



Федеральное агентство по образованию

ГОУ ВПО "Тульский государственный университет"

Технологический факультет



Кафедра "Автоматизированные станочные системы"

Курс "Разработка САПР"

# Разработка 2D-библиотек для КОМПАС

МЕТОДИЧЕСКИЕ УКАЗАНИЯ  
ДЛЯ ВЫПОЛНЕНИЯ ЛАБОРАТОРНЫХ РАБОТ  
СТУДЕНТАМИ СПЕЦИАЛЬНОСТИ 230104 САПР

ТУЛА 2007

Разработал к.т.н., доц. Троицкий Д.И.

Рассмотрено на заседании кафедры АСС

Зав. кафедрой АСС д.т.н., проф.

\_\_\_\_\_ Иноземцев А.Н.

## 1. Библиотеки в КОМПАСе

Для расширения возможностей САПР КОМПАС без изменения исходного кода в ней реализован механизм динамически подключаемых библиотек. Каждая библиотека – это файл с расширением `.rtw`. На самом деле это самая обычная `dll`-библиотека, у которой расширение `dll` заменено на `rtw`. Такую библиотеку легко написать, например, на Delphi. Она будет подключаться к КОМПАС так же, как и все прочие библиотеки, и будет иметь полный доступ к API КОМПАС для выполнения различных построений и манипуляций с графическими объектами и документами. Библиотека может иметь свою экранную форму, что резко упрощает ввод данных – можно использовать любые компоненты Delphi, подключить базы данных и т.д.

В большинстве случаев результатом работы библиотеки должно быть построение какого-либо фрагмента чертежа или 3D модели. Поскольку построение выполняется полностью программным путем, на получаемые модели не накладывается никаких ограничений – можно создать библиотеку, строящую зубчатое колесо с произвольным числом зубьев (вспомните Shaft 3D) и т.д.

Библиотеки разделяются по типу документа, в котором они производят построения. Для запуска 3D библиотеки необходимо, чтобы текущим документом была 3D модель или сборка, а для 2D библиотеки – чертеж или фрагмент. Сначала рассмотрим создание 2D библиотек.

В комплект поставки КОМПАС входит каталог SDK с файлом помощи `sdk.hlp`, содержащим минимум информации по объектам, процедурам и функциям API. Кроме того, в каталоге `SDK\Pascal\DelphiAuto` приведен ряд полезных примеров по созданию библиотек на Delphi.

## 2. Начнем сначала

Поставим задачу создания простейшей 2D библиотеки, которая выполняла бы следующие действия: выводила на экран форму для ввода четырех координат  $(x_1, y_1, x_2, y_2)$  и в текущем чертеже или фрагменте проводила бы отрезок из точки  $x_1, y_1$  в точку  $x_2, y_2$ . В Delphi создаем новый проект типа "DLL library" (вспоминаем курс "Программирование на языке высокого уровня"). В файле `.dpr` такого проекта содержится заголовок библиотеки со списком экспортируемых процедур. Для того, чтобы КОМПАС подключил нашу библиотеку, она должна экспортировать следующие процедуры и функции:

`LibraryName` - функция, возвращающая текстовое название библиотеки;

`LibraryId` – функция, возвращающая целый идентификатор библиотеки;

`LibraryEntry` – точка входа в библиотеку, или, проще говоря, процедура, выполняемая при запуске библиотеки.

Чтобы создаваемая библиотека автоматически получила расширение rtw, а не dll, надо использовать директиву компилятора {\$E rtw}. Тогда dpr-файл нашего проекта будет иметь вид:

```
library chanel;  
  
{$E rtw}  
  
uses  
    SysUtils,    Classes,    KsTLB;  
  
exports  
    LibraryName name 'LIBRARYNAME',  
    LibraryId name 'LIBRARYID',  
    LibraryEntry name 'LIBRARYENTRY';  
  
begin  
end.
```

Как видно, в операторе USES подключается модуль KsTLB, обеспечивающий взаимодействие с API КОМПАС. Этот файл (а также другие необходимые для компиляции модули) находится в каталоге Program Files\Ascon\KOMPAS\SDK\Include. Необходимо сказать Delphi, чтобы поиск файлов при компиляции выполнялся и в вышеуказанном каталоге. Идем в меню Tools→Environment Options→Library→Library Path и добавляем в список путей поиска файлов каталог Include.

Создаем новый модуль (File→New→Unit), сохраняем его (например, под именем main). В этом модуле мы должны описать реализацию процедур и функций LibraryName, LibraryId, LibraryEntry.

```
unit main;  
  
interface  
  
uses Windows, SysUtils, LDefin2D, ksConstTLB, ksAuto,  
    ksTLB, Forms;  
  
    // заголовки  
    procedure LIBRARYENTRY(command: WORD); Pascal;  
    function LIBRARYNAME : PChar; Pascal;  
    function LIBRARYID : Cardinal; Pascal;  
    // ссылки на КОМПАС и 2D документ
```

```

var
  Kompas: KompasObject;
  iDocument2D: ksDocument2D;

implementation

function LIBRARYNAME: PChar; pascal;
begin
  // произвольное название библиотеки
  Result := 'Библиотека построения отрезков'
end;

function LIBRARYID: UINT; pascal;
begin
  // произвольный идентификатор от 100 и выше
  Result := 100;
end;

procedure LIBRARYENTRY (command:WORD); pascal;
begin
  // связываем переменную Kompas с API КОМПАСа
  Kompas := KompasObject(CreateKompasObject);
  // если все в порядке...
  if Kompas <> nil then
    begin
      // связываем переменную iDocument2D
      // с текущим 2D документом
      iDocument2D := ksDocument2D(Kompas.ActiveDocument2D());
      if iDocument2D=nil then
        // сообщение об ошибке
        Kompas.ksMessage('Текущий документ не является
чертежом или фрагментом')
      else
        begin
          // своя процедура запуска библиотеки - см. ниже
          Run;
          Kompas.ksMessage('Работа библиотеки завершена')
        end;
      // освобождение памяти
      iDocument2D := nil;
    end;
    Kompas := nil
  end;
end;

```

end.

Процедура `LibraryEntry` может получать на вход целочисленный параметр, задействованный при вызове библиотеки из пунктов меню или с панели инструментов. Нам он пока не понадобится. Внутри процедуры выполняется проверка – является ли текущий документ 2D-документом. Обратите внимание, что мы пока не можем пользоваться никакими средствами Delphi для ввода-вывода, поэтому сообщения выводятся вызовом метода `ksMessage` объекта `Kompas`, т.е. средствами самого КОМПАСа.

## 2.1. Экранные формы в библиотеках

Пока библиотека ничего полезного не делает – у нее нет даже формы. Создадим форму (`File→New→Form`) и сохраним новый модуль под именем, например, `former.pas`. Самой форме дадим имя `MainForm`. Поместим на нее четыре компонента типа `TEdit`, кнопку "Отрезок" и кнопку "Готово". Сохраним проект.

Как и в любой другой библиотеке, форма не создается автоматически – ее надо вручную создать и отобразить. Кроме того, на время работы с формой надо отключить доступ к элементам интерфейса КОМПАС, иначе неизбежны попытки одновременного рисования в документе и пользователя, и библиотеки. Напишем процедуру `Run` в модуле `main`, которая все это выполняет. Разумеется, в оператор `USES` надо добавить ссылку на модуль с формой (`former`) и библиотеку работы с формами (`forms`).

```
procedure Run; // своя процедура запуска

var
  form : TMainForm; // ссылка на форму
begin
  // делаем нашу форму дочерней формой КОМПАСа
  Application.Handle := kompas.ksGetHWND;
  // закрыть доступ к компасу
  kompas.ksEnableTaskAccess(0);
  // создаем форму
  form:=TMainForm.Create(Application);
  // вывод формы
  form.ShowModal;
  // открыть доступ к компасу
  kompas.ksEnableTaskAccess(1);
  // отключаем нашу форму от КОМПАСа
  Application.Handle := 0
```

end;

Форма будет выводиться в *модальном режиме*. Это означает, что она заблокирует все остальные формы и, пока мы ее не закроем, доступ к КОМПАСу будет запрещен, что и требовалось. Чтобы модальную форму корректно закрыть, не обязательно выполнять метод Close. Достаточно в свойствах кнопки "Закрыть" установить ModalResult в значение mrOK. Это равносильно закрытию формы при нажатии на данную кнопку.

## 2.2. Начинаем рисовать

Самое интересное – имеющиеся в API команды построения изображений. Все они являются методами объекта iDocument2D. Давайте построим отрезок:

```
procedure TMainForm.Button1Click(Sender: TObject);  
begin  
    iDocument2D.KsLineSeg(StrToFloat(edit1.Text),  
        StrToFloat(edit2.Text), StrToFloat(edit3.Text),  
        StrToFloat(edit4.Text), 1)  
end;
```

Метод KsLineSeg(x1, y1, x2, y2, type) строит отрезок от точки x1, y1 до точки x2, y2 со стилем линии type. Стили линий имеют коды, приведенные в Приложении 2. Все рассматриваемые методы относятся к объекту Document2D и возвращают ссылку на созданный объект, которую можно запомнить в переменную типа Reference.

Рассмотрим основные методы создания 2D геометрии:

Примитив	Метод	Параметры
точка	KsPoint(x, y, style)	ставит точку с координатами x, y и стилем style. Стили точек приведены в приложении 1
отрезок	KsLineSeg(x1, y1, x2, y2, type)	проводит отрезок стилем линии type из точки (x1, y1) в точку (x2, y2)
прямая	KsLine(x, y, angle)	проводит бесконечную прямую через точку x, y под углов <b>в градусах</b> angle к положительному направлению оси OX
дуга	KsArcBy3Points(x1, y1, x2, y2, x3, y3, type)	строит дугу по трем точкам стилем линии type

Примитив	Метод	Параметры
дуга	<code>ksArcByAngle (xc, yc, rad, f1, f2, direction, type)</code>	строит дугу: xc, yc - координаты центра дуги, rad - радиус дуги, f1, f2 - начальный и конечный угол дуги в градусах, direction - направление отрисовки дуги: 1 - против часовой стрелки, -1 - по часовой стрелке, type - стиль линии
дуга	<code>ksArcByPoint (xc, yc, rad, x1, y1, x2, y2, direction, type)</code>	строит дугу: xc, yc - координаты центра дуги, rad - радиус дуги, x1, y1 - координаты начальной точки дуги, x2, y2 - координаты конечной точки дуги, direction - направление отрисовки дуги: 1 - против часовой стрелки, -1 - по часовой стрелке, type - стиль линии.
окружность	<code>ksCircle(xc, yc, rad, type)</code>	строит окружность с центром в точке xc,yc, радиусом rad и стилем линии type

### 2.3. Подключение и отладка библиотеки

После каждой компиляции библиотеки ее надо заново подключать к КОМПАСу, предварительно удалив предыдущую версию (что здорово действует на нервы...) В противном случае КОМПАС "держит" файл библиотеки и Delphi не может его перезаписать, выводя соответствующее сообщение об ошибке. Последовательность действий такова:

- изменили код библиотеки в Delphi;
- в КОМПАСе закрыли и удалили библиотеку;
- в Delphi откомпилировали библиотеку;
- в КОМПАСе подключили и запустили библиотеку.

### 3. Построение сложных объектов



Более сложные построения (эллипс, текст, размеры) требуют так много параметров, что передавать их через заголовок метода просто неудобно. Поэтому используется следующий прием: создается переменная (аналог типа RECORD), ее поля заполняются требуемыми значениями и эта переменная подается на вход соответствующему методу. Разумеется, при частом использовании таких объектов в программе разумно вынести код их построения в отдельные процедуры с соответствующими параметрами.

### Эллипс

Эллипс строится следующим кодом:

```
var par:ksEllipseParam;

begin
// создание структуры данных
par := ksEllipseParam(
kompas.GetParamStruct(ko_EllipseParam));
with par do
begin
Init; // создание структуры
xc := 50; // координаты центра
yc := 40;
a := 20; // оси
b := 10;
style := 1; // стиль линии
end;
iDocument2D.ksEllipse(par) // построение эллипса
end;
```

### NURBS-сплайн

Сплайны используются в чертежах, например, при построении профиля зубчатых колес. Сплайн – это составной объект, определяемый набором опорных точек. Рассмотрим пример построения сплайна:

```
var
par : ksNurbsPointParam;

begin
par := ksNurbsPointParam(
kompas.GetParamStruct(ko_NurbsPointParam));
with par do
begin
Init;
// 3 – порядок сплайна
// FALSE – незамкнутый (TRUE – замкнутый)
```

```

// 1 - стиль линии
iDocument2D.ksNurbs(3, False, 1);
x      := 0;
y      := 0;
weight := 1;
// добавили точку с координатами x, y и "весом" weight
iDocument2D.ksNurbsPoint(par);
x      := 20;
y      := 20;
weight := 1;
iDocument2D.ksNurbsPoint(par);
x      := 50;
y      := 10;
weight := 1;
iDocument2D.ksNurbsPoint(par);
x      := 70;
y      := 20;
weight := 1;
iDocument2D.ksNurbsPoint(par);
x      := 100;
y      := 0;
weight := 1;
iDocument2D.ksNurbsPoint(par);
x      := 50;
y      := -50;
weight := 1;
iDocument2D.ksNurbsPoint(par);
// завершили создание объекта
iDocument2D.ksEndObj
end
end;

```

Здесь сплайн 3-го порядка задан шестью точками.

### Текст

Текст отрисовывается методом `ksParagraph`, Это также составной объект. Для каждого элемента текста можно указывать большое количество параметров, приведенных в Приложении 4. Рассмотрим код, выводящий текст, показанный на Рис. 3.1.

*Первая строка*

***Вторая строка (жирный)***

Рис. 3.1 – Результат отрисовки текста.

```

var
  par : ksParagraphParam; // абзац
  itemParam : ksTextItemParam; // текст
  itemFont : ksTextItemFont; // шрифт

begin
  par := ksParagraphParam(
kompas.GetParamStruct(ko_ParagraphParam) );
  par.Init;
  // параметры абзаца
  with par do
    begin
      x      := 30; // левая верхняя точка
      y      := 30;
      height := 25; // высота
      width  := 20; // ширина
    end;
  // создаем абзац
  iDocument2d.ksParagraph( par );
  // параметры текста
  itemParam := ksTextItemParam(
kompas.GetParamStruct(ko_TextItemParam));
  itemParam.Init;
  // параметры шрифта
  itemFont := ksTextItemFont(itemParam.GetItemFont);
  itemFont.Init;
  with itemFont do
    begin
      // с новой строки
      SetBitVectorValue( NEW_LINE, true );
      // наименование шрифта
      FontName:='Arial';
      // высота текста в мм
      Height:=5;
    end;
  // собственно текст
  ItemParam.s:= 'Первая строка';
  // выводим текст
  iDocument2d.ksTextLine( itemParam );
  // следующая строка
  itemFont.Init;
  with itemFont DO

```

```

begin
  SetBitVectorValue( NEW_LINE, true );
  SetBitVectorValue( BOLD_ON, true );
  Height:=10;
end;
ItemParam.s := 'Вторая строка (жирный)';
iDocument2d.ksTextLine(itemParam);
iDocument2d.ksEndObj
end;

```

Метод SetBitVectorValue управляет оформлением текста (см. Приложение 4) и его разбивкой на строки (обратите внимание на константу NEW\_LINE).

**ЗАМЕЧАНИЕ.** Текст в основной надписи чертежа редактируется другими методами.

### Осевые линии

Как известно, осевые линии окружностей и дуг в КОМПАС строятся автоматически. Рассмотрим пример построения окружности с осевыми. Для их построения нужно запомнить ссылку на окружность в переменную типа Reference.

```

var par:ksCentreParam;
    c:reference;

begin
  // окружность, в с запомнили ссылку на нее
  c:=iDocument2D.ksCircle(150,150,20,1);
  // создаем структуру для параметров осевых
  par:=ksCentreParam(kompas.GetParamStruct(ko_CentreParam));
  with par do
    begin
      init;
      // ссылка на окружность
      BaseCurve:=c;
      // число осевых - две
      type_:=2;
      // длина осевой от центра вправо
      LenXPTail:=25;
      // длина осевой от центра влево
      LenXmTail:=25;
      // длина осевой от центра вверх
      LenYPTail:=25;
      // длина осевой от центра вниз

```

```

    LenYmTail:=25;
end;
// создание осевых
iDocument2D.ksCentreMarker(par)

```

### Штриховка

В чертежах часто встречается штриховка. Она делается хитрым образом: сначала дается команда начала штриховки, затем строятся объекты, образующие ее контур, а затем еще одна команда завершает построение.

Начинается штриховка методом

```
ksHatch(style, Angle, step, width, x0, y0)
```

который задает штрихование контура стилем `style` под углом `angle` с шагом `step` и толщиной линий `width`. Начальная точка штриховки `x0,y0`. Коды стилей штриховок приведены в Приложении 3. Затем в программе идут команды отрисовки самого контура, а завершается его построение командой `ksEndObj`.

Пример построения заштрихованной окружности:

```

iDocument2D.ksHatch(1, 0, 1, 10, 100, 100);
iDocument2D.ksCircle(100, 100, 20, 1); // контур
iDocument2D.ksEndObj;

```

### Размеры

#### Линейный размер

Простановка размера – дело непростое! Проставим размер со сложным текстом, как показано на **Ошибка! Источник ссылки не найден.**

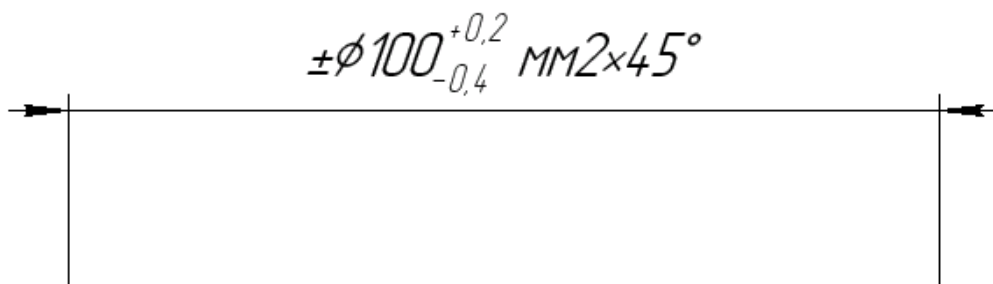


Рис. 3.2 – Линейный размер.

Размерный текст в общем случае состоит из следующих элементов:

- текст перед номиналом ( $\pm$ );
- символ перед номиналом ( $\phi$ );
- номинал ( $100$ );
- отклонение ( $+0,2$  и  $-0,4$ );

- единица измерения (мм);
- текст после номинала ( $2 \times 45^\circ$ ).

При создании размера нужно указать, какие элементы в тексте будут присутствовать и затем сформировать массив текстовых строк из необходимых элементов. В тексте можно использовать коды &04 для знака и &01 для знака градуса<sup>0</sup>. Коды символов диаметра и пр. приведены в Приложении 5.

```
var
  param      : ksLDimParam;
  dPar       : ksDimDrawingParam;
  sPar       : ksLDimSourceParam;
  tPar       : ksDimTextParam;
  str        : ksChar255;
  arrText    : ksDynamicArray;
  obj        : Reference;
begin
  // структура для параметров размера
  param := ksLDimParam
  ( kompas.GetParamStruct(ko_LDimParam) );
  // параметры отрисовки размера
  dPar := ksDimDrawingParam( param.GetDPar );
  // положение размерной линии
  sPar := ksLDimSourceParam( param.GetSPar );
  // параметры размерного текста
  tPar := ksDimTextParam ( param.GetTPar );
  dPar.Init;
  dPar.textBase := 0; // способ размещения текста
  // тип первой стрелки (1- внутри, 2 - снаружи)
  dPar.pt1      := 2;
  dPar.pt2      := 2; // тип второй стрелки
  dPar.ang      := -30; // угол наклона выноски
  dPar.lenght   := 20; // длина выноски
  sPar.Init;
  // начало первой выносной линии
  sPar.x1 := 50;
  sPar.y1 := 50;
  // начало второй выносной линии
  sPar.x2 := 150;
  sPar.y2 := 50;
  // смещение размерной линии
  sPar.dx := 0;
  sPar.dy := 20;
  // относительно какой точки задается смещение:
  // 1 - от 1, y1,
```

```

// 2 - от x2,y2
sPar.basePoint := 1;
tPar.Init( False );
// автопроставка номинала
tPar.SetBitFlagValue( _AUTONOMINAL, true );
// будет текст перед номиналом
tPar.SetBitFlagValue( _PREFIX, true );
// будет проставлено отклонение
tPar.SetBitFlagValue( _DEVIATION, true );
// будет проставлена единица измерения
tPar.SetBitFlagValue( _UNIT, true );
// будет текст после номинала
tPar.SetBitFlagValue( _SUFFIX, true );
// знак диаметра перед текстом
tPar.sign := 1;
// строка в 255 символов
str := ksChar255( Kompas.GetParamStruct(ko_Char255));
// массив текстовых строк
arrText := ksDynamicArray( tPar.GetTextArr );
// текст до номинала
str.str := '+';
// добавили в массив
arrText.ksAddArrayItem( -1, str );
// верхнее отклонение
str.str := '+0,2';
// добавили в массив
arrText.ksAddArrayItem( -1, str );
// нижнее отклонение
str.str := '-0,4';
// добавили в массив
arrText.ksAddArrayItem( -1, str );
// единица измерения
str.str := ' мм';
// добавили в массив
arrText.ksAddArrayItem( -1, str );
// текст после размера
// &04 - знак "x"
// &01 - знак градуса
str.str := '2&0445&01';
arrText.ksAddArrayItem( -1, str );
// создаем размер
iDocument2d.ksLinDimension( param )
end;

```

Отрисовка угловых, диаметральных размеров, знаков шероховатости, выносок и пр. делается аналогично. Примеры их построения приведены в файле step91.pas, входящем в комплект поставки КОМПАС. Разберитесь с ними самостоятельно.

#### Смена системы координат

В ряде случаев удобно вычислять координаты не в абсолютной системе координат, а во временной пользовательской. Для ее создания есть метод

```
ksMtr(x, y, Angle, scaleX, scaleY)
```

Здесь  $x, y$  - координаты начала локальной системы координат,  $angle$  - угол наклона системы координат в градусах,  $scaleX$  - масштаб локальной системы координат по оси X,  $scaleY$  - масштаб локальной системы координат по оси Y. Отменяется действие локальной системы координат методом `ksDeleteMtr`.

## 4. Работа с видами

Чертеж в КОМПАСе делится на виды. По умолчанию все отрисовываемые объекты оказываются на системном виде, что в общем-то неправильно. Элементы чертежа надо группировать по видам. Кроме того, каждый вид имеет свою систему координат и свой масштаб, что позволит пользователю библиотеки легко перемещать и масштабировать построенное программой изображение. Поэтому условимся, что все художества будут делаться на новом виде. Для его создания подойдет следующая процедура:

```
procedure MakeView(x0,y0:DOUBLE; nm:STRING);
// x0,y0 - точка привязки вида, nm - имя вида
var par:ksViewParam;
    n:INTEGER;

begin
  par := ksViewParam( kompas.GetParamStruct(ko_ViewParam) );
  with par do
    begin
      Init;
      // точка привязки вида
      x      := x0;
      y      := y0;
      scale_ := 1;
      Angle_ := 0;
      // вид будет активным, т.е. рисовать будем в нем
```



```

    state := stACTIVE;
    name := nm
end;
n:=0;
iDocument2D.ksCreateSheetView( par, n )
end;

```

Пример вызова:

```
MakeView(100,100,'busher');
```

Координаты левого нижнего угла вида указываются в абсолютной системе (точка 0,0 – левый нижний угол листа чертежа), имя виду дается произвольное.

## 5. Заполнение основной надписи

В документе типа "Чертеж" можно автоматизировать заполнение основной надписи. Каждая клетка основной надписи имеет свой номер (Рис. 5.1).

Например, заполним поля "Разработал" (ячейка 110) и "Дата" (ячейка 130):

```

var
    stamp      : ksStamp;
    itemParam  : ksTextItemParam;

begin
    // получаем ссылку на основную надпись
    stamp := ksStamp( iDocument2d.GetStamp );
    itemParam := ksTextItemParam(
kompas.GetParamStruct(ko_TextItemParam) );
    itemParam.Init;
    // открываем ее для редактирования
    stamp.ksOpenStamp;
    // Сделать ячейку с номером 110 текущей
    Stamp.ksColumnNumber(110);
    // Поместить в текущую ячейку одну текстовую строку
    ItemParam.s := 'Троицкий Д.И.';
    Stamp.ksTextLine( ItemParam );
    // Сделать ячейку с номером 130 текущей
    Stamp.ksColumnNumber(130);
    // Поместить в текущую ячейку одну текстовую строку -
// сегодняшнюю дату
    ItemParam.s := DateToStr(Now);
    Stamp.ksTextLine( ItemParam );
    // закрываем основную надпись
    stamp.ksCloseStamp

```

end;

Инв. № дубл.	22										
	21										
Взам. инв. №	21										
	22										
Подп. и дата	143	153	163	173	183	2					
	142	152	162	172	182						
200	141	151	161	171	181	1					
	140	150	160	170	180						
Инв. № подл.	19	Изм. / лист		№ докум.	Подп.	Дата	Лит.		Масса	Масштаб	
	10	Разраб.		110	120	130	40	41	42	5	6
19	Пров.		111	121	131	Лист		7	Листов		8
	Т.контр.		112	122	132	3					
10		113	123	133	9						
Н.контр.		114	124	134						31	
Утв.		115	125	135	32						

Рис. 5.1 – Нумерация ячеек основной надписи.

А как проверить, какого типа 2D документ является текущим – фрагмент (в нем основной надписи нет и попытка работы с ней вызовет сообщение об ошибке) или чертеж? Очень просто:

```
// получаем ссылку на основную надпись
stamp := ksStamp( iDocument2d.GetStamp );
itemParam := ksTextItemParam(
kompas.GetParamStruct(ko_TextItemParam) );
itemParam.Init;
// открываем ее для редактирования
if stamp.ksOpenStamp=0 then
begin
kompas.ksMessage
('Текущий документ не является чертежом');
Exit
end;
```

## 6. Пример разработки библиотеки

Пользуясь полученными знаниями, создадим полноценную библиотеку, выполняющую построение 2D чертежа втулки (Рис. 6.1) и заполнение основной надписи.

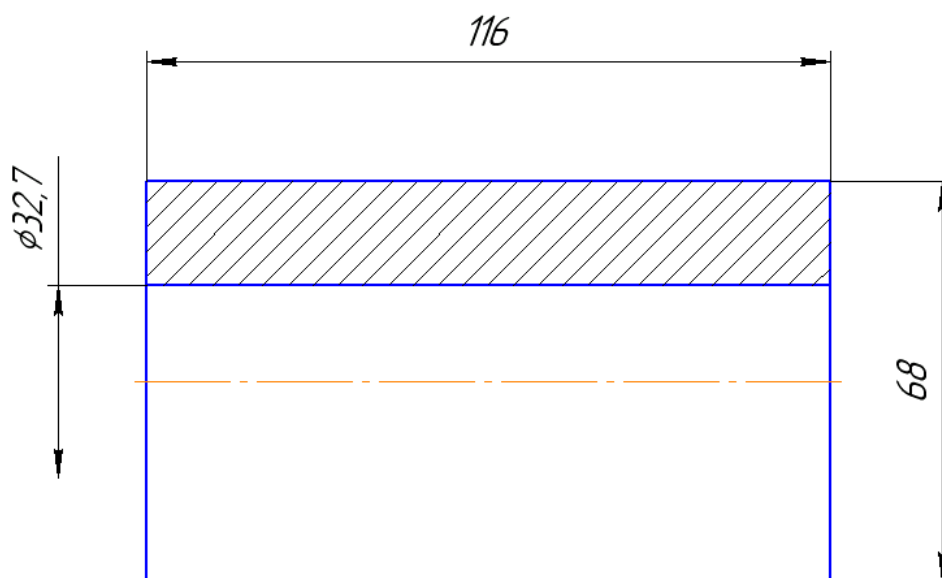


Рис. 6.1 – Чертеж втулки.

Одну точку чертежа надо будет зафиксировать. Примем левую нижнюю точку контура за (0,0) относительно системы координат вида.

Для облегчения жизни заготовим несколько процедур, выполняющих построение повторяющихся элементов. Прежде всего, это размеры. Создадим общую процедуру простановки линейного размера:

```

procedure LinDim(x1,y1,x2,y2,dx,dy:DOUBLE;
diamsign:boolean; pl1,pl2:boolean);

// простановка линейного размера
// x1,y2,x2,y2 - координаты начал выносных линий
// dx,dy - смещение размерной линии относительно точки
x1,x2
// diamsign - проставлять ли знак диаметра
// pl1, pl2 - рисовать ли первую и вторую выносные линии

var
  param      : ksLDimParam;
  dPar       : ksDimDrawingParam;
  sPar       : ksLDimSourceParam;
  tPar       : ksDimTextParam;
  str        : ksChar255;
  arrText    : ksDynamicArray;

begin
  // структура для параметров размера
  param := ksLDimParam( kompas.GetParamStruct

```

```

(ko_LDimParam) );
// параметры отрисовки размера
dPar := ksDimDrawingParam( param.GetDPar );
// положение размерной линии
sPar := ksLDimSourceParam( param.GetSPar );
// параметры размерного текста
tPar := ksDimTextParam ( param.GetTPar );
dPar.Init;
dPar.textBase := 0; // способ размещения текста
// тип первой стрелки (1-изнутри, 2-снаружи)
dPar.pt1 := 1;
dPar.pt2 := 1; // тип второй стрелки
// вкл/выкл. выносные линии
dPar.pl1 :=not(pl1);
dPar.pl2 :=not(pl2);
dPar.ang := 0; // угол наклона выноски
dPar.lenght := 0; // длина выноски
sPar.Init;
// начало первой выносной линии
sPar.x1 := x1;
sPar.y1 := y1;
// начало второй выносной линии
sPar.x2 := x2;
sPar.y2 := y2;
// смещение размерной линии
sPar.dx := dx;
sPar.dy := dy;
// относительно какой точки задается смещение:
// 1 - от 1,y1, 2 - от x2,y2
sPar.basePoint := 1;
tPar.Init( False );
// автопроставка номинала
tPar.SetBitFlagValue( _AUTONOMINAL, true );
if diamsign then
  tPar.sign := 1;
// создаем размер
iDocument2d.ksLinDimension( param )
end;

```

Для заполнения ячеек основной надписи заготовим процедуру, заносящую текст в ячейку с указанным номером:

```

procedure Stamp(c:byte; t:string);
var

```

```

    stamp      : ksStamp;
    itemParam  : ksTextItemParam;

begin
    // получаем ссылку на основную надпись
    stamp := ksStamp( iDocument2d.GetStamp );
    itemParam :=
ksTextItemParam( Kompas.GetParamStruct(ko_TextItemParam) );
    itemParam.Init;
    // открываем ее для редактирования
    if stamp.ksOpenStamp=0 then
        begin
            Kompas.KsMessage
('Текущий документ не является чертежом');
            Exit
        end;
    // Сделать ячейку текущей
    Stamp.ksColumnNumber(c);
    // Поместить в текущую ячейку одну текстовую строку
    ItemParam.s := t;
    Stamp.ksTextLine( ItemParam );
    // закрываем основную надпись
    stamp.ksCloseStamp
end;

```

Наконец, контур вокруг штриховки придется строить дважды: как собственно линии контура детали и как элементы, ограничивающие штриховку. Построение контура тоже нужно вынести в отдельную процедуру. А процедура создания вида MakeView у нас уже имеется – см. п. 4.

Ниже приведен текст всей процедуры построения чертежа втулки:

```

procedure DrawBusher
(x0,y0:double; nm:string; D1,D2,L:double);

// D1 - внутренний диаметр; D2 - наружный диаметр;
// L - длина втулки
// x0,y0 - координаты левого нижнего угла вида;
// nm - имя вида
// D1 - внутренний диаметр; D2 - наружный диаметр;
// L - длина втулки

procedure Contour; // контур штриховки

begin

```

```

iDocument2D.ksLineSeg(0, D2- (D2-D1) /2, L, D2- (D2-D1) /2, 1);
iDocument2D.ksLineSeg(0, D2- (D2-D1) /2, 0, d2, 1);
iDocument2D.ksLineSeg(L, D2- (D2-D1) /2, L, d2, 1);
iDocument2D.ksLineSeg(0, D2, L, D2, 1)
end;

begin
// создали вид
MakeView(x0, y0, nm);
// осевая линия
iDocument2D.ksLineSeg(-5, d2/2, L+5, d2/2, 3);
// внешний контур
iDocument2D.ksLineSeg(0, 0, 1, 0, 1);
iDocument2D.ksLineSeg(0, 0, 0, D2- (D2-D1) /2, 1);
iDocument2D.ksLineSeg(L, 0, L, D2- (D2-D1) /2, 1);
Contour;
// штриховка
iDocument2D.ksHatch(0, 45, 1, 5, 5, D2- (D2-D1) /2+5);
Contour;
iDocument2D.ksEndObj;
// длина
LinDim(0, 0, L, 0, 0, -20, false, true, true);
// наружный диаметр
LinDim(0, 0, 0, d2, -20, 0, true, true, true);
// внутренний диаметр
LinDim(0, (D2-D1) /2, 0, (D2-D1) /2+d1, -10, 0, true, false, true);
// Основная надпись
Stamp(1, 'Втулка');
Stamp(3, 'Сталь 40X');
Stamp(9, 'ТулГУ');
Stamp(8, '1');
Stamp(110, 'Троицкий Д.И. ');
Stamp(130, DateToStr(Now))
end;

```

При вызове процедуры ей на вход подаются параметры, вводимые пользователем:

```

procedure TMainForm.Button3Click(Sender: TObject);

var d1, d2, l:double;

begin
try

```

```
d1:=StrToFloat(Edit1.Text);  
d2:=StrToFloat(Edit2.Text);  
l:=StrToFloat(Edit4.Text)  
except  
  Kompas.kSendMessage('Неверные данные');  
  abort  
end;  
if D2>=d1 then  
  begin  
    Kompas.kSendMessage('Внутренний диаметр больше наружного');  
    abort  
  end;  
  DrawBusher(100,100, 'busher',d2, d1, l)  
end;
```

Как видите, создание библиотек для КОМПАС – дело непростое, но выполнимое.

**Успехов в программировании!**

## 7. Приложение 1. Стили точек

0	точка
1	крестик
2	х-точка
3	квадрат
4	треугольник
5	окружность
6	звезда
7	перечеркнутый квадрат
8	утолщенный плюс

## 8. Приложение 2. Коды стилей линий

1	основная линия
2	тонкая линия
3	осевая линия
4	штриховая линия
5	для линии обрыва
6	вспомогательная линия
7	утолщенная
8	штрихпунктирная линия с двумя точками
9	штриховая толстая линия
10	осевая толстая линия
11	тонкая линия, включаемая в штриховку
12	ISO штриховая линия
13	ISO штриховая линия (дл. пробел)
14	ISO штрихпунктирная линия (дл. штрих)
15	ISO штрихпунктирная линия (дл. штрих 2 пунктира)
16	ISO штрихпунктирная линия (дл. штрих 3 пунктира)
17	ISO пунктирная линия
18	ISO штрихпунктирная линия (дл. и кор. штрихи)
19	ISO штрихпунктирная линия (дл. и 2 кор. штриха)
20	ISO штрихпунктирная линия
21	ISO штрихпунктирная линия (2 штриха)
22	ISO штрихпунктирная линия (2 пунктира)
23	ISO штрихпунктирная линия (3 пунктира)
24	ISO штрихпунктирная линия (2 штриха 2 пунктира)
25	ISO штрихпунктирная линия (2 штриха 3 пунктира)



### 9. Приложение 3. Коды стилей штриховок

0	металл
1	неметалл
2	дерево
3	камень естественный
4	керамика
5	бетон
6	стекло
7	жидкость
8	естественный грунт
9	насыпной грунт
10	камень искусственный
11	железобетон
12	напряженный железобетон
13	дерево в продольном сечении
14	песок

### 10. Приложение 4. Коды свойств текста

Константа	Шестнадцатиричное значение	Смысл
INVARIABLE	0	не менять флаги текста
NUMERATOR	\$1	числитель
DENOMINATOR	\$2	знаменатель
END_FRACTION	\$3	конец дроби
UPPER_DEVIAT	\$4	верхнее отклонение
LOWER_DEVIAT	\$5	нижнее отклонение
END_DEVIAT	\$6	конец отклонений
S_BASE	\$7	основание выражения с под или надстрокой
S_UPPER_INDEX	\$8	верхний индекс выражения с под или надстрокой
S_LOWER_INDEX	\$9	нижний индекс выражения с под или надстрокой
S_END	\$10	конец выражения с под или надстрокой
SPECIAL_SYMBOL	\$11	спецзнак
SPECIAL_SYMBOL_END	\$12	конец спецзнака для спецзнаков с текстом

Константа	Шестнадцатиричное значение	Смысл
RETURN_BEGIN	\$13	начало для ввода следующих строк в спецзнаке с текстом, дробях, отклонениях
RETURN_DOWN	\$14	для ввода следующих строк в спецзнаке с текстом, дробях, отклонениях
RETURN_RIGHT	\$15	для ввода строк справа в спецзнаке с текстом, дробях, отклонениях
TAB	\$16	табуляция по текущему стилю
FONT_SYMBOL	\$17	символ фонта
ITALIC_ON	\$40	включить наклон
ITALIC_OFF	\$80	выключить наклон
BOLD_ON	\$100	включить жирное начертание
BOLD_OFF	\$200	выключить жирное начертание
UNDERLINE_ON	\$400	включить подчеркивание
UNDERLINE_OFF	\$800	выключить подчеркивание
NEW_LINE	\$1000	новая строка в параграфе

## 11. Приложение 5. Коды размеров

Стрелки (свойства pt1,pt2)

0	стрелки нет
1	стрелка внутри
2	стрелка снаружи
3	засечка
4	точка

Положение текста (свойство textBase)

0	в центре
1	на расстояние (или угол) textPos относительно первой точки
2	на расстояние (или угол) textPos относительно второй точки

Управление размерным текстом (свойство bitFlag)

Константа	Шестнадцатиричное значение	Смысл
_AUTONOMINAL	\$1	автоматическое определение номинального значения размера
_RECTTEXT	\$2	текст в рамке
_PREFIX	\$4	есть текст до номинала
_NOMINALOFF	\$8	нет номинала

Константа	Шестнадцатиричное значение	Смысл
_TOLERANCE	\$10	проставлять квалитет
_DEVIATION	\$20	проставлять отклонения
_UNIT	\$40	единица измерения
_SUFFIX	\$80	есть текст после номинала
_DEVIATION_INFORM	\$100	при включенном флаге _DEVIATION отклонения есть в массиве текстов (даже если простановка отклонений - не ручная)
_UNDER_LINE	\$200	размер с подчеркиванием
_BRACKETS	\$400	размер в скобках
_SQUARE_BRACKETS	\$800	размер в квадратных скобках, используется вместе с _BRACKETS

Значок перед размерным текстом (свойство Sign)

0	нет значка
1	диаметр $\varnothing$
2	квадрат
3	радиус $R$
>3	код символа из шрифта "Symbol type A"