Содержание

Изм.

Лист

№ докум.

Подпись

Дата

Лист

1

ОПП.01.09.02.03.2018.16.00.00.ПЗ

Разраб.

Турченко К. А.

Провер.

Барилова С.В.

Реценз.

Н. Контр.

Утверд.

**Отчет по производственной практике**

Лит.

Листов

40

Группа 3-П-2

1. [Техника безопасности 2](#_Toc517909897)

2. Инструменты средств проектирования

[3 Программа база данных «Расходы» 26](#_Toc517909899)

[Заключение 39](#_Toc517909900)

[Список литературы 40](#_Toc517909901)

1. Техника безопасности при работе с компьютером

Компьютерная техника в настоящее время используется практически во всех организациях. Применение работниками данного оборудования способно оказывать негативное влияние на их здоровье и даже привести к чрезвычайным ситуациям и несчастным случаям на производстве. Для того, чтобы этого избежать, работники должны соблюдать технику безопасности при работе с компьютером.

Негативные факторы при работе за компьютером:

* Работник, использующий в своей трудовой деятельности компьютер, может испытывать на себе влияние следующих негативных факторов:
* Электромагнитное и инфракрасное излучения;
* Шум работающего компьютера (или нескольких компьютеров);
* Риск поражения электрическим током в случае замыкания;
* Возможность возникновения возгорания.

Правила работы за компьютером:

Статья 212 ТК РФ обязывает работодателя разработать и утвердить инструкции по охране труда и обеспечить ознакомление с ними работников. Таким образом, в организации может быть утвержден специальный документ, предусматривающий правила выполнения трудовых операций с компьютерной техникой, который может называться, например, инструкцией по технике безопасности при работе за компьютером. Отдельные положения по данному вопросу могут содержаться в инструкции по охране труда для офисных работников.

В некоторых отраслях хозяйственной деятельности утверждаются типовые инструкции по охране труда. Например, Приказом Минсвязи РФ от 02.07.2001 N 162 утверждена Инструкция по охране труда при работе на персональном компьютере ТОИ Р-45-084-01.

Правила безопасности по каждому этапу работы с компьютером:

1. До начала работы: проверить исправность электропроводки, розеток и вилок компьютера, заземление ПК.

2. Во время работы:

* Необходимо аккуратно обращаться с проводами;
* Запрещается работать с неисправным компьютером;
* Нельзя заниматься очисткой компьютера, когда он находится под напряжением;
* Недопустимо самостоятельно проводить ремонт оборудования при отсутствии специальных навыков;
* Нельзя располагать рядом с компьютером жидкости, а также работать с мокрыми руками;
* Нельзя в процессе работы с ПК прикасаться к другим металлическим конструкциям (например, батареям);
* Не допускается курение и употребление пищи в непосредственной близости с ПК и др.

3. В аварийных ситуациях:

* При любых неполадках необходимо сразу отсоединить ПК от сети;
* В случае обнаружения оголенного провода незамедлительно оповестить всех работников и исключить контакт с проводом;
* В случае возникновения пожара принять меры по его тушению с использованием огнетушителей (работники должны знать, где они находятся);
* В случае поражения человека током оказать первую помощь и вызвать скорую медицинскую помощь.

4. По окончании работы:

* Выключить компьютер;
* Желательно провести влажную уборку рабочего места;
* Отключить электропитание.

Организация рабочего места:

Постоянная работа за компьютером вызывает отклонения в здоровье работника, в частности:

* Нагрузка на зрение приводит к его ухудшению, покраснениям глаз, возникновению «синдрома сухого глаза»;
* Несоблюдение нормативов организации рабочего места может привести к искривлению позвоночника, заболеваниям суставов и болям различного характера;
* Длительная концентрация внимания на экране вызывает переутомление.

Требования, предъявляемые СанПиН к рабочему месту пользователя ПК:

* Расстояние от монитора до глаз должно составлять от 600 до 700 мм, но не меньше 500;
* Стул работника должен быть регулируемым по высоте и обеспечивать возможность поворота и изменения позы во время работы;
* Высота стола – от 680 до 800 мм;
* Поверхность стола должна позволять оптимально разместить на ней все необходимое для работы и др.
* Перерывы в работе за компьютером

С целью избежать переутомления работника, СанПиН рекомендуют делать перерывы длительностью от 10 до 15 минут после 45 — 60 минут работы. Во время перерыва работнику следует выполнять гимнастику для глаз и физические упражнения.

Соблюдение правил работы за компьютером позволит снизить негативное воздействие компьютера на здоровье работника.

1. Инструменты средств проектирования

1. Инструменты программной инженерии (Software Engineering Tools)

Первые пять тем данной секции соответствуют первым пяти областям знаний SWEBOK - требования, проектирование, конструирование, тестирование и сопровождение. Следующие четыре темы касаются оставшихся четырех областей знаний – конфигурационного управления, управления программной инженерией, процессов и качества, соответственно. Также в данной секции представлена еще одна тема – “Дополнительные аспекты инструментального обеспечения” (в оригинале SWEBOK она называется “Miscellaneous”), посвященная таким вопросам как, например, интеграция инструментов, которые потенциально касаются всех классов инструментов.

1.1 Инструменты работы с требованиями (Software Requirements Tools)

SWEBOK говорит о том, что инструменты, применяемые для работы с требованиями могут быть классифицированы в две категории: средства моделирования (modeling) и средства трассировки (traceability). Однако, на практике, моделирование требований, все же, является частью управления требований, как, кстати, и трассировка. В принципе, инструменты трассировки могут быть рассмотрены как самостоятельная категория, в силу своей значимости при проведении анализа требований, в первую очередь, анализа влияний требований и изменений (т.н. “impact analysis”). Но моделирование требований лишь часть управления требованиями. Поэтому, в приведенной ниже классификации предлагаемая модификация оригинального SWEBOK состоит в том, что вместо “инструментов моделирования требований” используется термин “инструменты управления требованиями”, при сохранении оригинального содержания данной темы SWEBOK. Соответственно,

Инструменты управления требованиями (моделирования требований – Requirements modeling tools). Эти инструменты используются для извлечения (eliciting), анализа, специфицирования и проверки программных требований.

Инструменты трассировки требований (Requirement traceability tools). Эти инструменты становятся все более важными по мере повышения сложности программного обеспечения. В силе того, что они также относятся и к другим процессам жизненного цикла, здесь они представлены в качестве самостоятельной категории средств работы с требованиями.

Необходимо заметить, что трассировка является неотъемлемой частью полноценной работы с требованиями, что приводит к естественному объединению предлагаемых SWEBOK категорий инструментов в единый класс “инструментов управления требованиями”, функциональное содержание которых может варьироваться, например, в зависимости от сложности проектов и уровня зрелости процессов. Если мы обратимся, например, к модели CMMI Staged, мы увидим, что на 2-м уровне зрелости речь идет об “управлении требованиями” – Requirement Management, а на 3-м уровне зрелости обсуждается “разработка требований” – Requirement Development, обладающая более ёмким содержанием. В то же самое время, с технократической точки зрения, требования могут восприниматься и как элементы конфигураций, наравне с запросами на изменения и другими активами проекта (см. область знаний SWEBOK “Конфигурационное управление”). Таким образом, в ряде случаев (что подтверждается конкретными программными средствами, доступными на рынке программного обеспечения), в качестве инструмента работы с требованиями может выступать и система конфигурационного управления, если, конечно, она изначально не ограничена базовой функциональностью контроля версий <файлов>. С другой стороны, сегодняшние средства моделирования на основе UML и BPMN могут также рассматриваться как элементы инструментального обеспечения работы с требованиями, что часто отражается в их функциональности, включающей тесную интеграцию с “классическими” средствами управления требованиями, а сама интеграция воплощена не только в визуальном представлении работы с репозиториями требований, но и в автоматизации трассировки между моделями (и/или их элементами) и требованиями, соответственно.

1.2 Инструменты проектирования (Software Design Tools)

Эта тема охватывает инструменты для создания и проверки программного дизайна. Существует большое разнообразие таких инструментов, использующих различные нотации (соглашения, в том числе визуальные) и методы. Несмотря на такое разнообразие, (авторами SWEBOK) не было найдено адекватной классификации этих инструментов.

Однако, в данном случае, все же возможно разделение инструментов по нескольким критериям, например, применяемым базовым нотациям моделирования и проектирования (SADT/IDEF, UML, BPMN/BPEL, Microsoft DSL и т.п.) или целевым задачам (бизнес-моделирование, проектирование БД, объектно-ориентированное проектирование, интеграционное/SOA-проектирование и т.п.).

1.3 Инструменты конструирования (Software Construction Tools)

Данная тема касается инструментальных средств конструирования программного обеспечения, в соответствии с пониманием “конструирования”, заданным соответствующей областью знаний SWEBOK, рассматривавшейся ранее. Эти инструменты используются для производства и трансляции программного представления (например, исходного кода), достаточно детального и явного для машинного выполнения.

Редакторы (program editors). Эти инструменты используются для создания и модификации исходного кода программ и, возможно, ассоциированной с ними документации. Это могут быть как редакторы “общего назначения” (что на протяжении многих лет наблюдается в UNIX и unix-подобных средах) или специализированные редакторы с поддержкой специфики целевого языка программирования (что является, в большинстве случаев, прерогативой интегрированных сред разработки – IDE). В то же время, документирование все же является не только и не столько частью редактора, сколько самостоятельной функциональностью, пусть часто и тесно интегрированной с редактором.

Компиляторы и генераторы кода (compilers and code generators). Традиционно, компиляторы являлись неинтерактивными (командными) трансляторами исходного кода. Однако, существует тенденция (с точки зрения автора, более чем явная, что отмечено ниже) интеграции компиляторов и редакторов в интегрированные среды программирования. К этому классу также относятся препроцессоры, линковщики/загрузчики, а также генераторы кода (за исключением, может быть, объектно-ориентированных средств проектирования, поддерживающих связь с исходным кодом и имеющих тенденцию быть тесно интегрированными с новым поколением IDE).

Интерпретаторы (interpreters). Эти инструменты обеспечивают исполнение программ посредством эмуляции. Они могут поддерживать действия по конструированию программного обеспечения, предоставляя для исполнения программ окружение, более контролируемое и поддающееся наблюдению, чем это обычно способна сделать та или иная операционная система. Хочется отметить определенное “слияние”, если так можно выразиться, между компиляторами и интерпретаторами. Ярким тому свидетельством является использование так называемой just-in-time компиляции – компиляции “на лету”, когда промежуточный программный код, по мере исполнения или с опережением (например, в процессе запуска/загрузки программы) преобразуется в набор инструкций, исполняемых непосредственно средствами операционной системы, но под контролем среды исполнения, в первую очередь, с точки зрения безопасности. Такого рода подход стал родоначальником ряда современных программных платформ, например, Java и .NET. На этом фоне, возможно было бы объединить интерпретаторы с компиляторами и генераторами кода, как средства непосредственной подготовки (трансляции) исходного кода к исполнению.

Отладчики (debuggers). Эти инструменты было принято выделить в самостоятельную категорию, так как они поддерживают процесс конструирования программного обеспечения, но, в то же время, функционально отличаются от редакторов и компиляторов.

С точки зрения классификации инструментов необходимо выделить явно и давно присутствующие на рынке: “интегрированные средства разработки” (IDE - integrated developers environment), а также программные библиотеки/библиотеки компонент (frameworks, libraries, components), без которых просто невозможно представить сегодняшний процесс разработки, да и рынок программных средств, в целом. Кроме того, в данной теме можно говорить и о таких функционально ёмких “супер”-категориях, как «программная платформа» (например, Java, Java EE 8 и Microsoft .NET) и “платформа облачных вычислений”(например, Microsoft Azure, Amazon и др.), которые включают наравне с инструментами, как таковыми, и определенные модели конструирования, преобразования и выполнения кода. При таком подходе, вероятно, обоснованным было бы введение класса “элементарных” или “базовых инструментов конструирования”, к которому можно было бы отнести редакторы, компиляторы, интерпретаторы, отладчики, средства документирования и библиотеки, а также класса “комплексных средств конструирования” – интегрированных сред и различных платформ, что, безусловно, не претендует на истину в последней инстанции и является одной из возможных точек зрения.

1.4 Инструменты тестирования (Software Testing Tools)

Генераторы тестов (test generators). Эти инструменты помогают в разработке сценариев тестирования.

Средства выполнения тестов (test execution frameworks). Эти средства обеспечивают среду исполнения тестовых сценариев в контролируемом окружении, позволяющем отслеживать поведение объекта, подвергаемого тестированию.

Инструменты оценки тестов (test evaluation tools). Эти инструменты поддерживают оценку результатов выполнения тестов, помогая определить в какой степени и где именно обнаруженное поведение <тестируемого объекта> соответствует ожидаемому поведению.

Средства управления тестами (test management tools). Эти средства обеспечивают поддержку всех аспектов процесса тестирования программного обеспечения.

Инструменты анализа производительности (performance analysis tools). Эти инструменты используются для количественной оценки и анализа производительности программного обеспечения, являющегося специализированным видом тестирования, цель которого – в оценки поведения программ в части производительности, в отличие от тестирования <корректности> функционального поведения.

Последний класс инструментов тестирования, в какой-то степени, показывает недостаточностьпредложенной классификации, упуская, например, инструменты функционального тестирования, средства тестирования безопасности, инструменты тестирования пользовательского интерфейса, инструменты нагрузочного тестирования и др., соответствующие, различным целям тестирования, представленным в секции 2.2 области знаний SWEBOK “Тестирование”, и естественно задающим “подвиды” возможного класса “специализированных или целевых инструментов тестирования”, к которым, в частности, относится тестирование производительности.

1.5 Инструменты сопровождения (Software Maintenance Tools)

Эта тема охватывает инструменты, особенно важные для обеспечения сопровождения существующего программного обеспечения, подверженного модификациям. SWEBOK идентифицирует две категории таких инструментов:

Инструменты облегчения понимания (comprehension tools). Эти инструменты помогают человеку в понимании программ. Примерами могут служить различные средства визуализации.

Инструменты реинжиниринга (reengineering tools). Эти инструменты поддерживают деятельность по реинжинирингу, описанную в области знаний SWEBOK “Software Maintenance”.

Средства “обратного” инжиниринга (reverse engineering) помогают в процессе восстановления для существующего программного обеспечения таких артефактов, как спецификация и описание дизайна (архитектуры), которые, в дальнейшем, могут быть трансформированы для генерации нового продукта на основе функциональности существующего.

Последнее замечание, в сочетании с типичной функциональностью современных средств проектирования, поддерживающих анализ исходного кода (в случае объектно-ориентированных систем) и его визуализацию (в том числе, поведенческую, например, в виде диаграмм UML Sequence), позволяет объединить упомянутые категории инструментов в единый класс “инструментов реинжиниринга”. В то же время, деятельность по сопровождению и поддержке, в частности, касающаяся сбоев и исправления обнаруженных ошибок в программном обеспечении, требует, в определенной степени, отнесения к этой теме и средств конфигурационного управления, рассматриваемых ниже (например, в части обработки запросов на изменения).

1.6 Инструменты конфигурационного управления (Software Configuration Management Tools)

Инструменты конфигурационного управления делятся на три категории:

Инструменты отслеживания (tracking) дефектов, расширений и проблем.

Инструменты управления версиями.

Инструменты сборки и выпуска. Эти инструменты предназначены для управления задачами сборки и выпуска продуктов, а также включают средства инсталляции.

Дополнительная информация по данной теме представлена в области знаний SWEBOK “Конфигурационное управление”.

1.7 Инструменты управления инженерной деятельностью (Software Engineering Management Tools)

Средства управления деятельностью по программной инженерии делятся на три категории:

Инструменты планирования и отслеживания проектов. Эти средства используются календарного планирования работ, количественной оценки усилий и стоимостных ожиданий, связанных с проектами.

Инструменты управления рисками. Эти средства используются для идентификации, оценки ожиданий и мониторинга рисков.

Инструменты количественной оценки. Эти инструменты ведения измерений помогают в выполнении работ, связанных с программой количественной оценки, проводимой в отношении проектов программного обеспечения.

Функциональные аспекты управления инженерной деятельностью достаточно детально представлены в области знаний SWEBOK “Управление программной инженерией” (Software Engineering Management).

1.8 Инструменты поддержки процессов (Software Engineering Process Tools)

В описании этой темы в текущей версии SWEBOK наблюдается противоречие между кратким делением на категории инструментов и их более детальным определением. Скорее всего, такая несогласованность связана, в первую очередь, с отсутствием достигнутого консенсуса в этой области. Базируясь на обеих классификациях, упомянутых в SWEBOK, хотелость бы отметить несколько типов инструментов из “смежных” областей, имеющих особое значение в поддержке процессов программной инженерии:

Инструменты моделирования, позволяющие, в частности, описать и модель процессов, как таковую.

Инструменты управления проектами.

Инструменты конфигурационного управления, поддерживающие работу с актуальными версиями всего комплекса артефактов проекта и, что не менее важно, позволяющие задать поведенческие характеристики (в упрощенном понимании - workflow) и атрибуты этих артефактов в форме элементов конфигураций.

Ролевые платформы разработки программного обеспечения, охватывающие все стадии жизненного цикла и, на сегодняшний день, являющиеся развитием интегрированных средств разработки и CASE-инструментов в направлении поддержки “смежной” функциональности – управления требованиями, работ по конфигурационному управлению с поддержкой управления изменениями, тестирования и оценки качества.

Первые три вида инструментов в такой классификации позволяют описать применяемые процессы программной инженерии. Четвертый класс – “супер-интегрированные среды разработки”, называемые сегодня ролевыми платформами разработки, обеспечивают поддержку заданных процессов, описанных, например, в виде соответствующих правил на уровне глубоко интегрированных в такие среды инструментов конфигурационного управления.

1.9 Инструменты обеспечения качества (Software Quality Tools)

Средства обеспечения качества делятся на две категории:

Инструменты инспектирования. Эти средства используются для поддержки обзора (review) и аудита.

Инструменты (статического) анализа. Эти средства используются для анализа программных артефактов, данных, потоков работ и зависимостей. Такие инструменты предназначены для проверки определенных свойств или артефактов, в целом, на соответствие заданным характеристикам.

# Программа база данных «Аптека»

Программа «Аптека» показывает пользователю доступные медицинские препараты, их наименование, цену и фирму-производитель.

Семейный бюджет – это общая сумма доходов и расходов всех членов семьи в течение, определённого времени. Расходы - это денежные средства, затраченные на содержание семьи. Расходы бывают постоянные и непредвиденные.

Постоянные расходы - это платежи за услуги, которые необходимо оплачивать ежемесячно: коммунальные платежи, оплата за услуги связи, за автостоянку, за кредит, за детский сад, за учебу детей в образовательных заведениях. К постоянным расходам относятся расходы на продукты питания и медикаменты, на бытовую химию, на содержание автомобиля. Так же следует выделять деньги на карманные расходы, на проезд в общественном транспорте, на обеды на работе и в учебных заведениях.

Непредвиденные расходы - это такие расходы, которые часто неожиданно возникают в семье и которые просто невозможно отложить.

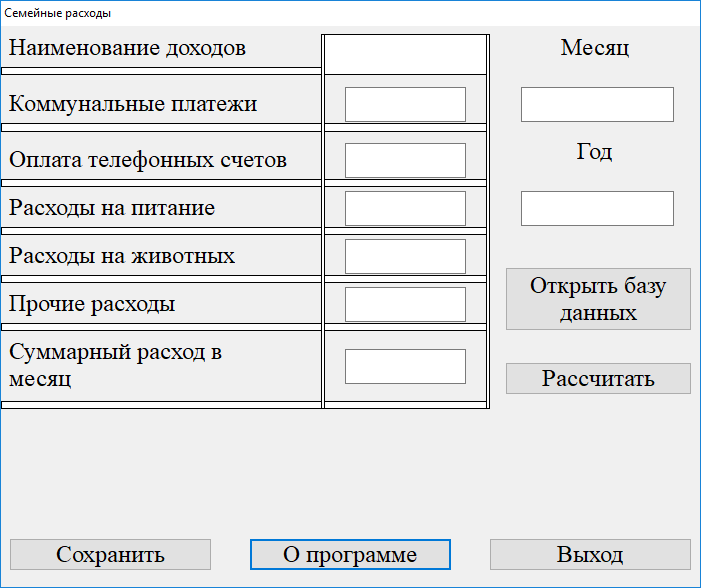


Рисунок 1

На рисунке 1 показано основная форма программы. На ней можно ввести данные о расходах семьи и сохранить их в базу данных с помощью кнопки «Сохранить». Для этого был использован следующий программный код:

Form2->ADOQuery1->Close();

Form2->ADOQuery1->Open();

Form2->ADOQuery1->Insert();

Form2->ADOQuery1->FieldByName("Коммунальные платежи")->AsString = Edit1->Text;

Form2->ADOQuery1->FieldByName("Оплата телефонных счетов")->AsString = Edit3->Text;

Form2->ADOQuery1->FieldByName("Расходы на питание")->AsString = Edit4->Text;

Form2->ADOQuery1->FieldByName("Расходы на животных")->AsString = Edit5->Text;

Form2->ADOQuery1->FieldByName("Прочие расходы")->AsString = Edit6->Text;

Form2->ADOQuery1->FieldByName("Суммарный расход")->AsString = Edit7->Text;

Form2->ADOQuery1->FieldByName("Месяц")->AsString = Edit8->Text;

Form2->ADOQuery1->FieldByName("Год")->AsString = Edit9->Text;

Form2->ADOQuery1->Post();

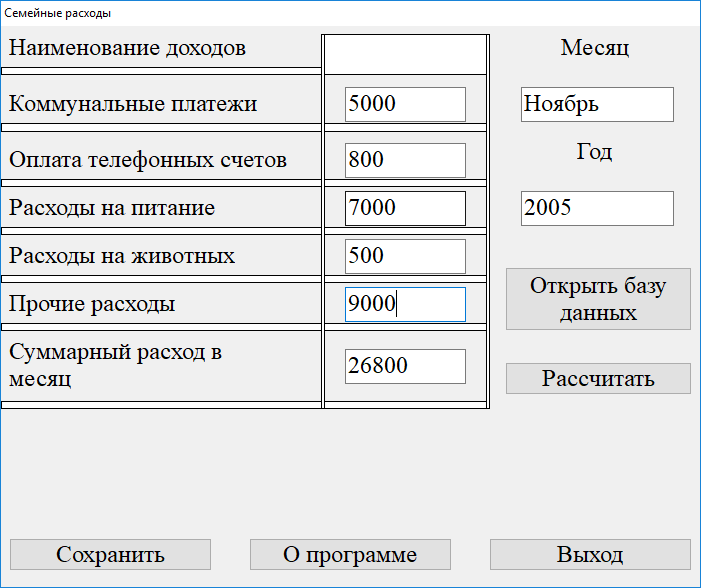


Рисунок 2

После ввода данных пользователь может узнать сумму его расходов за конкретный месяц с помощью кнопки «Рассчитать».

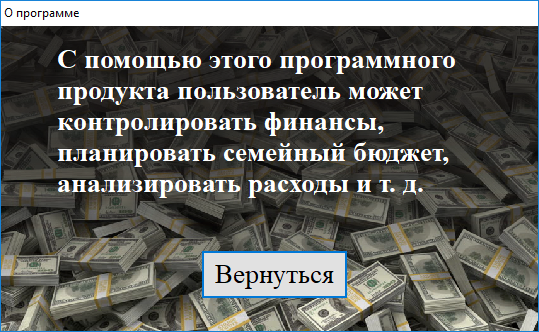


Рисунок 3

Нажав на кнопку «О программе» откроется окно с краткой информацией о возможностях программы. Так же, на данной форме будет кнопка «Вернуться», которая вернет пользователя на главную форму. Данная возможность была реализована с помощью кода:

Close();

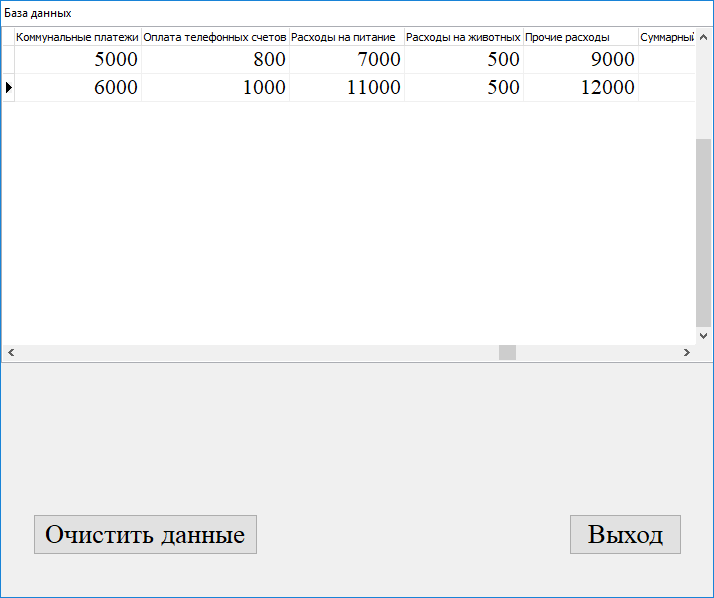


Рисунок 4

При нажатии кнопки «Открыть базу данных» откроется окно с таблицей, выводящей записанные в базу данных информацию. Программа предоставляет пользователю возможность удалять информацию. Для этого нужно выбрать нужную строчку и щелкнуть по кнопке «Очистить данные». После того, как пользователь подтвердит свое решение, выбранная строка будет удалена.

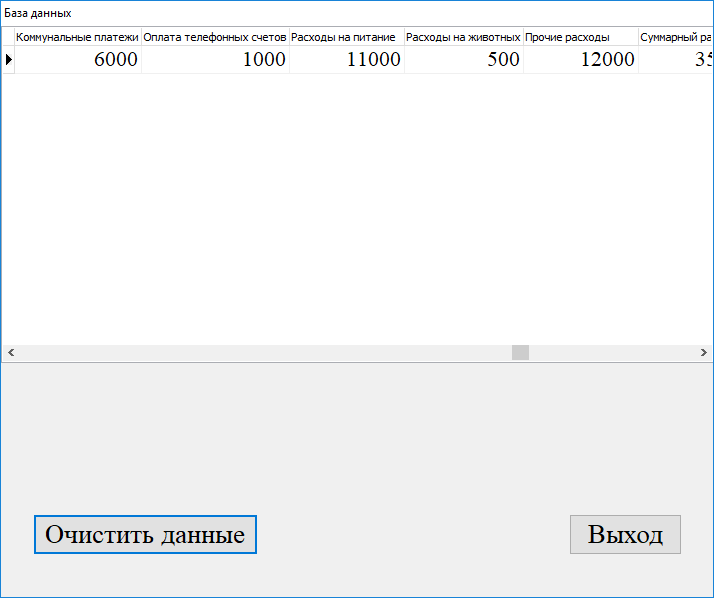


Рисунок 5

На рисунке 5 показан результат удаления строки с помощью кнопки «Очистить данные». Код для реализации данной процедуры:

AnsiString ds = "Вы действительно хотите удалить выбранную строку? \n\n";

if(MessageDlg(ds,mtWarning,TMsgDlgButtons()<<mbOK<<mbCancel,0))

{

ADOQuery1->Delete();

}

Для создания данного программного продукта были использованы следующие компоненты: TButton, TLabel, TEdit, TImage, TShape, DBGrid, ADOQuery, ADOConnection, DataSource.

Наиболее часто используемой кнопкой является кнопка Button, расположенная на странице библиотеки Standard. Основное с точки зрения внешнего вида свойство кнопки - Caption(надпись). В надписях кнопок можно предусматривать использование клавиш ускоренного доступа, выделяя для этого один из символов надписи. Перед символом, который должен соответствовать клавише ускоренного доступа, ставится символ амперсанда «&». Этот символ не появляется в надписи, а следующий за ним символ оказывается подчеркнутым. Тогда пользователь может вместо щелчка на кнопке нажать в любой момент клавишу Alt совместно с клавишей выделенного символа.

Основное событие любой кнопки - OnClick, возникающее при щелчке на ней. Именно в обработчике этого события записываются операторы, которые должны выполняться при щелчке пользователя на кнопке.

 Свойство Cancel, если установить его в true, определяет, что нажатие пользователем клавиши Esc будет эквивалентно щелчку на данной кнопке. Это свойство целесообразно задавать равным true для кнопок «Отменить» в различных диалоговых окнах, чтобы можно было выйти из диалога, нажав на эту кнопку или нажав клавишу Esc.

 Свойство Default, если его установить в true, определяет, что нажатие пользователем клавиши ввода Enter будет эквивалентно нажатию на данную кнопку, даже если данная кнопка в этот момент не находится в фокусе. Правда, это сработает, если в фокусе находится какой-то оконный компонент. Если же в момент нажатия Enter в фокусе находится другая кнопка, то все-таки сработает именно кнопка в фокусе. Если у нескольких кнопок на форме свойство Default задано равным true, то при нажатии Enter сработает та из них, которая находится раньше в последовательности табуляции.

TLabel

Отображает на форме прямоугольную область статического текста, который нельзя редактировать. Обычно текст представляет собой название другой компоненты. Текст названия является значением свойства Caption. Свойство Alignment определяет способ выравнивания текста. Чтобы размер шрифта автоматически соответствовал максимальному заполнению области, установите значение true свойства AutoSize.

TEdit

Отображает прямоугольную область редактируемого ввода одиночной строки информации на форме. Начальное содержимое области редактирования определяет строка, являющаяся значением свойства Text. TEdit является прямой производной от класса TCustomEdit, полностью наследуя его свойства, методы и события.

Внешнее оформление окон редактирования определяется свойством BorderStyle. Во всех рассматриваемых окнах вводимый и выводимый текст содержится в свойстве Text типа AnsiString. Это свойство можно устанавливать в процессе проектирования или задавать программно. Выравнивание текста, как это имело место в метках и панелях, невозможно. Перенос строк тоже невозможен. Текст, не помещающийся по длине в окно, просто сдвигается и пользователь может перемещаться по нему с помощью курсора. Свойство AutoSize в окнах редактирования имеет смысл, отличный от смысла аналогичного свойства меток автоматически подстраивается под размер текста только высота, но не ширина окна.

 Окна редактирования снабжены многими функциями, свойственными большинству редакторов. Например, в них предусмотрены типичные комбинации «горячих» клавиш: Ctrl+C копирование выделенного текста в буфер обмена Clipboard (команда Copy), Ctrl+X вырезание выделенного текста в буфер Clipboard (команда Cut), Ctrl+V вставка текста из буфера Clipboard в позицию курсора (команда Paste), Ctrl+Z отмена последней команды редактирования. Правда, пользователи часто не догадываются об этих возможностях редактирования. Так что полезно напоминать им об этом соответствующими подсказками.

 Свойство AutoSelect определяет, будет ли автоматически выделяться весь текст при передаче фокуса в окно редактирования. Его имеет смысл задавать равным true в случаях, когда при переключении в данное окно пользователь вероятнее всего будет заменять текущий текст, а не исправлять его. Имеются также свойства только времени выполнения SelLength, SelStart, SelText, определяющие соответственно длину выделенного текста, позицию перед первым символом выделенного текста и сам выделенный текст.

Компонент **Shape** представляет собой различные геометрические фигуры, соответствующим образом заштрихованные. Основное свойство этого компонента Shape Другое существенное свойство компонента - Brush (кисть). Это свойство является объектом типа TBrush, имеющим ряд подсвойств, в частности: цвет (Brush.Color) и стиль (Brush.Style) заливки фигуры. Третье из специфических свойство компонента Shape - Pen (перо), определяющее стиль линий.

Для вывода записей набора данных в табличном виде удобно использовать сетку, представленную компонентом **DBGrid**. Внешний вид сетки соответствует внутренней структуре таблицы БД и набора данных, при этом строке сетки соответствует запись, а столбцу поле. С помощью сетки пользователь управляет набором данных, поля которого в ней отображаются. Для навигации по записям и их просмотра используются полосы прокрутки и клавиши перемещения курсора. Для перехода в режим редактирования поля достаточно установить на него курсор и нажать любую алфавитно-цифровую клавишу. Переход в режим вставки новой записи выполняется нажатием клавиши <Insert>, после чего можно заполнять поля. Вставка записи происходит в том месте, где находится указатель текущей записи. Изменения, сделанные при редактировании или добавлении записи, подтверждаются нажатием клавиши <Enter>, или переходом к другой записи, или отменяются нажатием клавиши <Esc>. Для удаления записи следует нажать комбинацию клавиш <Ctrl>+<Delete>.

Характеристики сетки:

Сетка DBGrid отображает все записи, имеющиеся в наборе данных, т.е. числом строк в ней управлять нельзя.

 Основным свойством сетки DBGrid является свойство Columns типа TDBGridColumns, которое представляет собой массив (коллекцию) объектов Column типа TColumn, описывающих отдельные столбцы сетки.

Свойство SelectedIndex типа int задает номер текущего столбца в массиве columns, а свойство SelectedField указывает на объект типа TField, которому соответствует текущий столбец сетки.

Свойство Fieldcount типа int, доступное во время выполнения программы, содержит число видимых столбцов сетки, а свойство Fields [int FieldIndex] типа TField позволяет получить доступ к отдельным столбцам. Индекс определяет номер столбца в массиве столбцов и принимает значения в интервале 0..Fieidcount - 1.

Свойства Color и FixedColor типа TColor задают цвет сетки и ее фиксированных элементов соответственно. По умолчанию свойство color имеет значение clWindow (цвет фона Windows), а свойство FixedColor - значение clBtnFace (цвет кнопки).

Свойство TitleFont типа TFont определяет шрифт, используемый для вывода заголовков столбцов.

Доступ к параметрам сетки (например, для настройки) возможен через свойство Options типа TGridOptions.

Компонент DataSource действует как посредник между компонентами TDataSet (TTable, TQuery, TStoredProc) и компонентами Data Controls - элементами управления, обеспечивающими представление данных на форме. Компоненты TDataSet управляют связями с библиотекой Borland Database Engine (BDE), а компонент DataSource управляет связями с данными в компонентах Data Controls.

В типичных приложениях БД компонент DataSource, как правило, связан с одним компоненом TDataSet (TTable или TQuery) и с одним или более компонентами Data Controls (такими, как DBGrid, DBEdit и др.). Связь этого компонента с компонентами TDataSet и DataControls осуществляется с использованием следующих свойств и событий:

Cвойство DataSet компонента DataSource идентифицирует имя компонента TDataSet. Можно присвоить значение свойству DataSet на этапе выполнения или с помощью инспектора объектов на этапе проектирования.

Cвойство Enabled компонента DataSource активизирует или останавливает взаимосвязь между компонентами TDataSource и Data Controls. Если значение свойства Enabled равно true, то компоненты Data Controls, связанные с TDataSource, воспринимают изменения набора данных. Использование свойства Enabled позволяет временно разъединять визуальные компоненты Data Controls и TDataSource, например, для того, чтобы в случае поиска в таблице с большим количеством записей не отображать на экране пролистывание всей таблицы.

Свойство AutoEdit компонента DataSource контролирует, как инициируется редактирование в компонентах Data Controls. Если значение свойства AutoEdit равно true, то режим редактирования начинается непосредственно при получении фокуса компонентом Data Controls, связанным с данным компонентом TDataSet. В противном случае режим редактирования начинается, когда вызывается метод Edit компонента TDataSet, например, после нажатия пользователем кнопки Edit на компоненте DBNavigator. · Событие OnDataChange компонента DataSource наступает, когда происходит изменение значения поля, записи, таблицы, запроса.

Cобытие OnUpdateData компонента DataSource наступает, когда пользователь пытается изменить текущую запись в TDataSet. Обработчик этого события следует создавать, когда требуется соблюсти условия ссылочной целостности или ограничения, накладываемые на значения полей изменяемой базы данных.

  Компонент ADOQuery представляет собой запрос к базе данных. Это может быть как запрос, в результате которого возвращаются данные из базы (например, SELECT), так и запрос, не формирующий результирующего набора данных (например, INSERT). Компонент аналогичен компоненту Query из BDE. Все основные его свойства мы уже рассматривали в описании свойств компонентов Query и ADODataSet.

Компонент TADOQuery обеспечивает применение запросов SQL при работе с данными через ADO. Это могут быть запросы просмотра данных SELECT и запросы изменения данных INSERT, DELETE, UPDATE, ALTER TABLE, CREATE TABLE. Могут также выполняться хранимые процедуры. По своей функциональности он подобен стандартному компоненту запроса TQuery. В отличие от TADOCommand, этот компонент преимущественно предназначен для получения набора записей из одной или нескольких таблиц БД. Также как TQuery, TADOQuery имеет свойство DataSource, позволяющее передать параметры запроса от одного компонента другому.

Основное свойство компонента — SQL, содержащее запрос. Запрос SELECT выполняется методом Open или заданием значения true свойству Active. Выполнение запросов, не возвращающих множество данных (INSERT, DELETE,UPDATE, ALTER TABLE, CREATE TABLE), осуществляется методом ExecSQL.  
Любое изменение текста SQL во время выполнения приводит к разрыву соединения компонента с базой данных. Так что после задания нового значения SQL соединение надо восстанавливать методом Open или заданием true свойству Active. При этом если связь с базой данных осуществляется через компонент ADOConnection, надо учитывать указанную в описании этого компонента взаимосвязь свойства Active компонента ADOQuery и свойства Connected компонента ADOConnection.

Фактически все свойства и методы компонента ADOQuery совпадают со свойствами и методами компонента ADODataSet, за исключением рассмотренного свойства SQL и метода ExecSQL.

ADOConnection - ADO-соединение, используется для установки соединения с ADO-источником данных и обеспечивает поддержку транзакций.

Добавить заключение и список литературы!

# Заключение

В заключение отчета нужно сказать, что при прохождении производственной практики особых трудностей не возникало. Было очень интересно знакомиться с работой программиста. Данная практика поможет мне в дальнейшей работе программистом. Думаю, что опыт, полученный на данной практике, несомненно, пригодится мне в дальнейшем при построении своей профессиональной деятельности.

# Список литературы

Учебные издания, рекомендованные МО РФ и УМЦ СПО ФАС, для образовательных учреждений среднего профессионального образования.

1. Ашарина И.В., Крупская Ж.Ф. Язык С++ и объектно-ориентированное программирование в С++. Лабораторный практикум. Учебное пособие для вузов. – М.: Горячая линия – Телеком, 2015. – 232 с.

2. Лафоре Р. Объектно-ориентированное программирование в С++. Классика Computer Science. 4-е изд. – СПб.: Питер, 2016. – 928 с.

3. Культин Н.Б. C++ Builder в задачах и примерах. – СПб.: БХВ-Петербург, 2007. – 336 с.

4. Немцова Т.И., Голова С.Ю., Терентьев А.И. Программирование на языке высокого уровня. Программирование на языке С++: учебное пособие / под редакцией Л.Г. Гагариной. – М.: ИД "ФОРУМ": ИНФРА-М, 2014. – 512 с.: ил. – (Профессиональное образование)

5. А. Я. Архангельский - Программирование в C++ Builder  
Интернет-источники:

1. [www.cyberforum.ru/](https://vk.com/away.php?utf=1&to=http%3A%2F%2Fwww.cyberforum.ru%2F) - (cyberforum - форум программистов и сисадминов).

2. [https://ru.stackoverflow.com/](https://vk.com/away.php?utf=1&to=https%3A%2F%2Fru.stackoverflow.com%2F) - (Stack Overflow - участники данного форума отвечают на возникающие вопросы, а также пытаются разобраться в проблеме тех пользователей, которые опубликовывают на данном форуме свои вопросы).

3. [http://cppstudio.com/](https://vk.com/away.php?utf=1&to=http%3A%2F%2Fcppstudio.com%2F) (cppstudio – представляет своим пользователям доступ к статьям по различным разделам языка программирования C++).

4. [https://code-live.ru/](https://vk.com/away.php?utf=1&to=https%3A%2F%2Fcode-live.ru%2F) - (code-live – полезные статьи по различным разделам языка C++ и готовые проекты