**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 № 1

Группа:   
221-326

Выполнила:   
Арюпина С.Р.

Преподаватель:   
Джунковский А.В.

Дата выполнения:  
20.12.2023

г. Москва

**Оглавление**

[Теоретическая часть 3](#_Toc153611270)

[Практическая часть 6](#_Toc153611271)

[Список литературы: 30](#_Toc153611272)

# ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

**Сальниковое устройство, сальниковое уплотнение** или **сальник** — один из видов уплотнительных устройств подвижных соединений различных устройств и механизмов. Ввиду простоты своей конструкции это одно из самых распространённых и давно известных уплотнительных устройств. Названия сальниковая набивка, сальник, сальниковый узел и другие сохранились с тех времён, когда для уплотнения в этих устройствах использовалась пропитанная жиром пенька, в современной промышленности применяются другие материалы, в первую очередь фторопласты или асбестовое волокно.

Особенно широко сальники используются в трубопроводной арматуре, где они известны как уплотнение подвижных деталей (узлов) арматуры относительно окружающей среды, в котором применён уплотнительный элемент с принудительным созданием в нём напряжений, необходимых для создания требуемой герметичности.

Также сальниковые устройства, работающие на том же принципе, широко применяются в различных промышленных, судовых, автомобильных механизмах. Кроме подвижных узлов, сальники могут использоваться для уплотнения неподвижного оборудования, например в трубных и кабельных проходках.



Рисунок 1 — Виды сальников

Суть сальникового устройства в том, что на внешней стороне крышки или корпуса в том месте, где через них проходит шток или шпиндель, создаётся сальниковая камера (сальниковая коробка, сальница), в которую укладывается уплотнительный материал — сальниковая набивка. При помощи специальных устройств набивка поджимается вдоль оси шпинделя (штока), упираясь в стенки сальниковой камеры и уплотняя набивку. При сжатии набивки в ней создаются усилия, под действием которых она прижимается с одной стороны к стенке сальниковой камеры, а с другой — к цилиндрической поверхности шпинделя (штока). Таким образом создаётся герметичность и рабочая среда не проникает за пределы корпуса оборудования. В механизмах малых диаметров поджатие набивки производится накидной гайкой, больших — крышкой сальника (прижимной втулкой, грундбуксой) — при помощи откидных или анкерных болтов с гайками (обычно двух).

Сила трения, возникающая между сальниковой набивкой и штоком, препятствует последнему совершать необходимые перемещения, а при чрезмерных усилиях затяжки сальника делают их невозможными, поэтому для сальников имеют большое значение конструкторские и технологические решения, обеспечивающие их нормальную работу, среди них:

* материал набивки;
* размеры сальниковой камеры;
* конструкция деталей сальникового узла;
* материал штока (шпинделя), чистота обработки его поверхности и другие.

В некоторых случаях (среди арматуры, как правило, в регулирующих клапанах) для снижения трения применяются сальники со смазкой, которая подаётся извне через специальную маслёнку, в тяжелонагруженных механизмах применяется орошение штока водой, например в буровых насосах.

Современная сальниковая набивка представляет собой, как правило, шнур или кольца из асбеста с графитовой пропиткой. Также используются безасбестовые уплотнительные материалы из фторопласта или на основе графита. Вместо набивки иногда применяются манжеты из резины. Также применяются и многие другие материалы, что определяется конкретными условиями эксплуатации.

Сальники используются в различных промышленных и технических приложениях для предотвращения проникновения или выхода жидкостей, газов или пыли между двумя поверхностями, которые двигаются относительно друг друга. Они являются частью многих механических устройств и машин. Вот несколько областей, где сальники широко применяются:

1. Механизмы двигателей и трансмиссий автомобилей: Сальники используются в различных частях автомобиля, таких как коленчатые валы, распределительные валы, валы передач, чтобы предотвратить утечку масла и других жидкостей.
2. Промышленные насосы и компрессоры: В центробежных насосах, поршневых насосах и компрессорах сальники применяются для предотвращения утечек рабочих жидкостей.
3. Механизмы вращения и подшипники: Сальники могут использоваться в подшипниках и механизмах вращения для защиты от пыли и влаги.
4. Гидравлические и пневматические системы: В гидравлических и пневматических системах сальники предотвращают утечку рабочих жидкостей.
5. Промышленные насосы и компрессоры: Сальники применяются в широком спектре промышленных насосов и компрессоров для предотвращения утечек жидкостей или газов.
6. Электродвигатели: В некоторых электродвигателях используются сальники для предотвращения утечек смазочных материалов.
7. Пищевая и фармацевтическая промышленность: Сальники, соответствующие стандартам, применяются в оборудовании, используемом в пищевой и фармацевтической промышленности, чтобы обеспечивать герметичность и безопасность продукции.

Сальники могут различаться по своей конструкции и материалам в зависимости от конкретного применения.

Существует несколько различных типов сальников, и выбор конкретного типа зависит от условий эксплуатации, типа машины или оборудования, а также от того, какие жидкости или газы необходимо удерживать. Вот несколько основных типов сальников:

1. Одинарные сальники:

Это самый базовый тип сальников, состоящий из одного уплотнительного элемента, который предотвращает утечку между двумя поверхностями. Одинарные сальники просты в конструкции и применяются в различных машинах и механизмах.

1. Двойные сальники:

Двойные сальники включают два уплотнительных элемента, расположенных друг за другом. Между ними создается пространство, в которое может поступать дополнительное давление, что улучшает эффективность сальника. Этот тип часто используется в приложениях с высоким давлением.

1. Сальники с барьерной жидкостью:

Эти сальники включают систему барьерной жидкости, которая предотвращает проникновение вредных веществ в сальник и обеспечивает дополнительное уплотнение. Такие сальники применяются в условиях, где требуется особенно высокая степень герметичности.

1. Сальники с тросовой оплеткой:

Этот тип сальников обычно используется в поворотных механизмах, таких как валы. Он имеет тросовую оплетку вокруг уплотнительного элемента, обеспечивая дополнительную прочность и устойчивость к вращающимся движениям.

1. Сальники на основе материалов:

Существуют различные материалы, используемые для создания уплотнительных элементов сальников, такие как резина, полиуретан, фторопласт и другие полимеры. Выбор материала зависит от температурных условий, типа жидкости или газа, а также химической совместимости.

1. Пружинные сальники:

Эти сальники включают пружину, которая обеспечивает постоянное давление уплотнительного элемента к контактирующей поверхности. Применяются в местах, где нужно компенсировать износ или деформацию.

1. Сальники для высоких и низких температур:

Существуют специальные сальники, адаптированные для экстремальных температурных условий. Это могут быть сальники с теплоизоляцией или специальные материалы, устойчивые к низким или высоким температурам.

Выбор конкретного типа сальника зависит от конкретных требований приложения, и важно учитывать условия эксплуатации, химическую совместимость и другие факторы.

Сальник СКРО – это элемент, применяющийся для ввода кабельных изделий в электрооборудование и аппаратуру водонепроницаемого и герметичного исполнения. С одной стороны в сальник СКРО вводится кабель, а вторая сторона сальника развальцовывается с внутренней стороны электрооборудования.



Рисунок 2 — Сальники кабельные по ГОСТ

Аббревиатура СКРО обозначает: С – сальник, К – кабельный, Р – развальцованный, О – односторонний. Возможные типоразмеры сальников СКРО-16, СКРО-27, СКРО-42, СКРО-60, СКРО-76, СКРО-90. При необходимости, можно изготовить нестандартный типоразмер сальника. Стандарт на сальники: ГОСТ 4860.1-83, ГОСТ 4860.2-83.



Рисунок 3 — Внутреннее устройство сальника

Односторонние сальники типа СКРО представляют собой разнообразную группу уплотнительных устройств. В зависимости от конкретных требований, условий эксплуатации и особенностей применения, существует несколько вариаций односторонних сальников СКРО, предназначенных для различных отраслей и типов технических устройств. Рассмотрим основные виды односторонних сальников типа СКРО:

1. Сальники для насосов. Эти сальники широко применяются в насосных системах для предотвращения утечек рабочих жидкостей. Они обеспечивают герметичное соединение между вращающимся валом насоса и его корпусом, что особенно критично в случае насосов, используемых в химической промышленности или в системах водоснабжения.

2. Сальники для компрессоров. В компрессорных установках односторонние сальники СКРО применяются для предотвращения утечек газов. Они обеспечивают герметичность внутренних камер компрессора, что важно для эффективной работы систем сжатия газов.

3. Сальники для турбин. В энергетической отрасли сальники данного типа используются в турбинах для обеспечения герметичности и предотвращения утечек рабочих сред. Они поддерживают эффективность турбинных установок и предотвращают потери энергии.

4. Сальники для вентиляторов. В системах вентиляции и кондиционирования воздуха односторонние сальники СКРО играют роль в обеспечении герметичности вращающихся вентиляторов. Это важно для предотвращения утечек воздуха и поддержания эффективной работы системы.

5. Сальники для автомобильных двигателей. В автомобильной промышленности эти сальники применяются для уплотнения вращающихся элементов внутри двигателя, предотвращая утечку масел и жидкостей.

6. Сальники для оборудования пищевой промышленности. В оборудовании, используемом в производстве пищевых продуктов, односторонние сальники СКРО обеспечивают герметичность вращающихся частей, предотвращая контаминацию продуктов.

7. Сальники для химической промышленности. В условиях, где существует воздействие агрессивных химических веществ, эти сальники применяются для предотвращения утечек и обеспечения безопасности процессов.

Разнообразие односторонних сальников СКРО позволяет выбирать оптимальный вариант в зависимости от конкретных требований каждого технического приложения. Это делает их востребованными и универсальными элементами в различных отраслях промышленности и техники.

API (Application Programming Interface) КОМПАС-3D представляет собой программный интерфейс, предназначенный для взаимодействия внешних приложений с программным обеспечением САПР (система автоматизированного проектирования) "КОМПАС-3D". Этот мощный инструмент позволяет разработчикам интегрировать и автоматизировать процессы проектирования, управления данными и взаимодействия с геометрическими моделями в программе "КОМПАС-3D".

API КОМПАС-3D позволяет создавать скрипты и приложения для автоматизации повторяющихся задач проектирования. Это может включать в себя создание деталей, сборок, генерацию отчетов и многое другое.

Разработчики могут использовать API для интеграции "КОМПАС-3D" с другими программами и системами, обеспечивая единый поток данных и управление проектами.

API позволяет создавать специализированные приложения, расширяя функциональность "КОМПАС-3D" в соответствии с уникальными потребностями пользователя.

В рамках моей работы я использовал API 5 КОМПАС-3D для реализации автоматизированных процессов проектирования и создания пользовательских приложений.

Применение API 5 существенно ускорило процессы проектирования, уменьшило вероятность ошибок и обеспечило более гибкий и эффективный подход к работе с программой "КОМПАС-3D". Это позволило мне сосредоточиться на более сложных и творческих аспектах моей работы, обеспечивая при этом высокую точность и качество проектирования.

# ПРАКТИЧЕСКАЯ ЧАСТЬ

Цель работы: создать MFC SDI приложение с реализацией построения параметрической сборки.

Задачи: создать параметрическую сборку сальника

Прежде чем перейти к построению, получим запущенный экземпляр Компас 3D с установленной связью с нашим приложением.

Листинг 1 — Подключение к Компасу и создание документа

if (!pKompasApp5) {

CLSID InvAppClsid;

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

if (FAILED(hRes)) {

pKompasApp5 = nullptr;

return;

}

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

if (FAILED(hRes)) {

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;

}

}

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

if (FAILED(hRes)) {

return;

}

}

Далее приступим к построению составляющих сборку деталей (см. Рисунки 5 и 6). Наша сборка (см. Рисунки 7 и 8) будет состоять из одной детали «гнездо» (см. Рисунок 10), двух одинаковых деталей «шайба» (см. Рисунок 9) и одной детали «гайка», разной для двух исполнений (см. Рисунки 11 и 12).

Изображение выглядит как зарисовка, диаграмма, текст, Технический чертеж

Автоматически созданное описание

Рисунок 5 — Чертеж сборки

Изображение выглядит как текст, число, Шрифт, Параллельный

Автоматически созданное описание

Рисунок 6 — Размеры сборки

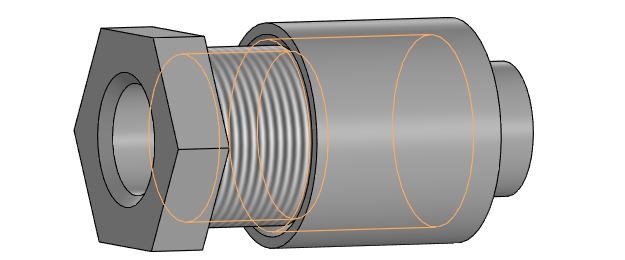


Рисунок 7 — Сборка исполнения 1

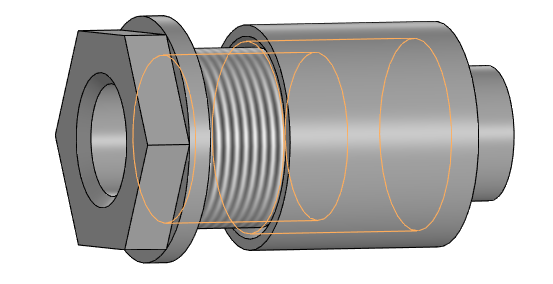


Рисунок 8 — Сборка исполнения 2



Рисунок 9 — Шайба

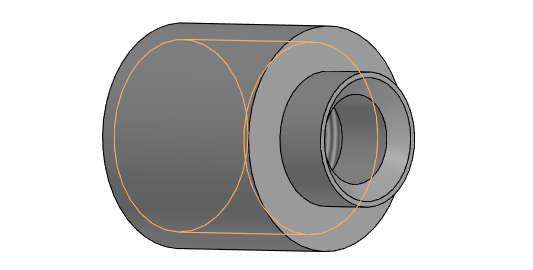


Рисунок 10 — Гнездо

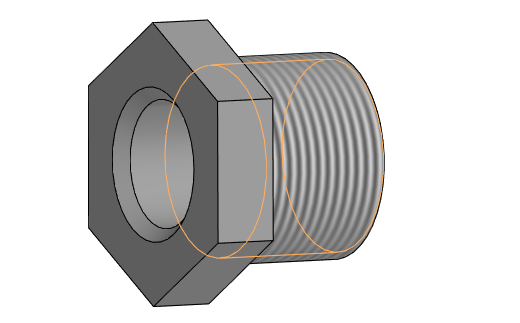


Рисунок 11 — Гайка исполнения 1

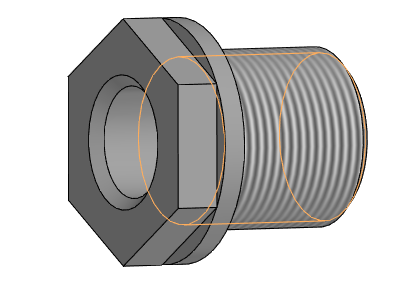


Рисунок 12 — Гайка исполнения 2

Начнем с построения гнезда. Создадим эскиз и выполним для него операцию вращения.

Листинг 2 — Создание эскиза и выдавливание его вращением

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();

double point[8][2];

point[0][0] = 0;

point[0][1] = d2 / 2;

point[1][0] = 0;

point[1][1] = D2 / 2;

point[2][0] = l;

point[2][1] = D2 / 2;

point[3][0] = l;

point[3][1] = D1 / 2;

point[4][0] = l2;

point[4][1] = D1 / 2;

point[5][0] = l2;

point[5][1] = 2;

point[6][0] = l - 2;

point[6][1] = 2;

point[7][0] = l - 2;

point[7][1] = d2 / 2;

p2DDoc->ksLineSeg(point[0][0], point[0][1], point[1][0], point[1][1], 1);

p2DDoc->ksLineSeg(point[1][0], point[1][1], point[2][0], point[2][1], 1);

p2DDoc->ksLineSeg(point[2][0], point[2][1], point[3][0], point[3][1], 1);

p2DDoc->ksLineSeg(point[3][0], point[3][1], point[4][0], point[4][1], 1);

p2DDoc->ksLineSeg(point[4][0], point[4][1], point[5][0], point[5][1], 1);

p2DDoc->ksLineSeg(point[5][0], point[5][1], point[6][0], point[6][1], 1);

p2DDoc->ksLineSeg(point[6][0], point[6][1], point[7][0], point[7][1], 1);

p2DDoc->ksLineSeg(point[7][0], point[7][1], point[0][0], point[0][1], 1);

p2DDoc->ksLineSeg(0, 0, l2, 0, 3);

pSketchDef->EndEdit();

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

ksBossRotatedDefinitionPtr pRotDef = pRotate->GetDefinition();

pRotDef->SetSketch(pSketch);

pRotDef->SetSideParam(TRUE, 360);

pRotate->Create();

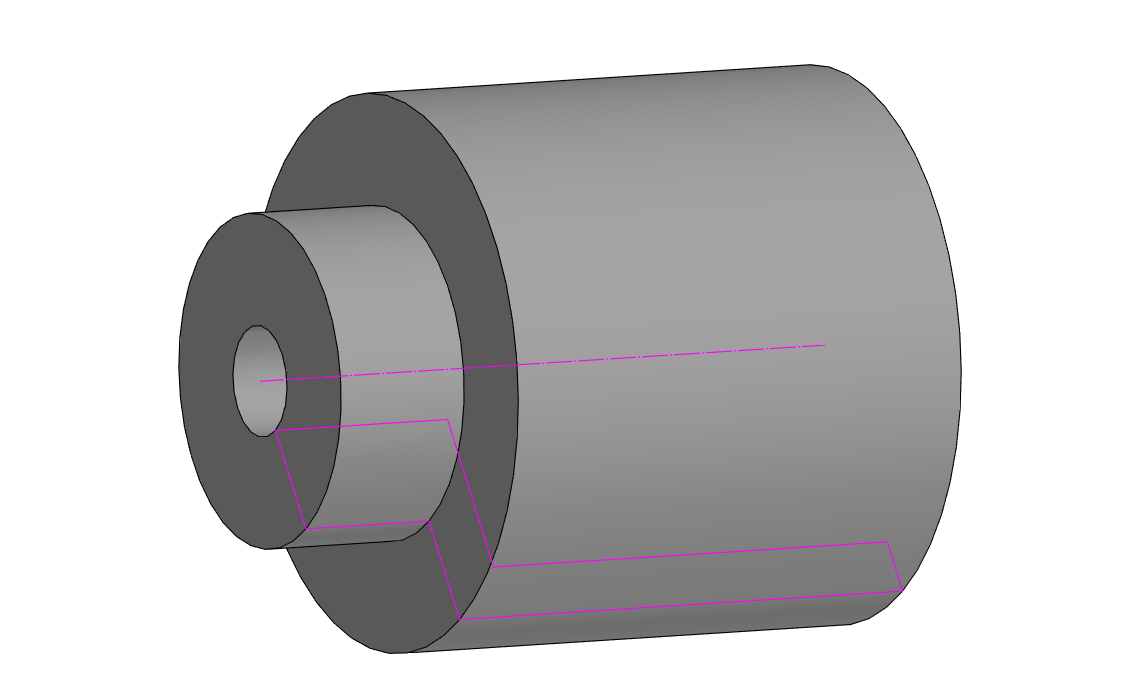


Рисунок 13 — Выдавливание эскиза гнезда

Далее вырежем отверстие в основании гнезда и сразу же применим к нему фаску:

Листинг 3 — Создание эскиза для отверстия, вырезание и добавление фаски

ksEntityPtr pSketch8 = pPart->NewEntity(o3d\_sketch);

pSketchDef = pSketch8->GetDefinition();

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

pSketch8->Create();

p2DDoc = pSketchDef->BeginEdit();

p2DDoc->ksCircle(0, 0, d1 / 2, 1);

pSketchDef->EndEdit();

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

ksCutExtrusionDefinitionPtr pExDef7 = pExtrude6->GetDefinition();

pExDef7->directionType = dtNormal;

pExDef7->SetSketch(pSketch8);

pExDef7->SetSideParam(true, etThroughAll, 0, 0, false);

pExtrude6->Create();

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

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

ksChamferDefinitionPtr pChamferDef = pChamfer->GetDefinition();

pChamferDef->SetChamferParam(true, 1.5, 1.5);

ksEntityCollectionPtr fl = pChamferDef->array();

fl->Clear();

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

{

ksEntityPtr ed = fledges->GetByIndex(i);

ksEdgeDefinitionPtr def = ed->GetDefinition();

ksVertexDefinitionPtr p1 = def->GetVertex(true);

ksVertexDefinitionPtr p2 = def->GetVertex(false);

if (p1 && p2)

{

double x1, y1, z1;

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

if ((int(fabs(x1)) == l2) && (d1/2 == int(fabs(z1)))) {

fl->Add(ed);

}

}

}

pChamfer->Create();

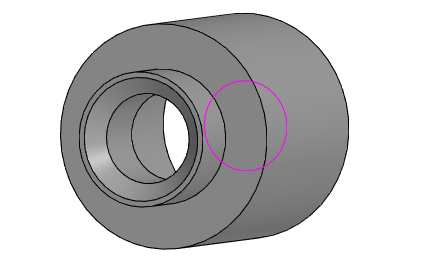


Рисунок 14 — Отверстие и фаска в гнезда

Следующий этап – находим грань для создания фаски на большем отверстии и плоскость для резьбы.

Листинг 4 — Поиск граней, создание резьбы

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

ksEntityPtr pChamfer1 = pPart->NewEntity(o3d\_chamfer);

pChamferDef = pChamfer1->GetDefinition();

pChamferDef = pChamfer1->GetDefinition();

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

fl = pChamferDef->array();

fl->Clear();

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

ksEntityPtr ed = fledges->GetByIndex(i);

ksEdgeDefinitionPtr def = ed->GetDefinition();

ksVertexDefinitionPtr p1 = def->GetVertex(true);

ksVertexDefinitionPtr p2 = def->GetVertex(false);

if (p1 && p2){

double x1, y1, z1;

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

if ((int(fabs(x1)) == l-2) && (int(fabs(y1)) == d2/2) && (0 == int(fabs(z1)))) {

ksFaceDefinitionPtr f = def->GetAdjacentFace(true);

ksEntityPtr face = f->GetEntity();

face->Update();

ksEntityPtr pThread = pPart->NewEntity(o3d\_thread);

ksThreadDefinitionPtr pThDef = pThread->GetDefinition();

pThDef->PutallLength(true);

pThDef->PutautoDefinDr(true);

pThDef->SetBaseObject(face);

pThDef->Putp(2.5);

pThread->Create();

}

if ((0 == fabs(z1)) && (x1 == 0) && (fabs(y1) == D2/2)) {

ksFaceDefinitionPtr f = def->GetAdjacentFace(true);

ksEntityPtr face = f->GetEntity();

face->Putname("nest\_circle");

face->Update();

}

}

if (def->IsCircle()){

ksVertexDefinitionPtr p1 = def->GetVertex(true);

ksVertexDefinitionPtr p2 = def->GetVertex(false);

if (p1 && p2){

double x1, y1, z1;

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

if ((d2 / 2 == fabs(y1)) && (x1 == 0)) {

fl->Add(ed);

}}}

}

pChamfer1->Create();

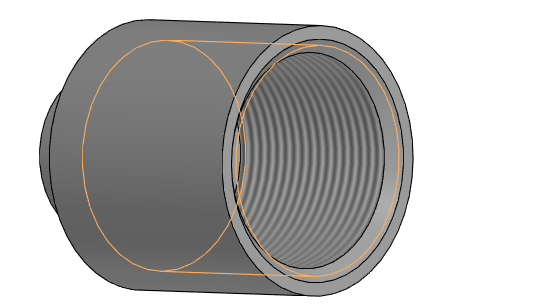


Рисунок 15 — Фаска на большом отверстии и резьба

Последний этап – поиск необходимых граней для дальнейшей сборки, добавление фасок и сохранение готовой детали.

Листинг 5 — Поиск граней в гнезде

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

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

{

ksEntityPtr ed = fledges->GetByIndex(i);

ksEdgeDefinitionPtr def = ed->GetDefinition();

ksVertexDefinitionPtr p1 = def->GetVertex(true);

ksVertexDefinitionPtr p2 = def->GetVertex(false);

if (p1 && p2)

{

double x1, y1, z1;

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

if ((l - 2 == fabs(x1)) && (int(fabs(z1)) == d1/2)) {

ksFaceDefinitionPtr f = def->GetAdjacentFace(true);

ksEntityPtr face = f->GetEntity();

face->Putname("nest\_plane");

face->Update();

}

if ((l2 == fabs(x1)) && (fabs(z1) == 0)) {

ksFaceDefinitionPtr f = def->GetAdjacentFace(true);

ksEntityPtr face = f->GetEntity();

face->Putname("nest\_plane2");

face->Update();

}

}

}

DeleteFile(L"C://Users//sophia//Desktop//kurs//nest.m3d");

pDoc->SaveAs("C://Users//sophia//Desktop//kurs//nest.m3d");

Изображение выглядит как текст, снимок экрана, круг, дизайн

Автоматически созданное описание

Рисунок 16 — Поиск граней в гнезде

Далее переходим к построению первого исполнения гайки. Для этого создаем новый документ, в эскизе строим шестиугольник и выдавливаем.

Листинг 6 — Новый документ, эскиз и выдавливание шестиугольника

Clab1Doc\* pDoc1 = (Clab1Doc\*)((CMainFrame\*)AfxGetMainWnd())->GetActiveDocument();

ASSERT\_VALID(pDoc1);

if (!pDoc1)

return;

if (pDoc1->isp1) {

pDoc = pKompasApp5->Document3D();

pDoc->Create(false, true);

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

ksEntityPtr pSketch3 = pPart->NewEntity(o3d\_sketch);

pSketchDef = pSketch3->GetDefinition();

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

pSketch3->Create();

p2DDoc = pSketchDef->BeginEdit();

ksRegularPolygonParamPtr rect = pKompasApp5->GetParamStruct(ko\_RegularPolygonParam);

rect->ang = 0;

rect->count = 6;

rect->describe = true;

rect->radius = S1 / 2;

rect->style = 1;

rect->xc = 0;

rect->yc = 0;

p2DDoc->ksRegularPolygon(rect, 0);

pSketchDef->EndEdit();

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

ksBaseExtrusionDefinitionPtr pExDef = pExtrude->GetDefinition();

pExDef->directionType = dtNormal;

pExDef->SetSideParam(true, etBlind, 5, 0, false);

pExDef->SetSketch(pSketch3);

pExtrude->Create();

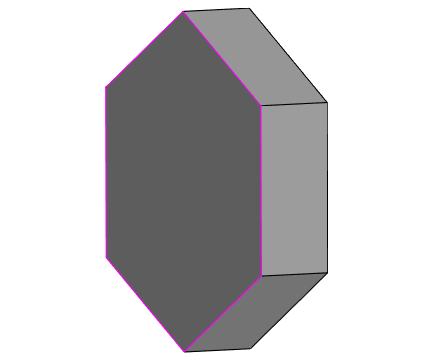


Рисунок 17 — Шестиугольник в первом исполнении гайки

Создаю эскиз под оставшеюся часть изделия и выдавливаю ее вращением.

Листинг 7 — Создание и выдавливание вращением эскиз

ksEntityPtr pSketch2 = pPart->NewEntity(o3d\_sketch);

pSketchDef = pSketch2->GetDefinition();

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

pSketch2->Create();

p2DDoc = pSketchDef->BeginEdit();

double point1[7][2];

point1[0][0] = 0;

point1[0][1] = d / 2;

point1[1][0] = 0;

point1[1][1] = d / 2 + 1;

point1[2][0] = -l1 / 3;

point1[2][1] = d / 2 + 1;

point1[3][0] = -l1 / 3;

point1[3][1] = d2 / 2;

point1[4][0] = -l1 + 1;

point1[4][1] = d2 / 2;

point1[5][0] = -l1;

point1[5][1] = d / 2 + 1;

point1[6][0] = -l1;

point1[6][1] = d / 2;

p2DDoc->ksLineSeg(point1[0][0], point1[0][1], point1[1][0], point1[1][1], 1);

p2DDoc->ksLineSeg(point1[1][0], point1[1][1], point1[2][0], point1[2][1], 1);

p2DDoc->ksLineSeg(point1[2][0], point1[2][1], point1[3][0], point1[3][1], 1);

p2DDoc->ksLineSeg(point1[3][0], point1[3][1], point1[4][0], point1[4][1], 1);

p2DDoc->ksLineSeg(point1[4][0], point1[4][1], point1[5][0], point1[5][1], 1);

p2DDoc->ksLineSeg(point1[5][0], point1[5][1], point1[6][0], point1[6][1], 1);

p2DDoc->ksLineSeg(point1[6][0], point1[6][1], point1[0][0], point1[0][1], 1);

p2DDoc->ksLineSeg(0, 0, -l1, 0, 3);

pSketchDef->EndEdit();

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

pRotDef = pRotate->GetDefinition();

pRotDef->SetSketch(pSketch2);

pRotDef->SetSideParam(TRUE, 360);

pRotate->Create();

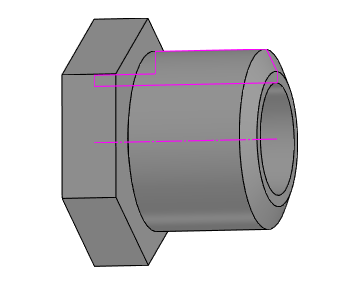


Рисунок 18 — Выдавливание эскиза в первом исполнении гайки

Далее создаю эскиз под отверстие, вырезаю его.

Листинг 8 — Создаю и выдавливаю отверстие

ksEntityPtr pSketch4 = pPart->NewEntity(o3d\_sketch);

pSketchDef = pSketch4->GetDefinition();

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

pSketch4->Create();

p2DDoc = pSketchDef->BeginEdit();

p2DDoc->ksCircle(0, 0, d / 2, 1);

pSketchDef->EndEdit();

ksEntityPtr pExtrude1 = pPart->NewEntity(o3d\_cutExtrusion);

ksCutExtrusionDefinitionPtr pExDef2 = pExtrude1->GetDefinition();

pExDef2->directionType = dtReverse;

pExDef2->SetSketch(pSketch4);

pExDef2->SetSideParam(true, etThroughAll, 0, 0, false);

pExtrude1->Create();

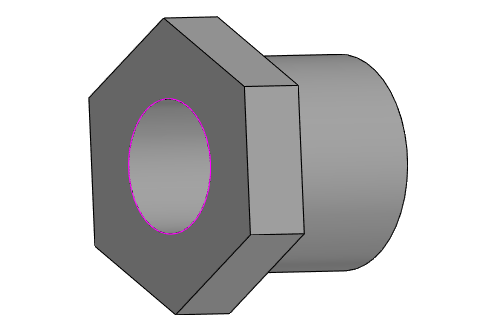


Рисунок 19 — Выдавливание отверстия в первом исполнении гайки

Находим грани для дальнейшей сборки и для фаски, применяем фаску и сохраняем получившуюся деталь.

Листинг 9 — Поиск граней, создание фаски, сохранение документа

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

ksEntityPtr pChamfer2 = pPart->NewEntity(o3d\_chamfer);

pChamferDef = pChamfer2->GetDefinition();

pChamferDef = pChamfer2->GetDefinition();

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

fl = pChamferDef->array();

fl->Clear();

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

ksEntityPtr ed = fledges->GetByIndex(i);

ksEdgeDefinitionPtr def = ed->GetDefinition();

ksVertexDefinitionPtr p1 = def->GetVertex(true);

ksVertexDefinitionPtr p2 = def->GetVertex(false);

if (p1 && p2){

double x1, y1, z1;

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

if ((5 == int(fabs(x1))) && (int(fabs(z1)) == int(S1 / 2))) {

ksFaceDefinitionPtr f = def->GetAdjacentFace(true);

ksEntityPtr face = f->GetEntity();

face->Putname("nut\_plane2");

face->Update();

}}

if (def->IsCircle()){

ksVertexDefinitionPtr p1 = def->GetVertex(true);

ksVertexDefinitionPtr p2 = def->GetVertex(false);

if (p1 && p2){

double x1, y1, z1;

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

if ((d / 2 == fabs(z1)) && (fabs(x1) <= 1)) {

fl->Add(ed);

}

if ((0 == fabs(z1)) && (fabs(x1) == l1)) {

ksFaceDefinitionPtr f = def->GetAdjacentFace(true);

ksEntityPtr face = f->GetEntity();

face->Putname("nut\_circle");

face->Update();

}

if ((d / 2 == fabs(z1)) && (fabs(x1) == l1)) {

ksFaceDefinitionPtr f = def->GetAdjacentFace(true);

ksEntityPtr face = f->GetEntity();

face->Putname("nut\_plane1");

face->Update();

}

if ((d2/2 == fabs(y1)) && (fabs(x1) == l1-1)) {

ksFaceDefinitionPtr f = def->GetAdjacentFace(true);

ksEntityPtr face = f->GetEntity();

face->Update();

ksEntityPtr pThread = pPart->NewEntity(o3d\_thread);

ksThreadDefinitionPtr pThDef = pThread->GetDefinition();

pThDef->PutallLength(true);

pThDef->PutautoDefinDr(true);

pThDef->SetBaseObject(face);

pThDef->Putp(2.5);

pThread->Create();

}

}

}

}

pChamfer2->Create();

DeleteFile(L"C://Users//sophia//Desktop//kurs//nut1.m3d");

pDoc->SaveAs("C://Users//sophia//Desktop//kurs//nut1.m3d");}

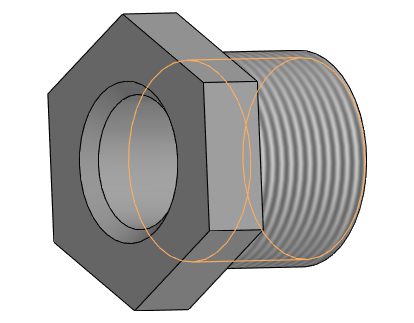


Рисунок 20 — Фаска в первом исполнении гайки

Переходим к созданию второго исполнения для гайки. Для этого создаем новый документ, в эскизе строим шестиугольник и окружность для отверстия, выдавливаем только шестиугольник и получаем готовое основание гайки с отверстием.

Листинг 10 — Создание нового документа, эскиза и выдавливание эскиза

pDoc = pKompasApp5->Document3D();

pDoc->Create(false, true);

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

ksEntityPtr pSketch10 = pPart->NewEntity(o3d\_sketch);

pSketchDef = pSketch10->GetDefinition();

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

pSketch10->Create();

p2DDoc = pSketchDef->BeginEdit();

ksRegularPolygonParamPtr rect1 = pKompasApp5->GetParamStruct(ko\_RegularPolygonParam);

rect1->ang = 0;

rect1->count = 6;

rect1->describe = true;

rect1->radius = S1 / 2;

rect1->style = 1;

rect1->xc = 0;

rect1->yc = 0;

p2DDoc->ksRegularPolygon(rect1, 0);

p2DDoc->ksCircle(0,0, d/2, 1);

pSketchDef->EndEdit();

ksEntityPtr pExtrude10 = pPart->NewEntity(o3d\_baseExtrusion);

ksBaseExtrusionDefinitionPtr pExDef1 = pExtrude10->GetDefinition();

pExDef1->directionType = dtNormal;

pExDef1->SetSideParam(true, etBlind, 4, 0, false);

pExDef1->SetSketch(pSketch10);

pExtrude10->Create();

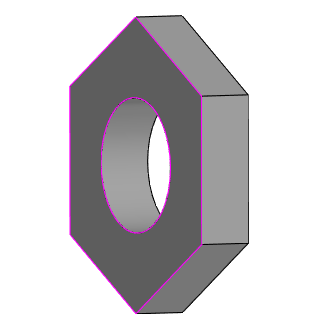


Рисунок 21 — Выдавливание эскиза во втором исполнении гайки

Создаем эскиз для оставшейся части объекта и выдавливаем его вращением.

Листинг 11 — Создание и выдавливание эскиза вращением

ksEntityPtr pSketch11 = pPart->NewEntity(o3d\_sketch);

pSketchDef = pSketch11->GetDefinition();

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

pSketch11->Create();

p2DDoc = pSketchDef->BeginEdit();

double point2[10][2];

point2[0][0] = 0;

point2[0][1] = d / 2;

point2[1][0] = 0;

point2[1][1] = D / 2;

point2[2][0] = 2;

point2[2][1] = D / 2;

point2[3][0] = 2;

point2[3][1] = d2 / 2;

point2[4][0] = 2;

point2[4][1] = d2 / 2;

point2[5][0] = 2;

point2[5][1] = d2 / 2;

point2[6][0] = l1 - 1;

point2[6][1] = d2 / 2;

point2[7][0] = l1;

point2[7][1] = d2 / 2 - 1;

point2[8][0] = l1;

point2[8][1] = d / 2;

point2[9][0] = 0;

point2[9][1] = d / 2;

p2DDoc->ksLineSeg(point2[0][0], point2[0][1], point2[1][0], point2[1][1], 1);

p2DDoc->ksLineSeg(point2[1][0], point2[1][1], point2[2][0], point2[2][1], 1);

p2DDoc->ksLineSeg(point2[2][0], point2[2][1], point2[3][0], point2[3][1], 1);

p2DDoc->ksLineSeg(point2[3][0], point2[3][1], point2[4][0], point2[4][1], 1);

p2DDoc->ksLineSeg(point2[4][0], point2[4][1], point2[5][0], point2[5][1], 1);

p2DDoc->ksLineSeg(point2[5][0], point2[5][1], point2[6][0], point2[6][1], 1);

p2DDoc->ksLineSeg(point2[6][0], point2[6][1], point2[7][0], point2[7][1], 1);

p2DDoc->ksLineSeg(point2[7][0], point2[7][1], point2[8][0], point2[8][1], 1);

p2DDoc->ksLineSeg(point2[8][0], point2[8][1], point2[9][0], point2[9][1], 1);

p2DDoc->ksLineSeg(point2[9][0], point2[9][1], point2[0][0], point2[0][1], 1);

p2DDoc->ksLineSeg(0, 0, l1, 0, 3);

pSketchDef->EndEdit();

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

pRotDef = pRotate->GetDefinition();

pRotDef->SetSketch(pSketch11);

pRotDef->SetSideParam(TRUE, 360);

pRotate->Create();

Изображение выглядит как дизайн

Автоматически созданное описание

Рисунок 22 — Выдавливание эскиза во втором исполнении гайки

Осталось найти грани, добавить фаску, резьбу и сохранить документ.

Листинг 12 — Поиск граней, создание фаски

ksEntityPtr pSketch7 = pPart->NewEntity(o3d\_sketch);

pSketchDef = pSketch7->GetDefinition();

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

pSketch7->Create();

p2DDoc = pSketchDef->BeginEdit();

p2DDoc->ksLineSeg(-4, -d / 2, -3, -d / 2, 1);

p2DDoc->ksLineSeg(-3, -d / 2, -4, -d / 2 - 1, 1);

p2DDoc->ksLineSeg(-4, -d / 2 - 1, -4, -d / 2, 1);

p2DDoc->ksLineSeg(0, 0, -4, 0, 3);

pSketchDef->EndEdit();

ksEntityPtr pCutRotated = pPart->NewEntity(o3d\_cutRotated);

ksCutRotatedDefinitionPtr pCutRotDef = pCutRotated->GetDefinition();

pCutRotDef->SetSketch(pSketch7);

pCutRotDef->SetSideParam(TRUE, 360);

pCutRotated->Create();

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

fl = pChamferDef->array();

fl->Clear();

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

ksEntityPtr ed = fledges->GetByIndex(i);

ksEdgeDefinitionPtr def = ed->GetDefinition();

ksVertexDefinitionPtr p1 = def->GetVertex(true);

ksVertexDefinitionPtr p2 = def->GetVertex(false);

if (p1 && p2){

double x1, y1, z1;

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

if ((4 == int(fabs(x1))) && (fabs(z1) == S1 / 2)) {

ksFaceDefinitionPtr f = def->GetAdjacentFace(true);

ksEntityPtr face = f->GetEntity();

face->Putname("nut2\_plane2");

face->Update();

}}

if (def->IsCircle()){

ksVertexDefinitionPtr p1 = def->GetVertex(true);

ksVertexDefinitionPtr p2 = def->GetVertex(false);

if (p1 && p2){

double x1, y1, z1;

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

if ((l1 == fabs(x1)) && (fabs(y1) == d / 2)) {

ksFaceDefinitionPtr f = def->GetAdjacentFace(true);

ksEntityPtr face = f->GetEntity();

face->Putname("nut2\_circle");

face->Update();

}

if ((d2 / 2 - 1 == int(fabs(y1))) && (fabs(x1) == l1)) {

ksFaceDefinitionPtr f = def->GetAdjacentFace(true);

ksEntityPtr face = f->GetEntity();

face->Putname("nut2\_plane1");

face->Update();

}

if ((d2 / 2 == fabs(y1)) && (fabs(x1) == 2) && (fabs(z1) == 0)) {

ksFaceDefinitionPtr f = def->GetAdjacentFace(true);

ksEntityPtr face = f->GetEntity();

face->Update();

ksEntityPtr pThread = pPart->NewEntity(o3d\_thread);

ksThreadDefinitionPtr pThDef = pThread->GetDefinition();

pThDef->PutallLength(true);

pThDef->PutautoDefinDr(true);

pThDef->SetBaseObject(face);

pThDef->Putp(2.5);

pThread->Create();

}

}

}

}

DeleteFile(L"C://Users//sophia//Desktop//kurs//nut2.m3d");

pDoc->SaveAs("C://Users//sophia//Desktop//kurs//nut2.m3d");

Изображение выглядит как снимок экрана, текст, диаграмма, круг

Автоматически созданное описание

Рисунок 23 — Поиск граней во втором исполнении гайки

Осталось создать последнюю деталь – шайбу. Для этого создаем новую деталь, эскиз с окружностью и выдавливаем его.

Листинг 13 — Создание нового документа, эскиза и выдавливание окружности

pDoc = pKompasApp5->Document3D();

pDoc->Create(false, true);

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

ksEntityPtr pSketch5 = pPart->NewEntity(o3d\_sketch);

pSketchDef = pSketch5->GetDefinition();

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

pSketch5->Create();

p2DDoc = pSketchDef->BeginEdit();

p2DDoc->ksCircle(0, 0, D3 / 2, 1);

p2DDoc->ksCircle(0, 0, d3 / 2, 1);

pSketchDef->EndEdit();

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

ksBaseExtrusionDefinitionPtr pExDef3 = pExtrude2->GetDefinition();

pExDef3->directionType = dtNormal;

pExDef3->SetSideParam(true, etBlind, S, 0, false);

pExDef3->SetSketch(pSketch5);

pExtrude2->Create();

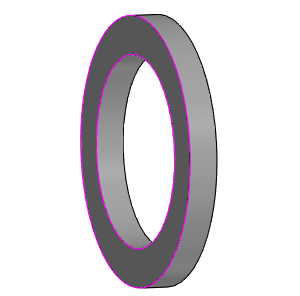


Рисунок 24 — Создание эскиза для шайбы

Находим грани для сборки и сохраняем документ.

Листинг 14 — Поиск и инициализация граней, создание фаски, сохранение документа.

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() == pExtrude3)

{

face->Putname("whasher\_circle");

face->Update();

}

}

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

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

{

ksEntityPtr ed = fledges->GetByIndex(i);

ksEdgeDefinitionPtr def = ed->GetDefinition();

ksVertexDefinitionPtr p1 = def->GetVertex(true);

ksVertexDefinitionPtr p2 = def->GetVertex(false);

if (p1 && p2)

{

double x1, y1, z1;

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

if ((0 == fabs(y1)) && (fabs(z1) == D3 / 2)) {

ksFaceDefinitionPtr f = def->GetAdjacentFace(true);

ksEntityPtr face = f->GetEntity();

face->Putname("whasher\_plane");

face->Update();

}

}

}

DeleteFile(L" C://Users//sophia//Desktop//kurs//washer.m3d");

pDoc->SaveAs("C://Users//sophia//Desktop//kurs//washer.m3d");

Изображение выглядит как текст, снимок экрана, круг, программное обеспечение

Автоматически созданное описание

Рисунок 25 — Поиск граней в шайбе

Теперь перейдем непосредственно к сборке построенных выше деталей. Для этого создадим новый документ соответствующего типа и загрузим в него один экземпляр детали гнездо, гайку в соответствии с выбранным исполнением и два экземпляра детали шайба.

Листинг 15 — Загрузка сохранённых деталей в файл сборки в соответствии с исполнением

pDoc = pKompasApp5->Document3D();

pDoc->Create(false, false);

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

ksPartPtr pnest, pscrew, pwasher1, pwasher2;

pDoc->SetPartFromFile("C://Users//sophia//Desktop//kurs//nest.m3d", pPart, true);

if (pDoc1->isp1) {

pDoc->SetPartFromFile("C://Users//sophia//Desktop//kurs//nut1.m3d", pPart, true);

}

if (pDoc1->isp2) {

pDoc->SetPartFromFile("C://Users//sophia//Desktop//kurs//nut2.m3d", pPart, true);

}

pDoc->SetPartFromFile("C://Users//sophia//Desktop//kurs//washer.m3d", pPart, true);

pDoc->SetPartFromFile("C://Users//sophia//Desktop//kurs//washer.m3d", pPart, true);

pnest = pDoc->GetPart(0);

pscrew = pDoc->GetPart(1);

pwasher1 = pDoc->GetPart(2);

pwasher2 = pDoc->GetPart(3);

Изображение выглядит как цилиндр, металл

Автоматически созданное описание

Рисунок 26 — Загрузка сохранённых деталей в сборку

Листинг 16 — Создание зависимостей для выбранного исполнения, обновление документа

ksEntityCollectionPtr col = pnest->EntityCollection(o3d\_face);

ksEntityPtr nest\_alignment = col->GetByName("nest\_circle", true, true);

ksEntityPtr nest\_4\_coincidence1 = col->GetByName("nest\_plane", true, true);

ksEntityPtr nest\_4\_coincidence2 = col->GetByName("nest\_plane2", true, true);

col = pwasher1->EntityCollection(o3d\_face);

ksEntityPtr washer1\_alignment = col->GetByName("whasher\_circle", true, true);

ksEntityPtr washer1\_4\_coincidence = col->GetByName("whasher\_plane", true, true);

col = pwasher2->EntityCollection(o3d\_face);

ksEntityPtr washer2\_alignment = col->GetByName("whasher\_circle", true, true);

ksEntityPtr washer2\_4\_coincidence = col->GetByName("whasher\_plane", true, true);

pDoc->AddMateConstraint(mc\_Concentric,washer1\_alignment, nest\_alignment,1,1, 0);

pDoc->AddMateConstraint(mc\_Concentric,washer2\_alignment, nest\_alignment,1,1, 0);

if (pDoc1->isp1) {

col = pscrew->EntityCollection(o3d\_face);

ksEntityPtr nut\_alignment = col->GetByName("nut\_circle", true, true);

ksEntityPtr nut\_4\_coincidence1 = col->GetByName("nut\_plane1", true, true);

ksEntityPtr nut\_4\_coincidence2 = col->GetByName("nut\_plane2", true, true);

pDoc->AddMateConstraint(mc\_Concentric,nut\_alignment,nest\_alignment,-1,1,0);

pDoc->AddMateConstraint(mc\_Distance,nut\_4\_coincidence1,washer2\_4\_coincidence,1,1,S);

pDoc->AddMateConstraint(mc\_Coincidence,nest\_4\_coincidence1,washer1\_4\_coincidence, -1, 1, 0);

pDoc->AddMateConstraint(mc\_Distance, nut\_4\_coincidence2,nest\_4\_coincidence2, 1, 1, L-5);

}

if (pDoc1->isp2) {

col = pscrew->EntityCollection(o3d\_face);

ksEntityPtr nut2\_alignment = col->GetByName("nut2\_circle", true, true);

ksEntityPtr nut2\_4\_coincidence1 = col->GetByName("nut2\_plane1",true,true);

ksEntityPtr nut2\_4\_coincidence2 = col->GetByName("nut2\_plane2",true,true);

pDoc->AddMateConstraint(mc\_Concentric, nut2\_alignment, nest\_alignment, -1, 1, 0);

pDoc->AddMateConstraint(mc\_Distance, nut2\_4\_coincidence1,washer2\_4\_coincidence, 1, 1, S);

pDoc->AddMateConstraint(mc\_Coincidence, nest\_4\_coincidence1,washer1\_4\_coincidence, -1, 1, 0);

pDoc->AddMateConstraint(mc\_Distance, nut2\_4\_coincidence2,nest\_4\_coincidence2, -1, 1, -L);

}

pDoc->RebuildDocument();

DeleteFile(L"C://Users//sophia//Desktop//kurs//oilseal.a3d");

pDoc->SaveAs("C://Users//sophia//Desktop//kurs//oilseal.a3d");

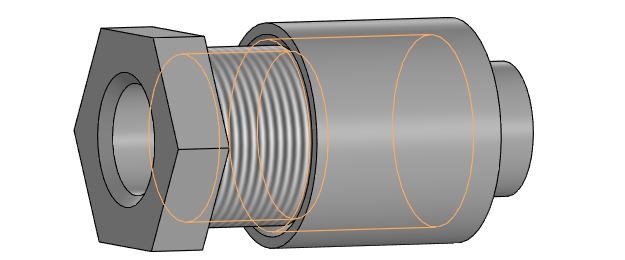


Рисунок 27 — Сборка с гайкой исполнения 1

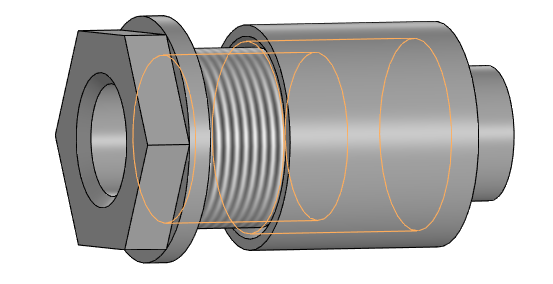


Рисунок 28 — Сборка с гайкой исполнения 2

# СПИСОК ЛИТЕРАТУРЫ:

1. *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.
2. *Норсеев Сергей.* Разработка приложений под компас в delphi : 2013. – 346 с.
3. *ООО «уса».* Сальники одинарного и двойного действия : сайт. – Москва, 2023– . – URL: <https://goo.su/wbvn> (дата обращения 16.12.2023).
4. *Прогресс-м.* Муфты – сальники сксо-16, сксо-27, сксо-42, сксо-60, сксо-76, сксо-90 гост 4860.1–83, гост 4860.2-83 : сайт. – Москва, 2023. – URL: https://prog-m.ru/salniki-skso/ (дата обращения 16.12.2023).
5. *Сиал*. Манжета армированная (сальник) 190х230х15 : сайт. – Москва, 2023– . – URL: [https://goo.su/emn3qy](https://goo.su/EMn3Qy) (дата обращения 16.12.2023).
6. *Импульс*. Сальники - назначение, монтаж и эксплуатация : сайт. – Москва, 2023– . – URL: <https://impuls-ek.ru/info/articles/salniki_naznachenie_montazh_i_ekspluatatsiya/> (дата обращения 16.12.2023).
7. *Nvph*. Сальники и армированные манжеты : сайт. – Москва, 2023– . – URL: https://nvph.ru/slaniki-manjeti/ (дата обращения 16.12.2023).