

МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ  
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Металлорежущие станки и инструменты»

# ОСНОВЫ АВТОМАТИЗАЦИИ КОНСТРУИРОВАНИЯ

*Методические рекомендации к курсовому проектированию  
для студентов специальности  
6-05-0611-01 «Информационные системы и технологии»  
дневной и заочной форм обучения*



Могилев 2025

УДК 004.4: 621.8  
ББК 32.973.26-02: 34.4  
О75

Рекомендовано к изданию  
учебно-методическим отделом  
Белорусско-Российского университета

Одобрено кафедрой «Металлорежущие станки и инструменты» «18» сентября  
2025 г., протокол № 2

Составители: канд. техн. наук, доц. С. Н. Хатетовский;  
д-р техн. наук, проф. П. Н. Громыко

Рецензент канд. техн. наук, доц. Е. В. Ильюшина

Методические рекомендации к курсовому проектированию предназначены  
для студентов специальности 6-05-0611-01 «Информационные системы и тех-  
нологии» дневной и заочной форм обучения.

Учебное издание

## ОСНОВЫ АВТОМАТИЗАЦИИ КОНСТРУИРОВАНИЯ

|                         |                   |
|-------------------------|-------------------|
| Ответственный за выпуск | С. Н. Хатетовский |
| Корректор               | И. В. Голубцова   |
| Компьютерная верстка    | М. М. Дударева    |

Подписано в печать . Формат 60×84/16. Бумага офсетная. Гарнитура Таймс.  
Печать трафаретная. Усл. печ. л. . Уч.-изд. л. . Тираж 56 экз. Заказ №

Издатель и полиграфическое исполнение:  
Межгосударственное образовательное учреждение высшего образования  
«Белорусско-Российский университет».  
Свидетельство о государственной регистрации издателя,  
изготовителя, распространителя печатных изданий  
№ 1/156 от 07.03.2019.  
Пр-т Мира, 43, 212022, г. Могилев.

© Белорусско-Российский  
университет, 2025

## Содержание

|                                                             |    |
|-------------------------------------------------------------|----|
| Введение.....                                               | 4  |
| 1 Основные методы классов API SOLIDWORKS на языке VBA ..... | 5  |
| 2 Основы программирования в среде САПР NX.....              | 20 |
| 3 Примерный перечень вариантов.....                         | 23 |
| 4 Структура и оформление курсовой работы .....              | 23 |
| Список литературы .....                                     | 24 |

## Введение

Под автоматизацией конструкторских работ подразумевается использование систем автоматизированного проектирования (САПР). Создание в среде САПР 3D-моделей изделия (детали, сборочного узла) является одной из первых стадий жизненного цикла изделия.

Большинство современных САПР позволяют создавать 3D-модели не только с помощью графического интерфейса, но и с помощью специального программного обеспечения (ПО). Это ПО использует библиотеки процедур и функций, называемые API (от Application Programming Interface). Как правило, процедуры и функции API соответствуют командам, выполняемым при помощи графического интерфейса.

Назначение API в САПР:

- расширение базового инструментария САПР;
- автоматизация создания 3D-моделей типовых деталей;
- параметрическое моделирование, когда в качестве параметров выступают специфичные для изделия параметры.

Целью курсового проектирования по дисциплине «Основы автоматизации конструирования» является получение навыков работы с API одной из САПР.

# 1 Основные методы классов API SOLIDWORKS на языке VBA

## 1.1 Основные классы

Объект класса **SldWorks** предоставляет доступ к приложению.

Объект класса **ModelDoc2** предоставляет доступ к документам приложения.

Объект класса **PartDoc** предоставляет доступ к документам типа Деталь.

Если в приложении активным является документ типа Деталь, одна и та же ссылка на данные объектов классов **ModelDoc2** и **PartDoc** возвращается следующим методом класса **SldWorks**:

### Public Property ActiveDoc() As Object

Объект класса **ModelDocExtension** предоставляет расширенный доступ к документу типа Деталь. Этот объект возвращается следующим свойством класса **ModelDoc2**:

### Public Property Extension() As ModelDocExtension

Объект класса **FeatureManager** позволяет работать с элементами дерева построения. Этот объект возвращается следующим свойством класса **ModelDoc2**:

### Public Property FeatureManager() As FeatureManager

Объект класса **SketchManager** предоставляет доступ к эскизам и возвращается следующим свойством класса **ModelDoc2**:

### Public Property SketchManager() As SketchManager

Элементы дерева построения представляются объектами класса **Feature** и создаются специальными методами некоторых классов, в том числе классов **FeatureManager** и **ModelDoc2**.

Такие элементы эскиза, как линия, дуга окружности (окружность), сплайн и некоторые другие, представляются объектами класса **SketchSegment**. Точкам эскиза соответствует класс **SketchPoint**. Линии эскиза, дуги окружности эскиза и сплайны эскиза дополнительно представляются объектами соответственно классов **SketchLine**, **SketchArc** и **SketchSpline**. Свойства и методы классов **SketchLine**, **SketchArc** и **SketchSpline** доступны по ссылке на данные соответствующих объектов класса **SketchSegment**. Элементы эскиза создаются специальными методами некоторых классов, в том числе классов **SketchManager** и **ModelDoc2**.

## 1.2 Идентификация элементов дерева построения

Чтение-запись имени элемента дерева построения. Свойство класса **Feature: Public Property Name() As String**.

Чтение типа элемента дерева построения. Свойство класса **Feature: Public Function GetTypeName2() As String**. Возвращаемое значение:

- вырез по сечениям – “BlendCut”;
- фаска – “Chamfer”;
- круговой массив – “CirPattern”;
- массив, управляемый кривой, – “CurvePattern”;
- вырез, полученный вытягиванием, – “Cut”;
- бобышка, полученная вытягиванием, – “Extrusion”;
- простое скругление – “Fillet”;
- спираль/винтовая кривая – “Helix”;
- вырез, созданный поворотом, – “RevCut”;
- бобышка, созданная поворотом, – “Revolution”;
- бобышка, созданная заметанием, – “Sweep”;
- вырез, созданный заметанием, – “SweepCut”;
- скругление переменного радиуса – “VarFillet”;
- справочная ось – “RefAxis”;
- справочная кривая – “ReferenceCurve”;
- справочная плоскость – “RefPlane”.

Получение первого элемента дерева построения. Метод класса **PartDoc: Public Function FirstFeature() As Object**. Возвращаемое значение: объект класса **Feature**.

Получение следующего элемента дерева построения. Метод класса **Feature: Public Function GetNextFeature() As Object**. Возвращаемое значение: объект класса **Feature**. Подавленные элементы также возвращаются.

Получение последнего добавленного элемента дерева построения. Метод класса **ModelDocExtension: Public Function GetLastFeatureAdded() As Feature**.

## 1.3 Выделение

Выделение элемента модели. Метод класса **ModelDocExtension: Function SelectByID2(ByVal Name As String, ByVal Type As String, ByVal X As Double, ByVal Y As Double, ByVal Z As Double, ByVal Append As Boolean, ByVal Mark As Long, ByVal Callout As Callout, ByVal SelectOption As Long ) As Boolean**. Аргументы:

- Name – имя элемента модели или пустая строка;
- Type – тип элемента модели или пустая строка;
- X, Y, Z – координаты точки элемента модели;
- Mark – номер элемента модели в выделении;
- Callout – объект класса **Callout** (или **Nothing**);
- SelectOption – опции – значение из перечисления **swSelectOption\_e**.

Если аргумент **Append** равен **True**, то невыделенный элемент модели будет добавлен к существующему выделению или выделенный элемент модели будет исключен из существующего выделения. Если **False**, то существующее выделение будет снято и создано новое из невыделенного элемента модели или существующее выделение, включающее выделенный элемент модели, будет оставлено без изменения. Возвращаемое значение: **True**, если элемент модели стал выделенным. Аргумент **Type** может иметь следующие значения:

- “DATUMPOINT” – справочная точка;
- “AXIS” – справочная ось;
- “PLANE” – справочная плоскость;
- “REFCURVE” – справочная кривая;
- “HELIX” – спираль/винтовая кривая;
- “SKETCH” – эскиз;
- “SKETCHSEGMENT” – элемент эскиза, если эскиз активен;
- “EXTSKETCHSEGMENT” – элемент эскиза, если эскиз неактивен;
- “SKETCHPOINT” – точка эскиза, если эскиз активен;
- “EXTSKETCHPOINT” – точка эскиза, если эскиз неактивен;
- “FACE” – грань;
- “EDGE” – кромка;
- “VERTEX” – вершина;
- “BODYFEATURE” – тело;
- “POINTREF” – точка;
- “NOTHING” – ничего;
- “EVERYTHING” – все.

Аргумент **Name** для размера должен быть полным именем (например, “D1@Sketch2@Part1.SLDPRT”). Координаты точки элемента модели должны соответствовать той системе координат, в которой элемент был создан. Элементы перечисления **swSelectOption\_e**:

- **swSelectOptionDefault** – клавиша Shift не используется;
- **swSelectOptionExtensive** – клавиша Shift используется.

#### ***1.4 Создание справочной геометрии дерева построения***

Создание справочной точки. Метод класса **FeatureManager: Function InsertReferencePoint(ByVal NRefPointType As Long, ByVal NRefPointAlongCurveType As Long, ByVal DDistance\_or\_Percent As Double, ByVal NumberOfRefPoints As Long ) As Variant**. Аргументы:

- **NRefPointType** – тип справочной точки – значение из перечисления **swRefPointType\_e**;
- **NRefPointAlongCurveType** – тип справочной точки на кривой – значение из перечисления **swRefPointAlongCurveType\_e**;
- **DDistance\_or\_Percent** – расстояние или процент длины, если аргумент **NRefPointAlongCurveType** равен **swRefPointAlongCurveDistance** или

**swRefPointAlongCurvePercentage** соответственно;

– **NumberOfRefPoints** – количество справочных точек, если аргумент **NRefPointAlongCurveType** равен **swRefPointAlongCurveEvenlyDistributed**. Возвращаемое значение: одномерный массив объектов класса **Feature**. Элементы перечисления **swRefPointType\_e**:

- **swRefPointInvalid** – недействительная;
- **swRefPointUndefined** – неопределена;
- **swRefPointAlongCurve** – вдоль кривой;
- **swRefPointCenterEdge** – центр кромки;
- **swRefPointFaceCenter** – центр грани;
- **swRefPointFaceVertexProjection** – проекция вершины;
- **swRefPointIntersection** – пересечение.

Создание справочной оси. Метод класса **ModelDoc2: Function InsertAxis2 (ByVal AutoSize As Boolean) As Boolean**. Аргументы: если аргумент **AutoSize** равен **True**, то справочная ось автоматически установит свою длину. Возвращаемое значение: **True**, если справочная ось была создана.

Создание справочной плоскости. Метод класса **FeatureManager: Function InsertRefPlane (ByVal FirstConstraint As Long, ByVal FirstConstraintAngleOrDistance As Double, ByVal SecondConstraint As Long, ByVal SecondConstraintAngleOrDistance As Double, ByVal ThirdConstraint As Long, ByVal ThirdConstraintAngleOrDistance As Double) As Object**. Аргументы:

- **FirstConstraint** – первое ограничение – значение из перечисления **swRefPlaneReferenceConstraints\_e**;
- **FirstConstraintAngleOrDistance** – угол или расстояние для первого ограничения;
- **SecondConstraint** – второе ограничение – значение из перечисления **swRefPlaneReferenceConstraints\_e**;
- **SecondConstraintAngleOrDistance** – угол или расстояние для второго ограничения;
- **ThirdConstraint** – третье ограничение – значение из перечисления **swRefPlaneReferenceConstraints\_e**;
- **ThirdConstraintAngleOrDistance** – угол или расстояние для третьего ограничения.

Возвращаемое значение: объект класса **RefPlane**. До создания справочной плоскости элементы модели должны быть выделены методом **SelectByID2** со следующими порядковыми номерами:

- 0 – первый элемент;
- 1 – второй элемент;
- 2 – третий элемент.

Элементы перечисления **swRefPlaneReferenceConstraints\_e**:

- **swRefPlaneReferenceConstraint\_Parallel** – параллельность;
- **swRefPlaneReferenceConstraint\_Perpendicular** – перпендикулярность;
- **swRefPlaneReferenceConstraint\_Coincident** – совпадение;



- **swRefPlaneReferenceConstraint\_Distance** – расстояние;
- **swRefPlaneReferenceConstraint\_Angle** – угол;
- **swRefPlaneReferenceConstraint\_Tangent** – касательность;
- **swRefPlaneReferenceConstraint\_Project** – проекция точки эскиза (вершины или начала координат);
- **swRefPlaneReferenceConstraint\_MidPlane** – средняя плоскость;
- **swRefPlaneReferenceConstraint\_OptionFlip** – реверсирование;
- **swRefPlaneReferenceConstraint\_OptionOriginOnCurve** – точка плоскости на кривой;
- **swRefPlaneReferenceConstraint\_OptionProjectToNearestLocation** – проекция точки эскиза (вершины или начала координат) на касательную плоскость к поверхности, ближайшую к точке эскиза (вершине или началу координат);
- **swRefPlaneReferenceConstraint\_OptionProjectAlongSketchNormal** – проекция точки эскиза на касательную плоскость к поверхности по нормали эскиза.

### 1.5 Создание эскизов

Создание 3D-эскиза или выход из активного 3D-эскиза. Метод класса **SketchManager: Public Sub Insert3DSketch(ByVal UpdateEditRebuild As Boolean)**. Аргументы: если аргумент **UpdateEditRebuild** равен **True**, то будут осуществлены обновление и перестройка.

Создание 2D-эскиза или выход из активного 2D-эскиза. Метод класса **SketchManager: Public Sub InsertSketch(ByVal UpdateEditRebuild As Boolean)**. Аргументы: если аргумент **UpdateEditRebuild** равен **True**, то будут осуществлены обновление и перестройка.

### 1.6 Управление отображением элементов эскиза

Чтение-запись типа добавления элемента эскиза в базу данных. Свойство класса **SketchManager: Public Property AddToDB() As Boolean**. Значение: **True** – элемент эскиза будет добавлен в базу данных напрямую, минуя отображение. Значение **False** по умолчанию необходимо установить до завершения работы программы.

Чтение-запись типа отображения элемента эскиза после его добавления в базу данных напрямую. Свойство класса **SketchManager: Public Property DisplayWhenAdded() As Boolean**. Значение: **False** – элемент эскиза не будет отображен после его добавления в базу данных напрямую. Значение **True** по умолчанию необходимо установить до завершения работы программы.

Создание бобышки вытягиванием. Метод класса **FeatureManager: Function FeatureExtrusion2(ByVal Sd As Boolean, ByVal Flip As Boolean, ByVal Dir As Boolean, ByVal T1 As Integer, ByVal T2 As Integer, ByVal D1 As Double, ByVal D2 As Double, ByVal Dchk1 As Boolean, ByVal Dchk2 As Boolean, ByVal Ddir1 As Boolean, ByVal Ddir2 As Boolean, ByVal Dang1 As Double, ByVal Dang2 As Double, ByVal OffsetReverse1 As Boolean, ByVal OffsetReverse2 As Boolean, ByVal TranslateSurface1 As Boolean, ByVal TranslateSurface2 As Boolean, ByVal Merge As Boolean, ByVal UseFeatScope As Boolean, ByVal**

**UseAutoSelect As Boolean, ByVal T0 As Integer, ByVal StartOffset As Double, ByVal FlipStartOffset As Boolean ) As Feature.** Аргументы:

- если аргумент **Sd** равен **True**, то вытягивание осуществляется в одном (первом) направлении;
- если аргумент **Flip** равен **True**, то направление будет реверсировано;
- если аргумент **Dir** равен **True**, то направление будет реверсировано;
- **T1** – значение из перечисления **swEndConditions\_e**, которое задает тип ограничения в первом направлении;
- **T2** – значение из перечисления **swEndConditions\_e**, которое задает тип ограничения во втором направлении;
- **D1** – смещение в первом направлении;
- **D2** – смещение во втором направлении;
- если аргумент **Dchk1** равен **True**, то вытягивание в первом направлении будет осуществляться с уклоном;
- если аргумент **Dchk2** равен **True**, то вытягивание во втором направлении будет осуществляться с уклоном;
- если аргумент **Ddir1** равен **True**, то уклон в первом направлении будет осуществлен внутрь;
- если аргумент **Ddir2** равен **True**, то уклон во втором направлении будет осуществлен внутрь;
- **Dang1** – уклон в первом направлении;
- **Dang2** – уклон во втором направлении;
- если аргумент **OffsetReverse1** равен **True**, то для случая, когда ограничение первого направления задается поверхностью, смещенной на расстояние **D1** относительно заданной поверхности, смещение будет осуществлено в направлении от эскиза;
- если аргумент **OffsetReverse2** равен **True**, то для случая, когда ограничение второго направления задается поверхностью, смещенной на расстояние **D2** относительно заданной поверхности, смещение будет осуществлено в направлении от эскиза;
- если аргумент **TranslateSurface1** равен **True**, то для случая, когда аргумент **T1** равен **swEndCondOffsetFromSurface**, вытягивание в первом направлении будет осуществляться до поверхности, которая смещена относительно заданной поверхности на величину **D1**;
- если аргумент **TranslateSurface2** равен **True**, то для случая, когда аргумент **T2** равен **swEndCondOffsetFromSurface**, вытягивание во втором направлении будет осуществляться до поверхности, которая смещена относительно заданной поверхности на величину **D2**;
- если аргумент **Merge** равен **True**, то бобышка будет объединена с пересекающимися ее телами;
- если аргумент **UseFeatScope** равен **True**, то бобышка будет объединена только с выделенными пересекающимися ее телами, если **False**, то бобышка может быть автоматически объединена со всеми пересекающимися ее телами как уже существующими, так и созданными потом;

- если аргумент **UseAutoSelect** равен **True**, то тела, пересекающие бобышку, будут выделены автоматически, если **False**, то эти тела должны быть выделены;
- **T0** – значение из перечисления **swStartConditions\_e**, которое задаст начальные условия вытягивания;
- **StartOffset** – смещение, если аргумент **T0** равен **swStartOffset**;
- если аргумент **FlipStartOffset** равен **True**, то в том случае, когда аргумент **T0** равен **swStartOffset**, направление смещения будет реверсировано.

Элементы перечисления **swEndConditions\_e**:

- **swEndCondBlind** – «вслепую»;
- **swEndCondThroughAll** – насквозь;
- **swEndCondThroughNext** – через следующее тело;
- **swEndCondUpToVertex** – до вершины;
- **swEndCondUpToSurface** – до поверхности;
- **swEndCondOffsetFromSurface** – расстояние от поверхности;
- **swEndCondMidPlane** – средняя плоскость;
- **swEndCondUpToBody** – до тела.

Элементы перечисления **swStartConditions\_e**:

- **swStartSketchPlane** – плоскость эскиза;
- **swStartSurface** – поверхность;
- **swStartVertex** – вершина;
- **swStartOffset** – смещение.

Создание бобышки по сечениям. Метод класса **FeatureManager: Function InsertProtrusionBlend2(ByVal Closed As Boolean, ByVal KeepTangency As Boolean, ByVal ForceNonRational As Boolean, ByVal TessToleranceFactor As Double, ByVal StartMatchingType As Short, ByVal EndMatchingType As Short, ByVal StartTangentLength As Double, ByVal EndTangentLength As Double, ByVal StartTangentDir As Boolean, ByVal EndTangentDir As Boolean, ByVal IsThinBody As Boolean, ByVal Thickness1 As Double, ByVal Thickness2 As Double, ByVal ThinType As Short, ByVal Merge As Boolean, ByVal UseFeatScope As Boolean, ByVal UseAutoSelect As Boolean, ByVal GuideCurveInfluence As Integer) As Feature**. Аргументы:

- если аргумент **Closed** равен **True**, то направляющая кривая будет автоматически замкнута;
- если аргумент **KeepTangency** равен **True**, то будут образованы касательные поверхности, если сечение состоит из касательных сегментов;
- если аргумент **ForceNonRational** равен **True**, то, когда сечение имеет круговые или эллиптические сегменты, будет осуществляться их аппроксимация для лучшего сглаживания;
- **TessToleranceFactor** – коэффициент, устанавливающий количество промежуточных сечений (по умолчанию равен 1, чем больше, тем больше будет создано промежуточных сечений);
- **StartMatchingType** – тип касания направляющей кривой с исходным сечением - значение из перечисления **swTangencyType\_e**;

- **EndMatchingType** – тип касания направляющей кривой с конечным сечением - значение из перечисления **swTangencyType\_e**;
- **StartTangentLength** – длина вектора, касательного к поверхности, в исходном сечении, если аргумент **StartMatchingType** равен **swTangencyDirectionVector** или **swTangencyNormalToProfile**;
- **EndTangentLength** – длина вектора, касательного к поверхности, в конечном сечении, если аргумент **EndMatchingType** равен **swTangencyDirectionVector** или **swTangencyNormalToProfile**;
- если аргумент **StartTangentDir** равен **False**, то направление касательного вектора в исходном сечении будет реверсировано;
- если аргумент **EndTangentDir** равен **False**, то направление касательного вектора в конечном сечении будет реверсировано;
- если аргумент **IsThinBody** равен **True**, то будет создано тонкостенное тело;
- **Thickness1** – толщина тонкостенного тела, если аргумент **ThinType** равен **swThinWallOneDirection** или **swThinWallOppDirection**, или **swThinWallMidPlane**; толщина тонкостенного тела в первом направлении, если аргумент **ThinType** равен **swThinWallTwoDirection**;
- **Thickness2** – толщина тонкостенного тела во втором направлении, если аргумент **ThinType** равен **swThinWallTwoDirection**;
- **ThinType** – тип толщины тонкостенного тела – значение из перечисления **swThinWallType\_e**;
- **Merge** – см. создать вытянутую бобышку;
- **UseFeatScope** – см. создать вытянутую бобышку;
- **UseAutoSelect** – см. создать вытянутую бобышку;
- **GuideCurveInfluence** – тип влияния направляющей кривой на сегменты сечения – значение из перечисления **swGuideCurveInfluence\_e**.

Для выделения необходимо использовать метод **SelectByID2**; сечению присваивается номер 1, направляющей кривой – номер 2, центральной линии – номер 4, начальному направлению касания – номер 8, конечному направлению касания – номер 32. Аргументы **StartMatchingType** и **EndMatchingType** могут принимать только следующие значения: **swTangencyNone**, **swTangencyNormalToProfile**, **swTangencyDirectionVector**, **swTangencyAllFaces**.

Элементы перечисления **swTangencyType\_e**:

- **swTangencyNone** – направление касательной к направляющей кривой не задано;
- **swTangencyNormalToProfile** – направление касательной к направляющей кривой по нормали к сечению;
- **swTangencyDirectionVector** – направление касательной к направляющей кривой по заданному направлению;
- **swTangencyAllFaces** – к направляющей кромке грани добавляются кромки соседних граней;
- **swMinimumTwist** – направление касательной к направляющей кривой будет обеспечивать минимальное скручивание сечения, устраняющее пересечение сечений.

Элементы перечисления **swThinWallType\_e**:

- **swThinWallOneDirection** – толщина тонкостенного тела в первом направлении;
- **swThinWallOppDirection** – толщина тонкостенного тела во втором направлении;
- **swThinWallMidPlane** – одинаковая толщина тонкостенного тела в двух направлениях;
- **swThinWallTwoDirection** – разная толщина тонкостенного тела в двух направлениях.

Элементы перечисления **swGuideCurveInfluence\_e**:

- **swGuideCurveInfluenceNextGuide** – до следующей направляющей кривой;
- **swGuideCurveInfluenceNextSharp** – до следующего острого угла сечения;
- **swGuideCurveInfluenceNextEdge** – до следующего сегмента сечения;
- **swGuideCurveInfluenceNextGlobal** – влияние на все сечение.

Создание бобышки заметанием. Метод класса **FeatureManager: Function InsertProtrusionSwept3 (ByVal Propagate As Boolean, ByVal Alignment As Boolean, ByVal TwistCtrlOption As Short, ByVal KeepTangency As Boolean, ByVal BAdvancedSmoothing As Boolean, ByVal StartMatchingType As Short, ByVal EndMatchingType As Short, ByVal IsThinBody As Boolean, ByVal Thickness1 As Double, ByVal Thickness2 As Double, ByVal ThinType As Short, ByVal PathAlign As Short, ByVal Merge As Boolean, ByVal UseFeatScope As Boolean, ByVal UseAutoSelect As Boolean, ByVal TwistAngle As Double, ByVal BMergeSmoothFaces As Boolean) As Feature**. Аргументы:

- если аргумент **Propagate** равен **True**, то бобышка будет распространена вдоль следующей кромки, касательной к выбранной кромке как пути;
- если аргумент **Alignment** равен **True**, то бобышка может выходить за пределы существующих тел, с которыми она пересекается и на которые она воздействует;
- **TwistCtrlOption** – опция контроля скручивания – значение из перечисления **swTwistControlType\_e**;
- если аргумент **KeepTangency** равен **True**, то, когда сечение имеет касательные сегменты, получаемые поверхности также будут касательными;
- если аргумент **BAdvancedSmoothing** равен **True**, то, когда сечение имеет круговые или эллиптические сегменты, будет осуществляться их аппроксимация для лучшего сглаживания;
- **StartMatchingType** – тип касания направляющей кривой с исходным сечением;
- **EndMatchingType** – тип касания направляющей кривой с конечным сечением;
- **IsThinBody** – см. создать бобышку по сечениям;
- **Thickness1** – см. создать бобышку по сечениям;
- **Thickness2** – см. создать бобышку по сечениям;
- **ThinType** – см. создать бобышку по сечениям;

- **PathAlign** – тип выравнивания сечения, которое имеет смысл, когда аргумент **TwistCtrlOption** равен **swTwistControlFollowPath**;
- **Merge** – см. создать бобышку по сечениям;
- **UseFeatScope** – см. создать бобышку по сечениям;
- **UseAutoSelect** – см. создать бобышку по сечениям;
- **TwistAngle** – угол скручивания конечного сечения, если аргумент **TwistCtrlOption** равен **swTwistControlConstantTwistAlongPath**;
- если аргумент **BMergeSmoothFaces** равен **True**, то гладкие грани будут объединены.

Для выделения необходимо использовать метод **SelectByID2**; сечению присваивается номер 1, направляющей кривой – номер 2, пути – номер 4. Аргумент **PathAlign** может принимать только следующие значения: 0 – выравнивание сечения относительно нормали к пути; 2 – выравнивание сечения относительно заданного направления; 3 – к пути в виде кромки грани добавляются кромки смежных граней, выравнивание сечения относительно нормали к пути. Аргументы **StartMatchingType** и **EndMatchingType** могут принимать только следующие значения: 0 – направление касательной к направляющей кривой не задано, 2 – направление касательной к направляющей кривой по направлению касательной к пути.

Элементы перечисления **swTwistControlType\_e**:

- **swTwistControlFollowPath** – сечение перемещается вдоль касательной к пути;
- **swTwistControlKeepNormalConstant** – выравнивание сечения относительно нормали к исходному сечению;
- **swTwistControlFollowPathFirstGuideCurve** – выравнивание сечения относительно нормали к пути, профилирование в соответствии с одной направляющей кривой;
- **swTwistControlFollowFirstSecondGuideCurves** – выравнивание сечения относительно нормали к пути, профилирование в соответствии с двумя направляющими кривыми;
- **swTwistControlConstantTwistAlongPath** – направление нормали к сечению постоянно относительно касательной к пути, равномерное скручивание сечения относительно этой касательной на заданный угол.

Создание бобышки поворотом. Метод класса **FeatureManager: Function FeatureRevolve2(ByVal SingleDir As Boolean, ByVal IsSolid As Boolean, ByVal IsThin As Boolean, ByVal IsCut As Boolean, ByVal ReverseDir As Boolean, ByVal BothDirectionUpToSameEntity As Boolean, ByVal Dir1Type As Integer, ByVal Dir2Type As Integer, ByVal Dir1Angle As Double, ByVal Dir2Angle As Double, ByVal OffsetReverse1 As Boolean, ByVal OffsetReverse2 As Boolean, ByVal OffsetDistance1 As Double, ByVal OffsetDistance2 As Double, ByVal ThinType As Integer, ByVal ThinThickness1 As Double, ByVal ThinThickness2 As Double, ByVal Merge As Boolean, ByVal UseFeatScope As Boolean, ByVal UseAutoSelect As Boolean) As Feature**. Аргументы:

- если аргумент **SingleDir** равен **True**, то бобышка будет создана в одном (первом) направлении;

- если аргумент **IsSolid** равен **True**, то будет создано твердое тело;
- если аргумент **IsThin** равен **True**, то будет создано тонкостенное тело;
- если аргумент **IsCut** равен **True**, то будет создан вырез поворотом;
- если аргумент **ReverseDir** равен **True**, то будет осуществлено реверсирование направления поворота;
- если аргумент **BothDirectionUpToSameEntity** равен **True**, то бобышка будет создана в обоих направлениях, причем аргумент **SingleDir** должен быть равен **False**, а каждый из аргументов **Dir1Type** и **Dir2Type** должен быть равен или **swEndCondUpToVertex**, или **swEndCondUpToSurface**, или **swEndCondOffsetFromSurface**;
- **Dir1Type** – тип ограничения направления 1 – значение из перечисления **swEndConditions\_e**;
- **Dir2Type** – тип ограничения направления 2 – значение из перечисления **swEndConditions\_e**;
- **Dir1Angle** – угол поворота в радианах в направлении 1;
- **Dir2Angle** – поворота в радианах в направлении 2;
- **OffsetReverse1** – см. создать бобышку вытягиванием;
- **OffsetReverse2** – см. создать бобышку вытягиванием;
- **OffsetDistance1** – смещение в направлении 1, если аргумент **Dir1Type** равен **swEndCondOffsetFromSurface**;
- **OffsetDistance2** – смещение в направлении 2, если аргумент **Dir2Type** равен **swEndCondOffsetFromSurface**;
- **ThinType** – см. создать бобышку по сечениям;
- **ThinThickness1** – см. создать бобышку по сечениям;
- **ThinThickness2** – см. создать бобышку по сечениям;
- **Merge** – см. создать бобышку вытягиванием;
- **UseFeatScope** – см. создать бобышку вытягиванием;
- **UseAutoSelect** – см. создать бобышку вытягиванием.

Выделение должно осуществляться методом **SelectByID2**, причем оси присваивается номер 16. Аргументы **Dir1Type** и **Dir2Type** могут принимать только следующие значения: **swEndCondBlind**, **swEndCondUpToVertex**, **swEndCondUpToSurface**, **swEndCondOffsetFromSurface** и **swEndCondMidPlane**.

Создание выреза вытягиванием. Метод класса **FeatureManager**: **Function FeatureCut3(ByVal Sd As Boolean, ByVal Flip As Boolean, ByVal Dir As Boolean, ByVal T1 As Integer, ByVal T2 As Integer, ByVal D1 As Double, ByVal D2 As Double, ByVal Dchk1 As Boolean, ByVal Dchk2 As Boolean, ByVal Ddir1 As Boolean, ByVal Ddir2 As Boolean, ByVal Dang1 As Double, ByVal Dang2 As Double, ByVal OffsetReverse1 As Boolean, ByVal OffsetReverse2 As Boolean, ByVal TranslateSurface1 As Boolean, ByVal TranslateSurface2 As Boolean, ByVal NormalCut As Boolean, ByVal UseFeatScope As Boolean, ByVal UseAutoSelect As Boolean, ByVal AssemblyFeatureScope As Boolean, ByVal AutoSelectComponents As Boolean, ByVal PropagateFeatureToParts As Boolean, ByVal T0 As Integer, ByVal StartOffset As Double, ByVal FlipStartOffset As Boolean) As Feature**. Аргументы: см. создать бобышку вытягиванием. Аргумент **NormalCut** равен

**True**, если вытянутый вырез создается перпендикулярно листу, **False** – если тело не является листом. Если аргумент **AssemblyFeatureScope** равен **True**, то вытянутый вырез может затрагивать только выделенные компоненты сборки, если **False** – то вытянутый вырез может затрагивать все компоненты сборки, как существующие, так и добавленные потом. Если аргумент **AutoSelectComponents** равен **True**, то компоненты сборки, пересекающие бобышку, будут выделены автоматически, если **False**, то эти компоненты должны быть выделены. Если аргумент **PropagateFeatureToParts** равен **True**, то вытянутый вырез будет передан в соответствующие файлы компонентов сборки.

Создание выреза по сечениям. Метод класса **FeatureManager: Function InsertCutBlend (ByVal Closed As Boolean, ByVal KeepTangency As Boolean, ByVal ForceNonRational As Boolean, ByVal TessToleranceFactor As Double, ByVal StartMatchingType As Short, ByVal EndMatchingType As Short, ByVal IsThinBody As Boolean, ByVal Thickness1 As Double, ByVal Thickness2 As Double, ByVal ThinType As Short, ByVal UseFeatScope As Boolean, ByVal UseAutoSelect As Boolean) As Feature**. Аргументы: см. создать бобышку по сечениям.

Создание выреза заметанием. Метод класса **FeatureManager: Function InsertCutSwept4 (ByVal Propagate As Boolean, ByVal Alignment As Boolean, ByVal TwistCtrlOption As Short, ByVal KeepTangency As Boolean, ByVal BAdvancedSmoothing As Boolean, ByVal StartMatchingType As Short, ByVal EndMatchingType As Short, ByVal IsThinBody As Boolean, ByVal Thickness1 As Double, ByVal Thickness2 As Double, ByVal ThinType As Short, ByVal PathAlign As Short, ByVal UseFeatScope As Boolean, ByVal UseAutoSelect As Boolean, ByVal TwistAngle As Double, ByVal BMergeSmoothFaces As Boolean, ByVal AssemblyFeatureScope As Boolean, ByVal AutoSelectComponents As Boolean, ByVal PropagateFeatureToParts As Boolean) As Feature**. Аргументы: см. создать бобышку заметанием.

Создание выреза поворотом. Метод класса **FeatureManager: Function FeatureRevolveCut2 (ByVal Angle As Double, ByVal ReverseDir As Boolean, ByVal Angle2 As Double, ByVal RevType As Integer, ByVal Options As Integer, ByVal UseFeatScope As Boolean, ByVal UseAutoSelect As Boolean, ByVal AssemblyFeatureScope As Boolean, ByVal AutoSelectComponents As Boolean, ByVal PropagateFeatureToParts As Boolean) As Feature**. Аргументы:

- Angle – угол поворота в радианах в направлении 1;
- если аргумент **ReverseDir** равен **True**, то направление поворота будет реверсировано;
- Angle2 – угол поворота в радианах в направлении 2;
- RevType – тип вращения – значение из перечисления **swRevolveType\_e**;
- Options – опции – значение **swAutoCloseSketch**, установка которого обуславливает выход из эскиза, если он активен;
- UseFeatScope – см. создать бобышку вытягиванием;
- UseAutoSelect – см. создать бобышку вытягиванием;
- AssemblyFeatureScope – см. создать вырез вытягиванием;
- AutoSelectComponents – см. создать вырез вытягиванием;



– **PropagateFeatureToParts** – см. создать вырез вытягиванием.

Создание закругления. Метод класса **FeatureManager**: **Function FeatureFillet(ByVal Options As Integer, ByVal R1 As Double, ByVal Ftyp As Integer, ByVal OverflowType As Integer, ByVal Radii As Object, ByVal SetBackDistances As Object, ByVal PointRadiusArray As Object ) As Object**. Аргументы:

- **Options** – опции – значение из перечисления **swFeatureFilletOptions\_e**;
- **R1** – радиус скругления, если аргумент **Ftyp** равен **swFeatureFilletType\_Simple**;
- **Ftyp** – тип скругления – значение из перечисления **swFeatureFilletType\_e**;
- **OverflowType** – тип ограничения скругления существующей кромкой – значение из перечисления **swFilletOverFlowType\_e**;
- **Radii** – массив, содержащий значения радиусов скругления в вершинах кромки, если аргумент **Ftyp** равен **swFeatureFilletType\_VariableRadius**, или значения радиусов скругления разных кромок;
- **SetBackDistances** – массив, содержащий расстояния от вершины вдоль кромки, определяющие положения точек, в которых задается уменьшающийся радиус скругления;
- **PointRadiusArray** – массив, содержащий радиусы скругления в заданных справочных точках кромки.

При выделении граней первой грани присваивается номер 2, второй грани присваивается номер 4. При создании полного скругления дополнительно к п. 1) центральной грани присваивается номер 512. Если аргумент **Ftyp** не равен **swFeatureFilletType\_Face**, то опции **swFeatureFilletUseHelpPoint**, **swFeatureFilletUseTangentHoldLine**, **swFeatureFilletCurvatureContinuous**, **swFeatureFilletConstantWidth**, **swFeatureFilletNoTrimNoAttached** не задаются. Если аргумент **Ftyp** равен **swFeatureFilletType\_VariableRadius**, то необходимо задать справочные точки.

Элементы перечисления **swFeatureFilletOptions\_e**:

- **swFeatureFilletPropagate** – распространение скругления на всю цепочку кромок, в которой соседние кромки являются касательными;
- **swFeatureFilletUniformRadius** – скругление одним и тем же радиусом разных кромок;
- **swFeatureFilletVarRadiusType** – скругление и прилегающая грань не являются касательными;
- **swFeatureFilletUseHelpPoint** – использование вспомогательной точки;
- **swFeatureFilletUseTangentHoldLine** – использование кривой сопряжения скругления и грани;
- **swFeatureFilletCornerType** – кромка между смежными скруглениями не скругляется;
- **swFeatureFilletAttachEdges** – присоединение кромок;
- **swFeatureFilletKeepFeatures** – сохранение элементов дерева построения, которые попадают в область действия скругления;
- **swFeatureFilletCurvatureContinuous** – непрерывная кривизна скругления граней;

- **swFeatureFilletConstantWidth** – постоянная ширина скругления;
- **swFeatureFilletNoTrimNoAttached** – после добавления переходного скругления грани не обрезаются и не соединяются;
- **swFeatureFilletReverseFace1Dir** – реверсирование нормали к грани 1;
- **swFeatureFilletReverseFace2Dir** – реверсирование нормали к грани 2;
- **swFeatureFilletPropagateFeatToParts** – скругление передается в соответствующие файлы компонентов сборки.

Элементы перечисления **swFeatureFilletType\_e**:

- **swFeatureFilletType\_Simple** – простое скругление;
- **swFeatureFilletType\_VariableRadius** – скругление переменного радиуса;
- **swFeatureFilletType\_Face** – добавление переходного скругления между парой граней;
- **swFeatureFilletType\_FullRound** – полное скругление.

Элементы перечисления **swFilletOverflowType\_e**:

- **swFilletOverflowType\_Default** – по умолчанию;
- **swFilletOverflowType\_KeepEdge** – если скругление ограничивается существующей кромкой, то скругление искажается вдоль кромки, а кромка не искажается;
- **swFilletOverflowType\_KeepSurface** – если скругление ограничивается существующей кромкой, то скругление не искажается, а искажается участок кромки, ограничивающий скругление.

Создание фаски. Метод класса **FeatureManager: Function InsertFeatureChamfer(ByVal Options As Integer, ByVal ChamferType As Integer, ByVal Width As Double, ByVal Angle As Double, ByVal OtherDist As Double, ByVal VertexChamDist1 As Double, ByVal VertexChamDist2 As Double, ByVal VertexChamDist3 As Double) As Feature**. Аргументы:

- **Options** – опции – значение из перечисления **swFeatureChamferOption\_e**;
- **ChamferType** – тип фаски – значение из перечисления **swChamferType\_e**;
- **Width** – ширина фаски, если аргумент **ChamferType** равен **swChamferAngleDistance**;
- **Angle** – угол, если аргумент **ChamferType** равен **swChamferAngleDistance**;
- **OtherDist** – другие расстояния, если аргумент **ChamferType** равен **swChamferDistanceDistance** или **swChamferVertex**;
- **VertexChamDist1** – расстояние на первой стороне, если аргумент **ChamferType** равен **swChamferDistanceDistance** или **swChamferVertex**;
- **VertexChamDist2** – расстояние на второй стороне, если аргумент **ChamferType** равен **swChamferDistanceDistance** или **swChamferVertex**;
- **VertexChamDist3** – расстояние на третьей стороне, если аргумент **ChamferType** равен **swChamferDistanceDistance** или **swChamferVertex**.

Элементы перечисления **swFeatureChamferOption\_e**:

- **swFeatureChamferFlipDirection** – реверсировать направление;
- **swFeatureChamferKeepFeature** – сохранить элементы дерева построения;
- **swFeatureChamferTangentPropagation** – распространить по касательной;
- **swFeatureChamferPropagateFeatToParts** – передать фаску в соответствующие файлы компонентов сборки.

Элементы перечисления **swChamferType\_e**:

- **swChamferAngleDistance** – угол и расстояние;
- **swChamferDistanceDistance** – расстояние и расстояние;
- **swChamferVertex** – вершина;
- **swChamferEqualDistance** – равные расстояния.

Создание линейного массива. Метод класса **FeatureManager: Function FeatureLinearPattern2(ByVal Num1 As Integer, ByVal Spacing1 As Double, ByVal Num2 As Integer, ByVal Spacing2 As Double, ByVal FlipDir1 As Boolean, ByVal FlipDir2 As Boolean, ByVal DName1 As String, ByVal DName2 As String, ByVal GeometryPattern As Boolean) As Feature**. Аргументы:

- Num1 – количество элементов в линейном массиве в направлении 1, включая исходный элемент;
- Spacing1 – расстояние между элементами в линейном массиве в направлении 1;
- Num2 – количество элементов в линейном массиве в направлении 2, включая исходный элемент;
- Spacing2 – расстояние между элементами в линейном массиве в направлении 2;
- если аргумент FlipDir1 равен **True**, то направление 1 будет реверсировано;
- если аргумент FlipDir2 равен **True**, то направление 2 будет реверсировано;
- DName1 – имя размера в направлении 1;
- DName2 – имя размера в направлении 2;
- если аргумент GeometryPattern равен **True**, то будет создан геометрический массив.

Если выделяются исходные элементы массива в виде элементов дерева построения, то им присваивается порядковый номер 4, а выделяемым направлениям – номера 1 и 2. Если выделяются исходные элементы массива в виде компонентов сборки, то им присваивается порядковый номер 1, а выделяемым направлениям – номера 2 и 4.

Создание кругового массива. Метод класса **FeatureManager: Function FeatureCircularPattern3(ByVal Number As Integer, ByVal Spacing As Double, ByVal FlipDirection As Boolean, ByVal DName As String, ByVal GeometryPattern As Boolean, ByVal EqualSpacing As Boolean) As Feature**. Аргументы:

- Number – количество элементов в круговом массиве, включая исходный элемент;
- Spacing – угловое (в радианах) расстояние между элементами в круговом массиве или общий угол (в радианах), если аргумент **EqualSpacing** равен **True**;
- если аргумент FlipDirection равен **True**, то направление будет реверсировано;
- DName – имя углового размера;
- если аргумент GeometryPattern равен **True**, то будет создан геометрический массив;
- если аргумент EqualSpacing равен **True**, то элементы в круговом массиве будут распределены равномерно.

При выделении исходных элементов в виде элементов дерева построения им необходимо присвоить порядковый номер 4, а при выделении исходных элементов в виде компонентов сборки им присваивается порядковый номер 1,

при этом при выделении оси ей присваивается порядковый номер 2.

Создание массива, управляемого кривой. Метод класса **ModelDoc2:SubFeatureCurvePattern**(**ByVal** Num1 **As Integer**, **ByVal** Spacing1 **As Double**, **ByVal** Num2 **As Integer**, **ByVal** Spacing2 **As Double**, **ByVal** FlipDir1 **As Boolean**, **ByVal** FlipDir2 **As Boolean**, **ByVal** EqualSpacing1 **As Boolean**, **ByVal** EqualSpacing2 **As Boolean**, **ByVal** UseCentroid **As Boolean**, **ByVal** AlignToSeed **As Boolean**, **ByVal** OffsetCurve **As Boolean**, **ByVal** PatternSeedOnly **As Boolean**). Аргументы:

- Num1 – количество элементов в направлении 1, включая исходный;
- Spacing1 – шаг в направлении 1;
- Num2 – количество элементов в направлении 2;
- Spacing2 – шаг в направлении 2;
- если аргумент FlipDir1 равен **True**, то направление 1 будет реверсировано;
- если аргумент FlipDir2 равен **True**, то направление 2 будет реверсировано;
- если аргумент EqualSpacing1 равен **True**, то в направлении 1 шаг будет одинаковый;
- если аргумент EqualSpacing2 равен **True**, то в направлении 2 шаг будет одинаковый;
- если аргумент UseCentroid равен **True**, то в качестве справочной точки будет использован центр;
- если аргумент AlignToSeed равен **True**, то элементы будут выравниваться относительно исходного элемента, если **False**, то элементы будут выравниваться относительно касательной к кривой;
- если аргумент OffsetCurve равен **True**, то расстояние от нормали к кривой до каждого элемента будет постоянным, если **False**, то расстояние от точки кривой до элемента будет постоянным;
- если аргумент PatternSeedOnly равен **True**, то в направлении 2 будет скопирован только исходный элемент.

## 2 Основы программирования в среде САПР NX

Программа для NX может быть создана на языке C++ в среде Visual Studio. Данная программа оформляется как библиотечный файл с расширением dll запускается в среде NX следующей командой: «File» → «Execute» → «NX Open».

Для разработки собственных инструментов и приложений система NX предлагает два основных модуля.

Модуль NX/Open API обеспечивает прямой программный интерфейс к системе NX, позволяя пользователю создавать приложения на наиболее известных на сегодняшний день языках программирования Visual Basic, C и Java. Модуль обеспечивает свободно расширяемую модель данных. Можно определить собственный объект на базе стандартных объектов NX. Этот объект будет изображаться, управляться и храниться в базе данных NX, как и любой стандартный объект. Можно написать приложение, которое будет работать как полностью интегрированная «внутренняя» функция NX или как самостоятельно выполняемая программа.

Модуль NX/Open++ содержит объектно-ориентированный интерфейс к NX. Написанный на языке C++, интерфейс использует все преимущества объ-

ектно-ориентированного программирования: наследование свойств, инкапсуляцию и полиморфизм. Модуль обеспечивает полный доступ к иерархической модели классов, разрешая пользователю перегружать операции, выводить свои собственные классы и создавать целиком новые объекты в NX. NX/Open++ полностью совместим с модулем NX/Open API и обеспечивает легкое решение сложных программных задач.

В стандартный шаблон программы на основе модуля NX/Open++ входит процедура CreateModel, которую пользователь должен написать в соответствии со своей задачей:

```
Session *psession=NULL;
Update *pupdate=NULL;
Part *ppart=NULL;
Session::UndoMarkId undomarkid;
UI *pui=NULL;
ListingWindow *plistingwindow=NULL;
ExpressionCollection *pexpressioncollection=NULL;
ExpressionCollection::iterator i_expressioncollection;
BaseFeatureCollection *pbasefeaturecollection=NULL;
BaseFeatureCollection::iterator i_basefeaturecollection;
FeatureCollection *pfeaturecollection=NULL;
FeatureCollection::iterator i_featurecollection;
DatumCollection *pdatumcollection=NULL;
DatumCollection::iterator i_datumcollection;
SketchCollection *psketchcollection=NULL;
SketchCollection::iterator i_sketchcollection;
SectionCollection *psectioncollection=NULL;
SectionCollection::iterator i_sectioncollection;
BodyCollection *pbodycollection=NULL;
BodyCollection::iterator i_bodycollection;
```

```
void CreateModel()
{
    psession=Session::GetSession();
    pupdate=psession->UpdateManager();
    pui=UI::GetUI();
    plistingwindow=psession->ListingWindow();
    plistingwindow->Open();
    ppart=psession->Parts()->Work();
    pexpressioncollection=ppart->Expressions();
    psketchcollection=ppart->Sketches();
    psectioncollection=ppart->Sections();
    pfeaturecollection=ppart->Features();
    pbasefeaturecollection=ppart->BaseFeatures();
    pdatumcollection=ppart->Datums();
```

```

    pbodycollection=ppart->Bodies();
}

#define UF_CALL(X)(report_error(__FILE__,__LINE__,__X,(X)))
static int report_error( char *file,int line,char *call,int irc)
{
    if(irc)
    {
        char err[133],
            msg[133];
        sprintf(msg,"*** ERROR code %d at line %d in %s:\n+++ ",irc,line,file);
        UF_get_fail_message(irc,err);
        UF_print_syslog(msg,FALSE);
        UF_print_syslog(err,FALSE);
        UF_print_syslog("\n",FALSE);
        UF_print_syslog(call,FALSE);
        UF_print_syslog(";\n",FALSE);
        if(!UF_UI_open_listing_window())
        {
            UF_UI_write_listing_window(msg);
            UF_UI_write_listing_window(err);
            UF_UI_write_listing_window("\n");
            UF_UI_write_listing_window(call);
            UF_UI_write_listing_window(";\n");
        }
    }
    return irc;
}

extern DllExport void ufusr(char *parm,int *returnCode,int rlen)
{
    if(UF_CALL(UF_initialize()))
    {
        return;
    }

    CreateModel();/процедура пользователя

    UF_CALL(UF_terminate());
}

extern int ufusr_ask_unload(void)
{
    return UF_UNLOAD_IMMEDIATELY;
}

```

### 3 Примерный перечень вариантов

Примерный перечень деталей для 3D-моделирования:

- винты по ГОСТ 10342;
- винты по ГОСТ 10338;
- винты по ГОСТ 10341;
- винты по ГОСТ 10336;
- винты по ГОСТ 1488;
- винты по ГОСТ 11738;
- винты по ГОСТ 17475;
- винты по ГОСТ 17474;
- винты по ГОСТ 17473;
- винты по ГОСТ 1491;
- болты для отверстия из-под развертки по ГОСТ 7817;
- болты по ГОСТ 15163;
- болты к станочным пазам по ГОСТ 12201;
- болты по ГОСТ 7805;
- болты к станочным пазам по ГОСТ 13152;
- болты по ГОСТ 7798;
- втулки: фаски, закругления, канавки для выхода шлифовального круга по ГОСТ 8820;
- валы: фаски, закругления, шпоночные пазы;
- валы: фаски, закругления, центровые отверстия по ГОСТ 14034;
- валы: фаски, закругления, канавки для выхода шлифовального круга по ГОСТ 8820;
- эвольвентные прямозубые зубчатые колеса;
- эвольвентные косозубые зубчатые колеса.

### 4 Структура и оформление курсовой работы

Пояснительная записка должна содержать следующие обязательные элементы: структура дерева конструирования; описание класса; разработка программы; тестирование программы; заключение; список использованной литературы; приложения. Пояснительная записка оформляется в соответствии с ГОСТ 2.105 *Общие требования к текстовым документам*.

Графическая часть должна содержать чертеж детали; схемы алгоритмов двух методов класса; диаграммы (классов, состояний и т. п.).

Введение должно быть кратким (не более двух страниц). Во введении должны быть отражены цель и задачи.

Технические параметры и исходные данные берутся из ГОСТа или задаются в задании. Необходимо предусмотреть минимум два варианта исполнения детали.

Структура дерева конструирования должна отражать иерархическую последовательность действий по созданию 3D-модели. Обязательным является де-

тальное описание ассоциативных связей между элементами 3D-модели. Дерево конструирования будет определять во многом структуру программного кода. В данном разделе можно привести макрокоманды по созданию 3D-модели, генерируемые САПР в автоматическом режиме.

Описание класса должно содержать перечень полей и методов, необходимых для объектно-ориентированного представления 3D-модели. Структура класса должна соответствовать исходным данным и параметрам, а также базироваться на структуре дерева построения.

В разделе разработки программы необходимо описать структурную схему программы, интерфейс, схемы алгоритмов (по ГОСТ 19.701). Также приводится обоснование выбора языка программирования и среды. В качестве интерфейса рекомендуется использовать диалоговые окна, поддерживаемые большинством современных САПР. Необходимо дать ссылки на приложение, в котором приведен программный код.

Тестирование программы состоит в выявлении и устранении ошибок. Тесты необходимо подобрать так, чтобы охватить возможные варианты работы программы. Приводятся необходимые скриншоты.

В заключении кратко излагают основные результаты работы.

Список использованной литературы оформляется в соответствии с ГОСТ 7.1 *Библиографическая запись. Библиографическое описание. Общие требования и правила оформления.*

Приложения должны содержать, по крайней мере, программный код.

## Список литературы

- 1 **Бутко, А. О.** Основы моделирования в САПР NX: учеб. пособие / А. О. Бутко, В. А. Прудников, Г. А. Цырков. – 2-е изд. – М.: ИНФРА-М, 2018. – 199 с.
- 2 **Берлинер, Э. М.** САПР конструктора машиностроителя: учебник / Э. М. Берлинер, О. В. Таратынов. – М.: ФОРУМ; ИНФРА-М, 2015. – 288 с.
- 3 **Культин, Н. Б.** С/С++ в задачах и примерах / Н. Б. Культин. – 3-е изд., доп. и испр. – СПб. : БХВ-Петербург, 2019. – 272 с.
- 4 **Лустенков, М. Е.** Детали машин: учеб. пособие / М. Е. Лустенков. – Могилев: Бел.-Рос. ун-т, 2018. – 240 с.
- 5 **Лустенков, М. Е.** Детали машин : учеб. пособие / М. Е. Лустенков. – 2-е изд., перераб. и доп. – Могилев : Бел.-Рос. ун-т, 2020. – 258 с.
- 6 **Жуков, В. А.** Детали машин и основы конструирования. Основы расчета и проектирования соединений и передач : учеб. пособие / В. А. Жуков. – 2-е изд. – М. : ИНФРА-М, 2021. – 416 с.