ТЕМА	Создание параметрической сборки «Втулочная муфта» с использованием API Компас-3D
ПРАКТИЧЕСКИЙ РЕЗУЛЬТАТ	
Назначение	Автоматизация процесса проектирования и сборки компонентов муфты с использованием Компас-3D.
Основные функции	• Генерация параметрической сборки "Втулочная муфта" с шестью конфигурациями. • Управление исполнениями с помощью удобного интерфейса.
Используемые технологии и платформы	• Язык программирования: C++. • Библиотека: Microsoft Foundation Classes (MFC). • Среда разработки: Microsoft Visual Studio. • CAD-система: Компас-3D (использование API).
ВЫПОЛНЕНИЕ РАБОТЫ	
Решаемые задачи	• Реализация интерфейса для выбора конфигураций сборки с использованием MFC. • Создание конфигуратора для построения параметрической сборки с помощью API Компас-3D. • Написание методов для построения деталей и сборк
Состав технической документации	• Руководство пользователя. • Программная спецификация (описание структуры кода и взаимодействия модулей). • Инструкция по интеграции с Компас-3D.

Состав графической части	• Иллюстрации интерфейса (главное окно приложения) — 8 рисунков. • Листинги—15 • Отдельные чертежи деталей (вал, втулка, штифт).

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Современные задачи в области машиностроения требуют эффективных методов проектирования, которые обеспечивают точность и гибкость работы с параметрическими моделями. При этом интеграция специализированного программного обеспечения, такого как Компас-3D, с пользовательскими приложениями становится важным направлением для автоматизации инженерных процессов. Однако стандартные инструменты Компас-3D не всегда обеспечивают удобство и скорость работы при проектировании сложных сборок, таких как "Втулочная муфта". Это делает актуальной разработку специализированного приложения, позволяющего упростить и ускорить процесс создания параметрических моделей.

Проблема: процесс настройки параметров сборок в КОМПАС-3D зачастую носит рутинный характер и требует значительных усилий, что снижает эффективность работы конструкторов и подчеркивает необходимость в автоматизации.

Цель работы: разработать Windows-приложение на базе MFC SDI, взаимодействующее с API Компас-3D, для автоматизированного построения параметрической сборки "Втулочная муфта".

Задачи работы:

1. Изучить принципы работы API Компас-3D и инструменты разработки MFC SDI.
2. Реализовать функции для построения параметрической сборки "Втулочная муфта" с различными конфигурациями.
3. Разработать интерфейс с таблицей выбора конфигураций и древовидным списком деталей.
4. Осуществить тестирование приложения и оценить его производительность.

Практическая ценность работы заключается в создании программного продукта, который автоматизирует процесс построения параметрических

моделей, снижает вероятность ошибок и упрощает взаимодействие пользователя с Компас-3D.

Содержание работы:

В теоретическом разделе рассматриваются особенности конструкции "Втулочной муфты", основные принципы работы с MFC SDI, а также возможности API Компас-3D для создания параметрических моделей.

- В практическом разделе описан процесс реализации приложения: создание интерфейса, работа с деревом компонентов, интеграция с API Компас-3D и разработка методов построения компонентов сборки.
- В заключении подведены итоги работы, оценены достигнутые результаты и предложены пути дальнейшего развития проекта.

Данный курсовой проект предоставляет удобное решение для проектирования параметрических сборок и может быть использован как основа для создания более сложных инструментов автоматизации проектирования.

1 ОБЗОР НАУЧНО-ТЕХНИЧЕСКОЙ ЛИТЕРАТУРЫ

Автоматизация проектирования началась с появления первых CAD-систем в 1960–1970 годах. Одним из ключевых этапов стало создание средств программной интеграции, таких как API, предоставляющих разработчикам возможность взаимодействия с ядром системы. Компас-3D, как отечественная CAD-система, активно развивает инструментарий для интеграции, предоставляя SDK для работы с API.[4]

На сегодняшний день существует множество методов и подходов к автоматизации проектирования. Среди них выделяются:

- использование параметрического моделирования, позволяющего создавать сборки с заданными размерами и конфигурациями;
- программирование интерфейсов и модулей, расширяющих функциональную возможность CAD-систем, таких как создание пользовательских библиотек и автоматизация построений;
- внедрение пользовательских приложений для конфигурирования сборок, что повышает точность и скорость проектирования;

Для разработки приложений, взаимодействующих с CAD-системами, часто применяются следующие технологии:

API используется для интеграции пользовательского приложения с Компас-3D. Это позволяет обмениваться данными между внешним приложением и CAD-системой, а также управлять возможностями Компас-3D, что существенно расширяет возможности программного обеспечения.[4]

MFC — это инструмент для создания графических интерфейсов на языке C++, который поддерживает взаимодействие с различными библиотеками. Использование MFC позволяет разработать удобный и функциональный

пользовательский интерфейс, интегрированный с CAD-системами, такими как Компас-3D.[1]

Ориентированное на события программирование обеспечивает гибкость и удобство обработки пользовательских действий. В этой модели программа реагирует на различные события (например, нажатие кнопок, изменения данных и т. д.), что позволяет создавать интерактивные и динамичные интерфейсы. Этот подход повышает удобство взаимодействия пользователя с приложением.

Втулочные муфты – это один из наиболее распространенных видов муфт, предназначенных для передачи вращающего момента между валами в механических передачах. Они просты в конструкции и эффективны при использовании в различных условиях эксплуатации.

Втулочные муфты используются для соединения соосных валов, обеспечивая передачу крутящего момента при минимальных относительных смещениях соединяемых валов. Они применяются в машиностроении, энергетике и других отраслях, где требуется надежное соединение валов с учетом стандартных нагрузок.

Для закрепления валов внутри втулки используются конические или цилиндрические отверстия. Размеры втулок и штифтов регламентируются ГОСТами, что обеспечивает их стандартизацию и универсальность применения.

Преимущества втулочных муфт:

- Простота конструкции. Отсутствие сложных механизмов облегчает производство и снижает стоимость;
- Универсальность. Подходят для соединения валов с различными диаметрами;
- Стандартизация. Изготовление деталей в соответствии с ГОСТами обеспечивает взаимозаменяемость и удобство эксплуатации;
- Удобство сборки и замены. Простота установки и замены деталей делает их эксплуатацию максимально эффективной;

- Надежность. Применение штифтов гарантирует прочное соединение и устойчивость к нагрузкам;

Для быстрого создания таких компонентов в системе КОМПАС-3D используются библиотеки стандартных изделий, которые включают широкий ассортимент крепежных элементов, трубопроводной арматуры, подшипников и других деталей. Однако стандартизованными являются не только крепеж и простые элементы. Многие более сложные механизмы также производятся и собираются в соответствии с требованиями ГОСТов, ОСТов и других стандартов. Моделирование подобных механизмов вручную часто становится трудоемкой задачей, которая нередко занимает больше времени, чем разработка уникальных деталей. Решением этой проблемы является приложение «Библиотека муфт» от компании АСКОН, которое упрощает процесс создания сложных моделей машиностроительных муфт и их интеграции в сборки для соединения валов.(Рисунок 1)

Рисунок 1 — Примеры муфт

Рисунок 2 — Внешний вид библиотеки

Основной функционал библиотеки «Библиотека муфт»:

- Автоматизированное создание моделей муфт:
  Библиотека предназначена для построения трехмерных моделей и сборочных чертежей стандартных муфт различных типов. Все типоразмеры соответствуют стандартам ГОСТ, что избавляет пользователя от необходимости поиска и выбора габаритных размеров;
- Вариативность конфигурации:
  Библиотека позволяет выбирать конфигурации муфт для длинных или

коротких концов валов, с коническими или цилиндрическими отверстиями и другими параметрами;

- Редактирование моделей:

  Пользователь может редактировать созданные модели как вручную, средствами КОМПАС-3D, так и через интерфейс библиотеки. Поддерживается замена одной модели муфты на другую с сохранением ориентации и позиции в сборке;(Рисунок 2)

- Поддержка 2D-черчения:

  Помимо 3D-моделей, библиотека предоставляет возможность вставки чертежей муфт в графические документы. Чертежи могут быть автоматически оформлены с вырезами, привязкой характерных точек и автоматической простановкой размеров;(Рисунок 3)

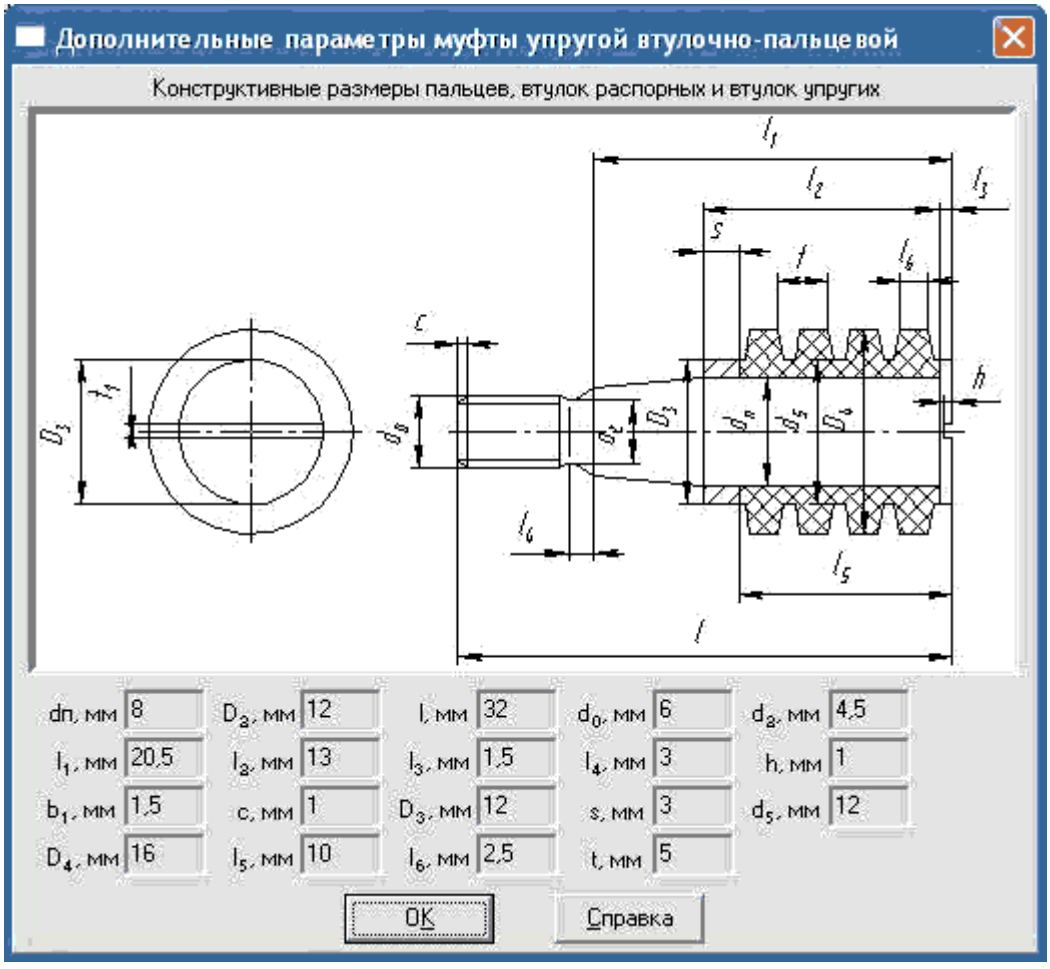


Рисунок 3 — Интерфейс для создания чертежей

- Создание спецификаций:

  Библиотека автоматически генерирует спецификации для вставленных в проект моделей и чертежей муфт;

- Настраиваемость:

  Пользователь может изменять цвет деталей, конфигурации, а также сохранять настройки для повторного использования;(Рисунок 4)

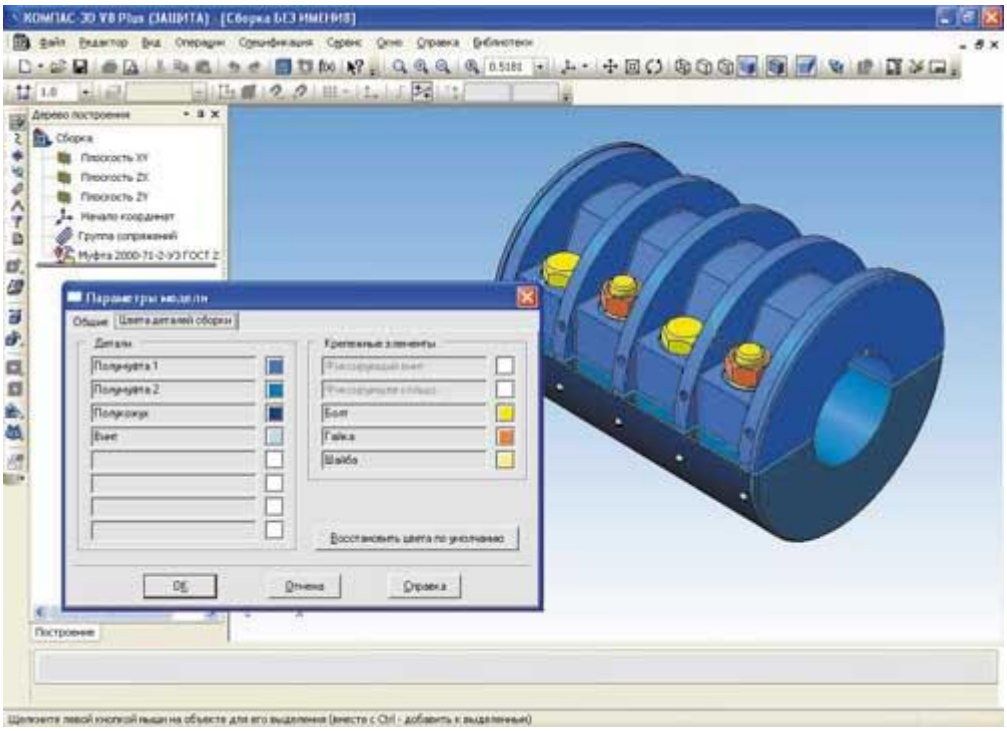


Рисунок 4— Окно для выбора цвета

Однако, в данной библиотеке не реализовано построение втулочных муфт.

2 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ МЕТОДА РЕШЕНИЯ ЗАДАЧИ

2.1 Реализация построения сборки

Начнем с объявления переменных для создания сплиттера (Листинг 1).

Листинг 1 — Создание переменных в файле MainFrm.

```
CToolBar          m_wndToolBar;
CStatusBar        m_wndStatusBar;
CSplitterWnd      m_wndSplitter;
```

Далее создадим механизм для разделения интерфейса на две части — дерево элементов построения и область основного контента. Для этого напишем код в методе OnCreateClient, который будет инициализировать сплиттер и связывать его элементы с документом (Листинг 2).

Листинг 2 — Создание сплиттера и инициализация ссылок

```
BOOL CMainFrame::OnCreateClient(LPCREATESTRUCT lpcs, CCreateContext* pContext)
{
	// TODO: добавьте специализированный код или вызов базового класса
	m_wndSplitter.CreateStatic(this, 1, 2);
	m_wndSplitter.CreateView(0, 0, RUNTIME_CLASS(CMyTreeView), CSize(250, 0),
pContext);
	m_wndSplitter.CreateView(0, 1, RUNTIME_CLASS(CCourseProjectView), CSize(0,
0), pContext);

	SetActiveView((CView*)m_wndSplitter.GetPane(0, 1));

	CCourseProjectDoc* pDoc = (CCourseProjectDoc*)GetActiveDocument();

	pDoc->m_pTreeView = (CMyTreeView*)m_wndSplitter.GetPane(0, 0);
	pDoc->m_pView = (CCourseProjectView*)m_wndSplitter.GetPane(0, 1);
	pDoc->m_pTreeView->m_pDoc = pDoc;

	return TRUE;
}
```

Через мастер классов создаем класс CMyTreeView. В заголовочном файле заводим необходимые переменные и создаем метод FillTree (Листинг 3).

Листинг 3 — Создание переменных и методов древовидного списка

```
CCourseProjectDoc* m_pDoc;

HTREEITEM m_hCoupling;    //Муфта
HTREEITEM m_hBushing;     //Втулка
HTREEITEM m_hPin;         //Штифт

void FillTree();
```

В файле MyTreeView.cpp добавим реализацию метода FillTree, который заполняет список необходимыми элементами. Этот метод будет вызываться для инициализации дерева или обновления его содержимого (Листинг 4).

Листинг 4 — Реализация метода FillTree

```
void CMyTreeView::FillTree()
{
	CTreeCtrl& tree = GetTreeCtrl();

	tree.DeleteAllItems();

	m_hCoupling = tree.InsertItem(L"Втулочная муфта", -1, -1, NULL,
	TVI_FIRST);

	m_hBushing = tree.InsertItem(L"Втулка", -1, -1, m_hCoupling, TVI_FIRST);
	m_hPin = tree.InsertItem(L"Штифт", -1, -1, m_hCoupling, TVI_FIRST);

	tree.SetCheck(m_hBushing, m_pDoc->m_bBushing);
	tree.SetCheck(m_hPin, m_pDoc->m_bPin);

	tree.Expand(m_hCoupling, TVE_EXPAND);

}
```

Реализуем обработку нажатия левой кнопкой мыши на элементы дерева в представлении CMyTreeView. Мы определяем, на какой узел дерева (Втулочная муфта, Втулка, Штифт) кликнул пользователь. Если клик попадает в область узла, этот узел выделяется. После этого обновим соответствующие флаги в связанном документе (m_bCoupling, m_bBushing, m_bPin), чтобы синхронизировать состояние дерева с внутренней логикой приложения. Таким образом, обеспечим корректное взаимодействие пользователя с интерфейсом и дальнейшую обработку выбранных данных (Листинг 5).

Листинг 5 — Реализация нажатия на левую кнопку мыши

```
void CMyTreeView::OnLButtonDown(UINT nFlags, CPoint point)
{
	// TODO: добавьте свой код обработчика сообщений или вызов стандартного

	CTreeCtrl& tree = GetTreeCtrl();

	CRect rc;

	tree.GetItemRect(m_hCoupling, &rc, false);
	if (rc.PtInRect(point))
		tree.SelectItem(m_hCoupling);

	tree.GetItemRect(m_hBushing, &rc, false);
	if (rc.PtInRect(point))
		tree.SelectItem(m_hBushing);

	tree.GetItemRect(m_hPin, &rc, false);
	if (rc.PtInRect(point))
		tree.SelectItem(m_hPin);

	if (tree.GetSelectedItem() == m_hCoupling)
	{
		m_pDoc->m_bCoupling = TRUE;
		m_pDoc->m_bBushing = FALSE;
		m_pDoc->m_bPin = FALSE;
	}

	if (tree.GetSelectedItem() == m_hBushing)
	{
		m_pDoc->m_bBushing = TRUE;
		m_pDoc->m_bCoupling = FALSE;
		m_pDoc->m_bPin = FALSE;
	}

	if (tree.GetSelectedItem() == m_hPin)
	{
		m_pDoc->m_bPin = TRUE;
		m_pDoc->m_bCoupling = FALSE;
		m_pDoc->m_bBushing = FALSE;
	}

	m_pDoc->m_pView->Invalidate();
	CTreeView::OnLButtonDown(nFlags, point);
}
```

В файле документа заводим необходимые переменные для построения муфты. Объявление этих переменных в файле документа обеспечивает централизованное управление параметрами и упрощает доступ к ним из различных частей приложения. (Листинг 6)

Листинг 6 — Создание переменных

```
CCourseProjectView* m_pView;
CMyTreeView* m_pTreeView;

BOOL m_bCoupling;   //Муфта
BOOL m_bBushing;    //Втулка
BOOL m_bPin;        //Штифт
```

В файле CourseProjectDoc.cpp в методе OnNewDocument добавляем дополнительную реализацию. Таким образом, обеспечивается корректная инициализация документа и синхронизация его состояния с интерфейсом при создании проекта (Листинг 7).

Листинг 7 — Инициализация переменных

```
BOOL CCourseProjectDoc::OnNewDocument()
{
	if (!CDocument::OnNewDocument())
		return FALSE;

	m_bCoupling = m_bBushing = m_bPin = FALSE;

	m_pTreeView->FillTree();
	return TRUE;
}
```

Определяем структуры для штифта, вала, втулки и муфты, каждая из которых содержит параметры, описывающие геометрию и свойства соответствующей детали, а также пути к связанным с ними файлам. Для работы с множественными экземплярами каждой детали используются «std::vector». Это позволяет хранить и обрабатывать коллекции штифтов, валов и втулок с различными параметрами, обеспечивая гибкость в конфигурировании муфты. Ключевыми методами класса являются функции CreatePin, CreateShaft, CreateBushing, которые строят 3D-модели деталей на основе параметров, заданных в соответствующих структурах. Метод CreateCoupling отвечает за сборку муфты, комбинируя созданные ранее детали и, определяя между ними сопряжения в среде КОМПАС-3D (Листинг 8).

Листинг 8 — Создание переменных и методов

```
public:
	CCourseProjectDoc* GetDocument() const;

	KompasObjectPtr kompas;

	//Структура для штифта
	struct Pin
	{
		double d, d1, l, c;
		CString name[5];
		CString filePath;
	};
	//Структура для вала
	struct Shaft
	{
		double D, d, l, c, L;
		CString filePath;
	};
	//Структура для втулки
	struct Bushing
	{
		double d, d1, D, L, l, c;
		CString name[8];
		CString filePath;
	};
	//Структура для муфты
	struct Coupling
	{
		CString name[3];
	};
	//Векторы (массивы) структур по деталям
	std::vector<Pin> vecPin;				//вектор для штифта
	std::vector<Bushing> vecBushing;		//вектор для втулки
	std::vector<Shaft> vecShaft;			//вектор для вала
	std::vector<Coupling> vecCoupling;		//вектор для муфты

	CButton m_Buttons[6];		//массив кнопок
	CString m_sfilePath;		//файловый путь

	//Методы для построения деталей
	void CreatePin(Pin& st);
	void CreateShaft(Shaft& st);
	void CreateBushing(Bushing& st);
	//Метод для построения сборки
	void CreateCoupling(Pin ring, Shaft shaft, Bushing bushing);
```

Добавим в заголовочный файл, содержащий директивы препроцессора для подключения необходимых библиотек и интерфейсов для взаимодействия с API КОМПАС-3D v22 Study. #pragma once гарантирует, что данный файл будет включен в проект только один раз, предотвращая возможные конфликты (Листинг 9).

Листинг 9 — Подключение необходимых библиотек

```
#include "C:\\Program Files\\ASCON\\KOMPAS-3D v21
Study\\SDK\\Include\\ksConstants.h"
#include "C:\\Program Files\\ASCON\\KOMPAS-3D v21
Study\\SDK\\Include\\ksConstants3D.h"

#import "C:\\Program Files\\ASCON\\KOMPAS-3D v21 Study\\SDK\\lib\\kAPI5.tlb"

using namespace Kompas6API5;
```

Добавляем реализацию метода для построения «Втулки» (Листинг 10).

Листинг 10 — Реализация метода построения для детали

```
// Метод для создания втулки
void CCourseProjectView::CreateBushing(Bushing& st)
{
	CoInitialize(NULL);
	HRESULT hRes;
	hRes = kompas.GetActiveObject(L"Kompas.Application.5");
	if (FAILED(hRes))
		kompas.CreateInstance(L"Kompas.Application.5");
	kompas->Visible = true;
	//создание документа КОМПАС:
	ksDocument3DPtr doc;
	doc = kompas->Document3D();
	doc->Create(false, true);
	doc = kompas->ActiveDocument3D();
	ksPartPtr part;
	part = doc->GetPart(pTop_Part);
	//эскиз под выдавливание
	ksEntityPtr sketch = part->NewEntity(o3d_sketch);
	ksSketchDefinitionPtr sketchDef = sketch->GetDefinition();
	sketchDef->SetPlane(part->GetDefaultEntity(o3d_planeXOY));
	sketch->Create();
	ksDocument2DPtr doc2D = sketchDef->BeginEdit();
	doc2D->ksCircle(0, 0, st.D / 2.f, 1);
	doc2D->ksCircle(0, 0, st.d / 2.f, 1);
	sketchDef->EndEdit();
	//операция выдавливания
	ksEntityPtr baseExtr = part->NewEntity(o3d_baseExtrusion);
	ksBaseExtrusionDefinitionPtr ExtrDef = baseExtr->GetDefinition();
	ExtrDef->SetSideParam(TRUE, etBlind, st.L, 0, TRUE);
	ExtrDef->SetSketch(sketch);
	baseExtr->Create();
	//операция смещенная плоскость
	ksEntityPtr planeXOZ = part->GetDefaultEntity(o3d_planeXOZ);
	ksEntityPtr plane = part->NewEntity(o3d_planeOffset);
	ksPlaneOffsetDefinitionPtr planeDef = plane->GetDefinition();
	planeDef->direction = TRUE;
	planeDef->offset = st.D / 2.f;
	planeDef->SetPlane(planeXOZ);
	plane->Create();
```

Продолжение листинга 10

```
        //эскиз под отверстия
        ksEntityPtr sketch2 = part->NewEntity(o3d_sketch);
        ksSketchDefinitionPtr sketchDef2 = sketch2->GetDefinition();
        sketchDef2->SetPlane(plane);
        sketch2->Create();
        ksDocument2DPtr doc2D2 = sketchDef2->BeginEdit();
        doc2D2->ksCircle(0, -st.l, st.d1 / 2.f, 1);
        doc2D2->ksCircle(0, -(st.L - st.l), st.d1 / 2.f, 1);
        sketchDef2->EndEdit();

        //операция вырез выдавливанием
        ksEntityPtr CutExtr = part->NewEntity(o3d_cutExtrusion);
        ksCutExtrusionDefinitionPtr CutExtrDef = CutExtr->GetDefinition();
        CutExtrDef->directionType = dtNormal;
        CutExtrDef->SetSketch(sketch2);
        CutExtrDef->SetSideParam(TRUE, etBlind, st.D, 0, FALSE);
        CutExtr->Create();
        //операция фаска
        ksEntityPtr Chamfers = part->NewEntity(o3d_chamfer);
        ksChamferDefinitionPtr ChamfersDef = Chamfers->GetDefinition();
        ChamfersDef->SetChamferParam(true, st.c, st.c);
        ksEntityCollectionPtr Collection = part->EntityCollection(o3d_edge);
        ksEntityCollectionPtr CollectionChamfers = ChamfersDef->array();
        Collection->SelectByPoint(0, st.d / 2.f, 0);
        CollectionChamfers->Add(Collection->First());
        Collection->Clear();
        Collection = part->EntityCollection(o3d_edge);
        Collection->SelectByPoint(0, st.D / 2.f, 0);
        CollectionChamfers->Add(Collection->First());
        Collection->Clear();
        Collection = part->EntityCollection(o3d_edge);
        Collection->SelectByPoint(0, st.d / 2.f, st.L);
        CollectionChamfers->Add(Collection->First());
        Collection->Clear();
        Collection = part->EntityCollection(o3d_edge);
        Collection->SelectByPoint(0, st.D / 2.f, st.L);
        CollectionChamfers->Add(Collection->First());
        Chamfers->Create();
        //операция сохранения детали
        CString name = L"Втулка";
        st.filePath = m_sfilePath;
        doc->fileName = _bstr_t(name);
        st.filePath += L"\\" + name + L".m3d";
        doc->SaveAs(_bstr_t(st.filePath));
}
```

На рисунке 5 представлен результат построения в Компас-3D.

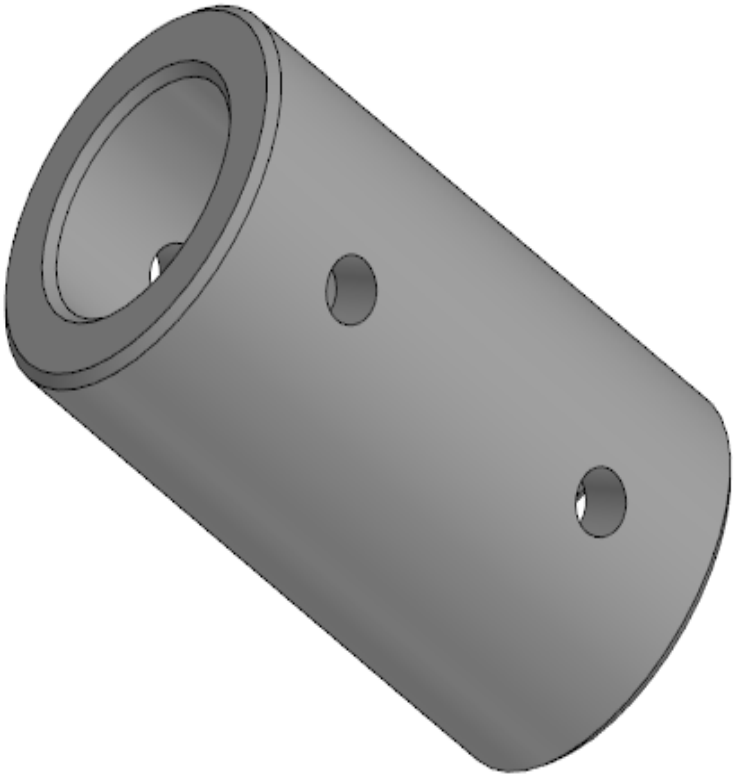


Рисунок 5 — Результат построения втулки

Аналогичным образом реализуем код для построения остальных деталей сборки. Для каждой детали, такой как вал, штифт и муфта, создадим отдельные методы, которые будут взаимодействовать с КОМПАС-3D для автоматического создания их 3D-моделей. В этих методах будем задавать соответствующие геометрические параметры и последовательности операций: построение эскизов, выполнение выдавливания, создание смещенных плоскостей или других необходимых операций. Такой подход позволит формировать каждую деталь сборки с учетом ее индивидуальных характеристик, обеспечивая согласованность параметров и точность выполнения моделей. Полный код построения представлен в Приложении А. Результат показан на рисунках 6-7.

Рисунок 6 — Вал

Рисунок 7 — Штифт

Добавим компонент “Втулка” в сборку. Сначала происходит загрузка 3D-модели втулки из файла, указанного в структуре bushing, в текущий документ. После загрузки, код получает доступ к коллекции геометрических элементов этой детали, конкретно к ребрам и граням, используя соответствующие методы API. Затем, с помощью метода SelectByPoint, происходит выбор определенных ребер и граней на поверхности втулки (Листинг 11).

Листинг 11 — Добавление детали «Втулка»

```
CoInitialize(NULL);
HRESULT hRes;
hRes = kompas.GetActiveObject(L"Kompas.Application.5");
if (FAILED(hRes))
      kompas.CreateInstance(L"Kompas.Application.5");
kompas->Visible = true;

//создание документа КОМПАС :
ksDocument3DPtr doc;
doc = kompas->Document3D();
doc->Create(false, false);
doc = kompas->ActiveDocument3D();
ksPartPtr part;
part = doc->GetPart(pTop_Part);

//добавление детали "Втулка"
doc->SetPartFromFile(_bstr_t(bushing.filePath), part, TRUE);
ksPartPtr partbushing = doc->GetPart(0);
ksEntityCollectionPtr bushingCollection = partbushing->EntityCollection(o3d_edge);
```

Реализуем добавление в сборку двух одинаковых деталей "Вал". Сначала загрузим первую деталь из указанного файла и подключим ее к сборке. Затем получим доступ к коллекциям объектов этой модели, таких как ребра и грани, для дальнейшей работы. Определим ключевые элементы, например, первое ребро и грань, которые будут использоваться для установки сопряжений.

Аналогичным образом добавим в сборку вторую деталь "Вал". Повторим процедуру загрузки файла, подключения детали к сборке и извлечения необходимых элементов для работы (Листинг 12).

Промежуточный результат представлен на рисунке 8.

Листинг 12 — Добавление валов

```
//добавление детали "Вал" x1
      doc->SetPartFromFile(_bstr_t(shaft.filePath), part, TRUE);
ksPartPtr partshaft = doc->GetPart(1);
ksEntityCollectionPtr shaftCollection = partshaft->EntityCollection(o3d_edge);
//ребро 1
ksEntityPtr edgeshaft;
shaftCollection->SelectByPoint(0, shaft.D / 2.f, 0);
edgeshaft = shaftCollection->First();
//грань
ksEntityPtr faceshaft;
shaftCollection->Clear();
shaftCollection = partshaft->EntityCollection(o3d_face);
shaftCollection->SelectByPoint(shaft.d / 2.f, 0, shaft.L - shaft.l);
faceshaft = shaftCollection->First();

//добавление детали "Вал" x2
doc->SetPartFromFile(_bstr_t(shaft.filePath), part, TRUE);
ksPartPtr partshaft2 = doc->GetPart(2);
ksEntityCollectionPtr shaftCollection2 = partshaft2->EntityCollection(o3d_edge);
//ребро 1
ksEntityPtr edgeshaft2;
shaftCollection2->SelectByPoint(0, shaft.D / 2.f, 0);
edgeshaft2 = shaftCollection2->First();
//грань
ksEntityPtr faceshaft2;
shaftCollection2->Clear();
shaftCollection2 = partshaft2->EntityCollection(o3d_face);
shaftCollection2->SelectByPoint(shaft.d / 2.f, 0, shaft.L - shaft.l);
faceshaft2 = shaftCollection2->First();
```

Рисунок 8 — Втулка с установленными валами

Добавим два штифта. Для каждого из них загружаем файл модели и получаем доступ к коллекциям элементов детали, таких как грани и ребра. Эти элементы определяются на основе заданных координат, чтобы обеспечить возможность задания сопряжений.

Для первой детали извлекаем нужную грань и ребро, которые будут использоваться в дальнейших операциях. После этого аналогичные действия повторяются для второй детали "Штифт": загружается файл, выделяются коллекции, а затем определяются требуемые геометрические элементы. Эти подготовительные шаги позволяют корректно настроить взаимодействие деталей внутри сборки (Листинг 13).

Листинг 13 — Добавление штифтов

```
//добавление детали "Штифт" x1
doc->SetPartFromFile(_bstr_t(pin.filePath), part, TRUE);
ksPartPtr partpin = doc->GetPart(3);
ksEntityCollectionPtr pinCollection = partpin->EntityCollection(o3d_face);
//грань
ksEntityPtr facepin;
pinCollection->SelectByPoint(pin.l, 0, 0);
facepin = pinCollection->First();
//ребро
ksEntityPtr edgepin;
pinCollection->Clear();
pinCollection = partpin->EntityCollection(o3d_edge);
pinCollection->SelectByPoint(pin.l, pin.d1 / 2.f - pin.c, 0);
edgepin = pinCollection->First();

//добавление детали "Штифт" x2
doc->SetPartFromFile(_bstr_t(pin.filePath), part, TRUE);
ksPartPtr partpin2 = doc->GetPart(4);
ksEntityCollectionPtr pinCollection2 = partpin2->EntityCollection(o3d_face);
//грань
ksEntityPtr facepin2;
pinCollection2->SelectByPoint(pin.l, 0, 0);
facepin2 = pinCollection2->First();
//ребро
ksEntityPtr edgepin2;
pinCollection2->Clear();
pinCollection2 = partpin2->EntityCollection(o3d_edge);
pinCollection2->SelectByPoint(pin.l, pin.d1 / 2.f - pin.c, 0);
```

После добавления всех деталей в сборку, следующим шагом является определение кинематических связей (сопряжений) между ними, что задаёт геометрические зависимости модели. Также установим параметры сохранения сборки (Листинг 14).

Листинг 14 — Установка сопряжений и сохранение сборки

```
//Установка сопряжений
doc->AddMateConstraint(mc_Concentric, edgebushing, edgeshaft, 1, 0, NULL);
doc->AddMateConstraint(mc_Concentric, edgebushing, edgeshaft2, -1, 0, NULL);
doc->AddMateConstraint(mc_Concentric, facebushing, faceshaft, 1, 0, NULL);
doc->AddMateConstraint(mc_Concentric, facebushing2, faceshaft2, 1, 0, NULL);
doc->AddMateConstraint(mc_Concentric, facebushing, edgepin, 1, 0, NULL);
doc->AddMateConstraint(mc_Tangency, facepin, facebushing3, -1, 0, NULL);
doc->AddMateConstraint(mc_Concentric, facebushing2, edgepin2, -1, 0, NULL);
doc->AddMateConstraint(mc_Tangency, facepin2, facebushing3, -1, 0, NULL);

//операция сохранения детали
CString name = L"Втулочная муфта";
CString filePath = m_sfilePath;
doc->fileName = _bstr_t(name);
filePath += L"\\" + name + L".a3d";
doc->SaveAs(_bstr_t(filePath));
```

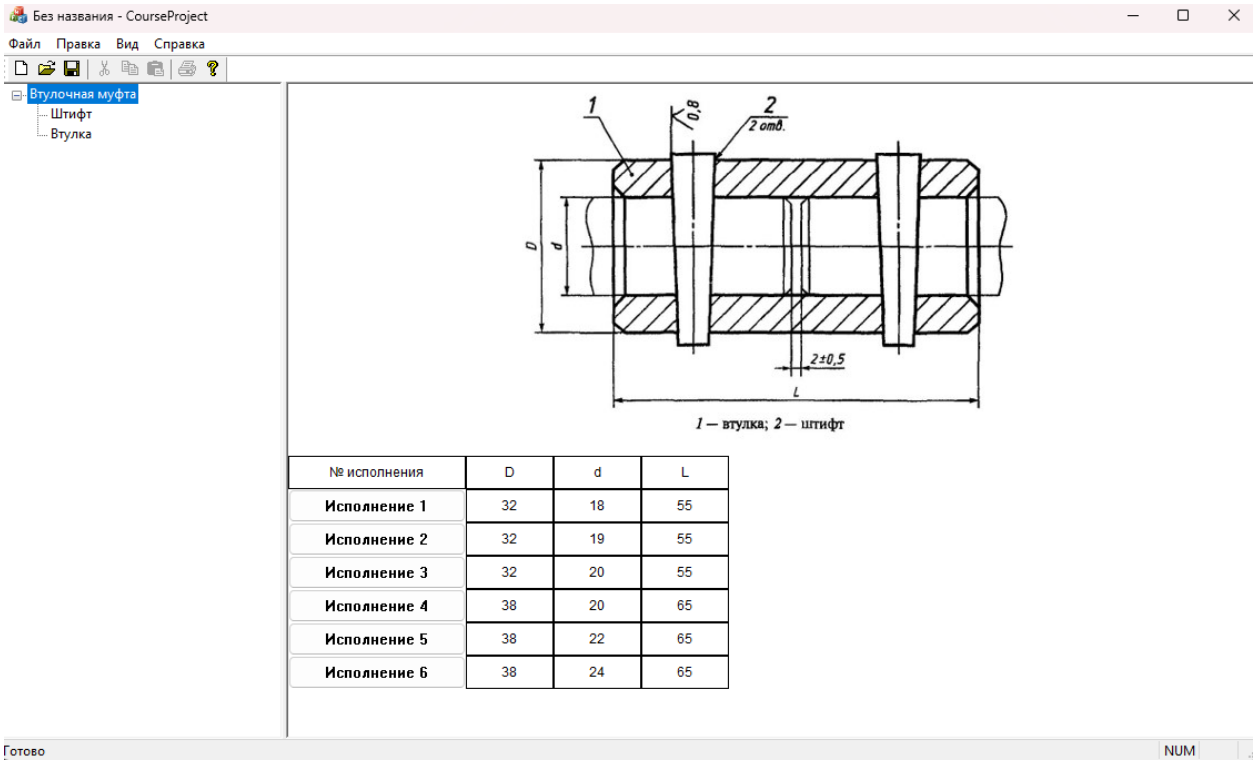
В результате выполнения программы формируется итоговая сборка муфты, представляющая собой модель, в которой все детали — втулка, два вала и два штифта — объединены в единую конструкцию. Каждая деталь занимает свое строго определенное место, а заданные сопряжения обеспечивают корректное взаимное расположение и соединение элементов (Рисунок 9).

Рисунок 9 — Сборка «Втулочная муфта»

2.2 Итоговый интерфейс приложения

Выбор пункта в древовидном списке активирует соответствующий блок интерфейса:

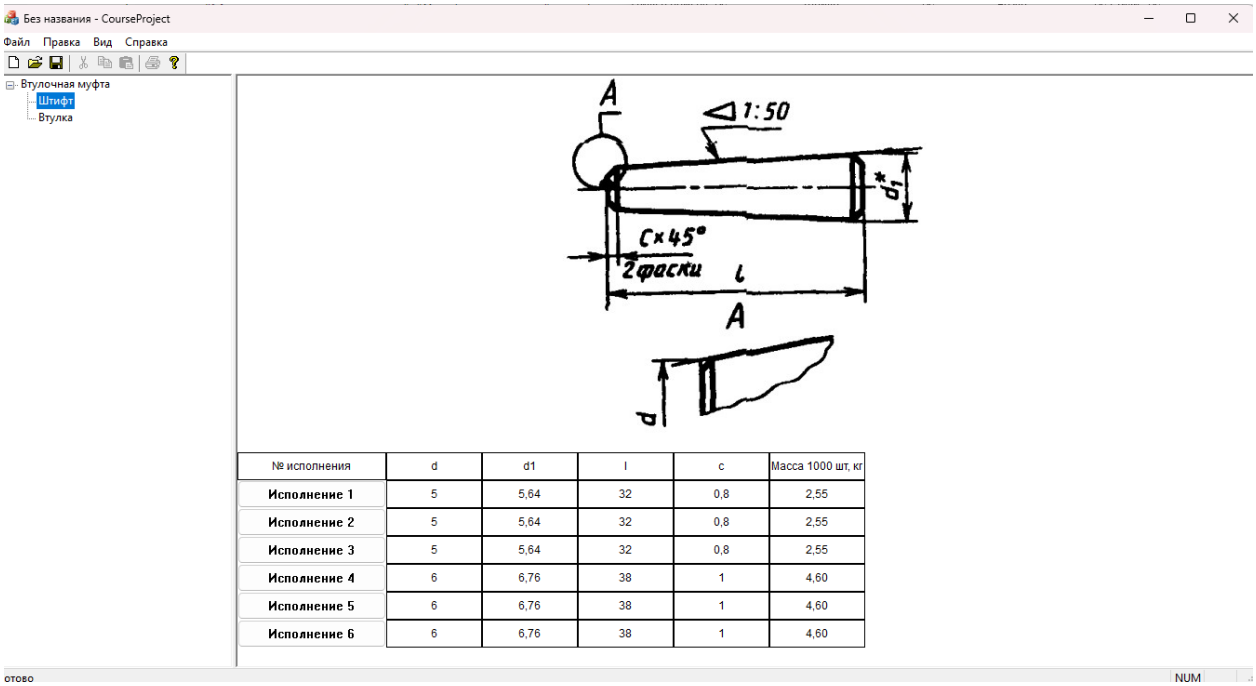
При нажатии на элемент «Втулочная муфта», на главном экране выводится сборочный чертеж муфты (ГОСТ 24246-96), таблица с исполнениями и кнопки, с помощью которых можно выбрать какое исполнение будет построено (Рисунок 10).



№ исполнения	D	d	L
Исполнение 1	32	18	55
Исполнение 2	32	19	55
Исполнение 3	32	20	55
Исполнение 4	38	20	65
Исполнение 5	38	22	65
Исполнение 6	38	24	65

Рисунок 10 — Окно для построения сборки «Втулочная муфта»

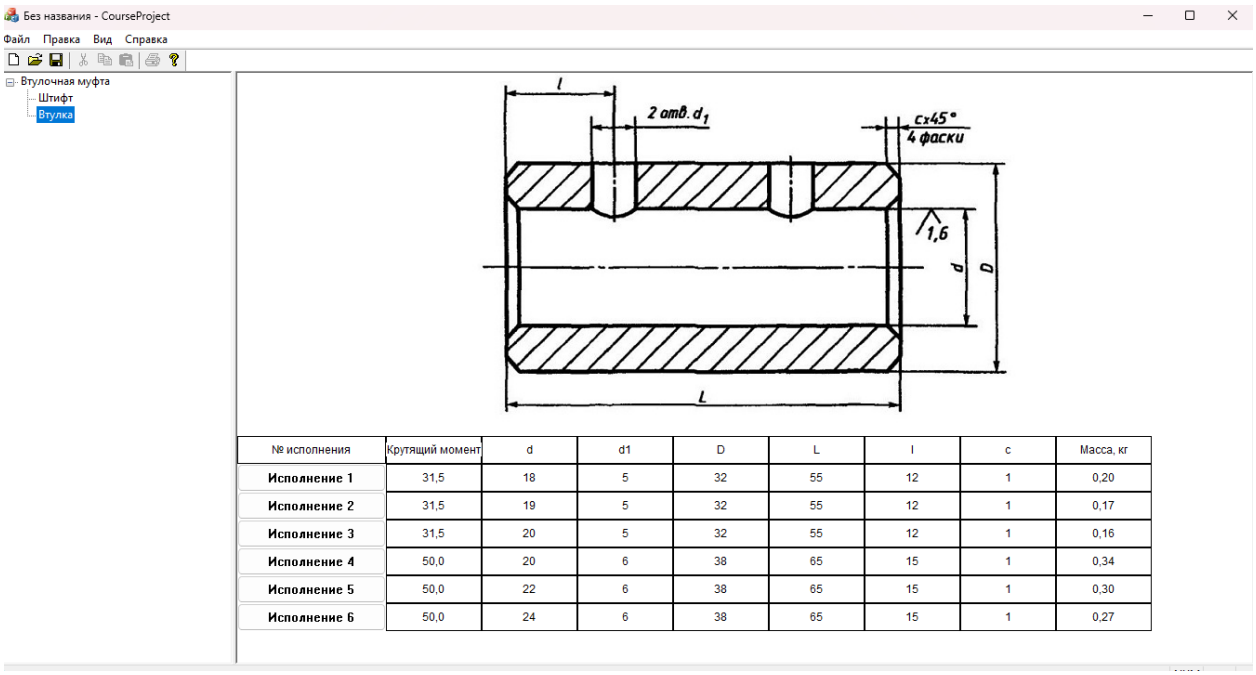
При нажатии на элемент «Штифт», на главном экране выводится чертеж штифта (ГОСТ 3129), таблица с исполнениями и соответствующие размеры для построения (Рисунок 11).



№ исполнения	d	d1	l	c	Масса 1000 шт, кг
Исполнение 1	5	5,64	32	0,8	2,55
Исполнение 2	5	5,64	32	0,8	2,55
Исполнение 3	5	5,64	32	0,8	2,55
Исполнение 4	6	6,76	38	1	4,60
Исполнение 5	6	6,76	38	1	4,60
Исполнение 6	6	6,76	38	1	4,60

Рисунок 11 — Окно для построения штифта

При нажатии на элемент «Втулка», на главном экране выводится чертеж втулки, таблица с исполнениями, размерами и массой изделия (Рисунок 12).



№ исполнения	Крутящий момент	d	d1	D	L	l	c	Масса, кг
Исполнение 1	31,5	18	5	32	55	12	1	0,20
Исполнение 2	31,5	19	5	32	55	12	1	0,17
Исполнение 3	31,5	20	5	32	55	12	1	0,16
Исполнение 4	50,0	20	6	38	65	15	1	0,34
Исполнение 5	50,0	22	6	38	65	15	1	0,30
Исполнение 6	50,0	24	6	38	65	15	1	0,27

Рисунок 12 — Окно для построения втулки

ЗАКЛЮЧЕНИЕ

Было разработано MFC SDI приложение, которое использует API Компас-3D для создания параметрической сборки "Втулочная муфта". Ключевая особенность решения — это параметрическое ядро, которое обеспечивает мгновенное изменение геометрии сборки в зависимости от заданных параметров.

Также, создан удобный интерфейс, с возможностью выбора конфигурации сборки. Реализован древовидный список, который дает возможность выбирать отдельные детали сборки. Это решение значительно улучшает управление процессом моделирования, предлагая пользователю интуитивно понятный интерфейс для настройки компонентов.

Таким образом, созданное приложение с использованием API Компас-3D для проектирования параметрической сборки "Втулочная муфта" представляет собой удобный инструмент, способный повысить эффективность работы инженеров и конструкторов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Корпорация Microsoft. Образовательно-справочный сайт по языку программировании C++. URL: https://learn.microsoft.com/ru-ru/cpp/mfc/sdi-and-mdi?view=msvc-170 (дата обращения: 18.12.2024).
2. Научно-производственная организация «Русские технологии». Справочный сайт по ГОСТу 4860.2-83. — 1995— . URL: https://rus-teh.org/products/cable-routes/gasket/139?ysclid=lqax1u13xp54950871 (дата обращения: 18.12.2024).
3. ООО «АСКОН-Системы проектирования». Образовательно-справочный сайт по SDK КОМПАС-3D. URL: https://help.ascon.ru/KOMPAS_SDK/22/ru-RU/index.html (дата обращения: 18.12.2024).
4. ВикиЧтение. Образовательно-справочный сайт по базовым интерфейсам API системы КОМПАС. — 2010— . URL: https://it.wikireading.ru/23741?ysclid=lqaxc0sxng274971479 (дата обращения: 18.12.2024).
5. ООО «САПР.РУ». Статья «Легко соединяем валы: АСКОН представляет „Библиотеку муфт“». URL: https://sapr.ru/article/16693?ysclid=m650kwhk91960835090 (дата обращения: 20.01.2025).
6. Решетов, Д. Н. Детали машин. Атлас конструкций: в 2 ч. Ч. 2 / Д. Н. Решетов. — 5-е изд., перераб. и доп. — Москва: Машиностроение, 1992. — 353 с. — ISBN 5-217-01507-1.
7. Чернов, Е. М., Курбатов, В. М. Детали машин: Справочник / Е. М. Чернов, В. М. Курбатов. — Москва: Машиностроение, 2004. — 616 с. — ISBN 5-217-02775-3.
8. Проскуряков, А. П. Microsoft Visual C++ и MFC: практическое программирование : учеб. пособие для студентов вузов / А. П.

Проскуряков. — Москва : БХВ-Петербург, 2021. — 480 с. — ISBN 978-5-9775-09024-3.

9. Глухов, С. А. САПР: теория и практика автоматизированного проектирования : учеб. пособие для студентов вузов / С. А. Глухов, А. С. Глухова. — Москва : Юрайт, 2020. — 400 с. — ISBN 978-5-534-11076-5.
10. Никулин, Н. А. Моделирование и проектирование машин: практика использования САПР : учеб. для студентов вузов / Н. А. Никулин. — Екатеринбург : Уральский издательский дом, 2020. — 320 с. — ISBN 978-5-7996-12345-8.
11. Шилов, И. П. Алгоритмы и технологии автоматизации инженерного проектирования : учеб. пособие для студентов вузов / И. П. Шилов. — Москва : Высшая школа, 2018. — 410 с. — ISBN 978-5-06-093451-2.
12. Плейлист «API Компас-3D C++» //Rutube. URL: https://rutube.ru/plst/239015/ (дата обращения: 18.01.2025).
13. StudyCAD. Обучение автоматизированному проектированию : [канал пользователя] // Youtube https://www.youtube.com/@studyCAD : [видеохостинг]. — URL: https://www.youtube.com/@studyCAD (дата обращения: 19.01.2025).
14. CreativeMechMan. Разработка механических систем : [канал пользователя]//Youtube:[видеохостинг].https://www.youtube.com/@CreativeMechMan — URL: (дата обращения: 19.01.2025).
15. Magnificent Tech Solutions : [канал пользователя] // Youtube : [видеохостинг].URL:https://www.youtube.com/@MagnificentTechSolutions (дата обращения: 19.01.2025).
16. Technology 21 : [канал пользователя] // Youtube : [видеохостинг]. — URL: https://www.youtube.com/@Technology21 (дата обращения: 19.01.2025).
17. Rrazvan : [канал пользователя] // Youtube : [видеохостинг]. — URL: https://www.youtube.com/@rrazvan (дата обращения: 19.01.2025)

18. Шилдт Г. C++ базовый курс / Г. Шилдт. — Диалектика-Вильямс, 2018. — 624 с. — URL: https://wow-only.ru/books/shield.pdf (дата обращения: 15.12.2025).
19. Тихомиров, Ю.В. самоучитель MFC / Ю.В. Тихомиров // Вычислительная техника. Программирование / Теоретические основы программирования. — 2006. — No 2. — с. 640 — 5-8206-0096-
20. Stack Overflow. Вопросы и ответы по программированию на C++. URL: https://stackoverflow.com/questions/tagged/c%2b%2b (дата обращения: 18.12.2024).

ПРИЛОЖЕНИЕ А. ИСХОДНЫЙ КОД ПОСТРОЕНИЯ СБОРКИ

Листинг 15 — Исходный код

```
// CourseProjectView.cpp: реализация класса CCourseProjectView
//

#include "pch.h"
#include "framework.h"
// SHARED_HANDLERS можно определить в обработчиках фильтров просмотра реализации
проекта ATL, эскизов
// и поиска; позволяет совместно использовать код документа в данным проекте.
#ifndef SHARED_HANDLERS
#include "CourseProject.h"
#endif

#include "CourseProjectDoc.h"
#include "CourseProjectView.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#endif

IMPLEMENT_DYNCREATE(CCourseProjectView, CView)

BEGIN_MESSAGE_MAP(CCourseProjectView, CView)
	// Стандартные команды печати
	ON_COMMAND(ID_FILE_PRINT, &CView::OnFilePrint)
	ON_COMMAND(ID_FILE_PRINT_DIRECT, &CView::OnFilePrint)
	ON_COMMAND(ID_FILE_PRINT_PREVIEW, &CView::OnFilePrintPreview)
	ON_WM_CREATE()
	ON_CONTROL_RANGE(BN_CLICKED, IDC_BUTTON_NEST_1, IDC_BUTTON_NEST_6,
&CCourseProjectView::OnButtonClick)
END_MESSAGE_MAP()


CCourseProjectView::CCourseProjectView() noexcept
{
	// TODO: добавьте код создания

}

CCourseProjectView::~CCourseProjectView()
{
}

BOOL CCourseProjectView::PreCreateWindow(CREATESTRUCT& cs)
{
	// TODO: изменить класс Window или стили посредством изменения
	//  CREATESTRUCT cs

	return CView::PreCreateWindow(cs);
}


// Рисование CCourseProjectView



void CCourseProjectView::OnDraw(CDC* pDC)
{
	CCourseProjectDoc* pDoc = GetDocument();
	ASSERT_VALID(pDoc);
```

```
	if (!pDoc)
		return;

	// TODO: добавьте здесь код отрисовки для собственных данных

	CRect clientRect;
	GetClientRect(&clientRect);

	int Height = clientRect.Height();	//Высота окна
	int Width = clientRect.Width();		//Ширина окна
	int headingWidth = 160;				//Ширина колонки под № исполения
	int cellHeight = 30;				//Высота одной ячейки
	double cellWidth = (Width - headingWidth) / 9.f;	//Ширина одной ячейки

if (pDoc->m_bCoupling)
{
	CFont font;
	font.CreatePointFont(90, L"Arial");
	CFont* pOldFont = pDC->SelectObject(&font);

	int imageHeight = CreateImage(pDC, Width, L"res\\Coupling.jpg");
	int x = 0;
	int y = imageHeight + 15;

	// Рисование шапки таблицы
	{
		CString CouplingBuff[4] = { L"№ исполнения", L"D", L"d", L"L" };

		CRect HeaderRect(x, y, x + headingWidth, y + cellHeight);
		pDC->Rectangle(HeaderRect);
		pDC->DrawText(CouplingBuff[0], HeaderRect, DT_SINGLELINE | DT_CENTER | DT_VCENTER);
		x += headingWidth;

		for (int i = 1; i < 4; i++)
		{
			CRect HeaderRect(x, y, x + cellWidth, y + cellHeight);
			pDC->Rectangle(HeaderRect);
			pDC->DrawText(CouplingBuff[i], HeaderRect, DT_SINGLELINE | DT_CENTER | DT_VCENTER);
			x += cellWidth;
		}
	}

	// Заполнение таблицы
	{
		for (int i = 0; i < vecCoupling.size(); i++)
		{
			y += cellHeight;
			x = 0;

			CRect ButtonRect(x, y, x + headingWidth, y + cellHeight);

			m_Buttons[i].MoveWindow(ButtonRect);
			m_Buttons[i].ShowWindow(SW_SHOW);

			x += headingWidth;

			for (int g = 0; g < 3; g++)
			{
				CRect HeaderRect(x, y, x + cellWidth, y + cellHeight);
				pDC->Rectangle(HeaderRect);
				pDC->DrawText(vecCoupling[i].name[g], HeaderRect, DT_SINGLELINE | DT_CENTER | DT_VCENTER);
```

```
                        x += cellWidth;
                    }
                }
        }

        pDC->SelectObject(pOldFont);
}
//Если в дереве выбрана втулка
if (pDoc->m_bBushing)
{
        CFont font;
        font.CreatePointFont(90, L"Arial");
        CFont* pOldFont = pDC->SelectObject(&font);

        int imageHeight = CreateImage(pDC, Width, L"res\\Bushing.jpg");
        int x = 0;
        int y = imageHeight + 15;

        // Рисование шапки таблицы
        {
                CString BushingBuff[9] = { L"№ исполнения", L"Крутящий момент", L"d",
L"d1", L"D", L"L", L"l", L"c", L"Масса, кг" };

                CRect HeaderRect(x, y, x + headingWidth, y + cellHeight);
                pDC->Rectangle(HeaderRect);
                pDC->DrawText(BushingBuff[0], HeaderRect, DT_SINGLELINE | DT_CENTER |
DT_VCENTER);
                x += headingWidth;

                for (int i = 1; i < 9; i++)
                {
                        CRect HeaderRect(x, y, x + cellWidth, y + cellHeight);
                        pDC->Rectangle(HeaderRect);
                        pDC->DrawText(BushingBuff[i], HeaderRect, DT_SINGLELINE |
DT_CENTER | DT_VCENTER);
                        x += cellWidth;
                }
        }

        // Заполнение таблицы
        {
                for (int i = 0; i < vecBushing.size(); i++)
                {
                        y += cellHeight;
                        x = 0;

                        CRect ButtonRect(x, y, x + headingWidth, y + cellHeight);

                        m_Buttons[i].MoveWindow(ButtonRect);
                        m_Buttons[i].ShowWindow(SW_SHOW);

                        x += headingWidth;

                        for (int g = 0; g < 8; g++)
                        {
                                CRect HeaderRect(x, y, x + cellWidth, y + cellHeight);
                                pDC->Rectangle(HeaderRect);
                                pDC->DrawText(vecBushing[i].name[g], HeaderRect,
DT_SINGLELINE | DT_CENTER | DT_VCENTER);
                                x += cellWidth;
                        }
                }
        }
```

```
	pDC->SelectObject(pOldFont);
}

	if (pDoc->m_bPin)
	{
		CFont font;
		font.CreatePointFont(90, L"Arial");
		CFont* pOldFont = pDC->SelectObject(&font);

		int imageHeight = CreateImage(pDC, Width, L"res\\Pin.jpg");
		int x = 0;
		int y = imageHeight + 15;

		// Рисование шапки таблицы
		{
			CString PinBuff[6] = { L"№ исполнения", L"d", L"d1", L"l",
L"c", L"Масса 1000 шт, кг" };

			CRect HeaderRect(x, y, x + headingWidth, y + cellHeight);
			pDC->Rectangle(HeaderRect);
			pDC->DrawText(PinBuff[0], HeaderRect, DT_SINGLELINE |
DT_CENTER | DT_VCENTER);
			x += headingWidth;

			for (int i = 1; i < 6; i++)
			{
				CRect HeaderRect(x, y, x + cellWidth, y + cellHeight);
				pDC->Rectangle(HeaderRect);
				pDC->DrawText(PinBuff[i], HeaderRect, DT_SINGLELINE |
DT_CENTER | DT_VCENTER);
				x += cellWidth;
			}
		}

		// Заполнение таблицы
		{
			for (int i = 0; i < vecPin.size(); i++)
			{
				y += cellHeight;
				x = 0;

				CRect ButtonRect(x, y, x + headingWidth, y +
cellHeight);


				m_Buttons[i].MoveWindow(ButtonRect);
				m_Buttons[i].ShowWindow(SW_SHOW);

				x += headingWidth;

				for (int g = 0; g < 5; g++)
				{
					CRect HeaderRect(x, y, x + cellWidth, y +
cellHeight);
					pDC->Rectangle(HeaderRect);
					pDC->DrawText(vecPin[i].name[g], HeaderRect,
DT_SINGLELINE | DT_CENTER | DT_VCENTER);
					x += cellWidth;
				}
			}
		}

		pDC->SelectObject(pOldFont);
	}
}
```

```
// Печать CCourseProjectView

BOOL CCourseProjectView::OnPreparePrinting(CPrintInfo* pInfo)
{
	// подготовка по умолчанию
	return DoPreparePrinting(pInfo);
}

void CCourseProjectView::OnBeginPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
{
	// TODO: добавьте дополнительную инициализацию перед печатью
}

void CCourseProjectView::OnEndPrinting(CDC* /*pDC*/, CPrintInfo* /*pInfo*/)
{
	// TODO: добавьте очистку после печати
}


// Диагностика CCourseProjectView

#ifdef _DEBUG
void CCourseProjectView::AssertValid() const
{
	CView::AssertValid();
}

void CCourseProjectView::Dump(CDumpContext& dc) const
{
	CView::Dump(dc);
}

CCourseProjectDoc* CCourseProjectView::GetDocument() const // встроена
неотлаженная версия
{
	ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CCourseProjectDoc)));
	return (CCourseProjectDoc*)m_pDocument;
}
#endif //_DEBUG


// Обработчики сообщений CCourseProjectView


//Обработчик создания окна
int CCourseProjectView::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
	if (CView::OnCreate(lpCreateStruct) == -1)
		return -1;

	// TODO:  Добавьте специализированный код создания
	for (int i = 0; i < 7; i++)
	{
		CString NumberButton;
		NumberButton.Format(L"Исполнение %i", i + 1);
		m_Buttons[i].Create(NumberButton, WS_CHILD | WS_VISIBLE |
BS_PUSHBUTTON, CRect(0, 0, 0, 0), this, IDC_BUTTON_NEST_1 + i);
	}


	//Исполнение 1
	{
```

```
            Bushing bushing;
            bushing.d = 18;
            bushing.d1 = 5;
            bushing.D = 32;
            bushing.L = 55;
            bushing.l = 12;
            bushing.c = 1;
            bushing.name[0] = L"31,5";
            bushing.name[1] = Conversion(bushing.d);
            bushing.name[2] = Conversion(bushing.d1);
            bushing.name[3] = Conversion(bushing.D);
            bushing.name[4] = Conversion(bushing.L);
            bushing.name[5] = Conversion(bushing.l);
            bushing.name[6] = Conversion(bushing.c);
            bushing.name[7] = L"0,20";
            vecBushing.push_back(bushing);

            Shaft shaft;
            shaft.D = bushing.d;
            shaft.c = bushing.c;
            shaft.d = bushing.d1;
            shaft.L = (bushing.L - 2) / 2.f + bushing.L * 0.15;
            shaft.l = bushing.L / 2.f - bushing.l - 1;
            vecShaft.push_back(shaft);

            Pin pin;
            pin.d = bushing.d1;
            pin.c = 0.8;
            pin.l = bushing.D;
            pin.d1 = pin.d + pin.l / 50.f;
            pin.name[0] = Conversion(pin.d);
            pin.name[1] = Conversion(pin.d1);
            pin.name[2] = Conversion(pin.l);
            pin.name[3] = Conversion(pin.c);
            pin.name[4] = L"2,55";
            vecPin.push_back(pin);

            Coupling coupling;
            coupling.name[0] = Conversion(bushing.D);
            coupling.name[1] = Conversion(bushing.d);
            coupling.name[2] = Conversion(bushing.L);
            vecCoupling.push_back(coupling);
      }

      //Исполнение 2
      {
            Bushing bushing;
            bushing.d = 19;
            bushing.d1 = 5;
            bushing.D = 32;
            bushing.L = 55;
            bushing.l = 12;
            bushing.c = 1;
            bushing.name[0] = L"31,5";
            bushing.name[1] = Conversion(bushing.d);
            bushing.name[2] = Conversion(bushing.d1);
            bushing.name[3] = Conversion(bushing.D);
            bushing.name[4] = Conversion(bushing.L);
            bushing.name[5] = Conversion(bushing.l);
            bushing.name[6] = Conversion(bushing.c);
            bushing.name[7] = L"0,17";
            vecBushing.push_back(bushing);

            Shaft shaft;
            shaft.D = bushing.d;
```

```
            shaft.c = bushing.c;
            shaft.d = bushing.d1;
            shaft.L = (bushing.L - 2) / 2.f + bushing.L * 0.15;
            shaft.l = bushing.L / 2.f - bushing.l - 1;
            vecShaft.push_back(shaft);

            Pin pin;
            pin.d = bushing.d1;
            pin.c = 0.8;
            pin.l = bushing.D;
            pin.d1 = pin.d + pin.l / 50.f;
            pin.name[0] = Conversion(pin.d);
            pin.name[1] = Conversion(pin.d1);
            pin.name[2] = Conversion(pin.l);
            pin.name[3] = Conversion(pin.c);
            pin.name[4] = L"2,55";
            vecPin.push_back(pin);

            Coupling coupling;
            coupling.name[0] = Conversion(bushing.D);
            coupling.name[1] = Conversion(bushing.d);
            coupling.name[2] = Conversion(bushing.L);
            vecCoupling.push_back(coupling);
        }
//Исполнение 3
{
        Bushing bushing;
        bushing.d = 20;
        bushing.d1 = 5;
        bushing.D = 32;
        bushing.L = 55;
        bushing.l = 12;
        bushing.c = 1;
        bushing.name[0] = L"31,5";
        bushing.name[1] = Conversion(bushing.d);
        bushing.name[2] = Conversion(bushing.d1);
        bushing.name[3] = Conversion(bushing.D);
        bushing.name[4] = Conversion(bushing.L);
        bushing.name[5] = Conversion(bushing.l);
        bushing.name[6] = Conversion(bushing.c);
        bushing.name[7] = L"0,16";
        vecBushing.push_back(bushing);

        Shaft shaft;
        shaft.D = bushing.d;
        shaft.c = bushing.c;
        shaft.d = bushing.d1;
        shaft.L = (bushing.L - 2) / 2.f + bushing.L * 0.15;
        shaft.l = bushing.L / 2.f - bushing.l - 1;
        vecShaft.push_back(shaft);

        Pin pin;
        pin.d = bushing.d1;
        pin.c = 0.8;
        pin.l = bushing.D;
        pin.d1 = pin.d + pin.l / 50.f;
        pin.name[0] = Conversion(pin.d);
        pin.name[1] = Conversion(pin.d1);
        pin.name[2] = Conversion(pin.l);
        pin.name[3] = Conversion(pin.c);
        pin.name[4] = L"2,55";
        vecPin.push_back(pin);

        Coupling coupling;
        coupling.name[0] = Conversion(bushing.D);
```

```
	coupling.name[1] = Conversion(bushing.d);
	coupling.name[2] = Conversion(bushing.L);
	vecCoupling.push_back(coupling);
}

//Исполнение 4
{
	Bushing bushing;
	bushing.d = 20;
	bushing.d1 = 6;
	bushing.D = 38;
	bushing.L = 65;
	bushing.l = 15;
	bushing.c = 1;
	bushing.name[0] = L"50,0";
	bushing.name[1] = Conversion(bushing.d);
	bushing.name[2] = Conversion(bushing.d1);
	bushing.name[3] = Conversion(bushing.D);
	bushing.name[4] = Conversion(bushing.L);
	bushing.name[5] = Conversion(bushing.l);
	bushing.name[6] = Conversion(bushing.c);
	bushing.name[7] = L"0,34";
	vecBushing.push_back(bushing);

	Shaft shaft;
	shaft.D = bushing.d;
	shaft.c = bushing.c;
	shaft.d = bushing.d1;
	shaft.L = (bushing.L - 2) / 2.f + bushing.L * 0.15;
	shaft.l = bushing.L / 2.f - bushing.l - 1;
	vecShaft.push_back(shaft);

	Pin pin;
	pin.d = bushing.d1;
	pin.c = 1;
	pin.l = bushing.D;
	pin.d1 = pin.d + pin.l / 50.f;
	pin.name[0] = Conversion(pin.d);
	pin.name[1] = Conversion(pin.d1);
	pin.name[2] = Conversion(pin.l);
	pin.name[3] = Conversion(pin.c);
	pin.name[4] = L"4,60";
	vecPin.push_back(pin);

	Coupling coupling;
	coupling.name[0] = Conversion(bushing.D);
	coupling.name[1] = Conversion(bushing.d);
	coupling.name[2] = Conversion(bushing.L);
	vecCoupling.push_back(coupling);
}

//Исполнение 5
{
	Bushing bushing;
	bushing.d = 22;
	bushing.d1 = 6;
	bushing.D = 38;
	bushing.L = 65;
	bushing.l = 15;
	bushing.c = 1;
	bushing.name[0] = L"50,0";
	bushing.name[1] = Conversion(bushing.d);
	bushing.name[2] = Conversion(bushing.d1);
	bushing.name[3] = Conversion(bushing.D);
	bushing.name[4] = Conversion(bushing.L);
```

```
        bushing.name[5] = Conversion(bushing.l);
        bushing.name[6] = Conversion(bushing.c);
        bushing.name[7] = L"0,30";
        vecBushing.push_back(bushing);

        Shaft shaft;
        shaft.D = bushing.d;
        shaft.c = bushing.c;
        shaft.d = bushing.d1;
        shaft.L = (bushing.L - 2) / 2.f + bushing.L * 0.15;
        shaft.l = bushing.L / 2.f - bushing.l - 1;
        vecShaft.push_back(shaft);

        Pin pin;
        pin.d = bushing.d1;
        pin.c = 1;
        pin.l = bushing.D;
        pin.d1 = pin.d + pin.l / 50.f;
        pin.name[0] = Conversion(pin.d);
        pin.name[1] = Conversion(pin.d1);
        pin.name[2] = Conversion(pin.l);
        pin.name[3] = Conversion(pin.c);
        pin.name[4] = L"4,60";
        vecPin.push_back(pin);

        Coupling coupling;
        coupling.name[0] = Conversion(bushing.D);
        coupling.name[1] = Conversion(bushing.d);
        coupling.name[2] = Conversion(bushing.L);
        vecCoupling.push_back(coupling);
}

//Исполнение 6
{
        Bushing bushing;
        bushing.d = 24;
        bushing.d1 = 6;
        bushing.D = 38;
        bushing.L = 65;
        bushing.l = 15;
        bushing.c = 1;
        bushing.name[0] = L"50,0";
        bushing.name[1] = Conversion(bushing.d);
        bushing.name[2] = Conversion(bushing.d1);
        bushing.name[3] = Conversion(bushing.D);
        bushing.name[4] = Conversion(bushing.L);
        bushing.name[5] = Conversion(bushing.l);
        bushing.name[6] = Conversion(bushing.c);
        bushing.name[7] = L"0,27";
        vecBushing.push_back(bushing);

        Shaft shaft;
        shaft.D = bushing.d;
        shaft.c = bushing.c;
        shaft.d = bushing.d1;
        shaft.L = (bushing.L - 2) / 2.f + bushing.L * 0.15;
        shaft.l = bushing.L / 2.f - bushing.l - 1;
        vecShaft.push_back(shaft);

        Pin pin;
        pin.d = bushing.d1;
        pin.c = 1;
        pin.l = bushing.D;
        pin.d1 = pin.d + pin.l / 50.f;
        pin.name[0] = Conversion(pin.d);
```

```
	pin.name[1] = Conversion(pin.d1);
	pin.name[2] = Conversion(pin.l);
	pin.name[3] = Conversion(pin.c);
	pin.name[4] = L"4,60";
	vecPin.push_back(pin);
		Coupling coupling;
		coupling.name[0] = Conversion(bushing.D);
		coupling.name[1] = Conversion(bushing.d);
		coupling.name[2] = Conversion(bushing.L);
		vecCoupling.push_back(coupling);
	}



	return 0;
}

//Обработчик нажания на одну из шести кнопок
void CCourseProjectView::OnButtonClick(UINT nID)
{
	//Получаем индекс кнопки в массиве m_Buttons
	int buttonIndex = nID - IDC_BUTTON_NEST_1;

	//Задаем путь куда будут сохранятся файлы:
	m_sfilePath = L"C:\\Users\\minch\\Desktop\\Курсовой проект\\Исполнения";
	CString numberIsp;
	numberIsp.Format(L"Исполнение %i", buttonIndex + 1);
	CString targetPath = m_sfilePath + L"\\" + numberIsp;

	//Создание папки с деталями по исполнениям
	if (!CreateDirectory(targetPath, NULL))
	{
		DWORD error = GetLastError();
		if (error == ERROR_ALREADY_EXISTS)
		{
			int index = 1;
			while (true)
			{
				CString indexStr;
				indexStr.Format(L"%i", index);
				CString newFolderName = numberIsp + L" (" + indexStr +
L")";
				CString newPath = m_sfilePath + L"\\" + newFolderName;
				if (CreateDirectory(newPath, NULL))
				{
					m_sfilePath = newPath;
					break;
				}
				index++;
			}
		}
		else
			AfxMessageBox(L"Ошибка при создании папки");
	}
	else
	{
		CreateDirectory(targetPath, NULL);
		m_sfilePath = targetPath;
	}

	//Вызываем методы построения деталей и передаем в качестве параметра
экземпляр структуры из массива по индексу buttonIndex
	CreateBushing(vecBushing[buttonIndex]);
	CreateShaft(vecShaft[buttonIndex]);
	CreatePin(vecPin[buttonIndex]);
```

```
	CreateCoupling(vecPin[buttonIndex], vecShaft[buttonIndex],
vecBushing[buttonIndex]);
}

// Метод для создания штифта
void CCourseProjectView::CreatePin(Pin& st)
{
	CoInitialize(NULL);
	HRESULT hRes;
	hRes = kompas.GetActiveObject(L"Kompas.Application.5");
	if (FAILED(hRes))
		kompas.CreateInstance(L"Kompas.Application.5");
	kompas->Visible = true;

	//создание документа КОМПАС:
	ksDocument3DPtr doc;
	doc = kompas->Document3D();
	doc->Create(false, true);
	doc = kompas->ActiveDocument3D();
	ksPartPtr part;
	part = doc->GetPart(pTop_Part);

	//эскиз под вращение:
	ksEntityPtr sketch = part->NewEntity(o3d_sketch);
	ksSketchDefinitionPtr sketchDef = sketch->GetDefinition();
	sketchDef->SetPlane(part->GetDefaultEntity(o3d_planeXOY));
	sketch->Create();
	ksDocument2DPtr doc2D = sketchDef->BeginEdit();
	doc2D->ksLineSeg(0, 0, 0, st.d / 2.f, 1);
	doc2D->ksLineSeg(0, st.d / 2.f, st.l, st.d1 / 2.f, 1);
	doc2D->ksLineSeg(st.l, st.d1 / 2.f, st.l, 0, 1);
	doc2D->ksLineSeg(st.l, 0, 0, 0, 1);
	doc2D->ksLineSeg(0, 0, 10, 0, 3); //осевая
	sketchDef->EndEdit();

	//операция вращения:
	ksEntityPtr rotation = part->NewEntity(o3d_bossRotated);
	ksBossRotatedDefinitionPtr rotationDef = rotation->GetDefinition();
	rotationDef->SetSideParam(TRUE, 360);
	rotationDef->directionType = dtNormal;
	rotationDef->SetSketch(sketch);
	rotation->Create();

	//операция фаска
	ksEntityPtr Chamfers = part->NewEntity(o3d_chamfer);
	ksChamferDefinitionPtr ChamfersDef = Chamfers->GetDefinition();
	ChamfersDef->SetChamferParam(true, st.c, st.c);
	ksEntityCollectionPtr Collection = part->EntityCollection(o3d_edge);
	ksEntityCollectionPtr CollectionChamfers = ChamfersDef->array();
	Collection->SelectByPoint(0, st.d / 2.f, 0);
	CollectionChamfers->Add(Collection->First());
	Collection->Clear();
	Collection = part->EntityCollection(o3d_edge);
	Collection->SelectByPoint(st.l, st.d1 / 2.f, 0);
	CollectionChamfers->Add(Collection->First());
	Chamfers->Create();

	//операция сохранения детали
	CString name = L"Штифт";
	st.filePath = m_sfilePath;
	doc->fileName = _bstr_t(name);
	st.filePath += L"\\" + name + L".m3d";
	doc->SaveAs(_bstr_t(st.filePath));
	if (!PathFileExists(m_sfilePath)) {
		AfxMessageBox(L"Папка для сохранения не существует!");
```

```
            return;
        }
}
// Метод для создания вала
void CCourseProjectView::CreateShaft(Shaft& st)
{
        CoInitialize(NULL);
        HRESULT hRes;
        hRes = kompas.GetActiveObject(L"Kompas.Application.5");
        if (FAILED(hRes))
              kompas.CreateInstance(L"Kompas.Application.5");
        kompas->Visible = true;

        //создание документа КОМПАС:
        ksDocument3DPtr doc;
        doc = kompas->Document3D();
        doc->Create(false, true);
        doc = kompas->ActiveDocument3D();
        ksPartPtr part;
        part = doc->GetPart(pTop_Part);

        //эскиз под выдавливание
        ksEntityPtr sketch = part->NewEntity(o3d_sketch);
        ksSketchDefinitionPtr sketchDef = sketch->GetDefinition();
        sketchDef->SetPlane(part->GetDefaultEntity(o3d_planeXOY));
        sketch->Create();
        ksDocument2DPtr doc2D = sketchDef->BeginEdit();
        doc2D->ksCircle(0, 0, st.D / 2.f, 1);
        sketchDef->EndEdit();

        //операция выдавливания
        ksEntityPtr baseExtr = part->NewEntity(o3d_baseExtrusion);
        ksBaseExtrusionDefinitionPtr ExtrDef = baseExtr->GetDefinition();
        ExtrDef->SetSideParam(TRUE, etBlind, st.L, 0, TRUE);
        ExtrDef->SetSketch(sketch);
        baseExtr->Create();

        //операция фаска
        ksEntityPtr Chamfers = part->NewEntity(o3d_chamfer);
        ksChamferDefinitionPtr ChamfersDef = Chamfers->GetDefinition();
        ChamfersDef->SetChamferParam(true, st.c, st.c);
        ksEntityCollectionPtr Collection = part->EntityCollection(o3d_edge);
        Collection->SelectByPoint(st.D / 2.f, 0, st.L);
        ksEntityCollectionPtr CollectionChamfers = ChamfersDef->array();
        CollectionChamfers->Add(Collection->First());
        Chamfers->Create();

        //операция смещенная плоскость
        ksEntityPtr planeXOZ = part->GetDefaultEntity(o3d_planeXOZ);
        ksEntityPtr plane = part->NewEntity(o3d_planeOffset);
        ksPlaneOffsetDefinitionPtr planeDef = plane->GetDefinition();
        planeDef->direction = TRUE;
        planeDef->offset = st.D / 2.f;
        planeDef->SetPlane(planeXOZ);
        plane->Create();

        //эскиз под отверстия
        ksEntityPtr sketch2 = part->NewEntity(o3d_sketch);
        ksSketchDefinitionPtr sketchDef2 = sketch2->GetDefinition();
        sketchDef2->SetPlane(plane);
        sketch2->Create();
        ksDocument2DPtr doc2D2 = sketchDef2->BeginEdit();
        doc2D2->ksCircle(0, -(st.L - st.l), st.d / 2.f, 1);
        sketchDef2->EndEdit();
```

```
	//операция вырез выдавливанием
	ksEntityPtr CutExtr = part->NewEntity(o3d_cutExtrusion);
	ksCutExtrusionDefinitionPtr CutExtrDef = CutExtr->GetDefinition();
	CutExtrDef->directionType = dtNormal;
	CutExtrDef->SetSketch(sketch2);
	CutExtrDef->SetSideParam(TRUE, etBlind, st.D, 0, FALSE);
	CutExtr->Create();

	//операция сохранения детали
	CString name = L"Вал";
	st.filePath = m_sfilePath;
	doc->fileName = _bstr_t(name);
	st.filePath += L"\\" + name + L".m3d";
	doc->SaveAs(_bstr_t(st.filePath));
}
// Метод для создания втулки
void CCourseProjectView::CreateBushing(Bushing& st)
{
	CoInitialize(NULL);
	HRESULT hRes;
	hRes = kompas.GetActiveObject(L"Kompas.Application.5");
	if (FAILED(hRes))
		kompas.CreateInstance(L"Kompas.Application.5");
	kompas->Visible = true;

	//создание документа КОМПАС:
	ksDocument3DPtr doc;
	doc = kompas->Document3D();
	doc->Create(false, true);
	doc = kompas->ActiveDocument3D();
	ksPartPtr part;
	part = doc->GetPart(pTop_Part);

	//эскиз под выдавливание
	ksEntityPtr sketch = part->NewEntity(o3d_sketch);
	ksSketchDefinitionPtr sketchDef = sketch->GetDefinition();
	sketchDef->SetPlane(part->GetDefaultEntity(o3d_planeXOY));
	sketch->Create();
	ksDocument2DPtr doc2D = sketchDef->BeginEdit();
	doc2D->ksCircle(0, 0, st.D / 2.f, 1);
	doc2D->ksCircle(0, 0, st.d / 2.f, 1);
	sketchDef->EndEdit();

	//операция выдавливания
	ksEntityPtr baseExtr = part->NewEntity(o3d_baseExtrusion);
	ksBaseExtrusionDefinitionPtr ExtrDef = baseExtr->GetDefinition();
	ExtrDef->SetSideParam(TRUE, etBlind, st.L, 0, TRUE);
	ExtrDef->SetSketch(sketch);
	baseExtr->Create();

	//операция смещенная плоскость
	ksEntityPtr planeXOZ = part->GetDefaultEntity(o3d_planeXOZ);
	ksEntityPtr plane = part->NewEntity(o3d_planeOffset);
	ksPlaneOffsetDefinitionPtr planeDef = plane->GetDefinition();
	planeDef->direction = TRUE;
	planeDef->offset = st.D / 2.f;
	planeDef->SetPlane(planeXOZ);
	plane->Create();

	//эскиз под отверстия
	ksEntityPtr sketch2 = part->NewEntity(o3d_sketch);
	ksSketchDefinitionPtr sketchDef2 = sketch2->GetDefinition();
	sketchDef2->SetPlane(plane);
	sketch2->Create();
	ksDocument2DPtr doc2D2 = sketchDef2->BeginEdit();
```

```
	doc2D2->ksCircle(0, -st.l, st.d1 / 2.f, 1);
	doc2D2->ksCircle(0, -(st.L - st.l), st.d1 / 2.f, 1);
	sketchDef2->EndEdit();

	//операция вырез выдавливанием
	ksEntityPtr CutExtr = part->NewEntity(o3d_cutExtrusion);
	ksCutExtrusionDefinitionPtr CutExtrDef = CutExtr->GetDefinition();
	CutExtrDef->directionType = dtNormal;
	CutExtrDef->SetSketch(sketch2);
	CutExtrDef->SetSideParam(TRUE, etBlind, st.D, 0, FALSE);
	CutExtr->Create();

	//операция фаска
	ksEntityPtr Chamfers = part->NewEntity(o3d_chamfer);
	ksChamferDefinitionPtr ChamfersDef = Chamfers->GetDefinition();
	ChamfersDef->SetChamferParam(true, st.c, st.c);
	ksEntityCollectionPtr Collection = part->EntityCollection(o3d_edge);
	ksEntityCollectionPtr CollectionChamfers = ChamfersDef->array();
	Collection->SelectByPoint(0, st.d / 2.f, 0);
	CollectionChamfers->Add(Collection->First());
	Collection->Clear();
	Collection = part->EntityCollection(o3d_edge);
	Collection->SelectByPoint(0, st.D / 2.f, 0);
	CollectionChamfers->Add(Collection->First());
	Collection->Clear();
	Collection = part->EntityCollection(o3d_edge);
	Collection->SelectByPoint(0, st.d / 2.f, st.L);
	CollectionChamfers->Add(Collection->First());
	Collection->Clear();
	Collection = part->EntityCollection(o3d_edge);
	Collection->SelectByPoint(0, st.D / 2.f, st.L);
	CollectionChamfers->Add(Collection->First());
	Chamfers->Create();

	//операция сохранения детали
	CString name = L"Втулка";
	st.filePath = m_sfilePath;
	doc->fileName = _bstr_t(name);
	st.filePath += L"\\" + name + L".m3d";
	doc->SaveAs(_bstr_t(st.filePath));
}
void CCourseProjectView::CreateCoupling(Pin pin, Shaft shaft, Bushing bushing)
{
	CoInitialize(NULL);
	HRESULT hRes;
	hRes = kompas.GetActiveObject(L"Kompas.Application.5");
	if (FAILED(hRes))
		kompas.CreateInstance(L"Kompas.Application.5");
	kompas->Visible = true;

	//создание документа КОМПАС :
	ksDocument3DPtr doc;
	doc = kompas->Document3D();
	doc->Create(false, false);
	doc = kompas->ActiveDocument3D();
	ksPartPtr part;
	part = doc->GetPart(pTop_Part);

	//добавление детали "Втулка"
	doc->SetPartFromFile(_bstr_t(bushing.filePath), part, TRUE);
	ksPartPtr partbushing = doc->GetPart(0);
	ksEntityCollectionPtr bushingCollection = partbushing-
>EntityCollection(o3d_edge);
	//ребро 1
	ksEntityPtr edgebushing;
```

```
	bushingCollection->SelectByPoint(0, bushing.d / 2.f + bushing.c, 0);
	edgebushing = bushingCollection->First();
	//грань 1
	ksEntityPtr facebushing;
	bushingCollection->Clear();
	bushingCollection = partbushing->EntityCollection(o3d_face);
	bushingCollection->SelectByPoint(bushing.d1 / 2.f, (bushing.D - bushing.d)
/ 4.f + bushing.d / 2.f, bushing.l);
	facebushing = bushingCollection->First();
	//грань 2
	ksEntityPtr facebushing2;
	bushingCollection->Clear();
	bushingCollection = partbushing->EntityCollection(o3d_face);
	bushingCollection->SelectByPoint(bushing.d1 / 2.f, (bushing.D - bushing.d)
/ 4.f + bushing.d / 2.f, bushing.L - bushing.l);
	facebushing2 = bushingCollection->First();
	//грань 3
	ksEntityPtr facebushing3;
	bushingCollection->Clear();
	bushingCollection = partbushing->EntityCollection(o3d_face);
	bushingCollection->SelectByPoint(0, bushing.D / 2.f, bushing.L / 2.f);
	facebushing3 = bushingCollection->First();

	//добавление детали "Вал" x1
	doc->SetPartFromFile(_bstr_t(shaft.filePath), part, TRUE);
	ksPartPtr partshaft = doc->GetPart(1);
	ksEntityCollectionPtr shaftCollection = partshaft-
>EntityCollection(o3d_edge);
	//ребро 1
	ksEntityPtr edgeshaft;
	shaftCollection->SelectByPoint(0, shaft.D / 2.f, 0);
	edgeshaft = shaftCollection->First();
	//грань
	ksEntityPtr faceshaft;
	shaftCollection->Clear();
	shaftCollection = partshaft->EntityCollection(o3d_face);
	shaftCollection->SelectByPoint(shaft.d / 2.f, 0, shaft.L - shaft.l);
	faceshaft = shaftCollection->First();

	//добавление детали "Вал" x2
	doc->SetPartFromFile(_bstr_t(shaft.filePath), part, TRUE);
	ksPartPtr partshaft2 = doc->GetPart(2);
	ksEntityCollectionPtr shaftCollection2 = partshaft2-
>EntityCollection(o3d_edge);
	//ребро 1
	ksEntityPtr edgeshaft2;
	shaftCollection2->SelectByPoint(0, shaft.D / 2.f, 0);
	edgeshaft2 = shaftCollection2->First();
	//грань
	ksEntityPtr faceshaft2;
	shaftCollection2->Clear();
	shaftCollection2 = partshaft2->EntityCollection(o3d_face);
	shaftCollection2->SelectByPoint(shaft.d / 2.f, 0, shaft.L - shaft.l);
	faceshaft2 = shaftCollection2->First();

	//добавление детали "Штифт" x1
	doc->SetPartFromFile(_bstr_t(pin.filePath), part, TRUE);
	ksPartPtr partpin = doc->GetPart(3);
	ksEntityCollectionPtr pinCollection = partpin->EntityCollection(o3d_face);
	//грань
	ksEntityPtr facepin;
	pinCollection->SelectByPoint(pin.l, 0, 0);
	facepin = pinCollection->First();
	//ребро
	ksEntityPtr edgepin;
```

```
	pinCollection->Clear();
	pinCollection = partpin->EntityCollection(o3d_edge);
	pinCollection->SelectByPoint(pin.l, pin.d1 / 2.f - pin.c, 0);
	edgepin = pinCollection->First();

	//добавление детали "Штифт" x2
	doc->SetPartFromFile(_bstr_t(pin.filePath), part, TRUE);
	ksPartPtr partpin2 = doc->GetPart(4);
	ksEntityCollectionPtr pinCollection2 = partpin2-
>EntityCollection(o3d_face);
	//грань
	ksEntityPtr facepin2;
	pinCollection2->SelectByPoint(pin.l, 0, 0);
	facepin2 = pinCollection2->First();
	//ребро
	ksEntityPtr edgepin2;
	pinCollection2->Clear();
	pinCollection2 = partpin2->EntityCollection(o3d_edge);
	pinCollection2->SelectByPoint(pin.l, pin.d1 / 2.f - pin.c, 0);
	edgepin2 = pinCollection2->First();

	//Установка сопряжений
	doc->AddMateConstraint(mc_Concentric, edgebushing, edgeshaft, 1, 0, NULL);
	doc->AddMateConstraint(mc_Concentric, edgebushing, edgeshaft2, -1, 0,
NULL);
	doc->AddMateConstraint(mc_Concentric, facebushing, faceshaft, 1, 0, NULL);
	doc->AddMateConstraint(mc_Concentric, facebushing2, faceshaft2, 1, 0,
NULL);
	doc->AddMateConstraint(mc_Concentric, facebushing, edgepin, 1, 0, NULL);
	doc->AddMateConstraint(mc_Tangency, facepin, facebushing3, -1, 0, NULL);
	doc->AddMateConstraint(mc_Concentric, facebushing2, edgepin2, -1, 0, NULL);
	doc->AddMateConstraint(mc_Tangency, facepin2, facebushing3, -1, 0, NULL);

	//операция сохранения детали
	CString name = L"Втулочная муфта";
	CString filePath = m_sfilePath;
	doc->fileName = _bstr_t(name);
	filePath += L"\\" + name + L".a3d";
	doc->SaveAs(_bstr_t(filePath));
}
// Метод для создания изображения
int CCourseProjectView::CreateImage(CDC* pDC, int Width, CString name_file)
{
	CImage image;
	image.Load(name_file);

	if (image.IsNull())
	{
		AfxMessageBox(L"Ошибка загрузки картинки");
		return 10;
	}

	int imageWidth = image.GetWidth();		//Ширина картинки
	int imageHeight = image.GetHeight();  //Высота картинки

	image.StretchBlt
	(
		pDC->GetSafeHdc(), Width / 2.f - imageWidth / 2.f, 0,
		imageWidth, imageHeight, 0, 0,
		imageWidth, imageHeight, SRCCOPY
	);

	return imageHeight;

}
```

```
// Служебный метод для преобразования числа в строку
CString CCourseProjectView::Conversion(double number)
{
	CString buff;
	buff.Format(L"%f", number);

	int dotIndex = buff.Find(L".");

	if (dotIndex != -1)
	{
		int Index = buff.GetLength() - 1;

		while (Index > dotIndex && buff[Index] == _T('0'))
			Index--;

		if (Index > dotIndex)
			buff = buff.Left(Index + 1);
		else
			buff = buff.Left(dotIndex);
	}

	buff.Replace(L".", L",");

	return buff;
}
```