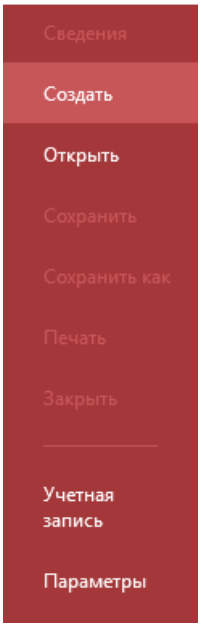


ПРОЕКТИРОВАНИЕ С MICROSOFT ACCESS

Книга разработчика баз данных

Часть 1

Введение в программирование на языке Microsoft VBA (Microsoft Visual Basic For Application)

Все права защищены

Перепечатка всей книги запрещена без письменного разрешения автора, анонсы и комментарии к материалам разрешены при указании ссылки на первоисточник

Авторы

Конюков Виктор, leadersoft.ru

… это место свободно …

ОГЛАВЛЕНИЕ	ССЫЛКИ ПО ТЕМЕ
1. Введение в проектирование	Вся книга в интернете
2. Определение переменных	Форум по Microsoft Access
3. Создание базы данных	Книга для чтения в формате RSS
4. Работа с таблицами	Загрузить книгу в pdf формате
5. Работа с SQL запросами	Загрузить исходники по теме (скоро …)
6. Создание модуля	Подписка на новости по Email
7. Создание формы	Сообщество Microsoft Access в Google +
8. Создание макроса	Заказ обучения или разработки программы
9. Создание панели меню	

ВВЕДЕНИЕ

Одним из самых важных моментов проектирования в Microsoft Access является создание интерфейса понятного не только Вам, но и пользователю.

Через некоторое время становиться ясно то, что для вас очевидно, для других нет. Ваши кнопки имеют не тот размер, расположены не в том месте, да и цвет у них серый - мрачный. А нужно сделать так, чтобы все блестело, сияло и прыгало на экране.

Таким образом, с этого момента Вы понимаете, что для разработки базы данных недостаточно иметь общие знания вузовского программирования, надо быть психологом, прорицателем, т.е. таким человеком, который будет угадывать мысли заказчика, еще до того, как стал писать ему базу данных. Поэтому и возникла идея написать эту книгу , чтобы помочь разработчикам, т.е. тем, кто "зарабатывает на хлеб, а возможно и масло" созданием программного обеспечения.

Как будет построена книга

> Придумать и написать хорошую книгу сразу невозможно - это тоже самое как создать "с нуля" программу без предварительных версий.

Придется эту книгу переделывать, какие-то разделы изменять, дописывать, удалять статьи, перемещать и т.п. Но надеюсь, что в результате получится некий интересный материал, который можно будет потом превратить в обычную книгу(и), т.е. pdf формат для общего чтения.

В книгу войдут не только информация по программированию или разработке интерфейса средствами Microsoft Access, но и также расширенные знания, т.е. использование Access в ASP.NET (сайты), WinForms (настольные приложения), т.е. там где нужно использовать для разработки Microsoft Visual Studio.

Если покажется, что читать ее скучно, то будут добавлены курьезные рассказы, анекдоты и другой материал на отвлеченные темы.

Что такое Microsoft Access

> Microsoft Access - эта программа позволяет работать с базами данных и содержит средства управления интерфейсом пользователя, например, отчетами и формами.

Программа распространяется как отдельное приложение, а также входит в пакет программ Microsoft Office. Скорость создания интерфейса в Microsoft Access выше, чем в других системах. Все это и привело к тому, что Access является признанным средством разработки во всем мире.

При этом изменения в программу могут вносить и специалисты, не имеющие навыков программирования. Программа также сертифицирована для применения в закрытых, оборонных разработках. Минусом программы можно назвать только работу в сети при числе пользователей более 25-30 человек, но это легко исправить если к этой программе подключить SQL Server. Вот состав базы данных

- Таблицы - они используются для хранения конкретных данных: чисел, текста, изображений в базе данных
- Язык SQL - это специальный язык управления применяется для выполнения запросов из таблиц базы данных
- Формы - используются для построения интерфейса, например, кнопок с которыми работает пользователь
- Отчеты - этот раздел программы позволяет вам создавать различные виды отчетов для вывода на печать их из базы данных
- VBA - язык программирования для создания взаимодействия интерфейса форм, отчетов, запросов и таблиц

Об автоматизации бизнеса

Не знаете, как начать автоматизировать свой бизнес. Все готовые программы имеют недостатки. Так может быть проще все проблемы решить через Microsoft Access.

Это лучшая программа, которую может освоить любой начинающий. При небольших финансовых затратах можно не только улучшить работу своего предприятия, но и открыть для себя новые направления в работе компании.

Учет клиентов

Разработка клиентских программ - это популярное направление в Microsoft Access. Такие программы заказывают те, кому нужно учесть мельчайшие детали переговоров с заказчиками. Постоянное напоминание им о том, что Вы существуете, рассылка писем, звонки и другие методы контактов позволят Вам не только сохранить, но и расширить базу базу клиентов.

Сети и Маркетинг

Москва, Санк-Петербург - это отличные рынки сбыта вашей продукции. Как правильно назначить торговых представителей и контролировать их деятельность. Это лучше сделать на Microsoft Access и SQL Server. Все это не только решит задачу удаленных контактов, но и позволит центральному офису получать отчеты в формате Access, Word, Excel быстро и своевременно влиять на деятельность вашего предприятия

Финансы

Вы - финансовый директор. Куда уплывают деньги, какой филиал или подразделение их неправильно расходует, как заштопать дыры в бюджете. Все это Вы пытаетесь решить с помощью электронных таблиц Excel. Но это простейший и элементарный этап действий. Совместите Access и Excel, и вы откроете для себя новые сферы финансового анализа.

Себестоимость

У вас производство. Затраты на сырье, зарплату, воду, электричество и офисные расходы - это все влияет на стоимость продукции. Можно ли решить задачу экономии этих ресурсов, не прибегая к сложным бухгалтерским программам. Не только можно, но и нужно сделать на базе Microsoft Access.

Склад и Реализация

Оказывается в Microsoft Access можно решать вопросы торговли очень быстро. Учет накладных, счетов фактур, наличных расчетов уже давно решены в Microsoft Access. Для многих руководителей компаний, вопроса купить готовую складскую программу или построить свою - не существует. Они сами, без языка программирования, строят такие проекты

Договора

Кто же НЕ использует Microsoft Word для составления договоров? Но намного лучше будет, если Вы создадите типовые договора и подключите к ним и Microsoft Access. Это не только решит задачу правильной организации документооборота в вашем офисе, но и позволит быстро заполнять договора, делать их поиск и выводить на печать.

Проблемы

Помните, что у Microsoft по данному продукту нет системы проверки качества на полную совместимость mde, accde, ade файлов Access 2000, 2002, 2003, 2007, 2010, 2013.

Поэтому созданные с помощью разных офисов файлы могут не запускаться на компьютерах из-за отсутствия библиотек, прав доступа даже при наличии только их программного обеспечения.

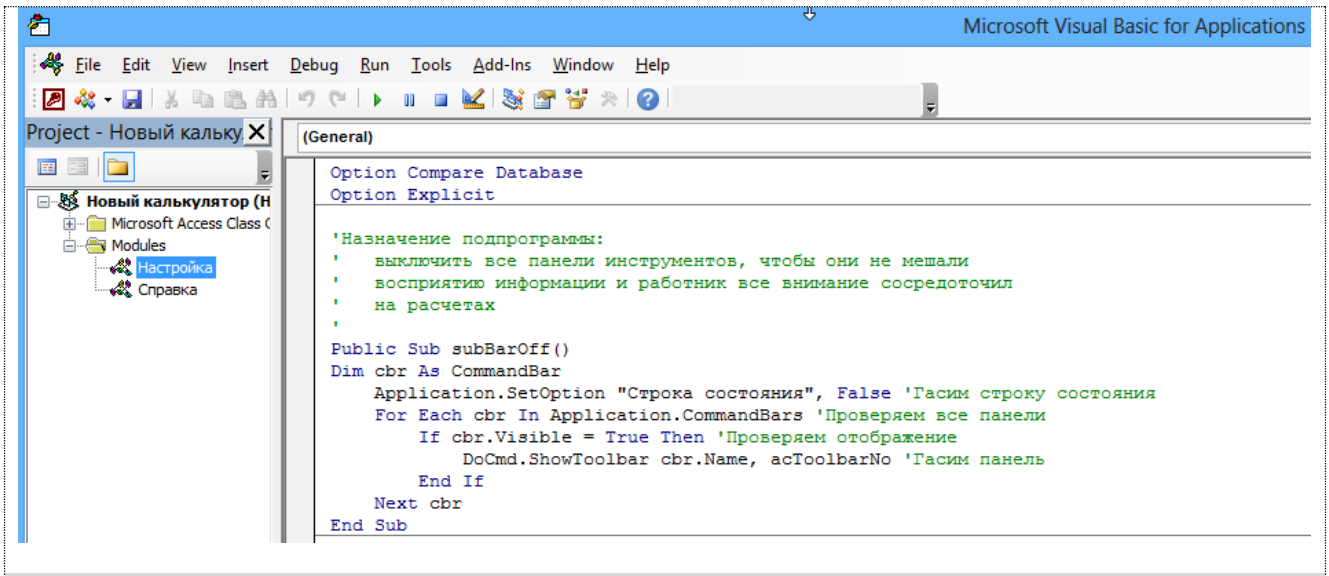
Обязательно нужно проверить и установку сервис пакетов продуктов. Большинство проблем можно избежать, если имеется полный доступ к исходным кодам программ, путем их перекомпиляции для конкретного офиса.

Также новые версии Access не поддерживают работу с SQL Server через ade и adp проекты. Эти файлы не читаются, но их можно переделать на другую платформу, т.е. mdb, accdb файлы. Таким образом, при заказе программы на Access нужно учитывать это обстоятельство.

ОПРЕДЕЛЕНИЕ ПЕРЕМЕННЫХ

Изучение программирования Microsoft Access начинаем с определения глобальных переменных.

Переменные бывают локальными и глобальными. Локальные - действуют на уровне функций, глобальные доступны из любого места программы. Существуют также и другие ограничения видимости переменных, например, классы. Для определения переменных в Access и ввода кода на Visual Basic существует специальная среда для работы с этими программами.



На этом снимке отображается специальная среда для ввода кода VBA и его отладки. С помощью ее разработчики вводят, проверяют на ошибки и отлаживают код (Debug). Функциональность среды разработки программ в Access 2016 версии не изменилась по сравнению с Access 2000, наоборот уменьшилась. Нет репликации, проектов adp и т.п.

Для начала определим название нашей программы. Пусть это будет приложение Microsoft Access «Новый калькулятор.mdb». Разработку его начнем с описания переменных, констант, которые потребуются при создании нашей базы данных.

appAccess – это объектная переменная, в которой мы будем создавать новую базу данных

appCM – это константа, которая управляет размером управляющих элементов в формах, отчетах. В Access элементы измеряются в твипах, а нам удобнее указывать размеры в сантиметрах, поэтому, и вводим эту переменную.

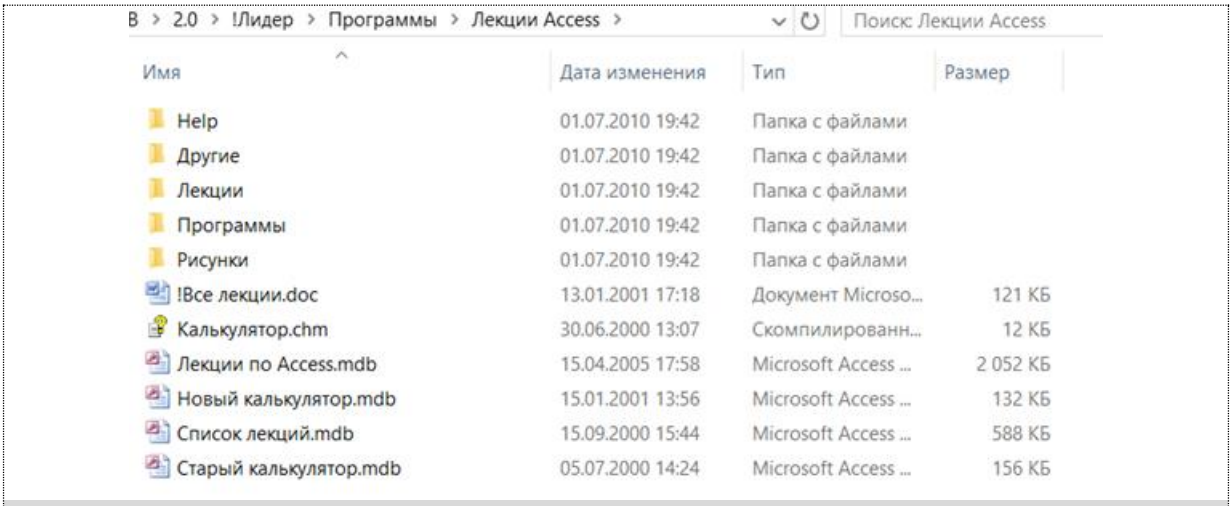
appFolder – эта переменная будет содержать главный каталог нашего приложения. Далее мы будем использовать ее для загрузки рисунков, программ, справки и других внешних элементов базы данных

```
Public appAccess As Access.Application 'Приложение программы
Public Const appCM As Integer = 567  'Размерность (1см=567 твип)
Public appFolder As String 'Папка приложения
```

Создание приложения

Продолжим создание приложения с разработки всех функций, которые требуются для работы базы данных. Сначала программным способом, т.е. без мастеров, создадим базу данных, затем таблицы, запросы и т.д.

Создадим главную функцию, которая будет связывать в единое целое создание объектов: таблиц, запросов, форм и т.п.



На этом снимке отображается файл Новый калькулятор.mdb, в котором будут создаваться таблицы, формы и другие объекты.

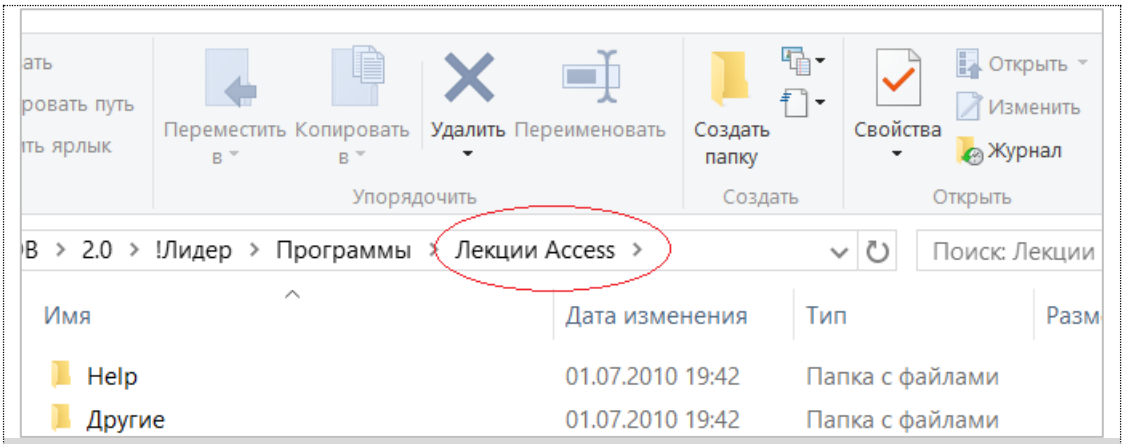
```
Public Function funCreateApp() As Boolean
    On Error GoTo 999 'Назначаем переход по ошибке
    appAccess = New Access.Application 'Создаем приложение
    appFolder = funGetAppFolder '<1> Находим папку

    funCreateDatabase "Новый калькулятор.mdb" '<2> Создаем базу данных
    funCreateTable "Калькулятор" '<3> Создаем таблицу
    funCreateQueries() '<4> Создаем запросы
    funCreateModule() '<5> Импортируем модули
    funCreateForm "Мой калькулятор"   '<6> Создаем форму
    funCreateReport "Отчет об итогах" '<7> Создаем отчет
    funCreateMacro "AutoExec" '<8> Создаем макрос
    funImportModule("Справка.bas", "Справка") '<9> Создаем справку
    funCreateMenu "Калькулятор" '<10> Создаем меню

    funCloseDatabase "Новый калькулятор.mdb" '<2> Закрываем программу
    funCreateApp = True 'Возвращаем значение
    Exit Function 'Выходим из программы
999:
    MsgBox Err.Description 'Сообщаем об ошибке
    Err.Clear() 'Очищаем поток от ошибок
End Function
```

Папка приложения

Профессионально выполненная программа всегда имеет внешние файлы, например basic-программы, рисунки *.bmp. Чтобы к ним обеспечить быстрый доступ, напишем функцию,

которая будет определять папку, где установлена основная программа.

Применяется эта программа для назначения данных глобальной переменной appFolder



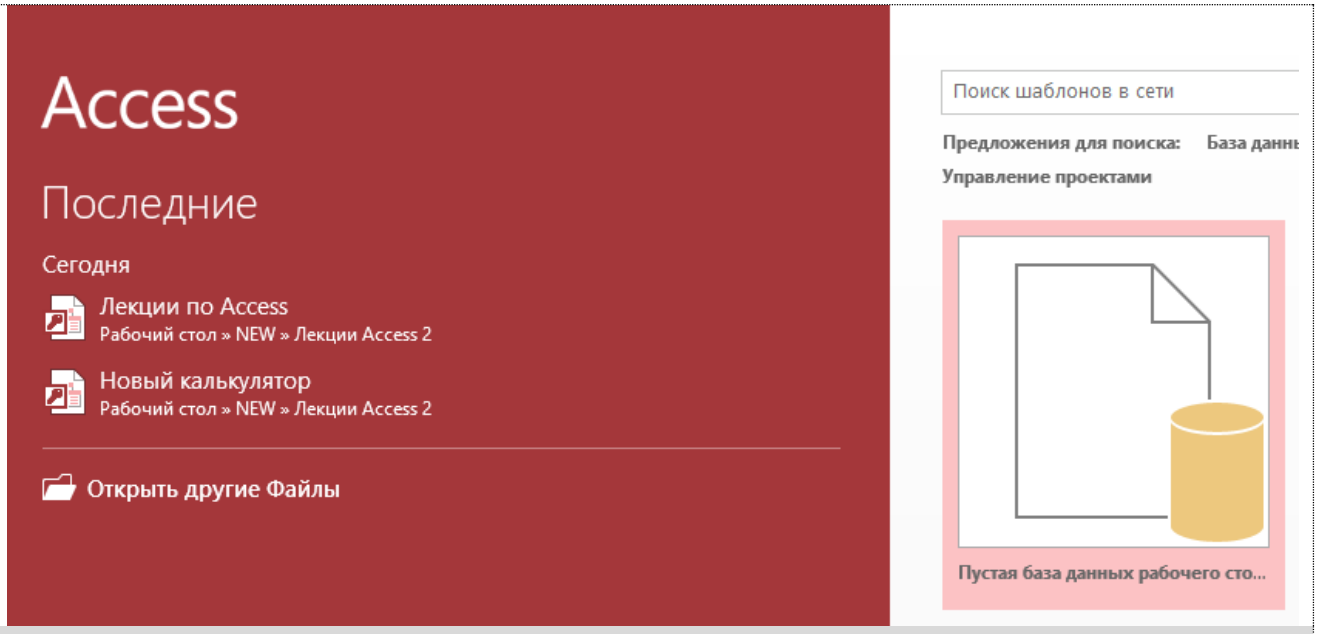
На этом снимке отображается папка программы, название которой мы хотим получить

```
Public Function funGetAppFolder() As String
    Dim fs
    On Error GoTo 999  'Назначаем переход по ошибке
    fs = CreateObject("Scripting.FileSystemObject") 'Создаем файловую систему
    funGetAppFolder = fs.GetFile(CurrentDb.Name).ParentFolder 'Находим папку
    fs = Nothing 'Уничтожаем переменную
    Exit Function 'Выходим из программы
999:
    MsgBox Err.Description 'Сообщаем об ошибке
    Err.Clear() 'Очищаем поток от ошибок

End Function
```

СОЗДАНИЕ БАЗЫ ДАННЫХ

В этой части статьи речь пойдет о создании базы данных средствами VBA. Сейчас восстанавливаем простые знания. далее будем их расширять, описывать формы, отчеты по разным базам данных.

И так после подготовительных действий, Вы можете приступить к созданию базы данных в файловой системе. Саму базу данных «Новый калькулятор.mdb» (переменная strMDB) создим программным методом, используя встроенный язык программирования Visual Basic For Application.



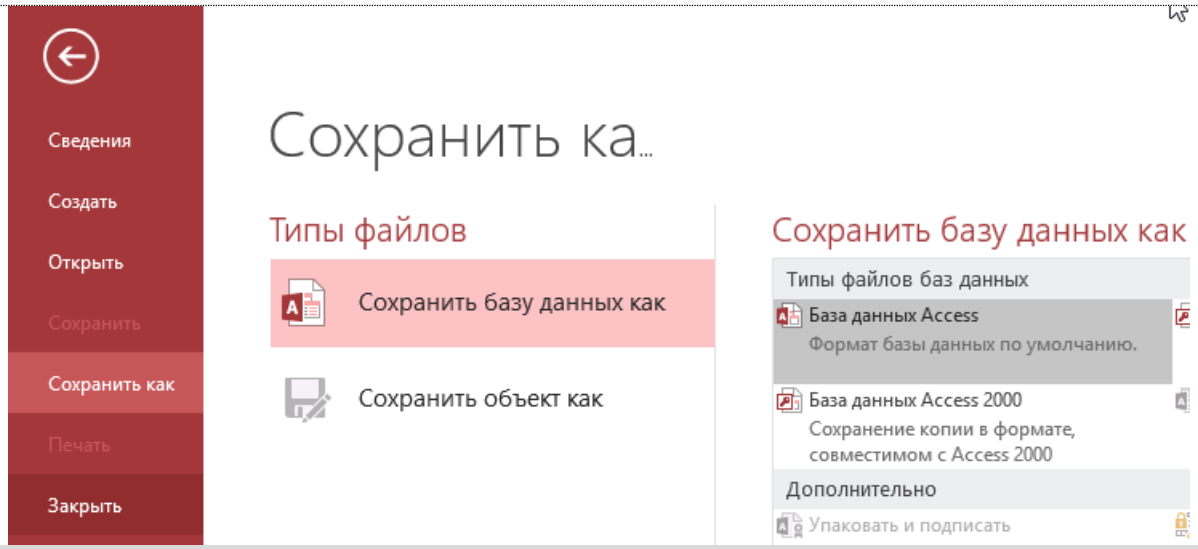
На этой форме показано как создать базу данных через интерфейс Microsoft Access 2016. Вид мастера для Access 2000 смотрите в примерах по калькулятору. Вообще у разных версий Access меняется только внешний вид форм. Найти нужную кнопку бывает очень сложно.

```
Public Function funCreateDatabase(strMDB As String) As Boolean
    Dim sFullPath As String
    On Error GoTo 999  'Назначаем переход по ошибке
    funCreateDatabase = False 'Возвращаем результат при ошибке

    sFullPath = appFolder & "\" & strMDB 'Полное имя файла
    If Dir(sFullPath) <> "" Then Kill sFullPath 'Уничтожаем старую базу данных
    DBEngine.CreateDatabase(sFullPath, dbLangCyrillic) 'Создаем базу данных
    appAccess.OpenCurrentDatabase sFullPath 'Открываем базу данных
    funInitReferences() '<2> Настраиваем ссылки базы данных
    funInitStartupProperties() '<2> Настраиваем стартовые свойства

    funCreateDatabase = True 'Возвращаем результат
    Exit Function 'Выходим из программы
999:
    MsgBox Err.Description 'Сообщаем об ошибке
    Err.Clear() 'Очищаем поток от ошибок
End Function
```

Закрытие базы данных

Если мы открыли базу данных, то ее необходимо закрыть. Это хороший способ программирования, хотя при закрытии окна Access (appAccess.Quit) база данных закрывается автоматически.

В этом примере база данных «Новый калькулятор.mdb» (переменная strMDB) будет закрыта, все ее объекты сохранены, а потом произойдет сжатие данных.



На этом снимке экрана показано, как закрыть базу данных через меню. Выбран пункт меню Сохранить как, если нажать меню Закрыть (Quit) база данных закроется.

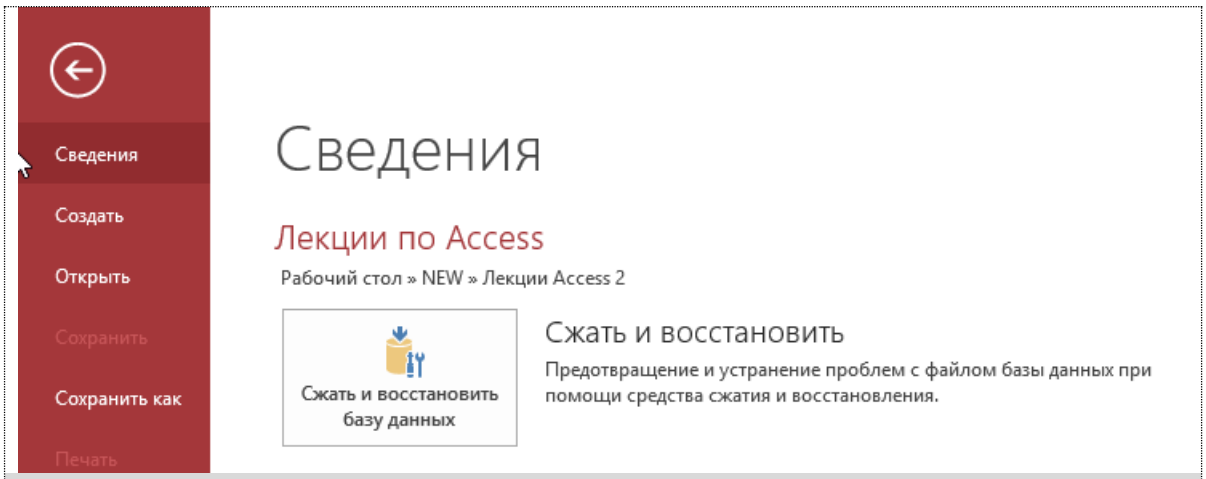
```
Public Function funCloseDatabase(strMDB As String) As Boolean
    On Error GoTo 999  'Назначаем переход по ошибке
    funCloseDatabase = False 'Возвращаем результат при ошибке

    appAccess.CloseCurrentDatabase() 'Закрываем базу данных
    appAccess.Quit acQuitSaveAll 'Выходим с сохранением
    funCompactDatabase strMDB '<2> Сжимаем базу данных

    funCloseDatabase = True 'Возвращаем результат
    Exit Function 'Выходим из программы
999:
    MsgBox Err.Description 'Сообщаем об ошибке
    Err.Clear() 'Очищаем поток от ошибок
End Function
```

Сжатие базы данных

При работе с базой данных Microsoft Access у Вас постепенно будет расти размер базы данных.

Это связано с тем, что, удаляя элементы или записи «физически» из базы они не удаляются, уничтожаются только ссылки.

Для уменьшения размера базы данных, напишем программу для сжатия ее файла. В качестве имени файла, передаваемого в программу, введем переменную StrMDB.



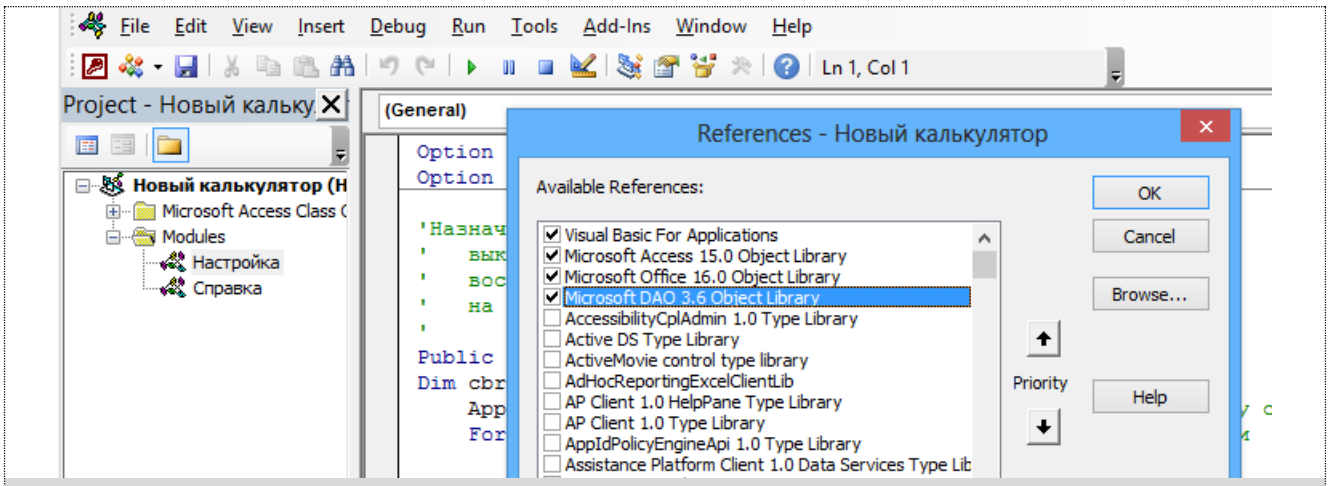
На этом снимке экрана показано, как сжать базу данных через меню Access 2016

```
Public Function funCompactDatabase(strMDB As String) As Boolean

    Dim tmpMDB As String, fs, sFullPath As String
    On Error GoTo 999  'Назначаем переход по ошибке
    funCompactDatabase = False 'Возвращаем результат при ошибке

    sFullPath = appFolder & "\" & strMDB 'Устанавливаем полное имя файла
    fs = CreateObject("Scripting.FileSystemObject") 'Создаем файловую систему
    tmpMDB = fs.GetTempName 'Получаем временный файл
    DBEngine.CompactDatabase(sFullPath, tmpMDB, dbLangCyrillic) 'Сжимаем базу дан
ных
    fs.CopyFile(tmpMDB, sFullPath, True) 'Переписываем файл
    Kill tmpMDB 'Удаляем временный файл
    fs = Nothing 'Уничтожаем объект файловой системы

    funCompactDatabase = True 'Возвращаем результат
    Exit Function 'Выходим из программы
999:
    MsgBox Err.Description 'Сообщаем об ошибке
    Err.Clear() 'Очищаем поток от ошибок
End Function
```

Настройка ссылок

При создании базы данных нам потребуются внешние функции, которые хранятся в специальных библиотеках, файлах с расширением DLL.

В этом примере показано, как создать программу, которая будет автоматически подключать нужные Вам библиотеки, а в качестве шаблона ссылок использует текущую базу данных. Например, ссылка на библиотеку Office хранится в файлеC:\Program Files\Microsoft Office\ Office\MSO9.DLL



На этом снимке экрана показано, как удалить или установить ссылку на библиотеку в интерфейсе редактора VBA для Access 2016 и для Access 2000.

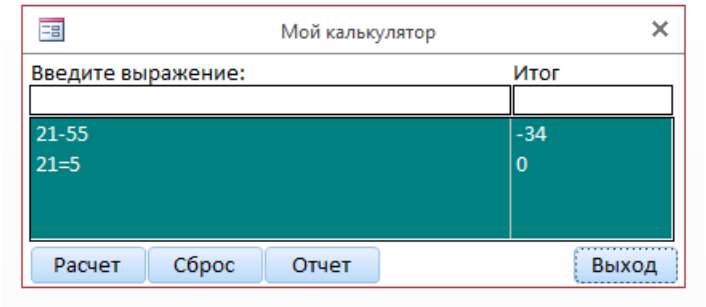
```
Public Function funInitReferences()
    Dim sFullPath As String, ref As Reference
    On Error GoTo 999  'Назначаем переход по ошибке

    With appAccess 'Выбираем базу данных
        For Each ref In .References
            If ref.BuiltIn = False Then _
                .References.Remove ref 'Удаляем лишние ссылки
        Next
    End With

    sFullPath = References("Office").FullPath
    appAccess.References.AddFromFile(sFullPath) 'Устанавливаем Office
    sFullPath = References("DAO").FullPath
    appAccess.References.AddFromFile(sFullPath) 'Устанавливаем DAO

    Exit Function 'Выходим из программы
999:
    MsgBox Err.Description 'Сообщаем об ошибке
    Err.Clear() 'Очищаем поток от ошибок
End Function
```

Настройка запуска

Когда Вы открываете базу данных созданную начинающим специалистом с помощью мастеров Access, то Вам сразу «бросается в глаза» некачественный интерфейс (лишние кнопки, меню, окна, разбросанные по экрану), который и дает Вам общую оценку его работе.

В этом примере давайте настроим базу данных так, чтобы в ней был начальный «приятный» интерфейс, а именно дадим базе данных заголовок, иконку, погасим ненужные окна и меню.

Совет. Изучите внимательно настройки базы данных в справочной системе Microsoft Access, введите их в свои программы, и тогда программное обеспечение, разрабатываемое Вами, будет иметь больший успех и уважение у пользователей.



На этом снимке экрана показан вид формы калькулятора в интерфейсе Microsoft Access 2016 без дополнительных настроек, т.е. нет верхнего меню.

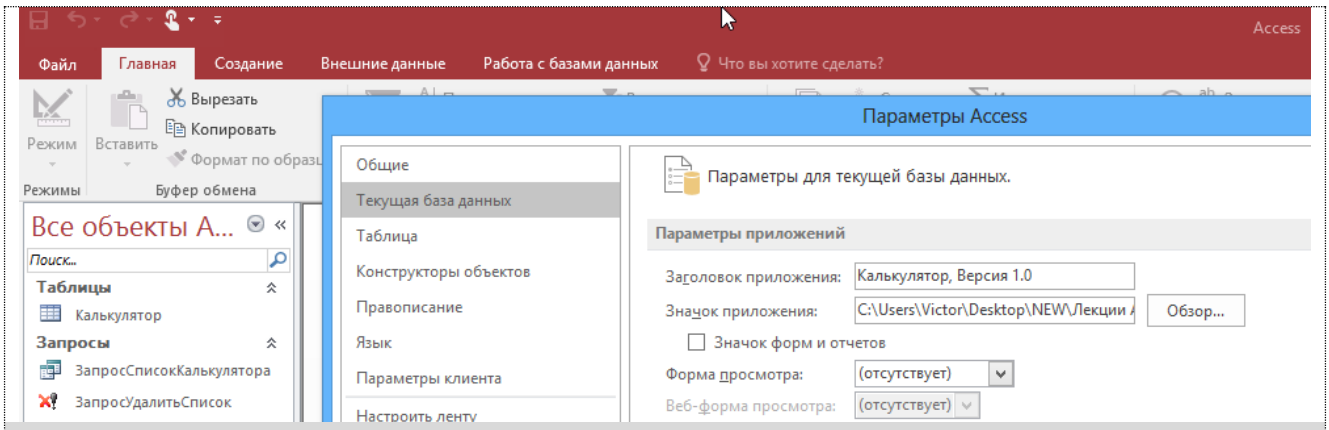
```
Public Function funInitStartupProperties()
    '<2> Изменяем заголовок
    dbChangeProperty("AppTitle", DB_Text, "Калькулятор, Версия 1.0")
    '<2> Изменяем иконку
    dbChangeProperty("AppIcon", DB_Text, appFolder & "\Рисунки\Лидер.ico")
    dbChangeProperty("StartupShowDBWindow", 1, False) '<2> Гасим окно БД
    dbChangeProperty("StartupShowStatusBar", 1, False) '<2> Гасим сообщения
    dbChangeProperty("AllowBuiltinToolbars", 1, True) '<2> Разрешаем меню
    dbChangeProperty("AllowFullMenus", 1, True) '<2> Разрешаем все меню
    dbChangeProperty("AllowToolbarChanges", 1, True) '<2> Разрешаем изменения
End Function
```

Свойство базы данных

В предыдущем разделе мы меняли заголовок базы данных, а здесь показан пример, как изменить свойство базы данных, используя встроенные функции Access.

Дополнение. Созданные свойства в отличие от переменных хранятся в базе данных постоянно, и Вам достаточно назначить их один раз, чтобы потом они уже использовались автоматически при открытии базы данных.

Например, если Вы погасили в каком-то сеансе с базой данных ее основное окно (StartupShowDBWindow = False), то в следующих сеансах работы оно не будет отображаться на экране.



На этом снимке экрана показана форма Параметры Access с помощью которой можно изменить настройки свойств редактируемой базы данных в Access 2016

```
'Название
'    Изменить/создать свойство базы данных
'Параметры:
'    strName - имя свойства (Description, Format ...)
'    varType - тип свойства (dbText, dbLong ...)
'    varValue - значение свойства
'
Function dbChangeProperty(strName As String, varType As Object, varValue As Objec
t) As Boolean
    Dim prp As Object, dbs As Database

    On Error GoTo 999 'Назначаем переход по ошибке
    dbChangeProperty = False 'Возвращаем результат при ошибке

    dbs = appAccess.CurrentDb 'Выбираем базу
    dbs.Properties(strName) = varValue 'Присваиваем значение

    dbChangeProperty = True 'Возвращаем результат
    Exit Function 'Выходим из программы
999:
    If Err = 3270 Then  'Свойство не найдено
        prp = dbs.CreateProperty(strName, varType, varValue) 'Создаем свойство
        dbs.Properties.Append prp 'Добавляем свойство
        Err.Clear() 'Очищаем поток от ошибки
        Resume Next 'Возвращаемся к следующему оператору
    End If
    Err.Clear() 'Очищаем поток от незнакомой ошибки
End Function
```

Проверка знаний

Теперь Вы научились создавать базу данных не с помощью мастера Access, а профессионально с помощью языка программирования Microsoft Visual Basic For Application.

Ваш результат Вы можете увидеть, нажав кнопку [Проверка N2] внизу экрана. Перед Вами отобразиться база данных, в которой пока не будет ни одного элемента.На следующем этапе программирования мы будем создавать таблицу «Калькулятор» с полями и необходимыми свойствами.

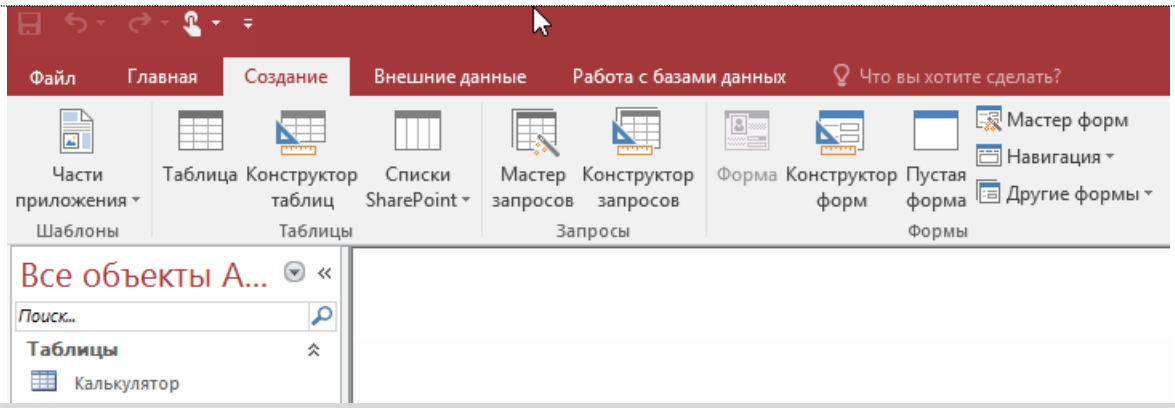
Примечание. Все примеры, указанные в книге, были протестированы на версии Access 2000 и 2016. Называется эта программа Лекции по Microsoft Access. В лекциях используются снимки экрана для Access 2000 для изучения разных видов интерфейса.

СОЗДАНИЕ ТАБЛИЦЫ

Работа с таблицами в Microsoft Access может осуществляться как мастерами, так и через язык программирование. В этой части книги рассказывается об этом способе разработки.

Вы научитесь создавать таблицу, проверять ее наличие в базе данных, создавать поля и изменять их свойства средствами VBA.

Наверное, Вы уже научились создавать таблицу с помощью мастера, этот способ указан слева. Теперь давайте попробуем создать ее программным способом. Иногда этот способ становится более предпочтительным, чем стандартный. В качестве переменной создаваемой таблицы будем использовать имя StrTable = «Калькулятор»



На этом снимке экрана показан интерфейс Microsoft Access 2016 для создания таблиц. Для этого в панели меню находится кнопка Конструктор таблиц

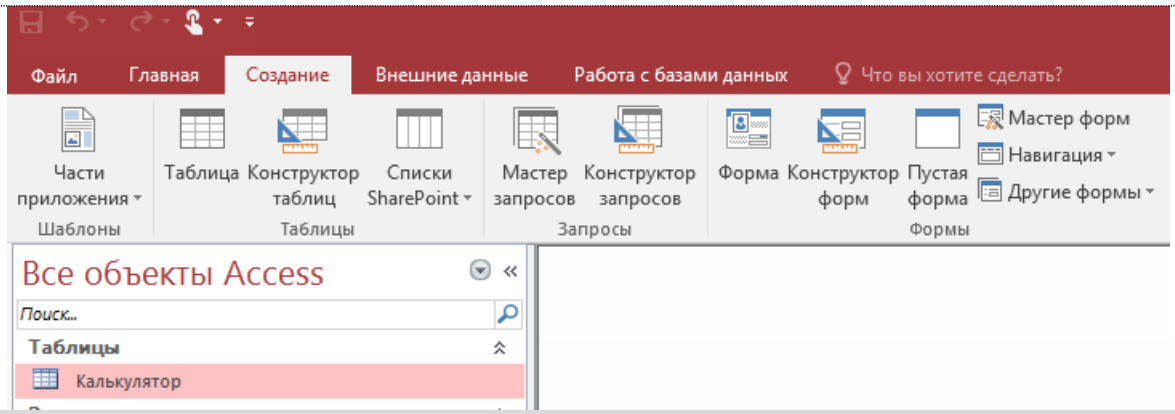
```
Public Function funCreateTable(strTable As String) As Boolean
    Dim dbs As Database, tdf As TableDef
    On Error GoTo 999 'Назначаем переход по ошибке
    funCreateTable = False 'Возвращаем результат при ошибке

    If funVerifyTable(strTable) = False Then 'Проверяем таблицу
        dbs = appAccess.CurrentDb 'Находим базу данных
        tdf = dbs.CreateTableDef(strTable)  'Создаем таблицу
        tdf.Fields.Append tdf.CreateField("Пункт", dbLong) 'Создаем 1 поле
        dbs.TableDefs.Append tdf 'Добавляем таблицу
```

```
        funCreateFields strTable '<3> Создаем поля
        funCreateTable = True 'Возвращаем результат
    End If

    Exit Function
999:
    MsgBox(Err.Description, vbCritical, "Создание таблицы") 'Сообщаем красиво об ошибке
    Err.Clear() 'Очищаем поток от ошибок
End Function
```

Проверка таблицы

> Очень часто при работе с базой данных Вам надо проверить наличие в базе данных объекта. На данном примере указан способ проверки таблицы в базе данных.

Если объекта нет, то случается ошибка и программа очищает поток от ошибок, возвращая значение False.



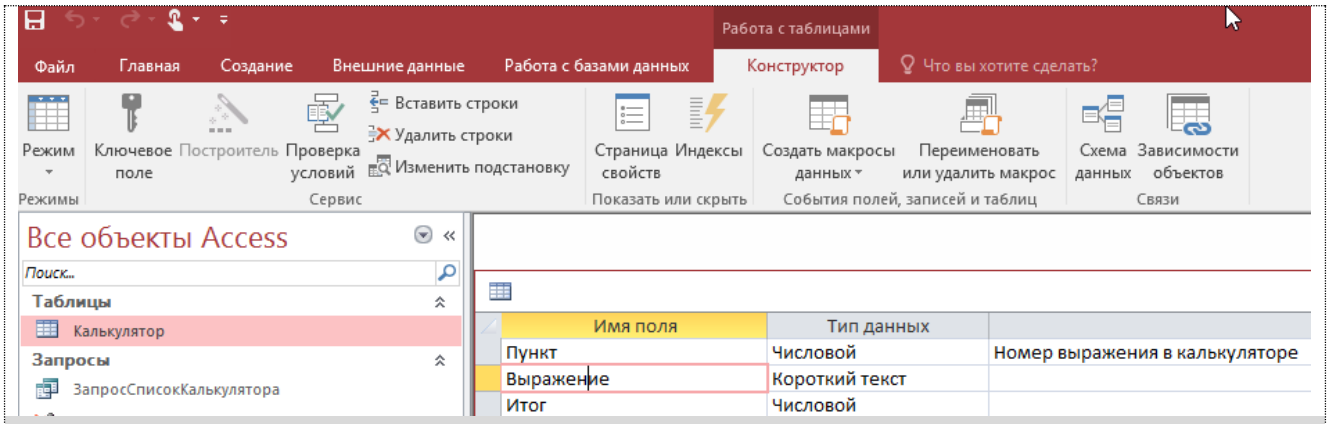
На этом снимке экрана в Access 2016 отображается таблица Калькулятор. Помните, что системные и другие таблицы в разных базах данных могут быть скрыты от просмотра.

```
Public Function funVerifyTable(strTable As String) As Boolean

    Dim tdf As TableDef
    On Error GoTo 999  'Назначаем переход по ошибке
    funVerifyTable = False 'Возвращаем результат при ошибке
    tdf = appAccess.CurrentDb.TableDefs(strTable) 'Находим объект
    If (tdf Is Nothing) = False Then funVerifyTable = True 'Проверяем объект
    tdf = Nothing 'Уничтожаем переменную

    Exit Function 'Выходим из программы
999:
    Err.Clear() 'Очищаем поток от ошибок
End Function
```

Создание полей таблицы

Создание полей с помощью программы является важным элементом программирования баз данных.

Так как в складском и бухгалтерском учете очень сложно учесть все поля таблиц из-за часто меняющихся требований к учету, естественно Вы должны предусмотреть в своих программах изменение базы данных различных версий по шаблону (метод называется репликацией). В данном примере в таблицу «Калькулятор», переменная strTable, добавляются два новых поля Выражение иИтог, а также изменяются свойства поля Пункт



На этом снимке экрана таблица Калькулятор в Access 2016 находится в режиме проектирования полей. Мы видим три поля, тип данных и описание.

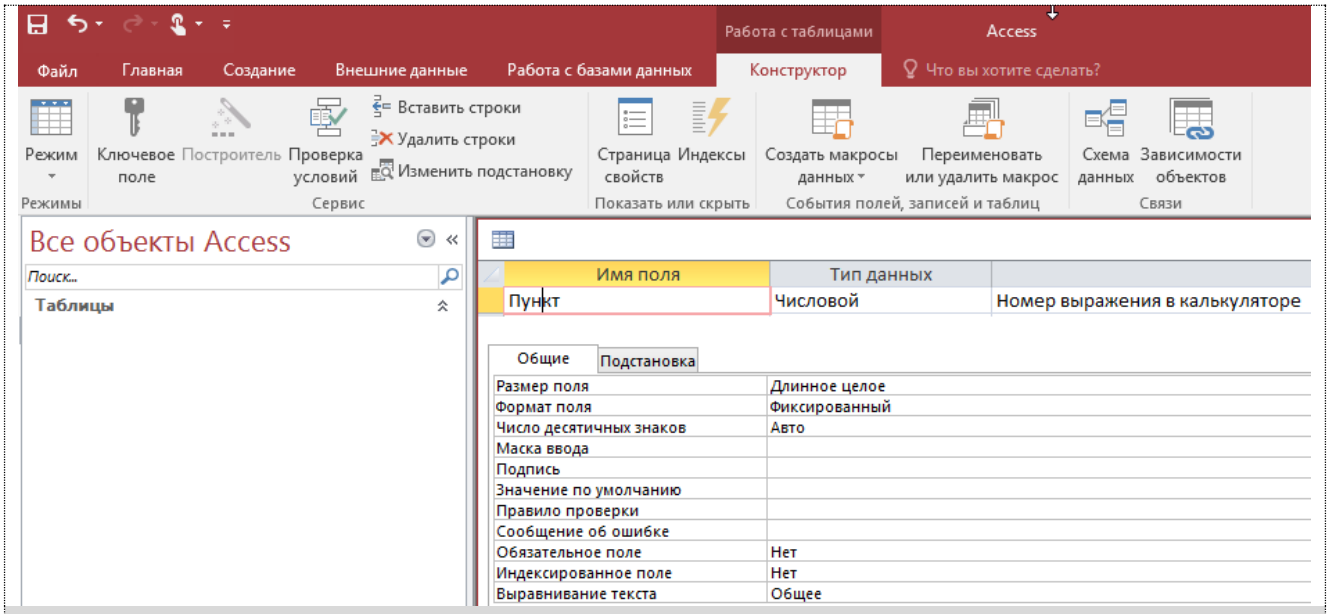
```
Public Function funCreateFields(strTable As String) As Boolean
    Dim dbs As Database, tdf As TableDef, fld As Field
    On Error GoTo 999 'Назначаем переход по ошибке
    funCreateFields = False 'Возвращаем результат при ошибке

    dbs = appAccess.CurrentDb 'Определяем базу данных
    tdf = dbs.TableDefs(strTable) 'Находим таблицу
    With tdf 'Выбираем таблицу для изменения
        .Fields.Append.CreateField("Выражение", dbText, 75) 'Создаем поле Выражение
        .Fields.Append.CreateField("Итог", dbDouble) 'Создаем поле Итог
    End With

    fld = tdf.Fields("Пункт") 'Изменяем свойства поля "Пункт"
    funChangeProperty(fld, "Description", dbText, "Номер выражения в калькуляторе") 'Изм
еняем описание
    funChangeProperty(fld, "Format", dbText, "Fixed") 'Назначаем фиксированный формат
    funChangeProperty(fld, "DecimalPlaced", dbByte, 0) 'Отключаем десятичные знаки

    fld = Nothing 'Уничтожаем переменную поля
    tdf = Nothing 'Уничтожаем переменную таблицы

    funCreateFields = True 'Возвращаем результат
    Exit Function 'Выходим из программы
999:
    MsgBox(Err.Description, vbCritical, "Создание таблицы") 'Сообщаем об ошибке
    Err.Clear() 'Очищаем поток от ошибок
End Function
```

Изменение свойств таблицы

Когда Вы создаете поле программным методом, некоторые его свойства, например, Описание (Description) не создаются автоматически.

Вам необходимо научиться изменять свойства полей базы данных, а в случае необходимости добавлять их в базу данных.

На этом примере показано, как изменить значение свойства поля базы данных. В данной программе также предусмотрено, что если свойство отсутствует, то оно будет добавлено в базу данных.



У каждого поля кроме его имени и типа данных есть дополнительные свойства. На этой форме в Access 2016 показано, что их можно увидеть в конструкторе таблицы

```
'Параметры:
'    fld - поле в таблице (Объект Field)
'    strName - имя свойства (Description, Format ...)
'    varType - тип свойства (dbText, dbLong ...)
'    varValue - значение свойства
'
Function funChangeProperty(fld As Field, strName As String, varType
                As Variant, varValue As Variant) As Boolean
    Dim prp As Object
    On Error GoTo 999 'Назначаем переход по ошибке
    funChangeProperty = False 'Возвращаем результат при ошибке

    fld.Properties(strName) = varValue 'Присваиваем значение полю

    funChangeProperty = True 'Возвращаем результат
    Exit Function 'Выходим из программы
999:
    If Err = 3270 Then  'Свойство не найдено
```

```
        prp = fld.CreateProperty(strName, varType, varValue) 'Создаем свойство
        fld.Properties.Append prp 'Добавляем свойство
        Err.Clear() 'Очищаем поток от ошибки
        Resume Next 'Возвращаемся к следующему оператору
    End If
    Err.Clear() 'Очищаем от незнакомой ошибки
End Function
```

Проверка знаний

Теперь Вы научились создавать таблицы и поля базы данных, а также изменять из свойства.

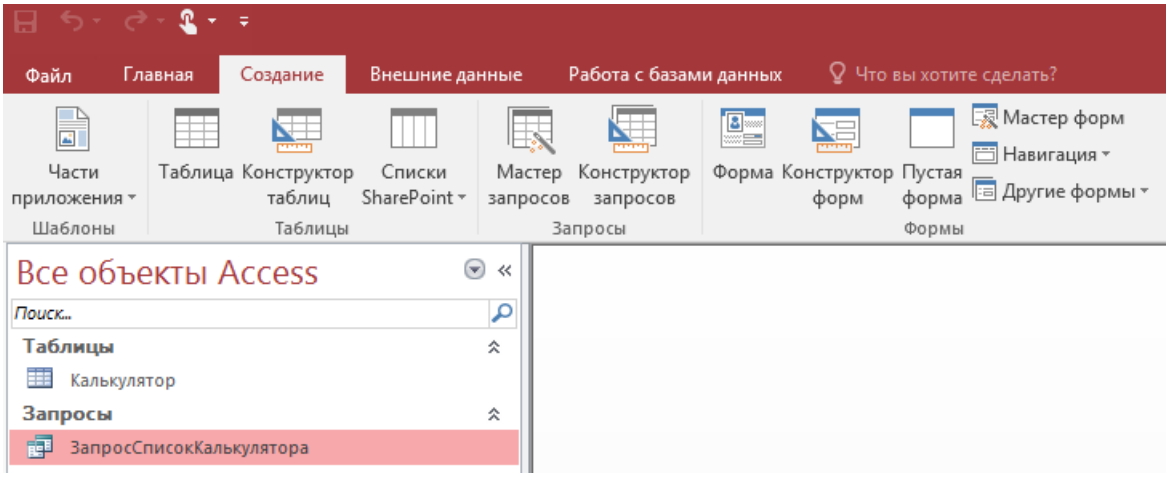
Ваш результат обучения Вы можете увидеть, нажав кнопку [Проверка N3] внизу экрана. Перед Вами отобразится база данных, в которой будет пустая таблица «Калькулятор». На следующем этапе программирования мы будем создавать запросы SQL, которые нужны для управления формой калькулятора. Итак для изучения следующей лекции нажмите кнопку Вперед>

СОЗДАНИЕ ЗАПРОСОВ

Запросы SQL к базе данных являются важным этапом проектирования. Практически нет ни одного случая, чтобы разработчики их не использовали.

Они нужны для фильтрации и сортировки данных, для редактирования и удаления записей. Часто используют запросы и в коде программы. Поэтому изучите эти простые примеры для работы с записями базы данных через VBA

В этом примере мы создадим 2 запроса SQL и сохраним их в базе данных. Первый запрос будет создан для выборки некоторых значений из таблицы, а второй для удаления значений из списка калькулятора.



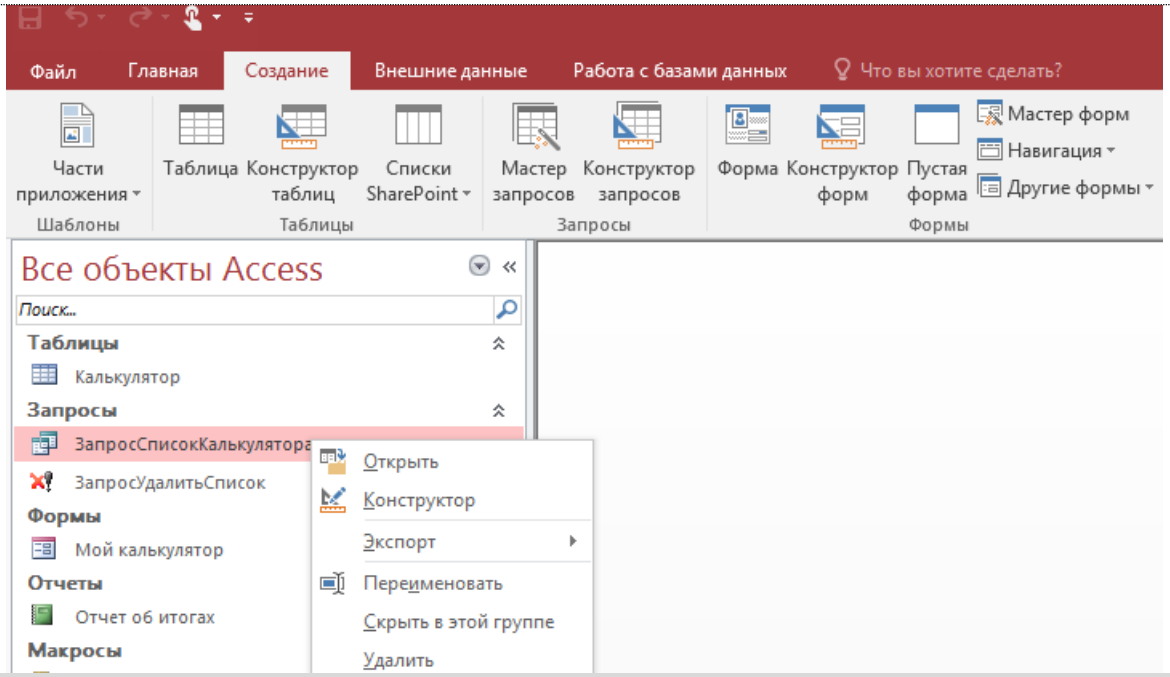
На этой форме в Access 2016 показано, как можно выбрать запрос и его изменить. Для этого используйте иконку Конструктор запросов.

```
Public Function funCreateQueries() As Boolean
    Dim dbs As Database, sSQL As String
    On Error GoTo 999 'Переходим по ошибке
    funCreateQueries = False  'Возвращаем результат при ошибке

    subDeleteQuery "ЗапросСписокКалькулятора" 'Удяляем старый запрос
    subDeleteQuery "ЗапросУдалитьСписок" 'Удяляем старый запрос

    dbs = appAccess.CurrentDb 'Выбираем базу данных
    With dbs
        sSQL = "SELECT Выражение, Итог FROM Калькулятор ORDER BY " & _
                       "Пункт DESC;"
        .CreateQueryDef("ЗапросСписокКалькулятора", sSQL) 'запрос на выборку

        sSQL = "DELETE Калькулятор.* FROM Калькулятор;"
        .CreateQueryDef("ЗапросУдалитьСписок", sSQL) 'запрос на удаление
    End With

    funCreateQueries = True 'Возвращаем результат
    Exit Function 'Выходим из программы
999:
    MsgBox Err.Description 'Сообщаем об ошибке
    Err.Clear() 'Очищаем поток от ошибок
End Function
```

Удаление запроса

Иногда при разработке базы данных Вам необходимо будет удалить запрос из программы. Посмотрите как это можно сделать мастером и через VBA

StrQueryName – это переменная, которая содержит имя запроса.



Запрос из базы данных можно удалить, используя контекстное меню. Сначала выделяем запрос и правой кнопкой вызываем меню.

```
Public Sub subDeleteQuery(strQueryName As String)
    On Error GoTo 999 'Включаем обработчик ошибок
    appAccess.DoCmd.DeleteObject(acQuery, strQueryName)  'Удаляем запрос
    Exit Sub 'Выходим из программы
999:
    Err.Clear() 'Очищаем поток от ошибок
     End Sub
```

Проверка знаний

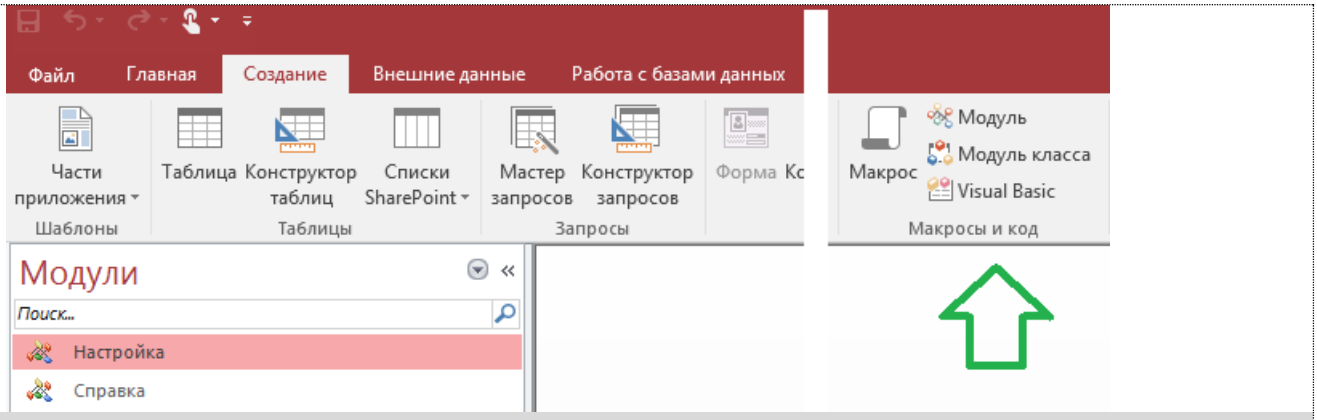
Теперь Вы научились создавать таблицы и запросы SQL, используя средства программирования в базе данных Microsoft Access.

Для закрепления материала и проверки ваших знаний нажмите кнопку [Проверка N4], для продолжения учебы и изучения импорта из внешних файлов нажмите кнопку Вперед>

СОЗДАНИЕ МОДУЛЯ

Под модулем в Microsoft Access понимается набор программ, написанных на языке Visual Basic.

Когда Вы разрабатываете базу данных, Вам потребуются общие программы, которые можно использовать при работе с формами или другими элементами базы данных. Давайте напишем небольшую программу, которая создает модуль «Настройка» путем импорта готовых Basic-программ.



На этом снимке базы данных показано меню для упрвления моделем. К сожалению кнопки очень небольшие и их невидно. Используйте Alt+F11 для открытия модуля

```
    Public Function funCreateModule() As Boolean
        On Error GoTo 999 'Переход по ошибке
        funImportModule("Настройка.bas", "Настройка") '<5> Добавляем программу
        Exit Function 'Выход из программы
999:
        MsgBox Err.Description 'Сообщаем об ошибке
        Err.Clear() 'Очищаем поток от ошибок
    End Function
```

Импорт программ модуля

Эта функция является продолжением предыдущих действий, описанных выше. С помощью нее мы сможем прочитать модуль «Настройка.bas» из папки «Программы» и разместить ее у себя в базе данных.

```
    'Параметры:
    '   strBasic - название файла «Настройка.bas»
    '   strModule - название модуля «Настройка»
    '
    Public Function funImportModule(strBasic As String, strModule As String) As Boolean
        Dim sFullPath As String
        On Error GoTo 999
        sFullPath = appFolder & "\Программы\" & strBasic 'Составляем файл
        If Dir(sFullPath) <> "" Then 'Проверяем файл
            With appAccess 'Указываем приложение
                .VBE.ActiveVBProject.VBComponents.Import sFullPath 'Импортируем программ
у
                .DoCmd.Save(acModule, strModule) 'Сохраняем программу
            End With
        Else
            MsgBox "Файл: " & sFullPath & " не существует!" 'Сообщаем об ошибке
        End If
        Exit Function 'Выходим из программы
999:
        MsgBox Err.Description 'Сообщаем об ошибке
        Err.Clear() 'Очищаем поток от ошибок
    End Function
```

Выключить панели инструментов

В предыдущем примере мы с Вами импортировали модуль «Настройка.bas». Далее дается описание программы, которая есть в этом модуле. Назначение ее - выключить все панели инструментов, чтобы они не мешали работе Калькулятора.

```
Public Sub subBarOff()
     Dim cbr As CommandBar
     Application.SetOption("Строка состояния", False) 'Гасим строку состояния
     For Each cbr In Application.CommandBars 'Проверяем все панели
         If cbr.Visible = True Then 'Проверяем отображение
             DoCmd.ShowToolbar(cbr.Name, acToolbarNo) 'Гасим панель
         End If
     Next cbr
 End Sub
```

Включить панель отчетов

В этом разделе опишем еще одну программу модуля «Настройка». Используется она для работы с отчетами. Назначение ее - включить панель отчетов, чтобы пользователь смог отпечатать список расчета калькулятора.

```
Public Sub subBarReport()
    subBarOff() 'Гасим все панели
    DoCmd.ShowToolbar("Строка меню", acToolbarYes) 'Отображаем
    DoCmd.ShowToolbar("Предварительный просмотр", acToolbarYes) 'Отображаем
End Sub
```

Проверка знаний

Теперь Вы научились создавать внешние функции, используя средства импорта Microsoft Access 2000. Для просмотра разработанных Вами объектов базы данных нажмите кнопку [Проверка N5]

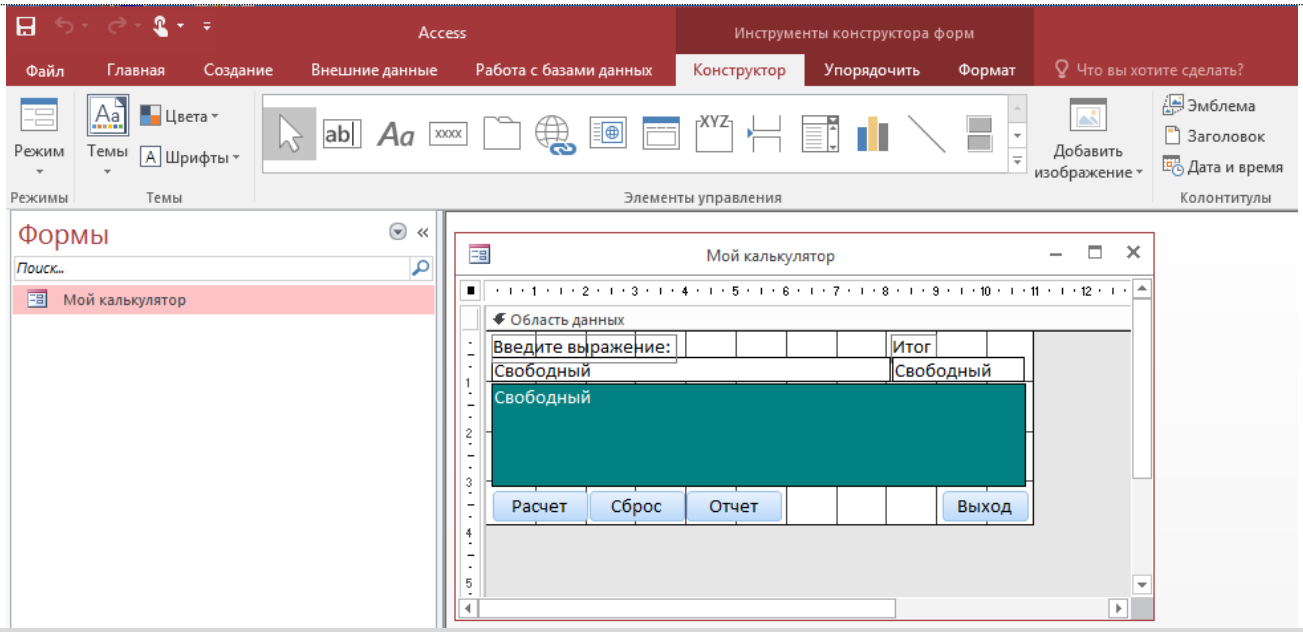
На следующем этапе проектирования мы будем создавать форму «Мой калькулятор» с запросами и программами, которые были разработаны в предыдущих лекциях. Этот этап создания базы данных самый сложный.

Итак, если Вы устали, то лучше отдохнуть 5 минут и 10 секунд. Если же Вам не терпится стать профессионалом и быстрее начать зарабатывать деньги разработкой баз данных, то нажмите кнопку Вперед>

СОЗДАНИЕ ФОРМЫ

> Самый сложный этап проектирования базы данных. Формы постоянно находятся в движении. Их переделывают, удаляют, изменяют и никто не бывает довольным текущим решением.

Создание форм в базе данных Access сложный и трудный этап программирования. Давайте напишем общую функцию, которая будет сама создавать нам форму и изменит ее настройки так, чтобы она выглядела красиво и с минимальным количеством объектов управления. Имя формы будет «Мой калькулятор» и запишем ее в переменную strForm.



Внешний вид панели элементов в режиме дизайнера Access 2016, т.е. тогда когда пользователь изменяет форму программы.

```
    Public Function funCreateForm(strForm As String) As Boolean
        Dim frm As Form

        On Error GoTo 999 'Переход по ошибке
        funCreateForm = False 'Значение возвращаемое по ошибке

        funDeleteForm strForm 'Удаляем старую форму
        frm = appAccess.CreateForm 'Создаем новую форму
        With frm 'Изменяем параметры формы
            .Caption = "Мой калькулятор" 'Вставляем заголовок
            .ScrollBars = 0 'Гасим полосы прокрутки
            .RecordSelectors = False 'Гасим область выделения
            .NavigationButtons = False 'Гасим кнопки перехода
            .DividingLines = False 'Гасим разделительные линии
            .AutoCenter = True 'Выравниваем форму по центру
            .BorderStyle = 3 'Устанавливаем диалоговую границу
            .Section(0).Height = 3.862 * appCM 'Изменяем высоту окна
            .Width = 10.926 * appCM 'Изменяем ширину окна
            .HasModule = True 'Разрешаем программы в форме
        End With

        funRestoreFormControls frm  'Создаем элементы формы
        funInsertFormModule frm 'Создаем модуль формы
        appAccess.DoCmd.Save, strForm 'Сохраняем форму
        appAccess.DoCmd.Close(acForm, strForm, acSaveYes) 'Закрываем форму

        funCreateForm = True 'Возвращаем результат
        Exit Function 'Выход из программы
999:
        MsgBox Err.Description 'Сообщаем об ошибке
        Err.Clear() 'Очищаем поток от ошибок
```

```
    End Function
```

Удаление формы

На этом этапе напишем программу для удаления формы strForm («Мой калькулятор»). Программа работает так, что если формы нет в базе данных, то программа выходит по ошибке через метку 999 и нам надо очистить поток от нее, чтобы в дальнейшем не было сообщений.

```
    Public Function funDeleteForm(strForm As String) As Boolean
        Dim frm As Form
        On Error GoTo 999 'Переход по ошибке
        funDeleteForm = False 'Возвращаемое значение при ошибке

        appAccess.DoCmd.DeleteObject(acForm, strForm) 'Удаляем форму

        funDeleteForm = True 'Возвращаем значение
        Exit Function
999:
        Err.Clear() 'Очищаем поток от ошибок
    End Function
```

Создание элементов формы

Теперь нам надо написать программу, которая создаст управляющие элементы, т.е. метки, кнопки и список формы, восстанавливая их из таблицы базы данных, например, «Калькулятор-форма». Этот метод необходимо изучить потому, что таким образом можно легко восстановить любой шаблон формы и в дальнейшем модифицировать его.

```
    Public Function funRestoreFormControls(frm As Form) As Boolean
        Dim ctl As Control, rst As Recordset, dbs As Database, i As Integer

        On Error GoTo 999 'Переход по ошибке
        funRestoreFormControls = False 'Возврат значения по ошибке

        dbs = CurrentDb 'Выбираем текущую базу данных
        rst = dbs.OpenRecordset("SELECT * FROM [Калькулятор-форма]")
        If rst.RecordCount = 0 Then  'Проверяем запрос
            rst.Close() 'Закрываем запрос
            Exit Function  'Выходим из программы
        End If

        With rst 'Создание элемента из запроса
            .MoveLast() 'Заполняем запрос данными
            .MoveFirst() 'Переходим на 1 запись
            For i = 0 To .RecordCount - 1 'Выполняем восстановление для каждой записи
                Select Case rst!ControlType
                    Case acCommandButton 'Создаем Кнопку
                        ctl = appAccess.CreateControl(frm.Name, rst!ControlType, , "", "
", rst!Left, rst!Top, rst!Width, rst!Height)
                        ctl.OnClick = "[Event Procedure]" 'Включаем обработку нажатия
                        ctl.Caption = rst!Caption 'Меняем название
                        ctl.ControlTipText = rst!ControlTipText 'Устанавливаем подсказку
                    Case acLabel 'Создаем Надпись
                        ctl = appAccess.CreateControl(frm.Name, rst!ControlType, , "", r
st!Caption, rst!Left, rst!Top, rst!Width, rst!Height)
```

```
                        ctl.BackColor = !BackColor 'Меняем фон
                    Case acTextBox 'Создаем Поле
                        ctl = appAccess.CreateControl(frm.Name, rst!ControlType, , "", "
", rst!Left, rst!Top, rst!Width, rst!Height)
                        ctl.SpecialEffect = 0 'Выбираем стандартное выделение
                        ctl.BorderColor = 0 'Выбираем черный цвет
                        ctl.BorderStyle = 1 'Выбираем обычную границу
                    Case acListBox 'Создаем Список
                        ctl = appAccess.CreateControl(frm.Name, rst!ControlType, , "", "
", rst!Left, rst!Top, rst!Width, rst!Height)
                        ctl.SpecialEffect = 0 'Выбираем стандартное выделение
                        ctl.BackColor = rst!BackColor 'Меняем фон списка
                        ctl.BorderColor = 0 'Выбираем стандартное выделение
                        ctl.ColumnCount = 2 'Устанавливаем число колонок
                        ctl.ColumnWidths = "7,932 см;1 см" 'Устанавливаем размер колонок
                        ctl.RowSource = "запросСписокКалькулятора" 'Выбираем запрос для
списка
                End Select
                ctl.Name = rst!Name 'Изменяем имя элемента
                ctl.ForeColor = !ForeColor 'Изменяем цвет символов
                rst.MoveNext() 'Переходим на следующую запись
            Next i
        End With
        rst.Close() 'Закрываем запрос

        funRestoreFormControls = True 'Возвращаем результат
        Exit Function 'Выходим из программы
999:
        MsgBox Err.Description 'Сообщаем об ошибке
        Err.Clear() 'Очищаем поток от ошибок
    End Function
```

Вставить в форму модуль

После создания формы, напишем программу, которая вставит «модуль программ» на Бэйсике в нашу форму. Это необходимо для того, чтобы обеспечить обработку событий формы, например, таких как открытие, загрузка, нажатие кнопок.

Далее в лекциях идет перечисление событий. Они отмечены на рисунках кнопкой.

```
Public Function funInsertFormModule(frm As Form) As Boolean
Dim s As String, mdl As Module
        On Error GoTo 999 'Переход по ошибке
        s = appFolder & "\Программы\Form_Мой калькулятор.bas"
        If Dir(s) <> "" Then 'Проверяем файл
            frm.Module.AddFromFile s 'Добавляем модуль
        End If
        Exit Function 'Выходи из программы
999:
        MsgBox Err.Description 'Сообщаем об ошибке
        Err.Clear() 'Очищаем поток от ошибок
    End Function
```

Открытие формы

Обработка события. Данная функция начинает работать при открытии формы. В ней мы размещаем информацию о предварительных настройках окна калькулятора, до его отображения на экране.

```
Private Sub Form_Open(Cancel As Integer)
    subBarOff() 'Выключить все меню и строку состояния

    'Настроить всплывающие подсказки кнопок
    Me.кнВыход.ControlTipText = "Выход из программы"
    Me.кнПечать.ControlTipText = "Печать отчета о расчетах"
    Me.кнРасчет.ControlTipText = "Расчет данных на калькуляторе"
    Me.кнСброс.ControlTipText = "Сбросить все вычисления"

    'Меняем настройки окна, установленные по умолчанию
    Me.Caption = "Мой калькулятор" 'Заголовок
    Me.ScrollBars = 0 'Полосы прокрутки
    Me.RecordSelectors = False 'Область выделения
    Me.NavigationButtons = False 'Кнопки перехода
    Me.DividingLines = False 'Разделительные линии

    'Меняем настройки списка задач
    Me.Список.BackColor = 8421376 'Цвет фона символов
    Me.Список.ForeColor = RGB(255, 255, 255) 'Цвет символов

    'Устанавливаем события
    Me.Выражение.AfterUpdate = "[Event Procedure]" 'После обновления
    Me.кнВыход.OnClick = "[Event Procedure]" 'Нажатие кнопки
    Me.кнПечать.OnClick = "[Event Procedure]" 'Нажатие кнопки
    Me.кнРасчет.OnClick = "[Event Procedure]" 'Нажатие кнопки
    Me.кнСброс.OnClick = "[Event Procedure]" 'Нажатие кнопки

    'Настраиваем параметры приложения
    DoCmd.RunCommand acCmdAppMaximize 'Увеличиваем приложение
    On Error Resume Next 'Отключаем диагностику ошибок меню
    Me.MenuBar = "Калькулятор" 'Устанавливаем меню
    Err.Clear() 'Очищаем поток от ошибки
End Sub
```

Расчет выражения

Обработка события. Эта программа нужна для того, чтобы после того, как пользователь введет свое выражение и нажмет кнопку «Enter», произошли необходимые расчеты, и список вычислений калькулятора обновился.

```
   Private Sub Выражение_AfterUpdate()
        Dim dbs As Database
        On Error GoTo 999  'Назначаем переход по ошибке
        If (Me.Выражение = "") Or _
            IsNull(Me.Выражение) Then Exit Sub 'Проверяем выражение
        dbs = CurrentDb 'Выбираем базу данных
        Me.Итог = Eval(Me.Выражение) 'Рассчитываем выражение
        subСохранитьВыражение() 'Сохраняем выражение в списке
        Me.Список.Requery() 'Пересматриваем список

        Me.Выражение = "" 'Сбрасываем выражение
        Me.кнПечать.SetFocus() 'Помещаем курсор для печати
```

```
        dbs = Nothing 'Обнуляем переменную
        Exit Sub 'Выходим из программы
999:
        Err.Clear() 'Очищаем поток от ошибок
        Me.Итог = "Ошибка" 'Сообщаем об ошибке
        Me.Выражение.SetFocus() 'Подготавливаем новый ввод данных
    End Sub
```

Выход из программы

Обработка события. Пользователю Вашей программы надо обеспечить выход из приложения. Для этого напишем событие так, чтобы после нажатия кнопки Выход, программа запросила о выходе из калькулятора, и в случае утвердительного ответа закрыла базу данных.

```
Private Sub кнВыход_Click()
    If MsgBox("Закрыть калькулятор?", vbInformation + vbOKCancel, "Калькулятор") = vbOK
Then
        Application.Quit acQuitSaveAll 'Выходим из приложения
    End If
End Sub
```

Открыть отчет

Обработка события. После проведенных вычислений нам может потребоваться отпечатать отчет. Для этого мы и создали кнопку Печать. Теперь мы должны написать событие, чтобы при нажатии кнопки отобразился документ "Отчет об итогах".

```
Private Sub кнПечать_Click()
     On Error Resume Next 'Отключаем диагностику ошибок
     DoCmd.OpenReport("Отчет об итогах", acViewPreview) 'Открываем отчет
     Err.Clear() 'В случае отсутствия данных очищаем поток ошибок
 End Sub
```

Рассчитать данные

Обработка событий. В нашем калькуляторе есть кнопка Расчет, вставим в программу событие, которое при нажатии данной кнопки будет проводить вычисление введенного в калькулятор выражения.

```
Private Sub кнРасчет_Click()
    If (Me.Выражение = "") Or IsNull(Me.Выражение) Then 'Проверяем выражение
        MsgBox("Введите выражение!", _
            vbInformation, Me.Name) 'Сообщаем об отсутствии данных
    Else
        Выражение_AfterUpdate()  'Обновляем выражение
    End If
End Sub
```

Сохранить выражение

Если мы хотим печатать данные о вычислениях, то надо уметь сохранять итоги расчетов калькулятора в таблице. Напишем функцию, которая будет это делать.

```
    Private Sub subСохранитьВыражение()
        Dim dbs As Database, rst As Recordset
        On Error GoTo 999  'Назначаем переход по ошибке
        dbs = CurrentDb 'Выбираем базу данных
        rst = dbs.OpenRecordset("SELECT * FROM [Калькулятор]")
        With rst
            If .RecordCount > 0 Then
                .MoveLast() '! Заполнить запрос SQL данными
            End If
            .AddNew() 'Добавить новую запись
            !Пункт = .RecordCount + 1 'Порядковый номер записи
            !Выражение = Me.Выражение 'Вычисляемое выражение
            !Итог = Me.Итог 'Итоговое вычисление
            .Update() '! Запомнить вычисления
        End With
        rst.Close() 'Закрываем запрос
        dbs = Nothing 'Уничтожаем переменную
        Exit Sub 'Выходим из программы
999:
        MsgBox Err.Description 'Сообщаем об ошибке
        Err.Clear() 'Очищаем поток от ошибок
    End Sub
```

Сбросить все вычисления

Обработка событий. И так при работе за калькулятором, Вам может потребоваться сбросить все вычисления, для этого напишем событие, которое будет срабатывать при нажатии кнопки Сброс.

```
    Private Sub кнСброс_Click()
        Dim dbs As Database
        On Error GoTo 999  'Назначаем переход по ошибке
        dbs = CurrentDb 'Выбираем базу данных
        dbs.Execute("DELETE * FROM [Калькулятор]", dbFailOnError) 'Удаляем расчеты
        Me.Итог = "" 'Удаляем итог
        Me.Выражение = "" 'Очищаем выражение
        Me.Список.Requery() 'Обновляем список
        Exit Sub 'Выходим из программы
999:
        MsgBox Err.Description 'Сообщаем об ошибке
        Err.Clear() 'Очищаем поток от ошибок
    End Sub
```

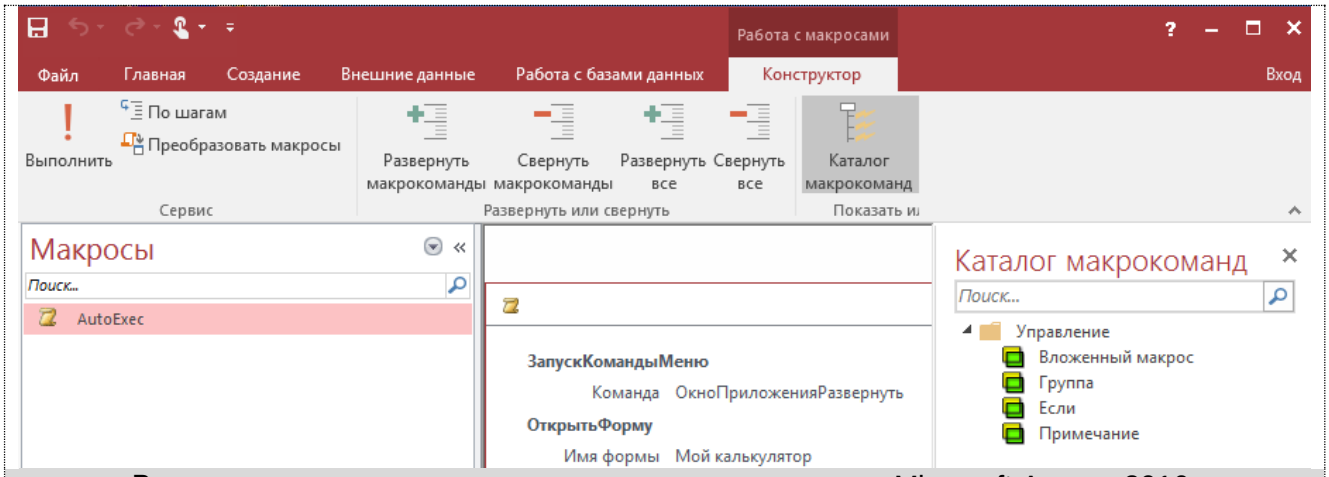
Проверка знаний

Теперь Вы научились создавать формы в Access и можете себя считать в какой-то степени профессионалом. Для просмотра Вашей работы нажмите кнопку [Проверка N6], для продолжения учебы, нажмите кнопку Вперед>

Внимание! Открыв форму калькулятора в режиме просмотра, у Вас не будет работать кнопка Отчет, т.к. Вы еще не прослушали раздел лекций по разработке отчетов.

СОЗДАНИЕ МАКРОСА

При открытии любой базы данных автоматически запускается макрос AutoExec, вспомните командный файл DOS «autoexec.bat». В этот макрос для примера можно вставить программы Access, например, команду увеличения приложения до максимального размера и команду открытия формы «Мой калькулятор».



Вот так примерно выглядит редактирование макроса в Microsoft Access 2016.

```
Public Function funCreateMacro(strMacro As String) As Boolean
        Dim frm As Form
        On Error GoTo 999 'Переход по ошибке
        funCreateMacro = False 'Возвращаем значение при ошибке

        'Импортируем макрос
        appAccess.DoCmd.TransferDatabase(acImport, "Microsoft Access", appFolder & "\Ста
рый калькулятор.mdb", acMacro, strMacro, strMacro)

        funCreateMacro = True 'Возвращаем значение
        Exit Function 'Выходим из программы
999:
        MsgBox Err.Description 'Сообщаем об ошибке
        Err.Clear() 'Очищаем поток от ошибок
    End Function
```

Проверка знаний

Теперь Вы научились создавать отчеты в Access и можете теперь создавать практически готовые программы. Для просмотра вашей формы нажмите кнопку [Проверка N7], для продолжения учебы, нажмите кнопку Вперед>

Проверка знаний

Теперь Вы научились создавать макросы в Access. Для просмотра объектов базы данных нажмите кнопку [Проверка N8], для продолжения учебы, нажмите кнопку Вперед>

СОЗДАНИЕ СПРАВКИ

Если Вы решили оставить общую справку по Access, которая вызывается нажатием F1, то можно написать программу, отображающую Ваш собственный файл: «Калькулятор.chm» при нажатии кнопки меню. Команда Shell выглядит так: hh.exe -mapid 103 Калькулятор.chm

```
   Public Function funCreateNewHelp()
        Dim fs, s As String, hlp As String
        On Error GoTo 999
        fs = CreateObject("Scripting.FileSystemObject") 'Создаем файловую систему
        s = fs.GetSpecialFolder(0) & "\hh.exe"  'Составляем hh.exe файл
        If Dir(s) <> "" Then 'Проверяем exe-файл
            hlp = fs.GetFile(CurrentDb.Name).ParentFolder & "\Калькулятор.chm" 'Находим
справку
            If Dir(hlp) <> "" Then 'Проверяем файл справки
                s = """" & s & """ -
mapid " & 103 & " """ & hlp & """" 'Составляем команду
                Shell(s, vbMaximizedFocus) 'Запускаем справку
            End If
        End If
        Exit Function 'Выходим из программы
999:
        Err.Clear() 'Очищаем поток от ошибок
    End Function
```

Создание помощника

В Access 2016 не работает. Ваша программа будет эффектно выглядеть, если Вы научитесь программировать помощника Access. Ниже написана функция, которая создает на экране сообщения в виде всплывающих ответов от «котенка».

Помните, что если Вы выключили помощника, то он при нажатии меню высвечиваться не будет, его необходимо включить снова.

```
Public Function funCreateAssistant()
       Dim bln As Balloon, s As String
       With Application.Assistant 'Выбираем помошника
           .Visible = True 'Вывод на экран
           .Animation = msoAnimationAppear 'Включаем анимацию
           bln = .NewBalloon 'Создаем сообщение
           With bln
               .Heading = "Калькулятор позволяет" 'Устанавливаем заголовок
               .Labels(1).Text = "Вводить выражения" 'Включаем меню 1
               .Labels(2).Text = "Вводить формулы" 'Включаем меню 2
               .BalloonType = msoBalloonTypeButtons 'Устанавливаем тип меню
               .Button = msoButtonSetNone 'Отключаем клавиши помощника
               Select Case .Show 'Отображаем помощника
                   Case 1 : s = "23-456/35" 'Назначаем 1 сообщение
                   Case 2 : s = "sin(0.5)"  'Назначаем 2 сообщение
               End Select 'Конец вариантов
           End With
           bln = .NewBalloon 'Создаем нового помощника
           With bln
               .Heading = "Пример выражения" 'Устанавливаем заголовок
```

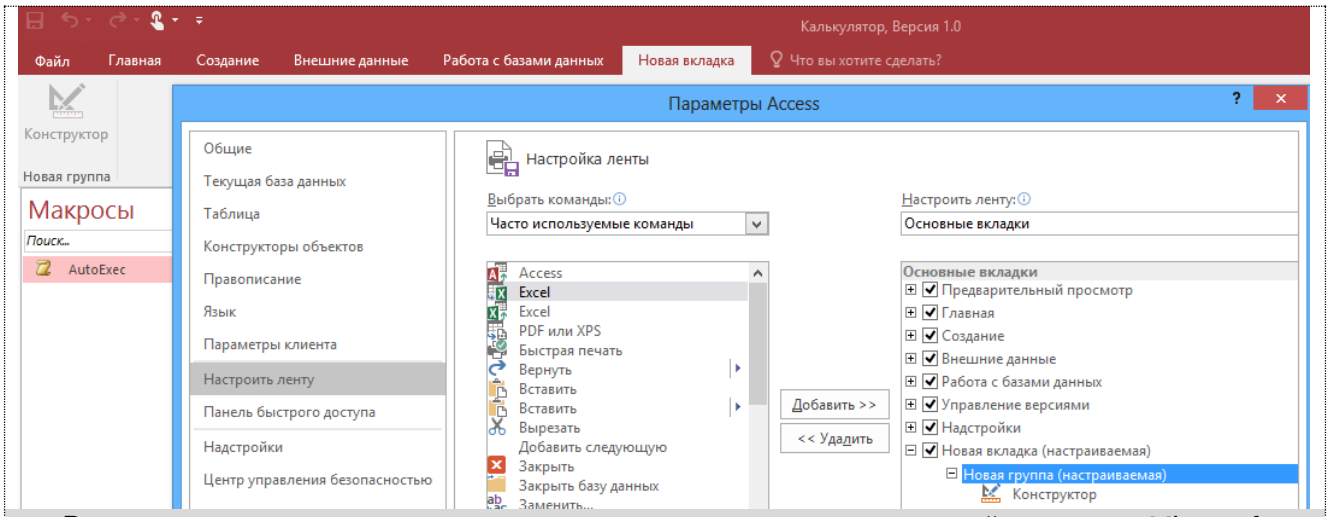
```
            .Text = s 'Выводим назначенный текст, см. выше
            .Button = msoButtonSetOK 'Отображаем клавишу OK
            .Show() 'Показываем помощника
        End With
    End With
End Function
```

Проверка знаний

Теперь Вы научились создавать различные справочные системы в Access и можете комментировать свою программу. Для просмотра объектов базы данных нажмите кнопку [Проверка N9], для продолжения учебы, нажмите кнопку Вперед>

СОЗДАНИЕ ПАНЕЛИ МЕНЮ

Большинство коммерческих программ имеет в верхней части приложения свою панель элементов, на которой есть разные кнопки. Давайте напишем программу, которая создает панель меню без кнопок.



Вот так примерно выглядит редактирование кнопок меню для новой вкладки в Microsoft Access 2016.

```
Public Function funCreateMenu(strMenu As String) As Boolean
        Dim myBar As CommandBar
        'Создаем панель меню
        myBar = appAccess.CommandBars.Add(strMenu, msoBarTop, True)
        funCreateMenuControls strMenu '<10> Создаем кнопки меню
        myBar.Visible = True 'Отображаем меню
        funCreateMenu = True 'Возвращаем результат
End Function
```

Создание кнопок меню

В этой части лекций рассказывается о программе, которая создает две кнопки Помощник и Справка. При нажатии их будет отображаться файл справки и помощник по Вашей программе.

```
Public Function funCreateMenuControls(strMenu As String) As Boolean
    Dim but As CommandBarButton

    'Добавляем первую кнопку
    but = appAccess.CommandBars(strMenu).Controls.Add(msoControlButton)
    With but
        .BeginGroup = True 'Начинаем размещение с начала группы
        .FaceId = 1 'Устанавливаем код кнопки
        .Style = msoButtonCaption 'Выбираем стандартный тип
        .Caption = "Справка" 'Называем кнопку
        .OnAction = "funCreateNewHelp" 'Определяем программу справки
    End With
    'Добавляем вторую кнопку
    but = appAccess.CommandBars(strMenu).Controls.Add(msoControlButton)
    With but
        .Caption = "Помощник" 'Называем кнопку
        .Style = msoButtonCaption 'Выбираем стандартный тип
        .FaceId = 2 'Устанавливаем код кнопки
        .OnAction = "funCreateAssistant" 'Определяем программу помощника
    End With
    funCreateMenuControls = True 'Возвращаем результат
End Function
```

Заключение

Вот и все. Вы полностью прошли первый этап обучения, и теперь только на практике, разрабатывая программы от простых к сложным, Вы сможете создавать законченные приложения по Access. Несколько советов по написанию программ

- названия функций давайте только на английском языке, могут возникнуть проблемы в сложных программах;
- старайтесь почаще их комментировать, тогда Вам легче будет разбираться в их структуре;
- не старайтесь писать очень большие базы данных, размер ее – это не показатель качества;

Если Вам все ясно, то Вы можете приступить к изучению практического задания по прослушанным лекциям. Для этого откройте папку Задание.