ΕΚΠΑΙΔΕΥΤΙΚΟ ΕΓΧΕΙΡΙΔΙΟ

ΤΕΤΡΑΔΙΟ ΜΑΘΗΤΗ

Ανάδοχος Έργου

SDC A.E.

Κασταμονής 99α & Μακρυγιάννη
142 35 Ν. Ιωνία
τηλ. 210-2719100 fax 210-2718133
url : www.sdc.gr

Το παρόν εκπονήθηκε στο πλαίσιο

του Υποέργου 13 «Προσαρμογή Λογισμικού-Φάση ΙΙΙ»

της Πράξης «Επαγγελματικό λογισμικό στην ΤΕΕ: επιμόρφωση και εφαρμογή»

(Γ' ΚΠΣ, ΕΠΕΑΕΚ, Μέτρο 2.3, Ενέργεια 2.3.2)

που συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση/Ευρωπαϊκό Κοινωνικό Ταμείο

Φορέας Υλοποίησης και Τελικός Δικαιούχος

Υπουργείο Εθνικής Παιδείας και Θρησκευμάτων
Ειδική Υπηρεσία Εφαρμογής Προγραμμάτων ΚΠΣ

Φορέας Λειτουργίας

Υπουργείο Εθνικής Παιδείας και Θρησκευμάτων
Διεύθυνση Σπουδών Δευτεροβάθμιας Εκπαίδευσης-Τμήμα Β'

Επιστημονικός Τεχνικός Σύμβουλος

Ερευνητικό Ακαδημαϊκό Ινστιτούτο Τεχνολογίας Υπολογιστών

ΥΠΟΥΡΓΕΙΟ ΕΘΝΙΚΗΣ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ ΕΠΕΑΕΚ

ΕΥΡΩΠΑΪΚΗ ΕΝΩΣΗ
ΣΥΓΧΡΗΜΑΤΟΔΟΤΗΣΗ
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

Η ΠΑΙΔΕΙΑ ΣΤΗΝ ΚΟΡΥΦΗ
Επιχειρησιακό Πρόγραμμα
Εκπαίδευσης και Αρχικής
Επαγγελματικής Κατάρτισης

ΠΕΡΙΕΧΟΜΕΝΑ

1. Γνωριμία με το περιβάλλον εργασίας της Visual Basic 5

Δραστηριότητα 1 : Δημιουργία εφαρμογής κονσόλας 5

Δραστηριότητα 2 : Εφαρμογή Φορμών 11

2. Τα χειριστήρια της Visual Basic – Προγραμματισμός με συμβάντα 17

Δραστηριότητα 1 : Δημιουργία εφαρμογής φορμών 17

Δραστηριότητα 2: Παίζοντας με τις ιδιότητες 27

3. Η έννοια των σταθερών και των μεταβλητών στη Visual Basic 30

Δραστηριότητα 1 : Σταθερές στην Visual Basic 30

Δραστηριότητα 2 : Η εντολή καταχώρησης στην Visual Basic 31

Δραστηριότητα 3: Ορισμός μεταβλητών και τύποι μεταβλητών στη Visual Basic 32

Δραστηριότητα 4: Οι αριθμητικές μεταβλητές αναλυτικότερα 34

Δραστηριότητα 5 : Πράξεις με αριθμητικές μεταβλητές 35

Δραστηριότητα 6: Αλφαριθμητικές μεταβλητές 37

4. Οι συναρτήσεις και υπορουτίνες στη Visual Basic 39

Δραστηριότητα 1 : Δημιουργία μιας υπορουτίνας 39

Δραστηριότητα 2: Υπορουτίνα με παραμέτρους. 42

Δραστηριότητα 3 : Υπορουτίνες που επιστρέφουν τιμές (συναρτήσεις). 44

Δραστηριότητα 4 : Υπορουτίνες με περισσότερες παραμέτρους. 46

Δραστηριότητα 5 : Κλήση υπορουτίνας από υπορουτίνα. 47

5. Εισαγωγή στα αντικείμενα στη Visual Basic 48

Δραστηριότητα 1 : Το αντικείμενο console 48

Δραστηριότητα 2: Το αντικείμενο Math 51

Δραστηριότητα 3: Το String σαν αντικείμενο. 52

Δραστηριότητα 4: Η κλάση Date. 54

6. Λογικές συναρτήσεις και παραστάσεις η ροή του προγράμματος 56

Δραστηριότητα 1 : η εντολή If .. then ... else 56

Δραστηριότητα 2: nested if 58

Δραστηριότητα 3: Οι εντολές AND , OR, NOT 59

Δραστηριότητα 4: Το σύνθετο σχήμα If…Then…ElseIf 63

Δραστηριότητα 5: Δομή Select Case 64

7. Πίνακες και Επαναληπτικές Δομές (3ωρες) 66

Δραστηριότητα 1 : Η δομή repeat 66

Δραστηριότητα 2 : Η δομή while. 70

Δραστηριότητα 3 : Η δομή for .. next 72

Δραστηριότητα 4 : Οι πίνακες στη Visual Basic 74

8. Εκσφαλμάτωση 77

Δραστηριότητα 1 : Συντακτικά λάθη. 77

Δραστηριότητα 2 : Επιλογές που επηρεάζουν τα συντακτικά λάθη. (Option Explicit) 78

Δραστηριότητα 3 : Επιλογές που επηρεάζουν τα συντακτικά λάθη. (Option Strict). 79

Δραστηριότητα 4 : Οι εντολές try … catch 80

Δραστηριότητα 5 : Λογικά λάθη 1. 82

Δραστηριότητα 6 : Λογικά λάθη 2 83

Φύλλο εργασίας

1. Γνωριμία με το περιβάλλον εργασίας της Visual Basic

Όνομα:

Τάξη:

Διάρκεια: 4 διδακτικές ώρες.

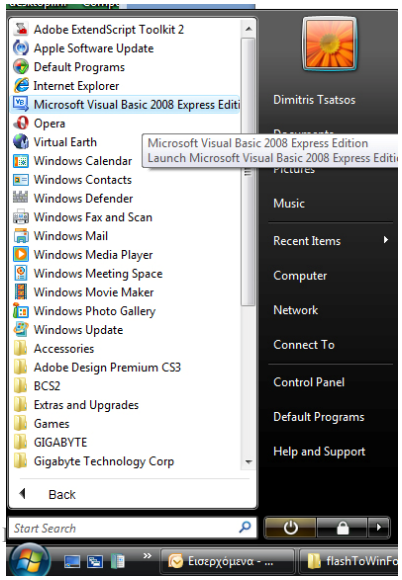
Διδακτικοί στόχοι:

Με το συγκεκριμένο φύλλο εργασίας

- Θα μάθεις πώς να δημιουργείς ένα νέο έργο με το περιβάλλον εργασίας της Visual Basic
- Θα γνωρίσεις τα παράθυρα που αποτελούν το περιβάλλον εργασίας της Visual Basic
- Θα μάθεις πώς θα μεταγλωττιστεί (compile) ένα έργο σε Visual Basic, ώστε να δημιουργήσεις μια εκτελέσιμη εφαρμογή.
- Θα γνωρίσεις τις «εφαρμογές κονσόλας» και τις «εφαρμογές φορμών» της Visual Basic

Δραστηριότητα 1 : Δημιουργία εφαρμογής κονσόλας

Στάδιο 1 : Δημιουργώντας ένα νέο Έργο (project).

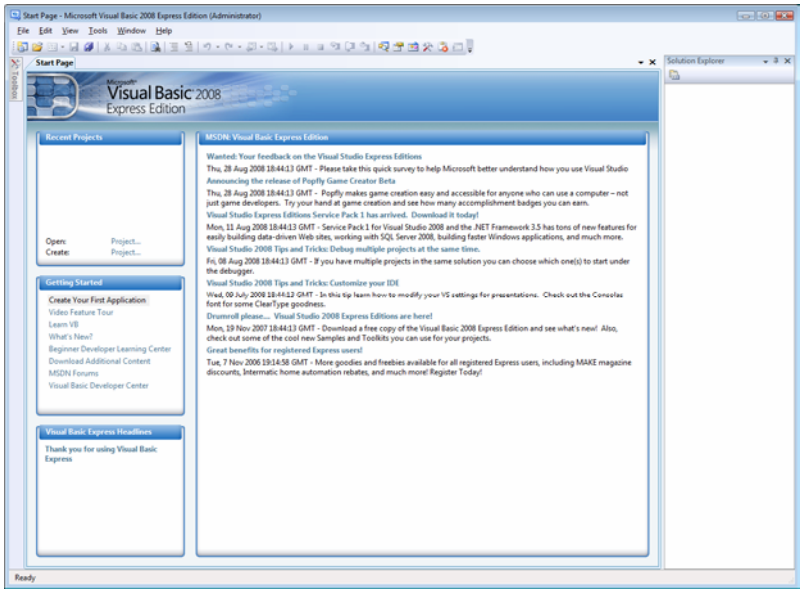


Για να ξεκινήσουμε το περιβάλλον της visual basic πρέπει να πάμε στο μενού έναρξη (start) των windows και να εντοπίσουμε τη συντόμευση «Microsoft Visual Basic 2008 Express Edition» ή την αντίστοιχη έκδοση που έχουμε στον υπολογιστή μας.

Κάνοντας κλικ στο εικονίδιο ξεκινάει το περιβάλλον εργασίας της γλώσσας.

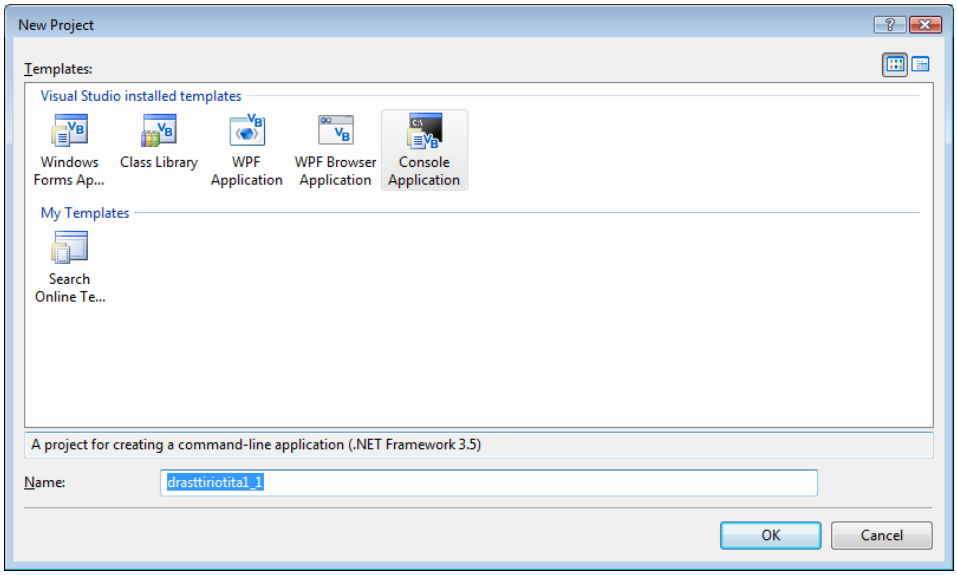
Με τη Visual Basic μπορούμε να δημιουργήσουμε εφαρμογές διαφόρων ειδών. Εδώ θα ασχοληθούμε με δυο τύπους εφαρμογών: τις «Εφαρμογές Κονσόλας» (Console application) και τις «Εφαρμογές Φορμών» (Windows Forms application). Για να δημιουργήσουμε μια εφαρμογή οποιουδήποτε τύπου με την Visual Basic πρέπει πρώτα να δημιουργήσουμε ένα «Έργο» (Project).

Η πρώτη οθόνη του περιβάλλοντος εργασίας της Visual Basic μας καλεί να δημιουργήσουμε ένα νέο έργο ή να ανοίξουμε ένα υπάρχον.



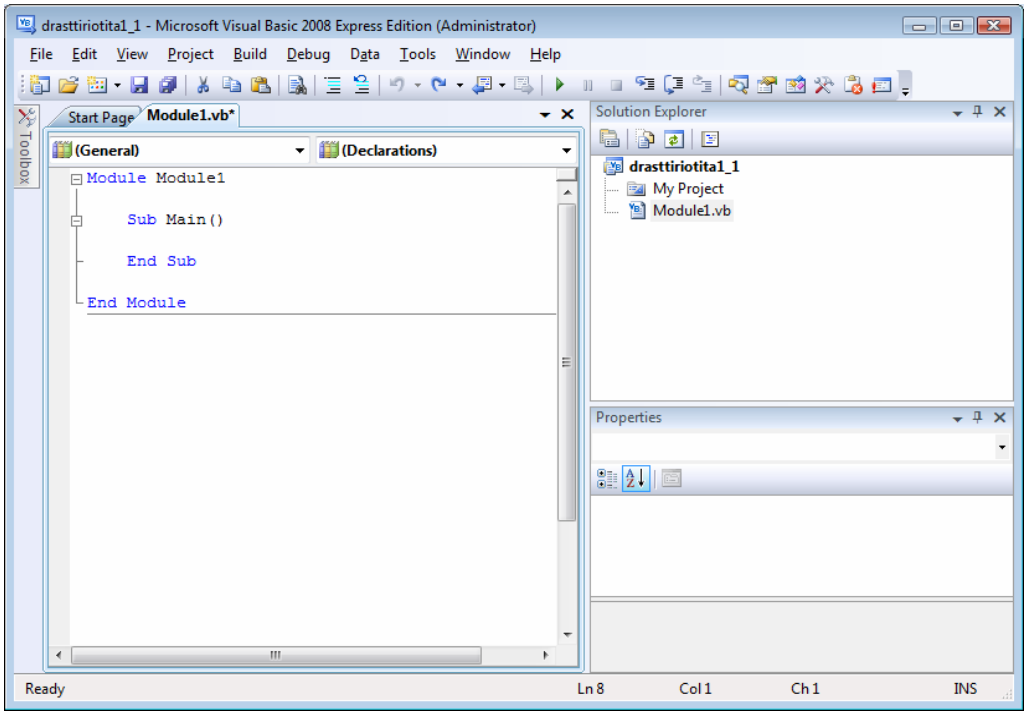
Πατήστε «**Create Project**» (Δημιουργία Έργου) στην αρχική οθόνη. Θα σας ανοίξει η οθόνη επιλογής. Σε αυτή την οθόνη επιλέγουμε τι τύπο εφαρμογής θέλουμε να φτιάξουμε.

Επιλέξτε «**Console application**» για να δημιουργήστε μια εφαρμογή κονσόλας και ονομάστε την drasttiriotita1_1



Στάδιο 2: Εξερεύνηση του περιβάλλοντος έργου (console application)

Το περιβάλλον εργασίας της Visual Basic δημιουργεί ένα νέο έργο, από το οποίο μπορούμε να δημιουργήσουμε (γράφοντας κώδικα) μια εφαρμογή κονσόλας.



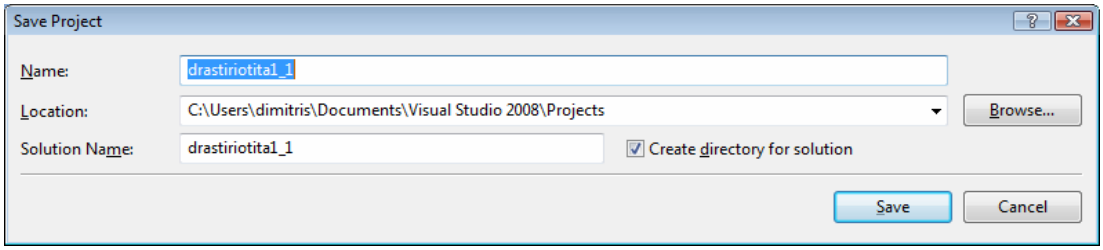
Το περιβάλλον εργασίας για το έργο μας είναι μοιρασμένο σε παράθυρα. Βλέπουμε αριστερά το παράθυρο κώδικα (Module1.vb), δεξιά πάνω το παράθυρο έργου (Solution Explorer) και δεξιά κάτω το παράθυρο ιδιοτήτων (Properties).

Ένα έργο στη Visual Basic αποτελεί τμήμα μιας «Λύσης» (Solution). Μέσα σε μια λύση μπορούν να ομαδοποιηθούν πολλά έργα. Οι εφαρμογές δημιουργούνται από την μεταγλώττιση (compilation) της Λύσης (Solution). Εμείς θα ασχοληθούμε με λύσεις που περιέχουν ένα έργο. Στο παράθυρο έργου βλέπουμε τα αρχεία που περιέχονται στη «Λύση» που δουλεύουμε.

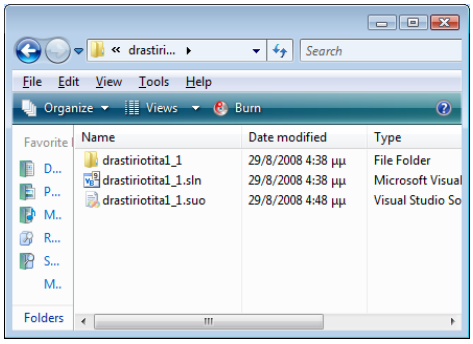
Επιλέξτε το αρχείο Module1.vb από το παράθυρο έργου και παρατηρήστε το παράθυρο ιδιοτήτων.

Στάδιο 3. Αποθήκευση και ανάκτηση του έργου

Πατήστε από το μενού «**File**» που βρίσκεται στην μπάρα την επιλογή «**Save all**». Στο παράθυρο που εμφανίζεται επιλέξτε τον φάκελο μέσα στον οποίο θα σώσετε τη δουλειά σας και πατήστε «**Αποθήκευση**» (Save).



Κλείστε το περιβάλλον εργασίας της Visual Basic. Χρησιμοποιώντας τον Windows Explorer βρείτε το φάκελο που σώσατε τη δουλειά σας.



Δείτε τα αρχεία που έχουν δημιουργηθεί. Υπάρχει το αρχείο drastiriotita1_1.sln στο πρώτο επίπεδο και ένας φάκελος με όνομα drastiriotita1_1 που περιέχει τα υπόλοιπα αρχεία που δημιούργησε το περιβάλλον εργασίας της visual basic.

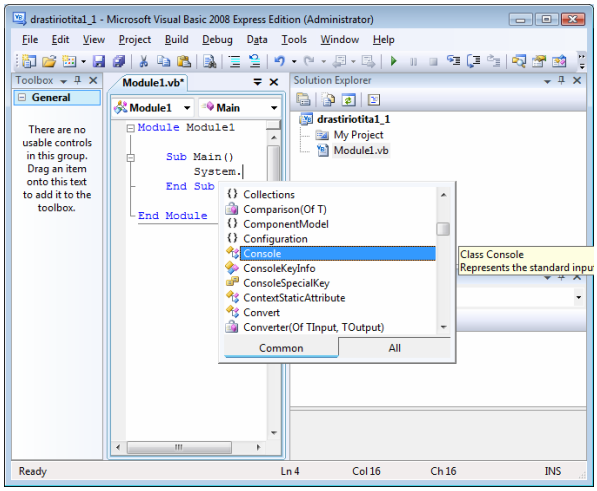
Για να ξανανοίξουμε το έργο που δημιουργήσαμε και να συνεχίσουμε να δουλεύουμε, μπορούμε να ανοίξουμε το αρχείο drastiriotita1_1.sln (κάνοντας διπλό κλικ επάνω του). Επίσης, μπορούμε να ανοίξουμε το περιβάλλον εργασίας της Visual basic και να πατήσουμε την επιλογή Open Project· από το παράθυρο διαλόγου που θα ανοίξει μπορούμε να εντοπίσουμε το drastiriotita1_1.sln και να το ανοίξουμε.

Στάδιο 4. Μεταγλώττιση(compilation)

Ανοίξτε ξανά το έργο που φτιάξατε. Από το παράθυρο έργου κάντε διπλό κλικ στο module1.vb. Με αυτή την ενέργεια ανοίγετε το παράθυρο κώδικα.

Ας γράψουμε τώρα το πρώτο μας πρόγραμμα.

Στο παράθυρο κώδικα μετά την γραμμή «*Sub Main()*» και πριν την γραμμή «*End Sub*» θα πληκτρολογήσουμε την εντολή «*System.Console.WriteLine("Hello world")*»



Πληκτρολογήστε:

System.

Το περιβάλλον εργασίας της Visual Basic εμφανίζει ένα drop down menu με όλες τις ιδιότητες, μεθόδους, γεγονότα κλπ που περικλείει το αντικείμενο «System».

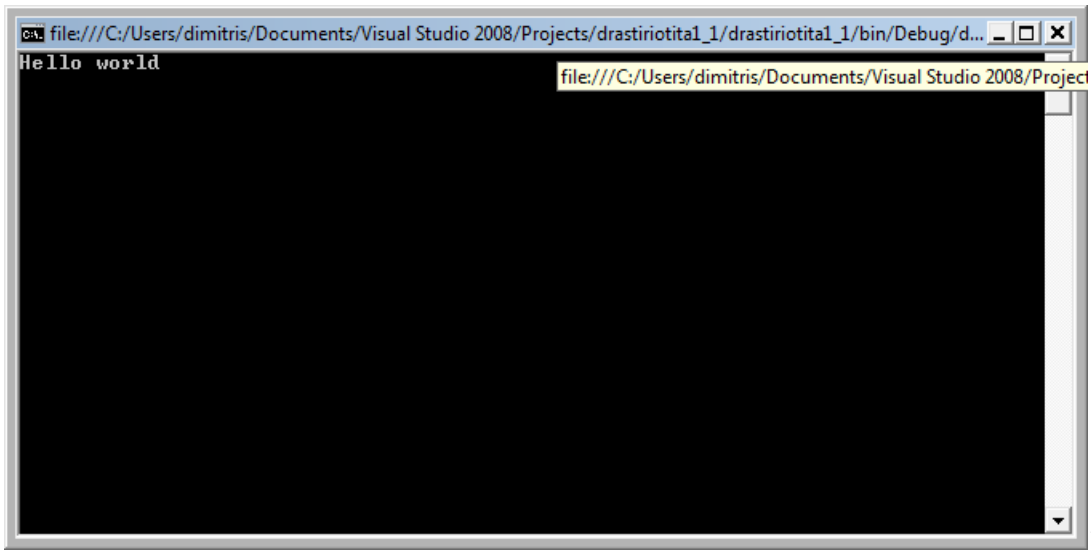
Η Visual Basic είναι μια αντικειμενοστραφής (Object oriented) γλώσσα προγραμματισμού. Όλες οι δυνατότητες που έχει είναι ομαδοποιημένες μέσα σε κλάσεις που τις παρέχει το «.Net Framework Class Library (FCL)». Η κλάση System είναι από τις σημαντικότερες. Η σχέση μεταξύ της κλάσης και των «παιδιών» της αναπαρίσταται με την «.». Πληκτρολογώντας λοιπόν System. Εμφανίζονται όλα τα «παιδιά» της κλάσης αυτής.

Συνεχίστε πληκτρολογώντας την εντολή «*System.Console.WriteLine(*». Η «WriteLine» είναι μια μέθοδος (method) που ανήκει στην κλάση «Console» που με τη σειρά της ανήκει στην κλάση «System». Η εντολή «WriteLine» εμφανίζει στην κονσόλα μια συμβολοσειρά (String). Οι μέθοδοι της Visual Basic συντάσσονται με παρενθέσεις στο τέλος. Μέσα στις παρενθέσεις βάζουμε τις παραμέτρους (arguments) της μεθόδου. Βάλτε την παράμετρο «“Hello world”» στην μέθοδο WriteLine και κλείστε τις παρενθέσεις.

Αλλάξτε γραμμή και συνεχίστε πληκτρολογώντας την επόμενη εντολή: «*System.Console.ReadLine()*». Η μέθοδος «ReadLine» σταματάει την εκτέλεση του προγράμματος και περιμένει να πληκτρολογήσουμε κάτι στην κονσόλα. Θα

καταλάβει πως τελειώσαμε την πληκτρολόγηση όταν πατήσουμε το πλήκτρο «Enter» και θα συνεχίσει την εκτέλεση του προγράμματος.

Τώρα ήρθε η ώρα να εκτελέσουμε το πρόγραμμα που γράψαμε. Από το μενού debug επιλέξτε «Start Debugging». Θα εμφανιστεί οθόνη κονσόλας που βλέπουμε παρακάτω.



Τι πρέπει να κάνουμε για να ολοκληρωθεί η εκτέλεση του προγράμματος; (Θυμηθείτε τι κάνει η μέθοδος ReadLine).

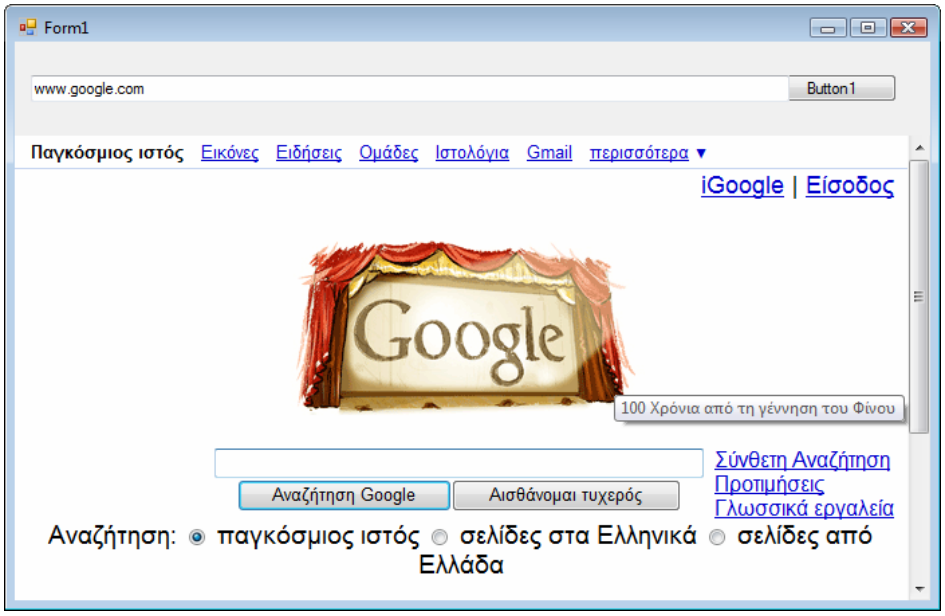
Κλείστε το περιβάλλον εργασίας της Visual Basic. Ερευνήστε στο φάκελο που σώσατε το έργο drastiriotita1_1. Ποια νέα αρχεία έχουν δημιουργηθεί.

Βρείτε μέσα στον φάκελο «bin/debug» το αρχείο drastiriotita1_1.exe και τρέξτε το κάνοντας διπλό κλικ πάνω του.

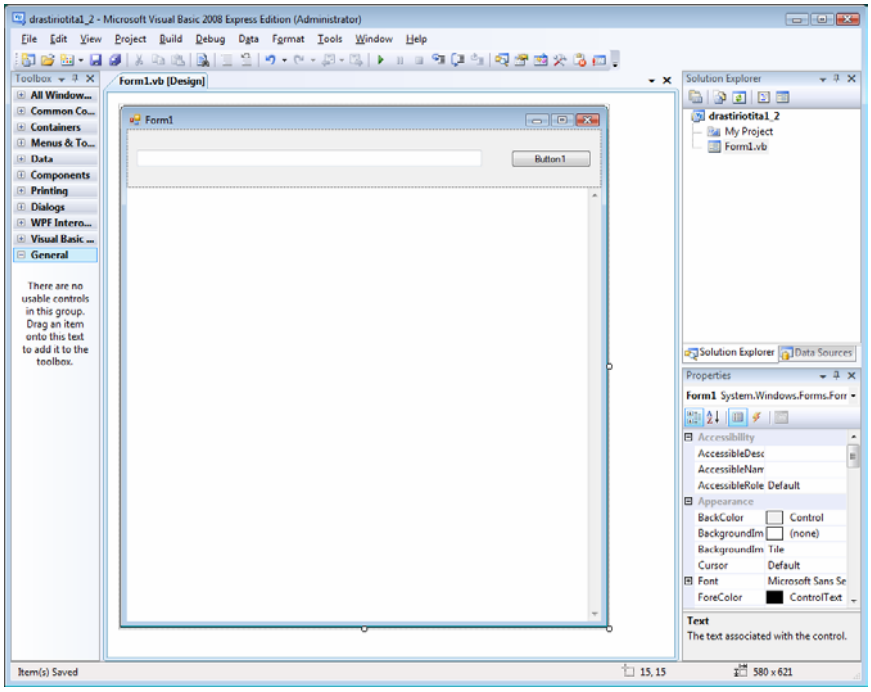
Δραστηριότητα 2 : Εφαρμογή Φορμών

Στάδιο 1 - Γνωριμία με την εφαρμογή φορμών

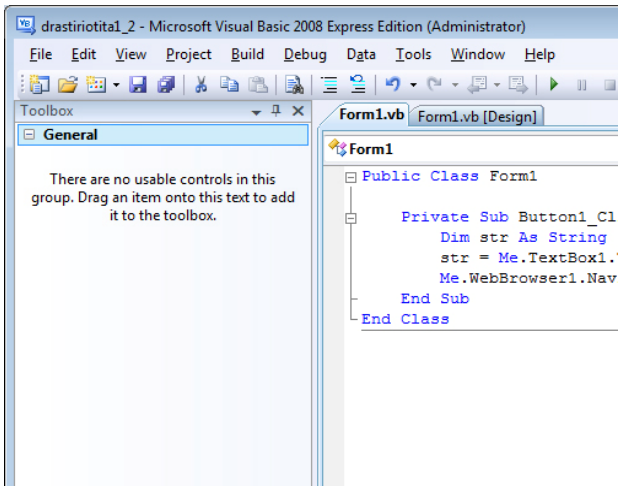
Ανοίξτε το έργο drastiriotita1_2.sln από τον φάκελο βοηθητικού υλικού «drastiriotita1_2» . Πατήστε από το μενού «**Debug->Start debugging**». Στο πεδίο στο πάνω μέρος της φόρμας πληκτρολογήστε «www.google.com» ή οποιοδήποτε άλλο site και μετά πατήστε το κουμπί «Button 1».



Το πρόγραμμα αυτό προσομοιάζει έναν web browser. Αν έχετε σύνδεση στο Internet θα εμφανιστεί η σελίδα που πληκτρολογήσατε. Κλείστε το παράθυρο της φόρμας για να τερματίσετε την εφαρμογή.



Στο περιβάλλον εργασίας της Visual Basic αυτή τη φορά δεν βλέπουμε το παράθυρο κώδικα, αλλά το παράθυρο σχεδιασμού της φόρμας. Για να δούμε τον κώδικα του προγράμματος πατάμε δεξί κλικ πάνω σε οποιοδήποτε σημείο της φόρμας και επιλέγουμε «**View Code**».

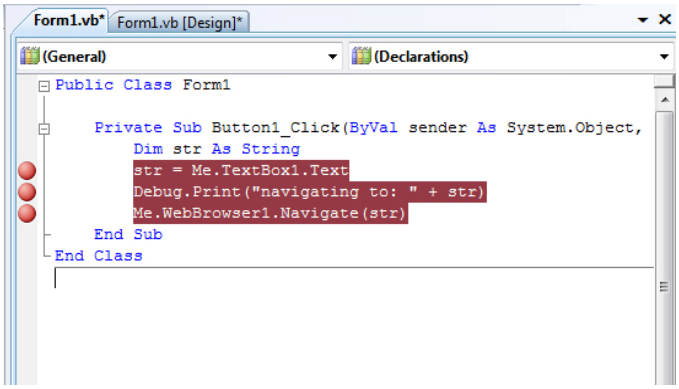


Εμφανίζεται το παράθυρο κώδικα σε ένα νέο TAB δίπλα στο παράθυρο σχεδιασμού. Στα αριστερά του περιβάλλοντος εργασίας της Visual Basic βρίσκεται το παράθυρο της εργαλειοθήκης. Μετακινηθείτε μεταξύ των δυο TAB (κώδικα και σχεδιασμού) της φόρμας Form1.vb. Τι παρατηρείτε στο παράθυρο της εργαλειοθήκης;

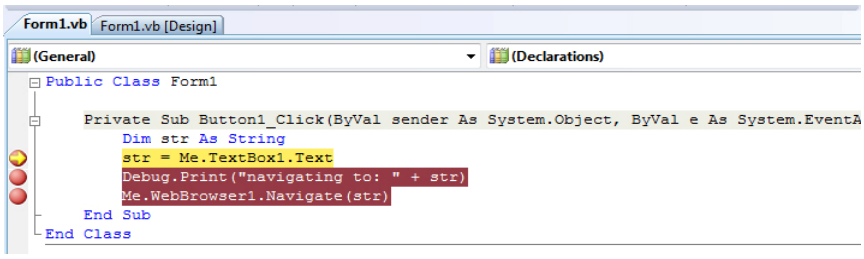
Πηγαίνετε στο παράθυρο σχεδιασμού της φόρμας και με το ποντίκι επιλέξτε τα διάφορα αντικείμενα που βρίσκονται εκεί (button, textbox, webbrowser). Ταυτόχρονα προσέξτε τις αλλαγές που γίνονται στο παράθυρο ιδιοτήτων. Τι παρατηρείτε;

Στάδιο 2 - breakpoints και παράθυρα άμεσης εκτέλεσης και τοπικών μεταβλητών.

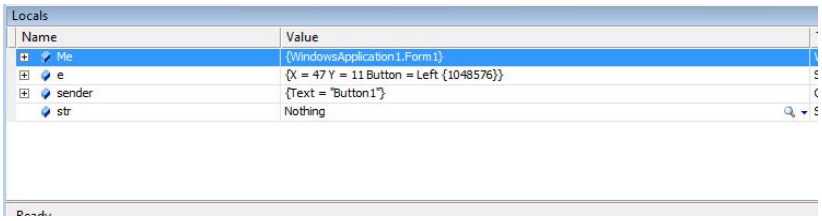
Κατά τη διάρκεια της δημιουργίας μια εφαρμογή ο προγραμματιστής χρειάζεται πολλές φορές να σταματήσει την εκτέλεση του προγράμματος και να ελέγξει τις τιμές που παίρνουν οι διάφορες μεταβλητές. Για να σταματήσουμε την εκτέλεση του προγράμματος πριν από μια εντολή βάζουμε breakpoints πατώντας αριστερά από την γραμμή εντολής που θέλουμε (ή επιλέγουμε τη γραμμή που θέλουμε και πατάμε στο μενού «**Debug->Toggle Breakpoints**»). Τοποθετήστε breakpoints όπως στο παρακάτω σχήμα και τρέξτε την εφαρμογή. Αντί για «**Debug->Start debugging**» μπορείτε να πατήσετε το πλήκτρο **F5**.



Πληκτρολογήστε «www.google.gr» στο πεδίο στο πάνω μέρος της φόρμας και πατήστε το «Button 1».



Η εκτέλεση του προγράμματος σταματά πριν από την εντολή που βρίσκεται το πρώτο breakpoint. Από το μενού «**Debug**» επιλέξτε «**Windows->Locals**». Ένα νέο παράθυρο ανοίγει στο κάτω μέρος της οθόνης.



Εδώ βλέπουμε τις τοπικές μεταβλητές και τις τιμές τους κατά τη διάρκεια εκτέλεσης του προγράμματος. Πατήστε **F11** για να εκτελεστεί η εντολή «*str = Me.TextBox1.Text*».

Τι άλλαξε στο παράθυρο Τοπικών μεταβλητών;

Από το μενού «**Debug**» επιλέξτε «**Windows**->**Immediate**» βλέπουμε ένα νέο παράθυρο στο περιβάλλον εργασίας της Visual Basic – Immediate Window-(Παράθυρο άμεσης εκτέλεσης). Πατήστε **F11** για να εκτελεστεί η εντολή «*Debug.Print("navigating to: " + str)*».

Τι βλέπετε στο παράθυρο άμεσης εκτέλεσης;

_____

_____

_____

Πατήστε δεξί κλικ στο Παράθυρο άμεσης εκτέλεσης και επιλέξτε «**Clear all**». Αφαιρέστε τα breakpoints από το παράθυρο κώδικα, πατήστε **F5** για να συνεχιστεί η εκτέλεση του προγράμματος. Στη φόρμα πατήστε «Button 1». Παρατηρήστε ξανά το παράθυρο άμεσης εκτέλεσης.

Τι πιστεύετε πως κάνει η εντολή Debug.Print.

_____

_____

_____

_____

_____

_____

Τερματίστε την εφαρμογή.

Πηγαίνετε στο Παράθυρο άμεσης εκτέλεσης και πληκτρολογήστε εκεί «*Debug.Print(«Hello world»)*» και μετά την εντολή «*Debug.Print(1+1)*». Η εντολή που επιστρέφει την τετραγωνική ρίζα ενός αριθμού X είναι η Math.Sqrt(X).

Τι πρέπει να γράψω στο Παράθυρο άμεσης εκτέλεσης ώστε να τυπωθεί η τετραγωνική ρίζα του 2;

_____

_____

Φύλλο εργασίας

2. Τα χειριστήρια της Visual Basic – Προγραμματισμός με συμβάντα

Όνομα:

Τάξη:

Διάρκεια: 3 διδακτικές ώρες.

Διδακτικοί στόχοι:

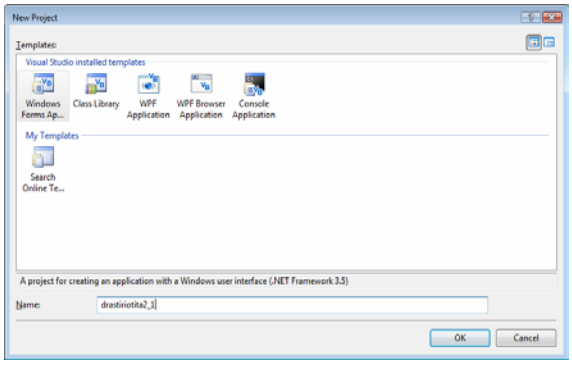
Με το συγκεκριμένο φύλλο εργασίας:

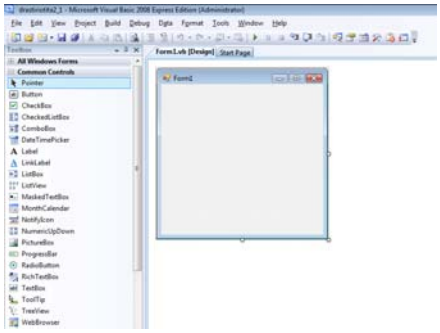
- Θα μάθεις να δημιουργείς εφαρμογές φορμών
- Θα μάθεις να τοποθετείς κουμπιά πάνω στις φόρμες
- Θα μάθεις να χειρίζεσαι τα συμβάντα γράφοντας κώδικα
- Θα μάθεις να τερματίζεις την εφαρμογή σου με κώδικα

Δραστηριότητα 1 : Δημιουργία εφαρμογής φορμών

Στάδιο 1 : Δημιουργία μιας φόρμας με αντικείμενα.

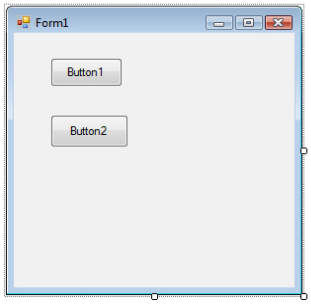
Ξεκινήστε το περιβάλλον της Visual Basic και δημιουργήστε ένα νέο έργο. Το έργο αυτό θα είναι εφαρμογή φορμών. Αποθηκεύστε το όπως μάθατε στο προηγούμενο φύλλο εργασίας, με το όνομα drastiriotita2_1.





Το περιβάλλον εργασίας της Visual Basic δημιουργεί ένα νέο έργο (project) που περιέχει μια φόρμα. Το κεντρικό παράθυρο στο περιβάλλον είναι το παράθυρο σχεδίασης της φόρμας. Όπως έχει αναφερθεί σε προηγούμενο μάθημα, το παράθυρο της εργαλειοθήκης (toolbox) είναι ενεργό όταν είναι επιλεγμένο το παράθυρο σχεδιασμού της φόρμας. Πάνω στη φόρμα μπορούμε να τοποθετήσουμε αντικείμενα που θα πάρουμε από την εργαλειοθήκη (toolbox).

Επιλέξετε το αντικείμενο «Button» από την εργαλειοθήκη και σχεδιάστε ένα «Button» πάνω στη φόρμα. Επαναλάβετε τη διαδικασία και δημιουργήστε ένα δεύτερο «Button». Η φόρμα σας θα μοιάζει κάπως έτσι:



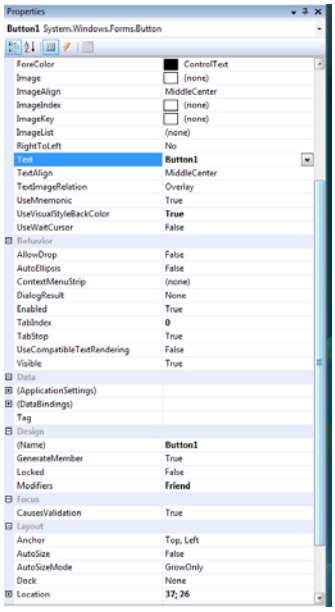
Πατήστε **F5** για να τρέξετε τη εφαρμογή. Πιέστε με το ποντίκι τα «Button» για να δείτε πως συμπεριφέρονται. Δοκιμάστε να πετύχετε την ίδια συμπεριφορά χρησιμοποιώντας το πληκτρολόγιο.

Κλείστε τη φόρμα και επιστρέψτε στο περιβάλλον εργασίας της Visual Basic.

Επιλέξτε το αντικείμενο «Label» από την εργαλειοθήκη και σχεδιάστε ένα «Label» πάνω στη φόρμα.

Στάδιο 2: Δίνοντας τίτλους στα αντικείμενα

Στην εφαρμογή που θα φτιάξουμε πατώντας το ένα κουμπί (Button) θα εμφανίζεται το κείμενο «Hello World» και πατώντας το άλλο κουμπί θα εμφανίζεται το κείμενο «Γειά σου κόσμε».



Επιλέξτε με το ποντίκι το αντικείμενο «Button1» που βρίσκεται πάνω στη φόρμα. Προσέξτε το παράθυρο ιδιοτήτων. Εκεί φαίνονται οι ιδιότητες (properties) του αντικειμένου και τις τιμές που αυτές έχουν. Μπορείτε να μεγαλώσετε το παράθυρο ιδιοτήτων και να βλέπετε περισσότερες ιδιότητες. Στο πάνω μέρος του παραθύρου βλέπετε το όνομα του αντικειμένου που έχει επιλεγεί, στην δική μας περίπτωση βλέπουμε το όνομα Button1.

Εντοπίστε τις ιδιότητες με όνομα «Text» και «Name». Και οι δύο αυτές ιδιότητες έχουν τιμή Button1. Όταν δημιουργείτε ένα νέο αντικείμενο οι ιδιότητές του πρέπει να έχουν κάποια τιμή. Η Visual Basic βάζει τιμές στις ιδιότητες του αντικειμένου που ο προγραμματιστής μπορεί να αλλάξει αλλάζοντας έτσι και τη συμπεριφορά και την εμφάνιση του αντικειμένου. Πειραματιστείτε αλλάζοντας τιμή στην ιδιότητα «Text».

Τι συμβαίνει στο Button1 που βρίσκεται στο παράθυρο σχεδιασμού της φόρμας όταν αλλάζετε την ιδιότητα text από το παράθυρο ιδιοτήτων;

_____

_____

_____

Δώστε τώρα την τιμή «Αγγλικά» στην ιδιότητα «Text» του Button1. Στη συνέχεια επιλέξτε το Button2 και δώστε την τιμή «Ελληνικά» στην ιδιότητα Text.

Επιλέξτε τώρα το αντικείμενο Label1 και δώστε την τιμή «??»

Η ιδιότητα «Text» υπάρχει σε πολλά από τα αντικείμενα που είναι διαθέσιμα από το toolbox. Η τιμή της ιδιότητας «Text» είναι μια συμβολοσειρά που εμφανίζεται πάνω στο αντικείμενο.

Πατήστε τώρα σε ένα σημείο της φόρμας που δεν υπάρχει κανένα αντικείμενο. Στο παράθυρο ιδιοτήτων πρέπει να φαίνεται επιλεγμένο το αντικείμενο Form1 δηλαδή η φόρμα μας. Βρείτε την ιδιότητα «Text» και αλλάξτε την τιμή της.

Τι αλλάζει στην εμφάνιση της φόρμας;

_____

_____

_____

Στάδιο 3: Αλλάζοντας τα ονόματα των αντικειμένων.

Επιλέξτε πάλι το κουμπί «Button1» από τη φόρμα και πηγαίνετε στο παράθυρο ιδιοτήτων. Εντοπίστε την ιδιότητα «Name». Δώστε της την τιμή «btnEn».

Τι άλλαξε στην εμφάνιση της φόρμας και τι στο παράθυρο ιδιοτήτων.

_____

_____

_____

Το όνομα (Name) που δώσατε στο κουμπί σάς χρειάζεται για να είναι προσβάσιμο το κουμπί – όπως και οποιοδήποτε άλλο αντικείμενο – από τον κώδικα. Εφόσον αλλάξατε το όνομά του, το συγκεκριμένο κουμπί από εδώ και μπρος θα το αποκαλείτε btnEn και όχι Button1. Τα ονόματα των αντικειμένων καλό είναι να προσδιορίζουν και τον τύπο τους, έτσι ώστε όταν κάποιος διαβάζει τον κώδικα που γράψατε να καταλαβαίνει σε τι τύπο αντικειμένου αναφέρεστε. Έτσι, τα ονόματα των κουμπιών (Buttons) συστήνεται να ξεκινούν να το btn. Έχετε βαφτίσει το κουμπί που θα εμφανίζει το αγγλικό μήνυμα btnEn (En ως συντομογραφία για το English). Αλλάξτε το όνομα του κουμπιού αυτού.

Αντίστοιχα το όνομα του κουμπιού που θα εμφανίζει το Ελληνικό μήνυμα θα πρέπει να διαμορφωθεί σε btnEl. Τα ονόματα στα αντικείμενα Label συστήνεται να ξεκινούν με το πρόθεμα lbl. Ας αλλάξουμε το όνομα του Label1 σε lblMessage.

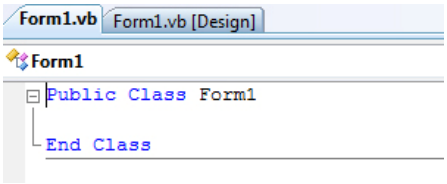
Στάδιο 4. Ελέγχοντας τα συμβάντα (events)

Πατήστε **F5** για να τρέξετε την εφαρμογή σας.

Πατώντας τα κουμπιά τώρα, η εφαρμογή σας δεν κάνει αυτό που θέλετε δηλαδή να εμφανίζει τα αντίστοιχα μηνύματα ανάλογα με το κουμπί που πατήσατε. Για να υλοποιήσετε αυτή τη δυνατότητα θα πρέπει να γράψετε και κώδικα.

Κλείστε την εφαρμογή, πηγαίνετε στο περιβάλλον εργασίας της Visual Basic, στο παράθυρο σχεδιασμού της φόρμας Form1.

Πατήστε δεξί κλικ πάνω στη φόρμα και επιλέξτε «**View Code**». Το περιβάλλον εργασίας της Visual Basic εμφανίζει το παράθυρο κώδικα της φόρμας Form1. Προσέξτε τον κώδικα που βρίσκεται εκεί.



Τώρα επιστρέψτε στο παράθυρο σχεδιασμού της φόρμας. Πατήστε διπλό κλικ πάνω στο κουμπί btnEn. Η Visual Basic σας επιστρέφει στο παράθυρο κώδικα.

Τι διαφοροποίηση υπάρχει στον κώδικα;

_____

_____

_____

Η Visual Basic έχει δημιουργήσει μια υπορουτίνα που είναι ο διαχειριστής συμβάντος (event handler). Ας την εξετάσουμε:

Private Sub btnEn_Click(	Εδώ δηλώνεται η αρχή και το όνομα της υπορουτίνας
ByVal sender As System.Object, ByVal e As System.EventArgs)	Εδώ δηλώνονται οι παράμετροι της
Handles btnEn.Click	Με αυτή τη δήλωση ζητάμε από τη

	Visual Basic να θεωρήσει αυτή την υπορουτίνα ως διαχειριστή συμβάντος για το συμβάν Click του κουμπιού btnEn
End Sub	Εδώ δηλώνεται το τέλος της υπορουτίνας

Οι εφαρμογές φορμών της Visual Basic είναι οδηγούμενες από συμβάντα (event driven). Δηλαδή απλώς περιμένουν μέχρι να συμβεί κάποιο συμβάν – όπως όταν ο χρήστης πατήσει κάποιο κουμπί – και τότε αναλαμβάνει δράση ο ανάλογος διαχειριστής συμβάντων. Η υπορουτίνα που ετοίμασε για μας η Visual Basic διαχειρίζεται το συμβάν του αντικειμένου btnEn που ονομάζεται click. Αυτό το συμβάν λαμβάνει χώρα όταν ο χρήστης πατήσει το κουμπί btnEn με το ποντίκι.

Το ζητούμενο από την εφαρμογή σας είναι όταν πατηθεί αυτό το κουμπί (btnEn) στη θέση του lblMessage να εμφανιστεί το μήνυμα «Hello World». Αυτή τη στιγμή το lblMessage εμφανίζει το μήνυμα «??». Θυμηθείτε πως αυτό το κάναμε αλλάζοντας την τιμή της ιδιότητας Text του lblMessage. Για να υλοποιήσετε την απαίτηση της εφαρμογής θα πρέπει ο κώδικάς σας να αλλάξει την τιμή της ιδιότητας Text του lblMessage. Αυτό γίνεται με την παρακάτω εντολή καταχώρησης.

Me.lblMessage.Text = "Hello world"

Πληκτρολογήστε την ανάμεσα στις γραμμές που ορίζουν την αρχή και το τέλος της υπορουτίνας διαχείρισης. Ο κώδικάς μας πρέπει να είναι ο εξής:

```
Private Sub btnEn_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnEn.Click
    Me.lblMessage.Text = "Hello world"
End Sub
```

Πατήστε **F5** για να τρέξετε την εφαρμογή και πατήστε το btnEn. Η εφαρμογή εμφανίζει αντί για «??» το μήνυμα «Hello world».

Στάδιο 5. Γράφοντας κώδικα για ένα συμβάν

Χρησιμοποιώντας όλα όσα έχετε μάθει στα προηγούμενα στάδια υλοποιήστε το υπόλοιπο της εφαρμογής. Δηλαδή όταν πατηθεί το κουμπί btnEl να εμφανιστεί το μήνυμα «Γεια σου κόσμε».

Στάδιο 6. Τερματισμός της εφαρμογής.

Μέχρι τώρα για να τερματίσετε την εφαρμογή σας κλείνατε απλώς την φόρμα. Ωστόσο, αυτό δεν είναι πάντα σωστό, καθώς οι πιο σύνθετες εφαρμογές πριν τερματίσουν χρειάζεται να κάνουν κάποιες ενέργειες (π.χ. να κλείσουν τα αρχεία που έχουν ανοίξει). Παρακάτω θα μάθετε πώς να τερματίζετε την εφαρμογή σας με κώδικα.

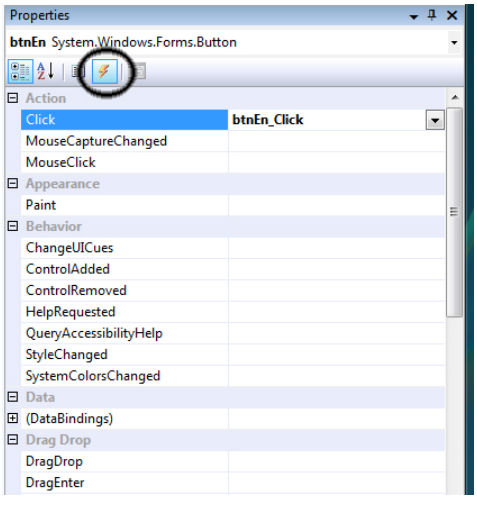
Σχεδιάστε ένα νέο κουμπί πάνω στην φόρμα και δώστε του όνομα btnExit και τίτλο «Έξοδος». Δημιουργήστε το διαχειριστή συμβάντος (event handler) του κουμπιού αυτού (πατώντας διπλό κλικ πάνω στο κουμπί). Συμπληρώστε την ακόλουθη εντολή μέσα στην υπορουτίνα διαχείρισης του συμβάντος:

Application.Exit()

Πατήστε **F5** για να τρέξετε την εφαρμογή. Δοκιμάστε το κουμπί «Έξοδος» (btnExit). Μόλις το πατήσετε η εφαρμογή θα τερματιστεί.

Στάδιο 7. Διαχείριση και άλλων συμβάντων.

Μέχρι τώρα μάθατε να διαχειριζόμαστε το συμβάν «Click» του αντικειμένου κουμπί (Button). Ωστόσο, αυτό δεν είναι το μοναδικό συμβάν που μπορείτε να διαχειριστείτε.



Τα συμβάντα ενός αντικειμένου που βρίσκεται πάνω σε μια φόρμα φαίνονται από το παράθυρο ιδιοτήτων πατώντας το κουμπί events που βρίσκεται στο πάνω μέρος του

παραθύρου ιδιοτήτων. Έχοντας επιλέξει το btnEn βλέπετε τα συμβάντα του αντικειμένου αυτού που μπορείτε να διαχειριστείτε.

Μόνο το συμβάν «Click» έχει τιμή. Η τιμή του είναι το όνομα της υπορουτίνας που ορίσατε ότι θα το διαχειρίζεται.

Επιλέξτε το συμβάν «MouseEnter». Αυτό το συμβάν ενεργοποιείται μόλις ο δείκτης του ποντικιού εισέλθει στην περιοχή της οθόνης που καταλαμβάνει το αντικείμενο. Πατήστε διπλό click πάνω στο παράθυρο ιδιοτήτων στο συμβάν «MouseEnter». Η Visual Basic γράφει πάλι λίγο κώδικα για σας δημιουργώντας μια υπορουτίνα διαχείρισης:

Private Sub btnEn_MouseEnter(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnEn.MouseEnter

End Sub

Εσείς πρέπει τώρα να προσθέσετε τον κώδικα, δηλαδή τις εντολές που θα εκτελούνται όταν συμβαίνει αυτό το συμβάν.

Επιστρέψτε στο παράθυρο σχεδιασμού της φόρμας και σχεδιάστε ένα νέο αντικείμενο Label. Δώστε του όνομα (Name) lblMessageHelp και τίτλο (Text) «??».

Γυρίστε στο παράθυρο κώδικα και στη υπορουτίνα διαχείρισης του συμβάντος «MouseEnter». Συμπληρώστε την παρακάτω γραμμή κώδικα:

Me.lblMessageHelp.Text = "Πατηστε για να δειτε ενα μηνυμα στα Αγγλικά"

Πατήστε **F5** για να τρέξετε τη εφαρμογή και περάστε το δείκτη του ποντικιού πάνω από το κουμπί btnEn. Το μήνυμα «Πατήστε για να δείτε ένα μήνυμα στα Αγγλικά» εμφανίζεται στο lblMessageHelp.

Τώρα θα πρέπει να κάνετε το μήνυμα να εξαφανίζεται όταν ο δείκτης του ποντικιού φεύγει από την περιοχή της οθόνης που καταλαμβάνει το κουμπί btnEn. Για να το κάνετε αυτό θα χρησιμοποιήσετε το συμβάν «MouseLeave». Θα γράψετε την υπορουτίνα διαχείρισης του συμβάντος κατευθείαν στο παράθυρο κώδικα.

Επιλέξτε την υπορουτίνα διαχείρισης που μόλις γράψατε αντιγράψτε την (Copy) και επικολλήστε την (Paste) πριν από την δήλωση «End Class».

Ο κώδικας στο παράθυρο κώδικα θα πρέπει να είναι ο εξής:

```
Public Class Form1

    Private Sub btnEn_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnEn.Click
        Me.lblMessage.Text = "Hello world"
    End Sub

    Private Sub btnEl_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnEl.Click
        Me.lblMessage.Text = "Τειά σου Κόσμε"
    End Sub

    Private Sub btnExit_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnExit.Click
        Application.Exit()
    End Sub

    Private Sub btnEn_MouseEnter(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnEn.MouseEnter
        Me.lblMessageHelp.Text = "Πατήστε για να δείτε ένα μήνυμα στα Αγγλικά"
    End Sub

    Private Sub btnEn_MouseEnter(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnEn.MouseEnter
        Me.lblMessageHelp.Text = " Πατήστε για να δείτε ένα μήνυμα στα Αγγλικά"
    End Sub
```

End Class

Τροποποιήστε το τμήμα κώδικα που αντιγράψατε σύμφωνα με τα παρακάτω:

Αλλάξτε το όνομα της υπορουτίνας σε btnEn_MouseLeave.

Αλλάξτε τη δήλωση *Handles btnEn.MouseEnter* σε *Handles btnEn.MouseLeave*

Αλλάξτε την εντολή *Me.lblMessageHelp.Text = " Πατήστε για να δείτε ένα μήνυμα στα Αγγλικά"* σε *Me.lblMessageHelp.Text = ""*

Η υπορουτίνα διαχείρισης θα μοιάζει ως εξής:

```
Private Sub btnEn_MouseLeave(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnEn.MouseLeave
    Me.lblMessageHelp.Text = ""
End Sub
```

Πατήστε **F5** για να τρέξετε τη εφαρμογή και περάστε το δείκτη του ποντικιού πάνω από το κουμπί btnEn μερικές φορές για να ελέγξετε τη συμπεριφορά του προγράμματος.

Στάδιο 8. Γράφοντας ξανά κώδικα για συμβάντα.

Δημιουργήστε τον αντίστοιχο κώδικα για το κουμπί btnEl. Όταν ο δείκτης του ποντικιού περνάει πάνω από το κουμπί αυτό θα πρέπει να εμφανίζεται το μήνυμα «Πατήστε για να δείτε ένα μήνυμα στα Ελληνικά».

Δραστηριότητα 2: Παίζοντας με τις ιδιότητες

Στάδιο 1. Δημιουργία εφαρμογής με δυο διαχειριστές συμβάντων

Δημιουργήστε μια νέα εφαρμογή φορμών. Αποθηκεύστε το έργο με όνομα drastiriotita2.

Πάνω στην φόρμα δημιουργήστε ένα κουμπί και δώστε του όνομα «btnUp» και τίτλο «Πάτησέ με».

Δημιουργήστε και ένα δεύτερο κουμπί με όνομα «btnDown» και τίτλο «Πάτησέ με».

Δημιουργήστε το διαχειριστή συμβάντος Click του κουμπιού btnUp και συμπληρώστε τον παρακάτω κώδικα.

```
Me.btnDown.Text = "Πάτησέ με"
Me.btnDown.Enabled = True
sender.text = "Με πάτησες"
sender.enabled = False
```

Δημιουργήστε το διαχειριστή συμβάντος Click του κουμπιού btnDown και συμπληρώστε τον παρακάτω κώδικα.

```
Me.btnUp.Text = "Πάτησέ με"
Me.btnUp.Enabled = True
sender.text = "Με πάτησες"
sender.enabled = False
```

Πατήστε **F5** για να τρέξετε την εφαρμογή.

Επεξήγηση των εντολών.

Πατώντας το κουμπί btnUp θα εκτελεστούν πρώτα οι εντολές

```
Me.btnDown.Text = "Πάτησέ με"
Me.btnDown.Enabled = True
```

Το αντικείμενο «Me» είναι μια σύμβαση της Visual Basic και αναφέρεται στη φόρμα πάνω στην οποία βρίσκεται ο κώδικας. Επειδή τα κουμπιά btnDown και btnUp βρίσκονται πάνω στη φόρμα θεωρούνται παιδιά της φόρμας και μπορείτε να αναφερθείτε σε αυτά βάζοντας μια τελεία (.) μετά το «Me». Η ιδιότητα Text ανήκει

στο κουμπί btnDown και έτσι μπορούμε να αναφερθούμε σε αυτή βάζοντας τελεία (.) μετά το όνομα του κουμπιού. Το ίδιο ισχύει για την ιδιότητα «Enabled», που μπορεί να πάρει μόνο δυο τιμές (true και false). Όταν έχει τιμή true το κουμπί μπορεί να πατηθεί, ενώ όταν έχει τιμή false το κουμπί είναι ανενεργό.

Οι επόμενες δυο γραμμές χρησιμοποιούν την παράμετρο «Sender». Προσέξτε στο ορισμό του διαχειριστή συμβάντος τις παραμέτρους:

Private Sub btnUp_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnUp.Click

Η παράμετρος sender αναφέρεται στο αντικείμενο που ενεργοποίησε το συμβάν. Σε αυτή την περίπτωση η παράμετρος sender αναφέρεται στο αντικείμενο *btnUp,* επειδή αυτό το αντικείμενο ενεργοποίησε το συμβάν.

Δοκιμάστε να αντικαταστήσετε την εντολή

sender.text = "Με πάτησες"

με την εντολή

Me.btnUp.text = "Με πάτησες"

Τρέξτε το πρόγραμμα. Παρατηρείτε κάποια διαφορά;

_____

_____

Στάδιο 2. Διαχείριση πολλών συμβάντων με την ίδια υπορουτίνα

Ανοίξτε το έργο drastiriotita2_2.

Τρέξτε την εφαρμογή. Δεν πρέπει να έχει καμία διαφορά με την εφαρμογή που φτιάξατε στο προηγούμενο στάδιο.

Τώρα ανοίξτε το παράθυρο κώδικα και παρατηρήστε τον κώδικα.

Public Class Form1

Private Sub btn_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnUp.Click, btnDown.Click

```
        Me. btnDown.Text = "Πάτησέ με"
        Me.btnDown.Enabled = True
        Me.btnUp.Text = "Πάτησέ με"
        Me.btnUp.Enabled = True
        sender.text = "Με πάτησες"
        sender.enabled = False
    End Sub

End Class
```

Υπάρχει μόνο μια υπορουτίνα που διαχειρίζεται τα συμβάντα Click και των δυο κουμπιών (btnUp, btnDown). Αυτό το πετυχαίνουμε αναφέροντας μετά τη δήλωση «Handles» τα συμβάντα που θα διαχειριστεί αυτή η υπορουτίνα χωρίζοντάς τα με κόμμα. Τα συμβάντα τα δηλώνουμε με τη εξής σύνταξη: *αντικείμενο.συμβάν* (π.χ. *Handles btnUp.Click, btnDown.Click*).

Όταν πατηθεί οποιοδήποτε από τα δυο κουμπιά το πρόγραμμα με τις εντολές

```
Me. btnDown.Text = "Πάτησέ με"
Me.btnDown.Enabled = True
Me.btnUp.Text = "Πάτησέ με"
Me.btnUp.Enabled = True
```

βάζει τίτλο «Πάτησέ με» και στα δυο κουμπιά και τα κάνει και τα δυο ενεργά.

Ακολούθως χρησιμοποιώντας την παράμετρο sender με τις εντολές

```
sender.text = "Με πάτησες"
sender.enabled = False
```

Βάζει τον τίτλο «Με πάτησες» στο κουμπί που πατήθηκε και το κάνει και ανενεργό.

Κλείστε το έργο drastiriotita2_2. Μετά επιστρέψτε στο έργο drastiriotita2 που φτιάξατε εσείς. Σβήστε τον κώδικα και δοκιμάστε να τον ξαναγράψετε με τον τρόπο που μάθατε στο έργο drastiriotita2_2.

Φύλλο εργασίας

3. Η έννοια των σταθερών και των μεταβλητών στη Visual Basic

Όνομα:

Τάξη:

Διάρκεια: 3 διδακτικές ώρες.

Διδακτικοί στόχοι:

Μετά την ολοκλήρωση των δραστηριοτήτων του συγκεκριμένου φύλλο εργασίας θα:

- γνωρίζετε τους τύπους των σταθερών και των μεταβλητών της Visual Basic
- γνωρίζετε πώς δηλώνονται οι μεταβλητές
- είστε ικανοί να κάνετε αριθμητικές και αλφαριθμητικές πράξεις μέσα από κώδικα της Visual Basic
- γνωρίζετε να βάζετε σχόλια σε ένα πρόγραμμα Visual Basic

Δραστηριότητα 1 : Σταθερές στην Visual Basic

Όταν γράφετε κώδικα χρειάζεται να αποτυπώνετε σταθερές τιμές με τρόπο που να τις καταλαβαίνει ο μεταγλωττιστής. Οι σταθερές τιμές χωρίζονται σε τρεις μεγάλες κατηγορίες : αριθμητικές , λογικές , αλφαριθμητικές

Οι αριθμητικές σταθερές αναπαριστούν αριθμούς. Για να αναπαρασταθεί μια αριθμητική σταθερά στον κώδικα της Visual Basic χρειάζεται να γραφτεί απλώς ο αριθμός που εκφράζει, π.χ 1 ή 1345678. Προσοχή δεν θα πρέπει να χωρίζετε τις χιλιάδες με τελεία (.). Για να γράψετε ένα δεκαδικό αριθμό χρησιμοποιείστε την τελεία (.) για να χωρίσετε το δεκαδικό από το ακέραιο μέρος π.χ. 210.10

Οι λογικές σταθερές είναι δυο (true και false).

Οι αλφαριθμητικές σταθερές αναπαριστούν μια σειρά χαρακτήρων. Στην Visual Basic οι αλφαριθμητικές σταθερές μπαίνουν μέσα σε αγγλικά εισαγωγικά(”), π.χ. “Hello World” ή “Στις 8 το βράδυ”.

Σαν σταθερές, επίσης, μπορούν να χρησιμοποιηθούν και πράξεις μεταξύ σταθερών, π.χ. για να γράψετε σαν σταθερά το διπλάσιο του 686 χωρίς να χρησιμοποιήσετε calculator μπορείτε να γράψετε κατευθείαν 686*2.

Γράψτε δίπλα σε κάθε σταθερά τι τύπου σταθερά είναι (αριθμητική, λογική, αλφαριθμητική)

18	
“2104896520”	
“A9”	
9.30	
true	
10/2	

Δραστηριότητα 2 : Η εντολή καταχώρησης στην Visual Basic

Όπως όλες οι γλώσσες προγραμματισμού έτσι και η Visual Basic χρησιμοποιεί μεταβλητές για να αποθηκεύσει τιμές. Η τιμή μιας μεταβλητής, όπως προκύπτει και από το όνομά της, μπορεί να μεταβληθεί κατά τη διάρκεια της εκτέλεσης του προγράμματος χρησιμοποιώντας εντολές καταχώρησης.

Η εντολή καταχώρησης στη Visual Basic αναπαρίσταται με το σύμβολο “=”. Μια απλή εντολή καταχώρησης είναι η παρακάτω:

a =1

Η παραπάνω εντολή διαβάζεται ως εξής: καταχώρησε στην μεταβλητή a την τιμή 1. Η μεταβλητή a θα διατηρήσει την τιμή 1 μέχρι να την αλλάξουμε με μια νέα εντολή καταχώρησης ή μέχρι να τερματιστεί το πρόγραμμα.

Άλλο παράδειγμα είναι το εξής:

str = “Hello World”

αυτό διαβάζεται : καταχώρησε στη μεταβλητή str την τιμή “Hello World”

Ένα πιο σύνθετο παράδειγμα

a=1
b=a

αυτό διαβάζεται:
καταχώρησε στην μεταβλητή a την τιμή 1.
καταχώρησε στην μεταβλητή b την τιμή της μεταβλητής a.
Μετά την εκτέλεση των δυο εντολών η μεταβλητή b θα έχει και αυτή την τιμή 1.

Παρακολουθήστε τις παρακάτω εντολές και γράψτε τι τιμή θα έχει η μεταβλητή c:

a = 3
b = 4
c = b
c = a
c = c

_____

_____

_____

Δραστηριότητα 3: Ορισμός μεταβλητών και τύποι μεταβλητών στη Visual Basic

Για να χρησιμοποιηθεί μια μεταβλητή πρέπει πρώτα να οριστεί. Αυτό σημαίνει πως δίνουμε εντολή στον υπολογιστή να δημιουργήσει μια μεταβλητή. Δηλαδή, ο υπολογιστής θα κρατήσει ένα χώρο στη μνήμη, στον οποίο θα μπορεί το πρόγραμμα μας να αποθηκεύει τιμές. Οι τιμές που θα αποθηκευτούν εκεί μπορεί να είναι διαφορετικών τύπων (θυμηθείτε τους τρεις τύπους σταθερών που προαναφέραμε). Αντίστοιχα και όταν δηλώνουμε μια μεταβλητή θα πρέπει να εξηγήσουμε στον

υπολογιστή τι τύπους τιμών σχεδιάζουμε να αποθηκεύσουμε εκεί, ώστε να κρατήσει τον ανάλογο χώρο στη μνήμη.

Ο ορισμός μιας μεταβλητής γίνεται με την δήλωση *Dim*. Μετά βάζουμε το όνομα της μεταβλητής, δηλαδή ένα όνομα, με το οποίο θα μπορούμε να αναφερθούμε σε αυτή την περιοχή της μνήμης έτσι ώστε να μπορούμε να αποθηκεύουμε τιμές και μετά να τις διαβάζουμε. Μετά μπαίνει η δήλωση as και στο τέλος ο τύπος της μεταβλητής, για παράδειγμα:

Dim num As Integer (δημιούργησε μια μεταβλητή στη οποία θα αποθηκεύουμε ακέραιες τιμές)
Dim str As String (δημιούργησε μια μεταβλητή στη οποία θα αποθηκεύουμε αλφαριθμητικές τιμές
Dim bool As Boolean (δημιούργησε μια μεταβλητή στη οποία θα αποθηκεύουμε λογικές τιμές)

Στις μεταβλητές μπορείτε να καταχωρήσετε τιμές που να αντιστοιχούν με τον τύπο τους.

Εντοπίστε αν οι παρακάτω καταχωρήσεις καταχωρούν τιμές σύμφωνα με το τύπο της μεταβλητής και γιατί (θεωρήστε ότι ισχύουν οι τρεις δηλώσεις μεταβλητών που δόθηκαν σαν παράδειγμα πιο πάνω).

num = 100	
num = 12.10	
str = 100	
str = “είμαι μαθητής”	
str = “18.30”	
bool = 217	
bool = false	

Αν δοκιμάσετε να καταχωρίσετε τιμή άλλου τύπου από τον τύπο της μεταβλητής, η Visual Basic θα προσπαθήσει να μετατρέψει την τιμή στον τύπο της μεταβλητής (το αποτέλεσμα δεν είναι πάντα το αναμενόμενο και πρέπει να αποφεύγεται). Αν δεν τα καταφέρει το πρόγραμμα σας θα τερματιστεί με ένα μήνυμα λάθους.

Δραστηριότητα 4: Οι αριθμητικές μεταβλητές αναλυτικότερα

Στην προηγούμενη δραστηριότητα ορίσατε μια μεταβλητή ως ακέραιο (Integer). Στο χώρο της μνήμης που κρατάει ο υπολογιστής για μια μεταβλητή τύπου Integer μπορείτε να αποθηκεύσετε τιμές από μείον 2 δισεκατομμύρια μέχρι δυο δισεκατομμύρια περίπου. Για την ακρίβεια από -2.147.483.648 εως 2.147.483.647 . Για να αποθηκεύσετε μεγαλύτερους ακεραίους ή δεκαδικούς αριθμούς χρειάζεστε άλλου τύπου μεταβλητές. Στον παρακάτω πίνακα παρατίθενται (από το αρχείο βοήθειας της Visual Basic) όλοι οι τύποι αριθμητικών μεταβλητών που μπορούν να οριστούν και οι τιμές που μπορούν να πάρουν.

Byte	0 through 255 (unsigned)		
Decimal	0 through +/- 79,228,162,514,264,337,593,543,950,335 (+/- 7.9...E+28) † with no decimal point; 0 through +/- 7.9228162514264337593543950335 with 28 places to the right of the decimal; smallest nonzero number is +/- 0.0000000000000000000000000001 (+/-1E-28) †		
Double (double-precision floating-point)	-1.79769313486231570E+308 through -4.94065645841246544E-324 † for negative values; 4.94065645841246544E-324 through 1.79769313486231570E+308 † for positive values		
Integer	-2,147,483,648 through 2,147,483,647 (signed)		

Long (long integer)	-9,223,372,036,854,775,808 through 9,223,372,036,854,775,807 (9.2...E+18 †) (signed)		
SByte	-128 through 127 (signed)		
Short (short integer)	-32,768 through 32,767 (signed)		
Single (single-precision floating-point)	-3.4028235E+38 through -1.401298E-45 † for negative values; 1.401298E-45 through 3.4028235E+38 † for positive values		
UInteger	0 through 4,294,967,295 (unsigned)		
ULong	0 through 18,446,744,073,709,551,615 (1.8...E+19 †) (unsigned)		
UShort	0 through 65,535 (unsigned)		

Σημειώστε στην τελευταία στήλη αν οι τιμές που μπορεί να πάρει κάθε τύπος είναι ακέραιες ή δεκαδικές.

Στη συνέχεια, σημειώστε με ένα ναι ή ένα όχι στην τελευταία στήλη αν ο τύπος μπορεί να πάρει αρνητικές τιμές (π.χ -2)

Δραστηριότητα 5 : Πράξεις με αριθμητικές μεταβλητές

Με τις μεταβλητές μπορούν να γίνουν αριθμητικές πράξεις, άλλωστε οι υπολογιστές αρχικά κατασκευάστηκαν γι αυτό το σκοπό.

Για να γίνουν αριθμητικές πράξεις με την visual Basic χρησιμοποιούνται οι εξής τελεστές:

Πρόσθεση	+	a = 1+2	Το a παίρνει τιμή 3
Αφαίρεση	-	a = 3-4	Το a παίρνει τιμή -1
Πολλαπλασιασμός	*	a = 4*5	Το a παίρνει τιμή 20
Διαίρεση	/	a = 9/2	Το a παίρνει τιμή 4.5
Ύψωση σε δύναμη	^	a = 2^3	Το a παίρνει τιμή 8
Ακέραιο μέρος διαίρεσης	\	a = 9/2	Το a παίρνει τιμή 4
Υπόλοιπο διαίρεσης	Mod	10 Mod 3	Το a παίρνει τιμή 1

Οι αριθμητικές πράξει μπορούν να γίνουν όχι μόνο με σταθερές τιμές, όπως στα παραπάνω παραδείγματα, αλλά και με τιμές μεταβλητών, π.χ.:

```
Dim num As Integer 'δημιουργησε μια μεταβλητη τυπου Integer με όνομα num
num = 10           'βάλε τιμή 10 στην μεταβλητή num
num = num + 1      'υπολόγισε την τιμή num + 1 και καταχώρησέ την στη μεταβλητή
num
```

Στην Visual Basic όταν χρειάζεται να μπουν σχόλια στον κώδικα χρησιμοποιείται το σύμβολο (‘). Η Visual Basic αγνοεί οτιδήποτε γραφτεί μετά από αυτό το σύμβολο και μέχρι το τέλος της γραμμής. Στο παραπάνω παράδειγμα σε σχόλια βρίσκεται ο ψευδοκώδικας του προγράμματος.

Στις παρακάτω γραμμές κώδικα συμπληρώστε στα σχόλια τις τιμές που παίρνουν οι μεταβλητές που ζητούνται μετά την εκτέλεση της εντολής.

```
Dim num As Integer
Dim int As Integer
num = 10       ' num : .....
```

```
num = num + 1   ' num : .....
int = 9         ' num : ..... ,  int :  .....
num = int \ 3   ' num : ..... ,  int :  .....
int = num + int ' num : ..... ,  int :  .....
num = int Mod 5 ' num : ..... ,  int :  .....
```

Όταν υπάρχουν πολύπλοκες αριθμητικές παραστάσεις π.χ. a = 2 * 3 + 2 * 4 με διαφορετικούς τελεστές, οι πράξεις μεταξύ των τελεστών του πολλαπλασιασμού και της διαίρεσης εκτελούνται πρώτα και μετά οι πράξεις μεταξύ τελεστών πρόσθεσης και αφαίρεσης. Στο παραπάνω παράδειγμα η μεταβλητή a θα πάρει τιμή 14 και είναι ισοδύναμο με το a = (2*3) + (2 * 4). Για να υπολογιστεί το a = 2 * (3 + 2) * 4, που δίνει την τιμή 40 στην μεταβλητή a, χρησιμοποιούνται παρενθέσεις για να εξηγηθεί στην Visual Basic ποια πράξη να εκτελέσει πρώτα.

Σημείωση. Όταν δηλώνουμε μια μεταβλητή μπορούμε να δώσουμε αρχική τιμή στην γραμμή της δήλωσης. Δηλαδή :

```
Dim num As Integer = 0
```

Αυτή η εντολή ειναι ισδυναμη με τις παρακάτω

```
Dim num As Integer
num = 0
```

Δραστηριότητα 6: Αλφαριθμητικές μεταβλητές

Υπάρχουν δυο τύποι αλφαριθμητικών μεταβλητών. Στις αλφαριθμητικές μεταβλητές μπορούν να δοθούν σαν τιμές χαρακτήρες. Ο τύπος Char μπορεί να πάρει τιμές που αποτελούνται από έναν χαρακτήρα, π.χ.:

```
Dim ch As Char
ch = "a"
Dim myChar As Char
myChar = ";"
```

Ο άλλος τύπος είναι ο τύπος String, με τον οποίο ασχοληθήκατε στη δραστηριότητα 3. Μπορεί να πάρει τιμές που αποτελούνται σειρές χαρακτήρων, π.χ.:

```
Dim str As String
str = "Μια φορά και έναν καιρό"
Dim myStr As String
myStr = "2108865328, το τηλέφωνο μου"
```

Με τις αλφαριθμητικές μεταβλητές μπορεί να γίνει μόνο μια πράξη, η σύνθεση (Concatenation). Ο τελεστής της σύνθεσης είναι ο "&". Εναλλακτικά, μπορεί να χρησιμοποιηθεί και ο τελεστής "+"

Παράδειγμα:

```
Dim strHello As String
strHello = "Hello"
Dim space As Char
space = " "
Dim strWorld As String
strWorld = "World"
Dim str As String
str = strHello & space & strWorld
```

Η μεταβλητή str θα πάρει την τιμή "Hello World" μετά την εκτέλεση των εντολών.

Συμπληρώστε παρακάτω στα σχόλια τις τιμες που θα πάρει η μεταβλητή str μετα την εκτέλεση της κάθε εντολής

```
Dim str As String
str = "Hello " + "World"   '_____
str = str & "."            '_____
str = "Hello+" & "world"   '_____
str = "Hello & World"      '_____
str = "Hello & " + "World" '_____
```

Φύλλο εργασίας

4. Οι συναρτήσεις και υπορουτίνες στη Visual Basic

Όνομα:

Τάξη:

Διάρκεια: 3 διδακτικές ώρες.

Διδακτικοί στόχοι:

Με το συγκεκριμένο φύλλο εργασίας:

- θα μάθετε να δημιουργείτε υπορουτίνες στη Visual Basic
- θα μάθετε να δημιουργείτε συναρτήσεις που επιστρέφουν τιμές
- θα αποκτήσετε μία πρώτη επαφή με τις έννοιες των τοπικών και καθολικών μεταβλητών.

Δραστηριότητα 1 : Δημιουργία μιας υπορουτίνας

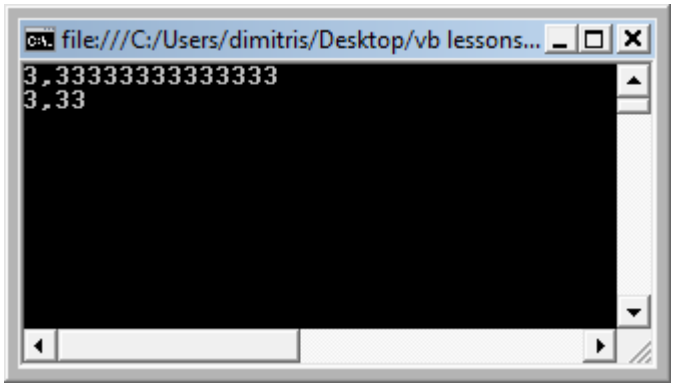
Στο προηγούμενο φύλλο εργασίας μάθατε πώς συντάσσονται οι εντολές καταχώρησης και επίσης πώς μπορείτε να κάνετε αριθμητικές πράξεις και να καταχωρήσετε το αποτέλεσμά τους σε μια μεταβλητή.

Στον κώδικα που ακολουθεί τοποθετήστε την τιμή 3.333… στην μεταβλητή num. Με τις υπόλοιπες εντολές προσπαθήστε να κρατήσετε μόνο τα δυο πρώτα δεκαδικά του περιοδικού αυτού αριθμού, έτσι ώστε να καταχωρήσετε στη μεταβλητή num τον αριθμό 3.33 .

```
Sub Main()
    Dim num As Double
    num = 10 / 3
    Console.WriteLine(num) ' δειξε την τιμή του num στην κονσολα
    Dim temp As Double
```

```
        temp = num * 100     ' Βάλε στη μεταβλητή temp την τιμή num *100 -> 333.33333...
        temp = temp \ 1      ' Βαλε στη μεταβλητή temp την τιμή (ακέραιο μερος της διάρεσης (temp δια 1) -> 333)
        temp = temp / 100  ' Βάλε στη μεταβλητή temp την τιμή temp δια 100
        num = temp         ' Βάλε στη μεταβλητή num την τιμή temp
        Console.WriteLine(num) ' δειξε την τιμή του num στην κονσολα
        Console.ReadLine()       ' διάβασε χαρακτήρες από το πληκτρολόγιο μεχρι να πατηθεί enter
    End Sub
```

Δημιουργήστε μια νέα εφαρμογή κονσόλας σε Visual Basic και πληκτρολογήστε τον κώδικα. Εμφανίζονται τα παρακάτω αποτελέσματα όταν τρέξετε την εφαρμογή:



Ο υπολογισμός που κάνατε (να βρείτε την τιμή ενός αριθμού με ακρίβεια δυο δεκαδικών ψηφίων) απαιτεί περισσότερες από μια εντολές. Οι εντολές αυτές θα ήταν καλό να ομαδοποιηθούν. Για την ομαδοποίηση εντολών η visual basic προσφέρει υπορουτίνες (subs) και συναρτήσεις (functions).

Συναντήσατε πρώτη φορά τις υπορουτίνες όταν γράψατε υπορουτίνες χειρισμού συμβάντων στο φύλλο εργασίας 2. Οι υπορουτίνες ομαδοποιούν μια σειρά εντολών. Όταν «καλείτε» μια υπορουτίνα εκτελούνται οι εντολές που περιλαμβάνει. Στο Φ.Ε.2 οι υπορουτίνες χειρισμού συμβάντων που φτιάξατε καλούνται από την Visual Basic όταν πατάτε το αντίστοιχο κουμπί. Ωστόσο, μπορείτε να φτιάξετε υπορουτινες που να τις καλείτε από τον κώδικα, σαν να ήταν εντολές της Visual Basic.

Όπως και οι μεταβλητές, οι υπορουτίνες πρέπει να δηλώνονται. Η δήλωση της υπορυτίνας γίνεται ως εξής:

```
Sub calcubateTwodecimalPlaces()

End Sub
```

Στην αρχή μπαίνει η δήλωση «Sub». Μετά το όνομα της υπορουτίνας ακολουθούμενο από παρενθέσεις. Μετά γράφετε τις εντολές που θα περιέχει η υπορουτίνα. Τέλος, μπαίνει η δήλωση «End Sub».

Ξαναγράψτε τον κώδικά σας με χρήση υπορουτίνας

```
Module Module1
   Dim num As Double
   Sub calcubateTwoDecimalPlaces ()
      Dim temp As Double
      temp = num * 100     ' Βάλε στη μεταβλητή temp την τιμή num *100 -> 333.33333...
      temp = temp \ 1      ' Βαλε στη μεταβλητή temp την τιμή (ακέραιο μερος της διάρεσης (temp δια 1) -> 333)
      temp = temp / 100  ' Βάλε στη μεταβλητή temp την τιμή temp δια 100
      num = temp         ' Βάλε στη μεταβλητή num την τιμή temp
   End Sub
   Sub Main()
      num = 10 / 3
      Console.WriteLine(num) ' δειξε την τιμή του num στην κονσολα
      calcubateTwoDecimalPlaces()      ' καλούμε την υπορουτίνα calcubateTwoDecimalPlaces
      Console.WriteLine(num) ' δειξε την τιμή του num στην κονσολα
      Console.ReadLine()       ' διάβασε χαρακτήρες από το πληκτρολόγιο μεχρι να πατηθεί enter
   End Sub

End Module
```

Παρατηρήστε τη δήλωση «Dim num As Double». Έχει μετακινηθεί έξω από τη δήλωση «Sub Main()».

Πληκτρολογήστε τον παραπάνω κώδικα. Δοκιμάστε να μεταφέρετε την δήλωση «Dim num As Double» μετά τη δήλωση «Sub Main()» και προσπαθήστε να τρέξετε το πρόγραμμα. Τι Παρατηρείτε;

.....
.....
.....
.....

Δραστηριότητα 2: Υπορουτίνα με παραμέτρους.

Οι υπορουτίνες χρησιμοποιούνται για να ομαδοποιηθούν εντολές και να γίνουν όσο το δυνατόν πιο ανεξάρτητες από το υπόλοιπο πρόγραμμα. Η υπορουτίνα που γράψατε υπολογίζει τα δυο πρώτα δεκαδικά ψηφία της μεταβλητής *num*. Φανταστείτε πως έχετε δυο μεταβλητές τις *num1* και *num2* που πρέπει να κάνετε τον ίδιο υπολογισμό. Δεν θα ήταν δόκιμο να γράψετε μια διαφορετική υπορουτίνα για κάθε μία. Γι αυτό το λόγο χρησιμοποιούνται οι παράμετροι. Η παράμετρος είναι μια μεταβλητή που «στέλνεται» σε μια υπορουτίνα. Έτσι μπορείτε να κάνετε μια υπορουτίνα να κάνει διαφορετικά πράγματα ανάλογα με τις παραμέτρους που της στέλνετε. Η δήλωση υπορουτίνας με παραμέτρους γίνεται ως εξής:

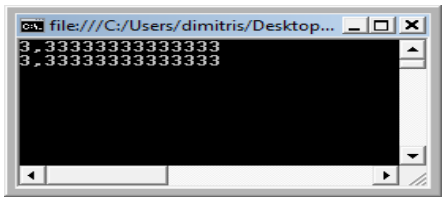
```
Sub calculate2DecimalPlaces(ByVal theNum As Double)

End Sub
```

Παρατηρήστε ότι μέσα στις παρενθέσεις δηλώνεται μια μεταβλητή μετά την δήλωση «ByVal». Όταν καλείτε την υπορουτίνα χρειάζεται να της στέλνετε μια τιμή που θα είναι η αρχική τιμή της παραμέτρου. Την τιμή αυτή τη στέλνετε βάζοντας μέσα στις παρενθέσεις μια μεταβλητή ή μια σταθερά. Δείτε τον παρακάτω κώδικα:

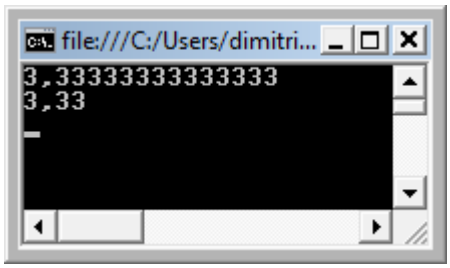
```
Module Module1

    Sub calculate2DecimalPlaces(ByVal theNum As Double)
        Dim temp As Double
        temp = theNum * 100   ' Βάλε στη μεταβλητή temp την τιμή num *100
        temp = temp \ 1       ' Βαλε στη μεταβλητή temp την τιμή (ακέραιο μερος της διάρεσης (temp δια 1)
        temp = temp / 100  ' Βάλε στη μεταβλητή temp την τιμή temp δια 100
        theNum = temp         ' Βάλε στη μεταβλητή num την τιμή temp
    End Sub

    Sub Main()
        Dim num1 As Double
        num1 = 10 / 3
        Console.WriteLine(num1) ' δειξε την τιμή του num στην κονσολα
        calculate2DecimalPlaces(num1)      ' καλούμε την υπορουτίνα calcubateTwoDecimalPlaces
        Console.WriteLine(num1) ' δειξε την τιμή του num στην κονσολα
```

```
        Console.ReadLine()      ' διάβασε χαρακτήρες από το πληκτρολόγιο μεχρι να πατηθεί enter

    End Sub

End Module
```

Έχετε δημιουργήσει μια υπορουτίνα με παράμετρο. Σαν παράμετρο όταν καλείτε την υπορουτίνα με την εντολή calculate2DecimalPlaces(num1) περνάτε την τιμή της μεταβλητής *num1*. Όταν αρχίζει να εκτελείται η υπορουτίνα, η μεταβλητή theNum που είναι το όνομα τής παραμέτρου παίρνει σαν τιμή την τιμή της μεταβλητής *num1*.
Προσέξτε τη δήλωση «Dim num1 As Double». Αυτή βρίσκεται μετά τη δήλωση «Sub Main()». Δοκιμάστε να την μεταφέρετε κάτω από τη δήλωση «Module Module1». Παρατηρείστε πως δεν υπάρχει πρόβλημα.
Πληκτρολογήστε και τρέξτε το πρόγραμμα.
Το αποτέλεσμα του προγράμματος είναι αυτό:



Ωστόσο, αυτό δεν είναι το επιθυμητό αποτέλεσμα. Ο κώδικας που γράψατε στη δραστηριότητα 1 είχε το παρακάτω αποτέλεσμα:



Προσπαθήστε να εξηγήσετε γιατί συμβαίνει αυτό.
(Γράψτε την εντολή «Console.WriteLine(theNum)» ακριβώς πριν από το τέλος της υπορουτίνας calculate2DecimalPlaces() για να κατανοήσετε καλύτερα το πρόβλημα).

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Δραστηριότητα 3 : Υπορουτίνες που επιστρέφουν τιμές (συναρτήσεις).

Στην προηγούμενη δραστηριότητα διαπιστώσαμε πως η μεταβλητή *theNum.* που είναι παράμετρος της υπορουτίνας. Αλλάξτε τιμή μέσα στην υπορουτίνα. Ωστόσο, η μεταβλητή *num1* δεν άλλαξε τιμή.

Ένας τρόπος για να μεταφέρετε μια τιμή από μια υπορουτίνα είναι να δημιουργήσετε μια συνάρτηση (function) αντί για μια υπορουτίνα. Συνάρτηση είναι μια υπορουτίνα που, αφού εκτελεστεί, επιστρέφει μια τιμή. Η δήλωση μιας συνάρτησης γίνεται ως εξής:

```
Function calculate2DecimalPlaces(ByVal theNum As Double) As Double

End Function
```

Η κλήση της συνάρτησης γίνεται καταχωρώντας την τιμή της συνάρτησης σε μια μεταβλητή:

```
num1 = calculate2DecimalPlaces(num1)
```

Παρατηρείστε στη δήλωση της συνάρτησης πως στο τέλος ακολουθεί και μια δήλωση τύπου («As Double» στην προκειμένη περίπτωσή). Επειδή η συνάρτηση επιστρέφει μια τιμή που μπορείτε να καταχωρήσετε σε μια μεταβλητή, πρέπει να δηλώσετε τι είδους τιμές θα επιστρέφει η συνάρτηση, όπως όταν δηλώνετε μεταβλητή.

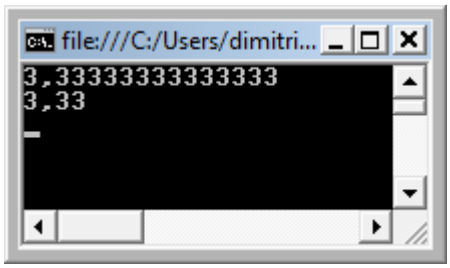
Παρακάτω βλέπετε τον κώδικα που χρησιμοποιεί τη συνάρτηση calculate2DecimalPlaces έτσι ώστε να δείξει τις σωστές τιμές στην οθόνη:

```
Module Module1
   Dim num1 As Double
   Function calculate2DecimalPlaces(ByVal theNum As Double) As Double
      Dim temp As Double
```

```
        temp = theNum * 100    ' Βάλε στη μεταβλητή temp την τιμή num *100 -> 333.33333...
        temp = (temp \ 1)     ' Βαλε στη μεταβλητή temp την τιμή (ακέραιο μερος της διάρεσης (temp δια 1) -> 333)
        temp = temp / 100  ' Βάλε στη μεταβλητή temp την τιμή temp δια 100
        theNum = temp         ' Βάλε στη μεταβλητή num την τιμή temp
        Return theNum
    End Function

    Sub Main()
        num1 = 10 / 3
        Console.WriteLine(num1) ' δειξε την τιμή του num στην κονσολα
        num1 = calculate2DecimalPlaces(num1)'καλούμε την υπορουτίνα calculate2DecimalPlaces με παράμετρο τη μεταβλητή num1 και καταχωρούμε την τιμή της στη μεταβλητή num1
        Console.WriteLine(num1) ' δειξε την τιμή του num στην κονσολα
        Console.ReadLine()      ' διάβασε χαρακτήρες από το πληκτρολόγιο μεχρι να πατηθεί enter
    End Sub

End Module
```

Πληκτρολογήστε και τρέξτε τον κώδικα.

Το αποτέλεσμα του προγράμματος είναι αυτό:



Τώρα έχετε το επιθυμητό αποτέλεσμα.

Παρατηρήστε στον κώδικα με ποια εντολή επιστρέφουμε τιμές από την συνάρτηση.

Η εντολή *return* διακόπτει την εκτέλεση των εντολών της συνάρτησης και επιστρέφει μια τιμή.

Δοκιμάστε να γράψετε μια συνάρτησης με όνομα piValue. Δεν θα έχει παραμέτρους και θα επιστρέφει πάντα την τιμή του αριθμού π (3.1415926535). Το π είναι τύπου *Double*.

……

……

……

……

……

……

Δραστηριότητα 4 : Υπορουτίνες με περισσότερες παραμέτρους.

Στις υπορουτίνες και τις συναρτήσεις μπορούν να περάσουν περισσότερες από μια παραμέτρους. Παρακάτω βλέπετε μια εξελιγμένη μορφή της συνάρτησης calculate2DecimalPlaces που ονομάζεται calculateNDecimalPlaces. Η πρώτη παράμετρος είναι ο αριθμός που θα στρογγυλοποιηθεί. Η δεύτερη παράμετρος είναι ο αριθμός των δεκαδικών ψηφίων που είναι το ζητούμενο.

```
Function calculateNDecimalPlaces(ByVal theNum As Double, ByVal decimalDigits
As Integer) As Double
    Dim temp As Double
    Dim multiplier As Integer
    multiplier = 10 ^ decimalDigits
    temp = theNum * multiplier
    temp = (temp \ 1)
    temp = temp / multiplier
    theNum = temp
    Return theNum
  End Function
```

Πληκτρολογήστε και τρέξτε τον κώδικα. Για να καλέσετε την υπορουτίνα αυτή, πρέπει να βάλετε δυο τιμές μέσα στις παρενθέσεις, πχ.

num1 = calculateNDecimalPlaces(num1, 0).

Η παραπάνω εντολή θα επιστρέψει τον αριθμό *num1* χωρίς δεκαδικά ψηφία.

Δοκιμάστε να γράψετε την συνάρτηση που περιγράφεται παρακάτω.

Η συνάρτηση ονομάζεται *embadonOrt* και υπολογίζει το εμβαδόν ενός ορθογωνίου. Παίρνει δυο παραμέτρους: βάση και ύψος. Όλες οι τιμές (βάση, ύψος, *embadonOrt*) είναι τύπου Double.

……

……

……

……

……

……

Δραστηριότητα 5 : Κλήση υπορουτίνας από υπορουτίνα.

Δοκιμάστε να γράψετε την συνάρτηση που περιγράφεται παρακάτω.

Η συνάρτηση ονομάζεται *embadonKyklou* και υπολογίζει το εμβαδόν ενός κύκλου (π*R^2). Παίρνει σαν παράμετρο την ακτίνα του κύκλου. Τον αριθμό π θα τον βρείτε καλώντας τη συνάρτηση *piValue* που γράψατε στη δραστηριότητα 3. Όλες οι τιμές (ακτίνα, π, *embadonKyklou*) είναι τύπου Double.

……

……

……

……

……

……

Φύλλο εργασίας

5. Εισαγωγή στα αντικείμενα στη Visual Basic

Όνομα:

Τάξη:

Διάρκεια: 3 διδακτικές ώρες.

Διδακτικοί στόχοι:

Με το συγκεκριμένο φύλλο εργασίας:

- Θα εξοικειωθείτε με το αντικείμενο *Console* της Visual Basic και θα μάθετε να το μεταχειρίζεστε για να παίρνετε input από τον χρήστη σε εφαρμογές κονσόλας καθώς και να δίνετε output στον χρήστη.
- Θα μάθετε τις βασικές μεθόδους του αντικειμένου Math
- Θα μάθετε να χειρίζεστε τις μεταβλητές τύπου String σαν αντικείμενα
- Θα μάθετε την κλάση Date για τον χειρισμό ημερομηνιών

Δραστηριότητα 1 : Το αντικείμενο console

Στα παραδείγματα με τα οποία έχετε ασχοληθεί μέχρι τώρα στις εφαρμογές κονσόλας χρησιμοποιήσατε τις εντολές «Console.WriteLine()» και «Console.ReadLine()». Οι εντολές αυτές αποτελούνται από δυο λέξεις που ενώνονται μεταξύ τους με τελεία(.). Η πρώτη λέξη και στις δυο, όπως φαίνεται, είναι η λέξη *Console*. *Console* είναι το όνομα ενός αντικειμένου που είναι διαθέσιμο στις εφαρμογές κονσόλας. Τα αντικείμενα μπορούν να θεωρηθούν σαν ένα σύνολο από κώδικα, τμήματα του οποίου μπορούν χρησιμοποιηθούν. Τα αντικείμενα για τον προγραμματιστή είναι μαύρα κουτιά (black boxes). Αυτό σημαίνει πως δεν βλέπει τις διεργασίες που γίνονται μέσα στο αντικείμενο, αλλά μόνο τις τιμές που δίνονται στην είσοδο και την

έξοδο. Τα αντικείμενα χρησιμοποιούνται σαν προκατασκευασμένα τμήματα κώδικα που –μέσα από ένα πρόγραμμα – μπορούν να τα επαναχρησιμοποιηθούν.

Η Visual Basic προσφέρει ορισμένα έτοιμα αντικείμενα όπως το αντικείμενο Console που είδατε στις εφαρμογές κονσόλας ή το αντικείμενο button που είδατε στις εφαρμογές φορμών. Ωστόσο, μπορείτε να χρησιμοποιήσετε και άλλα αντικείμενα που έχει φτιάξει κάποιος άλλος προγραμματιστής ή να φτιάξετε δικά σας.

Τα αντικείμενα διαθέτουν τις δυνατότητές τους μέσα από τις ιδιότητες (properties), τις μεθόδους (methods) και τα συμβάντα (events). Στο φύλλο εργασίας 2 διαχειριστήκατε το συμβάν *click* του αντικειμένου *button* και επίσης αλλάξατε τις τιμές στις ιδιότητες *text* και *enabled*. Στις εφαρμογές κονσόλας μεταχειριστήκατε τις μεθόδους *WriteLine* και *ReadLine* του αντικειμένου *Console*.

Στο προηγούμενο φύλλο εργασίας (ΦΕ4) μάθατε για τις υπορουτίνες (subs) και τις συναρτήσεις (functions) της Visual Basic. Οι μέθοδοι (methods) είναι υπορουτίνες και συναρτήσεις που είναι ενσωματωμένες σε ένα αντικείμενο. Μπορείτε να τις χρησιμοποιήσετε, αλλά δεν μπορείτε να δείτε τον κώδικά τους. Η μέθοδος *WriteLine* γράφει ένα string που της δίνεται σαν παράμετρο στην κονσόλα και μετά αλλάζει γραμμή. Το *Console* αντικείμενο διαθέτει και τη μέθοδο *Write* που γράφει ένα string που της δίνεται σαν παράμετρο στην κονσόλα χωρίς να αλλάζει γραμμή. Η μέθοδος *ReadLine* περιμένει να πληκτρολογήσετε κάτι στην κονσόλα μέχρι να πληκτρολογήσετε enter. Η *ReadLine* συμπεριφέρεται σαν συνάρτηση (function). Αυτά που πληκτρολογεί ο χρήστης τα επιστρέφει με την μορφή string.

Δημιουργήστε μια εφαρμογή κονσόλας και πληκτρολογήστε τον παρακάτω κώδικα:

```
Module Module1

  Sub Main()
    Console.Write("Γράψε το επώνυμό σου:")
    Dim lastName As String = Console.ReadLine()
    Console.WriteLine()
    Console.Write("Γράψε το όνομά σου:")
    Dim firstName As String = Console.ReadLine()
    Console.WriteLine()
```

```
        Console.WriteLine("Λέγεσαι : " + firstName + " " + lastName)
        Console.WriteLine()
        Console.WriteLine()
        Console.Write("Πάτα enter για να τερματιστεί η εφαρμογή")
        Console.ReadLine()
    End Sub

End Module
```

Τρέξτε την εφαρμογή και πληκτρολογήστε επώνυμο και όνομα για να δείτε την απάντηση που θα τυπωθεί στην οθόνη.

Το πρόγραμμα καταχωρεί το input που παίρνει από τον χρήστη στις δυο μεταβλητές: . Ακολούθως, χρησιμοποιεί την αλφαριθμητική πράξη της σύνθεσης (concatenation) για να εμφανίσει στην κονσόλα τη φράση «Λέγεσαι όνομα επώνυμο».

Αντικαταστήστε την εντολή
Console.WriteLine("Λέγεσαι : " + firstName + " " + lastName)
με τις ακόλουθες.

```
        Console.Write("Λέγεσαι : ")
        Console.Write(firstName)
        Console.Write(" ")
        Console.WriteLine(lastName)
```

Τρέξτε την εφαρμογή. Παρατηρήστε πως δεν υπάρχει διαφορά στο μήνυμα που εμφανίζεται στην κονσόλα.

Με όσα έχετε μάθει μέχρι τώρα φτιάξετε μια εφαρμογή κονσόλας με τις παρακάτω προδιαγραφές.

Στόχος της εφαρμογής: ο υπολογισμός του εμβαδού ενός τριγώνου.

Περιγραφή.

Εμφανίζεται το μήνυμα «Δώσε τιμή για βάση».
Ο χρήστης πληκτρολογεί μια τιμή
Εμφανίζεται το μήνυμα «Δώσε τιμή για ύψος»
Ο χρήστης πληκτρολογεί μια τιμή
Εμφανίζεται το μήνυμα «Το εμβαδόν του τριγώνου είναι: » και η τιμή του εμβαδού (που υπολογίζεται από τον τύπο Ε=β*υ/2)

Δραστηριότητα 2: Το αντικείμενο Math

Στο προηγούμενο φύλλο εργασίας (ΦΕ 4) δημιουργήσατε μια συνάρτηση (function) που στρογγυλοποιεί έναν δεκαδικό αριθμό. Η Visual Basic για να κάνει τη ζωή μας πιο εύκολη, ώστε όταν χρειαζόμαστε μια στρογγυλοποίηση να μην αναγκαζόμαστε να γράψουμε μια νέα συνάρτηση, έχει έτοιμη την συνάρτηση *round* καθώς και αρκετές ακόμα συναρτήσεις. Η συνάρτηση αυτή μαζί με όλες τις μαθηματικές συναρτήσεις έχουν ομαδοποιηθεί μέσα στο αντικείμενο Math.

Για παράδειγμα, αν θέλετε να εμφανίσετε στην κονσόλα τη στρογγυλοποίηση του αριθμού 3.14159 πρέπει να γράψετε:

```
Console.WriteLine(Math.Round(3.14159))
```

Άλλες συναρτήσεις που βρίσκονται στο αντικείμενο Math με τη μορφή method είναι:

Abs	Επιστρέφει τη απόλυτη τιμή ενός αριθμού
Ceiling	Στρογγυλοποιεί έναν δεκαδικό προς τα πάνω .π.χ. Math.Ceiling(6.2) επιστρέφει 7.
Floor	Στρογγυλοποιεί έναν δεκαδικό προς τα κάτω. π.χ. Math.Floor(6.8) επιστρέφει 6
Round	Στρογγυλοποιεί έναν δεκαδικό στον πλησιέστερο ακέραιο π.χ. Math. Round (6.8) επιστρέφει 7.ενώ Math. Round (6.2) επιστρέφει 6. Επίσης δέχεται και δεύτερη παράμετρο που προσδιορίζει στα πόσα ψηφία να γίνει η στρογγυλοποίηση.

Max	Επιστρέφει τον μεγαλύτερο από τους δυο αριθμούς που δέχεται σαν παράμετρο.
Min	Επιστρέφει τον μικρότερο από τους δυο αριθμούς που δέχεται σαν παράμετρο
Sqrt	Επιστρέφει την τετραγωνική ρίζα ενός αριθμού.
Cos , Sin,Tan	Τριγωνομετρικές συναρτήσεις (συνημίτονο, ημίτονο, εφαπτομένη)

Επίσης το αντικείμενο Math μας διαθέτει δυο σταθερές (constant) ιδιότητες (properties) που μας δίνουν μαθηματικές σταθερές:

Math.PI. Επιστρέφει τον αριθμό π.
Math.E.

Τροποποιήστε την εφαρμογή που δημιουργήσαμε στη δραστηριότητα 1, ώστε να υπολογίζει το εμβαδόν και την περίμετρο ενός κύκλου παίρνοντας σαν input μόνο την ακτίνα του.

Δραστηριότητα 3: Το String σαν αντικείμενο.

Στο φύλλο εργασίας 2 είδατε πως μπορείτε να δηλώσετε μεταβλητές στη Visual Basic. Οι αλφαριθμητικές μεταβλητές δηλώνονται ως τύπου String. Όταν δημιουργείτε μια μεταβλητή, δημιουργείτε ένα αντικείμενο. Το αντικείμενο αυτό είναι ένα στιγμιότυπο (instance) της κλάσης του. Η κλάση String είναι εξοπλισμένη με μια σειρά ιδιοτήτων και μεθόδων τις οποίες αποκτά αυτόματα κάθε αντικείμενο της κλάσης αυτής που δημιουργείται.

Για παράδειγμα, η ιδιότητα length μας επιστρέφει τον αριθμό των χαρακτήρων που περιέχει ένα string. Δείτε το παρακάτω παράδειγμα.

```
Dim str1 As String
str1 = "παπαδόπουλος" ' τυπώνει τον αριθμό 12
Console.WriteLine(str1.Length)
Dim str2 As String = "Μια φορά και έναν καιρό"
```

```
Console.WriteLine(str2.Length) 'τυπώνει τον αριθμό 23
```

Βλέπουμε πως και οι δυο μεταβλητές που δηλώσαμε διαθέτουν την ιδιότητα length.

Η κλάση String διαθέτει, όπως προαναφέρθηκε, και μια σειρά από μεθόδους. Η μέθοδος *Substring* συντάσσεται με δυο παραμέτρους τύπου integer. Μας επιστρέφει ένα νέο string που περιέχει τους χαρακτήρες από την θέση που δηλώνει η πρώτη παράμετρος και για μήκος όσο δηλώνει η δεύτερη παράμετρος.

Δείτε το παρακάτω παράδειγμα.

```
Dim str As String = "Μια φορά και έναν καιρό"        Dim lexi1 =
str.Substring(0, 3)
Console.WriteLine(lexi1) ' τυπώνει "Μια"
Console.ReadLine()
```

Ο πρώτος χαρακτήρας ενός String βρίσκεται στη θέση 0. Μπορείτε να πάρετε το χαρακτήρα που βρίσκεται σε οποιαδήποτε θέση ενός string με την ακόλουθη σύνταξη: string(θέση). Δείτε το παράδειγμα.

Μια άλλη μέθοδος της κλάσης String είναι η indexOf. Αυτή μας επιστρέφει τη θέση μέσα σε ένα string ενός άλλου string που περνάμε σαν παράμετρο. Δείτε το παράδειγμα.

```
Dim str = "Μια φορά και έναν καιρό"
Console.WriteLine(str.IndexOf("φορά")) ' τυπώνει "4"
```

Υπάρχουν πολλές άλλες μέθοδοι διαθέσιμες από την κλάση string.

Με αυτά που έχετε μάθει μέχρι τώρα δημιουργήστε μια εφαρμογή κονσόλας με τις παρακάτω προδιαγραφές:

Στόχος της εφαρμογής: βρίσκει και τυπώνει την πρώτη λέξη και το μήκος της από μια φράση που πληκτρολογεί ο χρήστης.

Περιγραφή:

Εμφανίζεται το μήνυμα «Πληκτρολόγησε μια φράση ».
Ο χρήστης πληκτρολογεί μια φράση

Εμφανίζεται το μήνυμα «Η πρώτη λέξη της φράσης σου είναι: » και η πρώτη λέξη Εμφανίζεται το μήνυμα «Η λέξη έχει x χαρακτήρα/ες» όπου x ο αριθμός χαρακτήρων της λέξης.

Δραστηριότητα 4: Η κλάση Date.

Μέχρι τώρα μάθατε πώς μπορούμε να δηλώσουμε αριθμητικές, αλφαριθμητικές και λογικές μεταβλητές. Όπως είδαμε οι αλφαριθμητικές μεταβλητές που δηλώνονται ως τύπου string αποτελούν στιγμιότυπα της κλάσης String. Παρακάτω θα μάθετε την κλάση Date. Με την κλάση αυτή μπορείτε να δημιουργήσετε μεταβλητές στις οποίες θα καταχωρείτε ημερομηνίες. Ένα παράδειγμα:

```
Dim dt As Date = Date.Now
Console.WriteLine(dt.ToString()) ' τυπώνει στην οθόνη τη σημερινή ημερομηνία με τη μορφή 22/9/2008 12:47:16 μμ
```

Στο παραπάνω παράδειγμα βλέπετε για πρώτη φορά τη μέθοδο ToString(). Αυτή η μέθοδος υπάρχει σε όλες τις κλάσεις που δίνει η Visual Basic και επιστρέφει πάντα ένα string που συνήθως είναι η τιμή του στιγμιότυπου της κλάσης σε αλφαριθμητική μορφή. Δοκιμάστε τον παρακάτω κώδικα:

Η κλάση Date αποθηκεύει μια ημερομηνία. Από αυτή την ημερομηνία μπορείτε να πάρετε όποιο μέρος θέλετε (π.χ. το έτος) χρησιμοποιώντας τις ιδιότητες και μεθόδους που διαθέτει η κλάση Date. Δείτε το παράδειγμα:

```
Dim dt As Date = Date.Now
Console.WriteLine(dt.Year) ' τυπώνει στην οθόνη το τρέχον έτος
```

Για να καταχωρήσετε μια ημερομηνία σε μια μεταβλητή τύπου date χρησιμοποιείστε τη μέθοδο Parse. Δείτε το παράδειγμα:

```
Dim dt As Date = Date.Parse("9/10/08")
Console.WriteLine(dt.ToString()) ' τυπώνει 9/10/2008 12:00:00 πμ
```

Μπορείτε να κάνετε πράξεις μεταξύ μεταβλητών τύπου date. Ωστόσο, το αποτέλεσμα που παίρνετε δεν είναι πάντα τύπου date. Δείτε το παράδειγμα:

```
Dim dt1 As Date = Date.Parse("9/10/08")
Dim dt2 As Date = Date.Parse("11/10/08")
Dim dtDiff As TimeSpan = dt2 - dt1
Console.WriteLine(dtDiff.Days) ' τυπώνει στην οθόνη 2 δηλαδή 2 ημέρες διαφορά.
```

Η διαφορά μεταξύ δυο μεταβλητών τύπου Date είναι μια μεταβλητή τύπου TimeSpan.

Χρησιμοποιώντας όσα μάθατε μέχρι τώρα, φτιάξτε μια εφαρμογή κονσόλας με τις παρακάτω προδιαγραφές:

Στόχος της εφαρμογής: βρίσκει την ηλικία του χρήστη, αφού ο χρήστης έχει πληκτρολογήσει την ημερομηνία γέννησης

Περιγραφή:

Εμφανίζεται το μήνυμα «Πληκτρολόγησε την ημερομηνία γέννησης με τη μορφή μέρα/μήνας/έτος ».
Ο χρήστης πληκτρολογεί μια ημερομηνία
Εμφανίζεται το μήνυμα «Με βάση τα στοιχεία που έδωσες σήμερα είσαι x ετών» όπου x η ηλικία του χρήστη σε έτη.

Φύλλο εργασίας

6. Λογικές συναρτήσεις και παραστάσεις η ροή του προγράμματος

Όνομα:

Τάξη:

Διάρκεια: 3 διδακτικές ώρες.

Διδακτικοί στόχοι:

Με το συγκεκριμένο φύλλο εργασίας

- Θα μάθεις τις εντολές διακλάδωσης If .. then ... else
- Θα μάθεις τις λογικές πράξεις AND και OR και πως αυτές συνδυάζονται με τις εντολές διακλάδωσης
- Θα μάθεις τις πιο σύνθετες εντολές διακλάδωσης If .. then ... elseIf και select .. case

Δραστηριότητα 1 : η εντολή If .. then ... else

Μέχρι τώρα μάθατε να γράφετε εντολές που το πρόγραμμα εκτελεί σειριακά (την μια μετά την άλλη). Ωστόσο, πολύ συχνά χρειαζόμαστε το πρόγραμμα μας να εκτελεί ορισμένες εντολές μόνο όταν ισχύουν συγκεκριμένες συνθήκες. Σε αυτή την περίπτωση πρέπει να ελέγξουμε τη συνθήκη και να δούμε αν ισχύει και εφόσον ισχύει να εκτελεστεί η κατάλληλη ομάδα των εντολών. Στη Visual Basic ο έλεγχος γίνεται με την εντολή If Then που συντάσσεται ως εξής:

If συνθήκη Then
εντολές
End if

Το παρακάτω παράδειγμα είναι μια συνάρτηση που επιστρέφει την απόλυτη τιμή ενός ακεραίου που της περνάμε σαν παράμετρο.

```
Function abs(ByVal x As Integer)
```

```
    If x < 0 Then
        x = x * -1
    End If
    Return x
End Function
```

Στο παραπάνω παράδειγμα το πρόγραμμα ελέγχει την τιμή της παραμέτρου x. Αν η παράμετρος είναι θετικός αριθμός πρέπει να επιστρέψει τον ίδιο. Αν όμως είναι αρνητικός αριθμός θα πρέπει να επιστρέψει τον αντίθετο. Με την εντολή If Then ελέγχουμε αν ο αριθμός είναι μικρότερος από το μηδέν (δηλαδή αρνητικός). Μόνο σε αυτή την περίπτωση, αν ισχύει δηλαδή η συνθήκη x < 0, εκτελείται η εντολή καταχώρησης x = x * -1. Οι εντολές που βρίσκονται μετά το *If x < 0 Then* δεν εκτελούνται, παρά μόνο όταν η συνθήκη είναι αληθής(true).
Πολλές φορές χρειάζεται το πρόγραμμά μας να κάνει μια ενέργεια αν ισχύει μια συνθήκη και μια διαφορετική στην περίπτωση που η συνθήκη αυτή δεν ισχύει. Σε αυτή την περίπτωση χρησιμοποιούμε την εντολή if Then Else. Αυτή συντάσσεται ως εξής:

If συνθήκη Then
εντολές
Else
εντολές
End if

Οι εντολές που βρίσκονται μετά το *If συνθήκη Then* εκτελούνται μόνον όταν η συνθήκη είναι αληθής (true), ενώ οι εντολές που βρίσκονται μετά το *else* εκτελούνται μόνον όταν η συνθήκη είναι ψευδής (false).

Το παρακάτω παράδειγμα είναι μια εφαρμογή κονσόλας που ανάλογα με τη ώρα της ημέρας τυπώνει «καλημέρα» ή «καλησπέρα».

```
Module Module1

    Sub Main()
        Dim dt As Date = Date.Now
        Debug.Print(dt.ToString())
```

```
        If dt.Hour < 12 Then
            Console.WriteLine("Καλημέρα")
        Else
            Console.WriteLine("Καλησπέρα")
        End If
        Console.ReadLine()
    End Sub

End Module
```

Η εφαρμογή εξετάζει την ώρα και αν είναι μικρότερη από δώδεκα τυπώνει «Καλημέρα» σε οποιαδήποτε άλλη περίπτωση τυπώνει «Καλησπέρα».

Τώρα δημιουργήστε την παρακάτω εφαρμογή.

Στόχος της εφαρμογής: Η εφαρμογή τυπώνει αν κυκλοφορούν σήμερα τα μόνα ή τα ζυγά αυτοκίνητα.

Περιγραφή:
Η εφαρμογή παίρνει τη σημερινή ημερομηνία.
Εξετάζει την ημέρα του μήνα (από την ιδιότητα *Day*) και αν αυτή διαιρείται ακριβώς με το 2 εμφανίζει το μήνυμα «Σήμερα κυκλοφορούν τα ζυγά» αλλιώς εμφανίζει το μήνυμα «Σήμερα κυκλοφορούν τα μονά.».
Χρησιμοποιείτε τον τελεστή *mod* για να υπολογίσετε το υπόλοιπο της διαίρεσης της ημερομηνίας με το 2.

Δραστηριότητα 2: nested if

Όπως αναφέρθηκε παραπάνω ανάμεσα στις εντολές *if Then* και *End If* μπαίνουν οι εντολές θα εκτελεστούν αν ισχύει η συνθήκη. Οι εντολές αυτές μπορεί να είναι οτιδήποτε, ακόμα και ένα νέο *If*. Σε αυτή την περίπτωση (που υπάρχει *if* μέσα σε *if*) αποκαλείται nested if.

Υλοποιήστε μια εφαρμογή κονσόλας που θα ζητάει τον λήγοντα από τον αριθμό κυκλοφορίας του χρήστη και αν ο χρήστης δεν μπορεί να κυκλοφορήσει στο κέντρο θα τυπώνει το ανάλογο μήνυμα.

Υλοποιήστε την εφαρμογή βασιζόμενοι στον παρακάτω ψευδοκώδικα που χρησιμοποιεί nested if.

- Ο χρήστης πληκτρολογεί το λήγοντα από την πινακίδα του αυτοκινήτου του.
- Το πρόγραμμα παίρνει τη σημερινή ημερομηνία.
- Εάν σήμερα δεν είναι Κυριακή (η ημέρα της εβδομάδας δίνεται από την ιδιότητα *DayOfWeek* του αντικειμένου *Date*. Η Δευτέρα είναι το 1 και η Κυριακή το 7).
- Εάν σήμερα δεν είναι Σάββατο:
    - Καταχώρησε στη μεταβλητή *pinakidaYpoloipo* το υπόλοιπο της διαίρεσης του λήγοντα της πινακίδας του χρήστη με το δύο.
    - Καταχώρησε στη μεταβλητή *meraYpoloipo* το υπόλοιπο της διαίρεσης της ημέρας του μήνα (day) με το δύο.
    - Εάν οι μεταβλητές *pinakidaYpoloipo* και *meraYpoloipo* δεν είναι ίσες, τότε τύπωσε «Σήμερα αν κατέβεις στο κέντρο, θα σε γράψουν».

Δραστηριότητα 3: Οι εντολές AND , OR, NOT

Στο προηγούμενο παράδειγμα το μήνυμα εμφανίζεται μόνο όταν ισχύον μια σειρά από συνθήκες. Για να το υλοποιήσουμε, εφαρμόσαμε την τεχνική προγραμματισμού με τα nested if. Σε αυτή τη δραστηριότητα θα την υλοποιήσετε χρησιμοποιώντας τον λογικό τελεστή *AND*.

Οι λογικοί τελεστές εφαρμόζονται μεταξύ λογικών μεταβλητών ή σταθερών. Στη Δραστηριότητα 1 χρησιμοποιήσατε την εντολή:

```
If dt.Hour < 12 Then
```

Αυτή η εντολή μπορεί να ξαναγράφει ως εξής:

```
Dim isProi As Boolean = dt.Hour < 12
If isProi Then
```

Η μεταβλητή *isProi* είναι μια λογική μεταβλητή και σε αυτή μπορείτε να χρησιμοποιήσετε λογικούς τελεστές.

Αν θέλετε να δημιουργήσετε μια σύνθετη συνθήκη μπορείτε να τη δημιουργήσετε από απλές συνθήκες που θα συνδεθούν μεταξύ τους με λογικούς τελεστές. Δείτε την εφαρμογή που φτιάξατε στη δραστηριότητα 1 πώς υλοποιείται με τη χρήση του τελεστή *AND*.

```
Module Module1


  Sub Main()
    Console.Write("δωσε τον λήγοντα από την πινακίδα του αυτοκινήτου σου")
    Dim pinakida As Integer = Console.ReadLine()
    Dim dt As Date = Date.Now
    Dim mera As Integer = dt.DayOfWeek
    Dim isNotSunday As Boolean = mera <> 7
    Dim isNotSaturday As Boolean = mera <> 6
    Dim pinakidaYpoloipo As Integer = pinakida Mod 2
    Dim meraYpoloipo As Integer = dt.Day Mod 2
    Dim isWronDayForMyCar As Boolean = pinakidaYpoloipo <> meraYpoloipo
    If isNotSunday And isNotSaturday And isWronDayForMyCar Then
      Console.WriteLine("Σήμερα αν κατέβεις στο κέντρο, θα σε γράψουν")
    Else
      Console.WriteLine("Μπορείς να πας στο κέντρο σήμερα")
    End If
    Console.ReadLine()


  End Sub


End Module
```

Παρατηρήστε πως δηλώνονται οι λογικές μεταβλητές για την αποθήκευση τις τιμές των συνθηκών (π.χ. *Dim isNotSunday As Boolean = mera <> 7*).

Στη συνέχεια συνδέστε τις λογικές μεταβλητές με τον τελεστή *AND*, έτσι ώστε να φτιάξετε μια σύνθετη συνθήκη.

```
If isNotSunday And isNotSaturday And isWronDayForMyCar Then
```

Θα μπορούσαμε να καταχωρήσουμε το αποτέλεσμα της συνθήκης αυτής σε μια νέα μεταβλητή π.χ.

```
    Dim denKikloforo As Boolean = isNotSunday And isNotSaturday And isWronDayForMyCar
    If denKikloforo Then
```

Η σύνθετη συνθήκη που δημιουργήσατε με τον τελεστή *AND* είναι αληθής μόνο όταν όλες οι λογικές μεταβλητές που τη συνθέτουν είναι αληθείς. Αν έστω και μια είναι ψευδής, τότε ολόκληρη η συνθήκη είναι ψευδής (false). Στις λογικές πράξεις όπως και στις αριθμητικές μπορούν να χρησιμοποιηθούν παρενθέσεις για να οριστεί η σειρά των πράξεων, π.χ.

```
Dim denKikloforo As Boolean = (isNotSunday And isNotSaturday) And isWronDayForMyCar
```

Ο τελεστής *OR* δίνει τιμή αληθή (true) αν έστω και μια από τις συνθήκες που αποτελούν μια σύνθετη συνθήκη είναι αληθής, ενώ δίνει τιμή ψευδή (false), αν όλες οι συνθήκες που αποτελούν μια σύνθετη συνθήκη είναι ψευδείς.

Δείτε το παρακάτω παράδειγμα όπου χρησιμοποιείται ο τελεστής *OR*.

```
Module Module1

  Sub Main()
    Dim dt As Date = Date.Now
    Dim mera As Integer = dt.DayOfWeek
    Dim isSunday As Boolean = (mera = 7)
    Dim isSaturday As Boolean = (mera = 6)
    If isSaturday Or isSunday Then
```

```
        Console.WriteLine("Σήμερα κάνεις ότι θες")
    Else
        Console.WriteLine("Σήμερα πρέπει να πας στη δουλειά")
    End If
    Console.ReadLine()

  End Sub

End Module
```

Με όσα μάθατε μέχρι τώρα, δημιουργήστε μια εφαρμογή κονσόλας με τις παρακάτω προδιαγραφές.

Στόχος της εφαρμογής: Ελέγχει αν ένα έτος είναι δίσεκτο χρησιμοποιώντας τα *AND* και *OR*.

Περιγραφή:
Η εφαρμογή ζητάει από το χρήστη να πληκτρολογήσει έναν αριθμό που αντιπροσωπεύει ένα έτος (ακέραιος τετραψήφιος αριθμός).
Στη συνέχεια ελέγχει αν το έτος αυτό είναι δίσεκτο.
Σημείωση: Δίσεκτα είναι τα έτη που διαιρούνται με το 4 και δεν διαιρούνται με το 100, εκτός αν διαιρούνται με το 400, π.χ το 1996 είναι δίσεκτο, το 2000 είναι δίσεκτο, αλλά το 1900 δεν είναι δίσεκτο.
Τέλος εμφανίζει ένα αντίστοιχο μήνυμα «Το έτος x είναι/ δεν είναι δίσεκτο».
Χρησιμοποιήστε τους τελεστές *and* και *or* για να δημιουργήσετε μια συνθήκη που να αληθεύει αν το έτος είναι δίσεκτο.

Δραστηριότητα 4: Το σύνθετο σχήμα If…Then…ElseIf

Η εντολή *If…Then…ElseIf* είναι μία άλλη μορφή συνδυασμού πολλών *If…Then*. Η σύνταξή της έχει ως εξής.

If συνθήκη Then
εντολές
ElseIf συνθήκη Then
εντολές
ElseIf συνθήκη Then
εντολές
Else
εντολές
End if

Το πρόγραμμα ελέγχει την πρώτη συνθήκη (μετά το *If*), αν είναι αληθής, εκτελεί τις εντολές μετά το πρώτο *Then*, αν όχι ελέγχει τη συνθήκη μετά το *ElseIf*, αν είναι αληθής εκτελεί τις εντολές μετά το *Then* , αν όχι συνεχίζει μέχρι να βρει μια συνθήκη αληθή. Αν καμία δε είναι αληθής, τότε εκτελούνται οι εντολές μετά το *Else*.

Η συγκεκριμένη σύνταξη μπορεί να γίνει πιο κατανοητή με το παράδειγμα του δίσεκτου έτους και με την υλοποίηση που προτείνεται παρακάτω.

Υλοποιήστε την παρακάτω εφαρμογή χρησιμοποιώντας τον ψευδοκώδικα που δίνεται πιο κάτω για να υπολογίσετε αν ένα έτος είναι δίσεκτο ή όχι.

Στόχος της εφαρμογής: Ελέγχει αν ένα έτος είναι δίσεκτο χρησιμοποιώντας το *If ..Then .. ElseIf.*

Περιγραφή:
Η εφαρμογή ζητάει από το χρήστη να πληκτρολογήσει έναν αριθμό που αντιπροσωπεύει ένα έτος. Ο αριθμός αποθηκεύεται στη μεταβλητή theYear.
Στη συνέχεια ελέγχει αν το έτος αυτό είναι δίσεκτο με τον τρόπο που ακολουθεί.

-Αποθήκευσε στη μεταβλητή *div4* το αποτέλεσμα της συνθήκης «το υπόλοιπο του theYear αν διαιρεθεί με το 4 είναι 0»

-Αποθήκευσε στη μεταβλητή *div100* το αποτέλεσμα της συνθήκης «το υπόλοιπο του theYear αν διαιρεθεί με το 100 είναι 0»

-Αποθήκευσε στη μεταβλητή *div400* το αποτέλεσμα της συνθήκης «το υπόλοιπο του theYear αν διαιρεθεί με το 400 είναι 0»

-Δήλωσε την μεταβλητή *isLeapYear*.

-Εάν ισχύει *div400* τότε κάνε το *isLeapYear* αληθές

-Αλλιώς εάν ισχύει *div100* τότε κάνε το *isLeapYear* ψευδές

-Αλλιώς εάν ισχύει *div4* τότε κάνε το *isLeapYear* αληθές

-Αλλιώς *isLeapYear* ψευδές

-Αν ισχύει *isLeapYear* τότε εμφάνισε «το έτος είναι δίσεκτο, αλλιώς εμφάνισε «το έτος δεν είναι δίσεκτο».

Δραστηριότητα 5: Δομή Select Case

Μια άλλη δομή που διευκολύνει τη συγγραφή κώδικα είναι η εντολή *select ..case*. Τη δομή αυτή τη χρησιμοποιούμε όταν χρειάζεται να συγκριθεί η τιμή μιας μεταβλητής έναντι πολλών τιμών. Αν, για παράδειγμα, θέλετε να τυπώσετε την μέρα της εβδομάδας θα μπορούσατε να χρησιμοποιήσετε τον παρακάτω κώδικα:

```
Dim dt As Date = Date.Now
Dim dayOfWeek As Integer = dt.DayOfWeek
Select Case dayOfWeek
   Case 1
      Console.WriteLine("Σήμερα ειναι Δευτέρα")
   Case 2
      Console.WriteLine("Σήμερα ειναι Τρίτη")
   Case 3
      Console.WriteLine("Σήμερα ειναι Τετάρτη")
   Case 4
      Console.WriteLine("Σήμερα ειναι Πέμπτη")
   Case 5
      Console.WriteLine("Σήμερα ειναι Παρσκευή")
```

```
    Case 6, 7
        Console.WriteLine("Σαββατοκύριακο, μπορείς να ξεκουραστεις")
End Select
Console.ReadLine()
```

Στο παραπάνω παράδειγμα συγκρίνονται διαδοχικά, η τιμή της μεταβλητής *dayOfWeek* με τους αριθμούς 1,2,3,4,5,(6 και 7). Βλέπετε ότι μπορείτε να συγκρίνετε σε μια *case* με περισσότερους από έναν αριθμούς. Επίσης, μπορείτε να βάλετε ένα εύρος. Π.χ. *Case 1 To 5*

Μπορείτε να γράψετε ένα πρόγραμμα με πολλούς τρόπους. Τώρα υλοποιήστε τον παραπάνω κώδικα χωρίς να χρησιμοποιήσετε τη δομή *select ..case*. Αντί για αυτή χρησιμοποιήστε τη δομή *If ..Then .. ElseIf*

Φύλλο εργασίας

7. Πίνακες και Επαναληπτικές Δομές (3ωρες)

Όνομα:

Τάξη:

Διάρκεια: 3 διδακτικές ώρες.

Διδακτικοί στόχοι:

Με το συγκεκριμένο φύλλο εργασίας

- Θα μάθεις να χρησιμοποιείς τις δομές επανάληψης while, repeat, for.
- Θα μάθεις τη διαφοροποίηση ανάμεσα στις δομές repeat και while
- Θα μπορείς να δημιουργείς πίνακες στη Visual Basic
- Θα μπορείς να προσπελάσεις όλα τα στοιχεία ενός πίνακα με τη χρήση της δομής for.

Δραστηριότητα 1 : Η δομή repeat

Ένα από τα πλεονεκτήματα των υπολογιστών είναι πως μπορούν να εκτελούν πολλές επαναλήψεις μιας εργασίας. Ο αριθμός των επαναλήψεων στον κώδικα καθορίζεται από τη χρήση των δομών επανάληψης.

Υπάρχουν τρεις δομές επανάληψης:

-Η δομή «while» που έχει τη γενική μορφή «όσο ισχύει η συνθήκη ελέγχου εκτέλεσε τις εντολές».

-Η δομή «repeat» που έχει τη γενική μορφή «εκτέλεσε τις εντολές και συνέχισε να τις εκτελείς όσο ισχύει η συνθήκη ελέγχου».

-Η δομή «for» που έχει τη γενική μορφή «εκτέλεσε τις εντολές για ορισμένο αριθμό επαναλήψεων».

-Η διαφορά των δομών «while» και «repeat» είναι πως ενώ στη δομή «while» μπορεί να μην εκτελεστεί καμία επανάληψη, στη δομή «repeat» εκτελείται τουλάχιστον μια επανάληψη.

Στην πρώτη δραστηριότητα θα δούμε τη δομή «repeat» χρησιμοποιώντας τη εφαρμογή υπολογισμού του δίσεκτου έτους. Όπως είχαμε δημιουργήσει την εφαρμογή αυτή ο χρήστης έδινε ένα έτος και το πρόγραμμα υπολόγιζε αν αυτό είναι δίσεκτο. Για να υπολογίσει ένα άλλο έτος ο χρήστης θα πρέπει να ξαναεκτελέσει το πρόγραμμα. Παρακάτω θα δείτε τον κώδικα τροποποιημένο ώστε να υπολογίζει όσα έτη δίνει ο χρήστης μέχρι αυτός να πατήσει <Enter> χωρίς να πληκτρολογήσει τίποτε άλλο.

Η δομή «repeat» ονομάζεται έτσι γιατί σε παλιότερες γλώσσες προγραμματισμού υλοποιούνταν με την εντολή *repeat*. Στην Visual Basic υλοποιείται με την εντολή *Do … Loop until* ή *Do .. loop While.*

Δείτε το κώδικα:

```
Module Module1

    Sub Main()
        Console.Write("Δώσε έτος:")
        Dim strTheYear As String = Console.ReadLine()
        Dim theYear As Integer
        Do
            theYear = strTheYear
            Dim div4 As Boolean = (theYear Mod 4) = 0
            Dim div100 As Boolean = (theYear Mod 100) = 0
            Dim div400 As Boolean = (theYear Mod 400) = 0

            Dim isLeapYear As Boolean
            If div400 Then
                isLeapYear = True
            ElseIf div100 Then
                isLeapYear = False
            ElseIf div4 Then
                isLeapYear = True
            Else
                isLeapYear = False
            End If
            If isLeapYear Then
                Console.WriteLine("Δισεκτο")
            Else
                Console.WriteLine("δεν ειναι Δισεκτο")
```

```
            End If
            Console.Write("Δώσε έτος:")
            strTheYear = Console.ReadLine()
        Loop Until strTheYear = ""
    End Sub

End Module
```

Επειδή ο κώδικας αυτός είναι πολύ μακρύς, για την καλύτερη κατανόησή του από τον ίδιο αλλά και από κάποιον άλλο προγραμματιστή που θα τον διαβάσει αργότερα, θα πρέπει να δομηθεί απομονώνοντας αυτόνομα κομμάτια του και ομαδοποιώντας τα σε υπορουτίνες και συναρτήσεις.

Δείτε το παράδειγμα:

```
Module Module1
    Function isThisAleapyear(ByVal theYear As Integer) As Boolean
        Dim div4 As Boolean = (theYear Mod 4) = 0
        Dim div100 As Boolean = (theYear Mod 100) = 0
        Dim div400 As Boolean = (theYear Mod 400) = 0
        Dim isLeapYear As Boolean
        If div400 Then
            isLeapYear = True
        ElseIf div100 Then
            isLeapYear = False
        ElseIf div4 Then
            isLeapYear = True
        Else
            isLeapYear = False
        End If
        Return isLeapYear
    End Function

    Sub Main()
        Console.Write("Δώσε έτος:")
        Dim strTheYear As String = Console.ReadLine()
        Dim theYear As Integer
        Do
            theYear = strTheYear
            If isThisAleapyear(theYear) Then
                Console.WriteLine("Το έτος " + theYear.ToString() + "
είναι δίσεκτο.")
            Else
                Console.WriteLine("Το έτος " + theYear.ToString() + "
δεν είναι δίσεκτο, μπορείς να παντρευτείς.")
            End If
            Console.Write("Δώσε έτος:")
            strTheYear = Console.ReadLine()
        Loop Until strTheYear = ""
    End Sub

End Module
```

Πριν εξετάσουμε τη δομή επανάληψης δείτε μερικά σημεία στον κώδικα.

Το *theYear.ToString()*

όπως είδαμε στο φύλλο εργασίας 5 η Visual Basic αντιμετωπίζει τα String σαν αντικείμενα. Το ίδιο συμβαίνει και με τις μεταβλητές τύπου Integer. Η Visual Basic τις αντιμετωπίζει σαν αντικείμενα. Η μέθοδος *ToString* επιστρέφει ένα string που αναπαριστά τον αριθμό. Έτσι μπορούμε να το χρησιμοποιήσουμε σε αλφαριθμητικές πράξεις.

Το *theYear = strTheYear*

Εδώ έχουμε μια ανορθόδοξη καταχώρηση. Καταχωρούμε ένα string σε έναν Integer. Σε αυτή την περίπτωση η Visual Basic προσπαθεί να μετατρέψει την τιμή που βρίσκεται στο δεξιό μέρος της καταχώρισης στον τύπο μεταβλητής που βρίσκεται στο αριστερό μέρος. Αν αυτό δεν είναι δυνατόν το πρόγραμμά μας θα «σκάσει», δηλαδή θα βγάλει ένα μήνυμα λάθους και θα σταματήσει τη λειτουργία του.

Do και *Loop Until*

Οι εντολές που επαναλαμβάνονται είναι οι εντολές που βρίσκονται μεταξύ των *Do* και *Loop Until*. Μετά το *Loop Until* ακλουθεί μια λογική συνθήκη. Όταν φτάσει εκεί το πρόγραμμα θα εξετάσει τη συνθήκη. Αν αυτή είναι ψευδής τότε θα επαναλάβει την εκτέλεση των εντολών από την εντολή *Do* και μετά. Αν είναι αληθής θα βγει από τη δομή επανάληψης και θα εκτελεστεί η αμέσως επόμενη εντολή, στην περίπτωση μας η εντολή *End Sub* που τερματιζει την εφαρμογή.

strTheYear = Console.ReadLine()

Επειδή οι εντολές που βρίσκονται μέσα στη δομή επανάληψης θα εκτελούνται μέχρι η συνθήκη να είναι αληθής οι παράγοντες που επηρεάζουν τη συνθήκη (μπορούν δηλαδή να της αλλάξουν τιμή) πρέπει να ανανεώνονται μέσα στη δομή επανάληψης. Στην περίπτωσή μας αυτό γίνεται με την τελευταία εντολή πριν από το *Loop Until*: *strTheYear = Console.ReadLine()*. Αν η συνθήκη δεν ανανεωθεί το πρόγραμμα θα πέσει σε ατέρμονη επανάληψη και δεν θα μπορεί να τερματιστεί φυσιολογικά.

Τώρα δημιουργήστε την παρακάτω εφαρμογή κονσόλας.

Στόχος της εφαρμογής: ζητάει από το χρήστη να πληκτρολογήσει μια λέξη μέχρι ο χρήστης να πληκτρολογήσει τη λέξη «bye», με αυτή τη λέξη τερματίζεται η εφαρμογή.

Δραστηριότητα 2 : Η δομή while.

Στις επόμενες δραστηριότητες θα δείτε μια εφαρμογή που χρησιμοποιεί και τις τρεις δομές επανάληψης. Σε αυτή τη δραστηριότητα θα ασχοληθείτε με τη δομή «while». Όπως αναφέρθηκε πιο πάνω με τη δομή «while» πρώτα γίνεται ο έλεγχος αν ισχύει η συνθήκη που ελέχγει τις επαναλήψεις και μετά εκτελούνται οι επαναλήψεις. Αυτό σημαίνει πως είναι πιθανόν να μην εκτελεστεί καμία επανάληψη. Στον παρακάτω κώδικα έχουμε μια εφαρμογή που υπολογίζει το παραγοντικό ενός αριθμού που πληκτρολογεί ο χρήστης. Η διαδικασία επαναλαμβάνεται μέχρι ο χρήστης να πληκτρολογήσει το αριθμό 0. Έχει μια μεγάλη διαφοροποίηση από τις εφαρμογές που είδατε μέχρι τώρα, κάνει έλεγχο στα δεδομένα που πληκτρολογεί ο χρήστης έτσι ώστε αν αυτός πληκτρολογήσει μη αποδεκτά δεδομένα το πρόγραμμα να συνεχίσει τη λειτουργία του. Δείτε τον κώδικα:

```
Module Module1

    Function calculateParagontiko(ByVal number As Integer) As String
        Return "Η συνάρτηση αυτή δεν είναι ακόμα έτοιμη"
    End Function

    Function getUserInput() As Integer
        Dim strTheNumber As String
        Dim theNumber As Integer
        Do
            Console.Write("Δωσε έναν αριθμό")
            Console.Write("0 για τερματισμό)")
            strTheNumber = Console.ReadLine()
        Loop Until IsNumeric(strTheNumber)
        theNumber = strTheNumber
        Return theNumber
    End Function

    Sub Main()
        Dim n As Integer
        n = getUserInput()
        Do While n <> 0
            Console.Write("Το παραγοντικό του " + n.ToString() + "
είναι ")
            Console.WriteLine(calculateParagontiko(n))
            n = getUserInput()
        Loop
    End Sub

End Module
```

Για να κατανοήσετε καλύτερα τη λειτουργία του προγράμματος δημιουργήστε μια νέα εφαρμογή κονσόλας και πληκτρολογήστε το παραπάνω. Μετά πάρτε το ρόλο του χρήστη και εισάγετε διάφορες τιμές αριθμητικές και μη.

Η εφαρμογή αποτελείται από την κεντρική υπορουτίνα και δυο συναρτήσεις. Η μια συνάρτηση *calculateParagontiko()* δεν είναι ακόμα έτοιμη και θα την δείτε στην επόμενη δραστηριότητα. Η άλλη *getUserInput()* επιστρέφει τον αριθμό που πληκτρολόγησε ο χρήστης αφού κάνει πρώτα έλεγχο πως ο χρήστης πληκτρολόγησε πράγματι έναν αριθμό. Ο έλεγχος γίνεται με την συνάρτηση *IsNumeric* της Visual Basic. Η συνάρτηση αυτή είναι μια λογική(Boolean) συνάρτηση, επιστρέφει δηλαδή τιμές *true* ή *false*. Η τιμή της γίνεται true όταν το string που δέχεται σαν παράμετρο είναι μετατρέψιμο σε αριθμό.

Η κεντρική υπορουτίνα του προγράμματος *Sub Main()* χρησιμοποιεί την δομή «while» για να επαναλαμβάνει την διαδικασία μέχρι ο χρήστης να πληκτρολογήσει την τιμή 0. Παρατηρήστε πως αν ο χρήστης πληκτρολογήσει την πρώτη φορά το 0 η συνάρτηση *calculateParagontiko* δεν εκτελειται ποτέ.

Η δομή while συντάσσεται ως εξής:

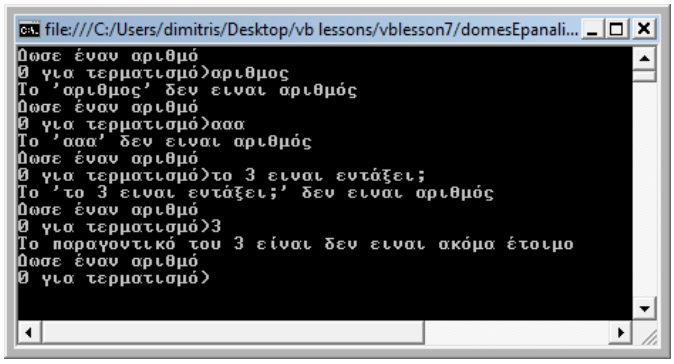
Do while συνθήκη
εντολές
Loop

Συγκρίνετε την εφαρμογή αυτή με την εφαρμογή που είδατε σαν παράδειγμα στη δραστηριότητα 1. Τι συμβαίνει εκεί αν ο χρήστης πατήσει για πρώτη φορά <Enter>;

Τώρα εξετάστε την συνάρτηση *getUserInput()*. Σε αυτή χρησιμοποιούμε τη δομή repeat για να εξασφαλίσουμε πως ο χρήστης έχει πληκτρολογήσει ένα string που μπορεί να μετατραπεί σε αριθμό.

Εξετάστε τη δομή της συνάρτησης και μετατρέψετε την ως εξής: όταν ο χρήστης πληκτρολογήσει κάτι που δεν είναι αριθμός τότε πρέπει να εμφανίζεται ένα μήνυμα που να λέει «το x δεν είναι αριθμός» οπού x το string που πληκτρολόγησε ο χρήστης.

Η κονσόλα όπου εκτελείται η εφαρμογή θα πρέπει να μοιάζει με το παρακάτω:



Σκεφτείτε η δομή «repeat» είναι η κατάλληλη για να υλοποιήσετε τη ζητούμενη συνάρτηση;

Δραστηριότητα 3 : Η δομή for .. next

Η τρίτη δομή επανάληψης είναι η δομή «for .. next». Η δομή αυτή εκτελεί ένα πακέτο εντολών για ένα συγκεκριμένο αριθμό επαναλήψεων. Δεν έχει συνθήκη ελέγχου για να τερματιστεί όπως οι δομές «repeat» και «while». Αντίθετα χαρακτηριστικό της είναι ένας ακέραιος που αποκαλείται δείκτης. Ο δείκτης έχει μια αρχική τιμή και μια τελική τιμή. Σε κάθε επανάληψη η τιμή του δείκτη αυξάνεται κατά 1 (ή κάποιον άλλον αριθμό όπως θα δούμε παρακάτω). Οι επαναλήψεις τερματίζονται όταν η τιμή του δείκτη γίνει μεγαλύτερη από την τελική τιμή.

Η σύνταξη της δομής « for .. next» είναι η εξής:

For i = αρχική τιμή *To* τελική τιμή
εντολές
Next

Για να κατανοήσετε καλλίτερα τη δομή «for .. next» φτιάξτε την παρακάτω εφαρμογή κονσόλας.

```
Module Module1

    Sub Main()
        For i = 0 To 10
            Console.WriteLine(i)
```

```
        Next
        Console.ReadLine()
    End Sub

End Module
```

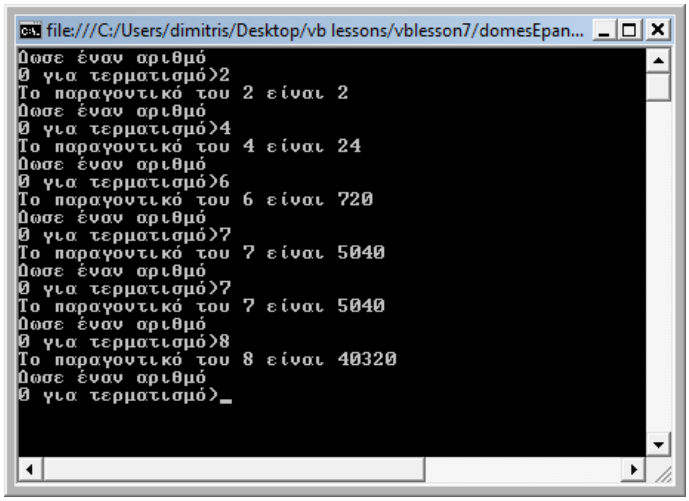
Η εφαρμογή αυτή τυπώνει στην οθόνη τους αριθμούς από το 0 μέχρι το 10, δηλαδή ένδεκα αριθμούς. Βλέπετε πως ο δείκτης αλλάζει τιμή μετά από κάθε επανάληψη. Ο δείκτης μπορεί να αλλάζει και με τιμές διαφορετικές του 1. Η ρυθμός μεταβολής του δείκτη ορίζεται με τη δήλωση *Step*. Ο παρακάτω κώδικας τυπώνει του άρτιους αριθμούς από το 0 μέχρι το 10.

```
Module Module1

    Sub Main()
        For i = 0 To 10 Step 2
            Console.WriteLine(i)
        Next
        Console.ReadLine()
    End Sub

End Module
```

Η παρακάτω εφαρμογή είναι μια απλή εφαρμογή που υπολογίζει το άθροισμα των αριθμών από το 1 έως το 100.

```
Module Module1

    Sub Main()
        Dim sum As Integer = 0
        For i = 0 To 100
            sum = sum + i
        Next
        Console.WriteLine(sum)
        Console.ReadLine()
    End Sub

End Module
```

Τώρα επιστρέψτε στην εφαρμογή που φτιάξατε στη δραστηριότητα 2 και συμπληρώστε την συνάρτηση που επιστρέφει το παραγοντικό ενός αριθμού. Η τελική οθόνη πρέπει να μοιάζει με το παρακάτω.

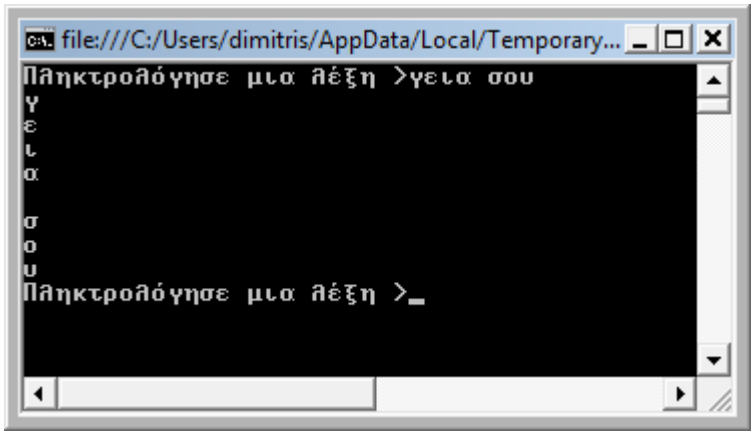


Δραστηριότητα 4 : Οι πίνακες στη Visual Basic

Όταν έχουμε πολλές όμοιες τιμές είναι καλό να μπορούμε να τις ομαδοποιούμε. Οι γλώσσες προγραμματισμού γι' αυτό τον σκοπό διαθέτουν τους πινάκες. Οι πίνακες είναι μεταβλητές που μέσα τους καταχωρούνται άλλες μεταβλητές. Για να έχουμε πρόσβαση σε μια από τις μεταβλητές που βρίσκονται μέσα σε έναν πίνακα χρησιμοποιούμε δείκτες.

Ένα παράδειγμα πίνακα που γνωρίζετε πολύ καλά ήδη είναι η μεταβλητή τύπου string. Αυτή η μεταβλητή αποτελείται από πολλές μεταβλητές τύπου char. Δείτε στο παρακάτω παράδειγμα πως μπορούμε να χειριστούμε ένα string σαν πίνακα.

```
Module Module1

    Sub Main()
        Console.Write("Πληκτρολόγησε μια λέξη >")
        Dim str As String = Console.ReadLine()
        Dim ch As Char
        Do While str <> ""
            For i = 0 To str.Length - 1
                ch = str(i)
                Console.WriteLine(ch)
            Next
            Console.Write("Πληκτρολόγησε μια λέξη >")
            str = Console.ReadLine()
        Loop
    End Sub

End Module
```



Στο παράδειγμα αυτό ο χρήστης πληκτρολογεί μια σειρά χαρακτήρων στο πληκτρολόγιο και μετά το πρόγραμμα εκτυπώνει ένα χαρακτήρα σε κάθε γραμμή. Η τελική τιμή της δομής «for» καθορίζεται από την ιδιότητα *Length* του string. Η ιδιότητα *Length* υπάρχει σε όλους του πίνακες και επιστρέφει τον αριθμό των μεταβλητών που έχουν αποθηκευτεί μέσα στον πίνακα. Οι μεταβλητές αποθηκεύονται στο πίνακα σε σειρά. Για να προσπελάσουμε μια μεταβλητή που είναι αποθηκευμένη σε έναν πίνακα χρειαζόμαστε να ξέρουμε την θέση της. Η προσπέλαση γίνεται με έναν δείκτη. Ο δείκτης ξεκινάει από τη τιμή μηδέν. Έτσι αν έχουμε ένα string με τιμή «Μια γάτα» μπορούμε να προσπελάσουμε το «Μ» βάζοντας για δείκτη την τιμή μηδέν. Έτσι ο κώδικας

```
Dim str As String = "Μια γάτα"
Console.Write(str(0))
```

θα τυπώσει το «Μ»

Το string όπως αναφέραμε είναι ένας πίνακας από μεταβλητές τύπου char. Μπορούμε να δηλώσουμε πίνακες από μεταβλητές οποιουδήποτε τύπου. Παρακάτω βλέπουμε πως μπορούμε να δηλώσουμε έναν πίνακα που να αποθηκεύουμε μεταβλητές τύπου string

```
Dim strArray(0 To 6) As String
strArray(0) = "Δευτέρα"
strArray(1) = "Τρίτη"
strArray(2) = "Τετάρτη"
strArray(3) = "Πέμπτη"
strArray(4) = "Παρασκευή"
strArray(5) = "Σάββατο"
strArray(6) = "Κυριακή"
```

με τη δήλωση *Dim strArray(0 To 6) As String* φτιάχνουμε μια σειρά από επτά μεταβλητές με τη μορφή πίνακα. Για να τις προσπελάσουμε χρησιμοποιούμε τον δείκτη. Όταν κάνουμε τη δήλωση του πινάκα στην παρένθεση γράφουμε πόσες θέσεις θα έχει ο πίνακας. Η δήλωση *Dim strArray(0 To 6) As String* είναι ισοδύναμη με τη δήλωση *Dim strArray(6) As String*. Απλώς η πρώτη είναι πιο κατανοητή από τη δεύτερη.

Στο «Φύλλο Εργασίας 6» είχαμε δημιουργήσει ένα πρόγραμμα που τύπωνε την ημέρα της εβδομάδας. Η παρακάτω εφαρμογή κάνει ακριβώς το ίδιο με τη χρήση πινάκων με λιγότερες γραμμές κώδικα.

```
Module Module1

    Sub Main()
        Dim strArray(0 To 7) As String
        strArray(0) = "Δευτέρα"
        strArray(1) = "Τρίτη"
        strArray(2) = "Τετάρτη"
        strArray(3) = "Πέμπτη"
        strArray(4) = "Παρασκευή"
        strArray(5) = "Σάββατο"
        strArray(6) = "Κυριακή"
        Dim dt As Date = Date.Now
        Dim dayOfWeek As Integer = dt.DayOfWeek
        Console.WriteLine(strArray(dayOfWeek - 1))
        Console.ReadLine()
    End Sub

End Module
```

Χρησιμοποιώντας όσα έχετε μάθει μέχρι τώρα δημιουργήστε μια εφαρμογή που θα ζητάει από το χρήστη να εισάγει 3 αριθμούς. Οι αριθμοί θα αποθηκεύονται σε έναν πίνακα και στο τέλος θα εμφανίζει το άθροισμά τους.

Φύλλο εργασίας

8. Εκσφαλμάτωση

Όνομα:

Τάξη:

Διάρκεια: 3 διδακτικές ώρες.

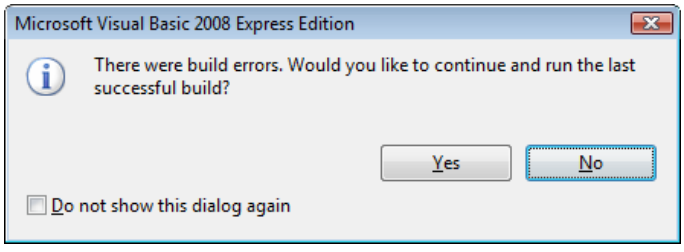
Διδακτικοί στόχοι:

Με το συγκεκριμένο φύλλο εργασίας

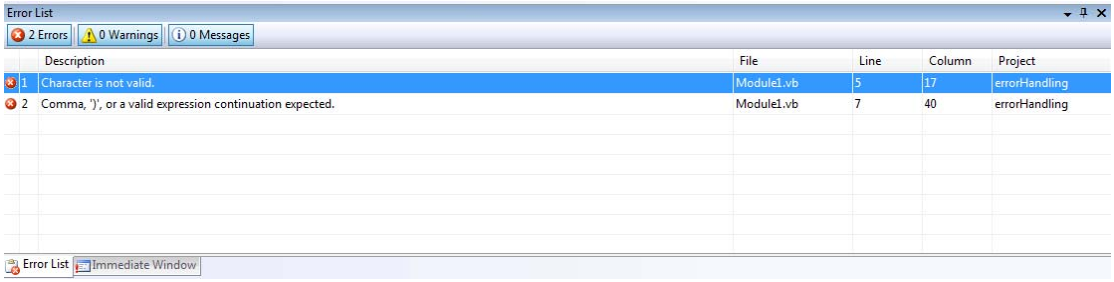
- Θα εξασκηθείς στην αντιμετώπιση λαθων0

Δραστηριότητα 1 : Συντακτικά λάθη.

Κατά τη διάρκεια της συγγραφής κώδικα, όπως ίσως ήδη έχετε διαπιστώσει, μπορούν να γίνουν λάθη που επηρεάζουν τη σωστή λειτουργία της εφαρμογής. Τα λάθη χωρίζονται σε τρείς μεγάλες κατηγορίες. Πρώτη τα συντακτικά λάθη. Συντακτικό λάθος σημαίνει πως οι εντολές που γράψαμε δεν είναι κατανοητές από τον μεταγλωττιστή (compiler). Σε αυτή την περίπτωση, όταν δοκιμάσουμε να τρέξουμε τον κώδικα που γράψαμε, ο compiler της Visual Basic θα μας εμφανίσει το παρακάτω μήνυμα.



Τα συντακτικά λάθη που έχουμε κάνει εμφανίζονται στο παράθυρο με τη λίστα λαθών που βρίσκεται (συνήθως) κάτω από το παράθυρο κώδικα.



Στις παρακάτω γραμμές κώδικα εντοπίστε τα συντακτικά λάθη.

```
Dim str As String
str = "";
Dim num As Integer = 0
Console.WriteLine ("Ο αριθμός "num" ειναι 0")
```

.....
.....
.....
.....
.....
.....

.....
.....
.....
.....
.....
.....

Δραστηριότητα 2 : Επιλογές που επηρεάζουν τα συντακτικά λάθη. (Option Explicit)

Η αυστηρότητα στη σύνταξη της Visual Basic επηρεάζεται από ορισμένες επιλογές του προγραμματιστή. Οι επιλογές αυτές γράφονται στις πρώτες γραμμές του προγράμματος με τη μορφή `Option χχχ On/Off`.

Η επιλογή `Option Explicit On` κάνει τον compiler να χτυπάει συντακτικό λάθος όταν χρησιμοποιούμε μια μεταβλητή που δεν έχουμε προηγουμένως δηλώσει. Παράδειγμα ο παρακάτω κώδικας δεν έχει συντακτικό λάθος (κατά τον compiler)

```
Option Explicit Off
Module Module1

    Sub Main()
```

```
        thisInt = 10
    End Sub

End Module
```

Ενώ ο κώδικας που βρίσκεται πιο κάτω έχει συντακτικό λάθος

```
Option Explicit On
Module Module1

    Sub Main()
        thisInt = 10
    End Sub

End Module
```

Βρείτε ποια είναι η default κατάσταση του `Option Explicit`. Δηλαδή αν δεν δηλώσουμε καθόλου του `Option Explicit` ο compiler συμπεριφέρεται σαν να είναι δηλωμένο on ή off?

.....
.....
.....
.....
.....
.....

Δραστηριότητα 3 : Επιλογές που επηρεάζουν τα συντακτικά λάθη. (Option Strict).

Στο κώδικα που είδατε μέχρι τώρα συναντήστε εντολές που μετατρέπουν ένα είδος μεταβλητής σε άλλο. Πχ.

```
        Dim str As String = Console.ReadLine()
        Dim num As Integer = str
```

Η δεύτερη εντολή μετατρέπει το αλφαριθμητικό σε ακέραιο επειδή το num είναι δηλωμένο σαν ακέραιος. Όπως έχει τονιστεί στο Φ.Ε 7 αυτή η τεχνική δεν είναι ασφαλής και δεν πρέπει να χρησιμοποιείται. Υπάρχουν ασφαλέστεροι τρόποι μετατροπής. Π.χ. για την μετατροπή οποιουδήποτε αντικειμένου σε αλφαριθμητικό θα πρέπει να χρησιμοποιείται η μέθοδος toString(). Γι αυτό το λόγο η Visual Basic προσφέρει την δήλωση `Option Strict` . Όταν η επιλογή αυτή είναι `On` τότε ο παραπάνω κώδικας θα χτυπήσει συντακτικό λάθος.

Βρείτε στον παρακάτω κώδικα ποιες εντολές θα χτυπήσουν συντακτικό λάθος.

```
Option Strict On
Module Module1
```

```
    Sub Main()
        Console.WriteLine("Ο αριθμός 10 ειναι άρτιος")
        Console.WriteLine("Ο αριθμός " + 11 + " ειναι περιττός")
        Dim str As String = 0
        Dim myName As String
        myName = "Δημήτρης"
        Dim myAge As Integer = 40
        Console.WriteLine(myName + ", " + myAge)
    End Sub

End Module
```

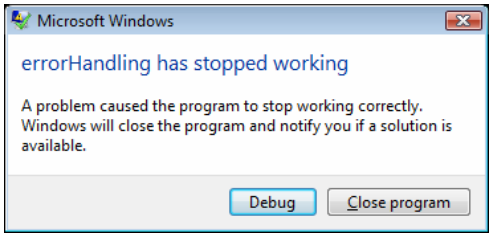
.....
.....
.....
.....
.....
.....

.....
.....
.....
.....
.....
.....

.....
.....
.....
.....
.....
.....

Δραστηριότητα 4 : Οι εντολές try … catch

Στις προηγούμενες δραστηριότητες μιλήσαμε για συντακτικά λάθη που είναι η πρώτη κατηγορία λαθών. Η δεύτερη κατηγορία είναι τα *run time error*. Τα λάθη αυτά συμβαίνουν όταν ο κώδικάς μας εκτελεί μια εσφαλμένη εντολή κατά τη διάρκεια της εκτέλεσης του. Σε αυτή την περίπτωση το πρόγραμμα θα εμφανίσει ένα μήνυμα λάθους σαν αυτό.



Δοκιμάστε τον παρακάτω κώδικα που έχετε ξαναδεί.

```
Module Module1

    Sub Main()
        Dim int As Integer = Console.ReadLine()
    End Sub

End Module
```

Επειδή δεν έχουμε *Option Strict On* ο compiler δεν θα εμφανίσει λάθος. Δημιουργήστε μια νέα εφαρμογή κονσόλας και αποθηκεύστε την. Πληκτρολογήστε τον παραπάνω κώδικα. Τρέξτε το πρόγραμμα και πληκτρολογήστε κάτι που δεν είναι αριθμός και δείτε τι συμβαίνει.

Μετά βρείτε το exe αρχείο που δημιούργησε ο compiler. (είναι στο φάκελο bin\debug). Τρέξτε το αρχείο και πληκτρολογήστε κάτι που δεν είναι αριθμός και δείτε τι συμβαίνει

Τέτοιου είδους λάθη είναι πολύ πιθανόν να συμβούν ακόμα και αν έχουν την επιλογή *Option Strict On* . Ωστόσο μπορείτε να τα διαχειριστείτε με τη δομή try .. catch.

Δείτε ένα παράδειγμα με τη χρήση αυτής της δομής.

```
Module Module1

    Sub Main()
        Try
            Dim int As Integer = Console.ReadLine()
        Catch ex As Exception
            Console.WriteLine("Δεν έδωσες σωστά στοιχεία.")
            Console.ReadLine()
        End Try
    End Sub
```

Πληκτρολογήστε το και τρέξτε το. Παρατηρείστε τη διαφορά. Το πρόγραμμα δεν τερματίζεται όταν ο χρήστης πληκτρολογήσει λάθος δεδομένα.

Όταν εκτελούνται οι εντολές που βρίσκονται μεταξύ του try και catch όταν συμβεί κάτι που θα προκαλέσει run time error το σταματά η εκτέλεση των εντολών και αρχίζουν να εκτελούνται οι εντολές που βρίσκονται μεταξύ catch και end try. Δείτε όμως το παρακάτω παράδειγμα.

```
Module Module1

    Sub Main()
        Dim int As Integer
        Try
            int = Console.ReadLine()
        Catch ex As Exception
            Console.WriteLine("Δεν έδωσες σωστά στοιχεία." + int)
            Console.ReadLine()
        End Try
    End Sub

End Module
```

Σε αυτό το παράδειγμα αν ο χρήστης δώσει λάθος δεδομένα θα προκληθεί πάλι run time error. Οι εντολές που βρίσκονται μεταξύ catch και end try δεν «προστατεύονται».

Στο παρακάτω παράδειγμα δείτε τη χρήση της μεταβλητής που η visual Basic μάς διαθέτει για να χειριζόμαστε τα λάθη (ex στον κώδικά μας).

```
Module Module1

    Sub Main()
        Dim int As Integer
        Try
            int = Console.ReadLine()
        Catch ex As Exception
            Console.WriteLine(ex.Message)
            Console.ReadLine()
        End Try
    End Sub

End Module
```

Η μεταβλητή ex έχει δηλωθεί σαν στιγμιότυπο (instance) της κλάσης Exception. Η κλάση αυτή δίνει την ιδιότητα Message τύπου string που περιέχει το μήνυμα λάθους.

Στο προηγούμενο Φ.Ε δημιουργήσατε μια εφαρμογή που υπολογίζει το παραγοντικό ενός αριθμού. Ωστόσο, αν προσπαθούσε ο χρήστης να υπολογίσει το παραγοντικό ενός μεγάλου αριθμού, το πρόγραμμα θα τερματιζόταν με run time error. Τροποποιήστε αυτή την εφαρμογή ώστε να μην τερματίζεται, αλλά να δίνει ένα μήνυμα λάθους στο χρήστη.

Δραστηριότητα 5 : Λογικά λάθη 1.

Η τρίτη κατηγορία σφαλμάτων είναι τα λογικά σφάλματα. Λογικό σφάλμα λέγεται ένα λάθος όταν το πρόγραμμα θα έπρεπε να έχει διαφορετική συμπεριφορά από αυτή που πραγματικά έχει. Δειτε για παράδειγμα τον παρακάτω κώδικα:

```
Module Module1

    Sub Main()

        Dim numbers() As Double = {1, 1, 1, 1, 1, 1}
        Dim sum = 0
        For i = 1 To numbers.Length - 1
            sum = sum + numbers(i)
        Next
        Console.WriteLine(sum)
        Console.ReadLine()
    End Sub


End Module
```

Ο προφανής στόχος αυτού του κώδικα είναι να υπολογίσει το σύνολο των τιμών του πίνακα numbers. Το αποτέλεσμα που θα εμφανιστεί στην οθόνη θα έπρεπε να είναι 6. Ωστόσο ο κώδικας εμφανίζει 5. Αυτό είναι ένα λογικό λάθος. Για να εντοπιστει πρέπει να παρακολουθήσετε την ροή του προγράμματος βήμα-βήμα

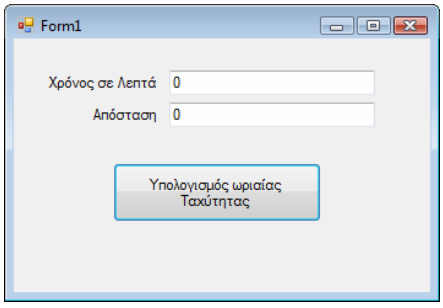
Αυτό γίνεται πατώντας Debug-> Step Into .ή πατώντας F8. Το πρόγραμμα εκτελεί την πρώτη εντολή και περιμένει. Ο προγραμματιστής μπορεί να δει τις τιμές που παίρνουν οι μεταβλητές μετά την εκτέλεση της πρώτης εντολής. Πατώντας ξανά F8 εκτελείται η επόμενη εντολή και μπορούμε να δούμε τις νέες τιμές που πήραν οι μεταβλητές.

Πληκτρολογήστε τον παραπάνω κώδικα και τρέξτε τον βήμα βήμα. Προσπαθήστε να εντοπίσετε το λάθος.

Δραστηριότητα 6 : Λογικά λάθη 2

Φτιάξτε μια νέα εφαρμογή φορμών.

Στην φόρμα τοποθετήστε ένα button, δυο labels και δυο textboxes. Δώστε τιμές στις ιδιότητες Text των αντικειμένων ώστε η φόρμα να μοιάζει με το παρακάτω.



Με το πρόγραμμα αυτό θα καταχωρούμε το χρόνο σε λεπτά και τη απόσταση που διανύθηκε σε χιλιόμετρα. Πατώντας το button1 το πρόγραμμα θα υλοποιίζει την μεση ωριαία ταχύτητα.

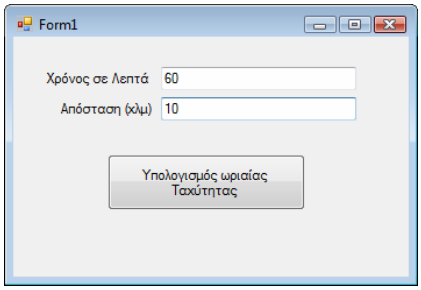
Δημιουργήστε το διαχεριστη συμβάντων του button1 (πατώντας διπλό κλικ πάνω του). Προσθέστε στον διαχειριστη συμβάτων τον πατακάτω κώδικα:

```
        Dim minutes As Integer = TextBox1.Text
        Dim kilometers As Double = TextBox2.Text
        Dim hours As Double = 0
        hours = minutes / 60
        MessageBox.Show("Μέση ωριαία ταχύτητα : " & GetKMPH(hours,
kilometers))
```

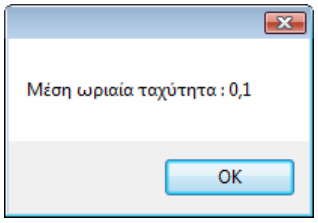
Ο κώδικας αυτός χρησιμοποιεί τη συνάρτηση GetKMPH(). Πληκτρολογήστε τη συνάρτηση αυτή ακριβώς όπως δίνεται παρακάτω.

```
    Function GetKMPH(ByVal kms As Double, ByVal hs As Double) As
String
        Return (kms / hs).ToString()
    End Function
```

Τωρα τρέξτε το πρόγραμμα. Δώστε τιμές για χρόνο 60 και για απόσταση 10. Αν κάποιος διανύσει 10 χλμ σε 60 λεπτά η μέση ωριαία ταχύτητά του είναι 10χλμ/ω. πατήστε το κουμπί υπολογισμού.



Αν έχετε πληκτρολογήσει σωστά τον κώδικα θα εμφανιστεί το παρακάτω μήνυμα.



Το αποτέλεσμα είναι προφανώς λάθος.

Χρησιμοποιώντας αυτά που έχετε μάθει εντοπίστε το λάθος. Ακολούθως διορθώστε την εφαρμογή ώστε να αντιμετωπίσει προβλήματα που θα προκύψουν αν ο χρήστης δεν πληκτρολογήσει αριθμούς ή αν πληκτρολογήσει την τιμή 0 στο πεδίο του χρόνου.

ΥΠΟΥΡΓΕΙΟ ΕΘΝΙΚΗΣ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ ΕΠΕΑΕΚ

ΕΥΡΩΠΑΪΚΗ ΕΝΩΣΗ
ΣΥΓΧΡΗΜΑΤΟΔΟΤΗΣΗ
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

Η ΠΑΙΔΕΙΑ ΣΤΗΝ ΚΟΡΥΦΗ
Επιχειρησιακό Πρόγραμμα
Εκπαίδευσης και Αρχικής
Επαγγελματικής Κατάρτισης