

## Лабораторная работа №4.

### Документ, содержащий 3D-модель или сборку (ksDocument3D)

Интерфейс *ksDocument3D* служит для описания документа, содержащего 3D-модель или сборку. Получить этот интерфейс можно двумя способами

Первый способ: с помощью метода *Document3D()* интерфейса *KompasObject*. Данный метод не имеет входных параметров и возвращает интерфейс *ksDocument3D*.

Второй способ: с помощью метода *ActiveDocument3D()* интерфейса *KompasObject*. Данный метод не имеет входных параметров и возвращает интерфейс *ksDocument3D* текущего документа, содержащего 3D-модель или сборку.

### Открытие существующей 3D модели (сборки)

Для открытия существующей 3D-модели (сборки) используется метод *Open()* интерфейса *ksDocument3D*. Ниже представлен прототип данного метода:

*Open(string filename, bool invisible)*

- *Filename* – определяет строку с именем открываемого файла
- *Invisible* – определяет видимость открываемого документа. Если *true*, то документ становится невидимым, иначе (*false*) видимым. Ниже приводится фрагмент программы, демонстрирующей открытие существующего документа:

```
public class open_doc_3d
{
    public static KompasObject kompas;
    public static ksDocument3D doc_3d;

    public static void Main()
    {
        kompas =
        (KompasObject)Marshal.GetActiveObject("Kompas.Application.5");
        doc_3d = (ksDocument3D)kompas.Document3D();
        doc_3d.Open("D:/kompas-
project/C#/laboratory_5/test_detail.m3d");

        kompas.Visible = true;
    }
}
```

### Создание новой 3D-модели (сборки)

Для создания новой 3D-модели (сборки) используется метод *Create()* интерфейса *ksDocument3D*. Ниже приводится прототип данного метода:

*Create(bool Visible, typeDoc)*

- Параметр *Visible* определяет видимость создаваемого документа. Если значение равно *true*, то документ невидим, иначе (*false*) документ видим.

- Параметр *typeDoc* определяет тип создаваемого документа. Если значение равно *true*, то создается деталь (3D-модель), иначе (*false*) создается сборка.

Ниже приводится фрагмент программы, демонстрирующей создание новой 3D-модели (детали):

```
public class create_doc_3d
{
    public static KompasObject kompas;
    public static ksDocument3D doc_3d;
    public static void Main()
    {
        kompas =
(KompasObject)Marshal.GetActiveObject("Kompas.Application.5");
        doc_3d = (ksDocument3D)kompas.Document3D();
        doc_3d.Create(false, false);
        kompas.Visible = true;
    }
}
```

### Заккрытие 3D-модели (сборки)

Для закрытия документа, содержащего 3D-модель или сборку, используется метод Clode() интерфейса ksDocument3D. Данный метод не имеет входных параметров. Нужно быть осторожным с данным методом, так как при его вызове изменения в документе не сохраняются. Ниже приводится фрагмент прорагммы, демонстрирующей закрытие 3D-модели:

```
public class close_doc_3d
{
    public static KompasObject kompas;
    public static ksDocument3D doc_3d;

    public static void Main()
    {
        kompas =
(KompasObject)Marshal.GetActiveObject("Kompas.Application.5");
        doc_3d = (ksDocument3D)kompas.Document3D();
        doc_3d.Create(false, false);
        kompas.Visible = true;
        System.Threading.Thread.Sleep(2000);
        doc_3d.close();
    }
}
```

### Сохранение 3D-модели (сборки)

Для сохранения 3D-модели (сборки) используется метод *SaveAs()* интерфейса *ksDocument3D*. В качестве единственного параметра данный метод принимает строку с наименованием файла, в котором следует сохранить 3D-модель (сборку).

Ниже приводится ключевой фрагмент программы, демонстрирующей сохранение 3D-модели:

```
public class save_doc_3d
{
    public static KompasObject kompas;
```

```

public static ksDocument3D doc_3d;

public static void Main()
{
    Kompas =
(KompasObject)Marshal.GetActiveObject("Kompas.Application.5");
    doc_3d = (ksDocument3D)Kompas.Document3D();
    doc_3d.Create(false, false);

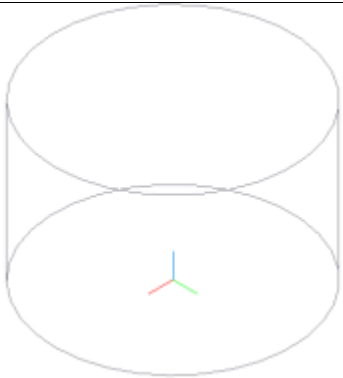

doc_3d.SaveAs("C:/Users/LeroiKudaibergenov/Desktop/detail1.m3d");
    doc_3d.close();
    Kompas.Quit();
}
}

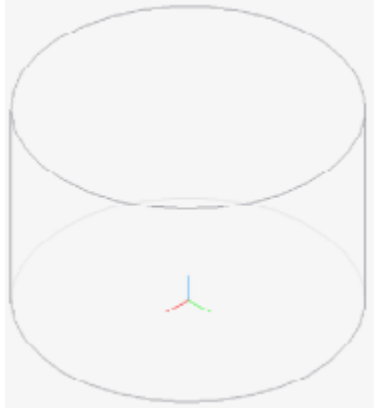
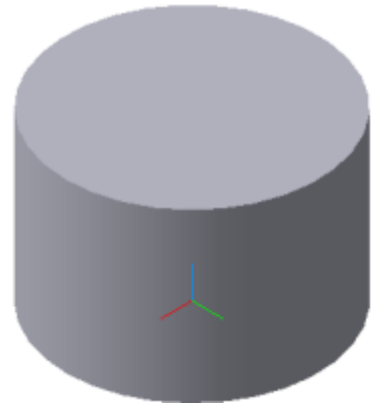
```

## Управление отображением 3D-модели

Основная часть свойств интерфейса *ksDocument3D* позволяет настраивать отображение 3D-модели (сборки). Ниже представлены свойства, отвечающие за отображение 3D-модели (сборки):

- *drawMode* – тип отображени 3D-модели (сборки). Принимает одно из значений, которые перечислены в таблице ниже;

Тип	Числовое значение	Описание	Пример
Vm_Wireframe	0	Каркас	
Vm_HiddenRemoved	1	Без невидимых линий	

Vm_HiddenThin	2	Невидимые линии тонкие	
Vm_shaded	3	полутоновый	

- ***hideAllPlaces*** – задает видимость начала координат. Если значение true, то начало координат невидимо, иначе (false) начало координат видимо;

- ***hideAllPlanes*** – задает видимость конструктивных (вспомогательных) плоскостей. Если true, то плоскости не видны, иначе (false) видны;

- ***hideAllAxis*** – задает видимость конструктивных (вспомогательных) осей. Если true, то оси не видны, иначе (false) видны;

- ***perspective*** – задает отображение 3D-модели в перспективной проекции. Если true, то перспективная проекция включена, иначе (false) перспективная проекция отключена.

- ***shadedWireframe*** – для полутонового изображения 3D-модели включает отображения каркаса. Если true, то модель отображается с каркасом, иначе (false) без каркаса.

### 3D-модель (ksPart)

Для описания самой 3D-модели (детали) используется интерфейс ksPart. Получить его можно с помощью метода GetPar интерфейса ksDocument3D. Данный метод требует указания номера получаемой 3D-модели или её типа. В таблице ниже перечислены существующие типы моделей.

Тип 3D-модели	Числовое значение	Описание
pInPlace_part	-4	3D-модель, редактируемая на месте
pNew_Part	-3	Новая 3D-модель
pEdit_Part	-2	Редактируемая 3D-модель
pTop_Part	-1	Главная 3D-модель, в составе которой находится новая или редактируемая или указанная 3D-модель

Стоит обратить внимание на то, что значения в данной таблице меньше нуля. Дело в том, что если в качестве параметра метода ***GetPart*** передать положительное число, то данное число интерпретируется как порядковый номер 3D-модели в сборке.

### Объект (***ksEntity***)

Интерфейс ***ksEntity*** служит для описания отдельно взятого объекта. В качестве объекта могут выступать эскиз, ось, плоскость, ломанная и тд. Ниже представлены свойства данного интерфейса:

- ***type*** – тип объекта, описываемого интерфейса ***ksEntity***. Он задается при получении этого интерфейса и сохраняется в качестве значения этого свойства;
- ***name*** – наименование объекта, которое будет отображаться в дереве построения. Изменять значение данного свойства нужно до создания объекта;
- ***hidden*** – видимость объекта. Если значение равно ***true***, то объект невидимый, иначе (***false***) объект видимый.

Далее рассмотрим основные методы данного интерфейса:

- ***Create()*** – создает объект;
- ***GetDefinition()*** – получить интерфейс параметров объекта;
- ***GetParent()*** – получить интерфейс ***ksPart*** 3D-модели (детали), которой принадлежит данный объект;
- ***IsCreated()*** – проверить существование объекта;
- ***IsIt()*** – проверить объект на принадлежность заданному типу;
- ***Update()*** – изменить свойства объекта. Данный метод нужен для перестроения объекта после изменения его свойств.

## Объекты, создаваемые системой КОМПАС по умолчанию

Для получения объекта, созданного системой КОМПАС по умолчанию, используется метод *GetDefaultEntity()* интерфейса *ksPart*. В качестве единственного параметра он принимает номер того объекта, интерфейс которого мы хотим получить. В таблице ниже перечислены все объекты, создаваемые системой КОМПАС по умолчанию и их номера.

Номер объекта	Числовое значение	Описание
o3d_planeXOY	1	Плоскость XOY
o3d_planeXOZ	2	Плоскость XOZ
o3d_planeYOZ	3	Плоскость YOZ
o3d_pointCS	4	Точка начала системы координат
o3d_axisOX	71	Ось OX
o3d_axisOY	72	Ось OY
o3d_axisOZ	73	Ось OZ

## Создание новых объектов

Для создания нового объекта используется метод *NewEntity()* интерфейса *ksPart*. В качестве единственного параметра данный метод принимает номер типа создаваемого объекта. В таблице ниже перечислены основные типы объектов, описываемых интерфейсом *ksEntity*.

Номер типа объекта	Числовое значение	Описание
O3d_sketch	5	эскиз
O3d_axis2Planes	9	ось по двум плоскостям
O3d_axis2Points	10	ось по двум точкам
O3d_axisConeFace	11	ось конической грани
O3d_axisEdge	12	ось, проходящая через ребро
O3d_axisOperation	13	ось операции
O3d_planeOffset	14	смещённая плоскость

O3d__planeAngle	15	плоскость под углом
O3d__plane3Points	16	плоскость по трем точкам
O3d_planeNormal	17	нормальная плоскость
O3d_planeTangent	18	касательная плоскость
O3d_planeEdgePoint	19	плоскость через ребро и вершину
O3d_planeParallel	20	плоскость через вершину параллельно другой плоскости
O3d_planePerpendicular	21	плоскость через вершину перпендикулярно ребру
O3d_planeLineToEdge	22	плоскость через ребро параллельно (перпендикулярно) другому ребру
O3d_planeLineToPlane	23	плоскость через ребро параллельно (перпендикулярно) грани
o3d_baseExtrusion	24	базовая операция выдавливания
o3d_bossExtrusion	25	приклеивание выдавливанием
o3d_cutExtrusion	26	вырезать выдавливанием
o3d_baseRotated	27	базовая операция вращения
o3d_bossRotated	28	приклеивание вращением
o3d_cutRotated	29	вырезать вращением
o3d_baseLoft	30	базовая операция по сечениям
o3d_bossLoft	31	приклеивание по сечениям

o3d_cutLoft	32	вырезать по сечениям
o3d_chamfer	33	операция «фаска»
o3d_fillet	34	операция «скругление»
o3d_meshCopy	35	операция копирования по сетке
o3d_circularCopy	36	операция копирования по концентрической сетке
o3d_curveCopy	37	операция копирования по кривой
o3d_circPartArray	38	операция массив по концентрической сетке для сборки
o3d_meshPartCopy	39	операция массив по сетке для сборки
o3d_curvePartArray	40	операция массив по кривой для сборки
o3d_derivePartArray	41	операция массив по образцу для сборки
o3d_incline	42	операция «уклон»
o3d_shellOperation	43	операция «оболочка»
o3d_ribOperation	44	операция «ребро жесткости»
o3d_baseEvolution	45	кинематическая операция
o3d_bossEvolution	46	приклеить кинематически
o3d_cutEvolution	47	вырезать кинематически
o3d_mirrorOperation	48	операция «зеркальный массив»
o3d_mirrorAllOperation	49	операция «зеркально отразить все»
o3d_cutByPlane	50	операция «сечение»



		поверхностью»
o3d_cutBySketch	51	операция «сечение эскизом»
o3d_holeOperation	52	отверстие
o3d_polyline	53	ломаная
o3d_conicSpiral	54	коническая спираль
o3d_spline	55	сплайн
o3d_cylindricSpiral	56	цилиндрическая спираль
o3d_importedSurface	57	импортированная поверхность
o3d_thread	58	условное изображение резьбы
o3d_EvolutionSurface	59	кинематическая поверхность
o3d_ExtrusionSurface	60	поверхность выдавливания
o3d_RotatedSurface	61	поверхность вращения
o3d_LoftSurface	62	поверхность по сечениям
o3d_MacroObject	63	макрообъект 3D
o3d_UnionComponents	64	операция объединения компонентов
o3d_MoldCavity	65	операция вычитания компонентов

### Эскиз. Параметры Эскиза (ksSketchDefinition)

Интерфейс *ksSketchDefinition* служит для описания параметров объекта «эскиз». Получить этот интерфейс можно с помощью метода *GetDefinition()* интерфейса *ksEntity*, описывающего эскиз. Ниже рассмотрены методы данного интерфейса:

- *BeginEdit()* – открывает режим редактирования эскиза. В случае успеха данный метод возвращает интерфейс *ksDocument2D*, с помощью которого и происходит построение самого эскиза;

- **EndEdit()** – закрывает режим редактирования эскиза. В случае успеха возвращает значение *true*;

- **GetLocation()** – возвращает смещение центра системы координат эскиза относительно проекции центра системы координат модели на плоскость эскиза. В случае успеха возвращает значение *true*;

- **GetPlane()** – возвращает интерфейс *ksEntity*, описывающий плоскость, в которой расположен эскиз;

- **SetLocation()** – устанавливает новое смещение центра системы координат эскиза относительно проекции центра системы координат модели на плоскости эскиза. В случае успеха данный метод возвращает значение *true*;

- **SetPlane()** – установить плоскость, в которой расположен эскиз. В качестве единственного параметра данный метод принимает интерфейс *ksEntity*, описывающий плоскость, в которой будет располагаться эскиз. В случае успеха данный метод возвращает значение *ццквцц*.

Ниже приводится фрагмент программы, демонстрирующий построение простого эскиза в плоскости XOY:

```
public class standart_sketch
{
    public static int pTop_Part = -1;
    public static short o3d_planeXOY = 1;
    public static short o3d_sketch = 5;

    public static KompasObject kompas;
    public static ksDocument3D doc_3d;
    public static ksPart part;
    public static ksEntity entityPlane, entityDraw;
    public static ksSketchDefinition sketch_def;
    public static ksDocument2D doc_2d;

    public static void Main()
    {
        // подключаемся к компасу
        kompas =
        (KompasObject)Marshal.GetActiveObject("Kompas.Application.5");
        // получаем интерфейс документа-модели
        doc_3d = (ksDocument3D)kompas.Document3D();
        // создаем документ и модель
        doc_3d.Create(false, true);
        // получаем интерфейс 3d модели
        part = (ksPart)doc_3d.GetPart(pTop_Part);
        // получаем интерфейс объекта эскиз
        entityDraw = (ksEntity)part.NewEntity(o3d_sketch);
        // получаем интерфейс параметров эскиза
        sketch_def = (ksSketchDefinition)entityDraw.GetDefinition();
        // получаем интерфейс объекта "плоскость XOY"
        entityPlane = (ksEntity)part.GetDefaultEntity(o3d_planeXOY);
        // устанавливаем плоскость XOY базовой для системы
        sketch_def.SetPlane(entityPlane);
    }
}
```

```

    // создаем эскиз
    entityDraw.Create();
    // входим в режим редактирования эскиза
    doc_2d = (ksDocument2D)sketch_def.BeginEdit();
    // строим окружность
    doc_2d.ksCircle(0, 0, 30, 1);
    // выходим из режима редактирования эскиза
    sketch_def.EndEdit();
    kompas.Visible = true;
}
}

```

### Операция выдавливания. Параметры выдавливания (ksBaseExtrusion Definition)

В случае, если интерфейс ksEntity описывает объект «базовая операция выдавливания» (o3d\_baseExtrusion), то его метод GetDefinition() возвращает интерфейс ksBaseExtrusionDefinition. У данного интерфейса всего одно свойство:

- directionType – задает направление выдавливания. В таблице представлены допустимые значения данного свойства.

Направление	Числовое значение	Описание
dtNormal	0	Прямое направление
dtReverse	1	Обратное направление
dtBoth	2	В обе стороны
dtMiddlePlane	3	От средней плоскости

Далее рассмотрим методы интерфейса ksBaseExtrusionDefinition:

- GetSketch() – возвращает интерфейс ksEntity, описывающий эскиз операции выдавливания;
- SetSideParam(bool side1; int type\_; double depth, draftValue; bool draftOutward) – установить параметры выдавливания.
  - 1) Side1 – направление выдавливания. Если true, то строится прямое выдавливание, иначе (false), то строится обратное выдавливание.
  - 2) Type\_ - тип выдавливания. Значения данного параметра представлены в таблице ниже.

Тип выдавливания	Числовое значение	Описание
etBlind	0	Строго на глубину

etThroughAll	1	Через всю деталь
etUpToVertexTo	2	На расстояние до вершины
etUpToVertexFrom	3	На расстояние за вершину
etUpToSurfaceTo	4	На расстояние до поверхности
etUpToSurfaceFrom	5	На расстояние за поверхность
etUpToNearSurface	6	До ближайшей поверхности

- Depth – глубина выдавливания.
- draftValue – угол уклона. Он исчерпывается между нормалью к плоскости эскиза операции выдавливания и линией, вдоль которой выдавливается эскиз.
- draftOutward – направление уклона. Если значение равно true, то уклон наружу, иначе (false) уклон внутрь.
- SetSketch – Установить эскиз операции выдавливания.

Ниже приводится фрагмент программы, демонстрирующий использование операцию выдавливания:

```

public static int pTop_Part = -1;
public static short o3d_planeXOY = 1;
public static short o3d_sketch = 5;
public static short o3d_baseExtrusion = 24;
public static short etBlind = 0;
public static int vm_Shaded = 3;

public static KompasObject kompas;
public static ksDocument3D doc_3d;
public static ksPart part;
public static ksEntity entityDraw, entityPlane, entityExtrusion;
public static ksSketchDefinition sketchDef;
public static ksDocument2D doc_2d;
public static ksBaseExtrusionDefinition baseExtrusionDef;
public static void Main()
{
    // подключаемся к Компасу
    kompas =
(KompasObject)Marshal.GetActiveObject("Kompas.Application.5");
    // получаем интерфейс документ-модели
    doc_3d = (ksDocument3D)kompas.Document3D();
    // создаем документ-модель
    doc_3d.Create(false, true);
    // получаем интерфейс компонента
    part = (ksPart)doc_3d.GetPart(pTop_Part);

```

```

// получаем интерфейс объекта
entityDraw = (ksEntity)part.NewEntity(o3d_sketch);
// получаем интерфейс параметров эскиза
sketchDef = (ksSketchDefinition)entityDraw.GetDefinition();
// получаем интерфейс объекта "плоскость XOY"
entityPlane = (ksEntity)part.GetDefaultEntity(o3d_planeXOY);
// устанавливаем плоскость XOY базовой для эскиза
sketchDef.SetPlane(entityPlane);
// создаем эскиз
entityDraw.Create();
// входим в режим редактирования эскиза
doc_2d = (ksDocument2D)sketchDef.BeginEdit();
// строим окружность
doc_2d.ksCircle(0, 0, 15, 1);
// выходим из режима редактирования эскиза
sketchDef.EndEdit();
// получаем интерфейса объекта "операция выдавливания"
entityExtrusion = (ksEntity)part.NewEntity(o3d_baseExtrusion);
// получаем интерфейс параметров операции "выдавливание"
baseExtrusionDef =
(ksBaseExtrusionDefinition)entityExtrusion.GetDefinition();
// Устанавливаем параметры операции выдавливания
baseExtrusionDef.SetSideParam(true, etBlind, 20, 0, true);
// устанавливаем эскиз операции выдавливания
baseExtrusionDef.SetSketch(entityDraw);
// создаем операцию выдавливания
entityExtrusion.Create();
// устанавливаем полутонное изображение модели
doc_3d.drawMode = vm_Shaded;
// включаем отображение каркаса
doc_3d.shadedWireframe = true;
// делаем компас видимым
kompas.Visible = true;
}

```

## Смещенная плоскость. Параметры смещенной плоскости (ksPlaneOffsetDefinition)

Смещенная плоскость представляет собой плоскость, расположенную параллельно заданной плоскости (она называется базовой) и удаленную от неё на фиксированное расстояние (смещение).

Параметры смещенной плоскости описываются интерфейсом ksPlaneOffsetDefinition. Этот интерфейс возвращает метод GetDefinition() интерфейса ksEntity. Ниже представлены свойства интерфейса ksPlaneOffsetDefinition:

- Direction – направление смещения от базовой плоскости. Если значение данного свойства равно true, то смещение имеет прямое направление, иначе (false), то смещение в обратном направлении. Чаще всего прямое направление указывает вверх.
- Offset – расстояние смещения от базовой плоскости.

- GetPlane() – возвращает интерфейс ksEntity, описывающий базовую плоскость.
- SetPlane() – устанавливает новую базовую плоскость.

Ниже приводится фрагмент программы, демонстрирующий создание смещенной плоскости:

```
public static int pTop_Part = -1;
public static short o3d_planeXOY = 1;
public static short o3d_planeOffset = 14;

public static KompasObject kompas;
public static ksDocument3D doc_3d;
public static ksPart part;
public static ksEntity entity_plane, entity_offset_plane;
public static ksPlaneOffsetDefinition plane_offset_def;
public static void Main()
{
    // подключаемся к компасу
    kompas =
(KompasObject)Marshal.GetActiveObject("Kompas.Application.5");
    // получаем интерфейс документа-модели
    doc_3d = (ksDocument3D)kompas.Document3D();
    // создаем документ-модель
    doc_3d.Create(false, true);
    // получаем интерфейс компонента
    part = (ksPart)doc_3d.GetPart(pTop_Part);
    // получаем интерфейс объекта "плоскость XOY"
    entity_plane = (ksEntity)part.GetDefaultEntity(o3d_planeXOY);
    // получаем интерфейс объекта "смещенная плоскость"
    entity_offset_plane = (ksEntity)part.NewEntity(o3d_planeOffset);
    // получаем интерфейс параметров смещенной плоскости
    plane_offset_def =
(ksPlaneOffsetDefinition)entity_offset_plane.GetDefinition();
    plane_offset_def.direction = true; // прямое направление смещения
    plane_offset_def.offset = 50; // величина смещения
    // устанавливаем базовую плоскость
    plane_offset_def.SetPlane(entity_plane);
    // создаем смещенную плоскость
    entity_offset_plane.Create();
    // включаем отображение плоскостей
    doc_3d.hideAllPlanes = false;
    // делаем компас видимым
    kompas.Visible = true;
}
```

## Операция «приклеивание выдавливанием». Параметры операции (ksBossExtrusionDefinition)

Параметры операции «приклеивание выдавливанием» задаются интерфейсом ksBossExtrusionDefinition. Его возвращает метод GetDefinition() интерфейса ksEntity. Интерфейс ksBossExtrusionDefinition является полной копией интерфейса ksBaseExtrusionDefinition, который мы рассматривали ранее.

В качестве примера напишу программу, которая строит простой вал, код данной программы представлен ниже:

```
public static int pTop_Part = -1;
public static short o3d_sketch = 5;
public static short o3d_planeXOY = 1;
public static short o3d_planeOffset = 14;
public static short o3d_base_extrusion = 24;
public static short o3d_boss_extrusion = 25;
public static short etBlind = 0;
public static short vm_Shaded = 3;

public static KompasObject kompas;
public static ksDocument3D doc_3d;
public static ksPart part;
public static ksEntity entity_sketch1, entity_sketch2,
entity_planeXOY,
entity_extrusion, entity_plane_offset, entity_boss_extrusion;
public static ksSketchDefinition sketch_def;
public static ksDocument2D doc_2d;
public static ksBaseExtrusionDefinition base_ex_def;
public static ksPlaneOffsetDefinition plane_offset_def;
public static ksBossExtrusionDefinition boss_ex_def;

public static void Main()
{
    //Подключаемся к КОМПАСу
    kompas =
(KompasObject)Marshal.GetActiveObject("Kompas.Application.5");
    //Получаем интерфейс документа-модели
    doc_3d = (ksDocument3D)kompas.Document3D();
    //Создаем документ-модель
    doc_3d.Create(false, true);
    //Получаем интерфейс компонента
    part = (ksPart)doc_3d.GetPart(pTop_Part);
    //Получаем интерфейс объекта "плоскость XOY"
    entity_planeXOY = (ksEntity)part.GetDefaultEntity(o3d_planeXOY);

    //Получаем интерфейс объекта "Эскиз"
    entity_sketch1 = (ksEntity)part.NewEntity(o3d_sketch);
    //Получаем интерфейс параметров эскиза
    sketch_def = (ksSketchDefinition)entity_sketch1.GetDefinition();
    //Устанавливаем плоскость XOY базовой для эскиза
    sketch_def.SetPlane(entity_planeXOY);
    //Создаем эскиз
    entity_sketch1.Create();
    //Входим в режим редактирования эскиза
    doc_2d = (ksDocument2D)sketch_def.BeginEdit();
    //Строим окружность
    doc_2d.ksCircle(0, 0, 15, 1);
    //Выходим из режима редактирования эскиза
    sketch_def.EndEdit();

    //Операция "выдавливание"
```



```

//Получаем интерфейс объекта "операция выдавливание"
entity_extrusion = (ksEntity)part.NewEntity(o3d_base_extrusion);
//Получаем интерфейс параметров операции "выдавливание"
base_ex_def =
(ksBaseExtrusionDefinition)entity_extrusion.GetDefinition();
//Устанавливаем параметры операции выдавливания
base_ex_def.SetSideParam(true, etBlind, 20, 0, true);
//Устанавливаем эскиз операции выдавливания
base_ex_def.SetSketch(entity_sketch1);
//Создаем операцию выдавливания
entity_extrusion.Create();
//Получаем интерфейс объекта "смещенная плоскость"
entity_plane_offset = (ksEntity)part.NewEntity(o3d_planeOffset);
//Получаем интерфейс параметров смещенной плоскости
plane_offset_def =
(ksPlaneOffsetDefinition)entity_plane_offset.GetDefinition();
//Направление смещения – прямое
plane_offset_def.direction = true;
//Смещение
plane_offset_def.offset = 20;
//Устанавливаем базовую плоскость
plane_offset_def.SetPlane(entity_planeXOY);
//Создаем смещенную плоскость
entity_plane_offset.Create();
//Эскиз 2
//Получаем интерфейс объекта "Эскиз"
entity_sketch2 = (ksEntity)part.NewEntity(o3d_sketch);
//Получаем интерфейс параметров эскиза
sketch_def = (ksSketchDefinition)entity_sketch2.GetDefinition();
//Устанавливаем смещенную плоскость базовой для эскиза
sketch_def.SetPlane(entity_plane_offset);
//Создаем эскиз
entity_sketch2.Create();
//Входим в режим редактирования эскиза
doc_2d = (ksDocument2D)sketch_def.BeginEdit();
//Строим окружность
doc_2d.ksCircle(0, 0, 10, 1);
//Выходим из режима редактирования эскиза
sketch_def.EndEdit();

//Приклеивание выдавливанием
//Получаем интерфейс объекта "приклеивание выдавливанием"
entity_boss_extrusion =
(ksEntity)part.NewEntity(o3d_boss_extrusion);
//Получаем интерфейс параметров операции "приклеивание
выдавливанием"
boss_ex_def =
(ksBossExtrusionDefinition)entity_boss_extrusion.GetDefinition();
//Устанавливаем параметры операции "приклеивание выдавливанием"
boss_ex_def.SetSideParam(true, etBlind, 50, 0, true);
//Устанавливаем эскиз для операции "приклеивание выдавливанием"
boss_ex_def.SetSketch(entity_sketch2);
//Создаем операцию "приклеивание выдавливанием"
entity_boss_extrusion.Create();
//Делаем плоскости невидимыми

```



```

doc_3d.hideAllPlanes = true;
//Устанавливаем полутонное изображение модели
doc_3d.drawMode = vm_Shaded;
//Включаем отображение каркаса
doc_3d.shadedWireframe = true;
//Делаем КОМПАС видимым
kompas.Visible = true;
}

```

## Операция «вырезание выдавливанием». Параметры операции (ksCutExtrusionDefinition)

Параметры операции «вырезание выдавливанием» задаются с помощью интерфейса ksCuExtrusionDefinition. Его возвращает метод GetDefiniton() интерфейса ksEntity, описывающего данную операцию. Данный интерфейс практически полностью идентичен интерфейсу ksBaseExtrusionDefinition, который мы рассматривали ранее. По сравнению с последним к интерфейсу ksCutExtrusionDefinition добавилось всего одно свойство:

Cut – определяет характер операции выдавливания, если true, то вычитание элементов («вырезание»), иначе (false), то пересечение элементов.

В качестве примера ниже представлен фрагмент программы, которая будет строить простой стакан:

```

namespace CutExtrusionDef
{
    public class CutExtrusionDef
    {
        public static int pTop_Part = -1;
        public static short o3d_planeXOY = 1;
        public static short o3d_sketch = 5;
        public static short o3d_planeOffset = 14;
        public static short o3d_baseExtrusion = 24;
        public static short o3d_cutExtrusion = 26;
        public static short dtNormal = 0;
        public static short etBlind = 0;
        public static short vm_Shaded = 3;

        public static KompasObject kompas;
        public static ksDocument3D doc_3d;
        public static ksPart part;
        public static ksEntity entity_plane_offset, entity_planeXOY,
entity_sketch1, entity_sketch2, entity_ex, entity_cut_ex;
        public static ksSketchDefinition sketch_def;
        public static ksDocument2D doc_2d;
        public static ksBaseExtrusionDefinition base_extrusion_def;
        public static ksPlaneOffsetDefinition plane_offset_def;
        public static ksCutExtrusionDefinition cut_extrusion_def;
    }
}

```

```

public static void Main()
{
    //Подключаемся к КОМПАСу
    Kompas =
(KompasObject)Marshal.GetActiveObject("Kompas.Application.5");
    //Получаем интерфейс документа-модели
    doc_3d = (ksDocument3D)Kompas.Document3D();
    //Создаем документ-модель
    doc_3d.Create(false, true);
    //Получаем интерфейс компонента
    part = (ksPart)doc_3d.GetPart(pTop_Part);
    //Получаем интерфейс объекта "плоскость XOY"
    entity_planeXOY =
(ksEntity)part.GetDefaultEntity(o3d_planeXOY);

    //Эскиз 1
    //Получаем интерфейс объекта "Эскиз"
    entity_sketch1 = (ksEntity)part.NewEntity(o3d_sketch);
    //Получаем интерфейс параметров эскиза
    sketch_def =
(ksSketchDefinition)entity_sketch1.GetDefinition();
    //Устанавливаем плоскость XOY базовой для эскиза
    sketch_def.SetPlane(entity_planeXOY);
    //Создаем эскиз
    entity_sketch1.Create();
    //Входим в режим редактирования эскиза
    doc_2d = (ksDocument2D)sketch_def.BeginEdit();
    //Строим окружность
    doc_2d.KsCircle(0, 0, 15, 1);
    //Выходим из режима редактирования эскиза
    sketch_def.EndEdit();

    //Операция "выдавливание"
    //Получаем интерфейс объекта "операция выдавливание"
    entity_ex = (ksEntity)part.NewEntity(o3d_baseExtrusion);
    //Получаем интерфейс параметров операции "выдавливание"
    base_extrusion_def =
(ksBaseExtrusionDefinition)entity_ex.GetDefinition();
    //Устанавливаем параметры операции выдавливания
    base_extrusion_def.SetSideParam(true, etBlind, 20, 0, true);
    //Устанавливаем эскиз операции выдавливания
    base_extrusion_def.SetSketch(entity_sketch1);
    //Создаем операцию выдавливания
    entity_ex.Create();

    //Смещенная плоскость
    //Получаем интерфейс объекта "смещенная плоскость"
    entity_plane_offset =
(ksEntity)part.NewEntity(o3d_planeOffset);
    //Получаем интерфейс параметров смещенной плоскости
    plane_offset_def =
(ksPlaneOffsetDefinition)entity_plane_offset.GetDefinition();
    //Направление смещения – прямое
    plane_offset_def.direction = true;
    //Смещение

```

```

plane_offset_def.offset = 20;
//Устанавливаем базовую плоскость
plane_offset_def.SetPlane(entity_planeXOY);
//Создаем смещенную плоскость
entity_plane_offset.Create();

//Эскиз 2
//Получаем интерфейс объекта "Эскиз"
entity_sketch2 = (ksEntity)part.NewEntity(o3d_sketch);
//Получаем интерфейс параметров эскиза
sketch_def =
(ksSketchDefinition)entity_sketch2.GetDefinition();
//Устанавливаем смещенную плоскость базовой для эскиза
sketch_def.SetPlane(entity_plane_offset);
//Создаем эскиз
entity_sketch2.Create();
//Входим в режим редактирования эскиза
doc_2d = (ksDocument2D)sketch_def.BeginEdit();
//Строим окружность
doc_2d.ksCircle(0, 0, 10, 1);
//Выходим из режима редактирования эскиза
sketch_def.EndEdit();

//Вырезание выдавливанием
//Получаем интерфейс объекта "операция вырезание
выдавливанием"
entity_cut_ex = (ksEntity)part.NewEntity(o3d_cutExtrusion);
//Получаем интерфейс параметров операции
cut_extrusion_def =
(ksCutExtrusionDefinition)entity_cut_ex.GetDefinition();
//Вычитание элементов
cut_extrusion_def.cut = true;
//Прямое направление
cut_extrusion_def.directionType = dtNormal;
//Устанавливаем параметры выдавливания
cut_extrusion_def.SetSideParam(true, etBlind, 15, 0, false);
//Устанавливаем эскиз операции
cut_extrusion_def.SetSketch(entity_sketch2);
//Создаем операцию вырезания выдавливанием
entity_cut_ex.Create();
//Делаем плоскости невидимыми
doc_3d.hideAllPlanes = true;
//Устанавливаем полутонное изображение модели
doc_3d.drawMode = vm_Shaded;
//Включаем отображение каркаса
doc_3d.shadedWireframe = true;
//Делаем КОМПАС видимым
kompas.Visible = true;

}

}

```

## Операция «выдавливание вращением». Параметры операции выдавливание вращением (ksBaseRotatedDefinition)

Параметры операции «выдавливание вращением» описываются интерфейсом ksBaseRotatedDefinition. Этот интерфейс возвращает метод GetDefinition() интерфейса ksEntity, описывающего эту операцию. Основным свойством интерфейса ksBaseRotatedDefinition является свойство directionType – тип направления выдавливания. Данное свойство аналогично свойству directionType интерфейса ksBaseExtrusionDefinition.

Ниже рассмотрим основные методы интерфейса ksBaseRotatedDefinition:

- GetSketch() – возвращает интерфейс ksEntity, описывающий эскиз операции «выдавливание вращением».
- SetSideParam() – устанавливает параметры операции «выдавливание вращением».
- SetSketch() – устанавливает эскиз операции «выдавливание вращением».

Ниже приводится фрагмент программы, демонстрирующей построение операции «выдавливание вращением»:

```
namespace base_rotated_def
{
    public class base_rotated_def
    {
        public static int pTop_Part = -1;
        public static short o3d_planeXOY = 1;
        public static short o3d_sketch = 5;
        public static short o3d_baseRotated = 27;
        public static short dtNormal = 0;
        public static short vm_Shaded = 3;

        public static KompasObject kompas;
        public static ksDocument3D doc_3d;
        public static ksPart part;
        public static ksEntity entity_planeXOY, entity_sketch,
entity_rotated;
        public static ksSketchDefinition sketch_def;
        public static ksBaseRotatedDefinition baseRotated_def;
        public static ksDocument2D doc_2d;

        public static void Main()
        {
            //Подключаемся к КОМПАСу
            kompas =
(KompasObject)Marshal.GetActiveObject("Kompas.Application.5");
            //Получаем интерфейс документа-модели
            doc_3d = (ksDocument3D)kompas.Document3D();
            //Создаем документ-модель
            doc_3d.Create(false, true);
        }
    }
}
```

```

//Получаем интерфейс компонента
part = (ksPart)doc_3d.GetPart(pTop_Part);
//Получаем интерфейс объекта "плоскость XOY"
entity_planeXOY =
(ksEntity)part.GetDefaultEntity(o3d_planeXOY);

//Эскиз
//Получаем интерфейс объекта "Эскиз"
entity_sketch = (ksEntity)part.NewEntity(o3d_sketch);
//Получаем интерфейс параметров эскиза
sketch_def =
(ksSketchDefinition)entity_sketch.GetDefinition();
//Устанавливаем плоскость XOY базовой для эскиза
sketch_def.SetPlane(entity_planeXOY);
//Создаем эскиз
entity_sketch.Create();
//Входим в режим редактирования эскиза
doc_2d = (ksDocument2D)sketch_def.BeginEdit();
//Строим окружность
doc_2d.kCircle(-30, 0, 15, 1);
//Строим осевую линию
doc_2d.kLineSeg(0, -15, 0, 15, 3);
//Выходим из режима редактирования эскиза
sketch_def.EndEdit();

//Операция "выдавливание вращением"
//Получаем интерфейс объекта "операция выдавливание
вращением"
entity_rotated = (ksEntity)part.NewEntity(o3d_baseRotated);
//Получаем интерфейс параметров операции "выдавливание
вращением"
baseRotated_def =
(ksBaseRotatedDefinition)entity_rotated.GetDefinition();
//Направление выдавливания – прямое
baseRotated_def.directionType = dtNormal;
//Устанавливаем параметры операции
baseRotated_def.SetSideParam(true, 90);
//Устанавливаем эскиз операции
baseRotated_def.SetSketch(entity_sketch);
//Создаем операцию
entity_rotated.Create();
//Включаем полутонное отображение модели
doc_3d.drawMode = vm_Shaded;
//Включаем отображение каркаса
doc_3d.shadedWireframe = true;
//Делаем КОМПАС видимым
kompas.Visible = true;
}
}
}

```

## Наклоненная плоскость. Параметры наклоненной плоскости (ksPlaneAngleDefenition)

Наклоненная плоскость представляет собой плоскость, расположенную под заданным углом по отношению к базовой плоскости. Параметры наклоненной плоскости задаются интерфейсом ksPlaneAngleDefenition. Его возвращает метод GetDefenition() интерфейса ksEntity.

У интерфейса ksPlaneAngleDefenition всего одно свойство:

- Angle – угол между наклоненной и базовой плоскостями.

Ниже представлены основные методы интерфейса ksPlaneAngleDefenition:

- GetAxis() – возвращает интерфейс ksEntity, описывающий ось, по которой пересекаются наклоненная и базовая плоскости;
- GetPlane() – возвращает интерфейс ksEntity, описывающий ось, по которой пересекаются наклоненная и базовая;
- SetAxis() – устанавливает новую ось, по которой пересекаются наклоненная и базовая плоскости. В качестве единственного параметра принимает интерфейс ksEntity, описывающий устанавливаемую ось. В случае успеха метод возвращает значение true;
- SetPlane() – устанавливает новую ось, по которой пересекаются наклоненная и базовая плоскости. В качестве единственного параметра принимает интерфейс ksEntity, описывающий устанавливаемую плоскость.

Ниже приводится фрагмент программы, демонстрирующий построение наклоненной плоскости:

```
namespace PlanwAngleDefinition
{
    public class PlanwAngleDefinition
    {
        public static int pTop_Part = -1;
        public static short o3d_planeXOY = 1;
        public static short o3d_planeYOZ = 3;
        public static short o3d_axis2Planes = 9;
        public static short o3d_planeAngle = 15;

        public static KompasObject kompas;
        public static ksDocument3D doc_3d;
        public static ksPart part;
        public static ksEntity EntityPlaneXOY, EntityPlaneYOZ,
EntityPlaneAngle, EntityAxis;
        public static ksAxis2PlanesDefinition axis_2_plane_def;
        public static ksPlaneAngleDefinition plane_angle_def;

        public static void Main()
        {
```

```

        //Подключаемся к КОМПАСу
        kompas =
(KompasObject)Marshal.GetActiveObject("Kompas.Application.5");
        //Получаем интерфейс документа-модели
        doc_3d = (ksDocument3D)kompas.Document3D();
        //Создаем документ-модель
        doc_3d.Create(false, true);
        //Получаем интерфейс компонента
        part = (ksPart)doc_3d.GetPart(pTop_Part);
        //Получаем интерфейс объекта "плоскость XOY"
        EntityPlaneXOY =
(ksEntity)part.GetDefaultEntity(o3d_planeXOY);
        //Получаем интерфейс объекта "плоскость YOZ"
        EntityPlaneYOZ =
(ksEntity)part.GetDefaultEntity(o3d_planeYOZ);

        //Ось, как результат пересечения двух плоскостей
        //Получаем интерфейс объекта
        EntityAxis = (ksEntity)part.NewEntity(o3d_axis2Planes);
        //Получаем интерфейс параметров объекта
        axis_2_plane_def =
(ksAxis2PlanesDefinition)EntityAxis.GetDefinition();
        //Устанавливаем первую плоскость
        axis_2_plane_def.SetPlane(1, EntityPlaneXOY);
        //Устанавливаем вторую плоскость
        axis_2_plane_def.SetPlane(2, EntityPlaneYOZ);
        //Создаем ось
        EntityAxis.Create();

        //Наклоненная плоскость
        //Получаем интерфейс объекта "наклоненная плоскость"
        EntityPlaneAngle = (ksEntity)part.NewEntity(o3d_planeAngle);
        //Получаем интерфейс параметров объекта "наклоненная
плоскость"
        plane_angle_def =
(ksPlaneAngleDefinition)EntityPlaneAngle.GetDefinition();
        //Угол наклона наклоненной плоскости
        plane_angle_def.angle = -45;
        //Устанавливаем базовую плоскость
        plane_angle_def.SetPlane(EntityPlaneXOY);
        //Устанавливаем ось
        plane_angle_def.SetAxis(EntityAxis);
        //Создаем наклоненную плоскость
        EntityPlaneAngle.Create();
        //Делаем оси невидимыми
        doc_3d.hideAllAxis = true;
        //Делаем плоскости видимыми
        doc_3d.hideAllPlanes = false;
        //Делаем КОМПАС видимым
        kompas.Visible = true;
    }
}

```



## Операция «приклеивание вращением». Параметры операции «приклеивание вращением» (ksBossRotatedDefenition)

Параметры операции «приклеивание вращением» задаются интерфейсом ksBossRotatedDefenition. Его возвращает метод GetDefenition интерфейса ksEntity.

Интерфейс ksBossRotatedDefenition аналогичен интерфейсу ksBaseRotatedDefenition.

Ниже приводится ключевой фрагмент программы, демонстрирующей использование операции «приклеивание вращением»:

```
namespace base_rotated_def
{
    public class base_rotated_def
    {
        public static int pTop_Part = -1;
        public static short o3d_planeXOY = 1;
        public static short o3d_sketch = 5;
        public static short o3d_baseRotated = 27;
        public static short dtNormal = 0;
        public static short vm_Shaded = 3;

        public static KompasObject kompas;
        public static ksDocument3D doc_3d;
        public static ksPart part;
        public static ksEntity entity_planeXOY, entity_sketch,
entity_rotated;
        public static ksSketchDefinition sketch_def;
        public static ksBaseRotatedDefinition baseRotated_def;
        public static ksDocument2D doc_2d;

        public static void Main()
        {
            //Подключаемся к КОМПАСу
            kompas =
(KompasObject)Marshal.GetActiveObject("Kompas.Application.5");
            //Получаем интерфейс документа-модели
            doc_3d = (ksDocument3D)kompas.Document3D();
            //Создаем документ-модель
            doc_3d.Create(false, true);
            //Получаем интерфейс компонента
            part = (ksPart)doc_3d.GetPart(pTop_Part);
            //Получаем интерфейс объекта "плоскость XOY"
            entity_planeXOY =
(ksEntity)part.GetDefaultEntity(o3d_planeXOY);

            //Эскиз
            //Получаем интерфейс объекта "Эскиз"
            entity_sketch = (ksEntity)part.NewEntity(o3d_sketch);
            //Получаем интерфейс параметров эскиза
```



```

        sketch_def =
(ksSketchDefinition)entity_sketch.GetDefinition();
        //Устанавливаем плоскость XOY базовой для эскиза
        sketch_def.SetPlane(entity_planeXOY);
        //Создаем эскиз
        entity_sketch.Create();
        //Входим в режим редактирования эскиза
        doc_2d = (ksDocument2D)sketch_def.BeginEdit();
        //Строим окружность
        doc_2d.ksCircle(-30, 0, 15, 1);
        //Строим осевую линию
        doc_2d.ksLineSeg(0, -15, 0, 15, 3);
        //Выходим из режима редактирования эскиза
        sketch_def.EndEdit();

        //Операция "выдавливание вращением"
        //Получаем интерфейс объекта "операция выдавливание
вращением"
        entity_rotated = (ksEntity)part.NewEntity(o3d_baseRotated);
        //Получаем интерфейс параметров операции "выдавливание
вращением"
        baseRotated_def =
(ksBaseRotatedDefinition)entity_rotated.GetDefinition();
        //Направление выдавливания – прямое
        baseRotated_def.directionType = dtNormal;
        //Устанавливаем параметры операции
        baseRotated_def.SetSideParam(true, 90);
        //Устанавливаем эскиз операции
        baseRotated_def.SetSketch(entity_sketch);
        //Создаем операцию
        entity_rotated.Create();
        //Включаем полутонное отображение модели
        doc_3d.drawMode = vm_Shaded;
        //Включаем отображение каркаса
        doc_3d.shadedWireframe = true;
        //Делаем КОМПАС видимым
        kompas.Visible = true;
    }
}

```

## Операция «вырезание вращением». Параметры операции «вырезание вращением» (ksCutRotatedDefenition)

Параметры операции «вырезание вращением» задаются с помощью интерфейса ksCutRotatedDefenition. Его возвращает метод GetDefenition() интерфейса ksEntity.

Интерфейс ksCutRotatedDefenition очень похож на интерфейс ksBaseRotatedDefenition, но имеет одно дополнительное свойство:

- Cut – признак выполняемой операции. Если true, то вычитание объектов, иначе (false), то пересечение объектов.

Ниже представлен фрагмент программы, демонстрирующей использование данного метода:

```

public class cut_rotated_def
{
    public static int pTop_Part = -1;
    public static short o3d_planeXOY = 1;
    public static short o3d_planeXOZ = 2;
    public static short o3d_sketch = 5;
    public static short o3d_baseExtrusion = 24;
    public static short o3d_cutRotated = 29;
    public static short dtNormal = 0;
    public static short etBlind = 0;
    public static short vm_Shaded = 3;

    public static KompasObject kompas;
    public static ksDocument3D doc_3d;
    public static ksPart part;
    public static ksEntity entity_sketch1, entity_sketch2,
entity_planeXOY, entity_planeXOZ, entity_extrusion,
    entity_cut_rotated;
    public static ksSketchDefinition sketch_def;
    public static ksDocument2D doc_2d;
    public static ksBaseExtrusionDefinition base_extrusion_def;
    public static ksCutRotatedDefinition cut_rotated_def;

    public static void Main()
    {
        //Подключаемся к КОМПАСу
        kompas =
(KompasObject)Marshal.GetActiveObject("Kompas.Application.5");
        //Получаем интерфейс документа-модели
        doc_3d = (ksDocument3D)kompas.Document3D();
        //Создаем документ-модель
        doc_3d.Create(false, true);
        //Получаем интерфейс компонента
        part = (ksPart)doc_3d.GetPart(pTop_Part);
        //Получаем интерфейс объекта "плоскость XOY"
        entity_planeXOY = (ksEntity)part.GetDefaultEntity(o3d_planeXOY);
        //Получаем интерфейс объекта "плоскость XOZ"
        entity_planeXOZ = (ksEntity)part.GetDefaultEntity(o3d_planeXOZ);

        //Эскиз 1
        //Получаем интерфейс объекта "Эскиз"
        entity_sketch1 = (ksEntity)part.NewEntity(o3d_sketch);
        //Получаем интерфейс параметров эскиза
        sketch_def = (ksSketchDefinition)entity_sketch1.GetDefinition();
        //Устанавливаем плоскость XOY базовой для эскиза
        sketch_def.SetPlane(entity_planeXOY);
        //Создаем эскиз
        entity_sketch1.Create();
        //Входим в режим редактирования эскиза
        doc_2d = (ksDocument2D)sketch_def.BeginEdit();
        //Строим окружность
        doc_2d.ksCircle(0, 0, 15, 1);
        //Выходим из режима редактирования эскиза
        sketch_def.EndEdit();
    }
}

```

```

//Операция "выдавливание"
//Получаем интерфейс объекта "операция выдавливание"
entity_extrusion = (ksEntity)part.NewEntity(o3d_baseExtrusion);
//Получаем интерфейс параметров операции "выдавливание"
base_extrusion_def =
(ksBaseExtrusionDefinition)entity_extrusion.GetDefinition();
//Устанавливаем параметры операции выдавливания
base_extrusion_def.SetSideParam(true, etBlind, 70, 0, true);
//Устанавливаем эскиз операции выдавливания
base_extrusion_def.SetSketch(entity_sketch1);
//Создаем операцию выдавливания
entity_extrusion.Create();

//Эскиз 2
//Получаем интерфейс объекта "Эскиз"
entity_sketch2 = (ksEntity)part.NewEntity(o3d_sketch);
//Получаем интерфейс параметров объекта "Эскиз"
sketch_def = (ksSketchDefinition)entity_sketch2.GetDefinition();
//Устанавливаем плоскость XOZ базовой для эскиза
sketch_def.SetPlane(entity_planeXOZ);
//Создаем эскиз
entity_sketch2.Create();
//Входим в режим редактирования эскиза
doc_2d = (ksDocument2D)sketch_def.BeginEdit();
//Строим прямоугольник, которым будем "вырезать"
doc_2d.ksLineSeg(10, -15, 10, -60, 1);
doc_2d.ksLineSeg(10, -15, 30, -15, 1);
doc_2d.ksLineSeg(30, -15, 30, -60, 1);
doc_2d.ksLineSeg(30, -60, 10, -60, 1);
//Строим ось
doc_2d.ksLineSeg(0, -60, 0, -15, 3);
//Выходим из режима редактирования эскиза
sketch_def.EndEdit();

//Операция "вырезание вращением"
//Получаем интерфейс объекта
entity_cut_rotated = (ksEntity)part.NewEntity(o3d_cutRotated);
//Получаем интерфейс параметров объекта
cut_rotated_def =
(ksCutRotatedDefinition)entity_cut_rotated.GetDefinition();
//Направление вращения – прямое
cut_rotated_def.directionType = dtNormal;
//Вычитание объектов
cut_rotated_def.cut = true;
//Устанавливаем параметры вращения
cut_rotated_def.SetSideParam(true, 360);
//Устанавливаем эскиз
cut_rotated_def.SetSketch(entity_sketch2);
//Создаем операцию
entity_cut_rotated.Create();
//Устанавливаем полутонное изображение модели
doc_3d.drawMode = vm_Shaded;
//Включаем отображение каркаса
doc_3d.shadedWireframe = true;
//Делаем КОМПАС видимым

```

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

## **Операция «выдавливания по сечениям». Массив объектов (ksEntityCollection)**

Интерфейс ksEntityCollection служит для описания массива объектов. Каждый элемент в этом массиве представляет собой интерфейс ksEntity, описывающий тот или иной объект.

Интерфейс ksEntityCollection не имеет свойств. Ниже представлены основные метода данного интерфейса:

- Add() – добавляет объект в конец массива;
- AddAt() – добавить в массив объект с заданным индексом. Первым параметром метода передается интерфейс ksEntity добавляемого объекта. Вторым параметром метода передается индекс вставляемого объекта в массиве. Нумерация объектов ведется с нуля;
- AddBefore() – вставить объект в массив перед другим объектом. Первым параметром метода передается интерфейс ksEntity добавляемого объекта. Вторым параметром метода передается интерфейс ksEntity объекта, перед которым нужно вставить добавляемый объект;
- Clear() – очищает массив. Сами объекты из модели не удаляются.
- DetachByBody() – удалить объект из массива. В качестве единственного параметра методу передается интерфейс объекта ksEntity, который нужно удалить из массива. При этом сам объект из модели не удаляется;
- DetachByIndex() – удалить объект из массива. В качестве единственного параметра методу передается значение индекса удаляемого объекта. Элементы массива нумеруются с нуля. Сам объект из модели не удаляется;
- FindIt ()– найти объект в массиве. В качестве единственного параметра данному методу передается интерфейс объекта ksEntity, который нужно найти в массиве.
- First() – возвращает интерфейс ksEntity первого элемента в массиве;
- GetByIndex() – возвращает интерфейс ksEntity объекта с заданным индексом. Номер запрашиваемого объекта в массиве передается в качестве единственного параметра метода. Элементы массива нумеруются с нуля;
- GetCount() – возвращает количество элементов (объектов) в массиве;
- Last() – возвращает интерфейс ksEntity последнего объекта в массиве;
- Next() – возвращает интерфейс ksEntity следующего объекта в массиве;
- Prev() – возвращает интерфейс ksEntity предыдущего объекта в массиве;
- Refresh() – обновить массив. В результате вызов этого метода происходит синхронизация объектов в массиве и объектов 3D-модели, которым соответствуют объекты в массиве;

- **SelectByPoint()** – удалить из массива все объекты, которые не содержат в себе или не проходят через заданную точку. Координаты точки (x, y, z) передаются в качестве параметров метода.

- **SetByIndex()** – заменить заданный объект массива новым объектом. Первым параметром метода передается интерфейс **ksEntity** нового (добавляемого) объекта. Вторым параметром метода передается значение индекса объекта, который нужно заменить новым объектом.

## **Параметры операции «выдавливание по сечениям» (ksBaseLoftDefenition)**

Параметры операции «выдавливание по сечениям» задается с помощью интерфейса **ksBaseLoftDefenition**. Его возвращает метод **GetDefenition()** интерфейса **ksEntity**, описывающего данную операцию. **ksBaseLoftDefenition** не имеет свойств. Ниже рассмотрены методы этого интерфейса:

- **SetLoftParam(bool closed)** – установить параметры операции **closed** равно **true**, то строится замкнутая траектория, иначе (**false**) строится разомкнутая траектория;

- **autoPath** – признак автоматического формирования. Если **true**, то траектория формируется автоматически, иначе (**false**), то для формирования траектории используются специальные точки, установленные на эскизах;

- **Sketchs()** – возвращает интерфейс массива эскизов **ksEntityCollection**. Каждый эскиз в этом массиве описывается интерфейсом **ksEntity**.

Ниже представлен фрагмент программы, демонстрирующей использование этой операции:

```
public class base_loft_defenitionClass1
{

    public static int pTop_Part = -1;
    public static short o3d_planeX0Y = 1;
    public static short o3d_sketch = 5;
    public static short o3d_planeOffset = 14;
    public static short o3d_baseLoft = 30;
    public static short dtNormal = 0;
    public static short etBlind = 0;
    public static short vm_Shaded = 3;

    public static KompasObject kompas;
    public static ksDocument3D doc_3d;
    public static ksPart part;
    public static ksEntity entity_sketch1, entity_sketch2,
entity_sketch3, entity_planeX0Y, entity_plane_offset1,
    entity_plane_offset2, entity_base_loft;
    public static ksEntityCollection entity_collection;
    public static ksSketchDefinition sketch_def;
    public static ksDocument2D doc_2d;
    public static ksPlaneOffsetDefinition plane_offset_def;
```

```

public static ksBaseLoftDefinition base_loft_def;

public static void Main()
{
    //Подключаемся к КОМПАСу
    Kompas =
(KompasObject)Marshal.GetActiveObject("Kompas.Application.5");
    //Получаем интерфейс документа-модели
    doc_3d = (ksDocument3D)Kompas.Document3D();
    //Создаем документ-модель
    doc_3d.Create(false, true);
    //Получаем интерфейс компонента
    part = (ksPart)doc_3d.GetPart(pTop_Part);
    //Получаем интерфейс объекта "плоскость XOY"
    entity_planeXOY = (ksEntity)part.GetDefaultEntity(o3d_planeXOY);

    //Эскиз 1
    //Получаем интерфейс объекта "Эскиз"
    entity_sketch1 = (ksEntity)part.NewEntity(o3d_sketch);
    //Получаем интерфейс параметров эскиза
    sketch_def = (ksSketchDefinition)entity_sketch1.GetDefinition();
    //Устанавливаем плоскость XOY базовой для эскиза
    sketch_def.SetPlane(entity_planeXOY);
    //Создаем эскиз
    entity_sketch1.Create();
    //Входим в режим редактирования эскиза
    doc_2d = (ksDocument2D)sketch_def.BeginEdit();
    //Строим окружность
    doc_2d.KsCircle(0, 0, 30, 1);
    //Выходим из режима редактирования эскиза
    sketch_def.EndEdit();

    //Смещенная плоскость 1
    //Получаем интерфейс объекта "смещенная плоскость"
    entity_plane_offset1 = (ksEntity)part.NewEntity(o3d_planeOffset);
    //Получаем интерфейс параметров смещенной плоскости
    plane_offset_def =
(ksPlaneOffsetDefinition)entity_plane_offset1.GetDefinition();
    //Направление смещения – прямое
    plane_offset_def.direction = true;
    //Смещение
    plane_offset_def.offset = 20;
    //Устанавливаем базовую плоскость
    plane_offset_def.SetPlane(entity_planeXOY);
    //Создаем смещенную плоскость
    entity_plane_offset1.Create();

    //Эскиз 2
    //Получаем интерфейс объекта "Эскиз"
    entity_sketch2 = (ksEntity)part.NewEntity(o3d_sketch);
    //Получаем интерфейс параметров эскиза
    sketch_def = (ksSketchDefinition)entity_sketch2.GetDefinition();
    //Устанавливаем смещенную плоскость базовой для эскиза
    sketch_def.SetPlane(entity_plane_offset1);
    //Создаем эскиз

```



```

entity_sketch2.Create();
//Входим в режим редактирования эскиза
doc_2d = (ksDocument2D)sketch_def.BeginEdit();
//Строим окружность
doc_2d.ksCircle(0, 0, 20, 1);
//Выходим из режима редактирования эскиза
sketch_def.EndEdit();

//Смещенная плоскость 2
//Получаем интерфейс объекта "смещенная плоскость"
entity_plane_offset2 = (ksEntity)part.NewEntity(o3d_planeOffset);
//Получаем интерфейс параметров объекта "смещенная плоскость"
plane_offset_def =
(ksPlaneOffsetDefinition)entity_plane_offset2.GetDefinition();
//Направление смещения – прямое
plane_offset_def.direction = true;
//Смещение
plane_offset_def.offset = 40;
//Устанавливаем базовую плоскость
plane_offset_def.SetPlane(entity_planeXOY);
//Создаем смещенную плоскость
entity_plane_offset2.Create();

//Эскиз 3
//Получаем интерфейс объекта "Эскиз"
entity_sketch3 = (ksEntity)part.NewEntity(o3d_sketch);
//Получаем интерфейс параметров объекта "Эскиз"
sketch_def = (ksSketchDefinition)entity_sketch3.GetDefinition();
//Устанавливаем смещенную плоскость базовой для эскиза
sketch_def.SetPlane(entity_plane_offset2);
//Создаем эскиз
entity_sketch3.Create();
//Входим в режим редактирования эскиза
doc_2d = (ksDocument2D)sketch_def.BeginEdit();
//Строим окружность
doc_2d.ksCircle(0, 0, 25, 1);
//Выходим из режима редактирования эскиза
sketch_def.EndEdit();

//Операция "выдавливание по сечениям"
//Получаем интерфейс объекта
entity_base_loft = (ksEntity)part.NewEntity(o3d_baseLoft);
//Получаем интерфейс параметров объекта
base_loft_def =
(ksBaseLoftDefinition)entity_base_loft.GetDefinition();
//Устанавливаем параметры операции
base_loft_def.SetLoftParam(false, true, true);
//Получаем интерфейс массива объектов (эскизов)
entity_collection = (ksEntityCollection)base_loft_def.Sketches();
//Очищаем массив объектов
entity_collection.Clear();
//Добавляем в массив созданные ранее эскизы
entity_collection.Add(entity_sketch1);
entity_collection.Add(entity_sketch2);
entity_collection.Add(entity_sketch3);

```

```

//Создаем операцию
entity_base_loft.Create();
//Делаем плоскости невидимыми
doc_3d.hideAllPlanes = true;
//Устанавливаем полутонное изображение модели
doc_3d.drawMode = vm_Shaded;
//Включаем отображение каркаса
doc_3d.shadedWireframe = true;
//Делаем КОМПАС видимым
kompas.Visible = true;
}
}

```

## Операция «приклеивание по сечениям». Параметры операции «приклеивание по сечениям» (ksBossLoftDefenition)

Параметры операции «приклеивание по сечениям» описываются интерфейсом ksBossLoftDefenition. Его возвращает метод GetDefenition() интерфейса ksEntity, описывающую данную операцию.

Интерфейс ksBossLoftDefinition полностью аналогичен интерфейсу ksBaseLoftDefenition.

Ниже представлен фрагмент программы, демонстрирующей использование данного метода:

```

namespace boss_loft_definition
{
    public class boss_loft_definition
    {
        public static int pTop_Part = -1;
        public static short o3d_planeXOY = 1;
        public static short o3d_sketch = 5;
        public static short o3d_planeOffset = 14;
        public static short o3d_baseExtrusion = 24;
        public static short o3d_bossLoft = 31;
        public static short dtNormal = 0;
        public static short etBlind = 0;
        public static short vm_Shaded = 3;

        public static KompasObject kompas;
        public static ksDocument3D doc_3d;
        public static ksPart part;
        public static ksEntity entity_sketch1, entity_sketch2,
entity_sketch3, entity_sketch4, entity_planeXOY,
        entity_plane_offset1, entity_plane_offset2,
entity_plane_offset3, entity_base_extrusion,
        entity_boss_loft;
        public static ksEntityCollection entity_collection;
        public static ksSketchDefinition sketch_def;
        public static ksDocument2D doc_2d;
        public static ksBaseExtrusionDefinition base_extrusion_def;
        public static ksPlaneOffsetDefinition plane_offset_def;
        public static ksBossLoftDefinition boss_loft_def;
    }
}

```



```

public static void Main()
{
    //Подключаемся к КОМПАСу
    Kompas =
(KompasObject)Marshal.GetActiveObject("Kompas.Application.5");
    //Получаем интерфейс документа-модели
    doc_3d = (ksDocument3D)Kompas.Document3D();
    //Создаем документ-модель
    doc_3d.Create(false, true);
    //Получаем интерфейс компонента
    part = (ksPart)doc_3d.GetPart(pTop_Part);
    //Получаем интерфейс объекта "плоскость XOY"
    entity_planeXOY =
(ksEntity)part.GetDefaultEntity(o3d_planeXOY);

    //Эскиз 1
    //Получаем интерфейс объекта "Эскиз"
    entity_sketch1 = (ksEntity)part.NewEntity(o3d_sketch);
    //Получаем интерфейс параметров эскиза
    sketch_def =
(ksSketchDefinition)entity_sketch1.GetDefinition();
    //Устанавливаем плоскость XOY базовой для эскиза
    sketch_def.SetPlane(entity_planeXOY);
    //Создаем эскиз
    entity_sketch1.Create();
    //Входим в режим редактирования эскиза
    doc_2d = (ksDocument2D)sketch_def.BeginEdit();
    //Строим окружность
    doc_2d.KsCircle(0, 0, 30, 1);
    //Выходим из режима редактирования эскиза
    sketch_def.EndEdit();

    //Операция выдавливания
    //Получаем интерфейс операции
    entity_base_extrusion =
(ksEntity)part.NewEntity(o3d_baseExtrusion);
    //Получаем интерфейс параметров операции
    base_extrusion_def =
(ksBaseExtrusionDefinition)entity_base_extrusion.GetDefinition();
    //Направление прямое
    base_extrusion_def.directionType = dtNormal;
    //Устанавливаем параметры операции
    base_extrusion_def.SetSideParam(true, etBlind, 20, 0, true);
    //Устанавливаем эскиз операции
    base_extrusion_def.SetSketch(entity_sketch1);
    //Создаем операцию
    entity_base_extrusion.Create();

    //Смещенная плоскость 1
    //Получаем интерфейс объекта "смещенная плоскость"
    entity_plane_offset1 =
(ksEntity)part.NewEntity(o3d_planeOffset);
    //Получаем интерфейс параметров смещенной плоскости

```

```

        plane_offset_def =
(ksPlaneOffsetDefinition)entity_plane_offset1.GetDefinition();
        //Направление смещения – прямое
        plane_offset_def.direction = true;
        //Смещение
        plane_offset_def.offset = 20;
        //Устанавливаем базовую плоскость
        plane_offset_def.SetPlane(entity_planeXOY);
        //Создаем смещенную плоскость
        entity_plane_offset1.Create();

        //Эскиз 2
        //Получаем интерфейс объекта "Эскиз"
        entity_sketch2 = (ksEntity)part.NewEntity(o3d_sketch);
        //Получаем интерфейс параметров эскиза
        sketch_def =
(ksSketchDefinition)entity_sketch2.GetDefinition();
        //Устанавливаем смещенную плоскость базовой для эскиза
        sketch_def.SetPlane(entity_plane_offset1);
        //Создаем эскиз
        entity_sketch2.Create();
        //Входим в режим редактирования эскиза
        doc_2d = (ksDocument2D)sketch_def.BeginEdit();
        //Строим окружность
        doc_2d.ksCircle(0, 0, 30, 1);
        //Выходим из режима редактирования эскиза
        sketch_def.EndEdit();
        ////////////////////////////////////////////
        //Смещенная плоскость 2
        //Получаем интерфейс объекта "смещенная плоскость"
        entity_plane_offset2 =
(ksEntity)part.NewEntity(o3d_planeOffset);
        //Получаем интерфейс параметров объекта "смещенная плоскость"
        plane_offset_def =
(ksPlaneOffsetDefinition)entity_plane_offset2.GetDefinition();
        //Направление смещения – прямое
        plane_offset_def.direction = true;
        //Смещение
        plane_offset_def.offset = 40;
        //Устанавливаем базовую плоскость
        plane_offset_def.SetPlane(entity_planeXOY);
        //Создаем смещенную плоскость
        entity_plane_offset2.Create();

        //эскиз 3
        //Получаем интерфейс объекта "Эскиз"
        entity_sketch3 = (ksEntity)part.NewEntity(o3d_sketch);
        //Получаем интерфейс параметров объекта "Эскиз"
        sketch_def =
(ksSketchDefinition)entity_sketch3.GetDefinition();
        //Устанавливаем смещенную плоскость базовой для эскиза
        sketch_def.SetPlane(entity_plane_offset2);
        //Создаем эскиз
        entity_sketch3.Create();
        //Входим в режим редактирования эскиза

```

```

doc_2d = (ksDocument2D)sketch_def.BeginEdit();
//Строим окружность
doc_2d.ksCircle(0, 0, 20, 1);
//Выходим из режима релактирования эскиза
sketch_def.EndEdit();

//Смещенная плоскость 3
//Получаем интерфейс объекта "смещенная плоскость"
entity_plane_offset3 =
(ksEntity)part.NewEntity(o3d_planeOffset);
//Получаем интерфейс параметров объекта "смещенная плоскость"
plane_offset_def =
(ksPlaneOffsetDefinition)entity_plane_offset3.GetDefinition();
//Направление смещения – прямое
plane_offset_def.direction = true;
//Смещение
plane_offset_def.offset = 60;
//Устанавливаем базовую плоскость
plane_offset_def.SetPlane(entity_planeXOY);
//Создаем смещенную плоскость
entity_plane_offset3.Create();

//эскиз 4
//Получаем интерфейс объекта "Эскиз"
entity_sketch4 = (ksEntity)part.NewEntity(o3d_sketch);
//Получаем интерфейс параметров объекта "Эскиз"
sketch_def =
(ksSketchDefinition)entity_sketch4.GetDefinition();
//Устанавливаем смещенную плоскость базовой для эскиза
sketch_def.SetPlane(entity_plane_offset3);
//Создаем эскиз
entity_sketch4.Create();
//Входим в режим редактирования эскиза
doc_2d = (ksDocument2D)sketch_def.BeginEdit();
//Строим окружность
doc_2d.ksCircle(0, 0, 25, 1);
//Выходим из режима релактирования эскиза
sketch_def.EndEdit();

//Операция "приклеивание по сечениям"
//Получаем интерфейс операции
entity_boss_loft = (ksEntity)part.NewEntity(o3d_bossLoft);
//Получаем интерфейс параметров операции
boss_loft_def =
(ksBossLoftDefinition)entity_boss_loft.GetDefinition();
//Устанавливаем параметры операции
boss_loft_def.SetLoftParam(false, true, true);
//Получаем интерфейс массива объектов (эскизов)
entity_collection =
(ksEntityCollection)boss_loft_def.Sketches();
//Очищаем массив объектов
entity_collection.Clear();
//Добавляем в массив созданные ранее эскизы
entity_collection.Add(entity_sketch2);
entity_collection.Add(entity_sketch3);

```

```

entity_collection.Add(entity_sketch4);
//Создаем операцию
entity_boss_loft.Create();
//Делаем плоскости невидимыми
doc_3d.hideAllPlanes = true;
//Устанавливаем полутонное изображение модели
doc_3d.drawMode = vm_Shaded;
//Включаем отображение каркаса
doc_3d.shadedWireframe = true;
//Делаем КОМПАС видимым
kompas.Visible = true;
    }
}

```

## Операция «вырезание по сечениям». Параметры операции «вырезание по сечениям» (ksCutLoftDefenition)

Параметры операции «вырезание по сечениям» задаются с помощью интерфейса ksCutLoftDefenition. Его возвращает метод GetDefinition() интерфейса ksEntity, описывающего эту операцию.

Интерфейс ksCutLoftDefenition очень похож на ksBaseLoftDefenition, но к нему добавилось ещё одно свойство:

Cut – определяет характер операции. Если true, то имеет место вычитание объектов (классическое вырезание). Иначе (false), то имеет место пересечение объектов. Ниже приведен фрагмент программы, демонстрирующей использование этой операции:

```

namespace cut_loft_definition
{
    public class cut_loft_definition
    {
        public static int pTop_Part = -1;
        public static short o3d_planeXOY = 1;
        public static short o3d_sketch = 5;
        public static short o3d_planeOffset = 14;
        public static short o3d_baseExtrusion = 24;
        public static short o3d_cutLoft = 32;
        public static short dtNormal = 0;
        public static short etBlind = 0;
        public static short vm_Shaded = 3;

        public static KompasObject kompas;
        public static ksDocument3D doc_3d;
        public static ksPart part;
        public static ksEntity entity_sketch_1, entity_sketch_2,
entity_sketch_3, entity_sketch_4, entity_plane_XOY,
        entity_plane_offset_1, entity_plane_offset_2,
entity_base_extrusion, entity_cut_loft;
        public static ksEntityCollection entity_collection;
        public static ksSketchDefinition sketch_def;
        public static ksDocument2D doc_2d;
        public static ksBaseExtrusionDefinition base_extrusion_def;
    }
}

```

```

public static ksPlaneOffsetDefinition plane_offset_def;
public static ksCutLoftDefinition cut_loft_def;

public static void Main()
{
    //Подключаемся к КОМПАСу
    Kompas =
(KompasObject)Marshal.GetActiveObject("Kompas.Application.5");
    //Получаем интерфейс документа-модели
    doc_3d = (ksDocument3D)Kompas.Document3D();
    //Создаем документ-модель
    doc_3d.Create(false, true);
    //Получаем интерфейс компонента
    part = (ksPart)doc_3d.GetPart(pTop_Part);
    //Получаем интерфейс объекта "плоскость XOY"
    entity_plane_XOY =
(ksEntity)part.GetDefaultEntity(o3d_planeXOY);

    //Эскиз 1
    //Получаем интерфейс объекта "Эскиз"
    entity_sketch_1 = (ksEntity)part.NewEntity(o3d_sketch);
    //Получаем интерфейс параметров эскиза
    sketch_def =
(ksSketchDefinition)entity_sketch_1.GetDefinition();
    //Устанавливаем плоскость XOY базовой для эскиза
    sketch_def.SetPlane(entity_plane_XOY);
    //Создаем эскиз
    entity_sketch_1.Create();
    //Входим в режим редактирования эскиза
    doc_2d = (ksDocument2D)sketch_def.BeginEdit();
    //Строим окружность
    doc_2d.KsCircle(0, 0, 30, 1);
    //Выходим из режима редактирования эскиза
    sketch_def.EndEdit();

    //Операция выдавливания
    //Получаем интерфейс операции
    entity_base_extrusion =
(ksEntity)part.NewEntity(o3d_baseExtrusion);
    //Получаем интерфейс параметров операции
    base_extrusion_def =
(ksBaseExtrusionDefinition)entity_base_extrusion.GetDefinition();
    //Направление прямое
    base_extrusion_def.directionType = dtNormal;
    //Устанавливаем параметры операции
    base_extrusion_def.SetSideParam(true, etBlind, 40, 0, true);
    //Устанавливаем эскиз операции
    base_extrusion_def.SetSketch(entity_sketch_1);
    //Создаем операцию
    entity_base_extrusion.Create();

    //Эскиз 2
    //Получаем интерфейс объекта "Эскиз"
    entity_sketch_2 = (ksEntity)part.NewEntity(o3d_sketch);
    //Получаем интерфейс параметров эскиза

```

```

        sketch_def =
(ksSketchDefinition)entity_sketch_2.GetDefinition();
        //Устанавливаем смещенную плоскость базовой для эскиза
        sketch_def.SetPlane(entity_plane_XOY);
        //Создаем эскиз
        entity_sketch_2.Create();
        //Входим в режим редактирования эскиза
        doc_2d = (ksDocument2D)sketch_def.BeginEdit();
        //Строим окружность
        doc_2d.koCircle(0, 0, 30, 1);
        //Выходим из режима редактирования эскиза
        sketch_def.EndEdit();

        //Смещенная плоскость 1
        //Получаем интерфейс объекта "смещенная плоскость"
        entity_plane_offset_1 =
(ksEntity)part.NewEntity(o3d_planeOffset);
        //Получаем интерфейс параметров смещенной плоскости
        plane_offset_def =
(ksPlaneOffsetDefinition)entity_plane_offset_1.GetDefinition();
        //Направление смещения – прямое
        plane_offset_def.direction = true;
        //Смещение
        plane_offset_def.offset = 20;
        //Устанавливаем базовую плоскость
        plane_offset_def.SetPlane(entity_plane_XOY);
        //Создаем смещенную плоскость
        entity_plane_offset_1.Create();

        //эскиз 3
        //Получаем интерфейс объекта "Эскиз"
        entity_sketch_3 = (ksEntity)part.NewEntity(o3d_sketch);
        //Получаем интерфейс параметров объекта "Эскиз"
        sketch_def =
(ksSketchDefinition)entity_sketch_3.GetDefinition();
        //Устанавливаем смещенную плоскость базовой для эскиза
        sketch_def.SetPlane(entity_plane_offset_1);
        //Создаем эскиз
        entity_sketch_3.Create();
        //Входим в режим редактирования эскиза
        doc_2d = (ksDocument2D)sketch_def.BeginEdit();
        //Строим окружность
        doc_2d.koCircle(0, 0, 10, 1);
        //Выходим из режима редактирования эскиза
        sketch_def.EndEdit();

        //Смещенная плоскость 2
        //Получаем интерфейс объекта "смещенная плоскость"
        entity_plane_offset_2 =
(ksEntity)part.NewEntity(o3d_planeOffset);
        //Получаем интерфейс параметров объекта "смещенная плоскость"
        plane_offset_def =
(ksPlaneOffsetDefinition)entity_plane_offset_2.GetDefinition();
        //Направление смещения – прямое
        plane_offset_def.direction = true;

```

```

//Смещение
plane_offset_def.offset = 40;
//Устанавливаем базовую плоскость
plane_offset_def.SetPlane(entity_plane_XOY);
//Создаем смещенную плоскость
entity_plane_offset_2.Create();

//эскиз 4
//Получаем интерфейс объекта "Эскиз"
entity_sketch_4 = (ksEntity)part.NewEntity(o3d_sketch);
//Получаем интерфейс параметров объекта "Эскиз"
sketch_def =
(ksSketchDefinition)entity_sketch_4.GetDefinition();
//Устанавливаем смещенную плоскость базовой для эскиза
sketch_def.SetPlane(entity_plane_offset_2);
//Создаем эскиз
entity_sketch_4.Create();
//Входим в режим редактирования эскиза
doc_2d = (ksDocument2D)sketch_def.BeginEdit();
//Строим окружность
doc_2d.ksCircle(0, 0, 25, 1);
//Выходим из режима редактирования эскиза
sketch_def.EndEdit();

//Операция "вырезание по сечениям"
//Получаем интерфейс операции
entity_cut_loft = (ksEntity)part.NewEntity(o3d_cutLoft);
//Получаем интерфейс параметров операции
cut_loft_def =
(ksCutLoftDefinition)entity_cut_loft.GetDefinition();
//Вычитание объектов
cut_loft_def.cut = true;
//Устанавливаем параметры операции
cut_loft_def.SetLoftParam(false, true, true);
//Получаем интерфейс массива объектов (эскизов)
entity_collection =
(ksEntityCollection)cut_loft_def.Sketches();
//Очищаем массив объектов
entity_collection.Clear();
//Добавляем в массив созданные ранее эскизы
entity_collection.Add(entity_sketch_2);
entity_collection.Add(entity_sketch_3);
entity_collection.Add(entity_sketch_4);
//Создаем операцию
entity_cut_loft.Create();
//Делаем плоскости невидимыми
doc_3d.hideAllPlanes = true;
//Устанавливаем полутонное изображение модели
doc_3d.drawMode = vm_Shaded;
//Включаем отображение каркаса
doc_3d.shadedWireframe = true;
//Делаем КОМПАС видимым
kompas.Visible = true;
}
}

```



## Кинематическая операция выдавливания. Параметры кинематической операции выдавливания (ksBaseEvolutionDefinition)

Параметры кинематической операции выдавливания задаются с помощью интерфейса ksBaseEvolutionDefinition. Его возвращает метод GetDefenition() интеерфейса ksEntity.

У интерфейса ksBaseEvolutionDefinition всего одно свойство.

- sketchShiftType – задает характер движения сечения по траектории.

Возможные значения данного свойства представлены в таблице ниже.

Значение	Описание
0	Сечение переносится параллельно самому себе
1	Сечение при переносе сохраняет свой исходный угол с траекторией.
2	Сечение выставляется ортогонально (перпендикулярно) траектории. При переносе эта ортогональность сохраняется.

Рассмотрим основные методы данного интерфейса:

- GetSketch() – возвращает интерфейс ksEntity, описывающий эскиз сечения;
- PathPartArray() – возвращает интерфейс ksEntityCollection, описывающий массив объектов ksEntity;
- SetSketch() – установить эскиз сечения.

Ниже приведен фрагмент программы, демонстрирующей использование этой операции:

```
namespace base_evolution_definition
{
    public class base_evolution_definition
    {
        public static int pTop_Part = -1;
        public static short o3d_planeX0Z = 2;
        public static short o3d_planeY0Z = 3;
        public static short o3d_sketch = 5;
        public static short o3d_baseEvolution = 45;
        public static short vm_Shaded = 3;

        public static KompasObject kompas;
        public static ksDocument3D doc_3d;
        public static ksPart part;
        public static ksEntity entity_sketch1, entity_sketch2,
entity_planeY0Z, entity_planeX0Z, entity_evolution;
        public static ksEntityCollection entity_collection;
        public static ksSketchDefinition sketch_def;
        public static ksDocument2D doc_2d;
        public static ksBaseEvolutionDefinition base_evolution_def;
```



```

public static void Main()
{
    //Подключаемся к КОМПАСу
    Kompas =
(KompasObject)Marshal.GetActiveObject("Kompas.Application.5");
    //Получаем интерфейс документа-модели
    doc_3d = (ksDocument3D)Kompas.Document3D();
    //Создаем документ-модель
    doc_3d.Create(false, true);
    //Получаем интерфейс компонента
    part = (ksPart)doc_3d.GetPart(pTop_Part);
    //Получаем интерфейс объекта "плоскость YOZ"
    entity_planeYOZ =
(ksEntity)part.GetDefaultEntity(o3d_planeYOZ);
    //Получаем интерфейс объекта "плоскость XOZ"
    entity_planeXOZ =
(ksEntity)part.GetDefaultEntity(o3d_planeXOZ);

    //Эскиз 1 (сечение)
    //Получаем интерфейс объекта
    entity_sketch1 = (ksEntity)part.NewEntity(o3d_sketch);
    //Получаем интерфейс параметров объекта
    sketch_def =
(ksSketchDefinition)entity_sketch1.GetDefinition();
    //Устанавливаем плоскость эскиза
    sketch_def.SetPlane(entity_planeYOZ);
    //Создаем эскиз
    entity_sketch1.Create();
    //Входим в режим редактирования эскиза
    doc_2d = (ksDocument2D)sketch_def.BeginEdit();
    //Строим окружность
    doc_2d.KsCircle(-20, -20, 20, 1);
    //Выходим из режима редактирования эскиза
    sketch_def.EndEdit();

    //Эскиз 2 (траектория)
    //Получаем интерфейс объекта
    entity_sketch2 = (ksEntity)part.NewEntity(o3d_sketch);
    //Получаем интерфейс параметров объекта
    sketch_def =
(ksSketchDefinition)entity_sketch2.GetDefinition();
    //Устанавливаем плоскость эскиза
    sketch_def.SetPlane(entity_planeXOZ);
    //Создаем эскиз
    entity_sketch2.Create();
    //Входим в режим редактирования эскиза
    doc_2d = (ksDocument2D)sketch_def.BeginEdit();
    //Строим траекторию
    doc_2d.KsLineSeg(0, -20, 30, -20, 1);
    doc_2d.KsArcByPoint(30, -40, 20, 30, -20, 50, -40, -1,
1);
    doc_2d.KsLineSeg(50, -40, 50, -80, 1);
    //Выходим из режима редактирования эскиза
    sketch_def.EndEdit();

```

```

        //Кинематическая операция выдавливания
        //Получаем интерфейс операции
        entity_evolution =
(ksEntity)part.NewEntity(o3d_baseEvolution);
        //Получаем интерфейс параметров операции
        base_evolution_def =
(ksBaseEvolutionDefinition)entity_evolution.GetDefinition();
        //Тип движения
        base_evolution_def.sketchShiftType = 1;
        //Устанавливаем эскиз сечения
        base_evolution_def.SetSketch(entity_sketch1);
        //Получаем массив объектов
        entity_collection =
(ksEntityCollection)base_evolution_def.PathPartArray();
        entity_collection.Clear();
        //Добавляем в массив эскиз с траекторией
        entity_collection.Add(entity_sketch2);
        //Создаем операцию
        entity_evolution.Create();
        //Устанавливаем полутонное изображение модели
        doc_3d.drawMode = vm_Shaded;
        //Включаем отображение каркаса
        doc_3d.shadedWireframe = true;
        //Делаем КОМПАС видимым
        kompas.Visible = true;
    }
}

```

## Кинематическая операция приклеивания. Параметры кинематической операции приклеивания (ksBossEvolutionDefenition)

Параметры кинематической операции приклеивания задаются с помощью интерфейса ksBossEvolutionDefenition. Его возвращает метод GetDefinition() интерфейса ksENtity.

Интерфейс ksBossEvolutionDefenition полностью аналогичен интерфейсу ksBaseEvolutionDefinition. Ниже приводится фрагмент программы, демонстрирующей использование этой операции:

```

namespace boss_evolution_definition
{
    public class boss_evolution_definition
    {
        public static int pTop_Part = -1;
        public static short o3d_planeX0Z = 2;
        public static short o3d_planeY0Z = 3;
        public static short o3d_sketch = 5;
        public static short o3d_planeOffset = 14;
        public static short o3d_baseExtrusion = 24;
        public static short o3d_bossEvolution = 46;
    }
}

```

```

public static short dtReverse = 1;
public static short etBlind = 0;
public static short vm_Shaded = 3;

public static KompasObject kompas;
public static ksDocument3D doc_3d;
public static ksPart part;
public static ksEntity entity_sketch1, entity_sketch2,
entity_sketch3, entity_planeYOZ, entity_planeXOZ,
entity_plane_offset, entity_boss_evolution,
entity_extrusion;
public static ksEntityCollection entity_collection;
public static ksSketchDefinition sketch_def;
public static ksDocument2D doc_2d;
public static ksPlaneOffsetDefinition plane_offset_def;
public static ksBaseExtrusionDefinition base_extrusion_def;
public static ksBossEvolutionDefinition boss_evolution_def;
public static void Main()
{
    //Подключаемся к КОМПАСу
    kompas =
(KompasObject)Marshal.GetActiveObject("Kompas.Application.5");
    //Получаем интерфейс документа-модели
    doc_3d = (ksDocument3D)kompas.Document3D();
    //Создаем документ-модель
    doc_3d.Create(false, true);
    //Получаем интерфейс компонента
    part = (ksPart)doc_3d.GetPart(pTop_Part);
    //Получаем интерфейс объекта "плоскость YOZ"
    entity_planeYOZ =
(ksEntity)part.GetDefaultEntity(o3d_planeYOZ);
    //Получаем интерфейс объекта "плоскость XOZ"
    entity_planeXOZ =
(ksEntity)part.GetDefaultEntity(o3d_planeXOZ);

    //Эскиз 1
    //Получаем интерфейс объекта
    entity_sketch1 = (ksEntity)part.NewEntity(o3d_sketch);
    //Получаем интерфейс параметров объекта
    sketch_def =
(ksSketchDefinition)entity_sketch1.GetDefinition();
    //Устанавливаем плоскость эскиза
    sketch_def.SetPlane(entity_planeYOZ);
    //Создаем эскиз
    entity_sketch1.Create();
    //Входим в режим редактирования эскиза
    doc_2d = (ksDocument2D)sketch_def.BeginEdit();
    //Строим окружность
    doc_2d.KsCircle(-20, -20, 20, 1);
    //Выходим из режима редактирования эскиза
    sketch_def.EndEdit();

    //Операция выдавливания
    //Получаем интерфейс операции

```

```

        entity_extrusion =
(ksEntity)part.NewEntity(o3d_baseExtrusion);
        //Получаем интерфейс параметров операции
        base_extrusion_def =
(ksBaseExtrusionDefinition)entity_extrusion.GetDefinition();
        //Обратное направление
        base_extrusion_def.directionType = dtReverse;
        //Устанавливаем параметры операции
        base_extrusion_def.SetSideParam(false, etBlind, 30, 0,
true);

        //Устанавливаем эскиз операции
        base_extrusion_def.SetSketch(entity_sketch1);
        //Создаем операцию
        entity_extrusion.Create();

        //Смещенная плоскость
        //Получаем интерфейс объекта
        entity_plane_offset =
(ksEntity)part.NewEntity(o3d_planeOffset);
        //Получаем интерфейс параметров объекта
        plane_offset_def =
(ksPlaneOffsetDefinition)entity_plane_offset.GetDefinition();
        //Обратное направление
        plane_offset_def.direction = false;
        //Смещение
        plane_offset_def.offset = 30;
        //Устанавливаем базовую плоскость
        plane_offset_def.SetPlane(entity_planeY0Z);
        //Создаем смещенную плоскость
        entity_plane_offset.Create();

        //Эскиз 2 (сечение)
        //Получаем интерфейс объекта
        entity_sketch2 = (ksEntity)part.NewEntity(o3d_sketch);
        //Получаем интерфейс параметров объекта
        sketch_def =
(ksSketchDefinition)entity_sketch2.GetDefinition();
        //Устанавливаем плоскость эскиза
        sketch_def.SetPlane(entity_plane_offset);
        //Создаем эскиз
        entity_sketch2.Create();
        //Входим в режим редактирования эскиза
        doc_2d = (ksDocument2D)sketch_def.BeginEdit();
        //Строим окружность
        doc_2d.kscircle(-20, -20, 20, 1);
        //Выходим из режима редактирования эскиза
        sketch_def.EndEdit();

        //Эскиз 3 (траектория)
        //Получаем интерфейс объекта
        entity_sketch3 = (ksEntity)part.NewEntity(o3d_sketch);
        //Получаем интерфейс параметров объекта
        sketch_def =
(ksSketchDefinition)entity_sketch3.GetDefinition();
        //Устанавливаем плоскость эскиза

```

```

        sketch_def.SetPlane(entity_planeXOZ);
        //Создаем эскиз
        entity_sketch3.Create();
        //Входим в режим редактирования эскиза
        doc_2d = (ksDocument2D)sketch_def.BeginEdit();
        //Строим траекторию
        doc_2d.ksArcByPoint(30, -40, 20, 30, -20, 50, -40, -1,
1);

        doc_2d.ksLineSeg(50, -40, 50, -80, 1);
        //Выходим из режима редактирования эскиза
        sketch_def.EndEdit();

        //Кинематическая операция приклеивания
        //Получаем интерфейс операции
        entity_boss_evolution =
(ksEntity)part.NewEntity(o3d_bossEvolution);
        //Получаем интерфейс параметров операции
        boss_evolution_def =
(ksBossEvolutionDefinition)entity_boss_evolution.GetDefinition();
        //Тип движения
        boss_evolution_def.sketchShiftType = 1;
        //Устанавливаем эскиз сечения
        boss_evolution_def.SetSketch(entity_sketch2);
        //Получаем массив объектов
        entity_collection =
(ksEntityCollection)boss_evolution_def.PathPartArray();
        entity_collection.Clear();
        //Добавляем в массив эскиз с траекторией
        entity_collection.Add(entity_sketch3);
        //Создаем операцию
        entity_boss_evolution.Create();
        //Делаем плоскости невидимыми
        doc_3d.hideAllPlanes = true;
        //Устанавливаем полутонное изображение модели
        doc_3d.drawMode = vm_Shaded;
        //Включаем отображение каркаса
        doc_3d.shadedWireframe = true;
        //Делаем КОМПАС видимым
        kompas.Visible = true;
    }
}

```