**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:   
Шумилов Р.Э.

Преподаватель:   
Джунковский А.В.

Дата выполнения:  
16.12.2023

**Оглавление**

[Теоретическая часть 3](#_Toc153611270)

[Практическая часть 6](#_Toc153611271)

[Список литературы: 30](#_Toc153611272)

# ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

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.   
  
вставить где применяется

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

Цели и задачи.

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)//проверка повторное ли подключение кнопки

{

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;

}

}

// Получаем интерфейс приложения Компас

hRes = pKompasAppUnk->QueryInterface(\_\_uuidof(KompasObject), (void\*\*)&pKompasApp5);//получем указатель на компасобджект

if (FAILED(hRes)) {

return;

}

}

Далее приступим к построению составляющих сборку деталей. Наша сборка будет состоять из, одной детали типа «гнездо», двумя одинаковыми деталями типа «шайба» и одной детали типа «гайка», разными для двух исполнений.

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

Изображение выглядит как цилиндр, металл, дизайн

Автоматически созданное описание

Рисунок 7 — Исполнение 1

Изображение выглядит как цилиндр, металл, соединитель

Автоматически созданное описание

Рисунок 8 — Исполнение 2

Изображение выглядит как объектив, круг, кольцо

Автоматически созданное описание

Рисунок 9 — Шайба

Изображение выглядит как цилиндр, круг, дизайн

Автоматически созданное описание

Рисунок 10 — Гнездо

Изображение выглядит как дизайн

Автоматически созданное описание

Рисунок 11 — Гайка исполнение 1

Изображение выглядит как цилиндр, автокомпонент, круг, часть

Автоматически созданное описание

Рисунок 12 — Гайка исполнение 2

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

Листинг 2 — Создание эскиза и выдавливание его вращением

//создание документа

pDoc = pKompasApp5->Document3D();

pDoc->Create(false, true);

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

//эскиз общий

ksEntityPtr pSketch = pPart->NewEntity(o3d\_sketch);//эскиз для 3д детали

ksSketchDefinitionPtr pSketchDef = pSketch->GetDefinition();

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

pSketch->Create();//открываем на редактирование

ksDocument2DPtr p2DDoc = pSketchDef->BeginEdit();

//массив для точек в эскизе

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;

//соединяю точки

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(-10, 0, 10, 0, 3);//рисую осевую

pSketchDef->EndEdit();//закрыли эскиз

//выдавливаю эскиз вращением

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

ksBossRotatedDefinitionPtr pRotDef = pRotate->GetDefinition();

pRotDef->SetSketch(pSketch);

pRotDef->SetSideParam(TRUE, 360);

pRotate->Create();//создали

Изображение выглядит как цилиндр, круг, дизайн

Автоматически созданное описание

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

//эскиз под малое отверстие

ksEntityPtr pSketch8 = pPart->NewEntity(o3d\_sketch);//эскиз для 3д детали

pSketchDef = pSketch8->GetDefinition();

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

pSketch8->Create();//открываем на редактирование

p2DDoc = pSketchDef->BeginEdit();//устанавливаем режим редактирования

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

pSketchDef->EndEdit();//закрыл эскиз

//вырезаю отверстие

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();

//фаска

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++)//цикл по всем граням

{

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();

Изображение выглядит как круг, дизайн

Автоматически созданное описание

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);//создаем коллекцию граней

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++)//цикл по всем граням{

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");//для соосности гнезда

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();

Изображение выглядит как круг, дизайн

Автоматически созданное описание

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);//создаем коллекцию граней

for (int i = 0; i < fledges->GetCount(); i++)//цикл по всем граням

{

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");//совпадение шайбы и гнезда

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\\Leonid\\Desktop\\POLITEXXX\\cem2\\proga\\kursovichok\\Kcreate\\nest.m3d");

pDoc->SaveAs("C:\\Users\\Leonid\\Desktop\\POLITEXXX\\cem2\\proga\\kursovichok\\Kcreate\\nest.m3d");

Изображение выглядит как текст, снимок экрана, круг, дизайн

Автоматически созданное описание

Рисунок 16 — Поиск граней в гнезде

Далее перехожу к построению первого исполнения гайки. Для этого создаю новый документ, а в нем эскиз, в эскизе строю шестиугольник и выдавливаю последний.

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();//открываем на редактирование

p2DDoc = pSketchDef->BeginEdit();

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

rect->ang = 0;//без наклона

rect->count = 6;//шестиугольник

rect->describe = true;

rect->radius = S1 / 2;//радиус вписанной окружности

rect->style = 1;//основной стиль линии

rect->xc = 0;//координаты центра вписанной окружности

rect->yc = 0;

p2DDoc->ksRegularPolygon(rect, 0);

pSketchDef->EndEdit();//закрыли искиз

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

ksBaseExtrusionDefinitionPtr pExDef = pExtrude->GetDefinition();

pExDef->directionType = dtNormal;//выбираем тип выдавливания

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

pExDef->SetSketch(pSketch3);//какой эскиз будем давить

pExtrude->Create();//создали

Изображение выглядит как дизайн

Автоматически созданное описание

Рисунок 17 — Шестиугольник в первом исполнении гайки

Создаю эскиз под оставшеюся часть изделия и выдавливаю ее вращением.

Листинг 7 — Создаю и выдавливаю вращением эскиз

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

pSketchDef = pSketch2->GetDefinition();

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

pSketch2->Create();//открываем на редактирование

p2DDoc = pSketchDef->BeginEdit();//устанавливаем режим редактирования

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;//здесь задаем значение фаски

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

point1[5][0] = -l1;

point1[5][1] = d / 2 + 1;//здесь задаем значение фаски

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(-10, 0, 10, 0, 3);//рисую осевую

pSketchDef->EndEdit();//закрыли искиз

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

pRotDef = pRotate->GetDefinition();

pRotDef->SetSketch(pSketch2);

pRotDef->SetSideParam(TRUE, 360);

pRotate->Create();//создали

Изображение выглядит как дизайн

Автоматически созданное описание

Рисунок 18 — Выдавливание эскиза в первом исполнении гайки

Далее создаю эскиз под отверстие, вырезаю его.

Листинг 8 — Создаю и выдавливаю отверстие

//эскиз для отверстия

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

pSketchDef = pSketch4->GetDefinition();

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

pSketch4->Create();//открываем на редактирование

p2DDoc = pSketchDef->BeginEdit();

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

pSketchDef->EndEdit();//закрыл эскиз

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();

Изображение выглядит как дизайн, круг

Автоматически созданное описание

Рисунок 19 — Выдавливание отверстия в первом исполнении гайки

Нахожу грани для дальнейшей сборки и для фаски, применяю фаску и сохраняю получившуюся деталь.

Листинг 9 — Поиск граней, создание фаски, сохранение документа

fledges = pPart->EntityCollection(o3d\_edge);//создаем коллекцию граней

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");//для соосности с гнездом

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\\Leonid\\Desktop\\POLITEXXX\\cem2\\proga\\kursovichok\\Kcreate\\nut1.m3d");

pDoc->SaveAs("C:\\Users\\Leonid\\Desktop\\POLITEXXX\\cem2\\proga\\kursovichok\\Kcreate\\nut1.m3d");}

Изображение выглядит как снимок экрана, круг, дизайн

Автоматически созданное описание

Рисунок 20 — Фаска в первом исполнении гайки

Перехожу к созданию второго исполнения для гайки. Создаю новый документ, эскиз, в нем строю шестиугольник и окружность для отверстия, выдавливаю только шестиугольник и получаю 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);

//искиз для шестиугольного основани

ksEntityPtr pSketch10 = pPart->NewEntity(o3d\_sketch);//эскиз для 3д детали

pSketchDef = pSketch10->GetDefinition();

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

pSketch10->Create();//открываем на редактирование

p2DDoc = pSketchDef->BeginEdit();//устанавливаем режим редактирования

//строю 6-ти угольник

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

rect1->ang = 0;//без наклона

rect1->count = 6;//шестиугольник

rect1->describe = true;

rect1->radius = S1 / 2;//радиус вписанной окружности

rect1->style = 1;//основной стиль линии

rect1->xc = 0;//координаты центра вписанной окружности

rect1->yc = 0;

p2DDoc->ksRegularPolygon(rect1, 0);

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

pSketchDef->EndEdit();//закрыли искиз

//выдавливаю шестиугольник

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

ksBaseExtrusionDefinitionPtr pExDef1 = pExtrude10->GetDefinition();

pExDef1->directionType = dtNormal;//выбираем тип выдавливания

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

pExDef1->SetSketch(pSketch10);//какой эскиз будем давить

pExtrude10->Create();//создали

Изображение выглядит как круг, дизайн

Автоматически созданное описание

Рисунок 21 — Выдавливание эскиза во втором исполнении гайки

Создаю эскиз для оставшейся части объекта и выдавливаю его вращением.

Листинг 11 — Создание и выдавливание эскиза вращением

ksEntityPtr pSketch11 = pPart->NewEntity(o3d\_sketch);//эскиз для 3д детали

pSketchDef = pSketch11->GetDefinition();

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

pSketch11->Create();//открываем на редактирование

p2DDoc = pSketchDef->BeginEdit();//устанавливаем режим редактирования

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;//толщина окружности

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

point2[3][0] = 2;//толщина окружности

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);//рисую осевую

pSketchDef->EndEdit();//закрыли эскиз

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

fledges = pPart->EntityCollection(o3d\_edge);//создаем коллекцию граней

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");//для соосности с гнездом

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\\Leonid\\Desktop\\POLITEXXX\\cem2\\Kcreate\\nut2.m3d");

pDoc->SaveAs("C:\\Users\\Leonid\\Desktop\\POLITEXXX\\cem2 \\Kcreate\\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 создаю новую деталь, эскиз с окружностью и выдавливаю 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);//эскиз для 3д детали

pSketchDef = pSketch5->GetDefinition();

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

pSketch5->Create();//открываем на редактирование

p2DDoc = pSketchDef->BeginEdit();//устанавливаем режим редактирования

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

pSketchDef->EndEdit();// закрыл эскиз

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

ksBaseExtrusionDefinitionPtr pExDef3 = pExtrude2->GetDefinition();

pExDef3->directionType = dtNormal;//выбираем тип выдавливания

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

pExDef3->SetSketch(pSketch5);//какой эскиз будем давить

pExtrude2->Create();//создали

Изображение выглядит как круг, Графика

Автоматически созданное описание

Рисунок 24 — Создание эскиза для шайбы

Создаю эскиз для отверстия и вырезаю его. Сразу же нахожу грани для сборки и сохраняю документ.

Листинг 14 поиск и инициализация граней, создание фаски, сохранение документа.

ksEntityPtr pSketch6 = pPart->NewEntity(o3d\_sketch);//искиз для 3д детали

pSketchDef = pSketch6->GetDefinition();

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

pSketch6->Create();//открываем на редактирование

p2DDoc = pSketchDef->BeginEdit();//устанавливаем режим редактирования

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

pSketchDef->EndEdit();

//вырезаю отверстие

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

ksCutExtrusionDefinitionPtr pExDef4 = pExtrude3->GetDefinition();

pExDef4->directionType = dtReverse;

pExDef4->SetSketch(pSketch6);

pExDef4->SetSideParam(true, etThroughAll, 0, 0, false);

pExtrude3->Create();

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

for (int i = 0; i < flFaces->GetCount(); i++)//цикл по всем граням

{

ksEntityPtr face = flFaces->GetByIndex(i);

ksFaceDefinitionPtr def = face->GetDefinition();

if (def->GetOwnerEntity() == pExtrude3)

{

face->Putname("whasher\_circle");//присваиваем имя грани

face->Update();//обновляем

}

}

fledges = pPart->EntityCollection(o3d\_edge);//создаем коллекцию граней

for (int i = 0; i < fledges->GetCount(); i++)//цикл по всем граням

{

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\\Leonid\\Desktop\\POLITEXXX\\cem2\\proga\\kursovichok\\Kcreate\\washer.m3d");

pDoc->SaveAs("C:\\Users\\Leonid\\Desktop\\POLITEXXX\\cem2\\proga\\kursovichok\\Kcreate\\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\\Leonid\\Desktop\\POLITEXXX\\cem2\\proga\\kursovichok\\Kcreate\\nest.m3d", pPart, true);

if (pDoc1->isp1) {

pDoc->SetPartFromFile("C:\\Users\\Leonid\\Desktop\\POLITEXXX\\cem2\\proga\\kursovichok\\Kcreate\\nut1.m3d", pPart, true);//1е исполнение гайки

}

if (pDoc1->isp2) {

pDoc->SetPartFromFile("C:\\Users\\Leonid\\Desktop\\POLITEXXX\\cem2\\proga\\kursovichok\\Kcreate\\nut2.m3d", pPart, true);//2е исполнение гайки

}

pDoc->SetPartFromFile("C:\\Users\\Leonid\\Desktop\\POLITEXXX\\cem2\\proga\\kursovichok\\Kcreate\\washer.m3d", pPart, true);

pDoc->SetPartFromFile("C:\\Users\\Leonid\\Desktop\\POLITEXXX\\cem2\\proga\\kursovichok\\Kcreate\\washer.m3d", pPart, true);

pnest = pDoc->GetPart(0);

pscrew = pDoc->GetPart(1); //исполнение гайки

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();//обновляем документ

Изображение выглядит как цилиндр, металл, дизайн

Автоматически созданное описание

Рисунок 27 — Сборка с гайкой исполнения 1

Изображение выглядит как цилиндр, металл, соединитель

Автоматически созданное описание

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).