VBA Reference Card

Line continuation, comments, assignment

```
i = i+2  ' Comment
s =  "long text A" & _
     "long text B"   ' Comment in last line only
```

Set f = Forms(0)	Store a reference
Set f = New Form_frmG	Create object, store ref
Set f = Nothing	Delete object if last ref

Conditional statements

```
If a=1 Then c=d+2                    Single statement

If a=1 Then
   c=d+2 . . .                       Multiple statements
ElseIf a=2 Then                      } Optional
   c=d / 2 . . .                     }
Else                                 } Optional
   c=0 . . .                         }
End If

Select Case zip
Case 4000
   type = a . . .
Case 4001, 5000 To 5999
   type = b . . .
Case Else                            } Optional
   type = c . . .                    }
End Select

On Error Resume Next                 Ignore error

On Error GoTo err                    Enable error handler
   . . .
err: MsgBox( . . . )                 Continue here at error

On Error GoTo 0                      Let VBA handle errors
```

Loops

```
While a<10                           May be empty loop
   c=c*2
   . . .                             Exit not allowed
Wend

Do While a<10                        May be empty loop
   c=c*2
   . . . Exit Do                     Exit optional
   . . .
Loop

Do                                   Loop at least once
   c=c*2
   . . . Exit Do                     Exit optional
   . . .
Loop While a<10

For i=1 To last Step 2               Step optional
   c=c*2                             May be empty loop
   . . . Exit For                    Exit optional
   . . .
Next i
Don't trust value of i when loop ends without Exit

For Each f In Forms                  Scan collection
   call print(f.name  . . . )
   . . . Exit For                    Exit optional
   . . .
Next
```

Declarations

```
Dim B, C As Byte          B is Variant, C is 0..255
Boolean                   True (<> 0), False (=0)
Integer                   16 bit, -32,786 .. 32,767
Long                      32 bit integer, -2.14E9 .. 2.14E9
Currency                  64 bit integer / 10,000
Single                    32 bit, -3.4E38 .. 3.4E38, 6 digits
Double                    64 bit, -1.8E308 .. 1.8E308, 14 digits
Date                      Double, days since 30. Dec 1899, 0:00
Object                    Reference to any object
Form                      Reference to any Form
Variant                   Any of the types or Null, Empty, Nothing,
Error - plus a type tag. All database fields are Variant
String                    Variable length, max 2E9 characters
String * 50               Fixed length, space filled

Initial values            String = "", Boolean = False
Number, date = 0          Database field = Null
Object = Nothing          Variant = Empty

Dim c(5, 1 To 6) As t     Same as c(0..5, 1..6)
Dim d( ) As Single        Dynamic array declaration
ReDim d(5, 1 To 6)        Statement
    Index range (re)defined, data lost
ReDim Preserve d(5, 1 To 8)
    Last index range redefined, data preserved
Erase d                   Releases memory for dynamic array

Type Customer             Simple modules only
    custID As Long
    custName As String * 50
    custAddress As String
End Type
Dim custTable(20) As Customer
```

Procedures = Subroutines and Functions

```
proc a, b, , d            Parenthesis-free notation
Call show(a, b, , d)      Subroutines only
res = fnc(a, b, , d)      Functions only

Sub show(a, b As t, Optional c, d)
   If IsMissing(c) Then . . .
   Exit Sub               Optional
   . . .
End Sub

Function fnc(a, b As t, Optional c, d) As String
                          As String is optional
   If IsMissing(c) Then . . .
   fnc= result . . .
   Exit Function          Exit optional
   . . .
End Function
```

Module and Scope

```
Dim a                     Visible in this module only
Public b                  Visible to all modules

Private Sub show(p)       Visible in this module only
    Dim c                 Visible in this sub only
    Static d              Visible in this sub only,
    . . .                 but survives calls
End Sub

Public Sub show(p)        Visible to all modules
    Dim c                 Visible in this sub only
    . . .
End Sub
```

Constants

23, -23, 0, -4.9E-20	**Decimal numbers**
&h29AFF, &o177	**Hex and Octal**
"Letter to:"	**Strings**
Chr(65), Chr(vbKeyA)	The text "A"
"John" & Chr(10) & "Doe"	Two-lines, Chr(10)=new line
"Don't say ""no"" "	Don't say "no"
"select * from g where a='simpson' ;"	
Single quotes are suited for SQL	
True, False	**Booleans**
	Date/time
#10/24/02#	24th Oct 2002
#10/24/02 14:15:00#	24th Oct 02 at 14:15
#10/24/02 2:15 pm#	24th Oct 02 at 14:15
Null, Empty	**Special values**
Nothing	Object reference to nothing

Constant declaration

```
Const max=10, start=#3/24/2#
```

Addressing

Forms(i)	Element in collection
Forms("frmCst" & i)	
Forms!frmCst2	Bang-operator
Me.Name, Me!name	Property~Control in module
Me.subLst.Form.name	Property in subform
Me.Parent.txtName	Control in main form
basCommon.simDate	Variable in foreign module
c(row, col)	Indexing an array
custTable(i).custID	Field in array of records

```
With Me.Recordset                    Apply before dot and bang
   .addr = .addr & zip
   !name = Null
   .MoveNext
   . . .
End With
```

Operators, decreasing precedence

Nulls: Any Null operand gives a Null result.

^	Exponentiation
-	Unary minus, 2*-3 = -6
*	Multiply, Result type is Integer, Double, etc.
/	Divide, Single or Double result
\	Integer divide, result truncated, 5\3 = 1
Mod	Modulus (remainder), 5 Mod 3 = 2
+ -	Add and subtract
&	Concatenation, String result
= <> < > <= >=	Equal, unequal, less than, etc.
Is	Compare two object references, e.g.
If r Is Nothing	Test for nil-reference
a Between 3 and 9	Not in VBA, okay in SQL
a IN (2, 3, 5, 7)	Not in VBA, okay in SQL
Not	Negation. Bit-wise negation for integers
And	Logical And. Bit-wise And of integers
Or	Logical Or. Bit-wise Or of integers
X	Exclusive Or. Bitwise on integers
Eqv	Logical equivalence. Bitwise on integers
Imp	Logical implication. Bitwise on integers

s Like "s?n" Wildcard compare. ? any char here.
any digit here. * any char sequence here.
[a-k] any letter between a and k here.

Simple conversion functions

Errors: "Invalid use of Null" for Null parameters
Overflow or type mismatch for bad parameters.

CByte("37")	=37. Overflow outside 0..255
CInt("2.6")	= 3
Round(3.1415)	= 3.0000 (Double)
CLng("99456")	= 99456
CCur(1/3)	=0.3333 (always 4 decimals)
CSng("-2.6e-2")	= -0.026
CDbl("-2.6")	= -2.6
CDbl(#12/31/1899#)	= 1.0
CDate("23-10-03")	= #10/23/2003#
Uses regional setting for input format	
CDate(1)	= #12/31/1899#
CStr(23)	= "23". No preceding space.
Str(23)	= " 23". Preceding space when >= 0
CStr(#10/23/2003#)	= "23-10-03"
Uses regional setting for output format	
CVar(X)	= X As Variant. X may be Null

String functions

Null parameters: A Null string as input will give the result Null. Null as another parameter is an error.

Asc("AB")	= 65, Ascii code for first character
Chr(65)	= "A", a one-letter string with this ascii character
Len("A_B")	= 3, length of string.
Left("abc", 2)	= "ab", leftmost two characters
Left("abc", 8)	= "abc", as many as available
Right("abc", 2)	= "bc", rightmost two characters
Mid("abcdef", 2, 3)	= "bcd", three chars, chars 2-4
LTrim(" ab ")	= "ab ", leading spaces removed
RTrim(" ab ")	= " ab", trailing spaces removed
Trim(" ab ")	= "ab", leading and trailing removed
Lcase("A-b")	= "a-b", lower case of all letters
Ucase("A-b")	= "A-B", upper case of all letters
Space(5)	= String of 5 spaces

Option Compare Text | Binary | Database
Option in start of module. Text: string comparison is case insensitive and follows regional settings.
Binary: comparison is based on the internal ASCII code.
Database: comparison is defined by the SQL-engine.

StrComp("ab", "abc")	= -1, first string smallest
StrComp("ab", "ab")	= 0, strings equal
StrComp("ac", "abc")	= 1, first string largest
If "ab" < "abc" . . .	Works just as well

Iif and Choose

Iif(a=a, b, c)	= b
Iif(a<>a, b, c)	= c
Iif(Null, b, c)	= c
Choose(2, a, b, c)	= b
Choose(4, a, b, c)	= Null
Choose(Null, a, b, c)	Error

Array bounds

LBound(d)	Lower bound for first index
LBound(d, 2)	Lower bound for second index
UBound(d)	Upper bound for first index
UBound(d, 3)	Upper bound for third index

Format function

Converts a value to a string, based on a format string. Format characters that are not placeholders, are shown as they are. Backslash+character is shown as the character alone, e.g. \d is shown as d.

Numeric placeholders

0	Digit, leading and trailing zero okay here
#	Digit, no leading or trailing zero here
.	Decimal point (or regional variant)
E- or e-	Exponent, use all placeholders
E+ or e+	Show exponent with plus or minus
%	Show number as percent

Format(2.3, "00.00")	= "02.30"
Format(2.36, "#0.0")	= "2.4"
Format(0.3, "##.0#")	= ".3"
Format(32448, "(00)00 00")	= "(03)24 48"
Format(32448, "##.#E+")	= "32.4E+3"
Format(32448, "##.#E-")	= "32.4E3"
Format(0.5, "#0.0%")	= "50.0%"
;	Separator between formats for positive, negative, zero, and null values.
Format(-3, "000;(000);zero;---")	= "(030)"

String placeholders

@	Character or space
&	Character or nothing
!	Cut off from left

Format("A123", "@@@@@@")	= "¬¬A123"
Format("A123", "&&&&&")	= "A123"
Format("A123", "(@@)-@")	= "(A1)-23"
Format("A123", "!(@@)-@")	= "(12)-3"

Date/time placeholders

Example: DT = #2/3/2002 14:07:09# (Sunday)

Format(DT, "yyyy-mm-dd hh:nn:ss")
= "2002-02-03 14:07:09"

Format(DT, "yy-mmm-d at h:nn am/pm")
= "02-feb-3 at 2:07 pm"

Format(DT, "dddd t\he y't\h \da\y of yyyy")
= "Sunday the 34'th day of 2002"

d	Day of month, no leading zero "3"
dd	Day of month, two digits "03"
ddd	Day of week, short text "Sun"
dddd	Day of week, full text "Sunday"
m	Month, no leading zero "2" (Interpreted as minutes after h)
mm	Month, two digits "02" (Interpreted as minutes after h)
mmm	Month, short text "Feb"
mmmm	Month, full text "February"
y	Day of year "34"
yy	Year, two digits "02"
yyyy	Year, four digits "2002"
h	Hour, no leading zero "14" or "2"
hh	Hour, two digits "14" or "02"
AM/PM	Show AM or PM here, hours 12-based
am/pm	Show am or pm here, hours 12-based
n	Minutes, no leading zero "7"
nn	Minutes, two digits "07"
s	Seconds, no leading zero "9"
ss	Seconds, two digits "09"

Type check functions

Returns True if v is declared with the type tested for, is a Variant currently with this type, or is a constant of this type. IsDate and IsNumeric also test whether v is a text that can be converted to that type.

IsArray(v)	Tests for any type of array
IsDate(v)	Tests whether v is a date or a string that can be converted to a date
IsEmpty(v)	Tests whether v is unallocated (Strings of length 0 are not Empty)
IsError (v)	Tests whether v is an error code
IsMissing (v)	Tests whether v is a parameter that is missing in the current call.
IsNull (v)	Tests whether v is of type Null. (Strings of length 0 are not Null)
IsNumeric(v)	Tests whether v is a numeric type (Byte, Integer, Currency, etc.) or a string that can be converted to a numeric type.
IsObject(v)	Tests whether v is a reference to an object, for instance a Form. True also if v is Nothing (the nil-pointer)
VarType(v)	Integer showing the type:

0	vbEmpty	8	vbString
1	vbNull	9	vbObject
2	vbInteger	10	vbError
3	vbLong	11	vbBoolean
4	vbSingle	12	vbVariant (array)
5	vbDouble	17	vbByte
6	vbCurrency	36	vbUserDefinedType
7	vbDate	8192	vbArray (added)

Date and time functions

A date value is technically a Double. The integer part is the number of days since 12/30-1899, 0:00. The fractional part is the time within the day.

Several functions accept date parameters as well as string parameters that represent a date and/or time.

Null parameters: Always give the result Null.

Now( )	= current system date and time
Date( )	= current date, integral date part
Time( )	= current time, fractional date part
Timer( )	= Number of seconds since midnight, with fractional seconds.
Date = . . .	Sets current system date
Time = . . .	Sets current system time
DateSerial(2002, 12, 25)	= #12/25/2002#
TimeSerial(12, 28, 48)	= 0.52 (Time 12:28:48)
Day(#12/25/02#)	= 25, the day as Integer
Month(#12/25/02#)	= 12, the month as Integer
Year(#12/25/02#)	= 2002, the year as Integer
Weekday(#12/25/02#)	= 4 (Sunday=1)
Hour(35656.52)	= 12 (Time 12:28:48)
Minute(35656.52)	= 28
Second(35656.52)	= 48

Control prefixes

cbo	Combobox	lbl	Label	bas	Module
chk	Checkbox	lst	Listbox	frm	Main form
cmd	Button	mni	Menu item	fsub	Subform form
ctl	Other	sub	Subform control	qry	Query
grp	Option group	tgl	Toggle button	qxtb	Crosstab qry
opt	Option button	txt	Text control	tbl	Table

DLookup, DMin, etc.

DLookup("name", "tblGuest", "guestID=7")
= name of guest with guestID=7.
All three parameters are texts inserted into SQL.

DMin("roomID", "tblRooms", "roomType=2")
= smallest room number among double rooms.

DMax, DSum, DCount, DAvg
Similar, just finds largest, sum, number of, average.
Null treatment, see SQL.

MsgBox

MsgBox("Text", vbYesNo+vbCritical) =vbYes
Also: vbInformation, vbQuestion, vbExclamation

Math functions

Sqr(x)	Square root of x. Sqr(9) = 3.
Sin(x), Cos(x), Tan(x), Atn(x)	Trigonometric functions. X measured in radian (180 degrees = pi = 3.141592 radian) Sin(0) = 0, Sin(3.141592 / 2) = 1
Exp(x)	e to the power of x (e = 2.7182...)
Log(x)	Natural logarithm of x. Log(e) = 1.
Rnd( )	A random number between 0 and 1. Type is Single.
Abs(x)	Returns x for x>=0, -x otherwise.
Sgn(x)	Returns 1 for x>0, 0 for x=0, -1 for x<0
Int(x)	Rounds x down to nearest integral value
Fix(x)	Rounds x towards zero
Hex(x)	Returns a string with the hexadecimal value of x. Hex(31) = "1F"
Oct(x)	Returns a string with the octal value of x. Oct(31) = "37"

Null allowed for x

Financial functions

NPV(0.12, d( )) The array d must be of type Double and contain a list of payments. Returns the net present value of these payments at an interest rate of 0.12, i.e. 12%.

IRR(d( )) The array d must be of type Double and contain a list of payments. Returns the internal rate of return, i.e. the interest rate at which these payments would have a net present value of 0. If the list of payments have many changes of sign, there are many answers, but IRR returns only one.

IRR(d( ), 0.1) The second parameter is a guess at the interest rate, to allow IRR to find a reasonable result.

SYD, NPer and many other financial functions are available for finding depreciated values, number of periods to pay a loan back, etc.

VBA short-cuts

VBA ↔ Access	Alt+F11
Property list	Ctrl+J
Constant list	Ctrl+Sh+J
Parameter list	Ctrl+I
Immediate	Ctrl+G
Run	F5
Step into	F8
Step over	Shift+F8
Break loop	Ctrl+Break
Object browser	F2
Close VBA/Appl	Alt+F4
In Form: User mode	F5

General short-cuts

Select full field	F2
Zoom window	Shift+F2
Combo open	Alt+Down
Next Form	Ctrl+F6
Upper/lower section	F6
Choose menu	Alt
Next menu/tab	Ctrl+Tab
Next application	Alt+Tab
Update	(Shift+) F9
Open properties	Alt+Enter
Close Form	Ctrl+F4
Design mode	Alt+V+Enter

Record set DAO 3.6

```
Dim rs As Recordset, clone As Recordset, Dim A( )
  s = "SELECT * . . . "          Or "tblCustomer"
  Set rs = CurrentDB.OpenRecordset(s)
  Set clone = rs.Clone
  While Not rs.EOF               EndOfFile (BOF similar)
    rs.Edit  (or rs.AddNew)      Prepare edit buffer
    rs ! fieldX = . . .          Change edit buffet
    rs.Update                    Update current record
    . . .
    rs.Delete                    Delete current record
    rs.MoveNext
  Wend
  A = rs.GetRows(n)              Copy n rows to A
  A(0, 3)                        First field of 4th record
  rs.Close
```

Other properties:

rs.AbsolutePosition = 0	
rs.Bookmark = clone.Bookmark	
rs.Move(n)	Move *current* n records back/forward
rs.MoveNext	. . . MovePrevious, MoveFirst, MoveLast
rs.FindFirst("a='simp' ")	. . . FindPrevious, FindNext, FindLast
rs.NoMatch	True if Find didn't succeed
rs.Requery	Re-compute query after changes
rs.RecordCount	Number of records currently loaded by database engine
rs.Name	String, SQL-statement for query, readonly
rs.DateCreated, rs.LastUpdated	Only for tables

SQL

SELECT name, zip FROM tblGuest **WHERE** ID=2;

SELECT tblTown.name **AS** address, tblGuest.name
FROM tblGuest **INNER JOIN** tblTown
ON tblGuest.zip = tblTown.zip
WHERE tblGuest.zip = 4000 **ORDER BY** name;
Or: . . . ORDER BY name, tblGuest.zip **DESC**;

SELECT stayID, **Min**(date) AS arrival
FROM tblRoomState WHERE state = 1
GROUP BY stayID;

Null handling:
ORDER BY: Null smaller than anything else.
Sum, Avg, Min, Max: Look at non-null values. Null if all are null.
Count: Counts non-null values. Zero if all are null (but Null for Crosstab).

SELECT name FROM tblGuest WHERE zip
IN (SELECT zip FROM tblTown WHERE name<"H");

SELECT . . . WHERE zip **NOT IN** (1200, 1202, 1205);

SELECT 0, "New" FROM tblDummy
UNION SELECT zip, name FROM tblTown;
Concatenates one table (here a single record 0, New) with another table. Field 1 under field 1, etc.

UPDATE tblGuest Updates records where . . .
SET name = "John Smith", zip = 4000 WHERE ID = 2;

INSERT INTO tblGuest (name, zip) Adds one record
VALUES ("Ahmet Issom", 5100);

INSERT INTO tblTemp Adds many records
SELECT * FROM tblGuest WHERE zip=4000;

DELETE FROM tblGuest WHERE ID = 2;