С.П. МИНЕЕВ
Ю.В. ЗУБКОВ

АВТОМАТИЗАЦИЯ СОЗДАНИЯ ЧЕРТЕЖЕЙ ДЕТАЛЕЙ ЭЛЕКТРИЧЕСКИХ МАШИН НА ЯЗЫКЕ VBA В СРЕДЕ ПРОЕКТИРОВАНИЯ AUTOCAD

Учебное пособие

Самара
Самарский государственный технический университет
2012

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

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

Кафедра электромеханики
и автомобильного электрооборудования

С.П. МИНЕЕВ
Ю.В. ЗУБКОВ

АВТОМАТИЗАЦИЯ СОЗДАНИЯ ЧЕРТЕЖЕЙ ДЕТАЛЕЙ ЭЛЕКТРИЧЕСКИХ МАШИН НА ЯЗЫКЕ VBA В СРЕДЕ ПРОЕКТИРОВАНИЯ AUTOCAD

Учебное пособие

Самара
Самарский государственный технический университет
2012

Печатается по решению редакционно-издательского совета СамГТУ

УДК 621.313
М 61

Минеев С.П.

М 61 **Автоматизация создания чертежей деталей электрических машин на языке VBA в среде проектирования AutoCAD**: учеб. пособие / *С.П. Минеев*, *Ю.В. Зубков*. – Самара: Самар. гос. техн. ун-т, 2012. – 105 с.: ил.

Рассмотрены технологии автоматизированного проектирования деталей электрических машин в среде проектирования AutoCAD на языке программирования VBA. Рассмотрены методы и свойства объектов, с помощью которых происходит автоматизированное создание чертежей.

Предназначено для бакалавров по направлению 13.03.02 "Электроэнергетика и электротехника", по дисциплинам «Проектирование электрических машин», «Автоматизация расчетов электрических машин», «Алгоритмизация, программирование и компьютерная графика в задачах электромеханики», а также может быть полезно студентам родственных направлений подготовки.

Авторы благодарят *Ю.А. Кальянову* за помощь в разработке данного пособия.

УДК 621.313
М 61

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

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 (САПР 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 инженерных кадров.

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, ось координат подшипникового щита проходит вдоль оси вращения машины, а другая – по поверхности замка служит измерительной и сборочной базой подшипникового 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.

Данное учебное пособие предназначено для студентов, обучающихся по специальности «Электромеханика». Однако может быть полезно для студентов других специальностей, а также для инженеров-электромехаников в практической работе. Пособие предназначено для студентов, которые уже знакомы с основами программирования на VBA. Для тех, кто не знаком с языком VBA лучше изучить пособия указанные в библиографическом списке.

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

С.П. Минеев

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.

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 существует язык программирования AutoLISP, но программирование на AutoLISP не всегда просто. Команды достаточно сложны, встроенная отладка ограничена, и достаточно немного программистов знакомы с AutoLISP.

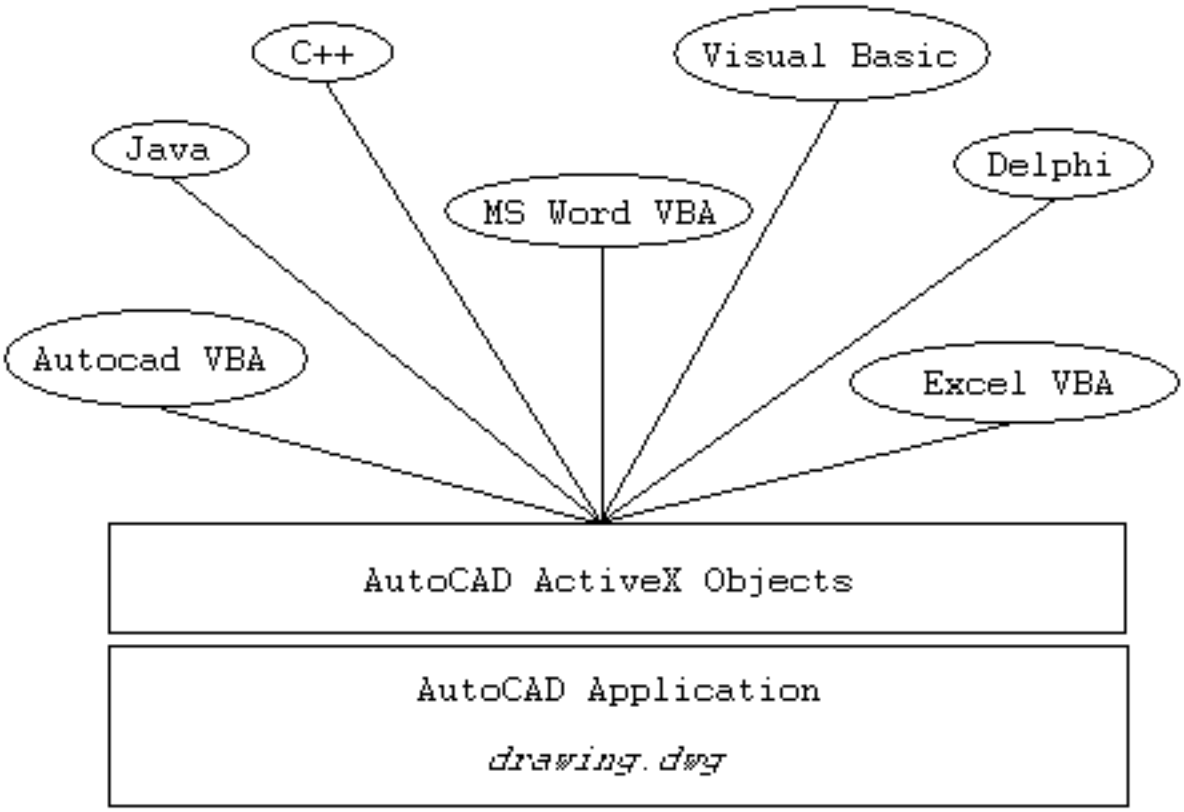


Рис. 1. Приложения с поддержкой ActiveX Automation

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. Так же доступен полный интерфейс Windows-программирования, так что программа может быть настолько сложна, насколько это возможно.

В данном учебном пособии будет рассмотрено автоматизированное проектирование деталей электрических машин с помощью языка

VBA. Построение объектов чертежа выполняется по точкам в двумерной системе координат. На примере построения чертежей листов якоря с различными пазами будут приведены расчётные данные характерных точек, формулы и тексты программ.

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

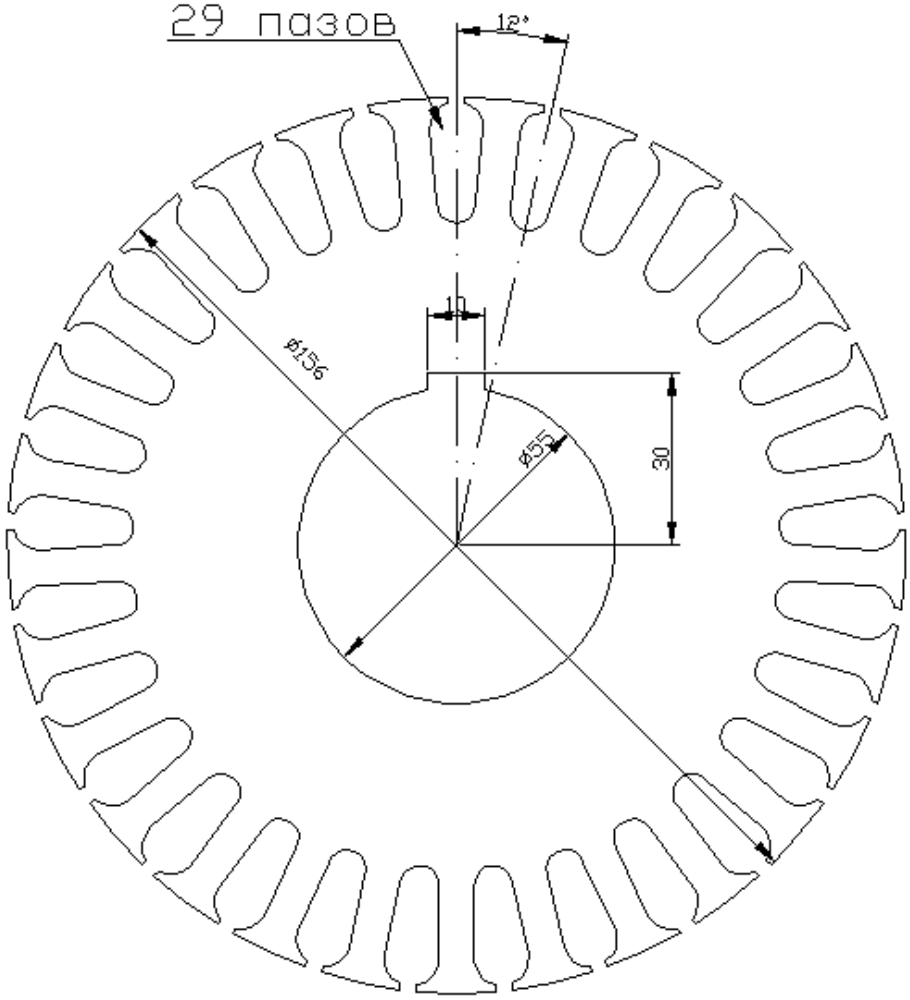
```
Sub Add()
 'новый документ чертежа
 Dim Listcherteja As AcadDocument
 'Создается новый документ
 Set Listcherteja = Documents.Add
End Sub
```

Кроме того, в учебном пособии следует обратить внимание на текст, выделенный **полужирным цветом**. Таким способом отмечены понятия, на которые нужно обратить особое внимание. К таким понятиям относятся определения, объекты, свойства, методы.

1. АВТОМАТИЗИРОВАННОЕ ПОСТРОЕНИЕ ЛИСТА ЯКОРЯ С ПАЗАМИ ОВАЛЬНОЙ ФОРМЫ

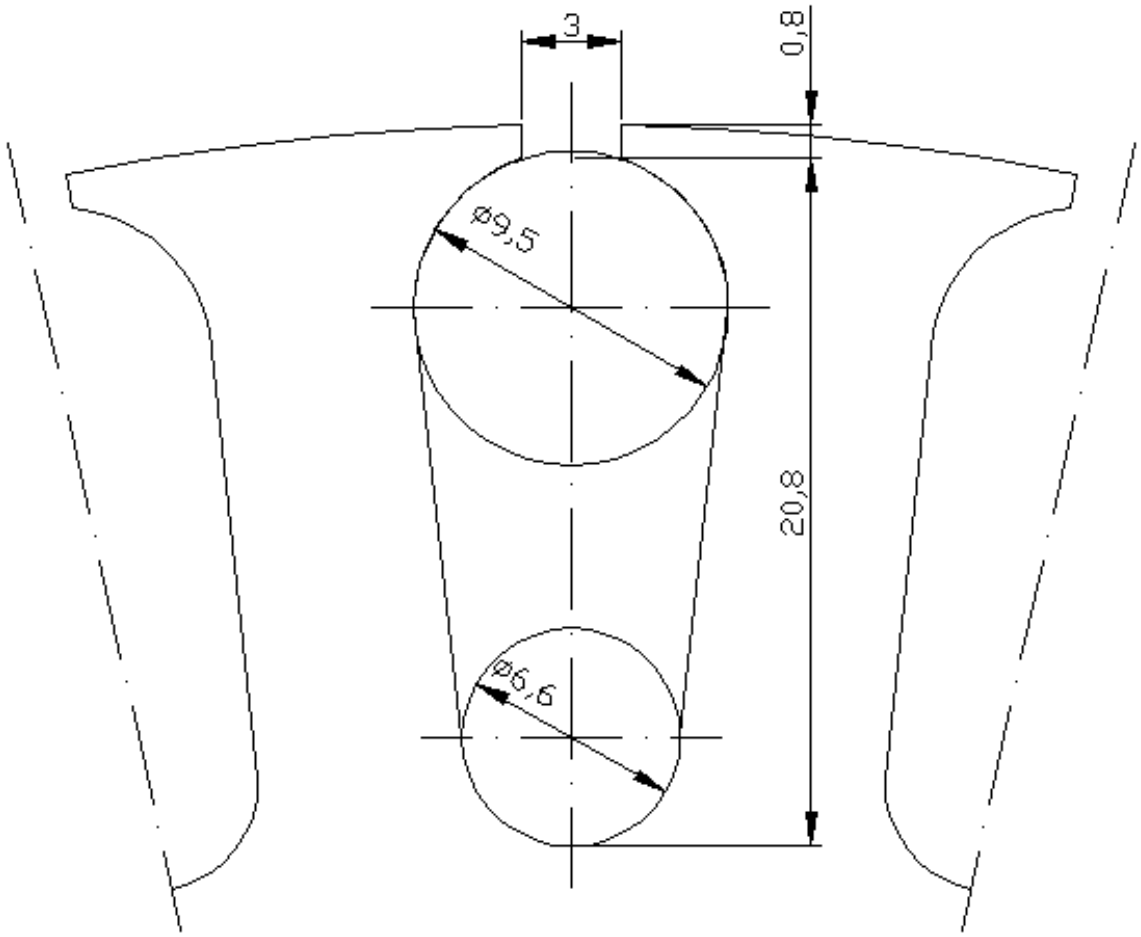
1.1. ПОСТАНОВКА ЗАДАЧИ И ИСХОДНЫЕ ДАННЫЕ ДЛЯ ПРОЕКТИРОВАНИЯ

Задача. Создать программу, которая будет вычерчивать лист якоря с пазами овальной формы, а также фрагмента пазовой части листа. Программа должна строить чертёж с различными исходными размерами. Программа должна строить чертеж в различных масштабах, согласно ГОСТ. Чертеж должен снабжаться простановкой размеров. На рис. 1.1 показан конечный вариант чертежа.



а

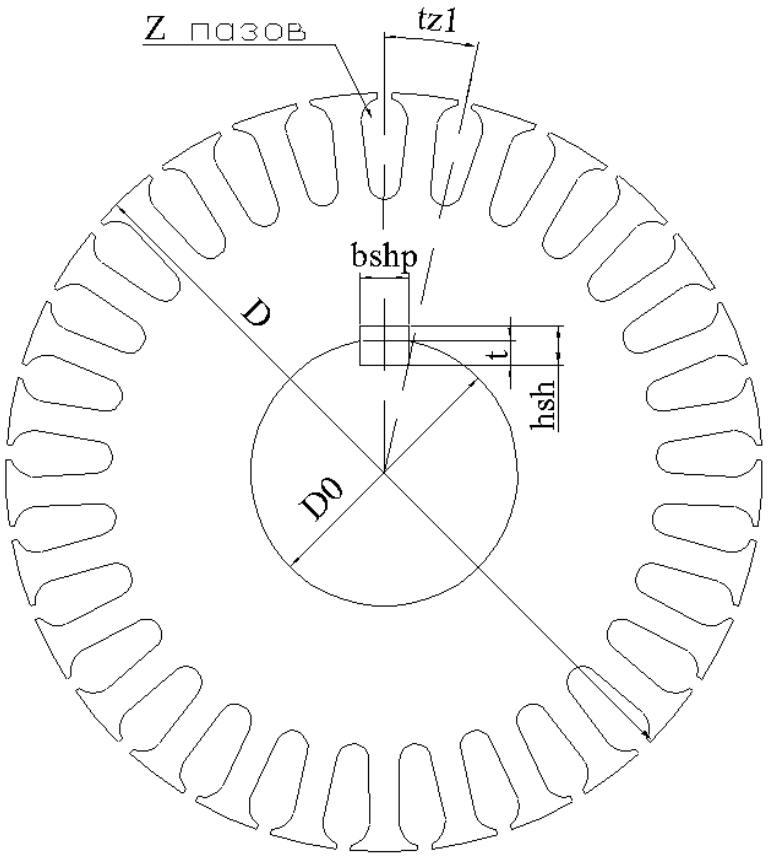
Рис. 1.1. Чертеж листа якоря:
а – лист якоря; *б* – пазовая часть листа в масштабе
(начало)



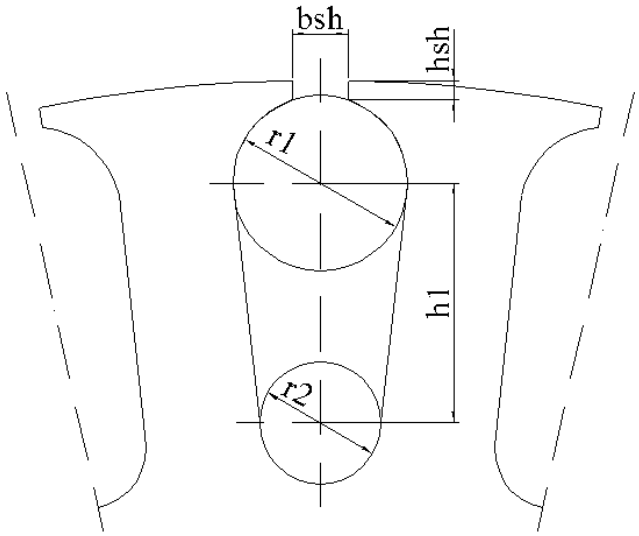
б

Рис. 1.1. Чертеж листа якоря:
а – лист якоря; *б* – пазовая часть листа в масштабе
(окончание)

Исходные данные для проектирования листа якоря берутся на основании электромагнитного расчёта электрической машины. На рис. 1.2 представлены размеры, на основании которых можно построить лист якоря. Размеры шпоночной канавки берутся на основании механического расчёта шпонки якоря. Шпоночной канавки может и не быть, если якорь насаживается на вал внатяг. В программе это также необходимо предусмотреть.



а



б

Рис. 1.2. Исходные размеры для проектирования:

а – размеры листа якоря; *б* – размеры пазовой части;
D – диаметр якоря; D0 – внутренний диаметр якоря; r1 – больший радиус паза;
r2 – меньший радиус паза; hsh – высота шлица паза; h1 – расстояние между центрами радиусов; bsh – ширина шлица паза; tz1 – зубцовое деление; Z – число пазов якоря;
hshp – высота шпонки; bshp – ширина шпонки; t – высота шпоночного канала

1.2. ПРОЕКТИРОВАНИЕ ФОРМЫ ДЛЯ ВВОДА ИСХОДНЫХ ВЕЛИЧИН

Для удобства использования программы создадим форму с элементами управления, в которые пользователь будет заносить исходные данные. На рис. 1.3 представлена такая форма. В нашем пособии мы не будем подробно останавливаться на том, как проектируется форма. Более подробную информацию о проектировании формы на языке VBA вы можете узнать в [2].

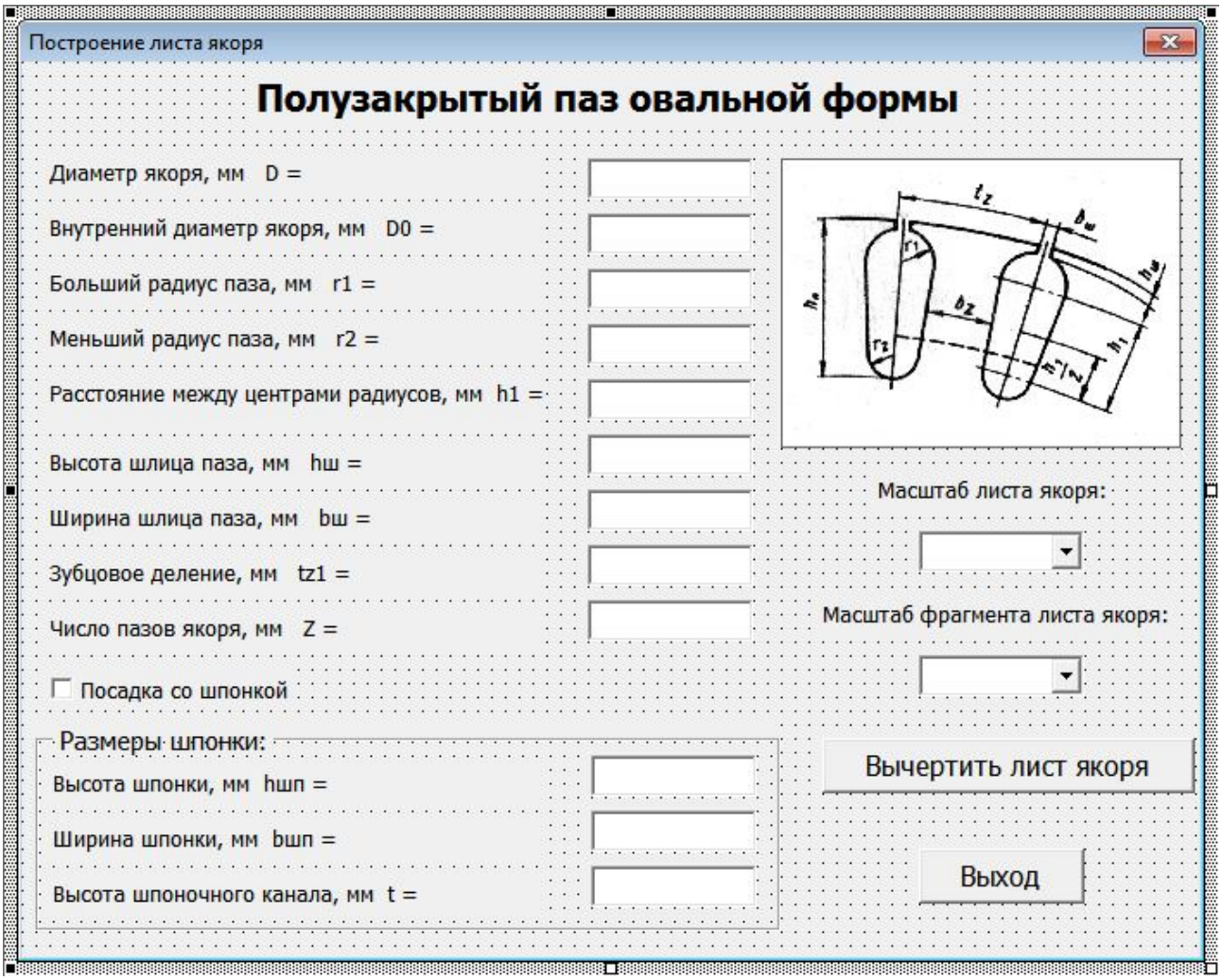


Рис. 1.3. Форма для ввода исходных величин

На форме размещены текстовые поля (TextBox) для ввода исходных данных, выпадающие списки (ComboBox) для выбора необходимого масштаба для листа якоря и пазовой части, кнопки (CommandButton) для запуска вычерчивания и выхода из программы. Так-

же есть флажок (CheckBox), с помощью которого пользователь выбирает, есть ли на листе статора шпоночная канавка.

В табл. 1.1 сведены названия объектов формы.

Таблица 1.1

Объекты формы

Название объекта	Класс объекта	Назначение
txtD.text	Текстовое поле	Диаметр якоря
txtD0.text	Текстовое поле	Внутренний диаметр якоря
txtR1.text	Текстовое поле	Больший радиус паза
txtR2.text	Текстовое поле	Меньший радиус паза
txtH1.text	Текстовое поле	Расстояние между центрами радиусов
txtHsh.text	Текстовое поле	Высота шлица паза
txtBsh.text	Текстовое поле	Ширина шлица паза
txtTz1.text	Текстовое поле	Зубцовое деление
txtZ.text	Текстовое поле	Число пазов якоря
txtHshp.text	Текстовое поле	Высота шпонки
txtBshp.text	Текстовое поле	Ширина шпонки
txtT.text	Текстовое поле	Высота шпоночного канала
chkShponka	Флажок	Наличие шпоночной канавки
cboMasshtabLista	Выпадающий список	Выбор масштаба листа якоря
cboMasshtabFragmenta	Выпадающий список	Выбор масштаба фрагмента пазовой части
cmdVichertit	Кнопка	Запуск вычерчивания
cmdExit	Кнопка	Выход из программы

Данные названия участвуют в тексте программы, поэтому им даны осмысленные названия. Объектам класса надпись (Label) и объект Image оставлены названия по умолчанию.

1.3. ПРОЦЕДУРА ЗАГРУЗКИ ФОРМЫ

Процедура обработки события **Загрузки формы** происходит, когда пользователь запускает программу. Ниже дан текст данной процедуры. В ней происходит автоматическое заполнение масштабов в выпадающих списках. А также некоторые другие операции. В комментариях к программе данные операции описываются. В тексте программы встречаются **переменные m** и **m2**. Переменная **m** предназна-

чена для хранения величины масштаба фрагмента листа якоря, а переменна **m2** – для хранения величины масштаба листа якоря. Также встречается переменная **NalichieShponki** – булева переменная, определяющая есть ли шпоночная канавка в листе якоря.

```
Private Sub UserForm_Initialize()

 'Если выпадающий список "Масштаб фрагмента листа якоря" не
заполнен
 If cboMasshtabFragmenta.text = "" Then

 'в текстовое поле масштаба фрагмента листа якоря добавляются
следующие масштабы:
 cboMasshtabFragmenta.AddItem "1 : 1"
 cboMasshtabFragmenta.AddItem "2 : 1"
 cboMasshtabFragmenta.AddItem "2,5 : 1"
 cboMasshtabFragmenta.AddItem "4 : 1"
 cboMasshtabFragmenta.AddItem "5 : 1"

 End If

 'Данная конструкция ветвления позволяет сохранить выбранный
пользователем масштаб фрагмента листа якоря,
 'при повторной активации формы:

 'Если масштаб фрагмента листа якоря равен 1 / 2.5
 If m = 1 Then
 'Активация строки "1 : 1" в выпадающем списке "Масштаба фрагмента
листа якоря"
 cboMasshtabFragmenta.text = cboMasshtabFragmenta.List(0)
 'Если масштаб фрагмента листа якоря равен 2
 ElseIf m = 2 Then
 'Активация строки "2 : 1" в выпадающем списке "Масштаба фрагмента
листа якоря"
 cboMasshtabFragmenta.text = cboMasshtabFragmenta.List(1)
 'Если масштаб фрагмента листа якоря равен 2.5
 ElseIf m = 2.5 Then
 'Активация строки "2,5 : 1" в выпадающем списке "Масштаба
фрагмента листа якоря"
 cboMasshtabFragmenta.text = cboMasshtabFragmenta.List(2)
 'Если масштаб фрагмента листа якоря равен 4
 ElseIf m = 4 Then
```

```
 'Активация строки "4 : 1" в выпадающем списке "Масштаба фрагмента
листа якоря"
 cboMasshtabFragmenta.text = cboMasshtabFragmenta.List(3)
 'Если масштаб фрагмента листа якоря равен 5
 ElseIf m = 5 Then
 'Активация строки "5 : 1" в выпадающем списке "Масштаба фрагмента
листа якоря"
 cboMasshtabFragmenta.text = cboMasshtabFragmenta.List(4)
 End If

 'Если выпадающий список "Масштаб листа якоря" не заполнен
 If cboMasshtabLista.text = "" Then

 'в текстовое поле масштаба листа якоря добавляются следующие
масштабы:
 cboMasshtabLista.AddItem "1 : 2,5"
 cboMasshtabLista.AddItem "1 : 2"
 cboMasshtabLista.AddItem "1 : 1"
 cboMasshtabLista.AddItem "2 : 1"
 cboMasshtabLista.AddItem "2,5 : 1"
 cboMasshtabLista.AddItem "4 : 1"
 cboMasshtabLista.AddItem "5 : 1"

 End If

 'Следующая конструкция ветвления позволяет сохранить выбранный
пользователем масштаб листа якоря,
 'при повторной активации формы:

 'Если масштаб листа якоря равен 1 / 2.5
 If m2 = 1 / 2.5 Then
 'Активация строки "1 : 2,5" в выпадающем списке "Масштаба листа
якоря"
 cboMasshtabLista.text = cboMasshtabLista.List(0)
 'Если масштаб листа якоря равен 1 / 2
 ElseIf m2 = 1 / 2 Then
 'Активация строки "1 : 2" в выпадающем списке "Масштаба листа
якоря"
 cboMasshtabLista.text = cboMasshtabLista.List(1)
 'Если масштаб листа якоря равен 1
 ElseIf m2 = 1 Then
 'Активация строки "1 : 1" в выпадающем списке "Масштаба листа
якоря"
 cboMasshtabLista.text = cboMasshtabLista.List(2)
```

```
 'Если масштаб листа якоря равен 2
 ElseIf m2 = 2 Then
 'Активация строки "2 : 1" в выпадающем списке "Масштаба листа
якоря"
 cboMasshtabLista.text = cboMasshtabLista.List(3)
 'Если масштаб листа якоря равен 2.5
 ElseIf m2 = 2.5 Then
 'Активация строки "2,5 : 1" в выпадающем списке "Масштаба листа
якоря"
 cboMasshtabLista.text = cboMasshtabLista.List(4)
 'Если масштаб листа якоря равен 4
 ElseIf m2 = 4 Then
 'Активация строки "4 : 1" в выпадающем списке "Масштаба листа
якоря"
 cboMasshtabLista.text = cboMasshtabLista.List(5)
 'Если масштаб листа якоря равен 5
 ElseIf m2 = 5 Then
 'Активация строки "5 : 1" в выпадающем списке "Масштаба листа
якоря"
 cboMasshtabLista.text = cboMasshtabLista.List(6)
 End If

 'Следующая конструкция ветвления позволяет сохранить наличие или
отсутсвие флажка "Посадка со шпокной"
 'при повторной активации формы:
 If NalichieShponki = True Then
 chkShponka.Value = True
 Else
 chkShponka.Value = False
 End If

End Sub
```

1.4. ПРОЦЕДУРА ОБРАБОТКИ СОБЫТИЯ, ИЗМЕНЕНИЯ ФЛАЖКА «ПОСАДКА СО ШПОНКОЙ», ПОЛЬЗОВАТЕЛЕМ

В данной процедуре определяется, какой вид посадки якоря на вал используется со шпонкой или в натяг.

```
Private Sub chkShponka_Click()

'Если не выбрана посадка со шпонкой
 If chkShponka.Value = False Then
```

```
 'рамка с размерами шпонки становится недоступной для выбора
пользователя
 fraShponka.Enabled = False
 'поле с надписью высота шпонки становится недоступным для выбора
пользователя
 Label12.Enabled = False
 'поле с надписью ширина шпонки становится недоступным для выбора
пользователя
 Label13.Enabled = False
 'поле с надписью высота шпоночного канала становится недоступным
для выбора пользователя
 Label14.Enabled = False
 'текстовое поле с веденной шириной шпонки становится недоступным
для выбора пользователя
 TxtBshp.Enabled = False
 'текстовое поле с веденной высотой шпонки становится недоступным
для выбора пользователя
 TxtHshp.Enabled = False
 'текстовое поле с веденной шпоночного канала становится
недоступным для выбора пользователя
 TxtT.Enabled = False
 NalichieShponki = False
 Else
 'Если выбрана посадка со шпонкой, то все поля остаются доступными
для выбора пользователя
 fraShponka.Enabled = True
 Label12.Enabled = True
 Label13.Enabled = True
 Label14.Enabled = True
 TxtBshp.Enabled = True
 TxtHshp.Enabled = True
 TxtT.Enabled = True
 NalichieShponki = True
 End If

End Sub
```

1.5. ВЫБОР МАСШТАБА ЛИСТА ЯКОРЯ И ЕГО ФРАГМЕНТА С ПОМОЩЬЮ ВЫПАДАЮЩЕГО СПИСКА

В следующих двух процедурах переменным m и m2 присваиваются значения масштаба для фрагмента листа якоря и самого листа якоря.

```
'Процедура обработки события выбора пользователем данных из
выподающего списка "Выбор масштаба фрагмента листа якоря"
Private Sub cboMasshtabFragmenta_Change()

 'Определение масштаба фрагмента листа якоря, выбранного
пользователем
 If cboMasshtabFragmenta.text = "1 : 1" Then m = 1
 If cboMasshtabFragmenta.text = "2 : 1" Then m = 2
 If cboMasshtabFragmenta.text = "2,5 : 1" Then m = 2.5
 If cboMasshtabFragmenta.text = "4 : 1" Then m = 4
 If cboMasshtabFragmenta.text = "5 : 1" Then m = 5

End Sub
'Процедура обработки события выбора пользователем данных из
выподающего списка "Выбор масштаба листа якоря"
Private Sub cboMasshtabLista_Change()

 'Определение масштаба листа якоря, выбранного пользователем
 If cboMasshtabLista.text = "1 : 2,5" Then m2 = 1 / 2.5
 If cboMasshtabLista.text = "1 : 2" Then m2 = 1 / 2
 If cboMasshtabLista.text = "1 : 1" Then m2 = 1
 If cboMasshtabLista.text = "2 : 1" Then m2 = 2
 If cboMasshtabLista.text = "2,5 : 1" Then m2 = 2.5
 If cboMasshtabLista.text = "4 : 1" Then m2 = 4
 If cboMasshtabLista.text = "5 : 1" Then m2 = 5

 Masshtab = cboMasshtabLista.text

End Sub
```

1.6. ПРОЦЕДУРА ОБРАБОТКИ СОБЫТИЯ НАЖАТИЯ КНОПКИ «ВЫЧЕРТИТЬ ЛИСТ ЯКОРЯ»

Стоит сразу отметить, что мы не будем в данной процедуре выполнять построение чертежа. Для этого будет использована процедура в отдельном модуле. А в данной процедуре будут проделаны подготовительные работы. Далее перечислены выполняемые виды работ.

Проверка на правильность ввода данных в текстовые поля. Проверяются, введены ли данные и введены ли именно числовые данные. Если данные введены не правильно, то выводятся соответствующие сообщения. Ниже показано, как происходит типичная проверка.

```
DD = IsNumeric(txtD.text)
If txtD.text = "" Then 'если не заполнено текстовое поле D
 'появляется сообщение
 MsgBox ("Не введены данные для переменной D")
 txtD.SetFocus 'устанавливаем фокус на текстовое поле D
ElseIf Not (DD) Then 'Если введено не число в текстовое поле D
 'появляется сообщение
 MsgBox ("Не правильно введены данные для переменной D")
 txtD.SetFocus
End If
```

Сначала определяем введено ли в текстовое поле числовые данные. Для этого предназначена функция IsNumeric. Данная функция возвращает значение True (Истина), если выражение в скобках содержит число, даже если оно выражено в тестовом формате. Функция возвращает значение False (Ложь), если выражение в скобках не содержит число.

В строке

```
If txtD.text = ""
```

с помощью конструкции ветвления (If… Then… ElseIf… EndIf) проверяем было ли что-либо введено в текстовое поле. Если никакого текста не было введено, то об этом выдается сообщение пользователю и делается активным поле, где данные не были введены:

```
MsgBox ("Не введены данные для переменной D")
txtD.SetFocus
```

Кроме того в данной конструкции ветвления в строке

```
ElseIf Not (DD)
```

проверяем было ли введено число. Если было введено не число, а какой-нибудь текст, то об этом выдается сообщение пользователю и делается активным поле, где данные были введены неправильно:

```
MsgBox ("Не правильно введены данные для переменной D")
txtD.SetFocus
```

Оператором

```
End If
```

закрывается конструкция ветвления.

Более подробную информацию о конструкциях ветвления и проверки правильности введенных данных смотрите в [3].

Если все данные пользователем были введены правильно, то происходит считывание данных из текстовых полей и присваивание их переменным. Выглядит это следующим образом:

```
D = CSng(txtD.text)
```

Функция **CSng** преобразует введенные данные из текстового формата текстового поля в числовой тип Single.

Назначаем расстояние от чертежа до его фрагмента. Для этого используется переменная k.

Определяем, выбрал пользователь посадку якоря со шпонкой или внатяг.

```
If chkShponka.Value = True Then
 'установка флага - шпонка чертится
 NalichieShponki = True

Else
 'снятие флага - шпонка не чертится
 NalichieShponki = False
End If
```

Запускаем процедуру **chertejovalpaz ()**, которая и выполняет построение чертежа.

Ниже представлен весь текст процедуры обработки события нажатия кнопки «Вычертить лист якоря».

```
'Процедура обработки события нажатия кнопки "Вычертить лист якоря"

Private Sub cmdVichertit_Click()
```

```
 'Объявляем переменные,которые показывают являются ли введенные в
текстовые поля данные,числами:

 'в текстовое поле диаметр якоря
 Dim DD As Boolean
 'в текстовое поле внутренний диаметр якоря
 Dim D0D0 As Boolean
 'в текстовое поле больший радиус паза
 Dim R1R1 As Boolean
 'в текстовое поле меньший радиус паза
 Dim R2R2 As Boolean
 'в текстовое поле высота шлица паза
 Dim HsHs As Boolean
 'в текстовое поле расстояние между центрами радиусов
 Dim H1H1 As Boolean
 'в текстовое поле ширина шлица паза
 Dim BsBs As Boolean
 'в текстовое поле зубцовое деление
 Dim TzTz As Boolean
 'в текстовое поле число пазов якоря
 Dim ZZ As Boolean
 'в текстовое поле высота шпонки
 Dim HshHsh As Boolean
 'в текстовое поле ширина шпонки
 Dim BshBsh As Boolean
 'в текстовое поле высота шпоночного канала
 Dim TT As Boolean

 pi = 3.141592
 radian = pi / 180


 'Выясняем являются ли введенные данные числами.

 DD = IsNumeric(txtD.text)
 D0D0 = IsNumeric(txtD0.text)
 R1R1 = IsNumeric(txtR1.text)
 R2R2 = IsNumeric(txtR2.text)
 HsHs = IsNumeric(txtHsh.text)
 H1H1 = IsNumeric(txtH1.text)
 BsBs = IsNumeric(txtBsh.text)
 TzTz = IsNumeric(txtTz1.text)
 ZZ = IsNumeric(txtZ.text)
 HshHsh = IsNumeric(TxtHshp.text)
```

```
 BshBsh = IsNumeric(TxtBshp.text)
 TT = IsNumeric(TxtT.text)


 If txtD.text = "" Then 'если не заполнено текстовое поле D
 'появляется сообщение
 MsgBox ("Не введены данные для переменной D")
 txtD.SetFocus 'устанавливаем фокус на текстовое поле D
 ElseIf Not (DD) Then 'Если введено не число в текстовое поле D
 'появляется сообщение
 MsgBox ("Не правильно введены данные для переменной D")
 txtD.SetFocus
 ElseIf txtD0.text = "" Then 'если не заполнено текстовое поле D0
 'появляется сообщение
 MsgBox ("Не введены данные для переменной D0")
 txtD0.SetFocus 'устанавливаем фокус на текстовое поле D0
 ElseIf Not (D0D0) Then 'Если введено не число в текстовое поле D0
 'появляется сообщение
 MsgBox ("Не правильно введены данные для переменной D0")
 txtD0.SetFocus
 ElseIf txtR1.text = "" Then 'если не заполнено текстовое поле r1
 'появляется сообщение
 MsgBox ("Не введены данные для переменной r1")
 txtR1.SetFocus 'устанавливаем фокус на текстовое поле r1
 ElseIf Not (R1R1) Then 'Если введено не число в текстовое поле r1
 'появляется сообщение
 MsgBox ("Не правильно введены данные для переменной r1")
 txtR1.SetFocus
 ElseIf txtR2.text = "" Then 'если не заполнено текстовое поле r2
 'появляется сообщение
 MsgBox ("Не введены данные для переменной r2")
 txtR2.SetFocus 'устанавливаем фокус на текстовое поле r2
 ElseIf Not (R2R2) Then 'Если введено не число в текстовое поле r2
 'появляется сообщение
 MsgBox ("Не правильно введены данные для переменной r2")
 txtR2.SetFocus
 ElseIf txtHsh.text = "" Then 'если не заполнено текстовое поле
hsh
 'появляется сообщение
 MsgBox ("Не введены данные для переменной hsh")
 txtHsh.SetFocus 'устанавливаем фокус на текстовое поле hsh
 ElseIf Not (HsHs) Then 'Если введено не число в текстовое поле
hsh
 'появляется сообщение
```

```
 MsgBox ("Не правильно введены данные для переменной hsh")
 txtHsh.SetFocus
 ElseIf txtH1.text = "" Then 'если не заполнено текстовое поле h1
 'появляется сообщение
 MsgBox ("Не введены данные для переменной h1")
 txtH1.SetFocus 'устанавливаем фокус на текстовое поле h1
 ElseIf Not (H1H1) Then 'Если введено не число в текстовое поле h1
 'появляется сообщение
 MsgBox ("Не правильно введены данные для переменной h1")
 txtH1.SetFocus
 ElseIf txtBsh.text = "" Then 'если не заполнено текстовое поле
bsh
 'появляется сообщение
 MsgBox ("Не введены данные для переменной bsh")
 txtBsh.SetFocus 'устанавливаем фокус на текстовое поле bsh
 ElseIf Not (BsBs) Then 'Если введено не число в текстовое поле
bsh
 'появляется сообщение
 MsgBox ("Не правильно введены данные для переменной bsh")
 txtBsh.SetFocus
 ElseIf txtTz1.text = "" Then 'если не заполнено текстовое поле
tz1
 'появляется сообщение
 MsgBox ("Не введены данные для переменной tz1")
 txtTz1.SetFocus 'устанавливаем фокус на текстовое поле tz1
 ElseIf Not (TzTz) Then 'Если введено не число в текстовое поле
tz1
 'появляется сообщение
 MsgBox ("Не правильно введены данные для переменной tz1")
 txtTz1.SetFocus
 ElseIf txtZ.text = "" Then 'если не заполнено текстовое поле Z
 'появляется сообщение
 MsgBox ("Не введены данные для переменной Z")
 txtZ.SetFocus 'устанавливаем фокус на текстовое поле Z
 ElseIf Not (ZZ) Then 'Если введено не число в текстовое поле Z
 'появляется сообщение
 MsgBox ("Не правильно введены данные для переменной Z")
 txtZ.SetFocus
 ElseIf TxtHshp.text = "" Then 'если не заполнено текстовое поле
hshp
 'появляется сообщение
 MsgBox ("Не введены данные для переменной hshp")
 TxtHshp.SetFocus 'устанавливаем фокус на текстовое поле hshp
```

```
 ElseIf Not (HshHsh) Then 'Если введено не число в текстовое поле
hshp
 'появляется сообщение
 MsgBox ("Не правильно введены данные для переменной hshp")
 TxtHshp.SetFocus
 ElseIf TxtBshp.text = "" Then 'если не заполнено текстовое поле
bshp
 'появляется сообщение
 MsgBox ("Не введены данные для переменной bshp")
 TxtBshp.SetFocus 'устанавливаем фокус на текстовое поле bshp
 ElseIf Not (BshBsh) Then 'Если введено не число в текстовое поле
bshp
 'появляется сообщение
 MsgBox ("Не правильно введены данные для переменной bshp")
 TxtBshp.SetFocus
 ElseIf TxtT.text = "" Then 'если не заполнено текстовое поле t
 'появляется сообщение
 MsgBox ("Не введены данные для переменной t")
 TxtT.SetFocus 'устанавливаем фокус на текстовое поле t
 ElseIf Not (TT) Then 'Если введено не число в текстовое поле t
 'появляется сообщение
 MsgBox ("Не правильно введены данные для переменной t")
 TxtT.SetFocus
 Else

 'переводим данные в числовую форму
 D = CSng(txtD.text)
 D0 = CSng(txtD0.text)
 r1 = CSng(txtR1.text)
 r2 = CSng(txtR2.text)
 hsh = CSng(txtHsh.text)
 h1 = CSng(txtH1.text)
 bsh = CSng(txtBsh.text)
 tz1 = CSng(txtTz1.text)
 Z = CSng(txtZ.text)
 hshp = CSng(TxtHshp.text)
 bshp = CSng(TxtBshp.text)
 t = CSng(TxtT.text)

 'Допущения для того, чтобы ось фрагмента листа якоря выступала с
каждой стороны на 5 мм
 'k - расстояние от листа якоря до фрагмента
 'Если масштаб фрагмента листа якоря = 1 / 2
 If m = 1 / 2 Then
```

```
 k = 42.5
 'Если масштаб фрагмента листа якоря = 1 / 2.5
 ElseIf m = 1 / 2.5 Then
 k = 43
 'Если масштаб фрагмента листа якоря = 1
 ElseIf m = 1 Then
 k = 40
 'Если масштаб фрагмента листа якоря = 2
 ElseIf m = 2 Then
 k = 35
 'Если масштаб фрагмента листа якоря = 2.5
 ElseIf m = 2.5 Then
 k = 32.5
 'Если масштаб фрагмента листа якоря = 4
 ElseIf m = 4 Then
 k = 25
 'Если масштаб фрагмента листа якоря = 5
 ElseIf m = 5 Then
 k = 20
 End If

 'Если пользователь выбрал посадку со шпонкой
 If chkShponka.Value = True Then
 'установка флага - шпонка чертится
 NalichieShponki = True

 Else
 'снятие флага - шпонка не чертится
 NalichieShponki = False
 End If

 chertejovalpaz

 End If
End Sub
```

1.7. ПРОЦЕДУРА ОБРАБОТКИ СОБЫТИЯ НАЖАТИЯ КНОПКИ «ВЫХОД»

При нажатии на кнопку выход выполняется следующая процедура:

```
Private Sub cmdExit_Click()
 End
End Sub
```

Оператор End завершает работу программы.

1.8. ХАРАКТЕРНЫЕ ТОЧКИ ПРИ ПОСТРОЕНИИ ЛИСТА ЯКОРЯ И ЕГО ФРАГМЕНТА

Построение объектов чертежа выполняется по точкам в двумерной системе координат. В программе координаты точек определяются относительно трехмерной системы координат (X, Y, Z). Первой определяется координата X, второй – Y, третей – Z, которая во всех случаях равна 0. Наличие координаты Z – обязательное требование среды программирования AutoCad.

За начало системы координат принимаем центр листа якоря. Относительно этой точки производятся все построения, которые выполняются пошагово.

В описании построения объектов выполняется следующий порядок:

1. Определение координат и параметров (радиусы, углы) объектов – вывод формул.

2. Преобразование выведенных формул к стандарту VBA.

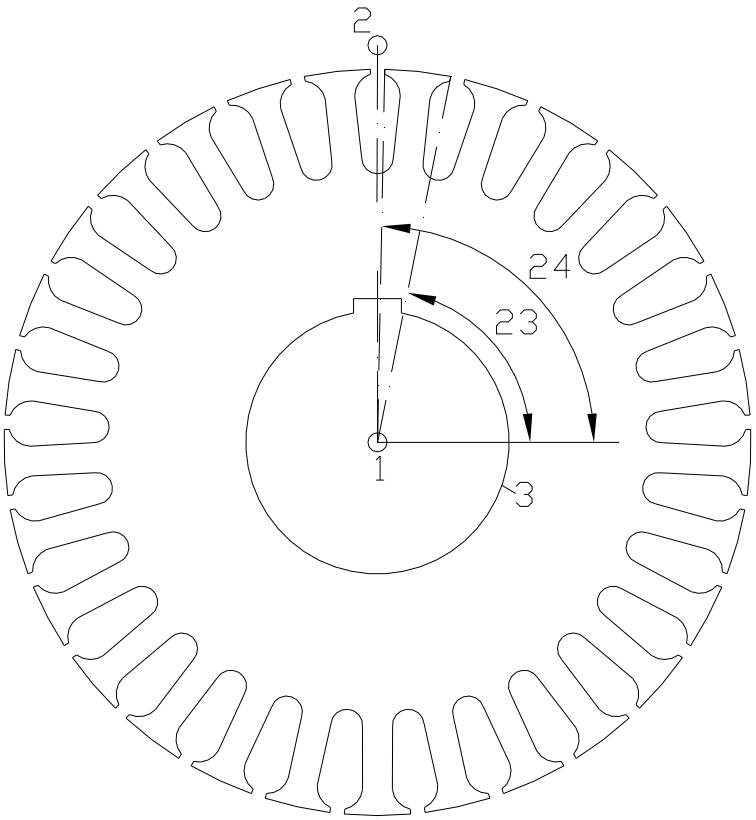
3. Указание используемой команды для построения объекта.

На рис. 1.4 представлены точки, по которым строится лист якоря и его фрагмент. Стоит отметить, что построение будет вестись только для половины паза. Вторая половина паза будет построена зеркальным отображением первой половины.

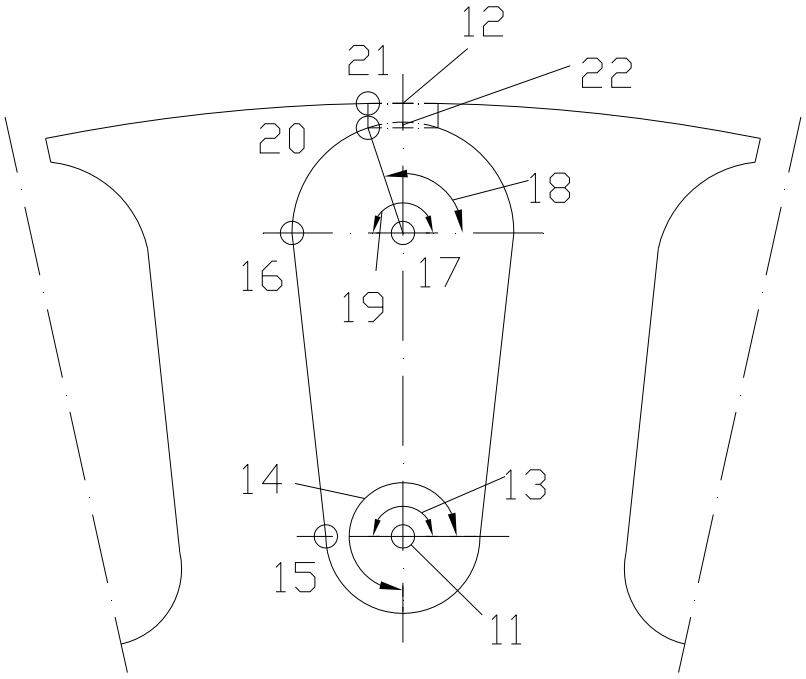
Далее все построения ведутся в процедуре **chertejovalpaz**, расположенной в модуле **Postroeniechertejya**. Текст программы размещается между строками начала и конца процедуры:

```
Sub chertejovalpaz()

End Sub
```



а

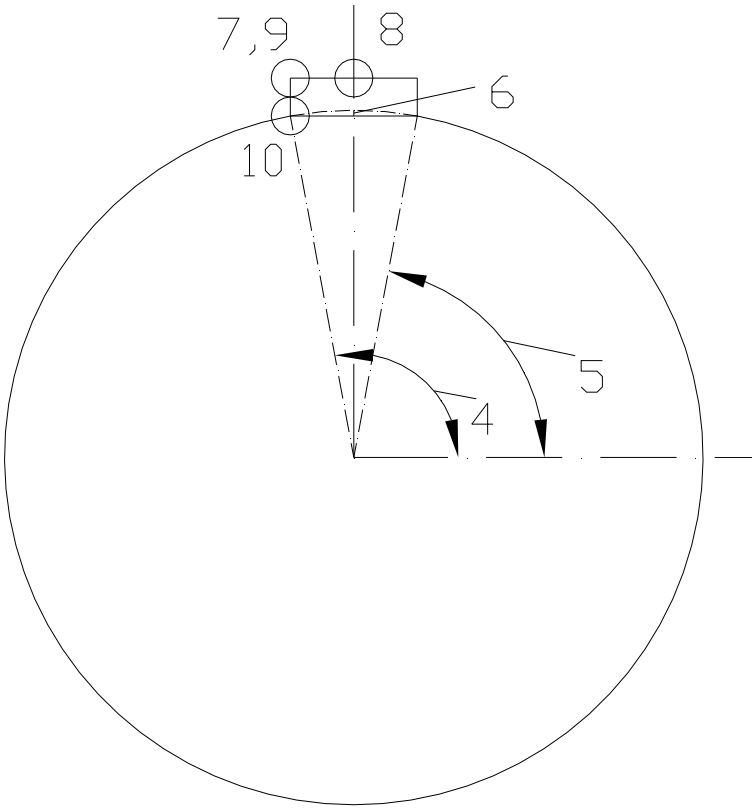


б

Рис. 1.4. Координаты точек, необходимые для построения листа якоря с пазами овальной формы:

а – лист якоря; *б* – фрагмент листа якоря; *в* – отверстие под вал

(начало)



в

Рис. 1.4. Координаты точек, необходимые для построения листа якоря с пазами овальной формы:

а – лист якоря; *б* – фрагмент листа якоря; *в* – отверстие под вал

(окончание)

1.9. СОЗДАНИЕ НОВОГО ЧЕРТЕЖА

Новый документ чертежа создаётся для того, чтобы программа случайно не создала чертёж в уже ранее выполненных чертежах. Новый документ чертежа создается с помощь метода **Add**.

```
'Создается новый документ
Set Listcherteja = Documents.Add
```

1.10. ЗАГРУЗКА ТИПОВ ЛИНИЙ ДЛЯ ОСЕВЫХ ЛИНИЙ

Так как в чертеже присутствуют осевые линии, то их необходимо сначала загрузить. Следующий фрагмент процедуры предназначен именно для этого.

```
'Загрузка типа линий для осевых линий.

'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 "acad.lin"

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

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**.

Более подробно о типах линий и работе с ними можно узнать в [1], [4].

1.11. ПОСТРОЕНИЕ ОСЕВОЙ ЛИНИИ ПАЗА (рис. 1.4, *а*, т.т. 1-2)

Отрезок строится с помощью метода **AddLine**. Предварительно нужно переключиться на режим построения осевых линий. После по-

строения с помощью свойства Lineweight устанавливаем толщину линии.

Начальные и конечные координаты точек для осевой линии:

$t1(0)=0:t1(1)=0:t1(2)=0$;

$t2(0)=0:t2(1)=\frac{D}{2}+5:\ t2(2)=0$.

На языке VBA:

```
'Активным типом линии становится "ACAD_ISO10W100" и построения
будут вестись с данным типом линии ("осевая линия")
ThisDrawing.ActiveLinetype =
ThisDrawing.Linetypes.Item("ACAD_ISO10W100")
'Осевая линия паза
Dim lineOs As AcadLine
'Начальная координата осевой линии паза
Dim t1(0 To 2) As Double
'Конечная координата осевой линии паза
Dim t2(0 To 2) As Double

'Определяем начальную и конечную координаты точек для осевой линии
 t1(0) = 0#: t1(1) = 0#: t1(2) = 0#
 t2(0) = 0#: t2(1) = D / 2 + 5#: t2(2) = 0#

'Строится осевая линия паза
Set lineOs = ThisDrawing.ModelSpace.AddLine(t1, t2)
'толщина линии 0,5 мм
lineOs.Lineweight = acLnWt050
```

В данном фрагменте программы используются некоторые методы и свойства, которые стоит описать чуть подробнее:

1. 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 начинают вести тем типом линии, который указан в данном свойстве. Если нужно изменить тип линии у уже существующего объекта, используют свойство **Linetype**.

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.

2. AddLine – метод, предназначенный для построения отрезка. В общем виде команда построения отрезка выглядит следующим образом:

```
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**, **PaperSpace 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**.

3. Lineweight – свойство, определяющее вес (толщину линии) для указанного нарисованного объекта либо используемое для установки веса линии по умолчанию, после чего все построения будут вестись линиями с указанным весом. 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.

Линия весом величиной 0 представляет собой наиболее тонкую линию, которую позволяет печатать указанное печатающее устройство. В пространстве модели показывается вес линии в 1 пиксель.

Более подробно об описанных объектах, свойствах и методах можно узнать в [1], [4].

1.12. ПОСТРОЕНИЕ ОТВЕРСТИЯ ПОД ВАЛ (рис. 1.4, *а*, т. 3)

Отверстие под вал может иметь два вида в зависимости от того, какая посадка якоря под вал предусмотрена – со шпонкой или без неё.

1. При наличии шпонки под вал.

Построение ведется с помощью дуги и отрезков.

В общем виде команда построения дуги выглядит следующим образом:

```
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.

Согласно правилам построения дуги определяем параметры дуговой части отверстия под вал.

Центр дуги (т1):

$$cD0(0)=0:cD0(1)=0:cD0(2)=0.$$

Сокращение «с» в формуле означает центр.

Радиус дуги:

$$rD0=\left(\frac{D0}{2}\right).$$

Сокращение «r» в формуле означает радиус.

Начальный и конечный углы дуги D0 в радианах (рис. 1.4, *в*, т. 4, т. 5):

$$ugnD0=\frac{\pi}{2}+\text{arctg}\left(\frac{\frac{bshp}{2}}{\frac{D0}{2}-hxr1}\right);$$

$$ugkD0=\frac{\pi}{2}-\text{arctg}\left(\frac{\frac{bshp}{2}}{\frac{D0}{2}-hxr1}\right),$$

где $hxr1$ – высота сегмента, ограниченного дугой диаметра вала и хордой, образованной шириной шпонки (рис. 1.4, *в*, т. 6):

$$hxr1=\frac{D0}{2}-\sqrt{\left(\frac{D0}{2}\right)^2-\left(\frac{bshp}{2}\right)^2}.$$

Сокращения *ugn* означает «угол начальный», *ugk* – «угол конечный».

Половина ширины шпонки (рис. 2.1, *а*, т.т. 7-8).

Начальные и конечные координаты точек для линии ширины половины шпонки:

$$t7(0)=-\frac{bshp}{2};\ t7(1)=\frac{D0}{2}+(hshp-t-hxr1);\ t7(2)=0;$$

$$t8(0)=0;\ t8(1)=\frac{D0}{2}+(hshp-t-hxr1);\ t8(2)=0.$$

Линия высоты шпонки (рис. 1.4, *в*, т.т. 9-10).

Начальные и конечные координаты точек для линии высоты половины шпонки:

$$t9(0) = -\frac{bshp}{2};\ t9(1) = \frac{D0}{2} + (hshp - t - hxr1);\ t9(2) = 0;$$

$$t10(0) = -\frac{bshp}{2};\ t10(1) = \frac{D0}{2} - hxr1;\ t10(2) = 0.$$

2. При отсутствии шпонки под вал.

Построение ведется с помощью окружности.

Чтобы создать окружность используется метод **AddCircle**.

В общем виде команда построения окружности выглядит следующим образом:

```
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.

Согласно правилам построения окружности определяем параметры отверстия под вал.

Координаты точки центра окружности диаметра вала (рис. 1.4, *а*, т. 1):

$$cD0(0) = 0 : cD0(1) = 0 : cD0(2) = 0.$$

Радиус дуги:

$$rD0 = \left(\frac{D0}{2}\right).$$

На VBA фрагмент построения отверстия под вал будет выглядеть следующим образом:

```
 'Построение ОКРУЖНОСТИ и ДУГИ диаметра вала

 'Центр окружности(дуги) диаметра вала
 Dim cpD0D0(0 To 2) As Double
 'Радиус окружности(дуги) диаметра вала
 Dim radiusD0 As Double

 cpD0(0) = 0#: cpD0(1) = 0: cpD0(2) = 0#
 rD0 = D0 / 2

 'Если стоит флаг - шпонка чертится, тогда строится ДУГА диаметра
вала,
 'половина ширины шпонки и ее высота
 If NalichieShponki = True Then

 'Дуга диаметра вала
 Dim arcObjD0 As AcadArc
 'Начальный угол дуги диаметра вала
 Dim ugnD0 As Double
 'Конечный угол дуги диаметра вала
 Dim ugkD0 As Double

 ugnD0 = pi / 2 + Atn(bshp / 2 / (D0 / 2 - hxr1))
 ugkD0 = pi / 2 - Atn(bshp / 2 / (D0 / 2 - hxr1))

 'Строится дуга диаметра вала
 Set arcObjD0 = ThisDrawing.ModelSpace.AddArc(cpD0, rD0, ugnD0,
ugkD0)
 'толщина линии 1 мм
 arcObjD0.Lineweight = acLnWt100

 'Построение линии половины ширины шпонки

 'Половина ширины шпонки
 Dim lineObjbshp As AcadLine
 'Начальная координата линии половины ширины шпонки
 Dim t7(0 To 2) As Double
 'Конечная координата линии половины ширины шпонки
 Dim t8(0 To 2) As Double
```

```
 ' Определяем начальную и конечную координаты точек для линии
половины ширины шпонки
 t7(0) = -bshp / 2: t7(1) = D0 / 2 + (hshp - t - hxr1): t7(2) = 0#
 t8(0) = 0: t8(1) = D0 / 2 + (hshp - t - hxr1): t8(2) = 0#

 'Строится линия половины ширины шпонки
 Set lineObjbshp = ThisDrawing.ModelSpace.AddLine(t7, t8)
 'толщина линии 1 мм
 lineObjbshp.Lineweight = acLnWt100

 'Построение линии высоты шпонки
 Dim lineObjhshp As AcadLine
 'Начальная координата линии высоты шпонки
 Dim t9(0 To 2) As Double
 'Конечная координата линии высоты шпонки
 Dim t10(0 To 2) As Double

 ' Определяем начальную и конечную координаты точек для линии
высоты шпонки
 t9(0) = -bshp / 2: t9(1) = D0 / 2 + (hshp - t - hxr1): t9(2) = 0#
 t10(0) = -bshp / 2: t10(1) = D0 / 2 - hxr1: t10(2) = 0#

 'Строится линия высоты шпонки
 Set lineObjhshp = ThisDrawing.ModelSpace.AddLine(t9, t10)
 'толщина линии 1 мм
 lineObjhshp.Lineweight = acLnWt100

 Else
 'Если не стоит флаг - шпонка чертится, тогда строится ОКРУЖНОСТЬ
диаметра вала

 'Окружность диаметра вала
 Dim circleD0 As AcadCircle
  'Строится окружность диаметра вала
 Set circleD0 = ThisDrawing.ModelSpace.AddCircle(centerPointD0,
radiusD0)
 'толщина линии 1 мм
 circleD0.Lineweight = acLnWt100

 End If
```

Более подробно о построении дуг и окружностей можно узнать в [1], [4].

1.13. ПОСТРОЕНИЕ ДУГИ РАДИУСА r2

Координаты центральной точки окружности r2 (рис. 1.4, *б*, т. 11):

$$cr2(0) = 0 : cr2(1) = \frac{D}{2} - hxr - hsh - r1 - h1 : \ \ cr2(2) = 0,$$

где hxr – высота сегмента, ограниченного дугой диаметра листа якоря и хордой, образованной шириной шлица (рис. 1.4, *б*, т. 12):

$$hxr = \frac{D}{2} - \sqrt{\left(\frac{D}{2}\right)^2 - \left(\frac{bsh}{2}\right)^2}.$$

Центр дуги:

$$rr2 = r2.$$

Начальный и конечный углы дуги r2 в радианах (рис. 1.4, *б*, т. 13, т. 14):

$$ugnr2 = \pi;$$

$$ugkr2 = \frac{3 \cdot \pi}{2}.$$

На VBA фрагмент построение будет выглядеть следующим образом:

```
'Построение r2
 'Дуга радиуса r2
Dim arcObjr2 As AcadArc
'Центр дуги радиуса r2
Dim cr2(0 To 2) As Double
'Радиус дуги r2
Dim rr2 As Double
'Начальный угол дуги радиуса r2
Dim ugnr2 As Double
'Конечный угол дуги радиуса r2
Dim ugkr2 As Double
```

```
cr2(0) = 0#: cr2(1) = D / 2 - hxr - hsh - r1 - h1: cr2(2) = 0#
rr2 = r2
ugnr2 = pi#
ugknr2 = 3 * pi / 2

'Строится дуга радиуса r2
Set arcObjr2 = ThisDrawing.ModelSpace.AddArc(cr2, rr2, ugnr2,
ugkr2)
'толщина линии 1 мм
arcObjr2.Lineweight = acLnWt100
```

1.14. ПОСТРОЕНИЕ ЛИНИИ, СОЕДИНЯЮЩЕЙ ДУГИ БОЛЬШЕГО И МЕНЬШЕГО РАДИУСА (БОКОВАЯ СТОРОНА ПАЗА) (рис. 1.4, *б*, т.т. 15-16)

Начальные и конечные координаты точек для линии, соединяющей r1 с r2:

$$t15(0) = -r2 : t15(1) = \frac{D}{2} - hxr - hsh - r1 - h1 : t15(2) = 0;$$

$$t16(0) = -r1 : t16(1) = \frac{D}{2} - hxr - hsh - r1 : t16(2) = 0.$$

На языке VBA:

```
'Построение линии, соединяющей r1 с r2

'Линия, соединяющая r1 с r2 (боковая стенка паза)
Dim lineObjr1_r2 As AcadLine
'Начальная координата линии, соединяющей r1 с r2
Dim t15(0 To 2) As Double
'Конечная координата линии, соединяющей r1 с r2
Dim t16(0 To 2) As Double
```

```
 'Определяем начальную и конечную координаты точек для линии,
соединяющей r1 с r2
 t15(0) = -r2: t15(1) = D / 2 - hxr - hsh - r1 - h1: t15(2) = 0
 t16(0) = -r1: t16(1) = D / 2 - hxr - hsh - r1: t16(2) = 0

 'Строится линия, соединяющая r1 с r2
 Set lineObjr1_r2 = ThisDrawing.ModelSpace.AddLine(t15, t16)
 'толщина линии 1 мм
 lineObjr1_r2.Lineweight = acLnWt100
```

1.15. ПОСТРОЕНИЕ ДУГИ РАДИУСА r1

Координаты центральной точки окружности r1 (рис. 1.4, *б*, т. 17):

$$c17(0) = 0 : c17(1) = \frac{D}{2} - hxr - hsh - r1 : \quad c17(2) = 0.$$

Радиус дуги: r1

Начальный и конечный углы дуги r1 в радианах (рис. 1.4, *б*, т. 18, т. 19):

$$ug18 = \text{Arctg}\left(\frac{\frac{bsh}{2}}{\sqrt{1 - \left(\frac{\frac{bsh}{2}}{r1} \right)^2}} \right) + \frac{\pi}{2};$$

$$ug19 = \pi.$$

На языке VBA:

```
'Построение r1

'Дуга радиуса r1
Dim arcObjr1 As AcadArc
'Центр дуги радиуса r1
Dim c17(0 To 2) As Double
'Радиус дуги r1
Dim rr1 As Double
```

```
'Начальный угол дуги радиуса r1, приведенный в радианах
Dim ug18 As Double
Dim ug19 As Double

c17(0) = 0: c17(1) = D / 2 - hxr - hsh - r1: c17(2) = 0#
rr1 = r1

ug18 = Atn((bsh / 2) / r1 / (Sqr(1 - ((bsh / 2) / r1) ^ 2))) + pi / 2
ug19 = pi

'Строится дуга радиуса r1
Set arcObjr1 = ThisDrawing.ModelSpace.AddArc(c17, rr1, ug18, ug19)
'толщина линии 1 мм
arcObjr1.Lineweight = acLnWt100
```

1.16. ЛИНИЯ ВЫСОТЫ ШЛИЦА (рис. 1.4, *б*, т.т. 20-21)

Начальные и конечные координаты точек для линии высоты шлица:

$$t20(0) = -\frac{bsh}{2} : t20(1) = \frac{D}{2} - hxr - hsh - hxr2 : t20(2) = 0;$$

$$t21(0) = -\frac{bsh}{2} : t21(1) = \frac{D}{2} - hxr : t21(2) = 0,$$

где $hxr2$ – высота сегмента, ограниченного дугой большего радиуса паза и хордой, образованной шириной шлица (рис. 1.4, *б*, т. 22):

$$hxr2 = r1 - \sqrt{r1^2 - \left(\frac{bsh}{2}\right)^2}.$$

На языке VBA:

```
'Строим линию высоты шлица

'Линия высоты шлица
Dim lineObjhsh As AcadLine
'Начальная координата линии высоты шлица
Dim t20(0 To 2) As Double
'Конечная координата линии высоты шлица
Dim t21(0 To 2) As Double
```

```
hxr2 = r1 - Sqr(r1 ^ 2 - (bsh / 2) ^ 2)

'Определяем начальную и конечную координаты точек для линии высоты
шлица
t20(0) = -bsh / 2: t20(1) = D / 2 - hxr - hsh - hxr2: t20(2) = 0
t21(0) = -bsh / 2: t21(1) = D / 2 - hxr: t21(2) = 0

'Строится линия высоты шлица
Set lineObjhsh = ThisDrawing.ModelSpace.AddLine(t20, t21)
'толщина линии 1 мм
lineObjhsh.Lineweight = acLnWt100
```

1.17. ЗЕРКАЛЬНОЕ ОТОБРАЖЕНИЕ ЧАСТИ ПАЗА

После вычерчивания части паза, необходимо зеркально отобразить уже выполненные части чертежа относительно осевой линии паза (т.т. 1-2). Зеркальному отображению подвергаются все элементы паза, а также при наличии шпоночной канавки и сама канавка. Для этого воспользуемся методом Mirror.

Команда зеркального отображения в общем виде:

```
RetVal = object.Mirror(Point1, Point2) ,
```

где **RetVal** – название объекта создаваемой зеркальной копии существующего объекта;

Object – название объекта, с которого создается зеркальная копия, им может быть любой объект рисования;

Point1, **Point2** – координаты первой и второй точек оси, относительно которой осуществляется зеркальное отображение. Представляют собой одномерный массив, состоящий из трех элементов, имеет тип **Double**;

Mirror – метод, который осуществляет зеркальное отображение, относительно оси, образуемой точками **Point1**, **Point2**.

На языке VBA:

```
'Зеркально отображаем половину паза и половину шпонки относительно
оси паза

'Объявляем зеркальные копии объектов
'Зеркальное отображение дуги радиуса r2
Dim mirrarcObjr2 As AcadArc
'Зеркальное отображение линии, соединяющей r1 с r2
Dim mirrlineObjr1_r2 As AcadLine
'Зеркальное отображение дуги радиуса r1
Dim mirrarcObjr1 As AcadArc
'Зеркальное отображение линии высоты шлица
Dim mirrlineObjhsh As AcadLine
'Зеркальное отображение линии половины ширины шпонки
Dim mirrlineObjbshp As AcadLine
'Зеркальное отображение линии высоты шпонки
Dim mirrlineObjhshp As AcadLine

'Зеркально отображаем нужные объекты
Set mirrarcObjr2 = arcObjr2.Mirror(t1, t2)
Set mirrlineObjr1_r2 = lineObjr1_r2.Mirror(t1, t2)
Set mirrarcObjr1 = arcObjr1.Mirror(t1, t2)
Set mirrlineObjhsh = lineObjhsh.Mirror(t1, t2)

'Если стоит флаг - шпонка чертится, тогда зеркально отображается
линия половины ширины шпонки и линия ее высоты
If NalichieShponki = True Then
 Set mirrlineObjbshp = lineObjbshp.Mirror(t1, t2)
 Set mirrlineObjhshp = lineObjhshp.Mirror(t1, t2)
End If
```

1.18. ДУГА НАКОНЕЧНИКА ЗУБЦА

Цент дуги:

$$\mathrm{clz}(0)=0:\mathrm{clz}(1)=0:\mathrm{clz}(2)=0.$$

Радиус дуги:

$$rlz=\frac{D}{2}.$$

Начальный и конечный углы дуги наконечника зубца в радианах (рис. 1.4, *а*, т. 23, т. 24):

$$ug23 = \frac{\pi}{2} - \left(\frac{2 \cdot \pi}{Z} - \text{arctg}\left(\frac{bsh/2}{D/2 - hxr} \right) \right);$$

$$ug24 = \frac{\pi}{2} - \text{arctg}\left(\frac{bsh/2}{D/2 - hxr} \right).$$

На языке VBA:

```
'Построение наконечника зубца
'Объявляем объекты и параметры построения наконечника зубца
'Дуга наконечника зубца
Dim arcObjlz As AcadArc
'Центр дуги наконечника зубца
Dim clz(0 To 2) As Double
'Радиус дуги наконечника зубца
Dim rlz As Double
'Начальный угол дуги наконечника зубца в радианах
Dim ug23 As Double
'Конечный угол дуги наконечника зубца в радианах
Dim ug24 As Double
clz(0) = 0#: clz(1) = 0#: clz(2) = 0#
rlz = D / 2
'Длина дуги, соответствующая ширине шлица - bsh
lbsh = D / 2 * 2 * Atn((bsh / 2) / (D / 2 - hxr))
'Длина дуги наконечника зубца
lz = tz1 - lbsh
ug23 = pi / 2 - ((2 * pi / Z) - Atn((bsh / 2) / (D / 2 - hxr)))
ug24 = pi / 2 - Atn((bsh / 2) / (D / 2 - hxr))
'Строится дуга наконечника зубца
Set arcObjlz = ThisDrawing.ModelSpace.AddArc(clz, rlz, ug23, ug24)
'толщина линии 1 мм
arcObjlz.Lineweight = acLnWt100
```

1.19. КОПИРОВАНИЕ ЭЛЕМЕНТОВ ПАЗА

Перед тем как создать лист якоря из элементов паза, сначала необходимо создать копию паза и перенести его в другое место, чтобы затем отмасштабировать перенесенную копию (см. рис. 1.1, *б*). На рис. 1.5 показаны характерные точки переноса паза и его масштабирования.

Для того чтобы создать копию паза, воспользуемся методом **Copy**.

Команда копирования в общем виде:

RetVal = object.Copy,

где **RetVal** – название объекта создаваемой копии существующего объекта;

Object – название объекта, который копируется, им может быть любой объект рисования;

Copy – метод, который осуществляет копирование.

На языке VBA:

```
'Копируем построенные элементы паза

'Копия осевой линии паза
Dim copylineOs As AcadLine
```

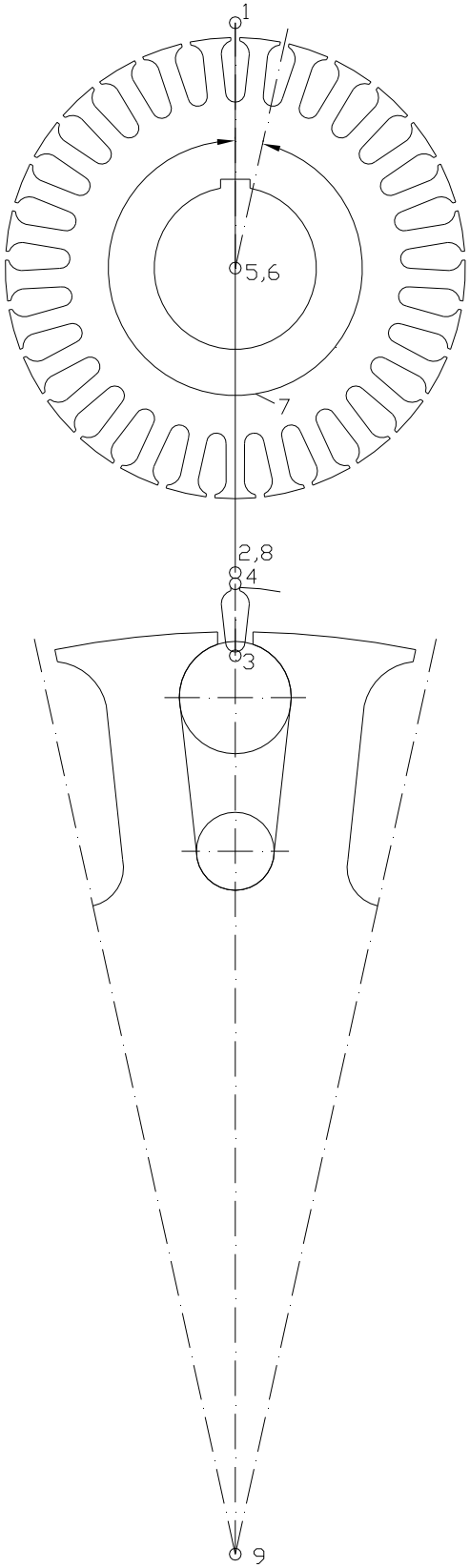


Рис. 1.5. Координаты точек, необходимые
для построения фрагмента листа якоря с пазами овальной формы

```
'Копия дуги радиуса r2

Dim copyarcObjr2 As AcadArc
'Копия линии,соединяющей r1 с r2
Dim copylineObjr1_r2 As AcadLine
'Копия дуги радиуса r1
Dim copyarcObjr1 As AcadArc
'Копия линии высоты шлица
Dim copylineObjhsh As AcadLine
'Копия зеркально отображенной относительно оси паза дуги радиуса
r2
Dim copymirrarcObjr2 As AcadArc
'Копия зеркально отображенной относительно оси паза
линии,соединяющей r1 с r2
Dim copymirrlineObjr1_r2 As AcadLine
'Копия зеркально отображенной относительно оси паза дуги радиуса
r1
Dim copymirrarcObjr1 As AcadArc
'Копия зеркально отображенной относительно оси паза линии высоты
шлица
Dim copymirrlineObjhsh As AcadLine
'Копия дуги наконечника зубца
Dim copyarcObjlz As AcadArc

'Операция копирования
Set copylineOs = lineOs.Copy()
Set copyarcObjr2 = arcObjr2.Copy()
Set copylineObjr1_r2 = lineObjr1_r2.Copy()
Set copyarcObjr1 = arcObjr1.Copy()
Set copylineObjhsh = lineObjhsh.Copy()
Set copymirrarcObjr2 = mirrarcObjr2.Copy()
Set copymirrlineObjr1_r2 = mirrlineObjr1_r2.Copy()
Set copymirrarcObjr1 = mirrarcObjr1.Copy()
Set copymirrlineObjhsh = mirrlineObjhsh.Copy()
Set copyarcObjlz = arcObjlz.Copy()
```

1.20. ПЕРЕНОС ЭЛЕМЕНТОВ ПАЗА

Для переноса объектов воспользуемся методом **Move**.
Общая команда переноса объекта:
object.Move Point1, **Point2**,

где **Object** – название перемещаемого объекта, им может быть любой объект рисования;

Point1, **Point2** – координаты первой и второй точек вектора, по которому осуществляется перенос объекта. Представляют собой одномерный массив, состоящий из трех элементов, имеет тип **Double**;

Move – метод, который осуществляет перенос.

Координаты линии, относительно которой осуществляется перемещение части паза (рис. 1.5, т.т. 1-2)

$$point1(0) = 0 : point1(1) = \frac{D}{2} + 5 : point1(2) = 0;$$

$$point2(0) = 0 : point2(1) = -\frac{D}{2} \cdot m2 - k : point2(2) = 0.$$

На языке VBA:

```
'Переносим скопированные элементы паза для построения фрагмента
листа якоря в другом масштабе

'Начальная координата линии переноса
Dim point1(0 To 2) As Double
'Конечная координата линии переноса
Dim point2(0 To 2) As Double

point1(0) = 0: point1(1) = D / 2 + 5: point1(2) = 0
point2(0) = 0: point2(1) = -D / 2 * m2 - k: point2(2) = 0

'Операция переноса
copylineOs.Move point1, point2
copyarcObjr2.Move point1, point2
copylineObjr1_r2.Move point1, point2
copyarcObjr1.Move point1, point2
copylineObjhsh.Move point1, point2
copymirrarcObjr2.Move point1, point2
copymirrlineObjr1_r2.Move point1, point2
copymirrarcObjr1.Move point1, point2
copymirrlineObjhsh.Move point1, point2
copyarcObjlz.Move point1, point2
```

1.21. ОСЕВАЯ ЛИНИЯ ФРАГМЕНТА ЛИСТА ЯКОРЯ

Меняем начальную и конечную координаты осевой линии у перенесенного фрагмента так, чтобы осевая линия выступала с каждой стороны на 5 мм.

Для того чтобы изменить начальную и конечную координаты оси, воспользуемся свойствами **StartPoint** и **EndPoint**.

Общая команда изменения координат начальной точки объекта:

```
object.StartPoint,
```

где **Object** – дуга, эллипс и линия (**Arc**, **Ellipse**, **Line**). К данным объектам применяется свойство **StartPoint** (рис. 1.6);

StartPoint – свойство, представляющее собой одномерный массив, состоящий из трех элементов (координат X, Y, Z), имеет тип **Variant**. Для дуги и эллипса данное свойство можно только определить (read-only). Для линии это свойство можно и изменить (read-write). Для редактирования дуги можно использовать свойства **EndAngle** и **Radius**. Для редактирования эллипса – свойства **EndAngle**, **MajorAxis**, and **RadiusRatio**.

Рис. 1.6. Начальные точки объектов: дуги, эллипса и линии

Общая команда изменения координат конечной точки объекта:

```
object.EndPoint,
```

где **Object** – дуга, эллипс и линия (**Arc**, **Ellipse**, **Line**). К данным объектам применяется свойство **EndPoint** (рис. 1.7);

EndPoint – свойство, представляющее собой одномерный массив, состоящий из трех элементов (координат X, Y, Z), Имеет тип **Variant**. Для дуги и эллипса данное свойство можно только определить (read-only). Для линии это свойство можно и изменить (read-write). Для редактирования дуги можно использовать свойства **EndAngle** и **Radius**. Для редактирования эллипса – свойства **EndAngle**, **MajorAxis**, and **RadiusRatio**.

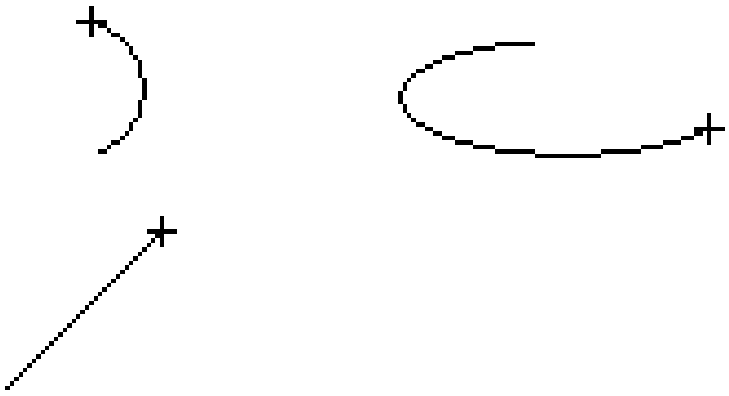


Рис. 1.7. Конечные точки объектов: дуги, эллипса и линии

Начальные и конечные координаты осевой линии на фрагменте листа якоря (рис. 1.5, т.т. 3-4):

$$startPointOs(0) = 0;$$

$$startPointOs(1) = -\frac{D}{2} \cdot m2 - k - 5 - hxr - hsh - r1 - h1 - r2 - \frac{5}{m};$$

$$startPointOs(2) = 0;$$

$$endPointOs(0) = 0;$$

$$endPointOs(1) = -\frac{D}{2} \cdot m2 - k - 5 \cdot \frac{m-1}{m};$$

$$endPointOs(2) = 0.$$

На языке VBA:

```
'Меняем начальную и конечную координаты осевой линии у
перенесенного фрагмента так,
'чтобы осевая линия выступала с каждой стороны на 5 мм
t1(0) = 0
t1(1) = -D / 2 * m2 - k - 5 - hxr - hsh - r1 - h1 - r2 - 5 / m
t1(2) = 0#
t2(0) = 0#
t2(1) = -D / 2 * m2 - k - 5 * (m - 1) / m
t2(2) = 0#

copylineOs.startPoint = t1
copylineOs.endPoint = t2

'Увеличение(уменьшение) листа якоря в m2 раз относительно центра -
точки basePoint1

'Координата точки, относительно которой происходит
увеличение(уменьшение) листа якоря в m2 раз
Dim basePoint1(0 To 2) As Double
basePoint1(0) = 0
basePoint1(1) = 0
basePoint1(2) = 0
```

1.22. МАСШТАБИРОВАНИЕ ЭЛЕМЕНТОВ ЛИСТА ЯКОРЯ

Выполненные элементы листа якоря выполняем в масштабе m2 относительно центра якоря (рис. 1.5, т. 5).

Масштабирование выполняется с помощью метода **ScaleEntity**.

Общая команда масштабирования объекта:

```
object.ScaleEntity BasePoint, ScaleFactor,
```

где **Object** – название масштабируемого объекта, им может быть любой объект рисования;

BasePoint – координата точки, относительно которой осуществляется масштабирование объекта. Представляет собой одномерный массив, состоящий из трех элементов, имеет тип Double;

ScaleFactor – величина масштаба объекта. Размеры объектов изменяются путем умножения на величину масштаба. При значении этой величины больше единицы происходит увеличение объекта. Если это значение лежит в пределах от 0 до 1 – уменьшение объекта и должно быть больше чем 0,0. Имеет тип Double;

ScaleEntity – метод, который осуществляет масштабирование (рис. 1.8).

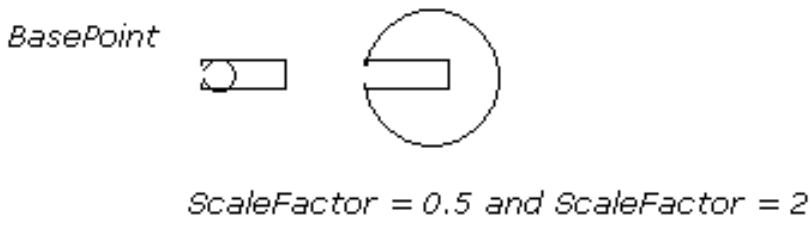


Рис. 1.8. Пояснительный рисунок к применению метода ScaleEntity

Координаты точки, относительно которой происходит увеличение (уменьшение) листа якоря в m2 раз:

$$basePoint1(0) = 0 : basePoint1(1) = 0 : basePoint1(2) = 0 .$$

На языке VBA:

```
'Увеличение(уменьшение) листа якоря в m2 раз относительно центра -
точки basePoint1

'Координата точки, относительно которой происходит
увеличение(уменьшение) листа якоря в m2 раз
Dim basePoint1(0 To 2) As Double
basePoint1(0) = 0
basePoint1(1) = 0
basePoint1(2) = 0

'Операция изменения масштаба
lineOs.ScaleEntity basePoint1, m2
arcObjr2.ScaleEntity basePoint1, m2
lineObjr1_r2.ScaleEntity basePoint1, m2
arcObjr1.ScaleEntity basePoint1, m2
lineObjhsh.ScaleEntity basePoint1, m2
```

```
mirrarcObjr2.ScaleEntity basePoint1, m2
mirrlineObjr1_r2.ScaleEntity basePoint1, m2
mirrarcObjr1.ScaleEntity basePoint1, m2
mirrlineObjhsh.ScaleEntity basePoint1, m2
arcObjlz.ScaleEntity basePoint1, m2
arcObjD0.ScaleEntity basePoint1, m2
lineObjbshp.ScaleEntity basePoint1, m2
lineObjhshp.ScaleEntity basePoint1, m2
mirrlineObjbshp.ScaleEntity basePoint1, m2
mirrlineObjhshp.ScaleEntity basePoint1, m2
```

1.23. ВЫЧЕРЧИВАНИЕ ЛИСТА ЯКОРЯ С ПОМОЩЬЮ ПОЛЯРНОГО МАССИВА

Так как построенный паз и дуга наконечника зубца абсолютно повторяются по окружности якоря, то для построения всего якоря можно воспользоваться так называемой операцией полярного (кругового) массива – копированием элементов по окружности относительно выбранного центра с указанием количества копий исходного объекта и угла заполнения.

Общая команда создания полярного массива:

```
RetVal = object.ArrayPolar (NumberOfObjects, AngleToFill, CenterPoint),
```

где **RetVal** – название создаваемого массива объекта методом **ArrayPolar**;

Object – название объекта, который размножается полярным массивом, им может быть любой объект рисования;

NumberOfObjects – количество элементов полярного массива. Должно быть целым положительным числом, больше 1, имеет тип Integer;

AngleToFill – угол заполнения, измеряется в радианах. Положительная величина определяет создание массива против часовой стрелки. Отрицательная – по часовой стрелки. Угол не может быть равен 0. Имеет тип Double;

CenterPoint – координата точки, относительно которой осуществляется операция полярного массива. Представляет собой одномерный массив, состоящий из трех элементов, имеет тип Double;

Полярный массив со следующими параметрами: Number Of Objects = 5, AngleToFill = 180, CenterPoint = 0,0,0 показан на рис. 1.9.

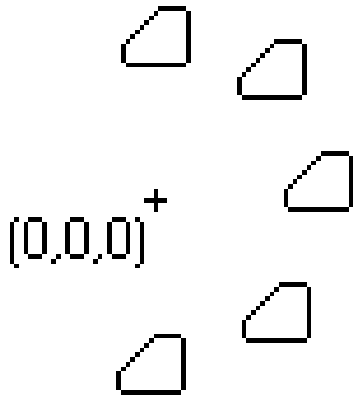


Рис. 1.9. Пояснительный рисунок к применению метода ArrayPolar

Количество объектов паза: число пазов Z.

Координаты точки относительно, которой происходит вычерчивание листа якоря (рис. 1.5, т. 6):

$$basePnt(0)=0 : basePnt(1)=0 : basePnt(2)=0,$$

где *angleToFil* – угол заполнения элементами пазово-зубцовой части с помощью полярного массива (рис. 1.5, т. 7):

$$angleToFill = 2\pi - \frac{2\pi}{z}.$$

На языке VBA:

```
' Получаем лист якоря с помощью полярного массива из ранее
вычерченных элементов.
' Число элементов полярного массива равно числу зубцов якоря (Z)


'Количество элементов полярного массива
```

```
Dim noOfObjects As Integer
'Координата точки, относительно которой происходит вычерчивание
листа якоря
Dim basePnt(0 To 2) As Double

Z = Z
basePnt(0) = 0: basePnt(1) = 0: basePnt(2) = 0

' Создаем Z (количество зубцов) копий объектов, вращая и копируя
их относительно точки (0,0,0):

'дуги радиуса r2
Dim retarcObjr2 As Variant
'линии, соединяющей r1 с r2
Dim retlineObjr1_r2 As Variant
'дуги радиуса r1
Dim retarcObjr1 As Variant
'линии высоты шлица
Dim retlineObjhsh As Variant
'зеркально отображенной дуги радиуса r2
Dim retmirrarcObjr2 As Variant
'зеркально отображенной линии, соединяющей r1 с r2
Dim retmirrlineObjr1_r2 As Variant
'зеркально отображенной дуги радиуса r1
Dim retmirrarcObjr1 As Variant
'зеркально отображенной линии высоты шлица
Dim retmirrlineObjhsh As Variant
'осевой линии паза
Dim retlineOs As Variant
'дуги наконечника зубца
Dim retarcObjlz As Variant
'угол заполнения элементами
Dim angleToFill As Double

angleToFill = 2 * pi - 2 * pi / Z

'Операция полярного массива
retarcObjr2 = arcObjr2.ArrayPolar(Z, angleToFill, basePnt)
retlineObjr1_r2 = lineObjr1_r2.ArrayPolar(Z, angleToFill, basePnt)
retarcObjr1 = arcObjr1.ArrayPolar(Z, angleToFill, basePnt)
retlineObjhsh = lineObjhsh.ArrayPolar(Z, angleToFill, basePnt)
retmirrarcObjr2 = mirrarcObjr2.ArrayPolar(Z, angleToFill, basePnt)
retmirrlineObjr1_r2 = mirrlineObjr1_r2.ArrayPolar(Z, angleToFill,
basePnt)
```

```
retmirrarcObjr1 = mirrarcObjr1.ArrayPolar(Z, angleToFill, basePnt)
retmirrlineObjhsh = mirrlineObjhsh.ArrayPolar(Z, angleToFill,
basePnt)
retlineOs = lineOs.ArrayPolar(2, angleToFill, basePnt)
retarcObjlz = arcObjlz.ArrayPolar(Z, angleToFill, basePnt)
```

1.24. ИЗМЕНЕНИЕ МАСШТАБА ФРАГМЕНТА ЛИСТА ЯКОРЯ

Масштаб фрагмента листа якоря происходит в m раз (величина, выбранная пользователем) относительно точки т. 8 (см. рис. 1.5).

Координаты точки, относительно которой происходит изменение масштаба фрагмента листа якоря:

$$basePoint(0) = 0;$$

$$basePoint(1) = -\frac{D}{2} \cdot m2 - k;$$

$$basePoint(2) = 0.$$

На языке VBA:

```
'Увеличение (уменьшение) фрагмента листа якоря в m раз
относительно точки basePoint

'Координата точки, относительно которой происходит
увеличение(уменьшение)
'фрагмента листа якоря в m раз
Dim basePoint(0 To 2) As Double
basePoint(0) = 0
basePoint(1) = -D / 2 * m2 - k
basePoint(2) = 0

'Операция масштабирования
copylineOs.ScaleEntity basePoint, m
copyarcObjr2.ScaleEntity basePoint, m
copylineObjr1_r2.ScaleEntity basePoint, m
copyarcObjr1.ScaleEntity basePoint, m
copylineObjhsh.ScaleEntity basePoint, m
copymirrarcObjr2.ScaleEntity basePoint, m
copymirrlineObjr1_r2.ScaleEntity basePoint, m
copymirrarcObjr1.ScaleEntity basePoint, m
copymirrlineObjhsh.ScaleEntity basePoint, m
copyarcObjlz.ScaleEntity basePoint, m
```

1.25. ВЫЧЕРЧИВАНИЕ ФРАГМЕНТА ЛИСТА ЯКОРЯ С ПОМОЩЬЮ ПОЛЯРНОГО МАССИВА

Получаем фрагмент листа якоря с помощью полярного массива из ранее перенесенных элементов.

Формирование происходит следующим образом: создаются две копии элементов паза, один из которых повернут на угол (360/Z) по часовой стрелке, другой – против часовой.

Координаты точки, относительно которой происходит формирование фрагмента листа якоря (рис. 1.5, т. 9):

$$basePnt1(0) = 0;$$

$$basePnt1(1) = -\frac{D}{2} \cdot m2 - k - \left(\frac{D}{2} + 5\right) \cdot m;$$

$$basePnt1(2) = 0.$$

```
' Получаем фрагмент листа якоря с помощью полярного массива
' из ранее вычерченных элементов.

'Координата точки, относительно которой происходит формирование
фрагмента листа якоря
Dim basePnt1(0 To 2) As Double
basePnt1(0) = 0
basePnt1(1) = -D / 2 * m2 - k - (D / 2 + 5) * m
basePnt1(2) = 0

'Формирование зубцовой зоны из скопированных элементов паза во
фрагменте чертежа листа якоря:

'формирование происходит следующим образом:
'создаются две копии элементов паза, один из которых повернут на
угол (360/Z) по часовой стрелке, другой - против
Dim retcopyarcObjr2 As Variant
Dim retcopylineObjr1_r2 As Variant
Dim retcopyarcObjr1 As Variant
Dim retcopylineObjhsh As Variant
Dim retcopymirrarcObjr2 As Variant
Dim retcopymirrlineObjr1_r2 As Variant
```

```
Dim retcopymirrarcObjr1 As Variant
Dim retcopymirrlineObjhsh As Variant
Dim retcopylineOs As Variant
Dim retcopyarcObjlz As Variant

'Операция полярного массива
retcopyarcObjr2 = copyarcObjr2.ArrayPolar(2, angleToFill,
basePnt1)
retcopylineObjr1_r2 = copylineObjr1_r2.ArrayPolar(2, angleToFill,
basePnt1)
retcopyarcObjr1 = copyarcObjr1.ArrayPolar(2, angleToFill,
basePnt1)
retcopylineObjhsh = copylineObjhsh.ArrayPolar(2, angleToFill,
basePnt1)
retcopymirrarcObjr2 = copymirrarcObjr2.ArrayPolar(2, -angleToFill,
basePnt1)
retcopymirrlineObjr1_r2 = copymirrlineObjr1_r2.ArrayPolar(2, -
angleToFill, basePnt1)
retcopymirrarcObjr1 = copymirrarcObjr1.ArrayPolar(2, -angleToFill,
basePnt1)
retcopymirrlineObjhsh = copymirrlineObjhsh.ArrayPolar(2, -
angleToFill, basePnt1)
retcopylineOs = copylineOs.ArrayPolar(2, angleToFill, basePnt1)
retcopylineOs = copylineOs.ArrayPolar(2, -angleToFill, basePnt1)
retcopyarcObjlz = copyarcObjlz.ArrayPolar(2, -angleToFill,
basePnt1)
```

Все элементы листа якоря построены. Далее можно переходить к автоматизированному нанесению размеров. Данная тема будет рассмотрена в следующем разделе.

2. АВТОМАТИЗИРОВАННОЕ НАНЕСЕНИЕ РАЗМЕРОВ И ВСПОМОГАТЕЛЬНОГО ТЕКСТА НА ЛИСТЕ ЯКОРЯ И ЕГО ФРАГМЕНТЕ

2.1. ХАРАКТЕРНЫЕ ТОЧКИ ДЛЯ НАНЕСЕНИЯ РАЗМЕРОВ

Все построения размеров ведутся также как и элементы чертежа – с помощью характерных точек (рис. 2.1).

2.2. ВСПОМОГАТЕЛЬНАЯ ОКРУЖНОСТЬ ДЛЯ ПРОСТАВЛЕНИЯ БОЛЬШЕГО ДИАМЕТРА ПАЗА РАДИУСА r1

Построим вспомогательную окружность *1* для простановки диаметра (рис. 2.1).

Координаты центра вспомогательной окружности для проставления диаметра радиуса r1 (рис. 2.1, т. 2):

$$cvspomogr2(0) = 0;$$

$$cvspomogr2(1) = -\frac{D}{2} \cdot m2 - k - m \cdot 5 \cdot \frac{m-1}{m} - 5 - m \cdot (hxr + hsh + r1 + h1);$$

$$cvspomogr2(2) = 0.$$

Радиус вспомогательной окружности для проставления диаметра радиуса r1(т. 3):

$$rvspomogr1 = m \cdot r1.$$

На языке VBA:

```
'Вспомогательная окружность для проставления диаметра радиуса r1
Dim circlevspomogr1 As AcadCircle
'Центр вспомогательной окружности для проставления диаметра
радиуса r1
Dim cvspomogr1(0 To 2) As Double
'Радиус вспомогательной окружности для проставления диаметра
радиуса r1
Dim rvspomogr1 As Double
```

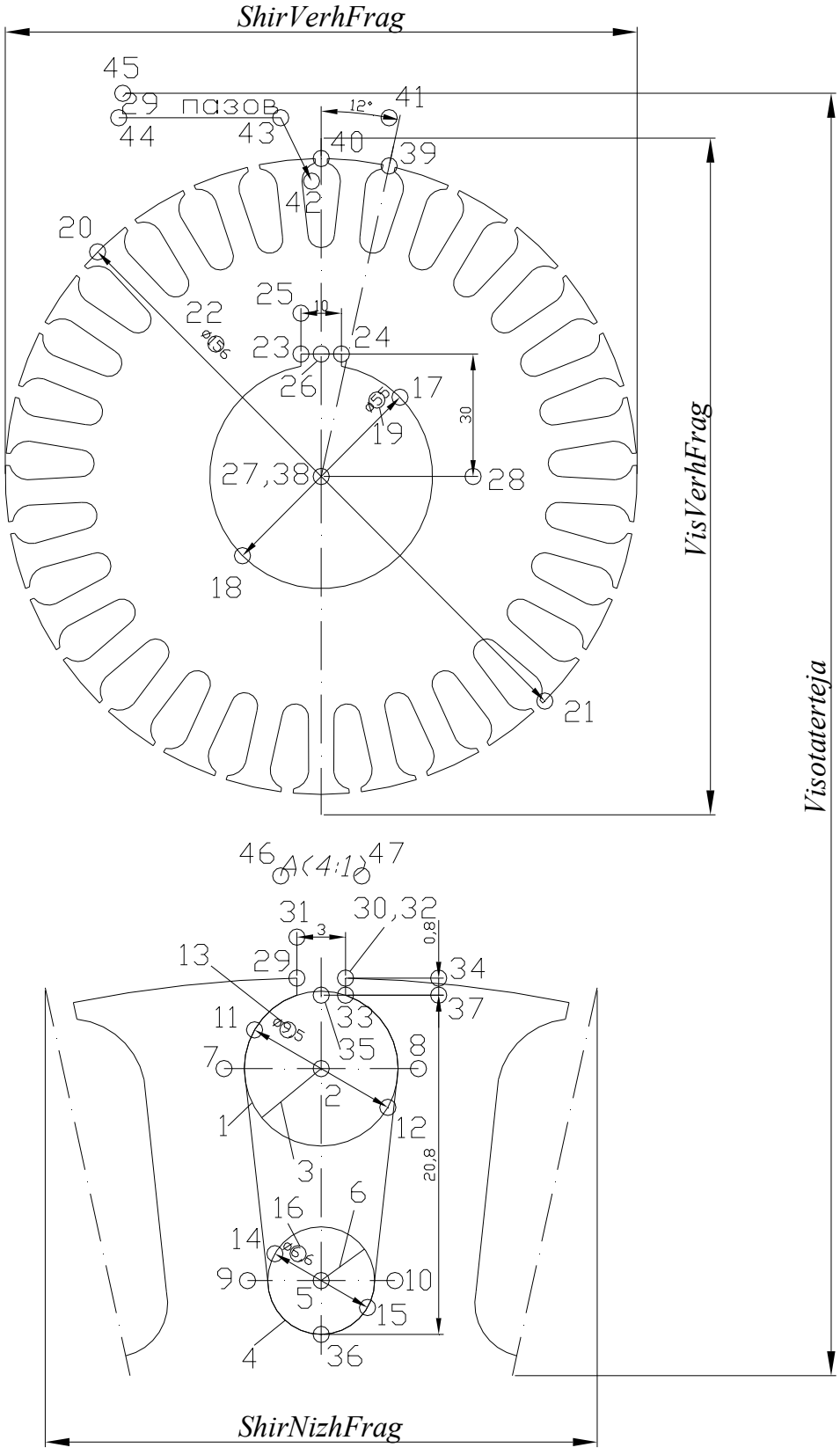


Рис. 2.1. Координаты точек, необходимые для нанесения размеров на чертеже листа

```
cvspomogr1(0) = 0
```

```
cvspomogr1(1) = -D / 2 * m2 - k - m * 5 * (m - 1) / m - 5 - m * (hxr + hsh + r1)
cvspomogr1(2) = 0
rvspomogr1 = m * r1

'Строится вспомогательная окружность для проставления диаметра
радиуса r1
Set circlevspomogr1 = ThisDrawing.ModelSpace.AddCircle(cvspomogr1,
rvspomogr1)
'толщина линии вспомогательной окружности 0,5 мм
circlevspomogr1.Lineweight = acLnWt050
```

2.3. ВСПОМОГАТЕЛЬНАЯ ОКРУЖНОСТЬ ДЛЯ ПРОСТАВЛЕНИЯ МЕНЬШЕГО ДИАМЕТРА ПАЗА РАДИУСА r2

Построим вспомогательную окружность *4* для простановки диаметра (рис. 2.1).

Координаты центра вспомогательной окружности для проставления диаметра радиуса r2 (рис. 2.1, т. 5):

$$cvspomogr2(0) = 0;$$

$$cvspomogr2(1) = -\frac{D}{2} \cdot m2 - k - m \cdot 5 \cdot \frac{m-1}{m} - 5 - m \cdot (hxr + hsh + r1 + h1);$$

$$cvspomogr2(2) = 0.$$

Радиус вспомогательной окружности для проставления диаметра радиуса r2 (рис. 2.1, т. 6):

$$rvspomogr2 = m \cdot r2.$$

На языке VBA:

```
'Вспомогательная окружность для проставления диаметра радиуса r2
Dim circlevspomogr2 As AcadCircle
'Центр вспомогательной окружности для проставления диаметра
радиуса r2
Dim cvspomogr2(0 To 2) As Double
'Радиус вспомогательной окружности для проставления диаметра
радиуса r2
Dim rvspomogr2 As Double
cvspomogr2(0) = 0
```

```
cvspomogr2(1) = -D / 2 * m2 - k - m * 5 * (m - 1) / m - 5 - m *
(hxr + hsh + r1 + h1)
cvspomogr2(2) = 0
rvspomogr2 = m * r2

'Строится вспомогательная окружность для проставления диаметра
радиуса r2
Set circlevspomogr2 = ThisDrawing.ModelSpace.AddCircle(cvspomogr2,
rvspomogr2)

'толщина линии вспомогательной окружности 0,5 мм
circlevspomogr2.Lineweight = acLnWt050
```

2.4. ГОРИЗОНТАЛЬНАЯ ОСЬ r1

Создадим горизонтальную осевую линию, проходящую по диаметру большего радиуса паза (рис. 2.1, т.т. 7-8).

Начальные и конечные координаты точек для горизонтальной осевой линии r1:

$$sOs\_r1(0) = -m \cdot r1 - 5;$$

$$sOs\_r1(1) = -\frac{D}{2} \cdot m2 - k - m \cdot 5 \cdot \frac{m-1}{m} - 5 - m \cdot (hxr + hsh + r1);$$

$$sOs\_r1(2) = 0;$$

$$eOs\_r1(0) = m \cdot r1 + 5;$$

$$eOs\_r1(1) = -\frac{D}{2} \cdot m2 - k - m \cdot 5 \cdot \frac{m-1}{m} - 5 - m \cdot (hxr + hsh + r1);$$

$$eOs\_r1(2) = 0.$$

На языке VBA:

```
'Осевые линии строятся типом линии "ACAD_ISO10W100"
ThisDrawing.ActiveLinetype =
ThisDrawing.Linetypes.Item("ACAD_ISO10W100")

'горизонтальная ось r1
Dim lineOs_r1 As AcadLine
'Начальная координата горизонтальной осевой линии r1
Dim sOs_r1(0 To 2) As Double
'Конечная координата горизонтальной осевой линии r1
```

```
Dim eOs_r1(0 To 2) As Double

'Определяем начальную и конечную координаты точек для
горизонтальной осевой линии r1
sOs_r1(0) = -m * r1 - 5
sOs_r1(1) = -D / 2 * m2 - k - m * 5 * (m - 1) / m - 5 - m * (hxr + hsh + r1)
sOs_r1(2) = 0#
eOs_r1(0) = m * r1 + 5
eOs_r1(1) = -D / 2 * m2 - k - m * 5 * (m - 1) / m - 5 - m * (hxr + hsh + r1)
eOs_r1(2) = 0#

'Строится горизонтальная осевая линия r1
Set lineOs_r1 = ThisDrawing.ModelSpace.AddLine(sOs_r1, eOs_r1)
'толщина линии 0,5 мм
lineOs_r1.Lineweight = acLnWt050
```

2.5. ГОРИЗОНТАЛЬНАЯ ОСЬ r2

Создадим горизонтальную осевую линию, проходящую по диаметру меньшего радиуса паза (рис. 2.1, т.т. 9-10).

Начальные и конечные координаты точек для горизонтальной осевой линии r2:

$$sOs\_r2(0) = -m \cdot r2 - 5;$$

$$sOs\_r2(1) = -\frac{D}{2} \cdot m2 - k - m \cdot 5 \cdot \frac{m-1}{m} - 5 - m \cdot (hxr + hsh + r1 + h1);$$

$$sOs\_r2(2) = 0;$$

$$eOs\_r2(0) = m \cdot r2 + 5;$$

$$eOs\_r2(1) = -\frac{D}{2} \cdot m2 - k - m \cdot 5 \cdot \frac{m-1}{m} - 5 - m \cdot (hxr + hsh + r1 + h1);$$

$$eOs\_r2(2) = 0.$$

На языке VBA:

```
'горизонтальная ось r2
Dim lineOs_r2 As AcadLine
'Начальная координата горизонтальной осевой линии r2
Dim sOs_r2(0 To 2) As Double
'Конечная координата горизонтальной осевой линии r2
Dim eOs_r2(0 To 2) As Double
```

```
'Определяем начальную и конечную координаты точек для
горизонтальной осевой линии r2
sOs_r2(0) = -m * r2 - 5
sOs_r2(1) = -D / 2 * m2 - k - m * 5 * (m - 1) / m - 5 - m * (hxr +
hsh + r1 + h1)
sOs_r2(2) = 0#

eOs_r2(0) = m * r2 + 5
eOs_r2(1) = -D / 2 * m2 - k - m * 5 * (m - 1) / m - 5 - m * (hxr +
hsh + r1 + h1)
eOs_r2(2) = 0#

 'Строится горизонтальная осевая линия r2
Set lineOs_r2 = ThisDrawing.ModelSpace.AddLine(sOs_r2, eOs_r2)

'толщина линии 0,5 мм
lineOs_r2.Lineweight = acLnWt050

'Возвращаемся к исходному типу линии "ByLayer"
ThisDrawing.ActiveLinetype = ThisDrawing.Linetypes.Item("ByLayer")
```

2.6. РАЗМЕР ДИАМЕТРА РАДИУСА r1

Нанесение диаметрального размера выполняется созданием объекта **AcadDimDiametric** с помощью метода **AddDimDiametric**. Размерная линия будет проходить под углом к горизонтальной линии. Поэтому в формулах для построения размера диаметра имеется компонент, указывающий на то, что он будет повернут. На рис. 2.2 показаны примеры простановки диаметров.

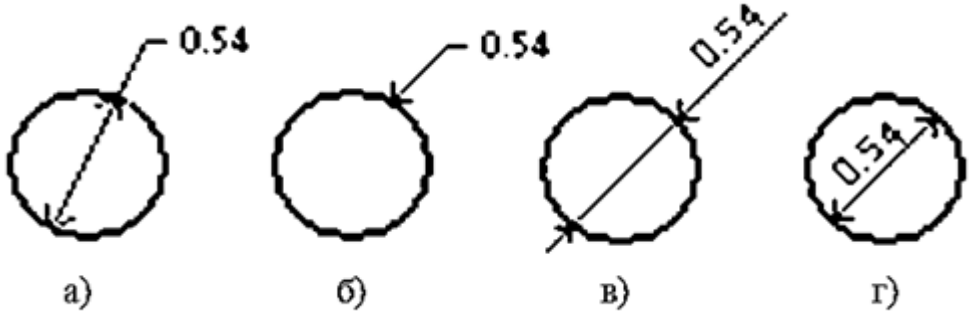


Рис. 2.2. Варианты проставления диаметрального размера

Общая команда построения диаметрального размера:

```
RetVal = object.AddDimDiametric(ChordPoint, FarChordPoint, LeaderLength),
```

где **RetVal** – название создаваемого объекта DimDiametric (диаметральный размер);

Object – объект или объекты, к которым применяется метод AddDimDiametric. В качестве объектов могут быть использованы: ModelSpace Collection, PaperSpace Collection, Block;

ChordPoint – одномерный массив, состоящий из трех элементов, представляющий собой координаты в трехмерном пространстве (X, Y, Z), определяющий первую точку размерной линии диаметра, находящейся на дуге или окружности. Имеет тип Double;

FarChordPoint – одномерный массив, состоящий из трех элементов, представляющий собой координаты в трехмерном пространстве (X, Y, Z), определяющий вторую точку размерной линии диаметра, находящейся на дуге или окружности. Имеет тип Double;

LeaderLength – положительная величина, представляющая расстояние от ChordPoint до текста, указывающего величину диаметра или до «полки» размерной линии (рис. 2.2, *а*, *б*). Имеет тип Double.

Кроме того, численное значение, проставляемое на размере, будет смещено от центра размерной линии вдоль размерной линии. Для смещения размерного текста используются свойства **TextMovement** и **TextPosition**. Данные свойства зависят друг от друга и используются в комплексе.

Свойство перемещения текста над размерной линией:

```
object.TextMovement,
```

где **Object** – объекты типа **Dim3PointAngular**, **DimAligned**, **DimAngular**, **DimDiametric**, **DimOrdinate**, **DimRadial**, **DimRotated**, к которым применяется свойство **TextMovement**;

TextMovement – свойство, которое может принимать значения указанные в табл. 2.1.

Таблица 2.1

Возможные значения свойства TextMovement

Значение свойства	Назначение
acDimLineWithText	Размерная линия вместе с текстом
acMoveTextAddLeader	Текст перемещается независимо от размерной линии, с выноской, нарисованной от текста до размерной линии
acMoveTextNoLeader	Текст перемещается независимо от размерной линии, без выноски

По умолчанию устанавливается **acMoveTextNoLeader**.

Свойство изменения координат позиции аннотационного текста у размерных линий:

```
object.TextPosition,
```

где **Object** – объекты типа Dim3PointAngular, DimAligned, DimAngular, DimDiametric, DimOrdinate, DimRadial, DimRotated, к которым применяется свойство TextPosition;

TextPosition – свойство, представляющее собой одномерный массив, состоящий из трех элементов (координат X, Y, Z). Имеет тип Variant (рис. 2.3).

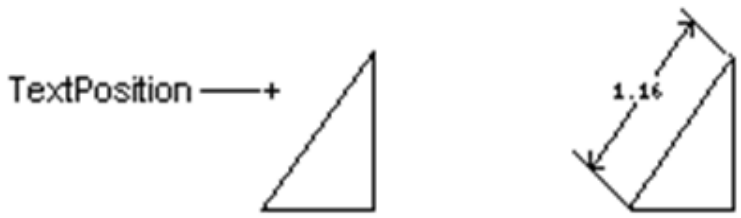


Рис. 2.3. Пояснительный рисунок
к применению свойства TextPosition

На чертежах, выполненных в масштабах, отличающихся от 1:1, текст размеров должен проставляться в реальных величинах. Если наносить размер обычном способом, то числовой размер на масшта-

бированном чертеже также будет отличаться от реального и изменяться в зависимости от выбранного масштаба. Чтобы этого избежать, нужно воспользоваться свойством LinearScaleFactor, которое позволяет вернуться к реальным размерам на масштабированных чертежах.

Свойство простановки реальных размеров при масштабе, отличающемся от 1:1:

```
object.LinearScaleFactor,
```

где **Object** – объекты типа DimAligned, DimDiametric, DimOrdinate, DimRadial, DimRotated, к которым применяется свойство LinearScaleFactor;

LinearScaleFactor – свойство, замещающее величину реального размера. Новая величина размера равна реальной, умноженной на число, присвоенное данному свойству. Это число должно быть больше 0,0. По умолчанию величина свойства равна 1. Имеет тип Double.

В нашем случае данное свойство для листа якоря и фрагмента листа якоря будут соответственно равны:

$$\text{LinearScaleFactor}_2 = \frac{1}{\text{m}_2};$$

$$\text{LinearScaleFactor} = \frac{1}{\text{m}}.$$

Координаты первой точки размерной линии диаметра радиуса r1 (рис. 2.1, т. 11):

$$chD\_r1(0) = -m \cdot r1 \cdot \text{Sin}\left(\frac{60 \cdot \pi}{180}\right);$$

$$chD\_r1(1) = -\frac{D}{2} \cdot m2 - k - m \cdot 5 \cdot \frac{m-1}{m} - 5 - m \cdot (hxr + hsh + r1) + m \cdot r1 \cdot \text{Cos}\left(\frac{60 \cdot \pi}{180}\right);$$

$$chD\_r1(2) = 0.$$

Координаты второй точки размерной линии диаметра радиуса r1 (рис. 2.1, т. 12):

$$fChD\_r1(0) = m \cdot r1 \cdot \mathrm{Sin}\left(\frac{60 \cdot \pi}{180}\right);$$

$$fChD\_r1(1) = -\frac{D}{2} \cdot m2 - k - m \cdot 5 \cdot \frac{m-1}{m} - 5 - m \cdot (hxr + hsh + r1) - m \cdot r1 \cdot \mathrm{Cos}\left(\frac{60 \cdot \pi}{180}\right);$$

$$fChD\_r1(2) = 0.$$

Координаты положения текста размера диаметра радиуса r1 (рис. 2.1, т. 13):

$$lD\_r1(0) = -\frac{m \cdot r1 \cdot \mathrm{Sin}\left(\frac{60 \cdot \pi}{180}\right)}{2};$$

$$lD\_r1(1) = -\frac{D}{2} \cdot m2 - k - sm \cdot 5 \cdot \frac{m-1}{m} - 5 - m \cdot (hxr + hsh + r1) + m \cdot r1 \cdot \mathrm{Cos}\left(\frac{60 \cdot \pi}{180}\right);$$

$$lD\_r1(2) = 0.$$

На языке VBA:

```
'Размер диаметра радиуса r1
Dim dimD_r1 As AcadDimDiametric
'Первая точка размерной линии
Dim chD_r1(0 To 2) As Double
'Вторая точка размерной линии
Dim fchD_r1(0 To 2) As Double
'Расстояние от первой точки размерной линии до текста размера
Dim lLD_r1 As Double
'Положение текста размера
Dim lD_r1(0 To 2) As Double
' Определение координат и параметров диаметрального размера
chD_r1(0) = -m * r1 * Sin(60 * radian)
chD_r1(1) = -D / 2 * m2 - k - m * 5 * (m - 1) / m - 5 - m * (hxr +
hsh + r1) + m * r1 * Cos(60 * radian)
chD_r1(2) = 0#
fchD_r1(0) = m * r1 * Sin(60 * radian)
fchD_r1(1) = -D / 2 * m2 - k - m * 5 * (m - 1) / m - 5 - m * (hxr
+ hsh + r1) - m * r1 * Cos(60 * radian)
fchD_r1(2) = 0#
lLD_r1 = 3
'Нанесение размера диаметра радиуса r1
Set dimD_r1 = ThisDrawing.ModelSpace.AddDimDiametric(chD_r1,
fchD_r1, lLD_r1)
'толщина линии 0,5 мм
dimD_r1.Lineweight = acLnWt050
```

```
'Перенос текста независимо от размерной линии без выноски
(позволяет увидеть линию между стрелками)
dimD_r1.TextMovement = acMoveTextNoLeader
'Определяем координаты точек
lD_r1(0) = -m * r1 * Sin(60 * radian) / 2
lD_r1(1) = -D / 2 * m2 - k - m * 5 * (m - 1) / m - 5 - m * (hxr +
hsh + r1) + m * r1 * Cos(60 * radian)
lD_r1(2) = 0
dimD_r1.TextPosition = lD_r1
'Установка масштаба линейного размера (простановка реального
размера диметра радиуса r1 при масштабе отличающимся от 1:1)
dimD_r1.LinearScaleFactor = 1 / m
```

2.7. РАЗМЕР ДИАМЕТРА РАДИУСА r2

Координаты первой точки размерной линии диаметра радиуса r2 (рис. 2.1, т. 14):

$$ch\_r2(0) = -m \cdot r2 \cdot \mathrm{Sin}\left(\frac{60 \cdot \pi}{180}\right);$$

$$chD\_r2(1) = -\frac{D}{2} \cdot m2 - k - m \cdot 5 \cdot \frac{m-1}{m} - 5 - m \cdot (hxr + hsh + r1 + h1) + m \cdot r2 \cdot \mathrm{Cos}\left(\frac{60 \cdot \pi}{180}\right);$$

$$chD\_r2(2) = 0.$$

Координаты второй точки размерной линии диаметра радиуса r2 (рис. 2.1, т. 15):

$$fChD\_r2(0) = m \cdot r2 \cdot \mathrm{Sin}\left(\frac{60 \cdot \pi}{180}\right);$$

$$fChD\_r2(1) = -\frac{D}{2} \cdot m2 - k - m \cdot 5 \cdot \frac{m-1}{m} - 5 - m \cdot (hxr + hsh + r1 + h1) + m \cdot r2 \cdot \mathrm{Cos}\left(\frac{60 \cdot \pi}{180}\right);$$

$$fChD\_r2(2) = 0.$$

Координаты положения текста размера диаметра радиуса r2 (рис. 2.1, т. 16):

$$lD\_r2(0) = -\frac{m \cdot r2 \cdot \mathrm{Sin}\left(\frac{60 \cdot \pi}{180}\right)}{2};$$

$$lD\_r2(1) = -\frac{D}{2} \cdot m2 - k - m \cdot 5 \cdot \frac{m-1}{m} - 5 - m \cdot (hxr + hsh + r1 + h1) + m \cdot r2 \cdot \mathrm{Cos}\left(\frac{60 \cdot \pi}{180}\right);$$

$$mD\_r2(2) = 0.$$

На языке VBA:

```
'Размер диаметра радиуса r2
Dim dimD_r2 As AcadDimDiametric
'Первая точка размерной линии
Dim chD_r2(0 To 2) As Double
'Вторая точка размерной линии
Dim fchD_r2(0 To 2) As Double
'Расстояние от первой точки размерной линии до текста размера
Dim lL_r2 As Double
'Положение текста размера
Dim l_r2(0 To 2) As Double

' Определение координат и параметров диаметрального размера
chD_r2(0) = -m * r2 * Sin(60 * radian)
chD_r2(1) = -D / 2 * m2 - k - m * 5 * (m - 1) / m - 5 - m * (hxr +
hsh + r1 + h1) + m * r2 * Cos(60 * radian)
chD_r2(2) = 0#
fchD_r2(0) = m * r2 * Sin(60 * radian)
fchD_r2(1) = -D / 2 * m2 - k - m * 5 * (m - 1) / m - 5 - m * (hxr
+ hsh + r1 + h1) - m * r2 * Cos(60 * radian)
fchD_r2(2) = 0#
lL_r2 = 3

'Нанесение размера диаметра радиуса r2
Set dimD_r2 = ThisDrawing.ModelSpace.AddDimDiametric(chD_r2,
fchD_r2, lL_r2)
'толщина линии 0,5 мм
dimD_r2.Lineweight = acLnWt050
'Перенос текста независимо от размерной линии без выноски
(позволяет увидеть линию между стрелками)
dimD_r2.TextMovement = acMoveTextNoLeader

l_r2(0) = -m * r2 * Sin(60 * radian) / 2
l_r2(1) = -D / 2 * m2 - k - m * 5 * (m - 1) / m - 5 - m * (hxr +
hsh + r1 + h1) + m * r2 * Cos(60 * radian)
l_r2(2) = 0
dimD_r2.TextPosition = l_r2
'Установка масштаба линейного размера (простановка реального
размера диметра радиуса r2 при масштабе отличающимся от 1:1)
```

```
dimD_r2.LinearScaleFactor = 1 / m
```

2.8. РАЗМЕР ДИАМЕТРА D0

Координаты первой точки размерной линии диаметра D0 (рис. 2.1, т. 17):

$$chD0(0) = \frac{m2 \cdot D0}{2} \cdot \mathrm{Sin}\left(\frac{45 \cdot \pi}{180}\right);$$

$$chD0(1) = \frac{m2 \cdot D0}{2} \cdot \mathrm{Cos}\left(\frac{45 \cdot \pi}{180}\right);$$

$$chD0(2) = 0.$$

Координаты второй точки размерной линии диаметра D0 (рис. 2.1, т. 18):

$$fChD0(0) = -\frac{m2 \cdot D0}{2} \cdot \mathrm{Sin}\left(\frac{45 \cdot \pi}{180}\right);$$

$$fChD0(1) = -\frac{m2 \cdot D0}{2} \cdot \mathrm{Cos}\left(\frac{45 \cdot \pi}{180}\right);$$

$$fChD0(2) = 0.$$

Координаты положения текста размера диаметра D0 (рис. 2.1, т. 19):

$$lD0(0) = \frac{m2 \cdot D0}{4};$$

$$lD0(1) = \frac{m2 \cdot D0}{4} + 5;$$

$$lD0(2) = 0.$$

На языке VBA:

```
'Размер диаметра D0
Dim dimD0 As AcadDimDiametric
'Первая точка размерной линии
Dim chD0(0 To 2) As Double
'Вторая точка размерной линии
Dim fchD0(0 To 2) As Double
'Расстояние от первой точки размерной линии до текста размера
Dim lLD0 As Double
```

```
'Положение текста размера
Dim lD0(0 To 2) As Double

' Определение координат и параметров диаметрального размера
chD0(0) = m2 * D0 / 2 * Sin(45 * radian)
chD0(1) = m2 * D0 / 2 * Cos(45 * radian)
chD0(2) = 0#
fchD0(0) = -m2 * D0 / 2 * Sin(45 * radian)
fchD0(1) = -m2 * D0 / 2 * Cos(45 * radian)
fchD0(2) = 0#
lLD0 = 3

'Нанесение размера диаметра D0
Set dimD0 = ThisDrawing.ModelSpace.AddDimDiametric(chD0, fchD0,
lLD0)
'толщина линии 0,5 мм
dimD0.Lineweight = acLnWt050
'Перенос текста независимо от размерной линии без выноски
(позволяет увидеть линию между стрелками)
dimD0.TextMovement = acMoveTextNoLeader

lD0(0) = m2 * D0 / 4
lD0(1) = m2 * D0 / 4 + 5
lD0(2) = 0
dimD0.TextPosition = lD0
'Установка масштаба линейного размера (простановка реального
размера диметра D0 при масштабе отличающимся от 1:1)
dimD0.LinearScaleFactor = 1 / m2
```

2.9. РАЗМЕР ДИАМЕТРА D

Координаты первой точки размерной линии диаметра D (рис. 2.1, т. 20):

$$chD(0) = -\frac{m2 \cdot D}{2} \cdot \mathrm{Sin}\left(-3 \cdot \pi \cdot \frac{D}{8} \cdot \frac{D0}{2}\right);$$

$$chD(1) = \frac{m2 \cdot D}{2} \cdot \mathrm{Cos}\left(-3 \cdot \pi \cdot \frac{D}{8} \cdot \frac{D0}{2}\right);$$

$$chD(2) = 0.$$

Координаты второй точки размерной линии диаметра D (рис. 2.1, т. 21):

$$fChD(0) = \frac{m2 \cdot D}{2} \cdot \mathrm{Sin}\left(-3 \cdot \pi \cdot \frac{D}{8} \cdot \frac{D0}{2}\right);$$

$$fChD(1) = -\frac{m2 \cdot D}{2} \cdot \mathrm{Cos}\left(-3 \cdot \pi \cdot \frac{D}{8} \cdot \frac{D0}{2}\right);$$

$$fChD(2) = 0.$$

Координаты положения текста размера диаметра D (рис. 2.1, т. 22):

$$lD(0) = -\frac{scalefactor2 \cdot D}{6};$$

$$lD(1) = \frac{scalefactor2 \cdot D0}{2} + 5;$$

$$lD(2) = 0.$$

На языке VBA:

```
'Размер диаметра D
Dim dimD As AcadDimDiametric
'Первая точка размерной линии
Dim chD(0 To 2) As Double
'Вторая точка размерной линии
Dim fchD(0 To 2) As Double
'Расстояние от первой точки размерной линии до текста размера
Dim lLD As Double
'Положение текста размера
Dim lD(0 To 2) As Double

' Определение координат и параметров диаметрального размера
chD(0) = -m2 * D / 2 * Sin(-3 * pi * D / 8 * D0 / 2)
chD(1) = m2 * D / 2 * Cos(-3 * pi * D / 8 * D0 / 2)
chD(2) = 0#

fchD(0) = m2 * D / 2 * Sin(-3 * pi * D / 8 * D0 / 2)
fchD(1) = -m2 * D / 2 * Cos(-3 * pi * D / 8 * D0 / 2)
fchD(2) = 0#
lLD = 3

'Нанесение размера диаметра D
Set dimD = ThisDrawing.ModelSpace.AddDimDiametric(chD, fchD, lLD)

'толщина линии 0,5 мм
```

```
dimD.Lineweight = acLnWt050

'Установка масштаба линейного размера (простановка реального
размера диметра D при масштабе отличающимся от 1:1)
dimD.TextMovement = acMoveTextNoLeader

lD(0) = -m2 * D / 6
lD(1) = m2 * D0 / 2 + 5
lD(2) = 0

dimD.TextPosition = lD

'Установка масштаба линейного размера (простановка реального
размера диметра D при масштабе отличающимся от 1:1)
dimD.LinearScaleFactor = 1 / m2
```

2.10. РАЗМЕР ШИРИНЫ ШПОНКИ

Размер ширины шпонки выполняется созданием объекта **AcadDimAligned** (линейный параллельный размер) с помощью метода **AddDimAligned** (рис. 2.4).

Общая команда построения линейного параллельного размера:

```
RetVal = object.AddDimAligned(ExtLine1Point, ExtLine2Point,
TextPosition),
```

где **RetVal** – название создаваемого объекта DimAligned (линейный параллельный размер);

Object – объект или объекты, к которым применяется метод AddDimAligned;

ExtLine1Point – одномерный массив, состоящий из трех элементов, представляющий собой координаты в трехмерном пространстве (X, Y, Z), определяющий первую точку выносной линии (точка, находящаяся на объекте, размер которого необходимо указать). Имеет тип Double;

ExtLine2Point – одномерный массив, состоящий из трех элементов, представляющий собой координаты в трехмерном пространстве (X, Y, Z), определяющий вторую точку выносной линии (точка, нахо-

дящаяся на объекте, размер которого необходимо указать). Имеет тип Double;

TextPosition – одномерный массив, состоящий из трех элементов, представляющий собой координаты в трехмерном пространстве (X, Y, Z), определяющий какую-либо из точек размерной линии. Имеет тип Double.

В линейных размерах размерная линия параллельна линии, соединяющей точки ExtLine1Point, ExtLine1Point.

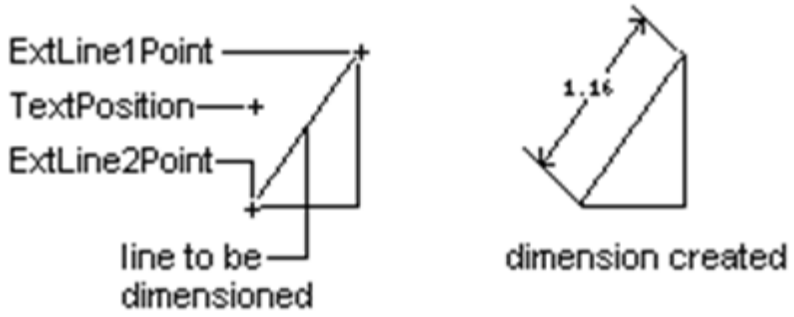


Рис. 2.4. Пояснительный рисунок
к применению метода AddDimAligned

Координаты первой точки выносной линии размера ширины шпонки (рис. 2.1, т. 23):

$$pbshp\_1(0) = -\frac{m2 \cdot bshp}{2};$$

$$pbshp\_1(1) = m2 \cdot \left(\frac{D0}{2} + (hshp - t - hxr1) \right);$$

$$pbshp\_1(2) = 0.$$

Координаты второй точки выносной линии размера ширины шпонки (рис. 2.1, т. 24):

$$pbshp\_2(0) = \frac{m2 \cdot bshp}{2};$$

$$pbshp\_2(1) = m2 \cdot \left(\frac{D0}{2} + (hshp - t - hxr1) \right);$$

$$pbshp\_2(2) = 0.$$

Координаты положения размерной линии ширины шпонки (рис. 2.1, т. 25):

$$lbshp(0) = -\frac{m2 \cdot bshp}{2};$$

$$lbshp(1) = m2 \cdot \left(\frac{D0}{2} + (hshp - t - hxr1) \right) + 10;$$

$$lb1(2) = 0.$$

На языке VBA:

```
If NalichieShponki = True Then

 'Размер ширины шпонки
 Dim dimbshp As AcadDimAligned
 'Первая точка выносной линии
 Dim pbshp_1(0 To 2) As Double
 'Вторая точка выносной линии
 Dim pbshp_2(0 To 2) As Double
 'Положение размерной линии
 Dim lbshp(0 To 2) As Double

 'Определение координат и параметров линейного размера
 pbshp_1(0) = -m2 * bshp / 2
 pbshp_1(1) = m2 * (D0 / 2 + (hshp - t - hxr1))
 pbshp_1(2) = 0#
 pbshp_2(0) = m2 * bshp / 2
 pbshp_2(1) = m2 * (D0 / 2 + (hshp - t - hxr1))
 pbshp_2(2) = 0#
 lbshp(0) = -m2 * bshp / 2
 lbshp(1) = m2 * (D0 / 2 + (hshp - t - hxr1)) + 10
 lbshp(2) = 0#

 'Нанесение размера ширины шпонки
 Set dimbshp = ThisDrawing.ModelSpace.AddDimAligned(pbshp_1,
pbshp_2, lbshp)
 'толщина линии 0,5 мм
 dimbshp.Lineweight = acLnWt050
 'Установка масштаба линейного размера (простановка реального
размера ширины шпонки при масштабе отличающимся от 1:1)
 dimbshp.LinearScaleFactor = 1 / m2
End If
```

2.11. РАЗМЕР ВЫСОТА ОТВЕРСТИЯ ПОД ВАЛ (ПРИ НАЛИЧИИ ШПОНОЧНОЙ КАНАВКИ)

Координаты первой точки выносной линии размера высота отверстия под вал (рис. 2.1, т. 26):

$$phshp\_1(0) = 0;$$

$$phshp\_1(1) = m \cdot \left(\frac{D0}{2} + (hshp - t - hxr1) \right);$$

$$phshp\_1(2) = 0.$$

Координаты второй точки выносной линии размера высота отверстия под вал (рис. 2.1, т. 27):

$$phshp\_2(0) = 0;$$

$$phshp\_2(1) = 0;$$

$$phshp\_2(2) = 0.$$

Координаты положения размерной линии (рис. 2.1, т. 28):

$$lhshp(0) = m2 \cdot \left(\frac{D0}{2} + 10 \right);$$

$$lhshp(1) = 0;$$

$$lhshp(2) = 0.$$

На языке VBA:

```
If NalichieShponki = True Then

 'Размер радиуса под вал+высота шпонки
 Dim dimhshp As AcadDimAligned
 'Первая точка выносной линии
 Dim phshp_1(0 To 2) As Double
 'Вторая точка выносной линии
 Dim phshp_2(0 To 2) As Double
 'Положение размерной линии
 Dim lhshp(0 To 2) As Double

 'Определение координат и параметров линейного размера
```

```
 phshp_1(0) = 0
 phshp_1(1) = m2 * (D0 / 2 + (hshp - t - hxr1))
 phshp_1(2) = 0#
 phshp_2(0) = 0
 phshp_2(1) = 0
 phshp_2(2) = 0#
 lhshp(0) = m2 * (D0 / 2 + 10)
 lhshp(1) = 0
 lhshp(2) = 0#

 'Нанесение размера радиуса под вал+высота шпонки
 Set dimhshp = ThisDrawing.ModelSpace.AddDimAligned(phshp_1,
phshp_2, lhshp)
 'толщина линии 0,5 мм
 dimhshp.Lineweight = acLnWt050
 'Установка масштаба линейного размера (простановка реального
размера радиуса под вал+высота шпонки при масштабе отличающимся от
1:1)
 dimhshp.LinearScaleFactor = 1 / m2
 'Округляем размер до десятых
 dimhshp.PrimaryUnitsPrecision = acDimPrecisionOne

End If
```

В тексте программы встречается свойство PrimaryUnitsPrecision, которое позволяет отображать необходимое количество знаков после запятой.

Общая команда отображения числа знаков после запятой в аннотационном тексте размерных линий:

```
object.PrimaryUnitsPrecision,
```

где **Object** – объекты типа DimAligned, DimDiametric, DimOrdinate, DimRadial, DimRotated, Tolerance, к которым применяется свойство LinearScaleFactor;

PrimaryUnitsPrecision – свойство, замещающее величину реального размера. Новая величина размера будет округлена в зависимости от константы, присвоенной данному свойству. В табл. 2.2 приведены возможные варианты констант.

Таблица 2.2

Значения свойства PrimaryUnitsPrecision

Значение свойства	Кол-во знаков
acDimPrecisionZero	0.
acDimPrecisionOne	0.0
acDimPrecisionTwo	0.00
acDimPrecisionThree	0.000
acDimPrecisionFour	0.0000
acDimPrecisionFive	0.00000
acDimPrecisionSix	0.000000
acDimPrecisionSeven	0.0000000
acDimPrecisionEight	0.00000000

2.12. РАЗМЕР ШИРИНЫ ШЛИЦА

Координаты первой точки выносной линии размера ширины шлица (рис. 2.1, т. 29):

$$pbsh\_1(0) = -\frac{m \cdot bsh}{2};$$

$$pbsh\_1(1) = -\frac{D}{2} \cdot m2 - k - m \cdot 5 \cdot \frac{m-1}{m} - 5 - m \cdot hxr;$$

$$pbsh\_1(2) = 0.$$

Координаты второй точки выносной линии размера ширины шлица (рис. 2.1, т. 30):

$$pbsh\_2(0) = \frac{m \cdot bsh}{2};$$

$$pbsh\_2(1) = -\frac{D}{2} \cdot m2 - k - m \cdot 5 \cdot \frac{m-1}{m} - 5 - m \cdot hxr;$$

$$pbsh\_2(2) = 0.$$

Координаты положения размерной линии ширины шлица (рис. 2.1, т. 31):

$$lbsh(0) = -\frac{m \cdot bsh}{2};$$

$$lbsh(1) = -\frac{D}{2} \cdot m2 - k - m \cdot 5 \cdot \frac{m-1}{m} + 5;$$

$lbsh(2) = 0.$

На языке VBA:

```
'Размер ширины шлица
Dim dimbsh As AcadDimAligned
'Первая точка выносной линии
Dim pbsh_1(0 To 2) As Double
'Вторая точка выносной линии
Dim pbsh_2(0 To 2) As Double
'Положение размерной линии
Dim lbsh(0 To 2) As Double

'Определение координат и параметров линейного размера
pbsh_1(0) = -m * bsh / 2
pbsh_1(1) = -D / 2 * m2 - k - m * 5 * (m - 1) / m - 5 - m * hxr
pbsh_1(2) = 0#
pbsh_2(0) = m * bsh / 2
pbsh_2(1) = -D / 2 * m2 - k - m * 5 * (m - 1) / m - 5 - m * hxr
pbsh_2(2) = 0#
lbsh(0) = -m * bsh / 2
lbsh(1) = -D / 2 * m2 - k - m * 5 * (m - 1) / m + 5
lbsh(2) = 0#

'Нанесение размера ширины шлица
Set dimbsh = ThisDrawing.ModelSpace.AddDimAligned(pbsh_1, pbsh_2,
lbsh)
'толщина линии 0,5 мм
dimbsh.Lineweight = acLnWt050

'Установка масштаба линейного размера (простановка реального
размера ширины шлица при масштабе отличающимся от 1:1)
dimbsh.LinearScaleFactor = 1 / m
```

2.13. РАЗМЕР ВЫСОТЫ ШЛИЦА

Координаты первой точки выносной линии размера высоты шлица (рис. 2.1, т. 32):

$$phsh\_1(0) = \frac{m \cdot bsh}{2};$$

$$phsh\_1(1) = -\frac{D}{2} \cdot m2 - k - m \cdot 5 \cdot \frac{m-1}{m} - 5 - m \cdot hxr;$$

$$phsh\_1(2) = 0.$$

Координаты второй точки выносной линии размера высоты шлица (рис. 2.1, т. 33):

$$psh\_2(0) = \frac{m \cdot bsh}{2};$$

$$phsh\_2(1) = -\frac{D}{2} \cdot m2 - k - m \cdot 5 \cdot \frac{m-1}{m} - 5 - m \cdot (hsh + hxr + hxr2);$$

$$phsh\_2(2) = 0.$$

Координаты положения размерной линии высоты шлица (рис. 2.1, т. 34):

$$lhsh(0) = m \cdot r1 + 10;$$

$$lhsh(1) = -\frac{D}{2} \cdot m2 - k - m \cdot 5 \cdot \frac{m-1}{m} - 5 - m \cdot hxr;$$

$$lhsh(2) = 0.$$

На языке VBA:

```
'Размер высоты шлица
Dim dimhsh As AcadDimAligned
'Первая точка выносной линии
Dim phsh_1(0 To 2) As Double
'Вторая точка выносной линии
Dim phsh_2(0 To 2) As Double
'Положение размерной линии
Dim lhsh(0 To 2) As Double

'Определение координат и параметров линейного размера
phsh_1(0) = m * bsh / 2
phsh_1(1) = -D / 2 * m2 - k - m * 5 * (m - 1) / m - 5 - m * hxr
phsh_1(2) = 0#

phsh_2(0) = m * bsh / 2
phsh_2(1) = -D / 2 * m2 - k - m * 5 * (m - 1) / m - 5 - m * (hsh +
hxr + hxr2)
phsh_2(2) = 0#
lhsh(0) = m * r1 + 10
lhsh(1) = -D / 2 * m2 - k - m * 5 * (m - 1) / m - 5 - m * hxr
lhsh(2) = 0#
```

```
'Нанесение размера высоты шлица
Set dimhsh = ThisDrawing.ModelSpace.AddDimAligned(phsh_1, phsh_2,
lhsh)
'толщина линии 0,5 мм
dimhsh.Lineweight = acLnWt050
'Изменение текстовой строки (простановка реального размера высоты
шлица при масштабе отличающимся от 1:1)
dimhsh.TextOverride = CStr(hsh)
'Размер пишется над первой выносной линией
dimhsh.HorizontalTextPosition = acFirstExtensionLine
```

В тексте программы встречаются свойства **TextOverride** и **HorizontalTextPosition**. Ниже даётся описание этих свойств.

2.14. ЗАМЕНА АННОТАЦИОННОГО ТЕКСТА У РАЗМЕРНЫХ ЛИНИЙ

Команда:

```
object.TextOverride,
```

где **Object** – объекты типа Dim3PointAngular, DimAligned, DimAngular, DimDiametric, DimOrdinate, DimRadial, DimRotated, к которым применяется свойство TextOverride;

TextOverride – свойство, заменяющее аннотационный текст на любую текстовую строку. Текстовая строка не должна содержать больше 256 символов. Имеет тип String.

Текстовая строка замещает реальную величину размера. Можно вернуть реальную величину размера задав свойству TextOverride пустую строку (""). Также возможно добавить новый текст к существующей реальной величине размера, путем добавления к текстовой строке скобок (<>) спереди, либо сзади. Например, если реальный размер равен 3,5, а TextString = "<> мм", тогда проставленный размер будет выглядеть как «3,5 мм».

2.15. ГОРИЗОНТАЛЬНАЯ ПОЗИЦИЯ ТЕКСТА

Команда:

```
object.HorizontalTextPosition,
```

где **Object** – объекты типа Dim3PointAngular, DimAligned, DimAngular, DimDiametric, DimRadial, DimRotated, к которым применяется свойство HorizontalTextPosition;

HorizontalTextPosition – свойство, заменяющее положение аннотационного текста над размерной линией. Возможны следующие значения данного свойства:

– acHorzCentered – текст располагается по центру вдоль размерной линии между выносными линиями;

– **acFirstExtensionLine** – текст следует за первой выносной линией;

– **acSecondExtensionLine** – текст следует за второй выносной линией;

– **acOverFirstExtension** – текст располагается выше и выравнивается над первой выносной линией;

– **acOverSecondExtension** – текст располагается выше и выравнивается над второй выносной линией.

2.16. РАЗМЕР ВЫСОТЫ ПАЗА

Координаты первой точки выносной линии размера высоты паза (рис. 2.1, т. 35):

$$php\_1(0) = 0;$$

$$php\_1(1) = -\frac{D}{2} \cdot m2 - k - m \cdot 5 \cdot \frac{m-1}{m} - 5 - m \cdot (hsh + hxr + hxr2);$$

$$php\_1(2) = 0.$$

Координаты второй точки выносной линии размера высоты паза (рис. 2.1, т. 36):

$$php\_2(0) = 0;$$

$$php\_1(1) = -\frac{D}{2} \cdot m2 - k - m \cdot 5 \cdot \frac{m-1}{m} - 5 - m \cdot (hsh + hxr + r1 + h1 + r2);$$

$$php\_2(2) = 0.$$

Координаты положения размерной линии высоты паза (рис. 2.1, т. 37):

$$lhp(0) = m \cdot r1 + 10;$$

$$lhp(1) = -\frac{D}{2} \cdot m2 - k - m \cdot 5 \cdot \frac{m-1}{m} - 5 - m \cdot (hsh + hxr + hxr2);$$

$$lhp(2) = 0.$$

На языке VBA:

```
'Размер высоты паза
Dim dimhp As AcadDimAligned
'Первая точка выносной линии
Dim php_1(0 To 2) As Double
'Вторая точка выносной линии
Dim php_2(0 To 2) As Double
'Положение размерной линии
Dim locationhp(0 To 2) As Double

'Определение координат и параметров линейного размера
php_1(0) = 0
php_1(1) = -D / 2 * m2 - k - m * 5 * (m - 1) / m - 5 - m * (hsh +
hxr + hxr2)
php_1(2) = 0#
php_2(0) = 0
php_2(1) = -D / 2 * m2 - k - m * 5 * (m - 1) / m - 5 - m * (hsh +
hxr + r1 + h1 + r2)
php_2(2) = 0#
lhp(0) = m * r1 + 10
lhp(1) = -D / 2 * m2 - k - m * 5 * (m - 1) / m - 5 - m * (hsh +
hxr + hxr2)
lhp(2) = 0#

'Нанесение размера высоты паза
Set dimhp = ThisDrawing.ModelSpace.AddDimAligned(php_1, php_2,
lhp)
'толщина линии 0,5 мм
dimhp.Lineweight = acLnWt050
```

```
'Установка масштаба линейного размера (простановка реального
размера высоты паза при масштабе отличающимся от 1:1)
dimhp.LinearScaleFactor = 1 / m
'Округляем размер до десятых
dimhp.PrimaryUnitsPrecision = acDimPrecisionOne
```

2.17. УГЛОВОЙ РАЗМЕР МЕЖДУ СОСЕДНИМИ ПАЗАМИ

Угловой размер выполняется созданием объекта **AcadDim3PointAngular** с помощью метода **AddDim3PointAngular**.

Общая команда построения углового размера:

```
RetVal = object.AddDimAngular(AngleVertex, FirstEndPoint,
                SecondEndPoint, TextPoint),
```

где **RetVal** – название создаваемого объекта DimAngular (угловой размер);

Object – объект или объекты, к которым применяется метод **AddDimAngular** (рис. 2.5);

AngleVertex – одномерный массив, состоящий из трех элементов, представляющий собой координаты в трехмерном пространстве (X, Y, Z), определяющий центр окружности, дуги или вершина двух пересекающихся выносных линий. Имеет тип Double;

FirstEndPoint – одномерный массив, состоящий из трех элементов, представляющий собой координаты в трехмерном пространстве (X, Y, Z), определяющий точку, через которую пройдет первая выносная линия. Имеет тип Double;

SecondEndPoint – одномерный массив, состоящий из трех элементов, представляющий собой координаты в трехмерном пространстве (X, Y, Z), определяющий точку, через которую пройдет вторая выносная линия. Имеет тип Double;

TextPoint – одномерный массив, состоящий из трех элементов, представляющий собой координаты в трехмерном пространстве (X, Y, Z), определяющий какую-либо из точек размерной линии. Имеет тип Double.

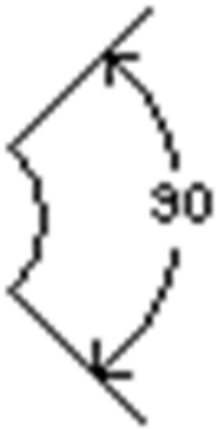


Рис. 2.5. Пояснительный рисунок к применению метода AddDimAngular

AngleVertex может быть равной **FirstEndPoint** или **SecondEndPoint**. Если необходимы выносные линии, они будут добавлены автоматически. **FirstEndPoint** и **SecondEndPoint** используются как начальные точки для выносных линий.

Координаты вершины угла, образованного двумя размерными линиями (рис. 2.1, т. 38):

$$AV(0) = 0:$$

$$AV(1) = 0:$$

$$AV(2) = 0$$

Координаты первой точки углового размера между соседними пазами (рис. 2.1, т. 39):

$$FP(0) = \frac{m2 \cdot D}{2} \cdot \mathrm{Sin}\left(\frac{360 / Z \cdot \pi}{180}\right);$$

$$FP(1) = \frac{m2 \cdot D}{2} \cdot \mathrm{Cos}\left(\frac{360 / Z \cdot \pi}{180}\right);$$

$$FP(2) = 0.$$

Координаты второй точки углового размера между соседними пазами (рис. 2.1, т. 40):

$$SP(0) = 0;$$

$$SP(1) = \frac{m2 \cdot D}{2};$$

$$SP(2) = 0.$$

Координаты точки позиции текста углового размера (рис. 2.1, т. 41):

$$TP(0) = \frac{m2 \cdot D}{2} \cdot \mathrm{Sin}\left(\frac{360 / Z \cdot \pi}{180}\right);$$

$$TP(1) = \frac{m2 \cdot D}{2} + 10;$$

$$TP(2) = 0.$$

На языке VBA:

```
'Угловой размер между соседними пазами
Dim DimPointAngularObj As AcadDim3PointAngular
'Вершина угла, образованная двумя размерными линиями
Dim AV(0 To 2) As Double
'Первая точка углового размера
Dim FP(0 To 2) As Double
'Вторая точка углового размера
Dim SP(0 To 2) As Double
'Точка позиции текста углового размера
Dim TP(0 To 2) As Double
'Dim Line1Suppressed As String, Line2Suppressed As String

'Определение координат и параметров углового размера
AV(0) = 0: AV(1) = 0: AV(2) = 0
FP(0) = D * m2 / 2 * Sin((360 / Z) * radian): FP(1) = D * m2 / 2 *
Cos((360 / Z) * radian): FP(2) = 0
SP(0) = 0: SP(1) = D * m2 / 2: SP(2) = 0
TP(0) = D * m2 / 2 * Sin((360 / Z) * radian): TP(1) = D * m2 / 2 +
10: TP(2) = 0

'Нанесение углового размера между соседними пазами
Set DimPointAngularObj =
ThisDrawing.ModelSpace.AddDim3PointAngular(AV, FP, SP, TP)
'толщина линии 0,5 мм
DimPointAngularObj.Lineweight = acLnWt050
```

2.18. ВЫНОСКА ДЛЯ УКАЗАНИЯ ЧИСЛА ПАЗОВ

Точки выноски показаны на рис. 2.1 (т.т. 42-43-44).

Выноска выполняется созданием объекта **AcadLeader** с помощью метода **AddLeader**.

Общая команда создания выноски:

```
RetVal = object.AddLeader(PointsArray, Annotation, Type),
```

где **RetVal** – название создаваемого объекта Leader (выноска);

Object – объект или объекты, к которым применяется метод AddLeader;

PointsArray – одномерный массив, представляющий собой координаты в трехмерном пространстве (X, Y, Z), определяющий точки через которые проходит выноска. Имеет тип Double. Для создания выноски необходимо указать как минимум две точки. Третья точка необязательна;

Annotation – аннотационный текст выноски. Имеет тип AcadMText;

Type – тип указателя. Может принимать следующие значения:

– **acLineNoArrow** – линия без стрелки;

– **acLineWithArrow** – линия со стрелкой;

– **acSplineNoArrow** – сплайн без стрелки;

– **acSplineWithArrow** – сплайн со стрелкой.

Выноска – линия, соединяющая аннотационный текст и объекты на рисунке (рис. 2.6).

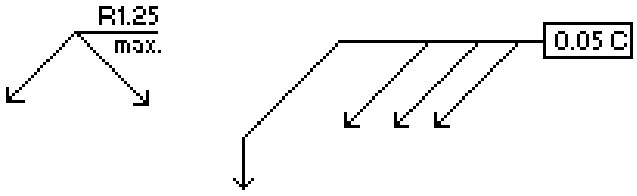


Рис. 2.6. Пример выносок

Координаты точек выносной линии:

$$points8(0) = -\frac{m2 \cdot r1}{2};$$

$$points8(1) = \frac{m2 \cdot D}{2} - m2 \cdot (hxr + hsh + r1);$$

$$points8(2) = 0;$$

$$points8(3) = -10;$$

$$points8(4) = \frac{m2 \cdot D}{2} + 10;$$

$$points8(5) = 0.$$

Над выноской должен располагаться текст, показывающий число пазов. Стоит обратить внимание, что фразы «21 паз», «32 паза», «28 пазов» имеют различную длину. Поэтому линия выноски, над которой размещается данный текст, также должна иметь различную длину.

Если Z = 1, 21, 31, 41, 51, 61, 71, 81, 91:

$$points8(6) = -38;$$

$$points8(7) = \frac{m2 \cdot D}{2} + 10;$$

$$points8(8) = 0.$$

Если Z = 2, 3, 4, 22, 23, 24, 32, 33, 34, 42, 43, 44, 52, 53, 54, 63, 63, 64, 72, 73, 74, 82, 83, 84, 92, 93, 94:

$$points8(6) = -45;$$

$$points8(7) = \frac{m2 \cdot D}{2} + 10;$$

$$points8(8) = 0.$$

Во всех остальных случаях:

$$points8(6) = -50;$$

$$points8(7) = \frac{m2 \cdot D}{2} + 10;$$

$points8(8) = 0.$

```
'Выноска для указания числа пазов
Dim leaderVinoska As AcadLeader
'Координаты точек выносной линии
Dim points8(0 To 8) As Double
'Тип указателя
Dim leaderType As Integer
'Пояснительный текст на выноске(указывается после нанесения
выноски)
Dim annotationObject As AcadMText

points8(0) = -r1 * m2 / 2: points8(1) = D * m2 / 2 - m2 * (hxr +
hsh + r1): points8(2) = 0
points8(3) = -10: points8(4) = D * m2 / 2 + 10: points8(5) = 0

'Тип указателя - стрелка
leaderType = acLineWithArrow

'Первоначально текст на выноске отсутствует
Set annotationObject = Nothing

'Конструкция, которая определяет длину выносной линии над которой
располагается пояснительный текст.Зависит от слов:
"паз","пазов","паза"
If Z = 1 Or Z = 21 Or Z = 31 Or Z = 41 Or Z = 51 Or Z = 61 Or Z =
71 Or Z = 81 Or Z = 91 Then
 points8(6) = -38: points8(7) = D * m2 / 2 + 10: points8(8) = 0
ElseIf Z = 2 Or Z = 3 Or Z = 4 Or Z = 22 Or Z = 23 Or Z = 24 Or Z
= 32 Or Z = 33 Or Z = 34 Or Z = 42 Or Z = 43 Or Z = 44 Or Z = 52
Or Z = 53 Or Z = 54 Or Z = 63 Or Z = 63 Or Z = 64 Or Z = 72 Or Z =
73 Or Z = 74 Or Z = 82 Or Z = 83 Or Z = 84 Or Z = 92 Or Z = 93 Or
Z = 94 Then
 points8(6) = -45: points8(7) = D * m2 / 2 + 10: points8(8) = 0
Else
 points8(6) = -50: points8(7) = D * m2 / 2 + 10: points8(8) = 0
End If

 ' Нанесение выноски
Set leaderVinoska = ThisDrawing.ModelSpace.AddLeader(points8,
annotationObject, leaderType)
'толщина линии 0,5 мм
leaderVinoska.Lineweight = acLnWt050
```

2.19. ТЕКСТ НАД ВЫНОСНОЙ ЛИНИЕЙ

Выноска выполняется созданием объекта **AcadMText** (Многострочный текст) с помощью метода **AddMText**.

Общая команда создания многострочного текста:

```
RetVal = object.AddMText(InsertionPoint, Width, Text),
```

где **RetVal** – название создаваемого объекта типа MText;

Object – объект или объекты типа ModelSpace Collection, PaperSpace Collection, Block, к которым применяется метод AddMText;

AddMText – метод, создающий объект MText в прямоугольной области, определенной точкой вставки (InsertionPoint) и шириной (Width) ограничивающего прямоугольника;

InsertionPoint – одномерный массив, состоящий из трех элементов, представляющий собой координаты в трехмерном пространстве (X, Y, Z), определяющий точку вставки ограничивающего прямоугольника объекта Mtext. Имеет тип Double. Режим: только запись (input-only);

Width – ширина ограничивающего прямоугольника объекта Mtext. Имеет тип Double. Режим: только запись (input-only);

Text – определенная текстовая строка для объекта Mtext. Имеет тип String. Режим: только запись (input-only).

Координаты левой верхней точки начала пояснительного текста над выносной линией (рис. 2.1, т. 45):

Если Z = 1, 21, 31, 41, 51, 61, 71, 81, 91:

$$corner(0) = -37;$$

$$corner(1) = \frac{m2 \cdot D}{2} + 16;$$

$$corner(2) = 0.$$

Если Z = 2, 3, 4, 22, 23, 24, 32, 33, 34, 42, 43, 44, 52, 53, 54, 63, 63, 64, 72, 73, 74, 82, 83, 84, 92, 93, 94:

$$corner(0) = -44;$$

$$corner(1) = \frac{m2 \cdot D}{2} + 16;$$

$$corner(2) = 0.$$

Во всех остальных случаях:

$$corner(0) = -49;$$

$$corner(1) = \frac{m2 \cdot D}{2} + 16;$$

$$corner(2) = 0.$$

На языке VBA:

```
'Текст над выносной линией
Dim MTextObjVinoska As AcadMText
'Dim insertionPointVinoska(0 To 2) As Double
'Левая верхняя точка начала пояснительного текста
Dim corner(0 To 2) As Double
'Ширина пояснительного текста
Dim width As Double
'Пояснительный текст
Dim text As String
'Высота текста
Dim height As Double
'Конструкция, которая определяет текст над выносной линией. Текст
включает в себя число пазов и
'одно из следующих слов: "паз","пазов","паза"
If Z = 1 Or Z = 21 Or Z = 31 Or Z = 41 Or Z = 51 Or Z = 61 Or Z =
71 Or Z = 81 Or Z = 91 Then
     text = CStr(Z) & " паз"
     corner(0) = -37: corner(1) = D * m2 / 2 + 16: corner(2) = 0
     width = 27
ElseIf Z = 2 Or Z = 3 Or Z = 4 Or Z = 22 Or Z = 23 Or Z = 24 Or Z
= 32 Or Z = 33 _
Or Z = 34 Or Z = 42 Or Z = 43 Or Z = 44 Or Z = 52 Or Z = 53 Or Z =
54 Or Z = 62 _
```

```
Or Z = 63 Or Z = 64 Or Z = 72 Or Z = 73 Or Z = 74 Or Z = 82 Or Z =
83 Or Z = 84 _
Or Z = 92 Or Z = 93 Or Z = 94 Then
     text = CStr(Z) & " паза"
     corner(0) = -44: corner(1) = D * m2 / 2 + 16: corner(2) = 0
     width = 35
Else
     text = CStr(Z) & " пазов"
     corner(0) = -49: corner(1) = D * m2 / 2 + 16: corner(2) = 0
     width = 40
End If

'Нанесение текста над выноской
Set annotationObject = ThisDrawing.ModelSpace.AddMText(corner,
width, text)
'толщина линии 0,5 мм
annotationObject.Lineweight = acLnWt050
'Меняем высоту текста
annotationObject.height = 5
```

В тексте программы встречается свойство height, которое отвечает за высоту объекта. В нашем случае это свойство отвечает за высоту многострочного текста и равно 5мм.

Свойство Высота объекта:

```
object.Height,
```

где **object** – объект или объекты типа Application, Attribute, AttributeReference, MText, PViewport, Raster, Shape, Text, TextStyle, Toolbar, Viewport, View, к которым применяется свойство Height;

Height – свойство, изменяющее высоту объекта. Имеет тип Double. Должно быть положительной величиной.

Для объекта Application свойство определяет высоту окна приложения и измеряется в пикселях.

Для объектов MText, Text свойство определяет высоту строчного шрифта и измеряется в текущих единицах, может использоваться совместно со свойством scalefactor для определения не только высоты, но и ширины объекта.

Для объекта Raster – определяет высоту растрового рисунка и измеряется в пикселях.

2.20. ТЕКСТ МАСШТАБА ФРАГМЕНТА

Координаты начальной точки текста масштаба фрагмента (рис. 2.1, т. 46):

$$insertionPointMashtab(0) = -10;$$

$$insertionPointMashtab(1) = -\frac{m2 \cdot D}{2} - 20;$$

$$insertionPointMashtab(2) = 0.$$

Координаты конечной точки текста масштаба фрагмента (рис. 2.1, т. 47):

$$alignmentPointMashtab(0) = 10;$$

$$alignmentPointMashtab(0) = -\frac{m2 \cdot D}{2} - 20;$$

$$alignmentPointMashtab(2) = 0.$$

На языке VBA:

```
'Текст масштаба фрагмента
Dim textObjMashtab As AcadText
'Точка начала текста масштаба фрагмента
Dim insertionPointMashtab(0 To 2) As Double
'Конечная точка текста масштаба фрагмента
Dim alignmentPointMashtab(0 To 2) As Double
'Текстовая строка, которая содежит масштаб фрагмента листа якоря
Dim textString As String
'Определение координат
insertionPointMashtab(0) = -10: insertionPointMashtab(1) = -D / 2
* m2 - 20: insertionPointMashtab(2) = 0
alignmentPointMashtab(0) = 10: alignmentPointMashtab(1) = -D / 2 *
m2 - 20: alignmentPointMashtab(2) = 0
'В зависимости от m меняется текстовая строка
If m = 1 / 2 Then
     textString = "A(1:2)"
ElseIf m = 1 / 2.5 Then
```

```
     textString = "A(1:2,5)"
ElseIf m = 1 Then
     textString = "A(1:1)"
ElseIf m = 2 Then
     textString = "A(2:1)"
ElseIf m = 2.5 Then
     textString = "A(2,5:1)"
ElseIf m = 4 Then
     textString = "A(4:1)"
ElseIf m = 5 Then
     textString = "A(5:1)"
End If
'Нанесение масштаба фрагмента
Set textObjMashtab = ThisDrawing.ModelSpace.AddText(textString,
insertionPointMashtab, 5)
'толщина линии 0,5 мм
textObjMashtab.Lineweight = acLnWt050
'Выравнивание текста по ширине (acAlignmentFit - ошибочно. Fit по
английски - вписанный, а делает выравнивание по ширине)
textObjMashtab.Alignment = acAlignmentFit
'Точка выравнивания текста
textObjMashtab.TextAlignmentPoint = alignmentPointMashtab
'Устанавливаем наклон текста - 15°
textObjMashtab.ObliqueAngle = 15 * radian
'Устанавливаем стиль текста - "RomanS"
textObjMashtab.StyleName = "RomanS"
```

В тексте фрагмента программы встречаются следующие свойства: Alignment, TextAlignmentPoint, ObliqueAngle, StyleName. Ниже даны описания этих свойств.

2.21. ВЫРАВНИВАНИЕ ТЕКСТА

Команда выравнивания текста:

```
object.Alignment,
```

где **object**– объект или объекты типа Attribute, AttributeRef, Text, к которым применяется свойство Alignment;

Alignment – свойство, определяющее вертикальное и горизонтальное выравнивание объекта (рис. 2.7). Может принимать следующие значения:

– **acAlignmentLeft** – выравнивание по левому краю;

– **acAlignmentCenter** – выравнивание по центру;

– **acAlignmentRight** – выравнивание по правому краю;

– **acAlignmentAligned** – выравнивание по ширине;

– **acAlignmentMiddle** – выравнивание по центру;

– **acAlignmentFit** – вписанный;

– **acAlignmentTopLeft** – выравнивание по левой верхней точке;

– **acAlignmentTopCenter** – выравнивание по верхней центральной точке;

– **acAlignmentTopRight** – выравнивание по верхней правой точке;

– **acAlignmentMiddleLeft** – выравнивание по средней левой точке;

– **acAlignmentMiddleCenter** – выравнивание по средней центральной точке;

– **acAlignmentMiddleRight** – выравнивание по средней правой точке;

– **acAlignmentBottomLeft** – выравнивание по нижней левой точке;

– **acAlignmentBottomCenter** – выравнивание по нижней центральной точке;

– **acAlignmentBottomRight** – выравнивание по нижней правой точке.

Текст, выровненный по **acAlignmentLeft**, использует свойство InsertionPoint для определения позиции текста.

Текст, выровненный по **acAlignmentAligned**, или **acAlignmentFit**, использует свойства **InsertionPoint** и **TextAlignmentPoint** одновременно для определения позиции текста.

Текст, выровненный по любой другой позиции, использует свойство **TextAlignmentPoint** для определения позиции текста.

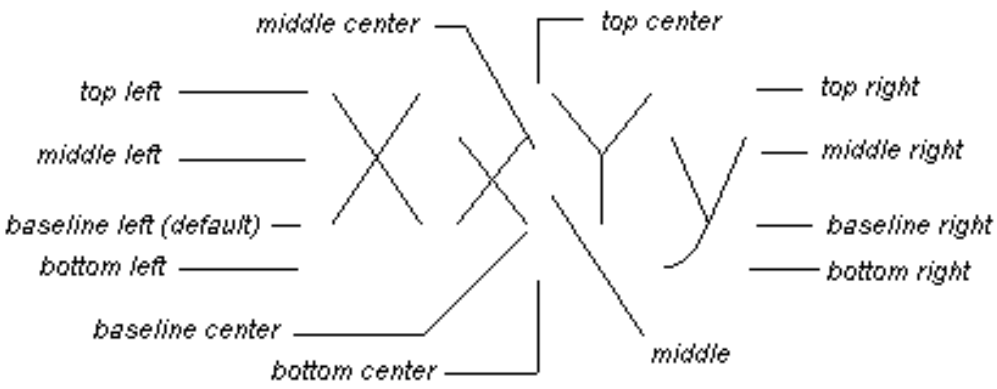


Рис. 2.7. Пояснительный рисунок
к применению свойства Alignment

2.22. ТОЧКА ВЫРАВНИВАНИЯ ТЕКСТА

Команда:

```
object.TextAlignmentPoint,
```

где **object** – объект или объекты типа Attribute, AttributeRef, Text, к которым применяется свойство TextAlignmentPoint;

TextAlignmentPoint – одномерный массив, состоящий из трех элементов, представляющий собой координаты в трехмерном пространстве (X, Y, Z), представляющий точку выравнивания объекта.

Для объекта текст Text свойство будет установлено (0, 0, 0) и будет доступно только в режиме чтения (read-only), когда свойство Alignment равно acAlignmentLeft. Позиция текста, чье выравнивание равно left, fit, or aligned, использует свойство InsertionPoint.

2.23. УГОЛ НАКЛОНА ТЕКСТА

Команда:

```
object.ObliqueAngle,
```

где **object** – объект или объекты типа Attribute, AttributeReference, Shape, Text, TextStyle, к которым применяется свойство ObliqueAngle;

ObliqueAngle – свойство, определяющее наклон объекта (рис. 2.8). Угол измеряется в радианах и лежит в диапазоне от -85 до +85 градусов. Положительный угол означает наклон вправо; к отрицательному углу будет автоматически добавлена величина 2*PI, чтобы конвертировать в положительную величину, при этом наклон объекта будет влево.

Угол наклона измеряется относительно вертикальной оси.

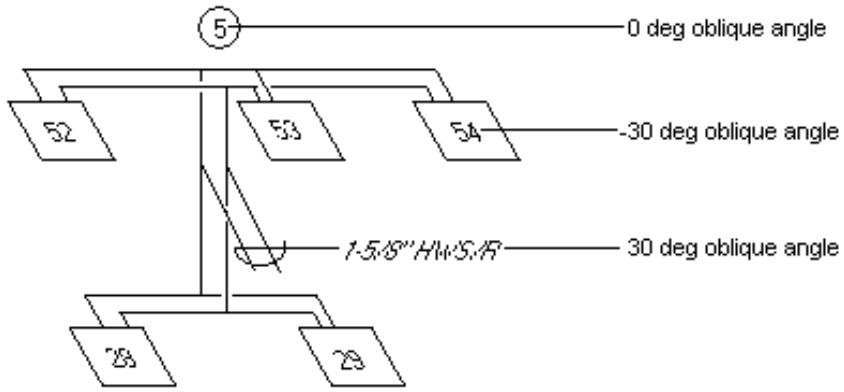


Рис. 2.8. Пояснительный рисунок к применению свойства ObliqueAngle

2.24. УСТАНОВКА СТИЛЯ ШРИФТА ТЕКСТА

Команда построения:

```
object.StyleName,
```

где **object** – объект или объекты типа Attribute, AttributeReference, DimAligned, Dim3PointAngular, DimAngular, DimDiametric, DimOrdinate, DimRadial, DimRotated, Leader, MLine, MText, Text, Tolerance, к которым применяется свойство ObliqueAngle;

StyleName – свойство, определяющее имя стиля, используемого в объекте. Имеет тип String. Режим: чтение-запись (read-write), для объ-

екта MLine – только чтение (read-only). По умолчанию приравнивается к текущему стилю.

Для изменения атрибутов заданного стиля текста нужно использовать объект TextStyle. Для изменения атрибутов заданного стиля размера – объект DimStyle.

Заданное имя должно быть заранее определено в чертеже.

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Используя команды редактирования объектов (копирование, зеркальное отражение, прямоугольный массив, круговой массив, перенос, вращение, масштаб, изменение координат точек), для двигателя серии 4А создайте программу, которая будет выполнять чертеж листа статора или ротора.

Условные обозначения:

δ – односторонний воздушный зазор между статором и ротором;

D_{a1} – внешний диаметр сердечника статора;

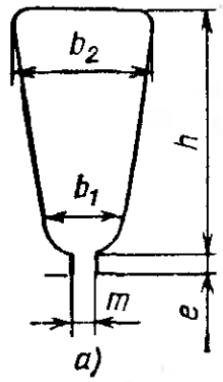
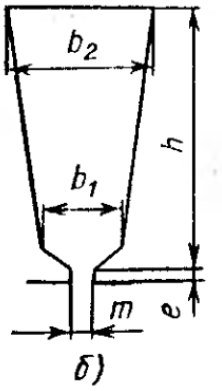
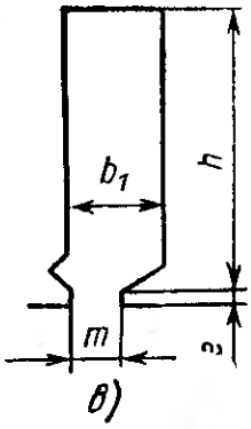
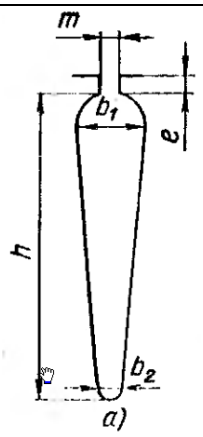
D_{i1} – внутренний диаметр сердечника статора.

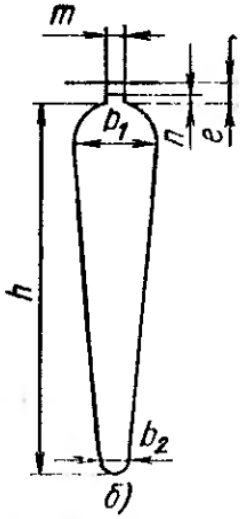
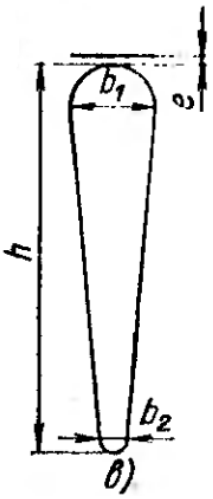
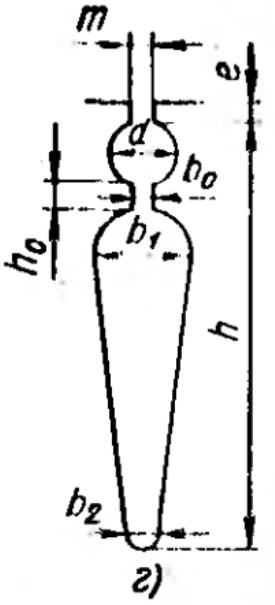
Наружный диаметр ротора рассчитывается по формуле

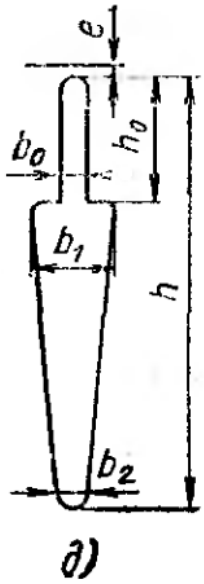
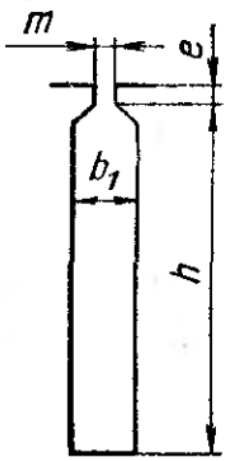
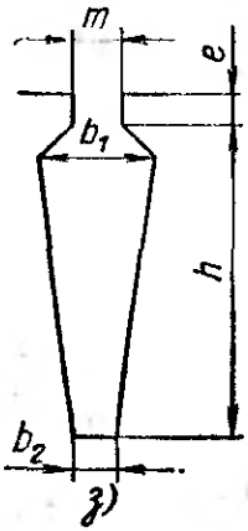
$$D_2 = D_{i1} \_ 2 \cdot \delta.$$

Варианты заданий

№	Вид чертежа	Вид паза	Размеры для справки, мм	Марка двигателя

1.	Лист статора		D_{a1}= 225 D_{i1} = 145 b_1 = 6,1 b_2 = 9,2 h = 17,8 e = 0,9 m = 3,5	4А132S4У3
2.	Лист статора		D_{a1}= 272 D_{i1} = 155 b_1 = 8,7 b_2 = 11,9 h = 20 e = 1 m = 4	4А160S2У3
3.	Лист статора		D_{a1}= 520 D_{i1} = 275 b_1 = 11,6 h = 45,9 e = 1,1 m = 6,4	4А28082У3
4.	Лист ротора		D_{a1}= 225 D_{i1} = 145 δ = 0,35 b_1 = 6 b_2 = 2,2 h = 24,7 e = 0,75 m = 1,5	4А132S4У3

№	Вид чертежа	Вид паза	Размеры для справки, мм	Марка двигателя
5.	Лист ротора		D_{a1}= 272 D_{i1} = 185 δ = 0,5 b_1 = 7,5 b_2 = 3,5 h = 34 e = 1 m = 1,5 n = 0,7	4А160М4У3
6.	Лист ротора		D_{a1}= 313 D_{i1} = 171 δ = 1 b_1 = 9,6 b_2 = 4,1 h = 31 e = 0,85	4А180М2У3
7.	Лист ротора		D_{a1}= 313 D_{i1} = 211 δ = 0,6 b_1 = 6,5 b_2 = 3,5 h = 36,8 e = 1 m = 1,5 h_0 = 3 b_0 = 2 d = 8,5	4А180М4/2У3

№	Вид чертежа	Вид паза	Размеры для справки, мм	Марка двигателя
8.	Лист ротора	 д)	D_{a1}= 520 D_{i1} = 275 δ = 1,3 b_1 = 9,3 b_2 = 6,5 h = 40 e = 0,5 h_0 = 15 b_0 = 5	4А315S2У3
9.	Лист ротора	 ж)	D_{a1}= 392 D_{i1} = 284 δ = 0,75 b_1 = 4,3 h = 40,2 e = 0,6 m = 1,5	4АНК225М8У3
10.	Лист ротора	 з)	D_{a1}= 272 D_{i1} = 185 δ = 0,5 b_1 = 9 b_2 = 4,5 h = 28,3 e = 1 m = 3,7	4НК160S4У3

ЗАКЛЮЧЕНИЕ

В данном пособии мы рассмотрели создание программы для автоматизированного построения листа якоря с овальными пазами. Предварительно необходимо было вывести координаты характерных точек, по которым и производилось построение листа якоря. Далее определялись программные объекты AutoCAD, с помощью которых выполнялись построения. Сами построения велись с помощью методов и свойств данных объектов. В пособии давались краткие описания этих методов и свойств. С некоторыми из них можно ознакомиться более подробно в [1]. При знании английского языка можно воспользоваться справочной системой VBA for AutoCAD, в которой кроме описания объектов, методов и свойств есть и хорошие примеры, на основе которых можно понять принцип автоматизированного создания графических объектов.

В заключение хочется сказать, что, используя возможности программирования, инженер может существенно упростить решение многих задач. И это касается не только автоматизированного создания чертежей, но и других областей автоматизации.

СПИСОК ЛИТЕРАТУРЫ

1. Минеев С.П. Основы программирования в AutoCAD. Технологии ActiveX Automation и VBA в среде проектирования AutoCAD для решения задач электромеханики: учеб. пособ. – Самара: Самар. гос. техн. ун-т, 2010.

2. Минеев С.П. VBA: Создание простейших программ, создание отчетов в документе Word: учеб. пособ. – Самара: Самар. гос. техн. ун-т, 2004.

3. Минеев С.П., Тулупов П.В., Макаричев Ю.А., Кальянова Ю.А. VBA: управление ходом программы, действия над массивами данных: учеб. пособ. – Самара: Самар. гос. техн. ун-т, 2005.

4. Справочная система AutoCAD.

СОДЕРЖАНИЕ

Учебное издание

МИНЕЕВ Сергей Петрович
ЗУБКОВ Юрий Валентинович

Автоматизация создания чертежей деталей электрических машин на языке VBA в среде проектирования AutoCAD

Редактор *Е.В. Абрамова*
Верстка *Ю.А. Петропольская*
Выпускающий редактор *Е.С. Захарова*

Подп. в печать 8.11.12
Формат 60×84 1/16. Бумага офсетная
Усл. п.л. 6,15. Уч.-изд. л. 6,1
Тираж 50 экз. Рег. № 158/12

Федеральное государственное бюджетное
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