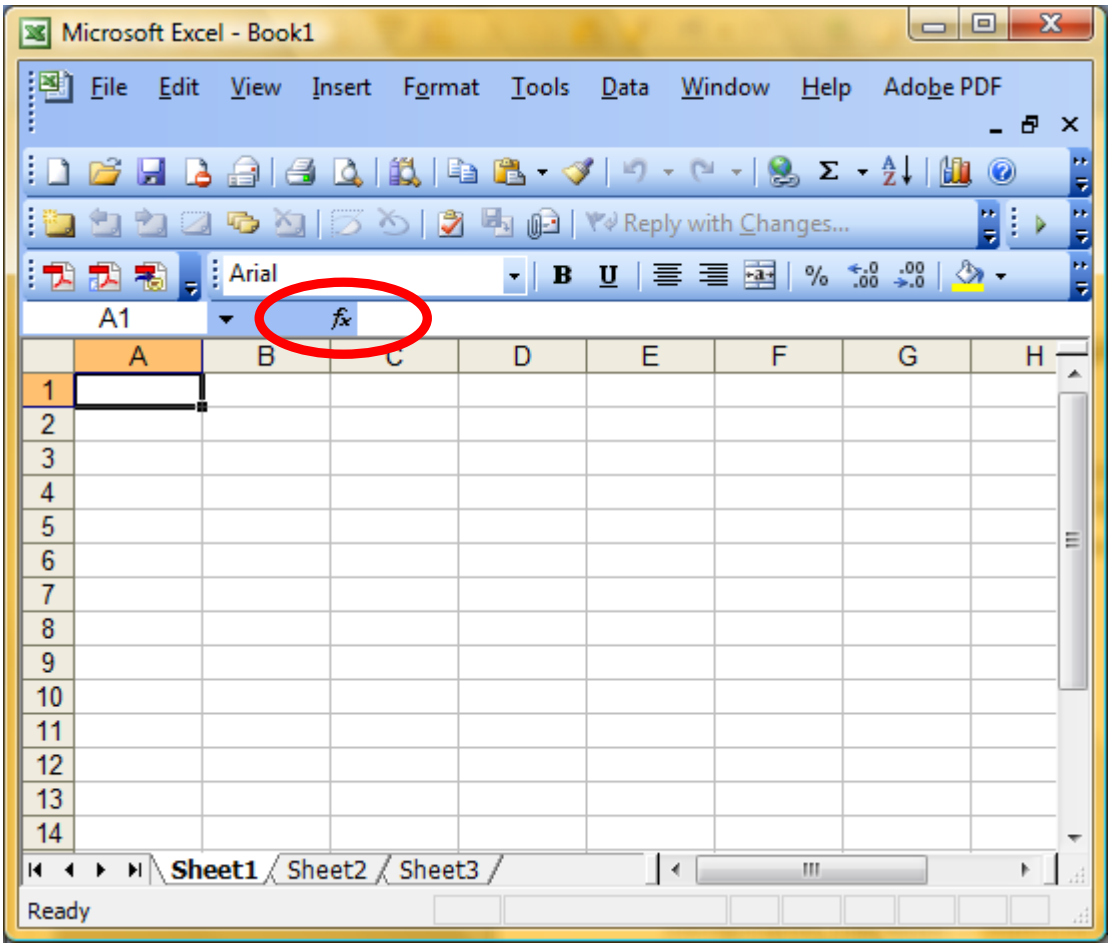
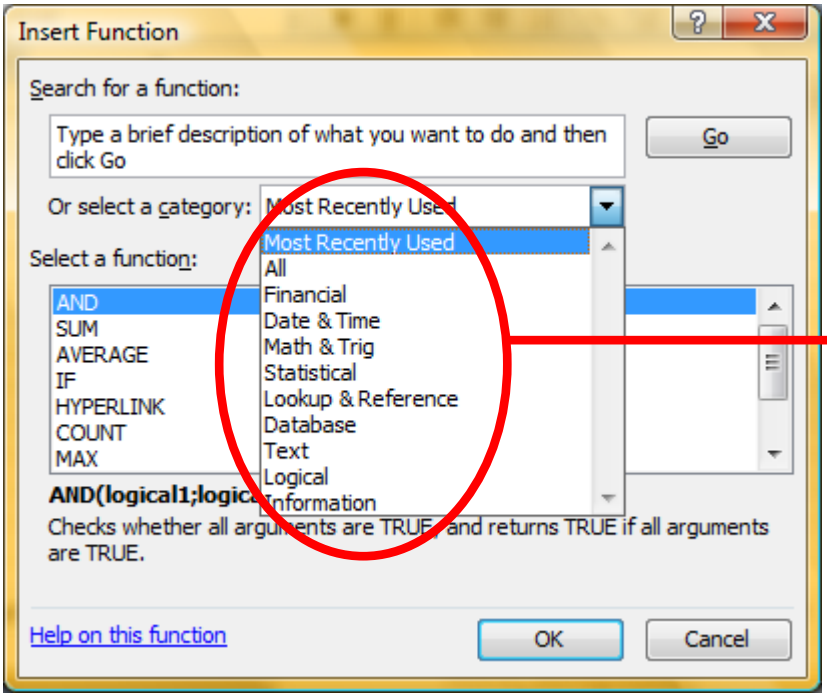
1. Εισαγωγικές Σημειώσεις για το Excel και την VBA

1.1. *Βασικές Συναρτήσεις του Excel (συνοπτικά)*

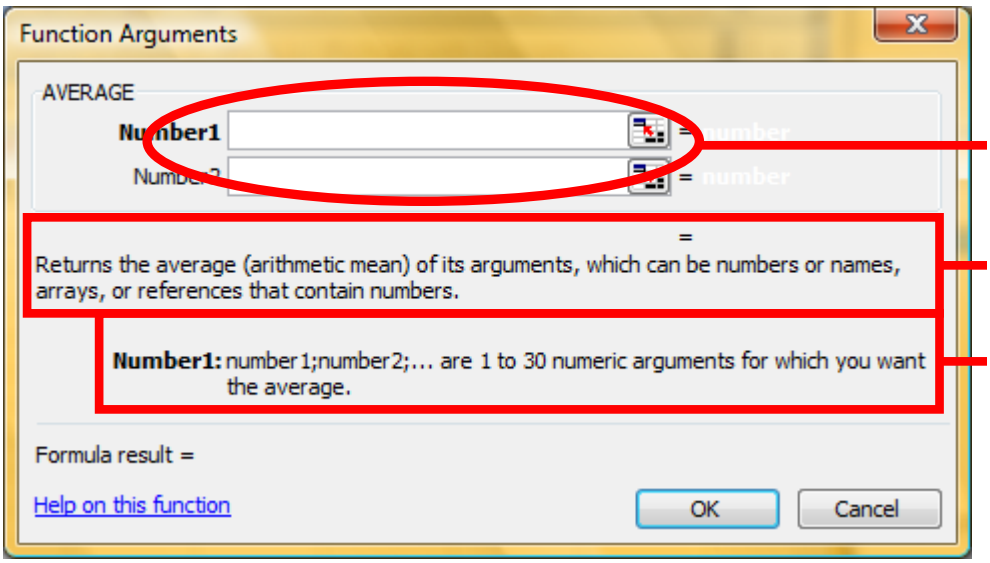
Για την εισαγωγή συναρτήσεων στο Excel μπορείτε να χρησιμοποιήσετε το κουμπί f_x που φαίνεται στην παρακάτω εικόνα :



Με την επιλογή αυτή το Excel παραθέτει μενού με όλες τις ενσωματωμένες συναρτήσεις. Για κάθε συνάρτηση παρατίθεται περιγραφή της λειτουργίας της καθώς και περιγραφή των ορισμάτων που πρέπει να εισάγουμε.



Οι κατηγορίες των συναρτήσεων. Αριστερά οι συναρτήσεις κάθε κατηγορίας.



Εισαγωγή των ορισμάτων της συνάρτησης

Περιγραφή της συνάρτησης

Περιγραφή των ορισμάτων

Απλές Συναρτήσεις

- Exp ( )
- Ln ( )
- Sqrt ( )
- Rand ( )
- Fact ( )
- Combin ( )

Συναρτήσεις Πινάκων – Matrices Functions

- Transpose (array) – δίνει τον ανάστροφο ενός πίνακα
- Mmult (array1,array2) – πολλαπλασιάζει 2 πίνακες
- Minverse (array) – δίνει τον αντίστροφο ενός πίνακα
- Determ(array) – δίνει την ορίζουσα ενός πίνακα

Για την χρήση των συναρτήσεων αυτών (αφορά τις συναρτήσεις όπου το αποτέλεσμα είναι πίνακας) θα πρέπει να πατήσετε τα πλήκτρα *Ctrl+Alt+Enter* αμέσως μετά την εισαγωγή της φόρμουλας.

Στατιστικές Συναρτήσεις :

- Average (array) – επιστρέφει τον αριθμητικό μέσο
- Stdev (array) – επιστρέφει το τυπικό σφάλμα
- Max (array) - επιστρέφει το μέγιστο από το σύνολο των ορισμάτων
- Min(array) - επιστρέφει το ελάχιστο από το σύνολο των ορισμάτων
- Quartile ( ) – βλ. περιγραφή από Excel
- Frequency ( )- βλ. περιγραφή από Excel
- Normsdist ( ) επιστρέφει την αθροιστική κατανομή για συγκεκριμένο z (για την τυπική κανονική κατανομή N(0,1))
- Normdist ( ) επιστρέφει την αθροιστική κατανομή για συγκεκριμένο z (για οποιαδήποτε κανονική κατανομή N(μ,σ))
- Normsinv( ) – Επιστρέφει το z για το οποίο η αθροιστική κατανομή έχει συγκεκριμένη τιμή (για την τυπική κανονική κατανομή N(0,1))
- Norminv ( ) Επιστρέφει το z για το οποίο η αθροιστική κατανομή έχει συγκεκριμένη τιμή (για οποιαδήποτε κανονική κατανομή N(μ,σ))

Ανάλυση Παλινδρόμησης και Συσχέτισης :

- Intercept (known y's,known x's)
- Slope (known y's,known x's)
- RSQ (known y's,known x's)
- Steyx (known y's,known x's)
- Correl (array1, array2)
- Covar (array1, array2)
- Linest (known y's,known x's) *χρειάζεται επιλογή περιοχής 5x2+Ctrl+Alt+Enter*

Συναρτήσεις Αναζήτησης (Lookup functions) :

- Vlookup (lookup_value, table_array, col_index_num, range_lookup)
- Hlookup(lookup_value, table_array, col_index_num, range_lookup)
- Match
- Index : Επιστρέφει την τιμή ενός πίνακα που βρίσκεται στη δηλωμένη γραμμή και δηλωμένη στήλη.

Άλλες Συναρτήσεις :

If and nested If

Η συνάρτηση "If "μας δίνει την δυνατότητα να έχουμε διαφορετικά αποτελέσματα για 2 συνθήκες που ορίζουμε. Στην συνάρτηση αυτή μπορούμε να εισάγουμε και άλλα "if " ώστε να έχουμε παραπάνω αποτελέσματα. Αν χρειαζόμαστε ν διαφορετικά αποτελέσματα κατά περίπτωση τότε θα πρέπει να χρησιμοποιήσουμε ν-1 δηλώσεις if.

Παράδειγμα

Ας υποθέσουμε ότι για ένα δηλωμένο ποσό Χ πρέπει να υπολογιστεί ο φόρος που πρέπει να καταβληθεί. Ο φόρος μέχρι τις 10.000€ είναι 5%, από 10.000€ – 20.000€ είναι 10% και για 20.000€ και πάνω είναι 15%. Η συνάρτηση θα είναι
=IF(X<=10000 ; X*5% ; IF(X<=20000 ; X*15% ; X*20%)) - όπου Χ η αναφορά στο κελί που περιέχει το ποσό. Έχουμε 3 δυνατά αποτελέσματα άρα θα χρησιμοποιήσουμε 2 δηλώσεις if.
Περιγραφή : Αν το ποσό είναι κάτω από 10.000 (άρα η συνθήκη TRUE) ο φόρος θα είναι Χ*5%. Διαφορετικά (η συνθήκη παίρνει την τιμή FALSE) θα είναι ίσο με το αποτέλεσμα του 2^{ου} if.

Logical Functions

- And
- Or
- Not

1.2. *Χρήση της VBA στο Excel.*

1.2.1. Εισαγωγικά

Η VBA είναι αντικειμενοστραφής γλώσσα προγραμματισμού.

1. Τα αντικείμενα ανήκουν σε συλλογές.
Για παράδειγμα η συλλογή των βιβλίων εργασίας περιέχει όλα τα ανοιχτά βιβλία (Workbooks), την συλλογή όλων των φύλλων εργασίας (Worksheets) , τη συλλογή των γραφημάτων κτλ.

2. Τα αντικείμενα ταξινομούνται με ιεραρχία.
Για παράδειγμα Workbook("Model.xls").Worksheets("inputs").Range("data")
δηλαδή το σύνολο των κελιών με την ονομασία «data» του φύλλου εργασίας «inputs» του βιβλίου εργασίας «Model.xls».

3. Τα αντικείμενα έχουν ιδιότητες
Οι ιδιότητες είναι τα χαρακτηριστικά ενός αντικειμένου, οι τιμές και οι ρυθμίσεις των οποίων περιγράφουν το αντικείμενο.
Οι τιμές των ιδιοτήτων είναι συνήθως αριθμοί, κείμενο, δηλώσεις True or False κτλ.

Για παράδειγμα
Application.ScreenUpdating = False ‘να μην κάνει refresh στην οθόνη
Range("B23").Name = "month2" ‘δίνει στο κελί B23 το όνομα month2
Range("B23").Value = 4000 ‘δίνει στο κελί B23 την τιμή 4000

4. Τα αντικείμενα έχουν μεθόδους
Ένα σύνολο δηλαδή από προκαθορισμένες λειτουργίες που μπορεί ένα αντικείμενο να χρησιμοποιήσει.

Για παράδειγμα
Range("A1:B3").Select ‘επιλέγει την συγκεκριμένη περιοχή
Range("A1:B10").Copy ‘ αντιγράφει την δηλωμένη περιοχή στο clipboard
Range("storerange").PasteSpecial ‘ κάνει επικόλληση των περιεχομένων που βρίσκονται στο clipboard στην περιοχή με το όνομα storerange.
Workbooks("Models.xls").Activate ‘ κάνει ενεργό βιβλίο εργασίας το βιβλίο «model.xls»
Sheets("inputs").Delete ‘ Διαγράφει το φύλλο εργασίας με το όνομα «inputs».

1.2.2. Γιατί να χρησιμοποιήσουμε την VBA ?

Στην πράξη προκύπτουν αρκετές περιπτώσεις όπου η απλή χρήση του Excel δεν προσφέρει το επιθυμητό αποτέλεσμα στον επιθυμητό χρόνο. Με την χρήση της ενσωματωμένης γλώσσας προγραμματισμού μπορούμε να επεκτείνουμε αρκετές από τις δυνατότητες του Excel και να πραγματοποιήσουμε διαδικασίες και υπολογισμούς πολύ πιο αποτελεσματικά και αποδοτικά.

Οι συνήθεις περιπτώσεις που κάποιος θα καταφύγει στην χρήση της vba είναι :

- Διαδικασίες που χρειάζονται αρκετά βήματα και που επαναλαμβάνονται συχνά.

Για παράδειγμα :
Εισαγωγή συγκεκριμένων τιμών από πολλά αρχεία. Η χρήση του κώδικα μπορεί να βοηθήσει στην αυτοματοποίηση του ανοίγματος των αρχείων, την αντιγραφή των τιμών στο νέο φύλλο και στο κλείσιμό τους με το πάτημα ενός κουμπιού.

- Επαναληπτικές υπολογιστικές διαδικασίες.

Για παράδειγμα η προσομοίωση Monte Carlo, όπου προσομοιώνουμε ένα τυχαίο πείραμα πάρα πολλές φορές και υπολογίζουμε το αποτέλεσμα του για κάθε φορά με σκοπό στο τέλος να βρούμε τον αριθμητικό μέσο όρων όλων των επαναλήψεων.

- Δημιουργία περίπλοκων συναρτήσεων πέραν αυτών που είναι ενσωματωμένες και που η χρήση τους είναι συχνά αναγκαία.

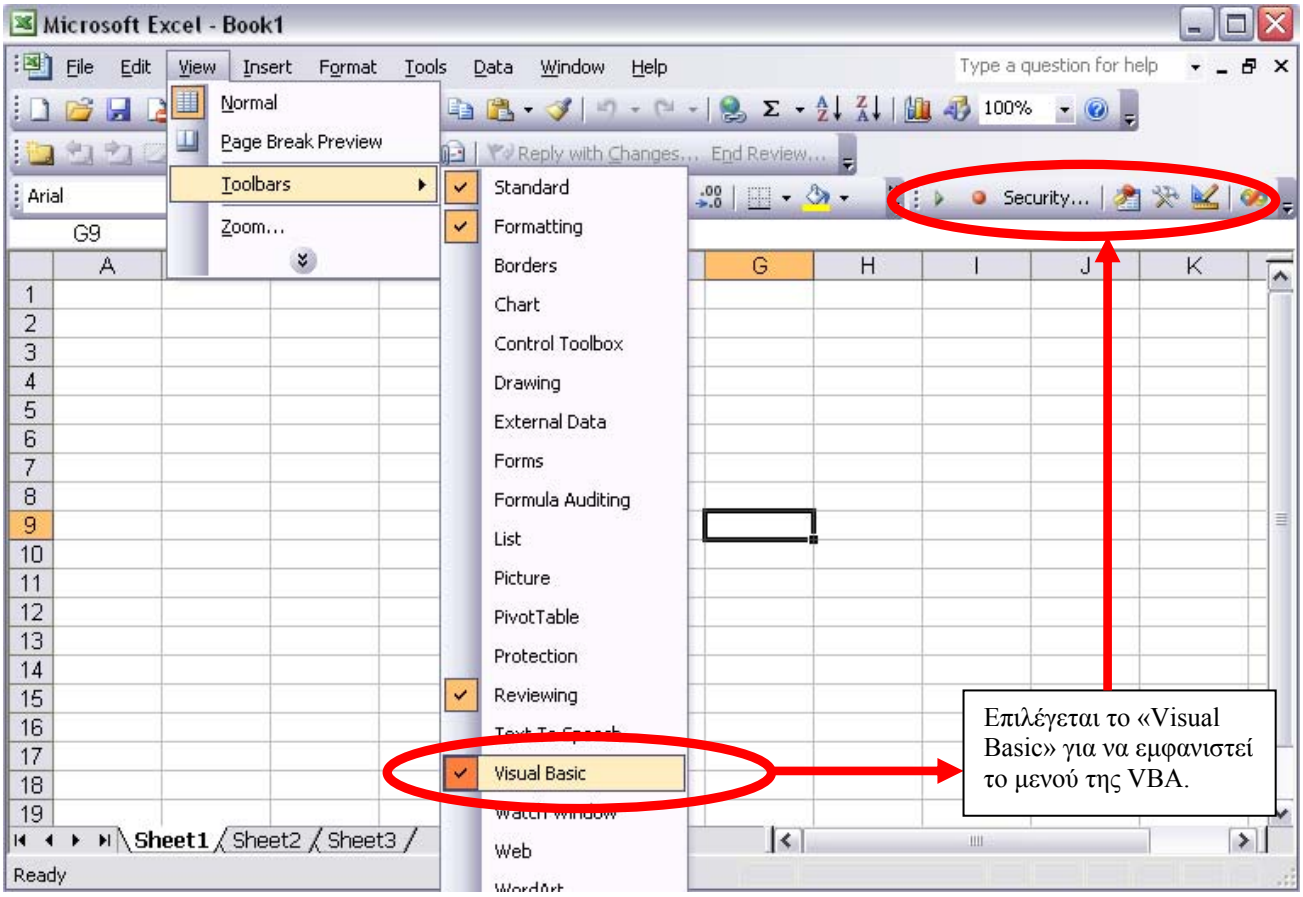
Έτσι η βασική χρήση της VBA αφορά :

- στην δημιουργία υπορουτίνων δηλαδή κομμάτια κώδικα που εκτελούν διάφορες διαδικασίες και περιέχουν αρκετές εντολές για τον σκοπό αυτό.

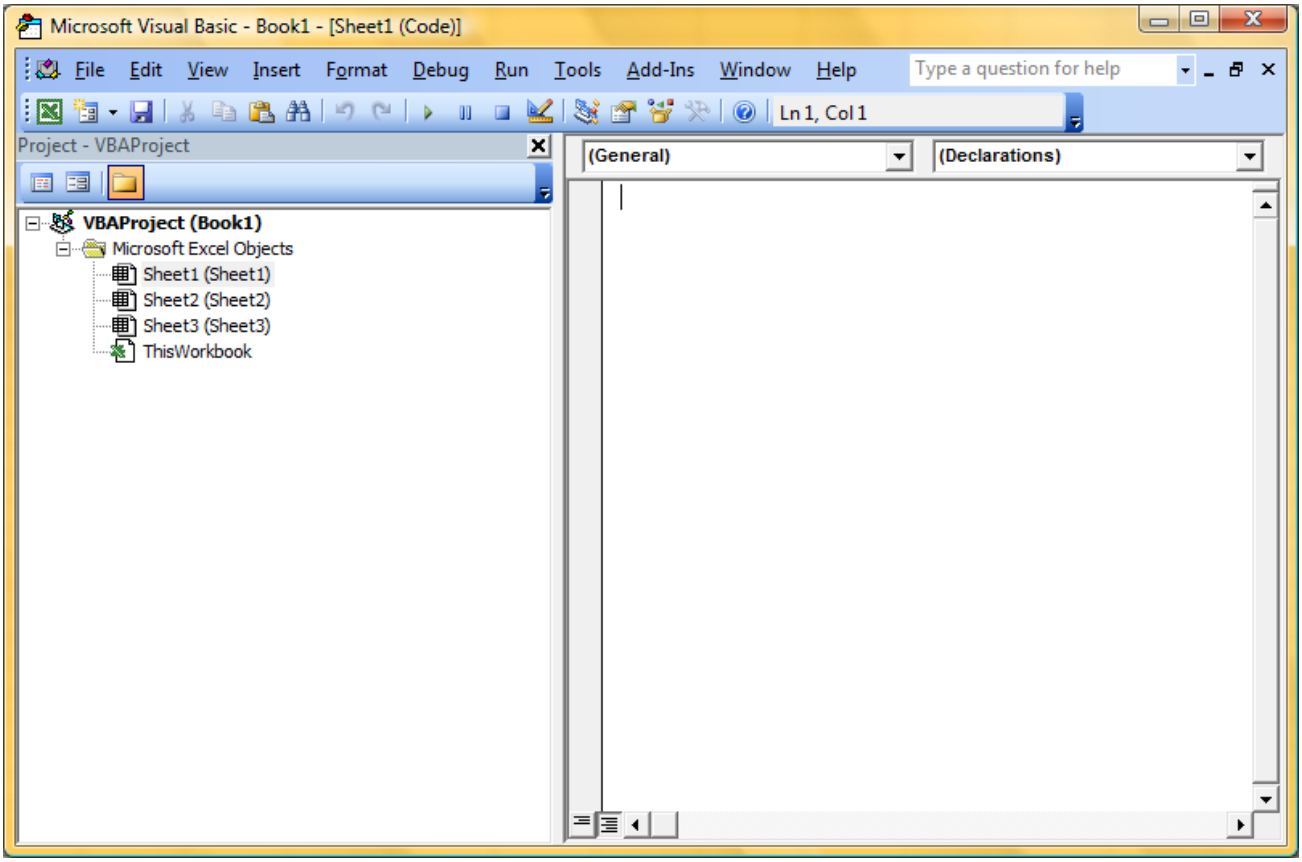
- στην δημιουργία νέων συναρτήσεων από τον χρήστη (user defined functions).
  Η διαφορά με τις υπορουτίνες είναι ότι οι συναρτήσεις δέχονται ορίσματα και επιστρέφουν τιμές.

1.2.3. VBA Editor

Για να εμφανίσετε το μενού ελέγχου για την VBA ακολουθήστε τις επιλογές που φαίνονται στην εικόνα.



Με την επιλογή του VBA editor ανοίγει το κύριο παράθυρο της vba στο οποίο θα ξεκινήσουμε να γράφουμε κώδικα.



Με την χρήση των πλήκτρων Alt+F11 μπορείτε να κάνετε εναλλαγή μεταξύ του Excel και του Editor της VBA.

1.2.4. Μεταβλητές και τύποι των δεδομένων

Οι μεταβλητές χρησιμοποιούνται για να αποθηκεύουμε και να χειριζόμαστε τα δεδομένα. Η μορφή των δεδομένων ανήκει σε διάφορους τύπους και αυτό επηρεάζει το μέγεθος της μνήμης που δεσμεύεται για την αποθήκευση τους.
Οι συνήθεις τύποι των δεδομένων είναι : integers, long, Booleans (True or False), dates, string, variant. Ο τύπος variant μπορεί να αναπαραστήσει όλα τα δεδομένα αλλά δεσμεύει περισσότερη μνήμη. Για να δηλώσει κάποιος μια μεταβλητή χρησιμοποιεί την δήλωση Dim (από την λέξη dimension – δηλαδή δίνω διάσταση στην μεταβλητή).
Με την εντολή Dim ορίζουμε και πίνακες. Για παράδειγμα αν θέλουμε να δημιουργήσουμε έναν πίνακα με όνομα vect διάστασης n x m θα γράψουμε : Dim vect (n,m).
Αν ο πίνακας είναι διάνυσμα δεν χρειάζεται δεύτερο όρισμα : Dim vect(n).

1.2.5. Sheet - Module

Όταν ανοίγουμε τον editor έχουμε την επιλογή να γράψουμε τον κώδικα κατευθείαν στον editor που αναφέρεται στο συγκεκριμένο φύλλο εργασίας (π.χ. Sheet1). Εναλλακτικά μπορούμε να εισάγουμε ένα module και να γράψουμε τον κώδικα εκεί. Το θετικό είναι ότι κάθε module μπορεί να αποθηκευτεί ξεχωριστά και να εισαχθεί σε πολλά φύλλα εργασίας. Αν για παράδειγμα έχουμε δημιουργήσει αρκετές συναρτήσεις που χρησιμοποιούμε συχνά και δεν περιέχονται στις συναρτήσεις του Excel είναι καλύτερα να τις γράψουμε σε ένα module ώστε να τις εισάγουμε κατευθείαν σε οποιοδήποτε βιβλίο εργασίας επιθυμούμε, έτσι ώστε να μπορούμε να εκτελούμε τις συναρτήσεις αυτές απευθείας.

1.2.6. Αρχικές δηλώσεις

Πριν από την δημιουργία μιας υπορουτίνας ή μιας νέας συνάρτησης μπορούμε να εισάγουμε 2 δηλώσεις που αφορούν όλον τον κώδικα που θα γράψουμε κάτω από αυτές.

Option Explicit : Μας υποχρεώνει να δηλώνουμε τον τύπο κάθε μεταβλητής που θα χρησιμοποιήσουμε στον κώδικα. Σε περίπτωση που αυτό δεν δηλωθεί δεν είναι απαραίτητο να καθορίζουμε τον τύπο κάθε μεταβλητής. Το πρόγραμμα θα χρησιμοποιεί ως τύπο έναν γενικό τύπο. Το μειονέκτημα στην περίπτωση αυτή είναι ότι οι μεταβλητές αυτές θα καταλαμβάνουν περισσότερο χώρο στη μνήμη (ο τύπος variant που αναφέρθηκε παραπάνω). Πρακτικά για τις μαθηματικές εφαρμογές που μας αφορούν μπορούμε να το αγνοήσουμε χωρίς προβλήματα, αλλά μπορεί να είναι χρήσιμο για τον έλεγχο και σωστή χρήση των μεταβλητών που έχουμε εισάγει καθώς σε λάθος καταχώρηση το πρόγραμμα θα μας ειδοποιήσει για το σφάλμα.

Option Base 1 : Σε περίπτωση που δεν εισάγουμε την έκφραση αυτή η «default» έκφραση που ισχύει είναι το Option Base 0. Η δήλωση αυτή αφορά την αρίθμηση των στοιχείων διανυσμάτων και πινάκων. Στην περίπτωση του Option Base 0 το πρώτο στοιχείο ενός διανύσματος θεωρείται ότι βρίσκεται στην θέση 0, το $2^{ο}$ στην θέση 1 κ.ο.κ. Στην περίπτωση του Option Base 1 το $1^{ο}$ στοιχείο έχει αρίθμηση 1, το $2^{ο}$ αρίθμηση 2 κ.ο.κ.
Η χρήση των 2 παραπάνω δηλώσεων εισάγεται πριν από τον κώδικα (πριν το Sub . . .) και ισχύει για όλες τις υπορουτίνες που θα δημιουργηθούν στο συγκεκριμένο module ή sheet.

1.2.7. Υπορουτίνες

Για να γράψουμε μία υπορουτίνα ξεκινάμε γράφοντας :

Sub newtask
όπου newtask είναι το όνομα της υπορουτίνας με το οποίο την καλούμε.
Αυτόματα το πρόγραμμα δημιουργεί την γραμμή End Sub.
Ο κώδικας περικλείεται ανάμεσα σε αυτές τις 2 γραμμές.

1.2.8. Εισαγωγή μεταβλητών

Η τιμή μίας μεταβλητής μπορεί να εισαχθεί :

- Απευθείας στον κώδικα (δεν συνίσταται)
- Από ένα κελί ενός φύλλου εργασίας του Excel
- Από Βάση Δεδομένων
- Από αρχείο
- Από Input Box (θα περιγραφεί παρακάτω)

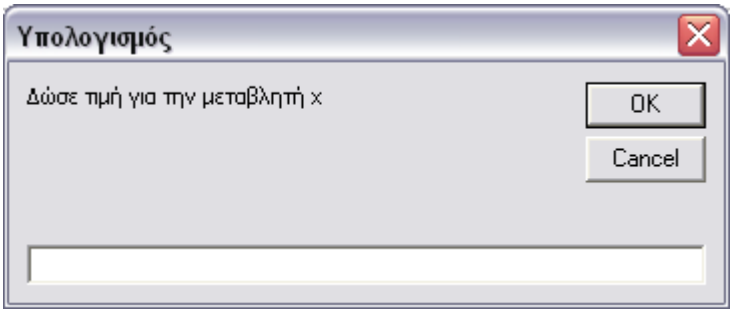
Θα ασχοληθούμε με την περίπτωση που οι τιμές των μεταβλητών έχουν εισαχθεί σε φύλλο εργασίας του Excel ή μέσω Input Box.

Για να αναφερθούμε σε ένα κελί μέσα στον κώδικα θα πρέπει να καθορίσουμε ακριβώς την ιεραρχία. Για παράδειγμα αν έχουμε δημιουργήσει την μεταβλητή x η οποία θα παίρνει τιμή από το κελί A1 που βρίσκεται στο φύλλο εργασίας «Sheet1» του βιβλίου εργασίας «Book1» τότε θα πρέπει να εισάγουμε μέσα στον κώδικα την έκφραση :
x = Workbooks(“Book1”).Worksheets(“Sheet1”).Range(“A1”).Value
Την ιδιότητα value μπορούμε να την παραλείψουμε.

Με την διαδικασία αυτή η μεταβλητή x έχει πάρει εσωτερικά στο πρόγραμμα μας την τιμή που έχει δοθεί στο κελί A1.
Σημείωση : *Αν το συγκεκριμένο βιβλίο εργασίας και το συγκεκριμένο φύλλο εργασίας είναι ενεργά δηλαδή βρισκόμαστε σε αυτά την ώρα που τρέξουμε τον κώδικα τότε η αναφορά σε αυτά θα μπορούσε να παραληφθεί δηλαδή θα μπορούσαμε να γράψουμε :*
x = Range(“A1”).Value ή x = Range(“A1”)

Προτείνεται σε περίπτωση που η αναφορά γίνεται σε συγκεκριμένο βιβλίο και φύλλο εργασίας να γράφεται ολόκληρη η διαδρομή για να αποφευχθούν λάθη.

Εναλλακτικά για να δώσουμε τιμή σε μία μεταβλητή θα μπορούσαμε να χρησιμοποιήσουμε έναν πίνακα εισόδου τιμής. Την συγκεκριμένη λειτουργία την εκτελεί η εντολή InputBox. Η σύνταξη είναι η ακόλουθη :
x = InputBox (“text1”, “text2”) όπου το 1^{ο} όρισμα είναι το κείμενο που θα εμφανιστεί ενώ το 2^{ο} είναι ο τίτλος του παραθύρου που εμφανίζεται.
Στο παράδειγμα θα μπορούσε να είναι : x = InputBox (“Δώσε τιμή για την μεταβλητή x” , “Υπολογισμός”) και το αποτέλεσμα είναι η εμφάνιση του παραθύρου που φαίνεται παρακάτω.



Η γλώσσα vba χρησιμοποιεί τους γνωστούς τελεστές (+,-,*,/,^) και έχει ενσωματωμένες κάποιες συναρτήσεις που μπορούμε να χρησιμοποιήσουμε χωρίς να χρειαστεί να καταφύγουμε στις συναρτήσεις του Excel.
Σε περιπτώσεις που θέλουμε να χρησιμοποιήσουμε μία συνάρτηση του Excel θα πρέπει να καλέσουμε την εφαρμογή του Excel. Η σύνταξη είναι : Application.Excel_Function (array1,array2), όπου Excel_Function είναι το όνομα της συνάρτησης του Excel που θέλουμε να χρησιμοποιήσουμε.
Στην VBA τα ορίσματα μιας συνάρτησης χωρίζονται μεταξύ τους με « , » και όχι με « ; ».

1.2.9. Διαχείριση Διανυσμάτων και Πινάκων στην VBA

- Όταν δεν υπάρχουν ενσωματωμένες συναρτήσεις μπορεί να χρειαστεί να επεξεργαστούμε τα στοιχεία ενός διανύσματος ένα ένα, οπότε θα χρειαστεί να εκτελέσουμε μια επαναληπτική διαδικασία (loop). Αρκετές φορές θα χρειαστεί να μετρήσουμε τα δεδομένα ώστε να καθορίσουμε τον αριθμό των επαναλήψεων που θα πρέπει να εκτελέσουμε. Για να το κάνουμε αυτό μπορούμε να χρησιμοποιήσουμε την ενσωματωμένη συνάρτηση του Excel "Count" (Application.Count(array)).
- Θα πρέπει πάντα να προσέχουμε την δήλωση Option Base ώστε να γίνεται σωστά η αρίθμηση και η αναφορά στα στοιχεία του πίνακα.
- Όταν εκτελούμε πράξεις πινάκων θα πρέπει να ελέγχουμε τις διαστάσεις των πινάκων. Θα πρέπει να συμπεριλαμβάνονται διαδικασίες ελέγχου ώστε να εξασφαλίζεται ότι τα δεδομένα είναι στη σωστή μορφή (διανύσματα γραμμής, διανύσματα στήλης).

1.2.10. Επαναληπτικές διαδικασίες

- For

Σύνταξη :

```
For i=1 to 10
…..
….
Next i
```

- Do while (επανέλαβε όσο ισχύει η συνθήκη)

```
Do while k>10
…….
…….
Loop
```

- Do Until (επανέλαβε μέχρι να ισχύσει η συνθήκη τερματισμού)

```
Do until k=10
……
…….
Loop
```

1.2.11. Διαδικασίες ελέγχου.

Πολύ χρήσιμη είναι η έκφραση Msgbox(“text & variable….”)
Η έκφραση αυτή εμφανίζει την τιμή της μεταβλητής που ζητάμε, ή κάποιο μήνυμα που εισάγουμε. Με την χρήση της μπορούμε να πληροφορηθούμε για ενδιάμεσα βήματα του κώδικα κατά την διαδικασία εκτέλεσης του.

1.2.12. Παραδείγματα

Για όλα τα παραπάνω θα χρησιμοποιήσουμε ένα απλό παράδειγμα που όμως καλύπτει αρκετά από τα προηγούμενα. Το παράδειγμα θα είναι η δημιουργία κώδικα που θα υπολογίζει το παραγοντικό ενός αριθμού που θα εισάγεται από τον χρήστη.

Έστω ότι η τιμή για την οποία θέλουμε να υπολογίσουμε το παραγοντικό εισάγεται από τον χρήστη στο κελί A1 του βιβλίου εργασίας book1, στο φύλλο sheet1.

```
Sub factorial1
Dim a as integer
a = Workbook(“book1”).Worksheets(“Sheet1”).Range(“A1”)  ‘το κελί από όπου θα πάρει τιμή η μεταβλητή a.
factorial=1
for i=1 to a
factorial = factorial*i
next i
Workbook(“book1”).Worksheets(“Sheet1”).Range(“B1”) = factorial
End Sub
```

Εναλλακτικά η τιμή θα μπορούσε να δοθεί σε ένα πίνακα εισόδου τιμών (Input Box).

```
Sub factorial2
Dim a as integer
a = InputBox(…..)
factorial=1
for i=1 to a
factorial = factorial*i
next i
Workbook(“book1”).Worksheets(“Sheet1”).Range(“B1”) = factorial
End Sub
```

Στο παρακάτω παράδειγμα δεν χρησιμοποιούνται τα κελιά του Excel αλλά οι δηλώσεις InputBox και MsgBox

```
Sub factorial3
Dim a as integer
a = InputBox(…..)
factorial=1
for i=1 to a
```

```
factorial = factorial*i
next i
MsgBox (……& factorial)
End Sub
```

Θα μπορούσαμε να χρησιμοποιήσουμε την συνάρτηση fact του Excel που υπολογίζει παραγοντικά ως εξής :

```
Sub factorial1
Dim a as integer
a = InputBox(…..)
factorial=Application.Fact(a)
MsgBox (……& factorial)
End Sub
```

Εναλλακτικά με χρήση του do while.

```
Sub factorial1()
Dim a As Integer
a = Range("A1")
factorial = 1
i = 1
Do While i <= a
factorial = factorial * i
i = i + 1
Loop
Range("B1") = factorial
End Sub
```

```
Sub factorial1
Dim a as integer
a = InputBox(…..)
factorial=1
i=1
Do While i <= a
factorial = factorial * i
i = i + 1
Loop
MsgBox (……& factorial)
End Sub
```

1.2.13. Concatenation

Για να ενώσουμε μεταβλητές ή κείμενο μπορούμε να χρησιμοποιήσουμε το σύμβολο «&» το οποίο μας επιτρέπει να συνενώσουμε τα μέρη που θέλουμε. Για παράδειγμα αν θέλουμε να δημιουργήσουμε το όνομα ενός αρχείου που το πρώτο μέρος του είναι η λέξη data και το δεύτερο μέρος του είναι η ημερομηνία π.χ. data_15/9/2008 θα μπορούσαμε να γράψουμε :

```
Sub filename
Dim name as string
Dim todate as date
todate = Range("A1")   'η ημερομηνία περιέχεται στο κελί A1
Name = "data_"& todate
End sub
```

1.2.14. Άνοιγμα Αρχείων

Ας υποθέσουμε ότι πρέπει να αντλήσουμε κάποιες τιμές από μια σειρά αρχείων που είναι ταξινομημένα βάσει ημερομηνίας. Έστω το όνομα του πρώτου αρχείου είναι data_1/1/2008.xls και βρίσκεται αποθηκευμένο στον φάκελο C:\Inputs .
Για το άνοιγμα του αρχείου θα χρησιμοποιηθεί η εντολή Workbook.Open("C:\Inputs\data_1/1/2008.xls"). Με την εντολή αυτή, ανοίγει το συγκεκριμένο βιβλίο εργασίας και είναι πια το ενεργό βιβλίο (active workbook). Έστω ότι χρειαζόμαστε να κρατήσουμε την τιμή που βρίσκεται στο κελί A2 του φύλλου εργασίας "data" και να το αποθηκεύσουμε σε ένα πίνακα εσωτερικά στο πρόγραμμα μας.

Για να το πετύχουμε αυτό :
1ον θα αποφασίσουμε σχετικά με την αρίθμηση των στοιχείων του πίνακα (Option Base statement)
2ον θα πρέπει να ορίσουμε τον πίνακα στον οποίο θα κρατάμε τα στοιχεία.
Έστω ότι τα δεδομένα μας είναι ημερήσια και θέλουμε να υπολογίσουμε τον μηνιαίο μέσο όρο αυτών. Άρα η διάσταση του πίνακα όπου θα αποθηκεύσουμε τα ημερήσια δεδομένα θα είναι 30 (Option Base 0 – Άρα το διάνυσμα έχει 31 στοιχεία). Για να ορίσουμε ένα πίνακα με αυτές τις διαστάσεις θα γράψουμε :
Dim data_matrix(30) όπου data_matrix είναι το όνομα που δίνουμε στην μεταβλητή μας (θα είναι τύπου variant).
Θα πρέπει να ανοίξουμε το βιβλίο εργασίας όπου βρίσκονται τα δεδομένα. Αυτό γίνεται σύμφωνα με τα παραπάνω Workbook.Open("C:\Inputs\data_1/1/2008.xls").

3ον θα πρέπει να αποθηκεύσουμε την τιμή που βρίσκεται στο κελί B2 του φύλλου εργασίας "data" στον πίνακα που δημιουργήσαμε. Δηλαδή :
Data_matrix(0) = Worksheets("data").Range("B2")
Με την παραπάνω δήλωση έχουμε αποθηκεύσει την τιμή του κελιού ως πρώτο στοιχείο του πίνακα (για Option Base 0).

4ον πρέπει να κλείσουμε το αρχείο. Η εντολή για αυτό είναι :
Workbook("data_1/1/2008.xls").Close
SaveChanges = False ' αυτό χρειάζεται για να μην αποθηκεύονται αλλαγές που ενδέχεται να γίνονται.

Έστω τώρα ότι θέλουμε η διαδικασία αυτή να γίνει αυτόματα για όλες τις μέρες του μήνα έτσι ώστε στο τέλος να υπολογίσουμε τον αριθμητικό μέσο των τιμών αυτών. Έτσι χρειάζεται να φτιάξουμε μια διαδικασία που θα ανοίγει ένα ένα τα βιβλία εργασίας για κάθε μέρα και θα αποθηκεύει την τιμή που θέλουμε στον πίνακα data_matrix.

Δημιουργούμε την μεταβλητή date1 ως : Dim date1 και ορίζουμε την μεταβλητή name δηλ. Dim name as string.

Η επαναληπτική διαδικασία θα είναι :
For i = 0 to 30
Date1 = 1/1/2008 + i

Day1=day(date1)
Month1= month(date1)
Year1 = year(date1)
Date2 = day1 & ”_” & month1 & “_” & year1
Set srcbook = "C:\Inputs\data_" & date2 & ".xls"
Workbook.Open(srcbook)
Data_matrix(i) = Range(“B2”)
Workbook(srcbook).Close
SaveChanges = False

Next i

1.2.15. Συναρτήσεις (User defined functions) – nested functions

Ένας από τους βασικούς λόγους χρήσης της vba είναι η δημιουργία συναρτήσεων από τον χρήστη. Υπάρχουν πολλές περιπτώσεις όπου δεν υπάρχουν ενσωματωμένες συναρτήσεις του Excel που να μας εξυπηρετούνε ή το αποτέλεσμα να είναι αποτέλεσμα που προκύπτει από την χρήση πολλών άλλων συναρτήσεων. Για τους λόγους αυτούς μπορούμε να δημιουργήσουμε δικές μας συναρτήσεις ώστε να κάνουμε τους υπολογισμούς γρηγορότερα.

Για την δημιουργία συναρτήσεων πηγαίνουμε στον vba editor. Για να ξεκινήσουμε την κατασκευή μιας τέτοιας συνάρτησης ξεκινάμε με την εντολή :

Function f_name(όρισμα1,όρισμα2, . . ., όρισμα n)
……
……
End function

Ο κώδικας περικλείεται και πάλι ανάμεσα στις 2 αυτές γραμμές και οι κανόνες ισχύουν ακριβώς όπως και στην περίπτωση δημιουργίας υπορουτίνων. Τα ορίσματα εισάγονται από τον χρήστη την στιγμή που καλεί την συνάρτηση όπως και με τις ενσωματωμένες συναρτήσεις του Excel. Τα ορίσματα δεν χρειάζεται να δηλωθούν σαν μεταβλητές.

Παράδειγμα
Ας δημιουργήσουμε για παράδειγμα τον κώδικα για την αποτίμηση ενός Ευρωπαϊκού δικαιώματος με την χρήση του κλασικού τύπου Black-Scholes.

Οι τύποι που θα πρέπει να χρησιμοποιήσουμε είναι οι παρακάτω :

$$BSCall(S,r,X,T-t,\sigma,q) = Se^{-q(T-t)}\cdot N(d_1) - Xe^{-r(T-t)}\cdot N(d_2)$$
$$BSPut(S,r,X,T-t,\sigma,q) = Xe^{-r(T-t)}\cdot N(-d_2) - Se^{-q(T-t)}\cdot N(-d_1)$$

όπου $d_1 = \frac{\ln(S/X) + (r + \sigma^2/2)\cdot(T-t)}{\sigma\sqrt{T-t}}$ και $d_2 = d1 - \sigma\sqrt{T-t}$

Τα ορίσματα της συνάρτησης είναι : $S, r, X, T-t, \sigma, q$
όπου :
X : Το strike price
T-t : Ο χρόνος μέχρι την λήξη
q : Η μερισματική απόδοση
σ : Η ετήσια μεταβλητότητα
S : Η τρέχουσα τιμή της υποκείμενης
r : το risk-free επιτόκιο

Θα μπορούσαν και οι δύο παραπάνω περιπτώσεις (Put και Call) να περιγραφούν από μία συνάρτηση η οποία θα παίρνει ως όρισμα την μεταβλητή i, όπου η τιμή της θα εξαρτάται με το αν θέλουμε να αποτιμήσουμε ένα Put ή ένα Call Option.
Έτσι στην γενική περίπτωση ο μαθηματικός τύπος θα είναι :
$BS(i, S, r, X, T-t, \sigma, q) = (i)\cdot Se^{-q(T-t)}\cdot N(i\cdot d_1) - (i)Xe^{-r(T-t)}\cdot N(i\cdot d_2)$
όπου :
i = 1 στην περίπτωση του Call option και
i = -1 στην περίπτωση του Put Option.

Για να κάνει κάποιος αυτούς τους υπολογισμούς στο Excel θα χρειαστεί να κάνει αρκετές πράξεις και να δαπανήσει αρκετό χρόνο και η πιθανότητα λάθους με την χρήση των συμβόλων είναι αρκετά μεγάλη. Για το λόγο αυτό είναι προτιμότερο να δημιουργήσουμε μία συνάρτηση που αποτιμά τα δικαιώματα αυτά και την οποία μπορούμε να καλούμε ανά πάσα στιγμή.

Έτσι η συνάρτηση την οποία θα ονομάσουμε BSOptValue και η οποία αποτιμά ένα Ευρωπαϊκό δικαίωμα αγοράς (iopt=1) ή πώλησης (iopt=-1) θα είναι :

Function BSOptValue(iopt, S, X, r, q, tyr, sigma)
```
'  Returns Black-Scholes Value (iopt=1 for call, -1 for put; q=div yld)
'  Uses BSDOne fn
'  Uses BSDTwo fn
   Dim ert, eqt, NDOne, NDTwo
   ert = Exp(-r * tyr)
   eqt = Exp(-q * tyr)

   NDOne = Application.NormSDist(iopt * BSDOne(S, X, r, q, tyr, sigma))
   NDTwo = Application.NormSDist(iopt * BSDTwo(S, X, r, q, tyr, sigma))
   BSOptValue = iopt * (S * eqt * NDOne - X * ert * NDTwo)

End Function
```

Function BSDOne(S, X, r, q, tyr, sigma)
```
'  Returns Black-Scholes d1 value
   BSDOne = (Log(S / X) + (r - q + 0.5 * sigma ^ 2) * tyr) / (sigma * Sqr(tyr))
End Function
```

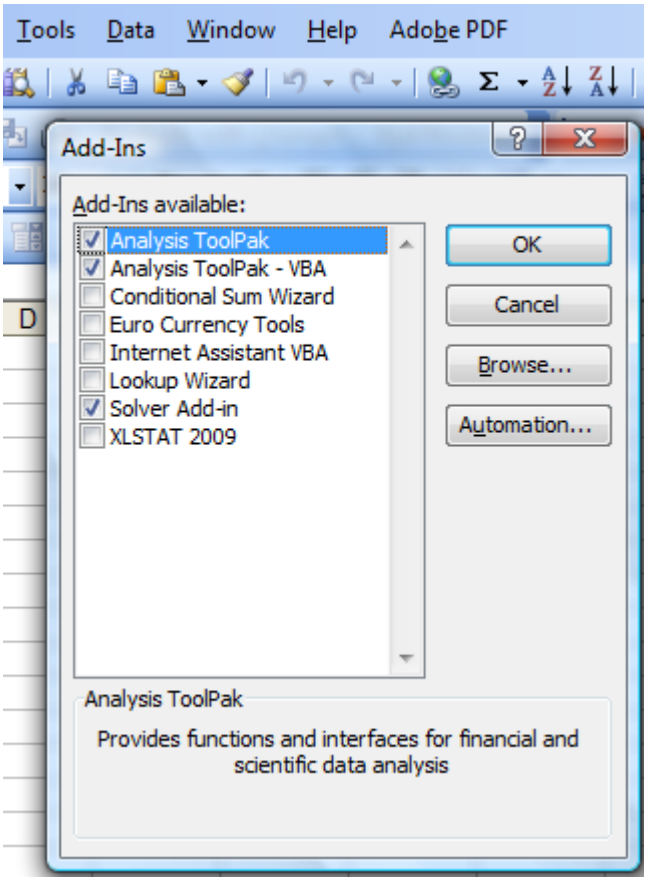
```
Function BSDTwo(S, X, r, q, tyr, sigma)
'  Returns Black-Scholes d2 value
   BSDTwo = (Log(S / X) + (r - q - 0.5 * sigma ^ 2) * tyr) / (sigma * Sqr(tyr))
End Function
```

Για να χρησιμοποιήσει κάποιος την συνάρτηση αυτή θα πρέπει να πάει στις συναρτήσεις (f_x), να επιλέξει user-defined functions και να επιλέξει την συνάρτηση. Εναλλακτικά μπορεί να γράψει κατευθείαν το όνομα της συνάρτησης και να εισάγει τα ορίσματα.

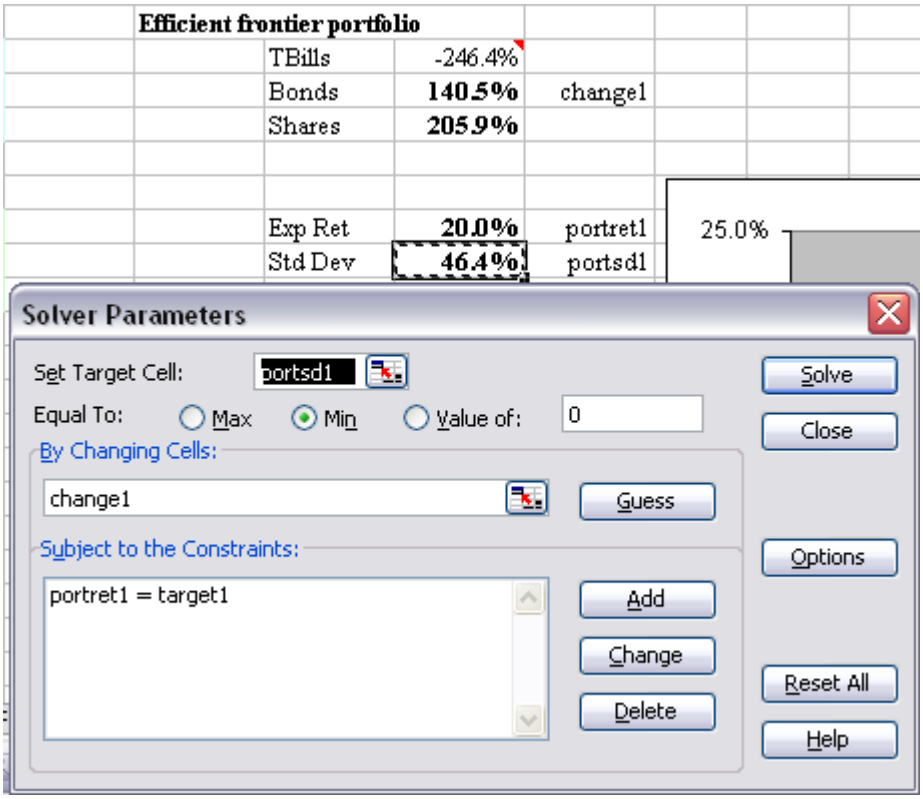
Στην περίπτωση που δημιουργούμε μία συνάρτηση μπορούμε μέσα στον κώδικα να καλέσουμε άλλες συναρτήσεις που έχουμε δημιουργήσει (nested functions). Αυτό που θα πρέπει να προσέχουμε είναι η συνάρτηση που καλούμε μέσα στον κώδικα, να έχει καθορισμένα τα ορίσματα της είτε μέσω των ορισμάτων της αρχικής συνάρτησης, είτε μέσω των μεταβλητών που ορίζονται στον κώδικα.

1.2.16. Solver στην VBA

Στην περίπτωση όπου κάποιος θέλει να εκτελέσει μία βελτιστοποίηση μίας συνάρτησης υπό κάποιους περιορισμούς μπορεί να χρησιμοποιήσει την ενσωματωμένη λειτουργία του "SOLVER". Η εφαρμογή αυτή βρίσκεται στην επιλογή Tools.
Στην περίπτωση όπου δεν εμφανίζεται θα πρέπει να την ενεργοποιήσετε από τα Tools → Add-Ins → Solver Add-in όπως φαίνεται στην παρακάτω εικόνα.



Με την επιλογή της εφαρμογής του Solver ανοίγει το παρακάτω μενού για να γίνει η εισαγωγή των παραμέτρων.

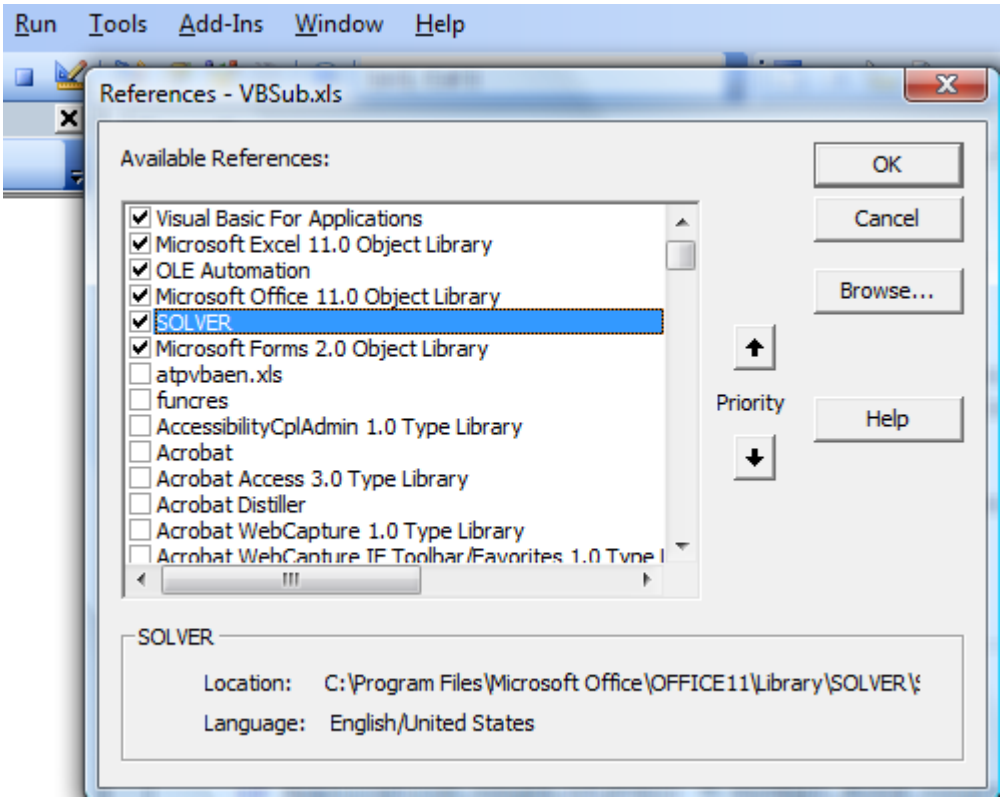


Τα πεδία είναι τα εξής :
"Set Target Cell" : Δίνουμε το κελί που περιλαμβάνει την συνάρτηση που θέλουμε να βελτιστοποιήσουμε.
"Equal To" : Δηλώνουμε αν θέλουμε μεγιστοποίηση, ελαχιστοποίηση ή μία συγκεκριμένη τιμή.
"By Changing Cells" : Εδώ δηλώνουμε τις μεταβλητές τις οποίες θέλουμε να υπολογίσουμε (λύση της βελτιστοποίησης).
"Subject to the Constraints" – Add, Change, Delete : Εδώ μπορούμε να εισάγουμε, να τροποποιήσουμε και να διαγράψουμε τους περιορισμούς του μοντέλου.

Όταν συμπληρωθούν τα πεδία η επίλυση γίνεται με το κουμπί "Solve".

Στην περίπτωση όμως που κάποιος θέλει να τρέξει μία σειρά από βελτιστοποιήσεις η παραπάνω διαδικασία μπορεί να γίνει αρκετά επίπονη έως αδύνατη, αν πρόκειται για πολύ μεγάλο αριθμό επαναλήψεων. Σε μία τέτοια περίπτωση η χρήση της VBA είναι αναγκαία και θα διευκολύνει πάρα πολύ τους απαιτούμενους υπολογισμούς.

Σημειώνεται ότι για να είναι δυνατή η χρήση του Solver στην VBA θα πρέπει να ενεργοποιηθεί το αντίστοιχο reference. Για να το κάνετε αυτό επιλέγετε στον Editor της VBA Tools → References → και από το μενού που ανοίγει επιλέγεται το SOLVER όπως φαίνεται παρακάτω.



Οι εντολές της VBA που σχετίζονται με τον Solver είναι οι παρακάτω :

SolverReset : Με την έκφραση αυτή μηδενίζονται όλα τα πεδία μέσα στον Solver.

Call SolverAdd(Range1, (1,2,3) , Range2) : Με την έκφραση αυτή εισάγουμε τους περιορισμούς. Στο Range1 γίνεται αναφορά στο κελί που περιέχει το αριστερό μέλος του περιορισμού ενώ στο Range2 εισάγεται το κελί που περιέχει το δεξί μέλος. Το ενδιάμεσο όρισμα (1,2,3) αναφέρεται στον τελεστή της σχέσης. Συγκεκριμένα : «1» είναι το «≤» , «2» είναι το «=» και το «3» είναι το «≥».

Call SolverOk(Range1, (1,2,3), 0, Range2) : Με την έκφραση αυτή δηλώνουμε την αντικειμενική συνάρτηση, τις μεταβλητές και το αν θέλουμε μέγιστο, ελάχιστο ή κάποια συγκεκριμένη τιμή. Συγκεκριμένα στο Range1 εισάγεται το κελί που περιλαμβάνει την αντικειμενική συνάρτηση. Στο Range2 εισάγονται τα κελιά που περιέχουν τις μεταβλητές (changing cells). Το $2^{ο}$ όρισμα (1,2,3) καθορίζει το εάν πρόκειται για μεγιστοποίηση («1»), ελαχιστοποίηση («2») ή για συγκεκριμένη τιμή («3»). Το $3^{ο}$ όρισμα (0) αποτελεί την τιμή που θέλουμε να έχει η συνάρτηση σε περίπτωση που έχουμε δηλώσει στο προηγούμενο όρισμα την τιμή «3». Το πεδίο αυτό είναι απαραίτητο να δηλώνεται παρότι στην περίπτωση μεγιστοποίησης και ελαχιστοποίησης δεν είναι αναγκαίο.

Call SolverChange(Range1, 2, Range2) : Με την έκφραση αυτή μπορούμε να κάνουμε αλλαγές σε έναν υπάρχον περιορισμό. Τα πεδία είναι τα ίδια με αυτά της διαδικασίας **SolverAdd.**

Call SolverSolve(True) : Με την έκφραση αυτή εκτελείται η βελτιστοποίηση.

SolverFinish : Με την έκφραση αυτή απενεργοποιείται ο Solver.

Παράδειγμα
Θέλουμε να δημιουργήσουμε μία διαδικασία (υπορουτίνα) η οποία θα υπολογίζει το Efficient Frontier για κάποια περιουσιακά στοιχεία. Η βελτιστοποίηση θα μας δίνει για κάθε ζητούμενη απόδοση, τα κατάλληλα βάρη έτσι ώστε να επιτυγχάνουμε την ελάχιστη διακύμανση. Το μοντέλο (Markowitz) είναι το εξής :

$$\min \sum_{i,j=1}^{N} w_i w_j \sigma_{ij}$$

s.t

$$\sum_{i=1}^{N} w_i = 1$$

$$\sum_{i=1}^{N} w_i r_i = \bar{r}$$

Για να δημιουργήσουμε όμως το Efficient Frontier θα πρέπει να εκτελέσουμε την παραπάνω βελτιστοποίηση για ένα εύρος αποδόσεων. Έτσι χρειαζόμαστε μία επαναληπτική διαδικασία στην οποία θα ξεκινάμε από μία αρχική ζητούμενη απόδοση (min_target) και η οποία σε κάθε επανάληψη - από το προκαθορισμένο σύνολο επαναλήψεων (niter1) - θα αυξάνει κατά ένα προκαθορισμένο ποσοστό (incr1). Με την αυτοματοποίηση της διαδικασίας αυτής θα μπορεί σε κάθε βήμα να εκτελείται η ζητούμενη βελτιστοποίηση για αυξανόμενες ζητούμενες αποδόσεις. Το αποτέλεσμα της κάθε επανάληψης είναι η απόδοση, η διακύμανση και τα βάρη για κάθε περιουσιακό στοιχείο.
Για την παρακάτω ρουτίνα έχουν δοθεί ονόματα στα κελιά που περιέχουν τις παραμέτρους που θα χρησιμοποιηθούν και επίσης έχουν γραφεί οι απαραίτητες συναρτήσεις στο φύλλο λογισμικού. Αυτά είναι :
Min_tgt : Σε αυτό γράφουμε την ελάχιστη απαιτούμενη απόδοση
Incr1 : Σε αυτό γράφουμε το ποσοστό που θα προστίθεται κάθε φορά στην απαιτούμενη απόδοση.
Niter : Σε αυτό γράφουμε τον αριθμό των επαναλήψεων που θέλουμε να κάνουμε.
Target1 : Είναι το κελί που περιέχει την απαιτούμενη απόδοση.
Change1 : Είναι τα κελιά που περιλαμβάνουν τις μεταβλητές (βάρη) και θα συμπληρωθούν από τον Solver.
Portret1: Στο κελί αυτό έχουμε την συνάρτηση που υπολογίζει την απόδοση του χαρτοφυλακίου σύμφωνα με τα βάρη συμμετοχής.
Portsd1 : Στο κελί αυτό έχουμε την συνάρτηση που υπολογίζει τον κίνδυνο του χαρτοφυλακίου σύμφωνα με τα βάρη συμμετοχής.
Τα 2 τελευταία κελιά θα έχουν αρχικά την τιμή 0 καθώς τα βάρη συμμετοχής θα προκύψουν από την ελαχιστοποίηση.

Ο κώδικας για την διαδικασία που περιγράφηκε είναι ο παρακάτω :

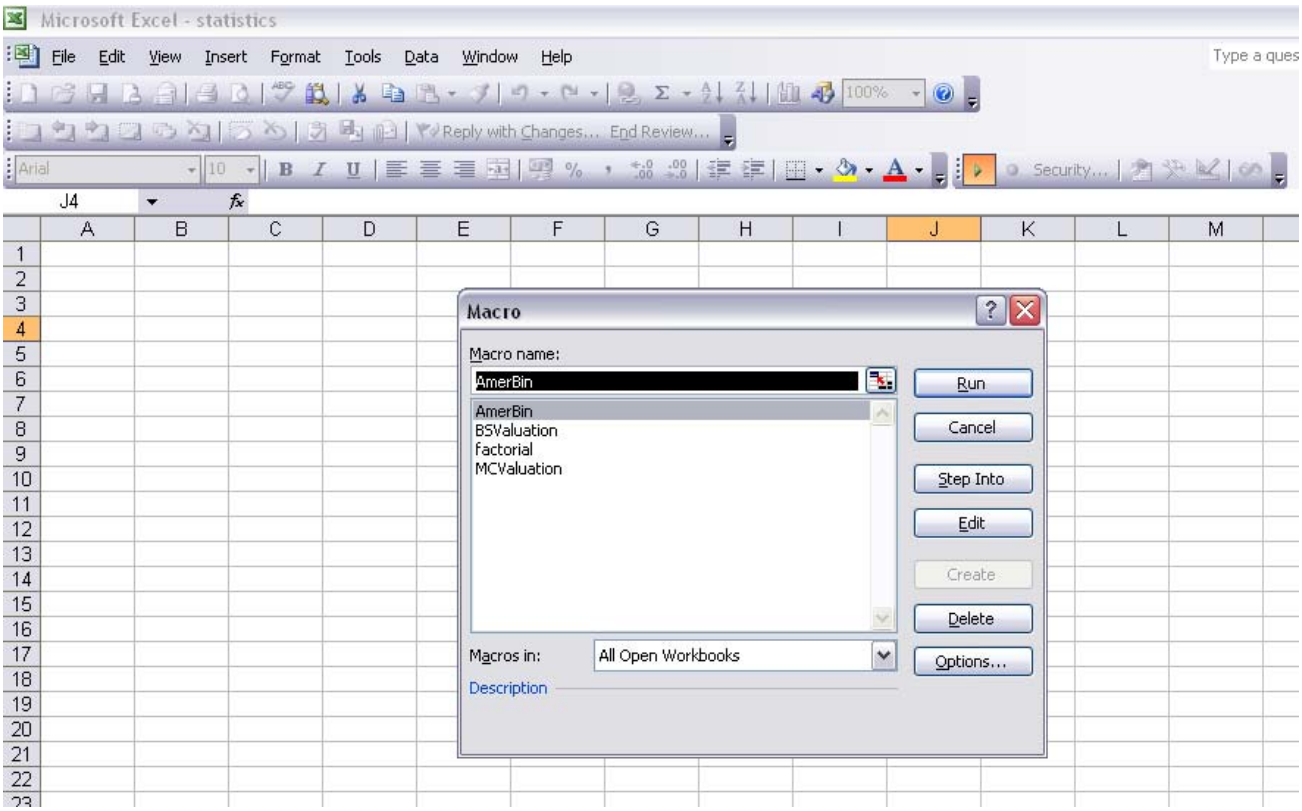
```
Sub Eff1()
'   repeated optimisation with given min_target & increment

'   initialisation
    Dim target1 As Double
    Dim incr As Single
    Dim i As Integer, niter As Integer
    target1 = Range("min_tgt1").Value
    incr = Range("incr1").Value
    niter = Range("niter1").Value
```

```
    SolverReset
    Call SolverAdd(Range("portret1"), 2, Range("target1"))
    Call SolverOk(Range("portsd1"), 2, 0, Range("change1"))

'   repeated part
'   Application.ScreenUpdating = False

    For i = 1 To niter
       Range("target1").Value = target1
       Call SolverChange(Range("portret1"), 2, Range("target1"))
       Call SolverSolve(True)
       SolverFinish

‘Antigrafi kai epikolisi twn apotelesmatwn

       Range("target1").Copy
       Range("Exp_ret").Offset(i, 0).PasteSpecial Paste:=xlValues
       Selection.NumberFormat = "0.0%"
       Range("portsd1").Copy
       Range("Exp_ret").Offset(i, 1).PasteSpecial Paste:=xlValues
       Selection.NumberFormat = "0.0%"
       Range("I10").Copy
       Range("Exp_ret").Offset(i, 2).PasteSpecial Paste:=xlValues
       Selection.NumberFormat = "0.0%"
       Range("I11").Copy
       Range("Exp_ret").Offset(i, 3).PasteSpecial Paste:=xlValues
       Selection.NumberFormat = "0.0%"
       Range("I12").Copy
       Range("Exp_ret").Offset(i, 4).PasteSpecial Paste:=xlValues
       Selection.NumberFormat = "0.0%"

       target1 = target1 + incr
    Next i

    Range("target1").Select
    Application.CutCopyMode = False
    Application.ScreenUpdating = True

    End Sub
```

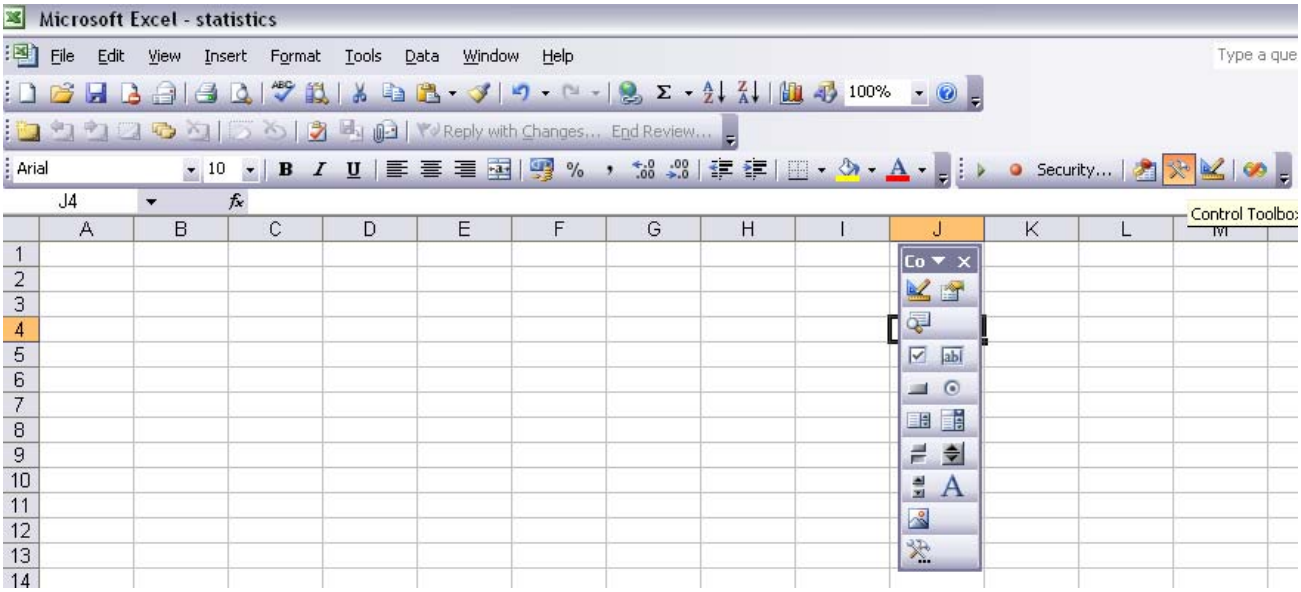
1.2.17. Εκτέλεση του κώδικα

1.2.17.1. Υπορουτίνες

Για να εμφανιστούν όλες οι υπορουτίνες που έχουμε δημιουργήσει και να μπορούμε να τις τρέξουμε μπορούμε από το μενού εργαλείων της VBA να πατήσουμε το πλήκτρο "Run Macro" που φαίνεται στην παρακάτω εικόνα. Αυτομάτως ανοίγει ένα παράθυρο το οποίο περιλαμβάνει όλες τις μακροεντολές που έχουμε γράψει. Μπορούμε να επιλέξουμε την μακροεντολή που επιθυμούμε και να πατήσουμε την επιλογή «Run» ώστε να εκτελεστεί ο κώδικας.



Εναλλακτικά θα μπορούσαμε στο φύλλο εργασίας όπου δουλεύουμε να δημιουργήσουμε πλήκτρα ώστε να μπορούμε να καλούμε γρηγορότερα τις εφαρμογές που θέλουμε. Για να το πετύχουμε αυτό επιλέγουμε από το μενού εργαλείων της vba την επιλογή "Control Toolbox" που φαίνεται και στην παρακάτω εικόνα. Έτσι με την δημιουργία ενός κουμπιού μπορούμε με δεξί click να κάνουμε «assign macro» και να αναθέσουμε την μακροεντολή που θέλουμε να εκτελείται με το πάτημα του.



1.2.17.2. Συναρτήσεις

Για να τρέξουμε μία συνάρτηση που δημιουργήσαμε μπορούμε να πάμε όπως και για τις ενσωματωμένες συναρτήσεις στο f_x και από την λίστα επιλογών να επιλέξουμε "User defined functions". Με την επιλογή αυτή θα εμφανιστούν όλες οι συναρτήσεις που έχουμε δημιουργήσει σε κάποιο module. Επιλέγουμε την συνάρτηση που μας ενδιαφέρει και στη συνέχεια εισάγουμε τα ορίσματα για να γίνουν οι υπολογισμοί.

1.2.18. Καταγραφή μακροεντολών – Record Macro

Η vba έχει ενσωματωμένη μια λειτουργία η οποία αναλαμβάνει να καταγράφει και να μετατρέπει σε κώδικα τις κινήσεις που κάνουμε στο Excel. Ο κώδικας που παράγεται δεν είναι πολύ φιλικός αλλά είναι πολύ χρήσιμος σε περιπτώσεις που θέλουμε να εκτελέσουμε διαδικασίες για τις οποίες δεν ξέρουμε τις εντολές. Για παράδειγμα αν θέλουμε να κάνουμε ένα γράφημα -ο κώδικας του οποίου χρειάζεται αρκετές παραμέτρους- είναι αρκετά πιο εύχρηστο να καταγράψουμε τις κινήσεις μας με τον recorder και να πάρουμε το κομμάτι του κώδικα που μας αφορά, να το ενσωματώσουμε στον δικό μας κώδικα κάνοντας τις απαραίτητες αλλαγές.