

ЛАБОРАТОРНАЯ РАБОТА №2

СОЗДАНИЕ ПРОСТОГО ПРИЛОЖЕНИЯ ДЛЯ РАБОТЫ С БАЗАМИ ДАННЫХ В C++BUILDER

Цель:

- 1) Изучение основных компонент для работы с данными в базах данных;
- 2) Создание простого приложения для работы с базой данных для своей предметной области (на примере базы данных для кафе: Кафе.mdb). Прodelать задания 1-5.

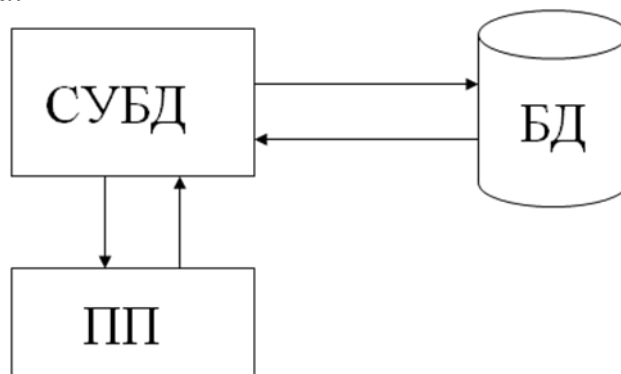
Краткие теоретические сведения

Программирование баз данных является одним из наиболее приоритетных и востребованных направлений в сфере разработки программного обеспечения.

Существует множество, как самих сред создания приложений, так и универсальных СУБД со встроенным языком (зачастую высокого уровня). Например, среды Borland Delphi, Microsoft Visual Studio, C++ Builder и др., СУБД со встроенными языками Visual FoxPro, Microsoft Access и другие.

Каждый из таких продуктов содержит широкий набор компонентов, методов и процедур для обработки сложнейших структурированных систем данных.

В C++Builder предусмотрена возможность разработки приложений баз данных (БД) для различных систем управления БД (СУБД). При этом реализуется цепочка:



где ПП – прикладная программа (приложение)

Таким образом, в C++Builder для доступа к базам данных из приложения используется цепочка «Приложение (ПП) -> СУБД -> База данных». Это означает, что при любом обращении к БД из приложения реально адресуется СУБД. СУБД, используя собственные функции, связывается непосредственно с базой данных.

Для работы с конкретной базой данных при этом необходимо знать:

- где БД физически расположена;
- параметры этой БД;
- общие параметры драйвера БД того типа, к которому принадлежит обрабатываемая БД;
- общие системные установки.

Таким образом, первоначально необходимо решить проблему подключения, разрабатываемой прикладной программы (ПП) к выбранной СУБД.

В зависимости от выбора СУБД для доступа к данным БД в С++Builder могут применяться различные компоненты с различных закладок. Например, закладки палитры компонент:

- **BDE** – (Borland Database Engine) - представляет собой библиотеку функций специализированного программного интерфейса (API) и набор драйверов для работы с множеством «популярных» СУБД. Должна устанавливаться на каждом компьютере, который использует приложения для работы с БД, написанные на С++Builder с использованием компонент, работающих с BDE. BDE может использоваться и независимо от С++Builder (например, прямой вызов API функций из любых других программ). BDE представляет собой набор динамических библиотек, которые "умеют" передавать запросы на получение или модификацию данных из приложения в нужную базу данных и возвращать результат обработки. В процессе работы библиотеки используют вспомогательные файлы языковой поддержки и информацию о настройках среды. В составе BDE поставляются стандартные драйверы, обеспечивающие доступ к СУБД Paradox, dBASE, FoxPro, система драйверов — SQL Links, с их помощью можно разрабатывать приложения для серверов Oracle, Informix, Sybase, DB2, InterBase. Эти драйверы необходимо устанавливать дополнительно. Помимо этого, в BDE имеется очень простой механизм подключения различных драйверов ODBC (к примеру, Microsoft Access). Кроме этого имеется шесть дополнительных DLL, обеспечивающих работу BDE с серверами Oracle и Microsoft SQL Server. Для BDE требуется отдельная установка, которая занимает порядка 15 Мбайт дискового пространства, а также специальная настройка псевдонимов.

- **InterBase** - для прямого доступа к СУБД InterBase. В этом случае можно не устанавливать BDE на компьютере, где будет запускаться конечное приложение. На странице InterBase Палитры компонентов содержатся компоненты доступа к данным, адаптированные для работы с сервером InterBase и объединенные названием InterBase Express. Компоненты из набора InterBase Express предназначены для работы с сервером InterBase версии не ниже 5.5. Их преимущество заключается в реализации всех функций за счет прямого обращения к API сервера InterBase. Благодаря этому существенно повышается скорость работы компонентов.

- **ADO** – (ActiveX Data Object) – поддержка технологии Microsoft ActiveX Data Objects (ADO), которая основана на возможностях COM, а

именно интерфейсов OLE DB. Технология ADO завоевала популярность у разработчиков, благодаря универсальности — базовый набор интерфейсов OLE DB имеется в каждой современной операционной системе Microsoft. Поэтому для обеспечения доступа приложения к данным достаточно лишь правильно указать провайдер соединения ADO и затем переносить программу на любой компьютер, где имеется требуемая база данных и, конечно, установленная ADO. В Палитре компонентов страница ADO содержит набор компонентов, позволяющих создавать полноценные приложения БД, обращающиеся к данным через ADO. Технология ADO и интерфейсы OLE DB обеспечивают для приложений единый способ доступа к источникам данных различных типов. Например, приложение, использующее ADO, может применять одинаково сложные операции и к данным, хранящимся на корпоративном сервере, к электронным таблицам и локальным СУБД. OLE DB представляет собой набор специализированных объектов COM, инкапсулирующих стандартные функции обработки данных, и специализированные функции конкретных источников данных и интерфейсов, обеспечивающих передачу данных между объектами. Согласно терминологии ADO, любой источник данных (база данных, электронная таблица, файл) называется хранилищем данных, с которым при помощи провайдера данных взаимодействует приложение. Технология ADO в целом включает в себя не только сами объекты OLE DB, но и механизмы, обеспечивающие взаимодействие объектов с данными и приложениями. На этом уровне важнейшую роль играют провайдеры ADO, координирующие работу приложений с хранилищами данных различных типов. Так как технология ADO основана на стандартных интерфейсах COM, которые являются системным механизмом Windows, это сокращает общий объем работающего программного кода и позволяет распространять приложения БД без вспомогательных программ и библиотек. ADO предустановлена в операционной системе, но нуждается в настраиваемых провайдерах данных.

- **dbExpress** - предоставляет кроссплатформенную поддержку как для Delphi и C++ Builder под Windows, так и для Kylix под Linux. Поставляется с драйверами dbExpress для работы с DB2, Informix, InterBase, MS SQL Server, MySQL и Oracle. Для создания бесплатных приложений подходят только MySQL и FireBird (InterBase-совместимая СУБД, распространяемая по лицензии Open Source). Драйверы реализованы в виде динамических библиотек, а при необходимости могут быть прикомпилированы непосредственно к исполняемому файлу приложения. Поэтому проблема распространения совместно с приложением средств доступа к данным в случае с dbExpress снимается полностью. Естественно, на компьютере должно быть установлено клиентское ПО соответствующего SQL сервера. Компоненты dbExpress располагаются в Палитре компонентов на одноименной странице. Технология dbExpress обеспечивает доступ к серверу баз данных при помощи драйвера, реализованного как динамическая библиотека. Для каждого сервера имеется своя динамическая библиотека.

- Также существует множество компонент для C++Builder «третьих» производителей для доступа к «популярным» СУБД (Oracle, dBase, MS SQL и др.), которые осуществляют прямой доступ к «своей» СУБД.

Задание 1. Создание приложения для работы с данными кафе

- Создайте на диске отдельную папку. Например, D:\Temp\Petrov\Proga_Caffe.
- Переместите в нее файл базы данных, созданной в лабораторной работе №1, - файл Кафе.mdb.
- Запустите C++Builder.
- Выберите в меню File/New/Application.
- Свойство Name у формы установите через Object Inspector в fmMain.
- Свойство Caption у формы установите через Object Inspector в Программа для кафе.
- Сохраните проект через меню File/Save Project As или File/Save all. В появившемся диалоговом окне сохраните файл модуля (юнита) проекта с именем Main.cpp в созданную ранее папку (там уже есть файл БД). В эту же папку сохраните следом файл проекта с именем Caffe.bpr. Откомпилированный проект будет иметь имя Caffe.exe.

Сейчас наш проект состоит из одного модуля Main и одной главной формы fmMain. Затем наш проект будет состоять из нескольких форм.

Задание 2. Подготовка главной формы fmMain

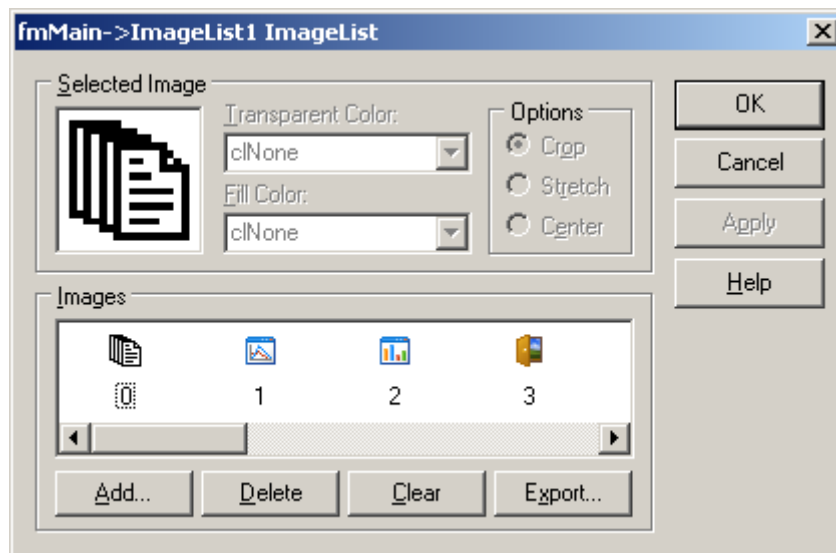
Данная форма отображается после запуска программы (Caffe.exe).

В настоящее время пользователи уже привыкли к меню с пиктограммами, характеризующими соответствующие команды, панелям инструментов, спискам с элементами, имеющими индивидуальные пиктограммы, и т.д.

Задание 2.1 Создание списка изображений (пиктограмм)

Для хранения списка пиктограмм используется компонент типа TImageList (список изображений), позволяющий хранить внутри себя изображения одинакового размера, на которые можно ссылаться по индексам, начинающимся с нуля. Стандартным размером является 16x16 (можно изменять). TImageList является невизуальным компонентом при запуске приложения.

После помещения его на форму с закладки Win32 необходимо дважды щелкнуть на компоненте, при этом откроется редактор списка картинок, с помощью которого можно добавить в список новые картинки или удалить существующие.



Использование списка картинок TImageList вместо отдельных изображений позволяет значительно экономить ресурсы Windows, т.к. для списка на самом деле хранится одна большая картинка, имеющая в качестве составных частей все необходимые картинки. Особенно это заметно при использовании больших наборов изображений.

В нашем приложении пиктограммы будут назначаться пунктам меню, кнопкам панелей инструментов.

- Разместите на форме с палитры Win32 компонент ImageList1.
- Нажмите левой кнопкой мыши на ImageList1 два раза. Появится окно редактора списка изображений.
- В появившемся диалоговом окне нажмите Add и выберите понравившуюся пиктограмму для действия по закрытию нашей программы. Нажмите Ок.
- Аналогично добавьте в список пиктограмм ImageList1 пиктограммы для действий, связанных с сотрудниками, блюдами, продажами, информации об авторах программы, запросах.

Для того, чтобы можно было назначить картинку любому меню, кнопке и пр. компонентам, необходимо вначале установить в свойстве Images компонента необходимый компонент типа TImageList, а затем в свойстве ImageIndex каждого элемента указать номер необходимой картинки.

Задание 2.2 Создание главного меню

Для создания главного меню используется компонент типа TMainMenu с закладки Standart. Это не визуальный компонент при запуске приложения. При запуске приложения пользователи видят не сам компонент, а меню, сгенерированное им.

Для этого необходимо поместить его на форму и дважды щелкнуть по нему. После того на экране появится окно редактора меню.

Сначала меню состоит из одного пустого элемента. Для добавления нового пункта меню можно просто начать изменять любой пустой *i* пункт меню в редакторе меню (он обведен пунктирным прямоугольником). Для вставки нового пункта меню в любой позиции необходимо нажать клавишу Insert. Для удаления пункта меню -клавишу Delete. Для создания подменю второго уровня — клавишу Ctrl+—>.

- Разместите на форме с палитры Standart компонент MainMenu1.
 - Свойство Images у MainMenu1 установите в ImageList1 – для того, чтобы подключить список пиктограмм к меню.
 - Нажмите левой кнопкой мыши на MainMenu1 два раза. Появится редактор меню.
 - В редакторе меню MainMenu1, выделив верхний левый угол, напишите в свойстве Caption (надпись) Файл.
 - Справа от раздела Файл напишите в свойстве Caption Данные.
 - Справа от раздела Данные напишите в свойстве Caption Документы.
 - Справа от раздела Документы напишите в свойстве Caption Запросы.
 - Ниже в блоке под Файл напишите в свойстве Caption Закреть программу.
 - Для пункта Закреть программу (выделите его левой кнопкой мыши) установите через Object Inspector:
 - свойство ImageIndex из выпадающего списка в код картинки для закрытия программы в связанном списке картинок;
 - свойство Shortcut – горячая клавиша – запишите Alt+X;
 - событие OnClick (закладка Events или два раза левой кнопкой мыши нажать на пункте Закреть программу): запишите метод Close() – закрыть.
 - Нажмите F9.
 - Проверьте работу пункта меню по закрытию программы.
- Заполнение меню продолжим позже.

Задание 2.3 Создание контекстного (локального, всплывающего) меню

Контекстное (локальное, всплывающее) меню обычно появляется на экране при нажатии правой кнопки мышки на визуальных компонентах. Для этого у этих компонентов должно быть установлено свойство PopUpMenu в инспекторе объектов из выпадающего списка доступных локальных меню TPopUpMenu.

Для редактирования состава локального меню необходимо, как и для главного меню формы, дважды щелкнуть на компоненте TPopUpMenu, при этом появится редактор меню, работа в котором аналогична, как и для TMainMenu.

- Разместите на форме с палитры Standart компонент RoripMenu1.
- Привяжем RoripMenu1 к форме. Для этого в свойстве RoripMenu у формы fmMain укажите из списка RoripMenu1.
- Выделите RoripMenu1.
- Свойство Images у RoripMenu1 установите в ImageList1 – для того, чтобы подключить список пиктограмм к меню.
- Нажмите левой кнопкой мыши на RoripMenu1 два раза. Появится редактор меню.
- В редакторе меню RoripMenu1, выделив верхний левый угол, напишите в свойстве Caption Файл.
- Выделите пункт меню Файл левой кнопкой мыши. Нажмите правую кнопку мыши и выберете команду Create Submenu для создания подменю. Для подменю напишите в свойстве Caption Заккрыть программу.
- Для пункта Заккрыть программу (выделите его левой кнопкой мыши) установите через Object Inspector (аналогично как для главного меню):
 - свойство ImageIndex из выпадающего списка в код картинки для закрытия программы в связанном списке картинок;
 - свойство Shortcut – горячая клавиша – запишите Alt+X;
 - событие OnClick (закладка Events или два раза левой кнопкой мыши нажать на пункте Заккрыть программу): запишите метод Close() – закрыть.
- Под Файл напишите в свойстве Caption знак -. Это позволит провести разделительную черту в меню.
- Под разделительной линией в свойстве Caption напишите Данные.
- Под Данные напишите Документы.
- Под Документы напишите Запросы.
- Нажмите F9.
- Проверьте работу пункта контекстного (локального) меню по закрытию программы. Для этого нажмите правую кнопку мыши на форме. Заполнение меню продолжим позже.

Задание 2.4 Создание панели инструментов

Для создания панели инструментов с кнопками используется компонент типа TToolbar с закладки Win32.

Для добавления в панель инструментов новых кнопок необходимо щелкнуть правой кнопкой на Toolbar и выбрать команду New Button (новая кнопка) или New Separator (новый разделитель). Также на панель инструментов можно заносить компоненты переносом с палитры, например строки ввода (TEdit), обычные кнопки (TButton) и др.

Для панели инструментов для вызова отображения всплывающих подсказок для визуальных компонентов необходимо указать значение свойства ShowHint панели равным true.

Кнопки панели инструментов имеют тип `TToolButton`. По умолчанию кнопки на панели инструментов отображаются только с картинкой без сопроводительной надписи. Для отображения с надписью (как в пункте меню) необходимо указать свойство `ShowCaptions` равным `True`.

- Разместите на форме с палитры Win32 компонент `ToolBar1`.
- Свойство `Images` у `ToolBar1` установите в `ImageList1` – для того, чтобы подключить список пиктограмм к меню.
- Свойство `ShowHint` у `ToolBar1` установите в `true` – для того, чтобы отражать всплывающие подсказки.
- Нажмите правую кнопку мыши на `ToolBar1`. Выберите команду `New Button`.
- Для кнопки на `ToolBar1` установите через `Object Inspector`:
 - свойство `ImageIndex` из выпадающего списка в код картинки для закрытия программы в связанном списке картинок;
 - свойство `Shortcut` – горячая клавиша – запишите `Alt+X`;
 - свойство `Hint` – всплывающая подсказка – напишите `Закрыть программу`;
 - событие `OnClick` (закладка `Events` или два раза левой кнопкой мыши нажать на пункте `Закрыть программу`): запишите метод `Close()` – закрыть.
- Нажмите `F9`.
- Проверьте работу кнопки панели инструментов по закрытию программы.

Заполнение кнопками в панели инструментов продолжим позже.

Внимание:

Создавая одно и то же действие по закрытию программы в главном меню, контекстном меню, панели инструментов мы проделывали одни и те же действия (задавали свойства, программировали события), так как действия в пунктах меню и кнопках панели инструментов обычно дублируют друг друга. Чтобы проделать назначение свойств и событий для действия однократно используют концепцию действий (акций) `TAction`. Для этого необходимо предварительно сформировать список акций через компонент `TActionList`.

Использование невидимых компонентов типа `TAction` позволяет централизованно определять различные действия (акции), которые потом просто указываются для использования в качестве элементов различных меню, кнопок на панели инструментов и т.д. через их свойство `Action`.

Если нужно выполнить такие действия, как: изменить название команды, назначить горячую клавишу, запретить или разрешить команду и др., достаточно указать это только для компонента `TAction`, а он автоматически распространит изменения на все связанные с ним пункты меню и кнопки.

Пункты меню, кнопки панели инструментов, кнопки через их свойство Action выполняют соответствующие действия/акции из списка акций (ActionList1). Это позволяет оптимизировать программный код, так как действия по кнопкам панели инструментов и пунктам меню дублируют друг друга.

Описав действие один раз через TActionList (определив при этом заголовок действия (Caption), горячие клавиши (ShortCut), всплывающие подсказки (Hint), пиктограмму из списка пиктограмм ImageList1 и пр.), его можно использовать/вызывать как действие различных компонентов (в нашем случае меню и панель инструментов).

После того как созданы все необходимые действия, их необходимо назначить соответствующим пунктам меню и кнопкам панели инструментов. У этих компонентов имеется свойство Action, значение которого можно установить в инспекторе объектов с помощью выпадающего списка всех доступных в форме действий.

Задание 2.5 Формирование списка действий (акций)

- Положите на форму с палитры Standart компонент ActionList1.
- Свойство Images у ActionList1 установите в ImageList1 – для того, чтобы подключить список пиктограмм к меню.
- Нажмите на ActionList1 два раза левой кнопкой мыши. Откроется редактор акций (Action List Editor).
- Нажмите клавишу Insert или правую кнопку мыши New Action для добавления действия (акции).

При выборе мышкой действия в списке действий оно становится доступным в инспекторе объектов.

Список основных свойств и событий действий (акций) приведены в таблицах 1.1 и 1.2:

Таблица 1.1 - Основные свойства объектов типа TAction

| Свойства | Тип | Комментарий |
|----------|---------|---|
| Caption | String | Название действия в меню |
| Category | String | Категория - используется для внутреннего упорядочивания действий внутри TActionList |
| Checked | Boolean | Отмечены ли галочкой пункты меню и нажаты |

| | | |
|------------|-----------|--|
| | | ли соответствующие кнопки |
| Enabled | Boolean | Разрешена ли команда |
| Hint | String | Всплывающая подсказка для кнопок |
| ImageIndex | Integer | Код картинки в связанном списке картинок |
| Shortcut | TShortcut | Код горячей клавиши для вызова действия |
| Visible | Boolean | Видимы ли пункты меню и кнопки |

Таблица 1.2 - Основные события объектов типа TAction

| Событие | Комментарии |
|-----------|---|
| OnExecute | Выполнение действия |
| OnUpdate | Обновление информации о действии. Здесь можно изменить любые свойства действия в зависимости от текущего состояния программы. |

- Созданную акцию выделите левой кнопкой мыши и через *Object Inspector* установите для акции *Выход из программы*:

- свойство *Name* в *actExit*;
- свойство *Caption* в *Выход из программы*;
- свойство *Hint* в *Выход из программы*;
- свойство *ImageIndex* в из выпадающего списка в код картинки для закрытия программы в связанном списке картинок;
- свойство *Shortcut* – горячая клавиша – запишите *Alt+Q*;
- событие *OnExecute* (закладка *Events* или два раза левой кнопкой мыши нажать на акции) запишите:

```
if (Application->MessageBox("Вы хотите выйти из программы?",
"Внимание", MB_YESNO + MB_ICONQUESTION) == IDYES)
    Close();
```

- Теперь привяжем созданную акцию пунктам меню, кнопке панели инструментов.

- Нажмите два раза на компоненте *MainMenu1*. Откроется редактор меню *MainMenu1*.

- под пунктом *Закрыть программу* в свойстве *Caption* напишите знак - . Это позволит провести разделительную черту в меню.

- Под разделительной линией в свойстве *Action* для нового пункта выберете из выпадающего списка имя созданной акции *actExit*. При этом будут определены свойства и события созданной акции.
- Нажмите два раза на компоненте *РорирМени1*. Откроется редактор меню *РорирМени1*.
- под пунктом *Закреть программу* в свойстве *Caption* напишите знак - . Это позволит провести разделительную черту в меню.
- Под разделительной линией в свойстве *Action* для нового пункта выберете из выпадающего списка имя созданной акции *actExit*. При этом будут определены свойства и события созданной акции.
- Нажмите правую кнопку мыши на *ToolBar1*. Выберете команду *New Button*.
- Для кнопки на *ToolBar1* установите через *Object Inspector* свойство *Action* выберете из выпадающего списка имя созданной акции *actExit*. При этом будут определены свойства и события созданной акции.

Таким образом, определив один раз действие (акцию) ее можно задавать различным пунктам, кнопкам, а не несколько раз определять одно и тоже.

По аналогии добавьте через редактор акций *ActionList1* акции:

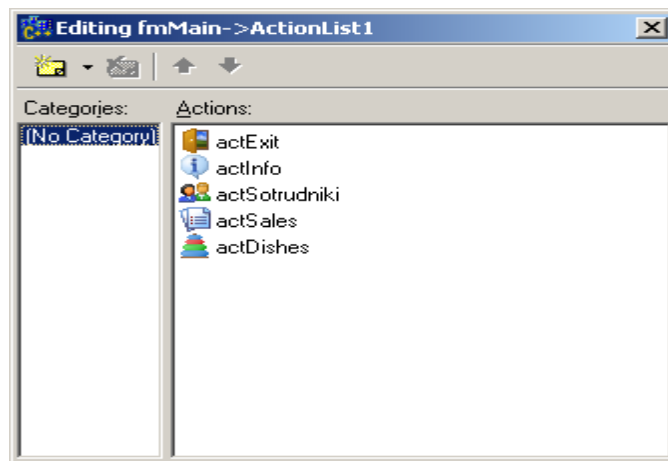
- для выдачи информации об авторах программы *actInfo*;
- для работы с данными по сотрудникам *actSotrudniki*;
- для работы с данными по блюдам *actDishes*;
- для работы с данными по продажам блюд *actSales*;
- для работы с данными запросов *actSQL*.

Укажите для каждой акции:

- свойство *Name* в имя соответствующей акции;
- свойство *Caption* в название акции на русском в визуальных компонентах;
- свойство *ImageIndex* в номер добавленной в *ImageList1* картинки;
- свойство *Hint* в текст соответствующей всплывающей подсказки;
- свойство *ShortCut* в необходимую комбинацию горячих клавиш;
- и пр. при желании (см. таблицу 1.1).

Событие *OnExecute* определим позже.

Например:



Затем назначьте главному меню, контекстному меню и панели инструментов соответствующие акции.

В главном меню:

- под пунктом Данные разместите actSotrudniki;
- под actSotrudniki разместите разделительную линию;
- под разделительной линией actDishes;
- под пунктом Документы actSales;
- под пунктом Запросы actSQL;
- под пунктом actExit разместите разделительную линию;
- под разделительной линией actInfo.

В контекстном меню:

- под пунктом actExit разместите разделительную линию;
- под разделительной линией actInfo;
- на пункте Данные создайте подменю, в котором укажите сначала пункт actSotrudniki, разделительную линию, actDishes;
- на пункте Документы создайте подменю, в котором укажите actSales;
- на пункте Запросы создайте подменю, в котором укажите actSQL.

На панели инструментов:

- после кнопки actExit разместите разделитель (New Separator);
- после разделите кнопку для actInfo;
- еще разделитель;
- после разделителя кнопку для actSotrudniki;
- кнопку для actDishes;
- еще разделитель;
- кнопку для actSales;
- еще разделитель;
- кнопку для actSQL.

Пока акции actInfo, actSotrudniki, actDishes, actSales, actSQL в меню, панели инструментов не обрабатывают, так как им не определено событие OnExecute. Как только оно будет определено, то действия будут обрабатывать.

- Зададим для акции actInfo событие OnExecute, для которого пропишите:

Application->MessageBox("(C)ТУСУР, КИБЭВС, Ваши ФИО и номер группы, 2008", "О программе", MB_ICONINFORMATION);

Примечание:

Согласно «Гражданскому кодексу Российской Федерации (часть четвертая)» от 18.12.2006 N 230-ФЗ (ред. от 01.12.2007) к объектам авторских прав относятся программы для ЭВМ.

Правообладатель для оповещения о принадлежащем ему праве вправе использовать знак охраны авторского права, который состоит из следующих элементов:

- латинской буквы "C" в окружности или скобках;
- имени или наименования правообладателя;
- года первого опубликования произведения.

Авторские права на произведение, созданное в пределах установленных для работника (автора) трудовых обязанностей (служебное произведение), принадлежат автору. Исключительное право на служебное произведение принадлежит работодателю, если трудовым или иным договором между работодателем и автором не предусмотрено иное.

Работодатель может при использовании служебного произведения указывать свое имя или наименование либо требовать такого указания.

Таким образом, знак охраны авторского права в данном случае может иметь вид:

(C)ТУСУР, КИБЭВС, Ваши ФИО, 2008.

Задание 2.6 Создание полосы состояния (строки статуса)

Для создания строки статуса используется компонент типа TStatusBar с закладки Win32.

Свойство SimplePanel в true – полоса состояний представляет собой единственную панель. Если свойство SimplePanel установлено в false, то полоса состояний является набором панелей, задаваемых свойством Panels. Основное свойство каждой панели – Text, в который заносится изображаемый в панели текст.

Программный доступ к текстам отдельных панелей можно проводить через индексированное свойство Items (начиная от нуля).

- Разместите с закладки Win32 компонент *StatusBar1*.
- Установите у *StatusBar1* свойство *SimplePanel* в *false*.
- Нажмите у *StatusBar1* через *Object Inspector* свойство *Panels*.

Появится окно редактора панелей строки статуса. В данном редакторе нажмите правую кнопку мыши и выберете команду *Add*. Добавится панель в строку статуса.

- Для добавленной панели с индексом 0 укажите свойство *Text* Программа для кафе, свойство *Width* в 120.

- Нажмите еще раз в редакторе команду *Add* и добавьте еще одну панель с индексом 1. Свойство *Text* для нее запрограммируем и будем показывать только после отработки акции *actInfo*. Для этого в событие *OnExecute* акции *actInfo* допишете:

StatusBar1->Panels->Items[1]->Text="(С)ТУСУР, КИБЭВС, 2008";

Задание 3. Подключение к БД

База данных для кафе имеет формат СУБД Microsoft Access—*Кафе.mdb*.

Для подключения к БД СУБД Microsoft Access будем использовать технологию ADO, так как технология ADO основана на стандартных интерфейсах COM, которые являются системным механизмом Windows, это сокращает общий объем работающего программного кода и позволяет распространять приложения БД без вспомогательных программ и библиотек.

Для размещения компонентов доступа к данным в приложении баз данных желательно использовать модуль данных (класс *TDataModule*).

В модуле данных можно размещать только невизуальные компоненты. Модуль данных доступен разработчику, как и любой другой модуль проекта, на этапе разработки. Пользователь приложения не может увидеть модуль данных во время выполнения приложения.

Модуль данных имеет мало общего со стандартной формой, хотя бы потому, что класс *TDataModule* происходит непосредственно от класса *TComponent*. У него почти полностью отсутствуют свойства и методы-обработчики событий, ведь от платформы для других невизуальных компонентов почти ничего не требуется, хотя потомки модуля данных, работающие в распределенных приложениях, выполняют весьма важную работу.

Преимуществом размещения компонентов доступа к данным в модуле данных является то, что изменение значения любого свойства проявится сразу же во всех обычных модулях, к которым подключен этот модуль данных. Кроме этого, все обработчики событий этих компонентов, т. е. вся

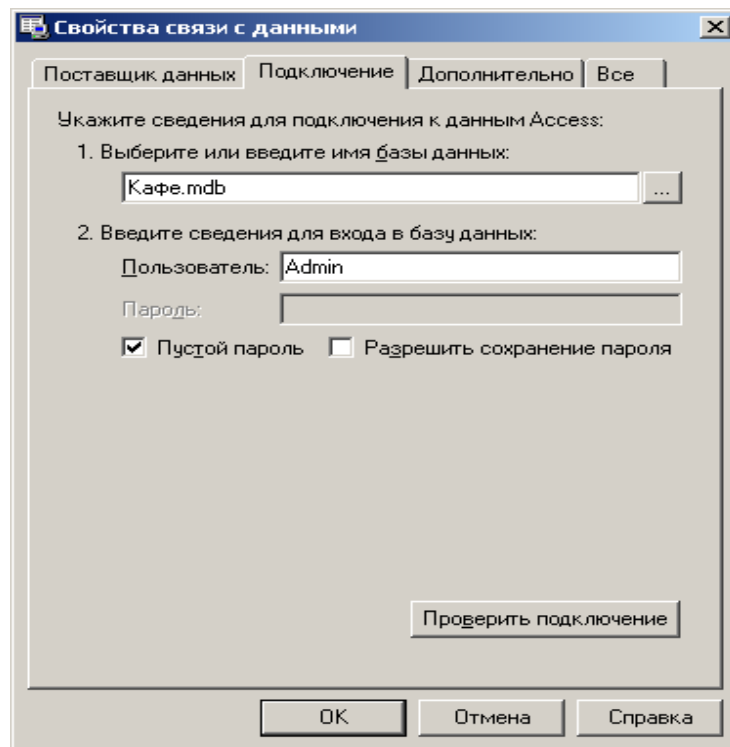
логика работы с данными приложения, собраны в одном месте, что тоже весьма удобно.

- Для создания модуля данных нажмите *File/New/Data Module*.
- Свойство *Name* у модуля данных укажите в *DMMain*.
- Нажмите *File / Save all* для сохранения модуля данных в проект. В диалоговом окне сохраните модуль данных в той же папке, что и другие модули проекта с именем *DM.cpp*.

Как уже говорилось ранее, для подключения к БД будем использовать технологию ADO.

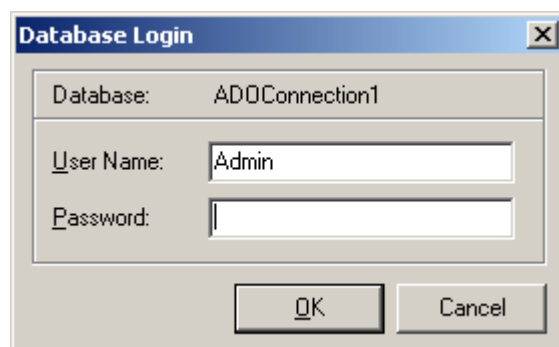
- Перейдите на закладку *ADO*.
- Разместите на модуле данных компонент *ADOConnection1* для подключения к БД. *ADOConnection* является невизуальным при запуске приложения.
- Для *ADOConnection1* установите свойства:
 - 1) через *Object Inspector* нажмите у *ADOConnection1* свойство *ConnectionString* (или для установки этого свойства можно нажать два раза левой кнопкой мыши на *ADOConnection1*).
 - 2) В появившемся диалоговом окне выберите *Use Connection String* и нажмите кнопку *Build...*
 - 3) В появившемся диалоговом окне перейдите на вкладку *Provider* (Поставщик данных). Среди списка *OLE DB Provider(s)* (Поставщиков OLE DB) выберите *Microsoft Jet 4.0 OLE DB Provider*. В *Microsoft Jet* систему баз данных можно воспринимать как менеджер данных, на которых системы баз данных, такие как *Microsoft Access*, строятся.
 - 4) Перейдите на закладку *Connection* (Подключение) и выберите или пропишите путь к базе данных (в нашем случае к файлу *Кафе.mdb*).
У нас база данных находится в той же папке, что и исполняемый файл *.exe*. Чтобы не быть «привязанным» к установленному пути к БД (например, в нашем случае *D:\Temp\Petrov\Proga_Caffe\Кафе.mdb*) оставьте только имя базы данных *Кафе.mdb*. В этом случае программа будет работать с БД с любого диска, при условии, что исполняемый файл и БД находятся в одной папке.

Мы не меняли логин и пароль к БД в программе *Microsoft Access*. Поэтому по умолчанию к БД *Access* используется пользователь (*User name*) *Admin*, а пароль пустой. Что и видим в настройках подключения к БД:



5) Нажмите кнопку *OK*.

6) Установите у *ADODConnection1* свойство *Connected* в *True*.
Будет запрошен диалог для ввода пользователя и его пароля:



В нашем случае надо ввести User Name *Admin* и пароль пустой.

7) чтобы каждый раз не вводить User Name и Password установите у *ADODConnection1* свойство *LoginPrompt* в *false*.

После подключения к БД подключим таблицы БД в нашем приложении.

Для подключения таблиц БД MS Access могут быть использованы различные компоненты, например, *TADOTable*, *TADOQuery*, *TADODataSet*.

Начнем изучение с компонента *TADOTable*.

Примечание:

TADOTable – реализует набор данных (НД), источником данных для которого является таблица БД.

TADOTable является невизуальным при запуске приложения.

Понятие набора данных несколько шире, чем понятие таблицы БД, так как набор данных может содержать:

- подмножество записей или полей определенной таблицы БД;
- записи, поля из нескольких таблиц БД;
- записи, поля значения которых формируются при помощи

вычислений (вычисляемые).

- Разместите на модуле данных DMMain компонент ADOTable1 для подключения таблицы Сотрудники в БД.

- Установите у ADOTable1 свойство Name в ADOT_Sotrudniki.

- Установите у ADOT_Sotrudniki свойство Connection в ADOConnection1.

- Установите у ADOT_Sotrudniki свойство TableName из списка таблиц БД в необходимую нам таблицу БД Сотрудники.

- Установите у ADOT_Sotrudniki свойство Active в true для соединения с таблицей Сотрудники в БД.

Примечание:

Если будете менять настройки у TADOTable, то необходимо свойство Active устанавливать в false (разрывать соединение с таблицей БД), а затем опять после смены настроек устанавливать в true (устанавливать соединение с таблицей БД).

- Разместите аналогично на модуле данных компоненты TADOTable для подключения таблиц Блюда, Продажи, Продано в БД.

- Установите им свойства Name соответственно таблицам:

- для таблицы Блюда - ADOT_Dishes;
- для таблицы Продажи - ADOT_Sales;
- для таблицы Продано - ADOT_DisheSale.

- Установите ADOT_Dishes, ADOT_Sales, ADOT_DisheSale свойство Connection в ADOConnection1.

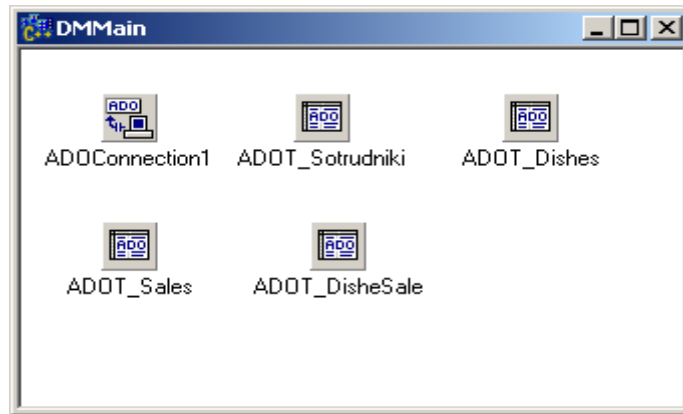
- Установите у ADOT_Dishes свойство TableName из списка таблиц БД в необходимую таблицу БД Блюда.

- Установите у ADOT_Sales свойство TableName из списка таблиц БД в необходимую таблицу БД Продажи.

- Установите у ADOT_DisheSale свойство TableName из списка таблиц БД в необходимую таблицу БД Продано.

- Установите у ADOT_Dishes, ADOT_Sales, ADOT_DisheSale свойство Active в true для соединения с таблицами БД.

Вид модуля данных имеет вид:



После подключения данных из созданной БД перейдем к отображению данных в приложении.

Компоненты TADOConnection и TADOTable являются невидимыми при запуске приложения.

Задание 4. Отображение данных БД в визуальных компонентах

Будем отражать данные БД через отдельную форму.

- *Нажмите в меню File/New/Form.*
- *Свойство Name у новой формы напишите в fmList.*
- *Сохраните форму в проект. Для этого нажмите File/Save all и в диалоговом окне сохраните новую форму в той же папке, что и другие модули проекта. При сохранении укажите новому модулю имя List.cpp.*
- *Перейдите на форму fmList.*
- *Перейдите на вкладку DataAccess. Найдите на данной вкладке компонент DataSource и разместите его объект (DataSource1) на форму fmList.*

Примечание:

Компонент TDataSource служит промежуточным звеном в цепочке: “набор данных – TDataSource – визуальные компоненты для работы с данными”.

Он позволяет устанавливать некоторые параметры наборов данных, устанавливать состояние наборов данных, отслеживать изменения в наборах данных.

TDataSource через свойство DataSet связывается с необходимым набором данных.

TDataSource является невидимым компонентом в запущенном приложении.

- *Наши наборы данных находятся на модуле данных в модуле DM, который включает в себя два файла:*

*DM.cpp – реализация методов;
DM.h – объявления классов модуля.*

*Данный файл DM.h необходимо подключить в форме fmList.
Для этого, находясь на форме fmList, выберете в меню File/Include Unit Hdr... (или нажмите Alt+F11). В диалоговом окне Use Unit выберете модуль DM.*

*После этого в модуле List.cpp появится запись #include "DM.h"
Или можно было просто в модуле List.cpp прописать #include "DM.h".*

-Установите у DataSource1 свойство DataSet в набор данных ADOT_Dishes на модуле данных DMMain: DMMain->ADOT_Dishes.

Далее визуальные компоненты для работы с данными в БД работают с TDataSource, т.е. согласно цепочке “набор данных – TDataSource – визуальные компоненты для работы с данными”.

К визуальным компонентам работы с данными в БД относятся компоненты со страниц: Data Controls, QReport и др.

Примеры визуальных компонент для работы с БД приведены в таблице 1.

Остальные будут рассмотрены в ходе дальнейшего выполнения лабораторных работ.

Таблица 1 – Примеры визуальных компонент для работы с БД

| Компонент | Назначение |
|-----------------|--|
| TDBText | Показывает "только для чтения" значение поля текущей записи набора данных. |
| TDBEdit | Обеспечивает просмотр и изменение значения поля текущей записи набора данных. Поля - любого типа, кроме полей комментариев и BLOB. |
| TDBMemo | Позволяет просматривать и корректировать значения Мемо-поля (поля комментария) в режиме текстового редактора. |
| TDBImage | Позволяет просматривать графические поля, заносить в них содержимое. |
| TDBGrid | Показывает содержимое полей набора данных в "табличном виде", когда записям соответствуют строки, |

| | |
|----------------------|--|
| | а полям - столбцы. |
| TDBCtrlGrid | Версия компонента TDBGrid, позволяющая показывать содержимое одной записи набора данных в нескольких строках. |
| TDBNavigator | Позволяет осуществлять навигацию по записям набора данных, переводить набор данных в состояние вставки, изменения, удаления, запоминания изменений. |
| TDBCheckBox | Обеспечивает просмотр и изменение значения поля типа Boolean текущей записи набора данных. |
| TDBListBox | Применяется, когда необходимо выбрать значение поля из предустановленного списка значений; значения показываются в виде строк в списке фиксированного размера. Содержимое списка определяется свойством Items. |
| TDBComboBox | Применяется для тех же целей, что и TDBListBox, но список выпадающий. |
| TDBRadioGroup | Обеспечивает возможность выбора значения для поля, которое может содержать фиксированное число вариантов значений. Значения показываются в виде радиокнопок. |
| TDBRichEdit | Имеет то же значение, что и TDBMemo, но позволяет работать с текстом формата RTF, включающим разные шрифты, графику и пр. |
| TDBChart | Используется для построения графиков. |

Рассмотрим более подробно использование визуальных компонентов.

Компонент TDBText:

Применяется для показа значения текстового поля текущей записи набора данных. Изменять значение, показываемое при помощи TDBText нельзя.

Для использования TDBText необходимо:

- свойство DataSource – имя необходимого компонента TDataSource, связанного с НД;

- свойство DataField - имя необходимого поля соответствующего НД.
- свойство Name – логическое имя компонента, по которому можно обратиться к конкретному компоненту TDBText.

Компонент TDBEdit:

Обеспечивает просмотр и изменение значения поля текущей записи набора данных. Поля - любого типа, кроме полей комментариев и BLOB.

Для использования TDBEdit необходимо:

- свойство DataSource – имя необходимого компонента TDataSource, связанного с НД;
- свойство DataField - имя необходимого поля соответствующего НД.
- свойство ReadOnly – если содержит true, то значение поля доступно только для чтения, если false – значение поля можно изменять.
- свойство Name – логическое имя компонента, по которому можно обратиться к конкретному компоненту TDBEdit.

При вводе значения в TDBEdit приложение автоматически отслеживает, чтобы введенное значение было совместимо по формату с полем набора данных. Ввод неверных данных блокируется путем генерации исключений.

Событие OnChange наступает при изменении значения поля.

Событие OnEnter наступает при получении и утрате фокуса управления компонентом TDBEdit.

Компонент TDBNavigator:

Позволяет осуществлять навигацию по записям набора данных, переводить набор данных в состояние вставки, изменения, удаления, запоминания изменений.

Компонент TDBNavigator является набором кнопок, с помощью которых пользователь может перемещаться между записями набора данных, выполнять операции вставки и удаления записей, вносить сделанные изменения в базу данных или отменять их. Полный вид TDBNavigator приведен на рисунке 1.

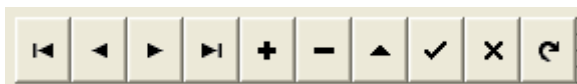










Рисунок 1 – Полный вид компонента TDBNavigator

В таблице 2 приведена расшифровка значений кнопок навигатора.

Таблица 2 – Расшифровка значений кнопок навигатора

| Кнопка | Значение |
|--------|--------------------------------|
| | Переместиться в начало НД |
| | Перейти к предыдущей записи НД |

| | |
|---|---------------------------------------|
|  | Перейти к следующей записи НД |
|  | Переместиться в конец НД |
|  | Добавить новую запись |
|  | Удалить существующую запись |
|  | Изменить существующую запись |
|  | Сохранить новую или измененную запись |
|  | Отменить ввод или изменение записи |
|  | Обновить содержание НД |

Для использования TDBNavigator необходимо:

- свойство DataSource – имя необходимого компонента TDataSource, связанного с НД;
- свойство VisibleButtons – определяет, какие кнопки будут отображены в навигаторе.

Компонент TDBCheckBox:

Позволяет «отметить» и «снять отметку» с логического поля в составе текущей записи НД.

Для использования TDBCheckBox необходимо:

- свойство DataSource – имя необходимого компонента TDataSource, связанного с НД;
- свойство DataField - имя необходимого поля соответствующего НД.
- свойство ReadOnly – если содержит true, то значение поля доступно только для чтения, если false – значение поля можно изменять.
- свойство Name – логическое имя компонента, по которому можно обратиться к конкретному компоненту TDBCheckBox.
- свойство Checked – позволяет определить отмечено ли поле, на которое ссылается TDBCheckBox (значение true) или не отмечено (false).
- свойство State – возвращает состояние поля. Возможные значения: cbChecked – поле отмечено, cbUnchecked – поле не отмечено, cbGrayed промежуточное состояние, когда поле не отмечено, но в нем показывается серый символ отметки. Он означает, что поле не содержит ни true, ни false, а содержит пустое значение. Это состояние присуще при добавлении записей.

Компонент TDBCheckBox можно связывать и не с логическим, а с символьным полем. В этом случае необходимо:

- свойство ValueChecked – устанавливает значения поля, при которых TDBCheckBox переходит в состояние cbChecked. При наличии нескольких значений они разделяются точкой с запятой DBCheckBox1->ValueChecked="Да;Yes;True";
- свойство ValueUnchecked – устанавливает значения поля, при которых TDBCheckBox переходит в состояние cbUnchecked. При наличии

нескольких значений они разделяются точкой с запятой: `DBCheckBox1->ValueUnchecked="Нет;No;False";`

Событие `OnClick` наступает, если на `TDBCheckBox` щелкнуть один раз левой кнопкой мыши.

Событие `OnEnter` наступает при получении и утрате фокуса управления компонентом `TDBCheckBox`.

Компонент TDBRadioGroup:

Служит для предоставления фиксированного набора возможных значений поля при помощи группы зависимых переключателей.

Для использования `TDBRadioGroup` необходимо:

- свойство `DataSource` – имя необходимого компонента `TDataSource`, связанного с НД;
- свойство `DataField` - имя необходимого поля соответствующего НД.
- свойство `ReadOnly` – если содержит `true`, то значение поля доступно только для чтения, если `false` – значение поля можно изменять.
- свойство `Name` – логическое имя компонента, по которому можно обратиться к конкретному компоненту `TDBRadioGroup`.
- свойство `Items` – определяет число и названия вариантов возможных значений поля, содержащихся в `TDBRadioGroup`.
- свойство `TDBRadioGroup->ItemIndex` – позволяет определить индекс текущего выбора. Возвращает номер выбранного значения в порядке, в котором они определены в `TDBRadioGroup->Items`. Отсчет ведется от нуля.
- свойство `Columns` – указывает сколько назначено столбцов для вывода переключателей.

Событие `OnClick` наступает, если на `TDBRadioGroup` щелкнуть мышью.

Событие `OnEnter` наступает при получении и утрате фокуса управления компонентом `TDBRadioGroup`.

Компонент TDBListBox:

Применяется, когда нужно выбрать значения поля из предустановленного списка значений. Возможные значения содержатся в качестве строк компонента `TDBListBox`.

Для использования `TDBListBox` необходимо:

- свойство `DataSource` – имя необходимого компонента `TDataSource`, связанного с НД;
- свойство `DataField` - имя необходимого поля соответствующего НД.
- свойство `ReadOnly` – если содержит `true`, то значение поля доступно только для чтения, если `false` – значение поля можно изменять.
- свойство `Name` – логическое имя компонента, по которому можно обратиться к конкретному компоненту `TDBListBox`.
- свойство `Items` – содержит список возможных значений поля, содержащихся в `TDBListBox`.

Событие `OnClick` наступает, если на `TDBListBox` щелкнуть мышью.

Событие OnDblClick наступает, если на TDBListBox щелкнуть двойным нажатием мыши.

Событие OnEnter наступает при получении и утрате фокуса управления компонентом TDBListBox.

Компонент TDBComboBox:

Этот компонент применяется для тех же целей, что и TDBListBox, но список выпадающий.

Остальные визуальные компоненты для работы с данными БД будут рассмотрены в ходе дальнейшего выполнения лабораторных работ.

- *Перейдите на форму fmList.*
 - *Перейдите на вкладку Data Controls.*
 - *С вкладки Data Controls разместите на форме fmList компонент DBGrid1.*
 - *Установите у компонента DBGrid1 свойство DataSource в DataSource1. После этого Вы должны увидеть отображение таблицы Блюда в компоненте DBGrid1.*
 - *С вкладки Data Controls разместите на форме fmList компонент DBNavigator1.*
 - *Установите у компонента DBNavigator1 свойство DataSource в DataSource1.*
 - *Если поменять у DataSource1 свойство DataSet в набор данных ADOT_Sotrudniki на модуле данных DMMain: DMMain-> ADOT_Sotrudniki, то Вы должны в DBGrid1 увидеть отображение таблицы Сотрудники. И так далее с другими таблицами. Значит, меняя программно у DataSource1 свойство DataSet в необходимый набор данных, можно отобразить данные всех наших таблиц БД, используя один DataSource1 и один DBGrid1 через одну форму.*
 - *Перейдите на форму fmMain, чтобы запрограммировать показ нашей формы с отображением данных разных таблиц на форме fmList.*
 - *Подключим данную форму fmList в форме fmMain. Прodelайте подключение модуля List аналогично с подключением модуля DM в форме fmList.*
- Модуль List включает в себя два файла:*
- List.cpp – реализация методов;*
 - List.h – объявления классов модуля.*
- Данный файл List.h необходимо подключить в форме fmMain.*
- Для этого, находясь на форме fmMain, выберете в меню File/Include Unit Hdr... (или нажмете Alt+F11). В диалоговом окне Use Unit выберете модуль List.*
- После этого в модуле Main.cpp появится запись #include "List.h"*
- Или можно было просто в модуле Main.cpp прописать #include "List.h".*

- Запрограммируем наши акции в *ActionList1*.
- Нажмите два раза левой кнопкой мыши на компоненте *ActionList1*.

Откроется редактор акций.

- Выделите акцию *actSotrudniki* и перейдите на событие *OnExecute* (через *Object Inspector* закладка *Events* или нажмите два раза мышью на акции).

- У акции *actSotrudniki* на событие *OnExecute* запишите:

```
fmList->DataSource1->DataSet=DMMain->ADOT_Sotrudniki;  
fmList->Caption="Список сотрудников";  
fmList->ShowModal();
```

Примечание:

Метод у формы *ShowModal()* позволяет показать форму модально, т.е. нельзя перейти в другую форму проекта пока не будет закрыта открытая модально форма.

Метод у формы *Show()* позволяет показать форму, и при открытой форме возможно переходить в другие формы проекта, не закрывая открытую форму.

- Наша акция *actSotrudniki* уже была ранее нами привязана к главному меню, контекстному меню, панели инструментов. Следовательно, запустив программу можно, нажимая соответствующие пункты меню или кнопки, просмотреть данные по сотрудникам.

- Прodelайте аналогично программирование события *OnExecute* акций *actDishes* – для показа данных по блюдам, для *actSales* – для показа данных по продажам.

- Нажмите *F9*.

- Просмотрите, что появилась возможность работать с данными разных таблиц через отдельную форму.

Примечание:

Для добавления записей в *TDBGrid* клавиша-*Ins* или курсор вниз, отказ-*Esc*, удаление *Ctrl+Delete*. Либо использовать функции *TDBNavigator*.

- Обратите внимание, что при отображении данных по блюдам *DBGrid1* не отображает поля типа *Метод* и графические поля. Отобразим их. Для этого перейдите на форму *fmList*.

- Перейдите на вкладку *DataAccess*. Найдите на данной вкладке компонент *DataSource* и разместите его объект на форму *fmList*.

- Для *DataSource2* установите свойство *Name* в *dsDishes* – так мы обозначим его привязку к набору данных *ADOT_Dishes* (для таблицы *Блюда*).

- Разметите на форме *fmList* с закладки *Standart* компонент *GroupBox1* – он является контейнером, объединяющим группу связанных органов управления.
- Установите у *GroupBox1* свойство *Caption* в *Информация о блюде*.
- На компонент *GroupBox1* разместите:
 - с закладки *Data Controls* разместите на компонент *GroupBox1* компонент *DBText1*. Установите для него:
 - свойство *Align* (выравнивание компонента внутри контейнера) в *alTop* (занять верхнюю часть контейнера, в данном случае *GroupBox1*);
 - свойство *DataSource* (источник данных) в *dsDishes*;
 - свойство *DataField* (имя поля) в *Название*;
 - с закладки *Data Controls* разместите на компонент *GroupBox1* компонент *DBImage1*. Установите для него:
 - свойство *Align* в *alTop*;
 - свойство *DataSource* в *dsDishes*;
 - свойство *DataField* в *Фото*;
 - с закладки *Additional* разместите на компонент *GroupBox1* компонент *Splitter1* (движок). Установите для него свойство *Align* в *alTop*. *Splitter1* позволит в процессе работы приложения изменять размеры блоков для *Фото* и *Состава* блюда;
 - с закладки *Standart* разместите на компонент *GroupBox1* компонент *Label1* (метка). Установите у *Label1* свойство *Caption* в *Состав блюда*: Установите для *Label1* свойство *Align* в *alTop*;
 - с закладки *Data Controls* разместите на компонент *GroupBox1* компонент *DBMemo1*. Установите для него:
 - свойство *Align* в *alClient* (занять всю клиентскую область контейнера, в данном случае *GroupBox1*);
 - свойство *DataSource* в *dsDishes*;
 - свойство *DataField* в *Состав*.
- Далее настроим форму *fmList*, что при работе приложения либо показывать (когда отражаются данные по блюдам), либо прятать компонент *GroupBox1*.
- Установите у *GroupBox1* свойство *Align* в *alRight* (занять правую часть контейнера, в данном случае формы *fmList*).
- С закладки *Additional* размещена форма компонент *Splitter2* (движок). Установите для него свойство *Align* в *alRight*. *Splitter2* позволит в процессе работы приложения изменять размеры блоков для *DBGrid1* и *GroupBox1*.
- У *DBGrid1* свойство *Align* установите в *alClient* (занять всю клиентскую область контейнера, в данном случае *fmList*).
- У компонента *DBNavigator1* установите свойство *Align* в *alTop*.
- На момент показа формы *fmList* будем переопределять вид формы. Для этого у формы *fmList* в событии *OnShow* пропишите:

```

if (DataSource1->DataSet == DMMain->ADOT_Dishes)
{ //для списка блюд особый вид
    GroupBox1->Visible=true;
    Splitter2->Left = GroupBox1->Left;
    Splitter2->Visible=true;
}
else
{
    GroupBox1->Visible=False;
    Splitter2->Visible=False;
}

```

- Нажмите F9.
- Посмотрите, что для списка блюд определен особый вид.

Примечание:

Таким образом, в основе процесса разработки лежит триада компонентов:

- невидимые компоненты набора данных;
 - невидимые компоненты TDataSource;
 - визуальные компоненты отображения данных.
-

Задание 5. Оформление отчета к лабораторной работе

Оформить отчёт со следующим содержанием:

1. Титульный лист.
 2. Цель работы.
 3. Постановка задачи.
 4. Краткая теория и ход выполнения заданий.
 5. Описание результатов.
- Приведите скриншоты полученной программы и листинг программы.
6. Заключение (выводы).