ΑΤΕΙ ΘΕΣΣΑΛΟΝΙΚΗΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΑΥΤΟΜΑΤΙΣΜΟΥ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«Ανάπτυξη Εκπαιδευτικού Υλικού στον Προγραμματισμό ΙΙΙ»

ΕΠΙΒΛΕΠΟΝΤΕΣ ΚΑΘΗΓΗΤΕΣ:

ΜΑΝΩΛΑΚΗΣ ΔΗΜΗΤΡΙΟΣ

ΣΑΛΟΝΙΚΙΔΗΣ ΔΙΟΝΥΣΙΟΣ

ΓΚΑΡΑΝΕ ΔΗΜΗΤΡΑ

ΜΑΡΤΙΟΣ 2018

Ευχαριστίες

Η παρούσα εργασία πραγματοποιήθηκε στα πλαίσια της πτυχιακής άσκησης του τμήματος Αυτοματισμού.

Σε αυτό το σημείο θα ήθελα να ευχαριστήσω θερμά τους ανθρώπους που μου εμπιστεύθηκαν το συγκεκριμένο θέμα καθώς και για την βοήθειά τους στην προσπάθειά μου αυτή.

Τον κ. Μανωλάκη Δημ. επιβλέπων καθηγητή του ΑΤΕΙ Θεσσαλονίκης του τμήματος Αυτοματισμού που επιμελήθηκε στην εκπόνηση της εργασίας.

Επίσης, θα ήθελα να ευχαριστήσω τον κ. Σαλονικίδη Διον. για τις πολύτιμες συμβουλές του και την υποστήριξή του καθ’ όλη τη διάρκεια εξέλιξης της εργασίας.

Το διδακτικό υλικό διατίθεται στους χρήστες του για αυστηρά εκπαιδευτική ή και προσωπική χρήση και μόνο για σκοπούς εκπαιδευτικούς.

Περίληψη

Η παρούσα εργασία πραγματεύεται την μάθηση, την διδασκαλία και την εξ’ αποστάσεως διδασκαλία με την βοήθεια της τεχνολογίας και των μέσων της.

Στόχος αυτής της εργασίας είναι να βοηθήσει τους φοιτητές του τμήματος ώστε να κατανοήσουν και να διδαχτούν την C# και την χρήση του Microsoft Visual C# μέσα από διάφορες εφαρμογές και παραδείγματα.

Η εργασία αποτελείται από τρία μέρη:

- Στο πρώτο μέρος παρουσιάζεται το θεωρητικό πλαίσιο της εργασίας.
- Το δεύτερο μέρος περιλαμβάνει την δημιουργία ηλεκτρονικού εκπαιδευτικού υλικού και ασκήσεων με το εργαλείο της Microsoft το PowerPoint για το εργαστηριακό μάθημα του Προγραμματισμού ΙΙΙ.
- Και τέλος, το τρίτο μέρος με τα τελικά αποτελέσματα των εκτελέσιμων αρχείων των εφαρμογών που θα ασχοληθούμε.

Περιεχόμενα

1. Η διδασκαλία ενηλίκων

Διδασκαλία είναι η μετάδοση γνώσης από έναν άνθρωπο σε έναν άλλον με σκοπό την αφομοίωση της γνώσης, δηλαδή τη μάθηση. Διακρίνεται σε φυσική διδασκαλία και σχολική διδασκαλία.

Η **φυσική διδασκαλία** πραγματοποιείτε κυρίως εκτός σχολείου, αυθόρμητα και χωρίς προετοιμασία. Η διαδικασία αυτή είναι αλληλεπιδραστική και όλοι όσοι συμμετέχουν διδάσκουν και διδάσκονται. Η φυσική διδασκαλία ξεκινά πριν το άτομο αρχίσει να φοιτά στο σχολείο από ποικίλες επιρροές από το άμεσο ή έμμεσο περιβάλλον του. Μέσω αυτής το άτομο αποκτά τις πρώτες του γνώσεις και δεξιότητες και διαμορφώνει τη βάση της συμπεριφοράς του.

Η **σχολική διδασκαλία** λαμβάνει χώρα μέσα σε θεσμοθετημένους φορείς και ακολουθεί συστηματικά τη σειρά μιας γραμμής παραγωγής της γνώσης στους μαθητές. Το περιεχόμενό της καθορίζεται από συγκεκριμένο αναλυτικό πρόγραμμα. Οι μαθητές είναι ανά ηλικία χωρισμένοι σε τάξεις και τους παρέχεται από κοινού εκπαίδευση και όχι εντοπισμένη στις ιδιαιτερότητες και ικανότητες του καθενός ξεχωριστά.

Η διδασκαλία διεξάγεται σύμφωνα με τις εκάστοτε ορθές παιδαγωγικές αρχές μέσα από τις οποίες επιδιώκει να ικανοποιήσει τις κοινωνικές ανάγκες, όπως για παράδειγμα να στελεχώσει αποδοτικά με ανθρώπινο δυναμικό το τρέχον και το προβλεπόμενο οικονομικό σύστημα. Έτσι αναλαμβάνει τον ρόλο και την ευθύνη της μάθησης και της απόκτησης δεξιοτήτων του ατόμου. Ο εκπαιδευτικός ξεκινά την διδασκαλία με βάση τους διδακτικούς στόχους που έχει προσδιορίσει, και επιδιώκει αυτοί οι στόχοι να γίνουν και στόχοι των μαθητών, να γίνουν δηλαδή μαθησιακοί στόχοι. Η διδασκαλία ως αρμοδιότητα του σχολείου ξεκίνησε απ’ την ανάγκη κάλυψης στοιχειωδών γνώσεων και βασικών πολιτιστικών δεξιοτήτων και εξελίχθηκε σταδιακά στην σημερινή γενική και επαγγελματική εκπαίδευση.

Εξ’ αποστάσεως διδασκαλία

Εξ’ αποστάσεως διδασκαλία είναι η υποβοηθούμενη από τα μέσα επικοινωνίας εκπαίδευση με καθόλου ή ελάχιστη διαπροσωπική ή σε τάξη επαφή μεταξύ μαθητή και εκπαιδευομένου. Βρίσκεται τα τελευταία χρόνια στο επίκεντρο του ενδιαφέροντος και ολοένα και περισσότεροι φορείς παροχής εκπαίδευσης ενηλίκων την υιοθετούν σε μια προσπάθεια να προσφέρουν εκπαίδευση σε ευρύτερο κοινό από αυτό που είναι δυνατόν να παρακολουθήσει εκπαιδευτικές δραστηριότητες με τις συμβατικές μεθόδους.

Οι συνεχώς αυξανόμενες ανάγκες της σύγχρονης πραγματικότητας για απόκτηση νέων γενικών και ειδικών γνώσεων και δεξιοτήτων και η αδυναμία των συμβατικών μορφών εκπαίδευσης να καλύψουν τις ανάγκες αυτές, έχουν οδηγήσει στην υιοθέτηση καινοτόμων μεθόδων εκπαίδευσης όπως είναι η εξ’ αποστάσεως εκπαίδευση.

Η εξ’ αποστάσεως εκπαίδευση μπορεί να διαχωριστεί σε σύγχρονη και ασύγχρονη.

Στην **σύγχρονη εκπαίδευση** η μάθηση γίνεται ταυτόχρονα. Ο εκπαιδευτής παραδίδει το μάθημα σε ζωντανή σύνδεση μέσω τηλεδιάσκεψης ή κάποιου live chatroom.

Στην **ασύγχρονη εκπαίδευση** ο εκπαιδευόμενος μαθαίνει όχι μόνο σε διαφορετικό χώρο από τον εκπαιδευτή αλλά και σε διαφορετικό χρόνο από αυτόν της παράδοσης του μαθήματος. Αυτή αποτελεί και την πιο διαδεδομένη μέθοδο από τις δύο.

Η πορεία της εξ’ αποστάσεως εκπαίδευσης ανά των χρόνων:

- Δι’ αλληλογραφίας, *μέσα του 19^{ου} αιώνα*
- Ηλεκτρονικά Μέσα, *αρχές του 20^{ου} αιώνα* (ραδιόφωνο, τηλεόραση, βίντεο)
- Αλληλεπιδραστικές τεχνολογίες, *παρόν (e-mail, discussion forum, τηλεδιάσκεψη)*

2. Ιστορία των Ηλεκτρονικών Υπολογιστών

Οι ηλεκτρονικοί υπολογιστές έχουν σχετικά σύντομη ιστορία. Ο πρώτος υπολογιστής που δημιουργήθηκε ήταν ο *Atanasoff-Berry Computer* το 1937 στο πανεπιστήμιο της Iowa για να επεξεργάζεται μεγάλο αριθμό παράλληλων εξισώσεων. Λίγο αργότερα κατασκευάστηκε ο *ENIAC (Electronic Numerical Integrator and Computer)*. Ήταν υπολογιστής του στρατού και χρησίμευε για γενικούς υπολογισμούς, αλλά η καλωδίωση έπρεπε να αναπροσαρμόζεται κάθε φορά που έπρεπε να εκτελεστεί μια νέα εργασία. Και στους δύο αυτούς υπολογιστές, από τη στιγμή που ορίζονταν η ηλεκτρονική καλωδίωση, μόνο τα δεδομένα εισάγονταν στη μνήμη του υπολογιστή.

Μεγάλη πρόοδο στην εξέλιξη των υπολογιστών έκανε ο μαθηματικός *John Von Neumann* που πρότεινε μια εναλλακτική προσέγγιση αντί της επαναδιάταξης της καλωδίωσης. Εισήγαγε την έννοια της αποθήκευσης των εντολών του υπολογιστή στην μνήμη του. Οι εντολές θα υπαγόρευαν τις διευθύνσεις και τις θέσεις στις οποίες οι ηλεκτρονικοί παλμοί θα έρρεαν με τον ίδιο ακριβώς τρόπο που οι καλωδιώσεις υπαγόρευαν την ροή. Για τους υπολογιστές αυτούς, τόσο τα δεδομένα όσο και οι εντολές του υπολογιστή αποτελούσαν είσοδο στη μνήμη.

3. Εντολές και Γλώσσες Προγραμματισμού

Θα πρέπει να γνωρίζουμε ότι οι εντολές που πληκτρολογούμε στη γλώσσα C πρέπει να μετατρέπονται σε ένα είδος δυαδικού κώδικα και να αποθηκεύονται στη μνήμη. Αυτός ο δυαδικός κώδικας αναφέρεται ως *γλώσσα μηχανής*.

Η γλώσσα μηχανής είναι η μόνη γλώσσα που μπορεί να καταλάβει ο υπολογιστής. Αποτελείται από εντολές σε δυαδικό κώδικα και είναι ειδική για το συγκεκριμένο επεξεργαστή που πρόκειται να χρησιμοποιηθεί. Κάθε βήμα που πρέπει να κάνει ο υπολογιστής πρέπει να γράφεται σε αυτές τις εντολές. Επειδή η γλώσσα μηχανής είναι περίπλοκη, τα περισσότερα προγράμματα γράφονται σε άλλες γλώσσες και μεταφράζονται σε γλώσσα μηχανής. Πολλές γλώσσες έχουν γραφτεί που να μπορούν να μεταφραστούν σε γλώσσα μηχανής. Κάποιες από αυτές είναι η *συμβολική γλώσσα (Assembly)* και οι γλώσσες υψηλού επιπέδου.

Στη *γλώσσα Assembly* όλα τα βήματα εντολών για τη λίστα εντολών που περιέχεται σε ένα πρόγραμμα γραμμένο σε γλώσσα μηχανής είναι απαραίτητα. Οι αγγλικές λέξεις μεταφράζονται στο δυαδικό κώδικα της γλώσσας μηχανής από ένα μεταφραστικό πρόγραμμα.

Οι *γλώσσες υψηλού επιπέδου* απλοποιούν ακόμη περισσότερο τις εντολές που χρειάζεται να γραφούν από τους προγραμματιστές. Σε αντίθεση με τη γλώσσα μηχανής, οι γλώσσες υψηλού επιπέδου επιτρέπουν στους προγραμματιστές να γράφουν προγράμματα χωρίς να δίνουν πολύ σημασία για την εσωτερική σχεδίαση της μηχανής στην οποία το πρόγραμμα πρόκειται να χρησιμοποιηθεί. Είναι απαραίτητο όμως το πρόγραμμα να είναι συμβατό με τον υπολογιστή. Αυτό καθιστά το πρόγραμμα μεταφέρσιμο από έναν υπολογιστή σε άλλον.

Οι γλώσσες υψηλού επιπέδου είναι σχεδιασμένες για να απλοποιούν το γράψιμο προγραμμάτων που προορίζονται για τη λύση συγκεκριμένου είδους προβλημάτων. Δηλαδή, άλλη γλώσσα σχεδιάστηκε για τη λύση επιστημονικών προγραμμάτων και άλλη για λογιστικά προβλήματα επιχειρήσεων.

Έτσι οι γλώσσες μπορούν να χωριστούν σε τέσσερα είδη:

- Διαδικαστικές (Procedural) ή προστακτικές (Imperative)
- Συναρτησιακές (Functional)
- Δηλωτικές (Declarative)
- Αντικειμενοστραφείς (Object Oriented)

Ακολουθεί ένας πίνακας με ορισμένες γλώσσες υψηλού επιπέδου.

Όνομα γλώσσας	**Είδος γλώσσας**	**Έτος δημιουργίας**
Fortran	Διαδικαστική	1950
Basic	Διαδικαστική	1960
Lisp	Συναρτησιακή	1950
Prolog	Δηλωτική	1970
Ada	Διαδικαστική	1970
Smalltalk	Αντικειμενοστραφής	1970
Pascal	Διαδικαστική	1970
C	Διαδικαστική	1970
C++	Αντικειμενοστραφής	1980

Αντικειμενοστραφής Προγραμματισμός

Αντικειμενοστραφής προγραμματισμός ή αλλιώς ΑΠ, ονομάζουμε ένα προγραμματιστικό υπόδειγμα το οποίο εμφανίστηκε στα τέλη της δεκαετίας του 60, αντικαθιστώντας σε μεγάλο βαθμό το παραδοσιακό υπόδειγμα του δομημένου προγραμματισμού. Πρόκειται για έναν τρόπο οργάνωσης των προγραμμάτων που γράφουμε.

Ο αντικειμενοστραφής προγραμματισμός γεννήθηκε και άρχισε να αναπτύσσεται όταν πλέον ήταν φανερό ότι οι παραδοσιακές προσεγγίσεις στον προγραμματισμό δεν μπορούσαν να ανταποκριθούν στις νέες απαιτήσεις ανάπτυξης προγραμμάτων. Επιπλέον, καθώς τα προγράμματα μεγάλωναν, γίνονταν υπερβολικά περίπλοκα. Διαπιστώθηκε ότι υπήρχαν αδυναμίες με την χρήση διαδικαστικών γλωσσών προγραμματισμού. Η κυριότερη αιτία είναι ότι οι διαδικαστικές γλώσσες δίνουν έμφαση στις ενέργειες που πρέπει να εκτελέσει ένα πρόγραμμα.

Στον ΑΠ ο χειρισμός σχετιζόμενων δεδομένων και των διαδικασιών που επενεργούν σε αυτά γίνεται από κοινού, μέσω μιας δομής δεδομένων που τα περιβάλλει ως μια οντότητα που έχει συγκεκριμένες ιδιότητες και μπορεί να εκτελέσει συγκεκριμένες ενέργειες. Αυτή η δομή δεδομένων καλείται ***αντικείμενο*** και αποτελεί πραγματικό στιγμιότυπο στη μνήμη ενός σύνθετου τύπου δεδομένων ονόματι ***κλάση***. Η κλάση είναι ένας ορισμός. Αυτό που ορίζει είναι ποιες ιδιότητες (attributes) και ποιες μεθόδους (methods) θα έχει ένα αντικείμενο.

Αντικείμενα

Βασικά στοιχεία του αντικειμενοστραφούς προγραμματισμού αποτελούν τα αντικείμενα. Στα προγράμματα, ένα αντικείμενο είναι η ομαδοποίηση κώδικα και δεδομένων, τα οποία χειριζόμαστε ενιαία. Τα δεδομένα αποτελούν τα χαρακτηριστικά ενός αντικειμένου και οι ενέργειες καθορίζουν τη συμπεριφορά του. Οι ενέργειες αναφέρονται και ως **μέθοδοι** (methods).

Η ανάπτυξη μιας αντικειμενοστραφούς εφαρμογής συνίσταται από τη δημιουργία και το χειρισμό αντικειμένων. Σε κάποια αντικειμενοστραφή περιβάλλοντα τα αντικείμενα της εφαρμογής μπορεί να δημιουργούνται είτε μέσω κώδικα, είτε με τη βοήθεια κατάλληλων γραφικών εργαλείων, ενώ σε άλλα περιβάλλοντα δημιουργούνται μόνο μέσω κώδικα.

Κλάσεις

Όλες οι γλώσσες προγραμματισμού έχουν ενσωματωμένους τύπους δεδομένων. Όπως για παράδειγμα ο τύπος δεδομένων int είναι ήδη προκαθορισμένος στην C++ και στα προγράμματα μας μπορούμε να δηλώνουμε μεταβλητές αυτού του τύπου. Με παρόμοιο τρόπο μπορούμε να ορίσουμε αντικείμενα της ίδιας κλάσης. Η κλάση αποτελεί το πρότυπο και καθορίζει τα δεδομένα και τις συναρτήσεις που θα περιληφθούν στα αντικείμενα αυτής της κλάσης. Θα πρέπει να τονιστεί ότι ο ορισμός της κλάσης δε δημιουργεί κανένα αντικείμενο, όπως η απλή ύπαρξη ενός τύπου int δε δημιουργεί καμία μεταβλητή. Συνεπώς, σε ένα αντικειμενοστραφές προγραμματιστικό περιβάλλον η υποστήριξη κλάσεων αποτελεί κυρίαρχο στοιχείο. Η κλάση είναι η στατική περιγραφή ενός συνόλου παρόμοιων αντικειμένων. Όλα τα αντικείμενα δημιουργούνται ως ακριβή αντίγραφα της κλάσης τους.

4. Τι είναι το .Net Framework

Το .Net Framework είναι μια επαναστατική πλατφόρμα η οποία δημιουργήθηκε από τη Microsoft για τον σχεδιασμό εφαρμογών. Περιλαμβάνει μια μεγάλη βιβλιοθήκη και υποστηρίζει πολλές γλώσσες προγραμματισμού όπως η C#, η C++, η Visual Basic, η Jscript, ή και πιο παλιές γλώσσες όπως η COBOL. Επιτρέπει την διαλειτουργικότητα, δηλαδή κάθε γλώσσα να μπορεί να χρησιμοποιεί κώδικα που έχει γραφτεί σε άλλες γλώσσες. Προγράμματα που έχουν γραφτεί για το .Net Framework εκτελούνται σε ένα περιβάλλον λογισμικού γνωστό ως Common Language Runtime (CLR) μια εφαρμογή εικονικής μηχανής που παρέχει σημαντικές υπηρεσίες όπως η ασφάλεια και η διαχείριση μνήμης. Η βιβλιοθήκη και το CLR μαζί αποτελούν το .Net Framework.

Αυτή τη γιγαντιαία βιβλιοθήκη κώδικα μπορεί να χρησιμοποιηθεί από γλώσσες προγραμματισμού όπως η C#, χρησιμοποιώντας τεχνικές αντικειμενοστραφούς προγραμματισμού. Κατηγοριοποιείτε σε διαφορετικές ενότητες και κάθε φόρα χρησιμοποιούνται τμήματά της ανάλογα με το αποτέλεσμα που θέλουμε να επιτύχουμε.

Το .Net Framework δεν περιλαμβάνει κανένα περιορισμό στο τι μπορεί να κάνει. Επιτρέπει τη δημιουργία των εφαρμογών των Windows, Web εφαρμογές, υπηρεσίες διαδικτύου και οτιδήποτε άλλο μπορείτε να σκεφτείτε. Επίσης, με τις εφαρμογές Web αξίζει να σημειωθεί ότι είναι, εξ ορισμού, multiplatform εφαρμογές, δεδομένου ότι οποιοδήποτε σύστημα με ένα πρόγραμμα περιήγησης στο Web μπορεί να έχει πρόσβαση σε αυτές.

Γράφοντας μια εφαρμογή με το .Net Framework σημαίνει σύνταξη κώδικα χρησιμοποιώντας τη βιβλιοθήκη κώδικα της .Net. Αυτό μπορεί να γίνει με το Visual Studio και με το Visual Studio Express. Το VS είναι ένα ισχυρό, ολοκληρωμένο περιβάλλον ανάπτυξης που υποστηρίζει C# και άλλα προγράμματα. Η VSE είναι μια έκδοση της VS που υποστηρίζει C# μόνο.

Οι γλώσσες προγραμματισμού συνήθως αποτελούνται από έναν compiler και ένα runtime περιβάλλον. Ο compiler μεταφράζει τον κώδικα σε εκτελέσιμο αρχείο που μπορεί να εκτελεστεί από τους χρήστες. Το runtime περιβάλλον παρέχει ένα σύνολο υπηρεσιών του λειτουργικού συστήματος, στον εκτελέσιμο κώδικα. Οι υπηρεσίες αυτές είναι ενσωματωμένες σε ένα επίπεδο runtime που επιτρέπει στον κώδικα να μην ασχολείται με λεπτομέρειες χαμηλού επιπέδου του λειτουργικού συστήματος. Τέτοιες λειτουργίες μπορεί να είναι η διαχείριση μνήμης, εγγραφή και ανάγνωση αρχείων κτλ. Πριν το .Net Framework κάθε γλώσσα είχε και το δικό της runtime περιβάλλον. Το περιβάλλον ενσωματωνόταν με τον εκτελέσιμο κώδικα και έπρεπε να εγκατασταθεί στο μηχάνημα του χρήστη. Το βασικό πρόβλημα με τα περιβάλλοντα αυτά, βρίσκεται στο ότι ήταν σχεδιασμένα για χρήση με μόνο μία γλώσσα. Δεν μπορούσαν να χρησιμοποιηθούν λειτουργίες από το περιβάλλον μιας

γλώσσας, σε μια άλλη. Έτσι, ένας από τους βασικούς στόχους του .Net Framework ήταν να ενοποιήσει τα runtime περιβάλλοντα, ώστε οι προγραμματιστές να μπορούν να χρησιμοποιούν μόνο ένα περιβάλλον. Η λύση που δόθηκε ήταν η Common Language Runtime (CLR). Το CLR παρέχει δυνατότητες όπως διαχείριση μνήμης, ασφάλεια, διαχείριση λαθών κτλ. και όλα αυτά, για κάθε γλώσσα που δουλεύει με το .Net Framework.

Το .Net Framework παρέχει πολλές κλάσεις για να βοηθήσει τους προγραμματιστές στην επαναχρησιμοποίηση κώδικα. Οι βιβλιοθήκες .Net Class Libraries περιέχουν κώδικα για προγραμματιστικά θέματα όπως εγγραφή/ ανάγνωση αρχείων, υποστήριξη βάσεων δεδομένων, δομές δεδομένων όπως στοίβες και ουρές κτλ. Επίσης, παρέχει ένα σύνολο εργαλείων για να βοηθήσει στην κατασκευή κώδικα που λειτουργεί με αυτό.

Πλεονεκτήματα .Net

- Είναι εγγενώς αντικειμενοστραφής πλατφόρμα
- Είναι ανεξάρτητο από γλώσσα προγραμματισμού. Σε μία εφαρμογή ένας προγραμματιστής μπορεί να γράφει κώδικα σε C#, ένας άλλος σε C++ και τα τμήματα που αναπτύσσει ο καθένας να συνεργάζονται μεταξύ τους χωρίς κανένα πρόβλημα.
- Η χρήση βιβλιοθηκών (assemblies) κάνει πολύ εύκολη την επαναχρησιμοποίηση κώδικα.
- Παρέχει πολύ εύκολη εγκατάσταση. Αρκεί να αντιγράψουμε τον κατάλογο της εφαρμογής σε έναν άλλον υπολογιστή και αυτή θα τρέξει άμεσα. Δεν υπάρχει installation.
- Παρέχει πληθώρα έτοιμων λειτουργιών που κάνουν την ανάπτυξη κώδικα πολύ εύκολη.
- Αυτοματοποιημένη διαχείριση μνήμης. Ο χρήστης δεν χρειάζεται να ασχοληθεί με αποδέσμευση μνήμης.

Μειονεκτήματα .Net

Τα μειονεκτήματα αφορούν ειδικά την ανάπτυξη βιντεοπαιχνιδιών. Αυτά είναι:

- Αυτοματοποιημένη διαχείριση μνήμης, ο χρήστης δεν χρειάζεται να ασχοληθεί με αποδέσμευση μνήμης.
- Το CLR εισάγει μία μικρή καθυστέρηση στην εκτέλεση της εφαρμογής.

5. Τι είναι η C#;

Όπως προαναφέρθηκε, η C# είναι μια ολοκληρωμένη αντικειμενοστραφής γλώσσα προγραμματισμού που μπορείτε να χρησιμοποιήσετε για να δημιουργήσετε εφαρμογές που θα τρέχουν με το .Net CLR. Αποτελεί την εξέλιξη των C και C++ γλωσσών και έχει δημιουργηθεί από την Microsoft ειδικά για να εργάζεται με την πλατφόρμα της Net. Βασικό χαρακτηριστικό της είναι ότι δεν παράγει απευθείας κώδικα μηχανής όπως η C++, αλλά ένα ενδιάμεσο κώδικα που στοχεύει το Net. Η C# έχει σχεδιαστεί για να ενσωματώσει πολλά από τα καλά χαρακτηριστικά των άλλων γλωσσών, ενώ ξεκαθαρίζει τα προβλήματά τους.

Η ανάπτυξη εφαρμογών χρησιμοποιώντας C# είναι απλούστερη απ' ότι με τη C++, επειδή η σύνταξη της γλώσσας είναι πιο εύκολη. Κατά καιρούς, ο κώδικας της C# είναι πιο φλύαρος από αυτό της C++, επειδή είναι ένας τύπος ασφαλούς γλώσσας. Σύμφωνα με τους ειδικούς, αυτό σημαίνει ότι μόλις κάποια δεδομένα ανατεθούν σε έναν τύπο, δεν μπορεί στη συνέχεια να μεταμορφωθεί σε έναν άλλον μη σχετικό τύπο. Κατά συνέπεια, οι αυστηροί κανόνες θα πρέπει να τηρούνται κατά την μετατροπή μεταξύ των διαφόρων τύπων, που σημαίνει ότι θα πρέπει να γράψουμε συχνά περισσότερο κώδικα για να πραγματοποιήσει την ίδια εργασία σε C# απ' ότι σε C++. Παρ' όλα αυτά έχουμε δύο πλεονεκτήματα: ο κώδικας είναι πιο ισχυρός και το debugging (εντοπισμός λαθών) είναι πιο απλό και η Net μπορεί να εντοπίζει το είδος από ένα κομμάτι δεδομένων οποιαδήποτε στιγμή.

Η C# είναι μια από τις διαθέσιμες γλώσσες που μπορείς να χρησιμοποιήσεις για την ανάπτυξη σε Net, αλλά είναι σίγουρα η καλύτερη. Έχει το πλεονέκτημα ότι είναι η μόνη γλώσσα που σχεδιάστηκε από το μηδέν για το .Net Framework και είναι η κύρια γλώσσα που χρησιμοποιείται στις εκδόσεις του Net που έχουν μεταφερθεί σε άλλα λειτουργικά συστήματα. Η C# μπορεί να κάνει χρήση κάθε χαρακτηριστικού της βιβλιοθήκης του Net Framework που έχει να προσφέρει. Στην τελευταία έκδοση το Net περιλαμβάνει αρκετές προσθήκες από την C#, εν μέρει ως απάντηση στα αιτήματα από προγραμματιστές, γεγονός που την καθιστά ακόμα πιο ισχυρή.

Μεταβλητές

Η C# υποστηρίζει τους παραδοσιακούς τύπους δεδομένων όπως *int* για ακεραίους, *float* για αριθμούς κινητής υποδιαστολής, *string* για κείμενο, *char* για χαρακτήρες. Επιπλέον, υποστηρίζει τύπους *class* και *struct* οι οποίοι επιτρέπουν στον χρήστη να ορίσει δικά του αντικείμενα.

Ο παρακάτω πίνακας περιλαμβάνει τους πιο κοινούς τύπους δεδομένων που χρησιμοποιούμε κατά τον προγραμματισμό με C#.

Τύπος	**Κατηγορία**	**Δεδομένα**	**Παράδειγμα χρήσης**
bool	Boolean	true ή false	bool emfanise = false;
byte	Integer	Ακέραιος αριθμός από 0-255	byte counter = 4;
int	Integer	Ακέραιος αριθμός 32 bit	int largenumber = 234532;
double	Real	Αριθμός κινητής υποδιαστολής μεγάλης ακριβείας	double x = 2342,45434;
float	Real	Αριθμός κινητής υποδιαστολής	float exchangerate = 324.543f;
char	Character	Χαρακτήρας ASCII 8 bit	char myChar = ‘a’;
string	String	Κείμενο	string username = “Nikos”;

Μερικές παρατηρήσεις πάνω στους τύπους δεδομένων. Οι τύποι *short* και *int* έχουν παρόμοια χρήση, μόνο που ο int έχει μεγαλύτερο εύρος, δηλαδή μπορεί να αποθηκεύσει περισσότερους αριθμούς απ’ ότι ο short. Αυτό συμβαίνει γιατί μια μεταβλητή τύπου short χρησιμοποιεί 16 bits στην μνήμη, ενώ μια int 32 bits. Το ίδιο ισχύει και για τους τύπους *float* (32bit) και *double* (64bit). Τέλος, ένας χαρακτήρας (char) ορίζεται με μονά εισαγωγικά και ένα κείμενο (string) με διπλά.

Δηλώσεις σταθερών

Στην C# οι σταθερές δηλώνονται με ένα ειδικό τύπο δεδομένου, τη λέξη κλειδί **const**. Μερικά παραδείγματα είναι τα παρακάτω:

public const int i=60;

protected const string name= "Pavlos" surname= null;

*protected internal const int j= j*2;*

Οι μόνες τιμές που επιτρέπεται να πάρουν οι σταθερές της C# είναι **string, literals, null**.

Ο προσδιοριστής (modifier) μιας σταθερής μεταβλητής για μία κλάση παίρνει τις ακόλουθες τιμές:

- **Public:** Με τον όρο αυτό δηλώνουμε πως η σταθερά της κλάσης είναι προσπελάσιμη από οποιοδήποτε σημείο μέσα στην κλάση ή και εκτός αυτής.
- **Protected:** Η σταθερά της κλάσης είναι προσπελάσιμη από κλάσεις που είναι υποκλάσεις της κλάσης που ορίζεται.
- **Private:** Η σταθερά της κλάσης είναι προσπελάσιμη μόνο μέσα στην κλάση που ορίζεται.
- **Internal:** Η σταθερά της κλάσης είναι προσπελάσιμη από οποιαδήποτε κλάση εντός της επικράτειας της κλάσης που ορίζεται.
- **Protected internal:** Η σταθερά της κλάσης είναι προσπελάσιμη από οποιαδήποτε κλάση εντός της επικράτειας της κλάσης που ορίζεται ή από κλάσεις που είναι υποκλάσεις της κλάσης που ορίζεται.
- **New:** Με τον όρο αυτό δεσμεύουμε μνήμη για την αποθήκευση της μεταβλητής.

Κλάσεις και αντικείμενα

Εκτός από αυτούς τους βασικούς τύπους δεδομένων, η C# υποστηρίζει και τον τύπο **class**. Η class μας επιτρέπει να ορίσουμε δικά μας αντικείμενα. Τα αντικείμενα, όπως αναλύσαμε και παραπάνω, είναι μια οντότητα που χαρακτηρίζεται από δεδομένα και συμπεριφορά. Τα αντικείμενα αυτά έχουν άμεση αντιστοίχιση με αντικείμενα του πραγματικού κόσμου. Ας δούμε ένα παράδειγμα για να το κατανοήσουμε καλύτερα. Ένα αυτοκίνητο σε έναν παιχνίδι αγώνων ράλι χαρακτηρίζεται από το χρώμα του, το πόσες πόρτες έχει, την ταχύτητα του, την κατεύθυνση του, αν έχει ταχύτητες ή είναι αυτόματο, το ποσοστό ζημίας από τρακάρισμα που έχει υποστεί. Αυτά είναι τα δεδομένα που περιγράφουν το αντικείμενο του αυτοκινήτου. Επιπλέον, μπορούμε να ορίσουμε και κάποιες λειτουργίες που επιδρούν και διαχειρίζονται τα δεδομένα αυτά. Μια λειτουργία θα μπορούσε να είναι η αλλαγή ταχύτητας. Μια άλλη, το πάτημα γκαζιού. Οι λειτουργίες αυτές αλλάζουν τις τιμές των δεδομένων του αντικειμένου.

Στην C# το αντικείμενο του αυτοκινήτου μεταφέρεται ως:

```
01 class Car
02 {
03     private float speed;
04     private int gear;
05
06     public Car()
07     {
08         speed = 0;
09         gear = 0;
10     }
11
12     public void IncreaseSpeed(float amount)
13     {
14         speed = speed + amount;
15     }
```

```
16
17    public void DecreaseSpeed (float amount)
18    {
19        speed = speed - amount;
20    }
21
22    public void ChangeGear(int newGear)
23    {
24        gear = newGear;
25    }
26 }
```

Με τον τρόπο αυτό ορίζουμε ένα νέο τύπο δεδομένων, το αντικείμενο Car, το οποίο περιγράφει με απλοϊκό τρόπο μια οντότητα αυτοκινήτου με διάφορες λειτουργίες. Το αντικείμενο περιλαμβάνει δύο μεταβλητές, speed και gear του αυτοκινήτου, συναρτήσεις για αύξηση και μείωση της ταχύτητας, και μία ειδική μέθοδο Car().

Για να ορίσουμε ένα αυτοκίνητο με βάση το πρότυπο αυτό το δηλώνουμε ως εξής:

```
1 Car Bmw;
```

Η παραπάνω δήλωση δημιουργεί μία μεταβλητή τύπου Car, αλλά δεν δεσμεύει μνήμη για το αντικείμενο αυτό καθ’ αυτό. Αυτό θα γίνει ως εξής:

```
2 Bmw = new Car();
```

Ο τελεστής new δεσμεύει όση ακριβώς μνήμη χρειάζεται για να αποθηκευτεί ένα αντικείμενο τύπου Car. Το Bmw είναι τώρα ένα νέο αντικείμενο τύπου Car. Για να αλλάξω τις ιδιότητες του νέου αυτοκινήτου καλώ απλά τις μεθόδους:

```
Bmw.ChangeGear(1);

Bmw.IncreaseSpeed(20);
```

Παρατηρούμε ότι οι μεταβλητές δηλώνονται ως private και οι μέθοδοι ως public. Σε ένα αντικείμενο, οτιδήποτε δηλώνεται ως private δεν είναι ορατό και προσπελάσιμο εκτός του αντικειμένου. Αντίθετα, οτιδήποτε δηλώνουμε ως public, είναι. Για παράδειγμα, η παρακάτω χρήση δεν είναι επιτρεπτή.

```
Bmw.speed = 15.0f;
```

Αυτό συμβαίνει γιατί η speed έχει δηλωθεί ως *private*. Ο μεταγλωττιστής της C# δεν θα επιτρέψει την ολοκλήρωση της δημιουργίας του εκτελέσιμου κώδικα και θα επισημάνει το λάθος.

Τέλος, η μέθοδος Car(). Η μέθοδος αυτή, που έχει πάντα το ίδιο όνομα με την *class* που την περιέχει ονομάζεται *constructor* και καλείτε πάντα όταν δεσμεύουμε μνήμη για ένα αντικείμενο του αντίστοιχου τύπου με τη χρήση του τελεστή *new*. Σκοπός της είναι η αρχικοποίηση του αντικειμένου δίνοντας, για παράδειγμα, κάποια αρχική τιμή στις μεταβλητές του. Στο συγκεκριμένο παράδειγμα τη χρησιμοποιούμε για να θέσουμε αρχικές τιμές 0 στις μεταβλητές speed και gear.

Εκτός της class, η C# υποστηρίζει τη δημιουργία νέων τύπων δεδομένων με τη χρήση της **struct**. Η struct είναι παρόμοια στην χρήση με την class. Ανακεφαλαιώνοντας, η C# υποστηρίζει βασικούς τύπους δεδομένων (primitive types), όπως ο int, short, char κτλ, και πιο σύνθετους όπως η class (reference types). Μια βασική διαφορά τους είναι ότι ένα βασικό τύπο μπορώ να τον ορίσω απευθείας:

Int value = 22;

ενώ έναν πιο σύνθετο τύπο, πρέπει να χρησιμοποιήσω τον τελεστή new για να δεσμεύσω μνήμη για αυτόν:

car fiat = new car();

Αυτή η διαφορά έχει μεγάλη επίπτωση στη χρήση τους και σε τι είδους μνήμη αποθηκεύονται, κάτι που έχει με τη σειρά του επίπτωση στην ανάπτυξη βιντεοπαιχνιδιών.

Μαθηματικοί τελεστές

Υπάρχουν πέντε απλές μαθηματικές πράξεις, δύο από αυτές (+ και -), έχουν και μονή και διπλή χρήση. Ο παρακάτω πίνακας παραθέτει κάθε μία από αυτές τις λειτουργίες, μαζί με ένα σύντομο παράδειγμα της χρήσης τους και το αποτέλεσμα όταν χρησιμοποιούνται με απλούς αριθμητικούς τύπους (ακέραιους και κινητής υποδιαστολής).

Τελεστής	Παράδειγμα	Παρατηρήσεις
+	Var1 = var2 + var3;	Πρόσθεση του var2 και var3
-	Var1 = var2 - var3;	Αφαίρεση του var2 από το var3
*	Var1 = var2 * var3;	Πολ/μός του var2 και var3
/	Var1 = var2 / var3;	Διαίρεση του var2 με το var3
%	Var1 = var2 % var3;	Εύρεση υπολοίπου της διαίρεσης του var2 με το var3
+	Var1 = +var2;	Το Var1 λαμβάνει την τιμή του var2 αυξημένη κατά +1
-	Var1 = -var2;	Το Var1 λαμβάνει την τιμή του var2 μειωμένη κατά -1
++	Var1 = ++var2;	Αύξηση του var2 κατά 1 πριν από την εκχώρηση
--	Var1 = --var2;	Μείωση του var2 κατά 1 πριν από την εκχώρηση
++	Var1 = var2++;	Αύξηση του var2 κατά 1 μετά από την εκτέλεση
--	Var1 = var2--;	Μείωση του var2 κατά 1 μετά από την εκχώρηση

Τελεστές εκχώρησης

Συνήθως, χρησιμοποιούμε τον απλό τελεστή ανάθεσης, =, αλλά υπάρχουν περισσότεροι οι οποίοι είναι και αρκετά χρήσιμοι. Όλοι οι τελεστές ανάθεσης λειτουργούν με παρόμοιο τρόπο, δηλαδή οδηγούν σε μια τιμή που ανατίθεται στην μεταβλητή στην αριστερή πλευρά τους με βάση τις πράξεις στην δεξιά πλευρά τους.

Τελεστής	Παράδειγμα	Παρατηρήσεις
=	Var1 = var2;	Η var1 παίρνει την τιμή της var2
+=	Var1 += var2;	Ισοδύναμο με var1 = var1 + var2
-+	Var1 -= var2;	Ισοδύναμο με var1 = var1 - var2
*=	Var1 *=var2;	Ισοδύναμο με var1 = var1 * var2
/=	Var1 /= var2;	Ισοδύναμο με var1 = var1 / var2
%=	Var1 %= var2;	Η var1 λαμβάνει σαν τιμή το υπόλοιπο της διαίρεσης var1/var2

Λογική Boolean (Τελεστές Σύγκρισης)

Ο τύπος bool που είδαμε νωρίτερα, μπορεί να πάρει μόνο μία από τις δύο τιμές, true ή false. Αυτός ο τύπος χρησιμοποιείται συνήθως για να καταγράψει το αποτέλεσμα κάποιας λειτουργίας, έτσι ώστε να μπορέσεις να δράσεις με αυτό το αποτέλεσμα. Ειδικότερα, οι τύποι bool χρησιμοποιούνται για να αποθηκεύσουν το αποτέλεσμα μιας σύγκρισης.

Για παράδειγμα, σκεφτείτε την περίπτωση στην οποία θέλουμε να εκτελέσουμε κώδικα με βάση το αν μια μεταβλητή, myVal, είναι μικρότερη του 10. Για να γίνει αυτό, θα πρέπει να έχουμε μια ένδειξη για το αν η δήλωση, myVal είναι μικρότερη του 10, είναι αληθής ή ψευδής. Αυτό σημαίνει ότι θα πρέπει να γνωρίζουμε το Boolean αποτέλεσμα μιας σύγκρισης.

Οι Boolean συγκρίσεις απαιτούν τη χρήση λογικών τελεστών σύγκρισης, τα οποία παρουσιάζονται στον παρακάτω πίνακα. Σε όλες τις περιπτώσεις εδώ, η var1 είναι μια μεταβλητή τύπου bool, ενώ οι τύποι var2 και var3 μπορεί να διαφέρουν.

Τελεστής	**Παράδειγμα**	**Παρατηρήσεις**
==	Var1 = var2 == var3;	Το Var1 είναι αληθές αν το var2 είναι ίσο με το var3, αλλιώς είναι ψευδές
!=	Var1 = var2 != var3;	Το Var1 είναι αληθές αν το var2 **δεν** είναι ίσο με το var3, αλλιώς είναι ψευδές
<	Var1 = var2 < var3;	Το Var1 είναι αληθές αν το var2 είναι μικρότερο από το var3, αλλιώς είναι ψευδές
>	Var1 = var2 > var3;	Το Var1 είναι αληθές αν το var2 είναι μεγαλύτερο από το var3, αλλιώς είναι ψευδές
<=	Var1 = var2 <= var3;	Το Var1 είναι αληθές αν το var2 είναι μικρότερο ή ίσο με το var3, αλλιώς είναι ψευδές
>=	Var1 = var2 >= var3;	Το Var1 είναι αληθές αν το var2 είναι μεγαλύτερο ή ίσο με το var3, αλλιώς είναι ψευδές

Μπορείτε να χρησιμοποιήσετε τέτοιους τελεστές σε αριθμητικές τιμές στον κώδικα:

bool isLessThan10;
isLessThan10 = myVal < 10;

Η μεταβλητή *isLessThan10* παίρνει τιμή true εάν η myVal πάρει τιμή μικρότερη του 10, αλλιώς παίρνει τιμή false.

Μπορείτε επίσης, να χρησιμοποιήσετε τους τελεστές σύγκρισης και σε άλλους τύπους, όπως string (κείμενο):

bool isKarli;

isKarli = myString == “Karli”;

Εδώ, η μεταβλητή isKarli είναι αληθής μόνο αν η myString πάρει σαν τιμή το κείμενο “Karli”.

Καθώς και να συγκρίνετε μεταβλητές με τιμές Boolean:

bool isTrue;

isTrue = myBool == true;

Λογικοί τελεστές

Κάποιοι άλλοι τελεστές Boolean προορίζονται ειδικά για εργασία με τιμές Boolean, όπως φαίνεται στον ακόλουθο πίνακα.

Τελεστής	Παράδειγμα	Παρατηρήσεις
!	var1 = !var2;	Το var1 είναι αληθές αν το var2 είναι ψευδές (Λογικό NOT)
& ή &&	var1 = var2 & var3;	Το var1 είναι αληθές αν το var2 και το var3 μαζί είναι αληθή αλλιώς είναι ψευδές. (Λογικό AND)
| ή ||	var1 = var2 | var3;	Το var1 είναι αληθές αν το var2 ή το var3 ή και τα δύο μαζί είναι αληθή αλλιώς είναι ψευδές. (Λογικό OR)
^	var1 = var2 ^ var3;	Το var1 είναι αληθές όταν ή το var2 ή το var3 είναι αληθή (Λογικό XOR)

Επομένως, το παραπάνω παράδειγμα μπορεί να γραφτεί και έτσι:

bool isTrue;

isTrue = myBool & true;

Διακλαδώσεις

Η διακλάδωση είναι πράξη ελέγχου ποιας γραμμής του κώδικα θα εκτελεστεί επόμενη. Η γραμμή για να μεταβείτε ελέγχεται από κάποιο είδος δήλωσης υπό όρους. Η δήλωση αυτή βασίζεται σε μια σύγκριση με τη χρήση λογικής Boolean.

Υπάρχουν αρκετές τεχνικές διακλάδωσης διαθέσιμες στην C#, αλλά θα ασχοληθούμε κυρίως με την εντολή **if**.

Η εντολή if είναι ένας πιο ευέλικτος και χρήσιμος τρόπος για τη λήψη αποφάσεων μέσα στη ροή του προγράμματος. Ένα απλό παράδειγμα χρήσης της εντολής if είναι το ακόλουθο, όπου το <test> αξιολογείτε και η γραμμή του κώδικα που ακολουθεί την εντολή, εκτελείτε αν <test> είναι αληθές:

```
if (<test>)
{
 <code executed if <test> is true>;
}
```

Αφού εκτελεστεί αυτός ο κώδικας - ή δεν εκτελεστεί, αν το <test> αξιολογηθεί σαν false – η εκτέλεση του προγράμματος συνεχίζεται στην επόμενη γραμμή.

Μπορούμε επίσης, να ορίσουμε πρόσθετο κώδικα χρησιμοποιώντας την εντολή else σε συνδυασμό με μια εντολή if. Αυτή η εντολή εκτελείτε αν το <test> είναι ψευδές. Δηλαδή:

```
if (<test>)
{
 <code executed if <test> is true>;
}
 else
{
 <code executed if <test> is false>;
}
```

Επαναληπτικές διαδικασίες

Μια επαναληπτική διαδικασία (iteration statement) συγκροτεί μια σύνθετη εντολή, η οποία επαναλαμβάνει εκτελεστικά το κύριο σώμα των εντολών που περιέχει, μέχρις ότου μία συνθήκη τερματισμού ικανοποιηθεί ή κάποια ειδική εντολή απόδρασης εκτελεσθεί. Οι επαναληπτικές διαδικασίες ονομάζονται απλά επαναλήψεις (loops).

Στις αντικειμενοστραφής γλώσσες προγραμματισμού οι επαναλήψεις περιέχουν τρεις βασικές ιδιότητες:

- **Αρχική τιμή διαδικασίας (αρχικοποίηση)**, όπου δίνονται αρχικές τιμές σε ορισμένες μεταβλητές για να αρχίσουν οι επαναλήψεις.

- **Έλεγχος επανάληψης (διακόπτης),** όπου κάποια συνθήκη ή και περισσότερες ορίζουν αν θα σταματήσουν οι επαναλήψεις.

- **Μεταβολή για την επόμενη επανάληψη,** όπου προσδιορίζονται οι μεταβολές που θα ισχύουν για την επόμενη επανάληψη.

- **Αναγκαστική πρώτη εκτέλεση,** αν θα εκτελεστεί αναγκαστικά την πρώτη φορά το σώμα εκτέλεσης των εντολών της σύνθετης επαναληπτικής διαδικασίας.

Εντολή	**Αρχικοποίηση**	**Διακόπτης**	**Μεταβολή**	**Πρώτη εκτέλεση**
while	Όχι	Αρχή	Όχι	Όχι
do-while	Όχι	Τέλος	Όχι	Ναι
for	Ναι	Αρχή	Ναι	Όχι

Οι εντολές while, do-while, for είναι κατασκευαστικά ισοδύναμες μεταξύ τους, όπου κάθε μία από τις εντολές μπορεί να εκφραστεί ισοδύναμα με οποιαδήποτε από τις υπόλοιπες. Όμως, θα ασχοληθούμε κυρίως με την εντολή for.

Οι αντικειμενοστραφής γλώσσες προγραμματισμού διαθέτουν μία τουλάχιστον εντολή **for**, όπου κάποια μεταβλητή ελέγχου παίρνει μια αρχική τιμή, επιδρά σε μία επικράτεια εντολών και στη συνέχεια παίρνει μια επόμενη τιμή μέχρις ότου η μεταβλητή αυτή ξεπεράσει ένα ανώτερο φράγμα, οπότε γίνεται και απόδραση της επαναληπτικής διαδικασίας. Η εντολή for έχει γενικό συντακτικό τύπο:

```
for (αρχικοποίηση;συνθήκη;ενημέρωση;)
    {
        επικράτεια εντολών;
    }
```

6. Visual Studio

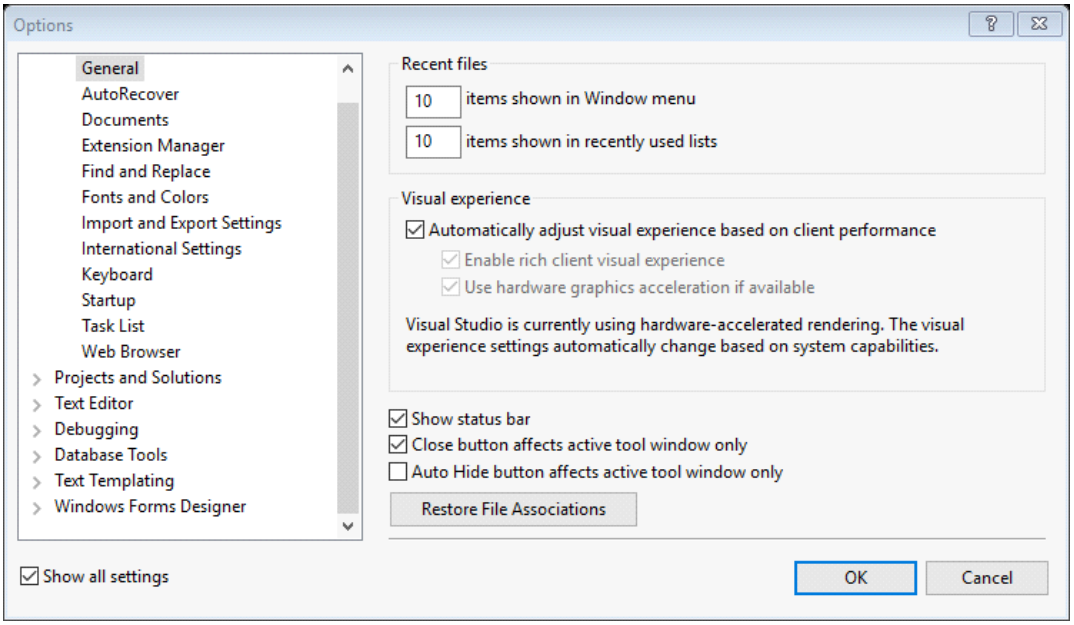
Το Visual Studio 2010 (Vs) είναι ολοκληρωμένο περιβάλλον ανάπτυξης εφαρμογών. Μπορεί να χρησιμοποιηθεί από απλές εφαρμογές γραμμής εντολών, μέχρι και πιο σύνθετους τύπους προγραμμάτων.

Εκτός από το VS η Microsoft παρέχει κι άλλα εργαλεία απλούστερης ανάπτυξης γνωστά ως Visual Studio Express Products που μπορείτε να τα προμηθευτείτε δωρεάν από το **http://www.microsoft.com/express.**

Ένα από αυτά τα προϊόντα, Visual C# 2010 Express, είναι και αυτό με το οποίο θα ασχοληθούμε παρακάτω αναλυτικά, καθώς και αυτό που χρησιμοποιήθηκε για την υλοποίηση των παραδειγμάτων εκμάθησης. Με το προϊόν αυτό μπορούμε να δημιουργήσουμε σχεδόν κάθε εφαρμογή σε C# που μπορεί να χρειαστούμε. Είναι μια συρρικνωμένη έκδοση του VS, που διατηρεί την ίδια εμφάνιση και αίσθηση.

Βασικές Ρυθμίσεις

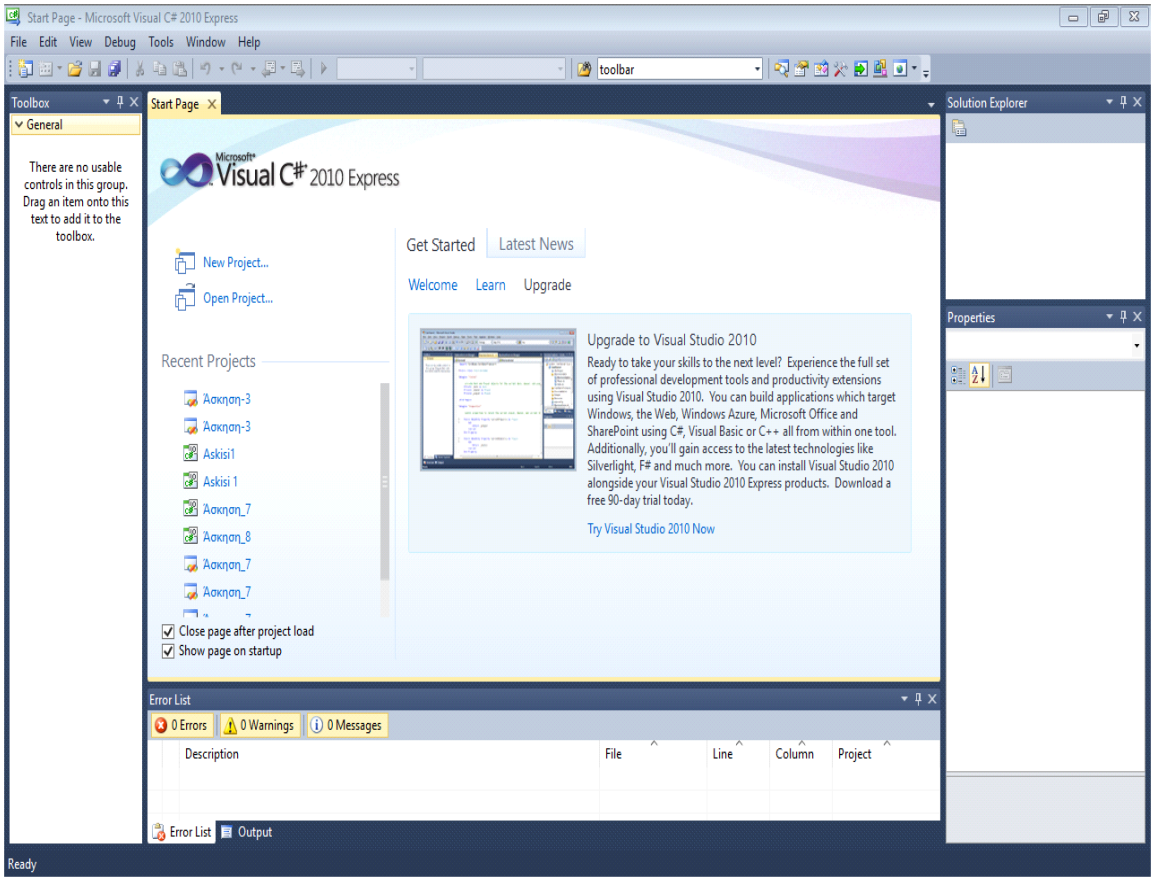
Κάνοντας εγκατάσταση του προγράμματος στον υπολογιστή μας θα πρέπει να κάνουμε κάποιες ρυθμίσεις. Πάμε στο μενού *Tools> Settings> Expert Settings*. Έπειτα, *Tools> Options* και τσεκάρουμε το *Show all settings*. Απ' το μενού που εμφανίζεται ανοίγουμε το *Projects and Solutions*, επιλέγουμε το *generals* και τσεκάρουμε το *Save new projects when created.*



Στη συνέχεια προσθέτουμε τα βασικά παράθυρα που θα χρειαστούμε. Αυτά είναι το *Toolbox*, ο *Solution Explorer*, το *Properties Window* και το *Error List*. Και τα τέσσερα θα τα βρούμε στο View. Διαμορφώνουμε τον χώρο εργασίας μας κατάλληλα, όπως φαίνεται και στην επόμενη σελίδα.

Δημιουργία μιας απλής Windows Form

Ανοίγοντας το πρόγραμμα εμφανίζεται αυτό το παράθυρο στην οθόνη.



Παρατηρούμε τα εξής:

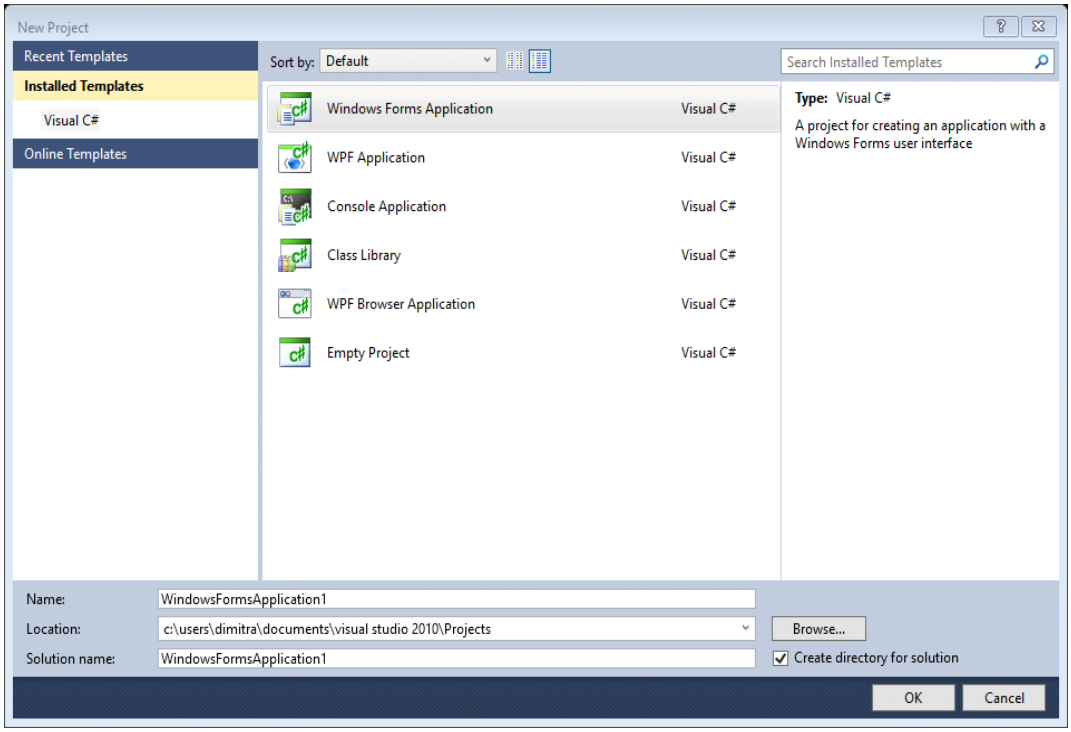
- Στο αριστερό μέρος της οθόνης το παράθυρο **Toolbox**. Εκεί θα βρούμε στη συνέχεια όλα τα απαραίτητα αντικείμενα που θα χρειαστούμε στις εφαρμογές μας.

- Δεξιά τα παράθυρα **Properties** και **Solution Explorer**. Στο Properties θα εμφανίζονται οι ιδιότητες του αντικειμένου όταν το έχουμε επιλέξει, ενώ στο Solution Explorer περιέχονται όλα τα στοιχεία του project.

- Τέλος, στο κάτω μέρος της οθόνης βλέπουμε το **Error List** όπου εκεί θα εμφανίζονται τα λάθη στον κώδικά μας όταν προχωράμε σε debug.

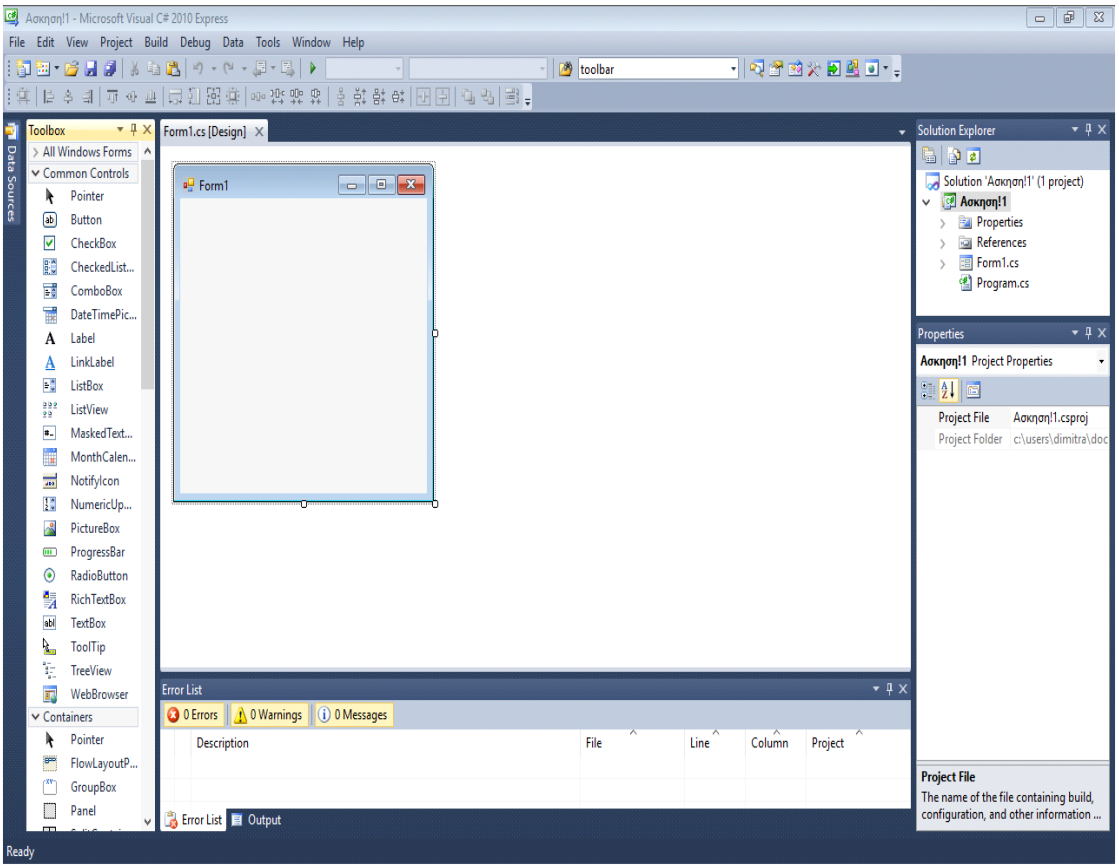
Πάμε να δούμε αναλυτικά την πρώτη εφαρμογή μας για να καταλάβουμε πως λειτουργεί το περιβάλλον της Microsoft Visual.

Από την αρχική οθόνη επιλέγουμε *File > New Project*. Ανοίγει το επόμενο παράθυρο



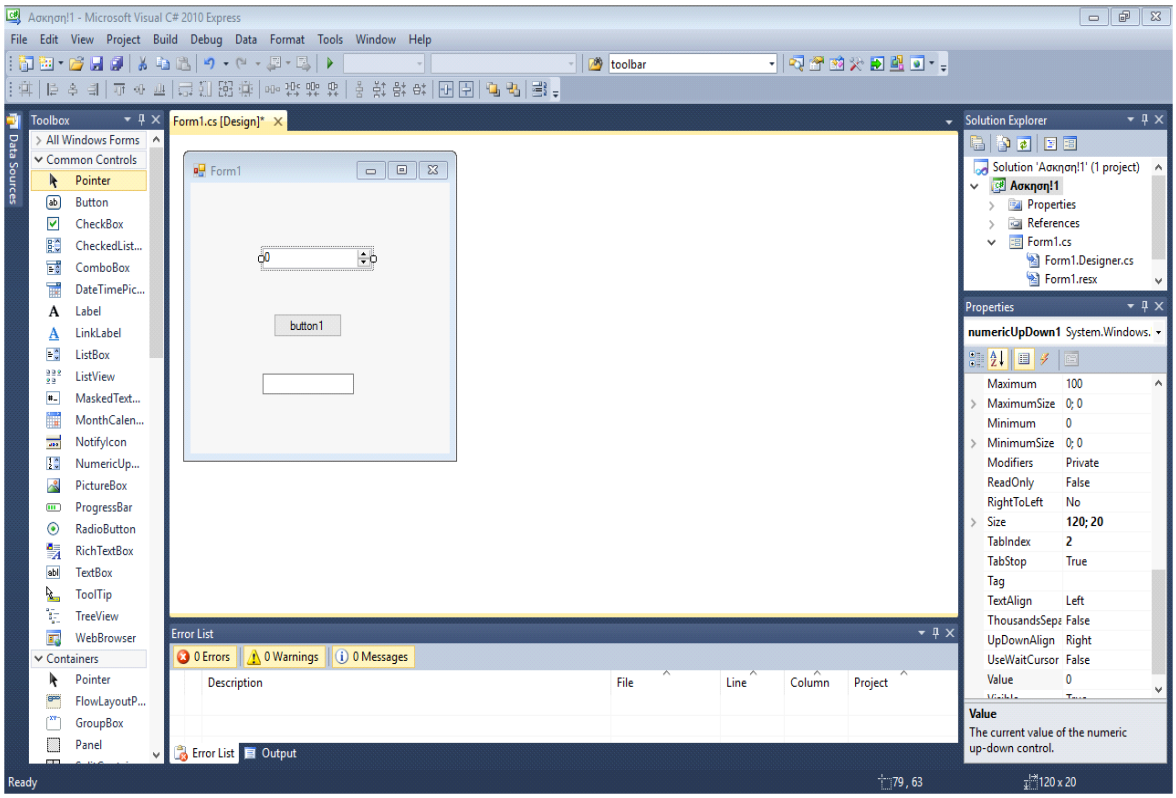
όπου επιλέγουμε το *Windows Form Application* με το οποίο θα ασχοληθούμε σε αυτό το βιβλίο. Πριν πατήσουμε το OK, συμπληρώνουμε τα πεδία που βρίσκονται στο κάτω μέρος του παραθύρου, με το όνομα που επιθυμούμε και την τοποθεσία που θέλουμε να αποθηκεύσουμε την εφαρμογή μας. Στο πεδίο Solution Name συμπληρώνεται αυτόματα το όνομα. Τέλος, τσεκάρουμε το *Create directory for solution* και πατάμε OK.

Εμφανίζεται το επόμενο παράθυρο με την *Form1* όπου θα αναπτύξουμε την εφαρμογή μας.

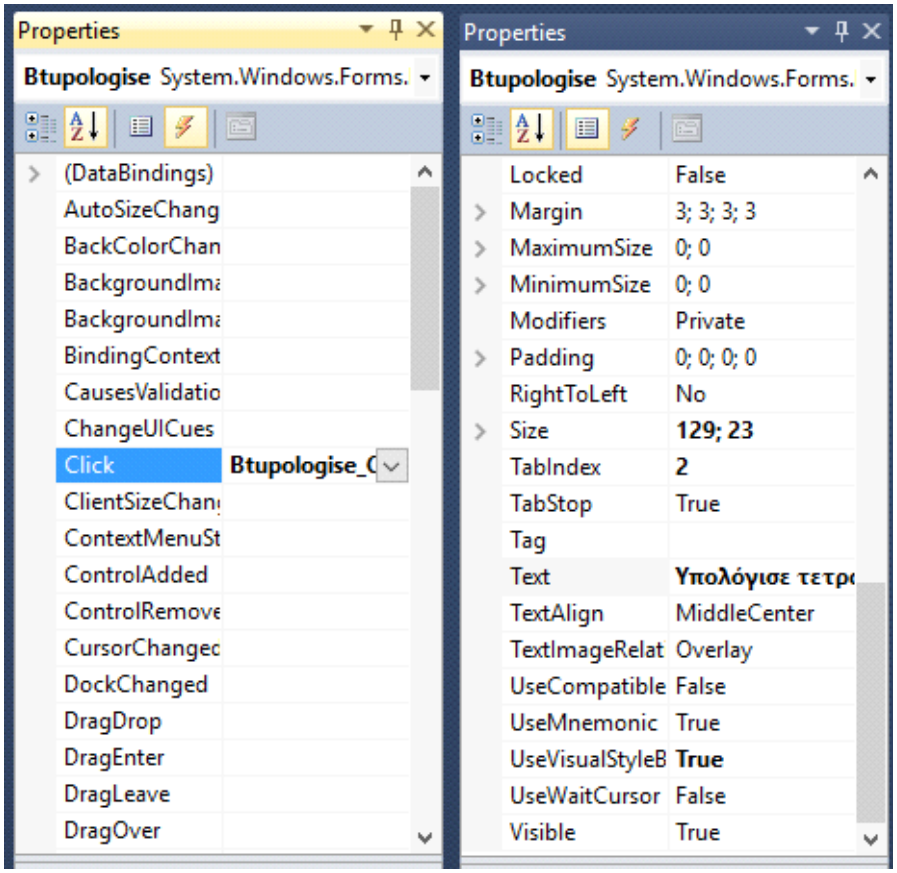


Σκοπός μας είναι να φτιάξουμε μια απλή εφαρμογή όπου ο χρήστης θα εισάγει μία τιμή και όταν πατήσει ένα μπουτόν να εμφανίζεται το τετράγωνό του.

Όπως αναφέρθηκε και παραπάνω, οτιδήποτε χρειαζόμαστε θα το βρούμε στο *Toolbox*. Συγκεκριμένα, για να εισάγει ο χρήστης τον αριθμό θα προσθέσουμε ένα nymericUpDown. Ένα *Button* για τον υπολογισμό του τετραγώνου και τέλος, ένα *Textbox* για την εμφάνιση του αποτελέσματος. Τα κλικάρω και τα βάζω στην αρχική μου φόρμα που θα έχει την εξής μορφή.



Παρατηρούμε ότι όλα τα αντικείμενα έχουν τα αρχικά τους ονόματα. Ακόμα και η *Form1*. Για να τα αλλάξουμε, επιλέγουμε κάθε ένα χωριστά και πηγαίνοντας στα *properties* στο πεδίο *Name*, βάζουμε αυτό που μας ζητείται. Θα πρέπει να κατανοήσουμε και να μην συγχέουμε το όνομα των αντικειμένων με το εμφανιζόμενο κείμενο σε αυτά. Αν για παράδειγμα στο *button1* θέλουμε να αλλάξουμε το κείμενο, αυτό δεν θα γίνει αλλάζοντας το όνομά του. Το όνομα του αντικειμένου πρέπει να είναι πάντα στα αγγλικά γιατί χρησιμοποιείτε στον κώδικά μας. Για να αλλάξουμε λοιπόν το εμφανιζόμενο κείμενο, επιλέγουμε το αντικείμενο και πηγαίνουμε στα *properties*. Βρίσκουμε το *Text* και εκεί πληκτρολογούμε το κείμενο που θέλουμε.



Εάν από τα *properties* επιλέξω το εικονίδιο του κεραυνού, βλέπω τα συμβάντα (Events) που αφορούν το αντικείμενο. Επιλέγοντας κάποιο από τα συμβάντα αυτά, πχ. «αν αλλάξει τιμή», «αν γίνει κλικ του ποντικιού» κτλ, μπορούμε μετά να γράψουμε κώδικα που θα εκτελείται γι' αυτό το συμβάν.

Προσθήκη κώδικα

Πάμε να προσθέσουμε κώδικα στην εφαρμογή μας. Για να εμφανιστεί το παράθυρο του κώδικα, κάνουμε διπλό κλικ πάνω στη φόρμα μας. Τότε, αυτόματα μεταφερόμαστε στον κώδικα της άσκησης που σε αρχική μορφή είναι κάπως έτσι:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace Ασκηση_1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {

        }
    }
}
```

Πολλά από τα χαρακτηριστικά ενός C# προγράμματος εμφανίζονται ήδη. Οι δηλώσεις **using** στην αρχή του προγράμματος καθορίζουν ποιες βιβλιοθήκες θα χρησιμοποιήσουμε στην εφαρμογή. Το **namespace** ορίζει έναν ¨χώρο¨ μέσα στον οποίο θα δημιουργούνται οι μεταβλητές στο πρόγραμμα. Είναι για καθαρά ονοματολογικούς σκοπούς και επιτρέπει τον διαχωρισμό δύο μεταβλητών με το ίδιο όνομα που κάθε μία ανήκει σε διαφορετικές βιβλιοθήκες. Μπορεί να το δει κανείς σαν το όνομα ενός φακέλου (folder) στο δίσκο. Δύο αρχεία με το ίδιο όνομα μπορούν να υπάρχουν σε δύο διαφορετικούς φακέλους. Για να κάνω αναφορά σε κάποιο από τα δύο χρησιμοποιώ και το όνομα του φακέλου. Το ίδιο ακριβώς συμβαίνει και με το namespace.

Σε αυτή τη μορφή δεν μπορεί να εκτελέσει καμία ενέργεια. Εμείς θέλουμε, κάθε φορά που γίνεται κλικ στο button να γίνεται υπολογισμός του τετραγώνου. Όποτε

πάμε στην φόρμα μας και κάνουμε διπλό κλικ πάνω στο button. Αμέσως δημιουργείται ένα νέο συμβάν όπου εκεί θα γράψουμε τον κώδικα για τον υπολογισμό.

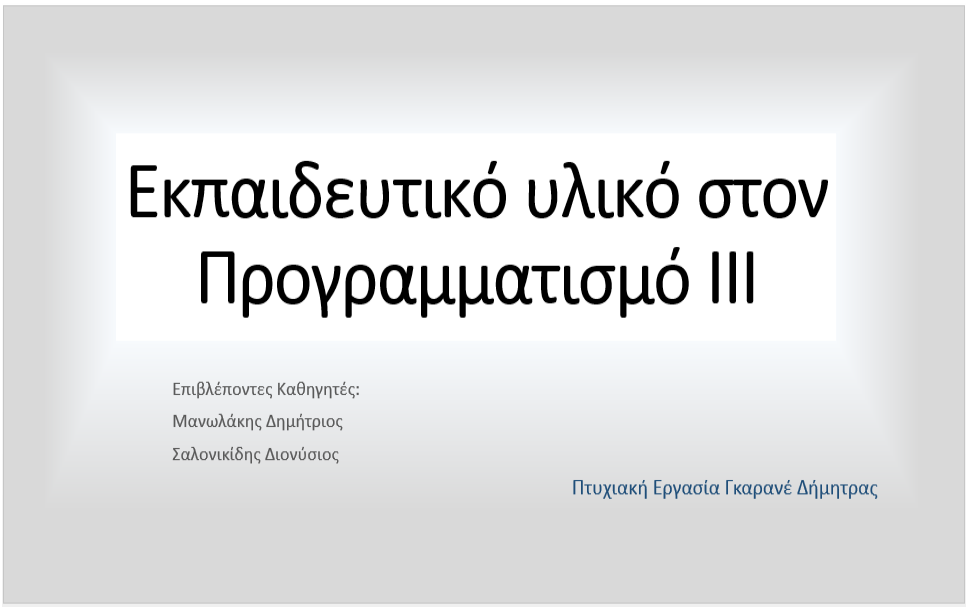
```
private void Btupologise_Click(object sender, EventArgs e)
    {
       decimal k;
       k = nUDarithmos.Value * nUDarithmos.Value;
       tBapotelesma.Text = Convert.ToString(k);
    }
```

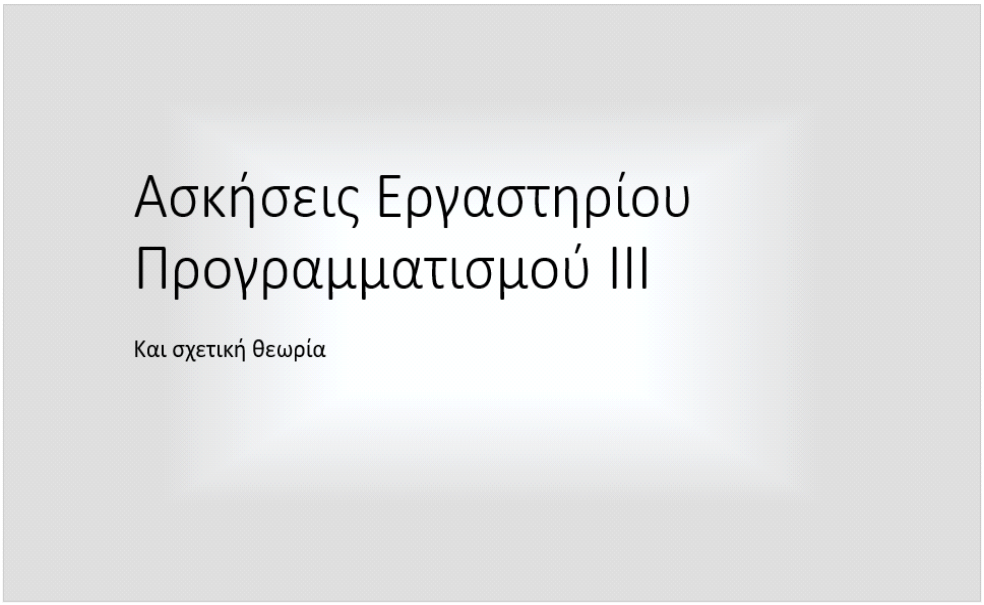
Εκτέλεση του project

Για να εκτελέσουμε το project πρώτα κάνουμε μεταγλώττιση και σύνδεση επιλέγοντας *Build> Build Solution*. Αν δεν εμφανιστεί κανένα λάθος, τότε προχωρούμε στην εκτέλεση. Ο απλούστερος τρόπος είναι από το *Debug> Start Debugging* ή με το εικονίδιο

7. Δημιουργία ηλεκτρονικού εκπαιδευτικού υλικού

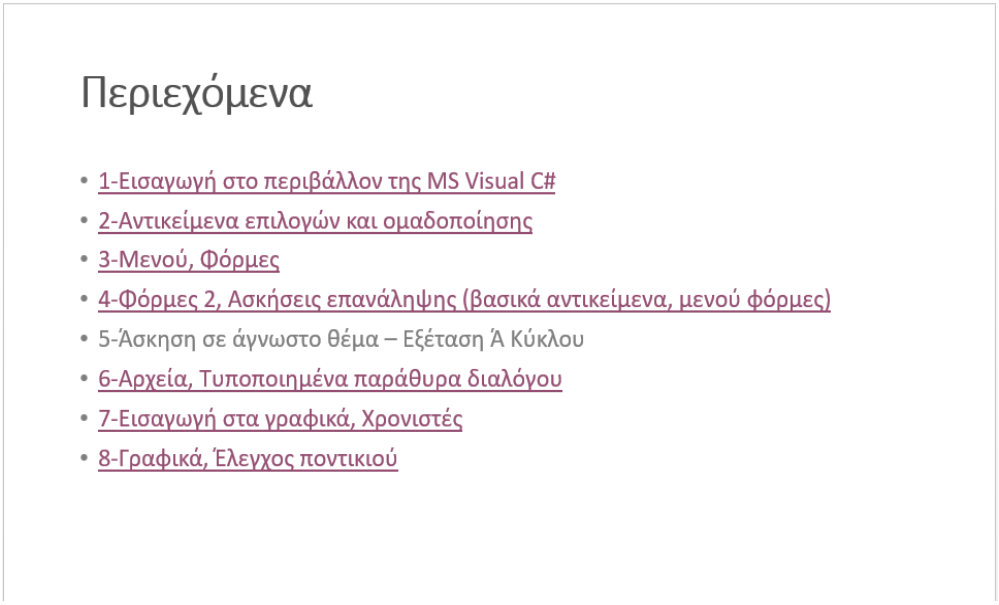
Στο δεύτερο μέρος της εργασίας ακολουθεί η δημιουργία του Power Point το οποίο βοηθά τον σπουδαστή βήμα βήμα στην εκπόνηση της κάθε εργαστηριακής άσκησης. Σκοπό του είναι να χρησιμοποιηθεί ως βοηθητικό εργαλείο για την βαθύτερη κατανόηση του προγραμματισμού.





Στις δύο πρώτες εικόνες ο μαθητευόμενος πληροφορείται το θέμα του υλικού, από ποιόν σχεδιάστηκε και ότι πρόκειται για τις ασκήσεις του εργαστηρίου του μαθήματος προγραμματισμού.

Στην επόμενη εικόνα παρουσιάζονται τα περιεχόμενα και ο διαχωρισμός τους σε ενότητες. Μόλις ο σπουδαστής πατήσει πάνω σε κάποια από αυτές τότε μεταφέρεται αυτόματα στην αντίστοιχη σελίδα.



Συμπεράσματα

Η ηλεκτρονική εκπαίδευση αποτελεί ένα σημαντικό βοήθημα στα χέρια των σπουδαστών. Το εκπαιδευτικό υλικό είναι πάντα και παντού διαθέσιμο και μπορεί να υποστηρίξει μεγάλο αριθμό ατόμων. Αποτελεί επίσης ένα οικονομικό μέσο εκμάθησης.

Η συγκεκριμένη πτυχιακή εργασία με το θεωρητικό μέρος, το ηλεκτρονικό εκπαιδευτικό υλικό και τα τελικά αποτελέσματα των εκτελέσιμων αρχείων των εφαρμογών διατίθεται στους χρήστες της με σκοπό την γρήγορη εκμάθηση της Visual C# και ελπίζει αυτό να επετεύχθη.

Βιβλιογραφία

- https://en.wikipedia.org/wiki/Main_Page
- https://msdn.microsoft.com/en-us/library/aa287558(v=vs.71).aspx
- <<Beginning Visual C# 2010>> ,Karli Watson, Christian Nagel, Jacob Hammer Pedersen, Jon D. Reid, Morgan Skinner – wrox.com
- Ευάγγελος Γ. Ούτσιος <Αντικειμενοστραφής Προγραμματισμός> - Τεχνολογικό Εκπαιδευτικό Ίδρυμα Σερρών Σχολή Τεχνολογικών Εφαρμογών Τμήμα Πληροφορικής και Επικοινωνιών – 2004
- C για μηχανικούς – εκδόσεις Τζιόλα
- Υπουργείο εθνικής παιδείας & θρησκευμάτων – Γενική γραμματεία εκπαίδευσης ενηλίκων - <<Η εξ αποστάσεως εκπαίδευση στην εκπαίδευση ενηλίκων>>
- Κοντογεώργου Δήμητρα – Μασαλά Χρυσάνθη << Δημιουργία εφαρμογής βάσης δεδομένων παροχής πληροφοριών σε κατάστημα ενδυμάτων για την εξυπηρέτηση πελατών >> Ανώτατο Εκπαιδευτικό Ίδρυμα Ανατολικής Μακεδονίας και Θράκης σχολή τεχνολογικών εφαρμογών τμήμα μηχανικών πληροφορικής – 2014