Некоммерческое акционерное общество

АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ

Кафедра компьютерных технологий

УТВЕРЖДАЮ
Декан ФАИТ
_____ Табултаев С.С.
«_____»_____ 2017 г.

TRPOSRV 5301- ТЕХНОЛОГИИ РАЗРАБОТКИ ПРОГРАММНОГО ОБСПЕЧЕНИЯ ДЛЯ СИСТЕМ РЕАЛЬНОГО ВРЕМЕНИ

Методические указания к выполнению расчетно-графические работы для магистрантов специальности 6М070400 - «Вычислительная техника и программное обеспечение» (научное и педагогическое направление)

Алматы 2017

СОСТАВИТЕЛЬ: к.ф.-м.н. А.А. Аманбаев « Технологии разработки программного обеспечения для систем реального времени». Расчетно-графические работы для магистрантов специальности 6М070400 - «Вычислительная техника и программное обеспечение» (научное и педагогическое направление). – Алматы: АУЭС, 2017.

Протокол № _6_ заседания кафедры от «_ _6__»__06___2017 г.

Заведующий кафедрой ММиПО _____ М.Ж.Байсалова

Расчетно-графические работы рассмотрен и утвержден на заседании учебно-методической комиссии факультета Аэрокосмических и информационных технологий протокол № __5__ от «___20.06__» 2017 г.

Расчетно-графические работы составлены согласно рабочей программе дисциплины «Технологии разработки программного обеспечения для систем реального времени» и является ознакомление магистрантов с вопросами проектирования сложных программных систем, обучение их методологии структурного анализа и проектирования SADT, освоение ими основ объектно- ориентированного подхода к проектированию программных систем и приобретение практических навыков применения современных технологии проектирования (CASE-технологии).

Введение

В дисциплине «Технологии разработки программного обеспечения для систем реального времени» изучаются вопросы современных технологий разработки программного обеспечения. Курс является логическим продолжением бакалаврские дисциплины. Объем аудиторных часов, выделенный на изучение дисциплины «Технологии разработки программного обеспечения для систем реального времени», не позволяет полностью охватить весь учебный материал, поэтому некоторые разделы вынесены на самостоятельное изучение. Самостоятельная работа магистранта (СРМ) предполагает не только проработку лекционного материала, подготовку к практическим и лабораторным занятиям, но и изучение дополнительного материала. Для проверки знаний, полученных в результате самостоятельной работы студента, в курсе «Технология проектирования программных систем» предусмотрены три расчетно-графические работы.

Настоящие методические указания включают варианты заданий, рекомендации к их выполнению и контрольные вопросы по трем темам:

1. Использование структурного подхода при проектировании программного продукта.
2. Реализация программы в выбранной среде программирования.
3. Тестирование. Отладка. Составление программной документации.

В приложениях содержится весь необходимый для выполнения работ справочный материал.

В качестве дополнительного источника информации для выполнения расчетно-графических работ следует использовать учебники, учебные пособия, методические разработки. Кроме того, информацию можно почерпнуть из периодических изданий по компьютерным технологиям, справочной литературы по различным программным продуктам, справочной системы каждой программы, имеющейся в памяти компьютера, а также, используя справочную систему Интернет.

Предлагаемые варианты заданий выбираются в соответствии с рекомендациями преподавателя и рассчитаны на разный уровень подготовки студентов. Преподаватель может по своему усмотрению изменить или дополнить расчетно-графическую работу, ориентируясь на общий уровень подготовки. Также по рекомендации преподавателя работы могут выполняться индивидуально или рабочей группой (до 4-х человек).

Расчетно-графические работы сдаются на проверку в сроки, установленные кафедрой. К каждой РГР необходимо приложить пояснительную записку, выполненную (согласно рекомендациям преподавателя) на листах формата А4 или в электронном виде (в формате *MS Word*), включающую титульный лист, постановку задачи, а также все необходимые пояснения по выполнению заданий. В случае если работа содержит ошибки, выполнена не в полном объеме или не соответствует выбранному варианту, она возвращается студенту для внесения исправлений.

После проверки каждая расчетно-графическая работа должна быть защищена.

1 Расчетно-графическая работа №1. Использование структурного подхода при проектировании программного продукта

Цель работы – получить практические навыки в области проектирования программного обеспечения и составления соответствующей документации.

1.1 Задание на расчетно-графическую работу

Проектирование программного продукта при использовании структурного подхода (предпроектные исследования предметной области; постановка задачи; разработка технического задания; выбор методов и разработка основных алгоритмов решения задачи; разработка структурной схемы программного продукта; выбор технологии, языка и среды программирования; разработка структурной схемы программного продукта; проектирование интерфейса пользователя).

а) согласно выбранному варианту провести предпроектные исследования предметной области и выполнить постановку задачи.

Разработка должна включать следующие модули:

- модуль теоретического обучения с тестом для самоконтроля (предусмотреть возможность возврата к теории при плохих результатах тестирования);
- модуль итогового тестирования с выдачей результатов, наличием ограничения времени, случайным выбором вопроса, записью результатов в файл;
- модуль защиты теста с использованием пароля, а также применением кодирования тестовых вопросов;

б) разработать техническое задание, которое должно включать титульный лист, введение, пункты по стандарту,

в) с помощью пакета MS Project организовать управление проектом, а именно: разработать график распределения работ и организовать контроль хода его выполнения;

г) обоснованно выбрать технологию программирования, язык и среду программирования.

д) разработать алгоритм решения задачи, выполнить его графическое описание способом, указанным в варианте задания;

е) выполнить уточнение спецификаций, составить словарь терминов.

Рекомендации к выполнению задания. Варианты выбираются с учетом рекомендаций преподавателя среди трех предложенных уровней сложности (сложность возрастает с увеличением номера уровня - I, II, III). Обоснование всех принятых решений, а также разработанные схемы и документы следует включить в пояснительную записку к расчетно-графической работе. При разработке технического задания необходимо учитывать специфику предметной области и включать те пункты по стандарту, которые имеет смысл рассматривать. При работе над проектом в MS Project определить

статистические данные проекта, базовый календарь, используемый в проекте, сформировать список задач проекта, определить длительность задач, зависимости между задачами, список ресурсов проекта. В процессе реализации проекта необходимо осуществлять постоянный контроль хода его выполнения. Необходимо регулярно вводить новые данные и отслеживать ход выполнения проекта с представлением таблицы со сводной информацией о фактическом состоянии проекта в конце каждой РГР.

1.2 Общие рекомендации к выполнению работы

1.2.1 Постановка задачи

В процессе постановки задачи четко формулируют назначение программного обеспечения и определяют основные требования к нему. Каждое требование представляет собой описание необходимого или желаемого свойства программного обеспечения. ***Различают функциональные требования***, определяющие функции, которые должно выполнять разрабатываемое программное обеспечение, и ***эксплуатационные требования***, определяющие особенности его функционирования.

Требования к программному обеспечению, имеющему ***прототипы***, обычно определяют по аналогии, учитывая структуру и характеристики уже существующего программного обеспечения. Для формулирования требований к программному обеспечению, не имеющему аналогов, иногда необходимо провести специальные исследования, называемые ***предпроектными***. В процессе таких исследований определяют разрешимость задачи, возможно, разрабатывают методы ее решения (если они новые) и устанавливают наиболее существенные характеристики разрабатываемого программного обеспечения. Для выполнения предпроектных исследований, как правило, заключают договор на выполнение научно-исследовательских работ. В любом случае этап постановки задачи заканчивается разработкой технического задания, фиксирующего принципиальные требования, и принятием основных проектных решений.

1.2.2 Способы описания алгоритмов

Именно для изображения **схем алгоритмов** программ в свое время был разработан ГОСТ 19.701-90, согласно которому каждой группе действий ставится в соответствие специальный блок. Хотя этот стандарт предусматривает блоки для обозначения циклов, он не запрещает и произвольной передачи управления, т. е. допускает использование команд условной и безусловной передачи управления при реализации алгоритма.

Представление алгоритма программы в виде схемы с точки зрения структурного программирования имеет два недостатка:

• предполагает слишком низкий уровень детализации, что часто скрывает суть сложных алгоритмов;

• позволяет использовать неструктурные способы передачи управления, причем часто на схеме алгоритма они выглядят проще, чем эквивалентные структурные.

Кроме схем, для описания алгоритмов можно использовать **псевдокоды**, **Flow-формы** и **диаграммы Насси-Шнейдермана**. Все перечисленные нотации с одной стороны базируются на тех же основных структурах, что и структурное программирование, а с другой - допускают разные уровни детализации.

Псевдокод - формализованное текстовое описание алгоритма (текстовая нотация). В литературе были предложены несколько вариантов псевдокодов.

Описать с помощью псевдокодов неструктурный алгоритм невозможно. Использование псевдокодов изначально ориентирует проектировщика только на структурные способы передачи управления, а потому требует более тщательного анализа разрабатываемого алгоритма. В отличие от схем алгоритмов, псевдокоды не ограничивают степень детализации проектируемых операций. Они позволяют соизмерять степень детализации действия с уровнем абстракции, на котором это действие рассматривают, и хорошо согласуются с основным методом структурного программирования - методом пошаговой детализации.

Flow-формы представляют собой графическую нотацию описания структурных алгоритмов, которая иллюстрирует вложенность структур. Каждый символ Flow-формы соответствует управляющей структуре и изображается в виде прямоугольника. Для демонстрации вложенности структур символ Flow-формы может быть вписан в соответствующую область прямоугольника любого другого символа. В прямоугольниках символов содержится текст на естественном языке или в математической нотации. Размер прямоугольника определяется длиной вписанного в него текста и размерами вложенных прямоугольников.

Диаграммы Насси-Шнейдермана являются развитием Flow-форм. Основное их отличие от Flow-форм заключается в том, что область обозначения условий и вариантов ветвления изображают в виде треугольников. Такое обозначение обеспечивает большую наглядность представления алгоритма.

Также как при использовании псевдокодов, описать неструктурный алгоритм, применяя Flow-формы или диаграммы Насси-Шнейдермана, невозможно (для неструктурных передач управления в этих нотациях просто отсутствуют условные обозначения). В то же время, являясь графическими, эти нотации лучше отображают вложенность конструкций, чем псевдокоды.

Общим недостатком Flow-форм и диаграмм Насси-Шнейдермана является сложность построения изображений символов, что усложняет практическое применение этих нотаций для описания больших алгоритмов.

1.2.3 Стиль оформления программы

С точки зрения технологичности хорошим считают стиль оформления программы, облегчающий ее восприятие, как самим автором, так и другими

программистами, которым, возможно, придется ее проверять или модифицировать. «Помните, программы читаются людьми», призывал Д. Ван Тассел, автор одной из известных монографий, посвященной проблемам программирования.

Именно исходя из того, что любую программу неоднократно придется просматривать, следует придерживаться хорошего стиля написания программ.

Стиль оформления программы включает:

• правила именования объектов программы (переменных, функций, типов, данных и т. п.);

• правила оформления модулей;

• стиль оформления текстов модулей.

1.2.3.1 Правила именования объектов программы. При выборе имен программных объектов следует придерживаться следующих правил:

• имя объекта должно соответствовать его содержанию, например:

Maxltem - максимальный элемент; *Nextltem* - следующий элемент;

• если позволяет язык программирования, можно использовать символ «_» для визуального разделения имен, состоящих из нескольких слов, например: *Max_Item, Next_Item;*

• необходимо избегать близких по написанию имен, например:

Index и *InDec.*

1.2.3.2 Правила оформления модулей. Каждый модуль должен предваряться заголовком, который, как минимум, содержит:

• название модуля;

• краткое описание его назначения;

• краткое описание входных и выходных параметров *с указанием единиц измерения;*

• список используемых (вызываемых) модулей;

• краткое описание алгоритма (метода) и/или ограничений;

• ФИО автора программы;

• идентифицирующую информацию (номер версии и/или дату последней корректировки, рисунок 1.1):

```
{*************************************************}
{*    Функция:  Length_Path(n:word; L: array of real):real   *}
{*    Цель: определение суммарной длины отрезков             *}
{*    Исходные данные:                                       *}
{*       n – количество отрезков,                            *}
{*       L – массив длин отрезков (в метрах)                 *}
{*    Результат:   длина (в метрах)                          *}
{*    Вызываемые модули:  нет                                *}
{*    Описание алгоритма:                                    *}
{*          отрезки суммируются методом накопления, n ≥ 0    *}
{*    Дата:  25.12.2001   Версия 1.01.                       *}
{*    Автор: Иванов И.И.                                     *}
{*    Исправления:   нет                                     *}
{*************************************************}
```

Рисунок 1.1 – Пример оформления модуля программы

1.2.3.3 Стиль оформления текстов модулей. Стиль оформления текстов модулей определяет использование отступов, пропусков строк и комментариев, облегчающих понимание программы. Как правило, пропуски строк и комментарии используют для визуального разделения частей модуля.

Для таких языков, как Pascal, C++ и Java, использование отступов позволяет прояснить структуру программы: обычно дополнительный отступ обозначает вложение операторов языка.

Несколько сложнее дело обстоит с комментариями. Опыт показывает, что переводить с английского языка каждый оператор программы не нужно: любой программист, знающий язык программирования, на котором написана программа, без труда прочитает тот или иной оператор. Комментировать следует цели выполнения тех или иных действий, а также группы операторов, связанные общим действием, т. е. комментарии должны содержать некоторую дополнительную (неочевидную) информацию.

Для языков низкого уровня, например, Ассемблера, стиль, облегчающий понимание, предложить труднее. В этом случае может оказаться целесообразным комментировать и блоки операторов, и каждый оператор.

1.2.4 Техническое задание

Техническое задание представляет собой документ, в котором сформулированы основные цели разработки, требования к программному продукту, определены сроки и этапы разработки и регламентирован процесс приемно-сдаточных испытаний. В разработке технического задания участвуют как представители заказчика, так и представители исполнителя. В основе этого документа лежат исходные требования заказчика, анализ передовых достижений техники, результаты выполнения научно-исследовательских работ, предпроектных исследований, научного прогнозирования и т. п.

Разработка технического задания выполняется в следующей последовательности. Прежде всего, устанавливают набор выполняемых функций, а также перечень и характеристики исходных данных. Затем определяют перечень результатов, их характеристики и способы представления. Далее уточняют среду функционирования программного обеспечения: конкретную комплектацию и параметры технических средств, версию используемой операционной системы и, возможно, версии и параметры другого установленного программного обеспечения, с которым предстоит взаимодействовать будущему программному продукту.

В случаях, когда разрабатываемое программное обеспечение собирает и хранит некоторую информацию или включается в управление каким-либо техническим процессом, необходимо также четко регламентировать действия программы в случае сбоев оборудования и энергоснабжения.

На техническое задание существует стандарт ГОСТ 19.201-78 «Техническое задание. Требования к содержанию и оформлению». В соответствии с этим стандартом техническое задание должно содержать следующие разделы:

• введение;
• основания для разработки;
• назначение разработки;
• требования к программе или программному изделию;
• требования к программной документации;
• технико-экономические показатели;
• стадии и этапы разработки;
• порядок контроля и приемки.

При необходимости допускается в техническое задание включать приложения.

Рассмотрим более подробно содержание каждого раздела.

Введение должно включать наименование и краткую характеристику области применения программы или программного продукта, а также объекта (например, системы) в котором предполагается их использовать. Основное назначение введения - продемонстрировать актуальность данной разработки и показать, какое место эта разработка занимает в ряду подобных.

Раздел ***Основания для разработки*** должен содержать наименование документа, на основании которого ведется разработка, организации, утвердившей данный документ, и наименование или условное обозначение темы разработки. Таким документом может служить план, приказ, договор и т.п.

Раздел ***Назначение разработки*** должен содержать описание функционального и эксплуатационного назначения программного продукта с указанием категорий пользователей.

Раздел ***Требования к программе или программному изделию*** должен включать следующие подразделы:

• требования к функциональным характеристикам;
• требования к надежности;
• условия эксплуатации;
• требования к составу и параметрам технических средств;
• требования к информационной и программной совместимости;
• требования к маркировке и упаковке;
• требования к транспортированию и хранению;
• специальные требования.

Наиболее важным из перечисленных выше является

- подраздел ***Требования к функциональным характеристикам***. В этом разделе должны быть перечислены выполняемые функции и описаны состав, характеристики и формы представления исходных данных и результатов. В этом же разделе при необходимости указывают критерии эффективности: максимально допустимое время ответа системы, максимальный объем используемой оперативной и/или внешней памяти и др.

Примечание. Если разработанное программное обеспечение не будет выполнять указанных в техническом задании функций, то оно считается не соответствующим техническому заданию, т. е. неправильным с точки зрения критериев качества. Универсальность будущего продукта также обычно специально не оговаривается, но подразумевается.

- в подразделе ***Требования к надежности*** указывают уровень надежности, который должен быть обеспечен разрабатываемой системой и время восстановления системы после сбоя. Для систем с обычными требованиями к надежности в этом разделе иногда регламентируют действия разрабатываемого продукта по увеличению надежности результатов (контроль входной и выходной информации, создание резервных копий промежуточных результатов и т. п.).

- в подразделе ***Условия эксплуатации***, указывают особые требования к условиям эксплуатации: температуре окружающей среды, относительной влажности воздуха и т. п. Как правило, подобные требования формулируют, если разрабатываемая система будет эксплуатироваться в нестандартных условиях или использует специальные внешние устройства, например для хранения информации. Здесь же указывают вид обслуживания, необходимое количество и квалификация персонала. В противном случае допускается указывать, что требования не предъявляются.

- в подразделе ***Требования к составу и параметрам технических средств*** указывают необходимый состав технических средств с указанием их основных технических характеристик: тип микропроцессора, объем памяти, наличие внешних устройств и т. п. При этом часто указывают два варианта конфигурации: минимальный и рекомендуемый.

- в подразделе ***Требования к информационной и программной совместимости*** при необходимости можно задать методы решения, определить язык или среду программирования для разработки, а также используемую операционную систему и другие системные и пользовательские программные средства, с которым должно взаимодействовать разрабатываемое программное обеспечение. В этом же разделе при необходимости указывают, какую степень защиты информации необходимо предусмотреть.

- в разделе ***Требования к программной документации*** указывают необходимость наличия руководства программиста, руководства пользователя, руководства системного программиста, пояснительной записки и т. п. На все эти типы документов также существуют ГОСТы. Правила их составления будут рассмотрены позже.

- в разделе ***Технико-экономические показатели*** рекомендуется указывать ориентировочную экономическую эффективность, предполагаемую годовую потребность и экономические преимущества по сравнению с существующими аналогами.

- в разделе ***Стадии и этапы разработки*** указывают стадии разработки, этапы и содержание работ с указанием сроков разработки и исполнителей.

- в разделе ***Порядок контроля и приемки*** указывают виды испытаний и общие требования к приемке работы.

В приложениях при необходимости приводят: перечень научно-исследовательских работ, обосновывающих разработку; схемы алгоритмов, таблицы, описания, обоснования, расчеты и другие документы, которые следует использовать при разработке.

В зависимости от особенностей разрабатываемого продукта разрешается уточнять содержание разделов, т. е. использовать подразделы, вводить новые разделы или объединять их.

В случаях, если какие-либо требования, предусмотренные техническим заданием, заказчик не предъявляет, следует в соответствующем месте указать «Требования не предъявляются».

Разработка технического задания - процесс трудоемкий, требующий определенных навыков. Наиболее сложным, как правило, является четкое формулирование основных разделов: введения, назначения и требований к программному продукту. В качестве примеров рассмотрим два технических задания на выполнение курсового проектирования, составленных по сокращенной схеме, и сравнительно полное техническое задание на выполнение госбюджетной научно-исследовательской работы.

После утверждения технического задания организация-разработчик непосредственно приступает к созданию программного обеспечения. Однако переход к следующему этапу разработки - **этапу уточнения спецификаций** требует принятия еще некоторых принципиальных решений, от которых во многом зависят как характеристики и возможности разрабатываемого программного обеспечения, так и особенности его разработки, начиная с выбора моделей этапа уточнения спецификаций.

1.2.5 Анализ требований и определение спецификаций.

Спецификациями называют точное формализованное описание функций и ограничений разрабатываемого программного обеспечения. Соответственно ***различают функциональные и эксплуатационные спецификации.*** Совокупность спецификаций представляет собой общую логическую модель проектируемого программного обеспечения.

Для получения спецификаций выполняют анализ требований технического задания, формулируют содержательную постановку задачи, выбирают математический аппарат формализации, строят модель предметной области, определяют подзадачи и выбирают или разрабатывают методы их решения. Часть спецификаций может быть определена в процессе предпроектных исследований и, соответственно, зафиксирована в техническом задании.

Спецификации процессов обычно представляют в виде краткого текстового описания, схем алгоритмов, псевдокодов, Flow-форм или диаграмм Насси-Шнейдермана. Для краткости и понятности описания не только разработчику, но и заказчику, чаще всего используют псевдокоды.

Словарь терминов – краткое описание основных понятий, используемых при составлении спецификаций, который включает определение основных понятий предметной области, описание структур элементов данных, их типов и форматов, а также всех сокращений и условных обозначений. Предназначен для повышения степени понимания предметной области и исключения риска возникновения разногласий при обсуждении моделей между заказчиками и разработчиками. Описание термина в словаре выполняется по схеме:

«термин – категория– краткое описание».

На этом этапе также целесообразно сформировать тесты для поиска ошибок в проектируемом программном обеспечении, обязательно указав ожидаемые результаты.

1.3 Контрольные вопросы

1.3.1 Какие задачи решаются на основных этапах разработки?

1.3.2 Какова цель предпроектных исследований?

1.3.3 В чем сложность разработки технического задания? Какой из его разделов считается основным и почему?

1.3.4 На чем основывается выбор архитектуры программного обеспечения?

1.3.5 Чем руководствуются при выборе языка программирования?

1.3.6 Как подбирается подход к программированию?

1.3.7 Какие способы используются для описания алгоритмов?

1.3.8 Какие способы описания алгоритмов подходят для разработки структурных программ?

1.3.9 Что представляют собой спецификации ПО?

1.3.10 Какие методы существуют для уточнения спецификаций?

2 Расчетно-графическая работа №2. Реализация программы в выбранной среде программирования

Цель работы – получить практические навыки построения структурной и функциональной схем разрабатываемого программного продукта, проектирования пользовательского интерфейса и реализации программного продукта.

2.1 Задание на расчетно-графическую работу

Для спроектированного в предыдущей расчетно-графического работе программного продукта:

а) разработать структурную схему;

б) разработать функциональную схему;

в) спроектировать пользовательский интерфейс;

г) выполнить программную реализацию.

Рекомендации к выполнению задания. При выполнении реализации следует создать исполняемый файл с расширением *.exe. Все разработанные схемы и принятые решения разместить в пояснительной записке, снабдив их соответствующими комментариями.

2.2 Общие рекомендации к выполнению расчетно-графической работы

2.2.1 Разработка структурной и функциональной схем

Процесс проектирования сложного программного обеспечения начинают с уточнения его структуры, т. е. определения структурных компонентов и связей между ними. Результат уточнения структуры может быть представлен в виде структурной и/или функциональной схем и описания (спецификаций) компонентов.

Структурная схема разрабатываемого программного обеспечения. Структурной называют схему, отражающую состав и взаимодействие по управлению частей разрабатываемого программного обеспечения.

Структурные схемы пакетов программ не информативны, поскольку организация программ в пакеты не предусматривает передачи управления между ними. Поэтому структурные схемы разрабатывают для каждой программы пакета, а список программ пакета определяют, анализируя функции, указанные в техническом задании.

Самый простой вид программного обеспечения - программа, которая в качестве структурных компонентов может включать только подпрограммы и библиотеки ресурсов.

Разработку структурной схемы программы обычно выполняют методом пошаговой детализации.

Структурными компонентами программной системы или программного комплекса могут служить программы, подсистемы, базы данных, библиотеки ресурсов и т. п.

Рис. 2.1 - Пример структурной схемы программного комплекса

Структурная схема программного комплекса демонстрирует передачу управления от программы-диспетчера соответствующей программе (рисунок 2.1).

Структурная схема программной системы, как правило, показывает наличие подсистем или других структурных компонентов. В отличие от программного комплекса отдельные части (подсистемы) программной системы интенсивно обмениваются данными между собой и, возможно, с основной программой. Структурная же схема программной системы этого обычно не показывает (рисунок 2.2).

Рисунок 2.2 - Пример структурной схемы программной системы

Пример структурной схемы разработки алгоритма программы построения графиков функций одной переменной на заданном интервале изменения аргумента [x|, x2] при условии непрерывности функции на всем интервале определения приведен на рисунке 2.3.

Рисунок 2.3 - Структурная схема программы построения графиков/таблиц функций

Подробная многоуровневая структурная схема той же программы будет выглядеть следующим образом (рисунок 2.4):

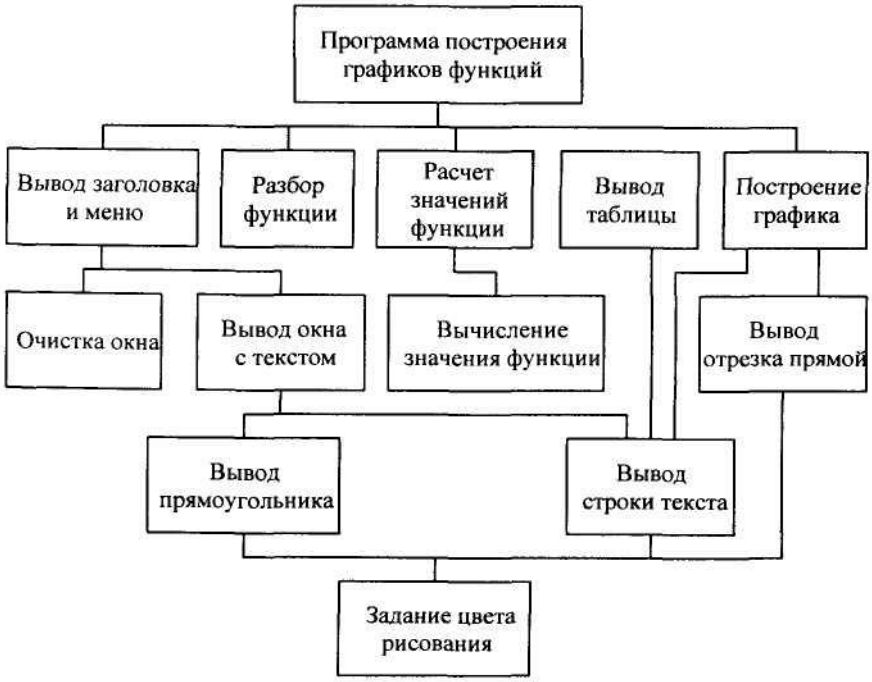

Рисунок 2.4 - Полная многоуровневая структурная схема программы построения графиков/таблиц

Более полное представление о проектируемом программном обеспечении с точки зрения взаимодействия его компонентов между собой и с внешней средой дает функциональная схема.

2.2.2 Выбор типа пользовательского интерфейса

Различают четыре типа пользовательских интерфейсов:

• *примитивные* - реализуют единственный сценарий работы, например, ввод данных - обработка - вывод результатов;

• *меню* - реализуют множество сценариев работы, операции которых организованы в иерархические структуры, например, «вставка»: «вставка файла», «вставка символа» и т. д.;

• *со свободной навигацией* - реализуют множество сценариев, операции которых не привязаны к уровням иерархии, и предполагают определение множества возможных операций на конкретном шаге работы; интерфейсы данной формы в основном используют Windows-приложения;

• *прямого манипулирования* - реализуют множество сценариев, представленных в операциях над объектами, основные операции инициируются перемещением пиктограмм объектов мышью, данная форма реализована в интерфейсе самой операционной системы Windows альтернативно интерфейсу со свободной навигацией.

Тип пользовательского интерфейса во многом определяет сложность и трудоемкость разработки, которые существенно возрастают в порядке перечисления типов. По последним данным до 80 % программного кода может реализовывать именно пользовательский интерфейс [49]. Поэтому понятно, что на ранних стадиях обучения программированию реализуют в основном примитивные интерфейсы и меню, хотя они и не удобны для пользователей.

Появление объектно-ориентированных визуальных сред разработки программного обеспечения, использующих событийный подход к программированию и в основном рассчитанных на создание интерфейсов со свободной навигацией, существенно снизило трудоемкость разработки подобных интерфейсов и упростило реализацию интерфейсов прямого манипулирования.

Таким образом, выбор двух последних типов интерфейсов предполагает использование одной из визуальных сред разработки программного обеспечения. Если соответствующие среды разработчику не доступны, то следует учитывать большую трудоемкость создания подобных интерфейсов.

Кроме того, выбор типа интерфейса включает выбор *технологии работы с документами.* Различают две технологии:

• однодокументная, которая предполагает однодокументный интерфейс (SDI - Single Document Interface);

• многодокументная, которая предполагает многодокументный интерфейс (MDI - Multiple Document Interface).

Многодокументную технологию используют, если программное обеспечение должно работать с несколькими документами одновременно, например, с несколькими текстами или несколькими изображениями. Однодоку-

ментную - если одновременная работа с несколькими документами не обязательна.

Трудоемкость реализации многодокументных интерфейсов с использованием современных библиотек примерно на 3...5 % выше, чем первого. Выбор типа влияет на трудоемкость более существенно (более подробно типы интерфейсов будут рассмотрены в § 8.1).

2.2.3 Выбор подхода к разработке

Если выбран интерфейс со свободной навигацией или прямого манипулирования, то, как указывалось выше, это практически однозначно предполагает использование событийного программирования и объектного подхода, так как современные среды визуального программирования, такие как Visual C++, Delphi, Builder C++ и им подобные, предоставляют интерфейсные компоненты именно в виде объектов библиотечных классов. При этом в зависимости от сложности предметной области программное обеспечение может реализовываться как с использованием объектов и, соответственно, классов, так и чисто процедурно. Исключение составляют случаи использования специализированных языков разработки Интернет-приложений, таких как Perl, построенных по совершенно другому принципу.

Примитивный интерфейс и интерфейс типа меню совместимы как со структурным, так и с объектным подходами к разработке. Поэтому выбор подхода осуществляют с использованием дополнительной информации.

Практика показывает, что объектный подход эффективен для разработки очень больших программных систем (более 100000 операторов универсального языка программирования) и в тех случаях, когда объектная структура предметной области ярко выражена.

Следует также учитывать, что необходимо осторожно использовать объектный подход при жестких ограничениях на эффективность разрабатываемого программного обеспечения, например, при разработке систем резального времени.

Во всех прочих случаях выбор подхода остается за разработчиком.

2.2.4 Выбор среды программирования

Средой программирования называют программный комплекс, который включает специализированный текстовый редактор, встроенные компилятор, компоновщик, отладчик, справочную систему и другие программы, использование которых упрощает процесс написания и отладки программ.

Последнее время широкое распространение получили упоминавшиеся выше среды визуального программирования, в которых программист получает возможность визуального подключения к программе некоторых кодов из специальных библиотек компонентов, что стало возможным с развитием объектно-ориентированного программирования.

Наиболее часто используемыми являются визуальные среды Delphi, C++ Builder фирмы Borland (Inprise Corporation), Visual **C++,** Visual Basic фирмы Microsoft, Visual Ada фирмы IBM и др.

Между основными визуальными средами этих фирм Delphi, C++ Builder и Visual C++ имеется существенное различие: визуальные среды фирмы

Microsoft обеспечивают более низкий уровень программирования «под Windows». Это является их достоинством и недостатком. Достоинством - так как уменьшается вероятность возникновения «нестандартной» ситуации, т. е. ситуации, не предусмотренной разработчиками библиотеки компонентов, а недостатком - так как это существенно загружает программиста «рутинной» работой, от которой избавлен программист, работающий с Delphi или C++ Builder. Много нареканий вызывает также интерфейс Visual C++, также ориентированный на «низкоуровневое» программирование.

В общем случае, если речь идет о выборе между этими средами, то он в значительной степени должен определяться характером проекта.

Рекомендации к выполнению задания. В пояснительную записку следует включить исходные тексты программ с указанием версии, иллюстрации окон интерфейса пользователя с пояснениями, таблицу тестирования. Справочные материалы могут быть встроены в программу или выполнены в виде отдельного прилагаемого файла. Текст справки следует привести в соответствующем разделе пояснительной записки. Дополнительная информация (об авторах, справочные данные по предметной области и т.д.) добавляется по желанию.

2.2.5 Пояснительная записка

Пояснительная записка должна содержать всю информацию, необходимую для сопровождения и модификации программного обеспечения: сведения о его структуре и конкретных компонентах, общее описание алгоритмов и их схемы, а также обоснование принятых технических и технико-экономических решений.

Содержание пояснительной записки по стандарту (ГОСТ 19.404-79) должно выглядеть следующим образом:

- введение;
- назначение и область применения;
- технические характеристики;
- ожидаемые технико-экономические показатели;
- источники, используемые при разработке.

В разделе Введение указывают наименование программы и документа, на основании которого ведется разработка.

В разделе Назначение и область применения указывают назначение программы и дают краткую характеристику области применения.

Раздел Технические характеристики должен содержать следующие подразделы:

- постановка задачи, описание применяемых математических методов и допущений и ограничений, связанных с выбранным математическим аппаратом;
- описание алгоритмов и функционирования программы с обоснованием принятых решений;

• описание и обоснование выбора способа организации входных и выходных данных;

• описание и обоснование выбора состава технических и программных средств на основании проведенных расчетов или анализов.

В разделе Ожидаемые технико-экономические показатели указывают технико-экономические показатели, обосновывающие преимущество выбранного варианта технического решения.

В разделе Источники, использованные при разработке, указывают перечень научно-технических публикаций, нормативно-технических документов и других научно-технических материалов, на которые есть ссылки в исходном тексте.

Пояснительная записка составляется профессионалами в области разработки программного обеспечения и для специалистов того же уровня квалификации. Следовательно, в ней уместно использовать специальную терминологию, ссылаться на специальную литературу и т. п.

2.2.6. Руководство пользователя

Как уже указывалось выше, в настоящее время часто используют еще один эксплуатационный документ, в который отчасти входит руководство системного программиста, программиста и оператора. Этот документ называют Руководством пользователя. Появление такого документа явилось следствием широкого распространения персональных компьютеров, работая на которых пользователи совмещают в своем лице трех указанных специалистов.

Составление документации для пользователей имеет свои особенности, связанные с тем, что пользователь, как правило, не является профессионалом в области разработки программного обеспечения. В книге С. Дж. Гримм даны рекомендации по написанию подобной программной документации:

- учитывайте интересы пользователей - руководство должно содержать все инструкции, необходимые пользователю;

- излагайте ясно, используйте короткие предложения;

- избегайте технического жаргона и узко специальной терминологии, если все же необходимо использовать некоторые термины, то их следует пояснить;

- будьте точны и рациональны - длинные и запутанные руководства обычно никто не читает, например, лучше привести рисунок формы, чем долго ее описывать.

Руководство пользователя, как правило, содержит следующие разделы:

- общие сведения о программном продукте;

- описание установки;

- описание запуска;

- инструкции по работе (или описание пользовательского интерфейса);

- сообщения пользователю.

Раздел Общие сведения о программе обычно содержит наименование программного продукта, краткое описание его функций, реализованных методов и возможных областей применения.

Раздел Установка обычно содержит подробное описание действий по установке программного продукта и сообщений, которые при этом могут быть получены.

В разделе Запуск, как правило, описаны действия по запуску программного продукта и сообщений, которые при этом могут быть получены.

Раздел Инструкции по работе обычно содержит описание режимов работы, форматов ввода-вывода информации и возможных настроек.

Раздел Сообщения пользователю должен содержать перечень возможных сообщений, описание их содержания и действий, которые необходимо предпринять по этим сообщениям.

2.2.7 . Руководство системного программиста

По ГОСТ 19.503-79 руководство системного программиста должно содержать всю информацию, необходимую для установки программного обеспечения, его настройки и проверки работоспособности. Кроме того, как указывалось выше, в него часто включают и описание необходимого обслуживания, которое раньше приводилось в руководстве оператора (ГОСТ 19.505-79) и/или руководстве по техническому обслуживанию (ГОСТ 19.508-79). В настоящее время данную схему используют для составления руководства системному администратору.

Руководство системного программиста должно содержать следующие разделы:

- общие сведения о программном продукте,
- структура,
- настройка,
- проверка,
- дополнительные возможности,
- сообщения системному программисту.

Раздел Общие сведения о программе должен включать описание назначения и функций программы, а также сведения о технических и программных средствах, обеспечивающих выполнение данной программы (например, объем оперативной памяти, требования к составу и параметрам внешних устройств, требования к программному обеспечению и т. п.).

В разделе Структура программы должны быть приведены сведения о структуре программы, ее составных частях, о связях между составными частями и о связях с другими программами.

В разделе Настройка программы должно быть приведено описание действий по настройке программы на условия практического применения.

В разделе Проверка программы должно быть приведено описание способов проверки работоспособности программы, например контрольные примеры.

В разделе Дополнительные возможности должно быть приведено описание дополнительных возможностей программы и способов доступа к ним.

В разделе Сообщения системному программисту должны быть указаны тексты сообщений, выдаваемых в ходе выполнения настройки и проверки

программы, а также в ходе ее выполнения, описание их содержания и действий, которые необходимо предпринять по этим сообщениям.

2.2.8. Основные правила оформления программной документации

При оформлении текстовых и графических материалов, входящих в программную документацию следует придерживаться действующих стандартов.

2.3 Контрольные вопросы

2.3.1 Что представляет собой структурная схема ПО?
2.3.2 В чем особенность функциональной схемы?
2.3.3. Как подбирается тип интерфейса пользователя?
2.3.4 Какие особенности человека следует учитывать при разработке пользовательских интерфейсов?
2.3.5 Какие особенности у интерфейса прямого манипулирования??
2.3.6 Каковы особенности интерфейса со свободной навигацией?
2.3.7 Как правильно подобрать среду программирования?
2.3.8 С какими сложностями сталкиваются при выбора подхода к разработке?
2.3.9 В каких случаях используют примитивные интерфейсы?
2.3.10 Как строятся функциональные схемы?

Список литературы

1. Орлов С.А., Технология разработки программного обеспечения. Учебник. - СПб: Питер, 2002.
2. С.В. Маклаков BPWin, и ERWin. CASE-разработки информационных систем. - М.:ДИАЛОГ-МИФИ, 2000 - 256 с.
3. Методология структурного анализа и проектирования SADT. Дэвид А. Марка и Клемент Мак Гоуэн. www.vernikov.
4. Синтес Антони, Освой самостоятельно объектно-ориентированное программирование за 21 день.: Пер. с англ. - М.: «Вильяме», 2002.
5. Грейди Буч, Джеймс Рамбо, Айвар Джекобсон, Язык UML. Руководство пользователя: Пер. с англ - М.: ДМК Пресс, 2001.
6. Зиндер Е.З. Бизнес-реинжиниринг и технологии системного проектирования. Учебное пособие. М., Центр Информационных Технологий, 1996.
7. Марка Д.А., Мак Гоуэн К. Методология структурного анализа и проектирования. М., "МетаТехнология", 1993.
8. Вендров А.М. Один из подходов к выбору средств проектирования баз данных и приложений. "СУБД", 1995, №3.
9. Калянов Г.Н. CASE. Структурный системный анализ (автоматизация и применение). М., "Лори", 1996.

10. Международные стандарты, поддерживающие жизненный цикл программных средств. М., МП "Экономика", 1996

11. Создание информационной системы предприятия. "Computer Direct", 1996, N2

12. Шлеер С., Меллор С. Объектно-ориентированный анализ: моделирование мира в состояниях. Киев, "Диалектика", 1993.

13. Barker R. CASE*Method. Entity-Relationship Modelling. Copyright Oracle Corporation UK Limited, Addison-Wesley Publishing Co., 1990.

14. Горчинская О.Ю. Designer/2000 - новое поколение CASE- продуктов фирмы ORACLE. "СУБД", 1995, №3.

15. Горин С.В., Тандоев А.Ю. Применение CASE-средства Erwin 2.0 для информационного моделирования в системах обработки данных. "СУБД", 1995, №3.

16. Горин С.В., Тандоев А.Ю. CASE-средство S-Designor 4.2 для разработки структуры базы данных. "СУБД", 1996, №1.