Московский государственный технический университет
гражданской авиации
Факультет прикладной математики и вычислительной техники
Кафедра прикладной математики

Коновалов В.М.

Прикладное программное обеспечение

ПОСОБИЕ

к выполнению курсовой работы

для студентов IV курса
специальности 230401
дневного обучения

Москва – 2009

Рецензент: проф. Кузнецов В.Л..

Коновалов В.М.

Пособие к выполнению курсовой работы по дисциплине "Прикладное программное обеспечение". – М.: МГТУ ГА, 2009

Представлен необходимый для выполнения курсовой работы методический материал, помогающий студенту определить состав, порядок выполнения и требования к оформлению разрабатываемого программного продукта.

Издается в соответствии с Учебным планом подготовки студентов специальности 230401 - ”Прикладная математика”, специализации “Математическое и программное обеспечение информационных систем на воздушном транспорте” в рамках учебной дисциплины “Прикладное программное обеспечение”.

Рассмотрено и рекомендовано к изданию в РИО МГТУ ГА на заседаниях кафедры ПМ и методического Совета ФПМиВТ 24 02 2009 г.

СОДЕРЖАНИЕ

Предисловие

Курсовая работа направлена на закрепление знаний и практических навыков:

- работы в инструментальной среде Visual Basic for Applications (VBA) для разработки программных средств автоматизации приложений;
- применения пакета прикладных программ MS Office (Excel, Word, Access), используемых в качестве **основных** приложений для разработки разнообразных пользовательских приложений;
- приобретения профессиональных приемов работы, путем создания средств автоматизации управления процессом обработки данных в пользовательских приложениях. В качестве таких средств в курсовой работе используются **макросы и модули VBA**.

VBA – это инструмент разработки приложений. Подобно другим средствам, например, Borland Delphi, Microsoft Visual C ++, VBA позволяет создавать полностью автоматические программные продукты. Такие программные средства автоматизации можно использовать, в частности, при оформлении документов (подготовка текстов), при вычислениях и анализе данных таблиц (электронных таблиц), при обработке данных, содержащихся в таблицах базы данных. Перечисленные функции по обработке данных реализуются оффисными прикладными программами: MS Word, MS Excel, MS Access и др. VBA является встроенной (интегрированной) уникальной средой для прикладных программ Office, расширяющей их функциональные возможности.

При решении задач с помощью VBA создается приложение - проект. Проекты VBA выполняются совместно с другими приложениями. Приложение, в котором разрабатывается и выполняется проект VBA, называется основным. Например, можно создать проект VBA, который работает вместе с Microsoft

Excel. В этом случае Excel является основным приложением. Не используя основное приложение, нельзя построить приложение VBA. В этом - основное различие между VBA и "чистым" языком программирования Visual Basic (VB), с помощью которого можно создать полностью самостоятельное приложение. Тем не менее, VBA учитывает специфические особенности основного приложения. Поэтому при разработке функционально ориентированных приложений, для которых можно явно определить основное приложение, предпочтение следует отдавать VBA, а не VB, что снизит трудоемкость программирования.

Для реализации цели обучения в рамках выполнения курсовой работы достаточно ограничиться основными приложениями, в качестве которых можно использовать офисные прикладные программы MS Excel, MS Word, MS Access, выполняющие функции по работе с вычислениями, текстами и хранимыми данными в базе данных.

1. Состав, порядок выполнения, требования к оформлению расчетно-пояснительной записки курсовой работы.

Основные усилия должны быть направлены на усвоение приемов разработки элементов VBA–приложения: форм, таблиц, отчетов, макросов, модулей, программирования объектов управления на форме. Тем самым создаются предпосылки к созданию собственно VBA–приложения, как охваченного единым замыслом программного продукта в виде завершенного проекта, что и является предметом выполнения курсовой работы.

Приступая к выполнению работы, студент должен иметь четкое представление о прикладной стороне решаемой задачи, которое он формирует **самостоятельно**, исходя из выбора по собственному усмотрению предметной области приложения. Обычно - это экономико-математическая модель, базирующаяся на информационной технологии, реализуемой средствами вычислительной техники в некоторых производственных условиях (на

предприятии, фирме, компании и т.п.). Здесь требуется полное понимание на уровне **содержательного** описания **того**, **что** будет являться предметом реализации на компьютере.

Следующим моментом является ответ на вопрос: **какие процедуры управления** информационным потоком следует **подвергнуть автоматизации**. Очевидно, что более высокая степень автоматизации процедур обработки данных способствует достижению более высокого уровня надежности обработки, поскольку исключаются ошибки ввода, свойственные ручному труду. Здесь отметим, что кроме чисто технической стороны в решении этой задачи существует и психологический аспект, связанный с более или менее удачным построением интерфейса пользователя с системой. Понятно, что более комфортные условия работы пользователя будут побуждать его к взаимодействию с системой, а не угнетать. Следовательно, разрабатывая интерфейсную часть, Вы должны уделить достаточно внимания для продумывания диалогов потенциального пользователя с системой, исключить как излишнюю перегруженность окон интерфейса командными кнопками, так и избежать недостатка в элементах управления автоматикой.

Исходя из общих соображений, сформулируем требования к **содержанию и оформлению** курсовой работы.

В содержание курсовой работы включить:

1. Текст, представляющий собой содержательное описание разрабатываемого приложения.

Здесь описывается целевое назначение и функции, которые должны быть реализованы в приложении.

2. Описание логической структуры приложения.

В этом разделе подробно описываются компоненты приложения для последующей разработки программных средств: перечень таблиц, описание их полей, взаимосвязь таблиц, расположение таблиц, например, на листах рабочей книги Excel (возможно, в различных книгах); спецификации имен; приводятся

схемы информационных потоков от ввода до получения конечного результата; выделяются “узкие” места, обусловленные ручными операциями; определяется перечень процедур обработки данных, которые целесообразно выполнять в автоматическом режиме; приводятся эскизы диалоговых окон с элементами управления, элементами меню, которые реализуют определенные ранее автоматические режимы и т.п.

3. Разработка программных модулей.

В этом разделе приводится описание разработки программных средств: характеристика среды, в которой реализованы программные модули; коды программ **с комментариями** (обязательно!), их привязка к элементам управления в диалоговых окнах; реакция программы на нештатные ситуации – процедуры обработки ошибок (обязательно!) и т.п.

4. Руководство пользователя.

Здесь следует представить инструкцию конечному пользователю о назначении командных элементов управления в диалоговых панелях, о последовательности отработки цепочек команд и т.п. Написать хорошее руководство - всегда трудно, поскольку нужно стать на точку зрения человека, быть может, впервые знакомящемуся с вашей разработкой. Стиль изложения должен быть простым и понятным.

5. Внешнее оформление пояснительной записки традиционно:

- Титульный лист (уточнить последнюю редакцию!)
- Индивидуальное название темы курсовой работы (не обобщенное!)
- Содержание (проставить ссылки на страницы!)
- Нумерация страниц
- Использованные литературные источники информации
- Пояснительная записка курсовой работы оформляется в виде электронного файла: «**Фамилия-ППО-КурсРаб.doc**». Возможно дополнительное оформление пояснительной записки с использованием печати на бумаге.

Примеры индивидуальных тем курсовой работы, выполнявшихся студентами.

1. Программный комплекс для расчета рентабельности авиационного подразделения.
2. Программа для расчета командировочных расходов сотрудника фирмы.
3. Автоматизированное рабочее место для контроля деятельности коммерческой структуры.
4. Автоматизированная рабочая книга для учета периодичности размещения рекламных модулей и своевременности оплаты услуг.
5. Система интерактивного расчета консолидированного баланса.
6. Автоматизированная обработка статистической информации о выступлении спортивных клубов.
7. Информационно-справочная система о торгово-закупочной деятельности фирмы.
8. Автоматизация выбора конфигурации при покупке компьютеров
9. Автоматизация процедуры ведения “Главной книги” предприятия.
10. Система графического анализа товарооборота фирмы.
11. Элементы автоматизации в работе продавца автомобилей при взаимодействии с клиентом по оформлению бланка расчета.
12. Автоматизированный учет результатов работы пункта обмена валют.
13. Автоматизация бухгалтерии: процедуры ведения книги хозяйственных операций.
14. АРМ менеджера туристической фирмы.
15. Автоматизация функций менеджера аэропорта по чартерным рейсам.
16. Процедуры автоматизации программного комплекса финдиректора фирмы.
17. Автоматизация выбора между двумя проектными решениями оборудования для технического обслуживания самолета на основе анализа статистики испытания опытных образцов посредством проектирования в среде VBA.

18. Процедуры автоматического взаимодействия таблиц бизнес-плана авиакомпании.

Предлагаемые темы for **advanced students**:

19. Автоматизация подготовки страниц Web с помощью VBS-скриптинга.
20. Разработка процедур автоматизации работы в Outlook на основе VBA.
21. Разработка процедур автоматизации работы в PowerPoint с помощью VBA.
22. Автоматизация работы в Project с помощью VBA.
23. Автоматизация работы в Binder с помощью VBA
24. Разработка процедур защиты проектов с использованием модулей VBA.
25. Разработка справочной системы приложения с выводом контекстной информации в строке состояния.

Для защиты курсовой работы студент должен представить расчетно-пояснительную записку, исполняемые файлы и подтвердить работоспособность разработанного приложения на компьютере.

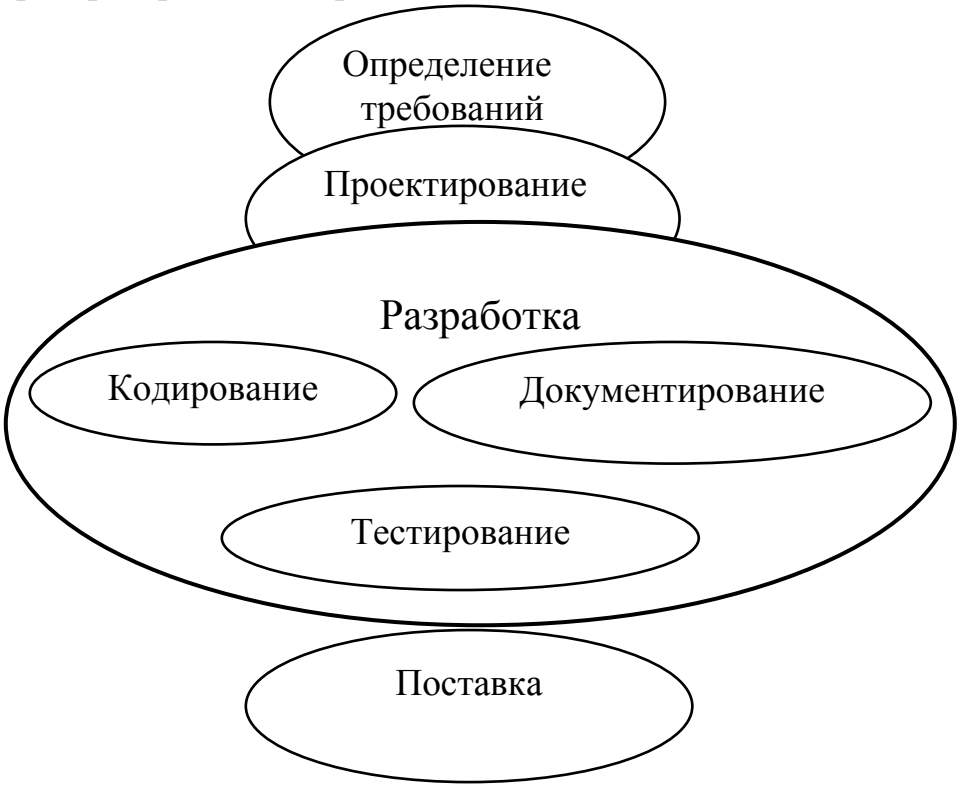
2. Принципы создания приложений.

Создание приложений - это процесс **описания**, **построения** и **поставки** программных продуктов. Программными продуктами является все программное обеспечение, включая операционные системы, среды разработки баз данных, инструменты для программирования, а также приложения, предназначенные для решения одной конкретной задачи. Хотя при детальном рассмотрении разработка каждого из этих типов программных продуктов отличается от других, основные действия, выполняемые при создании любого приложения, очень схожи. Подход к созданию программного обеспечения, который Вам предстоит использовать при выполнении курсовой работы, относится к созданию **объектно-ориентированных** приложений.

2.1. Этапы создания приложений.

Создание приложений проходит в несколько взаимосвязанных этапов. Понимание сути каждого из этапов поможет Вам правильно выбрать круг действий, которые необходимо выполнить для создания качественной программы. Остановимся на четырех основных фазах создания программного продукта, изображенных на рисунке ниже.

- ***Определение требований***. На данном этапе требуется выявить потенциальных пользователей создаваемого приложения и максимально точно описать предъявляемые ими требования.
- ***Проектирование***. На данном этапе требуется выполнить анализ задач, для решения которых разрабатывается приложение, а затем создать проект, удовлетворяющий требованиям пользователей.
- ***Разработка.*** На данном этапе требуется написать код, а затем отладить и документировать программу, убедившись, что она удовлетворяет требованиям пользователей.
- ***Поставка***. На данном этапе требуется записать программу на магнитный носитель и распространить ее среди пользователей.



Разработка программного обеспечения – итерационный процесс, некоторые этапы которого могут перекрываться. Обычно уточнение требований происходит одновременно с переходом к укрупненной разработке проекта, а детальная разработка еще продолжается, когда начинается кодирование.

3. Характеристика этапов проектирования.

3.1 Определение требований.

На этапе *определения требований* наиболее важно понять задачу, для решения которой будет использоваться программное обеспечение. Плохой проект является результатом плохого знания нужд пользователей, либо следствием непрерывно меняющихся требований. Так или иначе, прежде чем начать проектирование и кодирование, необходимо получить как можно больше сведений. Если возможно, поговорите с потенциальными пользователями программы, чтобы узнать ответы на следующие вопросы:

- Зачем нужен программный продукт?
- При решении какой задачи планируется использовать программный продукт?
- Как решается эта задача в настоящее время? Что пользователю нравится или не нравится в способе решения задачи?
- Как часто потребуется использовать новый программный продукт?
- Как новый продукт будет упрощать (или усложнять!) решение задачи?
- Должен ли программный продукт обеспечивать вывод на печать? Как будет использоваться напечатанная информация?
- В каких операционных средах должна работать программа? Какие операционные системы и аппаратные средства имеют пользователи?
- Работают ли пользователи в локальной вычислительной сети? Соединены ли они с Internet или intranet (корпоративная вычислительная система)?

- Какими мониторами располагают пользователи? Какова минимальная разрешающая способность используемых ими дисплеев?
- Если система – многопользовательская, какая требуется защита, какие накладываются ограничения и почему?

Располагая ответами на подобные, казалось бы, простейшие вопросы, легче понять проблему. Полученные сведения могут отличаться от первоначальных. Предположим, необходимо разработать инструмент для ввода информации в базу данных распространения продукции. После разговора с пользователями оказывается, что отдел сбыта уже имеет базу данных заказчиков, которая содержит общую информацию о покупателе, но не включает сведения о названии приобретенных товаров. Хотя это и не входило в исходные требования, целесообразно предусмотреть возможность импорта данных о покупателе из базы данных сбыта в базу данных распространения продукции. Такого рода сведения сильно влияют на реализацию проекта как с точки зрения структуры данных, так и интерфейса.

3.2. Проектирование программного продукта.

При проектировании рассматриваются задачи пользователей, данные, процессы и предполагаемые инструментальные средства разработки. Имея ясное представление о решаемой задаче, можно приступить к проектированию, используя следующие методы:

- *Создание модели данных*. Модель данных должна включать наиболее важную информацию, а также задачи пользователей.
- *Создание модели объектов*. Модель объектов наиболее полезна при использовании объектно–ориентированных инструментов разработки.
- *Создание концептуального прототипа*. Концептуальный прототип помогает яснее представить требования пользователя. Прототип следует построить максимально быстро.

- *Создание архитектуры*. При разработке архитектуры легче представить внутренние компоненты программы.

Создание модели данных. Моделью данных в большинстве случаев является диаграмма обработки данных или иная схема, которая содержит основные объекты и показывает взаимосвязь этих объектов и задач пользователя. Модель определяется требованиями к программе, поэтому необходимо, чтобы требования максимально точно соответствовали задачам пользователя. Кроме того, наличие алгоритма обработки данных – один из главных факторов успешной разработки программного обеспечения.

Для создания модели данных:

1. Выявить задачи, которые выполняет пользователь (или группа пользователей).

Предположим, что пользователь выполняет работу с вышеупомянутой базой данных распространения продукции и базой данных сбыта. Тогда решаемыми задачами могут быть, во-первых, импортирование записей о покупателях в базу данных распространения продукции, во-вторых, ручной ввод информации о покупателях, наконец, добавление в записи сведений о заказанных товарах.

2. Определить основные объекты и процессы.

Основными объектами в базе данных распространения продукции являются покупатели, товары и заказы. Покупатель имеет имя, адрес, номер телефона. Товар – идентификатор, номер серии, цвет. Заказ – количество, дату заказа, способ оплаты. Процессами являются создание отчетов о распространении продукции по покупателям и регионам и удаление клиентов, которые не заказали ни одного товара в течение последних двух лет.

3. Выявить одинаковые данные, задачи и процессы.

Например, при импорте записей и вводе их вручную, выполняется одна и та же проверка условий на значение.

4. Создать диаграмму, которая показывает взаимосвязь задач пользователя, объектов и процессов.

В модели для базы данных распространения продукции целесообразно показать влияние действий пользователя на информацию о покупателе, включая ввод, удаление и печать этих данных.

Объектно–ориентированное проектирование. При использовании объектно-ориентированного подхода и графического интерфейса, программный продукт можно рассматривать как совокупность объектов. Объектно-ориентированный подход состоит в том, чтобы создать приложение, которое состоит из отдельных компонентов. При этом структура объектов, события и методы контролируются пользователем.

Для эффективного применения объектно-ориентированного подхода полезно представить требования заказчика в форме модели, которая включала бы объекты приложения и связанные с ними свойства, методы и события. Последнее продиктовано тем, что при использовании объектно-ориентированного подхода большинство задач пользователя не так – то просто представить в форме объектов: необходимые действия нелегко втиснуть в рамки моделей объектов. Например, для обыденной речи свойственна такая фраза: "Я хочу написать письмо". Но вряд ли кому – то придет в голову сказать: "Я хочу, чтобы мой документ выполнил метод *Создать*". Конечно, это упрощенный пример, но он отражает основную проблему применения объектно-ориентированного подхода: в результате его применения можно создать приложение, с которым будет неудобно работать. Выход здесь один – использовать наряду с данным подходом другие методы проектирования.

Создание концептуального прототипа. В концептуальном прототипе отражается влияние модели данных на интерфейс. Прототип состоит из форм, меню, панелей инструментов и других компонентов интерфейса, а также включает примеры записей. Однако не рекомендуется задерживаться на данном этапе и до бесконечности совершенствовать интерфейс.

Основное назначение прототипа состоит в том, чтобы выяснить мнение заказчика о предложенном интерфейсе. С помощью концептуального прототипа легче ответить на следующие вопросы о модели данных:

- Является ли разбивка на объекты оправданной и интуитивной?
- Облегчают ли действия пользователей объекты и связанные с ними задачи?

При разработке интерфейса попытайтесь ответить на следующие вопросы:

- Облегчает ли интерфейс действия пользователя?
- Являются ли внешний вид и действия с помощью прототипа интуитивными для пользователя?
- Имеются ли средства повышения скорости работы с приложением?

Чтобы узнать мнение об интерфейсе, обратитесь к своим коллегам или реальным пользователям. Более точные сведения можно получить, проведя тесты удобства использования.

Будет ли принят концептуальный прототип за основу или нет – зависит от его соответствия разработанному проекту. Если демонстрационная версия создана без учета требований технического задания, иерархия функций, объектов, использование переменных, а также соглашения по именованию элементов управления не продуманы, то, вероятно, такой прототип конечного продукта будет отвергнут. Прототип должен соответствовать требованиям технического задания на проектирование и учитывать реальные потребности пользователей, включая обработку требуемых объемов информации. Кроме того, правильно разработанный прототип должен допускать любые усовершенствования в будущем. Как правило, первые прототипы обычно используются для уточнения общих требований к готовому программному продукту.

Создание технического проекта. Одновременно с разработкой концептуального прототипа создается технический проект или архитектура, которая поддерживает модель данных и позволяет эффективно выполнять задачи пользователей. Технический проект (архитектура) определяет главные технологические компоненты и их взаимосвязь. В объектно-ориентированном

приложении наименьшими составляющими технического проекта являются интерфейс и связанный с ним код, источники информации, а также другие компоненты, с помощью которых обеспечивается взаимодействие между интерфейсом и данными. Интерфейс можно разработать в Visual Basic, HTML или Visual C^{++}. Источниками данных могут служить Microsoft Access, источники данных ODBC, например, база данных SQL Server, текстовые файлы или документы других приложений (рабочий лист Microsoft Excel или документ Microsoft Word). Компонентами управления данными могут быть объекты OLE Automation, библиотеки ODBC, объекты доступа к данным (DAO, ADO), либо ядро Visual Basic Jet.

Тесты на удобство использования. Тестирование на удобство использования – это процесс оценки простоты использования продукта с помощью набора тестов. Тесты помогают выявить трудности, возникающие у пользователя при работе с программой, а также определить причины этих затруднений, вне зависимости от того, с чем они связаны: с использованием средств программного продукта, внешним видом интерфейса или полнотой документации.

Для проведения тестирования:

1. Определите средства, которые необходимо протестировать.
2. Выберите пользователей, которые предположительно будут использовать программный продукт.
3. Разработайте несколько упражнений, которые должен выполнить пользователь.
4. Проведите тестирование.
5. Обобщите результаты тестирования и проанализируйте их.
6. Дайте рекомендации по программному продукту, основываясь на проведенном анализе.

Традиционным и наиболее интересным типом тестов на удобство использования являются тесты на выполнение задач. При проведении этих

тестов производится наблюдение за пользователями в то время, когда они решают поставленные задачи с использованием разработанной системы. Данный тип тестирования чрезвычайно полезен для проверки удобства использования средств и выявления проблем.

4. Разработка программного продукта.

Никогда нельзя сказать, когда заканчивается этап проектирования, а когда начинается этап разработки. Обычно, еще не завершены вопросы проектирования, а уже начато программирование, что совершенно нормально. В большей степени это относится к использованию объектно-ориентированных, визуальных средств программирования, например, VBA, которые предназначены для быстрой разработки прототипа. Используя эти средства, создание и тестирование программного продукта можно начинать в любой момент проектирования, даже если детали проекта до конца еще не ясны. Имея перед глазами разработанный проект в деталях, легче оценить, удовлетворяет ли он предъявляемым требованиям или нет, а затем изменить его.

Этап разработки включает несколько циклов кодирования и тестирования функциональных возможностей программного продукта. Процесс кодирования и тестирования является ключевым в разработке любого проекта.

4.1. Разработка программы.

После определения основных объектов и компонентов проекта и выбора инструментальных средств разработки, можно начинать кодирование. При использовании объектно-ориентированного инструмента разработки, включающего графический интерфейс, необходимо выполнить следующие действия:

- Создать с помощью графических средств инструмента разработки интерфейсные объекты, включая формы, элементы управления в формах, а также меню.
- Задать свойства форм, меню и элементов управления, определив также их внешний вид, поведение и состояние интерфейсных объектов, и указать источники данных для элементов управления.
- Разработать процедуры, запускаемые в ответ на события, которые возникают в результате действий пользователя, включая создание программ, выполняемых при выборе команды меню, нажатии кнопки, получении фокуса элементом управления, вводе данных и перемещении к другому элементу управления, а также процедур обработки ошибок.
- Протестировать программу.

При кодировании необходимо проверить и отладить процедуры, проведя первоначальное тестирование программы.

4.2. Тестирование программного продукта.

Обычно тестирование, выполняемое программистом, не столь всесторонне, как это требуется. При разработке программного продукта в условиях коллективной работы (в компании, разрабатывающей программное обеспечение), функции кодирования и тестирования выполняются разными специалистами. При этом проблеме проверки работоспособности приложений уделяется огромное внимание. В случае разработки приложения одним человеком (в данном случае – курсовая работа), на него возлагается и ответственность за тестирование приложения. Но ведь и масштаб проекта несоизмеримо меньший, чем у корпоративной разработки!

Типы тестирования. Вне зависимости от того, выполняет ли проверку программного обеспечения группа специалистов или оно тестируется

самостоятельно, чтобы обнаружить ошибки, требуется выполнить каждый из следующих типов тестов:

- *Функциональное тестирование*. Функциональное тестирование включает проверку каждой команды меню, панели инструментов и каждой операции, которую выполняет система. Хотя функциональное тестирование гарантирует работу всех средств системы по отдельности, однако, при этом работа всего программного продукта в целом не проверяется.
- *Тестирование приложения*. При выполнении тестирования приложения моделируются действия пользователя. Моделируя различные комбинации действий, можно найти ошибки, относящиеся к интеграции компонентов, а также недоработки, которые не были обнаружены при функциональном тестировании.
- *Тестирование на скорость выполнения*. Тестирование на скорость выполнения позволяет определить темп выполнения задач. При этом измеряется время отклика системы на определенные действия, например, запуск приложения, обновление экрана, переход из одного окна в другое, выполнение запроса, сохранение документа и т. п. Обычно при этом определяются параметры технико-экономической эффективности. Например, одновременно с измерением времени на выполнение запроса записывается также размер базы данных.
- *Предельное тестирование*. Предельное тестирование позволяет определить пропускную способность программного продукта. Производится последовательный запуск задач при увеличении размеров файлов или баз данных, и таким образом выявляются предельные значения объемов данных, при которых еще возможно нормальное завершение выполняемых задач.

Максимальный объем данных является существенным ограничением при программировании. Если пропускная способность низка, то необходимо определить, как можно повысить значение этого параметра. Возможно, потребуется оптимизировать код или использовать другие алгоритмы.

Здесь отметим, что профессиональный подход к тестированию программного обеспечения предполагает использование автоматизированных тестов, например, поставляемого Microsoft приложения MSTest.

Ошибки, выявляемые при тестировании. Ошибки классифицируются по степени серьезности:

- *Фатальные ошибки*. Сбои в системе, которые приводят к потере ее работоспособности или порче данных.
- *Серьезные ошибки*. Ошибки, которые не позволяют достичь желаемого результата при использовании некоторого средства или функции программного продукта, включая невозможность исполнения содержащейся в меню или диалоговом окне команды, выполнения процесса или форматирования текста.
- *Маленькие ошибки*. Ошибки, которые вызывают неудобство работы с системой, например, ошибки в задании порядка перехода от одного элемента управления к другому в форме ввода информации. К данному типу ошибок можно отнести также беспричинные сообщения об ошибках.
- *Незначительные ошибки*. К данному типу ошибок относят орфографические ошибки в надписях или не выровненные элементы управления.

После переноса программного продукта на другие компьютеры, могут проявиться недоработки, связанные с установкой и работой в другом окружении.

На момент сдачи программного продукта в эксплуатацию может оказаться, что на устранение выявленных в процессе тестирования ошибок нет времени, либо их устранение может затронуть основы уже разработанной версии продукта. Тем не менее, обнаруженные ошибки по уровню своей серьезности не исключают распространение программного продукта. В таких случаях программный продукт сопровождается листом качества, в котором перечисляются число, степень серьезности имеющихся ошибок и указывается на возможность их устранения в следующей версии программного продукта.

Например, в листе качества программного продукта может содержаться следующая информация:

- Приемлемое время запуска программы: менее 5 секунд.
- Время для импорта исходных данных (5000 записей): 48 минут. Рекомендуется сократить время импорта, по крайней мере, до 20 минут.
- Время на обновление 1000 записей: 8 минут. Рекомендуется сократить время обновления, по крайней мере, до 4 минут.
- Фатальные ошибки, приводящие к потере системой работоспособности или порче данных, не обнаружены.
- Выявлено две серьезных ошибки. Отчеты печатаются неправильно, и не работает флажок "Печать в файл". Эти ошибки не являются настолько серьезными, чтобы остановить распространение программного продукта, но в следующей версии необходимо устранить их.
- Выявлено 10 маленьких ошибок, из – за которых не следует останавливать распространение программного продукта, но в следующей версии необходимо устранить их.
- выявлено 25 незначительных ошибок, из–за которых не следует останавливать распространение программного продукта, но в следующей версии необходимо устранить их, чтобы придать программе законченность.
- Интерфейс для ввода и поиска нового продукта неудобен и требует внесения изменений в новой версии.

4.3. Документирование программного продукта.

Составление документации по программному продукту необходимо начать на этапе разработки. Проверка написанной документации должна быть тщательной, как и проверка программного обеспечения. К документации, сопровождающей программный продукт, можно отнести (наряду с печатной продукцией) такие электронные средства документирования, как справочные

файлы, обеспечивающие при выполнении задач контекстную подсказку. Кроме того, в продукт можно встроить мастера и обучающие программы.

Безусловно, документирование программного продукта Вашей курсовой работы должно быть выполнено, хотя и не в таком объеме, какого требуют фирменные разработки. Что касается встраивания электронных средств документирования в программный продукт, то это может служить отдельной темой курсовой работы. При подготовке электронных руководств можно использовать специальные приложения, например, Microsoft Help Compiler, поставляемое вместе с профессиональной версией Visual Basic, или Robohelp, поставляемое вместе с приложением Blue Sky Software.

Поставка. С самого начала работы над программным продуктом следует принять решение о том, в каком виде он будет распространяться: на CD–дисках, либо отправкой его в Internet или Intranet, либо просто установкой его на совместно используемом диске.

Кроме того, необходимо обеспечить пользователя информацией о том, как инсталлировать программный продукт. Пользователь составляет свое мнение о продукте, начиная с его установки, поэтому следует сделать инсталляцию максимально простой. Лучше всего написать программу установки (обычно SETUP.EXE) и дать к ней некоторые пояснения (файл README).

Возможно, Вам не придется решать проблему поставки программного продукта, тем более, созданного в рамках курсовой работы. Тем не менее, свои соображения по этому вопросу следует отразить в пояснительной записке, что создаст впечатление **завершенности** проделанной работы.

Приложение.

Здесь приведена форма-шаблон, которая заполняется студентом и сдается преподавателю не позже 2-й недели VIII-ого семестра, в течение которого выполняется курсовая работа. Ниже даются некоторые пояснения, относящиеся к выполнению работы в контексте прохождения последовательности ее этапов.

Техническое задание

на выполнение курсовой работы по дисциплине
"Прикладное программное обеспечение"
кафедра "Прикладная математика", спец. 230401

Цель работы:
выполнить проектирование и программную реализацию сетевого VBA-приложения, клиентская и серверная части которого взаимодействуют в соответствии с выбранной технологией доступа к внешним данным.

Ф.И.О._____
Группа_____

Предметная область
разработки:_____
_____
Автоматизируемые
функции:_____
_____

НазваниеКР:_____
_____
_____
Основное Приложение
для клиентской части
VBA-приложения:_____

Основное Приложение/Сервер
для серверной части
VBA- приложения _____

Используемая технология (технологии)
доступа к внешним данным:_____
_____

Форма отчетности:
I. Комплект файлов, включающий:
   1. файл расчетно-пояснительной записки;
   2. исполняемые файлы разработанного приложения;
II. Подтверждение работоспособности разработанного приложения на контрольном примере

Взаимодействие Windows-приложений, реализующее прямой доступ к данным.

КР.1. Разработать сценарий, моделирующий клиент-серверный режим взаимодействия приложения-**клиента** с приложением-**сервером** в соответствии с ТЗ на курсовую работу. Сценарий взаимодействия должен отражать **технологию** прямого доступа к внешним данным, которая основывается на модели **Universal Data Access** (Microsoft) и обеспечивает высокоуровневый API (интерфейс прикладного программирования).

Замечание 1. Клиент-серверный режим взаимодействия реализуется либо как **локальный** процесс (на одной и той же машине), либо как **сетевой** процесс (в локальной сети, в частности). Очевидно, что относительное расположение клиента и сервера в информационной среде, определяет то, какой из вышеназванных процессов будет использован.

Для студентов, выбравших в качестве основы взаимодействия **локальный** процесс, при реализации режима «клиент-сервер» наименее ресурсозатратной технологией доступа к внешним данным может оказаться объектно-ориентированная модель программирования **DAO** - в версиях DAO/MS Jet или DAO/ODBCDirect.

Для студентов, выбравших **сетевой** процесс для реализации режима «клиент-сервер», наиболее приемлемыми технологиями доступа к внешним данным могут быть DAO/ODBCDirect, **ADO** over OLE DB, **ADO.NET**.

В случае использования ADO.NET, возможен переход к Three-tier архитектуре (трехуровневой), допускающей взаимодействие сервера с "отключаемым" клиентом.

Замечание 2. Какое из приложений примет статус клиента или сервера, а также, каково их взаимное расположение в инфраструктуре – решать Вам. Наиболее типичная ситуация в случае выбора **локального варианта:** в качестве **сервера** используется Access-приложение, а в качестве **клиента** – VB/VBA-приложение, для которого в качестве **основного** выбирается любое из приложений, входящих в состав пакета MS Office.

В случае **сетевого варианта**, в качестве **клиента** обычно используется VB/VBA-приложение на основе любого приложения из пакета MS Office, а в качестве **сервера** - SQL-сервер.

Замечание 3. Выполнить графическое изображение фрагмента архитектуры, соответствующего выбранной технологии прямого доступа в соответствии с которым приложения исполняют последовательность сеансов обмена данными в автоматическом режиме.

КР.2. Реализовать **VB/VBA** –код приложений, в соответствии с ТЗ на курсовую работу, для взаимодействия **по разработанному сценарию.** В качестве инструментальной среды разработки можно использовать интегрированные среды IDE VB6.0, IDE Visual Studio2003 и последующих релизов Visual Studio.

КР.3. Отладить **VB/VBA**-код клиент-серверного режима взаимодействия приложений.

Замечание 1. Код VB/VBA должен включать в качестве **обязательных** компонент **инструкции перехвата и процедуры обработки ошибок,** а также **инструкции регистрации событий** в текстовом файле (*.log). Последовательность событий, происходящих в процессе взаимодействия приложений, **протоколировать** в форме отчета, автоматически записываемого, к примеру, в текстовый файл с помощью инструкций **Print #** или **Write #** и читаемого инструкциями **Line Input #** или **Input #**, соответственно (см. документацию по соответствующим инструкциям).

Замечание 2. Отладка программного комплекса выполняется на **предварительно** подготовленных исходных данных контрольной задачи.

Замечание 3. Результаты работы готовить в форме **расчетно-пояснительной записки** курсовой работы, с соблюдением соответствующих правил оформления, изложенных в пособии выше.