МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

С.П. МИНЕЕВ

ОСНОВЫ ПРОГРАММИРОВАНИЯ В AUTOCAD. ТЕХНОЛОГИИ ACTIVEX AUTOMATION И VBA В СРЕДЕ ПРОЕКТИРОВАНИЯ AUTOCAD ДЛЯ РЕШЕНИЯ ЗАДАЧ ЭЛЕКТРОМЕХАНИКИ

Учебное пособие

Самара
Самарский государственный технический университет

2015

УДК 004.432:621.3

Основы программирования в AutoCAD. Технологии Activex Aautomation и VBA в среде проектирования AutoCAD для решения задач электромеханики: Учеб.пособ./ *С.П.Минеев*; Самар. гос. техн. ун-т. Самара, 2015. 83 с.

Предназначено для изучения теоретической части и самостоятельной работы студента и соответствует учебной программе курса «Алгоритмизация, программирования и компьютерная графика в задачах электромеханики». Содержит примеры, задания на самостоятельную работу, вопросы для самоконтроля.

Задачи пособия – вовлечение каждого читателя в осознанную самостоятельную работу над текстом пособия, обеспечение непременного овладения базовыми знаниями визуального программирования.

Предназначено для студентов электротехнических специальностей учреждений высшего профессионального образования для глубокой и самостоятельной проработки и самоконтроля. ISBN 978-5-7964-0978-7

Ил. 18. Табл. 10. Библиогр.: 5 назв.

Печатается по решению редакционно-издательского совета
Самарского государственного технического университета

Рецензент канд. техн. наук *А.Н. Проценко*

ISBN 978-5-7964-0978-7

ПРЕДИСЛОВИЕ

Проектирование электрических машин требует глубоких профессиональных знаний не только в электромеханике, но и в других областях. Одна из таких областей – системы автоматизированного проектирования электрических машин (СПАР ЭМ). Уже сейчас САПР ЭМ занимают основное место в конструкторских и технологических бюро электротехнических заводов.

Каждый день на заводах электротехнической промышленности ведется работа по созданию новых и модернизации выпускаемых электрических машин. От интенсивности работ по созданию новой и модернизации старой продукции зависит экономическое положение заводов и НИИ, однако большинство инженеров-электромехаников ежедневно значительную часть времени тратят на стандартные расчеты и корректировку чертежей. Системы автоматизированного проектирования электрических машин (САПР ЭМ) призваны освободить инженеров от рутинной работы, обеспечив большую творческую отдачу.

Большинство задач проектирования электрических машин сочетает в себе необходимость выполнения как вычислений, так и процедур графического характера. Основные затраты времени конструктора идут не на выбор принципиального решения, четко вырисовываемого в его воображении, а на перенос мысленного образа на бумагу с соблюдением всех правил машиностроительного черчения.

В процессе создания САПР ЭМ разработаны математические модели чертежей активной части электрических машин. Исходными данными для выпуска чертежей активной части являются результаты электромагнитного расчета.

Программы фрагментов сборочных единиц и деталей создают с использованием интерактивной графической системы. Для программирования фрагмента необходимо задать положение базовой системы координат сборочной единицы или детали, а также описать элементы чертежа фрагмента. Чертеж фрагмента детали задается в базовой системе координат, т.е. относительно такой системы координат, которая определяет положение детали относительно других деталей при ее работе в электрической машине. При составлении сборочного чертежа сопрягаются системы координат деталей друг с другом. За оси координат принимают осевые линии отверстий и валов, оси симметрии и т.п. Например, одна ось координат подшипникового щита проходит вдоль оси вращения машины, а другая – по поверхности замка и служит измерительной и сборочной базой подшипни-кового щита.

Для описания чертежа фрагмента используют чертежные примитивы – прямые линии, точки, дуги. Кроме того, при программировании фрагментов широко применяют команды аффинного преобразования, позволяющие перемещать какой-либо фрагмент, поворачивать его на некоторый угол, изображать его в увеличенном или уменьшенном масштабе, строить новый элемент, симметричный данному.

Учебное пособие предназначено для студентов, обучающихся по специальности «Электромеханика», однако может быть также полезно для студентов других специальностей, а также для инженеров-электромехаников в практической работе.

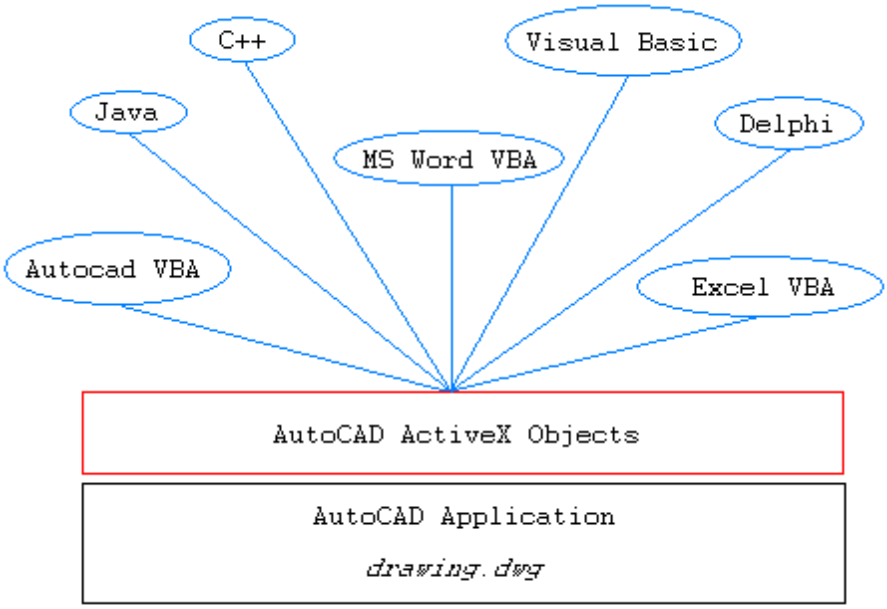
Автор с благодарностью примет все замечания и пожелания читателей и просит направлять их по адресу: г. Самара, ул. Первомайская, д.18, ауд.134.

ВВЕДЕНИЕ

ActiveX Automation – это технология Microsoft, которая позволяет обеспечить доступ к объектам одного программного продукта для использования этих объектов из другого программного продукта. VBA – это язык программирования, базирующийся на стандартном

Visual BASIC, который Microsoft внедряет в приложения как своего производства, так и других компаний (AutoCAD программа, созданная Autodesk). Используя VBA, можно создавать программы, которые управляют теми частями приложения, которые предоставлены через ActiveX Automation.

ActiveX Automation позволяет создавать приложения на любом языке, который поддерживает интерфейс ActiveX Automation. На рис. 1 показаны приложения и языки программирования, которые имеют доступ к AutoCAD через ActiveX Automation. Все приложения и языки программирования кроме VBA имеют собственные отдельные среды проектирования, которые не интегрированы в AutoCAD. VBA встроен в среду AutoCAD, что позволяет отказаться от приобретения дополнительных программных средств.



Р и с. 1. Приложения с поддержкой ActiveX Automation

Среда проектирования AutoCAD позволяет создавать чертежную документацию с помощью различных инструментов. VBA позволяет автоматизировать создание данных чертежей, получая доступ к этим инструментам через ActiveX Automation.

Средства автоматизации в AutoCAD существовали и ранее. До появления VBA в среде AutoCAD существовал и до сих пор суще-

ствует язык программирования AutoLISP, но программирование на AutoLISP не всегда просто. Команды достаточно сложны, встроенная отладка ограничена, и немногие программисты знакомы с AutoLISP.

С помощью ActiveX Automation программирование в AutoCAD доступно для специалистов в Visual BASIC и VBA, а так как BASIC был разработан как язык для начинающих, а Visual BASIC продолжает традицию легкого освоения, то, изучив несколько простых программных концепций, можно легко создавать свои собственные приложения для AutoCAD.

VBA хорошо интегрирован с Windows. VBA имеет полный доступ к файловой системе Windows, что позволяет создавать и удалять файлы и просматривать каталоги. Также доступен полный интерфейс Windows-программирования, так что программа может быть настолько сложна, насколько это возможно.

Так как учебное пособие посвящено программированию, то здесь встречаются примеры текстов программ, которые выглядят следующим образом:

```
Sub Add()
  'новый документ чертежа
  Dim Listcherteja As AcadDocument
  'Создается новый документ
  Set Listcherteja = Documents.Add
End Sub
```

Кроме того, следует обратить внимание на текст, выделенный **полужирным шрифтом**. Таким способом отмечены понятия, на которые нужно обратить особое внимание: определения, объекты, свойства, методы.

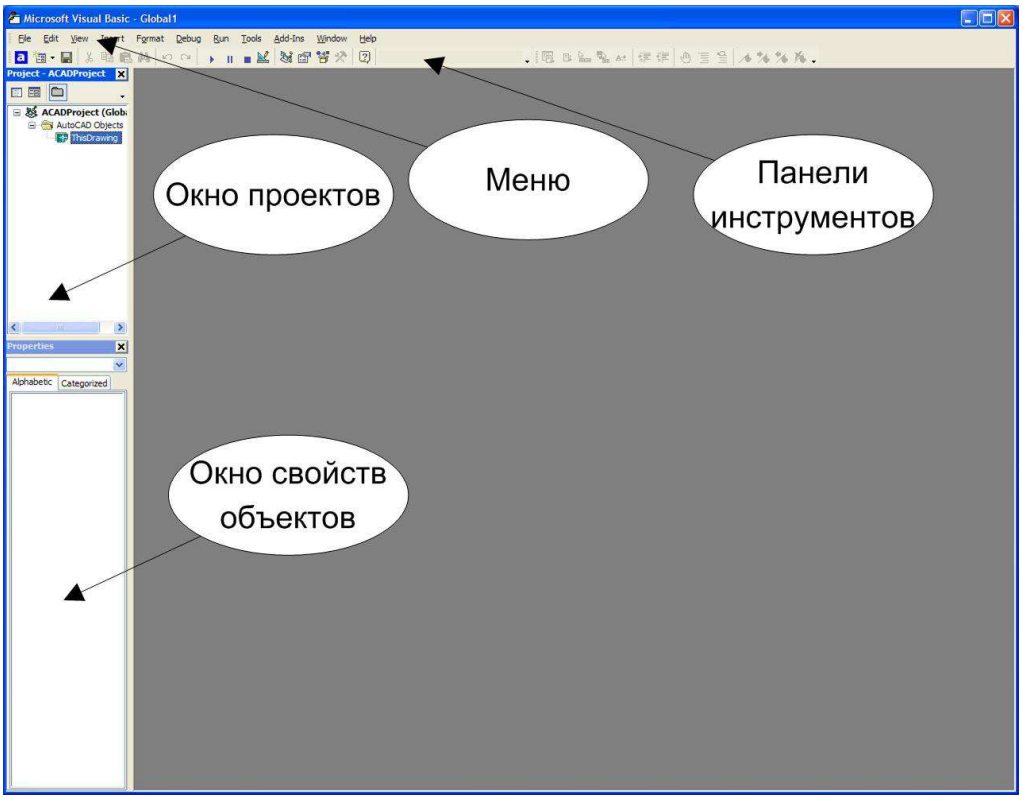
ИНТЕРФЕЙС СРЕДЫ VBA

Изучение VBA начнем со среды программирования, в которой придется работать. Из Autocad среду программирования можно вызвать несколькими способами:

нажав на клавиатуре комбинацию клавиш [Alt+F11];

через меню – Сервис – Макросы – Редактор Visual Basic.

После вызова редактора с помощью вышеперечисленных способов на экране появится **окно среды программирования** (р и с. 2).



Р и с. 2. Среда программирования VBA

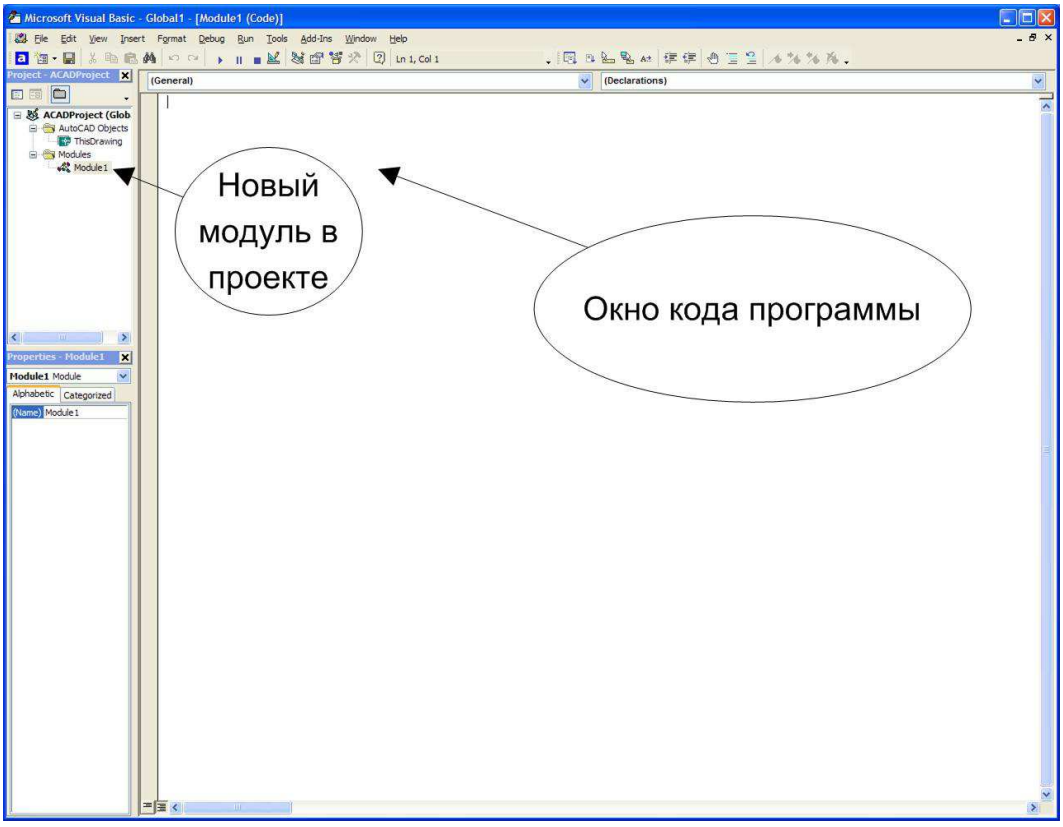
Среда программирования при первом запуске состоит из следующих частей: **меню**, **панелей инструментов**, **окна проектов**, **окна свойств**. С помощью меню и панелей инструментов осуществляется управление средой программирования. Окно проектов (Project – ACADProject) содержит в себе открытые в данный момент проекты. Каждый проект может содержать **формы** и **модули** с **кодом**. С помощью **окна свойств** можно изменять свойства **объектов** формы во время проектирования пользовательского интерфейса – среды общения между пользователем и программой.

Модуль (Module) – это контейнер для **кода** программ, который может содержать любое количество процедур и функций. Модуль может содержать **процедуры** и **функции.**

Пользовательская форма (UserForms) – это интерфейс, который выглядит как обычное окно других приложений и предназначенное для организации диалога между пользователем и программой. Этот диалог выглядит со стороны пользователя, как нажатие на кнопку с помощью мыши, выбор элемента из выпадающего списка, ввод данных в текстовое поле, включение-выключение флажков и использование других элементов управления. Программа общается с пользователем с помощью этих же элементов управления.

Код – это текст программы, который пишет программист, размещая в модулях и в коде форм. Модули и формы отображаются в окне проектов.

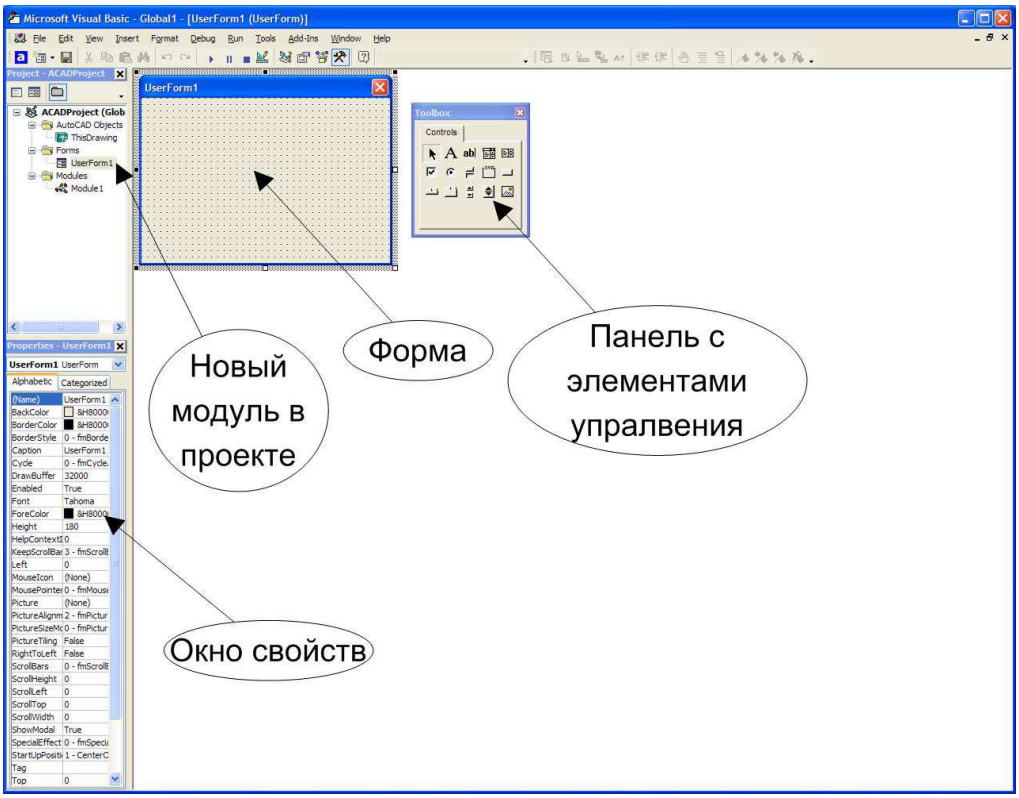
В новом проекте формы и модули отсутствуют. Новый модуль в проект можно вставить с помощью меню Insert – Module, а новую форму – Insert – UserForm. Если в проект вставить новый модуль, то среда программирования будет выглядеть, как на р и с. 3.



Р и с. 3. Среда программирования с новым модулем

Если модуль активный, то в среде программирования появляется **окно кода программ**. В этом окне программистом пишется текст программы. Код программы должен находиться в контейнерах модуля – **процедурах**, либо **функциях**.

Если в проект вставить новую форму, то среда программирования примет вид, как на р и с. 4.



Р и с. 4. Среда программирования с новой формой

В среде программирования появляется контейнер – **пользовательская форма** (или просто **форма**). Выше говорилось о предназначении формы. Вместе с формой также появляется **панель с элементами управления**. Сама по себе форма не способна взаимодействовать с пользователем. Взаимодействие происходит с помощью элементов управления (табл.1). Данные элементы помещаются на форму с помощью перетаскивания из панели с элементами управления.

Форма является **объектом**. Элементы управления также являются объектами, но являются **объектами формы**, т.е. принадлежат форме (находятся на форме).

Кроме того, существуют объекты AutoCAD. К ним, например, относятся такие графические примитивы, как точка, линия, дуга, окружность и др. Именно для работы с такими объектами и был встроен VBA в AutoCAD.

Объект – элемент, который обладает **свойствами**, **методами** и **событиями** (функциями, которые отличают один объект от другого). Поскольку при программировании идет работа с объектами формы, объектами AutoCAD, то само программирование называется **объектно-ориентированным**. **Объекты**, **свойства**, **методы** и **события** являются фундаментальными понятиями в объектно-ориентированном программировании. Разговор о перечисленных понятиях пойдет ниже.

Таблица 1

Элементы управления формы

Название элемента	Обозначение на панели с элементами управления	Префикс	Пример свойства Name
Form (Форма)		frm	frmSample
Label (Надпись)		lbl	lblPick
TextBox (Текстовое окно)		txt	txtEdit
ComboBox (Поле со списком)		cbo	cboChoose
ListBox (Список)		lst	lstSpisok
Check Box (Флажок)		chk	chkTop
Option Button (Кнопка выбора)		opt	optBlue
ToggleButton (Выключатель)		tog	togVkluch
Frame (Рамка, группа)		fra	fraStud
CommandButton (Кнопка)		cmd	cmdBeep

ScrollBar (Полоса прокрутки)		scr	scrPolosa
SpinButton (Счетчик)		spn	spnSchet
Image (Рисунок)		img	imgPicture

После внедрения новой формы в проект изменяется **окно свойств** (см. р и с. 4). Данное окно заполнятся **свойствами** формы. В этом же окне отображаются свойства элементов управления, помещенных на форму, если сделать элемент управления активным. Одновременно могут отображаться свойства как одного элемента управления, так и нескольких. Чтобы отображались свойства нескольких элементов, они должны быть выделены с помощью мыши. При этом отображаются только общие свойства элементов.

Свойство – это характеристика объекта (имя, цвет, размер, расположение в пространстве, надписи и др.).

В окне свойств отображаются только свойства формы и объектов формы (элементов управления формы). Свойства объектов AutoCAD в этом окне не отображаются.

СОЗДАНИЕ НОВОГО ЛИСТА ЧЕРТЕЖА

Команда добавления нового чертежа в коллекцию чертежей

RetVal = object.Add,

где **RetVal** – название создаваемого чертежа; **Object** – коллекция **Documents**. С помощью метода **Add** к данной коллекции добавляется новый объект (новый лист чертежа).

Пример. Вставьте в проект VBA новый модуль. Поместите в модуль следующую процедуру:

```
Sub Add()

  'новый документ чертежа
  Dim Listcherteja As AcadDocument
  'Создается новый документ
```

```
    Set Listcherteja = Documents.Add

End Sub
```

СОЗДАНИЕ С ПОМОЩЬЮ VBA ГРАФИЧЕСКИХ ПРИМИТИВОВ

В среде AutoCAD графические примитивы создаются с помощью панели инструментов «Рисование» (р и с. 5).

Р и с. 5. Панель инструментов «Рисование»

В среде VBA графические примитивы создаются с помощью **методов**. Для каждого из примитивов существует свой отдельный метод.

Метод – это действие, совершаемое объектом. Метод является основополагающим понятием в объектно-ориентированном программировании.

СОЗДАНИЕ ГРАФИЧЕСКОГО ПРИМИТИВА «ОТРЕЗОК»

Линии могут состоять из одного сегмента или из нескольких соединенных сегментов, но каждый сегмент состоит из отельных объектов **Отрезок** (**Line**). Данный объект используют, когда необходимо редактировать каждый сегмент индивидуально. Если необходимо нарисовать серию сегментов как один целый объект, то нужно использовать объект **Полилинию** (**LightweightPolyline**)

Чтобы создать отрезок, используется метод **AddLine**. Чтобы редактировать или узнать информацию об отрезке, используют методы и свойства, указанные в табл. 2

Таблица 2

Методы, свойства и события объекта Отрезок (Line)

Методы	Свойства	События

ArrayPolar	Angle	Modified
ArrayRectangular	Application	
Copy	Document	
Delete	Delta	
GetBoundingBox	EndPoint	
GetExtensionDictionary	Handle	
GetXData	HasExtensionDictionary	
Highlight	Hyperlinks	

Окончание табл. 2

Методы	Свойства	События
IntersectWith	Layer	
Mirror	Length	
Mirror3D	Linetype	
Move	LinetypeScale	
Offset	Lineweight	
Rotate	Normal	
Rotate3D	ObjectID	
ScaleEntity	OwnerID	
SetXData	PlotStyleName	
TransformBy	StartPoint	
Update	Thickness	
	TrueColor	
	Visible	

Команда построения отрезка

RetVal = object.AddLine(StartPoint, EndPoint),

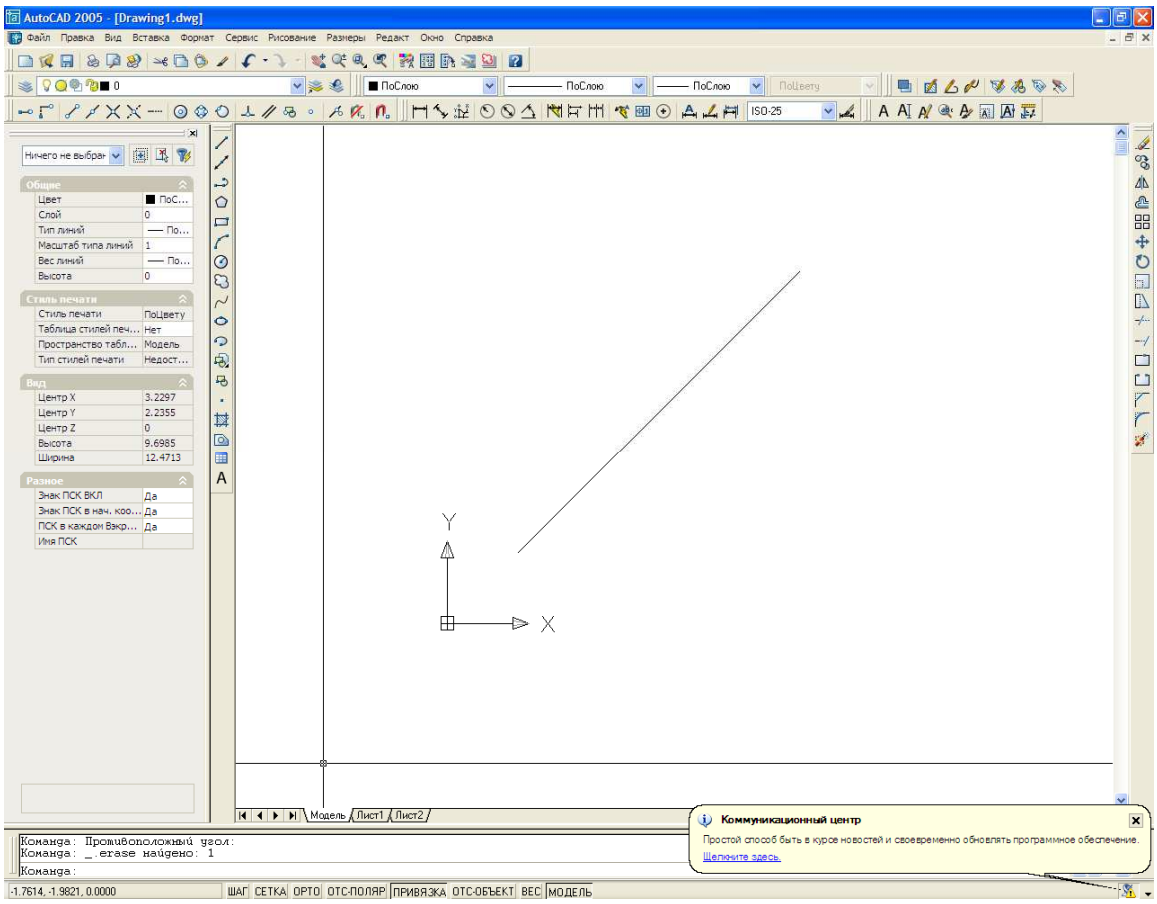
где **RetVal** – название создаваемого объекта **«отрезок»**; **Object** – объект или объекты, к которым применяется метод **AddLine.** В качестве объектов могут быть использованы **ModelSpace Collection, Paper-**

Space Collection, Block; **StartPoint** – координаты первой точки отрезка в трехмерном пространстве. Представляет собой одномерный массив, состоящий из трех элементов, имеет тип **Double**; **EndPoint** – координаты второй точки отрезка в трехмерном пространстве. Представляет собой одномерный массив, состоящий из трех элементов, имеет тип **Double**.

Пример. Вставьте в проект VBA новый модуль. Поместите в модуль следующую процедуру:

```
    Sub Example_AddLine()
    'Данный пример добавляет отрезок в модельное про-
странство чертежа

    Dim lineObj As AcadLine
    Dim startPoint(0 To 2) As Double
    Dim endPoint(0 To 2) As Double

    'Определяем начальную и конечную точки отрезка
    startPoint(0) = 1#: startPoint(1) = 1#: startPoint(2) = 0#
    endPoint(0) = 5#: endPoint(1) = 5#: endPoint(2) = 0#
    ' Помещаем отрезок в модельное пространство
    Set lineObj = ThisDraw-
ing.ModelSpace.AddLine(startPoint, endPoint)
    ZoomAll

    End Sub
```

Запустив данную программу с помощью кнопки  , располагающейся на панели инструментов, получим в чертеже отрезок (р и с. 6).



Р и с. 6. Отрезок, построенный с помощью программы

Рассмотрим процедуру создания отрезка подробнее.

`Sub Example_AddLine()` – данная команда открывает процедуру. Ключевое слово **Sub** говорит о том, что это начало процедуры, **Example_AddLine()** – название процедуры. В скобках указываются параметры процедуры. В нашем случае процедура не содержит параметров.

`Dim lineObj As AcadLine` – команда объявления объекта «отрезок».

Dim – инструкция, предназначенная для объявления переменных, массивов, объектов, ссылок на объекты.

As – ключевое слово, после которого указывается тип объявляемых переменных, массивов, объектов. В нашем случае типом объявляемого объекта является отрезок (**AcadLine)**.

```
Dim startPoint(0 To 2) As Double
Dim endPoint(0 To 2) As Double
```

В этих двух строках объявляются начальная и конечная точки отрезка. Каждая из этих точек является массивом, состоящим из трех элементов – координат по оси x, y, z. Тип массива – **Double**, дробное число с плавающей точкой двойной точности.

```
    startPoint(0) = 1#: startPoint(1) = 1#:
startPoint(2) = 0#
    endPoint(0) = 5#: endPoint(1) = 5#: endPoint(2) = 0#
```

В этих двух строках определяются координаты начальной и конечной точек отрезка. За координату X отвечает элемент массива с индексом (0), за координату Y– элемент массива с индексом (1), за координату Z – элемент массива с индексом (2).

Знаком «=» элементу массива присваивается числовое значение координаты.

```
    Set lineObj = ThisDraw-
ing.ModelSpace.AddLine(startPoint, endPoint)
```

В этой строке создается объект отрезок.

Set – инструкция, позволяющая установить ссылку на создаваемый объект. За инструкцией следует название создаваемого объекта (ссылки на объект). В нашем примере ссылкой на объект является **lineObj**.

Знаком «=» ссылке на объект **lineObj** присваивается создаваемый объект. В нашем случае создаваемый объект – отрезок. В дальнейшем для управления созданным отрезком (изменение свойств) можно обращаться через ссылку **lineObj**.

ThisDrawing – объект AutoCAD, представляет собой текущий чертеж.

ModelSpace – объект AutoCAD, представляет собой модельное пространство чертежа, включающее в себя все созданные объекты. В нашем примере изначально модельное пространство не содержало в себе объектов.

AddLine – метод, добавляющий к объектам модельного пространства новый отрезок.

(startPoint, endPoint) – координаты начальной и конечной точек. В нашем примере определялись ранее.

Рассмотрим подробнее строку **ThisDrawing.ModelSpace.AddLine**

Точкой разделяются друг от друга объекты, а также свойства и методы. Первым указывается объект-контейнер, который может содержать другие объекты. Следующим указывается объект, находящийся внутри объекта-контейнера. При этом между ними ставится точка. В нашем примере объект **ThisDrawing** содержит в себе объект **ModelSpace.**

Объект, находящийся внутри объекта-контейнера, может сам являться контейнером, т.е. содержать в себе другие объекты. Эта вложенность объектов напоминает собой матрешку, в которой находятся другие матрешки, вложенные друг в друга.

Последним указывается метод либо свойство. От объекта они разделяются также точкой. При этом свойства либо метод относятся к объекту, от которого они отделены точкой, т.е. к самому вложенному объекту. В нашем примере метод **AddLine** относится к объекту **ModelSpace.**

`ZoomAll` – команда, показывающая весь чертеж целиком.

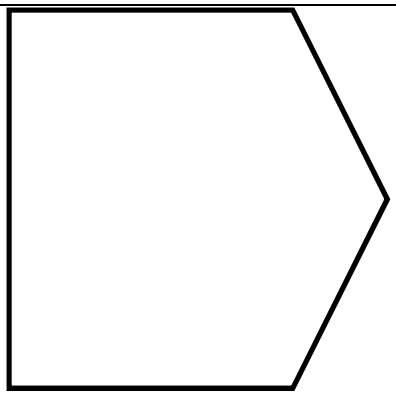
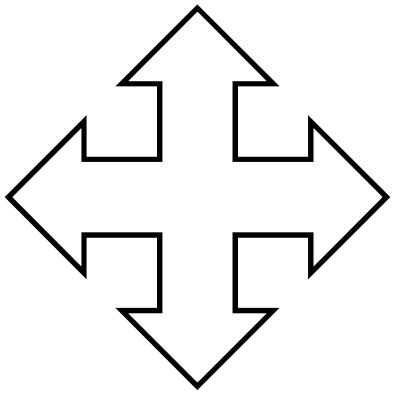
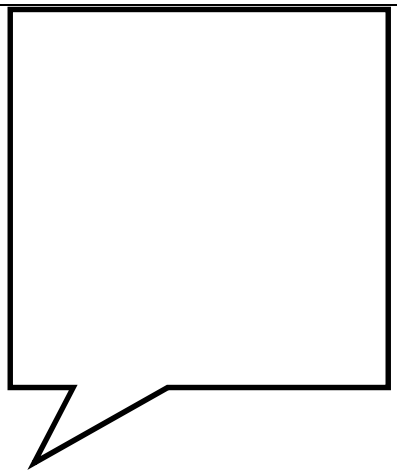
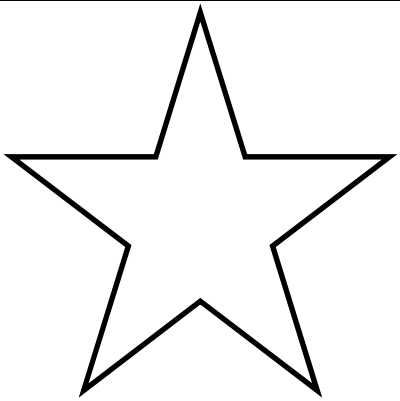
`End Sub` – команда, завершающая процедуру **Sub Example_AddLine()**

Самостоятельная работа №1
СОЗДАНИЕ ОТРЕЗКА С ПОМОЩЬЮ МЕТОДА ADDLINE

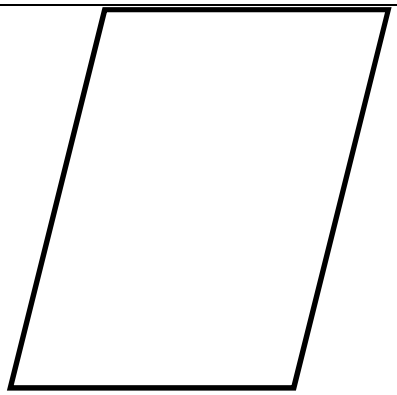
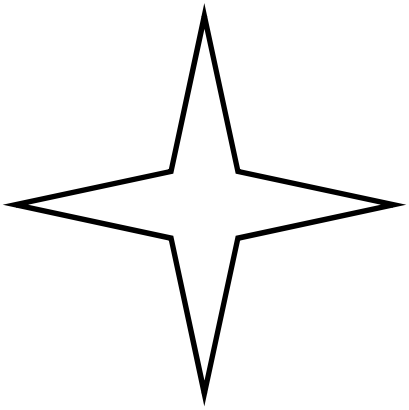
Создайте программы для создания чертежа двух фигур. Координаты фигур выберите самостоятельно. При создании чертежей используйте метод AddLine. Варианты заданий даны в табл.1.1.

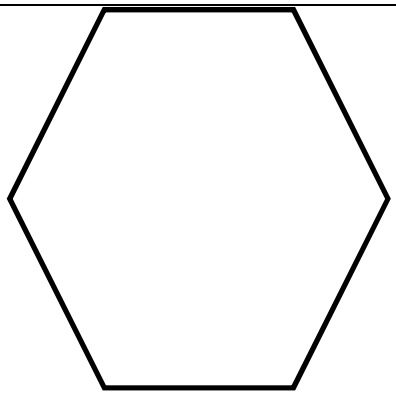
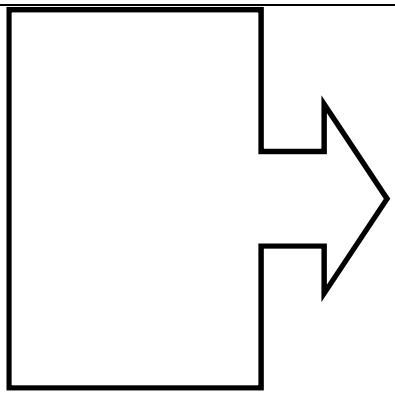
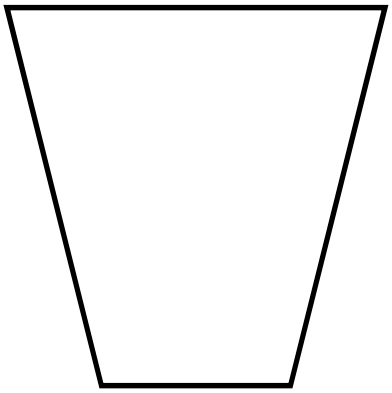
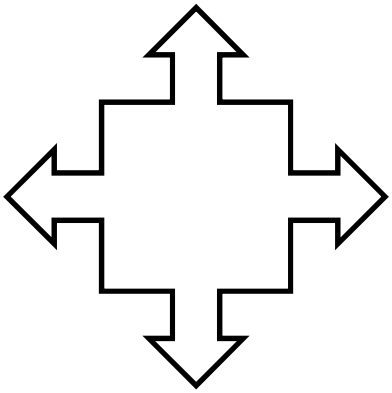
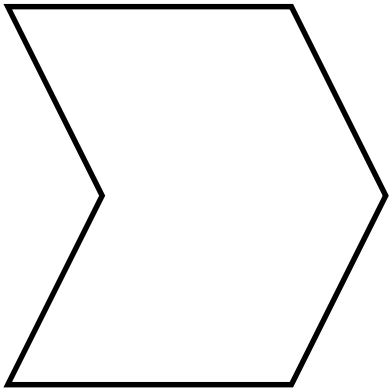
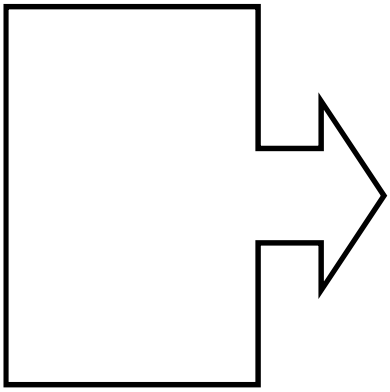
Таблица 1.1

Варианты заданий

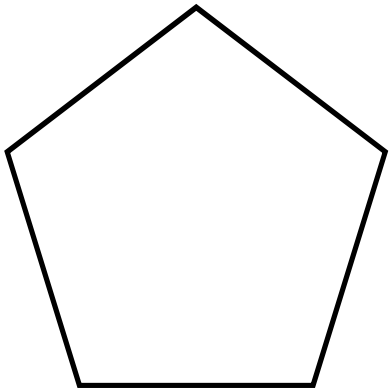
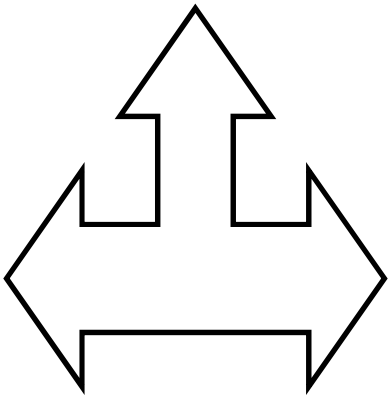
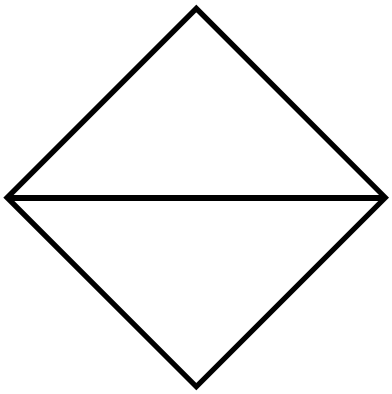
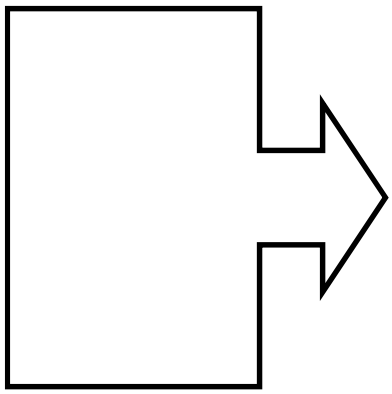
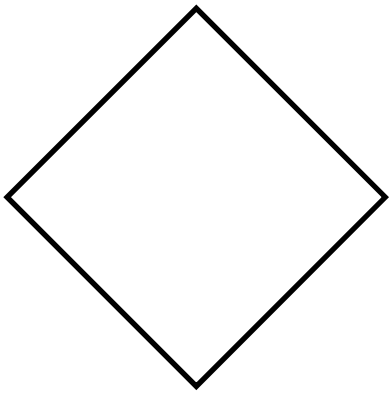
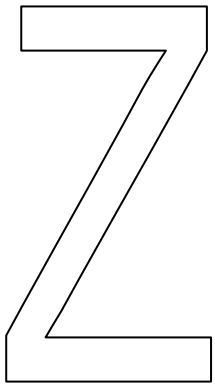
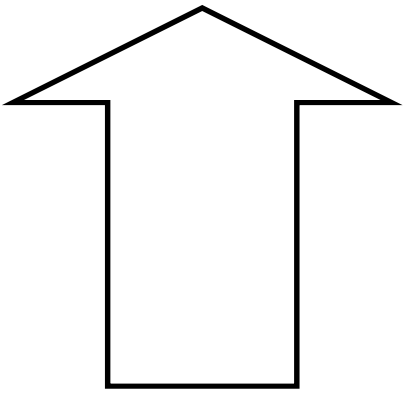
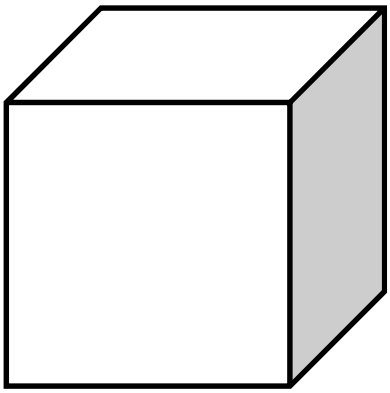
Номер варианта	Задание №1	Задание №2
1		
2		

Продолжение табл. 1.1

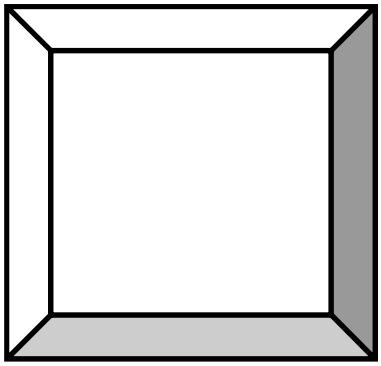
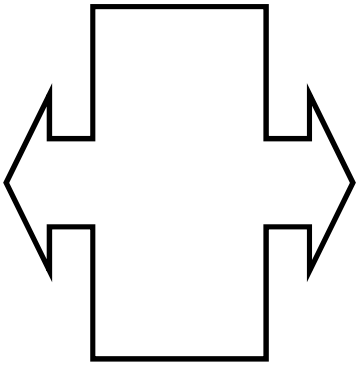
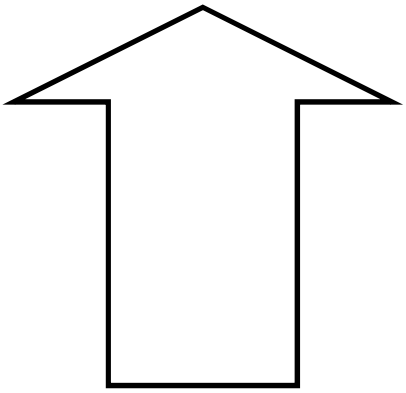
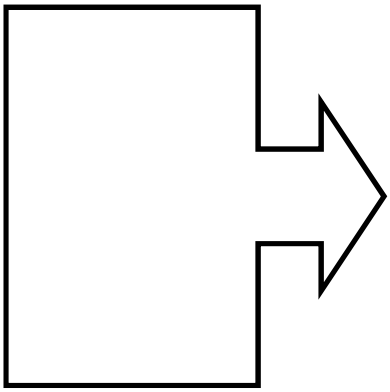
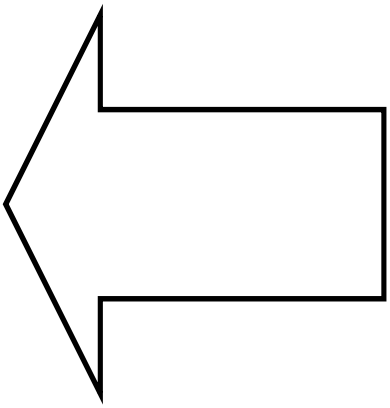
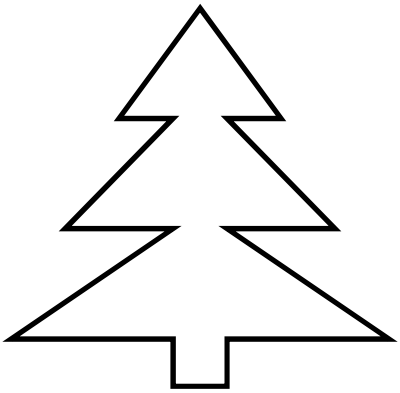
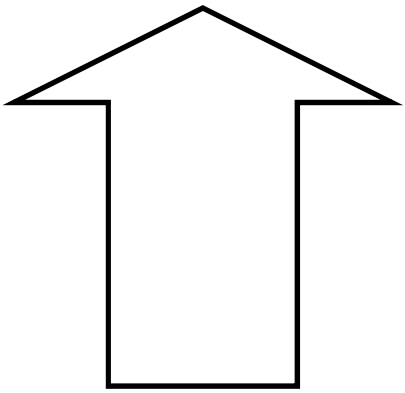
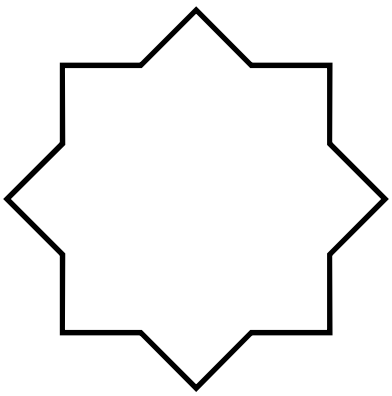
3		

4		
5		
6		

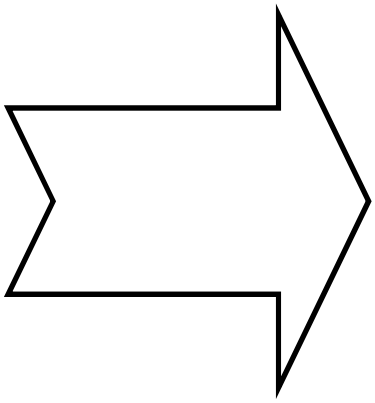
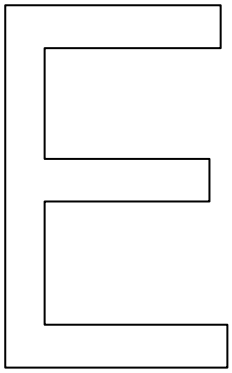
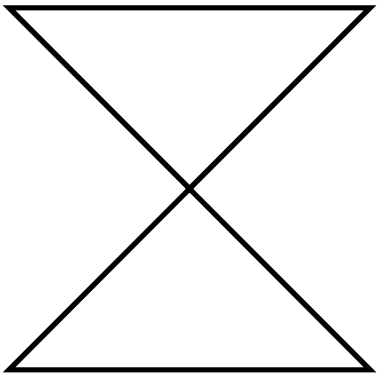
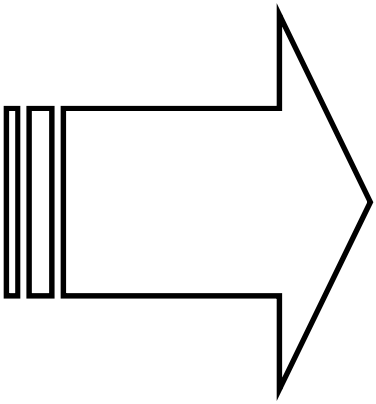
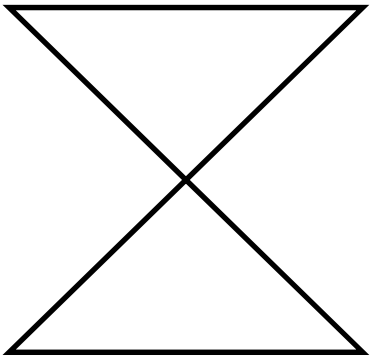
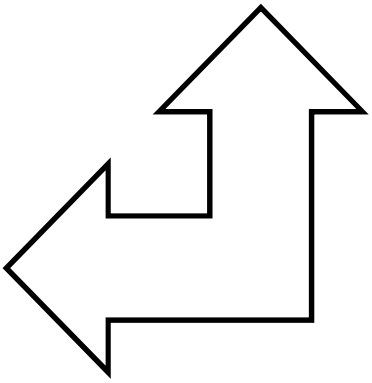
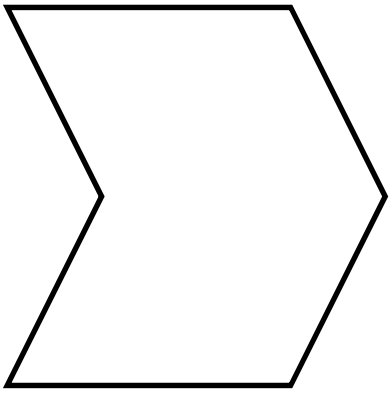
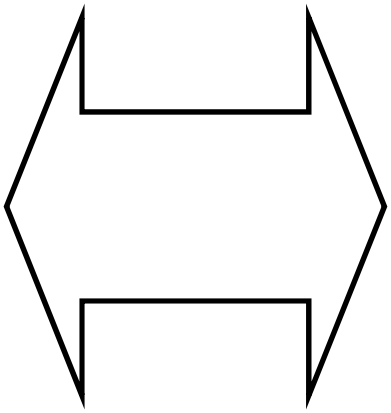
Продолжение табл. 1.1

7		
8		
9		
10		

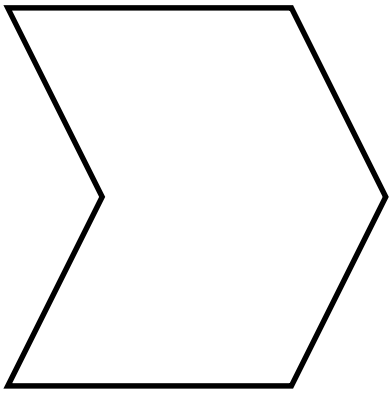
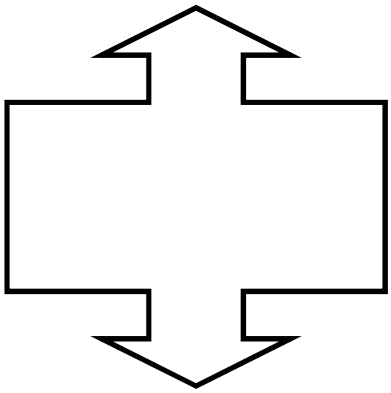
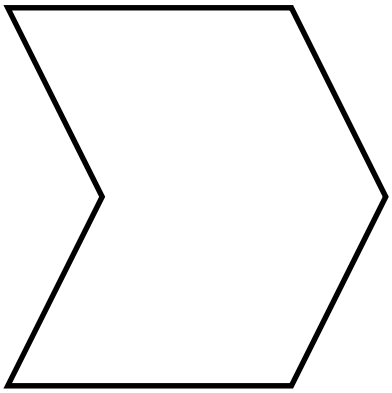
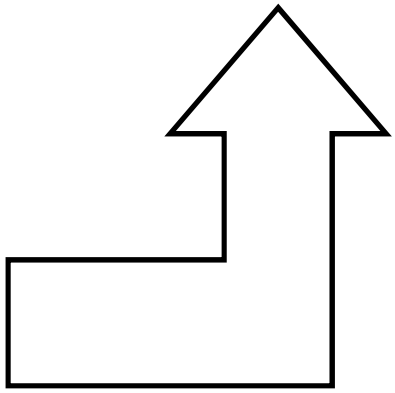
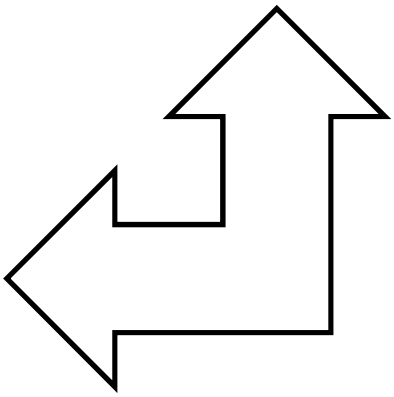
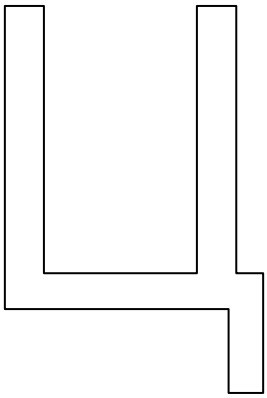
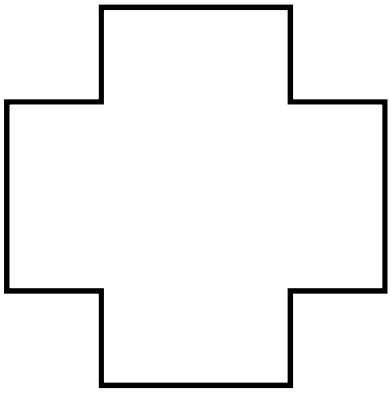
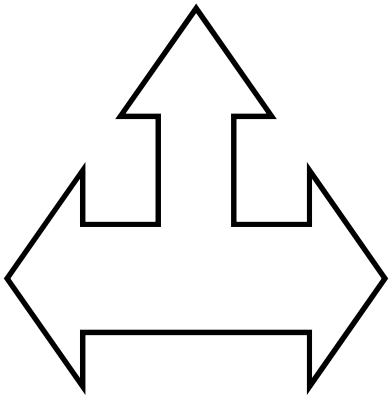
Продолжение табл. 1.1

11		
12		
13		
14		

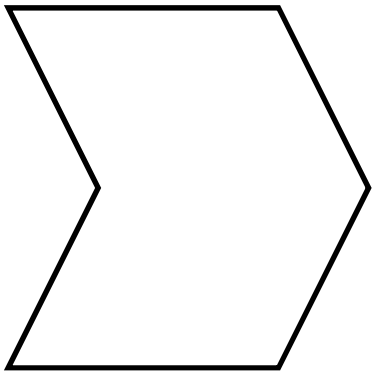
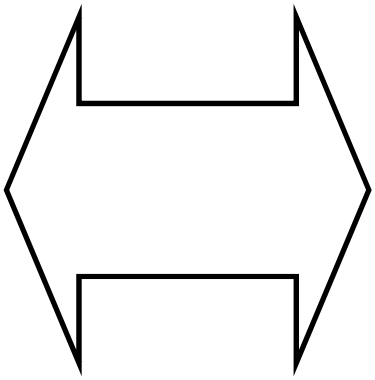
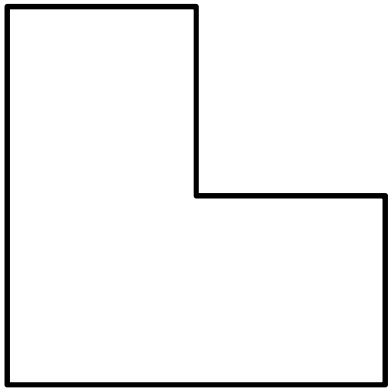
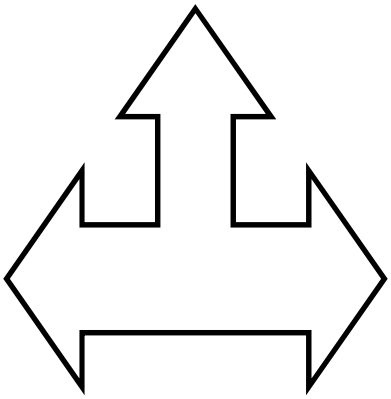
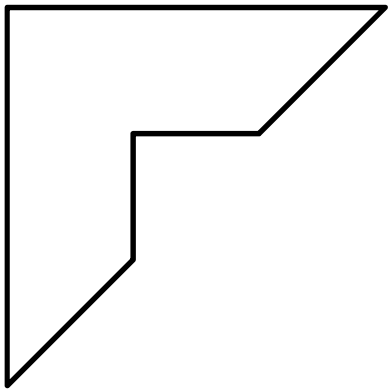
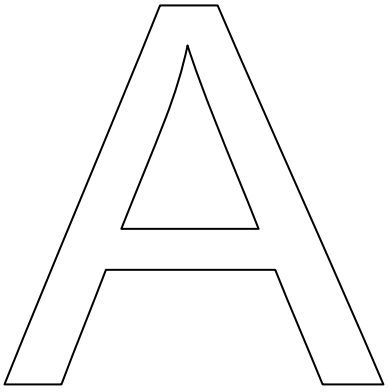
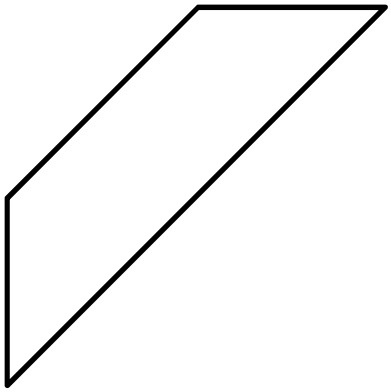
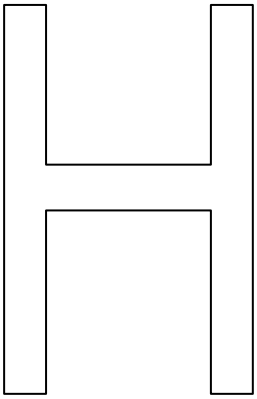
Продолжение табл. 1.1

15		
16		
17		
18		

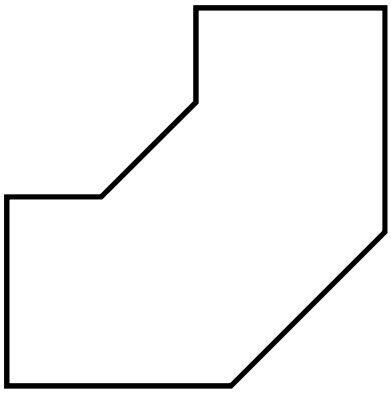
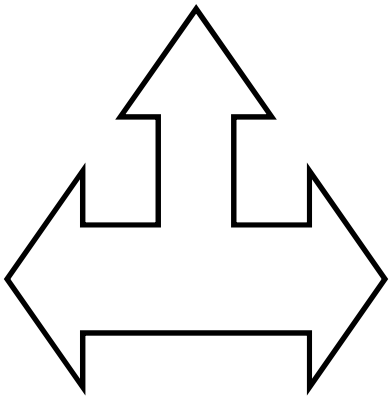
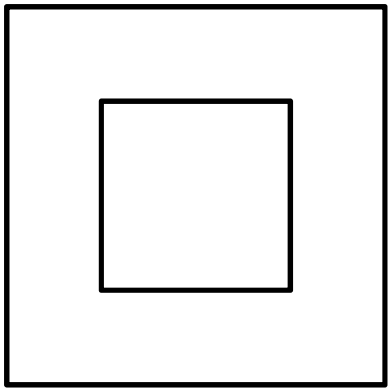
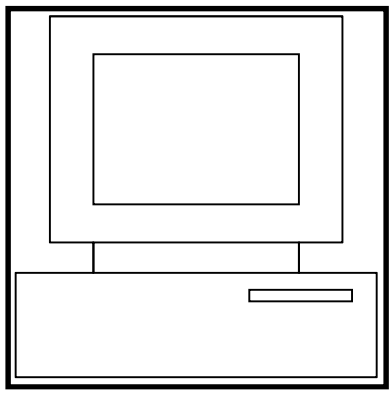
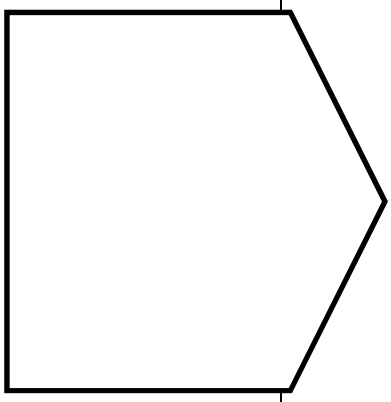
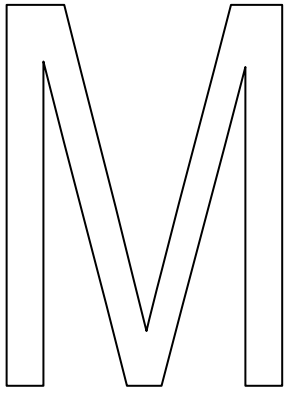
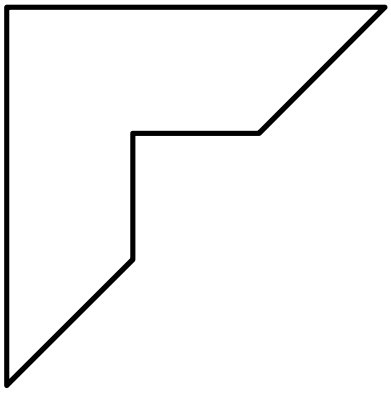
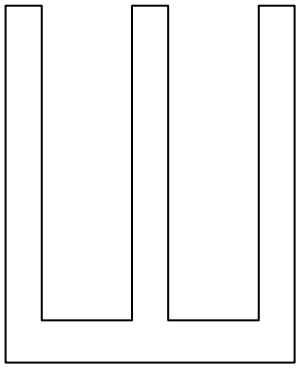
Продолжение табл. 1.1

19		
20		
21		
22		

Продолжение табл. 1.1

23		
24		
25		
26		

Окончание табл. 1.1

27		
28		
29		
30		

СОЗДАНИЕ ГРАФИЧЕСКОГО ПРИМИТИВА «ОКРУЖНОСТЬ»

Чтобы создать окружность используется метод **AddCircle** (рис. 1.1).

Команда построения окружности

```
RetVal = object.AddCircle(Center, Radius),
```

где **RetVal** – название создаваемого объекта **«окружность»**; **Object** – объект или объекты, к которым применяется метод **AddCircle.** В качестве объектов могут быть использованы **ModelSpace Collection, PaperSpace Collection, Block**; **Center** – координаты центра окружности в трехмерном пространстве. Представляет собой одномерный массив, состоящий из трех элементов, имеет тип **Double**; **Radius –** радиус окружности, имеет тип **Double**. Должен быть положительным числом.

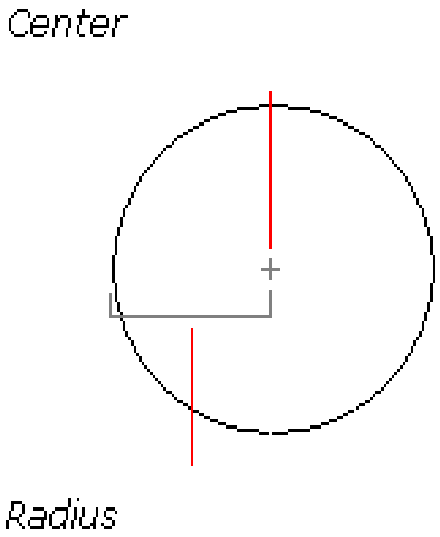


Рис. 1.1. Пояснительный рисунок
к применению метода AddCircle

Чтобы создать окружность, используется метод **AddCircle**. Чтобы редактировать или узнать информацию об окружности, используют методы и свойства, указанные в табл.1.2.

Таблица 1.2

Методы, свойства и события объекта Окружность (Circle)

Методы	Свойства	События
ArrayPolar	Application	Modified
ArrayRectangular	Area	
Copy	Center	
Delete	Circumference	
GetBoundingBox	Diameter	
GetExtensionDictionary	Document	
GetXData	Handle	
Highlight	HasExtensionDictionary	
IntersectWith	Hyperlinks	
Mirror	Layer	
Mirror3D	Linetype	
Move	LinetypeScale	
Offset	Lineweight	
Rotate	Normal	
Rotate3D	ObjectID	
ScaleEntity	OwnerID	
SetXData	PlotStyleName	
TransformBy	Radius	
Update	Thickness	
	TrueColor	
	Visible	

Пример. Вставьте в модуль проекта VBA следующую процедуру:

```
Sub Example_AddCircle()

  ' Данный пример добавляет круг в модельное простран-
ство чертежа
Dim circleObj As AcadCircle
```

```
    Dim centerPoint(0 To 2) As Double
    Dim radius As Double

    ' Определяем параметры круга
    centerPoint(0) = 0#: centerPoint(1) = 0#:
centerPoint(2) = 0#
    radius = 5#

    ' Помещаем круг в модельное пространство
    Set circleObj = ThisDraw-
ing.ModelSpace.AddCircle(centerPoint, radius)
    ZoomAll

    End Sub
```

Запустив данную программу с помощью кнопки , располагающейся на панели инструментов, получим на чертеже круг (рис. 1.2).

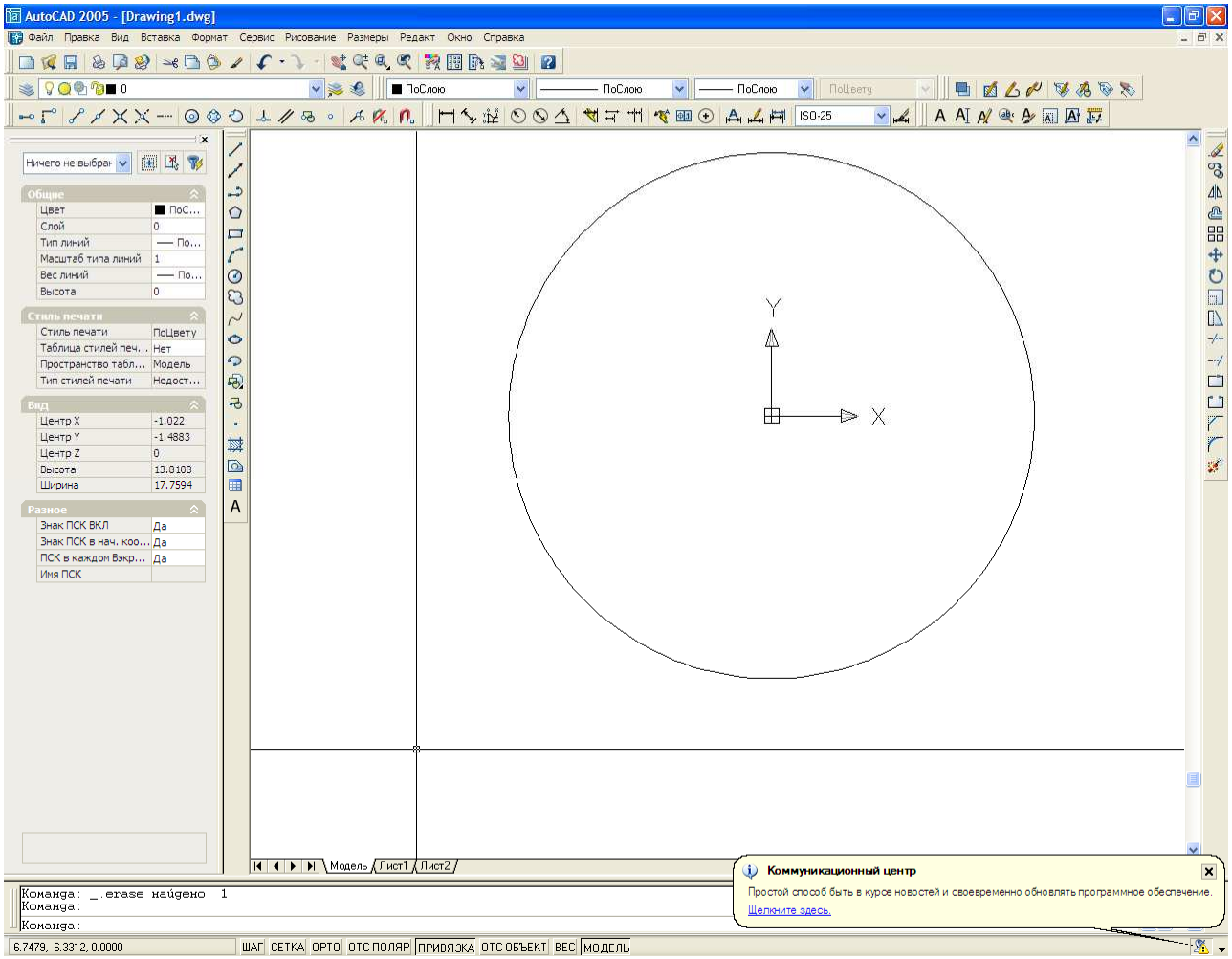


Рис. 1.2. Окружность, построенная с помощью программы

Рассмотрим некоторые строки программы создания объекта «круг» подробнее.

`Dim circleObj As AcadCircle` – команда объявления объекта «Круг», где **AcadCircle** – тип объявляемого объекта.

`Dim centerPoint(0 To 2) As Double` – в этой строке объявляется массив, представляющий собой координаты центра круга в трехмерном пространстве. Режим только ввод (input-only).

`Dim radius As Double` – в этой строке объявляется величина радиуса круга. Режим только ввод (input-only).

```
Set circleObj = ThisDraw-
ing.ModelSpace.AddCircle(centerPoint, radius)
```

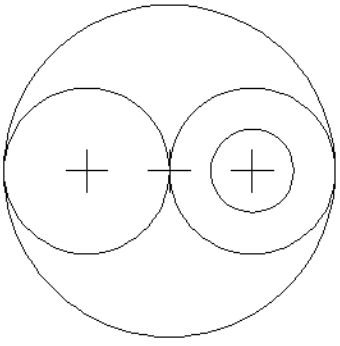
С помощью этой строки в чертеже создается объект «круг», где **AddCircle** – метод, добавляющий в чертеж круг. В скобках указаны параметры круга – координаты центра и радиус.

Самостоятельная работа №2
СОЗДАНИЕ КРУГА С ПОМОЩЬЮ МЕТОДА ADDCIRCLE

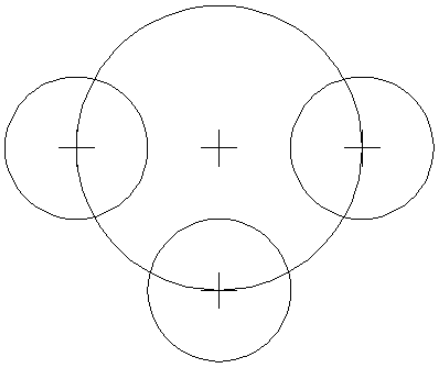
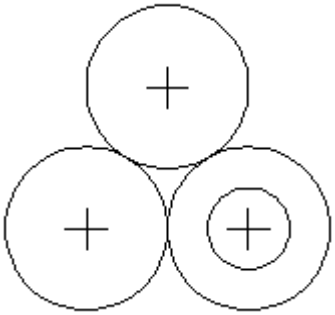
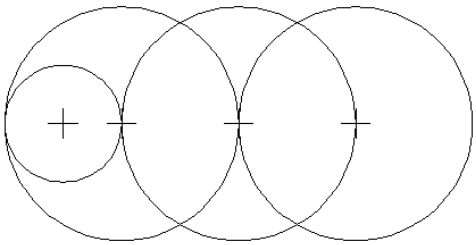
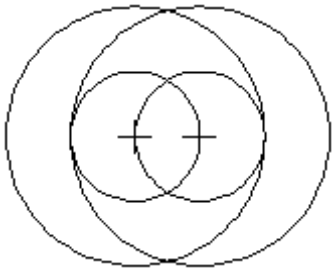
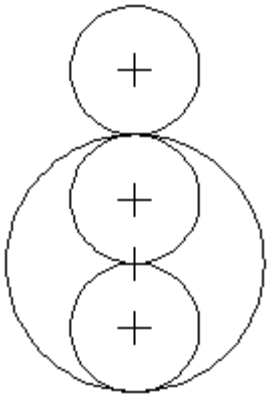
Создайте программу для создания чертежа фигур. Координаты центра кругов и их радиусы выберите самостоятельно. При создании чертежей используйте метод AddCircle. Крестиками обозначены центры кругов. Варианты заданий даны в табл. 2.1.

Таблица 2.1

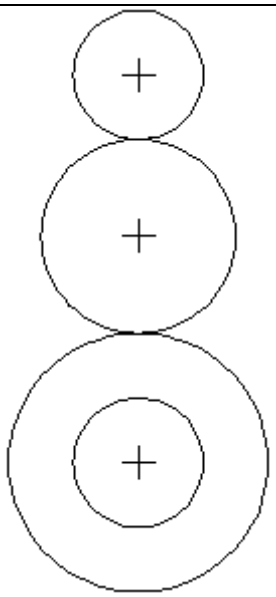
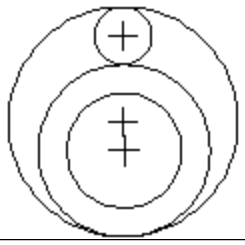
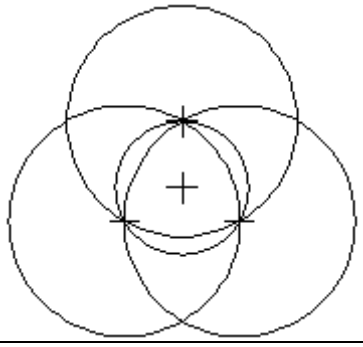
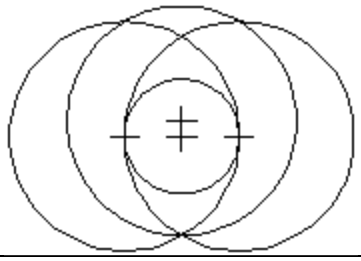
Варианты заданий

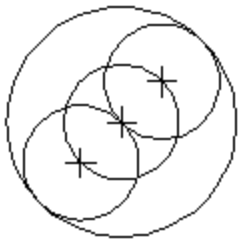
Номер варианта	Задание	Примечание
1		Две окружности имеют одинаковые радиусы

Продолжение табл. 2.1

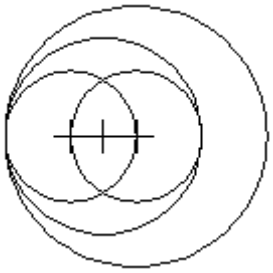
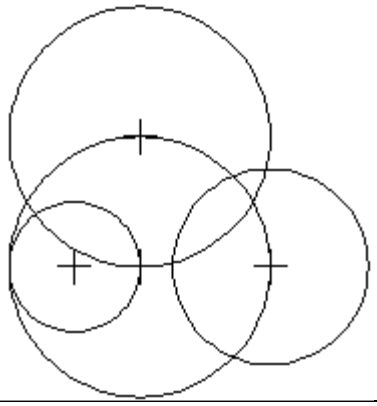
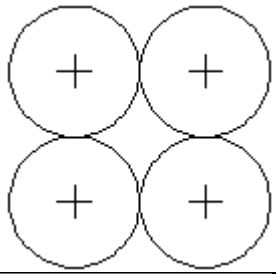
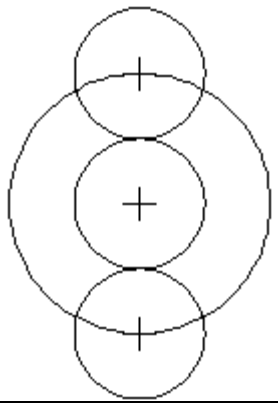
Номер варианта	Задание	Примечание
2		Три окружности имеют одинаковые радиусы, и их центры лежат на большей окружности
3		Три окружности имеют одинаковые радиусы
4		Три окружности имеют одинаковые радиусы
5		По две пары окружностей имеют одинаковые радиусы
6		Три окружности имеют одинаковые радиусы

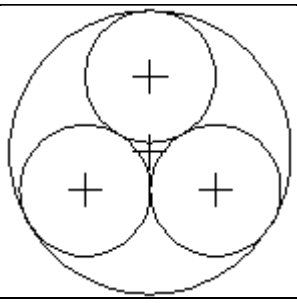
Продолжение табл. 2.1

Номер варианта	Задание	Примечание
7		
8		
9		Три окружности имеют одинаковые радиусы
10		Три окружности имеют одинаковые радиусы

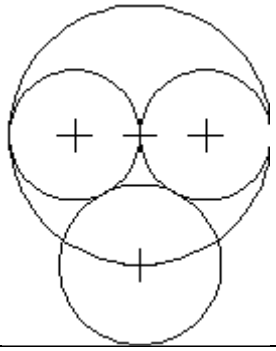
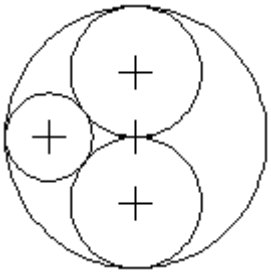
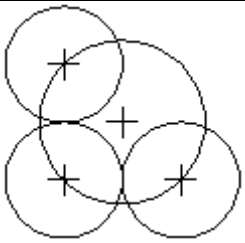
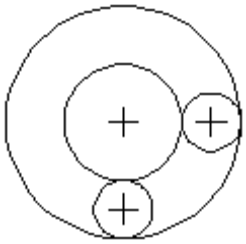
11		Три окружности имеют одинаковые радиусы

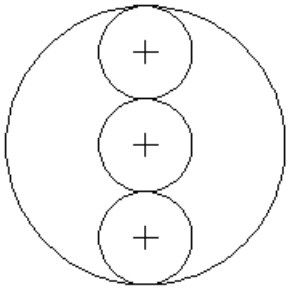
Продолжение табл. 2.1

Номер варианта	Задание	Примечание
12		Две окружности имеют одинаковые радиусы
13		Две окружности имеют одинаковые радиусы
14		Все окружности имеют одинаковые радиусы
15		Три окружности имеют одинаковые радиусы

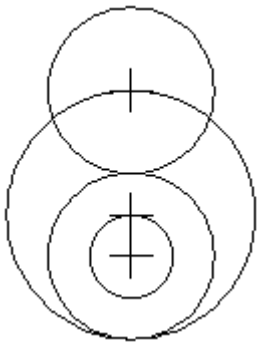
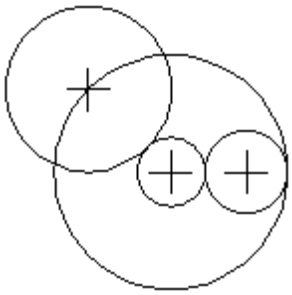
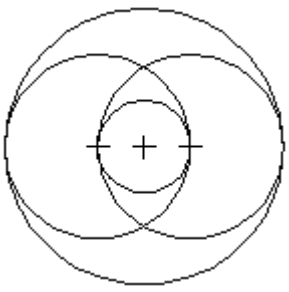
16		Три окружности имеют одинаковые радиусы

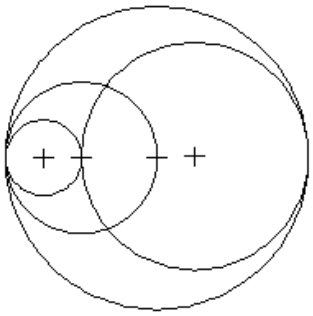
Продолжение табл. 2.1

Номер варианта	Задание	Примечание
17		Две окружности имеют одинаковые радиусы
18		Две окружности имеют одинаковые радиусы
19		Три окружности имеют одинаковые радиусы
20		Две окружности имеют одинаковые радиусы

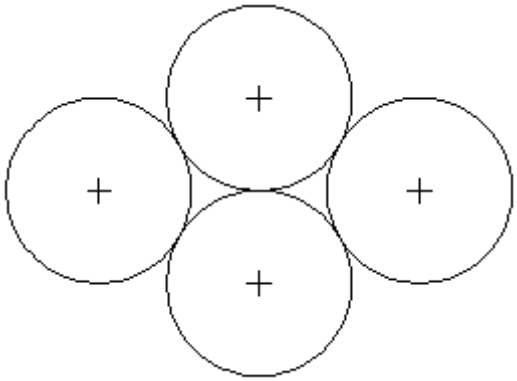
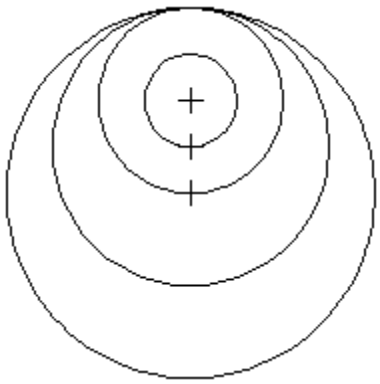
21		Три окружности имеют одинаковые радиусы

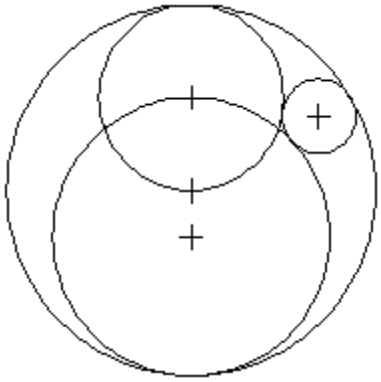
Продолжение табл. 2.1

Номер варианта	Задание	Примечание
22		Две окружности имеют одинаковые радиусы
23		
24		Две окружности имеют одинаковые радиусы

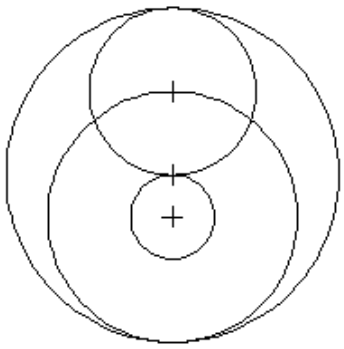
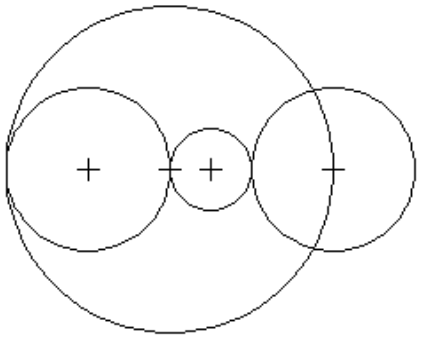
25		

Продолжение табл. 2.1

Номер варианта	Задание	Примечание
26		Все окружности имеют одинаковые радиусы
27		

28		Самая маленькая окружность касается всех трех окружностей

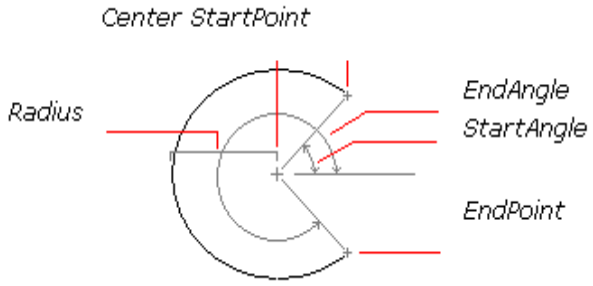
Окончание табл. 2.1

Номер варианта	Задание	Примечание
29		
30		Две окружности имеют одинаковые радиусы

СОЗДАНИЕ ГРАФИЧЕСКОГО ПРИМИТИВА «ДУГА»

Дуга всегда строится против часовой стрелка от начальной до конечной точки. Свойства StartНачальная точка и конечная точка дуги рассчитываются через свойства **StartAngle**, **EndAngle** и **Radius**.

Чтобы создать дугу, используется метод **AddArc** (рис. 2.1). Чтобы редактировать или узнать информацию о дуге, используют методы и свойства, указанные в табл. 2.2.



Р и с. 2.1. Пояснительный рисунок
к применению метода AddArc

Таблица 2.2

Методы, свойства и события объекта Дуга (Arc)

Методы	Свойства	События
ArrayPolar	Application	Modified
ArrayRectangular	ArcLength	
Copy	Area	
Delete	Center	
GetBoundingBox	Document	
GetExtensionDictionary	EndAngle	
GetXData	EndPoint	
Highlight	Handle	
IntersectWith	HasExtensionDictionary	
Mirror	Hyperlinks	
Mirror3D	Layer	
Move	Linetype	
Offset	LinetypeScale	

Rotate	Lineweight	
Rotate3D	Normal	
ScaleEntity	ObjectID	
SetXData	ObjectName	
TransformBy	OwnerID	
Update	PlotStyleName	
	Radius	
	StartAngle	
	StartPoint	
	Thickness	
	TotalAngle	
	TrueColor	
	Visible	

Команда построения дуги

RetVal = object.AddArc(Center, Radius, StartAngle, EndAngle) ,

где **RetVal** – название создаваемого объекта **«дуга»; Object** – объект или объекты, к которым применяется метод **AddArc.** В качестве объектов могут быть использованы **ModelSpace Collection, PaperSpace Collection, Block**; **Center** – координаты центра дуги в трехмерном пространстве. Представляет собой одномерный массив, состоящий из трех элементов, имеет тип **Double**; **Radius –** радиус дуги, имеет тип **Double**; **StartAngle, EndAngle** – начальные и конечные углы дуги, определяемые в радианах. Начальный угол должен быть меньше конечного угла. Направление построения дуги против часовой стрелки.

Пример. Вставьте в проект VBA новый модуль. Поместите в модуль следующую процедуру:

```
Sub Example_AddArc()
    'Данный пример создает дугу в пространстве модели

    Dim arcObj As AcadArc
```

```
        Dim centerPoint(0 To 2) As Double
        Dim radius As Double
        Dim startAngleInDegree As Double
        Dim endAngleInDegree As Double

        'Определяем окружность
        centerPoint(0) = 0#: centerPoint(1) = 0#: center-
Point(2) = 0#
        radius = 5#
        startAngleInDegree = 10#
        endAngleInDegree = 230#

        'Конвертируем углы в градусах в углы в радианах
        Dim startAngleInRadian As Double
        Dim endAngleInRadian As Double
        startAngleInRadian = startAngleInDegree *
3.141592 / 180#
        endAngleInRadian = endAngleInDegree * 3.141592 /
180#

        'Создаем дугу в пространстве модели.
        Set arcObj = ThisDraw-
ing.ModelSpace.AddArc(centerPoint, radius, startAn-
gleInRadian, endAngleInRadian)
        ZoomAll
    End Sub
```

Запустив данную программу с помощью кнопки  , располагающейся на панели инструментов, получим на чертеже дугу (рис. 2.2).

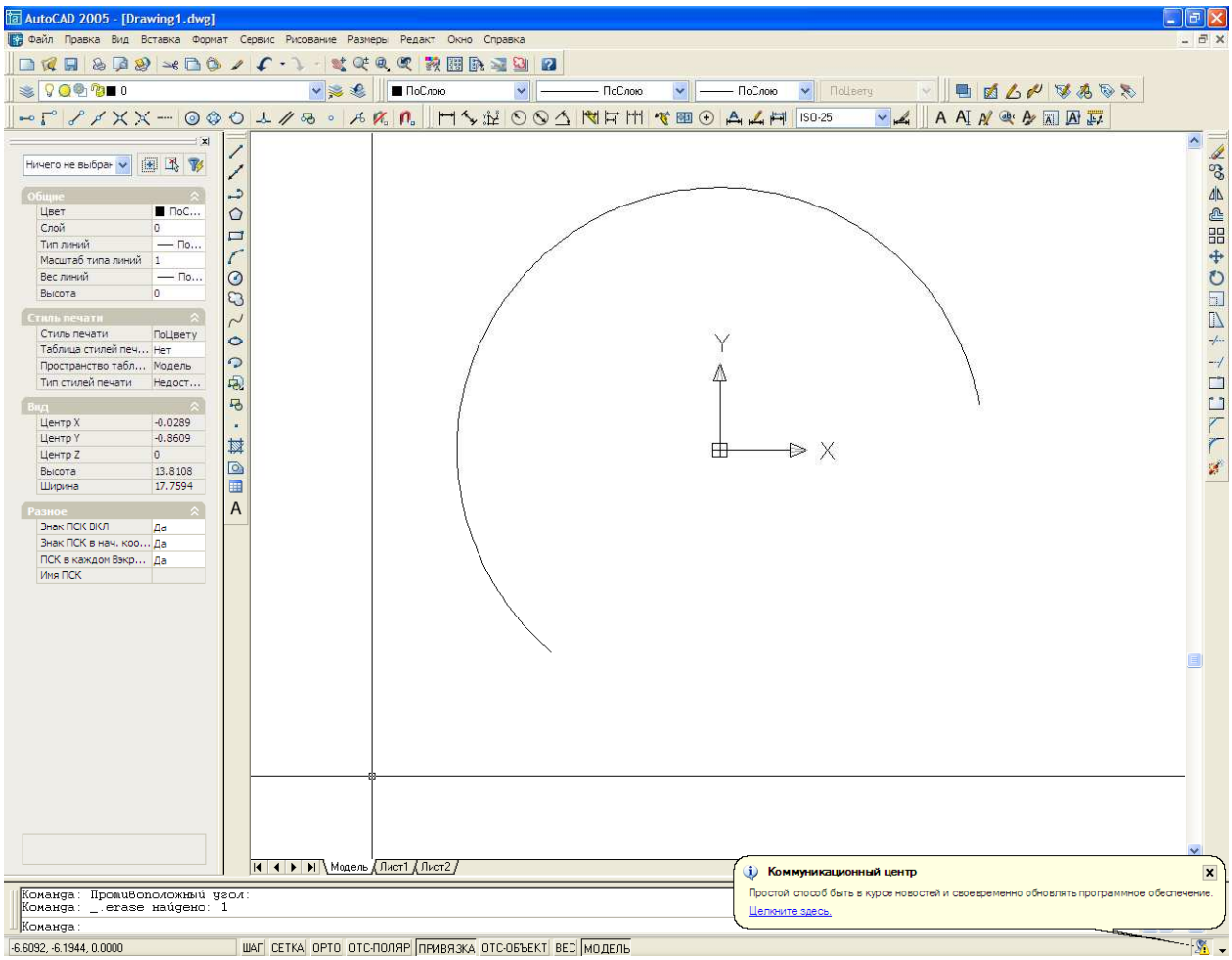


Рис. 2.2. Дуга, построенная с помощью программы

Рассмотрим некоторые строки программы создания объекта «Дуга» подробнее.

`Dim arcObj As AcadArc` – команда объявления объекта «Дуга», где **AcadArc** – тип объявляемого объекта.

`Dim centerPoint(0 To 2) As Double` – в этой строке объявляется массив, представляющий собой координаты центра дуги в трехмерном пространстве.

`Dim radius As Double` – в этой строке объявляется величина радиуса дуги.

`Dim startAngleInDegree As Double` – в этой строке объявляется начало дуги – угол смещения относительно оси X против часовой стрелки. Начальная точка дуги будет располо-

жена на расстоянии **radius** от центра и смещена на угол **startAngleInDegree**

`Dim endAngleInDegree As Double` – в этой строке объявляется конец дуги – угол смещения относительно оси X против часовой стрелки. Конечная точка дуги будет расположена на расстоянии **radius** от центра и смещена на угол **startAngleInDegree**

`centerPoint(0) = 0#: centerPoint(1) = 0#: centerPoint(2) = 0#` – определяем центр дуги.

`radius = 5#` – определяем радиус.

`startAngleInDegree = 10#` – определяем начальный угол дуги в градусах.

`endAngleInDegree = 230#` – определяем конечный угол дуги в градусах.

```
Dim startAngleInRadian As Double
Dim endAngleInRadian As Double
startAngleInRadian = startAngleInDegree * 3.141592 / 180#
endAngleInRadian = endAngleInDegree * 3.141592 / 180#
```

В этой части программы объявляются переменные углов дуги в радианах. Затем по известной формуле этим переменным присваиваются значения преобразованных углов из градусов в радианы.

```
Set arcObj = ThisDrawing.ModelSpace.AddArc(centerPoint, radius, startAngleInRadian, endAngleInRadian)
```

С помощью этой строки в чертеже создается объект «Дуга», где **AddArc** – метод, добавляющий в чертеж дугу. В скобках указаны параметры дуги – координаты центра, радиус, начальный угол и конечный угол.

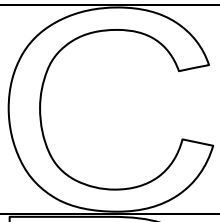
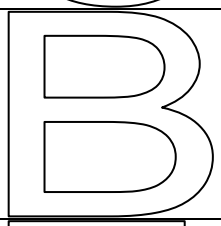
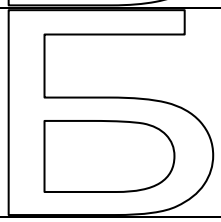
Самостоятельная работа №3

СОЗДАНИЕ ДУГИ С ПОМОЩЬЮ МЕТОДА ADDARC

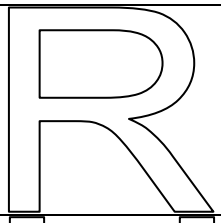
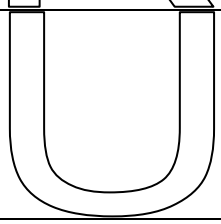
Создайте программу для создания чертежа фигур. Координаты центров, радиусов, углов выберите самостоятельно. При создании чертежей используйте метод AddArc, AddLine . Варианты заданий даны в табл. 3.1.

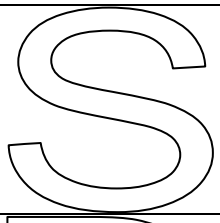
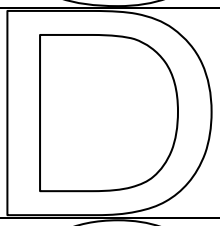
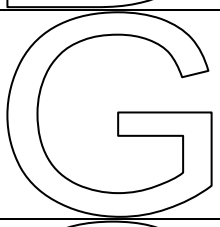
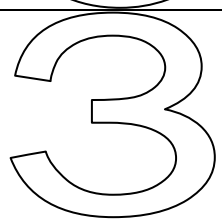
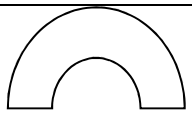
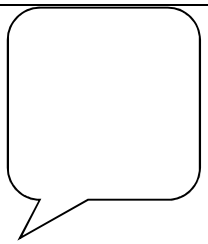
Таблица 3.1

Варианты заданий

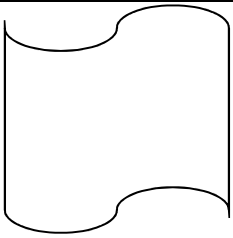
Номер варианта	Задание
1	
2	
3	

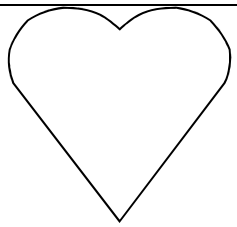
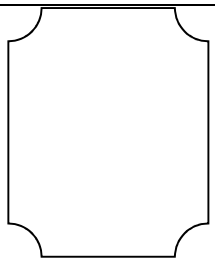
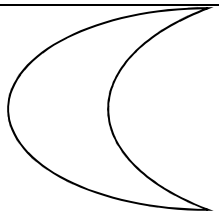
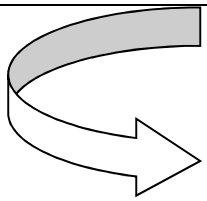
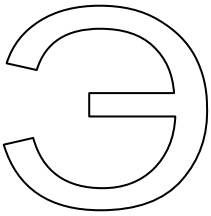
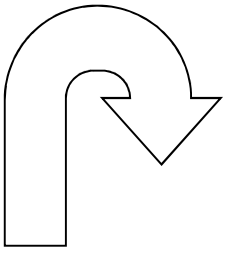
Продолжение табл. 3.1

Номер варианта	Задание
4	
5	

6	
7	
8	
9	
10	
11	

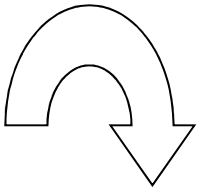
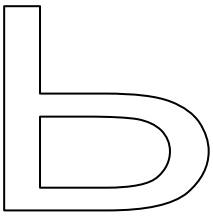
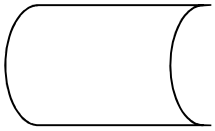
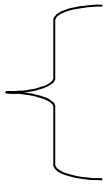
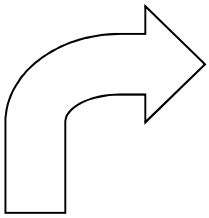
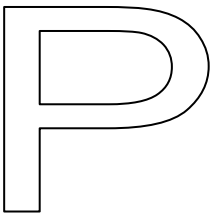
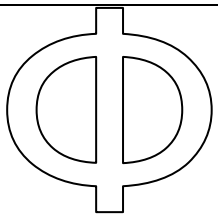
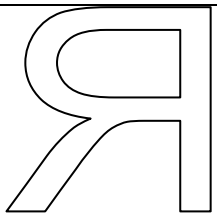
Продолжение табл. 3.1

Номер варианта	Задание
12	

13	
14	
15	
16	
17	
18	

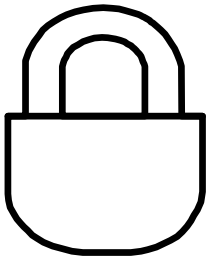
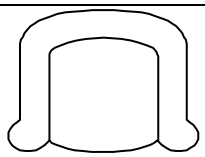
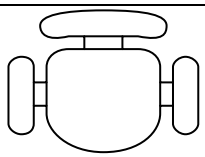
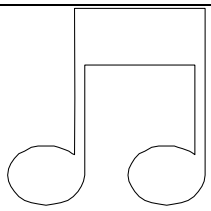
Продолжение табл. 3.1

Номер варианта	Задание

19	
20	
21	
22	
23	
24	
25	
26	

Окончание табл. 3.1

Номер варианта	Задание

27	
28	
29	
30	

СОЗДАНИЕ ГРАФИЧЕСКОГО ПРИМИТИВА – ДВУМЕРНАЯ ПОЛИЛИНИЯ (Lightweightpolyline)

Полилиния представляет собой связанную последовательность сегментов, все эти сегменты являются единым объектом. Полилинии могут состоять из линейных и дуговых сегментов, а также из любых их сочетаний. Полилинии используются, если предполагается работа с набором сегментов как с целым.

Чтобы создать полилинию, используется метод **AddLightweightPolyline.** Чтобы редактировать или узнать информацию о полилинии используют методы и свойства, указанные в табл. 3.2.

Таблица 3.2

Методы, свойства и события объекта Полилиния (LightweightPolyline)

Методы	Свойства	События
AddVertex	Application	Modified
ArrayPolar	Area	
ArrayRectangular	Closed	
Copy	ConstantWidth	
Delete	Coordinate	
Explode	Coordinates	
GetBoundingBox	Document	
GetBulge	Elevation	
GetExtensionDictionary	Handle	
GetWidth	HasExtensionDictionary	
GetXData	Hyperlinks	
Highlight	Layer	
IntersectWith	Length	
Mirror	Linetype	
Mirror3D	LinetypeGeneration	
Move	LinetypeScale	
Offset	Lineweight	
Rotate	Normal	
Rotate3D	ObjectID	
ScaleEntity	OwnerID	
SetBulge	PlotStyleName	
SetWidth	Thickness	
SetXData	TrueColor	
TransformBy	Visible	
Update		

Команда построения полилинии

RetVal = object. AddLightweightPolyline (VerticesList),

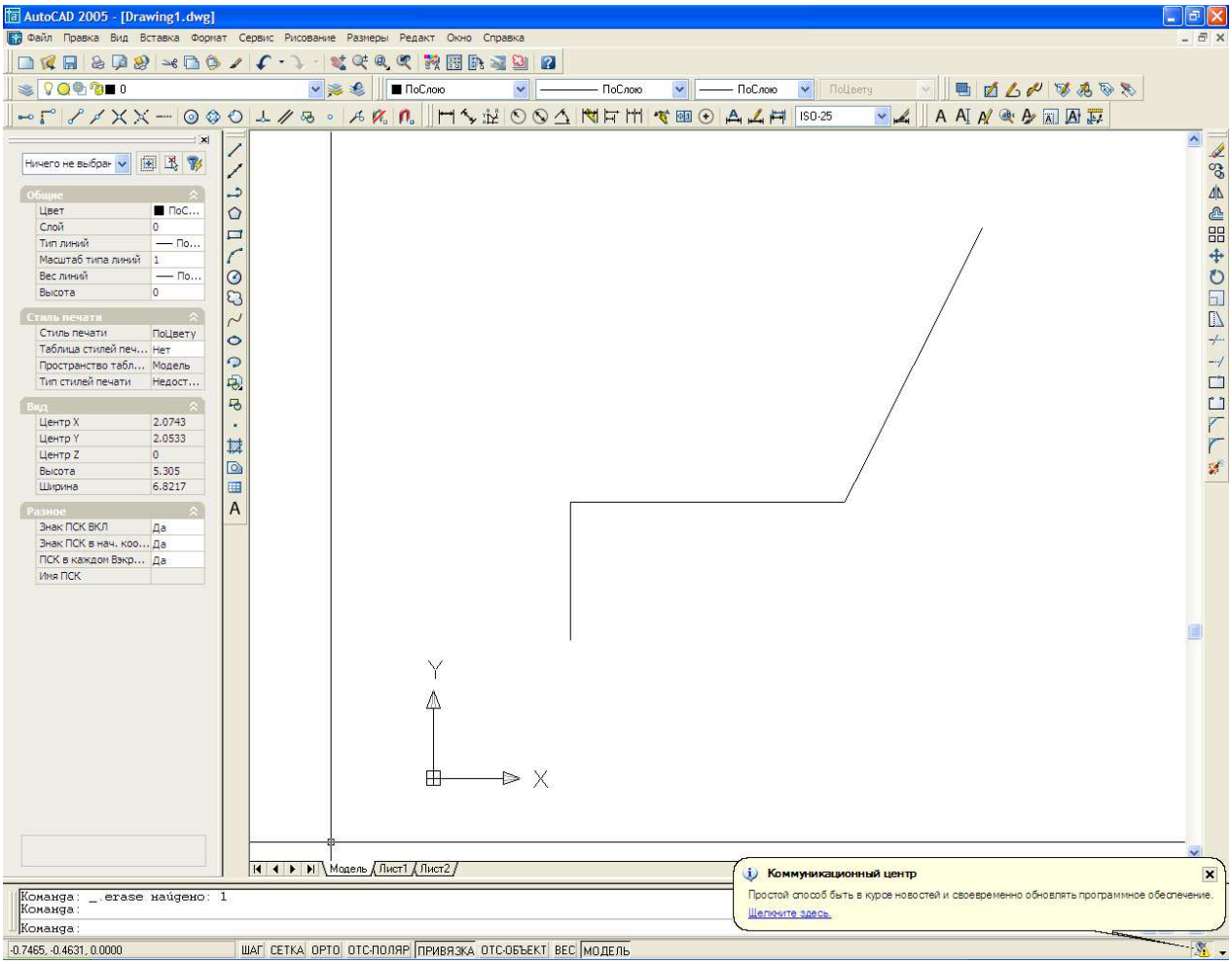
где **RetVal** – название создаваемого объекта **«двумерная полилиния»**; **Object** – объект или объекты, к которым применяется метод **AddLightweightPolyline.** В качестве объектов могут быть использованы **ModelSpace Collection, PaperSpace Collection, Block**; **VerticesList** – координаты точек, входящих в полилинию. Представляет собой одномерный массив типа **Double**, в котором указываются координаты X,Y. Размер массива должен быть кратным 2**.** Для построения полилинии необходимо наличие, как минимум, координат двух точек (четыре элемента массива).

Чтобы добавить дуговой сегмент, вначале создают полилинию с линейным сегментом, а затем применяют метод **SetBulge**, который преобразует линейный сегмент в дуговой.

Пример. Вставьте в проект VBA новый модуль. Поместите в модуль следующую процедуру:

```
    Sub Example_AddLightWeightPolyline()
        'Данный пример создает двумерную полилинию в простран-
стве модели

        Dim plineObj As AcadLWPolyline
        Dim points(0 To 9) As Double

        'Определяем двумерную полилинию
        points(0) = 1: points(1) = 1
        points(2) = 1: points(3) = 2
        points(4) = 2: points(5) = 2
        points(6) = 3: points(7) = 2
        points(8) = 4: points(9) = 4

        'Создаем объект «двумерная полилиния в пространстве мо-
дели

        Set plineObj = ThisDraw-
ing.ModelSpace.AddLightWeightPolyline(points)
        ZoomAll

    End Sub
```

Запустив данную программу с помощью кнопки , располагающейся на панели инструментов, получим на чертеже полилинию (р и с. 3.1).



Р и с. 3.1. Полилиния, построенная с помощью программы

Рассмотрим некоторые строки программы создания объекта «Двумерная полилиния» подробнее.

`Dim plineObj As AcadLWPolyline` – команда объявления объекта «Двумерная полилиния», где **AcadLWPolyline** – тип объявляемого объекта.

`Dim points(0 To 9) As Double` – в этой строке объявляется массив, в котором будут содержаться координаты полилинии.

```
points(0) = 1: points(1) = 1
points(2) = 1: points(3) = 2
points(4) = 2: points(5) = 2
points(6) = 3: points(7) = 2
points(8) = 4: points(9) = 4
```

В данных строках элементам массива присваиваются координаты точек, из которых будет состоять полилиния. Каждая пара элементов массива передает координаты одной точки полилинии. Четный элемент массива – это координата X. Нечетный элемент массива – координата Y. В нашем примере 10 элементов массива – это пять точек полилинии.

```
Set plineObj = ThisDraw-
ing.ModelSpace.AddLightWeightPolyline(points)
```

С помощью этой строки в чертеже создается объект «Двумерная полилиния», где **AddLightWeightPolyline** – метод, добавляющий полилинию в пространство модели чертежа. В скобках указаны параметры полилинии – массив, содержащий координаты.

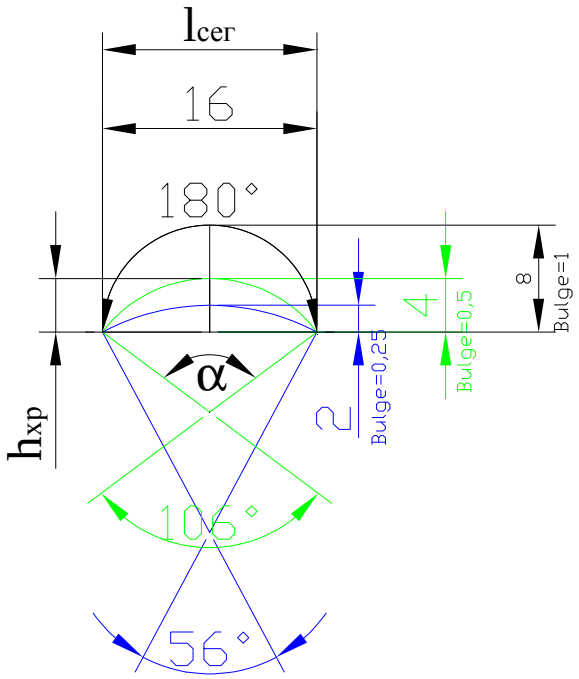
ПРЕОБРАЗОВАНИЕ ПРЯМОЛИНЕЙНОГО СЕГМЕНТА ПОЛИЛИНИИ В ДУГОВОЙ

При создании полилинии с помощью метода **AddLightweightPolyline** невозможно сразу создать дуговой сегмент полилини, поэтому поступают следующим образом: сначала строят линейный сегмент полилинии, а затем преобразуют его в дуговой. Для преобразования линейного сегмента в дуговой используют метод **SetBulge**.

Команда преобразования линейного сегмента в дуговой

object.SetBulge Index, Value,

где Object – объект или объекты, имеющие тип LightweightPolyline, Polyline, к которым применяется данный метод; Index – индекс расположения сегмента в объекте. Первый сегмент имеет индекс 0. Имеет тип Integer; Value – величина выпуклости дугового сегмента для указанного индекса, имеет тип Double.



Р и с. 3.2. Зависимость величины выпуклости
от величин длины сегмента, высоты хорды,
центрального угла дуги

Величина выпуклости равна тангенсу ¼ от центрального угла дуги, начало и конец которой находятся в точках, ограничивающих выбранный сегмент. Отрицательная величина выпуклости указывает, что дуга пойдет по часовой стрелке от первой точки выбранного сегмента до второй, положительная – против часовой.

На р и с. 3.2 показаны дуги с различными величинами выпуклости (bulge).

В общем случае величина выпуклости

$$\text{Bulge} = \text{tg}\left(\frac{\alpha}{4}\right),$$

где α – центральный угол дуги.

Эту же величину можно вычислить следующим образом:

$$\text{Bulge} = \frac{h_{xp}}{l_{сег}} \cdot 2,$$

где h_{xp} – высота хорды; $l_{сег}$ – длина сегмента полилинии.

Например:

$$\text{Bulge} = \text{tg}\left(\frac{106}{4}\right) = 0{,}5;$$

$$\text{Bulge} = \frac{4}{16} \cdot 2 = 0{,}5.$$

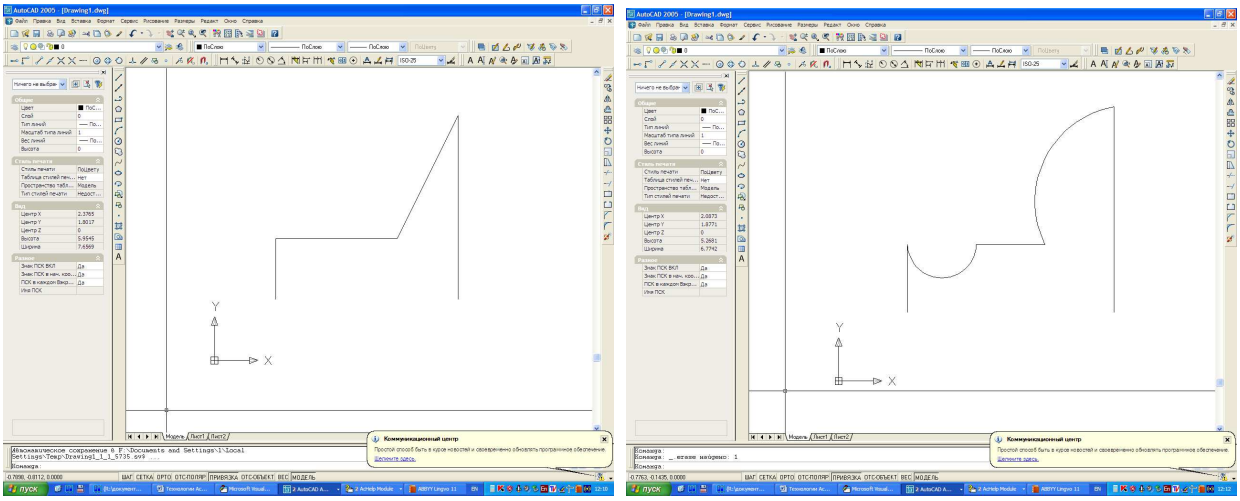
При Bulge = 0 сегмент остается прямолинейным; при Bulge = 1 дуговой сегмент представляет собой половину круга.

Пример. Вставьте в проект VBA новый модуль. Поместите в модуль следующую процедуру:

```
Sub Example_SetBulge()

    ' Данный пример создает полилинию в пространстве модели.
        ' И затем изменяет линейные сегменты полилинии на дуговые.

        Dim plineObj As AcadLWPolyline
        Dim points(0 To 11) As Double

        'Определяем точки двумерной полилинии
        points(0) = 1: points(1) = 1
        points(2) = 1: points(3) = 2
        points(4) = 2: points(5) = 2
        points(6) = 3: points(7) = 2
        points(8) = 4: points(9) = 4
        points(10) = 4: points(11) = 1

        'Создаем объект "двумерная полилиния" в пространстве модели
        Set plineObj = ThisDrawing.ModelSpace.AddLightWeightPolyline(points)
        ZoomAll

        'Преобразуем в дуговые 1-ый и 3-ий сегменты полилинии
```

```
        plineObj.SetBulge 1, 1
        plineObj.SetBulge 3, -0.5
        plineObj.Update

End Sub
```

Запустив данную программу с помощью кнопки  , располагающейся на панели инструментов, получим на чертеже полилинию (р и с. 3.3).



а *б*

Р и с. 3.3. Полилиния, построенная с помощью программы:
а – до применения метода SetBulge; *б* – после применения метода SetBulge

Полилиния в данном примере строится так же, как и в предыдущем. Затем с помощью метода **SetBulge** первый и третий сегменты преобразуются в дуговые. Следует обратить внимание, что отсчет сегментов начинается с 0. Первый сегмент имеет величину выпуклости 1, поэтому имеет форму полукруга. Так как эта величина положительная, то дуговой сегмент получил выпуклость в такую сторону, что направление дуги от вершины 1 до вершины 2 стало по часовой стрелке. Третий сегмент имеет величину выпуклости -0,5, и ее направление против часовой стрелки.

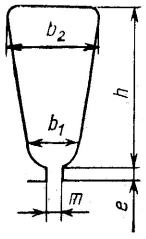
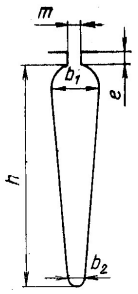
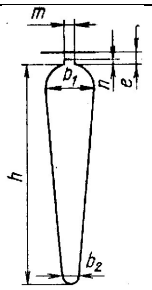
`plineObj.Update` – данная команда (метод **Update**) обновляет объект **plineObj** в чертеже.

Самостоятельная работа №4
СОЗДАНИЕ ПОЛИЛИНИИ С ПОМОЩЬЮ МЕТОДОВ ADDLIGHTWEIGHTPOLYLINE И SETBULGE

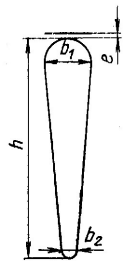
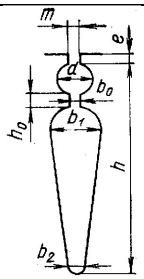
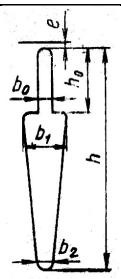
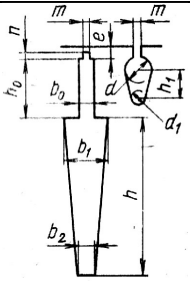
Создайте программу для вычерчивания паза листов ротора либо статора асинхронного двигателя. При создании чертежей используйте методы **AddLightweightPolyline** и **SetBulge**. Варианты заданий даны в табл. 4.1.

Таблица 4.1

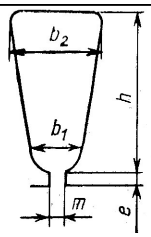
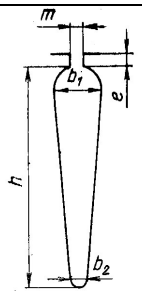
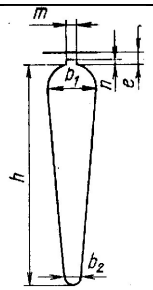
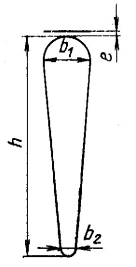
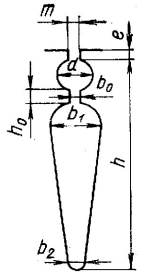
Варианты заданий

Номер варианта	Типоразмер электро-двигателя	Форма паза	Размеры, мм
1	4А71В4У3	 Паз статора	b_1 = 5,2 b_2 = 7,3 h = 11,6 e = 0,5 m = 2,0
2	4А71В4У3	 Паз ротора	b_1 = 5,0 b_2 = 1,5 h = 12,6 e = 0,5 m = 1,0
3	4А160S4У3	 Паз ротора	b_1 = 7,5 b_2 = 3,5 h = 34,0 e = 1,0 m = 1,5 n = 0,7

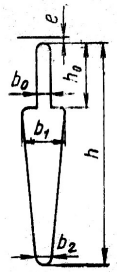
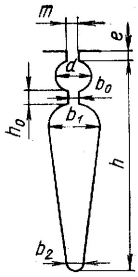
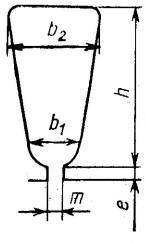
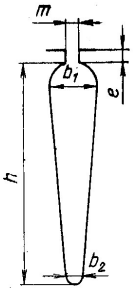
Продолжение табл. 4.1

Номер варианта	Типоразмер электродвигателя	Форма паза	Размеры, мм
4	4А160S2У3	 Паз ротора	b_1 = 7,0 b_2 = 4,9 h = 29,0 e = 1,0
5	4АР160S4У3	 Паз ротора	b_1 = 6,1 b_2 = 3,3 h = 34,0 e = 1,0 m = 1,5 h_0 = 2,0 b_0 = 2,0 d = 8,9
6	4А250S2У3	 Паз ротора	b_1 = 6,8 b_2 = 3,6 h = 36,5 e = 1,5 h_0 = 13,5 b_0 = 4,0
7	4А180М12/6У3	 Паз ротора	b_1 = 6,8 b_2 = 4,4 h = 16,0 h_1 = 8,0 e = 1,0 m = 1,5 n = 0,7 h_0 =14 b_0 =3,4 d =4,5 d_1 =3,5

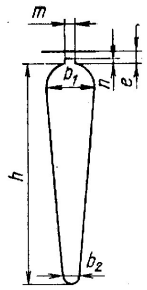
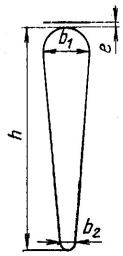
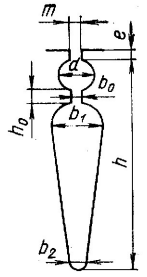
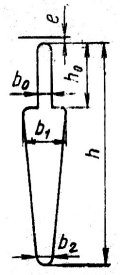
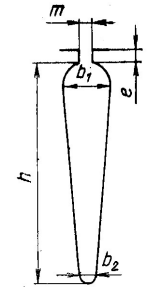
Продолжение табл. 4.1

Номер варианта	Типоразмер электродвигателя	Форма паза	Размеры, мм
8	4А112МА8У3	 Паз статора	$b_1 = 4,5$ $b_2 = 6,3$ h = 17,5 e = 0,5 m = 3,0
9	4А90L6У3	 Паз ротора	$b_1 = 5,0$ $b_2 = 2,1$ h = 16,5 e = 0,5 m = 1,0
10	4А180S4У3	 Паз ротора	$b_1 = 8,9$ $b_2 = 3,2$ h = 39,8 e = 1,0 m = 1,5 n = 0,7
11	4А180М2У3	 Паз ротора	$b_1 = 9,6$ $b_2 = 4,1$ h = 31,0 e = 0,85
12	4АР160М6У3	 Паз ротора	$b_1 = 4,9$ $b_2 = 2,5$ h = 31,2 e = 1,0 m = 1,5 $h_0 = 2,0$ $b_0 = 2,0$ d = 6,2

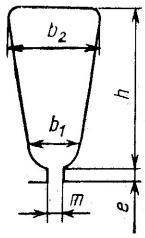
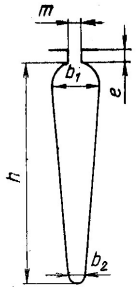
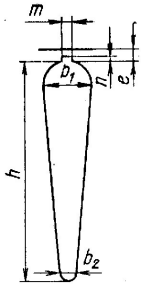
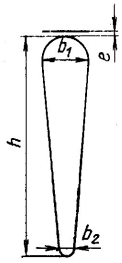
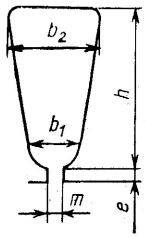
Продолжение табл. 4.1

Номер варианта	Типоразмер электродвигателя	Форма паза	Размеры, мм
13	4А250S2У3		b_1 = 9,3 b_2 = 6,5 h = 40,0 e = 0,5 h_0 = 15,0 b_0 = 5,0
14	4АР180М4У3	 Паз ротора	b_1 = 6,5 b_2 = 3,5 h = 36,8 e = 1,0 m = 1,5 h_0 = 3,0 b_0 = 2,0 d = 8,5
15	4А132М8У3	 Паз статора	b_1 = 4,8 b_2 = 7,1 h = 17,6 e = 0,9 m = 3,5
16	4А132S4У3	 Паз ротора	b_1 = 6,0 b_2 = 2,2 h = 24,7 e = 0,75 m =1,5

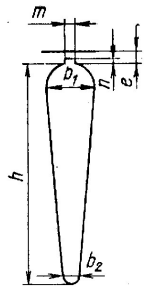
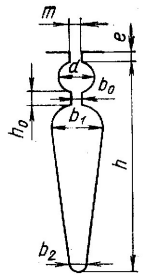
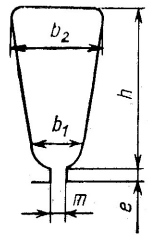
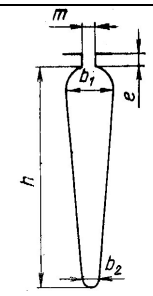
Продолжение табл. 4.1

Номер варианта	Типоразмер электродвигателя	Форма паза	Размеры, мм
17	4А250М8У3	 Паз ротора	b_1 = 8,8 b_2 = 3,4 h = 54,0 e = 1,0 m = 1,5 n = 0,7
18	4А200М2У3	 Паз ротора	b_1 = 6,9 b_2 = 5,6 h = 34,4 e = 1,0
19	4АР200М4У3	 Паз ротора	b_1 = 8,0 b_2 = 3,4 h = 44,0 e = 1,0 m = 1,5 h_0 = 4,0 b_0 = 2,0 d = 9,2
20	4А355S2У3	 Паз ротора	b_1 = 10,4 b_2 = 7,5 h = 42,0 e = 1,0 h_0 = 15,0 b_0 = 5,0
21	4А80В4У3	 Паз ротора	b_1 = 4,5 b_2 = 1,5 h = 16,4 e = 0,5 m = 1,0

Продолжение табл. 4.1

Номер варианта	Типоразмер электродвигателя	Форма паза	Размеры, мм
22	4АА50А2У3	 Паз статора	b_1 = 8,7 b_2 =10,9 h = 9,6 e = 0,5 m = 1,8
23	4АА56А2У3	 Паз ротора	b_1 = 4,1 b_2 = 1,5 h = 10,2 e = 0,5 m = 1,0
24	4АН180S8У3	 Паз ротора	b_1 = 6,2 b_2 = 2,4 h = 40,3 e = 1,0 m = 1,5 n =0,7
25	4А225М2У3	 Паз ротора	b_1 = 8,1 b_2 = 4,8 h = 35,0 e = 1,0
26	4П71А2У3	 Паз статора	b_1 = 5,9 b_2 = 7,5 h = 9,3 e = 0,5 m = 2,0

Окончание табл. 4.1

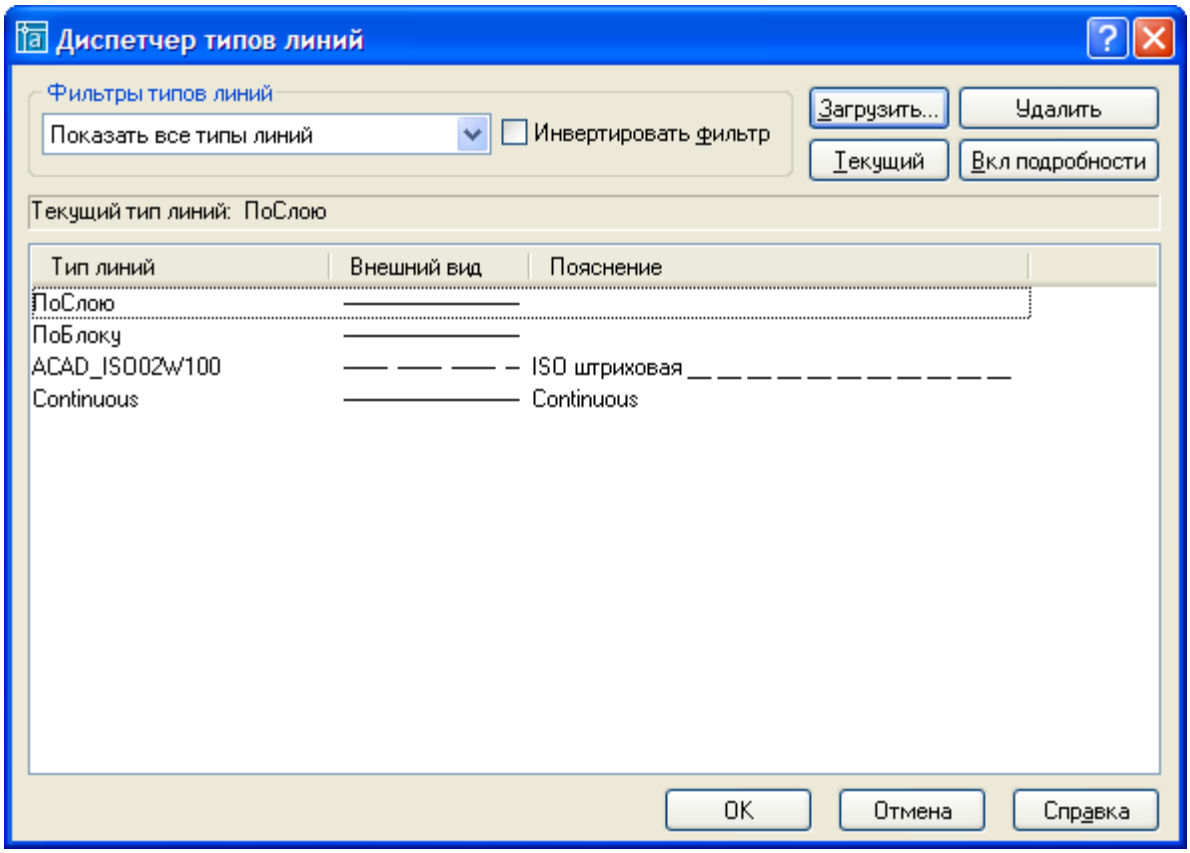
Номер варианта	Типоразмер электродвигателя	Форма паза	Размеры, мм
27	4АС250S6У3	 Паз ротора	b_1 = 8,8 b_2 = 7,5 h = 18,8 e = 1,0 m = 1,5 n = 0,7
28	4АР250S4У3	 Паз ротора	b_1 = 7,0 b_2 = 3,0 h = 50,0 e = 1,0 m = 1,5 h_0 = 5,0 b_0 = 2,0 d =7,0
29	4А90L6У3	 Паз статора	b_1 = 4,7 b_2 = 6,6 h = 13,8 e = 0,5 m = 2,7
30	4А100S4У3	 Паз ротора	b_1 = 5,1 b_2 = 1,5 h = 19,3 e = 0,5 m =1,0

Типы линий применяются для того, чтобы можно было отличить один объект от другого, тем самым делая работу с рисунком более удобной. Тип линий описывается повторяющейся последовательностью штрихов, точек и пробелов, наносимых вдоль прямой или кривой. Типы линий можно присваивать как слоям, так и отдельным объектам рисунка.

В описании типа линий можно задавать масштаб для регулирования относительных длин штрихов и пробелов. Пользователь имеет возможность создавать собственные типы линий.

Типы линий различаются по именам, а сама последовательность штрихов и точек, относительные длины штрихов и пробелов, а также характеристики включаемых текстовых элементов и форм задаются в описании типа линий. Можно использовать имеющиеся в AutoCAD типы линий или создавать собственные.

Типы линий хранятся в файлах описаний типов линий. Эти файлы имеют расширение .lin. В одном LIN-файле может храниться множество описаний простых и сложных типов линий. Пользователь может добавлять новые типы линий в существующие LIN-файлы, а также создавать новые LIN-файлы. Содержимое LIN-файлов можно просмотреть в диспетчере типов линий (р и с. 4.1.).



Р и с. 4.1. Менеджер типов линий

Для просмотра и загрузки типа линий в среде AutoCAD нужно поступить следующим образом.

1. Из меню «Формат» выбрать «Типы линий».

2. В диалоговом окне «Диспетчер типов линий» нажать кнопку «Загрузить».

3. В диалоговом окне «Загрузка/перезагрузка типов линий» выбрать тип линий и нажать «ОК».

Если в списке отсутствует необходимый тип линий, нажать кнопку «Файл». В диалоговом окне «Выбор файла типов линий» выбрать LIN-файл для загрузки и нажать «Открыть». Типы линий из выбранного LIN-файла появляются в списке. Выбрать тип линий и нажать «ОК».

Для выделения нескольких типов линий нужно удерживать в нажатом состоянии клавишу CTRL; для выделения сразу всех типов линий, расположенных между указанными типами, следует использовать клавишу SHIFT, удерживая ее в нажатом состоянии.

4. Нажать «ОК».

Прежде чем использовать какой-либо тип линий, его нужно загрузить.

В поставку AutoCAD включены LIN-файлы acad.lin и acadiso.lin.

Каждый из них предназначен для определенной системы измерений:

- для британских единиц необходимо использовать файл acad.lin.
- для метрической системы следует использовать файл acadiso.lin.

Оба файла содержат несколько сложных типов линий.

Тип линии можно присвоить уже существующим объектам или сделать тип линии активным, после чего дальнейшие графические построения будут вестись указанным типом линии.

ЗАГРУЗКА ТИПА ЛИНИИ

Команда построения отрезка

object.Load LineTypeName, FileName,

где **Object** – коллекция **Linetypes**. С помощью метода **Load** в данную коллекцию загружается указанный тип линии из файла, содержащего набор типов линий; LineTypeName **– имя загружаемо-**

го типа линии. Имеет тип String**. Режим: только ввод (input-only); FileName –** имя файла, содержащего набор типов линий. Имеет тип **String**. Режим: только ввод (input-only).

УСТАНОВКА УКАЗАННОГО ТИПА ЛИНИИ АКТИВНЫМ (ТЕКУЩИМ) ДЛЯ РИСОВАНИЯ

Команда установки указанного типа линии активным (текущим)

object.ActiveLinetype,

где **Object** – объект Document, для которого устанавливается свойство ActiveLinetype; **ActiveLinetype** – свойство, которое позволяет активировать указанный тип линии (**Linetype**), где **Linetype** – объект, характеризующий линию, которая состоит из комбинации тире, точек и пробелов.

Перед использованием данного свойства нужно загрузить необходимый тип линии с помощью метода **Load**.

Свойство **ActiveLinetype** используют до начала рисования. Все построения начинаются вестись тем типом линии, который указан в данном свойстве. Если нужно изменить тип линии уже у существующего объекта, используют свойство **Linetype**.

Пример. Вставьте в проект VBA новый модуль. Поместите в модуль следующую процедуру:

```
Sub Load_ActiveLinetype()

   'Новый тип линии
   Dim linetypeName As String
   linetypeName = "ACAD_ISO10W100"

   'Данный тип линии заносится в коллекцию уже существующих (загруженных) типов линий
   'Тип линии заносится из файла "acadiso.lin"

   On Error Resume Next 'При ошибке загрузки линии пропускается следующая строка программы
   ThisDrawing.Linetypes.Load linetypeName, "acadiso.lin"
```

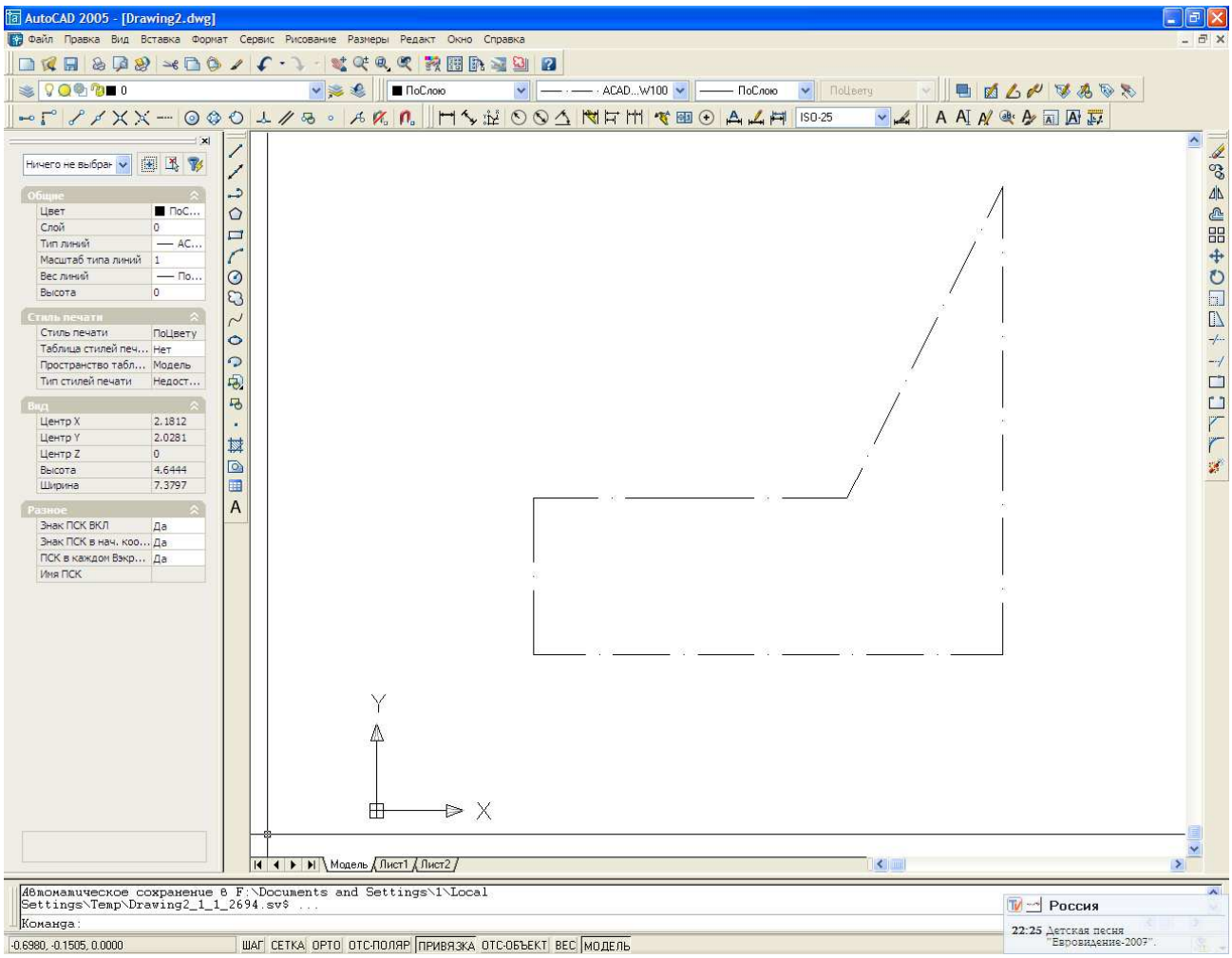
```
    'Если данный тип линии уже существует, то появляется сообщение о том, что этот тип линии уже существует и происходит загрузка линии
    If Err.Description = "Duplicate record name" Then

    MsgBox "Тип линии " & linetypeName & "' уже существует.", , "Загрузка линии"

    End If

   'Активным типом линии становится "ACAD_ISO10W100" и построения будут вестись с данным типом линии  ("осевая линия")
    ThisDrawing.ActiveLinetype = ThisDrawing.Linetypes.Item("ACAD_ISO10W100")

      ' Создание полилинии
      Dim plineObj As AcadLWPolyline
      Dim points(0 To 11) As Double
      points(0) = 1: points(1) = 1
      points(2) = 1: points(3) = 2
      points(4) = 2: points(5) = 2
      points(6) = 3: points(7) = 2
      points(8) = 4: points(9) = 4
      points(10) = 4: points(11) = 1
      Set plineObj = ThisDrawing.ModelSpace.AddLightWeightPolyline(points)
      plineObj.Closed = True
      plineObj.LinetypeScale = 0.03

End Sub
```

Запустив данную программу с помощью кнопки  , располагающейся на панели инструментов, получим на чертеже полилинию (рис. 4.2).

`Dim linetypeName As String` – объявляется переменная типом **String**, в которой будет содержаться имя типа линии.

`linetypeName = "ACAD_ISO10W100"` – переменной присваивается название; такое название должно содержаться в файле, содержащем типы линии.

`ThisDrawing.Linetypes.Load linetypeName, "acadiso.lin"` – в коллекцию **Linetypes** (типы линий) объекта **ThisDrawing** (текущий чертеж) загружается тип линии, содержащийся в файле "**acadiso.lin**", название типа линии содержится в переменной **linetypeName**.



Р и с. 4.2. Полилиния, построенная с использованием свойства ActiveLinetype

Рассмотрим некоторые строки программы подробнее.

`ThisDrawing.ActiveLinetype = ThisDrawing.Linetypes.Item("ACAD_ISO10W100")` – свойству **ActiveLinetype** присваивается название типа линии, загруженного в коллекцию **Linetypes**. Название типа линии указано в скобках. После выполнения данной команды все построения ведутся с указанным типом линии. Изменение типа линии объекта происходит до его построения.

`plineObj.LinetypeScale = 0.03` – свойству **LinetypeScale** (масштаб типа линий) присваивается значение 0,03. Без использования этого свойства у объектов малых размеров тип линии не будет виден на чертеже, а объект будет нарисован сплошной линией. Описание этого свойства будет рассмотрено ниже.

ИЗМЕНЕНИЕ ТИПА ЛИНИЙ У СУЩЕСТВУЮЩЕГО ОБЪЕКТА

Команда установки указанного типа линии активным (текущим)

object.Linetype,

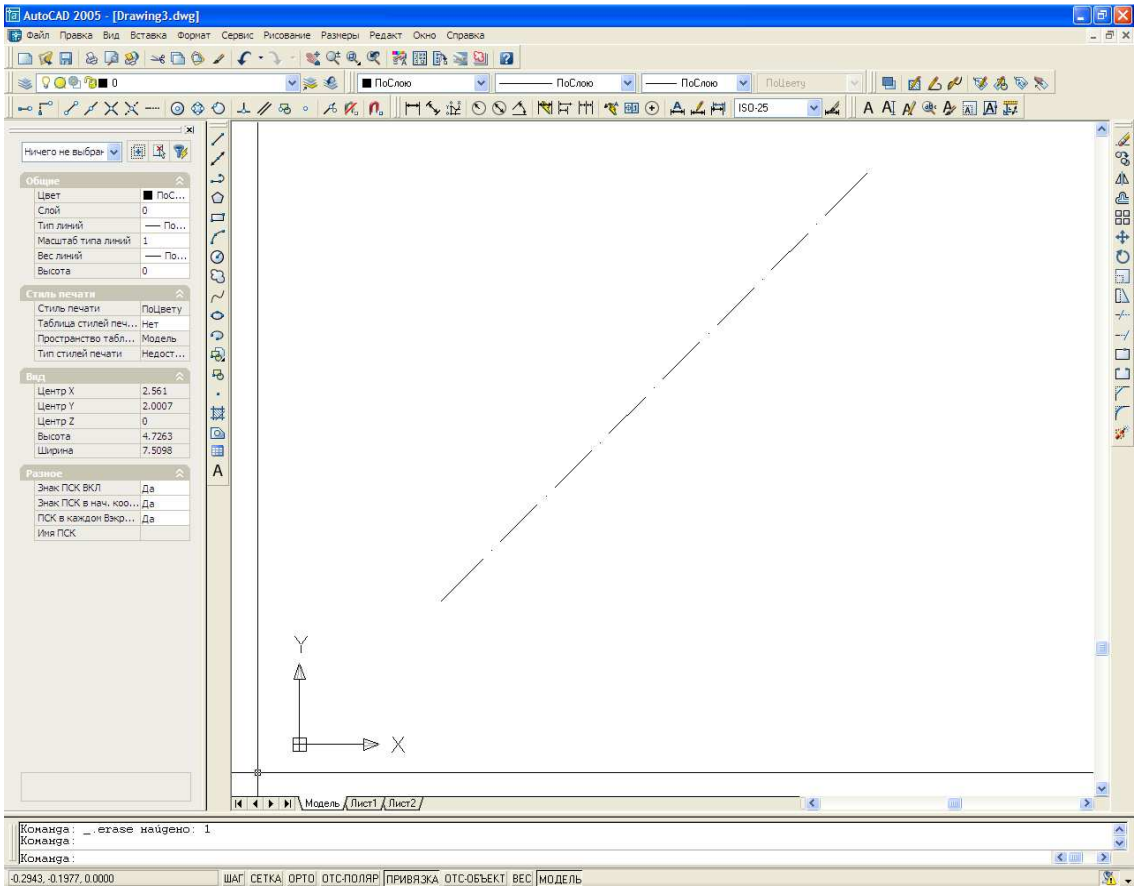
где **object** – все объекты рисования, у которых изменяется тип линии с помощью свойства **Linetype**; **Linetype** – свойство, устанавливающее или определяющее тип линии существующего объекта. Режим – чтение-запись (read-write).

Пример. Вставьте в проект VBA новый модуль. Поместите в модуль следующую процедуру:

```
    Sub Example_Linetype()
        'Данный пример пытается найти уже загруженный тип
линии "ACAD_ISO10W100"
        'Если данный тип не найден, то он добавляется из
файла acadiso.lin
        'Затем создается отрезок и к нему применяется тип
линии "ACAD_ISO10W100"

        'Поиск типа линии "ACAD_ISO10W100" в загруженных
типах линии
        Dim entry As AcadLineType
        Dim found As Boolean
        found = False
        For Each entry In ThisDrawing.Linetypes
            If StrComp(entry.Name, "ACAD_ISO10W100", 1) =
0 Then
                found = True
                Exit For
            End If
        Next
        If Not (found) Then ThisDrawing.Linetypes.Load
"ACAD_ISO10W100", "acadiso.lin"

        ' Создание отрезка
        Dim lineObj As AcadLine
        Dim startPoint(0 To 2) As Double
        Dim endPoint(0 To 2) As Double
        startPoint(0) = 1#: startPoint(1) = 1#:
startPoint(2) = 0#
        endPoint(0) = 4#: endPoint(1) = 4#: endPoint(2) = 0#
```

```
        Set lineObj = ThisDraw-
ing.ModelSpace.AddLine(startPoint, endPoint)

        ' Изменение типа линии отрезка
        lineObj.Linetype = "ACAD_ISO10W100"
        ' Установка масштаба типа линии
        lineObj.LinetypeScale = 0.03
        ZoomAll

End Sub
```

Запустив данную программу с помощью кнопки , располагающейся на панели инструментов, получим на чертеже отрезок (р и с. 4.3).



Р и с. 4.3. Отрезок, построенный программой
с использованием свойства LineType

Рассмотрим программу подробнее.

`Dim entry As AcadLineType` – в данной строке объявляется объект **AcadLineType**

```
Dim found As Boolean
found = False
```

В этих строках объявляется логическая переменная, и ей присваивается значение False (ложь). Это значение будет оставаться таким, пока не будет найден указанный тип линии.

`For Each entry In ThisDrawing.Linetypes` – цикл, который перебирает типы линий в коллекции типов линий текущего чертежа.

```
If StrComp(entry.Name, "ACAD_ISO10W100", 1) = 0 Then
    found = True
    Exit For
End If
```

Если имя перебираемого типа оказывается "ACAD_ISO10W100", то переменной **found** присваивается значение **True** (Истина), и происходит досрочный выход из цикла.

```
If Not (found) Then ThisDrawing.Linetypes.Load
"ACAD_ISO10W100", "acadiso.lin"
```

Если искомый тип линии не найден в коллекции загруженных типов, то осуществляется загрузка данного типа из файла. После этой строки осуществляется построение отрезка. Как строится отрезок, было рассмотрено выше.

`lineObj.Linetype = "ACAD_ISO10W100"` – свойству **Linetype** построенного отрезка присваивается значение (название типа линии), указанное после знака «=». Изменение типа линии объекта происходит после его построения.

`lineObj.LinetypeScale = 0.03` – свойству **LinetypeScale** (масштаб типа линий) присваивается значение 0,03. Без использования этого свойства у объектов малых размеров тип линии не будет виден на чертеже, а объект будет нарисован сплошной линией. Описание этого свойства будет рассмотрено ниже.

МАСШТАБИРОВАНИЕ ТИПОВ ЛИНИЙ

Один и тот же тип линий может использоваться в различных масштабах. При этом коэффициент масштабирования пользователь может задавать как для всех объектов сразу, так и отдельно для каждого объекта.

По умолчанию глобальный и индивидуальный масштабы типов линий имеют значение 1.0. Чем меньше масштаб, тем большее число элементарных фрагментов типа линий умещается в единице длины. Так, например, при коэффициенте 0.5 в одной единице длины умещается два элементарных фрагмента типа линий. Если длина объекта меньше длины элементарного фрагмента типа линий, то такой объект отрисовывается сплошной линией. В таких случаях для отображения слишком коротких отрезков можно использовать малые значения коэффициента масштабирования типов линий.

Команда масштабирования типа линии объекта

object.LinetypeScale,

где **object** – все объекты рисования, у которых изменяется масштаб типа линии с помощью свойства **LinetypeScale**; **Linetype** – свойство, устанавливающее или определяющее масштаб типа линии существующего объекта. Режим – чтение-запись (read-write). Имеет тип Double. Может быть только положительной величиной. По умолчанию равно 1.

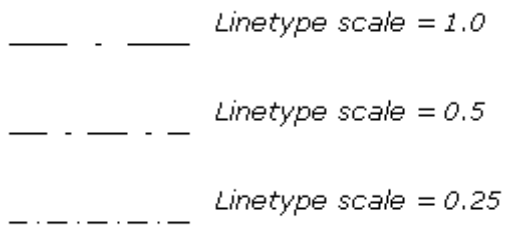


Рис. 4.4. Пояснительный рисунок
к применению свойства LinetypeScale

Как используется свойство **LinetypeScale** (рис. 4.4) было рассмотрено выше в примерах использования типов линий.

ВЕС (ТОЛЩИНА) ЛИНИИ

Веса линий могут применяться для графического выделения различных объектов как на экране, так и при выводе на печать.

Веса линий позволяют получать тонкие и толстые линии, что полезно при оформлении чертежей (построение разрезов, сечений, нанесении размеров), карт и т.д. Например, назначив слоям различные веса линий, можно облегчить идентификацию линий объектов, вспомогательных линий, осевых линий. Для отображения линий в соответствии с весами в среде AutoCAD следует включить кнопку «ВЕС» в строке состояния.

Веса линий могут использоваться при выводе любых объектов за исключением шрифтов True Type, растровых изображений, точек и заливок (для 2М фигур). На виде в плане толщина начертания широкой полилинии определяется ее шириной, а не весом линий, однако на других видах отображение широких полилиний выполняется с учетом установленных весов линий. Информация о весе линий сохраняется при экспорте рисунка в другие форматы и при копировании объектов в буфер обмена.

В пространстве модели каждому значению веса линий соответствует определенное число пикселов, определяющих видимую на экране толщину линий, независимую от выполнения зумирования. В связи с этим в пространстве модели видимая толщина линий может не соответствовать их действительной толщине. Так, например, для отображения полилинии с действительной толщиной 5 мм следует устанавливать не вес линий, а толщину полилинии в соответствующее значение.

Веса линий можно также использовать для управления выводом объектов на печать.

Команда использования веса линии

object.Lineweight,

где **Object** – все объекты рисования, у которых изменяется вес линии с помощью свойства **Lineweight**, а также объект **Layer** (слой); **Lineweight** – свойство, определяющее вес (толщину линии) для указанного нарисованного объекта либо для установки веса линии по умолчанию, после чего все построения будут вестись указанным весом линии. Режим – чтение-запись (read-write) Данное свойство может принимать следующие значения:

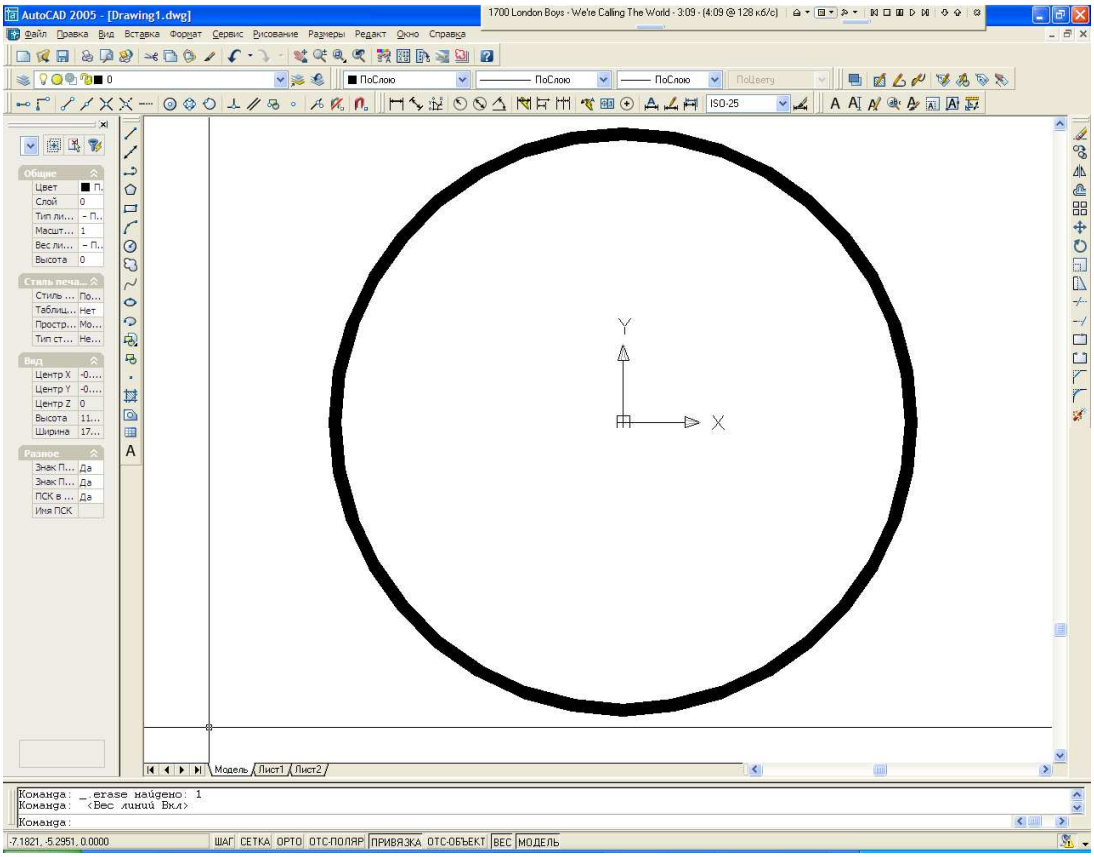
acLnWtByLayer, acLnWtByBlock, acLnWtByLwDefault, acLnWt000, acLnWt005, acLnWt009, acLnWt013, acLnWt015, acLnWt018, acLnWt020, acLnWt025, acLnWt030, acLnWt035, acLnWt040, acLnWt050, acLnWt053, acLnWt060, acLnWt070, acLnWt080, acLnWt090, acLnWt100, acLnWt106, acLnWt120, acLnWt140, acLnWt158, acLnWt200, acLnWt211

Вес линии величиной 0 строит тончайшую линию, позволяемую указанным печатающим устройством, и показывает толщину в 1 пиксель в пространстве модели.

Пример. Вставьте в проект VBA новый модуль. Поместите в модуль следующую процедуру:

```
    Sub Example_LineWeight()
        ' Данный пример создает окружности в пространстве модели и за-
тем
        ' определяет текущий вес линии у окружности. Затем вес линии
        ' изменяется на новую величину.

        Dim circleObj As AcadCircle
        Dim centerPoint(0 To 2) As Double
        Dim radius As Double

        ' Определяем окружность
        centerPoint(0) = 0#: centerPoint(1) = 0#: centerPoint(2) = 0#
        radius = 5#

        ' Создаем окружность в пространстве модели
        Set circleObj = ThisDrawing.ModelSpace.AddCircle(centerPoint,
radius)
        ZoomAll

        ' Определяем вес линии у окружности
        MsgBox "Текущий вес линии для окружности - " & cir-
cleObj.Lineweight
        ' Изменяем вес линии объекта
        circleObj.Lineweight = acLnWt211
        circleObj.Update
        MsgBox "Текущий вес линии для окружности - " & cir-
cleObj.Lineweight
    End Sub
```

Запустив данную программу с помощью кнопки , располагающейся на панели инструментов, получим в чертеже окружность. Если затем в строке состояния включить кнопку «ВЕС», увидим, что толщина линии изменилась (р и с. 4.5).



Р и с. 4.5. Окружность, построенная программой
с использованием свойства Lineweight

Рассмотрим программу подробнее.

```
MsgBox "Текущий вес линии для окружности - " & circleObj.Lineweight
```

В данной строке с помощью инструкции **MsgBox** выводится сообщение о текущем весе линии. Поскольку данная инструкция не входит в тематику учебного пособия, то ее полное описание приводиться здесь не будет. Подробное описание по использованию инструкции **MsgBox** можно найти в справочной системе VBA либо в другой литературе.

circleObj.Lineweight – в данной строке возвращается (режим: read) текущее значение веса лини объекта **circleObj**.

`circleObj.Lineweight = acLnWt211` – в данной строке, в отличие от предыдущей, объекту устанавливается (режим: write) вес линии.

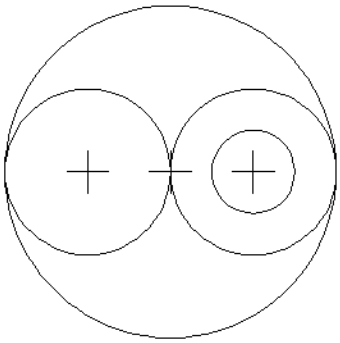
Самостоятельная работа №5
ЗАГРУЗКА ТИПОВ ЛИНИЙ С ПОМОЩЬЮ МЕТОДА. УСТАНОВКА ТИПА ЛИНИИ. УСТАНОВКА ВЕСА ЛИНИИ

На основе программы, созданной в самостоятельной работе №2, создайте программу, в которой окружности будут выполняться различными типами линий и различными весами. При загрузке типов линий используйте файл acadiso.lin. Если при построении линии отображаются сплошной линией, используйте свойство **LinetypeScale**. Если тип линии отсутствует в файле acadiso.lin, обратитесь к преподавателю, чтобы он назначил другой тип линии.

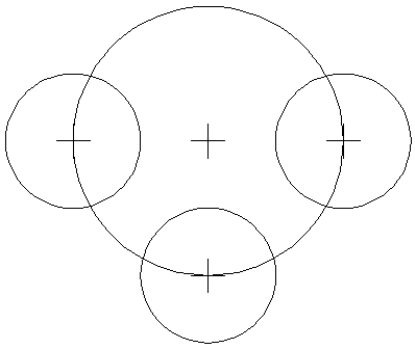
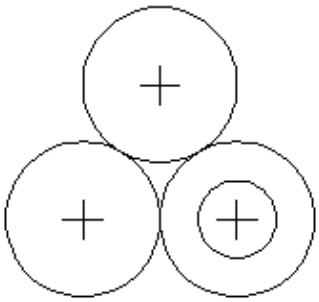
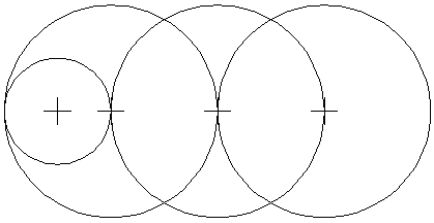
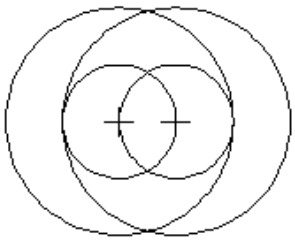
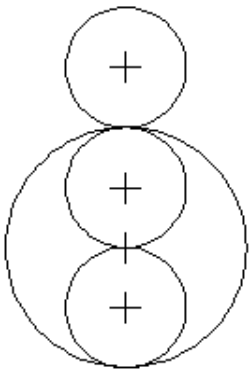
При создании чертежей используйте метод **Load**, свойства **ActiveLinetype, Linetype, LinetypeScale, Lineweight**. Варианты заданий даны в табл. 5.1.

Таблица 5.1

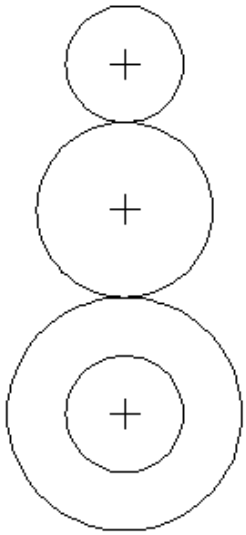
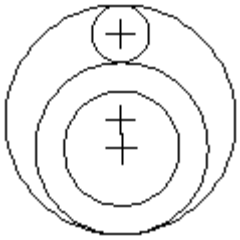
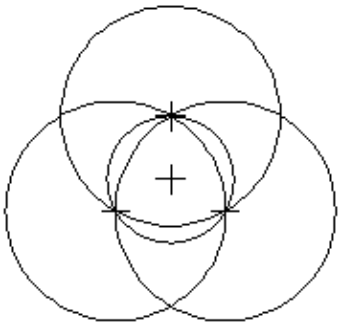
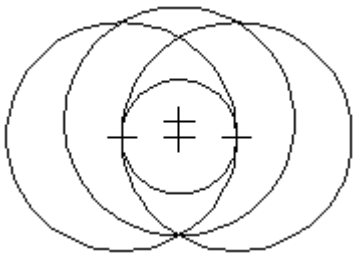
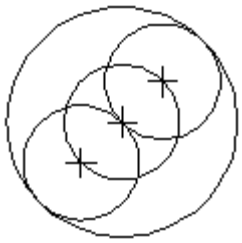
Варианты заданий

Номер варианта	Задание	Типы линий	Вес линий, мм
1		Пунктирная ACAD_ISO13W100 невидимая	0,25 0,60 1,00

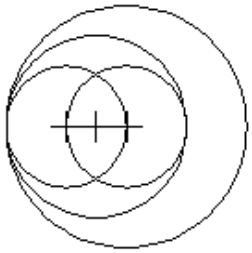
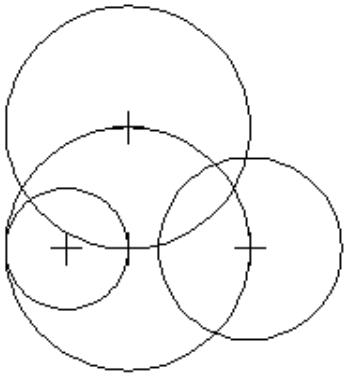
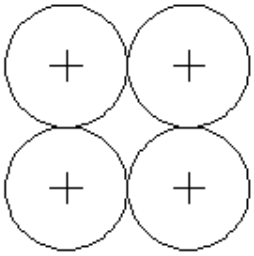
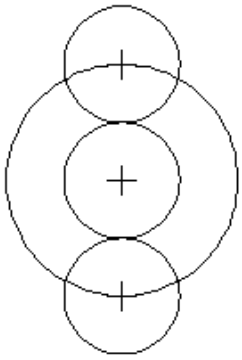
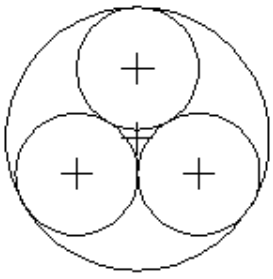
Продолжение табл. 5.1

Номер варианта	Задание	Типы линий	Вес линий, мм
2		Рант ACAD_ISO15W100 штриховаяX2 ACAD_ISO08W100	2,00 1,20 0,40 0,80
3		РантX2 осеваяX2 штриховая2 ACAD_ISO12W100	0,25 0,00 0,70 1,58
4		ACAD_ISO06W100 невидимая фантомX2 штрихпунктирнаяX2	0,20 0,13 1,00 1,40
5		Штриховая линия_сгибаX2 рант2 ACAD_ISO04W100	0,90 0,35 0,15 0,09
6		ACAD_ISO10W100 ACAD_ISO05W100 пунктирнаяX2 фантом2	0,60 0,30 2,11 1,06

Продолжение табл. 5.1

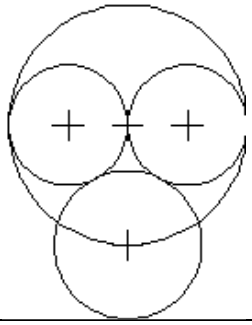
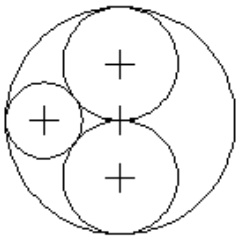
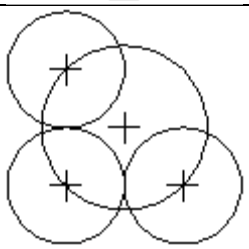
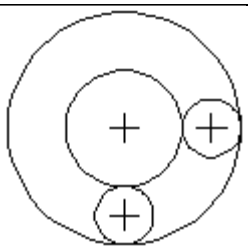
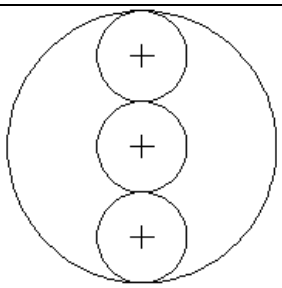
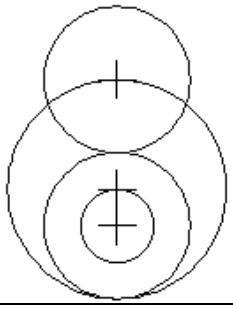
Номер варианта	Задание	Типы линий	Вес линий, мм
7		Штрихпунктирная2 невидимая2 ACAD_ISO07W100 линия_сгиба	0,53 0,50 0,18 0,05
8		ACAD_ISO02W100 ACAD_ISO11W100 осевая2 Фантом	0,13 1,20 0,60 0,20
9		НевидимаяX2 штрихпунктирная пунктирная ACAD_ISO09W100	2,11 0,70 0,25 0,05
10		Линия_сгиба2 ACAD_ISO14W100 ACAD_ISO03W100 пунктирная2	1,06 0,40 2,00 1,58
11		Осевая ACAD_ISO13W100 пунктирная2 штриховаяX2	0,00 0,35 1,40 0,80

Продолжение табл. 5.1

Номер варианта	Задание	Типы линий	Вес линий, мм
12		ACAD_ISO06W100 ACAD_ISO10W100 штрихпунктирнаяX2 осевая2	1,00 0,18 0,50 0,09
13		ACAD_ISO14W100 ACAD_ISO08W100 пунктирнаяX2 фантомX2	0,30 0,90 0,53 0,15
14		Линия_сгибаX2 штриховая ACAD_ISO12W100 ACAD_ISO02W100	0,80 1,40 0,25 0,13
15		ОсеваяX2 ACAD_ISO11W100 невидимая ACAD_ISO04W100	0,09 0,70 0,30 1,00
16		ACAD_ISO03W100 ACAD_ISO13W100 фантом2 пунктирная	0,00 0,15 0,40 2,11

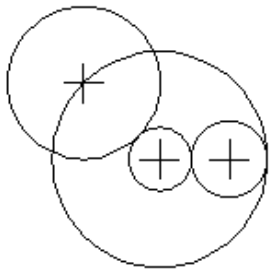
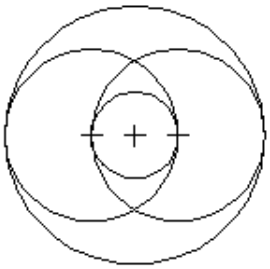
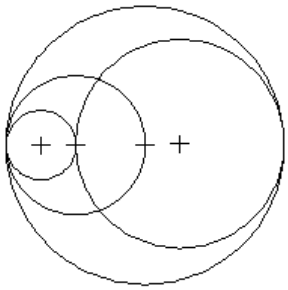
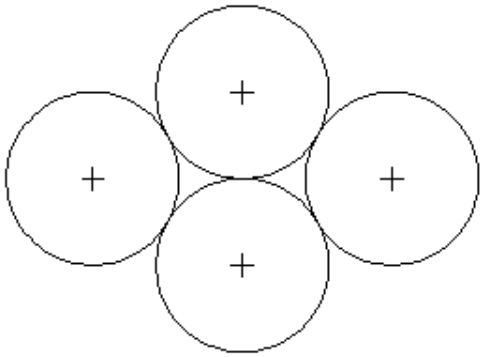
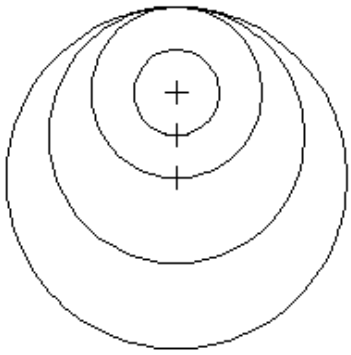
Продолжение табл. 5.1

Номер	Задание	Типы линий	Вес ли-

варианта			ний, мм
17		ACAD_ISO05W100 ACAD_ISO15W100 осевая фантом	2,00 0,05 0,60 0,20
18		ACAD_ISO07W100 линия_сгиба штрихпунктирная рант	0,18 0,35 1,20 1,58
19		ACAD_ISO09W100 рант2 штрихпунктирная2 линия_сгиба2	0,50 0,53 0,90 1,06
20		Невидимая2 невидимаяX2 рантX2 штриховая2	0,90 0,20 0,09 0,00
21		ACAD_ISO13W100 невидимая2 пунктирнаяX2 ACAD_ISO09W100	0,25 0,35 2,11 1,06
22		Рант2 штриховая2 ACAD_ISO15W100 ACAD_ISO05W100	0,05 0,40 1,58 1,00

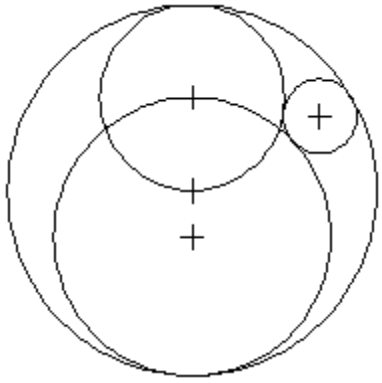
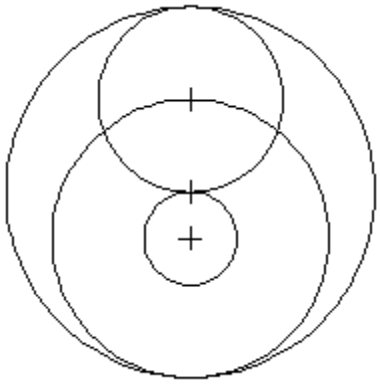
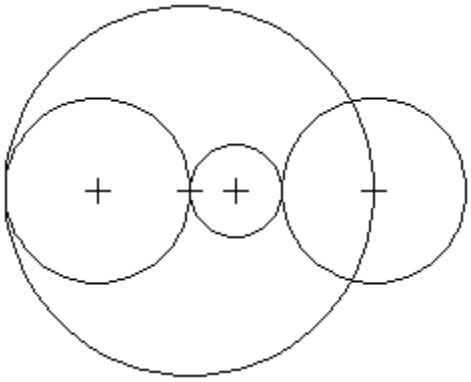
Продолжение табл. 5.1

Номер варианта	Задание	Типы линий	Вес линий, мм

23		ACAD_ISO10W100 невидимаяX2 Рант Штриховая	0,13 0,50 0,80 2,00
24		Линия_сгиба фантом штрихпунктирнаяX2 ACAD_ISO08W100	0,15 0,18 0,53 1,40
25		Осевая пунктирная2 штрихпунктирная ACAD_ISO07W100	0,30 0,60 0,70 1,20
26		ACAD_ISO06W100 линия_сгиба2 пунктирная рантX2	0,15 0,30 0,05 0,70
27		ACAD_ISO04W100 фантом2 осеваяX2 ACAD_ISO14W100	0,18 0,35 0,60 2,00

Окончание табл. 5.1

Номер варианта	Задание	Типы линий	Вес линий, мм

28		ACAD_ISO11W100 линия_сгибаX2 осевая2 фантомX2	0,13 0,40 0,80 1,20
29		ACAD_ISO02W100 ACAD_ISO12W100 невидимая штриховаяX2	0,00 0,25 1,00 1,40
30		ACAD_ISO03W100 штрихпунктирная2 фантом осевая	1,06 0,53 2,11 0,50

Самостоятельная работа №6
ЗАГРУЗКА ТИПОВ ЛИНИЙ С ПОМОЩЬЮ МЕТОДА. УСТАНОВКА ТИПА ЛИНИИ. УСТАНОВКА ВЕСА ЛИНИИ

На основе программы, созданной в самостоятельной работе №4, создайте программу, в которой паз будет выполняться весом 1мм. Постройте для паза осевую линию весом 0,5мм. При загрузке типов линий используйте файл acadiso.lin. Если при построении линии отображаются сплошной линией, используйте свойство **LinetypeScale**. Если тип линии отсутствует в файле acadiso.lin, обратитесь к преподавателю, чтобы он назначил другой тип линии.

При создании чертежей используйте метод **Load**, свойства **ActiveLinetype, Linetype, LinetypeScale, Lineweight**.

ЗАКЛЮЧЕНИЕ

В работе были рассмотрены основы программирования для AutoCAD, построение простейших объектов (отрезок, окружность, дуга, полилиния). Кроме того, были рассмотрены некоторые свойства этих объектов, необходимые для построения чертежей (типы линий и вес линий). Выполнив предложенные самостоятельные задания, можно на практике научиться использовать объекты, методы и свойства объектов AutoCAD.

Естественно, что в рамках пособия была рассмотрена лишь малая часть возможностей программирования для AutoCAD. Оно и не претендует на полноту охвата. Цель работы – показать, что создавать графические объекты с помощью программы совсем не сложно.

Имея под рукой такой инструмент, как VBA, инженер может решать огромный комплекс задач, одной из которых является создание графических объектов. При этом на изучение этого языка программирования не придется тратить большое количество времени, поскольку он прост в освоении. В дальнейшем затраченное время окупится за счет ускорения решения рутинных задач, оставляя высвобожденное время на решение других задач.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое ActiveX Automation и VBA?
2. Для чего нужен VBA в AutoCAD?
3. Как загрузить среду программирования VBA?
4. Что такое модуль? Как вставить модуль в проект?
5. Что такое пользовательская форма? Как вставить пользовательскую форму в проект?
6. В чем отличие модуля от формы?

7. Как создать новый лист чертежа с помощью VBA?
8. Что такое свойство объекта?
9. Что такое метод объекта?
10. Какими двумя способами можно создать прямой отрезок?
11. Какой метод используется для создания отрезка?
12. Какие параметры необходимо задать для построения отрезка?
13. Как в программе объявляется графический объект (отрезок, окружность, дуга, полилиния)?
14. Как объявляются характерные точки объекта (начальные и конечные точки, центр)? Чем они являются как объекты программирования? Какой тип они имеют?
15. Как выглядит команда, создающая графические объекты (отрезок, окружность, дуга, полилиния)?
16. Какая команда показывает весь чертеж целиком?
17. Какой метод используется для создания окружности?
18. Какие параметры необходимы для построения окружности?
19. Какой метод используется для создания дуги?
20. Какие параметры необходимы для построения дуги?
21. Какая единица измерений у углов в среде программирования VBA?
22. Как перевести градусы в радианы и наоборот?
23. Чем отличается фигура, построенная с помощью отрезка, и фигура построенная, с помощью полилинии? В каких случаях следует использовать отрезок, а в каких – полилинию?
24. Какой метод используется для создания полилинии?
25. Какие параметры необходимы для построения полилинии?
26. Сколько элементов массива необходимо задать для создания отрезка с помощью полилинии?
27. Какой метод применяется для создания дугового сегмента в полилинии?
28. Какие параметры входят в метод для преобразования прямолинейного сегмента полилинии в дуговой? Объясните их назначение.

29. Как вычислить необходимую величину выпуклости дугового сегмента?
30. Какой индекс имеет первый сегмент полилинии?
31. На что влияют положительная и отрицательная величины выпуклости дугового сегмента?
32. Что собой представляет дуговой сегмент при величине выпуклости, равной 1; равной 0?
33. Для чего применяются типы линий? Что они собой представляют?
34. Каким объектам можно присваивать тип линии?
35. В каких файлах хранятся типы линий?
36. С помощью какого метода загружаются типы линий?
37. Какие параметры входят в метод для загрузки типа линии?
38. Какие свойства применяются для установки типа линии объектам? Чем они отличаются друг от друга?
39. Какое свойство и к каким объектам нужно применить, чтобы дальнейшие построения велись другим типом линии?
40. Какое свойство используется для изменения типа линии у существующего объекта?
41. Какое свойство предназначено для установки масштаба типа линии? Когда нужно использовать это свойство?
42. Чем отличается отрезок, к которому применили свойство масштаба типа линии, равный 1 и равный 0,5?
43. Что такое вес линии?
44. Какое свойство используется для установки веса линии?
45. Можно ли присвоить весу линии любое значение?

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Кравчик А.Э., Шлаф, М.М., Афонин, В.И., Соболенская Е.А.* Асинхронные двигатели серии 4А: Справочник. М.: Энергоиздат, 1982. 504 с.
2. *Дружинин Н.С., Чувиков Н.Т.* Черчение. – М.: Высш. шк., 1982. 224 с.
3. VBA: Создание простейших программ, создание отчетов в документе Word: Учеб. пособ. / *С.П. Минеев;* Самар. гос. техн. ун-т. Самара, 2004.
4. VBA: управление ходом программы, действия над массивами данных: Учеб. пособ. / *С.П. Минеев , П.В. Тулупов, Ю.А. Макаричев, Ю.А. Кальянова;* Самар. гос. техн. ун-т. Самара, 2005.
5. Справочная система AutoCAD 2005: Русская версия.

ОГЛАВЛЕНИЕ

Учебное издание

МИНЕЕВ Сергей Петрович

**Основы программирования в AutoCAD.
Технологии Activex Aautomation и VBA
в среде проектирования AutoCAD
для решения задач электромеханики**

Редактор С. И. К о с т е р и н а
Технический редактор В.Ф. Е л и с е е в а
Компьютерная верстка Е.Э. П а р с а д а н я н

Подп. в печать 29.10.15.
Формат 60x84 1/16. Бумага офсетная.
Печать офсетная.
Усл. п.л. 4,88. Усл. кр.-отт. 4,88. Уч.- изд.л. 4,79.
Тираж 100 экз. С. – 3 43.

Государственное образовательное учреждение
высшего профессионального образования
«Самарский государственный технический университет»
443100. г. Самара, ул. Молодогвардейская, 244. Главный корпус

Отпечатано в типографии
Самарского государственного технического университета
443100. г. Самара, ул. Молодогвардейская, 244. Корпус №8