Piccola guida a VBA per Word

- **La differenza tra VBA (Visual Basic for Applications) e VB6 (Visual Basic versione 6)**
- **Il modello di un documento Word**
- **Il progetto di un documento Word**
- **Come creare un nuovo modello di documento**
- **Cosa sono le macro**
- **Come far apparire l'elenco delle macro di un documento Word ed accedere al progetto del documento**
- **Come creare una routine che viene eseguita automaticamente all'apertura del documento Word**
- **Come creare o eliminare subroutines che possono essere richiamate dal codice della macro**
- **Routines e variabili *Public* e *Private***
- **Come creare una macro per mezzo del registratore delle macro**
- **Come creare una macro scrivendo una routine in un modulo del progetto e assegnandogli una combinazione di tasti**
- **Le librerie necessarie perché le macro funzionino**
- **Utilizzare il registratore delle macro per imparare VBA e VB6**
- **Come entrare in VBA per Word**
- **Come visualizzare tutte le macro di un documento Word**
- **Come attivare le macro nel caso compaia il messaggio che esse sono inattivate per motivi di protezione da virus**
- **La scheda proprietà di un modulo o form**
- **Come sbloccare una macro che si interrompe con segnalazione di errore ed apertura dell'editor VBA**
- **Il font *UniCode***
- **La Guida in linea di VBA**
- **Come accedere ad un database access da un documento Word tramite una macro in codice VBA**
- **Passaggio di parametri ad una routine per valore o per riferimento**
- **Generare un carattere a partire da un codice di carattere**
- **Debugging di un codice VBA**
- **Le molteplici possibilità offerte da VBA**
- **Istruzioni ADO per ricercare un determinato record nel RecordSet**
- **Istruzioni ADO per aggiungere un record ad un database Access**
- **I metodi dell'oggetto *Selection***
- **La strutturazione di un documento Word in paragrafi e le relative istruzioni VBA**
- **Istruzioni VBA per cercare una stringa nel documento**
- **Come dotare un foglio Excel di una nuova funzione**

❍ La differenza tra VBA (Visual Basic for Applications) e VB6 (Visual Basic versione 6)

VBA (Visual Basic for Applications) e VB6 (Visual Basic versione 6) sono due programmi che presentano delle differenze.
Il primo è disponibile gratuitamente, il secondo è a pagamento
I form disponibili in VBA sono limitati a quelli presenti nella casella degli strumenti. Cliccando due volte su un form nella finestra di progetto (appare il form), se è visualizzata la casella degli strumenti, è attiva l'opzione Mediante VBA→Menu→Strumenti→Controlli aggiuntivi, ma in realtà i controlli che si aggiungono in tal modo alla casella degli strumenti spesso non funzionano. Invece i form disponibili in VB6 sono estremamente potenti. Mediante essi ci si può interfacciare senza problemi anche ai database.
Mediante VB6 si possono creare dei file *.exe mentre questo non è possibile con VBA.
Un programma VBA viene lanciato da una macro o dalla casella F8 o all'apertura del documento (se il codice è inserito nella routine AUTOOPEN), mentre un programma VB6 viene lanciato da un form o dalla routine MAIN del modulo del progetto VB6.
I form creati con VBA e importati in VB6 con la funzione di importazione funzionano malissimo in VB6, tanto che è preferibile creare ex novo un form in VB6 e copiarvi le routine contenute nei form VBA.
Mentre un form VBA, una volta caricato con *Load UserForm* e attivato con *UserForm.Show* non è più gestibile dalla routine da cui lo si è lanciato. Ad esempio non si possono scrivere i numeri di un conto alla rovescia in una label del form mediante il seguente codice:

```
Load UserForm
UserForm.Show
For Puntatore = 100 to 1, step 1
 UserForm.Label1.Caption = Puntatore
Next Puntatore
```

perché il controllo passa – con *UserForm.Showi* al form e questo rimane immobile senza far nulla.
Ne deriva che, se non si inserisce e non si fa eseguire una istruzione del tipo:

```
UnLoad UserForm
UnLoad Me
UserForm.Hide
Me.Hide
End
```

nel form, il controllo non ritornerà più alla routine che, da un modulo del progetto o da una macro, ha attivato il form.
Invece lo stesso codice mostrato sopra, nel caso di form VB6, funziona, perché il form è accessibile dalla routine che lo ha attivato.

❍ Il modello di un documento Word

Ogni documento Word fa riferimento ad un modello. Il modello di default è *Normal.dot*; il documento possiede la impaginazione, gli stili e le macro del modello.
Anche un documento Excel può essere creato in riferimento ad un modello; è però molto più difficoltoso cambiare il modello di riferimento con un altro.

❍ Il progetto di un documento Word

Ad ogni documento è associato un "Project" visibile nella finestra in alto a sinistra, composto di diversi gruppi di oggetti (in certe versioni di Office questi oggetti sono raggruppati in moduli):
Gruppo "Moduli", comprendenti il modulo "NewMacros" che viene creato automaticamente quando si registra una macro e altri moduli creati dall'utente mediante VBA→Menu→Inserisci→Modulo. Sia in NewMacros che nei moduli creati dall'utente è possibile inserire routines mediante VBA→Menu→Inserisci→Routine e definire delle variabili pubbliche (tramite la parola chiave "Public" o private).
Oggetto "ThisDocument": è un singolo oggetto, non un gruppo, che contiene nella sua scheda delle proprietà varie impostazioni sul documento.
Gruppo "Moduli di classe", che l'utente può inserire nel progetto tramite VBA→Menu→Inserisci→Modulo di classe, che fanno la stessa cosa dei moduli ordinari ma in modo più complicato.
Gruppo dei Forms, che il programmatore utilizza per dialogare con l'utente. Ogni form può contenere del codice, di solito in una routine di evento (=che si attiva quando si verifica un evento riguardante il form, es. quando si clicca su di esso) o in una routine non di evento.
Riferimento al modello.

Come creare un nuovo modello di documento

Si può creare un modello da un qualsiasi documento Word mediante Word→Menu→Salva con nome impostando il tipo file a "modello di documento" e dando un nome al modello, impostando i margini e tutti i settaggi desiderati e poi chiudendolo.
Un modello di documento si può aprire come un file word per modificarne le impostazioni.

Si attribuisce un (nuovo) modello ad un documento mediante Word→Menu→Strumenti→Modelli e aggiunte. Quando compare il box "Modelli e aggiunte" si preme il pulsante "Applica", si seleziona il modello desiderato, e si preme prima il tasto "Apri" e poi il tasto "Ok" del box di dialogo. Questo trasferisce macro e forms; se si vogliono trasferire anche gli stili occorre servirsi del pulsante "Libreria" dello stesso box di dialogo o mettere la spunta alla opzione "Aggiorna automaticamente gli stili del documento".

Cosa sono le macro

Una macro può essere scritta come routine o registrata tramite il registratore di macro, ovvero si può trovare nella libreria delle macro di Word.
Una macro registrata col registratore delle macro è una sequenza di operazioni corrispondente alla pressione di una sequenza di tasti.
Una macro scritta come routine è un codice contenuto in uno dei moduli del progetto del modello o del documento, che utilizza le variabili e le procedure di VBA per interagire col documento.
Una macro registrata o scritta si può essere attivata da una combinazione di tasti (es. CTRL+C) oppure con una icona che word consente di generare e inserire in una barra comandi.
Una macro della libreria macro è accessibile tramite il form che compare digitando ALT-F8 da dentro un documento Word (vedi più avanti).
Una macro può essere memorizzata in un modulo del progetto relativo al documento (e allora sarà accessibile solo da quel documento, ovvero in un modulo di un modello (e allora è accessibile da tutti i

documenti basati su quel modello). Le macro della libreria Word sono accessibili da qualsiasi documento.

- **Come far apparire l'elenco delle macro di un documento Word ed accedere al progetto del documento**

Per far apparire l'elenco delle macro di un documento Word si utilizza ALT–F8

- **Come creare una routine che viene eseguita automaticamente all'apertura del documento Word**

Se nel modulo del progetto di un documento Word si crea una routine chiamata AUTOOPEN questa si attiverà automaticamente all'apertura del documento

- **Come creare o eliminare subroutines che possono essere richiamate dal codice della macro**

Entro un modulo di progetto si può inserire una routine o una funzione tramite VBA➞Menu➞Inserisci➞Routine (o Function) e determinando se si vuole che la routine sia *Private* (cioè visibile solo entro il modulo) o *Public* (cioè visibile alle routine dell'intero progetto)
Entro un modulo di progetto una routine o una macro può chiamare una qualsiasi subroutine *Private* o *Public* con l'istruzione:
Call SubRoutine
Sia le routines che le macro di un modulo si possono eliminare senza problemi mediante la cancellazione, nel modulo, del loro codice.

- **Routines e variabili *Public* e *Private***

Le routines e le variabili "Public" dei moduli non di classe sono visibili da tutti gli altri moduli. Per visualizzarne l'elenco è sufficiente scrivere il nome del modulo seguito da un punto.
Le variabili "private" di un modulo sono visibili da tutte le routines del modulo ma non da routines di altri moduli.

- **Come creare una macro per mezzo del registratore delle macro**

Per creare una macro si può usare il registratore delle macro (fare molta attenzione, nel box "Registra Macro" a dove viene memorizzata la macro (casella "Memorizza macro in" e casella "Salva le modifiche in" del Box di assegnazione dei tasti alla macro).
Dopo aver creato la macro, mediante il box "Macro" che compare digitando ALT-F8 è possibile vedere dove è stata messa la macro ed eventualmente cancellarla e registrarla nuovamente immettendola nel modulo desiderato.
Attenzione: E' sempre preferibile dare ad una macro il nome della combinazione di tasti che le si associa, perché poi diviene difficile scoprire quale combinazione le si è associata.

❍ Come creare una macro scrivendo una routine in un modulo del progetto e assegnandogli una combinazione di tasti

Per creare la macro si può anche creare una routine *Public* mediante VBA➞Menu➞Inserisci➞Routine e poi assegnarle una combinazione di tasti da Word mediante Word➞Menu➞Strumenti➞Personalizza➞Tastiera: nella casella "Categorie" del box "Personalizza tastiera" che in tal modo appare, selezionare "Macro"; la casella di destra prenderà allora il nome "Macro" e mostrerà le macro. Dopo aver selezionato una macro, utilizzando la casella "Nuova combinazione" si può attribuirle una combinazione di tasti, badando che non risulti già assegnata ad altra macro o comando (questo provocherà il non-funzionamento della combinazione dei tasti: in tal caso occorre andare a rimuovere la combinazione di tasti dal comando o macro che la possedevano, prima di assegnarli alla nostra macro).

❍ Le librerie necessarie perché le macro funzionino

Perché le routine VBA funzionino correttamente occorre assicurarsi che siano selezionate, dal menu VBA➞Menu➞Strumenti➞Riferimenti anche le seguenti librerie:

Microsoft Word Object Library
Microsoft Office Object Library
Microsoft Forms Object Library
Microsoft ActiveX Data Objects Library

❍ Utilizzare il registratore delle macro per imparare VBA e VB6

Se si vuole conoscere le istruzioni per compiere una determinata azione entro un documento Office (es. Word) si può registrare una macro tramite Word➞Strumenti➞Macro➞Registra nuova macro e andare poi a vedere il codice che VBA vi ha inserito.

❍ Come entrare in VBA per Word

Si può entrare in VBA da un qualsiasi documento Office. Da un documento Word si può entrare in tre modi: a) Word➞Menu➞Strumenti➞Macro➞Visual Basic Editor; b) ALT-F11; c) ALT-F8 seguito dalla pressione del pulsante "Modifica".

❍ Come visualizzare tutte le macro di un documento Word

Per visualizzare tutte le macro di un documento Word e scriverne un breve commento occorre digitare ALT-F8
Nel Box "Macro" che appare c'è la finestra "Macro in" che permette di vedere le macro presenti nel modello del documento (di solito Normal.dot) e quelle presenti
Nel box compaiono sia le macro che le routines "Public" (non quelle "Private") presenti nei moduli del progetto del documento.

- **Come attivare le macro nel caso compaia il messaggio che esse sono inattivate per motivi di protezione da virus**

Se compare il messaggio che le macro sono inattivate occorre settare a "Basso" il livello di protezione tramite Word➞Menu➞Strumenti➞Macro➞Protezione➞Bassa
Attenzione: dopo aver impostato la protezione, occorre chiudere e riaprire Word perché la modifica abbia effetto.

- **La scheda proprietà di un modulo o form**

Ogni modulo o form e l'oggetto ThisDocument hanno una "Scheda delle proprietà" che di solito è visibile in basso a sinistra (se non visibile la si può rendere visibile selezionando il modulo nella finestra del progetto e scegliendo VBA➞Menu➞Visualizza➞Finestra proprietà
La scheda delle proprietà è molto importante per un form, ma è utile anche per un modulo, ad es. per rinominarlo: basta cambiare la proprietà "Name" nella scheda.

- **Come sbloccare una macro che si interrompe con segnalazione di errore ed apertura dell'editor VBA**

Se una macro si blocca per una qualsiasi ragione e la esecuzione rimane sospesa (spesso compare il messaggio "impossibile eseguire la macro in modalità interruzione") occorre resettarla mediante VBA➞Menu➞Esegui➞Ripristina
Se qualche modifica alle impostazioni di un file o ad una macro non dà il risultato atteso, si può chiudere e riaprire il file (Word➞Menu➞Chiudi) oppure chiudere e riaprire l'applicativo (Word➞Menu➞Esci).

- **Il font *UniCode***

Le ultime versioni di Office supportano lo standard Unicode di codifica dei caratteri. A differenza dello standard ASCII che per la codifica ha a disposizione solo 8 bit, per un totale di 256 caratteri, la codifica Unicode ha a disposizione 16 bit, per un totale di 64.000 caratteri (individuati dai numeri che vanno da –32.000 a + 32.000). Il solo font Office codificato Unicode è *Arial Unicode MS*; impostando un simile font e selezionando Word➞Menu➞Inserisci➞Simbolo si avrà accesso ad una enorme libreria di caratteri, comprendente l'alfabeto latino, greco, Cirillico, Armeno, Cinese, Giapponese, Indonesiano, nonché ampi set di caratteri matematici e tecnici.

- **La Guida in linea di VBA**

Il significato e l'uso di una istruzione o di una parola chiave che si utilizza per scrivere codice in un modulo o macro VBA viene mostrato dalla scheda "Guida in linea di Visual Basic" che compare digitando F1 quando il cursore è sull'istruzione.

❍ Come accedere ad un database Access da un documento Word tramite una macro in codice VBA

Ci sono due diverse librerie per accedere ad un database da un documento Word: le librerie DAO (Microsoft Data Access Objects) e ADO (Microsoft ActiveX Data Object). I comandi DAO sono più semplici, ma i comandi ADO sono quelli che vengono attualmente sviluppati e migliorati da Microsoft e sono più flessibili e potenti.
Con il metodo ADO MyRecordset.Fields("AUTORE").Value si ottiene il valore del campo autore del record su cui ci si è posizionati. Attenzione: non mettere "AUTORE" tra parentesi quadre come in una istruzione SQL: la stringa deve contenere il nome preciso attribuito al campo della tabella che si è aperta.
Con le seguenti istruzioni ADO:

```
MyRecordset.MoveFirst
CriteriDiRicerca = " AUTORE = " & Chr(34) & "Russell" & Chr(34)
NumeroRecordDaSaltarePrimaDiIniziareLaRicerca = 0
DirezioneDiRicerca = adSearchForward
PuntoInizioRicerca = 1
MyRecordset.Find(CriteriDiRicerca,NumeroRecordDaSaltarePrimaDiIniziareLaRicerca,
DirezioneDiRicerca,PuntoInizioRicerca)
If MyRecordSet EOF then
 msgbox("La ricerca non è andata a buon fine: infatti si è arrivati alla End Of File")
EndIf
```

Se non si è arrivati alla EOF (End Of File) ci si ritrova posizionati sul record che risponde al criterio.
Ma questa istruzione funziona bene solo con campi numerici, mentre non funziona con campi stringa.
Si noti che l'istruzione DAO di apertura di una tabella di database può contenere una istruzione SQL che consente di filtrare i dati:

```
MyRecordset.Open Source:="SELECT * FROM [LIBRI] ORDER BY [LIBRI].[AUTORE]",
ActiveConnection:=MyConnection, CursorType:=adOpenKeyset
```

Nelle istruzioni ADO per la connessione a un database la istruzione *MyConnection.State* dà valore zero se la apertura del database non è riuscita.

❍ Passaggio di parametri ad una routine per valore o per riferimento

Il passaggio di parametri ad una routine può avvenire *ByRef* e *ByVal*.
Nel caso delle istruzioni:

```
Private Sub RoutineChiamante
 Dim Numero as Integer
Numero = 10
 MsgBox(DividiPerDue(Numero))
 Msgbox(Numero)
End Sub

Private Function DividiPerDue(ByRef Dividendo)
 DividiPerDue = Dividendo/2
endif
```

Produce sullo schermo il messaggio "5" (il valore della divisione) seguito dal messaggio "5" (il valore attuale della variabile *Numero*.
Nel caso delle istruzioni:

```
Private Sub RoutineChiamante
 Dim Numero as Integer
Numero = 10
 MsgBox(DividiPerDue(Numero))
 Msgbox(Numero)
End Sub

Private Function DividiPerDue(ByVal Dividendo)
 DividiPerDue = Dividendo/2
endif
```

Produce sullo schermo il messaggio "5" (il valore della divisione) seguito dal messaggio "10" (il valore attuale della variabile *Numero*.
Nel primo caso, con l'istruzione *ByRef* è stata passata alla funzione proprio la variabile *Numero*, che la funzione ha diviso per due. In altre parole, la funzione ha utilizzato per i calcoli la stessa variabile *Numero* utilizzata da *RoutineChiamante*.
Nel secondo caso, con l'istruzione *ByVal* è stata passato alla funzione solo il valore della variabile *Numero*. Questo valore è stato immesso nella variabile *Dividendo* della funzione *DividiPerDue*, e questa variabile non coincide con *Numero*, ma è una sua copia. Perciò *Numero* non risulta diviso per due.

❍ Generare un carattere a partire da un codice di carattere

Per creare un carattere a partire da un codice o ottenere un codice a partire da un carattere si possono usare le istruzioni *Chr(CodiceCarattere)* e *Asc(Carattere)* oppure le istruzioni – che funzionano molto meglio – *ChrW(CodiceCarattere)* e *AscW(Carattere)*. Per ottenere il codice *ChrW* di un carattere è sufficiente scrivere in una macro il codice

```
ALT_N()
 MsgBox(AscW(Selection.Range.Characters(1))
End Sub
```

Alla digitazione di ALT-N comparirà a schermo il valore *AscW* del primo carattere della stringa evidenziata sullo schermo.

❍ Debugging di un codice VBA

Si può far eseguire un codice passo-passo impostando un punto di interruzione in prossimità di una istruzione (cliccando sul margine alla sua sinistra per farvi apparire un punto) e poi, una volta che il programma si è arrestato a quel punto, far eseguire una istruzione alla volta con *F8*. Si può passare dal codice al documento Word per vedere gli effetti delle istruzioni, ma senza cliccare col mouse nel documento, bensì utilizzando per visualizzarne le varie parti i cursori verticali e orizzontali.
Nella Finestra immediata si possono modificare durante la esecuzione del codice i valori delle variabili. Posizionando il cursore sopra il nome di una variabile apparirà un tag che fornisce il valore attuale di quella variabile.

❍ Le molteplici possibilità offerte da VBA

VBA contiene anche istruzioni che consentono di controllare ogni aspetto della stampa di un documento (per vederle si deve registrare una macro con la selezione della stampa di un documento); di creare, aprire o chiudere un file; di aprire un file e posizionarsi al suo interno; di stabilire se in una

data directory vi sia un file; di modificare lo zoom dello schermo; di inserire una tabella, dimensionarla e riempirla; di copiare un paragrafo di Word nel campo di un database; di creare un database; di cancellare o settare a *True* in serie i campi di un database; di spostarsi tra i record e i campi di un database; di eliminare da un documento Word tutti i paragrafi con un dato colore o con una data dimensione o con un dato font; di ricercare una stringa utilizzando i caratteri jolly; di attribuire uno stile ad un paragrafo; di creare dei riferimenti ipertesto (*ALT-X*) all'interno di un documento Word o tra documenti Word; di inserire un riferimento bibliografico nel punto del documento che si desidera (*ALT-B*); copiare dei record da un database ad un altro (l'operazione da menu è più difficile di quanto si creda); eseguire operazioni di filtraggio di un database mediante istruzioni SQL e visualizzare il risultato in un file Word; utilizzare un Database Access per memorizzare dei quiz da far apparire in un file Word; scrivere un documento Word in modo automatico (es. un test a risposte multiple, partendo da un database Access); e molte altre cose ancora.

❍ Istruzioni ADO per ricercare un determinato record nel *RecordSet*

Per cercare, con le istruzioni ADO, un determinato record nel RecordSet, l'istruzione MyRecordset.Find, che funziona bene per la ricerca di un dato valore in campi numerici, non funziona per la ricerca in campi stringa. In questo secondo caso è opportuno usare il seguente codice:

```
1   MyRecordset.MoveFirst
2  MyRecordset.Move (-1)
3  For Puntatore = 1 To MyRecordset.RecordCount + 1
4   MyRecordset.Move (1)
5   If Not (IsNull(MyRecordset.Fields("autore"))) Then
6    If MyRecordset.EOF Then Exit For
7    If MyRecordset.Fields("AUTORE") = AutoreDaCercare Then Exit For
8   End If
9  Next Puntatore
10  If MyRecordset.EOF Then
11          MsgBox ("IL NOME INDICATO NON COMPARE NEL DATABASE COME AUTORE")
12  Else
13   MsgBox (MyRecordset.Fields("AUTORE") & ", " & MyRecordset.Fields("TITOLO") & ", " &
14   MyRecordset.Fields("EDITORE"))
15 End If
```

La istruzione della riga 4 fa muovere, ad ogni loop for..next il puntatore dei record avanti di un record; in tal modo viene percorso l'intero insieme dei record.
La istruzione 6 fa uscire dal ciclo se verifica che si sia raggiunta la fine del file (EOF: End Of File)
La istruzione 7 fa uscire dal ciclo se verifica che il nome dell'autore cercato corrisponde a quello dell'autore contenuto nel record corrente e in caso affermativo fa uscire dal loop.
La condizione 5 serve ad evitare errori che sarebbero generati dall'istruzione 7 se il campo autore fosse un campo NULL.
La istruzione 10 verifica se si è o meno raggiunta la fine del file. In caso affermativo il record non è stato trovato e viene visualizzato a schermo il relativo messaggio.
Si noti la differenza tra la istruzione MyRecordset.Move(1) e MyRecordset.MoveFirst: la prima di esse attua uno spostamento *relativo*, cioè di un record a partire da quello su cui si è attualmente; invece la seconda opera uno spostamento *assoluto* al primo record.

❍ Istruzioni ADO per aggiungere un record ad un database Access

Per aggiungere un record ad un database si usano le seguenti istruzioni ADO:

1 MyRecordset.AddNew
2 MyRecordset.Fields("AUTORE").Value = "Russell"
3 MyRecordset.Fields("TITOLO").Value = "An Inquiry into Meaning and Truth"
4 MyRecordset.Fields("EDITORE").Value = "Oxford University Press"
5 MyRecordset.Update

Con le istruzioni 2, 3, 4 si scrive nel nuovo record. L'operazione deve essere conclusa con l'istruzione 5.

I metodi dell'oggetto *Selection*

Le istruzioni *Selection*… sono importantissime per manipolare il documento. Qui di seguito se ne dà una breve illustrazione.

Selection.Range
E' un oggetto corrispondente al testo evidenziato nel documento. Consente di accedere sia ai caratteri dell'area evidenziata sia alla sua formattazione, sia alla sua posizione nel documento. Se non è stata evidenziata alcuna area (cursore ridotto a punto di inserimento) consente tuttavia di avere informazioni relative al paragrafo in cui si trova.

Selection.Range.Text
E' la stringa di caratteri contenuta nella parte evidenziata

Selection.Range.Characters(1)
Restituisce il primo carattere dell'area evidenziata

Selection.Paragraphs(1).Range
InizioSelezione = Selection.Range.Start
Scrive nella variabile InizioSelezione il numero che corrisponde alla posizione dell'inizio dell'area attualmente selezionata entro il documento

InizioSelezione = Selection.Range.End
Scrive nella variabile InizioSelezione il numero che corrisponde alla posizione della fine dell'area attualmente selezionata entro il documento
Individua il primo paragrafo della selezione e lo seleziona tutto

Selection.SetRange Start:=100, End:=200
Evidenzia la porzione di testo che va dalla posizione 100 del documento alla posizione 200 del documento. I numeri che individuano una posizione nel documento non corrispondono ai caratteri.

Selection.InsertAfter "Topolino"
Scrive "Topolino" alla fine della selezione

Selection.Typetext "Topolino"
Scrive "Topolino" al posto della stringa selezionata nel documento.

Selection.Range.Font.Name
Restituisce il font (Times New Roman, Arial… ecc.) del primo carattere dell'area evidenziata

Selection.Range.ParagraphFormat.LeftIndent
Restituisce la distanza dal margine del foglio del margine sinistro del corpo di scrittura

Selection.Range.ParagraphFormat.FirstLineIndent

Restituisce il rientro che la prima riga ha rispetto alla posizione delle righe successive del corpo di scrittura. Così, se *LeftIndent* = 2 e *FirstLineIndent* = –1 questo vuol dire che il paragrafo è allineato a sinistra a 2 cm. dal margine del foglio, tranne la prima riga che è allineata a 2 – 1 = 1 cm. dal margine dal foglio (prima riga "sporgente" del menu "Paragrafo").

Selection.Range.Characters.Count
Ci dice di quanti caratteri è composta la selezione

❍ La strutturazione di un documento Word in paragrafi e le relative istruzioni VBA

Ogni documento è visto da VBA come un insieme di caratteri, ognuno dei quali ha un numero, o come un insieme di linee, o come un insieme di paragrafi. Quello dei paragrafi è l'insieme più interessante e utile. La istruzione *ActiveDocument.Paragraphs(1)* individua il primo paragrafo del documento. Una volta individuato un paragrafo si può accedere ai suoi caratteri e alla sua formattazione con l'oggetto *Range*: *ActiveDocument.Paragraphs(1).Range.Text* è ad esempio la stringa di caratteri contenuta nel paragrafo.
La istruzione *ActiveDocument.Paragraphs(3).Range.Characters.Count* ci dice ad es. di quanti caratteri è composto il terzo paragrafo del documento.
La istruzione *ActiveDocument.Paragraphs(2).Range.ParagraphFormat.LeftIndent* ci dice a che distanza dal margine sinistro è allineato il secondo paragrafo del documento.
La istruzione *ActiveDocument.Paragraphs(2).Range.Characters(3).Font.Size* ci dice la grandezza del terzo carattere (8,10, 12, ecc.) del secondo paragrafo del documento.

❍ Istruzioni VBA per cercare una stringa nel documento

La istruzione *Selection.Find* consente di cercare una stringa nel documento, esattamente come l'opzione "Trova" del menu. Per sapere come funziona è sufficiente (a parte richiamare la guida in linea VBA digitando F1 quando il cursore è sulla parola *Find* di *Selection.Find*) regitrare in una macro una operazione di ricerca tramite l'opzione "Trova" del menu "Modifica" di Word.

❍ Come dotare un foglio Excel di una nuova funzione

Si può creare una funzione Excel utilizzabile nelle formule delle celle come se fosse una delle sue funzioni logiche o matematiche entrando nell'editor di VBA con ALT-F11, creando un modulo mediante VBA→Menu→Inserisci→Modulo e nel modulo creando una Function mediante VBA→Menu→Inserisci→Routine e selezionando "Function" e "Public". Una funzione deve dichiarare il valore che produce come risultato e i parametri che accetta in ingresso, nel modo seguente:
Public Function DividiPerDue(Dividendo as Integer, Divisore as Integer) as Integer
DividiPerDue = Dividendo/Divisore
End Sub
Si noti che il valore viene prodotto (nell'unica istruzione della Function) attribuendolo al nome della Funzione.
A questo punto, in un foglio Excel si può scrivere una formula del tipo:
=DividiPerDue(B5;B6)