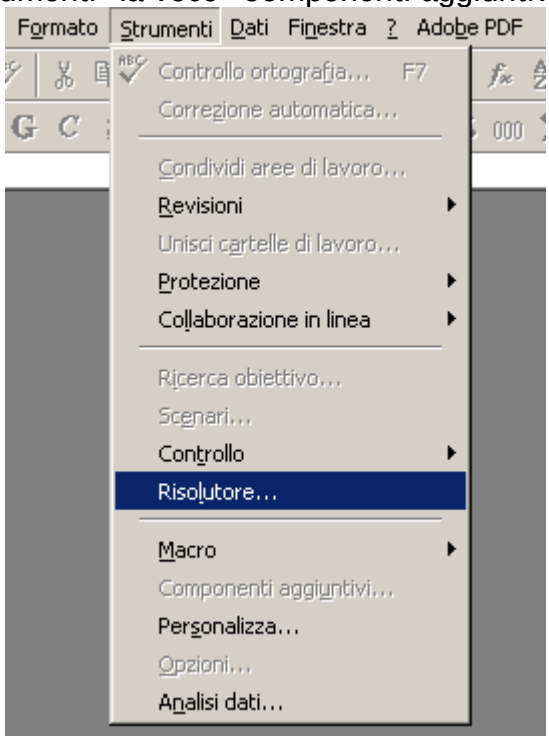
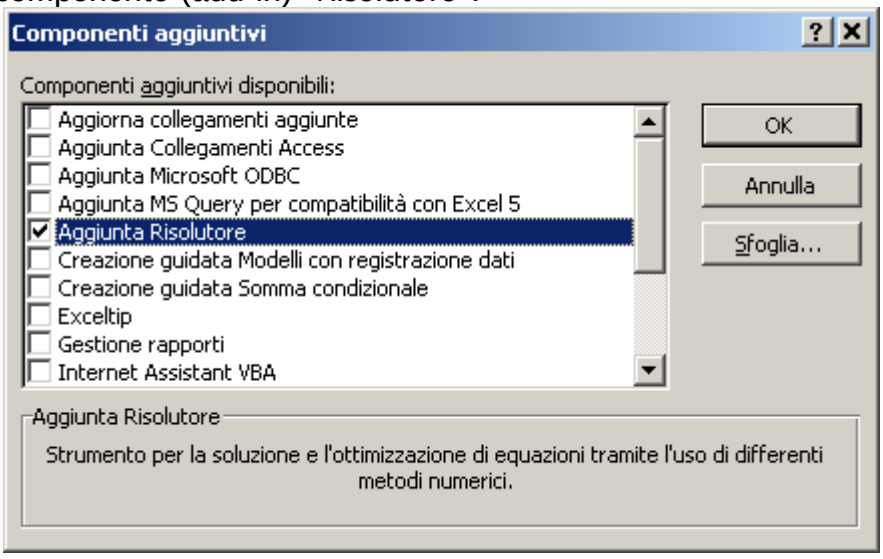
HOW TO 01: Risolvere un'equazione in un'incognita con Excel

Selezionare dal menù "Strumenti" la voce "Componenti aggiuntivi..."



Aggiungere il componente (add-in) "Risolutore":



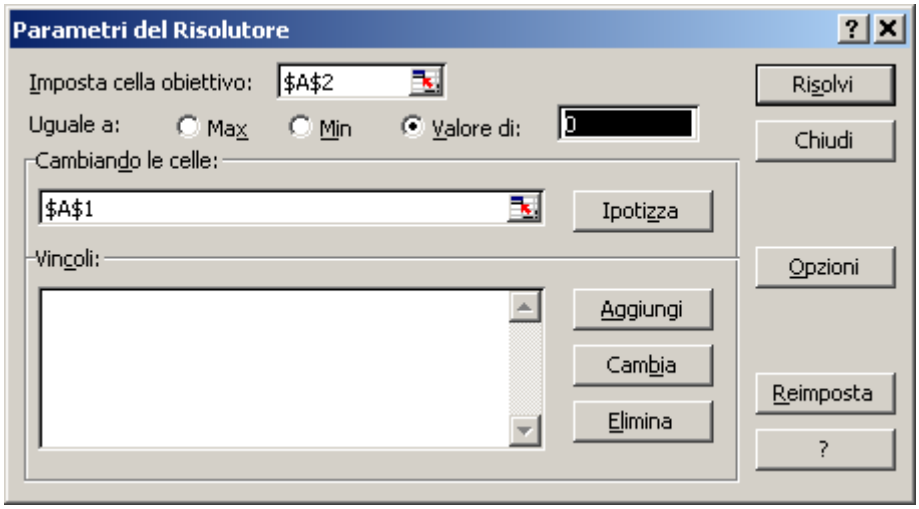
Tornare al foglio di Excel e proporsi di risolvere l'equazione: $3x^2 - 8x + 2 = 0$

Si sceglie di porre nella cella A1 un valore di primo tentativo per la variabile x rispetto cui risolvere l'equazione. Nella cella A2 viene scritta l'equazione dipendente dalla cella A1:

	A	B
1	6	
2	=3*A1^2-8*A1+2	
3		
4		

Si selezioni ora il menù: Strumenti → Risolutore.

Dato che si deve azzerare la cella A2 che è funzione della cella A1 ecco i valori da imporre:



I riferimenti alle celle obiettivo e variabile vengono ottenuti cliccando con il mouse sulle rispettive celle di cui sopra. Si noti che il problema, essendo di azzeramento, richiede di impostare il valore della cella obiettivo pari a zero.

Il valore finale ottenuto è:

	A	B
1	2.387425851	
2	-2.25623E-07	
3		

quindi una delle due soluzioni del problema è: $x = 2.387425851$.

N.B.: il Risolutore in realtà può anche risolvere dei piccoli problemi di ottimo (minimo/massimo) vincolato.

HOW TO 02: Excels Keyboard Shortcuts

This list gives you a glance of (nearly) all shortcut combinations in Excel. Please take the time to read it. I'm sure you'll find many of them useful. (http://www.asap-utilities.com/)

Shortcut key	Action	Menu equivalent comments	Version
Ctrl+A	Select All	None	All
Ctrl+B	Bold	Format, Cells, Font, Font Style, Bold	All
Ctrl+C	Copy	Edit, Copy	All
Ctrl+D	Fill Down	Edit, Fill, Down	All
Ctrl+F	Find	Edit, Find	All
Ctrl+G	Goto	Edit, Goto	All
Ctrl+H	Replace	Edit, Replace	All
Ctrl+I	Italic	Format, Cells, Font, Font Style, Italic	All
Ctrl+K	Insert Hyperlink	Insert, Hyperlink	Excel 97/2000
Ctrl+N	New Workbook	File, New	All
Ctrl+O	Open	File, Open	All
Ctrl+P	Print	File, Print	All
Ctrl+R	Fill Right	Edit, Fill Right	All
Ctrl+S	Save	File, Save	All
Ctrl+U	Underline	Format, Cells, Font, Underline, Single	All
Ctrl+V	Paste	Edit, Paste	All
Ctrl W	Close	File, Close	Excel 97/2000
Ctrl+X	Cut	Edit, Cut	All
Ctrl+Y	Repeat	Edit, Repeat	All
Ctrl+Z	Undo	Edit, Undo	All
F1	Help	Help, Contents and Index	All
F2	Edit	None	All
F3	Paste Name	Insert, Name, Paste	All
F4	Repeat last action	Edit, Repeat. Works while not in Edit mode.	All
F4	While typing a formula, switch between absolute/relative refs	None	All
F5	Goto	Edit, Goto	All
F6	Next Pane	None	All
F7	Spell check	Tools, Spelling	All
F8	Extend mode	None	All
F9	Recalculate all workbooks	Tools, Options, Calculation, Calc,Now	All
F10	Activate Menubar	N/A	All
F11	New Chart	Insert, Chart	All
F12	Save As	File, Save As	All
Ctrl+:	Insert Current Time	None	All
Ctrl+;	Insert Current Date	None	All
Ctrl+"	Copy Value from Cell Above	Edit, Paste Special, Value	All

Ctrl+'	Copy Formula from Cell Above	Edit, Copy	All
Shift	Hold down shift for additional functions in Excel's menu	none	Excel 97/2000
Shift+F1	What's This?	Help, What's This?	All
Shift+F2	Edit cell comment	Insert, Edit Comments	All
Shift+F3	Paste function into formula	Insert, Function	All
Shift+F4	Find Next	Edit, Find, Find Next	All
Shift+F5	Find	Edit, Find, Find Next	All
Shift+F6	Previous Pane	None	All
Shift+F8	Add to selection	None	All
Shift+F9	Calculate active worksheet	Calc Sheet	All
Shift+F10	Display shortcut menu	None	All
Shift+F11	New worksheet	Insert, Worksheet	All
Shift+F12	Save	File, Save	All
Ctrl+F3	Define name	Insert, Names, Define	All
Ctrl+F4	Close	File, Close	All
Ctrl+F5	XL, Restore window size	Restore	All
Ctrl+F6	Next workbook window	Window, ...	All
Shift+Ctrl+F6	Previous workbook window	Window, ...	All
Ctrl+F7	Move window	XL, Move	All
Ctrl+F8	Resize window	XL, Size	All
Ctrl+F9	Minimize workbook	XL, Minimize	All
Ctrl+F10	Maximize or restore window	XL, Maximize	All
Ctrl+F11	Inset 4.0 Macro sheet	None in Excel 97. In versions prior to 97 - Insert, Macro, 4.0 Macro	All
Ctrl+F12	File Open	File, Open	All
Alt+F1	Insert Chart	Insert, Chart...	All
Alt+F2	Save As	File, Save As	All
Alt+F4	Exit	File, Exit	All
Alt+F8	Macro dialog box	Tools, Macro, Macros in Excel 97 Tools,Macros - in earlier versions	Excel 97/2000
Alt+F11	Visual Basic Editor	Tools, Macro, Visual Basic Editor	Excel 97/2000
Ctrl+Shift+F3	Create name by using names of row and column labels	Insert, Name, Create	All
Ctrl+Shift+F6	Previous Window	Window, ...	All
Ctrl+Shift+F12	Print	File, Print	All
Alt+Shift+F1	New worksheet	Insert, Worksheet	All
Alt+Shift+F2	Save	File, Save	All
Alt+=	AutoSum	No direct equivalent	All
Ctrl+`	Toggle Value/Formula display	Tools, Options, View, Formulas	All
Ctrl+Shift+A	Insert argument names into formula	No direct equivalent	All
Alt+Down arrow	Display AutoComplete list	None	Excel 95
Alt+'	Format Style dialog box	Format, Style	All
Ctrl+Shift+~	General format	Format, Cells, Number, Category, General	All
Ctrl+Shift+!	Comma format	Format, Cells, Number, Category,	All

		Number	
Ctrl+Shift+@	Time format	Format, Cells, Number, Category, Time	All
Ctrl+Shift+#	Date format	Format, Cells, Number, Category, Date	All
Ctrl+Shift+$	Currency format	Format, Cells, Number, Category, Currency	All
Ctrl+Shift+%	Percent format	Format, Cells, Number, Category, Percentage	All
Ctrl+Shift+^	Exponential format	Format, Cells, Number, Category,	All
Ctrl+Shift+&	Place outline border around selected cells	Format, Cells, Border	All
Ctrl+Shift+_	Remove outline border	Format, Cells, Border	All
Ctrl+Shift+*	Select current region	Edit, Goto, Special, Current Region	All
Ctrl++	Insert	Insert, (Rows, Columns, or Cells) Depends on selection	All
Ctrl+-	Delete	Delete, (Rows, Columns, or Cells) Depends on selection	All
Ctrl+1	Format cells dialog box	Format, Cells	All
Ctrl+2	Bold	Format, Cells, Font, Font Style, Bold	All
Ctrl+3	Italic	Format, Cells, Font, Font Style, Italic	All
Ctrl+4	Underline	Format, Cells, Font, Font Style, Underline	All
Ctrl+5	Strikethrough	Format, Cells, Font, Effects, Strikethrough	All
Ctrl+6	Show/Hide objects	Tools, Options, View, Objects, Show All/Hide	All
Ctrl+7	Show/Hide Standard toolbar	View, Toolbars, Stardard	All
Ctrl+8	Toggle Outline symbols	None	All
Ctrl+9	Hide rows	Format, Row, Hide	All
Ctrl+0	Hide columns	Format, Column, Hide	All
Ctrl+Shift+(	Unhide rows	Format, Row, Unhide	All
Ctrl+Shift+)	Unhide columns	Format, Column, Unhide	All
Alt or F10	Activate the menu	None	All
Ctrl+Tab	In toolbar: next toolbar	None	Excel 97/2000
Shift+Ctrl+Tab	In toolbar: previous toolbar	None	Excel 97/2000
Ctrl+Tab	In a workbook: activate next workbook	None	Excel 97/2000
Shift+Ctrl+Tab	In a workbook: activate previous workbook	None	Excel 97/2000
Tab	Next tool	None	Excel 97/2000
Shift+Tab	Previous tool	None	Excel 97/2000
Enter	Do the command	None	Excel 97/2000
Shift+Ctrl+F	Font Drop Down List	Format, Cells, Font	All
Shift+Ctrl+F+F	Font tab of Format Cell Dialog box	Format, Cells, Font	Before 97/2000

Shift+Ctrl+P	Point size Drop Down List	Format, Cells, Font	All

A special thanks goes out to Allan Bethel who provided most of the shortcuts in this list!

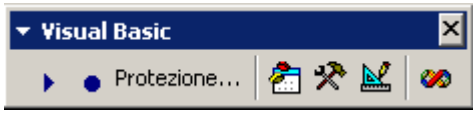
HOW TO 03: Iniziare ad usare le potenzialità di VBA

VBA è l'acronimo di Visual Basic for Applications. È il linguaggio di programmazione della suite Office di Microsoft. Con VBA è possibile creare delle macro, delle vere e proprie routine, delle finestre interattive e degli add-in per i programmi: Word, Excel, PowerPoint, Access.
VBA ha la stessa sintassi di Visual Basic anche se non contiene la totalità delle istruzioni/comandi di tale linguaggio di programmazione. La sovrapposizione formale è comunque molto spinta.

Vediamo una prima applicazione che utilizzando VBA permetta di automatizzare un compito altrimenti ripetitivo e noioso di Excel.

Si desidera creare una tabellina della moltiplicazioni analoga a quelle presenti nell'ultima pagina dei quaderni delle elementari.

- Si lancia Excel e si apre un nuovo file.
- Si rinomina il "foglio 1", cliccando due volte in basso a sinistra, dandogli ad esempio il nome: "Tabellina". In questo modo mnemonicamente sarà più semplice ricordare il riferimento a tal foglio di lavoro.
- Nello spazio libero alla destra dei bottoni della toolbar di Excel, cliccando con il pulsante destro del mouse attiviamo l'opzione: "Visual Basic". Comparirà la seguente finestra:

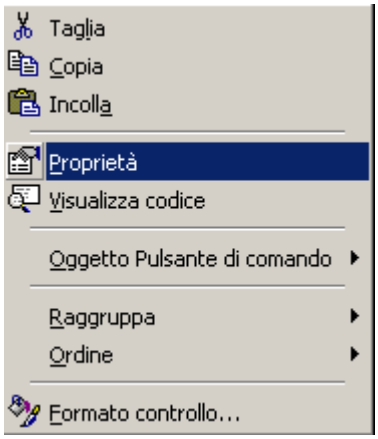


- Cliccando sulla icona "Casella degli strumenti" (martello e chiave inglese) compare una nuova finestra che contiene alcuni dei numerosi strumenti che possono popolare una pagina di Excel o anche una finestra appartenente ad un progetto di Excel:

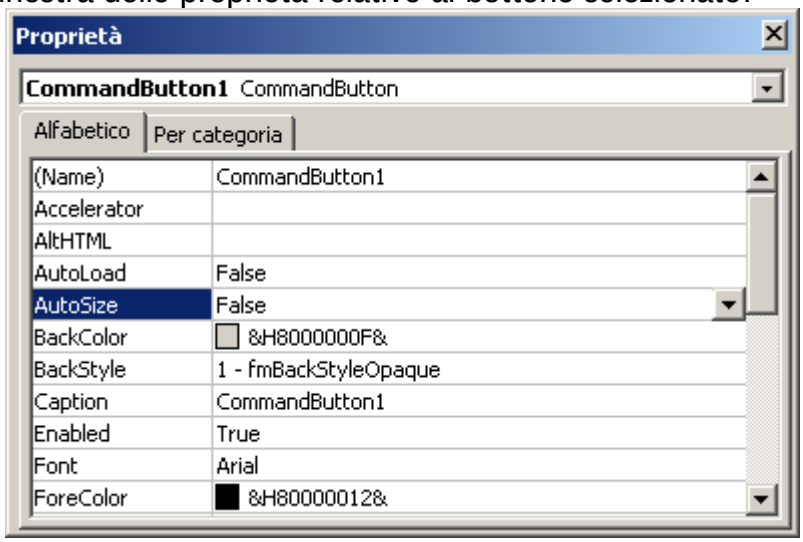
- Creeremo ora un pulsante di comando,  , selezionandolo dalla casella degli strumenti. Tale pulsante verrà introdotto direttamente nella pagina di Excel che abbiamo chiamato "Tabellina". Per fare ciò è sufficiente dopo aver cliccato sul pulsante di comando,  , passare al foglio Excel e tenendo premuto il pulsante sinistro del

mouse disegnare un rettangolo. Rilasciando il pulsante sinistro un nuovo bottone "CommandButton1" compare sulla foglio di Excel:

- Tale bottone può essere ridimensionato. Cambiamo subito il suo nome ed il suo aspetto. Il nome del bottone verrà usato a livello programmativo per riferirsi a tale oggetto. L'aspetto permette un miglioramento della fruibilità da parte dell'utente che utilizza il programma. Per modificare le proprietà del bottone occorre cliccare con il pulsante destro del mouse su di esso e selezionare il menù "Proprietà":

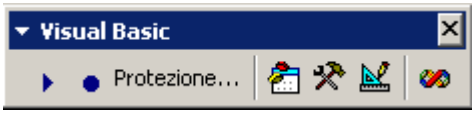


- Compare la finestra delle proprietà relative al bottone selezionato:

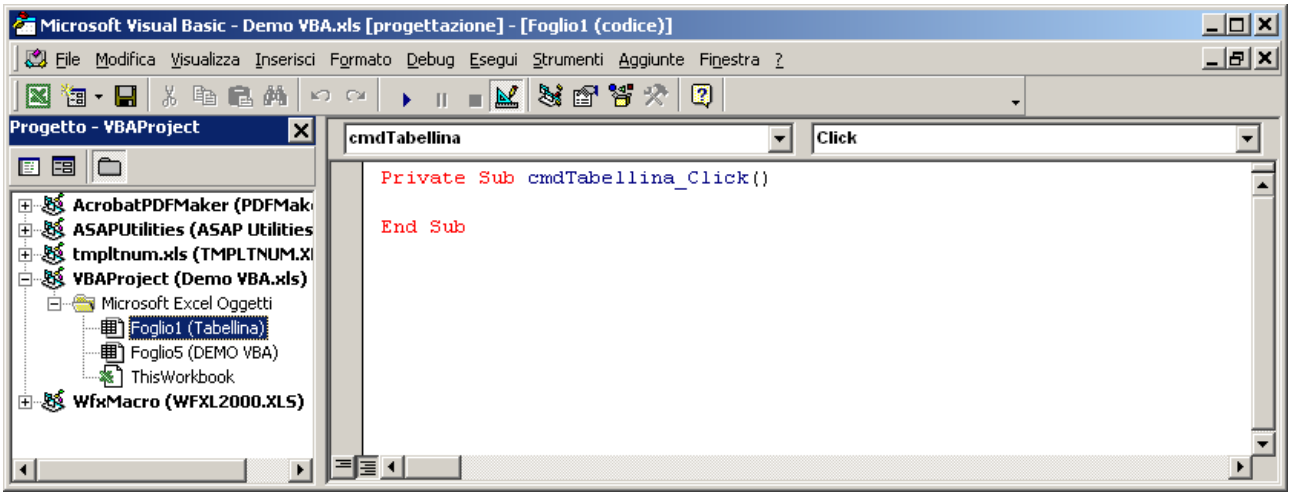


- Andremo a modificare il "Name" del bottone in "cmdTabellina" e la Caption in "Crea tabellina…". È opportuno identificare ogni tipologia di oggetto creato in VBA con un prefisso di tre lettere che riassume la sua operatività. Segue un nome mnemonico che identifica la funzione di quell'oggetto. Un bottone di comando avrà come prefisso "cmd", una casella di testo (text button) "txt", un bottone di scelta (option button) "opt", una immagine (picture box) "pic", ….

- Ecco come appare dopo la modifica il bottone:

- Andremo ora a scrivere il codice VBA necessario a produrre automaticamente la tabellina. Per fare ciò occorre attivare la "Modalità progettazione" appartenente alla finestra 

premendo il bottone con la squadra e la matita. Tale bottone alterna la modalità progettazione (scrittura del codice) con quella esecuzione (test del programma). Cliccando ora due volte sul bottone cmdTabellina compare la finestra di codice di VBA ove inserire le righe di comando per l'esecuzione del codice:



- VBA provvede automaticamente a creare la struttura della routine:

```
Sub cmdTabellina_Click()
```

Il programmatore dovrà scrivere il codice desiderato all'interno di tale contenitore delimitato inferiormente dalla istruzione "End Sub". Ecco di seguito le istruzioni inserite per costruire la tabellina. La tabellina conterrà tutti i prodotti tra 1 e 10, partirà dalla prima casella in alto a sinistra. Occorre sapere che Excel riconosce un riferimento matriciale, riga-colonna, alle celle dei suoi fogli di lavoro. Sicché la cella A1 corrisponde al riferimento (1,1) la cella C2 al riferimento (2,3) e la cella B5 al riferimento (5,2):

cmdTabellina | Click

```
Private Sub cmdTabellina_Click()

   With Worksheets("Tabellina")
      .Activate
      .Cells.Clear
      MsgBox "Ho pulito tutte le celle del foglio di Excel...", vbInformation, "Demo VBA"
      For i = 1 To 10
         For j = 1 To 10
            .Cells(i, j).Value = i * j
         Next j
      Next i
   End With

   MsgBox "Ho terminato di riempire la tabellina!!!", vbInformation, "Demo VBA"

End Sub
```

Analizziamo insieme il codice scritto:

- l'istruzione "With … End With" racchiude altre istruzioni che fanno riferimento all'oggetto Worksheets("Tabellina"). Ogni volta che si vorrà utilizzare una proprietà (qualità) o un metodo (funzione) di tale oggetto basterà scrivere .proprietà oppure .funzione anziché dover ripetere anche il nome esteso dell'oggetto. Tale abbreviazione di codice è appunto permessa dall'istruzione "With".
- L'istruzione Worksheets permette di focalizzare l'attenzione su uno specifico foglio di lavoro, in questo caso il foglio "Tabellina".
- .Activate attiva il foglio specifico (la routine cmdTabellina_Click() potrebbe infatti essere invocata anche da un altro foglio. In questo modo viene visualizzato ed attivato il foglio "Tabellina".
- .Cells.Clear focalizza l'attenzione sulla proprietà Cells ovvero su tutte le celle del foglio ed esegue il metodo Clear ovvero pulisce il loro contenuto (numeri, formule, riferimenti,…).
- DoEvents
- L'istruzione Msgbox"Messaggio",stileMessaggio,"Titolo finestra" visualizza una finestra di messaggio a video.
- Finalmente ha inizio il riempimento della tabellina tramite un doppio ciclo "For … Next"
- .Cells(i,j) punta alla riga i-esima e alla colonna j-esima. Il valore di tale cella sarà pari al prodotto i*j.
- Alla fine si avvisa l'utente che la creazione della tabellina è terminata.

Per eseguire il codice appena creato occorre tornare alla finestra di Excel e attivare l'opzione di esecuzione (test del programma) tramite il bottone:  . Cliccando sul bottone cmdTabellina si vedrà l'effetto dell'esecuzione della routine cmdTabellina_Click().

HOW TO 04: Creare la mia prima macro

In Excel è possibile automatizzare dei compiti ripetitivi al fine di farli eseguire automaticamente al programma anziché doverli effettuare ogni volta a mano.
L'obiettivo della creazione di questa prima macro in Excel è quello di colorare di giallo lo sfondo di una cella, porre il testo in grassetto e colorarlo di rosso.
Per fare ciò scriviamo un testo qualsiasi in una cella qualsiasi di un foglio di Excel. Scriviamo ad esempio: "Euclide". Attiviamo quindi con il mouse una nuova cella.
Excel è in grado di memorizzare le azioni effettuate dall'utente e trascriverle in codice VBA. Excel nel corso della creazione di una macro funziona da registratore e traduttore di codice. Registra ogni azione intrapresa dall'utente all'interno del programma e la traduce alla fine in linguaggio VBA. È quindi possibile editare il codice scritto e analizzarlo, comprenderlo, modificarlo, eliminare le porzioni non interessanti e quindi salvare il tutto.
Per attivare la registrazione della macro occorre selezionare il menù: Strumenti→Macro…→Registra nuova macro…
Alla comparsa della finestra di impostazione della macro digitiamo il nome della macro, ad esempio: "LaMiaPrimaMacro" e premiamo il bottone OK.
Subito compare in primo piano una finestrella con un bottone di stop registrazione:

che andremo ad premere quando avremo terminato di eseguire le azioni che compariranno nel codice della macro.
A questo punto eseguiamo la sequenza di operazioni che ci siamo prefissati per modificare il look della cella specifica ed alla fine premiamo il bottone di "interrompi registrazione".
Attivando VBA tramite i tasti Alt+F11 è possibile osservare il seguente codice:

```
Sub LaMiaPrimaMacro()

    ' LaMiaPrimaMacro Macro
    ' Macro registrata il 29/03/2004 da Davide Manca

    Range("C5").Select
    With Selection.Interior
        .ColorIndex = 6
        .Pattern = xlSolid
    End With
    Selection.Font.Bold = True
    Selection.Font.ColorIndex = 3

End Sub
```

Tale codice è contenuto nel "Modulo1" sotto il gruppo "Moduli" del foglio di Excel su cui abbiamo registrato la macro.
Si nota in verde il commento introdotto automaticamente da Excel.
L'istruzione seguente indica che abbiamo attivato esattamente la cella C5 e che l'abbiamo riempita con il colore 6 (giallo) e che il font è stato reso in grassetto (Bold) tramite l'istruzione "True" e il colore del testo è diventato il numero 3 (rosso).

Per rieseguire la macro occorre selezionare dal menù Strumenti→Macro la voce: Macro... (Alt+F8) e quindi cliccare sulla macro prescelta premendo alla fine il bottone OK.
Se però eseguiamo la macro attivando una cella diversa da quella su cui abbiamo operato in precedenza sembrerà non succedere nulla. Come mai? Dove abbiamo sbagliato? Come mai Excel non risponde?
Nessun problema! Excel sta eseguendo quanto trova scritto nella macro "LaMiaPrimaMacro". La prima istruzione pone la selezione esattamente sulla cella C5 non su altre. Ecco perché, cosiffatta, la macro serve a ben poco. La nostra intenzione è quella di poterla eseguire su qualsiasi cella selezionata dall'utente.
Occorre quindi modificare la macro. Proviamo a registrarla nuovamente avendo già selezionato la cella su cui andremo ad operare modificando soltanto il suo look.
Il codice che si ottiene è:

```
Sub LaMiaPrimaMacro()

    ' LaMiaPrimaMacro Macro
    ' Macro registrata il 29/03/2004 da Davide Manca

    Range("C5").Select
    Selection.Interior.ColorIndex = 6
    Selection.Font.ColorIndex = 3
    Selection.Font.Bold = True

End Sub
```

Il codice appare più compatto anche se i punti salienti praticamente non sono cambiati.
La macro appare ancora non utilizzabile. Proviamo però ad eliminare la prima istruzione:

Range("C5").Select

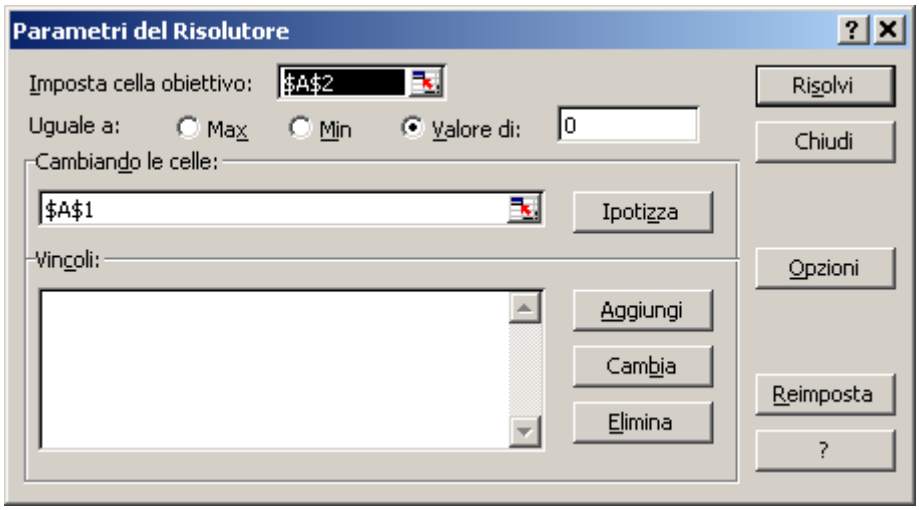
Anziché cancellare completamente la linea di istruzione è sufficiente all'inizio commentarla anteponendo un segno di apice (si noti che il colore della linea cambia diventando verde, ovvero un commento che **non** viene eseguito dall'interprete di VBA).
Se adesso scriviamo qualcosa in una nuova cella e proviamo ad eseguire nuovamente la macro ci accorgeremo che finalmente il tutto funziona. Anche la nuova cella è stata cioè modificata nel suo look analogamente alla cella di partenza: C5.
È ora suggeribile cancellare la riga commentata dato che tutto funziona al fine di rendere più leggibile il codice.
Si noti che la macro così operante è chiamabile da altre porzioni di codice appartenenti allo stesso foglio tramite l'istruzione:

call NomeMacro

nel nostro caso quindi: "call LaMiaPrimaMacro".

HOW TO 05: Creare una macro che utilizzi il Risolutore di Excel

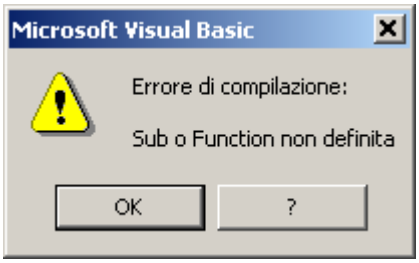
Si supponga di voler automatizzare il compito di risoluzione di un'equazione algebrica esteso ad un certo range di celle.
Occorrerà creare una routine in grado di risolvere una specifica equazione algebrica. Quindi si procederà a eseguire tale routine estendendola e applicandola al range di celle prefissato (ad esempio tramite un ciclo "For… Next").
Questa sezione mostra come risolvere in automatico un'equazione algebrica tramite una macro dedicata utilizzando come variabile la cella A1 e come cella obiettivo (funzione) la cella A2. La macro avrà nome: "AzzeraEq".
Si rimanda al "HOW TO 01" per i dettagli relativi al Risolutore.
Creiamo la macro secondo le modalità comuni introdotte nel "HOW TO 04".
Registriamo quindi le operazioni per azzerare l'equazione tramite la finestra di opzioni del Risolutore:



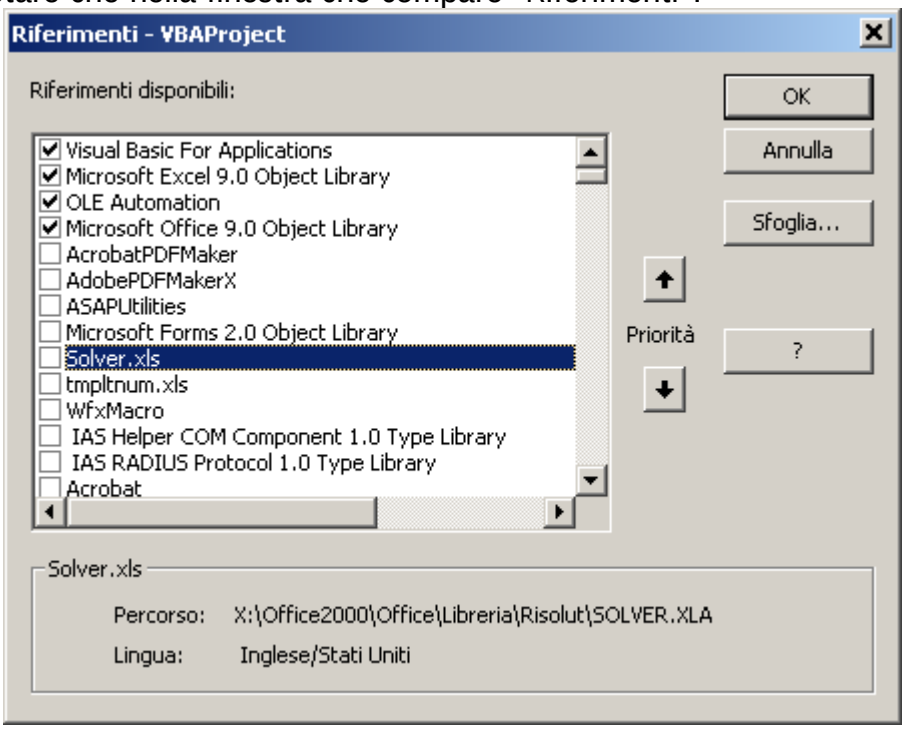
Ecco il codice prodotto da Excel e presente in VBA:

```
Sub AzzeraEq()

    ' AzzeraEq Macro
    ' Macro registrata il 29/03/2004 da Davide Manca

    SolverOk SetCell:="$A$2", MaxMinVal:=3, ValueOf:="0", ByChange:="$A$1"
    SolverSolve

End Sub
```

Se ora proviamo a modificare il valore della cella A1 oppure la struttura dell'equazione contenuta nella cella A2 potremo rieseguire la macro AzzeraEq lavorando appunto su tali celli.
Il problema è che se proviamo ad eseguire la macro "AzzeraEq" il codice non funziona e riceviamo il seguente errore:



Dove abbiamo sbagliato. Può essere di aiuto anche se la spiegazione è sufficientemente criptica premere il bottone di Help "?" e leggere le possibile cause che hanno scatenato l'errore.
Il motivo dell'errore è che in VBA stiamo utilizzando un oggetto della classe Solver senza avere introdotto un **riferimento** ad esso. Possiamo cioè dire che Excel non sa quale oggetto riferirsi dato che non conosce il riferimento alla classe cui appartiene.
Per creare il riferimento occorre attivare VBA (Alt+F11) e quindi sotto il menù "Strumenti" selezionare la voce "Riferimenti...".
È possibile notare che nella finestra che compare "Riferimenti":



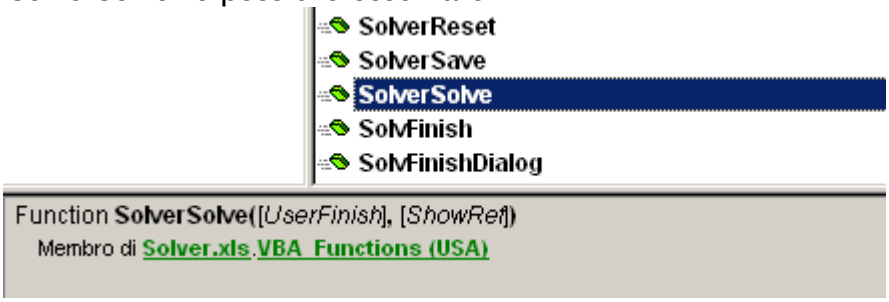
è assente il riferimento a: "Sover.xls" che appartiene al file Solver.xla. A seconda della lingua utilizzata per tale add-in tale riferimento potrebbe chiamarsi Risolutore.xls e puntare al file: Risolutore.xla.
occorre a questo punto porre un segno di spunta sul riferimento: Solver.xls e premere il bottone OK.
Provando a eseguire nuovamente la macro AzzeraEq tutto dovrebbe finalmente funzionare correttamente.

HOW TO 06: Automatizzare ulteriormente la macro relativa al Risolutore di Excel

Supponiamo di dover eseguire la macro prodotta al punto “HOW TO 05” un centinaio di volte in quanto inserita in un procedura iterativa (For… Next). Occorrerà, cosi come è stata strutturata, premere per un centinaio di volte il bottone OK che comparirà ogniqualvolta sarà invocata dal codice contenuto nel ciclo iterativo.
È possibile dire ad Excel che vogliamo che il tasto OK sia premuto automaticamente?
La risposta è probabilmente sì, anche se non sappiamo come comportarci.
Probabilmente occorrerà analizzare le opzioni messe a disposizione dalla classe Solver. Per fare ciò occorre attivare VBA (Alt+F11) e quindi cercare il codice della macro “AzzeraEq”.
Ci posizioniamo su una delle istruzioni relative al Risolutore:

```
SolverOk SetCell:="$A$2", MaxMinVal:=3, ValueOf:="0", ByChange:="$A$1"
SolverSolve
```

E premiamo il tasto F2 che attiva il “Visualizzatore oggetti”. Vedremo un elenco lunghissimo di “Classi” e “Membri”. Selezioniamo dal combo box “Tutte le librerie” le voce “Solver.xls” o comunque il riferimento attivato al punto “HOW TO 05” e quindi analizziamo ciò che viene visualizzato.
Sotto la voce “SolverSolve” è possibile osservare:



ci viene detto che tale istruzione è una funzione che può ricevere via lista due argomenti: “UserFinish” e “ShowRef”.
Occorre ora dotarsi di una certa dose di iniziativa ed intuito.
Ignoriamo da principi cosa significhi “ShowRef” e focalizziamo l’attenzione su “UserFinish”.
Dato che tali variabili sono tra parentesi quadre vuol dire che sono opzionali.
Speriamo che “UserFinish” sia una variabile logica e che quindi assuma valori “true” o “false”.
Possiamo quindi ipotizzare che oltre ad invocare la funzione SolverSolve semplicemente possa funzionare anche l’istruzione seguente:

```
call SolverSolve(true)
```

Torniamo ora al codice VBA della macro “AzzeraEq” ed introduciamo tale modifica.
Attiviamo quindi il foglio di Excel ed eseguiamo la macro “AzzeraEq”.
Tutto OK ???!!!
Abbiamo avuto un pizzico di fortuna ed anche un po’ di intuito…

A proposito anziché scrivere:	`call SolverSolve(true)`
è anche possibile scrivere:	`SolverSolve true`
o anche:	`SolverSolve UserFinish := true`