Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Институт естественных и точных наук
Факультет математики, механики и компьютерных технологий
Кафедра прикладной математики и программирования
Направление подготовки Прикладная математика и информатика

РАБОТА ПРОВЕРЕНА
Рецензент, д.т.н., старший
научный сотрудник
_____Б.М.Суховилов
«____»_____2017г.

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой, д.ф.-м.н.,
доцент
_____А.А.Замышляева
«____»_____2017 г.

Разработка программных средств для управления предприятием общественного питания

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУрГУ–01.03.02.2017.29.ПЗ ВКР

Руководитель работы, доцент
кафедры ПМиП
_____/ М.Ю. Катаргин
«____»_____2017 г.
Автор работы
Студентка группы ЕТ-482
_____/ А.С. Кондакова
«____»_____2017 г.
Нормоконтролер, доцент
_____/ Т.Ю. Оленчикова
«____»_____2017 г.

Челябинск 2017

АННОТАЦИЯ

Кондакова А.С. Разработка программных средств для управления предприятием общественного питания.– Челябинск: ЮУрГУ, ЕТ–482, 47 с., 33 ил., 10 табл., библиогр. список – 11 наим., 1 прил.

В работе рассмотрен процесс работы фирмы «Аквилон». Рассмотрены аналоги, выявлены существенные отличия. На основании этого было принято решения разработать свою базу данных. Для реализации приложения были выбраны Borland C++Builder и Microsoft SQL Server 2008 R2. Построена схема базы данных и алгоритмы работы приложения.

Результатом работы стал готовый программный продукт, который предназначен для управления кафе, принадлежащего фирме «Аквилон». Реализованная программа позволяет добавлять и редактировать продукты, блюда, состав блюд, комплексные обеды, состав комплексных обедов, типы экскурсий, а также единицы измерения. Реализован поиск продуктов и блюд в таблицах.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Каждый из нас хотя бы раз был на экскурсии. ООО «Аквилон» занимается организацией досуга в виде катания на собачьих упряжках, тематических экскурсий и живого общения с сибирскими хаски и самоедскими лайками. Экскурсии бронируются заранее. Экскурсии бывают длительными, и в процессе проведения экскурсии экскурсантам предлагается комплексный обед. Целью данной работы является разработка программы для управления кафе, принадлежащего фирме «Аквилон».

В первом разделе был проведен обзор существующих решений, были сформулированы требования к программе.

Во втором разделе разработана модель ER-диаграмма, модель базы данных, алгоритм работы приложения и функции, необходимые для должного функционирования программы.

В третьем разделе были выявлены требования к интерфейсу программы и, в соответствии с ними, был реализован понятный и удобный интерфейс. Разработаны и описаны модули приложения, которые позволяют добавлять и редактировать продукты, блюда, состав блюд, комплексные обеды, состав комплексных обедов, типы экскурсий, а также единицы измерения. Также реализован поиск продуктов и блюд в таблицах. Проведено испытание программы на тестовых данных.

1 АНАЛИЗ ТРЕБОВАНИЙ. ОБЗОР СУЩЕСТВУЮЩИХ РЕШЕНИЙ

1.1 Постановка задачи и анализ требований

Данная работа выполняется для предприятия «Аквилон».

Необходимо разработать программу для ввода и редактирования данных о приходе и расходе продуктов, о блюдах и их составе, об экскурсиях, их составе и заказанных комплексных обедах и для учёта расхода запасов продуктов.

Пользователями системы являются сотрудники хаски-центра, которые не имеют навыков программирования. Поэтому программа должна иметь интуитивно понятный интерфейс пользователя, не требовать сложных настроек при установке, быть надежной и простой в эксплуатации.

В программе план закупок продуктов должен записываться в файл в формате Excel.

Для решения поставленной задачи необходимо:

1) разработать схему базы данных;
2) разработать интерфейс пользователя;
3) реализовать и протестировать программу.

1.2 Введение в процесс работы предприятия «Аквилон»

«Аквилон» занимается организацией досуга в виде катания на собачьих упряжках, тематических экскурсий и живого общения с сибирскими хаски и самоедскими лайками. Экскурсии предполагаются как кратковременные, так и длительные. Кратковременные экскурсии проходят без перерыва. Продолжительность кратковременной экскурсии составляет от получаса до двух часов. Продолжительная экскурсия длится максимум 7 часов.

После окончания кратковременной экскурсии посетители могут по желанию покушать в столовой. Во время длительных экскурсий предполагается перерыв на обед. При бронировании билета на экскурсию есть возможность заказать один или несколько из предлагаемых комплексных обедов не позже чем за неделю до назначенной даты экскурсии. Если заказ был осуществлен при бронировании, то посетители просто подходят к раздаче блюд и забирают выбранный ранее комплексный обед. При желании посетитель может взять дополнительные блюда к обеду. Если комплексный обед не заказывался, то они могут купить в столовой любой набор блюд из меню дня непосредственно на свой вкус. Если предполагается продолжение экскурсии, то по окончании трапезы посетители возвращаются к экскурсии.

Для экскурсии заранее планируется дата, время начала и время окончания, а также заранее известно, сколько человек будет на экскурсии. Экскурсия проводится для группы посетителей. Количество участников экскурсии может варьироваться от 10–15 человек до 30–35 человек. Экскурсия может быть заказана организацией, например, школой, а также индивидуально. В случае индивидуальных частных заказов посетители формируются в группы произвольно

или присоединяются к уже сформированной группе. Также можно заказать урезанную по формату экскурсию даже на двух человек.

В заявке указывается, сколько человек в группе планируется на экскурсию, сколько комплексных обедов нужно этой группе лиц, фамилия, имя, отчество и номер телефона контактного лица группы. Для каждого посетителя предусмотрена возможность получить комплексный обед, поэтому при формировании заявки на экскурсию можно указывать состав желаемого комплексного обеда.

Для экскурсии предусматривается количество обедов, которые требуются для ее обслуживания. Для каждого из блюд меню предполагается использование некоторых продуктов.

В комплексный обед входят: суп, второе блюдо и гарнир на выбор, хлеб, чай, выпечка к чаю.

Каждое блюдо готовится из определенных продуктов, также определены нормы расхода продуктов, и нормы взаимозаменяемости продуктов в соответствии с требованиями санитарных правил и норм. Если для приготовления какого-то блюда какого-то продукта нет в наличии, то согласно таблице взаимозаменяемости продуктов можно изменить состав блюда. Если продукта, которым по таблице должен заменяться отсутствующий продукт нет в таблице, так же нет в наличии, то блюдо, в состав которого входят эти продукты снимается с реализации до тех пор, пока продукты не появятся на складе.

В основном ассортименте фиксированный набор блюд, но каждый день в зависимости от того, есть ли продукты, необходимые для приготовления блюда на складе, меню может изменяться, например, может не быть какой-то позиции.

Если какой-то продукт по какой-то причине закупается по разной цене, то на цену конечного блюда это никак не влияет, так как эта разница уже вложена в наценку. Продукты закупаются согласно составленному графику либо через поставщиков, либо лично сотрудниками хаски-центра, отвечающими за пополнение запасов.

Прием продуктов осуществляется администратором, который отмечает поступление. Расход продуктов рассчитывается программой, согласно количеству проданных блюд и комплексных обедов.

1.3 Обзор существующих решений

1.3.1 БЭСТ-5.Питание

БЭСТ-5.Питание – программа, предоставляемая одной из существующих российских компаний разработчиков программного обеспечения для бизнеса**.**

В программном комплексе «БЭСТ-5. Питание» учтены требования и особенности учета питания на предприятиях различного профиля.

Для детских учреждений (детских садов, школ, детских домов и интернатов) меню ведется на каждый день и по приемам пищи, осуществляется контроль

рациона питания по энергетической ценности и химическому составу, натуральным нормам.

Для санаторно-курортных и лечебных предприятий расчет меню может вестись по диетам. Программа поставляется с заполненным справочником стандартных диет, который может быть самостоятельно дополнен или изменен. Возможно ведение системы питания по типу «шведский стол». Поддерживается ведение нескольких видов типового (примерного) меню для каждой категории питающихся и, если необходимо, для каждой диеты.

Для кафе и общественных столовых меню ведется по заказам, предусмотрен режим реализации блюд с наценкой, расчет меню массового мероприятия (банкет, торжество, корпоративный вечер и др.). Возможно оформление прихода готовой продукции штуками или весом.

Для предприятий оказывающих услуги по организации питания (комбинаты школьного и социального питания) предусмотрен специальный режим, обеспечивающий составление и расчет меню по приемам и категориям питающихся с учетом наценки и по каждому заказчику в отдельности, формируется акт оказанных услуг.

Для детского, санаторного и лечебного питания предусмотрены режимы контроля рационов питания по энергетической ценности, химическому составу, включая витамины и микроэлементы, и стоимости [11].

1.3.2 Poster POS

Poster POS – облачная касса для кафе и магазинов, которая может осуществлять работу на операционных системах iPad и Android. Разработчики обещают надежную работу приложения без интернета. Админ-панель Poster находится на защищенном сервере в интернете и доступна только администратору по логину и паролю. Для просмотра продаж и управления заведением достаточно подключиться к интернету и сделать это можно из любой точки мира. Рабочее место кассира или официанта работает на iPad, планшетах на Android, на любом ноутбуке, компьютере или моноблоке. Чеки печатаются на термальном принтере Epson TM-T20 [10].

1.3.3 Forecast NOW!

Forecast NOW! – программа для закупщиков, которая оптимизирует складские запасы; автоматизирует расчёт заказов торговых компаний.

После разовой настройки программа в фоновом режиме отслеживает и анализирует историю продаж для склада, оптимизирует и формирует заказы на основе прогнозирования спроса, предоставляет мощную аналитическую отчетность для принятия стратегических решений. Также существует вариант формирования заказов в автоматическом режиме без участия пользователя. Можно оставить программу рассчитывать заказы на ночь – утром все сделанные заказы будут доступны [8].

1.3.4 Oremax

Oremax – комплексное программное обеспечение для организации управленческого и оперативного учета. Имеется два механизма управления запасами: плановые закупки на склад с возможностью настройки алгоритма планирования и поставки под заказ. Обеспечивает прозрачный контроль над закупками. Реализован учёт товаров на удалённых складах. Главным ограничением является то, что систему можно использовать только в оптовой торговле, использование её в розничных фирмах – не предусмотрено, а входящий в состав продукта модуль "Производство" не может быть введён у клиентов с большими объёмами производства [9].

В отличие от всех уже существующих решений, я предлагаю оптимальную программу именно для предприятия «Аквилон». Здесь отсутствуют ненужные пользователю функции, и, наоборот, есть только используемые в ходе работы функции, появилась возможность бронировать комплексные обеды заранее, при заказе экскурсии.

В программе можно осуществлять следующие действия:

- ввод и редактирование данных о приходе и расходе продуктов, о блюдах и их составе, об экскурсиях, их составе и заказанных комплексных обедах;
- планирование и управление запасом продуктов.

1.4 Обоснование выбора платформы разработки

Для разработки базы данных был выбран Microsoft SQL Server 2008 R2. Для разработки клиентского приложения был выбран Borland C++Builder.

Microsoft SQL Server – система управления реляционными базами данных (РСУБД), разработанная корпорацией Microsoft. Основной используемый язык запросов – Transact-SQL, создан совместно Microsoft и Sybase. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия.

SQL Server – это база данных, которая уже шесть лет признается наименее уязвимой по результатам тестов на уязвимость в Национальном институте стандартов и технологий США.

Важное преимущество Microsoft SQL Server Developer Edition – бесплатный доступ для разработки и тестирования.

Borland C++ Builder – выпущенное компанией Borland средство быстрой разработки приложений, позволяющее создавать приложения на языке C++, используя при этом среду разработки и библиотеку компонентов Delphi.

Формы являются основой приложений C++ Builder. Создание пользовательского интерфейса приложения заключается в добавлении в окно формы элементов объектов C++ Builder, называемых компонентами. 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.

Преимуществом Borland C++ Builder является интуитивная понятность и простота использования.

1.5 Выводы по разделу

В настоящее время в «Аквилон» нельзя попасть на экскурсию без предварительной записи, для этого необходимо ее забронировать. Бронирование происходит по телефону, максимум за неделю до даты проведения эксукрсии. Для удобства сотрудников хаски-центра было принято решение разработать программу, позволяющую бронировать экскурсии и комплексные обеды для посетителей. Предполагается, что в основе программы будет лежать база данных, которую предстоит разработать.

Был проведен обзор существующих решений, в ходе которого было выявлено, что ни одна из существующих программ не удовлетворяет требованиям, заявленным фирмой, так как в этих программах нет необходимых функций для фирмы «Аквилон».

В данном разделе также были сформулированы требования к программе, план закупок продуктов должен записываться в файл в формате Excel.

2 ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ

2.1 ER-диаграмма

Выделено пять сильных сущностей: «Экскурсии», «Посетители», «Комплексный обед», «Блюда», «Продукты», «Продавцы» (рисунок 2.1).

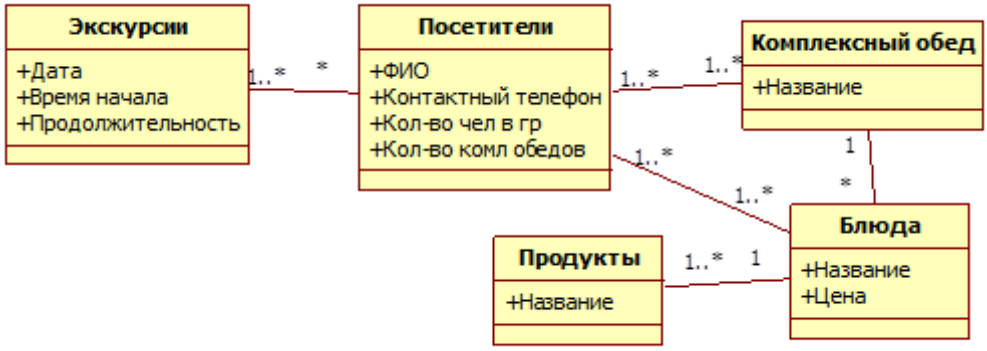


Рисунок 2.1 – ER-диаграмма

2.2 Перевод ER-модели в реляционную форму

Была разработана следубщая база данных (рисунок 2.2).

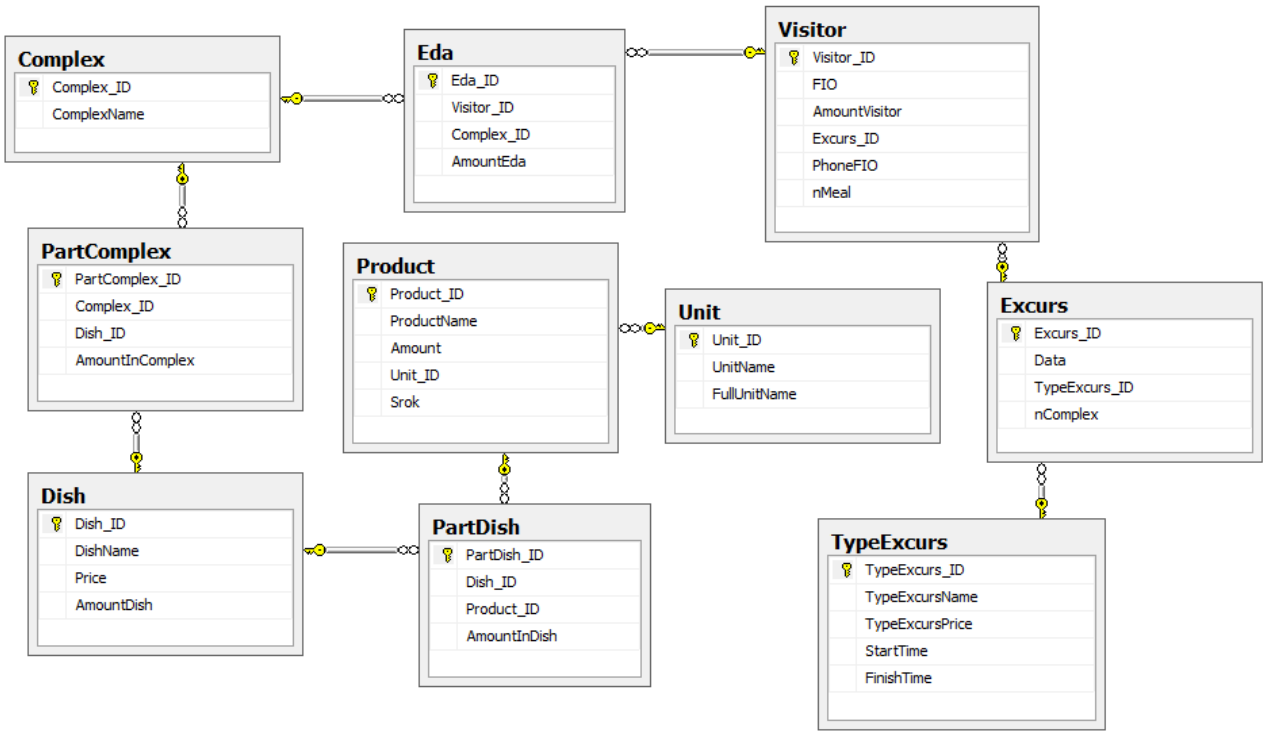


Рисунок 2.2 – Схема базы данных

Существуют различные типы экскурсий, для каждой из которых задано уникальное имя, цена, время начала и окончания. Данные хранятся в таблице *TypeExcurs*.

Происходит планирование экскурсий на конкретную дату, экскурсии могут проводиться более одного раза в день. Данные о датах проведения экскурсий хранятся в таблице *Excurs.* В состав участников экскурсий могут входить как отдельные лица, так и группы. В каждом случае в базе данных фиксируются данные лица подавшего заявку на участие в экскурсии. Эти данные хранит таблица *Visitor.* Таблица содержит поля: ФИО, № телефона, число заявленных участников.

Посетители при бронировании мест на экскурсию могут заказать комплексные обеды, данные об этом хранятся в таблице *Eda*, там указано сколько комплексных обедов каждого типа заказано посетителями экскурсии.

В таблице *Complex* хранятся различные типы комплексных обедов, а в таблице *PartComplex* перечислен состав комплексных обедов, а именно, блюдо и его количество в обеде.

Все возможные блюда перечислены в таблице *Dish*, заданы название блюда, цена, и имеющееся в наличии количество.

Состав блюда хранится в таблице *PartDish*, где указано какое количество продуктов требуется для конкретного блюда.

Наименование продукта, имеющееся на складе количество и соответствующая единица измерения описаны в таблице *Product*. В таблице *Unit* перечислены все единицы измерения (штука, килограмм и т.д.)

Для полного ознакомления с используемой базой данных рассмотрим все её таблицы и их атрибуты, описание представлено в таблицах с 1 по 10.

В таблице 2.1 содержится информация о блюдах.

Таблица 2.1

Dish (Блюда)

Имя Атрибута	Тип	Обязательность	Значение по умолчанию	Первичный ключ	Внешний ключ	Описание
Dish_ID	int	+	–	+	–	Идентификатор блюда
DishName	varchar(50)	+	–	–	–	Название блюда
Price	money	+	–	–	–	Цена блюда
AmountDish	int	+	–	–	–	Количество готовых блюд

Данная таблица состоит из четырёх полей: идентификатор блюда, название блюда, цена блюда и количество готовых блюд. Таблица необходима для хранения информации обо всех возможных блюдах.

В таблице 2.2 содержится информация о продуктах.

Таблица 2.2

Product (Продукты)

Имя Атрибута	Тип	Обязательность	Значение по умолчанию	Первичный ключ	Внешний ключ	Описание
Product_ID	int	+	–	+	–	Идентификатор продукта
ProductName	varchar(50)	+	–	–	–	Название продукта
Amount	float	+	–	–	–	Количество продукта на складе
Unit_ID	int	+	–	–	+	Идентификатор единицы измерения
Srok	smallint	+	–	–	–	Срок хранения продукта

Данная таблица состоит из пяти полей: идентификатор продукта, название продукта, количество продукта на складе, идентификатор единицы измерения и срок хранения продукта. Поле Srok является вычисляемым. Таблица необходима для хранения информации обо всех возможных продуктах, которые используются в приготовлении блюд.

В таблице 2.3 содержится информация о составах блюд.

Данная таблица состоит из четырёх полей: идентификатор состава блюда, идентификатор блюда, идентификатор продукта и количество продукта в блюде. Таблица необходима для хранения информации о составах блюд.

В таблице 2.4 содержится информация о единицах измерения.

Данная таблица состоит из трёх полей: идентификатор единицы измерения, сокращенное название единицы измерения и полное название единицы измерения. Таблица необходима для хранения информации об единицах измерения.

В таблице 2.5 содержится информация о комплексных обедах.

Таблица 2.3

PartDish (Состав блюда)

Имя Атрибута	Тип	Обязательность	Значение по умолчанию	Первичный ключ	Внешний ключ	Описание
PartDish_ID	int	+	–	+	–	Идентификатор состава блюда
Dish_ID	int	+	–	–	+	Идентификатор блюда
Product_ID	int	+	–	–	+	Идентификатор продукта
AmountInDish	float	+	–	–	–	Количество продукта в блюде

Таблица 2.4

Unit (Единицы измерения)

Имя Атрибута	Тип	Обязательность	Значение по умолчанию	Первичный ключ	Внешний ключ	Описание
Unit_ID	int	+	–	+	–	Идентификатор единицы измерения
UnitName	varchar(5)	+	–	–	–	Сокращенное название единицы измерения
FullUnitName	varchar(20)	+	–	–	–	Полное название единицы измерения

Таблица 2.5

Complex (Единицы измерения)

Имя Атрибута	Тип	Обязательность	Значение по умолчанию	Первичный ключ	Внешний ключ	Описание
Complex_ID	int	+	–	+	–	Идентификатор комплексного обеда
ComplexName	varchar(50)	+	–	–	–	Название комплексного обеда

Данная таблица состоит из двух полей: идентификатор комплексного обеда и название комплексного обеда. Таблица необходима для хранения информации о комплексных обедах.

В таблице 2.6 содержится информация о составе комплексных обедов.

Таблица 2.6

PartComplex (Состав комплексного обеда)

Имя Атрибута	Тип	Обязательность	Значение по умолчанию	Первичный ключ	Внешний ключ	Описание
PartComplex_ID	int	+	–	+	–	Идентификатор состава комплексного обеда
Complex_ID	int	+	–	–	+	Идентификатор комплексного обеда
Dish_ID	int	+	–	–	+	Идентификатор блюда
AmountInComplex	int	+	–	–	–	Количество блюд каждого типа в комплексном обеде

Данная таблица состоит из четырёх полей: идентификатор состава комплексного обеда, идентификатор комплексного обеда, идентификатор блюда и количество блюд каждого типа в комплексном обеде. Таблица необходима для хранения информации о составе комплексных обедов.

В таблице 2.7 содержится информация о типах экскурсий.

Таблица 2.7

TypeExcurs (Типы экскурсий)

Имя Атрибута	Тип	Обязательность	Значение по умолчанию	Первичный ключ	Внешний ключ	Описание
TypeExcurs_ID	int	+	–	+	–	Идентификатор типа экскурсии
TypeExcursName	varchar(100)	+	–	–	–	Название типа экскурсии
TypeExcursPrice	money	+	–	–	–	Цена типа экскурсии
StartTime	time(0)	+	–	–	–	Время начала экскурсии
FinishTime	time(0)	+	–	–	–	Время окончания экскурсии

Данная таблица состоит из пяти полей: идентификатор типа экскурсии, название типа экскурсии, цена типа экскурсии, время начала экскурсии и время окончания экскурсии. Таблица необходима для хранения информации о типах экскурсий.

В таблице 2.8 содержится информация о запланированных экскурсиях. Данная таблица состоит из четырёх полей: идентификатор экскурсии, дата экскурсии, идентификатор типа экскурсии и количество комплексных обедов на экскурсию. Поле nComplex является вычисляемым. Таблица необходима для хранения информации о запланированных экскурсиях.

В таблице 2.9 содержится информация о запланированных экскурсиях.

Данная таблица состоит из шести полей: идентификатор посетителя, фамилия, имя, отчество посетителя, количество посетителей в группе, идентификатор комплексных обедов, телефон посетителя и количество комплексных обедов на группу. Поле nMeal является вычисляемым. Таблица необходима для хранения информации обо всех посетителях экскурсии.

Таблица 2.8

Excurs (Экскурсии)

Имя Атрибута	Тип	Обязательность	Значение по умолчанию	Первичный ключ	Внешний ключ	Описание
Excurs_ID	int	+	–	+	–	Идентификатор экскурсии
Data	datetime	+	–	–	–	Дата экскурсии
TypeExcurs_ID	int	+	–	–	+	Идентификатор типа экскурсии
nComplex		–	–	–	–	Количество комплексных обедов на экскурсию

Таблица 2.9

Visitor (Посетители)

Имя Атрибута	Тип	Обязательность	Значение по умолчанию	Первичный ключ	Внешний ключ	Описание
Visitor_ID	int	+	–	+	–	Идентификатор посетителя
FIO	varchar(30)	+	–	–	–	ФИО посетителя
AmountVisitor	int	+	–	–	–	Количество посетителей в группе
Excurs_ID	int	+	–	–	+	Идентификатор комплексных обедов
PhoneFIO	varchar(11)	+	–	–	–	Телефон посетителя
nMeal		-	–	–	–	Количество обедов

В таблице 2.10 содержится информация о запланированных экскурсиях.

Таблица 2.10

Eda (Состав заказа посетителей на обеды)

Имя Атрибута	Тип	Обязательность	Значение по умолчанию	Первичный ключ	Внешний ключ	Описание
Eda_ID	int	+	–	+	–	Идентификатор состава заказа
Visitor_ID	int	+	-	–	+	Идентификатор посетителя
Complex_ID	int	+	–	–	+	Идентификатор комплексного обеда
AmountEda	int	-	–	–	–	Количество комплексных обедов конкретного типа

Данная таблица состоит из четырёх полей: идентификатор состава заказа, идентификатор посетителя, идентификатор комплексного обеда и количество комплексных обедов конкретного типа. Таблица необходима для хранения информации о составе заказа посетителей на комплексные обеды.

Отношения находятся в 1НФ, так как нет повторяющихся групп данных.

Отношения находятся во 2НФ, так как все атрибуты полностью функционально зависят от первичных ключей.

Отношения находятся в 3НФ, так как нет транзитивных зависимостей атрибутов.

2.3 Реализованные функции и процедуры

Необходимо было разработать функцию, которая бы рассчитывала какие продукты необходимо закупить на период от текущей даты до выбранной даты.

Была разработана функция PlanProd, возвращающая таблицу. Ниже приведен ее листинг.

```
USE [Sklad]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
```

```
CREATE FUNCTION [dbo].[PlanProd](@d datetime)
RETURNS @r TABLE(
   id int identity(1,1) not null,
   Product_ID int,
   ProductName varchar(50),
   WeHave float, -- имеется на настоящий момент
   ToBuy float, -- требуется закупить
   UnitName varchar(10)
)
AS begin
insert into @r(Product_ID,ProductName,WeHave,ToBuy,UnitName)
select Product_ID,ProductName,Amount,dbo.HowMuchProductToBuy(Product_ID,
@d),UnitName
from Product p, Unit u
where p.Unit_ID=u.Unit_ID
RETURN
End
```

Чтобы узнать, сколько нужно закупить продукта, чтобы хватило до выбранной даты была реализована функция HowMuchProductToBuy Ниже приведен ее листинг.

```
USE [Sklad]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE FUNCTION [dbo].[HowMuchProductToBuy](@Product_ID int, @d
datetime)
RETURNS float
AS
BEGIN
declare @srok smallint, @nDay smallint, @WeHave float, @result float
set @nDay=CAST(@d-GETDATE() as int)
--срок годности продукта и имеющееся количество
select @srok=srok,@WeHave=Amount from Product where
Product_ID=@Product_ID
--берем меньшее из @srok и @nDay
if @srok<@nDay set @nDay=@srok
set @result=@nDay*dbo.ProductOutgo(@Product_ID)-@WeHave
if @result<0 set @result=0
return @result
END
```

Количество продукта, которое необходимо закупить рассчитывается по формуле:

@result=@nDay*dbo.ProductOutgo(@Product_ID)-@WeHave, (1)

где @nDay – число дней, на которые идет расчет;
@WeHave – имеющееся в наличии количество продукта;
ProductOutgo(@Product_ID) – функция, которая вычисляет среднесуточный расход продукта за последние 30 дней. Ниже приведен ее листинг.

```
USE [Sklad]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE FUNCTION [dbo].[ProductOutgo](@Product_ID int)
RETURNS float
AS
BEGIN
declare @nDay int, @n int
set @nDay=30
declare @d datetime, @Rashod float
select @d=MIN(Data) from Excurs
set @n=cast(GETDATE()-@d as int)
if(@n<30)set @nDay=@n
set @Rashod=isnull((select
SUM(ed.AmountEda*pc.AmountInComplex*pd.AmountInDish)
from Excurs e, Visitor v,Eda ed,Complex c, PartComplex pc, Dish d, PartDish pd
where e.Excurs_ID=v.Excurs_ID
  and v.Visitor_ID=ed.Visitor_ID
  and c.Complex_ID=ed.Complex_ID
  and pc.Complex_ID=c.Complex_ID
  and d.Dish_ID=pd.Dish_ID
  and pd.Product_ID=@Product_ID
  and e.Data>=GETDATE()-@nDay),0)
return @Rashod/@nDay
END
```

Функция insProduct используется для добавления нового продукта. Параметрами функции являются название продукта, его количество на складе, идентификатор единицы измерения и срок хранения продукта.

```
USE [Sklad]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
```

```
CREATE PROCEDURE [dbo].[insProduct] @ProductName varchar(50), @Amount
float, @Unit_ID int,
  @Srok smallint, @id int output
AS
BEGIN
   SET NOCOUNT ON;
   insert into Product(ProductName, Amount, Unit_ID, Srok)
   values(@ProductName, @Amount, @Unit_ID, @Srok)set @id=scope_identity( )
END
```

Функция insDish используется для добавления нового блюда. Параметрами функции являются название блюда, его цена и количество блюд в наличии.

```
USE [Sklad]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[insDish] @DishName varchar(50), @Price money,
@AmountDish int, @id int output
AS
BEGIN
SET NOCOUNT ON;
insert into Dish(DishName, Price, AmountDish)
values(@DishName, @Price, @AmountDish)
set @id=scope_identity( )
END
```

Функция insPartDish используется для добавления продукта в состав блюда. Параметрами функции являются идентификатор продукта, входящего в состав блюда, количество продукта в блюде и идентификатор самого блюда.

```
USE [Sklad]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[insPartDish] @Product_ID int, @AmountInDish
float, @Dish_ID int, @id int output
AS
BEGIN
   SET NOCOUNT ON;
   insert into PartDish(Dish_ID, Product_ID, AmountInDish)
   values(@Dish_ID, @Product_ID, @AmountInDish)
  set @id=scope_identity( )
END
```

Функция ins Unit используется для добавления новой единицы измерения. Параметрами функции являются сокращенное название единицы измерения и полное название единицы измерения.

```
USE [Sklad]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[insUnit] @UnitName varchar(5), @FullUnitName
varchar(20), @id int output
AS
BEGIN
    SET NOCOUNT ON;
    insert into Unit(UnitName, FullUnitName)
    values(@UnitName, @FullUnitName)
   set @id=scope_identity( )
END
```

Функция insComplex используется для добавления нового комплексного обеда. Параметром функции является название комплексного обеда.

```
USE [Sklad]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[insComplex] @ComplexName varchar(50), @id int
output
AS
BEGIN
    SET NOCOUNT ON;
    insert into Complex(ComplexName)
    values(@ComplexName)
   set @id=scope_identity( )
END
```

Функция insPartComplex используется для добавления блюда в состав комплексного обеда. Параметрами функции являются идентификатор комплексного обеда, в который добавляется блюдо, идентификатор блюда, количество блюда в комплексном обеде.

```
USE [Sklad]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
```

```
GO
CREATE PROCEDURE [dbo].[insPartComplex] @Complex_ID int, @Dish_ID int,
@AmountInComplex int, @id int output
AS
BEGIN
    SET NOCOUNT ON;
    insert into PartComplex(Complex_ID,Dish_ID, AmountInComplex)
    values(@Complex_ID, @Dish_ID, @AmountInComplex)
   set @id=scope_identity( )
END
```

Функция insTypeExcurs используется для добавления новых типов экскурсий. Параметрами функции являются название типа экскурсии, цена за экскурсию, время начала и время окончания экскурсии.

```
USE [Sklad]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[insTypeExcurs] @TypeExcursName varchar(100),
@TypeExcursPrice money, @StartTime time(0),
@FinishTime time(0), @id int output
AS
BEGIN
    SET NOCOUNT ON;
    insert into TypeExcurs(TypeExcursName, TypeExcursPrice, StartTime,
FinishTime)
    values(@TypeExcursName, @TypeExcursPrice, @StartTime, @FinishTime)
   set @id=scope_identity( )
END
```

Функция insExcurs используется для добавления экскурсии на конкретную дату. Параметрами функции являются дата, на которую запланирована экскурсия и идентификатор типа экскурсии.

```
USE [Sklad]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[insExcurs] @Data date, @TypeExcurs_ID int, @id
int output
AS
BEGIN
    SET NOCOUNT ON;
```

```
    insert into Excurs(Data, TypeExcurs_ID)
    values(@Data, @TypeExcurs_ID)
    set @id=scope_identity( )
END
```

Функция insVisitor используется для добавления посетителей в запланированную экскурсию. Параметрами функции являются фамилия, имя, отчество контактного лица, количество посетителей, идентификатор экскурсии и контактный телефон.

```
USE [Sklad]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[insVisitor] @FIO varchar(30), @AmountVisitor
int, @Excurs_ID int, @PhoneFIO varchar(11), @id int output
AS
BEGIN
    SET NOCOUNT ON;
    insert into Visitor(FIO, AmountVisitor, Excurs_ID, PhoneFIO)
    values(@FIO, @AmountVisitor, @Excurs_ID, @PhoneFIO)
    set @id=scope_identity( )
END
```

Функция insEda используется для добавления информации о заказанных конкретным посетителем экскурсии комплексных обедах. Параметрами функции являются идентификатор посетителя экскурсии, идентификатор комплексного обеда и количество заказанных комплексных обедов определенного типа.

```
USE [Sklad]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[insEda] @Visitor_ID int, @Complex_ID int,
@AmountEda int, @id int output
AS
BEGIN
    SET NOCOUNT ON;
    insert into Eda(Visitor_ID, Complex_ID, AmountEda)
    values(@Visitor_ID, @Complex_ID, @AmountEda)
    set @id=scope_identity( )
END
```

Функция nVisitor используется для вычисления количества посетителей конкретной экскурсии. Входным параметром функции является идентификатор

экскурсии, для которой вычисляется количество посетителей. Выходным параметром функции является количество посетителей экскурсии.

```
USE [Sklad]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE FUNCTION [dbo].[nVisitor](@Excurs_ID int)
RETURNS int AS
BEGIN
declare @Amount int
select @Amount=SUM(coalesce(AmountVisitor,0))
from Visitor
where Excurs_ID=@Excurs_ID
return @Amount
END
```

Функция nComplex используется для вычисления количества комплексных обедов, заказанных посетителем экскурсии. Входным параметром функции является идентификатор посетителя, для которого вычисляется количество заказанных им комплексных обедов. Выходным параметром функции является количество заказанных посетителем комплексных обедов.

```
USE [Sklad]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE FUNCTION [dbo].[nComplex](@Visitor_ID int)
RETURNS int AS
BEGIN
declare @Amount int
select @Amount=SUM(coalesce(AmountEda,0))
from Eda
where Visitor_ID=@Visitor_ID
return @Amount
END
```

Функция nEdaForVisitor используется для вычисления количества комплексных обедов, заказанных посетителем экскурсии. Входным параметром функции является идентификатор посетителя, для которого вычисляется количество заказанных им комплексных обедов. Выходным параметром функции является количество заказанных посетителем комплексных обедов.

```
USE [Sklad]
GO
```

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE FUNCTION [dbo].[nEdaForVisitor](@Visitor_ID int)
RETURNS smallint
AS BEGIN
return isnull((select sum(AmountEda) from Eda where Visitor_ID=@Visitor_ID), 0)
END
```

2.4 Разработка алгоритмов программы

2.4.1 Основной алгоритм

После запуска приложения происходит инициализация приложения, создаются все окно приложения, после чего выполняется цикл обработки сообщений. На каждом шаге цикла из системной очереди выбирается сообщение для активного окна и вызывается соответствующий обработчик. Цикл прекращает работу, если поступает сообщение об уничтожении окна. Схема данного алгоритма приведена на рисунке 2.3.

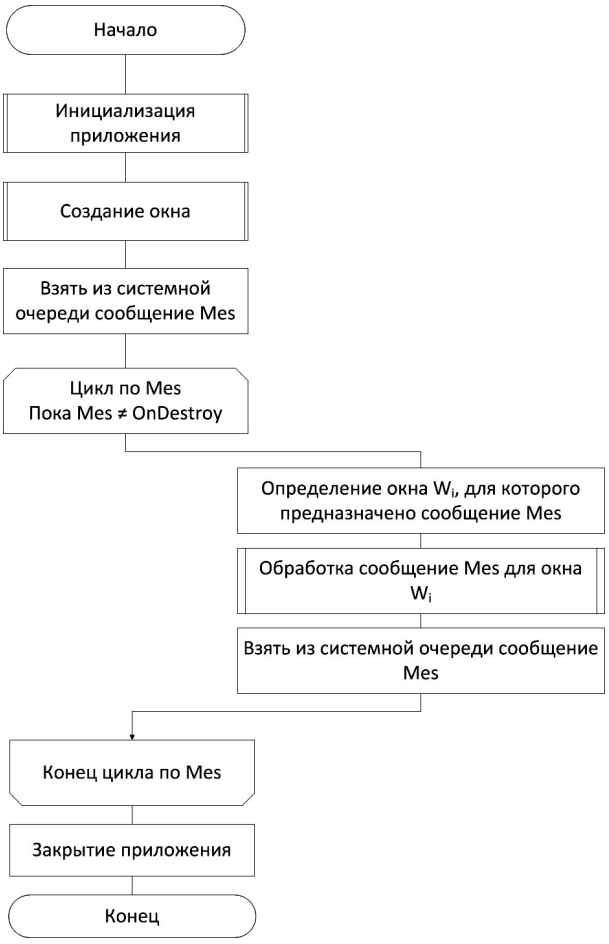


Рисунок 2.3 – Основной алгоритм программы

2.4.2 Алгоритм выбора пункта меню

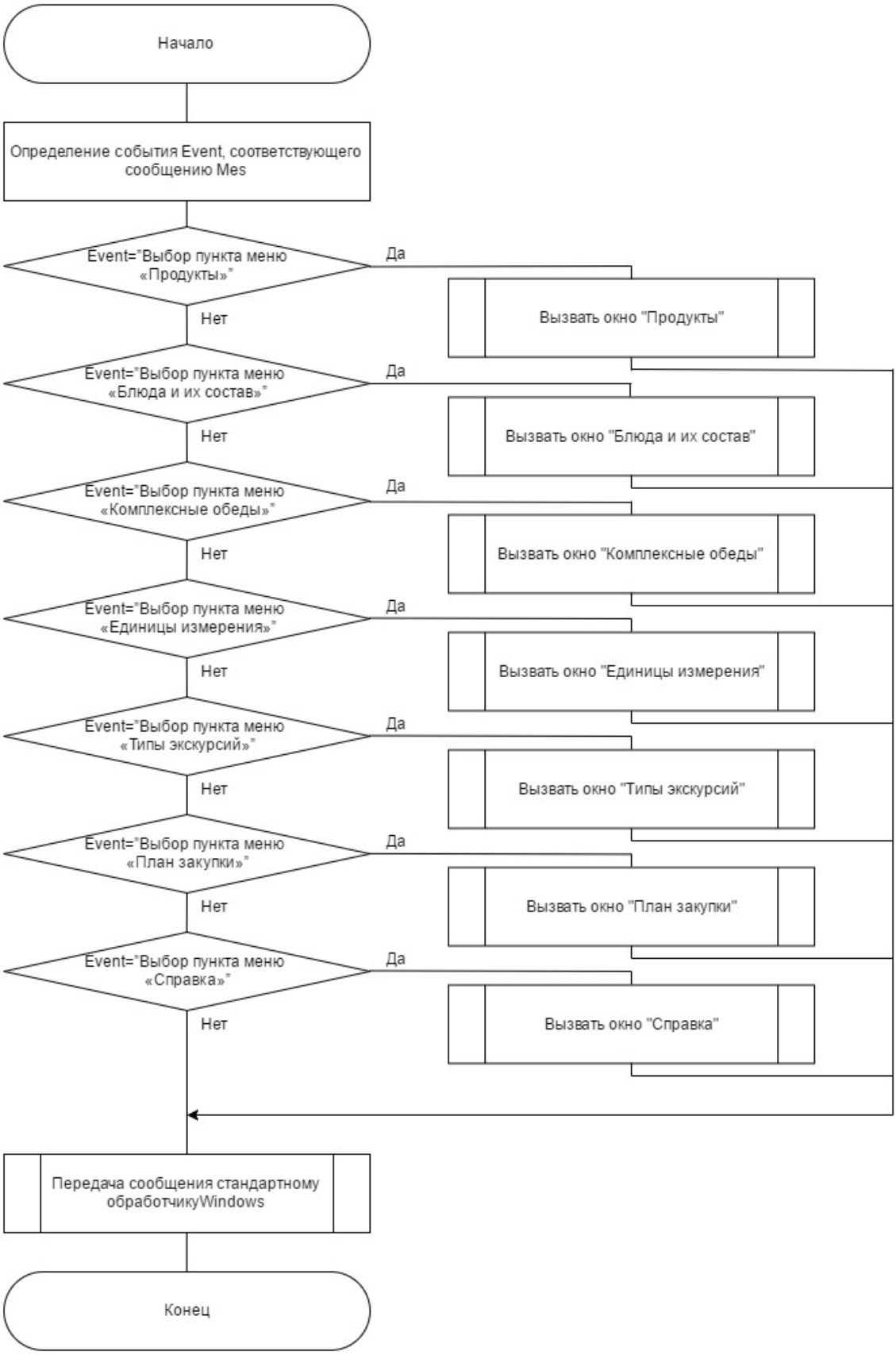


Рисунок 2.4. – Алгоритм выбора пункта меню

При выборе пункта «Продукты», происходит открытие справочного окна со списком имеющихся продуктов.

При выборе пункта «Блюда и их состав», происходит открытие справочного окна со списком имеющихся блюд и их составов.

При выборе пункта «Комплексные обеды», происходит открытие справочного окна со списком комплексных обедов и их составов.

При выборе пункта «Единицы измерения», происходит открытие справочного окна со списком единиц измерения.

При выборе пункта «Типы экскурсий», происходит открытие справочного окна со списком типов экскурсий.

При выборе пункта «План закупки», происходит открытие окна с единиц измерения.

При выборе пункта «Справка», происходит открытие окна, в котором содержится информация о разработчике программы.

Схема данного алгоритма приведена на рис. 2.4.

2.5 Выводы по разделу

В данном разделе разработана модель ER-диаграмма, в ней выделено пять сильных сущностей: «Экскурсии», «Посетители», «Комплексный обед», «Блюда», «Продукты», «Продавцы». Также разработана модель базы данных, в которую входит 10 таблиц, а именно:

- TypeExcurs – таблица, в которой хранятся различные типы экскурсий;
- Excurs – таблица, в которой хранятся данные о датах проведения экскурсий;
- Visitor – таблица, в которой хранится список посетителей экскурсий;
- Eda – таблица, в которой хранятся данные о заказанных комплексных обедах посетителей экскурсий;
- Complex – таблица, в которой хранятся различные типы комплексных обедов;
- PartComplex – таблица, в которой хранятся составы всех комплексных обедов;
- Dish – таблица, в которой хранятся данные о различных блюдах;
- PartDish – таблица, в которой хранятся составы всех блюд;
- Product – таблица, в которой хранятся данные о различных продуктах;
- Unit – таблица, в которой хранятся данные о различных единицах измерения.

Разработан алгоритм работы приложения и функции, необходимые для должного функционирования программы.

3 РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ

3.1 Разработка интерфейса

Стоит рассказать о стандарте интерфейса. Вся информация из базы отображается в таблицах. При нажатии ПКМ появляется контекстное меню, которое позволяет добавлять/изменять/удалять записи в таблицах. Также реализованы горячие клавиши Ins/Enter/Del соответственно для добавления/изменения/удаления записей в таблицах. Добавление и редактирование происходит в отдельных формах. Для удобства пользователя переход между полями редактирования может происходить при нажатии клавиши табуляции. Предусмотрена блокировка ненужных клавиш в полях редактирования. Например, в полях, относящихся к количеству нельзя вводить буквы. Также для большего удобства пользователя реализован поиск по таблицам продуктов и блюд. При вводе текста в поле поиска происходит позиционирование в таблице. Так же реализованы табуляция, позволяющая производить быстрый переход с одного поля редактирования на другое; горячие клавиши, например, для создания новой записи.

3.2 Описание программы

Данная программа написана под операционную систему Windows, на языке программирования C. Для её функционирования должно быть установлено следующее программное обеспечение:

1) Microsoft SQL Server 2008 R2 ;
2) Microsoft Office Excel 2003 и выше

3.3 Описание модулей

1) MainUnit.cpp – главный модуль программы. В нем отображается список запланированных экскурсий, список посетителей экскурсии и список комплексных обедов, заказанных посетителями.
2) DbUnit.cpp – модуль, предназначенный для подключения к базе данных.
3) Product.cpp – модуль, предназначенный для отображения списка продуктов и поиска необходимого продукта.
4) ProductsWnd.cpp – модуль, предназначенный для добавления нового и редактирования уже существующего продукта.
5) ProdDishUnit.cpp – модуль, предназначенный для отображения списка блюд и их состава, а также для поиска необходимого блюда.
6) DishWnd.cpp – модуль, предназначенный для добавления нового и редактирования уже существующего блюда.
7) PartDishWnd.cpp – модуль, предназначенный для добавления и редактирования продукта в составе блюда.

8) Complex.cpp – модуль, предназначенный для отображения списка комплексных обедов и их состава.

9) ComplexWnd.cpp – модуль, предназначенный для добавления и редактирования комплексных обедов.

10) PartComplexWnd.cpp – модуль, предназначенный для добавления и редактирования блюда в составе комплексного обеда.

11) TypeExcurs.cpp – модуль, предназначенный для отображения списка типов экскурсий.

12) TypeExcursWnd.cpp – модуль, предназначенный для добавления и редактирования типов экскурсий.

13) Units.cpp – модуль, предназначенный для отображения списка единиц измерения.

14) UnitsWnd.cpp – модуль, предназначенный для добавления и редактирования единиц измерения.

15) ExcursWnd.cpp – модуль, предназначенный для добавления и редактирования экскурсий на конкретную дату.

16) VisitorWnd.cpp – модуль, предназначенный для добавления и редактирования посетителей в экскурсии.

17) EdaWnd.cpp – модуль, предназначенный для добавления и редактирования комплексных обедов, заказанных посетителями экскурсии.

18) Report.cpp – модуль, предназначенный для формирования плана закупки продуктов на конкретную дату.

19) Spravka.cpp – модуль, предназначенный для отображения информации о разработчике программы.

3.4 Пример работы программы

На рисунке 3.1 показано главное окно программы.

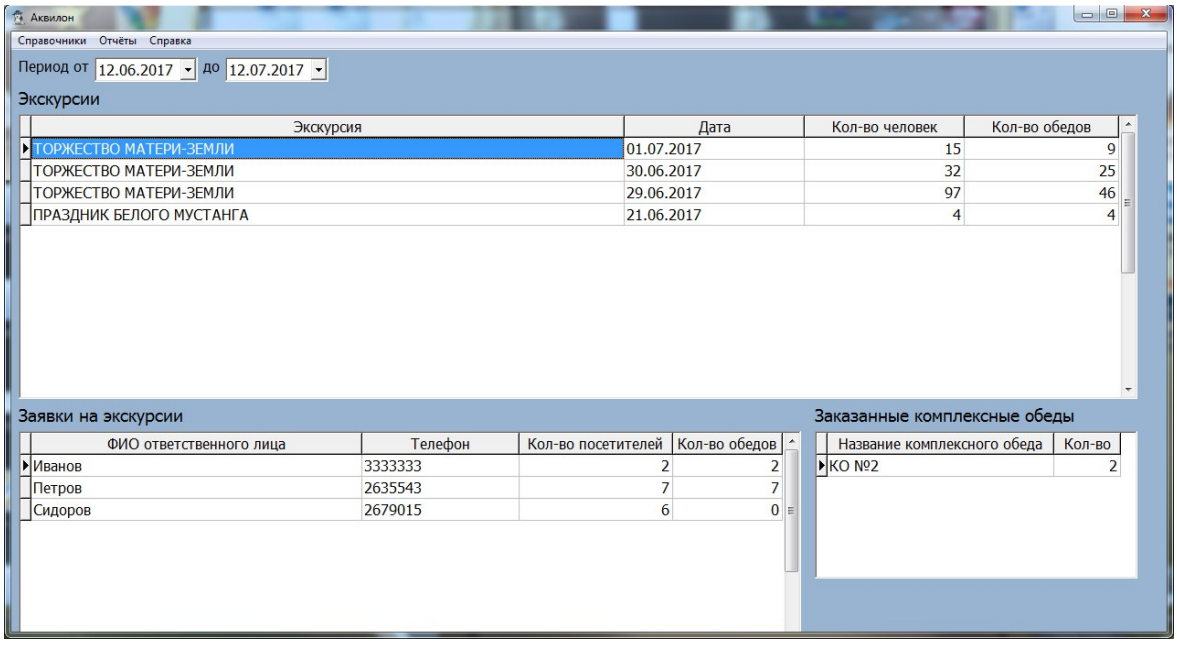


Рисунок 3.1 – Главное окно программы

На головной форме расположены три таблицы: таблица экскурсий, заявок на экскурсию и заказанных комплексных обедов.

В таблице экскурсий отображаются все экскурсии за выбранный пользователем период, их дата, количество человек в экскурсии и количество комплексных обедов, заказанных членами экскурсии.

При добавлении экскурсии появляется окно, в котором нужно выбрать название экскурсии и выбрать дату, на которую экскурсия будет запланирована. После добавления соответствующие изменения отображаются в редактируемой таблице (рисунок 3.2).

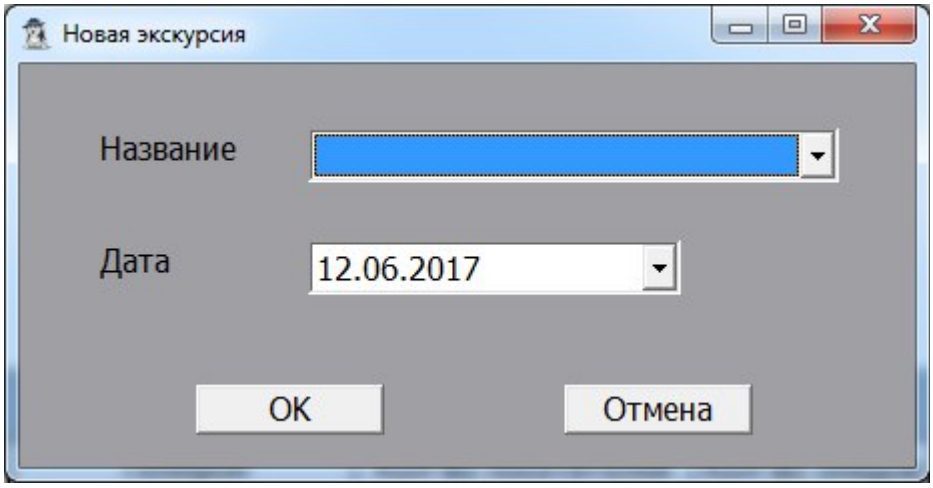


Рисунок 3.2 – Окно добавления новой экскурсии

При изменении данных об уже запланированной экскурсии появляется окно, в котором уже подставлены все данные и их можно поменять на свое усмотрение. После изменения соответствующие изменения отображаются в редактируемой таблице (рисунок 3.3).

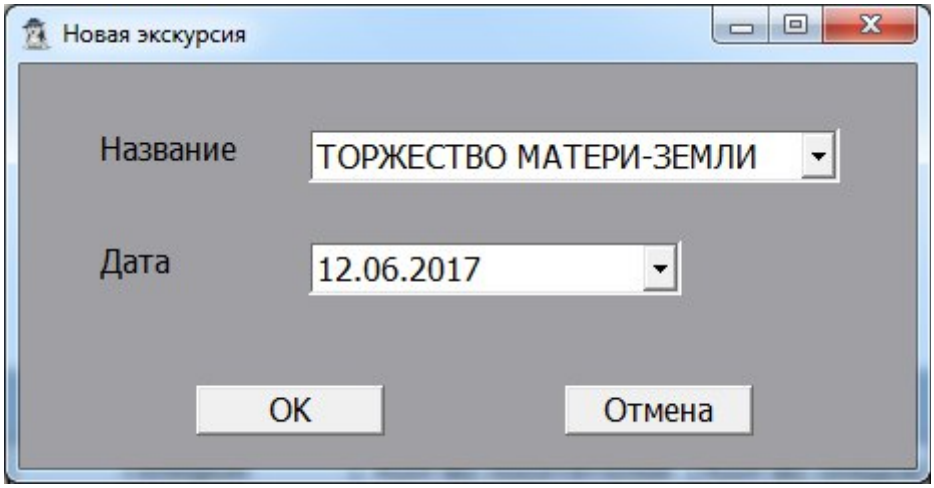


Рисунок 3.3 – Окно изменения существующей экскурсии

В таблице заявок на экскурсию отображается фамилия, имя, отчество ответственного лица группы, контактный телефон, количество посетителей в группе и количество комплексных обедов на группу. При добавлении посетителей и обедов соответствующие изменения отображаются в таблице экскурсий.

При добавлении заявки на экскурсию появляется окно, в котором нужно ввести фамилию, имя, отчество контактного лица, количество посетителей в группе и контактный телефон (рисунок 3.4).

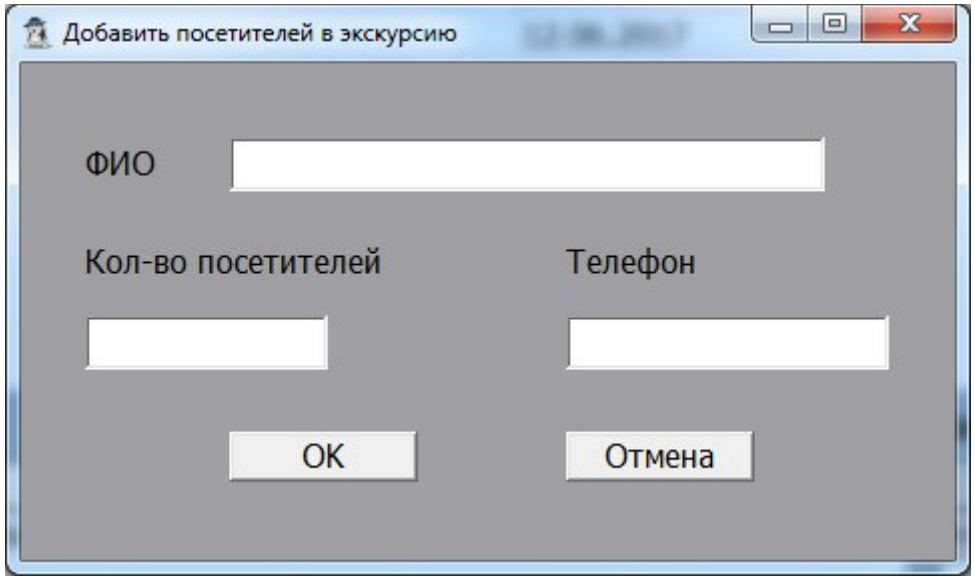


Рисунок 3.4 – Окно добавления посетителей в экскурсию

При изменении данных об уже оформленной заявке на экскурсию появляется окно, в котором уже подставлены все данные и их можно поменять на свое усмотрение. После изменения соответствующие изменения отображаются в таблице (рисунок 3.5).

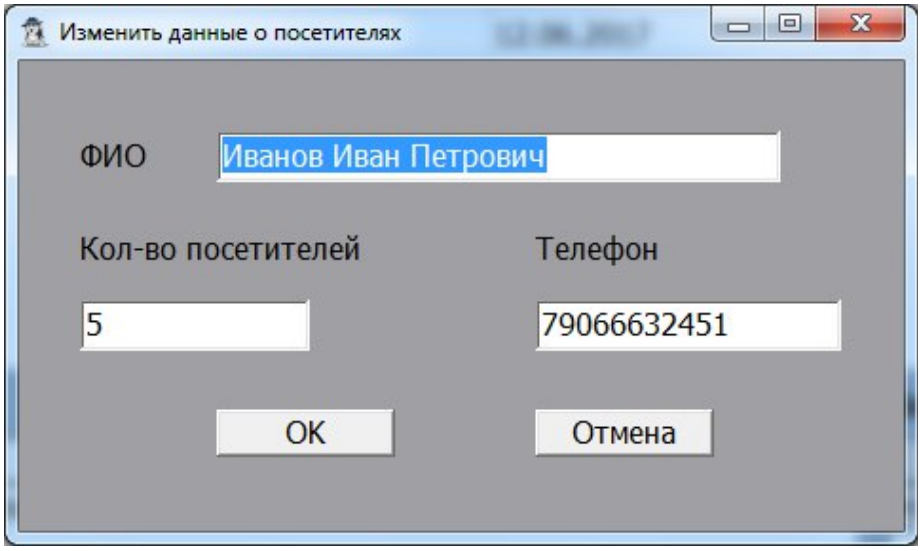


Рисунок 3.5 – Окно изменения данных о посетителе экскурсии

В таблице заказанных комплексных обедов отображается название комплексного обеда и его количество. Посетитель может выбрать несколько разных типов обеда в неограниченном количестве. При добавлении комплексного обеда соответствующие изменения отображаются в таблице заявок на экскурсию в поле, соответствующему посетителю, к которому относится комплексный обед.

При добавлении комплексного обеда для посетителя экскурсии появляется окно, в котором нужно выбрать желаемый обед и его количество (рисунок 3.6).

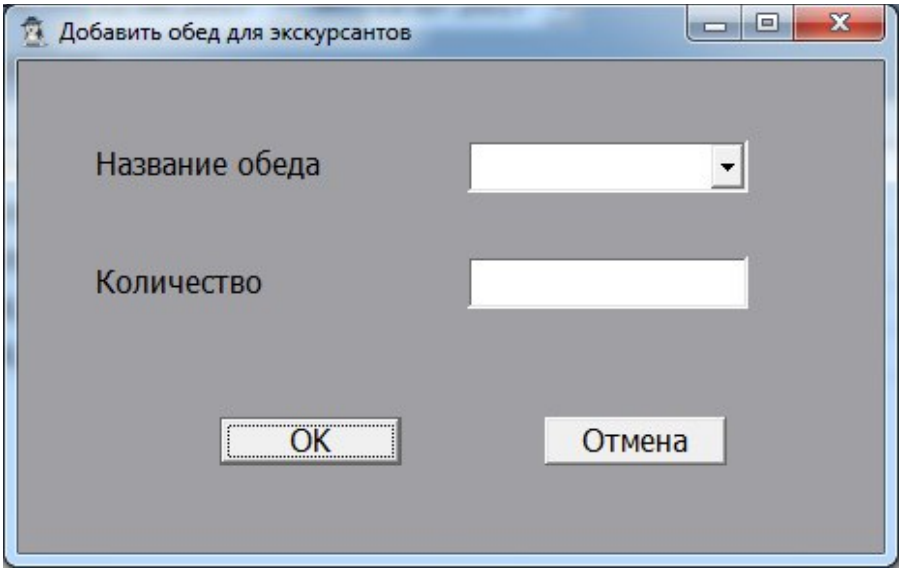


Рисунок 3.6 – Окно добавления комплексного обеда для посетителя экскурсии

При изменении данных об уже оформленной заявке на комплексный обед появляется окно, в котором уже подставлены все данные и их можно поменять на свое усмотрение (рисунок 3.7).

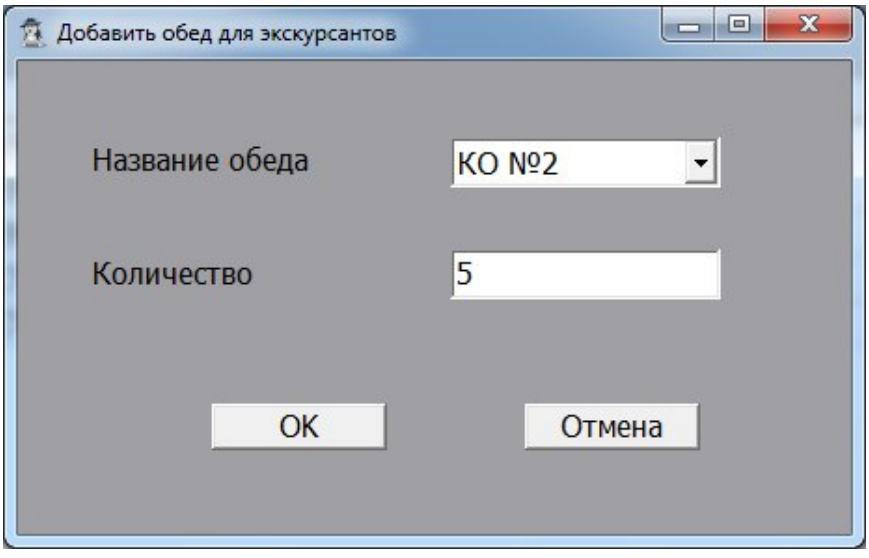
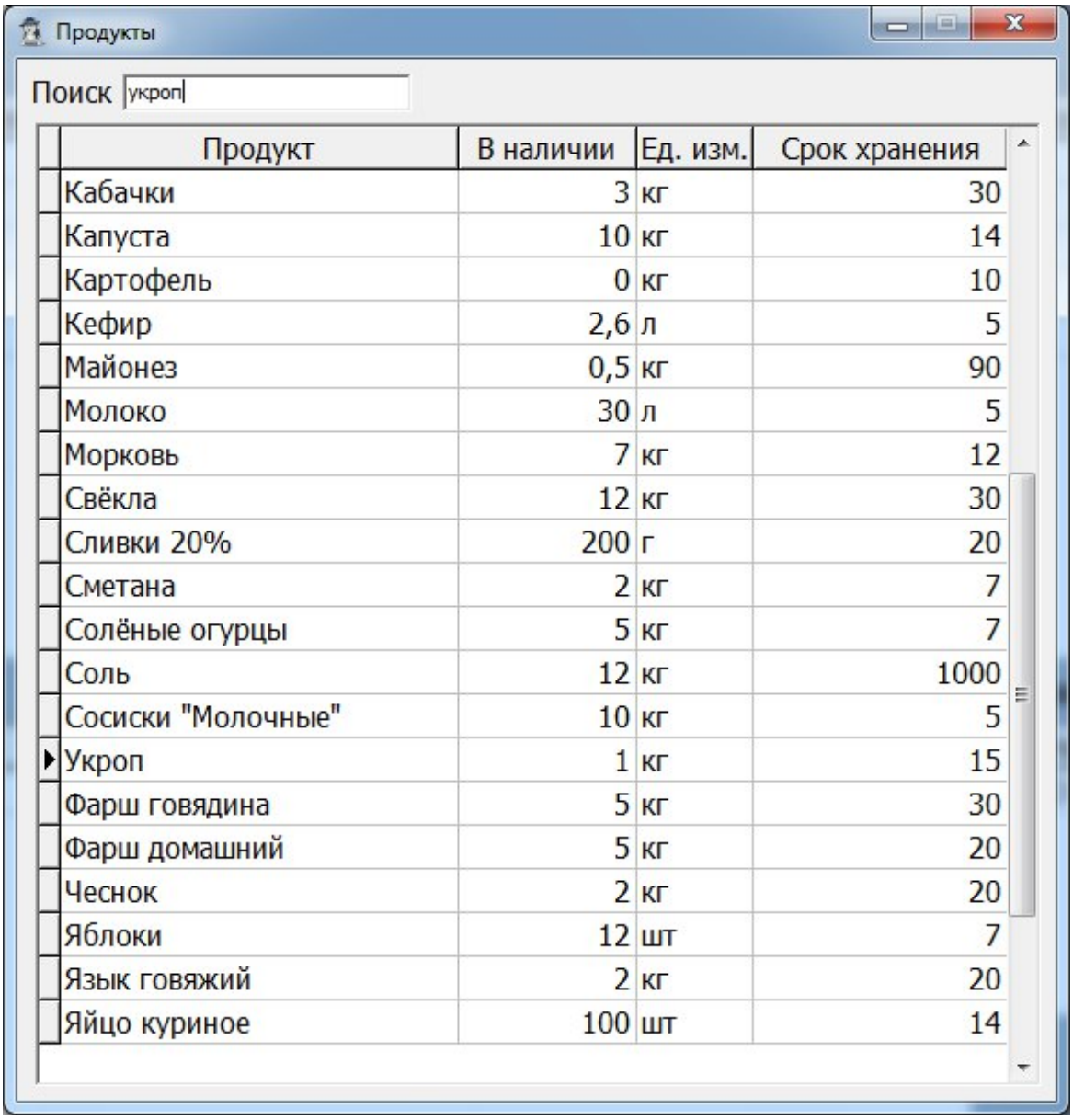


Рисунок 3.7 – Окно изменения комплексного обеда для посетителя экскурсии

При выборе пункта меню «справочники» и подпункта «продукты» появляется окно, в котором отображен перечень всевозможных продуктов, их количество на складе, единица измерения продукта и срок хранения. Для удобства пользователя имеется возможность поиска по продуктам, для этого нужно ввести название продукта в строку поиска и произойдет позиционирование, если таковой продукт есть в справочнике, что видно на рисунке. После добавления соответствующие изменения отображаются в таблице (рисунок 3.8).



Продукт	В наличии	Ед. изм.	Срок хранения
Кабачки	3	кг	30
Капуста	10	кг	14
Картофель	0	кг	10
Кефир	2,6	л	5
Майонез	0,5	кг	90
Молоко	30	л	5
Морковь	7	кг	12
Свёкла	12	кг	30
Сливки 20%	200	г	20
Сметана	2	кг	7
Солёные огурцы	5	кг	7
Соль	12	кг	1000
Сосиски "Молочные"	10	кг	5
Укроп	1	кг	15
Фарш говядина	5	кг	30
Фарш домашний	5	кг	20
Чеснок	2	кг	20
Яблоки	12	шт	7
Язык говяжий	2	кг	20
Яйцо куриное	100	шт	14

Рисунок 3.8 – Окно с перечнем продуктов

При добавлении нового продукта появляется окно, в котором нужно ввести название продукта, количество, срок хранения и выбрать соответствующую единицу измерения. После добавления соответствующие изменения отображаются в редактируемой таблице (рисунок 3.9).

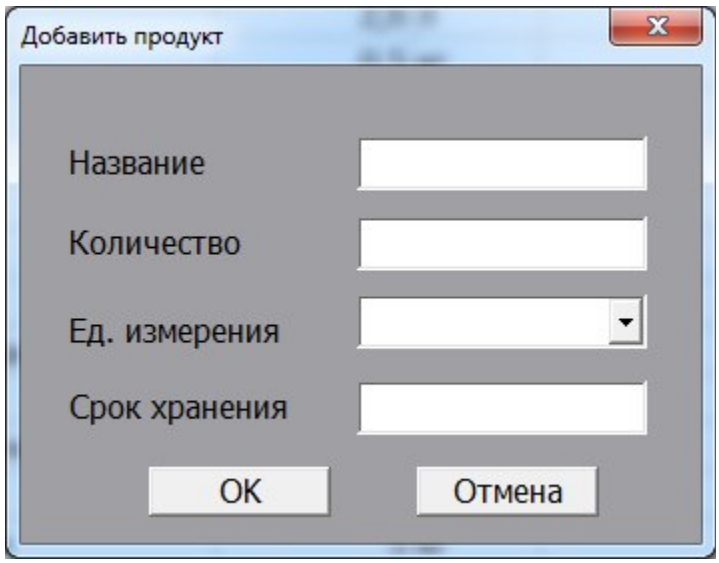


Рисунок 3.9 – Окно добавления продукта

При изменении данных об уже существующем продукте появляется окно, в котором уже подставлены все данные и их можно поменять по своему усмотрению. После изменения соответствующие данные отображаются в редактируемой таблице (рисунок 3.10).

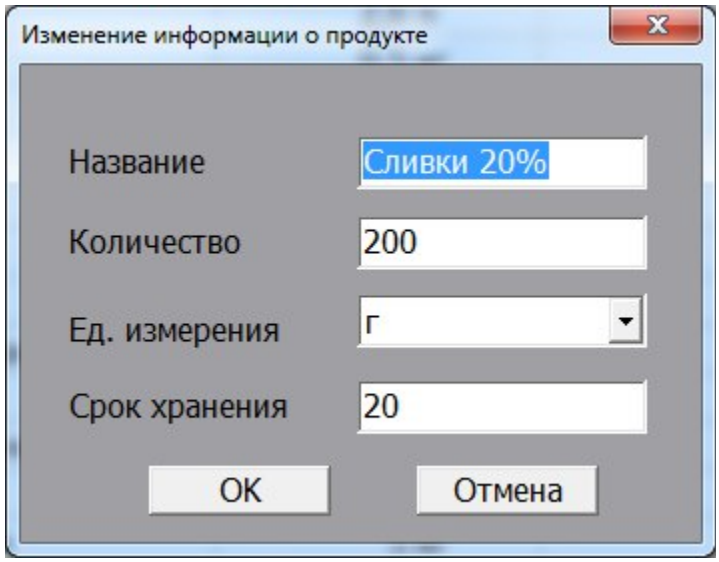


Рисунок 3.10 – Окно редактирования продукта

При выборе пункта меню «справочники» и подпункта «блюда и их состав» появляется окно, в котором в одной таблице отображен перечень всевозможных блюд, их цена и наличие в готовом виде, а в другой таблице при выборе блюда отображается его состав, а именно, название продукта, входящего в блюдо, количество продукта в блюде и единица измерения продукта. Для удобства пользователя имеется возможность поиска по блюдам, для этого нужно ввести

название блюда в строку поиска и произойдет позиционирование, если таковое блюдо есть в справочнике, что видно на рисунке (рисунок 3.11).

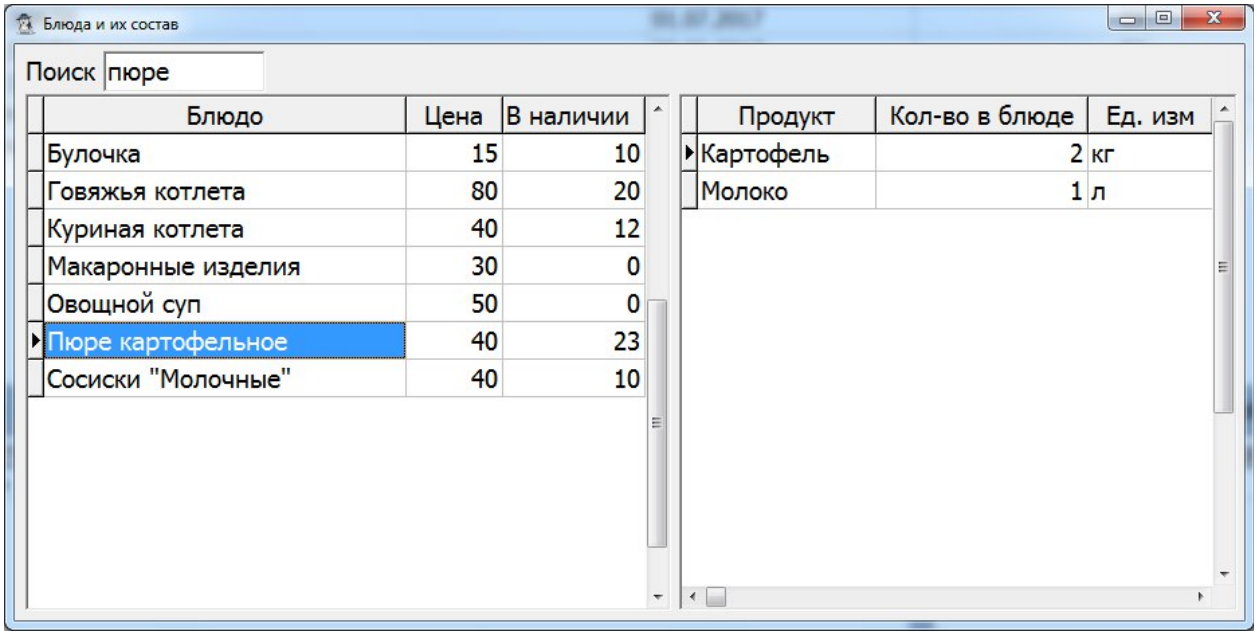


Рисунок 3.11 – Окно списка блюд и их составов

При добавлении нового блюда появляется окно, в котором нужно ввести название продукта, цену и количество готовых блюд. После добавления соответствующие изменения отображаются в таблице блюд (рисунок 3.12).

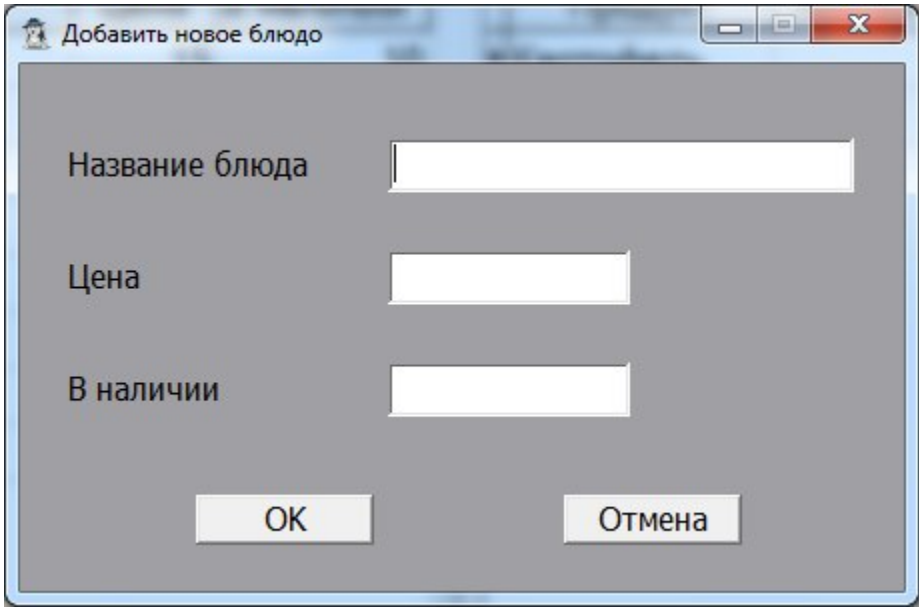


Рисунок 3.12 – Окно добавления нового блюда

При изменении данных об уже существующем блюде появляется окно, в котором уже подставлены все данные и их можно поменять по своему

усмотрению. После изменения соответствующие данные отображаются в редактируемой таблице (рисунок 3.13).

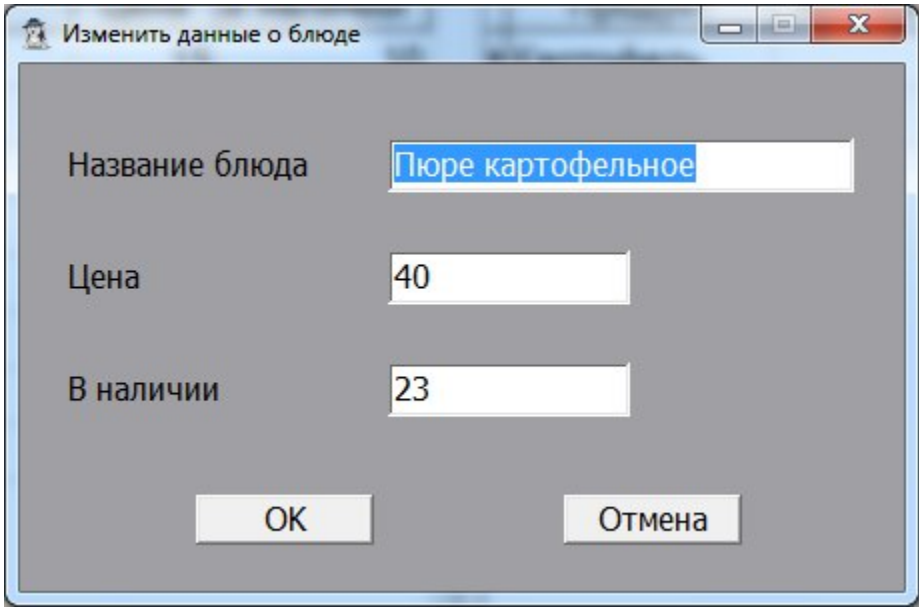


Рисунок 3.13 – Окно редактирования информации о блюде

При добавлении продукта в состав блюда появляется окно, в котором нужно ввести название продукта и количество продукта в блюде. После добавления соответствующие изменения отображаются в таблице состава блюд. Единица измерения подставляется автоматически (рисунок 3.14).

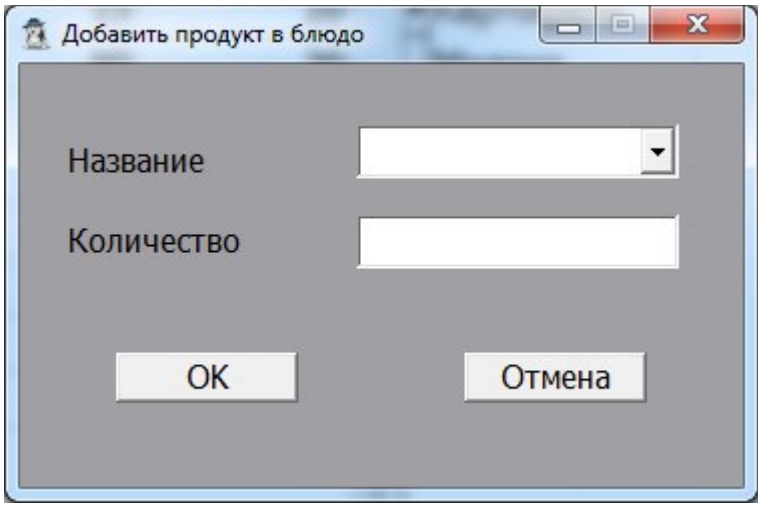


Рисунок 3.14 – Окно добавления продукта в состав блюда

При изменении данных об уже существующем продукте в составе блюда появляется окно, в котором уже подставлены все данные и их можно поменять по своему усмотрению. После изменения соответствующие данные отображаются в редактируемой таблице (рисунок 3.15).

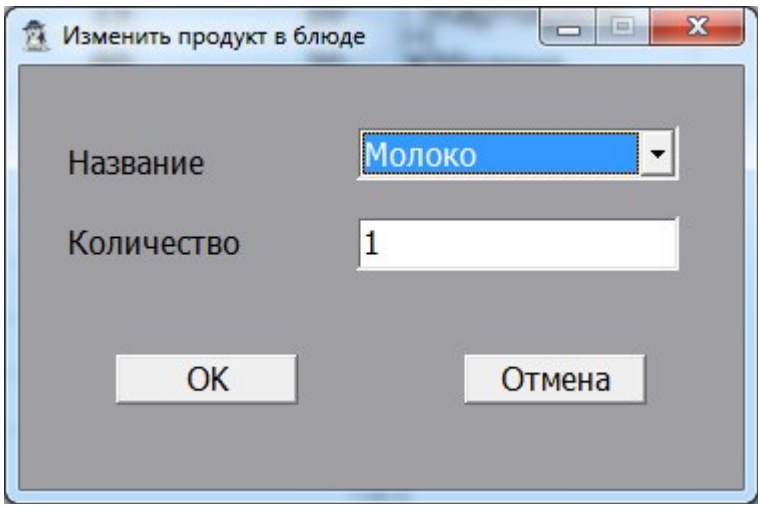


Рисунок 3.15 – Окно редактирования информации о продукте в составе блюда

При выборе пункта меню «справочники» и подпункта «Комплексные обеды» появляется окно, в котором в одной таблице отображен перечень всевозможных комплексных обедов, а в другой таблице при выборе комплексного обеда отображается его состав, а именно, название блюда и его количество в обеде (рисунок 3.16).

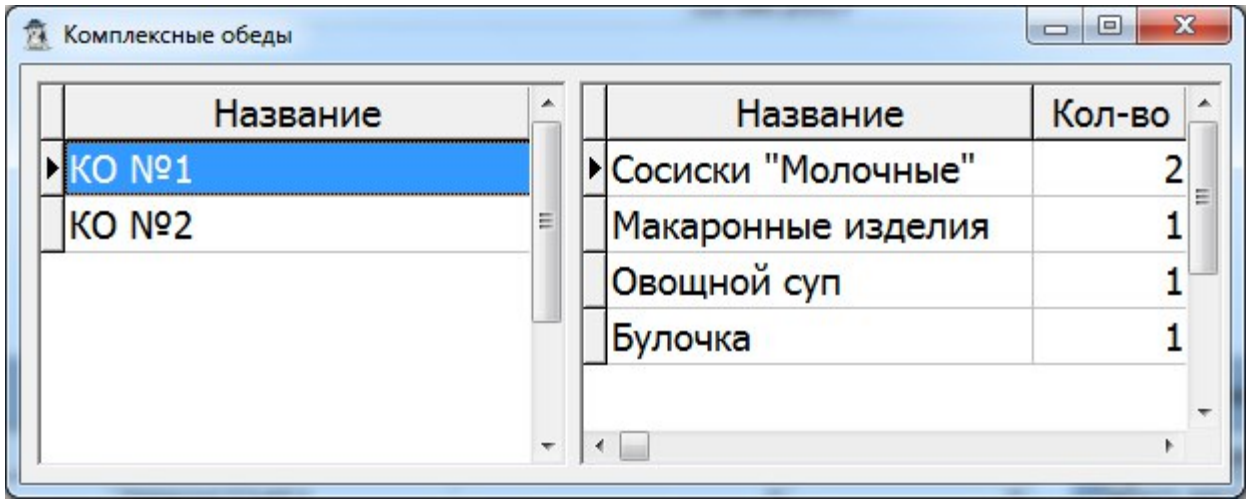


Рисунок 3.16 – Справочное окно «Комплексные обеды»

При добавлении комплексного обеда появляется окно, в котором нужно ввести название комплексного обеда. После добавления соответствующие изменения отображаются в таблице комплексных обедов (рисунок 3.17).

При изменении данных об уже существующем комплексном обеде появляется окно, в котором уже подставлены все данные и их можно поменять по своему усмотрению. После изменения соответствующие данные отображаются в редактируемой таблице (рисунок 3.18).

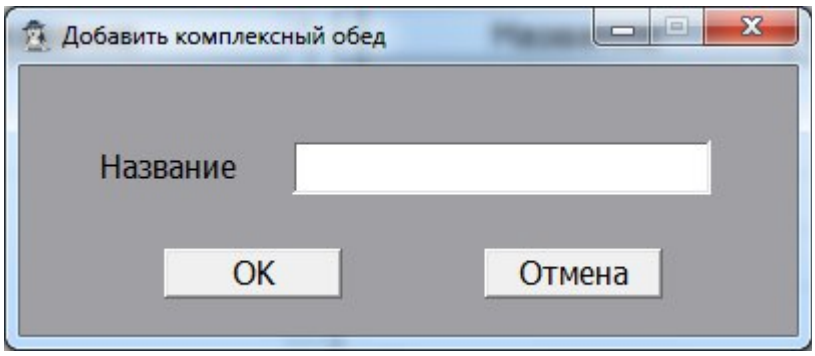


Рисунок 3.17 – Окно добавления нового комплексного обеда

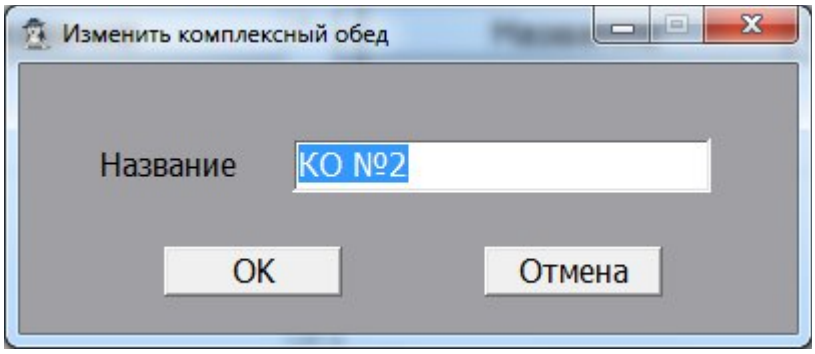


Рисунок 3.18 – Окно редактирования существующего комплексного обеда

При добавлении блюда в состав комплексного обеда появляется окно, в котором нужно выбрать название блюда и количество блюд в обеде. После добавления соответствующие изменения отображаются в таблице состава комплексного обеда (рисунок 3.19).

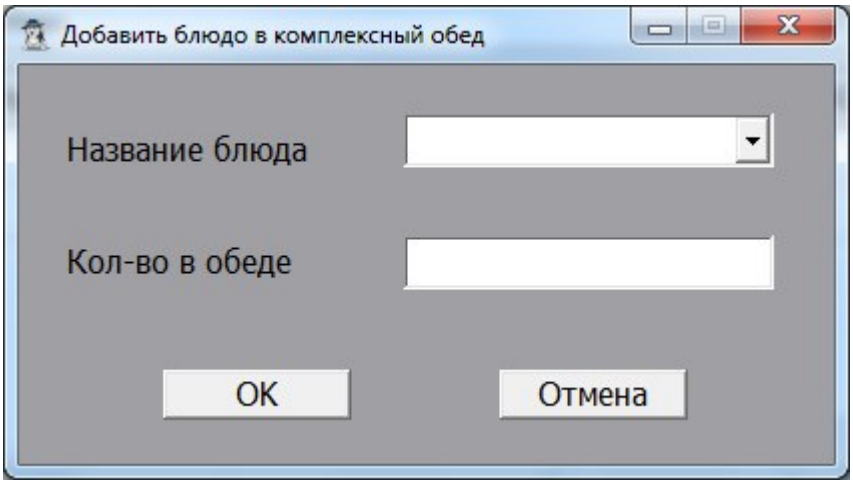


Рисунок 3.19 – Окно добавления нового блюда в состав комплексного обеда

При изменении данных об уже существующем блюде в составе комплексного обеда появляется окно, в котором уже подставлены все данные и их можно поменять по своему усмотрению. После изменения соответствующие данные отображаются в редактируемой таблице (рисунок 3.20).

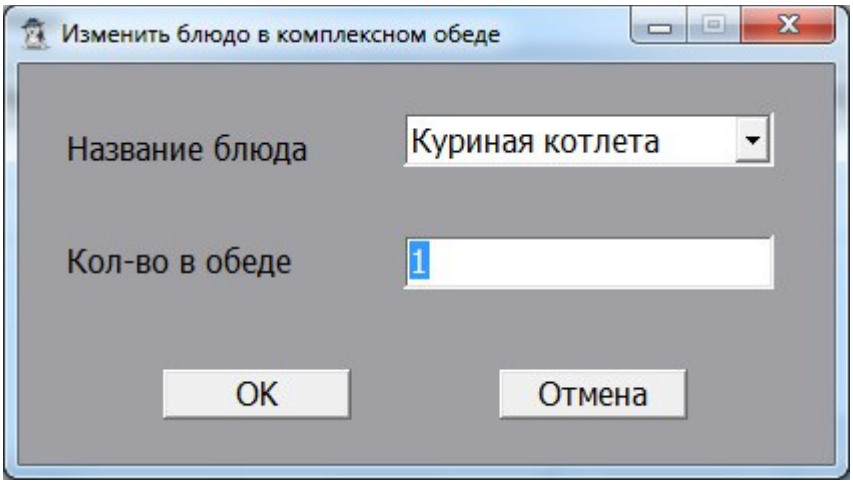


Рисунок 3.20 – Окно редактирования существующего блюда в составе комплексного обеда

При выборе пункта меню «справочники» и подпункта «Единицы измерения» появляется окно, в котором отображен перечень всевозможных единиц измерения, а именно, их сокращенное название и полное название (рисунок 3.21).

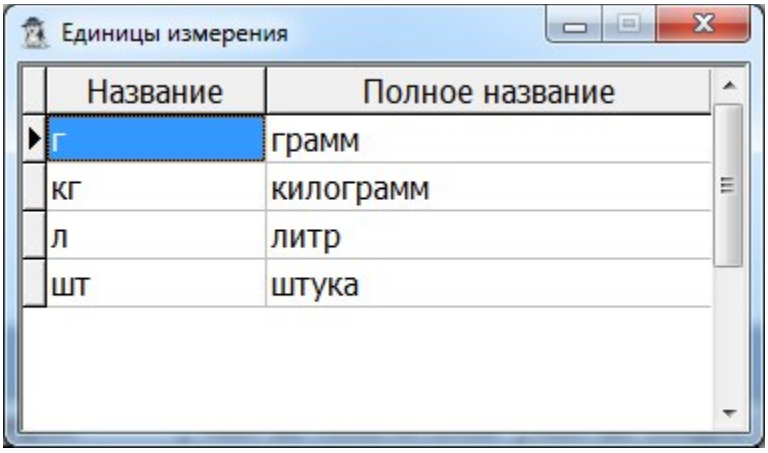


Рисунок 3.21 – Справочное окно «Единицы измерения»

При добавлении новой единицы измерения появляется окно, в котором нужно ввести название блюда и количество блюд в обеде. После добавления соответствующие изменения отображаются в таблице состава комплексного обеда (рисунок 3.22).

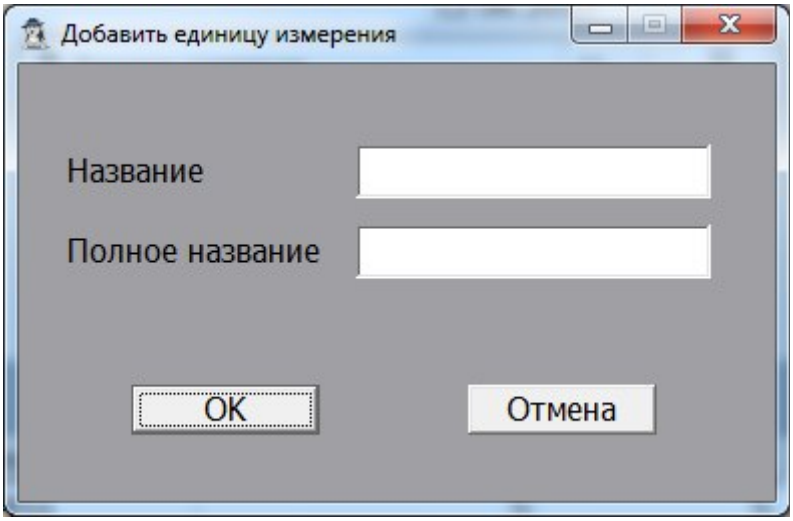


Рисунок 3.22 – Окно добавления новой единицы измерения

При изменении данных об уже существующей единице измерения появляется окно, в котором уже подставлены все данные и их можно поменять по своему усмотрению. После изменения соответствующие данные отображаются в редактируемой таблице (рисунок 3.23).

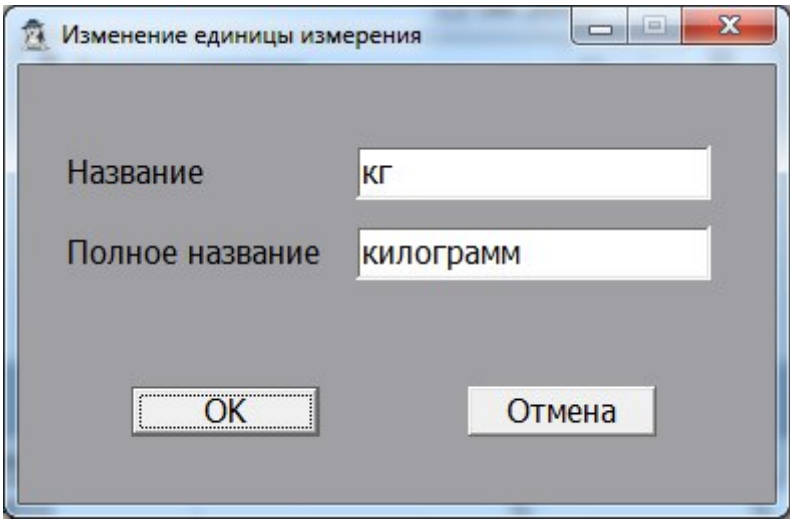


Рисунок 3.23 – Окно редактирования существующей единицы измерения

При выборе пункта меню «справочники» и подпункта «Типы экскурсий» появляется окно, в котором отображен перечень существующих типов экскурсий, а именно, их название, цена, время начала и время окончания экскурсии (рисунок 3.24).

При добавлении нового типа экскурсии появляется окно, в котором нужно ввести название типа экскурсии, цену, время начала и время окончания экскурсии. После добавления соответствующие изменения отображаются в таблице (рисунок 3.25).

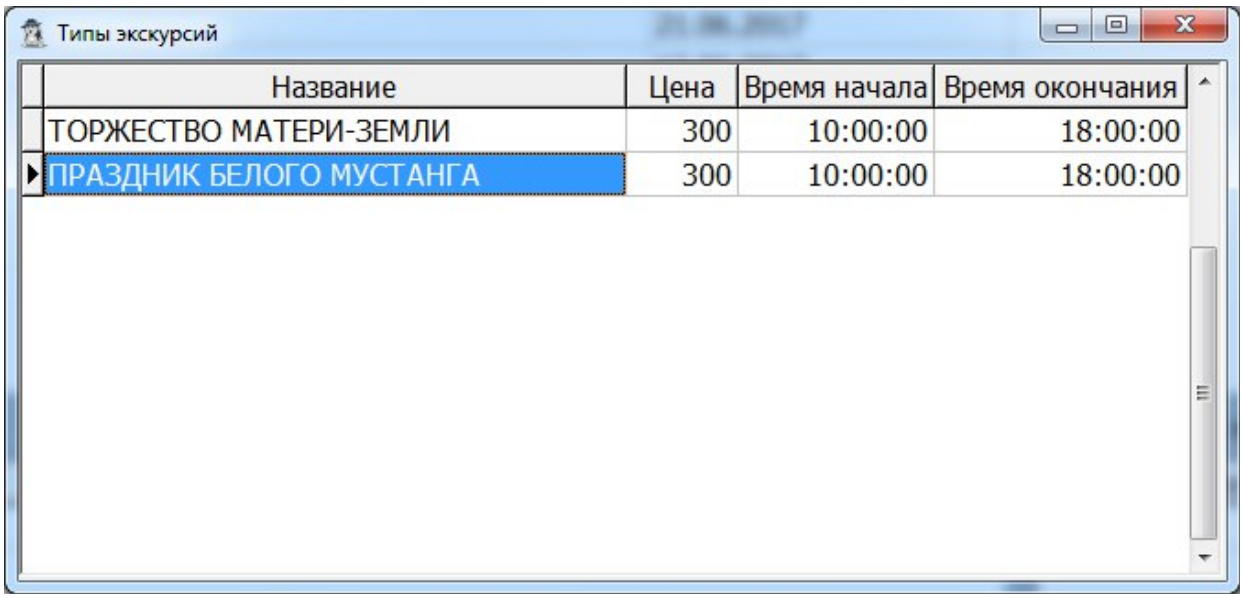


Рисунок 3.24 – Справочное окно «Типы экскурсий»

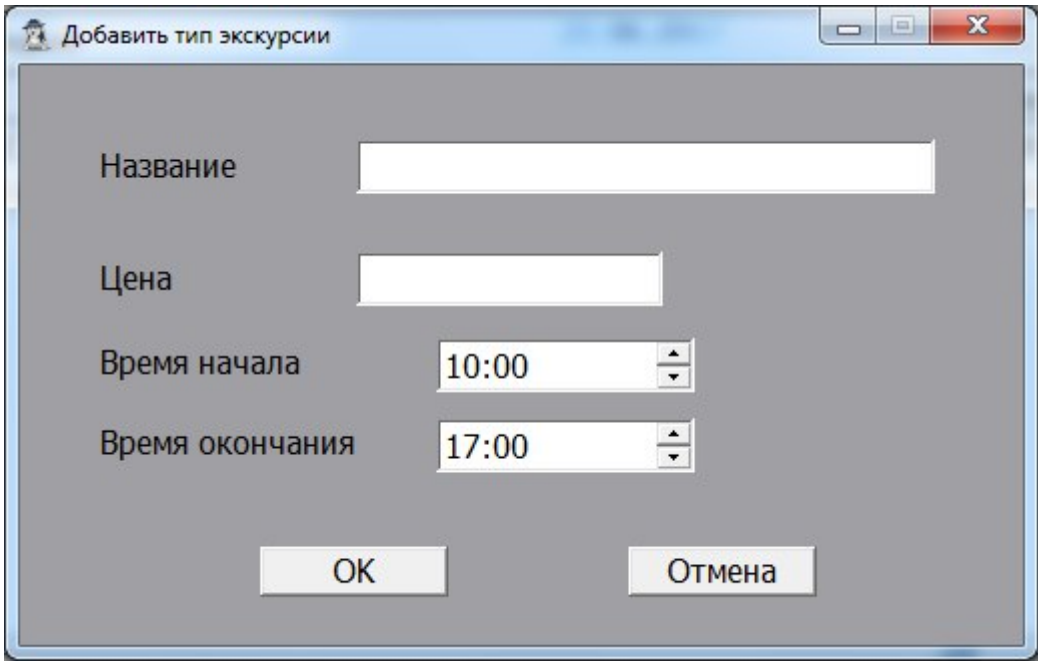


Рисунок 3.25 – Окно добавления нового типа экскурсий

При изменении данных об уже существующем типе экскурсий появляется окно, в котором уже подставлены все данные и их можно поменять по своему усмотрению. После изменения соответствующие данные отображаются в редактируемой таблице (рисунок 3.26).

При выборе пункта меню «отчеты» и подпункта «План закупок» появляется окно, в котором необходимо выбрать дату, до которой произойдет планирование закупки продуктов. После этого нужно нажать кнопку «В Excel» (рисунок 3.27).

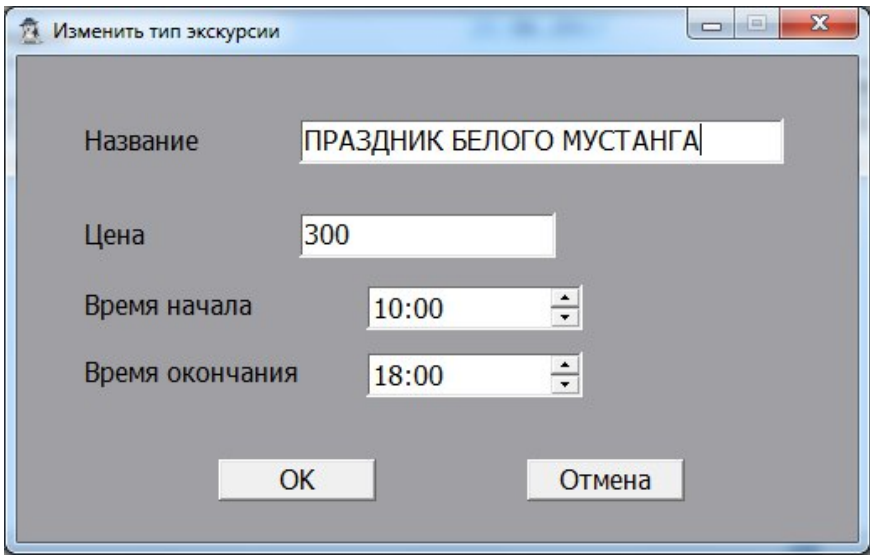


Рисунок 3.26 – Окно редактирования существующего типа экскурсии

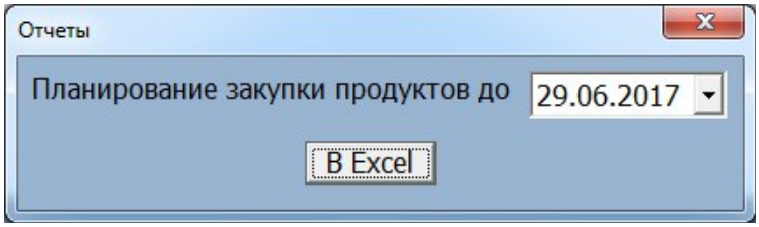


Рисунок 3.27 – Окно «План закупок»

После чего происходит переход в Excel, там рисуется таблица с перечислением списка продуктов, единицы измерения, количества продукта в наличии и количества, которое нужно закупить. При желании таблицу можно сохранить и распечатать (рисунок 3.28).

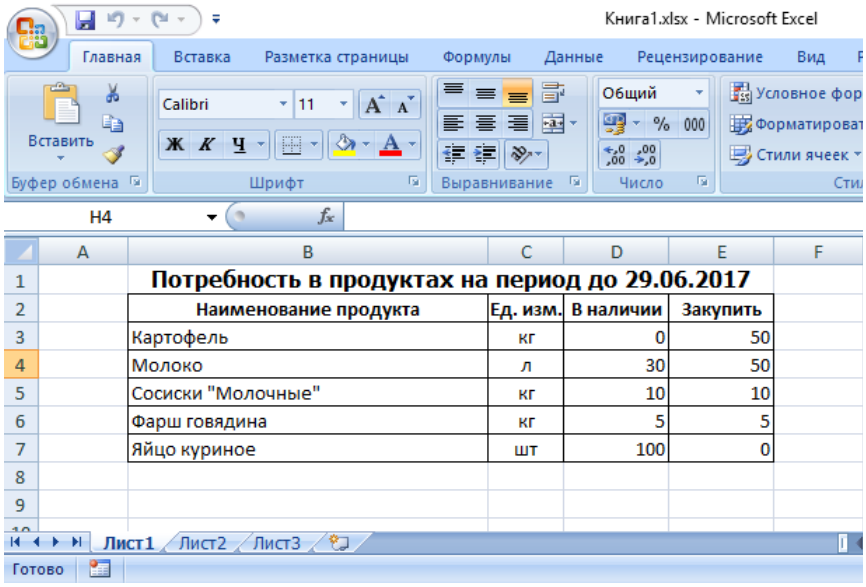


Рисунок 3.28 – Выведенный план закупки продуктов в Excel

При выборе пункта меню «Справка» появляется окно, в котором указаны сведения о разработчике программы (рисунок 3.29).

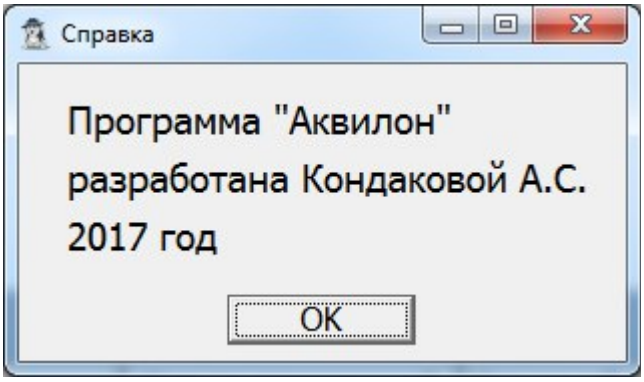


Рисунок 3.29 – Окно «Справка»

3.5 Выводы по разделу

В этом разделе были выявлены требования к интерфейсу программы и, в соответствии с ними, был реализован понятный и удобный интерфейс.

Разработаны и описаны модули приложения, которые позволяют добавлять и редактировать продукты, блюда, состав блюд, комплексные обеды, состав комплексных обедов, типы экскурсий, а также единицы измерения. Реализован поиск продуктов и блюд в таблицах.

На конечном этапе реализации программы было произведено тестирование разработанного приложения. Программа проверялась на тестовых данных, были использованы не все продукты, не все блюда, составы блюд были не полностью заполнены. В ходе тестирования программы были выявлены ошибки. Они были исправлены, программа доработана и отлажена. Была добавлена функция планирования запаса продуктов.

ЗАКЛЮЧЕНИЕ

В ходе работы над выпускной квалификационной работой были выполнены следующие задачи:

- сформулированы требования к разрабатываемому программному продукту;
- рассмотрены схожие поставленной задачи, выявлены их достоинства и недостатки, учтенные затем при разработке;
- проведен обоснованный выбор средств и сред для разработки приложения;
- спроектирована база данных;
- составлены алгоритмы работы приложения;
- разработан программный продукт, соответствующий поставленным к нему требованиям;
- протестирована работа приложения на тестовом наполнении базы данных;
- разработана и оформлена программная документация к приложению.

В результате работы получен функционирующий программный продукт. Испытания программы на тестовых данных прошли успешно.

В настоящее время происходит процесс внедрения программных средств на предприятие, в дальнейшем программа будет дорабатываться в соответствии с пожеланиями заказчика.

Возможно расширение функциональности за счет изменений в структуре базы данных, повышение удобства работы с приложением за счет изменения разметки и визуального оформления.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Архангельский, А.Я. Программирование в C++ Builder / А.Я. Архангельский. – М.: Бином, 2014. – 1230 с.
2. Дейтел, Х. Как программировать на C / Х. Дейтел, П. Дейтел. – М.: Бином-Пресс, 2009. – 910 с.
3. Краморенко Н.В. Базы данных / Н.В. Краморенко. – Владивосток.: ДВГУ, 2004. – 85 с.
4. Культин, Н.Б. Самоучитель C++ Builder / Н.Б. Культин. – СПб.: БВХ - Петербург, 2004. – 320 с.
5. Петкович, Д. Microsoft SQL Server 2008 / Д. Петкович; пер. с англ. А. Бондаря. – Спб.: БХВ-Петербург, 2009. – 752 с.
6. Подбельский, В.В. Программирование на языке C / В.В. Подбельский, С.С. Фомин. – М.; Финансы и статистика, 2004. – 600 с.
7. Рейсдорф, К. Borland C++ Builder. Освой самостоятельно / К. Рейсдорф, К. Хендерсон. – М.: Бином, 1998. – 700 с.
8. Forecast NOW! [Электронный ресурс] URL: https://fnow.ru/ (Дата обращения 20.02.2017)
9. Oremax [Электронный ресурс] URL: https://oremax.ru/ (Дата обращения 20.02.2017)
10. Poster POS [Электронный ресурс] URL: https://joinposter.com (Дата обращения 20.02.2017)
11. БЭСТ Программы для бизнеса [Электронный ресурс] URL: https://bestnet.ru/programs/pitanie/ (Дата обращения 20.02.2017)

ПРИЛОЖЕНИЕ 1

Текст программы

MainUnit.cpp

```
#include <vcl.h>
#pragma hdrstop

#include "MainUnit.h"
#include "LoginUnit.h"
#include "DbUnit.h"
#include "ProdDishUnit.h"
#include "Units.h"
#include "Complex.h"
#include "TypeExcurs.h"
#include "Product.h"
#include "VisitorWnd.h"
#include "Msg.h"
#include "Tools.h"
#include "EdaWnd.h"
#include "ExcursWnd.h"
#include "Report.h"
#include "Spravka.h"
TADOConnection *Conn;
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TfrmMain *frmMain;

struct REASON{ // причины неприятностей и сообщения
   int ErrCode;
   char *Msg;
};
static REASON Reason[]={
   {USER_CANCEL,"Пользователь отказался"},
     {NO_INIFILE,"Не найден ini - файл программы"},
     {DB_NOTOPEN,"Не удалось открыть БД"},
     {NO_SERVERNAME,"В ini-файле не указан сервер"},
     {NO_DATABASENAME,"В ini-файле не указан база данных"},
};

//---------------------------------------------------------------------------
__fastcall TfrmMain::TfrmMain(TComponent* Owner)
```

```
    : TForm(Owner)
{
}

//---------------------------------------------------------------------------
void __fastcall TfrmMain::FormCreate(TObject *Sender){
int Ret,i;
AnsiString ServerMessage,s;
bool Found;

CloseMainForm=false;
if((Ret=Login(DataMod->ADOConnection,ServerMessage))!=0){
    // найдем причину отказа в регистрации
    for(i=0;i<sizeof(Reason)/sizeof(REASON);i++){
        if(Reason[i].ErrCode==Ret){
            Found=true;
            break;
        }
    }
    if(Found){
        s=Reason[i].Msg;
    } else {
        s="Программа не может стартовать по неизвестной причине";
    }
    s=s+"\n"+ServerMessage;
    MessageBox(0,s.c_str(),"Фатальная ошибка",MB_OK);
}

frmMain->Width=1280;
frmMain->Height=800;

}

//---------------------------------------------------------------------------
void __fastcall TfrmMain::FormActivate(TObject *Sender){
Conn=DataMod->ADOConnection;
if(!DataMod->ADOConnection->Connected){
    this->Close();
}

// установка периода дат для которого отображаются данные в программе
// по умолчанию конец периода - конец месяца,
// начало периода - сегодня
DateStart=ServerDate(Conn);
```

```
DateFin=DateStart+30;
dtp1->DateTime=DateStart;
dtp2->DateTime=DateFin;
queExcurs->Parameters->ParamByName("d1")->Value=DateStart;
queExcurs->Parameters->ParamByName("d2")->Value=DateFin;
queExcurs->Active=true;
queVisitor->Active=true;
queEda->Active=true;
}

//---------------------------------------------------------------------------
void __fastcall TfrmMain::mnuDishClick(TObject *Sender){
TfrmProdDish *f=new TfrmProdDish(this);
f->ShowModal();
delete f;
return;
}

//---------------------------------------------------------------------------
void __fastcall TfrmMain::mnuEdIzmClick(TObject *Sender){
TfrmUnits *f=new TfrmUnits(this);
f->ShowModal();
delete f;
return;
}

//---------------------------------------------------------------------------
void __fastcall TfrmMain::mnuComplexClick(TObject *Sender){
TfrmComplex*f=new TfrmComplex(this);
f->ShowModal();
delete f;
return;
}

//---------------------------------------------------------------------------
void __fastcall TfrmMain::N2Click(TObject *Sender){
TfrmTypeExcurs*f=new TfrmTypeExcurs(this);
f->ShowModal();
delete f;
return;
}

//---------------------------------------------------------------------------
void __fastcall TfrmMain::queExcursAfterScroll(TDataSet *DataSet){
```

```
queVisitor->Active=false;
queVisitor->Parameters->ParamByName("Excurs_ID")->Value=
  queExcurs->FieldByName("Excurs_ID")->AsInteger;
queVisitor->Active=true;
queEda->Active=false;
queEda->Parameters->ParamByName("Visitor_ID")->Value=
  queVisitor->FieldByName("Visitor_ID")->Value;
queEda->Active=true;
}

//---------------------------------------------------------------------------
void __fastcall TfrmMain::queVisitorAfterScroll(TDataSet *DataSet){
queEda->Active=false;
queEda->Parameters->ParamByName("Visitor_ID")->Value=
  queVisitor->FieldByName("Visitor_ID")->AsInteger;
queEda->Active=true;
}

//---------------------------------------------------------------------------
void __fastcall TfrmMain::N5Click(TObject *Sender){
TfrmProduct*f=new TfrmProduct(this);
f->ShowModal();
delete f;
return;
}

//---------------------------------------------------------------------------
// Добавить посетителя в экскурсию
//---------------------------------------------------------------------------
void __fastcall TfrmMain::mnuInsVisitorClick(TObject *Sender){
TfrmVisitorWnd *f=new TfrmVisitorWnd(this);
f->Caption="Добавить посетителей в экскурсию";
f->queInsVisitor->Active=true;
f->edtFIO->Text="";
f->edtPhoneFIO->Text="";
f->edtAmountVisitor->Text="";
if(f->ShowModal()!=mrOk){
      delete f;
      return;
}
Variant FIO=f->edtFIO->Text;
Variant PhoneFIO=f->edtPhoneFIO->Text;
Variant AmountVisitor=f->edtAmountVisitor->Text;
Variant Excurs_ID=queExcurs->FieldByName("Excurs_ID")->AsInteger;
```

```
    TADOStoredProc *st=CreateStProc(DataMod->ADOConnection, "dbo.insVisitor",
      "@FIO", ftString,ptInput,FIO,
      "@AmountVisitor", ftInteger ,ptInput, AmountVisitor,
      "@Excurs_ID", ftInteger,ptInput,Excurs_ID,
      "@PhoneFIO", ftString,ptInput,PhoneFIO,
      "@id", ftInteger,ptOutput,(Variant)0,
      0);
    st->ExecProc();
    int ID=int(st->Parameters->ParamValues["@id"]);
    delete st;
    delete f;

    SetCursorPosition(queVisitor,"Visitor_ID",ID);
    RedrawAllGrids();
    }

    //---------------------------------------------
    void TfrmMain::RedrawAllGrids(){
    // перерисовать все grid
    Variant idExc,idVis,idEd; // идентификаторы текущих записей в grid-ах
    idExc=queExcurs->IsEmpty() ? Null() : queExcurs->FieldByName("Excurs_ID")-
>Value;
    idVis=queVisitor->IsEmpty() ? Null() : queVisitor->FieldByName("Visitor_ID")-
>Value;
    idEd=queEda->IsEmpty() ? Null() : queEda->FieldByName("Eda_ID")->Value;
    queEda->Active=false;
    queVisitor->Active=false;
    queExcurs->Active=false;

    queExcurs->Active=true;

    queExcurs->Active=true;
    SetCursorPosition(queExcurs,"Excurs_ID",idExc);
    }

    //---------------------------------------------
    // Редактировать посетителя в экскурсии
    //---------------------------------------------
    void __fastcall TfrmMain::mnuUpdVisitorClick(TObject *Sender){
    if(queVisitor->IsEmpty()){
        // нечего редактировать
        return;
    }
```

```
    TfrmVisitorWnd *f=new TfrmVisitorWnd(this);
    f->Caption="Изменить данные о посетителях";
    Variant Visitor_ID=queVisitor->FieldByName("Visitor_ID")->Value;
    Variant FIO=queVisitor->FieldByName("FIO")->Value;
    // подставить в окно редактирования текущее значение
    // названия блюда
    Variant Price=queVisitor->FieldByName("PhoneFIO")->Value;
    Variant AmountVisitor=queVisitor->FieldByName("AmountVisitor")->Value;


    f->edtFIO->Text=queVisitor->FieldByName("FIO")->AsString;
    f->edtPhoneFIO->Text=queVisitor->FieldByName("PhoneFIO")->AsString;
    f->edtAmountVisitor->Text=queVisitor->FieldByName("AmountVisitor")-
>AsString;
    if(f->ShowModal()!=mrOk){
        delete f;
        return;
    }
    TADOQuery *q=CreateQuery(DataMod->ADOConnection,
     "update Visitor set "
     "  FIO=:FIO, "
     "  AmountVisitor=:AmountVisitor, "
     "  PhoneFIO=:PhoneFIO "
     "where Visitor_ID=:Visitor_ID",
     "FIO",ftString,(Variant)f->edtFIO->Text,
     "AmountVisitor",ftInteger,(Variant)f->edtAmountVisitor->Text,
     "PhoneFIO",ftString,(Variant)f->edtPhoneFIO->Text,
     "Visitor_ID",ftInteger,Visitor_ID,
     0);
    StartTran(Conn);
    try{
        q->ExecSQL();
        CommitTran(Conn);
    }
    catch(Exception &e){
        RollbackTran(Conn);
        AnsiString s="";
        s="Не удалось изменить запись.\n";
        s=s+e.Message;
        Msg(USER_ERROR,NULL,s.c_str());
    }

    delete q;
    delete f;
```

```
    SetCursorPosition(queVisitor,"Visitor_ID",Visitor_ID);
    RedrawAllGrids();
    }

    //---------------------------------------------------------------------------
    // Удалить посетителя из экскурсии
    //---------------------------------------------------------------------------
    void __fastcall TfrmMain::mnuDelVisitorClick(TObject *Sender){
     if(queVisitor->IsEmpty()){
         // нечего удалять
         return;
    }
    if(!queEda->IsEmpty()){
         Msg(USER_ERROR,NULL,
          "Удаление невозможно, пока за этим лицом числятся заказанные
комплексные обеды");
         return;
    }

    if(0!=Msg(QUESTION,"Да|Нет","Вы действительно хотите удалить запись?")){
         return;
    }

    int ID=queVisitor->FieldByName("Visitor_ID")->AsInteger;
    int ID1=NearestRecordID(queVisitor,"Visitor_ID");
    TADOQuery *q=CreateQuery(DataMod->ADOConnection,
      "delete from Visitor where Visitor_ID=:Visitor_ID",
      "Visitor_ID",ftInteger,(Variant)ID,
      0);
    q->ExecSQL();
    delete q;
    SetCursorPosition(queVisitor,"Visitor_ID",ID1);
    RedrawAllGrids();
    }

    //---------------------------------------------------------------------------
    // Добавить комплексный обед для посетителя
    //---------------------------------------------------------------------------
    void __fastcall TfrmMain::mnuInsComplClick(TObject *Sender)
    {
    TfrmEdaWnd *f=new TfrmEdaWnd(this);
    f->queInsEda->Active=true;
    f->Caption="Добавить обед для экскурсантов";
    f->cmbComplex->KeyValue=0;
```

```
f->edtAmountEda->Text="";
if(f->ShowModal()!=mrOk){
      delete f;
      return;
}

Variant Visitor_ID=queVisitor->FieldByName("Visitor_ID")->AsInteger;
Variant Complex_ID=f->cmbComplex->KeyValue;
Variant AmountEda=f->edtAmountEda->Text;

TADOStoredProc *st=CreateStProc(DataMod->ADOConnection, "dbo.insEda",
  "@Visitor_ID", ftInteger,ptInput,Visitor_ID,
  "@Complex_ID", ftInteger,ptInput, Complex_ID,
  "@AmountEda", ftInteger,ptInput,AmountEda,
  "@id", ftInteger,ptOutput,(Variant)0,
  0);
st->ExecProc();
int ID=int(st->Parameters->ParamValues["@id"]);
delete st;
delete f;
SetCursorPosition(queEda,"Complex_ID",ID);
RedrawAllGrids();
}

//---------------------------------------------------------------------------
// Изменить комплексный обед для посетителя
//---------------------------------------------------------------------------
void __fastcall TfrmMain::mnuUpdComplClick(TObject *Sender)
{
 if(queEda->IsEmpty()){
      // нечего редактировать
      return;
}
TfrmEdaWnd *f=new TfrmEdaWnd(this);
f->Caption="Обед для экскурсантов";
int Eda_ID=queEda->FieldByName("Eda_ID")->AsInteger;
Variant Complex_ID=queEda->FieldByName("Complex_ID")->Value;
f->cmbComplex->KeyValue=queEda->FieldByName("Complex_ID")->AsInteger;

// подставить в окно редактирования текущее значение
// названия продукта
f->edtAmountEda->Text=queEda->FieldByName("AmountEda")->AsInteger;
if(f->ShowModal()!=mrOk){
      delete f;
```

```
      return;
}
TADOQuery *q=CreateQuery(DataMod->ADOConnection,
  "update Eda set "
  "  Complex_ID=:Complex_ID, "
  "  AmountEda=:AmountEda "
  "where Eda_ID=:Eda_ID",
  "Complex_ID",ftInteger,f->cmbComplex->KeyValue,
  "AmountEda",ftInteger,(Variant)f->edtAmountEda->Text,
  "Eda_ID",ftInteger,Variant(Eda_ID),
  0);
q->ExecSQL();
delete q;
delete f;
SetCursorPosition(queEda,"Complex_ID",Complex_ID);
RedrawAllGrids();
}

//---------------------------------------------------------------------------
// Удалить комплексный обед для посетителя
//---------------------------------------------------------------------------
void __fastcall TfrmMain::mnuDelComplClick(TObject *Sender){
  if(queEda->IsEmpty()){
      // нечего удалять
      return;
}

if(0!=Msg(QUESTION,"Да|Нет","Вы действительно хотите удалить запись?")){
      return;
}
int ID=queEda->FieldByName("Eda_ID")->AsInteger;
int ID1=NearestRecordID(queEda,"Eda_ID");
TADOQuery *q=CreateQuery(DataMod->ADOConnection,
  "delete from Eda where Eda_ID=:Eda_ID",
  "Eda_ID",ftInteger,(Variant)ID,
  0);
q->ExecSQL();
delete q;
SetCursorPosition(queEda,"Eda_ID",ID1);
RedrawAllGrids();
}

//---------------------------------------------
void TfrmMain::ReopenExcurs(){
```

```
DateStart=dtp1->DateTime;
DateFin=dtp2->DateTime;
queExcurs->Active=false;
queExcurs->Parameters->ParamByName("d1")->Value=DateStart;
queExcurs->Parameters->ParamByName("d2")->Value=DateFin;
queExcurs->Active=true;
}

//---------------------------------------------
void __fastcall TfrmMain::dtp1Change(TObject *Sender){
ReopenExcurs();
}

//---------------------------------------------------------------------------
void __fastcall TfrmMain::dtp2Change(TObject *Sender){
ReopenExcurs();
}

//---------------------------------------------------------------------------
// Добавить экскурсию
//---------------------------------------------------------------------------
void __fastcall TfrmMain::InsExcClick(TObject *Sender){
TfrmExcursWnd *f=new TfrmExcursWnd(this);
//f->queInsExcurs->Active=true;
f->Caption="Новая экскурсия";
f->cmbExcurs->KeyValue=0;
f->dtp->DateTime=RoundDateTime(ServerDate(DataMod->ADOConnection));
f->queInsExcurs->Active=true;
f->cmbExcurs->KeyValue=0;

if(f->ShowModal()!=mrOk){
      delete f;
      return;
}

Variant TypeExcurs_ID=f->cmbExcurs->KeyValue;
Variant Data=f->dtp->DateTime;

TADOStoredProc *st=CreateStProc(DataMod->ADOConnection, "dbo.insExcurs",
  "@Data", ftDateTime,ptInput,Data,
  "@TypeExcurs_ID", ftInteger,ptInput,TypeExcurs_ID,
  "@id", ftInteger,ptOutput,(Variant)0,
  0);
st->ExecProc();
```

```
int ID=int(st->Parameters->ParamValues["@id"]);
delete st;
delete f;

queVisitor->Active=false;
queEda->Active=false;
SetCursorPosition(queExcurs,"TypeExcurs_ID",ID);
}

//---------------------------------------------------------------------------
// Изменить экскурсию
//---------------------------------------------------------------------------
void __fastcall TfrmMain::UpdEcxClick(TObject *Sender){
if(queExcurs->IsEmpty()){
      // нечего редактировать
      return;
}
TfrmExcursWnd *f=new TfrmExcursWnd(this);
f->Caption="Редактирование экскурсии";
int Excurs_ID=queExcurs->FieldByName("Excurs_ID")->AsInteger;
Variant TypeExcurs_ID=queExcurs->FieldByName("TypeExcurs_ID")->Value;
f->queInsExcurs->Active=true;
f->cmbExcurs->KeyValue=TypeExcurs_ID;
Variant d=queExcurs->FieldByName("Data")->AsDateTime;
f->dtp->DateTime=d;

if(f->ShowModal()!=mrOk){
      delete f;
      return;
}
TypeExcurs_ID=f->cmbExcurs->KeyValue;
d=RoundDateTime(f->dtp->DateTime);

TADOQuery *q=CreateQuery(DataMod->ADOConnection,
  "update Excurs set "
  "  Data=:Data, "
  "  TypeExcurs_ID=:TypeExcurs_ID "
  "where Excurs_ID=:Excurs_ID",
  "Data",ftDateTime,d,
  "TypeExcurs_ID",ftInteger,TypeExcurs_ID,
  "Excurs_ID",ftInteger,Variant(Excurs_ID),
  0);
q->ExecSQL();
delete q;
```

```
    SetCursorPosition(queExcurs,"Excurs_ID",Excurs_ID);
    delete f;
    }

    //---------------------------------------------------------------------------
    // Удалить экскурсию
    //---------------------------------------------------------------------------
    void __fastcall TfrmMain::DelExcClick(TObject *Sender){
    if(queExcurs->IsEmpty()){
        // нечего удалять
        return;
    }
    if(!queVisitor->IsEmpty()){
        Msg(USER_ERROR,NULL,"Удаление невозможно пока имеются заявки на
экскурсию");
        return;
    }
    if(0!=Msg(QUESTION,"Да|Нет","Вы действительно хотите удалить запись?")){
        return;
    }

    int ID=queExcurs->FieldByName("Excurs_ID")->AsInteger;
    int ID1=NearestRecordID(queExcurs,"Excurs_ID");
    TADOQuery *q=CreateQuery(DataMod->ADOConnection,
      "delete from Excurs where Excurs_ID=:Excurs_ID",
      "Excurs_ID",ftInteger,(Variant)ID,
      0);
    q->ExecSQL();
    delete q;
    SetCursorPosition(queExcurs,"Excurs_ID",ID1);
    }

    //---------------------------------------------------------------------------
    void __fastcall TfrmMain::mnuPlanClick(TObject *Sender){
    TfrmReport *f=new TfrmReport(this);
    f->ShowModal();
    delete f;
    return;
    }

    //---------------------------------------------------------------------------
    void __fastcall TfrmMain::grdExcursKeyDown(TObject *Sender, WORD
&Key,TShiftState Shift){
    switch(Key){
```

```
        case VK_INSERT:
                InsExcClick(grdExcurs);
                break;
        case VK_DELETE:
                DelExcClick(grdExcurs);
                break;
        case VK_RETURN:
                UpdEcxClick(grdExcurs);
                break;
        default:
                Key=0;
    }
    }
    //---------------------------------------------------------------------------

    void __fastcall TfrmMain::grdVisitorKeyDown(TObject *Sender, WORD &Key,
          TShiftState Shift)
    {
    switch(Key){
        case VK_INSERT:
                InsExcClick(grdVisitor);
                break;
        case VK_DELETE:
                DelExcClick(grdVisitor);
                break;
        case VK_RETURN:
                UpdEcxClick(grdVisitor);
                break;
        default:
                Key=0;
    }
    }
    //---------------------------------------------------------------------------

    void __fastcall TfrmMain::grdComplexExcursKeyDown(TObject *Sender, WORD
&Key,
        TShiftState Shift)
    {
    switch(Key){
        case VK_INSERT:
                InsExcClick(grdComplexExcurs);
                break;
        case VK_DELETE:
                DelExcClick(grdComplexExcurs);
```

```
            break;
        case VK_RETURN:
            UpdEcxClick(grdComplexExcurs);
            break;
        default:
            Key=0;
}
}
//---------------------------------------------------------------------------

void __fastcall TfrmMain::N3Click(TObject *Sender)
{
TfrmSpravka*f=new TfrmSpravka(this);
f->ShowModal();
delete f;
return;
}
//---------------------------------------------------------------------------
```

DbUnit.cpp

```
#include <vcl.h>
#pragma hdrstop

#include "DbUnit.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TDataMod *DataMod;
//---------------------------------------------------------------------------
__fastcall TDataMod::TDataMod(TComponent* Owner)
    : TDataModule(Owner)
{
}
//---------------------------------------------------------------------------
```

Product.cpp

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Product.h"
#include "ProductsWnd.h"
#include "DbUnit.h"
#include "Tools.h"
#include "Msg.h"
```

```
extern TADOConnection *Conn;
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TfrmProduct *frmProduct;
//---------------------------------------------------------------------------
__fastcall TfrmProduct::TfrmProduct(TComponent* Owner)
    : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TfrmProduct::FormActivate(TObject *Sender){
queProduct->Active=true;
//AjustGridWidth(grdProduct);
}

//---------------------------------------------------------------------------
// Добавить продукт
//---------------------------------------------------------------------------
void __fastcall TfrmProduct::mnuInsProdClick(TObject *Sender){
TfrmProductsWnd *f=new TfrmProductsWnd(this);
f->Caption="Добавить продукт";
f->queInsProd->Active=true;
f->cmbUnit->KeyValue=0;
f->edtAmount->Text="";
f->edtProductName->Text="";
f->edtSrok->Text="";

if(f->ShowModal()!=mrOk){
    delete f;
    return;
}
Variant ProductName=f->edtProductName->Text;
AnsiString cAmount=f->edtAmount->Text;
Variant Unit_ID=f->cmbUnit->KeyValue;
Variant Srok=f->edtSrok->Text.ToIntDef(1);

TADOStoredProc *st=CreateStProc(DataMod->ADOConnection, "dbo.insProduct",
 "@ProductName", ftString,ptInput,ProductName,
 "@Amount",ftString,ptInput,Variant(cAmount),
 "@Unit_ID",ftInteger,ptInput,Unit_ID,
 "@Srok",ftInteger,ptInput,Srok,
 "@id", ftInteger,ptOutput,(Variant)0,
 0);
```

```
    st->ExecProc();
    int ID=int(st->Parameters->ParamValues["@id"]);
    delete st;

    SetCursorPosition(queProduct,"Product_ID",ID);
    delete f;
    }

    //---------------------------------------------------------------------------
    // Изменить информацию о продукте
    //---------------------------------------------------------------------------
    void __fastcall TfrmProduct::mnuUpdProdClick(TObject *Sender){
    if(queProduct->IsEmpty()){
        // нечего редактировать
        return;
    }
    TfrmProductsWnd *f=new TfrmProductsWnd(this);
    f->Caption="Изменение информации о продукте";
    Variant Product_ID=queProduct->FieldByName("Product_ID")->Value;

    Variant Unit_ID=queProduct->FieldByName("Unit_ID")->Value;
    f->cmbUnit->KeyValue=Unit_ID;
    // подставить в окно редактирования текущее значение
    // названия единицы измерения
    f->edtProductName->Text=queProduct->FieldByName("ProductName")-
>AsString;
    f->edtAmount->Text=queProduct->FieldByName("Amount")->AsFloat;
    f->edtSrok->Text=queProduct->FieldByName("Srok")->AsInteger;

    if(f->ShowModal()!=mrOk){
        delete f;
        return;
    }

    TADOQuery *q=CreateQuery(DataMod->ADOConnection,
      "update Product set "
      "  ProductName=:ProductName, "
      "  Amount=:Amount, "
      "  Unit_ID=:Unit_ID, "
      "  Srok=:Srok "
      "where Product_ID=:Product_ID",
      "ProductName",ftString,(Variant)f->edtProductName->Text,
      "Amount",ftFloat,(Variant)SetDecimalSeparator(f->edtAmount-
>Text).ToDouble(),
```

```
  "Unit_ID",ftString,(Variant)f->cmbUnit->KeyValue,
  "Srok",ftInteger,(Variant)f->edtSrok->Text.ToIntDef(1),
  "Product_ID",ftInteger,Product_ID,
  0);
StartTran(Conn);
try{
      q->ExecSQL();
      CommitTran(Conn);
}
catch(Exception &e){
      RollbackTran(Conn);
      AnsiString s="";
      s="Не удалось изменить запись.\n";
      s=s+e.Message;
      Msg(USER_ERROR,NULL,s.c_str());
}
delete q;
SetCursorPosition(queProduct,"Product_ID",Product_ID);
delete f;
}

//---------------------------------------------------------------------------
// Удалить продукт
//-----------------------------------------
void __fastcall TfrmProduct::mnuDelProdClick(TObject *Sender){
if(queProduct->IsEmpty()){
      // нечего удалять
      return;
}
if(0!=Msg(QUESTION,"Да|Нет","Вы действительно хотите удалить запись?")){
      return;
}

int ID=queProduct->FieldByName("Product_ID")->AsInteger;
int ID1=NearestRecordID(queProduct,"Product_ID");
TADOQuery *q=CreateQuery(DataMod->ADOConnection,
  "delete from PartDish where Product_ID=:Product_ID",
  "Product_ID",ftInteger,(Variant)ID,
  0);
TADOQuery *qq=CreateQuery(DataMod->ADOConnection,
  "delete from Product where Product_ID=:Product_ID",
  "Product_ID",ftInteger,(Variant)ID,
  0);
q->ExecSQL();
```

```
qq->ExecSQL();
delete q;
delete qq;
SetCursorPosition(queProduct,"Product_ID",ID1);
}

//---------------------------------------
// Позиционирует искомый продукт
//---------------------------------------
void __fastcall TfrmProduct::edtFindProdChange(TObject *Sender){
Variant ID;
TLocateOptions opt;

ID=Variant(edtFindProd->Text.c_str());
opt.Clear();
opt<<loPartialKey;
opt<<loCaseInsensitive;
queProduct->Locate("ProductName", ID, opt);
}
//---------------------------------------------------------------------------
```

ProductsWnd.cpp

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "ProductsWnd.h"
#include "DbUnit.h"
#include "Tools.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TfrmProductsWnd *frmProductsWnd;
//---------------------------------------------------------------------------
__fastcall TfrmProductsWnd::TfrmProductsWnd(TComponent* Owner)
    : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TfrmProductsWnd::FormActivate(TObject *Sender)
{
queInsProd->Active=true;
}
//---------------------------------------------------------------------------
```

```
void __fastcall TfrmProductsWnd::InsProdClick(TObject *Sender){
ModalResult=mrOk;
}
//---------------------------------------------------------------------------

void __fastcall TfrmProductsWnd::CancelClick(TObject *Sender){
ModalResult=mrCancel;
}

//---------------------------------------------------------------------------
void __fastcall TfrmProductsWnd::edtAmountKeyPress(TObject *Sender, char
&Key){
if(!(GoodFloatKey(Key) || Key=='+')){
    Key=0;
}
}
//---------------------------------------------------------------------------

void __fastcall TfrmProductsWnd::FormCanResize(TObject *Sender, int
&NewWidth,
    int &NewHeight, bool &Resize)
{
//Resize = False;
}

//---------------------------------------------------------------------------
void __fastcall TfrmProductsWnd::edtSrokKeyPress(TObject *Sender, char &Key){
if(!GoodIntKey(Key)){
    Key=0;
}
}
//---------------------------------------------------------------------------
```

ProdDishUnit.cpp

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "ProdDishUnit.h"
#include "DbUnit.h"
#include "Tools.h"
#include "Msg.h"
#include "DishWnd.h"
#include "PartDishWnd.h"
```

```
extern TADOConnection *Conn;
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TfrmProdDish *frmProdDish;
//---------------------------------------------------------------------------
__fastcall TfrmProdDish::TfrmProdDish(TComponent* Owner)
    : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TfrmProdDish::FormActivate(TObject *Sender){
queDish->Active=true;
quePartDish->Active=true;
}

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

void __fastcall TfrmProdDish::queDishAfterScroll(TDataSet *DataSet)
{
quePartDish->Active=false;
quePartDish->Parameters->ParamByName("Dish_ID")->Value=
  queDish->FieldByName("Dish_ID")->AsInteger;
quePartDish->Active=true;
}
//---------------------------------------------------------------------------

void __fastcall TfrmProdDish::grdProductDrawColumnCell(TObject *Sender,
      const TRect &Rect, int DataCol, TColumn *Column, TGridDrawState State)
{
//
}

//---------------------------------------------------------------------------
// Добавить блюдо
//---------------------------------------------------------------------------
void __fastcall TfrmProdDish::mnuInsDishClick(TObject *Sender){
TfrmDishWnd *f=new TfrmDishWnd(this);
f->Caption="Добавить новое блюдо";
f->edtPrice->Text="";
f->edtDishName->Text="";
f->edtAmountDish->Text="0";
if(f->ShowModal()!=mrOk){
    delete f;
```

```
        return;
    }
    Variant DishName=f->edtDishName->Text;
    Variant Price=f->edtPrice->Text;
    Variant AmountDish=f->edtAmountDish->Text;

    TADOStoredProc *st=CreateStProc(DataMod->ADOConnection, "dbo.insDish",
      "@DishName", ftString,ptInput,DishName,
      "@Price", ftDouble ,ptInput,Price,
      "@AmountDish", ftInteger ,ptInput, AmountDish,
      "@id", ftInteger,ptOutput,(Variant)0,
      0);
    st->ExecProc();
    int ID=int(st->Parameters->ParamValues["@id"]);
    delete st;

    SetCursorPosition(queDish,"Dish_ID",ID);
    delete f;
    }

    //---------------------------------------------------------------------------
    // Добавить продукт в блюдо
    //---------------------------------------------------------------------------
    void __fastcall TfrmProdDish::mnuInsProdDishClick(TObject *Sender)
    {
     TfrmPartDishWnd *f=new TfrmPartDishWnd(this);
     f->Caption="Добавить продукт в блюдо";
    f->queInsPartDish->Active=true;
    f->cmbProduct->KeyValue=0;
    f->edtAmountInDish->Text="0";
    if(f->ShowModal()!=mrOk){
        delete f;
        return;
    }
    AnsiString AmountInDish=f->edtAmountInDish->Text;
    Variant Product_ID=f->cmbProduct->KeyValue;
    Variant Dish_ID=queDish->FieldByName("Dish_ID")->AsInteger;

    TADOStoredProc *st=CreateStProc(DataMod->ADOConnection,
"dbo.insPartDish",
      "@Product_ID", ftInteger,ptInput,Product_ID,
      "@AmountInDish",ftString,ptInput,Variant(AmountInDish),
      "@Dish_ID",ftInteger,ptInput,Dish_ID,
      "@id", ftInteger,ptOutput,(Variant)0,
```

```
  0);
    st->ExecProc();
    int ID=int(st->Parameters->ParamValues["@id"]);
    delete st;

    SetCursorPosition(quePartDish,"Product_ID",ID);
    delete f;

    }

    //---------------------------------------------------------------------------
    // Изменить продукт в составе блюда
    //---------------------------------------------------------------------------
    void __fastcall TfrmProdDish::mnuUpdProdDishClick(TObject *Sender)
    {
    if(quePartDish->IsEmpty()){
        // нечего редактировать
        return;
    }
    TfrmPartDishWnd *f=new TfrmPartDishWnd(this);
    f->Caption="Изменить продукт в блюде";
    int PartDish_ID=quePartDish->FieldByName("PartDish_ID")->AsInteger;
    Variant Dish_ID=queDish->FieldByName("Dish_ID")->AsInteger;
    Variant Product_ID=quePartDish->FieldByName("Product_ID")->Value;
    f->cmbProduct->KeyValue=quePartDish->FieldByName("Product_ID")-
>AsInteger;


    // подставить в окно редактирования текущее значение
    // названия продукта
    f->edtAmountInDish->Text=quePartDish->FieldByName("AmountInDish")-
>AsInteger;
    if(f->ShowModal()!=mrOk){
        delete f;
        return;
    }
     //Product_ID=:Product_ID and Dish_ID=:Dish_ID
    TADOQuery *q=CreateQuery(DataMod->ADOConnection,
     "update PartDish set "
     "  Product_ID=:Product_ID, "
     "  AmountInDish=:AmountInDish,"
     "  Dish_ID=:Dish_ID "
     "where PartDish_ID=:PartDish_ID",
     "Product_ID",ftInteger,f->cmbProduct->KeyValue,
```

```
  "AmountInDish",ftInteger,(Variant)f->edtAmountInDish->Text,
  "Dish_ID",ftInteger,Dish_ID,
  "PartDish_ID",ftInteger,Variant(PartDish_ID),
  0);
q->ExecSQL();
delete q;
SetCursorPosition(quePartDish,"Product_ID",Product_ID);
delete f;
}

//---------------------------------------------------------------------------
// Удалить продукт из состава блюда
//---------------------------------------------------------------------------
void __fastcall TfrmProdDish::mnuDelProdDishClick(TObject *Sender)
{
  if(quePartDish->IsEmpty()){
      // нечего удалять
      return;
}

if(0!=Msg(QUESTION,"Да|Нет","Вы действительно хотите удалить запись?")){
      return;
}
Variant Dish_ID=queDish->FieldByName("Dish_ID")->AsInteger;
int ID=quePartDish->FieldByName("Product_ID")->AsInteger;
int ID1=NearestRecordID(quePartDish,"Product_ID");
TADOQuery *q=CreateQuery(DataMod->ADOConnection,
  "delete from PartDish where Product_ID=:Product_ID and Dish_ID=:Dish_ID",
  "Product_ID",ftInteger,(Variant)ID,
  "Dish_ID",ftInteger,Dish_ID,
  0);
q->ExecSQL();
delete q;
SetCursorPosition(quePartDish,"Product_ID",ID1);
}
//---------------------------------------------------------------------------
// Удалить блюдо
//---------------------------------------------------------------------------
void __fastcall TfrmProdDish::mnuDelDishClick(TObject *Sender){
 if(queDish->IsEmpty()){
      // нечего удалять
      return;
}
if(0!=Msg(QUESTION,"Да|Нет","Вы действительно хотите удалить запись?")){
```

```
    return;
}

int ID=queDish->FieldByName("Dish_ID")->AsInteger;
int ID1=NearestRecordID(queDish,"Dish_ID");
TADOQuery *q=CreateQuery(DataMod->ADOConnection,
  "delete from PartDish where Dish_ID=:Dish_ID",
  "Dish_ID",ftInteger,(Variant)ID,
  0);
TADOQuery *qq=CreateQuery(DataMod->ADOConnection,
  "delete from Dish where Dish_ID=:Dish_ID",
  "Dish_ID",ftInteger,(Variant)ID,
  0);
q->ExecSQL();
qq->ExecSQL();
delete q;
delete qq;
SetCursorPosition(queDish,"Dish_ID",ID1);
}

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

// Позиционирует искомое блюдо
//---------------------------------------------------------------------------
void __fastcall TfrmProdDish::edtFindDishChange(TObject *Sender)
{
Variant ID;
TLocateOptions opt;

ID=Variant(edtFindDish->Text.c_str());
opt.Clear();
opt<<loPartialKey;
opt<<loCaseInsensitive;
queDish->Locate("DishName", ID, opt);
}

//---------------------------------------------------------------------------
// Изменить информацию о блюде
//---------------------------------------------------------------------------
void __fastcall TfrmProdDish::mnuUpdDishClick(TObject *Sender){
if(queDish->IsEmpty()){
    // нечего редактировать
    return;
}
```

```
TfrmDishWnd *f=new TfrmDishWnd(this);
f->Caption="Изменить данные о блюде";
Variant Dish_ID=queDish->FieldByName("Dish_ID")->Value;
// подставить в окно редактирования текущее значение
// названия блюда
Variant Price=queDish->FieldByName("Price")->Value;
Variant AmountDish=queDish->FieldByName("AmountDish")->Value;


f->edtDishName->Text=queDish->FieldByName("DishName")->AsString;
f->edtPrice->Text=queDish->FieldByName("Price")->AsString;
f->edtAmountDish->Text=queDish->FieldByName("AmountDish")->AsString;
if(f->ShowModal()!=mrOk){
     delete f;
     return;
}
TADOQuery *q=CreateQuery(DataMod->ADOConnection,
 "update Dish set "
 "  DishName=:DishName, "
 "  AmountDish=:AmountDish, "
 "  Price=:Price "
 "where Dish_ID=:Dish_ID",
 "DishName",ftString,(Variant)f->edtDishName->Text,
 "AmountDish",ftInteger,(Variant)f->edtAmountDish->Text,
 "Price",ftDouble,(Variant)f->edtPrice->Text,
 "Dish_ID",ftInteger,Dish_ID,
 0);
StartTran(Conn);
try{
     q->ExecSQL();
     CommitTran(Conn);
}
catch(Exception &e){
     RollbackTran(Conn);
     AnsiString s="";
     s="Не удалось изменить запись.\n";
     s=s+e.Message;
     Msg(USER_ERROR,NULL,s.c_str());
}

delete q;
SetCursorPosition(queDish,"Dish_ID",Dish_ID);
delete f;
}
```

DishWnd.cpp

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "DishWnd.h"
#include "DbUnit.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TfrmDishWnd *frmDishWnd;
//---------------------------------------------------------------------------
__fastcall TfrmDishWnd::TfrmDishWnd(TComponent* Owner)
    : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TfrmDishWnd::InsDishClick(TObject *Sender)
{
ModalResult=mrOk;
}
//---------------------------------------------------------------------------
void __fastcall TfrmDishWnd::CancelClick(TObject *Sender)
{
ModalResult=mrCancel;
}
//---------------------------------------------------------------------------
```

PartDishWnd.cpp

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "PartDishWnd.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TfrmPartDishWnd *frmPartDishWnd;
//---------------------------------------------------------------------------
__fastcall TfrmPartDishWnd::TfrmPartDishWnd(TComponent* Owner)
    : TForm(Owner)
{
}
```

```
//---------------------------------------------------------------------------
void __fastcall TfrmPartDishWnd::FormActivate(TObject *Sender){
queInsPartDish->Active=true;
}

//---------------------------------------------------------------------------
void __fastcall TfrmPartDishWnd::InsProdClick(TObject *Sender){
ModalResult=mrOk;
}

//---------------------------------------------------------------------------
void __fastcall TfrmPartDishWnd::CancelClick(TObject *Sender){
ModalResult=mrCancel;
}
//---------------------------------------------------------------------------
```

Complex.cpp

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Complex.h"
#include "DbUnit.h"
#include "Tools.h"
#include "Msg.h"
#include "ComplexWnd.h"
#include "DishWnd.h"
#include "PartComplexWnd.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TfrmComplex *frmComplex;
//---------------------------------------------------------------------------
__fastcall TfrmComplex::TfrmComplex(TComponent* Owner)
    : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TfrmComplex::FormActivate(TObject *Sender)
{
queComplex->Active=true;
quePartComplex->Active=true;
}
```

```
//---------------------------------------------------------------------------
void __fastcall TfrmComplex::queComplexAfterScroll(TDataSet *DataSet){
quePartComplex->Active=false;
quePartComplex->Parameters->ParamByName("Complex_ID")->Value=
  queComplex->FieldByName("Complex_ID")->AsInteger;
quePartComplex->Active=true;
}

//---------------------------------------------------------------------------
// Добавить комплексный обед
//---------------------------------------------------------------------------
void __fastcall TfrmComplex::mnuInsComplexClick(TObject *Sender){
TfrmComplexWnd *f=new TfrmComplexWnd(this);
f->Caption="Добавить комплексный обед";
f->edtComplexName->Text="";
if(f->ShowModal()!=mrOk){
      delete f;
      return;
}
Variant ComplexName=f->edtComplexName->Text;

   TADOStoredProc *st=CreateStProc(DataMod->ADOConnection,
"dbo.insComplex",
     "@ComplexName", ftString,ptInput,ComplexName,
     "@id", ftInteger,ptOutput,(Variant)0,
     0);
   st->ExecProc();
   int ID=int(st->Parameters->ParamValues["@id"]);
   delete st;

   SetCursorPosition(queComplex,"Complex_ID",ID);
   delete f;
   }

   //---------------------------------------------------------------------------
   // Изменить комплексный обед
   //---------------------------------------------------------------------------
   void __fastcall TfrmComplex::mnuUpdComplexClick(TObject *Sender){
   if(queComplex->IsEmpty()){
        // нечего редактировать
        return;
   }
   TfrmComplexWnd *f=new TfrmComplexWnd(this);
```

```
    f->Caption="Изменить комплексный обед";
    Variant Complex_ID=queComplex->FieldByName("Complex_ID")->Value;


    // подставить в окно редактирования текущее значение
    // названия единицы измерения
    f->edtComplexName->Text=queComplex->FieldByName("ComplexName")-
>AsString;
    if(f->ShowModal()!=mrOk){
        delete f;
        return;
    }
    TADOQuery *q=CreateQuery(DataMod->ADOConnection,
     "update Complex set "
     "  ComplexName=:ComplexName "
     "where Complex_ID=:Complex_ID",
     "ComplexName",ftString,(Variant)f->edtComplexName->Text,
     "Complex_ID",ftInteger,Complex_ID,
     0);
    q->ExecSQL();
    delete q;
    SetCursorPosition(queComplex,"Complex_ID",Complex_ID);
    delete f;
    }

    //---------------------------------------------------------------------------
    // Удалить комплексный обед
    //---------------------------------------------------------------------------
    void __fastcall TfrmComplex::mnuDelComplexClick(TObject *Sender){
    if(queComplex->IsEmpty()){
        // нечего удалять
        return;
    }
    if(0!=Msg(QUESTION,"Да|Нет","Вы действительно хотите удалить запись?")){
        return;
    }

    int ID=queComplex->FieldByName("Complex_ID")->AsInteger;
    int ID1=NearestRecordID(queComplex,"Complex_ID");
    TADOQuery *q=CreateQuery(DataMod->ADOConnection,
     "delete from Complex where Complex_ID=:Complex_ID",
     "Complex_ID",ftInteger,(Variant)ID,
     0);
    q->ExecSQL();
```

```
    delete q;
    SetCursorPosition(queComplex,"Complex_ID",ID1);
    }

    //---------------------------------------------------------------------------
    // Добавить блюдо в комплексный обед
    //---------------------------------------------------------------------------
    void __fastcall TfrmComplex::mnuInsPartComplexClick(TObject *Sender)
    {
    TfrmPartComplexWnd *f=new TfrmPartComplexWnd(this);
    f->queInsPartComplex->Active=true;
    f->edtAmountInComplex->Text="1";
    f->cmbDish->KeyValue=0;
    f->Caption="Добавить блюдо в комплексный обед";
    if(f->ShowModal()!=mrOk){
        delete f;
        return;
    }

    Variant Dish_ID=f->cmbDish->KeyValue;
    Variant Complex_ID=queComplex->FieldByName("Complex_ID")->AsInteger;
    Variant AmountInComplex=f->edtAmountInComplex->Text;

    TADOStoredProc *st=CreateStProc(DataMod->ADOConnection,
"dbo.insPartComplex",
      "@Complex_ID", ftInteger,ptInput,Complex_ID,
      "@Dish_ID", ftInteger,ptInput,Dish_ID,
      "@AmountInComplex", ftInteger ,ptInput, AmountInComplex,
      "@id", ftInteger,ptOutput,(Variant)0,
      0);
    st->ExecProc();
    int ID=int(st->Parameters->ParamValues["@id"]);
    delete st;

    SetCursorPosition(quePartComplex,"Dish_ID",ID);
    delete f;
    }
    //---------------------------------------------------------------------------
    // Удалить блюдо из комплексного обеда
    //---------------------------------------------------------------------------
    void __fastcall TfrmComplex::mnuDelPartComplexClick(TObject *Sender)
    {
      if(quePartComplex->IsEmpty()){
        // нечего удалять
```

```
        return;
    }

    if(0!=Msg(QUESTION,"Да|Нет","Вы действительно хотите удалить запись?")){
        return;
    }
    Variant Complex_ID=queComplex->FieldByName("Complex_ID")->AsInteger;
    int ID=quePartComplex->FieldByName("Dish_ID")->AsInteger;
    int ID1=NearestRecordID(quePartComplex,"Dish_ID");
    TADOQuery *q=CreateQuery(DataMod->ADOConnection,
      "delete from PartComplex where Dish_ID=:Dish_ID and
Complex_ID=:Complex_ID",
      "Dish_ID",ftInteger,(Variant)ID,
      "Complex_ID",ftInteger,Complex_ID,
      0);
    q->ExecSQL();
    delete q;
    SetCursorPosition(quePartComplex,"Dish_ID",ID1);
    }

    //---------------------------------------------------------------------------
    // Изменить блюдо в комплексном обеде
    //---------------------------------------------------------------------------
    void __fastcall TfrmComplex::mnuUpdPartComplexClick(TObject *Sender)
    {
    if(quePartComplex->IsEmpty()){
        // нечего редактировать
        return;
    }
    TfrmPartComplexWnd *f=new TfrmPartComplexWnd(this);
    f->Caption="Изменить блюдо в комплексном обеде";
    int PartComplex_ID=quePartComplex->FieldByName("PartComplex_ID")-
>AsInteger;
    Variant Complex_ID=queComplex->FieldByName("Complex_ID")->AsInteger;
    Variant Dish_ID=quePartComplex->FieldByName("Dish_ID")->Value;
    f->cmbDish->KeyValue=quePartComplex->FieldByName("Dish_ID")->AsInteger;

    // подставить в окно редактирования текущее значение
    // названия продукта
    f->edtAmountInComplex->Text=quePartComplex-
>FieldByName("AmountInComplex")->AsInteger;
    if(f->ShowModal()!=mrOk){
        delete f;
        return;
```

```
}
TADOQuery *q=CreateQuery(DataMod->ADOConnection,
  "update PartComplex set "
  "  Dish_ID=:Dish_ID, "
  "  AmountInComplex=:AmountInComplex, "
  "  Complex_ID=:Complex_ID "
  "where PartComplex_ID=:PartComplex_ID",
  "Dish_ID",ftInteger,f->cmbDish->KeyValue,
  "AmountInComplex",ftInteger,(Variant)f->edtAmountInComplex->Text,
  "Complex_ID",ftInteger,Complex_ID,
  "PartComplex_ID",ftInteger,Variant(PartComplex_ID),
  0);
q->ExecSQL();
delete q;
SetCursorPosition(quePartComplex,"Dish_ID",Dish_ID);
delete f;
}
//---------------------------------------------------------------------------
```

ComplexWnd.cpp

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "ComplexWnd.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TfrmComplexWnd *frmComplexWnd;
//---------------------------------------------------------------------------
__fastcall TfrmComplexWnd::TfrmComplexWnd(TComponent* Owner)
      : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TfrmComplexWnd::InsDishClick(TObject *Sender)
{
ModalResult=mrOk;
}
//---------------------------------------------------------------------------

void __fastcall TfrmComplexWnd::CancelClick(TObject *Sender)
{
ModalResult=mrCancel;
```

```
}
//---------------------------------------------------------------------------
```
PartComplexWnd.cpp
```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "PartComplexWnd.h"
#include "DbUnit.h"
#include "Tools.h"
#include "Msg.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TfrmPartComplexWnd *frmPartComplexWnd;
//---------------------------------------------------------------------------
__fastcall TfrmPartComplexWnd::TfrmPartComplexWnd(TComponent* Owner)
    : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TfrmPartComplexWnd::FormActivate(TObject *Sender)
{
queInsPartComplex->Active=true;
}
//---------------------------------------------------------------------------

void __fastcall TfrmPartComplexWnd::InsProdClick(TObject *Sender)
{
ModalResult=mrOk;
}
//---------------------------------------------------------------------------

void __fastcall TfrmPartComplexWnd::CancelClick(TObject *Sender)
{
ModalResult=mrCancel;
}
//---------------------------------------------------------------------------
```
TypeExcurs.cpp
```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop
```

```
#include "TypeExcurs.h"
#include "DbUnit.h"
#include "Tools.h"
#include "Msg.h"
#include "TypeExcursWnd.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TfrmTypeExcurs *frmTypeExcurs;

TDateTime df;
//---------------------------------------------------------------------------
__fastcall TfrmTypeExcurs::TfrmTypeExcurs(TComponent* Owner)
    : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TfrmTypeExcurs::FormActivate(TObject *Sender)
{
queTypeExcurs->Active=true;
}
//---------------------------------------------------------------------------
void __fastcall TfrmTypeExcurs::mnuInsTypeExcursClick(TObject *Sender){
TfrmTypeExcursWnd *f=new TfrmTypeExcursWnd(this);
f->Caption="Добавить тип экскурсии";
f->queInsTypeExcurs->Active=true;
f->edtTypeExcursName->Text="";
f->edtTypeExcursPrice->Text="";

if(f->ShowModal()!=mrOk){
    delete f;
    return;
}
Variant TypeExcursName=f->edtTypeExcursName->Text;
Variant TypeExcursPrice=f->edtTypeExcursPrice->Text;
Variant StartTime=f->dtpStartTime->DateTime;
Variant FinishTime=f->dtpFinishTime->DateTime;

TADOStoredProc *st=CreateStProc(DataMod->ADOConnection,
"dbo.insTypeExcurs",
  "@TypeExcursName", ftString,ptInput,TypeExcursName,
  "@TypeExcursPrice", ftString,ptInput,TypeExcursPrice,
  "@StartTime", ftDateTime,ptInput,StartTime,
```

```
  "@FinishTime", ftDateTime,ptInput,FinishTime,
  "@id", ftInteger,ptOutput,(Variant)0,
  0);
st->ExecProc();
int ID=int(st->Parameters->ParamValues["@id"]);
delete st;

SetCursorPosition(queTypeExcurs,"TypeExcurs_ID",ID);
delete f;
}

//---------------------------------------------------------------------------
// Удалить тип экскурсии
//---------------------------------------------------------------------------
void __fastcall TfrmTypeExcurs::mnuDelTypeExcursClick(TObject *Sender){
if(queTypeExcurs->IsEmpty()){
    // нечего удалять
    return;
}
if(0!=Msg(QUESTION,"Да|Нет","Вы действительно хотите удалить запись?")){
    return;
}

int ID=queTypeExcurs->FieldByName("TypeExcurs_ID")->AsInteger;
int ID1=NearestRecordID(queTypeExcurs,"TypeExcurs_ID");
TADOQuery *q=CreateQuery(DataMod->ADOConnection,
  "delete from TypeExcurs where TypeExcurs_ID=:TypeExcurs_ID",
  "TypeExcurs_ID",ftInteger,(Variant)ID,
  0);
q->ExecSQL();
delete q;
SetCursorPosition(queTypeExcurs,"TypeExcurs_ID",ID1);
}
//---------------------------------------------------------------------------

void __fastcall TfrmTypeExcurs::mnuUpdTypeExcursClick(TObject *Sender){
if(queTypeExcurs->IsEmpty()){
    // нечего редактировать
    return;
}
TfrmTypeExcursWnd *f=new TfrmTypeExcursWnd(this);
f->Caption="Изменить тип экскурсии";
int TypeExcurs_ID=queTypeExcurs->FieldByName("TypeExcurs_ID")-
>AsInteger;
```

```
    f->edtTypeExcursName->Text=queTypeExcurs-
>FieldByName("TypeExcursName")->AsString;
    f->edtTypeExcursPrice->Text=queTypeExcurs-
>FieldByName("TypeExcursPrice")->AsString;
    f->dtpStartTime->DateTime=queTypeExcurs->FieldByName("StartTime")-
>AsDateTime;
    f->dtpFinishTime->DateTime=queTypeExcurs->FieldByName("FinishTime")-
>AsDateTime;

    // подставить в окно редактирования текущее значение
    // названия продукта

    if(f->ShowModal()!=mrOk){
        delete f;
        return;
    }
    TADOQuery *q=CreateQuery(DataMod->ADOConnection,
     "update TypeExcurs set "
     "  TypeExcursName=:TypeExcursName, "
     "  TypeExcursPrice=:TypeExcursPrice, "
     "  StartTime=:StartTime, "
     "  FinishTime=:FinishTime "
     "where TypeExcurs_ID=:TypeExcurs_ID",
     "TypeExcursName",ftString,(Variant)f->edtTypeExcursName->Text,
     "TypeExcursPrice",ftString,(Variant)f->edtTypeExcursPrice->Text,
     "StartTime",ftDateTime,(Variant)f->dtpStartTime->DateTime,
     "FinishTime",ftDateTime,(Variant)f->dtpFinishTime->DateTime,
     "TypeExcurs_ID",ftInteger,Variant(TypeExcurs_ID),
     0);
    q->ExecSQL();
    delete q;
    delete f;
    SetCursorPosition(queTypeExcurs,"TypeExcurs_ID",TypeExcurs_ID);
    }
    //---------------------------------------------------------------------------
```

TypeExcursWnd.cpp

```
    //---------------------------------------------------------------------------

    #include <vcl.h>
    #pragma hdrstop

    #include "TypeExcursWnd.h"
    #include "DbUnit.h"
    //---------------------------------------------------------------------------
```

```
#pragma package(smart_init)
#pragma resource "*.dfm"
TfrmTypeExcursWnd *frmTypeExcursWnd;
//---------------------------------------------------------------------------
__fastcall TfrmTypeExcursWnd::TfrmTypeExcursWnd(TComponent* Owner)
    : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TfrmTypeExcursWnd::FormActivate(TObject *Sender)
{
queInsTypeExcurs->Active=true;
}
//---------------------------------------------------------------------------
void __fastcall TfrmTypeExcursWnd::InsProdClick(TObject *Sender)
{
ModalResult=mrOk;
}
//---------------------------------------------------------------------------

void __fastcall TfrmTypeExcursWnd::CancelClick(TObject *Sender)
{
ModalResult=mrCancel;
}
//---------------------------------------------------------------------------
```

Units.cpp

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Units.h"
#include "DbUnit.h"
#include "UnitsWnd.h"
#include "Tools.h"
#include "Msg.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TfrmUnits *frmUnits;
//---------------------------------------------------------------------------
__fastcall TfrmUnits::TfrmUnits(TComponent* Owner)
    : TForm(Owner)
```

```
{
}
//---------------------------------------------------------------------------
void __fastcall TfrmUnits::FormActivate(TObject *Sender)
{
queUnit->Active=true;
}
//---------------------------------------------------------------------------

void __fastcall TfrmUnits::mnuInsUnitClick(TObject *Sender)
{
TfrmUnitsWnd *f=new TfrmUnitsWnd(this);
f->Caption="Добавить единицу измерения";
f->queInsUnit->Active=true;
f->edtUnitName->Text="";
f->edtFullUnitName->Text="";
if(f->ShowModal()!=mrOk){
      delete f;
      return;
}
Variant UnitName=f->edtUnitName->Text;
Variant FullUnitName=f->edtFullUnitName->Text;

TADOStoredProc *st=CreateStProc(DataMod->ADOConnection, "dbo.insUnit",
  "@UnitName", ftString,ptInput,UnitName,
  "@FullUnitName", ftString,ptInput,FullUnitName,
  "@id", ftInteger,ptOutput,(Variant)0,
  0);
st->ExecProc();
int ID=int(st->Parameters->ParamValues["@id"]);
delete st;

SetCursorPosition(queUnit,"Unit_ID",ID);
delete f;
}
//---------------------------------------------------------------------------

void __fastcall TfrmUnits::mnuDelUnitClick(TObject *Sender)
{
if(queUnit->IsEmpty()){
      // нечего удалять
      return;
}
if(0!=Msg(QUESTION,"Да|Нет","Вы действительно хотите удалить запись?")){
```

```
    return;
}

int ID=queUnit->FieldByName("Unit_ID")->AsInteger;
int ID1=NearestRecordID(queUnit,"Unit_ID");
TADOQuery *q=CreateQuery(DataMod->ADOConnection,
  "delete from Unit where Unit_ID=:Unit_ID",
  "Unit_ID",ftInteger,(Variant)ID,
  0);
q->ExecSQL();
delete q;
SetCursorPosition(queUnit,"Unit_ID",ID1);
}
//---------------------------------------------------------------------------

void __fastcall TfrmUnits::mnuUpdUnitClick(TObject *Sender)
{
 if(queUnit->IsEmpty()){
    // нечего редактировать
    return;
}
TfrmUnitsWnd *f=new TfrmUnitsWnd(this);
f->Caption="Изменение единицы измерения";
Variant Unit_ID=queUnit->FieldByName("Unit_ID")->Value;
// подставить в окно редактирования текущее значение
// названия единицы измерения
f->edtUnitName->Text=queUnit->FieldByName("UnitName")->AsString;
f->edtFullUnitName->Text=queUnit->FieldByName("FullUnitName")->AsString;
if(f->ShowModal()!=mrOk){
    delete f;
    return;
}
TADOQuery *q=CreateQuery(DataMod->ADOConnection,
  "update Unit set "
  "  UnitName=:UnitName, "
  "  FullUnitName=:FullUnitName "
  "where Unit_ID=:Unit_ID",
  "UnitName",ftString,(Variant)f->edtUnitName->Text,
  "FullUnitName",ftString,(Variant)f->edtFullUnitName->Text,
  "Unit_ID",ftInteger,Unit_ID,
  0);
q->ExecSQL();
delete q;
SetCursorPosition(queUnit,"Unit_ID",Unit_ID);
```

```
delete f;
}
//---------------------------------------------------------------------------
```
UnitsWnd.cpp
```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "UnitsWnd.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TfrmUnitsWnd *frmUnitsWnd;
//---------------------------------------------------------------------------
__fastcall TfrmUnitsWnd::TfrmUnitsWnd(TComponent* Owner)
    : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TfrmUnitsWnd::FormActivate(TObject *Sender)
{
queInsUnit->Active=true;
}
//---------------------------------------------------------------------------
void __fastcall TfrmUnitsWnd::InsProdClick(TObject *Sender)
{
ModalResult=mrOk;
}
//---------------------------------------------------------------------------
void __fastcall TfrmUnitsWnd::CancelClick(TObject *Sender)
{
ModalResult=mrCancel;
}
//---------------------------------------------------------------------------
```
ExcursWnd.cpp
```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "ExcursWnd.h"
#include "DbUnit.h"
#include "Msg.h"
```

```
#include "Tools.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TfrmExcursWnd *frmExcursWnd;

//---------------------------------------------------------------------------
__fastcall TfrmExcursWnd::TfrmExcursWnd(TComponent* Owner)
    : TForm(Owner)
{
}

//---------------------------------------------------------------------------
void __fastcall TfrmExcursWnd::FormActivate(TObject *Sender){
//
}

//---------------------------------------------------------------------------
void __fastcall TfrmExcursWnd::InsProdClick(TObject *Sender){
ModalResult=mrOk;
}

//---------------------------------------------------------------------------
void __fastcall TfrmExcursWnd::CancelClick(TObject *Sender){
ModalResult=mrCancel;
}
//---------------------------------------------------------------------------
```

VisitorWnd.cpp

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "VisitorWnd.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TfrmVisitorWnd *frmVisitorWnd;
//---------------------------------------------------------------------------
__fastcall TfrmVisitorWnd::TfrmVisitorWnd(TComponent* Owner)
    : TForm(Owner)
{
}
//---------------------------------------------------------------------------
```

```
void __fastcall TfrmVisitorWnd::InsProdClick(TObject *Sender)
{
ModalResult=mrOk;
}
//---------------------------------------------------------------------------
void __fastcall TfrmVisitorWnd::CancelClick(TObject *Sender)
{
ModalResult=mrCancel;
}
//---------------------------------------------------------------------------
void __fastcall TfrmVisitorWnd::FormActivate(TObject *Sender)
{
queInsVisitor->Active=true;
}
//---------------------------------------------------------------------------
```

EdaWnd.cpp

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "EdaWnd.h"
#include "Msg.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TfrmEdaWnd *frmEdaWnd;
//---------------------------------------------------------------------------
__fastcall TfrmEdaWnd::TfrmEdaWnd(TComponent* Owner)
    : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TfrmEdaWnd::InsProdClick(TObject *Sender){
if(edtAmountEda->Text.IsEmpty()){
    Msg(USER_ERROR,NULL,"Вы не указали количество");
    return;
}
ModalResult=mrOk;
}
//---------------------------------------------------------------------------
void __fastcall TfrmEdaWnd::CancelClick(TObject *Sender)
{
ModalResult=mrCancel;
```

```
}
//---------------------------------------------------------------------------
void __fastcall TfrmEdaWnd::FormActivate(TObject *Sender)
{
queInsEda->Active=true;
}
//---------------------------------------------------------------------------
```

Report.cpp

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Report.h"
#include "DbUnit.h"
#include "Tools.h"
#include "Excel.h"
#include "excel_2k.h"
#include "Msg.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TfrmReport *frmReport;
//---------------------------------------------------------------------------
__fastcall TfrmReport::TfrmReport(TComponent* Owner)
    : TForm(Owner)
{
}

//---------------------------------------------------------------------------
void __fastcall TfrmReport::FormActivate(TObject *Sender){
dtp1->Date=RoundDateTime(ServerDate(DataMod->ADOConnection)+10);
}

//---------------------------------------------------------------------------
struct COL{
    char *Header; // заголовок
    char *FieldName;
    char *Format;
    int Width; // ширина поля
    char Type; // тип поля 'c'-строка 'f'-float
    int Align; // выравнивание
};
```

```
static COL Col[]={
     {"Наименование продукта","ProductName",NULL,36,'c',xlLeft},
     {"Ед. изм.","UnitName",NULL,7,'c',xlCenter},
     {"В наличии","WeHave","0,00",10,'f',xlRight},
     {"Закупить","ToBuy","0.00",10,'f',xlRight}
};

#define FIRSTROW 2
#define FIRSTCOL 2
//-----------------------------------
void __fastcall TfrmReport::ToExcelClick(TObject *Sender){
char buf[30];
Variant d;
AnsiString s,FieldName;
int k,row,col,nCol,i;
double f;

EXCEL_APP *xl=new EXCEL_APP;
d=dtp1->DateTime;

AnsiString Sql="select ",Comma="";
for(i=0;i<sizeof(Col)/sizeof(COL);i++){
     Sql=Sql+Comma+Col[i].FieldName;
     Comma=",";
}
Sql=Sql+" from dbo.PlanProd(:d) order by ProductName";
TADOQuery *q=CreateQuery(DataMod->ADOConnection,Sql,
  "d",ftDateTime,d,
  NULL);

/* шапка */
xl->Show();
nCol=sizeof(Col)/sizeof(COL),i;
xl->MergeCells(FIRSTROW-1,FIRSTCOL,FIRSTROW-1,FIRSTCOL+nCol-1);
s="Потребность в продуктах на период до ";
s=s+dtp1->DateTime.DateString();
xl->PutVal(FIRSTROW-1,FIRSTCOL,s);
xl->FontNameAndSize("Tahoma",12);
xl->FontBold(true);
xl->SelectRange(FIRSTROW-1,FIRSTCOL,FIRSTROW-1,FIRSTCOL);
xl->HorAlign(xlCenter);

for(i=0;i<nCol;i++){
     xl->PutVal(FIRSTROW,FIRSTCOL+i,Col[i].Header);
```

```
        xl->SelectRange(FIRSTROW,FIRSTCOL+i,FIRSTROW,FIRSTCOL+i);
        xl->SetColumnWidth(Col[i].Width);
        xl->HorAlign(xlCenter);
        xl->FontBold(true);
    }
    q->Active=true;
    for(k=0,q->First();!q->Eof;k++,q->Next()){
        row=FIRSTROW+k+1;
        for(i=0;i<nCol;i++){
            col=FIRSTCOL+i;
            switch(Col[i].Type){
                case 'c':
                    s=q->FieldByName(Col[i].FieldName)->AsString;
                    break;
                case 'f':
                    f=q->FieldByName(Col[i].FieldName)->AsFloat;
                    sprintf(buf,"%12.1lf",f);
                    s=SetDecimalSeparator(buf).Trim();
                    break;
                default:
                    Msg(PROGRAMMER,NULL,"Непредусмотренный тип в
TfrmReport::ToExcelClick");
                    delete xl;
                    delete q;
                    return;
            }
            xl->PutVal(row,col,s);
            xl->SelectRange(row,col,row,col);
            xl->HorAlign(Col[i].Align);
        } // for i
    }
    xl->SelectRange(FIRSTROW,FIRSTCOL,FIRSTROW+k,FIRSTCOL+nCol-1);
    xl->PutSetka();
    delete q;
    delete xl;
    }

    //---------------------------------------------------------------------------
```

Spravka.cpp

```
    //---------------------------------------------------------------------------
    #include <vcl.h>
    #pragma hdrstop

    #include "Spravka.h"
```

```
#include "DbUnit.h"
#include "MsgUnit.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TfrmSpravka *frmSpravka;
//---------------------------------------------------------------------------
__fastcall TfrmSpravka::TfrmSpravka(TComponent* Owner)
      : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TfrmSpravka::InsProdClick(TObject *Sender)
{
ModalResult=mrOk;
}
//---------------------------------------------------------------------------
```