СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ
АЭРОКОСМИЧЕСКИЙ УНИВЕРСИТЕТ
имени академика М. Ф. Решетнева

Е. П. Моргунов
О. Н. Моргунова

Технология программирования

Курсовое проектирование

Красноярск – 2011

Министерство образования и науки Российской Федерации

Сибирский государственный аэрокосмический университет
имени академика М. Ф. Решетнева

Е. П. Моргунов
О. Н. Моргунова

Технология программирования

Курсовое проектирование

*Рекомендовано к изданию научно-методической комиссией
института информатики и телекоммуникаций*

Красноярск – 2011

УДК 004.4
ББК 32.973.26
М 79

Рецензент:
доктор технических наук, профессор А. Н. Антамошкин
(Сибирский государственный аэрокосмический университет)

Моргунов, Е. П.
М 79 Технология программирования. Курсовое проектирование : учеб.-метод. пособие / Е. П. Моргунов, О. Н. Моргунова ; Сиб. гос. аэрокосмич. ун-т. – Красноярск, 2011. – 87 с.

В учебно-методическом пособии содержатся указания по выполнению курсового проекта по дисциплине «Технология программирования». Даются конкретные рекомендации по выбору темы проекта и оформлению проектной документации. Приводятся примеры документов.

Пособие предназначено для студентов, обучающихся по направлениям 230100 – «Информатика и вычислительная техника» и 230200 – «Информационные системы». Оно может быть полезно широкому кругу студентов, знакомых с основами программирования, впервые приступающих к разработке программных продуктов.

УДК 004.4
ББК 32.973.26

Оглавление

1. Введение

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

Из песни

Уважаемые студенты-программисты, пособие, представленное вашему вниманию, призвано помочь вам выполнить курсовой проект на высоком уровне. Курсовой проект – это серьезная работа, она в принципе отличается от тех небольших заданий, которые вы выполняете в рамках практических занятий по информационно-компьютерным дисциплинам. Но поскольку в будущей профессиональной жизни вам придется решать масштабные задачи, а не только выполнять упражнения, то курсовое проектирование должно подготовить вас к решению подобных задач.

Выполнение курсового проекта способствует выработке умения увидеть проблему, которая может быть решена средствами информационных технологий. Осознав проблему, вы затем должны четко сформулировать ваши цели и задачи, в результате решения которых проблема была бы устранена. Затем следует приступать к проектированию программы и лишь после этого можно садиться за компьютер и непосредственно писать (конструировать) исходный текст. Кстати говоря, есть хорошая книга Р. Акоффа «Искусство решения проблем». Она не о программировании, а о методах решения проблем в различных областях. Программисты также могут найти в ней массу интересных и полезных идей.

Пособие не претендует на то, чтобы дать детальные указания на все случаи жизни. Вряд ли это возможно, да и нужно ли? Авторы считали бы свою задачу выполненной, если бы вы после прочтения этого документа активизировали свою деятельность, стали более творчески подходить к выполнению предстоящей курсовой работы, стали задавать больше сложных вопросов.

Поскольку предела совершенству нет, то авторы с благодарностью примут все конструктивные предложения и пожелания по дальнейшему улучшению данной работы.

Мы надеемся, что изучение материала, изложенного в учебном пособии, будет способствовать расширению вашего профессионального кругозора и повышению уровня квалификации.

2. Выбор темы курсового проекта

Выбор темы – это очень важный этап, поскольку именно он определяет направление и успех всей последующей работы. На принятие решения относительно темы влияют различные факторы, которые мы и рассмотрим в этой главе. Кроме того, важным вопросом является формулирование названия темы, поскольку оно должно удовлетворять целому ряду критериев. Выбрать тему непросто, поэтому мы предложим ряд возможных направлений курсового проектирования.

2.1. Факторы, влияющие на выбор темы

Что такое фактор? Это то, что является действующей (движущей) силой или существенным обстоятельством в какой-то ситуации, в каком-то процессе, например, в процессе принятия важного решения.

Надеемся, что решение о выборе темы курсового проекта является для вас важным. 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. 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 – о них речь пойдет дальше – 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. Курсовой проект – это не типовой расчет, в котором вам для решения предлагаются уже сформулированные задачи. Нужно учиться видеть проблемы в окружающей вас среде и самостоятельно формулировать задачи по решению этих проблем. 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.

А теперь рассмотрим факторы, которые имеют место в данной ситуации.

1. Ваши перспективные планы. Если вы собираетесь связать свою жизнь с наукой, то планируйте поступать в магистратуру и далее – в аспирантуру. В таком случае имеет смысл подумать о выборе темы, которая будет вам интересна все годы обучения на нашем факультете и сможет впоследствии стать темой вашей магистерской диссертации, а в перспективе войдет составной частью и в диссертацию кандидатскую. Естественно, такая тема должна содержать в себе некоторую научную составляю-

щую. Это не может быть, к примеру, очередной Интернет-магазин. Более ориентированы на науку такие темы, как, например: проблемы машинного перевода с иностранных языков, математическое моделирование в экономике, теоретические проблемы компиляторов, разработка проблемно-ориентированного языка программирования. Известно, что российский 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.

Еще один вариант построения вашей карьеры – отправиться на производство в качестве специалиста в области информационных технологий. Как нам видится, в этом случае у вас есть еще целый ряд направлений. Вы можете стать программистом-разработчиком, системным администратором, менеджером по продажам компьютерной техники или программного обеспечения.

Если вас интересует только программирование (или вам так кажется) и вы хотели бы заниматься серьезными разработками, тогда вам нужно подумать над тем, что вам интереснее – системное или прикладное программирование, операционные системы или базы данных, Интернет-технологии или компьютерная графика. Вам необходимо как можно скорее наращивать вашу квалификацию. Имеет смысл выйти за рамки стандартной образовательной программы и самостоятельно познакомиться с какой-то новой технологией или глубоко изучить какую-либо традиционную технологию, например, Remote Procedure Calls в операционной системе UNIX. В этом случае итогом вашей работы может стать не только новая программная разработка, но также и 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 вуз. Конечно, такие нестандартные темы требуется более детально обсудить с преподавателем.

Если в своих мечтах вы видите себя всемогущим системным администратором, повелителем сетевых протоколов, от всевидящего ока которого не ускользнет ни одна проблема в подведомственной вам компьютерной сети, то это также должно отразиться на выборе темы курсового проекта. Одной из возможных тем может быть такая: мониторинг работы компьютеров в локальной сети учебной аудитории. Цель работы – написать программу (или комплекс программ), позволяющий администратору учебной аудитории следить за работой всех подведомственных ему компьютеров со своего рабочего места, т. е. дистанционно. При этом он должен иметь возможность устанавливать необходимые программные продукты, контролировать заполнение жестких дисков, своевременно определять наличие проблем в локальной сети и т. д. Пользовательский интерфейс можно создать на основе Web-технологий.

Менеджер по продажам компьютерной техники или программного обеспечения не обязан быть прекрасным программистом, однако он должен знать основы программирования. Если вашим призванием как раз и является работа менеджера в сфере информационных технологий, то в этом случае, вероятно, имеет смысл выбирать традиционную тему курсового проекта, не требующую знаний, выходящих за рамки стандартной образовательной программы. Возможно, одной из таких тем могла бы стать разработка Интернет-магазина.

Если же вы поняли, что информационные технологии не смогут стать для вас делом всей жизни, и собираетесь сменить сферу вашей деятельности, то не стоит совершенно отказываться от попыток приобрести хотя бы минимальные знания в области программирования. Конечно, получить отличную оценку за курсовой проект вы вряд ли сможете, но приобретенные знания, тем не менее, дадут вам конкретные конкурентные преимущества перед теми людьми, которые не изучали информационные технологии на факультете информатики. Поскольку в наши дни компьютер стал спутником практически всех профессий, то у вас будет преимущество перед «некомпьютерными» коллегами, где бы вы ни работали в будущем.

2. Ваши интересы помимо программирования. Вам было бы интереснее работать над своей программой, если ее тема была бы созвучна вашему хобби или какому-то серьезному занятию. Эти занятия могут дать богатую пищу для размышления о выборе интересной темы для написания прикладной программы.

Приведем лишь несколько примеров:

– математика (конечно же, математика идет под номером один);

– физика (например, моделирование физических процессов в различных областях: оптика, гидравлика, электричество и т. д.);

– экономика (в этой области можно сделать интересную работу, поскольку в настоящее время много экономических данных доступно в сети Интернет);

– прогнозирование будущего в социально-экономической сфере (тут не обойтись без математики);

– спорт (если вы занимаетесь спортом, то можно создать программу, которая помогала бы вам в планировании тренировочного процесса, позволяла бы вести учет ваших выступлений на различных соревнованиях и личных рекордов в различных упражнениях);

– демография (моделирование процессов изменения численности населения в глобальном и локальном масштабе);

– автомобили и автотранспорт (моделирование движения автомобиля в различных дорожных условиях с учетом действия на автомобиль всех физических сил, моделирование городских транспортных потоков).

3. Объем и уровень ваших знаний в настоящий момент времени. Допустим, тема выбрана. Возникает закономерный вопрос: хватит ли у вас квалификации эту тему реализовать? Конечно, ваша квалификация растет,

но будет ли скорость ее роста достаточной, чтобы в нужный момент времени достичь требуемого уровня? Выполнение курсовой работы по выбранной теме может потребовать от вас расширения и углубления знаний в уже знакомой вам сфере, а может потребовать и изучения каких-то новых для вас технологий, языков программирования, систем управления базами данных и т. п. Принимаясь за самостоятельное изучение новой технологии, нужно трезво оценить свои силы. Курсовая работа имеет конкретный срок завершения, поэтому, если вы выберете слишком сложную тему, вы не успеете завершить работу к сроку. Ведь оценка вам будет выставлена в итоге не за то, что вы изучили, например, самый перспективный язык программирования, а за ваши собственные разработки, выполненные на этом языке. Хотя, конечно, умение самостоятельно осваивать новые знания является очень ценным качеством специалиста.

4. Ваше здоровье, ваша усидчивость, терпение и готовность к затратам времени. Реализация сложной темы может потребовать очень много времени, значительную часть которого вам придется провести за компьютером. Программирование – это тяжелый труд. Часами просиживать за компьютером, играя в компьютерные игры или общаясь с партерами в социальной сети, могут многие люди. Но далеко не каждый человек сможет так же часами сидеть за компьютером, напряженно всматриваясь в каждую строчку исходного текста в поисках трудноуловимой логической ошибки. Подумайте, сможете ли вы уделять курсовой работе столько времени и сил, сколько нужно для реализации выбранной темы.

5. Реализуемость темы в заданные сроки. Пусть тема вами выбрана, квалификация у вас – высокая, желание ее совершенствовать – неуемное, любовь к программированию – фанатичная, готовность к работе за компьютером – круглосуточная. Казалось бы, успех гарантирован. Но есть еще один важный момент – это масштаб предстоящей работы. Нужно соотнести масштаб работы и количество времени, которым вы располагаете. Даже если считать один день за два – день плюс ночь, то нужно понимать, что в таком режиме невозможно программировать очень долго. Вы должны помнить, что курсовой проект – это учебная работа. Срок ее завершения назначается не вами и даже не вашим начальником или вашим заказчиком, как это было бы в реальной работе. С начальником или заказчиком в ряде случаев можно договориться о переносе сроков. Однако срок завершения курсового проекта назначается деканатом, а деканат работает в рамках образовательных стандартов и инструкций, поэтому договориться с деканатом о переносе срока нельзя. Следовательно, нужно планировать свою работу таким образом, чтобы к назначенному сроку у вас был работающий программный продукт. Пусть качество этого продукта будет несколько хуже, чем вы хотели бы и могли бы добиться, если бы у вас было больше времени. Но времени зачастую не хватает. Если у вас будут самые прекрасные идеи, но они не будут реализованы в программном коде, то, к сожалению, хорошей оценки вы не получите. Поэтому в условиях ограни-

чения времени имеет смысл выбирать не самые эффектные и красивые, но при этом весьма трудоемкие, решения, а те решения, которые работают – просто работают. При этом нужно стараться проектировать исходный код таким образом, чтобы сохранялась возможность переработки отдельных модулей или процедур на тот случай, если в конце семестра у вас останется немного времени. Конечно, не нужно понимать этот совет, как поощрение использования примитивных алгоритмов и структур данных, которые вы включите в свой проект, даже не попытавшись найти более грамотное инженерное решение.

6. Степень новизны вашей темы, наличие аналогичных программных разработок. Конечно, хорошо бы всегда и во всем быть первым и для курсового проекта придумать оригинальную тему, до сих пор никем еще не разработанную. Однако выбор темы, которая уже была кем-то освоена ранее, не означает, что вы поступаете бесперспективно. Во-первых, вы можете поучиться у более опытного специалиста, изучая его программный код. Ведь, к примеру, художники на ранних этапах своего становления просто копируют работы мастеров, изучая их манеру письма. Во-вторых, изучив один вариант решения задачи и оценив его сильные и слабые стороны, возможно, вы будете в состоянии предложить другое решение. Оно может быть либо более простым в реализации, либо более компактным в смысле объема исходного текста, либо может использовать другую библиотеку стандартных процедур, либо чем-то еще будет отличаться от реализации, предложенной ранее. А в-третьих, есть много примеров программных продуктов, предназначенных для выполнения одних и тех же функций. При этом каждый из однотипных продуктов имеет своих сторонников. Это и браузеры, и файловые менеджеры, и текстовые редакторы и т. д. и т. п. Кто знает, возможно, ваша разработка со временем сможет составить конкуренцию уже известным программным продуктам.

Однако если вы все же выбираете действительно новую тему, по которой вы не смогли найти аналогичных разработок, нужно быть вдвойне осторожным при оценке реальных сроков завершения проекта. Если, например, вам потребуется использование какой-либо библиотеки процедур, то нужно оценить сложность ее использования, ознакомившись с примерами программ, входящими в дистрибутивный комплект библиотеки. Если вам необходимо знание тех или иных сетевых протоколов или новых информационных технологий, оцените объем технической документации, которую вам придется прочитать, прежде чем вы сможете приступить к проектированию.

Если же тема вам интересна, она звучит привлекательно, но вы даже не знаете, с чего начать, какими технологиями воспользоваться, то посоветуйтесь с преподавателем. Возможно, что вы переоцениваете свои силы и имеющийся у вас уровень квалификации.

7. Наличие информации по выбранной теме. Сможете ли вы найти техническую и другую информацию, которой будет достаточно, чтобы

приступить к реализации выбранной вами темы? Возможно, вам понадобится описание нового для вас языка программирования или специальной библиотеки процедур (например, математических функций). На каком языке приведено это описание: на русском или на английском? Если оно на английском языке, то сможете ли вы читать это описание с такой скоростью, чтобы его прочтение не стало для вас делом всей жизни?

Конечно, первоисточником технической информации является техническая документация. К ней, в частности, относятся:

– описания новых информационных технологий, предоставляемые их разработчиками на своих сайтах в открытом доступе (например, описание языка XML на сайте www.w3.org);

– описания программных продуктов, поставляемые в составе дистрибутивного комплекта;

– различные стандарты, например, те, которые регламентируют работу CGI-скриптов (множество стандартов, касающихся различных Web-технологий, можно найти на сайте www.ietf.org);

– исходные тексты свободно распространяемых программных продуктов с открытым исходным кодом (например, на сайте sourceforge.net).

Попытайтесь поискать ссылки на такую информацию в Интернете на различных форумах, не забывайте и о книгах.

Вам могут потребоваться описания математических и других методов, если вы планируете использовать или реализовывать их в вашем программном продукте. Как правило, такие методы подробно описаны в различных книгах.

8. Возможность дальнейшего развития темы. Независимо от ваших перспективных планов (см. п. 1), в вашей студенческой жизни будет еще не одна курсовая работа. Начинать каждую из них с мучений по поводу выбора темы, наверное, нерационально. Поэтому имеет смысл выбирать тему с учетом возможности ее дальнейшего развития. Конечно, при изучении различных учебных дисциплин может и не получиться показывать преподавателям один и тот же программный продукт. Но если эти программные продукты будут хотя бы немного соприкасаться, стыковаться друг с другом, то вам уже будет легче. А к моменту выполнения вашей первой выпускной квалификационной работы (бакалаврской) у вас уже будут конкретные наработки, как алгоритмические, так и технологические. Ваша жизнь станет значительно более спокойной, а квалификационная работа получится гораздо более достойной.

В дополнение к уже сказанному можно дать еще ряд советов по решению задачи выбора темы курсового проекта.

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

2. Посмотрите курсовые проекты 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.

3. А может быть, вы уже что-то программируете для себя? Тогда можно обсудить вашу программу с преподавателем. Возможно, она вполне подходит на роль курсового проекта.

4. Конечно, авторы этих строк отдают себе отчет в том, что некоторые из вас ставят перед собой скромные цели – образно говоря, выжить, а в качестве курсового проекта сделать хоть что-нибудь. В этом случае можно посоветовать сделать работу с использованием Интернет-технологий. Эти технологии сейчас очень актуальны, и знания в этой области еще могут вам когда-нибудь пригодиться.

5. 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. При выполнении курсового проекта иногда допускается работа вдвоем или даже втроем. Подбирая себе коллег, учитывайте 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 кажутся скучными, лучше поискать себе других партнеров. Помните: вы будете делать общее дело, и поэтому будет лучше, когда это дело интересно всем участникам творческой группы.

2.2. Формулирование названия курсового проекта

Говорят, как корабль назовете, так он и поплывет. Конечно, курсовой проект в сфере информационных технологий очень отличается от проекта корабля, но, тем не менее, роль, которую играет название вашей работы, значительна.

Тема и название – это не одно и то же. В толковых словарях сказано, что тема – это *предмет* описания, изображения, исследования, выступления, дискуссии. Очевидно, что в силу богатства выразительных средств русского языка одну и ту же тему можно сжато описать несколькими способами, т. е. дать ей разные названия. Например, название самого известного романа Л. Н. Толстого – «Война и мир». Но если мы скажем, что темой романа являются – буквально – война и мир, то это будет значительным упрощением или даже примитивизацией содержания романа. При такой формулировке темы можно подумать, что эта книга – о военном ис-

кусстве и о том, как правильно заключать мир с неприятелем. Аналогично, темой пьесы М. Горького «На дне» *не является* жизнь обитателей подводного царства. Таким образом, на основе одного только названия художественного произведения не всегда можно даже приблизительно предположить, о чем идет речь в этом произведении, т. е. какова его тема.

Перейдя от высокой литературы к нашим технологиям, приведем пример. Пусть темой вашего проекта будет файловый менеджер (коих уже написано великое множество). Тема вполне понятная. Можно даже оставить эту формулировку в качестве названия – файловый менеджер. Однако если посмотреть на существующие программы этого класса, то все они имеют какое-то *другое* название: Norton Commander, Demos Commander, Midnight Commander и т. д. На основе одного только такого названия невозможно сказать что-то определенное о назначении этих программ. Конечно, сейчас мы все прекрасно знаем это назначение. Но если бы это были совершенно неизвестные программы, то «вычислить» их назначение на основе только названия было бы весьма трудно.

Можно предложить компромиссный вариант, например: файловый менеджер «Русский Навигатор». В таком комплексном названии отражен как вид программы, так и ее фирменное название. На наш взгляд, для курсового проекта это тоже не самый лучший вариант. Кто знает вашу программу? Кто ей пользуется? Скорее всего, несколько человек, включая вас. Но если вы выбираете для своего курсового проекта такое вот «рыночное» название, то вряд ли оно украсит приложение к диплому. В это приложение вписываются не только все ваши оценки за экзамены и курсовые проекты, но еще и темы курсовых проектов. Хорошо, если через два-три года, к моменту завершения учебы ваш программный продукт станет по-настоящему известным. А если нет? Представьте себе в вашем приложении к диплому такое:

> Технология программирования
> Курсовой проект на тему: файловый менеджер «Русский Навигатор»

О чем скажет эта запись работнику кадровой службы при приеме вас на работу? Пожалуй, о вашем большом самомнении. Уж лучше тогда назвать работу бесхитростно: файловый менеджер. И все. Тем более что никто вам не запрещает в курсовом проекте написать просто – файловый менеджер, а распространяя вашу разработку в сети Интернет, назвать ее «Русский Навигатор». В случае если ваш «Русский Навигатор» будет иметь ошеломляющий успех у тысяч пользователей, вы сможете с гордостью сказать тому самому работнику кадровой службы, что это и есть ваш курсовой проект.

Подводя итог рассуждениям, можно сказать так: то, что уместно для произведения художественной литературы или для программного продукта, выводимого на рынок, может оказаться совершенно неподходящим для учебной работы. Беря в руки книгу, название которой привлекло нас своей

необычностью, мы читаем аннотацию и введение, получая тем самым первое представление о теме этого произведения. Беря в руки ваш диплом с приложением, работник кадровой службы какого-нибудь серьезного предприятия или организации читает только названия ваших курсовых проектов. Он не читает техническую документацию, поэтому сами названия должны давать ясное представление о темах ваших курсовых проектов. Эти названия должны говорить, в какой сфере вы имеете наиболее высокую квалификацию.

Попробуем предложить критерии для оценки качества названия курсового проекта. Итак, название должно:

– отражать суть работы;

– соответствовать объему работы;

– быть благозвучным;

– быть кратким и емким;

– быть русскоязычным.

Рассмотрим ряд гипотетических примеров названий программных продуктов, разрабатываемых в рамках курсового проекта.

1. **База данных.** Название является неудачным, поскольку оно слишком широкое, представляет собой наименование целого направления в информационных технологиях, а не конкретную разработку.

2. **База данных «Аптека».** Это название уже более конкретное. Однако оно также не очень удачное. В нем можно увидеть, по меньшей мере, два изъяна. Во-первых, оно подчеркивает, что результатом разработки является именно база данных (структура базы данных и сами данные), при этом ничего нельзя предположить о прикладных программах, необходимых для обслуживания и использования этой базы данных. Созданы такие программы или нет? Скорее всего, должны быть созданы. Однако это не следует из названия явно, поэтому приходится догадываться. Во-вторых, термин «Аптека» слишком абстрактный. Поясним это утверждение. Предположим, что в рамках компьютеризации деятельности аптеки в принципе возможна реализация следующих подсистем (и не только их):

– учет наличия и продаж лекарств;

– прогнозирование потребности в лекарствах и планирование заказов лекарств;

– учет финансовых результатов деятельности аптеки;

– кадровый учет в аптеке (или, как сейчас обычно говорят, управление персоналом).

Тогда возникает вопрос, что включает в себя термин «Аптека»: какую-то одну из перечисленных подсистем, две из них или все перечисленные подсистемы?

Можно предложить такой вариант названия (конечно, он может быть уточнен в зависимости от фактического состава подсистем): информационная система учета и планирования продаж лекарств в аптеке. Из такого названия ясно, что предполагается не только база данных, без которой не

бывает настоящей информационной системы, но счастливый пользователь увидит и прикладные программы, которые будут созданы для работы с этой базой данных. При этом пользователю (и лицу, которое будет знакомиться с приложением к вашему диплому) станет понятно, что управление персоналом в данном программном продукте не реализовано. Плохо это или хорошо, нужна эта подсистема или нет – это уже другой вопрос. И наконец, название говорит о том, что программный продукт предназначен для аптеки, а не для организации, продающей лекарства оптовыми партиями.

3. **Информационная система для учета наличия и продаж лекарств, прогнозирования потребности в лекарствах и планирования заказов лекарств, учета финансовых результатов деятельности и управления персоналом аптеки.** Такое название слишком длинное. Даже автор вряд ли сможет воспроизвести его на память. Целесообразно подумать о сокращении названия до разумных пределов. Кстати, что считать разумным пределом? Психологи говорят, что число понятий, объектов, терминов, которыми человек способен без затруднений оперировать, равно семи (возможно, плюс-минус два). Наверное, название, состоящее из семи-восьми слов, является посильным для восприятия. Поэтому для программного продукта, реализующего все перечисленные подсистемы, предложим такой вариант названия: комплексная информационная система управления деятельностью аптеки.

Если вы выберете предложенный нами вариант названия, но в вашем программном продукте реализуете только подсистему учета продаж лекарств, то будет иметь место несоответствие между названием продукта и реальным его содержанием. Название не будет соответствовать объему работы, оно будет, образно говоря, шире содержания.

4. **Apteka 1.0.** Названия хуже, наверное, быть не может. Во-первых, из названия вообще не ясно назначение программного продукта и объем сделанной работы. Во-вторых, название написано на русском языке, но латинскими буквами. Что это дает для понимания сути работы? В-третьих, указан номер версии. Вряд ли номер версии целесообразно указывать в названии учебной работы. Что он показывает? Может быть, степень зрелости проекта? Представьте себе, что тема вашего проекта – «Apteka 4.0», а оценка, полученная вами – «удовлетворительно». Судя по номеру версии, продукт находится на весьма продвинутой стадии развития. Но судя по оценке, качество продукта все еще низкое. Если вам хочется показать, что вы являетесь продуктивным программистом, то вы можете все версии вашего программного продукта выложить на своем сайте в сети Интернет.

3. Требования к выполнению и оформлению работы

Поскольку курсовой проект является учебной работой, то необходимо сформулировать конкретные требования к объему работы и составу технической документации, которая должна быть оформлена в результате выполнения проекта.

3.1. Общие требования

3.1.1. Состав документации и общие требования к ее оформлению

Сначала перечислим все разделы документации, которые должны быть представлены в отчете по курсовому проекту.

1. Титульный лист.
2. Содержание.
3. Введение.
4. Состав творческой группы.
5. План выполнения разработки.
6. Документ-концепция.
7. Архитектура и системные требования.
8. Технический проект.
9. Исходные тексты программ.
10. Методика тестирования.
11. Руководство пользователя.
12. Руководство программиста.
13. Заключение.
14. Список использованной литературы.
15. Приложения.

Общие требования к оформлению документации по курсовому проекту следующие.

1. Текст следует писать литературным техническим языком. Не допускается использование компьютерного жаргона, шуток, просторечных слов и выражений.

2. Необходимо использовать шрифт Times New Roman или Arial, размер – от 10 до 14 пунктов. Номера страниц следует проставлять внизу листа посередине строки. Не следует использовать пестрые цветовые решения для представления текста: это технический документ, а не рекламный проспект.

3. Не следует включать в документацию пространные цитаты из учебников или материалы учебного характера из сети Интернет. Помните: это проект, а не реферат. Вы – не составитель, вы – автор, вы – разработчик. Основной упор нужно сделать на изложение *ваших* идей и техниче-

ских решений, а не на пересказ чужих мыслей, пусть даже и самых гениальных.

4. Документацию нужно представить в электронном виде в формате doc или pdf (кроме исходных текстов программ). При этом можно выбрать один из двух вариантов компоновки документации:

– все разделы документации включить в единый файл;

– крупные разделы документации (например, «Документ-концепцию») представить в виде отдельных файлов, а в соответствующих разделах главного документа сделать ссылки на эти файлы. Например, в разделе «Документ-концепция» главного документа в этом случае необходимо написать:

Данный раздел документации представлен в виде отдельного документа Concept.doc.

5. Исходные тексты программ нужно представить именно в том виде, в котором они используются для компиляции в системе программирования (например, тексты на языке C/C++) либо для запуска в операционной системе (например, тексты на языке Perl). Не нужно включать исходные тексты в документ формата doc или pdf.

6. После завершения работы нужно предоставить преподавателю не только проектную документацию, но также и исходные тексты программ для размещения вашего курсового проекта на FTP-сервере факультета.

3.1.2. Объем работы

Требования к объему исходных текстов необходимы по двум причинам: во-первых, это учебная работа, преследующая учебные цели, и она должна быть регламентирована не только по оформлению, тематике и другим показателям, но также и по объему; во-вторых, вы должны знать минимальные требования, невыполнение 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.

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 программного кода операционных систем или систем управления базами данных (хотя, конечно, такие проекты разрабатываются коллективно). Даже простая в алгоритмическом отношении, но

большая по размеру программа в чем-то гораздо сложнее изощренной, но *маленькой* по объему. Нужно уметь видеть все связи между модулями, все особенности их взаимодействия. Такие умения не вырабатываются при написании небольших программ, пусть даже и реализующих очень сложные алгоритмы.

Э. Синк, автор книги «Бизнес для программистов», пишет: «Один из моих излюбленных приемов – спрашивать, сколько строк кода написал человек за всю свою карьеру. Все отвечают по-разному. Одни говорят, что не имеют ни малейшего понятия. Другие говорят мне, что это дурацкий вопрос, и с жаром приводят все доводы, доказывающие, что умение программировать не исчисляется строками кода. Ну, хорошо, допустим. И все равно мне нравится этот вопрос. Я уверен, что хорошим программистом можно стать только в процессе написания кода. Чем больше, тем лучше. Именно поэтому я хочу знать, сколько кода у вас за плечами».

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**.

В книге Э. Реймонда «Искусство программирования для UNIX» приводится высказывание Кена Томпсона, являющегося автором операционной системы UNIX: «В один из наиболее продуктивных дней я удалил тысячу строк кода». Это высказывание лишь на первый взгляд кажется парадоксальным.

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

Итак, подведем итог обоснованиям. Цель – научиться писать большие программы. Способ достижения цели – выбор плодотворной темы, избавляющей вас от необходимости добавлять в программу совершенно не нужные, надуманные функции ради того, чтобы получить заветные две или три тысячи строк текста.

Конкретные количественные требования к объему исходных текстов сформулирует преподаватель с учетом сложности вашей темы, выбранного языка программирования и других критериев. Мы же предлагаем ориентировочные требования. Объем исходного текста программы с комментариями (при коллективной работе – в расчете на одного участника творческой группы) должен составить не менее:

– 500 строк на оценку «удовлетворительно»;

– 1000 строк на оценку «хорошо»;

– 3000 строк на оценку «отлично».

При этом учитывается только тот исходный код, который написан вами вручную. Код, сгенерированный автоматически различными генера-

торами кода, не учитывается. Он не учитывается не потому, что такие генераторы не нужны или вредны, а потому, что это – учебная работа. Одной из ее целей является развитие у вас способности писать исходный код самостоятельно: в реальной работе невозможно полагаться только на генераторы кода, ведь они применимы далеко не всегда и не везде.

Количество комментариев должно составлять 20–25 процентов от объема исходного текста, т. е. в среднем одна строка на 3–4 строки исходного текста.

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.

3.2. Титульный лист

Оформление титульного листа должно соответствовать традиционным требованиям, которые предъявляются ко всем курсовым работам, выполняемым студентами нашего вуза. Нужно представить следующие сведения:

– полное наименование высшего учебного заведения;

– наименование кафедры;

– тему курсового проекта;

– фамилию и инициалы студента, выполнившего проект (если работа коллективная, то приводятся сведения обо всех авторах работы);

– фамилию и инициалы преподавателя, принявшего работу;

– город и год разработки.

В Приложении 6 приведен пример оформления титульного листа.

3.3. Содержание

Содержание – это также традиционный элемент текста пояснительной записки любого курсового проекта. В состав документации входят как относительно небольшие разделы, так и весьма объемные. Например, раздел «Документ-концепция» состоит из большого числа подразделов и имеет многоуровневую структуру. В содержании этот раздел нужно представить только одной строкой, без деталировки по всем его подразделам, например:

4. Документ-концепция 6

Раздел «Архитектура и системные требования» организован аналогично. Его в содержании также нужно представить только одной строкой, без деталировки по всем его подразделам.

Если ряд разделов документации представлен в виде отдельных файлов, то все равно необходимо отразить эти разделы в содержании. Это позволит другим людям получить полное представление о составе документации. В тексте каждого такого раздела необходимо сделать ссылку на отдельный документ: данный раздел представлен в виде отдельного документа.

3.4. Введение

Введение не должно быть большим, вполне достаточно одной страницы для того, чтобы отразить следующие вопросы.

1. Актуальность выбранной вами темы. Тема актуальна – что это означает? Это значит, что она соответствует требованиям сегодняшнего дня. Она укладывается в русло основных тенденций развития информационных технологий. При этом актуальность темы учебной работы, которой является курсовой проект, можно рассматривать не только с какой-то всеобщей, глобальной позиции, но и с точки зрения вашего профессионального роста. Поэтому возможно, что для вас в настоящий момент более актуальной будет не та технология, о которой сейчас говорить весь компьютерный мир, а какая-то менее яркая технология или область знаний, которая лежит в основе этой суперсовременной технологии. Поэтому реализация выбранной вами темы сможет помочь вам на следующем этапе изучения информационных технологий, будет способствовать становлению вас как специалиста.

2. Цель, которую вы ставите перед собой как результат выполнения курсового проекта. Зачастую цель формулируется примерно так: написать программу для выполнения таких-то функций (например, файловый менеджер). В такой формулировке эта цель выглядит, как самоцель, программа ради программы. Такая формулировка не показывает, что даст вам (а может быть, и обществу) достижение этой цели. Такую цель мог бы поставить перед вами ваш начальник, который думает за вас и определяет ваши задачи и вашу сферу ответственности. Но когда цель перед собой ставите вы сами, то нужно смотреть немного дальше, мыслить все-таки более масштабно. Подумайте, зачем вам именно файловый менеджер, а не, скажем, программа для планирования тренировочного процесса в одном из видов спорта?

Вспомните о ваших перспективных планах, которые мы обсуждали в разделе, посвященном выбору темы. Целью вашего курсового проекта может быть, например, следующее: создать технологическую базу для разработки программного продукта «А», использование которого в сфере «Б» позволит повысить эффективность выполнения операций типа «В». Например: изучить методы создания динамических Web-сайтов с применением технологии Ajax и систем управления базами данных, что позволит

мне принять участие в международной команде разработчиков программного продукта такого-то, предназначенного для решения таких-то задач.

Ведь вашей, если так можно выразиться, суперцелью на данном этапе является повышение квалификации. Поэтому и актуальность работы, и ее цель следует рассматривать именно с этой позиции.

Принято также формулировать и ряд задач, решение которых как раз и служит достижению поставленной цели. Задачами, решение которых позволит достичь вышеприведенной цели, могут быть следующие:

1. Изучить основные конструкции языков программирования Perl и JavaScript, познакомиться с основными возможностями системы управления базами данных PostgreSQL, познакомиться с технологией Ajax.

2. Проанализировать несколько существующих программных продуктов, построенных на основе этого инструментария.

3. Спроектировать и реализовать собственный программный продукт (например, информационную систему «Домашняя библиотека») с применением указанных технологий.

4. Провести тестирование и внедрение (если возможно, конечно) программного продукта в опытную эксплуатацию.

При подведении итогов выполнения курсового проекта в разделе «Заключение» необходимо показать, что поставленные задачи решены и цель достигнута.

3.5. Состав творческой группы

Поскольку важным качеством современного программиста является умение работать в коллективе, то преподаватель может разрешить выполнение курсового проекта не только индивидуально, но и в составе микроколлектива – вдвоем или втроем. В этом случае необходимо включить в текст документации список участников творческой группы с распределением обязанностей между ними. Разработчики могут иметь следующие специализации:

– архитектор;
– администратор;
– ведущий программист;
– программист;
– ответственный за тестирование;
– ответственный за документацию.

Поскольку число видов специализации больше, чем численность вашей творческой группы, то необходимо совмещение функций. Важно учитывать квалификацию и творческие наклонности участников коллектива при назначении им тех или иных обязанностей.

3.6. План выполнения разработки

Когда вы разрабатываете что-то «для себя», то все планы зачастую находятся в виртуальном состоянии в вашей голове. Но курсовой проект является учебной работой, поэтому план проведения разработки необходим.

План строится по принципу «что – кто – когда». В нем отражаются следующие сведения:

– вид выполняемой работы;

– фамилия, имя и отчество разработчика, ответственного за ее выполнение (если ответственных несколько, то нужно указать их всех);

– срок завершения этой работы (этого этапа или этой процедуры, или этого программного модуля и т. д.).

3.7. Документ-концепция

В разных авторитетных источниках предлагаются различные названия документа, создаваемого на первом этапе разработки программного продукта. В книге Д. Леффингуэлла «Принципы работы с требованиями к программному обеспечению» предлагается название «документ-концепция». В книге И. Соммервилла «Инженерия программного обеспечения» предлагается название «пользовательские требования». Отечественные ГОСТы рекомендуют называть такой документ «техническим заданием».

Мы предпочитаем термин «документ-концепция». В Приложении 1 приводится пример такого документа. Главным разделом в нем является раздел «функциональные требования». Они описывают те функции, которые должна выполнять ваша программа. Эти требования называются «пользовательскими», потому что они формулируются на языке, понятном пользователю. Обратите внимание, что каждое требование имеет номер и заголовок, которые позволяют делать ссылки на эти требования в других документах, например, в документе «Архитектура и системные требования».

Типичный пример неверно сформулированных функциональных требований таков: «Программа должна позволять вводить, корректировать, удалять и выбирать информацию из базы данных». И все – на этом требования закончены. Это неправильно потому, что такое требование является настолько обобщенным, что подходит практически для любой информационной системы и не только для нее. Однако на основе такого требования невозможно приступить к дальнейшему проектированию программного продукта, поскольку совершенно не отражена его специфика. Такое требование абсолютно бесполезно!

В «Документе-концепции» есть большой раздел, посвященный нефункциональным требованиям. Структура этого раздела очень сложная,

имеет много уровней иерархии. Возможно, что не все из приведенных нефункциональных требований будут иметь отношение к вашему проекту. Если какое-то из нефункциональных требований не имеет отношения к вашей разработке, то не исключайте данный пункт из документации, а пишите: «не требуется» или «нет требований», или «нет особых требований». Это будет означать, что вы не просто механически отбросили ненужный пункт, а осмыслили его и осознанно написали: «не требуется». Впоследствии у вас будет возможность вернуться к этому пункту и, может быть, пересмотреть свое решение. Если же вы совсем исключите данный пункт из документации, то потом уже вряд ли вспомните о нем, в результате может пострадать качество вашего проекта.

Если отсутствие требований может быть непонятно тому, кто будет читать вашу документацию, то в ряде случаев можно добавлять пояснение: «нет требований, т. к. …».

В этом документе приводятся и требования к руководствам пользователя и программиста. Обратите внимание, что это еще не сами руководства, а только требования к ним.

В конце документа размещены справочные разделы: глоссарий и список сокращений.

3.8. Архитектура и системные требования

Пример такого документа приводится в Приложении 2. На этапе архитектурного проектирования определяются самые общие черты устройства будущего программного продукта. Кроме собственно архитектуры, в документе содержатся и системные требования.

Согласно авторитетным источникам, к которым можно отнести книги Д. Леффингуэлла и И. Соммервилла (их библиографические описания приведены в списке рекомендуемой литературы), системные требования представляют собой детализированные пользовательские требования. На наш взгляд, на практике не всегда получается именно так. В приведенном примере документа «Архитектура» значительная часть пользовательских требований без дальнейшей детализации включена в состав требований, предъявляемых к той или иной подсистеме программного продукта.

Нужно помнить о том, что написание проектной документации не является самоцелью. Назначение этой документации в том, чтобы помочь в процессе проектирования и программирования создаваемой программной системы. Поэтому, на наш взгляд, если пользовательское требование в том виде, в котором оно сформулировано в «Документе-концепции», может использоваться разработчиком для проектирования и программирования, то дальнейшая детализация такого требования нецелесообразна. Конечно, вреда от дальнейшей детализации не будет, но это неоправданно увеличит объем документации и, следовательно, затянет сроки выполнения работы. Такой упрощенный подход к созданию документации, несколько отходя-

щий от строгих требований классиков, допустим в том случае, когда масштаб программного продукта не слишком велик (объем исходных текстов порядка 50 тысяч строк), а авторами «Документа-концепции» и «Архитектуры», а также проектировщиками и программистами являются одни и те же лица (или даже одно лицо).

3.9. Технический проект

В техническом проекте необходимо представить, в первую очередь, основные структуры данных и алгоритмы, которые будут реализованы в программном коде.

При использовании каскадной модели организации разработки в технический проект включаются основные алгоритмы и описания структур данных для всего программного продукта (для всех его подсистем).

При использовании пошаговой модели организации разработки можно на каждом шаге создавать объединенный документ, включающий в себя:

– совокупность основных проектных решений, касающихся структур данных и алгоритмов, разрабатываемых на конкретном шаге;

– пояснения по программной реализации модулей на данном шаге разработки;

– методику тестирования, включающую в себя как тестирование новых модулей или алгоритмов, реализованных на данном шаге, так и тестирование всей программной системы в целом.

Если программным продуктом является информационная система, включающая в себя базу данных, то в состав документации можно дополнительно ввести диаграммы «сущность–связь». Структуру базы данных можно с целью сокращения объема документации не приводить в документе «Технический проект», а представить в виде текстового файла с SQL-операторами для создания таблиц и других объектов базы данных (индексов, триггеров и т. д.). Такой файл может затем быть использован непосредственно для генерирования структуры базы данных. В Приложении 5 приведен пример такого SQL-файла.

В Приложении 3 приведен пример документа, составляемого при использовании пошаговой модели организации разработки. Для представления алгоритмов в нем использован так называемый псевдокод. Конечно, это дело вашего вкуса и предпочтений: можно использовать и блок-схемы.

3.10. Исходные тексты программ

Сначала поговорим об оформлении исходных текстов ваших программ, а затем – о том, что нужно отразить в этом разделе документации.

Исходные тексты программ должны быть оформлены единообразно. С этой целью рекомендуется разработать свой собственный стандарт ко-

дирования или воспользоваться одним из известных стандартов. Одним из примеров является стандарт GNU (www.gnu.org/prep/standards/). В среде операционной системы FreeBSD можно также увидеть аналогичный документ, введя команду

```
man style
```

Если вы используете заимствованный стандарт кодирования, то нужно отразить это в п. 6.2.3 «Документа-концепции» («Требования к стандартам»). Если вами разработан свой собственный стандарт кодирования, то нужно включить его в документацию в качестве приложения (и отразить это в содержании). В Приложении 4 приведен пример стандарта кодирования для языка Perl.

Если вы не используете никакого стандарта кодирования, то все равно необходимо соблюсти минимальный набор требований к оформлению ваших исходных текстов. Вот эти требования:

1. В каждом программном файле (модуле) должен содержаться заголовок (в виде комментария), в котором отражается следующая информация:

– название программы;

– название модуля (не имя файла, например, cities.pl, а название, отражающее назначение модуля, например, «Справочник городов»);

– автор модуля;

– дата разработки модуля;

– номер версии модуля (или программы в целом).

Такой заголовок дает представление о том, что данный модуль является частью единой программной системы и имеет конкретного автора (или авторов).

2. Каждая функция или процедура должна быть оформлена надлежащим образом, а именно: должно быть указано ее назначение и описаны все формальные параметры.

3. Ширина текста (или длина строки) не должна превышать ширину экрана, т. е. 80 символов.

4. 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.

Теперь о документации. В разделе «Исходные тексты программ» вы должны представить следующие сведения:

– перечень программных модулей с указанием языка программирования и типа связности;

– перечень всех пар (или групп) взаимодействующих модулей с указанием типа сцепления между ними;

– меры Холстеда для каждого программного модуля;

– общий объем исходных текстов (число строк программного кода);

– общее число таблиц в базе данных и другие сведения, дающие представление о масштабе вашего проекта.

3.11. Методика тестирования

Методика тестирования должна включать методы черного и белого ящиков. Для реализации методов белого ящика должны быть написаны специальные тестовые программы. Для реализации методов черного ящика должны быть подготовлены тестовые наборы данных. Эти наборы по возможности должны быть представлены в виде текстовых файлов, которые можно тем или иным способом подать на вход тестируемой программы. Если автоматизированное тестирование невозможно, тогда нужно описать процедуру проведения ручного тестирования с использованием подготовленных тестовых наборов данных.

Для тестирования методом белого ящика нужно выбрать 2–3 процедуры (модуля), являющиеся наиболее важными для правильного функционирования программы.

По результатам тестирования должен быть сформирован отчет, содержащий сопоставление ожидаемых результатов с фактическими. Результаты сопоставления можно представить в виде таблицы.

Важное требование к методике тестирования – повторяемость (воспроизводимость) всех тестов при защите курсового проекта.

3.12. Руководство пользователя

В данном руководстве необходимо отразить следующие вопросы:

– порядок запуска программы;

– порядок использования функций программы (можно привести снимки экранных форм);

– описания возможных проблем и способов их решения;

– перечень сообщений, выводимых программой, и способов реагирования на них.

Руководство пользователя предназначено для пользователя, значит, писать его нужно языком, который понятен пользователю. Конечно, круг пользователей зависит от вида программного продукта. Компилятором пользуются программисты, следовательно, руководство для пользователя компилятора будет написано «компьютерным» языком. Напротив, руководство пользователя бухгалтерской программы не должно включать в себя, например, детальное описание физической структуры файлов базы данных, используемых в этой программе. Если это научная программа, то имеет смысл привести методики проведения вычислений, описать порядок интерпретирования результатов, оценки погрешностей и т. п.

3.13. Руководство программиста

В данном руководстве необходимо отразить следующие вопросы:

– перечень программных модулей и конфигурационных файлов, входящих в состав программного продукта (с указанием не только имени каждого файла, но также и его назначения);

– порядок компиляции (если этап компиляции необходим) и установки программы;

– перечень дополнительных программных продуктов, необходимых для функционирования программы;

– необходимые настройки операционной среды и самого программного продукта.

Руководство программиста составляется для программиста и в терминах, понятных программисту. В нем должны быть описаны «внутренности» программы, даны способы устранения неисправностей в ее работе, описана физическая структура файлов данных и т. п.

Если пользователь получает не только открытые исходные коды, но и проектную документацию, то в руководстве программиста можно обойтись без дублирования тех сведений, которые представлены в проектной документации. Курсовой проект – это как раз тот случай: пользователь, в лице которого выступает преподаватель, получает всю проектную документацию, что позволяет вам сократить руководство программиста до разумных пределов.

Можно оформить отдельное руководство по инсталляции в виде текстового файла с именем, например, INSTALL. Целесообразность такого шага обосновывается тем, что не все программные продукты требуют для своего функционирования наличия графической среды, но для прочтения файлов форматов doc или pdf наличие такой среды требуется. Таким образом, может случиться так, что пользователь будет вынужден устанавливать графическую среду только ради того, чтобы прочесть инструкцию по установке программного продукта. Вряд ли стоит так утруждать пользователя.

При наличии отдельного руководства по инсталляции необходимо отразить это в содержании.

3.14. Заключение

В этом разделе документации нужно подвести итог выполненной работы. Главное – это показать, что поставленная цель достигнута. При этом недостаточно лишь заявить: цель достигнута. Необходимо обосновать свое утверждение. Поскольку необходимым условием достижения цели является решение всех задач, поставленных во «Введении», то сначала нужно показать, что все задачи решены, а уже затем переходить к формулированию заключительного вывода по работе.

Факт решения задач должен быть очевиден и для вас, и для преподавателя. Немного конкретизируем это утверждение, взяв за основу те задачи, которые мы сформулировали во «Введении» в качестве примера.

Подтверждением решения первой задачи будет фактическое написание исходных текстов ваших программ на языках программирования Perl и JavaScript с использованием технологии Ajax. Кроме того, вы использовали и СУБД PostgreSQL для управления вашими данными. Вы можете написать кратко о том, что в языке программирования вам понравилось, что не понравилось, что вызвало наибольшие трудности при его изучении и использовании.

В соответствии со второй задачей вы должны были проанализировать несколько существующих программных продуктов, построенных на основе инструментария, выбранного вами для реализации вашего проекта. В рамках этого анализа вы сравнивали функциональные возможности программ. Результаты анализа следовало отразить в «Документе-концепции» (п. 3.2. Конкурирующие программные продукты). В «Заключении» же напишите о том, насколько трудно вам было знакомиться с исходными текстами программных продуктов, которые вы анализировали. Дайте вашу оценку этих исходных текстов (стиль написания программного кода, сложность алгоритмов, наличие и понятность комментариев и т. п.). Если эти программные продукты сопровождались проектной и эксплуатационной документацией, напишите, помогла ли она вам разобраться в архитектуре и функциях этих программных продуктов. Напишите, научились ли вы чему-либо у авторов этих программ, возможно, подсмотрели какой-то интересный прием программирования. Таким образом, здесь вы можете отойти от формализованного описания функций и возможностей рассмотренных вами программ и изложить ваши впечатления о них в свободном формате (разумеется, в корректной форме).

Третья задача – собственно проектирование и реализация ваших идей. Здесь можно кратко пояснить, что из задуманного удалось реализовать, что является вашей главной удачей, а что, возможно, не получилось. При этом желательно объяснить, почему получилось не все задуманное: не хватило времени; не хватило квалификации; выбрали слишком сложный вариант реализации какого-то алгоритма; неправильно спланировали работу, в результате чего пришлось переделывать ряд программных модулей; испытывали трудности с установкой программного обеспечения, выбранного вами для разработки и т. д.

Умение критически оценить полученные результаты поможет вам лучше выполнить следующий курсовой проект и выпускную квалификационную работу.

При этом не нужно думать, что преподаватель снизит вам оценку, прочитав ваши самокритичные умозаключения.

Последняя, четвертая, задача предполагала проведение тестирования вашего программного продукта и внедрение его в опытную эксплуатацию,

если это возможно. Поэтому напишите, как проходило тестирование. Здесь не нужно повторно описывать методику тестирования, которая уже была представлена в соответствующем разделе документации. Желательно написать о трудностях, которые вы преодолевали, о том, как вы придумывали тесты. Скажите несколько слов и о том, кто использует вашу программу: вы сами, ваши товарищи, работники какого-либо предприятия. Возможно, вы разместили свою разработку в сети Интернет.

Теперь вы можете заявить о степени достижения поставленной цели. Цель может быть достигнута полностью, а может – и частично.

На этом ваше повествование может быть завершено.

Мы отдаем себе отчет в том, что рекомендации по написанию «Заключения», приведенные нами, не являются общепринятыми. Поэтому вы можете уточнить у вашего преподавателя стиль написания этого «Заключения».

При выполнении же выпускной квалификационной работы всякие критические высказывания в свой адрес в пояснительной записке *неуместны*. Ведь на защите выпускной работы вы должны предстать в качестве уже сформировавшегося специалиста, поэтому написать в пояснительной записке, что у вас что-то не получилось, значит – признать собственную профессиональную несостоятельность. Так делать, конечно, не нужно.

3.15. Список использованной литературы

При выполнении курсового проекта вы будете использовать различные учебники, руководства, а также электронные ресурсы, найденные вами в сети Интернет. Библиографические описания использованных источников нужно представить в вашей документации. Выполняются библиографические описания в соответствии с целым рядом стандартов, входящих в Систему стандартов по информации, библиотечному и издательскому делу (СИБИД). Вам следует руководствоваться следующими основными стандартами:

ГОСТ 7.1–2003. Библиографическая запись. Библиографическое описание. Общие требования и правила составления.

ГОСТ 7.80–2000. Библиографическая запись. Заголовок. Общие требования и правила составления.

ГОСТ 7.82–2001. Библиографическая запись. Библиографическое описание электронных ресурсов. Общие требования и правила составления.

Эти стандарты представляют собой весьма пространные документы, особенно ГОСТ 7.1–2003. Однако для их практического использования можно ограничиться изучением только примеров, приведенных в заключительной части каждого из этих документов.

В качестве примеров приведем библиографические описания двух полезных книг и одного Web-сайта.

1. Соммервилл, И. Инженерия программного обеспечения [Текст] / Иан Соммервилл ; пер. с англ. А. А. Минько [и др.] ; под ред. А. А. Минько. – 6-е изд. – М. ; СПб. ; Киев : Вильямс, 2002. – 624 с. : ил. – Библиогр.: с. 603–617 (352, 35 назв.). – Предм. указ.: с. 618–623. – Парал. тит. англ. – Перевод изд.: Software engineering / Ian Sommerville. 6th ed. London [etc.] : Pearson Education, 2001. – 3500 экз. – ISBN 5-8459-0330-0 (рус.). – ISBN 0-201-39815-X (англ.).

2. Леффингуэлл, Д. Принципы работы с требованиями к программному обеспечению. Унифицированный подход [Текст] / Дин Леффингуэлл, Дон Уидриг ; пер. с англ. и ред. Н. А. Ореховой. – М. : Вильямс, 2002. – 446, [2] с. : ил. – Парал. тит. англ. – Прилож.: с. 369–438. – Библиогр.: с. 439–440. – Предм. указ.: с. 441–445. – Перевод изд.: Leffingwell, Dean. Managing Software Requirements. A Unified Approach / Dean Leffingwell, Don Widrig. Boston : Addison-Wesley, [2000]. – 3500 экз. – ISBN 5-8459-0275-4 (рус.). – ISBN 0-2016-1593-2 (англ.).

3. Российская государственная библиотека [Электронный ресурс] / Центр информ. технологий РГБ ; ред. Власенко Т. В. ; Web-мастер Козлова Н. В. – Электрон. дан. – М. : Рос. гос. б-ка, 1997– . – Режим доступа: http://www.rsl.ru, свободный. – Загл. с экрана. – Яз. рус., англ.

Обратите внимание на ряд особенностей оформления библиографических описаний. Согласно ГОСТ 7.1–2003, все описание состоит из так называемых областей, а области – из элементов. Знаки препинания, которые используются для разделения этих областей и элементов, называются предписанными. Каждой области описания, кроме первой, предшествует знак точка и тире, который ставится перед первым элементом области. Обратите внимание, что используется именно тире, а не дефис. Вообще, нужно внимательно относиться к выбору между тире и дефисом в ряде ситуаций. Например, при указании диапазона страниц, на которых размещен какой-либо справочный материал, содержащийся в книге, используется тире, а для разделения групп цифр в стандартном номере ISBN используются дефисы.

Для более четкого разделения областей и элементов, а также для различения предписанной и грамматической пунктуации применяют пробелы в один печатный знак до и после предписанного знака (поэтому при просмотре библиографических описаний может показаться, что кое-где поставлены «лишние» пробелы). Исключение составляют точка и запятая – пробелы оставляют только *после* них.

Чтобы лучше понять правила расстановки пробелов в библиографическом описании, можно открыть текст ГОСТ 7.1–2003 в редакторе Microsoft Word и включить режим отображения служебных символов. При просмотре файла в этом режиме пробелы будут представлены в виде точек.

Стандарты предусматривают возможность составления не только полного библиографического описания, но также и краткого, если в этом есть необходимость или если краткого описания достаточно для однозначной идентификации книги, статьи и т. д. В курсовом проекте можно использовать краткие описания. Приведем краткие описания тех же источников.

1. Соммервилл, И. Инженерия программного обеспечения [Текст] : пер. с англ. / Иан Соммервилл. – 6-е изд. – М. ; СПб. ; Киев : Вильямс, 2002. – 624 с. : ил. – ISBN 5-8459-0330-0.

2. Леффингуэлл, Д. Принципы работы с требованиями к программному обеспечению. Унифицированный подход [Текст] : пер. с англ. / Дин Леффингуэлл, Дон Уидриг. – М. : Вильямс, 2002. – 446, [2] с. : ил. – ISBN 5-8459-0275-4.

3. Российская государственная библиотека [Электронный ресурс] / Центр информ. технологий РГБ ; ред. Власенко Т. В. ; Web-мастер Козлова Н. В. – Электрон. дан. – М. : Рос. гос. б-ка, 1997– . – Режим доступа: http://www.rsl.ru, свободный.

3.16. Приложения

В этот раздел можно поместить материалы вспомогательного характера, например, ваш собственный стандарт кодирования.

4. Заключение

Предложенная в пособии структура технической документации (проектной и эксплуатационной) не является – и не может являться – единственно правильной. Она разработана на основе объединения рекомендаций, приведенных в различных литературных источниках, и опыта авторов пособия. Еще раз напомним, что целью проекта является не разработка проектной документации, а разработка программного продукта. Поэтому документация должна служить главной цели – помочь сделать качественный программный продукт. Следовательно, относиться к предложенной в пособии структуре документации нужно творчески. Если специфика вашей разработки требует добавления еще каких-то разделов, то необходимо их добавить. Например, если вы разрабатываете научную программу, то может потребоваться включить в состав проектной документации краткое описание той научной теории или методики, которую вы используете или реализуете в вашем программном продукте.

Возможно, по окончании учебы в вузе вам доведется работать в серьезной организации, в которой принят внутрифирменный стандарт оформления технической документации. Даже если он будет отличаться (и это вполне вероятно) от того, что предложен в нашем пособии, навыки и культура оформления документации, приобретенные при выполнении курсового проекта, позволят вам проводить разработку в соответствии с требованиями, принятыми в конкретной организации.

В приведенном списке рекомендуемой литературы представлены издания не только по программной инженерии, но и по языкам программирования. Поскольку постоянно выходят новые книги, то не стоит ограничиваться только этим списком.

Желаем вам, уважаемый читатель, проектировать и программировать с удовольствием!

5. Рекомендуемая литература

1. Брайант, Р. Компьютерные системы: архитектура и программирование [Текст] : пер. с англ. / Р. Брайант, Д. О'Халларон. – СПб. : БХВ-Петербург, 2005. – 1104 с.

2. Браун, М. Perl. Архив программ [Текст] : пер. с англ. / М. Браун. – М. : БИНОМ, 2001. – 720 с.

3. Канер, С. Тестирование программного обеспечения [Текст] : пер. с англ. / Сэм Канер, Джек Фолк, Енг Кек Нгуен. – Киев : ДиаСофт, 2000. – 544 с.

4. Керниган, Б. Практика программирования [Текст] : пер. с англ. / Б. Керниган, Р. Пайк. – СПб. : Невский диалект, 2001. – 381 с.

5. Кристиансен, Т. Perl: библиотека программиста [Текст] : пер. с англ. / Т. Кристиансен, Н. Торкингтон. – СПб. : Питер, 2001. – 736 с.

6. Леффингуэлл, Д. Принципы работы с требованиями к программному обеспечению. Унифицированный подход [Текст] : пер. с англ. / Дин Леффингуэлл, Дон Уидриг. – М. : Вильямс, 2002. – 446, [2] с.

7. Митчелл, М. Программирование для Linux. Профессиональный подход [Текст] : пер. с англ. / М. Митчелл, Дж. Оулдем, А. Самьюэл. – М. : Вильямс, 2002. – 288 с.

8. Реймонд, Э. Искусство программирования для Unix [Текст] : пер. с англ. / Э. Реймонд. – М. : Вильямс, 2005. – 544 с.

9. Рочкинд, М. Программирование для UNIX [Текст] : пер. с англ. / М. Рочкинд. – М. : Русская редакция ; СПб. : БХВ-Петербург, 2005. – 704 с.

10. Синк, Э. Бизнес для программистов. Как начать свое дело [Текст] : пер. с англ. / Э. Синк. – СПб. : Питер, 2008. – 256 с.

11. Снейдер, Й. Эффективное программирование TCP/IP. Библиотека программиста [Текст] : пер. с англ. / Й. Снейдер. – СПб. : Питер, 2001. – 320 с.

12. Соммервилл, И. Инженерия программного обеспечения [Текст] : пер. с англ. / Иан Соммервилл. – 6-е изд. – М. ; СПб. ; Киев : Вильямс, 2002. – 624 с.

13. Тейнсли, Д. Linux и UNIX:программирование в shell. Руководство разработчика [Текст] : пер. с англ. / Д. Тейнсли. – Киев : BHV, 2001. – 464 с.

14. Харт, Дж. Системное программирование в среде Windows [Текст] : пер. с англ. / Дж. Харт. – М. : Вильямс, 2005. – 592 с.

15. Эбен, М. FreeBSD. Энциклопедия пользователя [Текст] : пер. с англ. / Майкл Эбен, Брайан Таймэн. – Киев : ДиаСофт, 2002. – 736 с.

Приложения

Приложение 1. Документ-концепция

Информационная система «Электронный архив»

Документ-концепция

© Е. П. Моргунов

История исправлений и дополнений

Дата	Версия	Описание	Автор
01.04.2005	0.5	Исходная версия	Е. П. Моргунов
20.06.2009	1.0	Проведены реструктуризация и дополнение документа с учетом рекомендаций, предложенных в книгах И. Соммервилла и Д. Леффингуэлла (их библиографические описания см. ниже)	Е. П. Моргунов

Примечание. Дата выпуска версии 0.5 – приблизительная.

1. Введение

1.1. Цель документа-концепции

Цель данного документа состоит в сборе и анализе исходной информации для разработки, определении высокоуровневых потребностей пользователей и формулировании функций продукта.

1.2. Назначение и общая характеристика продукта

Настоящий продукт предназначен для решения разнообразных задач управления информационным массивом, который накапливается и используется любым человеком в процессе интеллектуальной деятельности (в том числе, и связанной с отдыхом). Информация, интересующая конкретного человека, может содержаться в печатных и электронных изданиях, аудио- и видеоматериалах, рукописях, ксерокопиях и т. д. Зачастую требуется не только отыскать нужную информацию, например, цитату, но также и указать ее источник. Следовательно, возникает необходимость накапливания и структуризации как содержательной информации, ради получения которой и проводится проработка различных источников (их чтение, просмотр, выписывание цитат и т. п.), так и библиографической информации. Ее можно условно назвать метаинформацией.

Программный продукт в технологическом плане будет представлять собой информационную систему, построенную на основе «большой» СУБД и Web-технологий. Это позволит в качестве клиентского места использовать Web-браузер, а при необходимости – организовать удаленный доступ к базе данных через Internet/Intranet. В содержательном плане программный продукт должен выполнять следующие функции:

– электронный каталог библиографических записей о книгах, журналах, статьях и др., имеющихся у пользователя или заинтересовавших его, т. е. попавших в его поле зрения, в сферу его интересов;

– хранилище цитат из проработанных источников;

– система учета экземпляров книг, журналов и других объектов, хранящихся у пользователя, а также, если это необходимо пользователю, в библиотеках или у других лиц;

– механизм связывания библиографических записей с полнотекстовыми документами в электронной форме (при их наличии), позволяющий организовать быстрый доступ к таким документам.

Использование программного продукта позволит:

– экономить время, традиционно затрачиваемое на поиск библиографической информации и цитат в массиве неупорядоченных файлов и печатных материалов;

– экономить время, требующееся на оформление библиографических списков в рефератах, курсовых работах, научных статьях, диссертациях. Удобнее получать всю информацию библиографического характера из единого центра, только один раз затратив время на корректное оформление каждой записи, вновь вводимой в базу данных электронного архива;

– организовать обмен библиографической информацией с другими студентами, аспирантами, коллегами;

– повысить уровень культуры в работе с библиографической информацией.

Сферы применения программного продукта:

– индивидуальное использование;

– коллективное использование, например, в научном коллективе, занимающемся работами по одной тематике.

1.3. Ссылки и использованная литература

1.3.1. Список документов, упоминаемых в документе-концепции

1. Архитектура и системные требования (спецификация).

1.3.2. Список источников, к которым можно обратиться за справками в процессе разработки

1. Стандарты:

– ГОСТ 7.1–2003 «СИБИД. Библиографическая запись. Библиографическое описание. Общие требования и правила составления»;

– ГОСТ 7.11–2004 «СИБИД. Библиографическая запись. Сокращение слов и словосочетаний на иностранных европейских языках»;

– ГОСТ 7.12–93 «СИБИД. Библиографическая запись. Сокращение слов на русском языке. Общие требования и правила»;

– ГОСТ 7.60–90 «СИБИД. Издания. Основные виды. Термины и определения»;

– ГОСТ 7.80–2000 «СИБИД. Библиографическая запись. Заголовок. Общие требования и правила составления»;

– ГОСТ 7.82–2001 «СИБИД. Библиографическая запись. Библиографическое описание электронных ресурсов. Общие требования и правила составления»;

– ОСТ 29.130-97 «Издания. Термины и определения».

2. Система автоматизации библиотек ИРБИС. Общее описание системы [Текст]. – М. : ГПНТБ России, 2002. – 260 с.

1.3.3. Список использованной литературы

1. Соммервилл, И. Инженерия программного обеспечения [Текст] / Иан Соммервилл ; пер. с англ. А. А. Минько [и др.] ; под ред. А. А. Минько. – 6-е изд. – М. ; СПб. ; Киев : Вильямс, 2002. – 624 с. : ил. – Библиогр.: с. 603–617 (352, 35 назв.). – Предм. указ.: с. 618–623. – Парал. тит. англ. – Перевод изд.: Software engineering / Ian Sommerville. 6th ed. London [etc.] : Pearson Education, 2001. – 3500 экз. – ISBN 5-8459-0330-0 (рус.). – ISBN 0-201-39815-X (англ.).

2. Леффингуэлл, Д. Принципы работы с требованиями к программному обеспечению. Унифицированный подход [Текст] / Дин Леффингуэлл, Дон Уидриг ; пер. с англ. и ред. Н. А. Ореховой. – М. : Вильямс, 2002. – 446, [2] с. : ил. – Парал. тит. англ. – Прилож.: с. 369–438. – Библиогр.: с. 439–440. – Предм. указ.: с. 441–445. – Перевод изд.: Leffingwell, Dean. Managing Software Requirements. A Unified Approach / Dean Leffingwell, Don Widrig. Boston : Addison-Wesley, [2000]. – 3500 экз. – ISBN 5-8459-0275-4 (рус.). – ISBN 0-2016-1593-2 (англ.).

2. Описание пользователей

2.1. Виды пользователей и их краткие описания

2.1.1. Ученый

Имеет высокий уровень культуры интеллектуального труда. Читает много разнообразной научной литературы, как периодической, так и непериодической. Пишет научные статьи, доклады для выступления на конференциях и учебно-методические работы.

2.1.2. Аспирант

Стремится стать ученым и старается делать то же, что описано в п. 2.1.1.

2.1.3. Студент

Объем прорабатываемой научной и учебной литературы – меньше, чем у ученого или аспиранта. Однако объем прочитываемой художественной, научно-популярной и развлекательной литературы может быть большим, чем у ученого, т. к. студент имеет больше свободного времени. Кроме того, студент может иметь в качестве хобби, например, коллекционирование аудиозаписей.

2.1.4. «Обычный» человек

Среди людей, не относящихся к категории ученых, также нередко встречаются любители книг, имеющие большие домашние библиотеки. Многие люди накапливают архивы фото-, аудио- и видеоматериалов, которые трудно содержать в порядке без использования компьютерной системы учета.

2.2. Среда пользователя

Большинство пользователей используют операционную систему (ОС) Windows, хотя в настоящее время все большую популярность набирает ОС Linux, относящаяся к классу UNIX-подобных ОС.

Зачастую проработанные литературные источники нигде не фиксируются, выписки, сделанные из них, находятся в разрозненных файлах или на бумажных носителях. Для формирования списков использованных источников (библиографических списков), например, при написании реферата или статьи каждый раз используется метод «с нуля». Поэтому пользователям трудно выполнить библиографические описания в соответствии с ГОСТами

2.3. Основные потребности пользователя

Общая потребность (возможно, не до конца осознаваемая пользователями в силу разных причин): получить инструмент, позволяющий упорядочить всю библиографическую информацию, которой вынужден оперировать пользователь, и, тем самым, сократить затраты времени на эту деятельность. Также важной потребностью может являться создание электронного каталога домашней библиотеки (включающей не только книги, но также и видео-, фото-, аудиоматериалы).

2.3.1. Ученый

Необходима возможность отслеживания выхода новых выпусков периодических изданий и их оперативной проработки, сохранения цитат в базе данных, формирования списков литературы при написании научных статей и учебно-методических работ.

2.3.2. Аспирант

То же, что и для ученого.

2.3.3. Студент

То же, что и для ученого, с той лишь разницей, что студент не пишет учебно-методических работ, но пишет рефераты, курсовые и дипломные работы.

2.3.4. «Обычный» человек

Необходим инструмент, который мог бы помочь в упорядочивании домашней библиотеки, фонотеки, видеотеки.

3. Состояние рынка и конкурирующие продукты

3.1. Характеристика рынка

На исследуемом рынке представлены два типа программных продуктов:

– «большие» профессиональные автоматизированные библиотечные информационные системы (АБИС), такие, как «ИРБИС», «Академия+» и др., предназначенные для использования в библиотеках;

– «малые» информационные системы, предназначенные для домашнего использования. Последние, как правило, являются программами-каталогизаторами, т. е. позволяют вводить в базу данных описания книг и выполнять выборки из базы данных по различным критериям.

«Большие» АБИС не подходят для персонального использования, т. к. они чрезмерно сложны для пользователя, не имеющего специального библиографического образования. Кроме того, они содержат целый ряд подсистем, предназначенных для выполнения функций, совершенно не нужных в домашних условиях (например, определение книгообеспеченности, списание книг, учет читателей и т. д.). Немаловажным фактором является и тот факт, что подобные системы не являются свободно-распространяемыми – они стоят дорого.

«Малые» же информационные системы, напротив, слишком просты. Они, как правило, хорошо выглядят с технологической стороны: имеют возможности настройки интерфейса пользователя, умеют добывать описания книг на Web-сайтах книжных магазинов (например, Amazon.com), позволяют настроить шрифты, добавить новые таблицы в базу данных и новые поля в таблицы и т. д. Однако с точки зрения соответствия стандартам, принятым в библиотечном деле, эти продукты далеки от совершенства.

Таким образом, необходим программный продукт, условно говоря, среднего класса, который сочетал бы в себе сильные стороны программных продуктов обоих типов, но при этом не был бы слишком сложным в освоении его пользователем, не имеющим специальных знаний.

Потенциальный круг пользователей весьма широк: от ученых до аспирантов и студентов, от работников умственного труда до обычных любителей книги, коллекционирующих также видеофильмы и музыкальные произведения.

Основные технологии, применяемые в отрасли для разработки подобных программных продуктов, следующие: языки программирования – Delphi, C/C++, Visual Basic; СУБД – CDS/ISIS (ЮНЕСКО), Microsoft Access и др. «Малые» информационные системы не работают под управлением операционной системы UNIX (Linux, FreeBSD).

3.2. Конкурирующие программные продукты

3.2.1. Общая характеристика

К недостаткам существующих «больших» библиотечных информационных систем (при оценке их с позиции индивидуального использования в домашних условиях) можно отнести следующее:

– большое количество функций, не требующихся обычному пользователю;

– сложная система ввода данных, требующая от пользователя специальных знаний в области библиографической деятельности;

– высокая цена программного продукта;

– отсутствие возможности сделать собственные пометки к проработанным источникам (книгам, статьям и т. д.) и сохранить цитаты из этих источников.

Особенностью «больших» библиотечных информационных систем является то, что библиографическое описание в них формируется алгоритмическим способом в соответствии с библиографическими стандартами из элементарных данных, вводимых пользователем.

К недостаткам существующих «малых» информационных систем можно отнести следующее:

– отсутствие возможности формирования библиографических описаний, соответствующих стандартам (ГОСТ 7.1–2003 или аналогичному международному стандарту). Вместо цельного библиографического описания имеется только набор сведений об объектах, внесенных в базу данных: авторы, заглавие, место издания и т. д. В отчетах эти сведения группируются в некое подобие описания, но оно не соответствует никаким стандартам;

– отсутствие развитой возможности конфигурирования программного продукта (с использованием конфигурационного файла или базы данных). Конфигурирование сводится, в основном, к настройке внешнего представления программы и данных, но оно мало помогает в ускорении работы оператора при вводе данных;

– неразвитость системы вспомогательных средств, сокращающих объем работы пользователя при вводе однотипных данных (подобные средства есть во всех Internet-браузерах);

3.2.2. Система автоматизации библиотек «ИРБИС»

Разработчик: Государственная публичная научно-техническая библиотека России (ГПНТБ). Сайты: http://www.gpntb.ru, http://www.elnit.org.

«Большая» библиотечная информационная система. Очень мощная, сложная, дорогая.

Фрагмент описания программного продукта, взятый с сайта разработчика (стиль и орфография сохранены)

Основные характеристики:

– поддержка произвольного количества баз данных, составляющих Электронный каталог или представляющих собой проблемно-ориентированные библиографические базы данных;

– технология автоматического формирования словарей, на основе которых реализуется быстрый поиск по любым элементам описания и их сочетаниям;

– средства для ведения и использования Авторитетных файлов, баз данных УДК, ББК, ГРНТИ и Тезауруса;

– поддержка традиционных «бумажных» технологий: от печати форм заказа/подписки и листов книги суммарного учета до печати всех видов каталожных карточек;

– технологии, ориентированные на использование штрих-кодов и радиометок на экземплярах изданий и читательских билетах;

– поддержка многоязычия на основе UNICODE, т. е. возможность ввода на любых языках мира;

– поддержка ссылок от библиографических описаний на полные тексты, графические данные и другие внешние объекты (включая ресурсы Интернет);

– средства для создания и ведения полнотекстовых баз данных (электронной библиотеки);

– специальные средства для создания имидж-каталогов по ретрофонду библиотеки на основе графических образов каталожных карточек и автоматического распознавания их текстов;

– средства для перевода пользовательских интерфейсов на другие языки;

– широкий набор сервисных средств, обеспечивающих удобство и наглядность пользовательских интерфейсов, упрощающих процесс ввода, исключающих ошибки и дублирование информации;

– широкие возможности для адаптации к условиям работы конкретной библиотеки, включая средства создания уникальных рабочих профилей для всех категорий пользователей;

– открытость, позволяющая пользователю самостоятельно вносить изменения в широких пределах: от изменения входных и выходных форм до разработки оригинальных приложений.

«Оригинальное программное обеспечение системы написано на Delphi с использованием библиотеки ISIS32.DLL (Bireme, Бразилия). Физическая структура БД соответствует СУБД CDS/ISIS (ЮНЕСКО)» (цитата из описания программного продукта).

3.2.3. Автоматизированная библиотечная информационная система «Академия+»

Разработчик: Центр автоматизированных технологий «Ростехноком». Сайты: http://www.rostechnocom.ru; http://www.academy-plus.ru.

«Большая» библиотечная информационная система. Очень мощная, сложная, дорогая.

Фрагмент описания программного продукта, взятый с сайта разработчика (стиль и орфография сохранены)

Основные характеристики:

– масштабируемая трехуровневая клиент-серверная архитектура;

– независимость от программно-аппаратной платформы (любая аппаратная платформа: IBM PC, SUN и др.; любая операционная система: Windows, UNIX, LINUX);

– независимость от СУБД (любая реляционная СУБД: ORACLE, MS SQL Server, My SQL, PostgresSQL и др.);

– работа системы в Internet и Intranet без ограничения количества пользователей;

– поддержка UNICODE и штрих-кодирования на программном уровне.

3.2.4. BookCAT

Разработчик: FNProgramvare (Норвегия). Сайт: http://www.fnprg.com.

«Малая» информационная система. Самая развитая из систем подобного класса.

3.2.5. Учет книг

Разработчик: «Простой Софт». Сайт: http://www.simple-soft.ru, http://www.prostoysoft.ru.

Фрагмент описания программного продукта, взятый с сайта разработчика (стиль и орфография сохранены)

Основные функции программы

Ведение базы книг, журналов. Каталогизация

В базе данных содержится информация о книгах, журналах. Предусмотрены такие поля как – название, авторы, категория, тип, издательство, серия, формат, год издания, количество страниц, тираж, обложка, ISBN, УДК, № шкафа, № полки, блок, подблок, время добавления и т. д. Для каждой книги показываются все ее читатели (которые читали эту книгу ранее и читают сейчас).

Предусмотрены удобные способы сортировки и фильтрации данных, что позволяет быстро найти нужные книги. Любую таблицу базы можно распечатать, экспортировать в MS Word, MS Excel или текстовый формат CSV. Имеется импорт из других источников данных в формате CSV.

Учет должников по возврату книг, журналов

Система фиксирует информацию о читателях – ФИО, контактная информация, выданные книги, даты выдачи и возврата книг. Контролируя значение поля с датой возврата книги можно легко вести учет должников. Таблица «На руках» показывает список всех выданных на руки книг и журналов.

Функциональные возможности программы

С помощью программы вы сможете делать следующее:

Создавать, изменять, удалять записи, поля, таблицы.

Импортировать данные в любую таблицу базы данных из текстовых файлов.

Удалять дублированные записи с одинаковым названием и автором. Можно настроить по-другому.

Сортировать таблицы по любому полю, включая сортировку по нескольким полям (до 3-х) удерживая клавишу Shift.

Фильтровать таблицу по любому полю, используя следующие операторы: =, >, >=, <, <=, <>, «Содержит», «Не содержит», «Начинается с», «Не начинается с», «Кончается на», «Не кончается на», LIKE, NOT LIKE.

Группировать одинаковые данные в любом поле, когда таблица отсортирована по этому полю (для отмеченных полей в свойствах таблицы).

Помечать записи как «Избранное», тогда они будут отображаться оранжевым цветом. Цвет задается в свойствах таблицы.

Помечать записи как «Мертвое» («Неинтересное»), тогда они будут отображаться серым (или другим) цветом.

Настраивать правила цветовыделения. Вы сами определяете, какие строки, каким цветом и при каких условиях выделять.

Строить дерево по любым полям с произвольным количеством уровней для иерархического отображения данных любой таблицы.

Изменять данные в любом поле (кроме ID и вычисляемых полей) прямо в таблице или в отдельной форме (выбирается в настройках), отмечать несколько записей, удалять, печатать, экспортировать отмеченные.

Изменять или удалять сразу несколько записей в любой таблице базы данных с помощью формы «Групповое обновление».

Создавать новые хранимые поля для таблиц следующих типов: текстовое, числовое, Да/Нет, Дата и время.

Создавать вычисляемые поля для таблиц, например можно создать поле с формулой «[Поле 1] / [Поле 2]».

Создавать вычисляемые поля, значения которых будут браться из других таблиц. Например, можно вывести имя должника из таблицы «Должники».

Создавать новые таблицы с абсолютно такими же возможностями по действиям с ними, как и у любой другой таблицы

Привязывать ниспадающие списки полей к другим таблицам для легкого выбора значений из них при редактировании в таблице или для выбора из других форм при редактировании в форме.

Задавать произвольное количество подчиненных таблиц для любой таблицы, для чего необходимо задать привязку по полям в свойствах таблицы.

Менять порядок следования полей в любой таблице, используя перетаскивание или с помощью формы «Настройки».

Переименовывать поля таблиц и названия самих таблиц в соответствии со спецификой вашего бизнеса. (Точнее говоря, их лейблы.)

Печатать текущее представление любой таблицы с учетом видимости полей, их ширины и порядка.

Экспортировать данные любой таблицы в MS Excel или текстовый CSV-файл с учетом текущего представления таблицы.

Экспортировать текущую запись в MS Word на основе файла-шаблона с закладками, соответствующими названиям полей.

Работать с несколькими файлами баз данных, создавать новые базы данных, разумеется, можно также открывать их с помощью MS Access.

3.2.6. eLibPro

Разработчик: Songs Technologies. Сайт: http://songstech.com.

«Малая» информационная система. Использует формат данных mdb (Microsoft Access).

3.2.7. Librarian Pro

Разработчик: Koingo Software. Сайт: http://www.koingosw.com.

«Малая» информационная система.

3.2.8. HomiStorage

Разработчик: Hominoid Software. Сайт: http://www.hominoid.nm.ru.

«Малая» информационная система.

3.2.9. All My Books

Разработчик: Bolide Software. Сайт: http://www.bolidesoft.com.

«Малая» информационная система.

3.3. Определение позиции продукта на рынке

Для	ученых, аспирантов, студентов и всех других пользователей,
которые	хотят иметь возможность упорядочить свою коллекцию книг, журналов, видео- и аудиоматериалов, рукописей и электронных документов.
«Электронный архив»	является автоматизированной информационной системой для управления домашней библиотекой.
Наш продукт	позволит всегда иметь под рукой качественную библиографическую информацию.

В отличие от	конкурирующих «малых» информационных систем, таких, как BookCAT, «Учет книг» и др.,
наш продукт	разработан с учетом требований стандартов на библиографическое описание, использует «большую» СУБД, имеет развитые средства помощи пользователю для ускорения ввода данных.

4. Функциональные требования (функции продукта)

4.1. Обязательные функции для первой версии

Название «первая» является условным. Номер версии должен быть определен на основе фактического уровня готовности программного продукта. Предположительно это будет версия 0.5.

ПФТ1. Соответствие требованиям ГОСТ 7.1–2003

1. Архитектура программного продукта и структура базы данных должны быть разработаны с учетом требований ГОСТ 7.1–2003, а именно: с учетом наличия определенных областей библиографического описания и элементов этих областей. Причем, необходимо учесть, что ряд элементов в библиографическом описании могут повторяться (например, сведения об ответственности, место издания, примечание и др.).

2. Должна быть обеспечена возможность хранения в базе данных библиографических записей о книгах, статьях, электронных ресурсах, научных конференциях и т. д.

Обоснование. Указанный ГОСТ является основным стандартом, регламентирующим создание библиографических описаний любых объектов: книг, журналов, диссертаций, материалов конференций, статей, нот, карт, рукописей, патентных документов, электронных ресурсов и т. д.

Ссылки на спецификацию. Пп. 3.1, 3.9.

ПФТ2. Ввод библиографических описаний в базу данных в готовом виде

1. Библиографические описания должны вводиться в базу данных сразу в готовом виде, а не формироваться из элементарных фрагментов библиографического описания, как это делается в «больших» АБИС (например, «ИРБИС»). Таким образом, библиографическое описание формируется пользователем, а не программным продуктом.

2. Должна быть предусмотрена возможность ввода в базу данных детальных сведений, соответствующих областям и элементам областей библиографического описания (заглавие, сведения об ответственности и т. д.). Детальные сведения могут быть введены позднее, если пользователь нуждается в этом, но могут и не вводиться совсем.

Обоснование. Предлагаемая схема действий является инверсией по отношению к традиционной схеме. Такая схема позволит реализовать различные способы использования программного продукта. Простейший способ предусматривает только ввод библиографических описаний в целом, без ввода детальных данных по каждой области библиографического описания. Такой подход упрощает и ускоряет работу на начальном этапе, но в этом случае пользователь не сможет выполнять целый ряд поисковых операций в базе данных.

Ссылки на спецификацию. П. 3.9.

ПФТ3. Обработка наличия различных вариантов имен собственных

1. База данных должна быть спроектирована таким образом, чтобы наличие различных вариантов написания фамилий персоналий и наименований организаций, переименований городов и т. п. не препятствовало однозначной идентификации таких объектов пользователем.

2. Пользователь должен иметь возможность при выборе любого из вариантов, представленных в базе данных, получить выборку, включающую и все другие варианты.

Обоснование. Подобные разночтения фамилий персоналий могут появиться, например, при переводе иностранных книг на русский язык (*Вильямс* и *Уильямс*). Поэтому если пользователь производит поиск книг автора по фамилии *Вильямс*, то в выборку должны быть включены и книги этого же автора, но в переводе представленного как *Уильямс*, а в оригинале – как *Williams*. При этом в выборку не должны попасть книги однофамильцев, поскольку это уже различные персоналии.

Ссылки на спецификацию. Пп. 3.7, 3.9.

ПФТ4. Фиксирование результатов проработки источников

1. Должна быть предусмотрена возможность фиксирования даты и результатов проработки источников (книг, журналов, статей и т. д.).

2. Должна быть предусмотрена возможность выставления оценок проработанным источникам. Пользователь должен иметь возможность сам назначить показатели, по которым выставляются эти оценки.

3. Пользователь должен иметь возможность ввести в базу данных содержание (оглавление) проработанного источника.

Обоснование.

Ссылки на спецификацию. П. 3.9.

ПФТ5. Хранение выписок из проработанных источников

1. Должна быть предусмотрена возможность хранения выписок в базе данных.

2. Должна быть предусмотрена возможность связывания таких выписок, которые могут храниться также и в файлах, имеющих различные форматы (Microsoft Word, HTML, ASCII), с соответствующими библиографическими описаниями.

Обоснование. Необходимо избавить пользователя от утомительного поиска нужного файла методом визуального просмотра каталогов операционной системы (как зачастую бывает в действительности).

Ссылки на спецификацию. П. 3.9.

ПФТ6. Система ключевых слов

1. Должна быть предусмотрена система ключевых слов.

2. Должна быть предусмотрена возможность проведения выборки библиографических описаний для каждого ключевого слова.

Обоснование.

Ссылки на спецификацию. П. 3.9.

ПФТ7. Простые средства поиска информации

1. Должны быть предусмотрены простые средства поиска в базе данных и выборки информации из нее. На последующих этапах разработки эти средства должны быть значительно усилены.

Обоснование.

Ссылки на спецификацию. Пп. 3.2, 3.7.

ПФТ8. Учет экземпляров

1. Должна быть предусмотрена возможность описания экземпляров книг, журналов, дисков, рукописей и т. д., находящихся как в собственности пользователя, так и в библиотеках или у других лиц.

Обоснование.

Ссылки на спецификацию. П. 3.9.

4.2. Дополнительные функции для первой версии

ПФТ9. Формирование тематических библиографических списков

1. Должна быть предусмотрена возможность формирования библиографических списков, включающих описания группы объектов (книг, статей и т. д.), содержащихся в базе данных.

2. Списки должны формироваться без повторного ввода библиографических описаний в базу данных (без дублирования данных в ней).

Обоснование. Такие списки могут включать, например, литературу, предназначенную для выполнения одной задачи или характеризующую одну тему, или относящуюся к одному периоду времени (статьи за конкретный год по конкретной тематике) и т. п. При подготовке научной статьи или реферата ученый или студент сможет включить в такой список соответствующие источники и затем перенести сформированный список в статью или реферат в качестве перечня цитируемых источников.

Ссылки на спецификацию. П. 3.9.

ПФТ10. Развитые средства помощи пользователю для ускорения ввода данных

1. Должен быть создан механизм для ускоренного ввода типовых фрагментов текста в поля библиографического описания.

Обоснование. Такими типовыми фрагментами могут быть: фамилии авторов, сведения, относящиеся к заглавию, названия издательств, тексты примечаний и т. д.

Ссылки на спецификацию. П. 3.2, 3.7.

4.3. Будущие функции

ПФТ11. Развитые средства поиска информации

1. Должны быть предусмотрены различные средства поиска в базе данных и выборки информации из нее по различным критериям.

Обоснование.

Ссылки на спецификацию. П. 3.2, 3.7.

ПФТ12. Иерархическая система ключевых слов

1. Система ключевых слов должна быть организована по иерархическому принципу.

2. Должна быть предусмотрена возможность выполнения поиска информации в соответствии с требуемым уровнем иерархии ключевых слов.

Обоснование. Реализация этого требования позволит пользователю регулировать объем выбираемых библиографических описаний и точность соответствия запросу. Например, для ключевого слова «Linux» словом-родителем будет ключевое слово «операционная система», а для него, в свою очередь, таким словом-родителем будет «система». Отметим, что в известных нам программных продуктах, используемых в библиотеках, иерархический подход к организации системы ключевых слов не реализован.

Ссылки на спецификацию. П. 3.7, 3.9.

ПФТ13. Формирование библиографического описания из элементарных данных

1. Должен быть реализован и традиционный способ алгоритмического формирования библиографического описания из элементарных данных, предварительно введенных пользователем в области и элементы библиографического описания (заглавие, сведения об ответственности, издатель, место издания, год издания и т. д.).

Обоснование. Реализация этого требования позволит опытному пользователю ускорить процесс ввода сведений в базу данных при полномасштабном использовании программного продукта.

Ссылки на спецификацию. П. 3.7.

5. Основные способы использования программного продукта и сценарии работы с ним

5.1. Простой способ использования программного продукта

Пользователь вводит в базу данных только библиографические описания (полные и краткие либо только краткие). Он не использует систему справочников. Он не вводит в базу данных детальные сведения, содержащиеся в областях библиографического описания (заглавие, сведения об ответственности, издатель, место издания, год издания и т. д.). В таком случае пользователь лишен возможности выполнять б*о*льшую часть процедур поиска сведений в базе данных.

5.2. Полномасштабный способ использования программного продукта

Пользователь использует систему справочников, дополняет их новыми данными по мере необходимости. Он вводит в базу данных полные и краткие библиографические описания (либо только краткие, если описание выполняется не на основе оригинального объекта, а на основе вторичной информации, например, библиографического списка в книге, диссертации, статье). Он вводит в базу данных детальные сведения, содержащиеся в областях библиографического описания (заглавие, сведения об ответственности, издатель, место издания, год издания и т. д.). Также пользователь ведет

учет экземпляров (книг, журналов, дисков, рукописей и т. д.), как находящихся в собственности пользователя, так и находящихся в библиотеках или у других лиц.

6. Нефункциональные требования

6.1. Требования к программному продукту

6.1.1. Требования к инсталляции

Необходимо разработать процедуру инсталляции программного продукта.

6.1.2. Требования к эксплуатации

6.1.2.1. Требования к удобству эксплуатации (практичность)

6.1.2.1.1. Необходимое время подготовки пользователя для достижения минимальной производительности

Приблизительно 1–2 дня. Пользователю необходимо ознакомиться с ГОСТ 7.1–2003. Он может ограничиться только просмотром примеров библиографических описаний, приведенных в Приложении к этому ГОСТу.

6.1.2.1.2. Время выполнения типичных задач или транзакций, осуществляемых пользователем

Ввод полного и краткого библиографических описаний для одной книги (статьи, диска и т. д.) должен занимать не более 10 минут. Ввод дополнительных сведений, в соответствии с областями библиографического описания, должен занимать не более 1 часа. По мере повышения квалификации пользователя эти временны*е* интервалы должны уменьшиться до 5 минут и 20 минут соответственно. Это должно быть достигнуто за счет того, что по мере приобретения навыков работы с программным продуктом пользователь должен постепенно заполнить различные справочники, содержащиеся в базе данных, а также настроить конфигурационный файл, содержащий типовые фрагменты библиографических описаний (имена и отчества авторов, наименования часто встречающихся издательств и т. п.).

6.1.2.1.3. Сравнение практичности новой системы с уже существующими современными системами

Практичность должна быть не хуже, чем у всех систем, перечисленных в документе-концепции.

6.1.2.1.4. Следование соглашениям и стандартам, разработанным для человеко-машинного интерфейса

Пользовательский интерфейс программного продукта должен отвечать, как минимум, требованиям здравого смысла. Хотя это требование неконкретное, но, поскольку автором документа-концепции и разработчиком будет одно и то же лицо, а также в

силу ограниченности времени на разработку, особых требований относительно соответствия стандартам, разработанным для человеко-машинного интерфейса, на данном этапе не предъявляется. В дальнейшем возможны изменения технологии создания пользовательского интерфейса, например, использование каскадных таблиц стилей (CSS).

6.1.2.1.5. Прочие требования к удобству эксплуатации

ПНТ1. Экспорт и импорт данных

1. Должна быть реализована возможность экспорта данных из системы в виде команд языка SQL и импорта данных, представленных в виде таких команд.

Обоснование. Это позволит организовать обмен данными между пользователями предлагаемой системы. Поскольку оформление библиографических записей представляет собой трудную задачу для большинства студентов и аспирантов, то в процессе заполнения индивидуальной базы данных каждого пользователя важную роль может сыграть взаимный обмен данными. В частности, преподаватель, рекомендуя студентам перечень источников для изучения материала конкретной учебной дисциплины, мог бы выдавать им этот перечень сразу в той форме, которая позволяет выполнить его импорт в базу данных каждого студента.

Ссылки на спецификацию. П. 3.10.

ПНТ2. Конфигурирование программного продукта

1. Должна быть предусмотрена возможность конфигурирования программного продукта.

2. Конфигурационные параметры должны описывать, в основном, значения полей документов, принимаемые по умолчанию и позволяющие пользователю ускорить процесс ввода данных.

3. В первой версии программного продукта реализация конфигурирования может быть выполнена на основе использования текстового конфигурационного файла, изменения в который должны вноситься пользователем в текстовом редакторе.

4. В последующих версиях должна быть разработана интерактивная система конфигурирования программного продукта.

Обоснование. Реализация этого требования позволит пользователю ускорить процесс ввода данных, сделать его более удобным и уменьшить число ошибок ввода.

Ссылки на спецификацию. П. 3.8.

6.1.2.2. Требования к режиму эксплуатации

6.1.2.2.1. Доступность (сколько дней в неделю и часов в сутки должна быть доступна система)

Эпизодически, по мере возникновения потребности.

6.1.2.2.1. Прочие требования к режиму эксплуатации

Нет требований.

6.1.2.3. Прочие требования к эксплуатации

Нет требований.

6.1.3. Требования к эффективности

6.1.3.1. Требования к производительности

6.1.3.1.1. Время ответа для транзакции (среднее, максимальное)

Время получения результатов при выполнении выборки библиографических описаний не должно превышать 2–3 секунды.

6.1.3.1.2. Пропускная способность (число транзакций в секунду)

Нет требований.

6.1.3.1.3. Емкость (число пользователей или транзакций, которые может обслужить система)

Нет требований.

6.1.3.1.4. Режимы снижения производительности (допустимые режимы работы при ухудшении параметров системы)

Нет требований.

6.1.3.1.5. Время обновления экрана

Нет требований.

6.1.3.1.6. Время реакции на действия пользователя

Нет требований, т. к. все операции по вводу и корректировке данных не связаны с обработкой больших объемов данных.

6.1.3.2. Требования к ресурсам

6.1.3.2.1. Клиентский компьютер

Оперативная память: объем, достаточный для нормальной работы Web-браузера. Объем дискового пространства: нет требований. Процессор: любой современный процессор.

6.1.3.2.2. Сервер

Оперативная память: объем, достаточный для нормальной работы Web-сервера и сервера баз данных. Объем дискового пространства: приблизительно 100–150 Мб с учетом СУБД PostgreSQL (дополнительный требуемый объем зависит от объема электронных документов). Процессор: определяется требованиями, которые предъявляет СУБД PostgreSQL (см. документацию на текущую версию этой СУБД).

6.1.3.3. Прочие требования к эффективности

Нет требований.

6.1.4. Требования к надежности

6.1.4.1. Вероятность отказа

Нет требований, т. к. от программного продукта не требуется непрерывное функционирование.

6.1.4.2. Частота отказов

Нет требований, т. к. от программного продукта не требуется непрерывное функционирование.

6.1.4.3. Среднее время безотказной работы (среднее время между двумя последовательными сбоями)

Не менее отрезка времени, требующегося для ввода данных об одном объекте библиографического описания (книге, статье и т. д.).

6.1.4.4. Вероятность готовности системы к использованию

Нет требований, т. к. от программного продукта не требуется непрерывное функционирование.

6.1.4.5. Среднее время восстановления после сбоя (отказа)

Нет требований, т. к. программный продукт не является критическим по данному показателю.

6.1.4.6. Вероятность порчи данных при сбое (отказе)

Нет требований, т. к. предполагается, что СУБД обеспечивает откат незавершенных транзакций, гарантируя тем самым целостность базы данных.

6.1.4.7. Прочие требования к надежности

Нет требований.

6.1.5. Требования к переносимости

6.1.5.1. Поддерживаемые операционные системы

6.1.5.1.1. Клиентский компьютер

Операционные системы UNIX (FreeBSD, Linux) и Windows (XP/2003/Vista/7). На клиентском компьютере работает Web-браузер.

6.1.5.1.2. Сервер

Операционная система UNIX (FreeBSD, Linux). В будущем предполагается перенос продукта в среду операционной системы Windows (XP/2003/Vista/7).

6.1.5.2. Объем машинно-зависимых операторов и подсистем

Таких операторов и подсистем нет.

6.1.5.3. Прочие требования к переносимости

Нет требований.

6.1.6. Требования к защищенности

ПНТ3. Аутентификация пользователей

1. При входе пользователя в систему должна выполняться аутентификация.

Обоснование. Это необходимо для организации разграничения полномочий доступа к базе данных.

Ссылки на спецификацию. Пп. 3.1, 3.9.

ПНТ4. Интерактивная система управления разграничением полномочий доступа к базе данных

1. Должна быть разработана интерактивная система для управления разграничением полномочий доступа различных пользователей к базе данных.

Обоснование.

Ссылки на спецификацию. Пп. 3.5, 3.9.

ПНТ5. Ведение журнала операций с базой данных

1. Должна быть предусмотрена возможность ведения журнала всех операций с базой данных, изменяющих ее состояние: ввод записей, их обновление и удаление.

2. Должна быть предусмотрена возможность просмотра истории всех операций с базой данных с применением специального пользовательского интерфейса.

Обоснование. В случае ошибочного изменения или удаления данных пользователь будет иметь возможность «реанимировать» их.

Ссылки на спецификацию. П. 3.9.

6.1.7. Требования к безопасности

При инсталляции программного продукта необходимо сохранять резервные копии всех конфигурационных файлов, принадлежащих другим программным продуктам, в случае внесения изменений в эти файлы.

6.1.8. Требования к удобству сопровождения

Должны быть подробные комментарии в исходных текстах программ.

6.1.9. Требования к точности числовых вычислений

Нет требований.

6.1.10. Прочие требования к программному продукту

ПНТ6. Организация коллективной работы

1. Должна быть предусмотрена возможность коллективного использования программного продукта через локальную сеть или Internet, а также на одном компьютере.
2. Должен быть организован раздельный доступ к данным, принадлежащим различным пользователям.

Обоснование.

Ссылки на спецификацию. Пп. 3.1, 3.9.

6.2. Организационные требования

6.2.1. Выходные требования

6.2.1.1. Сроки разработки и изготовления

Первый работающий вариант программного продукта должен быть готов к 15 сентября 2006 г.

6.2.1.2. Сопутствующая документация

Нет сопутствующей документации.

6.2.1.3. Прочие выходные требования

Нет требований.

6.2.2. Требования к реализации

6.2.2.1. Модель организации разработки

Модель пошаговой разработки.

6.2.2.2. Методы проектирования и документирования разработки

Нет требований.

6.2.2.3. Языки программирования и инструментальные средства

1. Языки – Perl (версия 5.8.x) и JavaScript; СУБД – PostgreSQL (версия 8.x); Web-сервер – Apache (версия 2.2.x). Инструментальные средства: отладчик языка Perl; отладчик языка JavaScript, встроенный в Web-браузер. Библиотеки: интерфейс к СУБД PostgreSQL – pgperl (до версии 7.3 входил в состав дистрибутива СУБД PostgreSQL).
2. Рассмотреть также возможность использования системы управления версиями программного продукта (Subversion, Git и др.).
3. Рассмотреть возможность использования общего интерфейса DBI/DBD к СУБД PostgreSQL.

6.2.2.4. Локализация и интернационализация

Необходимо реализовать поддержку локализации (l10n) и интернационализации (i18n) на более поздних этапах разработки. Поддерживаемые языки: русский и английский.

6.2.3. Требования к стандартам

1. Использовать стандарты кодирования и оформления исходных текстов программ, аналогичные стандартам проекта GNU.
2. Комментарии в исходных текстах программ должны быть написаны на русском языке.

6.2.4. Требования к лицензированию

Программный продукт должен распространяться по лицензии, аналогичной лицензии GNU GPL.

6.2.5. Требования к дистрибуции (распространению) и вопросы цены

1. Распространение продукта будет производиться, в основном, через Web-сайт http://www.morgunov.org.
2. Программный продукт будет распространяться бесплатно в исходных текстах.

6.2.6. Вопросы авторских прав

Авторские права принадлежат Моргунову Е. П.

6.3. Внешние требования

6.3.1. Требования к взаимодействию с другими системами

1. Необходимо организовать взаимодействие с СУБД PostgreSQL, которая будет использоваться для хранения данных, и с Web-сервером Apache.
2. Взаимодействие с другими *прикладными* программными продуктами не предполагается.

6.3.2. Юридические требования

Для разработки программного продукта должны использоваться только инструменты с лицензией, разрешающей их бесплатное использование.

6.3.3. Этические требования

Нет требований.

6.4. Прочие требования

Нет требований.

7. Требования к документации

7.1. Руководство пользователя

1. Должно содержать:

– основы методики составления библиографического описания (на основе ГОСТ 7.1–2003);

– примеры библиографических описаний;

– описание процедур работы с продуктом;

– глоссарий;

– индекс для поиска по тексту.

2. Форма руководства – электронная, формат – pdf, html. Должна быть предусмотрена возможность вывода руководства на печать (полностью или отдельных разделов).

7.2. Руководство программиста

Должно содержать следующие сведения (но не обязательно ограничиваться ими):

– описание порядка установки программного продукта и его первичной настройки;

– описание модулей программы и их взаимосвязей;

– описание типовой структуры модуля и порядка его функционирования.

7.3. Интерактивная подсказка

1. Должна быть контекстно-зависимой и иметь развитую систему ссылок и функцию поиска по тексту.

2. Должна быть в состоянии предложить пользователю подсказки по заполнению полей экранных форм.

7.4. Руководства по инсталляции и конфигурированию и файл ReadMe

Должны быть представлены в электронной форме (формат файла – ASCII). Файл ReadMe должен содержать перечень изменений, проведенных в новой версии, а также список замеченных ошибок.

7.5. Маркировка и упаковка

В процессе инсталляции необходимо выводить информацию о версии программного продукта и о его авторе. При распространении продукта на CD-дисках предусмотреть наличие маркировки с информацией о наименовании и версии продукта, а также сведений об авторе.

8. Глоссарий и список сокращений

8.1. Глоссарий

8.2. Список сокращений

АБИС – автоматизированная библиотечная информационная система

ББК – библиотечно-библиографическая классификация

БД – база данных

ГПНТБ – Государственная публичная научно-техническая библиотека России

ГРНТИ – Государственный рубрикатор научно-технической информации

ОС – операционная система

ПНТ – пользовательское нефункциональное требование

ПФТ – пользовательское функциональное требование

СИБИД – Система стандартов по информации, библиотечному и издательскому делу

СУБД – система управления базами данных

УДК – универсальная десятичная классификация (система классификации информации, широко используется во всем мире для систематизации произведений науки, литературы и искусства, периодической печати, различных видов документов и организации картотек)

ЮНЕСКО (англ. UNESCO – United Nations Educational, Scientific and Cultural Organization) – Организация Объединённых Наций по вопросам образования, науки и культуры

CSS – Cascading Style Sheets (каскадные таблицы стилей)

CSV – Comma separated values (текстовый формат, предназначенный для представления табличных данных)

GNU – рекурсивный акроним от англ. GNU’s Not UNIX (проект, основанный Ричардом Столлманом – http://www.gnu.org)

GNU GPL – GNU General Public License (лицензия на свободное программное обеспечение, созданная в рамках проекта GNU)

I18n – internationalization (интернационализация)

ISBN – International Standard Book Number (международный стандартный номер книги – уникальный номер книжного издания, необходимый для распространения книги в торговых сетях и автоматизации работы с изданием)

L10n – localization (локализация)

Приложение 2. Архитектура и системные требования

Информационная система «Электронный архив»

Архитектура и системные требования (спецификация)

© Е. П. Моргунов

История исправлений и дополнений

Дата	Версия	Описание	Автор
01.05.2005	0.5	Исходная версия	Е. П. Моргунов
20.06.2009	1.0	Проведены реструктуризация и дополнение документа с учетом рекомендаций, предложенных в книгах И. Соммервилла и Д. Леффингуэлла (их библиографические описания см. ниже)	Е. П. Моргунов
01.05.2010	1.1	Добавлены новые модули; произведен отказ от фреймовой организации окна браузера	Е. П. Моргунов

Примечание. Дата выпуска версии 0.5 – приблизительная.

1. Введение

1.1. Цель документа

Цель данного документа состоит в описании высокоуровневой архитектуры программного продукта и распределении функций и требований по его подсистемам.

1.3. Ссылки и использованная литература

1.3.1. Список документов, упоминаемых в документе «Архитектура и системные требования»

1. Документ-концепция.

1.3.2. Список источников, к которым можно обратиться за справками в процессе разработки

1. Стандарты:

– ГОСТ 7.1–2003 «СИБИД. Библиографическая запись. Библиографическое описание. Общие требования и правила составления»;

– ГОСТ 7.11–2004 «СИБИД. Библиографическая запись. Сокращение слов и словосочетаний на иностранных европейских языках»;

– ГОСТ 7.12–93 «СИБИД. Библиографическая запись. Сокращение слов на русском языке. Общие требования и правила»;

– ГОСТ 7.60–90 «СИБИД. Издания. Основные виды. Термины и определения»;

– ГОСТ 7.80–2000 «СИБИД. Библиографическая запись. Заголовок. Общие требования и правила составления»;

– ГОСТ 7.82–2001 «СИБИД. Библиографическая запись. Библиографическое описание электронных ресурсов. Общие требования и правила составления»;

– ОСТ 29.130-97 «Издания. Термины и определения».

2. Система автоматизации библиотек ИРБИС. Общее описание системы [Текст]. – М. : ГПНТБ России, 2002. – 260 с.

1.3.3. Список использованной литературы

1. Соммервилл, И. Инженерия программного обеспечения [Текст] / Иан Соммервилл ; пер. с англ. А. А. Минько [и др.] ; под ред. А. А. Минько. – 6-е изд. – М. ; СПб. ; Киев : Вильямс, 2002. – 624 с. : ил. – Библиогр.: с. 603–617 (352, 35 назв.). – Предм. указ.: с. 618–623. – Парал. тит. англ. – Перевод изд.: Software engineering / Ian Sommerville. 6th ed. London [etc.] : Pearson Education, 2001. – 3500 экз. – ISBN 5-8459-0330-0 (рус.). – ISBN 0-201-39815-X (англ.).

2. Леффингуэлл, Д. Принципы работы с требованиями к программному обеспечению. Унифицированный подход [Текст] / Дин Леффингуэлл, Дон Уидриг ; пер. с англ. и ред. Н. А. Ореховой. – М. : Вильямс, 2002. – 446, [2] с. : ил. – Парал. тит. англ. – Прилож.: с. 369–438. – Библиогр.: с. 439–440. – Предм. указ.: с. 441–445. – Перевод изд.: Leffingwell, Dean. Managing Software Requirements. A Unified Approach / Dean Leffingwell, Don Widrig. Boston : Addison-Wesley, [2000]. – 3500 экз. – ISBN 5-8459-0275-4 (рус.). – ISBN 0-2016-1593-2 (англ.).

2. Архитектура программного продукта и модульный состав подсистем

2.1. Архитектура программного продукта

Архитектура программного продукта представлена на рис. 1. Основной программной подсистемой является *пользовательская подсистема*. Состав модулей и управляющие связи пользовательской подсистемы (т. е. иерархия вызовов модулей) показаны на рис. 2. В качестве модели управления выбрана модель централизованного управления (модель «вызов–возврат»). *Административная подсистема* предназначена для управления полномочиями доступа различных пользователей к базе данных из программных модулей. Эта подсистема опирается на группу так называемых *административных таблиц* базы данных. В этих таблицах хранятся метаданные о пользователях программного продукта. При запуске каждого модуля пользовательской подсистемы выполняется проверка полномочий доступа конкретного пользователя именно на основе этих таблиц. *Подсистема конфигурирования* предназначена для настройки программного продукта в соответствии с потребностями пользователя. Параметры конфигурации сохраняются в текстовых файлах.

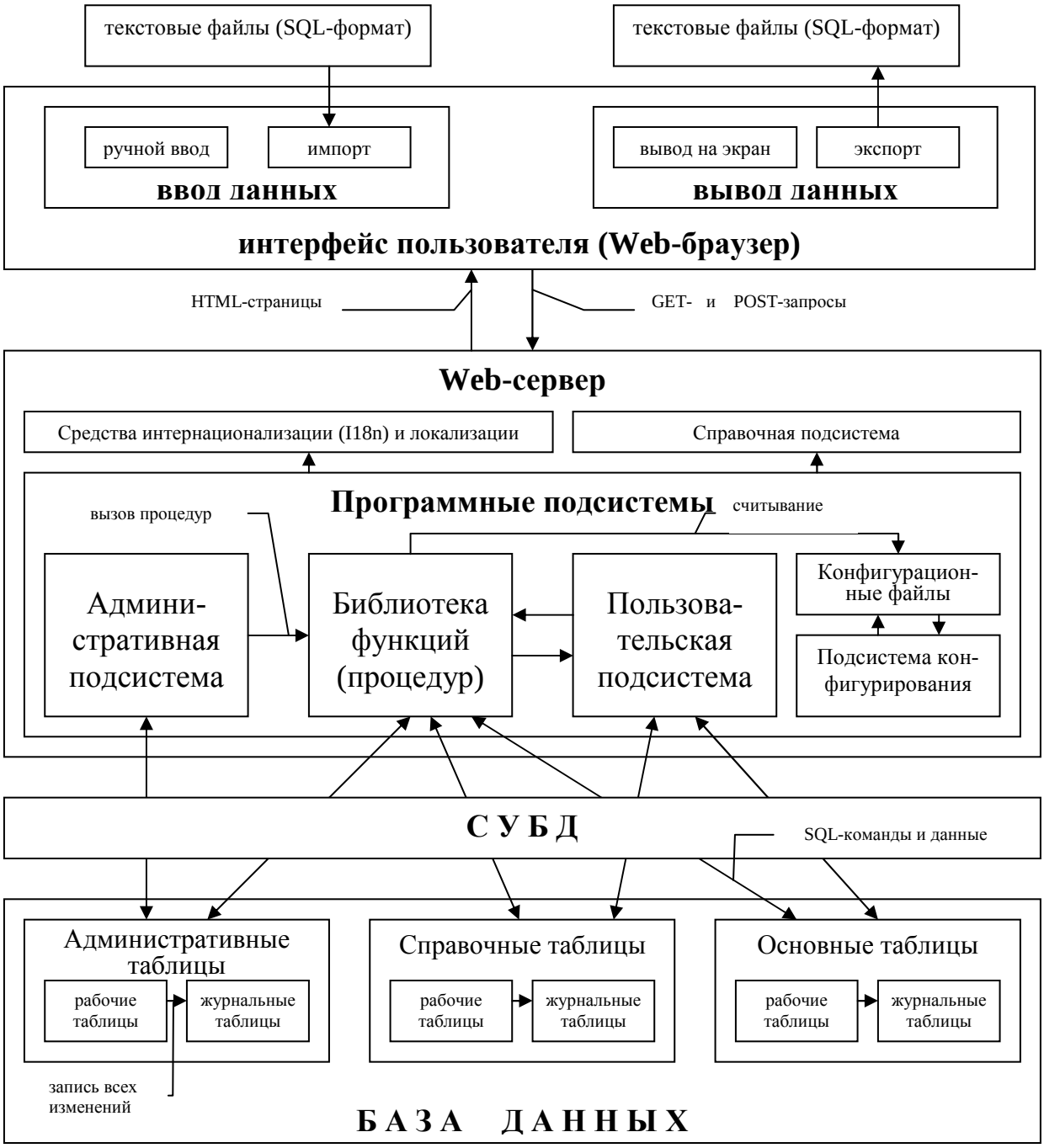

Рис. 1. Архитектура программного продукта

Важную роль играет *библиотека функций (процедур)*. Она содержит типовые процедуры для выполнения основных задач по работе с базой данных (ввод записи в таблицу, обновление и удаление записей), процедуры для формирования SQL-операторов на основе данных, введенных пользователем в поля формы в окне браузера, процедуру для считывания конфигурационных параметров, процедуры проверки полномочий пользователя и другие процедуры.

База данных условно разделена на три группы таблиц. Кроме административных таблиц есть также *справочные таблицы*, предназначенные для хранения справочных данных (персоналии, издательства и др.), и *основные таблицы*, предназначенные для хранения библиографических описаний и их элементов. Все три группы таблиц базы данных разделены на два класса: *рабочие таблицы* и *журнальные таблицы*. В рабочих

таблицах хранится текущая версия данных, с этими таблицами работает пользователь, он получает данные именно из этих таблиц. Журнальные таблицы предназначены для хранения истории изменений данных в рабочих таблицах. При выполнении операций обновления и удаления данных, содержащихся в рабочих таблицах, происходит также запись измененных версий этих данных в соответствующие журнальные таблицы. Журнальная таблица существует для каждой рабочей таблицы и содержит все ее поля, а также ряд дополнительных служебных полей.

Справочная подсистема должна быть контекстно-чувствительной. Должны быть предусмотрены *средства интернационализации (I18n) и локализации (L10n)*. Предполагается поддержка двух языков: русского и английского. Интерфейс пользователя реализуется с помощью Web-браузера.

Для облегчения и ускорения процессов ввода и вывода данных предназначены средства (процедуры) *импорта* и *экспорта* данных, представленных в виде SQL-операторов.

2.2. Модульный состав пользовательской подсистемы

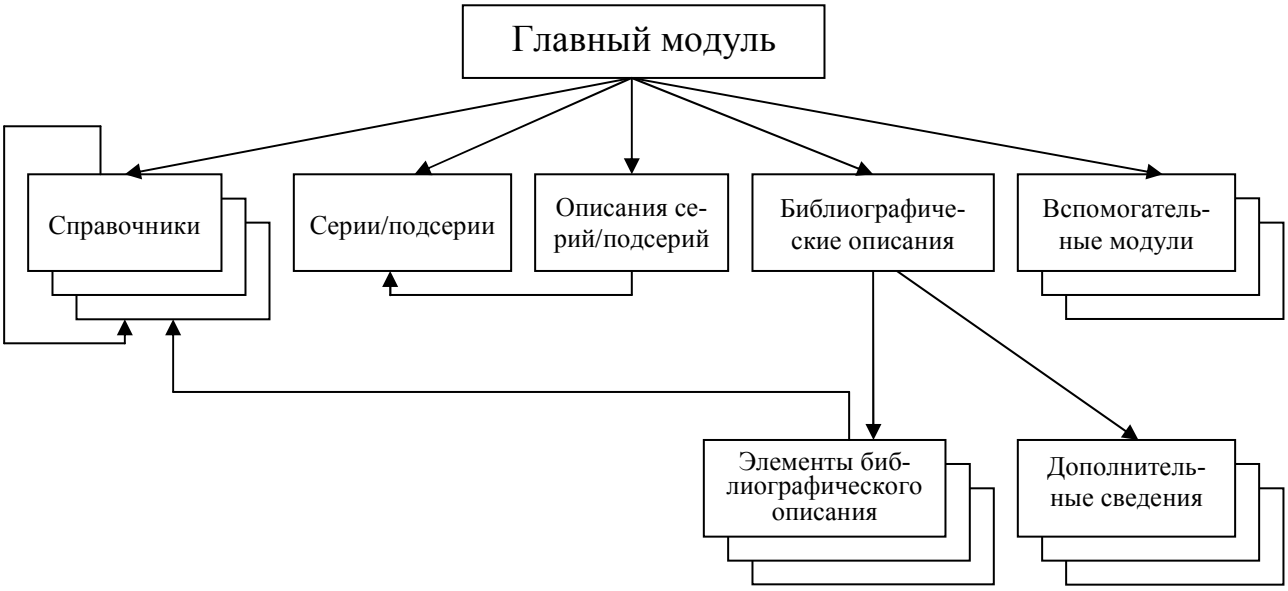

Рис. 2. Структура вызовов модулей пользовательской подсистемы

На рис. 2 стрелками указаны направления вызовов модулей (от вызывающего модуля к вызываемому). Справочники вызываются не только из главного модуля, но также и из модулей, отвечающих за обработку элементов библиографического описания. Кроме того, справочники могут вызываться один из другого.

В состав справочников на рис. 2 входят:

1. Валюты

Книги могут быть приобретены не только за рубли, но и за другие валюты.

2. Виды заглавий

Согласно ГОСТ 7.1–2003 возможны заглавия различных видов: основное, параллельное и др.

3. Виды мероприятий

Это относится к научным мероприятиям: конференция, симпозиум и др.

4. Виды признаков объектов библиографического описания

Согласно ОСТ 29.130–97: научное издание, учебное издание и др. Эта информация может быть полезной при формировании списков литературы по различным критериям: например, учебной литературы.

5. Виды справочных материалов

В состав справочного аппарата изданий могут входить различные указатели (предметный, именной и др.), примечания и т. д.

6. Виды стандартных номеров (ISBN, ISSN и др.)

Есть еще авторский знак, коды УДК и ББК.

7. Виды статуса мероприятий

Это относится к научным мероприятиям, например, конференциям: всероссийская, международная, региональная и т. д.

8. Виды тиражей

Основной тираж, допечатка тиража.

9. Виды ученых степеней

Кандидат технических наук, доктор технических наук и т. д.

10. Виды характеристик объектов библиографического описания

Пользователь может самостоятельно назначить эти характеристики, например, сложность, полезность и т. д.

11. Виды языковой обработки элементов библиографических описаний

В оригинальном виде, в переводе, в транскрипции, в транслитерации. Это может быть нужно при описании иностранных книг или статей, когда, например, нет возможности сделать описание на языке оригинала (например, японском или китайском).

12. Города

Места издания книг, проведения конференций, местонахождения организаций и др.

13. Группы признаков объектов библиографического описания

Согласно ОСТ 29.130–97: виды изданий по целевому назначению, виды изданий по читательскому адресу и др.

14. Группы специфических обозначений материала

Печатные материалы, аудиоматериалы и др. Позаимствовано у АБИС «ИРБИС».

15. Каналы поступления

Откуда поступил объект (книга, журнал, файл и т.д.): куплен в книжном магазине, «добыт» в сети Internet, подарен и т. д.

16. Коды видов деятельности организаций, учреждений

Образовательная, научная и т. д.

17. Коды видов упаковки

Переплет, бумажная обложка и др.

18. Коды источников библиографического описания

Где взята информация для составления библиографического описания: из оригинального объекта, из списка литературы в другой книге, из прайс-листа книжного магазина и др.

19. Коды общего обозначения материала

Согласно ГОСТ 7.1–2003: текст, электронный ресурс, карты и др.

20. Коды физической формы, в которой присутствуют единицы хранения

В оригинальном виде, ксерокопия и др. Это необходимо для выяснения доступности, например, книг для чтения.

21. Научные специальности

Согласно номенклатуре Высшей аттестационной комиссии Министерства образования и науки России. Например, 05.13.01 «Системный анализ, управление и обработка информации».

22. Отрасли науки

Технические науки, экономические науки и т. д.

23. Роли

Роль (функция), в которой выступает персоналия или организация в данном издании, например, автор, редактор, переводчик и т. д.

24. Специфические обозначения материала

Страница (с.), лист (л.), электронно-оптический диск (электрон.-опт. диск) и др.

25. Страны

В базу данных могут попасть сведения о книгах, изданных за рубежом.

26. Типы связей между объектами библиографического описания

Том многотомного издания, перевод и т. д. Это может быть полезно при выявлении, например, оригинального издания и его переводов.

27. Языки

Русский, английский и др.

*28. Виды деятельности организаций, учреждений

Виды деятельности конкретных организаций, учреждений. Например, «Наука» – издательская, Московский государственный университет – образовательная и научная.

*29. Ключевые слова

Ключевые слова (ими также могут быть и целые словосочетания), назначенные книгам, статьям и другим объектам в качестве характерных признаков, отражающих их содержание, тематику излагаемого в этих источниках материала.

*30. Конференции, съезды, симпозиумы, семинары, выставки ...

Описания конкретных конференций, съездов, симпозиумов, семинаров, выставок.

*31. Организации, учреждения, временные коллективы

Описания конкретных организаций, учреждений, временных коллективов: издательств, библиотек, вузов и др.

*32. Персоналии

Это все те люди, которые имеют отношение к объектам, описанным в базе данных: авторы, редакторы, владельцы интересующих нас книг и т. д.

Примечание. Знаком * обозначены важные справочники.

Наименования областей библиографического описания и элементов этих областей определены в ГОСТ 7.1–2003. В состав модулей, отвечающих за обработку элементов библиографического описания, на рис. 2 входят (наименования областей библиографического описания выделены курсивом, они не являются наименованиями модулей):

Область заглавия и сведений об ответственности

1. Заглавия

Издание может иметь более одного заглавия (например, параллельное заглавие).

2. Сведения об ответственности

Список всех лиц и/или организаций, принимавших участие в выпуске данного издания (авторы, редакторы, переводчики и др.).

Область издания

3. Сведения об издании

Например, издание 2-е, исправленное и дополненное (изд. 2-е, испр. и доп.).

Область специфических сведений

4. Электронные ресурсы

5. Сериальные документы

6. Нормативные документы (стандарты и ТУ)

Область выходных данных

7. Место издания, изготовления

Возможно более одного места издания для данного издания.

8. Издатель, изготовитель, распространитель

Возможно более одного издателя для данного издания.

9. Дата издания, печатания, изготовления

Область физической характеристики

10. Специфическое обозначение материала и объем; сопроводительный материал

Например, 254 с. : ил. (иллюстрации).

Область серии

11. Серии

Наименование серии или серий, в которые входит данное издание.

Область примечания

12. Примечания

В качестве примечаний выступают сведения о библиографических списках, о различных указателях (именном, предметном и др.), сведения об оригинальном иностранном издании (для переводных изданий) и т. д.

13. Тираж

Область стандартного номера и условий доступности

14. Стандартные номера (ISBN, ISSN, ...)

Для конкретного издания их может быть более одного, например, для оригинального иностранного издания и для русскоязычного переводного издания, для многотомного издания в целом и для отдельного тома.

15. Условия доступности

16. Цена

В состав модулей, отвечающих за обработку дополнительных сведений, относящихся к объектам библиографического описания, на рис. 2 входят:

Дополнительные библиографические сведения

17. Объекты-контейнеры

Этот модуль отвечает за формирование связи между аналитическим библиографическим описанием, например, научной статьи и библиографическим описанием конкретного выпуска (номера) научного журнала, в котором эта статья помещена. Еще одним примером является установление связи между библиографическим описанием многотомного издания в целом и библиографическими описаниями отдельных томов. Таким образом, объект-контейнер в приведенных примерах – это конкретный выпуск (номер) научного журнала и многотомное издание в целом. Наличие этого модуля должно упростить процедуру получения сведений, например, обо всех томах многотомного издания.

18. Языки текста

Текст издания может быть представлен на нескольких языках.

19. Признаки объекта

Признаки конкретного объекта (книги, журнала, электронного издания и др.) определяются на основе справочника «Виды признаков объектов библиографического описания».

Использование

20. Проработка содержания

Описание вида проработки (чтение текста, просмотр и т.д.) и даты проработки книги, статьи и т. д.

21. Общая характеристика

Общая характеристика проработанного издания, сведения о полезных фрагментах текста, заинтересовавших читателя (т. е. номера страниц), короткие цитаты.

22. Детальные характеристики

Виды таких характеристик выбирает сам пользователь из справочника «Виды характеристик объектов библиографического описания», оценки выставляются по 10-балльной шкале.

23. Ключевые слова

Ключевые слова, назначенные конкретному объекту. Они выбираются из справочника «Ключевые слова».

24. Содержание (оглавление)

Содержание (оглавление) может быть введено частично или полностью.

25. Справочные материалы

См. описание справочника «Виды справочных материалов»

26. Участие в группировках (списках)

Конкретный объект (книга, статья) может быть включен в различные тематические списки, например, список книг для подготовки курсовой работы по какой-либо дисциплине. В этом модуле мож-

но просмотреть описания тематических списков, в которые включен данный объект, и включить объект в другие списки.

Наличие

27. Экземпляры (единицы хранения)

Могут быть описаны не только те экземпляры, которые хранятся у пользователя программного продукта, но также и экземпляры, хранящиеся в библиотеках.

В состав вспомогательных модулей на рис. 2 входят:

1. Описания временных группировок (списков) объектов библиографического описания

Это и есть тематические списки: описания и полные составы, т. е. перечни объектов, входящих в каждый список.

3. Системные требования

3.1. Требования к программному продукту в целом

Пользовательские требования (см. документ-концепцию):

– ПФТ1. Соответствие требованиям ГОСТ 7.1–2003;

– ПНТ3. Аутентификация пользователей;

– ПНТ6. Организация коллективной работы.

При проектировании программного продукта необходимо учесть все пользовательские требования, приведенные в этом списке, даже если они не детализированы в системных требованиях.

СТ1. Организация доступа пользователя к идентификаторам объектов

1. Интерфейс пользователя и база данных должны быть организованы таким образом, чтобы пользователь имел возможность работать непосредственно с идентификаторами объектов, хранящихся в базе данных.

Обоснование. Это позволит в ряде случаев ускорить процесс ввода и корректировки данных за счет того, что в некоторых областях библиографического описания наиболее часто используется ограниченное множество объектов (это имеет место, например, для издательств), поэтому пользователь может запомнить идентификаторы этих объектов. Данный подход также позволит сделать более гибким процесс конфигурирования программного продукта за счет того, что в конфигурационном файле пользователь сможет указывать непосредственно идентификаторы, а не наименования объектов, тем самым будет устранена возможность описок.

3.2. Подсистема «Интерфейс пользователя»

Пользовательские требования (см. документ-концепцию):

– ПФТ7. Простые средства поиска информации;

– ПФТ10. Развитые средства помощи пользователю для ускорения ввода данных;

– ПФТ11. Развитые средства поиска информации.

При проектировании интерфейса пользователя необходимо учесть все пользовательские требования, приведенные в этом списке, даже если они не детализированы в системных требованиях.

Данная подсистема не состоит из конкретных модулей. Она является, условно говоря, виртуальной, поскольку весь интерфейс формируется динамически модулями программной пользовательской подсистемы, которая отвечает также и за взаимодействие с базой данных. Модулей, которые предназначены только для формирования интерфейса пользователя, не предусмотрено.

СТ2. Общая схема организации интерфейса пользователя

1. Необходимо отказаться от использования фреймовой структуры организации окна браузера. В верхней части окна должна содержаться информация о командах, доступных пользователю. Нижняя часть окнам предназначена для ввода и вывода данных, т. е. для визуализации данных, извлеченных по запросу пользователя из базы данных, либо для организации HTML-формы для запроса новых данных у пользователя.

Обоснование. В версии HTML 5 фреймы не поддерживаются.

СТ3. Гибкая схема работы пользователя при вводе данных

1. Необходимо предусмотреть возможность вызова модулей-справочников при вводе данных в основные таблицы базы данных. При этом пользователь должен иметь возможность ввести требуемые данные в окно HTML-формы посредством осуществления выбора из списка, сформированного модулем-справочником.

2. Необходимо предоставлять пользователю также и возможность работы непосредственно с идентификаторами объектов, представленных в базе данных (например, вводить непосредственно идентификатор издательства или идентификатор автора).

3. Необходимо предусмотреть возможность использования значений «по умолчанию». Такие значения пользователь может ввести в конфигурационный файл.

4. Необходимо для большинства полей предусмотреть возможность формирования коротких списков, состоящих из наиболее часто используемых значений, вводимых пользователем в это поле. Например, для поля с условным названием «Фамилия, имя, отчество персоналии» один список может содержать наиболее распространенные имена, а второй список – отчества. Особенностью этих списков является то, что они не хранятся в справочных таблицах базы данных, а формируются на основе конфигурационного файла (в который их вносит пользователь). Такие списки вызываются на экран с помощью HTML-ссылки. При работе с описанными списками пользователь просто выбирает нужное значение из списка, и оно копируется в то поле HTML-формы, для которого этот список предназначен.

Обоснование. Способ ввода непосредственно идентификаторов позволит сократить число манипуляций пользователя с «мышью» и клавиатурой при частом вводе ограниченного множества идентификаторов (например, если пользователь часто работает с книгами двух-трех издательств, то он может просто запомнить идентификаторы этих издательств).

СТ4. Индикация наличия данных в областях библиографического описания

1. Необходимо при выводе HTML-формы с библиографическим описанием выводить также знаки (признаки), показывающие наличие записей в таблицах базы данных, соответствующих областям библиографического описания (сведения об ответственности, место издания и т. д.).

2. Должно быть реализовано два подхода:

– «бинарный», т. е. такой, при котором пользователь видит только сам факт наличия таких записей, но не видит их количества, поэтому для получения сведений об их количестве пользователь должен запустить соответствующий модуль, ответственный за конкретную область библиографического описания;

– «количественный», при котором пользователь видит количество записей в каждой области библиографического описания.

Обоснование. Это позволит упростить получение информации о степени полноты ввода данных в области библиографического описания. Без реализации этого требования пользователю пришлось бы для получения информации о наличии в базе данных, например, сведений об ответственности или месте издания вызывать модули, ответственные за те или иные области библиографического описания, т. е. производить больше манипуляций с «мышью» и клавиатурой.

СТ5. Подтверждение деструктивных действий пользователя

1. При попытке пользователя выполнить одно из деструктивных действий, например, удаление записи из базы данных, должен быть выведен запрос на подтверждение этих действий.

Обоснование.

СТ6. Контекстно-зависимый способ активации/деактивации полей в HTML-формах

1. При вводе данных в поля HTML-форм необходимо запрещать (деактивировать поле) или разрешать (активировать поле) доступ к тем или иным полям в зависимости от значений данных, введенных пользователем в другие поля.

Обоснование.

3.3. Подсистема «Средства интернационализации и локализации»

См. документ-концепцию (п. 6.2.2.4).

3.4. Справочная подсистема

См. документ-концепцию (п. 7.3).

3.5. Административная подсистема

Пользовательские требования (см. документ-концепцию):

– ПНТ4. Интерактивная система управления разграничением полномочий доступа к базе данных.

При проектировании административной подсистемы необходимо учесть все пользовательские требования, приведенные в этом списке, даже если они не детализированы в системных требованиях.

СТ7. Предоставление полномочий пользователю на уровне отдельных процедур модулей

1. Полномочия, предоставляемые конкретному пользователю, должны связываться с конкретными процедурами конкретных модулей.

Пример. Пользователю admin может быть предоставлено право выполнения процедуры корректировки библиографического описания, а пользователю test_user такое право может не быть предоставлено.

2. Необходимо использовать специальные приемы для сокращения числа записей в административных таблицах базы данных с целью упрощения как алгоритмов предоставления полномочий пользователю, так и алгоритмов обработки этих записей при выяснении объема полномочий пользователя.

Пример. Если необходимо предоставить пользователю право выполнять *все* процедуры во *всех* модулях, то это должно быть сделано путем ввода в специальную административную таблицу базы данных лишь *одной* записи с указанием специальных ключевых слов (например, «ALL») как для наименований модулей, так и для наименований процедур.

3. При попытке пользователя выполнить процедуру, на выполнение которой у него нет полномочий, должно быть выведено сообщение.

Обоснование. Предлагаемый подход будет служить дополнением к системе привилегий доступа к таблицам базы данных, которая (система) организуется средствами СУБД. Кроме того, это позволит избежать вывода системных сообщений, которые генерирует СУБД в случаях, когда у пользователя недостаточно полномочий доступа к таблицам. Вместо этого пользователь получит понятное ему сообщение о том, что у него нет полномочий для выполнения данной процедуры.

3.6. Библиотека функций (процедур)

Нет особых требований.

3.7. Пользовательская подсистема

Пользовательские требования (см. документ-концепцию):

– ПФТ3. Обработка наличия различных вариантов имен собственных;

– ПФТ7. Простые средства поиска информации;

– ПФТ10. Развитые средства помощи пользователю для ускорения ввода данных;

– ПФТ11. Развитые средства поиска информации;

– ПФТ12. Иерархическая система ключевых слов;

– ПФТ13. Формирование библиографического описания из элементарных данных.

При проектировании пользовательской подсистемы необходимо учесть все пользовательские требования, приведенные в этом списке, даже если они не детализированы в системных требованиях.

СТ8. Проверка полномочий пользователя перед выполнением каждой операции

1. Должна быть организована проверка полномочий пользователя перед выполнением каждой операции. При отсутствии таких полномочий следует выводить сообщение.

Обоснование.

СТ9. Вывод сообщений об ошибках

1. При выполнении операций с базой данных должны выводиться сообщения об ошибках при их возникновении.

Обоснование.

3.8. Подсистема конфигурирования

Пользовательские требования (см. документ-концепцию):

– ПНТ2. Конфигурирование программного продукта.

При проектировании подсистемы конфигурирования необходимо учесть все пользовательские требования, приведенные в этом списке, даже если они не детализированы в системных требованиях.

3.9. База данных

Пользовательские требования (см. документ-концепцию):

– ПФТ1. Соответствие требованиям ГОСТ 7.1–2003;

– ПФТ2. Ввод библиографических описаний в базу данных в готовом виде;

– ПФТ3. Обработка наличия различных вариантов имен собственных;

– ПФТ4. Фиксирование результатов проработки источников;

– ПФТ5. Хранение выписок из проработанных источников;

– ПФТ6. Система ключевых слов;

– ПФТ8. Учет экземпляров;

– ПФТ9. Формирование тематических библиографических списков;

– ПФТ12. Иерархическая система ключевых слов;

– ПНТ3. Аутентификация пользователей;

– ПНТ4. Интерактивная система управления разграничением полномочий доступа к базе данных;

– ПНТ5. Ведение журнала операций с базой данных;

– ПНТ6. Организация коллективной работы.

При проектировании базы данных необходимо учесть все пользовательские требования, приведенные в этом списке, даже если они не детализированы в системных требованиях.

СТ10. Состав таблиц базы данных

1. В базе данных должны быть предусмотрены таблицы для хранения следующих сведений:

– всех справочников согласно перечню модулей-справочников, приведенному в разделе 2.2 настоящего документа;

– сформированных библиографических описаний;

– результатов проработки источников;

– выписок из проработанных источников;

– иерархической системы ключевых слов;

– данных о сериях и подсериях;

– данных об экземплярах описанных объектов;

– тематических библиографических списков.

Обоснование.

СТ11. Разделение базы данных на группы таблиц

1. База данных должна быть условно разделена на следующие группы таблиц: административные таблицы, справочные таблицы и основные таблицы.

Примечание. К основным таблицам относятся те таблицы, которые не являются справочными или административными.

Обоснование. Это позволит лучше структурировать программные подсистемы.

СТ12. Поддержка иерархий объектов в базе данных

1. Должна быть предусмотрена возможность создания иерархий объектов в таблицах базы данных за счет введения в таблицы полей с условным названием «Идентификатор родительского объекта».

Обоснование. Такие иерархии могут возникать, например, при описании организаций (Российская академия наук имеет в своем подчинении ряд академических институтов; у издательств могут быть филиалы и т. д.).

СТ13. Использование согласованных правил именования полей и таблиц базы данных

1. Должна использоваться единая система именования полей таблиц и самих таблиц базы данных. Однотипные поля должны иметь префиксы и/или постфиксы (например, постфикс «_fname», т. е. full name, может означать полное имя объектов какого-либо типа, а постфикс «_sname», т. е. short name, – краткое имя).

Обоснование.

СТ14. Разграничение полномочий доступа пользователей к базе данных

1. В каждой таблице базы данных должны быть предусмотрены поля для хранения имени пользователя, выполнившего операцию с конкретной таблицей, и времени выполнения этой операции.

Обоснование.

3.10. Средства импорта и экспорта данных

Пользовательские требования (см. документ-концепцию):
– ПНТ1. Экспорт и импорт данных.

При проектировании базы данных необходимо учесть все пользовательские требования, приведенные в этом списке, даже если они не детализированы в системных требованиях.

4. Интерфейсы между подсистемами

4.1. Программные подсистемы ↔ база данных

1. Взаимодействие между программными подсистемами и базой данных организуется с помощью пакета Pg.pm, написанного на языке программирования Perl. Ранее

этот пакет поставлялся в составе дистрибутива СУБД PostgreSQL (до версии 7.3). В программных модулях SQL-запросы к базе данных формируются в виде символьных строк, которые передаются системе управления базой данных (СУБД) с помощью специальных процедур, входящих в состав пакета Pg.pm.

4.2. Пользовательская подсистема ↔ административная подсистема

1. Взаимодействие между этими подсистемами является косвенным и организуется посредством вызовов процедур из библиотеки процедур, которая взаимодействует с административными таблицами базы данных.

4.3. Пользовательская подсистема ↔ подсистема конфигурирования

1. Взаимодействие между этими подсистемами является косвенным и организуется за счет обращения к библиотеке процедур, в состав которой входят процедуры, предназначенные для считывания конфигурационных файлов.

5. Глоссарий и список сокращений

5.1. Глоссарий

5.2. Список сокращений

АБИС – автоматизированная библиотечная информационная система
БД – база данных
ПНТ – пользовательское нефункциональное требование
ПФТ – пользовательское функциональное требование
СТ – системное требование
СУБД – система управления базами данных

I18n – internationalization (интернационализация)
L10n – localization (локализация)

Приложение 3. Описание шага разработки при использовании пошаговой модели разработки

Информационная система «Электронный архив»

Шаг разработки 4

© Е. П. Моргунов

1. Основные задачи

1. Разработать и реализовать механизм вызова подсказок для ввода типовых фрагментов данных в поля.

2. Реализовать простейшие механизмы поиска (по авторам, по ключевым словам).

3. Протестировать реализованные механизмы.

Предполагаемый срок завершения шага 4 – 10 ноября 2009 г.

2. Реализуемые и учитываемые требования

Реализуемые требования – это такие требования, которые на данном шаге разработки выполняются целиком и полностью, не требуют каких-либо дополнительных доработок на последующих шагах, т. е. могут считаться завершенной частью работы.

Учитываемые требования – это такие требования, которые на данном шаге разработки принимаются к сведению, и разработка проводится с учетом их положений и рекомендаций, однако при этом такое требование не является полностью выполненным (исчерпанным).

2.1. Реализуемые требования

Пользовательские требования (см. документ-концепцию):

– ПФТ3. Обработка наличия различных вариантов имен собственных;

– ПФТ6. Система ключевых слов (п. 2);

– ПФТ7. Простые средства поиска информации;

– ПФТ10. Развитые средства помощи пользователю для ускорения ввода данных;

– ПНТ2. Конфигурирование программного продукта (пп. 1, 2, 3).

Системные требования (см. архитектуру и системные требования):

– СТ3. Гибкая схема работы пользователя при вводе данных (п. 4);

– СТ4. Индикация наличия данных в областях библиографического описания;

– СТ6. Контекстно-зависимый способ активации/деактивации полей в HTML-формах.

2.2. Учитываемые требования

Учитываемых требований нет.

3. Проектирование

Приведены только основные алгоритмы и технические решения, принятые на шаге разработки 4.

1. Алгоритм выборки всех объектов, в области сведений об ответственности которых фигурирует конкретная персоналия

Алгоритм должен быть реализован в модуле «Персоналии» (persons.pl).

В таблице «Персоналии» (Persons) есть следующие поля:

– pers_id – уникальный идентификатор персоналии;

– pers_name – полное имя, отчество, фамилия;

– pers_id_orig – код персоналии, являющейся началом цепочки (для учета разночтений, написаний на различных языках).

ВЫБРАТЬ ЗАПИСИ из таблицы Persons для четырех типов персоналий:

– персоналия, которую выбрал пользователь из списка персоналий;
– персоналия, которая является для выбранной пользователем персоналии началом цепочки (на основе поля pers_id_orig);
– персоналии, у которых выбранная пользователем персоналия является началом цепочки (на основе поля pers_id_orig);
– персоналии, у которых началом цепочки является та же персоналия, что и у той, которую выбрал пользователь

СФОРМИРОВАТЬ массив идентификаторов персоналий на основе полученной выборки

ВЫБРАТЬ ЗАПИСИ из таблиц roles и resp_stat
на основе массива идентификаторов персоналий выбираем наименования ролей, которые «играла» указанная персоналия во всех объектах библиографического описания

СФОРМИРОВАТЬ заголовок списка библиографических описаний на основе полученной выборки наименований ролей

ВЫБРАТЬ ЗАПИСИ из таблиц bib_recs b и resp_stat
выбираем записи с библиографическими описаниями объектов, в судьбе которых принимала участие персоналия, выбранная пользователем (используем сформированный массив идентификаторов персоналий)

ВЫВЕСТИ заголовок списка библиографических описаний

ВЫВЕСТИ список библиографических описаний

2. Алгоритм выборки всех объектов, которым назначено выбранное пользователем ключевое слово

Алгоритм должен быть реализован в модуле «Ключевые слова» (keywords.pl).

В таблице «Ключевые слова» (Keywords) есть следующие поля:
– keyword_code – код ключевого слова;
– keyword_name –ключевое слово (выражение);
– keyword_code_orig – код ключевого слова, являющегося началом цепочки (для учета переименований, разночтений, сокращений, написаний на различных языках)

ВЫБРАТЬ ЗАПИСИ из таблицы Keywords для четырех типов ключевых слов:
– то слово, которое выбрал в списке пользователь;
– то слово, которое является для выбранного пользователем слова началом цепочки (на основе поля keyword_code_orig);
– слова, у которых выбранное пользователем слово является началом цепочки (на основе поля keyword_code_orig);
– слова, у которых началом цепочки является то же слово, что и у того слова, которое выбрал пользователь

СФОРМИРОВАТЬ массив идентификаторов ключевых слов и заголовок списка на основе полученной выборки

ВЫБРАТЬ ЗАПИСИ из таблиц obj_keywords и bib_recs
выбираем записи с библиографическими описаниями объектов, которым назначено одно из ключевых слов, идентификаторы которых содержатся в сформированном массиве идентификаторов ключевых слов

ВЫВЕСТИ заголовок списка библиографических описаний

ВЫВЕСТИ список библиографических описаний

3. Система помощи пользователю для ускорения ввода данных

Эта система должна предоставлять возможность вводить типовые фрагменты текста в поля HTML-формы путем нажатия на HTML-кнопку или с помощью выбора ссылки. Для работы этой системы, которая условно называется системой «быстрой» вставки фрагментов текста в поля HTML-формы, в конфигурационном файле должны быть предусмотрены специальные параметры. Структура наименований этих параметров такова:
– код модуля (например, BIB_RECS);
– имя поля, для которого предназначены параметры (например, BIB_REC_HEAD);

– специальные постфиксы, которые начинаются с символов _HELP_.

Параметры образуют пары: первый элемент такой пары имеет окончание _TITLE (это текст ссылки или надпись на кнопке), второй элемент – _VALUE (это фрагмент текста для вставки в поле). Таких пар для каждого поля может быть несколько; в HTML-форме ссылки (кнопки) следуют в том же порядке, в котором их описания расположены в конфигурационном файле. Элементов _VALUE для каждого элемента _TITLE может быть более одного. В этом случае при выборе ссылки (нажатии на кнопку) открывается новое окно в браузере, и все параметры _VALUE, относящиеся к этому параметру _TITLE, выводятся в этом окне в виде HTML-ссылок. Принципиальное отличие единичной ссылки (кнопки) от группы ссылок в окне состоит в том, что в первом случае имя ссылки (или надпись на кнопке) может отличаться от того текста, который будет вставлен в поле HTML-формы. Во втором же случае имя ссылки, выведенной в списке ссылок в новом окне браузера, совпадает с текстом, который будет вставлен в поле HTML-формы.

Для формирования таких «подсказок» могут использоваться как кнопки, так и ссылки. В первом случае имя параметра-заголовка будет оканчиваться на …_HELP_BTN_TITLE (BTN – button), во втором случае – …HELP_LINK_TITLE. Окончание имен параметров, представляющих собой фрагмент вставляемого текста, в обоих случаях одинаковое: …_HELP_VALUE.

Примеры параметров ссылок (кнопок) для быстрой вставки фрагментов текста в поле BIB_REC_FULL модуля BIB_RECS на экране. Для одного поля HTML-формы может быть предусмотрено несколько кнопок или ссылок.

BIB_RECS_BIB_REC_FULL_HELP_LINK_TITLE	[Текст]
BIB_RECS_BIB_REC_FULL_HELP_VALUE	[Текст]
BIB_RECS_BIB_REC_FULL_HELP_LINK_TITLE	Библ.
BIB_RECS_BIB_REC_FULL_HELP_VALUE	Библиогр.: с.
BIB_RECS_BIB_REC_FULL_HELP_BTN_TITLE	М., Е. П.
BIB_RECS_BIB_REC_FULL_HELP_VALUE	Моргунов, Е. П.
BIB_RECS_BIB_REC_FULL_HELP_BTN_TITLE	Е. П. М.
BIB_RECS_BIB_REC_FULL_HELP_VALUE	Е. П. Моргунов
BIB_RECS_BIB_REC_FULL_HELP_BTN_TITLE	Журналы
BIB_RECS_BIB_REC_FULL_HELP_VALUE	Вестник Сиб. гос. аэрокосмич. ун-та
BIB_RECS_BIB_REC_FULL_HELP_VALUE	Информационные технологии
BIB_RECS_BIB_REC_FULL_HELP_VALUE	Системы управления

Алгоритм обработки конфигурационного файла (модифицированный в связи с созданием системы помощи пользователю при вводе данных)

Символом # обозначены комментарии.

ПЕРЕМЕННЫЕ

порядковый номер ссылки (кнопки) для вызова «быстрой» вставки текста (нумерация ведется в пределах конкретного поля HTML-формы)
hot_help_index

порядковый номер содержательной части, той, которая вставляется в поле формы, когда пользователь мышью выбирает эту ссылку (или кнопку) (нумерация ведется в пределах одного «комплекта» параметров для «быстрой» вставки текста)
help_value_index

начальная часть имени параметра, предназначенного для организации помощи пользователю при вводе текста (состоит из имени программного модуля и имени поля)
module_field запомненное значение
current_module_field значение в текущей строке конфигурационного файла
ИНИЦИАЛИЗИРОВАТЬ module_field значением «пустая строка»
ИНИЦИАЛИЗИРОВАТЬ current_module_field значением «пустая строка»

ОТКРЫТЬ конфигурационный файл

ЦИКЛ ПОКА не считана последняя строка конфигурационного файла
    РАЗДЕЛИТЬ текущую строку файла на две части: имя параметра и его значение

    ЕСЛИ имя параметра оканчивается на HELP_LINK_TITLE или
                                    HELP_BTN_TITLE
        ВЗЯТЬ начальную часть имени параметра
        ЗАПИСАТЬ ее в переменную current_module_field
        ЕСЛИ current_module_field = module_field, т. е. параметр относится
            к тому же полю HTML-формы
            УВЕЛИЧИТЬ счетчик hot_help_index на 1
        ИНАЧЕ   # теперь текущим стало новое поле
            ПРИСВОИТЬ module_field = current_module_field
            ПРИСВОИТЬ счетчику hot_help_index значение 1
        КОНЕЦ ЕСЛИ
        # как только встретился параметр …HELP_(LINK|BTN)_TITLE, мы
        # должны начинать отсчет параметров …HELP_VALUE с начала,
        # поскольку нумерация самих фрагментов для «быстрой» вставки
        #производится в пределах поля
        ПРИСВОИТЬ help_value_index значение 1
        ДОБАВИТЬ к имени параметра порядковый номер hot_help_index,
                который изменяется в пределах поля
    ИНАЧЕ ЕСЛИ имя параметра оканчивается на HELP_VALUE
        ДОБАВИТЬ к имени параметра порядковый номер hot_help_index,
                который изменяется в пределах поля, символ
                подчеркивания «_» и порядковый номер фрагмента текста
                help_value_index
        УВЕЛИЧИТЬ счетчик help_value_index на 1
    КОНЕЦ ЕСЛИ
    ЗАПИСАТЬ имя и значение параметра в массив конфигурационных параметров
КОНЕЦ ЦИКЛА

ЗАКРЫТЬ конфигурационный файл

ВОЗВРАТИТЬ массив конфигурационных параметров

4. Реализация

Для реализации механизма формирования подсказок для ввода типовых фрагментов данных в поля использовался язык JavaScript.

Для непосредственной вставки текста в поле HTML-формы использовался программный код на языке JavaScript, размещенный в открытом доступе на сайтах:

http://parentnode.org/javascript/working-with-the-cursor-position/;
http://pastebin.parentnode.org/78;
http://www.htmlcenter.com/blog/insert-text-into-textareas-using-javascript/.

5. Тестирование

Методы тестирования были очень простыми (примитивными) по следующей причине: разработчиком и пользователем программного продукта является одно и то же лицо. Более тщательное тестирование отложено на последующие шаги разработки.

Стратегия черного ящика.

1. Тестирование всех вновь введенных параметров конфигурационного файла на основе метода эквивалентного разбиения (присваивание параметрам конкретных значений, а также использование конфигурационного файла с исключенными параметрами).

Выявлен следующий дефект: механизм «быстрой» вставки фрагментов текста в поля HTML-формы не позволяет ввести символ «’».

Тестирование показало пригодность программного продукта для решения поставленных задач.

6. Эксплуатационная документация

6.1. Руководство пользователя

Не оформлялось.

6.2. Руководство программиста

Не оформлялось.

6.3. Руководства по инсталляции и конфигурированию

Краткое руководство по инсталляции оформлено в виде текстового файла INSTALL, размещенного в подкаталоге src/install дистрибутивного комплекта файлов.

7. Версия программного продукта

Принято решение присвоить этой версии программного продукта номер 0.4.
Сформирован дистрибутивный комплект в виде файла e_arch-0.4.tgz.

Перечень еще не реализованных требований:

Пользовательские требования (см. документ-концепцию):

– ПФТ5. Хранение выписок из проработанных источников (п. 2);

– ПФТ11. Развитые средства поиска информации;

– ПФТ12. Иерархическая система ключевых слов;

– ПФТ13. Формирование библиографического описания из элементарных данных;

– ПНТ1. Экспорт и импорт данных;

– ПНТ2. Конфигурирование программного продукта (п. 4);

– ПНТ4. Интерактивная система управления разграничением полномочий доступа к базе данных;

– ПНТ5. Ведение журнала операций с базой данных (п. 2);

– ПНТ6. Организация коллективной работы (п. 2);

– 6.1.1. Требования к инсталляции

Необходимо разработать процедуру инсталляции программного продукта.

– *6.1.2.1.4. Следование соглашениям и стандартам, разработанным для человеко-машинного интерфейса*

…использование каскадных таблиц стилей (CSS).

– 6.1.5.1. Поддерживаемые операционные системы

В будущем предполагается перенос продукта в среду операционной системы Windows (XP/2003/Vista/7).

– 6.2.2.3. Языки программирования и инструментальные средства

2. Рассмотреть также возможность использования системы управления версиями программного продукта (Subversion, Git и др.).

3. Рассмотреть возможность использования общего интерфейса DBI/DBD к СУБД PostgreSQL.

– 6.2.2.4. Локализация и интернационализация

Необходимо реализовать поддержку локализации (l10n) и интернационализации (i18n) на более поздних этапах разработки. Поддерживаемые языки: русский и английский.

– 6.2.3. Требования к стандартам

1. Использовать стандарты кодирования и оформления исходных текстов программ, аналогичные стандартам проекта GNU.

– 6.2.4. Требования к лицензированию

Программный продукт должен распространяться по лицензии, аналогичной лицензии GNU GPL.

– 7.1. Руководство пользователя;

– 7.2. Руководство программиста;

– 7.3. Интерактивная подсказка;

– 7.4. Руководства по инсталляции и конфигурированию и файл ReadMe.

Системные требования (см. архитектуру и системные требования):

– СТ5. Подтверждение деструктивных действий пользователя.

8. Глоссарий и список сокращений

8.1. Глоссарий

8.2. Список сокращений

БД – база данных
ПНТ – пользовательское нефункциональное требование
ПФТ – пользовательское функциональное требование
СТ – системное требование
СУБД – система управления базами данных

Приложение 4. Стандарт кодирования

Информационная система «Электронный архив»

Стандарт кодирования

© Е. П. Моргунов

История исправлений и дополнений

Дата	Версия	Описание	Автор
10.11.2009	0.5	Исходная версия	Е. П. Моргунов

1. Общие требования

1. 1. Идентификаторы

Длина идентификаторов – не более 32 символов.

Составные имена идентификаторов должны формироваться с использованием символа подчеркивания:

```
$target_win
```

Не должны использоваться заглавные буквы в начале каждого компонента идентификатора. Пример недопустимого идентификатора:

```
$TargetWin
```

В именах идентификаторов не должны использоваться русские слова, записанные латинскими буквами. Пример недопустимого идентификатора:

```
$chislo_zapisey
```

1. 2. Структурные отступы

В качестве структурного отступа следует использовать 2 символа «пробел», символов табуляции не использовать.

1.3. Комментарии

При объявлениях переменных короткие комментарии следует выравнивать в одну колонку:

```
my( $conn, $result, $i );       # для работы с БД
my @row;                        # для хранения одной записи из выборки
my $command;                    # строка команды для работы с БД

my $keyword_code;               # код ключевого слова, выбранного
                                # пользователем
my @keywords = ();              # массив кодов ключевых слов,
                                # которые по "цепочке" связаны со словом,
                                # выбранным пользователем
my $headline;                   # заголовок списка биб. описаний
```

1.4. Заголовки модулей

В каждом программном модуле и конфигурационном файле должен присутствовать заголовок следующего вида:

```
# ------------------------------------------------------------------------
# Система:  "Электронный архив"
# Модуль:   ключевые слова
# Автор:    Моргунов Е.П.
# Дата:     8.02.2006
# Версия:   0.1
# ------------------------------------------------------------------------
```

2. Переменные

2.1. Скалярные переменные

При оформлении SQL-операторов в виде символьных строк следует придерживаться следующих правил:

– использовать оператор конкатенации;

– ключевые слова языка SQL писать заглавными буквами и выравнивать в одну колонку, в подзапросах выравнивать также по одной колонке;

– слева и справа от символов операций (=, <, > и др.) ставить по одному пробелу:

```
$command = "SELECT k1.keyword_code, k1.keyword_name " .
           "FROM keywords k1 " .
           "WHERE k1.keyword_code = $keyword_code OR " .
                 "k1.keyword_code = ( SELECT k2.keyword_code_orig " .
                                     "FROM keywords k2 " .
                                     "WHERE k2.keyword_code = " .
                                           "$keyword_code ) OR " .
                 "k1.keyword_code_orig = $keyword_code OR " .
                 "k1.keyword_code_orig = ( SELECT k3.keyword_code_orig " .
                                          "FROM keywords k3 " .
                                          "WHERE k3.keyword_code = " .
```

```
                                                    "$keyword_code ) " .
          "ORDER BY k1.keyword_name";
```

2.2. Массивы и хеш-массивы

При инициализации массивов с большим количеством элементов следует использовать подобную конструкцию:

```
# массив имен полей таблицы Persons
my @f_names =
   ( 'pers_id',          # Уникальный идентификатор персоналии
     'pers_name',        # Полное имя, отчество, фамилия
                         # ПРИМЕЧАНИЕ. В поле записывать первой
                         #             фамилию и ставить запятую.
     'pers_life_years', # Годы жизни
     'pers_add_info',    # Дополнительные сведения (младший/старший,
                         # отец/сын, ... род занятий, ...)
     'pers_id_orig',     # Код персоналии, являющейся началом цепочки
                         # (для учета разночтений, написаний на различных
                         # языках)
     'lang_code',        # Код языка, на котором приведено написание
                         # Ф. И. О. персоналии (1 - русский язык)
     'process_code',     # Код вида обработки имени персоналии
                         # (1 - оригинальное, т.е. в том виде, как
                         # приведено в источнике библиограф. описания)
     'pers_addr',        # Адрес
     'pers_email',       # Электронный адрес
     'comment'           # Примечание
   );
```

При обращении к элементу массива необходимо ставить пробелы слева и справа от индекса:

```
$esc_val = $hot_help_info[ 1 ][ 0 ][ 1 ];
```

Оформление хеш-массивов аналогично оформлению массивов.

3. Процедуры

3.1. Тело процедуры

При оформлении тела процедуры следует придерживаться следующих правил:

– в имени процедуры первым компонентом по возможности должен быть глагол, отражающий суть операций, выполняемых этой процедурой;

– должны быть описаны все параметры, получаемые процедурой;

– должно быть приведено краткое наименование (назначение) процедуры.

```
# ------------------------------------------------------------------------
# определение полномочий пользователя на выполнение указанной процедуры
# указанного модуля
# ------------------------------------------------------------------------
sub can_work
{
```

```
  # параметры: имя пользователя, условные имена модуля и процедуры,
  #            CGI-объект
  my $user      = shift;
  my $module    = shift;
  my $procedure = shift;
  my $query     = shift;
. . . .
. . . .
. . . .
  return ( 1 );
}
```

В следующем примере комментарии выровнены в одну колонку:

```
# ---------------------------------------------------------------------
# формирование текста функции на языке JavaScript, которая выполняет
# присваивание значений полей из справочника полям на главной форме,
# находящейся в ВЫЗЫВАЮЩЕМ модуле
# ---------------------------------------------------------------------
sub create_assign_values_func
{
  # параметры
  my $entity_id_fname = shift;   # имя поля, являющегося идентификатором
                                 # некой сущности (ради этого поля все и
                                 # затевается) (например, код персоналии)
  my $entity_name_fname = shift; # имя поля, являющегося описанием некой
                                 # сущности (например, Ф. И. О. персоналии)
  my $add_info_fname = shift;    # имя поля, содержащего дополнительную
                                 # информацию (его может не быть)
```

3.2. Вызовы процедур

При вызове процедур необходимо ставить пробелы после открывающей скобки и перед закрывающей скобкой:

```
make_letters( $script_name, $add_params );
```

При вызовах процедур в случае наличия большого количества параметров должна применяться подобная конструкция:

```
common_modes( query => $query,
              module => $module,
              user => $user,
              make_form => \&make_form,
              make_command => \&make_command,
              check_param => \&check_param,
              create_condition => \&create_condition,
              do_query => \&do_query,
              do_view => \&do_view,
              do_add => \&do_add,
              do_del => \&do_del,
              do_find => \&do_find,
              add_rec => \&add_rec,
              update_rec => \&update_rec,
              action => $script_name
            );
```

Скобки допустимо расставлять и таким образом:

```
http_html_headers( query => $query,
                   target => "work",
                   BGCOLOR => $CFG{ COLOR_WORK } );
```

4. Управляющие конструкции

4.1. Условные операторы

При оформлении условных операторов следует придерживаться следующих правил:

– ставить пробел после ключевого слова, а также после открывающей и перед закрывающей скобками в условном выражении;

– фигурные скобки, ограничивающие тело блока операторов, ставить под первым символом ключевого слова.

```
if ( ! $query->param( 'enable_select' ) )
{
  goto_main_page();
}
```

Вложенные условные операторы оформлять по этим же правилам:

```
# если модуль вызван в режиме справочника для оказания помощи пользователю
# при выборе конкретного ключевого слова ...
if ( $query->param( 'enable_select' ) )
{
  # ... то сформируем функцию JavaScript для присваивания значений полей
  # из текущей записи таблицы Keywords полям в ВЫЗЫВАЮЩЕМ окне ...
  if ( $query->param( 'caller_name' ) eq 'OBJ_KEYWORDS' )
  {
    # ... полям keyword_code и keyword_name в модуле OBJ_KEYWORDS
    create_assign_values_func( 'keyword_code', 'keyword_name' );
  }
  elsif ( $query->param( 'caller_name' ) eq 'KEYWORDS' )
  {
    # ... полям keyword_code_orig и keyword_name_orig в модуле KEYWORDS
    create_assign_values_func( 'keyword_code_orig', 'keyword_name_orig' );
  }
}
```

4.2. Циклы

При оформлении циклов следует придерживаться следующих правил:

– ставить пробел после ключевого слова, а также после открывающей и перед закрывающей скобками в условии цикла;

– фигурные скобки, ограничивающие тело цикла, ставить под первым символом ключевого слова.

```
while ( @row = $result->fetchrow )
{
  $headline .= $row[ $result->fnumber( 'keyword_name' ) ] . "; ";
  push( @keywords, $row[ $result->fnumber( 'keyword_code' ) ] );
}
```

4.3. Оператор возврата

При оформлении операторов возврата следует придерживаться следующих правил:

– ставить пробел после ключевого слова, а также после открывающей и перед закрывающей скобками.

```
return ( $command );
```

5. Прочие требования

5.1. Сложно-структурируемые операторы

Операторы со сложной структурой могут возникать, когда на языке Perl формируются конструкции другого языка (HTML или JavaScript). В таких случаях необходимо принимать индивидуальные решения в соответствии с общим стилем оформления исходных текстов, описанным в настоящем документе, в частности:

– необходимо соблюдать правила использования пробелов при вызовах процедур и при обращениях к элементам массивов и хеш-массивов;

– в тех случаях, когда символ «пробел» имеет не оформительский смысл, а функциональный, необходимо исходить именно из функционального смысла.

```
print "<TD>",
      qq{ <A HREF = "javascript:void( 0 )" onClick =
          "assign_values( $row[ $result->fnumber( 'keyword_code' ) ],
                          '$keyword_name' )">
           $row[ $result->fnumber( 'keyword_name' ) ]</A> },
      "</TD>\n";

print "<A HREF = \"javascript:void( 0 )\"
          NAME = \"$hot_help_info{ $key }[ 0 ][ 1 ]\"
          onClick =
          \"if ( $field_name.disabled == false )
          { $field_name.value = '$esc_val';
             $additional_code;
             $field_name.focus();
          }
          else
          {
            window.alert( 'Поле заблокировано. Ввод невозможен' );
          }
          \">" . $hot_help_info{ $key }[ 0 ][ 1 ] . "</A>\n";
```

Приложение 5. Описание структуры базы данных

```
-- ------------------------------------------------------------------------
-- Система:   "Электронный архив"
-- Модуль:    Структуры таблиц базы данных "Электронный архив"
-- Автор:     Моргунов Е. П.
-- Дата:      01.11.2010
-- Версия:    0.5
-- ------------------------------------------------------------------------

. . . . . .
. . . . . .
-- ------------------------------------------------------------------------
-- Таблица "Языки"
-- ------------------------------------------------------------------------
CREATE TABLE Langs                    -- Languages
  ( lang_code int2,                   -- Код языка
    lang_fname text                   -- Наименование языка полное
      UNIQUE NOT NULL,
    lang_sname text                   -- Наименование языка краткое
      UNIQUE NOT NULL,
    comment text,                     -- Примечание
    -- служебная информация
    who_chg text DEFAULT USER,        -- Кем добавлена (изменена) запись
    when_chg timestamp
      DEFAULT CURRENT_TIMESTAMP,      -- Когда добавлена (изменена) запись
    PRIMARY KEY ( lang_code )
  );


-- ------------------------------------------------------------------------
-- Таблица "Персоналии"
-- ------------------------------------------------------------------------
CREATE TABLE Persons
  ( pers_ID int4,               -- Уникальный идентификатор персоналии
    pers_name text NOT NULL,    -- Полное имя, отчество, фамилия
                                -- ПРИМЕЧАНИЕ. В поле записывать первой
                                --              фамилию и ставить запятую.
    pers_life_years text,       -- Годы жизни
    pers_add_info text,         -- Дополнительные сведения
                                -- (младший/старший,
                                -- отец/сын, ... род занятий, ...)
    pers_ID_orig int4,          -- Код персоналии, являющейся началом
                                -- цепочки
                                -- (для учета разночтений, написаний
                                -- на различных языках)
    lang_code int2              -- Код языка, на котором приведено
                                -- написание
      NOT NULL DEFAULT 1,       -- Ф. И. О. персоналии (1 - русский язык)
    process_code int2           -- Код вида обработки имени персоналии
      NOT NULL DEFAULT 1,       -- (1 - оригинальное, т.е. в том виде, как
                                -- приведено в источнике библиограф.
                                -- описания)
    pers_addr text,             -- Адрес
    pers_email text,            -- Электронный адрес
    comment text,               -- Примечание
    -- служебная информация
    who_chg text DEFAULT USER,   -- Кем добавлена (изменена) запись
    when_chg timestamp
      DEFAULT CURRENT_TIMESTAMP, -- Когда добавлена (изменена) запись
    PRIMARY KEY ( pers_ID ),
```

```
    FOREIGN KEY ( lang_code )
      REFERENCES Langs ( lang_code )
      ON DELETE SET NULL
      ON UPDATE CASCADE,
    FOREIGN KEY ( process_code )
      REFERENCES Process_types ( process_code )
      ON DELETE SET NULL
      ON UPDATE CASCADE,
    FOREIGN KEY ( pers_ID_orig )
      REFERENCES Persons ( pers_ID )
      ON DELETE SET NULL
      ON UPDATE CASCADE
  );

CREATE INDEX Persons_ind_1 ON Persons ( pers_name );

. . . . . .
. . . . . .
```

Приложение 6. Титульный лист

Министерство образования и науки Российской Федерации

Сибирский государственный аэрокосмический университет
имени академика М. Ф. Решетнева

Институт информатики и телекоммуникаций

Кафедра информатики и вычислительной техники

Курсовой проект по дисциплине
«Технология программирования»

Тема: _____

Выполнил: студент гр. БИ-91
И. И. Хакеров
Проверил: Е. П. Моргунов

г. Красноярск – 2011

Приложение 7. Содержание

Содержание

Учебное издание

МОРГУНОВ Евгений Павлович
МОРГУНОВА Ольга Николаевна

Технология программирования

Курсовое проектирование

Учебно-методическое пособие

Печатается в авторской редакции
Компьютерная верстка *Е. П. Моргунова*

Подписано в печать 7.02.2011. Формат 60×84/16. Бумага офсетная.
Печать плоская. Усл. печ. л. 5,11. Уч.-изд. л. 3,62. Тираж 100 экз.
Заказ № ___.

Отпечатано в отделе копировально-множительной техники
Сиб. гос. аэрокосмич. ун-та.
660014, г. Красноярск, просп. им. газ. «Красноярский рабочий», 31.