ΑΚΑΔΗΜΙΑ ΕΜΠΟΡΙΚΟΥ ΝΑΥΤΙΚΟΥ

ΜΑΚΕΔΟΝΙΑΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΘΕΜΑ : ΚΑΤΑΣΚΕΥΗ ΣΕ VB.NET (VISUAL-BASIC) ΕΙΚΟΝΙΚΟΥ ΕΡΓΑΣΤΗΡΙΟΥ ΗΛΕΚΤΡΟΝΙΚΩΝ

ΣΠΟΥΔΑΣΤΗΣ : ΔΑΝΔΗΣ ΒΑΣΙΛΕΙΟΣ

ΕΠΙΒΛΕΠΩΝ

ΚΑΘΗΓΗΤΗΣ : ΔΡ. ΥΑΚΙΝΘΟΣ ΧΑΡΑΛΑΜΠΟΣ

ΝΕΑ ΜΗΧΑΝΙΩΝΑ

2015

ΑΚΑΔΗΜΙΑ ΕΜΠΟΡΙΚΟΥ ΝΑΥΤΙΚΟΥ
ΜΑΚΕΔΟΝΙΑΣ
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΘΕΜΑ : ΚΑΤΑΣΚΕΥΗ ΣΕ VISUAL-BASIC ΕΙΚΟΝΙΚΟΥ ΕΡΓΑΣΤΗΡΙΟΥ ΗΛΕΚΤΡΟΝΙΚΩΝ

ΣΠΟΥΔΑΣΤΗΣ : ΔΑΝΔΗΣ ΒΑΣΙΛΕΙΟΣ

ΑΜ : 4728

ΗΜΕΡΟΜΗΝΙΑ ΠΑΡΑΔΟΣΗΣ : 24/09/2015

Βεβαιώνεται η ολοκλήρωση της παραπάνω πτυχιακής εργασίας

Ο καθηγητής

Περίληψη

Στην παρούσα πτυχιακή μελετήθηκαν ορισμένες δυνατότητες της γλώσσας προγραμματισμού visual basic (VB), καθώς και κατασκευάστηκαν μερικά εικονικά εργαστήρια ηλεκτρονικών σχετικά με τη διδακτέα ύλη της Ακαδημίας Εμπορικού Ναυτικού.

Στόχος της πτυχιακής εργασίας είναι τόσο να επεξηγήσει τα προγράμματα – εικονικά εργαστήρια που κατασκευάστηκαν, όσο και να δώσει τη δυνατότητα σε κάθε ενδιαφερόμενο να ξεκινήσει κάτι δικό του. Θα είναι, δηλαδή, και μια προσπάθεια εισαγωγής στον προγραμματισμό.

Βέβαια οι εφαρμογές τις VB, όπως και κάθε γλώσσας προγραμματισμού, είναι αμέτρητες, οπότε και δεν πρόκειται για ένα, τρόπο τινά, εγχειρίδιο εκμάθησής της, αλλά για μία εργασία η οποία παραθέτει ένα σύνολο πληροφοριών για τα προγράμματα που περατώθηκαν.

Πιο συγκεκριμένα θα γίνουν αναφορές στην ιστορική εξέλιξη της VB από το 1991, που κυκλοφόρησε, έως σήμερα, σε ορισμένες δυνατότητες εφαρμογών που έχει, στις εντολές που χρησιμοποιήθηκαν για την κατασκευή των εικονικών εργαστηρίων, σε χαρακτηριστικά της γνωρίσματα και συσχετισμούς με παράγωγες γλώσσες από αυτήν και τέλος στα ίδια τα εργαστήρια ηλεκτρονικών που κατασκευάστηκαν.

Abstract

In the present thesis certain potentialities of the programming language visual basic (VB) were studied, as well as some virtual electronics labs were built relevant to the curriculum of Merchant Marine Academy.

Purpose of this thesis is so to explain the programs – virtual labs that were built, as to give the potentiality to anyone interested to commence his own. It will also be an effort of an introduction to programming.

Nevertheless the applications of VB, like every other's programming language, are countless, so this is not going to be a tutorial manual, but a project that will appose a set of information related to the programs that were built.

More specifically there will be reports to VB's historic evolution from 1991, that was released, until today, to certain application potentials, to the commands used in building the virtual labs, to distinctive features and connections with derivative programming languages and finally to the virtual labs themselves.

Πρόλογος

Προγραμματισμός υπολογιστών είναι το σύνολο των διαδικασιών σύνταξης ενός προγράμματος, ως υλοποίηση κάποιων αλγορίθμων ύστερα από προσεκτική σχεδίαση, για την αυτοματοποιημένη εκτέλεση εργασιών ή την επίλυση κάποιου προβλήματος από έναν υπολογιστή. Περιλαμβάνει τον έλεγχο του προγράμματος για την επαλήθευση της ακρίβειας και της ορθότητας του – αποσφαλμάτωση (debugging), και την προπαρασκευή των οδηγιών με τις οποίες ένας υπολογιστής θα εκτελέσει τις εντολές που καθορίζονται στις προδιαγραφές του προγράμματος.

Θεμελιώδη ρόλο διαδραματίζουν οι διαφορετικές γλώσσες προγραμματισμού, δηλαδή οι τυποποιημένες σειρές εντολών απαραίτητες για τη σύνθεση ενός προγράμματος. Ο πηγαίος κώδικας, δηλαδή το κείμενο που έχει συνταχθεί σε μια συγκεκριμένη γλώσσα, πρέπει να μεταφραστεί σε γλώσσα μηχανής από εξειδικευμένο λογισμικό (compiler), πρόγραμμα στο οποίο συντάσσεται ο κώδικας στην εκάστοτε γλώσσα. Ο compiler το αποσφαλματώνει και τελικά δημιουργεί το εκτελέσιμο από τον υπολογιστή προϊόν.

Η VB είναι γλώσσα προγραμματισμού τρίτης γενιάς οδηγούμενη από συμβάντα (event driven) και έχει ολοκληρωμένο περιβάλλον ανάπτυξης (IDE – integrated development environment) από τη Microsoft για το μοντέλο προγραμματισμού COM (component object model). Η VB θεωρείται μία σχετικά εύκολη γλώσσα προγραμματισμού στην εκμάθηση και τη χρησιμοποίηση, λόγω των χαρακτηριστικών της, καθώς έχει γραφικό περιβάλλον χρήστη (GUI – graphical user interface) και συγγένεια με τη γλώσσα BASIC. Προέρχεται από τη BASIC και επιτρέπει την ταχεία ανάπτυξη εφαρμογών (RAD – rapid application development) με GUI, πρόσβαση σε βάσεις δεδομένων χρησιμοποιώντας αντικείμενα (data access objects) και τη δημιουργία στοιχείου ελέγχου ActiveX και αντικειμένων.

Σε VB είναι δυνατόν να δημιουργηθούν από μια απλή εφαρμογή, από προγραμματιστή που χρησιμοποιεί στοιχεία που παρέχονται από την ίδια τη VB, μέχρι προγράμματα που χρησιμοποιούν Windows API (application programming interfaces), τα οποία είναι ομάδες εφαρμογών χρησιμοποιούμενα από τον πυρήνα του λειτουργικού συστήματος των windows και απαιτούν εξωτερικές λειτουργικές δηλώσεις.

Κεφάλαιο 1

ΕΙΣΑΓΩΓΗ

1.1 Η ΕΝΝΟΙΑ ΤΟΥ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Προγραμματισμός είναι η σχεδίαση οδηγιών για να ζητήσουμε από έναν υπολογιστή να εκτελέσει ορισμένες εργασίες γρηγορότερα και αποδοτικότερα απ' ότι θα μπορούσε ένας άνθρωπος. Μια μονάδα επεξεργασίας υπολογιστή (CPU) μπορεί να καταλάβει και να διαβάσει μόνο τα 0 και 1 (δυαδικό σύστημα), οπότε και η παλαιότερη γλώσσα προγραμματισμού, η γλώσσα μηχανής, είναι εξαιρετικά δύσκολη στην εκμάθηση και χρήση. Έτσι εφευρέθηκαν γλώσσες προγραμματισμού βασισμένες σε ανθρώπινα πρότυπα γραφής, οι οποίες χρησιμοποιούν συγκεκριμένες εντολές και συντάσσονται σε περιβάλλον εξειδικευμένου λογισμικού, το οποίο με τη σειρά του τις μεταφράζει σε γλώσσα μηχανής. Μια τέτοια γλώσσα είναι και η VB.

1.2 Η ΙΣΤΟΡΙΚΗ ΕΞΕΛΙΞΗ ΤΗΣ VB

Το 1990 ξεκίνησε το 'σχέδιο κεραυνός' (project thunder) . Το 1991 κυκλοφόρησε η VB 1.0 για Windows στο παγκόσμιο συνέδριο εμπορίου στην Ατλάντα και ακολούθησε η έκδοσή της για DOS το 1992. Αργότερα το 1992 κυκλοφόρησε η VB 2.0 με πιο εύχρηστο περιβάλλον χρήστη και μεγαλύτερη ταχύτητα. Η έκδοση της VB 3.0 ήταν η αποκάλυψη το καλοκαίρι του 1993, η οποία θεωρήθηκε ως επαγγελματική. Το 1995 η VB 4.0 πλέον έβγαινε σε 3 εκδόσεις: standard, professional και enterprise και μπορούσε να κατασκευάσει 16-bit και 32-bit προγράμματα Windows. Το 1997 κυκλοφόρησε η VB 5.0 με ακόμα περισσότερες δυνατότητες, ενώ η τελική έκδοση 6.0 ήταν το 1998. Η υποστήριξη της VB 6.0 από τη Microsoft θα τελείωνε το 2005 αλλά πήρε παράταση έως το 2008.

Από το 2002 είχε ήδη κυκλοφορήσει μια διαφορετική έκδοση η VB.NET, η οποία ήταν και είναι η διάδοχος της VB (πλέον έχει καταργηθεί η κατάληξη .NET). Το αν η VB.NET είναι η πραγματική διάδοχος της VB είναι ένα θέμα που έχει διχάσει την κοινότητα των προγραμματιστών. Υπάρχουν καινούριες προσθήκες και υποστηρίζονται καινούρια χαρακτηριστικά καθώς και υπάρχουν αλλαγές σε ορισμένες εντολές και τύπους δεδομένων. Η μεγαλύτερη, ωστόσο, διαφορά είναι στη σημασιολογία τους. Ενώ η αρχική VB ήταν event-driven, όπως προαναφέρθηκε, πλέον είναι event-based.

Κεφάλαιο 2

Η VB(.NET) ΣΤΟ VISUAL STUDIO

2.1 Η σύγχρονη μορφή της VB και εφαρμογές της

Από το 2005 η VB.NET γίνεται επίσημα VB και κυκλοφορεί μέσα στο πακέτο του Visual Studio. Η παρούσα πτυχιακή πραγματοποιήθηκε σε Visual Studio 2010 ενώ η τελευταία έκδοση είναι αυτή του 2015. Οι διαφορές είναι αμελητέες και ο κώδικας γραφής ακριβώς ο ίδιος. Οι εκδόσεις των Visual Studio είναι εμπορικές, διατίθενται δηλαδή επί πληρωμή, εκτός από το Visual Studio Express και Visual Studio Community που είναι δωρεάν. Το Visual Studio είναι ο compiler της VB, δηλαδή το λογισμικό όπου γίνεται η κωδικοποίηση της γλώσσας, χρησιμοποιώντας 'δηλώσεις' για να καθοριστούν πράξεις.

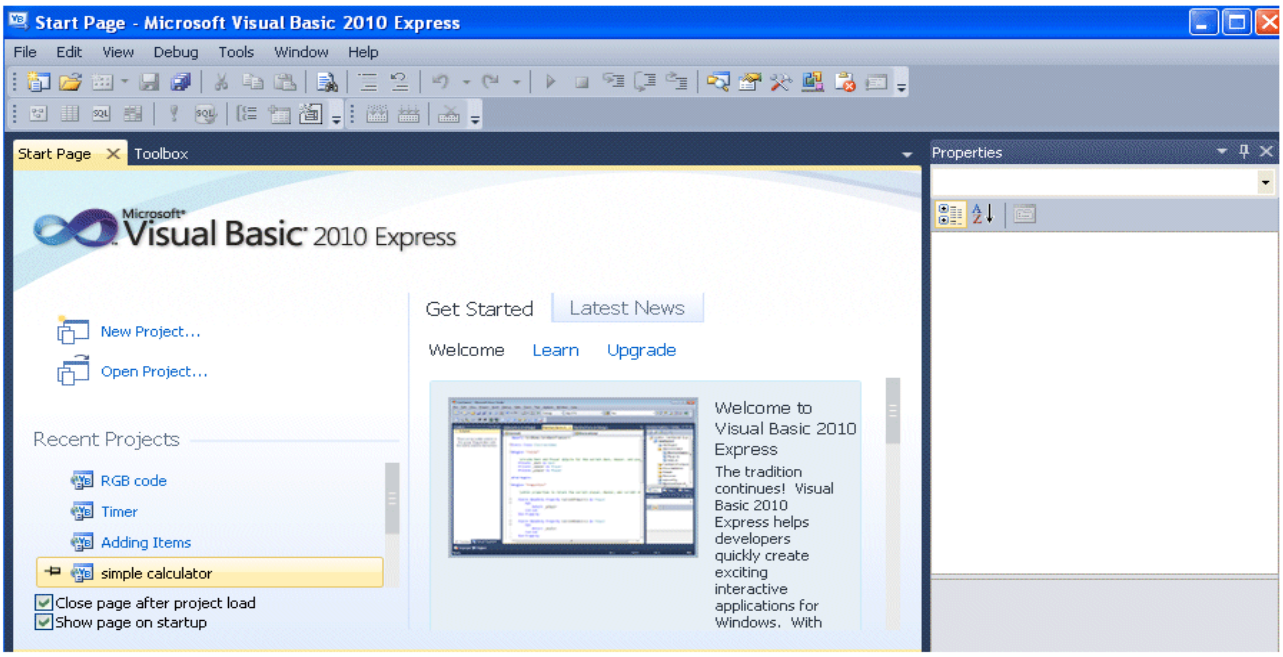
Σε VB μπορούν να κατασκευαστούν προγράμματα για οποιοδήποτε σκοπό. Μαθηματικής φύσεως προγράμματα όπως γεωμετρικοί πρόοδοι, ταυτόχρονης επίλυσης εξισώσεων, πρώτων αριθμών, παραγοντοποίησης, κατασκευής γραφικών παραστάσεων κλπ. Επιστήμονες μπορούν να κατασκευάσουν προγράμματα υπολογισμού τροχιάς, αρμονικών κινήσεων, πορείας ουράνιων σωμάτων και πάει λέγοντας. Εφαρμογές στις επιχειρήσεις για απογραφή αποθεμάτων, υπολογισμό απόσβεσης, μισθοδοσία, υπολογισμό απόδοσης επενδύσεων κλπ. Ακόμα είναι δυνατή η κατασκευή πολυμέσων και παιχνιδιών. Ουσιαστικά οι δυνατότητες δημιουργίας προγραμμάτων είναι απεριόριστες και βασίζονται στις ανάγκες του καθενός.

Το ενσωματωμένο περιβάλλον ανάπτυξης όταν ξεκινάς το Visual Studio 2010 Express αποτελείται από ορισμένους τομείς:

- Ο new project/open project τομέας για να ξεκινήσει ή να συνεχίσει να δουλεύει κάποιος ένα πρόγραμμα
- Ο recent projects τομέας όπου φαίνεται μια λίστα με τα πρόσφατα προγράμματα που δημιουργήθηκαν
- Η get started καρτέλα που παρέχει συμβουλές ή τη δυνατότητα αναβάθμισης
- Η latest news καρτέλα που ενημερώνει για νέες εκδόσεις
- Ο τομέας properties που δίνει τη δυνατότητα ρυθμίσεων

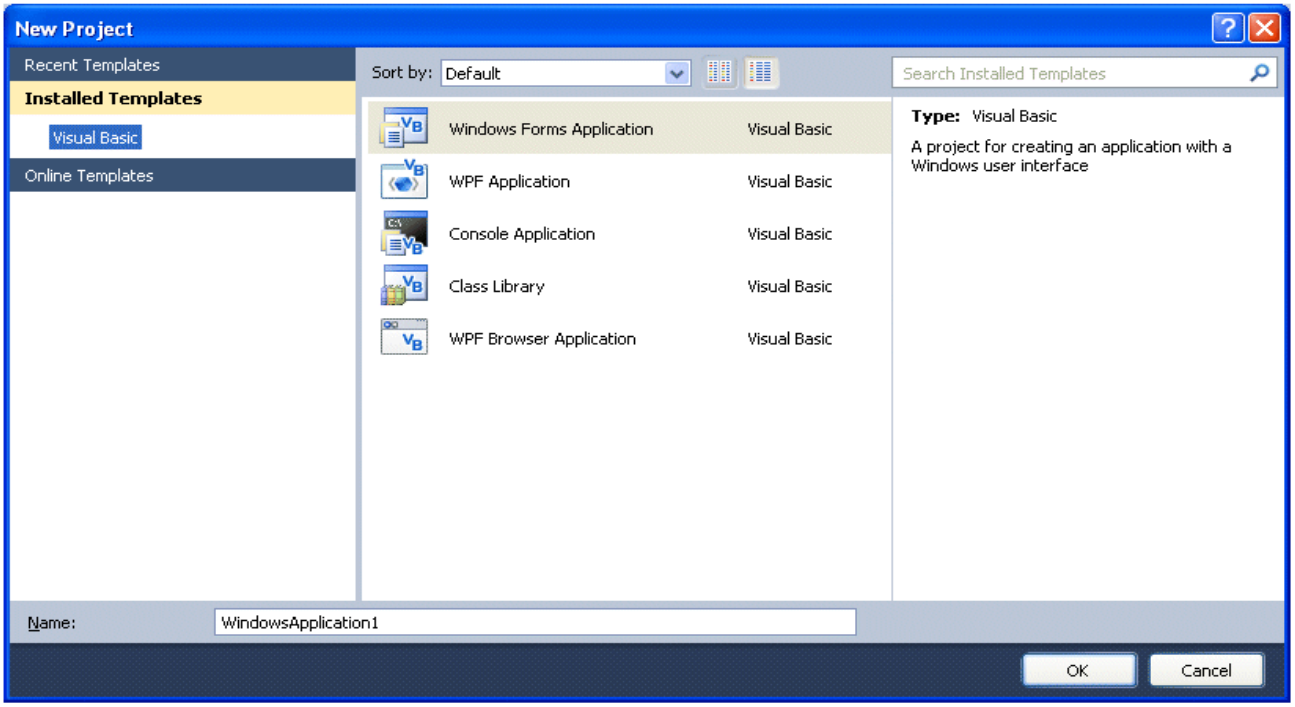
2.2 Το γραφικό περιβάλλον

Ξεκινώντας το Visual Studio:



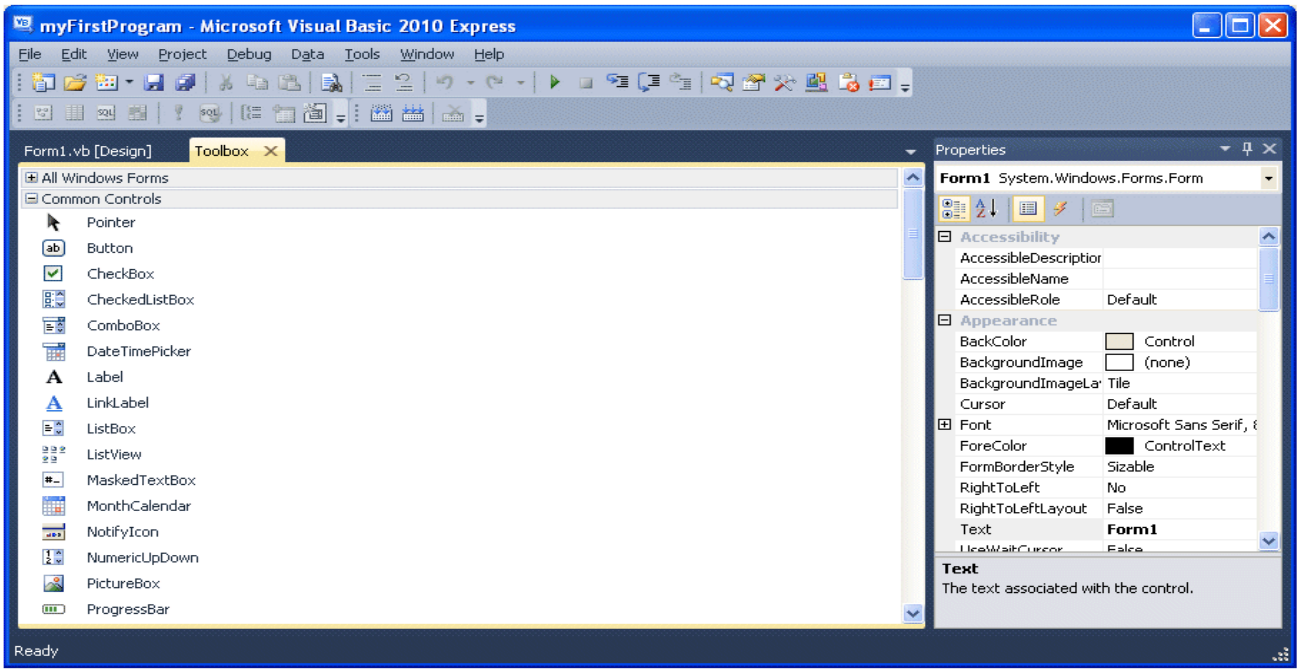
Εικόνα 2.2.1 Αρχικό παράθυρο Visual Studio

Για τη δημιουργία νέας εφαρμογής γίνεται η επιλογή του new project και εμφανίζεται το νέο παράθυρο VB2010:

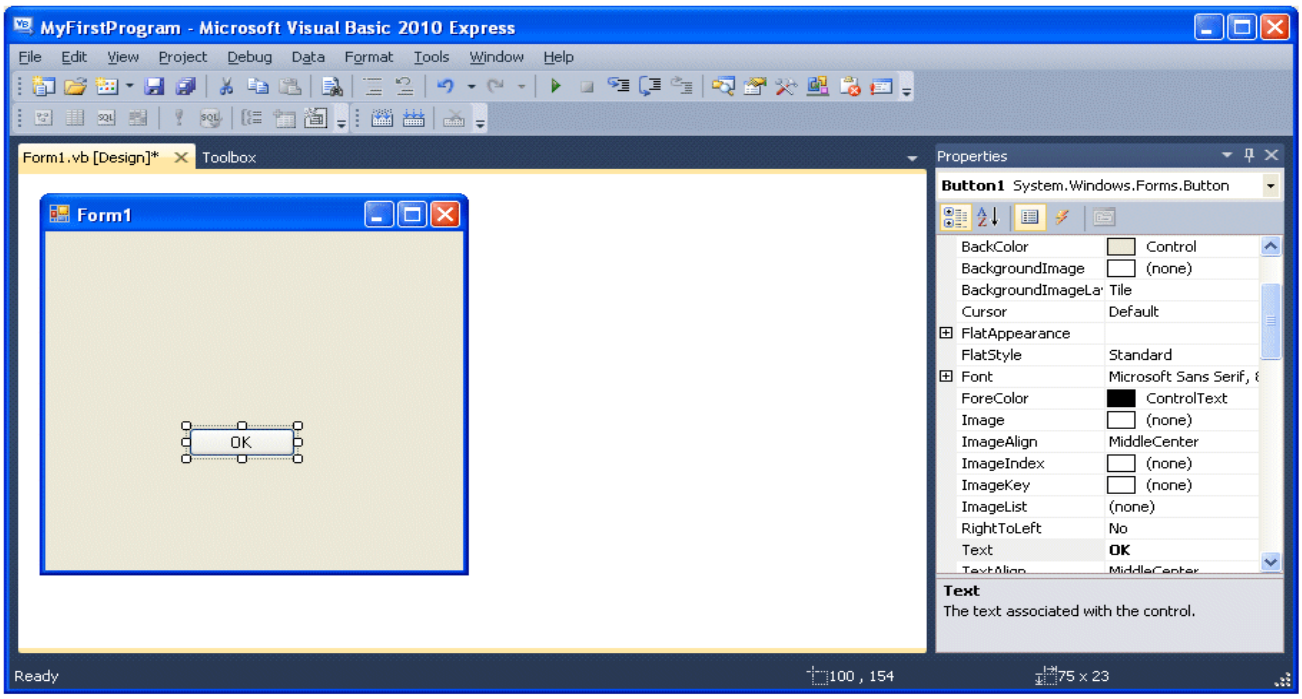


Εικόνα 2.2.2 New project dialog box

Το dialog box προσφέρει 5 επιλογές. Τα προγράμματά μας κατασκευάστηκαν σε windows form application οπότε και αυτή θα είναι η επιλογή μας. Επίσης παρέχεται το προκαθορισμένο όνομα WindowsApplication1, το οποίο και θα αλλάξουμε σε myFirstProgram. Πατώντας το OK κουμπί εισερχόμαστε στο γραφικό περιβάλλον της VB.



Εικόνα 2.2.3 Καρτέλα toolbox IDE Windows



Εικόνα 2.2.4 Καρτέλα form1.vb (design) έχοντας επιλέξει ‘button’ από το toolbox (με τη μέθοδο ‘drag and drop’) και έχοντας μετονομάσει το κείμενο – ‘text’ σε ‘OK’

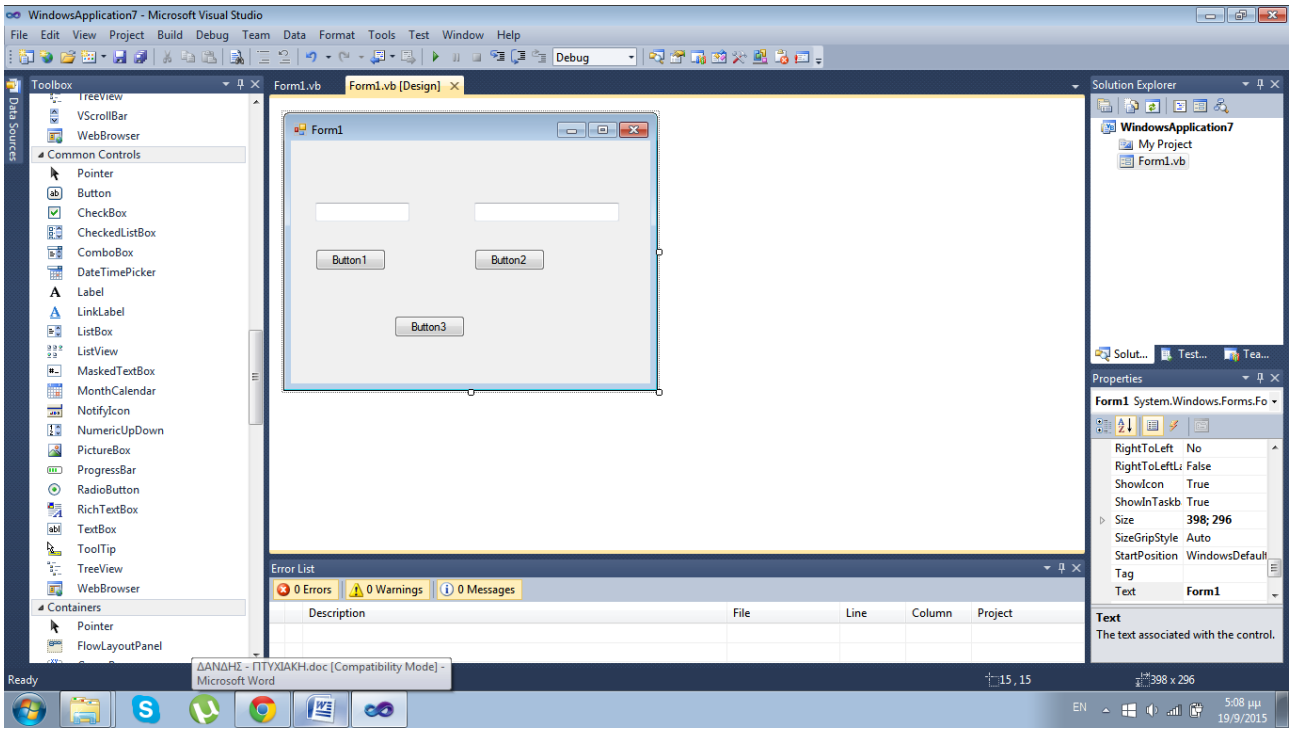
Κεφάλαιο 3

ΠΡΟΓΡΑΜΜΑΤΑ ΚΑΙ ΕΝΤΟΛΕΣ

3.1 Ξεκινώντας με ένα απλό πρόγραμμα

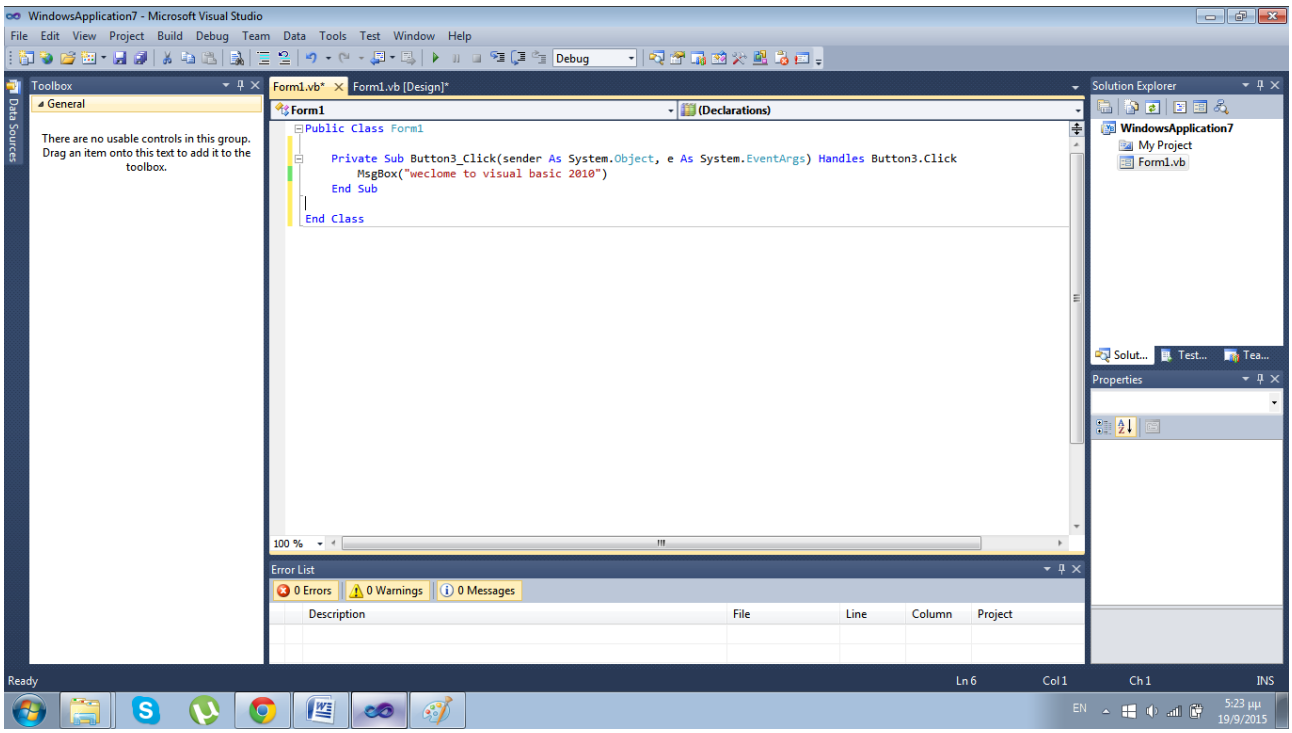
Το σημαντικότερο πράγμα για το γράψιμο ενός προγράμματος είναι το τρίπτυχο **ΤΙ** θέλω να κάνω, **ΠΟΥ** θέλω να το κάνω, **ΠΟΤΕ** θέλω να το κάνω να φαίνεται! Αυτό θα εξηγηθεί αναλυτικότερα με παραδείγματα στη συνέχεια.

Πριν ο χρήστης ξεκινήσει να προγραμματίζει, εννοείται πως υπάρχει μια κεντρική ιδέα. Παίρνοντας ως πρότυπο τις εικόνες του 2^{ου} κεφαλαίου θα φτιάξουμε ένα απλό πρόγραμμα που θα μας καλωσορίζει στη VB με 3 διαφορετικούς τρόπους.



Εικόνα 3.1.1 form1.vb (design) έχοντας επιλέξει 3 button (κουμπιά) και 2 textbox (πλαίσια κειμένου) από το toolbox για τις ανάγκες του προγράμματός μας

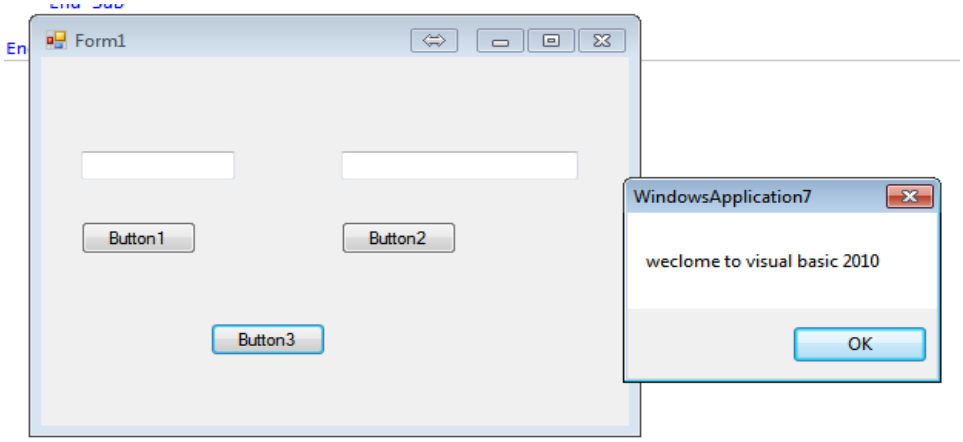
Για να ξεκινήσουμε τη γραφή του κώδικα, πρέπει απλά να κάνουμε double-click στο εργαλείο που θέλουμε να προγραμματίσουμε. Στην προκειμένη περίπτωση θα ξεκινήσουμε απ' το button3.



Εικόνα 3.1.2 form1.vb το παράθυρο όπου γίνεται η γραφή του κώδικα

Ο κώδικας σε αυτή την εικόνα είναι μόνο η μία σειρά που γράφει 'MsgBox("welcome to visual basic 2010")'. Τα υπόλοιπα παράγονται αυτόματα από τον compiler. Όλες οι δηλώσεις πρέπει να έχουν αρχή και τέλος, όπως φαίνεται από τον τρόπο που ξεκινάει η φόρμα μας 'Public Class Form1… End Class' και η δημιουργία μιας εντολής 'Private Sub Button3_Click … End Sub'. Οι private sub εντολές είτε δημιουργούνται αυτόματα είτε τις κατασκευάζουμε για να εισάγουμε δηλώσεις ή συμβάντα που χρησιμοποιούμε στο πρόγραμμά μας.

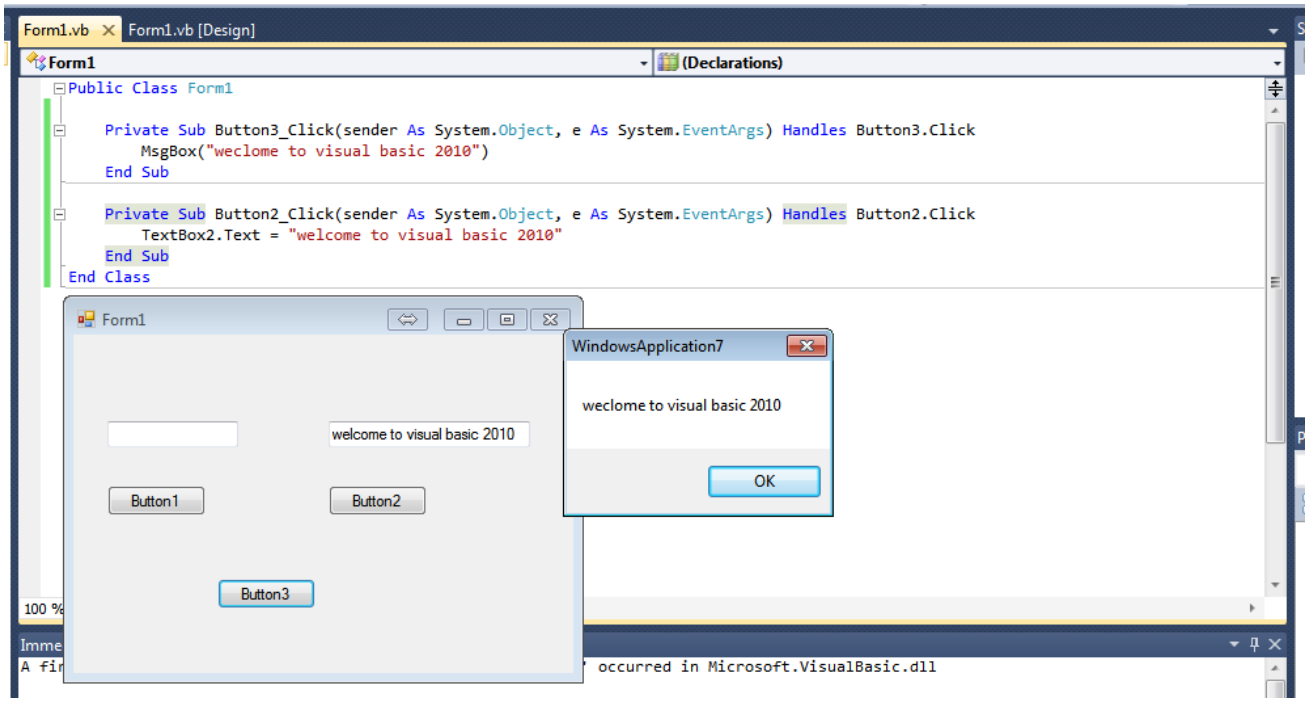
Για να δούμε το αποτέλεσμα του προγράμματός μας πατάμε το πλήκτρο 'play' στη γραμμή εργαλείων ή αλλιώς 'start debugging'.



Εικόνα 3.1.3 Παράδειγμα προγράμματος σε VB

Εμφανίζεται πρώτα το παράθυρο form1 και κάνοντας click στο button3 εμφανίζεται το μήνυμα καλωσορίσματος.

Με τον ίδιο τρόπο προγραμματίζουμε το button2. Διπλό click για να ανοίξει καινούρια ενότητα εντολής και γράφουμε τον καινούριο μας κώδικα.



Εικόνα 3.1.4 Παράδειγμα προγράμματος σε VB

Παρατηρούμε πως η 'Private Sub Button2_Click … End Sub' δημιουργήθηκε και εδώ ενώ ο κώδικάς μας είναι το 'TextBox2.Text = "welcome to visual basic 2010"'. Αυτή τη φορά το κάνοντας click το button2 εμφανίζεται το μήνυμα πάνω από αυτό (αυτό είναι το TextBox2), ενώ κάνοντας click το button3 συνεχίζει και εμφανίζεται το παράθυρο καλωσορίσματος.

Ο 3ος και τελευταίος κώδικας θα βοηθήσει καλύτερα στην κατανόηση του τρίπτυχου του προγραμματιστή – ΤΙ, ΠΟΥ, ΠΟΤΕ.

Ο κώδικας θα είναι λίγο πιο πολύπλοκος καθ' 'ότι θα κάνουμε το μήνυμα να εμφανίζεται στο πλαίσιο κειμένου (TextBox1) που βρίσκεται πάνω από το button1 μία λέξη κάθε φορά που θα κάνουμε click το button1.

```
    Private Sub Button1_Click(sender As System.Object, e As System.EventArgs) Handles Button1.Click
        TextBox1.TextAlign = HorizontalAlignment.Center
        If TextBox1.Text = "" Then
            TextBox1.Text = "welcome"
        ElseIf TextBox1.Text = "welcome" Then
            TextBox1.Text = "to"
        ElseIf TextBox1.Text = "to" Then
            TextBox1.Text = "visual"
        ElseIf TextBox1.Text = "visual" Then
            TextBox1.Text = "basic"
        ElseIf TextBox1.Text = "basic" Then
            TextBox1.Text = "2010 !!!"
        End If
    End Sub
End Class
```

Εικόνα 3.1.5 Αλλαγή κειμένου με κάθε click του button1

Εντός της ‘Private Sub Button1… End Sub’ έχουμε γράψει έναν κώδικα ώστε με κάθε click να αλλάζει το κείμενο. Το textbox2 δεν αλλάζει όσες φορές και να πατήσουμε το button2 γιατί πολύ απλά ο κώδικας υποδεικνύει στον υπολογιστή ότι με το πάτημα του button2 θέλω να εμφανίζεται το μήνυμα.

Η πρώτη γραμμή κώδικα το ‘TextAlign’ είναι για να εμφανίζεται το μήνυμα στο κέντρο του textbox2, εικαστική παρέμβαση και όχι ιδιαίτερης σημασίας. Η εντολή που κάνει τη διαφορά είναι η ‘If … End If’ και κάνει αυτό ακριβώς που φαίνεται, δηλαδή ‘κοιτάει’ τι γράφει το κείμενο και πράττει ανάλογα. ΑΝ το κείμενο γράφει ‘κενό’ (όταν ανοίγουμε δηλαδή το πρόγραμμα) ΤΟΤΕ με το πάτημα του κουμπιού button1 το κείμενο να γίνει ‘welcome’ και πάει λέγοντας.

Στην τελευταία γραμμή του κώδικά μας μπορούμε να γράψουμε ακόμα μια εντολή ώστε να μη σταματάει στο ‘2010 !!!’ αλλά να ξεκινάει πάλι απ’ την αρχή…

```
ElseIf Textbox1.Text = “2010!!!” Then
TextBox1.Clear ()
```

Έτσι σβήνει το κείμενο (2010!!!) και ξεκινάει απ’ την αρχή

ΤΙ θέλω να κάνω...να πατάω το button1 και να εμφανίζεται ένα διαφορετικό κείμενο κάθε φορά που το πατάω.

ΠΟΥ θέλω να το κάνω...αυτό φαίνεται καλύτερα σε πιο πολύπλοκα προγράμματα όπου αυτό που κάνουμε επενεργεί σε συγκεκριμένο σημείο του, αλλά εδώ για χάριν επεξήγησης ας πούμε ότι το **ΠΟΥ** είναι το textbox1.

ΠΟΤΕ θέλω να το κάνω...όποτε βλέπω μια συγκεκριμένη λέξη, να πατάω το button1 και να αλλάζει.

Έχουμε λοιπόν το πρώτο ολοκληρωμένο μας πρόγραμμα σε VB. Επιλέγοντας έναν από τους 3 τρόπους (καλύτερα όχι τον 3^{ο} ακόμα) φτιάξαμε ένα παράθυρο με ένα μήνυμα καλωσορίσματος στη VB.

3.2 Δηλώσεις και απλές εντολές

Οτιδήποτε πρόκειται να χρησιμοποιήσουμε στη VB, είτε είναι αριθμός είτε λέξη είτε οτιδήποτε άλλο, πρέπει να το δηλώσουμε. Σε ορισμένες περιπτώσεις, όπως στο πρόγραμμα της προηγούμενης ενότητας, γίνεται να το αποφύγουμε, βάζοντας την έννοιά μας σε "αυτάκια". Αυτό γίνεται με την έννοια Dim και μια σειρά διαφορετικών συντάξεων αλλά οι 2 συνηθέστεροι που συντελούν και στη αποφυγή σφάλματος λόγω συστήματος είναι:

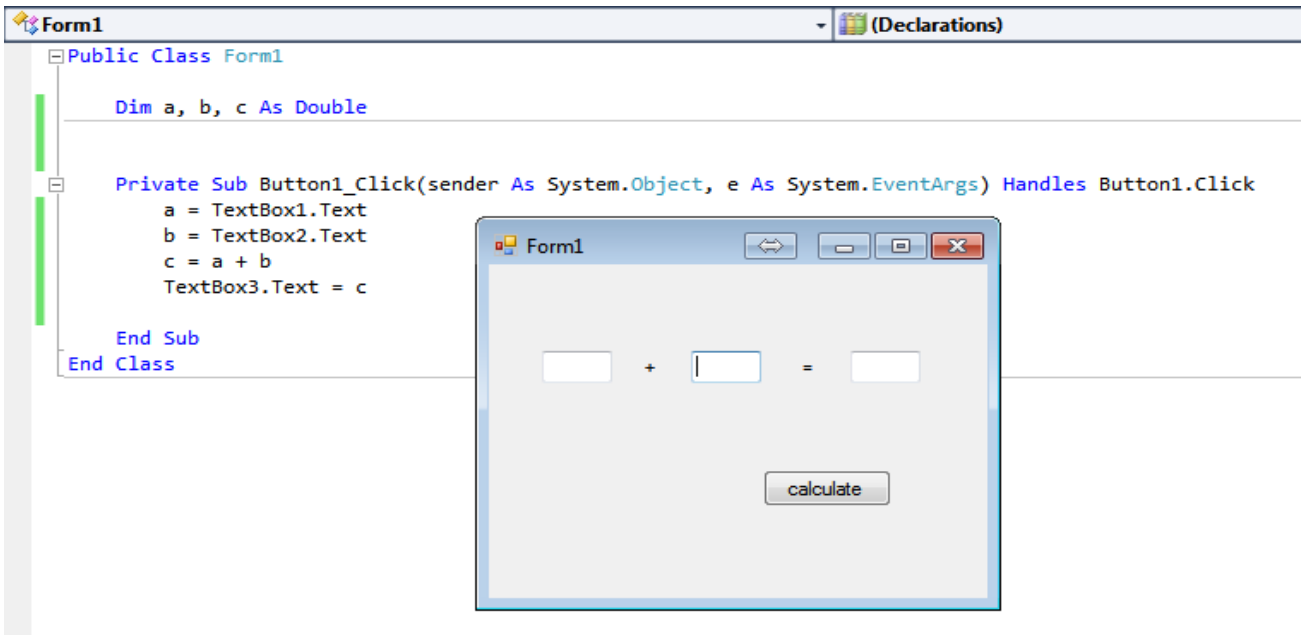
1. Dim x As *something1
2. Dim x As *something2 = a

Το Dim … As είναι η έκφραση και το x είναι η ποσότητα που δηλώνουμε. Στη δεύτερη περίπτωση το x το δηλώνουμε ως κάτι με αρχική τιμή a. Μπορεί να είναι αριθμός (ακέραιος, δεκαδικός, μιγαδικός κλπ.), έκφραση (μεγάλη ή μικρή, ψευδής ή αληθής κλπ.) ή ακόμα και διεπαφή (κάλεσμα καινούριας κλάσης, συμβάντος κλπ.).

Κάνοντας διπλό click στο παράθυρο μας, οπουδήποτε και όχι συγκεκριμένα σε ένα εργαλείο όπως κάναμε με τα buttons προηγουμένως, εμφανίζεται η φόρμα όπου γράφουμε τον κώδικα με τα όρια της κλάσης 'Private Class Form1 … End Class'. Εκεί γίνονται, για να υπάρχουν μαζεμένες για ευκολία ελέγχου, οι δηλώσεις των παραμέτρων, τις οποίες έχουμε τη δυνατότητα να καλέσουμε οποτεδήποτε κατά τη διάρκεια του προγράμματός μας μέσα στις διαδικασίες 'Private Sub … End Sub'. Δεν είναι απαραίτητο να δηλωθούν όλες οι παράμετροι από την αρχή, καθώς κατά τη διάρκεια γραφής ενός προγράμματος μπορεί να προκύψουν κι άλλες, αλλά σε κάθε πρόταση του κώδικα ο compiler ελέγχει αν αναγνωρίζει μια δήλωση και αν δεν την αναγνωρίζει υπογραμμίζεται αυτόματα και θεωρείται ως εσφαλμένη έως ότου δηλωθεί.

Μερικές φορές μπορεί απλά να υπάρχουν λανθασμένοι συσχετισμοί ή λανθασμένη χρήση των παραμέτρων και να μη λειτουργεί το πρόγραμμα με τον επιθυμητό τρόπο. Γι' αυτό υπάρχει το line break, μια πράξη ελέγχου που επισημαίνεις γραμμές του προγράμματος βάζοντας κουκίδες στο αριστερό όριο της φόρμας και το debugging γίνεται κάθε φορά που κάνεις click to 'start debugging' κουμπί, μέχρι το σημείο του line break. Έτσι ελέγχεις το πρόγραμμα σε τμήματα και βλέπεις ποιο είναι αυτό το κομμάτι που έχει γίνει το λάθος. Ουσιαστικά τεμαχίζεις το πρόγραμμα σε μικρότερα για τον ευκολότερο έλεγχό του.

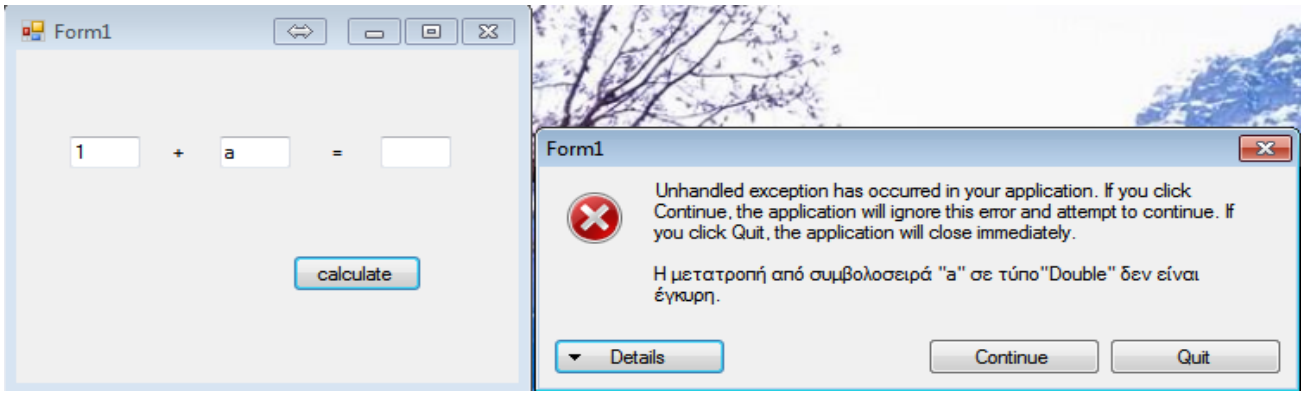
Μεγάλη προσοχή απαιτείται στη σειρά των γεγονότων. Ο υπολογιστής εκτελεί εντολές με τη σειρά που γράφονται από το χρήστη. Διαφορετικό πράγμα είναι οι δηλώσεις και κάλεσμα εξισώσεων και μεταβλητών που μπορούν να γίνουν οποιαδήποτε στιγμή και διαφορετικό πράγμα η εκτέλεση μέσα σε μία υποκλάση.



Εικόνα 3.2.1 Απλό πρόγραμμα πρόσθεσης σε VB

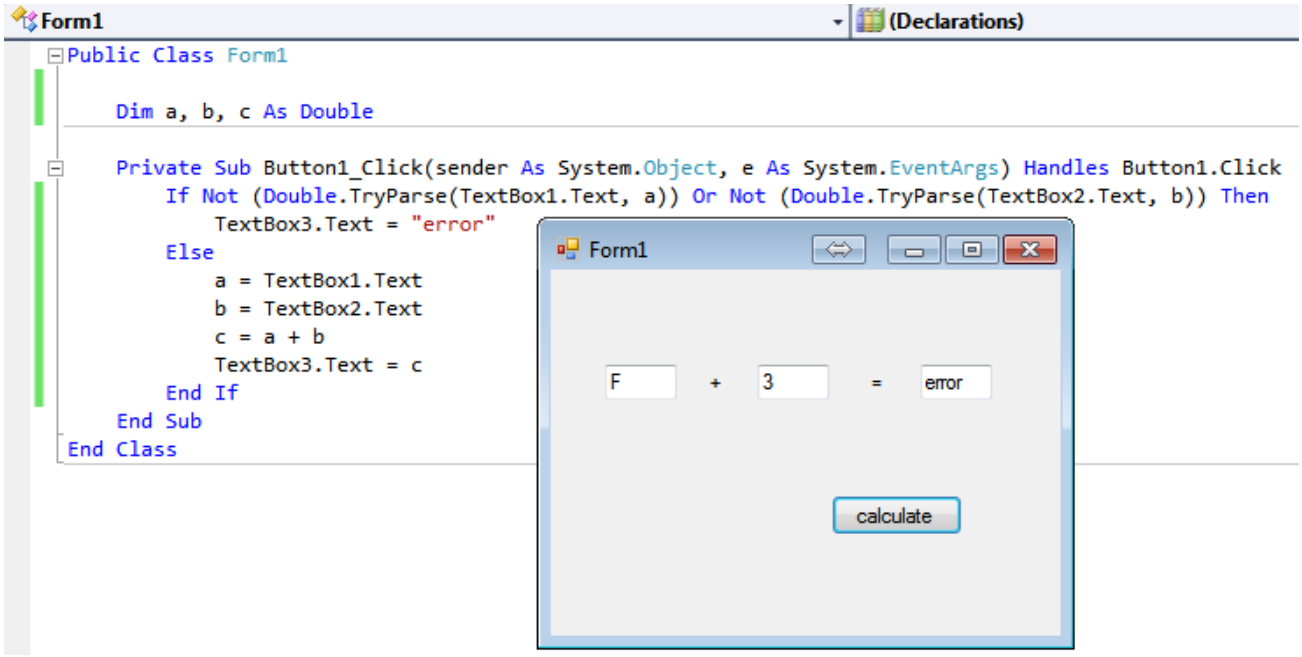
Η δήλωση 'Double' θέτει τα a, b και c ως αριθμούς δεκαδικούς. Αυτή θα μπορούσε να γίνει και εντός της υποκλάσης 'Private Sub … End Sub'. Οι δηλώσεις που συντάχθηκαν εντός της υποκλάσης δεν είναι δυνατόν να διατυπωθούν με άλλη σειρά είτε η κάθε μία χωριστά είτε ως σύνολο. Σε μια ισότητα το δεύτερο μέρος είναι αυτό που καθορίζει το πρώτο και η σειρά που γράφονται από πάνω προς τα κάτω είναι και η σειρά που εκτελούνται.

Το παραπάνω πρόγραμμα είναι διαμορφωμένο απλοϊκά για την επίδειξη του σωστού τρόπου γραφής, αλλά ελλιπές. Αν στην εφαρμογή που δημιουργήσαμε βάλουμε οτιδήποτε άλλο εκτός από αριθμούς, δηλαδή γράμματα, σύμβολα κλπ, παίρνουμε 'system error'.



Εικόνα 3.2.2 Σφάλμα εφαρμογής εξ' αιτίας ακάλυπτων ενδεχομένων

Στην παραπάνω περίπτωση τα a, b και c έχουν δηλωθεί ως αριθμοί. Όταν εισάγονται γράμματα ο υπολογιστής δεν μπορεί να το αντιμετωπίσει γιατί ο προγραμματιστής δεν του υπέδειξε τον τρόπο. Το σωστό πρόγραμμα είναι αρκετά πιο πολύπλοκο γιατί περιλαμβάνει μια εντολή που κάνει έλεγχο στις τιμές και αν δεν πάρει αντιστοιχία σε αυτό που της ζητείται επιστρέφει το λάθος. Η πολυπλοκότητα οφείλεται στον τρόπο χρήσης και σύνταξης της.



Εικόνα 3.2.3 Απλή πράξη πρόσθεσης που προλαμβάνει την 'κατάρρευση' της εφαρμογής αν εισαχθεί οτιδήποτε άλλο εκτός από αριθμούς

Γίνεται χρήση If – If Not , Else , Or και TryParse. Το If με το If Not χρησιμοποιούνται με τον ίδιο τρόπο αλλά σε αντίθετες καταστάσεις και μεταφράζονται ως εξής: ΑΝ ισχύει κάτι, ΤΟΤΕ να γίνει το τάδε, ΑΛΛΙΩΣ το τάδε, ΤΕΛΟΣ ΑΝ. Το TryParse είναι η εντολή που ελέγχει μια δήλωση και συντάσσεται, όπως φαίνεται, ακολουθώντας το σύνολο των τιμών που έχουμε δώσει (double είναι ένα εύρος τιμών δεκαδικών αριθμών – θα μπορούσε επίσης να ελέγχει λέξεις ή ένα άλλο πεδίο ορισμού) και ακολουθούμενο από το μέρος που διεξάγει τον έλεγχο.

Γενικά στον προγραμματισμό χρειάζεται συνεχής έλεγχος για ενδεχόμενα που είναι δεδομένα για τον ανθρώπινο νου, λόγω συνδυαστικής σκέψης, αλλά όχι για τον υπολογιστή που εκτελεί εντολές σε επίπεδο που του παρέχεται από τον χρήστη.

Κεφάλαιο 4

ΚΑΤΑΣΚΕΥΗ ΕΡΓΑΣΤΗΡΙΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ

4.1 Η ιδέα για την κατασκευή εργαστηρίων ηλεκτρονικών σε VB

Στη σημερινή εποχή οι γνώσεις σε θέματα ηλεκτρονικών είναι απαραίτητες. Σε όλους τους τομείς χρησιμοποιείται η αυτοματοποιημένη τεχνολογία στα πλαίσια της καλυτέρευσης της ποιότητας της ζωής. Εργαστήρια, βιοτεχνίες, ιδρύματα, νοικοκυριά, μέσα μεταφοράς μέχρι και γεωργία και κτηνοτροφία στηρίζονται πλέον ως επί το πλείστον σε αυτοματοποιημένα μηχανήματα. Όλα αυτές οι συσκευές που συναντάμε καθημερινά ελέγχονται από 'ηλεκτρονικούς εγκεφάλους', δηλαδή υπολογιστικούς μικροεπεξεργαστές, που δεν είναι άλλο από υπολογιστές προγραμματισμένους να εκτελούν συγκεκριμένες λειτουργίες.

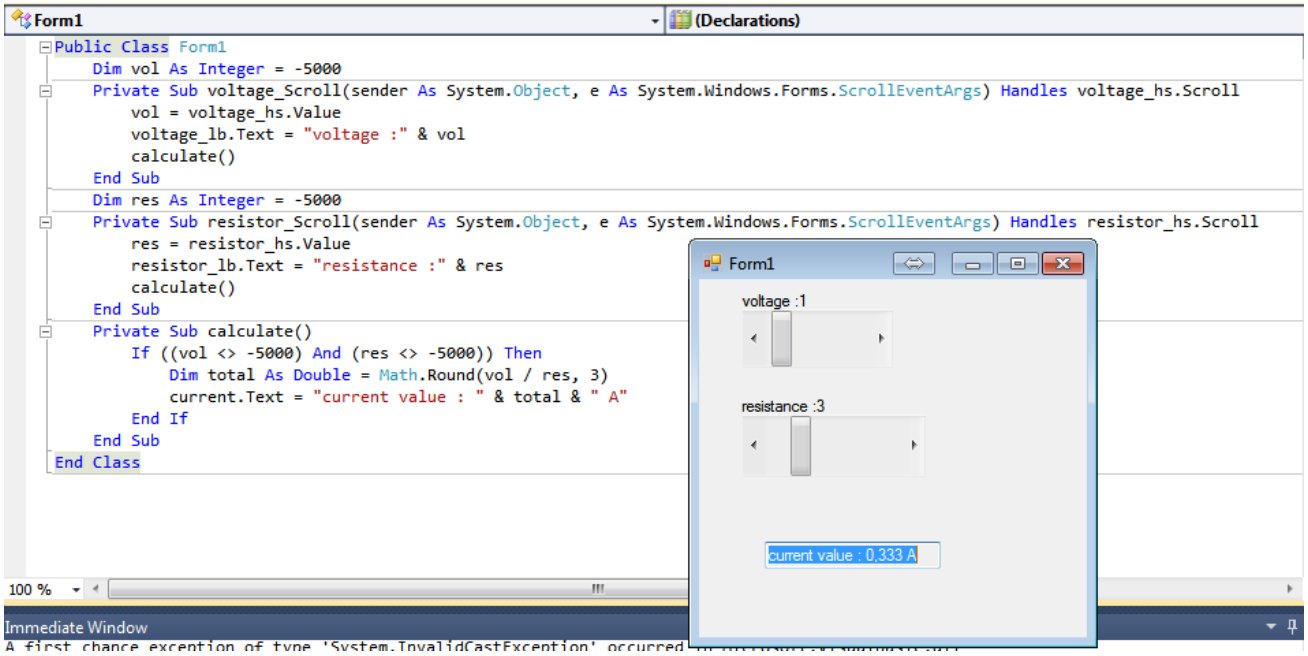
Υπάρχουν πολλές γλώσσες προγραμματισμού που συντελούν στην κατασκευή κάθε λογής μικροεπεξεργαστών: C, Fortran, Java, Python, PHP και φυσικά Visual Basic είναι ελάχιστες από τις πιο γνωστές που υπάρχουν, ενώ κάθε γλώσσα έχει και τις παράγωγές της.

Καθ' ότι οι δυνατότητες κατασκευής προγραμμάτων είναι απεριόριστες, η παρούσα πτυχιακή εργασία έγινε με θέμα την κατασκευή προγραμμάτων σχετικών με το καθ' αυτό μάθημα Ηλεκτρονικών της Ακαδημίας Εμπορικού Ναυτικού του τμήματος μηχανικών. Επίσης, χρησιμοποιήθηκαν στοιχεία των μαθημάτων ηλεκτροτεχνίας και ηλεκτρικών μηχανών, που σχετίζονται άμεσα, και η κεντρική ιδέα ήταν κατασκευή εφαρμογών που εξυπηρετούν τις ανάγκες των μαθημάτων αυτών.

Στη συνέχεια του κεφαλαίου επρόκειτο να παρατεθούν, να μελετηθούν, να εξηγηθούν ακόμα και να εξελιχθούν αυτές οι εργασίες με σκοπό την προσπάθεια χρησιμοποίησής τους από σπουδαστές της Σχολής.

4.2 Εφαρμογές υπολογισμού έντασης ρεύματος

Το πρώτο πρόγραμμα είναι εφαρμογή υπολογισμού έντασης ρεύματος με μεταβλητή τροφοδοσία (τάση) και μεταβλητή αντίσταση. Ένα σχετικά απλό πρόγραμμα ως γνωριμία με το εργαστήριο των ηλεκτρονικών.



Εικόνα 4.2.1 Εφαρμογή υπολογισμού ρεύματος με μεταβλητές τάση και αντίσταση

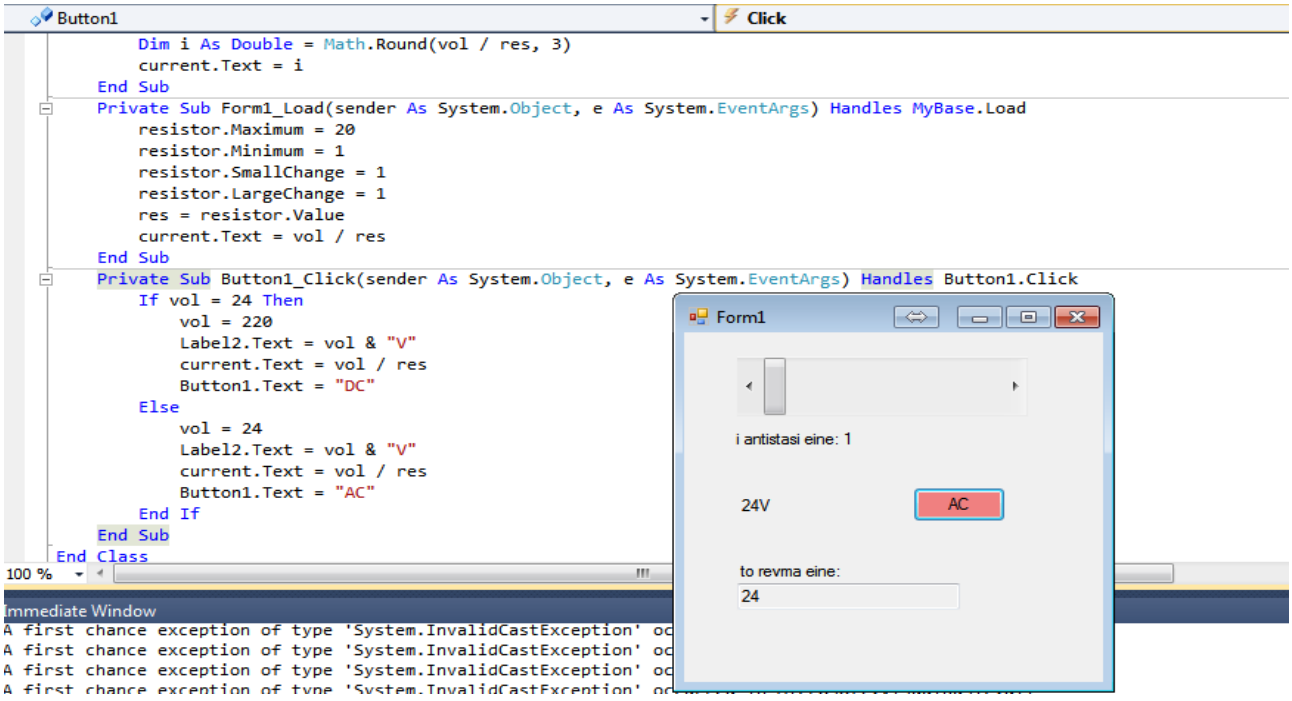
Όπως προαναφέρθηκε οι κατηγορίες ‘Private’ δημιουργούνται αυτόματα με διπλό click σε αντικείμενα ή συμβάντα. Ο κώδικάς μας είναι οι δηλώσεις των μεταβλητών vol, res και current, οι οποίες παρατηρούμε πως έχουν γίνει εκτός υποκλάσεων και ακριβώς πριν χρησιμοποιήσουμε το καθένα απ’ αυτά Είτε έτσι είτε να είχαν δηλωθεί όλα στην αρχή είναι ακριβώς το ίδιο πράγμα, αλλά αν είχαν δηλωθεί εντός των υποκλάσεων τότε θα μπορούσαμε να τα χρησιμοποιήσουμε μόνο εκεί μέσα ή να τα δηλώνουμε σε κάθε υποκλάση που τα καλούμε. Τα αντικείμενα που φέρουν την κατάληξη _hs είναι τα ονόματα από τα scrollbars, δηλαδή οι μπάρες μεταβολής των τιμών τάσης και αντίστασης με τα βελάκια. Τα όρια των μπαρών έχουν τεθεί στο form1.design στις επιλογές – ιδιότητες. Η δήλωση voltage_lb.text = “voltage :” & vol λέει στον υπολογιστή ότι το κείμενο (.text) της ταμπέλας τροφοδοσίας (voltage_lb – lb από το label) θα είναι voltage και vol .Το “voltage” παρατίθεται σε αυτάκια γιατί είναι μια παράμετρος που δεν χρειάζεται να την καλέσουμε ή να τη χρησιμοποιήσουμε και δεν την έχουμε δηλώσει στην κλάση μας, το vol είναι η τιμή της τάσης που της δίνεται από την πρώτη μπάρα (vol = voltage_hs.Value). Το Voltage_hs είναι το όνομα της μπάρας και η τιμή υποδεικνύεται από το .Value.

Το ίδιο ισχύει και για τις ονοματολογίες των άλλων 2 μεταβλητών. Κάτι άξιο προς παρατήρηση είναι η αρχικοποίηση των τιμών των μεταβλητών μας ως -5000. Αν δεν γίνει αρχικοποίηση τιμών ο compiler δίνει τις προκαθορισμένες από το σύστημα που είναι 0 για αριθμούς, False για Boolean (ψευδής ή αληθής δήλωση) και γενικά αντιλαμβάνονται από το σύστημα ως 'κενό' ή τίποτα'. Η αρχικοποίηση παραμέτρων με ακανόνιστες τιμές δεν είναι κάτι το σπάνιο. Δεν είναι ορατές προς το χρήστη της εφαρμογής και αποφεύγονται συγχύσεις του συστήματος με την τιμή 'κενό'.

Τέλος η πράξη vol /res συνοδεύεται από τη δήλωση Math.Round και ακολουθείται από το ,3. Το Math είναι μία δήλωση που περιλαμβάνει μαθηματικές εξισώσεις, όπως το Round, στρογγυλοποίηση, στο $3^{ο}$ δεκαδικό ψηφίο (,3).

Η εφαρμογή αυτή δίνει την τιμή της έντασης του ρεύματος στο πλαίσιο κάτω από τις μπάρες κάθε φορά που αλλάζουμε την τιμή της τάσης ή της αντίστασης.

Η επόμενη εφαρμογή είναι λίγο διαφορετική. Χρησιμοποιεί μεταβλητή αντίσταση, όπως η πρώτη, αλλά η τροφοδοσία αλλάζει με ένα κουμπί από συνεχή 24V σε εναλλασσόμενη 220V τάση.



Εικόνα 4.2.2 Εφαρμογή υπολογισμού ρεύματος με μεταβλητή αντίσταση σε AC ή DC

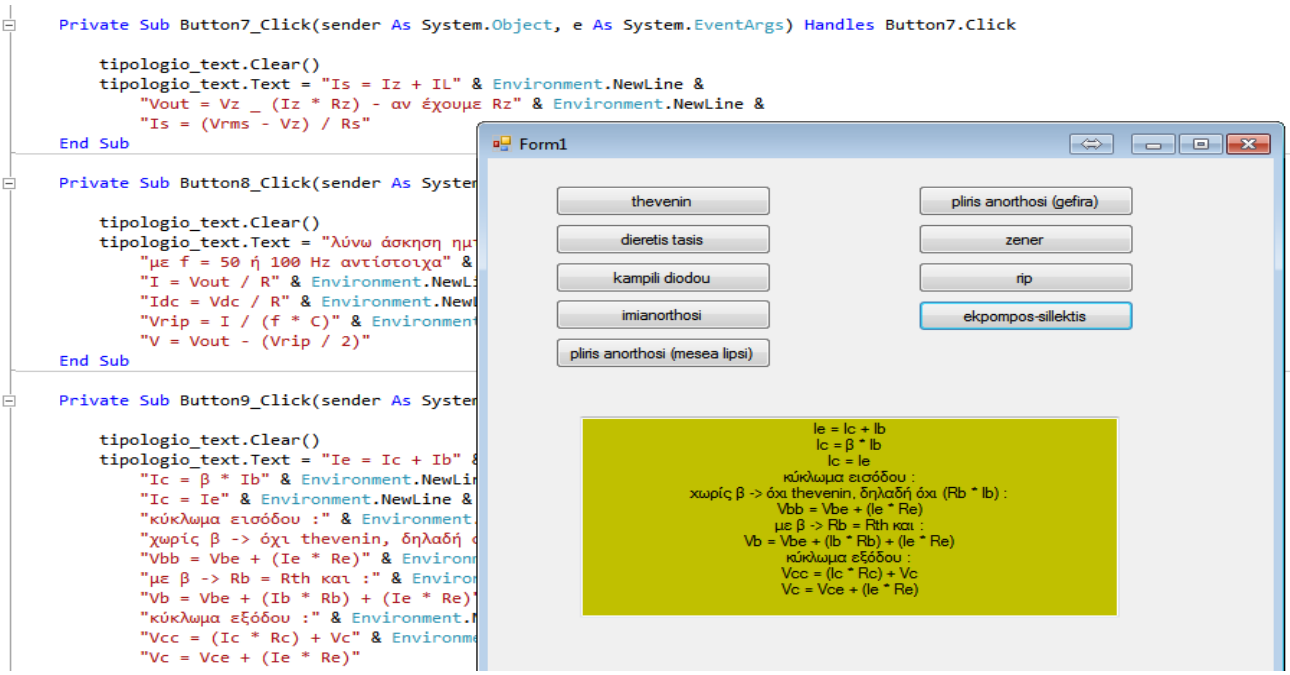
Στη δεύτερη εφαρμογή τα όρια της μπάρας έχουν δηλωθεί μέσα στον κώδικα και όχι στον πίνακα ιδιοτήτων της μπάρας και οι τιμές της τάσης είναι σταθερές στα 220V ή στα 24V και ελέγχονται από το πάτημα ενός κουμπιού.

Στην υποκλάση 'Private Sub Button1_Click …' δηλώνονται τα εξής:

AN η τιμή της τάσης είναι 24V TOTE με το πάτημα του κουμπιού η τάση να γίνεται 220V, η ταμπέλα που δείχνει την τάση (αριστερά απ' το κουμπί που στην εικόνα 4.2.2 λέει 24V) να παίρνει την τιμή της τάσης και τη δήλωση "V" (Label2.Text = vol & "V"), το κείμενο του ρεύματος να γίνεται vol / res (δηλαδή το textbox που βρίσκεται κάτω και γράφει 24 να πραγματοποιεί την πράξη τάση δια αντίσταση) και το κείμενο που αναγράφεται στο κουμπί να μετατρέπεται σε 'DC' (ο διακόπτης δείχνει την κατάσταση που θα επιφέρει το πάτημά του, οπότε εφ' όσον είμαστε σε AC το κουμπί πρέπει να δείχνει DC). Το Else στη συνέχεια δηλώνει την ίδια διαδικασία από AC – 220V σε DC – 24V.

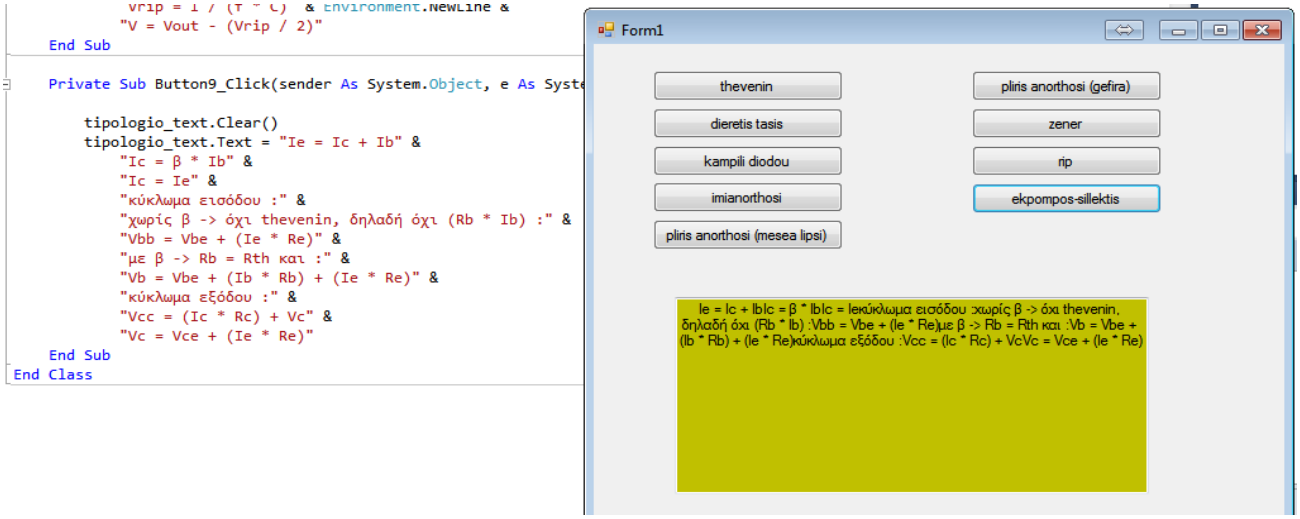
4.3 Εφαρμογή τυπολογίου μαθήματος ηλεκτρονικών

Μία διαφορετική εφαρμογή, χωρίς υπολογισμούς, που έχει συγκεντρωμένο το τυπολόγιο του μαθήματος των ηλεκτρονικών. Κάθε κουμπί αντιπροσωπεύει μια ενότητα του μαθήματος και με το πάτημα καθενός απ' αυτά, παρουσιάζεται και το αντίστοιχο τυπολόγιο.



Εικόνα 4.3.1 Εφαρμογή τυπολογίου μαθήματος ηλεκτρονικών

Σε κάθε υποκλάση πατήματος αντίστοιχου κουμπιού το πρώτο πράγμα που δηλώνεται είναι το σβήσιμο κειμένου (tipologio_text.clear()), δηλαδή ο καθαρισμός του χρωματισμένου μας textbox από προηγούμενα κείμενα και η εμφάνιση του αντίστοιχου καινούριου. Δεν έχουμε αρχικοποιημένες δηλώσεις μεταβλητών αφού κάθε φορά το κείμενό μας αλλάζει οπότε χρησιμοποιούμε κείμενο σε 'αυτάκια'. Το καινούριο στοιχείο σε αυτή την εφαρμογή είναι το Environment.NewLine. Το Environment είναι δήλωση που επενεργεί στο περιβάλλον του χρήστη και έχει και αυτή μια σειρά λειτουργιών. Προσοχή! Είναι ουσιαστική παρέμβαση! Είναι η εντολή που αλλάζει σειρά στο κείμενο, μεταφέρει δηλαδή την επόμενη εξίσωση στην επόμενη σειρά. Αν είχε παραληφθεί αυτή η εντολή το πρόγραμμα θα είχε αυτή τη μορφή:



Εικόνα 4.3.2 Παράλειψη στον κώδικα που οδηγεί σε σύγχυση

Δεν έχει σημασία πως φαίνεται ότι έχει γραφτεί ο κώδικας. Ο υπολογιστής χρειάζεται την εντολή για να καταλάβει ότι πρέπει να τα εμφανίσει σε διαφορετικές σειρές στο χρήστη.

4.4 Εφαρμογή γραφικής παράστασης

Η κατασκευή γραφικής παράστασης σε VB είναι αρκετά πιο πολύπλοκη διαδικασία. Από το toolbox της καρτέλας form1.vb (design) επιλέγουμε charts, που είναι μια λειτουργία που επιτρέπει την κατασκευή διαγραμμάτων όλων των ειδών και ο προγραμματισμός του είναι λίγο περίπλοκος όπως θα δούμε παρακάτω.

Form1.vb | Form1.vb [Design]

Form1 | calc_interval

```
Public Class Form1
    Dim resistance As Integer = 50
    Dim frequency As Integer = 25
    Dim interval As Double = 0
    Dim omega As Double = 0
    Dim tasi As Double = 0
    Dim reuma As Double = 0
    Dim pi As Double = 3.1415
    Dim time As Double = 0
    Dim x As Integer = 0
    Dim array(500) As Integer
    Private Sub calc_omega()
        omega = 2 * pi * frequency
    End Sub
    Private Sub calc_interval()
        interval = interval + (1 / (frequency * 250))
    End Sub
    Private Sub calc_tasi()
        tasi = 220 * Math.Sin(omega * interval)
    End Sub
    Private Sub calc_reuma()
        reuma = tasi / resistance
    End Sub
    Private Sub Form1_Load(sender As System.Object, e As System.EventArgs) Handles MyBase.Load
```

Form1.vb | Form1.vb [Design]

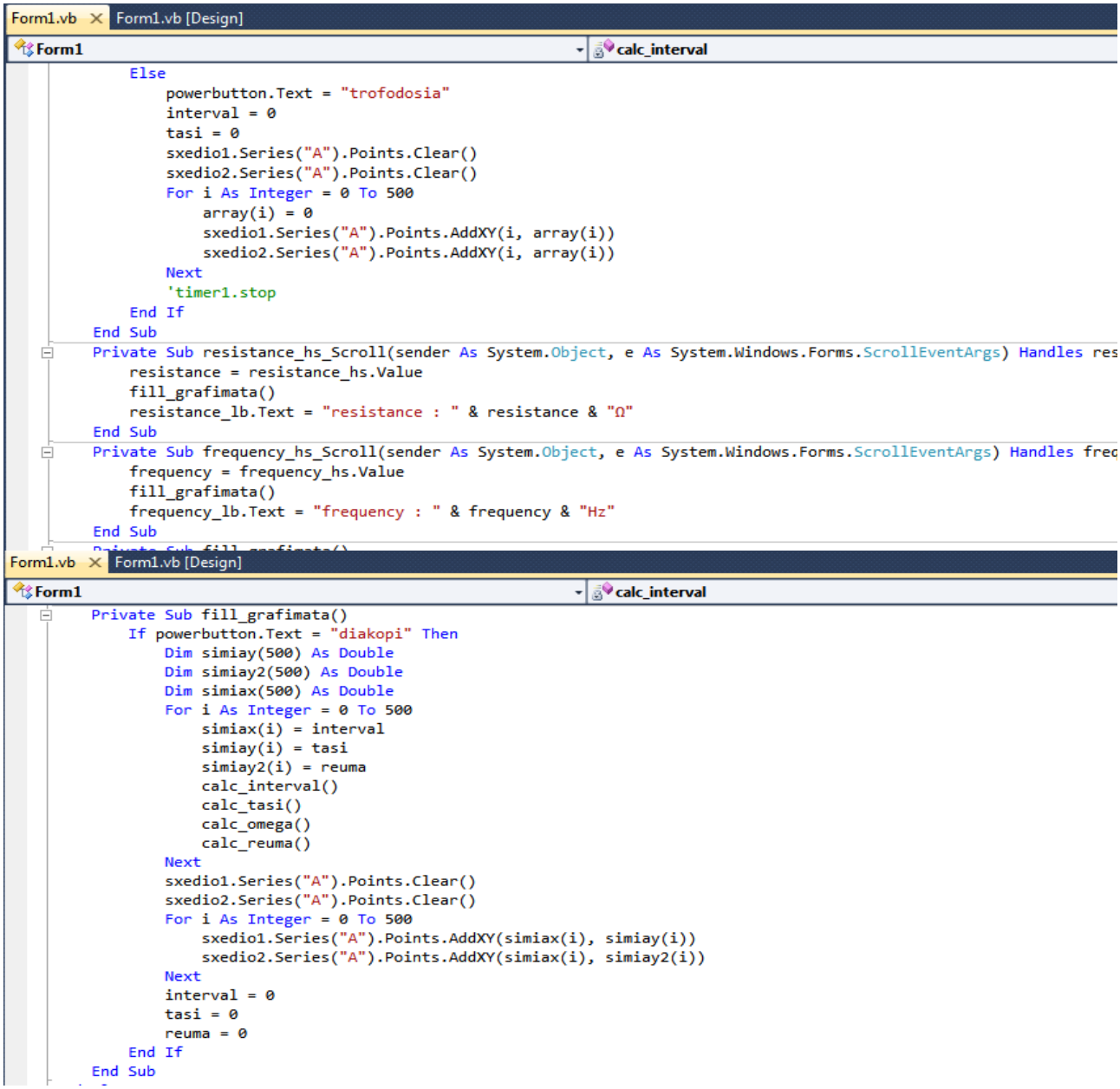
Form1 | calc_interval

```
    Private Sub calc_reuma()
        reuma = tasi / resistance
    End Sub
    Private Sub Form1_Load(sender As System.Object, e As System.EventArgs) Handles MyBase.Load
        calc_omega()
        sxedio1.Series.Clear()
        sxedio1.Series.Add("A")
        sxedio1.ChartAreas("ChartArea1").AxisX.Minimum = 0
        sxedio1.ChartAreas("ChartArea1").AxisX.Maximum = 0.04
        sxedio1.ChartAreas("ChartArea1").AxisX.Interval = 0.01
        sxedio1.ChartAreas("ChartArea1").AxisX.Title = "sec"
        sxedio1.ChartAreas("ChartArea1").AxisY.Minimum = -220
        sxedio1.ChartAreas("ChartArea1").AxisY.Maximum = 220
        sxedio1.ChartAreas("ChartArea1").AxisY.Interval = 55
        sxedio1.ChartAreas("ChartArea1").AxisY.Title = "volt"
        sxedio1.Series("A").ChartType = DataVisualization.Charting.SeriesChartType.Line
        For i As Integer = 0 To 500
            array(i) = 0
            sxedio1.Series("A").Points.AddXY(i, array(i))
        Next
        sxedio2.Series.Clear()
        sxedio2.Series.Add("A")
        sxedio2.ChartAreas("ChartArea1").AxisX.Minimum = 0
        sxedio2.ChartAreas("ChartArea1").AxisX.Maximum = 0.04
```

Form1.vb | Form1.vb [Design] | Step Into (F8)

Form1 | calc_interval

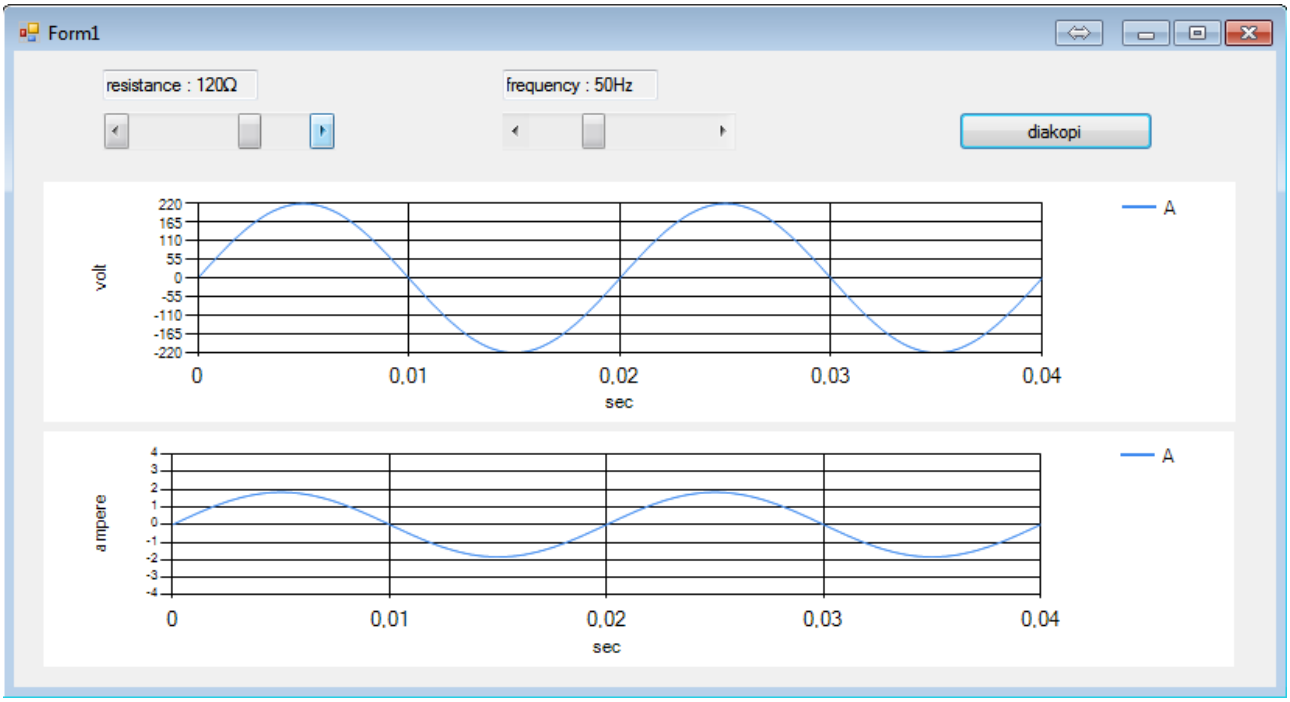
```
        Next
        sxedio2.Series.Clear()
        sxedio2.Series.Add("A")
        sxedio2.ChartAreas("ChartArea1").AxisX.Minimum = 0
        sxedio2.ChartAreas("ChartArea1").AxisX.Maximum = 0.04
        sxedio2.ChartAreas("ChartArea1").AxisX.Interval = 0.01
        sxedio2.ChartAreas("ChartArea1").AxisX.Title = "sec"
        sxedio2.ChartAreas("ChartArea1").AxisY.Minimum = -4
        sxedio2.ChartAreas("ChartArea1").AxisY.Maximum = 4
        sxedio2.ChartAreas("ChartArea1").AxisY.Interval = 1
        sxedio2.ChartAreas("ChartArea1").AxisY.Title = "ampere"
        sxedio2.Series("A").ChartType = DataVisualization.Charting.SeriesChartType.Line
        For i As Integer = 0 To 500
            array(i) = 0
            sxedio2.Series("A").Points.AddXY(i, array(i))
        Next
    End Sub
    Private Sub powerbutton_Click(sender As System.Object, e As System.EventArgs) Handles powerbutton.Click
        If powerbutton.Text = "trofodosia" Then
            powerbutton.Text = "diakopi"
            'timer1.interval = 100
            fill_grafimata()
        Else
            powerbutton.Text = "trofodosia"
```



Εικόνα 4.4.1 Η γραφή του κώδικα της γραφικής παράστασης

Οι μεταβλητές που χρησιμοποιούνται με σειρά όπως εμφανίζονται είναι αντίσταση και συχνότητα ως ακέραιοι, περίοδος, γωνιακή ταχύτητα, τάση, ρεύμα, το π και ο χρόνος ως δεκαδικοί, και τα x και y ως ακέραιοι (τυπικά το array δηλώνει ένα εύρος τιμών που ανάλογα πως θα χρησιμοποιηθούν στους υπολογισμούς θα παραχθούν και ο αντίστοιχος αριθμός συναρτήσεων). Στη συνέχεια δημιουργούνται υποκλάσεις υπολογισμών των μεγεθών μας και αμέσως μετά ορίζονται τα διαγράμματα με την εντολή sxedio1.Series, sxedio1.ChartAreas, sxedio2.Series και sxedio2.Chartareas. Η ονοματολογία πρέπει να γίνει ακριβώς όπως ορίζεται από τις ιδιότητες (properties) στο form.1 (design) και η ισότητα DataVisualization.Charting.SeriesChartType.Line δηλώνει τη μορφή του διαγράμματος (αν θα είναι γραφική παράσταση, πίτα, γράφημα κλπ). Έπειτα

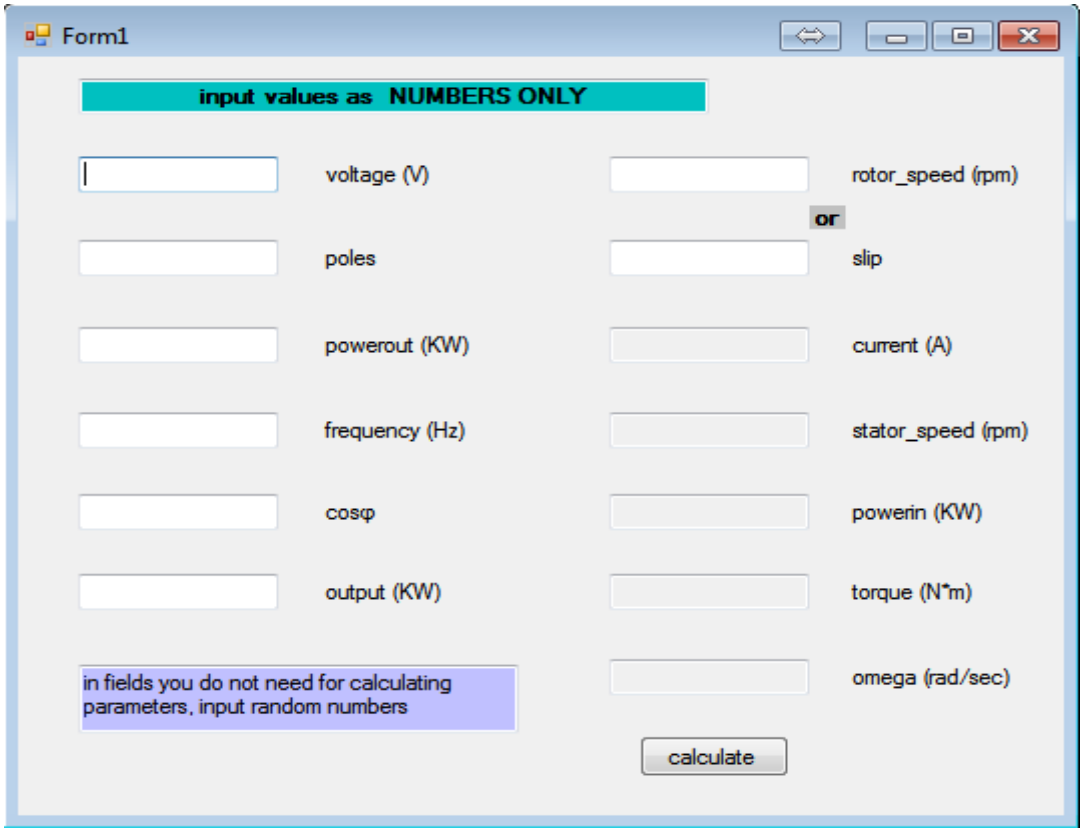
γράφονται οι εντολές για το τι θα συμβαίνει με το πάτημα του κουμπιού (αν γράφει τροφοδοσία με το πάτημα να ξεκινάει, ενώ αν γράφει διακοπή να σταματάει). Το κουμπί είναι ουσιαστικά ένα start – stop button. Κατόπιν πραγματοποιείται ο προγραμματισμός των μεταβλητών μπαρών συχνότητας και αντίστασης - αλλάζοντας τις τιμές τους αλλάζει και η μορφή των διαγραμμάτων όπως θα παρουσιαστούν στη συνέχεια. Τέλος, παρατηρώντας εντός ορισμένων υποκλάσεων εμφανίζεται η εντολή fill_grafimata(), οπότε φτιάχνουμε μια υποκλάση για να δηλώσουμε τι πραγματοποιείται με αυτή την εντολή. Είναι η εντολή που κατασκευάζει τη γραφική παράσταση με την εκκίνηση της τροφοδοσίας. Το προϊόν του προγράμματός μας είναι η εξής γραφική παράσταση.



Εικόνα 4.4.2 Εφαρμογή γραφικής παράστασης

Με τη μεταβολή της αντίστασης αλλάζει το πλάτος της έντασης του ρεύματος και με τη μεταβολή της συχνότητας αλλάζει η διάρκεια της περιόδου. Η εξίσωση της μεταβλητής interval σε σχέση με το εύρος του array επενεργεί στον αριθμό τον περιόδων που εμφανίζονται στο γράφημά μας.

4.5 Υπολογισμός στοιχείων ηλεκτρικών μηχανών



Εικόνα 4.5.1 Εφαρμογή υπολογισμού μεγεθών ηλεκτρομηχανής

Σε αυτό το πρόγραμμα εισάγουμε τα γνωστά στοιχεία της μηχανής στα 6 πρώτα πεδία και σε ένα από τα 2 επόμενα όπως υποδεικνύουν οι οδηγίες και πατώντας το κουμπί ‘υπολογισμός’ (calculate) υπολογίζονται τα υπόλοιπα. Είναι σχεδιασμένο να δίνει αποτελέσματα μόνο αν συμπληρωθούν 7 από τα 8 πρώτα πεδία με αριθμούς. Σε περίπτωση εισαγωγής παραμέτρων διαφόρων από αριθμούς ή κενών πεδίων τα πεδία των αποτελεσμάτων δίνουν αποτέλεσμα ‘ανεπαρκείς δεδομένα’. Είναι εφαρμογή ικανή να δώσει λύση σε προβλήματα υπολογισμού ηλεκτρικών μηχανών. Αν το πρόβλημα δίνει, για παράδειγμα, 3 στοιχεία και ζητάει 1, τότε θα πρέπει να εισαχθούν τυχαίοι αριθμοί στα υπόλοιπα πεδία για να δώσει αποτέλεσμα. Προγραμματίστηκε κατά αυτόν τον τρόπο λόγω πολυπλοκότητας χρήσης εξισώσεων. Θα ήταν εφικτό να ολοκληρωθεί το πρόγραμμα δίνοντας αποτελέσματα για οποιοδήποτε πεδίο, εισάγοντας δεδομένα σε οποιοδήποτε από τα 13 πεδία αλλά θα έπρεπε κατά τον προγραμματισμό του να γραφεί ένα τεράστιο σύνολο συσχετισμών των εξισώσεων.

Form1 (Declarations)

```
Public Class Form1
    Dim powerout As Double
    Dim torque As Double
    Dim frequency As Double
    Dim voltage As Double
    Dim stator_speed As Double
    Dim rotor_speed As Double
    Dim slip As Double
    Dim poles As Double
    Dim current As Double
    Dim cosϕ As Double
    Dim powerin As Double
    Dim output As Double
    Dim omega As Double
    Dim count As Integer = 0
    Private Sub Button1_Click(sender As System.Object, e As System.EventArgs) Handles Button1.Click
        count = 0
        If Not (Double.TryParse(TextBox1.Text, voltage)) Then
            voltage = 5000000
        Else
            count = count + 1
        End If
        If Not (Double.TryParse(TextBox2.Text, poles)) Then
            poles = 5000000
        Else
```

Form1 (Declarations)

```
            count = count + 1
        End If
        If Not (Double.TryParse(TextBox3.Text, powerout)) Then
            powerout = 5000000
        Else
            count = count + 1
        End If
        If Not (Double.TryParse(TextBox4.Text, frequency)) Then
            frequency = 5000000
        Else
            count = count + 1
        End If
        If Not (Double.TryParse(TextBox5.Text, cosϕ)) Then
            cosϕ = 5000000
        Else
            count = count + 1
        End If
        If Not (Double.TryParse(TextBox6.Text, output)) Then
            output = 5000000
        Else
            count = count + 1
        End If
        If Not (Double.TryParse(TextBox7.Text, rotor_speed)) Then
            rotor_speed = 5000000
```

Form1 (Declarations)

```
        If Not (Double.TryParse(TextBox7.Text, rotor_speed)) Then
            rotor_speed = 5000000
        Else
            count = count + 100
        End If
        If Not (Double.TryParse(TextBox8.Text, slip)) Then
            slip = 5000000
        Else
            count = count - 100
        End If
        If Not (Double.TryParse(TextBox9.Text, current)) Then
            current = 5000000
        End If
        If Not (Double.TryParse(TextBox10.Text, stator_speed)) Then
            stator_speed = 5000000
        End If
        If Not (Double.TryParse(TextBox11.Text, powerin)) Then
            powerin = 5000000
        End If
        If Not (Double.TryParse(TextBox12.Text, torque)) Then
            torque = 5000000
        End If
        If Not (Double.TryParse(TextBox13.Text, omega)) Then
            omega = 5000000
        End If
```

```
Form1                                                    (Declarations)
        If Not (Double.TryParse(TextBox13.Text, omega)) Then
            omega = 5000000
        End If
        findcorrect()
    End Sub
    Private Sub findcorrect()
        If count = 106 Then
            calcns()
            calcslip()
            calcomega()
            calctorque()
            calcpin()
            calccurrent()
            showresults()
        ElseIf count = -94 Then
            calcns()
            calcnm()
            calcomega()
            calctorque()
            calcpin()
            calccurrent()
            showresults()
        ElseIf count = 6 Then
        Else
            showerror()
Form1                                                    (Declarations)
        ElseIf count = 6 Then
        Else
            showerror()
        End If
    End Sub
    Private Sub calcns()
        stator_speed = Math.Round(120 * frequency / poles, 3)
    End Sub
    Private Sub calcslip()
        slip = Math.Round((stator_speed - rotor_speed) / stator_speed, 3)
    End Sub
    Private Sub calctorque()
        torque = Math.Round(powerout / omega, 3)
    End Sub
    Private Sub calcomega()
        omega = Math.Round(2 * Math.PI * frequency, 3)
    End Sub
    Private Sub calcpin()
        powerin = Math.Round(powerout / output, 3)
    End Sub
    Private Sub calccurrent()
        current = Math.Round(powerin / (Math.Sqrt(3) * voltage * cosϕ), 3)
    End Sub
    Private Sub calcnm()
        rotor_speed = Math.Round((1 - slip) * stator_speed, 3)
Form1                                                    (Declarations)
    End Sub
    Private Sub calcnm()
        rotor_speed = Math.Round((1 - slip) * stator_speed, 3)
    End Sub
    Private Sub showresults()
        TextBox7.Text = rotor_speed
        Friend WithEvents TextBox7 As System.Windows.Forms.TextBox
        TextBox10.Text = stator_speed
        TextBox11.Text = powerin
        TextBox12.Text = torque
        TextBox13.Text = omega
    End Sub
    Private Sub showerror()
        TextBox7.Text = "not enough data"
        TextBox8.Text = "not enough data"
        TextBox9.Text = "not enough data"
        TextBox10.Text = "not enough data"
        TextBox11.Text = "not enough data"
        TextBox12.Text = "not enough data"
        TextBox13.Text = "not enough data"
    End Sub
End Class
```

Εικόνα 4.5.2 Ο κώδικας της εφαρμογής υπολογισμού στοιχείων ηλεκτρομηχανής

Στο τελευταίο μας πρόγραμμα χρησιμοποιείται η εντολή ‘count’. Πρώτα η TryParse κάνει έλεγχο στις τιμές των μεταβλητών και η count μετράει γεγονότα για να χρησιμοποιήσουμε το άθροισμά τους που θα δείχνει σε ποια πεδία έχουν εισαχθεί τιμές και να δίνει τα αντίστοιχα αποτελέσματα στα υπόλοιπα. Οι showresults και showerrors υποκλάσεις καλούνται για να δώσουν τα αποτελέσματα στα κενά πεδία.

Με αυτή την τελευταία εφαρμογή ολοκληρώνεται και η παρουσίαση των εικονικών εργαστηρίων σε VB.

ΕΠΙΛΟΓΟΣ

Τα εικονικά αυτά εργαστήρια κατασκευάστηκαν ύστερα από μελέτη των αντίστοιχων μαθημάτων της Ακαδημίας Εμπορικού Ναυτικού, και σκοπός τους είναι η πρακτική χρήση από σπουδαστές ή καθηγητές στα πλαίσια της διδασκαλίας. Κάθε ενδιαφερόμενος μπορεί χρησιμοποιήσει τις εφαρμογές για διευκόλυνσή του στα εν λόγω μαθήματα ή ακόμα καλύτερα να τροποποιήσει κάποιον κώδικα ώστε να αναπτύξει μια εφαρμογή που να καλύπτει επακριβώς τις ανάγκες του. Για να γίνει αυτό θα χρειαστεί γνώσεις επί του προγραμματισμού σε VB, που αν δεν τις κατέχει η μελέτη της πτυχιακής αυτής εργασίας είναι ένας καλός οδηγός για ξεκίνημα.

Όπως έχει ήδη αναφερθεί δεν υπάρχει όριο στο τι μπορεί κανείς να κατασκευάσει μέσω του προγραμματισμού. Το μόνο που χρειάζεται είναι επαρκείς γνώσεις. Όμως δεν φτάνουν μόνο οι γνώσεις μιας γλώσσας προγραμματισμού, αλλά και η επιστημονική κατάρτιση στον τομέα όπου στοχεύει το εκάστοτε πρόγραμμα. Δεν είναι μόνο ο κώδικας που έχει συγκεκριμένο τρόπο δόμησης, είναι και οι αντίστοιχες εξισώσεις που χρησιμοποιούνται και είναι αναγκαίο να συνδυαστούν κατάλληλα μεταξύ τους.

Ο προγραμματισμός είναι μια συναρπαστική επιστήμη και θα ήταν πολύ χρήσιμο να διδασκόταν στα σχολεία ακόμα και σε πολύ μικρές ηλικίες, αντί για τον απλό ψευδοκώδικα που διδάσκουν σε μαθητές λυκείου σε τμήματα τεχνολογικής κατεύθυνσης. Η αυτοματοποιημένη τεχνολογία είναι το παρόν και το μέλλον και όσο περισσότερα γνωρίζουμε για αυτή τόσο καλύτερα μπορούμε να την εκμεταλλευτούμε.

Βιβλιογραφία

1. https://msdn.microsoft.com/en-us/library/sh9ywfdk.aspx
2. Root, Randal; Romero Sweeney, Mary (2006). A tester's guide to .NET programming
3. Plant, Robert T.; Murrell, Stephen (2007). An executive's guide to information
4. Plant, Robert T.; Murrell, Stephen (2007). An executive's guide to information technology
5. George, Mack. "History of Visual Basic". June 2002. George Mack, 3rd edition
6. Krill, Paul (2009-02-27). "Microsoft converging programming languages | Developer World"
7. "Main Procedure in Visual Basic". MSDN – Developer Center

Περιεχόμενα