**- Data Type: is container type to allocate the memory and store the data on declared variables**

**Example:**

**A =11 ‘ here A is variable/temp. memory and 11 is data/value**

**B =33**

**- Operator**

**- If condition : is decision making statement**

**- Select Case**

**- Loop and its Type**

**VBA Data Types**

When you declare a variable, you should also identify its data type. You're probably already very familiar with data types because you assign data types to table fields. VBA uses the same data types to define a variable.

The most important job of a data type is to ensure the validity of your data. Specifying a data type won't keep you from entering an invalid value, but it will keep you from entering an invalid *type*. If you omit the data type, VBA applies the Variant data type to your variable—it's the most flexible and VBA won't guess at what the data type should be. Table 3.1 compares VBA's many data types.

**Table 3.1 VBA Data Type Comparison**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Data Type or Subtype** | **Required Memory** | **Default Value** | **VBA Constant** | **Range** |
| Integer | 2 bytes | 0 | vbInteger | –32,768 to 32,767 |
| Long Integer | 4 bytes | 0 | vbLong | –2,147,483,648 to 2,147,486,647 |
| Single | 4 bytes | 0 | vbSingle | –3402823E38 to –1.401298E–45 or 1.401298E–45 to 3.402823E38 |
| Double | 8 bytes | 0 | vbDouble | 1.123456789123456(15th decimal position)  –1.79769313486232E308 to –4.94065645841247E–324 or 1.79769313486232E308 to 4.94065645841247E–324 |
| Currency | 8 bytes | 0 | vbCurrency | –922,337,203,477.5808 to 922,337,203,685,477.5807 |
| Date | 8 bytes | 00:00:00 | vbDate | January 1, 100 to December 31, 9999 |
| Fixed String | String's length | Number of spaces to accommodate string | vbString | 1 to 65,400 characters |
| Variable String | 10 bytes plus the number of characters | Zero- length string ("shsjhsg111222") | vbString | 0 to 2 billion characters |
| Object | 4 bytes | Nothing (vbNothing) | vbObject | Any Access object, ActiveX component or Class object : workbook, worksheet, rows etc. |
| Boolean | 2 bytes | False | vbBoolean | –1/true or 0/false |
| Variant | 16 bytes | Empty (vbEmpty) | vbVariant | Same as Double |
| Decimal | 14 bytes | 0 | vbDecimal | -79,228,162,514,264,337,593,543,950,335 to 79,228,162,514,264,337,593,543,950,335 or –7.2998162514264337593543950335 to 7.9228162514264337593543950335 |
| Byte | 1 byte | 0 | vbByte | 0 to 255 |

**The Boolean Data Type**

Use the Boolean numeric data type to store logical data that contains only two values: on and off, true and value, yes and no, and so on. The keywords True and False are predefined constants and are interchangeable with the values –1 and 0, respectively. To illustrate these keywords, enter the following statements, one at a time in the VBE's Immediate window, as shown in Figure 3.7:

?True = 0

?True = -1

?False = 0

?False = -1

?True = False

## Operators

An **Operator** can be defined using a simple expression - 4 + 5 is equal to 9. Here, 4 and 5 are called **operands** and + is called **operator**. VBA supports following types of operators −

* Arithmetic Operators
* Comparison Operators
* Logical (or Relational) Operators
* Concatenation Operators

## The Arithmatic Operators

Following arithmetic operators are supported by VBA.

Assume variable A holds 5 and variable B holds 10, then −

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| + | Adds the two operands | A + B will give 15 |
| - | Subtracts the second operand from the first | A - B will give -5 |
| \* | Multiplies both the operands | A \* B will give 50 |
| / | Divides the numerator by the denominator | B / A will give 2 |
| % | Modulus operator and the remainder after an integer division | B % A will give 0 |
| ^ | Exponentiation operator | B ^ A will give 100000 |

## The Comparison Operators

There are following comparison operators supported by VBA.

Assume variable A holds 10 and variable B holds 20, then −

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| = | Checks if the value of the two operands are equal or not. If yes, then the condition is true. | (A = B) is False. |
| <> | Checks if the value of the two operands are equal or not. If the values are not equal, then the condition is true. | (A <> B) is True. |
| > | Checks if the value of the left operand is greater than the value of the right operand. If yes, then the condition is true. | (A > B) is False. |
| < | Checks if the value of the left operand is less than the value of the right operand. If yes, then the condition is true. | (A < B) is True. |
| >= | Checks if the value of the left operand is greater than or equal to the value of the right operand. If yes, then the condition is true. | (A >= B) is False. |
| <= | Checks if the value of the left operand is less than or equal to the value of the right operand. If yes, then the condition is true. | (A <= B) is True. |

## The Logical Operators

Following logical operators are supported by VBA.

Assume variable A holds 10 and variable B holds 0, then −

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| AND | Called Logical AND operator. If both the conditions are True, then the Expression is true.  T T = T  T F = F  F T = F  F F = F | a<>0 AND b<>0 is False. |
| OR | Called Logical OR Operator. If any of the two conditions are True, then the condition is true.  T T = T  T F = T  F T = T  F F = F | a<>0 OR b<>0 is true. |
| NOT | Called Logical NOT Operator. Used to reverse the logical state of its operand. If a condition is true, then Logical NOT operator will make false. | NOT(a<>0 OR b<>0) is false. |
| XOR | Called Logical Exclusion. It is the combination of NOT and OR Operator. If one, and only one, of the expressions evaluates to be True, the result is True. | (a<>0 XOR b<>0) is true. |

## The Concatenation Operators

Following Concatenation operators are supported by VBA.

Assume variable A holds 5 and variable B holds 10 then −

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| + | Adds two Values as Variable. Values are Numeric | A + B will give 15 |
| & | Concatenates two Values | A & B will give 510 |

Assume variable A = "Microsoft" and variable B = "VBScript", then −

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| + | Concatenates two Values | A + B will give MicrosoftVBScript |
| & | Concatenates two Values | A & B will give MicrosoftVBScript |

# How to use the if condition (VBA)

There are following types of if condition:

1. If condition
2. If else condition
3. If else if else if else condition / ladder if else
4. Nested if else : if inside if

### If...Then...Else Statement

Sub Using\_IF()  
 ' Dimension the variable.  
 Dim x As Integer  
 ' Place a value in x.  
 x = Int(Rnd \* 100)  
 ' Display the value of x.  
 MsgBox "The value of x is " & x & "."  
 ' Test to see if x less than or equal to 10.  
 If x <= 10 Then  
 ' Display a message box.  
 MsgBox "X is <=10"  
 ' Test to see if x less than or equal to 40 and greater than 10.  
 ElseIf x <= 40 And x > 10 Then  
 MsgBox "X is <=40 and > 10"  
 ' Test to see if x less than or equal to 70 and greater than 40.  
 ElseIf x <= 70 And x > 40 Then  
 MsgBox "X is <=70 and > 40"  
 ' Test to see if x less than or equal to 100 and greater than 70.  
 ElseIf x <= 100 And x > 70 Then  
 MsgBox "X is <= 100 and > 70"  
 ' If none of the above tests returned true.  
 Else  
 MsgBox "X does not fall within the range"  
 End If  
 End Sub

# How to use the CASE Statement (VBA)

## Description

The Microsoft Excel CASE statement has the functionality of an IF-THEN-ELSE statement.

The CASE statement is a built-in function in Excel that is categorized as a Logical Function. It can be used as a VBA function (VBA) in Excel. As a VBA function, you can use this function in macro code that is entered through the Microsoft Visual Basic Editor.

## Syntax

The syntax for the CASE statement in Microsoft Excel is:

Select Case test\_expression

Case condition\_1

result\_1

Case condition\_2

result\_2

...

Case condition\_n

result\_n

[ Case Else

result\_else ]

End Select

## Example (as VBA Function)

The CASE statement can only be used in VBA code in Microsoft Excel.

Let's look at some Excel CASE statement function examples and explore how to use the CASE statement in Excel VBA code:

Select Case LRegion

Case "N"

LRegionName = "North"

Case "S"

LRegionName = "South"

Case "E"

LRegionName = "East"

Case "W"

LRegionName = "West"

End Select

With the Excel CASE statement, you can also use the To keyword to specify a range of values. For example:

Select Case LNumber

Case 1 To 10

LRegionName = "North"

Case 11 To 20

LRegionName = "South"

Case 21 To 30

LRegionName = "East"

Case Else

LRegionName = "West"

End Select

With the Excel CASE statement, you can also comma delimit values. For example:

Select Case LNumber

Case 1, 2

LRegionName = "North"

Case 3, 4, 5

LRegionName = "South"

Case 6

LRegionName = "East"

Case 7, 11

LRegionName = "West"

End Select

And finally, with the Excel CASE statement, you can also use the Is keyword to compare values. For example:

Select Case LNumber

Case Is < 100

LRegionName = "North"

Case Is < 200

LRegionName = "South"

Case Is < 300

LRegionName = "East"

Case Else

LRegionName = "West"

End Select

**- Loop and its Type**

**:loop is iterator or repeation of statement**

**There are following types of loop**

1. **While loop**
2. **For loop**
3. **Do while**
4. **Do until**
5. **Foreach**