Κεφάλαιο 4ο

Αλγόριθμοι – Λογισμικό

4.1 Αλγόριθμοι - Προγράμματα

Η αντιμετώπιση και η επίλυση καθημερινών αλλά και άλλων πιο σύνθετων προβλημάτων, απαιτεί την εκτέλεση λογικών βημάτων, σε μια προκαθορισμένη σειρά, τα οποία οδηγούν σε ένα αποτέλεσμα. Πολλές φορές οι άνθρωποι είτε εκούσια είτε ακούσια πραγματοποιούν τέτοιου είδους διεργασίες για την αντιμετώπιση των πιο απλών ή των πιο σύνθετων εργασιών τους. Η εισαγωγή των Ηλεκτρονικών Υπολογιστών στην καθημερινότητα του ανθρώπου είχε ως στόχο κυρίως την επίλυση προβλημάτων που χαρακτηρίζονται από επαναληψιμότητα ή από δύσκολους μαθηματικούς υπολογισμούς. Γίνεται άμεσα αντιληπτό ότι η ανάγκη για την καταγραφή ενός προβλήματος και των απαραίτητων βημάτων που απαιτούνται για την επίλυση του, επιβάλλεται ακόμα περισσότερο λόγω της «μηχανιστικής» λειτουργίας των Ηλεκτρονικών Υπολογιστών. Επομένως ορίζεται ο Αλγόριθμος (algorithm) ως «μια σειρά στοιχειωδών διαδικασιών, οι οποίες νοούνται ότι απευθύνονται προς σειριακή (βήμα προς βήμα) εκτέλεση από κάποιο ιδεατό μηχανικό σύστημα».

Η σειρά των εργασιών για την δημιουργία ενός αλγορίθμου έχουν ως εξής:

- Καλή κατανόηση του προβλήματος

- Ανάλυση των δομών και τυχών υπο-προβλημάτων που εμπεριέχονται στο γενικό πρόβλημα
- Καθορισμός των δεδομένων εισόδου και των δεδομένων εξόδου (input/output)
- Δημιουργία του κυρίου μέρους του αλγορίθμου με την καταγραφή των βημάτων επίλυσης και παραγωγής των αποτελεσμάτων (διασύνδεση των δεδομένων εισόδου με τα δεδομένα εξόδου / αποτελέσματα)

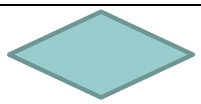
Τα βασικά χαρακτηριστικά ενός σωστού αλγορίθμου είναι τα εξής :

- Ακριβής περιγραφή των δεδομένων εισόδου καθώς και των αναμενόμενων αποτελεσμάτων.
- Τα βήματα του αλγορίθμου θα πρέπει να είναι ξεκάθαρα, ο αριθμός τους πεπερασμένος και να ολοκληρώνονται σε καθορισμένο και όχι άπειρο χρόνο.
- Επιπλέον, τόσο το κάθε βήμα όσο και ο αλγόριθμος συνολικά θα πρέπει να οδηγεί σε ακριβή αποτελέσματα (όχι διφορούμενα) και για τα ίδια δεδομένα εισόδου να παράγονται τα ίδια δεδομένα εξόδου.

4.1.1 Λογικό Διάγραμμα Ροής (*flowchart*) – Ψευδοκώδικας (*pseudocode*)

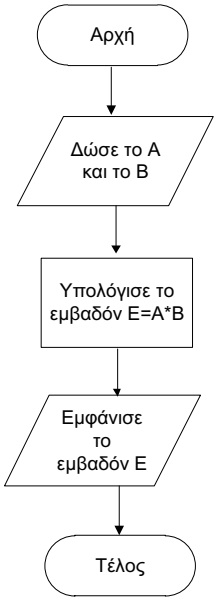
Ένας σχηματικός τρόπος παράστασης του αλγόριθμου είναι το **λογικό διάγραμμα**. Αυτό παρουσιάζει, με τη χρήση διαφόρων σχημάτων, τα είδη των διαδικασιών που πρέπει να ακολουθηθούν για την επίλυση του προβλήματος. Τα σχήματα αυτά συνδέονται μεταξύ τους με βέλη που δείχνουν τη λογική σειρά - πορεία επίλυσης του προβλήματος. Τα συνηθέστερα σχήματα, μαζί με τις στοιχειώδεις διαδικασίες που παριστάνουν αναλύονται στον πίνακα που ακολουθεί.

Πίνακας 4.1. *Συμβολισμός σχημάτων διαγραμμάτων ροής*

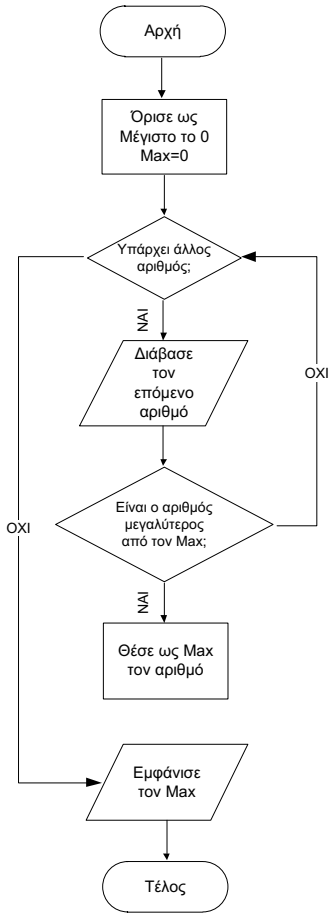
Σχήμα – διαδικασία	**Σύμβολο**
Οβάλ: αρχή και τέλος του προγράμματος	
Πλάγιο παραλληλόγραμμο: εισαγωγή / εξαγωγή δεδομένων – Input / Output	
Ορθογώνιο: υπολογισμός μιας αριθμητικής έκφρασης (εντολές εκχώρησης)	
Ρόμβος: λήψη αποφάσεων (εντολές ελέγχου)	
Κύκλος: συνέχιση του λογικού διαγράμματος σε άλλη σελίδα	

Το διάγραμμα ροής προσφέρει την «οπτικοποίηση» των βημάτων ενός προβλήματος, εκτός των πολύ σύνθετων περιπτώσεων όπου είναι δύσκολη η δημιουργία του. Μία ακόμα χρήσιμη αναπαράσταση του αλγορίθμου είναι ο Ψευδοκώδικας (*pseudocode*). Ο ψευδοκώδικας είναι η αναπαράσταση ενός αλγορίθμου σε φυσική γλώσσα. Στην συνέχεια παρουσιάζονται δυο παραδείγματα αλγορίθμων, των αντίστοιχων διαγραμμάτων ροής αλλά και της αναπαράστασης σε ψευδοκώδικα.

Πίνακας 4.2. *Υπολογισμός Εμβαδού Τετραγώνου/Ορθογωνίου*

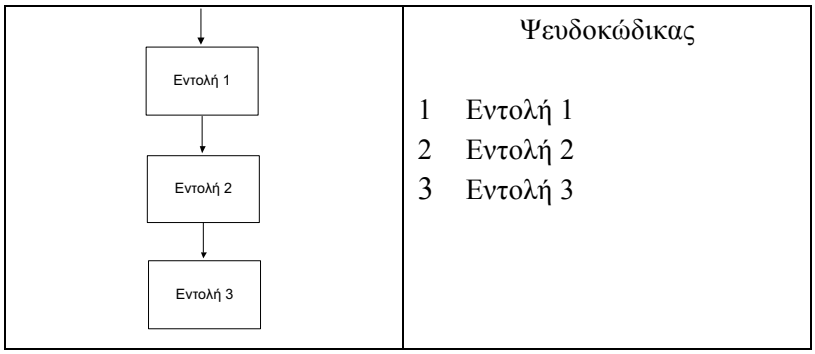
Διάγραμμα Ροής	**Ψευδοκώδικας**
	**1.** Αρχή **2.** Διάβασε το Α και το Β **3.** Υπολόγισε το εμβαδόν, Ε=Α*Β **4.** Εμφάνισε το εμβαδόν Ε **5.** Τέλος

Πίνακας 4.3. *Εύρεση Μέγιστου Αριθμού*

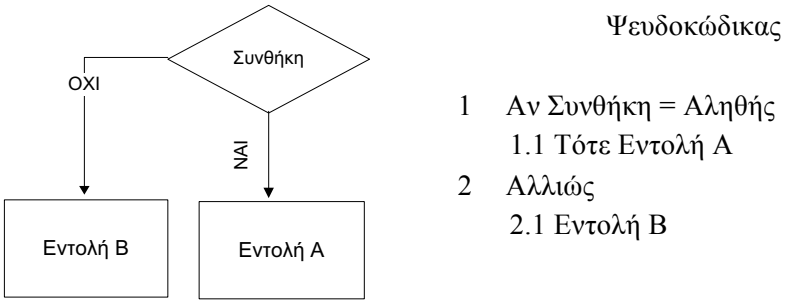
Διάγραμμα Ροής	**Ψευδοκώδικας**
	**1.** Αρχή **2.** Όρισε **<u>Μέγιστο</u>** το 0 (μηδέν) – max=0 **3.** Αν υπάρχει και άλλος αριθμός συνέχισε 3.1 Διάβασε τον επόμενο αριθμό. 3.2 Αν (ο αριθμός είναι μεγαλύτερος από το Μέγιστο) τότε 3.2.1 Όρισε ως Μέγιστο την τιμή του αριθμού 3.3 Τέλος Αν **4.** Τέλος Αν **5.** Εμφάνισε το Μέγιστο **6.** Τέλος

4.1.2 Αλγοριθμικές Δομές

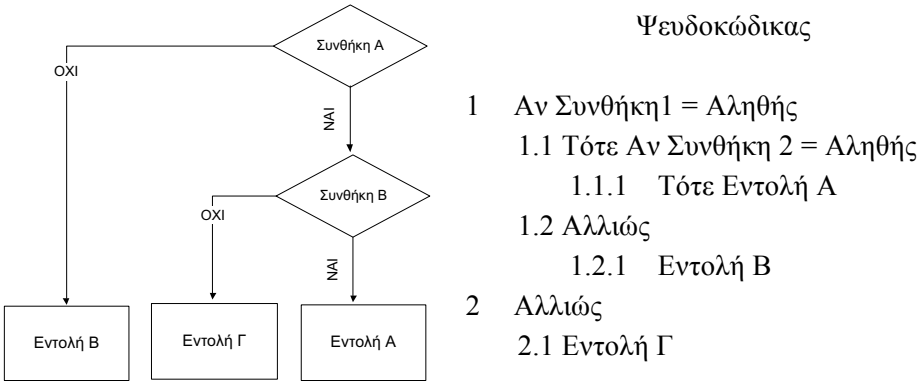
Οι βασικές αλγοριθμικές δομές είναι η ακολουθία ή διαδοχή, η επιλογή και η επανάληψη. Οι δύο πρώτες περιπτώσεις αλγοριθμικών δομών περιέχονται στα παραδείγματα της προηγούμενης παραγράφου. Στην συνέχεια και για την καλύτερη κατανόηση τους, καθώς και της συγγραφής αλγορίθμων σε μορφή ψευδοκώδικα ακολουθεί αναλυτική παρουσίαση των αλγοριθμικών δομών και των παραλλαγών τους.



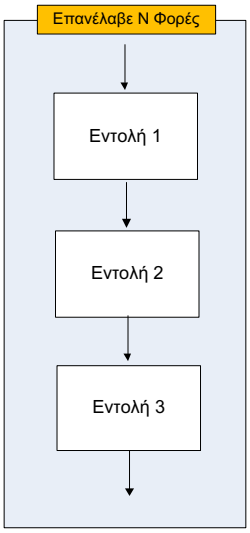
Εικόνα 4.1. *Ακολουθία ή διαδοχή εντολών*



Εικόνα 4.2. *Επιλογή (Αν ... τότε ... αλλιώς)*



Εικόνα 4.3. *Φωλιασμένη (nested) Επιλογή (Αν ... τότε ...αν ... αλλιώς)*



Ψευδοκώδικας

```
1   Αν Συνθήκη1 = Αληθής
    1.1 Τότε Αν Συνθήκη 2 = Αληθής
        1.1.1   Τότε Εντολή Α
    1.2 Αλλιώς
        1.2.1   Εντολή Β
2   Αλλιώς
    2.1 Εντολή Γ
```

Εικόνα 4.4. *Επανάληψη εντολών*

4.2 Λογισμικό (Software)

Μαζί με την εμφάνιση των Ηλεκτρονικών Υπολογιστών, ως υλικό (*hardware*), αναπτύχθηκε και ο τομέας της παραγωγής λογισμικού (*software*), δηλαδή των προγραμμάτων που θέτουν σε λειτουργία τα διάφορα μέρη και μονάδες, ενώ παράλληλα είναι υπεύθυνα για την εκτέλεση των εφαρμογών. Όπως γίνεται αντιληπτό, δυο είναι οι μεγάλες κατηγορίες λογισμικού στο χώρο των Ηλεκτρονικών Υπολογιστών (ανεξάρτητα από την κατηγορία τους), το λογισμικό του συστήματος ή λειτουργικό σύστημα (*operating system*) και τα λογισμικά των εφαρμογών (*application software*). Η ανάπτυξη λογισμικών, ανεξάρτητα από το σε ποια κατηγορία ανήκουν, στηρίχθηκε στις λεγόμενες γλώσσες προγραμματισμού, στις οποίες γράφονταν οι εντολές των εφαρμογών και μετατρέπονταν στη συνέχεια σε μια κατανοητή για τον υπολογιστή μορφή. Η παραγωγή λογισμικού, ως δραστηριότητα στο χώρο των Ηλεκτρονικών Υπολογιστών, έχει σήμερα ξεπεράσει σε δυναμικότητα την παραγωγή υλικού και θεωρείται από τους πιο κερδοφόρους και ταχύτατα αναπτυσσόμενους παραγωγικούς κλάδους της οικονομίας, οδηγώντας σε μια σειρά από νέα επαγγέλματα και ειδικότητες.

Στη συνέχεια του κεφαλαίου παρουσιάζονται αναλυτικά οι παραπάνω έννοιες, ενώ στο δεύτερο μέρος του βιβλίου πραγματοποιείται μια εκτενής περιγραφή των βασικών βιβλιοθηκονομικών εφαρμογών, οι οποίες στηρίζονται σε ηλεκτρονικούς υπολογιστές.

4.2.1 Λειτουργικό Σύστημα (Operating System)

Λειτουργικό Σύστημα (ΛΣ) είναι μια συλλογή προγραμμάτων που ελέγχουν και συντονίζουν τις λειτουργίες ενός Ηλεκτρονικού Υπολογιστή. Πιο συγκεκριμένα, τα προγράμματα αυτά χρησιμοποιούνται για τη διαχείριση των μονάδων, όπως μονάδες εισόδου / εξόδου, κεντρική και περιφερειακή μνήμη, κ.λπ., καθώς επίσης και για τη δημιουργία κατάλληλου περιβάλλοντος λειτουργίας προγραμμάτων και εφαρμογών.

Το λειτουργικό σύστημα επιτρέπει στον Ηλεκτρονικό Υπολογιστή να «επιβλέπει» και να συντονίζει αυτόματα – χωρίς την παρέμβαση του χρήστη – τις λειτουργίες του. Ένα απλό παράδειγμα είναι οι διαδικασίες που συμβαίνουν στο παρασκήνιο (*background*) για την εμφάνιση ενός χαρακτήρα που πατιέται στο πληκτρολόγιο. Είναι γνωστό ότι ο Ηλεκτρονικός Υπολογιστής δεν είναι σε θέση να κάνει τίποτα αν δεν διαθέτει το κατάλληλο πρόγραμμα. Επομένως ακόμη και για αυτήν, τη φαινομενικά απλή ενέργεια, απαιτείται ένα ειδικό λογισμικό που να ανταποκρίνεται στην αίτηση διακοπής που έρχεται από το πληκτρολόγιο. Το λογισμικό αυτό, γραμμένο συνήθως σε μια γλώσσα χαμηλού επιπέδου (*συμβολική γλώσσα με άμεση αντιστοίχιση στη γλώσσα μηχανής*), είναι ένα από τα πολλά ειδικά προγράμματα του λειτουργικού συστήματος.

Το εν λόγω λογισμικό αποτελείται από εντολές που εκτελούνται χωρίς να γίνονται αντιληπτές από το χρήστη αλλά και από εντολές προσιτές στον χρήστη, όπως οι εντολές διαχείρισης της περιφερειακής μνήμης. Το λειτουργικό σύστημα συγκροτείται από έναν πυρήνα (*kernel*) ή εποπτεύον πρόγραμμα (*supervisor*) που παραμένει συνεχώς στη μνήμη για την εκτέλεση βασικών λειτουργιών αλλά και την κλήση στη μνήμη άλλων τμημάτων του, όποτε αυτά χρειαστούν.

4.2.1.1 Υπηρεσίες του Λειτουργικού Συστήματος

Οι βασικές υπηρεσίες ενός λειτουργικού συστήματος είναι οι εξής:

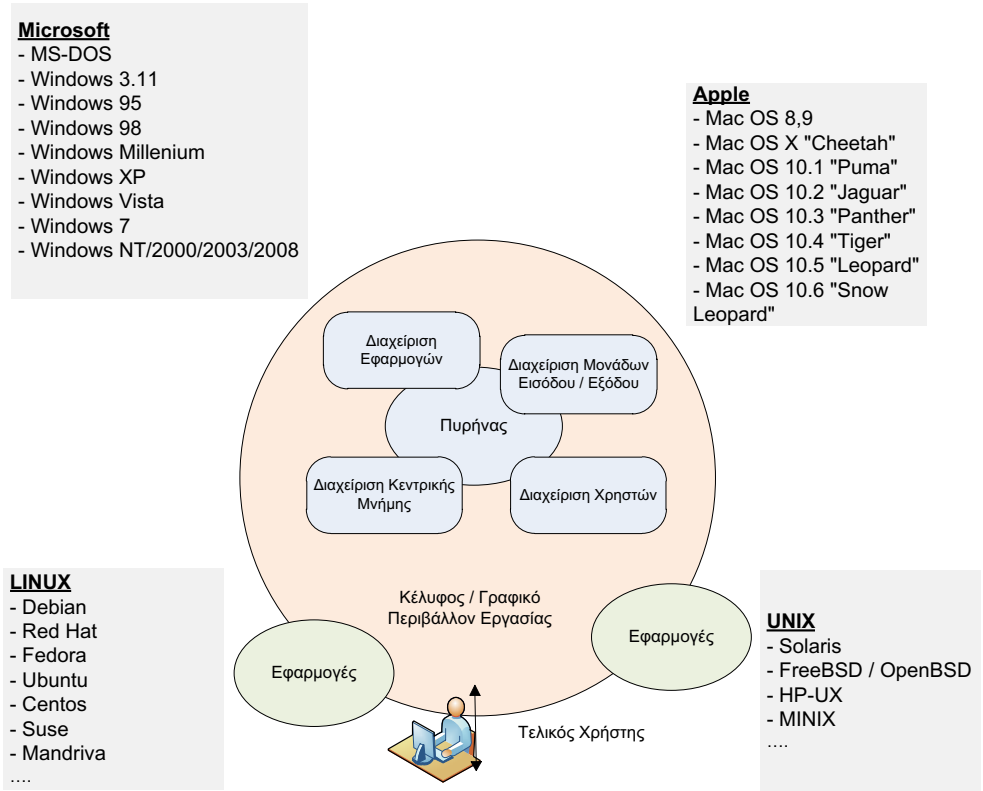
- ***Εκκίνηση του Ηλεκτρονικού Υπολογιστή:*** Απαραίτητη προϋπόθεση για τη λειτουργία του Ηλεκτρονικού Υπολογιστή είναι η αρχική φόρτωση και η συνεχής παρουσία του πυρήνα του λειτουργικού συστήματος στην κεντρική μνήμη (RAM) του υπολογιστή.. Αυτό επιτυγχάνεται μέσα από μια εξειδικευμένη εφαρμογή (ρουτίνα) αποθηκευμένη στη μνήμη ROM, η οποία εκτελείται μόλις τεθεί το σύστημα σε λειτουργία. Πρώτη ενέργεια αυτής της ρουτίνας είναι ο έλεγχος της μνήμης. Αν τα αποτελέσματα του ελέγχου είναι επιτυχή, τότε μεταφέρεται από το σκληρό δίσκο (περιφερειακή μνήμη) ένα μικρό τμήμα του λειτουργικού συστήματος, το πρόγραμμα εκκίνησης. Αυτό στη συνέχεια φορτώνει στην κεντρική μνήμη το υπόλοιπο μέρος του πυρήνα. Τότε πλέον εμφανίζεται το γραφικό περιβάλλον με την οθόνη πιστοποίησης των χρηστών, τα εικονίδια και διάφορες εφαρμογές. Από το σημείο αυτό και

μετά ο χρήστης μπορεί να εκτελέσει κάποια από τα προγράμματα του λειτουργικού συστήματος ή κάποια εφαρμογή.

- ***Διαχείριση της κεντρικής μνήμης:*** Το λειτουργικό σύστημα κάθε στιγμή εξετάζει τη διαθέσιμη χωρητικότητα της κεντρικής μνήμης και τοποθετεί στην κατάλληλη θέση τα προγράμματα που τρέχουν.
- ***Διαχείριση της περιφερειακής μνήμης:*** Το λειτουργικό σύστημα είναι ενημερωμένο για την κατάσταση των αρχείων του συστήματος του μόνιμου μέσου αποθήκευσης (δες *2.5.6.2 Περιφερειακή Μνήμη - Μαγνητικοί δίσκοι*). Διαχειρίζεται τους αποθηκευτικούς χώρους (directories / folders) και τα αρχεία με βάση τις εντολές του χρήστη. Αναζητά τους ελεύθερους χώρους που υπάρχουν στο μέσο και τους συνδέει λογικά ώστε να αποθηκεύει το κάθε αρχείο σε ένα φαινομενικά ενιαίο χώρο.
- ***Έλεγχος των μονάδων εισόδου / εξόδου:*** Ελέγχει και συντονίζει τη ροή των δεδομένων μεταξύ των διαφόρων συσκευών εισόδου / εξόδου και της ΚΜΕ σε συνεργασία με το βασικό υποσύστημα εισόδου / εξόδου που αποτελεί μέρος της Μονάδας Έλεγχου.
- ***Επεξεργασία εντολών και εφαρμογών:*** Είναι υπεύθυνο για την εκτέλεση των βοηθητικών προγραμμάτων του λειτουργικού συστήματος αλλά και των εφαρμογών που ο χρήστης επιθυμεί κάθε φορά. Συγκεκριμένα στην περίπτωση των εφαρμογών, κανονίζει τη μεταφορά του προγράμματος συνολικά ή κατά τμήματα από την περιφερειακή στην κεντρική μνήμη, με την οποία πλέον επικοινωνεί η ΚΜΕ για την ανάκληση, την αποκωδικοποίηση και την εκτέλεση κάθε εντολής (*δες 2.5.5 Επικοινωνία ΚΜΕ και Κύριας Μνήμης – Εκτέλεση Εντολών*)
- ***Δυνατότητα πολυ-εργασίας (multitask):*** Όλα τα σύγχρονα λειτουργικά συστήματα παρέχουν τη δυνατότητα της παράλληλης εκτέλεσης περισσοτέρων της μιας εργασιών, οπότε δίνεται διαδοχικά για απειρο-ελάχιστο χρονικό διάστημα ο έλεγχος της ΚΜΕ στο κάθε ενεργό πρόγραμμα (time-sharing). Είναι τόσο μεγάλη η συχνότητα εναλλαγής που ο χρήστης έχει την εντύπωση της ταυτόχρονης εκτέλεσης.
- ***Δυνατότητα πολυ-χρησίας (multiuser):*** Τα σύγχρονα λειτουργικά συστήματα δίνουν τη δυνατότητα να χρησιμοποιούν πολλοί χρήστες την ίδια ΚΜΕ, είτε ταυτόχρονα είτε με την εναλλαγή τον λογαριασμών τους. Σε αυτήν την περίπτωση, υπάρχουν ειδικοί μηχανισμοί προστασίας των δεδομένων κάθε χρήστη.
- ***Δυνατότητα πολυ-επεξεργασίας (multiprocessing):*** Οι υπερ-υπολογιστές και κάποια μεγάλα συστήματα διαθέτουν περισσότερους του ενός επεξεργαστές (ΚΜΕ) για γρηγορότερη εκτέλεση των προγραμμάτων. Σ' αυτήν την περίπτωση συνοδεύονται και από ειδικά λειτουργικά συστήματα που επιτρέπουν τον συντονισμό της ταυτόχρονης λειτουργίας των επεξεργαστών.

4.2.1.2 Σύγχρονα Λειτουργικά Συστήματα

Όλα τα σύγχρονα λειτουργικά συστήματα χαρακτηρίζονται από τα εξελιγμένα γραφικά περιβάλλοντα εργασίας (*Graphical User Interface – GUI*), τα οποία προσφέρουν στο χρήστη εύκολη πρόσβαση στις διάφορες εφαρμογές αλλά και έναν αποδοτικό τρόπο εργασίας, ελαχιστοποιώντας την πληκτρολόγηση εντολών και μεγιστοποιώντας την χρήση του ποντικιού. Η εξέλιξη αυτών των περιβαλλόντων και η εφεύρεση νέων τρόπων διάδρασης με τον τελικό χρήστη αποτελεί την βασική μέριμνα των κατασκευαστών λογισμικού λειτουργικών συστημάτων και καθορίζει σημαντικά το μερίδιο της αγοράς που κατέχει ο καθένας από αυτούς.



Εικόνα 4.5. *Δομή Λειτουργικού Συστήματος – Σύγχρονοι κατασκευαστές και εκδόσεις*

4.2.2 Γλώσσες Προγραμματισμού

Οι γλώσσες προγραμματισμού αποτελούν το μέσο για την υλοποίηση όλου του λογισμικού, το οποίο εκτελούν οι ηλεκτρονικοί υπολογιστές. Δεδομένου ότι οι ηλεκτρονικοί υπολογιστές αποτελούνται από ηλεκτρονικά κυκλώματα τα οποία κατανοούν μόνο δυαδικά σύμβολα, τα προγράμματα και η δημιουργία αυτών των προγραμμάτων γίνεται με τη χρήση της γλώσσας μηχανής (*machine language*), η οποία δεν είναι τίποτα άλλο από δυαδικές συμβολοσειρές. Η συγγραφή εντολών και προγραμμάτων σε δυαδική μορφή, αφενός ήταν μια πολύ επίπονη διαδικασία και αφετέρου οδηγούσε σε πολλά λάθη τους αρχικούς προγραμματιστές. Με την εξέλιξη όμως των ηλεκτρονικών υπολογιστών (βλέπε *2.2. Γενιές Ηλεκτρονικών Υπολογιστών*) δημιουργήθηκαν γλώσσες προγραμματισμού οι οποίες πλησίαζαν όλο και πιο κοντά στον ανθρώπινο τρόπο έκφρασης. Αρχικά λοιπόν δημιουργήθηκαν οι συμβολικές γλώσσες (*assembly languages*), οι οποίες μαζί με τις γλώσσες μηχανής αποτελούσαν τις γλώσσες χαμηλού επιπέδου (*low level languages*). Στην συνέχεια δημιουργήθηκαν οι γλώσσες υψηλού επιπέδου (*high level languages*), όπως Fortran, Algol, Basic, Pascal, C, C++, Java κ.λπ., των οποίων τα βασικά χαρακτηριστικά ήταν η συγγραφή και η δημιουργία των εφαρμογών με λεκτικά σύνολα και εντολές κοντά στην φυσική γλώσσα του ανθρώπου, αλλά και το μεγάλο πλήθος έτοιμων ρουτινών (βιβλιοθήκες κώδικα) για επαναχρησιμοποίηση.

Γλώσσα προγραμματισμού Basic

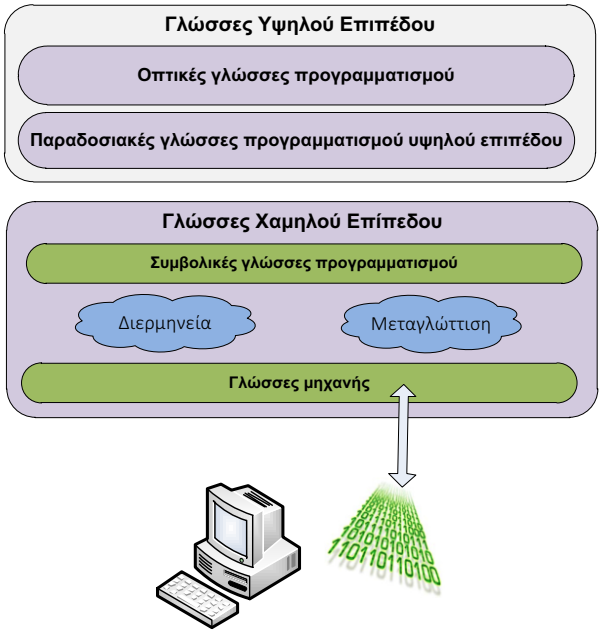
```
10 PRINT "Hello, world!"
20 END
```

Γλώσσα προγραμματισμού Pascal

```
program Hello;
begin
  writeln('Hello, world!');
end.
```

Εικόνα 4.6. *Παραδείγματα γλωσσών προγραμματισμού – Εμφάνιση της φράσης «Hello, world!»*

Τελευταία εξέλιξη στον χώρο των γλωσσών προγραμματισμού είναι η δημιουργία εφαρμογών με τη χρήση γραφικών περιβαλλόντων εργασίας και την ελαχιστοποίηση της ανάγκης για πληκτρολόγηση κώδικά (*source code*). Οι γλώσσες αυτές ονομάζονται οπτικές γλώσσες προγραμματισμού (*visual programming languages*) και έδωσαν τη δυνατότητα ακόμα και σε αρχάριους προγραμματιστές να δημιουργούν εφαρμογές με γραφικό περιβάλλον εργασίας προς τον τελικό χρήστη σε πολύ γρήγορο χρόνο.



Εικόνα 4.7. *Γλώσσες προγραμματισμού*

Για την καλύτερη κατανόηση των γλωσσών προγραμματισμού θα χρησιμοποιηθεί η γλώσσα Basic με σκοπό την παρουσίαση των διαφόρων τύπων.

Α. Εντολή εκχώρησης

Πίνακας 4.4. *Παραδείγματα εντολών εκχώρησης*

Παράδειγμα 1	**Παράδειγμα 2**	**Παράδειγμα 3**	**Παράδειγμα 4**
X = 10	X = A	X = A + B	X = 2 / (A+B) – C * SIN(D) + 3*(B+C)
Με την εντολή αυτή η τιμή αποθηκεύεται στη μεταβλητή.	Με την εντολή αυτή το περιεχόμενο της μεταβλητής (θέσης μνήμης) Α αποθηκεύεται και στη μεταβλητή Χ. Πρέπει να δοθεί προσοχή στο γεγονός ότι δεν πρόκειται για μαθηματική ισότητα η οποία μας πληροφορεί ότι οι δύο μεταβλητές Χ και Α είναι ίσες αλλά για μια διαδικασία, όπου διαβάζεται το περιεχόμενο της δεξιά μεταβλητής και εγγράφεται σαν περιεχόμενο και στην αριστερά μεταβλητή.	Το δεξιό μέλος μπορεί εκτός από μεταβλητή να είναι και μια μαθηματική έκφραση διαφόρων μεταβλητών, οπότε αρχικά γίνεται ο υπολογισμός της έκφρασης και κατόπιν αποθηκεύεται το αποτέλεσμα στην αριστερά μεταβλητή. Αναλυτική περιγραφή των επιμέρους στοιχειωδών διαδικασιών που λαμβάνουν χώρα κατά την εκτέλεση της παραπάνω εντολής δίνεται στο Κεφάλαιο2, παράγραφος 2.5.5.1	Το δεξιό μέλος μπορεί να είναι μια οποιαδήποτε μαθηματική έκφραση που να περιέχει προσθέσεις, αφαιρέσεις, πολλαπλασιασμούς διαιρέσεις, παρενθέσεις για αλλαγή της σειράς των πράξεων, συναρτήσεις, όπως ημίτονο, συνημίτονο κ.λπ.

Β. Εντολές εισόδου/εξόδου

Πίνακας 4.5. *Παραδείγματα εντολών εισόδου / εξόδου*

Παράδειγμα 1	**Παράδειγμα 2**
INPUT X	PRINT X
Με την εντολή αυτή η ΚΜΕ τίθεται σε αναμονή μέχρις ότου δοθεί κάποια τιμή από τη συσκευή εισόδου (το πληκτρολόγιο). Μόλις δοθεί η τιμή, μεταφέρεται και εγγράφεται στη μεταβλητή Χ. Αναλυτική	Με την εντολή αυτή εμφανίζεται στη συσκευή εξόδου (την οθόνη) το περιεχόμενο της μεταβλητής Χ.

περιγραφή των επιμέρους στοιχειωδών διαδικασιών που λαμβάνουν χώρα κατά την εκτέλεση της παραπάνω εντολής δίνεται στο Κεφάλαιο 2, παράγραφος 2.5.5.2.	

Γ. Εντολές ελέγχου

Πίνακας 4.6. *Παραδείγματα εντολών ελέγχου*

<table>
<tr><th colspan="2">Παράδειγμα 1</th><th>Παράδειγμα 2</th></tr>
<tr><td>IF (LE) THEN
Instructions-1
ELSE
Instructions-2
END IF</td><td>X = 10
Y = 20
IF (X>Y) THEN
Z = 30
ELSE
Z = 40
END IF *</td><td>INPUT A,B
IF A=0 THEN
IF B=0 THEN
PRINT (“ΑΟΡΙΣΤΗ ΕΞΙΣΩΣΗ”)
ELSE
PRINT (“ΑΔΥΝΑΤΗ ΕΞΙΣΩΣΗ”)
END IF
ELSE
X=B/A
PRINT(“Η ΛΥΣΗ ΕΙΝΑΙ =”, X)
END IF</td></tr>
<tr><td colspan="2">Με την εντολή αυτή γίνεται αρχικά έλεγχος της αλήθειας μιας λογικής έκφρασης (logical expression - LE) και αν είναι αληθής εκτελούνται οι εντολές που είναι γραμμένες μετά το THEN. Αν είναι ψευδής τότε εκτελούνται οι εντολές που είναι γραμμένες μετά το ELSE, εκτός αν δεν υπάρχει το κομμάτι ELSE οπότε εκτελείται η επόμενη εντολή μετά το END IF.

* Μετά την εκτέλεση των παραπάνω εντολών το περιεχόμενο της μεταβλητής Z είναι 40.</td><td>Το παραπάνω τμήμα προγράμματος επιλύει μια πρωτοβάθμια εξίσωση ΑΧ=Β.</td></tr>
</table>

Δ. Εντολές επανάληψης

Πίνακας 4.7. *Παραδείγματα εντολών επανάληψης*

<table>
<tr><th colspan="2">Παράδειγμα 1</th><th colspan="2">Παράδειγμα 2</th></tr>
<tr><td>FOR I = 1 TO N
…
…
NEXT I</td><td>INPUT N
S = 0
FOR I = 1 TO N
S = S + I
NEXT I
PRINT S *</td><td>Do While / Until (LE)
…
Loop

ή

Do
…
Loop While / Until (LE)</td><td>INPUT N
S=0
I=1
Do WHILE I<=N
S=S+I (συμμετοχή στο
άθροισμα)
I=I+1 (αύξηση μετρητή)
Loop
PRINT S *</td></tr>
<tr><td colspan="2">Με την εντολή αυτή επαναλαμβάνονται οι εντολές μεταξύ της FOR και της NEXT και χρησιμοποιείται όταν ο αριθμός επαναλήψεων των εντολών είναι προκαθορισμένος (N). Όλες οι εντολές που περιέχονται μεταξύ του FOR και του NEXT επαναλαμβάνονται N φορές.

* Μετά την εκτέλεση των παραπάνω εντολών η μεταβλητή S περιέχει το άθροισμα των N πρώτων φυσικών αριθμών 1 + 2 + 3 + ... + N.</td><td colspan="2">Η εντολή Do χρησιμοποιείται στην περίπτωση που δεν είναι εκ των προτέρων καθορισμένος ο αριθμός των επαναλήψεων.
Στην περίπτωση της πρώτης εκδοχής της εντολής, η εκτέλεση ή μη των εντολών μεταξύ της DO και της LOOP εξαρτάται από την λογική έκφραση (LE) και εκτελούνται όσο ισχύει (χρήση WHILE) ή όσο δεν ισχύει (χρήση UNTIL).
Στην περίπτωση της δεύτερης εκδοχής της εντολής, οι εντολές μεταξύ της DO και της LOOP εκτελούνται οπωσδήποτε για μια τουλάχιστον φορά διότι ο έλεγχος της λογικής έκφρασης γίνεται μετά το πέρας των εντολών. Η συνέχιση εκτέλεσης ή μη των εντολών εξαρτάται από το αν ισχύει η λογική έκφραση (χρήση WHILE) ή όχι (χρήση UNTIL).

* Μετά την εκτέλεση των παραπάνω εντολών η μεταβλητή S περιέχει το άθροισμα των N πρώτων φυσικών αριθμών 1 + 2 + 3 + ... + N. Απαιτείται η τοποθέτηση του WHILE στην αρχή ώστε στην περίπτωση που N=0 να μην εκτελεστούν οι εντολές της άθροισης και του μετρητή.</td></tr>
</table>

4.2.2.1 Διερμηνεία και μεταγλώττιση (πλεονεκτήματα – μειονεκτήματα)

Η σειρά των εντολών του προγράμματος της γλώσσας υψηλού επιπέδου αποτελεί το λεγόμενο πηγαίο κώδικα (*source code*). Η σειρά αυτή των εντολών αποθηκεύεται σε ένα αρχείο στο δίσκο. Στη συνέχεια πρέπει οι εντολές αυτές να

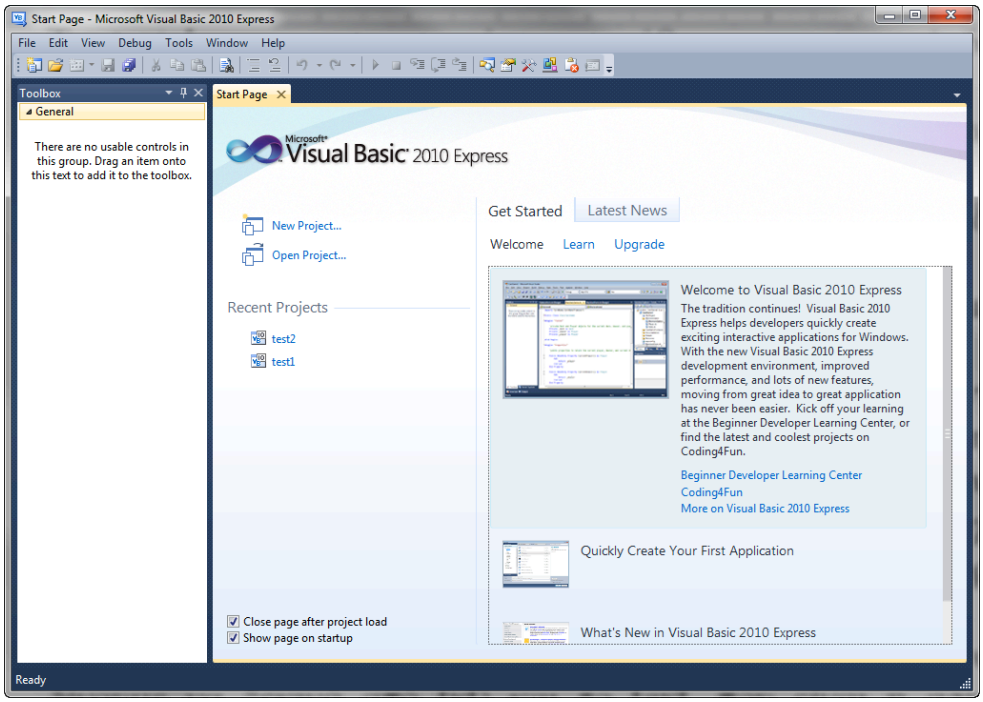
μετατραπούν σε μια μορφή που είναι κατανοητή από την Κεντρική Μονάδα Επεξεργασίας του ηλεκτρονικού υπολογιστή. Η μορφή αυτή είναι ως γνωστόν η δυαδική μορφή και ονομάζεται γλώσσα μηχανής. Για τη μετατροπή αυτή χρησιμοποιούνται δύο είδη λογισμικού, γνωστά ως διερμηνείς (*interpreters*) και μεταγλωττιστές (*compilers*). Η διαφορά τους είναι ότι στην περίπτωση της διερμηνείας η μετατροπή γίνεται εντολή προς εντολή και αμέσως μετά η εντολή εκτελείται, ενώ στην περίπτωση της μεταγλώττισης, ολόκληρος ο πηγαίος κώδικας μετατρέπεται σε γλώσσα μηχανής, στο λεγόμενο εκτελέσιμο κώδικα (*executable code*), ο οποίος αποθηκεύεται σε ένα αρχείο, το λεγόμενο εκτελέσιμο αρχείο, το οποίο εκτελείται συνολικά.

Το χαρακτηριστικό των ονομάτων αυτών των αρχείων είναι η επέκταση *exe* ή *com*. Ο πηγαίος κώδικας μπορεί να περιέχει γραμματικά / συντακτικά λάθη (*grammar* ή *syntax error*), λάθη δηλαδή που αφορούν στη σύνταξη των εντολών. Αυτά τα λάθη εντοπίζονται στο στάδιο της διερμηνείας ή της μεταγλώττισης. Ακόμη όμως και στην περίπτωση που ένα πρόγραμμα είναι καθαρό πλέον από γραμματικά ή συντακτικά λάθη είναι πιθανό να έχει τα λεγόμενα λογικά λάθη (*logical errors*), δηλαδή το είδος και η σειρά των εντολών του να μην οδηγούν στο σωστό αποτέλεσμα. Τα λάθη αυτά εντοπίζονται με την εκτέλεση του προγράμματος με ειδικά δεδομένα δοκιμών (*testing*), οπότε είναι δυνατόν να ελεγχθεί κατά πόσο προκύπτουν τα γνωστά εκ των προτέρων αποτελέσματα. Μετά από μια σειρά δοκιμών, εντοπισμών των λογικών λαθών και αντίστοιχων διορθώσεων το πρόγραμμα είναι απαλλαγμένο τόσο από συντακτικά όσο και από λογικά λάθη οπότε είναι έτοιμο προς χρήση.

Το πλεονέκτημα του λογισμικού διερμηνείας είναι η αμεσότητα διερμηνείας και εκτέλεσης κάθε εντολής ξεχωριστά, με αποτέλεσμα τον ευκολότερο εντοπισμό των λαθών, αλλά το μειονέκτημά του είναι ότι ο χρόνος εκτέλεσης επιβαρύνεται κάθε φορά με τον χρόνο διερμηνείας. Αντίθετα ο μεταγλωττιστής έχει κάνει την μετατροπή μόνο μία φορά, οπότε η εκτέλεση του προγράμματος είναι πιο γρήγορη, εφόσον δεν μεσολαβεί και πάλι η μεταγλώττιση ενώ ταυτόχρονα είναι ανεξάρτητη από την παρουσία του λογισμικού του μεταγλωττιστή. Σε αυτή την περίπτωση η μεγαλύτερη δυσκολία στον εντοπισμό των λαθών υποβοηθείται από ειδικά προγράμματα διόρθωσης γνωστά ως debuggers.

4.2.2.2 Visual Basic

Η Visual Basic είναι μια γλώσσα προγραμματισμού υψηλού επιπέδου, η οποία στηρίζεται στη γλώσσα προγραμματισμού Basic (*Beginner's All Purpose Symbolic Instruction Code*). Στη σημερινή εποχή η τελευταία έκδοση της Visual Basic είναι η Microsoft Visual Basic 2010 Express, η οποία διατίθεται δωρεάν, ως μέρος του Visual Studio 2010 Express, από την ίδια την εταιρεία. Στο περιβάλλον της Visual Basic ο προγραμματιστής έχει τη δυνατότητα με απλά εργαλεία να σχεδιάσει ένα αποτελεσματικό και λειτουργικό γραφικό περιβάλλον διεπαφής με το χρήστη.



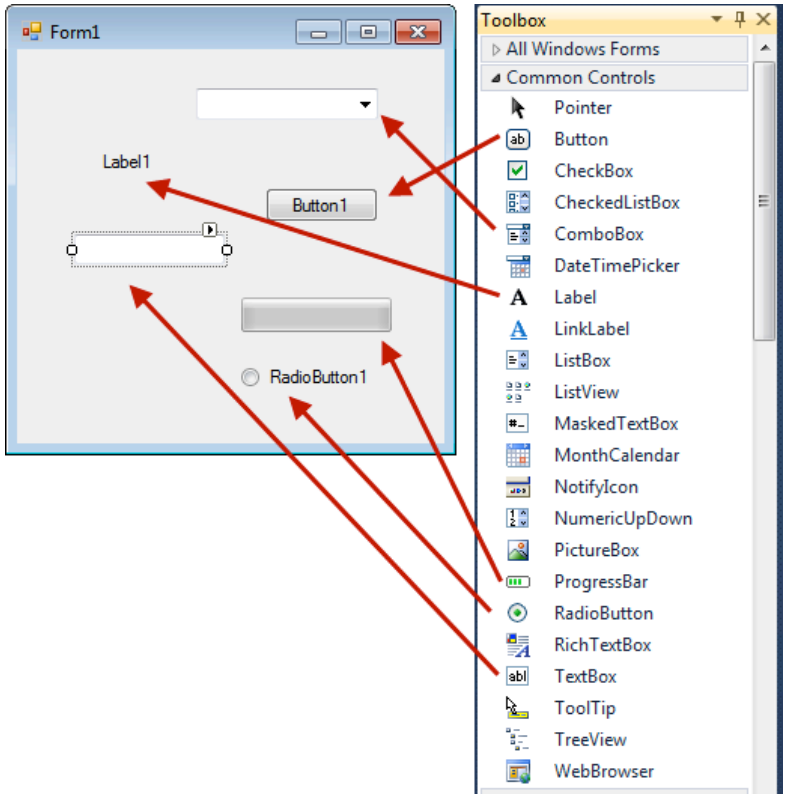
Εικόνα 4.8. *Περιβάλλον εργασίας Microsoft Visual Express*

Η έννοια του προγράμματος αντικαθίσταται από το λεγόμενο project, το οποίο αποτελείται από μια βασική φόρμα (*form*), πάνω στην οποία μπορεί ο προγραμματιστής να μεταφέρει όλα τα απαραίτητα εργαλεία διαλόγου της εφαρμογής με το χρήστη, τα κυριότερα από τα οποία είναι τα απλά πλαίσια παρουσίασης κειμένου (*labels*), τα πλαίσια εισαγωγής κειμένου (*text boxes*), τα πλήκτρα εκτέλεσης διαδικασιών (*command buttons*), τα πλαίσια παρουσίασης μηνυμάτων (*message boxes*), οι λίστες επιλογής (*drop down menu*), τα κουμπιά επιλογών (radio buttons) κ.λπ.

Τα *text boxes* χρησιμοποιούνται συνήθως για εισαγωγή δεδομένων από το χρήστη, ενώ τα labels και σπανιότερα τα message boxes για εξαγωγή των αποτελεσμάτων. Από την άλλη η κεντρική διαδικασία του προγράμματος ενεργοποιείται από το χρήστη μέσω ενός command button.

Το label, το text box και το command button είναι δομές δεδομένων με διάφορα πεδία, κάποια από τα οποία μπορεί να συμπληρώσει ο προγραμματιστής κατά τη φάση σχεδίασης. Τα πιο σημαντικά πεδία για το label είναι το όνομα (*name*), όπου δηλώνουμε ένα χαρακτηριστικό τίτλο για το πλαίσιο με σκοπό να αναφερόμαστε σε αυτό (καλό είναι το όνομα που δίνουμε να ξεκινά από *lbl*) και το caption, που είναι η αρχική ταμπέλα που εμφανίζεται σ' αυτό κατά την έναρξη της φάσης

λειτουργίας. Αντίστοιχα για το textbox είναι το name, όπου δηλώνουμε ένα όνομα για το πλαίσιο με σκοπό να αναφερόμαστε σε αυτό (καλό είναι το όνομα που δίνουμε να ξεκινά από txt) και το text, που είναι η αρχική ταμπέλα που εμφανίζεται σ' αυτό κατά την έναρξη της φάσης λειτουργίας. Τέλος για το command box, το name, όπου δηλώνουμε ένα όνομα για το πλήκτρο με σκοπό να αναφερόμαστε σε αυτό (καλό είναι το όνομα που δίνουμε να ξεκινά από cmd) και το caption, που είναι η ταμπέλα που εμφανίζεται πάνω στο πλήκτρο κατά τη φάση λειτουργίας και δηλώνει τη διαδικασία, η οποία ενεργοποιείται μέσω αυτού.



Εικόνα 4.9. *Φόρμα και εργαλεία διαλόγου*

(Name)	**Label1**
AccessibleDescription	
AccessibleName	
AccessibleRole	Default
AllowDrop	False
Anchor	Top, Left
AutoEllipsis	False
AutoSize	**True**
BackColor	Control
BorderStyle	None
CausesValidation	True
ContextMenuStrip	(none)
Cursor	Default
Dock	None
Enabled	True
FlatStyle	Standard
Font	Microsoft Sans Serif; 8,25
ForeColor	ControlText
GenerateMember	True
Image	(none)
ImageAlign	MiddleCenter
ImageIndex	(none)

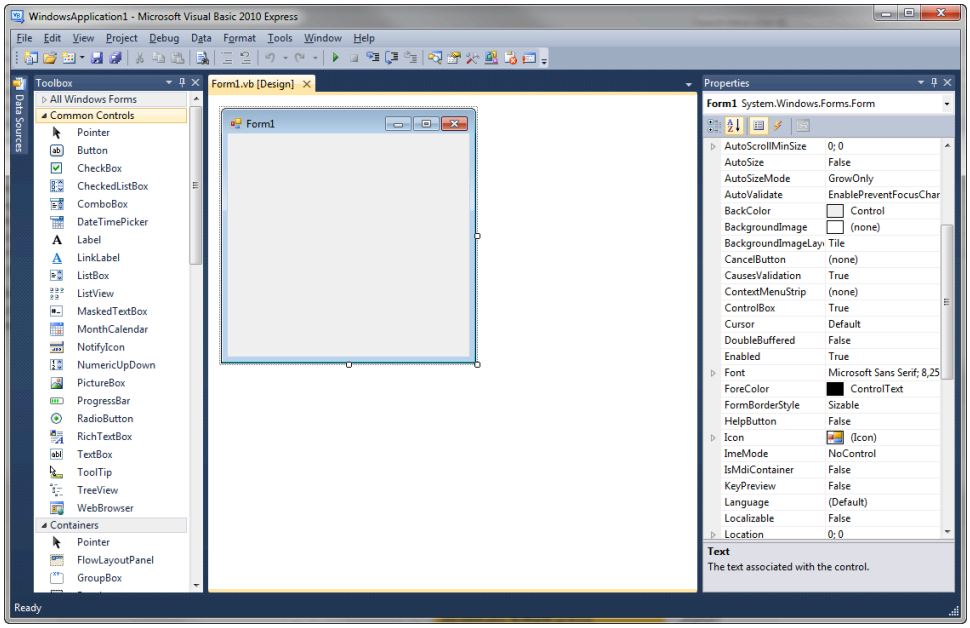
Εικόνα 4.10. *Βασικά πεδία ιδιοτήτων εργαλείου label*

Για την καλύτερη κατανόηση των πιο πάνω εννοιών ακολουθεί η παρουσίαση δυο παραδειγμάτων, βήμα προς βήμα.

Α. Παράδειγμα ενός message box (Project showMessage)

Η εγκατάσταση του περιβάλλοντος Microsoft Visual Basic Express μπορεί να πραγματοποιηθεί σχετικά εύκολα με την μεταφόρτωση του προγράμματος από το δικτυακό τόπο της εταιρείας Microsoft[1]. Από την αρχική οθόνη της εφαρμογής επιλέγεται (βλέπε Εικόνα 4.8) η λειτουργία «New Project…». Ως τύπος project επιλέγεται το «Windows Forms Applications». Μετά την επιλογή αυτή εμφανίζεται η αρχική οθόνη του project και η βασική φόρμα (form – Form1).

[1] http://www.microsoft.com/express/



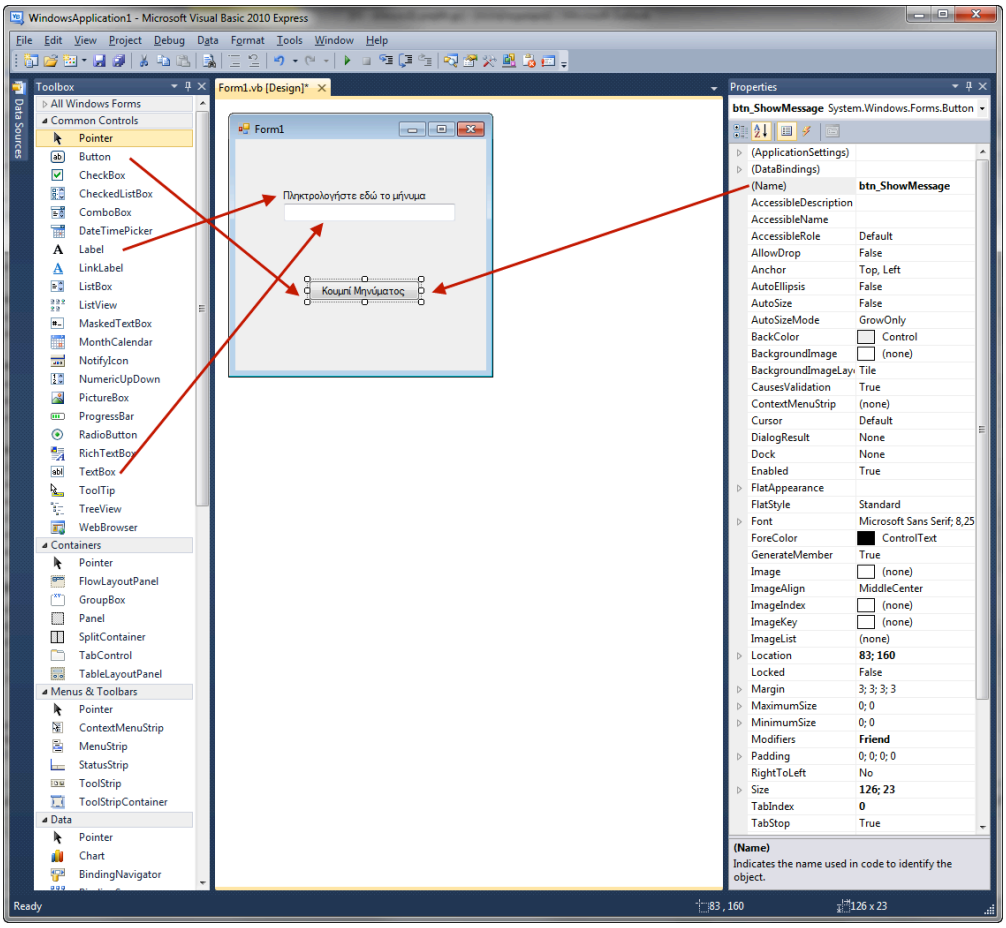
Εικόνα 4.11. *Αρχική οθόνη νέου project*

Σκοπός του project είναι η εμφάνιση ενός μηνύματος, μέσω ενός πλαισίου μηνυμάτων (*message box*). Το μήνυμα θα εμφανίζεται με το πάτημα ενός κουμπιού από τον χρήστη της εφαρμογής, αφού πρώτα έχει εισαχθεί σε ένα πλαίσιο κειμένου (text box). Συμπληρωματικό στοιχείο του text box είναι ένα εργαλείο (*component*) τύπου caption, το οποίο θα πληροφορεί το χρήστη για τη σωστή χρήση του.

Πίνακας 4.8. *Απαραίτητα components για το project showMessage*

Type	Name	Caption / Text
Command Button	btn_ShowMessage	Εμφάνιση Μηνύματος
Text Box	txtbox_message	-
Label	lbl_messageboxlabel	Πληκτρολογήστε εδώ το μήνυμα

Επομένως, προστίθενται από την εργαλειοθήκη τα αντίστοιχα components και ρυθμίζονται οι ιδιότητες τους (name ή caption ή text).



Εικόνα 4.12. *Προσθήκη εργαλείων στη φόρμα*

Η διάδραση με τον τελικό χρήστη πραγματοποιείται μέσα από το command button και το text box. Ειδικότερα για το πρώτο, πρέπει να οριστεί το είδος των ενεργειών που εκτελούνται όταν το κουμπί ενεργοποιηθεί από τον χρήστη. Πριν όμως προχωρήσουμε στην συγγραφή της ρουτίνας που καλείται με το πάτημα του command button, πρέπει να οριστούν οι μεταβλητές που χρησιμοποιούνται και ανταλλάσσονται μεταξύ διαφόρων εργαλείων της φόρμας. Ο ορισμός αυτών των μεταβλητών, όπως και κάθε αρχικής πληροφορίας που αφορά την εφαρμογή καταχωρείται μετά τη δήλωση της φόρμας (*Public Class Form1*). Η εμφάνιση του παραθύρου με τον κώδικα σχετικά με την Form1_Load() επιτυγχάνεται με ένα διπλό κλικ του ποντικιού στον κενό χώρο της φόρμας. Η μεταβλητή που πρέπει να δηλωθεί για τις ανάγκες της εφαρμογής είναι η μεταβλητή τύπου string (κειμένου) με όνομα message_txt.

```
Public Class Form1
    Dim message_txt As String
....
End Class
```

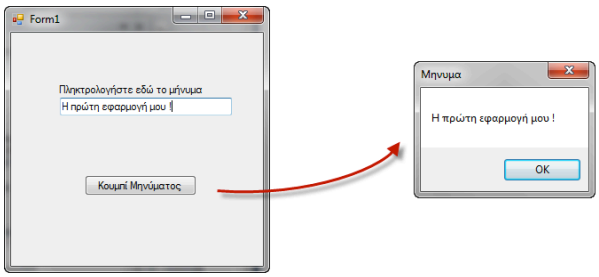
Με τη βοήθεια της μεταβλητής αυτής θα «περάσει» το περιεχόμενο του text box ως όρισμα στη συνάρτηση που θα «κληθεί» όταν ενεργοποιηθεί το command button. Η απόδοση τιμής σε μια μεταβλητή κειμένου (string) ή αριθμού (integer ή single) γίνεται συνήθως μέσω του πεδίου text ενός text box. Πιο συγκεκριμένα στο project του παραδείγματος έχει ως εξής:

```
message_txt = txt_box_message.Text
```

Μέσα στη ρουτίνα ShowMessage_Click(), (από το όνομα του command button, ShowMessage). ενεργοποιείται ο κώδικας που υλοποιεί την όλη διαδικασία.

```
Private Sub btn_ShowMessage_Click(ByVal sender As Sys-
tem.Object, ByVal e As System.EventArgs) Handles
btn_ShowMessage.Click
        message_txt = txt_box_message.Text
        MsgBox(message_txt, vbOKOnly, "Μηνυμα")
    End Sub
```

Η εντολή MsgBox δέχεται ως ορίσματα το κείμενο του text box, κάποιες ρυθμίσεις σχετικά με το ποια κουμπιά ελέγχου θα εμφανίσει, καθώς και των τίτλο του παραθύρου, στο οποίο θα εμφανιστεί το μήνυμα. Η δοκιμή του project και τυχόν αποσφαλμάτωσης (debugging) πραγματοποιείται με την εκτέλεση της εντολής Debug→Start Debugging. Αν όλα είναι σωστά θα μας εμφανιστεί το παράθυρο του project με τη μια και μοναδική φόρμα. Αν πληκτρολογηθεί κείμενο στο text box και μετά πατηθεί το command button, το αποτέλεσμα απεικονίζεται στο σχήμα που ακολουθεί.



Εικόνα 4.13. *Εκτέλεση της εφαρμογής Project showMessage*

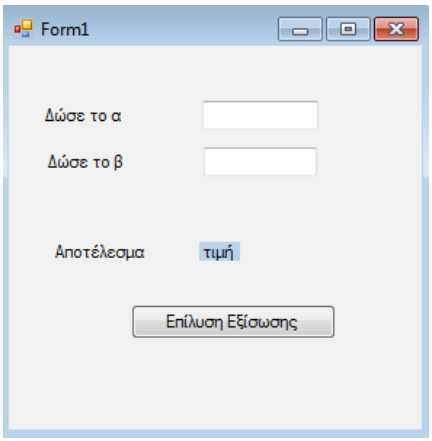
Β. Επίλυση Πρωτοβάθμιας εξίσωσης (αx=β)

Η φόρμα περιλαμβάνει τα εξής εργαλεία :

Πίνακας 4.8. *Απαραίτητα components για το project showMessage*

Type	Name	Caption / Text
Label	lblA	Δώσε το α
Label	lblB	Δώσε το β
Text Box	txtA	
Text Box	txtB	
Label	lblResult	τιμή
Label	lblResult_title	Αποτέλεσμα
Command Button	cmdSolve	Επίλυση Εξίσωσης

Πρώτα πρέπει να δημιουργηθεί το project και να πραγματοποιηθεί η προσθήκη όλων των πιο πάνω εργαλείων στην βασική φόρμα.



Εικόνα 4.14. *Φόρμα επίλυσης πρωτοβάθμιας εξίσωσης*

Οι αναγκαίες μεταβλητές στο συγκεκριμένο project περιγράφονται στον κώδικα που ακολουθεί.

```
Public Class Form1
    Public a As Double
    Public b As Double
    Public x As Double
…
End Class
```

Όλη η λογική θα περιέχεται στη συνάρτηση cmdSolve_Click(), η οποία θα ενεργοποιείται όταν ο χρήστης πατάει το αντίστοιχο κουμπί. Στη συνέχεια δίδονται δυο παραλλαγές κώδικα για την υλοποίηση του συγκεκριμένου project.

1η παραλλαγή

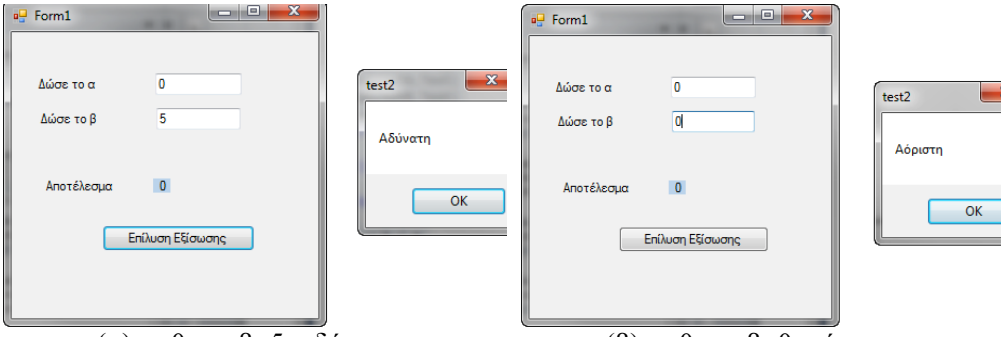
```
Private Sub cmdSolve_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles cmdSolve.Click
        a = Val[2](txtA.Text)
        b = Val(txtB.Text)
        If a = 0 And b = 0 Then
            MsgBox("Αόριστη")
        Else
            If a = 0 And b <> 0 Then
                MsgBox("Αδύνατη")
            Else
                x = b / a
                lblResult.Text = Str[3](x)
            End If
        End If
    End Sub
```

2η παραλλαγή

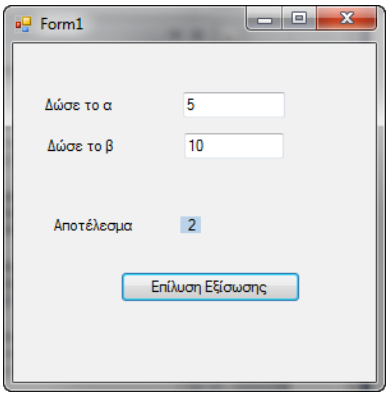
```
Private Sub cmdSolve_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles cmdSolve.Click
        a = Val(txtA.Text)
        b = Val(txtB.Text)
        If a = 0 Then
            If b = 0 Then
                MsgBox("Αόριστη")
            Else
                MsgBox("Αδύνατη")
            End If
        Else
            x = b / a
            lblResult.Text = Str(x)
        End If
    End Sub
```

[2] Συνάρτηση που μετατρέπει το κείμενο σε αριθμό
[3] Συνάρτηση που μετατρέπει τον αριθμό σε κείμενο

Τα αποτελέσματα με διάφορες περιπτώσεις δεδομένων εισόδου, περιγράφονται παρακάτω.



(α) α=0 και β=5, αδύνατη (β) α=0 και β=0, αόριστη

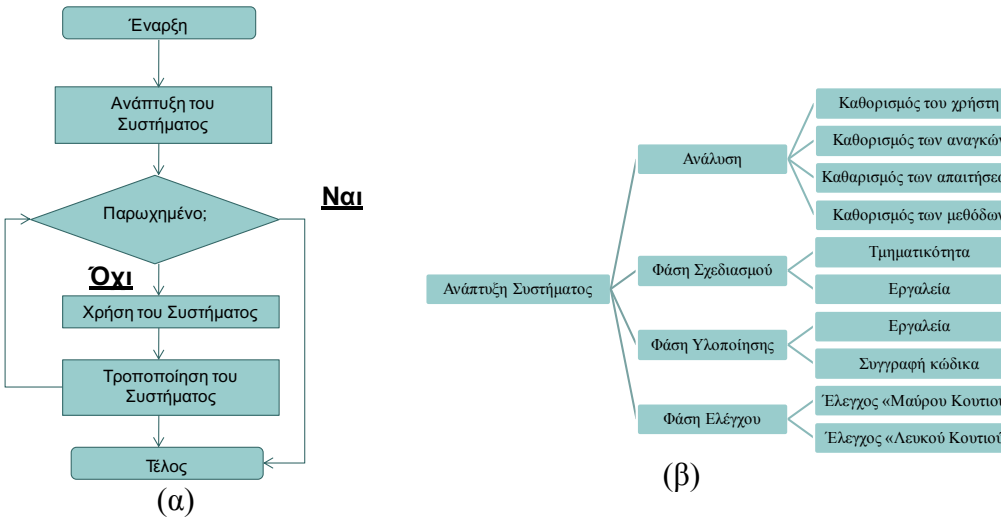


(γ) α=5 και β=10, x=2

Εικόνα 4.15. *Αποτελέσματα project επίλυσης πρωτοβάθμιας εξίσωσης*

4.2.3 Διαδικασία παραγωγής ενός λογισμικού πακέτου

Τα πακέτα λογισμικού που παράγονται, ανεξάρτητα από την κατηγορία στην οποία ανήκουν (π.χ. λειτουργικά συστήματα, εφαρμογές γραφείου, βιβλιοθηκονομικές εφαρμογές κ.λπ.), παρουσιάζουν πολύ μεγαλύτερη πολυπλοκότητα από τις «απλοϊκές» εφαρμογές των προηγούμενων παραδειγμάτων. Για το λόγο αυτό μια σειρά από εξειδικευμένα στελέχη και επιστήμονες, οργανωμένοι σε ομάδες, ασχολούνται με τη δημιουργία λογισμικών πακέτων, ακολουθώντας συγκεκριμένες φάσεις υλοποίησης (βλέπε σχήματα που ακολουθούν).



Εικόνα 4.16. *Αποτελέσματα project επίλυσης πρωτοβάθμιας εξίσωσης*

Περιγραφικά η ροή και τα βήματα που ακολουθούνται για την ανάπτυξη ενός λογισμικού πακέτου έχει ως εξής: ο αναλυτής του προς υλοποίηση συστήματος λογισμικού σε συνεργασία με την ομάδα προγραμματιστών συναντάται με τον πελάτη - χρηματοδότη του συστήματος καθώς και με μια επιλεγμένη ομάδα χρηστών (*user group*), ώστε με βάση τις απαιτήσεις των χρηστών (*user requirements*), να τεθούν κάποιες γενικές λειτουργικές προδιαγραφές (*operational specifications*). Οι προδιαγραφές αυτές θα περιγράφουν με γενικό τρόπο τη λειτουργία του προγράμματος και θα αποτελούν το σημείο αναφοράς τόσο για τον αναλυτή όσο και για τον πελάτη. Σε αυτές τις συμφωνημένες προδιαγραφές θα ανατρέχουν όποτε υπάρχει κάποιο σκοτεινό σημείο ή διαφορά μεταξύ τους (*συμφωνία*). Αυτές πρέπει να έχει υπόψη του ο πελάτης για να ξέρει τι πρέπει να περιμένει από το πρόγραμμα που έχει παραγγείλει, ώστε οι απαιτήσεις του να είναι μέσα στα συμφωνημένα πλαίσια.

Ενώ όμως η αρχική συνοπτική περιγραφή είναι αρκετή για την επικοινωνία του αναλυτή με τον πελάτη, είναι ανεπαρκής για την επικοινωνία του αναλυτή με την ομάδα των προγραμματιστών που πρόκειται να υλοποιήσουν το πρόγραμμα. Έτσι οι αρχικές προδιαγραφές θα γίνουν ο οδηγός για μια λεπτομερή συνολική σχεδίαση (*design*) του συστήματος, όπου θα περιλαμβάνονται και νέες προδιαγραφές που θα έχουν σχέση με την υλοποίηση (*implementation*) του προγράμματος σε μια συγκεκριμένη γλώσσα προγραμματισμού. Οι προδιαγραφές αυτές θα πρέπει να περιγράφουν με όσο το δυνατό μεγαλύτερη ακρίβεια την αρχιτεκτονική του συστήματος, τις επιμέρους μονάδες (*modules*) καθώς και την επικοινωνία μεταξύ τους και θα χρησιμοποιούνται από τους προγραμματιστές σαν οδηγός τόσο για τις υλοποιήσεις των επιμέρους μονάδων όσο και για την σύνδεση και ενσωμάτωσή

τους στο συνολικό σύστημα. Θα χρησιμοποιούνται επίσης και από τον αναλυτή για να ελέγχει κατά πόσο εκτέλεσαν σωστά τις οδηγίες του οι προγραμματιστές.

Στη συνέχεια ο κάθε προγραμματιστής αναλαμβάνει την υλοποίηση της μονάδας ή των μονάδων για τις οποίες έχει ευθύνη, γράφοντας ένα αναλυτικό αλγόριθμο, τον οποίο στη συνέχεια υλοποιεί σε πρόγραμμα, χρησιμοποιώντας μια κατάλληλη γλώσσα προγραμματισμού.

Μετά την αρχική υλοποίηση της μονάδας λογισμικού, ο προγραμματιστής προβαίνει σε μια σειρά από δοκιμές (*testing*) με σκοπό τον εντοπισμό και τη διόρθωση των λαθών (*debugging*). Κατόπιν μπορεί να ακολουθήσει κάποια επίδειξη (*demo*) στον αναλυτή, τον πελάτη και τους χρήστες, μέσα από την οποία μπορεί να προκύψουν παράπονα σε σχέση με τη λειτουργικότητα της μονάδας. Οι οποιεσδήποτε διαφωνίες θα λυθούν με βάση τις τεθείσες προδιαγραφές. Αν είναι ευθύνη του προγραμματιστή, είναι υποχρεωμένος να κάνει τις αλλαγές που χρειάζονται. Αν πρόκειται για διαφορετική προσέγγιση των προδιαγραφών, με την έννοια ότι άλλα εννοούσε ο πελάτης και αλλιώς το είχε εκφράσει, γίνεται μια επανεξέταση του όλου ζητήματος και γίνονται αποδεκτές κάποιες μετατροπές, εφόσον δεν αλλάζουν σημαντικά το κόστος του έργου. Μετά από τις αλλαγές, πρέπει να γίνει και πάλι επίδειξη της μονάδας και κάποιες ίσως τελικές ρυθμίσεις (*fine tuning*) με τις οποίες ολοκληρώνεται η μονάδα.

Η ανάλυση των απαιτήσεων του χρήστη, η σχεδίαση του συστήματος και ο καθορισμός τόσο των γενικών λειτουργικών προδιαγραφών όσο και των λεπτομερέστερων προδιαγραφών υλοποίησης του συστήματος (προγράμματος) συνήθως δεν παραμένει σταθερή και αμετάβλητη κατά την πορεία του έργου. Αντίθετα, ιδίως στα μεγάλα συστήματα-προγράμματα, οι προδιαγραφές μπορεί να αλλάξουν γιατί κάποιες λεπτομέρειες κρίνονται ιδιαίτερα χρονοβόρες και επομένως ακριβές για την υλοποίησή τους, οπότε τροποποιούνται ή αποσύρονται. Επίσης κάποια σημεία του συστήματος γίνονται περισσότερο κατανοητά στην πορεία του έργου από το χρήστη, ο οποίος στην αρχή τουλάχιστον δυσκολεύεται να παρακολουθήσει την αυστηρή λογική και ορολογία του αναλυτή, με αποτέλεσμα να απαιτεί στα σημεία αυτά κάποιες αλλαγές. Σε μια τέτοια περίπτωση, εξετάζει, όπως αναφέρθηκε, ο αναλυτής σε συνεργασία με τους προγραμματιστές κατά πόσο και σε ποιο βαθμό είναι εφικτές οι επιθυμητές αλλαγές και ενημερώνουν τον πελάτη, ώστε να καταλήξουν και πάλι σε κάποιο νέο συμφωνημένο σύστημα προδιαγραφών.

Μετά την υλοποίηση όλων των μονάδων που συνιστούν το σύστημα ακολουθεί η διαδικασία συνένωσης (*merging*) σε ένα συνολικό σύστημα. Στο σημείο αυτό έχει ιδιαίτερη σημασία η ποιότητα της διεπαφής (*interface*) με τον χρήστη, από όπου καθορίζεται η απλότητα της εργονομίας και η ευχρηστία του συστήματος. Μετά την υλοποίηση του συνολικού συστήματος πρέπει να γίνουν και πάλι δοκιμές υπό την εποπτεία του αναλυτή, με σκοπό τον εντοπισμό και τη διόρθωση λαθών που

σχετίζονται με τη σύνδεση των επιμέρους μονάδων καθώς και τη διεπαφή με τον χρήστη. Τέλος είναι απαραίτητη μια τελική επίδειξη του συνολικού συστήματος εκ μέρους του αναλυτή στον πελάτη και τους χρήστες. Κάποιες μικρές αλλαγές σε σχέση με την εργονομία του συστήματος είναι πιθανόν εφικτές σε συνεννόηση με τους προγραμματιστές, μετά από τις οποίες εισέρχεται το προϊόν σε μια σειρά αυστηρών τελικών δοκιμών.

Τέλος γίνεται παράδοση και εγκατάσταση (*installation*) του πακέτου και ίσως μια στοιχειώδης ή λεπτομερειακή εκπαίδευση των χρηστών.

Είναι ευνόητο ότι στην περίπτωση υλοποίησης απλών προγραμμάτων, αναλυτής και προγραμματιστής (ή ακόμα και χρήστης) είναι ένα και το αυτό πρόσωπο.

Τα βήματα που ακολουθούνται από τις διάφορες υπο-ομάδες της ανάπτυξης του λογισμικού είναι τα παρακάτω:

- **Βήμα 1** - Πελάτης – Εταιρεία Λογισμικού / Προγραμματιστές
- **Βήμα 2** - Ομάδα χρηστών (user group) → user requirements → operational specifications
- **Βήμα 3** - Συμφωνία → Μελέτη, Σχέδιο Αποδοχής κτλ.
- **Βήμα 4** - Σχεδιασμός (design) → προδιαγραφές → υλοποίηση (implementation) ανά μονάδα - module
- **Βήμα 5** - Δοκιμές (testing), αποσφαλμάτωση (debugging) – fine tuning τελικές ρυθμίσεις
- **Βήμα 6** - Έλεγχος Ποιότητας – Σχόλια / παρατηρήσεις πελάτη
- **Βήμα 7** - Συνένωση (merging) – χτίσιμο (building) – παραγωγή έκδοσης
- **Βήμα 8** - Παραγωγή διεπαφής χρήστη (GUI – Graphical User Interface)
- **Βήμα 9** - Εγκατάσταση – Πιλοτική Λειτουργία – Τεστ Αποδοχής
- **Βήμα 10** - Παραγωγική Λειτουργία – Συντήρηση - Αναβάθμιση

4.2.4 Ομάδα ανάπτυξης λογισμικών πακέτων

Οι ειδικότητες που εμπλέκονται στην ανάπτυξη λογισμικών πακέτων, καθώς και η αναλυτική περιγραφή των καθηκόντων τους παρουσιάζονται παρακάτω.

Υπεύθυνος Μηχανογράφησης (*information technology manager*)

- Έχει τη γενική εποπτεία της πορείας των έργων σε συνεργασία με τους αναλυτές που έχουν αναλάβει τα συγκεκριμένα έργα.

- Ενημερώνεται από τον Μηχανικό Συστημάτων για τις εξελίξεις στον χώρο του υλικού και του λογισμικού, εξετάζει τις προτάσεις τους για τυχόν αγορές ή αναβαθμίσεις και προβαίνει στις σχετικές ενέργειες.
- Ενημερώνεται από τους αναλυτές των επιμέρους έργων για τις ανάγκες σε προσωπικό, εξοπλισμό και λογισμικό και προβαίνει στις σχετικές ενέργειες.

Αναλυτής συστημάτων (*system analyst*)

- Προΐσταται του έργου που του έχει ανατεθεί και είναι ο άμεσα υπεύθυνος γι' αυτό, τόσο από τεχνικής όσο και από διοικητικής πλευράς.
- Καταστρώνει το χρονοδιάγραμμα του έργου, φροντίζει για τη στελέχωση του με το απαιτούμενο προσωπικό, και παρακολουθεί την εξέλιξη του με σκοπό την ομαλή εφαρμογή του χρονοδιαγράμματος, την έγκαιρη παράδοση των παραδοτέων, την τήρηση των προδιαγραφών που έχουν τεθεί και την πλήρη τεκμηρίωση των εργασιών που έχουν εκτελεστεί.
- Συνεργάζεται με τον πελάτη και τους χρήστες για τον καθορισμό, με βάση τις απαιτήσεις τους, των λειτουργικών προδιαγραφών και των δεδομένων ελέγχου, καθώς επίσης για την τελική εγκατάσταση του συστήματος και την εκπαίδευση των χρηστών.
- Συνεργάζεται με την ομάδα των προγραμματιστών για τη λεπτομερή σχεδίαση και τον καθορισμό των προδιαγραφών που αφορούν την αρχιτεκτονική του λογισμικού, τα επιμέρους προγράμματα καθώς και την επικοινωνία τους.

Μηχανικός Συστημάτων (*system engineer*)

- Παρακολουθεί τις εξελίξεις στο χώρο του υλικού και του λογισμικού και κάνει προτάσεις για τυχόν αγορές νέων συστημάτων ή αναβάθμιση των υπαρχόντων.
- Εγκαθιστά τις νέες εκδόσεις λογισμικού στους Η/Υ (Λειτουργικά Συστήματα, επεξεργαστές κειμένου, μεταγλωττιστές, Συστήματα Διαχείρισης Βάσεων Δεδομένων DBMS's κλπ)
- Παρέχει τεχνική βοήθεια στο προσωπικό με σκοπό το σωστό χειρισμό του υλικού και του λογισμικού.
- Είναι ο υπεύθυνος για την επίλυση των προβλημάτων που παρουσιάζονται στο υλικό ή το λογισμικό και στην περίπτωση που το πρόβλημα δεν είναι άμεσα αντιμετωπίσιμο έχει την ευθύνη να καλέσει το σχετικό τεχνικό αντιπρόσωπο.

Προγραμματιστής (*programmer*)

- Σχεδιάζει τον αλγόριθμο και κωδικοποιεί τα βήματά του με σκοπό την υλοποίηση της μονάδας λογισμικού που έχει αναλάβει.
- Εισηγείται στον αρμόδιο αναλυτή τροποποιήσεις, βελτιώσεις ή νέα προσέγγιση στο πρόβλημα που του ανατίθεται.
- Προβαίνει σε ελέγχους του προγράμματος, ώστε αυτό να συμπεριφέρεται σύμφωνα με τις προδιαγραφές.
- Φέρει σε πέρας τις εργασίες που του ανατίθενται μέσα στα προκαθορισμένα χρονικά όρια και αναφέρει εγκαίρως τυχόν παρεκκλίσεις από αυτά.

Χρήστης (*user*)

- Εκτελεί διάφορα προγράμματα βάσει των οδηγιών καθώς και της εκπαίδευσης που έχει λάβει.
- Μεριμνά για την ύπαρξη στο χώρο του Η/Υ των αναλώσιμων (δισκετών, CD κλπ) που απαιτούνται για τις εργασίες.
- Λαμβάνει αντίγραφα ασφαλείας (*back-up*) των αρχείων σε προκαθορισμένα τακτά χρονικά διαστήματα.
- Καταχωρεί σε ημερολόγιο (*logging*) το είδος και το χρόνο εκτελέσεως κάθε εργασίας, καθώς και τις τυχόν βλάβες που παρουσιάζονται.