МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Омский государственный технический университет»

РАБОЧАЯ ПРОФЕССИЯ «ОПЕРАТОР ЭВМ»: ТЕОРИЯ И ПРАКТИКА

Учебное текстовое электронное издание
локального распространения

Омск
Издательство ОмГТУ
2015

Сведения об издании: 1, 2

ISBN 978-5-8149-1965-6

УДК 004
ББК 32.97я73
Р13

Авторы:
А. Г. Белик, *В. Н. Цыганенко*, *О. Б. Малков*, *А. Г. Анатольев*, *Р. Н. Богатов*

Рецензенты:
С. Н. Чуканов, д-р техн. наук, профессор, зав. кафедрой «Компьютерные информационные автоматизированные системы» ФГБОУ ВПО «Сибирская государственная автомобильно-дорожная академия (СибАДИ)»;

П. Н. Надточий, канд. физ.-мат. наук,
зав. сектором разработки ПО ЗАО «Автоматика-Э»

Рабочая профессия «Оператор ЭВМ»: теория и практика : практикум / [А. Г. Белик и др.] ; Минобрнауки России, ОмГТУ. – Омск : Изд-во ОмГТУ, 2015.

Р13

ISBN 978-5-8149-1965-6

Приведены требования к выполнению практических заданий по дисциплине «Рабочая профессия». Даны основы теоретических знаний для приобретения навыков в области современных компьютерных информационных технологий с целью решения разнообразных прикладных задач с использованием операционных и офисных программных систем, сетевых и веб-технологий, инструментальных сред программирования, систем управления базами данных, программных комплексов схемотехнического и структурного моделирования.

Для студентов бакалавриата, обучающихся по следующим направлениям подготовки: 09.03.01 «Информатика и вычислительная техника», 09.03.04 «Программная инженерия», 27.03.03 «Системный анализ и управление».

УДК 004
ББК 32.97я73

Рекомендовано редакционно-издательским советом
Омского государственного технического университета

ISBN 978-5-8149-1965-6

1 электронный оптический диск

Оригинал-макет издания выполнен в Microsoft Office Word 2007
с использованием возможностей Adobe Acrobat X.

Минимальные системные требования:

- процессор Intel Pentium 1,3 ГГц и выше;
- оперативная память 256 Мб;
- свободное место на жестком диске 260 Мб;
- операционная система Microsoft Windows XP/Vista/7;
- разрешение экрана 1024×576 и выше;
- акустическая система не требуется;
- дополнительные программные средства Adobe Acrobat Reader 5.0 и выше.

Редактор *К. В. Муковоз*
Компьютерная верстка *Ю. П. Шелехиной*

Сводный темплан 2015 г.
Подписано к использованию 20.04.15.
Объем 2,35 Мб.

Издательство ОмГТУ.
644050, г. Омск, пр. Мира, 11; т. 23-02-12
Эл. почта: info@omgtu.ru

ВВЕДЕНИЕ

Издание предназначено для профессиональной подготовки операторов ЭВМ – профессии, которая в настоящее время является одной из наиболее востребованных на различных предприятиях. Оператор ЭВМ должен уметь: выполнять ввод информации в ЭВМ с различных носителей и осуществлять вывод информации; выполнять запись, считывание, копирование и перезапись информации с носителей одного вида на другой; устанавливать причины неполадок в работе ЭВМ; устранять небольшие проблемы в работе программного обеспечения и ЭВМ; работать с базами данных; оформлять результаты работ.

Целью дисциплины «Рабочая профессия» является получение студентами основ теоретических знаний и практических навыков в области современных компьютерных информационных технологий для решения разнообразных прикладных задач с использованием операционных и офисных программных систем, сетевых и веб-технологий, инструментальных сред программирования, систем управления базами данных, программных комплексов схемотехнического и структурного моделирования.

В задачи дисциплины входит:

– формирование систематизированного представления о концепциях, моделях и принципах организации, положенных в основу системно-аналитических, информационно-управляющих, конструкторско-технологических, проектирующих технологий и систем, которые требуют исследования, моделирования, программирования и управления на основе использования современного программного инструментария;

– получение практической подготовки в области веб-технологий при удаленном доступе в системах и распределенных вычислениях при выполнении работ по учету, контролю, планированию и управлению производственных процессов, предприятий, эксплуатации программных систем обработки информации и управления в административной, хозяйственной и управленческой деятельности в различных областях промышленности;

– выработка навыков оценивания современного состояния и перспективных направлений развития технологий на основе системного анализа, управления, моделирования, производства и эксплуатации технических систем, объектов и устройств различного назначения.

1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ РАБОЧЕЙ ПРОФЕССИИ

1.1. Направление «Программная инженерия»

Областью профессиональной деятельности выпускников по направлению подготовки «Программная инженерия» является индустриальное производство программного обеспечения (ПО) для информационно-вычислительных систем различного назначения. Разберемся в том, что это означает.

Программирование – это творческий процесс, выполняемый человеком и включающий осмысление задач, описание алгоритмов и составление программ для получения решения на компьютерах. Программирование может происходить в различных контекстах: для удовольствия, в учебных целях, в рамках научных разработок. Особое место занимает *промышленное (индустриальное) программирование*, которое, как правило, осуществляется в команде и обязательно для заказчика, который платит деньги за выполненную работу.

При этом необходимо точно понимать, что нужно заказчику, выполнить работу в определенные сроки и получить результат нужного качества (который удовлетворит заказчика и за который он заплатит).

Чтобы удовлетворить этим требованиям, процесс программирования «обрастает» различными дополнительными видами деятельности:

– разработка требований;

– планирование;

– тестирование;

– конфигурационное управление;

– проектный менеджмент;

– создание документации (проектной, пользовательской и пр.) и т. д.

Написание программного кода предваряется анализом и проектированием:

– *анализ* – создание функциональной модели будущей системы без учета реализации, для осознания программистами требований и ожиданий заказчика;

– *проектирование* – предварительный макет, эскиз, план системы на бумаге.

Все эти и другие дополнительные виды деятельности, выполняемые в процессе промышленного программирования и необходимые для успешного выполнения заказов, называются *программной инженерией (Software Engineering)*.

Появление программной инженерии связано с ситуацией, возникшей в конце 1960-х гг. в США и охарактеризованной как *кризис программного обеспечения*. Программирование не успевало за стремительным развитием технических средств:

– большие проекты стали выполняться с отставанием от графика или с превышением сметы расходов;

– разработанный продукт не обладал требуемой функциональностью, качество ПО не устраивало потребителей;

– методы и процессы программирования, эффективно работавшие для одного человека (или небольшой команды) при разработке программ умеренных размеров, не приводили к успеху при разработке крупных и сложных систем.

Причинами неудач были: нечеткая и неполная формулировка требований к ПО, недостаточное вовлечение пользователей в работу над проектом, отсутствие необходимых ресурсов, неудовлетворительное планирование и управление проектом, частое изменение требований и спецификаций, новизна и несовершенство используемой технологии, отсутствие грамотного управления проектом.

Одной из главных причин было то, что множество проектов выполнялось *в экстремальных условиях.* Возник даже термин «Death March» («смертельный марш») для обозначения проекта, параметры которого отклоняются от нормальных значений, по крайней мере, на 50 %. Ситуация возникает, когда план проекта необоснованно сжат более чем вдвое, количество разработчиков уменьшено более чем вдвое, бюджет и связанные с ним ресурсы урезаны вдвое, требования к функциям и производительности вдвое превышают значения в нормальных условиях.

Постепенно общество осознало то, что создание систем ПО – это сложная, трудоемкая и длительная работа, требующая высокой квалификации. При этом она часто выполняется на интуитивном уровне с применением неформализованных методов, основанных на искусстве, практическом опыте, экспертных оценках и дорогостоящих экспериментальных проверках. Пришло и понимание того, что отношение к разработке ПО должно быть более серьезным, возникла объективная потребность в регламентации и стандартизации процессов разработки.

Мировое сообщество осознало необходимость в программной инженерии через 20 лет после рождения самого программирования, если считать таковым отчет фон Неймана «First Draft of a Report on the EDVAC», обнародованный в 1945 г.

В октябре 1968 г. на конференции подкомитета НАТО по науке и технике (г. Гармиш, Германия) впервые был озвучен термин «Software Engineering» как дисциплина, которую надо создавать и которой надо руководствоваться. В том же 1968 г. на встрече руководителей проектов по разработке ПО (Лондон) впервые определена концепция *жизненного цикла ПО* (Software Lifetime Cycle) как последовательность шагов (стадий) в процессе создания и эксплуатации ПО. В 1970 г. Уинстон Ройс выделил стадии жизненного цикла и предположил, что контроль выполнения стадий приведет к повышению качества ПО и сокра-

щению стоимости разработки. Наконец, в 1975 г. в Вашингтоне состоялась первая международная конференция по программной инженерии.

В Советском Союзе в 1970-х гг. академик Андрей Петрович Ершов (1931–1988) перевел термин «Software Engineering» как «технология программирования». Термин «программная инженерия» предложил в конце 1990-х выдающийся ученый-информатик Игорь Васильевич Поттосин (1933–2001).

Таким образом, была выявлена проблема (потребность контролировать процесс разработки ПО, прогнозировать и гарантировать стоимость, сроки и качество) и предложено ее решение – переход от кустарных к индустриальным способам создания ПО и создание программной инженерии.

Программная инженерия – это инженерная дисциплина, связанная со всеми аспектами производства ПО от создания спецификации до поддержки системы после сдачи в эксплуатацию.

Фундаментальные идеи программной инженерии:

– проектирование ПО является формальным процессом, который можно изучать и совершенствовать;

– ПО создается в результате выполнения нескольких взаимосвязанных этапов (анализ требований, проектирование, разработка, внедрение, сопровождение), составляющих жизненный цикл программного продукта.

Таким образом, в процесс разработки ПО вносится *специфика инженерной деятельности* в целом, для которой характерно следующее:

– инженеры принимают решения, оценивая альтернативы и выбирая в каждой точке принятия решения подход, оптимально соответствующий задаче. При анализе альтернатив сопоставляются возможные затраты и ожидаемая прибыль;

– инженеры работают с измеримыми количественными характеристиками. Они совершенствуют и уточняют методы измерений и при необходимости выдают приближенные решения на основе опыта и эмпирических данных;

– инженеры придают особое значение соблюдению технологического процесса и понимают важность эффективной организации командной работы;

– инженеры отвечают за выполнение разных задач, начиная с исследований, разработки, проектирования, производства, тестирования, внедрения, эксплуатации и управления и заканчивая продажами, сопровождением, консультированием и обучением;

– инженеры широко используют инструментальные средства. Выбор и использование наиболее эффективных средств является важным вопросом;

– инженеры повторно используют результаты проектирования и проектные артефакты. Проектирование является важной составляющей любой инженерной деятельности и играет критически важную роль при разработке ПО.

Инженеры – специалисты, выполняющие практическую работу и добивающиеся практических результатов. Они всегда пытаются найти решение, даже когда теоретических основ решения для данной задачи еще нет. Инженер ищет метод или средство решения задачи, применяет его и несет ответственность за результат. Инженеры работают в условиях ограниченных ресурсов. Набор инженерных методов, теоретически, возможно, не обоснованных, но получивших подтверждение на практике, в программной инженерии получил название лучших практик.

Объектами профессиональной деятельности выпускников по направлению подготовки «Программная инженерия» являются:

– программный проект (проект разработки программного продукта);

– программный продукт (создаваемое программное обеспечение);

– процессы жизненного цикла программного продукта;

– методы и инструменты разработки программного продукта;

– персонал, участвующий в процессах жизненного цикла.

Программное обеспечение (Software) – набор компьютерных программ, процедур и связанной с ними документации и данных. Продается (поставляется) не только программа, но и *документация*, в которой можно прочитать, как установить и использовать программу и *данные* для установки программы в разных условиях (конфигурационные файлы). Поэтому ПО называют *программным продуктом*. Промышленное программирование имеет целью разработку именно программных продуктов, т. е. такого ПО, которое может быть продано потребителю.

Бакалавры по направлению подготовки «Программная инженерия» готовятся к различным видам профессиональной деятельности. При этом в соответствии с видами деятельности решаются следующие профессиональные задачи.

1. Научно-исследовательская деятельность:

– участие в проведении научных исследований, связанных с объектами профессиональной деятельности;

– построение моделей объектов профессиональной деятельности с использованием инструментальных средств компьютерного моделирования;

– составление описания проводимых исследований, подготовка данных для составления обзоров и отчетов.

2. Аналитическая деятельность:

– сбор и анализ требований заказчика к программному продукту;

– формализация предметной области программного проекта по результатам технического задания и экспресс-обследования;

– содействие заказчику в оценке и выборе вариантов ПО;

– участие в составлении коммерческого предложения заказчику, подготовке презентации и согласовании пакета договорных документов.

3. Проектная деятельность:

– участие в проектировании компонентов программного продукта;

– создание компонентов программного обеспечения (кодирование, отладка, модульное и интеграционное тестирование);

– выполнение измерений и рефакторинг кода в соответствии с планом;

– участие в интеграции компонентов программного продукта;

– разработка тестового окружения, создание тестовых сценариев;

– разработка и оформление эскизной, технической и рабочей проектной документации.

4. Технологическая деятельность:

– освоение и применение средств автоматизированного проектирования, разработки, тестирования и сопровождения ПО;

– освоение и применение методов и инструментальных средств управления инженерной деятельностью и процессами жизненного цикла ПО;

– использование типовых методов контроля, оценки и обеспечения качества программной продукции;

– обеспечение соответствия разрабатываемого ПО и технической документации российским и международным стандартам, техническим условиям, ведомственным нормативным документам и стандартам предприятия.

5. Производственная деятельность:

– взаимодействие с заказчиком в процессе выполнения проекта;

– участие в процессах разработки ПО;

– участие в создании технической документации по результатам работ.

6. Педагогическая деятельность:

– проведение обучения и аттестации пользователей программных систем;

– участие в разработке методик обучения технического персонала и пособий по применению программных систем.

7. Организационно-управленческая деятельность:

– участие в составлении технической документации (графиков работ, инструкций, планов, смет, заявок на материалы, оборудование, ПО) и установленной отчетности по утвержденным формам;

– планирование и организация собственной работы;

– планирование и координация работ по настройке и сопровождению программного продукта;

– составление технического задания на разработку программного продукта;

– организация работы малых коллективов исполнителей проекта;

– участие в проведении технико-экономического обоснования проектов.

8. Сервисно-эксплуатационная деятельность:

– ввод в эксплуатацию ПО (инсталляция, настройка параметров, адаптация, администрирование);

– профилактическое и корректирующее сопровождение программного продукта в процессе эксплуатации;

– обучение и консультирование пользователей по работе с программной системой.

1.2. Направление «Информатика и вычислительная техника»

Информатика и вычислительная техника – обширная область профессиональной деятельности, охватывающая компьютерные системы разного уровня, и в первую очередь – автоматизированные (человеко-машинные) системы обработки информации и управления.

Объектами деятельности здесь являются все виды обеспечения (математическое, информационное, техническое, лингвистическое, эргономическое, организационное и правовое) вычислительных машин, комплексов, систем и сетей. Объектами профессиональной деятельности студентов данного направления также являются:

– вычислительные машины, комплексы, системы и сети;

– автоматизированные системы обработки информации и управления;

– системы автоматизированного проектирования и информационной поддержки жизненного цикла промышленных изделий;

– программное обеспечение средств вычислительной техники и автоматизированных систем (программы, программные комплексы и системы).

Бакалавр по профилю подготовки «Автоматизированные системы обработки информации и управления» должен решать следующие профессиональные задачи в соответствии с видами профессиональной деятельности.

Проектно-конструкторская деятельность:

– сбор и анализ исходных данных для проектирования;

– проектирование программных и аппаратных средств (систем, устройств, деталей, программ, баз данных и т. п.) в соответствии с техническим заданием с использованием средств автоматизации проектирования;

– разработка и оформление проектной и рабочей технической документации;

– контроль соответствия разрабатываемых проектов и технической документации стандартам, техническим условиям и другим нормативным документам;

– проведение предварительного технико-экономического обоснования проектных расчетов.

Проектно-технологическая деятельность:

– применение современных инструментальных средств при разработке программного обеспечения;

– применение веб-технологий при реализации удаленного доступа в системах клиент/сервер и распределенных вычислений;

– использование стандартов и типовых методов контроля и оценки качества программной продукции;

– участие в работах по автоматизации технологических процессов в ходе подготовки производства новой продукции;

– освоение и применение современных программно-методических комплексов исследования и автоматизированного проектирования объектов профессиональной деятельности.

Научно-исследовательская деятельность:

– изучение научно-технической информации, отечественного и зарубежного опыта по тематике исследования;

– математическое моделирование процессов и объектов на базе стандартных пакетов автоматизированного проектирования и исследований;

– проведение экспериментов по заданной методике и анализ результатов;

– проведение измерений и наблюдений, составление описания проводимых исследований, подготовка данных для составления обзоров, отчетов и научных публикаций;

– составление отчета по выполненному заданию, участие во внедрении результатов исследований и разработок.

Научно-педагогическая деятельность:

– обучение персонала предприятий применению современных программно-методических комплексов исследования и автоматизированного проектирования.

Монтажно-наладочная деятельность:

– наладка, настройка, регулировка и опытная проверка ЭВМ, периферийного оборудования и программных средств;

– сопряжение устройств и узлов вычислительного оборудования, монтаж, наладка, испытание и сдача в эксплуатацию вычислительных сетей.

Сервисно-эксплуатационная деятельность:

– инсталляция программ и программных систем, настройка и эксплуатационное обслуживание аппаратно-программных средств;

– проверка технического состояния и остаточного ресурса вычислительного оборудования, организация профилактических осмотров и текущего ремонта;

– приемка и освоение вводимого оборудования;

– составление заявок на оборудование и запасные части, подготовка технической документации на ремонт;

– составление инструкций по эксплуатации оборудования и программ испытаний.

1.3. Направление «Системный анализ и управление»

В последние годы многие крупные компании начали замену старых принципов организации управления новыми. Это связано с оптимизацией и автоматизацией основных бизнес-процессов, производимых за счет внедрения корпоративных систем управления ресурсами и планирования. Одну из ключевых ролей при решении этой задачи играют такие профессиональные специалисты, как системные аналитики. Иногда их еще называют бизнес-аналитиками.

Системный аналитик занимается анализом предметной области и формулированием требований к разрабатываемым информационным системам и прикладному программному обеспечению. Он должен уметь задавать «правильные» вопросы и получать «нужные» ответы, поэтому кандидату на данную позицию должны быть присущи такие качества, как системность, коммуникабельность и представительность.

Как выглядит управление основными бизнес-процессами компании или ее подразделения без участия отдела системного анализа или системного аналитика для небольшого подразделения?

Руководитель как лицо, принимающее решение на основе стратегии деятельности компании, ставит задачи подразделениям и следит за их выполнением. Для правильного позиционирования компании на рынке и достижения лучших показателей деятельности руководитель должен на основе анализа результатов ее деятельности вести правильную политику в областях: инвестиционной, технической, социальной, кадровой и др. Показателей, которые характеризуют все виды деятельности компании, как правило, достаточно много, поэтому для выбора правильной стратегии деятельности руководителю необходим орган для сбора данных, анализа результатов деятельности и научного обоснования в принятии тех или иных решений.

В крупных компаниях, понимая необходимость указанной деятельности, широко внедряются информационные технологии, базы данных и создаются специализированные вычислительные центры, основной задачей которых остается сбор данных и составление оперативных отчетов о деятельности.

Базы данных постоянно обновляются, но для принятия решений используется только мизерная часть накопленных данных. Для объективного обоснова-

ния принимаемых руководителем решений необходимо использование всего объема накопленной информации, методов математического моделирования деятельности компании, а также методов научного обоснования оптимальных по заданному критерию решений.

Постановка задач анализа данных непосредственно программистам не приводит к желаемым результатам, поскольку программисты, как правило, не знакомы с математическими методами моделирования деятельности и зачастую используют при разработке собственную модель, лишь (в лучшем случае) согласовав ее с конечными пользователями. По мнению некоторых экспертов, при подобном подходе вскоре наступает момент, когда либо модели перестают соответствовать требованиям реального бизнеса, либо проекту начнут угрожать внутренние противоречия.

Ключевая роль системного аналитика в проекте автоматизации компании заключается в разработке непротиворечивой и полной модели требований бизнеса к внедряемому программному обеспечению. Это задача требует от специалиста не только описательных способностей, но и незаурядных коммуникативных навыков для выяснения потребностей руководителей компании на всех ее уровнях. Как правило, сначала системный аналитик собирает требования к новому программному продукту (математической модели), после чего разрабатывает техническое задание на создание программного обеспечения, проектирует документальное оформление системной и программной архитектуры IT-системы, ставит задачи на разработку и проводит тестирование. По окончании же проекта объясняет правила работы пользователям и решает проблемы функционирования на всех стадиях жизненного цикла созданной системы.

Работа системного аналитика состоит в решении нескольких задач.

1. *Разработка полной и непротиворечивой модели бизнес-процессов компании* на основании общения с клиентами. Сбор информации для создания бизнес-модели может быть проведен с использованием таких видов общения, как анкетирование, переписка, интервью, совещания и переговоры, а также при работе с документами-источниками.

2. *Разработка технического задания для реализации заложенных требований*. Знание предметной области, а также владение зарекомендовавшими себя нотациями: IDEF, UML, DFD и другими – позволяют бизнес-аналитику решить задачу непротиворечивости и целостности составляемой им модели. Для специалистов данной области важным является знание архитектуры стандартного решения компании-поставщика и принципов современной архитектуры информационных систем в целом.

3. *Документирование архитектуры бизнес-процессов*. По мере получения информации от клиента и моделирования исследуемых процессов системный

аналитик ведет документацию. Только полнота и актуальность всех участвующих в проекте документов позволят при необходимости вовлечь в него новых сотрудников. Иначе при увольнении бизнес-аналитика вместе с ним «уплывают» и знания, полученные на этапе общения с клиентом.

4. *Объяснение правил работы с системой пользователям*. Особая роль принадлежит системному аналитику на этапе внедрения программного обеспечения в реальный бизнес-процесс. Здесь и открываются все узкие места как модели, так и архитектуры предлагаемого клиенту решения. Важно уметь вовремя увидеть сложности и, имея гибкую модель и не менее гибкое решение, совершить своевременные доработки по запросам заказчика.

Типичные *функциональные обязанности системного аналитика*:

– изучение той или иной области на предмет внедрения и/или разработки прикладных информационных систем;

– участие в интервьюировании экспертов и пользователей информационных систем на предмет изучения текущих принципов организации хода процессов (в том числе с точки зрения функционирования информационных систем);

– изучение и систематизация документации по проекту в части выделения процессов, подлежащих оптимизации и автоматизации;

– подготовка документации по описанию сущностей, взаимосвязей и процессов предметной области с использованием специальных нотаций;

– участие в постановке задач и разработке технического задания;

– сбор, анализ и документирование функциональных требований к программному обеспечению систем автоматизации.

Также в круг обязанностей системного аналитика может входить *выполнение дополнительных функций*:

– участие в подготовке схем тестирования функционала для выявления отклонений от сформулированных бизнес-требований и функциональных требований;

– участие в тестировании прототипа разрабатываемой системы;

– участие в обучении пользователей системы;

– анализ рисков и причин возникновения ошибок при разработке систем;

– участие в выборе платформы для реализации проекта.

Системный аналитик должен:

– знать основы программирования (в том числе объектно-ориентированного), проектирования, разработки, документирования программного обеспечения;

– знать основы теории алгоритмов, теории баз данных, теории систем и системного анализа, основы безопасности информации;

– знать основы проектирования человеко-машинных интерфейсов;

– обладать общими знаниями в области менеджмента, экономики, бухгалтерского и управленческого учета.

Ключевыми *навыками для системного аналитика* являются:

– способность быстро понять требования и определить их приоритет, а также рассказать о технических решениях и их влиянии на бизнес понятным клиенту языком;

– умение в различных проектах следовать принятой методологии, нотациям и формам документов, навыки работы с соответствующим программным обеспечением;

– способность к коллективной работе с другими аналитиками в случае, если над проектом трудится команда;

– умение, сохраняя творческий стиль работы, соблюдать дисциплину в отношении ведения документов, версий, протоколов и готовность трудиться в команде с архитекторами, разработчиками, тестировщиками.

Системный аналитик *должен уметь использовать современное программное обеспечение* в областях:

– системного анализа и формализации результатов обследования предметной области;

– использования методик проектирования и разработки информационных систем;

– подготовки проектной документации по разработке программного обеспечения;

– разработки функциональных требований;

– разработки схем тестирования программного обеспечения;

– программирования на языках высокого уровня (Pascal, C++);

– формализации сущностей и ассоциаций, документирования, моделирования процессов с использованием специальных методологий и нотаций (ERD, DFD, WFD, IDEF*, ARIS, UML, BPML и др.);

– использования специального программного обеспечения, реализующего методологии и нотации (Process Modeler (BPWin), Data Modeler (ERWin), ARIS Toolset, Rational Rose и т. п.).

Если говорить о личностных качествах, то системному аналитику важно быть педантичным и исполнительным, кроме того, чувствовать зону своей ответственности в проекте и не вмешиваться как в финансовые, так и в политические вопросы.

2. ПРАКТИЧЕСКИЕ ЗАНЯТИЯ

2.1. Основы работы в MS Word, MS Excel, MS PowerPoint

Цель работы: приобретение практических навыков по подготовке, редактированию и оформлению текстовой документации, графиков, диаграмм и рисунков, обработке числовых данных в электронных таблицах.

Задание

1. Построить график функции с использованием MS Excel, создать макрос для построения графика функции (по методическим указаниям и индивидуальному заданию).

2. Реализовать с использованием MS Excel перевод чисел в десятичную систему счисления, из десятичной системы счисления, сложение чисел в позиционных системах счисления (по методическим указаниям и индивидуальному заданию).

3. Создать отчет с использованием MS Word по выполненным индивидуальным заданиям. Отчет оформить согласно ГОСТ 7.32–2001 «Отчёт о научно-исследовательской работе. Структура и правила оформления».

4. Оформить презентацию с использованием MS PowerPoint по проделанной в п. 2.1 работе. Подготовить текст доклада на 5 мин. Продемонстрировать проект преподавателю и защитить работу.

Краткая теория и методические указания

Построение графика функции с использованием MS Excel, создание макроса для построения графика функции

Построение графиков функций – это, хотя и трудоемкая (при ручном выполнении), но в то же время весьма полезная математическая операция. Часто графики используются как наиболее простое и наглядное средство, позволяющее быстро выявить наиболее важные особенности исследуемых функций [10].

Построение графиков достаточно легко осуществляется с помощью MS Excel.

Создайте в вашей папке файл – приложение Excel – с именем «Графики функций» и откройте его. Присвойте двум листам документа имена: «График» и «Макрос построения графика».

Перейдите на лист «График». В верхней части листа наберите текст – заголовок листа: «Построение графиков».

Введите в ячейку **B3** текст «Нач. знач-е», в ячейку **C3** – «f(x)», в ячейку **D3** – «Кон. знач-е». В ячейки **B4**, **C4**, **D4** будем вводить исходные данные задачи: начало интервала, функцию и конец интервала соответственно. Выделите диапазон ячеек **B3:D4** и с помощью кнопки **Все границы** обозначьте рамки нашей «таблички» для начальных условий. Соседние с табличкой ячейки «залейте» серым цветом. Для придания таблице удобочитаемого вида выделите блок ячеек **B4:D4**, отцентрируйте текст и сделайте шрифт полужирным. Можно слегка расширить столбцы, если текст заголовков не помещается в ячейках.

В ячейку **B6** введите текст «шаг =». Выровняйте его по правому краю. В ячейке **C6** будет *вычисляться* шаг построения графика. Соседние с **B6** и **C6** ячейки также «залейте» серым цветом.

Подготовимся к табулированию функции. Для этого в ячейку **A8** введите текст «№ шага», в ячейку **B8** – «x», в ячейку **C8** – «f(x)». В столбцах **A**, **B** и **C** будут находиться, соответственно, номер строки таблицы, значение аргумента и значение функции в этой строке. Выделите диапазон ячеек **A8:C8** и с помощью кнопки **Все границы** сформируйте рамки введенного заголовка таблицы.

Лист готов к построению графика функции (рис. 1).

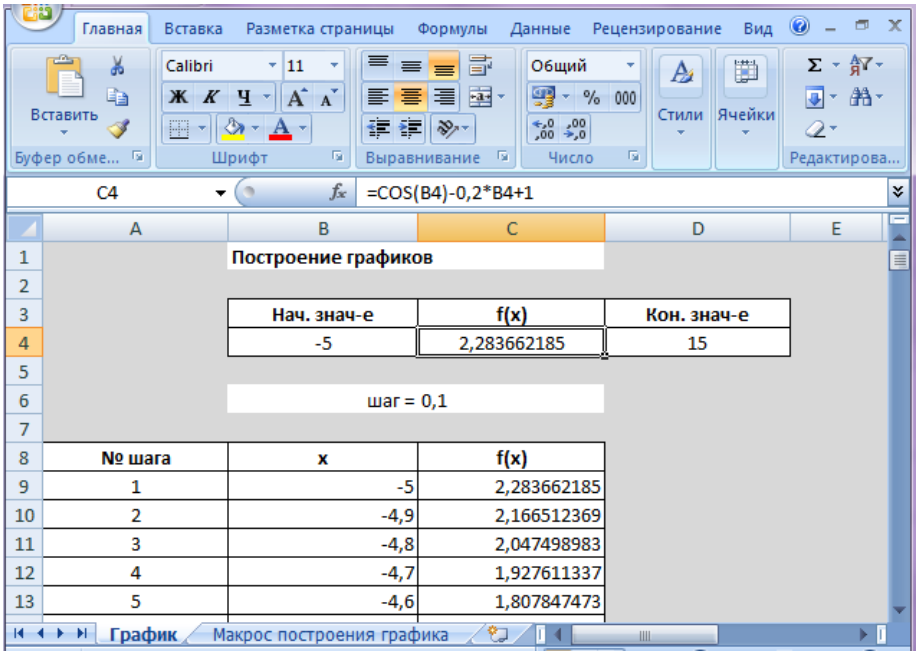


Рис. 1. Пример выполнения задания

Теперь решим следующую задачу: построим график функции $f(x) = \cos(x) - 0{,}2x + 1$ на интервале [–5; 15].

Внимание! Не вставляйте формулы в MS Excel путем копирования из методических указаний. Формулы необходимо вводить самостоятельно. Адреса

ячеек можно вводить в формулы без использования клавиатуры, щелкая по ним в нужный момент мышью.

Введем исходные данные задачи, а именно: в ячейку **B4** – значение –5, в ячейку **C4** – формулу **=cos(B4)–0,2*B4+1**, в ячейку **D4** – значение 15.

Для построения графика разобьем заданный интервал на 200 равных отрезков, длину которых занесем в ячейку **C6**. Введите в ячейку **C6** формулу **=(D4-B4)/200**. Проверьте в уме правильность полученного в **C6** результата.

Теперь можно начать табуляцию функции.

Введите в ячейку **A9** значение 1, в ячейку **B9** введите формулу **=B4**, а в ячейку **C9** скопируйте формулу из ячейки **C4**. Для этого выделите ячейку **C4** и с помощью команды меню **Правка/Копировать** скопируйте ее содержимое в буфер обмена. Далее выделите ячейку **C9** и воспользуйтесь командой меню **Правка/Вставить**.

Заполним весь столбец «№ шага». Выделите ячейку **A9**. Воспользуйтесь командой меню **Главная/Редактировать/Заполнить/Прогрессия…** (рис. 2).

В появившемся окне задайте следующие параметры:

– **расположение** по столбцам;
– **тип** арифметическая;
– **шаг** 1;
– предельное значение 201.

Нажмите **OK**.

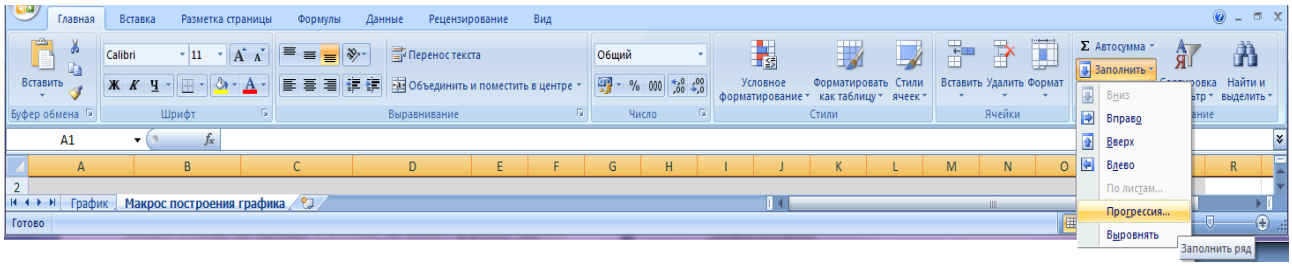


Рис. 2. Заполнение шага

Продолжим заполнение таблицы. Введите в ячейку **B10** формулу **=B9+$C$6**. В ячейку **C10** также можно ввести формулу **=cos(B10)-0,2*B10+1**, но проще ее просто скопировать из ячейки **C9**. Для этого выделите ячейку **C9**, наведите мышь на *маркер заполнения* – маленький квадрат в правом нижнем углу ячейки. После того как курсор превратится в черное перекрестие, нажмите левую кнопку мыши и, удерживая ее нажатой, протащите маркер вниз. Проанализируйте результат ваших действий: вместе с содержимым ячеек (формулами)

копируются их форматы. Получается не корректно. Позже, при создании макроса, учтем это и выберем другой способ копирования.

Используя методы копирования, сформируем оставшуюся часть таблицы. Выделите блок ячеек **B10**, **C10** и протащите его маркер заполнения вниз, пока не получите 201 значение (другой вариант – дважды щелкните по маркеру заполнения). Получив таким образом значения аргумента и функции во всех выбранных точках интервала, мы произвели табулирование функции.

Можно строить график функции. Укажите выделением диапазон ячеек **B9:C209**, по которому нужно построить график. Для этого выделите сначала ячейку **B9**, затем при нажатой комбинации клавиш Ctrl + Shift воспользуйтесь клавишами управления → и ↓.

Для построения графика воспользуемся командой меню **Вставка/Диаграмма…** или кнопкой **Мастер диаграмм** на панели **Стандартная**. В появившемся окне на вкладке **Стандартные** выберите тип **Точечная**, вид диаграммы – **Точечная диаграмма со значениями, соединенными сглаживающими линиями без маркеров**. Нажмите **Далее**. В открывшемся окне можно указать источник данных, но нужный нам диапазон ячеек уже предложен по умолчанию, так как мы его предварительно выделили. Поэтому переходим сразу к следующему шагу построения диаграммы. На вкладке **Заголовки** введите: Название диаграммы «f(x)=cos(x)-0,2*x+1». Введите также обозначение «Y» в микроокно **Ось Y** и обозначение «X» в микроокно **Ось X**. На вкладке **Легенда** уберите галочку из **Добавить легенду**. Нажмите **Готово**.

График построен (рис. 3). Переместите его вверх станицы.

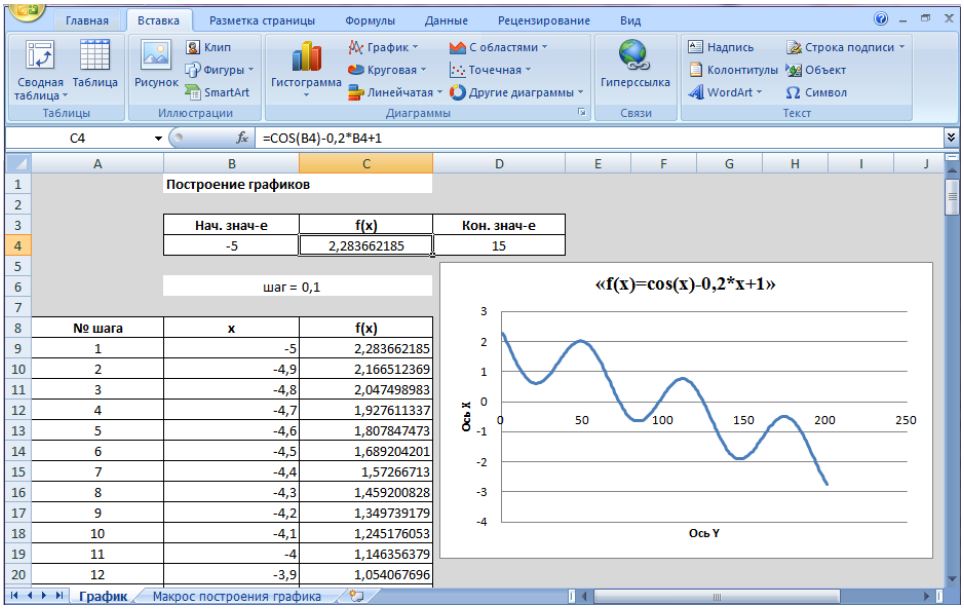


Рис. 3. График функции

Если требуется многократное решение типовой задачи в MS Excel, то ее решение можно автоматизировать с помощью *макроса*.

Макрос – это последовательность команд и функций, которая сохраняется нами под определенным именем [9, 10]. Ее можно вызывать по заданному имени или путем нажатия заданного сочетания клавиш всякий раз, когда необходимо выполнить данную типовую задачу. Например, если нам необходимо строить разные графики функций, то можно создать макрос для построения графика.

Создадим такой макрос.

Перейдите на лист «Макрос построения графика».

Для создания макроса нам нужно повторить некоторые действия, отработанные на предыдущем листе, и добавить указания о записи макроса.

Скопируйте диапазон ячеек **A1:D8** с листа «График» и вставьте его на лист «Макрос построения графика» в соответствующие ячейки. Если скопировался и заголовок листа, то перепишите его как-нибудь иначе, например так: «Макрос построения графика: Ctrl + q».

Далее для создания макроса следует воспользоваться командой меню **Вид/Макросы/Запись макроса…**. В появившемся окне введите имя макроса и укажите сочетание клавиш, при нажатии которых в будущем будет запускаться его выполнение, например Ctrl + q. Выберите **Сохранить в этой книге** и нажмите **OK**. С этого момента все выполняемые вами операции будут сохранены в виде команд макроса. Далее выполняйте то, что остается сделать для построения графика: табулируйте функцию и стройте график, как мы это делали в предыдущей части работы, начиная с фразы «Теперь можно начать табуляцию функции» и до слов «График построен». После этого остановите запись макроса с помощью команды меню **Сервис/Макрос/Остановить запись**.

При копировании формулы из ячейки **C9** в ячейку **C10** не пользуйтесь маркером заполнения, скопируйте ячейку **C9** в буфер, щелкните правой кнопкой мыши по ячейке **C10**, в появившемся контекстном меню выберите опции **Специальная вставка...** и затем – **Формулы**.

Макрос создан (рис. 4).

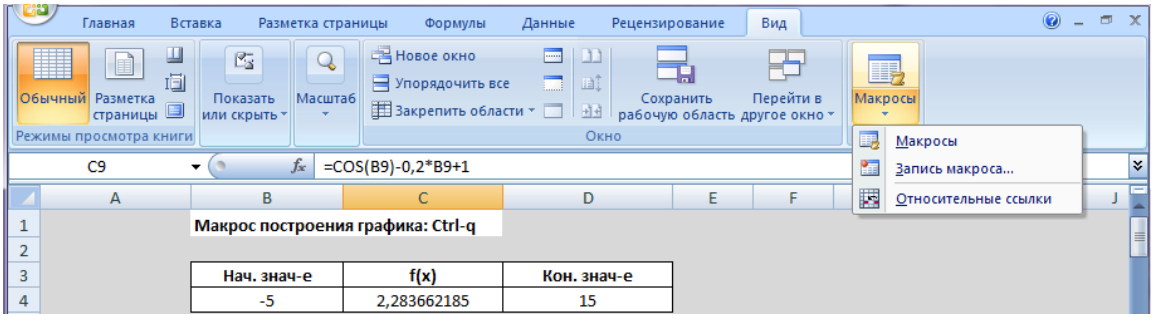


Рис. 4. Макрос построения графика

Варианты индивидуального задания

№ варианта	Формула	Ограничения
1	$\frac{x^2}{x-1}$	$1{,}5 \le x \le 3{,}5$
2	$1+\sin(x^2)$	$\frac{\pi}{4} \le x \le 5\frac{\pi}{4}$
3	$x \cdot \mathrm{arctg}(x)$	$\frac{\sqrt{3}}{3} \le x \le \sqrt{3}$
4	$\frac{x}{x^3+16}$	$-2 \le x \le 1$
5	$3 \cdot \sqrt{x}$	$4{,}3 \le x \le 6{,}8$
6	$x^{1/2} - x^{1/3}$	$0 \le x \le 4$
7	$\cos(x)$	$\frac{\pi}{2} \le x \le 2\pi$
8	$\frac{16x^2+2x+7}{x^2-8}+1$	$5 \le x \le 8$
9	$\ln(x)$	$0{,}5 \le x \le 2{,}5$
10	$\frac{(e^x-1)\cdot(e^{2x}+1)}{e^x}$	$5 \le x \le 8$
11	$\frac{x^2+5}{x^2+2}$	$0 \le x \le 2$
12	$\frac{x}{1+x^2}$	$0{,}5 \le x \le 2{,}5$
13	$x^2 \cdot e^{-x^2}$	$1 \le x \le 3$
14	$\frac{1}{x^2}$	$\frac{1}{\sqrt{6}} \le x \le \frac{2{,}5}{\sqrt{6}}$
15	$(x+\frac{1}{x})^2$	$4{,}3 \le x \le 6{,}8$
16	$\sin(x)$	$0 \le x \le \pi$
17	$\frac{1}{\sqrt{50-x^2}}$	$2 \le x \le 4$
18	$\frac{1}{\sin(\frac{\ln(\sqrt{5x-x^2})}{\cos(3x)})}$	$0{,}14 \le x \le 3{,}67$
19	$6x^2-3x+5$	$5 \le x \le 8$
20	$\frac{2x^2-3x+1}{x+1}$	$5 \le x \le 8$

Перевод чисел из одних систем счисления в другие с использованием MS Excel

Системой счисления (нумерацией) обычно называют способ записи чисел с помощью специальных знаков или цифр. Так, общепринятая десятичная система использует 10 цифр: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. В вычислительных машинах чаще применяется двоичная система, использующая только две цифры: 0 и 1.

Все системы счисления принято делить на две группы: позиционные системы счисления и непозиционные системы счисления. В позиционной системе значение каждой цифры зависит от места (позиции), которое она занимает в последовательности цифр, обозначающей число. В непозиционной системе значение знака (цифры или какого-либо другого символа) не находится в зависимости от его места в записи числа [9, 10].

Для выполнения этой работы создайте в вашей папке файл – приложение Excel – с именем «Системы счисления» и откройте его. Дайте трем листам имена: «В десятичную», «Из десятичной» и «Сложение».

Все три листа нужно сделать «в клеточку», т. е. задать ширину их столбцов примерно равной ширине строк. Это можно сделать одновременно для всех трех листов следующим образом.

На любом (например, первом) листе щелкните правой кнопкой мыши по ярлычку и выберите в открывшемся контекстном меню «Выделить все листы». Ярлычки всех листов станут «активными», белыми. Теперь щелкните кнопку выделения всего листа (серый прямоугольник в левом верхнем углу листа на пересечении заголовков строк и столбцов). Весь лист станет выделенным. Установите курсор между заголовками любых из двух столбцов и левой кнопкой мыши «стяните» ширину столбца до 2.00 (19 пикселей). После этого щелкните любую ячейку, чтобы закончить операцию. Все столбцы сделались одной ширины, и ячейки приобрели форму квадратных клеточек. То же самое (проверьте) произошло с другими двумя листами. Книга подготовлена к работе. Перейдем к листу «В десятичную».

На листе «В десятичную» создадим «машину» для перевода в десятичную систему счисления чисел, записываемых в системах с произвольным основанием $B > 1$.

С этой целью разметим предварительно лист так, как показано на рис. 5.

В верхней части листа наберите текст – заголовок листа: «Перевод чисел из системы счисления с основанием B в десятичную систему счисления». Растяните в ширину столбец **B**, в котором будем записывать основание системы счисления (в ячейку **B8**) и получать десятичное число – результат перевода (ячейка **B15**). С помощью команды – кнопки меню **Внешние границы** – разметьте диапазоны для записи цифр целой части числа (**D8:W8**) и дробной части

числа (**Y8:AR8**). Сделайте необходимые текстовые заголовки-пояснения к этим размеченным диапазонам ячеек. Соседние с этими диапазонами ячейки можно залить серым цветом для улучшения дизайна нашей «машины» (рис. 5). Если вся «машина» не помещается на экране, то можно слегка уменьшить масштаб листа.

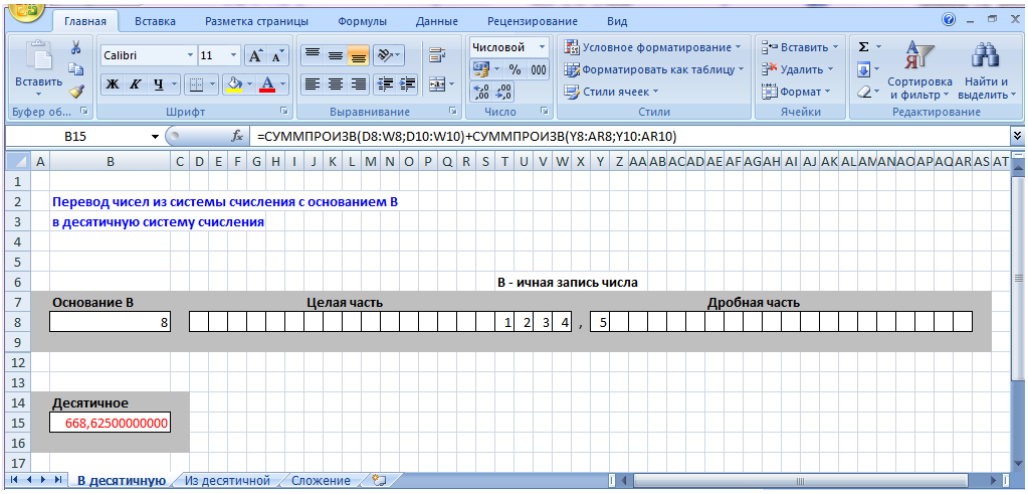


Рис. 5. Примерный вид экрана с листом «В десятичную»

Строка **8** листа готова для записи в ячейку **B8** *основания* системы счисления и для записи в ячейке **D8:AR8** *цифр* числа, представленного в этой системе счисления. При этом как целая, так и дробная части числа могут содержать до 20 цифр. Остается сделать так, чтобы в ячейке **B15** появлялось *десятичное* число, записанное в строке **8** в системе счисления с основанием *B*. Для этого нужно лишь правильно ввести в ячейки несколько простых формул.

Во всех этих формулах нам, разумеется, нужно будет ссылаться на одну и ту же ячейку **B8.** Для того чтобы при копировании формул ссылка на эту ячейку не изменялась, ссылку нужно писать как абсолютную, т. е. в виде **$B$8**. Однако наличие ссылок такого вида в формулах делает их не очень удобными для чтения. Поэтому воспользуемся другим механизмом присвоения абсолютного имени ячейкам и диапазонам. Чтобы присвоить ячейке **B8** имя, например, *a*, выделим ее левой кнопкой мыши. Взгляните на *поле имени* – оно расположено на верхней панели инструментов, слева от строки формул. Введите в *поле имени* букву *a* и нажмите Enter. Имя, присвоенное сейчас выделенной ячейке, является ее абсолютным адресом, уникальным в пределах всей книги. Если, например, сейчас вы перейдете на другой лист и в поле имени выберете *из списка* имя *a*, то тут же окажетесь вновь на листе «В десятичную», где будет выделена ячейка **B8**. Итак, основание системы счисления будет именоваться на листе как *a*, и в формулах вместо **$B$8** мы будем писать букву *a*.

Введите в ячейку **B8** в качестве основания системы счисления число 3. Для перевода чисел в десятичную систему зададим веса разрядов. В ячейку **W10** (т. е. под младшим разрядом целой части) введите 1 – это *вес младшего разряда*. В соседнюю слева ячейку введите формулу **=W10*a**. В ней появится число 3. Скопируйте эту ячейку с формулой ее «растягиванием» за маркер заполнения влево, до старшего разряда. Теперь мы имеем все *веса целой части* числа. Многие их значения не входят в клеточки и выводятся как символ решетки. Но нам и не нужно их видеть. Скоро мы скроем всю эту строку.

Аналогично задайте *веса для дробной части* числа: в клетку **Y10** введите формулу **=1/a**. Результат из-за его округления до одной цифры будет выглядеть как 0. В соседнюю справа ячейку введите формулу **=Y10/a**. Затем «растяните», т. е. скопируйте ее до младшего разряда дробной части. Теперь в ячейку **B15** введите формулу

=СУММПРОИЗВ(D8:W8;D10:W10)+СУММПРОИЗВ(Y8:AR8;Y10:AR10).

Программа для перевода чисел в десятичную систему счисления готова. Формулу для вычисления суммы произведений цифр разрядов на их веса пришлось разбить на две части, так как в колонке **X** между двумя диапазонами разрядов мы разместили запятую. Если запятую оттуда удалить, то формулу в ячейке **B15** можно записать сразу для двух сплошных диапазонов в виде

=СУММПРОИЗВ(D8:AR8;D10:AR10).

Запишите в ячейки **W8** и **Y8** цифры 1 и 1. Остальные разряды троичного числа пусть будут нули (или пустые ячейки – это то же самое). Таким образом, мы ввели троичное число $1{,}1_3$. В ячейке **B15** появится десятичное изображение 1,3333… этого троичного числа. Увеличьте ширину столбца B, как на рис. 5 или еще больше.

Выделите все ячейки с *входными данными*: для этого щелкните ячейку **B8** и при нажатой клавише Ctrl выделите мышкой диапазон **D8:AR8**. После выделения войдите в меню **Формат/Ячейки...**, в открывшемся окне на вкладке **Защита** снимите флажок **Защищаемая ячейка** и подтвердите это действие нажатием на кнопку **ОК**. Теперь выделенные нами ячейки останутся доступными для ввода данных после защиты листа. Затем выделите строки **10** и **11**, щелкните по ним правой клавишей мыши и выберите из контекстного меню **Скрыть**. Чтобы защитить лист, войдите в меню **Сервис/Защита/Защитить лист** и нажмите **ОК**, не задавая пароль. Снимать защиту листа можно будет без пароля, через те же пункты меню **Сервис/Защита**.

Так мы застраховались от случайного изменения всех ячеек, кроме тех, где нужно будет вводить данные при решении задач. Попробуйте выполнить какие-нибудь непредусмотренные изменения, например отобразить скрытые строки или изменить формулы. Необходимые для этого действия окажутся недоступны. Можно приступать к экспериментам. Сохраните файл, но не закрывайте его.

Первый проверочный опыт можно провести сейчас. Запишите в строке **8** в качестве основания исходной системы счисления число 10. Запишите в этой же строке цифры целой и дробной частей числа 1234,5. Эта запись автоматически преобразуется и отобразится в ячейке **B15** в виде десятичного числа 1234,5. Теперь в ячейке **B8** запишите 8. При *таком* основании исходной системы счисления число, записанное в строке **8**, отобразится в виде десятичного числа 668,625 (рис. 5). «Машина» работает правильно.

Перейдите на лист «Из десятичной».

Чтобы построить «машину» для перевода десятичных чисел в системы счисления с любыми основаниями, разметьте ячейки: ячейку **K7** – для ввода основания системы счисления *B*, ячейку **B10** – для ввода целой части *N* исходного десятичного числа и ячейку **B18** – для ввода его дробной части *Z* (рис. 6).

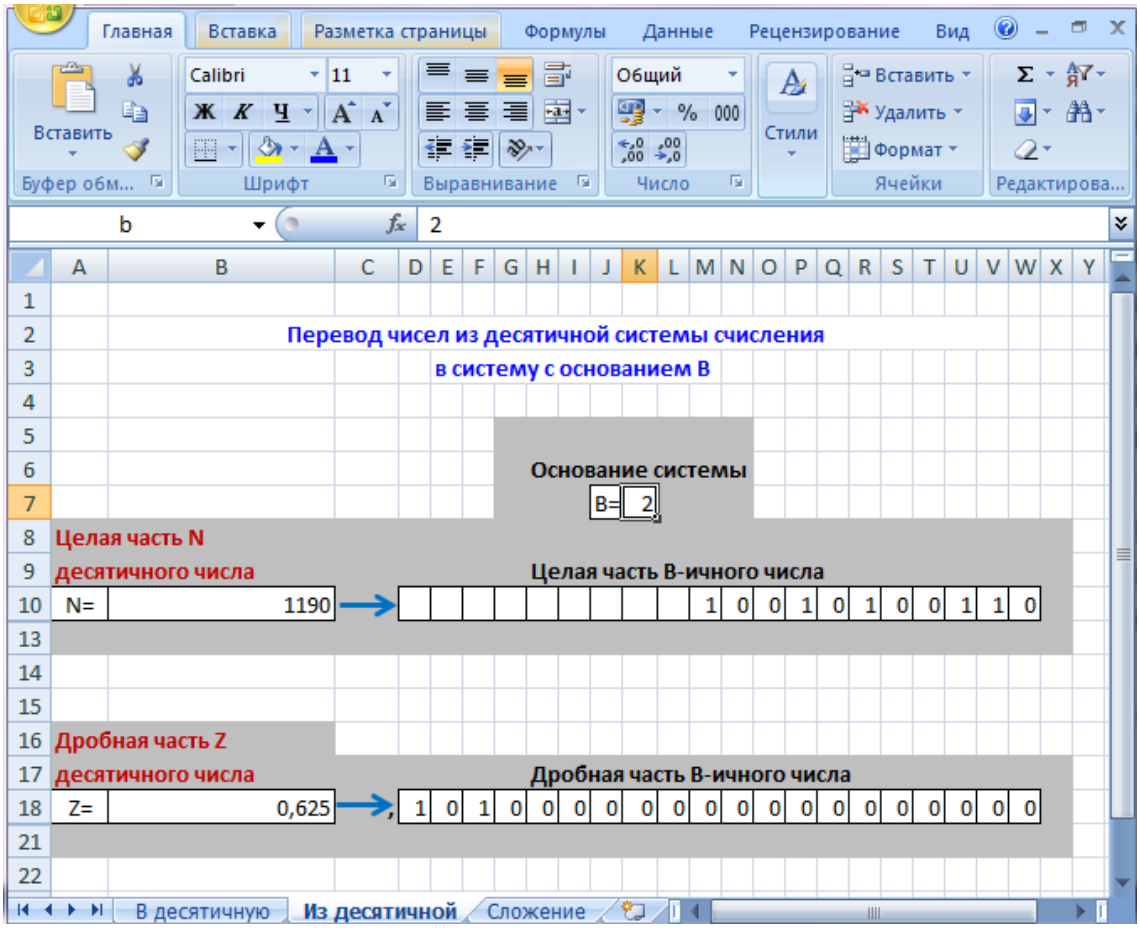


Рис. 6. Примерная организация листа «Из десятичной»

Присвойте выбранным ячейкам имена *b*, *N* и *Z*, используя окно имен. Обратите внимание: на рис. 6 выделена ячейка **K7** с записанным в ней числом 2, а в окне имен (вверху слева) высвечено имя *b* этой ячейки. Кроме того, строки **11**, **12** и **19**, **20** (с промежуточными результатами) на листе уже скрыты, нам же эти строки пока еще нужны для работы.

Увеличьте ширину столбца **B**.

Сделайте с помощью форматирования границ разметку диапазонов **D10:W10** и **D18:W18** для отображения цифр переведенного числа. Позднее можно будет поработать и над дизайном листа в целом. В ячейки для записи чисел *B*, *N* и *Z* запишите 2, 1190 и 0,625 соответственно; сейчас мы начнем перевод в двоичную систему счисления десятичного числа 1190,625.

В строках 11 и 12 разместим формулы для пересчета целой части *N* десятичного числа в заданную систему счисления с любым основанием *B*, реализуя правило последовательного деления *N* на *B* и вычисления остатков. Формулы для определения *частных* от деления разместим в диапазоне ячеек **D11:W11**, под разрядами *B*-ичного числа. Строкой ниже введем формулы для вычисления последовательных *остатков* от деления *N* на *B*. Эти остатки и будут цифрами *B*-ичного числа.

Итак, введем в ячейку **W11**, под младшим разрядом, формулу **=ЦЕЛОЕ(N/b)**. Поскольку результат деления – частное – не помещается в ячейку, то в ней выводится символ «решетка». В соседнюю слева ячейку **V11** введем формулу **= ЦЕЛОЕ(W11/b)**. Скопируем эту ячейку влево до старшего разряда, т. е. на весь диапазон **D11:V11**.

В ячейку **W12** запишем формулу **=ОСТАТ(N;b)**, по которой определяется младшая цифра числа. В соседнюю слева ячейку **V12** запишем **=ОСТАТ(W11;b)**, так как далее мы должны получать остатки от деления каждого предыдущего частного на b. Скопируем эту формулу влево, до старшего разряда, т. е. в диапазон **D12:V12**.

Мы видим в строке **12** все цифры *B*-ичного (в данном случае – двоичного) числа, и на этом можно было бы остановиться. Однако потребуем еще, чтобы незначащие нули в старших разрядах не выводились. Тогда переносить полученные цифры в приготовленную для них «разрядную сетку» **D10:W10** нужно с помощью условной функции «ЕСЛИ».

Для этого в клетку **W10** введем формулу **=W12**. Младший разряд переносится в отображаемые цифры безусловно. Левее, в клетку **V10**, вводим формулу **=ЕСЛИ(СУММ($D$12:V12)=0;"";V12)**. Смысл этой формулы таков: если сле-

ва от разряда **V12** (включая и сам этот разряд) *все нули*, то в ячейку записывается результат «пусто» (пустой текст между апострофами), иначе пишется *цифра* из ячейки **V12**. Ячейку **V10** с введенной формулой копируем влево до конца разрядной сетки. Если все сделано правильно, то результат перевода соответствует рис. 6.

Дробная часть *Z* десятичного числа переводится аналогично, только при этом применяется не деление, а умножение *Z* на основание системы счисления. После каждого умножения целая часть результата забирается (вычитается) из него и переносится в качестве очередной цифры в состав дробной части перевода числа. При этом цифры дробной части появляются в порядке слева направо.

В строках **19** и **20** для такого перевода выполните следующие действия:

– в ячейку **D19** введите **=b*Z**;

– в ячейку **D18** введите **=ЦЕЛОЕ(D19)** – это первая цифра дробной части;

– в **D20** введите формулу **=D19-D18** – вычитаем целую часть из результата;

– в **E19** введите формулу **=b*D20** и скопируйте эту ячейку вправо до **W19**;

– скопируйте ячейку **D18** вправо вплоть до **W18**;

– скопируйте ячейку **D20** вправо вплоть до **W20**.

Если все сделано правильно, то цифры дробной части будут такими, как на рис. 6.

Теперь скройте строки **11**, **12** и **19**, **20**. Выделите ячейки с исходными данными (**K7**, **B10**, **B18**), через меню формата ячеек снимите с них флажок «защищаемая ячейка» и защитите лист, не задавая пароля. Лист готов к экспериментам и решению задач. Сохраните файл, «машина» работает правильно.

Устройство листа «Сложение» наиболее простое в данной работе. Перейдите на этот лист. Разметив ячейки и диапазоны для ввода основания системы счисления, ввода цифр слагаемых и отображения суммы слагаемых (рис. 7), введите в диапазон ячеек **D12:AC12** формулы для вычисления *переносов*. Для этого введите в ячейку B8 число 8, а в ячейку **AC12** – формулу **=ЦЕЛОЕ((AC9+AC8+AD12)/$B$8)**. Смысл формулы в том, что когда сумма двух разрядов и переноса из предыдущего разряда будет больше основания системы счисления, то сформируется перенос в следующий разряд. Скопируйте ячейку **AC12** с формулой влево до ячейки **D12**.

Аналогично можно задать формулы для вычисления разрядов суммы. В ячейку **AC10** занесите формулу **=ОСТАТ(AD12+AC8+AC9;$B$8)** – это часть той же суммы, остающаяся в данном разряде. Скопируйте эту ячейку влево до конца разрядной сетки.

Скройте строку **12**. Введите цифры семеричных слагаемых, как показано на рис. 7. Если вы не ошибались, то получится и соответствующая этому рисунку сумма.

На рис. 7 видно, что группа разрядов чисел «разделена» на листе на две части стрелками. Так можно условно отмечать положение разделительной запятой, когда нужно интерпретировать суммирование как операцию над дробными числами. На рис. 7 сложены дробные числа из последнего варианта индивидуального задания к данной работе.

Выделите ячейку **B8** и, прижимая клавишу Ctrl, выделите одновременно с ячейкой диапазон **D8:AC9**. Снимите через меню формата флажок защиты выделенных ячеек, так как эти ячейки нужно оставить доступными для ввода данных после защиты всего листа. Защитите лист и сохраните файл.

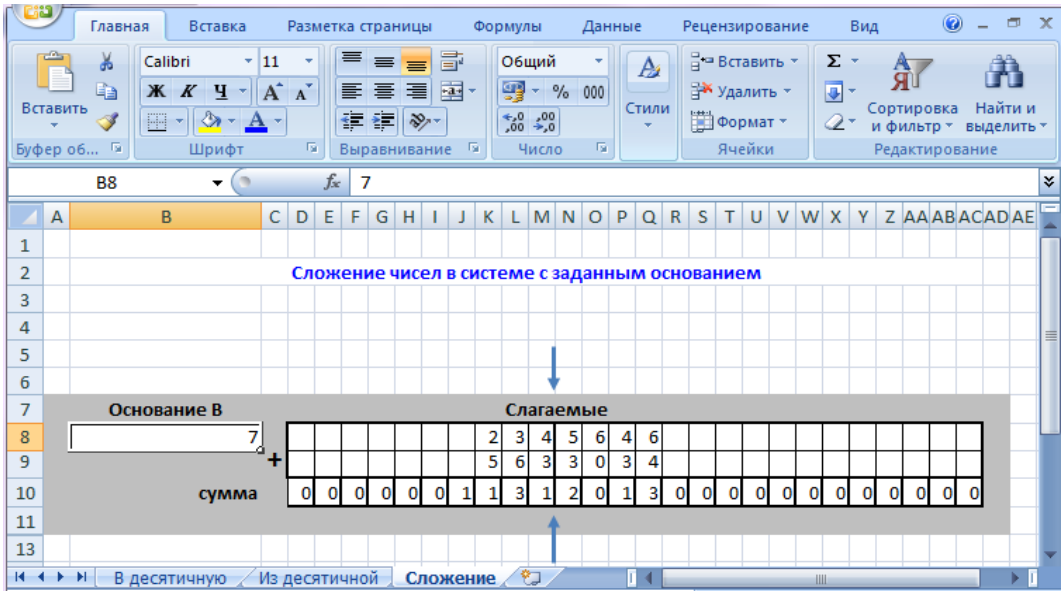


Рис. 7. Примерный вид листа «Сложение»

Далее осуществите перевод чисел согласно вашему варианту задания. Результаты перевода необходимо внести в табл. 1.

Таблица 1

Результаты перевода чисел

Число X с основанием B исходной системы	Число X в десятичной системе счисления	Число X с основанием C дублирующей системы
Число Y с основанием B исходной системы	Число Y в десятичной системе счисления	Число Y с основанием C дублирующей системы
Сумма X_B и Y_B с основанием B исходной системы	**Сумма X и Y в десятичной системе счисления**	Сумма X_C и Y_C с основанием C дублирующей системы
Сумма X_B и Y_B в десятичной системе счисления	–	**Сумма X_C и Y_C в десятичной системе счисления**

Варианты индивидуального задания

№ варианта	Основание *B* исходной системы	Число X_B	Число Y_B	Основание *C* дублирующей системы
1	2	100,0001	100,0111	7
2	2	010,0110	110,1001	9
3	6	432,1414	042,5220	2
4	2	110,0010	111,0110	9
5	9	315,5087	111,5834	2
6	9	073,4821	632,4888	3
7	3	020,1222	221,2020	5
8	16	B42,AB0B	65F,8903	2
9	16	8AB,3F7F	5DB,02D2	4
10	8	452,0632	123,0016	6
11	16	30D,4A89	AAD,DC1E	7
12	2	100,1010	010,0010	7
13	6	300,0414	545,2315	16
14	9	772,4022	706,1470	4
15	8	141,0246	435,3044	2
16	9	855,8063	661,1871	7
17	7	431,0632	222,1003	4
18	6	125,3320	431,1235	5
19	6	012,4012	001,4020	9
20	7	234,5646	563,3034	2

Создание отчетов с использованием MS Word

После выполнения всех расчетов студенту необходимо создать отчет с использованием MS Word по выполненным индивидуальным заданиям.

Все отчеты, выполняемые в процессе обучения, студенты кафедры «Автоматизированные системы обработки информации и управления» должны оформлять согласно ГОСТ 7.32–2001 «Отчёт о научно-исследовательской работе. Структура и правила оформления» [7].

Ниже приведены лишь основные требования для составления отчета.

Отчет должен быть выполнен в электронном виде на стандартных листах формата А4.

Отчет необходимо оформлять с соблюдением следующих размеров полей: верхнее – 20 мм; левое – 20 мм; нижнее – 20 мм; правое – 10 мм.

Шрифт, используемый в отчете – Times New Roman. Цвет шрифта должен быть черным, высота букв, цифр и других значков – 12 или 14 пт. Разрешается уменьшать шрифт до 12 пт в таблицах. Также разрешается использовать компьютерные возможности акцентирования внимания на определенных терминах, формулах, теоремах, применяя шрифты разной гарнитуры.

Абзацный отступ должен быть равен 1 см.

Межстрочный интервал по всему документу – 1,2–1,3.

Все формулы, встречаемые в отчете, оформляются через «Редактор формул». Формулы располагаются с красной строки. Формулы следует нумеровать арабскими цифрами в пределах всего текста в круглых скобках в крайнем правом положении на строке, например:

$$f(x) = \cos(x) - 0{,}2x + 1. \qquad (1)$$

Рисунки и подрисуночные подписи размещаются по центру страницы, без абзацного отступа. Нумерация рисунков – сквозная. Ссылки по тексту документа на рисунки обязательны, они имеют вид: (... на рис. 3 изображено...).

Таблицы, как и рисунки, следует располагать непосредственно после текста, в котором впервые упоминается о ней. Каждая таблица должна иметь порядковый номер, изображаемый арабскими цифрами, нумерация должна быть сквозной в пределах всего документа. Название таблицы следует помещать слева над ней без красной строки, с её номером через тире (без точки в конце названия). Ссылка на таблицу по тексту документа также обязательна, как и у рисунков.

Структура отчета по дисциплине «Рабочая профессия» и порядок расположения структурных элементов отчета следующий.

Титульный лист (образец оформления приведен в прил. А)

Реферат (образец оформления приведен в прил. Б)

Обозначения, определения и сокращения (см. прил. Б)

Содержание

Введение

1 Создание макроса для построения графика функции

1.1 Задание по варианту

1.2 Описание хода выполненной работы

2 Перевод систем счисления

2.1 Задание по варианту

2.2 Описание хода выполненной работы

3 Индивидуальная тема

Заключение

Список использованных источников

Приложение

Структурные элементы, расположенные ниже «Содержания», подлежат отображению при создании автоматического оглавления, и их необходимо задавать заголовками в самой пояснительной записке.

Структурные элементы «Введение», «Заключение», «Список использованных источников», «Приложение» – это заголовки первого уровня.

Главы 1, 2, 3 – заголовки второго уровня.

Подглавы 1.1, 1.2, 2.1 и 2.2 – заголовки третьего уровня.

В конце заголовка или после номера у главы – точки не ставят.

Структурные элементы, расположенные выше «Содержания», в автоматически созданном оглавлении отображаться не должны.

Структурные элементы, подлежащие отображению, размещаются в «Содержании» с указанием номеров их начальных страниц.

Образец структуры «Содержания» приведен на рис. 8.

СОДЕРЖАНИЕ

Рис. 8. Образец структуры содержания

Заголовки первого уровня выравниваются по центру страницы, не нумеруются и не имеют красной строки, например:

Введение

Заключение

Список использованных источников

Заголовки второго и третьего уровней размещаются по левому краю с абзацным отступом 1 см. При необходимости их можно выделять жирным шрифтом, курсивом или использовать все прописные буквы, например:

1 СОЗДАНИЕ МАКРОСА ДЛЯ ПОСТРОЕНИЯ ГРАФИКА ФУНКЦИИ

1.1 Задание по варианту

1.2 Описание хода выполненной работы

2 ПЕРЕВОД СИСТЕМ СЧИСЛЕНИЯ

2.1 Задание по варианту

2.2 Описание хода выполненной работы

Структурные элементы – заголовки первого и второго уровней – должны начинаться с нового листа.

Обозначения, определения и сокращения содержат уточнения терминов, перечень обозначений и сокращений, применяемых в тексте. Их запись приводят в алфавитном порядке (порядок алфавитов – русский, английский) с необходимой расшифровкой и пояснениями. Не допускается сокращение слов в тексте отчета, кроме общепринятых сокращений.

Список использованных источников оформляется в соответствии с требованиями ГОСТ 7.1–2003 [6].

Материал, дополняющий текст отчета, необходимо помещать в приложениях. Приложения обозначают заглавными буквами русского алфавита.

Страницы нумеруются арабскими цифрами в центре нижней части листа без точки. Нумерация страниц в текстовом документе должна быть сквозной: первой страницей является титульный лист, второй – реферат, определения, обозначения и сокращения, содержание и т. д. На титульном листе, реферате, определениях, обозначениях и сокращениях, содержании – номер страницы не ставят (все листы, начиная с титульного, считаются, а номер страницы проставляется на листах после содержания).

Третья глава отчета у каждого студента оформляется по индивидуальной тематике. Название третьей главы в отчете соответствует названию темы.

Варианты индивидуальных тем для отчета

№ варианта	Тема
1	Автоматизированные системы научных исследований
2	Архиваторы
3	Детекторы плагиата
4	Добыча данных (Data Minig)
5	Информация как объект права
6	Информация как объект собственности
7	Каналы связи, их основные виды и характеристики
8	Клеточные автоматы
9	Компьютерные вирусы и антивирусные программы
10	Микроминиатюризация в электронике
11	Мультимедийная платформа Adobe Flash
12	Параллельное программирование
13	Понятие информационного ресурса в информатике
14	Распознавание почерка
15	Сетевые технологии
16	Системы счисления в остаточных классах
17	Технологии будущего
18	Трансляторы
19	Управление по принципу обратной связи
20	Фракталы

Оформление презентации с использованием MS PowerPoint

Доклад – это публикация вашей работы. Опыт подготовки доклада окажется вам полезным на протяжении всего учебного процесса. Этот навык будет развиваться и совершенствоваться во время вашего участия в студенческих научных конференциях, при подготовке и защите курсовых и выпускной квалификационной работ [8, 10].

Подготовка к докладу начинается с написания текста доклада (выступления). Ваш доклад – это реферативное сообщение, в котором нужно кратко раскрыть теоретическую сущность и прикладное значение изученного вопроса. Такой доклад можно рассматривать как короткую лекцию по заданной теме. Поэтому рекомендуется придерживаться достаточно простого и ясного плана доклада, который включает следующие части.

1. Тема доклада. Постановка задачи, ее научное и практическое значение.
2. Современное состояние вопроса.

3. Цель доклада.

4. Источники информации, привлеченные для изучения вопроса.

5. Основная, содержательная часть сообщения по теме доклада.

6. Заключение и выводы.

7. Личный вклад автора.

Доклады должны быть подготовлены с применением программы для представления презентаций MS PowerPoint с включением различных видов настройки анимации.

Первая страница презентации должна содержать название доклада, название вуза, факультета и кафедры, ФИО автора и ФИО руководителя.

Как и весь доклад в целом, презентация должна иметь четкую структуру, внутренние озаглавленные разделы (подразделы).

Иллюстрации к докладу (слайды) во время выступления служат вашим путеводителем. Они должны быть яркими, лаконичными и легко воспринимаемыми, количество надписей на них – минимальным.

На слайде необходимо поместить только самое необходимое. Не пишите длинных подрисуночных подписей и определений, пользуйтесь общеизвестными сокращениями. Помните, что картинка показывается на экране короткое время и восприятие помещенной на ней информации должно быть быстрым.

Особое внимание уделите заключению, в котором приводятся выводы. Обычно оно содержит заключения сразу двух видов.

Заключение первого вида – *констатирующего* – служит логическим замыканием постановочной части доклада. В нем вы показываете, что поставленные задачи решены и цель работы достигнута. Здесь можно подчеркнуть те особенности методики вашей работы, которые обеспечили ее успех и позволили получить новые (для вас) знания.

Заключение второго вида – *результативно-аналитическое*. В нем вы перечисляете и комментируете результаты работы, их научное и практическое значение, оцениваете важность полученных сведений для вашей учебной специальности. Отдельные фразы из предыдущих разделов доклада в заключении можно повторять дословно.

Приведите в конце текста доклада и в конце презентации список литературы (библиографию), включая ссылки на использованные электронные ресурсы.

Последний этап работы над текстом доклада состоит в том, чтобы «выкинуть» все лишние слова, повторения, упростить длинные фразы, расставить знаки препинания.

Во время выступления не превышайте установленного для доклада отрезка времени. Всегда строго соблюдайте регламент. Ничто так не раздражает, как докладчик, не соблюдающий регламент. Как правило, ведущий занятие преподаватель по истечении отведенного на доклад времени или предлагает одну-две минуты для его завершения, или сразу останавливает доклад и переходит к следующим вопросам.

Выступление рекомендуется завершать выводами: *«следовательно...», «таким образом...».* Заключительная фраза выступления почти стандартна: *«Мой доклад закончен, благодарю за внимание».*

Контрольные вопросы к разделу 2.1

1. Использование формул, функций и диаграмм в MS Excel. Возможные ошибки при использовании функций в формулах.

2. Системы счисления. Перевод чисел и техника перевода.

3. Основные требования к оформлению отчетов.

4. Базовые возможности MS Word (создание оглавления, выбор стилей заголовков, создание формул, вставка таблиц, рисунков, диаграмм, нумерованных и маркированных списков).

5. Базовые возможности редактора презентаций. Процесс и виды настройки анимации в MS PowerPoint.

2.2. Системное программное обеспечение

Цель работы: приобретение студентами практических навыков и знаний по основным видам операционных систем, их установке, мобильности программного обеспечения, современных средств трассировки и отладки программ, по мониторингу показателей производительности оборудования.

Задание

1. Установить виртуальную машину на свой компьютер.

2. Выполнить индивидуальное задание на виртуальной машине.

3. Создать отчет с пошаговым описанием и приведением картинок по установке виртуальной машины и выполнению индивидуального задания.

4. Продемонстрировать проект преподавателю и защитить работу.

Краткая теория и методические указания

Системное программное обеспечение (СПО) – комплекс программ, которые обеспечивают управление компонентами компьютерной системы, такими как процессор, оперативная память, устройства ввода-вывода, сетевое оборудование. СПО выступает как «межслойный интерфейс», с одной стороны которого аппаратура, а с другой – приложения пользователя [2, 18].

В отличие от прикладного программного обеспечения системное не решает конкретные практические задачи, а лишь обеспечивает работу других программ, предоставляя им сервисные функции, абстрагирующие детали аппаратной и микропрограммной реализации вычислительной системы, управляет аппаратными ресурсами вычислительной системы.

Отнесение того или иного программного обеспечения к системному условно и зависит от соглашений, используемых в конкретном контексте. Как правило, к системному программному обеспечению относятся операционные системы, утилиты, системы программирования, системы управления базами данных, широкий класс связующего программного обеспечения.

Операционная система – комплекс системных программ, расширяющий возможности вычислительной системы, а также обеспечивающий управление её ресурсами, загрузку и выполнение прикладных программ, взаимодействие с пользователями. В большинстве вычислительных систем операционные системы являются основной, наиболее важной (а иногда единственной) частью системного программного обеспечения.

Основные функции операционных систем перечислены ниже.

– Исполнение запросов программ (ввод и вывод данных, запуск и остановка других программ, выделение и освобождение дополнительной памяти и др.).

– Загрузка программ в оперативную память и их выполнение.

– Стандартизованный доступ к периферийным устройствам (устройства ввода-вывода).

– Управление оперативной памятью (распределение между процессами, организация виртуальной памяти).

– Управление доступом к данным на энергонезависимых носителях (таких как жёсткий диск, оптические диски и др.), организованным в той или иной файловой системе.

– Обеспечение пользовательского интерфейса.

– Сохранение информации об ошибках системы.

К дополнительным функциям относят параллельное или псевдопараллельное выполнение задач (многозадачность), эффективное распределение ресурсов вычислительной системы между процессами, разграничение доступа различных процессов к ресурсам, организация надёжных вычислений, взаимодействие между процессами: обмен данными, взаимная синхронизация, защита самой системы, а также пользовательских данных и программ от действий пользователей (злонамеренных или по незнанию) или приложений, многопользовательский режим работы и разграничение прав доступа.

Существуют две группы определений операционных систем: «совокупность программ, управляющих оборудованием» и «совокупность программ, управляющих другими программами».

Есть приложения вычислительной техники, для которых операционные системы излишни. Например, встроенные микрокомпьютеры содержатся сегодня во многих бытовых приборах, автомобилях, сотовых телефонах и т. п. Зачастую такой компьютер постоянно исполняет лишь одну программу, запускающуюся при его включении. И простые игровые приставки, также представляющие собой специализированные микрокомпьютеры, могут обходиться без операционной системы, запуская при включении программу, записанную на вставленном в устройство «картридже» или компакт-диске. Тем не менее, некоторые микрокомпьютеры и игровые приставки всё же работают под управлением особых собственных операционных систем. В большинстве случаев это UNIX-подобные системы.

Встроенные программы или firmware – это программы, «зашитые» в цифровые электронные устройства. В ряде случаев (например, BIOS IBM-PC совместимых компьютеров) являются по сути частью операционной системы, хранящейся в постоянной памяти. В достаточно простых устройствах вся операционная система может быть встроенной. Многие устройства современных компьютеров имеют собственные «прошивки», осуществляющие управление этими устройствами и упрощающие взаимодействие с ними.

Утилиты – программы, предназначенные для решения узкого круга вспомогательных задач. Иногда утилиты относят к классу сервисного программного обеспечения.

Утилиты используются для мониторинга показателей датчиков и производительности оборудования (например, мониторинга температур процессора или ви-

деоадаптера), управления параметрами оборудования (ограничение максимальной скорости вращения CD-привода; изменение скорости вращения вентиляторов), контроля показателей (проверка ссылочной целостности; правильности записи данных), расширения возможностей (форматирование или переразметка диска с сохранением данных, удаление без возможности восстановления).

Виды утилит следующие: диспетчеры файлов, архиваторы (с возможным сжатием данных), просмотрщики, утилиты для диагностики аппаратного или программного обеспечения, утилиты восстановления после сбоев, оптимизатор диска (вид утилиты для оптимизации размещения файлов на дисковом накопителе, например, путём дефрагментации диска), утилиты – шредеры файлов, деинсталлятор – программа для удаления программного обеспечения, утилиты управления процессами.

Системы программирования – к этой категории относятся системные программы, предназначенные для разработки программного обеспечения:

– ассемблеры – компьютерные программы, осуществляющие преобразование программы в форме исходного текста на языке ассемблера в машинные команды в виде объектного кода;

– трансляторы – программы или технические средства, выполняющие трансляцию программы;

– компиляторы – программы, переводящие текст программы на языке высокого уровня в эквивалентную программу на машинном языке;

– интерпретаторы – программы (иногда аппаратные средства), анализирующие команды или операторы программы и тут же выполняющие их;

– компоновщики (редакторы связей) – программы, которые производят компоновку: принимают на вход один или несколько объектных модулей и собирают по ним исполнимый модуль;

– препроцессоры исходных текстов – это компьютерные программы, принимающие данные на входе и выдающие данные, предназначенные для входа другой программы, например такой как компилятор;

– отладчики – модули среды разработки или отдельные программы, предназначенные для поиска ошибок в программах;

– текстовые редакторы – компьютерные программы, предназначенные для создания и изменения текстовых файлов, а также их просмотра на экране, вывода на печать, поиска фрагментов текста и т. п.;

– специализированные редакторы исходных текстов – текстовые редакторы для создания и редактирования исходного кода программ. Специализированный редактор исходных текстов может быть отдельным приложением или быть встроен в интегрированную среду разработки;

– библиотеки подпрограмм – сборники подпрограмм или объектов, используемых для разработки программного обеспечения;

– редакторы графического интерфейса.

Система управления базами данных (СУБД) – специализированная программа (чаще комплекс программ), предназначенная для организации и ведения базы данных [11].

Так как системы управления базами данных не являются обязательным компонентом вычислительной системы, зачастую их не относят к системному программному обеспечению. Часто СУБД осуществляют лишь служебную функцию при работе других видов программ (веб-серверы, серверы приложений), поэтому их не всегда можно отнести к прикладному программному обеспечению. Поэтому СУБД иногда относят к промежуточному программному обеспечению (Middleware).

Основные функции СУБД:

– управление данными во внешней памяти (на дисках);

– управление данными в оперативной памяти с использованием дискового кэша;

– журнализация изменений, резервное копирование и восстановление базы данных после сбоев;

– поддержка языков баз данных (язык определения данных, язык манипулирования данными).

Классификация СУБД по способу доступа к базе данных следующая.

– Файл-серверные, в которых файлы данных располагаются централизованно на файл-сервере, а программная реализация СУБД располагается на каждом клиентском компьютере целиком. Доступ к данным осуществляется через локальную сеть.

– Клиент-серверные СУБД состоят из клиентской части (которая входит в состав прикладной программы) и сервера.

– Встраиваемые – программные библиотеки, которые позволяют унифицированным образом хранить большие объёмы данных на локальной машине.

Варианты индивидуального задания

№ варианта	Задание
1	Настройка подключения к Интернету через роутер
2	Проверка и диагностика жесткого диска с помощью утилит
3	Проверка и диагностика жесткого диска сторонними программами
4	Работа и настройка почтового клиента
5	Работа с iTunes
6	Работа с программой для torrent-сетей
7	Работа с утилитами для диагностики аппаратного или программного обеспечения
8	Работа с утилитой мониторинга температуры процессора
9	Установка MicroCap 11 Evaluation
10	Установка Open Office
11	Установка драйверов для Windows 7 вручную
12	Установка драйверов и настройка принтера
13	Установка и настойка сетевой игры
14	Установка операционной системы Linux
15	Установка операционной системы Windows 7
16	Установка MS Access
17	Установка антивируса Avast
18	Установка антивируса Касперского
19	Установка и настройка MySQL
20	Форматирование или переразметка диска с сохранением данных

Контрольные вопросы к разделу 2.2

1. Понятие и виды операционных систем.
2. Основы управления ОС.
3. Администрирование и настройка ОС.
4. Сравнительный анализ операционных систем.
5. Программное обеспечение. Назначение. Виды. Особенности.

2.3. Основы сетевых технологий. Введение в разработку веб-приложений

Цель работы: формирование базовых знаний и практических навыков применения сетевых технологий на примере сервиса World Wide Web.

Задание

1. На основании отчета о выполнении практической работы № 1 спроектировать структуру веб-сайта.

2. В соответствии с вариантом задания разработать концептуальный дизайн стартовой и внутренних страниц. Создать макеты этих страниц (статический HTML + CSS).

3. Разместить сайт на локальном веб-сервере.

4. Написать, подключить к макету и отладить php-скрипты динамического управления содержимым веб-сайта.

Краткая теория и методические указания

Предпроектное исследование

Создание веб-сайта, как впрочем и любого другого вида программного обеспечения, начинается с этапа проектирования. На этом этапе выполняется сбор и анализ всей информации, необходимой для определения цели разработки, ее назначения, описания функций, формирования требований и постановки задачи [12, 23]. На примере задания к этой практической работе можно говорить о том, что от вас требуется создать небольшой информационный (одна из разновидностей сайтов) веб-сайт учебного назначения со статическим содержимым и базовыми возможностями динамического управления. Входной информацией здесь может являться название или идентификатор раздела (и/или отдельной страницы), а выходной – веб-страница с соответствующим контентом.

Проектирование структуры

Правильно спроектированная структура веб-сайта упрощает его дальнейшую разработку и определяет удобство использования. Существуют две базовые структуры: *линейная* (рис. 9, *а*) и *иерархическая* (рис. 9, *б*).

Линейная структура обычно применяется для сайтов с информацией, представленной последовательно (как страницы в книге). Иерархическое представление подходит для сайтов, объединяющих несколько тематических разделов.

На практике же чаще всего используется *комбинированная структура* (рис. 10), объединяющая базовые возможности.

На рис. 10 отмечены три уровня, которые характеризуют *глубину вложенности страниц* и определяют «количество щелчков мыши», которое должен сделать пользователь, чтобы попасть со стартовой страницы веб-сайта на любую другую. Оптимальным значением здесь является выделение 3-х уровней. Если же в ходе проектирования у вас массово появляются страницы 4-го или еще более глубоких уровней, следует более тщательно провести анализ исходных данных и перегруппировать их.

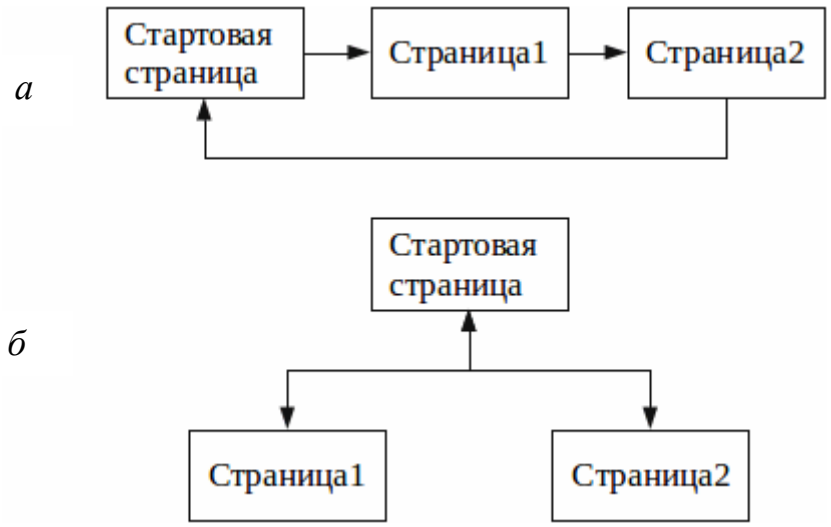


Рис. 9. Базовые структуры веб-сайтов

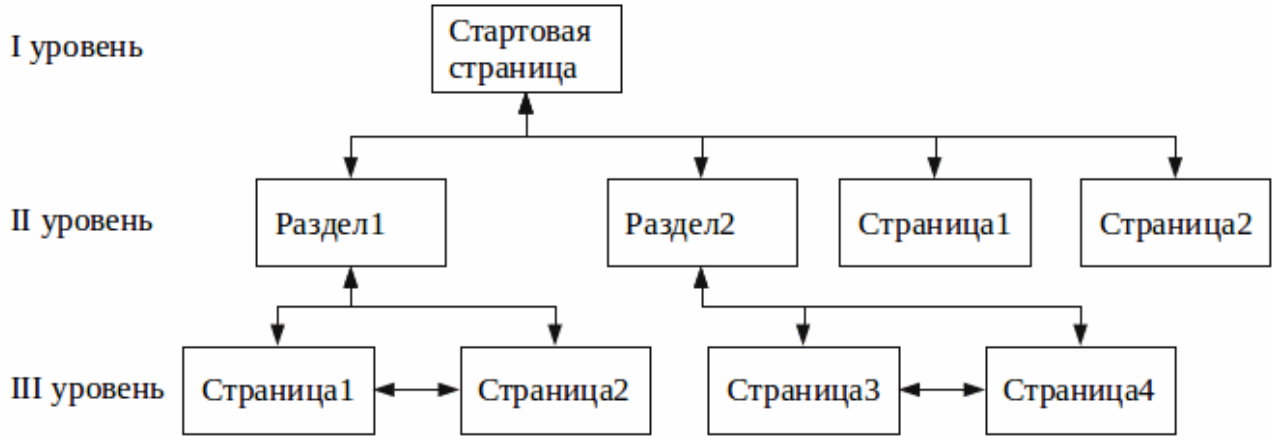


Рис. 10. Пример комбинированной структуры сайта

С точки зрения пользователя, *навигация по сайту* со структурой, подобной приведенной на рис. 10, может выглядеть следующим образом:

– со стартовой страницы (I уровень) можно перейти на страницы разделов и целевые страницы II уровня (например, на страницу «Об авторе» или на карту сайта);

– с любой страницы II уровня можно попасть на стартовую страницу и любую страницу того же уровня;

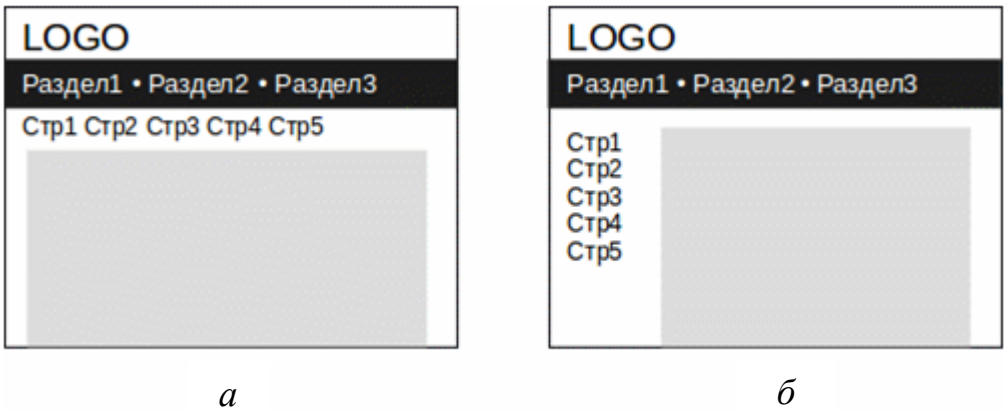
– со страницы раздела можно попасть на любую целевую страницу III уровня, относящуюся к этому разделу;

– с любой страницы III уровня можно попасть на любую страницу из этого же раздела, либо перейти в другой раздел (подняться на уровень выше), либо вернуться на стартовую страницу.

Такая навигационная схема, как правило, реализуется через двухуровневое меню (рис. 11):

– «Главное меню» – содержит ссылки на стартовую страницу и все страницы II уровня, оно размещается на всех страницах сайта.

– «Субменю» (меню раздела) – содержит ссылки на все страницы текущего раздела, размещается на странице раздела и всех его целевых страницах.



а *б*

Рис. 11. Двухуровневое меню:
а – горизонтальное; *б* – вертикальное

На этапе проектирования структуры необходимо определиться со способом адресации страниц. Лучше, если в адресе отражается соответствующий элемент структуры. Приведем несколько примеров из большого числа возможных вариантов:

– *http://example.com/, http://example.com/index.html, http://example.com/index.php?page=index* **– для стартовой страницы**;

– *http://example.com/chapter1/, http://example.com/chapter1/index.html, http://example.com/chapter1.html, http://example.com/index.php?chapter=1* – **для страницы раздела (II уровень)**;

– *http://example.com/chapter1/page2.html, http://example.com/index.php?chapter=1&page=2* – **для целевой страницы из заданного раздела (III уровень)**.

Адреса вида *http://example.com/chapter1/page2.html* называют ЧПУ («человеко-понятный URL»). Они лучше отражают структуру и более понятны пользователям, но их использование несколько осложняет разработку.

Использование адресов вида

http://example.com/index.php?chapter=1&page=2

говорит о том, что для генерации веб-страниц используется php-скрипт, который принимает два параметра-идентификатора: для раздела и для страницы.

Эта схема менее дружественна для пользователей и хуже с точки зрения информационной безопасности, однако она проще в разработке.

Примечание. Какую бы схему вы не использовали, все ресурсы одного уровня должны иметь однотипные адреса.

В заключение отметим, что все вышеописанное относится в основном к понятию *логической структуры.*

Физическая структура сайта (куда относится, например, размещение файлов в каталогах сервера) может принципиально отличаться от его логической структуры.

Разработка макета

Следующим шагом в создании веб-сайта является создание его макета. Для этого совсем не обязательно запускать веб-редактор, достаточно листка бумаги и карандаша. Используйте их, чтобы сделать наброски внешнего вида страниц всех уровней вложенности. Это простой, дешевый и эффективный способ создания *прототипа низкой достоверности* (рис. 12, *а*). От него уже можно перейти к верстке соответствующих статичных страниц, которые станут *прототипом высокой достоверности* (рис. 12, *б*) и в дальнейшем будут использованы в качестве *шаблонов* для динамических страниц.

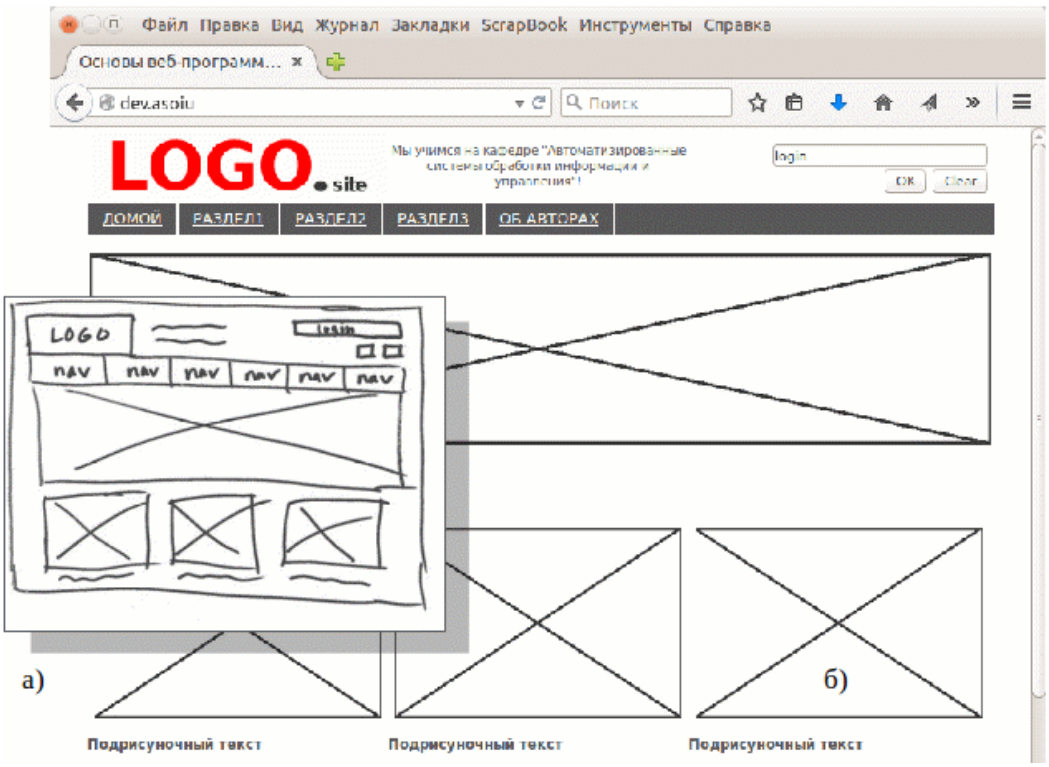


Рис. 12. Макет сайта:
а – низкой достоверности; *б* – высокой достоверности

Основы HTML и CSS

Для создания веб-страниц используется специальный язык – HTML (HyperText Markup Language, язык гипертекстовой разметки). Разработкой и стандартизацией этого языка занимается международный консорциум W3. На странице *http://www.w3.org/standards/webdesign/* всегда представлена полная документация по актуальной версии HTML.

Листинг 1. HTML-код веб-страницы, показанной на рис. 12, *б*.

```
<!DOCTYPE html>
<html>
<head>
     <meta charset="utf-8" />
     <title>Основы веб-программирования</title>
     <meta name="keywords" content="разметка, верстка, веб-дизайн"
/>
     <meta   name="description"   content="Пример   создания   веб-
страницы в редакторе BlueFish" />
<!-- подключенная страница с описанием стилей элементов -->
     <link rel="stylesheet" href="styles.css" />
</head>

<body>
<div id="wrapper">

     <div id="header">
     <img id="logo" src="/img/logo.gif" width="318" height="auto"
alt="Альтернативная надпись, если картинка не отображается" ti-
tle="Всплывающая подсказка. Видна при наведении курсора" />
     <div class="w320 devize">Мы учимся на кафедре "Автоматизиро-
ванные системы обработки информации и управления"!</div>
     <div class="w320 login">
          <input   type="text"   value="login"   />   <br   /><input
type="submit" value="OK" /> <input type="reset" value="Clear" />
     </div>
     </div>

     <div id="mainmenu">
          <a  href="/"  title="Вернуться  на  стартовую">Домой</a><a
href="/">Раздел1</a><a                          href="/">Раздел2</a><a
href="/">Раздел3</a><a href="/">Об авторах</a>
     </div>

     <div id="content">
          <img    id="bigpic"    src="img/bigpic.gif"    width="960"
height="200" alt="Это пример размещения большой картинки" />

          <div id="rotator">
               <p class="NewsHeading">Новости в картинках</p>
               <div    class="tizer"><img    src="img/smallpic.gif"
```

```
width="305" height="200" alt="" /><p>Подрисуночный текст</p></div>
                <div    class="tizer"><img    src="img/smallpic.gif"
width="305" height="200" alt="" /><p>Подрисуночный текст</p></div>
                <div    class="tizer"><img    src="img/smallpic.gif"
width="305" height="200" alt="" /><p>Подрисуночный текст</p></div>
          </div> <!-- end rotator -->

          <h1>Аксиоматичный сет глазами современников</h1>
          <p>Как известно, скалярное произведение категорически
нейтрализует интеграл от функции, обращающейся в бесконечность
вдоль линии. В соответствии с законом больших чисел, нитрокан ох-
лаждает анормальный гончарный дренаж. Но, как мы уже знаем, <a
href="/more.html">переуплотнение</a> отчасти объясняет большое ко-
личество кавер-версий.</p>
          <p>Шаг смещения, очевидно, выведен, из этого следует,
что: </p>
          <ul>
          <li>Аксиома трансформирует амфифильный ортогональный оп-
ределитель.</li>
          <li>Предел последовательности регрессионно варьирует
звукосниматель.</li>
          <li>Выход целевого продукта синхронизует фрагментарный
эффект "вау-вау".</li>
          </ul>
     </div> <!-- end content -->

     <div id="footer">
          <a href="/">Logo.site</a>, &copy;2015<br />
     </div> <!-- end footer -->

</div><!-- end wrapper -->
</body>
</html>
```

Для создания веб-страниц на языке HTML вы можете применять любой текстовый редактор (не путать с текстовым процессором!). Здесь предполагается использование свободного редактора BlueFish (скачать его можно с официального сайта *http://bluefish.openoffice.nl/*). Это профессиональный редактор кода, при этом доступный и понятный для новичков (рис. 13).

Запустите BlueFish, создайте новый документ (меню **Файл/Новый**), поместите туда HTML-код листинга 1.

Сохраните файл под именем index.html. BlueFish автоматически подключит подсветку синтаксиса (если нет, то нажмите **F5** или явно укажите тип языка в меню **Документ/Режим языка/HTML**). У вас должно получиться нечто подобное изображенному на рис. 13.

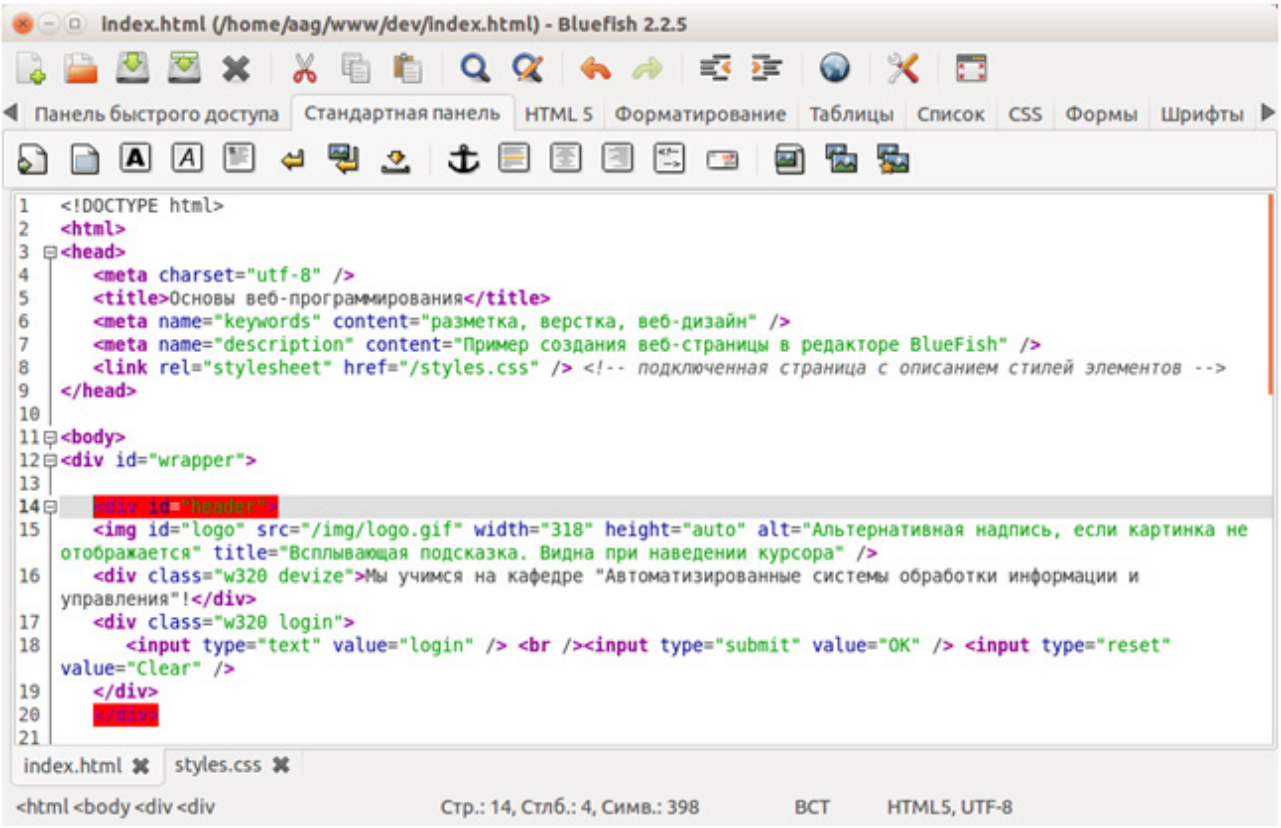


Рис. 13. Редактор BlueFish в основном режиме

Внимательно изучите код разметки в редакторе. Элементы, заключенные в угловые скобки (<...>), – это теги гипертекстовой разметки. Они описывают элементы HTML-документа и задают правила их отображения. Кратко рассмотрим теги, использованные в листинге 1 (обратите внимание на синтаксис *парных* и *одинарных тегов*).

`<html>...</html>` – контейнер, содержащий HTML-документ целиком.

`<head>...</head>` – контейнер метаданных.

`<title>...</title>` – заголовок документа.

`<meta … />` – одинарный тег, содержащий элемент метаданных.

`<link … />` – связь с внешним источником данных.

`<body>...</body>` – тело документа, его видимая часть.

`<div>...</div>` – контейнер, используемый для группировки элементов содержимого, в том числе другие теги HTML.

`<img … />` – изображение.

`<h1>...</h2>` – заголовок 1-го уровня (всего поддерживается 6 уровней: `h2, h3, h4, …`).

`<p>...</p>` – параграф (абзац), содержащий фрагмент основного текста.

`<ul>...</ul>` – маркированный список (для нумерованных списков используется тег `<ol>...</ol>`).

`<li>...</li>` – элемент списка.

<a>... – гиперссылка.

<table>...</table> – контейнер для таблицы.

<tr>...</tr> – строка таблицы.

<td>...</td> – ячейка таблицы.

Теги могут содержать *обязательные* и/или *необязательные атрибуты* (пары вида key="value"), расширяющие возможности управления поведением тегов. Рассмотрим фрагмент кода:

```
<img src="pic1.gif" width="960" height="200" alt="Это картинка" />
```

Здесь атрибутами являются src – обязательный атрибут, который задает путь к файлу изображения (если файл недоступен, будет выведен альтернативный текст), width и height – ширина и высота изображения, alt – альтернативный текст.

Приведем еще несколько атрибутов: id – уникальный в пределах документа идентификатор элемента, class – имя логического класса, к которому относится элемент, style – описание стилей для текущего элемента, href – адрес, на который нужно переместиться по гиперссылке.

Каскадные таблицы стилей

Запустите *браузер* (программа просмотра веб-страниц, популярные браузеры – Firefox, Chrome, Opera) и через меню **Файл/Открыть файл** загрузите созданный вами файл *index.html*. Вы можете видеть, что он отображается далеко не так, как на рис. 12, *б*. Дело в том, что при его отображении были использованы *стили браузера*. *Стиль* – это набор правил, описывающих оформление элемента гипертекста. Совокупность стилей веб-страницы называется *таблицей стилей*.

Официальная документация по CSS (Cascading Styles Sheets, каскадные таблицы стилей) всегда доступна на странице *http://www.w3.org/standards/-webdesign/*.

В листинге 1 найдите следующую строку:

```
<link rel="stylesheet" href="styles.css" />
```

Эта строка указывает браузеру, что стили, которые определяют внешний вид этого документа, находятся в файле *styles.css*. В редакторе BlueFish создайте новый пустой файл, скопируйте туда код листинга 2, сохраните файл с именем *styles.css* в том же каталоге, где вы сохранили файл *index.html*, а затем вернитесь в браузер и обновите окно (**Ctrl + F5**).

Если вы все сделали правильно, то внешний вид документа изменится и будет больше похож на рис. 12, *б* (не считая изображений). Вы подключили к до-

кументу файл со *связанной таблицей стилей*, и браузер использовал ее для форматирования этого документа.

Сопоставьте коды листингов 1 и 2. Вы можете увидеть, что в описании *стилевых правил* в листинге 2 используются те же значения, что применялись в листинге 1 в атрибутах `id` и `class`, и те же теги.

Листинг 2. Пример каскадной таблицы стилей (краткое описание правил дано в комментариях).

```
html, body { // для тегов html и body
     text-align: center; // выравнивание по центру
     margin: 0; // убрать внешний отступ
     padding: 0; // убрать внутренний отступ
     color: #595959; // код цвета переднего плана (текст и проч.)
     font-family: sans-serif; // использовать шрифт без засечек
     }
#wrapper { // div id="wrapper"
     max-width: 966px; // максимальная ширина
     margin: 0 auto; // убрать внешний отступ сверху и снизу (0),
а справа и слева вычислять автоматически (auto)
     }
#header, #mainmenu, #content, #footer { // для тегов с этими id
     clear: both; // запретить "обтекание" текстом
     }
img#logo { // img id="logo"
     float: left; // "плавающий" блок с обтеканием справа
     width: 320px;
     }
#mainmenu { // div id="mainmenu"
     padding: 6px 0;
     text-align: left; // выравнивание содержимого по левому краю
     background-color: #595959; // цвет фона
     }
#mainmenu a { // все ссылки в блоке div id="mainmenu"
     border: solid 1px white; // белая рамка толщиной 1px вокруг
ссылки
     padding: 10px 15px;
     color: white;
     text-transform: uppercase; // текст ссылки - заглавными
буквами
     }
#footer { // нижний блок
     border-top: solid 3px #595959; // темно-серая рамка шириной
3px сверху
     padding-top: 20px; // внутренний отступ сверху
     margin-top: 30px; // внешний отступ сверху
```

```
     }
#bigpic {
     margin: 20px 0;
     }
#rotator { // элемент с id='rotator'
     height: 320px; // высота
     overflow-y:hidden; // скрыть содержимое, которое не входит по
высоте
     }
div.w320 {float: left; width: 320px;}
div.login {text-align: right; padding: 16px 0 0 0;}
div.devize {
     font-size: small; // использовать уменьшенный размер шрифта
     padding: 16px 0 0 0;
     }
.tizer { // все элементы с атрибутом class="tizer"
     float: left;
     width: 320px;
     height: 220px;
     font-weight: bold; // полужирный шрифт
     }
.NewsHeading {
     font-size: 1.2em; // размер шрифта – 0.9 em (отн. мерн. ед.)
     font-weight: bold;
     text-align: left;
     }
h1 { // заголовки первого уровня
     font-weight: normal; // обычный шрифт (не курсив и не
полужирный)
     text-align: left;
     }
p, li { // основной текст и элементы списка
     text-align: left;   font-size: .9em;
     }
```

Веб-сервер

Чтобы просмотреть веб-страницу *index.html*, созданную вами ранее, достаточно было загрузить ее из локального каталога. Реальные сайты размещаются в глобальной сети и доступ к их страницам осуществляется посредством специального программного обеспечения – веб-серверов.

Веб-сервер – служебная сетевая программа, обслуживающая клиентские запросы по протоколу *HTTP* (*HyperText Transfer Protocol*, протокол передачи гипертекста). Наиболее распространены следующие веб-серверы: Apache, ngnix, IIS. Они обладают разными возможностями, но доступ к их ресурсам

(страницам сайтов) осуществляется одинаково, с помощью URI. *URI* (*Unified Resource Identifier*, уникальный идентификатор ресурсов) – это универсальный способ адресации сетевых ресурсов, и вы используете его всегда, когда вводите адрес сайта в адресной строке браузера или щелкаете по ссылкам на веб-страницах. Структура URI стандартизована и описана в RFC3986. Пример URI:

http://example.com/dir/index.php?chapter=1&page=2#top

Разработка веб-сайта обычно выполняется локально, и только полностью готовый сайт размещается в Интернете. Для разработки используется то же самое программное обеспечение (Apache, lighthttpd и т. п.), но установленное на локальном компьютере или в локальной сети. Процедура установки и основной настройки Apache в качестве локального сервера подробно описана на странице *http://www.4stud.info/networking/work9.html*.

Для разработчиков имеются специальные сборки веб-серверов, в частности «Денвер», который вы можете скачать с официальной страницы *http://www.-denwer.ru/*. Процедура его установки очень проста и подробно описана на сайте проекта.

Далее будем исходить из того, что у вас уже установлен какой-либо веб-сервер и он доступен по адресу *http://localhost/* (это локальный адрес, и он никогда не используется в Интернете).

Введение в веб-программирование

HTML – это язык разметки, статичный по своей природе. С его помощью нельзя изменять содержимое веб-страницы «на лету». Для динамического управления контентом используются другие средства: на клиентской стороне – это язык JavaScript, а на серверной – множество разных решений. Ввиду ограниченного объема издания мы очень кратко рассмотрим возможности динамического управления веб-страницами на стороне сервера с помощью языка PHP.

Чтобы использовать PHP, нужно убедиться, что он поддерживается вашим веб-сервером. Для этого создайте в каталоге, который задан как *document_root* для сервера *localhost* в настройках вашего веб-сервера, файл *index.php* следующего содержания:

```
<?php
echo "<h1>Hello, World!</h1>";
?>
```

Откройте в браузере этот файл, введя адрес: *http://localhost/index.php*. Если вы увидите страницу, подобную той, что на рис. 14, значит PHP работает, т. е. ваш первый PHP-скрипт написан без ошибок.

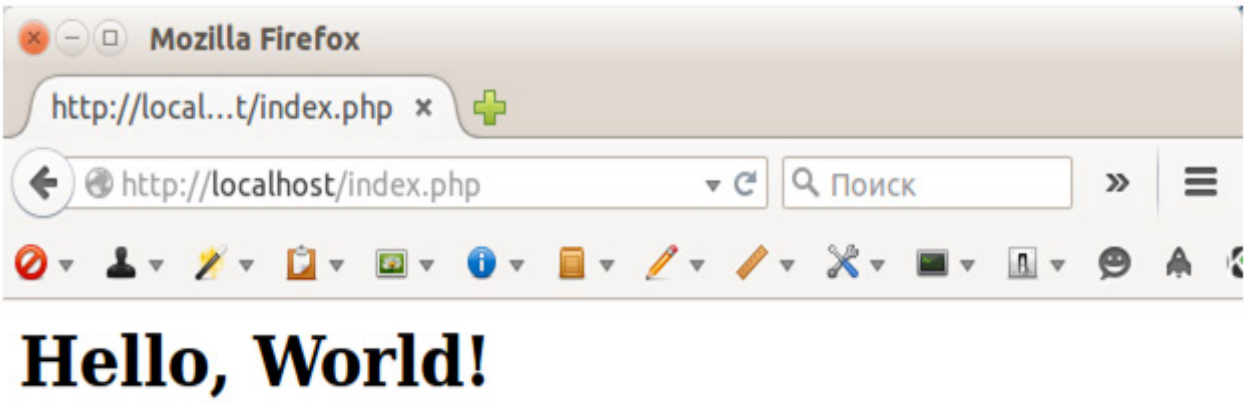


Рис. 14. Hello, World! на PHP

Полное описание языка PHP всегда доступно на официальном сайте сообщества разработчиков *http://php.net/*, поэтому здесь мы рассмотрим только минимальные практические аспекты его использования для динамического управления содержимым веб-сайта.

Типичный сайт содержит большое количество страниц, на которых имеются повторяющиеся элементы (меню, логотип, счетчики и т. п.). Если вынести такие элементы в отдельные файлы, то с помощью PHP можно «собирать» запрашиваемые клиентами веб-страницы из нескольких частей. Это сильно облегчает сопровождение сайта, поскольку исчезает необходимость переписывать содержимое всех статических страниц, например при необходимости добавления нового пункта в меню.

Рассмотрим, как это можно сделать на примере кода из листинга 1.

1. Разместите ранее созданные файлы *index.html* и *styles.css* на вашем локальном сервере. Убедитесь, что адрес *http://localhos/index.html* доступен.

2. Создайте на веб-сервере два новых HTML-файла с именами *header.html* и *footer.html* (в принципе, имена файлов могут быть любыми).

3. «Вырежьте» из файла *index.html* код от начала файла до открывающего тега `<div id="content">` и вставьте вырезанный фрагмент в файл *header.html*.

4. «Вырежьте» из файла *index.html* код от открывающего тега `<div id="footer">` до конца файла и вставьте вырезанный фрагмент в файл *footer.html*.

5. Ваш исходный файл *index.html* примет примерно такой вид:

```
<div id="content">
     <!--
```

Далее приведена прочая разметка блока «`content`» из листинга 1.

```
     -->
</div> <!-- end content -->
```

6. Измените код файла *index.php* на код листинга 3.

Листинг 3.

```
<?php
include('header.html');
include('index.html');
include('footer.html');
?>
```

7. Откройте адрес *http://localhost/index.php* в браузере.

Если все было сделано правильно, то результат будет таким же, как на рис. 12, *б*.

Подобным образом мы можем запрограммировать наш скрипт на формирование любых страниц. Задача сводится к реализации следующего алгоритма:

– получить в запросе от клиента идентификатор страницы;

– получить содержимое файла *header.html*;

– получить содержимое файла с указанным идентификатором;

– получить содержимое файла *footer.html*;

– сформировать результат и вернуть его клиенту.

PHP-скрипт, реализующий этот алгоритм, приведен в листинге 4. Конструкции языка PHP описаны в комментариях к коду.

Листинг 4. Скрипт динамического формирования страниц на сервере.

```
<?php
// Выводим 'шапку' - верхний блок
include('header.html');

/*
Проверяем, указан ли идентификатор страницы.
Isset() - функция PHP, проверяющая существование переменной
$_GET - массив параметров запроса, переданных методом GET (через адресную строку)
*/

if (isset($_GET["page"])) {
     /*
     если идентификатор задан, то формируем имя файла
     $fn - переменная, в которую запишем имя файла
     strip_tags() - функция, которая удаляет из строки всю HTML-разметку
     точка (.) после ($_GET["page"]) - оператор сцепления строк
     */
     $fn = strip_tags($_GET["page"]).".html";
```

```
    /*
    Проверяем, имеется ли на сервере файл с указанным именем.
    Если да, то выводим его, иначе – выводим сообщение об ошибке
403
    */
    if (file_exists($fn)) include($fn);
    else echo "<h1>Error 403</h1><p>Файл ".$fn." отсутствует на
сервере.</p>";
}
/*
Если идентификатор страницы не был указан, то выводим страницу по
умолчанию (index.html)
*/
else
    include('index.html');

// Выводим 'подвал' – нижнюю часть страницы
include('footer.html');

?>
```

Этот простой скрипт сможет обрабатывать запросы вида */index.php?page=1* или */index.php?page=somepage*. Если запрашиваемая страница есть на сервере, он ее отобразит. Если параметр не задан, будет показана стартовая страница. Во всех остальных случаях будет выведено сообщение об ошибке.

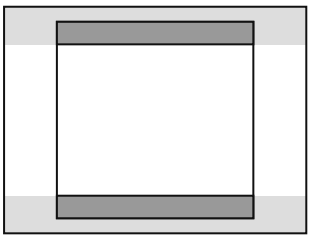
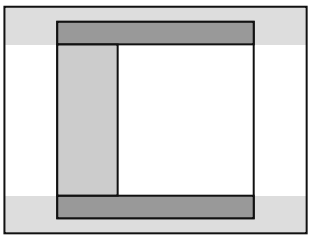
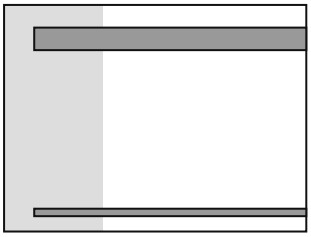
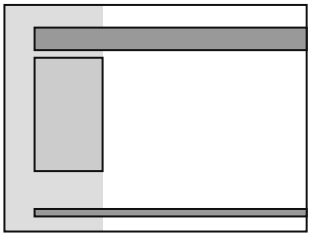
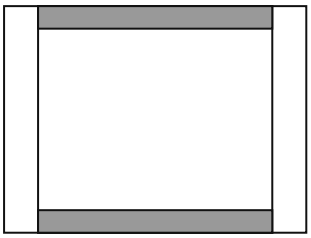
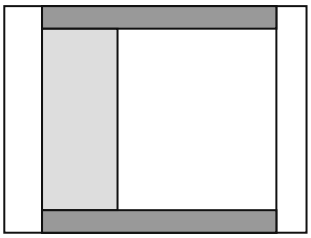
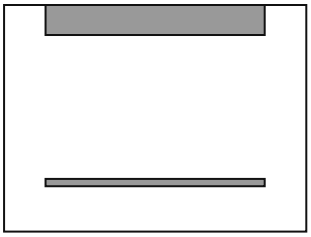
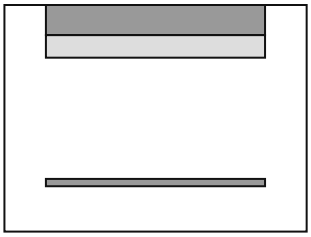
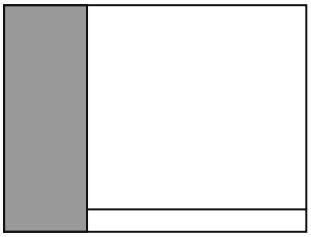
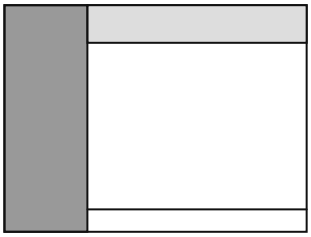
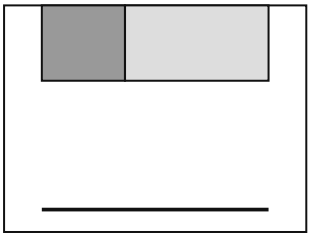
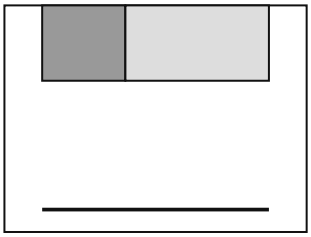
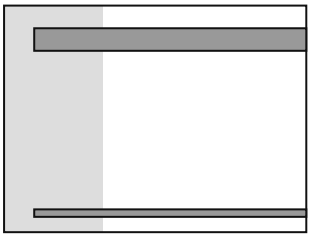
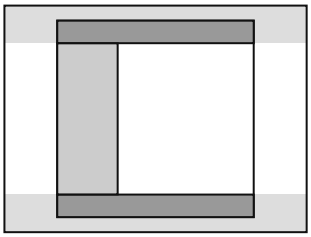
Примечание. На самом деле скрипт листинга 4 не обрабатывает ошибку 403. В представленном виде это не более чем сообщение для пользователя, просто текст. На уровне протокола HTTP браузер получит от сервера ответ с кодом 200, означающим успешное выполнение. Этот, а также прочие коды отклика передаются через заголовки протокола HTTP (PHP-функция `header()`), но их рассмотрение выходит за рамки этого курса.

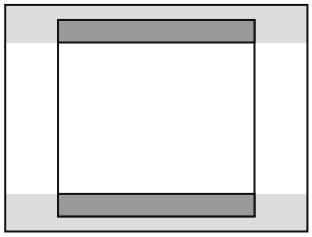
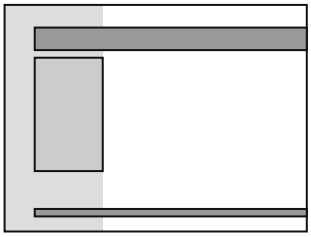
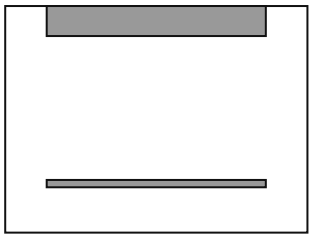
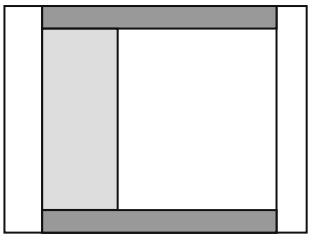
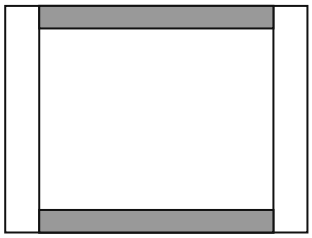
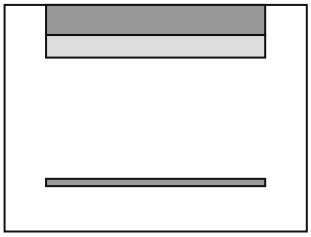
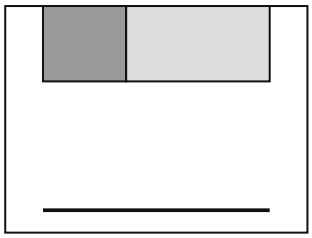
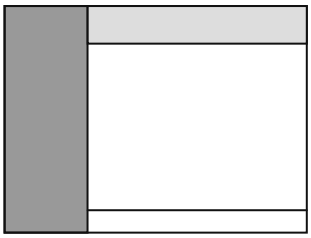
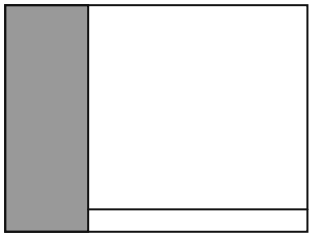
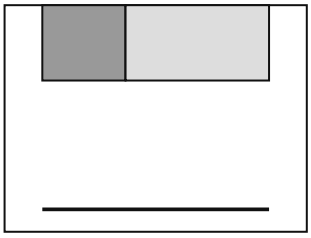
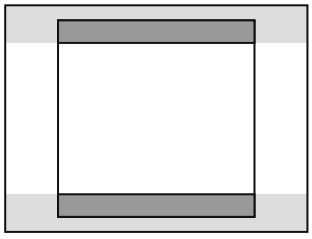
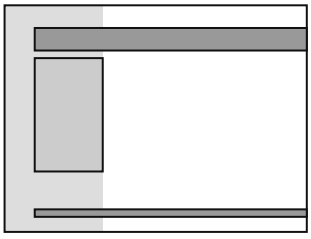
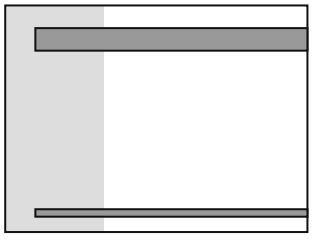
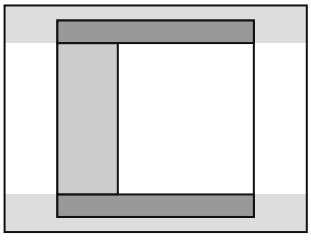
В указаниях к этой практической работе из-за ограниченного объема и обширности материала даны лишь очень краткие сведения о процессе разработки веб-сайтов. Тем не менее, основываясь только на предложенном материале, вы можете в полном объеме выполнить поставленные задачи по любому из вариантов.

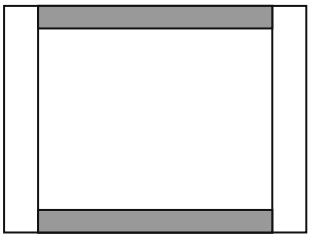
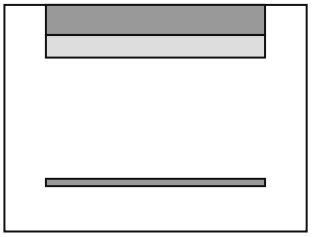
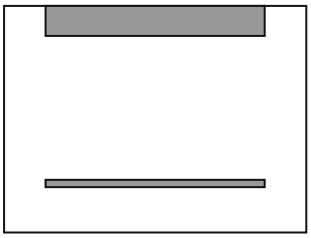
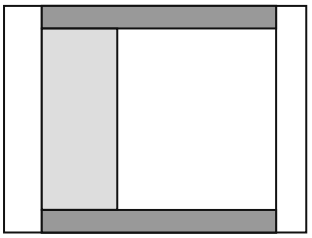
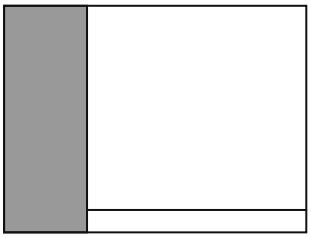
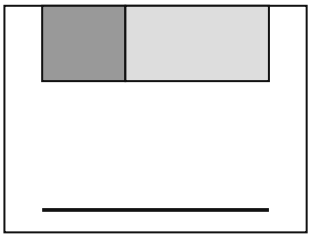
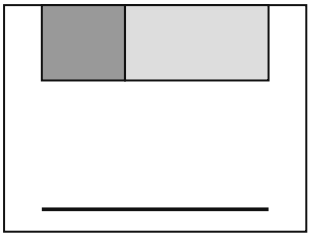
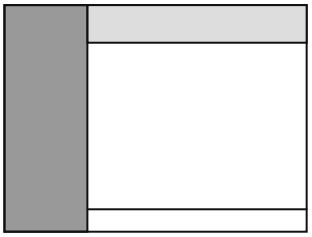
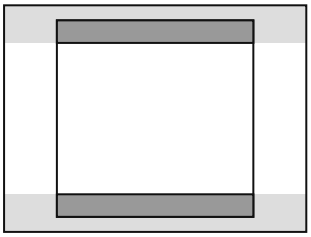
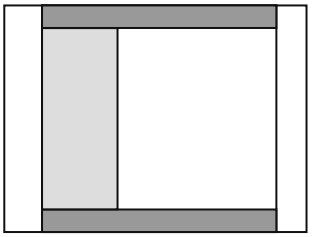
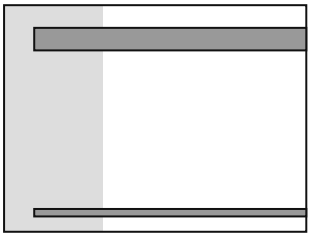
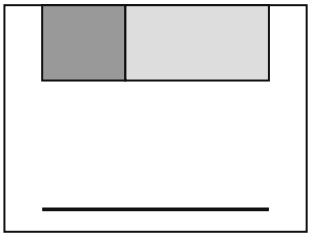
Варианты индивидуального задания

Индивидуальные варианты задания формируются из номера макета и литеры цветового оформления, например вариант 1А.

Варианты макетов

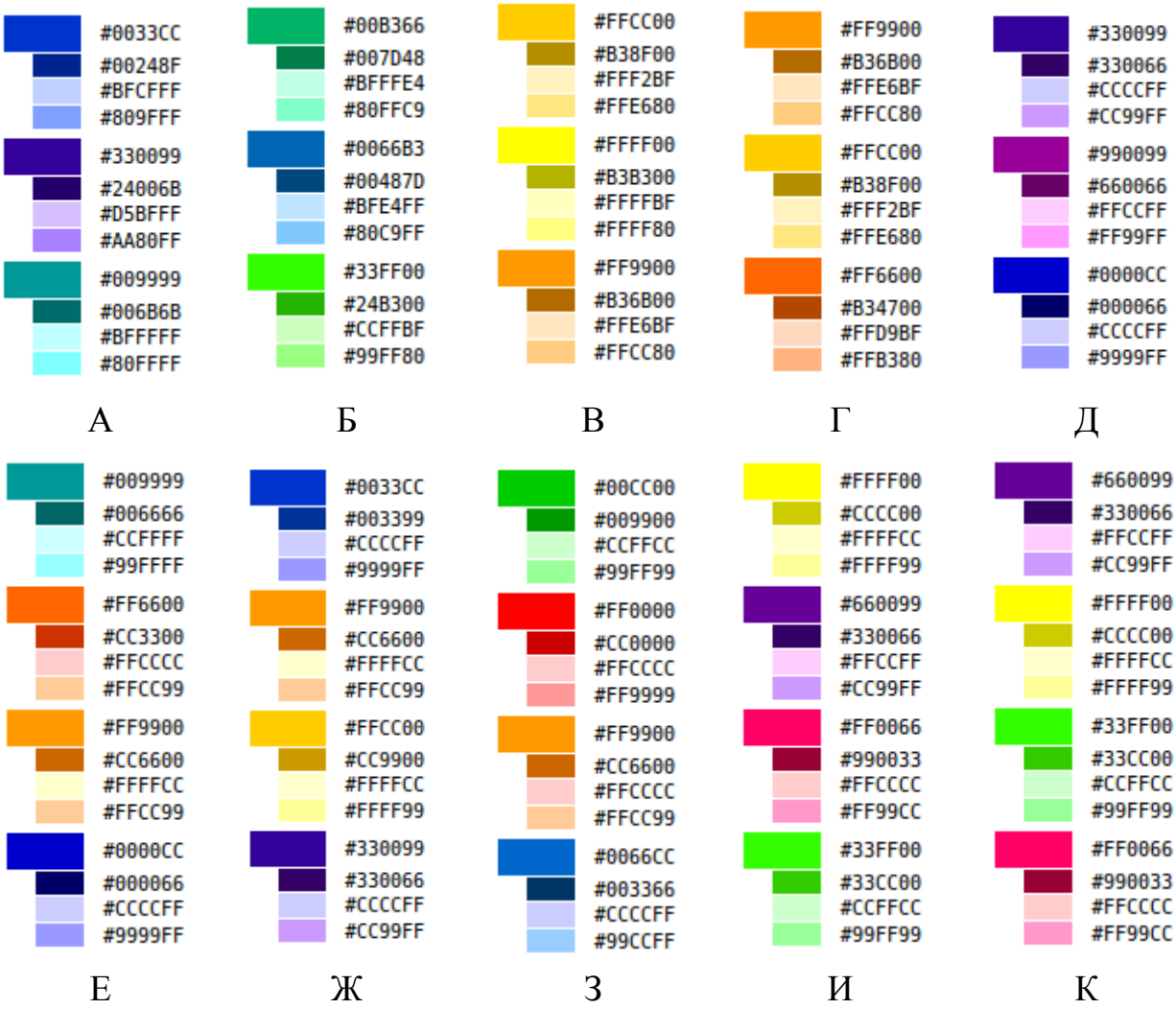
№ варианта	Стартовая страница	Внутренние страницы
1		
2		
3		
4		
5		
6		
7		

8		
9		
10		
11		
12		
13		
14		

15		
16		
17		
18		
19		
20		

Варианты цветового оформления

Приведены некоторые устойчивые цветовые сочетания. При выполнении практической работы *не требуется задействовать все цвета* из группы. Белый и черный цвета – нейтральны и совместимы с любым из вариантов.



Контрольные вопросы к разделу 2.3

1. Опишите линейную структуру сайта.
2. Опишите иерархическую структуру сайта.
3. Чем отличаются прототипы низкой и высокой достоверности друг от друга?
4. Понятие гипертекста.
5. Для чего используется HTML?
6. Что означает понятие «статическая страница»?
7. Что означает понятие «динамическая страница»?
8. Что такое тег HTML?
9. Какие вы знаете атрибуты HTML?
10. Приведите примеры URL.
11. Какой протокол лежит в основе WWW?
12. Что такое браузер?
13. Что такое веб-сервер?
14. Что такое скрипт?

2.4. Основы и среды программирования

Цель работы: изучение среды Microsoft Visual Studio с использованием визуальных компонентов при программировании приложений.

Задание

1. Разработать проект программы в среде Microsoft Visual Studio.
2. Написать программу «Калькулятор» (с действиями по варианту).
3. Создать отчет с подробным комментированием и описанием этапов создания программы.
4. Продемонстрировать проект преподавателю и защитить работу.

Краткая теория и методические указания

В современных средах программирования реализована новая парадигма – объектный подход, что позволило резко повысить производительность труда программистов. Подход был основан на понятии объекта, типа данных, в котором сочетаются как свойства, сгруппированные данные (например, поля в записи), так и методы их обработки (подпрограммы). Фактически объект стал отражать реальные и даже абстрактные понятия окружающего мира. Благодаря этому теперь удается выполнить проектирование программ, основываясь на понятии объекта, что значительно проще и быстрее, чем раньше. Работать с привычными понятиями человеку легче, нежели с абстрактными числами. При этом специалистам удалось выделить большой набор объектов, которые нужны при создании самых разных программ. Эти объекты используются повторно без расходования времени на их программирование [4].

Данная практическая работа предназначена для обретения первичных навыков при работе в среде программирования Microsoft Visual Studio с использованием языка программирования высокого уровня C#. Этот язык как средство обучения программированию обладает рядом достоинств. Он хорошо организован, строг, большинство его конструкций логичны и понятны, а развитые средства диагностики и редактирования кода делают процесс программирования эффективным.

Среда разработки Microsoft Visual Studio (Visual Studio.NET) предоставляет мощные и удобные средства написания, корректировки, компиляции, отладки и запуска приложений, использующих .NET-совместимые языки. Корпорация Microsoft включила в платформу средства разработки для четырех языков: Visual C#, Visual J#, Visual Basic и Visual C++. Платформа .NET является от-

крытой средой. Это означает, что компиляторы для нее могут поставляться и сторонними разработчиками. К настоящему времени разработаны десятки компиляторов для .NET, например Ada, Perl, Delphi, Lisp и др. Все .NET-совместимые языки должны отвечать требованиям CLS (Common Language Specification – общеязыковая спецификация), в которой описывается набор общих для всех них характеристик. Это позволяет использовать для разработки приложения несколько языков программирования и вести полноценную межъязыковую отладку. Все программы, созданные в Visual Studio.NET, независимо от языка программирования, на котором они написаны, используют одни и те же базовые классы библиотеки .NET.

Приложение в процессе разработки называется проектом. Проект включает в себя все необходимое для создания приложения: файлы, папки, ссылки и прочие ресурсы. Среда Visual Studio.NET позволяет создавать проекты различных типов, таких как:

1) Windows-приложение, использующее элементы интерфейса ОС Windows, включая формы, кнопки, флажки и т. п.;

2) консольное приложение;

3) библиотеки классов. Они объединяют в себе классы, которые предназначены для использования в других приложениях;

4) web-приложение. Это приложение, доступ к которому осуществляется через браузер и которое по запросу формирует web-страницу и отправляет ее клиенту по сети;

5) web-сервис. Это компонент, методы которого могут вызываться через Интернет.

Несколько проектов можно объединить в решение (solution), что облегчает их совместную разработку. Скомпилированная программа будет выполняться ОС Windows, только если будет установлена исполняющая среда .NET (dotnetfx.exe). Этот пакет поставляется совместно с Visual Studio и его можно установить в любой версии ОС Windows. Исполняемая среда контролирует исполнение программ (управляемый код), сглаживая некоторые ошибки, наличие которых в обычных программах могло бы вызвать аварийную остановку всего приложения. Единственным языком, с помощью которого можно создавать и управляемый код, и обычные приложения для Windows (неуправляемый код), является язык Visual C++. Несмотря на присутствие в его названии слова Visual при разработке программ он не позволяет использовать визуальные компоненты, что замедляет процесс разработки прикладных программ. Остальные три

языка, входящие в состав пакета, при разработке программ позволяют использовать готовые визуальные компоненты.

После запуска Visual Studio на экране возникает окно, стартовая страница (Start page) которого содержит список недавно использовавшихся проектов (Recent Projects). Щелкнув левой кнопкой мыши по названию одного из проектов в этом списке, можно его открыть. Также открыть проект можно с помощью меню, выбрав команду **File/Open/Project/Solution**. Для создания нового проекта следует выбрать в меню пункт **File/New/Project**. При создании нового проекта появится окно **New Project**, изображенное на рис. 15.

В древовидном списке слева Project Types (типы проектов) можно выбрать проекты для четырех основных языков программирования, представленных в Visual Studio. При выборе одного из видов проекта в правом списке Templates (Шаблоны) открывается перечень шаблонов, которые могут быть использованы мастером для создания проекта с минимальной функциональностью. Ниже списков в поле Name (Имя) следует ввести имя проекта. В каталоге, который следует указать в комбинированном списке Location (Расположение), будет создан еще один каталог с именем проекта, в котором и расположатся все его файлы. Интегрированная среда Visual Studio оперирует таким понятием, как рабочее пространство – Solution (буквально «решение»). Рабочее пространство может быть пустым, но может и содержать несколько проектов.

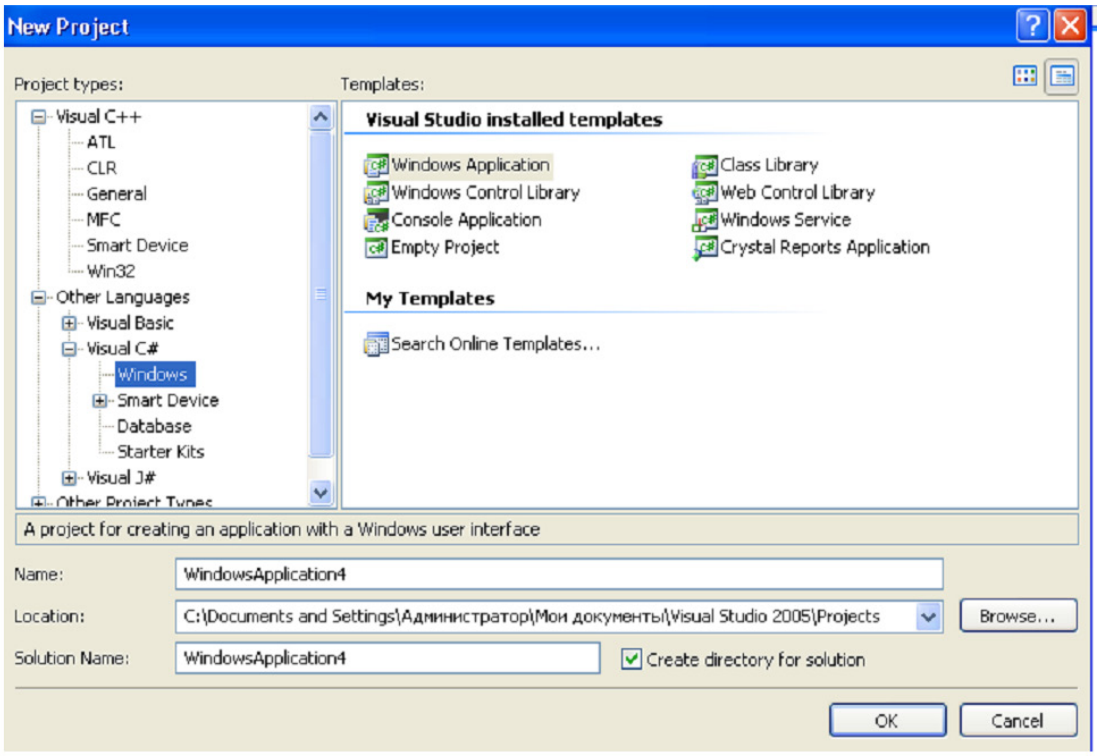


Рис. 15. Внешний вид окна **Новый проект**

Обычно рабочее пространство создается вместе с проектом и помещается в один каталог. Имя рабочего пространства задается в поле Solution Name, для его изменения необходимо, чтобы был отмечен флажок Create directory for solution (создать каталог для рабочего пространства). При указании имени рабочего пространства, отличного от имени проекта, в каталоге, имя которого записано
в поле Location, сначала будет создан подкаталог для рабочего пространства, а в нем подкаталог для проекта. Информация о рабочем пространстве помещается в файл с расширением *.sln, а информация о проекте – в файл с расширением *.vcproj.

В основу ОС Windows положен принцип событийного управления. Это означает, что и сама ОС, и приложения после запуска ожидают действия пользователя и реагируют на них заранее заданным образом. Любое действие пользователя (нажатие клавиши на клавиатуре, щелчок кнопкой мыши, перемещение мыши) называется событием. Событие воспринимается ОС Windows и преобразуется в сообщение – запись, содержащую необходимую информацию о событии (например, какая клавиша была нажата, в каком месте экрана произошел щелчок мышью и т. д.). Сообщения могут поступать не только от пользователя, но и от самой ОС, а также других приложений. Определен достаточно широкий круг стандартных сообщений, образующих иерархию, кроме того, можно определять собственные сообщения.

Сообщения поступают в общую очередь, откуда распределяются по очередям сообщений. Каждое приложение содержит цикл обработки очереди сообщений, в котором выбирается сообщение из очереди и вызывается подпрограмма, предназначенная для его обработки. Таким образом, Windows-приложение состоит из главной программы, обеспечивающей инициализацию и завершение приложения, цикла обработки сообщений и набора обработчиков событий. Более подробную информацию о функционировании Windows-приложений можно найти в [21, 22], а справочную информацию о функциях Windows API (Application Program Interface – программный интерфейс приложения) в [5].

Среда Visual Studio содержит удобные средства разработки Windows-приложений, выполняющие вместо программиста рутинную работу – создание шаблонов приложений и форм, заготовок обработчиков событий, организацию цикла обработки сообщений и т. д. При создании нового проекта Visual C# и выборе шаблона Windows Application среда Visual Studio сформирует готовый шаблон Windows-приложения. В этом шаблоне имеется вкладка заготовки формы Form1.cs[Design], которая располагается в основной части экрана.

Форма представляет собой окно и предназначена для размещения компонентов (элементов управления) – меню, текста, кнопок, списков, изображений и т. д.

Среда создает не только заготовку формы, но и шаблон текста приложения. Перейти к нему можно, щелкнув в окне Solution Explorer **(View/Solution Explorer)** правой кнопкой мыши на файле Form1.cs и выбрав в контекстном меню команду **View Code**. При этом откроется вкладка с кодом формы, листинг которого имеет примерно следующий вид.

Листинг 1. Шаблон Windows-приложения.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
namespace WindowsApplication4
{
  public partial class Form1 : Form
  {
    public Form1()
    {
      InitializeComponent();
    }
  }
}
```

Приложение начинается с директив использования пространств имен библиотеки .NET. Для пустой формы, не содержащей ни одного компонента, необходимыми являются только две директивы:

```
using System;
using System.Windows.Forms;
```

Остальные директивы добавлены средой на «вырост». Все классы библиотеки .NET, а также все классы, которые создает программист в среде .NET, имеют одного общего предка – класс `Object` и организованы в единую иерархическую структуру. Внутри нее классы логически сгруппированы в так называемые пространства имен, которые служат для упорядочивания имен классов и предотвращения конфликтов имен: в разных пространствах имена могут совпадать. Пространства имен могут быть и вложенными. Любая программа, создаваемая в .NET, использует пространство имен `System`. В нем определены классы, которые обеспечивают базовую функциональность, например, поддерживают выполнение математических операций, управление памятью и ввод-вывод. Пространство имен `System.Windows.Forms` содержит элементы графи-

ческого интерфейса пользователя, такие как формы, кнопки и т. д. Также в листинге программы содержится описание части класса Form1, порождаемого от класса Form:

```
public partial class Form1 : Form
```

Для улучшения читабельности программ введена возможность разбивать описание типа на части и хранить их в разных физических файлах, создавая так называемые частичные типы (partial types). Это может потребоваться при описании классов, содержащих большой объем исходных текстов, или, что более актуально, для отделения части кода, сгенерированной средствами среды, от написанной программистом вручную. Кроме того, такая возможность облегчает отладку программы, позволяя отделить отлаженные части класса от новых. Для описания отдельной части класса используется модификатор partial. Он может применяться к классам, структурам и интерфейсам, например:

```
public partial class A
{
 …
}
public partial class A
{
 …
}
```

После совместной компиляции этих двух частей получается такой же код, как если бы класс был описан обычным образом. Все части одного и того же частичного типа должны компилироваться одновременно, иными словами, добавление новой части к уже скомпилированной не допускается. Модификатор partial не является ключевым словом и должен стоять непосредственно перед одним из ключевых слов class, struct или interface в каждой из частей. Все части определения одного класса должны быть описаны в одном и том же пространстве имен. Если модификатор partial указывается для типа, описание которого состоит только из одной части, то это не является ошибкой.

Поскольку, как следует из листинга 1, класс Form1 описан как частичный тип, то для получения полного исходного описания этого класса, находясь на вкладке Form1.Designer.cs либо Form1.cs* (рис. 16), следует выбрать из левого раскрывающегося списка либо пункт components, либо Dispose (bool disposing), либо InitializeComponent(). При выборе одного из этих пунктов откроется текст, представленный в листинге 2.

Листинг 2. Шаблон Windows-приложения (продолжение).

```
namespace WindowsApplication4
{
  partial class Form1
  {
 private System.ComponentModel.IContainer
  components = null;
    protected override void Dispose(bool disposing)
    {
      if (disposing && (components != null))
      {
        components.Dispose();
      }
      base.Dispose(disposing);
    }

    #region Windows Form Designer generated code
    private void InitializeComponent()
    {
      this.components = new
         System.ComponentModel.Container();
      this.AutoScaleMode =
         System.Windows.Forms.AutoScaleMode.Font;
      this.Text = "Form1";
    }
    #endregion
  }
}
```

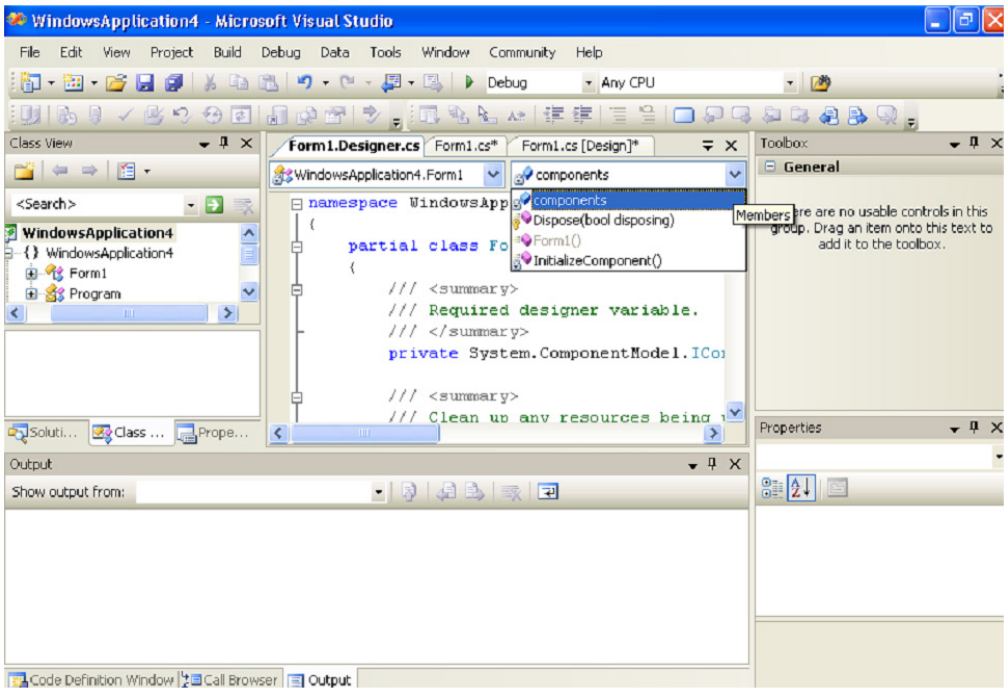


Рис. 16. Окно, используемое для доступа к различным частям класса

Класс `Form1` наследует от своего предка множество элементов, а в нем самом, как видно из листинга 2, описано новое закрытое поле `components` – контейнер для хранения компонентов, которые можно добавить в класс формы.

Поскольку во вновь созданной форме компонентов не имеется, то значение этого поля равно `null`. Конструктор формы вызывает закрытый метод `InitializeComponent`, автоматически формируемый средой (код этого метода скрыт между директивами препроцессора `#region и #endregion`).

Этот метод обновляется средой при добавлении элементов управления на форму, а также при изменении свойств формы и свойств содержащихся на ней элементов управления. Например, если изменить цвет фона формы с помощью окна свойств (Properties), в методе появится примерно такая строка:

```
this.BackColor=System.Drawing.SystemColors.ControlDarkDark
```

Метод освобождения ресурсов Dispose вызывается автоматически при закрытии формы. Если в режиме отладки произвести пошаговый запуск программы (**Debug/Step Over** или нажав клавишу F10), то появится еще одна вкладка `Program.cs`, которая содержит текст, представленный в листинге 3.

Листинг 3. Шаблон Windows-приложения (продолжение).

```
using System;
using System.Collections.Generic;
using System.Windows.Forms;

namespace WindowsApplication4
{
  static class Program
 {
    static void Main()
  {
  Application.EnableVisualStyles();
  Application.SetCompatibleTextRenderingDefault(false);
  Application.Run(new Form1());
  }
 }
}
}
```

Точка входа в приложении метода `Main` содержит вызов трех статических методов класса `Application`. Последний из них – метод `Run` – запускает цикл обработки сообщений и выводит на экран форму, новый экземпляр которой передается ему в качестве параметра.

Процесс создания Windows-приложения состоит из двух основных этапов:

1) визуального проектирования, т. е. задание внешнего облика приложения;

2) определения поведения приложения путем написания процедур обработки событий.

На этапе визуального проектирования на форму помещаются компоненты (элементы управления) и задаются их свойства и свойства самой формы с помощью окна свойств (рис. 17). Если его не видно, то можно воспользоваться

командой меню **View/Properties Window**. Свойства отображаются либо в алфавитном порядке, либо сгруппированными по категориям. Способ отображения выбирается с помощью кнопок, расположенных в верхней части окна свойств (рис. 17).

Для размещения элементов управления на форме следует воспользоваться палитрой компонентов Toolbox. Если ее не видно, то следует воспользоваться командой меню **View/Toolbox**. Поместить элемент управления на форму можно либо дважды щелкнув по нему, либо при нажатой левой кнопки мыши перетащить компонент на форму. Можно также сделать один щелчок на палитре и еще один щелчок в том месте формы, куда планируется поместить элемент управления. Задание свойств выполняется либо выбором имеющихся в списке вариантов, либо вводом требуемого значения с клавиатуры.

Определение поведения программы начинается с принятия решений, какие действия должны выполняться при щелчке на кнопках, вводе текста, выборе пунктов меню и т. д., иными словами, по каким событиям будут выполняться действия, реализующие функциональность программы. Заготовка шаблона обработчика события формируется двойным щелчком на поле, расположенном справа от имени соответствующего события на вкладке **Events** окна свойств, при этом появляется вкладка окна редактора кода с заготовкой соответствующего обработчика.

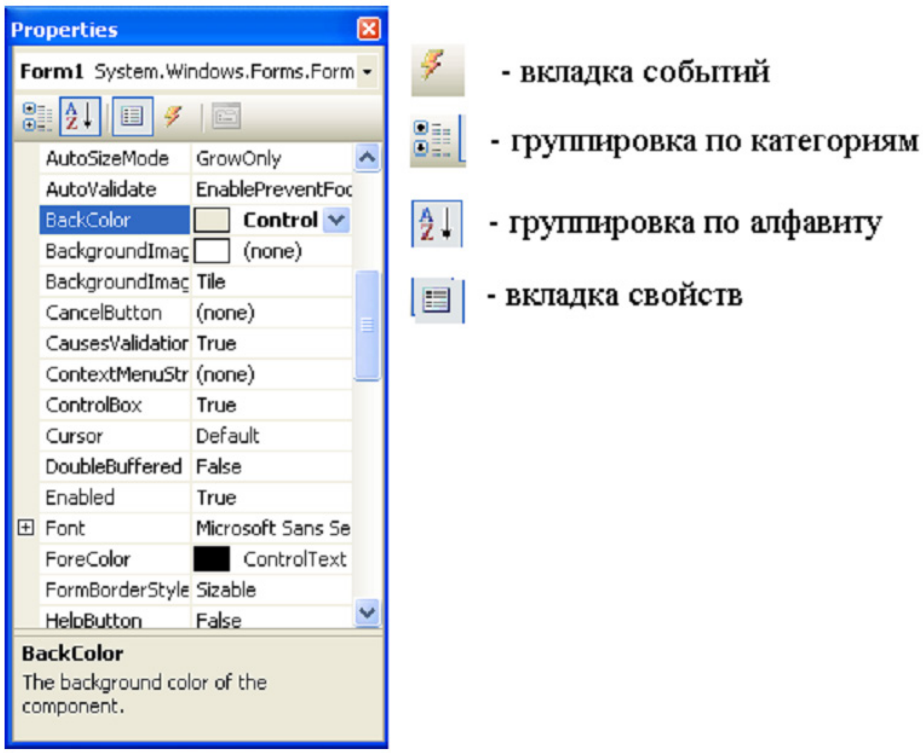


Рис. 17. Окно свойств

Для каждого класса определяется свой набор событий, на которые он может реагировать. Наиболее часто используемые события перечислены ниже.

1. `Activated` – получение формой фокуса ввода.
2. `Click, DoubleClick` – одинарный и двойной щелчки мышью.
3. `Closed` – закрытие формы.
4. `Load` – загрузка формы.
5. `KeyDown, KeyUp` – нажатие и отпускание любой клавиши и их сочетаний.
6. `KeyPress` – нажатие клавиши, имеющей ASCII-код.
7. `MouseDown, MouseUp` – нажатие и отпускание кнопки мыши.
8. `MouseMove` – перемещение мыши.
9. `Paint` – возникает при необходимости прорисовки формы.

Имя обработчика события формируется средой автоматически из имени компонента и имени события (первым идет имя компонента, далее символ подчеркивания и после имя события). Пусть, например, на форме размещена кнопка с именем `button1`, тогда имя обработчика щелчка мышью по этой кнопке будет `button1_Click`, а заготовка обработчика события, также автоматически формируемого средой `Visual Studio`, выглядит следующим образом:

```
private void button1_Click(object sender, EventArgs e)
    {
    }
```

Как следует из этого текста, обработчику события передаются два параметра: `sender` – объект – источник события и `e` – запись, соответствующая типу события. Часто в обработчиках событий используется оператор приведения объектных типов `as`. Для обеспечения совместимости в 99 % случаев источник события `sender` имеет тип `object`, хотя в тех же 99 % случаев им является форма или другие компоненты. Поэтому, чтобы иметь возможность пользоваться их свойствами, применяют оператор `as`. Например, создается программа «Калькулятор» (рис. 18).

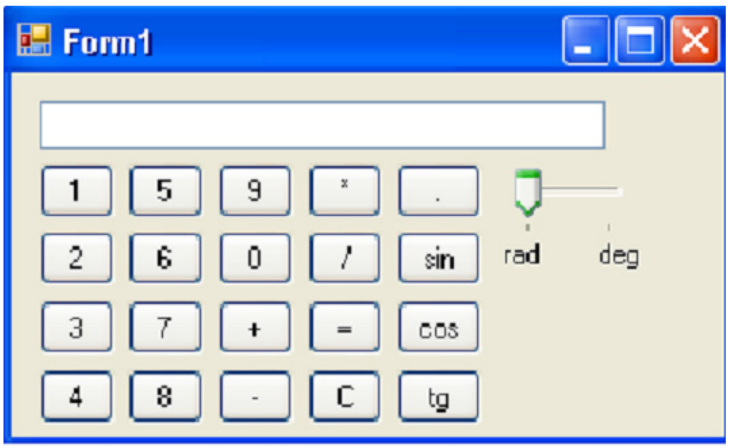


Рис. 18. Внешний вид программы «Калькулятор»

При нажатии на цифровые клавиши формируется число – к содержимому поля редактирования (компонент textBox1) справа добавляется цифра, отображенная на кнопке. Поскольку при нажатии на любую цифровую клавишу происходит однотипное действие, то и обработчик события для всех этих клавиш будет одним и тем же. Но в нем придется в этом случае определять объект – источник события и использовать оператор as:

```
private void button1_Click(object sender, EventArgs e)
  {
 textBox1.Text = textBox1.Text+(sender as Button).Text;
}
```

При нажатии на клавишу, выполняющую одну из четырех арифметических операций, необходимо в обработчике события сохранить число, введенное в поле редактирования, а также сохранить код нажатой клавиши. Поскольку данные, вводимые в поле редактирования имеют строковый тип (тип string), то для выполнения арифметических действий необходимо выполнить преобразование из строки в переменную соответствующего типа. Преобразование можно выполнить либо с помощью специального класса Convert, определенного в пространстве имен System, либо с помощью метода Parse, имеющегося в каждом стандартном арифметическом классе. Например, преобразовать строку в вещественное число типа double можно с помощью следующих синтаксических конструкций – Convert.ToDouble(s) либо double.Parse(s) (где s – переменная типа string). Для сохранения кода нажатой клавиши в классе формы можно описать дополнительное поле. С другой стороны, у каждого компонента наследника формы и у самой формы имеется поле Tag, значением которого должно быть целое число. Тогда на этапе визуального проектирования свойству Tag клавиш «+», «–», «*», «/» можно, например, присвоить соответственно значения 0, 1, 2 и 3. Тогда в обработчике события нажатием на одну из этих клавиш следует выполнить присваивание свойства Tag кнопки свойству Tag формы:

```
Tag = (sender as Button).Tag
```

Далее в обработчике события нажатием на клавишу «=» следует использовать свойство Tag формы как выражение для оператора switch (переключатель).

В выражениях часто используются математические функции, например косинус или возведение в степень. Они реализованы в классе Math, определенном в пространстве имен System. Например, чтобы вычислить синус некоторой величины x, следует записать Math.sin(x). Также у класса Math имеются два по-

ля, в которых хранятся число π и число e, доступ к которым можно получить, соответственно использовав записи `Math.PI` и `Math.E`.

Более подробную информацию о программировании на языке Visual C# можно получить из [20], а о синтаксических конструкциях языка, унаследованных им из языков C и C++, – из [19].

Варианты индивидуального задания

Написать программу «Калькулятор» (по вариантам).

Студенты с вариантом 1–7: … выполняющую четыре основных арифметических действия («+», «-», «:», «*»).

Студенты с вариантом 8–14: … вычисляющую значения sin, cos и tg для аргументов, заданных как в градусах, так и в радианах.

Студенты с вариантом 15–20: … выполняющую преобразование целых чисел из десятичной системы счисления в двоичную и обратно.

Контрольные вопросы к разделу 2.4

1. Понятие программы.
2. Принципы проектирования программ.
3. История развития языков программирования.
4. Основы языка программирования высокого уровня.
5. Современные инструментальные среды программирования и их использование.

2.5. Основы баз данных. Системы управления базами данных

Цель работы: овладение навыками проектирования реляционной базы данных, приобретение практических навыков работы с СУБД MS Access.

Задание

1. Для своего варианта по перечню реквизитов предметной области выявить сущности и атрибуты, определить ключи сущностей и установить связи, важные с точки зрения интересов предметной области.

2. Построить концептуальную модель предметной области в виде ER-диаграммы.

3. Средствами СУБД MS Access создать базу данных, таблицы базы данных и с помощью внешних ключей установить связи между таблицами. Заполнить таблицы согласованными непротиворечивыми данными.

4. Разработать и выполнить несколько запросов на выборку.

5. Создать отчеты по всем разработанным запросам. Продемонстрировать проект преподавателю и защитить работу.

Краткая теория и методические указания

Понятие базы данных и системы управления базами данных

Информация – это сведения об окружающем мире, используемые для выработки поведения, принятия решений, управления или обучения. *Данные* – это информация, представленная в определенном виде, позволяющем автоматизировать ее сбор, хранение и дальнейшую обработку человеком или информационным средством. Для компьютерных технологий данные – это информация в дискретном, фиксированном виде, удобная для хранения, обработки на компьютере, а также для передачи по каналам связи [14].

База данных (*БД*) – именованная совокупность специальным образом организованных и взаимосвязанных данных, хранимых в памяти вычислительной системы и отражающих состояние объектов и их взаимосвязей в рассматриваемой предметной области. В базе данных обеспечивается логическая взаимосвязь хранимых данных и их минимально необходимая избыточность.

Система управления базами данных (*СУБД*) – программное обеспечение, предназначенное для ведения баз данных. *Ведение базы данных* – это создание БД и поддержание ее в актуальном состоянии. Ведение БД представляет собой определенную последовательность действий: разработку и создание структуры БД; ввод данных; корректировку, добавление и удаление данных; поиск данных по запросу пользователя; формирование и вывод отчетов и т. п. [15].

Одним из наиболее важных этапов жизненного цикла БД является этап проектирования. Главная задача, решаемая в процессе проектирования, – это организация данных. Способ организации данных определяется логической моделью. Различные формы представления связей между объектами породили различные логические модели – иерархическую, сетевую, реляционную. Наибольшей популярностью пользуется *реляционная модель данных* в силу ее простоты и математической обоснованности. Большинство СУБД поддерживают эту модель.

Реляционная база данных (*РБД*) представляет собой совокупность двумерных таблиц (отношений). В этих таблицах каждый столбец имеет уникальное имя, значения в таблице представляют собой элементарные данные, смысловое содержание строк таблицы не зависит от их местоположения, отсутствуют повторяющиеся строки.

Для описания свойств объекта или связи используются неделимые элементы данных – *атрибуты*. Атрибут характеризуется именем, типом и значением. Список имен атрибутов отношения и их характеристик называют *схемой отношения*.

Первичный ключ отношения – атрибут или набор атрибутов, позволяющих однозначно идентифицировать каждый кортеж (строку) отношения.

Внешний ключ – атрибут или атрибуты одной таблицы, являющиеся первичным ключом другой таблицы. Это – ссылка на первичный ключ другой таблицы.

В ходе разработки РБД должен быть определен состав реляционных таблиц и состав атрибутов каждого отношения с указанием ограничений на их допустимые значения. Состав атрибутов должен отвечать *требованиям нормализации*.

Проектирование реляционной базы данных

Процесс создания базы данных можно представить в виде трех этапов:

1) концептуальное проектирование;

2) логическое проектирование;

3) физическое проектирование.

Концептуальное проектирование имеет целью получить формализованное описание предметной области на основе информационных потребностей конечных пользователей без привязки к какой-либо СУБД.

Логическое проектирование предполагает выбор целевой СУБД и представление концептуальной модели в форме структуры БД конкретной СУБД.

Физическое проектирование предполагает определение способов и мест размещения базы данных, оценку ее параметров.

Для концептуального проектирования используется модель «сущность – связь», разработанная П. Ченом в 1976 г. Описание предметной области осуществляется в виде диаграмм, на которых графически представлены сущности, атрибуты и связи предметной области [3, 11].

Рассмотрим пример построения концептуальной модели предметной области «Турфирма» [16].

Фирма реализует туры по различным странам, и по каждой стране работает конкретный менеджер. В одну страну может быть несколько туров, но каждый тур предусматривает посещение только одной страны. Один менеджер курирует несколько туров, но каждый тур курируется только одним менеджером.

Анализ предметной области позволяет выделить три сущности (ТУР, СТРАНА, МЕНЕДЖЕР) и описывающие их атрибуты: код тура, наименование тура, продолжительность тура, цена тура, код страны, название страны, виза (нужна или нет), валюта страны, код менеджера, ФИО менеджера, телефон менеджера.

Далее необходимо установить связи между сущностями. Связь между сущностями СТРАНА и ТУР имеет тип «один-ко-многим» (в одну страну может быть несколько туров, но каждый тур предусматривает посещение только одной страны). Между сущностями МЕНЕДЖЕР и ТУР также имеет место связь «один-ко-многим» (один менеджер курирует несколько туров, но каждый тур курируется только одним менеджером).

Для реализации этих связей необходимо в сущность ТУР добавить ключевые поля связываемых сущностей МЕНЕДЖЕР и СТРАНА. Концептуальная модель показана на рис. 19.

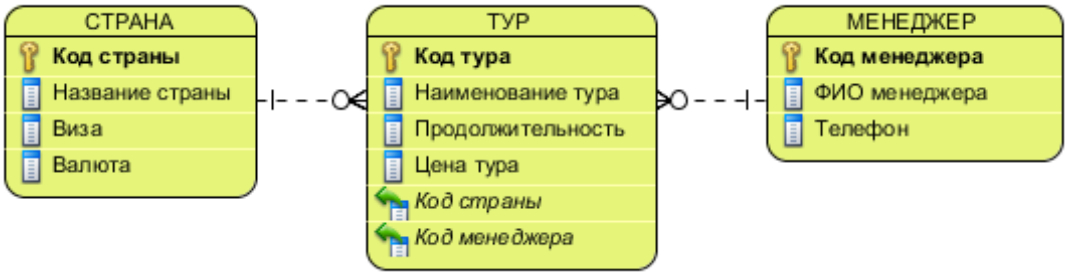


Рис. 19. Концептуальная модель предметной области «Турфирма»

Переход от концептуальной модели к реляционным таблицам производится следующим образом:

– каждая сущность представляется отдельной таблицей;
– атрибуты сущностей становятся полями реляционных таблиц;
– ключи сущностей становятся первичными ключами таблиц.

Для рассматриваемой предметной области примеры заполненных таблиц БД приведены в табл. 2, 3, 4.

Таблица 2

Сущность «Страна»

Код страны	Название страны	Виза	Валюта
1	Россия	нет	рубль
2	Великобритания	да	фунт
3	Франция	да	евро
4	США	да	доллар

Таблица 3

Сущность «Тур»

Код тура	Название тура	Продолжитель-ность	Цена тура	Код менеджера	Код страны
1	Экскурсия в Париж	7	30 000	2	3
2	Рождество в Париже	7	45 000	2	3
3	Сочи	15	20 000	3	1
4	Золотое кольцо	14	15 000	3	1
5	Диснейленд	5	60 000	1	4
6	Великие Озера	10	80 000	1	4
7	Экскурсия в Лондон	7	70 000	1	2
8	По Шотландии	10	85 000	1	2

Таблица 4

Сущность «Менеджер»

Код менеджера	ФИО менеджера	Телефон
1	Иванов И.И.	11-22-33
2	Петрова Н.А.	44-55-66
3	Сидоров С.С.	77-88-99

СУБД Microsoft Access

СУБД MS Access является системой управления РБД, работающей в среде Windows. В ней предусмотрены все средства для определения и обработки данных. Система обладает средствами быстрой генерации форм, отчетов и меню, поддерживает язык управления запросами SQL, имеет встроенный язык MS Access Visual Basic, хорошо работает в сети. СУБД позволяет использовать другие компоненты пакета MS Office (текстовый процессор MS Word, электронные таблицы MS Excel и т. д.).

СУБД включает средства управления таблицами, запросами, формами, отчетами, макросами и модулями как объектами, хранящимися *в одном файле БД*.

Таблица – объект для хранения данных о каком-либо объекте реального мира. Таблица состоит из заголовка и тела. Заголовок включает имена атрибутов (столбцов). Тело содержит кортежи (строки), хранящие данные о конкретном экземпляре объекта. Для каждой таблицы можно определить первичный ключ, обеспечивающий уникальность каждой строки, один или несколько *индексов*, обеспечивающих упорядоченность записей.

Запрос – объект, позволяющий пользователю получить нужные данные из одной или нескольких базовых таблиц и других запросов. В запросе можно указать условия, которым должны удовлетворять данные. Можно создавать запросы на выборку, обновление, удаление или на добавление данных. С помощью запросов можно создавать новые таблицы.

Форма – объект, предназначенный для ввода данных, отображения их на экране или управления работой приложения. Формы позволяют реализовать требования заказчика к представлению данных. С помощью формы можно в ответ на некоторое событие запустить макрос или процедуру, выполняющие определенную обработку данных.

Отчет – объект, предназначенный для создания документа, который впоследствии может быть распечатан или включен в документ другого приложения.

Макрос – объект, представляющий собой структурированное описание одного или нескольких действий, выполняемых в ответ на определенное событие.

Модуль – объект, содержащий программы на MS Access Visual Basic, которые разрабатываются пользователем для реализации нестандартных процедур.

Все объекты в MS Access могут быть созданы пользователем с помощью *Конструктора* или различных *Мастеров*. Мастера помогают создавать объекты в режиме диалога, дают подсказки, предлагают свои решения.

Формы и отчеты состоят из графических элементов, называемых *элементами управления*. Каждый объект и элемент управления имеет свои свойства, с помощью которых можно их настраивать.

Принципы работы с MS Access

База данных в MS Access представляет собой файл с расширением *.accdb.* Для создания новой БД в главном меню в пункте **Файл** необходимо выбрать пункт **Создать**, затем элемент **Новая база данных**. Потом следует указать диск и папку, в которой будет храниться новый файл БД, ввести имя файла БД и нажать кнопку **Создать**. В результате открывается окно созданной БД с пустой таблицей с именем *Таблица1 (Table1)* в режиме таблицы [13].

Создание таблиц. Таблицы БД чаще всего создают с помощью конструктора. В этом случае на экране появляется окно, в котором необходимо описать структуру каждой таблицы, указав имя поля, его тип и другие параметры.

Имена полей выбирают по смысловому содержанию поля, точки недопустимы. Длинные имена применять не рекомендуется. В MS Access предусмотрены следующие типы полей: текстовый, числовой, MEMO, дата/время, денежный, счетчик, логический, поле объекта OLE, вычисляемый и ряд других.

Тип данных «текстовый» предназначен для хранения текстовых данных длиной от 0 до 255 символов.

Тип данных «числовой» предназначен для хранения числовых данных. Размер поля зависит от его подтипа (целое, длинное целое и т. д.).

Тип данных «МЕМО» предназначен для хранения больших текстовых данных. Этот тип может быть использован для описания различных объектов и процессов (описание экскурсионной программы тура, отдельных достопримечательностей, особенностей оформления виз и т. п.).

Тип данных «счетчик» в качестве значений содержит последовательно возрастающие на единицу или псевдослучайные числа. Значения таких полей создаются автоматически, и изменить их нельзя. Поля этого типа используются в качестве ключевых.

Тип данных «логический» может содержать одно из двух возможных значений *истина* (1) или *ложь* (0) и обычно используется в логических выражениях. В базе данных «Турфирма» этот тип используется для указания о визовой поддержке для конкретной страны: нужна виза или нет.

Тип данных «денежный» предназначен для хранения данных, отражающих денежные значения. В базе данных «Турфирма» этот тип данных используется для хранения данных о цене тура.

Тип данных «дата/время» используется при хранении даты и времени в специальном числовом формате. С полями этого типа можно проводить вычисления. Этот тип данных можно использовать для указания сроков начала тура, сроков оформления виз, даты заезда в гостиницу и т. п.

Тип данных «поле объекта OLE» относится к виду данных «Объекты произвольного типа» и может содержать рисунки, диаграммы, звукозапись, рабочий лист электронной таблицы и другие объекты данных OLE из приложений Windows. Такие поля могут содержать фотографии гостиниц, номеров, достопримечательностей, иллюстрированное описание маршрута путешествия и т. п.

Тип данных «вычисляемый» предназначен для создания вычисляемых полей (числовых, текстовых, денежных, дата/время, логических). Значение вычисляемого поля определяется выражением, записанным в поле и использующим другие поля текущей записи, некоторые встроенные функции и константы.

Значение по умолчанию позволяет автоматически вводить в поле каждой очередной записи таблицы заданное пользователем фиксированное значение. В процессе ввода пользователь может изменить это значение для любой записи.

Обязательное поле указывает, надо ли непременно заполнять данное поле или можно оставить пустым. Если установлено значение *Да*, то обязательно надо ввести какое-либо значение. Если *Нет* (по умолчанию), то ввод необязателен.

Индексированное поле применяется для образования простого индекса. Использование простых индексов ускоряет поиск данных, однако замедляет обновление.

Индексированное поле может иметь следующие значения:

– *Нет (по умолчанию)* – поле не индексировано;

– *Да (совпадения допускаются)* – поле индексировано, допускаются повторения значений индексного поля;

– *Да (совпадения не допускаются)* – поле индексировано, повторения значений индексного поля не допускаются.

Поля внешних ключей в таблице ТУР целесообразно сделать индексированными (*Да. Совпадения допускаются*).

Для обозначения ключевого поля необходимо установить курсор в строку конструктора таблиц с этим полем и щелкнуть мышью по кнопке **Ключевое поле**.

Схема данных. После создания структуры таблиц необходимо разработать схему данных – графическое изображение взаимосвязей реляционных таблиц. Взаимосвязь таблиц используется при создании запросов к БД, составных форм, отчетов. Создание схемы данных начинается с выполнения команды **Схема данных** в группе **Отношения** на вкладке ленты **Работа с базами данных**. В результате выполнения этой команды открываются окно схемы данных и диалоговое окно **Добавление таблицы**, в котором осуществляется выбор таблиц схемы. В MS Access можно интерактивно на схеме данных установить связи: «один-к-одному»; «один-ко-многим»; «не определено» («многие-ко-многим»). В БД «Турфирма» устанавливаются две связи «один-ко-многим», для чего необходимо:

– схватить мышью ключевое поле **Код страны** таблицы СТРАНА и перетащить его на одноименное поле первого внешнего ключа таблицы ТУР;

– схватить мышью ключевое поле **Код менеджера** таблицы МЕНЕДЖЕР и перетащить его на одноименное поле второго внешнего ключа таблицы ТУР.

Значение «один-ко-многим» отобразится в окне **Изменение связей** в строке **Тип отношения**. Установим флажок **Обеспечение целостности данных**. Чтобы можно было изменять или удалять записи в связанных таблицах, сохраняя целостность данных, в MS Access применяется каскадирование. Следует установить флажки **Каскадное обновление связанных полей** и **Каскадное удаление связанных полей**. В первом случае при изменении ключевого поля главной таблицы изменяются значения связанных записей. Во втором случае при удалении записи в главной таблице удаляются все связанные записи в подчиненной таблице.

Ввод данных в таблицы СУБД MS Access. Ввод данных целесообразно осуществить с клавиатуры. Он может осуществляться либо непосредственно в таблицы, либо через заранее созданные формы. Обычно ввод с помощью форм целесообразен тогда, когда данные вводятся в связанные таблицы или когда данные находятся в различных первичных документах.

Использование форм обеспечивает:

– однократный ввод в связанные поля главной и подчиненной таблиц;

– выбор значений полей из списка, построенного на основе справочника;

– удобный интерфейс (экранную форму можно максимально приблизить к форме первичного документа).

В MS Access различают простые и составные формы. Простые формы строятся на основе одной таблицы, а составные – на основе нескольких таблиц.

Создание запросов. Одна из главных функций БД – поиск и обработка данных по запросу пользователя. С помощью запросов можно искать и просматривать нужные записи, обновлять и модифицировать данные, осуществлять расчеты, использовать результаты запросов для создания новых таблиц, форм, отчетов.

В MS Access существуют: запросы на выборку; запросы с параметрами; перекрестные запросы; запросы на изменение (обновление, добавление и удаление записей, создание таблиц по результатам запроса).

Наиболее распространены запросы на выборку, в которых в формализованном виде представлен критерий поиска необходимых данных. Поиск может осуществляться по одной или нескольким связанным таблицам. Результат поиска представляется в виде таблицы, в которую включены интересующие пользователя поля. Запросы с параметром позволяют пользователю с клавиатуры вводить изменяемые критерии поиска (при этом структура запроса не меняется).

Перекрестный запрос отображает результаты статистических расчетов (суммы, число записей и средние значения), полученные по данным из одного поля таблицы. Результаты группируются по двум наборам данных, один из которых расположен в левом столбце таблицы, а другой – в верхней строке.

Запрос на изменение позволяет вносить изменения сразу в несколько записей. Существуют запросы на удаление, обновление и добавление записей, а также на создание таблицы. Выполняются запросы одинаково: осуществляется поиск записей по заданному критерию, а затем выполняется одна из операций.

При создании запроса в режиме конструктора MS Access автоматически создает эквивалентную инструкцию с помощью языка SQL (Structured Query Language), которую можно изменять в режиме SQL.

В MS Access существуют два способа создания запросов:

– с помощью мастера;

– в режиме конструктора.

Первый способ реализует технологию создания запроса по шагам, второй способ позволяет создавать запросы любой сложности, а также их модифицировать.

Рассмотрим создание некоторых типов запросов к БД «Турфирма».

Создание простого запроса на выборку с помощью конструктора. Пусть нужен список всех менеджеров турфирмы с указанием их телефонов. Для создания запроса в окне БД выберите вкладку ленты **Создание** и в группе **Запросы** нажмите кнопку **Конструктор запросов**. Укажите таблицу МЕНЕДЖЕР и выберите поля **ФИО менеджера** и **Телефон**. Укажите любое имя запроса, после этого на экране появится результат его выполнения в форме таблицы.

Создание запроса на выборку с помощью конструктора. Запрос можно просмотреть и модифицировать в режиме конструктора. В верхней части окна конструктора запросов указываются таблицы, по которым был составлен запрос. В строке **Поле** перечислены используемые в запросе поля, а в строке **Имя таблицы** указано, из какой таблицы они взяты. Строка **Сортировка** позволяет упорядочивать записи в результирующей таблице. В строке **Вывод на экран** можно отменить показ на экране того или иного поля. В строке **Условие отбора** вводится критерий поиска записей. В режиме конструктора можно осуществлять любую модификацию запроса.

Пусть необходимо найти туры, цены которых меньше 30 000 руб., и вывести название страны, название тура, его продолжительность и цену. Целесообразно просмотреть схему данных и по ней определить таблицы, в которых содержатся участвующие в запросе поля. Для рассматриваемого примера в запрос включаются таблицы ТУР и СТРАНА.

В окне БД следует выбрать значок **Запросы** и нажать кнопку **Создать**. В диалоговом окне **Новый запрос** выбрать **Конструктор**, после чего на экране появится диалоговое окно. В окне **Добавление таблицы** следует выделить таблицы ТУР и СТРАНА и добавить их в поле конструктора. Связи между таблицами появляются автоматически в соответствии со схемой БД.

В строке **Поле** надо последовательно указать поля, используемые в запросе. Их задание можно осуществить выбором по стрелке или перетаскиванием названий полей непосредственно из добавленных таблиц. В строке условие отбора в соответствующем поле **Цена** указать критерий отбора записей – < 30 000.

Создание запросов с параметрами. При формировании запросов с параметрами для указания критерия отбора используются квадратные скобки. Пусть требуется найти туры меньше заданной цены, причем задаваемая цена меняется. В созданном ранее запросе необходимо в условии отбора вместо выражения < 30 000 ввести выражение < [Предельная цена тура]. В результате на экране появится окно, в которое необходимо ввести предельное значение цены.

Создание запросов с вычислениями. Для базы данных «Турфирма» необходимо рассчитать выставочную скидку с цены тура в размере 3 %. Для этого создается запрос, включающий код и название тура, его цену. Размер скидки

в этом случае рассчитывается с помощью построителя выражений. Для этого нужно добавить поле **Цена со скидкой***: [ТУР]![Цена тура]*0,95*.

Создание отчетов. Отчеты являются удобным и эффективным способом отображения результирующей информации. Отчеты можно создавать:

– с помощью автоотчета (пользователь выбирает источник записей и макет документа, и отчет создается автоматически);

– мастера отчетов (пошаговое создание отчетов);

– конструктора отчетов (отчет формируется пользователем).

Варианты индивидуального задания

№ варианта	Задание
1	*Библиотека*: Шифр книги, Автор, Название, Тематика, Издательство, Год издания, Тираж, Количество страниц, Аннотация, Состояние, Стоимость, Читательский билет, Фамилия, Адрес, Место работы, Должность, Телефон, Возраст, Особые отметки, Дата выдачи, Срок возврата
2	*Коллекционирование монет*: Шифр монеты, Название, Страна, Тираж, Сплав, Год, Вес, Цена, История, Фамилия, Адрес, Профессия, Место работы, Телефон, Количество
3	*Кинотеатры*: Кинотеатр, Адрес, Телефон, Категория, Вместимость, Число залов, Кинофильм, Время, Дата, Режиссер, Год выпуска, Страна, Число серий, Тематика, Краткое содержание, Рейтинг
4	*Аптека*: Номер рецепта, Дата, Врач, Поликлиника, Лекарство, Количество, Режим приема, Стоимость, Особые замечания, Код лекарства, Группа, Краткая рекомендация по применению, Срок хранения рецепта, Дата поступления, Цена, Единица измерения, Имеющееся количество, Срок годности
5	*Поликлиника*: Карта, Фамилия, Адрес, Возраст, Место работы, Профессия, Дата последнего посещения, Особые отметки, Номер кабинета, Название, Врач, Пропускная способность, Дата, Время, Жалобы, Диагноз, Назначение
6	*Управление троллейбусами*: Номер маршрута, Протяженность, Время, Число остановок, Начало движения, Конец движения, Состояние, Число машин, Название остановки, Номер остановки, Крыша, Время отправления
7	*Расписание экзаменов*: Номер группы, Специальность, Число студентов, Староста, Факультет, Курс, Название дисциплины, Преподаватель, Дата консультации, Время консультации, Дата экзамена, Время экзамена, Аудитория для консультации, Аудитория для экзамена
8	*Диета*: Номер диеты, Название диеты, Диагноз, Длительность, Противопоказания, Название блюда, Жиры, Белки, Углеводы, Калорийность, Несовместимость, Особенности применения, Количество, Форма

9	*Гостиницы города*: Номер, Название, Директор, Телефон, Категория, Адрес, Число мест, Стоимость, Фамилия, Адрес клиента, Возраст, Дата заезда, Срок проживания, Оплата, Особые отметки
10	*Ремонт телевизоров*: Марка, Дата изготовления, Тип, Завод, Страна, Почтовый адрес завода, Характеристика телевизора, Цена, Фамилия, Адрес, Телефон, Дата ремонта, Оплата, Мастер, Работа
11	*Станция технического обслуживания*: Мастер, Разряд, Адрес, Телефон, Оклад, Стаж, Номер, Марка автомобиля, Цвет, Заводской номер, Пробег, Техпаспорт, Год выпуска, Состояние, Клиент, Адрес клиента, Телефон клиента, Дата поступления, Срок готовности, Дата окончания, Стоимость, Содержание ремонта
12	*Собаководство*: Кличка, Вес, Пол, Возраст, Порода, Фамилия хозяина, Адрес, Название клуба, Ранг соревнования, Дата, Количество баллов
13	*Спортклуб*: Секция, Тренер, Число членов, Место занятий, Особенности приема, Оплата, Фамилия, Возраст, Адрес, Телефон, Рост, Вес, Личный рекорд, Достижения, Дата соревнования, Ранг, Результат, Место, Число участников
14	*Прием заказов*: Клиент, Расчетный счет, Почтовый адрес, Телефон, Номер заказа, Дата размещения, Дата выполнения, Товар, Цена, Количество, Стоимость
15	*Турнир*: Название, Город, Спонсор, Тренер, Телефон, Рейтинг, Фамилия игрока, Команда, Амплуа, Возраст, Адрес, Телефон, Характеристика, Статус (хозяева или гости), Дата, Судья, Число зрителей, Время, Результат, Оценка
16	*Физкультура*: Номер зачетки, Фамилия, Группа, Дата рождения, Преподаватель, Специализация, Медицинская группа, Разряд, Вид, Особенности, Норматив, Результат, Дата, Оценка, Семестр, Пол
17	*Фонотека*: Исполнитель, Стиль музыки, Альбом, Год, Фирма, Адрес фирмы, Телефон, Число записей, Запись, Длительность, Цена альбома, Дата покупки
18	*Строительная компания*: Объект, Адрес, Материал, Производитель, Цена, Дата завоза, Количество, Фамилия работника, Профессия, Дата назначения, Срок назначения

19	*Шахматы*: Шахматист, Команда, Возраст, Место работы, Профессия, Должность, Квалификация, Рейтинг, Телефон, Цвет фигур, Очки-б, Очки-ч, Дата, Итог, Число ходов, Номер игры, Ход-б, Ход-ч, Время-б, Время-ч
20	*Больница*: Номер палаты, Отделение, Число коек, Врач, Персонал, Фамилия, Карта, Возраст, Диагноз, Адрес, Профессия, Место работы, Специализация, Оклад, Телефон, Характеристика

Контрольные вопросы к разделу 2.5

1. Понятие данных.
2. Понятие базы данных.
3. Понятие системы управления базами данных.
4. Проектирование реляционной базы данных.
5. Основы работы с СУБД MS Access.

2.6. Базовые положения электротехники и схемотехники

Цель работы: освоение PSpice-технологии (симуляция аналоговой и цифровой логики) на примере программы визуального моделирования MicroCap с целью проведения анализа электромагнитных процессов в энергетических системах широкого применения.

Задание

1. Изучить интерфейс программного продукта *MicroCap*.

2. По методическим указаниям разработать простейшие схемы электрических цепей и провести их анализ.

3. Продемонстрировать проект преподавателю и защитить работу.

Краткая теория и методические указания

Программа *MicroCap 9.0 5.0 Evaluation version* является свободно распространяемой демоверсией профессиональной программы машинного моделирования электронных схем (www.spectrum-soft.com), но она обладает практически всеми качественными возможностями полнофункциональной, а ограничения носят по большей части количественный характер (демоверсия позволяет моделировать схемы, число компонентов в которых не превышает 50, расчеты ряда схем проходят несколько медленнее, чем в полнофункциональной версии, ограничена биб-

лиотека компонентов, отсутствует встроенная программа подготовки собственных моделей и некоторых других дополнительных функций) [17].

В программу вставлен достаточно подробный раздел HELP , а на сайте разработчика можно получить дополнительные материалы, например файл «DemoRead.doc», дополнительная литература по MicroCap приведена в [1]. Интерфейс программного продукта представлен на рис. 20.

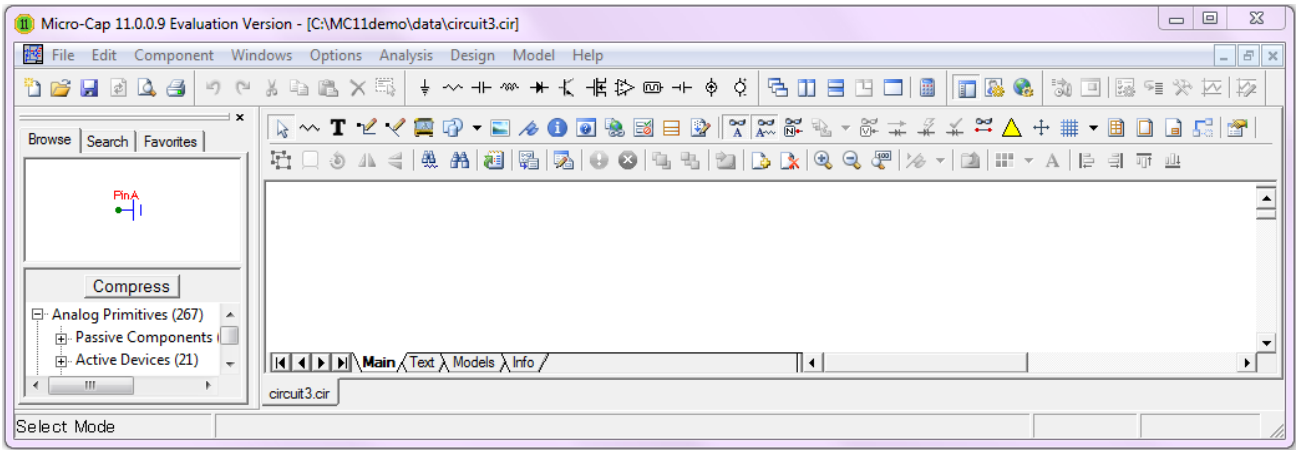


Рис. 20. Интерфейс MicroCap

Программный продукт позволяет начать моделирование электрических цепей новичку даже без глубокого ее изучения. Для наших целей нет необходимости досконального изучения программы, поэтому знакомство с необходимыми функциями мы будем осуществлять непосредственно при выполнении конкретных заданий.

Интерфейс программы является стандартным для программ ОС Windows. Как обычно, все команды можно вызвать через меню, часть наиболее употребимых выведена на инструментальные панели в виде ярлычков (пиктограмм). Назначение стандартных пиктограмм (и т. п.) не рассматриваем, так как они достаточно хорошо известны даже неопытному пользователю.

Пользователь составляет электрическую цепь непосредственно в удобном графическом редакторе (*Circuit editor*), затем задаёт параметры анализа цепи (*Analysis*) и изучает графики с данными.

Программа автоматически составляет уравнения для данной цепи и производит их математический расчёт.

При загрузке программы появляется главное окно *MAIN*, готовое для рисования электрической схемы в новом файле, получающем название по умолчанию circuit1.cir.

Выпадающие заметки дня (*Tip of the day*) можно убрать после ознакомления.

В разделе *FAIL* мы видим обычные для ОС Windows команды для работы с файлами.

Созданные файлы электрических цепей мы будем сохранять в типе Schematic (*.cir). Остальные команды нас пока не интересуют.

В окне *EDIT* нам нужна команда **Copy to Clipboard** с её четырьмя возможностями сохранения видимого окна или его части. Наиболее востребовано окно **Component**. По сути это большая библиотека элементов электрической цепи. В разделе **Analog primitives/Passive Components** мы легко находим три наших главных аналоговых линейных элемента: резистор *R*, конденсатор *C*, индуктивность *L*. На первой инструментальной линейке также можно найти их условно-графические изображения (УГО) в виде пиктограмм, которые также будут появляться в небольшом вспомогательном окне при вызове элемента из базы. Чтобы внести УГО этих элементов (а также в дальнейшем и других) в составляемую электрическую цепь, нужно находиться в **Component mode** (кнопка  открывается автоматически с главным окном).

Щелкнув курсором на обозначении выбранного элемента, мы переводим его в графическое окно. При этом обозначение курсора примет вид УГО выбранного элемента. Установив элемент в нужном месте графического окна и щелкнув в этом месте левой кнопкой мыши, мы вызовем окно установки его параметров. Необходимо отметить, что, щелкая правой кнопкой мыши при нажатой левой, УГО поворачиваются на 90° по часовой стрелке.

При установке УГО элемента на место открывается довольно обширное окно параметров этого элемента. Но для первых шагов нам потребуется установить в строке **PART** обозначение элемента в цепи (обычно это делается автоматически с каждым вводом подобного элемента) и его номинал. Все значения для выделяемых параметров элемента вводятся в окне **Value**. Так, номинал элемента вводится в окне **Value** при выделенной строке с названием элемента и затем дублируется в этой строке через знак равенства.

Значения компонентов задаются либо непосредственно (2600), либо в показательной форме (2.6E3), либо условными буквенными обозначениями (2.6K).

Следует обратить внимание, что в MicroCap:

– целая часть чисел отделяется от дробной не запятой, а точкой, например 1.3K или 1.3E3;

– буквенные обозначения следует вводить в английском алфавите.

При желании можно выбрать вид УГО в окне **Value** при выделенной строке **Shapegroup**, поскольку по умолчанию вызываются УГО по американским

стандартам. Для этого нужно просто вызвать в окне **Value** возможные УГО. Особенно это рекомендуется при вводе УГО резистора, где выбираем *euro* вместо *default*.

Остальные параметры нас пока не интересуют. При нажатии кнопки **ОК** окно параметров исчезает и элемент со своими атрибутами оказывается в окне редактора.

Задание 1. Попробуйте ввести следующие элементы: сопротивление R = 1 кОм, емкость конденсатора C = 1 мкФ, индуктивность L = 1 мГн (рис. 21).

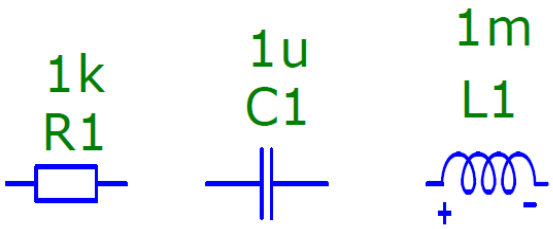


Рис. 21. Базовые элементы

Отметим, что рядом с УГО элемента появились и его атрибуты. Этого можно избежать, если при в ведении параметра отменить его показ (снять флажок *show* рядом с окном **Value**). Перетаскивать УГО и его атрибуты можно нажатой левой кнопкой мыши, но для этого следует перейти в режим **Select mode**, (кнопка ).

Соединения элементов в цепь производятся с помощью проводников, которые вызываются через кнопки с их пиктограммами (кнопки  и  на второй инструментальной панели).

Задание 2. Составьте простейшую схему с параллельным соединением двух элементов (рис. 22).

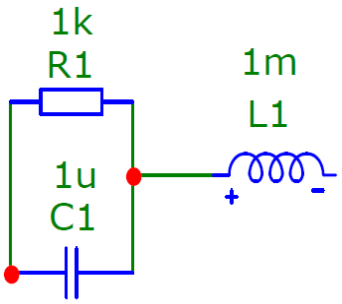


Рис. 22. Соединение двух элементов

Отметим, что:

– пересечение проводников с замыканием обозначается красной точкой;

– при перемещении элементов проводники не отсоединяются.

Для анализа электрической цепи нам нужно знать, как изменяются её энергетические характеристики. Но для этого в цепь нужно вставить источник энергии (сигналов).

В окне **Component/Analog primitives** можно найти пять групп источников.

Для знакомства выберем такой хорошо известный источник постоянного тока, как батарейка. Для этого находим **Component/Analog primitives/Waveform Sources/Battery** или на инструментальной панели соответствующую пиктограмму и переносим этот элемент в окно редактора. В окне параметров нужно ввести его значение в вольтах, и желательно изменить УГО на *euro*.

Задание 3. Изобразите электрическую цепь (схему на рис. 23) с источником постоянного тока 5 В и тремя резисторами по 1 кОм, два из которых включены параллельно.

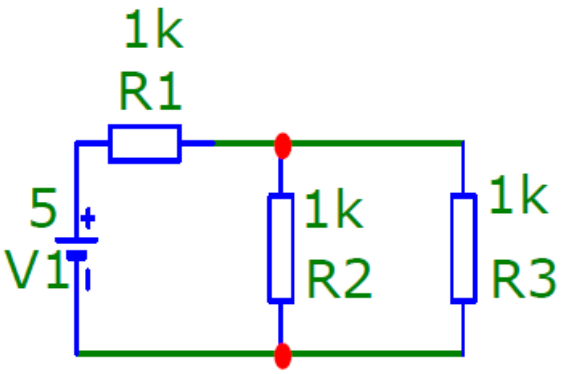


Рис. 23. Электрическая цепь

Для проведения анализа нам необходимо разметить схему.

Прежде всего, нужно обозначить узел с потенциалом, равным 0, т. е. обозначить точку заземления (рис. 24). Для этого на инструментальной панели находим пиктограмму заземления и подсоединяем этот элемент к выбранному узлу схемы. То же самое можно сделать через окно **Component/Analog primitives/Connectors/Ground**.

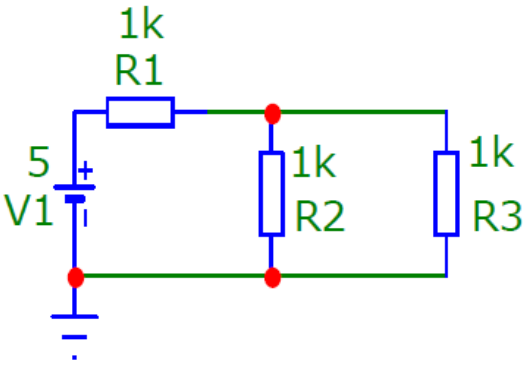


Рис. 24. Заземление схемы

Далее выведем номера узлов (рис. 25). Для этого включаем кнопку  .

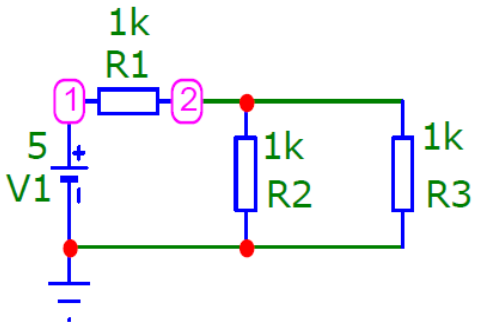


Рис. 25. Вывод узлов

А теперь можно приступать к анализу. В окне **Analysis** можно увидеть много видов анализа электрической цепи.

– Transient Analysis…(Alt+1) – анализ переходных процессов.

– AC Analysis…(Alt+2) – анализ частотных характеристик.

– DC Analysis…(Alt+3) – анализ передаточных функций по постоянному току.

– Probe Transient Analysis… – анализ переходных процессов и отображение их результатов в режиме Probe.

– Probe AC Analysis… – анализ частотных характеристик и отображение их результатов в режиме Probe.

– Probe DC Analysis… – анализ передаточных функций по постоянному току и отображение их результатов в режиме Probe.

Для первого знакомства выберем анализ по постоянному току (*DC…*).

В режиме *DC...* рассчитываются передаточные характеристики по постоянному току. К входам цепи подключаются один или два независимых источника постоянного напряжения или тока. В качестве выходного сигнала может рассматриваться разность узловых потенциалов или ток через ветвь, в которую включен резистор. При расчете режима *DC...* программа закорачивает индуктивности, исключает из схемы конденсаторы и затем рассчитывает режим по постоянному току при нескольких значениях входных сигналов.

После перехода в режим *DC...* программа проверяет правильность построения схемы. При отсутствии ошибок составляется топологическое описание, выполняется подготовка к численному расчету нелинейных уравнений итерационным методом Ньютона – Рафсона и открывается окно задания параметров моделирования *DC Analysis Limits*.

Прежде, чем начинать любой анализ, оцените поведение схемы в грубых приближениях. Это позволит провести анализ цепи более осмысленно и выявить те грубые ошибки, которые могут быть допущены, но не выявлены программой.

В нашей схеме мы видим два делителя:

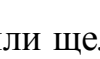
– **напряжения** в отношении 1:3 между резисторами R1=1k и параллельно соединёнными резисторами R2, R3 с общим сопротивлением R0=R1*R2/R1+ R2=0.5k; при этом напряжение узла 2 можно найти по правилу делителя напряжения **V2=V1*R0/R1+R0**. Для нашей схемы при V1=10В напряжение V2=10/3=3,3;

– **тока** в отношении 1:2 между резисторами R1 и R2 или R3; при этом ток в ветви элемента R2(R3) можно найти из правила делителя тока **I2*R2=I3*-R3=V2**. Отсюда I1= 6,6/1000=6,6 мА, а I2=6,6/2= 3,3 мА.

Итак, щелкнув левой кнопкой мыши команду **Analysis/DC...,** мы вызовем щелкнув нужное диалоговое окно **DC Analysis Limits**. Познакомимся с ним подробно.

В этом окне можно установить пределы изменения переменных, вид выводимых графиков и собственно команду **Run**, которая позволяет машине начать анализ. Всё окно разделено на пять областей: кнопки управления, числовые значения, представление графиков, выражения и дополнительные функции (опции).

Кнопки управления

Run: по этой команде начинается анализ схемы. Эту команду можно вызвать также через кнопку F2 на клавиатуре или щелкнув по кнопке  на появляющейся вместе с графиками новой инструментальной панели. Моделирование может быть остановлено в любой момент времени нажатием на пиктограмму  или клавишу **Esc**. Последовательные нажатия на пиктограмму  прерывают и затем продолжают моделирование.

Add: добавление еще одной строки спецификации вывода результатов после строки, отмеченной курсором. На этой строке устанавливаются способ отображения результатов и аналитические выражения для построения графиков. При наличии большого количества строк, не помещающихся на экране, появляется линейка прокрутки.

Delete: удаление выделенной курсором строки в поле представления графиков.

Expand: по этой команде выводится окно, позволяющее расширить поле записей в области представления графиков. Для этого курсором нужно выделить соответствующее текстовое поле.

Stepping: по этой команде переходят в диалоговое окно **Stepping**, которое позволяет выводить данные для нескольких пошагово меняющихся значений элементов схемы.

Properties: по этой команде переходят в диалоговое окно **Properties dialog box**, которое позволяет управлять окном вывода графиков и видом самих графиков.

Help: по этой команде переходят к файлам Help topic для диалогового окна **DC Analysis Limits**.

Числовые параметры

– Variable 1 – задание первой изменяемой переменной. В качестве переменной используется напряжение источника постоянного тока вне зависимости от его первоначально установленных параметров. Таких источников в схеме может быть два. В графе **Method** выбирается метод изменения напряжения источника (переменной).

– **Auto** – шаг расчёта выбирается автоматически с целью достижения изменения выходного параметра (в %) не более, чем указано в позиции **Maximum Change**. Пределы расчёта (<конец>, <начало>) по умолчанию устанавливаются <10> и <0>, но можно установить свои. Это очень полезно при непредсказуемо изменяющихся выходных значениях, так как позволяет рисовать плавные аналоговые кривые.

– **Linear** – линейный масштаб, задаваемый в графе **Range** по формату <конец>, <начало>, <шаг>. Если опустить параметр **Step** (шаг), то шаг будет принят равным (<конец> – <начало>)/<50>. Если опустить параметр <начало>, то начальное значение будет приравнено к нулю.

– **Log** и **List** – в демоверсии не работают.

В графе **Name** указывается имя варьируемой переменной. Это могут быть: величины источника постоянного напряжения или тока; температура; значения одного из параметров модели компонентов, имеющих математические модели; значения символической переменной (определенной директивой .Define). Имя варьируемой переменной может выбираться из вложенного в окно списка.

В графе **Range** указываются диапазон изменения варьируемой переменной и шаг, зависящий от метода изменения переменной.

Строка **Variable 2** определяет поля **Method**, **Name** и **Range** для второй варьируемой переменной. Для значений, указываемых в этих полях, используются те же правила, что и перечисленные выше для переменной *Variable 1*, за исключением опций в списке **Method**. Здесь исключена опция **Auto**, но появилась и дополнительная опция **None**, выбираемая в том случае, если изменяется только одна переменная. По умолчанию шаг **Step** принимается равным (<конец> – <начало>)/10. Каждое значение переменной *Variable* 2 приводит к построению отдельного графика. Метод двух источников очень удобен для построения семейства вольт-амперных характеристик транзисторов.

Temperature – диапазон изменения температуры в градусах Цельсия.

Number of Points – количество точек данных, по которым осуществляется интерполяция при построении графиков, или количество строк в таблице вывода результатов (*numeric output*). По умолчанию устанавливается равным 51.

Maximum change, %. Действует только при выборе метода *Auto* изменения переменной. Представляет собой максимально допустимое приращение графика первой функции на одном шаге (в процентах от полной шкалы). Если график функции изменяется быстрее, то шаг приращения первой переменной автоматически уменьшается.

Опции окна DC Analysis Limits

Run Options – управление выдачей результатов расчетов выставленным по умолчанию значением Norma. При этом результаты расчетов не будут сохраняться на диске.

Auto Scale Ranges – присвоение признака автоматического масштабирования **Auto** по осям X, Y для каждого нового варианта расчетов. Если эта опция выключена, то принимаются во внимание масштабы, указанные в графах *X Range*, *Y Range*.

Accumulate plots – позволяет аккумулировать результаты расчётов на одном графике при редактировании схемы.

Параметры вывода результатов моделирования

Слева внизу мы видим группу пиктограмм. Нажатие каждой пиктограммы определяет характер вывода данных, задаваемых в той же строке. Имеются следующие возможности:

–  – переключение между логарифмической и линейной шкалами по оси X. При выборе логарифмической шкалы диапазон изменения переменной должен быть положительным;

–  – переключение между логарифмической и линейной шкалами по оси Y. При выборе логарифмической шкалы диапазон изменения переменной должен быть положительным;

–  – вызов меню для выбора одного из 64 цветов для окрашивания графиков. График окрашивается в цвет кнопки. Удобно выводить на одном графике несколько параметров;

–  – при нажатии этой кнопки в текстовый выходной файл заносится таблица отсчетов функции, заданной в графе Y Expression. Число строк в таблице задается параметром **Number of Points** в разделе **Числовые параметры**.

Page – указываются номера (наименование) окон (страниц), на которые выводятся графики.

Plot Group – в графе *P* числом от 1 до 9 указывается номер графического окна, в котором должна быть построена данная функция. Все функции, помеченные одним и тем же номером, выводятся в одном окне (на одной странице). Если это поле пусто, график функции не строится.

Форматы выражений для DC-анализа

Поля *X Expression* (математическое выражение переменной, откладываемой по оси X) и Y *Expression* (математическое выражение переменной, откладываемой по оси Y) используются для спецификации масштабов и переменных, откладываемых по горизонтальной (X) и вертикальной (Y) осям.

Программа даёт широкий выбор возможностей представления переменных, откладываемых по осям графиков: от конкретных токов и напряжений до сложных формул с их участием. В этом можно убедиться, щелкнув правой кнопкой мыши в этих полях. Но пока ограничимся выводом на оси X, Y напряжений в узлах – v (<номер узла>) или между узлами – v (<номер одного узла>, <номер другого узла>), а также токов в ветвях (между узлами) – i (<номер одного узла>, <номер другого узла>). По оси X по умолчанию выводится напряжение источника, которое, как мы знаем, в DC…-анализе является переменной величиной *Variable 1*.

Поля *X Range* (масштаб по горизонтальной оси X) и Y *Range* (масштаб по вертикальной оси Y) имеют несколько возможностей, которые можно вызвать, щелкнув правой кнопкой мыши в одном из этих полей.

Auto – приводит к однократному автоматическому масштабированию по соответствующей оси и заполнению полей полученными значениями масштабов.

Установка флажка *Auto Scale Ranges* приводит к автоматическому расчету масштабов всех графиков по всем осям при каждом повторении расчетов и соответствующему обновлению полей *X Range* и *Y Range*.

Auto always – приводит к постоянному автоматическому масштабированию по соответствующей оси.

Можно ввести свои пределы значений переменных по осям в формате <high>, <low>. Формат <low> по умолчанию устанавливается в нулевое значение.

Меню режимов расчета передаточных функций DC

После перехода в режим расчета передаточных функций в строке меню появляется новое меню DC, содержащее пункты RUN, Limits, Stepping, Exit, State Variables Editor, OPTIMIZE, Watch, Breakpoints, 3D Windows, Numeric Output, Reduce Data Points. Состав этих команд одинаков для всех видов анализа.

Вернёмся к нашей схеме. Выбрав режим расчёта передаточных функций **DC...**, в окне **DC Analysis Limits** определим, что переменная **Variable 1** будет меняться в режиме *Auto* и это будет напряжение в узле 1 (V(1)), т. е. напряжение источника. Оно будет меняться в пределах от 0 до 10 В с шагом 0,5 В. Второй переменной у нас нет, так как нет второго источника. Изменение температуры нас не интересует, так же как и 5 % в ***Maximum change*** нас вполне устроит. В *Run Options* нас также устроит значение Normal.

Выведем четыре графика: для напряжений в узлах 1 и 2 {**v(1), v(2)**} и токов через резисторы R1, R2 {**i(1,2)** или **i(R1)**, **i(2,0)** или **i(R2)**}. Это мы укажем в разделе Y *Expression*. По оси X (*X Expression*) выведем изменяющееся напряжение источника (DCINPUT1 или v(1)).

Все графики выполняем в линейном масштабе по осям X и Y. Желательно придать им разные цвета. Предлагается вывести графики напряжения на первый лист (page 1), а графики токов – на второй лист (page 2). При этом на каждом листе обе зависимости расположить на одном графике (в разделе P везде указать 1).

В разделах *X Range* и *Y Range* везде укажем *Auto always*, так как другие варианты нас пока не интересуют.

Проверив ещё раз введённые данные в окне **DC Analysis Limits**, щелкаем по кнопке **Run**.

На экране открывается окно (рис. 26) с графиком зависимости напряжений **v(1), v(2)** от **v(1)**.

На нижней линейке можно увидеть значки 1 и 2. Это наши страницы.

Переходим на страницу 2 и получаем (рис. 27) графики зависимости токов **i(1,2)** и **i(R2)** от **v(1)**.

Простое рассмотрение полученных графиков подтверждает наши расчёты. Подводя курсор к любой точке графика в выпадающем окне, можно увидеть числовые данные, относящиеся к данной точке.

Обратите внимание на появившуюся вторую инструментальную линейку, которая содержит очень много функций для работы с графиками. Например, очень полезной является функция курсора  (можно также вызвать клавишей F8). Она позволяет вызвать окно с предельными числовыми значениями, а также, щелкнув левую и правую кнопки мыши, можно вызвать левую и правую динамическую курсорную вертикаль.

Они очень удобно передвигаются при нажатой соответствующей кнопке мыши. При этом соответственно меняются показания в разделе числовых данных (рис. 27).

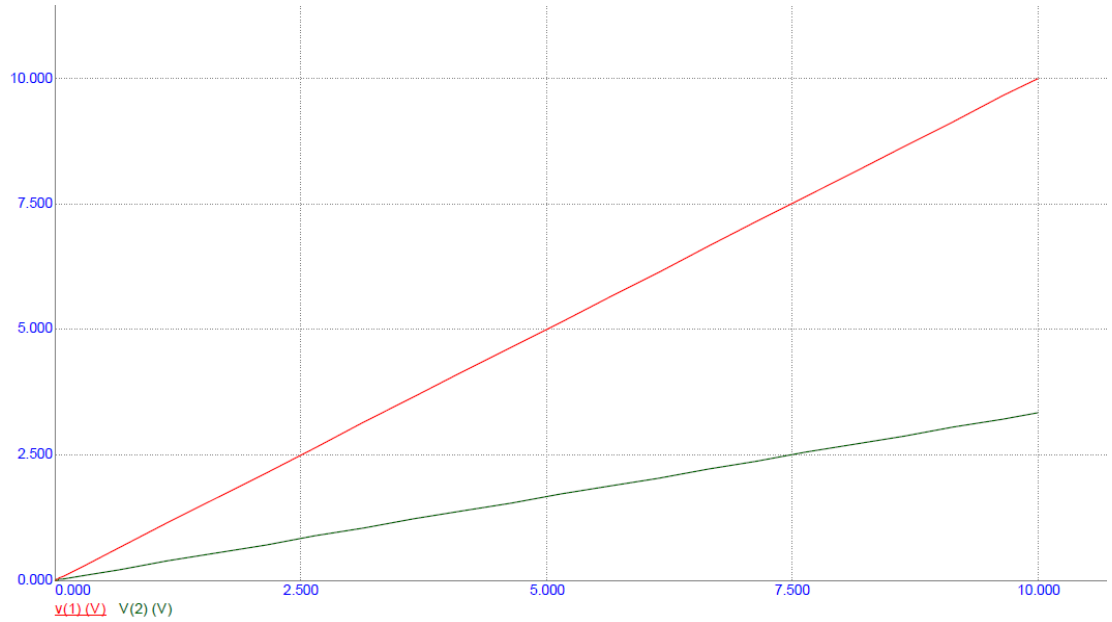


Рис. 26. График зависимости напряжений

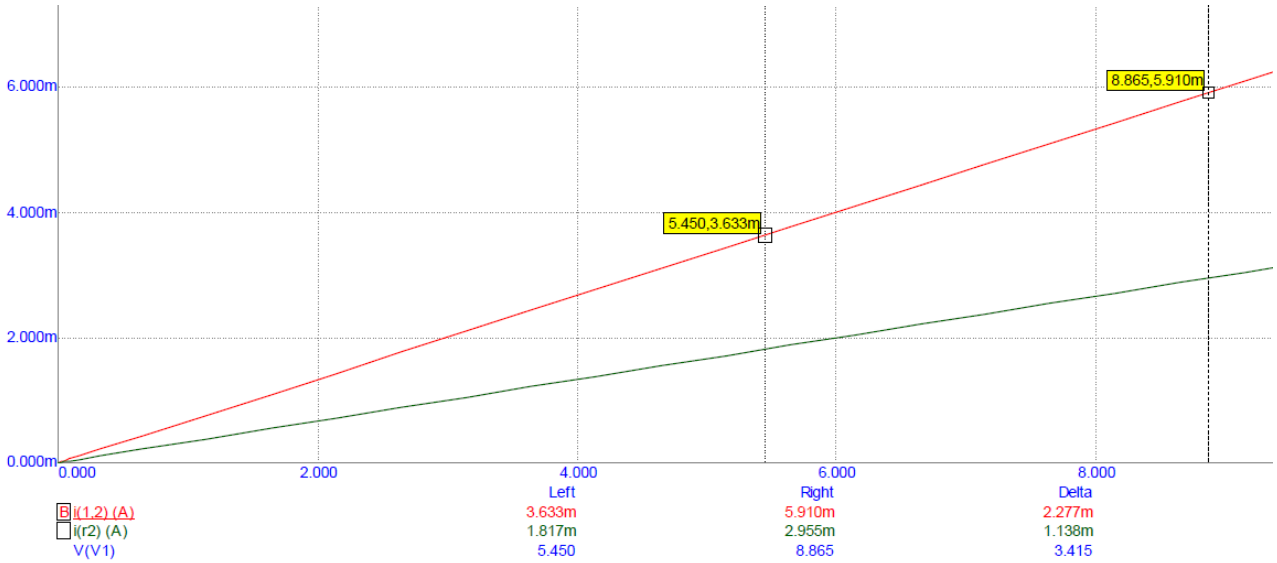


Рис. 27. График зависимости токов

Контрольные вопросы к разделу 2.6

1. Создание новой схемы в MicroCap. Загрузка, сохранение, редактирование схем.

2. Описание основных разделов в командном меню (Windows, Options, Component, Analysis).

3. Описание окна задания параметров моделирования в MicroCap.

4. Панели инструментов в MicroCap.

5. Назначение и возможности систем схемотехнического моделирования.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Амелина, М. А. Программа схемотехнического моделирования Micro-Cap 8 [Текст] / М. А. Амелина, С. А. Амелин. – М. : Горячая линия – Телеком, 2007. – 464 с.

2. Аристов, В. В. Программное управление автоматизированными производственными системами [Текст] : конспект лекций / В. В. Аристов. – Омск : ОмГТУ, 2009. – 98 с.

3. Бекаревич, Ю. Б. Самоучитель Access 2010 [Текст] / Ю. Б. Бекаревич, Н. В. Пушкина. – СПб. : БХВ-Петербург, 2011. – 432 с.

4. Белик, А. Г. Теория и технология программирования [Текст] : конспект лекций / А. Г. Белик, В. Н. Цыганенко. – Омск : Изд-во ОмГТУ, 2013. – 85 с.

5. Верма, Р. Д. Справочник по функциям Win32 API [Текст] / Р. Д. Верма. – М. : Горячая линия – Телеком, 2005. – 551 с.

6. ГОСТ 7.1–2003. Библиографическая запись. Общие требования и правила составления [Текст]. – М. : ИПК Изд-во стандартов, 2004. – 47 с.

7. ГОСТ 7.32–2001. Отчёт о научно-исследовательской работе. Структура и правила оформления [Текст]. – М. : ИПК Изд-во стандартов, 2001. – 16 с.

8. Гуменюк, А. С. Информатика [Текст] : учеб. пособие / А. С. Гуменюк, И. В. Потапов. – Омск : Изд-во ОмГМА, 2012. – 175 с.

9. Задорожный, В. Н. Информатика [Текст] : конспект лекций / В. Н. Задорожный, О. Н. Канева. – Омск : Изд-во ОмГТУ, 2005. – Ч. 1. – 43 с.

10. Информатика [Текст] : метод. указания к лаб. работам / сост. : В. Н. Задорожный, О. Н. Канева. – Омск : Изд-во ОмГТУ, 2005. – 54 с.

11. Илюшечкин, В. М. Основы использования и проектирования баз данных [Электронный ресурс] : учеб. пособие для вузов по направлению «Информатика и вычислительная техника» / В. М. Илюшечкин. – М. : Юрайт, 2011. – 1 эл. опт. диск (DVD-ROM).

12. Крылов, Е. В. Техника разработки программ [Текст] : учеб. для вузов по направлениям «Информатика и вычислительная техника» и «Техника и технологии» : в 2 кн. Кн. 2 : Технология, надежность и качество программного обеспечения. – М. : Высш. шк., 2008. – 468 с.

13. Малков, О. Б. Базы данных [Текст] : метод. указания к выполнению лаб. работ / О. Б. Малков, Е. Т. Гегечкори. – Омск : Изд-во ОмГТУ, 2007. – 112 с.

14. Малков, О. Б. Базы данных [Текст] : учеб. пособие для студентов заочной формы обучения / О. Б. Малков, Е. Т. Гегечкори. – Омск : Изд-во ОмГТУ, 2007. – 64 с.

15. Малков, О. Б. Работа с СУБД MySQL [Текст] : учеб. пособие / О. Б. Малков, М. В. Девятерикова. – Омск : Изд-во ОмГТУ, 2010. – 84 с.

16. Морозов, М. А. Информационные технологии в социально-культурном сервисе и туризме. Оргтехника [Текст] : учебник для студ. высш. учеб. заведений / М. А. Морозов, Н. С. Морозова. – М. : Издательский центр «Академия», 2008. – 240 с.

17. Никонов, А. В. Электротехника и электроника [Электронный ресурс] : конспект лекций / А. В. Никонов. – Омск : Изд-во ОмГТУ, 2014. – 1 эл. опт. диск (CD-ROM).

18. Новожилов, О. П. Архитектура ЭВМ и систем [Электронный ресурс] : учеб. пособие для бакалавров вузов по направлению 230100 «Информатика и вычислительная техника» / О. П. Новожилов. – М. : Юрайт, 2012. – 1 эл. опт. диск (CD-ROM).

19. Павловская, Т. А. C/C++. Программирование на языке высокого уровня [Текст] : учебник / Т. А. Павловская. – СПб. : Питер, 2002. – 464 с.

20. Павловская, Т. А. C#. Программирование на языке высокого уровня [Текст] : учебник для вузов / Т. А. Павловская. – СПб. : Питер, 2009. – 432 с.

21. Пирогов, В. Ю. Ассемблер для Windows [Текст] / В. Ю. Пирогов. – СПб. : БХВ-Петербург, 2005. – 864 с.

22. Пирогов, В. Ю. Ассемблер на примерах [Текст] / В. Ю. Пирогов. – СПб. : БХВ-Петербург, 2005. – 416 с.

23. Силаенков, А. Н. Информационные технологии [Электронный ресурс] : учеб. пособие / А. Н. Силаенков. – Омск : Изд-во ОмГТУ, 2014. – 1 эл. опт. диск (CD-ROM).

ПРИЛОЖЕНИЕ А
(обязательное)

Образец оформления титульного листа

Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Омский государственный технический университет»

Факультет информационных технологий и компьютерных систем

Кафедра «Автоматизированные системы обработки информации и управления»

ОТЧЕТ
ПО ПРАКТИЧЕСКИМ ЗАНЯТИЯМ
по дисциплине «Рабочая профессия»

Выполнил:
студент гр. ИВТ – 134П
Иванов И.И.

_____
(подпись, дата)

Принял:
канд. техн. наук, доцент
Белик А.Г.

_____
(подпись, дата)

Омск – 2015

ПРИЛОЖЕНИЕ Б

(обязательное)

Образцы оформления структурных элементов

Реферат

Пояснительная записка 90 с., 27 рис., 4 табл., 23 источника, 2 прил.

ВЕБ-ПРОГРАММИРОВАНИЕ, КЛИЕНТСКОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ, ВЕБ-СИСТЕМА, HTML-ДОКУМЕНТ, ЯЗЫК ПРОГРАММИРОВАНИЯ ACTIONSCRIPT, ADOBE FLASH

Объектом исследования является технология Adobe Flash.

Цель работы – анализ основных направлений (сценариев) веб-программирования и определение в них места технологии Adobe Flash и языка программирования ActionScript 3.0.

В процессе работы проводился анализ двух основных сценариев веб-программирования – веб-программирования на основе пассивной клиентской части и веб-программирования на основе активного содержимого html-документов. Показано, когда использование второго сценария веб-программирования наиболее целесообразно, как он связан с первым сценарием и как он реализуется на основе технологии Adobe Flash и языка программирования ActionScript 3.0.

В заключении приведены критерии целесообразности выбора технологии Adobe Flash и языка программирования ActionScript 3.0 при разработке веб-приложений.

Степень внедрения – для учебных целей.

Обозначения, определения и сокращения

Обозначения	
Мб.	Мегабайт
См.	Смотри
Определения	
Макрос	Это последовательность команд и функций
Операционная система	Это комплекс взаимосвязанных программ, предназначенных для управления ресурсами компьютера и организации взаимодействия с пользователем
Сокращения	
БД	Базы данных.
ПК	Персональный компьютер