ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΔΥΤΙΚΗΣ ΕΛΛΑΔΟΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ Τ.Ε.

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ
ΑΡΙΘΜΟΣ ΠΤΥΧΙΑΚΗΣ 1436

Βασικές γνώσεις και εκπαίδευση στο προγραμματισμό με C.

ΕΙΣΗΓΗΤΗΣ: ΚΑΡΕΛΗΣ ΔΗΜΗΤΡΙΟΣ

ΕΞΕΤΑΣΤΕΣ:Η. ΣΤΑΘΑΤΟΣ

Π. ΒΛΑΧΟΠΟΥΛΟΣ

Φοιτητές : ΣΕΡΕΤΗΣ ΔΗΜΗΤΡΙΟΣ

ΑΡΓΥΡΟΠΟΥΛΟΣ ΣΩΚΡΑΤΗΣ

ΠΑΤΡΑ 2014

Περιεχόμενα

Ημερομηνία 19/11/2013

ΕΙΣΑΓΩΓΗ

Στη παρούσα εργασία θα γίνει αναφορά στην γλώσσα προγραμματισμού C. Η γλώσσα C είναι μια διαδικαστική γλώσσα προγραμματισμού. Η C έχει δυνατότητες δομημένου προγραμματισμού και επιτρέπει τη χρήση αναδρομής (αλλά όχι και εμφωλευμένων συναρτήσεων) . Ο σχεδιασμός της περιλαμβάνει δομές που μεταφράζονται αποδοτικά σε τυπικές εντολές μηχανής (machine instructions) και εξ αιτίας αυτού χρησιμοποιείται συχνά σε εφαρμογές που παλιότερα γράφονταν σε συμβολική γλώσσα (assembly language). Στη C δεν επιβάλλεται κάποια συγκεκριμένη μορφή στον πηγαίο κώδικα.

Ακόμα, στη C όλος ο εκτελέσιμος κώδικας περιέχεται σε υπορουτίνες οι οποίες ονομάζονται «συναρτήσεις» (όχι με την αυστηρή έννοια του συναρτησιακού προγραμματισμού). Οι παράμετροι περνιούνται στις συναρτήσεις πάντα με τιμή (pass-by-value). Το πέρασμα με αναφορά (pass-by-reference) γίνεται έμμεσα στην ουσία, περνώντας, ως παραμέτρους των συναρτήσεων, δείκτες στις μεταβλητές των οποίων θέλουμε να αλλάζουμε τις τιμές μέσα από τις συναρτήσεις. Κάποιες από τις συναρτήσεις που έχει το σύστημα, είναι οι printf(), scanf(), strlen() και άλλες, αλλά υπάρχει και η δυνατότητα να δημιουργήσουμε και δικές μας συναρτήσεις. Οι πιο πολλές είχαν το όνομα main(). Τα προγράμματα της C πάντα αρχίζουν με την εκτέλεση των εντολών της συνάρτησης main() και μετά η main() καλεί άλλες συναρτήσεις.

Μερικά από τα χαρακτηριστικά της C είναι :

- Έχει ένα πολύ μικρό σταθερό πλήθος λέξεων-κλειδιών (keywords), το οποίο περιλαμβάνει ένα πλήρες σύνολο δομών/εντολών ελέγχου ροής: for, if/else, while, switch, και do/while, goto.
- Υπάρχει μόνο ένας χώρος ονομάτων (namespace) και τα ονόματα (μεταβλητών, συναρτήσεων, κ.τ.λ.) που ορίζονται από το χρήστη δεν διακρίνονται με κάποιο τρόπο από τις λέξεις-κλειδιά της γλώσσας.
- Υπάρχει ένα μεγάλο πλήθος αριθμητικών και λογικών τελεστών, όπως οι: +, +=, ++, -, -=, --, *, *=, /, /=, ==, &, &&, |, ||, ~, κ.ά.

Μπορούν να οριστούν πίνακες, αν και δεν υπάρχει ειδική λέξη-κλειδί για τον ορισμό τους.

ΣΤΟΧΟΣ

Ο στόχος της παρούσας εργασίας είναι ο φοιτητής να κατανοήσει τις βασικές έννοιες της γλώσσας προγραμματισμού C αλλά και να τη διαχωρίσει από άλλες γλώσσες προγραμματισμού, μέσα από τη θεωρία και από απλά παραδείγματα κώδικα.

ΕΥΧΑΡΙΣΤΙΕΣ

Καταρχήν, θα θέλαμε να ευχαριστήσουμε τον επιβλέποντα καθηγητή της διπλωματικής μας εργασίας κ. Καρέλη Δημήτριο, για την πολύτιμη καθοδήγησή και την υποστήριξη του, σε κάθε φάση της εκπόνησής της αλλά και για την εμπιστοσύνη που μας έδειξε.

Τέλος, θα θέλαμε να εκφράσουμε την ευγνωμοσύνη μας στις οικογένειες μας για την στήριξη όλων αυτών των χρόνων, για την ολοκλήρωση των σπουδών μας.

1. ΒΑΣΙΚΑ ΣΤΟΙΧΕΙΑ ΠΡΟΓΡΑΜΜΑΤΟΣ

Αντικείμενο του παρόντος συγγράμματος είναι η εισαγωγή του αναγνώστη στη λογική του προγραμματισμού Η/Υ. Πιο συγκεκριμένα θα πραγματοποιηθεί ανάλυση σχετικά με τον δομημένο/ διαδικαστικό προγραμματισμό.

Στην πληροφορική δομημένος προγραμματισμός (structured programming) ή διαδικαστικός προγραμματισμός (procedural programming) είναι μία προσέγγιση στον προγραμματισμό[1], η οποία βασίζεται στην έννοια της κλήσης διαδικασίας. Η διαδικασία, γνωστή επίσης και ως ρουτίνα, υπορουτίνα, μέθοδος ή συνάρτηση (δεν σχετίζεται άμεσα με τη μαθηματική έννοια της συνάρτησης), είναι απλά ένα αυτοτελές σύνολο εντολών προς εκτέλεση. Ο δομημένος προγραμματισμός βασίζεται στην αρχή του διαίρει και βασίλευε, καθώς διασπά το βασικό πρόβλημα σε μικρότερα υποπροβλήματα (γνωστά επίσης και ως εργασίες). Κάθε εργασία με πολύπλοκη περιγραφή διαιρείται σε μικρότερες, έως ότου οι εργασίες να είναι αρκετά μικρές, περιεκτικές και εύκολες προς κατανόηση.
Στόχος είναι η κατανόηση των αρχών του προγραμματισμού και η εμπέδωση της φιλοσοφίας του, έτσι ώστε ο αναγνώστης να μπορεί χωρίς δυσχέρειες να προχωρήσει σε άλλες μορφές προγραμματισμού, όπως ο αντικειμενοστραφής προγραμματισμός (object–oriented programming).

Στην προσπάθεια αυτή θα χρησιμοποιηθεί ως πλατφόρμα μία γλώσσα προγραμματισμού υψηλού επιπέδου, η γλώσσα C. Ο όρος γλώσσα υψηλού επιπέδου υποδηλώνει ότι δεν είναι κατασκευασμένη για να λειτουργεί σε συγκεκριμένη αρχιτεκτονική υπολογιστή αλλά δύναται να λειτουργήσει σε πληθώρα αρχιτεκτονικών, γεγονός που σημαίνει ότι

- είναι πολύ εύκολο να εκτελεσθεί σε οποιοδήποτε υπολογιστή με μικρές αλλαγές και όχι με επανασχεδιασμό του.

- Κάθε γλώσσα υψηλού επιπέδου χρειάζεται έναν μεταγλωττιστή (compiler) ο οποίος μεταφράζει τις γραμμές ενός προγράμματος σε γλώσσα μηχανής ώστε να γίνεται κατανοητό το πρόγραμμα από τον υπολογιστή και να εκτελείται.

Ως γλώσσα υψηλού επιπέδου παρουσιάζει τα εξής πλεονεκτήματα:

- Αναγνωσιμότητα , Τα προγράμματα διαβάζονται εύκολα.
- Συντήρηση , Τα προγράμματα είναι εύκολο να συντηρηθούν.
- Μεταφερσιμότητα , Τα προγράμματα μεταφέρονται εύκολα σε διαφορετικά λειτουργικά συστήματα.

Η εμφάνιση δομής στις γλώσσες προγραμματισμού υψηλού επιπέδου ξεκίνησε από τον τρόπο υπολογισμού των εκφράσεων και τον τρόπο αναπαράστασης δεδομένων.
Αναπτύχθηκαν τέσσερεις βασικές δομές εντολών μέσω των οποίων μπορεί να αναπτυχθεί οποιοδήποτε πρόγραμμα οι οποίες είναι οι εξής :

1. ομάδα εντολών, όπου μια ή περισσότερες εντολές σχηματίζουν μια ομάδα όπου η κάθε εντολή εκτελείται η μια μετά την άλλη,
2. εντολές απόφασης, με τις οποίες δύναται να αλλάξει η σειρά εκτέλεσης των εντολών ενός προγράμματος (ροή προγράμματος) ανάλογα με την τιμή που θα προκύψει από την αποτίμηση μιας έκφρασης,
3. εντολές επανάληψης, οι οποίες δίνουν τη δυνατότητα για επαναληπτική εκτέλεση μιας εντολής ή μιας ομάδας εντολών είτε για συγκεκριμένο αριθμό επαναλήψεων, είτε όσο ικανοποιείται μια λογική συνθήκη
4. συναρτήσεις και διαδικασίες, όπου μια ομάδα εντολών μπορεί να αντιμετωπισθεί ως μια εντολή, δίνοντας τη δυνατότητα για μια ιεραρχική δομή στα προγράμματα.

1.1 Χαρακτηριστικά της γλώσσας C.

Στη C δεν επιβάλλεται κάποια συγκεκριμένη μορφή στον πηγαίο κώδικα (όπως, για παράδειγμα, συνέβαινε στις αρχικές εκδόσεις της Fortran). Ο προγραμματιστής, χωρίς να αγνοεί φυσικά το συντακτικό της γλώσσας, είναι ελεύθερος να δώσει όποια μορφή θέλει στον κώδικα που γράφει (free-format source). Το ελληνικό ερωτηματικό (;) χρησιμοποιείται ως τερματιστής εντολών (και όχι ως διαχωριστής, όπως στην Pascal, παραδείγματος χάριν) και τα άγκιστρα ({}) χρησιμοποιούνται για την ομαδοποίηση εντολών (όπως τα begin/end στην Pascal).

Ακόμα, στη C όλος ο εκτελέσιμος κώδικας περιέχεται σε υπορουτίνες οι οποίες ονομάζονται «συναρτήσεις» (όχι με την αυστηρή έννοια του συναρτησιακού προγραμματισμού). Οι παράμετροι περνιούνται στις συναρτήσεις πάντα με τιμή (pass-by-value). Το πέρασμα με αναφορά (pass-by-reference) γίνεται έμμεσα στην ουσία, περνώντας, ως παραμέτρους των συναρτήσεων, δείκτες στις μεταβλητές των οποίων θέλουμε να αλλάζουμε τις τιμές μέσα από τις συναρτήσεις.

Η C έχει ακόμα τα εξής χαρακτηριστικά:

· Έχει ένα πολύ μικρό σταθερό πλήθος λέξεων-κλειδιών (keywords), το οποίο περιλαμβάνει ένα πλήρες σύνολο δομών/εντολών ελέγχου ροής: for, if/else, while, switch, και do/while, goto.
· Υπάρχει μόνο ένας χώρος ονομάτων (namespace) και τα ονόματα (μεταβλητών, συναρτήσεων, κ.τ.λ.) που ορίζονται από το χρήστη δεν διακρίνονται με κάποιο τρόπο από τις λέξεις-κλειδιά της γλώσσας.
· Υπάρχει ένα μεγάλο πλήθος αριθμητικών και λογικών τελεστών, όπως οι: +, +=, ++, -, -=, --, *, *=, /, /=, ==, &, &&, |, ||, ~, κ.ά.
· Σε μία εντολή μπορεί να γίνουν παραπάνω από μια εκχωρήσεις τιμών.
· Η τιμή που επιστρέφει μια συνάρτηση, μπορεί να αγνοηθεί εάν δεν χρειάζεται.
· Ο ορισμός των τύπων των μεταβλητών είναι στατικός και απαραίτητος, αλλά γίνονται έμμεσες μετατροπές από τη γλώσσα. Για παράδειγμα, παραστάσεις με τύπο χαρακτήρα μπορούν να χρησιμοποιηθούν σε σημεία που απαιτείται ακέραιος.

· Η σύνταξη των δηλώσεων ονομάτων προσομοιάζει την χρήση αυτών μέσα στον εκτελέσιμο κώδικα. Η C δεν έχει ειδική λέξη-κλειδί για τον ορισμό ονομάτων (όπως είναι η "var" στην Pascal, για παράδειγμα) ή συναρτήσεων (όπως η "function", πάλι στην Pascal). Μια γραμμή που ξεκινάει με το όνομα ενός τύπου, εκλαμβάνεται σαν ορισμός μεταβλητής ή συνάρτησης, ανάλογα με τον αν υπάρχουν, ή όχι, παρενθέσεις που περικλείουν (τυπικές) παραμέτρους συνάρτησης.

· Ο χρήστης μπορεί να ορίσει δικούς του τύπους (και σύνθετους), εάν το επιθυμεί. Μπορεί επίσης να ορίσει και τύπους εγγραφών (structs στη C, records σε άλλες γλώσσες).

· Μπορούν να οριστούν πίνακες, αν και δεν υπάρχει ειδική λέξη-κλειδί για τον ορισμό τους (όπως το "array" στην Pascal). Η δεικτοδότηση τους γίνεται με χρήση αγκυλών ([]), αν και πολύ συχνά γίνεται χρήση αριθμητικής δεικτών. Το πρώτο στοιχείο κάθε πίνακα δεικτοδοτείται πάντα από το μηδέν (0). Π.χ.: Το στοιχείο month[0] είναι το πρώτο στοιχείο του πίνακα month. Τέλος, δεν υπάρχουν τελεστές για την σύγκριση ή εκχώρηση πινάκων.

· Είναι δυνατή η δημιουργία απαριθμήσιμων τύπων με τη χρήση της λέξης-κλειδί "enum", οι οποίοι μπορούν να χρησιμοποιηθούν όπου οι ακέραιοι και αντίστροφα.

· Δεν υπάρχει ιδιαίτερος τύπος για αλφαριθμητικά, τα οποία παραδοσιακά υλοποιούνται και αντιμετωπίζονται σαν πίνακες από χαρακτήρες, και έχουν έναν μηδενικό χαρακτήρα να σημαδεύει το τέλος τους (null-terminated arrays of characters).

· Είναι δυνατή η άμεση προσπέλαση χαμηλού επιπέδου στη μνήμη του υπολογιστή με τη χρήση δεικτών.

· Οι υπορουτίνες που δεν επιστρέφουν τιμή ("procedures" σε άλλες γλώσσες) είναι συναρτήσεις που ορίζονται να είναι τύπου "void" (ψευδοτύπος που δείχνει την απουσία επιστρεφόμενης τιμής, αλλά χρησιμοποιείται και για δείκτες που δεν δείχνουν σε αντικείμενο συγκεκριμένου τύπου).

· Δεν μπορούν να οριστούν συναρτήσεις μέσα σε άλλες συναρτήσεις (εμφωλιασμένες).

· Οι δείκτες σε συναρτήσεις και δεδομένα επιτρέπουν την υλοποίηση πολυμορφισμού στην πράξη.

· Ο προεπεξεργαστής της γλώσσας επιτρέπει τον ορισμό μακροεντολών, την συγχώνευση αρχείων πηγαίου κώδικα, καθώς και την μεταγλώττιση υπό συνθήκες.

· Αρχεία πηγαίου κώδικα μπορούν να μεταγλωττιστούν χωριστά και να συνδεθούν μαζί, ενώ υπάρχει η δυνατότητα ελέγχου της ορατότητας συναρτήσεων και μεταβλητών στα άλλα αρχεία (πέρα από αυτό στο οποίο ορίζονται) με το χαρακτηρισμό τους ως "static" ή "extern".

· Οι πολύπλοκες λειτουργίες, όπως οι λειτουργίες εισόδου/εξόδου, ο χειρισμός των αλφαριθμητικών, καθώς και οι μαθηματικές συναρτήσεις, έχουν ανατεθεί, με συνεπή τρόπο, στις αντίστοιχες βιβλιοθήκες.

· Η C δεν διαθέτει κάποιες από τις δυνατότητες νεώτερων γλωσσών, όπως τον προσανατολισμό στα αντικείμενα και την συλλογή απορριμμάτων (garbage collection).

1.2 Ιστορική Αναδρομή

Η γλώσσα C οποία αναπτύχθηκε στις αρχές της δεκαετίας του ’70 από τον Dennis Ritchie στα Bell Labs. σύμφωνα με τον D. Ritchie, η πιο δημιουργική περίοδος υπήρξε το 1972. Η νέα γλώσσα ονομάστηκε "C" λόγω του ότι πολλά από τα χαρακτηριστικά της προήλθαν από μια παλαιότερη γλώσσα, η οποία ονομαζόταν "B". ο Ken Thompson το παρουσιάζει ως απλούστευση μιας έκδοσης της γλώσσας προγραμματισμού BCPL, αλλά είχε επίσης δημιουργήσει μία γλώσσα που ονομαζόταν Bon προς τιμήν της συζύγου του Bonnie.

Μέχρι το 1973, η C είχε γίνει αρκετά ισχυρή και αποτελεσματική, ώστε το μεγαλύτερο μέρος του πυρήνα του UNIX (UNIX kernel), γραμμένο αρχικά σε PDP-11/20 assembly, επανεγγράφηκε σε C. Ήταν ένας από τους πρώτους πυρήνες που υλοποιήθηκε σε μια γλώσσα διαφορετική της assembly. (Προηγούμενα παραδείγματα περιλαμβάνουν το Multics system (γραμμένο σε PL/I), και το MCP (Master Control Program) για το Burroughs B5000 γραμμένο σε ALGOL το 1961.)

Με την πάροδο του χρόνου η γλώσσα C άρχισε να χρησιμοποιείται και σε άλλα πεδία εφαρμογών, πέραν του προγραμματισμού συστημάτων. Η εμφάνιση των μεταγλωττιστών της γλώσσας στο MS–DOS και ο μεγάλος αριθμός προγραμμάτων βιβλιοθήκης που κατασκευάστηκαν, οδήγησαν τη γλώσσα στο απόγειό της στα τέλη της δεκαετίας του 1980. Βέβαια η γλώσσα γνώρισε πολλές αλλαγές, οδηγούμενη τελικά στην επονομαζόμενη ANSI έκδοση. Η τελευταία ενημέρωση της γλώσσας έγινε το 1999.

Από την ήδη υπάρχουσα γλώσσα προγραμματισμού C αναπτύχθηκε η γλώσσα C++ από τον Μπιάρνε Στρούστρουπ το 1979, στα εργαστήρια Bell της AT&T.

1.3 Μεταγλώττιση προγράμματος

΄Οπως προαναφέρθηκε κάθε υπολογιστής διαθέτει τη δική του γλώσσα μηχανής, εντολές της οποίας μπορεί να εκτελεί μόνο. Συνεπώς, ένα πρόγραμμα σε οποιαδήποτε γλώσσα προγραμματισμού υψηλού επιπέδου, όπως είναι η C, για να μπορέσει να εκτελεσθεί πρέπει πρώτα να μεταγλωττισθεί (compilation). Η μεταγλώττιση είναι μια διαδικασία κατά την Εισαγωγή στις Γλώσσες Προγραμματισμού :

- ➢ Γλώσσα Μηχανής και Συμβολικές Γλώσσες
- ➢ Γλώσσες Υψηλού Επιπέδου
- ➢ Τυπικός ορισμός μιας Γλώσσας
- ➢ Μεταγλώττιση προγράμματος

Ένα πρόγραμμα π.χ σε γλώσσα C, μεταφράζεται στη γλώσσα μηχανής του υπολογιστή στον οποίο πρόκειται να εκτελεσθεί. Ο μεταγλωττιστής (compiler) ο οποίος εκτελεί τη μεταγλώττιση ενός προγράμματος σε γλώσσα υψηλού επιπέδου σε πρόγραμμα σε γλώσσα μηχανής είναι από τα πιο σημαντικά προγράμματα του λειτουργικού συστήματος ενός υπολογιστή.

1.4 Εργαλεία ανάλυσης προβλημάτων

Για την ανάλυση ενός προβλήματος και την κατάρτιση του μοντέλου που θα υλοποιηθεί σε κώδικα υπάρχουν τρία εργαλεία:

1. Η φυσική γλώσσα (natural language), σύμφωνα με την οποία το πρόβλημα αναλύεται σε απλές προτάσεις της καθομιλουμένης γλώσσας, χρησιμοποιώντας το συντακτικό αυτής.
2. Το διάγραμμα ροής (flow chart), σύμφωνα με το οποίο απεικονίζεται γραφικά η εξέλιξη του προβλήματος, με χρήση ειδικών συμβόλων.
3. Ο ψευδοκώδικας (pseudocode), ο οποίος μετασχηματίζει τη φυσική γλώσσα σε μία σειρά προτάσεων που χρησιμοποιούν το συντακτικό γλώσσας προγραμματισμού, χωρίς να ακολουθούν επακριβώς το φορμαλισμό συγκεκριμένης γλώσσας.

1.5 Ένα απλό πρόγραμμα σε C

΄Ενα πρόγραμμα σε C συνίσταται από μια ή περισσότερες συναρτήσεις. Η μια από αυτές είναι η συνάρτηση main η οποία είναι η κύρια συνάρτηση ενός προγράμματος σε C που σημαίνει ότι η εκτέλεση του προγράμματος ξεκινά από τη συνάρτηση main και κάποιες ή όλες οι υπόλοιπες συναρτήσεις καλούνται μέσα από αυτή.

Οι συναρτήσεις ενός προγράμματος διακρίνονται στις εξής κατηγορίες :

• σε αυτές της πρότυπης βιβλιοθήκης (standard lib) της ANSI C και

• σε αυτές που ο χρήστης δημιουργεί (user defined).

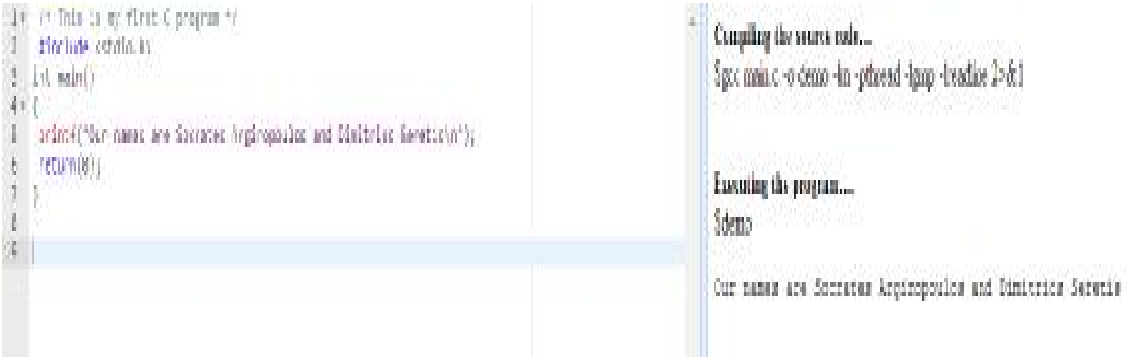
Για να γίνει χρήση κάποιων συναρτήσεων από βιβλιοθήκη του περιβάλλοντος προγραμματισμού σε ένα πρόγραμμα C, θα πρέπει να συμπεριληφθούν τα αρχεία επικεφαλίδες (τα οποία φέρουν την επέκταση .h) στα οποία είναι δηλωμένες οι συναρτήσεις αυτές. Αυτό πραγματοποιείται με την εντολή #include του προεπεξεργαστή (preprocessor) της C.

Οι εντολές συμπερίληψης αρχείων επικεφαλίδας σε ένα πρόγραμμα C εμφανίζονται στην αρχή ενός προγράμματος C. Στον παρακάτω πίνακα παρουσιάζεται ένα μικρό σύνολο αρχείων επικεφαλίδας που διαθέτει η ANSI C καθώς και περιγραφή του περιεχομένου τους.

Αρχείο Περιγραφή Συναρτήσεων	Αρχείο Περιγραφή Συναρτήσεων
stdarg.h	Διαχείρισης Μεταβλητών Λιστών Ορισμάτων
io.h	Διαχείρισης εισόδου / εξόδου σε χαμηλό επίπεδο
imits.h & float.h	Ορίζει τα όρια των ακεραίων
math.h	Μαθηματικές Συναρτήσεις
mem.h	Διαχείριση Δυναμικής Μνήμης
stdio.h	Διαχείριση εισόδου και εξόδου σε υψηλό επίπεδο
stdlib.h	Συχνά χρησιμοποιούμενες συναρτήσεις
string.h & ctype.h	Συναρτήσεις Επεξεργασίας αλφαριθμητικών
time.h	Διαχείρισης ώρας και ημερομηνίας

Παρακάτω δίνεται ένα πρόγραμμα σε C το οποίο όταν εκτελεσθεί τυπώνεται στην οθόνη το κείμενο “Our names are Socrates Argiropoulos and Dimitrios Seretis”. Το πρόγραμμα αυτό αποτελείται μόνο από την συνάρτηση main το σώμα της οποίας ορίζεται από το αριστερό άγκιστρο και ολοκληρώνεται με το δεξί άγκιστρο. Στο εσωτερικό της main γίνεται κλήση της συνάρτησης printf της πρότυπης βιβλιοθήκης με όρισμα το κείμενο ‘‘Our names are Socrates Argiropoulos and Dimitrios Seretis’’ (γραμμή με αριθμό 5 του προγράμματος). Η συνάρτηση printf της πρότυπης

βιβλιοθήκης για να χρησιμοποιηθεί στο εν λόγω πρόγραμμα πρέπει να προηγηθεί η ενσωμάτωση του αρχείου επικεφαλίδα stdio.h στο οποίο είναι δηλωμένη (γραμμή με αριθμό 2 στο πρόγραμμα) και το οποίο περιβάλλεται από τα σύμβολα <>



Επεξήγηση του προγράμματος

Πιο αναλυτικά:

1 Πως γράφω σχόλια.

Η πρώτη γραμμή περιέχει ένα σχόλιο:

/* This is my first C program */

2 Η εντολή #include και τα αρχεία επικεφαλίδων.

Η δεύτερη γραμμή του προγράμματος είναι η ακόλουθη:

#include<stdio.h> Η γραμμή αυτή ξεκινά με ένα #, το οποίο ακολουθείται από την εντολή include. Το include είναι μια εντολή του προεπεξεργαστή η οποία ψάχνει να βρει το αρχείο stdio.h. Επί πλέον η εντολή include ζητά από τον προεπεξεργαστή να τοποθετήσει το αρχείο stdio.h στη θέση της εντολής μέσα στο πρόγραμμα.

Τα αρχεία τα οποία περιλαμβάνονται με την εντολή #include, όπως το stdio.h, ονομάζονται αρχεία επικεφαλίδων και περιέχουν συναρτήσεις έτοιμων προς χρήση για τον προγραμματιστή. Εκτός από το stdio.h υπάρχουν και άλλα αρχεία επικεφαλίδων, όπως τα math.h, stdlib.h, string.h κτλ. Κάθε φορά ο προγραμματιστής πρέπει να περιλαμβάνει τα απαραίτητα αρχεία επικεφαλίδων στην αρχή του κώδικά του.

3 Η συνάρτηση main() αποτελεί μια ειδική συνάρτηση της C.

Κάθε πρόγραμμα πρέπει να περιέχει μία και μόνο μία συνάρτηση main() η οποία εκτελείται πάντα πρώτη ακόμη και εάν είναι στο κάτω μέρος του προγράμματος. Στο συγκεκριμένο παράδειγμα η συνάρτηση main έχει δηλωθεί ως ακέραιου τύπου (int) ενώ δεν περιέχει κάποιο όρισμα

4 Η συνάρτηση printf() εμφανίζει και τυπώνει μηνύματα στην οθόνη μας. Ο χαρακτήρας \n ο οποίος εμφανίζεται στο τέλος του μηνύματος λέει στον υπολογιστή να μετακινήσει τον δρομέα στην αρχή της επόμενης γραμμής. Η συνάρτηση printf() ορίζεται στο αρχείο επικεφαλίδας stdio.h.

5 Η εντολή return.

Όλες οι συναρτήσεις της C μπορούν να επιστρέφουν τιμές. Στο συγκεκριμένο παράδειγμα η συνάρτηση main η οποία έχει δηλωθεί ως ακέραια συνάρτηση επιστέφει την τιμή μηδέν και με αυτόν τον τρόπο το πρόγραμμα τερματίζεται κανονικά.

Στο παράδειγμα χρησιμοποιείται online μεταγλωττιστής, όπου εμφανίζει το μήνυμα που δίνει το πρόγραμμα, «Our names are Socrates Argiropoulos and Dimitrios Seretis»

Το πρώτο πρόγραμμα στη c , διαβάζει τα ονόματα και τα εκτυπώνει στην οθόνη.

```
#include <stdio.h>

main()

{

char  msg[]= "Dimitris Seretis and Socrates Argiropoulos";

//clrscr();

printf("\nEinai to prwto programma se  c");

printf("\nta onomata mas  einai : %s", msg);

scanf("%s", &msg); /* ektypwsh tou mhnymatos */

} /* end of main */
```

```
1 #include <stdio.h>
2 main()
3 {
4 char  msg[]= "Dimitris Seretis and Socrates Argiropoulos";
5 //clrscr();
6 printf("\nEinai to prwto programma se  c");
7 printf("\nta onomata mas  einai : %s", msg);
8 scanf("%s", &msg); /* ektypwsh tou mhnymatos */
9 } /* end of main */
```

2. ΜΕΤΑΒΛΗΤΕΣ–ΣΤΑΘΕΡΕΣ

2.1 Μεταβλητές

Οι μεταβλητές είναι το πιο θεμελιώδες μέρος οποιαδήποτε γλώσσας. Μία μεταβλητή είναι ένα συμβολικό όνομα που μπορεί να δεχτεί διάφορες τιμές. Με τη βοήθεια των μεταβλητών εισάγονται δεδομένα στο πρόγραμμα και σε αυτές αποθηκεύονται τα αποτελέσματα των υπολογισμών για να εμφανιστούν στην οθόνη ή να χρησιμοποιηθούν στη συνέχεια του προγράμματος.

Μία μεταβλητή χαρακτηρίζεται από τον τύπο των δεδομένων που περιέχει. Οι περισσότερες γλώσσες χρησιμοποιούν τους ίδιους γενικούς τύπους δεδομένων, όπως οι ακέραιοι , οι αριθμοί κινητής υποδιαστολής, οι χαρακτήρες κ.α. Για να δηλώσετε τι τύπου δεδομένα θα περιέχει μία μεταβλητή πριν το όνομά της θα γράψετε την κατάλληλη δεσμευμένη λέξη.

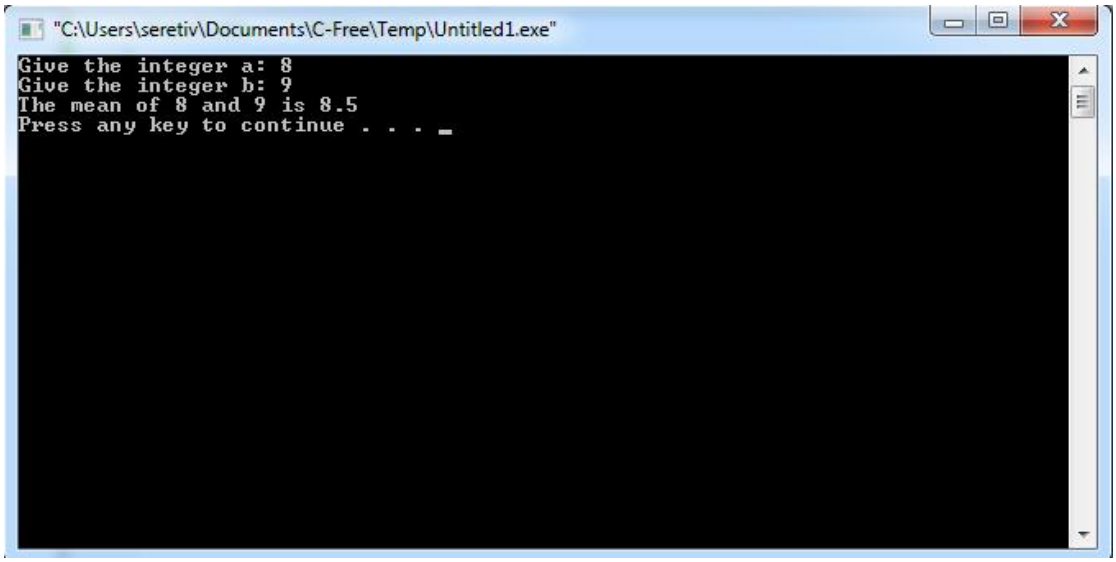
Ως όνομα μπορείτε να δώσετε όποιο όνομα επιθυμείτε αλλά όχι κάποια δεσμευμένη λέξη από τη C++ . Να είστε προσεχτικοί με την χρήση των ονομάτων των μεταβλητών. Θα πρέπει όπως ακριβώς τη δηλώσετε έτσι ακριβώς να την καλείτε. Η C++ είναι ευαίσθητη στη χρήση μικρών ή κεφαλαίων χαρακτήρων σε αντίθεση με την Pascal. Έτσι εάν μία μεταβλητή τη δηλώσαμε με μικρούς χαρακτήρες δεν μπορούμε να την χρησιμοποιήσουμε στη συνέχεια με κεφαλαία.

2.2 Δηλώσεις μεταβλητών

Όλες οι μεταβλητές στη C πρέπει να δηλωθούν προτού χρησιμοποιηθούν, ώστε ο compiler να γνωρίζει τον τύπο τους όταν τις πρωτοσυναντήσει. Ο κανόνας είναι πως μια δήλωση μεταβλητών συνίσταται από ένα όνομα τύπου ακολουθούμενο από ένα ή περισσότερα (χωριζόμενα με κόμμα) ονόματα μεταβλητών. Οι δηλώσεις μεταβλητών μπορούν να γραφτούν στο χώρο του κειμένου εκτός συναρτήσεων. Η λίστα των μεταβλητών που δηλώνονται τερματίζεται με το σύμβολο ; (semicolon) όπως σχεδόν και όλες οι άλλες εντολές στη C.

Π.χ. ένα πρόγραμμα που υπολογίζει το μέσο όρο δυο ακέραιων τιμών:

```
#include   <stdio.h>

int        a;
double     mean;

main() {
           int         b;
           double      useless_variable_1;

           printf("Give the integer a: ");
           scanf("%d", &a);
           printf("Give the integer b: ");
           scanf("%d", &b);
           mean = 0.5*(a+b);
           printf("The mean of %d and %d is %g\n", a, b, mean);
}
```



2.3 Σταθερές

Στη C, όπως και σε κάθε γλώσσα προγραμματισμού, ο προγραμματιστής μπορεί να ορίσει και σταθερές η τιμή των οποίων δεν αλλάζει καθ΄ όλη τη διάρκεια εκτέλεσης του προγράμματος. Ακολουθούνται οι ίδιοι κανόνες ονοματολογίας που αναφέρθηκαν και για τις μεταβλητές με τη διαφορά ότι το όνομα μιας σταθεράς συντίθεται μόνο από κεφαλαία γράμματα. Υπάρχουν δύο κατηγορίες σταθερών στη C:

• οι literal (κυριολεκτικές): στην εντολή ανάθεσης π = 3.14159 η literal σταθερά είναι ο αριθμός κινητής υποδιαστολής 3.14159.

• οι συμβολικές, όπου στη literal σταθερά 3.14159 αντιστοιχούμε το συμβολικό όνομα PI το οποίο και χρησιμοποιούμε στο πρόγραμμα. Υπάρχουν 2 τρόποι στη C για να ορισθεί μια τέτοια αντιστοιχία :

1. με τον προσδιοριστή const: const float PI = 3.14159;

2. Μέσω της εντολής του προεπεξεργαστή #define:
   #define PI 3.14159

Στις αριθμητικές παραστάσεις χρησιμοποιείται πλέον το συμβολικό όνομα PI. ΄Οταν είναι επιθυμητή η αναίρεση μιας συμβολικής σταθεράς αυτό μπορεί να γίνει με τη χρήση της εντολής του προεπεξεργαστή #undef. ΄Ετσι για να αναιρέσουμε την σταθερά PI γράφουμε :
#undef PI
Μια ακέραια σταθερά εξ ορισμού είναι τύπου int και προσημασμένη (signed) εκτός και αν είναι μεγάλος ακέραιος αριθμός οπότε εκλαμβάνεται ως long. Αν είναι επιθυμητό για τις ανάγκες του κώδικα μια μικρή ακέραια σταθερά να εκληφθεί από τον μεταγλωττιστή ως long ή μη προσημασμένη θα πρέπει να έχει ως κατάληξη το L ή U, π.χ. 2048L ή 2048U ή 2048UL. Μια ακέραια σταθερά επίσης μπορεί να αναπαρασταθεί στο οκταδικό ή στο δεκαεξαδικό χρησιμοποιώντας το πρόθεμα 0 και 0x ή 0X αντίστοιχα,

Οι σταθερές κινητής υποδιαστολής περιέχουν μια υποδιαστολή (π.χ. 23.45) ή μια δύναμη ($2^{\varepsilon-3}$) (που είναι ο 2×10^{-3}) ή και τα δύο και ο τύπος τους είναι double εκτός και αν υπάρχει η κατάληξη :
• F ή f οπότε υποδηλώνεται μια σταθερά float

• L ή l οπότε υποδηλώνεται μια σταθερά double

2.4 Βασικοί τύποι δεδομένων

Η C διαθέτει τέσσερεις βασικούς τύπους δεδομένων :
• char: πρόκειται για ακεραίους μεγέθους ενός byte που σημαίνει ότι μπορεί να αναπαρασταθούν το πολύ 256 χαρακτήρες. Ο τύπος αυτός είναι ικανός να αναπαραστήσει τους ASCII χαρακτήρες που στο σύνολο τους είναι 128.
• int: πρόκειται για ακεραίους μεγέθους μεγαλύτερου του ενός byte και μικρότερου αυτού που μπορεί να αναπαρασταθεί από το μηχάνημα.
• float: αριθμοί κινητής υποδιαστολής απλής ακρίβειας
• double: αριθμοί κινητής υποδιαστολής διπλής ακρίβειας

Μια σημαντική παράμετρος, ενός τύπου δεδομένων στη C είναι το μέγεθος του που είναι το πλήθος των byte που καταλαμβάνει ένα δεδομένο αυτού του τύπου. Το μέγεθος σε byte για κάθε τύπο μπορούμε να το υπολογίσουμε με τη βοήθεια του τελεστή μεγέθους sizeof τυλίγοντας το όνομα του τύπου με παρενθέσεις δηλαδή sizeof(<όνομα_τύπου>). Για παράδειγμα για να υπολογίσουμε το μέγεθος του τύπου float γράφουμε sizeof(float). Τα μεγέθη των βασικών τύπων ορίζονται ως σταθερές στα αρχεία < limits.h > και < float.h >. Συνεπώς, σε κάθε πρόγραμμα στο οποίο πρόκειται να χρησιμοποιήσουμε τις σταθερές αυτές πρέπει στην αρχή να συμπεριλάβουμε τα παραπάνω αρχεία επικεφαλίδες δηλαδή :
#include < limits.h >
#include < float.h >

Η C διαθέτει ένα σύνολο προσδιοριστών οι οποίοι εφαρμόζονται στους βασικούς

τύπους. ΄Ετσι για τον τύπο int υπάρχουν οι προσδιοριστές short και long. Για παράδειγμα γράφοντας short int ή short ορίζεται ένας μικρού μεγέθους ακέραιος και long int ή long ορίζεται ένας μεγάλου μεγέθους ακέραιος. Σε κάθε υπολογιστικό σύστημα αυτό που ισχύει είναι sizeof (short) ≤ sizeof (int) ≤ sizeof (long). Στους περισσότερους προσωπικούς υπολογιστές το μέγεθος του int και long είναι 4 byte (32-bit) ενώ του short είναι 2 byte (16-bit).

Επίσης στους περισσότερους προσωπικούς υπολογιστές το μέγεθος ενός αριθμού κινητής υποδιαστολής απλής ακρίβειας (float) είναι 4 byte (32-bit) και ενός διπλής ακρίβειας (double) είναι 8 byte (64-bit).

Το πρότυπο που χρησιμοποιείται για την αναπαράσταση των αριθμών κινητής υποδιαστολής είναι το 754 της IEEE (Institute of Electrical and Electronics Engineers).

Μπορεί να χρησιμοποιηθεί ο προσδιοριστής long για τον τύπο double, δηλαδή έχουμε έναν τύπο long double, προκειμένου να αυξήσουμε το μέγεθος και κατ΄ επέκταση την ακρίβεια ενός αριθμού κινητής υποδιαστολής. Βέβαια θα πρέπει να υποστηρίζει και η μηχανή μεγαλύτερη ακρίβεια διαφορετικά ο τύπος long double εκφυλίζεται ουσιαστικά στον double.

Δύο ακόμα προσδιοριστές είναι οι signed (προσημασμένος) unsigned (μη προσημασμένος) οι οποίοι μπορούν να εφαρμοστούν μόνο στον τύπο char και σε όλους τους ακεραίους αφού σύμφωνα με το πρότυπο 754 της IEEE όλοι οι αριθμοί κινητής υποδιαστολής είναι προσημασμένοι. Οι προσδιοριστές αυτοί καθορίζουν αν ένας αριθμός είναι προσημασμένος ή όχι. Ο τύπος char είναι ένας προσημασμένος ακέραιος των 8-bit πράγμα που σημαίνει ότι το πιο σημαντικό bit αναπαραριστά το πρόσημο του, το 1 αντιστοιχεί στο − και το 0 στο +. Για παράδειγμα ο 8-bit ακέραιος 10000001 αναπαριστά το −1 (το περισσότερο σημαντικό ψηφίο δεν λογίζεται στον υπολογισμό του μέτρου του αριθμού) και με τον προσδιοριστή unsigned αναπαριστά το 128 αφού το περισσότερο σημαντικό ψηφίο λογίζεται πλέον στο μέτρο του αριθμού, δηλαδή $1\times2^7 + 1\times2^0 = 128$. Με άλλα λόγια ο τύπος unsigned char καλύπτει τους ακέραιους από 0 . . . 255 ενώ ο προσημασμένος από −128 . . . 127. Οι προσδιοριστές αυτοί εφαρμόζονται με τον ίδιο τρόπο και σε όλες τις εκδόσεις του τύπου int.

Στη C αν θέλουμε να μετονομάσουμε έναν ήδη υπάρχοντα τύπο δεδομένων μπορούμε να χρησιμοποιήσουμε τη λέξη κλειδί typedef ως εξής :

typedef < newTypeName > < oldTypename >;

Στο τέλος της παραπάνω εντολής μετονομασίας τίθεται ο χαρακτήρας ; που στη C είναι ο χαρακτήρας που δηλώνει το τέλος μιας εντολής. Για παράδειγμα αν θέλουμε να μετονομάσουμε το τύπο int σε integer γράφουμε :

typedef integer int;

ενώ για δημιουργήσουμε τον τύπο byte για την αναπαράσταση των ακεραίων από 0 . . . 255 γράφουμε :

typedef byte unsigned char;

2.5 **Εμβέλεια** (Scope) **Μεταβλητών**

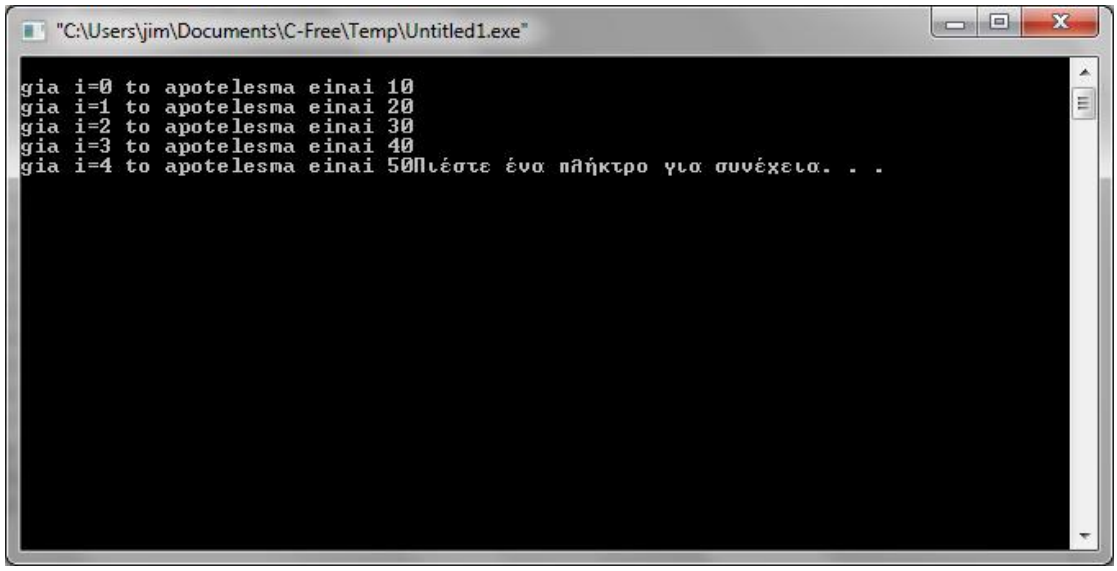
Η ***εμβέλεια*** *(scope)* μίας μεταβλητής περιορίζεται στο σώμα της συνάρτησης που θα δηλωθεί ή εάν αποτελεί παράμετρο τότε πάλι στο σώμα της συνάρτησης στην οποία είναι παράμετρος. Με τον όρο εμβέλεια εννοείται που ακριβώς η μεταβλητή είναι ορατή, δηλαδή που μπορεί να χρησιμοποιηθεί. Στην περίπτωση που δύο μεταβλητές ακόμη και με ίδιο όνομα και τύπο έχουν δηλωθεί σαν τοπικές δύο συναρτήσεων τότε η C αντιμετωπίζει σαν εντελώς διαφορετικές και ξένες μεταξύ τους. Βέβαια εδώ τίθεται το εξής ερώτημα: τι θα γινόταν αν μία μεταβλητή έπρεπε να είναι ορατή σε όλο το πρόγραμμα και σε όλες τις συναρτήσεις του; Για να συμβεί αυτό θα πρέπει η μεταβλητή να δηλωθεί στην αρχή του προγράμματος (αμέσως μετά τα πιθανά include / define). Οι μεταβλητές αυτού του τύπου ονομάζονται **καθολικές μεταβλητές** (global variables).

Κάθε μεταβλητή στην C χαρακτηρίζεται από δύο παραμέτρους: τον τύπος της και την εμβέλειά της. Οι τύποι εμβέλειας είναι οι ακόλουθες τέσσερις :

1. auto: Όλες οι μεταβλητές όπως έχουν χρησιμοποιηθεί μέχρι τώρα εκτός των καθολικών. Οι τοπικές μεταβλητές μίας συνάρτησης σε περίπτωση που δεν δηλωθούν διαφορετικά θεωρούνται auto - αυτόματες.
2. extern: Οι μεταβλητές αυτής της τάξης θεωρούνται εξωτερικές μεταβλητές, δηλαδή, σε περίπτωση μετάφρασης πολλών προγραμμάτων C ταυτόχρονα, μπορούν να χρησιμοποιηθούν σε όλα τους και να μην είναι περιορισμένες μόνο στο πρόγραμμα που δηλώθηκαν.
3. register: Υπάρχει περίπτωση κάποιες μεταβλητές να επιδρούν πολλή στην ταχύτητα ενός προγράμματος (για παράδειγμα η μεταβλητή ελέγχου ενός for – βρόγχου). Θα ήταν σαφώς ενδεικνυόμενο αυτές οι μεταβλητές να αποθηκευθούν σε μία θέση μνήμης η οποία προσφέρεται για γρηγορότερες προσπελάσεις όπως οι καταχωρητές (registers). Αυτές είναι οι λεγόμενες register μεταβλητές. Βέβαια δεν είναι δυνατόν να δηλωθούν όλες οι μεταβλητές με αυτόν τον τρόπο γιατί είναι περιορισμένος ο αριθμός των καταχωρητών οπότε η C παίρνει την «πρωτοβουλία» και ακυρώνει κάποιες από αυτές τις δηλώσεις.
4. static: Οι στατικές μεταβλητές μοιάζουν με τις αυτόματες αλλά με την εξής διαφορά: Όταν μία συνάρτηση κληθεί έχοντας αυτόματες τοπικές μεταβλητές αυτές με την λήξη της συνάρτησης εξαφανίζονται με αποτέλεσμα εάν αργότερα η συνάρτηση ξανακληθεί να έχει χαθεί η προηγούμενη τιμή τους. Σε αντίθεση, εάν οι τοπικές μεταβλητές έχουν δηλωθεί σαν στατικές δεν χάνουν την τιμή τους ακόμη και μετά την χρήση μίας συνάρτησης με αποτέλεσμα όταν η ίδια συνάρτηση ξανακληθεί να έχουν διατηρήσει την τιμή από την προηγούμενη κλήση της.

Παράδειγμα

```
#include <stdio.h>
int synartisi();
main()
{
int a;
register int i;
for (i=0; i<5; i++) {
a = synartisi();
printf("\ngia i=%d to apotelesma ths synarthshs %d",i,a);
}
}
int synartisi()
{
static int k;
register i;
for (i=0; i<10; i++)
k++;
return k;
}
```



Στο παραπάνω πρόγραμμα οι εκτυπώσεις είναι οι ακόλουθες:
Για i=1 το αποτέλεσμα της συνάρτησης είναι 10
Για i=2 το αποτέλεσμα της συνάρτησης είναι 20
Για i=3 το αποτέλεσμα της συνάρτησης είναι 30
Για i=4 το αποτέλεσμα της συνάρτησης είναι 40
Για i=5 το αποτέλεσμα της συνάρτησης είναι 50

Ενώ θα περίμενε κανείς συνέχεια το ίδιο αποτέλεσμα της συνάρτησης. Αυτό οφείλεται στο ότι η μεταβλητή k δεν χάνει την τιμή που είχε πάρει από την προηγούμενη κλήση της συνάρτησης και αυτό επειδή έχει δηλωθεί σαν static.

Τέλος, οι register μεταβλητές έχουν σαν αποτέλεσμα την ταχύτερη εκτέλεση του προγράμματος.

2.6 Ειδικοί χαρακτήρες

Ο κώδικας ASCII

Υπάρχει ένας κώδικας αντιστοίχισης, ο ASCII (American Standard Code for Information Interchange), ανάμεσα στους αριθμούς από 0 έως 255 - αυτοί είναι οι αριθμοί που μπορούν να γραφούν με 8 δυαδικά ψηφία - και σε διάφορα σύμβολα, που μπορεί να υπάρχουν στο πληκτρολόγιό σας, μπορεί και όχι. Τα σύμβολα αυτά συμπεριλαμβάνουν όλα τα γράμματα του λατινικού αλφαβήτου, μικρά και κεφαλαία, σημεία στίξης, διαφόρων ειδών παρενθέσεις, και διάφορα άλλα σύμβολα που δεν έχουν κάποιο σχήμα αλλά χρησιμοποιούνται κατά σύμβαση για κάποιο συγκεκριμένο σκοπό. Υπάρχει για παράδειγμα στο Unix ένας χαρακτήρας, ο line-feed (LF) με αριθμό 10, που χρσιμοποιείται για να δηλώσει ότι έχει φτάσει το τέλος της τρέχουσας γραμμής. Έτσι όταν ένα πρόγραμμα διαβάζει ένα αρχείο και συναντήσει το χαρακτήρα LF συμπεραίνει συνήθως (αυτό πραγματικά είναι θέμα του συγκεκριμένου προγράμματος) ότι έχει επέλθει αλλαγή γραμμής και ενεργεί ανάλογα. Αν, για παράδειγμα, το πρόγραμμα αυτό είναι πρόγραμμα εκτύπωσης στον εκτυπωτή τότε, κατά πάσα πιθανότητα, θα στείλει στον εκτυπωτή εντολή να αλλάξει γραμμή εκτύπωσης προτού συνεχίσει να του στέλνει άλλους χαρακτήρες για εκτύπωση.

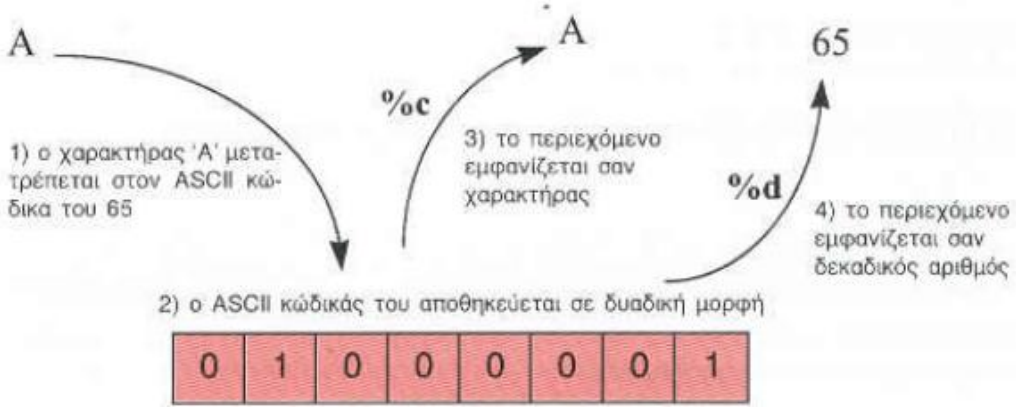
Για να είμαστε λίγο πιο ακριβείς, ο κώδικας ASCII μόνο αντιστοιχεί χαρακτήρες στα νούμερα από 0 έως 127 (αριθμοί που μπορούν να παρασταθούν με 7 δυαδικά ψηφία). Πολλά σύμβολα, όπως για παράδειγμα αυτά της Ελληνικής αλφαβήτου μένουν απ' έξω, και το κομμάτι από 128 έως 255 χρησιμοποιείται, μεταξύ άλλων, και για την παράσταση αλφαβήτων διαφορετικών από το λατινικό (το οποίο παρίσταται από το standard ASCII κάτω από το 128).

Μπορείτε να δείτε στον παρακάτω πίνακα τους χαρακτήρες για τα ASCII codes από 0 έως 127.

Χαρακτήρας	**Λειτουργία**
\n	αλλαγή γραμμής
\t	αλλαγή στήλης (tab)
\'	προσθήκη αποστρόφου
\\	προσθήκη χαρακτήρας backslash
\"	προσθήκη εισαγωγικού
\0	χαρακτήρας με ASCII = 0 (null)

\037	χαρακτήρας με ASCII = 37 (οκταδικό)
\x1f	χαρακτήρας με ASCII = 1f (δεκαεξαδικό)

Αποθήκευση και ανάκτηση ASCII χαρακτήρα



3. Τελεστές

Οι τελεστές (operators) είναι σύμβολα ή λέξεις που αναπαριστούν συγκεκριμένες διεργασίες, οι οποίες εκτελούνται επί ενός ή περισσότερων δεδομένων. Τα δεδομένα καλούνται τελεστέοι (operands) και μπορούν να είναι μεταβλητές, σταθερές ή ακόμη και κλήσεις συναρτήσεων.

Οι τελεστές χρησιμοποιούνται για το σχηματισμό εκφράσεων. Μία έκφραση, εν γένει, αποτελείται από έναν ή περισσότερους τελεστέους και από έναν ή περισσότερους Τελεστές.

Κατηγορία	Ενδεικτικοί τελεστές
Αριθμητικοί	+ - * /
Λογικοί	&& || !
Συσχετιστικοί	> >= = = !=
Διαχείρισης bits	>> & ! ^
Διαχείρισης μνήμης	& [] . ->

3.1 Αριθμητικοί Τελεστές

Η C διαθέτει επτά συνολικά αριθμητικούς τελεστές με τους οποίους μπορεί να σχηματισθούν αριθμητικές εκφράσεις. Από αυτούς δύο είναι μοναδιαίοι τελεστές : ++, −− οι οποίοι μπορεί να βρίσκονται είτε δεξιά είτε αριστερά μιας μεταβλητής. ΄Ετσι όταν εκτελείται μια έκφραση της μορφής x++ αυτό που πρακτικά γίνεται είναι να αυξηθεί η μεταβλητή x κατά 1 δηλαδή οι εκφράσεις x + + και + + x είναι ισοδύναμες με την x = x + 1. ΄Ομοια και οι x − − και − − x είναι ισοδύναμες με την x = x − 1.

Είναι εύλογο να αναρωτηθεί κανείς ποιά η διαφορά μεταξύ των εκφράσεων x + + και της + + x. Η διαφορά είναι ότι όταν εκτελείται η x + + πρώτα επιστρέφεται η τρέχουσα τιμή της x και μετά αυξάνεται η μεταβλητή κατά ένα. Ενώ όταν εκτελείται η + + x πρώτα αυξάνεται η μεταβλητή κατά 1 και στη συνέχεια επιστρέφεται η νέα τιμή της μεταβλητής. Για να το κατανοήσουμε καλύτερα ας δούμε τον παρακάτω κώδικα :

```
int z, y, x = 5;
y = + + x;
z = x + +;
```

Μετά την εκτέλεση των δύο εκφράσεων η μεταβλητή y θα έχει την τιμή 6, η z την τιμή 6 και η x θα έχει την τιμή 7.

Οι υπόλοιποι πέντε τελεστές : +, -, /, *, % είναι δυαδικοί τελεστές. Αξίζει να τονισθεί ότι ο τελεστής / ορίζει την πράξη της ακέραιας διαίρεσης ενώ ο τελεστής % ορίζει την πράξη modulo δηλαδή η έκφραση x%y επιστρέφει το υπόλοιπο της διαίρεσης της μεταβλητής x με τη μεταβλητή y.

Σε μια αριθμητική έκφραση οι μοναδιαίοι τελεστές έχουν τη μεγαλύτερη προτεραιότητα. Ακολουθούν οι τελεστές *,/,% με την ίδια προτεραιότητα ενώ τελευταίοι είναι οι τελεστές + και -. Σε περίπτωση που σε μια έκφραση εμπλέκονται περισσότεροι από έναν τελεστές με την ίδια προτεραιότητα τότε η εκτέλεση των πράξεων γίνεται από αριστερά προς τα δεξιά. Οι παραπάνω προτεραιότητες μπορούν να αλλάξουν με την εισαγωγή παρενθέσεων.

Ας δούμε μερικά παραδείγματα :

• 8%3*6. Δεδομένου ότι ο % και ο * έχουν την ίδια προτεραιότητα, ο υπολογισμός θα γίνει από αριστερά προς τα δεξιά, δηλαδή το αποτέλεσμα εκτέλεσης της έκφρασης είναι το 12.

• 3+4*6. Επειδή ο πολλαπλασιασμός έχει μεγαλύτερη προτεραιότητα από αυτόν της πρόσθεσης η έκφραση επιστρέφει 27.

• (3+4)*6. Με την εισαγωγή των παρενθέσεων αλλάζει η προτεραιότητα, οπότε πρώτα εκτελούνται οι πράξεις εντός των παρενθέσεων και μετά ο πολλαπλασιασμός δηλαδή η έκφραση αυτή επιστρέφει 42.

• w * x/y * z. Επειδή οι τελεστές * και / έχουν την ίδια προτεραιότητα, οι πράξεις θα εκτελεσθούν από αριστερά προς τα δεξιά, δηλαδή ((w* x)/y) * y.

• w*x/y+z/y. Η έκφραση αυτή αποτελείται από δύο υπο εκφράσεις. Η μια είναι αυτή που βρίσκεται αριστερά του τελεστή + που έχει την μικρότερη προτεραιότητα και η δεύτερη είναι αυτή που βρίσκεται στα δεξιά του. Επειδή και στις δύο υποεκφράσεις οι τελεστές έχουν την ίδια προτεραιότητα τότε η εκτέλεση των πράξεων θα γίνει από αριστερά προς τα δεξιά. Δηλαδή, η ισοδύναμη έκφραση είναι η ((w * x)/y) + (z/y).

Πρόγραμμα το οποίο κάνει πράξεις με αριθμούς.

Πιο συγκεκριμένα βρίσκει :

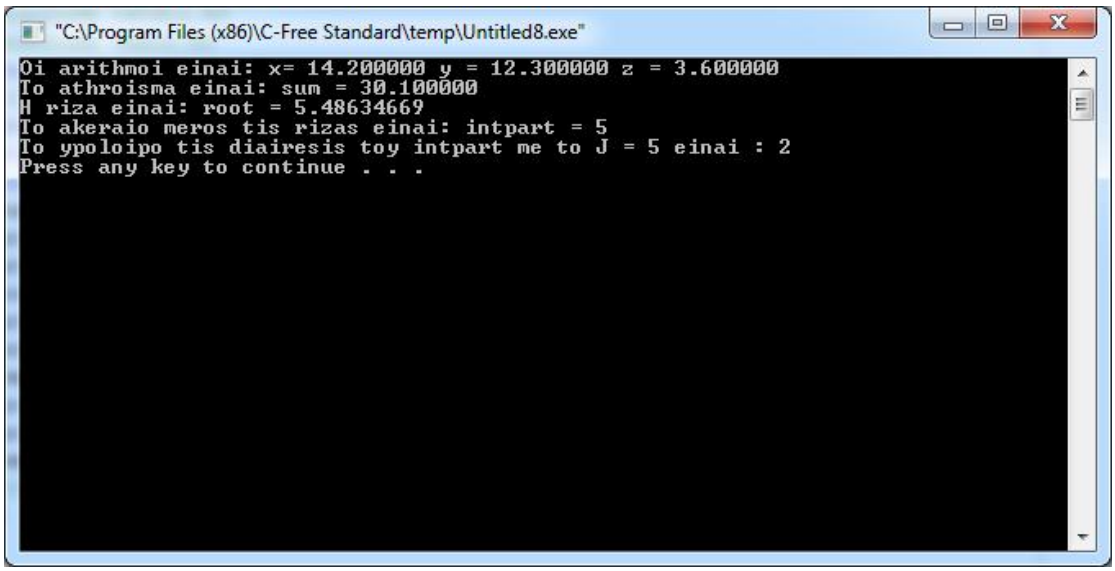
- Το άθροισμα sum τριών πραγματικών αριθμών x=14.2, y=12.3, z=3.6.
- την ρίζα root του αθροίσματος
- ακέραιο μέρος intpart του root.
- το υπόλοιπο remainder της διαίρεσης τουintpart με τον ακέραιο J=3

```
#include <stdio.h>

#include <math.h>

void main()

 {

double x = 14.2, y = 12.3, z = 3.6;

double sum, root;

int intpart, J=3, remainder;

sum = x + y + z;

printf("Oi arithmoi einai: x= %lf y = %lf z = %lf \n", x, y, z);

printf("To athroisma einai: sum = %lf\n", sum);

root=pow(sum, 0.5);

printf("H riza einai: root = %10.8lf\n", root);

intpart= (int) root;

remainder = intpart % J;

printf("To akeraio meros tis rizas einai: intpart = %d\n", intpart);

printf("To ypoloipo tis diairesis toy intpart me to J = %d einai : %d\n", intpart,

remainder);
```

//return ;

}

```c
#include <stdio.h>
#include <math.h>
void main()
 {
double x = 14.2, y = 12.3, z = 3.6;
double sum, root;
int intpart, J=3, remainder;
sum = x + y + z;
printf("Oi arithmoi einai: x= %lf y = %lf z = %lf \n", x, y, z);
printf("To athroisma einai: sum = %lf\n", sum);
root=pow(sum, 0.5);
printf("H riza einai: root = %10.8lf\n", root);
intpart= (int) root;
remainder = intpart % J;
printf("To akeraio meros tis rizas einai: intpart = %d\n", intpart);
printf("To ypoloipo tis diairesis toy intpart me to J = %d einai : %d\n", intpart,
remainder);
//return ;
}
```



3.2 Τελεστές Μετατόπισης

Η C διαθέτει δύο τελεστές μετατόπισης, , που μετατοπίζουν τα bit μιας ακέραιας μεταβλητής αριστερά ή δεξιά ορισμένο αριθμό θέσεων. Η έκφραση x n (x n) μετατοπίζει τα bit της μεταβλητής x, n θέσεις δεξιά (αριστερά) θέτοντας 0 στις n θέσεις αριστερά (δεξιά) και επιστρέφεται το αποτέλεσμα χωρίς να αλλάξει η μεταβλητή x. Η προτεραιότητα των τελεστών μετατόπισης είναι μικρότερη από αυτή των αριθμητικών τελεστών.

΄Εστω ότι x = $(12)_{10}$ = $(00001100)_2$ και n = 2 τότε η έκφραση x n (δεξιά μετατόπιση) μετατοπίζει την τιμή της μεταβλητής x δύο θέσεις δεξιά θέτοντας 0 στις δύο περισσότερο σημαντικές θέσεις και επιστρέφει (00000011) $_2$= $(3)_{10}$. Αν τώρα n = 3 τότε η έκφραση x n μετατοπίζει την τιμή της μεταβλητής x δύο θέσεις αριστερά θέτοντας 0 στις δύο λιγότερο σημαντικές θέσεις και επιστρέφει $(01100000)_2$ = $(96)_{10}$ ενώ η έκφραση x n επιστρέφει (00000001_2) = $(1)_{10}$. Να τονισθεί ότι η μεταβλητή x δεν αλλάζει τιμή, δηλαδή εξακολουθεί η τιμή της να είναι $(12)_{10}$ = $(00001100)_2$.

Κάτω από ειδικές συνθήκες, οι τελεστές μετατόπισης μπορούν να χρησιμοποιηθούν για τον πολλαπλασιασμό και την διαίρεση ενός ακεραίου με δυνάμεις του 2. Αν κατά την αριστερή μετατόπιση κατά n θέσεις, δεν χάνονται bit από το περισσότερο σημαντικό μέρος του αριθμού τότε ο ακέραιος έχει πολλαπλασιασθεί με το 2n . Η δεξιά μετατόπιση κατά n θέσεις είναι η ακέραια διαίρεση, όπου το κλασματικό μέρος αποκόπτεται. Για παράδειγμα μετατοπίζοντας το 7 μια θέση δεξιά προκύπτει το 3 που είναι το αποτέλεσμα της ακέραιας διαίρεσης με το 2.

Τι γίνεται όμως όταν οι ακέραιοι είναι προσημασμένοι, σε μια δεξιά και σε μια αριστερή μετατόπιση ; Υπάρχουν δύο κατηγορίες μετατοπίσεων :

• Η λογική μετατόπιση όπου οι αριθμοί αντιμετωπίζονται ως μη προσημασμένοι.

• Η αριθμητική μετατόπιση, όπου σε κάθε αριστερή μετατόπιση το bit προσήμου διατηρείται και σε κάθε δεξιά μετατόπιση στις περισσότερο σημαντικές θέσεις εισάγεται 0 (για θετικούς) ή 1 (για αρνητικούς) ανάλογα με την τιμή του bit προσήμου.

3.3 Τελεστές ανάθεσης

Οι τελεστές ανάθεσης (assignment operators) εκτελούν κάποια πράξη ανάμεσα στους τελεστέους και εκχωρούν το αποτέλεσμα σε έναν από τους τελεστέους:

• x* = 10; εκτελεί την πράξη του πολλαπλασιασμού μεταξύ των x και 10 και εκχωρεί το αποτέλεσμα στον τελεστέο x. Αντιστοιχεί στην πρόταση x = x * 10;

• x* = y + 1; Αντιστοιχεί στην πρόταση x = x * (y + 1); κι όχι στην πρόταση x = x * y + 1;

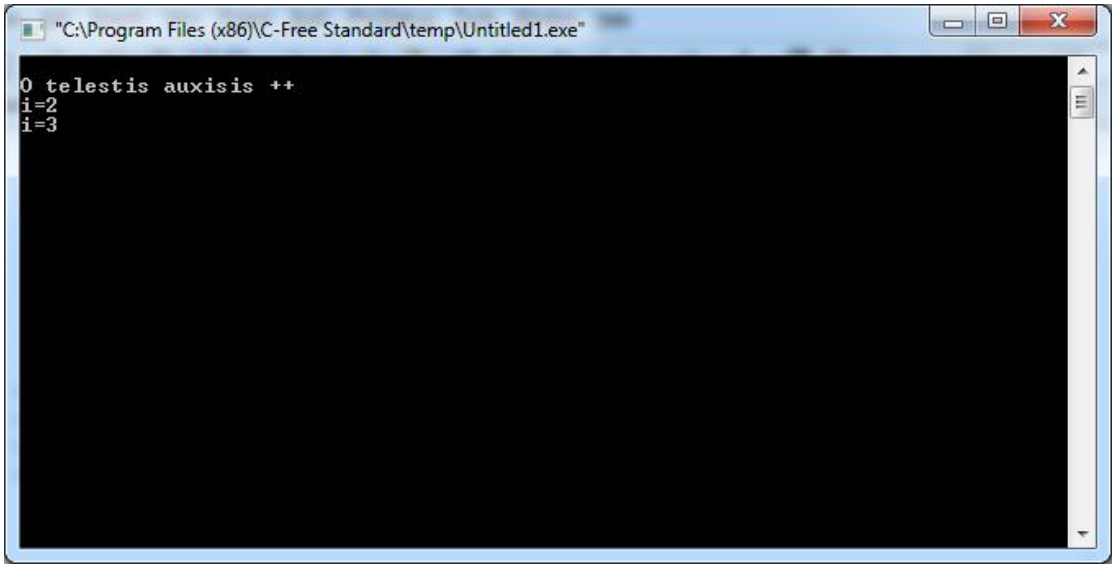
Τελεστές ανάθεσης δημιουργούν κι οι τελεστές διαχείρισης δυαδικών ψηφίων (bitwise operators). Οι τελεστές αυτοί είναι: >>= <<= &= ^= |= .

Οι τελεστές ανάθεσης μαζί με τους τελεστές αύξησης/μείωσης γίνονται αιτία δημιουργίας παρενεργειών (side effects), για το λόγο αυτό αναφέρονται και ως παρενεργοί τελεστές (side effect operators). Οι παρενέργειες αυτές έχουν ως αποτέλεσμα την απροσδιόριστη συμπεριφορά του συστήματος ως προς τον τρόπο υπολογισμού της τιμής της μεταβλητής i σε εκφράσεις όπως: i = n[i++]; ή i = ++i + 1;

Πρόγραμμα εκμάθησης του τελεστή αύξησης ++

```
#include <stdio.h>

main()

{

int i;

i = 0;

while (i++ == 0)

{

 printf("\nO telestis auxisis ++");

 printf("\ni=%d", ++i);

}

printf("\ni=%d", i);

scanf("%d", &i);

} /* end of main */
```

```
1  #include <stdio.h>
2  main()
3  {
4  int i;
5  i = 0;
6  while (i++ == 0)
7  {
8   printf("\nO telestis auxisis ++");
9   printf("\ni=%d", ++i);
10 }
11 printf("\ni=%d", i);
12 scanf("%d", &i);
13 } /* end of main */
```



3.4 Συσχετιστικοί τελεστές

Οι συσχετιστικοί τελεστές (relational operators) συγκρίνουν δύο τελεστέους. Οι βασικοί τελεστές της κατηγορίας αυτής παρατίθενται στον ακόλουθο πίνακα:

Τελεστής	Δράση
<	μικρότερο από
>	μεγαλύτερο από
<=	μικρότερο ή ίσον από
>=	μεγαλύτερο ή ίσον από
==	Ίσο
!=	!= διάφορο

Για κάθε ένα από αυτούς το αποτέλεσμα της έκφρασης που βρίσκεται στα αριστερά του τελεστή σύγκρισης συγκρίνεται με το αποτέλεσμα της έκφρασης που βρίσκεται στα δεξιά του και επιστρέφεται το αποτέλεσμα της σύγκρισης : 1 αν ικανοποιείται η σχέση και 0 αν δεν ικανοποιείται.

Οι τελεστές <, <=, >, >= έχουν μεγαλύτερη προτεραιότητα έναντι των ! = και ==. Για παράδειγμα η έκφραση x == y > z είναι ισοδύναμη με την x == (y > z). Επίσης, οι συσχετιστικοί τελεστές έχουν μικρότερη προτεραιότητα έναντι των αριθμητικών τελεστών και των τελεστών μετατόπισης. Για παράδειγμα, η έκφραση x + 6 > y είναι ισοδύναμη με την έκφραση (x + 6) > y.

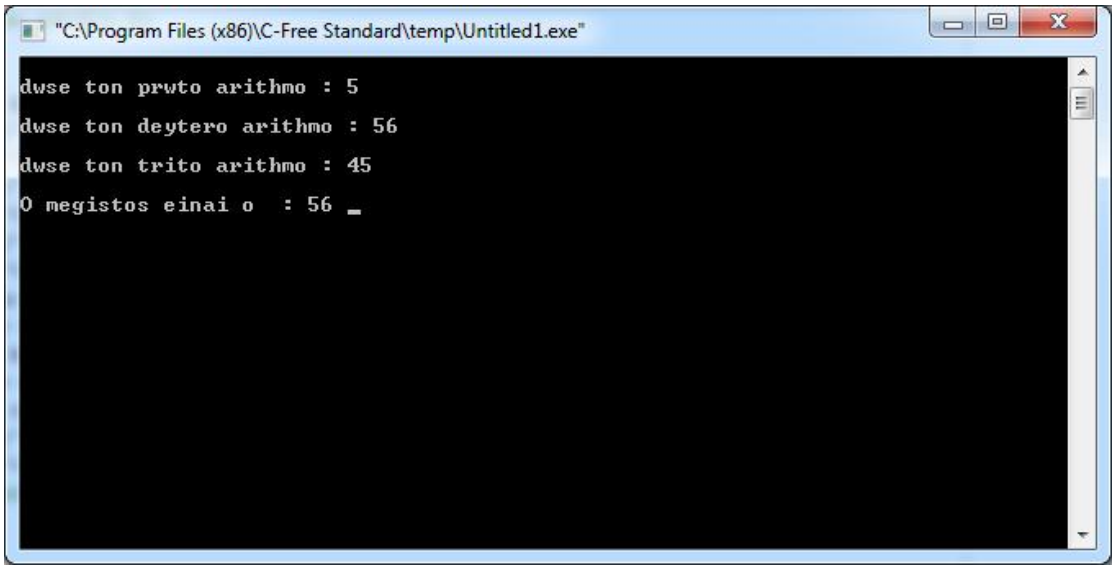
Επίπεδο Τελεστής	Επίπεδο Τελεστής
1	() [ ] → .
2	! ~ + + – – ∗(τελεστής έμμεσης αναφοράς) (cast) &(διεύθυνση αντικειμένου) sizeof +(μοναδιαίος) -(μοναδιαίος)
3	∗(πολλαπλασιασμός) / %
4	+ –
5	«»
6	> < >= <=
7	== ! =
8	& (Λογικό ΚΑΙ)
9	^
10	|
11	&&
12	k
13	? :
14	= + = – = ∗ = / = % = &=^= | = = =
15	,

Πρόγραμμα για εύρεση του μέγιστου αριθμού από τρεις ακέραιους αριθμούς με την χρήση των τελεστών.

```
#include <stdio.h>

main()

{

int a, b, c, max;

printf("\ndwse ton prwto arithmo : ");

scanf("%d", &a);

printf("\ndwse ton deytero arithmo : ");

scanf("%d", &b);

printf("\ndwse ton trito arithmo : ");

scanf("%d", &c);

max = a;

if (b > max)
```

```
 max = b;

if (c > max)

 max = c;

printf("\nO megistos einai o  : %d ", max);

scanf("%d", &a);

} /* end of main */
```

```
#include <stdio.h>
main()
{
int a, b, c, max;
printf("\ndwse ton prwto arithmo : ");
scanf("%d", &a);
printf("\ndwse ton deytero arithmo : ");
scanf("%d", &b);
printf("\ndwse ton trito arithmo : ");
scanf("%d", &c);
max = a;
if (b > max)
 max = b;
if (c > max)
 max = c;
printf("\nO megistos einai o  : %d ", max);
scanf("%d", &a);
} /* end of main */

```



3.5 Λογικοί τελεστές

Οι λογικοί τελεστές δρουν επί ενός ή δύο τελεστέων και λειτουργούν με βάση τη δίτιμη άλγεβρα Boole. Τόσο οι είσοδοι όσο και οι έξοδοι μπορούν να λάβουν μόνο δύο τιμές, TRUE και FALSE. Οι τελεστές της κατηγορίας αυτής παρατίθενται στον παρακάτω πίνακα.

Τελεστής	Δράση
&&	Λογικό AND
||	Λογικό OR
!	Λογικό NOT

Ενώ παρακάτω περιγράφεται ο τρόπος λειτουργίας τους (πίνακας αληθείας):

Σ1	Σ2	Σ1&&Σ2	Σ1||Σ2	!Σ1
Α	Α	Α	Α	Ψ
Α	Ψ	Ψ	Α	
Ψ	Α	Ψ	Α	Α
Ψ	Ψ	Ψ	Ψ	

4. Εντολές Ελέγχου Ροής Προγράμματος

Τα προγράμματα αποτελούνται από προτάσεις, οι οποίες εκτελούνται με κάποια σειρά. Ο πιο συνηθισμένος τρόπος εκτέλεσης είναι ο ακολουθιακός: δύο ή περισσότερες προτάσεις βρίσκονται διατεταγμένες η μία μετά την άλλη και εκτελούνται διαδοχικά. Ωστόσο ορισμένες φορές επιβάλλεται να γίνουν λογικές επιλογές (με χρήση λογικών τελεστών και τελεστών συσχέτισης).

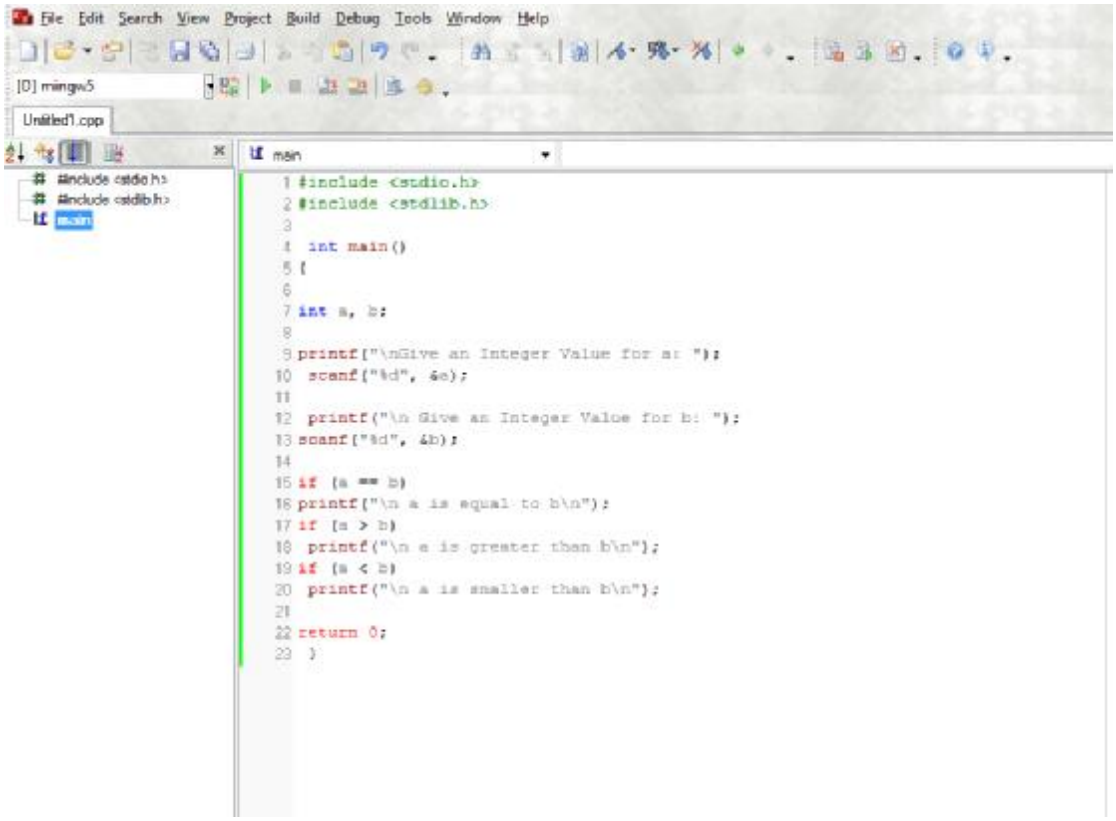
4.1 Η Εντολή Απόφασης If

Όταν μια συνθήκη ικανοποιείται τότε εκτελείται μια έκφραση διαφορετικά εκτελείται μια άλλη έκφραση. Στη C μια εντολή ή ένα σύνολο εντολών μπορεί να εκτελεστεί υπό συνθήκη με τη χρήση της εντολής απόφασης if η σύνταξη της οποίας έχει ως εξής :

if (συνθήκη) εντολή;
if ¯(συνθήκη) σύνθετη-εντολή

όπου στη θέση της συνθήκης μπορεί να είναι μια λογική έκφραση ή μια οποιαδήποτε έκφραση, καθώς κάθε έκφραση επιστρέφει μια τιμή η οποία αν είναι 0 εκλαμβάνεται από την if ως ψευδής διαφορετικά εκλαμβάνεται ως αληθής. ΄Οπως φαίνεται από την παραπάνω σύνταξη μια απλή εντολή ή μια σύνθετη εντολή μπορεί να εκτελεσθεί μόνο αν είναι αληθής η συνθήκη. Μετά την εκτέλεση της εντολής απόφασης, η ροή του προγράμματος συνεχίζεται με την επόμενη της εντολής if. Με την σύνταξη της εντολής απόφασης που ακολουθεί είναι δυνατή η προσθήκη εναλλακτικής εντολής για την περίπτωση που δεν ικανοποιείται η συνθήκη.

Με την σύνταξη της εντολής απόφασης που ακολουθεί είναι δυνατή η προσθήκη εναλλακτικής εντολής για την περίπτωση που δεν ικανοποιείται η συνθήκη.
if (συνθήκη) εντολή-1 ;
else εντολή-2 ;
if (συνθήκη) σύνθετη-εντολή-1
else σύνθετη-εντολή-2
Κατά συνέπεια, όταν η συνθήκη είναι αληθής εκτελείται η εντολή-1/σύνθετη-εντολή-1 διαφορετικά εκτελείται η εντολή-2/σύνθετη-εντολή-2. Και πάλι με την ολοκλήρωση της εκτέλεσης της εντολής απόφασης if else η ροή του προγράμματος συνεχίζεται με την επόμενη της εντολής απόφασης.



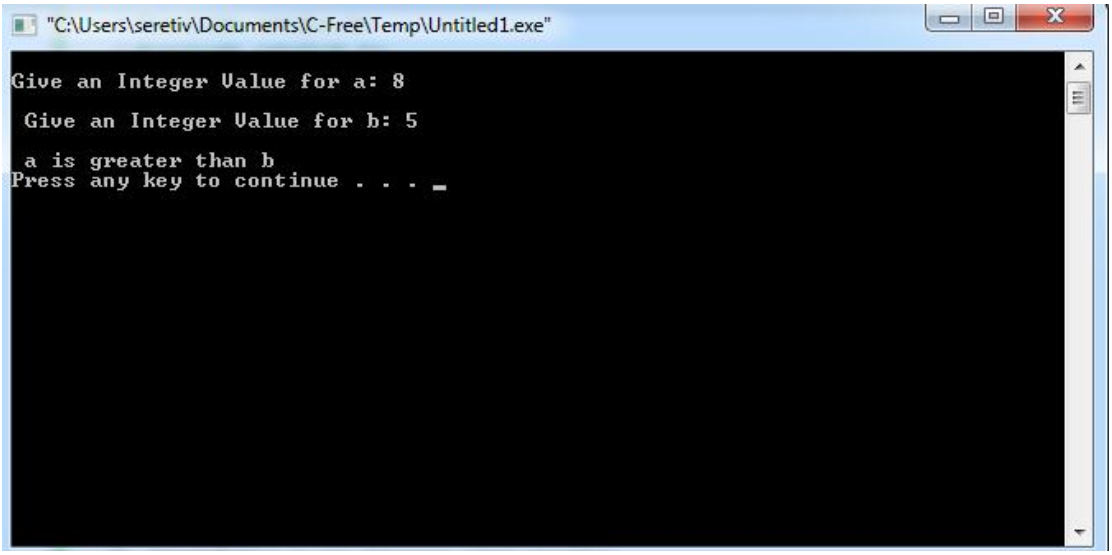
- Αρχικά ζητείται από τον χρήστη μέσω των εντολών των γραμμών 9 και 10 να δώσει έναν ακέραιο ο οποίος αποθηκεύεται στη μεταβλητή a.

- Στη συνέχεια, μέσω των εντολών των γραμμών 12 και 13 ζητείται άλλος ένας ακέραιος, ο οποίος καταχωρείται στη μεταβλητή b.

- Η εκτέλεση του προγράμματος συνεχίζεται με την εντολή απόφασης της γραμμής 15.

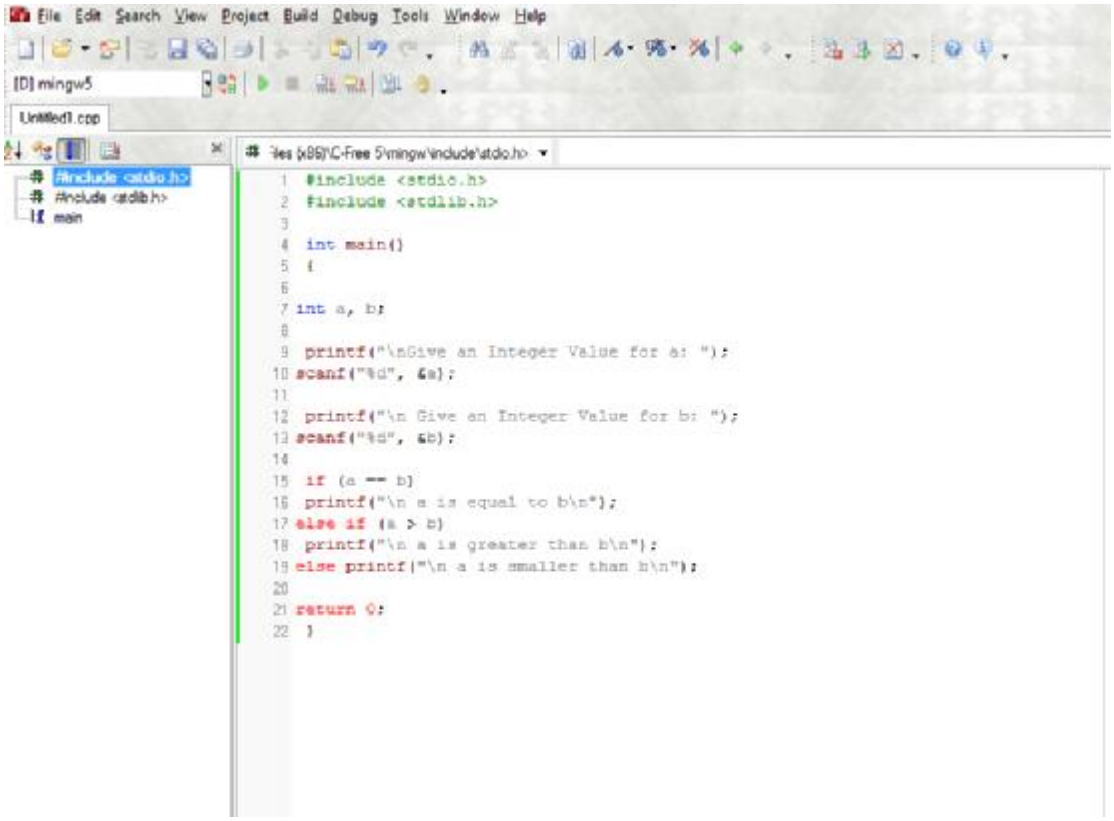
- Αν η τιμή της μεταβλητής a είναι ίση με αυτή της b τυπώνεται το μήνυμα που αφορά την ισότητα.

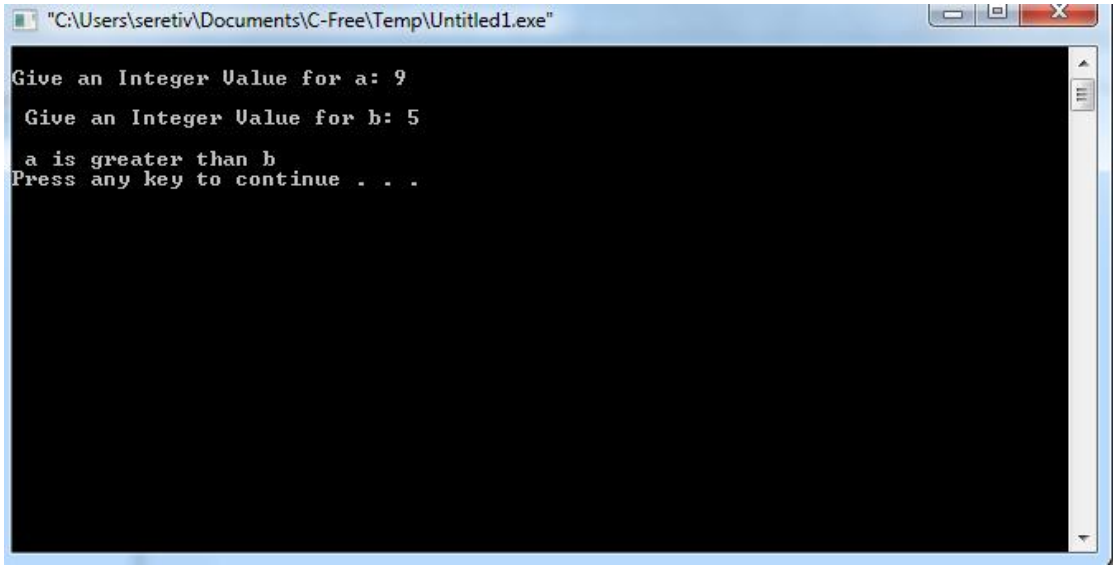
- Στη συνέχεια, εκτελείται η εντολή απόφασης 16 στην οποία ελέγχεται αν η μεταβλητή a είναι μμεγαλύτερη από την b.
- Ενώ το πρόγραμμα ολοκληρώνεται με την εκτέλεση της εντολής απόφασης της γραμμής 19, όπου τώρα ελέγχεται αν η μεταβλητή a είναι μικρότερη από τη b.



Στο παραπάνω παράδειγμα γίνονται πάντα τρεις έλεγχοι ακόμα και όταν η σχέση των μεταβλητών έχει προσδιοριστεί με τον πρώτο έλεγχο. Για να γραφεί ένας πιο αποδοτικός κώδικας μπορούμε να αξιοποιήσουμε το γεγονός ότι η σύνθετη εντολή σε μια εντολή απόφασης μπορεί να είναι επίσης μια εντολή απόφασης. Η σύνταξη της εντολής απόφασης στην περίπτωση αυτή είναι η εξής :

if (συνθήκη-1) σύνθετη-εντολή-1
else if (συνθήκη-2) σύνθετη-εντολή-2
else σύνθετη-εντολή-3
επόμενη-εντολή;


File Edit Search View Project Build Debug Tools Window Help
[D] mingw5
Untitled1.cpp
#include <stdio.h>
#include <stdlib.h>
main
#include <stdio.h>
#include <stdlib.h>
int main()
{
int a, b;
printf("\nGive an Integer Value for a: ");
scanf("%d", &a);
printf("\n Give an Integer Value for b: ");
scanf("%d", &b);
if (a == b)
printf("\n a is equal to b\n");
else if (a > b)
printf("\n a is greater than b\n");
else printf("\n a is smaller than b\n");
return 0;
}

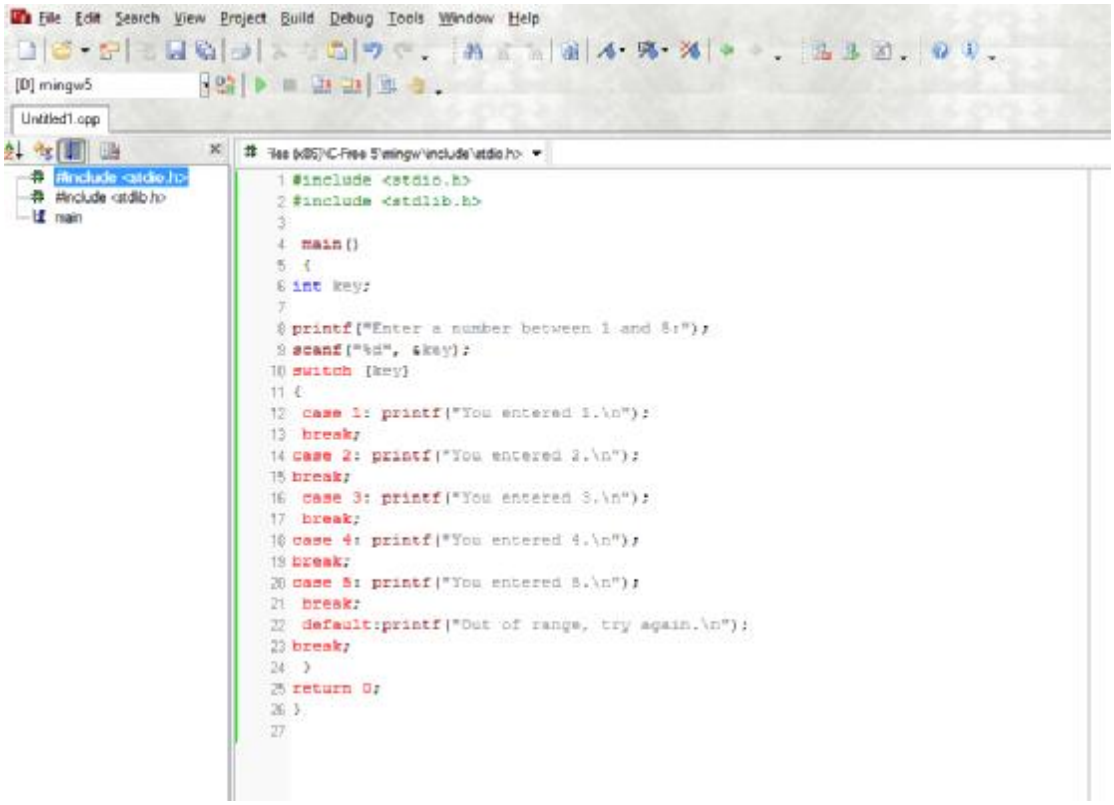

"C:\Users\seretiv\Documents\C-Free\Temp\Untitled1.exe"
Give an Integer Value for a: 9
Give an Integer Value for b: 5
a is greater than b
Press any key to continue . . .

4.2 **Η Εντολή Επιλογής** switch

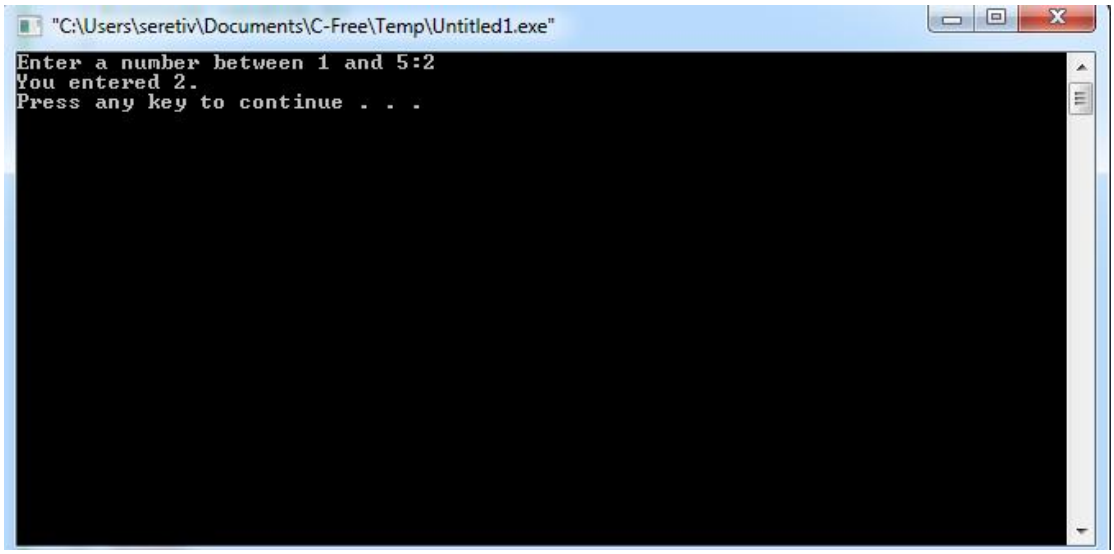
Όπως η **if** έτσι και η **switch** μας επιτρέπει την διακλάδωση του προγράμματος σε διαφορετικά **block** κώδικα ανάλογα με την περίπτωση που ισχύει. Επειδή όμως οι περιπτώσεις της **switch** είναι σταθερές και όχι λογικές εκφράσεις, μπορεί να ισχύει μόνο μια περίπτωση, και άρα να εκτελεστεί μόνο ένα τμήμα κώδικα. Η **switch** συντάσσεται ως εξής:

```
switch (μεταβλητή){
        case σταθερά1:  εντολή_Α;
                        break;
        case σταθερά2:  εντολή_Β;
                        break;
        case σταθερά3:  εντολή_Γ;
                        break;
        default:        εντολή_Δ;
}
```

Υπολογίζεται αρχικά η έκφραση, και στη συνέχεια αναζητείται η **case** με ετικέτα ισότιμη με την τιμή που υπολογίσθηκε. Η εκτέλεση συνεχίζεται από την **case** που βρέθηκε στο προηγούμενο βήμα μέχρι τέλος της **switch**. Η **default** επιλογή, εκτελείται όταν καμία από τις **case** δεν ταυτίζεται με την έκφραση. Μπορεί να παραληφθεί, όμως είναι καλή τακτική να εισάγεται.



Το παραπάνω πρόγραμμα παίρνει από τον χρήστη έναν αριθμό από το 1 ως το 5 και τυπώνει στην οθόνη τον αριθμό που έδωσε ο χρήστης.



5. ΠΡΟΤΑΣΕΙΣ ΕΠΑΝΑΛΗΨΗΣ - ΒΡΟΧΟΙ

Οι προτάσεις επανάληψης αποτελούν ένα ισχυρό εργαλείο προγραμματισμού καθώς μπορούν να κωδικοποιήσουν και να συμπυκνώσουν επαναλαμβανόμενες λειτουργίες, δημιουργώντας ένα βρόχο (loop). Ορίζονται ως προτάσεις που επαναλαμβάνουν ένα μπλοκ εντολών είτε για όσες φορές το επιθυμούμε είτε έως ότου πληρωθεί μία συνθήκη τερματισμού. Η πλήρωση του κριτηρίου τερματισμού οδηγεί στην περάτωση του βρόχου.

Εάν δεν υπάρχει συγκεκριμένος αριθμός επαναλήψεων ή συνθήκη τερματισμού, ο βρόχος θα εκτελείται αενάως και θα καλείται ατέρμων βρόχος (infinite loop), γεγονός που οδηγεί ως επί το πλείστον σε σφάλμα.

Εάν ο βρόχος τελειώνει μετά το πέρας ενός ορισμένου αριθμού επαναλήψεων τότε καλείται βρόχος οδηγούμενος από μετρητή. Εάν περατώνεται με την πλήρωση ενός κριτηρίου τερματισμού ονομάζεται βρόχος οδηγούμενος από γεγονός. Επιπρόσθετα, στις περισσότερες γλώσσες προγραμματισμού υπάρχει μία δεύτερη κατηγοριοποίηση των προτάσεων επανάληψης: α) εκείνες στις οποίες ο έλεγχος του κριτηρίου τερματισμού γίνεται στην αρχή του βρόχου, επονομαζόμενες βρόχοι με συνθήκη εισόδου (pre-test loops), και β) εκείνες στις οποίες ο έλεγχος του κριτηρίου τερματισμού γίνεται στο τέλος του βρόχου, επονομαζόμενες βρόχοι με συνθήκη εξόδου (post-test loops).

5.1 Η εντολή while

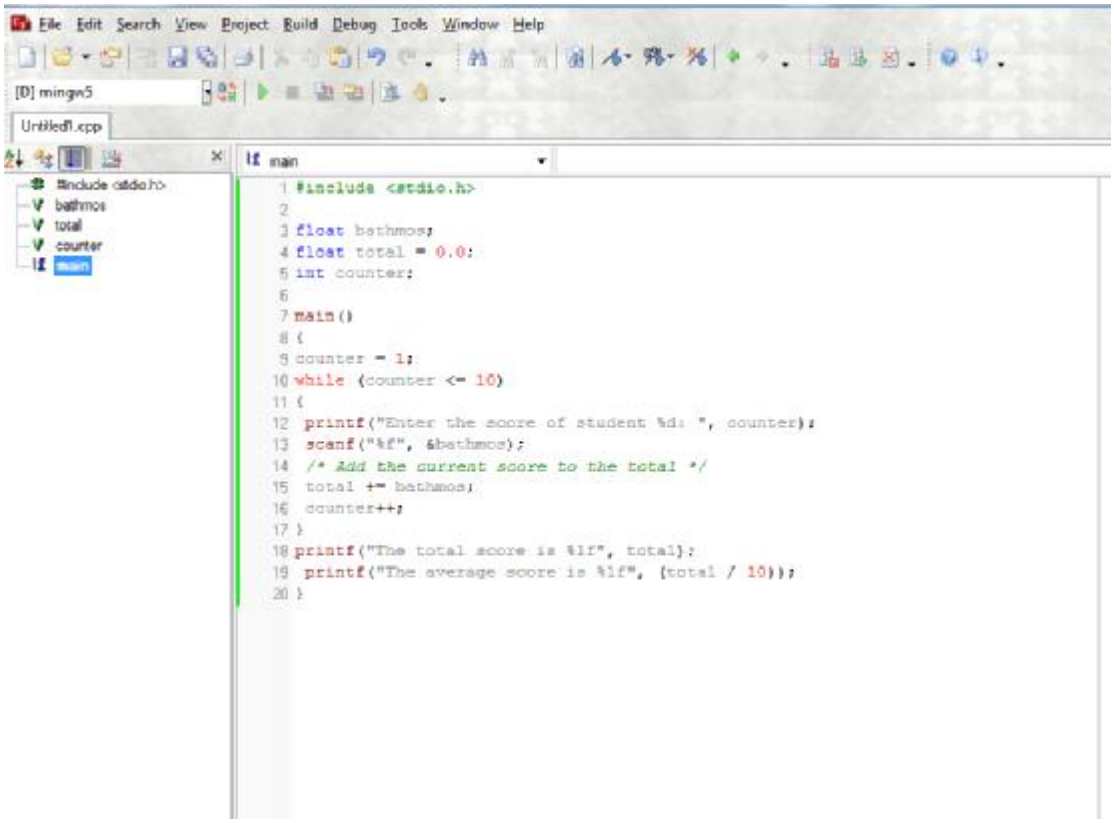
Η while χρησιμοποιείτε όταν δεν γνωρίζουμε τον αριθμό των επαναλήψεων ή υπάρχει περίπτωση οι εντολές να μην εκτελεστούν ούτε μία φορά. Εκτελούνται όσο η συνθήκη (expression) είναι αληθής (true).

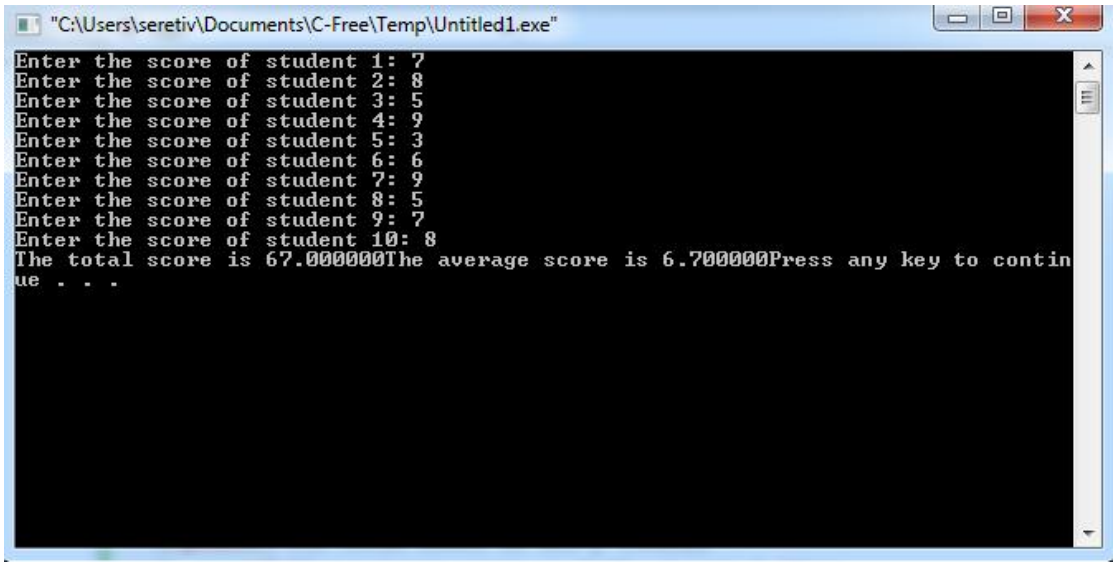
Η γενική σύνταξη είναι η εξής:

while (έκφραση) εντολή;

while (έκφραση) σύνθετη-εντολή

Πρόγραμμα το οποίο ζητάει επαναληπτικά τους βαθμούς 10 σπουδαστών και υπολογίζει το άθροισμα των βαθμών (total) και τον μέσο όρο αυτών:





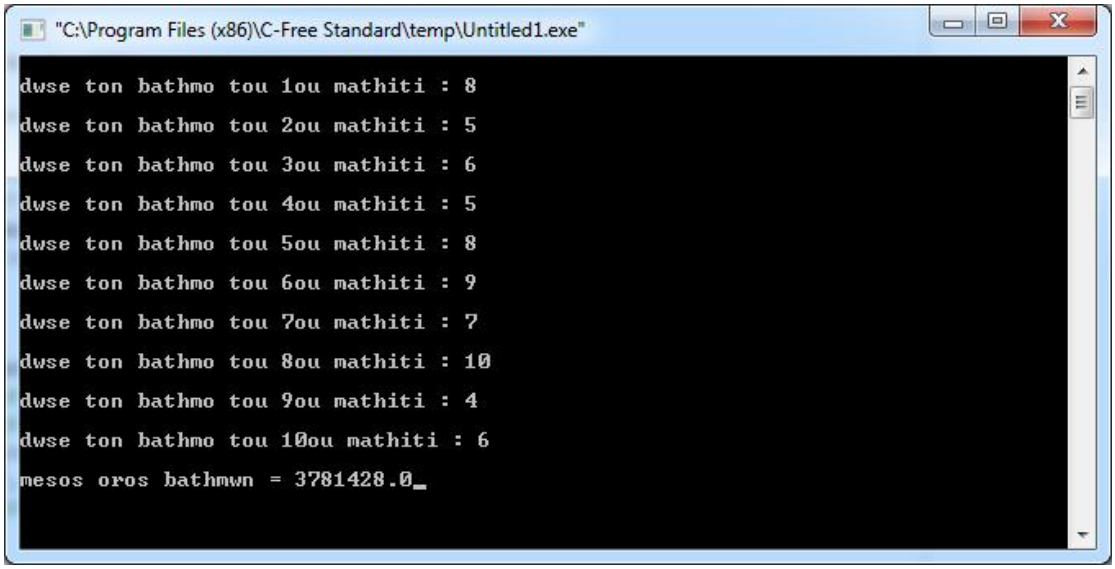
Πρόγραμμα εκμάθησης του βρόχου while

Διαβάζει τους βαθμούς 10 μαθητών και αν ο βαθμός είναι μικρότερος του 0 ή μεγαλύτερος του 20, δεν θα λαμβάνεται υπόψη στο μέτρημα .

```
#include <stdio.h>

main()

{
```

```
int bathmos, i, sum;
i = 1;
while (i <= 10)
{
 printf("\ndwse ton bathmo tou %dou mathiti : ", i);
 scanf("%d", &bathmos);
 if (bathmos < 0 || bathmos > 20)
 printf("\nmh apodektos bathmos");
 else
 {
 i++;
 sum += bathmos; /* athroisma twn bathmwn */
} /* end of else */
} /* end of while */
printf("\nmesos oros bathmwn = %.1f", sum/10.0);
scanf("%d", &bathmos);
} /* end of main */
```

```c
#include <stdio.h>
main()
{
int bathmos, i, sum;
i = 1;
while (i <= 10)
{
 printf("\ndwse ton bathmo tou %dou mathiti : ", i);
 scanf("%d", &bathmos);
 if (bathmos < 0 || bathmos > 20)
 printf("\nmh apodektos bathmos");
 else
 {
 i++;
 sum += bathmos; /* athroisma twn bathmwn */
} /* end of else */
} /* end of while */
printf("\nmesos oros bathmwn = %.1f", sum/10.0);
scanf("%d", &bathmos);
} /* end of main */
```



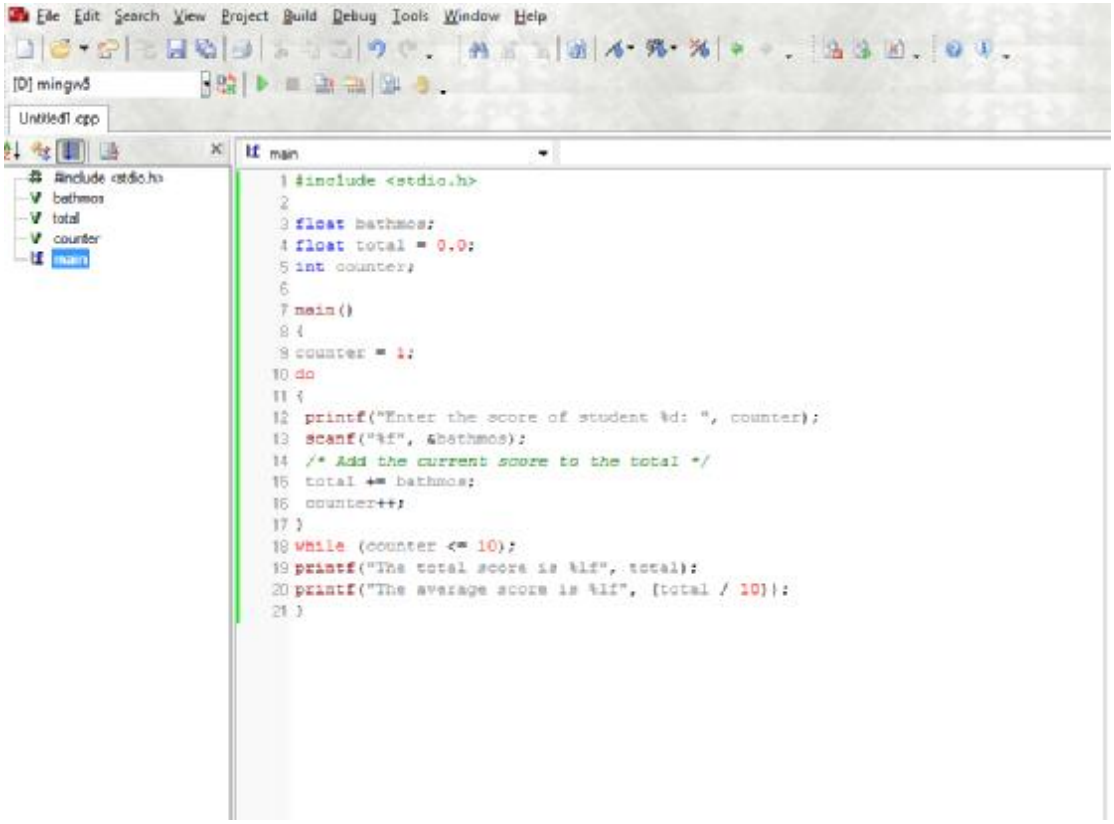
5.2 Η εντολή do-while

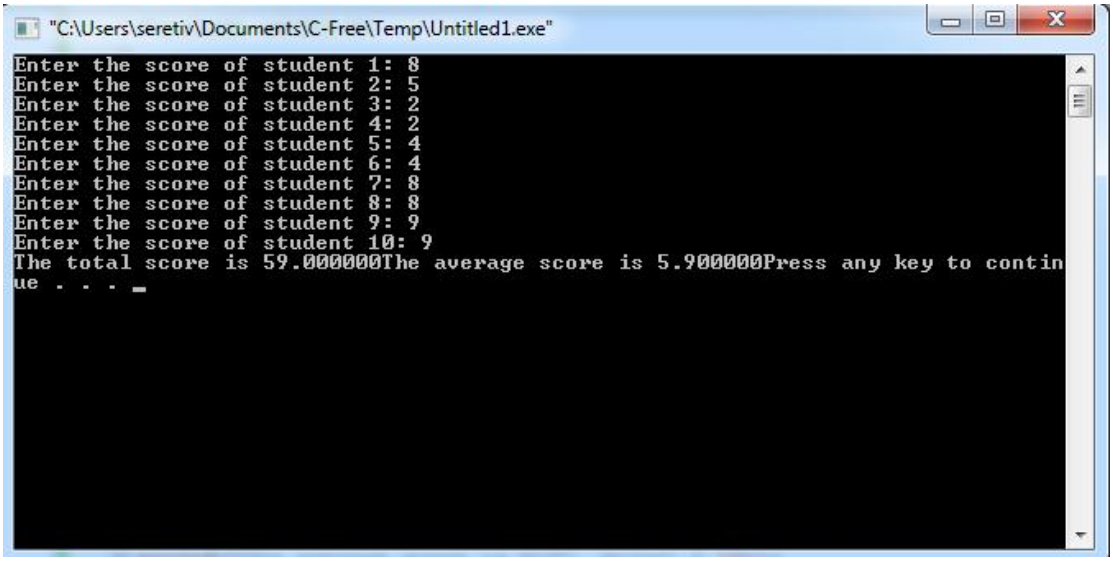
Η εντολή do..while, χρησιμοποιείτε όταν επίσης δεν γνωρίζουμε τον αριθμό των επαναλήψεων, αλλά θέλουμε το μπλοκ εντολών μας να εκτελεστεί τουλάχιστον μία φορά. Έτσι ο έλεγχος λαμβάνει χώρα αφού εκτελεσθεί η εντολή του βρόχου μια φορά, δηλαδή στο τέλος. Φυσικά ακόμα και αν γνωρίζουμε τον αριθμό των επαναλήψεων, και πάλι μπορούμε να προσαρμόσουμε τον κώδικά μας. Η γενική σύνταξη της είναι η εξής:

do εντολή; while έκφραση;

do σύνθετη-εντολή while έκφραση;

Πρόγραμμα το οποίο ζητάει επαναληπτικά τους βαθμούς 10 σπουδαστών και υπολογίζει το άθροισμα των βαθμών (total) και τον μέσο όρο αυτών:

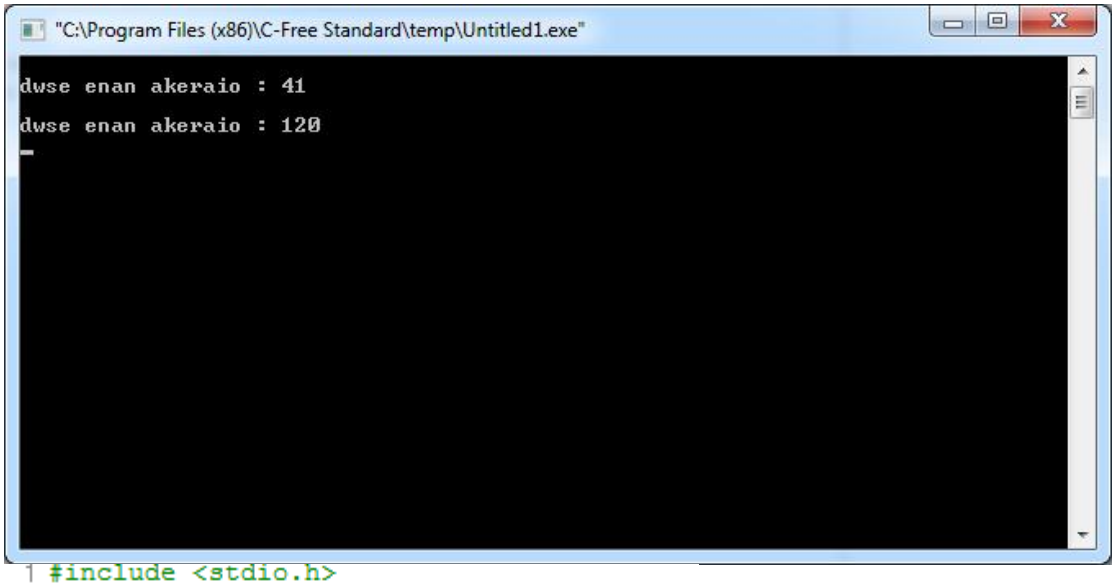




Πρόγραμμα εκμάθησης του βρόχου do while

ανάγνωση ακεραίων με χρήση του βρόχου do while μέχρι που να συναντήσει κάποιον που να είναι μεγαλύτερος από 100 και μικρότερος από 200

```
#include <stdio.h>

main()

{

int i;

do

{

 printf("\ndwse enan akeraio : ");

 scanf("%d", &i);

} while (i < 100 || i > 200);

scanf("%d", &i);

} /* end of main */
```




```c
#include <stdio.h>
main()
{
int i;
do
{
 printf("\ndwse enan akeraio : ");
 scanf("%d", &i);
} while (i < 100 || i > 200);
scanf("%d", &i);
} /* end of main */
```

5.3 Η εντολή for

Η for συνήθως χρησιμοποιείτε για έναν συγκεκριμένο αριθμό επαναλήψεων. Ωστόσο είναι δυνατό να αντικαταστήσει και τις δύο προηγούμενες με συγκεκριμένα expressions και εντολές (statements).

Η γενική σύνταξη είναι η ακόλουθη:

for (παράσταση_1; παράσταση_2; παράσταση_3){

γραμμή_1;

γραμμή_2;

.

.

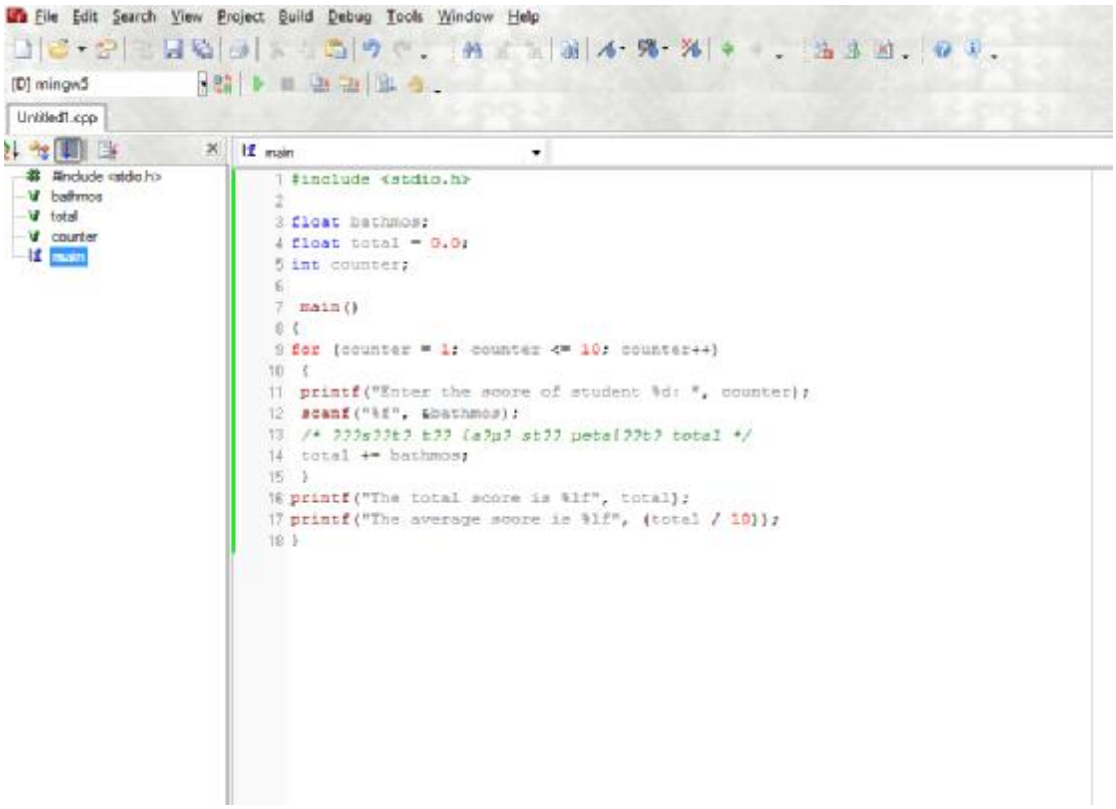
}

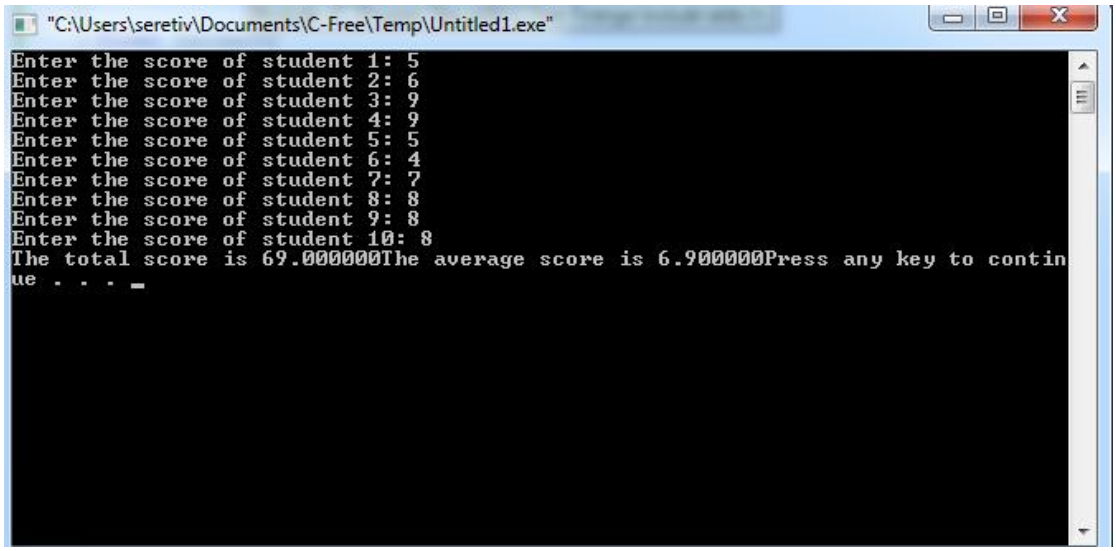
Το κυρίως σώμα της for αποτελείται από ένα σύνολο γραμμών (γραμμή_1, γραμμή_2 κτλ.) οι οποίες εκτελούνται ανάλογα με την λογική που προσδιορίζουν οι παραστάσεις:
παράσταση_1, παράσταση_2 και παράσταση_3. Ποιο συγκεκριμένα η ακριβής εκτέλεση της παραπάνω εντολής έχει ως ακολούθως:

- Πρώτα εκτελείται η παράσταση_1.
- Στη συνέχεια ελέγχεται εάν η παράσταση_2 είναι αληθής ή ψευδής. Εάν είναι αληθής εκτελούνται οι γραμμές που αποτελούν το κυρίως σώμα της for. Εάν είναι ψευδής τερματίζεται εδώ η εντολή for.
- Στη συνέχεια εκτελείται η παράσταση_3.
- Ελέγχεται πάλι εάν η παράσταση_2 είναι αληθής ή ψευδής. Εάν είναι αληθής εκτελούνται οι γραμμές που αποτελούν το κυρίως σώμα της for. Εάν είναι ψευδής τερματίζεται εδώ η εντολή for.
- Εκτελείται πάλι η παράσταση_3.
- Ελέγχεται πάλι εάν η παράσταση_2 είναι αληθής ή ψευδής. Εάν είναι αληθής εκτελούνται οι γραμμές που αποτελούν το κυρίως σώμα της for. Εάν είναι ψευδής τερματίζεται εδώ η εντολή for.
- Εκτελείται πάλι η παράσταση_3 και ελέγχεται εάν η παράσταση_2 είναι αληθής ή ψευδής. Αυτό συνεχίζεται έως ότου κάποια στιγμή η παράσταση_2 γίνει ψευδής οπότε και τερματίζεται η for.

Με απλά λόγια η εντολή for αποτελεί ένα βρόγχο ο οποίος εκτελείται n φορές (όπου n ακέραιος αριθμός). Ο αριθμός n εξαρτάται από την λογική που προσδιορίζουν οι παραστάσεις: παράσταση_1, παράσταση_2 και παράσταση_3.

Πρόγραμμα το οποίο ζητάει επαναληπτικά τους βαθμούς 10 σπουδαστών και υπολογίζει το άθροισμα των βαθμών (total) και τον μέσο όρο αυτών:





Πρόγραμμα που θα πραγματοποιεί εύρεση μεγαλύτερου και μικρότερου βαθμού 10 μαθητών και ποιοι μαθητές έχουν τους βαθμούς αυτούς

```
#include <stdio.h>

main()

{

int a[10], i, max, min, i_max, i_min;

/* kataxwrhsh vathmwn */

for (i=0; i<10; i++)

 do

{

 printf("\ndwse ton  %do bathmo : ", i);

 scanf("%d", &a[i]);

 } while (a[i] < 0 || a[i] > 20);

max = a[0];

min = a[0];

 i_max = 0;

 i_min = 0;

for (i=1; i<10; i++)

{

 if (a[i] > max)

 {

 max = a[i];

 i_max = i;

 }

 if (a[i] < min)

 {

 min = a[i];

 i_min = i;
```

}

} /* end of for */

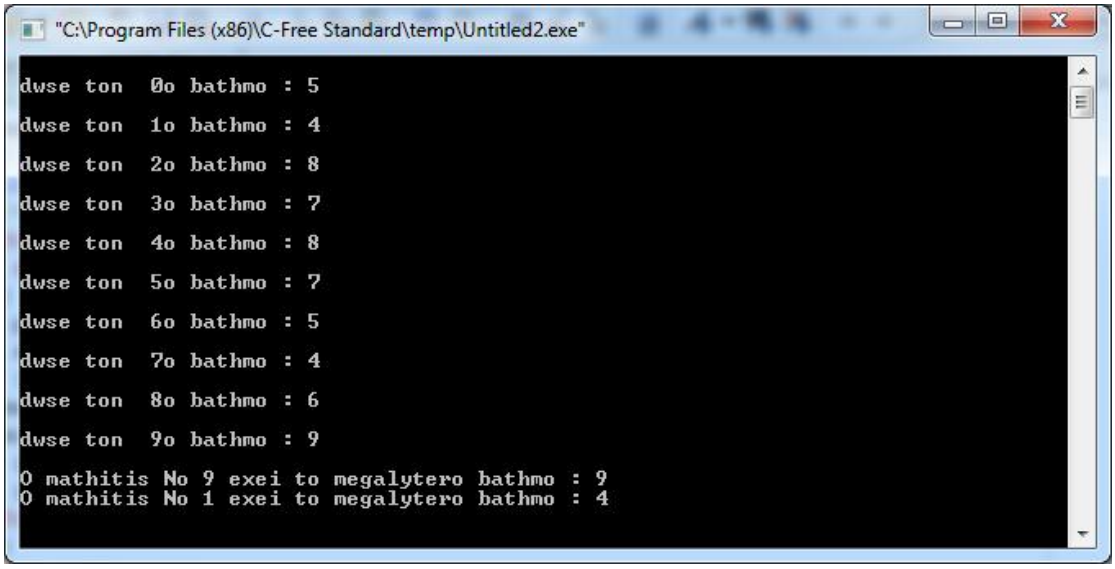
printf("\nO mathitis No %d exei to megalytero bathmo : %d", i_max, max);

printf("\nO mathitis No %d exei to megalytero bathmo : %d", i_min, min);

scanf("%d", &i);

} /* end of main */

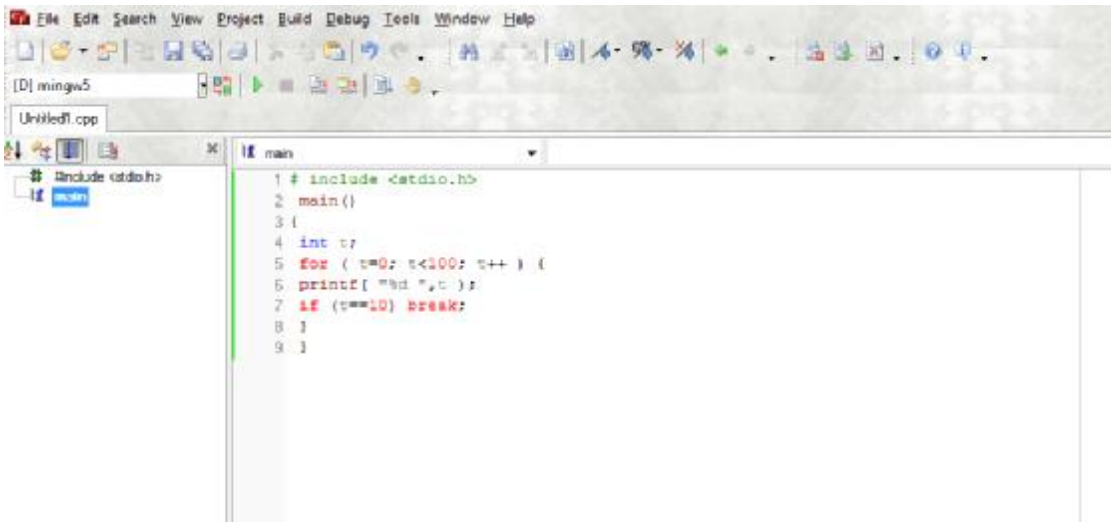
```c
#include <stdio.h>
main()
{
int a[10], i, max, min, i_max, i_min;
/* kataxwrhsh vathmwn */
for (i=0; i<10; i++)
 do
{
 printf("\ndwse ton  %do bathmo : ", i);
 scanf("%d", &a[i]);
 } while (a[i] < 0 || a[i] > 20);
max = a[0];
min = a[0];
 i_max = 0;
 i_min = 0;
for (i=1; i<10; i++)
{
 if (a[i] > max)
 {
 max = a[i];
 i_max = i;
 }
 if (a[i] < min)
 {
 min = a[i];
 i_min = i;
 }
} /* end of for */
printf("\nO mathitis No %d exei to megalytero bathmo : %d", i_max, max);
printf("\nO mathitis No %d exei to megalytero bathmo : %d", i_min, min);
scanf("%d", &i);
} /* end of main */

```

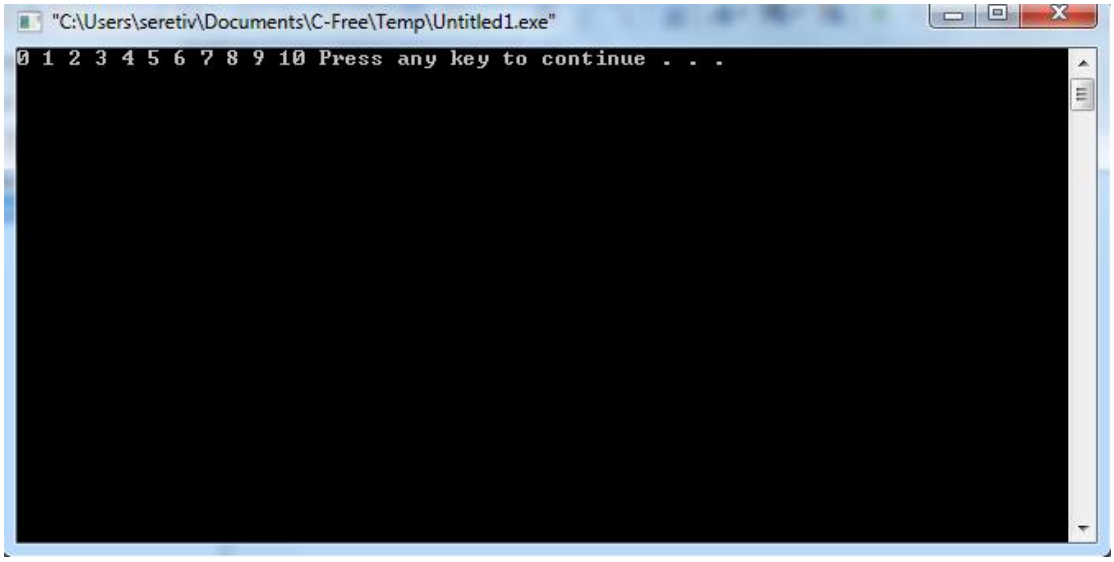


5.4 Οι εντολές break και continue

Η εντολή break χρησιμοποιείται όταν θέλουμε να σταματήσουμε την εκτέλεση ενός βρόγχου και να βγούμε από αυτόν. Χρησιμοποιείται για να τερματίσει αμέσως την εκτέλεση ενός βρόχου, μεταβιβάζοντας τον έλεγχο του προγράμματος στην εντολή που βρίσκεται αμέσως μετά το βρόχο.

Το πρόγραμμα που ακολουθεί εμφανίζει στην οθόνη τους αριθμούς 1 έως 10 και στη συνέχεια τερματίζεται γιατί η break υπερφαλαγγίζει τη συνθήκη ελέγχου του βρόχου t<100.





Η εντολή continue χρησιμοποιείται όταν θέλουμε να επιστρέψουμε στην κορυφή του βρόγχου αγνοώντας τις υπόλοιπες γραμμές του σώματός του. Το παρακάτω παράδειγμα κάνει κατανοητή την χρήση του continue.

Στους βρόχους while και do–while η εντολή continue υποχρεώνει τον έλεγχο του προγράμματος να περάσει κατευθείαν στη συνθήκη ελέγχου και να προχωρήσει κατόπιν στην επεξεργασία του βρόχου. Στην περίπτωση της for ο υπολογιστής εκτελεί πρώτα το τμήμα του βρόχου και κατόπιν τη συνθήκη ελέγχου, προτού συνεχισθεί η εκτέλεση του βρόχου.

Πρόγραμμα εκμάθησης των εντολών break και continue

```
#include <stdio.h>

main()

{

char ch;

while (1)

{

 printf("\ndwse enan xaraktira : ");

 ch = getchar();

putchar(ch);

 if (ch == '#')

 break; /* exodos apo ton vroxo */

 if (ch < 'A' || ch > 'Z')
```

continue; /* phgainei sthn korifi tou vroxou */

else

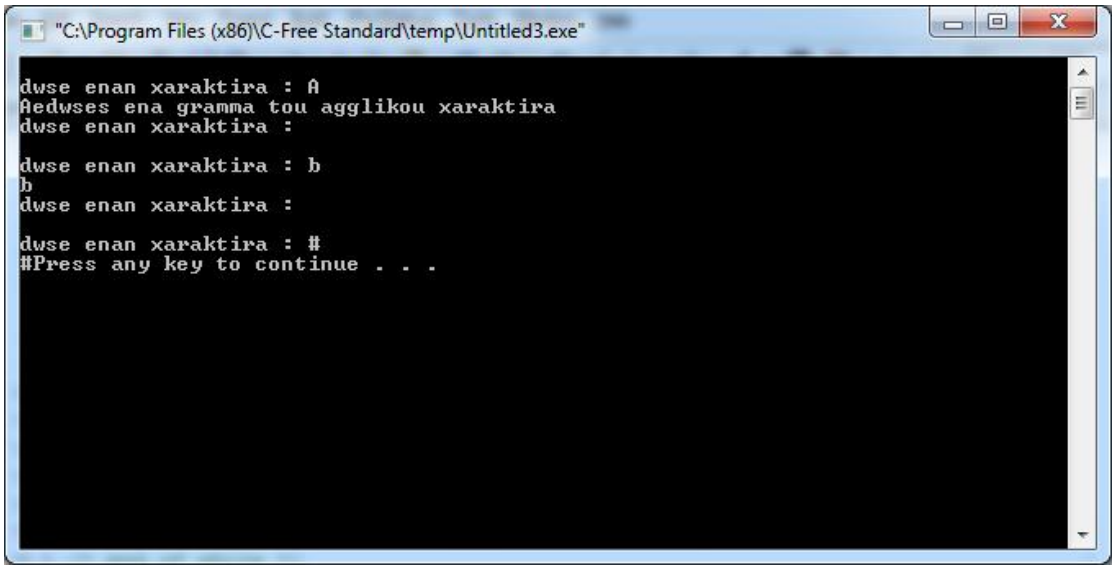
printf("edwses ena gramma tou agglikou xaraktira");

} /* end of while */

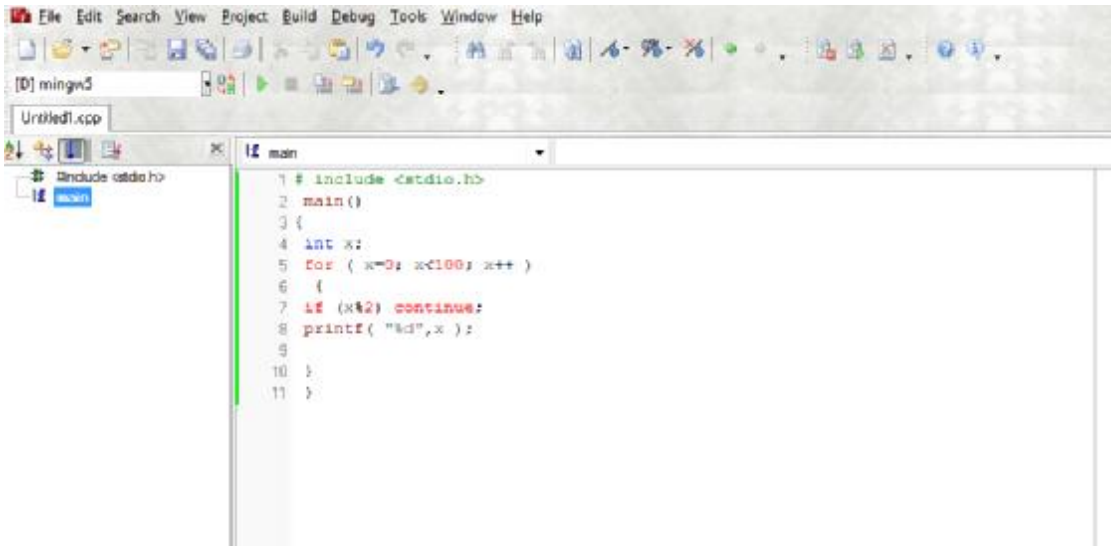
scanf("%c", &ch); }

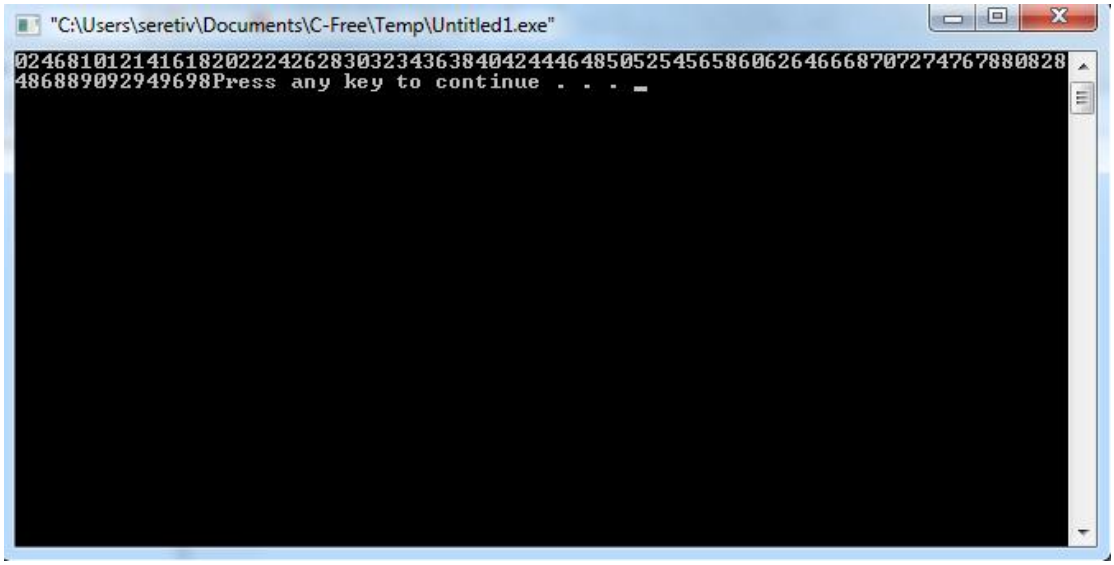
/* end of main */

```c
#include <stdio.h>
main()
{
char ch;
while (1)
{
 printf("\ndwse enan xaraktira : ");
 ch = getchar();
putchar(ch);
 if (ch == '#')
 break; /* exodos apo ton vroxo */
 if (ch < 'A' || ch > 'Z')
 continue; /* phgainei sthn korifi tou vroxou */
 else
 printf("edwses ena gramma tou agglikou xaraktira");
} /* end of while */
scanf("%c", &ch);
} /* end of main */

```



Το πρόγραμμα που ακολουθεί εμφανίζει στην οθόνη μόνο τους άρτιους αριθμούς.





6. Πίνακες

6.1 Μονοδιάστατοι πίνακες

Ο πίνακας (array) είναι μια συλλογή τιμών δεδομένων η οποία έχει τα εξής χαρακτηριστικά :

1. Είναι Διατεταγμένη, δηλαδή τα στοιχεία του πίνακα ξεχωρίζουν από τη σειρά τους : πρώτο, δεύτερο κλπ.
2. Είναι Ομοιογενής: οι τιμές σε έναν πίνακα είναι του ιδίου τύπου οι οποίες είναι αποθηκευμένες σε διαδοχικές θέσεις μνήμης. Κάθε μια από τις τιμές του πίνακα ονομάζεται στοιχείο του πίνακα (element). Για τον ορισμό στη C ενός μονοδιάστατου πίνακα απαιτούνται ο ορισμός του τύπου των στοιχείων του πίνακα και ο καθορισμός του πλήθους των στοιχείων του. Μια καλή προγραμματιστική τεχνική είναι το μέγεθος ενός πίνακα να ορίζεται μέσω μιας συμβολικής σταθεράς καθώς η τροποποίηση του είναι πιο έυκολη και το πρόγραμμα είναι περισσότερο ευανάγνωστο. Ο πίνακας χρησιμοποιείται για την αποθήκευση και διαχείριση δεδομένων κοινού τύπου και αποτελεί, μαζί με τους δείκτες, από τα πλέον ισχυρά εργαλεία της γλώσσας C.

• Η δήλωση του πίνακα ακολουθεί τον εξής φορμαλισμό: τύπος_δεδομένου όνομα_πίνακα[μέγεθος]
Διακρίνονται τρία τμήματα: α) ο τύπος δεδομένου (float, int, char, double), β) το όνομα
του πίνακα και γ) ο αριθμός των στοιχείων που απαρτίζουν τον πίνακα. Έτσι, μία τυπική
δήλωση ενός πίνακα 31 στοιχείων κινητής υποδοαστολής είναι η ακόλουθη:
float temp[31];

• Η αναφορά σε στοιχείο πίνακα γίνεται με συνδυασμό του ονόματος και ενός δείκτη
(index), ο οποίος εκφράζει τη σειρά τού στοιχείου μέσα στον πίνακα:
temp[0]: πρώτο στοιχείο του πίνακα
temp[1]: δεύτερο στοιχείο του πίνακα
temp[30]: τελευταίο (τριακοστό πρώτο) στοιχείο του πίνακα

• Η απόδοση αρχικής τιμής κατά τη δήλωση του πίνακα γίνεται με χρήση του τελεστή
ανάθεσης ως εξής:
float temp[5] = {1,2,-4.2,6,8}: αρχικοποιούνται και τα 5 στοιχεία του πίνακα temp.
float temp[5] = {1,2,-4.2}: αρχικοποιούνται τα 3 πρώτα στοιχεία του πίνακα
temp, δηλαδή τα temp[0], temp[1], temp[2].

• Η ανάγνωση και εκτύπωση ενός πίνακα γίνονται κατά στοιχείο, με τους κανόνες που ισχύουν για κάθε τύπο δεδομένου:

```
for ( i=0; i<ar_size; i++ )
{
scanf( "%f",&ar[i] );
printf( "ar[%d]=%f",i,ar[i] );
 }
```

Παρατηρήσεις

1) Όταν αποδίδονται αρχικές τιμές μπορεί να παραληφθεί το μέγεθος του πίνακα. Ο υπολογιστής θα υπολογίσει αυτόματα πόσα είναι τα στοιχεία του πίνακα από τον αριθμό των αρχικών τιμών που δίδονται. Η παρακάτω δήλωση `char d_ar[ ] = {'a', 'b', 'c', 'd'};` έχει ως αποτέλεσμα τη δημιουργία ενός πίνακα χαρακτήρων (char) τεσσάρων στοιχείων με αρχικές τιμές:

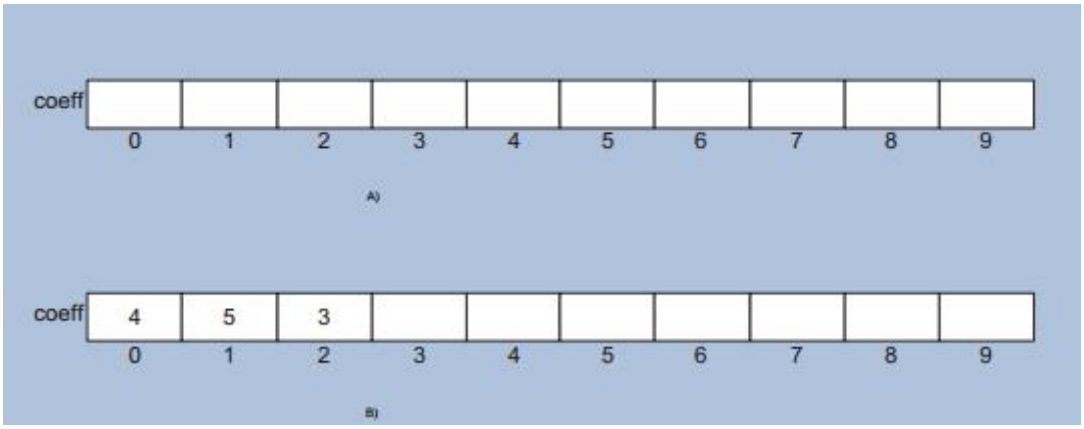
```
d_ar[0] = 'a'
d_ar[1] = 'b'
d_ar[2] = 'c'
d_ar[3] = 'd'
```

2) Το γεγονός ότι οι δείκτες των στοιχείων ενός πίνακα ξεκινούν από το 0 κι όχι από το 1 μπορεί αρχικά να προκαλέσει σύγχυση αλλά αντανακλά τη φιλοσοφία της C, η οποία επιδιώκει να παραμείνει ο προγραμματισμός κοντά στην αρχιτεκτονική του υπολογιστή. Το 0 αποτελεί το σημείο εκκίνησης για τους υπολογιστές. Εάν η αρίθμηση των στοιχείων πίνακα ξεκινούσε από το 1, όπως π.χ. συμβαίνει στη FORTRAN, ο μεταγλωττιστής θα έπρεπε να αφαιρέσει τη μονάδα από κάθε αναφορά σε δείκτη στοιχείου για να ληφθεί η
πραγματική διεύθυνση ενός στοιχείου. Επομένως, η επιλογή της C παράγει πιο αποτελεσματικό κώδικα.

3) Υπάρχει διαφορά ανάμεσα στη δήλωση πίνακα και στην αναφορά στοιχείου πίνακα. Σε μία δήλωση, ο δείκτης καθορίζει το μέγεθος του πίνακα. Σε μία αναφορά στοιχείου πίνακα, ο δείκτης προσδιορίζει το στοιχείο του πίνακα στο οποίο αναφερόμαστε. Π.χ. στη δήλωση `int temp[31];` το 31 δηλώνει τον αριθμό των στοιχείων του πίνακα. Αντίθετα στη `temp[13]=21;` το 13 δηλώνει το συγκεκριμένο στοιχείο (14ο) του πίνακα, στο οποίο αναφερόμαστε και αποδίδουμε την τιμή 21.

4) Μπορεί να βρεθεί το μέγεθος σε bytes ενός πίνακα χρησιμοποιώντας τον τελεστή sizeof. Για παράδειγμα, εάν θεωρηθεί ο πίνακας int ar[5]; η έκφραση sizeof(ar) δίνει τιμή 20 επειδή ο πίνακας αποτελείται από 5 ακεραίους των 4 bytes.
Στη sizeof θα πρέπει να περιλαμβάνεται μόνο το όνομα του πίνακα. Αν περιληφθεί δείκτης ενός στοιχείου, τότε θα εξαχθεί το μέγεθος του στοιχείου. Για παράδειγμα, η έκφραση sizeof(ar[0]) δίνει τιμή 4.
Χρησιμοποιώντας ένα συνδυασμό των παραπάνω μπορεί να βρεθεί ο αριθμός των στοιχείων του πίνακα. Η έκφραση sizeof(ar)/sizeof(ar[0]) δίνει 5, τον αριθμό δηλαδή των στοιχείων του πίνακα ar.

Παράδειγμα μονοδιάστατου πίνακα :



Οι παρακάτω εντολές αναθέτουν τιμές στα στοιχεία του πίνακα που αντιστοιχούν στους αναφερόμενους αριθμοδείκτες :
coeff [0] = 4;
coeff [1] = 5;
coeff [2] = 3;

6.2 Πολυδιάστατοι πίνακες

Για να ορίσουμε στη C ένα διδιάστατο πίνακα ακεραίων X, των 3 γραμμών και 3 στηλών γράφουμε int X[3][3]; Όπου ο πρώτος αριθμοδείκτης καθορίζει το πλήθος των γραμμών και ο δεύτερος το πλήθος των στηλών.

Οι πολυδιάστατοι πίνακες είναι πίνακες, τα στοιχεία των οποίων είναι επίσης πίνακες. Η πρόταση int array[4][12]; δηλώνει τη μεταβλητή array ως πίνακα 4 στοιχείων, όπου το κάθε στοιχείο είναι πίνακας 12 στοιχείων ακεραίων. Η C δε θέτει περιορισμό στον
αριθμό των διαστάσεων των πινάκων. Αν και ο πολυδιάστατος πίνακας αποθηκεύεται στη μνήμη ως μία ακολουθία στοιχείων μίας διάστασης, μπορούμε

να το θεωρούμε ως πίνακα πινάκων. Για παράδειγμα, έστω το επόμενο «μαγικό τετράγωνο», του οποίου οι γραμμές οριζόντια, κάθετα και διαγώνια δίνουν το ίδιο άθροισμα:

17	24	1	8	15
23	5	7	14	16
4	6	13	20	22
10	12	19	21	3
11	18	25	2	9

Για να αποθηκευθεί το τετράγωνο αυτό σε πίνακα θα μπορούσε να γίνει η ακόλουθη δήλωση:

```
int magic[5][5]= { {17, 24, 1, 8, 15},
{23, 5, 7, 14, 16}
{4, 6, 13, 20, 22},
{10, 12, 19, 21, 3},
{11, 18, 25, 2, 9}
};
```

Για την αναφορά σε στοιχείο ενός πολυδιάστατου πίνακα θα πρέπει να καθορισθούν τόσοι δείκτες όσοι είναι αναγκαίοι. Έτσι, η έκφραση magic[1] αναφέρεται στη δεύτερη γραμμή του πίνακα, ενώ η magic[1][3] αναφέρεται στο τέταρτο στοιχείο της δεύτερης γραμμής του πίνακα.

Παρατηρήσεις

1. Όπως και με τους πίνακες μίας διάστασης, έτσι και στους πολυδιάστατους πίνακες εάν αμεληθεί να δοθεί το μέγεθος του πίνακα, ο μεταγλωττιστής θα το καθορίσει αυτόματα με βάση τον αριθμό αρχικών τιμών που παρουσιάζονται. Ωστόσο στους πολυδιάστατους πίνακες μπορεί να παραληφθεί ο αριθμός των στοιχείων μόνο της πρώτης διάστασης, καθώς ο μεταγλωττιστής μπορεί να τον υπολογίσει από τον αριθμό των αρχικών τιμών που διατίθενται. Η παρακάτω δήλωση αξιοποιεί τη δυνατότητα αυτή του μεταγλωττιστή:

```
int ar[ ][3][2]= { { {1, 1}, {0, 0}, {1, 1} },
                   { {0, 0}, {1, 2}, {0, 1} }
                 };
```

Με την παραπάνω δήλωση ο ar δηλώνεται αυτόματα ως πίνακας 2x3x2.

2. Η παρακάτω δήλωση: `int ar[ ][ ]={ 1, 2, 3, 4, 5, 6};` είναι ανεπίτρεπτη επειδή ο μεταγλωττιστής δεν μπορεί να γνωρίζει τι είδους θα ήταν αυτός ο πίνακας. Θα μπορούσε να το θεωρήσει είτε πίνακα 2x3 είτε 3x2.

3. Η παρακάτω δήλωση: printf("%d",array[1,2]); είναι λανθασμένη στη γλώσσα C αλλά δεν εντοπίζεται από το μεταγλωττιστή και οδηγεί σε ανεπιθύμητα αποτελέσματα. Η σωστή είναι printf("%d",array[1][2]);

Πρόγραμμα ανάθεσης τιμών σε πίνακα

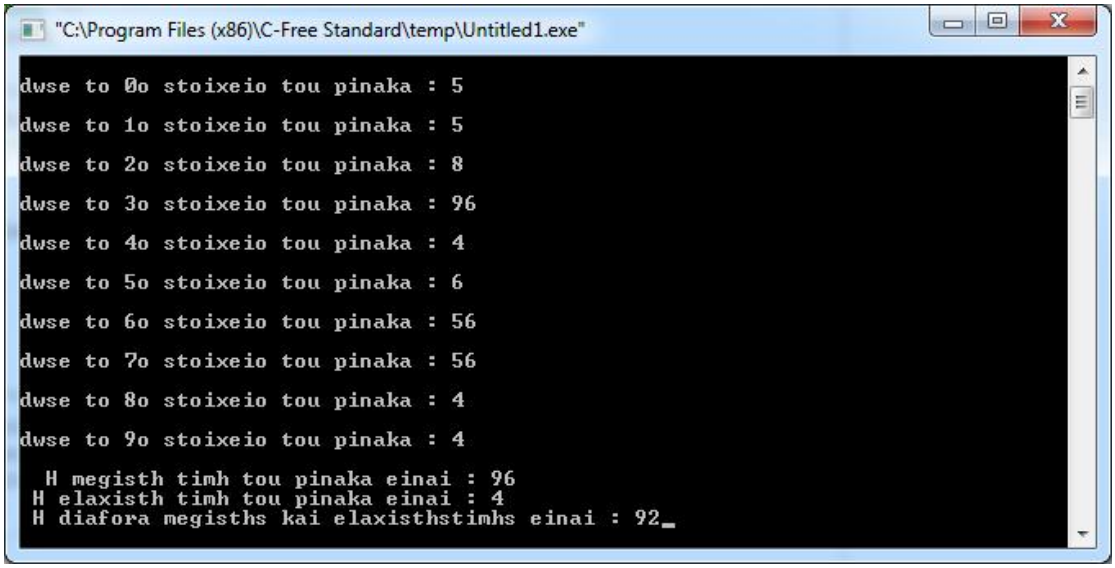
Πραγματοποιείται καταχώρηση ακέραιων τιμών σ' έναν πίνακα 10 θέσεων και να βρεθεί η μέγιστη και η ελάχιστη τιμή του πίνακα καθώς και η μεταξύ τους διαφορά.

```
#include <stdio.h>

main()

{

int i, a[10], max, min;

/* kataxwrisi timwn */

for (i=0; i<10; i++)

{

 printf("\ndwse to %do stoixeio tou pinaka : ", i);

 scanf("%d", &a[i]);

} /* end of for */

max = a[0];

min = a[0];

for (i=1; i<10; i++)

{

 if (a[i] > max)

 max = a[i];

 if (a[i] < min)

 min = a[i];

} /* end of for */

printf("\n  H megisth timh tou pinaka einai : %d", max);

printf("\n H elaxisth timh tou pinaka einai : %d", min);

printf("\n H diafora megisths kai elaxisthstimhs einai : %d", max-min);
```

scanf("%d", &i);

} /* end of main */

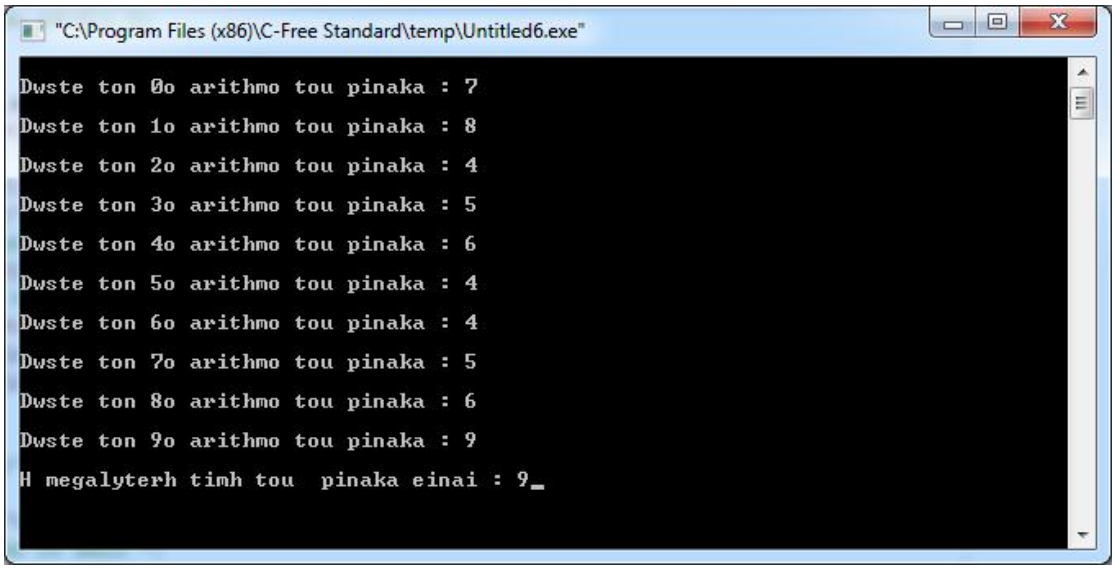
```c
#include <stdio.h>
main()
{
int i, a[10], max, min;
/* kataxwrisi timwn */
for (i=0; i<10; i++)
{
 printf("\ndwse to %do stoixeio tou pinaka : ", i);
 scanf("%d", &a[i]);
} /* end of for */
max = a[0];
min = a[0];
for (i=1; i<10; i++)
{
 if (a[i] > max)
 max = a[i];
 if (a[i] < min)
 min = a[i];
} /* end of for */
printf("\n  H megisth timh tou pinaka einai : %d", max);
printf("\n H elaxisth timh tou pinaka einai : %d", min);
printf("\n H diafora megisths kai elaxisthstimhs einai : %d", max-min);
scanf("%d", &i);
} /* end of main */
```



Πρόγραμμα στο οποίο να διαβαστούν 10 τιμές σ' έναν πίνακα ακεραίων και να βρεθεί η μμεγαλύτερη τιμή.

```
#include <stdio.h>

main()

{

int i, max, a[10];

for (i=0; i<10; i++)

{

 printf("\nDwste ton %do arithmo tou pinaka : ", i);

 scanf("%d", &a[i]);

} /* end of for */

max=a[0];

/* edw vriskoume th megalyterh timh tou pinaka */

for (i=1; i<10; i++)

 if (a[i]>max)

 max=a[i];



printf("\nH megalyterh timh tou  pinaka einai : %d", max);


scanf("%d", &i);

} /* end of main */
```

```c
#include <stdio.h>
main()
{
int i, max, a[10];
for (i=0; i<10; i++)
{
 printf("\nDwste ton %do arithmo tou pinaka : ", i);
 scanf("%d", &a[i]);
} /* end of for */
max=a[0];
/* edw vriskoume th megalyterh timh tou pinaka */
for (i=1; i<10; i++)
 if (a[i]>max)
 max=a[i];
printf("\nH megalyterh timh tou  pinaka einai : %d", max);
scanf("%d", &i);
} /* end of main */
```



7. Το αλφαριθμητικό

Το αλφαριθμητικό ή συμβολοσειρά (string) είναι ένας πίνακας χαρακτήρων που τερματίζει με το μηδενικό (null) χαρακτήρα. Ο μηδενικός χαρακτήρας έχει ASCII κωδικό 0 και αναπαρίσταται από την ακολουθία διαφυγής \0.

• Η δήλωση του αλφαριθμητικού ακολουθεί τον εξής φορμαλισμό:

char όνομα[μήκος]

Διακρίνονται τρία τμήματα:

α) ο τύπος δεδομένου, ο οποίος είναι πάντοτε char

β) το όνομα του αλφαριθμητικού

γ) το μήκος του. Έτσι, μία τυπική δήλωση ενός αλφαριθμητικού 30 χαρακτήρων έχει την ακόλουθη μορφή:

```
char book_title[30]
```

• Τα αλφαριθμητικά μπορούν να εμφανίζονται μέσα στον κώδικα όπως οι αριθμητικές σταθερές, αποτελώντας τις αλφαριθμητικές σταθερές, όπου οι χαρακτήρες της περικλείονται σε διπλά εισαγωγικά. Για την αποθήκευσή της χρησιμοποιείται ένας πίνακας χαρακτήρων, με το μεταγλωττιστή να θέτει αυτόματα στο τέλος του αλφαριθμητικού ένα μηδενικό χαρακτήρα για να προσδιορίσει το τέλος του. Έτσι, η αλφαριθμητική σταθερά “Hello” απαιτεί για αποθήκευση 6 bytes, όπως φαίνεται παρακάτω:

‘H’ ‘e’ ‘l’ ‘l’ ‘o’ ‘\0’.

Παρατήρηση: Θα πρέπει να σημειωθεί η διαφορά ανάμεσα στη σταθερά χαρακτήρα ‘A’ και την αλφαριθμητική σταθερά “A”. Η πρώτη απαιτεί 1 byte για αποθήκευση, ενώ η δεύτερη απαιτεί ένα byte για το χαρακτήρα A κι ένα byte για το null.

7.1 Αρχικοποίηση αλφαριθμητικού

Η ανάθεση τιμής με τη δήλωση ακολουθεί το γενικό κανόνα απόδοσης αρχικής τιμής σε πίνακα:

```
char jisoc[ ] = {‘0’, ‘-’, ‘4’, ‘9’, ‘-’, ‘7’, ‘4’, ‘3’, ‘-’, ‘3’, ‘\0’};
```

Στην πράξη χρησιμοποιείται η ακόλουθη εναλλακτική και πιο συμπαγής μορφή με χρήση αλφαριθμητικής σταθεράς:

```
char jisoc [ ] = “0-49-743-3”;
```

Ο προγραμματιστής πρέπει να περιλάβει ως τελευταία τιμή το null.

7.2 Εισαγωγή αλφαριθμητικού

Η εισαγωγή αλφαριθμητικού από την κύρια είσοδο γίνεται με τη μορφοποιούμενη συνάρτηση scanf και τον προσδιοριστή %s. Η πρόταση scanf(

"%s", jisoc); διαβάζει την κύρια είσοδο ως αλφαριθμητικό και αποθηκεύει την τιμή στη μεταβλητή jisoc. Δε χρειάζεται ο τελεστής & πριν από το όνομα της μεταβλητής jisoc όπως συνέβαινε με τους άλλους τύπους δεδομένων, γιατί το όνομα του αλφαριθμητικού αναπαριστά τη διεύθυνση του πρώτου στοιχείου του.

Εναλλακτικά, η εισαγωγή αλφαριθμητικού μπορεί να γίνει με χρήση της συνάρτησης gets, η γενική μορφή της οποίας είναι gets(όνομα_πίνακα_χαρακτήρων). Καλείται η gets με το όνομα του πίνακα χαρακτήρων ως όρισμα, χωρίς δείκτη. Με την επιστροφή από τη gets το αλφαριθμητικό θα έχει περασθεί στον πίνακα χαρακτήρων. Η gets θα διαβάζει χαρακτήρες από το πληκτρολόγιο έως ότου πατηθεί το ENTER.

Παρατηρήσεις:
1. Το αλφαριθμητικό αποθηκεύεται σε έναν πίνακα χαρακτήρων μαζί με το τελικό μηδενικό χαρακτήρα. Κατά συνέπεια, όταν δηλώνεται το μέγεθος του πίνακα, έστω N, σε αυτόν μπορεί να αποθηκευθεί αλφαριθμητικό μέγιστου μήκους N-1.
2. Θα πρέπει να σημειωθεί ότι τόσο η scanf όσο και η gets δεν εκτελούν έλεγχο ορίων στον πίνακα χαρακτήρων με τον οποίο καλούνται. Εάν π.χ. δηλωθεί char jisoc [30] και το αλφαριθμητικό είναι μεγαλύτερο από το μέγεθος του jisoc, ο πίνακας θα ξεπερασθεί.

7.3 Εκτύπωση αλφαριθμητικού

Η εκτύπωση αλφαριθμητικής σταθεράς γίνεται με την printf χωρίς τη χρήση προσδιοριστή. Απλώς της δίνεται η προς εκτύπωση αλφαριθμητική σταθερά:

```
printf( "Hello" );
```

Η εκτύπωση αλφαριθμητικού γίνεται με την printf χρησιμοποιώντας τον προσδιοριστή %s. Η παρακάτω πρόταση printf("The ISBN code is: %s", jisoc); θα έχει ως αποτέλεσμα να τυπωθεί στην οθόνη η πρόταση The ISBN code is: 0-49-743-3 .

Εναλλακτικά, η εκτύπωση αλφαριθμητικής σταθεράς και αλφαριθμητικού μπορεί να γίνει με χρήση της συνάρτησης puts, η γενική μορφή της οποίας είναι puts(όνομα_πίνακα_χαρακτήρων). Καλείται η puts με το όνομα του πίνακα χαρακτήρων ως όρισμα, χωρίς δείκτη, π.χ. puts(isbn). Βέβαια, η puts παρουσιάζει το μειονέκτημα ότι δεν παρέχει δυνατότητες μορφοποίησης της εξόδου.

7.4 **Συναρτήσεις αλφαριθμητικών**

Η C υποστηρίζει μία ποικιλία συναρτήσεων για το χειρισμό των αλφαριθμητικών. Οι συναρτήσεις αυτές βρίσκονται στο αρχείο κεφαλίδας <string.h>. Οι πιο συνηθισμένες παρουσιάζονται στον ακόλουθο πίνακα, στον οποίο η δεύτερη και η τρίτη στήλη περιέχουν τα ονόματα των συναρτήσεων όταν η λειτουργία τους επιδρά σε ολόκληρο το αλφαριθμητικό ή στους πρώτους n χαρακτήρες, αντίστοιχα:

Λειτουργία	Όλοι οι χαρακτήρες	Οι n πρώτοι χαρακτήρες
Εύρεση μήκους string	strlen()	
Αντιγραφή string	stcpy()	strncpy()
Συνένωση 2 strings	strcat()	strncat()
Σύγκριση 2 strings	strcmp()	strncmp()
Εύρεση χαρακτήρα σε string	strchr()	strrchr()
Εύρεση string σε string	strstr()	

8. **Συναρτήσεις**

Το κλειδί της αποδοτικής επίλυσης προβλημάτων, είναι η διάσπασή τους σε μικρότερα και πιο απλά προβλήματα των οποίων η λύση είναι απλούστερη, και στη συνέχεια η συνένωσή τους ώστε να αποδοθεί η συνολική λύση του όλου προβλήματος. Η C παρέχει τις συναρτήσεις οι οποίες μπορούν να υλοποιήσουν αυτήν την "top-down" τεχνική προγραμματισμού. Κάθε πρόγραμμα λοιπόν, μπορεί να αποτελείται από ένα σύνολο συναρτήσεων (συμπεριλαμβανομένης και της main). Η εκτέλεση ενός προγράμματος βέβαια αρχίζει και τελειώνει στην main αλλά είναι δυνατή η κλήση άλλων συναρτήσεων μέσα στην main, οι οποίες είτε έχουν συμπεριληφθεί στο πρόγραμμα (όπως η printf για παράδειγμα που συμπεριλαμβάνεται στην βιβλιοθήκη stdio.h), είτε έχουν δημιουργηθεί από τον προγραμματιστή.

Η C υποστηρίζει την χρήση συναρτήσεων (functions) αλλά όχι διαδικασιών (procedures) όπως άλλες γλώσσες προγραμματισμού. Μία συνάρτηση είναι η περιγραφή μίας διαδικασίας η οποία αποδίδει (επιστρέφει) μία τιμή η και πιθανώς καμία. Η γενική σύνταξη μίας συνάρτησης είναι η ακόλουθη:

```
Τύπος_επιστρεφόμενου_δεδομένου  Όνομα_Συνάρτησης(Τύποι
παραμέτρων της συνάρτησης)
{
        ....
        Σώμα συνάρτησης
        ...
        [Προαιρετικά] return Επιστρεφόμενη τιμή;
}
```

Μία συνάρτηση περιλαμβάνει τρεις φάσεις:

- τη δήλωση (προαιρετικά) , ονομάζεται και πρωτότυπο
- τον ορισμό, υποστηρίζει το πέρασμα εισόδων στη συνάρτηση
- την κλήση ή τις κλήσεις , όπου επιστρέφει μια τιμή

8.1 Δήλωση συνάρτησης

Στη δήλωση μίας συνάρτησης παρουσιάζεται το «πρότυπο» ή «πρωτότυπο» συνάρτησης, το οποίο αποτελείται από τρία τμήματα, όπου ορίζονται:

· Το όνομα: θα πρέπει να είναι ενδεικτικό της λειτουργίας της.

· Οι είσοδοι (εφόσον υπάρχουν): μία λίστα τυπικών ορισμάτων ή παραμέτρων (formal parameters), αποτελούμενη από ονόματα μεταβλητών εισόδου και τύπων δεδομένων.

· Ο τύπος της εξόδου: τύπος δεδομένων της επιστρεφόμενης τιμής, εφόσον επιστρέφεται τιμή (προκαθορισμένος τύπος: int).

<τύπος επιστρεφόμενου δεδομένου > <όνομα συνάρτησης>(τύποι παραμέτρων);

Για παράδειγμα, η πρόταση :
int maximum_two_integers(int first_integer, int second_integer);
αποτελεί τη δήλωση μίας συνάρτησης ονόματι maximum_two_integers, η οποία δέχεται δύο εισόδους, τις ακέραιες μεταβλητές first_integer και second_integer, και επιστρέφει ακέραια έξοδο (ο τύπος int πριν από το όνομά της).
Η δήλωση των συναρτήσεων γίνεται πριν από τη main(), συνήθως μετά τις εντολές προεπεξεργαστή (#include, #define).

8.2 Ορισμός συνάρτησης

Ο ορισμός μίας συνάρτησης περιλαμβάνει το πρότυπο συνάρτησης χωρίς το καταληκτικό ερωτηματικό, ακολουθούμενο από το σώμα της συνάρτησης, το οποίο αναπτύσσεται μέσα σε άγκιστρα.

Για παράδειγμα, ο ορισμός της συνάρτησης maximum_two_integers() είναι ο ακόλουθος:

```
int maximum_two_integers( int first_integer, int second_integer )
{
 if (first_integer > second_integer) return(first_integer);
 else return(second_integer);
}
```

Παρατηρήσεις:

1. Εάν η συνάρτηση έχει έξοδο, αυτή θα πρέπει να επιστρέφεται με χρήση της εντολής return στο τέλος του σώματος της συνάρτησης. Εάν δεν έχει έξοδο, ο <τύπος δεδομένων επιστροφής> αντικαθίσταται από τη λέξη void, π.χ.

```
void print_max_two_integers( int first_integer, int second_integer )
{
 if (first_integer > second_integer) printf( "max=%d\n",first_integer );
 else printf( "max=%d\n",first_integer );
}
```

2. Οι συναρτήσεις μπορούν να έχουν τις δικές τους εσωτερικές μεταβλητές, όπως ακριβώς έχει η main().

8.3 Κλήση συνάρτησης

Μία συνάρτηση ενεργοποιείται όταν κληθεί. Εάν η συνάρτηση δεν επιστρέφει τιμή, η κλήση της γίνεται από ένα σημείο του προγράμματος ως εξής:
<όνομα συνάρτησης> (πρώτο όρισμα, ..., τελευταίο όρισμα);
Η κλήση (χρήση) των συναρτήσεων σε ένα πρόγραμμα (στην main) γίνεται με το ακόλουθο γενικό τρόπο:
Όνομα μεταβλητής = όνομα συνάρτησης (πραγματικές παράμετροι);
ή στην περίπτωση που η συνάρτηση είναι του τύπου void:
όνομα συνάρτησης (πραγματικοί παράμετροι);

Οι τιμές πρώτο όρισμα, ..., τελευταίο όρισμα καλούνται πραγματικά ορίσματα ή πραγματικές παράμετροι (actual parameters). Τα πραγματικά ορίσματα χωρίζονται με κόμμα και περικλείονται σε παρενθέσεις, αντιστοιχούν δε ένα προς ένα στη λίστα τυπικών ορισμάτων. Ακόμη κι όταν δεν υπάρχουν ορίσματα οι παρενθέσεις είναι υποχρεωτικές, καθώς δηλώνουν στο μεταγλωττιστή ότι το όνομα αντιπροσωπεύει συνάρτηση και όχι μεταβλητή. Τα πραγματικά ορίσματα μπορούν να είναι σταθερές, τιμές μεταβλητών ή αλλά σε κάθε περίπτωση πρέπει να είναι ίδιου τύπου με τα τυπικά ορίσματα. Για παράδειγμα, η κλήση της συνάρτησης print_max_two_integers() μπορεί να λάβει την ακόλουθη μορφή:

```
x=10;

print_max_two_integers( x, 32 );

<επόμενη πρόταση>;
```

Όταν **κληθεί** η συνάρτηση, το τυπικό όρισμα first_integer αντιστοιχεί στο πραγματικό όρισμα x, οπότε first_integer=x=10, και το τυπικό όρισμα second_integer λαμβάνει την τιμή 32. Ο έλεγχος του προγράμματος περνάει στις επόμενες προτάσεις της συνάρτησης. Μετά την εκτέλεση και της τελευταίας πρότασης η συνάρτηση τερματίζει και ο έλεγχος μεταφέρεται στο κυρίως πρόγραμμα, στην <επόμενη πρόταση>;.

Όταν η συνάρτηση έχει **έξοδο** υπάρχει μία μικρή διαφοροποίηση, για παράδειγμα η κλήση της maximum_two_integers() μπορεί να λάβει την ακόλουθη μορφή:

```
x=10;
y=maximum_two_integers( x, 32 );
<επόμενη πρόταση>;
```

Η διαφορά έγκειται στο ότι στο τέλος της συνάρτησης η επιστρεφόμενη τιμή θα πρέπει να αποδοθεί σε μία μεταβλητή του κυρίως προγράμματος, στην παρούσα

περίπτωση στην y. Κατά συνέπεια, αφενός μεν η y θα πρέπει να είναι ίδιου τύπου με την επιστρεφόμενη τιμή (integer στο συγκεκριμένο παράδειγμα), αφετέρου δε μετά το τέλος της συνάρτησης ο έλεγχος του προγράμματος περνά στην πρόταση ανάθεσης

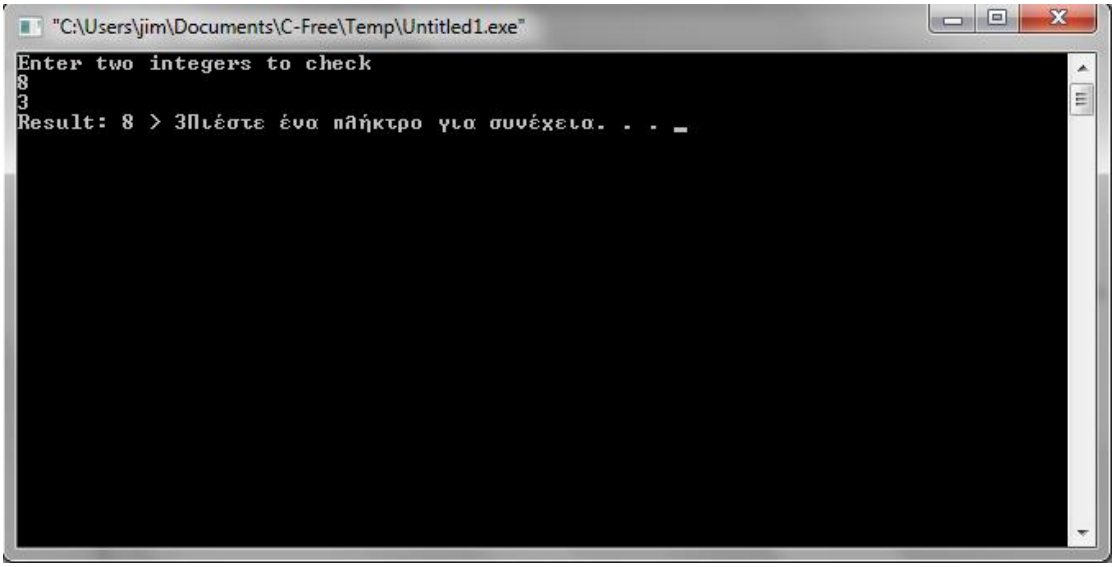
```
y=maximum_two_integers( x, 32 );
```

και ακολούθως στην <επόμενη πρόταση>;

Παράδειγμα

```
#include <stdio.h>
int main(){
    int numb1, numb2;
    printf("Enter two integers to check\n");
    scanf("%d %d",&numb1,&numb2);
    if(numb1==numb2) //checking whether two integers are equal.
        printf("Result: %d = %d",numb1,numb2);
    else
      if(numb1>numb2) //checking whether numb1 is greater than numb2.
        printf("Result: %d > %d",numb1,numb2);
      else
        printf("Result: %d > %d",numb2,numb1);
return 0;
}
```

Το πρόγραμμα θα παίρνει ως είσοδο 3 ακέραιους αριθμούς και θα υπολογίζει το άθροισμα τους, αν το άθροισμα είναι μεγαλύτερο του μηδέν να βρίσκει τον μέσο όρο ενώ σε διαφορετική περίπτωση να δίνει τον μεγαλύτερο. Οι παράμετροί της είναι δύο ακέραιοι αριθμοί, ο numb1 και ο numb2.



Είναι δυνατόν, μία συνάρτηση να μην επιστρέφει καμία τιμή, ή και να μην έχει καμία παράμετρο η τέλος και τα δύο. Σε αυτή την περίπτωση ο τύπος της συνάρτησης ή των παραμέτρων δηλώνεται σαν void.

8.4 Τύποι Συναρτήσεων

Και για τις συναρτήσεις πρέπει να δηλώνεται ο τύπος τους, ο οποίος είναι ίδιος με τον τύπο της τιμής επιστροφής της συνάρτησης. Συναρτήσεις χωρίς τιμή επιστροφής δηλώνονται σαν τύπου void και αν δεν δηλωθεί ο τύπος μιας συνάρτησης, τότε η C θεωρεί ότι είναι τύπου int. Η δήλωση του τύπου είναι μέρος του ορισμού της συνάρτησης και αναφέρεται στην τιμή επιστροφής και όχι στα ορίσματα της συνάρτησης. Μια δήλωση της συνάρτησης λέει στο πρόγραμμα τι τύπου είναι η συνάρτηση, ενώ ο ορισμός της συνάρτησης παρέχει τον πραγματικό κώδικά της. Η δήλωση μιας συνάρτησης πρέπει να γίνεται πριν από το σημείο όπου χρησιμοποιείται η συνάρτηση.

Στην ANSI C, μπορούμε στη δήλωση μιας συνάρτησης να δηλώνουμε και τον τύπο των μεταβλητών της. Το αποτέλεσμα είναι ένα πρωτότυπο συνάρτησης, δηλ. μια δήλωση συνάρτησης όπου καθορίζονται ο τύπος της τιμής επιστροφής, ο αριθμός των ορισμάτων και ο τύπος των ορισμάτων της συνάρτησης.

Ακολουθούν παραδείγματα :

```
int imax(int, int);

int imax(int a, int b);
```

8.5 Αναδρομή

Μία συνάρτηση είναι αναδρομική εάν καλεί (χρησιμοποιεί) τον εαυτό της με άμεσο ή έμμεσο τρόπο. Γενικά, αναδρομή σημαίνει ότι ορίζεται κάτι και στον ορισμό χρησιμοποιείται το οριζόμενο. Στα μαθηματικά πάρα πολλές συναρτήσεις ορίζονται αναδρομικά με πλέον γνωστή την συνάρτηση του παραγοντικού ενός ακέραιου αριθμού. Ο μη αναδρομικός ορισμός του παραγοντικού είναι:

$N! = 1 \cdot 2 \cdot 3 \cdot L \cdot N$

ενώ ο αναδρομικός ορισμός του είναι:

$$N! = \begin{cases} N \cdot (N-1)!, \forall N > 1 \\ 1, N = 0,1 \end{cases}$$

Βασικά πρόκειται περί ακριβώς του ίδιου ορισμού με την διαφορά ότι ο αναδρομικός ορισμός χρησιμοποιεί το οριζόμενο (παραγοντικό) στον ορισμό του. Επίσης, μία βασική διαφορά είναι ότι στον αναδρομικό ορισμό μίας συνάρτηση

πρέπει οπωσδήποτε να φαίνεται που ακριβώς τελειώνει (στο προκείμενο παράδειγμα εάν το *N* γίνει 1 ή 0).

Στην C όλες οι συναρτήσεις μπορούν να κληθούν αναδρομικά (δηλαδή να καλέσουν τον εαυτό τους), ακόμη και η main.

Γενικά, οι απλές αναδρομικές συναρτήσεις ακολουθούν κάποιους πρότυπους κανόνες:

α) Πάντα πρέπει να υπάρχει ένα τέλος, δηλαδή εκείνο το οριακό σημείο που η αναδρομή τελειώνει (if (t<=1) και if (N==1) στα δύο προηγούμενα παραδείγματα),

β) Εκφράζεται η αναδρομή με τον γενικό της τύπο όπου συνήθως είναι η ζητούμενη πράξη σε συνδυασμό με κλήση της ίδιας συνάρτησης αλλά πάντα με μείωση κάποιας από τις παραμέτρους της (return t*parag(t-1) ή return N + athr(N-1) στα δύο προηγούμενα παραδείγματα) .

8.6 Μαθηματικά και C

Η βιβλιοθήκη *math.h* περιέχει πολλές έτοιμες συναρτήσεις. Η χρήση της βιβλιοθήκης αυτής απαιτεί τον διακόπτη *–lm* στο τέλος της διαταγής μετάφρασης (για συστήματα UNIX), πχ

gcc abc.c –o abc.out –lm.

Μερικές από αυτές είναι οι ακόλουθες:

Συνάρτηση Επεξήγηση

double cos(double x)	Συνημίτονο γωνίας x σε rad
double sin(double x)	Ημίτονο γωνίας x σε rad
double tan(double x)	Εφαπτόμενη γωνίας x σε rad
double exp(double x)	Ex
double log(double x)	Log(x)
double log10 (double x)	$Log_{10}(x)$
double pow (double x, double y)	X^y
double sqrt(double x)	√x
double cbrt(double x) 3 *x*	$\sqrt[3]{x}$
int ceil(double x)	Επιστρέφει τον μικρότερο ακέραιο που είναι μεγαλύτερος του αριθμού x, δηλ. ceil(2.0011)→3
int floor(double x)	Επιστρέφει τον στρογγυλοποιημένο ακέραιο του x, δηλ. floor(2.21) →2

8.7 Βασικές συναρτήσεις

printf():

Η συνάρτηση printf() εμφανίζει στην οθόνη μορφοποιημένα δεδομένα ή/και συνοδευτικό κείμενο.

Η γενική μορφή της συνάρτησης είναι:

printf (**ΣΕΙΡΑ_ΕΛΕΓΧΟΥ**, v1, v2,... , vn)

Τα " v1, v2,... , vn ", είναι ονόματα μεταβλητών, σταθερές ή εκφράσεις.

Η ΣΕΙΡΑ_ΕΛΕΓΧΟΥ περικλείεται μέσα σε διπλά εισαγωγικά (") και καθορίζει τον τρόπο εμφάνισης των δεδομένων.

Η ΣΕΙΡΑ_ΕΛΕΓΧΟΥ μπορεί να περιλαμβάνει:

- **επεξηγηματικό κείμενο**, οτιδήποτε κείμενο θέλουμε
- **προσδιοριστές**, που **μπαίνουν απαραίτητα** μέσα στην ΣΕΙΡΑ_ΕΛΕΓΧΟΥ, μόνο όταν θέλουμε να εμφανίσουμε κάποιο στοιχείο (μεταβλητές, σταθερές ή εκφράσεις)
- **χαρακτήρες** ελέγχου, που τους χρησιμοποιούμε για την διαμόρφωση της εμφάνισης

Η **ΣΕΙΡΑ_ΕΛΕΓΧΟΥ** δεν είναι απαραίτητο να περιλαμβάνει όλα τα παραπάνω. Εξαρτάται από το τι θέλουμε να κάνουμε με την printf().

Οι **προσδιοριστές** των στοιχείων τοποθετούνται μέσα στην ΣΕΙΡΑ_ΕΛΕΓΧΟΥ όπου θέλουμε να εμφανιστούν τα δεδομένα, έχουν σχέση με τον τύπο των δεδομένων που εμφανίζονται στην printf() και αρχίζουν με τον χαρακτήρα %. Ο αριθμός των προσδιοριστών είναι ίσος με τον αριθμό των στοιχείων.

\a	Ηχητικό σήμα (<BELL>)
\b	Ο χαρακτήρας <BACKSPACE> (διάστημα πίσω)
\f	Ο χαρακτήρας νέας σελίδας (<FORM FEED>)
\n	Νέας γραμμής (<LINE FEED>)
\r	Επιστροφής (<CR>)
\t	Οριζοντίου προκαθορισμένου διαστήματος (<TAB>)
\v	Κατακόρυφου διαστήματος(<VTAB>)
\'	Εμφάνιση του απλού εισαγωγικού
\"	Εμφάνιση του διπλού εισαγωγικού
\\	Εμφάνιση της ανάποδης πλαγίας καθέτου
\?	Εμφάνιση του λατινικού ερωτηματικού

\xhhh	Εμφάνιση του χαρακτήρα hhh σε δεκαεξαδικό αριθμό
\ooo	Εμφάνιση του χαρακτήρα ooo σε δεκαεξαδικό αριθμό

Παράδειγμα

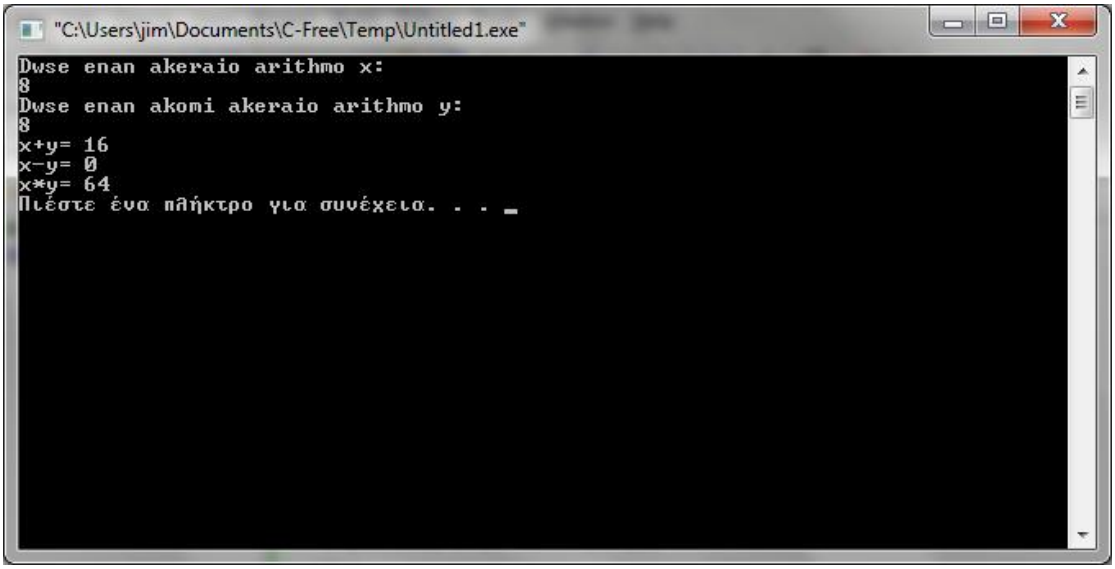
```
#include <stdio.h>
int main()
{
int x;
int y;
printf("Dwse enan akeraio arithmo x:\n");
scanf("%d", &x);
printf("Dwse enan akomi akeraio arithmo y:\n");
scanf("%d", &y);
printf("x+y= %d\n",x+y);
printf("x-y= %d\n",x-y);
printf("x*y= %d\n",x*y);
}
```

Περιγραφή

Το πρόγραμμα ζητά από τον χρήστη δύο ακέραιους αριθμούς(printf)

και θα εκτυπώνει στην οθόνη (scanf):

- Το άθροισμά τους
- Την διαφορά τους
- Το γινόμενό τους.



scanf():

Διαβάζει από το πληκτρολόγιο μορφοποιημένες τιμές μεταβλητών.

scanf (ΣΕΙΡΑ_ΕΛΕΓΧΟΥ, &v1, &v2,... , &vn)

Η συνάρτηση scanf() μοιάζει πολύ με την printf() με μιά μεγάλη διαφορά. Όπως βλέπετε στην περιγραφή της δεν αναφέρεται σε πραγματικές μεταβλητές, αλλά σε διευθύνσεις μεταβλητών (Όταν δηλώνεται μία μεταβλητή σ' ένα πρόγραμμα, ουσιαστικά δεσμεύεται στην μνήμη του υπολογιστή μία περιοχή, η οποία θα φιλοξενήσει κατά την διάρκεια της εκτέλεσης του προγράμματος τις τιμές που θα δώσουμε για αυτή την μεταβλητή. Αυτή η περιοχή μνήμης έχει μία συγκεκριμένη διεύθυνση -πρόκειται για κάποιον συνήθως δεκαεξαδικό αριθμό- που είναι η διεύθυνση της μεταβλητής.)

Στην γλώσσα προγραμματισμού C, γιά ν' αναφερθούμε στην διεύθυνση μιας μεταβλητής χρησιμοποιούμε το όνομα της μεταβλητής προσθέτοντας μπροστά το σύμβολο "&".

Δηλαδή η έκφραση &v1 εννοεί την διεύθυνση της μεταβλητής v1

Προσέξτε όμως για το διάβασμα μιας συμβολοσειράς, δεν χρησιμοποιούμε &, διότι το όνομα της μεταβλητής_συμβολοσειράς είναι ουσιαστικά η διεύθυνση του πρώτου χαρακτήρα.

puts():

Εμφανίζει στην οθόνη μία σειρά χαρακτήρων (συμβολοσειρά). puts(<συμβολοσειρά>|<μεταβλητή_συμβολοσειράς>);

Είναι πιο απλή και πιο γρήγορη από την printf() αλλά χρησιμοποιείται μόνο για συμβολοσειρές (όχι για τις μεταβλητές ή τις σταθερές). Η puts() μετά την εμφάνιση της συμβολοσειράς, τυπώνει τον χαρακτήρα \n, δηλαδή επιστρέφει στην επόμενη σειρά. Ο χαρακτήρας \0 που υποδεικνύει το τέλος της συμβολοσειράς χρησιμοποιείται από την puts() και καλό θα είναι να προσέχετε, ώστε η συμβολοσειρά που έχετε δώσει για εκτύπωση να περιέχει στο τέλος τον χαρακτήρα \0.

gets():

Διαβάζει από το πληκτρολόγιο (είσοδος -stdin) μία σειρά χαρακτήρων (συμβολοσειρά). gets(<μεταβλητή_συμβολοσειράς>);

Ιδιαίτερα χρήσιμη για διαλογική συζήτηση με τον υπολογιστή.

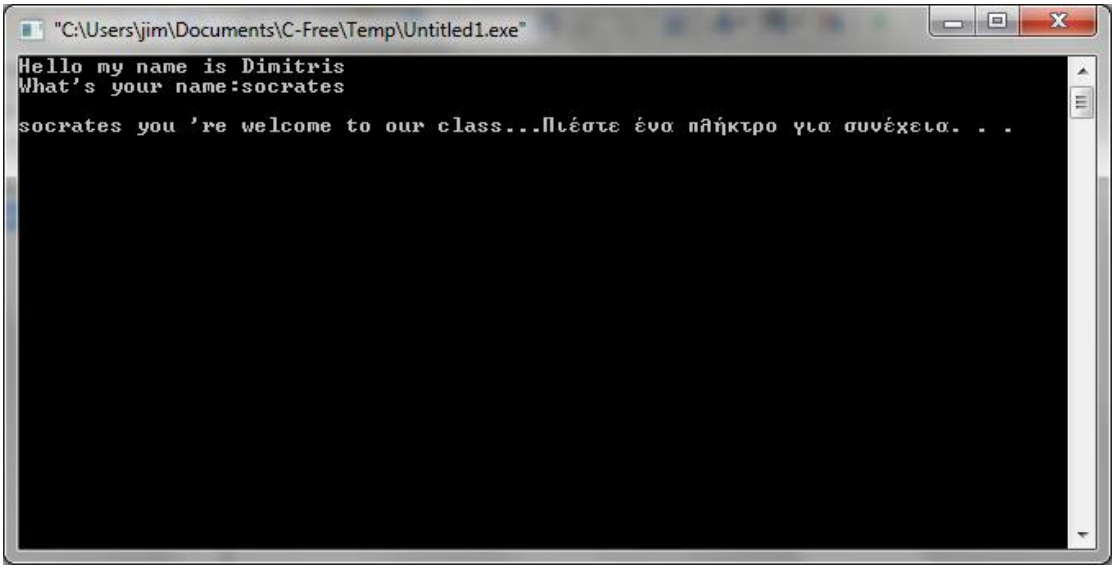
Παράδειγμα

```
char name[20];

printf("What's your name:"); gets(name); printf("Hello "); puts(name);
```

Η gets() διαβάζει από το πληκτρολόγιο μέχρι να συναντήσει τον χαρακτήρα \n (πράγμα που συμβαίνει όταν πατήσουμε το <ENTER>). Ο χαρακτήρας \0 μπαίνει αυτόματα στο τέλος της συμβολοσειράς που πληκτρολογείτε.

Παράδειγμα puts – gets

```
main()

 { char aaa[20];

 char bbb[10] = "Dimitris";

printf("Hello my name is ");

puts(bbb);

 printf("What's your name:");

 gets(aaa);

printf("\n");

 printf("%s you 're welcome to our class...",aaa);

}
```



Περιγραφή

Η putchar():

- Εμφανίζει στην οθόνη ένα μόνο χαρακτήρα.

Η getchar():

- Διαβάζει από το πληκτρολόγιο ένα χαρακτήρα. Η συνάρτηση δεν δέχεται παραμέτρους, απλά παίρνει έναν χαρακτήρα από το πληκτρολόγιο και δεν χρειάζεται να δώσουμε <ENTER>.

9. Δείκτες

9.1 Εισαγωγή στους Δείκτες (Pointers)

Κάθε μεταβλητή σχετίζεται με μία θέση στην κύρια μνήμη του υπολογιστή, η οποία χαρακτηρίζεται από τη διεύθυνσή της. Στη γλώσσα μηχανής μπορεί να γίνει άμεση χρήση αυτής της διεύθυνσης για να αποθηκευθούν ή να ανακληθούν δεδομένα. Αντίθετα, στις γλώσσες προγραμματισμού υψηλού επιπέδου οι διευθύνσεις δεν είναι άμεσα ορατές από τον προγραμματιστή καθώς καλύπτονται από το μανδύα των συμβολικών ονομάτων, τα οποία το σύστημα αντιστοιχεί στις πραγματικές διευθύνσεις.

Η γλώσσα C, θέλοντας να δώσει στον προγραμματιστή τη δυνατότητα συγγραφής αποδοτικού κώδικα, υποστηρίζει την άμεση διαχείριση των περιεχομένων της μνήμης, εισάγοντας την έννοια του δείκτη (pointer). Ο δείκτης αποτελεί μία μεταβλητή που περιέχει μία διεύθυνση μνήμης

Χρησιμοποιήσαμε προηγουμένως τον τελεστή & για να βρούμε τη διεύθυνση της μεταβλητής p. Η &p δηλαδή είναι ένας δείκτης της p. Η πραγματική διεύθυνση είναι ένας 16δικός αριθμός και η συμβολική παράσταση &p είναι μια σταθερά δείκτη. Η μεταβλητή p δεν πρόκειται να αλλάξει διεύθυνση μέσα στο πρόγραμμα, αν και μπορεί να αλλάξει τιμή. Η C έχει και μεταβλητές δείκτη που έχουν μια διεύθυνση ως τιμή.

9.2 Δήλωση δείκτη

Η δήλωση μίας μεταβλητής δείκτη ακολουθεί τον εξής φορμαλισμό: <τύπος_δεδομένων> * <όνομα_δείκτη>;

π.χ. int *pnum;

Δύο βασικοί τελεστές που σχετίζονται με τιμές δεικτών :

& **Διεύθυνση_του**

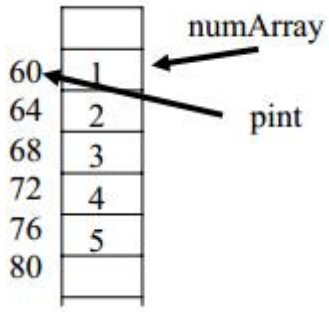
* **Δεικτοδοτούμενη_τιμή**

Όταν ένας δείκτης έχει αποθηκευμένη μία διεύθυνση έχει επικρατήσει να λέγεται ότι ο δείκτης «δείχνει» στη διεύθυνση. Στη δήλωση δείκτη ο **<τύπος_δεδομένων>** αφορά στο είδος των δεδομένων που αποθηκεύονται στη διεύθυνση που «δείχνει» ο δείκτης.
Τον τύπο δεδομένων ακολουθεί ο **αστερίσκος**, ο οποίος είναι απαραίτητος γιατί προσδιορίζει ότι δηλώνεται μία μεταβλητή δείκτη κι όχι μία «κανονική» μεταβλητή. Ο αστερίσκος συνδέεται με το όνομα και όχι με τον τύπο δεδομένων.
Τη δήλωση δείκτη κλείνει το όνομα του δείκτη. Επιλέγεται με βάση τις ίδιες συμβάσεις που ισχύουν στις κανονικές μεταβλητές. Ωστόσο, συνήθως ο αρχικός χαρακτήρας του ονόματος δείκτη είναι το p.

9.3 Ανάθεση τιμής σε δείκτη

Η ανάθεση τιμής σε δείκτη μπορεί να γίνει με έναν από τους ακόλουθους τέσσερις τρόπους: Με χρήση πινάκων, δεδομένου ότι το όνομα ενός πίνακα αντιστοιχεί στη διεύθυνση του πρώτου στοιχείου του. Έτσι, με τον ακόλουθο κώδικα αποδίδεται στο δείκτη ακεραίων pint η τιμή 60, δηλαδή η διεύθυνση του πρώτου στοιχείου του πίνακα numArray.

```
int numArray[5] = {1,2,3,4,5};
int *pint;
pint = numArray;
```



Ανάθεση τιμής σε δείκτη με χρήση πίνακα

9.4 Αριθμητικοί δείκτες

Η C διαθέτει μόνο δύο πράξεις για δείκτες, προσαύξηση και μείωση. Η μορφή που μπορούν να πάρουν οι εκφράσεις είναι οι ακόλουθες :

pnum=pint+y; ή pnum=pint-y;

όπου οι pnum, pint είναι δείκτες ίδιου τύπου και ο y είναι ακέραιος.

Όταν προσαυξάνουμε ένα δείκτη κατά 1, ο δείκτης αυξάνεται αυτόματα ώστε να παραπέμπει στο επόμενο στοιχείο

- Έχει νόημα μόνο όταν ο δείκτης δείχνει σε πίνακα!
- Ο compiler γνωρίζει το είδος των δεδομένων (από τη δήλωση pointer) και αυξάνει τη διεύθυνση του δείκτη, σύμφωνα με το μέγεθος του τύπου δεδομένων.
- Η ίδια λογική ισχύει και για τη μείωση pointer.

9.5 **Ο ειδικός δείκτης** NULL

Ειδική τιμή που δηλώνει ότι ένας δείκτης “δείχνει” σε μια μη έγκυρη διεύθυνση. Δεν επιτρέπεται η πρόσβαση στην τιμή ενός δείκτη που έχει τιμή NULL. Μια τέτοια κλήση έχει ως αποτέλεσμα, συνήθως, την κατάρρευση του προγράμματος.

```
p = NULL;
*p = 1; /* Το πρόγραμμα θα καταρρεύσει */
```

9.6 **Δείκτες σε συναρτήσεις**

Ο δείκτης σε συνάρτηση είναι μια από τις πιο δύσκολες έννοιες στη C. Ουσιαστικά ο δείκτης κρατά τη διεύθυνση εισόδου (entry point) του εκτελέσιμου στιγμιότυπου της συνάρτησης. Η χρήση τους δεν είναι πολύ συνηθισμένη σε προγράμματα εφαρμογών, όμως είναι αρκετά κοινή σε βιβλιοθήκες λογισμικού. Εκεί συνήθως οι δείκτες σε συναρτήσεις εμφανίζονται ως παράμετροι σε κλήσεις άλλων συναρτήσεων.

Αυτή η τεχνική είναι ιδιαίτερα χρήσιμη όταν έχουμε εναλλακτικές συναρτήσεις που μπορεί να χρησιμοποιηθούν για να εκτελέσουν παρόμοιες εργασίες σε δεδομένα. Τότε μπορούμε να περάσουμε τα δεδομένα και τη συνάρτηση που θα χρησιμοποιήσουμε ως παραμέτρους στη κλήση μιας συνάρτησης ελέγχου (ή οδήγησης). Σύντομα θα δούμε δύο τέτοια παραδείγματα συναρτήσεων ταξινόμησης (qsort) και αναζήτησης (bsearch) . Εύκολα μπορείτε να ενσωματώσετε μια δική σας συνάρτηση μέσω της συνάρτησης ελέγχου.

Η δήλωση δείκτη σε συνάρτηση είναι ως εξής:

```
int (*pf) ();
```

Αυτή η δήλωση απλά κρατά μια θέση δείκτη *pf σε μια συνάρτηση που επιστρέφει ένα int. Ακόμη δεν δείχνεται κάποια συγκεκριμένη συνάρτηση.

Αν έχουμε μια συνάρτηση int f() απλά γράφουμε:

```
pf = &f;
```

Για την καλύτερη λειτουργία της προτυποποίησης είναι καλό οι δηλώσεις να έχουν πιο συγκεκριμένη μορφή :

```
int f(int);
int (*pf) (int) = &f;
```

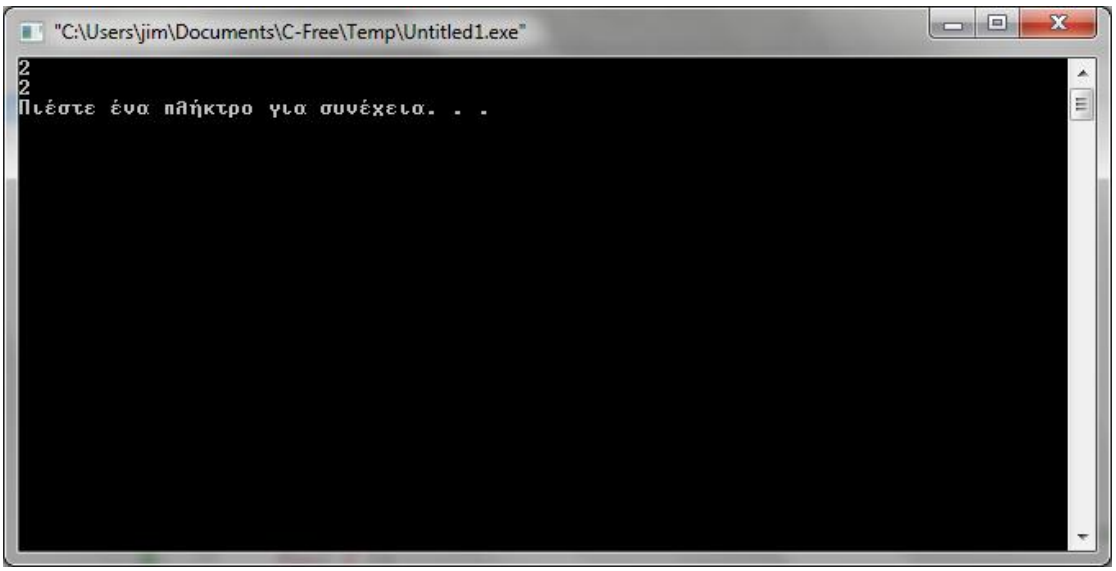
Τώρα η f() επιστρέφει έναν int και δέχεται ένα int ως παράμετρο.

Με αυτή τη δήλωση τα παρακάτω είναι ισοδύναμα:

ans = f(5);
ans = pf(5);

Παράδειγμα

```
#include <stdio.h>
void my_int_func(int x)
{
    printf( "%d\n", x );
}


int main()
{
    /* pointer to function declaration */
    void (*foo)(int);

    /* pointer to function initialisation */
    foo = &my_int_func;

    /* call my_int_func (note that you do not need to write (*foo)(2)
) */
    foo( 2 );

    /* but if you want to, you may */
    (*foo)( 2 );

    return 0;
}
```



Περιγραφή

Η πρότυπη συνάρτηση βιβλιοθήκης qsort() είναι πολύ χρήσιμη, γιατί είναι σχεδιασμένη να ταξινομεί ένα πίνακα βάσει ενός κλειδιού οποιουδήποτε τύπου αρκεί τα στοιχεία του πίνακα να είναι οποιουδήποτε (συγκεκριμένου, ενιαίου) τύπου.

Η συνάρτηση qsort() ορίζεται στη πρότυπη βιβλιοθήκη stdlib.h:

void qsort(void *base, size_t num_elements, size_t element_size, int (*compare)(void const *, void const *));

Η παράμετρος base δείχνει στον πίνακα προς ταξινόμηση. Το num_elements ορίζει το μέγεθος του πίνακα (σε bytes) --συνήθως το size_t προκαθορίζεται σε int. Το element_size ορίζει το μέγεθος κάθε στοιχείου του πίνακα (σε bytes). Η τελική παράμετρος compare είναι δείκτης σε μια συνάρτηση, που δέχεται δύο παραμέτρους -- δείκτες στα κλειδιά με γενικό τύπο (void) και επιστρέφει ένα int.

Η συνάρτηση qsort() καλεί τη συνάρτηση compare() η οποία πρέπει να οριστεί από το χρήστη, για να ορίσει το τρόπο σύγκρισης των στοιχείων του πίνακα. Σημειώστε οτι η qsort() διατηρεί την ανεξαρτησία της από τον τύπο δεδομένων του πίνακα αναθέτοντας την ευθύνη της τελικής σύγκρισης στο χρήστη. Ο int που επιστρέφει η compare() τυπικά ακολουθεί παρακάτω σύμβαση (για αύξουσα ταξινόμηση):

μικρότερος του μηδενός

: αν η τιμή της πρώτης παραμέτρου είναι μικρότερη της δεύτερης ίσος με το μηδέν
: αν η τιμή της πρώτης παραμέτρου είναι ίση με αυτή της δεύτερης μεγαλύτερος του μηδενός
: αν η τιμή της πρώτης παραμέτρου είναι μεγαλύτερη της δεύτερης
Με αυτό το τρόπο μπορούμε να ταξινομήσουμε αρκετά σύνθετες δομές δεδομένων. Για παράδειγμα, για να ταξινομήσουμε ένα πίνακα με στοιχεία δομής Record και κλειδί int:

```
typedef struct {
      int   key;
          struct other_data;
} Record;
```

Μπορούμε να γράψουμε μια συνάρτηση σύγκρισης, recordcompare:

```
int recordcompare(void const *a, void  const *b) {
   return ( ((Record *)a)->key - ((Record *)b)->key );
}
```

Έστω οτι έχουμε ένα πίνακα array με στοιχεία Record και μέγεθος πίνακα arraylength. Μπορούμε να καλέσουμε τη συνάρτηση qsort() ως εξής:

```
qsort(arπray, arraylength, sizeof(Record), recordcompare);
```

10. Δημιουργία και Χρήση Δομών

10.1 Δημιουργία Δομής (struct)

Όπως έχουμε δει, οι πίνακες επιτρέπουν την ομαδοποίηση ίδιων στοιχείων (π.χ. floats). Ωστόσο, είναι πιθανόν να θέλουμε να ομαδοποιήσουμε δεδομένα τα οποία διαφοροποιούνται μεταξύ τους ως προς τον τύπο. Αυτό επιτυγχάνεται με χρήση του struct:

```
struct struct_name
{
struct_field1
struct_field2
.......
struct_fieldN
}
```

Έστω για παράδειγμα ότι θέλουμε να φτιάξουμε έναν τύπο ο οποίος περιέχει πληροφορία για τους υπαλλήλους μίας εταιρίας:

```
struct Employee
```

{

char name[50];

unsigned int age;

float salary;

unsigned int code_number;

};

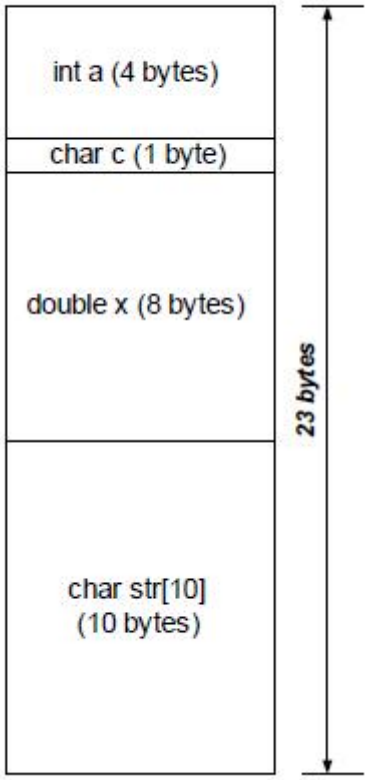
Για να δηλώσουμε μία μεταβλητή για τον παραπάνω τύπο γράφουμε για παράδειγμα:

struct Employee employee1;

Για να προσπελάσουμε τα πεδία μίας δομής χρησιμοποιούμε την τελεία (.). Για παράδειγμα, μπορούμε να γράψουμε:

employee1.salary = 1050.5

Το μέγεθος (σε bytes) μίας δομής είναι ίσο με το άθροισμα των μεγεθών των επιμέρους μεταβλητών. Στο ακόλουθο σχήμα φαίνεται ένα παράδειγμα μίας δομής η οποία έχει για πεδία: έναν ακέραιο, έναν χαρακτήρα, έναν double και έναν πίνακα χαρακτήρων 10 θέσεων:



10.2 Δείκτες σε struct

Για να ορίσουμε έναν δείκτη σε δομή, κάνουμε ότι και με τα υπόλοιπα είδη μεταβλητών:

```
struct structName *varName;
```

Για παράδειγμα:

```
struct Employee *E
```

Για να προσπελάσουμε ένα πεδίο ενός δείκτη σε δομή κάνουμε το εξής:

```
(*varName).fieldName;
```

Για παράδειγμα:

```
(*E).code_number;
```

Επειδή οι δείκτες για δομές χρησιμοποιούνται πολύ συχνά, παρέχεται ένας εναλλακτικός τρόπος συμβολισμός σαν συντομογραφία. Αν pVarName είναι δείκτης σε δομή, τότε η έκφραση pVarName -> fieldname αναφέρεται στο συγκεκριμένο πεδίο. Πρέπει να προσέξουμε ότι όταν θέλουμε να περάσουμε μία δομή σαν όρισμα σε μία συνάρτηση και να αλλάξουμε το περιεχόμενο ενός πεδίου, τότε πρέπει να χρησιμοποιήσουμε δείκτη σε δομή (όπως ακριβώς συμβαίνει και με όλα τα είδη μεταβλητών).

Πρόγραμμα εκμάθησης των πινάκων δομών (struct)

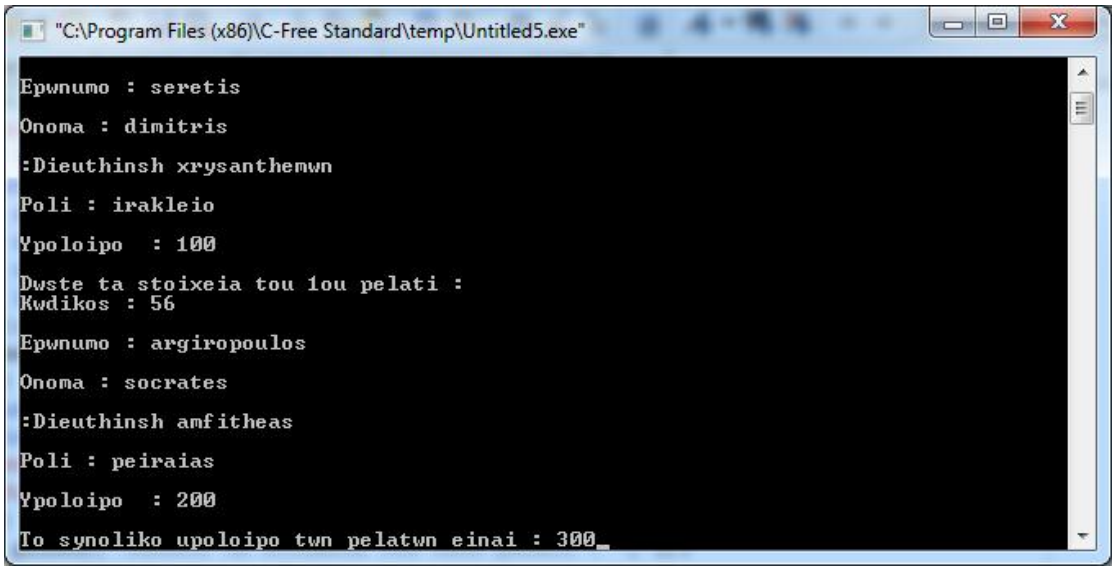
Δημιουργία μιας δομής πελάτη με τα εξής στοιχεία : κωδικός, επώνυμο, όνομα, διεύθυνση, πόλη και υπόλοιπο και να καταχωρηθούν τιμές για 10 υπαλλήλους σ' έναν πίνακα δομών - να εκτυπωθούν το επώνυμο και το όνομα των πελατών που έχουν υπόλοιπο μεγαλύτερο από 100000 καθώς και το συνολικό υπόλοιπο όλων των πελατών.

```
#include <stdio.h>

struct pelates {

 int code;

char eponymo[15];

 char onoma[10];

 char address[20];

 char poli[10];

 long ypoloipo;

}; /* end of struct */

main()

{

int i;

long sum=0l;

struct pelates pel[10];

for (i=0; i<2; i++)

{

 printf("\nDwste ta stoixeia tou %dou pelati : ", i);

 printf("\nKwdikos : ");

 scanf("%d", &pel[i].code);

 printf("\nEpwnumo : ");

 scanf("%s", pel[i].eponymo);

 printf("\nOnoma : ");

scanf("%s", pel[i].onoma);

 printf("\n:Dieuthinsh ");
```

```
scanf("%s", pel[i].address);
printf("\nPoli : ");
scanf("%s", pel[i].poli);
printf("\nYpoloipo  : ");
scanf("%ld", &pel[i].ypoloipo);
sum += pel[i].ypoloipo;
if (pel[i].ypoloipo > 100000)
printf("\nO %s %s exei upoloipo %ld",pel[i].eponymo, pel[i].onoma, pel[i].ypoloipo);
} /* end of for */
printf("\nTo synoliko upoloipo twn pelatwn einai : %ld", sum);
scanf("%d", &i);
} /* end of main */
```

```c
#include <stdio.h>
struct pelates {
 int code;
char eponymo[15];
 char onoma[10];
 char address[20];
 char poli[10];
 long ypoloipo;
}; /* end of struct */
main()
{
int i;
long sum=0l;
struct pelates pel[10];
for (i=0; i<2; i++)
{
 printf("\nDwste ta stoixeia tou %dou pelati : ", i);
 printf("\nKwdikos : ");
 scanf("%d", &pel[i].code);
 printf("\nEpwnumo : ");
 scanf("%s", pel[i].eponymo);
 printf("\nOnoma : ");
scanf("%s", pel[i].onoma);
 printf("\n:Dieuthinsh ");
 scanf("%s", pel[i].address);
 printf("\nPoli : ");
 scanf("%s", pel[i].poli);
 printf("\nYpoloipo  : ");
 scanf("%ld", &pel[i].ypoloipo);
 sum += pel[i].ypoloipo;
 if (pel[i].ypoloipo > 100000)
 printf("\nO %s %s exei upoloipo %ld",pel[i].eponymo, pel[i].onoma, pel[i].ypoloipo);
} /* end of for */
```

```
34 printf("\nTo synoliko upoloipo twn pelatwn einai : %ld", sum);
35 scanf("%d", &i);
36 } /* end of main */
```



ΒΙΒΛΙΟΓΡΑΦΙΑ

1. Brian W. Kernighan, Dennis M. Ritchie: "Η Γλώσσα Προγραμματισμού C ", Prentice Hall (Ελληνική Μετάφραση,εκδ. Κλειδάριθμος), 1988.

2. Herbert Schildt: "Odhgìc thc C ", 3η έκδ., McGraw-Hill (Ελληνική Μετάφραση Γκιούρδας), 2004.

3. Γ. Σ. Τσελίκης - Ν. Δ. Τσελίκας, «C: από τη Θεωρία στην Εφαρμογή»,Β' Έκδοση, 2012

4. H.M. Deitel, P.J. Deitel, C Προγραμματισμός, Εκδόσεις Γκιούρδα, 2003

5. E. Roberts, Η τέχνη και επιστήμη της C, Εκδόσεις Κλειδάριθμος, 2005.

6. A. Kelley, I. Pohl, A Book on C, Addison-Wesley, 1997.
7. A. Hunt, D. Thomas, The Pragmatic Programmer: From Journeyman to Master, Addison-Wesley, 1999.

8. Noel Kalicharan, "C by Example", University of Cambridge, 1994

Ηλεκτρονική

1. Hunt, D. Thomas, The Pragmatic Programmer: From Journeyman to Master, Addison-Wesley, 1999.
2. http://gd.tuwien.ac.at/languages/c/programming-bbrown/