Министерство образования Российской Федерации

Сибирский государственный аэрокосмический университет

Факультет информатики и систем управления

Кафедра информатики и вычислительной техники

Методические указания по выполнению курсовой работы по дисциплине «Технология программирования»

Курс – 2
Специальности: 552800 и 220200

Автор: Моргунов Евгений Павлович

Красноярск 2003

Содержание

Введение

Главное, ребята, сердцем не стареть,
Песню, что придумали, до конца допеть

Из песни

Уважаемые коллеги-программисты, перед вами первый вариант «Методических указаний». Цель этого документа – помочь вам выполнить курсовую работу на хорошем уровне. Для достижения поставленной цели будут использованы все доступные автору этих строк средства: убеждение, ссылки на личный опыт, дружеские советы и т.д. Возможно, некоторые части текста имеют несколько назидательный характер, но не судите строго. Автор надеется, что он имеет на это некоторое право, во-первых, по долгу службы, как старший товарищ, желающий младшим только добра, а во-вторых, как программист, имеющий определенный опыт профессиональной деятельности.

Эти «Методические указания» не претендуют на то, чтобы дать детальнейшие указания на все случаи жизни. Вряд ли это возможно, да и нужно ли? Автор считал бы свою задачу выполненной, если бы вы после прочтения сего документа активизировали свою деятельность, стали более творчески подходить к выполнению предстоящей курсовой работы, стали задавать больше сложных вопросов, начали самостоятельно читать книги, которые перечислены в списке литературы, приведенном в конце этого текста.

Естественно, первая, выражаясь языком программистов, версия «Методических указаний» не может быть идеальной. Автор с благодарностью примет все конструктивные предложения и пожелания по улучшению данной работы.

Цели и задачи курсового проектирования

Курсовой проект – это большая работа, которая в принципе отличается от тех небольших заданий, которые вы выполняли до сих пор. Но поскольку в вашей будущей профессиональной жизни вам придется решать большие задачи, а не только выполнять упражнения, то данный проект и должен стать вашей первой большой задачей.

Его влияние на становление специалиста огромно. Курсовой проект призван способствовать выработке у вас умения увидеть проблему, которая может быть решена средствами информационных технологий. Затем вы должны четко сформулировать ваши цели и задачи, чтобы в итоге выполнения работы вы не оказались в ситуации, которую можно описать словами: **что** получили, **то** и хотели. Затем следует приступать к проектированию программы и лишь после этого можно садиться за компьютер и непосредственно сочинять исходный текст. Кстати, есть хорошая книжка об умении решать проблемы: Р. Акофф. «Искусство решения проблем». Она не о программировании, а о решении проблем вообще в различных областях. В ней масса интересных примеров.

Важным качеством настоящего современного программиста является **умение работать в коллективе**. Государственный образовательный стандарт требует, чтобы выпускник нашего факультета имел навыки коллективной работы. Но даже если бы в стандарте этого требования не было, все равно такое требование продиктовано самой жизнью. Ни для кого не секрет, что разработать операционную систему, например, Windows, исходный код которой состоит из нескольких *миллионов* строк, одному человеку не под силу. Таким образом, программисты просто обречены работать коллективно. Вы также должны будете пройти через это. Конечно, на пути коллективной работы есть немало подводных камней. Автором этих строк практика коллективной работы над курсовым проектом используется очень давно, и ему доводилось неоднократно видеть студенческие горе-коллективы, в которых один пишет, а остальные, образно говоря, только «сочувствуют» ему, ожидая, когда он, наконец, завершит работу. Но всегда, каждый год, бывали замечательные коллективы, в которых каждый участник работал с полной отдачей, не пытаясь обмануть ни товарищей по совместной работе, ни преподавателя. Естественно, преподаватель – это не сыщик, он не в состоянии проверить каждого студента, заглянуть ему в душу, так что возможность обмануть преподавателя есть, только пользоваться ею не следует. Тем же, кто считает иначе, можно напомнить поговорку: жизнь все поправит. Может быть, в один прекрасный день такой «специалист» очнется на бирже труда (или в службе по трудоустройству) и вспомнит эти предостережения. Но хочется надеяться, что среди вас все-таки больше тех, кто действительно хочет стать специалистом. Поэтому коллективная работа будет в целом полезной и продуктивной.

Как придумать свою тему или выбрать одну из предложенных тем

Выбор темы – очень важный этап в выполнении курсового проекта. От правильного выбора зависит вся ваша дальнейшая работа по реализации проекта. Если тема очень простая, то вам не удастся много «выжать» из нее. Программа неизбежно получится слишком маленькой для того, чтобы претендовать на признание ее в качестве проекта. Ни в коем случае не следует высасывать исходный код из пальца. Вы должны выбрать такую тему, чтобы она – при условии тщательной проработки – была очень продуктивной, чтобы не приходилось что-то искусственно придумывать ради увеличения объема программы. Одним из формальных требований к курсовой работе будет требование к количеству строк исходного текста.

Выбрать тему курсовой работы вы должны самостоятельно. Конечно, вы можете (и должны) посоветоваться с преподавателем. Однако вам следует понимать, что интеллектуальное иждивенчество – это плохое качество, и от него нужно избавляться всеми силами. В жизни никто и никогда не будет давать вам несколько вариантов задачи, оберегая вас тем самым от списывания и освобождая от необходимости работать своей головой. Все эти «варианты» способствуют выработке пассивности, приучают к ожиданию готовых, кем-то сформулированных, задач. Нужно учиться самостоятельно формулировать задачи. Никакой преподаватель не в состоянии предложить *каждому* из вас тему, которая была бы интересной для вас, новой, посильной, перспективной, не похожей на другие темы и т.д. и т.п. Поэтому ваше активное участие в выборе темы просто необходимо.

Что же может помочь вам в решении этой многотрудной задачи? Вот лишь некоторые идеи:

1. Где работают ваши родители? Подумайте, может быть, они вам предложат тему для разработки?

2. Что вас интересует кроме программирования? Может быть, вы что-то коллекционируете, занимаетесь спортом, имеете собаку и т.д. Все эти занятия могут дать богатую пищу для размышления о выборе интересной темы для написания прикладной программы.

3. Если же вас интересует только программирование (или вам так кажется), тогда вам нужно подумать над тем, что вам интереснее – системное или прикладное программирование, операционные системы или базы данных, Internet-технологии или компьютерная графика. Вы можете взять в качестве темы изучение какой-то технологии (например, разобраться, что же такое Remote Procedure Calls в операционной системе UNIX). Тогда итогом вашей работы может стать отчет или даже методическое руководство по изучению данного вопроса. Этим руководством впоследствии смогут воспользоваться ваши младшие товарищи, которые сейчас, может быть, только готовятся к поступлению в наш университет. Конечно, такие нестандартные темы требуется более детально обсудить с преподавателем.

4. Вы можете посмотреть на FTP-сервере факультета курсовые работы прошлых лет. Просматривая работы ваших предшественников, вы можете что-то позаимствовать для себя, даже случайно наткнуться на интересную идею. При этом выполнять работу нужно все равно самостоятельно, не следует слепо копировать чужие стили, подходы, приемы. Нужно изучать эти работы критически. Конечно, может возникнуть соблазн позаимствовать одну из таких работ

целиком и таким образом облегчить себе жизнь. Но согласитесь, что это просто некрасиво.

5. А может быть, вы уже что-то программируете для себя? Тогда можно обсудить вашу программу с преподавателем и попытаться совместить полезное с приятным: вы сделали бы полезную программу для собственных нужд и отчитались перед преподавателем по курсовой работе.

6. Некоторые из вас, возможно, планируют поступать в магистратуру (и далее – в аспирантуру). Это очень здорово. Но в таком случае имеет смысл подумать над тем, чтобы выбрать такую тему, которая будет вам интересна все годы обучения на нашем факультете, которая может впоследствии стать темой вашей магистерской диссертации или дипломного проекта. Если перевести эти слова на русский язык, то можно коротко сказать так: выбор темы, связанной с наукой – это тоже очень хороший выбор. Ну, например, вас могут интересовать проблемы машинного перевода с иностранных языков или математическое моделирование в экономике, или другие интересные вещи. Уместно напомнить вам, что наш замечательный физик, лауреат Нобелевской премии Жорес Алферов начал заниматься той темой, за которую ему и была присуждена премия, еще учась на третьем курсе института. Этот пример говорит о пользе более ранней специализации. Так что выбирайте себе тему с учетом дальнейших перспектив, а не на один семестр.

7. Конечно, автор этих строк отдает себе отчет в том, что некоторые из вас ставят перед собой скромные цели – выжить, а в качестве курсовой работы сделать хоть что-нибудь. В этом случае можно посоветовать следующее: попробуйте сделать работу с использованием Internet-технологий. Вспомните те лабораторные работы, которые вы выполняли в первом семестре. Эти технологии сейчас очень актуальны, и такие умения еще могут вам когда-нибудь пригодиться.

8. Когда вам совсем плохо: идей нет и нет идей насчет того, где найти идею, тогда вы можете обратиться к преподавателю. Но это последнее средство. Не прибегайте к нему, пока не испробовали все вышеприведенные способы.

Ну и еще несколько замечаний. Как вы помните, выше шла речь о коллективной работе над курсовым проектом (кстати, состав такого коллектива 2–3 человека). При выборе коллег рекомендуется учитывать цели каждого участника будущей творческой группы. Если одного устраивает оценка «удовлетворительно», а другой меньше чем на «отлично» не согласен, то не стоит работать вместе. Если один обожает изучать, к примеру, тонкости работы операционных систем, а другому они противны, лучше поискать себе других партнеров. Помните: вы будете делать общее дело, и поэтому будет лучше, когда это дело интересно всем участникам творческой группы.

И в завершение этого раздела коснемся тезиса «Программа должна быть как можно короче». Его часто используют студенты в качестве возражения в ответ на требование (его вы увидите далее по тексту) создать программу определенного объема, например, 2000 строк. В ответ можно сказать следующее:

1. Уважающий себя специалист должен быть **в состоянии написать программу большого размера**: ведь многие даже довольно простые прикладные программы состоят из двух-трех десятков тысяч строк текста (не говоря уже об объемах текста операционных систем или систем управления базами данных). Даже простая, но *большая* программа в чем-то гораздо сложнее изощренной, но *маленькой* по объему. Большое ведь еще нужно уметь охватить своим умом. Нужно уметь видеть все связи между модулями, все особенности

их взаимодействия. Такие умения не вырабатываются при написании небольших программ, пусть даже очень сложных в алгоритмическом отношении.

2. Тема должна быть настолько продуктивной и плодотворной, чтобы без искусственного раздувания она дала нужный объем программного кода, т.е. когда даже при максимальном сокращении программного кода (в частности, за счет использования функций вместо повторяющихся фрагментов текста) написать более короткую программу по данной теме **невозможно.**

3. Написание как можно более короткой программы имело бы смысл только в том случае, если бы все выполняли работу по одной и той же теме.

Итак, повторим все еще раз коротко. Цель – научиться писать большие программы. Способ достижения цели – выбор плодотворной темы, избавляющей вас от необходимости высасывать программный код из пальца ради того, чтобы получить заветные две или три тысячи строк текста.

Формальные требования к курсовой работе

Такие требования необходимы по двум причинам: во-первых, это учебная работа, преследующая учебные цели, и она должна быть регламентирована по объему, оформлению, применяемым средствам и т.д.; во-вторых, вы должны знать минимальные требования, несоблюдение которых гарантированно не позволяет получить желаемую оценку. Если провести аналогию с математикой, то выполнение таких требований является *необходимым*, но не *достаточным* условием. Да и вообще курсовой проект не должен выполняться за один вечер, иначе это не проект, а упражнение.

Требования

1. Курсовой проект выполняется коллективно. Состав творческой группы: 2 или 3 человека. Допускается сотрудничество студентов из разных групп одного потока.

2. Распределение обязанностей в творческой группе – на усмотрение ее участников.

3. Темы – практически любые. Можно взять за основу те темы, которые предложены в настоящем руководстве. Не забывайте о том, что тема должна быть плодотворной, иначе вам не удастся написать большую программу, которая вправе претендовать на статус проекта.

4. Операционная система, в которой должна выполняться разработка – только UNIX (FreeBSD, Linux, SCO, Sun и т.д.).

5. Язык программирования – C/C++, Perl, Shell, Java, Python, Tcl/Tk и т.д. Не допускается использование сред визуального программирования, т.к. с такими средами вы познакомитесь при изучении других учебных дисциплин. Программист не должен бояться черного экрана и командной строки. Сказанное ни в коей мере не умаляет удобства, полезности, быстроты разработки, присущих средам RAD (Rapid Application Development), но, тем не менее, не только такие среды используются в настоящее время при разработке программ, и вы должны не только уметь двигать мышью, но и писать исходный текст в обычном текстовом редакторе.

6. Объем работы (все требования даны в расчете на одного участника творческой группы):

– объем исходного текста программы с комментариями не менее:
- 1000 строк – на оценку «удовлетворительно»;
- 2000 строк – на оценку «хорошо»;
- 5000 строк – на оценку «отлично»;

Примечание. Количество комментариев – 20–25 процентов от объема исходного текста, т.е. в среднем одна строка на 3-4 строки исходного текста. Рекомендации по написанию комментариев смотрите в конспекте лекций.

– объем проектной документации:
- 6 страниц – на оценку «удовлетворительно»;
- 8 страниц – на оценку «хорошо»;
- 10 страниц – на оценку «отлично».

7. Требования к тестированию программы:

– должно быть использовано не менее двух методов черного и двух методов белого ящика;

– должно быть протестировано (в расчете на одного участника творческой группы) не менее:

– 1 функции (модуля) – на оценку «удовлетворительно»;

– 2 функций (модулей) – на оценку «хорошо»;

– 3 функций (модулей) – на оценку «отлично»;

– должна быть протестирована программа как единое целое (используя методы черного ящика);

– все тесты и их результаты должны быть описаны в документации. Тесты должны быть такими, чтобы их можно было воспроизвести при защите курсовой работы. Обязательно должны быть написаны специальные программы для тестирования. Их объем включается в общий объем ваших исходных текстов.

8. Требования к проектной документации. Состав документации:

– титульный лист;

– содержание;

– список участников творческой группы с распределением обязанностей (архитектор, администратор, программист, ответственный за тестирование, ответственный за документацию);

– планы работы (как первоначальный, так и поэтапный). Не забывайте, что в плане должны быть указаны виды работ, сроки их выполнения и ответственные за их выполнение;

– план по качеству (напишите его на основе соответствующего раздела из электронного конспекта лекций). Не пишите слишком много требований в этот план. Помните, что все такие требования должны быть реализуемыми и проверяемыми. Если что-то невозможно проверить или обеспечить, то такое требование нецелесообразно включать в план. Не забывайте также о различиях между планом разработки и планом по качеству;

– техническое задание;

– технический проект;

– описание методики тестирования и тестовые наборы данных;

– руководство программиста;

– руководство пользователя.

9. Дополнительные требования к документации:

– обязательно использовать любые графические схемы в количестве не менее (в расчете на одного участника творческой группы):

– 1 схемы – на оценку «удовлетворительно»;

– 2 схем – на оценку «хорошо»;

– 3 схем – на оценку «отлично»;

– не допускать обилия орфографических ошибок. Если вы сами не в состоянии проверить свой текст, поручите это редактору Microsoft Word;

– не допускать использования разговорного жаргона (например, таких слов, как «мамка», «генерить», «компилить» и т.п.). Нужно выражать свои мысли только литературным техническим языком.

10. Требования к оформлению исходных текстов:

– в каждом модуле должен быть заголовок, включающий наименование программы, наименование и назначение модуля, автора, дату и номер версии;

– каждая функция должна быть оформлена надлежащим образом, а именно: должны быть описаны все формальные параметры и указано ее назначение;

– ширина текста не должна превышать ширину экрана, т.е. 80 символов;

– структурные отступы, с помощью которых выделяются вложенные управляющие конструкции while, if, for, должны аккуратно соблюдаться в едином стиле на протяжении всей программы (если вы выбрали величину такого отступа равной, например, двум символам, то следуйте этой схеме постоянно).

11. После завершения работы нужно предоставить преподавателю не только проектную документацию, но также и исходные тексты для размещения их на FTP-сервере факультета.

12. При выполнении курсовой работы следует пользоваться рекомендациями, приведенными в электронном конспекте лекций.

Пример

Для творческой группы, состоящей из трех человек, которые претендуют на оценку «хорошо», должны быть выполнены такие требования:

– объем программы – 6000 строк с комментариями (включая и тестовые программы):

– объем проектной документации – 24 страницы:

– должно быть протестировано не менее 6 функций (модулей) программы:

– должно быть включено в документацию не менее 6 графических схем.

Следует помнить, что выполнение количественных требований не влечет за собой автоматического получения соответствующей оценки, т.к. она зависит не только от объема, но также и от качества работы. Поэтому не забывайте о качестве работы, об интересных идеях.

Предлагаемые темы и направления работы

В этом разделе приводятся краткие описания нескольких возможных тем и направлений работы. Они могут послужить отправной точкой для ваших собственных размышлений. Конечно, данный перечень очень краток.

Между темой и направлением есть некоторое различие. Оно заключается в степени обобщения: тема – это нечто более конкретное, а направление – это более широкое понятие, в рамках которого можно выбрать более узкую тему. Например, в рамках направления «Операционные системы» можно выбрать такую тему, как «Начальная загрузка операционной системы», а в рамках направления «Сетевое администрирование» можно предложить тему «Программа для мониторинга работы компьютеров в учебной аудитории».

Направление «Базы данных»

В качестве системы управления базами данных (СУБД) можно использовать PostgreSQL или MySQL. В качестве средств создания интерфейса пользователя можно использовать Web-интерфейс, т.е. браузер клиента. Программы можно написать на языках C/C++ или Perl. Познакомиться с примерами технологии такого типа можно по программам автора настоящих «Методических указаний». Владение Web-технологиями очень актуально в настоящее время, поэтому внимательно прислушайтесь к этим словам.

Одной из целей данного направления является освоение языка SQL, который является стандартом для «больших» (СУБД). Нужно написать программы (скрипты) для генерации таблиц БД и распределения привилегий доступа к таблицам.

Учет и анализ шахматных партий

Основная задача или идея: это не программа для игры в шахматы, а программа для анализа сыгранных партий. Она должна позволять организовать хранение партий в базе данных (на уровне ходов), учитывать типы дебютов, вести учет шахматистов, чьи партии представлены в базе данных. Анализ может включать в себя получение ответов на такие вопросы:

- как обычно играет шахматист такой-то в такой-то ситуации;
- счет в партиях;
- число ходов, число партий, сыгранных за период времени такой-то и т.п..

Планирование тренировочного процесса

Если вы занимаетесь спортом (или даже физкультурой), то разумно направить свой талант программиста на создание программы, которая помогала бы вам в планировании тренировочного процесса. Если, к примеру, вы занимаетесь тяжелой атлетикой, то следует предусмотреть в программе такие вещи, как: виды упражнений, нагрузка, самочувствие, антропометрические показатели. Логично было бы проводить анализ достижений:

– графики нагрузки во времени;

– предполагаемые достижения;
– учет личных достижений в различных упражнениях.

Домашняя библиотека

Такая программа должна позволять вести учет книг, находящихся в личном пользовании, а также всех полезных книг и журнальных статей, попавших в поле зрения. Также можно организовать хранение и поиск выписок (цитат) из различных источников. Это может быть полезно при подготовке рефератов (конечно, если брать готовые рефераты в сети Internet, то тогда вообще ничего не нужно, но это совсем другой случай).

Примерная структура базы данных (она очень простая и приведена лишь в качестве стартовой информации для начала собственного проектирования):

Таблица «Книга»
– Наименование
– Код издательства
– Год издания
– Код ББК
– Код ISBN
– Ключевые слова

Таблица «Издательство»
– Код
– Наименование
– Код города

Таблица «Персоналии»
– Фамилия
– Имя
– Отчество
– Год рождения
– Страна проживания
– Уникальный внутренний код

Таблица «Авторы книг»
– ISBN или внутренний код
– Код автора
– Порядковый номер автора в описании книги

Идея – сделать **таблицу «Ключевые слова»** (тогда поля «Ключевые слова» в таблице «Книга» не будет):

– ISBN
– Ключевое слово

Направление «Операционные системы, сетевое и системное администрирование»

Данное направление нацелено на более глубокое изучение программирования на таких языках, как shell (это язык командного интерпретатора в опера-

ционной системе UNIX) и Perl. Без хорошего владения этими языками не может быть хорошего администратора, а плохой никому не нужен.

Мониторинг работы компьютеров в локальной сети учебной аудитории

Цель – сделать программу (или комплекс программ), позволяющий администратору учебной аудитории следить за работой всех подведомственных ему компьютеров со своего рабочего места, т.е. дистанционно: устанавливать необходимые программные продукты, контролировать заполнение жестких дисков, своевременно определять наличие проблем в локальной сети и т.д. Для удобства можно сделать Web-интерфейс к такой программе

Изучение принципов создания конфигурационных утилит, используемых при установке программных продуктов из исходных текстов

Если посмотреть любой программный продукт, доступный в исходных текстах в операционной системе UNIX, то можно увидеть в комплекте поставки программу под именем **configure**. Она предназначена для подготовки исходных текстов программного продукта непосредственно к установке на данном компьютере с данной версией операционной системы UNIX. Эта служебная программа формирует make-файлы, которые затем уже используются при компиляции исходных текстов.

Программа **configure** пишется на языке shell. Целью вашего курсового проекта должно стать доскональное изучение принципов работы такой программы. В качестве отчета нужно представить *детально прокомментированную* программу **configure** (например, из пакета Apache или deco) и текстовое описание принципов построения такой программы и приемов, применяемых в ней.

Другие темы

Изучение принципов работы текстовых редакторов

Суть работы – взять исходный текст тестового редактора, например, JOE, и изучить его. Написать отчет с изложением основных принципов построения редактора и описанием его функциональных блоков, основных алгоритмов и структур данных. Например, описать механизм реализации «откатов» (undo), т.е. отмены операций.

Список полезной литературы

1. Кантор М. Управление программными проектами. Практическое руководство по разработке успешного программного обеспечения: Пер. с англ. – М: Издательский дом «Вильямс», 2002. – 176 с.
2. Бек К. Экстремальное программирование. – СПб.: Питер, 2002. – 224 с.
3. Соммервилл И. Инженерия программного обеспечения, 6-е издание.: Пер. с англ. – М: Издательский дом «Вильямс», 2002. – 624 с.
4. Леффингуэлл Д, Уидриг Д. Принципы работы с требованиями к программному обеспечению. Унифицированный подход: Пер. с англ. – М: Издательский дом «Вильямс», 2002. – 448 с.
5. Гровф Дж., Вайнберг П. SQL: Полное руководство: Пер. с англ. – 2-е изд., перераб. и доп. – К.: Издательская группа BHV, 2001. – 816 с.
6. Коннолли Т., Бегг К., Страчан А. Базы данных: проектирование, реализация и сопровождение. Теория и практика, 2-е изд.: Пер. с англ.: Уч. пос. – М: Издательский дом «Вильямс», 2000. – 1120 с.
7. Эбен М., Таймэн Б. FreeBSD. Энциклопедия пользователя: Пер. с англ. / Майкл Эбен, Брайан Таймэн. – К.: ООО «ТИД «ДС», 2002. – 736 с.
8. Кристиансен Т., Торкингтон Н. Perl: библиотека программиста. – СПб.: Питер, 2001. – 736 с.
9. Браун М. Perl. Архив программ. – М: ЗАО «Издательство БИНОМ», 2001. – 720 с.
10. Тейнсли Д. Linux и UNIX:программирование в shell. Руководство разработчика: Пер. с англ. – К.: Издательская группа BHV, 2001. – 464 с.
11. Снейдер Й. Эффективное программирование TCP/IP. Библиотека программиста. – СПб.: Питер, 2001. – 320 с.
12. Митчелл М., Оулдем Дж., Самьюэл А. Программирование для Linux. Профессиональный подход: Пер. с англ. – М: Издательский дом «Вильямс», 2002. – 288 с.
13. Керниган Б., Пайк Р. Практика программирования / Пер. с англ. – СПб.: Невский диалект, 2001. – 381 с.
14. Канер С. и др. Тестирование программного обеспечения: Пер. с англ. / Сэм Канер, Джек Фолк, Енг Кек Нгуен. – К.: Издательство «ДиаСофт», 2000. – 544 с.