МИНИСТЕРСТВО ПО РАЗВИТИЮ
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И
КОММУНИКАЦИЙ РЕСПУБЛИКИ УЗБЕКИСТАН

ТАШКЕНТСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ
ТЕХНОЛОГИЙ

К защите.
Заведующий кафедры «СПП»
к.т.н. Керимов К.Ф. _____
«__» _____ 2015 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

ТЕМА: «РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ СРЕДСТВ АВТОМАТИЗАЦИИ РЕСТОРАННОГО БИЗНЕСА НА C++BUILDER»

Выпускник Арипов Н.М.
_____

Научный руководитель Решетов Э.С.
_____

Рецензент к.т.н. Зиядуллаев Д.Ш.
_____

Консультант по БЖД Абдуллаева С.М.
_____

ТАШКЕНТ – 2015 г.

МИНИСТЕРСТВО ПО РАЗВИТИЮ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И КОММУНИКАЦИЙ РЕСПУБЛИКИ УЗБЕКИСТАН
ТАШКЕНТСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Факультет Программный инжиниринг кафедра СПП
Направление (специальность) 5330200 – Информатика и информационные технологии

УТВЕРЖДАЮ
Зав кафедрой _____
«_____» _____ 2015 г.

ЗАДАНИЕ

на выпускную квалификационную работу
Арипова Нурсултана Марат угли

(фамилия, имя, отчество)

1. Тема работы «Разработка программного обеспечения средств автоматизации ресторанного бизнеса на C++ Builder»
2. Утверждена приказом по университету от «22» 01 2015___ г. № 80-16
3. Срок сдачи законченной работы : 10.06.2015.
4. Исходные данные к работе: Среда разработки C++ Builder 6 .
5. Содержание расчётно – пояснительной записи (перечень подлежащих разработке вопросов) ___Введение, обзорная часть, основная часть, безопасность жизнедеятельности, заключение, список литературы, приложение.
6. Перечень графического материала Слайды презентации
7. Дата выдачи задания:5.02.2015

Руководитель _____
(подпись)

Задание принял _____
(подпись)

8. Консультанты по отдельным разделам выпускной работы

Раздел	Ф.И.О руководителя	Подпись дата	
		Задание выдал	Задание получил
1. Введение	Решетов Э.С.		
2.Теоретическая часть	Решетов Э.С.		
3. Основная часть	Решетов Э.С.		
4. БЖД	Абдуллаева С.М.		
5. Заключение	Решетов Э.С.		

9. График выполнения работы

№	Наименование раздела работы	Срок Выполнения	Отметка руководителя о выполнении
1.	Введение	12.04-15.04	
2.	Теоретическая часть	16.04-23.04	
3.	Основная часть	24.04-17.05	
4..	БЖД	18.05-24.05	
6.	Заключение	25.05-29.05	

Выпускник _____ «_____» _____ 2015 г.
(подпись)

Руководитель _____ «_____» _____ 2015 г.
(подпись)

Аннотация

Выпускная работа посвящена разработке программного обеспечения средств автоматизации ресторанного бизнеса. Программное обеспечение представляет собой приложение под Windows XP, реализованное средствами языка C++ Builder 6. Приложение является законченным программным продуктом.

Annotation

Final work is dedicated to the development of software for the automation of restaurant business. The software is an application for Windows XP, implemented by means of language C ++ Builder 6. The application is a complete software package .

Мазмуннома

Битирув иш ресторан бизнес сохасида автоматлаштириш воситаларининг дастурий таъминотини ишлаб чикишга багишланган. Дастурий таъминот Windows XP учун, C++Builder 6 тилида узи билан илова булиб такдим этилади. Илова, тайёр дастурий махсулот булиб хисобланади.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Глава нашего государства Ислам Каримов, постоянно развивает сферу ИКТ, разрабатывая различные программы и определяя перспективы направления дальнейшей деятельности. Для этого наш Президент принял постановление "О мерах по дальнейшему внедрению и развитию современных информационно-коммуникационных технологий.

Актуальность темы. Современные информационные технологии - это методы и средства для сбора, хранения, обработки и получения информации на основе современных средств вычислительной техники.

Составными частями любой информационной системы являются БД и приложение для обработки данных. Появление персональных машин класса Pentium, семейства операционных систем Windows фирмы Microsoft и различного программного обеспечения позволяют автоматизировать ручные операции, вести любые виды работ по накоплению информации, ее обработки и получению различных выходных форм.

Объем обрабатываемой информации, которой занимаются современные люди, с каждым годом увеличивается. После достижения независимости нашей Родины мы начали уделять самое серьезное внимание развитию телекоммуникаций и информационных технологий. Вместе с тем в нынешнее скоротечное время, когда процессы информатизации приобретают все более глобальный характер, когда конкуренция усиливается и информация становится не только товаром, но и действенным идеологическим оружием, сама жизнь и национальные интересы Узбекистана решительно выдвигают все новые и новые задачи.

В частности, задача разработки программного обеспечения средств автоматизации ресторанного бизнеса на C++ Builder является одной из важнейших задач автоматизации человеческой деятельности. Ввиду чего,

поставленная в выпускной работе задача является актуальной в настоящее время.

Цель работы – разработка программного обеспечения средств автоматизации информационной системы ресторанного бизнеса на C++ Builder .

В выпускной работе ставится задача — разработать проект базы данных для накопления необходимой информацией. Разработать приложение, позволяющее вести учет и контроль за базой данных. База данных должна быть спроектирована с учетом реализации запросов различного типа по получению информации.

Одной из широко используемых сред разработки является Borland C++Builder, который позволяет создавать различные приложения: от простейших однооконных приложений до программ распределения баз данных .

При этом необходимо решить следующие задачи:

- необходимо создать наглядный и удобный интерфейс для пользования данным программным продуктом;
- создать надежную и защищенную базу данных;
- определить основные принципы для обеспечения оптимальной работы;
- провести анализ полученных результатов.

Правильно подобранные программы позволяют экономить время выполнения работ и делают их более точными, и быстрыми.

Для удобного и функционального написания подобного программного обеспечения лучше всего подходит среда объектно-ориентированного программирования на языке C++ Builder.

Основной упор объектно-ориентированной модели в C++ Builder делается на максимальном использовании кода. Это позволяет разработчикам строить приложения весьма быстро из заранее подготовленных объектов, а

также дает им возможность создавать свои собственные объекты для среды C++ Builder. Никаких ограничений по типам объектов, которые могут создавать разработчики, не существует. Действительно, все в C++ Builder написано на нём же, поэтому разработчики имеют доступ к тем же объектам и инструментам, которые использовались для создания среды разработки. В результате нет никакой разницы между объектами, поставляемыми Borland или третьими фирмами, и объектами, которые вы можете создать.

Пояснительная записка выпускной работы состоит из следующих частей:

- Введения;
- Обзорной части;
- Основной части;
- Раздела БЖД;
- Заключения;
- Списка используемой литературы;
- Приложений.

В обзорной части представлены следующие вопросы: современный подход к автоматизации ресторанного бизнеса, требования к базам данных, информационные системы и базы данных, C++ Builder и базы данных, процессор баз данных Borland Database Engine, архитектура и функции BDE.

В основной части представлены разделы: постановка задачи, руководство пользователю, руководство программисту, функциональная связь модулей.

В заключении сделан вывод по практической реализации поставленной задачи, требованиям пользователя, целостности выбранного метода решения задачи, а также изложены рекомендации по использованию проекта и возможности его расширения в будущем.

В приложении приводятся тексты программных модулей.

1. ОБЗОРНАЯ ЧАСТЬ. База данных и СУБД.

1.1. Современный подход к автоматизации ресторанного бизнеса

Создание любой системы предприятия общепита предполагает в первую очередь удовлетворение гастрономических нужд населения. Эта сфера является отображением экономического развития страны. Степень этого развития прямо влияет на доходы населения, от размера которых зависит желание тратить часть этих доходов на услуги и товары, не исключая сферы по предоставлению услуг общественного питания.

Было проведено большое количество научных исследований для изучения вкусов потребителя, чтобы по максимуму удовлетворить их или в отдельных случаях влиять на них. На эти темы было опубликовано много публикаций.

Вкусовые проявления специалисты делят на следующие виды: групповые, индивидуальные, этнические и национальные. Эти виды вкусовых проявлений влияют на выбор типа питания и впоследствии на выбор соответствующего типа предприятия. Итак, обращая внимание на нужды и привязанности населения, фирмы и другие организации, которые специализируются на общественном питании, включая также само государство, открывают рестораны разных видов: с национальной кухней, рестораны концептуальные, молодежные кафе, закусочные, столовые.

Все эти заведения рассчитаны на определенную ячейку рынка и учитывают социально-экономические потребности своих клиентов. Работа этих предприятий будет слаженной и удачной только при правильной организации, которая предусматривает четко определенную систему конкретного типа предприятия. Причем, конкретный тип предприятия должен соответствовать определенным требованиям к предоставляемым услугам.

Автоматизация ресторанного бизнеса включает в себя следующее: современный интуитивно понятный графический интерфейс; ведение полной истории всех действий персонала; подробнейшая информация обо всех операциях, проводимых на предприятии; доступность для авторизованных пользователей ко всей информации и отчетам через интернет из любой точки мира; возможность подключения нескольких фискальных регистраторов к одному рабочему месту кассира, что позволяет оформлять продажу с одного рабочего места от лица нескольких юридических лиц без дополнительных действий кассира или официантов; круглосуточная без праздников и выходных техническая поддержка клиентов в режиме 24/7; регулярный выход новых версии системы; бесплатное обновление программы организациям, заключившим договор на техобслуживание; обучение персонала предприятия работе в системе; индивидуальный подход к каждому клиенту; гарантия 3 года на все программные модули системы.

1.2. Понятие базы данных и СУБД

База данных – это набор сведений, относящихся к определенной теме или задаче, такой как отслеживание заказов клиентов или хранение коллекции звукозаписей. Цель любой информационной системы – обработка данных об объектах реального мира. Если база данных хранится не на компьютере или на компьютере хранятся только ее части, приходится отслеживать сведения из целого ряда других источников, которые пользователь должен скоординировать и организовать самостоятельно.

Создавая базу данных, пользователь стремится упорядочить информацию по различным признакам и быстро извлекать выборку с произвольным сочетанием признаков. Сделать это возможно, только если данные структурированы.

Структурирование – это введение соглашений о способах представления данных.

Неструктурированными называют данные, записанные, например, в текстовом файле.

Пользователями базы данных могут быть различные прикладные программы, программные комплексы, а также специалисты предметной области, выступающие в роли потребителей или источников данных, называемые конечными пользователями.

В современной технологии баз данных предполагается, что создание базы данных, ее поддержка и обеспечение доступа пользователей к ней осуществляются централизованно с помощью специального программного инструментария – системы управления базами данных.

СУБД используют для управления, создания и использования БД. СУБД должна предоставлять доступ к данным любым пользователям, включая и тех, которые практически не имеют и (или) не хотят иметь представления о:

- физическом размещении в памяти данных и их описаний;
- механизмах поиска запрашиваемых данных;
- проблемах, возникающих при одновременном запросе одних и тех же данных многими пользователями (прикладными программами);
- способах обеспечения защиты данных от некорректных обновлений и (или) несанкционированного доступа;
- поддержании баз данных в актуальном состоянии.

В качестве СУБД для реализации базы данных была использована Microsoft Access. Microsoft Access является настольной СУБД реляционного типа, которая имеет все необходимые средства для выполнения перечисленных выше функций. Система управления базами данных Microsoft Access является одним из самых популярных приложений в семействе настольных СУБД. Все версии Access имеют в своем арсенале средства, значительно упрощающие ввод и обработку данных, поиск данных и предоставление информации в виде таблиц, графиков и отчетов. Начиная с версии Access 2000, появились также Web-страницы доступа к данным, которые пользователь может просматривать с помощью программы Internet Explorer. Помимо этого, Access позволяет использовать электронные таблицы и таблицы из других настольных и серверных баз данных для

хранения информации, необходимой приложению. Присоединив внешние таблицы, пользователь Access будет работать с базами данных в этих таблицах так, как если бы это были таблицы Access. При этом и другие пользователи могут продолжать работать с этими данными в той среде, в которой они были созданы.

Достоинством Access является то, что она имеет очень простой графический интерфейс, который позволяет не только создавать собственную базу данных, но и разрабатывать простые и сложные приложения. В отличие от других настольных СУБД, Access хранит все данные в одном файле, хотя и распределяет их по разным таблицам.

1.3. Требования к базам данных

Итак, хорошо спроектированная база данных:

- Удовлетворяет всем требованиям пользователей к содержимому базы данных. Перед проектированием базы необходимо провести обширные исследования требований пользователей к функционированию базы данных.
- Гарантирует непротиворечивость и целостность данных. При проектировании таблиц нужно определить их атрибуты и некоторые правила, ограничивающие возможность ввода пользователем неверных значений. Для верификации данных перед непосредственной записью их в таблицу база данных должна осуществлять вызов правил модели данных и тем самым гарантировать сохранение целостности информации.
- Обеспечивает естественное, легкое для восприятия структурирование информации. Качественное построение базы позволяет делать запросы к базе более “прозрачными” и легкими для понимания; следовательно,

снижается вероятность внесения некорректных данных и улучшается качество сопровождения базы.

- Удовлетворяет требованиям пользователей к производительности базы данных. При больших объемах информации вопросы сохранения производительности начинают играть главную роль, сразу “высвечивая” все недочеты этапа проектирования.

Следующие пункты представляют основные шаги проектирования базы данных:

- Определить информационные потребности базы данных.
- Проанализировать объекты реального мира, которые необходимо смоделировать в базе данных. Сформировать из этих объектов сущности и характеристики этих сущностей (например, для сущности “деталь” характеристиками могут быть “название”, “цвет”, “вес” и т.п.) и сформировать их список.
- Поставить в соответствие сущностям и характеристикам - таблицы и столбцы (поля) в нотации выбранной Вами СУБД (Paradox, dBase, FoxPro, Access, Clipper, InterBase, Sybase, Informix, Oracle и т.д.).
- Определить атрибуты, которые уникальным образом идентифицируют каждый объект.
- Выработать правила, которые будут устанавливать и поддерживать целостность данных.
- Установить связи между объектами (таблицами и столбцами), провести нормализацию таблиц.

- Спланировать вопросы надежности данных и, при необходимости, сохранения секретности информации.

Шаги проектирования базы данных

Первый шаг состоит в определении:

I. Информационных потребностей базы данных. Он включает в себя опрос будущих пользователей для того, чтобы понять и задокументировать их требования. Следует выяснить следующие вопросы:

- сможет ли новая система объединить существующие приложения или их необходимо будет кардинально переделывать для совместной работы с новой системой;
- какие данные используются разными приложениями; смогут ли Ваши приложения совместно использовать какие-либо из этих данных;
- кто будет вводить данные в базу и в какой форме; как часто будут изменяться данные;
- достаточно ли будет для Вашей предметной области одной базы или Вам потребуется несколько баз данных с различными структурами;
- какая информация является наиболее чувствительной к скорости ее извлечения и изменения.

II. Следующий шаг включает в себя анализ объектов реального мира, которые необходимо смоделировать в базе данных.

Формирование концептуальной модели базы данных включает в себя:

- идентификацию функциональной деятельности Вашей предметной области. Например, если речь идет о деятельности предприятия, то в качестве функциональной деятельности можно идентифицировать

ведение учета работающих, отгрузку продукции, оформление заказов и т.п.

- идентификацию объектов, которые осуществляют эту функциональную деятельность, и формирование из их операций последовательности событий, которые помогут Вам идентифицировать все сущности и взаимосвязи между ними. Например, процесс “ведение учета работающих” идентифицирует такие сущности как РАБОТНИК, ПРОФЕССИЯ, ОТДЕЛ.
- идентификацию характеристик этих сущностей. Например, сущность РАБОТНИК может включать такие характеристики как Идентификатор Работника, Фамилия, Имя, Отчество, Профессия, Зарплата.
- идентификацию взаимосвязей между сущностями. Например, каким образом сущности РАБОТНИК, ПРОФЕССИЯ, ОТДЕЛ взаимодействуют друг с другом? Работник имеет одну профессию (для простоты!) и значится в одном отделе, в то время как в одном отделе может находиться много работников.

III. Третий шаг заключается в установлении соответствия между сущностями и характеристиками предметной области и отношениями и атрибутами в нотации выбранной СУБД. Поскольку каждая сущность реального мира обладает некими характеристиками, в совокупности образующими полную картину ее проявления, можно поставить им в соответствие набор отношений (таблиц) и их атрибутов (полей).

Перечислив все отношения и их атрибуты, уже на этом этапе можно начать устранять излишние позиции. Каждый атрибут должен появляться только один раз, и вы должны решить, какое отношение будет являться владельцем какого набора атрибутов.

IV. На четвертом шаге определяются атрибуты, которые уникальным образом идентифицируют каждый объект. Это необходимо для того, чтобы

система могла получить любую единичную строку таблицы. Вы должны определить первичный ключ для каждого из отношений. Если нет возможности идентифицировать кортеж с помощью одного атрибута, то первичный ключ нужно сделать составным - из нескольких атрибутов. Хорошим примером может быть первичный ключ в таблице работников, состоящий из фамилии, имени и отчества. Первичный ключ гарантирует, что в таблице не будет содержаться двух одинаковых строк. Во многих СУБД имеется возможность помимо первичного определять еще ряд уникальных ключей. Отличие уникального ключа от первичного состоит в том, что уникальный ключ не является главным идентифицирующим фактором записи и на него не может ссылаться внешний ключ другой таблицы. Его главная задача - гарантировать уникальность значения поля.

V. Пятый шаг предполагает выработку правил, которые будут устанавливать и поддерживать целостность данных. Будучи определенными, такие правила в клиент-серверных СУБД поддерживаются автоматически - сервером баз данных; в локальных же СУБД их поддержание приходится возлагать на пользовательское приложение.

Эти правила включают:

- определение типа данных
- выбор набора символов, соответствующего данной стране
- создание полей, опирающихся на домены
- установка значений по умолчанию
- определение ограничений целостности
- определение проверочных условий.

VI. На шестом шаге устанавливаются связи между объектами (таблицами и столбцами) и производится очень важная операция для исключения избыточности данных - нормализация таблиц. Каждый из различных типов

связей должен быть смоделирован в базе данных. Существует несколько типов связей:

- связь “один-к-одному”
- связь “один-ко-многим”
- связь “многие-ко-многим”.

Связь “один-к-одному” представляет собой простейший вид связи данных, когда первичный ключ таблицы является в то же время внешним ключом, ссылающимся на первичный ключ другой таблицы. Такую связь бывает удобно устанавливать тогда, когда невыгодно держать разные по размеру (или по другим критериям) данные в одной таблице. Например, можно выделить данные с подробным описанием изделия в отдельную таблицу с установлением связи “один-к-одному” для того чтобы не занимать оперативную память, если эти данные используются сравнительно редко. Связь “один-ко-многим” в большинстве случаев отражает реальную взаимосвязь сущностей в предметной области. Она реализуется уже описанной парой “внешний ключ-первичный ключ”, т.е. когда определен внешний ключ, ссылающийся на первичный ключ другой таблицы. Именно эта связь описывает широко распространенный механизм классификаторов. Имеется справочная таблица, содержащая названия, имена и т.п. и некие коды, причем, первичным ключом является код. В таблице, собирающей информацию - назовем ее информационной таблицей - определяется внешний ключ, ссылающийся на первичный ключ классификатора. После этого в нее заносится не название из классификатора, а код. Такая система становится устойчивой от изменения названия в классификаторах. Имеются способы быстрой “подмены” в отображаемой таблице кодов на их названия как на уровне сервера БД (для клиент-серверных СУБД), так и на уровне пользовательского приложения. Но об этом - в дальнейших уроках. Связь “многие-ко-многим” в явном виде в реляционных базах данных не поддерживается. Однако имеется ряд способов косвенной реализации такой

связи, которые с успехом возмещают ее отсутствие. Один из наиболее распространенных способов заключается во введении дополнительной таблицы, строки которой состоят из внешних ключей, ссылающихся на первичные ключи двух таблиц. Например, имеются две таблицы: КЛИЕНТ и ГРУППА_ИНТЕРЕСОВ. Один человек может быть включен в различные группы, в то время как группа может объединять различных людей. Для реализации такой связи “многие-ко-многим” вводится дополнительная таблица, назовем ее КЛИЕНТЫ_В_ГРУППЕ, строка которой будет иметь два внешних ключа: один будет ссылаться на первичный ключ в таблице КЛИЕНТ, а другой - на первичный ключ в таблице ГРУППА_ИНТЕРЕСОВ. Таким образом в таблицу КЛИЕНТЫ_В_ГРУППЕ можно записывать любое количество людей и любое количество групп. Итак, после определения таблиц, полей, индексов и связей между таблицами следует посмотреть на проектируемую базу данных в целом и проанализировать ее, используя правила нормализации, с целью устранения логических ошибок. Важность нормализации состоит в том, что она позволяет разбить большие отношения, как правило, содержащие большую избыточность информации, на более мелкие логические единицы, группирующие только данные, объединенные “по природе”. Таким образом, идея нормализации заключается в следующем. Каждая таблица в реляционной базе данных удовлетворяет условию, в соответствии с которым в позиции на пересечении каждой строки и столбца таблицы всегда находится единственное значение, и никогда не может быть множества таких значений. После применения правил нормализации логические группы данных располагаются не более чем в одной таблице. Это дает следующие преимущества:

- данные легко обновлять или удалять
- исключается возможность рассогласования копий данных
- уменьшается возможность введения некорректных данных.

Процесс нормализации заключается в приведении таблиц в так называемые нормальные формы. Существует несколько видов нормальных форм: первая нормальная форма (1НФ), вторая нормальная форма (2НФ), третья нормальная форма (3НФ), нормальная форма Бойса-Кодда (НФБК), четвертая нормальная форма (4НФ), пятая нормальная форма (5НФ). С практической точки зрения, достаточно трех первых форм - следует учитывать время, необходимое системе для “соединения” таблиц при отображении их на экране. Поэтому мы ограничимся изучением процесса приведения отношений к первым трем формам.

Этот процесс включает:

- устранение повторяющихся групп (приведение к 1НФ)
- удаление частично зависимых атрибутов (приведение к 2НФ)
- удаление транзитивно зависимых атрибутов (приведение к 3НФ).

VII. Седьмой шаг является последним в нашем списке, но не последним по важности в процессе проектирования базы данных. На этом шаге мы должны спланировать вопросы надежности данных и, при необходимости, сохранения секретности информации. Для этого необходимо ответить на следующие вопросы:

- кто будет иметь права (и какие) на использование базы данных
- кто будет иметь права на модификацию, вставку и удаление данных
- нужно ли делать различие в правах доступа
- каким образом обеспечить общий режим защиты информации и т.п.

1.4. Информационные системы и базы данных

Цель любой информационной системы — обработка данных об объектах реального мира. В широком смысле слова база данных — это совокупность сведений о конкретных объектах реального мира в какой-либо предметной области. Под предметной областью принято понимать часть реального мира, подлежащего изучению для организации управления и, в конечном счете, автоматизации, например предприятие, вуз и т д.

Создавая базу данных, пользователь стремится упорядочить информацию по различным признакам и быстро извлекать выборку с произвольным сочетанием признаков. Сделать это возможно, только если данные структурированы.

Структурирование — это введение соглашений о способах представления данных.

Неструктурированными называют данные, записанные, например, в текстовом файле.

Пользователями базы данных могут быть различные прикладные программы, программные комплексы, а также специалисты предметной области, выступающие в роли потребителей или источников данных, называемые конечными пользователями.

В современной технологии баз данных предполагается, что создание базы данных, ее поддержка и обеспечение доступа пользователей к ней осуществляются централизованно с помощью специального программного инструментария — системы управления базами данных.

База данных (БД) — это поименованная совокупность структурированных данных, относящихся к определенной предметной области.

Система управления базами данных (СУБД) — это комплекс программных и языковых средств, необходимых для создания баз данных,

поддержания их в актуальном состоянии и организации поиска в них необходимой информации.

Централизованный характер управления данными в базе данных предполагает необходимость существования некоторого лица (группы лиц), на которое возлагаются функции администрирования данными, хранимыми в базе.

По технологии обработки данных базы данных подразделяются на централизованные и распределенные.

Централизованная база данных хранится в памяти одной вычислительной системы. Если эта вычислительная система является компонентом сети ЭВМ, возможен распределенный доступ к такой базе. Такой способ использования баз данных часто применяют в локальных сетях ПК.

Распределенная база данных состоит из нескольких, возможно пересекающихся или даже дублирующих друг друга частей, хранимых в различных ЭВМ вычислительной сети. Работа с такой базой осуществляется с помощью системы управления распределенной базой данных (СУРБД).

По способу доступа к данным базы данных разделяются на базы данных с локальным доступом и базы данных с удаленным (сетевым) доступом.

Системы централизованных баз данных с сетевым доступом предполагают различные архитектуры подобных систем:

• файл-сервер;

• клиент-сервер.

Файл-сервер. Архитектура систем БД с сетевым доступом предполагает выделение одной из машин сети в качестве центральной (сервер, файлов). На такой машине хранится совместно используемая централизованная БД. Все другие машины сети выполняют функции рабочих станций, с помощью которых поддерживается доступ пользовательской системы к централизованной базе данных. Файлы базы данных в соответствии с

пользовательскими запросами передаются на рабочие станции, где в основном и производится обработка. При большой интенсивности доступа к одним и тем же данным производительность информационной системы падает. Пользователи могут создавать также на рабочих станциях локальные БД, которые используются ими монопольно.

Клиент-сервер. В этой концепции подразумевается, что помимо хранения централизованной базы данных центральная машина (сервер базы данных) должна обеспечивать выполнение основного объема обработки данных. Запрос на данные, выдаваемый клиентом (рабочей станцией), порождает поиск и извлечение данных на сервере. Извлеченные данные (но не файлы) транспортируются по сети от сервера к клиенту. Спецификой архитектуры клиент-сервер является использование языка запросов SOL.

Понятие базы данных тесно связано с такими понятиями структурных элементов, как поле, запись, файл (таблица).

Поле — элементарная единица логической организации данных, которая соответствует неделимой единице информации — реквизиту. Для описания поля используются следующие характеристики:

- имя, например. Фамилия, Имя, Отчество, Дата рождения;
- тип, например, символьный, числовой, календарный;
- длина, например, 15 байт, причем будет определяться максимально возможным количеством символов;
- точность для числовых данных, например два десятичных знака для отображения дробной части числа.

Запись — совокупность логически связанных полей. Экземпляр записи — отдельная реализация записи, содержащая конкретные значения ее полей.

Файл (таблица) — совокупность экземпляров записей одной структуры.

В структуре записи файла указываются поля, значения которых являются ключами первичными (ПК), которые идентифицируют экземпляр записи, и вторичными (ВК), которые выполняют роль поисковых или

группировочных признаков (по значению вторичного ключа можно найти несколько записей).

Ядром любой базы данных является модель данных. Модель данных представляет собой множество структур данных, ограничений целостности и операций манипулирования данными. С помощью модели данных могут быть представлены объекты предметной области и взаимосвязи между ними.

Модель данных — совокупность структур данных и операций их обработки.

СУБД основывается на использовании иерархической, сетевой или реляционной модели, на комбинации этих моделей или на некотором их подмножестве.

Рассмотрим три основных типа моделей данных: иерархическую, сетевую и реляционную.

Иерархическая структура представляет совокупность элементов, связанных между собой по определенным правилам. Объекты, связанные иерархическими отношениями, образуют ориентированный граф (перевернутое дерево).

К основным понятиям иерархической структуры относятся: уровень, элемент (узел), связь. Узел — это совокупность атрибутов данных, описывающих некоторый объект. На схеме иерархического дерева узлы представляются вершинами графа. Каждый узел на более низком уровне связан только с одним узлом, находящимся на более высоком уровне. Иерархическое дерево имеет только одну вершину (корень дерева), не подчиненную никакой другой вершине и находящуюся на самом верхнем (первом) уровне. Зависимые (подчиненные) узлы находятся на втором, третьем и т.д. уровнях. Количество деревьев в базе данных определяется числом корневых записей.

К каждой записи базы данных существует только один (иерархический) путь от корневой записи.

В сетевой структуре при тех же основных понятиях (уровень, узел, связь) каждый элемент может быть связан с любым другим элементом.

Понятие реляционный (англ. relation — отношение) связано с разработками известного американского специалиста в области систем баз данных Е. Кодда.

Эти модели характеризуются простотой структуры данных, удобным для пользователя табличным представлением и возможностью использования формального аппарата алгебры отношений и реляционного исчисления для обработки данных.

Реляционная модель ориентирована на организацию данных в виде двумерных таблиц. Каждая реляционная таблица представляет собой двумерный массив и обладает следующими свойствами:

- каждый элемент таблицы — один элемент данных;
- все столбцы в таблице однородные, т.е. все элементы в столбце имеют одинаковый тип (числовой, символьный и т.д.) и длину;
- каждый столбец имеет уникальное имя;
- одинаковые строки в таблице отсутствуют;
- порядок следования строк и столбцов может быть произвольным.

Отношения представлены в виде таблиц, строки которых соответствуют кортежам или записям, а столбцы — атрибутам отношений, доменам, полям.

Поле, каждое значение которого однозначно определяет соответствующую запись, называется простым ключом (ключевым полем). Если записи однозначно определяются значениями нескольких полей, то такая таблица базы данных имеет составной ключ.

Чтобы связать две реляционные таблицы, необходимо ключ первой таблицы ввести в состав ключа второй таблицы (возможно совпадение ключей); в противном случае нужно ввести в структуру первой таблицы внешний ключ — ключ второй таблицы.

Информационный объект — это описание некоторой сущности (реального объекта, явления, процесса, события) в виде совокупности логически связанных реквизитов (информационных элементов). Такими сущностями для информационных объектов могут служить: цех, склад, материал, вуз, студент, сдача экзаменов и т.д.

Информационный объект определенного реквизитного состава и структуры образует класс (тип), которому присваивается уникальное имя структуры образует класс (тип), которому присваивается уникальное имя (символьное обозначение), например Студент, Сессия, Стипендия.

Информационный объект имеет множество реализации — экземпляров, каждый из которых представлен совокупностью конкретных значений реквизитов и идентифицируется значением ключа (простого — один реквизит или составного — несколько реквизитов). Остальные реквизиты информационного объекта являются описательными. При этом одни и те же реквизиты в одних информационных объектах могут быть ключевыми, а в других - описательными. Информационный объект может иметь несколько ключей.

Одни и те же данные могут группироваться в таблицы (отношения) различными способами, т.е. возможна организация различных наборов отношений взаимосвязанных информационных объектов. Группировка атрибутов в отношениях должна быть рациональной, т.е. минимизирующей дублирование данных и упрощающей процедуры их обработки и обновления.

Определенный набор отношений обладает лучшими свойствами при включении, модификации, удалении данных, чем все остальные возможные наборы отношений, если он отвечает требованиям нормализации отношений.

Нормализация отношений — формальный аппарат ограничений на формирование отношений (таблиц), который позволяет устранить дублирование, обеспечивает непротиворечивость хранимых в базе данных, уменьшает трудозатраты на ведение (ввод, корректировку) базы данных.

Выделены три нормальные формы отношений и предложен механизм, позволяющий любое отношение преобразовать к третьей (самой совершенной) нормальной форме.

- Первая нормальная форма

Отношение называется нормализованным или приведенным к первой нормальной форме, если все его атрибуты простые (далее неделимы). Преобразование отношения к первой нормальной форме может привести к увеличению количества реквизитов (полей) отношения и изменению ключа.

Например, отношение Студент = (Номер, Фамилия, Имя, Отчество, Дата, Группа) наводится в первой нормальной форме.

- Вторая нормальная форма

Чтобы рассмотреть вопрос приведения отношений ко второй нормальной форме, необходимо дать пояснения к таким понятиям, как функциональная зависимость и полная функциональная зависимость.

Описательные реквизиты информационного объекта логически связаны с общим для них ключом, эта связь носит характер функциональной зависимости реквизитов.

Функциональная зависимость реквизитов — зависимость, при которой экземпляре информационного объекта определенному значению ключевого реквизита соответствует только одно значение описательного реквизита.

Такое определение функциональной зависимости позволяет при анализе всех взаимосвязей реквизитов предметной области выделить самостоятельные информационные объекты.

В случае составного ключа вводится понятие функционально полной зависимости.

Функционально полная зависимость не ключевых атрибутов заключается в том, что каждый не ключевой атрибут функционально зависит от ключа, но не находится в функциональной зависимости ни от какой части составного ключа.

Отношение будет находиться во второй нормальной форме, если оно находится в первой нормальной форме, и каждый не ключевой атрибут функционально полно зависит от составного ключа.

- Третья нормальная форма

Понятие третьей нормальной формы основывается на понятии нетранзитивной зависимости.

Транзитивная зависимость наблюдается в том случае, если один из двух описательных реквизитов зависит от ключа, а другой описательный реквизит зависит от первого описательного реквизита.

Отношение будет находиться в третьей нормальной форме, если оно находится во второй нормальной форме, и каждый неключевой атрибут нетранзитивно зависит от первичного ключа.

Для устранения транзитивной зависимости описательных реквизитов необходимо провести «расщепление» исходного информационного объекта. В результате расщепления часть реквизитов удаляется из исходного информационного объекта и включается в состав других (возможно, вновь созданных) информационных объектов.

Все информационные объекты предметной области связаны между собой. Различаются связи нескольких типов, для которых введены следующие обозначения:

- один к одному (1:1);
- один ко многим (1 : M);
- многие ко многим (M : M).

Связь один к одному (1:1) предполагает, что в каждый момент времени одному экземпляру информационного объекта A соответствует не более одного экземпляра информационного объекта B и наоборот.

При связи один ко многим (1:M) одному экземпляру информационного объекта A соответствует 0, 1 или более экземпляров объекта B, но каждый

экземпляр объекта B связан не более чем с 1 экземпляром объекта A. Графически данное соответствие имеет вид.

Связь многие ко многим (M:M) предполагает, что в каждый момент времени одному экземпляру информационного объекта A соответствует 0, 1 или более экземпляров объекта B и наоборот.

1.5. C++ Builder и базы данных

Базы данных считаются основным достоинством C++ Builder. Это действительно так. Хотя язык и не создавался специально под эту предметную область программирования, но реализация работы с данными здесь просто поражает. Даже специализированные языки, которые предназначены для работы с базами данных (такие, как MS Visual FoxPro), явно уступают C++ Builder по простоте и мощи программирования.

C++ Builder скрывает все сложности и в то же время предоставляет широчайшие возможности при создании баз данных. Практически любую задачу в этой предметной области можно реализовать средствами этого языка, причем за довольно короткий промежуток времени. Главное здесь то, что реализация приложения очень удобна и проста в понимании.

Для работы с базами в C++ Builder есть несколько наборов компонентов. Каждый набор очень хорошо подходит для решения определенного круга задач. Почему такое разнообразие компонентов? Все они используют разные технологии доступа к данным и отличаются по своим возможностям. Microsoft встроила в свои в свои продукты разработки только технологию доступа к данным ADO, собственной разработки. Фирма Borland предоставила разнообразные средства, работающие через разные технологии, и не ограничивает программиста только своими разработками. Такое

положение вещей дает громадные преимущества. Помимо этого, есть группы компонентов, которые могут использоваться в любом случае.

Произведем краткий обзор имеющихся средств доступа к базам данных.

На вкладке Data Access расположены основные компоненты доступа к данным. Эти компоненты общие для всех и могут использоваться совместно с другими группами компонентов.

На вкладке Dala Controls расположены компоненты для отображения и редактирования данных в таблицах. Эти компоненты так же используются в независимости от используемой на данный момент технологии доступа к данным.

Вкладка BDE содержит компоненты, позволяющие получить доступ к базам данных по технологии, разработанной фирмой Borland, под названием Borland Database Engine. Она хорошо работает базами данных, например, такими как Paradox и dBase.

DBExpress — это новая технология доступа к данным фирмы Borland. Она отличается большей гибкостью и хорошо подходит для программирования клиент-серверных приложений, использующих базы данных. Компоненты с одноименной вкладки желательно использовать с базами данных, построенных по серверной технологии, например, Oracle, DB2 или MySQL.

ADO (Active Data Objects) — технология доступа к данным, разработанная корпорацией Microsoft. Очень хорошая библиотека, но использовать ее желательно только с базами данных Microsoft, а именно MS Access или MS SQL Server. Ее также можно использовать, если у вас специфичный сервер баз данных, который может работать только через ODBC.

1.6. Выбор и обоснование инструментальных средств обработки

Для создания своего проекта была выбрана среда программирования C++ Builder.C++ Builder представляет собой SDI-приложение, главное окно которого содержит настраиваемую инструментальную панель (слева) и палитру компонентов (справа). Помимо этого, по умолчанию при запуске C++ Builder появляются окно инспектора объектов (слева) и форма нового приложения (справа). Под окном формы приложения находится окно редактора кода.

Формы являются основой приложений C++ Builder. Создание пользовательского интерфейса приложения заключается в добавлении в окно формы элементов объектов C++ Builder, называемых компонентами. Компоненты C++ Builder располагаются на палитре компонентов, выполненной в виде многостраничного блокнота. Важная особенность C++ Builder состоит в том, что он позволяет создавать собственные компоненты и настраивать палитру компонентов, а также создавать различные версии палитры компонентов для разных проектов. ++ Builder первоначально создавалась только для платформы MicrosoftWindows. Поздние версии, содержащие Кроссплатформенную компонентную библиотеку Borland, поддерживают и Windows и Linux.

В 2003 Borland выпустила C++ BuilderX (CBX), написанный при помощи той же инфраструктуры, что и JBuilder, который при этом был мало похож на C++ Builder или Delphi. Этот продукт предназначался для разработки больших программ для крупных предприятий, но коммерческого успеха не достиг. В конце 2004 года Borland объявила, что продолжит развитие классического C++ Builder и объединит его со средой разработки Delphi, прекратив, таким образом, разработку C++ BuilderX.

Спустя примерно год после этого объявления, Borland выпустила BorlandDeveloperStudio 2006, который включал в себя Borland C++ Builder 2006, предлагавший улучшенное управление конфигурацией и отладкой. BorlandDeveloperStudio 2006 - единственный полноценный комплект,

содержащий Delphi, C++Builder и C#Builder.

Используя Borland C++ Builder, можно создать приложения, работающие как с однопользовательскими базами данных (БД), так и с серверными системами управления базами данных (СУБД), такими как Oracle, Sybase, Informix, Interbase, MS SQL Server, DB2, а также с ODBC-источниками. Возможности C++ Builder, связанные с созданием приложений, использующих базы данных, весьма обширны для того, чтобы описать их в одной статье. Поэтому сегодня мы рассмотрим лишь простейшие возможности работы с таблицами баз данных.

Набор данных в C++ Builder является потомком абстрактного класса TDataSet (абстрактный класс - это класс, от которого можно порождать другие классы, но нельзя создать экземпляр объекта данного класса). Например, классы TQuery, TTable и TStoredProc, содержащиеся на странице палитры компонентов Data Access, - наследники TDBDataSet, который, в свою очередь, является наследником TDataSet. TDataSet содержит абстракции, необходимые для непосредственного управления таблицами или запросами, обеспечивая средства для того, чтобы открыть таблицу или выполнить запрос и перемещаться по строкам.

В типичных приложениях с базами данных 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:

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

2. ОСНОВНАЯ ЧАСТЬ. Технологическая часть.

2.1 Постановка задачи

1. Разработать программное обеспечение средств автоматизации ресторанного бизнеса на C++ Builder.
2. Программное обеспечение предназначено для регулирования количества продуктов, составления меню, расчета расходов для ресторана.

2.2. Руководство программисту

1.Цель: разработать программное обеспечение средств автоматизации ресторанного бизнеса .

2.Язык программирования: C++ Builder.

3.Утилита для создания структуры базы данных: Borland Database Desktop 7.0.

4.Движок баз данных: Borland Database Engine 5.0.

5.Тип базы данных: локальная.

6.Формат СУБД: Paradox 7.

1.Основные функции системы следующие:

- составление, просмотр, изменение существующих меню, распечатка меню на любой период;
- просмотр списка всех блюд, рецептов приготовления блюд, продуктов входящих в рецепт, с возможностью распечатки всех элементов;
- внесение изменения в уже существующее меню, рецепт, список продуктов;

- составление, просмотр, изменение существующих остатков и имеющееся остатков продуктов.

- составление сравнительного отчета между необходимым количеством продуктов на запланированное меню и имеющимся в наличии остатком.

- отображение в сравнительном отчете последней цены закупки каждого продукта.

2. Общим требованием системы является создание дружественного интерфейса, с помощью которого пользователь мог бы легко и быстро занести информацию в базу данных, по вышеперечисленным требованиям.

3. Программа должна выполнять следующие функции:

1. Добавление информации;
2. Правка информации;
3. Вывод информации;
4. Удаление информации;
5. Поиск информации;
6. Вывод информации на печать;

4.Минимальные аппаратные требования: P1, 32 Mb ОЗУ, mouse, клавиатура, 20 Mb свободного места на жестком диске;

5. Минимальные системные требования: Windows 98/ME/2000/XP, для работы программы требуется наличие BDE.

2.2.1. Модули

1. DataModule.cpp – Модуль данных;
2. AddNewUser. cpp – Модуль окна изменения данных таблиц “Sotrudniki”, “Login”, “Doljnosti”;
3. NewTovar. cpp – Модуль окна внесения и изменения данных таблицы “Tovar”, “Sklad”;
4. Main. cpp – Модуль главного окна программы;
5. Entertosystem. cpp – Модуль окна проверки значений при входе в систему;
6. TovarOnSklad. cpp – Модуль окна изменения данных таблиц “Sklad”, “Tovar”;
7. SpisSosklada. cpp - Модуль окна внесения и изменения данных таблицы “Sklad”, “Spisanie”;
8. Zakaz. cpp – Модуль внесения изменений в таблицы “Sklad”, “Ingrid”, “TempZakaz”,” SaveZakaz”, “JurnalZakazov”;
9. JurnalProdaj cpp - Модуль внесения, просмотра данных таблицы “JurnalProdaj”;
10. JurnalSpisaniy. cpp - Модуль внесения, просмотра данных таблицы “Spisanie”
11. Report. cpp – Модуль вывода отчётов;

DATAMODULE.CPP

Компоненты:

DSLogin:: TdataSource – Компонент связи набора данных с компонентом;

TableLogin:: Ttable - Таблица;

DSDoljnosti:: TDataSource– Компонент связи набора данных с компонентом;

TableDoljnosti:: TTable - Таблица;

DsPostavshiki:: TDataSource– Компонент связи набора данных с компонентом;

TablePostavshiki:: TTable - Таблица;

DsSklad:: TdataSource– Компонент связи набора данных с компонентом;

DsSpis:: TDataSource– Компонент связи набора данных с компонентом;

TableSklad:: TTable - Таблица;

TableSpis:: TTable - Таблица;

DsQuery:: TDataSource– Компонент связи набора данных с компонентом;

Query:: Tquery – Компонент, связи с запросом SQL;

DsTovar: TDataSource– Компонент связи набора данных с компонентом;

TableTovar:: TTable - Таблица;

DSSotrudniki:: TdataSource – Компонент связи набора данных с компонентом;

TableSotrudniki:: TTable; - Таблица;

Query1:: TQuery – Компонент, связи с запросом SQL;

DSQuery1:: TDataSource – Компонент связи набора данных с компонентом;

DSQuery2:: TDataSource – Компонент связи набора данных с компонентом;

Query2:: TQuery – Компонент, связи с запросом SQL;

DSTempZakaz:: TDataSource TDataSource – Компонент связи набора данных с компонентом;

TableTempZakaz:: Ttable - Таблица;

DataSource1:: TDataSource TDataSource – Компонент связи набора данных с компонентом;

DSSaveZakaz:: TDataSource – Компонент связи набора данных с компонентом;

TableSaveZakaz:: TTable - Таблица;

dsJurnalZakazov:: TDataSource – Компонент связи набора данных с компонентом;

TableJurnalZakazov:: TTable - Таблица;

ADDNEWUSER.CPP

Функции:

btnAddNewUserClick(TObject *Sender) – Щелчок на кнопке;

procedure btnCancelClick(TObject *Sender) – Щелчок на кнопке;

FormClose(TObject *Sender) – Закрытие формы;

FormShow(TObject *Sender) – Отображение формы;

Button3Click(TObject *Sender) – Щелчок на кнопке;

DBGrid1CellClick(TObject *Sender) - Щелчок на ячейке сетки;

CbFamSotrClick(TObject *Sender) – Щелчок по выпадающему списку;

CbLoginClick(TObject *Sender) – Щелчок по выпадающему списку;

btnNewPassClick(TObject *Sender) – Щелчок на кнопке;

btnPostClick(TObject *Sender) – Щелчок на кнопке;

BitBtn3Click(TObject *Sender) – Щелчок на кнопке;

BitBtn2Click(TObject *Sender) – Щелчок на кнопке;

FormKeyDown(TObject *Sender) – Событие нажатия клавиши на форме;

GLAV.CPP

Функции:

EnterToSystemClick(TObject *Sender) – Щелчок по кнопке;

AddLoginClick(TObject *Sender) - Щелчок по кнопке;

UsersClick(TObject *Sender) - Щелчок по пункту меню;

FormCreate TObject *Sender) – Создание формы;

FormClose(TObject *Sender) – Закрытие формы;

N7Click(TObject *Sender) – Выбор пункта меню;

N9Click(TObject *Sender) – Выбор пункта меню;

N11Click(TObject *Sender) – Выбор пункта меню;

N12Click(TObject *Sender) – Выбор пункта меню;

N3Click(TObject *Sender) – Выбор пункта меню;

N5Click(TObject *Sender) – Выбор пункта меню;

N14Click(TObject *Sender) – Выбор пункта меню;

ChangeUserClick(TObject *Sender) – Выбор пункта меню;

ExitClick(TObject *Sender) – Выбор пункта меню;

N15Click(TObject *Sender) – Выбор пункта меню;

FormShow(TObject *Sender) – Отображение формы;

N10Click(TObject *Sender) – Выбор пункта меню.

OpenTable(TObject *Sender) –Прописывание алиаса к базе данных.

ENTERTOSYSTEM.CPP

Функции

FormClose(TObject *Sender) – Закрытие формы;

btnCancelClick(TObject *Sender) – Щелчок на кнопке;

btnEnterClick(TObject *Sender) – Щелчок на кнопке;

FormShow(TObject *Sender) – Отображение формы;

FormKeyDown(TObject *Sender) – событие нажатия кнопки на форме.

TOVARONSKLAD.CPP

Функции

FormClose(TObject *Sender) – Закрытие формы;

btnBayClick TObject *Sender) – Щелчок на кнопке;

btnAllowancesClick(TObject *Sender) – Щелчок на кнопке;

btnJurnalPokupokClick(TObject *Sender) – Щелчок на кнопке;

btnAllowancesJornalClick(TObject *Sender) – Щелчок на кнопке;

Button2Click(TObject *Sender) – Щелчок на кнопке;

btnNextClick(TObject *Sender) – Щелчок на кнопке;

DBGrid1CellClick(TObject *Sender) – Щелчок по записи в сетке;

btnPreClick(TObject *Sender) – Щелчок на кнопке;

btnFirstClick(TObject *Sender) – Щелчок на кнопке;

btnLastClick(TObject *Sender) – Щелчок на кнопке;

EdNameKeyUp(TObject *Sender) – Событие нажатия клавиши на текстовое поле;

FormShow(TObject *Sender) – Отображение формы;

FormKeyDown(TObject *Sender) – Событие нажатия клавиши на форме.

NEWTOVAR.CPP

Функции

FormClose(TObject *Sender) – Закрытие формы;

CbproductEnter(TObject *Sender) – Щелчок на выпадающем списке;

PostavshikEnter(TObject *Sender) – Щелчок на выпадающем списке;

FormShow(TObject *Sender) – Отображение формы;

FormKeyDown(TObject *Sender) – Событие нажатия клавиши на форме.

SPISSOSKLADA. CPP

Функции

procedure FormClose(Sender TObject *Sender) – Закрытие формы;

btnCancelClick(TObject *Sender) – Щелчок на кнопке;

btnOkClick(TObject *Sender) – Щелчок на кнопке;

FormShow(TObject *Sender) – Отображение формы;

FormKeyDown(TObject *Sender) – Событие нажатия клавиши на форме.

ZAKAZ.CPP

Функции

FormShow(TObject *Sender) – Отображение формы;

btnGOClick(TObject *Sender) – Щелчок на кнопке;

btnInsertClick(TObject *Sender) – Щелчок на кнопке;

FormKeyDown(TObject *Sender) – Событие нажатия клавиши на форме;

WeekDay –Определение дня недели.

JURNALPRODAJ.CPP

Функции

DateTimePicker1Change(TObject *Sender) – Изменение даты;

FormShow(TObject *Sender) – Отображение формы;

SpeedButton1Click(TObject *Sender) – Щелчок на кнопке;

RadioButton1Click(TObject *Sender) – Щелчок на радиокнопке;

FormKeyDown(TObject *Sender) – Событие нажатия клавиши на форме;

Change – Изменение SQL-запроса.

JURNALSPISANIY.PAS

Функции

DateTimePicker1Change(TObject *Sender) – Изменение даты;

FormShow(TObject *Sender) – Отображение формы;

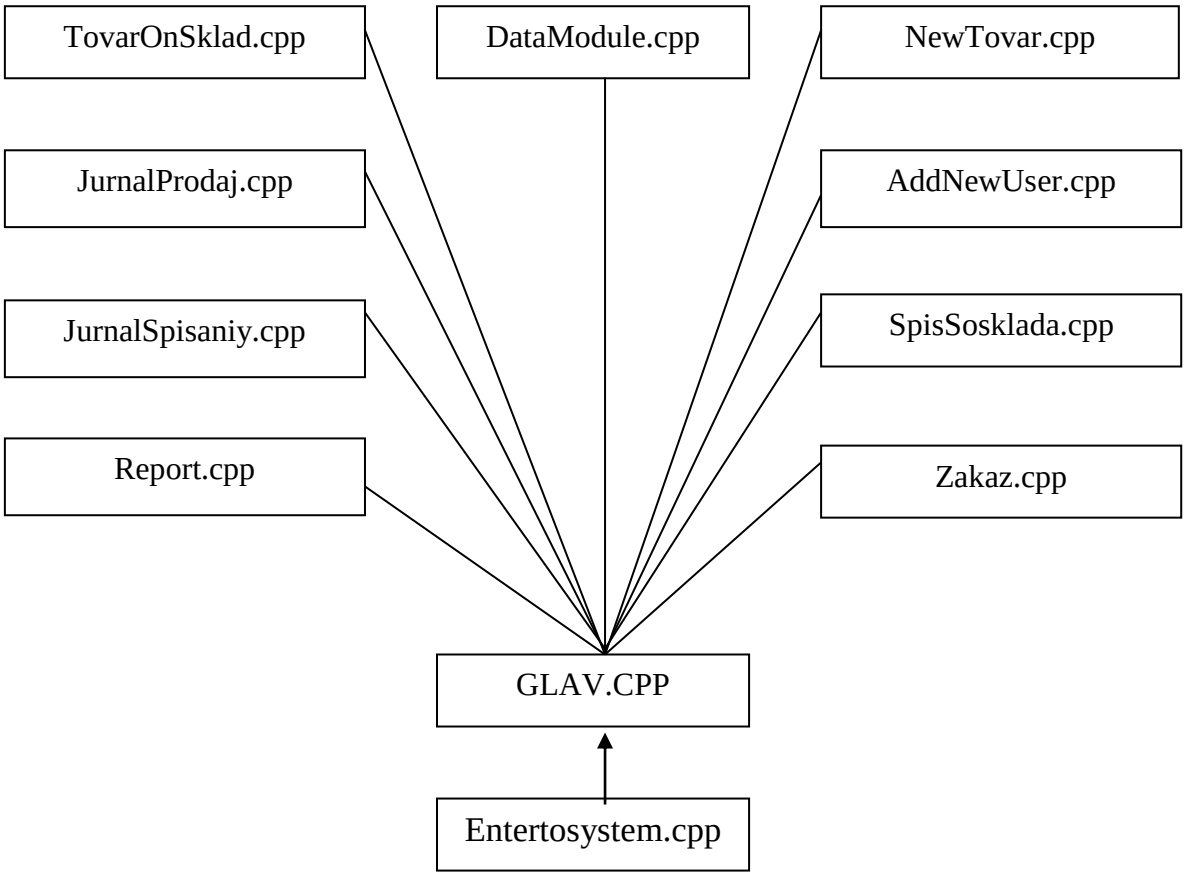
SpeedButton1Click(TObject *Sender) – Щелчок на кнопке;

RadioButton1Click(TObject *Sender) – Щелчок на радиокнопке;

FormKeyDown(TObject *Sender) – Событие нажатия клавиши на форме;

Change – Изменение SQL-запроса.

2.2.2. Взаимосвязь модулей



Программа состоит из файла проекта Mobile.cpp и восьми подключенных к нему модулей:

1. unMain.cpp – Главный модуль программы;
2. UAbout.cpp –Модуль о программе, выводит информацию о программе;
3. UDB.cpp – Модуль содержит основной функционал работы с Базой данных;
4. UDistrict.cpp – Содержит методы для добавления, редактирования и удаления
5. UDistrictsSelect.pas – Модуль содержит методы для работы со списком USpalsh.pas – Всплывающая подсказка;

2.3. ИНСТРУКЦИЯ ПОЛЬЗОВАТЕЛЮ

Для того, чтобы запустить программу запустите файл Restoran.exe.

Далее появится окно с надписью “Учет продуктов в ресторане: главное меню.”

В этом окне присутствуют три вкладки и кнопки управления.

Три вкладки: Файл, Управление, Справка.

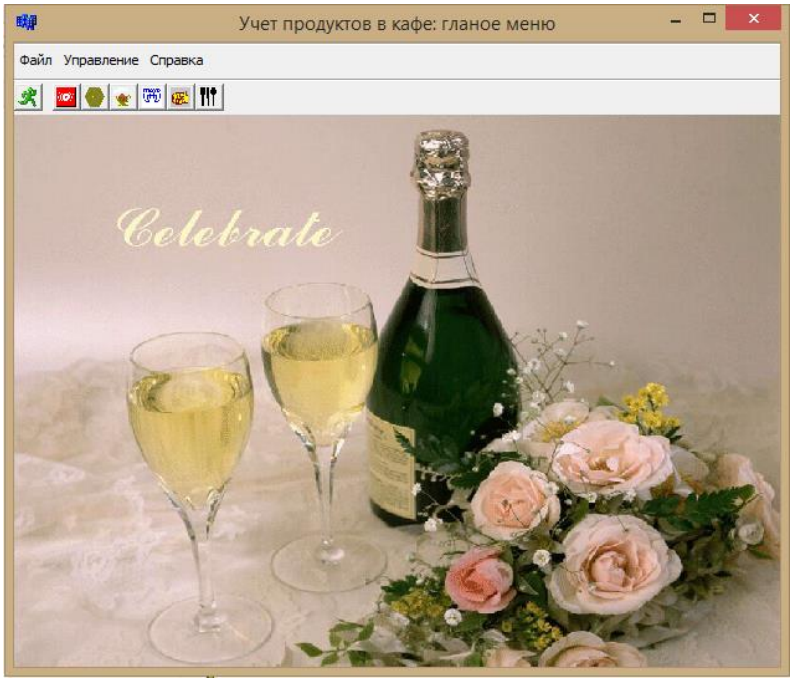


Рис.1

Нажав кнопку “Файл”, вы увидите функцию выхода.

Нажав кнопку “Управление”, вы получите доступ к вкладкам “Единица измерения”, ”Продукты”, ”Категории блюд”, ”Виды блюд”, ”Меню”, ”Список блюд”.

Во вкладке “Единица измерения” можно указывать в каких единицах измерений будут указываться продукты и их краткое название и добавлять свои единицы измерений.

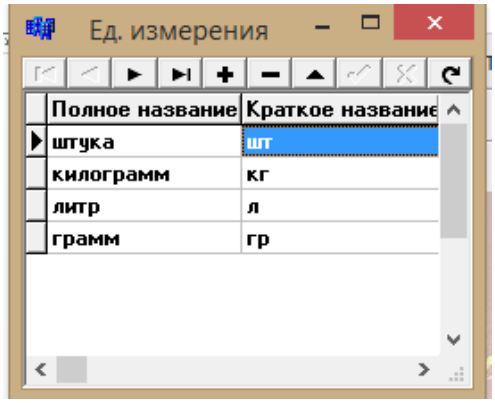


Рис.2

Во вкладке “Продукты” вы можете указывать какие продукты будут использоваться в приготовлении различных блюд, добавлять и редактировать по своему усмотрению. Также записываются закупки продуктов, то есть дата и цена покупки.

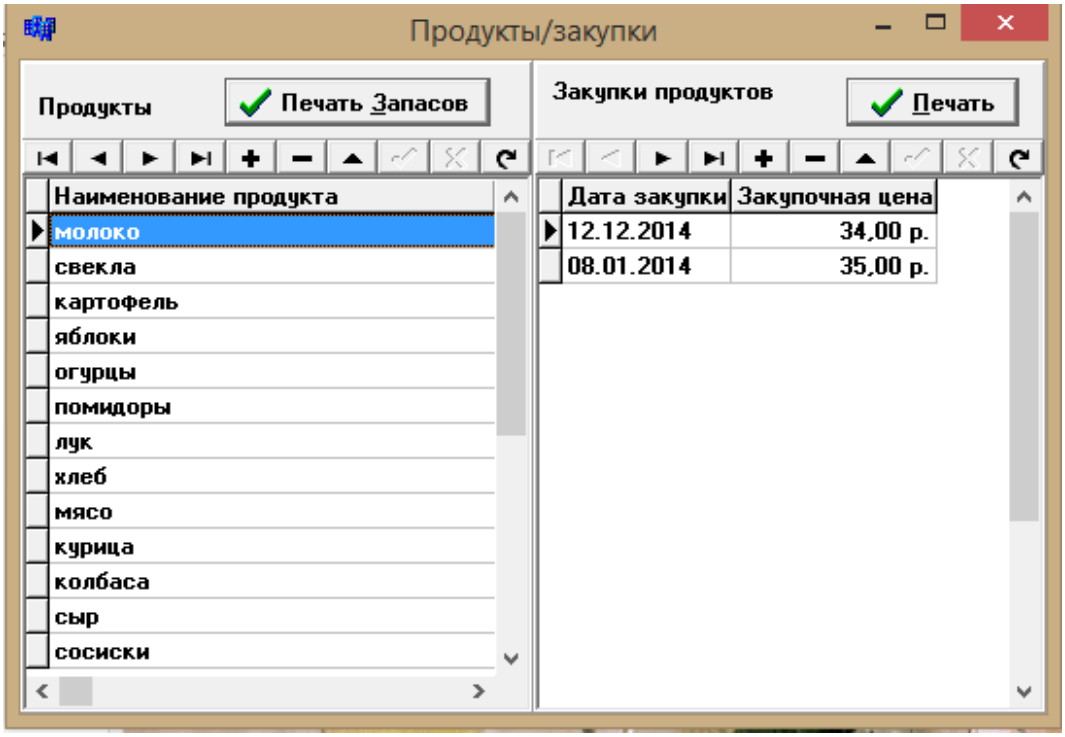


Рис.3

Во вкладке “Категории блюд” записывается список категорий меню. Список категорий меню: Завтрак, обед, полдник, ужин, ланч и так далее.

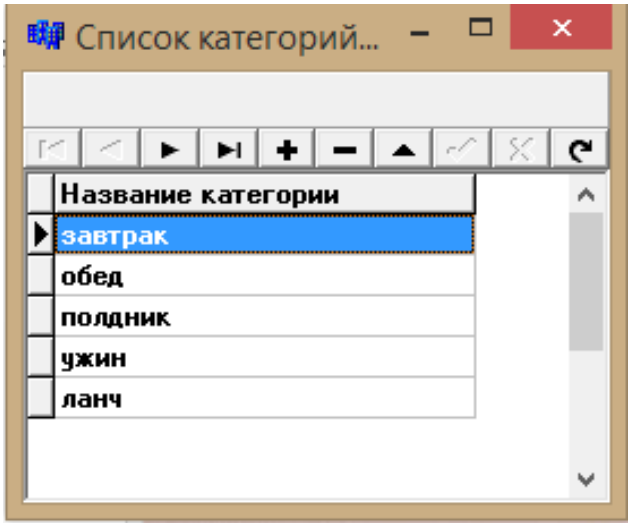


Рис.4

Во вкладке “Виды блюд” указывается вид блюда, то есть горячее, первое, десерт и далее по-вашему усмотрению.

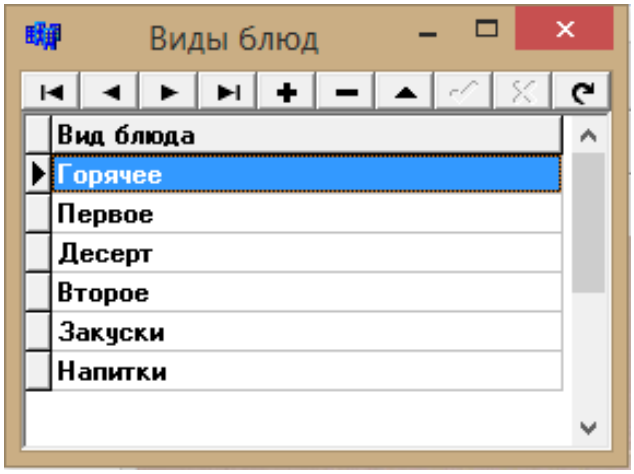


Рис.5

Во вкладке “Меню” указывается несколько параметров: Содержимое меню (завтрак, ланч); Список блюд (заливное, язык). Можно установить диапазон дат. Сформировать расходы по этим датам. Также расход продуктов на блюдо.

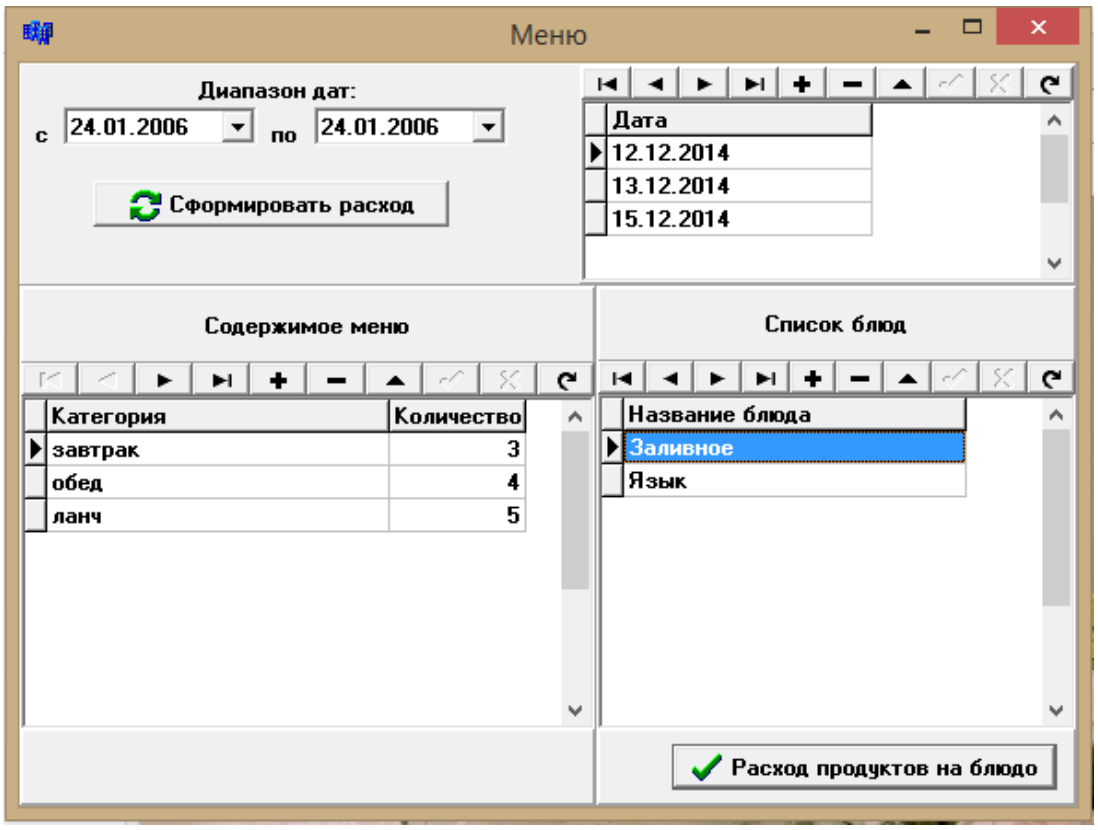


Рис.6

Во вкладке ”Список блюд” указывается вид блюда, название блюда и список затраченных на приготовление продуктов.

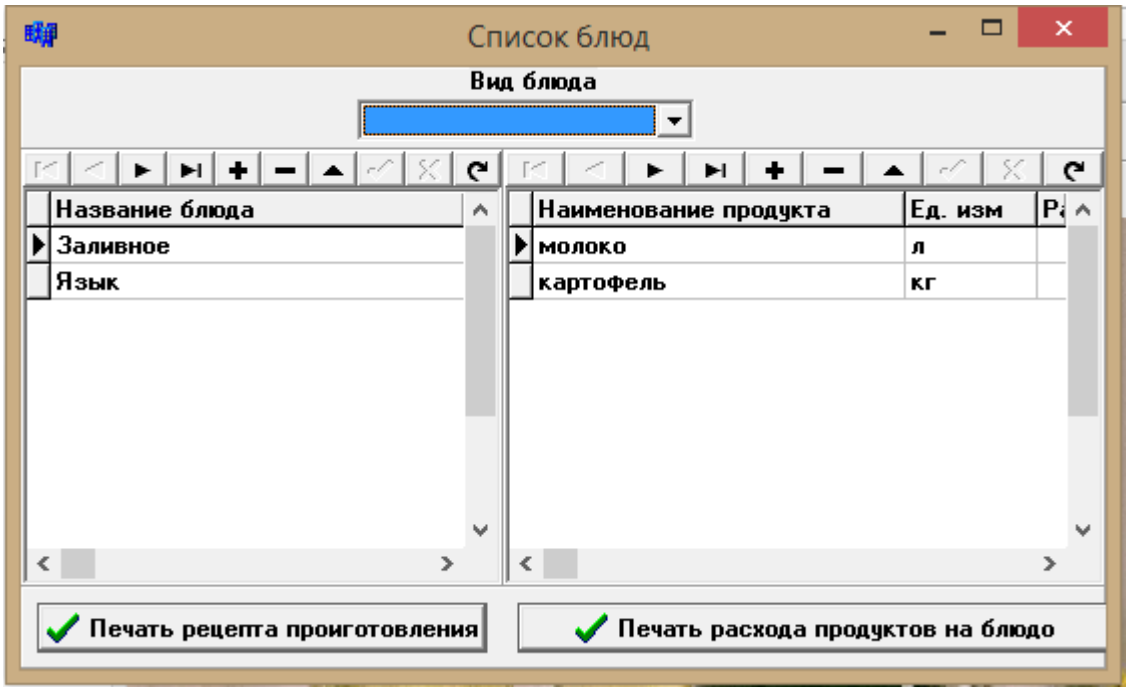


Рис.7

Нажав кнопку “Справка”, вы получаете информацию о самой программе и информацию об авторе.

Имеются горячие клавиши, по которым вы можете моментально обратиться к той или иной функции. “Единица измерения”, ”Продукты”, ”Категории блюд”, ”Виды блюд”, ”Меню”, ”Список блюд”, ”Выход”.

3.БЕЗОПАСНОСТЬ ЖИЗНЕДЕЯТЕЛЬНОСТИ

3.1. Рациональная организация рабочего места

Усложнение производственных процессов и оборудования изменили функции человека в современном производстве: возросла ответственность решаемых задач; увеличился объем информации, воспринимаемой работающим и быстродействие оборудования. Работа человека стала сложнее, возросла нагрузка на нервную систему и снизилась нагрузка физическая. В ряде случаев человек стал наименее надежным звеном системы «человек-машина». Возникла задача обеспечения надежности и безопасности работы человека на производстве. Эту задачу решает эргономика и инженерная психология.

Эргономика (от греческого ergon - работа и nomos - закон) - научная дисциплина, изучающая человека в условиях его деятельности, связанной с использованием машин. Цель эргономики - оптимизация условий труда в системе "человек-машина" (СЧМ). Эргономика определяет требования человека к технике и условия ее функционирования. Эргономичность техники является наиболее обобщенным показателем свойств и других показателей техники.

Связь человека с окружающей средой и параметрами рабочего места.

Рабочее место - это зона, в которой совершается трудовая деятельность исполнителя или группы исполнителей. Рабочие места могут быть индивидуальными и коллективными, универсальными, специализированными и специальными.

Общие требования, которые должны соблюдаться при проектировании рабочих мест, следующие:

- достаточное рабочее пространство для человека;
- оптимальное положение тела работающего;
- достаточные физические, зрительные и слуховые связи между человеком и машиной;

- оптимальное размещение рабочего места в помещении;
- допустимый уровень действия факторов производственных условий; оптимальное размещение информационного и моторного поля;
- наличие средств защиты от производственных опасностей.

Конструирование должно обеспечивать зоны оптимальной и легкой досягаемости моторного поля рабочего места и оптимальную зону информационного поля рабочего места. Угол обзора по отношению к горизонтали должен составлять 30-40°.

Выбор рабочего положения должен учитывать усилия, затрачиваемые человеком, размах движений, необходимость перемещений, темп операций. Выбор рабочей позы должен учитывать физиологию человека, а параметры рабочего места определяются выбором положения тела при работе (сидя, стоя, переменно).

Рабочие места для выполнения работ «сидя» организуются при легкой работе и средней тяжести, а при тяжелой - рабочая поза -"стоя".

В конструкции оборудования и организации рабочего места необходимо предусматривать возможности регулирования отдельных элементов, чтобы обеспечить оптимальное положение работающего.

Проектирование оборудования должно обеспечить его соответствие антропометрическим и биомеханическим характеристикам человека на основе учета динамики изменения размеров тепла при его перемещении, диапазона движений в суставах.

Для учета в конструкции оборудования антропометрических данных необходимо:

- определить контингент людей, для которых предназначено оборудование;
- выбрать группу антропометрических признаков;
- установить процент работающих, которому должно удовлетворять оборудование;
- определить границы интервала размеров (усилий), которые должны

быть реализованы в оборудовании.

При проектировании используют антропометрические размеры тела, причем учитываются различия в размерах тела мужчин и женщин, национальные, возрастные, профессиональные. Для определения границ интервалов, в которых учитывается процент населения, используется система перцетелей. Конструкция оборудования должна обеспечивать возможность использования по меньшей мере для 90% потребителей.

Для работы в положении "сидя" используются различные рабочие сиденья. Различают рабочие сиденья для длительного и кратковременного пользования. Общие требования для сидений длительного пользования следующие: сидение должно обеспечивать позу, уменьшающую статистическую работу мышц; создавать условия для возможности изменения рабочей позы; не затруднять деятельность систем организма; обеспечивать свободное перемещение относительно рабочей поверхности, иметь регулируемые параметры; иметь полумягкую обивку. Для кратковременного пользования рекомендуются жесткие стулья и различного типа табуреты.

В условиях растущей механизации и автоматизации производственных процессов особое значение приобретают средства отображения информации об объекте управления. Широкое использование получила информационная модель, то есть организованная по определенным правилам информация о состоянии объекта управления. К информационным моделям предъявляются следующие требования:

- содержание информационной модели должно адекватно отображать объект управления;
- информационная модель должна обеспечивать оптимальный информационный баланс;
- форма и композиция информационной модели должна соответствовать задачам трудового процесса и возможностям человека по приему информации.

Практика позволяет наметить последовательность разработки информационной модели: определение задач системы, очередность их решения и источников информации; составление перечня объектов управления и их признаков; распределение объектов по степени важности; распределение функции между автоматикой и человеком; выбор системы кодирования объектов и составление общей композиции модели; определение исполнительных действий человека.

В процессе конструирования информационной модели определяются места размещения средств информации на рабочем месте, выбираются размеры знаков и компоновка. Средства отображения размещаются в поле зрения наблюдателя с учетом оптимальных углов и зон наблюдения. Размеры знаков наблюдения определяются с учетом максимальной точности и скорости восприятия информации, а также яркости знаков, величины контраста, использования цвета. Оптимальной яркостью считаются значения, при которых обеспечивается максимальная контрастная чувствительность. Величина ее будет тем больше, чем меньше размер объекта различения. Оптимальная зона величины контраста равна 60-90%.

В работе глаза имеет место определенная инерционность, что требует учета времени экспозиции зрительного сигнала и временных интервалов для ощущения раздельности сигналов следующих один за другим. В большинстве случаев время экспозиции сигнала должно быть не менее 50 мс. Каждая разновидность индикаторов имеет свою область использования: индикаторы с подсветкой применяются для отображения качественной информации, требующей немедленной реакции оператора; стрелочные индикаторы используются для чтения измеряемых параметров; интегральные индикаторы для совмещения информации сразу о нескольких параметрах.

При проектировании рабочего места должны учитываться правила экономики движений: при работе двумя руками движения их должны быть

одновременными и симметричными; движения должны быть плавными и закругленными, ритмичными **и** привычными для работающего. Конструкция оборудования должна учитывать правила, касающиеся скорости и точности рабочих движений. Например, наиболее быстрое движение к себе; в горизонтальной плоскости скорость рук больше, чем ввертикальной; точность движений лучше в положении сидя, чем стоя и т.д. Органы управления, используемые на рабочем месте, должны соответствовать общим требованиям эргоногетики: направление движения органов управления должно соответствовать движению связанного с ним индикатора; соответствие расположения органов управления последовательности работы оператора; удобство использования; создание в органах правления механического сопротивления и т.п. Помимо этого, к каждому виду органов давления соответствует своя область использования и особые требования к размерам, форме, усилию и т.п.

На автоматизированном рабочем месте оператора-связиста (оператор в диспетчерской) в общем случае используются:

- средства отображения информации индивидуального пользования (блоки отображения, устройства сигнализации и так далее);
- средства управления и ввода информации (пульт дисплея, клавиатура управления, отдельные органы управления и так далее);
- устройства связи и передачи информации (модемы, телеграфные и телефонные аппараты):
- устройства документирования и хранения информации (устройства печати, магнитной записи и так далее);
- вспомогательное оборудование (средства оргтехники, хранилища для носителей информации, устройства местного освещения).

На автоматизированном рабочем месте должна быть обеспечена информационная и конструктивная совместимость используемых технических средств, антропометрических и психофизиологических характеристик человека.

При организации рабочего места должны быть учтены не только факторы, отражающие опыт, уровень профессиональной подготовки, индивидуально-личностные свойства операторов-связистов, но и факторы, характеризующие соответствие форм, способов представления и ввода информации психофизиологическим возможностям человека.

При оптимизации процедур взаимодействия операторов-связистов с техническими средствами в условиях автоматизации эргономические факторы выступают в качестве основных, обуславливающих вероятностно-временные характеристики и напряженность работы. Эти факторы являются чувствительными к вариациям индивидуально-личностных свойств оператора.

В комплекте рабочей мебели большое значение имеет конструкция производственного стула, так как от него зависит поза работника, а следовательно, и затрата энергии и степень его утомляемости. Рабочее сиденье должно иметь требуемые размеры, соответствующие антропометрическим данным человека и быть подвижным. Наиболее удобны стулья и кресла с регулируемым наклоном спинки и высотой сиденья. Изменяя высоту сиденья от уровня пола и угол наклона спинки, можно найти положение, наиболее соответствующее трудовому процессу и индивидуальным особенностям работника.

Как правило, все поверхности письменных и рабочих столов должны быть на уровне локтя при рабочем положении человека. При выборе высоты стола необходимо учитывать сидит человек во время работы или стоит.

Неудобная высота стола снижает эффективность работы и вызывает быстрое утомление. Отсутствие достаточного пространства для коленей и ступней вызывает постоянное раздражение работника. Минимальная рабочая высота стола должна быть не менее 725 мм. Как показывает практика, для рабочего среднего роста высота рабочего стола принимается 800 мм. Для работника другого роста можно изменить высоту

рабочего стула или положение его подножки так, чтобы расстояние от предмета обработки до глаз рабочего по высоте было равным примерно 450 мм.

Дисплей должен размещаться на столе или подставке так, чтобы расстояние наблюдения на экране не превышало 700 мм (оптимальное расстояние 450 – 500 мм). Экран дисплея по высоте должен быть расположен так, чтобы угол между центром экрана и горизонтальной линией взгляда составлял 200. В горизонтальной плоскости угол наблюдения экрана не должен превышать 600. Пульт дисплея должен быть размещен на столе или подставке так, чтобы высота клавиатуры пульта по отношению к полу составляла 650 - 720 мм. При размещении пульта на стандартном столе высотой 750 мм необходимо использовать кресло с регулируемой высотой сиденья (450 - 380 мм) и подставку для ног.

Документ (бланк) для ввода оператором данных рекомендуется располагать на расстоянии 450 - 500 мм от глаза оператора, преимущественно слева, при этом угол между экраном дисплея и документом в горизонтальной плоскости должен составлять 30 40°. Угол наклона клавиатуры должен быть равен 15°.

Экран дисплея, документы и клавиатура пульта дисплея должны быть расположены так, чтобы перепад яркостей поверхностей, зависящий от их расположения относительно источника света, не превышал 1 : 1 0 (рекомендуемое значение

1 : 3). При номинальных значениях яркостей изображения на экране 50 - 100 кд/м2 освещенность документа должна составлять 300 - 500 лк.

Рабочее место следует оборудовать таким образом, чтобы движения работника были бы наиболее рациональные, наименее утомительные.

Устройства документирования и другие, нечасто используемые технические средства, рекомендуется располагать справа от оператора в зоне максимальной досягаемости, а средства связи слева, чтобы освободить правую руку для записей.

3.2. ЧРЕЗВЫЧАЙНЫЕ СИТУАЦИИ

В теории БЖД ЧС — это совокупность событий, результат наступления которых характеризуется одним или несколькими из следующих признаков:

а) опасность для жизни и здоровья значительного числа людей;

б) существенное нарушение экологического равновесия в районе чрезвычайной ситуации;

в) выход из строя систем жизнеобеспечения и управления, полное или частичное прекращение хозяйственной деятельности;

г) значительный материальный и экономический ущерб;

д) необходимость привлечения больших, как правило, внешних по отношению к району ЧС сил и средств для спасения людей и ликвидации последствий;

е)психологический дискомфорт для больших групп людей.

Характерно, что ЧС возникает внешне неожиданно, внезапно. Конкретизация определения ЧС достигается введением количественных мер опасностей.

Классификация ЧС.

По причинам ЧС бывают природные, техногенные, антропогенные, экологические, социальные.

К **природным (стихийным)** ЧС относятся опасные природные явления или процессы, имеющие чрезвычайный характер и приводящие к нарушению повседневного уклада жизни более или менее значительных групп населения, человеческим жертвам, уничтожению материальных ценностей. К ним относятся землетрясения, наводнения, цунами, извержения вулканов, селевые потоки, оползни, обвалы, ураганы и смерчи, массовые лесные и торфяные пожары, снежные заносы и лавины. К числу стихийных бедствий относятся также засухи, длительные проливные

дожди, сильные устойчивые морозы, эпидемии, эпизоотии, эпифитотии, массовое распространение вредителей лесного и сельского хозяйства.

Стихийные бедствия могут происходить: в результате быстрого перемещения вещества (землетрясения, оползни); в процессе высвобождения внутриземной энергии (вулканическая деятельность, землетрясения); при повышении общего уровня рек, озер и морей (наводнения, цунами); под воздействием необычайно сильного ветра (ураганы, циклоны). Некоторые стихийные бедствия (пожары, обвалы, оползни и др.) могут возникнуть в результате действий самих людей, но последствия их всегда являются результатом действия сил природы. Для каждого стихийного бедствия характерно наличие присущих ему поражающих факторов, неблагоприятно воздействующих на состояние здоровья человека.

Стихийные бедствия являются трагедией всего государства и, особенно, для тех районов, где они возникают. В результате стихийных бедствий страдает экономика страны, так как при этом разрушаются производственные предприятия, уничтожаются материальные ценности и, самое главное, возникают потери среди людей, гибнет их жилье и имущество. Кроме того, стихийные бедствия создают крайне неблагоприятные условия для жизни населения, что может быть причиной вспышек массовых инфекционных заболеваний. Количество людей, пострадавших от стихийных бедствий, может быть весьма значительным, а характер поражений очень разнообразным. Больше всего люди страдают от наводнений (40% от общего урона), ураганов (20%), землетрясений и засух (по 15%). Около 10% общего ущерба приходится на остальные виды стихийных бедствий.

Ряд советских и зарубежных специалистов, приводя данные о потерях при крупнейших бедствиях, предполагают, что в будущем в связи с ростом и концентрацией населения аналогичные по силе катастрофы будут сопровождаться увеличением числа жертв в десятки раз.

Техногенными ЧС принято считать внезапный выход из строя машин, механизмов и агрегатов во время их эксплуатации, сопровождающийся серьезными нарушениями производственного процесса, взрывами, образованием очагов пожаров,радиоактивным, химическим или биологическим заражениям больших территорий, групповым поражениям (гибелью) людей.К техногенным ЧС относятся аварии на промышленных объектах, строительстве, а также на железнодорожном, воздушном, автомобильном, трубопроводном и водном транспорте, в результате которых образовались пожары, разрушения гражданских и промышленных зданий, создалась опасность радиационного загрязнения, химического и бактериального заражения местности, произошло растекание нефтепродуктов и агрессивных (ядовитых) жидкостей на поверхности земли и воды и возникли другие последствия, создающие угрозу населению и окружающей среде.

Характер последствий техногенных катастроф зависит от вида аварии, ее масштабов и особенностей предприятия, на котором возникла авария (от вида транспорта и обстоятельств, при которых произошла авария).

Антропогенные ЧС являются следствием ошибочных действий персонала. Этот класс ЧС может происходить на тех же объектах, что и техногенные ЧС. Отличие состоит лишь в том, что техногенные ЧС не связаны с человеческим фактором непосредственно.

К чрезвычайным ситуациям экологического характера можно отнести: интенсивную деградацию почвы и ее загрязнение тяжелыми металлами (кадмий, свинец, ртуть, хром и т. д.) и другими вредными веществами; загрязнение атмосферы вредными химическими веществами, шумом, электромагнитными полями; кислотные дожди; разрушение озонового слоя и т. д.

К **социальным ЧС** относятся события, происходящие в социуме (грабежи, насилия), межнациональные конфликты, сопровождающиеся

применением силы; противоречия между государствами с применением оружия.

По скорости распространения опасности ЧС могут быть классифицированы на: внезапные (землетрясения, взрывы, транспортные аварии и т. д.); стремительные (пожары, гидродинамические аварии с образованием волны прорыва, аварии с выбросом газообразных СДЯВ ит. д.); умеренные (паводковые наводнения, извержения вулканов, аварии с выбросом радиоактивных, веществ); плавные с медленно распространяющейся опасностью (засухи, эпидемии, аварии на промышленных очистных сооружениях, загрязнение почвы и воды вредными химическими веществами и т. д.).

По масштабности ЧС можно подразделить на пять типов: локальные (объектовые), местные, региональные, национальные и глобальные. При локальных (объектовых) ЧС последствия ограничиваются пределами объекта народного хозяйства и могут быть устранены за счет его сил и ресурсов.

Местные ЧС имеют масштабы распространения в пределах населённого пункта, в том числе крупного города административного района, нескольких районов или области и могут быть устранены за счет сил и ресурсов области.

В региональных ЧС последствия ограничиваются пределами нескольких областей или экономического района и могут быть ликвидированы за счет сил и ресурсов республики. Национальные ЧС имеют последствия, охватывающие несколько экономических районов или республик, но не выходящие за пределы страны. Ликвидация таких ЧС осуществляется силами и ресурсами государства, зачастую с привлечением иностранной помощи.

При глобальной ЧС ее последствия выходят за пределы страны и распространяются на другие государства. Эти последствия устраняются

как силами каждого государства на своей территории, так и силами международного сообщества.

Последствия ЧС могут быть самыми разнообразными. Они зависят от вида, характера чрезвычайной ситуации и масштаба ее распространения.

Основными видами последствий ЧС являются: гибель, заболевания людей, разрушения, радиоактивное загрязнение, химическое заражение, бактериальное заражение. Следует подчеркнуть, что на людей, находящихся в экстремальных условиях ЧС, наряду с различными поражающими факторами действуют и психотравмирующие обстоятельства, представляющие собой обычно комплекс сверхсильных раздражителей, вызывающих нарушение психической деятельности в виде так называемых реактивных (психогенных) состояний. При этом психогенное воздействие экстремальных условий складывается не только из прямой, непосредственной угрозы жизни человека, но и опосредованной, связанной с ожиданием ее реализации вне зон поражения. Если радиусы воздействия опасных и вредных факторов ЧС можно с той или иной, степенью достоверности определить заблаговременно расчет путем, то радиус психологического воздействия в реальной действительности может иметь самые различные значения. В ряде случаев он, возможно, будет во много раз превосходить радиусы воздействия других поражающих факторов.

Независимо от происхождения и типа в развитии чрезвычайных ситуаций можно выделить четыре характерных стадии (фазы): зарождения, инициирования, кульминационную и затухания (ликвидации последствий).

Установить продолжительность стадии зарождения, причем весьма приблизительно, можно только с помощью регулярной статистики отказов, сбоев, «локальных» аварий, данных наблюдений сейсмических, метереологических, противоселевых и других станций.

3.3. Модели производственных процессов с точки зрения экологии

Любой производственный процесс представляет собой некоторую систему, органически связанную с внешней средой. Такая производственная система получает из окружающей среды исходное сырье, материалы, энергию, а отдает в нее готовую продукцию и всевозможные отходы. Функционирование системы осуществляется благодаря потоку энергии, подводимой извне (электрической, солнечной и т.п.) либо генерируемой внутри системы за счет физико-химических процессов. К отходам относятся все вещества и материалы, тепловые выбросы, физические и биологические агенты, которые попадают во внешнюю среду и в дальнейшем уже не участвуют в получении продукции или энергии.

Если пользоваться представлениями термодинамики, то, как и все системы, технологические процессы в принципе подразделяются на три категории: незамкнутые (открытые), замкнутые и изолированные. Абсолютное большинство реальных технологических процессов относятся к категории *незамкнутых*. *Замкнутыми* считаются такие системы, у которых отсутствует обмен с внешней средой веществом, но возможен обмен энергией. Технологическим аналогом замкнутой системы может служить такой процесс, в котором полностью отсутствуют отходы химических веществ - твердые, жидкие и газообразные выбросы. Например, конечная сборка изделия из готовых деталей. При этом обмен с внешней средой исходным сырьем и готовой продукцией во внимание не принимается, хотя продукцию также можно рассматривать как отложенный отход. Теоретически возможны и *изолированные* процессы, которые не дают ни материальных, ни энергетических отходов.

В общем случае все технологические процессы можно рассматривать с точки зрения их экологического соответствия. Относительно *экологичными* можно считать такие технологические процессы и производства, воздействие которых на окружающую среду в рамках определенных количественных соотношений не нарушает нормального функционирования природных

экосистем. *Не экологичные* техпроцессы создают повышенную техногенную нагрузку и оказывает негативное воздействие на состояние окружающей природной среды.

Не экологичным может быть любой технологический процесс. Так, замкнутый техпроцесс, не имеющий отвода химических веществ в окружающую среду, нельзя считать экологичным, если он сопровождается вредными физическими воздействиями: тепловыми выбросами, шумами, электромагнитными полями и т.п.

Экологичность производственных процессов можно оценить с помощью *метода сырьевых балансов,* который основан на законах сохранения: *масса всех используемых ресурсов (сырья, топлива, воды и т.п.) в конечном итоге равна массе готовых продуктов и промышленных отходов.* Рассмотрим схемы материальных потоков в производствах разной степени замкнутости (рис. 10.2). Приняты следующие обозначения:

R -поток ресурсов (исходное сырье, основные и вспомогательные материалы, полуфабрикаты);

W -поток отходов (химические вещества и энергия), загрязняющий среду и уносящий определенную часть полезных ресурсов;

W_y - поток уловленных отходов;

P - поток готовой продукции.

Незамкнутому производственному процессу соответствует следующее уравнение материально-технического баланса:

$$R = P + W = (R - W_y) + W$$

Скобки в уравнении указывают на единство потока (ресурсов и отходов). «Отходность производства» можно оценить по коэффициенту $К_{отх} = W/R$. Соответственно коэффициент безотходности $К_б = P/R$. Производственный процесс, предусматривающий очистку загрязняющих потоков, представлен схемой 10.2, Б, а при использовании уловленных

веществ W_y в качестве вторичного сырья - схемой 10.2, В. В последнем случае материально-технический баланс описывается системой уравнений:

$$(R + W_y) = (R + W_y - W) + W;$$
$$W = (W - W_y) + W_y.$$

В замкнутом производственном цикле происходит полная переработка и утилизация потока отходов W_y, который вновь возвращается в сферу производства. Здесь потоки Wи W_y количественно равны, а поток готовой продукции *P* соответствует потоку *R*.

В ряде работ рассматриваются математические модели экологичности техпроцессов с различными схемами входных, промежуточных и выходных потоков. В качестве характеристик потоков принимаются не только массовые расходы вещества, но и его концентрации, температура, давление, расход тепла и другие физические параметры, связанные между собой балансовыми уравнениями. Методы моделирования производственных процессов оказываются полезными при решении задач оптимизации технологий по экологическим критериям.

Экологизация и снижение природоемкости производства предполагают сокращение валового внесения в природную среду техногенных эмиссии. Сделать производство полностью безотходным невозможно. Задача вовсе не сводится к тому, чтобы устранить абсолютно все экологически отрицательные последствия производственных процессов. Ставить такую задачу равносильно намерению изобрести вечный двигатель второго рода - безэнтропийный. Условно безотходными могут быть только отдельные стадии технологического цикла производства. Тем не менее, существуют теории безотходных процессов и отдельные положения, касающиеся этой проблемы.

Так, согласно определению, принятому на семинаре Европейской экономической комиссии ООН по малоотходной технологии (Ташкент,

1984), *«безотходная технология - это такой способ производства продукции (процесс, предприятие, территориально-производственный комплекс), при котором наиболее рационально и комплексно используются сырье и энергия в цикле «сырьевые ресурсы - производство - потребление - вторичные сырьевые ресурсы* таким образом, что любые воздействия на окружающую среду не нарушают ее нормального функционирования».*

Иногда, особенно в зарубежной литературе, употребляется термин «чистое производство», под которым понимают технологическую стратегию, предотвращающую загрязнение окружающей среды и понижающую до минимума риск для людей и окружающей среды. Применительно к процессам - это рациональное использование сырья и энергии, исключение применения токсичных сырьевых материалов, уменьшение количества и степени токсичности всех выбросов и отходов, образующихся в процессе производства. С точки зрения продукции чистое производство означает уменьшение ее воздействия на окружающую среду в течение всего жизненного цикла продукта от добычи сырья до утилизации (или обезвреживания) после использования. Чистое производство достигается путем улучшения технологии, применением ноу-хау и/или улучшением организации производства. Отметим, что эти определения не подразумевают возможности полной безотходности производства.

ЗАКЛЮЧЕНИЕ

С каждым днем все глубже и шире проникает информация во все сферы человеческой деятельности, способствуя тем самым их развитию, совершенствованию и переходу на качественно новый уровень. Программный продукт, разрабатываемый в выпускной квалификационной работе, позволяет четко наладить делопроизводство ресторанного бизнеса.

Результатом работы является программа, представляющая собой электронное хранилище данных о продуктах и рецептах блюд, подаваемых в ресторане, а также штатное расписание сотрудников заведения и количество мест для посетителей. Предлагаемый проект разработан для работы, как на локальном, так и на глобальном уровне, то есть имеется возможность передачи информации по сетям. Используя данное приложение можно самостоятельно вести базу данных.

Актуальностью предложенной задачи является возможность прозрачного просмотра состояния ресторана на текущий момент времени.

СПИСОК ЛИТЕРАТУРЫ

1. Каримов И.А. Узбекистан на пороге XXI века: угрозы безопасности, условия стабильности и гарантии прогресса. Ташкент: Узбекистан, 1997

2. И. А. Каримов «Узбекистон XXI аср бусагасида: хавфсизликка тахдид, баркарорлик шартлари ва тараккиёт кафолатлари». «Узбекистон» 1997.

3. И. А. Каримов «Узбекистон XXI асрга интилмоқда», «Узбекистон»,1999.

4. Узбекистон Республикаси Консгитуцияси. Т, 1998.

5. Ўзбекистон Республикасининг 1998 йил 1 май қонуни тахририда-Ўзбекистон Республикаси Олий Мажлисининг ахборотномаси, 1998 йил 5-6 сон, 102 модда.

6. «Ахолини ва худуддарни табиий ва техноген хусусиятли фавкулодда вазиятларда мухофаза килиш тугрисида» 20.08.1999 йил кабул килинган Узбекистон Республикаси конуни.

7. Экология и безопасность жизнедеятельности: Учебное пособие для студентов ВУЗов/ ред. Л. А. Муравий, 2002.

8. ЁрматовҒ.Ё., Исамухамедов Ё.У. Меҳнатни мухофаза қилиш. Дарслик. Ўзбекистан нашриёти. Тошкент 2002.

9. Белов С.В. Безопасность жизнедеятельности М.: Высшая школа. 2003.

10. Смайли Джон. Учимся программировать на C++ вместе с Джоном Смайли. – СПб: ООО «ДиаСофтЮП», 2003.-560с.

11. Шилдт Г. Искусство программирования на C++, - СПБ.: БХВ – Петербург, 2005, - 496 с.

12. Романов Б.А. Практикум по программированию на C++: Учебное пособие. СПб.: BXB-Петербург, Новосибирск: Из-во НГТУ, 2006.- 432с.

13. Павел Храмцов "Поиск и навигация в Internet". http://www.osp.ru/cw/1996/20/31.htm

14. How Intranet Search Tools and Spiders Work http://linux.manas.

ПРИЛОЖЕНИЕ

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "UnMenu.h"
#include "UnVid.h"
#include "UnInch.h"
#include "UnProd.h"
#include "UnBluda.h"
#include "UnBMenu.h"
#include "UnSpisok.h"
#include "UnProg.h"
#include "UnDM.h"
//---------------------------------------------------------------------------
#pragma package (smart_init)
#pragma resource "*.dfm"
TFmMenu *FmMenu;
//AnsiString st;
//---------------------------------------------------------------------------
__fastcall TFmMenu::TFmMenu(TComponent* Owner)
        : TForm(Owner)
{
}
```

```
//---------------------------------------------------------------------------
void __fastcall TFmMenu::ActExitExecute(TObject *Sender)
{
  Close();
}
//---------------------------------------------------------------------------
void __fastcall TFmMenu::ActVidExecute(TObject *Sender)
{
  FmVid->Show();
}
//---------------------------------------------------------------------------
void __fastcall TFmMenu::ActCatExecute(TObject *Sender)
{
  FmCat->Show();
}
//---------------------------------------------------------------------------
void __fastcall TFmMenu::ActInchExecute(TObject *Sender)
{
  FmInch->Show();
}
//---------------------------------------------------------------------------
void __fastcall TFmMenu::ActProdExecute(TObject *Sender)
{
   FmProd->Show();
```

```
}
//---------------------------------------------------------------------------
void __fastcall TFmMenu::ActMenuExecute(TObject *Sender)
{
  FmBMenu->Show();
}
//---------------------------------------------------------------------------
void __fastcall TFmMenu::ActProgExecute(TObject *Sender)
{
  FmProg->Show();
}
//---------------------------------------------------------------------------
void __fastcall TFmMenu::Action1Execute(TObject *Sender)
{
  ShowMessage("Программа выполнена...");
}
//---------------------------------------------------------------------------
void __fastcall TFmMenu::Action2Execute(TObject *Sender)
{
  FmBluda->ShowModal();
}
//---------------------------------------------------------------------------
void __fastcall TFmMenu::FormActivate(TObject *Sender)
{
```

```
 DM->TbVid->Active = false;
 DM->TbBludo->Active = false;
 DM->TbRasch->Active = false;
 DM->TbProd->Active = false;
DM->TbProd1->Active = false;
DM->TbZak->Active = false;
DM->TBInch->Active = false;
DM->TbDMenu->Active = false;
DM->TbMenu->Active = false;
DM->TbSpisok->Active = false;
DM->TbList->Active = false;
DM->TbBludo1->Active = false;
DM->QRaschod->Active = false;

DM->TbProd->DatabaseName = GetCurrentDir()+"\\base\\";
DM->TbProd1->DatabaseName = GetCurrentDir()+"\\base\\";
DM->TbBludo->DatabaseName = GetCurrentDir()+"\\base\\";
DM->TbVid->DatabaseName = GetCurrentDir()+"\\base\\";
DM->TbRasch->DatabaseName = GetCurrentDir()+"\\base\\";
DM->QRaschod->DatabaseName = GetCurrentDir()+"\\base\\";
DM->TbBludo1->DatabaseName = GetCurrentDir()+"\\base\\";
DM->TbSpisok->DatabaseName = GetCurrentDir()+"\\base\\";
DM->TbList->DatabaseName = GetCurrentDir()+"\\base\\";
DM->TbZak->DatabaseName = GetCurrentDir()+"\\base\\";
```

```
  DM->TBInch->DatabaseName = GetCurrentDir()+"\\base\\";
  DM->TbDMenu->DatabaseName = GetCurrentDir()+"\\base\\";
  DM->TbMenu->DatabaseName = GetCurrentDir()+"\\base\\";
  //
  //
  DM->TBInch->Active = true;
  DM->TbVid->Active = true;
  //ShowMessage(DM->TbProd->DatabaseName);
  DM->TbBludo->Active = true;
  DM->TbBludo1->Active = true;

  DM->TbZak->Active = true;
  DM->TbRasch->Active = true;
  DM->TbProd->Active = true;
  DM->TbProd1->Active = true;
  DM->TbMenu->Active = true;
  DM->TbDMenu->Active = true;
  DM->TbList->Active = true;
  DM->TbSpisok->Active = true;
  DM->QRaschod->Active = true;
}
//---------------------------------------------------------------------------
void __fastcall TFmMenu::FormCreate(TObject *Sender)
{
```

```
  //st = GetCurrentDir()+ "\\base\\";
  //ShowMessage(st);
}
//---------------------------------------------------------------------------
void __fastcall TFmMenu::ToolButton8Click(TObject *Sender)
{
  Action2->Execute();
}
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
//---------------------------------------------------------------------------
//---------------------------------------------------------------------------
WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)
{
      try
      {
            Application->Initialize();
            Application->CreateForm(__classid(TForm12), &Form12);   //создание
форм
            Application->CreateForm(__classid(TForm1), &Form1);
            Application->CreateForm(__classid(TForm2), &Form2);
            Application->CreateForm(__classid(TForm3), &Form3);
            Application->CreateForm(__classid(TForm4), &Form4);
            Application->CreateForm(__classid(TForm5), &Form5);
            Application->CreateForm(__classid(TForm6), &Form6);
            Application->CreateForm(__classid(TForm7), &Form7);
            Application->CreateForm(__classid(TForm8), &Form8);
```

```
        Application->CreateForm(__classid(TForm9), &Form9);
        Application->CreateForm(__classid(TForm10), &Form10);
        Application->CreateForm(__classid(TForm11), &Form11);
        Application->CreateForm(__classid(TForm13), &Form13);
        Application->CreateForm(__classid(TForm14), &Form14);
        Application->CreateForm(__classid(TForm15), &Form15);
        Application->CreateForm(__classid(TForm16), &Form16);
        Application->CreateForm(__classid(TForm17), &Form17);
        Application->CreateForm(__classid(TForm18), &Form18);
        Application->CreateForm(__classid(TForm19), &Form19);
        Application->Run();
    }
    catch (Exception &exception)
    {
        Application->ShowException(&exception);
    }
    catch (...)
    {
        try
        {
            throw Exception("");
        }
        catch (Exception &exception)
        {
            Application->ShowException(&exception);
        }
    }
    return 0;
}
//---------------------------------------------------------------------------
```

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "parol.h"
#include "glavnaya_forma.h"
#include "glavnaya_forma2.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm12 *Form12;
//---------------------------------------------------------------------------
__fastcall TForm12::TForm12(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm12::BitBtn2Click(TObject *Sender)
AnsiString login=Edit2->Text;
AnsiString bl;
AnsiString password=Edit3->Text;
AnsiString bp;
if (Edit2->Text==""&&Edit3->Text!="")
        {
          Label7->Visible=true;
          Label7->Caption="Введите Логин";
          return;
        }
if (Edit2->Text==""&&Edit3->Text=="")
        {
           Label7->Visible=true;
           Label7->Caption="Введите Логин и Пароль";
```

```
        return;
      }
if (Edit2->Text!=""&&Edit3->Text=="")
      {
        Label7->Visible=true;
        Label7->Caption="Введите Пароль";
        return;
      }
if (Edit2->Text!=""&&Edit3->Text!="")
{
ADOTable1->First();
for(int i = 0;i < ADOTable1->RecordCount; i++)
{
bl=ADOTable1->FieldByName("User")->AsString;
bp=ADOTable1->FieldByName("Pass")->AsString;
if(login == bl && password == bp)
{
Form12->Hide();
 Form1->Show();
}
else
ADOTable1->Next();
}
}
}
//---------------------------------------------------------------------------

void __fastcall TForm12::Label4Click(TObject *Sender)
{
 Form12->Hide();
```

```
 Form13->Show();
}
//---------------------------------------------------------------------------

void __fastcall TForm12::Label3MouseEnter(TObject *Sender)
{
Label3->Font->Color = clBlack;
Label3->Font->Size = 16;
}
//---------------------------------------------------------------------------

//---------------------------------------------------------------------------

void __fastcall TForm12::Label3MouseLeave(TObject *Sender)
{
Label3->Font->Color = clBlack;
Label3->Font->Size = 14;
}
//---------------------------------------------------------------------------

void __fastcall TForm12::Label4MouseLeave(TObject *Sender)
{
Label4->Font->Color = clBlack;
Label4->Font->Size = 14;

}
//---------------------------------------------------------------------------

void __fastcall TForm12::Label4MouseEnter(TObject *Sender)
{
```

```
Label4->Font->Color = clBlack;
Label4->Font->Size = 16;
}
//---------------------------------------------------------------------------

void __fastcall TForm12::FormCreate(TObject *Sender)
{
Form12->Visible = true;
Edit2->Clear();
Edit3->Clear();
Label6->Visible = false;
BitBtn1->Visible = false;
BitBtn2->Visible = false;
Edit2->Visible = false;
Edit3->Visible = false;
Label5->Visible = false;
Label7->Visible = false;
}
//---------------------------------------------------------------------------

void __fastcall TForm12::Label3Click(TObject *Sender)
{
Label3->Visible = false;
Label4->Visible = false;
Label5->Visible = true;
Edit2->Visible = true;
Edit3->Visible = true;
Label3->Visible = false;
Label4->Visible = false;
Label7->Visible = false;
```

```
Label6->Visible = true;
BitBtn1->Visible = true;
BitBtn2->Visible = true;
}
//---------------------------------------------------------------------------

void __fastcall TForm12::BitBtn1Click(TObject *Sender)
{
Label3->Visible = true;
Label4->Visible = true;
Form12->Visible = true;
Edit2->Clear();
Edit3->Clear();
Label6->Visible = false;
BitBtn1->Visible = false;
BitBtn2->Visible = false;
Edit2->Visible = false;
Edit3->Visible = false;
Label5->Visible = false;
Label7->Visible = false;
}
//---------------------------------------------------------------------------

//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "glavnaya_forma.h" #include "avtori.h"
#include "chitateli.h"
```

```
#include "nazvaniya.h"
#include "raspisanie.h"
#include "razdeli.h"
#include "vidachaknig.h"
#include "sotrudniki.h"
#include "raspsotr.h"
#include "filtr_avtori.h"
#include "filtr_nazvaniya.h"
#include "parol.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------


void __fastcall TForm1::N2Click(TObject *Sender) {
Form2->Show();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::N6Click(TObject *Sender) {
Form3->Show();
}
//---------------------------------------------------------------------------
```

```
void __fastcall TForm1::N3Click(TObject *Sender)
{
Form4->Show();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::N11Click(TObject *Sender)
{
Form5->Show();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::N4Click(TObject *Sender)
{
Form6->Show();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::N7Click(TObject *Sender)
{
Form7->Show();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::N9Click(TObject *Sender)
{
Form8->Show();
}
//---------------------------------------------------------------------------
```

```
void __fastcall TForm1::N10Click(TObject *Sender)
{
Form9->Show();
}
//---------------------------------------------------------------------------


void __fastcall TForm1::N12Click(TObject *Sender) {
Application->Terminate();
}
//---------------------------------------------------------------------------


void __fastcall TForm1::FormCreate(TObject *Sender)
{

}
//---------------------------------------------------------------------------

//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "glavnaya_forma2.h"
#
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
```

```
TForm13 *Form13;
//---------------------------------------------------------------------------
__fastcall TForm13::TForm13(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm13::N2Click(TObject *Sender)
{
Form14->Show();
}
//---------------------------------------------------------------------------
void __fastcall TForm13::N4Click(TObject *Sender)
{
Form16->Show();
}
//---------------------------------------------------------------------------
void __fastcall TForm13::N11Click(TObject *Sender)
{
Form17->Show();
}
//---------------------------------------------------------------------------
void __fastcall TForm13::N3Click(TObject *Sender)
{
Form18->Show();
}
//---------------------------------------------------------------------------
void __fastcall TForm13::N12Click(TObject *Sender)
{
Application->Terminate();
```

```
}
//---------------------------------------------------------------------------

void __fastcall TForm13::FormCreate(TObject *Sender)
{

}
//---------------------------------------------------------------------------

//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "avtori2.h"
#include "filtr_avtori2.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm14 *Form14;
//---------------------------------------------------------------------------
__fastcall TForm14::TForm14(TComponent* Owner)
        : TForm(Owner)
{
String filename = "biblioteka.mdb"; // код подключения библиотеки из папки в которой она находится
  if(!FileExists(ExtractFilePath(Application->ExeName)+filename))
{
   AnsiString ds = "Файл базы данных не обнаружен \n\n\t" +filename;
   ShowMessage( ds );
```

```
}
else
 {
  String WayToBase=ExtractFilePath(Application->ExeName)+filename;

  ADOConnection1->ConnectionString =
"Provider=Microsoft.Jet.OLEDB.4.0;User ID=Admin;Data
Source="+WayToBase+";Mode=Share Deny None;Jet OLEDB:System
database="";Jet OLEDB:Registry Path="";Jet OLEDB:Database Password="";Jet
OLEDB:Engine Type=5;Jet OLEDB:Database Locking Mode=1;Jet
OLEDB:Global Partial Bulk Ops=2;Jet OLEDB:Global Bulk Transactions=1;Jet
OLEDB:New Database Password="";Jet OLEDB:Create System
Database=False;Jet OLEDB:Encrypt Database=False;Jet OLEDB:Don't Copy
Locale on Compact=False;Jet OLEDB:Compact Without Replica Repair=False;Jet
OLEDB:SFP=False;";

  ADOConnection1->Connected = true;
  ADOTable1->Active = true;
  ADOTable1->Open();
  }
}
//---------------------------------------------------------------------------

void __fastcall TForm14::BitBtn1Click(TObject *Sender) // код для кнопки
«Поиск»
{
if( !Form15->Visible )
 Form15->Show();
}
//---------------------------------------------------------------------------
```

```
void __fastcall TForm14::FormCreate(TObject *Sender)
{

}
//---------------------------------------------------------------------------



#include <vcl.h>
#pragma hdrstop

#include "filtr_avtori.h"
#include "avtori.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm10 *Form10;
//---------------------------------------------------------------------------
__fastcall TForm10::TForm10(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm10::BitBtn1Click(TObject *Sender) {
Edit1->Clear();
}
//---------------------------------------------------------------------------
void __fastcall TForm10::BitBtn7Click(TObject *Sender) {
Edit2->Clear();
}
```

```
//---------------------------------------------------------------------------
void __fastcall TForm10::BitBtn2Click(TObject *Sender) {
Edit2->Text = Edit2->Text + " AND ";
}
//---------------------------------------------------------------------------
void __fastcall TForm10::BitBtn3Click(TObject *Sender) {
 Edit2->Text = Edit2->Text + " OR ";
}
//---------------------------------------------------------------------------
void __fastcall TForm10::BitBtn4Click(TObject *Sender)
{
Edit2->Text = Edit2->Text + " < ";
}
//---------------------------------------------------------------------------
void __fastcall TForm10::BitBtn5Click(TObject *Sender)
{
Edit2->Text = Edit2->Text + " > ";
}
//---------------------------------------------------------------------------
void __fastcall TForm10::BitBtn6Click(TObject *Sender)
{
Form2->ADOTable1->Filter = Edit2->Text;
Form2->ADOTable1->Filtered = true;
}

//---------------------------------------------------------------------------
//---------------------------------------------------------------------------
void __fastcall TForm10::FormClose(TObject *Sender, TCloseAction &Action)
{
Form2->ADOTable1->Filtered = false;
```

```
}

void __fastcall TForm10::FormCreate(TObject *Sender)
{

}
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "sotrudniki.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm8 *Form8;
//---------------------------------------------------------------------------
__fastcall TForm8::TForm8(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------

void __fastcall TForm8::FormCreate(TObject *Sender{
String filename = "biblioteka.mdb";
  if(!FileExists(ExtractFilePath(Application->ExeName)+filename))
{
   AnsiString ds = "Файл базы данных не обнаружен \n\n\t" +filename;
   ShowMessage( ds );
}
```

```
else
 {
  String WayToBase=ExtractFilePath(Application->ExeName)+filename;

  ADOConnection1->ConnectionString =
"Provider=Microsoft.Jet.OLEDB.4.0;User ID=Admin;Data
Source="+WayToBase+";Mode=Share Deny None;Jet OLEDB:System
database="";Jet OLEDB:Registry Path="";Jet OLEDB:Database Password="";Jet
OLEDB:Engine Type=5;Jet OLEDB:Database Locking Mode=1;Jet
OLEDB:Global Partial Bulk Ops=2;Jet OLEDB:Global Bulk Transactions=1;Jet
OLEDB:New Database Password="";Jet OLEDB:Create System
Database=False;Jet OLEDB:Encrypt Database=False;Jet OLEDB:Don't Copy
Locale on Compact=False;Jet OLEDB:Compact Without Replica Repair=False;Jet
OLEDB:SFP=False;";

  ADOConnection1->Connected = true;
  ADOTable1->Active = true;
  ADOTable1->Open();
  }
}
//---------------------------------------------------------------------------
```