**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ**   
**РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ**

**УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «МОСКОВСКИЙ**

**ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**



Расчетно-пояснительная записка

к курсовому проекту по  
дисциплине: «Программирование в САПР»

Вариант № 12

Группа:   
221-326

Выполнил:   
Колбешин Г.С.

Преподаватель:   
Джунковский А.В.

Дата выполнения:  
19.12.2023

**Аннотация**

Этот проект представляет собой SDI MFC приложение на C++, предназначенное для создания параметрических сборок компенсирующих упругих муфт в приложении КОМПАС 3D. Пользователь может выбирать различные исполнения и муфты с определенными крутящими моментами, а затем строить выбранную муфту в приложении КОМПАС 3D.

## *Особенности*

* **Интерфейс SDI MFC**: Проект построен с использованием SDI MFC для удобного взаимодействия с пользователем.
* **Дерево конфигурации (CMyTreeView)**: Реализовано дерево, позволяющее пользователю выбирать конкретные исполнения и параметры муфт.
* **Диалоговое окно первого исполнения (CNewDialog)**: Диалог для выбора параметров первого исполнения.
* **Диалоговое окно второго исполнения (CNewDialog2)**: Диалог для выбора параметров второго исполнения.
* **Интеграция с КОМПАС 3D**: Проект взаимодействует с API КОМПАС 3D для построения выбранной муфты в приложении.

## *Использование*

1. **Выбор исполнения**: В дереве конфигурации выберите нужное исполнение, кликнув по соответствующему узлу.
2. **Выбор муфты**: Внутри исполнения выберите муфту с определенным крутящим моментом, используя соответствующие диалоговые окна.
3. **Построение**: Нажмите кнопку "Построить", чтобы построить выбранную муфту в КОМПАС 3D.

## *Требования к среде разработки*

* **Visual Studio**: Проект разработан с использованием Visual Studio.
* **КОМПАС 3D API**: Необходима установленная библиотека или API КОМПАС 3D для взаимодействия с приложением.

## *Структура проекта*

* **MainApp (kursach)**: Исходный код основного приложения.
* **CMyTreeView**: Реализация дерева конфигурации.
* **CNewDialog**: Диалоговое окно первого исполнения.
* **CNewDialog2**: Диалоговое окно второго исполнения.
* **Resources**: Ресурсы приложения.
* **kursachDoc**: Документы проекта.
* **kursachView**: Представления проекта
* **MainFrm**: Главное окно приложения.

*Ознакомиться с проектом:*

**Github**: https://github.com/SenyashaGo/parametric-assembly-of-the-compensating-elastic-coupling

**Оглавление**

Теоретическая часть4-8

* 1. Муфты 4-7
  2. API КОМПАС-3D 8

Практическая часть9-56

2.1 Подключение к API Kompas 3D9

2.2 Анализ сборки10-12

2.3 Создание параметров размеров деталей и сборки (1 исполнение)13

2.4 Создание звездочки (1 исполнение)14-18

2.5 Создание полумуфты (1 исполнение)19-28

2.6 Создание сборки (1 исполнение)29-30

2.7 Создание параметров размеров деталей и сборки(2 исполнение)31-33

2.8 Создание звездочки (2 исполнение)34-40

2.9 Создание полумуфты (2 исполнение)41-54

2.10 Создание сборки (2 исполнение)55-56

Заключение57

Список литературы58

Приложения59-61

Теоретическая часть

* 1. **Муфты**

Большинство машин и технологических систем состоит из отдельных узлов, механизмов и агрегатов, цепочкой передающих энергию от двигателя к исполнительным органам. Для обеспечения кинематической и силовой связи между отдельными элементами этой цепочки (узлами, механизмами и агрегатами) используют специальные соединительные устройства – муфты.



Рисунок 1 — Виды муфт

Муфтой называют устройство для соединения концов валов или валов со свободно установленными на них деталями (зубчатыми колесами, шкивами, звездочками и т. п.). Муфты передают вращающий момент без изменения его величины и направления.   
Некоторые типы муфт при передаче вращающего момента способны выполнять дополнительные функции – поглощать вибрации, удары и толчки, предохранять машину от перегрузок, компенсировать несоосность и перекосы соединяемых валов, в определенных режимах автоматически разъединять и соединять отдельные механизмы и агрегаты машины без остановки двигателя.

*Классификация муфт*

Многообразие требований, предъявляемых к муфтам, и различные условия их работы обусловили создание большого количества конструкций муфт. В зависимости от функционального назначения и особенностям конструкции муфты классифицируются по управляемости и по степени снижения динамических нагрузок, а также по основным конструктивным признакам.

По управляемости муфты подразделяют на неуправляемые, управляемые и самоуправляемые.

*Неуправляемые* (нерасцепляемые) муфты осуществляют постоянное соединение валов между собой. К этому типу относят глухие, жесткие и упругие компенсирующие муфты. Некоторые типы неуправляемых муфт способны поглощать динамические нагрузки на агрегаты и узлы машины при передаче вращающего момента, но разъединять или соединять эти элементы конструкции машины во время ее работы они не могут;

*Управляемые* (сцепные) муфты допускают во время работы соединение (сцепление) и разъединение (расцепление) валов посредством механизма управления. Управление работой таких муфт осуществляется с участием человека (иногда – специализированной компьютерной программы), т. е. процесс управления не является автоматическим, а зависит от внешнего управляющего фактора, а не от условий, в которых работает муфта или соединяемые элементы машины.

*Самоуправляемые* муфты способны автоматически соединять или разъединять валы при изменении заданного режима работы машины. Такие муфты, благодаря своей конструкции, могут самостоятельно (без внешнего управления) осуществлять сцепление или расцепление отдельных элементов машины в случае перегрузки или при достижении определенного режима работы.

К самоуправляемым относят обгонные муфты, центробежные муфты и предохранительные муфты.

*Обгонные* муфты служат для передачи вращающего момента только в одном направлении, разъединяя валы, если их относительное вращение изменяется на противоположное.

*Центробежные* муфты срабатывают при достижении определенной частоты вращения соединяемых валов.

*Предохранительные* муфты разъединяют валы при превышении допустимой величины передаваемого крутящего момента. Самоуправляемые предохранительные муфты могут выполняться с разрушающимся соединительным элементом или с автоматическим разъединителем.

По степени снижения динамических нагрузок муфты подразделяют на жесткие и упругие.

*Жесткие* муфты не сглаживают динамические нагрузки, возникающие при передаче вращающего момента – удары, толчки, вибрации и т. п. Такие муфты обеспечивают постоянную кинематическую и динамическую связь между валами.

*Упругие* муфты способны сглаживать вибрации, толчки и удары между валами благодаря наличию в их конструкции упругих элементов – пружин, резиновых втулок или прокладок и т. п.

По конструктивным особенностям различают следующие основные типы механических муфт:

*Глухие* (жесткие) муфты:

* втулочные;
* фланцевые;
* продольно-свёртные.

*Компенсирующие жесткие* муфты (компенсирующие радиальные, осевые и угловые смещения валов):

* шарнирные муфты (компенсируют угловое смещение валов до 45°);
* зубчатые;
* цепные.

*Компенсирующие упругие* муфты (компенсирующие динамические нагрузки):

* муфты с торообразной или лепестковой оболочкой;
* втулочно-пальцевые;
* муфты со звёздочкой.

*Сцепные* муфты (соединяющие или разъединяющие валы посредством специального управляемого механизма):

* муфты кулачково-дисковые;
* кулачковые муфты;
* фрикционные;
* центробежные.

*Самоуправляемые* (автоматические) муфты:

* обгонные муфты (передающие вращение только в одном направлении);
* центробежные (срабатывающие при достижении определенной частоты вращения);
* предохранительные муфты (ограничивающие передаваемый крутящий момент).

Перечисленные типы муфт относятся к механическим устройствам. Кроме механических муфт в машинах и механизмах нередко применяются муфты с гидравлическим, магнитным или электромагнитным управлением.

Механизмы имеют широкую сферу применения и встречаются в большинстве устройств. Например, механические узлы служат для решения следующих задач:

* Передача энергии от одного вала к другому, которые располагаются на единой оси или под углом друг к другу.
* Подключение или отключение ведомого вала.
* Компенсация возникающих вибраций и ударных нагрузок.
* Защита привода от перегрузок во время эксплуатации.

В зависимости от этого механизмы могут устанавливаться на оборудовании для пищевой и химической отрасли, текстильных и печатных станках, компрессорах, насосах, турбинах, экструдерах, упаковочных машинах и испытательных стендах.

* 1. **API КОМПАС-3D**

API КОМПАС-3D представляет собой эффективный программный интерфейс, спроектированный для взаимодействия внешних приложений с программным обеспечением САПР "КОМПАС-3D". Этот инструмент открывает широкие возможности для разработчиков, позволяя интегрировать и автоматизировать различные аспекты проектирования, управления данными и взаимодействия с геометрическими моделями в рамках "КОМПАС-3D".

Используя API КОМПАС-3D, разработчики могут создавать скрипты и приложения, направленные на автоматизацию повторяющихся задач проектирования. Это может включать в себя генерацию деталей, сборок, формирование отчетов и многое другое. Гибкость и функциональность API КОМПАС-3D позволяют индивидуализировать процессы с учетом уникальных потребностей пользователя.

* 1. **Практическая часть.**

**2.1. Подключение к API Kompas 3D.**

Перед началом построения деталей необходимо подключить интерфейс компаса, а также сделать Компас 3D видимым.

CComPtr<IUnknown> pKompasAppUnk = nullptr;

if (!pKompasApp5)

{

// Получаем CLSID для Компас

CLSID InvAppClsid;

HRESULT hRes = CLSIDFromProgID(L"Kompas.Application.5", &InvAppClsid);

if (FAILED(hRes)) {

pKompasApp5 = nullptr;

return;

}

// Проверяем есть ли запущенный экземпляр Компас

//если есть получаем IUnknown

hRes = ::GetActiveObject(InvAppClsid, NULL, &pKompasAppUnk);

if (FAILED(hRes)) {

// Приходится запускать Компас самим так как работающего нет

// Также получаем IUnknown для только что запущенного приложения Компас

TRACE(L"Could not get hold of an active Inventor, will start a new session\n");

hRes = CoCreateInstance(InvAppClsid, NULL, CLSCTX\_LOCAL\_SERVER, \_\_uuidof(IUnknown), (void\*\*)&pKompasAppUnk);

if (FAILED(hRes)) {

pKompasApp5 = nullptr;

return;

}

}

// Получаем интерфейс приложения Компас

hRes = pKompasAppUnk->QueryInterface(\_\_uuidof(KompasObject), (void\*\*)&pKompasApp5);

if (FAILED(hRes)) {

return;

}

}

// делаем Компас видимым

pKompasApp5->Visible = true;

Листинг 1 – подключение к Компасу

**2.2. Анализ сборки.**

Далее проанализируем документ с исполнениями муфт, определим из каких деталей состоит сборка. Сборка состоит из двух полумуфт и звездочки.

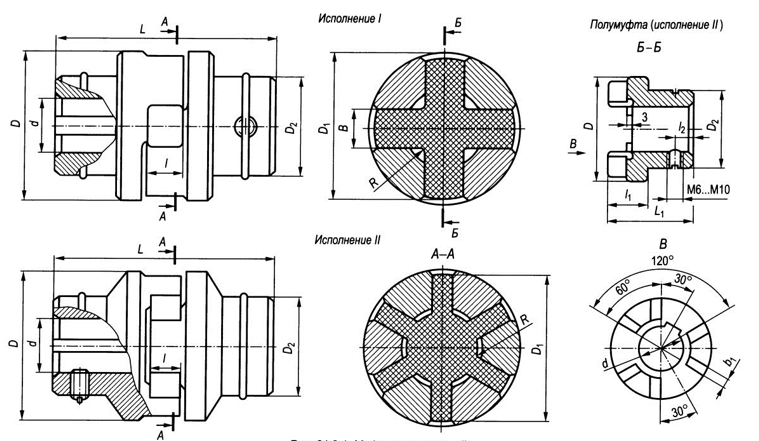


Рисунок 2 — Чертеж сборки

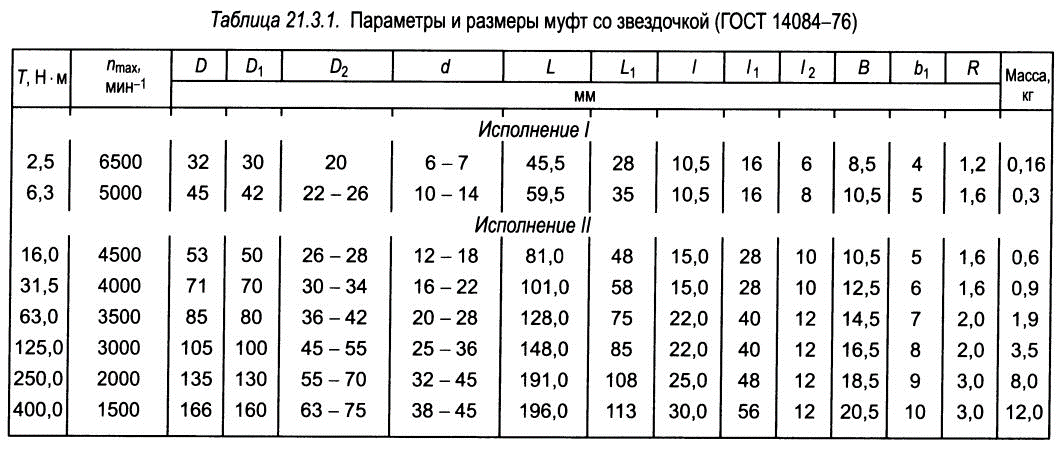


Рисунок 3 — Таблица размеров сборки

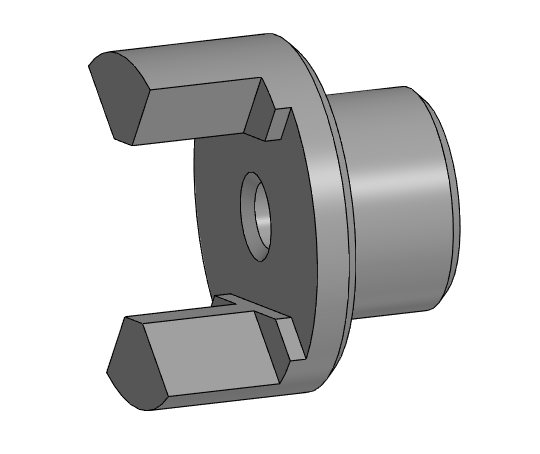


Рисунок 4 — Полумуфта исполнение 1

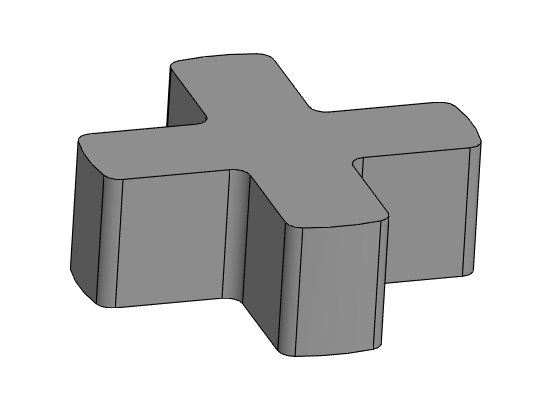


Рисунок 5 — Звездочка исполнение 1

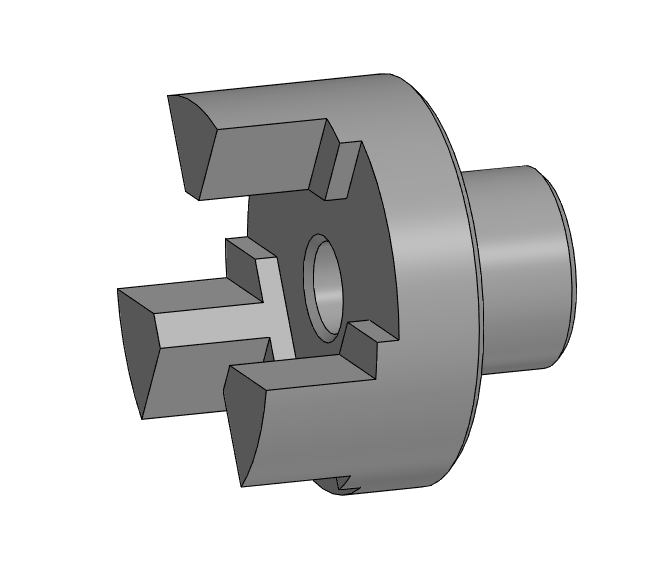


Рисунок 6 — Полумуфта исполнение 2

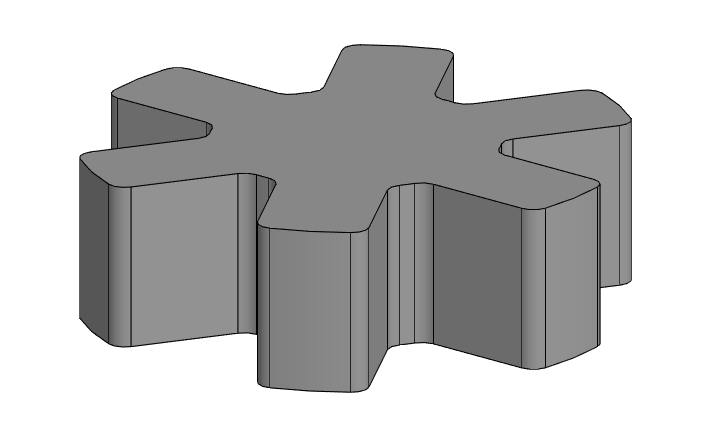


Рисунок 7 — Звездочка исполнение 2

**2.3. Создание параметров размеров деталей и сборки (1 исполнение).**

Выбор параметров будет осуществляться с помощью Combo Box.

void CNewDialog::OnCbnSelchangeCombo1()

{

// Получаем индекс выбранного элемента

int selectedIndex = execution.GetCurSel();

// Проверяем выбранный элемент и устанавливаем значения переменных соответственно

if (selectedIndex == 0) // Выбрано "1"

{

D = 32;

D1 = 30;

D2 = 20;

d = 6;

L = 45.5;

L1 = 28;

l = 10.5;

l1 = 16;

l2 = 6;

B = 8;

b1 = 4;

R = 1.2;

lDeep = 16;

B1 = 16;

}

else if (selectedIndex == 1) // Выбрано "2"

{

D = 45;

D1 = 42;

D2 = 22;

d = 10;

L = 59.5;

L1 = 35;

l = 10.5;

l1 = 16;

l2 = 8;

B = 10;

b1 = 5;

R = 1.6;

lDeep = 23;

B1 = 20;

}

// Обновляем данные на диалоговом окне

UpdateData(FALSE);

}

Листинг 2 – Создание параметров сборки

**2.4. Создание звездочки (1 исполнение).**

Для начала создадим эскиз самой звездочки, а затем выдавим его.

pDoc = pKompasApp5->Document3D();

pDoc->Create(false, true);

pPart = pDoc->GetPart(pTop\_Part);

ksEntityPtr pSketch = pPart->NewEntity(o3d\_sketch);

ksSketchDefinitionPtr pSketchDef = pSketch->GetDefinition();

pSketchDef->SetPlane(pPart->GetDefaultEntity(o3d\_planeXOY));

pSketch->Create();

ksDocument2DPtr p2DDoc = pSketchDef->BeginEdit();

UpdateData();

//создаем эскиз для звездочки муфты

double plus\_B = B;

double plus\_D1 = D1;

double plus\_R = R;

p2DDoc->ksLineSeg(-plus\_D1 / 2, 0, -plus\_D1 / 2, plus\_B / 2, 1);

p2DDoc->ksLineSeg(-plus\_D1 / 2, plus\_B / 2, -plus\_B / 2, plus\_B / 2, 1);

p2DDoc->ksLineSeg(-plus\_B / 2, plus\_B / 2, -plus\_B / 2, plus\_D1 / 2, 1);

p2DDoc->ksLineSeg(-plus\_B / 2, plus\_D1 / 2, plus\_B / 2, plus\_D1 / 2, 1);

p2DDoc->ksLineSeg(plus\_B / 2, plus\_D1 / 2, plus\_B / 2, plus\_B / 2, 1);

p2DDoc->ksLineSeg(plus\_B / 2, plus\_B / 2, plus\_D1 / 2, plus\_B / 2, 1);

p2DDoc->ksLineSeg(plus\_D1 / 2, plus\_B / 2, plus\_D1 / 2, 0, 1);

p2DDoc->ksLineSeg(plus\_D1 / 2, 0, plus\_D1 / 2, -plus\_B / 2, 1);

p2DDoc->ksLineSeg(plus\_D1 / 2, -plus\_B / 2, plus\_B / 2, -plus\_B / 2, 1);

p2DDoc->ksLineSeg(plus\_B / 2, -plus\_B / 2, plus\_B / 2, -plus\_D1 / 2, 1);

p2DDoc->ksLineSeg(plus\_B / 2, -plus\_D1 / 2, -plus\_B / 2, -plus\_D1 / 2, 1);

p2DDoc->ksLineSeg(-plus\_B / 2, -plus\_D1 / 2, -plus\_B / 2, -plus\_B / 2, 1);

p2DDoc->ksLineSeg(-plus\_B / 2, -plus\_B / 2, -plus\_D1 / 2, -plus\_B / 2, 1);

p2DDoc->ksLineSeg(-plus\_D1 / 2, -plus\_B / 2, -plus\_D1 / 2, 0, 1);

pSketchDef->EndEdit();

//выдавливание эскиза звездочки

ksEntityPtr pExtrudeDowel = pPart->NewEntity(o3d\_baseExtrusion);

ksBaseExtrusionDefinitionPtr pBaseDef = pExtrudeDowel->GetDefinition();

pBaseDef->directionType = dtNormal;

pBaseDef->SetSketch(pSketch);

pBaseDef->SetSideParam(true, etBlind, l, 0, false);

pExtrudeDowel->Create();

Листинг 3 – Создание эскиза звездочки и его выдавливания

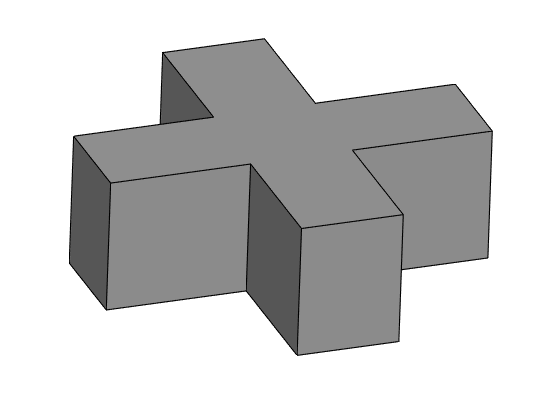


Рисунок 8 — Результат Листинга 3

Затем для скругления краев звездочки построим смещенную плоскость, на ней создадим эскиз окружности, в конце сделаем вырезание пересечением.

//смещенная плоскость для скругления краев

ksEntityPtr pPlane = pPart->NewEntity(o3d\_planeOffset);

ksPlaneOffsetDefinitionPtr pPlaneDef = pPlane->GetDefinition();

pPlaneDef->direction = true;

pPlaneDef->offset = l;

pPlaneDef->SetPlane(pPart->GetDefaultEntity(o3d\_planeXOY));

pPlane->Create();

//эскиз окружности для скругления краев

ksEntityPtr pSketch1 = pPart->NewEntity(o3d\_sketch);

pSketchDef = pSketch1->GetDefinition();

pSketchDef->SetPlane(pPlane);

pSketch1->Create();

p2DDoc = pSketchDef->BeginEdit();

p2DDoc->ksCircle(0, 0, plus\_D1 / 2, 1);

p2DDoc->ksCircle(0, 0, plus\_D1, 1);

pSketchDef->EndEdit();

//вырезание ПЕРЕСЕЧЕНИЕМ для скругления краев звездочки

ksEntityPtr pExtrude2 = pPart->NewEntity(o3d\_cutExtrusion);

ksCutExtrusionDefinitionPtr pCutDef = pExtrude2->GetDefinition();

pCutDef->directionType = dtNormal;

pCutDef->SetSketch(pSketch1);

pCutDef->SetSideParam(true, etThroughAll, 0, 0, false);

pExtrude2->Create();

Листинг 4 – Скругление краев звездочки.

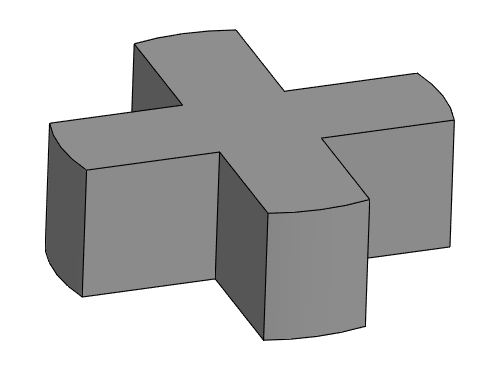


Рисунок 9 — Результат Листинга 4

//получившиеся скругленные края называем circle для дальнейшего применения соосности

ksEntityCollectionPtr flFaces = pPart->EntityCollection(o3d\_face);

for (int i = 0; i < flFaces->GetCount(); i++)

{

ksEntityPtr face = flFaces->GetByIndex(i);

ksFaceDefinitionPtr def = face->GetDefinition();

if (def->GetOwnerEntity() == pExtrude2)

{

face->Putname("circle");

face->Update();

}

}

Листинг 5 – Даем название грани.

//создаем скругление радиусом R, обзываем оставшиеся грани звезды

ksEntityCollectionPtr fledges1 = pPart->EntityCollection(o3d\_edge);

ksEntityPtr pFillet = pPart->NewEntity(o3d\_fillet);

ksFilletDefinitionPtr pFilletDef = pFillet->GetDefinition();

pFilletDef->radius = plus\_R;

ksEntityCollectionPtr fl1 = pFilletDef->array();

fl1->Clear();

int ng = 0;

int ng2 = 0;

for (int i = 0; i < fledges1->GetCount(); i++)

{

ksEntityPtr ed = fledges1->GetByIndex(i);

ksEdgeDefinitionPtr def = ed->GetDefinition();

if (def->IsStraight())

{

ksVertexDefinitionPtr p1 = def->GetVertex(true);

ksVertexDefinitionPtr p2 = def->GetVertex(false);

double x1, y1, z1, x2, y2, z2;

if (p1 && p2)

{

p1->GetPoint(&x1, &y1, &z1);

p2->GetPoint(&x2, &y2, &z2);

if (abs(z1) == l && abs(int(x1)) == B/2 && abs(int(y1))==B/2)

{//называем верхнюю грань

ksFaceDefinitionPtr f = def->GetAdjacentFace(true);

ksEntityPtr face = f->GetEntity();

face->Putname("starTOP");

face->Update();

}

if (abs(x1 - x2) < 0.0001 && abs(y1 - y2) < 0.0001)

{//скругления

fl1->Add(ed);

if (z1 == 0 && int(y1) == -B / 2 && (ceil(abs(x1)) == D1 / 2) && ng==0) {

//называем внутреннюю часть зуба

ksFaceDefinitionPtr f = def->GetAdjacentFace(true);

ksEntityPtr face = f->GetEntity();

face->Putname("star\_tor2");

face->Update();

ng++;

}

}

if (abs(x1 - x2) < 0.0001 && abs(y1 - y2) < 0.0001)

{

fl1->Add(ed);

if (z1 == 0 && int(y1) == B / 2 && (ceil(abs(x1)) == D1 / 2) && ng2 == 0) {

//называем другую внутреннюю часть зуба

ksFaceDefinitionPtr f = def->GetAdjacentFace(true);

ksEntityPtr face = f->GetEntity();

face->Putname("star\_tor");

face->Update();

ng2++;

}

}

}

}

}

pFillet->Create();

Листинг 6 – Создаем скругления, даем названия граням.

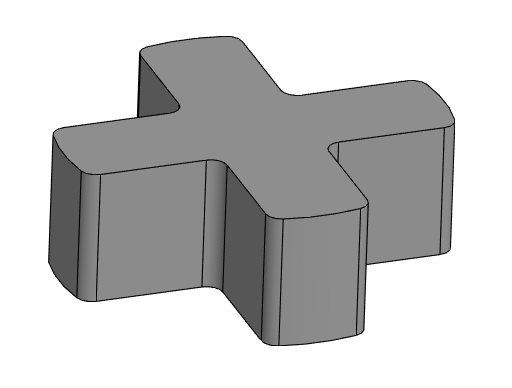


Рисунок 10 — Результат Листинга 6

//сохраняем звезду

pDoc->SaveAs("D:\\kursachTEST\\star1.m3d");

Листинг 7 – Сохраняем звезду.

**2.5. Создание полумуфты (1 исполнение).**

Прежде чем приступить к созданию полумуфты, необходимо создать новый документ, потом создать эскиз муфты, а затем выдавить ее вращением.

//создаем новый документ c муфтой 1 исполнения (2 зуба)

pDoc = pKompasApp5->Document3D();

pDoc->Create(false, true);

pPart = pDoc->GetPart(pTop\_Part);

pSketch = pPart->NewEntity(o3d\_sketch);

pSketchDef = pSketch->GetDefinition();

pSketchDef->SetPlane(pPart->GetDefaultEntity(o3d\_planeXOY));

pSketch->Create();

p2DDoc = pSketchDef->BeginEdit();

UpdateData();

double mufta\_1isp\_B1 = B1;

double mufta\_1isp\_d1 = D2;

double mufta\_1isp\_R = R;

double mufta\_1isp\_D = D;

double mufta\_1isp\_D1 = D1;

double mufta\_1isp\_D2 = D2;

double mufta\_1isp\_b = 2;

double mufta\_1isp\_d = d;

double mufta\_1isp\_L1 = L1;

double mufta\_1isp\_l = lDeep;

double mufta\_1isp\_B = B/2;

double mufta\_1isp\_L = L;

double mufta\_1isp\_l\_deep = lDeep;

//эскиз для муфты

p2DDoc->ksLineSeg(0, -mufta\_1isp\_B1 / 2, 0, -mufta\_1isp\_D / 2, 1);

p2DDoc->ksLineSeg(0, -mufta\_1isp\_D / 2, mufta\_1isp\_l, -mufta\_1isp\_D / 2, 1);

p2DDoc->ksLineSeg(mufta\_1isp\_l, -mufta\_1isp\_D / 2, mufta\_1isp\_l, -mufta\_1isp\_D2 / 2, 1);

p2DDoc->ksLineSeg(mufta\_1isp\_l, -mufta\_1isp\_D2 / 2, mufta\_1isp\_L1, -mufta\_1isp\_D2 / 2, 1);

p2DDoc->ksLineSeg(mufta\_1isp\_L1, -mufta\_1isp\_D2 / 2, mufta\_1isp\_L1, -mufta\_1isp\_d / 2, 1);

p2DDoc->ksLineSeg(mufta\_1isp\_L1, -mufta\_1isp\_d / 2, mufta\_1isp\_L1 - mufta\_1isp\_l\_deep, -mufta\_1isp\_d / 2, 1);

p2DDoc->ksLineSeg(mufta\_1isp\_L1 - mufta\_1isp\_l\_deep, -mufta\_1isp\_d / 2, mufta\_1isp\_L1 - mufta\_1isp\_l\_deep, -mufta\_1isp\_B1 / 2, 1);

p2DDoc->ksLineSeg(mufta\_1isp\_L1 - mufta\_1isp\_l\_deep, -mufta\_1isp\_B1 / 2, 0, -mufta\_1isp\_B1 / 2, 1);

p2DDoc->ksLineSeg(-10, 0, 10, 0, 3);

pSketchDef->EndEdit();

//выдавливаем вращением полученный эскиз

ksEntityPtr pRotate = pPart->NewEntity(o3d\_bossRotated);

ksBossRotatedDefinitionPtr pRotDef = pRotate->GetDefinition();

pRotDef->SetSketch(pSketch);

pRotDef->SetSideParam(TRUE, 360);

pRotate->Create();

Листинг 8 – Создание нового документа, эскиза муфты, выдавливания.

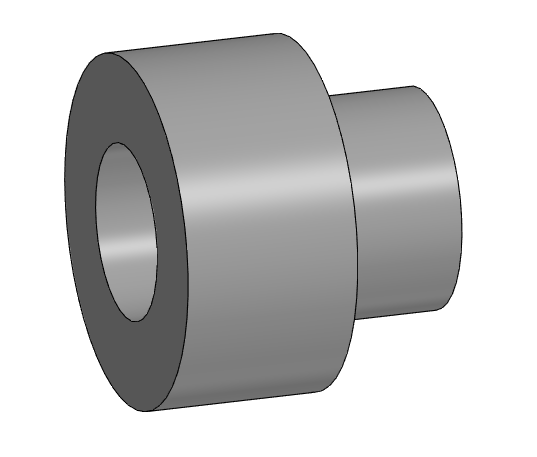


Рисунок 11 — Результат Листинга 8

//эскиз для образования зубов муфты

ksEntityPtr pSketch\_muft\_f = pPart->NewEntity(o3d\_sketch);

ksSketchDefinitionPtr pSketchDef\_muft\_f = pSketch\_muft\_f->GetDefinition();

pSketchDef\_muft\_f->SetPlane(pPart->GetDefaultEntity(o3d\_planeYOZ));

pSketch\_muft\_f->Create();

p2DDoc = pSketchDef\_muft\_f->BeginEdit();

UpdateData();

p2DDoc->ksLineSeg(-30, mufta\_1isp\_B1 / 2, 30, mufta\_1isp\_B1 / 2, 1);

p2DDoc->ksLineSeg(30, mufta\_1isp\_B1 / 2, 30, -mufta\_1isp\_B1 / 2, 1);

p2DDoc->ksLineSeg(30, -mufta\_1isp\_B1 / 2, -30, -mufta\_1isp\_B1 / 2, 1);

p2DDoc->ksLineSeg(-30, -mufta\_1isp\_B1 / 2, -30, mufta\_1isp\_B1 / 2, 1);

pSketchDef\_muft\_f->EndEdit();

//вырезаем выдавливанием ранее созданный эскиз

double glubina\_rect = mufta\_1isp\_L1 - mufta\_1isp\_l;

ksEntityPtr pExtrude3 = pPart->NewEntity(o3d\_cutExtrusion);

ksCutExtrusionDefinitionPtr pCutDef3 = pExtrude3->GetDefinition();

pCutDef3->directionType = dtNormal;

pCutDef3->SetSketch(pSketchDef\_muft\_f);

pCutDef3->SetSideParam(true, etBlind, glubina\_rect, 0, false);

pExtrude3->Create();

Листинг 9 – Создание зубов муфты.

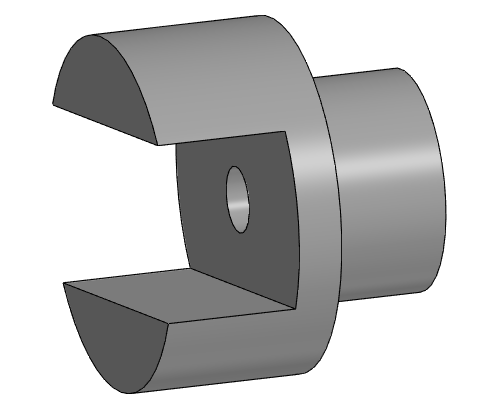


Рисунок 12 — Результат Листинга 9

//создаем эскиз для дальейшего выреза зуба муфты под 45 градусов

ksEntityPtr pSketch\_muft\_pl = pPart->NewEntity(o3d\_sketch);

ksSketchDefinitionPtr pSketchDef\_muft\_pl = pSketch\_muft\_pl->GetDefinition();

pSketchDef\_muft\_pl->SetPlane(pPart->GetDefaultEntity(o3d\_planeYOZ));

pSketch\_muft\_pl->Create();

p2DDoc = pSketchDef\_muft\_pl->BeginEdit();

UpdateData();

p2DDoc->ksLineSeg(0, sqrt(pow(mufta\_1isp\_B, 2) + pow(mufta\_1isp\_B, 2)), -100\*cos(45), 100 \* cos(45) + sqrt(pow(mufta\_1isp\_B, 2) + pow(mufta\_1isp\_B, 2)), 1);

p2DDoc->ksLineSeg(0, sqrt(pow(mufta\_1isp\_B, 2) + pow(mufta\_1isp\_B, 2)), 100 \* cos(45), 100 \* cos(45) + sqrt(pow(mufta\_1isp\_B, 2) + pow(mufta\_1isp\_B, 2)), 1);

p2DDoc->ksLineSeg(-100 \* cos(45), 100 \* cos(45) + sqrt(pow(mufta\_1isp\_B, 2) + pow(mufta\_1isp\_B, 2)), 100 \* cos(45), 100 \* cos(45) + sqrt(pow(mufta\_1isp\_B, 2) + pow(mufta\_1isp\_B, 2)), 1);

p2DDoc->ksLineSeg(0, 0, -150 \* cos(45), 150 \* cos(45), 1);

p2DDoc->ksLineSeg(0, 0, 150 \* cos(45), 150 \* cos(45), 1);

p2DDoc->ksLineSeg(-150 \* cos(45), 150 \* cos(45), 150 \* cos(45), 150 \* cos(45), 1);

pSketchDef\_muft\_pl->EndEdit();

//вырезаем эскиз зубов

ksEntityPtr pExtrude4 = pPart->NewEntity(o3d\_cutExtrusion);

ksCutExtrusionDefinitionPtr pCutDef4 = pExtrude4->GetDefinition();

pCutDef4->directionType = dtNormal;

pCutDef4->SetSketch(pSketchDef\_muft\_pl);

pCutDef4->SetSideParam(true, etBlind, l, 0, false);

pExtrude4->Create();

Листинг 10 – Создание эскиза и дальнейшее вырезание зуба.

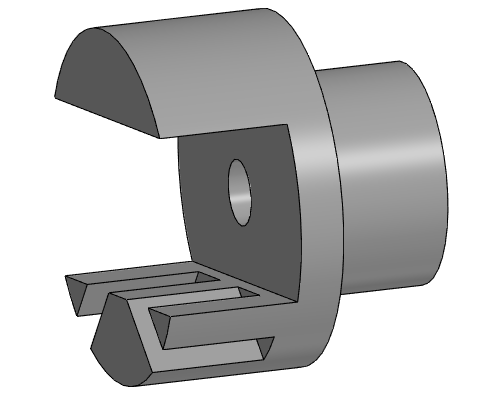


Рисунок 13 — Результат Листинга 10

//аналогично создаем эскиз для другого зуба

ksEntityPtr pSketch\_muft\_pl2 = pPart->NewEntity(o3d\_sketch);

ksSketchDefinitionPtr pSketchDef\_muft\_pl2 = pSketch\_muft\_pl2->GetDefinition();

pSketchDef\_muft\_pl2->SetPlane(pPart->GetDefaultEntity(o3d\_planeYOZ));

pSketch\_muft\_pl2->Create();

p2DDoc = pSketchDef\_muft\_pl2->BeginEdit();

UpdateData();

p2DDoc->ksLineSeg(0, -sqrt(pow(mufta\_1isp\_B, 2) + pow(mufta\_1isp\_B, 2)), -100 \* cos(45), -(100 \* cos(45) + sqrt(pow(mufta\_1isp\_B, 2) + pow(mufta\_1isp\_B, 2))), 1);

p2DDoc->ksLineSeg(0, -sqrt(pow(mufta\_1isp\_B, 2) + pow(mufta\_1isp\_B, 2)), 100 \* cos(45), -(100 \* cos(45) + sqrt(pow(mufta\_1isp\_B, 2) + pow(mufta\_1isp\_B, 2))), 1);

p2DDoc->ksLineSeg(-100 \* cos(45), -(100 \* cos(45) + sqrt(pow(mufta\_1isp\_B, 2) + pow(mufta\_1isp\_B, 2))), 100 \* cos(45), -(100 \* cos(45) + sqrt(pow(mufta\_1isp\_B, 2) + pow(mufta\_1isp\_B, 2))), 1);

p2DDoc->ksLineSeg(100, 0, -100, 0, 1);

p2DDoc->ksLineSeg(100, 0, 100, -100, 1);

p2DDoc->ksLineSeg(100, -100, -100, -100, 1);

p2DDoc->ksLineSeg(-100, -100, -100, 0, 1);

pSketchDef\_muft\_pl2->EndEdit();

//вырезаем полученный зуб

ksEntityPtr pExtrude5 = pPart->NewEntity(o3d\_cutExtrusion);

ksCutExtrusionDefinitionPtr pCutDef5 = pExtrude5->GetDefinition();

pCutDef5->directionType = dtNormal;

pCutDef5->SetSketch(pSketchDef\_muft\_pl2);

pCutDef5->SetSideParam(true, etBlind, l, 0, false);

pExtrude5->Create();

Листинг 11 – Создание эскиза и дальнейшее вырезание второго зуба.

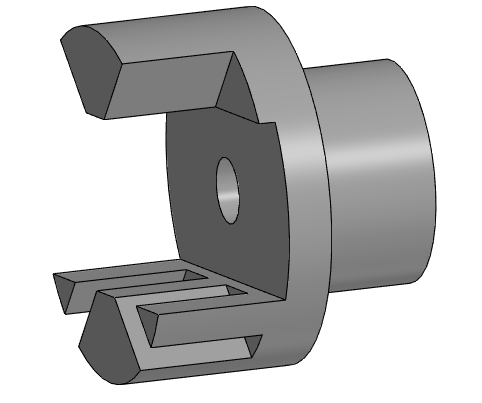


Рисунок 14 — Результат Листинга 11

//для первого зуба создаем выступы для звезды

double glubina\_tre2 = mufta\_1isp\_L1 - mufta\_1isp\_l;

ksEntityPtr pSketch\_muft\_pl3 = pPart->NewEntity(o3d\_sketch);

ksSketchDefinitionPtr pSketchDef\_muft\_pl3 = pSketch\_muft\_pl3->GetDefinition();

pSketchDef\_muft\_pl3->SetPlane(pPart->GetDefaultEntity(o3d\_planeYOZ));

pSketch\_muft\_pl3->Create();

p2DDoc = pSketchDef\_muft\_pl3->BeginEdit();

UpdateData();

p2DDoc->ksLineSeg(0, 0, -80, 80, 1);

p2DDoc->ksLineSeg(0, 0, 80, 80, 1);

p2DDoc->ksLineSeg(-80, 80, 80, 80, 1);

p2DDoc->ksLineSeg(100, -1, -100, -1, 1);

p2DDoc->ksLineSeg(100, -1, 100, 100, 1);

p2DDoc->ksLineSeg(100, 100, -100, 100, 1);

p2DDoc->ksLineSeg(-100, 100, -100, -1, 1);

pSketchDef\_muft\_pl3->EndEdit();

//вырезаем, чтобы появились выступы для звезды

ksEntityPtr pExtrude6 = pPart->NewEntity(o3d\_cutExtrusion);

ksCutExtrusionDefinitionPtr pCutDef6 = pExtrude6->GetDefinition();

pCutDef6->directionType = dtNormal;

pCutDef6->SetSketch(pSketchDef\_muft\_pl3);

pCutDef6->SetSideParam(true, etBlind, glubina\_tre2, 0, false);

pExtrude6->Create();

Листинг 12 – Создание выступов для первого зуба.

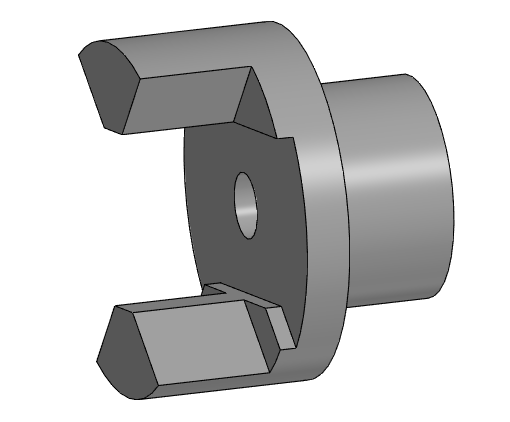


Рисунок 15 — Результат Листинга 12

//аналогично для второго зуба (эскиз)

ksEntityPtr pSketch\_muft\_pl4 = pPart->NewEntity(o3d\_sketch);

ksSketchDefinitionPtr pSketchDef\_muft\_pl4 = pSketch\_muft\_pl4->GetDefinition();

pSketchDef\_muft\_pl4->SetPlane(pPart->GetDefaultEntity(o3d\_planeYOZ));

pSketch\_muft\_pl4->Create();

p2DDoc = pSketchDef\_muft\_pl4->BeginEdit();

UpdateData();

p2DDoc->ksLineSeg(0, 0, -80, -80, 1);

p2DDoc->ksLineSeg(0, 0, 80, -80, 1);

p2DDoc->ksLineSeg(-80, -80, 80, -80, 1);

p2DDoc->ksLineSeg(100, 1, -100, 1, 1);

p2DDoc->ksLineSeg(100, 1, 100, -100, 1);

p2DDoc->ksLineSeg(100, -100, -100, -100, 1);

p2DDoc->ksLineSeg(-100, -100, -100, 1, 1);

pSketchDef\_muft\_pl4->EndEdit();

//аналогично для второго зуба (вырез)

ksEntityPtr pExtrude7 = pPart->NewEntity(o3d\_cutExtrusion);

ksCutExtrusionDefinitionPtr pCutDef7 = pExtrude7->GetDefinition();

pCutDef7->directionType = dtNormal;

pCutDef7->SetSketch(pSketchDef\_muft\_pl4);

pCutDef7->SetSideParam(true, etBlind, glubina\_tre2, 0, false);

pExtrude7->Create();

Листинг 13 – Создание выступов для второго зуба.

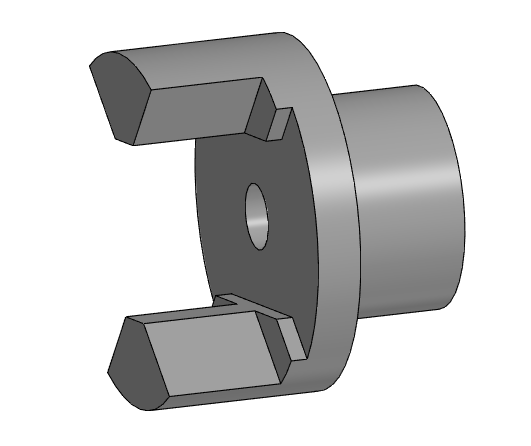


Рисунок 16 — Результат Листинга 13

////создаем фаски и обзываем грани для создания сборки

ksEntityPtr pLoft = pPart->NewEntity(o3d\_bossLoft);

ksBossLoftDefinitionPtr pLoftDef = pLoft->GetDefinition();

pLoftDef->SetLoftParam(FALSE, TRUE, TRUE);

ksEntityCollectionPtr sk = pLoftDef->Sketchs();

sk->Clear();

sk->Add(pExtrude2);

sk->Add(pExtrude3);

sk->Add(pExtrude4);

sk->Add(pExtrude5);

sk->Add(pExtrude6);

sk->Add(pExtrude7);

pLoft->Create();

ksEntityCollectionPtr fledges = pPart->EntityCollection(o3d\_edge);

ksEntityPtr pChamfer = pPart->NewEntity(o3d\_chamfer);

ksChamferDefinitionPtr pChamferDef = pChamfer->GetDefinition();

pChamferDef->SetChamferParam(false, 1, 1);

ksEntityCollectionPtr fl = pChamferDef->array();

fl->Clear();

int j = 0;

int m = 0;pExtrude7->Create();

Листинг 14 – Создание фасок.

for (int i = 0; i < fledges->GetCount(); i++)

{

ksEntityPtr ed = fledges->GetByIndex(i);

ksEdgeDefinitionPtr def = ed->GetDefinition();

if (def->IsCircle())

{

ksVertexDefinitionPtr p1 = def->GetVertex(true);

ksVertexDefinitionPtr p2 = def->GetVertex(false);

double x1, y1, z1, x2, y2, z2;

if (p1 && p2)

{

p1->GetPoint(&x1, &y1, &z1);

p2->GetPoint(&x2, &y2, &z2);

if (abs(z1) < 0.001)

{

//создание фасок

fl->Add(ed);

}

}

}

if (def->IsStraight()) {

ksVertexDefinitionPtr p1 = def->GetVertex(true);

ksVertexDefinitionPtr p2 = def->GetVertex(false);

double x1, y1, z1, x2, y2, z2;

if (p1 && p2)

{

p1->GetPoint(&x1, &y1, &z1);

p2->GetPoint(&x2, &y2, &z2);

if (x1 == l && y1<-B) {

ksFaceDefinitionPtr f = def->GetAdjacentFace(true);

ksEntityPtr face = f->GetEntity();

if (f->IsPlanar() && j == 0) {

//называем боковую грань зуба

face->Putname("Face4Assembly");

face->Update();

j++;

}

if (!f->IsPlanar()) {

//называем внешнюю цилиндрическую часть муфты

face->Putname("Face174Assembly");

face->Update();

}

}

if (x1 == 0 && m == 0) {

ksFaceDefinitionPtr f = def->GetAdjacentFace(true);

ksEntityPtr face = f->GetEntity();

if (f->IsPlanar()) {

//называем верхнюю плоскость зуба

face->Putname("Face777Assembly");

face->Update();

m++;

}

}

}

}

}

pChamfer->Create();

Листинг 15 – Создание фасок (продолжение).

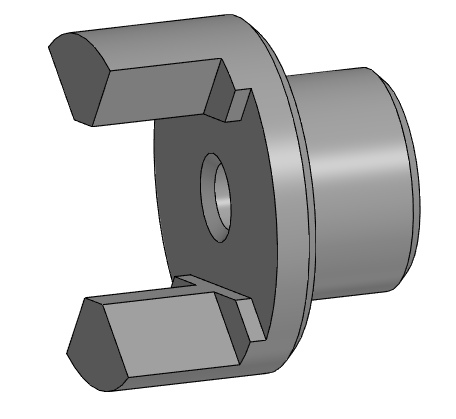


Рисунок 17 — Результат Листинга 15

//сохраняем муфту

pDoc->SaveAs("D:\\kursachTEST\\mufta.m3d");

Листинг 16 – Сохраняем муфту.

**2.6. Создание сборки (1 исполнение).**

После того, как все детали сделаны, мы можем приступать к созданию сборки.

// создание нового документы сборки

pDoc = pKompasApp5->Document3D();

pDoc->Create(false, false);

pPart = pDoc->GetPart(pTop\_Part);

ksPartPtr pBoss, pGear1, pGear2;

//добавляем в сборку ранее созданные детали

pDoc->SetPartFromFile("D:\\kursachTEST\\star1.m3d", pPart, true);

pDoc->SetPartFromFile("D:\\kursachTEST\\mufta.m3d", pPart, true);

pDoc->SetPartFromFile("D:\\kursachTEST\\mufta.m3d", pPart, true);

pBoss = pDoc->GetPart(0);

pGear1 = pDoc->GetPart(1);

pGear2 = pDoc->GetPart(2);

//присваиваем граням переменные для создания зависимостей

ksEntityCollectionPtr col2 = pBoss->EntityCollection(o3d\_face);

ksEntityPtr BossFace4Assemly0 = col2->GetByName("star\_tor", true, true);

ksEntityPtr BossFace4Assemly1 = col2->GetByName("starTOP", true, true);

ksEntityPtr BossFace4Assemly2 = col2->GetByName("circle", true, true);

ksEntityPtr BossFace4Assemly3 = col2->GetByName("star\_tor2", true, true);

col2 = pGear1->EntityCollection(o3d\_face);

ksEntityPtr Gear1Face4Assemly = col2->GetByName("Face4Assembly", true, true);

ksEntityPtr Gear1Face4Assemly1 = col2->GetByName("Face174Assembly", true, true);

ksEntityPtr Gear1Face4Assemly2 = col2->GetByName("Face777Assembly", true, true);

col2 = pGear2->EntityCollection(o3d\_face);

ksEntityPtr Gear2Face4Assemly = col2->GetByName("Face4Assembly", true, true);

ksEntityPtr Gear2Face4Assemly1 = col2->GetByName("Face174Assembly", true, true);

ksEntityPtr Gear2Face4Assemly2 = col2->GetByName("Face777Assembly", true, true);

//создаем зависимости (соосность)

pDoc->AddMateConstraint(mc\_Concentric, BossFace4Assemly2, Gear1Face4Assemly1, -1, 1, 0);

pDoc->AddMateConstraint(mc\_Concentric, BossFace4Assemly2, Gear2Face4Assemly1, 1, 0, 0);

//создаем зависимости (совпадение)

pDoc->AddMateConstraint(mc\_Coincidence, Gear1Face4Assemly, BossFace4Assemly0, -1, 1, 0);

pDoc->AddMateConstraint(mc\_Coincidence, Gear2Face4Assemly, BossFace4Assemly3, -1, 1, 0);

pDoc->AddMateConstraint(mc\_Coincidence, Gear2Face4Assemly2, BossFace4Assemly1, 1, 0, 0);

pDoc->RebuildDocument();

//сохраняем документ сборки

pDoc->SaveAs("D:\\kursachTEST\\Assembly.3d");

Листинг 17 – Создаем сборку и сохраняем ее.

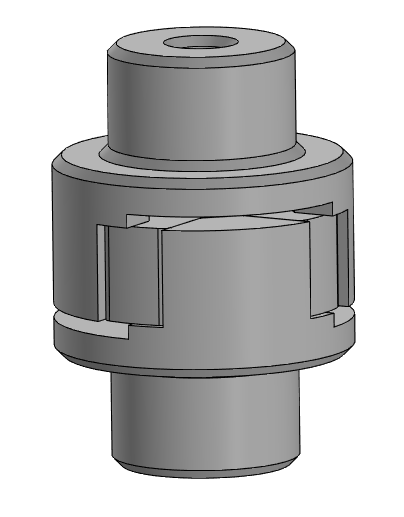


Рисунок 18 — Результат Листинга 17

**2.7. Создание параметров размеров деталей и сборки (2 исполнение).**

Выбор параметров будет осуществляться с помощью Combo Box.

void CNewDialog2::OnCbnSelchangeCombo2()

{

// Получаем индекс выбранного элемента

int selectedIndex = execution2.GetCurSel();

// Проверяем выбранный элемент и устанавливаем значения переменных соответственно

if (selectedIndex == 0) // Выбрано "1"

{

D = 53;

D1 = 50;

D2 = 26;

d = 12;

L = 81;

L1 = 48;

l = 15;

l1 = 28;

l2 = 10;

B = 10;

b1 = 5;

R = 1.6;

lDeep = 30;

B1 = 14;

d\_star = 26;

H = 15;

}

else if (selectedIndex == 1) // Выбрано "2"

{

D = 71;

D1 = 70;

D2 = 30;

d = 16;

L = 101;

L1 = 58;

l = 15;

l1 = 28;

l2 = 10;

B = 12;

b1 = 6;

R = 1.6;

lDeep = 40;

B1 = 16;

d\_star = 30;

H = 15;

}

Листинг 18 – Создание параметров сборки

else if (selectedIndex == 2) // Выбрано "3"

{

D = 85;

D1 = 80;

D2 = 36;

d = 20;

L = 128;

L1 = 75;

l = 22;

l1 = 40;

l2 = 12;

B = 14;

b1 = 7;

R = 2;

lDeep = 50;

B1 = 16;

d\_star = 30;

H = 22;

}

else if (selectedIndex == 3) // Выбрано "4"

{

D = 105;

D1 = 100;

D2 = 45;

d = 25;

L = 148;

L1 = 85;

l = 22;

l1 = 40;

l2 = 12;

B = 16;

b1 = 8;

R = 2;

lDeep = 60;

B1 = 25;

d\_star = 36;

H = 22;

}

else if (selectedIndex == 4) // Выбрано "5"

{

D = 135;

D1 = 130;

D2 = 55;

d = 32;

L = 191;

L1 = 108;

l = 25;

l1 = 48;

l2 = 12;

B = 18;

b1 = 9;

R = 3;

lDeep = 80;

B1 = 32;

d\_star = 45;

H = 25;

}

Листинг 18 – Создание параметров сборки (продолжение)

else if (selectedIndex == 5) // Выбрано "6"

{

D = 166;

D1 = 160;

D2 = 63;

d = 38;

L = 196;

L1 = 113;

l = 30;

l1 = 56;

l2 = 12;

B = 20;

b1 = 10;

R = 3;

lDeep = 80;

B1 = 38;

d\_star = 56;

H = 30;

}

// Обновляем данные на диалоговом окне

UpdateData(FALSE);

}

Листинг 18 – Создание параметров сборки (продолжение)

**2.8. Создание звездочки (2 исполнение).**

Для начала создадим эскиз круга, а затем выдавим его.

pDoc2 = pKompasApp55->Document3D();

pDoc2->Create(false, true);

pPart2 = pDoc2->GetPart(pTop\_Part);

ksEntityPtr pSketch = pPart2->NewEntity(o3d\_sketch);

ksSketchDefinitionPtr pSketchDef = pSketch->GetDefinition();

pSketchDef->SetPlane(pPart2->GetDefaultEntity(o3d\_planeXOY));

pSketch->Create();

ksDocument2DPtr p2DDoc = pSketchDef->BeginEdit();

UpdateData();

double star\_D1 = D1;

double star\_r = R;

double star\_H = H;

double star\_d = d\_star;

double star\_B = B;

//создаем вспомогательный эскиз круга для создания звезды

p2DDoc->ksCircle(0, 0, star\_d / 2, 1);

pSketchDef->EndEdit();

//выдавливаем эскиз

ksEntityPtr pExtrudeDowel\_st = pPart2->NewEntity(o3d\_baseExtrusion);

ksBaseExtrusionDefinitionPtr pBaseDef\_st = pExtrudeDowel\_st->GetDefinition();

pBaseDef\_st->directionType = dtNormal;

pBaseDef\_st->SetSketch(pSketch);

pBaseDef\_st->SetSideParam(true, etBlind, star\_H, 0, false);

pExtrudeDowel\_st->Create();

Листинг 19 – Создание эскиза круга и его выдавливания

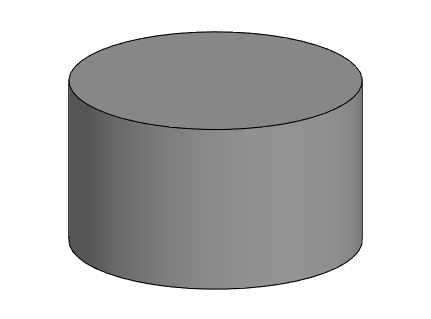


Рисунок 19 — Результат Листинга 19

Затем создадим смещенную плоскость для дальнейшего построения лучей звезды, в конце произведем выдавливание эскиза луча.

//смещенная плоскость для создания лучиков звезды

ksEntityPtr pPlane\_st = pPart2->NewEntity(o3d\_planeOffset);

ksPlaneOffsetDefinitionPtr pPlaneDef\_st = pPlane\_st->GetDefinition();

pPlaneDef\_st->direction = true;

pPlaneDef\_st->offset = star\_H;

pPlaneDef\_st->SetPlane(pPart2->GetDefaultEntity(o3d\_planeXOY));

pPlane\_st->Create();

//создание эскиза луча звезды (прямоугольник)

ksEntityPtr pSketch\_st = pPart2->NewEntity(o3d\_sketch);

pSketchDef = pSketch\_st->GetDefinition();

pSketchDef->SetPlane(pPlane\_st);

pSketch\_st->Create();

p2DDoc = pSketchDef->BeginEdit();

p2DDoc->ksLineSeg(0, star\_B / 2, star\_D1, star\_B / 2, 1);

p2DDoc->ksLineSeg(star\_D1, star\_B / 2, star\_D1, -star\_B / 2, 1);

p2DDoc->ksLineSeg(star\_D1, -star\_B / 2, 0, -star\_B / 2, 1);

p2DDoc->ksLineSeg(0, star\_B / 2, 0, -star\_B / 2, 1);

pSketchDef->EndEdit();

//выдавливание эскиза луча

ksEntityPtr pExtrudeDowel\_st1 = pPart2->NewEntity(o3d\_baseExtrusion);

ksBaseExtrusionDefinitionPtr pBaseDef\_st1 = pExtrudeDowel\_st1->GetDefinition();

pBaseDef\_st1->directionType = dtReverse;

pBaseDef\_st1->SetSketch(pSketch\_st);

pBaseDef\_st1->SetSideParam(false, etBlind, star\_H, 0, false);

pExtrudeDowel\_st1->Create();

Листинг 20 – Смещенная плоскость, построение луча звезды, выдавливание.

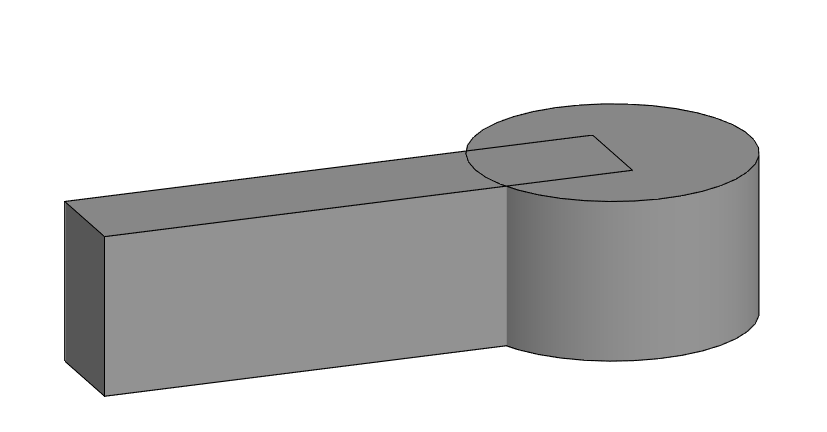


Рисунок 20 — Результат Листинга 20

//круговой массив

ksEntityPtr pEx1 = pPart2->NewEntity(o3d\_circularCopy);

//Получаем интерфейс параметрова операции

ksCircularCopyDefinitionPtr CircularCopyDefinition = pEx1->GetDefinition();

//Количество копий в радиальном направлении

CircularCopyDefinition->count1 = 1;

//Устанавливаем ось операции

CircularCopyDefinition->SetAxis(pPart2->GetDefaultEntity(o3d\_axisOZ));

//Устанавливаем параметры копирования

CircularCopyDefinition->SetCopyParamAlongDir(6, 60, FALSE, FALSE);

//Получаем массив копируемых элементов

ksEntityCollectionPtr EntityCollection = CircularCopyDefinition->GetOperationArray();

EntityCollection->Clear();

//Заполняем массив копируемыъ элементов

EntityCollection->Add(pExtrudeDowel\_st1);

//Создаем операцию

pEx1->Create();

Листинг 21 – Создаем круговой массив для копирования лучей.

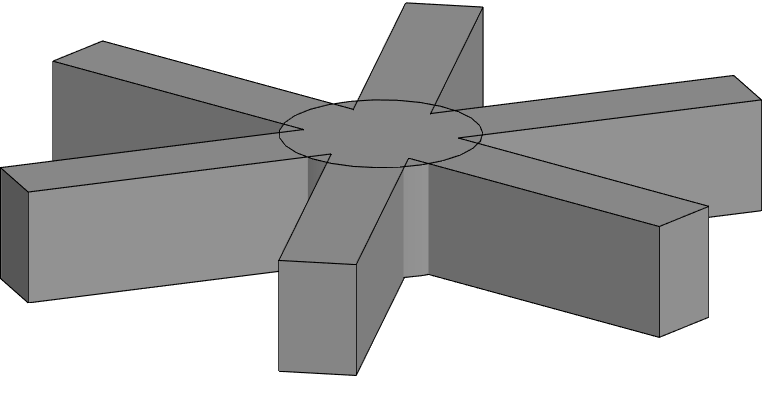


Рисунок 21 — Результат Листинга 21

//создание на ранее созданной смещенной плоскости эскизов кругов для дальнейшего обрезание лучей и их скругления

ksEntityPtr pSketch\_st1 = pPart2->NewEntity(o3d\_sketch);

pSketchDef = pSketch\_st1->GetDefinition();

pSketchDef->SetPlane(pPlane\_st);

pSketch\_st1->Create();

p2DDoc = pSketchDef->BeginEdit();

p2DDoc->ksCircle(0, 0, star\_D1 / 2, 1);

p2DDoc->ksCircle(0, 0, star\_D1 \* 2, 1);

pSketchDef->EndEdit();

//вырезаем выдавливанием эскиз кругов

ksEntityPtr pExtrude\_st = pPart2->NewEntity(o3d\_cutExtrusion);

ksCutExtrusionDefinitionPtr pCutDef\_st = pExtrude\_st->GetDefinition();

pCutDef\_st->directionType = dtNormal;

pCutDef\_st->SetSketch(pSketch\_st1);

pCutDef\_st->SetSideParam(true, etThroughAll, 0, 0, false);

pExtrude\_st->Create();

//получившиеся скругленные края называем circle для дальнейшего применения соосности

ksEntityCollectionPtr flFaces = pPart2->EntityCollection(o3d\_face);

for (int i = 0; i < flFaces->GetCount(); i++)

{

ksEntityPtr face = flFaces->GetByIndex(i);

ksFaceDefinitionPtr def = face->GetDefinition();

if (def->GetOwnerEntity() == pExtrude\_st)

{

face->Putname("circle");

face->Update();

}

}

Листинг 22 – Создание кругов на смещенной плоскости, даем название краям.

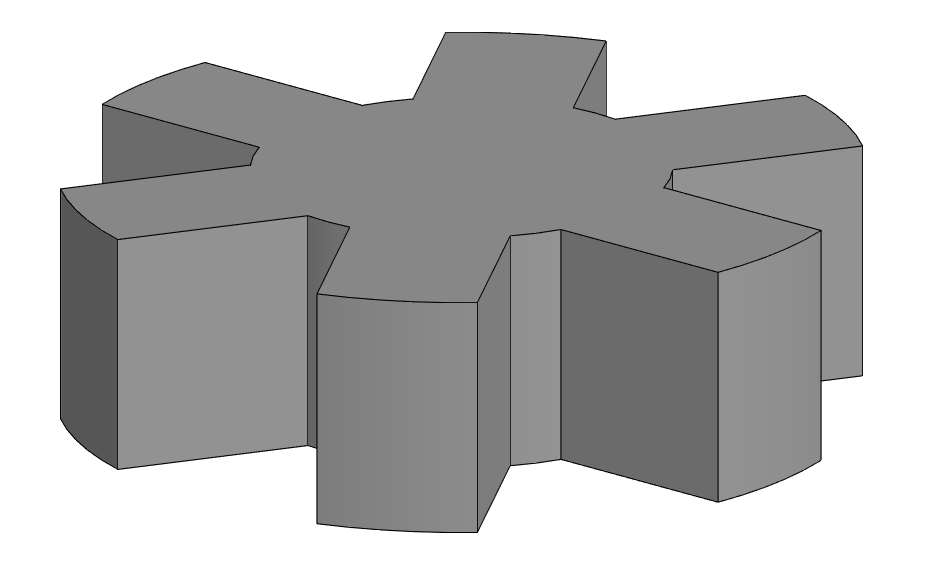


Рисунок 22 — Результат Листинга 22

Листинг 23 – Создаем скругления, даем названия граням.

//создаем необходимые скругления

ksEntityCollectionPtr fledges1 = pPart2->EntityCollection(o3d\_edge);

ksEntityPtr pFillet = pPart2->NewEntity(o3d\_fillet);

ksFilletDefinitionPtr pFilletDef = pFillet->GetDefinition();

pFilletDef->radius = R;

ksEntityCollectionPtr fl1 = pFilletDef->array();

fl1->Clear();

int k = 0;

int k2 = 0;

int k3 = 0;

for (int i = 0; i < fledges1->GetCount(); i++)

{

ksEntityPtr ed = fledges1->GetByIndex(i);

ksEdgeDefinitionPtr def = ed->GetDefinition();

if (def->IsStraight())

{

ksVertexDefinitionPtr p1 = def->GetVertex(true);

ksVertexDefinitionPtr p2 = def->GetVertex(false);

double x1, y1, z1, x2, y2, z2;

if (p1 && p2)

{

p1->GetPoint(&x1, &y1, &z1);

p2->GetPoint(&x2, &y2, &z2);

if (abs(z1) == l && ceil(abs(x1)) == D1 / 2 && abs(int(y1)) != B / 2 && k == 0)

{

//называем верхнюю грань

ksFaceDefinitionPtr f = def->GetAdjacentFace(true);

ksEntityPtr face = f->GetEntity();

face->Putname("starTOP");

face->Update();

k++;

}

if (abs(x1 - x2) < 0.0001 && abs(y1 - y2) < 0.0001)

{

if (z1 == 0 && int(y1) == -B / 2 && k3 == 1 && ceil(abs(x1)) == D1 / 2) {

//называем внутреннюю часть зуба

ksFaceDefinitionPtr f = def->GetAdjacentFace(true);

ksEntityPtr face = f->GetEntity();

face->Putname("star\_tor2");

face->Update();

k3++;

}

}

if (abs(x1 - x2) < 0.0001 && abs(y1 - y2) < 0.0001)

{

fl1->Add(ed);

if (z1 == 0 && k2 == 0) {

//называем другую внутреннюю часть зуба

ksFaceDefinitionPtr f = def->GetAdjacentFace(true);

ksEntityPtr face = f->GetEntity();

face->Putname("star\_tor");

face->Update();

k2++;

k3 = 1;

}

}

}

}

}

pFillet->Create();

Листинг 23 – Создаем скругления, даем названия граням (продолжение).

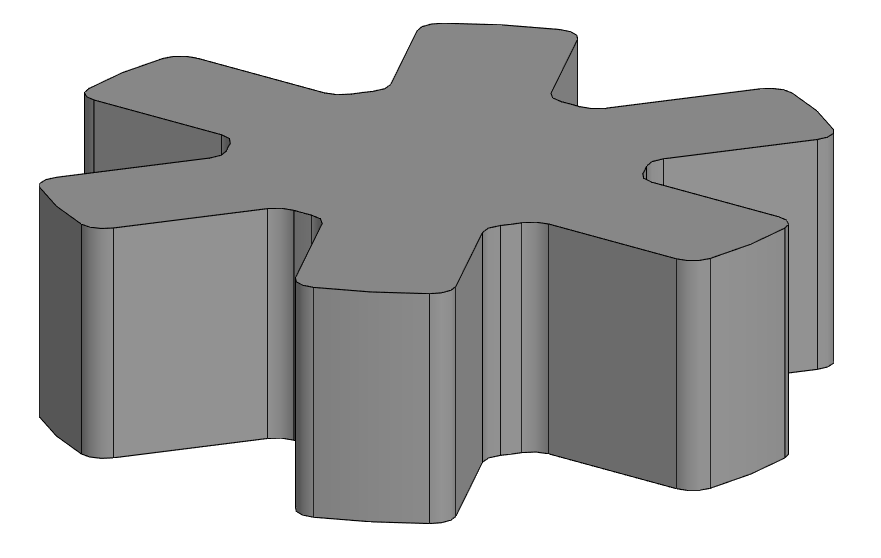


Рисунок 23 — Результат Листинга 23

//сохраняем звезду

pDoc2->SaveAs("D:\\kursachTEST\\star2.m3d");

Листинг 24 – Сохраняем звезду.

**2.9. Создание полумуфты (2 исполнение).**

Прежде чем приступить к созданию полумуфты, необходимо создать новый документ, потом создать эскиз муфты, а затем выдавить ее вращением.

//создаем новый документ, где будем строить муфту

pDoc2 = pKompasApp55->Document3D();

pDoc2->Create(false, true);

pPart2 = pDoc2->GetPart(pTop\_Part);

pSketch = pPart2->NewEntity(o3d\_sketch);

pSketchDef = pSketch->GetDefinition();

pSketchDef->SetPlane(pPart2->GetDefaultEntity(o3d\_planeXOY));

pSketch->Create();

p2DDoc = pSketchDef->BeginEdit();

UpdateData();

double mufta\_2isp\_B1 = B1;

double mufta\_2isp\_D = D;

double mufta\_2isp\_D1 = D1;

double mufta\_2isp\_D2 = D2;

double mufta\_2isp\_d = d;

double mufta\_2isp\_L1 = L1;

double mufta\_2isp\_l = lDeep;

double mufta\_2isp\_B = B/2;

double mufta\_2isp\_L = L;

double mufta\_2isp\_l\_deep = lDeep;

//эскиз для муфты

p2DDoc->ksLineSeg(0, -mufta\_2isp\_B1 / 2, 0, -mufta\_2isp\_D / 2, 1);

p2DDoc->ksLineSeg(0, -mufta\_2isp\_D / 2, mufta\_2isp\_l, -mufta\_2isp\_D / 2, 1);

p2DDoc->ksLineSeg(mufta\_2isp\_l, -mufta\_2isp\_D / 2, mufta\_2isp\_l, -mufta\_2isp\_D2 / 2, 1);

p2DDoc->ksLineSeg(mufta\_2isp\_l, -mufta\_2isp\_D2 / 2, mufta\_2isp\_L1, -mufta\_2isp\_D2 / 2, 1);

p2DDoc->ksLineSeg(mufta\_2isp\_L1, -mufta\_2isp\_D2 / 2, mufta\_2isp\_L1, -mufta\_2isp\_d / 2, 1);

p2DDoc->ksLineSeg(mufta\_2isp\_L1, -mufta\_2isp\_d / 2, mufta\_2isp\_L1 - mufta\_2isp\_l\_deep, -mufta\_2isp\_d / 2, 1);

p2DDoc->ksLineSeg(mufta\_2isp\_L1 - mufta\_2isp\_l\_deep, -mufta\_2isp\_d / 2, mufta\_2isp\_L1 - mufta\_2isp\_l\_deep, -mufta\_2isp\_B1 / 2, 1);

p2DDoc->ksLineSeg(mufta\_2isp\_L1 - mufta\_2isp\_l\_deep, -mufta\_2isp\_B1 / 2, 0, -mufta\_2isp\_B1 / 2, 1);

p2DDoc->ksLineSeg(-10, 0, 10, 0, 3);

pSketchDef->EndEdit();

//выдавливаем вращением полученный эскиз

ksEntityPtr pRotate2 = pPart2->NewEntity(o3d\_bossRotated);

ksBossRotatedDefinitionPtr pRotDef2 = pRotate2->GetDefinition();

pRotDef2->SetSketch(pSketch);

pRotDef2->SetSideParam(TRUE, 360);

pRotate2->Create();

Листинг 25 – Создание нового документа, эскиза муфты, выдавливания.

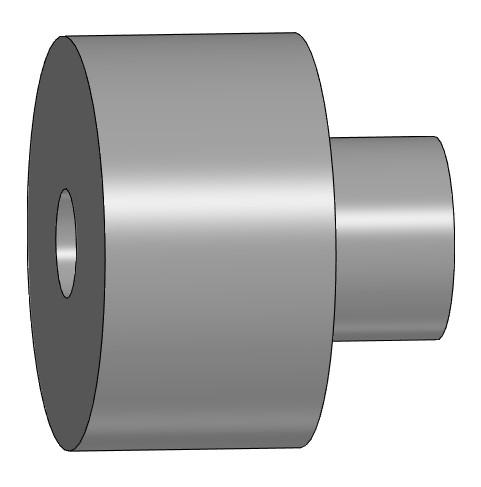


Рисунок 24 — Результат Листинга 25

//создание эскиза двух треугольников с углом 60 градусов для вырезания зубьев муфты

ksEntityPtr pSketch\_muft\_pl7 = pPart2->NewEntity(o3d\_sketch);

ksSketchDefinitionPtr pSketchDef\_muft\_pl7 = pSketch\_muft\_pl7->GetDefinition();

pSketchDef\_muft\_pl7->SetPlane(pPart2->GetDefaultEntity(o3d\_planeYOZ));

pSketch\_muft\_pl7->Create();

p2DDoc = pSketchDef\_muft\_pl7->BeginEdit();

UpdateData();

double angle\_BAC\_degrees = 60; // Угол BAC

double length\_BC2 = mufta\_2isp\_D \* 5; // Длина стороны BC2

double y\_A2 = 0;

double x\_B2 = length\_BC2 \* cos(angle\_BAC\_degrees \* PI / 180.0);

double y\_B2 = length\_BC2 \* sin(angle\_BAC\_degrees \* PI / 180.0);

double x\_C2 = -length\_BC2 \* cos(angle\_BAC\_degrees \* PI / 180.0);

double y\_C2 = length\_BC2 \* sin(angle\_BAC\_degrees \* PI / 180.0);

double length\_BC = mufta\_2isp\_D \* 3; // Длина стороны BC

// Координаты точки A (начальная точка)

double x\_A = 0;

double y\_A = B;

double y\_A3 = -B;

// Координаты точки B (конечная точка первой линии)

double x\_B = length\_BC \* cos(angle\_BAC\_degrees \* PI / 180.0);

double y\_B = y\_A + length\_BC \* sin(angle\_BAC\_degrees \* PI / 180.0);

double y\_B3 = -y\_A + length\_BC \* sin(angle\_BAC\_degrees \* PI / 180.0);

// Координаты точки C (конечная точка второй линии)

double x\_C = -length\_BC \* cos(angle\_BAC\_degrees \* PI / 180.0);

double y\_C = y\_A + length\_BC \* sin(angle\_BAC\_degrees \* PI / 180.0);

double y\_C3 = -y\_A + length\_BC \* sin(angle\_BAC\_degrees \* PI / 180.0);

p2DDoc->ksLineSeg(x\_A, y\_A, x\_B, y\_B, 1);

p2DDoc->ksLineSeg(x\_A, y\_A, x\_C, y\_C, 1);

p2DDoc->ksLineSeg(x\_B, y\_B, x\_C, y\_C, 1);

p2DDoc->ksLineSeg(x\_A, y\_A2, x\_B2, y\_B2, 1);

p2DDoc->ksLineSeg(x\_A, y\_A2, x\_C2, y\_C2, 1);

p2DDoc->ksLineSeg(x\_B2, y\_B2, x\_C2, y\_C2, 1);

pSketchDef\_muft\_pl7->EndEdit();

//вырезаем полученные эскизы

ksEntityPtr pExtrude9 = pPart2->NewEntity(o3d\_cutExtrusion);

ksCutExtrusionDefinitionPtr pCutDef9 = pExtrude9->GetDefinition();

pCutDef9->directionType = dtNormal;

pCutDef9->SetSketch(pSketchDef\_muft\_pl7);

pCutDef9->SetSideParam(true, etBlind, H, 0, false);

pExtrude9->Create();

Листинг 26 – Создание зуба муфты.

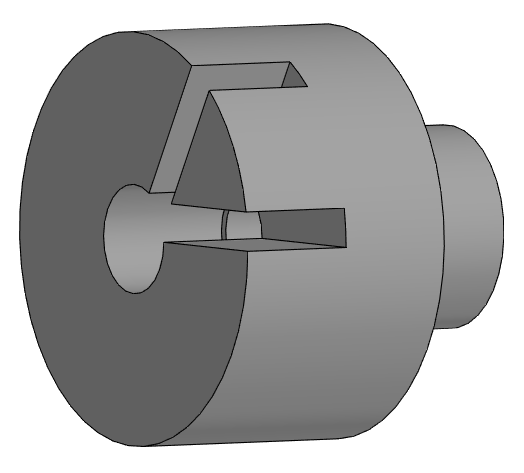


Рисунок 25 — Результат Листинга 26

//строим два треугольника для создания выступов у зубьев

double glubina\_tre4 = H+(L1-lDeep-H);

ksEntityPtr pSketch\_muft\_pl5 = pPart2->NewEntity(o3d\_sketch);

ksSketchDefinitionPtr pSketchDef\_muft\_pl5 = pSketch\_muft\_pl5->GetDefinition();

pSketchDef\_muft\_pl5->SetPlane(pPart2->GetDefaultEntity(o3d\_planeYOZ));

pSketch\_muft\_pl5->Create();

p2DDoc = pSketchDef\_muft\_pl5->BeginEdit();

UpdateData();

double length\_BC69 = mufta\_2isp\_D \* 5; // Длина стороны BC

double length\_BC77 = mufta\_2isp\_D \* 3; // Длина стороны BC

// Координаты точки A (начальная точка)

double x\_A69 = 0;

double y\_A69 = -sqrt(pow(mufta\_2isp\_B, 2) + pow(mufta\_2isp\_B, 2));

double y\_A77 = 0;

// Координаты точки B (конечная точка первой линии)

double x\_B69 = length\_BC69 \* cos(angle\_BAC\_degrees \* PI / 180.0);

double y\_B69 = y\_A69 + length\_BC69 \* sin(angle\_BAC\_degrees \* PI / 180.0);

double x\_B77 = length\_BC77 \* cos(angle\_BAC\_degrees \* PI / 180.0);

double y\_B77 = length\_BC77 \* sin(angle\_BAC\_degrees \* PI / 180.0);

// Координаты точки C (конечная точка второй линии)

double x\_C69 = -length\_BC69 \* cos(angle\_BAC\_degrees \* PI / 180.0);

double y\_C69 = y\_A69 + length\_BC69 \* sin(angle\_BAC\_degrees \* PI / 180.0);

double x\_C77 = -length\_BC77 \* cos(angle\_BAC\_degrees \* PI / 180.0);

double y\_C77 = y\_A77 + length\_BC77 \* sin(angle\_BAC\_degrees \* PI / 180.0);

// Рисование линий

p2DDoc->ksLineSeg(x\_A69, y\_A69, x\_B69, y\_B69, 1);

p2DDoc->ksLineSeg(x\_A69, y\_A69, x\_C69, y\_C69, 1);

p2DDoc->ksLineSeg(x\_B69, y\_B69, x\_C69, y\_C69, 1);

p2DDoc->ksLineSeg(x\_A69, y\_A77, x\_B77, y\_B77, 1);

p2DDoc->ksLineSeg(x\_A69, y\_A77, x\_C77, y\_C77, 1);

p2DDoc->ksLineSeg(x\_B77, y\_B77, x\_C77, y\_C77, 1);

pSketchDef\_muft\_pl5->EndEdit();

//вырезаем эскизы

ksEntityPtr pExtrude11 = pPart2->NewEntity(o3d\_cutExtrusion);

ksCutExtrusionDefinitionPtr pCutDef11 = pExtrude11->GetDefinition();

pCutDef11->directionType = dtNormal;

pCutDef11->SetSketch(pSketchDef\_muft\_pl5);

pCutDef11->SetSideParam(true, etBlind, glubina\_tre4, 0, false);

pExtrude11->Create();

Листинг 27 – Создание выступа зуба муфты.

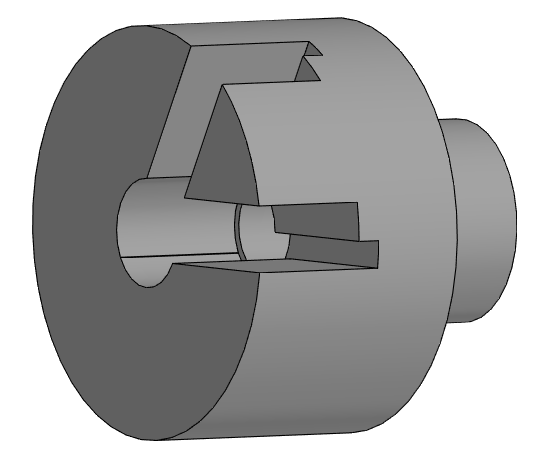


Рисунок 26 — Результат Листинга 27

//создаем эскиз треугольников для удаления лишних элементов, получившися в итоге предыдущих операций

ksEntityPtr pSketch\_muft\_pl174 = pPart2->NewEntity(o3d\_sketch);

ksSketchDefinitionPtr pSketchDef\_muft\_pl174 = pSketch\_muft\_pl174->GetDefinition();

pSketchDef\_muft\_pl174->SetPlane(pPart2->GetDefaultEntity(o3d\_planeYOZ));

pSketch\_muft\_pl174->Create();

p2DDoc = pSketchDef\_muft\_pl174->BeginEdit();

double length\_BC174 = mufta\_2isp\_D \* 4; // Длина стороны BC

// Координаты точки A (начальная точка)

double x\_A174 = 0;

double y\_A174 = 0;

// Координаты точки B (конечная точка первой линии)

double x\_B174 = length\_BC174 \* cos(angle\_BAC\_degrees \* PI / 180.0);

double y\_B174 = length\_BC174 \* sin(angle\_BAC\_degrees \* PI / 180.0);

// Координаты точки C (конечная точка второй линии)

double x\_C174 = -length\_BC174 \* cos(angle\_BAC\_degrees \* PI / 180.0);

double y\_C174 = length\_BC174 \* sin(angle\_BAC\_degrees \* PI / 180.0);

double angle\_shift\_degrees = 60; // Угол смещения

// Преобразование угла смещения в радианы

double angle\_shift\_radians = angle\_shift\_degrees \* PI / 180.0;

// Новые координаты точек B и C с учетом смещения

double x\_B174\_shifted = x\_B174 \* cos(angle\_shift\_radians) - y\_B174 \* sin(angle\_shift\_radians);

double y\_B174\_shifted = x\_B174 \* sin(angle\_shift\_radians) + y\_B174 \* cos(angle\_shift\_radians);

double x\_C174\_shifted = x\_C174 \* cos(angle\_shift\_radians) - y\_C174 \* sin(angle\_shift\_radians);

double y\_C174\_shifted = x\_C174 \* sin(angle\_shift\_radians) + y\_C174 \* cos(angle\_shift\_radians);

// Рисование смещенных линий

p2DDoc->ksLineSeg(x\_A174, y\_A174, x\_B174\_shifted, y\_B174\_shifted, 1);

p2DDoc->ksLineSeg(x\_A174, y\_A174, x\_C174\_shifted, y\_C174\_shifted, 1);

p2DDoc->ksLineSeg(x\_B174\_shifted, y\_B174\_shifted, x\_C174\_shifted, y\_C174\_shifted, 1);

pSketchDef\_muft\_pl174->EndEdit();

//вырезаем полученный эскиз

ksEntityPtr pExtrude174 = pPart2->NewEntity(o3d\_cutExtrusion);

ksCutExtrusionDefinitionPtr pCutDef174 = pExtrude174->GetDefinition();

pCutDef174->directionType = dtNormal;

pCutDef174->SetSketch(pSketchDef\_muft\_pl174);

pCutDef174->SetSideParam(true, etBlind, glubina\_tre4, 0, false);

pExtrude174->Create();

Листинг 28 – Создание эскиза и его выдавливание для того, чтобы убрать лишнее.

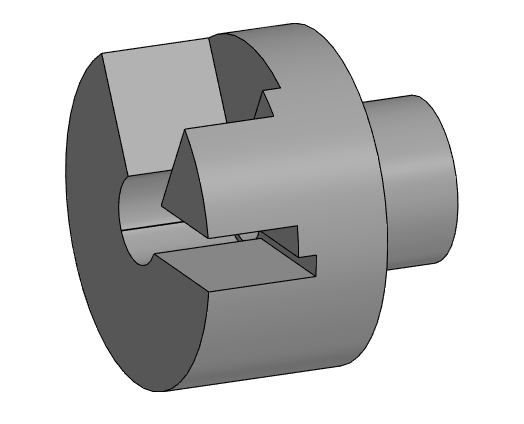


Рисунок 27 — Результат Листинга 28

//создаем эскиз для выдавливания прямоугольника, который выпрямляет внутренние части зубьев

ksEntityPtr pSketch\_muft\_pl777 = pPart2->NewEntity(o3d\_sketch);

ksSketchDefinitionPtr pSketchDef\_muft\_pl777 = pSketch\_muft\_pl777->GetDefinition();

pSketchDef\_muft\_pl777->SetPlane(pPart2->GetDefaultEntity(o3d\_planeYOZ));

pSketch\_muft\_pl777->Create();

p2DDoc = pSketchDef\_muft\_pl777->BeginEdit();

// Координаты вершин прямоугольника относительно его центра

double y\_A777 = 0;

double x\_A777 = -300;

double y\_B777 = 0;

double x\_B777 = 300;

double y\_C777 = B1;

double x\_C777 = 300;

double y\_D777 = B1;

double x\_D777 = -300;

// Рисование прямоугольника

p2DDoc->ksLineSeg(x\_A777, y\_A777, x\_B777, y\_B777, 1);

p2DDoc->ksLineSeg(x\_B777, y\_B777, x\_C777, y\_C777, 1);

p2DDoc->ksLineSeg(x\_C777, y\_C777, x\_D777, y\_D777, 1);

p2DDoc->ksLineSeg(x\_D777, y\_D777, x\_A777, y\_A777, 1);

pSketchDef\_muft\_pl777->EndEdit();

//вырезаем полученный прямоугольник

ksEntityPtr pExtrude777 = pPart2->NewEntity(o3d\_cutExtrusion);

ksCutExtrusionDefinitionPtr pCutDef777 = pExtrude777->GetDefinition();

pCutDef777->directionType = dtNormal;

pCutDef777->SetSketch(pSketchDef\_muft\_pl777);

pCutDef777->SetSideParam(true, etBlind, glubina\_tre4, 0, false);

pExtrude777->Create();

Листинг 29 – Создание эскиза и дальнейшее вырезание зуба.

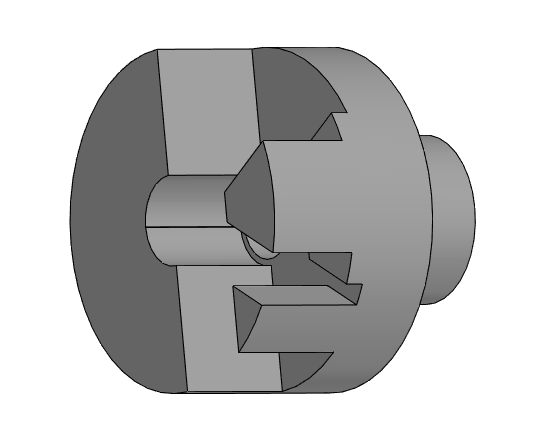


Рисунок 28 — Результат Листинга 29

//создаем круговой массив для повторения (3 раза) операций создания зубьев, выступов и прямоугольника

ksEntityPtr pEx2 = pPart2->NewEntity(o3d\_circularCopy);

//Получаем интерфейс параметрова операции

ksCircularCopyDefinitionPtr CircularCopyDefinition2 = pEx2->GetDefinition();

//Количество копий в радиальном направлении

CircularCopyDefinition2->count1 = 1;

//Устанавливаем ось операции

CircularCopyDefinition2->SetAxis(pPart2->GetDefaultEntity(o3d\_axisOX));

//Устанавливаем параметры копирования

CircularCopyDefinition2->SetCopyParamAlongDir(3, 120, FALSE, FALSE);

//Получаем массив копируемых элементов

ksEntityCollectionPtr EntityCollection2 = CircularCopyDefinition2->GetOperationArray();

EntityCollection2->Clear();

//Заполняем массив копируемыъ элементов

EntityCollection2->Add(pExtrude11);

EntityCollection2->Add(pExtrude9);

EntityCollection2->Add(pExtrude174);

EntityCollection2->Add(pExtrude777);

//Создаем операцию

pEx2->Create();

Листинг 30 – Создание кругового массива.

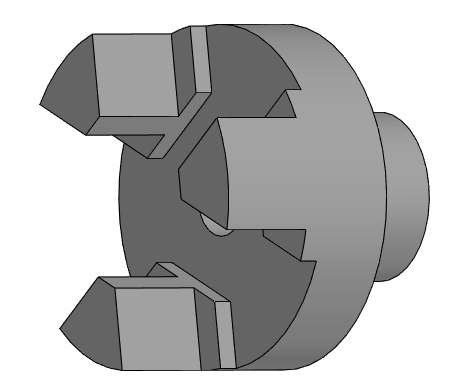


Рисунок 29 — Результат Листинга 30

//называем плоскости, создаем фаски

ksEntityCollectionPtr fledges = pPart2->EntityCollection(o3d\_edge);

ksEntityPtr pChamfer = pPart2->NewEntity(o3d\_chamfer);

ksChamferDefinitionPtr pChamferDef = pChamfer->GetDefinition();

pChamferDef->SetChamferParam(false, 1, 1);

ksEntityCollectionPtr fl = pChamferDef->array();

fl->Clear();

int j = 0;

int m = 0;

Листинг 31 – Создание фасок.

for (int i = 0; i < fledges->GetCount(); i++)

{

ksEntityPtr ed = fledges->GetByIndex(i);

ksEdgeDefinitionPtr def = ed->GetDefinition();

if (def->IsCircle())

{

ksVertexDefinitionPtr p1 = def->GetVertex(true);

ksVertexDefinitionPtr p2 = def->GetVertex(false);

double x1, y1, z1, x2, y2, z2;

if (p1 && p2)

{

p1->GetPoint(&x1, &y1, &z1);

p2->GetPoint(&x2, &y2, &z2);

if (abs(z1) < 0.001)

{

//создаем фаски

fl->Add(ed);

}

}

}

if (def->IsStraight()) {

ksVertexDefinitionPtr p1 = def->GetVertex(true);

ksVertexDefinitionPtr p2 = def->GetVertex(false);

double x1, y1, z1, x2, y2, z2;

if (p1 && p2)

{

p1->GetPoint(&x1, &y1, &z1);

p2->GetPoint(&x2, &y2, &z2);

if (x1==l) {

ksFaceDefinitionPtr f = def->GetAdjacentFace(true);

ksEntityPtr face = f->GetEntity();

if (f->IsPlanar() && j ==0) {

//называем боковую грань зуба

face->Putname("Face4Assembly");

face->Update();

j++;

}

if (!f->IsPlanar()){

//называем внешнюю цилиндрическую часть муфты

face->Putname("Face174Assembly");

face->Update();

}

}

if (x1 == 0 && m ==0) {

ksFaceDefinitionPtr f = def->GetAdjacentFace(true);

ksEntityPtr face = f->GetEntity();

if (f->IsPlanar()) {

//называем верхнюю плоскость зуба

face->Putname("Face777Assembly");

face->Update();

m++;

}

}

}

}

}

pChamfer->Create();

Листинг 31 – Создание фасок (продолжение).

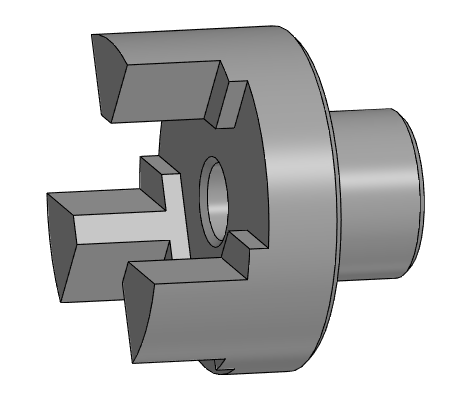


Рисунок 30 — Результат Листинга 31

//сохраняем муфту

pDoc2->SaveAs("D:\\kursachTEST\\mufta2.m3d");

Листинг 32 – Сохраняем муфту.

**2.10. Создание сборки (2 исполнение).**

После того, как все детали сделаны, мы можем приступать к созданию сборки.

// создание нового документы сборки

pDoc2 = pKompasApp55->Document3D();

pDoc2->Create(false, false);

pPart2 = pDoc2->GetPart(pTop\_Part);

ksPartPtr pBoss, pGear1, pGear2;

//добавляем в сборку ранее созданные детали

pDoc2->SetPartFromFile("D:\\kursachTEST\\star2.m3d", pPart2, true);

pDoc2->SetPartFromFile("D:\\kursachTEST\\mufta2.m3d", pPart2, true);

pDoc2->SetPartFromFile("D:\\kursachTEST\\mufta2.m3d", pPart2, true);

pBoss = pDoc2->GetPart(0);

pGear1 = pDoc2->GetPart(1);

pGear2 = pDoc2->GetPart(2);

//присваиваем граням переменные для создания зависимостей

ksEntityCollectionPtr col2 = pBoss->EntityCollection(o3d\_face);

ksEntityPtr BossFace4Assemly0 = col2->GetByName("star\_tor", true, true);

ksEntityPtr BossFace4Assemly1 = col2->GetByName("starTOP", true, true);

ksEntityPtr BossFace4Assemly2 = col2->GetByName("circle", true, true);

ksEntityPtr BossFace4Assemly3 = col2->GetByName("star\_tor2", true, true);

col2 = pGear1->EntityCollection(o3d\_face);

ksEntityPtr Gear1Face4Assemly = col2->GetByName("Face4Assembly", true, true);

ksEntityPtr Gear1Face4Assemly1 = col2->GetByName("Face174Assembly", true, true);

ksEntityPtr Gear1Face4Assemly2 = col2->GetByName("Face777Assembly", true, true);

col2 = pGear2->EntityCollection(o3d\_face);

ksEntityPtr Gear2Face4Assemly = col2->GetByName("Face4Assembly", true, true);

ksEntityPtr Gear2Face4Assemly1 = col2->GetByName("Face174Assembly", true, true);

ksEntityPtr Gear2Face4Assemly2 = col2->GetByName("Face777Assembly", true, true);

//создаем зависимости (соосность)

pDoc2->AddMateConstraint(mc\_Concentric, BossFace4Assemly2, Gear1Face4Assemly1, -1, 1, 0);

pDoc2->AddMateConstraint(mc\_Concentric, BossFace4Assemly2, Gear2Face4Assemly1, 1, 0, 0);

//создаем зависимости (совпадение)

pDoc2->AddMateConstraint(mc\_Coincidence, Gear1Face4Assemly, BossFace4Assemly0, -1, 1, 0);

pDoc2->AddMateConstraint(mc\_Coincidence, Gear2Face4Assemly, BossFace4Assemly3, -1, 1, 0);

pDoc2->AddMateConstraint(mc\_Coincidence, Gear2Face4Assemly2, BossFace4Assemly1, 1, 0, 0);

pDoc2->RebuildDocument();

//сохраняем документ сборки

pDoc2->SaveAs("D:\\kursachTEST\\Assembly.a3d");

Листинг 33 – Создаем сборку и сохраняем ее.

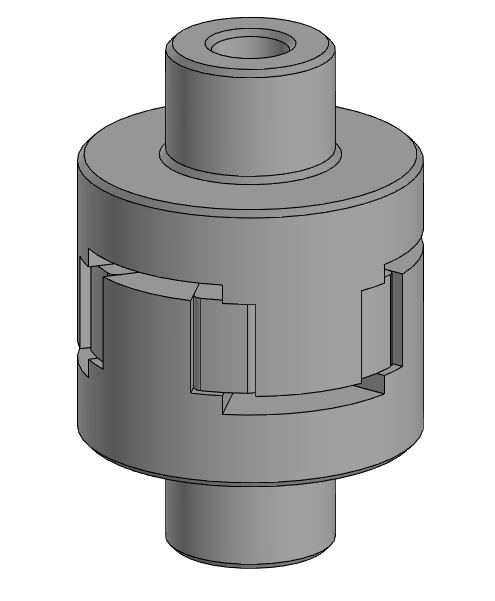


Рисунок 31 — Результат Листинга 33

**Заключение**

В ходе выполнения курсовой работы была разработана и реализована параметрическая сборка компенсирующей упругой муфты с использованием технологии SDI MFC на языке программирования C++. Проект предоставляет пользователям удобный инструмент для выбора и настройки муфт с учетом их упругих характеристик.

Одной из ключевых особенностей разработанного приложения является гибкость конфигурации, предоставляя пользователям широкий выбор исполнений муфт и возможность настройки параметров, таких как крутящий момент. Интеграция с приложением КОМПАС 3D с использованием API позволяет автоматизировать процесс построения сборок, что делает проект более эффективным и удобным в использовании.

Разработанный интерфейс, включающий в себя дерево конфигураций и диалоговые окна, обеспечивает интуитивно понятное взаимодействие с приложением, делая процесс выбора и настройки муфт понятным и удобным.

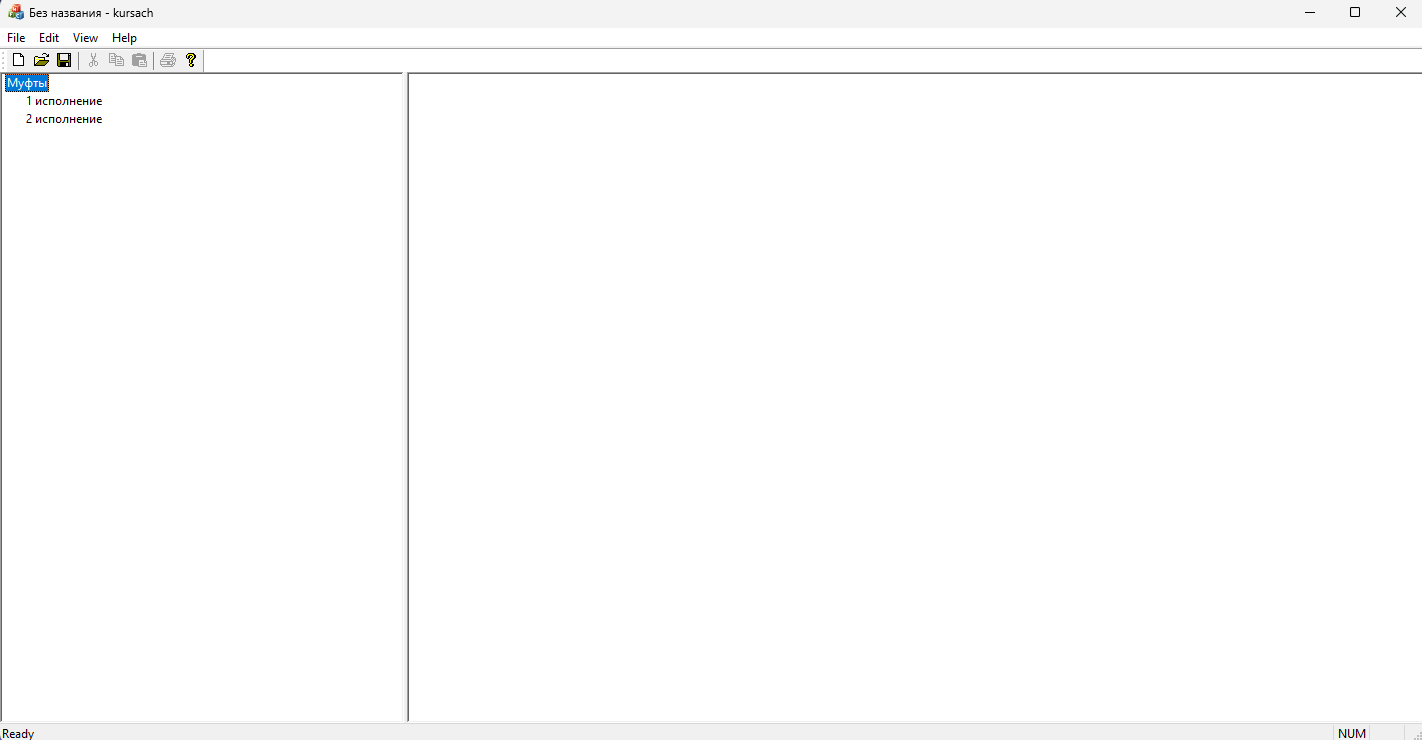
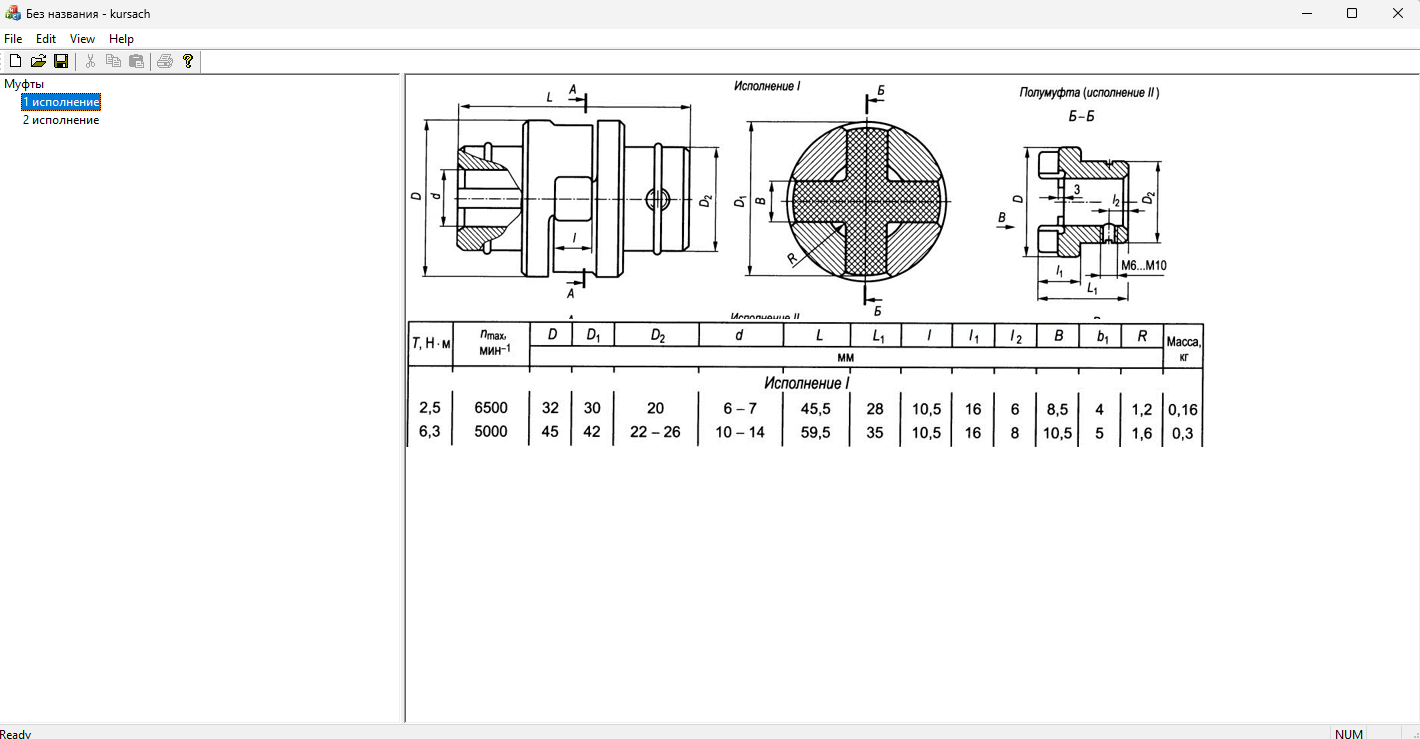
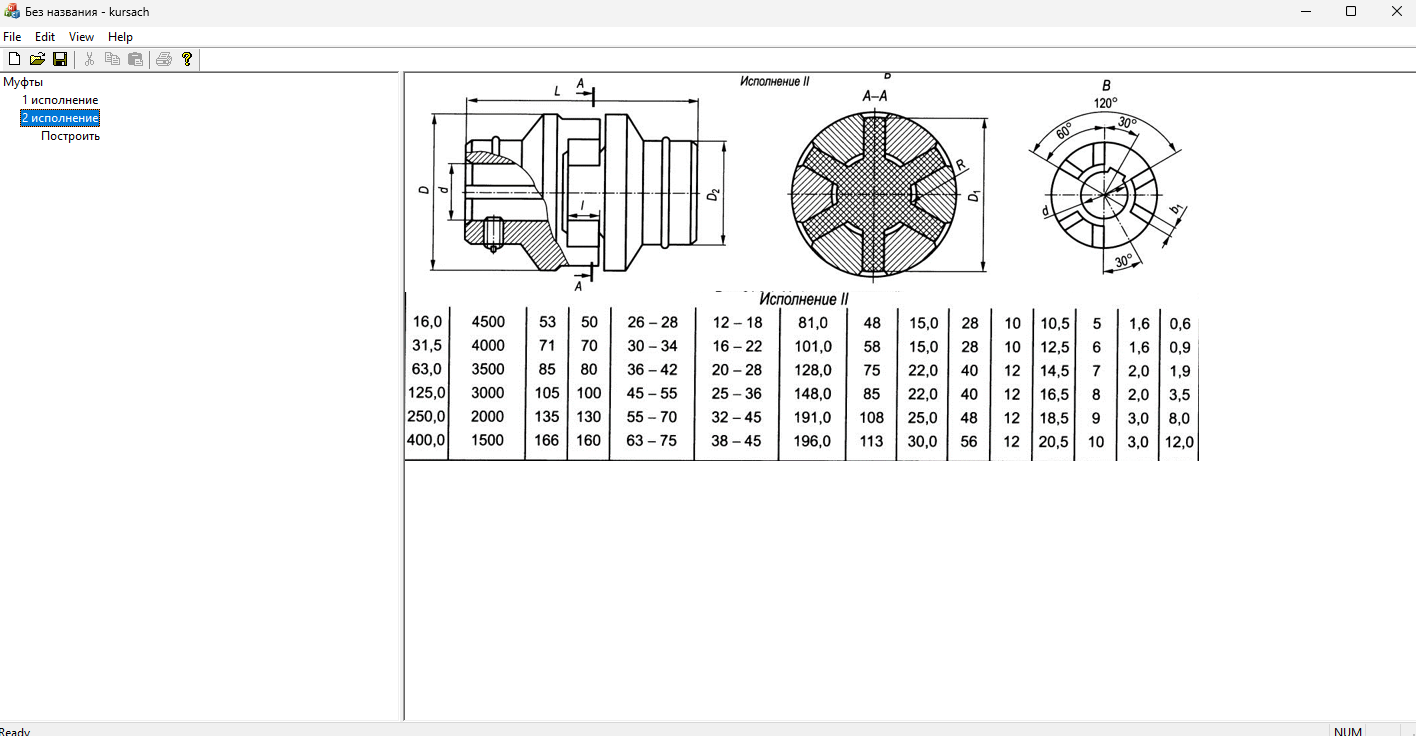
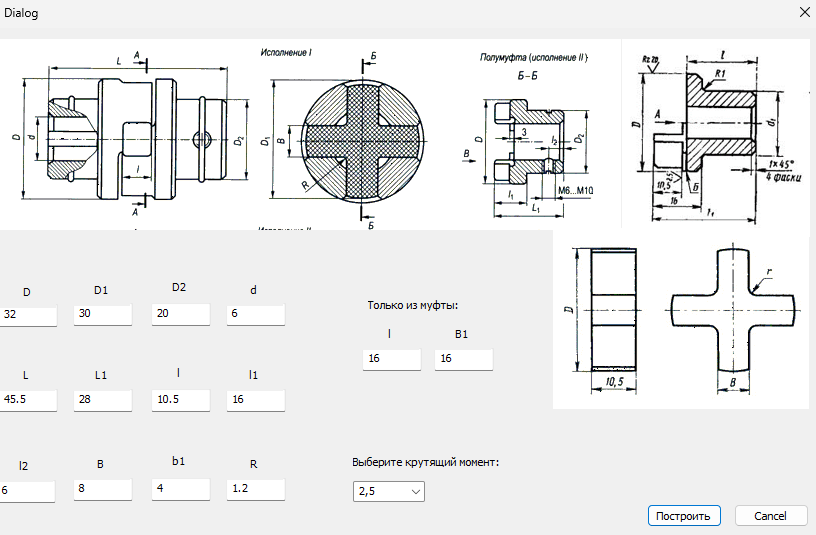
Проект представляет собой важный шаг в области автоматизации проектирования упругих муфт и может быть использован в инженерных приложениях для оптимизации процессов проектирования и моделирования.

В заключение, проделанная работа открывает перспективы для дальнейших исследований и развития в области параметрического проектирования упругих соединений, предоставляя инженерам и конструкторам инструмент для более эффективного и точного проектирования механических систем.

Список литературы

1. *Норсеев Сергей.* Разработка приложений под компас в delphi : 2013. – 346 с.
2. *Детали машин* : атлас конструкций: учеб. пособие для студентов машиностроительных специальностей вузов / Б.А. Байков, В.Н. Богачев, А.В. Буланже и др.; под общ. ред. д-ра техн. наук проф. Д.Н. Решетова. – 5-е изд., переработ. и доп. – Москва : 1992 . –296 с.
3. <https://learn.microsoft.com/ru-ru/cpp/mfc/reference/mfc-classes?view=msvc-170>

Приложения

1. SDI приложение 
2. Первое исполнение 
3. Второе исполнение 
4. Диалоговое окно первого исполнения 
5. Диалоговое окно второго исполнения 