

# AVB201 - Beginning Access VBA

Access VBA Programming for  
experienced Windows users

D. H. D'Urso and Associates

*P.O. Box 6142*

*Laguna Niguel, CA 92607*

*949-489-1472*

*<http://www.dhdursoassociates.com>*

# AVB201 - Beginning Access VBA

- Introduction (s)
- Course Packet
  - Student Questionnaire
  - Curriculum overview
  - Syllabus
  - List of Useful Books
  - PowerPoint handouts for all sessions
  - Evaluation form in back
  - Training certificate
  - CD with all slides and exercise files

# AVB201 – Beginning Access VBA

- Quick pace for experienced windows users
- Does assume prior knowledge of Access
- Power-user, not programmer, oriented

# AVB201 – Beginning Access VBA

## Course Topics:

- Variables
- Statements
- Functions
- Procedures (Subs)

# AVB201 – Beginning Access VBA

## Course Format:

- 2 Sessions
- Lecture
- Demo
- Student "hands-on" - by the end of the class the student will have constructed some small but functional code fragments
- Exercises are cumulative - later examples build on objects created earlier

# AVB201 – Beginning Access VBA

## Course Schedule (3 hour sessions):

- Session 1
  - Event-driven programming
- Session 2
  - Language and Constructs

# AVB201

## Visual Basic For Applications MS Access, Beginning Course

D. H. D'Urso and Associates

*P.O. Box 6142*

*Laguna Niguel, CA 92607*

*949-489-1472*

*<http://www.dhdursoassociates.com>*

# OUTLINE

- SESSION 1: Event driven Programming in Access (3 hours).
- SESSION 2: VBA Language Constructs and Programming Techniques (3 hours).
- AVB202: Working Programmatically with Data (3 hours).

# Session 1:

## Event Driven Programming in Access

- Why VBA ?
- Using events and event procedures in event driven programming environment.
- Using VBA Editor.
- Event types in Access.
- Differentiating between functions and sub procedures.
- Organizing VBA code into modules.
- Using Variables, Scope, and Data types.
- Exploring and using built-in functions.

## 1.1 Why VBA ?

- Greatly extends the functionality of Macros.
- Allows customized interactions with users.
- Enables complex operations on records.
- Enables complete control over the Access Object Model and its methods and properties.
- Capable of making reports and forms that can make decisions based on user choices, database data, and user supplied values.
- Offers improved error handling capabilities.
- Programming is FUN !!

## 1.2 Events and Event Procedures

- Event is when something happens to any object in your application. Ex: User clicks a button (object) on a form (also an object).
- Event Procedure is a piece of VBA code that is associated with this object and this event. Since event code is programmed the event can trigger any number of actions. Ex: deleting a record and closing the form.
- In such event driven programming model, the user is in control of when and what should happen next in the application.

## 1.3 Using VBA Editor

- VBA editor is the place where you write your VBA code.
- Demonstration: Make “Add New” command button on a Products form.
- VBA editor keeps your code organized into modules, functions, procedures, and/or classes (more on module types later). You can write, document, test, compile, and debug your code with supplied tools >> explore VBA menus.
- We will use the VBA editor for the rest of this course.

## ..cont

### Exercise:

1. Add “Delete” button to Products form and check the procedure.
2. Explore changing the code and see how VBA automatically notifies you of certain errors.
3. Run procedure by clicking the “Delete” command button.

## 1.4 Types of Events in Access

Events in Access can be categorized into several groups:

1. Windows (Form, Report): opening, closing, resizing.
2. Data: making current, deleting, updating.
3. Focus: activating, entering, exiting.
4. Keyboard: pressing or releasing a key.
5. Mouse: clicking a mouse button.
6. Print: formatting, printing.
7. Error and timing: when error occurs in code or some time passes.

# Commonly Used Events

- Events are normally associated with a form, report, or some control on a form. Some common events are:
  1. Forms: On Open, On Load, Before Update, On Unload, On Current, On Delete, Before Delete.
  2. Controls: On Click, Before Update, After Update, On Double Click.
  3. Reports: On Open, On Activate, On Close, On No Data.

**NOTE: Single user action can also trigger several events that then run in succession. For example when user opens a form all of the following events occur:**

**Open >> Load >> Resize >> Activate >> Current**

# Event Flow

**NOTE: A single user action can also trigger several events that then run in succession. For example when the user opens a form all of the following events occur:**

**Open >> Load >> Resize >> Activate >> Current**

## 1.5 Differentiating between Functions and Sub Procedures.

- Functions return a value to the calling code, usually providing the result of the function's operation.
- Sub Procedures execute the code in the procedure but do not return any values.
- Both, functions and procedures, can accept multiple input values from the calling program that can be used inside their respective operations.

## ..cont

- Function Syntax:

**Function FunctionName(passed arguments) As “Data Type of Return”**

**..some code..**

**FunctionName = “Return Value”**

**End Function**

- Sub Procedure Syntax:

**Sub SubName(passed arguments)**

**..some code..**

**End Sub**

## 1.6 Organizing VBA Code into Modules

- All event procedures and any other VBA code are grouped and placed into “Modules”.
- MS Access has 4 types of modules: Standard, Form, Report, and Class.
- Each form and report has its own form or report module. All event procedures associated with a particular form or report should reside in their respective module.
- Standard module holds any other common or non form/report specific code.
- Class modules support object oriented programming approach (OOP) to development.

**Exercise:** Access any of the module types. In all cases you work with the same VBA editor.

## ..cont

This is the place where we start writing some of our own code.

Interactive Exercise:

1. Create new standard module.
2. Type in the following code:

```
Function Area (Height As Double, Width As Double) As Double
```

```
    Area = Width * Height
```

```
End Function
```

3. Optionally compile code (under Debug menu).
4. Run code in Immediate window: ?Area(3,4)

## ..**cont**

And now try the following procedure:

```
Sub MyLoop()  
  Dim loopcount As Integer, i As Integer  
  loopcount = 3  
  For i = 1 To loopcount  
    Debug.Print "Loop ", i  
  Next i  
  Beep  
End Sub
```

More about “For” statement in **Session 2**.

## 1.7 Variables, Scope, and Data Types

- Variables are containers that hold your data in programs.
- Each variable should have a type. VBA supports a wide range of data types that are not identical to data types declared on fields in tables. Knowing both groups and how to map them is essential for good programming.
- If you forget to assign type to a variable in VBA, then the variable is treated as a “Variant”. Variants assume data type when data is loaded in to the variable.

**NOTE: !! Avoid using Variants”. They slow the execution and make for poor documentation.**

## Some often used data types in VBA

- Boolean (True or False)
- Currency (formatted number \$56.67)
- Double, Single (51.145678)
- Integer (321)
- Date (#8/28/04#)
- String (“Any word or sentence”)

Q: What are their equivalents in database tables?

# Scope of Variables (and Procedures)

- **Scope determines where can variable be used. This depends on the place where and also how it was declared.**
- **Variables declared inside functions or subs must be declared with a Dim statement. They are always local to that sub or function..**

## ...cont

- **Variables declared outside any function or sub are always global to that module. Whether they are also available to other modules, depends on what you precede their declarations with:**
  1. **“Private Area As Double” would be same as “Dim Area As Double”.**
  2. **“Public Area As Double” on the other hand is global to the application.**

**NOTE: !!You can not use Public declaration inside function or sub. Use Dim ONLY.**

## ..cont

- **Functions and Procedures also have a scope.**
- **Preceding function or procedure declaration with a word “Public” or “Private” makes that function or sub available “in the module where declared only” OR they can be called (used) from any place in the application. Ex: Procedure in one module can call Public functions declared in any other module.**
- **Public and Private declarations are useful to avoid confusion with keeping track of multiple names used for variables, functions, and procedures. Similar operations on two different forms can use procedures with same name as long as they are both declared as private in their respective modules.**

## ..cont

### Exercise:

1. Modify the MyLoop procedure to accept the number of loops. Call the new one MyAreaLoop
2. Call the Area function procedure from the MyAreaLoop procedure and pass it a length of 10 and width of 1. (You will have to set up a variable of type double for area.)
3. Run the MyAreaLoop procedure.

## ..cont

### Exercise:

```
Sub MyAreaLoop(numloops As Integer)
    Dim loopnum As Double, i As Integer
    For i = 1 To numloops
        loopnum = i
        Debug.Print "Area ", i, Area(10, loopnum)
    Next i
    Beep
End Sub
```

## ..cont

### Mini-Quiz:

1. If you change the Area function to private does myarealoop still work? Why?
2. How could you use global variables? Should you?

## 1.8 Built-in Functions and Procedures

- Beside custom made functions and procedures there are some that are built-in and part of the Access VBA language. We will see several examples in the following Sessions.
- “Beep” is an example of built-in procedure and performs a single operation: It makes computer beep. Once.
- Built-in functions can be used anywhere in your program.
- Knowing what is available is essential for efficient programming so we don't go and try to reinvent the wheel.

# Some Common built-in Functions

- Today = Date() returns today's date.
- UCase(String) converts all letters in string to upper case.
- Round(3.89) rounds a value in parenthesis to nearest integer (4 here).
- Len(String) returns number of characters in a string.

**NOTE:** There are close to 100 built in functions available for almost any area of data manipulation. Check some reference book for complete list or consult an Online Help in Access. We will introduce and use several in the hours ahead.

## ..cont

**Built-in functions are grouped into several categories:**

- **Conversion (converts one data type to another. UCase, LCase, etc..)**
- **Date/Time**
- **Financial**
- **Mathematical**
- **String Manipulation (extract parts of string, truncate, etc..)**
- **Programming (check for missing values, determine data types etc..)**
- **Domain (work on several values at once and perform statistical operations such as average, finding maximum / minimum value etc..)**

## ..cont

Try a couple from the immediate window:

### Immediate

```
fname = "Dan"  
lname = "Durso"  
email = lcase(left(fname,1)) &lcase(lname)  
?email  
ddurso  
?date()  
2/25/2005
```

# AVB201 – Introduction to VBA

**Session 1 END**