

Лабораторная работа №4.

Тема: установка различных размеров

Простой линейный размер

Несмотря на кажущуюся простоту задачи простановки простого линейного размера при ее реализации приходится использовать несколько различных интерфейсов описывающих различные параметры проставляемого размера. Далее мы их рассмотрим.

Строка (ksChar255)

Интерфейс *ksChar255* предназначен для описания простой строки длиной до 255 символов. Получить данный интерфейс можно при помощи метода ***GetParamStruct()*** интерфейса *KompasObject*. Для этого в качестве единственного параметра данному методу нужно передать константу *ko_Char255*.

У данного интерфейса всего одно свойство – *str*. Данное свойство содержит саму строку, описываемую этим интерфейсом.

Метод у данного интерфейса также один – *Init()*. Он очищает свойство *str* данного интерфейса.

Динамический массив (ksDynamicArray)

Интерфейс *ksDynamicArray* предназначен для описания динамических массивов. В данный массив можно передавать различные интерфейсы. В контексте задачи простановки линейного размера нас будет интересовать динамический массив интерфейсов *ksChar255*, то есть строк.

Получить данный интерфейс можно при помощи метода ***GetDynamicArray()*** интерфейса *KompasObject*. В качестве единственного параметра данный метод принимает тип создаваемого массива. Ниже представлены типы динамических массивов.

Символьные обозначения типа массива	Число обозначение типа массива	Интерфейс – элемент массива	Краткое описание интерфейса – элемента массива
CHAR_STR_ARR	1	Динамический массив указателей на интерфейсы <i>ksChar255</i>	динамический массив указателей на строки символов
POINT_ARR	2	Динамический массив указателей на интерфейсы <i>ksMathPointParam</i>	динамический массив указателей на математические точки (структура <i>MathPointParam</i>)

)
CURVE_PATTERN_ARR	2	Динамический массив указателей на интерфейсы <i>ksCurvePattern</i>	динамический массив указателей на участки штриховой линии (структура <u><i>CurvePattern</i></u>)
TEXT_LINE_ARR	3	Динамический массив указателей на интерфейсы <i>ksTextLineParam</i>	динамический массив строк текста (структура <u><i>TextLineParam</i></u>)
TEXT_ITEM_ARR	4	Динамический массив указателей на интерфейсы <i>ksTextItemParam</i>	динамический массив компонент строк текста (структура <u><i>TextItemParam</i></u>)
ATTR_COLUMN_ARR	5	Динамический массив указателей на интерфейсы <i>ksColumnInfoParam</i>	динамический массив колонок атрибутов (структура <u><i>ColumnInfo</i></u>)
USER_ARR	6	Динамический пользовательский массив	динамический пользовательский массив
POLYLINE_ARR	7	Динамический массив указателей на интерфейсы <i>ksDynamicArray</i> типа <i>POINT_ARR</i>	динамический массив полилиний (указателей массивов <i>POINT_ARR</i>)
RECT_ARR	8	Динамический массив указателей на интерфейсы <i>ksRectParam</i>	динамический массив габаритных прямоугольников (структура <u><i>RectParam</i></u>)
LIBRARY_STYLE_ARR	9	Динамический массив указателей на интерфейсы <i>ksLibraryStyleParam</i>	динамический массив структур параметров для стиля в библиотеке стилей(<u><i>LibraryStyleParam</i></u>)
VARIABLE_ARR	10	Динамический массив указателей на интерфейсы <i>ksVariable</i>	динамический массив структур параметров параметрических переменных (<u><i>VariableParam</i></u>)
CURVE_PATTERN_ARR_EX	11	Динамический массив указателей на интерфейсы <i>ksCurvePattern</i>	динамический массив указателей на участки штриховой линии (структура <u><i>CurvePatternEx</i></u>)
LIBRARY_ATTR_TYPE_ARR	12	Динамический массив указателей на интерфейсы	динамический массив структур параметров для типа атрибута в библиотеке

		<i>ksLibraryAttrTypeParam</i>	типов атрибутов (<u><i>LibraryAttrTypeParam</i></u>)
NURBS_POINT_ARR	13	Динамический массив указателей на интерфейсы <i>ksNurbsPointParam</i>	динамический массив структур точек кривой <i>NURBS</i>
DOUBLE_ARR	14	Динамический массив указателей на интерфейсы <i>ksDoubleValue</i>	динамический массив значений типа <i>double</i>
CONSTRAINT_ARR	15	Динамический массив указателей на интерфейсы <i>ksConstraintParam</i>	динамический массив структур параметрических ограничений (<u><i>ConstraintParam</i></u>)
CORNER_ARR	16	Динамический массив указателей на интерфейсы <i>ksCornerParam</i>	динамический массив структур параметров скругленных (или усеченных) углов прямоугольников и многоугольников (<u><i>CornerParam</i></u>)
DOC_SPCOBJ_ARR	17	Динамический массив указателей на интерфейсы <i>ksDocAttachedSpcParam</i>	динамический массив структур параметров документов, подключенных к объекту спецификации (<u><i>DocAttachedSpcParam</i></u>)
SPCSUBSECTION_ARR	18	Динамический массив указателей на интерфейсы <i>ksSpcSubSectionParam</i>	динамический массив структур параметров подраздела спецификации (<u><i>SpcSubSectionParam</i></u>)
SPCTUNINGSEC_ARR	19	Динамический массив указателей на интерфейсы <i>ksSpcTuningSectionParam</i>	динамический массив структур параметров настройки раздела спецификации (<u><i>SpcTuningSectionParam</i></u>)
SPCSTYLECOLUMN_ARR	20	Динамический массив указателей на интерфейсы <i>ksSpcStyleColumnParam</i>	динамический массив структур параметров стиля колонки спецификации (<u><i>SpcStyleColumnParam</i></u>)
SPCSTYLESEC_ARR	21	Динамический массив указателей на интерфейсы <i>ksSpcStyleSectionParam</i>	динамический массив структур параметров стиля раздела спецификации (<u><i>SpcStyleSectionParam</i></u>)

QUALITYITEM_ARR	22	Динамический массив указателей на интерфейсы <i>ksQualityItemParam</i>	динамический массив структур параметров интервала качества (<i>QualityItemParam</i>)
LTVARIANT_ARR	23	Динамический массив указателей на интерфейсы <i>ksLtVariant</i>	динамический массив структур для хранения данных некоторого типа (<i>LtVariant</i>)
TOLERANCEBRANCH_ARR	24	Динамический массив указателей на интерфейсы <i>ksToleranceBranch</i>	динамический массив структур параметров "опоры" допуска формы (<i>ToleranceBranch</i>)
HATCHLINE_ARR	25	Динамический массив указателей на интерфейсы <i>ksHatchLineParam</i>	динамический массив структур (<i>HatchLineParam</i>)
TREENODEPARAM_ARR	26	Динамический массив указателей на интерфейсы <i>ksTreeNodeParam</i>	динамический массив структур узла дерева <i>TreeNodeParam</i>
CHAR_STR_ARR_W	27		динамический массив указателей на строки символов. Используется в Unicode-вских структурах и функциях.

ksSetArrayItem() позволяет изменить значение элемента массива, ниже представлен прототип данного метода:

ksSetArrayItem(int index, const item IDispatch), где *index* – индекс изменяемого элемента массива, а *IDispatch* – устанавливаемое значение.

Параметры размерной надписи (ksDimTextParam)

Интерфейс *ksDimTextParam* позволяет задать параметры размерной надписи. Ниже представлены свойства этого интерфейса:

bitFlag – набор флагов, определяющий «состав» размерной надписи, то есть перечень тех элементов из которых состоит размерная надпись. В таблице ниже представлены эти флаги вместе с их описаниями и числовыми значениями (в шестнадцатеричной системе счисления)

Символьное обозначение флага	Числовое значение флага	Описание
_AUTONOMINAL	1	Автоматическое определение размера и

		проставка его в размерной надписи
_RECTTEXT	2	Надпись выводится в рамке
_PREFIX	4	Есть текст перед самым числовым размером (номиналом)
_NOMINALOFF	8	Сам размер (номинал) не выводится
_TOLERANCE	10	В состав размерной надписи входит квалитет
_DEVIATION	20	В состав размерной надписи входят отклонения
_UNIT	40	В состав размерной надписи входит единица измерения
_SUFFIX	80	В состав размерной надписи входит текст после самого размера (номинала)
_DEVIATION_INFORM	100	Согласно документации КОМПАС данный флаг говорит о том, что мы вручную указываем отклонения без учета устанавливаемого квалитета
UNDER_LINE	200	Размерная надпись выводится с подчеркиванием
_BRACKETS	400	Размерная надпись выводится в круглых скобках
_SQUARE_BRACKETS	800	Используется только совместно с флагом _BRACKETS. Говорит о том, что размерная надпись выводится в квадратных скобках

Sign – определяет тип условного значка перед размером. Ниже представлены значения этого поля для разных условных значков:

- 0 – нет никакого значка;
- 1 – диаметр;
- 2 – квадрат;
- 3 – радиус.

StringFlag - определяет тип динамического массива, в котором должны храниться строки с элементами размерной надписи. Если это поле принимает значение **true**, то массив должен иметь тип **TEXT_LINE_ARR**, если же это поле равно **false**, то массив должен иметь тип **CHAR_STR_ARR**. Единственная разница между этими массивами в том, какой интерфейс является элементом массива.

Style – задает системный стиль текста. В таблице ниже представлены все системные стили текстов из документации КОМПАС

Номер стиля текста	Описание
0	Стиль по умолчанию
1	Обычный текст
2	Текст для технических требования
3	Текст размерной надписи
4	Текст в обозначении шероховатости
5	Текст на позиционной линии-выноске
6	Текст над/под полкой линии-выноске
7	Текст на ответвлении линии-выноске
8	Текст в обозначении допуска формы
9	Текст для заголовка таблицы
10	Текст для ячейки таблицы
11	Текст для линии разреза
12	Текст для стрелки направления взгляда
13	Текст в обозначении неуказанной шероховатости
14	Текст в обозначении изменения

Ниже представлены методы данного интерфейса.

GetBitFlagValue(int bitFlag) – позволяет проверить установлен отдельно взятый флаг в свойстве **bitFlag** или нет. Единственный параметр данного метода представляет собой проверяемый флаг из таблицы доступных флагов для поля **bitFlag** данного интерфейса.

GetTextArr() – возвращает интерфейс динамического массива **ksDynamicArray**, в котором хранятся строки с элементами выводимой размерной надписи.

Init – служит для обнуления интерфейса **ksDimTextParam**, то есть сброса всех его свойств в значение по умолчанию.

SetBitFlagValue(int val; bool state) – служит для изменения состояния флагов в свойстве **bitFlag**. Параметр state описывает, что нужно сделать с этим флагом: **true** – установить; **false** – сбросить.

SetTextArr() – служит для изменения динамического массива строк, в котором хранятся строки с элементами размерной надписи.

Параметры отрисовки размера (ksDimDrawingParam)

Интерфейс *ksDimDrawingParam* позволяет описать параметры отрисовки размера, то есть то, как отрисовывать размер, как он должен выглядеть. Ниже представлены свойства данного интерфейса:

- **Ang** – задает угол наклона ножки выносной полки;
- **Pl1** – определяет нужно ли отрисовывать выносную линию. Если данное свойство равняется true, то выносная линия не отрисовывается, в противном случае отрисовывается;
- **Pl2** – аналогично свойству pl1, но для второй выносной линии;
- **Length** – длина ножки выносной полки;
- **Pt1** – задает тип стрелки у первой выносной линии. Возможные значения данного свойства и их интерпретация представлены в таблице ниже;

Числовое значение	Описание
0	Стрелки нет
1	Стрелка внутри
2	Стрелка снаружи
3	Засечка
4	Точка

- **Pt2** – аналогично свойству pl1, но для второй выносной линии;
- **shelfDir** – направления выносной полки. Ниже представлены все допустимые значения этого свойства;

Числовое значение	Описание
-1	Полка влево
0	Полки нет
1	Полка вправо
2	Полка вверх
3	Полка вниз

- **textBase** – задает характер размещения размерной надписи. В таблице ниже перечислены возможные значения данного свойства;

Числовое значение	Описание
0	Размерная надпись выравнивается по центру размерной линии
1	Размерная надпись размещается на фиксированном расстоянии от первой точки
2	Размерная надпись размещается на фиксированном расстоянии от второй точки
3	Общая размерная линия

- **textPos** – в случае если свойство **textBase** принимает значение 0, 1 или 2, данное свойство задает фиксированное расстояние, на которое размерная надпись удаляется от одной из концевых точек размерной линии.

Метод у данного интерфейса всего один – **Init()**. Он не имеет входные параметры и служит для сброса всех свойств данного интерфейса в значения по умолчанию.

Параметры привязки линейного размера (ksLDimSourceParam)

Интерфейс **ksLDimSourceParam** также служит для описания параметров отрисовки линейного размера. Ниже представлены его свойства:

- **basePoint** – определяет от какой из двух концевых точек размерной линии следует откладывать вектор **[dx;dy]**;
- **dx** и **dy** – задают ориентацию размерной линии. Конкретная их интерпретация зависит от свойства **ps** этого же интерфейса;
- **ps** – определяет ориентацию размерной линии и то, как следует интерпретировать параметры **dx** и **dy**. Возможные значения данного параметра описаны в таблице ниже.

Значение ps	Описание	Интерпретация dx	Интерпретация dy
0	Размерная линия располагается горизонтально	игнорируется	Расстояние между размерной линией и измеряемым отрезком
1	Размерная линия располагается вертикально	Расстояние между размерной линией и измеряемым отрезком	игнорируется
2	Размерная линия располагается параллельно измеряемому отрезку	Задают вектор, в направление которого строится размерная линия. При этом сама размерная линия остается параллельно измеряемому отрезку	
3	Выносная линия располагается по вектору [dx; dy]	Вектор [dx; dy] задает ориентацию и размер одной из выносных линий. Вторая выносная линия располагается параллельно первой, но, может иметь несколько другой размер, из-за чего размерная линия располагается не параллельно измеряемому отрезку. Почему возникает такое различие в длине я так и не понял.	
4	Параллельно измеряемому отрезку с выносными линиями по вектору [dx; dy]	Аналогично предыдущему, только в этот раз размерная линия гарантированно располагается параллельно измеряемому отрезку	

- *x1* и *y1* – координаты первой точки измеряемого отрезка.
- *x2* и *y2* – координаты второй точки измеряемого отрезка.

Метод у данного интерфейса всего один: *Init()*. Он не имеет входных параметров и служит для обнуления всех свойств данного интерфейса.

Параметры линейного размера (ksLDimParam)

Интерфейс *ksLDimParam* представляет собой объединение трех ранее рассмотренных интерфейсов. *ksDimDrawingParam* (определяет параметры отрисовки линейного размера), *ksLDimSourceParam* (определяет параметры привязки линейного размера) и *ksDimTextParam* (определяет параметры размерной надписи).

У интерфейса *ksLDimParam* нет свойств, поэтому переходим к рассмотрению его методов:

- *GetDPar()* – возвращает интерфейс *ksDimDrawingParam*;
- *GetSPar()* – возвращает интерфейс *ksLDimSourceParam*;
- *GetTPar()* – возвращает интерфейс *ksDimTextParam*;
- *SetDPar()* – устанавливает новый интерфейс *ksDimDrawingParam*, передаваемый ему в качестве единственного параметра;
- *SetSPar()* – устанавливает новый интерфейс *ksLDimSourceParam*, передаваемый ему в качестве единственного параметра;
- *SetTPar()* – устанавливает новый интерфейс *ksLDimSourceParam*, передаваемый ему в качестве единственного параметра.

Построение линейного размера

Для построения линейного размера предназначен метод *ksLinDimension()* интерфейса *ksDocument2D*. В случае успеха данный метод возвращает числовой идентификатор построенного размера, а в случае ошибки – нуль.

Ниже приводится фрагмент программы, в которой строится простой горизонтальный размер:

```
doc_2d.kLineSeg(100, 100, 130, 100, 1);
// Получаем интерфейс ksLDimParam
LdimPar = (ksLDimParam)kompas.GetParamStruct((short)
    Kompas6Constants.StructType2DEnum.ko_LDimParam);
// Получаем параметры интерфейса ksDimDrawingParam
DimDrawingPar = (ksDimDrawingParam)LdimPar.GetDPar();
// Устанавливаем параметры отрисовки размера
DimDrawingPar.ang = 0; // Угол наклона ножки выносной полки
DimDrawingPar.lenght = 0; // Длина ножки выносной полки
DimDrawingPar.pl1 = false; // Отрисовывать первую выносную линию
DimDrawingPar.pl2 = false; // Отрисовывать вторую выносную линию
DimDrawingPar.pt1 = 1; // У первой выносной линии стрелка внутри
DimDrawingPar.pt2 = 1; // У второй выносной линии стрелка внутри
DimDrawingPar.shelfDir = 0; // Выносной полки нет
DimDrawingPar.textBase = 0; // Текст размещается в центре
```

```

DimDrawingPar.textPos = 0; // Автоматическое размещение текста
// Получаем интерфейс
LdimSourcePar = (ksLDimSourceParam)LdimPar.GetSPar();
// Устанавливаем параметры привязки линейного размера
LdimSourcePar.basePoint = 1; // Рисуем от первой линии ко второй
// вектор первой точки измеряемого

```

отрезка

```

LdimSourcePar.dx = 0;
LdimSourcePar.dy = 10;
LdimSourcePar.ps = 0; // Размерная линия размещена горизонтально
// Координаты первой точки измеряемого отрезка
LdimSourcePar.x1 = 100;
LdimSourcePar.y1 = 100;
// Координаты второй точки измеряемого отрезка
LdimSourcePar.x2 = 130;
LdimSourcePar.y2 = 100;
// Получаем интерфейс ksDimTextParam
DimTextPar = (ksDimTextParam)LdimPar.GetTPar();
// Устанавливаем параметры размерной надписи
DimTextPar.bitFlag = 0; // Будем выводить только номинал
DimTextPar.sign = 0; // Значка нет
DimTextPar.stringFlag = false;
// Получаем интерфейс ksChar255
char255 = (ksChar255)kompas.GetParamStruct((short)
    Kompas6Constants.StructType2DEnum.ko_Char255);
// Получаем интерфейс ksDynamicArray
DynamicArray = (ksDynamicArray)DimTextPar.GetTextArr();
DynamicArray.ksClearArray();
// Строка с номиналом
char255.str = "30";
DynamicArray.ksAddArrayItem(-1, char255);
// Строим сам размер
doc_2d.ksLinDimension(LdimPar);

kompas.Visible = true;

```

Результат работы данного фрагмента представлен на рисунке 1.

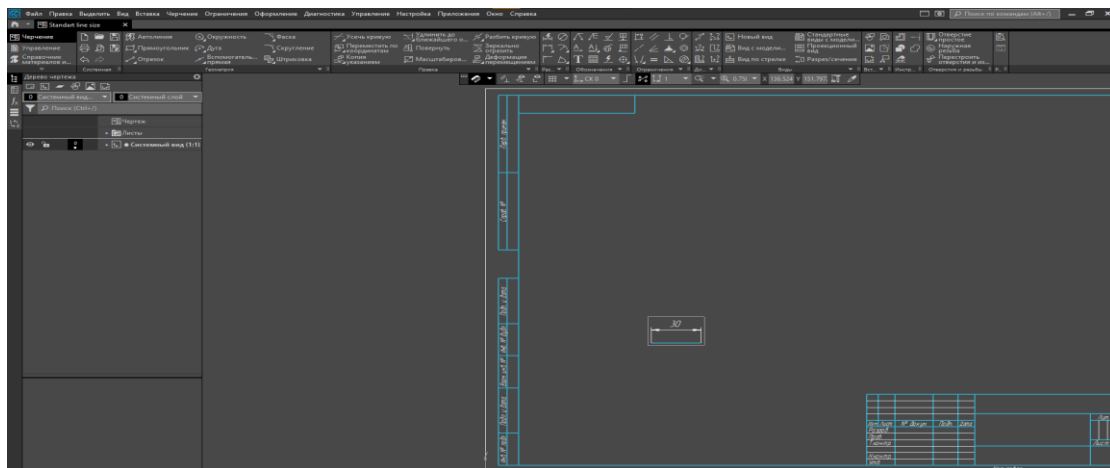


Рисунок 1 – Построение простого линейного размера

Горизонтальный размер со стрелками снаружи

Иногда требуется, чтобы стрелки на размерной линии размещались снаружи. Это особенно актуально, когда проставляемый размер невелик и места для простановки самого размера и стрелок просто не хватает. Ниже приводится фрагмент программы решающий данную задачу:

```
// Отрезок длину которого будем проставлять в размере
doc_2d.ksLineSeg(100, 100, 200, 100, 1);
// Получаем интерфейс ksLDimParam
ldimPar = (ksLDimParam)kompas.GetParamStruct((short)
    Kompas6Constants.StructType2DEnum.ko_LDimParam);
// Получаем интерфейс ksDimDrawingParam
dimDrawingPar = (ksDimDrawingParam)ldimPar.GetDPar();
dimDrawingPar.ang = 0; // угол наклона ножки выносной полки
dimDrawingPar.lenght = 0; // длина ножки выносной полки
dimDrawingPar.pl1 = false; // Отрисовывать первую выносную линию
dimDrawingPar.pl2 = false; // Отрисовывать вторую выносную линию
dimDrawingPar.pt1 = 2; // У первой выносной линии стрелка внутри
dimDrawingPar.pt2 = 2; // У второй выносной линии стрелка внутри
dimDrawingPar.shelfDir = 0; // Выносной полки нет
dimDrawingPar.textBase = 0; // Текст размещается в центре
dimDrawingPar.textPos = 0; // Автоматическое размещение текста
// Получаем интерфейс ksDimSourceParam
ldimSourcePar = (ksLDimSourceParam)ldimPar.GetSPar();
// Устанавливаем параметры привязки линейного размера
ldimSourcePar.basePoint = 1; // Рисуем от первой точки ко второй
// Вектор направления размерной надписи
ldimSourcePar.dx = 0;
ldimSourcePar.dy = 10;
ldimSourcePar.ps = 0; // Размерная линия размещена горизонтально
// Координаты первой точки измеряемого отрезка
ldimSourcePar.x1 = 100;
ldimSourcePar.y1 = 100;
// Координаты второй точки измеряемого отрезка
ldimSourcePar.x2 = 200;
ldimSourcePar.y2 = 100;
// Получаем интерфейс ksDimTextParam
dimTextParam = (ksDimTextParam)ldimPar.GetTPar();
// Устанавливаем параметры основной надписи
dimTextParam.bitFlag = flags_arr[0]; // выводим квалитет
dimTextParam.sign = 0; // Значка нет
dimTextParam.stringFlag = false;

// Строим сам размер
doc_2d.ksLinDimension(ldimPar);

kompas.Visible = true;
```

Вертикальный размер

Хорошо с горизонтальным размером разобрались, перейдем к вертикальному размеру. Его простановка ничуть не сложнее простановки горизонтального размера. Ниже приводится фрагмент программы, демонстрирующий решение данной задачи:

```
int SHEET_TYPE = 4;
KompasObject Kompas;
ksDocument2D doc_2d;
ksChar255 char255;
ksDocumentParam doc_par;
ksSheetOptions sheet_opt;
ksStandartSheet st_sheet;
ksDynamicArray dyn_arr;
ksLDimParam ldim_par;
ksDimDrawingParam dimDrawing_par;
ksLDimSourceParam ldimSource_par;
ksDimTextParam dimText_par;

Kompas =
(KompasObject)Marshal.GetActiveObject("Kompas.Application.5");
doc_2d = (ksDocument2D)Kompas.Document2D();
doc_par = (ksDocumentParam)Kompas.GetParamStruct((short)
    Kompas6Constants.StructType2DEnum.ko_DocumentParam);
doc_par.Init();
doc_par.author = System.Environment.UserName;
doc_par.fileName = "Standart line size";
doc_par.type = (short)DocumentTypeEnum.ksDocumentDrawing;
doc_2d.ksCreateDocument(doc_par);

sheet_opt = (ksSheetOptions)Kompas.GetParamStruct((short)
    Kompas6Constants.StructType2DEnum.ko_SheetOptions);
doc_2d.ksGetDocOptions(SHEET_TYPE, sheet_opt);
sheet_opt.sheetType = false;
st_sheet = (ksStandartSheet)sheet_opt.GetSheetParam(false);
st_sheet.format = 3;
st_sheet.direct = true;
st_sheet.multiply = 1;
sheet_opt.shtType = 1;

doc_2d.ksSetDocOptions(SHEET_TYPE, sheet_opt);

doc_2d.ksLineSeg(200, 200, 200, 100, 1);
// Получаем интерфейс ksLDimParam
ldim_par = (ksLDimParam)Kompas.GetParamStruct((short)
    Kompas6Constants.StructType2DEnum.ko_LDimParam);
// Получаем параметры интерфейса ksDimDrawingParam
dimDrawing_par = (ksDimDrawingParam)ldim_par.GetDPar();
// Устанавливаем параметры отрисовки размера
dimDrawing_par.ang = 0; // Угол наклона ножки выносной полки
dimDrawing_par.lenght = 0; // Длина ножки выносной полки
dimDrawing_par.pl1 = false; // Отрисовывать первую выносную линию
dimDrawing_par.pl2 = false; // Отрисовывать вторую выносную линию
dimDrawing_par.pt1 = 1; // У первой выносной линии стрелка внутри
```

```

dimDrawing_par.pt2 = 1; // У второй выносной линии стрелка внутри
dimDrawing_par.shelfDir = 0; // Выносной полки нет
dimDrawing_par.textBase = 0; // Текст размещается в центре
dimDrawing_par.textPos = 0; // Автоматическое размещение текста
// Получаем интерфейс
ldimSource_par = (ksLDimSourceParam)ldim_par.GetSPar();
// Устанавливаем параметры привязки линейного размера
ldimSource_par.basePoint = 1; // Рисуем от первой линии ко второй
// вектор первой точки измеряемого

```

отрезка

```

ldimSource_par.dx = 10;
ldimSource_par.dy = 0;
ldimSource_par.ps = 1; // Размерная линия размещена вертикально
// Координаты первой точки измеряемого отрезка
ldimSource_par.x1 = 200;
ldimSource_par.y1 = 200;
// Координаты второй точки измеряемого отрезка
ldimSource_par.x2 = 200;
ldimSource_par.y2 = 100;
// Получаем интерфейс ksDimTextParam
dimText_par = (ksDimTextParam)ldim_par.GetTPar();
// Устанавливаем параметры размерной надписи
dimText_par.bitFlag = 0; // Будем выводить только номинал
dimText_par.sign = 0; // Значка нет
dimText_par.stringFlag = false;
// Получаем интерфейс ksChar255
char255 = (ksChar255)kompas.GetParamStruct((short)
    Kompas6Constants.StructType2DEnum.ko_Char255);
// Получаем интерфейс ksDynamicArray
dyn_arr = (ksDynamicArray)dimText_par.GetTextArr();
dyn_arr.ksClearArray();
// Строка с номиналом
char255.str = "100";
dyn_arr.ksAddArrayItem(-1, char255);
// Строим сам размер
doc_2d.ksLinDimension(ldim_par);

kompas.Visible = true;

```

Вертикальный размер с обрывом

При простановке больших размеров часто прибегают к такой хитрости: вместо того, чтобы ставить весь размер обозначают только одну из его крайних частей, а вторую опускают. Такой размер называют размером с обрывом. Ниже приводится фрагмент программы, реализующий простановку такого размера:

```

int SHEET_TYPE = 4;
KompasObject kompas;
ksDocument2D doc_2d;
ksChar255 char255;
ksDocumentParam doc_par;

```

```

ksSheetOptions sheet_opt;
ksStandartSheet st_sheet;
ksDynamicArray dyn_arr;
ksLDimParam ldim_par;
ksDimDrawingParam dimDrawing_par;
ksLDimSourceParam ldimSource_par;
ksDimTextParam dimText_par;

kompas =
(KompasObject)Marshal.GetActiveObject("Kompas.Application.5");
doc_2d = (ksDocument2D)kompas.Document2D();
doc_par = (ksDocumentParam)kompas.GetParamStruct((short)
    Kompas6Constants.StructType2DEnum.ko_DocumentParam);
doc_par.Init();
doc_par.author = System.Environment.UserName;
doc_par.fileName = "Standart line size";
doc_par.type = (short)DocumentTypeEnum.ksDocumentDrawing;
doc_2d.ksCreateDocument(doc_par);

sheet_opt = (ksSheetOptions)kompas.GetParamStruct((short)
    Kompas6Constants.StructType2DEnum.ko_SheetOptions);
doc_2d.ksGetDocOptions(SHEET_TYPE, sheet_opt);
sheet_opt.sheetType = false;
st_sheet = (ksStandartSheet)sheet_opt.GetSheetParam(false);
st_sheet.format = 3;
st_sheet.direct = true;
st_sheet.multiply = 1;
sheet_opt.shtType = 1;

doc_2d.ksSetDocOptions(SHEET_TYPE, sheet_opt);

doc_2d.ksLineSeg(200, 200, 200, 100, 1);
// Получаем интерфейс ksLDimParam
ldim_par = (ksLDimParam)kompas.GetParamStruct((short)
    Kompas6Constants.StructType2DEnum.ko_LDimParam);
// Получаем параметры интерфейса ksDimDrawingParam
dimDrawing_par = (ksDimDrawingParam)ldim_par.GetDPar();
// Устанавливаем параметры отрисовки размера
dimDrawing_par.ang = 0; // Угол наклона ножки выносной полки
dimDrawing_par.lenght = 0; // Длина ножки выносной полки
dimDrawing_par.pl1 = true; // Отрисовывать первую выносную линию
dimDrawing_par.pl2 = false; // Отрисовывать вторую выносную линию
dimDrawing_par.pt1 = 0; // У первой выносной линии стрелка внутри
dimDrawing_par.pt2 = 1; // У второй выносной линии стрелка внутри
dimDrawing_par.shelfDir = 0; // Выносной полки нет
dimDrawing_par.textBase = 0; // Текст размещается в центре
dimDrawing_par.textPos = 0; // Автоматическое размещение текста
// Получаем интерфейс
ldimSource_par = (ksLDimSourceParam)ldim_par.GetSPar();
// Устанавливаем параметры привязки линейного размера
ldimSource_par.basePoint = 1; // Рисуем от первой линии ко второй
// вектор первой точки измеряемого
отрезка
ldimSource_par.dx = 10;
ldimSource_par.dy = 0;

```



```

ldimSource_par.ps = 1; // Размерная линия размещена вертикально
// Координаты первой точки измеряемого отрезка
ldimSource_par.x1 = 200;
ldimSource_par.y1 = 200;
// Координаты второй точки измеряемого отрезка
ldimSource_par.x2 = 200;
ldimSource_par.y2 = 100;
// Получаем интерфейс ksDimTextParam
dimText_par = (ksDimTextParam)ldim_par.GetTPar();
// Устанавливаем параметры размерной надписи
dimText_par.bitFlag = 0; // Будем выводить только номинал
dimText_par.sign = 0; // Значка нет
dimText_par.stringFlag = false;
// Получаем интерфейс ksChar255
char255 = (ksChar255)kompas.GetParamStruct((short)
    Kompas6Constants.StructType2DEnum.ko_Char255);
// Получаем интерфейс ksDynamicArray
dyn_arr = (ksDynamicArray)dimText_par.GetTextArr();
dyn_arr.ksClearArray();
// Строка с номиналом
char255.str = "100";
dyn_arr.ksAddArrayItem(-1, char255);
// Строим сам размер
doc_2d.ksLinDimension(ldim_par);

kompas.Visible = true;

```

Произвольный размер с обрывом

С горизонтальным и вертикальным размером будем считать, что разобрались. Но как быть, если нам нужно проставить размер произвольно наклоненного отрезка? На самом деле это легко решаемая задача. Ниже приводится фрагмент программы, решающей эту задачу:

```

int SHEET_TYPE = 4;
KompasObject kompas;
ksDocument2D doc_2d;
ksChar255 char255;
ksDocumentParam doc_par;
ksSheetOptions sheet_opt;
ksStandartSheet st_sheet;
ksDynamicArray dyn_arr;
ksLDimParam ldim_par;
ksDimDrawingParam dimDrawing_par;
ksLDimSourceParam ldimSource_par;
ksDimTextParam dimText_par;

kompas =
(KompasObject)Marshal.GetActiveObject("Kompas.Application.5");
doc_2d = (ksDocument2D)kompas.Document2D();
doc_par = (ksDocumentParam)kompas.GetParamStruct((short)
    Kompas6Constants.StructType2DEnum.ko_DocumentParam);
doc_par.Init();

```

```

doc_par.author = System.Environment.UserName;
doc_par.fileName = "Standart line size";
doc_par.type = (short)DocumentTypeEnum.ksDocumentDrawing;
doc_2d.ksCreateDocument(doc_par);

sheet_opt = (ksSheetOptions)kompas.GetParamStruct((short)
    Kompas6Constants.StructType2DEnum.ko_SheetOptions);
doc_2d.ksGetDocOptions(SHEET_TYPE, sheet_opt);
sheet_opt.sheetType = false;
st_sheet = (ksStandartSheet)sheet_opt.GetSheetParam(false);
st_sheet.format = 3;
st_sheet.direct = true;
st_sheet.multiply = 1;
sheet_opt.shtType = 1;

```

```

doc_2d.ksSetDocOptions(SHEET_TYPE, sheet_opt);

```

```

doc_2d.ksLineSeg(100, 100, 200, 200, 1);
// Получаем интерфейс ksLDimParam
ldim_par = (ksLDimParam)kompas.GetParamStruct((short)
    Kompas6Constants.StructType2DEnum.ko_LDimParam);
// Получаем параметры интерфейса ksDimDrawingParam
dimDrawing_par = (ksDimDrawingParam)ldim_par.GetDPar();
// Устанавливаем параметры отрисовки размера
dimDrawing_par.ang = 0; // Угол наклона ножки выносной полки
dimDrawing_par.lenght = 0; // Длина ножки выносной полки
dimDrawing_par.pl1 = false; // Отрисовывать первую выносную линию
dimDrawing_par.pl2 = false; // Отрисовывать вторую выносную линию
dimDrawing_par.pt1 = 1; // У первой выносной линии стрелка внутри
dimDrawing_par.pt2 = 1; // У второй выносной линии стрелка внутри
dimDrawing_par.shelfDir = 0; // Выносной полки нет
dimDrawing_par.textBase = 0; // Текст размещается в центре
dimDrawing_par.textPos = 0; // Автоматическое размещение текста
// Получаем интерфейс
ldimSource_par = (ksLDimSourceParam)ldim_par.GetSPar();
// Устанавливаем параметры привязки линейного размера
ldimSource_par.basePoint = 1; // Рисуем от первой линии ко второй
// вектор первой точки измеряемого

```

отрезка

```

ldimSource_par.dx = 0;
ldimSource_par.dy = 10;
ldimSource_par.ps = 2; // Размерная линия размещена вертикально
// Координаты первой точки измеряемого отрезка
ldimSource_par.x1 = 100;
ldimSource_par.y1 = 100;
// Координаты второй точки измеряемого отрезка
ldimSource_par.x2 = 200;
ldimSource_par.y2 = 200;
// Получаем интерфейс ksDimTextParam
dimText_par = (ksDimTextParam)ldim_par.GetTPar();
// Устанавливаем параметры размерной надписи
dimText_par.bitFlag = 1; // Будем выводить только номинал
dimText_par.sign = 0; // Значка нет
dimText_par.stringFlag = false;
// Получаем интерфейс ksChar255

```



```

char255 = (ksChar255)kompas.GetParamStruct((short)
    Kompas6Constants.StructType2DEnum.ko_Char255);
// Получаем интерфейс ksDynamicArray
dyn_arr = (ksDynamicArray)dimText_par.GetTextArr();
dyn_arr.ksClearArray();
// Строка с номиналом
dyn_arr.ksAddArrayItem(-1, char255);
// Строим сам размер
doc_2d.ksLinDimension(ldim_par);

kompas.Visible = true;

```

Горизонтальный размер с выносной полкой

В данной главе приводится пример работы с выносной полкой. Ниже приводится фрагмент программы, демонстрирующий работу с выносной полкой:

```

int SHEET_TYPE = 4;
KompasObject kompas;
ksDocument2D doc_2d;
ksChar255 char255;
ksDocumentParam doc_par;
ksSheetOptions sheet_opt;
ksStandartSheet st_sheet;
ksDynamicArray dyn_arr;
ksLDimParam ldim_par;
ksDimDrawingParam dimDrawing_par;
ksLDimSourceParam ldimSource_par;
ksDimTextParam dimText_par;

kompas =
(KompasObject)Marshal.GetActiveObject("Kompas.Application.5");
doc_2d = (ksDocument2D)kompas.Document2D();
doc_par = (ksDocumentParam)kompas.GetParamStruct((short)
    Kompas6Constants.StructType2DEnum.ko_DocumentParam);
doc_par.Init();
doc_par.author = System.Environment.UserName;
doc_par.fileName = "Standart line size";
doc_par.type = (short)DocumentTypeEnum.ksDocumentDrawing;
doc_2d.ksCreateDocument(doc_par);

sheet_opt = (ksSheetOptions)kompas.GetParamStruct((short)
    Kompas6Constants.StructType2DEnum.ko_SheetOptions);
doc_2d.ksGetDocOptions(SHEET_TYPE, sheet_opt);
sheet_opt.sheetType = false;
st_sheet = (ksStandartSheet)sheet_opt.GetSheetParam(false);
st_sheet.format = 3;
st_sheet.direct = true;
st_sheet.multiply = 1;
sheet_opt.shtType = 1;

doc_2d.ksSetDocOptions(SHEET_TYPE, sheet_opt);

doc_2d.ksLineSeg(100, 100, 200, 100, 1);

```

```

// Получаем интерфейс ksLDimParam
ldim_par = (ksLDimParam)kompas.GetParamStruct((short)
    Kompas6Constants.StructType2DEnum.ko_LDimParam);
// Получаем параметры интерфейса ksDimDrawingParam
dimDrawing_par = (ksDimDrawingParam)ldim_par.GetDPar();
// Устанавливаем параметры отрисовки размера
dimDrawing_par.ang = 60; // Угол наклона ножки выносной полки
dimDrawing_par.lenght = 10; // Длина ножки выносной полки
dimDrawing_par.pl1 = false; // Отрисовывать первую выносную линию
dimDrawing_par.pl2 = false; // Отрисовывать вторую выносную линию
dimDrawing_par.pt1 = 1; // У первой выносной линии стрелка внутри
dimDrawing_par.pt2 = 1; // У второй выносной линии стрелка внутри
dimDrawing_par.shelfDir = 1; // Выносной полки нет
dimDrawing_par.textBase = 0; // Текст размещается в центре
dimDrawing_par.textPos = 0; // Автоматическое размещение текста
// Получаем интерфейс
ldimSource_par = (ksLDimSourceParam)ldim_par.GetSPar();
// Устанавливаем параметры привязки линейного размера
ldimSource_par.basePoint = 1; // Рисуем от первой линии ко второй
// вектор первой точки измеряемого

```

отрезка

```

ldimSource_par.dx = 0;
ldimSource_par.dy = 10;
ldimSource_par.ps = 0; // Размерная линия размещена вертикально
// Координаты первой точки измеряемого отрезка
ldimSource_par.x1 = 100;
ldimSource_par.y1 = 100;
// Координаты второй точки измеряемого отрезка
ldimSource_par.x2 = 200;
ldimSource_par.y2 = 100;
// Получаем интерфейс ksDimTextParam
dimText_par = (ksDimTextParam)ldim_par.GetTPar();
// Устанавливаем параметры размерной надписи
dimText_par.bitFlag = 1; // Будем выводить только номинал
dimText_par.sign = 0; // Значка нет
dimText_par.stringFlag = false;
// Получаем интерфейс ksChar255
char255 = (ksChar255)kompas.GetParamStruct((short)
    Kompas6Constants.StructType2DEnum.ko_Char255);
// Получаем интерфейс ksDynamicArray
dyn_arr = (ksDynamicArray)dimText_par.GetTextArr();
dyn_arr.ksclearArray();
// Строка с номиналом
dyn_arr.kscAddArrayItem(-1, char255);
// Строим сам размер
doc_2d.kscLinDimension(ldim_par);

kompas.Visible = true;

```

Угловой размер

Интерфейс *ksADimSourceParam* служит для описания параметров того, как следует располагать угловой размер. Рассмотрим свойства данного интерфейса:

- *Xc, yc* – координаты центра размерной дуги. Обычно он совпадает с вершиной угла, размер которого проставляется;
- *X1, y1* – координаты точки начала первой выносной линии;
- *X2, y2* – координаты точки начала второй выносной линии;
- *Ang1* – угол наклона к горизонтали первой выносной линии;
- *Ang2* – угол наклона к горизонтали второй выносной линии;
- *Dir* – направление отрисовки размера дуги;
- *Rad* – радиус размерной дуги.

Метод у данного интерфейса всего один – *Init()*.

Параметры углового размера (ksADimParam)

Получить данный интерфейс можно с помощью метода *GetParamStruct()* интерфейса *KompasObject*, для этого в качестве единственного параметра ему нужно передать значение *ko_ADimParam*.

Свойств у интерфейса *ksADimParam* нет, поэтому рассмотрим его методы:

- *GetDPar()* – возвращает интерфейс *ksDimDrawingParam*;
- *GetSPar()* – возвращает интерфейс *ksLDimSourceParam*;
- *GetTPar()* – возвращает интерфейс *ksDimTextParam*;
- *SetDPar()* – устанавливает новый интерфейс *ksDimDrawingParam*, передаваемый ему в качестве единственного параметра;
- *SetSPar()* – устанавливает новый интерфейс *ksLDimSourceParam*, передаваемый ему в качестве единственного параметра;
- *SetTPar()* – устанавливает новый интерфейс *ksLDimSourceParam*, передаваемый ему в качестве единственного параметра.

Построение углового размера

Для непосредственного построения углового размера используется метод *ksAngDimension()* интерфейса *ksDocument2D*. В качестве единственного параметра этому методу передается интерфейс параметров углового размера *ksADimParam*.

Ниже представлен фрагмент программы, осуществляющий простановку углового размера:

```
int SHEET_TYPE = 4;  
KompasObject kompas;  
ksDocument2D doc_2d;  
ksChar255 char255;
```

```

ksDocumentParam doc_par;
ksSheetOptions sheet_opt;
ksStandartSheet st_sheet;
ksDynamicArray dyn_arr;
ksLDimParam ldim_par;
ksDimDrawingParam dimDrawing_par;
ksLDimSourceParam ldimSource_par;
ksDimTextParam dimText_par;
ksADimSourceParam adimSource_par;
ksADimParam adim_par;

kompas =
(KompasObject)Marshal.GetActiveObject("Kompas.Application.5");
doc_2d = (ksDocument2D)kompas.Document2D();
doc_par = (ksDocumentParam)kompas.GetParamStruct((short)
    Kompas6Constants.StructType2DEnum.ko_DocumentParam);
doc_par.Init();
doc_par.author = System.Environment.UserName;
doc_par.fileName = "Standart line size";
doc_par.type = (short)DocumentTypeEnum.ksDocumentDrawing;
doc_2d.ksCreateDocument(doc_par);

sheet_opt = (ksSheetOptions)kompas.GetParamStruct((short)
    Kompas6Constants.StructType2DEnum.ko_SheetOptions);
doc_2d.ksGetDocOptions(SHEET_TYPE, sheet_opt);
sheet_opt.sheetType = false;
st_sheet = (ksStandartSheet)sheet_opt.GetSheetParam(false);
st_sheet.format = 3;
st_sheet.direct = true;
st_sheet.multiply = 1;
sheet_opt.shtType = 1;

doc_2d.ksSetDocOptions(SHEET_TYPE, sheet_opt);

doc_2d.ksLineSeg(100, 100, 100, 200, 1);
doc_2d.ksLineSeg(100, 100, 200, 100, 1);
// Получаем интерфейс ksLDimParam
ldim_par = (ksLDimParam)kompas.GetParamStruct((short)
    Kompas6Constants.StructType2DEnum.ko_LDimParam);
// Получаем параметры интерфейса ksDimDrawingParam
dimDrawing_par = (ksDimDrawingParam)ldim_par.GetDPar();
// Устанавливаем параметры отрисовки размера
dimDrawing_par.ang = 0; // Угол наклона ножки выносной полки
dimDrawing_par.lenght = 0; // Длина ножки выносной полки
dimDrawing_par.pl1 = false; // Отрисовывать первую выносную линию
dimDrawing_par.pl2 = false; // Отрисовывать вторую выносную линию
dimDrawing_par.pt1 = 1; // У первой выносной линии стрелка внутри
dimDrawing_par.pt2 = 1; // У второй выносной линии стрелка внутри
dimDrawing_par.shelfDir = 0; // Выносной полки нет
dimDrawing_par.textBase = 0; // Текст размещается в центре
dimDrawing_par.textPos = 0; // Автоматическое размещение текста
// Получаем интерфейс ksADimParam
adim_par = (ksADimParam)kompas.GetParamStruct((short)
    Kompas6Constants.StructType2DEnum.ko_ADimParam);
// Получаем интерфейс ksADimSourceParam

```

```

adimSource_par = (ksADimSourceParam)adim_par.GetSPar();
// Координаты центра размерной дуги
adimSource_par.xc = 100;
adimSource_par.yc = 100;
// Координаты точки первой выносной линии
adimSource_par.x1 = 100;
adimSource_par.y1 = 100;
// Координаты точки второй выносной линии
adimSource_par.x2 = 100;
adimSource_par.y2 = 100;
// Начальный и конечный угол размерной дуги
adimSource_par.ang1 = 0;
adimSource_par.ang1 = 90;
adimSource_par.dir = -1; // против часовой стрелки
adimSource_par.rad = 15; // радиус размерной дуги
// Получаем интерфейс ksDimTextParam
dimText_par = (ksDimTextParam)adim_par.GetTPar();
// Устанавливаем параметры размерной надписи
dimText_par.bitFlag = 1; // Будем выводить только номинал
dimText_par.sign = 0; // Значка нет
dimText_par.stringFlag = false;
dimText_par.style = 3;
// Получаем интерфейс ksDynamicArray
dyn_arr = (ksDynamicArray)dimText_par.GetTextArr();
dyn_arr.ksClearArray();
// Строим сам угловой размер
doc_2d.ksAngDimension(adim_par);

kompas.Visible = true;

```

Диаметральный размер (ksRDimDrawingParam)

Интерфейс *ksRDimDrawingParam* служит для задания параметров отрисовки диаметрального и радиального размеров. Ниже приведены свойства данного интерфейса:

- *Ang* – угол наклона размерной линии;
- *Pt1* – тип стрелки у первой концевой точки размерной линии;
- *Pt2* – тип стрелки у второй концевой точки размерной линии;
- *shelfDir* – наличие и ориентация выносной полки;
- *textpost* -если строится размер с выносной полкой, тогда задает длину «ножки» выносной полки. Если же размер строится без выносной полки, тогда это свойство определяет расположение текста (обычно по середине), в противном случае задает расстояние между концевой точкой размерной линии и текстом размерной надписи.

Параметры диаметального размера (ksRDimParam)

Интерфейс *ksRDimParam* не имеет свойств, поэтому сразу рассмотрим методы данного интерфейса:

- *GetDPar()* – возвращает интерфейс *ksDimDrawingParam*;
- *GetSPar()* – возвращает интерфейс *ksLDimSourceParam*;
- *GetTPar()* – возвращает интерфейс *ksDimTextParam*;
- *SetDPar()* – устанавливает новый интерфейс *ksDimDrawingParam*, передаваемый ему в качестве единственного параметра;
- *SetSPar()* – устанавливает новый интерфейс *ksLDimSourceParam*, передаваемый ему в качестве единственного параметра;
- *SetTPar()* – устанавливает новый интерфейс *ksLDimSourceParam*, передаваемый ему в качестве единственного параметра.

Получить интерфейс *ksRDimParam* можно с помощью метода *GetParamStruct()* интерфейса *KompasObject* для этого в качестве параметра нужно передать константу *ko_RDimParam*

Ниже приведен фрагмент программы, которая демонстрирует построение диаметального размера:

```
int SHEET_TYPE = 4;
KompasObject kompas;
ksDocument2D doc_2d;
ksDocumentParam doc_par;
ksSheetOptions sheet_opt;
ksStandartSheet st_sheet;
ksDynamicArray dyn_arr;
ksDimTextParam dimText_par;
ksRDimParam rDim_par;
ksRDimDrawingParam rDimDrawing_par;
ksRDimSourceParam rDimSource_par;

kompas =
(KompasObject)Marshal.GetActiveObject("Kompas.Application.5");
doc_2d = (ksDocument2D)kompas.Document2D();
doc_par = (ksDocumentParam)kompas.GetParamStruct((short)
    Kompas6Constants.StructType2DEnum.ko_DocumentParam);
doc_par.Init();
doc_par.author = System.Environment.UserName;
doc_par.fileName = "Диаметральный размер";
doc_par.type = (short)DocumentTypeEnum.ksDocumentDrawing;
doc_2d.kCreateDocument(doc_par);

sheet_opt = (ksSheetOptions)kompas.GetParamStruct((short)
    Kompas6Constants.StructType2DEnum.ko_SheetOptions);
doc_2d.kGetDocOptions(SHEET_TYPE, sheet_opt);
sheet_opt.sheetType = false;
st_sheet = (ksStandartSheet)sheet_opt.GetSheetParam(false);
st_sheet.format = 3;
st_sheet.direct = true;
```



```

st_sheet.multiply = 1;
sheet_opt.shtType = 1;

doc_2d.ksSetDocOptions(SHEET_TYPE, sheet_opt);
doc_2d.ksCircle(150, 150, 30, 1);

// Получаем интерфейс ksRDimParam
rDim_par = (ksRDimParam)kompas.GetParamStruct((short)
    Kompas6Constants.StructType2DEnum.ko_RDimParam);
rDimDrawing_par = (ksRDimDrawingParam)rDim_par.GetDPar();
rDimDrawing_par.ang = 30;
rDimDrawing_par.pt1 = 1;
rDimDrawing_par.pt2 = 1;
rDimDrawing_par.shelfDir = 0;
rDimDrawing_par.textPos = 0;
rDimSource_par = (ksRDimSourceParam)rDim_par.GetSPar();
rDimSource_par.xc = 150;
rDimSource_par.yc = 150;
rDimSource_par.rad = 30;
// Получаем интерфейс ksDimTextParam
dimText_par = (ksDimTextParam)rDim_par.GetTPar();
// Устанавливаем параметры размерной надписи
dimText_par.bitFlag = 1; // Будем выводить только номинал
dimText_par.sign = 1; // Значка нет
dimText_par.stringFlag = false;
dimText_par.style = 3;
// Получаем интерфейс ksDynamicArray
dyn_arr = (ksDynamicArray)dimText_par.GetTextArr();
dyn_arr.ksClearArray();
// Строим сам угловой размер
doc_2d.ksDiamDimension(rDim_par);

kompas.Visible = true;

```

Радиальный размер

Для построения радиального размера используется метод *ksRadDimension()* интерфейса *ksDocument2D*. В качестве единственного параметра данный метод принимает интерфейс параметров радиального размера *ksRDimParam*, рассмотренный выше. Пример фрагмента программы, демонстрирующей построение радиального размера представлен ниже:

```

int SHEET_TYPE = 4;
KompasObject kompas;
ksDocument2D doc_2d;
ksDocumentParam doc_par;
ksSheetOptions sheet_opt;
ksStandartSheet st_sheet;
ksDynamicArray dyn_arr;
ksDimTextParam dimText_par;
ksRDimParam rDim_par;
ksRDimDrawingParam rDimDrawing_par;
ksRDimSourceParam rDimSource_par;

```

```

kompas =
(KompasObject)Marshal.GetActiveObject("Kompas.Application.5");
doc_2d = (ksDocument2D)kompas.Document2D();
doc_par = (ksDocumentParam)kompas.GetParamStruct((short)
    Kompas6Constants.StructType2DEnum.ko_DocumentParam);
doc_par.Init();
doc_par.author = System.Environment.UserName;
doc_par.fileName = "Диаметральный размер";
doc_par.type = (short)DocumentTypeEnum.ksDocumentDrawing;
doc_2d.ksCreateDocument(doc_par);

sheet_opt = (ksSheetOptions)kompas.GetParamStruct((short)
    Kompas6Constants.StructType2DEnum.ko_SheetOptions);
doc_2d.ksGetDocOptions(SHEET_TYPE, sheet_opt);
sheet_opt.sheetType = false;
st_sheet = (ksStandartSheet)sheet_opt.GetSheetParam(false);
st_sheet.format = 3;
st_sheet.direct = true;
st_sheet.multiply = 1;
sheet_opt.shtType = 1;

doc_2d.ksSetDocOptions(SHEET_TYPE, sheet_opt);
doc_2d.ksArcByAngle(150, 150, 30, 0, 90, 1, 1);

// Получаем интерфейс ksRDimParam
rDim_par = (ksRDimParam)kompas.GetParamStruct((short)
    Kompas6Constants.StructType2DEnum.ko_RDimParam);
rDimDrawing_par = (ksRDimDrawingParam)rDim_par.GetDPar();
rDimDrawing_par.ang = 30;
rDimDrawing_par.pt1 = 0;
rDimDrawing_par.pt2 = 1;
rDimDrawing_par.shelfDir = 0;
rDimDrawing_par.textPos = 0;
rDimSource_par = (ksRDimSourceParam)rDim_par.GetSPar();
rDimSource_par.xc = 150;
rDimSource_par.yc = 150;
rDimSource_par.rad = 30;
// Получаем интерфейс ksDimTextParam
dimText_par = (ksDimTextParam)rDim_par.GetTPar();
// Устанавливаем параметры размерной надписи
dimText_par.bitFlag = 1; // Будем выводить только номинал
dimText_par.sign = 3; // Значка нет
dimText_par.stringFlag = false;
dimText_par.style = 3;
// Получаем интерфейс ksDynamicArray
dyn_arr = (ksDynamicArray)dimText_par.GetTextArr();
dyn_arr.ksClearArray();
// Строим сам угловой размер
doc_2d.ksDiamDimension(rDim_par);

kompas.Visible = true;

```


Осевая линия. Параметры математической точки и осевой линии.

Интерфейс *ksMathPointParam* используется для задания параметров некой математической точки. От обычной точки она отличается тем, что не показывается на чертежах.

Получить интерфейс *ksMathPointParam* можно с помощью метода *GetParamStruct()* интерфейса *KompasObject*. Для этого нужно передать методу константу *ko_MathPointParam*.

Свойств у данного интерфейса всего два: *x*, *y*. Можно догадаться, что они задают координаты точки, которую описывает данный интерфейс.

Интерфейс *ksAxisLineParam* служит для задания параметров осевой линии. Получить данный интерфейс можно с помощью метода *GetParamStruct* интерфейса *KompasObject*, для этого в качестве параметра нужно передать методу константу *ko_AxisLineParam*. Свойств у данного интерфейса нет. Поэтому сразу рассмотрим методы данного интерфейса:

- *Init()* – инициализирует интерфейс *ksAxisLineParam*;
- *GetBegPoint* – возвращает интерфейс *ksMathPointParam*, описывающий первую точку осевой линии;
- *GetEndPoint* – возвращает интерфейс *ksMathPointParam*, описывающий вторую точку осевой линии.

Ниже приводится фрагмент программы, демонстрирующий построение осевой линии:

```
int SHEET_TYPE = 4;
KompasObject kompas;
ksDocument2D doc_2d;
ksDocumentParam doc_par;
ksSheetOptions sheet_opt;
ksStandartSheet st_sheet;
ksDynamicArray dyn_arr;
ksDimTextParam dimText_par;
ksRDimParam rDim_par;
ksRDimDrawingParam rDimDrawing_par;
ksRDimSourceParam rDimSource_par;
ksRectangleParam rectangle_par;
ksAxisLineParam axisLine_par;
ksMathPointParam BegPoint, EndPoint;

kompas =
(KompasObject)Marshal.GetActiveObject("Kompas.Application.5");
doc_2d = (ksDocument2D)kompas.Document2D();
doc_par = (ksDocumentParam)kompas.GetParamStruct((short)
    Kompas6Constants.StructType2DEnum.ko_DocumentParam);
doc_par.Init();
```

```

doc_par.author = System.Environment.UserName;
doc_par.fileName = "Диаметральный размер";
doc_par.type = (short)DocumentTypeEnum.ksDocumentDrawing;
doc_2d.ksCreateDocument(doc_par);

sheet_opt = (ksSheetOptions)kompas.GetParamStruct((short)
    Kompas6Constants.StructType2DEnum.ko_SheetOptions);
doc_2d.ksGetDocOptions(SHEET_TYPE, sheet_opt);
sheet_opt.sheetType = false;
st_sheet = (ksStandartSheet)sheet_opt.GetSheetParam(false);
st_sheet.format = 3;
st_sheet.direct = true;
st_sheet.multiply = 1;
sheet_opt.shtType = 1;

doc_2d.ksSetDocOptions(SHEET_TYPE, sheet_opt);
rectangle_par = (ksRectangleParam)kompas.GetParamStruct((short)
    Kompas6Constants.StructType2DEnum.ko_RectangleParam);
rectangle_par.x = 150;
rectangle_par.y = 200;
rectangle_par.ang = 0;
rectangle_par.height = 50;
rectangle_par.width = 100;
rectangle_par.style = 1;
doc_2d.ksRectangle(rectangle_par, 0);

axisLine_par = (ksAxisLineParam)kompas.GetParamStruct((short)
    Kompas6Constants.StructType2DEnum.ko_AxisLineParam);
BegPoint = (ksMathPointParam)axisLine_par.GetBegPoint();
BegPoint.x = 150;
BegPoint.y = 225;
EndPoint = (ksMathPointParam)axisLine_par.GetEndPoint();
EndPoint.x = 250;
EndPoint.y = 225;
doc_2d.ksAxisLine(axisLine_par);

kompas.Visible = true;

```

Обозначение центра (ksCentreParam)

Интерфейс *ksCentreParam* служит для задания параметров обозначения центра. Получить его можно с помощью метода *GetParamStruct* интерфейса *KompasObject*, для этого нужно передать методу константу *ko_CentreParam*. Рассмотрим свойства данного интерфейса:

- *Angle* – согласно документации КОМПАС это угол наклона обозначения центра к горизонтали.
- *baseCurve* – числовой идентификатор геометрического объекта, центр которого мы обозначаем
- *lenXmTail* – длина горизонтальной полуоси в отрицательном направлении оси X.

- *lenXpTail* – длина горизонтальной полуоси в положительном направлении оси X.
- *lenYmTail* – длина вертикальной полуоси в отрицательном направлении оси Y.
- *lenYpTail* – длина вертикальной полуоси в положительном направлении оси Y.
- *Type* – тип обозначения центра. Данное свойство определяет, как следует обозначить центр. Возможные значения данного свойства и их интерпретация представлены в таблице ниже.

Значение свойства	Описание
0	Обозначать в виде маленького крестика
1	В виде одной (горизонтальной) оси
2	Две оси

- *X и y* – координаты точки пересечения осей обозначения центра.

Ниже приводится фрагмент программы, демонстрирующей построение обозначения центра:

```
int SHEET_TYPE = 4;
int CircleId;
KompasObject kompas;
ksDocument2D doc_2d;
ksDocumentParam doc_par;
ksSheetOptions sheet_opt;
ksStandartSheet st_sheet;
ksCentreParam centre_par;

kompas =
(KompasObject)Marshal.GetActiveObject("Kompas.Application.5");
doc_2d = (ksDocument2D)kompas.Document2D();
doc_par = (ksDocumentParam)kompas.GetParamStruct((short)
    Kompas6Constants.StructType2DEnum.ko_DocumentParam);
doc_par.Init();
doc_par.author = System.Environment.UserName;
doc_par.fileName = "Диаметральный размер";
doc_par.type = (short)DocumentTypeEnum.ksDocumentDrawing;
doc_2d.ksCreateDocument(doc_par);

sheet_opt = (ksSheetOptions)kompas.GetParamStruct((short)
    Kompas6Constants.StructType2DEnum.ko_SheetOptions);
doc_2d.ksGetDocOptions(SHEET_TYPE, sheet_opt);
sheet_opt.sheetType = false;
st_sheet = (ksStandartSheet)sheet_opt.GetSheetParam(false);
st_sheet.format = 3;
st_sheet.direct = true;
st_sheet.multiply = 1;
sheet_opt.shtType = 1;

doc_2d.ksSetDocOptions(SHEET_TYPE, sheet_opt);
```

```

CircleId = doc_2d.ksCircle(150, 150, 30, 1);
centre_par = (ksCentreParam)kompas.GetParamStruct((short)
    Kompas6Constants.StructType2DEnum.ko_CentreParam);
centre_par.angle = 0;
centre_par.baseCurve = CircleId;
centre_par.lenXmTail = 0;
centre_par.lenXpTail = 0;
centre_par.lenYmTail = 0;
centre_par.lenYpTail = 0;
centre_par.standXmTail = true;
centre_par.standXpTail = true;
centre_par.standYmTail = true;
centre_par.standYpTail = true;
centre_par.type = 2;
centre_par.x = 150;
centre_par.y = 150;
doc_2d.ksCentreMarker(centre_par);

kompas.Visible = true;

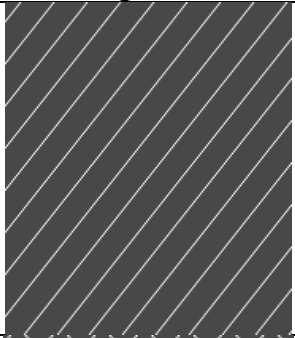
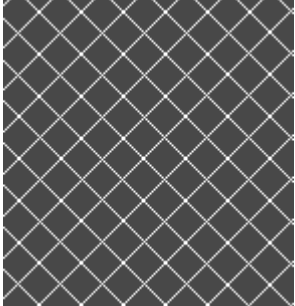
```

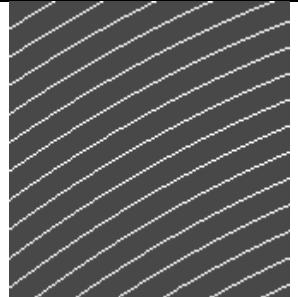

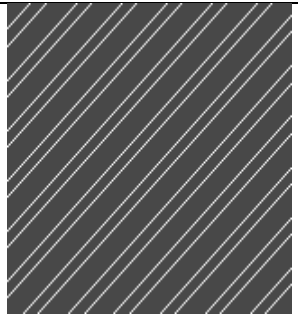
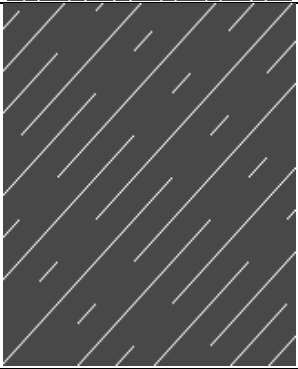
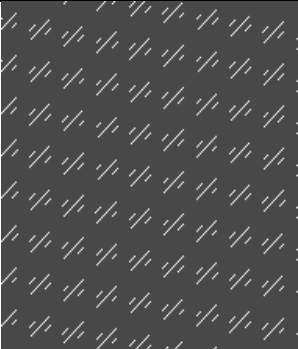
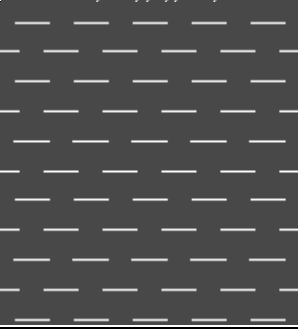
Штриховка. Способ первый

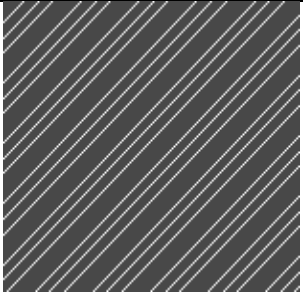
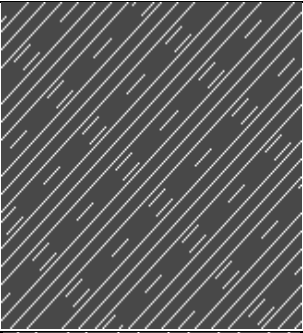
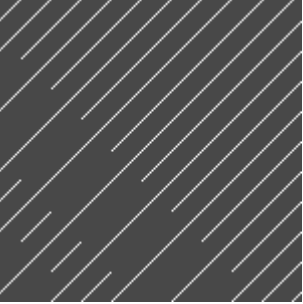
Построение штриховки осуществляется в три этапа:


- 1) Осуществляется инициализация процесса построения штриховки. Для этого вызывается метод *ksHatch()* интерфейса *ksDocument2D*. На этом этапе определяются основные параметры штриховки (стиль, шаг, угол наклона штриховой линии).
- 2) Обозначаются границы заштрихованной области.
- 3) Осуществляется остановка процесса построения штриховки. Для этого вызывается метод *ksEndObj()* интерфейса *ksDocument2D*.

В таблице ниже представлены допустимые стили штриховки с их описаниями.

Номер стиля	Материал	Изображение
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	
----	-------	--

Ниже представлен фрагмент программы, демонстрирующей заштриховывание замкнутой области:

```

int SHEET_TYPE = 4;
KompasObject kompas;
ksDocument2D doc_2d;
ksDocumentParam doc_par;
ksSheetOptions sheet_opt;
ksStandartSheet st_sheet;
ksRectangleParam rectangle_par;

kompas =
(KompasObject)Marshal.GetActiveObject("Kompas.Application.5");
doc_2d = (ksDocument2D)kompas.Document2D();
doc_par = (ksDocumentParam)kompas.GetParamStruct((short)
    Kompas6Constants.StructType2DEnum.ko_DocumentParam);
doc_par.Init();
doc_par.author = System.Environment.UserName;
doc_par.fileName = "Штриховка - способ первый";
doc_par.type = (short)DocumentTypeEnum.ksDocumentDrawing;
doc_2d.kCreateDocument(doc_par);

sheet_opt = (ksSheetOptions)kompas.GetParamStruct((short)
    Kompas6Constants.StructType2DEnum.ko_SheetOptions);
doc_2d.kGetDocOptions(SHEET_TYPE, sheet_opt);
sheet_opt.sheetType = false;
st_sheet = (ksStandartSheet)sheet_opt.GetSheetParam(false);
st_sheet.format = 3;
st_sheet.direct = true;
st_sheet.multiply = 1;
sheet_opt.shtType = 1;

doc_2d.kSetDocOptions(SHEET_TYPE, sheet_opt);

rectangle_par = (ksRectangleParam)kompas.GetParamStruct((short)
    Kompas6Constants.StructType2DEnum.ko_RectangleParam);
rectangle_par.x = 150;
rectangle_par.y = 150;
rectangle_par.ang = 0;
rectangle_par.height = 30;
rectangle_par.width = 70;
rectangle_par.style = 1;
doc_2d.kRectangle(rectangle_par, 0);

```

```
doc_2d.ksHatch(14, 45, 1, 0, 101, 101);  
doc_2d.ksRectangle(rectangle_par, 0);  
doc_2d.ksEndObj();
```

```
kompas.Visible = true;
```

Штриховка. Способ второй

В данном способе построения штриховки заштриховываемая область задается иначе. Здесь эта область должна реально существовать на чертеже. Для получения числового идентификатора области используется метод *ksMakeEncloseContours(int dr, x, y)* интерфейса *ksDocument2D*.

С параметрами *x* и *y* всё ясно, они задают координаты произвольной точки внутри замкнутого контура, идентификатор которого мы хотим получить. Параметр *gr* определяет числовой идентификатор группы геометрических объектов, подмножеством которой должны являться объекты группы, идентификатор которой мы хотим получить.

Интерфейс *ksHatchParam* используется для задания параметров штриховки. Получить его можно с помощью метода *GetParamStruct()* интерфейса *KompasObject*. Рассмотрим свойства данного интерфейса:

- *Ang* – угол наклона линий штриховки к горизонтали;
- *Boundaries* – числовой идентификатор области, которую нужно заштриховать. Его можно получить с помощью метода *ksMakeEncloseCountours* интерфейса *ksDocument2D*;
- *Color* – цвет линий штриховки в формате BGR (обратная от RGB);
- *Sheeting* – согласно документации «тип угла штриховки». Но на практике игнорируется;
- *Step* – шаг штриховки;
- *Style* – стиль штриховки;
- *Width* – согласно документации «ширина полосы штриховки», но на практике игнорируется;
- *X u y* – также игнорируются.

Ниже приведен фрагмент программы, демонстрирующей построение штриховки данным методом:

```
int SHEET_TYPE = 4;  
KompasObject kompas;  
ksDocument2D doc_2d;  
ksDocumentParam doc_par;  
ksSheetOptions sheet_opt;  
ksStandartSheet st_sheet;  
ksRectangleParam rectangle_par;  
ksHatchParam hatch_par;  
int GroupId;
```



```

kompas =
(KompasObject)Marshal.GetActiveObject("Kompas.Application.5");
doc_2d = (ksDocument2D)kompas.Document2D();
doc_par = (ksDocumentParam)kompas.GetParamStruct((short)
    Kompas6Constants.StructType2DEnum.ko_DocumentParam);
doc_par.Init();
doc_par.author = System.Environment.UserName;
doc_par.fileName = "Штриховка - способ второй";
doc_par.type = (short)DocumentTypeEnum.ksDocumentDrawing;
doc_2d.ksCreateDocument(doc_par);

sheet_opt = (ksSheetOptions)kompas.GetParamStruct((short)
    Kompas6Constants.StructType2DEnum.ko_SheetOptions);
doc_2d.ksGetDocOptions(SHEET_TYPE, sheet_opt);
sheet_opt.sheetType = false;
st_sheet = (ksStandardSheet)sheet_opt.GetSheetParam(false);
st_sheet.format = 3;
st_sheet.direct = true;
st_sheet.multiply = 1;
sheet_opt.shtType = 1;

doc_2d.ksSetDocOptions(SHEET_TYPE, sheet_opt);

rectangle_par = (ksRectangleParam)kompas.GetParamStruct((short)
    Kompas6Constants.StructType2DEnum.ko_RectangleParam);
rectangle_par.x = 150;
rectangle_par.y = 150;
rectangle_par.ang = 0;
rectangle_par.height = 30;
rectangle_par.width = 70;
rectangle_par.style = 1;
doc_2d.ksRectangle(rectangle_par, 0);

GroupId = doc_2d.ksMakeEncloseContours(0, 151, 151);
hatch_par = (ksHatchParam)kompas.GetParamStruct((short)
    Kompas6Constants.StructType2DEnum.ko_HatchParam);
hatch_par.ang = 45;
hatch_par.boundaries = GroupId;
hatch_par.color = 000000;
hatch_par.sheeting = 0;
hatch_par.step = 1;
hatch_par.style = 1;
hatch_par.width = 0;
hatch_par.x = 101;
hatch_par.y = 101;
doc_2d.ksHatchByParam(hatch_par);

kompas.Visible = true;

```

Заштриховка двух и более непересекающихся областей

Очень часто на разрезах встречаются сквозные отверстия. Из-за них вся заштриховываемая область разбивается на две или более областей. Ниже приводится фрагмент программы решающей данную задачу:

```
int SHEET_TYPE = 4;
KompasObject Kompas;
ksDocument2D doc_2d;
ksDocumentParam doc_par;
ksSheetOptions sheet_opt;
ksStandartSheet st_sheet;
ksRectangleParam rectangle_par;
ksHatchParam hatch_par;
int GroupID, RectID;

Kompas =
(KompasObject)Marshal.GetActiveObject("Kompas.Application.5");
doc_2d = (ksDocument2D)Kompas.Document2D();
doc_par = (ksDocumentParam)Kompas.GetParamStruct((short)
    Kompas6Constants.StructType2DEnum.ko_DocumentParam);
doc_par.Init();
doc_par.author = System.Environment.UserName;
doc_par.fileName = "Штриховка - способ второй";
doc_par.type = (short)DocumentTypeEnum.ksDocumentDrawing;
doc_2d.kCreateDocument(doc_par);

sheet_opt = (ksSheetOptions)Kompas.GetParamStruct((short)
    Kompas6Constants.StructType2DEnum.ko_SheetOptions);
doc_2d.kGetDocOptions(SHEET_TYPE, sheet_opt);
sheet_opt.sheetType = false;
st_sheet = (ksStandartSheet)sheet_opt.GetSheetParam(false);
st_sheet.format = 3;
st_sheet.direct = true;
st_sheet.multiply = 1;
sheet_opt.shtType = 1;

doc_2d.kSetDocOptions(SHEET_TYPE, sheet_opt);

rectangle_par = (ksRectangleParam)Kompas.GetParamStruct((short)
    Kompas6Constants.StructType2DEnum.ko_RectangleParam);
rectangle_par.x = 150;
rectangle_par.y = 150;
rectangle_par.ang = 0;
rectangle_par.height = 30;
rectangle_par.width = 70;
rectangle_par.style = 1;
doc_2d.kRectangle(rectangle_par, 0);

GroupID = doc_2d.kMakeEncloseContours(0, 151, 151);
rectangle_par.x = 170;
RectID = doc_2d.kRectangle(rectangle_par, 0);
doc_2d.kAddObjGroup(GroupID, RectID);
hatch_par = (ksHatchParam)Kompas.GetParamStruct((short)
```

```
Kompas6Constants.StructType2DEnum.ko_HatchParam);  
hatch_par.ang = 45;  
hatch_par.boundaries = GroupID;  
hatch_par.color = 000000;  
hatch_par.sheeting = 0;  
hatch_par.step = 1;  
hatch_par.style = 1;  
hatch_par.width = 0;  
hatch_par.x = 101;  
hatch_par.y = 101;  
doc_2d.kshatchByParam(hatch_par);  
  
kompas.Visible = true;
```

Задание.

За основу взять объект из прошлой лабораторной работы. Проставить нужные размеры, нанести штриховку в зависимости от типа материала, а также проставить осевую линию, если это необходимо