Треугольник с заданной стороной и двумя прилежащими углами

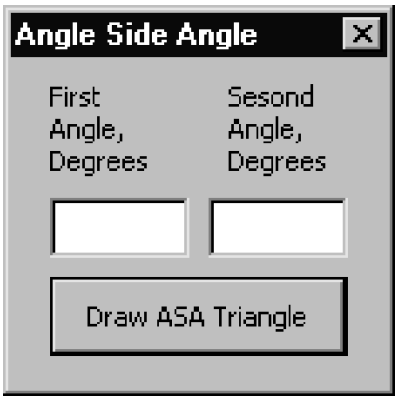


Рис. 34.15. Форма к упражнению 2

Упражнение 3

Напишите программу, которая будет рисовать треугольник с заданной стороной и длинами двух других сторон (рис. 34.16). Создайте форму, сходную с представленной на рис. 34.17.

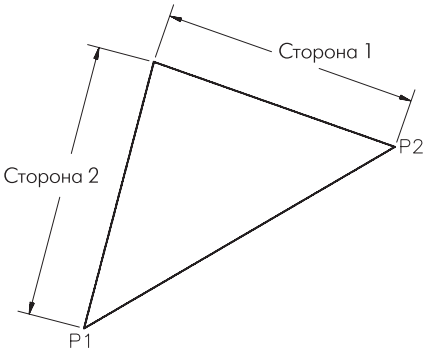


Рис. 34.16. Треугольник с заданной стороной и длинами двух других сторон

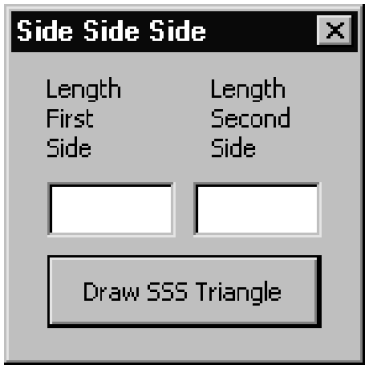


Рис. 34.17. Форма к упражнению 3

Дополнительные примеры приложений VBA

В справочной системе программы содержится богатый набор примеров приложений, написанных на языке Visual Basic. Файлы образцов находятся в папке Sample/VBA, находящейся в папке программы AutoCAD 2004, а также на компакт-диске. Еще одним богатым источником приложений и справочной информации по Visual Basic является Интернет.

Упражнения и задачи

Упражнение 4

Напишите программу, рисующую равносторонний треугольник внутри окружности (рис. 34.18).

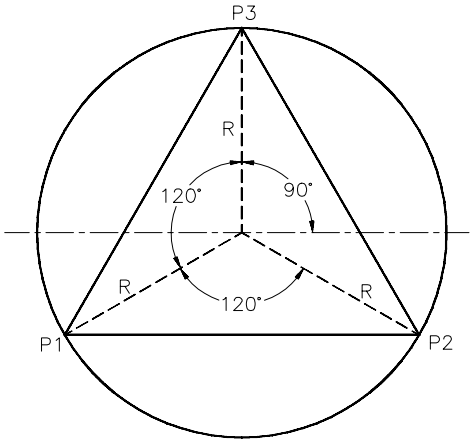


Рис. 34.18. Равносторонний треугольник внутри окружности

Упражнение 5

Напишите программу, рисующую квадрат со стороной S, в который вписана окружность (рис. 34.19). Положение квадрата задается углом, который одна из его сторон образует с осью абсцисс. Программа должна предусматривать ввод пользователем координат исходной точки P1, длины стороны квадрата S и величины угла ANG при помощи формы или графического интерфейса AutoCAD.

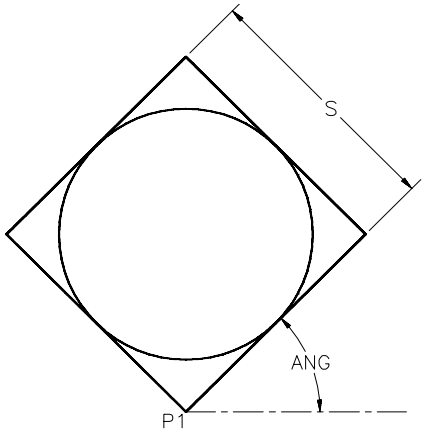


Рис. 34.19. Квадрат со стороной S, образующей угол ANG с осью абсцисс

Упражнение 6

Напишите программу, которая будет отображать на экране сначала линию,

образующую угол ANG с осью абсцисс, а затем семейство из числа N параллельных ей линий, расположенных на расстоянии S друг от друга (рис. 34.20). Программа должна предусматривать ввод всех необходимых ей данных.

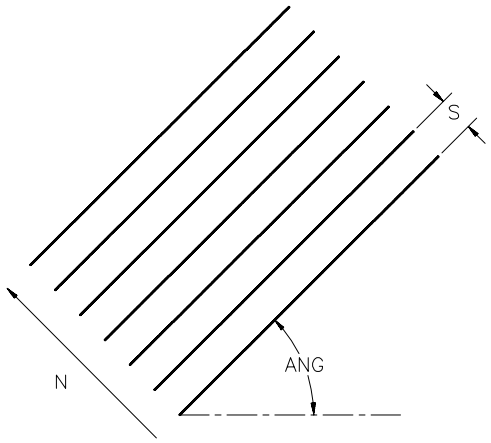


Рис. 34.20. N параллельных линий, расположенных на расстоянии S друг от друга

Упражнение 7

Напишите программу, которая будет рисовать отверстие с осевыми линиями (рис. 34.21). Программа должна предусматривать ввод длины и ширины отверстия, а также типа линии для изображения осевых линий с помощью формы или графического интерфейса AutoCAD.

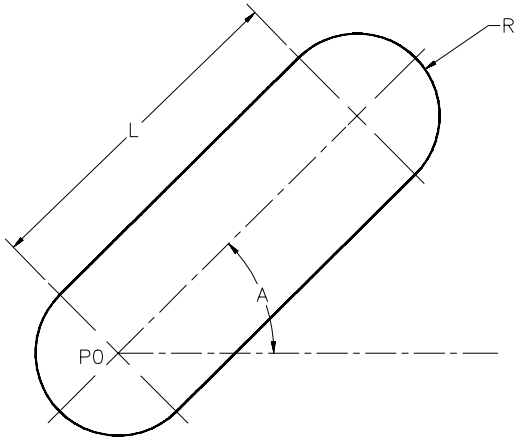


Рис. 34.21. Отверстие длиной L и радиусом R

Упражнение 8

Напишите программу, рисующую пару общих касательных к двум окружностям (рис. 34.22). Программа должна обеспечивать ввод значений диаметров окружностей и расстояния между их центрами с помощью формы или графического интерфейса AutoCAD.

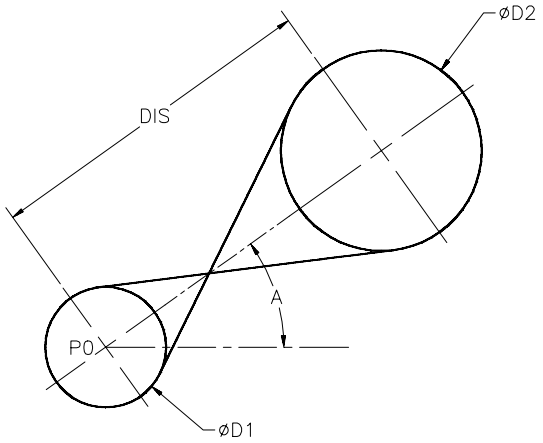
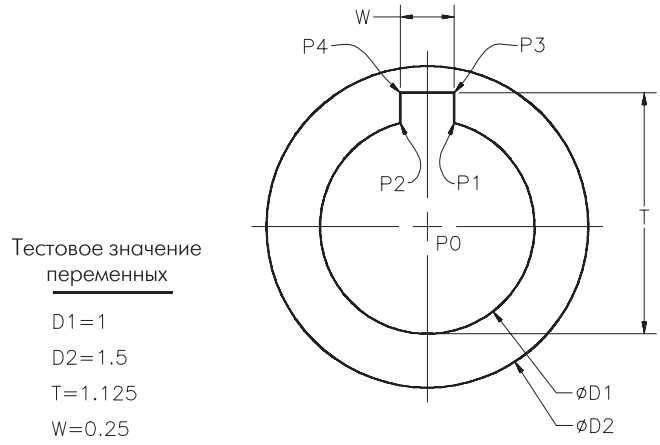


Рис. 34.22. Окружности с общими касательными

Упражнение 9

Напишите программу, которая будет рисовать втулку с отверстием для ключа (рис. 34.23). Программа должна обеспечивать ввод величин, указанных на рисунке, с помощью формы или графического интерфейса AutoCAD. Введите в программу значения величин, указанные на рисунке.



Тестовое значение переменных

D1=1

D2=1.5

T=1.125

W=0.25

Рис. 34.23. Втулка с отверстием для ключа

Упражнение 10

Напишите программу, которая будет рисовать бегунок, показанный на рис. 34.24.

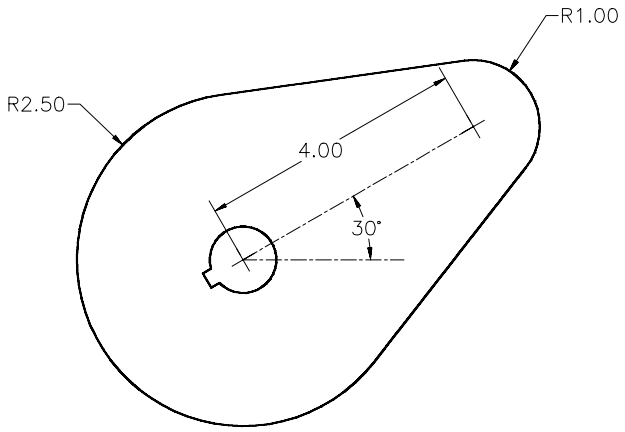


Рис. 34.24. Бегунок к упражнению 10

Упражнение 11

Напишите программу, которая будет рисовать бегунок, показанный на рис. 34.25.

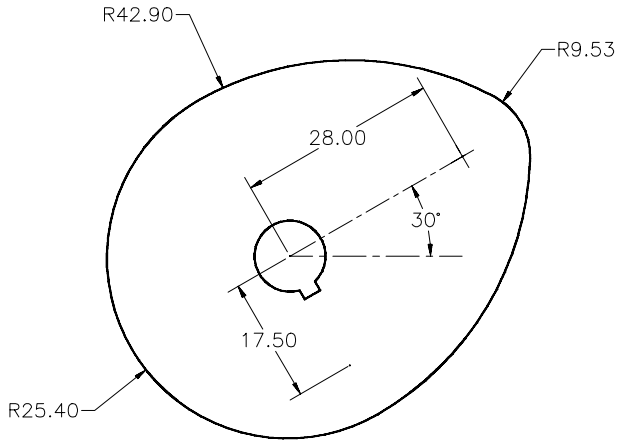


Рис. 34.25. Бегунок к упражнению 11

Задача 1

Создайте следующий параметрический рисунок в прямоугольнике шириной 320 мм и длиной 200 мм. Блок зависит от числа Q, называемого *счетчиком наборов данных*. Величины, показанные на рис. 34.26, задаются как X = Q + 160; Z = 240 – X; Y = 120 – Z; R = Z/4. Расстояние между видами должно быть равно 40 мм, а между видом и прямоугольником — 20 мм (для наиболее близко расположенных точек). Круг должен быть расположен в центре блока.

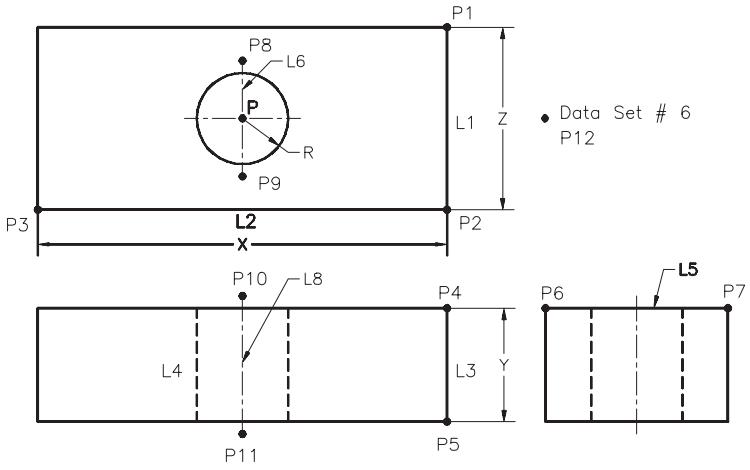


Рис. 34.26. Параметрический блок с отверстием