Министерство науки и высшего образования Российской Федерации
Чайковский филиал
федерального государственного автономного
образовательного учреждения высшего образования
«Пермский национальный исследовательский
политехнический университет»

Кафедра Автоматизации, информационных и инженерных технологий

МП 12.8-2022

МЕТОДИЧЕСКИЕ ПРЕДПИСАНИЯ К КУРСОВОЙ РАБОТЕ ПО ДИСЦИПЛИНЕ «СИСТЕМЫ АВТОМАТИЗИРОВАННОГО ПРОЕКТИРОВАНИЯ»

Направление: 09.03.01 Информатика и вычислительная техника

Очная форма обучения
Заочная форма обучения

Чайковский, 2022

Красильников. С.Н. Методические предписания по выполнению курсовой работы по дисциплине «Системы автоматизированного проектирования » / Сост. к. техн. наук С.Н. Красильников – Чайковский: Пермский национальный исследовательский политехнический университет, 2022. – 39 с.

Методические предписания предназначены для выполнения курсовой работы по дисциплине « Системы автоматизированного проектирования ». В методических предписаниях кратко излагается необходимая теория, порядок выполнения и оформления курсовой работы, варианты заданий и рекомендуемая литература.

Методические предписания рекомендованы обучающимся по образовательным программам высшего образования по направлениям подготовки по направлениям подготовки 09.03.01 Информатика и вычислительная техника

Рецензент: доктор техн. наук, профессор кафедры АИИТ Е.А. Морозов

Методические предписания для студентов по выполнению курсовой работы рассмотрены и одобрены на заседании кафедры Автоматизации, информационных и инженерных технологий ЧФ ПНИПУ 26.09.2022 г., протокол № 4.

Методические предписания для студентов по выполнению курсовой работы рекомендованы методической комиссией ЧФ ПНИПУ для использования в учебном процессе (протокол № 1 от 29.09.2022 г.)

СОДЕРЖАНИЕ

1. ЦЕЛЬ И ЗАДАЧИ КУРСОВОЙ РАБОТЫ

Целью курсовой работы: Исследование современных САПР. Практическое освоение приемов создания специализированных САПР, основанных на функционале универсальной САПР (КОМПАС).

Основными задачами курсовой работы являются:

- поиск, анализ и обработка информации по указанной теме;
- классификация и конструктивные особенности современной САПР;
- применение САПР на производстве;
- построение структуры САПР;
- построение математической модели;
- разработка специализированной САПР.

Курсовая работа состоит из 2х частей:

1. Теоретические сведения о предмете исследования.
2. Создание САПР на базе Компас 3D.

Первая часть работы заключается в поиске, анализе и обработке информации по указанной теме и представляет собой реферативное сообщение об предмете исследования. Первая часть курсовой работы должна содержать сведения, актуальные на сегодняшний день и описывающие предмет исследования с позиций классификации, конструктивных особенностей, принципов функционирования и применения САПР на производстве.

Вторая, практическая часть заключается в создание собственной САПР на базе Компас 3D.

Разработать САПР создающую по минимальному количеству исходных данных 3D - модель зубчатого колеса.

Необходимо разработать структуру объекта (поля и методы). Разработать головную программу, обеспечивающую ввод/вывод данных и взаимодействие с объектом, а также графическое отображение информации. Разработать процедуры взаимодействия программы с КОМПАС.

Необходимо чтобы после вызова библиотеки пользователем появляется диалоговое окно, в котором он задает модуль, количество зубьев, ширину зубчатого венца, а также угол наклоназубьев колеса. По введенным параметрам, после нажатия кнопки Построение, библиотека должна сгенерировать трехмерную модель колеса.

Структурными элементами курсовой работы являются:

1. Титульный лист.
2. Содержание
3. Введение
4. Основная часть.
5. Выводы (заключение)
6. Список литературы.
7. Приложение.

Титульный лист должен содержать наименование темы, фамилию и инициалы студента, наименование группы, фамилию и инициалы преподавателя, год выполнения работы. Пример оформления приведен в приложении А.

Введение–обосновывается выбор темы, определяемый её актуальностью, формулируется её проблема и круг вопросов, необходимых для её решения. Здесь же

определяется цель работы с её расчленением на взаимосвязанный комплекс задач, подлежащих решению, для раскрытия темы. **Актуальность →Цель → Задачи.**

Основная часть–содержит несколько глав, каждая из которых делится на параграфы. В данной части работы излагается основной материал темы. Основное требование: четкость изложения, логичные переходы между параграфами, отсутствие второстепенных деталей (это не способ чтобы получить требуемый объем).

Заключение–излагаются выводы и предложения, полученные студентом в процессе выполнения курсовой работы. Выводы пишутся в тезисном стиле (по пунктам).

Список литературы–включает список используемых источников в количестве не менее 10.

Приложения–включает вспомогательный материал и данные, не вынесенные в основную часть, но являющиеся необходимыми при реализации задачи курсовойработы. Например: схемы, графики, объемные таблицы, исходный код программы, дополнительные иллюстрации.

Объем курсовой работы должен быть от30до40-страниц печатного текста (не включая приложений).

2. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ
ЗУБЧАТЫЕ ПЕРЕДАЧИ

Зубчатые передачи являются наиболее распространёнными типами механических передач. Они находят широкое применение во всех отраслях машиностроения, в частности в металлорежущих станках, автомобилях, тракторах, сельхозмашинах и т.д., в приборостроении, часовой промышленности и др. Их применяют для передачи мощностей от долей до десятков тысяч киловатт при окружных скоростях до 150 м/с и передаточных числах до нескольких сотен и даже тысяч, с диаметром колёс от долей миллиметра до 6 м и более.

Зубчатая передача относиться к передачам зацеплением с непосредственным контактом пары зубчатых колёс. Меньшее из колёс передачи принято называть шестерней, а большее – колесом. Зубчатая передача предназначена в основном для передачи вращательного движения.

Классификация зубчатых передач

1. По взаимному расположению геометрических осей валов различают передачи:
   a. с параллельными осями – цилиндрические (рис. 2.1 а-г);
   b. с пересекающимися осями – конические (рис. 2.1 д; е);
   c. со скрещивающимися осями – цилиндрические винтовые (рис.2.1 ж);
   d. конические гипоидные и червячные (рис. 2.1 з);
   e. реечная передача (рис. 2.1 и).

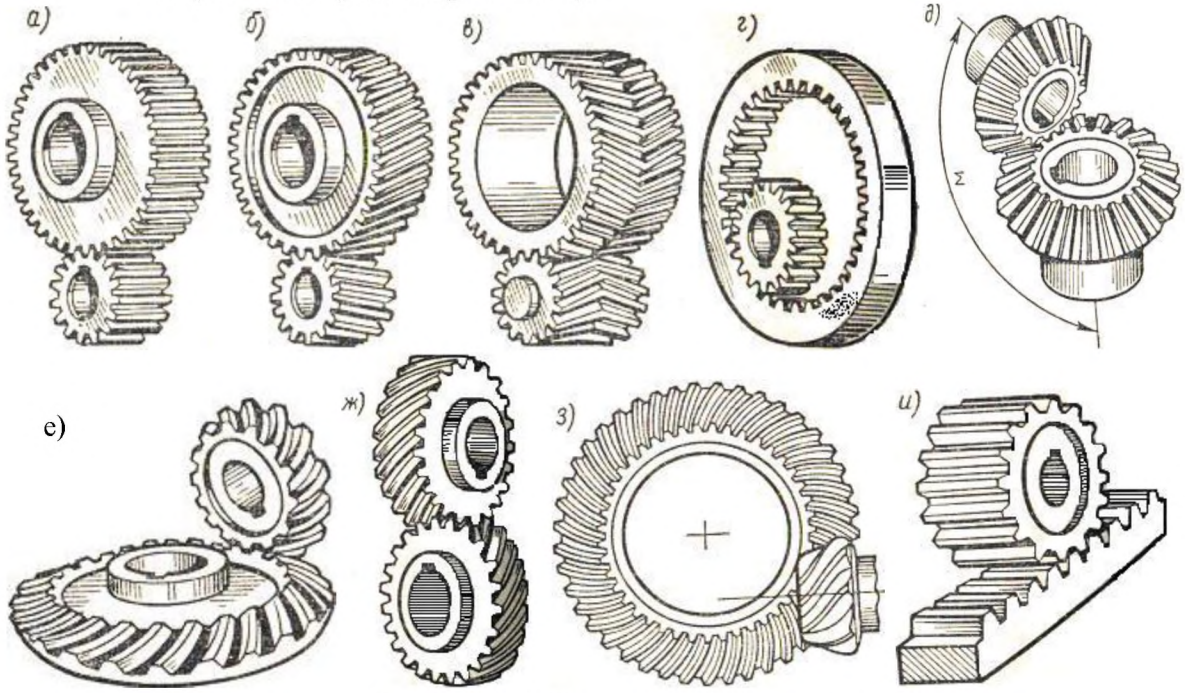

Рисунок 2.1 -Виды зубчатых передач

2. В зависимости от взаимного расположения зубчатых колёс:
   a. с внешним зацеплением (колёса передач вращаются в противоположных направлениях);
   b. с внутренним зацеплением (направление вращения колёс совпадают).
3. По расположению зубьев на поверхности колёс различают передачи:
   a. прямозубые;
   b. косозубые;
   c. шевронные;
   d. с круговым зубом.

Геометрия зубчатых колес

Поверхности взаимодействующих зубьев должны обеспечивать постоянство передаточного числа. Основная теорема зацепления: общая нормаль, проведенная через точку касания профилей, делит расстояние между центрами зубчатых колес на части, обратно пропорциональные угловым скоростям (рис.2.2). Все геометрические параметры зубчатых колес стандартизованы. В прямозубой передаче зубья входят в зацепление сразу по всей длине. Это явление сопровождается ударами и шумом, сила которых возрастает с увеличением окружной скорости ν колёс. Как правило, применяется в открытом и реже в закрытом исполнении.

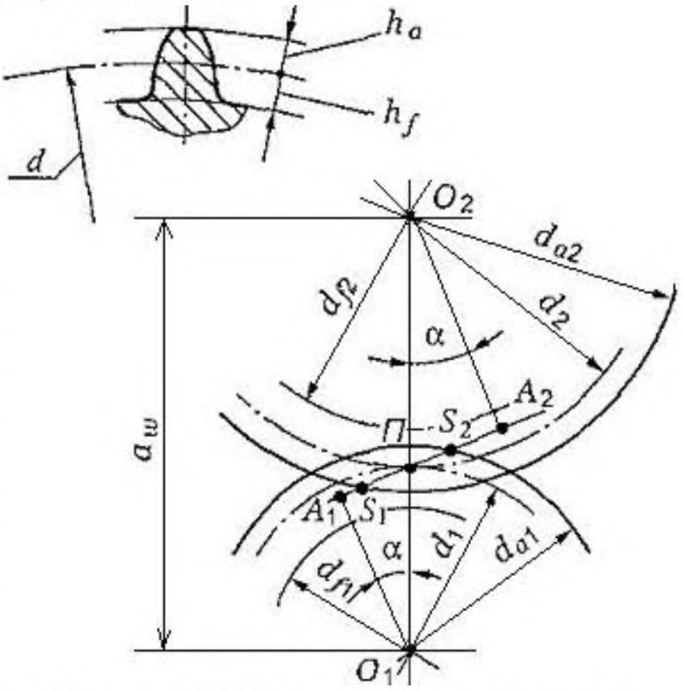

Рисунок 2.2 - Геометрические параметры зубчатых колес

Где *П* – полюс зацепления; *A1, A2*-линия зацепления, *S1, S2* – длина активной линии зацепления; α- угол зацепления; a_ω- межосевое расстояние; *d1, d2*- диаметры делительных окружностей; h_a, h_f- высота головки и ножки зуба; d_{f1}, d_{f2}- диаметры окружностей впадин, d_{a1}, d_{a2}- диаметры окружностей выступов. Основной параметр зубчатых колес – модуль *m*. Модуль равен отношению окружного шага зубьев p_t по делительной окружности к числу π

$$m = p_t / \pi \tag{2.1}$$

Делительная окружность делит зуб на две части: головку и ножку. Передаточное отношение

$$u = \omega_1 / \omega_2 = n_1 / n_2 = z_1 / z_2 \tag{2.2}$$

Значение *u* ограничивается габаритами передачи. По СТ СЭВ 229-75 значения *u* (1 ряд): 1; 1,25; 1,6; 2; 2,5; 3,15; 4; 5; 6,3 и т.д.

Для одноступенчатых стандартных редукторов не рекомендуется принимать u>5,0. Основные геометрические размеры определяют в зависимости от модуля m числа зубьев *z*: Делительная окружность - *d*, начальная окружность – *dw*. Диаметры делительный и начальный.

$$d_1 = d_\omega = mz_1 \tag{2.3}$$

$$d_2 = d_\omega = mz_2 \tag{2.4}$$

В соответствии с параметрами исходного контура зубчатой рейки получим диаметры вершин *da* и впадин *df* зубьев:

$$d_a = d + 2h_a = d + 2m \tag{2.5}$$

$$d_f = d - 2h_1 = d - 2{,}5m \tag{2.6}$$

Межосевое расстояние передачи

$$a_\omega = (d_1 + d_2)/2 = d_1(u+1)/2 = mz(u+1)/2 = m(z_2 + z_1)/2 = mz_\Sigma/2 \tag{2.7}$$

Здесь $z_\Sigma = z_1 + z_2$- суммарное число зубьев. z_Σ. Зная определяют число зубьев шестерни $z_1 = z_\Sigma/(u+1)$ и колёса $z_2 = z_\Sigma - z_1$. Значение z_1 округляют в ближайшую

сторону до целого числа. Для прямозубых колёс $z_{1\min}=17$. Значения межосевого расстояния a_ω, мм, выбирают из ряда чисел: 40, 50, 63, 80, 125, 140, 160, 180, 200, 224, 250, 280, 315, 355, 400, 450, 500, …, 2500 (СТ СЭВ – 75).

Из формулы (2.3) находим

$$d_1 = 2a_\omega/(u+1); \quad d_2 = 2a_\omega u/(u+1). \tag{2.8}$$

Ширина зубчатого венца колеса

$$b_2 = \psi_a a_\omega, \tag{2.9}$$

где ψ_a - коэффициент ширины венца колеса. Ширина венца шестерни при твёрдости рабочих поверхностей зубьев менее 350 НВ:

$$b_1 = 1{,}12 b_2, \tag{2.10}$$

Значения *b1* и *b2* принимают из ряда чисел *Ra40*. Более широкая шестерня учитывает возможное осевое смещение зубчатых колёс из-за неточности сборки, кроме того, это важно при приработке зубьев, когда более твёрдая шестерня перекрывает по ширине более мягкое колесо. При твёрдости рабочих поверхностей зубьев обоих колёс более 350 НВ принимают *b1* и *b2* (колёса не прирабатываются).

Понятие о зубчатых колесах со смещением

При заданном модуле изменение числа зубьев приводит к изменению формы зуба (рис.2.4). С уменьшением числа зубьев колеса толщина зуба в основании уменьшается, и при некотором минимальном значении z появляется подрез зуба режущей кромкой инструмента. Улучшение профиля зуба называется корригированием.

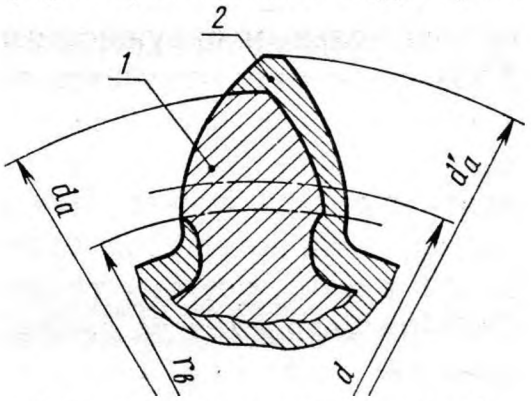

Рисунок 2.3 -Корригирование зуба

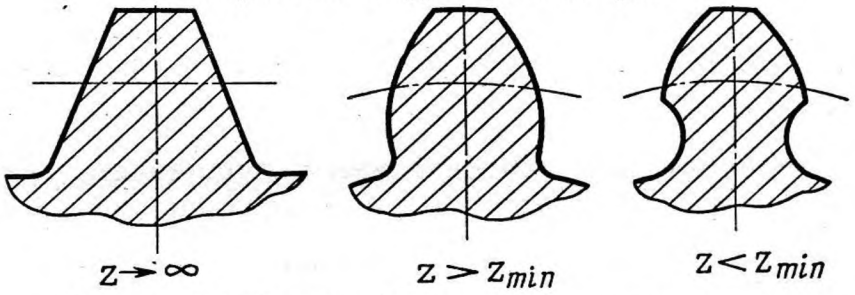

Рисунок 2.4 -Форма зубьев с различным смещением

Корригирование достигается смещением инструментальной рейки. Смещение зубьев (модификацию) применяют:

- для устранения подрезания зубьев при z < zmin;
- для повышения прочности зубьев путем увеличения их толщины;
- для увеличения радиуса в точке касания (увеличивается контактная прочность);

– для получения заданного межосевого расстояния.

При нарезании колес со смещением инструмент сдвигается от центра заготовки (положительное смещение) или к центру (отрицательное смещение) рис.2.4. Модификация бывает высотной и угловой. При высотной модификации колесо и шестерню изготовляют с противоположным смещением. Шестерню изготовляют с положительным смещением, колесо — с отрицательным смещением. Межосевое расстояние и угол зацепления не меняются. При угловой модификации суммарный коэффициент смещения отличен от нуля, а межцентровое расстояние и угол зацепления меняются. При = 20° - минимальное число зубьев 13 < z,min < 17.

Цилиндрическая прямозубая зубчатая передача

Цилиндрическая прямозубая зубчатая передача относится к передачам зацеплением непосредственного контакта рис.2.5. Применяется при окружных скоростях $v \le 2м \setminus с$

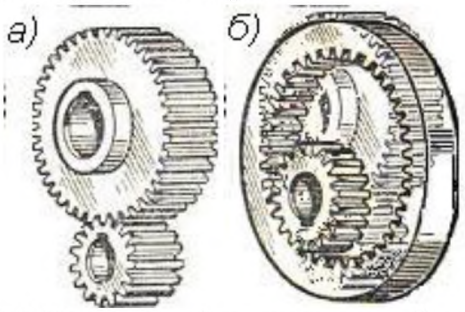

Рисунок 2.5-Наружное а) и внутреннее б) зацепление

Цилиндрическая косозубая зубчатая передача

Цилиндрические колеса, у которых зубья расположены по винтовым линиям на делительном диаметре, называют косозубыми. При работе такой передачи зубья входят в зацепление не сразу по всей длине, как в прямозубой, а постепенно; передаваемая нагрузка распределяется на несколько зубьев. В результате по сравнению с прямозубой повышается нагрузочная способность, увеличивается плавность работы передачи и уменьшается шум. Поэтому косозубые передачи имеют преимущественное распространение рис. 2.6.

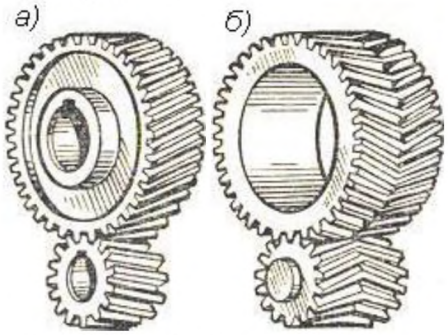

Рисунок 2.6-Цилиндрическая косозубая а) и шевронная б) передача

С увеличением угла наклона β линии зуба плавность зацепления и нагрузочная способность передачи увеличиваются рис.2.7, но при этом увеличивается и осевая сила F_a, что нежелательно. Поэтому в косозубых передачах принимают угол $\beta = 7...20^{\circ}$.

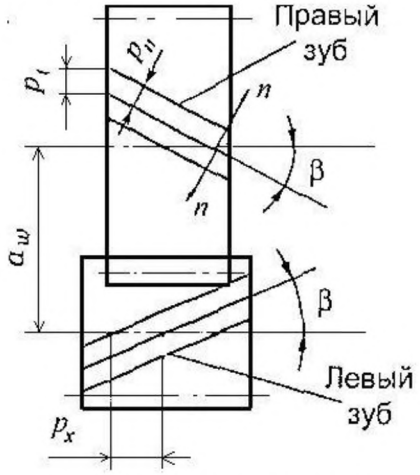

Рисунок 2.7-Геометрия косозубых колес

Основные геометрические размеры зависят от модуля и числа зубьев. При расчёте косозубых колёс учитывают два шага: нормальный шаг зубьев p_n - в нормальном сечении, окружной шаг p_t – в торцовом сечении; при этом $p_t = p_n/\cos\beta$

Соответственно шагам имеем два модуля зубьев:

$$m_t = p_t/\pi, \tag{2.11}$$

$$m_n = p_n/\pi \tag{2.12}$$

при этом

$$m_t = m_n/\cos\beta \tag{2.13}$$

где m_t и m_n – окружной и нормальный модули зубьев.

За расчётный принимают модуль m_n, значение которого должно соответствовать стандартному. Это объясняется следующим: для нарезания косых зубьев используется тот же инструмент, что и для прямозубых, но с соответствующим поворотом инструмента относительно заготовки на угол β. Поэтому профиль косого зуба в нормальном сечении совпадает с профилем прямого зуба; следовательно, m_n=m.

Диаметры делительный и начальный

$$d = d_\omega = m_t z = m_n z/\cos\beta \tag{2.14}$$

Диаметры вершин и впадин зубьев

$$d_a = d + 2m_n \tag{2.15}$$

$$d_f = d - 2{,}5m_n \tag{2.16}$$

Межосевое расстояние

$$a_\omega = (d_1 + d_2)/2 = d_1(u+1)/2 = m_t(z_1 + z_2)/2 = m_n z_1(u_1+1)/(2\cos\beta) \tag{2.17}$$

Коническая зубчатая передача

Конические зубчатые колёса применяют в передачах, оси валов которых пересекаются под некоторым межосевым углом Σ. Обычно $\Sigma = 90^\circ$ рис.2.8. Применяют во всех отраслях машиностроения, где по условиям компоновки машины необходимо передать движение между пересекающимися осями валов. Конические передачи сложнее цилиндрических, требуют периодической регулировки.

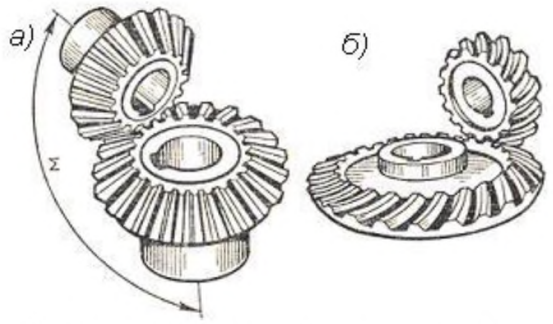

Рисунок 2.8-Коническая прямозубая передача а), передача с круговым зубом б)

Для нарезания зубчатых конических колес необходим специальный инструмент. В сравнении с цилиндрическими конические передачи имеют большую массу и габарит, сложнее в монтаже. Кроме того, одно из конических колёс, как правило шестерня, располагается консольно. При этом, вследствие повышенной деформации консольного вала, увеличиваются неравномерность распределения нагрузки по ширине зубчатого венца и шум. Конические колёса бывают с прямыми и круговыми зубьями.Передаточное числа при межосевом угле $\Sigma = 90^\circ$

$$u = \omega_1/\omega_2 = n_1/n_2 = z_2/z_1 = d_{e2}/d_{e1} = ctg\delta_1 = tg\delta_2\,. \tag{2.18}$$

Для конической прямозубой передачи рекомендуется u=2, 2,5; 3,15; 4, для передачи с круговыми зубьями возможны более высокие значения u; наибольшее значение u=6,3.

Геометрические параметры конического зубчатого колеса

Основные геометрические размеры определяют в зависимости от модуля и числа зубьев. Высота и толщина зубьев конических колёс постепенно уменьшается по мере приближения к вершине конуса. Соответственно изменяются шаг, модуль и делительные диаметры, которых может быть бесчисленное множество. Основные геометрические размеры имеют обозначения, принятые для прямозубых конических передач рис.2.9.

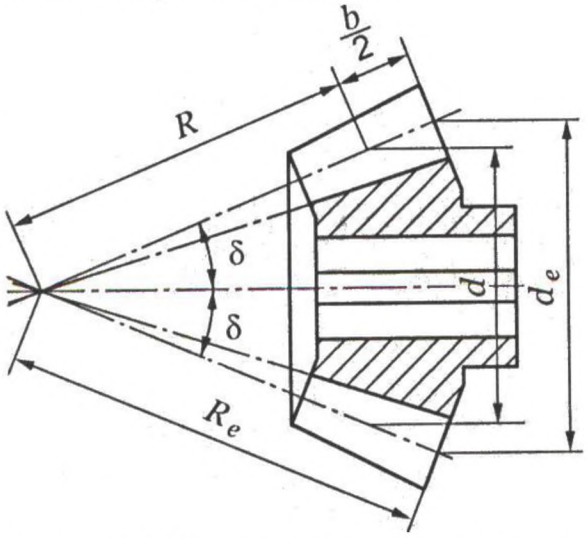

Рисунок 2.9-Геометрия конического колеса

Внешний диаметр d_e: $$d_e = m_e z, \quad (2.19)$$

где m_e - максимальный модуль зубьев – внешний окружной модуль, полученный по внешнему торцу колеса. Внешнее конусное расстояние

$$R_e = 0.5\sqrt{d_{e1}^2 + d_{e2}^2} = 0.5 m_e \sqrt{z_1^2 + z_2^2} = \frac{d_{e2}}{2u}\sqrt{\left(u^2+1\right)}, \quad (2.20)$$

Среднее конусное расстояние: $$R = R_e - 0.5b, \quad (2.21)$$

где b – ширина зубчатого венца колеса

$$b = K_{be} R_e \le 0.285 R_e, \quad (2.22)$$

где K_{be}- коэффициент ширины зубчатого венца относительно внешнего конусного расстояния. δ_1, δ_2 - углы делительных конусов;

Средний модуль

$$m = m_e R_e / R_e = m_e - (b \sin\delta_1)/z_1 \approx 0.857 m_e, \quad (2.23)$$

Средние делительные диаметры:

$$d_1 = m z_1 \approx 0.855 d_{e1}, \quad (2.24)$$

$$d_2 = m z_2 \approx 0.857 d_{e2}, \quad (2.25)$$

В соответствии с исходным контуром прямозубых конических колёс радиальный зазор $c = 0,2 m_e$, тогда внешняя высота головки зуба $h_{ae} = m_e$ и внешняя высота ножки зуба $h_{fe} = 1.2 m_e$

$$d_{ae1} = d_{e1} + 2 m_e \cos\delta_1, \quad (2.26)$$

$$d_{ae2} = d_{e2} + 2 m_e \cos\delta_2 \quad (2.27)$$

Угол ножки зуба $\theta_f = arctg(h_e / R_e)$

Угол головки зуба $\theta_{a1} = \theta_{f2}; \theta_{a2} = \theta_{f1}$

3. ПРАКТИЧЕСКАЯ ЧАСТЬ
СТРУКТУРА СПЕЦИАЛИЗИРОВАННОЙ САПР

Во многих случаях одних средств параметризации для автоматизации тех или иных действий в процессе проектирования недостаточно, и новые проектируемые 3D модели или чертежи хоть и схожи с эталоном, но имеют различия, не позволяющие использовать параметрические зависимости при построении. Например, когда какие либо значения принимаются конструктивно или выбираются из справочников в зависимости от третьих величин. Иногда расчетные параметры модели изменяются дискретно (например, модуль зубчатых колес всегда согласовывается со стандартными значениями и не может принимать значений, отличных от приведенных в ГОСТ) или не связаны аналитически с любым другим параметром.

Для определения таких параметров в библиотеку необходимо заложить достаточно сложный и гибкий алгоритм. Он может включать расчеты любой сложности, условия определения параметров, различные ограничения, связь с файлами данных и т. п. В таком случае никак не обойтись без программирования.

Для этой цели программный пакет КОМПАС3D располагает очень мощными инструментальными средствами разработки дополнительных модулей (прикладных библиотек) — КОМПАСМастер, которые позволяют использовать всю силу современного объектно-ориентированного программирования совместно с функциями КОМПАС для создания очень гибких и функциональных приложений. Хорошовладея одним из языков программирования и основами трехмерного моделирования в КОМПАС3D, можно научиться самостоятельно разрабатывать различныепо структуре программные модули для решения узкопрофильных задач конструирования. Такие приложения смогут производить сложные вычисления, самостоятельно выбирать необходимые параметры из баз данных, обмениваться даннымис внешними приложениями и, в конце концов, построить 3Dмодель или чертежнеограниченной сложности с учетом всех параметров (4 и 5 лабораторные работы).

Выбор того, что применять (параметризацию или программирование), зависит отпоставленных перед вами задач. Если вам необходим просто набор надежно хранимых и удобных в использовании параметрических элементов, а главное — еслис созданием самих этих элементов нет никаких проблем, конечно, лучше прибегнуть к параметризации и создавать простые библиотеки типовых элементов. Однако если вы планируете вводить в проектируемый модуль сложные аналитические расчеты, предполагаете, что модуль будет принимать решения вместопроектировщика, взаимодействовать с внешними приложениями, считывать или сохранять данные, то этот модуль должен быть полноценной программой. К средствам программирования следует прибегать лишь тогда, когдавам необходимо создавать собственную САПР.

Практика разработки подключаемых модулей на языках Delphi, C++и др. далеко не нова. Очень много известных приложений трехмерной графикиформируют свою архитектуру открытой, предоставляя пользователям возможность расширять функциональность программ. Такими приложениями являютсяAutoCAD, Adobe Photoshop, 3ds Max (некоторые плагины для 3ds Max значительно расширяют функционал программы) и др. В этом разделе будет рассказаноо создании пользовательских приложений на базе КОМПАС3D в одной из самыхпопулярных на сегодня сред программирования — Delphi.

КОМПАС предоставляет доступ как к функциям КОМПАС-График, таки к функциям трехмерного моделирования в КОМПАС-3D. Сам доступ может реализоваться двумя путями:

- с использованием экспортных функций, оформленных в виде DLLмодулей, которые разработчик подключает к своей программе при создании плоских чертежей, и с применением COMобъектов — при программном формировании твердотельных моделей;
- при помощи технологии Automation (автоматизации). Эта технология реализована через API (ApplicationProgrammingInterface, интерфейс прикладного программирования) системы КОМПАС. Управление и взаимодействие с системой при этом оформлено через интерфейсы IDispatch.

Технология COM, автоматизация и интерфейсы IDispatch

Поскольку программирование не является темой данной книги, мы не будемуглубляться в суть понятий технологии COM и автоматизации. Я опишу эти вопросы вкратце, чтобы вы имели некоторое представление.

С начала 1990 годов корпорация Microsoft разрабатывает технологию, позволяющую создавать гибкие модульные программы таким образом, чтобы отдельныемодули можно было писать на разных языках программирования, но чтобы приэтом обеспечивалась их полная взаимозаменяемость при использовании в различных программных пакетах. На сегодня эта технология полностью сформированаи называется COM (Component Object Model, модель компонентных объектов).

Технология COM описывает методологию реализации компонентов программного обеспечения: объектов, которые могут повторно использоваться, могут бытьнеоднократно подключены к разным приложениям. Повторное использованиекомпонентов стало логическим следствием эволюции объектно-ориентированного программирования и получило название компонентно-ориентированного подхода. Концепция компонентно-ориентированного программирования предусматривает полное отделение внутренних функций компонента от функций доступак нему извне. Теперь, обращаясь к компоненту, необязательно знать его внутреннее устройство, для этого достаточно лишь иметь информацию о том, как вызывать его функции. Другими словами, необходимо знать, как взаимодействоватьс компонентом и какой у него интерфейс. Такая функциональность в COM достигается за счет поддержки одного или нескольких интерфейсов, которые используются другими программами для доступа к внутренним членам и методам компонента.

Таким образом, интерфейс— это то, что размещено между двумя объектами и обеспечивает связь между ними. Интерфейсориентированное программирование представляет собой технологию разработки программного обеспечения, жестко нацеленную на использование интерфейсов. Интерфейс является своего рода связующимзвеном, предоставленным управляющей программе для доступа к COMобъекту.

Объект COM — конкретный экземпляр COM класса, завершенный объект с собственными членами данных и методами, который может легко встраиваться в программы или распространяться как отдельный программный продукт. COMобъектпредставляет собой или DLLбиблиотеку или EXEпрограмму для Windows, которыеможно создавать в любой среде программирования, способной поддерживать нужныйформат представления. COMобъект может иметь много функций, доступ к

которымпроисходит через его интерфейсы. Любой COM объект должен иметь по крайнеймере одни интерфейс IUnknown, хотя на самом деле имеет их значительно больше.

В свое время разработчики технологии COM пришли к заключению, что долженсуществовать некоторый механизм запроса интерфейса для определения его возможностей. Этот запрос также должен существовать для того, чтобы обеспечитьобновление клиентскими программами значения счетчика ссылок на данный интерфейс. Данный счетчик определяет, как долго интерфейс будет оставаться доступным, прежде чем он будет удален из памяти. Именно для этой цели существуетспециальный интерфейс, который называется IUnknown. Он имеет настольковажное значение, что основной принцип спецификации COM требует, чтобы всеCOMобъекты кроме своих специализированных интерфейсов поддерживали интерфейс IUnknown. Причем каждый определенный для объекта интерфейс долженпроисходить от IUnknownили от интерфейса, который в свою очередь сам происходит от IUnknown. Данный принцип называется наследованием интерфейсов. В интерфейсе IUnknownреализовано лишь три метода: Queryinterface(), AddRef()иRelease(). Метод Queryinterface()определяет, является ли полученный интерфейс нужным. Методы AddRef()и Release()используются для подсчета ссылок на данный интерфейс при его применении многими программами. Перед началомиспользования COMобъекта клиент вызывает метод COM, тем самым увеличиваяколичество ссылок на интерфейс на единицу. После окончания работы с интерфейсом клиент должен вызвать функцию Release(), чтобы уменьшить количество ссылок на единицу. Когда счетчик ссылок для всех интерфейсов станет равным нулю,значит, объект больше никем не используется и его можно выгружать из памяти.

На сегодняшний день технология COM используется практически во всех серьезныхпрограммах. Приведу один пример. Предположим, что пользователю в каком нибудьотчете нужно поместить электронную таблицу с расчетами, которые ссылаются на определенные параметры в тексте. Чтобы выполнить любые вычисления без использования технологии COM, пришлось бы постоянно переключаться между двумя программами (Word и Excel), а информацию копировать (вырезать и вставлять). Припомощи технологии COM можно применять функции электронной таблицы прямо в текстовом редакторе и автоматически форматировать полученный результат. Возможность реализовать операции такого рода называется автоматизацией.

Цель автоматизации состоит в том, чтобы дать возможность программе предоставлять в использование сервисы, которые в ней присутствуют. Основной особенностью автоматизации является возможность комбинировать функции различных специализированных приложений в одном модуле. COM дает возможность программам передавать свою информацию в другие приложения и модули. Если бы каждая программа или приложение сценарий могли бы поддерживать указатели и процедуру обхода указателей, то проблема была бы решена. Однако в некоторых языках программирования есть определенная трудность с процедурой обхода таблицы указателей. Некоторые из них, например Visual Basic, не поддерживают указатели напрямую. Для решения этой проблемы был разработан специальный интерфейс, который разрешает любым языкам программирования, в том числе таким, как Visual Basic, обращаться к методам COM компонентов. Этот интерфейс получил название IDispatch.

Любая программа, которая предоставляет свои возможности другим приложениям (поддерживает автоматизацию), может делать это через интерфейс IDispatch. Интерфейс IDispatch происходит от базового интерфейса модели COM IUnknown, тем не менее, в отличие от других COMинтерфейсов, IDispatchсодержит метод Invoke(). Его можно использовать для действительного выполнения методов, которые поддерживает COMобъект. Клиент можетвыполнить любой метод COMобъекта, вызвав метод Invoke()интерфейсаIDispatch. Этот механизм работает при помощи диспинтерфейса (интерфейсадиспетчеризации). Диспинтерфейс определяет методы, которые будут доступныблагодаря использованию метода Invoke()интерфейса IDispatch.

Интерфейсы IDispatchможно применять в любой из наиболее распространенных сегодня сред программирования (Visual C++ Studio, Borland Delphi, BorlandC++Builder, Visual Basic). Именно этим КОМПАС выгодно отличается отКОМПАС: вам не придется изучать малознакомый язык программирования, вы можете создавать свои приложения в той среде, к которой привыкли.

Базовые интерфейсы API системы КОМПАС

Взаимодействие внешнего приложения или подключаемого модуля с системой КОМПАС (с функциями моделирования, математическими функциями ядра системы и пр.) осуществляется посредством программных интерфейсов, называемых API. В КОМПАС на данный момент существуют API двух версий: API 5 и API 7. Обе версии реализуют различные функции системы и взаимнодополняют друг друга. Отсюда, полагаю, очевидно, что обе версии программныхинтерфейсов в равной мере поддерживаются и развиваются с учетом самих изменений в системе.В основном, для создания полноценных подключаемых модулей достаточно методов и свойств интерфейсов API 5.

Главным интерфейсом API системы КОМПАС является KompasObject. Получитьуказатель на этот интерфейс (если быть точным, на интерфейс приложения API 5) можно с помощью экспортной функции CreateKompasObject(). Методы этогоинтерфейса, главные из которых представлены в табл. 3.1, реализуют наиболее общие функции работы с документами системы, системными настройками, файлами,а также дают возможность получить указатели на другие интерфейсы (интерфейсыдинамического массива, работы с математическими функциями, библиотек моделей или фрагментов и различных структур параметров определенного типа).

Таблица 3.1 - Методы интерфейса KompasObject

Метод	Описание
ActiveDocument2D	Позволяет получить указатель на активный графический документ
ActiveDocument3D	Дает возможность получить указатель на активный трехмерный документ
Document2D	Позволяет получить указатель на интерфейс графического документа (чертежа или фрагмента)
Document3D	Дает возможность получить указатель на интерфейс трехмерного документа (детали или сборки)
GetDynamicArray	Возвращает указатель на интерфейс динамического массива
GetMathematic2D	Возвращает указатель на интерфейс для работы с математическими функциями в графическом документе
GetParamStruct	Один из самых важных методов. Позволяет получить интерфейс структуры параметров объекта определенного типа (например, параметры прямоугольника, эллипса, штриховки, размеров и т. д.)

Окончание таблицы 3.1

Метод	Описание
ksAttachKompasLibrary	Подключает библиотеку (добавляет ее в пункт главного меню Библиотеки)
ksChoiceFile	Выводит диалог для выбора файла для чтения
ksDetachKompasLibrary	Отключает библиотеку
ksDrawKompasDocument	Отрисовывает КОМПАС-документ, присланный из внешней программы или прикладного модуля, как слайд в указанном окне
ksEnableTaskAccess	Позволяет разрешить или запретить доступ пользователя к окну КОМПАС (применяется при работе библиотек)
ksError	Выдает сообщение об ошибке
ksGetApplication7	Возвращает указатель на интерфейс приложения API 7
ksGetDocOptions	Позволяет получить определенную структуру параметров настроек текущего документа
ksSetDocOptions	Дает возможность получить установить определенную структуру параметров настроек текущего документа
ksGetHWindow	Позволяет получить дескриптор главного окна КОМПАС
ksGetSysOptions	Дает возможность получить системные настройки по определенному типу
ksSetSysOptions	Позволяет установить системные настройки по определенному типу
ksGetSystemVersion	Дает возможность получить номер версии системы
ksMessage	Выдает сообщение произвольного содержания в окне КОМПАС
ksPrintKompasDocument	Выполняет печать КОМПАС-документа
ksPrintPreviewWindow	Запускает окно просмотра документа перед печатью
ksResultNULL	Обнуляет результат работы библиотеки, если ошибка не фатальная
ksSaveFile	Выдает диалог сохранения файла
ksSystemPath	Позволяет получить системный путь определенного типа (например, каталог системных файлов КОМПАС, каталог файлов библиотек и пр.)
Quit	Закрывает КОМПАС
SpcActiveDocument	Позволяет получить указатель на интерфейс активного в данный момент документа-спецификации
SpcDocument	Дает возможность получить указатель на интерфейс документа-спецификации

Описание прототипов всех приведенных функций, а также других методов, составляющих интерфейс KompasObject, вы найдете в справке по API КОМПАС.Файл этой справки размещается в каталоге SDK, находящейся в папке, в которойустановлен КОМПАС (например, по умолчанию это C:\Program Files\ASCON\KOMPAS-3D V12\SDK)

Другой важный интерфейс API 5 — интерфейс документа модели ksDocument3D.Получить его можно с помощью методов интерфейса KompasObject:

- ActiveDocument3D— для уже существующего и активного в данный момент документа;
- Document3D— если вы планируете создавать новый трехмерный документ.

Свойства (члены данных) этого интерфейса позволяют динамически управлятьнастройками любого трехмерного документа системы из вашего модуля. Наиболее используемые из них приведены в табл. 3.2.2.

Таблица 3.2 - Свойства интерфейса ksDocument3D

Свойство	Тип данных	Описание
author	WideString	Имя автора документа
drawMode	Integer	Тип отображения модели (каркас, без невидимых линий, невидимые линии тонкие, полутоновое)
fileName	WideString	Имя файла документа модели
hideAllAxis	WordBool	Скрыть/показать конструктивные оси
hideAllPlace	WordBool	Скрыть/показать начала координат
hideAllPlanes	WordBool	Скрыть/показать плоскости
hideAllSketches	WordBool	Скрыть/показать эскизы
hideAllSurfaces	WordBool	Скрыть/показать поверхности
hideAllThreads	WordBool	Скрыть/показать изображения резьбы
invisibleMode	WordBool	Режим редактирования документа (видимый или невидимый)
perspective	WordBool	Признак отображения перспективной проекции
reference	Integer	Указатель документа (детали или сборки)
shadedWireframe	WordBool	Полутоновое изображение с каркасом модели

Методы этого интерфейса позволяют программного управлять трехмерным документом, как сборкой и ее компонентами, так и отдельной деталью. Обратите внимание: именно управлять самим документом, но не выполнять в нем трехмерныеоперации! Методы, которые могут пригодиться вам при программировании подключаемых модулей, собраны в табл. 3.3.

Таблица 3.3 - Методы интерфейса ksDocument3D

Метод	Описание
Close	Позволяет закрыть документ
Create	Дает возможность создать пустой документ (деталь или сборку)
CreatePartFromFile	Позволяет создать деталь в сборке
CreatePartInAssembly	Возвращает указатель на интерфейс детали, создаваемой в сборке
DeleteObject	Позволяет удалить трехмерный объект (деталь, операцию, объект вспомогательной геометрии, сопряжения и пр.)
EntityCollection	Дает возможность получить указатель на массив элементов, выбранных в документе (например, операций или компонентов сборки для их копирования по массиву)
GetObjParam	Позволяет прочитать параметры объекта в структуру данных (по определенному типу параметров)
ksSetObjParam	Позволяет установить параметры объекта в структуру данных (по определенному типу параметров)
GetObjectType	Дает возможность получить тип трехмерного объекта
GetPart	Очень важный метод, возвращающий указатель на интерфейс компонента (детали или подсборки) в сборке
IsActive	Дает возможность проверить, активен ли документ
IsDetail	Возвращает TRUE, если трехмерный документ является деталью
Open	Позволяет запустить редактирование документа-модели
PartCollection	Возвращает указатель на интерфейс динамического массива компонентов, вставленных в сборку
RebuildDocument	Дает возможность перестроить документ
Save	Позволяет сохранить трехмерный документ в файл с именем ksDocument3D::fileName
SaveAs	Дает возможность сохранить трехмерный документ под указанным именем
SetActive	Позволяет активировать документ
SetPartFromFile	Дает возможность вставить в сборку компонент из файла или библиотеки моделей
UpdateDocumentParam	Позволяет обновить настройки документа

Важнейший из перечисленных в табл. 3.2.3 методов — ksDocument3D::GetPart.Входящим параметром этой функции является целочисленная переменная type_,которая определяет, интерфейс какого именно компонента сборки возвращать. Данная переменная имеет несколько предопределенных значений (констант):

- plnPlace_Part(равняется –4) — метод возвращает указатель на компонент, который находится в режиме контекстного редактирования (то есть редактирования «на месте»);
- pNew_Part(–3) — создает в модели новый компонент и возвращает указатель на него;
- pEdit_Part(–2) — возвращает указатель на редактируемый компонент (с помощью библиотеки);
- pTop_Part(–1) — верхний компонент, в состав которого входит или новый, или редактируемый, или указанный компонент;
- все остальные значения (от 0 и выше) отвечают номеру компонента в дереве построения, то есть возвращается указатель на существующий в сборке компонент.

Метод ksDocument3D::GetPartвозвращает указатель на интерфейс детали иликомпонента сборки — ksPart. Свойства и методы этого интерфейса (часть из которых приведена в табл. 3.4 и табл. 3.5) управляют состоянием компонентов сборки, они почти полностью дублируют команды контекстного меню и панели свойств,доступные пользователю при работе с тем или иным компонентом.

Таблица 3.4 - Свойства интерфейса ksPart

Свойство	Тип данных	Описание
excluded	WordBool	Определяет, исключен или нет компонент из расчета (из дерева построения)
fileName	WideString	Имя файла, из которого вставлен компонент
fixedComponent	WordBool	Определяет, является ли компонент зафиксированным
hidden	WordBool	Задает видимость компонента (скрыт или нет)
name	WideString	Имя компонента в дереве построений
standardComponent	WordBool	Определяет, является ли данный компонент стандартным (то есть библиотечным элементом)

Как видите, все эти методы лишь управляют состоянием компонента (задают видимость, размещение, определяют, запущено ли редактирование компонента), ноне дают возможность ничего создавать (выполнять операции).

Для программной реализации всех трехмерных операций, которые пользователивыполняют в трехмерных документах системы КОМПАС3D, в API реализованединый интерфейс ksEntity— интерфейс элемента модели. Этот интерфейсможно получить с помощью метода ksPart::NewEntity, которому необходимо передать тип создаваемого элемента. Типов элементов в системе, как и в APIсистемы, большое множество. Каждому из них отвечает своя целочисленная константа и свой собственный интерфейс параметров. Именно с помощью настроек(свойств и методов) этих интерфейсов и создаются любые возможные объектыв деталях и сборках КОМПАС3D. Некоторые константы с описанием типа элемента и интерфейса, к которому они относятся, приведены в табл. 3.6.

Сведения об остальных вы можете почерпнуть из файла справки SDK.hlp, поставляемого вместе с системой КОМПАС.

Таблица 3.5 - Методы интерфейса ksPart

Метод	Описание
BeginEdit	Позволяет запустить режим редактирования компонента
ColorParam	Дает возможность получить указатель на интерфейс параметров цвета и визуальных свойств компонента
EndEdit	Закрывает режим редактирования компонента «на месте»
EntityCollection	Формирует динамический массив трехмерных объектов и возвращает указатель на его интерфейс (например, операций для копирования по массиву)
GetDefaultEntity	Возвращает указатель на интерфейс объекта, создаваемого системой в трехмерном документе по умолчанию. Таких объектов всего четыре: начало координат и три ортогональных плоскости
GetPart	Позволяет получить указатель на интерфейс компонента
GetPlacement	Дает возможность получить указатель на интерфейс местоположения компонента в сборке
SetPlacement	Позволяет установить новое положение компонента в сборке
IsDetail	Позволяет проверить, является ли компонент деталью
NewEntity	Наиболее используемый метод: создает интерфейс нового трехмерного объекта и возвращает указатель на него
UpdatePlacement	Дает возможность изменить местоположение компонента, заданное в ksPart:: SetPlacement

Таблица 3.2.6 - Типы объектов трехмерного документа

Идентификатор	Числовое значение	Описание	Интерфейс параметров
o3d_planeXOY	1	Плоскость XOY	ksPlaneParam
o3d_planeXOZ	2	Плоскость 2XOZ	ksPlaneParam
o3d_planeYOZ	3	Плоскость YOZ	ksPlaneParam
o3d_sketch	5	Эскиз трехмерной операции	ksSketchDefinition
o3d_axis2Planes	9	Вспомогательная ось на пересечении двух плоскостей	ksAxis2PlanesDefinition
o3d_planeOffset	14	Смещенная плоскость	ksPlaneOffsetDefinition
o3d_planeAngle	15	Плоскость под углом	ksPlaneAngleDefinition
o3d_bossExtrusion	25	Операция выдавливания	ksBossExtrusionDefinition
o3d_cutExtrusion	26	Операция вырезания выдавливанием	ksCutExtrusionDefinition
o3d_bossRotated	28	Операция вращения	ksBossRotatedDefinition
o3d_cutRotated	29	Вырезание вращением	ksCutRotatedDefinition
o3d_bossLoft	31	Приклеить материал по сечениям	ksBossLoftDefinition
o3d_cutLoft	32	Вырезать по сечениям	ksCutLoftDefinition
o3d_meshCopy	35	Копирование по сетке	ksMeshCopyDefinition
o3d_circularCopy	36	Копирование по концентрической сетке	ksCircularCopyDefinition
o3d_circPartArray	38	Массив по концентрической сетке в сборке	ksCircularPartArrayDefinition
o3d_meshPartArray	39	Массив по сетке в сборке	ksMeshPartArrayDefinition
o3d_bossEvolution	46	Приклеить кинематически	ksBossEvolutionDefinition
o3d_cutEvolution	47	Вырезать кинематически	ksCutEvolutionDefinition
o3d_cutByPlane	50	Сечение плоскостью	ksCutByPlaneDefinition
o3d_cutBySketch	51	Сечение по эскизу	ksCutBySketchDefinition

Члены данных интерфейса ksEntitysoответствуют свойствам трехмерных элементов модели.

Среди методов наиболее важными являются три следующих:

– Create— создает трехмерную операцию или объект вспомогательной геометрии по заданным настройкам;

- ColorParam— возвращает указатель на интерфейс настроек цвета и оптических свойств элемента;
- GetDefinition— получает указатель на интерфейс параметров объекта определенного типа (параметры данного трехмерного элемента). Именно с помощью этого метода можно получить указатель на любой интерфейс, приведенный в столбце «Интерфейс параметров» табл. 3.2.6.

Таким образом, создание какойлибо трехмерной операции пользовательской программой сводится к такой последовательности шагов.

1. Инициализация главного интерфейса приложения API — KompasObject. Онинициализируется один раз для всего сеанса работы программы.
2. Инициализация интерфейса трехмерного документ ksDocument3D, с последующим созданием нового документа или получением указателя на активный документ.
3. Создание компонента и получение на него указателя (интерфейс ksPart). Для сборки это может быть готовый компонент, компонент, вставленный из файла или созданный «на месте». Для детали необходимо получить указатель на компонент типа pTop_Part.
4. Создание с помощью метода ksPart::NewEntity интерфейса нужной нам операции. При этом в метод передается соответствующий идентификатор (на пример, для выдавливания — o3d_bossExtrusion).
5. Получение с помощью метода ksEntity::GetDefinition указателя на интерфейс параметров конкретной операции (для выдавливания этим интерфейсом является ksBossExtrusionDefinition). Настройка этих параметров необходимым пользователю образом.
6. Создание операции с помощью метода ksEntity::Create.

Кроме перечисленных, в API системы КОМПАС существует еще большое множество различных интерфейсов, отвечающих за тот или иной аспект работы с программой. Небольшая их часть описана в табл. 3.7.

Таблица 3.7 - Некоторые дополнительные интерфейсы API КОМПАС

Интерфейс	Описание
ksPartCollection	Интерфейс массива компонентов сборки
ksMacro3DDefinition	Интерфейс трехмерного макрообъекта
ksMateConstraintCollection	Интерфейс набора сопряжений сборки
ksMateConstraint	Интерфейс структуры параметров сопряжения
ksMathematic2D	Интерфейс математических функций в графическом документе
ILibHPObject	Интерфейс для работы с характерными точками графического элемента
ksDynamicArray	Интерфейс динамического массива параметров
ksPhantom	Интерфейс фантомного отображения
ksEntityCollection	Интерфейс массива объектов модели

Программная реализация трехмерной операции

Рассмотрим выполнение трехмерной формообразующей операции вручную (то естьв самом КОМПАС) и с помощью воображаемого подключаемого модуля. В качестве примера выберем обычную операцию выдавливания на основе несложного эскиза, содержащего изображение окружности. Раздел *var* функции, реализующейвыполнение операции, должен выглядеть подобно листингу 3.1.

Листинг 3.1 Раздел объявлений функции построения операции выдавливания

```
var
    doc3 : ksDocument3D;// интерфейс трехмерного документа
    doc2 : ksDocument2D;
        // интерфейс графического документа
        // используется для создания изображения в эскизе
    iPart : ksPart;// интерфейс детали
    planeXOY : ksEntity;
        // интерфейс плоскости, на которой будет размещен эскиз
    iSketch : ksEntity;// интерфейс эскиза
    iSketchDef : ksSketchDefinition;// интерфейс параметров эскиза
    iBossExtrusion : ksEntity;// интерфейс операции выдавливания
    iBossDef : ksCutExtrusionDefinition;
        // интерфейс параметров операции выдавливания
    color : ksColorParam;
        // интерфейс параметров цвета операции выдавливания
```

Предположим, что документдеталь и эскиз с окружностью радиусом 40 мм на плоскости *XY* уже созданы (это значит, что объекты doc3, doc2, planeXOY, iSketch, iSketchDef уже созданы и проинициализированы).

Для выполнения операции выдавливания пользователь на панели инструментов ***Редактирование детали*** нажимает кнопку ***Операция выдавливания***. При этом на панели свойств, которая по умолчанию закреплена в нижней части окна программы,отображаются вкладки с настройками данной операции (некоторые значения установлены по умолчанию), в результате чего пользователь получает возможностьизменять эти настройки (рис. 3.1).

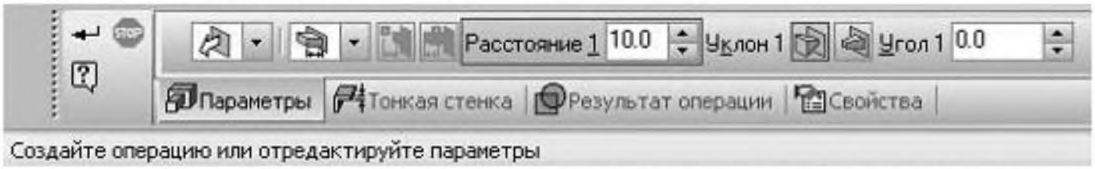

Рисунок 3.1-Настройки операции выдавливания на панели свойств

Программно это все реализуется следующим образом (листинг 3.2).

Листинг 3.2.Получение интерфейсов операции выдавливания и ее параметров

```
// iPart — указатель на объект класса ksPart,
// верхний элемент дерева построения детали
// создание интерфейса операции выдавливания
// с помощью метода ksPart::NewEntity,
// которому передаем идентификатор o3d_bossExtrusion
iBossExtrusion := ksEntity(iPart.NewEntity(o3d_bossExtrusion));
if (iBossExtrusion <> nil) then
begin
// если создание прошло успешно
// инициализируем интерфейс настроек операции выдавливания
iBossDef := ksBossExtrusionDefinition(iBossExtrusion.GetDefinition);
if (iBossDef <> nil) then
begin
... ...
end; end;
```

Рассмотрим параметры операции выдавливания и способы их настройки.

1. *Первый способ.* Направление выдавливания в КОМПАС выбирается из раскрывающегося списка Направление на панели свойств (рис. 3.2).

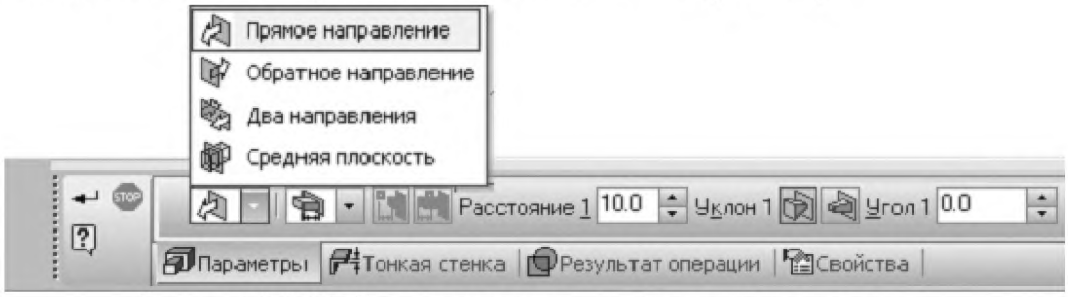

Рисунок 3.2-Выбор направления операции выдавливания

Программным аналогом направления является свойство directionType(тип — SmallInt) интерфейса ksBossExtrusionDefinition(или ksBaseExtrusion-Definition). Для него предопределены четыре значения:

- dtNormal(прямое направление) — направление добавления материала совпадает с направлением нормали к базовой плоскости (то есть плоскости, в которой размещается эскиз выдавливания);
- dtReverse(обратное направление) — направление выдавливания противоположно направлению нормали;
- dtBoth(в оба направления) — величина выдавливания задается отдельно для каждого направления;
- dtMiddlePlane(средняя плоскость) — выдавливание происходит в обе стороны от базовой плоскости на одинаковое расстояние, равное половине от указанной пользователем величины выдавливания.

Направление выдавливания устанавливается очень просто (листинг 3.3).

Листинг 3.3 Задание направления выдавливания

```
iBossExtrusion := ksEntity(iPart.NewEntity(o3d_bossExtrusion));
if (iBossExtrusion <> nil) then
  begin
    iBossDef := ksBossExtrusionDefinition(iBossExtrusion.GetDefinition);
if (iBossDef <> nil) then
begin
// задаем направление выдавливания
iBossDef.directionType := dtNormal;
end;
end;
```

2. *Второй способ выдавливания* (рис. 3.3).

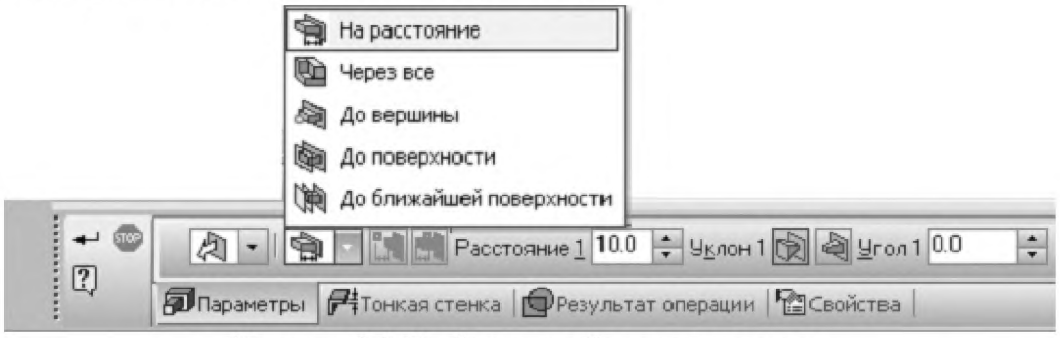

Рисунок 3.3-Выбор способа выдавливания

Для этого типа настроек в API системы также предусмотрены константы:

- *etBlind(на расстояние)* — проводит выдавливание на величину, заданную в поле ***Расстояние 1***(или ***Расстояние 2***, если задано направление dtReverseилиdtBoth);
- *etThroughAll(через все)* — при наличии другой геометрии (других формообразующих операций) выдавливание происходит через все тело детали; значение величины выдавливания игнорируется;
- *etUpToVertexTo, etUpToVertexFrom(до вершины)* — система переходит в режим ожидания указания пользователем вершины, до которой будет осуществлено выдавливание. Введенное значение величины выдавливания игнорируется. В программе эту вершину следует задать явно еще на этапе разработки проекта;
- *etUpToSurfaceTo, etUpToSurfaceFrom(до поверхности)* — аналогично режиму построения до вершины, в данном режиме система ожидает выбора пользователем трехмерной поверхности. Значение величины выдавливания также игнорируется. Как и для вершины, плоскость в программе следует указывать явно;
- *etUpToNearSurface(к ближайшей поверхности)* — выдавливание проводится к ближайшей поверхности, которую система автоматически определяетв направлении выдавливания.

3. ***Расстояние выдавливания*** — вводится пользователем в соответствующем текстовом поле (полях).

4. ***Угол наклона операции выдавливания.*** На панели свойств задается направлениенаклона (внутрь или наружу), а также сам угол наклона.

Параметры операции, описанные в пп. 2–4, устанавливаются с помощью единого метода интерфейса ksBossExtrusionDefinition::SetSideParam(листинг 3.4).

Листинг 3.4.Задание способа, величины выдавливания, а также угла наклона

```
iBossExtrusion := ksEntity(iPart.NewEntity(o3d_bossExtrusion));
if (iBossExtrusion <> nil) then
    begin
        iBossDef := ksBossExtrusionDefinition(iBossExtrusion.GetDefinition);
      if (iBossDef <> nil) then
        begin
            iBossDef.directionType := dtNormal;
/* Если первый параметр имеет значение true, это значит, что все следующие
параметры  задаются для направления выдавливания dtNormal. Если установить
значение параметра равным false, значит все следующие параметры определяются для
обратного направления. Второй параметр задает способ выдавливания (etBlind);
третий параметр — величина выдавливания (25мм); четвертый параметр false — уклон
вглубь (true — наружу) последний параметр — величина уклона в градусах */
                iBossDef.SetSideParam(true, etBlind, 25, false, 10);
        end;
    end;
```

5. На вкладке Тонкая стенкапользователь может управлять параметрами толщиныи способа формирования тонкой стенки или установить режим выдавливаниясплошного тела (рис. 3.4).

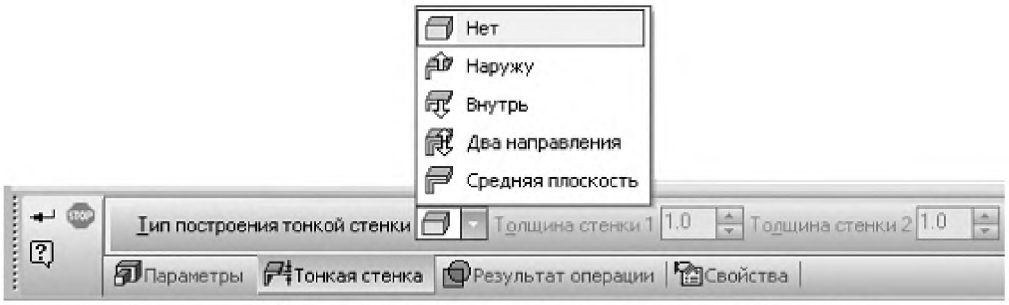

Рисунок 3.4-Выбор режима построения тонкой стенки

В программе это задается следующим образом (листинг 3.5). В примере выполняется операция выдавливания без тонкой стенки.

Листинг 3.5. Задание параметров тонкой стенки

```
iBossExtrusion := ksEntity(iPart.NewEntity(o3d_bossExtrusion));
if (iBossExtrusion <> nil) then
        begin
            iBossDef := ksBossExtrusionDefinition(iBossExtrusion.GetDefinition);
          if (iBossDef <> nil) then
            begin
                        iBossDef.directionType := dtNormal;
                        iBossDef.SetSideParam(true, etBlind, 25, false, 10);
/* первый параметр false указывает на то, надо ли строить тонкую стенку второй параметр задает направление выдавливания третий и четвертый параметры определяют толщину стенки  в прямом и обратном направлениях соответственно */
                        iBossDef.SetThinParam(false, dtNormal, 0, 0);
                        // устанавливаем эскиз операции
                        iBossDef.SetSketch(iSketch);
              end;
        end;
```

В этом фрагменте кода также был задан эскиз операции с помощью методаksBossExtrusionDefinition::SetSketch. Мы приняли, что сам эскиз, как и упоминалось ранее, был создан до начала выполнения операции выдавливания.

На вкладке ***Свойства***можно также настроить цвет и оптические свойства создаваемого элемента. При программном построении элемента для этого сначала необходимо инициализировать еще один дополнительный интерфейс — ksColorParam.C помощью его свойств можно управлять визуальными характеристиками трехмерного элемента. Внесенные в функцию построения изменения показаны в листинге 3.6.

Листинг 3.6.Установка визуальных свойств

```
iBossExtrusion := ksEntity(iPart.NewEntity(o3d_bossExtrusion));
if (iBossExtrusion<>nil) then
        begin
// присваиваем имя трехмерной операции, которое будет отображено в дереве построения
                iBossExtrusion.name := "Программная операция выдавливания";
// получаем указатель на интерфейс параметров цвета  и оптических свойств
                color := ksColorParam(iBossExtrusion.ColorParam);
                color.color := RGB(255, 0, 0); //с помощью макроса RGB задаем цвет —красный
                color.specularity := 0.8; // задаем уровень зеркальности (в долях единицы)
                color.shininess := 1; // и блеск
```

```
        iBossDef := ksBossExtrusionDefinition(iBossExtrusion.GetDefinition);
        if (iBossDef <> nil) then
                begin
                        iBossDef.directionType := dtNormal;
                        iBossDef.SetSideParam(true, etBlind, 25, false, 10);
                        iBossDef.SetThinParam(false, dtNormal, 0, 0);
                        iBossDef.SetSketch(iSketch);
                end;
    end;
```

Теперь все свойства настроены нужным образом, и можно создавать саму трехмерную операцию выдавливания. В КОМПАС для этого необходимо нажать кнопку***Создать объект***, в программе — вызвать метод Createобъекта интерфейса ksEntity.Окончательный вид функции приведен в листинге 3.2.7.

Листинг 3.7. Операция выдавливания

```
iBossExtrusion := ksEntity(iPart.NewEntity(o3d_bossExtrusion));
if (iBossExtrusion <> nil) then
    begin
        iBossExtrusion.name := "Программная операция выдавливания";
        color := ksColorParam(iBossExtrusion.ColorParam);
        color.color := RGB(255, 0, 0);
        color.specularity := 0.8;
        color.shininess := 1;
        iBossDef :=ksBossExtrusionDefinition(iBossExtrusion.GetDefinition);
        if (iBossDef <> nil) then
                begin
                        iBossDef.directionType := dtNormal;
                        iBossDef.SetSideParam(true, etBlind, 25, false, 10);
                        iBossDef.SetThinParam(false, dtNormal, 0, 0);
                        iBossDef.SetSketch(iSketch);
                        iBossExtrusion.Create; // создаемоперацию
                end;
    end;
```

Результат проделанной работы, как «вручную», так и с помощью подключаемогомодуля, показан на рис. 3.5. Изображенный трехмерный элемент, как вы толькочто сами убедились, вполне может быть создан без какоголибо вмешательствапользователя, с помощью API КОМПАС.

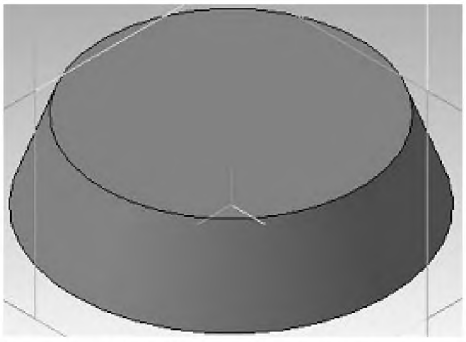

Рисунок 3.5-Трехмерный элемент, созданный программно

Очевидно, что точно таким же образом вы можете автоматизировать выполнениелюбой последовательности любых трехмерных формообразующих операций, построение вспомогательной геометрии, формирование массивов и пр., что в результате даст вам готовую трехмерную модель изделия.

Необходимо выполнить заготовку конструкторской библиотеки, как это было сделано в лабораторной работе №4.

Конструкторская библиотека для КОМПАС3D представляет собой обычную DLL(Dynamic Link Library — динамически подключаемая библиотека Windows), только с расширением RTW. Такая библиотека подключается к КОМПАС, как и любая другая.

Чтобы RTW библиотека, написанная на Delphi, могла взаимодействовать с КОМПАС, в ней обязательно должны присутствовать как минимум три функции:

- LibraryEntry - точка входа в библиотеку;
- LibraryName - возвращает имя библиотеки, отображаемое в менеджере библиотек;
- LibraryId - возвращает идентификатор библиотеки (позже его можно использовать при работе с самой библиотекой, для подключения списка меню ее команд и пр.).

Все эти функции обязательно должны быть экспортными, то есть экспортируемыми из данной DLL, чтобы система КОМПАС могла их видеть и вызывать. По этой причине их обязательно нужно вынести в раздел exportsприкладной библиотеки.

СкопируйтеизпапкиSDK\Includevкаталогсвашимпроектом \dcuфайлыksAuto.pas, ksConstTLB, ks2DCOM_TLB.pas, KsTLB.pas, LDefin2D, LDefin3D, LibTool, Ltdefine. Включитеданныефайлыввашпроект.

Поскольку взаимодействие с пользователем планируется осуществлять с помощью диалогового окна, в проект необходимо добавить диалоговую форму. Для этого выполните команду File->New->Form, после чего в инспекторе объектов настройте часть свойств формы.

После создания формы Delphi автоматически сгенерировал модуль (unit) с кодом ее описания. Удалите из этого модуля описание глобальной переменной GearsForm типа TGearsForm и сохраните модуль под именем BuildUnit.pas.

Добавьте на форму по четыре текстовых метки (TLabel) и поля ввода (TEdit), а также две кнопки (TButton). Присвойте им заголовки Построение и Отмена. Разместите указанные элементы управления приблизительно так, как показано на рис. 3.6.

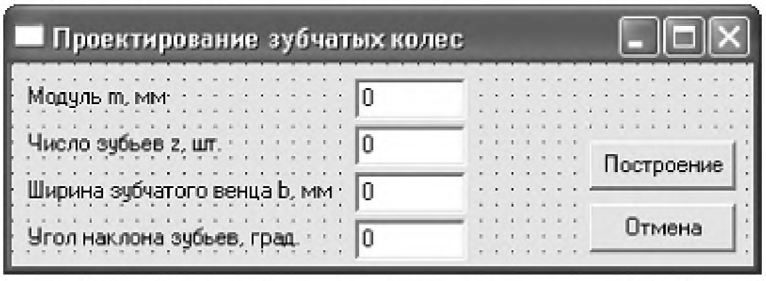

Рисунок 3.6 - Форма будущего диалогового окна библиотеки

После создания формы необходимо обеспечить ее вывод в окне КОМПАС. Задача, на первый взгляд, сложная, но решается просто. [10]
Для вывода диалогового окна библиотеки нужно сделать следующее.

1. Получить дескриптор главного окна КОМПАС.
2. Запретить доступ пользователю к главному окну программы.
3. Создать объект диалогового окна и вывести его на экран в модальном режиме.
4. После закрытия пользователем окна библиотеки уничтожить окно и вернуть управление главным окном КОМПАС пользователю.
5. Обнулить дескриптор приложения.

Реализовать эту последовательность действий лучше всего в процедуре точки входа в библиотеку (листинг 3.8).

Листинг 3.8 Вывод диалогового окна библиотеки

```
procedure LibraryEntry(command: WORD); pascal;
var GearsForm : TGearsForm;
begin
    kompas := KompasObject(CreateKompasObject);
    if (kompas = nil) then exit;
    Application.Handle:=kompas.ksGetHWindow; //получаем дескриптор главного окна КОМПАС
    kompas.ksEnableTaskAccess(0); // запрещаем доступ к главному окну
    GearsForm := TGearsForm.Create(Application); // создаем объект диалогового окна
    GearsForm.ks := kompas;
    GearsForm.ShowModal; // выводим диалог на экран
    GearsForm.Free; // удаляем объект
    kompas.ksEnableTaskAccess(1); // возвращаемдоступкокну
    Application.Handle := 0;
    kompas := nil;
end;
```

В приведенном фрагменте кода есть одна, на первый взгляд, непонятная строка — GearsForm.ks:= kompas;. В действительности, здесь все просто. Поскольку обработка построения зубчатого колеса будет выполнена внутри класса формы TGearsForm, то для того, чтобы в нем не получать заново указатель на интерфейс KompasObject, этот указатель передается внешней переменной ks класса TGearsForm. Разумеется, эту переменную (типа KompasObject) необходимо предварительно добавить в раздел publicописания класса формы. Перед этим подключите в разделе usesмодуля BuildUnitследующие файлы (модули): ksTLB, ksAuto, ksConstTLB, LDefin3D, LDefin2Dи math(последний не относится к КОМПАС API, это стандартный модуль математических функций Delphi).

Скомпилируйте проект и подключите полученную библиотеку Gears3D.rtw к КОМПАС. Запустите ее и убедитесь, что после выполнения ее единственной команды в центре главного окна появляется созданное нами диалоговое окно.

Перейдем к реализации обработчиков щелчка на кнопках. Начнем со второй (***Отмена***), поскольку ее обработчик чрезвычайно прост. Щелкните дважды в редакторе форм на кнопке ***Отмена*** в обработчике, автоматически созданном в редакторекода, введите всего одну строку, закрывающую форму (листинг 3.9).

Листинг 3.9 Обработчик щелчка на кнопке Отмена

```
procedure TGearsForm.Button2Click(Sender: TObject);
begin
    if CloseQuery then Close;
end;
```

Процедура обработки нажатия кнопки Построение намного сложнее. Условно ееможно разделить на три части.

1. Расчет геометрических параметров зубчатого колеса по введенным исходным данным.
2. Создание пустого документа КОМПАС- Деталь.
3. Собственно построение модели зубчатого колеса.

Программное построение модели колеса реализуем такой последовательностьютрехмерных операций.

1. Сначала программно в плоскости *XOY* создается эскиз, содержащий контур половины сечения колеса. На основании этого эскиза выполняется операция вращения, формирующая заготовку зубчатого колеса.

2. Далее в плоскости *YOZ*строится второй эскиз с четырьмя окружностями, над которыми выполняется операция вырезания в два направления. Таким образом, мы получим отверстия в диске.

3. Следующим шагом является выполнение выреза между зубьями в венце колеса. Для построения выреза воспользуемся способом заключающимся в построении выреза с помощью операции ***Вырезать по сечениям.*** При этом в модели колеса строится ряд эскизов-сечений, плоскости которых удалены от боковой поверхности колеса на величину $l = i \cdot b/(n_c - 1)$ (где b — ширина колеса, n_c — количество сечений или эскизов, i - порядковый номер эскиза). Для нашей библиотеки достаточно будет трех эскизов: по два на торцевых плоскостях колеса и один посредине — на плоскости *YOZ*. Это значит, что библиотека должна будет построить две вспомогательные плоскости, удаленные в обе стороны от плоскости *YOZ* на половину ширины венца зубчатого колеса. В каждой из трех плоскостей (двух вспомогательных и ортогональной *YOZ*) будет создано изображение эскиза выреза между зубьями, повернутое относительно вертикальной оси на угол $\alpha = 2 \cdot l \cdot tg\beta/d_k$, где β — угол наклона линии зуба, d_k —делительный диаметр зубчатого колеса. Для первой плоскости вместо l необходимо подставить 0, для второй (*YOZ*) — $\beta/2$, для третьей — b .

4. В завершении создается ось на пересечении плоскостей *XOZ* и *XOY*. Относительно этой оси формируется массив по концентрической сетке вырезов между зубьями колеса. Количество копий устанавливается равным количеству зубьев колеса.

Первый этап реализации построения: расчета геометрических характеристик создаваемого колеса (листинг 3.10).

Листинг 3.10. Расчет параметров колеса

```
Procedure TGearsForm.Button1Click(Sender: TObject);
var /* раздел объявления переменных все объекты приведенных интерфейсов используются при построении */
    doc3 : ksDocument3D;
    iPart : ksPart;
    PlaneXOY : ksEntity;
    PlaneXOZ : ksEntity;
    PlaneYOZ : ksEntity;
    SketchEntity : ksEntity;
    iSketchDef : ksSketchDefinition;
    doc : ksDocument2D;
    r : reference;
    iBaseRotatedEntity : ksEntity;
    Color : ksColorParam;
    iBaseRotatedDef : ksBaseRotatedDefinition;
    iSketch1Entity : ksEntity;
    iSketch1Def : ksSketchDefinition;
```

```
        iCutExtrusion : ksEntity;
        iCutExtrusionDef : ksCutExtrusionDefinition;
        iOffsetPlaneEntity : ksEntity;
        iOffsetPlaneDef : ksPlaneOffsetDefinition;
        iSketch2Entity : ksEntity;
        iSketch2Def : ksSketchDefinition;
        iSketch3Entity : ksEntity;
        iSketch3Def : ksSketchDefinition;
        iOffsetPlane1Entity : ksEntity;
        iOffsetPlane1Def : ksPlaneOffsetDefinition;
        iSketch4Entity : ksEntity;
        iSketch4Def : ksSketchDefinition;
        iCutLoftEntity : ksEntity;
        iCutLoftDef : ksCutLoftDefinition;
        Collect : ksEntityCollection;
        iAxis : ksEntity;
        iAxis2PlDef : ksAxis2PlanesDefinition;
        iCircularCopy : ksEntity;
        iCirCopyDef : ksCircularCopyDefinition;
        Collect1 : ksEntityCollection;
        // геометрические параметры колеса
        module : double;
        Lm, Dm : double;
        Dv : double;
        b_k, c : double;
        d_k, d_fk, d_ak : double;
        delta0 : double;
        z : integer;
        beta : double;
        Dotv : double;
        alfa1, alfa2 : double;
begin
        Hide; // прячем диалоговое окно
        // считываем параметры, введенные пользователем в окне
        module := StrToFloat(Edit1.Text);
        z := StrToInt(Edit2.Text);
        Lm := StrToFloat(Edit3.Text);
        beta := StrToFloat(Edit4.Text);
        Dv := round(Lm/1.4); // диаметр отверстия под вал
        b_k := Lm; // ширину маточины и ширину колеса принимаем равными
        Dm := 1.8*Dv; // диаметр маточины
        c := round(0.35*b_k); // толщина диска, соединяющего маточину с ободом
        delta0 := round(2.5*module/cos(DegToRad(beta))); // толщина обода
        d_k := module*z; // делительный диаметр колеса
        d_ak := d_k+2*module; // диаметр выступов
        d_fk := d_k-2.5*module; // диаметр впадин
```

```
        Dotv := (d_fk - 2*delta0 + Dm)/2; //диаметр размещения центров отверстий в диске
                // создание детали...
                // построение модели...
        Close; // закрываем форму
end;
```

Если сейчас собрать приложение и попробовать запустить библиотеку, ничего происходить не будет, потому что пока ничего не создается и не строится.

Следующий этап построения намного более интересен — он заключается в программном создании документа КОМПАС-Деталь (листинг 3.11). В данном листинге раздел описания переменных и расчет параметров колеса пропущен, а приведен только фрагмент кода, реализующий создание документа-детали. В процедуру построения (обработчик нажатия кнопки **Построение**) этот фрагмент должен бытьвставлен сразу после расчетов.

Листинг3.11 Создание документа детали

```
        // получаем указатель на интерфейс трехмерного документа
doc3 := ksDocument3D(ks.Document3D());
        // создаем документ
        // параметр false — в видимом режиме
        // параметр true— документ-деталь
if doc3.Create(false, true) then
        begin
                // заполняем параметры документа
                doc3.author := “                ”;
                doc3.comment := “Зубчатоеколесо”;
                doc3.drawMode := 3;
                doc3.perspective := true;
                doc3.UpdateDocumentParam();
        end else exit;
if (doc3 = nil) then// проверяем, как прошла инициализация
        begin
                ks.ksMessage(“Не удалось создать документ!”);
                exit;
        end;
```

Откомпилировав и запустив приложение, вы сможете наблюдать, как после закрытия диалогового окна (нажатия кнопки **Построение**) программа сама создаст пустой документ КОМПАС-Деталь.

В листинге 3.12 приведен с небольшими сокращениями код построения трехмерной модели.

Листинг 3.12 Построение модели колеса

```
iPart := ksPart(doc3.GetPart(pNew_Part)); // получаем указатель на интерфейс детали
if (iPart <> nil) then
begin
        // интерфейсыортогональныхплоскостей
        PlaneXOY := ksEntity(iPart.GetDefaultEntity(o3d_planeXOY));
        PlaneXOZ := ksEntity(iPart.GetDefaultEntity(o3d_planeXOZ));
        PlaneYOZ := ksEntity(iPart.GetDefaultEntity(o3d_planeYOZ));
```

```
   // интерфейс эскиза (половина контура сечения колеса)
   iSketchEntity := ksEntity(iPart.NewEntity(o3d_sketch));
   if (iSketchEntity <> nil) then
      begin
         // интерфейс параметров эскиза
         iSketchDef := ksSketchDefinition(iSketchEntity.GetDefinition);
               if (iSketchDef <> nil) then
                  begin
                     if (PlaneXOY <> nil) then
                        begin
                           // устанавливаем плоскость,  на которой создается эскиз
                           iSketchDef.SetPlane(PlaneXOY);
                           iSketchEntity.Create;
                           // запускаем процесс редактирования эскиза
                           // doc— указатель на интерфейс ksDocument2D
                           doc := ksDocument2D(iSketchDef.BeginEdit);
                           if (doc <> nil) then
                            begin
                                  /* вычерчиваем изображение эскиза  с помощью методов
                            интерфейса ksDocument2D */
                                  // код пропущен
                            end;
                             // завершение редактирования эскиза
                            iSketchDef.EndEdit;
                     end;
            end;
end;
// интерфейс базовой операции вращения
iBaseRotatedEntity := ksEntity(iPart.NewEntity(o3d_baseRotated));
// интерфейс параметров цвета и визуальных свойств
Color := ksColorParam(iBaseRotatedEntity.ColorParam);
Color.specularity := 0.8;
Color.shininess := 1;
if (iBaseRotatedEntity <> nil) then
   begin // интерфейс параметров вращения
            iBaseRotatedDef :=
ksBaseRotatedDefinition(iBaseRotatedEntity.GetDefinition);
            if (iBaseRotatedDef <> nil) then
               begin // настройка параметров вращения
                     iBaseRotatedDef.SetThinParam(false, dtNormal, 1, 1);
                     iBaseRotatedDef.SetSideParam(true, 360);
                     iBaseRotatedDef.toroidShapeType := false;
                     iBaseRotatedDef.SetSketch(iSketchEntity);
                // создаем операцию вращения результат – заготовка зубчатого колеса
                   iBaseRotatedEntity.Create;
            end;
end;
```

```
   // интерфейс эскиза (отверстия в диске)
iSketch1Entity := ksEntity(iPart.NewEntity( o3d_sketch ));
if (iSketch1Entity <> nil) then
   begin
         iSketch1Def := ksSketchDefinition(iSketch1Entity.GetDefinition);
         if (iSketch1Def <> nil) then
            begin
               if (PlaneYOZ <> nil) then
               begin
               // размещаем эскиз на плоскости YOZ
                  iSketch1Def.SetPlane(PlaneYOZ);
                  iSketch1Entity.Create;
                  doc := ksDocument2D(iSketch1Def.BeginEdit);
                  if (doc <> nil) then
                     begin
                  // изображение в эскизе — 4 окружности
                  // создаются вызовом метода ksDocument2D::ksCircle
                        doc.ksCircle(0, Dotv/2, 0.4*(d_fk/2-delta0-Dm/2), 1);
                        doc.ksCircle(0, -Dotv/2, 0.4*(d_fk/2-delta0-Dm/2), 1);
                        doc.ksCircle(Dotv/2, 0, 0.4*(d_fk/2-delta0-Dm/2), 1);
                        doc.ksCircle(-Dotv/2, 0, 0.4*(d_fk/2-delta0-Dm/2), 1);
                      end;
                   iSketch1Def.EndEdit;
                     end;
         end;
end;
   // интерфейс операции Вырезать выдавливанием
iCutExtrusion := ksEntity(iPart.NewEntity(o3d_cutExtrusion));
if (iCutExtrusion <> nil) then
   begin // интерфейс параметров вырезания
         iCutExtrusionDef :=
               ksCutExtrusionDefinition(iCutExtrusion.GetDefinition);
         if (iCutExtrusionDef <> nil) then
               begin
                 iCutExtrusionDef.SetSketch(iSketch1Entity); // настройка параметров
               iCutExtrusionDef.directionType := dtBoth; // направление
                 // величина вырезания по каждому из направлений
                 iCutExtrusionDef.SetSideParam(true, etBlind, c/2,0, false);
                 iCutExtrusionDef.SetSideParam(false, etBlind, c/2, 0, false);
                 iCutExtrusionDef.SetThinParam(false, 0, 0, 0);
                 iCutExtrusion.Create; // создаемотверстиявдиске
               end;
   end;
   // интерфейс смещенной плоскости
iOffsetPlaneEntity := ksEntity(iPart.NewEntity(o3d_planeOffset));
if (iOffsetPlaneEntity <> nil) then
```

```
begin
        // интерфейс параметров смещенной плоскости
        iOffsetPlaneDef :=
        ksPlaneOffsetDefinition(iOffsetPlaneEntity.GetDefinition);
        if (iOffsetPlaneDef <> nil) then
        begin
                // величина, базовая плоскость и другие  параметры смещения
                iOffsetPlaneDef.Offset := b_k/2;
                iOffsetPlaneDef.SetPlane(PlaneYOZ);
                iOffsetPlaneDef.direction := false;
                // делаемплоскостьскрытой
                iOffsetPlaneEntity.Hidden := true;
                // создаем вспомогательную плоскость
                iOffsetPlaneEntity.Create;
           end;
   end;
// эскиз первого выреза между зубьями
iSketch2Entity := ksEntity(iPart.NewEntity(o3d_sketch));
if (iSketch2Entity <> nil) then
   begin
        iSketch2Def := ksSketchDefinition(iSketch2Entity.GetDefinition);
        if (iSketch2Def <> nil) then
        begin
                // базовая плоскость— вспомогательная iOffsetPlaneEntity
                iSketch2Def.SetPlane(iOffsetPlaneEntity);
                iSketch2Entity.Create;
                doc := ksDocument2D(iSketch2Def.BeginEdit);
                        alfa1 := 360/z;
                        doc.ksMtr(0, 0, 90, 1, 1);
                /* вычерчивание изображения эскиза вместо эвольвент для
        простоты берем обычные дуги по трем точкам */
                // код пропущен
                        doc.ksDeleteMtr;
                iSketch2Def.EndEdit;
        end;
   end;
   // интерфейс второго эскиза выреза между зубьями
   iSketch3Entity := ksEntity(iPart.NewEntity(o3d_sketch));
   if (iSketch3Entity <> nil) then
      begin
        iSketch3Def := ksSketchDefinition(iSketch3Entity.GetDefinition);
        if (iSketch3Def <> nil) then
        begin
                // строим на плоскости YOZ
                iSketch3Def.SetPlane(PlaneYOZ);
                iSketch3Entity.Create;
```

```
                doc := ksDocument2D(iSketch3Def.BeginEdit);
                alfa2 := -RadToDeg(b_k*tan(DegToRad(beta))/d_k);
                        doc.ksMtr(0, 0, 90, 1, 1);
                /*вычерчивание изображения эскиза вместо эвольвент для
        простоты берем обычные дуги по трем точкам*/
                // кодпропущен
                        doc.ksDeleteMtr;
                iSketch3Def.EndEdit;
            end;
        end;
// вторая смещенная плоскость
iOffsetPlane1Entity := ksEntity(iPart.NewEntity(o3d_planeOffset));
if (iOffsetPlane1Entity <> nil) then
    begin
        iOffsetPlane1Def :=
        ksPlaneOffsetDefinition(iOffsetPlane1Entity.GetDefinition);
        if (iOffsetPlane1Def <> nil) then
        begin
                // величина смещения та же
                iOffsetPlane1Def.Offset := b_k/2;
                // направление противоположное
                iOffsetPlane1Def.direction := true;
                iOffsetPlane1Def.SetPlane(PlaneYOZ);
                // делаем плоскость скрытой
                iOffsetPlane1Entity.Hidden := true;
                // создаем смещенную плоскость
                iOffsetPlane1Entity.Create;
        end;
    end;
// третий (последний) эскиз выреза между зубьями
iSketch4Entity := ksEntity(iPart.NewEntity(o3d_sketch));
if (iSketch4Entity <> nil) then
    begin
        iSketch4Def := ksSketchDefinition(iSketch4Entity.GetDefinition);
        if (iSketch4Def<>nil) then
        begin
                // базовая плоскость — только что созданная смещенная
                iSketch4Def.SetPlane(iOffsetPlane1Entity);
                iSketch4Entity.Create;
                doc := ksDocument2D(iSketch4Def.BeginEdit);
                alfa2 := -RadToDeg(2*b_k*tan(DegToRad(beta))/d_k);
                        doc.ksMtr(0, 0, 90, 1, 1);
                        /* вычерчивание изображения эскиза вместо эвольвент для
                простоты берем обычные дуги по трем точкам */
                // код пропущен
                        doc.ksDeleteMtr;
```

```
                iSketch4Def.EndEdit;
            end;
    end;
// интерфейс операции Вырезать по сечениям
iCutLoftEntity := ksEntity(iPart.NewEntity(o3d_cutLoft));
if (iCutLoftEntity <> nil) then
    begin
            // интерфейс параметров операции по сечениям
            iCutLoftDef := ksCutLoftDefinition(iCutLoftEntity.GetDefinition);
            if (iCutLoftDef <> nil) then
                begin
                    // интерфейс массива ksEntityCollection
                    // коллекции эскизов для вырезания по сечениям
                    Collect := ksEntityCollection(iCutLoftDef.Sketchs);
                    // добавляем эскизы в колекцию
                    Collect.Add(iSketch2Entity);
                    Collect.Add(iSketch3Entity);
                    Collect.Add(iSketch4Entity);
                    // создаем операцию по сечениям
                    // результат — первый вырез между зубьями в венце колеса
                    iCutLoftEntity.Create;
            end;
    end;
// интерфейс вспомогательной оси на пересечении двух плоскостей
iAxis := ksEntity(iPart.NewEntity(o3d_axis2Planes));
if (iAxis <> nil) then
    begin
            // интерфейс параметров вспомогательной оси
            // напересеченииплоскостей
           iAxis2PlDef := ksAxis2PlanesDefinition(iAxis.GetDefinition);
           if (iAxis2PlDef <> nil) then
                begin
                    // задаемплоскости
                    iAxis2PlDef.SetPlane(1, PlaneXOZ);
                    iAxis2PlDef.SetPlane(2, PlaneXOY);
                    // делаем ось невидимой
                    iAxis.hidden := true;
                    iAxis.Create; // создаем вспомогательную ось
                end;
    end;
// интерфейс операции Массив по концентрической сетке
iCircularCopy := ksEntity(iPart.NewEntity(o3d_circularCopy));
if (iCircularCopy <> nil) then
    begin
            // интерфейспараметровоперациикопированияпомассиву
            iCirCopyDef :=
```

```
                ksCircularCopyDefinition(iCircularCopy.GetDefinition);
                if (iCirCopyDef <> nil) then
                begin
                    // коллекция операций для копирования
Collect1 :=ksEntityCollection(iCirCopyDef.GetOperationArray);
                    // операция всего лишь одна – вырезание зуба
                Collect1.Add(iCutLoftEntity);
                  // количество копий, равно количеству зубьев
                    iCirCopyDef.count2 := z;
                    iCirCopyDef.factor2 := true;
                    iCirCopyDef.SetAxis(iAxis); // ось копирования
    // создаем концентрический массив– колесо готово!
    iCircularCopy.Create;
                end;
      end;
    end;
```

Пример работы. Вновь соберите (перекомпилируйте) библиотеку. Перейдите в окно КОМПАСи запустите приложение из менеджера библиотек. Введите исходные данные длязубчатого колеса (например, модуль — 3,5 мм, количество зубьев — 56, ширинаzубчатого венца — 60 мм и угол наклона линии зубьев — 15°) и нажмите кнопку **Построение**. Всего за несколько секунд программа построит по указанным данным 3D-моделькосозубого зубчатого колеса (рис. 3.7).

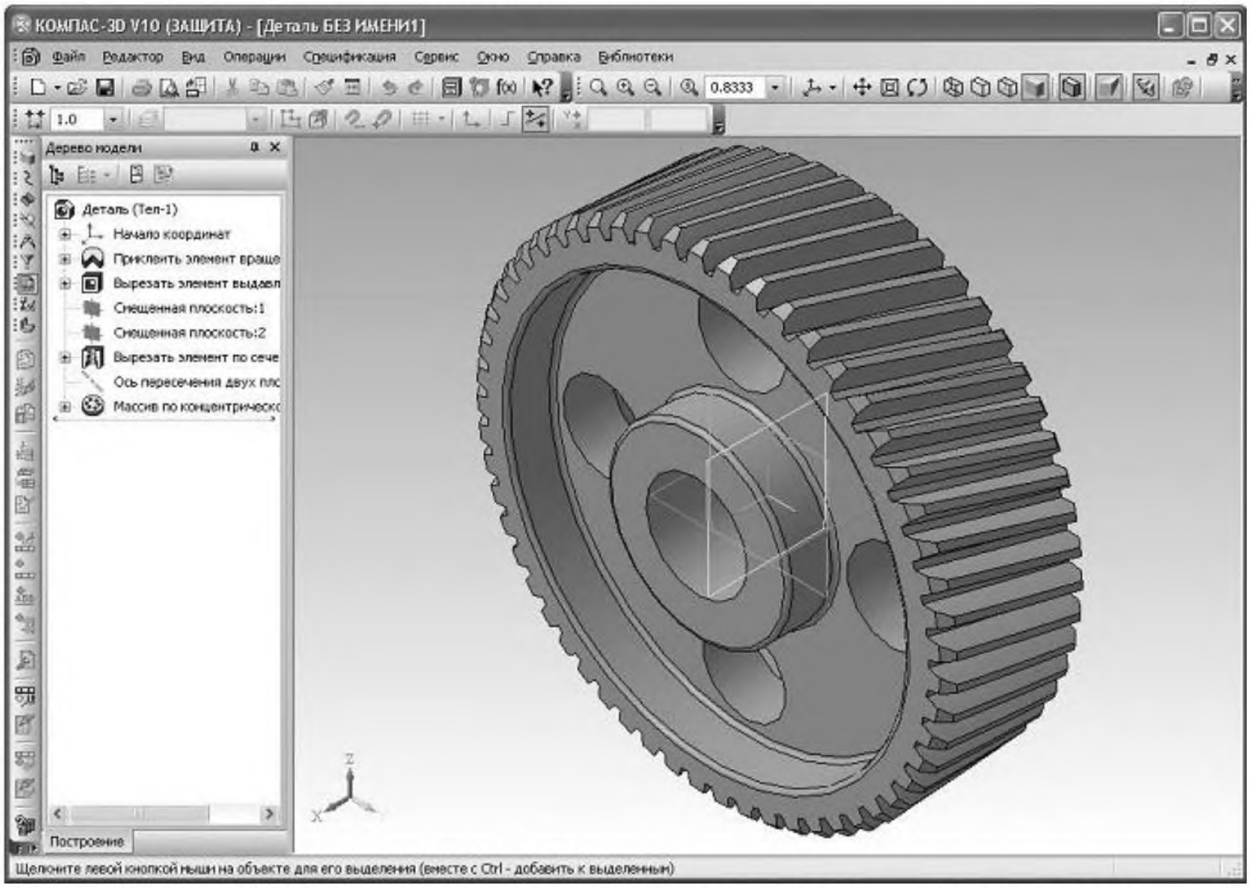

Рисунок 3.7 Трехмерная модель зубчатого колеса, созданная программно

С помощью такой небольшой утилиты вы можете создавать зубчатые колеса практически любых размеров, с произвольным углом наклона зубьев, а также прямозубые.

ТРЕБОВАНИЯ К ОТЧЕТУ ПО КУРСОВОЙ РАБОТЕ

Отчет должен включать следующие разделы:

- титульный лист;
- содержание;
- введение;
- основная часть
  - Теоретические сведения о предмете исследования;
  - Создание САПР на базе Компас 3D–ЗУБЧАТАЯ ПЕРЕДАЧА;
    - Назначение САПР Компас 3D–ЗУБЧАТАЯ ПЕРЕДАЧА;
    - Структура САПР;
    - Математическая модель;
    - Описание всех использованных методови их переменных;
    - Привести исходный код с пояснением (интересные части);
    - Описание интерфейса;
    - Результат работы;
      - 3D модель;
      - Дерево сборки;
- заключение;
- список литературы;
- приложение (исходный код).

Отчет оформляется в соответствии следующим параметрам:

- Поля: левое 2, правое 1, верхнее и нижнее 2.
- Шрифт 14п, интервал 1,5.
- Абзац 1,5.
- Заголовки должны отделяться от текста отступом с верху и с низу, равным 1,5.
- Заголовки разных уровней разделяются отступом 0,8
- Оформление рисунков:
  - Рисунки располагаются по центру, подпись снизу
  - Ссылка на рисунок должна быть в тексте отчета
  - Нумерация рисунков двухуровневая, в пределах главы
- Оформление таблиц
  - Таблицы должны быть растянуты по ширине листа.
  - Подпись и наименование таблицы располагается сверху слева
  - Ссылка на таблицу должна быть в тексте отчета
  - Нумерация таблиц двухуровневая, в пределах главы
- Оформление приложений:
  - Приложение должно нумероваться буквами алфавита (Приложение А, Б) кроме букв Ё, З, Й, О, Ы, Ъ, Ь.
  - На приложение должны быть ссылки в тексте отчета
- Оформление списка литературы
  - Список должен быть нумерованным
  - Оформляется согласно ГОСТ 7.0.5-2008 «Библиографическая ссылка. Общие требования и правила оформления»
  - На каждый элемент списка должна быть ссылка в тексте отчета
- Страницы должны быть пронумерованы за исключением титульного листа.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Чайковский филиал
федерального государственного бюджетного
образовательного учреждения высшего образования
«Пермский национальный исследовательский политехнический университет»
(ЧФ ПНИПУ)

Кафедра автоматизации, информационных и инженерных технологий

КУРСОВАЯ РАБОТА
по дисциплине: «Системы автоматизированного проектирования»
Тема: «»

Выполнил
студент группы АСУ-14-1 Иванов И.И.

Принял
к.ф.-м.н., доцент кафедры АИИТ Селиванов К.М.

201__г.

ЛИТЕРАТУРА

1. Погорелов, В.И. Auto CAD 2009: 3D- моделирование/ В.И. Погорелов. – СПб: БХВ-Петербург, 2009. – 400с.
2. Норенков, И.П. Основы автоматизированного проектирования: учебник для вузов / И.П. Норенков. – 4 -е изд., перераб. и доп. – М.: Изд-во МГТУ, 2009. – 430с
3. ГОСТ16530-83. Передачи зубчатые. – М.: Идз-во стандартов, 1991. – 70 с.
4. ГОСТ16531-83. Передачи зубчатые цилиндрические. – М.: Идз-во стандартов, 1991. – 28 с.
5. ГОСТ16532-83. Передачи зубчатые цилиндрические эвольвентные внешнего зацепления. – М.: Идз-во стандартов, 1991. – 38 с.
6. ГОСТ2.403-83 Правила выполнения рабочих чертежей цилиндрических зубчатых колес. – М.: Изд-во стандартов, 1991. – 10 с.
7. ГОСТ2.409-83 Правила выполнения чертежей зубчатых(шлицевых) соединений. – М.: Издательство стандартов, 1991. – 10с.
8. Чекмарев А.А. Начертательная геометрия и черчение: Учебник для вузов по технической специальности. М.: ВЛАДОС, 1999. –471 с.
9. Левицкий В.С. Машиностроительное черчение и автоматизированное выполнение чертежей. – М.: Высшая школа, 2003.– 428 с.
10. Кидрук М. И.КОМПАС-3D V10 на 100 % (+CD). — СПб.: Питер, 2009. — 560 с.