**Лабораторная работа №4.**

# **Кинематическая операция вырезания. Параметры кинематической операции вырезания (ksCutEvolutionDefinition)**

Параметры кинематической операции вырезания задаются с помощью интерфейса ksCutEvolutionDefinition. Его возвращает метод GetDefinition() интерфейс ksEntity.

Единственное отличие интерфейса ksCutEvolutionDefinition от ksBaseEvolutionDefinition состоит в том, что у первого появилось одно дополнительное свойство по сравнению со вторым:

* Cut – указывает характер операции. Если true, то имеет место вычитание объектов (классическое вырезание), иначе (false) имеет место пересечение объектов.

Ниже представлен фрагмент программы, демонстрирующей использование данной операции:

namespace cut\_evolution\_defenition

{

public class cut\_evolution\_defenition

{

public static int pTop\_Part = -1;

public static short o3d\_planeXOY = 1;

public static short o3d\_planeXOZ = 2;

public static short o3d\_planeYOZ = 3;

public static short o3d\_sketch = 5;

public static short o3d\_baseExtrusion = 24;

public static short o3d\_cutEvolution = 47;

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\_planeXOY, entity\_planeYOZ,

entity\_planeXOZ, entity\_extrusion, entity\_cut\_evolution;

public static ksEntityCollection entity\_collection;

public static ksSketchDefinition sketch\_def;

public static ksDocument2D doc\_2d;

public static ksBaseExtrusionDefinition base\_extrusion\_def;

public static ksCutEvolutionDefinition cut\_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 XOY"

entity\_planeXOY = (ksEntity)part.GetDefaultEntity(o3d\_planeXOY);

//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\_planeXOY);

//Создаем эскиз

entity\_sketch1.Create();

//Входим в режим редактирования эскиза

doc\_2d = (ksDocument2D)sketch\_def.BeginEdit();

//Строим окружность

doc\_2d.ksCircle(50, 25, 50, 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();

//Направление выдавливания прямое

base\_extrusion\_def.directionType = dtNormal;

//Устанавливаем параметры операции

base\_extrusion\_def.SetSideParam(true, etBlind, 80, 0, true);

//Устанавливаем эскиз операции

base\_extrusion\_def.SetSketch(entity\_sketch1);

//Создаем операцию

entity\_extrusion.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\_planeYOZ);

//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.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 вырезания

//Получаем интерфейс операции

entity\_cut\_evolution = (ksEntity)part.NewEntity(o3d\_cutEvolution);

//Получаем интерфейс параметров операции

cut\_evolution\_def = (ksCutEvolutionDefinition)entity\_cut\_evolution.GetDefinition();

//Вычитане объектов

cut\_evolution\_def.cut = true;

//Тип движения

cut\_evolution\_def.sketchShiftType = 1;

//Устанавливаем эскиз сечения

cut\_evolution\_def.SetSketch(entity\_sketch2);

//Получаем массив объектов

entity\_collection = (ksEntityCollection)cut\_evolution\_def.PathPartArray();

entity\_collection.Clear();

//Добавляем в массив эскиз с траекторией

entity\_collection.Add(entity\_sketch3);

//Создаем операцию

entity\_cut\_evolution.Create();

//Устанавливаем полутоновое изображение модели

doc\_3d.drawMode = vm\_Shaded;

//Включаем отображение каркаса

doc\_3d.shadedWireframe = true;

//Делаем КОМПАС видимым

kompas.Visible = true;

}

}

# **Операция «сечение плоскостью». Параметры операции «сечение плоскостью» (ksCutByPlaneDefenition)**

Параметры операции «сечение плоскостью» задаются с помощью интерфейса ksCutByPlaneDefinition. Его возвращает метод GetDefinition() интерфейса ksEntity, описывающего эту операцию. Ниже представлены свойства данного интерфейса:

* Direction – задает направление отсечения. Если true, то прямое направление, иначе (false) обратное направление;
* ksChooseType – область применения операции. В таблице ниже представлены допустимые значения данного свойства.

|  |  |  |
| --- | --- | --- |
| **Область применения** | **Значение** | **Описание** |
| ksChBodiesAndParts | 1 | Компоненты и тела |
| ksChParts | 2 | Компоненты |
| ksChBodies | 3 | Все компоненты |

Теперь рассмотрим методы интерфейса ksCutByPlaneDefinition:

* GetPlane() - возвращает интерфейс ksEntity, описывающий секущую плоскость;
* SetPlane() - устанавливает новую секущую плоскость. В качестве единственного параметра данный метод принимает интерфейс ksEntity, описывающий новую секущую плоскость.

Ниже представлен фрагмент программы, демонстрирующей использование данной операции:

namespace cut\_by\_plane\_defenition

{

public class cut\_by\_plane\_defenition

{

public static int pTop\_Part = -1;

public static short o3d\_planeXOY = 1;

public static short o3d\_planeYOZ = 3;

public static short o3d\_sketch = 5;

public static short o3d\_axis2Planes = 9;

public static short o3d\_planeAngle = 15;

public static short o3d\_baseExtrusion = 24;

public static short o3d\_cutByPlane = 50;

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\_planeXOY, entity\_planeYOZ, entity\_plane\_angle, entity\_extrusion, entity\_cut\_by\_plane,

entity\_axis, entity\_draw;

public static ksSketchDefinition sketch\_def;

public static ksDocument2D doc\_2d;

public static ksBaseExtrusionDefinition base\_extrusion\_def;

public static ksAxis2PlanesDefinition axis\_plane\_def;

public static ksPlaneAngleDefinition plane\_angle\_def;

public static ksCutByPlaneDefinition cut\_by\_plane\_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 YOZ"

entity\_planeYOZ = (ksEntity)part.GetDefaultEntity(o3d\_planeYOZ);

//Эскиз

//Получаем интерфейс объекта "Эскиз"

entity\_draw = (ksEntity)part.NewEntity(o3d\_sketch);

//Получаем интерфейс параметров эскиза

sketch\_def = (ksSketchDefinition)entity\_draw.GetDefinition();

//Устанавливаем плоскость XOY базовой для эскиза

sketch\_def.SetPlane(entity\_planeXOY);

//Создаем эскиз

entity\_draw.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 "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();

//Направление выдавливания прямое

base\_extrusion\_def.directionType = dtNormal;

//Устанавливаем параметры операции выдавливания

base\_extrusion\_def.SetSideParam(true, etBlind, 50, 0, true);

//Устанавливаем эскиз операции выдавливания

base\_extrusion\_def.SetSketch(entity\_draw);

//Создаем операцию выдавливания

entity\_extrusion.Create();

//Ось на пересечении двух плоскостей

//Получаем интерфейс объекта

entity\_axis = (ksEntity)part.NewEntity(o3d\_axis2Planes);

//Получаем интерфейс параметров объекта

axis\_plane\_def = (ksAxis2PlanesDefinition)entity\_axis.GetDefinition();

//Устанавливаем плоскости

axis\_plane\_def.SetPlane(1, entity\_planeXOY);

axis\_plane\_def.SetPlane(2, entity\_planeYOZ);

//Создаем ось

entity\_axis.Create();

//Наклоненная плоскость

//Получаем интерфейс объекта

entity\_plane\_angle = (ksEntity)part.NewEntity(o3d\_planeAngle);

//Получаем интерфейс параметров объекта

plane\_angle\_def = (ksPlaneAngleDefinition)entity\_plane\_angle.GetDefinition();

//Угол наклона

plane\_angle\_def.angle = 60;

//Устанавливаем ось

plane\_angle\_def.SetAxis(entity\_axis);

//Устанавливаем базовую плоскость

plane\_angle\_def.SetPlane(entity\_planeXOY);

//Создаем наклоненную плоскость

entity\_plane\_angle.Create();

//Операция "сечение плоскостью"

//Получаем интерфейс объекта

entity\_cut\_by\_plane = (ksEntity)part.NewEntity(o3d\_cutByPlane);

//Получаем интерфейс параметров объекта

cut\_by\_plane\_def = (ksCutByPlaneDefinition)entity\_cut\_by\_plane.GetDefinition();

//Направление - обратное

cut\_by\_plane\_def.direction = false;

//Устанавливаем секущую плоскость

cut\_by\_plane\_def.SetPlane(entity\_plane\_angle);

//Создаем операцию

entity\_cut\_by\_plane.Create();

//Делаем все плоскости невидимыми

doc\_3d.hideAllPlanes = true;

//Делаем все оси невидимыми

doc\_3d.hideAllAxis = true;

//Устанавливаем полутоновое изображение модели

doc\_3d.drawMode = vm\_Shaded;

//Включаем отображение каркаса

doc\_3d.shadedWireframe = true;

//Делаем КОМПАС видимым

kompas.Visible = true;

}

}

# **Операция «сечение эскизом». Параметры операции «сечение эскизом» (ksCutBySketchDefinition)**

Параметры операции «сечение эскизом» задаются с помощью интерфейса ksCutBySketchDefinition. Его возвращает метод GetDefinition() интерфейса ksEntity, описывающего данную операцию. Ниже представлены свойства данного интерфейса:

* Direction – направление отсечения;
* chooseType – область применения операции. Значение данногосвойства аналогично значению свойства chooseType интерфейса ksCutByPlane.

Ниже рассмотрены методы интерфейса ksCutBySketch:

* GetSketch() – возвращает интерфейс ksEntity, описывающий секущий эскиз;
* SetSketch() – устанавливает новый секущий эскиз.

Ниже представлен фрагмент программы, демонстрирующей использование этой операции:

namespace cut\_by\_sketch\_defenition

{

public class cut\_by\_sketch\_defenition

{

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 o3d\_cutBySketch = 51;

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\_planeXOY, entity\_sketch, entity\_extrusion, entity\_cut\_by\_sketch;

public static ksSketchDefinition sketch\_def;

public static ksDocument2D doc\_2d;

public static ksBaseExtrusionDefinition base\_extrusion\_def;

public static ksCutBySketchDefinition cut\_by\_sketch\_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

//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(20, 20, 20, 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();

//Направление выдавливания прямое

base\_extrusion\_def.directionType = dtNormal;

//Устанавливаем параметры операции выдавливания

base\_extrusion\_def.SetSideParam(true, etBlind, 50, 0, true);

//Устанавливаем эскиз операции выдавливания

base\_extrusion\_def.SetSketch(entity\_sketch);

//Создаем операцию выдавливания

entity\_extrusion.Create();

//Эскиз

//Получаем интерфейс объекта

entity\_sketch = (ksEntity)part.NewEntity(o3d\_sketch);

//Получаем интерфейс параметров объекта

sketch\_def = (ksSketchDefinition)entity\_sketch.GetDefinition();

//Устанавливаем плоскость эскиза

sketch\_def.SetPlane(entity\_planeXOY);

//Создаем эскиз

entity\_sketch.Create();

//Входим в режим редактирования эскиза

doc\_2d = (ksDocument2D)sketch\_def.BeginEdit();

//Строим сам эскиз

doc\_2d.ksLineSeg(20, 20, 20, 50, 1);

doc\_2d.ksLineSeg(20, 20, 50, 20, 1);

//Выходим из режима редактирования эскиза

sketch\_def.EndEdit();

//Операция "сечение эскизом"

//Получаем интерфейс объекта

entity\_cut\_by\_sketch = (ksEntity)part.NewEntity(o3d\_cutBySketch);

//Получаем интерфейс параметров объекта

cut\_by\_sketch\_def = (ksCutBySketchDefinition)entity\_cut\_by\_sketch.GetDefinition();

//Направление - прямое

cut\_by\_sketch\_def.direction = true;

//Устанавливаем эскиз операции

cut\_by\_sketch\_def.SetSketch(entity\_sketch);

//Создаем операцию

entity\_cut\_by\_sketch.Create();

//Делаем все плоскости невидимыми

doc\_3d.hideAllPlanes = true;

//Устанавливаем полутоновое изображение модели

doc\_3d.drawMode = vm\_Shaded;

//Включаем отображение каркаса

doc\_3d.shadedWireframe = true;

//Делаем КОМПАС видимым

kompas.Visible = true;

}

}

# **Скругление. Параметры скругления (ksFilletDefinition)**

Параметры скругления задаются с помощью интерфейса ksFilletDefinition. Его возвращает метод GetDefinition() интерфейса ksEntity, описывающего скругление. Ниже рассмотрены свойства данного интерфейса:

* Radius – радиус скругления;
* Tangent – признак продолжения скругления по касательным ребрам.

Метод у интерфейса ksFilletDefinition всего один:

* Array\_() – возвращает интерфейс ksEntityCollection массива скругляемых граней или ребер.

Ниже представлен фрагмент программы, демонстрирующей построение скругления:

namespace fillet\_defenition

{

public class fillet\_defenition

{

public static int pTop\_Part = -1;

public static short o3d\_planeXOY = 1;

public static short o3d\_sketch = 5;

public static short o3d\_face = 6;

public static short o3d\_baseExtrusion = 24;

public static short o3d\_fillet = 34;

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, entity\_planeXOY, entity\_extrusion, entity\_fillet;

public static ksEntityCollection entity\_collection\_fillet, entity\_collection\_part;

public static ksSketchDefinition sketch\_def;

public static ksDocument2D doc\_2d;

public static ksBaseExtrusionDefinition base\_extrusion\_def;

public static ksFilletDefinition fillet\_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

//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(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, 50, 0, true);

//Устанавливаем эскиз операции выдавливания

base\_extrusion\_def.SetSketch(entity\_sketch);

//Создаем операцию выдавливания

entity\_extrusion.Create();

//Скругление

//Получаем интерфейс объекта "скругление"

entity\_fillet = (ksEntity)part.NewEntity(o3d\_fillet);

//Получаем интерфейс параметров объекта "скругление"

fillet\_def = (ksFilletDefinition)entity\_fillet.GetDefinition();

//Радиус скругления

fillet\_def.radius = 5;

//Не продолжать по касательным ребрам

fillet\_def.tangent = false;

//Получаем массив граней детали

entity\_collection\_part = (ksEntityCollection)part.EntityCollection(o3d\_face);

//Получаем массив скругляемых граней

entity\_collection\_fillet = (ksEntityCollection)fillet\_def.array();

entity\_collection\_fillet.Clear();

//Заполняем массив скругляемых граней

entity\_collection\_fillet.Add(entity\_collection\_part.GetByIndex(0));

//Создаем скругление

entity\_fillet.Create();

//Устанавливаем полутоновое изображение модели

doc\_3d.drawMode = vm\_Shaded;

//Включаем отображение каркаса

doc\_3d.shadedWireframe = true;

//Делаем КОМПАС видимым

kompas.Visible = true;

}

}

# **Фаска. Параметры фаски (ksChamferDefinition)**

Параметры фаски задаются с помощью интерфейса ksChamferDefinition. Его возвращает метод GetDefinition() интерфейса ksEntity, описывающего фаску. У интерфейса ksChamferDefintion всего одно свойство:

* Tangent – признак продолжения фаски по касательным ребрам.

Ниже приведены методы интерфейса ksChamferDefinition:

* Array\_() – возвращает интерфейс массива ksEntityCollection граней или ребер;
* SetChamferParam(bool transfer, double distance1, distance2) – устанавливает основные параметры фаски;
* GetChamferParam() – позволяет получить основные параметры фаски.

Ниже представлен фрагмент программы, демонстрирующей построение фаски:

namespace chamfer\_defenition

{

public class chamfer\_defenition

{

public static int pTop\_Part = -1;

public static short o3d\_planeXOY = 1;

public static short o3d\_sketch = 5;

public static short o3d\_face = 6;

public static short o3d\_baseExtrusion = 24;

public static short o3d\_chamfer = 33;

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, entity\_planeXOY, entity\_extrusion, entity\_chamfer;

public static ksEntityCollection entity\_collection\_chamfer, entity\_collection\_part;

public static ksSketchDefinition sketch\_def;

public static ksDocument2D doc\_2d;

public static ksBaseExtrusionDefinition base\_extrusion\_def;

public static ksChamferDefinition chamfer\_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

//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(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, 50, 0, true);

//Устанавливаем эскиз операции выдавливания

base\_extrusion\_def.SetSketch(entity\_sketch);

//Создаем операцию выдавливания

entity\_extrusion.Create();

//Фаска

//Получаем интерфейс объекта "фаска"

entity\_chamfer = (ksEntity)part.NewEntity(o3d\_chamfer);

//Получаем интерфейс параметров объекта "скругление"

chamfer\_def = (ksChamferDefinition)entity\_chamfer.GetDefinition();

//Не продолжать по касательным ребрам

chamfer\_def.tangent = false;

//Устанавливаем параметры фаски

chamfer\_def.SetChamferParam(true, 8, 8);

//Получаем массив граней детали

entity\_collection\_part = (ksEntityCollection)part.EntityCollection(o3d\_face);

//Получаем массив граней, на которых будет строиться фаска

entity\_collection\_chamfer = (ksEntityCollection)chamfer\_def.array();

entity\_collection\_chamfer.Clear();

//Заполняем массив граней, на которых будет строится фаска

entity\_collection\_chamfer.Add(entity\_collection\_part.GetByIndex(0));

//Создаем фаску

entity\_chamfer.Create();

//Устанавливаем полутоновое изображение модели

doc\_3d.drawMode = vm\_Shaded;

//Включаем отображение каркаса

doc\_3d.shadedWireframe = true;

//Делаем КОМПАС видимым

kompas.Visible = true;

}

}