3084

Министерство образования
Республики Беларусь

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра «Тракторы»

В.В. Равино
Ю.Е. Атаманов

ИНФОРМАТИКА

Минск 2007

Министерство образования Республики Беларусь
БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра «Тракторы»

В.В. Равино, Ю.Е. Атаманов

ИНФОРМАТИКА

Методическое пособие
по выполнению курсовой работы
для студентов специальностей
1-37 01 03 «Тракторостроение»,
1-37 01 04 «Многоцелевые гусеничные и колесные машины»,
1-37 01 05 «Городской электрический транспорт»

Минск 2007

УДК 004 (076.5)
ББК 32.97
Р 13

Рецензенты:
канд. техн. наук, доцент каф.
«Автомобили», БНТУ *В.А. Кусяк*,
канд. техн. наук, доцент кафедры
«Техническая эксплуатация автомобилей» БНТУ
А.С. Гурский

Равино, В.В.

Р13 Информатика: методическое пособие по выполнению курсовой работы для студентов специальностей 1-37 01 03 «Тракторостроение», 1-37 01 04 «Многоцелевые гусеничные и колесные машины» и 1-37 01 05 «Городской электрический транспорт» / В.В. Равино, Ю.Е. Атаманов. – Мн.: БНТУ, 2007. - 83 с.

ISBN 978-985-479-614-7.

В методическом пособии даны общие рекомендации по выполнению и оформлению курсовой работы по дисциплине «Информатика». Рассмотрены стандартные приемы алгоритмизации практических задач, а также методика написания программы на языке Borland Delphi для конкретного примера.

Представлен перечень заданий по выполнению курсовой работы для трех специальностей. В приложениях приведены основные справочные данные, необходимые для реализации математической модели и таблица классов исключительных ситуаций.

Методическое пособие предназначено для студентов специальностей 1-37 01 03 – «Тракторостроение», 1-37 01 04 – «Многоцелевые гусеничные и колесные машины» и 1-37 01 05 – «Городской электрический транспорт», а также может быть полезно студентом других специальностей при изучении дисциплины «Информатика»

УДК 004 (076.5)
ББК 32.97

ISBN 978-985-479-614-7

© Равино В.В.,
Атаманов Ю.Е., 2007
© БНТУ, 2007

Предисловие

Настоящее методическое пособие по выполнению курсовой работы разработано в соответствии с программой дисциплины «Информатика для» для студентов специальностей I-37 01 03 «Тракторостроение», I-37 01 04 «Многоцелевые гусеничные и колесные машины» и I-37 01 05 «Городской электрический транспорт». Данное методическое пособие не претендует на роль руководства по программированию в среде *Borland Delphi* или по работе с редактором *MS Word*. Эту задачу выполняет специальная литература. Перечень рекомендуемой литературы приведен в конце пособия.

Задачей данного пособия является обеспечение студентов методическими материалами, необходимыми при выполнении индивидуального задания по курсовой работе по дисциплине «Информатика».

В методическом пособии рассмотрены общие приемы постановки, программирования технических задач в среде *Delphi* и требования к оформлению пояснительной записки к курсовой работе. Основное внимание в пособии уделено тщательно подобранным и весьма полезным практическим приемам и методам исследования конкретных технических задач, встречающихся в инженерной практике.

Среда *Delphi* решила проблему быстрого создания эффективных *Windows* приложений. Благодаря использованию объектно-ориентированного языка *Delphi* программы получаются стройными, понятными и разрабатываются значительно быстрее, чем на других языках программирования. Выполненный код получается достаточно эффективным, компиляция проектов происходит очень быстро. Курсовая работа по дисциплине «Информатика» способствует закреплению у студентов навыков программирования, полученных ими во время изучения этой дисциплины.

При разработке методического пособия авторы использовали свой многолетний опыт преподавания дисциплины «Информатика» в Белорусском национальном техническом университете и опыт руководства курсовыми работами студентов по указанной дисциплине. Настоящее методическое пособие отрабатывалось на студентах более трех лет. После выполнения и защиты курсовой работы по дисциплине «Информатика» многие студенты высказывали свои пожелания и предложения, направленные на совершенствование содержания курсовой работы и ее тематики. Часть этих пожеланий и предложений учтены авторами настоящем методическом пособии. Авторы с большой признательностью отмечают участие студентов группы 101159 П.М. Галямова и 101153 С.С. Семченкова при подборе и верстке методического пособия.

Мы также выражаем свою благодарность рецензентам к.т.н., доценту кафедры «Автомобили» БНТУ В.А. Кусяку и к.т.н., доценту кафедры «Техническая эксплуатация автомобилей» БНТУ А.С. Гурскому за конкретную критику методического пособия.

В.В. Равино, Ю.Е. Атаманов

Введение

Главными целями выполнения курсовой работы по дисциплине «Информатика» являются закрепление, углубление и обобщение студентами знаний, полученных во время обучения, и выработка у них умения самостоятельно применять эти знания для творческого решения конкретных практических инженерных задач.

Данное методическое пособие должно способствовать высококачественному и своевременному выполнению студентами наиболее трудной части курсовой работы: разработке алгоритма и программы для решения инженерной задачи, уменьшить непроизводительные потери времени на поиск необходимых материалов, выбор методов решения поставленной задачи.

При выполнении работы студент подтверждает владение такими основными навыками и приемами, полученными при изучении дисциплины «Информатика», как:

– решение и алгоритмизация вычислительных задач, встречающихся в инженерной практике;

– программирование на языке высокого уровня с учетом структурного подхода;

– создание простейших приложений в среде визуального программирования;

– оформление текстовой документации сопровождения информационных разработок и программных продуктов.

Курсовая работа выполняется с использованием программных средств офисного назначения и визуального программирования.

Задача курсовой работы – создание законченного программного приложения прикладного технического характера, выполненного по индивидуальному заданию.

Для выполнения курсовой работы необходимо дополнительно использовать литературу, список которой приведен в конце методического пособия.

При выполнении курсовой работы, студент пользуется консультациями руководителя курсовой работы. В процессе выполнения курсовой работы студент должен следить за правильностью принятых допущений, расчетных схем, результатов расчета, руководствоваться выполненными аналогичными типовыми механико-математическими моделями.

1. СТРУКТУРА ПОЯСНИТЕЛЬНОЙ ЗАПИСКИ

Пояснительная записка к курсовой работе должна иметь следующую структуру.

Титульный лист
Бланк задания
Реферат
Содержание
Введение
1. Математическое решение задачи
2. Алгоритмизация вычислительных процессов
3. Таблица идентификаторов
4. Разработка интерфейса пользователя
5. Структура программного приложения
6. Разработка справочной системы
7. Расчет контрольного примера
Заключение
Список использованных источников информации
Доклад по курсовой работе в виде презентации PowerPoint
Приложения (при необходимости)

Структура подразделов (если необходимо) определяется студентом самостоятельно. Отсутствие каких-либо разделов, перечисленных выше, не допускается.

Пояснительная записка к курсовой работе выполняется средствами *MS Word*, представлена в формате **rtf** и иметь кодировку текста Windows-1251. Основной текст должен быть набран шрифтом Times New Roman Cyr черного цвета с высотой 14 пт, через полтора интервала. Формулы набираются с использованием *MS Equation* или *MathType*. Рукописный текст не допускается.

Поля в документе: верхнее и нижнее – 20 мм, левое – 30 мм, правое 15 мм.

Оформление пояснительной записки должно соответствовать ГОСТ 7.32-2001.

Все исходные данные должны вводиться из файла исходных данных, а результаты выводиться в таблицу результатов в тех единицах, которые указаны в задании. При закрытии программы последний вариант введенных данных должен запоминаться. Расчеты выполняются в системе СИ.

К защите курсовой работы студент представляет:

- комплект печатных документов на листах формата А4:

1) пояснительная записка курсовая работа;

2) опись файлов курсовой работы;

3) комплект материалов презентации;

- материалы на электронном носителе информации (на дискетах 3,5" или компакт-диске):

1) в каталоге DOC - пояснительная записка курсовой работы;

2) в каталоге PRG - исходные файлы курсовой работы;

3) в каталоге EXE - файл программы, а также дополнительные файлы, необходимые для работы программы;

4) файлы презентации;

5) опись файлов курсовой работы.

Студент несет полную ответственность за полноту и правильность представляемых файлов и содержащуюся в них информацию.

Наименования в тексте (подписи к рисункам, графикам, таблицам) должны иметь уникальное обозначение, формирование которого рекомендуется осуществлять в автоматическом режиме.

Слайды презентации должны содержать материалы по всем разделам курсовой работы, а также выводы.

Опись файлов курсовой работы оформляется в соответствии с рисунком 1.1. Файл описи представляется в виде отдельного файла index в формате rtf, имеющего кодировку текста Windows-1251. Файл описи размещается на первом носителе курсовой работы.

ОПИСЬ ФАЙЛОВ КУРСОВОЙ РАБОТЫ

Имя файла	Объем, Kb	Содержание

Рисунок 1.1 – Пример описи файлов курсовой работы

Защита курсовой работы осуществляется в аудитории, оснащенной техническими средствами и программным обеспечением, необходимыми для проведения доклада.

Содержание доклада подготавливается в виде файла презентации, который воспроизводится при помощи технических средств и программного обеспечения по ходу доклада. В процессе защиты демонстрируется функционирование разработанного программного обеспечения, а также иллюстративный материал, позволяющий раскрыть замысел курсовой работы.

Презентация по теме курсовой работы не должна превышать 5-8 минут.

При необходимости хранения файлов, превышающих емкость носителя, они подлежат архивации. Рекомендуется использовать самораспаковывающиеся файловые архивы.

Для обеспечения сохранности информации и защиты ее от внесения изменений, исправлений, несанкционированного копирования файлы следует оснастить защитой. Файлы курсовой работы могут быть открыты только для чтения.

2. ОФОРМЛЕНИЕ ОСНОВНЫХ РАЗДЕЛОВ

2.1. Титульный лист

Титульный лист является первой страницей пояснительной записки курсовой работы. Он подписывается студентом-исполнителем и преподавателем – руководителем курсовой работы. Номер страницы на титульном листе не проставляется. Пример оформления титульного листа приведен в приложении 1.

2.2. Бланк заданий

Бланк задания является второй страницей пояснительной записки. Выдается руководителем курсовой работы. При получении задания, студент расписывается на бланке о получении задания и ставит дату получения задания.

Результаты текущего выполнения работы визируются преподавателем и являются основой для текущего контроля и аттестации студентов. При выполнении менее 30 % работы к календарному сроку плановой защиты индивидуальное задание подлежит замене.

2.3. Содержание

Содержание пояснительной записки к курсовой работе должно отражать ее структуру и включать в себя не более трех уровней вложенности (разделы, подразделы). Содержание должно быть сгенерировано автоматическими средствами текстового редактора *MS Word*.

Для этого предварительно название каждого пункта содержания должны быть отформатированы в виде заголовков редактора *MS Word*. Чтобы это сделать нужно выделить текст пункта содержания и нажать пиктограмму  или в главном меню выбрать *Формат>Стили и форматирование*, указать стиль заголовка и нажать кнопку *Применить*. Пунктам содержания пояснительной записки одного уровня следует назначать одинаковое форматирование.

После форматирования пунктов содержания пояснительной записки необходимо поместить курсор на странице документа *MS Word*, на которой будет генерироваться содержание пояснительной записки. Выбрать в головном меню редактора *MS Word* команду *Вставка>Ссылка>Оглавление и указатели* и открыть вкладку *Оглавление* (рисунок 2.1).

В более ранних версиях редактора *MS Word* вид этого окна несколько иной, но это не влияет на приемы работы с ним.

В данном окне следует установить флаг в позициях *Показать номера страниц* и *Номера страниц по правому краю*. В качестве заполнителя желательно выбирать точки.

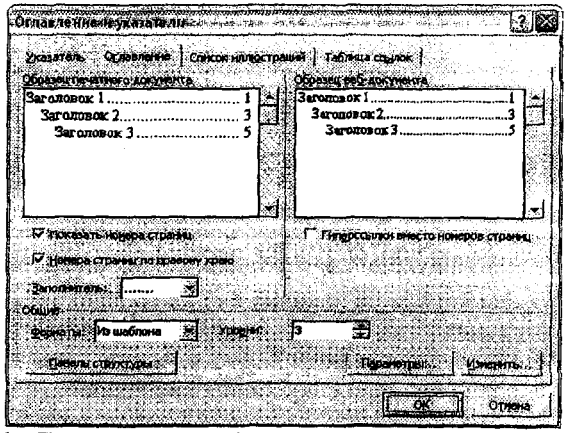


Рисунок 2.1 - Диалоговое окно *Оглавление и указатели* для *MS Word 2003*

После выполнения указанных выше действий - нажать кнопку *ОК*. В позиции курсора автоматически сгенерируется содержание пояснительной записки к курсовой работе. Для перехода из содержания к заголовку в тексте необходимо щелкнуть по его названию при нажатой клавише *Ctrl*.

Для обновления оглавления нужно установить курсор в поле содержания пояснительной записки и нажать клавишу *F9* (или воспользоваться контекстным меню).

2.4. Введение

Введение должно отражать цель и задачи выполнения конкретного задания курсовой работы. Во введении приводятся возможные области применения разработки. Указываются основные приемы, инструменты, программные средства решения задачи и основные результаты, полученные в процессе выполнения курсовой работы.

2.5. Основные разделы курсовой работы

2.5.1. Математическое решение задачи

Вводятся обозначения параметров, участвующих в расчетных формулах. Приводится расчетная схема задачи с принятыми допущениями и с обозначениями кинематических и силовых параметров. Анализируются значения параметров, их пределы изменения или неизменяемость во время выполнения расчетов.

Производится выбор и обоснование математического метода решения поставленной задачи. Математическое решение задачи приводится в формульном варианте.

Пример. Определить радиус поворота R гусеничного трактора в зависимости от поперечной базы B трактора и приращения скорости гусеницы Δv.

Радиус поворота гусеничного трактора рассчитывается по формуле:

$$R = 0{,}5B\frac{v}{\Delta v},$$

где R – радиус поворота трактора, м;

B – поперечная база трактора, м;

v – линейная скорость движения центра масс трактора, м/с;

Δv – приращение скорости гусеницы трактора, м/с.

Расчетная схема представлена на рисунке 2.2.

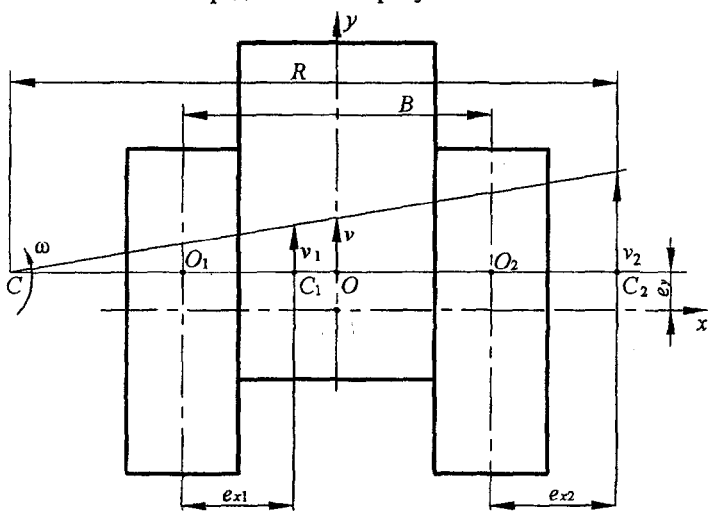


Рисунок 2.2 – Расчетная схема поворота гусеничного трактора

На рисунке обозначено:

C_1, C_2 – полюсы вращения гусениц;

e_{x1}, e_{x2}, e_y – смещения полюсов вращения гусениц относительно их геометрических центров O_1 и O_2;

v_1 и v_2 – скорости отстающей и забегающей гусениц.

На расчетной схеме поворота гусеничного трактора не показано Δv, т.к. данная величина рассчитывается по формуле:

$$\Delta v = \frac{v_2 - v_1}{2}.$$

Сводные данные по расчетным параметрам представлены в таблице 2.1.

Таблица 2.1 - Сводные данные по расчетным параметрам

Обозначение в формуле	Диапазон изменения	Примечания
B	1,6 м	Постоянная величина
v	1...30 км/ч	Варьируемая величина
Δv	0,1...1 м/с	Варьируемая величина

При разработке программы для расчета радиуса поворота гусеничного трактора необходимо использовать операторы циклов (*for... to... do, Repeat... Until* или *While... do*).

2.5.2. Алгоритмизация вычислительных процессов

В пояснительной записке приводится блок-схема алгоритма решения математической модели в виде графического объекта. Алгоритм должен отвечать требованиям и правилам алгоритмизации, а команды в блоках алгоритма - отвечать языку системы команд решателя. Рисунок алгоритма должен отвечать правилам создания графических объектов в текстовом редакторе.

Данный способ по сравнению с другими способами записи алгоритма имеет ряд преимуществ. Он наиболее нагляден: каждая операция вычислительного процесса изображается отдельной геометрической фигурой. Кроме того, графическое изображение алгоритма наглядно показывает разветвления путей решения задачи в зависимости от различных условий, повторение отдельных этапов вычислительного процесса и другие детали.

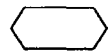
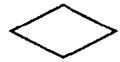
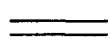
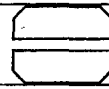
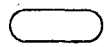
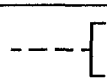
В настоящее время действует единая система программной документации (ЕСПД), которая устанавливает правила разработки, оформления программ и программной документации. В ЕСПД определены и правила оформления блок-схем алгоритмов (ГОСТ 19.701-90, ИСО 5807-85).

Операции обработки данных и носители информации изображаются на схеме соответствующими блоками. Большая часть блоков по построению условно вписана в прямоугольник со сторонами a и b. Минимальное значение a равно 10 мм, увеличение a производится на число, кратное 5 мм. Размер b=15 мм. Для отдельных блоков допускается соотношение между a и b, равное 1:2. В пределах одной схемы рекомендуется изображать блоки одинаковых размеров. Все блоки нумеруются. Виды и назначение основных блоков приведены в таблице 2.2.

Таблица 2.2- Условные обозначения блоков схем алгоритмов программ

Наименование	Обозначение	Функции
1	2	3
1. Данные		Символ отображает данные. Ввод или вывод данных. Носитель данных не определен.

Продолжение табл. 2.2

1	2	3
2. Процесс		Выполнение операции или группы операции, в результате которых изменяется значение, форма представления или расположение данных.
3. Предопределенный процесс		Использование ранее созданных и отдельно написанных программ (подпрограмм).
4. Подготовка		Символ отображает модификацию команды или группы команд с целью воздействия на некоторую последующую функцию (установка переключателя, инициализация программы и т. п.)
5. Решение		Выбор направления выполнения алгоритма в зависимости от некоторых переменных условии.
6. Параллельные действия		Символ отображает синхронизацию двух или более параллельных операций
7. Границы цикла		Начало и конец цикла
8. Символ линии		Символ отображает поток данных. При необходимости добавляют стрелки.
9. Пунктирная линия		Символ отображает альтернативную связь между двумя или более объектами. Также используют для обведения аннотированного участка.
10. Соединитель		Указание связи между прерванными линиями, соединяющими блоки. Соответствующие символы должны содержать одни и те же уникальные обозначения
11. Терминатор		Начало-конец программы (подпрограммы)
12. Комментарий		Используется для добавления описательных надписей в целях объяснений или примечаний.
13. Пропуск		Используется для отображения пропуска символа или группы символов.

Линии, соединяющие блоки и указывающие последовательность связей между ними должны проводиться параллельно линиям рамки. Стрелка в конце линии не ставится, если линия направлена слева направо или сверху вниз. В блок может входить несколько линий, то есть блок может яв-

ляться преемником любого числа блоков. Из блока (кроме логического) может выходить только одна линия. Логический блок может иметь в качестве продолжения одни из двух блоков, и из него выходят две линии. Если на схеме имеет место слияние линий, то место пересечения выделяется точкой. В случае, когда одна линия подходит к другой и слияние их явно выражено, точку можно не ставить.

Блок-схема должна отображать все разветвления, циклы и обращения к подпрограммам, содержащиеся в программе.

2.5.3. Таблица идентификаторов

Приводится описание всех основных и вспомогательных переменных, используемых при разработке программы с указанием их имен, смыслового содержания, типа данных, области видимости и т. д. Необходимо обоснованно назначить и привести в записке всестороннюю характеристику для каждой переменной.

Пример оформления используемых переменных представлен в таблице 2.3.

Таблица 2.3- Характеристика переменных, используемых в программе

Переменная в формуле	Обозначение в программе	Тип данных	Область видимости	Пояснение
R	R	вещественный	глобальная	Радиус поворота
v	v	вещественный	глобальная	Скорость трактора
v_{min}	vMin	вещественный	глобальная	Минимальная скорость трактора
v_{max}	vMax	вещественный	глобальная	Максимальная скорость трактора
Δv	deltaV	вещественный	глобальная	Приращение скорости гусеницы
Δv_{min}	deltaVmin	вещественный	глобальная	Минимальное приращение скорости гусеницы
Δv_{max}	deltaVmax	вещественный	глобальная	Максимальное приращение скорости гусеницы
B	B	вещественный	глобальная	Ширина колеи
-	i	целый	локальная	Параметр цикла

2.5.4. Разработка интерфейса пользователя

В данном разделе приводится подробное описание действий, которые будут разрешены пользователю разработанного в курсовой работе программного продукта:

- как и где можно задать исходные данные,
- какие элементы управления будут доступны пользователю,
- как и где будет отражен результат

Приводится иллюстрация внешнего вида интерфейса.

Пример. При запуске программы появляется главное окно, представленное на рисунке 2.3

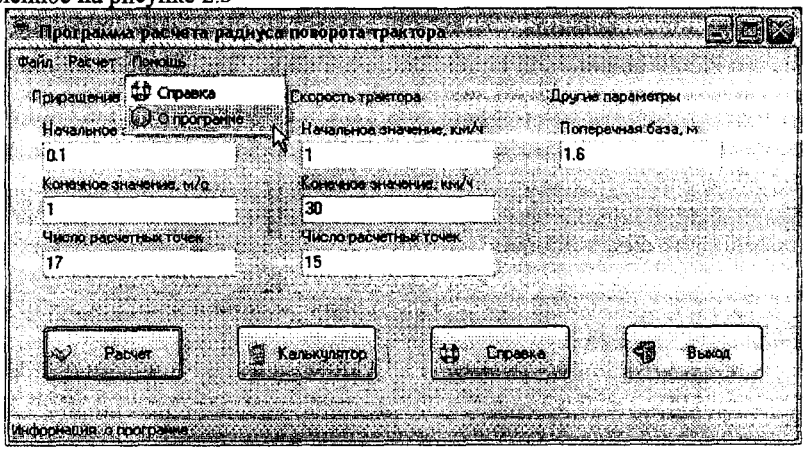


Рисунок 2.3 – Стартовое окно программы

Исходные данные для удобства ввода разбиты на три логических блока: *Приращение скорости гусеницы*, *Скорость трактора* и *Другие параметры*. Для проведения расчетов необходимо нажать кнопку *Расчет*. Если требуется выполнить дополнительные вычисления, то можно воспользоваться калькулятором, который вызывается нажатием кнопки *Калькулятор*.

Программа содержит падающее меню и контекстное меню (рисунок 2.4) из которого можно выполнить основные действия с программой.

Рисунок 2.4 – Контекстное меню программы

Для облегчения понимания работы с программой содержится статусная строка, в которую выводится справочная информация о любом объекте программы, к которому подводится курсор.

После выполнения расчетов результаты отображаются на новой форме в виде таблицы (рисунок 2.5).

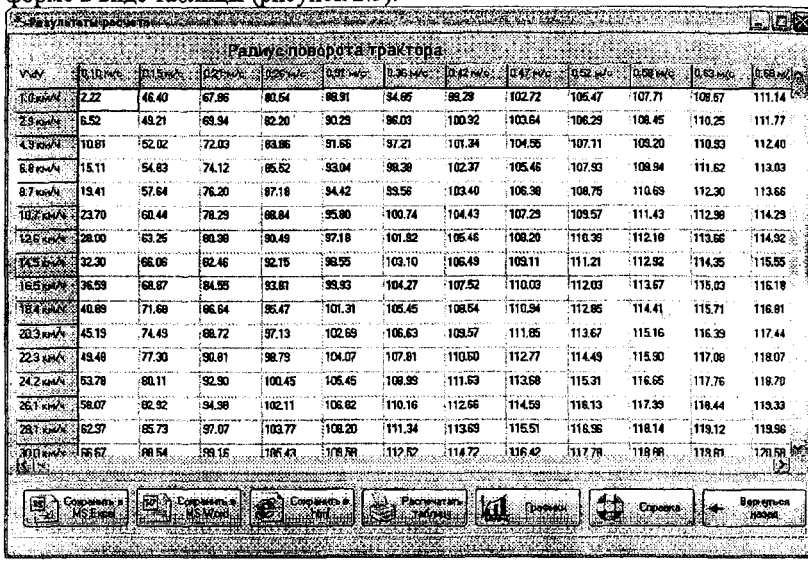


Рисунок 2.5 – Результаты расчета

На данной форме имеются ряд кнопок позволяющих провести постобработку полученных результатов. Можно сохранить результаты в *MS Excel*, *MS Word*, в формате *html*. Есть возможность построить графические зависимости или вывести результирующую таблицу на печатающее устройство.

При нажатии на кнопку *Графики* загружается новая форма (рисунок 2.6), на которой представлены графические зависимости радиуса поворота гусеничного трактора: от скорости движения при различных значениях приращений скорости гусеницы; от приращения скорости гусеницы при движении трактора с различной скоростью. Также предусмотрена возможность построения трехмерного графика - зависимость радиуса поворота трактора при одновременном изменении сразу двух параметров, указанных выше.

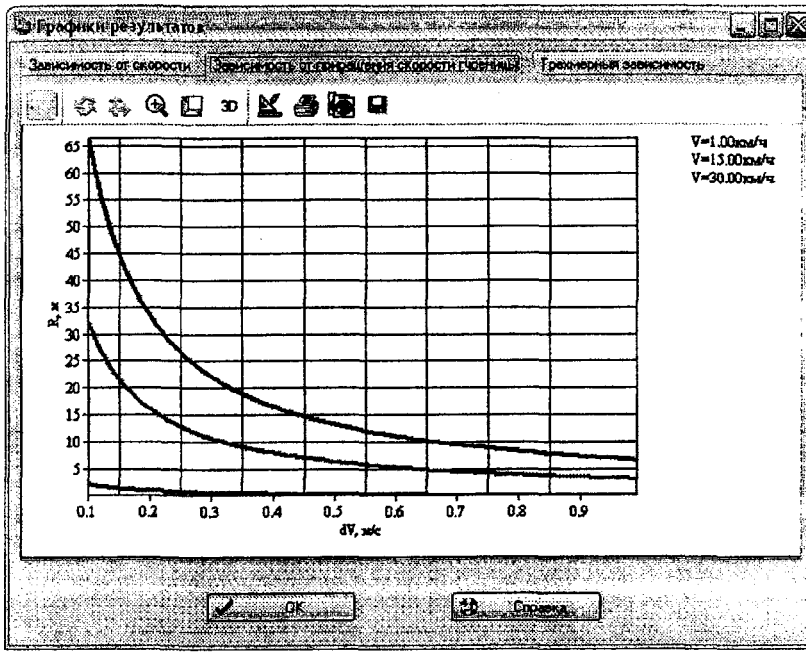


Рисунок 2.6 – Окно представления графических зависимостей.

Студентом в данном разделе должны быть представлены описания всех форм, меню и т. п., использованных в курсовой работе.

2.5.5. Структура программного приложения

В данном разделе приводятся сведения по реализации программного приложения в среде программирования *Delphi*.

Даются описания форм, модулей и т. д., их назначения, связей между ними.

Приводится описание объектов, примененных при разработке приложения с указанием их имен, значений важных свойств, назначенных автором, описание событий, связанных с этими управляющими элементами. Приводится описание процедур.

В данном разделе на отдельной странице размещается листинг (распечатка) программного кода решения задачи с дополнительными пояснениями (при необходимости).

Программа должна содержать следующие обязательные элементы:

- Падающее и контекстное меню.
- Статусную строку.
- Структурную обработку исключительных ситуаций.

– Динамически подключаемую библиотеку.
– Анимацию (располагается на форме *О программе*).
– Вызов внешнего приложения, например, калькулятора.
– Возможность импорта данных в *MS Excel* и *MS Word*.
– Графики зависимостей исследуемой функции от одного параметра при нескольких (не менее трех) фиксированных значениях второго параметра.
– Трехмерный график от двух изменяющихся параметров.
– Использование системных диалогов (печать результатов и графиков, открытие файлов, сохранение результатов и т. п.).
– Ввод-вывод данных через внешний файл.
– Вызов справочной информации на каждой форме.
– Другие необходимые в каждом конкретном случае элементы.

Ниже представлены некоторые фрагменты примеров перечисленных выше элементов программы.

2.5.5.1. Падающее и контекстное меню

Практически любое приложение должно иметь меню, поскольку именно меню дает наиболее удобный доступ к функциям программы. Существует несколько различных типов меню: главное меню с выпадающими списками разделов, каскадные меню, в которых разделу первичною меню ставится в соответствие список подразделов, и всплывающие или контекстные меню, появляющиеся, если пользователь щелкает правой кнопкой мыши на каком-то компоненте.

В *Delphi* меню обычно создаются компонентами *MainMenu*  — главное меню, и *PopupMenu*  — всплывающее (контекстное) меню. Оба компонента расположены на странице *Standard*.

Основное требование к меню — их стандартизация. Это требование относится ко многим аспектам меню: месту размещения заголовков меню и их разделов, форме самих заголовков, клавишам быстрого доступа, организации каскадных меню. Цель стандартизации — облегчит пользователю работу с приложением. Надо, чтобы пользователю не приходилось думать, в каком меню и как ему надо открыть или сохранить файл, как ему получить справку и т. д. Для осуществления всех этих операций у пользователя, поработавшего хотя бы с несколькими приложениями, вырабатывается стойкий автоматизм действий и недопустимо этот автоматизм ломать.

Названия разделов меню должны быть привычными пользователю. Если не понятно, как назвать какой-то раздел, то не следует изобретать свое имя, а попробовать найти аналогичный раздел в какой-нибудь русифицированной программе *Microsoft* для *Windows*. Названия должны быть краткими и понятными. Не следует использовать фразы, да и вообще больше двух слов, поскольку это перегружает экран и замедляет выбор пользователя. Названия разделов должны начинаться с заглавной буквы. Применительно к английским названиям разделов существует требование,

чтобы каждое слово тоже начиналось с заглавной буквы. Но применительно к русским названиям это правило не применяется.

В каждом названии раздела должен быть выделен подчеркиванием символ, соответствующий клавише быстрого доступа к разделу (клавиша &). Хотя вряд ли такими клавишами часто пользуются, но традиция указания таких клавиш незыблема.

Многие разделы меню желательно снабжать пиктограммами, причем пиктограммы для стандартных разделов должны быть общепринятыми, знакомыми пользователю.

Главное и контекстное меню создаются с помощью *Конструктора Меню*, который вызывается щелчком по соответствующей пиктограмме.

Контекстное меню привязано к конкретным компонентам. Оно всплывает, если во время нахождения данного компонента в фокусе, пользователь щелкнет правой кнопкой мыши. Обычно в контекстное меню включают те команды главного меню, которые в первую очередь могут потребоваться при работе с данным компонентом.

Один из вариантов вызова созданного ранее контекстного меню (рисунок 4) представлен ниже (обрабатывается событие *MouseUp* на форме)

```
procedure TFormHaupt.FormMouseUp(Sender: TObject;
Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
Var FCursor: TPoint;
     begin
  if Button=mbRight then
   begin
    GetCursorPos(FCursor);
    PopupMenu1.Popup(FCursor.X,FCursor.Y)
   end
    end;
```

После добавления данной процедуры контекстное меню будет вызываться нажатием правой кнопки мыши в любой области формы.

2.5.5.2. Статусная строка

Статусная строка создается с использованием компонента *SlatusBar*  , расположенного на странице *WIN32*, и представляет собой ряд панелей, отображающих полосу состояния в стиле *Windows*. Обычно эта полоса размещается внизу формы.

Свойство *SimplePanel* определяет, включает ли полоса состояния одну или множество панелей. Если *SimplePanel=true*, то вся полоса состояния представляет собой единственную панель, текст которой задается свойством *SimpleText*. Если же *SimplePanel=false*, то полоса состояния является набором панелей, задаваемых свойством *Panels*. В этом случае

свойство *SizeGrip* определяет, может ли пользователь изменять размеры панелей в процессе выполнения приложения.

Основное свойство каждой панели — *Text*, в который заносится отображаемый в панели текст. Его можно занести в процессе проектирования, а затем можно изменять программно во время выполнения. Другое существенное свойство панели — *Width* (ширина).

Программный доступ к текстам отдельных панелей можно осуществлять двумя способами: через индексированное свойство *Panels* или через его индексированное подсвойство *Items*. Например, два следующих оператора дадут идентичный результат:

```
StatusBar1.Panels[0].Text:='текст1';
```

или

```
StatusBar1.Panels.Items[0].Text:= 'текст1';
```

Оба они напечатают текст *«текст1»* в первой панели.

Можно задать подсказки для каждого компонента (свойство *Hint*), а затем выводить эти данные в статусную строку при наведении курсора на данный компонент. Реализовать это можно, например, следующим способом.

В разделе описания процедур добавляем (вручную) следующую надпись:

```
procedure ShowHint(Sender: TObject);//пишем вручную
```

Фрагмент описания представлен ниже:

```
type
 TFormHaupt = class(TForm)
  MainMenu1: TMainMenu;
  N1: TMenuItem;
  N2: TMenuItem;
  RzGroupBox1: TRzGroupBox;
  XPManifest1: TXPManifest;
  LabeledEdit1: TLabeledEdit;
  RzBitBtn1: TRzBitBtn;
  RzBitBtn2: TRzBitBtn;
  PopupMenu1: TPopupMenu;
  StatusBar1: TStatusBar;
  RzBalloonHints1: TRzBalloonHints;
  procedure FormCreate(Sender: TObject);
  procedure RzBitBtn1Click(Sender: TObject);
  procedure ShowHint(Sender: TObject);{дописываем
вручную}
  procedure FormCloseQuery(Sender: TObject; var Can-
Close: Boolean);
  procedure FormMouseUp(Sender: TObject; Button:
TMouseButton;Shift: TShiftState; X, Y: Integer);
  procedure RzGroupBox1MouseUp(Sender: TObject; But-
ton: TMouseButton;Shift: TShiftState; X, Y: Integer);
```

```
  procedure RzBitBtn2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
```

После этого можно создать еще две процедуры:

```
procedure TFormHaupt.FormCreate(Sender: TObject);
    begin
{данная процедура активизирует режим отображения под-
сказок, т. е. текстовых сообщений, указанных в свой-
стве Hint объекта}
Application.OnHint := ShowHint
    end;

procedure TFormHaupt.ShowHint(Sender: TObject);
    begin
{эта подпрограмма выводит в статусную строку тексто-
вые сообщения, записанные в свойстве Hint объекта}
StatusBar1.SimpleText:=Application.Hint
    end;
```

2.5.5.3. Структурная обработка исключительных ситуаций

В любом созданном приложении возможны ошибки. Кроме субъективных ошибок по вине программиста, существуют объективные ошибки, которые иногда нельзя избежать во время проектирования приложения, но они могут быть обнаружены во время его работы.

Исключительная ситуация (exception) – это состояние, возникающее в процессе выполнения программы из-за любых ошибок или ошибочных условий.

Хорошая программа должна работать безошибочно, без зацикливания и зависания.

Среда *Delphi* поможет произвести структурную обработку исключительных ситуаций.

Когда в программе обнаруживается ошибка, происходит генерация исключительной ситуации. Нормальное выполнение программы прерывается, и управление передается специальной части кода, которая выполняет обработку исключительной ситуации (рисунок 2.7). После обработки исключительной ситуации возврат в точку ее возникновения не происходит, а выполняются действия, следующие за телом обработчика. В итоге удается отделить смысловую часть алгоритма от обработчиков ошибок, в результате чего программа становится более простой, понятной и отказоустойчивой.

Рисунок 2.7 - Возникновение исключительной ситуации

Структурная обработка исключительных ситуаций - это совокупность операторов, позволяющих программисту, когда возникает ошибка (исключительная ситуация), связаться с кодом программы, подготовленным для обработки такой ошибки. Такая совокупность операторов как бы «охраняет» фрагмент кода программы и определяет обработчика ошибки, который вызывается, если что-то происходит не так в «охраняемом» участке кода.

Синтаксис обработки исключительных ситуаций. Всего существует два типа защищенных участков:

try..except и

try..finally

Первый тип используется для обработки исключительных ситуаций. Его синтаксис:

```
try
Statement 1;{защищенные от ошибок операторы}
Statement 2;
...
except
on Exception 1 do Statement; {операторы обработки исключительной ситуации}
on Exception 2 do Statement;
...
else
StatementN; {обработка всех остальных ошибок (обработчик по умолчанию)}
end;
```

Для уверенности в том, что ресурсы, занятые приложением, освободятся в любом случае, можно использовать конструкцию второго типа. Код, расположенный в части *finally*, выполняется в любом случае, даже если возникает исключительная ситуация. Соответствующий синтаксис представлен ниже:

```
try
Statement1; {защищенные от ошибок операторы}
Statement2;
...
```

```
finally
StatementN; {эти операторы всегда выполняются}
end;
```

Исключительные ситуации в *Delphi* описываются классами. Каждый класс соответствует определенному типу исключительных ситуаций. В то время, когда в программе возникает исключительная ситуация, создается объект соответствующего класса, который переносит информацию об этой ситуации из места возникновения в место обработки.

Классы исключительных ситуаций *Delphi* образуют иерархию, корнем которой является класс *Exeption*. Все имена классов исключительных ситуаций начинаются с буквы *E* (от слова *Exception*). В приложении приведена таблица с подробным описанием операторов, позволяющих реагировать на возникающие ошибки, их необходимо использовать в приложениях для защиты от зависания.

Пример использования в программе

```
//Необходимо ввести массу машины
 try      {Начало блока обработки исключительной ситуации}
  Massa:=StrToFloat(Edit1.Text); {считываются данные из компонента Edit1}
 except
   on EConvertError Do {ошибка преобразования строки}
     begin
      MessageDlg('Некорректно введены данные',
mtWarning,[mbOK],0);
{сообщение о неверном вводе данных}
      Edit1.SetFocus; {передача фокуса компоненту в котором были введены неверные данные}
      Exit
     end
 end; {Конец блока обработки исключительной ситуации}
```

Однако необходимо учитывать, что, например, введенные данные не приводят к возникновению исключительной ситуации, но, тем не менее, препятствуют нормальной работе приложения.

Рассматривая приведенный выше пример, можно ввести массу машины равной -5. В результате исключение *EConvertError* не наступит, но программа правильно работать не будет, поскольку масса машины должна быть положительным числом. В подобных ситуациях необходимо вводить дополнительную обработку, корректности введенных данных. Пример такой обработки приведен ниже.

```
 If (Massa<=1000) or (Massa>10000) Then
  begin
   MessageDlg('Значение массы машины должно находиться '
```

```
    +'в пределах от 1000 до 10000 кг',
    mtWarning,[mbOK],0);
    Edit1.SetFocus; {передача фокуса компоненту в ко-
тором были введены неверные данные}
    Exit
  end;
```

В приложении 11 представлены классы исключительных ситуаций и их описание.

2.5.5.4. Динамически подключаемые библиотеки

Динамически подключаемые библиотеки (*DLL — Dynamic Link Library*) являются ключевым компонентом при написании любого современного приложения *Windows*. Они представляют собой программные модули, содержащие процедуры и функции, данные или ресурсы, которые могут совместно использоваться несколькими различными приложениями *Windows*. Одно из основных и, пожалуй, главных назначений библиотек *DLL* - позволить загружать процедуры и функции в память только в случае необходимости во время выполнения приложения, а не компилировать их в само приложение и таким образом загружать в память всякий раз при запуске, независимо от того понадобятся они либо нет. Кроме того, несколько приложений могут использовать одни и те же процедуры и функции, хранящиеся в *DLL*, одновременно. И самое главное, библиотеки *DLL* являются универсальными относительно языка программирования, то есть *DLL* может быть написана на языке *C++*, а само приложение, использующее библиотеку — на *Delphi*.

Таким образом, преимущества динамически подключаемых библиотек:

1. *Универсальность*. Любой программист, зная описания и имена функций, находящихся в библиотеке, может использовать их. При этом ему совсем не обязательно знать, как работают эти функции. Главное - результат.

2. *Удобность отладки*. Можно разместить несколько важных функций в библиотеке и объявить их в программе. При этом программист будет работать с библиотекой, отлаживая эти функций, не трогая основную программу.

3. *Хранилища ресурсов*. В *DLL* можно хранить ресурсы, такие как рисунки, формы, иконки меню, и т. д.

4. *Модульность и расширение*. С помощью библиотек можно легко создавать плагины, расширяющие стандартные возможности программы. Т. е. можно не выпускать различные версии программы, а выпускать плагины или модифицированные (например, с исправленными ошибками) версии библиотек.

5. *Совместное использование*. Если библиотека загружена, то её могут использовать и другие приложения.

6. *Экономия ресурсов*. Библиотеку можно загрузить и выгрузить тогда, когда это действительно необходимо. Например, программа в нужный момент загрузила *DLL*, вызвала функцию, сделала работу и выгрузила библиотеку до следующего раза.

Создание динамически подключаемой библиотеки. Для создания новой динамической библиотеки нужно выбрать из меню *File* пункт *New* и затем *Other*. В окне создания нового проекта (рисунок 2.8) нужно выбрать на закладке *New* пункт *DLL Wizard*.

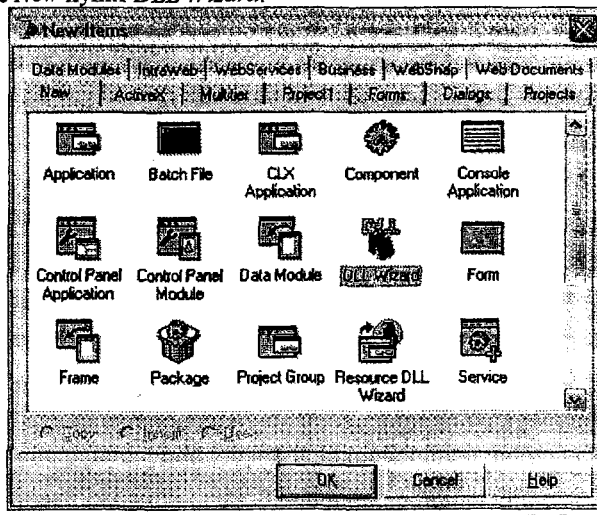


Рисунок 2.8 - Окно создания нового проекта *DLL*.

В результате будет создан пустой проект с одним только модулем, содержащим следующий текст:

```
library Project2;
uses
  SysUtils,
  Classes;
{$R *.res}
begin
end.
```

Сохранить созданный проект необходимо в отдельную папку, присвоив (желательно) осмысленное имя. Для этого необходимо выбрать из меню *File* пункт *Save All*, и будет предложено сохранить только один проект и никаких модулей. Указанное имя проекта буде присвоено откомпилированному *dll* файлу. Например, если имя проекта *Tractor.dpr*, то библиотека будет иметь имя *Tractor.dll*.

Теперь нужно добавить в библиотеку необходимое количество функций. В приведенном ниже примере добавляется функция с именем *Radius*.

```
library Tractor;
uses
  SysUtils,
  Classes;
Function Radius(V,deltaV,B:Real):Real; stdcall;
begin
 Radius:=(V/deltaV)*0.5*B;
end;
exports Radius index 10;
{$R *.res}
begin
end.
```

Ключевое слово *StdCall*, которое стоит после типа возвращаемого функцией значения, говорит о том, что для вызова процедуры нужно использовать стандартный тип вызова. Если не указать ключевое слово *StdCall*, то параметры будут передаваться способом, заложенным фирмой *Borland*. Этот способ работает быстрее, но он не совместим со стандартными правилами. Это означает, что если к библиотеке будут обращаться программы сторонних разработчиков, например на языках *Visual C++* или других то будут проблемы.

После описания функции идёт ключевое слово *exports*. За ним должны идти описания подпрограмм, которые должны быть доступны внешним программам. Если функцию *Radius* не описать в разделе *exports*, то её будет невозможно вызвать из внешней программы.

После имени функции может стоять ключевое слово *index* с числом. Данный индекс не является обязательным, в этом случае вызов подпрограммы будет осуществляться по ее имени. Однако когда программе нужно будет выполнить функцию *Radius*, то она будет просматривать все функции динамической библиотеки и искать функцию с указанным именем. Это очень неэффективно и перед первым вызовом будет ощущаться большая задержка. Чтобы хоть немного ускорить процесс вызова таких функций желательно использовать индексы. В качестве которого выступает целое число в диапазоне от 0 до 32767. Индексы и имена должны быть уникальными!

Теперь необходимо *откомпилировать* (а не запустить) проект (*Ctrl+F9* или выбрать команду *Compile* из меню *Project*). В результате будет создана динамическая библиотека *Tractor.dll*.

Не нужно пытаться запускать проект, потому что это библиотека, и она не может выполняться самостоятельно. Поэтому при попытке запуска появится сообщение об ошибке (рисунок 2.9):

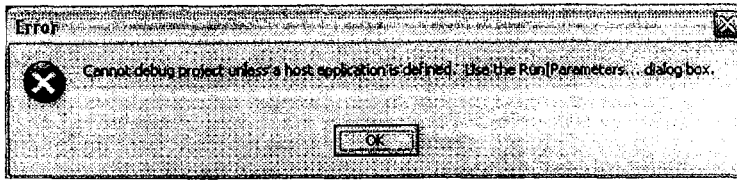


Рисунок 2.9 – Сообщение компилятора при попытке запуска проекта *DLL*-библиотеки

Использование динамически подключаемой библиотеки. Связать функцию из *DLL*-библиотеки с приложением можно двумя методами: статической и динамической загрузки. Рассмотрим способом статической загрузки как наиболее простой и легкий в использовании метод, не требующий от программиста знания *WinAPI*. Однако у этого способа имеется и ряд недостатков. В частности, библиотека загружается при запуске приложения и выгружается при завершении, таким образом, нет экономии ресурсов; при отсутствии хотя бы одной из необходимых библиотек, или подпрограмм в библиотеке, дальнейшее выполнение приложения не представляется возможным.

Для использования функций или процедур из библиотеки необходимо в разделе *implementation* в программном модуле приложения дать ссылку на соответствующую подпрограмму. Фрагмент кода приведен ниже:

```
...
    implementation
    function  Radius(V,deltaV,B:Real):Real;  external
'Tractor.dll' index 10;
    {$R *.dfm}
...
```

В данном примере приведено ключевое слово *external* после которого следует имя подключаемой *DLL* библиотеки. Желательно, чтобы сама библиотека располагалась в том же каталоге, что и основная программа. Теперь функцию можно использовать в программе.

2.5.5.5. Создание анимации

Каждый студент примерно представляет себе принципы создания анимации: совокупность множества кадров, отличающихся немного друг от друга. Это при быстром поочередном просмотре кадров и создает иллюзию движения. В своей работе вряд ли придется рисовать с помощью *Delphi* мультфильмы. Для этого имеются совершенно другие инструменты. Но некоторые простые анимации - оживление изображений, иногда желательно делать. Например, при создании какой-нибудь обучающей программы можно оживить какие-то схемы или условные изображения механизмов, чтобы показать в движении взаимодействие их отдельных составляющих.

В качестве примера создадим анимацию движения объекта на колесах по дороге (рисунок 2.10).

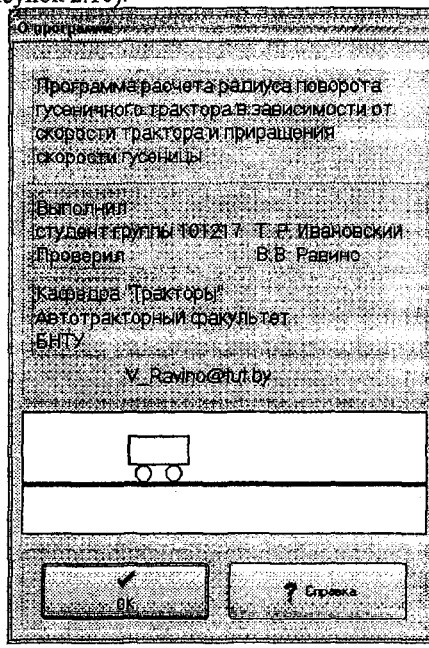


Рисунок 2.10 – Окно *О программе* с анимированным изображением

Порядок создания следующий.

Открываем новое приложение. Переносим на форму компоненты *Image* (страница *Additional*) и *Timer* (страница *System*). Анимация будет запускаться при создании формы (событие *Create*) и воспроизводится непрерывно.

Таймер будет задавать темп смены кадров. Следует задать значение свойства *Interval* таймера равным 100 (поскольку интервал задается в миллисекундах, то это значение соответствует 0,1 секунды). Размещение компонентов закончено. Теперь надо ввести текст программы. Ее раздел *implementation* имеет следующий вид:

```
const
  h=52; {длина корпуса}
  b=25; {высота корпуса}
  yz=60; {высота (сверху вниз!!) расположения дороги}
  rk=7; {радиус колеса}
...
var
    xn:integer;
...
```

```
Procedure drawing;
    begin
with FormAbout.Image1.Canvas do
        begin
rectangle(xn,yz-2*rk-b,xn+h,yz-2*rk);{корпус машины}
ellipse((xn+h div 4-rk),(yz-2*rk),
        (xn+h div 4+rk),yz); {одно колесо машины}
ellipse((xn+h div 4-rk+h div 2),(yz-2*rk),
        (xn+h div 4+rk+h div 2),yz){другое колесо}
        end
    end;

procedure clean;
    begin
FormAbout.image1.Canvas.rectangle(0, 0,
   FormAbout.image1.Width, FormAbout.image1.Height)
    end;

procedure road;
    begin
with FormAbout.image1.Canvas do
        begin
 pen.Width:=4;{перо толщиной 4 единицы}
 moveto(0,yz+2);{перемещаем курсор в указанную точку}
 lineto(FormAbout.Image1.Width,yz+2);{рисуем линию}
 pen.Width:=1 {перо толщиной 1 единица}
        end
    end;

procedure TFormAbout.FormCreate(Sender: TObject);
    begin
xn:=1
    end;

procedure TFormAbout.Timer1Timer(Sender: TObject);
    begin
clean;
road;
drawing;
xn:=xn+10;
if xn>=(image1.Width-h) then xn:=1
    end;
```

Формирование анимации заключается в использовании (в данном случае) трех пользовательских подпрограмм *clean*, *road* и *drawing*. Процедура *clean* очищает окно от всех объектов. Процедура *road* создает гори-

зонтальную линию, соответствующей дороге, по которой движется объект. Процедура *drawing*создает сам движущийся объект (прямоугольный корпус и два колеса). Следующий объект рисуется со смещением по горизонтали на 10 единиц. Далее кадры повторяются с интервалом в 0,1 секунды (свойство таймера) до тех пор, пока форма не будет закрыта.

Оператор

```
if xn>=(image1.Width-h) then xn:=1;
```

позволяет отрисовывать изображение заново, когда объект достиг правой границы окна.

2.5.5.6. Вызов внешнего приложения

Часто при работе с программой необходимо вызвать внешнее приложение, например, калькулятор, файл со справочными данными и т. п. Сделать это можно используя функцию *WinExec*.

Синтаксис функции следующий:

```
function WinExec(CmdLine:PChar;
                 CmdShow:integer):integer;
```

Параметр *CmdLine* является указателем на строку с нулевым символом в конце, содержащую имя выполняемого файла и, если необходимо, параметры командной строки.

Если имя указано без пути, то *Windows* ищет выполняемый файл в следующей последовательности:

1. Каталог, из которого загружено приложение.
2. Текущий каталог,
3. Системный каталог *Windows*.
4. Список каталогов из переменной окружения *PATH*.

Параметр *CmdShow* определяет форму представления окна запускаемого приложения *Windows*. Для приложений не *Windows*, для файлов *PIF* и т. д. состояние окна определяет само приложение.

Достоинством функции *WinExec* является ее совместимость с ранними версиями *Windows*.

При работе с *Win32* функция *WinExec* завершает работу, если вызванное приложение вызывает функцию *GetMessage* или заканчивается выделенный лимит времени. Таким образом, ожидание можно прервать, предусмотрев в процессе, запущенном с помощью *WinExec*, в нужный момент вызов функции *GetMessage*.

Примеры.

Запуск программы *file.exe*

```
WinExec ('file.exe', SW_RESTORE);
```

Запуск *MS-DOS*

```
WinExec('COMMAND.COM',SW_RESTORE);
```

2.5.5.7. Импорт данных в Microsoft Excel и Microsoft Word

Общеизвестно, что текстовый процессор *MS Word* и табличный процессор *MS Excel* имеют огромные возможности и постоянно совершенствуются. Они получили очень широкое распространение, и, самое главное, они - очень удобный инструмент для создания документов любой сложности.

В качестве примеров ниже рассмотрены несколько функций, которые позволят запустить *MS Word*, создать документ, изменить документ (записать текст), сохранить документ и закрыть *MS Word*. Для создания объекта и его использования применяем переменную *W* типа *Variant* и библиотеку *ComObj*.

Рассмотрим следующий фрагмент кода:

```
uses ComObj;
var W:variant;
Function CreateWord:boolean;
begin
 CreateWord:=true;
 try
 W:=CreateOleObject('Word.Application');
 except
 CreateWord:=false
 end;
End;
```

Для получения доступа к объекту *Word.Application* в функции *CreateWord* используем конструктор *CreateOleObject ('Word. Application')*. Если редактор MS Word не установлен в системе, то будет сгенерирована ошибка, и получится значение функции равное *false*, если *MS Word* установлен, и объект будет создан, то получится значение функции равное *true*.

Эта функция создает объект *W*, свойства и методы которого можно использовать в дальнейшем. Если выполнить функцию *CreateWord*, то *MS Word* будет запущен, но не появится на экране, потому что по умолчанию он запускается в фоновом режиме. Чтобы его активировать (сделать видимым) или деактивировать (сделать невидимым), необходимо использовать свойство *Visible* объекта *W*. Оформить это можно в виде функции *VisibleWord*:

```
Function VisibleWord (visible:boolean):boolean;
begin
 VisibleWord:=true;
 try
 W.visible:= visible;
 except
 VisibleWord:=false
 end
End;
```

Для создания документа используется объект *Documents* объекта *W*. Этот объект имеет метод *Add*, используя который, и создается новый документ. Пример приведен ниже.

```
Function AddDoc:boolean;
Var Doc_:variant;
begin
 AddDoc:=true;
 try
 Doc_:=W.Documents;
 Doc_.Add;
 except
 AddDoc:=false
 end
End;
```

После создания документа необходимо записать какой-либо текст непосредственно в документ. Для этого используется функция *SetTextToDoc*:

```
Function SetTextToDoc(text_: string;InsertAfter_:
boolean): boolean;
var Rng_:variant;
begin
 SetTextToDoc:=true;
 try
 Rng_:=W.ActiveDocument.Range;
 if InsertAfter_
  then Rng_.InsertAfter(text_)
  else Rng_.InsertBefore(text_);
 except
 SetTextToDoc:=false
 end
End;
```

В этой функции используется объект *Range* и его методы *InsertAfter* и *InsertBefore* для того, чтобы вставить текст в документ с позиции курсора или до позиции курсора.

После того, как документ создан и в него записан текст, его необходимо сохранить. Для этого используется метод *SaveAs* объекта *ActiveDocument*. Функция *SaveDocAs* использует этот метод и сохраняет документ в заданный файл.

```
Function SaveDocAs(file_:string):boolean;
begin
 SaveDocAs:=true;
 try
 W.ActiveDocument.SaveAs(file_);
 except
 SaveDocAs:=false
```

```
 end
End;
```

Закрыть сохраненный документ можно, используя метод *Close* объекта *ActiveDocument*.

```
Function CloseDoc:boolean;
begin
 CloseDoc:=true;
 try
 W.ActiveDocument.Close;
 except
 CloseDoc:=false
 end
End;
```

Закрыть редактор *MS Word* можно, используя метод *Quit* объекта *W*.

```
Function CloseWord:boolean;
begin
 CloseWord:=true;
 try
 W.Quit;
 except
 CloseWord:=false
 end
End;
```

Таким образом, можно создать документ, записать в него текст, сохранить документ и отобразить его на экране монитора.

Работа с *MS Excel* осуществляется аналогичным образом. Самыми необходимыми функциями являются следующие: активизация *MS Excel*, создание новой книги, открытие ранее созданной книги, отображение книги (книг) и приложения *MS Excel*, запись информации в ячейку, запись книги на диск и выход (закрытие книги и приложения). Для создания объекта *Excel.Application* используя переменная *E* типа *Variant* и библиотека *ComObj*.

Доступ к объекту *Excel.Application* в функции *CreateExcel* получается с использованием процедуры *CreateOleObject ('Excel.Application')* стандартной библиотеки *ComObj*. Если *MS Excel* не установлен в системе, то будет сгенерирована ошибка и значение функции будет равно *false*; если *MS Excel* установлен и объект будет создан, то получится значение функции равное *true*. Эта функция создает объект *E*, свойства и методы которого можно использовать в дальнейшем.

```
uses ComObj, Classes;
var E:variant;
Function CreateExcel:boolean;
begin
 CreateExcel:=true;
 try
```

```
  E:=CreateOleObject('Excel.Application');
 except
  CreateExcel:=false
 end
end
End;
```

Если выполнить функцию *CreateExcel*, то *MS Excel* будет запущен, но не будет отображен, потому что по умолчанию он запускается в фоновом режиме. Чтобы его активизировать (сделать видимым) или деактивировать (сделать невидимым), нужно использовать свойство *Visible* объекта *E*. Пример оформлен в виде функции *VisibleExcel*.

```
Function VisibleExcel(visible:boolean): boolean;
begin
 VisibleExcel:=true;
 try
  E.visible:=visible;
 except
  VisibleExcel:=false
 end
End;
```

Для создания рабочей книги используется метод *Add*, коллекции *Workbooks* объекта *E*. Ниже приведен пример, с помощью которого не только создается новая книга, но и получаеся на нее ссылка

```
Function AddWorkBook:boolean;
begin
 AddWorkBook:=true;
 try
  E.Workbooks.Add;
 except
  AddWorkBook:=false
 end
End;
```

Чтобы открыть ранее созданную рабочую книгу, используется та же коллекция *Workbooks* объекта *E* и метод *Open*. Пример представлен ниже.

```
Function OpenWorkBook(file_: string):boolean;
begin
 OpenWorkBook:=true;
 try
  E.Workbooks.Open(file_);
 except
  OpenWorkBook:=false
 end
End;
```

При работе с книгой, желательно иметь возможность добавлять или удалять в ней листы и присваивать им имена. Для работы с листами книги

используется коллекция *Sheets*. Добавить новый лист можно, используя метод *Add* этой коллекции. Функция *AddSheet* реализует эту возможность и присваивает новому листу выбранное пользователем имя.

```
Function AddSheet(newsheet:string):boolean;
begin
 AddSheet:=true;
 try
  E.Sheets.Add;
  E.ActiveSheet.Name:=newsheet;
 except
  AddSheet:=false
 end
End;
```

Метод *Delete*, коллекции *Sheets* дает возможность удалять созданные листы. При удалении возможно появление диалогового окна *MS Excel*, которое потребует подтверждения операции. Для отключения диалогового окна *MS Excel* используется оператор `E.DisplayAlerts:=False`.

Пример функции для удаления листов.

```
Function DeleteSheet(sheet:variant):boolean;
begin
 DeleteSheet:=true;
 try
  E.DisplayAlerts:=False;
  E.Sheets[sheet].Delete;
  E.DisplayAlerts:=True;
 except
  DeleteSheet:=false
 end
End;
```

Обычно книга *MS Excel* содержит более одного листа. Их количество содержится в свойстве *Count* коллекции *Sheets*. Для активации любого листа книги необходимо использовать процедуру *Select*. Например:

```
Sheets.Item(a_).Select
```

Все описанные выше возможности можно реализовать в *Delphi* как набор отдельных функций. Примеры приведены ниже:

```
Function CountSheets:integer; {получаем количество
листов книги}
begin
try
CountSheets:=E.ActiveWorkbook.Sheets.Count;
except
CountSheets:=-1
end
End;
```

```
Function GetSheets(value:TStrings):boolean;
// записываем листы книги в value
var a_:integer;
begin
GetSheets:=true;
value.Clear;
try
for a_:=1 to E.ActiveWorkbook.Sheets.Count do
value.Add(E.ActiveWorkbook.Sheets.Item[a_].Name);
except
GetSheets:=false;
value.Clear
end
End;

Function SelectSheet (sheet:variant):boolean;
// выбираем лист
begin
SelectSheet:=true;
try
E.ActiveWorkbook.Sheets.Item[sheet].Select;
except
SelectSheet:=false
end
End;
```

После внесения изменений необходимо сохранить рабочую книгу. Для этого используется метод *SaveAs* коллекции *Workbooks* или объекта *ActiveWorkbook*. Функция *SaveWorkBookAs* реализует эту возможность на *Delphi*.

```
Function SaveWorkBookAs(file_:string): boolean;
begin
SaveWorkBookAs:=true;
try
E.DisplayAlerts:=False;
E.ActiveWorkbook.SaveAs(file_);
E.DisplayAlerts:=True;
except
SaveWorkBookAs:=false
end
End;
```

Одновременно может быть открыто несколько книг, в которые вносится или из которых получается информация. Их количество содержится в свойстве *Count* коллекции *WorkBooks*.

Активизируется любая книга из списка процедурой *Activate*:

```
Windows("Книга1").Activate;
```

Для закрытия книги используется метод *Close* коллекции *Workbooks* или объекта *ActiveWorkbook*. Функция *CloseWorkBook* закрывает активный документ.

```
Function CloseWorkBook:boolean;
begin
 CloseWorkBook:=true;
 try
  E.ActiveWorkbook.Close;
 except
  CloseWorkBook:=false
 end
End;
```

MS Excel закрывается методом *Quit* объекта *Application*.

```
Function CloseExcel:boolean;
begin
 CloseExcel:=true;
 try
  E.Quit;
 except
  CloseExcel:=false
 end
End;
```

Для более подробного изучения возможностей по работе с *MS Word* и *MS Excel* следует обратиться к источнику [7]. Здесь же в качестве примера представлены фрагменты процедур по импорту данных в *MS Word* и *MS Excel*.

Импорт в *MS Word*

```
procedure TFormRez.RzBitBtn6Click(Sender: TObject);
{в раздел uses надо вставить ComObj}
var
  WordApp, NewDoc, WordTable: OLEVariant;
  iRows, iCols, iGridRows, jGridCols: Integer;
begin
  if Rzsavedialog3.Execute=false then exit; {Указыва-
ем имя файла Word}
  try
    WordApp := CreateOleObject('Word.Application');
{запускаем Word}
  except
    Exit
  end;
  NewDoc := WordApp.Documents.Add; {Создаем новый до-
кумент в среде Word}
  iCols := StringGrid1.ColCount;
```

```
  iRows := StringGrid1.RowCount;
{Перемнные количества строк и столбцов импортируемой
таблицы}
WordTable:=NewDoc.Tables.Add(WordApp.Selection.Range,
iRows, iCols); {Формируем заготовку таблицы}
  WordTable.Range.Font.Size :=8; {Размер текста сим-
волов в таблице 8}
 for iGridRows := 1 to iRows do
  for jGridCols := 1 to iCols do
   WordTable.Cell(iGridRows, jGridCols).Range.Text:=
    StringGrid1.Cells[jGridCols - 1, iGridRows - 1];
{заполняем таблицу в Word данными из StringGrid}
  WordApp.ActiveDocument.SaveAs(RzSaveDialog3.FileName); {Со-
храняем документ под указанным ранее именем}
  WordApp.Quit; {Выход из Word}
  WordApp := Unassigned;
  NewDoc := Unassigned;
  WordTable := Unassigned
end;
```

Импорт в MS Excel

```
procedure TFormRez.RzBitBtn2Click(Sender: TObject);
{в раздел uses надо вставить ComObj}
Var XLApp,sheet,colum:variant;
i,j,m,n:integer;
begin
  if Rzsavedialog2.Execute=false then exit;{указываем
имя файла Excel}
  try
XLApp:=CreateOleObject('Excel.Application'); {запус-
каем Excel}
  except
    Exit
  end;
XLApp.Workbooks.Add(-4167);{Создаем рабочую книгу}
XLApp.Workbooks[1].WorkSheets[1].Name:='Результат';
{Создаем рабочий лист с именем Результат}
Colum:=XLApp.Workbooks[1].WorkSheets['Результат'].Columns;
n:=StrToInt(FormHaupt.LabeledEdit4.text);
m:=StrToInt(FormHaupt.LabeledEdit3.text);
{количество строк и столбцов импортируемой таблицы}
 for i:=1 to m+1 do
Colum:=XLApp.Workbooks[1].WorkSheets['Результат'].Rows;
 Colum.columns[1].Font.Bold:=true; {шрифт первого
столбца Excel полужирный}
```

```
 Colum.Rows[1].Font.Bold:=true; {шрифт первой строки
Excel полужирный}
 Colum.Rows[1].Font.italic:=true; {шрифт первой
строки Excel курсив}
 Colum.Rows[2].Font.Bold:=true; {шрифт второй строки
Excel полужирный}
 Sheet:=XLApp.Workbooks[1].WorkSheets['Результат'];
  for i:=0 to m do
     for j:=0 to n do
sheet.cells[j+2,i+1]:=StringGrid1.Cells[i,j];
{копирование данных из Stringgrid}
 sheet.cells[1,2]:='Радиус поворота трактора'; {Заго-
ловок таблицы}
{После внесения изменений необходимо сохранить рабочую
книгу. Для этого используем метод SaveAs коллекции Work-
books или объекта ActiveWorkbook. Функция SaveWorkBookAs
реализует эту возможность на Delphi. Используем
E.DisplayAlerts:=False(True) для отключения (включения)
диалогового окна подтверждения записи}
XLApp.ActiveWorkbook.SaveAs(RzSaveDialog2.FileName);
{Сохраняем Excel под указанным выше именем}
XLApp.Quit {Выход их Excel}
end;
```

При необходимости можно аналогичным образом импортировать и другие данные в *MS Word* и *MS Excel*.

2.5.5.8. Построение графических зависимостей

Построение графических зависимостей удобнее всего выполнять с использованием компонента *Chart* или *Chart Pro*  .

Каждый такой объект представляет серию данных, характеризующихся определенным стилем отображения: тем или иным графиком. Каждый компонент *Chart* может включать несколько серий. При отображении графика каждая серия будет соответствовать одной кривой на графике (рисунок 2.11).

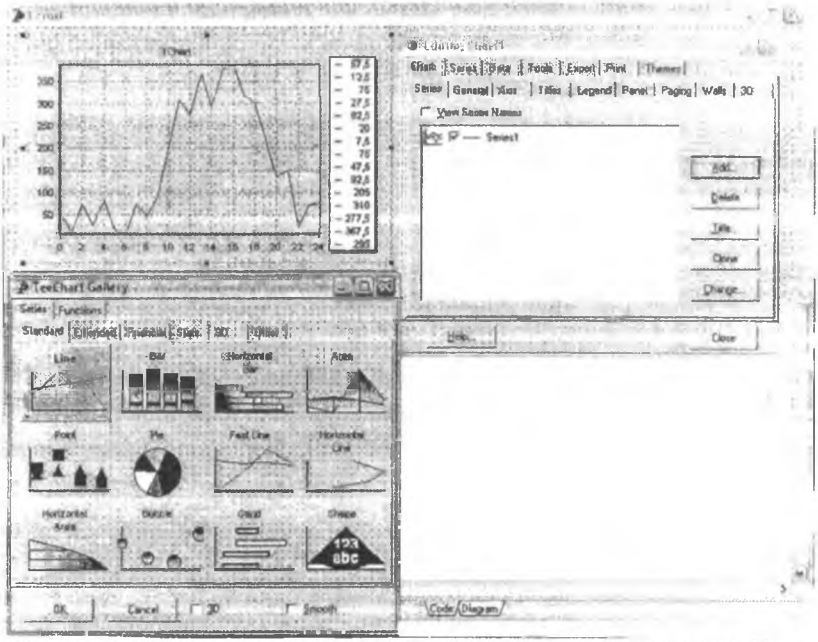


Рисунок 2.11 – Предварительная настройка графиков в компоненте *Chart*

Для задания отображаемых значений надо использовать методы серий *Series*. Основными методами при построений графиков являются *Clear* и *AddXY*.

Метод *Clear* очищает серию от занесенных ранее данных.

Метод *AddXY имеет следующий формат*:

```
AddXY(Const AXValue, AYValue:Double;
      Const ALabel:String; AColor: TColor)
```

и позволяет добавить новую точку в график функции. Параметры *AXValue* и *AYValue* соответствуют аргументу и функции. Параметр *ALabel* - метка, которая будет отображаться на диаграмме и в легенде, *AColor* – цвет кривой графика. Параметры *ALabel* и *AColor* являются не обязательными и их можно не задавать.

Таким образом, процедура построения графика, например, синусоиды может выглядеть следующим образом.

```
var x:real;
    begin
  Chart1.Title.Caption:='Синусоида';
  Series1.Clear;
  x:=0;
     Repeat
   Series1.AddXY(x, sin(x));
```

```
    x:=x+0.01
      Until x>=2*pi
    end;
```

Оператор *Clear* нужен обязательно, потому что в процессе работы приложения происходит обновление графиков. Без него повторное выполнение метода *AddXY* добавит новые точки, не удалив прежние.

Построение трехмерной зависимости осуществляется методом *AddXYZ*, однако такая возможность существует только в *Chart Pro*.

Метод *AddXYZ имеет следующий формат*:

```
AddXYZ(Const AX, AY, AZ: TChartValue)
```

и работает аналогично, рассмотренным выше способом.

В качестве примера построим график функции

$$f(x, y) := \sin(x) + \cos(y).$$

```
var x,y,f:real;
    begin
   x:=-2*pi;
 Repeat
  y:=-2*pi;
   repeat
    f:=sin(x)+cos(y);
    Series1.AddXYZ(x,f,y);
    y:=y+0.05
   until (y>=2*pi);
  x:=x+0.05
 until (x>=2*pi)
    end;
```

Полученный результат представлен на рисунке 2.12.

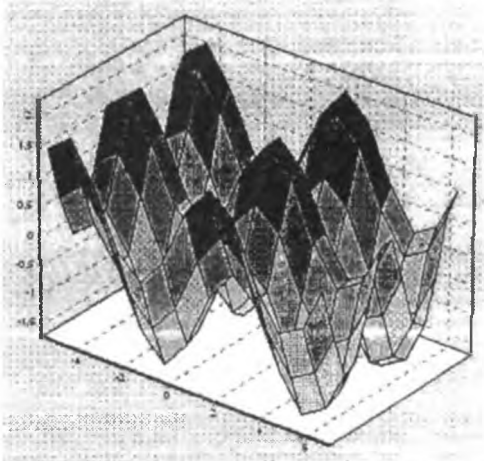


Рисунок 2.12 – Трехмерный график, построенный с использованием компонента *Chart Pro*

Аналогичным способом строятся графики, практически любой сложности.

2.5.5.9. Использование системных диалогов

В приложениях часто приходится выполнять стандартные действия: открывать и сохранять файлы, задавать атрибуты шрифтов, выбирать цвета палитры, производить контекстный поиск и замену и т. п.

В *Delphi* существуют простые для использования компоненты, реализующие соответствующие диалоговые окна *Windows*. Основная часть их размещена на странице *Dialogs* (рисунок 2.13).

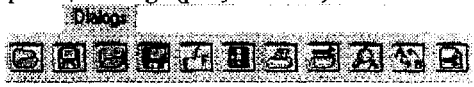


Рисунок 2.13 - Компоненты страницы *Dialogs*

Все компоненты стандартных диалогов со страницы *Dialogs* являются невизуальными компонентами, так что место их размещения на форме не имеет значения. При обращении к этим компонентам вызываются стандартные диалоги, вид которых зависит от версии *Windows* и настройки системы. Так что при запуске одного и того же приложения на компьютерах с разными системами диалоги будут выглядеть по-разному. Например, при русифицированной версии *Windows* все их надписи будут русскими, а при англоязычной версии надписи будут на английском языке.

Основной метод, которым производится обращение к любому диалогу, *Execute*. Эта функция открывает диалоговое окно и, если пользователь произвел в нем какой-то выбор, то функция возвращает *true*. При этом в свойствах компонента - диалога запоминается выбор пользователя, который можно прочитать и использовать в дальнейших операциях. Если же пользователь в диалоге нажал кнопку *Отмена (Cancel)* или клавишу *Esc*, то функция *Execute* возвращает *false*. Поэтому стандартное обращение к диалогу имеет вид:

```
if <имя компонента-диалога> Execute
then <операторы, использующие выбор пользователя>
```

При использовании диалогов открытия или сохранения файлов, необходимо предварительно задать фильтр типов файлов по расширению.

Делается это с помощью свойства *Filter* в *Инспекторе Объектов*. В окне редактора (рисунок 2.14)

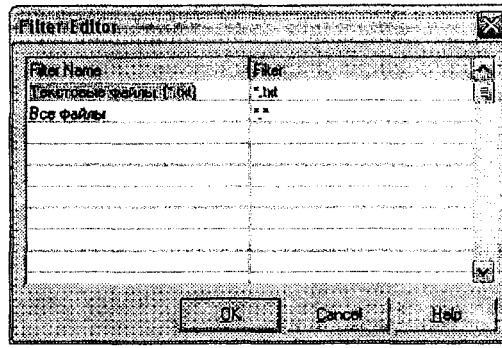


Рисунок 2.14 – Окно редактора свойства *Filter*

В его левой панели *Filter Name* записывается текст, который увидит пользователь в выпадающем списке *Тип файла* диалога. А в правой панели *Filter* записываются разделенные точками с запятой шаблоны фильтра. В примере задано два фильтра: текстовых файлов с расширением *.txt* и любых файлов с шаблоном *.*

Рассмотрим два примера использования системных диалогов.

Пример использования *OpenDialog*. Пусть требуется открыть файл для чтения. Содержимое строки файла необходимо вывести в компонент *Edit1*. Для этого размещаем на форме компоненты *Edit1*, *OpenDialog1* и *Button1*. Программный код представлен ниже.

```
procedure TForm1.Button1Click(Sender: TObject);
 var
  Fr:TextFile; //файловая переменная
  S: String;  // содержимое компонента Edit1
  Begin
if OpenDialog1.Execute then {Активизация диалогового
окна}
    begin
   AssignFile(Fr, OpenDialog1.FileName); {Указывается
где и какой файл необходимо открыть}
    Reset(Fr); //Открыть файл для чтения
    Readln(Fr, S); //Чтение первой строки из файла
    Edit1.Text:= S; {Помещение строки в компонент
Edit1}
   CloseFile(Fr) // Закрытие файла
    end
  End;
```

Нажатие на кнопку *Button1* открывает диалоговое окно открытия необходимого файла. После выбора файла и нажатия на кнопку в *Edit* записывается строка файла.

Пример использования *SaveDialogs*. Пусть требуется сохранить в файл содержимое компонента *Edit1*. Для этого размещаем на форме компонент *Edit1*, *SaveDialog1* и *Button1*. Программный код представлен ниже.

```
procedure TForm1.Button1Click(Sender: TObject);
var
Fw: TextFile; // файловая переменная
S: string; {содержимое компонента Edit1}
     Begin
 if SaveDialog1.Execute then
{Активизация диалогового окна}
       begin
  AssignFile(Fw, SaveDialog1.FileName);
{Указываем где и с каким именем будет сохранен файл}
  Rewrite(Fw); // открываем файл для записи
  S:=Edit1.Text; {присваиваем переменной S содержимое
компонента Edit}
  Writeln(Fw, S); //запись данных в файл
  CloseFile(Fw) //закрытие файла
      end
    End;
```

Нажатие на кнопку *Button1* открывает диалоговое окно сохранения файла под необходимым именем. После указания пути расположения, имени файла и нажатия на кнопку, в файл записывается содержимое компонента *Edit1*.

Для вывода на печать можно использовать компоненты *PrintDialog*, *PrintSetupDialog* и *PageSetupDialog*.

Компонент *PrintDialog* реализует стандартное диалоговое окно выбора параметров для печати документа. Компонент *PrintSetupDialog* реализует стандартное диалоговое окно для настройки печатающего устройства. Компонент *PageSetupDialog* реализует стандартное диалоговое окно настройки страницы перед выводом на печатающее устройство.

В качестве примера приведены две процедуры. Первая организует печать графика на всю страницу формата А4 в альбомной ориентации, вторая – выводит на печать содержимое компонента *RichEdit*.

```
procedure TForm1.BitBtn2Click(Sender: TObject);
var h,w:longint;
    OldOrientation : TPrinterOrientation;
begin
 If PrintDialog1.Execute Then {Печать графика}
  begin
 Screen.Cursor := crHourGlass; {устанавливаем вид
курсора}
 OldOrientation:=Printer.Orientation; {запоминаем
ориентацию бумаги}
```

```
 Printer.Orientation:=poLandScape; {устанавливаем
альбомную ориентацию}
  try
    Printer.BeginDoc;     {начало работы принтера}
    try
      h:=Printer.PageHeight; {идентификатор высоты
принтера}
      w:=Printer.PageWidth;  {идентификатор ширины
принтера}
     {начало печати компонентов диаграммы}
   Chart1.PrintPartial(Rect(w div 15, {левая граница}
              h div 10, {верхняя граница}
              w - (w div 20), {правая граница}
              h - (h div 20) )); {нижняя граница}
   Printer.EndDoc;{конец настоек и собственно печать}
    except
    on Exception do{если возникла какая-то ошибка...}
      Begin
        Printer.Abort;
        Printer.EndDoc;
        Raise {обработка исключения, прерываем про-
цесс печати}
      end
    end; {конец защищенного блока try}
  finally  Printer.Orientation:=OldOrientation; {вос-
станавливаем ориентацию страницы}
    Screen.Cursor:=crDefault {восстанавливаем вид
курсора}
  end
end
end; {конец процедуры печати графика}

procedure TForm1.BitBtn4Click(Sender: TObject);
begin
 If PrintDialog1.Execute
      Then RichEdit1.Print('Печать документа')
end;
```

Аналогичным образом можно работать и с другими системными диалогами.

2.5.5.10. Ввод-вывод данных через внешний файл

При работе с программой часто требуется запомнить последний вариант введенных исходных данных. Это особенно актуально, если необхо-

димо ввести большое количество исходных данных. Сделать это можно, например, используя текстовые файлы.

Для доступа к файлам чаще всего используется специальная файловая переменная. Она связывается с указанным файлом процедурой *AssignFile*. Эта процедура имеет синтаксис:

```
procedure AssignFile(var F: File, S: string);
```

где *F* - файловая переменная (для удобства пользования в курсовой работе следует использовать для ввода данных из файла файловые переменные с именами *Fr*, *Fr1*, *Fr2* и т. д., а для записи данных в файл – переменные *Fw*, *Fw1*, *Fw2* и т. д.); *S* - строка, содержащая имя файла.

Например, оператор

```
AssignFile (F, 'Daten.txt');
```

связывают файловую переменную *F* с файлом *Daten.txt*.

Открытие, например, для ввода исходных данных, существующего файла осуществляется процедурой *Reset*, формат которой следующий:

```
procedure Reset(var Fr: File);
```

Файловая переменная *Fr* перед обращением к этой процедуре должна быть связана с файлом.

Создание и открытие нового файла, например для записи полученных результатов, осуществляется процедурой *Rewrite*, формат которой следующий:

```
procedure Rewrite(var Fw: File);
```

Файловая переменная *Fw* перед обращением к этой процедуре должна быть связана с файлом.

После выполнения различных операций чтения и записи файл должен быть закрыт процедурой *CloseFile*:

```
procedure CloseFile(var F: File);
```

Текстовые файлы состоят из последовательностей символов, разбитых на строки. В *Delphi* предопределен тип *TextFile*, соответствующий текстовому файлу. Таким образом, объявление файловой переменной может иметь вид:

```
var Fr, Fw: TextFile;
```

Запись данных в текстовый файл осуществляется процедурой

```
procedure Write(var F: TextFile;<список выражений>);
```

Выражения могут быть типов *Char*, *Integer*, *Real*, *String*, упакованных строк, *Boolean*. При этом может использоваться форматирование.

Аналогичная процедура *Writeln* отличается от *Write* только тем, что после записи пишет символ перехода на новую строку, т. е. *Writeln* формирует одну строку.

Чтение данных из текстового файла осуществляется последовательно от его начала процедурой

```
procedure Read(var F:TextFile;<список переменных>);
```

Аналогичная процедура *ReadLn* отличается от *Read* только тем, что после чтения переводит текущую позицию в файле на новую строку. Если

в процедуре *ReadLn* не задай список переменных, то она просто пропускает текущую строку и переходит к следующей.

Пример реализации файлового ввода-вывода данных для условия задачи, приведенного в разделе *2.5.1*, представлен ниже.

Прежде всего, следует создать текстовый файл, например, с именем *Daten.txt* и записать в него следующие данные

```
1.6
1
30
0.1
1
17
15
```

Далее, используя событие *FormCreate*, присвоить переменным числовые значения из файла.

```
AssignFile(Fr,'Daten.txt');
  Reset(Fr);
   Readln(Fr,B);
   Readln(Fr,vMin);
   Readln(Fr,vMax);
   Readln(Fr,deltaVmin);
   Readln(Fr,deltaVmax);
   Readln(Fr,n);
   Readln(Fr,m);
CloseFile(Fr);
```

Затем эти значения присваиваются соответствующим компонентам для ввода исходных данных:

```
LabeledEdit1.Text:=FloatToStr(deltaVmin);
LabeledEdit2.Text:=FloatToStr(deltaVmax);
LabeledEdit3.Text:=IntToStr(n);
LabeledEdit4.Text:=FloatToStr(Vmin);
LabeledEdit5.Text:=FloatToStr(Vmax);
LabeledEdit6.Text:=IntToStr(m);
LabeledEdit7.Text:=FloatToStr(B);
```

После выполнения этих операций в компонентах *LabeledEdit* будут находиться значения переменных, введенных из файла. Теперь эти значения можно менять непосредственно на форме, а для того чтобы они «запомнились» в программе нужно на событие *OnClick* кнопки *Расчет* написать программный код записи измененных исходных данных в файл. Следует обратить внимание на то, что нужно указать одинаковое имя для файла ввода и вывода исходных данных:

```
 AssignFile(Fw,'Daten.txt');
  Rewrite(Fw);
WriteLn(Fw,StrToFloat(LabeledEdit7.Text(B)));
WriteLn(Fw,StrToFloat(LabeledEdit4.Text(vMin)));
```

```
WriteLn(Fw,StrToFloat(LabeledEdit5.Text(vMax)));
WriteLn(Fw,StrToFloat(LabeledEdit1.Text(deltaVmin)));
WriteLn(Fw,StrToFloat(LabeledEdit2.Text(deltaVmax)));
WriteLn(Fw,StrToInt(LabeledEdit3.Text(n)));
WriteLn(Fw,StrToInt(LabeledEdit6.Text(m)));
 CloseFile(Fw);
```

Т. к. теперь будет всегда запоминаться последний вариант введенных данных, то при ошибочном вводе данных, они также будут запоминаться (за исключением ситуаций, когда сработают блоки *try..except* или *try..finally*). Поэтому в программе необходимо предусмотреть восстановление данных на значения по умолчанию. Сделать это можно, например, создав кнопку *По умолчанию*, и записать в нее следующий программный код:

```
LabeledEdit1.Text:=FloatToStr(0.1);
LabeledEdit2.Text:=FloatToStr(1);
LabeledEdit3.Text:=IntToStr(17);
LabeledEdit4.Text:=FloatToStr(1);
LabeledEdit5.Text:=FloatToStr(30);
LabeledEdit6.Text:=IntToStr(15);
LabeledEdit7.Text:=FloatToStr(1.6);
```

2.5.5.11. Дополнительные элементы программы

Дополнительные элементы программы не оказывают принципиального влияния на работу программы в целом, но позволяют повысить комфортность работы с программой. Ниже приведено несколько подобных примеров.

Пример 1. Подтверждающий запрос при выходе из программы (рисунок 2.15).

Рисунок 2.15 – Запрос при выходе из программы

```
procedure TFormHaupt.FormCloseQuery(Sender: TObject;
  var CanClose: Boolean);
    begin
   if Application.MessageBox(
     'Действительно завершить работу?',
     'Подтверждение',MB_YESNO)=IDYES
   then CanClose:=True;
   else CanClose:=False
    end;
```

Пример 2. Разделитель между целой частью и дробной

```
DecimalSeparator:='.';
```

Пример 3. Создание «бегущей» строки. Для этого понадобятся компоненты Label  и Timer  . Пример программы.

```
procedure TForm1.Timer1Timer(Sender: TObject);
Const
LengthGoString = 10;
GoString = 'В конце строку желательно повторить,'+
' чтоб получить эффект кольцевого движения! В конце
ст';
Const
i: Integer = 1;
begin
Label1.Caption:=Copy(GoString,i,LengthGoString);
Inc(i);
If Length(GoString)-LengthGoString < i then i:=1
end;
```

Пример 3. Отключение (включение) системного меню формы и кнопок *Minimize*, *Maximize*, и *Close* во время выполнения программы.

```
procedure TForm1.Button1Click(Sender: TObject);
     begin
{Выключение}
Form1.BorderIcons := Form1.BorderIcons -
[biSystemMenu, biMinimize, biMaximize]
     end;

procedure TForm1.Button2Click(Sender: TObject);
     begin
{Включение}
Form1.BorderIcons := Form1.BorderIcons +
 [biSystemMenu, biMinimize, biMaximize]
     end;
```

Пример 4. Автоматическое изменение ширины колонок, в *StringGrid* чтобы вместить самую длинную строчку в колонке.

```
procedure AutoSizeGridColumn(Grid : TStringGrid; col-
umn : integer);
var
     i : integer;
     temp : integer;
     max : integer;
begin
        max := 0;
```

```
        for i := 0 to (Grid.RowCount - 1) do
        begin
                temp :=
Grid.Canvas.TextWidth(grid.cells[column, i]);
                if temp > max then max := temp
        end;
        Grid.ColWidths[column] := Max +
Grid.GridLineWidth + 3
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
     AutoSizeGridColumn(StringGrid1, 1)
{Вызов процедуры автоматического изменения ширины ко-
лонок}
end;
```

Студент может самостоятельно использовать или добавлять, понравившиеся ему дополнительные элементы программы.

2.6. Разработка справочной системы

Каждая программа должна обеспечивать пользователю доступ к справочной системе, содержащей исчерпывающую информацию о программе и о том, как с ней работать.

Справочная система программ, работающих в *Windows*, в том числе и справочная система *Delphi*, представляет собой набор файлов определенной структуры, используя которые программа *Winhelp*, являющаяся составной частью *Windows*, выводит справочную информацию по запросу (требованию) пользователя.

Основным элементом справочной системы являются *hlp*-файлы, в которых находится справочная информация. В простейшем случае справочная система программы может представлять собой один единственный *hlp*-файл.

Создать справочную систему (*HLP*-файл) можно, например, при помощи поставляемой вместе с *Delphi* программы *Microsoft Help Workshop* (файл *hcw.exe*). Исходным "материалом" для создания *hlp*-файла является текст справочной информации, представленный в виде *rtf*-файла.

Процесс создания справочной системы состоит из четырех важных составляющих:

1) подготовка *rtf*-файла;
2) создание и компилирование файлов справочной системы;
3) создание содержания справки;
4) использование справочной системы в программе.

2.6.1.1. Оформление разделов

Как известно, справка обычно разбивается на разделы. В *rtf*-файле каждый раздел должен начинаться заголовком и заканчиваться символом *разрыв страницы* (меню *Вставка>Разрыв...*) Пример приведен на рисунке 16.

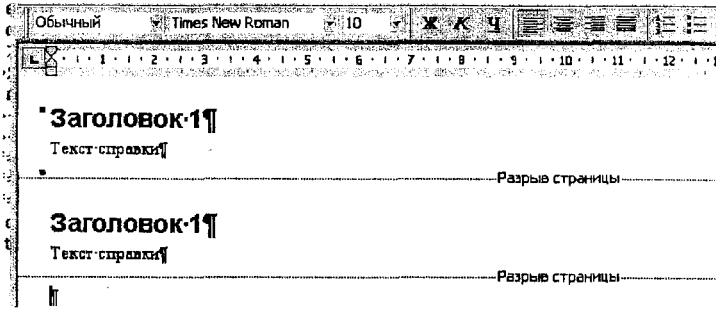


Рисунок 2.16 – Разбивка документа

Кроме этого, раздел должен содержать уникальный идентификатор. Для его установки нужно поместить текстовый курсор перед первым символом заголовка и из меню *Вставка* выбрать пункт *Сноска...* В появившемся диалоговом окне (рисунок 2.17) в разделе *Нумерация* устанавливается радиокнопку в положение *Другая*, а в окно ввести символ диеза ('#'). После нажатия на кнопку *OK MS Word* предложит ввести текст сноски, что и необходимо сделать. Следует заметить, что если текст сноски начинается с префикса *IDH_*, то во время компиляции справочной системы будет проверена корректность всех ссылок данного раздела.

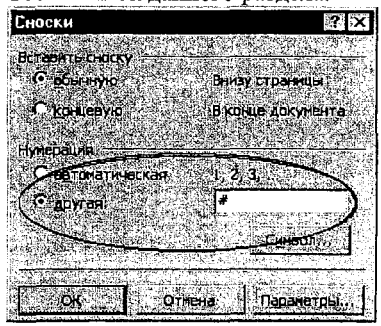


Рисунок 2.17 – Задание сносок

В результате документ будет выглядеть следующим образом (рисунок 2.18):

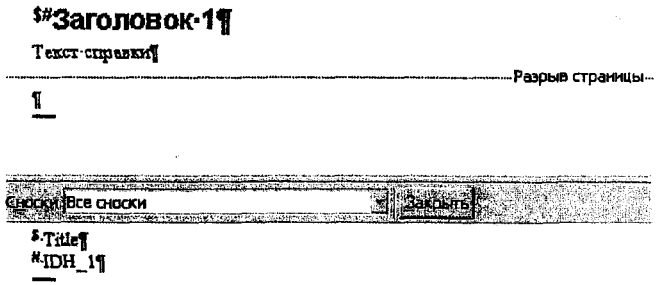


Рисунок 2.18 – Документ MS Word с указанными сносками

Для связывания разных разделов используются слова-ссылки, при нажатии на которые осуществляется переход к нужной ветке справки. Для того, чтобы сделать слово ссылкой, нужно выделить его и подчеркнуть двойной линией (меню *Формат>Шрифт>Подчеркивание>Двойное*). После этого, сразу за словом-ссылкой, требуется поместить без пробела идентификатор нужного раздела (текст сноски). Сама ссылка оформляется скрытым текстом (рисунок 2.19). При запуске справки ссылка будет выделена цветом и одинарным подчеркиванием.

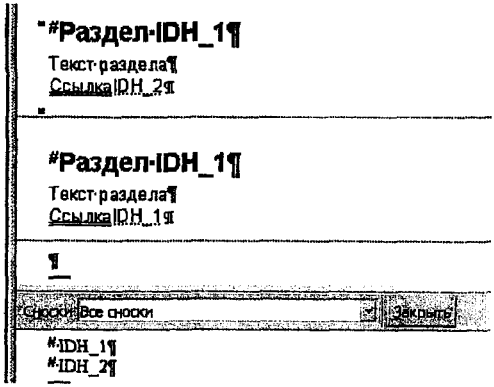


Рисунок 2.19 – Связывание различных разделов

2.6.1.2. Организация поиска по разделам

Для того, чтобы включить возможность поиска по какому-либо разделу, нужно перед его заголовком поставить сноску $ (рисунок 20), текстом которой должно служить название раздела в поисковой системе. Сноска "$". Это то, что отображается в окне *Topics Found*, вкладке *Поиск* и так далее.

Рисунок 2.20 – Задания поиска по справке

2.6.1.3. Оформление списка ключевых слов

Для каждого раздела справки можно создать список ключевых слов. Для этого нужно перед заголовком раздела установить сноску *K* (рисунок 2.21), а в текст сноски - записать все ключевые слова, разделив их точкой с запятой. При работе справочной системы ключевые слова всех разделов будут отображены в закладке *Указатель*.

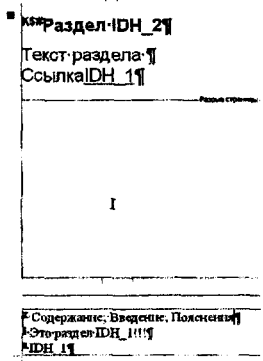


Рисунок 2.21 – Задание ключевых слов

В справке можно использовать не только ссылки на другие разделы, но и ссылки на комментарии (например, для объяснения какого-нибудь термина). Во время работы справочной системы такие ссылки выделяются цветом и подчеркиванием пунктирной линией, при нажатии на них появляется вплывающее окно с текстом комментария. В *RTF*-файле комментарии оформляются так же, как и разделы, но они не должны начинаться с заголовка. Ссылку на комментарий нужно подчеркнуть одной линией и сразу за ней написать его идентификатор скрытым текстом.

2.6.2. Создание файла справочной системы

Созданию справки осуществляется на основе набранного *RTF*-файла. После запуска *Microsoft Help Workshop* (файл *hcw.exe*) необходимо создать новый проект, выбрав пункт меню *File>New>Help Project* и появится диалоговое окно представленное на рисунке 2.22.

Нажав на кнопку *Files...* и в появившемся диалоговом окне при помощи кнопки *Add* необходимо добавить к проекту созданный ранее *rtf*-файл.

Чтобы программа, использующая справочную систему, могла получить доступ к конкретному разделу справочной информации, нужно определить числовые значения для идентификаторов разделов. Чтобы это сделать, надо в окне проекта справочной системы нажать кнопку *Map*, в результате чего откроется диалоговое окно *Map*. В этом окне нужно нажать кнопку *Add* и в поле *Topic ID*, открывшегося диалогового окна *Add Map Entry*, ввести идентификатор раздела справки, а в поле *Mapped numeric value* - соответствующее идентификатору числовое значение. В поле *Comment* можно ввести комментарий - название раздела справочной системы, которому соответствует идентификатор.

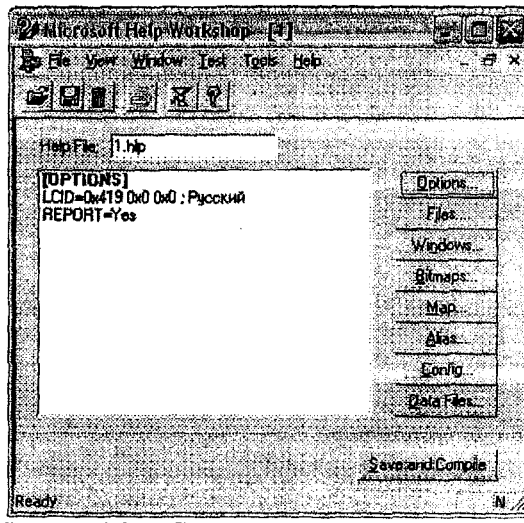


Рисунок 2.22 – Создание нового проекта справки

2.6.3. Создание содержания

Для создания содержания необходимо выбрать меню *File>New>Help Contents* и ввести имя и заголовок содержания. При помощи кнопок *Add Above* (*Добавить над*) и *Add Below* (*Добавить под*) необходимо создать нужные папки (*Heading*) и пункты содержания (*Topic*). При добавлении пункта нужно вводить его название в поле *Title*, в *Topic ID* - идентификатор раздела справки, на который ссылается этот пункт, в *Help File* - имя файла справки, в котором находится этот раздел. Кнопки *Move Right* и *Move Left* служат для изменения иерархии пунктов. После сохранения содержания надо открыть проект справки (**.hpj*), нажать на

кнопку *Options*, активизировать закладку *Files* и в поле *Contents file* ввести имя файла содержания либо выбрать его при помощи кнопки *Browse*.

Последний шаг - компиляция, выполняется нажатием кнопки *Compile*.

В заключении хотелось бы отметить, что существует довольно большое количество программных продуктов, позволяющих существенно автоматизировать, а соответственно и облегчить, процесс создания справочной системы. Порекомендовать можно программу *Crimson Help*, которая выполнена в виде надстройки к редактору *MS Word* и особенно стоит обратить внимание на *Help&Manual* компании *ES Software*, пожалуй, самой мощной и удобной в своем классе.

2.6.4. Использование справочной системы в программе

Для того чтобы во время работы программы пользователь, нажав клавишу *F1*, мог получить справочную информацию, надо чтобы свойство *HelpFile* главного окна приложения содержало имя файла справочной системы, а свойство *HelpContext* числовой идентификатор нужного раздела. Т. е. используются идентификаторы разделов справочной системы перечислены в разделе *MAP* файла проекта справочной системы.

Файл справочной системы приложения лучше поместить в ту папку, в которой находится файл исполняемой программы.

Если в диалоговом окне есть кнопка *Справка*, то для кнопки создается процедура обработки события *OnClick*, которая обращением к функции *winhelp* запускает справочный файл *.hlp*. При вызове функции *Winhelp* в качестве параметров указываются: идентификатор окна, которое запрашивает справочную информацию; имя файла справочной системы; константа, определяющая действие, которое должна выполнить программа *Windows Help* и уточняющий параметр.

Если необходимо вывести конкретный раздел справки, то в качестве параметра, определяющего действие, используется константа *Help_Context*. Уточняющий параметр в этом случае задает раздел справки, который будет выведен на экран.

Пример использования:

```
procedure TFormHaupt.BitBtn1Click(Sender: TObject);
begin
WinHelp(Handle,'NewProject.hlp',Help_context,1)
end;
```

При нажатии на кнопку *Справка* появляется соответствующий раздел справочной системы (рисунок 2.23)

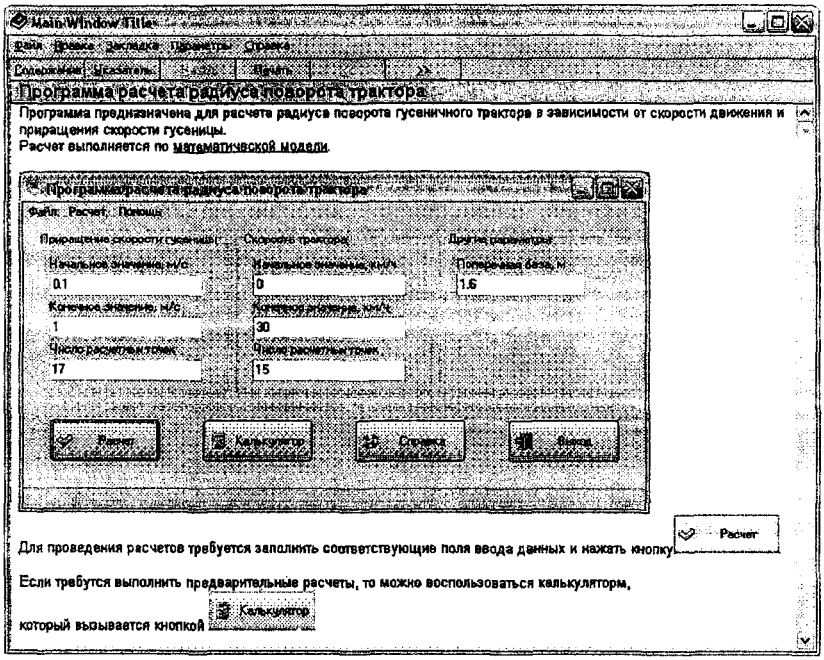


Рисунок 2.23 – Созданный файл справки

В пояснительной записке к курсовой работе необходимо привести скриншоты всех разделов справочной системы.

2.7. Расчет контрольного примера

В данном разделе пояснительной записки к курсовой работе приводятся результаты расчета в соответствии с исходными данными, указанными в бланке задания. Результаты расчета представляются в виде двумерных графических зависимостей первой переменной при нескольких (не менее трех) значениях второй переменной, графических зависимостей второй переменной при нескольких (не менее трех) значениях первой, и трехмерной зависимости от двух рассчитываемых переменных. Также в данном разделе записки должны быть приведены результатов расчета, импортированные в *MS Excel* и *MS Word*.

2.8. Заключение

В заключении даются краткие выводы о проделанной работе (по этапам ее выполнения), приводится краткая характеристика примененных методов и использованного прикладного программного обеспечения. Также в

заключении следует указать возможные дальнейшие пути развития и расширения разработки, возможности расширения круга задач и т. п.

2.9. Список использованных источников информации

Указывается список литературы, использованной студентом при выполнении курсовой работы. При использовании материалов, опубликованных *Internet* , указывается адрес сайта, страницы.

Оформление списка использованных источников информации должно соответствовать ГОСТ 7.1.

На *каждый* использованный источников информации должна быть ссылка в пояснительной записке к курсовой работе с указанием страницы, откуда была взята информация.

Ссылки в тексте пояснительной записки на использованные источники следует приводить в квадратных скобках.

3. ЗАЩИТА КУРСОВОЙ РАБОТЫ

К защите курсовой работы допускаются студенты у которых проверена работоспособность программы, а также проверена и подписана руководителем пояснительная записка к курсовой работе

Доклад по содержанию и полученным результатам курсовой работы производится в виде презентации, созданной средствами *MS PowerPoint* с показом слайдов (минимум 8...10 слайдов). В докладе на слайдах должны быть отражены следующие вопросы:

- Тема и цель исследования.
- Механико-математическая модель исследуемого объекта (допущения, расчетная схема с указанием сил и моментов, уравнения, описывающие движение расчетной схемы или ее элементов).
- Исходные данные.
- Результаты исследований, представленные в виде графических и табличных зависимостей.
- Анализ полученных результатов.
- Заключение по полученным результатам расчетов.

4. ЗАДАНИЯ ДЛЯ ВЫПОЛНЕНИЯ КУРСОВОЙ РАБОТЫ

4.1. Специальность 1-37 01 03 – «Тракторостроение»

1. Рассчитать сопротивление плуга F_s (кН) в зависимости от глубины пахоты h_p (см) и скорости движения трактора v (км/ч).

$$F_s = f_p G + k_p h_p b_p + \varepsilon h_p b_p v^2,$$

где f_p – коэффициент трения рабочих органов плуга о почву; G – вес, Н; k_p – удельное сопротивление на единицу площади поперечного сечения пласта, МПа; b_p – ширина захвата, м; ε - коэффициент увеличения тягового сопротивления при увеличении скорости, Нс^2/м^4.

2. Рассчитать нормальные нагрузки на передний N_1 (кН) и задний N_2 (кН) мосты колесного трактора в зависимости от крюкового усилия F_{kp} (кН) и базы трактора L (м).

$$N_1 = \frac{bG - h_{kp}F_{kp}}{L};\ N_2 = \frac{(L-b)G + h_{kp}F_{kp}}{L},$$

где G – вес трактора, Н; b – расстояние от центра масс трактора до задней оси, м; h_{kp} – высота точки сцепки, м.

3. Рассчитать массу проектируемого трактора m (т) с задними ведущими колесами в зависимости от крюкового усилия F_{kp} (кН) и коэффициента сопротивления движению f

$$m = \frac{\Delta_{lim} F_{kp}}{(\lambda\varphi - f)g},$$

где Δ_{lim} – коэффициент возможной перегрузки; g – ускорение свободного падения, м/с^2; φ – коэффициент сцепления движителя с почвой; λ - коэффициент нагрузки ведущих колес.

4. Рассчитать потребную мощность двигателя P (кВт) проектируемого трактора в зависимости от крюковой нагрузки F_{kp} (кН) и скорости движения v (км/ч).

$$P = \frac{\Delta_{lim} F_{kp} v}{\chi\eta},$$

где Δ_{lim} – коэффициент возможной перегрузки; χ – коэффициент приспособляемости двигателя; η – тяговый КПД трактора.

5. Рассчитать глубину колеи h (см), образуемую тракторным движителем в зависимости от давления движителя p (кПа) и буксования δ (%).

$$h = \frac{\sigma_0}{k} \cdot \frac{1}{1 - p/\sigma_0} \cdot \frac{1+\delta}{1-0{,}5\delta} \operatorname{arcth} \frac{p}{\sigma_0},$$

где σ_0 – предел прочности грунта на одноосное сжатие, Па; k – коэффициент объемного смятия грунта, Н/м^3.

6. Рассчитать тормозной момент на колесах трактора, необходимый для удержания машины на склоне M_t (кН), в зависимости от массы трактора G_t (т) и угла уклона, на котором должен стоять трактор α_{max}.

$$M_t = G_t g r_k \left(\sin\alpha_{\max} - f\cos\alpha_{\max}\right),$$

где r_k – радиус колеса, м; f – коэффициент сопротивления движению; g – ускорение свободного падения, м/с^2.

7. Рассчитать давление на тормозной барабан колодочного тормоза p (Па), в зависимости от радиуса тормозного барабана R (м) и тормозного момента M_t (Нм).

$$p = \frac{M_t}{2\mu R^2 b\beta},$$

где μ – коэффициент трения тормозного элемента; b – ширина тормозной накладки, м; β - угол обхвата одной тормозной колодки.

8. Рассчитать коэффициент рабочих ходов машинотракторного агрегата φ при грушевидном прямом ходе движения, в зависимости от длины гона L (м) и радиуса поворота R (м).

$$\varphi = \frac{L}{L + 6R + 2e},$$

где e – длина выезда, м.

9. Рассчитать силу замыкания ленточных тормозов F_c (кН), в зависимости от момента трения тормоза M (Нм) и угла охвата барабана тормозной лентой α.

$$F_c = \frac{M\left(e^{\mu\alpha}+1\right)a}{\left(e^{\mu\alpha}-1\right)bR},$$

где μ – коэффициент трения ленты по барабану; R – радиус барабана, м; a, b – плечи крепления тормозных лент, м.

10. Рассчитать подачу насоса гидроусилителя рулевого управления Q_c (см^3/об), в зависимости от площади поршня гидромотора усилителя A (см^2) и частоты вращения рулевого колеса n_r (об/мин).

$$Q_c = AS\frac{\pi n_r}{30\gamma\eta_0} - \frac{\Delta Q_c}{\eta_0},$$

где S – полный ход поршня при повороте направляющих колес из одного крайнего положения в другое, м; γ – угол поворота рулевого колеса, соответствующий повороту направляющих колес из одного крайнего положения в другое; η_0 – объемный КПД насоса; ΔQ_c – потери жидкости через золотник, см^3/об.

11. Рассчитать жесткость рессоры c_r (Н/м), в зависимости от периода вертикальных колебаний T_z (с) и веса подрессоренных частей трактора Q_p (кН).

$$c_r = \frac{4\pi^2 Q_p}{n g T_z^2},$$

где n – число рессор с каждой стороны трактора; g – ускорение свободного падения, м/с^2.

12. Рассчитать силу предварительного натяжения гусеницы F_b (кН), в зависимости от расстояния между осями поддерживающих катков l_0 (м) и стрелы прогиба гусеницы между поддерживающими катками f (мм).

$$F_b = \frac{q_g l_0}{8f},$$

где q_g – вес единицы длины гусеницы, Н.

13. Рассчитать требуемый диаметр поддерживающего катка D_p (м), в зависимости от общего количества поддерживающих катков z и базы трактора L (м).

$$D_p = 2M_y \frac{t_g(2+0{,}5z)}{q_g L(\mu - f)},$$

где M_y – момент сопротивления проворачиванию уплотнений, Нм; t_g – шаг гусеницы, м; q_g – вес единицы длины гусеницы, Н; μ - коэффициент трения гусеницы о ролик; f - коэффициент сопротивления вращению катка.

14. Рассчитать звукоизоляцию ограждающей конструкции кабины при отсутствии демпфирования R (дВ), в зависимости от поверхностной массы звукоизолирующего материала m_z (кг/м) и угла между нормалью к ограждению и направлением падения звуковой волны θ_z.

$$R = 10 \cdot \lg\left[1 + \left(\frac{\pi f m_z \cos\theta_z}{\rho_c}\right)^2\right],$$

где f – частота колебаний, Гц; ρ_c – акустическое сопротивление материала ограждения, Па·с.

15. Рассчитать наименьшую ширину поворотной полосы L_p (м) машинотракторного агрегата при беспетлевом дугообразном способе поворота, в зависимости от радиуса поворота R_0 (м) и кинематической ширины агрегата (расстояния от продольной оси симметрии трактора до крайней точки агрегата) d_k (м).

$$L_p = 1{,}1R_0 + 0{,}5d_к + e,$$

где e – длина выезда, м.

16. Рассчитать требуемый диаметр пальца трака гусеницы с открытым металлическим шарниром $d_ш$ (мм), в зависимости от веса трактора G_t (кН) и ширины гусеницы b_g (м).

$$d_ш = \frac{1{,}3\varphi G_t}{q b_g},$$

где φ - коэффициент сцепления гусениц с грунтом; q – среднее давление в проушинах траков, Па.

17. Рассчитать потребную мощность гидропривода гидронавесной системы трактора $N_ш$ (Вт), в зависимости от максимальной массы навесной

машины m_n (кг) и максимального вертикального перемещения центра масс навесной машины H_n (м).

$$N_{tr} = \frac{m_n g H_n k}{\eta t},$$

где g – ускорение свободного падения, м/с^2; k – коэффициент запаса мощности; η - КПД навесной системы; t – время полного подъема навесной машины, определяемое из условий агротехники, с.

18. Рассчитать крутящий момент двигателя трактора M_{tr} (Нм), в зависимости от коэффициента, приспособляемости по моменту k_M и частоты вращения коленчатого вала n (об/мин).

$$M_{tr} = M_N \left[a + b\frac{n}{n_N} + c\left(\frac{n}{n_N}\right)^2 \right],$$

где n_N – частота вращения коленчатого вала двигателя при максимальной мощности, об/мин; M_N - крутящий момент при максимальной мощности (Нм); $a = \frac{k_M k_\omega (2-k_\omega)-1}{k_\omega (2-k_\omega)-1}$; $b = -\frac{2k_\omega (k_M - 1)}{k_\omega (2-k_\omega)-1}$; $c = \frac{k_\omega^2 (1-k_M)}{k_\omega (2-k_\omega)-1}$ вспомогательные коэффициенты; k_ω – коэффициент, приспособляемости по частоте.

19. Рассчитать угловую скорость вала гидромотора объемной гидропередачи механизма поворота ω_{gm} (рад/с), в зависимости от объема гидромотора V_{gm} (см^3) и параметра регулирования насоса ε_n.

$$\omega_{gm} = \frac{V_n \varepsilon_n}{V_{gm}} \eta_{voln} \eta_{volgm} \omega_n,$$

где V_n - рабочий объем насоса, см^3; η_{voln} - объемный КПД насоса; η_{volgm} - объемный КПД гидромотора; ω_n – угловая скорость вала насоса, рад/с.

20. Рассчитать требуемый рабочий объем гидромотора объемной гидропередачи механизма поворота V_{gm} (см^3), в зависимости от момента сопротивления повороту трактора M_s (кНм) и перепада давлений рабочей жидкости в гидропередаче Δp (МПа).

$$V_{gm} = \frac{M_s}{i_{gm} \eta_{volgm} \Delta p \eta_{gmg}},$$

где i_{gm} – передаточное число привода гидромотора; η_{volgm} - объемный КПД гидромотора; η_{gmg} - гидромеханический КПД гидромотора.

21. Рассчитать максимальную мощность, потребляемую объемной гидропередачей механизма поворота N_g (кВт), в зависимости от рабочего объема насоса V_n (см^3) и перепада давлений рабочей жидкости в гидропередаче Δp (МПа).

$$N_g = \frac{V_n \Delta p \omega_d}{\eta_{gmn} i_n \eta_n},$$

где ω_d – угловая скорость коленчатого вала двигателя, рад/с; η_{gmn} – гидромеханический КПД насоса; i_n – передаточное число привода насоса; η_{volm} - объемный КПД насоса.

22. Рассчитать показатель проходимости трактора P в зависимости от его сцепного веса G_s (Н) и крюковой нагрузки F_{kp} (кН).

$$P = \frac{\varphi_{max} G_s}{fG + F_{kp}},$$

где φ_{max} - коэффициент сцепления с почвой; f – коэффициент сопротивления движению; G – вес трактора, Н.

23. Рассчитать удельную мощность трения дисков фрикционного тормоза P_{ud} (Вт/м^2) в зависимости от момента трения тормоза M_{tr} (Нм) и числа пар трения z.

$$P_{ud} = \frac{M_{tr}\left|\omega_r\right|}{\pi\left(r_e^2 - r_i^2\right)z},$$

где ω_r – относительная угловая скорость скольжения дисков, рад/с; r_e и r_i – наружный и внутренний радиусы фрикционных дисков, м.

24. Рассчитать требуемый диаметр опорного катка D_{op} (м), в зависимости от общего количества опорных катков n_k и нагрузки на каток G_{st} (Н).

$$D_{op} = \frac{0{,}35 G_{st} \dfrac{2E_g E_k}{E_g + E_k}}{n_k [\sigma_k] b},$$

где E_g и E_k – модули упругости материалов трака гусеницы и катка, Па; $[\sigma_k]$ – допустимое напряжение, $[\sigma_k]$=200 МПа; b – ширина обода катка, b=0,6…0,8 м.

25. Рассчитать силу сопротивления движению гусеничного трактора за счет смятия грунта движителем и образования колеи F_{spr} (Н) в зависимости от ширины гусеницы b (м) и максимального удельного давления q_{max} (кПа).

$$F_{spr} = \frac{2b\sigma_0^2}{k} \ln\left(\operatorname{ch}\left(\frac{q_{max}}{\sigma_0}\right)\right),$$

где σ_0 – предел прочности грунта на одноосное сжатие, Па; k – коэффициент объемного смятия грунта, Н/м^3.

26. Рассчитать тормозной момент, требуемый для остановки гусеничного трактора на горизонтальной поверхности без тяги на крюке M_t (Нм), в зависимости от эксплуатационной массы трактора m_e (кг) и замедления при торможении ε (м/с^2).

$$M_t = \frac{\varepsilon r_d m_e \eta_{kp} \eta_g}{u_g z},$$

где r_d – динамический радиус колеса, м; η_{kp} – КПД конечной передачи; η_g – КПД гусеничного движителя; u_g – передаточное число от тормоза к ведущему колесу; z – число одновременно работающих тормозов.

27. Рассчитать тормозной момент, развиваемый одним тормозом, необходимый для удержания трактора на склоне $M_{t\alpha}$ (Нм), в зависимости от эксплуатационного веса трактора G_e (кН) и угла склона α.

$$M_{t\alpha} = \frac{r_d G_e}{u_g}(\sin\alpha - f\cos\alpha),$$

где r_d – динамический радиус колеса, м; u_g – передаточное число от тормоза к ведущему колесу; f – коэффициент сопротивления перекатыванию трактора.

28. Рассчитать равнодействующую нормальных сил действующих на рессоры гусеничного трактора N (кН), в зависимости от подрессоренного веса трактора G_p (Н) и нагрузки на крюке F_{kp} (кН).

$$N = G_n\cos\alpha + 2F_{cb}\sin\psi_s + 2F_p\sin\psi_d + F_{kp}\mathrm{tg}\gamma,$$

где α - угол уклона поверхности пути; ψ_s и ψ_d - углы наклона передней и задней ветвей гусеницы; F_{cb} - натяжение в свободной ветви гусеницы, Н; F_p - усилие в рабочей ветви гусеницы, Н; γ - угол отклонения крюковой нагрузки в вертикальной плоскости.

29. Рассчитать толщину стенок трубопровода гидросистемы трактора δ_t (мм), в зависимости от максимального давления насоса p_{max} (МПа) и средней скорости течения жидкости по трубопроводу v_{cp} (м/с).

$$\delta_t = \frac{p_{max}}{[\sigma_p]}\sqrt{\frac{Q_n}{\pi v_{cp}}},$$

где Q_n – подачи насоса, л/мин; $[\sigma_p]$ – допускаемое напряжение на разрыв, для стальных бесшовных труб $[\sigma_p]$=50…60 МПа, для латунных труб $[\sigma_p]$=25 МПа.

4.2. Специальность I-37 01 04 – «Многоцелевые гусеничные и колесные машины»

1. Рассчитать нормальные напряжения в грунте σ (МПа) в зависимости от его деформации h (мм) и ширины колеса b (м) по формуле М.Г. Беккера

$$\sigma = \left(\frac{k_c}{b} + k_\varphi\right)h^n,$$

где k_c – коэффициент сцепления грунта, Н/м$^{1+n}$; k_φ – коэффициент трения грунта, Н/м$^{2+n}$; n – число проходов.

2. Рассчитать нормальные напряжения в грунте σ (МПа) в зависимости от его деформации h (мм) и коэффициента объемного смятия грунта k (Н/м^3) по формуле В.В. Кацыгина

$$\sigma = \sigma_0\,\mathrm{th}\left(\frac{k}{\sigma_0}h\right),$$

где σ_0 – предел прочности грунта на одноосное сжатие, Па.

3. Рассчитать радиальную деформацию шины h_{sh} (мм) в зависимости от нормальной нагрузки G_k (кН) и давления воздуха в шине p_w (МПа)

$$h_{sh} = \frac{G_k}{2\pi p_w \sqrt{r_0 r_c}},$$

где r_0 – свободный радиус колеса, м; r_c – радиус сечения шины, м.

4. Рассчитать глубину колеи h (мм) в зависимости от нормальной нагрузки G_k (кН) на колесо и ширины колеса b (м)

$$h = \sqrt[3]{\frac{G_k^3}{k^2 b^2 D_c}},$$

где D_c – свободный диаметр колеса, м; k – коэффициент объемного смятия грунта, Н/м^3.

5. Рассчитать угол поворота наружного управляемого колеса машины α_e в зависимости от угла поворота внутреннего управляемого колеса α_i и расстоянием между осями шкворней B (м).

$$\alpha_e = \arcsin\frac{l_p \sin(\theta - \alpha_i)}{B^2 + l_p^2 - 2Bl_p \sin(\theta - \alpha_i)} + \\ + \arccos\frac{l_p(1 - 2\cos^2\theta) + B[\cos\theta - \cos(\theta - \alpha_i)]}{\sqrt{B^2 + l_p^2 - 2Bl_p \cos(\theta - \alpha_i)}} - \theta,$$

где θ – угол установки рулевых рычагов; l_p – длина рулевых рычагов, м.

6. Рассчитать момент трения многодисковой фрикционной муфты (тормоза) M_{tr} (Нм) в зависимости от числа пар трения z и усилия сжатия дисков F_{sg} (Н).

$$M_{tr} = \mu F_{sg} r_e z,$$

где μ – коэффициент трения фрикционных дисков; r_e – радиус действия силы трения, эквивалентной действию всех элементарных сил трения на площади контакта фрикционной пары, м.

7. Рассчитать динамический радиус колеса r_d (м) в зависимости от давления воздуха в шине p_w (МПа) и нормальной нагрузки на колесо G_k (кН).

$$r_d = r_0 - \frac{G_k}{2\pi p_w \sqrt{r_0 r_c}},$$

где p_w - давление воздуха в шине, МПа; r_0 – свободный радиус колеса, м; r_c – радиус сечения шины, м.

9. Рассчитать радиус поворота машины с «ломающейся» рамой R (м) в зависимости от угла «складывания» α полурам и длины базы машины L (м).

$$R = \frac{\sqrt{l_1^2 + l_2^2 + l_1 l_2 \cos\alpha}}{tg\left[\arcsin\left(\frac{l_2}{L}\sin\alpha\right)\right] + tg\left[\arcsin\left(\frac{l_1}{L}\sin\alpha\right)\right]},$$

где l_1, l_2 – длина первой и второй полурам машины, м.

10. Рассчитать момент трения M_f (Нм), развиваемый во фрикционном сцеплении при его включении в зависимости от текущего времени t (с) и номинального момента двигателя $M_{пот}$ (Нм).

$$M_f = \beta M_{пот}\left(1 - e^{-kt}\right),$$

где β – коэффициент запаса сцепления; k – коэффициент, характеризующий скорость нарастания момента трения.

11. Рассчитать число пар трения z фрикционного сцепления в зависимости от допустимого давления на накладки $[p]$ (Па) и момента трения M_f (Нм).

$$z = \frac{M_f}{\frac{2}{3}\pi\mu[p]\left(R_e^3 - R_i^3\right)},$$

где μ – коэффициент трения; R_e – наружный радиус фрикциона, м; R_i – внутренний радиус фрикциона, м.

12. Рассчитать поправочный коэффициент на касательную силу тяги колеса β_F в зависимости от силы тяги колеса F_k (кН) и коэффициента сцепления колеса с грунтом φ.

$$\beta_F = \begin{cases} \dfrac{\sqrt{1-\left(\dfrac{F_k}{\varphi G_k}\right)^2}}{1+0{,}375\left(\dfrac{F_k}{\varphi G_k}\right)} & \text{при } \sqrt{F_k^2 + F_b^2} \le 0{,}5\varphi G_k \\ \sqrt{1-\left(\dfrac{F_k}{\varphi G_k}\right)^2} & \text{при } \sqrt{F_k^2 + F_b^2} > 0{,}5\varphi G_k, \end{cases}$$

где G_k – вес, приходящийся на колесо, Н; F_b – боковая сила, действующая на колесо, Н.

13. Рассчитать угол закручивания торсионного вала θ в градусах в зависимости от рабочей длины вала l_p (м) и закручивающего момента M_θ (Нм).

$$\theta = \frac{M_\theta l_p}{G_p I_p},$$

где G_p – модуль упругости при кручении, Н/м; $I_p = \dfrac{\pi d^4}{32}$ - полярный момент инерции сечения вала, м^4; d – диаметр вала, м.

14. Рассчитать резонансную частоту колебаний ограждающей конструкции кабины f_{kr} (Гц), в зависимости от толщины ограждений h_z (мм) и плотности материала ограждения ρ (кг/м^3).

$$f_{kr} = \frac{c^2}{2\pi h}\sqrt{\frac{12\rho(1-\mu)^2}{E}},$$

где c – скорость звука в воздухе, м/с; E – модуль упругости материала ограждения, Н/м; μ – коэффициент Пуассона материала ограждения, Н/м^2.

15. Рассчитать окружную силу, возникающую в крестовине сателлита дифференциала F_{kr} (кН), в зависимости от момента, ограничиваемого сцеплением движителя с опорной поверхностью M_φ (кНм) и числа сателлитов n_s.

$$F_{kr} = \frac{M_\varphi}{n_s r_k},$$

где r_k – средний радиус действия окружной силы на крестовине, м.

16. Рассчитать предполагаемый пробег гусеницы с резинометаллическим шарниром до ее разрушения S_g (км), в зависимости от касательного напряжения, возникающего в резиновой втулке τ_{max} (МПа) и шага шарниров (траков) гусеницы t_g, м.

$$S_g = \frac{2\left[\dfrac{196}{\left(10\tau_{max}-3\right)^2}-2\right]10^5 z t_g}{m},$$

где z – число траков на одной гусенице; m – число точек перегиба на гусеничном обводе.

17. Рассчитать диаметр торсионного вала, обеспечивающего заданную жесткость подвески в статическом положении d_t (мм), в зависимости от длины балансира L_b (м) и массы подрессоренной части остова машины m_p (кг).

$$d_t = \sqrt[4]{\frac{L_b L_t \cos\beta}{0{,}1G}\left(mL_b\cos\beta + m_p g \operatorname{tg}\beta\right)},$$

где L_t – длина торсиона, м; β - угол наклона балансира в статическом положении; m – модуль жесткости торсиона, Н/м; g – ускорение свободного падения, м/с^2; G – модуль упругости второго рода, Н/м.

18. Рассчитать напряжение кручения буферной пружины конического типа с витками прямоугольного сечения τ_b (МПа), в зависимости от нагрузки на пружину Q (Н) и шага пружины a (м).

$$\tau_b = \frac{(R+r)Q}{\eta_1 a b^2},$$

где R, r – радиусы нижнего и верхнего оснований конической пружины, м; η_1 – вспомогательный коэффициент, η_1=0,22...0,31; b – толщина витка пружины, м.

19. Рассчитать жесткость буферной пружины конического типа с витками прямоугольного сечения c_b (Н/м), в зависимости от числа рабочих витков пружины n и толщина витка пружины b (м).

$$c_b = \frac{2\eta a b^3 G}{\pi n (R+r)\left(R^2+r^2\right)},$$

где R, r – радиусы нижнего и верхнего оснований конической пружины, м; η – вспомогательный коэффициент, η=0,05…0,31; a – шаг пружины, м; G – модуль упругости второго рода, Н/м.

20. Рассчитать деформацию буферной пружины конического типа с витками прямоугольного сечения c_b (Н/м), в зависимости от числа рабочих витков пружины n и нагрузки на пружину Q (Н).

$$h_b = \frac{\pi n(R+r)\left(R^2+r^2\right)Q}{2\eta a b^3 G},$$

где R, r – радиусы нижнего и верхнего оснований конической пружины, м; η – вспомогательный коэффициент, η=0,05…0,31; a – шаг пружины, м; G – модуль упругости второго рода, Н/м; b – толщина витка пружины, м.

21. Рассчитать проводимость отверстий резонатора K_r, в зависимости от числа отверстий n и площади отверстия S (м^2).

$$K_r = \frac{nS}{l_h + 0{,}8\sqrt{S}},$$

где l_h – длина отверстия (толщина перфорированной панели), м.

22. Рассчитать удельную работу трения синхронизатора W_s (Дж/м^2), в зависимости от момента трения синхронизатора M_s (Нм) и начальной относительной скорости включаемой синхронизатором элементов ω_{s0} (рад/с).

$$W_s = \frac{0{,}5 M_s \omega_{s0} t_s}{2\pi r_s b_s},$$

где t_s – время включения синхронизатора, с; r_s - радиус фрикционного кольца по образующей конуса, м; b_s – ширина фрикционного кольца по образующей конуса, м.

23. Рассчитать момент, развиваемый насосом гидротрансформатора M_g (кН), в зависимости от частоты вращения насосного колеса n_g (об/мин) и активного диаметра D (м).

$$M_g = \rho \lambda n_g^2 D^5,$$

где ρ – плотность рабочей жидкости, кг/м^3; λ – коэффициент момента.

24. Рассчитать жесткость пружины подвески машины c_p (Н/м), в зависимости от среднего диаметра пружины D_{sr} (м) и числа рабочих витков пружины n.

$$c_p = \frac{Gd^4}{8D_{sr}^3 n},$$

где d –диаметр сечения витка пружины, м; G – модуль упругости второго рода, Н/м.

25. Рассчитать касательную силу тяги колеса F_k (кН) в зависимости от его буксования δ (%) и нормальной нагрузки на колесо G_k (Н).

$$F_k = \varphi_{max} G_k \left(1 - e^{-k\delta}\right),$$

где φ_{max} - коэффициент сцепления с почвой; k – коэффициент кривой буксования.

26. Рассчитать изгибающий момент M_{iz} (Нм), действующий на трак гусеничной ленты при движении по грунту в зависимости от максимальной нагрузки на каток Q (кН) и ширины гусеничной ленты b_g (м).

$$M_{iz} = \frac{0{,}25Q}{\sqrt[4]{\frac{kb_g}{4EJ}}},$$

где k – податливость грунта, k=0,05…0,15 МПа; E – модуль упругости материала трака, Н/м; J – момент инерции сечения трака, м^4.

27. Рассчитать площадь поверхности теплоотдачи бака A_{tb} (м^2) из условия поддержания рабочего температурного режима масла, в зависимости от максимальной рабочей температуры масла t_{max} (^{0}C) и подачи насоса Q_n (л/мин).

$$A_{tb} = 14\frac{p_n \cdot 6 \cdot 10^4 Q_n}{\xi\left(t_{max} - t_a\right)},$$

где p_n – противодавление в напорной магистрали при холостой работе насоса, p_n=0,3…0,4 МПа; ξ – коэффициент теплоотдачи, для окрашенных баков из листовой стали ξ=9…10,5 Дж/(м$^{2.0}$С·с), для чугунных баков ξ=5,5…7 Дж/(м$^{2.0}$С·с); t_a – температура воздуха, t_a=30…40^{0}С.

28. Рассчитать период вертикальных колебаний подрессоренной части машины T_z (с), в зависимости от веса подрессоренных частей машины G_p (Н) и числа рессор с каждой стороны машины n.

$$T_z = 2\pi\sqrt{\frac{G_p}{2nc_p g}},$$

где c_p – жесткость рессоры, Н/м; g – ускорение свободного падения, м/с^2.

29. Рассчитать давление гусеничного движителя на грунт q (кПа), в зависимости от массы машины M (т) и длины опорной поверхности L (м).

$$q = \frac{Mg}{2bL},$$

где b – ширина гусеницы, м; g – ускорение свободного падения, м/с^2.

4.3. Специальность I-37 01 05 – «Городской электрический транспорт»

1. Рассчитать максимальную крутизну подъема пути i (‰), который может преодолеть трамвайный поезд, в зависимости от коэффициента сцепления колес с рельсом φ и массы состава m_{sost} (т).

$$i = 1000\frac{\varphi}{1 + m_{sost}/m_t},$$

где m_t – масса тягового вагона, т.

2. Рассчитать величину электродвижущей силы двигателя E (В) в зависимости от угловой скорости якоря ω_r (рад/с) и магнитного потока главного полюса двигателя F_m (Вб).

$$E = \frac{p\omega_r N F_m}{2\pi a_p},$$

где p – число пар полюсов; N – число проводников обмотки якоря; a_p – число параллельных ветвей обмотки якоря.

3. Рассчитать коэффициент мощности выпрямителя ζ_p (Вт) в зависимости от реактивного сопротивления X (Ом) и эффективного значения электродвижущей силы E (В).

$$\zeta_p = \frac{2\sqrt{2}}{\pi k} - \frac{2\xi I_d X k_0}{\pi E k},$$

где k – отношение эффективного тока вторичной обмотки трансформатора к среднему выпрямленному току; ξ – коэффициент, учитывающий влияние омического падения напряжения в цепи переменного тока на среднее выпрямленное напряжение; I_d – величина выпрямленного тока (А); k_0 – отношение коэффициента трансформации на данной ступени регулирования к наименьшему коэффициенту трансформации, при котором получается номинальная электродвижущая сила холостого хода вторичной обмотки трансформатора.

4. Рассчитать собственное сопротивление R_{so} (Ом) якорной цепи тягового электродвигателя троллейбуса в зависимости от номинального тока I_{nom} (А) и КПД привода η_{nom}.

$$R_{so} = 0{,}5\frac{U_{nom}}{I_{nom}}(1-\eta_{nom}),$$

где U_{nom} –номинальное напряжение в сети, В.

5. Рассчитать собственную индуктивность L_{so} (Гн) якорной цепи тягового электродвигателя троллейбуса в зависимости от номинального тока I_{nom} (А) и номинальной частоты вращения ротора электродвигателя ω_{nom} (об/мин).

$$L_{so} = k_L \frac{U_{nom}}{p_n \omega_{nom} I_{nom}},$$

где U_{nom} –номинальное напряжение в сети, В; k_L – коэффициент бескомпенсационной обмотки; p_n – число пар полюсов.

6. Рассчитать скорость скольжения элементов шины троллейбуса в пятне контакта относительно дороги Δv (м/с) в зависимости от буксования δ (%) и деформации шины h_{sh} (м).

$$\Delta v = \left[\frac{(r_c - h_{sh})^2 + x^2}{(1-\delta)(r_c - h_{sh})^2} - 1\right] v_k,$$

где r_c – радиус сечения шины, м; x – поперечная координата, м; v_k – скорость качения колеса, м/с.

7. Рассчитать величину коэффициента сопротивления уводу шины k_y в зависимости от нормальной нагрузки на колесо G_k (Н) и давления воздуха в шине $p_в$ (МПа).

$$k_y = \begin{cases} c\left[1{,}7\dfrac{h}{D_c} - 12{,}7\left(\dfrac{h}{D_c}\right)^2\right] p_w b^2 & \text{при} \dfrac{h}{D_c} \le 0{,}088; \\ c\left[1{,}7\dfrac{h}{D_c} - 12{,}7\left(\dfrac{h}{D_c}\right)^2\right] p_w b^2 & \text{при} \dfrac{h}{D_c} > 0{,}088, \end{cases}$$

где $\dfrac{h}{D_c} = 0{,}42\dfrac{G_k}{p_w D_c^2}\sqrt{\dfrac{D_c}{b}}$; D_c – свободный диаметр колеса, м; h – деформация шины, м; b – ширина шины, м; c – безразмерный коэффициент, зависящий от конструкции шины; для шин обычной конструкции c=100.

8. Рассчитать угловую скорость вращения якоря тягового электродвигателя троллейбуса ω_{dv} (рад/с) в зависимости от скорости движения v (м/с) и числа перевозимых пассажиров n_{pas}.

$$\omega_{dv} = \frac{v u_{tr}}{0{,}96\left(r_c - \dfrac{n_{pas} m_{pas} g}{2\pi p_w n_k \sqrt{r_0 r_s}}\right)},$$

где r_0 – свободный радиус колеса, м; r_s – радиус сечения шины, м; p_w - давления воздуха в шине, МПа; u_{tr} – передаточное число трансмиссии; n_k - количество колес троллейбуса; m_{pas} – масса одного пассажира, кг; g – ускорение свободного падения, м/с^2.

9. Рассчитать реактивную проводимость трансформатора B_t (См) в зависимости от тока холостого хода $I_{x\%}$ (А) и номинальной мощности трансформатора S_{nom} (Вт).

$$B_t = \frac{I_{x\%} S_{nom}}{100 U_{nom}^2},$$

где U_{nom} - номинальное напряжение обмотки трансформатора, В.

10. Рассчитать значение пускового ускорения трамвайного поезда a (м/с^2) в зависимости от полной массы поезда m_s (кг) и коэффициента сцепления φ колес с рельсами

$$a = \frac{1000 m_s g \varphi - w_0}{1000 m_s (1 + \delta_{vr})},$$

где w_0 – удельное сопротивление движению, Н/кН; g – ускорение свободного падения, м/с^2; δ_{vr} –коэффициент.

11. Рассчитать полную массу прицепных вагонов трамвайного поезда m_t (кг) по условиям сцепления колес моторного вагона с рельсами в зависимости от его сцепной массы m_s (кг) и массы вагонов m_v (кг).

$$m_t = \frac{1000\varphi m_s g - m_v (w_0' + i) g}{(w_0'' + i) g},$$

где w_0' – основное удельное сопротивление движению моторного вагона, Н/кН; w_0'' – основное удельное сопротивление движению прицепных ваго-

нов, Н/кН; g – ускорение свободного падения, м/с^2; i – крутизна подъема, ‰; φ - коэффициент сцепления колеса с рельсом.

12. Рассчитать максимальный момент сопротивления повороту колеса на месте M_{cmax} (Нм) в зависимости от нагрузки на колесо G_k (Н) и деформации шины h_{sh} (м).

$M_{cmax} = 0,375\varphi G_k\sqrt{l_k b_k}$, где $l_k = 1,5\sqrt{(2r_c - h_{sh})h_{sh}}$, $b_k = 1,5\sqrt{(2b - h_{sh})h_{sh}}$,

где r_c – радиус сечения шины, м; b – ширина шины, м; φ - коэффициент сцепления колеса с дорогой.

13. Рассчитать величину коэффициента сцепления колеса с рельсом φ в зависимости от скорости движения v (м/с) и коэффициента a_c, зависящего от рода поставленной в расчетах задачи.

$$\varphi = \frac{\eta_c}{a_c + 0,04v},$$

где η_c – коэффициент, учитывающий неравенство нагрузок между ведущими колесными парами тележки, η_c=0,92...0,96; a_c – коэффициент, равный

- при определении максимальной нагрузки тягового электродвигателя a_c=2,3;
- при трогании или экстренном торможении a_c=3,7;
- для расчетов связанных с безопасностью a_c=6.

14. Рассчитать основное удельное сопротивление трамвайного вагона w_0 (Н/кН) в зависимости от скорости движения трамвая v (м/с) и массы трамвая m (т) при разных режимах движения

езда под током $w_0 = 2,5 + (30 + 0,04v^2)m$;

езда без тока $w_0 = 3,0 + (40 + 0,05v^2)m$.

15. Рассчитать угол поворота наружного управляемого колеса троллейбуса α_e в зависимости от угла поворота внутреннего управляемого колеса α_i и базы троллейбуса L (м), при идеальной рулевой трапеции.

$$\operatorname{ctg}\alpha_e - \operatorname{ctg}\alpha_i = \frac{B}{L},$$

где B – расстояние между осями шкворней, м.

16. Рассчитать критическую скорость троллейбуса v_{kp} (км/ч) в зависимости от его массы m (т) и базы L (м).

$$v_{kp} = L\sqrt{\frac{k_{\psi 1}k_{\psi 2}}{m\left[ak_{\psi 1} - (L-a)k_{\psi 2}\right]}},$$

где a – расстояние от центра масс троллейбуса до передней оси, м; $k_{\psi 1}$, $k_{\psi 2}$ – коэффициенты сопротивления уводу колес переднего и заднего мостов, Н/рад.

17. Рассчитать потребную мощность тягового электродвигателя троллейбуса P_{dv} (кВт) в зависимости от скорости движения v (км/ч) и количества перевозимых пассажиров n.

$$P_{dv} = \frac{v}{\eta_{tr} z_{dv}} \left[\varphi \left(m_s + n m_{pas} \right) g + k_w A_w v^2 \right],$$

где m_{pas} – масса одного пассажира, кг; g – ускорение свободного падения, м/с^2; η_{tr} – КПД трансмиссии троллейбуса; z_{dv} – число тяговых электродвигателей; m_s - масса состава, кг; k_w – коэффициент, учитывающий сопротивление воздуха; A_w – площадь лобового сопротивления трамвая, м^2; φ - коэффициент сцепления колеса с дорогой.

18. Рассчитать нормальные нагрузки на переднюю R_1 (кН) и заднюю R_2 (кН) тележки трамвая при разгоне и торможении в зависимости от количества перевозимых пассажиров n_{pas} ускорения (замедления) трамвая j (Н/с^2).

$$R_1 = \frac{\left(m_s + m_{pas} n_{pas} \right) \left[b g \cos\alpha + h \left(\mp j - g \sin\alpha \right) \right]}{L};$$

$$R_2 = \frac{\left(m_s + m_{pas} n_{pas} \right) \left[g \left(L - b \right) \cos\alpha + h \left(\pm j + g \sin\alpha \right) \right]}{L},$$

где b – расстояние от центра масс трамвая до задней оси, м; m_s – снаряженная масса трамвая, кг; m_{pas} – масса пассажира, кг; g – ускорение свободного падения, м/с^2; α - угол подъема пути; h – высота расположения центра масс трамвая, м; L – база трамвая, м.

19. Рассчитать активное сопротивление двухобмоточного трансформатора R_t (Ом) в зависимости от номинальной мощности трансформатора S_{nom} (Вт) и потерь короткого замыкания трансформатора ΔP_k (Вт).

$$R_t = \frac{\Delta P_k U_{nom}^2}{S_{nom}^2},$$

где U_{nom} - номинальное напряжение обмотки трансформатора, В.

20. Рассчитать реактивное сопротивление двухобмоточного трансформатора X_t (Ом) в зависимости от напряжения короткого замыкания трансформатора $U_{k\%}$ (В) и номинальной мощности трансформатора S_{nom} (Вт).

$$X_t = \frac{U_{k\%} U_{nom}^2}{S_{nom} 100},$$

где U_{nom} - номинальное напряжение обмотки трансформатора, В.

21. Рассчитать удельное активное сопротивление проводника фазы линии r_θ (Ом/км) в зависимости от температуры окружающей среды θ (^{0}С) и удельного активного сопротивление линии при температуре окружающей среды 20^{0}С r_0 (Ом/км).

$$r_\theta = r_0(1+0{,}004(\theta-20)).$$

22. Рассчитать зарядную мощность линии Q_b (Вт) в зависимости от напряжения в линии U (В) и длины линии l (м).

$$Q_b = U^2 b_0 L,$$

где b_0 - удельная реактивная проводимость линии, См.

23. Рассчитать потери электроэнергии по методу времени наибольших потерь ΔW_n (Вт) в зависимости от времени τ (ч) и максимальной мощности источника S_n (Вт)

$$\Delta W_n = \frac{S_n^2}{U^2} R\tau,$$

где U - напряжение в сети, В; R – сопротивление линии, Ом.

24. Рассчитать удельную реактивную проводимость линии b_0 (См) в зависимости от частоты f (Гц) и удельной рабочей емкости C_0 (Ф)

$$b_0 = 2\pi f C_0.$$

25. Рассчитать нагрузочные потери реактивной мощности в трансформаторе ΔQ_{HT} (Вт) в зависимости от напряжения короткого замыкания трансформатора $U_{k\%}$ (В) и мощности трансформатора S (Вт);

$$\Delta Q_{HT} = \frac{U_{k\%}}{100} \cdot \frac{S^2}{S_{nom}},$$

где S_{nom} – номинальная мощности трансформатора, Вт.

26. Рассчитать нагрузочные потери активной мощности в трансформаторе ΔP_{HT} (Вт) в зависимости от потерь мощности на корону ΔP_k (Вт) и мощности трансформатора S (Вт);

$$\Delta P_{HT} = \Delta P_k \left(\frac{S}{S_{nom}} \right)^2,$$

где S_{nom} – номинальная мощности трансформатора, Вт.

27. Рассчитать удельную активную проводимость линии g_0 (См) в зависимости от напряжения в сети U (В) и удельных потерь мощности на корону ΔP_{k0} (Вт/м).

$$g_0 = \frac{\Delta P_{k0}^{\ 2}}{U^2}.$$

28. Рассчитать активную проводимость трансформатора G_t (Вт) в зависимости от потерь активной мощности холостого хода ΔP_x (Вт) и номинальной напряжения трансформатора U_{nom} (В).

$$G_t = \frac{\Delta P_x}{U_{nom}^2}.$$

29. Рассчитать потери мощности на корону ΔP_k (Вт) в линиях электропередач в зависимости от длины провода L (м) и удельных потерь мощности на корону ΔP_{k0} (Вт/м).

$$\Delta P_k = \Delta P_{k0} L.$$

Литература

Основная

1. Архангельский,А.Я. Программирование в Delphi 7. - М.: ООО «Бином-Пресс», 2003 г. — 1152 с.

2. Кэнту, М. Delphi 7 для профессионалов. – СПб.: Питер, 2004. – 1101 с.

3. Сухарев,М.В. Основы Delphi. Профессиональный подход. - СПб.: Наука и Техника, 2004. — 600 с.

4. Бакнелл Джулиан М. Фундаментальные алгоритмы и структуры данных в Delphi: Пер. с англ./Джулиан М. Бакнелл. - СПб: ООО «ДиаСофтЮП», 2003.- 560 с.

5. Фленов,М. Е. Библия Delphi. - СПб.: БХВ-Петербург, 2004. - 880 с.

6. Стивене,Р.Delphi. Готовые алгоритмы / Род Стивене; Пер. с англ. Мерещука П. А. - 2-е изд., стер. - М.: ДМК Пресс; СПб.: Питер, 2004. - 384 с.

7. Корняков,В. Программирование документов и приложений MS Office в Delphi. - СПб.: БХВ-Петербург, 2005. - 496 с.

Дополнительная

8. Тракторы: Теория: Учебник / Под общ. ред. В.В. Гуськова. – М.: Машиностроение, 1988. – 376 с.

9. Барский,И.Б. Конструирование и расчет тракторов. – М.: Машиностроение, 1980. – 335 с.

10. Тракторы. Проектирование, конструирование и расчет / Под общ. ред. И.П. Ксеневича. – М.: Машиностроение, 1991. – 544 с.

11. Автомобили. Конструкция, конструирование и расчет. Системы управления и ходовая часть / Под ред. А.И. Гришкевича. – Мн.:Выш. шк., 1987. – 200 с.

12. Автомобили: Конструкция, конструирование и расчет. Трансмиссия / Под ред. А.И. Гришкевича. – Мн: Выш. шк., 1985. – 240 с.

13. Теория электрической тяги / В.Е. Розенфельд, И.П. Исаев, Н.Н. Сидоров, М.И. Озеров; Под ред. И.П. Исаева. – М.: Транспорт, 1995. – 294 с.

14. Фираго,Б.Н. Теория электропривода. Учеб. пособие / Б.И. Фираго, Л.Б. Павлячик. – Мн.: ЗАО «Техноперспектива», 2004. – 527 с.

15. Федин,В.Т. Электрические системы и сети. Терминология и задачи для решения: Метод. пособие к практическим занятиям по дисц. «Электрические системы и сети» и «Установившиеся режимы электрических систем и сетей» для студ. электроэнергетических спец. вузов. — Мн.: БНТУ, 2004. – 96 с.

ПРИЛОЖЕНИЕ 1

Образец титульного листа

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Автотракторный факультет

Кафедра «Тракторы»

Группа 101217

КУРСОВАЯ РАБОТА

по дисциплине

«ИНФОРМАТИКА»

РАССЧИТАТЬ РАДИУС ПОВОРОТА ГУСЕНИЧНОГО ТРАКТОРА В ЗАВИСИМОСТИ ОТ СКОРОСТИ ДВИЖЕНИЯ И ПРИРАЩЕНИЯ СКОРОСТИ ГУСЕНИЦЫ

Выполнил
студент Т. Р. Ивановский

Проверил В.В. Равино

Минск 2007

ПРИЛОЖЕНИЕ 2
Образец оформления реферата

РЕФЕРАТ

25 стр., 7 рис., 6 табл., 8 источн., 1 прил.

РАДИУС ПОВОРОТА, СКОРОСТЬ ДВИЖЕНИЯ, ПРИРАЩЕНИЯ СКОРОСТИ ГУСЕНИЦЫ, БЛОК-СХЕМА, DELPHI, СПРАВКА

ПРИЛОЖЕНИЕ 3

Таблица П.1 – Значения коэффициентов аппроксимирующей экспоненты φ_{max} и k для различных почвенно-дорожных условий

Тип трактора	Почвенный фон	φ_{max}	k
Колесные 4Х2			
	Сухой бетон	0,76	23,89
	Стерня зерновых на суглинистом черноземе	0,70	8,48
	Стерня зерновых на супеси	0,60	7,69
	Поле, подготовленное под посев, на суглинистом черноземе и супеси	0,55	7,01
Колесные 4Х4 пропашные			
с разными колесами	Стерня зерновых на суглинистом черноземе	0,60	6,50
с одинаковыми колесами	Поле, подготовленное под посев, на суглинистом черноземе	0,55	5,58
Колесные 4Х4 пахотные			
с разными колесами	Стерня зерновых на суглинистом черноземе	0,67	6,87
с одинаковыми колесами	Поле, подготовленное под посев, на суглинистом черноземе	0,60	6,93
Гусеничные			
	Стерня зерновых на тяжелом суглинистом черноземе	0,75	69,77
	Стерня зерновых на среднесуглинистом черноземе	0,67	44,78
	Поле подготовленное на стерне и тяжелосуглинистом черноземе	0,62	27,99

ПРИЛОЖЕНИЕ 4

Таблица П.2 – Значения коэффициентов аппроксимирующей экспоненты φ_{max} и k для трамвайного колеса и различного состояния рельса

Состояние рельса	φ_{max}	k
Сухой чистый обезжиренный	0,57	194,23
Сухой чистый	0,51	138,96
Чистый, политый водой	0,42	301,32
Мокрый с подачей песка	0,24	245,59
Чистый сухой, с подачей песка	0,61	156,27
Замасленный рельс	0,16	148,97

ПРИЛОЖЕНИЕ 5

Таблица П.3 - Значения коэффициентов сцепления шины с дорогой

Опорная поверхность		Коэффициент сцепления для шин	
Наименование	Состояние	высокого давления	низкого давления
Асфальтобетонное покрытие	Сухое	0,50...0,70	0,70...0,80
	Мокрое	0,35...0,45	0,45...0,55
	Загрязненное	0,25...0,45	0,25... 0,40
Снег	Рыхлый укатанный	0.20...0,30	0,20... 0,40
		0,15...0,20	0,20...0,25
Обледенелая дорога, лед		0,08...0,15	0,10...0,20

ПРИЛОЖЕНИЕ 6

Таблица П.4 - Значения коэффициентов сопротивления качению шины в ведомом режиме

Опорная поверхность		Коэффициент сопротивления качению
Наименование	Состояние	
Асфальтобетонное покрытие	В хорошем	0,015...0,018
	В удовлетворительном	0,018...0,020
Снежная дорога	Укатанная	0,030...0,050
Обледенелая дорога		0,015...0,018

ПРИЛОЖЕНИЕ 7

Таблица П.5 - Значения коэффициента сцепления колесной пары с рельсом

Состояние поверхности рельсов	Режим тяги			Режим торможения		
	max	среднее	min	max	среднее	min
Сухая обезжиренная	0,84	0,58	0.32	0,240	0.167	0.091
Чистая политая водой	0,61	0,41	0,20	0.174	0,113	0,052
Замасленная	0,24	0,16	0.08	0,068	0.046	0,023

ПРИЛОЖЕНИЕ 8

Таблица П.6 - Удельные сопротивления при качении колесной пары по рельсу, Н/кН

От трения качения колесной пары по рельсу	0,3...0,6
Сопротивление движению трамвая на роликовых подшипниках	0,01...0,02
От трения скольжения колесной пары по рельсам	0,1
От неправильного формирования колесных пар и установки их в тележки	0,2
От виляния вагона	0,15
От ударов колесной пары на стыках рельсов	0,3...0,6

ПРИЛОЖЕНИЕ 9

Таблица П.7 - Значения коэффициентов динамики, ускорений и показателя плавности хода трамвая

Оценка плавности хода вагона	Коэффициент динамики k_d		Ускорение вагона, м/с^2		Показатель плавности хода
	вертикальный	горизонтальный	вертикальное	горизонтальное	
Отличная	до 0,1	до 0,05	1,0	0,5	до 1
Хорошая	0,10...0,15	0,05...0,10	1,0...1,5	0,5...1,0	до 2
Удовлетворительная (допустимая для пассажирских вагонов)	0,16...0,20	0,11...0,15	1,6...2,0	1,1...2,0	до 3.25
Допустимая (для грузовых вагонов)	0,21...0,35	0,16...0,25	2,1...3,5	2,1...3,0	до 4
Непригодная для регулярного движения (по воздействию на конструкцию и организм человека)	0,36 и более	0,26 и более	3,6 и более	3,1 и более	до 5
Небезопасная при длительном движении (по устойчивости вагона и воздействию на организм человека)	более 0,7	более 0,4	более 7,0	более 5,0	более 5

ПРИЛОЖЕНИЕ 10

Таблица П.7 - Основные технические характеристики трамвайных вагонов

Показатели	Четырехосные								Сочлененные				
	T-2SU	T-3SU	КТМ-5М	КТМ 71-608	Т6В5	РВЗ-6М2	ЛМ-2000	АКСМ 60102	GT8D	ЛВС-97	АКСМ 743	GT8D	К4000
1	2	3	4	5	6	7	8	9	10	11	12	13	14
Габаритные размеры, м													
длина кузова	14,0	14,0	15,1	1525	15,3	14,1	14,1	15,3	26,0	22,0	26,0	26.0	28,4
ширина кузова	2,5	2,5	2,6	2,5	2,5	2,6	2,5	2,5	2,37	2,55	2,5	2,4	2,65
высота от головки рельса	3,05	3,05	3,15	3,10	3,15	3,15	3,10	3,15	3,19	3,15	3,15	3,165	3,36
база вагона	6,4	6,4	6,4	7,35	7,5	6,4	6,4	7,5	6+6,7+6	7,5	9,3+9,3	9,04	10,1
база тележки	1,9	1,9	1,94	1,94	1,90	1,94	1,94	1,94	1,80	1,94	1,4 1	1,8	1,8/1,9
Вместимость:													
число мест для сидения	38	36	32	32	40	38	17	30	69	-	66	78	68
нормальное наполнение	94	95	140	135	128	119	150	120	218	180	184	178	185
максимальное наполнение	—	—	224	238	168	197	—	211	300	270	302	—	—
Масса вагона, т													
снаряженная	—	17,3	18,0	21,0	18,4	18,4	18,0	20,0	30,5	30,0	30,5	31,0	35,0
при номин. заполнении	—	—	—	—	28,0	25,3	—	34,8	45,0	—	45,57	—	—

Окончание табл. П.7

1	2	3	4	5	6	7	8	9	10	11	12	13	14
Тип ТЭД	ТМ 022	ТЕ 022	ДК-259	ДК-259Е	ТЕ 023	ДК-259	ДК-259Е	ДК-263Б	GBD 120D	ДК-259Е	ДК-263 Б	MLU 3443	4LXA 1442
Номинальная мощность, кВт	40	40	40	50	45	40	50	80	120	50	80	160	120
Число ТЭД	4	4	4	4	4	4	4	4	2	4	4	4	4
Система управления ТЭД	—	—	РК	РК	ти	РК	РК/ IGBT	ТИ	непо-средст-венная	РК/ IGBT	ТИ	IGBT	GTO
Передаточное число редуктора	—	—	—	—	7,36	7,17	—	—	—	—	—	5,06	7,225
Максим. скорость с номин. загрузкой, км/ч	65	65	—	75	65	65	75	62	—	75	62	80	80
Средн. пусковое ускорение, м/с^2	1,8	1,8	1,6	1,5	1,8	1,2	—	—	—	—	—	1,1	—
Замедление при электрическом торможении, м/с^2	1,0	1,0	1,5	1,5	—	—	—	—	—	—	—	1.1	
Замедление экстренное, м/с^2	2,3	2,5	2,55	3,0	2,3	3,0	—	—	—	—	—	3,0	—

Таблица П.8 – Классы исключительных ситуаций

Имя класса	Описание
EAbort	"молчаливое" исключение, предназначенное для намеренного прерывания вычислений и быстрого выхода из глубоко вложенных процедур и функций. Генерируется процедурой **Abort**
EAbstractError	попытка вызвать абстрактный метод
EAccesViolation	приложение осуществило доступ к неверному адресу в памяти, обычно это обозначает, что программа обратилась за данными по неинициализированному указателю
EArrayError	ошибка манипулирования с потомками класса **TBaseArray:** использование ошибочного индекса элемента массива, добавление слишком большого числа элементов в массив фиксированной длины, попытка вставки элемента в отсортированный массив
EAssertionFailed	попытка использования ложного выражения
EBitsError	ошибка доступа к массиву булевых величин
ECacheError	ошибка построения кэша в кубе решений
EHeapExeption	класс **EheapException,** его потомки - **EOutOfMemory** и **EinvalidPointer** - используются, чтобы оперировать при неудачном распределении динамической памяти и неправильных операциях с указателями
EOutOfMemory	свободная оперативная память исчерпана
EOutOfResources	свободных ресурсов нет
EInvalidPointer	попытка освободить недействительный указатель. Обычно это означает, что указатель уже освобожден
EInOutError	ошибка доступа к файлу или устройству ввода вывода. Код ошибки содержится в поле **ErrorCode.** Значение кода ошибки: 2 - файл не обнаружен; 3 - неправильное файловое имя; 4 - слишком большой файл; 5 - доступ не возможен; 100 - EOF; 101 - диск полный; 106 - неправильный ввод
EIntError	общий класс исключительных ситуаций целочисленной арифметики, от которого порождены классы **EDivByZero, ERangeError, EIntOverFlow**
EDivByZero	попытка деления целого числа на нуль
ERangeError	выход за границы диапазона целого числа или результата целочисленного выражения
EIntOverFlow	переполнение в результате целочисленной операции

EmathError	общий класс исключительных ситуаций вещественной математики, от которого порождены классы **EInvalidOp, EZerodivide, EOverflow, EUnderflow, EInvalidArgument**
EInvalidOp	неверный код операции вещественной математики
EZeroDivide	попытка деления вещественного числа на нуль
EOverflow	потеря старших разрядов вещественного числа в результате переполнения разрядной сетки
EUnderflow	потеря младших разрядов вещественного числа в результате переполнения разрядной сетки
EInvalidArgument	исключительная ситуация, возникающая при решении финансовых вычислений и функций, выход из области действия одного из параметров
EExternalExeption	исключительная ситуация операционной системы, которая не соответствует ни одному из стандартных классов исключительных ситуаций **Delphi.** Это, например, может быть исключительная ситуация, возникшая в **DLL** – библиотеке, разработанной на **C ++**
EPrivilige	попытка выполнить привилегированную инструкцию процессора, на которую программа не имеет права
EStackOverflow	стек приложения не может быть больше увеличен
EControlC	во время работы консольного приложения пользователь нажал комбинацию клавиш **Ctrl + C**
EinvalidCast	неудачная попытка приведения объекта к другому классу с помощью оператора **as**
EConvertError	**EConvertError** - происходит в случае возникновения ошибки при выполнении функций **StrToInt** и **StrToFloat**, когда преобразование строки в соответствующий числовой тип невозможна
EVariantError	невозможность преобразования варьируемой переменной из одного формата в другой
EPropReadOnly	попытка установки значения свойства в объекте **OLE Automation,** которое доступно только по записи
EPropWriteOnly	попытка получения значения свойства в объекте **OLE Automation,** которое доступно только по записи

Содержание

Учебное издание

РАВИНО Виктор Валерьевич
АТАМАНОВ Юрий Евгеньевич

ИНФОРМАТИКА

Методическое пособие
по выполнению курсовой работы
для студентов специальностей
1-37 01 03 «Тракторостроение»,
1-37 01 04 «Многоцелевые гусеничные и колесные машины»,
1-37 01 05 «Городской электрический транспорт»

Технический редактор М.И. Гриневич

Подписано в печать 02.03.2007.
Формат $60\times84^{1}/_{16}$. Бумага офсетная.
Отпечатано на ризографе. Гарнитура Таймс.
Усл. печ. л. 4,88. Уч.-изд. л. 3,81. Тираж 130. Заказ 73.

Издатель и полиграфическое исполнение:
Белорусский национальный технический университет.
ЛИ № 02330/0131627 от 01.04.2004.