Министерство образования и науки Российской Федерации

Федеральное агентство по образованию РФ

Владивостокский государственный университет
экономики и сервиса

ТЕХНОЛОГИЯ ПРОГРАММИРОВАНИЯ

Руководство к выполнению курсовой работы
по специальностям
23010165 «Вычислительные машины, комплексы и сети»
23020165 «Информационные системы и технологии»

Владивосток
Издательство ВГУЭС
2009

Руководство к выполнению курсовой работы по дисциплине «Технология программирования» составлено в соответствии с требованиями государственного образовательного стандарта высшего профессионального образования РФ. Содержит организационно-методические указания и рекомендации для выполнения курсовой работы.

Предназначено студентам по специальностям 23010165 «Вычислительные машины, комплексы и сети» и 23020165 «Информационные системы и технологии».

Составитель: Б.К. Васильев, доцент кафедры ИСКТ.

Утверждено на заседании кафедры ИСКТ.

Рекомендовано к изданию УМК ИИБС.

ЗАДАЧИ КУРСОВОЙ РАБОТЫ

Курсовая работа является одним из основных видов самостоятельной работы студентов в вузе, направленной на изучение, закрепление, углубление и обобщение знаний по учебным дисциплинам профессиональной подготовки, освоение элементов научно-исследовательской работы, и может быть рекомендована к участию в научных конференциях, конкурсах студенческих и профессиональных работ и даже служить основой дипломной работы. По данной дисциплине курсовая работа является коллективным трудом, выполняемым студентами под руководством преподавателя.

Объем курсовой работы зависит от размера коллектива разработчиков, сложности задачи и ее характера. Типовой размер пояснительной записки для коллектива из 4 разработчиков – 60 страниц, программного текста 1000-1500 строк исходного текста, не включая систему помощи.

Основной задачей курсовой работы по дисциплине «Технология программирования» является закрепление знаний по управлению процессом разработки программного обеспечения (ПО), приобретение навыков работы в команде, ведении записей при аттестациях, тестировании и верификации ПО.

Курсовая работа состоит из двух основных процессов – проектирования и исследования, проводящихся одновременно и выполняющихся командой разработчиков.

Целью ***проектирования*** является разработка программного продукта, представляющего законченное средство для решения задач некоторой предметной области. Требования к разрабатываемому продукту описаны в следующем разделе. В результате выполнения проекта может сложиться ситуация, когда выявляются непредвиденные обстоятельства и особенности реализации задачи, приводящие к срыву графика и невозможности выполнения работы в срок полностью. В этом случае проект закрывается в установленные сроки как незавершенный.

Целью ***исследования*** является анализ процесса разработки, реализация и описание какого-либо из сопутствующих разработке подчиненных процессов, – управление качеством, управление временем, управление рисками и т.д. В случае отрицательного результат выполнения проекта в исследовании подробно описываются и анализируются причины, приведшие к краху проекта. Обоснование неудачи должно быть доказательным.

Студенты имеют возможность формировать творческий коллектив для выполнения курсовой работы и определять распределение обязанностей в проекте самостоятельно. Размер коллектива определяется возможностью решения поставленных задач, объемом выполняемого исследования, рассчитанного на выполнение в течение семестра.

Задачи определяются тематикой, предложенной ведущим преподавателем, сотрудниками кафедры или сторонними лицами, именуемыми далее заказчиками. Темы курсовых работ должны предоставлять возможность реализации всех требований, перечисленных далее. На основе выбранной темы совместно членами команды и руководителем разрабатываются техническое задание и график выполнения работ.

ТРЕБОВАНИЯ, ПРЕДЪЯВЛЯЕМЫЕ К РАЗРАБАТЫВАЕМОМУ ПО

Программный проект, реализуемый в рамках курсовой работы должен удовлетворять определенным требованиям. Во-первых, этот проект должен удовлетворять всем традиционным свойствам проекта, быть реализуемым в отведенные сроки, во-вторых, он должен иметь определенную практическую или теоретическую ценность и степень новизны, преимущественно реализационного характера. См. далее раздел «Темы».

Программное обеспечение, создаваемое при выполнении проекта, в свою очередь, должно удовлетворять ряду требований, подробно рассматриваемых далее.

К обязательным требованиям относятся следующие:

- целостность программного продукта;
- переносимость (мобильность);
- наличие графического пользовательского интерфейса с событийно-управляемыми элементами;
- наличие документации, состав которой согласовывается с заказчиком;
- должен выполняться коллективно при непосредственном участии всех членов коллектива (бригады программистов).

Процессы проектирования и разработки, а также процессы выполнения тестирования и опытной эксплуатации должны быть описаны в пояснительной записке, где также должны быть представлены графики выполнения работ с указанием сроков расчетных, фактических и объяснением причин отклонения от графика.

Целостность

Программный продукт должен быть реализован таким образом, чтобы его можно было выполнять и тестировать без использования дополнительных библиотек и иных вспомогательных средств. Средства внутренней диагностики должны быть «вкомпилированы» в ПО.

Желательно выполнять статическую компоновку всех модулей и программных компонентов проекта.

Мобильность

Для обеспечения переносимости ПО необходимо использовать только те внешние компоненты (элементы управления, библиотечные функции, протоколы), которые являются общими для всех операционных систем и платформ (Linux, Windows32, UNIX, I-586, RISC).

В языковых конструкциях не должны применяться выражения, значения которых зависят от длины машинного слова и порядка следования байта в слове, наличия или отсутствия выравнивания компонентов структур и записей по границам слов в памяти [9].

Наличие графического интерфейса

Для работы пользователя с программой предусматривается графический пользовательский интерфейс (GUI), за исключением проектов, в которых отсутствие GUI оговорено отдельно. Реализация GUI не должна нарушать требования мобильности.

Рекомендуемыми средствами для создания мобильного GUI являются FLTK[7], QT, gtk+ и другие библиотеки, работа которых гарантируется в ОС Windows и Linux.

Графический интерфейс пользователя, создаваемый для проекта, должен быть интуитивно понятен, иметь возможность управления как с использованием графического указателя (мыши), так и клавиатуры. В состав GUI желательно включить средства помощи (контекстно-зависимой и общей).

Интерфейсная часть организует лишь ввод данных и визуализацию полученных результатов, расчетную же часть лучше всего реализовывать в виде отдельного модуля (или модулей). Интерфейсную часть программы лучше разделить на два блока. Первый блок должен осуществлять ввод исходных данных, а второй блок должен осуществлять визуализацию полученных результатов.

Программа может быть написана с использованием объектно-ориентированного подхода (ООП) [6,9] или использовать методы структурного программирования [5,10,12], но графический интерфейс предпочтительнее всего реализовывать с помощью ООП.

Документация

Обязательной документацией (бумажной) является техническое задание и график выполнения работ.

Состав дополнительной документации (безбумажной) определяется для каждого проекта индивидуально и может включать:

- описание программы;
- руководство программиста;
- руководство по эксплуатации (руководство оператора);
- руководство по сопровождению (описание средств внутренней отладки);
- описание процесса разработки, используемых моделей жизненного цикла [12,16];
- статические и динамические обзоры ПО в ходе разработки[16].

При оформлении документации (она оформляется в виде приложения к пояснительной записке) необходимо следовать рекомендациям внутривузовского стандарта [15].

Коллективная работа над проектом

Поскольку работа над КР выполняется коллективно, необходимо на стадии проектирования произвести распределение обязанностей между членами бригады и разработать графики выполнения работ исполнителями. Команда разработчиков ПО начинает, исследует и развивает проект продукта [12,16].

Лидером команды является менеджер проекта программного продукта (в бригаде курсового проекта таким человеком должен быть студент непререкаемым авторитетом). Эта команда состоит из специалистов, которые занимаются разработкой совместно:

- инженеров-разработчиков (специалистов по инженерии программирования и программистов);
- технических писателей;
- дизайнеров;
- инженеров тестирования;
- инженеров качества;
- архитекторов программных проектов и интеграторов.

В небольшом проекте приходится совмещать нескольких специалистов в лице одного исполнителя. Команда полностью проектирует и разрабатывает продукт. Обычно команда разработчиков продукта включает в себя несколько команд разработчиков компонентов, но для малого коллектива это неприменимо.

Очень важным моментов в планировании проектных работ является распределение обязанностей между членами рабочего коллектива. Это распределение определяет график выполнения работ и описывается в пояснительной записке. Графики выполнения составляются с учетом графа зависимостей выполняемых работ и распределения обязанностей в коллективе разработчиков.

Временные рамки и графики работ

Как всякий проект, программная часть курсовой работы ограничена строгими временными рамками, начало проекта совпадает с моментом получения темы (первая неделя семестра), время окончания – зачетная неделя.

Для соблюдения временных рамок проекта во время его выполнения предполагается управление временем (при включении в техническое задание соответствующего пункта).

В соответствии с разработанными графиками работ производится расчет времени выполнения проекта и, в случае необходимости, производится корректировка технического задания (обязательно согласование с руководителем!). Расчет времени может производиться на основе лабораторной работы, предусмотренной в курсе дисциплины [7].

Пояснительная записка

Пояснительная записка (ПЗ) является частью отчета по курсовой работе и должна содержать описание процесса разработки, архитектуру разработанного программного средства, описание средств управления проектом в соответствии с ТЗ.

Состав пояснительной записки

– Титульный лист
– Содержание
– Введение – постановка задачи
– Выбор средств и обоснование выбора
– Распределение обязанностей в группе исполнителей и описание процесса совместной разработки проекта

Описание работы:

– Распределение обязанностей в коллективе
– Описание применяемых алгоритмов, обоснование их выбора, анализ временной сложности алгоритма (в случае необходимости)
– Описание архитектура приложения (включая связи между функциями или модулями, с описанием типов связей)
– Описание структуры и функций графического интерфейса
– Выводы
– Список литературы и электронных использованных ресурсов
– Приложения

Титульный лист является первой страницей ПЗ и должен содержать следующие сведения: наименование учебного заведения (ВГУЭС), тему КР, сведения об исполнителях работы, сведения о руководителе (преподавателе), наименование места и год выполнения. Образец титульного листа приведен в Приложении 1. Не забудьте заменить XX, YY, ZZ на реальные цифры.

Содержание включает перечень основных элементов структуры ПЗ с указанием номеров страниц, с которых начинается их месторасположение или гиперссылки на них. Для сборки содержания можно воспользоваться встроенными средствами редактора.

Во ***введении*** описывается постановка задачи, делаются ссылки на прототипы и указывается место разработки в предметной области и в области программирования.

Описание работы должно содержать текстовые материалы и числовые данные, отражающие существо, методику и отдельные результаты, достигнутые в ходе выполнения курсовой работы. Материал основной части рекомендуется делить на разделы и подразделы. При этом каждый раздел должен содержать законченную информацию, логически вписывающуюся в общую структуру ПЗ и способствующую достижению целей работы.

Первой частью обычно должен являться раздел, посвященный формированию творческого коллектива и ***распределению обязанностей***. Описываются обязанности каждого из членов бригады, обосновывается соответствующий выбор, приводится перечень назначаемых видов деятельности и список выполняемых работ.

В ***описании применяемых алгоритмов*** должны освещаться следующие вопросы:

- общая постановка задачи;
- математическая постановка задачи;
- ссылки на теоретический материал, использованный для решения поставленной задачи;
- методические аспекты решения поставленной задачи;
- оценка временной сложности алгоритма, описание вычислительных экспериментов по измерению времени работы, параметры исходных данных, расход памяти, исследуемые зависимости и т.д.

Описание архитектура приложения содержит интерфейсы вызова всех функций проекта, входных и выходных параметров, если их смысл неочевиден, связи между функциями или модулями, с описанием типов связей, взаимозависимостей модулей (граф вызовов). Пример описания функций приведен в прил. 5.

Желательно привести структурную схему разрабатываемого приложения.

В этой же части рассматриваются требования к тестированию созданного во время выполнения работы программному обеспечению; методы тестирования на различных стадиях проектирования, наборы тестов.

Описание структуры и функций графического интерфейса выполняется для описания назначения элементов управления, структуры меню и обработчиков событий, использованных для этих элементов.

Выводы характеризуют итоги проделанной работы, предложения и рекомендации по продолжению исследований.

Список литературы – это упорядоченный в алфавитно-хронологической последовательности или а порядке цитирования в тексте перечень библиографических описаний документальных источников информации по теме выполненной работы. В списке следует указывать автора, наименование источника, издательство, год издания. Примером

правильно оформленного списка может служить выполненный в алфавитном порядке список литературы к настоящему документу.

Все работы, включенные в список, должны иметь в тексте ссылки в виде номера элемента списка, заключенного в квадратные скобки. В случае необходимости можно сослаться на несколько источников, при этом в квадратных скобках приводится список номеров, разделенных запятыми.

В ***приложения*** выносится исходный текст выполненных программ, объемные таблицы и графические материалы численных экспериментов, дополнительная документация (по выбору одно из рекомендованных приложений – руководство оператора или руководство по сопровождению или руководство системного программиста),

Оформление пояснительной записки

Пояснительная записка может быть выполнена в следующих вариантах:

1. В стандартном редакторе OpenOffice (версии 2.3 и выше), без использования макросов. Формулы и графики реализуются средствами ОО.

2. В редакторе MS Word (2000 или другой версии, поддерживающейся в ОО). При этом не допускается включение макросов и OLE-объектов. Форматирование должно быть выполнено таким образом, чтобы текст корректно отображался после импорта в OpenOffice.

3. В формате гипертекстовой разметки html (полученным не средством экспорта из MSWord), в любом из доступных редакторов html.

Пояснительная записка не печатается в бумажном варианте, а предоставляется в виде файла. В состав бумажных документов входит только техническое задание, составленное в двух экземплярах и графики выполнения работ.

Для оформления ПЗ должно быть предусмотрено время в графике работ.

Оформление приложений

Тексты программ, оформляются в виде приложений к ПЗ и представляют файлы, пригодные для компиляции. Тем не менее, они должны соответствовать требованию пригодности для сопровождения. В тексте программ обязательно наличие форматирования и необходимых для понимания комментариев. При добавлении комментария использовать принцип «среднего инженера». Это означает, что конструкции языка должны быть понятны любому инженеру со средней квалификацией. Все используемые объекты и операторы должны иметь пояснения, если

только выполняемые действия или название объектов не являются понятными без комментариев.

Рассмотрим примеры типичных ошибок, допускаемых студентами в тексте программ (Приложение 3, все строки текста пронумерованы для удобства описания, далее указаны номера строк с ошибками).

1. Отсутствует форматирование с использованием отступов, облегчающее чтение (повсеместно);
2. Наличие неоправданных комментариев (строки 20, 24, 32);
3. Неоправданное наличие пустых строк (48-51);
4. Англоязычный комментарий (1-2, 55-56);
5. Отсутствие комментариев в существенных операторах (58-83);
6. Наличие длинных строк (45);
7. Присутствие в тексте «магических» констант (45,67, 70, 71, 77, 78, 83);
8. Зависимость от одной ОС (4, 25-31).

Пример правильно оформленной программы приведен в приложении 4. Отметим позитивные моменты в приведенном фрагменте:

1. Объяснение определяемых констант (1-5);
2. Описание назначения переменных (6-7, 17, 25-29, 31-32);
3. Описание назначения функций (56-59);
4. Осмысленные названия переменных (14-16, 19-29, 66-67);
5. Отсутствие комментариев для очевидных операций (38-54, 76-84);
6. Англоязычный вывод в потоки stdout и stderr (72-73);
7. Длинные строки разбиты на несколько для удобства обозрения в редакторе (9-11,34-35,61-62, 69-70);
8. Использование отступов – (хотя и не всегда соблюдается);
9. Использование пустых строк в качестве комментариев (см. рекомендации Алена Голуба [9]).

В книге Спинеллиса [14] приводится множество других примеров, описываются как удобочитаемые конструкции, так и типичные прегрешения разработчиков, не задумывающихся о тех, кто будет использовать код повторно.

ПРОГРАММНЫЙ ПРОЕКТ

Собственно говоря, как и любой другой проект, программный проект (ПП) имеет определенные рамки, прежде всего это рамки – временные. Срок выполнения проекта можно определить, если произвести анализ необходимых работ и произвести разбиение проекта на законченные отдельные виды деятельности. К ним относятся не только алгоритмизация и кодирование, но и освоение новых программных продуктов, приобретение необходимых программных и аппаратных средств, тестирование и верификация.

Конечной целью ПП в курсовой работе является реализация ПО в соответствии с требованиями технического задания (ТЗ), (подробно см. следующий раздел). Используемая модель реализации представляет совокупность взаимодействующих как последовательно, так и конкурентно выполняющихся процессов. Это процессы начального запуска работ (инициализация), процесс планирования, собственно реализации частей проекта (выполнение), процесс управления и этап завершения – интеграционное тестирование, оформление документации, защита работы.

Каждый вид деятельности в ПП характеризуется временной оценкой (сроком выполнения), одни виды деятельности не могут быть начаты прежде чем будут выполнены другие (существует зависимость видов деятельности). Некоторые виды деятельности могут выполняться параллельно различными людьми (конкурентно), а некоторые – нет.

ТЕХНИЧЕСКОЕ ЗАДАНИЕ

Для успешного выполнения проекта, все требования и спецификации разрабатываемого ПО должны быть формализованы и отражены в техническом задании. Особенности реализации, дополнительные (по отношению к приведенным обязательным) требования заказчика, особые условия разработки, тестирования и эксплуатации также включаются в ТЗ.

Разработка ТЗ – составная часть работы, выполняемая на раннем этапе проекта, необходимо предусматривать время на выработку системных требований, согласование всех дополнительных пунктов с заказчиком. Эту работу необходимо предусматривать в проекте и включать в графики выполнения работ. Результатом разработки ТЗ является документ, подписанный всеми исполнителями и заказчиком. В том случае, если заказчик и руководитель – не одно и то же лицо, требуется виза руководителя.

Проект, выполняемый без надлежащего оформления ТЗ, не допускается к защите. Включенные в ТЗ пункты, требующие дополнительного согласования с руководителем, вписываются в ТЗ и заверяются подписями исполнителя и руководителя в указанные в исходном тексте сроки. Пример правильно составленного ТЗ приведен в прил. 2.

МОДЕЛЬ ПРОЦЕССА РАЗРАБОТКИ

Моделирование

Моделирование — метод исследования систем на основе переноса изучаемых свойств системы на объекты другой природы. Это один из основных методов исследования окружающей действительности. Инженерная методика изготовления моделей является устоявшейся и повсеместно принятой. Она позволяет решить несколько важных задач:

- визуализировать систему;
- определить структуру системы и ее поведение;
- документировать принимаемые решения.

Наконец, моделирование — это попытка решить проблему сложности.

Существует четыре основных принципа моделирования [6].

- Выбор модели оказывает определяющее влияние на подход к решению проблемы и на то, как будет выглядеть это решение.
- Каждая модель может быть воплощена с разной степенью абстракции.
- Лучшими моделями являются те, которые ближе к реальности.
- Следует использовать совокупность нескольких моделей.

В общем случае выделяют четыре типа моделей:

- *Математические модели,* представляющие собой систему математических уравнений, адекватно описывающих изучаемое явление или объект.
- *Физические модели,* основанные на использовании эффекта масштаба в случае возможности пропорционального применения всего комплекса изучаемых свойств.
- *Ситуационные модели,* представляющие собой описание ситуаций, в которых предстоит действовать изучаемому объекту.
- *Электрические модели,* позволяющие построить электрическую цепь, эквивалентную любому дифференциальному уравнению.

Очевидно, что в данной курсовой работе можно использовать только математические и ситуационные модели, первые – для описания процессов предметной области, вторые для описания процесса разработки [6,13]. Еще раз отметим, что выбор адекватных моделей процессов предметной области и жизненного цикла может определить успешную реализацию проекта.

Модель жизненного цикла разработки

Жизненный цикл программного обеспечения — весь период его разработки и эксплуатации, начиная с момента возникновения замысла и заканчивая прекращением всех видов его использования.

Существует простейшее представление жизненного цикла, которое включает следующие стадии:

- анализ;
- проектирование (кодирование);
- программирование;
- тестирование и отладка;
- эксплуатация.

Цикл разработки программного обеспечения тесно связан с технологиями программирования, поэтому должен рассматриваться без отрыва от технологий разработки.

Для небольших проектов, в которых существует хорошо определенные спецификации, не предполагается итерационного совершенствования продукта, известны необходимые этапы и содержание выполняемых работ, можно применять простейшую – каскадную модель. Для большинства курсовых проектов эта модель подходит.

Очень часто каскадный жизненный цикл называют каскадной или водопадной или классической моделью, подчеркивая, что разработка рассматривается как последовательность этапов, причем переход на следующий этап происходит только после полного завершения работ на текущем этапе.

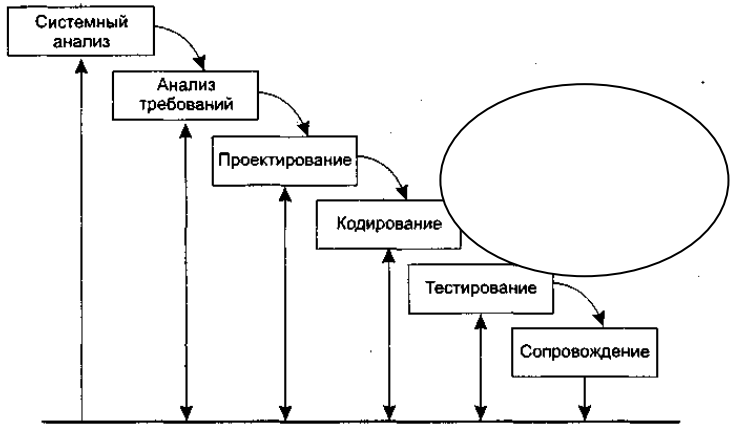
Охарактеризуем содержание основных этапов.

Подразумевается, что разработка начинается на системном уровне и проходит через анализ, проектирование, кодирование, тестирование и сопровождение. При этом моделируются действия стандартного инженерного цикла.

Системный анализ задает роль каждого элемента в компьютерной системе, взаимодействие элементов друг с другом. Поскольку ПО является лишь частью большой системы, то анализ начинается с определения требований ко всем системным элементам и назначения подмножества этих требований программному «элементу». Необходимость системного подхода явно проявляется, когда формируется интерфейс ПО с другими элементами (аппаратурой, людьми, базами данных). На этом же этапе начинается решение задачи планирования проекта ПО. В ходе планирования проекта определяются объем проектных работ и их риск, необходимые трудозатраты, формируются рабочие задачи и план-график работ.

Анализ требований относится к программному элементу — программному обеспечению. Уточняются и детализируются его функции, характеристики и интерфейс.

Все определения документируются в *спецификации анализа.* Здесь же завершается решение задачи планирования проекта.



Классический жизненный цикл разработки ПО [12]

Проектирование состоит в создании представлений:

– архитектуры ПО;
– модульной структуры ПО;
– алгоритмической структуры ПО;
– структуры данных;
– входного и выходного интерфейса (входных и выходных форм данных).

Исходные данные для проектирования содержатся в *спецификации анализа,* то есть в ходе проектирования выполняется трансляция требований к ПО во множество проектных представлений. При решении задач проектирования основное внимание уделяется качеству будущего программного продукта.

Кодирование состоит в переводе результатов проектирования в текст на языке программирования.

Тестирование — выполнение программы для выявления дефектов в функциях, логике и форме реализации программного продукта.

Сопровождение — это внесение изменений в ходе эксплуатации ПО с целью устранения ошибок, совершенствования кода, улучшения качественных или количественных характеристик. В курсовой работе этап сопровождения отсутствует, но его необходимо учитывать при составлении документации и в реализации проекта.

В случае использования в курсовой работе других моделей [16], необходимо их использование обосновать.

Рекомендуемая модель процессов на всех этапах разработки приведена в прил. 6. Определите состав процессов инициализации, выполнения и управления для своего проекта самостоятельно.

ВЫБОР СРЕДСТВ РЕАЛИЗАЦИИ

Компиляторы и среды разработки выбираются исходя из специфики задачи, решаемой в проекте и их выбор делается разработчиками (за исключением случаев, когда иное оговорено в техническом задании).

Рекомендуемыми компиляторами являются gcc (для языков C, C++ и оболочки kdevelop, geany) и их портированные версии в windows (MinGW, cygwin).

Для языка Pascal рекомендуемым компилятором является Free pascal (delphy-совместимый), а оболочкой – Lazarus.

Не рекомендуется использовать такие системы разработки интерфейсов, которые являются платформенно-зависимыми, например, C++ Visual Studio (компании Microsoft), C++ Workshop Visual (компании Sun), Delphi Suite (компании Borland Inc.). См. п. 2.3 для рекомендуемых средств.

Средства управления процессом разработки выбираются разработчиками самостоятельно.

Готовые повторно используемые компоненты можно использовать без ограничений, основные ресурса для их поиска – сайты свободно распространяемого программного обеспечения в исходных текстах:

http://www.freshmeat.net

http://www.sourceforge.net

http://www.gnu.org.

Многие ресурсы могут быть найдены в локальной сети ВГУЭС [1].

Примеры реализации

Многочисленные примеры реализации курсовых работ (как удачные, так и неудачные) можно посмотреть в локальной сети ВГУЭС по адресу ftp://bkv.vvsu.ru/pub/Kurs2kx.

ПРЕДМЕТНАЯ ОБЛАСТЬ

Задания на курсовую работу могут представлять задачи из самых разнообразных предметных областей, наиболее часто используемые предметные области, –

– Численные методы анализа;
– Графические редакторы (специализированные);
– Приложения с синтаксическими парсерами;
– Решение дифференциальных и интегральных уравнений (например, из области математической физики) имитационным моделированием;
– Геоинформационные системы;
– Обратные задачи математической физики;
– Обработка результатов научных экспериментов;
– Графические интерфейсы к эмуляторам процессоров;
– Дискретная математика.

Знакомство с предметной областью, ее терминологией и описанием процессов и моделей предметной области входит в исследовательскую часть курсовой работы. Для выполнения такого ознакомления необходимо предусматривать в графике выполнения работ соответствующее время.

Источники для знакомства с предметными областями можно получить в локальной сети ВГУЭС [1,2] или в Интернете, например, на сайте [3]. Для получения консультаций обратитесь к руководителю, возможно назначение консультанта по предметной области из сторонних организаций.

ТЕМЫ

Примерные темы курсовых работ с краткой аннотацией, темы, предлагавшиеся в 2000–2007 гг., отмечены как «повторные». Повторные темы назначаются только при измененных требованиях к реализации проекта (язык программирования, технология разработки или состав компонентов).

– Морфологический анализ названия органического химического соединения (по ИЮПАК) и получение из него структурной формулы. Количество исполнителей – 3, предметная область – органическая химия, нет хороших реализаций, наличие открытых исходных текстов – неизвестно, первичная тема.

– Моделирование работы произвольных логических схем. Количество исполнителей – 3, предметная область – цифровая схемотехника, есть неплохие реализации, наличие открытых исходных текстов – есть, повторная тема.

– Компилятор с упрощенной версии языка Си для микропроцессоров или микроконтроллеров (например, MCS-51). Количество исполнителей – 4, предметные области – цифровая схемотехника и конструирование компиляторов, есть неплохие реализации, наличие открытых исходных текстов – есть, повторная тема.

– Генерация блок-схем из текста программы на языке Си. Количество исполнителей – 4, предметная область – программирование, нет реализации, наличие открытых исходных текстов – неизвестно, повторная тема.

– Визуализация произвольной функции двух переменных. Количество исполнителей – 3, предметная область – теория функций действительных переменных, есть неплохие реализации, наличие открытых исходных текстов – есть, повторная тема.

– Конструктор молекул (шаростержневая модель). Количество исполнителей – 4, предметная область – химия и компьютерная графика, есть неплохие реализации, наличие открытых исходных текстов – есть, повторная тема.

– Двухмерная экологическая модель. Количество исполнителей – 3, предметная область – моделирование и дифференциальные уравнения, есть неплохие реализации, наличие открытых исходных текстов – есть, повторная тема.

– Нахождение мест возможной оптимизации в исходном тексте программы. Количество исполнителей – 4, предметная область – программирование, реализации неизвестны, наличие открытых исходных текстов – неизвестно, первичная тема.

– Представление цифрового рельефа на триангуляционной сетке. Количество исполнителей – 4, предметная область – компьютерная графика и геоинформационные системы, есть реализации, наличие открытых исходных текстов – неизвестно, повторная тема.

– Аппроксимация поверхности сплайнами. Иллюстрация к бикубическим сплайнам. Количество исполнителей – 4, предметная область – компьютерная графика, есть реализации, наличие открытых исходных текстов – есть, первичная тема.

– Реализация нестандартных элементов управления. Кнопки произвольной формы, трехмерные кнопки, валюаторы. Количество исполнителей – 4, предметная область – графические пользовательские интерфейсы, нет реализаций, наличие открытых исходных текстов – нет, первичная тема.

– Решение интегрального уравнения Фредгольма 1-го рода с симметричным ядром. Количество исполнителей – 3, предметная область – численные метода математического анализа. Нет хорошей реализации, наличие открытых исходных текстов – есть, повторная тема.

– Матричный калькулятор. Количество исполнителей – 4, предметная область –конструирование интерфейсов и компиляторов, есть неплохие реализации, наличие открытых исходных текстов – есть, повторная тема.

– Множества. Реализация абстрактного типа данных в языке Си для работы с множествами с графической иллюстрацией. Количество исполнителей – 3, предметная область – программирование, реализации неизвестны, наличие открытых исходных текстов – неизвестны, первичная тема.

– Визитка. Декларативный язык описания визитной карточки и его реализация для вывода на языке PCL5. Количество исполнителей – 4, предметная область – конструирование компиляторов, реализации неизвестны, наличие открытых исходных текстов – неизвестны, повторная тема.

– Перенос в русском языке. Количество исполнителей – 3, предметная область – лингвистика, есть неплохие реализации, наличие открытых исходных текстов – есть, повторная тема.

– Аффинное преобразование растровых изображений. Количество исполнителей – 3, предметная область – компьютерная графика, есть неплохие реализации, наличие открытых исходных текстов – неизвестны, повторная тема.

– Графическое представление абстрактного типа данных – двоичных деревьев. Количество исполнителей – 3, предметная область – программирование, есть реализации, наличие открытых исходных текстов – есть, повторная тема.

– Изображение шаростержневой модели молекулы средствами OpenGL. Количество исполнителей – 3, предметная область – химия и компьютерная графики, есть неплохие реализации, наличие открытых исходных текстов – есть, повторная тема.

– Пятимерный куб. Количество исполнителей – 2, предметная область – линейная алгебра и компьютерная графика, есть неплохие реализации, наличие открытых исходных текстов – есть, повторная тема.

– Многомерный курсор. Количество исполнителей – 3, предметная область –компьютерная графика, нет хорошей реализации, наличие открытых исходных текстов – нет, повторная тема.

– Сравнение бинарных изображений. Количество исполнителей – 3, предметная область – компьютерная графика и обработка изображений, есть реализации, наличие открытых исходных текстов – есть, повторная тема.

– Конструирование логических схем (редактор схем). Количество исполнителей – 4, предметная область – цифровая схемотехника, есть неплохие реализации, наличие открытых исходных текстов – неизвестно, повторная тема.

– Анализ сетевого графика выполнения проекта (нахождение графического пути). Количество исполнителей – 3, предметная область – программирование и дискретная математика, есть реализации, наличие открытых исходных текстов – неизвестны, повторная тема.

– Графический эмулятор pdp11, основанный на симуляторе Супника. Количество исполнителей – 3, предметная область – конструирование графических интерфейсов, есть неплохие реализации, наличие открытых исходных текстов – есть, повторная тема.

– Генерация синтаксических диаграмм по грамматическим правилам. Количество исполнителей – 4, предметная область – конструирование компиляторов и компьютерная графика, реализации неизвестны, наличие открытых исходных текстов – неизвестно, повторная тема.

– Задача Дидоны с произвольной береговой линией. Количество исполнителей – 3, предметная область – теория оптимизации и вариационное исчисление, есть неплохие реализации, наличие открытых исходных текстов – есть, повторная тема.

– Многоколоночный фильтр для текстов с пропорциональным шрифтом и известной метрикой. Количество исполнителей – 3, предметная область – программирование, реализации неизвестны, наличие открытых исходных текстов – нет, повторная тема.

– Редактор графов. Количество исполнителей – 3, предметная область – дискретная математика и компьютерная графика, есть неплохие реализации, наличие открытых исходных текстов – есть, повторная тема.

– Анализ графов (основные алгоритмы работы с графами). Количество исполнителей – 4, предметная область – дискретная математика, есть неплохие реализации, наличие открытых исходных текстов – есть, повторная тема.

– Реализация фотоальбома: масштабирование, повороты и переносы растровых изображений. Количество исполнителей – 3, предметная область – компьютерная графика, есть неплохие реализации, наличие открытых исходных текстов – неизвестно, повторная тема.

– Редактор химических формул (2D изображений структурных формул). Количество исполнителей – 4, предметная область – химия и компьютерная графика, есть неплохие реализации, наличие открытых исходных текстов – есть, первичная тема.

– Констуктор молекул с 3D-интерфейсом на OpenGL с использованием фрагментов-заготовок. Количество исполнителей – 4, предметная область – компьютерная графика и органическая химия, есть неплохие реализации, наличие открытых исходных текстов – есть, повторная тема.

– Модель потока в трубе переменного сечения. Количество исполнителей – 3, предметная область – моделирование физики сплошных сред, реализации нет, наличие открытых исходных текстов – нет, повторная тема.

– Метод оптимизации многомерной функции многоточечным гольф-методом (методом «роя»). Количество исполнителей – 3, предметная область – численные методы анализа и теория оптимизации, есть реализации, наличие открытых исходных текстов – есть, повторная тема.

– Пользовательский интерфейс и симулятор процессора «VAX-11/780». Количество исполнителей – 4, предметная область – эмуляторы, реализации неизвестны, наличие открытых исходных текстов – неизвестны, первичная тема.

ЗАЩИТА РАБОТ

Выполненная курсовая работа сдается студентом руководителю в установленный срок (до наступления зачетной недели). Научный руководитель (заказчик) дает отзыв с указанием сильных и слабых сторон выполненной курсовой работы и ставит предварительную оценку. Работа, не соответствующая предъявляемым требованиям, возвращается студенту на доработку.

Курсовые работы, получившие положительный отзыв, допускаются к открытой публичной защите. Во время защиты студенту предоставляется возможность продемонстрировать работу своей программы, обосновывать выбранные средства реализации и отстаивать свою точку зрения.

Процедура защиты работы

Порядок обсуждения курсовой работы предусматривает: ответы студентов на вопросы преподавателей кафедры и других лиц, присутствующих на защите, выступление научного руководителя; право выступать с замечаниями и пожеланиями имеют все присутствующие.

На защите должны присутствовать все студенты, принимавшие участие в разработке, все они участвуют в демонстрации своего продукта, отвечают на вопросы, объясняют логику работы программы и получают одинаковые оценки по результатам защиты, за исключением ситуаций, рассматриваемых в п.10.2.

Оценка курсовой работы

Решение об оценке курсовой работы принимается преподавателями кафедры по результатам анализа представленной курсовой работы, представления разработанной программы студентами и их ответов на вопросы. Оценка по итогам защиты курсовой работы проставляется в ведомость (в т.ч. электронный вариант ведомости) и зачетные книжки студентов научным руководителем.

При оценке учитываются качества разработанного **программного продукта,** в том числе такие, как:

- Функциональность;
- Удобство;
- Эффективность;
- Дизайн интерфейса.

Учитывается качества **процесса разработки**, в том числе:

- Реализация алгоритмизация задачи;
- Слаженность работы команды;

– Реализация управлением качеством;
– Интеграция.;

Качество разработанной **документации**:

– Понятность специалисту в предметной области;
– Простота изложения;
– Полнота описания;
– Наличие иллюстраций и примеров использования.

Качество и полнота **пояснительной записки**, ее состав и соответствие выполненной работе.

Оценка вклада отдельных участников проекта требуется, если налицо неравномерное распределение работ, очевидны слабые стороны проекта или исследования ил пояснительной записки, за которые отвечал только один из исполнителей.

Выставляемая в ведомость и матрикул оценка в этом случае может быть неодинаковой для членов бригады исполнителей.

СПИСОК РЕКОМНДУЕМОЙ ЛИТЕРАТУРЫ И ЭЛЕКТРОННЫЕ РЕСУРСА

1. ftp://bkv.vvsu.ru/BOOKS – книги по программированию, INTRANET-ресурс.
2. ftp://bkv.vvsu.ru/pub/TP – материалы к занятиям по дисциплине «Технология программирования», компиляторы, примеры выполненных курсовых работ, постоянно меняющееся содержание в соответствии с текущими заданиями.
3. http://www.proklondike.ru – книги по программированию.
4. Адаменко А.Н., Кучуков А.М. Логическое программирование и VisualProlog. – СПб.: БХВ-Петербург, 2003. – 992 с.
5. Ахо, Д. Хопкрофт, Дж. Ульман, Структуры данных и алгоритмы. – М.; СПб.: Вильямс, 2001.
6. Буч Гради. Объектно-ориентированный анализ и проектирование с примерами приложений. Rational, Санта-Клара, Калифорния, пер.с англ. ред. И. Романовский, Ф. Андреев. -электронный ресурс, ftp://bkv.vvsu.ru/pub/TP/BOOKS/GradyBooch.chm
7. Васильев Б.К. Использование технологий готовых решений в программировании: лабораторный практикум. – Владивосток, ВГУЭС, 2004. – 43 с.
8. Гайдышев И. Анализ и обработка данных: специальный справочник. – СПб.: Питер, 2001. – 752 с.
9. Голуб А.Ф., Правила программирования С и С++. М., Бином, 1996.
10. Кнут Д., Искусство программирования для ЭВМ, Т. 1-3. – М.: Мир, 1978. (переизданы в 2001 г.)
11. Костельцов А.В. Построение интерпретаторов и компиляторов. – СПб.: Наука и Техника, 2001. – 224 с.
12. Орлов С. Технологии разработки программного обеспечения. – СПб.: Питер, 2002. – 464 с.
13. Рамбо Дж., Якобсон А., Буч Г. UML: специальный справочник. – СПб.: Питер, 2002. – 656 с.
14. Спинеллис, Диомидис. Анализ программного кода на примере проектов Open Source. – М.: Вильямс, 2004 – 528 с.
15. Стандарты Владивостокского государственного университета экономики и сервиса. СТО 1.005-2004. – Владивосток, Изд-во ВГУЭС, 2007.
16. Шафер Д.Ф., Фатрепп Р.Т., Шафер Л.И. Управление программными проектами: достижение оптимального качества при минимуме затрат. – М., Вильямс,2003. – 1136 с.

ПРИЛОЖЕНИЯ

Приложение 1

Образец титульного листа

ВЛАДИВОСТОКСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ЭКОНОМИКИ И СЕРВИСА
ИНСТИТУТ ИИБС
КАФЕДРА ИСКТ

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К курсовой работе по дисциплине
«Технология программирования»

Тема: Фрактальный ландшафт

Выполнили студенты группы ВМ XX – YY:
Иванов И.И.
Петров П.П.
Сергеев С.С.

Руководитель:
Васильев Борис Константинович

Владивосток
200 г.

Приложение 2

Образец технического задания

Техническое задание на курсовую работу по дисциплине
«Технология программирования»

Тема: фрактальный ландшафт

Предметная область: компьютерная графика
Заказчик: Васильев Б.К.
Руководитель: он же ☺.
Количество исполнителей: 3
Исполнители:
1. _Иванов И.И._ группа ВМ-XX-YY
_____
2.__Петров П.П._ группа ВМ-XX-YY
_____
3.__Сергеев С.С. группа ВМ-XX-YY
4._____(_____)
причины изменения состава

Исходные данные: Алгоритмы построения фрактальных ландшафтов, степень детализации.

Выходные данные: графическое трехмерное представление ландшафта в виде поверхности.

Описание ПО: программа для расчета и отображения цифровых ландшафтов, генерируемых фрактальными методами (горы, острова, реки), использующая подключаемые модули расчета и отображения. Основное назначение – проверка и тестирование модулей генерации и отображения, предполагаемое использование – в компьютерных играх. Необходимо реализовать несколько расчетных модулей для разных типов ландшафта.

Язык реализации: C или C++.

Поддерживаемые платформы: Linux, Windows32 (XP/2000/Vista, видеоадаптеры с аппаратной поддержкой OpenGL, – Nvidia, ATI/AMD).

Используемые программные средства: на выбор исполнителей.

Дополнительные требования к используемым средствам: вывод средствами OpenGL.

Дополнительные требования к процессу разработки: ежемесячные отчеты должны предоставляться заказчику.

Дополнительные требования к тестированию ПО: диагностические средства должны быть встроены в проект средствами условной компиляции.

Дополнительные требования к эксплуатации: вычисление фракталов должно производиться на процессоре с тактовой частотой 2 гГц при хорошей детализации (7-8 уровней для массива точек 200x200) не более чем за 2 сек.

Дополнительные требования к сопровождению и документации: представить описания параметров вызова всех функций, входящих в программу. Обязательно наличие комментариев в модулях отображения.

Источники:

Никулин Е.А. Компьютерная геометрия и алгоритмы машинной графики. СПб.: БХВ-Петербург, 2003, -560 С.

Шикин Е.В., Боресков А.В. Компьютерная графика. Динамика, реалистические изображения. – М.: ДИАЛОГ-МИФИ, 1995, – 288 С.

http://www.algolist.ru – русскоязычный сайт с описанием алгоритмов.

http://www.freshmeat.net – англоязычный сайт с описанием свободно распространяемого ПО и поисковой машиной.

График выполнения работ: (ленточный~~, сетевой~~): разработать и пред-

Ненужное зачеркнуть

ставить к утверждению в течение первой недели.

Исполнители:

_____ (_____)
_____ (_____)
_____ (_____)
_____ (_____)

Руководитель:

_____ (_Васильев Б.К._)

Ленточный график работ согласован (прикладывается к ТЗ):

Исполнители:

_____ (_____)
_____ (_____)
_____ (_____)
_____ (_____)

Руководитель:

_____ (_Васильев Б.К._)

Приложение 3

Пример неправильного оформления текста программы.

```
 01 // polygonView.cpp: implementation of the
 02 // CPolygonView class
 03 //
 04 #include "stdafx.h"
 05 #include "polygon.h"
 06 #include "polygonDoc.h"
 07 #include "polygonView.h"
 08 #ifdef _DEBUG
 09 #define new DEBUG_NEW
010 #undef THIS_FILE
011 static char THIS_FILE[] = __FILE__;
012 #endif
013 #define drand48()(((float) rand())/((float) \
014  RAND_MAX))
015 #define srand48(x) (srand((x)))
016 #define MAXLEVEL 8
017 #define ulong unsigned long
018 int DrawAxes = 0;
019 ////////////////////////////////////////////
020 // CPolygonView
021 IMPLEMENT_DYNCREATE(CPolygonView, CView)
022 BEGIN_MESSAGE_MAP(CPolygonView, CView)
023 //{{AFX_MSG_MAP(CPolygonView)
024 ON_WM_CREATE()
025 ON_WM_DESTROY()
026 ON_WM_SIZE()
027 ON_WM_MOUSEWHEEL()
028 ON_WM_LBUTTONDOWN()
029 ON_WM_LBUTTONUP()
030 ON_WM_MOUSEMOVE()
031 //}}AFX_MSG_MAP
032 // Standard printing commands
033 ON_COMMAND(ID_FILE_PRINT, CView::OnFilePrint)
034 ON_COMMAND(ID_FILE_PRINT_DIRECT,
035 CView::OnFilePrint)
036 ON_COMMAND(ID_FILE_PRINT_PREVIEW,
037 CView::OnFilePrintPreview)
038 END_MESSAGE_MAP()
039 void CPolygonView::CamChange()
040 {
```

```
041 glView-
port(0,0,m_oldRect.right,m_oldRect.bottom);
042 glMatrixMode(GL_PROJECTION);
043 glLoadIdentity();
044 gluPerspec-
tive(45.0f,(GLdouble)m_oldRect.right/m_oldRect.bott
om, 1.0f, 40.0f);
045 gluLookAt(0,CamDolly,CamDolly, 0,0,0, 0,0,1);
046 glMatrixMode(GL_MODELVIEW);
047
048 }
049 void CPolygonView::FractalTree(int level)
050 {
051 long savedseed; /* need to save seeds while
052 building tree too */
053 if (level == Level) {
054 glPushMatrix();
055 glRotatef(drand48()*180, 0, 1, 0);
056 glCallList(STEMANDLEAVES);
057 glPopMatrix();
058 } else {
059 glCallList(STEM);
060 glPushMatrix();
061 glRotatef(drand48()*180, 0, 1, 0);
062 glTranslatef(0, 1, 0);
063 glScalef(0.7, 0.7, 0.7);
064 savedseed = (long)((ulong)drand48()*ULONG_MAX);
065 glPushMatrix();
066 glRotatef(110 + drand48()*40, 0, 1, 0);
067 glRotatef(30 + drand48()*20, 0, 0, 1);
068 FractalTree(level + 1);
069 glPopMatrix();
070 srand48(savedseed);
071 savedseed = (long)((ulong)drand48()*ULONG_MAX);
072 glPushMatrix();
073 glRotatef(-130 + drand48()*40, 0, 1, 0);
074 glRotatef(30 + drand48()*20, 0, 0, 1);
075 FractalTree(level + 1);
076 glPopMatrix();
077 srand48(savedseed);
078 glPushMatrix();
079 glRotatef(-20 + drand48()*40, 0, 1, 0).
```

Приложение 4

Пример корректно выполненных комментариев в тексте

```
 01  #define SHADX 5 //размер тени от кнопки по x
 02  #define SHADY 5 //размер тени от кнопки по y
 03  #define SHADC 3 //цвет тени
 04  #define MASHT 3 //масштаб кнопки
 05  #define LEFTBM 1 //идентификатор левой кн. мыши
 06  // Далее идет массив, в котором содержатся
 07  // названия кнопок интерфейса текст. процессора:
 08  static
 09  char*Names[]={"Load","Save","Close","Exit",
010  "Menu","Help","Print","Scan","Read","Stop",
011  "Run","Macr"};
012  // Возможно добавление новых кнопок,
013  // их редактирование и удаление старых.
014  struct xyTag {
015  int x;
016  int y;
017  }; //структура, описывающая положение кнопки.
018  struct buttonTag {
019  struct buttonTag *prev; // предыдущая кнопка
020  struct buttonTag *next; // следующая кнопка
021  struct xyTag (*verts)[];
022  // структура с указателем на кол-во вершин
023  // кнопки
024  int numverts; // количество вершин
025  char *text; // текст кнопки
026  int colrb; // цвет окаймления
027  int colrp; // цвет внутренней области
028  enum onoff pressed; // состояние кнопки
029  };
030  /* структура, описывающая саму кнопку (кнопки
031  представлены в виде линейного списка) */
032  void getminmax(struct buttonTag *button,
033  int *minx, int *miny, int *maxx, int *maxy)
034  {
035  int c;
036  *minx = (*button->verts)[0].x;
037  *miny = (*button->verts)[0].y;
038  *maxx = (*button->verts)[0].x;
039  *maxy = (*button->verts)[0].y;
040  for (c = 1; c < button->numverts; c++)
```

```
041 {
042 if ((*button->verts)[c].x < *minx)
043 *minx = (*button->verts)[c].x;
044 if ((*button->verts)[c].y < *miny)
045 *miny = (*button->verts)[c].y;
046 if ((*button->verts)[c].x > *maxx)
047 *maxx = (*button->verts)[c].x;
048 if ((*button->verts)[c].y > *maxy)
049 *maxy = (*button->verts)[c].y;
050 }
051 } /*Эта функция предназначена для нахождения
052 минимальной и максимальной координат (по x и
053 по y) кнопки, т.е. координаты ее верхнего ле
054 вого и нижнего правого угла. */
055 int addbutton(int numverts, int colrp,
056 int colrb, char *text, int masht)
057 {
058 int c;
059 double alfa;
060 struct buttonTag *button;
061 int minx, miny, maxx, maxy;
062 if ((button = (struct buttonTag *)
063 malloc(sizeof(struct buttonTag))) == NULL)
064 {
065 printf("Not enough memory to allocate \
066 buffer\n");
067 return 1;
068 }
069 button->numverts = numverts;
070 if (last) last->next = button;
071 button->prev = last;
072 button->next = NULL;
073 button->colrp = colrp;
074 button->colrb = colrb;
075 button->pressed = OFF;
076 strcpy(button->text, text);
077 alfa = 2 * 3.14159265 / numverts
```

Приложение 5

Пример корректного описания функций

Описание функций:
В функции **main** решается прямая задача (находится матрица *U*) и осуществляется графический вывод;
double fx(float x, float a) – аппаратная функция,
a – параметр аппаратной функции (полуширина).
double zx(float x) – позволяет выбрать искомую функцию
из следующих:
double OnePik(double r) – треугольная функция.
double step(float x) – прямоугольная функция.

double MatrK (void) Решает интеграл $\int_c^d K(x,t)K(x,s)dt$

в результате получается квадратная матрица **K[100]*[100].**

void VectorB(void) – решает интеграл $\int_c^d K(s,x)U(s,x)dx$

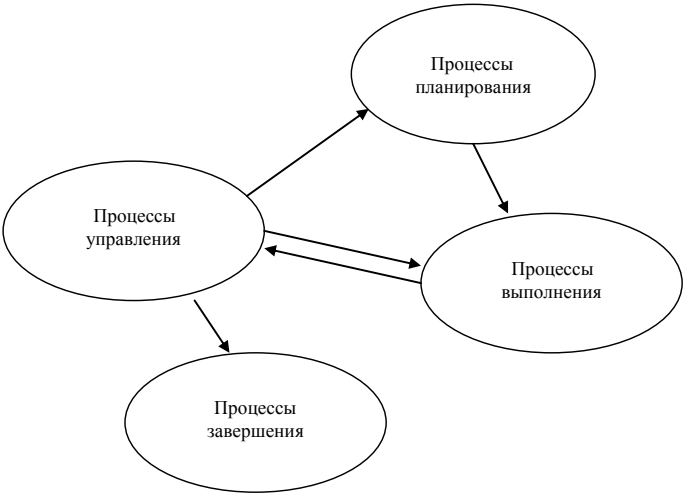
в результате получается матрица **b[100].**
void reg(float a) – регуляризирует с коэффициентом ***a***
матрицу ***K***;
void eqsys(void) – решает систему уравнений *K**z=*B*,
где *z*- искомая функция;
double Norm(double mtrx[],n) – Возвращает максимальное
значение матрицы **mtrx[]** размером **n**;
Описание программы:
Вначале решается прямая задача методом свертки:

$$\int_0^{100} f(x-x_0)z(x)dx = U$$

Далее находится вектор *B* и матрица *K*;
Далее матрица *K* регуляризируется; и решается система уравнений в результате чего находится искомая функция *f*;
В конце результат решения выводится на экран.

Приложение 6

Модель процессов на всех этапах выполнения проекта



СОДЕРЖАНИЕ

ЗАДАЧИ КУРСОВОЙ РАБОТЫ 1
ТРЕБОВАНИЯ, ПРЕДЪЯВЛЯЕМЫЕ К РАЗРАБАТЫВАЕМОМУ ПО 5
Целостность 5
Мобильность 5
Наличие графического интерфейса 6
Документация 6
Коллективная работа над проектом 7
Временные рамки и графики работ 7
Пояснительная записка 8
Состав пояснительной записки 8
Оформление пояснительной записки 10
Оформление приложений 10
ПРОГРАММНЫЙ ПРОЕКТ 12
ТЕХНИЧЕСКОЕ ЗАДАНИЕ 13
МОДЕЛЬ ПРОЦЕССА РАЗРАБОТКИ 14
Моделирование 14
Модель жизненного цикла разработки 14
ВЫБОР СРЕДСТВ РЕАЛИЗАЦИИ 17
Примеры реализации 17
ПРЕДМЕТНАЯ ОБЛАСТЬ 18
ТЕМЫ 19
ЗАЩИТА РАБОТ 23
Процедура защиты работы 23
Оценка курсовой работы 23
СПИСОК РЕКОМНДУЕМОЙ ЛИТЕРАТУРЫ
И ЭЛЕКТРОННЫЕ РЕСУРСА 25
ПРИЛОЖЕНИЯ 26

Учебно-методическое издание

Составитель
Борис Константинович Васильев

ТЕХНОЛОГИЯ ПРОГРАММИРОВАНИЯ

Руководство к выполнению курсовой работы
по специальностям
23010165 «Вычислительные машины, комплексы и сети»
23020165 «Информационные системы и технологии»

В авторской редакции
Компьютерная верстка Н.А. Игнатьевой

Лицензия на издательскую деятельность ИД № 03816 от 22.01.2001

Подписано в печать 27.03.2009. Формат 60×84/16.
Бумага писчая. Печать офсетная. Усл. печ. л. 2,2.
Уч.-изд. л. 2,1. Тираж 50 экз. Заказ

Издательство Владивостокский государственный университет
экономики и сервиса
690600, Владивосток, ул. Гоголя, 41
Отпечатано в типографии ВГУЭС
690600, Владивосток, ул. Державина, 57
36