ΤΕΧΝΟΛΟΓΙΚΟ ΙΔΡΥΜΑ ΛΑΡΙΣΑΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ &
ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΣΗΜΕΙΩΣΕΙΣ ΓΙΑ ΤΟ ΜΑΘΗΜΑ

ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ Ι

(Γλώσσα Προγραμματισμού C)

Δρ. Ηλίας Κ. Σάββας,
Χρήστος Κοκκινόπουλος
ΛΑΡΙΣΑ, Ιανουάριος 2005

Πίνακας Περιεχομένων

1. ΕΙΣΑΓΩΓΗ

Ο πρώτος υπολογιστής του κόσμου, βάρους 17 τόνων και μεγέθους τέτοιου που γέμιζε ένα 9m x 17m δωμάτιο, ήταν τόσο μεγάλος όσο ένας δεινόσαυρος. Ονομαζόταν ENIAC (Electronic Numerator Integrator Analyser and Computer) και δημιουργήθηκε το 1946. Όταν αυτός ο τεράστιος υπολογιστής έλυνε ένα πρόβλημα, έκανε μάλλον μεγάλους και διάφορους θορύβους μιας και χρησιμοποιούσε 17.000 ηλεκτρονικές λυχνίες οι οποίες ύστερα από λίγο θερμαινόταν έτσι ώστε πολλές από αυτές να σπάνε με αποτέλεσμα μια ολόκληρη ομάδα ανθρώπων ασχολούταν με την αντικατάσταση τους. Βέβαια, πριν τον ENIAC χρειαζόταν ένα δωμάτιο γεμάτο με «έξυπνους» ανθρώπους για να λύσουν αντίστοιχα προβλήματα.

Μπορεί η εποχή της Πληροφορικής να άρχισε κάνοντας τα πρώτα της βήματα πολύ αργά, αλλά μετά το 1971 όπου η Intel ανέπτυξε το πρώτο ολοκληρωμένο κύκλωμα (computer chip) μεγέθους 12 χιλιοστών και με δυνατότητες 12 φορές του ENIAC, η ανάπτυξή της ήταν πλέον ραγδαία και η τάση σαφής: οι υπολογιστές να γίνονται όλο και πιο μικροί και όλο και πιο γρήγοροι.

Είναι κοινά παραδεκτό πλέον, ότι από περίπου την δεκαετία του 60 οι υπολογιστές «έχουν μπει για τα καλά στην ζωή μας». Έχουν επηρεάσει σχεδόν όλες τις επιστήμες. Την Ιατρική, την Φυσική, τις Επικοινωνίες και πολλές άλλες όπως την Αρχαιολογία ή την Νομική. Έχουν επηρεάσει τον τρόπο που λειτουργούν οι επιχειρήσεις και την τελευταία δεκαετία έχουν εισβάλλει και στα σπίτια μας. Το διαδίκτυο (Internet) είναι γνωστό μέχρι και στους μαθητές των Δημοτικών μας σχολείων. Ιστοσελίδες έχουν δημιουργήσει και δημιουργούν όλες οι κοινωνικές τάξεις. Από επιστήμονες και τον κόσμο των επιχειρήσεων μέχρι απλοί μαθητές ή και παιδιά προσχολικής ηλικίας. Νέες λέξεις έχουν εισβάλλει δυναμικά στην ζωή μας όπως «δίκτυα», «κατανεμημένα συστήματα», «πελάτης / διακομιστής» και πολλές άλλες. Η νέα κοινωνία, η Κοινωνία της Πληροφορικής αλλάζει δραματικά γρήγορα την ζωή μας και μας επιβάλλει νέους ρυθμούς και συνήθειες. Για παράδειγμα ποιος μπορούσε πριν λίγα χρόνια να φανταστεί ότι το ηλεκτρονικό ταχυδρομείο θα μας ήταν τόσο μα τόσο απαραίτητο ή ότι θα μπορούσαμε πλέον να υποβάλλουμε τις φορολογικές μας δηλώσεις και να κάνουμε τραπεζικές συναλλαγές από τον υπολογιστή του σπιτιού μας. Και το κυριότερο είναι ότι η Πληροφορική αλλάζει και εξελίσσεται με ρυθμούς που δεν έχουν παρατηρηθεί ξανά σε καμία άλλη επιστήμη, ούτε καν στην Χημεία κατά την διάρκεια του πρώτου Παγκόσμιου Πολέμου ή στην Μηχανική κατά την Βιομηχανική Επανάσταση.

Είναι πλέον γεγονός ότι ζούμε σε μια εποχή ραγδαίων εξελίξεων όπου η Πληροφορική έχει τον κυρίαρχο ρόλο. Είναι λογικό λοιπόν που λέγεται από τους ειδικούς ότι ζούμε στην εποχή της Πληροφορικής που δημιουργεί την Κοινωνία της Πληροφορικής. Μερικές χαρακτηριστικές αλλαγές αλλά και νέες ανακαλύψεις που οφείλονται κατά κύριο λόγο στην Πληροφορική είναι οι ακόλουθες:

⇒ Αξονικός τομογράφος, Μαγνητικός τομογράφος,
⇒ Μετάφραση του DNA, Επίδραση στο DNA,
⇒ Κινητή Τηλεφωνία,
⇒ CD/DVD,
⇒ Νέες μορφές αποθήκευσης και επεξεργασίας εικόνων και ήχων,
⇒ Καλύτερη και ταχύτερη πρόβλεψη καιρού,

⇒ Καλύτερα και τελειότερα οπλικά συστήματα (δυστυχώς),
⇒ Καλύτερος έλεγχος εναέριας κυκλοφορίας,
⇒ Βελτιστοποίηση της Αστυνομικής έρευνας,
⇒ Βελτιστοποίηση των τεχνητών ανθρώπινων μελών, και αναρίθμητες άλλες.

Και εάν αυτά δεν είναι αρκετά για να πεισθεί κάποιος για την επίδραση της Πληροφορικής στη ζωή του σύγχρονου ανθρώπου, αρκεί να σκεφτεί ότι είναι δυνατόν η νοικοκυρά να ψωνίζει στο Διαδίκτυο τα καθημερινά της ψώνια ή μπορεί κάποιος να βρει το έτερον του ήμισυ με τη βοήθεια της Πληροφορικής στο Διαδίκτυο.

1.1 Ιστορική Αναδρομή

Κάποτε στο παρελθόν ο άνθρωπος άρχισε να διακρίνει τον ένα άνθρωπο από τους δύο και από τους πολλούς. Άρχισε να μετράει (κυριολεκτικά) τους φίλους του, τους εχθρούς του, τα ζώα του και να συγκρίνει τους πληθυσμούς τους. Ίσως αργότερα χρησιμοποίησε τα δάχτυλά του ή άλλους στοιχειώδεις τρόπους αρίθμησης μέχρις ότου περί το 2000 π.Χ. εφευρέθηκε η πρώτη αθροιστική μηχανή: ο άβακας.
Με την πρόοδο των Μαθηματικών και την ανάπτυξη της άλγεβρας από τους Άραβες ωρίμασε η ιδέα της «συνταγής» για την εκτέλεση των πράξεων. Έτσι παρουσιάσθηκε η έννοια του αλγόριθμου.

Το επόμενο βήμα ήταν μάλλον η συνειδητοποίηση του ότι οι πράξεις μπορούν να γίνουν αθροίζοντας μήκη (αρχή του λογαριθμικού κανόνα), που ακολούθησε την εφεύρεση των λογαρίθμων από τον Napier περί το 1600 μ.Χ. Το 1642 ο Pascal (1623-1662) κατασκεύασε την πρώτη αθροιστική μηχανή που λειτουργούσε με οδοντωτούς τροχούς και μοχλούς. Ένας βελτιωμένος τύπος της μηχανής αυτής, που κατασκευάστηκε από τον Leibnitz το 1673, μπορούσε να εκτελεί και πολλαπλασιασμούς και διαιρέσεις.

Τον 19ο αιώνα, ο Charles Babbage (1792-1871) συνέλαβε την ιδέα της προγραμματισμένης εκτελέσεως των πράξεων και κατασκεύασε το 1822 την «μηχανή των διαφορών». Έντεκα χρόνια αργότερα, πάλι ο ίδιος επινόησε την «αναλυτική μηχανή» για την επεξεργασία μεγάλου όγκου αριθμητικών πληροφοριών. Όμως, η περιορισμένη τεχνολογία εκείνης της εποχής δεν επέτρεψε την πραγματοποίηση και ωφέλιμη χρήση αυτών των μηχανών.

Είναι περίεργο ότι τα επόμενα 100 χρόνια δεν έχουν να παρουσιάσουν σημαντική πρόοδο στον τομέα των υπολογιστών, εκτός από μερικές τεχνολογικές βελτιώσεις όπως η εισαγωγή των διάτρητων δελτίων από τον Hollerich (ιδρυτή της IBM). Στα σχετικά Μαθηματικά όμως, η πρόοδος στο ίδιο χρονικό διάστημα είναι σημαντική. Κυρίως αναπτύσσεται η μαθηματική λογική, η θεωρία των αλγορίθμων, η θεωρία των αυτομάτων και άλλα.

Με την εγκαθίδρυση της βιομηχανικής κοινωνίας, η τεχνολογία φθάνει πλέον σε ένα βαθμό τελειότητας, απαραίτητο για την κατασκευή της πρώτης σύγχρονης υπολογιστικής μηχανής. Ο Howard Aiken με την βοήθεια της IBM παρουσιάζει το 1944 τον Mark I. Ο Mark I χρειάζεται 0,3sec για μία πρόσθεση ή αφαίρεση, 4sec για ένα πολλαπλασιασμό και 10sec για μία διαίρεση. Ένα έτος αργότερα, οι Eckert

και Mauchly κατασκευάζουν τον ENIAC (Electronic Numerical Integrator And Calculator). Αυτός είναι ο πρώτος σύγχρονος Ηλεκτρονικός Υπολογιστής με ηλεκτρονικές λυχνίες, ρευματονόμους κ.λ.π., και είναι 100 φορές ταχύτερος από τον Mark I .

Από τότε παρατηρείται ραγδαία ανάπτυξη των ΗΥ οι οποίοι αρχίζουν πλέον να μπαίνουν στη ζωή του κοινού ανθρώπου. Η ανάπτυξη αυτή χωρίζεται σε διάφορες « εποχές» ή «γενιές» ΗΥ. Η πρώτη γενιά έχει σαν βασικό χαρακτηριστικό της την χρήση της ηλεκτρονικής λυχνίας με αντιπροσώπους τον ENIAC και άλλους. Οι ΗΥ της δεύτερης γενιάς (περί το 1955) χρησιμοποιούν ημιαγωγούς (transistors) και ο χρόνος εκτέλεσης μίας πράξεως είναι της τάξεως των 100μsec . Στην τρίτη γενιά (μετά το 1965) έχουμε τα ολοκληρωμένα κυκλώματα. Ο χρόνος που απαιτείται για την εκτέλεση μίας πράξεως είναι της τάξεως του 1μsec. Χαρακτηριστικό της εξέλιξης αυτής είναι και η συνεχής μείωση του όγκου του ΗΥ. Αυτό είναι αποτέλεσμα της αντικατάστασης των λυχνιών από ημιαγωγούς και στη συνέχεια από ολοκληρωμένα κυκλώματα.

Περί τα τέλη της δεκαετίας του 1970 αρχίζουν και εμφανίζονται οι πρώτοι «προσωπικοί υπολογιστές». Είναι σχετικά φθηνοί και καταλαμβάνουν μικρό όγκο. Η ραγδαία εξέλιξή τους με την παράλληλη μείωση της τιμής τους είχε σαν αποτέλεσμα να φθάσουν οι ΗΥ στο σημείο που είμαστε σήμερα. Ταυτόχρονα, η χρήση άλλων τεχνικών όπως η παράλληλη επεξεργασία, τα κατανεμημένα συστήματα και άλλες επιφέρουν την δραματική αύξηση των δυνατοτήτων των ΗΥ.

Τέλος, ένα σημαντικό βήμα στην ανάπτυξη των ΗΥ ήταν και η επινόηση και χρήση των δικτύων ΗΥ. Από τα τέλη της δεκαετίας του 1950, οι επιστήμονες είχαν αρχίσει να πειραματίζονται με την δικτύωση πολλών ΗΥ. Περί τα μέσα της επόμενης δεκαετίας, ήδη τα δίκτυα ΗΥ χρησιμοποιούνται από αεροπορικές εταιρείες, Τράπεζες, Στρατό και πολλές άλλες επιχειρήσεις. Το αποκορύφωμα αυτής της τεχνολογίας, οδήγησε στην γέννηση του Διαδικτύου (Internet) με τις γνωστές πλέον χρήσεις και δυνατότητές του.

Και που πάμε; Υπάρχει η πιθανότητα οι ΗΥ από υπηρέτες να γίνουν κυρίαρχοι του ανθρώπινου γένους; Πολλοί έχουν υποστηρίξει αυτή την άποψη από άγνοια και θαυμασμό για τους ΗΥ αλλά τέτοιος κίνδυνος δεν υπάρχει. Ο ΗΥ είναι και θα εξακολουθήσει να είναι μια μηχανή. Σήμερα έχει μνήμη και αύριο θα αποκτήσει και άλλες μηχανικές λειτουργίες του ανθρώπινου εγκέφαλου αλλά ποτέ δεν θα αποκτήσει την ικανότητα του ανθρώπου να δημιουργεί και να σκέφτεται ελεύθερα. Υπάρχει όμως ο κίνδυνος να νομίσει ο άνθρωπος ότι μπορεί να τον αντικαταστήσει ένας ΗΥ. Για τον επιστήμονα, ο κίνδυνος αυτός παίρνει συγκεκριμένη μορφή στην επιστημονική έρευνα. Εκεί ο ΗΥ πρέπει να αποτελεί όργανο που βοηθάει στις πράξεις και στον έλεγχο της ορθότητας των ιδεών μας. Σ' ένα πρόβλημα, το μέρος που λύνει ο ΗΥ δεν αποτελεί από μόνο του μια επιστημονική εργασία. Είναι μόνο ένα αναπόσπαστο αλλά επουσιώδες τμήμα του προβλήματος και έτσι μόνο πρέπει να κριθεί σε μια αξιολόγηση μιας εργασίας.

1.2 Το Μέλλον

Όλη αυτή η υπολογιστική ισχύς που είναι διαθέσιμη σήμερα σε κάθε είδος υπολογιστικού συστήματος που μπορεί να είναι ένας σταθμός εργασίας ή ένα φορητό υπολογιστή laptop ή ακόμη και ένα κινητό τηλέφωνο τέταρτης γενιάς σε συνδυασμό με την εξέλιξη των τηλεπικοινωνιών, αλλάζει τον τρόπο που ζούμε, εργαζόμαστε ή ταξιδεύουμε. Δεν είναι μακριά στο μέλλον η εποχή όπου θα μπαίνουμε σε ένα εστιατόριο και θα παραγγέλνουμε σε ένα ψηφιακό σερβιτόρο ενώ παράλληλα θα ζητάμε να ακούσουμε την μουσική της προτίμησής μας διαβάζοντας μια ψηφιακή εφημερίδα, και όλα αυτά κάνοντας χρήση μίας υπολογιστικής (και όχι μόνο) συσκευής στο μέγεθος του σημερινού κινητού τηλεφώνου.

Η νανοτεχνολογία κατασκευάζει ήδη συσκευές ναονομεγέθους (ένα nanometer είναι 40.000 φορές μικρότερο του πάχους μίας ανθρώπινης τρίχας). Είναι ήδη προφανές από όλα αυτά ότι θα επηρεασθεί άμεσα ο τρόπος της ζωής μας από τα φάρμακα που χρησιμοποιούμε, την ενέργεια που χρειαζόμαστε, τα τρόφιμα που καταναλώνουμε, τα κτίρια κ.α.

2. ΕΠΙΚΟΝΩΝΙΑ ΑΝΘΡΩΠΟΥ - ΗΥ

2.1 Λογική Ανάλυση Προβλήματος

Είναι πλέον γνωστό ότι ένας Η/Υ δεν μπορεί να λύσει οτιδήποτε πρόβλημα παρά μπορεί μόνο να κάνει στοιχειώδεις πράξεις και να παίρνει κάποιες μηχανικές αποφάσεις. Και σαν να μην φτάνει αυτό, περιμένει από εμάς (χρήστη) να του πούμε τι πράξεις πρέπει να κάνει και τι κριτήρια να χρησιμοποιήσει.

Τη στιγμή που αρχίζει να δημιουργείται ένα πρόβλημα που μπορεί να λύσει ένας Η/Υ τίθεται αυτόματα το ερώτημα: τι πρέπει να κάνει κανείς χονδρικά για να λύσει το πρόβλημα; Η προεργασία αυτή αποτελεί μια λογική ανάλυση του προβλήματος, καταλήγει δε στην διατύπωση ορισμένων βημάτων που πρέπει να κάνουμε (διαδοχικά ή ταυτόχρονα) για να λύσουμε το πρόβλημα. Η λογική ανάλυση ενός προβλήματος είναι μερικές φορές πολύ δύσκολη και απαιτεί γνώσεις και ευφυΐα που μόνον ο άνθρωπος συνδυάζει. Το αποτέλεσμα της λογικής ανάλυσης ενός προβλήματος, δηλαδή τα απαιτούμενα βήματα για την λύση, δεν είναι μονοσήμαντο για ένα πρόβλημα. Εκτός αυτού διαφορετικά προβλήματα μπορεί να απαιτούν τελείως διαφορετικά βήματα. Για τα προβλήματα εκείνα που απαιτούν την χρήση ενός Η/Υ η λογική υποδεικνύει τα ακόλουθα βήματα:

1 **Φραστική Διατύπωση του Προβλήματος.** Προσπαθούμε να διατυπώσουμε με λόγια το πρόβλημα όσο πιο καλά γίνεται. Καθορίζουμε τα δεδομένα, τον αντικειμενικό σκοπό (δηλ. τι θα θεωρήσουμε απάντηση στο πρόβλημα) και τους γενικούς κανόνες που θα διέπουν την επίλυση του προβλήματος.

2 **Μαθηματική Διατύπωση του Προβλήματος.** Διατυπώνουμε το πρόβλημα με μαθηματική αυστηρότητα και σαφήνεια. Είναι φανερό ότι η επιτυχία αυτού του βήματος εξαρτάται από την ικανότητα μας να κάνουμε συγκεκριμένο κάτι μάλλον αφηρημένο. Εξαρτάται όμως και από το ίδιο το πρόβλημα. Άλλα προβλήματα επιδέχονται εύκολη μαθηματική διατύπωση ή είναι σχεδόν έτοιμα από την αρχή και άλλα είναι από την φύση τους λιγότερο ή περισσότερο ασυμβίβαστα με την μαθηματική γλώσσα.\

3 **Αρχική Επίλυση του Προβλήματος.** Εκθέτουμε το πώς θα λύσουμε το πρόβλημα και εκτελούμε εκείνες τις πράξεις που δεν παρουσιάζουν δυσκολία στην εκτέλεση τους από έναν άνθρωπο.

4 **Αριθμητική Ανάλυση του Προβλήματος.** Χρησιμοποιώντας μεθόδους αριθμητικής ανάλυσης διατυπώνουμε το μέρος εκείνο του προβλήματος, που θα λυθεί από τον Η/Υ έτσι ώστε να μην μένει παρά μόνο η εκτέλεση των πράξεων.

5 Προγραμματισμός και Χρήση του Η/Υ. Στο στάδιο αυτό επιλύουμε το μερικό πρόβλημα (όπως αυτό διατυπώθηκε στο προηγούμενο στάδιο) χρησιμοποιώντας τον Η/Υ. Επειδή το βήμα αυτό ενδιαφέρει περισσότερο, είναι σκόπιμο να αναλυθεί στα εξής χαρακτηριστικά υπό-προβλήματα:

5.1 ***Λογικό Διάγραμμα*** **(ή διάγραμμα ροής ή flowchart ή block diagram).** Καθορίζουμε την σειρά των πράξεων που θα κάνει ο Η/Υ. Η σειρά αυτή περιγράφεται με το λογικό διάγραμμα που δείχνει περιγραφικά την πορεία που θα ακολουθήσει ο Η/Υ. (Μερικές φορές αυτό το στάδιο κρίνεται σκόπιμο είτε να παραληφθεί είτε να αντικατασταθεί από πιο νέες τεχνικές όπως η top-down ή bottom-up σχεδίαση).

5.2 ***Εκλογή Γλώσσας***. Ανάλογα με τη φύση του προβλήματος και τις γνώσεις μας, επιλέγουμε μια γλώσσα (απ' αυτές φυσικά που καταλαβαίνει ο Η/Υ) για να του μεταβιβάσουμε το πρόβλημα.

5.3 ***Προγραμματισμός***. Από το λογικό διάγραμμα γράφουμε στη γλώσσα που διαλέξαμε το τι πρέπει να κάνει ο Η/Υ. Έτσι παίρνουμε το πρόγραμμα δηλ, μια σειρά από δεδομένα, οδηγίες παρατηρήσεις για τον Η/Υ. Αυτό το πρόγραμμα ονομάζεται πηγαίο πρόγραμμα (source program).

5.4 ***Εκτέλεση του Προγράμματος και Έλεγχος.*** Τροφοδοτούμε τον Η/Υ με το πρόγραμμα και ελέγχουμε τα αποτελέσματα που μας δίνει. Αν είναι δυνατόν ελέγχουμε όλες τις δυνατές περιπτώσεις και στη συνέχεια διορθώνουμε ή βελτιώνουμε το πρόγραμμα.

2.2 Αλγόριθμοι

Το πρόβλημα ή το μέρος του προβλήματος που θα δοθεί στον Η/Υ πρέπει να είναι σαφές και να εξελίσσεται σύμφωνα με προκαθορισμένους νόμους (για την εκτέλεση των πράξεων) και κριτήρια (για τη λήψη αποφάσεων από τον Η/Υ). Το σύνολο των οδηγιών που θα δώσουμε στον Η/Υ αποτελεί μια συνταγή, έναν αλγόριθμο όπως λέμε. Ο αλγόριθμος αυτός σε συνδυασμό με τους κανόνες λειτουργίας του Η/Υ θα δώσει ένα αιτιοκρατικά προκαθορισμένο αποτέλεσμα. Αλγόριθμους συναντάμε πολλούς και καθημερινά στην ζωή μας, όπως π.χ. στην συνταγή ενός γιατρού, στο δέσιμο μιας γραβάτας, κλπ.

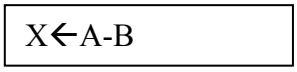
Οι αλγόριθμοι αυτοί όμως είναι συχνά αφελείς αλλά και ατελείς. Χωρίς να δώσουμε μαθηματικό ορισμό του αλγόριθμου, ας αναφερθούμε σε μερικές ιδιότητες που πρέπει να έχει ένας καλός αλγόριθμος:

1 Δεν πρέπει να περιέχει αμφιβολίες, δηλ. σημεία όπου μια απόφαση πρέπει να ληφθεί χωρίς να έχουμε πει το πώς να ληφθεί.
2 Πρέπει να είναι αποτελεσματικός, δηλ. να δίνει το επιδιωκόμενο αποτέλεσμα σε πεπερασμένο χρόνο.
3 Πρέπει να είναι γενικός, δηλ. να εφαρμόζεται σε όσο το δυνατόν περισσότερες παρόμοιες περιπτώσεις.
4 Πρέπει να είναι αποδοτικός, δηλ. να φθάνει στο αποτέλεσμα με τον καλύτερο δυνατόν τρόπο.
5 Τέλος πρέπει να μπορεί να εκτελεσθεί από μια μηχανή.

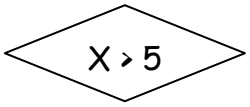
Προφανώς, εμείς θα συναντήσουμε αλγόριθμους που μπορούν να εκτελεσθούν από μια συγκεκριμένη μηχανή, έναν Η/Υ.

Για να περιγράψουμε έναν αλγόριθμο βολικά και παραστατικά για την ανθρώπινη λογική χρησιμοποιούμε το λογικό διάγραμμα. Το λογικό διάγραμμα αποτελεί μια εικόνα του τι θα γίνει μέσα στον Η/Υ και βοηθάει πολύ στο γράψιμο του προγράμματος και αποτελείται από διάφορα απλά γεωμετρικά σχήματα, από τα οποία τα πιο συνηθισμένα είναι τα εξής:

⇒ *Ένα ορθογώνιο παραλληλόγραμμο* : Δηλώνει μια ενδιάμεση πράξη (όχι απόφαση), που περιγράφεται κατάλληλα μέσα στο ορθογώνιο, πχ



⇒ *Ένας ρόμβος* : Δηλώνει μια απόφαση και ανάλογα με τις δυνατές περιπτώσεις που παρουσιάζονται μπορεί να γίνει κάποιο άλλο πολύγωνο, πχ

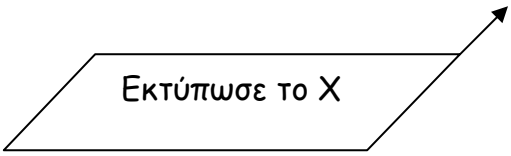


⇒ *Ένα οβάλ* : Δηλώνει την αρχή ή το τέλος του λογικού διαγράμματος και μελλοντικά και του προγράμματος, πχ

⇒ *Ένας μικρός κύκλος* : Ενώνει δύο κομμάτια του λογικού διαγράμματος που αναγκασθήκαμε να χωρίσουμε (π.χ. γιατί το χαρτί δεν μας έπαιρνε).

⇒ *Ένα μακρόστενο εξάπλευρο* : Παριστάνει συνοπτικά μια σχετικά μεγάλη ομάδα πράξεων που δεν θέλουμε να τις δηλώσουμε αναλυτικά στο σημείο αυτό (π.χ. ένα υποπρόγραμμα ή μια διαδικασία).

⇒ *Ένα πλάγιο παραλληλόγραμμο* : Δηλώνει είσοδο δεδομένων ή εξαγωγή αποτελεσμάτων, ανάλογα με την φορά του βέλους, πχ



Φυσικά, υπάρχουν και άλλα τέτοια σχήματα αλλά μάλλον τα παραπάνω είναι από τα πιο βασικά. Η γραμμή που συνδέει τα διάφορα αυτά σχήματα φέρει ένα βέλος κάθε φορά που διακόπτεται από ένα σχήμα. Το βέλος αυτό δείχνει την σειρά διαγραφής του λογικού διαγράμματος που στην ουσία είναι η σειρά εκτέλεσης των οδηγιών από τον Η/Υ.

Εκτός από το λογικά διαγράμματα, ένας άλλος τρόπος περιγραφής ενός αλγόριθμου είναι με την χρήση φυσικής γλώσσας και παράλληλα με συγκεκριμένα βήματα να δίνεται η λεπτομερής περιγραφή του. Αυτή είναι και η μέθοδος που θα ακολουθήσουμε στα παραδείγματα που ακολουθούν.

Παραδείγματα

1. Αν αγοράσουμε ΚΑ κιλά ροδάκινα Βέροιας με Α δρχ. και ΚΒ κιλά ροδάκινα Νάουσας με Β δρχ., ποια είναι τα ακριβότερα; Να γραφεί αλγόριθμος που θα λύνει αυτό το δύσκολο πρόβλημα.

Λύση με χρήση φυσικής γλώσσας:

1. Εισαγωγή των δεδομένων ΚΑ, Α, ΚΒ, Β
2. Χ ← Α / ΚΑ
3. Υ ← Β / ΚΒ
4. Εάν Χ > Υ τότε ακριβότερα είναι της Βέροιας και ο αλγόριθμος τερματίζεται.
5. Εάν Χ < Υ τότε ακριβότερα είναι της Νάουσας και ο αλγόριθμος πάλι τερματίζεται.
6. Εάν Χ = Υ τότε κοστίζουν το ίδιο.

2. Ας υποθέσουμε ότι ο Η/Υ δέχεται 3 αριθμούς Α, Β, C και ζητάμε τον μεγαλύτερο. Ποιος είναι ο αλγόριθμος για το πρόβλημα;

Λύση με χρήση φυσικής γλώσσας:

3. Εισαγωγή των δεδομένων Α, Β, C
4. Εάν Α > Β
    a. τότε MAX ← Α
    b. αλλιώς MAX ← Β
5. Εάν C > MAX
    a. τότε MAX ← C
6. Εκτύπωσε το MAX γιατί είναι το μέγιστο από τα τρία
7. Τέλος

Αξιοσημείωτα είναι τα σημεία όπου δίνουμε στο MAX μία τιμή π.χ. MAX ← Α. Αν και μοιάζει με εξίσωση, η MAX ← Α δεν είναι εξίσωση. Είναι μία συντομογραφία της πράξης που δίνει στο MAX την τιμή του Α. Την ίδια πράξη μπορούμε να την δηλώσουμε και με το σύμβολο της ισότητας, π.χ. MAX = Α, δηλ. η τιμή της μεταβλητής Α αποδίδεται στο MAX.

3. Κάθε σπουδαστής εξετάζεται σε 5 μαθήματα. Για να περάσει το εξάμηνο πρέπει να περάσει τουλάχιστον 4 με βαθμό 5 ή μεγαλύτερο στο καθένα και να συγκεντρώσει τουλάχιστον μέσο όρο 6. Να γίνει αλγόριθμος που θα λύνει αυτό το πρόβλημα.

Λύση με χρήση φυσικής γλώσσας:

4. Εισαγωγή των πέντε βαθμών, Α, Β, C, D, Ε
5. Έστω Ρ ← 0 (έστω ότι δεν πέρασε κανένα μάθημα – Αρχική τιμή)
6. Έστω ΜΟ ← 0 (έστω ότι έχει μέσο όρο μηδέν – Αρχική τιμή)
7. Εάν Α > 5 τότε Ρ ← Ρ +1
8. Εάν Β > 5 τότε Ρ ← Ρ +1
9. Εάν C > 5 τότε Ρ ← Ρ +1
10. Εάν D > 5 τότε Ρ ← Ρ +1
11. Εάν Ε > 5 τότε Ρ ← Ρ +1
12. ΜΟ ← (Α+Β+C+D+Ε) / 5
13. Εάν (Ρ > 3 και ΜΟ > 5)

a. τότε Πέρασε
b. αλλιώς Έμεινε

14. Τέλος Αλγόριθμου

2.3 Γλώσσες Προγραμματισμού

Το πρόβλημα της επικοινωνίας του ανθρώπου με τον Η/Υ ακόμα δεν έχει λυθεί εντελώς. Αυτό που μένει είναι το πώς θα συνεννοηθούμε μαζί του για να του δώσουμε δεδομένα και οδηγίες για την επίλυση κάποιου προβλήματος, π.χ. να μας λύσει μια εξίσωση ή ένα λογιστικό πρόβλημα, και όχι πλέον για το π.χ. πώς θα μεταφέρουμε πληροφορίες στο δίσκο ή πώς θα αντιγράψουμε ένα αρχείο. Δηλαδή, μιλάμε για εργασίες που δεν ανήκουν στη σφαίρα ειδικότητάς του Λειτουργικού Συστήματος (ΛΣ). Αλλά ο Η/Υ -ο ίδιος ο Η/Υ και όχι το ΛΣ- δεν μιλάει ούτε Ελληνικά ούτε Αγγλικά ούτε και καμία από τις γνωστές ανθρώπινες γλώσσες. Πρέπει δηλ. να του δώσουμε να καταλάβει τι θέλουμε να μας κάνει και όταν τελειώσει την εργασία του πρέπει να μας τη δώσει σε κατανοητή μορφή.

Αυτό το πρόβλημα μπορεί να λυθεί με δύο τρόπους. Ή ο άνθρωπος να μάθει τη γλώσσα του Η/Υ ή αυτός να μάθει την δικιά μας. Σαν αποτέλεσμα του προηγούμενου ήταν να δημιουργηθούν δύο κατηγορίες γλωσσών προγραμματισμού : τις γλώσσες που είναι προσανατολισμένες προς την μηχανή και τις γλώσσες που είναι προσανατολισμένες προς τον άνθρωπο.

Βασικά ο Η/Υ καταλαβαίνει μία και μοναδική γλώσσα, την γλώσσα μηχανής (machine language). Εδώ, οι οδηγίες και τα δεδομένα δίνονται στον Η/Υ με την μορφή δυαδικών ψηφίων (κώδικες επικοινωνίας). Αυτήν η γλώσσα είναι αμέσως κατανοητή από τον Η/Υ αλλά πολύ δύσκολη στην εκμάθησή της από τον άνθρωπο (αρκεί να φανταστούμε ότι το αντίστοιχο της διαταγής PRINT σε κώδικα ASCII είναι, 10100001010010100100110011101010100).

Γι' αυτόν τον λόγο έχουμε επινοήσει παραλλαγές αυτής της γλώσσας κωδικοποιώντας και αντιπροσωπεύοντας ένα μέρος της (π.χ. τις εντολές) με συνδυασμούς γραμμάτων και αριθμών που απομνημονεύονται πιο εύκολα. Ακόμη μπορούμε να παραστήσουμε συμβολικά το μέρος της μνήμης που θα χρησιμοποιηθεί για το πρόβλημα και έτσι προκύπτει μια άλλη γλώσσα η -έτσι λέγεται- assembly. Για να γίνει κατανοητή μια τέτοια γλώσσα από τον Η/Υ πρέπει πρώτα να μεταφρασθεί σε γλώσσα μηχανής. Η μετάφραση γίνεται από τον ίδιο τον Η/Υ με ένα πρόγραμμα που λέγεται assembler. Αυτού του τύπου οι γλώσσες ανήκουν στη κατηγορία των προσανατολισμένων προς την μηχανή γιατί στην δημιουργία τους λαμβάνεται υπ' όψιν η μηχανή και αγνοείται το πρόβλημα. Αυτό έχει σαν συνέπεια την ύπαρξη πολλών μειονεκτημάτων σ' αυτές τις γλώσσες. Είναι δύσκολες, έχουν στην δομή τους πολλές αιτίες για λάθη και είναι προσανατολισμένες μόνο σε μία κατηγορία Η/Υ (για την ακρίβεια μικροεπεξεργαστών). Σαν πλεονέκτημα μπορεί να θεωρηθεί η μεγάλη ταχύτητα εκτέλεσης προγραμμάτων που είναι γραμμένα σ΄ αυτές τις γλώσσες.

Οι προσανατολισμένες προς τον άνθρωπο - ή πρόβλημα - γλώσσες έχουν δημιουργηθεί με σκοπό να ξεπερασθούν αυτά τα μειονεκτήματα. Η δομή και η λογική τους μοιάζει με αυτήν της φυσικής γλώσσας των αλγορίθμων. Στην γραφή τους χρησιμοποιούνται τα γράμματα της Αγγλικής γλώσσας, οι αριθμοί καθώς και μερικά

σύμβολα όπως τα * - + / ( ) κ.α. Το σύνολο των οδηγιών που δίνεται στον Η/Υ σε μια τέτοια γλώσσα αποτελεί το πηγαίο πρόγραμμα (source program). Φυσικά, για την κάθε μία απ΄ αυτές τις γλώσσες ο Η/Υ χρειάζεται ένα μεταφραστικό πρόγραμμα (compiler ή interpreter) για να μεταφράσει το πηγαίο πρόγραμμα σε γλώσσα μηχανής. Έτσι προκύπτει το αντικείμενο πρόγραμμα (object program). Το αντικείμενο πρόγραμμα με την βοήθεια του ΛΣ (Loader) θα μετατραπεί σε εκτελέσιμο πρόγραμμα (executable program) και θα είναι πλέον έτοιμο να εκτελεσθεί από τον Η/Υ.

Μερικές από τις περισσότερες γνωστές γλώσσες προγραμματισμού, τις προσανατολισμένες προς τον άνθρωπο -με χρονολογική σειρά εμφάνισης- είναι η FORTRAN, η COBOL, η ALGOL, η BASIC, η PASCAL, η C, η ADA, η C++ και πολλές άλλες.

Η **FORTRAN** (από το FORmula TRANslation = μετάφραση τύπων) δημιουργήθηκε το 1955 από την IBM για την επίλυση επιστημονικών και τεχνικών προβλημάτων. Βασικά της πλεονεκτήματα είναι η εύκολη διατύπωση μαθηματικών εκφράσεων και το μικρό πλήθος εντολών. Από το 1955 η FORTRAN βελτιώθηκε πολλές φορές με αποτέλεσμα να έχουμε τις εκδόσεις FORTRAN IV, V, 66 όπως και τις πιο πρόσφατες FORTRAN 77, FORTRAN 88 , Parallel FORTRAN κ.α.

Η **COBOL** (από το COmmon Business Oriented Language = γλώσσα προσανατολισμένη στο οικονομικοεμπορικό τομέα) δημιουργήθηκε στα τέλη της δεκαετίας του 50 και προορίζεται για οικονομικά και εμπορικά προβλήματα. Αυτά τα προβλήματα παρουσιάζουν μεγάλο όγκο δεδομένων και σχετικά μικρή επεξεργασία (αντίθετα με τα επιστημονικά ή τεχνικά). Χαρακτηριστικό της γλώσσας είναι το πάρα πολύ πλούσιο λεξιλόγιό της.

Η **ALGOL** (από το ALGOrithmic Language = αλγοριθμική γλώσσα), παρουσιάσθηκε το 1958 σαν αποτέλεσμα μιας διεθνούς συνεργασίας αλλά έχει υποστεί πολλές βελτιώσεις από τότε με επικρατέστερη την έκδοση ALGOL-68. Προορίζεται για επιστημονικοτεχνικά προβλήματα. Έχει σχεδόν όλες τις ιδιότητες της FORTRAN και μεγάλη ευελιξία στη χρήση της. Οι νέες δομές που πρωτοχρησιμοποιήθηκαν στην ALGOL, είχαν σαν αποτέλεσμα να αλλάξουν τα πρότυπα των γλωσσών και να δημιουργηθούν νέα δεδομένα στη φιλοσοφία τους. Η ALGOL θεωρείται ο πρόγονος σχεδόν όλων των σύγχρονων γλωσσών προγραμματισμού όπως η Pascal ή η C .

Η **BASIC** άρχισε να αναπτύσσεται το 1964 στο Dartmouth College των ΗΠΑ από τους J. Kemeny και T. Kurtz. Αρχικά η γλώσσα προορίζονταν καθαρά για εκπαιδευτικούς σκοπούς, γι' αυτό και ονομάσθηκε Beginner's All purpose Symbolic Instruction Code. Σιγά σιγά όμως αναπτύχθηκε και συμπληρώθηκε σε σημείο που να θεωρείται σήμερα σαν μια αρκετά καλή γλώσσα κυρίως για αρχάριους αλλά και για σοβαρές εφαρμογές. Κύρια χαρακτηριστικά της είναι η υπερβολικά απλή της σύνταξη και το επίσης απλό της λεξιλόγιο. Γενικά, μπορεί να χαρακτηρισθεί σαν ένα υποσύνολο της FORTRAN.

Η **PASCAL** οφείλει το όνομά της στο Γάλλο μαθηματικό Blaise Pascal (1623 - 1662). Δημιουργήθηκε από το N. Wirth το 1971 παίρνοντας πολλά στοιχεία από την ALGOL. Σήμερα θεωρείται η καθεαυτό εκπαιδευτική γλώσσα στην εκπαίδευση της πληροφορικής αλλά έχει να επιδείξει και σημαντικές εμπορικές εφαρμογές.

Η **C**, μάλλον η πιο δημοφιλής γλώσσα του σήμερα, αν και σχεδιάστηκε (από τον D. Ritcie για γλώσσα συστημάτων (το 1972), ανήκει και αυτή στις λεγόμενες ALGOL - like (σαν την ALGOL) γλώσσες, όπως και η Pascal . Βασικά χαρακτηριστικά της το πολύ μικρό αλλά δυναμικό λεξιλόγιο της και οι ισχυρές δομές που υποστηρίζει. Σήμερα, περίπου το 80% του σύγχρονου software είναι γραμμένο σε C. Είναι δε και η φυσική γλώσσα του UNIX.

Η **ADA** (προς τιμή της Augusta Ada Byron (1815 - 1851) που ήταν βοηθός του C. Babbage και η κόρη του γνωστού μας ποιητή Λόρδου Βύρωνα) είναι πιο νέα γλώσσα. Δημιουργήθηκε το 1979 για τις ανάγκες του Υπουργείου Άμυνας των ΗΠΑ. Φυσικά, σήμερα χρησιμοποιείται σε αρκετά ευρύ επίπεδο. Επειδή είναι νέα γλώσσα, αυτή τη στιγμή είναι ακόμη στη φάση της κριτικής, μια φάση που όλες οι γλώσσες έχουν περάσει. Βασικά σχεδιάστηκε για μεγάλα συστήματα ΗΥ και πολλοί επιστήμονες της πληροφορικής διαφωνούν ως προς το αν μπορεί να χρησιμοποιηθεί σε μικρότερους ΗΥ.

Η **C++** αποτελεί ένα υπερσύνολο της C και οι επιπλέον δυνατότητες της γλώσσας αυτής είναι ότι επιτρέπει αντικειμενοστραφή (object oriented) προγραμματισμό.

Η **Java**, μία πολύ σύγχρονη αντικειμενοστραφής γλώσσα προγραμματισμού, αποτελεί την πιο δημοφιλή γλώσσα για προγραμματισμό στο Διαδίκτυο. Είναι μία δυναμική και συνεχώς αναπτυσσόμενη και βελτιούμενη γλώσσα.

Εκτός από τις παραπάνω υπάρχει μια πληθώρα ακόμη γλωσσών προγραμματισμού οι οποίες είναι λιγότερο ή περισσότερο γνωστές. Οι πιο πολλές από αυτές όμως, είναι εξειδικευμένες σε κάποιο συγκεκριμένο τομέα όπως λ.χ. η **Prolog** σαν γλώσσα τεχνητής νοημοσύνης.

2.4 Μεταφραστικά Προγράμματα

Όπως προαναφέρθηκε υπάρχουν δύο ειδών μεταφραστικά προγράμματα:

⇒ οι μεταφραστές τύπου ***compiler***, και
⇒ οι μεταφραστές τύπου ***interpreter***.

Η διαφορά τους κυρίως έγκειται στον τρόπο που μεταφράζουν το πρόγραμμα σε γλώσσα μηχανής. Οι compilers δημιουργούν το αντικείμενο πρόγραμμα το οποίο αν είναι απαλλαγμένο από λάθη δεν χρειάζεται να ξαναμεταφρασθεί όταν θα θελήσουμε να το ξανατρέξουμε. Αυτό δίνει μεγάλη ταχύτητα εκτέλεσης του προγράμματος αλλά και μερικές δυσκολίες ως προς την διόρθωση. Από την άλλη μεριά, οι interpreters μεταφράζουν την κάθε γραμμή του προγράμματος ξεχωριστά, και εάν δεν ανιχνεύσουν λάθη την δίνουν στον Η/Υ για να την εκτελέσει χωρίς να την αποθηκεύουν μεταφρασμένη με συνέπεια, εκτέλεση για δεύτερη φορά της ίδιας γραμμής να συνεπάγεται μετάφραση από την αρχή. Στην συνέχεια παίρνουν την επόμενη γραμμή του προγράμματος και επαναλαμβάνουν την ίδια διαδικασία μέχρι το τέλος του προγράμματος. Άρα το βασικό μειονέκτημα των interpreters είναι το ότι είναι πολύ αργοί αλλά σαν πλεονέκτημα μπορεί να χαρακτηρισθεί η ευκολία διόρθωσης λαθών στο πρόγραμμα.

3. ΕΙΣΑΓΩΓΗ ΣΤΗ C

3.1 Πλεονεκτήματα της C

Η C είναι μία σχετικά μικρή γλώσσα. Το λεξιλόγιό της απαρτίζεται από πολύ λίγες λέξεις (ονομάζονται δεσμευμένες λέξεις), και θεωρείται σαν μια πανίσχυρη γλώσσα προγραμματισμού. Αυτή τη δύναμή της η C την αντλεί από τις τέλεια δομημένες εντολές ελέγχου της όσο και από τους τύπους δεδομένων της. Μερικά από τα πλεονεκτήματα της είναι τα ακόλουθα:

⇒ Η C είναι η φυσική γλώσσα του λειτουργικού συστήματος UNIX το οποίο είναι το σημαντικότερο λειτουργικό σύστημα σήμερα (μαζί με τους κλώνους τους όπως SOLARIS, LINUX κλπ).

⇒ Η C είναι συμβατή με όλες τις υπολογιστικές μηχανές. Ένα πρόγραμμα C μπορεί πολύ εύκολο και με λίγο κόπο να μετατραπεί (αν είναι απαραίτητο) ώστε να τρέξει σε μία διαφορετική μηχανή από αυτή που πρωτοκατασκευάστηκε.

⇒ Η C είναι λακωνική. Σε μία της πρόταση είναι δυνατόν να συνδυαστούν πολλές διαδικασίες που σε άλλες γλώσσες θα απαιτούσαν πολλές προτάσεις. Βέβαια αυτό είναι και πλεονέκτημα και μειονέκτημα ανάλογα με τον προγραμματιστή αλλά σε γενικές γραμμές αυτή η κομψότητα της γλώσσας αποτελεί ένα από τα ισχυρά χαρακτηριστικά της.

⇒ Τέλος, η C είναι από την φύση της μία αρθρωτή γλώσσα. Κάθε πρόγραμμα μπορεί να αποτελείται από πολλές επί μέρους συναρτήσεις όπως και από πολλά ανεξάρτητα εξωτερικά προγράμματα. Αυτό κάνει εύκολο τον δομημένο προγραμματισμό όπως επίσης δίνει δυνατότητες χρήσης έτοιμου κώδικα. Εξάλλου η επαναχρησιμοποίηση του κώδικα (program reusability) είναι μια έννοια που ξεπήδησε από την C.

3.2 Προετοιμασία Προγραμματισμού

Ο προγραμματισμός Η/Υ είναι η διαδικασία μεταφοράς αλγορίθμων σε μία γλώσσα κατανοητή στον Η/Υ. Ένας αλγόριθμος αποτελείται από αυστηρά καθορισμένα βήματα. Ο κάθε άνθρωπος χρησιμοποιεί αλγόριθμους στην καθημερινή του ζωή όπως για παράδειγμα σε μία τραπεζική συναλλαγή ή σε μία μαγειρική συνταγή. Η κύρια διαφορά με τους αλγόριθμους που απευθύνονται σε Η/Υ είναι ότι ο Η/Υ έχει μηδενικές ανοχές σε ότι αφορά πιθανά λάθη του αλγορίθμου ή ασάφειες. Ο άνθρωπος παίρνει πρωτοβουλίες και ξεπερνάει συνήθως τέτοια εμπόδια. Ο Η/Υ ποτέ! Γενικά η διαδικασία επίλυσης ενός προβλήματος με την χρήση ενός Η/Υ και φυσικά κάποιας γλώσσας προγραμματισμού μπορεί να συνοψισθεί στα ακόλουθα βήματα:

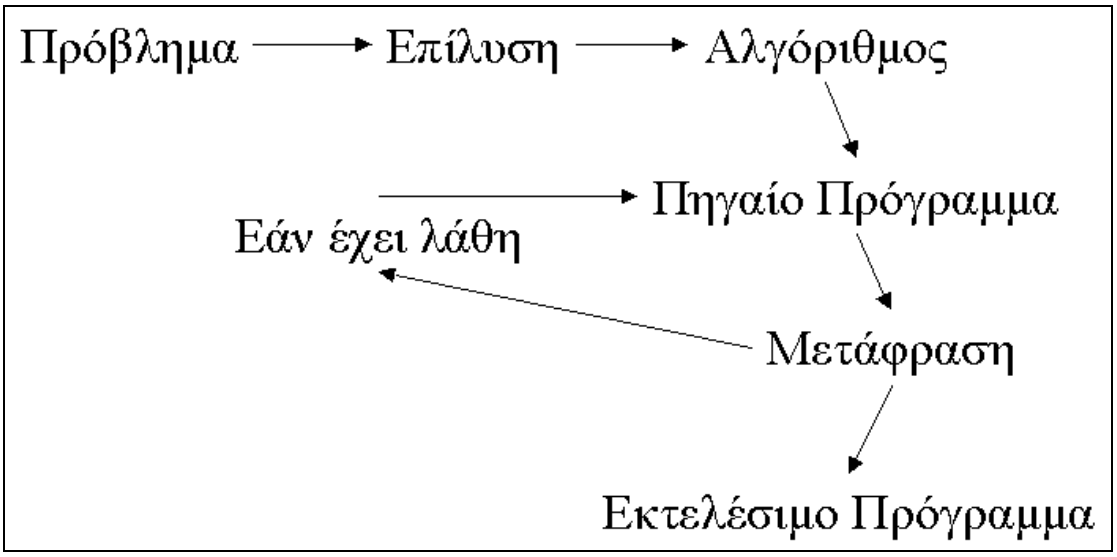
1 Συγκεκριμενοποίηση του προβλήματος,
2 Εύρεση του αλγόριθμου επίλυσης,
3 Μεταφορά του αλγορίθμου σε πρόγραμμα (στην προκείμενη περίπτωση C),
4 Έλεγχος του προγράμματος, και τέλος
5 Χρήση του προγράμματος.

6 Βέβαια η μεταφορά του αλγορίθμου επίλυσης σε πρόγραμμα μπορεί να αναλυθεί ακόμη πιο πολύ, ως εξής:

6.1 Γράψιμο του προγράμματος. Εδώ απαιτείται κάποιος κειμενογράφος. Βέβαια μια γλώσσα προγραμματισμού δεν έχει τις απαιτήσεις μιας ανθρώπινης γλώσσας και κατά συνέπεια ένας επεξεργαστής κειμένου όπως ο Microsoft Word είναι λίγο πολύ άχρηστος. Ένας απλός κειμενογράφος όπως το text pad των Microsoft Windows ή ένας text editor γενικότερα είναι υπεραρκετός. Το αρχείο προγράμματος που θα δημιουργηθεί ονομάζεται **πηγαίος κώδικας** (source code)

6.2 Το επόμενο βήμα είναι η μετάφραση του πηγαίου κώδικα σε γλώσσα μηχανής. Αυτό θα γίνει με την βοήθεια κάποιου μεταφραστικού προγράμματος. Το αποτέλεσμα είναι να δημιουργηθεί ο **εκτελέσιμος κώδικας** (executable code). Αυτό είναι και το αρχείο που τρέχει. Βέβαια αυτό μπορεί να περιέχει λάθη. Σε τέτοια περίπτωση ο προγραμματιστής πρέπει να επανέλθει στον πηγαίο κώδικα για να τα διορθώσει και μετά να ξαναμεταφράσει και ούτω καθεξής μέχρι να βεβαιωθεί ότι το πρόγραμμα δεν περιέχει πλέον λάθη.

Γενικά η κρίσιμη αυτή ακολουθία μπορεί να παρασταθεί στο *Σχήμα 1*.



Σχήμα 1. Η κρίσιμη ακολουθία

3.3 Το Πρώτο Πρόγραμμα

Το παρακάτω πρόγραμμα (το πρώτο πρόγραμμα σε C) εκτυπώνει στην οθόνη του Η/Υ ένα μήνυμα, το κλασσικό πρώτο πρόγραμμα της C, «Γεια σου Κόσμε!». Ανάλογα με το περιβάλλον που θα τρέξει το πρόγραμμα είναι δυνατόν τα μηνύματα να είναι και στα Ελληνικά αλλά αυτό πρέπει να δοκιμασθεί πρώτα (στη συνέχεια, τα μηνύματα θα είναι πάντα με λατινικούς χαρακτήρες). Η αρίθμηση υπάρχει για την ευκολία αναφοράς σε συγκεκριμένα σημεία του προγράμματος και μόνο.

```
1. #include <stdio.h>
2. int main() {
```

```
3.      printf("Γειά σου Κόσμε!\n");
4.      return 0;
5.  }
```

Το παραπάνω πρόγραμμα ας υποτεθεί ότι έχει γραφεί με την βοήθεια κάποιου κειμενογράφου και έχει αποθηκευτεί χρησιμοποιώντας με κάποιο όνομα χρησιμοποιώντας (πάντα) την προέκταση .c , έστω Hello.c Η μεταγλώττιση του προγράμματος γίνεται με την εντολή

cc Hello.c ή
gcc Hello.c

ανάλογα με τον διαθέσιμο μεταγλωττιστή (compiler). (Στο παρών βιβλίο όλες οι εντολές αφορούν λειτουργικά συστήματα τύπου UNIX.) Εάν δεν παρουσιασθούν λάθη κατά την διάρκεια της μεταγλώττισης, τότε ο μεταφραστής θα δημιουργήσει το εκτελέσιμο πρόγραμμα που θα έχει το όνομα a.out. Για να εκτελεσθεί (τρέξει) αυτό το πρόγραμμα αρκεί να πληκτρολογηθεί η διαταγή

a.out

και το αποτέλεσμα θα είναι να εμφανισθεί στην οθόνη του Η/Υ το μήνυμα:

Γειά σου Κόσμε!

Παρατηρήσεις στο πρώτο πρόγραμμα

Αρχικά σε ένα πρόγραμμα πρέπει να δηλωθεί ποιες από τις βιβλιοθήκες θα ενσωματωθούν με το κυρίως πρόγραμμα (εάν είναι απαραίτητες). Αυτό γίνεται με την οδηγία προς τον προεπεξεργαστή της C με την εντολή:

#include <όνομα βιβλιοθήκης.h>

Η βιβλιοθήκη **stdio.h** (STandard Input Output) παρέχει το πρόγραμμα με βοηθητικές συναρτήσεις που συσχετίζονται με τις ροές εισόδου / εξόδου. Η ροή εισόδου σχετίζεται με τι θα χρειαστεί ένα πρόγραμμα από τον έξω κόσμο (π.χ. πληκτρολόγιο) και η ροή εξόδου με το τι θα δείξει ένα πρόγραμμα στον έξω κόσμο (π.χ. τι θα εκτυπώσει και πού). Η προέκταση **.h** υποδηλώνει ότι πρόκειται για αρχείο βιβλιοθήκη (header file).

Κάθε πρόγραμμα πρέπει να περιέχει μία συνάρτηση που ονομάζεται **main( )** και ουσιαστικά από εκεί αρχίζει και η εκτέλεση του προγράμματος. Οι παρενθέσεις που ακολουθούν την main υποδηλώνουν ότι αυτή η συνάρτηση δεν έχει παραμέτρους (βέβαια μπορεί και να δέχεται παραμέτρους). Η λέξη **int** στην αρχή δηλώνει ότι η συνάρτηση **main( )** θα επιστρέψει έναν ακέραιο αριθμό (αυτό που στα μαθηματικά ονομάζεται πεδίο τιμών συνάρτησης ενώ μέσα στις παρενθέσεις δίνεται το πεδίο ορισμού).

Τα άγκιστρα { και } δείχνουν που αρχίζει και που τελειώνει η συνάρτηση. Επίσης δείχνουν την αρχή και το τέλος ενός σύνολο εντολών όπως θα φανεί σε επόμενα κεφάλαια.

Η **printf( )** αποτελεί μία συνάρτηση εκτύπωσης (ροή εξόδου) και μάλιστα με δυνατότητες μορφοποίησης. Εμπεριέχεται στην βιβλιοθήκη **stdio.h**. Θα εκτυπώσει ότι υπάρχει μέσα στα εισαγωγικά εκτός από τους χαρακτήρες που ακολουθούν μία ανάποδη κάθετο (πχ \n). Αυτοί οι χαρακτήρες δίνουν κάποιες οδηγίες που αφορούν την εκτύπωση όπως για παράδειγμα ο n που αναγκάζει την **printf( )** να προχωρήσει σε επόμενη νέα γραμμή.

Η εντολή **return** δηλώνει τι θα πρέπει να επιστρέψει μία συνάρτηση. Στην προκείμενη περίπτωση τίποτα δηλαδή μηδέν (και αυτή η εντολή εάν δεν δηλωθεί η main σαν int δεν χρειάζεται να γραφεί).

Παράδειγμα:

Να γραφεί ένα πρόγραμμα που να εκτυπώνει το παραπάνω μήνυμα αλλά σε 3 διαφορετικές γραμμές.

```
#include <stdio.h>
int main()
 {
        printf("\nΓειά \nσου \nΚόσμε!\n");
        return 0;
}
```

Το πρόγραμμα θα μπορούσε να είχε και γραφεί και ως εξής (με ακριβώς ίδιο αποτέλεσμα:

```
#include <stdio.h>
int main()
{
        printf("\nΓειά ");
        printf("\nσου "):
        printf(\nΚόσμε!\n");
        return 0 ;
}
```

3.4 Ασκήσεις

1. Να γραφεί πρόγραμμα το οποίο θα εμφανίζει στην οθόνη το ακόλουθο μήνυμα «Αυτήν είναι η πρώτη μου προσπάθεια με την C”.

2. Να γραφεί πρόγραμμα το οποίο θα εμφανίζει στην οθόνη τα στοιχεία σας με την ακόλουθη μορφή :

Επώνυμο Όνομα
Διεύθυνση
ΤΚ, Πόλη

3. Να γραφεί πρόγραμμα το οποίο θα εμφανίζει στην οθόνη το ακόλουθο σχήμα :

```
**********
*        *
*        *
**********
```

4. Να γραφεί πρόγραμμα το οποίο θα εμφανίζει στην οθόνη το παρακάτω μενού επιλογών :
   1. Εισαγωγή στοιχείων
   2. Διόρθωση στοιχείων
   3. Διαγραφή στοιχείων
   4. Έξοδος

.

5. Να γραφεί πρόγραμμα το οποίο θα εμφανίζει στην οθόνη το ακόλουθο μήνυμα :

ΤΕΙ Λάρισας

Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Τεχνολογίας Πληροφορικής και Τηλεπικοινωνιών

6. Να γραφεί πρόγραμμα το οποίο θα εμφανίζει στην οθόνη τα ακόλουθα στοιχεία :

Διαμέρισμα	Κοινόχρηστα
ΙΣ1	25 €
ΙΣ2	19 €
Α1	10 €
Α2	35 €
Β1	23 €
Β2	42 €

4. ΜΕΤΑΒΛΗΤΕΣ, ΣΤΑΘΕΡΕΣ, ΤΕΛΕΣΤΕΣ & ΠΑΡΑΣΤΑΣΕΙΣ

Οι τελεστές χειρίζονται μεταβλητές και σταθερές και σχηματίζουν παραστάσεις. Αυτά τα τέσσερα στοιχεία – μεταβλητές, σταθερές, τελεστές και παραστάσεις – αποτελούν το υπόστρωμα της γλώσσας C.

4.1 Αναγνωριστικά

Η γλώσσα C ορίζει τα αναγνωριστικά (identifiers) σαν ονόματα που χρησιμοποιούνται για να γίνεται αναφορά σε μεταβλητές, συναρτήσεις, ετικέτες και σε διάφορα άλλα πράγματα που ορίζονται από το χρήστη. Στη C ένα αναγνωριστικό μπορεί να ποικίλλει από ένα χαρακτήρα μέχρι πολλούς χαρακτήρες. Ο πρώτος χαρακτήρας πρέπει να είναι γράμμα ή το σύμβολο της υπογράμμισης ενώ οι επόμενοι χαρακτήρες μπορούν να είναι γράμματα, αριθμοί ή το σύμβολο της υπογράμμισης.

Η C κάνει διάκριση μεταξύ κεφαλαίων και μικρών γραμμάτων. Για παράδειγμα, τα **grade** και **GRADE** είναι δύο διαφορετικά αναγνωριστικά. Ένα αναγνωριστικό δεν μπορεί να είναι το ίδιο με κάποια λέξη κλειδί της C, και δεν πρέπει να έχει το ίδιο όνομα με τις συναρτήσεις που έχουμε γράψει είτε υπάρχουν στην βιβλιοθήκη της C.

Παρακάτω αναφέρονται ορισμένα παραδείγματα σωστών και λάθος αναγνωριστικών.

Σωστό	**Λάθος**
counter	3counter
temp12	exp!new
long_term	long…term

4.2 Τύποι Δεδομένων

Όλες οι μεταβλητές στην C πρέπει να δηλώνονται πριν χρησιμοποιηθούν. Αυτό χρειάζεται γιατί η C πρέπει να γνωρίζει τον τύπο δεδομένων μιας μεταβλητής για να μπορέσει να μεταγλωττίσει σωστά τις εντολές που περιέχουν τη συγκεκριμένη μεταβλητή. Στην C, υπάρχουν τέσσερις βασικοί τύποι δεδομένων : *χαρακτήρας*, *ακέραιος*, *κινητής υποδιαστολής* και *κινητής υποδιαστολής χωρίς ακρίβεια*. Οι λέξεις κλειδιά που χρησιμοποιούνται για να δηλώσουν μεταβλητές αυτών των τύπων είναι οι **char**, **int**, **float** και **double**.

Οι μεταβλητές τύπου **char** χρησιμοποιούνται για να κρατάνε χαρακτήρες ASCII σαν τους **α**, **β**, **γ**. Οι μεταβλητές τύπου **int** μπορούν να περιέχουν ακέραιες ποσότητες που δεν έχουν ανάγκη από κλασματικό μέρος.

Οι μεταβλητές τύπου **float** και **double** περιλαμβάνονται στα προγράμματα είτε όταν απαιτούνται κλασματικά μέρη είτε όταν η εφαρμογή χρειάζεται πολύ μεγάλους αριθμούς. Η διαφορά ανάμεσα σε μια μεταβλητή **float** και σε μια μεταβλητή **double**

βρίσκεται στο μέγεθος του μεγαλύτερου (και του μικρότερου) αριθμού που μπορούν να κρατάνε. Στον παρακάτω πίνακα δίνεται το ελάχιστο μεγεθος και το εύρος των βασικών τύπων δεδομένων της C.

Τύπος	Ελάχιστο πλάτος σε bit	Εύρος
char	8	0 έως 255
int	16	-32768 έως 32767
float	32	3.4E -38 έως 3.4E + 38
double	64	1.7E – 308 έως 1.7E + 308

4.3 Τροποποιητές Τύπων

Οι βασικοί τύποι δεδομένων μπορεί να έχουν μπροστά τους διάφορους τροποποιητές (modifiers). Ο τροποποιητής χρησιμοποιείται για να αλλάζει τη σημασία του βασικού τύπου ώστε να ικανοποιεί καλύτερα της ανάγκες διάφορων καταστάσεων. Παρακάτω αναφέρεται μια λίστα των τροποποιητών :

signed	προσημασμένος
unsigned	απρόσημος
long	μεγάλος
short	μικρός

Μπορούμε να χρησιμοποιούμε τους τροποποιητές **signed**, **unsigned**, **long** και **short** μπροστά από τους βασικούς τύπους **char** και **int**. Μπορούμε πάντως να χρησιμοποιούμε και το **long** μπροστά από το **double**. Παρ' όλο που επιτρέπεται, η χρήση του **signed** στους ακέραιους είναι περιττή, γιατί η εξ' ορισμού δήλωση του ακεραίου αφορά έναν προσημασμένο αριθμό.

Στον παρακάτω πίνακα παρουσιάζονται όλοι οι επιτρεπτεί συνδιασμοί των βασικών τύπων με τους τροποποιητές τύπων.

Τύπος	Ελάχιστο πλάτος σε bit	Εύρος
char	8	-128 έως 127
unsigned char	8	0 έως 255
signed char	8	-128 έως 127
int	16	-32768 έως 32767
unsigned int	16	0 έως 65535
signed int	16	-32768 έως 32767
short int	16	-32768 έως 32767
unsigned short int	16	0 έως 65535
signed short int	16	-32768 έως 32767
long int	32	-2147483648 έως 2147483649
signed long int	32	-2147483648 έως 2147483649
unsigned long int	32	0 έως 4294967296
float	32	3.4^{E} -38 έως 3.4E +38
double	64	1.7^{E} – 308 έως 1.7E + 308
long double	64	3.4^{E} – 4932 έως 1.1E + 4932

4.4 Δήλωση Μεταβλητών

Η γενική μορφή της εντολής που δηλώνει μία μεταβλητή είναι :
τύπος λίστα_μεταβλητών;
Ο *τύπος* πρέπει να είναι ένας αποδεκτός τύπος δεδομένων της C, ενώ η *λίστα_μεταβλητών* μπορεί να αποτελείται από ένα ή περισσότερα αναγνωριστικά χωρισμένα με κόμματα.

Παράδειγμα:

```
#include <stdio.h>
int main()
{
        int a, b, c;
        float balance;

        return 0;
}
```

4.5 Αρχικές Τιμές Μεταβλητών

Στην C, μπορούμε να δίνουμε τιμές στις περισσότερες μεταβλητές τη στιγμή που τις δηλώνουμε. Η γενική μορφή της απόδοσης αρχικής τιμής είναι :

τύπος όνομα_μεταβλητής = σταθερά;

Το βασικό πλεονέκτημα της απόδοσης αρχικών τιμών στις μεταβλητές είναι ότι μειώνεται ελαφρώς η ποσότητα του κώδικα στο πρόγραμμα.

Παράδειγμα:

```
#include <stdio.h>
int main()
{
        int i = 0;

        return 0;
}
```

4.6 Οι Λέξεις – Κλειδιά της C

Όπως όλες οι γλώσσες προγραμματισμού, έτσι και η C αποτελείται από λέξεις – κλειδιά και από συντακτικούς κανόνες οι οποίοι εφαρμόζονται σε κάθε λέξη – κλειδί. Μια λέξη – κλειδί (keyword) είναι ουσιαστικά μια διαταγή και, σε ένα μεγάλο βαθμό, οι λέξεις – κλειδιά μιας γλώσσας καθορίζουν αυτά που μπορούν να γίνουν και τον τρόπο με τον οποίο θα γίνουν.

Οι λέξεις- κλειδιά του Προτύπου ANSI φαίνονται στον ακόλουθο πίνακα.

auto	**double**	**int**	**struct**
break	**else**	**long**	**switch**
case	**enum**	**register**	**typedef**
char	**extern**	**return**	**union**
const	**float**	**short**	**unsigned**
continue	**for**	**signed**	**void**
default	**goto**	**sizeof**	**volatile**
do	**if**	**static**	**while**

Όλες οι λέξεις –κλειδιά είναι με μικρά γράμματα. Όπως έχουμε αναφέρει η C κάνει διάκριση κεφαλαίων – μικρών, έτσι το **while** είναι μια λέξη – κλειδί, ενώ το **WHILE** δεν είναι. Δεν επιτρέπετε να χρησιμοποιηθεί μια λέξη – κλειδί σαν όνομα μεταβλητής.

4.7 Σχόλια

Στην C μπορούμε να τοποθετούμε σχόλια οπουδήποτε μέσα στο πρόγραμμα. Τα σχόλια βρίσκονται ανάμεσα σε δύο σημειωτές. Ο σημειωτής της αρχής του σχολίου είναι ο /* ενώ ο σημειωτής του τέλους του σχολίου είναι ο */.

```
/*
 αυτό είναι ένα σχόλιο
 */
```

Αν θέλουμε να χρησιμοποιήσουμε σχόλια σε μία μόνο γραμμή μπορούμε να γράψουμε

```
// αυτό είναι ένα σχόλιο
```

4.8 Σταθερές

Στην C, οι σταθερές είναι τιμές που δεν μπορούν να τροποποιηθούν από το πρόγραμμα. Οι σταθερές μπορούν να ανήκουν σε οποιονδήποτε βασικό τύπο δεδομένων. Ο τρόπος παράστασης της κάθε σταθεράς εξαρτάται από τον τύπο της. Οι σταθερές χαρακτήρα τοποθετούνται μέσα σε μονά εισαγωγικά (‘a’, ‘β’, ‘γ’). Οι ακέραιες σταθερές είναι αριθμοί χωρίς κλασματικά μέρη (5, 10, 50). Οι σταθερές κινητής υποδιαστολής χρησιμοποιούν την υποδιαστολή ακολουθούμενη από το κλασματικό μέρος του αριθμού (3.14, 9.81). Παρακάτω αναφέρονται ορισμένα παραδείγματα σταθερών.

Τύπος δεδομένου	**Σταθερές**
char	‘a’ ‘\t’ ‘7’
int	2 234 2456 -453
long int	32000 -43

short int	7 -10	80	
unsigned int	9000	879 30000	
float	132.34	3.44e -3	
double	132.34	132456	-0.96758

Παράδειγμα:

```
#include <stdio.h>
#define DEFAULT_I 29
int main()
{
        char c_val;
        float f_val;
        int i_val1, i_val2;
        i_val1 = DEFAULT_I;
        i_val2 = 3333;
        c_val = 'Q';
        f_val = 23.5 * 2.5;
        printf("Οι τιμές είναι %c  %d  %d  και %f\n", c_val, i_val1, i_val2, f_val);
        return 0;
}
```

4.9 Δεκαεξαδικές και Οκταδικές Σταθερές

Στον προγραμματισμό είναι ευκολότερο σε ορισμένες περιπτώσεις να χρησιμοποιούμε ένα σύστημα αρίθμησης με βάση το 8 ή το 16 αντί για το 10. Το σύστημα αρίθμησης που βασίζεται στο 8 αποκαλείται *οκταδικό* και χρησιμοποιεί τα ψηφία 0 έως 7. Στο οκταδικό, ο αριθμός 10 είναι ίσος με τον αριθμό 8 του δεκαδικού. Το σύστημα αρίθμησης που βασίζεται στο 16 καλείται *δεκαεξαδικό*. Χρησιμοποιεί τα ψηφία 0 έως 9 και τα γράμματα A έως F, τα οποία αντιπροσωπεύουν τους αριθμούς 10 έως 15, αντίστοιχα. Για παράδειγμα ο αριθμός 10 του δεκαεξαδικού είναι ο αριθμός 16 του δεκαδικού. Εξαιτίας της συχνότητας με την οποία χρησιμοποιούνται αυτά τα δύο συστήματα αρίθμησης, η C μας επιτρέπει να καθορίζουμε τις ακέραιες σταθερές σε δεκαεξαδικό ή το οκταδικό αντί για το δεκαδικό, αν το θέλουμε. Μια σταθερά σε δεκαεξαδική μορφή πρέπει να αρχίζει από **0x**. Μια οκταδική σταθερά αρχίζει με ένα μηδενικό. Για παράδειγμα

```
int hex = 0xFF;                 // 255 στο δεκαδικό
into ct = 011;                  // 9 στο δεκαδικό
```

4.10 Αλφαριθμητικές Σταθερές

Εκτός από τις σταθερές των προκαθορισμένων τύπων δεδομένων, η C υποστηρίζει άλλον έναν τύπο σταθερών : την *αλφαριθμητική σταθερά*. Το αλφαριθμητικό είναι ένα σύνολο χαρακτήρων μέσα σε διπλά εισαγωγικά. Για παράδειγμα, το «**αυτό είναι μία δοκιμή**» είναι ένα αλφαριθμητικό. Δεν πρέπει να μπερδεύουμε τα αλφαριθμητικά με τους χαρακτήρες. Μια σταθερά χαρακτήρα τοποθετείται μέσα απλά εισαγωγικά, για

παράδειγμα ‘Α’. Το “Α” από την άλλη μεριά είναι ένα αλφαριθμητικό που περιέχει μόνο ένα γράμμα.

4.11 Σταθερές Ανάποδης Καθέτου

Η χρήση απλών εισαγωγικών σε όλες τις σταθερές χαρακτήρα είναι αρκετή για τους περισσότερους εκτυπώσιμους χαρακτήρες, αλλά ορισμένοι, όπως είναι η νέα γραμμή, είναι αδύνατο να δοθούν από το πληκτρολόγιο. Γι’ αυτό το λόγο η C παρέχει τις ειδικές σταθερές ανάποδης καθέτου, οι οποίες αναφέρονται στον ακόλουθο πίνακα

Κωδικός	**Σημασία**
\b	οπισθοχώρηση
\f	αλλαγή σελίδας
\n	νέα γραμμή
\r	επαναφορά κεφαλής
\t	οριζόντιος στηλογνώμονας
\”	διπλό εισαγωγικό
\’	απλό εισαγωγικό
\?	λατινικό ερωτηματικό
\\	ανάποδη κάθετος
\v	κατακόρυφος στηλογνώμονας
\000	οκταδικός αριθμός
\xhhh	δεκαεξαδικός αριθμός

Παράδειγμα:

```
#include <stdio.h>
int main()
{
        printf(“Αλλαγή γραμμής \n”);
        printf(“%d \t %d”,2,3);
        return 0;
}
```

4.12 Η συνάρτηση printf( )

Για να στείλουμε κάποια δεδομένα στην οθόνη χρησιμοποιούμε την συνάρτηση **printf( )**. Η γενική μορφή της **printf( )** είναι :

printf(“αλφαριθμητικό ελέγχου”, λιστα ορισμάτων)

Το αλφαριθμητικό ελέγχου περιέχει ή χαρακτήρες που θα εμφανιστούν στην οθόνη, ή εντολές φόρμας που λένε στην **printf( )** πώς να εμφανίσει τα υπόλοιπα ορίσματα, ή και τα δύο. Οι κωδικοί φόρμας φαίνονται στον ακόλουθο πίνακα

Κωδικός	**Έξοδος**

%c	απλός χαρακτήρας
%d	δεκαδικός ακέραιος με πρόσημο
%i	δεκαδικός ακέραιος με πρόσημο
%e	αριθμός κινητής υποδιαστολής, συμβολισμός e
%f	αριθμός κινητής υποδιαστολής, δεκαδικός συμβολισμός
%g	χρησιμοποιεί το συντομότερο από %e ή %f
%o	οκταδικός ακέραιος χωρίς πρόσημο
%s	αλφαριθμητικό
%u	δεκαδικός ακέραιος χωρίς πρόσημο
%x	δεκαεξαδικός ακέραιος χωρίς πρόσημο (a – f)
%%	εμφανίζει το σύμβολο %
%p	εμφανίζει ένα δείκτη

Το πρότυπο της συνάρτησης **printf( )** βρίσκεται στο **stdio.h**.

Μια εντολή φόρμας, όπως φαίνεται από τον παραπάνω πίνακα, περιέχει πρώτα το σύμβολο % και στην συνέχεια τον κωδικό φόρμας. Ο αριθμός των ορισμάτων πρέπει να είναι ίσος με τον αριθμό των εντολών φόρμας, και μάλιστα πρέπει να έχουν την ίδια σειρά. Για παράδειγμα, η παρακάτω κλήση της **printf( )**

printf("%d %c %s",5,'a',"δοκιμή");
εμφανίζει **5 α δοκιμή**.

Οι εντολές φόρμας μπορεί να έχουν μετατροπείς που καθορίζουν το μήκος του πεδίου, τον αριθμό των δεκαδικών θέσεων, και μια σημαία ευθυγράμμισης από τα αριστερά. Ένας ακέραιος που τοποθετείται μεταξύ του συμβόλου % και μιας εντολής φόρμας ενεργεί σαν *καθοριστικό ελαχίστου πλάτους πεδίου*. Αυτό το καθοριστικό κάνει τον υπολογιστή να συμπληρώσει την έξοδο με κενά ή μηδενικά για να εξασφαλίσει ότι η έξοδος έχει κάποιο συγκεκριμένο ελάχιστο μήκος. Αν το αλφαριθμητικό ή ο αριθμός είναι μεγαλύτεροι από αυτό το ελάχιστο, η **printf( )**, θα τους εμφανίσει ολόκληρους ακόμα και αν ξεπερνάνε το ελάχιστο. Εξ' ορισμού ο υπολογιστής χρησιμοποιεί κενά διαστήματα για να συμπληρώνει την έξοδο. Αν θέλουμε να χρησιμοποιήσουμε μηδενικά, πρέπει να τοποθετήσουμε το 0 πριν από το καθοριστικό πλάτους του πεδίου. Για παράδειγμα, το **%05d** θα συμπληρώσει έναν αριθμό με μήκος μικρότερο των πέντε ψηφίων χρησιμοποιώντας μηδενικά μέχρι το συνολικό μήκος του αριθμού να γίνει πέντε.

Για να καθορίσουμε τον αριθμό των δεκαδικών θέσεων που θέλουμε να εμφανίζονται σε έναν αριθμό κινητής υποδιαστολής, τοποθετούμε μια υποδιαστολή μετά το καθοριστικό μήκους πεδίου και στη συνέχεια τον αριθμό των δεκαδικών θέσεων που θέλουμε να εμφανίζονται. Για παράδειγμα το **%10.4f** θα εμφανίσει έναν αριθμό πλάτους τουλάχιστον 10 χαρακτήρων με τέσσερις δεκαδικές θέσεις. Όταν εφαρμόζουμε μια φόρμα σαν κι αυτή σε αλφαριθμητικά ή ακεραίους, ο αριθμός που ακολουθεί την τελεία καθορίζει το μέγιστο μήκος του πεδίου. Για παράδειγμα το **%5.7s** θα εμφανίσει ένα αλφαριθμητικό μήκους τουλάχιστον πέντε χαρακτήρων και όχι μεγαλύτερο από 7 χαρακτήρες.Αν το αλφαριθμητικό είναι μεγαλύτερο από το μέγιστο μήκος πεδίου, ο υπολογιστής θα αποκόψει τους χαρακτήρες που βρίσκονται στο τέλος.

Εξ' ορισμού η έξοδος είναι ευθυγραμμισμένη από τα δεξιά : αν το πλάτος του πεδίου είναι μεγαλύτερο από τα δεδομένα που εμφανίζονται, ο υπολογιστής θα τοποθετήσει τα δεδομένα στο δεξιό άκρο του πεδίου. Μπορούμε να ευθυγραμμίσουμε τις πληροφορίες από τα αριστερά τοποθετώντας ένα σύμβολο – μετά το %. Για παράδειγμα το **-%10.2f** θα ευθυγραμμίσει από τα αριστερά έναν αριθμό κινητής υποδιαστολής με δύο δεκαδικές θέσεις σε ένα πεδίο δέκα χαρακτήρων.

Υπάρχουν δύο μετατροπείς εντολών φόρμας που επιτρέπουν στην **printf( )** να εμφανίζει ακέραιους **long** και **short**. Μπορούμε να εφαρμόσουμε αυτούς τους μετατροπείς στους μετατροπείς τύπων **d, i, o, u,** και **x**. Ο μετατροπέας **l** λέει στην **printf( )** ότι ακολουθεί ένα τύπος δεδομένων **long**. Για παράδειγμα, το **%ld** λέει στον υπολογιστή ότι θα εμφανιστεί ένας **long int**. Ο μετατροπέας **l** μπορεί να βρίσκεται μπροστά από τις εντολές κινητής υποδιαστολής **e, f** και **g** και δηλώνει ότι ακολουθεί ένας **double**. Ο μετατροπέας **h** λέει στην **printf( )** να εμφανίσει έναν **short int**. Έτσι το **%hu** δηλώνει ότι τα δεοδμένα είναι τύπου **short unsigned int**.

Παρακάτω παρουσιάζονται μερικά παραδείγματα.

Εντολή printf( )	**Έξοδος**
("%-5.2f",123.234)	123.23
("%5.2.f",3.234)	3.23
("%10s","hello")	hello
("%-10s","hello")	hello
("%5.7s",123456789")	1234567

4.13 Η συνάρτηση scanf( )

Η γενικής χρήσης ρουτίνα για είσοδο από την κονσόλα είναι η **scanf( )**. Μπορεί να διαβάζει όλους τους ενσωματωμένους τύπους δεδομένων, και μπορεί να μετατρέπει αυτόματα τους αριθμούς στην κατάλληλη εσωτερική μορφή. Η γενική μορφή της **scanf( )** είναι

scanf("αλφαριθμητικό ελέγχου», λίστα ορισμάτων);

Το πρότυπο της **scanf( )** βρίσκεται στο **stdio.h**. Το αλφαριθμητικό ελέγχου αποτελείται από τρεις κατηγορίες χαρακτήρων :
- καθοριστικά φόρμας
- χαρακτήρες κενών
- χαρακτήρες μη-κενών

Μπροστά από τα καθοριστικά φόρμας της εισόδου τοποθετείται ένα σύμβολο %, το οποίο πληροφορεί την **scanf( )** έναν η περισσότερους χαρακτήρες κενού που βρίσκονται στον buffer εισόδου. Ένας χαρακτήρας κενού είναι είτε ένα κενό διάστημα, είτε ένας στηλογνώμονας, ή μια νέα γραμμή. Ουσιαστικά, ένας χαρακτήρας κενού μέσα στο αλφαριθμητικό ελέγχου υποχρεώνει την **scanf( )** να διαβάσει, αλλά όχι και να αποθηκεύσει, οποιονδήποτε αριθμό (συμπεριλαμβανομένου του μηδενός) κενών χαρακτήρων , μέχρι να συναντηθεί ο πρώτος χαρακτήρας μη-κενού.

Ένας χαρακτήρας μη-κενού μέσα στο αλφαριθμητικό ελέγχου υποχρεώνει τη **scanf( )** να διαβάζει και να απορρίπτει κάποιο συγκεκριμένο χαρακτήρα. Για παράδειγμα, το αλφαριθμητικό ελέγχου **"%d,%d"** υποχρεώνει την **scanf( )** πρώτα να διαβάσει έναν ακέραιο, στη συνέχεια να διαβάζει και να απορρίπτει ένα κόμμα, και τέλος να διαβάζει άλλον έναν ακέραιο. Αν ο υπολογιστής δεν βρεί τον χαρακτήρα που έχει καθοριστεί, η **scanf( )** θα τερματιστεί.

Όλες οι μεταβλητές που χρησιμοποιούνται για να δέχονται τιμές μέσω της **scanf( )** πρέπει να περνάνε μέσω των διευθύνσεων τους. Αυτό σημαίνει ότι όλα τα ορίσματα πρέπει να είναι δείκτες στις μεταβλητές που χρησιμοποιούνται σαν ορίσματα. Για παράδειγμα, για να διαβαστεί ένας ακέραιος στη μεταβλητή **count**, θα χρησιμοποιήσουμε την παρακάτω κλήση της **scanf( )**

scanf("%d",&count);

Η συνάρτηση **scanf( )** διαβάζει αλφαριθμητικά σε πίνακες χαρακτήρων, και το όνομα του πίνακα χωρίς δείκτη είναι η διεύθυνση του πρώτου στοιχείου του πίνακα. Έτσι για να διαβάσουμε ένα αλφαριθμητικό στον πίνακα χαρακτήρων **address**
scanf("%s",address);

Σ' αυτήν την περίπτωση, ο **address** είναι ήδη ένας δείκτης και δεν χρειάζεται να έχει μπροστά του τον τελεστή **&**.

Πρέπει να ξεχωρίζουμε τα στοιχεία των δεδομένων εισόδου χρησιμοποιώντας κενά διαστήματα, στηλογνώμονες, ή νέες γραμμές. Έτσι η

scanf("%d%d",&a,&b);

δέχεται μια είσοδο σαν το 10 20, αλλά δεν μπορεί να δεχτεί το 10,20. Όπως και με την **printf( )**, οι κωδικοί φόρμας της **scanf( )** πρέπει να βρίσκονται στην ίδια σειρά με τις μεταβλητές της λίστας ορισμάτων που δέχονται την είσοδο.

Όταν τοποθετείτε ένας αστερίσκος (*) μετά το % και πριν από τον κωδικό φόρμας, τα δεδομένα του καθοριζόμενου τύπου θα διαβαστούν, αλλά δεν θα γίνει η εκχώρηση της τιμής. Έτσι στην

scanf("%d%*c%d",&x,&y);

αν δοθεί σαν είσοδος το **10/20**, η **x** θα πάρει την τιμή 10, το σύμβολο της διαίρεσης θα απορριφθεί, και η **y** θα πάρει την τιμή 20.

Οι εντολές φόρμας μπορούν να καθορίσουν ένα μετατροπέα μεγίστου μήκους πεδίου. Αυτός ο μετατροπέας είναι ένας ακέραιος αριθμός που τοποθετείται ανάμεσα στο % και στον κωδικό της εντολής φόρμας και περιορίζει τον αριθμό των χαρακτήρων που διαβάζονται για οποιοδήποτε πεδίο. Για παράδειγμα, για να μην διαβαστούν παραπάνω από 20 χαρακτήρες στο **str**, θα γράφαμε

scanf("%20s",str);

Αν ο buffer εισόδου έχει παραπάνω από 20 χαρακτήρες, τότε η επόμενη κλήση στην είσοδο ξεκινάει από εκεί που τελειώνει αυτή η κλήση. Για παράδειγμα, αν δώσουμε

ΑΒΓΔΕΖΗΘΙΚΛΜΝΞΟΠΡΣΤΥΦΧΨΩ

σαν απόκριση στην παραπάνω κλήση, η **scanf( )** θα τοποθετήσει στο **str** μόνο τους πρώτους 20 χαρακτήρες, δηλαδή μέχρι και το **Υ**, λόγω του καθοριστικού μέγιστου μήκους. Η συνάρτηση δεν χρησιμοποιεί ακόμη τους υπόλοιπους χαρακτήρες, ΦΧΨΩ.

Αν καλέσουμε άλλη μια φορά την **scanf( )**

scanf("%s",str);

τότε ο υπολογιστής θα τοποθετήσει το ΦΧΨΩ στο **str**. Η είσοδος σε ένα πεδίο μπορεί να τελειώσει πριν φτάσουμε στο μέγιστο μήκος πεδίου αν ο υπολογιστής συναντήσει έναν κενό χαρακτήρα. Σ' αυτήν την περίπτωση η **scanf( )** προχωράει στο επόμενο πεδίο.

Παρ' όλο που χρησιμοποιούνται κενά διαστήματα, στηλογνώμονες, και νέες γραμμές σαν διαχωριστικά πεδίων, όταν διαβάζεται ένας χαρακτήρας, ο υπολογιστής διαβάζει όλους τους χαρακτήρες χωρίς διακρίσεις. Για παράδειγμα, αν πληκτρολογήσουμε x y, η scanf("%c%c%c",&a,&b,&c);

θα επιστρέψει τον χαρακτήρα **x** στην **a**, ένα κενό διάστημα στην **b**, και τον χαρακτήρα **y** στην **c**.

Αν στο αλφαριθμητικό ελέγχου έχουμε διάφορους χαρακτήρες, συμπεριλαμβανομένων των κενών διαστημάτων, στηλογνωμόνων, και νέων γραμμών, ο υπολογιστής θα χρησιμοποιήσει αυτούς τους χαρακτήρες για να εντοπίσει και να απορρίψει χαρακτήρες από τον buffer εισόδου. Ο υπολογιστής θα απορρίψει όλους τους ίδιους χαρακτήρες Για παράδειγμα, αν δοθεί σαν είσοδος το **10t20** η

scanf("%st%s",&x,&y);

θα τοποθετήσει το 10 στο **x** και το 20 στο **y**. Ο υπολογιστής θα απορρίψει το χαρακτήρα **t** γιατί υπάρχει το **t** στο αλφαριθμητικό ελέγχου. Η

scanf("%s ",name);

δεν θα επιστρέψει, μέχρι να πληκτρολογήσουμε ένα χαρακτήρα αφού πληκτρολογήσυομε ένα χαρακτήρα κενού. Αυτό συμβαίνει, γιατί το κενό διάστημα μετά το **%s** λέει στην **scanf( )** να διαβάζει και να απορρίπτει τα κενά διαστήματα, τους στηλογνώμονες και τις νέες γραμμές.

4.14 Τελεστές

Ο τελεστής (operator) είναι ένα σύμβολο το οποίο λέει στο μεταγλωττιστή να εκτελέσει ειδικές μαθηματικές ή λογικές πράξεις. Η C διαθέτει τρείς γενικές κατηγορίες τελεστών: τους αριθμητικούς (arithmetic), τους συσχετιστικούς

(relational) και λογικούς (logical) και τους τελεστές για bit (bitwise). Σ' αυτό το κεφάλαιο θα εξετάσουμε μόνο τους αριθμητικούς τελεστές, τον τελεστή εκχώρησης τιμής και τον τελεστή **sizeof**.

Αριθμητικοί Τελεστές

Οι αριθμητικοί τελεστές φαίνονται στον πίνακα που ακολουθεί:

Τελεστής	Πράξη
-	αφαίρεση, αρνητικό πρόσημο
+	Πρόσθεση
*	πολλαπλασιασμός
/	Διαίρεση
%	ακέραιο υπόλοιπο διαίρεσης
--	μείωση κατά ένα
++	αύξηση κατά ένα

Οι αριθμητικοί τελεστές μπορούν να εφαρμοστούν σε όλους σχεδόν τους ενσωματωμένους τύπους δεδομένων που διαθέτει η C. Η διαίρεση δύο ακεραίων θα δώσει ακέραιο αποτέλεσμα ενώ η διαίρεση πραγματικών αριθμών θα δώσει πραγματικό αποτέλεσμα. Έτσι το 10/3 θα δώσει αποτέλεσμα 3 εφόσον έχουμε ακέραια διαίρεση.

Ο τελεστής υπολοίπου % παράγει το υπόλοιπο μιας διαίρεσης ακεραίων. Δεν μπορεί να χρησιμοποιηθεί με τους τύπους **float** ή **double**.

Παράδειγμα:

```
#include <stdio.h>
int main()
{
        int x, y;
        x = 10;
        y = 3;
        printf("%d\n", x/y);            // θα δώσει 3
        printf("%d\n", x % y);          // θα δώσει 1
        x = 1;
        y = 2;
        printf("%d %d\n", x /y, x % y);   // θα δώσει 0 και 1
        return 0;
}
```

Η τελευταία **printf( )** θα δώσει 0 και 1 γιατί το ½ στην διαίρεση ακεραίων είναι 0 με υπόλοιπο 1.

Το σύμβολο – πολλαπλασιάζει τον τελεστέο του με το -1. Έτσι αν τοποθετηθεί μπροστά από κάποιον αριθμό, θα έχουμε αλλαγή του προσήμου του.

Τελεστής Εκχώρησης Τιμής

Ο τελεστής εκχώρησης τιμής για την C είναι το σύμβολο =. Ο τελεστής εκχώρησης τιμής «εχκωρεί» την τιμή της δεξιάς πλευράς στην μεταβλητή της αριστερής πλευράς.

Παράδειγμα:

```
#include <stdio.h>
int main()
{
        int a, b, sum;
        printf("Δώσε δύο αριθμούς : ");
        scanf("%d %d",&a,&b);
        sum = a + b;
        printf("Το άθροισμα των δύο αριθμών είναι %d \n",sum);
        return 0;
}
```

Ο Τελεστής sizeof

Η C περιλαμβάνει τον τελεστή χρόνου μεταγλώττισης **sizeof** ο οποίος επιστρέφει το μέγεθος της μεταβλητής ή του τύπου που χρησιμοποιείται σαν τελεστέος. Η λέξη-κλειδί **sizeof** βρίσκεται μπροστά από τη μεταβλητή ή το όνομα του τύπου που χρησιμοποιείται σαν τελεστέος. Όταν ο **sizeof** εφαρμόζεται σε ένα τύπο δεδομένων, τότε ο τύπος πρέπει να βρίσκεται μέσα σε παρενθέσεις. Για παράδειγμα
sizeof(int)

Παράδειγμα:

```
#include <stdio.h>
int main()
{
        printf("Ενας int καταλαμβάνει %d bytes\n",sizeof(int));
        return 0;
}
```

Τελεστές Μοναδιαίας Αύξησης και Μείωσης

Η C διαθέτει δυό χρήσιμους τελεστές. Τον τελεστή μοναδιαίας αύξησης ++ και τον τελεστή μοναδιαίας μείωσης --. Ο τελεστής μοναδιαίας αύξησης προσθέτει 1 στον τελεστέο του, ενώ ο τελεστής μοναδιαίας μείωσης αφαιρεί 1 από τον τελεστέο του.
Έτσι οι παρακάτω πράξεις:

```
x++;
x--;
```

είναι ίδιες με τις

```
x = x + 1;
x = x – 1;
```

Τόσο ο μοναδιαίος τελεστής αύξησης όσο και ο μοναδιαίος τελεστής μείωσης μπορεί να βρίσκεται στην αρχή ή στο τέλος του τελεστέου. Για παράδειγμα, μπορεί να γραφεί το

```
x = x + 1;
σαν
x++;
ή σαν
++x;
```

Υπάρχει όμως, μια σημαντική διαφορά όταν τους χρησιμοποιούμε σε μιά παράσταση. Όταν ο μοναδιαίος τελεστής αύξησης ή μείωσης, βρίσκεται μπροστά από τον τελεστέο του, η C εκτελεί την πράξη της αύξησης ή της μείωσης πριν χρησιμοποιήσει την τιμή του τελεστέου. Όταν ο τελεστής βρίσκεται μετά τον τελεστέο του η C χρησιμοποιεί την τιμή του τελεστέου πριν τον αυξήσει ή τον μειώσει κατά 1. Για παράδειγμα :

```
x = 10;
y = ++x;
```

Η C αποδίδει στην **y** την τιμή **11** γιατί αυξάνει πρώτα την x και στη συνέχεια αποδίδει την τιμή του στην **y**. Αν όμως ο κώδικας είχε γραφεί

```
x = 10;
y = x++;
```

η C θα είχε αποδώσειστην **y** την τιμή **10** και στη συνέχεια θα είχε αυξήσει κατά **1** την τιμή της **x**. Και στις δύο περιπτώσεις η x παίρνει την τιμή 11, η διαφορά όμως βρίσκεται στο πότε παίρνει αυτήν την τιμή. Η σειρά προτεραιότητας των αριθμητικών τελεστών δίνεται παρακάτω :

Υψηλή προτεραιότητα	++ --
	-
	* / %
Χαμηλή προτεραιότητα	+ -

Όταν έχουμε ισοδύναμους τελεστές, ο υπολογιστής δίνει προτεραιότητα στον τελεστή που βρίσκεται αριστερά. Μπορούμε βέβαια, να χρησιμοποιήσουμε παρενθέσεις για να αλλάξουμε την σειρά προτεραιότητας. Η C αποδίδει σε μια πράξη ή σε μια σειρά πράξεων ένα υψηλότερο επίπεδο προτεραιότητας, όταν αυτές βρίσκονται μέσα σε παρενθέσεις.

Τελεστές Αντικατάστασης και Παραστάσεις

Παραστάσεις όπως η

```
i = i + 4
```

όπου η αρχική μεταβλητή του αριστερού μέλους επαναλαμβάνεται αμέσως στην αρχή του δεξιού μέλους, μπορούν να γραφούν σαν

```
i += 4
```

Ο τελεστής += λέγεται τελεστής αντικατάστασης (assignment operator). Άλλοι τελεστές αντικατάστασης είναι οι : **-=, *=, /=, %=**.

Αν οι παράσταση_1 και παράσταση_2 είναι παραστάσεις, τότε η

παράσταση_1 τελεστής= παράσταση_2

είναι ισοδύναμη με την

παράσταση_1 = (παράσταση_1) τελεστής (παράσταση_2)

Δηλαδή
x *= y + 2
είναι ισοδύναμο με
x = x * (y + 2)
και όχι
x = x * y + 2

Παράδειγμα:

```
 #include <stdio.h>
int main()
{
        int a,b;
        int temp;
        printf("Δώσε την πρώτη μετβλητή : ");
        scanf("%d",&a);
        temp = a;
        printf("Δώσε την δύτερη μετβλητή : ");
        scanf("%d",&b);
        a += b;
        printf("%d\n",a);
        a = temp;
        a -= b;
        printf("%d\n",a);
        a = temp;
        a *= b;
        printf("%d\n",a);
        a = temp;
        a /= b;
        printf("%d\n",a);
        a = temp;
        a %= b;
        printf("%d\n",a);
        return 0;
}
```

Παραστάσεις

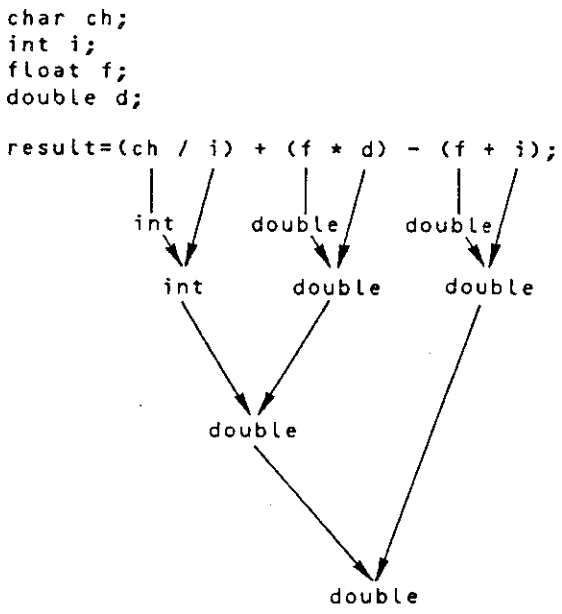
Οι τελεστές, οι σταθερές, και οι μεταβλητές είναι τα συστατικά των παραστάσεων. Μια παράσταση στη C είναι οποιοσδήποτε έγκυρος συνδιασμός αυτών των στοιχείων. Επειδή οι παραστάσεις έχουν την τάση να ακολουθούν τους γενικούς κανόνες της άλγεβρας, συχνά θεωρούνται σαν κάτι το δεδομένο.

Μετατροπή Τύπων σε Παραστάσεις

Όταν έχουμε σταθερές και μεταβλητές διαφορετικών τύπων σε μια παράσταση, η C τις μετατρέπει όλες στον ίδο τύπο. Ο μεταγλωττιστής της C θα μετατρέψει πράξη προς πράξη όλους τους τελεστέους προς τα «επάνω» στον τύπο του μεγαλύτερου τελεσταίου, όπως περιγράφεται στους παρακάτω κανόνες μετατροπής τύπων:

1. Όλοι οι τύποι **char** και **short int** μετατρέπονται σε **int**. Όλοι οι τύποι **float** μετατρέπονται σε **double**.
2. Για όλα τα ζευγάρια τελεστέων, η σειρά μετατροπής είναι η παρακάτω. Αν ο ένας από τους τελεστέους είναι **long double**, και ο άλλος τελεσταίος μετατρέπεται σε **long double**. Αν ο ένας τελεστέος είναι **double**, και ο άλλος τελεστέος μετατρέπεται σε **double**. Αν ο ένας από τους τελεστέους είναι **long** και ο άλλος τελεστέος μετατρέπεται σε **long**. Αν ο ένας από τους τελεστέους είναι **unsigned** και ο άλλος τελεστέος μετατρέπεται σε **unsigned**.

Αφού ο μεταγλωττιστής εφαρμόσει αυτούς τους κανόνες μετατροπής, το κάθε ζευγάρι τελεστέων θα γίνει ενός τύπου, οπότε το αποτέλεσμα της κάθε πράξης θα είναι το ίδιο με τον τύπο και των δύο τελεστέων. Για παράδειγμα, ας δούμε τις μετατροπές τύπων που συμβαίνουν παρακάτω:



Πρώτα, ο υπολογιστής μετατρέπει τον χαρακτήρα **ch** σε **int** και την **float f** σε **double**. Στη συνέχεια μετατρέπεται το αποτέλεσμα **ch/i** σε **double** γιατί το **f * d** είναι **double**. Το τελικό αποτέλεσμα είναι **double** γιατί και οι δύο τελεστέοι είναι **double**.

Προσαρμογή Τύπων

Μπορούμε να υποχρεώσουμε μια παράσταση να είναι κάποιου συγκεκριμένου τύπου, χρησιμοποιώντας μια δομή που ονομάζεται *προσαρμογή* (cast). Η γενική μορφή της είναι :

(τύπος) παράσταση

όπου τύπος είναι ένας από τους καθιερωμένους τύπους της C. Για παράδειγμα, αν ο **x** είναι ακέραιος και θέλουμε να είμαστε βέβαιοι ότι ο υπολογιστής θα θεωρήσει ότι η παράσταση **x/2** είναι τύπου **float** ώστε να υπάρχει κλασματικό μέρος, μπορούμε να γράψουμε

(float) x / 2

Εδώ η προσαρμογή **(float)** συνδιάζεται με τον **x**, οπότε το **2** μετατρέπεται σε **float** και έτσι και το αποτέλεσμα είναι **float** (βέβαια θα μπορούσα να γράψουμε **x / 2.0** οπότε το αποτέλεσμα θα ήταν πάλι **float**). Αν όμως γράφαμε

(float) (x / 2)

τότε ο υπολογιστής θα εκτελέσει πρώτα την ακέραια διαίρεση, και μετά θα μετατρέψει το αποτέλεσμα που προκύπτει σε **float**.

Παράδειγμα:

```
#include <stdio.h>
int main()
{
        float result;
        int x;
        printf("Δώσε έναν ακέραιο αριθμό : ");
        scanf("%d",&x);
        result = (float) x / 2;
        printf("%.2f\n",result);
        result = (float) (x / 2);
        printf("%.2f\n",result);
        return 0;
}
```

Κενά Διαστήματα και Παρενθέσεις

Μπορούμε να τοποθετούμε κενά διαστήματα σε μια παράσταση, σύμφωνα με τις προτιμήσεις μας, για να κάνουμε το πρόγραμμα περισσότερο ευανάγνωστο. Για παράδειγμα οι παρακάτω δύο παραστάσεις είναι ίδιες :

x = 5/y*(26/x);
x = 5 / y * (26 / x)

Η χρήση παρενθέσεων δεν αποτελεί λάθος ούτε καθυστερεί τον υπολογισμό της παράστασης. Καλό είναι να χρησιμοποιούμε παρενθέσεις για να γίνεται σαφής η ακριβής σειρά υπολογισμού, τόσο για τον ευατό μας όσο και για τους άλλους που μπορεί να χρειαστεί να καταλάβουν το πρόγραμμά μας.

x = y / 2 – 23 * grade – 43;
x = (y / 2) _ ((23 * grade) – 43);

Όπως είναι προφανές η δεύτερη παράσταση διαβάζεται ευκολότερα.

Άσκηση 2.1

Να γραφεί πρόγραμμα το οποίο :
α) θα ζητάει να πληκτρολογηθούν δύο αριθμοί,
β) θα υπολογίζει και θα εμφανίζει το γινόμενο και το πηλίκο τους.

```
#include <stdio.h>
int main()
{
        int a,b;
        printf("Δώσε δύο αριθμούς :");
        scanf("%d %d",&a,&b);
        printf("Το γινόμενο των αριθμών είναι %d\n",a * b);
        printf("Το πηλίκο των αριθμών είναι %.2f\n",(float) a / b);
        return 0;
}
```

Άσκηση 2.2

Να γραφεί πρόγραμμα το οποίο :
α) θα ζητάει να πληκτρολογηθεί το ύψος h από το οποίο αφήνεται ελεύθερο ένα σώμα,
β) θα υπολογίζει και θα εμφανίζει στην οθόνη των ταχύτητα u με την οποία φτάνει στο έδαφος.

Η ταχύτητα δίνεται από τη σχέση $u = \sqrt{2gh}$ (g = 9.81 m/sec^2).

```
#include <stdio.h>
#include <math.h>
int main()
{
        int h;
        float u;
        printf("Δώσε το ύψος h : ");
        scanf("%d",&h);
        u = sqrt(2 * 9.81 * h);
        printf("Η ταχύτητα υ είναι : %.2f\n",u);
        return 0;
}
```

Άσκηση 2.3

Να γραφεί πρόγραμμα το οποίο θα διαβάζει μια απόσταση σε μέτρα, θα την μετατρέπει σε ίντσες και θα εμφανίζει το αποτέλεσμα.

Είναι γνωστό ότι 1 μέτρο = 39,3700787 ίντσες.

```
#include <stdio.h>
int main()
{
        int meters;
        float in;
        printf("Δώσε την απόσταση σε μέτρα : ");
        scanf("%d",&meters);
        in = meters * 39.3700787;
        printf("Η απόσταση σε ίντσες είναι %.2f\n",in);
        return 0;
}
```

Άσκηση 2.4

Να γραφεί πρόγραμμα το οποίο :

α) θα ζητάει από το πληκτρολόγιο το μήκος L ενός εκκρεμούς,
β) θα υπολογίζει και θα εμφανίζει στην οθόνη την περίοδο του εκκρεμούς.

Η περίοδος του εκκρεμούς δίνεται από τον τύπο $T = 2\pi\sqrt{\frac{L}{g}}$ όπου g=9,81 m/sec^2

```
#include <stdio.h>
#include <math.h>
int main()
{
        int L;
        float T;
        printf("Δώσε το μήκος του εκκρεμούς : ");
        scanf("%d",&L);
        T = 2 * 3.14 * sqrt(L / 9.81);
        printf("Η περίδος του εκκρεμούς είναι %.2f\n",T);
        return 0;
}
```

Άσκηση 2.5

Δίνεται ένα χρηματικό ποσό σε ευρώ. Να γραφεί πρόγραμμα που θα υπολογίζει και θα τυπώνει το ποσό αυτό σε δολάρια.
Δίνεται ότι η ισοτιμία των δύο νομισμάτων είναι 1 € = 0,9834 $

```
#include <stdio.h>
int main()
{
        int euro;
        float dollar;
        printf("Δώσε το χρηματικό ποσό σε ευρώ : ");
        scanf("%d",&euro);
        dollar = euro * 0.9834;
        printf("Το ποσό σε δολλάρια είναι : %.2f\n",dollar);
        return 0;
}
```

4.15 Ασκήσεις

1. Να γραφεί πρόγραμμα που θα διαβάζει τις πλευρές ενός ορθογωνίου παραλληλογράμμου και θα υπολογίζει και θα εμφανίζει την περίμετρο και το εμβαδόν του. **Σημείωση** : Αν οι πλευρές του ορθογωνίου παραλληλογράμμου είναι οι a και b τότε Περίμετρος = 2 • (a + b) και Εμβαδόν = a • b.
2. Να γραφεί πρόγραμμα το οποίο θα διαβάζει από το πληκτρολόγιο την ακτίνα ενός κύκλου και θα υπολογίζει και θα εμφανίζει το μήκος της περιφέρειας και το εμβαδόν του. **Σημείωση** : Αν η ακτίνα του κύκλου είναι R τότε Μήκος περιφέρειας = 2 • π • R και Εμβαδόν = π • R^2 (π = 3,14).
3. Είναι γνωστό ότι 1 Kbyte = 1024 bytes και 1 byte = 8 bits. Να γραφεί πρόγραμμα το οποίο :
α. θα διαβάζει τη χωρητικότητα ενός αρχείου δεδομένων σε Kbytes,
β. θα τη μετατρέπει σε bytes και bits,
γ. θα εμφανίζει τον αριθμό των Kbytes, των bytes και των bits του αρχείου.

4. Ένα κινητό κινείται με σταθερή επιτάχυνση a = 5 m/sec^2. Να γραφεί πρόγραμμα το οποίο :
α. θα ζητάει από το πληκτρολόγιο τον χρόνο κίνησης του κινητού,
β. θα υπολογίζει και θα εκτυπώνει στην οθόνη την ταχύτητα και το διάστημα που διένυσε το κινητό. Δίνονται οι σχέσεις υ = α • t και $s = \frac{1}{2}at^2$.

5. Να γραφεί πρόγραμμα το οποίο :
α. θα διαβάζει από το πληκτρολόγιο την ακτίνα r και το ύψος h ενός κυλίνδρου,
β. θα υπολογίζει και θα εμφανίζει στην οθόνη τον όγκο V και το συνολικό εμβαδόν A της επιφάνειάς του. Δίνονται οι σχέσεις : V = πr^{2}h και A = 2πrh + 2πr^2 (π = 3,14).

6. Σε ένα Video Club προσφέρονται για την ενοικίαση DVD δύο τρόποι πληρωμής :
▫ Εγγραφή μέλους 25 € και τιμή ενοικίασης DVD 1,5 €.
▫ Χωρίς εγγραφή, τιμή ενοικίασης DVD 2 €.
Να γραφεί πρόγραμμα το οποίο :
α. θα δέχεται των αριθμό των DVD που ενοικίασε ένας πελάτης και
β. θα υπολογίζει και θα εμφανίζει το κόστος ενοικίασης και με τους δύο τρόπους πληρωμής.

7. Μια πολυκατοικία έχει τρία διαμερίσματα. Το διαμέρισμα του 1^{ου} ορόφου πληρώνει το 27 % των κοινοχρήστων, το διαμέρισμα του 2^{ου} ορόφου το 33 % των κοινοχρήστων και το διαμέρισμα του 3^{ου} ορόφου το 40 %. Να γραφεί πρόγραμμα το οποίο :
α. θα διαβάζει από το πληκτρολόγιο το σύνολο των δαπανών,
β. θα υπολογίζει και θα εμφανίζει στην οθόνη τα κοινόχρηστα κάθε διαμερίσματος.

8. Να γραφεί πρόγραμμα το οποίο :
α. θα ζητάει από το πληκτρολόγιο τα μήκη των πλευρών a,b και c ενός τριγώνου,
β. θα υπολογίζει και θα εμφανίζει στην οθόνη το εμβαδόν E του τριγώνου.
Το εμβαδόν E δίνεται από τη σχέση $E = \sqrt{t(t-a)(t-b)(t-c)}$ όπου t είναι η ημιπερίμετρος του τριγώνου κια δίνεται από τη σχέση t = (a + b + c) / 2.

9. Να γραφεί πρόγραμμα το οποίο θα εμφανίζει τα ψηφία δοθέντος διψήφιου ακέραιου αριθμού με αντίστροφη σειρά. Δηλαδή αν ο αριθμός που δίνει ο χρήστης από το πληκτρολόγιο είναι ο 38 τότε το πρόγραμμα πρέπει να εμφανίσει τον 83.

10. Να γραφεί πρόγραμμα το οποίο θα δέχεται σαν είσοδο έναν τριψήφιο ακέραιο αριθμό και θα εμφανίζει το άθροισμα των ψηφίων του. Δηλαδή αν δοθεί ο αριθμός 385 το πρόγραμμα πρέπει να εμφανίσει το άθροισμα των ψηφίων του δηλαδή 16.

11. Να γραφεί πρόγραμμα το οποίο θα δέχεται σαν είσοδο έναν τριψήφιο ακέραιο αριθμό και θα επιστρέφει το αποτέλεσμα της διαίρεσή του με το τελευταίο ψηφίο του. Δηλαδή αν ο αριθμός είναι ο 389 και διαιρεθεί με το τελευταίο ψηφίο του πού είναι το 9 πρέπει να δώσει σαν αποτέλεσμα το 43,22.

12. Δίνεται το παρακάτω μαθηματικό παιχνίδι :
α) Σκέψου έναν αριθμό.
β) Πρόσθεσε το 5.
γ) Διπλασίασε το αποτέλεσμα.
δ) Αφαίρεσε το 4.
ε) Διαίρεσε το αποτέλεσμα με το 2.
στ) Αφαίρεσε τον αριθμό πού σκέφτηκες.
Το αποτέλεσμα είναι 3.
Να γραφεί πρόγραμμα που να υλοποιεί το παραπάνω παιχνίδι και να επαληθεύει το αποτέλεσμά του.

13. Να γραφεί πρόγραμμα το οποίο θα διαβάζει την θερμοκρασία περιβάλλοντος σε βαθμούς Κελσίου, θα τη μετατρέπει σε βαθμούς Φαρενάϊτ και θα την εμφανίζει στην οθόνη.

Δίνεται η σχέση $F = \frac{9}{5}C + 32$ όπου F είναι η θερμοκρασία σε βαθμούς Φαρενάϊτ και C η θερμοκρασία σε βαθμούς Κελσίου.

14. Να γραφεί πρόγραμμα το οποίο :

α) θα ζητάει από το πληκτρολόγιο τους βαθμούς της προόδου και της τελικής εξέτασης ενός σπουδαστή, στο μάθημα Προγραμματισμός Ι,

β) θα υπολογίζει και θα εμφανίζει στην οθόνη τον τελικό βαθμό του φοιτητή στο συγκεκριμένο μάθημα.

Δίνεται ότι ο βαθμός της προόδου συμμετέχει στην τελική βαθμολογία με ποσοστό 30% ενώ η τελική εξέταση με ποσοστό 70%. (Δηλαδή αν ο βαθμός προόδου κάποιου σπουδαστή είναι 7,5 και ο βαθμός της τελικής εξέτασης είναι 6, τότε ο τελικός βαθμός είναι 7,5 * 30 % + 6 * 70%).

15. Να γραφεί πρόγραμμα το οποίο :

α) θα διαβάζει από το πληκτρολόγιο τις ακαθάριστες αποδοχές ενός υπαλλήλου,

β) θα υπολογίζει και θα εμφανίζει τις ασφαλιστικές εισφορές, τον αναλογούντα φόρο και τις καθαρές μηνιαίες αποδοχές του υπαλλήλου.

Δίνεται ότι το ποσοστό των ασφαλιστικών εισφορών είναι το 30% των ακαθάριστων αποδοχών και ο ανάλογος φόρος το 10% του ποσού (ακαθάριστες αποδοχές – ασφαλιστικές εισφορές).

5. ΣΤΟΙΧΕΙΑ ΛΟΓΙΚΗΣ & ΔΟΜΕΣ ΕΛΕΓΧΟΥ

5.1 Λογικές Προτάσεις

Λογικές είναι οι προτάσεις που παίρνουν μία από τις δύο λογικές τιμές : ΑΛΗΘΗΣ (τιμή διάφορη του μηδενός για την C) ή ΨΕΥΔΗΣ (τιμή ίση με μηδέν για την C).

Μια λογική πρόταση μπορεί να αποτελείται από σταθερές, μεταβλητές και αριθμητικές εκφράσεις που συνδέονται μεταξύ τους με τους συγκριτικούς τελεστές. Οι συγκριτικοί τελεστές που υποστηρίζονται από την C δίνονται στον παρακάτω πίνακα :

Τελεστής	Σύγκριση
<	μικρότερο
>	μεγαλύτερο
<=	μικρότερο ή ίσο
>=	μεγαλύτερο ή ίσο
!=	διάφορο
= =	ίσο με

Οι συγκριτικοί τελεστές επιδρούν στά δεδομένα όλων των βασικών τύπων.

Παράδειγμα
```
#include <stdio.h>
int main()
{
    int a = 3;
    int b = 5;
    if (b > a)
    printf("Ο αριθμός 5 είναι μεγαλύτερος του αριθμού 3");
    return 0;
}
```

Η σύγκριση χαρακτήρων γίνεται με βάση τον κώδικα αναπαράστασης των χαρακτήρων στην ακολουθία χαρακτήρων ASCII.

Παράδειγμα
```
#include <stdio.h>
int main()
{
    char ch1 = 'A';
    char ch2 = 'B';
    if (ch2 > ch1)
        printf("Ο χαρακτήρας A είναι μεγαλύτερος από τον χαρακτήρα B");
    return 0;
}
```
Η σύγκριση λογικών δεδομένων έχει έννοια μόνο στην περίπτωση των τελεστών = = , != .

Παράδειγμα

```
# include <stdio.h>
#define FALSE 0
#define TRUE 1
int main()
{
    int a = TRUE;
    int b = TRUE;
    if  (a == b)
    printf("είναι αληθείς και οι δύο μεταβλητές");
    return 0;
}
```

ή με κάποια διαφορετική μορφή

Παράδειγμα

```
# include <stdio.h>
int main()
{
    enum bool {false,true};
    enum bool a = true;
    enum bool b = true;
    if  (a == b)
        printf("είναι αληθείς και οι δύο μεταβλητές");
    return 0;
}
```

5.2 Λογικές Πράξεις

Οι λογικές πράξεις μπορούν να γίνουν μεταξύ δεδομένων λογικού τύπου (λογικών μεταβλητών και λογικών σταθερών). Οι βασικές λογικές πράξεις είναι τρείς και υλοποιούνται με τους λογικούς τελεστές ! (άρνηση), && (σύζευξη), || (διάζευξη).

Άρνηση
Η άρνηση μιάς λογικής πρότασης είναι ΑΛΗΘΗΣ (ή ΨΕΥΔΗΣ) όταν η πρόταση είναι αντίστοιχα ΑΛΗΘΗΣ (ή ΨΕΥΔΗΣ).

Παράδειγμα . Η παράσταση ! (3 > 5) δίνει αποτέλεσμα 1 (ΑΛΗΘΗΣ) αφού η παράσταση (3 > 5) δίνει αποτέλεσμα 0 (ΨΕΥΔΗΣ).

Σύζευξη
Η σύζευξη δύο λογικών προτάσεων έχει τιμή ΑΛΗΘΗΣ όταν και οι δύο προτάσεις έχουν τιμή ΑΛΗΘΗΣ. Αν μία από τις προτάσεις έχει τιμή ΨΕΥΔΗΣ, τότε το αποτέλεσμα της σύζευξης έχει τιμή ΨΕΥΔΗΣ.

Παράδειγμα. Η παράσταση (3 < 5) && (6 > 4) δίνει αποτέλεσμα 1 (ΑΛΗΘΗΣ) αφού και η παράσταση (3 < 5) δίνει αποτέλεσμα 1 (ΑΛΗΘΗΣ) και η παράσταση (6 > 4).

Διάζευξη

Η διάζευξη δύο λογικών προτάσεων έχει τιμή ΑΛΗΘΗΣ όταν μία τουλάχιστον από τις δύο προτάσεις έχει τιμή ΑΛΗΘΗΣ. Μόνο αν και οι δύο προτάσεις έχουν τιμή ΨΕΥΔΗΣ, το αποτέλεσμα της διάζευξης έχει τιμή ΨΕΥΔΗΣ.

Παράδειγμα. Η παράσταση (3 > 5) || (4 > 3) δίνει αποτέλεσμα 1 (ΑΛΗΘΗΣ) αφού η παράσταση (3 > 5) δίνει αποτέλεσμα 0 (ΨΕΥΔΗΣ) αλλά η παράσταση (4 > 3) δίνει αποτέλεσμα 1 (ΑΛΗΘΗΣ).

Τα αποτελέσματα των λογικών πράξεων δίνονται συγκεντρωτικά στον παρακάτω πίνακα που είναι γνωστός και ως πίνακας αληθείας λογικών πράξεων.

Λογική Μεταβλητή	X	Y	! X	X && Y	X || Y
Τιμή	ΑΛΗΘΗΣ	ΑΛΗΘΗΣ	ΨΕΥΔΗΣ	ΑΛΗΘΗΣ	ΑΛΗΘΗΣ
	ΑΛΗΘΗΣ	ΨΕΥΔΗΣ	ΨΕΥΔΗΣ	ΨΕΥΔΗΣ	ΑΛΗΘΗΣ
	ΨΕΥΔΗΣ	ΑΛΗΘΗΣ	ΑΛΗΘΗΣ	ΨΕΥΔΗΣ	ΑΛΗΘΗΣ
	ΨΕΥΔΗΣ	ΨΕΥΔΗΣ	ΑΛΗΘΗΣ	ΨΕΥΔΗΣ	ΨΕΥΔΗΣ

Τονίζουμε ξανά ότι για την C η τιμή **ΑΛΗΘΗΣ** είναι οποιοσδήποτε αριθμός διάφορος του 0 ενώ η τιμή **ΨΕΥΔΗΣ** είναι ο αριθμός 0.

5.3 Ιεραρχία Τελεστών στις Λογικές Πράξεις

⇒ Ο τελεστής άρνησης ! έχει την υψηλότερη προτεραιότητα.
⇒ Οι αριθμητικοί τελεστές (*, /, %, +, -) έχουν μεγαλύτερη προτεραιότητα από τους τελεστές συσχέτισης.
⇒ Οι τελεστές συσχέτισης (>, >=, <, <=) έχουν όλοι την ίδια προτεραιότητα.
⇒ Την ακριβώς επόμενη προτεραιότητα έχουν οι τελεστές ισότητας (==, !=).
⇒ Οι λογικοί τελεστές (&&, ||) έχουν την χαμηλότερη προτεραιότητα και οι παραστάσεις που συνδέονται με αυτούς υπολογίζονται από τα αριστερά προς τα δεξιά - ο υπολογισμός θα σταματήσει μολις εξακριβωθεί ότι το αποτέλεσμα είναι ΑΛΗΘΗΣ ή ΨΕΥΔΗΣ.

Ακολουθεί η πινακοποίηση της προτεραιότητας των τελεστών συσχέτισης και των λογικών τελεστών.

Τελεστές	Σειρά εκτέλεσης των πράξεων σε περίπτωση ίδια προτεραιότητας
!	
> >= < <=	από αριστερά προς τα δεξιά
== !=	από αριστερά προς τα δεξιά
&& ||	από αριστερά προς τα δεξιά

Παραδείγματα

⇒ Επειδή οι τελεστές συσχέτισης και οι λογικοί τελεστές έχουν χαμηλότερη προτεραιότητα από τους αριθμητικούς τελεστές, η παράσταση 10 > 1 + 12 υπολογίζεται σαν να την είχαμε γράψει 10 > (1 + 12).

⇒ Η φυσική σειρά υπολογισμού της παράστασης 1 && !0 || 1 είναι η ακόλουθη : πρώτα υπολογίζεται το !0 και δίνει αποτέλεσμα 1 (ΑΛΗΘΗΣ) μετά υπολογίζεται η παράσταση 1 && 1 που δίνει αποτέλεσμα 1 (ΑΛΗΘΗΣ) και τελικά υπολογίζεται η παράσταση 1 || 1 που δίνει τιμή 1 (ΑΛΗΘΗΣ)

⇒ αν όμως χρησιμοποιήσουμε παρενθέσεις, δηλαδή 1 && ! (0 || 1) τότε : πρώτα θα υπολογιστεί η παράσταση (0 || 1) που δίνει 1 (ΑΛΗΘΗΣ) μετά θα υπολογιστεί η άρνηση ! 1 που δίνει 0 (ΨΕΥΔΗΣ) μετά θα υπολογιστεί η παράσταση 1 && 0 που δίνει αποτέλεσμα 0 (ΨΕΥΔΗΣ)

⇒ Θα πρέπει να σημειώσουμε ότι οι δύο διαφορετικές τιμές που μπορεί να πάρει μια λογική μεταβλητή, ονομάζονται κατά σύμβαση ΑΛΗΘΗΣ και ΨΕΥΔΗΣ. Θα μπορούσε αντίστοιχα να είναι ΑΝΟΙΚΤΟ, ΚΛΕΙΣΤΟ ή οτιδήποτε άλλο.

Θα μπορούσαμε να προσομοιώσουμε διάφορες λειτουργίες ή καταστάσεις με μία λογική μεταβλητή, όπως :
Ηλεκτρικός διακόπτης : Η τιμή ΑΛΗΘΗΣ θα μπορούσε να αντιστοιχεί στην κατάσταση «ανοιχτός διακόπτης» («δεν περνάει ρεύμα») και η τιμή ΨΕΥΔΗΣ στην κατάσταση «κλειστός διακόπτης» («περνάει ρεύμα»).

Παράδειγμα
```
# include <stdio.h>
#define CLOSE 0
#define OPEN 1
int main()
{
    int a = OPEN;
    int b = OPEN;
    if  (a == b)
        printf("οι διακόπτες είναι ανοικτοί");
    return 0;
}
```

Φωτεινός σηματοδότης : Η τιμή ΑΛΗΘΗΣ θα μπορούσε να αντιστοιχεί στην κατάσταση «πράσινο» («επιτρέπεται η διέλευση») και η τιμή ΨΕΥΔΗΣ στην κατάσταση «κόκκινο» («απαγορεύεται η διέλευση»).

Παράδειγμα
```
# include <stdio.h>
#define RED 0
#define GREEN 1
int main()
{
    int a = GREEN;
    int b = GREEN;
    if  (a == b)
        printf("τα φανάρια είναι πράσινα. Περάστε γρήγορα !");
    return 0;
}
```

5.4 Δομή Ελέγχου

Για την επίλυση πολύπλοκων προβλημάτων πολλές φορές καθίσταται αναγκαίο κάποιες εντολές του προγράμματος να μην εκτελούνται με τη σειρά που είναι γραμμένες.

Η **δομή ελέγχου** είναι η δομή που χρησιμοποιείται για την επιλογή μιας επεξεργασίας μεταξύ δύο ή περισσότερων διαφορετικών διαδικασιών ανάλογα με την τιμή μιας λογικής συνθήκης.

Η δομή ελέγχου χρησιμοποιείται με δύο διαφορετικές εκδοχές : την **απλή εντολή έλεγχου** και τη **σύνθετη εντολή ελέγχου**.

5.5 Απλή Εντολή Ελέγχου

Η σύνταξη της εντολής είναι:

```
if (συνθήκη ελέγχου)
{
εντολή 1
εντολή 2
        ...
εντολή ν
}
```

Τα άγκιστρα έναρξης (**{**) και τερματισμού (**}**) πρέπει να υπάρχουν οπωσδήποτε στην περίπτωση που η εντολή ελέγχου «αφορά» την εκτέλεση ή όχι μιάς ομάδας εντολών. Στην περίπτωση που η ομάδα εντολών αποτελείται από μία μόνο εντολή τότε αυτά μπορεί να παραλείπονται.

Λειτουργία της εντολής
Αν η συνθήκη ελέγχου έχει τιμή ΑΛΗΘΗΣ, εκτελείται η ομάδα εντολών που βρίσκεται μεταξύ του άγκιστρου έναρξης και του άγκιστρου τερματισμού. Διαφορετικά, η συγκεκριμένη ομάδα εντολών αγνοείται και ή ροή του προγράμματος μεταφέρεται στην εντολή που ακολουθεί μετά το άγκιστρο τερματισμού.
Είναι πολύ σημαντικό να κατανοήσουμε από την αρχή ότι η δομή ελέγχου διακόπτει τη γραμμική σειρά ανάμεσα στις εντολές του προγράμματος και στην εκτέλεσή τους. Σε κάθε περίπτωση, για μία είσοδο δεδομένων θα ακολουθηθεί μόνο μία από τις δυνατές διαδρομές.

Άσκηση 3.4.1
Να γραφεί πρόγραμμα που θα ζητάει από το πληκτρολόγιο έναν αριθμό και θα υπολογίζει και θα εμφανίζει την απόλυτη τιμή του. Η άσκηση να λυθεί χωρίς την χρήση έτοιμης συνάρτησης υπολογισμού της απόλυτης τιμής ενός αριθμού.

```
#include <stdio.h>
int main()
{
int num;
    printf("Δώσε έναν αριθμό : ");
```

```
scanf("%d",&num);
if (num < 0)
        num *= -1;
        printf("Η απόλυτη τιμή του αριθμού που έδωσες είναι : %d",num);
    return 0;
}
```

Άσκηση 3.4.2

Σε τρείς διαφορετικούς αγώνες πρόκρισης για την Ολυμπιάδα της Αθήνας στο άλμα εις ύψος ένας αθλητής πέτυχε τις επιδόσεις a, b, c. Να γραφεί πρόγραμμα το οποίο :
α. θα διαβάζει τις τιμές των επιδόσεων a, b, c,
β. θα υπολογίζει και θα εμφανίζει τη μέση τιμή των παρακάτω τιμών
γ. θα εμφανίζει το μήνυμα «Προκρίθηκε», αν η παραπάνω μέση τιμή είναι μεγαλύτερη των 8 μέτρων.

```
#include <stdio.h>
int main()
{
    float a, b, c, mo;
printf("Δώσε τις τιμές των τριών επιδόσεων : ");
scanf("%f %f %f",&a,&b,&c);
mo = (a + b + c)/3;
printf("Η μέση τιμή των τριών επιδόσεων είναι %.2f\n",mo);
if (mo > 8)
                printf("Προκρίθηκε.\n");
return 0;
}
```

'Ασκηση 3.4.3

Σε τρείς διαφορετικές θέσεις της Λάρισας μετρήθηκαν το μήνα Ιανουάριο τρείς διαφορετικές θερμοκρασίες a, b, c. Να γραφεί πρόγραμμα το οποίο :
α. διαβάζει τις τιμές a,b και c.
β. εμφανίζει τη μέση τιμή των θερμοκρασιών.
γ. εμφανίζει το μήνυμα «Παγωνιά», αν η μέση τιμή της θερμοκρασίας είναι μικρότερη από το μηδέν (0).
δ. υπολογίζει και εμφανίζει τη μεγαλύτερη και τη μικρότερη από τις θερμοκρασίες αυτές.

```
#include <stdio.h>
int main()
{
float a, b, c, mo, max, min;
printf("Δώσε τις τρείς θερμοκρασίες : ");
scanf("%f %f %f",&a,&b,&c);
mo = (a + b + c) / 3;
printf("Η μέση τιμή των θερμοκρασιών είναι : %.2f\n",mo);
if (mo < 0)
        printf("Παγωνιά.\n");
max = a;
if (b > max)
    max = b;
if (c > max)
```

```
                max = c;
min = a;
if (b < min)
                min = b;
if (c < min)
                min = c;
printf("Η μεγαλύτερη θερμοκρασία είναι %.2f\n",max);
printf("Η μικρότερη θερμοκρασία είναι %.2f\n",min);
return 0;
}
```

'Ασκηση 3.4.4

Ένας επενδυτής θέλει να οργανώσει τη διαδικασία διαχείρισης των μετοχών του. Για τον σκοπό αυτό χρειάζεται πρόγραμμα το οποίο :
α. θα ζητάει το αριθμό των μετοχών που έχει ο επενδυτής, την τιμή αγοράς ανά μονάδα και την τρέχουσα τιμή της μετοχής σε €,
β. θα υπολογίζει και θα εμφανίζει το τρέχον ποσοστό κέρδους %,
γ. αν το ποσοστό κέρδους είναι μεγαλύτερο από 30%, θα εμφανίζει το μήνυμα «Πώληση μετοχών», και θα υπολογίζει και θα εκτυπώνει το κέρδος σε €.

```
#include <stdio.h>
int main()
{
int m;  // m -> ο αριθμός των μετοχών
float a,t,p,k;
    // a -> τιμή αγοράς, t -> τρέχουσα τιμή, p -> ποσοστό κέρδους, k -> κέρδος
    printf("Δώσε τον αριθμό των μετοχών : ");
scanf("%d",&m);
printf("Δώσε την τιμή αγοράς ανά μονάδα και την τρέχουσα τιμή : ");
scanf("%f %f",&a,&t);
p = 100 * (t - a) / a;
printf("Ποσοστό κέρδους = %.2f %\n",p);
if (p > 30)
{
        printf("Πώληση μετοχών\n");
        k = p * m * a / 100;
        printf("Κέρδος %.2f €\n",k);
}
return 0;
}
```

Άσκηση 3.4.5

Δίνεται ένας τριψήφιος ακέραιος αριθμός από το πληκτρολόγιο. Να γραφεί πρόγραμμα που θα εμφανίζει το μήνυμα «Καρκινικός αριθμός» αν ο συγκεκριμένος αριθμός είναι καρκινικός αριθμός (Καρκινικοί αριθμοί είναι οι αριθμοί που διαβάζονται ίδια από αριστερά προς τα δεξιά και από δεξιά προς τα αριστερά. (Παράδειγμα ο αριθμός 303, 909).

```
#include <stdio.h>
int main()
{
int num, a, b, c, number;
```

```
printf("Δώσε έναν τριψήφιο ακέραιο αριθμό ");
scanf("%d",&num);
c = num / 100;
b = (num % 100) / 10;
a = (num % 100) % 10;
number = a * 100 + b * 10 + c;
if (num == number)
        printf("Καρκινικός Αριθμός\n");
return 0;
}
```

5.5 Σύνθετη Εντολή Ελέγχου

Η εντολή **if - else** αναφέρεται και ως σύνθετη εντολή ελέγχου. Με την εντολή αυτή επιλέγεται μία από τις δύο επεξεργασίες που μπορούν να γίνουν, ανάλογα με την τιμή μιας συνθήκης ελέγχου. Η σύνταξη της εντολής είναι:

```
if (συνθήκη ελέγχου)
        ομάδα εντολών 1
else
ομάδα εντολών 2
```

Λειτουργία της εντολής
Αν ισχύει η συνθήκη ελέγχου εκτελείται η **ομάδα εντολών 1**. Διαφορετικά εκτελείται η **ομάδα εντολών 2**.

Εμφωλευμένες δομές ελέγχου
Η γενική μορφή της εντολής **if - else** καλύπτει την επιλογή μιας από δύο εναλλακτικές περιπτώσεις. Όταν οι εναλλακτικές περιπτώσεις είναι περισσότερες από δύο, τότε μπορούν να χρησιμοποιηθούν πολλές εντολές if η μία μέσα στην άλλη, ***οι εμφωλευμένες δομές ελέγχου***, όπως ονομάζονται.

Παράδειγμα

```
#include <stdio.h>
int main()
{
        float b,y;
        printf ("Δώσε το βάρος και το ύψος του ατόμου : ");
        scanf("%f %f",&b,&y);
        if (b < 80)
                if (y < 1.80)
                        printf("Κανονικό άτομο.\n");
        return 0;
}
```

Πολύ συχνά οι εντολές που έχουν γραφεί με εμφωλευμένα **if**, μπορούν να γραφούν πιο απλά χρησιμοποιώντας σύνθετες εκφράσεις. Το προηγούμενο παράδειγμα μπορεί να γραφεί ως εξής :

Παράδειγμα

```
#include <stdio.h>
int main()
{
        float b,y;
        printf ("Δώσε το βάρος και το ύψος του ατόμου : ");
        scanf("%f %f",&b,&y);
        if (b < 80 && y < 1.80)
                printf("Κανονικό άτομο.\n");
        return 0;
}
```

Μια άλλη μορφή ελέγχου είναι η εντολή **if - else if - else**. Η σύνταξη της εντολής είναι :

```
if (συνθήκη ελέγχου 1)
        ομάδα εντολών 1
else if (συνθήκη ελέγχου 2)
        ομάδα εντολών 2
…
else
        ομάδα εντολών ν
```

Λειτουργία της εντολής

Όταν κάποια από τις συνθήκες ελέγχου είναι αληθής, εκτελούνται οι εντολές που βρίσκονται στο αντίστοιχο τμήμα.

Παράδειγμα

```
#include <stdio.h>
int main()
{
        int x;
        printf("Δώσε έναν ακέραιο αριθμό : ");
        scanf("%d",&x);
        if (x > 0)
                printf("Ο αριθμός είναι θετικός.\n");
        else if (x < 0)
                printf("Ο αριθμός είναι αρνητικός.\n");
        else
                printf("Ο αριθμός είναι το μηδέν.\n");
        return 0;
}
```

Άσκηση 3.5.1

Δίνονται δύο αριθμοί από το πληκτρολόγιο. Να βρεθεί και να εμφανιστεί στην οθόνη ο μεγαλύτερος από αυτούς.

```
#include <stdio.h>
int main()
{
        int a,b;
```

```
        printf("Δώσε δύο αριθμούς : ");
        scanf("%d %d",&a,&b);
        if (a > b)
                printf("Μεγαλύτερος είναι ο %d\n",a);
        else
                printf("Μεγαλύτερος είναι ο %d\n",b);
        return 0;
}
```

Όπως αναφέραμε και προηγουμένως, κάθε κλάδος της δομής ελέγχου μπορεί να περιέχει οποιεσδήποτε εντολές. Συνεπώς μπορεί να περιλαμβάνει και άλλες δομές ελέγχου. Οι δομές αυτές όπως αναφέραμε λέγονται **εμφωλευμένες δομές ελέγχου**. Είναι πιο περίπλοκες και χρησιμοποιούνται σε περιπτώσεις σύνθετων ή διαδοχικών ελέγχων.

Άσκηση

Να γραφεί πρόγραμμα το οποίο θα ζητάει τις τιμές των παραμέτρων α και β και θα επιλύει την πρωτοβάθμια εξίσωση ax + b = 0.

Η διερεύνηση της πρωτοβάθμιας εξίσωσης από τα μαθηματικά είναι :

Αν $\alpha \neq 0$	Η εξίσωση έχει μοναδική λύση $x = -\frac{b}{a}$
Αν $\alpha = 0$	και $\beta \neq 0$, η εξίσωση είναι αδύνατη
	και $\beta = 0$, η εξίσωση είναι αόριστη

```
#include <stdio.h>
int main()
{
        float a,b,x;
        printf("Πληκτρολόγησε τις τιμές των παραμέτρων α και β : ");
        scanf("%f %f",&a,&b);
        if (a != 0)
        {
                x = -b /a;
                printf("Η λύση της εξίσωση είναι %.2f\n",x);
        }
        else
                if (b != 0)
                        printf("Η εξίσωση είναι αδύνατη.\n");
                else
                        printf("Η εξίσωση είναι αόριστη.\n");
        return 0;
}
```

Η παραπάνω άσκηση μπορεί να λυθεί και αλλιώς αν υοθετήσουμε το στυλ γραφής της C (αρκεί να θυμηθούμε ότι η τιμή μηδέν είναι ΨΕΥΔΗΣ και κάθε τιμή διάφορη του μηδενός ΑΛΗΘΗΣ).

```
#include <stdio.h>
int main()
{
        float a,b,x;
```

```
        printf("Πληκτρολόγησε τις τιμές των παραμέτρων α και β : ");
        scanf("%f %f",&a,&b);
        if (a)    // αν το α <> 0 τότε είναι ΑΛΗΘΗΣ
        {
                x = -b /a;
                printf("Η λύση της εξίσωση είναι %.2f\n",x);
        }
        else
                if (b)     // αν το b <> 0 τότε είναι ΑΛΗΘΗΣ
                        printf("Η εξίσωση είναι αδύνατη.\n");
                else
                        printf("Η εξίσωση είναι αόριστη.\n");
        return 0;
}
```

Άσκηση 3.5.3

Ένας καταθέτης έχει στο λογαριασμό του 30000 €. Να γραφεί πρόγραμμα το οποίο :
α. θα ζητάει το ποσό ανάληψης,
β. θα υπολογίζει και θα εμφανίζει το υπόλοιπο του λογαριασμού,
γ. θα εμφανίζει το μήνυμα «Η συναλλαγή είναι αδύνατη», αν το ποσό ανάληψης είναι μεγαλύτερο από το ποσό κατάθεσης.

```
 #include <stdio.h>
#define POSO 30000
int main()
{
        float a,y; // a -> ποσό ανάληψης, y -> υπόλοιπο λογαριασμού
        printf("Δώσε το ποσό ανάληψης :  ");
        scanf("%f",&a);
        y = POSO – a;
        if ( y > 0)
                printf("Το υπόλοιπο του λογαριασμού σας είναι %.2f €\n",y);
        else
                printf("Η συναλλαγή είναι αδύνατη.\n");
        return 0;
}
```

Άσκηση 3.5.4

Ένα γραφείο ενοικίασης αυτοκινήτων εφαρμόζει την τιμολογιακή πολιτική που φαίνεται στον πίνακα που ακολουθεί. Να αναπτύξετε πρόγραμμα το οποίο :
α. θα διαβάζει τον αριθμό χιλιομέτρων ενός πελάτη,
β. θα υπολογίζει τη χρεώση του πελάτη ανάλογα με τον αριθμό χιλιομέτρων που έκανε,
γ. να εμφανίζει τη λέξη «Χρέωση» και τη χρεώση του πελάτη.

Πάγιο 29,35 €	
Αριθμός χιλιομέτρων (Km)	Χρέωση (€ / Km)
1-100	0,13
101-1000	0,16
1001 και άνω	0,18

```
#include <stdio.h>
```

```
#define PAGIO 29.35
int main()
{
        int Km;         //τα χιλιόμετρα του πελάτη
        float x; //η χρεωσή του
        printf("Δώσε τα χιλιόμετρα του πελάτη (τιμή > 0) : ");
        scanf("%d",&Km);
        if (Km <= 100)
                        x = PAGIO + Km * 0.13;
        else if (Km <= 1000)
                                x = PAGIO + Km * 0.16;
        else
                x = PAGIO + Km * 0.18;
        printf("Χρέωση = %.2f €\n",x);
        return 0;
}
```

Άσκηση 3.5.5

Για τον υπολογισμό του ανωτάτου επιτρεπτού ορίου σφυγμών ενός ανθρώπου αρκεί να αφαιρέσουμε την ηλικία του από το 200. Είναι επίσης γνωστό ότι ο αριθμός σφυγμών αυξάνεται ανάλογα με τη δραστηριότητα ως εξής :

βάδισμα	αύξηση κατά 30 %
ποδηλασία	αύξηση κατά 70 %
τρέξιμο	αύξηση κατά 100 %

Να γραφεί πρόγραμμα το οποίο :

α. θα διαβάζει την ηλικία ενός ατόμου και τον αριθμό των σφυγμών του σε κατάσταση ηρεμίας,

β. θα εμφανίζει τον αριθμό των σφυγμών του ατόμου σε κατάσταση ηρεμίας,

γ. θα υπολογίζει και θα εμφανίζει το ανώτατο επιτρεπτό όριο σφυγμών και τον αναμενόμενο αριθμό σφυγμών για κάθε μία από τις παραπάνω δραστηριότητες,

δ. θα εμφανίζει κατάλληλο μήνυμα για το ποιες δραστηριότητες επιτρέπονται για το συγκεκριμένο άτομο.

```
#include <stdio.h>
int hl,sf,asf;
float sft,sfp,sfb;
/*      hl -> ηλικία ατόμου
        sf -> αριθμός σφυγμών σε κατάσταση ηρεμίας
        asf -> ανώτατο επιτρεπτό όριο σφυγμών
        sfb -> αριθμός σφυγμών κατά το βάδισμα
        sfp -> αριθμός σφυγμών κατά την ποδηλασία
        sft -> αριθμός σφυγμών κατά το τρέξιμο
*/
int main()
{
printf("Δώσε την ηλικία του ατόμου : ");
scanf("%d",&hl);
printf("Δώσε τν αριθμό των σφυγμών σε κατάσταση ηρεμίας : ");
scanf("%d",&sf);
asf = 200 – hl;
sfb = sf + sf * 30  / 100 ;
```

```
sfp = sf + sf * 70 / 100;
sft = sf + sf * 100 / 100;
printf("Αριθμός σφυγμών σε κατάσταση ηρεμίας : %d\n",sf);
printf("Ανώτατο επιτρεπτό όριο σφυγμών : %d\n",asf);
printf("Αριθμός σφυγμών κατά το βάδισμα : %.2f\n",sfb);
printf("Αριθμός σφυγμών κατά την ποδηλασία : %.2f\n",sfp);
printf("Αριθμός σφυγμών κατά το τρέξιμο : %.2f\n",sft);
if (asf >= sfb)
        if (asf >= sfp)
                if (asf >= sft)
                        printf("Επιτρέπονται όλες οι δραστηριότητες.\n");
                else
                        printf("Επιτρέπεται το βάδισμα και το ποδήλατο.\n");
        else
                printf("Επιτρέπεται το βάδισμα.\n");
else
        printf("Δεν επιτρέπεται καμιά δραστηριότητα.\n");
return 0;
}
```

Άσκηση 3.5.6

Σ' ένα κέντρο αδυνατίσματος χρειάζεται, για τον καθορισμό των προγραμμάτων που πρόκειται να ακολουθήσουν οι πελάτες του, η τιμή του Δείκτη Μάζας Σώματος (ΔΜΣ) κάθε ατόμου. Ο δείκτης αυτός δίνεται από τη σχέση :

$$\Delta\text{ΜΣ} = \frac{\text{Βάρος}(Kgr)}{\text{Ύψος}^2(m)}$$

Να δημιουργηθεί πρόγραμμα το οποίο :
α) θα διαβάζει το βάρος και το ύψος ενός ατόμου,
β) θα υπολογίζει και θα εμφανίζει τον Δείκτη Μάζας Σώματος του ατόμου,
γ) θα εμφανίζει ένα από τα παρακάτω μηνύματα ανάλογα με την τιμή του Δείκτη Μάζας Σώματος :

Μήνυμα	**ΔΜΣ**
Αδύνατος	ΔΜΣ < 18.5
Φυσιολογικός	18.5 ≤ ΔΜΣ < 25
Υπέρβαρος	25 ≤ ΔΜΣ < 30
Παχύσαρκος	ΔΜΣ ≥ 30

```
#include <stdio.h>
#include <math.h> // σ' αυτήν την βιβλιοθήκη «ορίζεται» η συνάρτηση pow
int main()
{
        float b,y,dms; // b -> βάρος, y -> ύψος, dms -> δείκτης μάζας σώματος
        printf("Δώσε το βάρος του ατόμου : ");
        scanf("%f",&b);
        printf("Δώσε το ύψος του ατόμου : ");
        scanf("%f",&y);
        dms = b / pow(y,2.0); // σύνταξη  -> double pow(double base,double exp)
        printf("Δείκτης Μάζας Σώματος : %4.2f\n",dms);
        if (dms < 18.5)
```

```
                printf("Αδύνατος.\n");
        else if (dms < 25)
                printf("Φυσιολογικός.\n");
        else if (dms < 30)
                printf("Υπέρβαρος.\n");
        else
                printf("Παχύσαρκος.\n");
        return 0;
}
```

Άσκηση 3.5.7

Να γίνει πρόγραμμα που να λύνει και να διερευνά τη λύση μιας δευτεροβάθμιας εξίσωσης $ax^2 + \beta x + \gamma = 0$ στο σύνολο των πραγματικών αριθμών. Το πρόγραμμα θα δέχεται σαν είσοδο τους συντελεστές α,β και γ της εξίσωσης και θα τυπώνει τις ρίζες της, αν υπάρχουν ή κάποιο μήνυμα που μας πληροφορεί ότι η εξίσωση δεν έχει λύση ή ότι είναι αόριστη.

```
#include <stdio.h>
#include <math.h>
int main()
{
        float a,b,c,x1,x2,D;
        printf("Δώσε τους συντελεστές α,β και γ : ");
        scanf("%f %f %f",&a,&b,&c);
        if (a == 0)
                printf("Η εξίσωση είναι πρωτοβάθμια.\n");
        else  {
                D = pow(b,2.0) – 4 * a * c;
        if (D > 0)  {
        x1 = (-b + sqrt(D)) / (2 * a);
        x2 = (-b – sqrt(D)) / (2 * a);
        printf("Οι ρίζες της εξίσωσης είναι %.2f και %.2f\n",x1,x2);
                }
                else if (D == 0) {
                                        x1 = -b / (2 * a);
                        printf("Μία ρίζα διπλή %.2f\n",x1);
                }
        else
                 printf("Η εξίσωση δεν έχει ρίζες στους πραγματικούς αριθμούς.\n");
}
        return 0;
}
```

Στην παραπάνω άσκηση χρησιμοποιήθηκαν οι μαθηματικές συναρτήσεις **pow( )** και **sqrt( )**. Όλες οι μαθηματικές συναρτήσεις προϋποθέτουν την ύπαρξη του αρχείου επικεφαλίδας **math.h** στο πρόγραμμα μας.
Η σύνταξη των εντολών είναι :
double pow(double base,double exp)
Η συνάρτηση **pow( )** επιστρέφει το base υψωμένο στη δύναμη exp ($base^{exp}$).
double sqrt(double num)

Η συνάρτηση **sqrt( )** επιστρέφει την τετραγωνική ρίζα του num. Αν κάνουμε κλήση της **sqrt( )** με αρνητικό όρισμα, θα συμβεί λάθος πεδίου ορισμού

.

Άσκηση 3.5.8

Μιά εταιρία διαθέτει στους υπαλλήλους της ειδικές κάρτες με τις οποίες μπορούν να προμηθεύονται καφέδες και αναψυκτικά από ειδικούς αυτόματους πωλητές. Να αναπτύξετε πρόγραμμα το οποίο :
α. θα δέχεται την επιλογή προϊόντος και την επιθυμητή ποσότητα,
β. θα υπολογίζει και θα εμφανίζει το κόστος αγοράς,
γ. θα διαβάζει το διαθέσιμο υπόλοιπο της κάρτας. Σε περίπτωση που το υπόλοιπο δεν επαρκεί θα εμφανίζει ανάλογο μήνυμα, ενώ αν το υπόλοιπο επαρκεί θα υπολογίζει και θα εμφανίζει το νέο υπόλοιπο της κάρτας, μετά την αγορά των προϊόντων.

Στον παρακάτω πίνακα εμφανίζονται τα προϊόντα που προσφέρει κάθε μηχάνημα και οι αντίστοιχες τιμές τους :

Προϊόν	**Τιμή σε €**
Ελληνικός καφές	0,50
Καφές espresso	0,90
Πορτοκαλάδα – Λεμονάδα	0,60
Εμφιαλωμένο νερό	0,40

```
#include <stdio.h>
int main()
{
        int e, p; // e -> επιλογή προϊόντος, p -> ποσότητα
        float y,k; // y -> διαθέσιμο υπόλοιπο κάρτας, k -> κόστος προϊόντων
        printf("1. Ελληνικός καφές - 0,50 €\n");
        printf("2. Καφές espresso – 0,90 €\n");
        printf("3. Πορτοκαλάδα – Λεμονάδα – 0,60 €\n");
        printf("4. Εμφιαλωμένο νερό – 0,40 €\n");
        printf(" Δώσε την επιλογή σου (1-4) : ");
        scanf("%d",&e);
        printf("Δώσε την ποσότητα του προϊόντος : ");
        scanf("%d",&p);
        printf("Δώσε το υπόλοιπο της κάρτας : ");
        scanf("%f",&y);
        if (e == 1)
                k = p * 0.50;
        else if (e == 2)
                k = p * 0.90;
        else if (e == 3)
                k = p * 0.60;
        else
                k = p * 0.40;
        printf("Κόστος αγοράς : %.2f\n",k);
        if (k > y)
                printf("Το υπόλοιπο της κάρτας δεν επαρκεί.\n");
        else {
                y = y – k;
                printf("Ολοκλήρωση της συναλλαγής. Νέο υπόλοιπο %.2f\n",y);
        }
    return 0;
```

```
}
```

'Ασκηση 3.5.9

Να αναπτυχθεί πρόγραμμα το οποίο θα δέχεται ένα έτος και θα ελέγχει αν αυτό είναι δίσεκτο ή όχι, εμφανίζοντας το αντίστοιχο μήνυμα. Ένα έτος είναι δίσεκτο εφόσον διαιρείται ακριβώς με το 4. Αν όμως είναι επαιώνιο, θα πρέπει να διαιρείται ακριβώς με το 400. Επαιώνια είναι τα έτη που διαιρούνται ακριβώς με το 100 (Για παράδειγμα 1700, 1800, 2000 κ.α.).

```
#include <stdio.h>
int main()
{
	enum bool {false,true};
	enum bool disekto;
	int etos;
	printf("Δώσε ένα έτος : ");
	scanf("%d",&etos);
	if (etos % 100 == 0)
		if (etos % 400 == 0)
			disekto = true;
		else
			disekto = false;
	else
		if (etos % 4 == 0)
			disekto = true;
		else
			disekto = false;
	if (disekto == true)
		printf("Το έτος %d είναι δίσεκτο.\n",etos);
	else
		printf("Το έτος %d δεν είναι δίσεκτο.\n",etos);
	return 0;
}
```

Άσκηση 3.5.10

Μια εταιρεία εμπορίας αυτοκινήτων πρότεινε σε έναν υποψήφιο αγοραστή τα παρακάτω προγράμματα χρηματοδότησης :
$1^{ο}$ πρόγραμμα : Προκαταβολή του 40% της αρχικής αξίας του αυτοκινήτου και 12 δόσεις ίσες με το 1/18 της αρχικής αξίας του αυτοκινήτου.
$2^{ο}$ πρόγραμμα : Προκαταβολή του 50% της αρχικής αξίας του αυτοκινήτου και 24 δόσεις ίσες με το 1/40 της αρχικής αξίας του αυτοκινήτου.
Να γραφεί πρόγραμμα το οποίο :
α. θα διαβάζει την αρχική αξία του αυτοκινήτου και
β. θα τυπώνει το μήνυμα «$1^{ο}$ Πρόγραμμα» ή «$2^{ο}$ Πρόγραμμα», ανάλογα με το ποιο από αυτά είναι προσφορότερο για τον υποψήφιο αγοραστή (δηλαδή αντιστοιχεί στο μικρότερο συνολικό ποσό πληρωμής).

```
#include <stdio.h>
int main()
{
	float a, prog1, prog2; //a -> αρχική αξία του αυτοκινήτου
	printf("Δώσε την αρχική αξία του αυτοκινήτου : ");
```

```
        scanf("%f",&a);
        prog1 = a * 40 / 100 + 12 * a / 18;
        prog2 = a * 50 / 100 + 24 * a / 40;
        if (prog1 < prog2)
                printf("1ο Πρόγραμμα.\n");
        else
                printf("2ο Πρόγραμμα.\n");
        return 0;
}
```

5.6 Παραστάσεις υπό Συνθήκη

Ένας διαφορετικός τρόπος για να εκφράσουμε μια πρόταση της μορφής **if - else** είναι με τη χρήση του τελεστή υπό συνθήκη. Η σύνταξη της εντολής είναι:

συνθήκη ελέγχου **?** *παράσταση 1* **:** *παράσταση 2*

Λειτουργία της εντολής
Εάν η συνθήκη ελέγχου είναι αληθής (όχι μηδέν) τότε όλη η παράσταση υπό συνθήκη έχει την ίδια τιμή με την παράσταση 1. Εάν η συνθήκη ελέγχου είναι ψευδής (μηδέν), τότε ολόκληρη η παράσταση υπό συνθήκη έχει την ίδια τιμή με την παράσταση 2.
▫ Μπορούμε να χρησιμοποιήσουμε την παράσταση υπό συνθήκη όταν έχουμε μια μεταβλητή, στην οποία μπορούν να καταχωρηθούν δύο πιθανές τιμές.
Μια πρόταση **if - else** μπορεί να πετύχει το ίδιο αποτέλεσμα, μ' αυτό του τελεστή υπό συνθήκη. Όμως οι παραστάσεις υπό συνθήκη είναι πιο συμπαγείς και συνήθως οδηγούν σε πιο συμπαγή κώδικα γλώσσας μηχανής.

Παράδειγμα

```
#include <stdio.h>
int main()
{
        int x, y;
        printf("Δώσε έναν ακέραιο αριθμό : ");
        scanf("%d",&y);
        x = (y > 0) ? y : -y;
        printf("Η απόλυτη τιμή του αριθμού %d είναι %d\n",y,x);
        return 0;
}
```

Το παραπάνω παράδειγμα υπολογίζει την απόλυτη τιμή ενός αριθμού y. Αυτό γίνεται με τη χρήση της παράστασης υπό συνθήκη **(y > 0) ? y : -y**. Εάν το y είναι μεγαλύτερο του μηδενός τότε **x = y**, αλλιώς **x = -y**.

Αντίστοιχα η παράσταση υπό συνθήκη μπορεί να αντικατασταθεί με την εντολή **if … else**.

```
if (y > 0)
        x = y;
else
```

```
x = -y;
```

Οι παρενθέσεις δεν χρειάζονται γύρω από την συνθήκη μίας παράστασης υπό συνθήκη, αφού η προτεραιότητα του **?** είναι πολύ χαμηλή, μόλις πάνω απ' αυτήν του τελεστή εκχώρησης τιμής. Ωστόσο συνιστούμε να βάζετε τις παρανθέσεις γιατί κάνουν το τμήμα συνθήκης της παράστασης, πιο ευδιάκριτο.

Παράδειγμα

```
#include <stdio.h>
int main()
{
        int max, a, b;
        printf("Δώσε δύο ακέραιους αριθμούς : ");
        scanf("%d %d",&a,&b);
        max = (a > b) ? a : b;
        printf("Ο μεγαλύτερος αριθμός από τους %d και %d είναι ο %d\n",a,b,max);
        return 0;
}
```

Το παραπάνω παράδειγμα υπολογίζει τον μεγαλύτερο από δύο δοθέντες αριθμούς a και b. Αυτό γίνεται με τη χρήση της παράστασης υπό συνθήκη **(a > b) ? a : b**. Εάν το α είναι μεγαλύτερο του b τότε **max = a**, αλλιώς **max = b**.

Αντίστοιχα η παράσταση υπό συνθήκη μπορεί να αντικατασταθεί με την εντολή **if … else**.

```
if (a > b)
        max = a;
else
        max = b;
```

Άσκηση 3.6.1

Να γράψετε πρόγραμμα το οποίο :
α. θα διαβάζει δύο ακέραιους αριθμούς,
β. αν είναι ο ένας άρτιος και ο άλλος περιττός, να εμφανίζει το άθροισμά τους, αλλιώς να εμφανίζει τη διαφορά τους.

```
#include <stdio.h>
 int main()
{
        int a, b, result;
        printf("Δώσε δύο ακέραιους αριθμούς : ");
        scanf("%d %d",&a,&b);
        result = (a % 2 == 0 && b % 2 != 0) ? a + b : a – b;
        printf("Το αποτέλεσμα είναι %d\n",result);
        return 0;
}
```

Άσκηση 3.6.2

Να γράψετε πρόγραμμα το οποίο :
α. θα διαβάζει δύο ακέραιους αριθμούς a και b,

β. θα υπολογίζει τη διαφορά του μικρότερου από το μεγαλύτερο,
γ. θα ελέγχει αν η διαφορά αυτή είναι άρτιος ή περιττός αριθμός και θα εμφανίζει το σχετικό αποτέλεσμα.

```
#include <stdio.h>
int main()
{
        int a, b, result;
        printf("Δώσε δύο ακέραιους αριθμούς : ");
        scanf("%d %d",&a,&b);
        result = (a > b) ? a – b : b – a;
        printf("Το αποτέλεσμα είναι %s.\n",(result % 2 == 0) ? "άρτιος":"περιττός");
        return 0;
}
```

5.7 Πολλαπλή Επιλογή (switch και break)

Με τη χρήση της δομής **if - else** είναι εύκολο να γράψουμε προγράμματα που επιλέγουν μεταξύ δύο εναλλακτικών περιπτώσεων. Στην περίπτωση όμως που θέλουμε να επιλέγουμε μεταξύ πολλών εναλλακτικών περιπτώσεων, μπορούμε να χρησιμοποιήσουμε τη δομή **if - else if - else**, αλλά σε πολλές περιπτώσεις είναι πιο κατάλληλο να χρησιμοποιούμε την εντολή **switch**. Η σύνταξη της εντολής είναι:

```
switch (επιλογέας)
{
      case τιμή 1       :   ομάδα εντολών 1;
                            break;
      case τιμή 2       :   ομάδα εντολών 2;
                            break;
      ...
      case τιμή ν       :   ομάδα_εντολών ν;
                            break;
      default           :   ομάδα εντολών default;
                            break;
}
```

Λειτουργία της εντολής

⇒ «Αποτιμάται» η τιμή του επιλογέα.
⇒ Ανάλογα με την τιμή του επιλογέα (κάποια τιμή από τις τιμή 1, τιμή 2, … , τιμή ν), επιλέγεται μία από τις εναλλακτικές ομάδες εντολών και εκτελείται.
⇒ Αν η τιμή του επιλογέα δεν αντιστοιχεί σε καμία περίπτωση, τότε εκτελούνται οι εντολές μετά τη λέξη **default**.
⇒ Η εντολή **break** εξαναγκάζει το πρόγραμμα να διακόψει την πρόταση **switch** και να παραλείψει τις επόμενες προτάσεις μετά τη **switch**. Χωρίς την εντολή **break**, θα εκτελεστεί κάθε πρόταση από την τιμή ταύτισης (τιμή 1, … , τιμή ν) μέχρι και το τέλος της πρότασης **switch**.

Κανόνες

Ο επιλογέας πρέπει να είναι μια μεταβλητή ή παράσταση **βαθμωτού τύπου**. Παίρνει δηλαδή **διακριτές τιμές** (ακέραιες ή και απλούς χαρακτήρες) και συνεπώς δεν μπορεί να είναι πραγματικού τύπου.

Παράδειγμα

```
#include <stdio.h>
int main()
{
        int num;
        printf("Δώσε έναν αριθμό : ");
        scanf("%d",&num);
        switch (num % 2)
        {
                case 0 :printf("Ο αριθμός είναι άρτιος.\n");
                                break;
                case 1 :printf("Ο αριθμός είναι περιττός.\n");
                                break;
        }
        return 0;
}
```

⇒ Οι τιμές στις λίστες πρέπει να είναι του ίδιου τύπου με τον επιλογέα.

⇒ Μία λίστα μπορεί να αποτελείται από μία ή περισσότερες τιμές χωρισμένες με την ετικέττα **case** τιμη**:**

Παράδειγμα

```
#include <stdio.h>
int main()
{
        char ch;
        printf(" a. Εισαγωγή\n");
        printf(" β. Διόρθωση\n");
        printf(" γ. Διαγραφή\n");
        printf("Δώσε την επιλογή σου (α – γ) : ");
        scanf("%c",&ch);
        switch (ch)
        {
                case 'α' :
                case 'A':       printf("Επέλεξες το α ή A.\n");
                                break;
                case 'β' :
                case 'B' :      printf("Επέλεξες το β ή B.\n");
                                break;
                case 'γ' :
                case 'Γ' :
                                printf("Επέλεξες το γ ή Γ.\n");
                                break;
                default :       printf("Έκανες λάθος επιλογή.\n");
                                break;
```

```
        }
}
```

⇒ Δεν επιτρέπεται να εμφανίζεται η ίδια τιμή σε περισσότερες από μία λίστα τιμών.
⇒ Κατά την εκτέλεση της εντολής πρέπει η τιμή του επιλογέα να υπάρχει σε κάποια από τις λίστες τιμών. Εκτελείται τότε μόνο η ομάδα εντολών που αντιστοιχεί στη λίστα αυτή.
⇒ Σε περίπτωση που η τιμή του επιλογέα δεν υπάρχει σε καμία λίστα, τότε δεν εκτελείται καμία ομάδα εντολών η εκτελείται η ομάδα εντολών της περίπτωσης **default**. Η παράθεση της περίπτωσης **default** στη σύνταξη της εντολής είναι προφανώς, προαιρετική.

Switch ή if - else ;

Πότε πρέπει να χρησιμοποιείται η **switch** και πότε η δομή **if - else** ; Συχνά δεν υπάρχει δυνατότητα επιλογής. Δεν μπορεί να χρησιμοποιηθεί η **switch** αν πρέπει να υπολογιστεί μια μεταβλητή ή μία έκφραση τύπου **float**. Ούτε είναι κατάλληλο να χρησιμοποιηθεί η **switch** όταν μία μεταβλητή πρέπει να βρίσκεται μέσα σε μία συγκεκριμένη περιοχή τιμών. Για παράδειγμα, είναι απλό να γραφεί το ακόλουθο :
if (grade >= 0 && grade <= 100)

Για να γραφεί το παραπάνω τμήμα κώδικα με χρήση της **switch**, θα έπρεπε να δημιουργηθούν λίστες για κάθε ακέραιο από 0 μέχρι 100. Στις περιπτώσεις όμως που μπορεί να χρησιμοποιηθεί η **switch** τα προγράμματα θα «τρέχουν» γρηγορότερα και θα απαιτούν λιγότερο κώδικα.

Άσκηση 3.7.1
Στα Τεχνολογικά Εκπαιδευτικά Ιδρύματα χρησιμοποιείται η δεκάβαθμη κλίμακα βαθμολόγησης. Η επίδοση των σπουδαστών χαρακτηρίζεται ως εξής :
Απόρριψη : βαθμός μικρότερος του 5
Καλώς : βαθμός 5 ή 6
Λίαν καλώς : βαθμός 7 ή 8
Άριστα : βαθμός 9 ή 10
Να γραφεί πρόγραμμα το οποίο θα διαβάζει το βαθμό ενός σπουδαστή σε ένα μάθημα και θα εμφανίζει το αντίστοιχο μήνυμα επίδοσης.

```
#include <stdio.h>
int main()
{
        int bathmos;
        printf("Δώσε την βαθμολογία του σπουδαστή : ");
        scanf("%d",&bathmos);
        switch (bathmos)
        {
                case 5 :
        case 6 :        printf("Καλώς.\n");
                                break;
                case 7 :
        case 8 :printf("Λίαν καλώς.\n");
                                break;
```

```
            case 9 :
        case 10 :       printf("Άριστα.\n");
                        break;
            default :   printf("Απόρριψη.\n");
                        break;
        }
        return 0;
}
```

Άσκηση 3.7.2

Να γραφεί πρόγραμμα το οποίο :
α. θα διαβάζει έναν ακέραιο αριθμό από 0 μέχρι 6,
β. θα εμφανίζει την αντίστοιχη ημέρα της εβδομάδας (0 για Κυριακή, 1 για Δευτέρα, … , 6 για Σάββατο).

```
#include <stdio.h>
int main()
{
        int choise;
        printf("Δώσε έναν ακέραιο αριθμό (0 – 6) : ");
        scanf("%d",&choise);
        switch (choise)
        {
            case 0    : printf("Κυριακή\n");
                        break;
            case 1    : printf("Δευτέρα\n");
                        break;
            case 2    : printf("Τρίτη\n");
                        break;
            case3     : printf("Τετάρτη\n");
                        break;
            case4     : printf("Πέμπτη\n");
                        break;
            case5     : printf("Παρασκευή\n");
                        break;
            case6     : printf("Σάββατο\n");
                        break;
        }
        return 0;
}
```

Άσκηση 3.7.3

Ο παρακάτω πίνακας ισοτιμιών που ακολουθεί δίνει την ισοτιμία του € με τα νομίσματα μερικών χωρών εκτός της ζώνης του Ευρώ.
Να γραφεί πρόγραμμα το οποίο :
α. θα δέχεται στην είσοδο ένα ποσό σε € και έναν αριθμό επιλογής του νομίσματος, από 1 μέχρι 8,
β. θα υπολογίζει και θα εμφανίζει το ισότιμο ποσό στο νόμισμα επιλογής.

Αριθμός νομίσματος	Νομίσματα	Ισοτιμία σε €
1	Δολάριο ΗΠΑ	0,9972
2	Λίρα Αγγλίας	0,6499

3	Φράγκο Ελβετίας	1,4865
4	Γέν Ιαπωνίας	1,1710
5	Κορόνα Νορβηγίας	7,4846
6	Δολάριο Καναδά	1,5545
7	Κορόνα Σουηδίας	9,3741
8	Λίρα Κύπρου	0,5819

```
#include <stdio.h>
int main()
{
        int ep;
        float poso, iso;
        printf("Δώσε το ποσό σε € : ");
        scanf("%f",&poso);
        printf("Δώσε επιλογή νομίσματος (1 – 8) : ");
        scanf("%d",&ep);
        switch (ep)
        {
          case 1  :  iso = poso / 0.9972;
                     break;
          case 2  :  iso = poso / 0.6499;
                     break;
          case 3  :  iso = poso / 1.4865;
                     break;
          case 4  :  iso = poso / 1.1710;
                     break;
          case 5  :  iso = poso / 7.4846;
                     break;
          case 6  :  iso = poso / 1.5545;
                     break;
          case 7  :  iso = poso / 9.3741;
                     break;
          case 8  :  iso = poso / 0.5819;
                     break;
        }
        printf("Η ισοτιμία είναι : %.2f\n",iso);
        return 0;
}
```

5.8 Ασκήσεις

1. Σε τρία διαφορετικά σημεία της Λάρισας καταγράφηκαν στις 12 το μεσημέρι οι θερμοκρασίες α, b, c.
Να γραφεί πρόγραμμα που :
α. θα διαβάζει τις θερμοκρασίες a, b, c.
β. θα υπολογίζει και θα εμφανίζει τη μέση τιμή των παραπάνω θερμοκρασιών.
γ. θα εμφανίζει το μήνυμα «ΚΑΥΣΩΝΑΣ» αν η μέση τιμή είναι μεγαλύτερη των 37 βαθμών Κελσίου.

2. Ένα σύστημα κλιματισμού ελέγχεται μέσω υπολογιστή. Να γραφεί πρόγραμμα το οποίο θα διαβάζει τη θερμοκρασία δωματίου και θα εμφανίζει το κατάλληλο μήνυμα ενεργοποίησης του συστήματος κλιματισμού ως εξής :
α. «ΘΕΡΜΑΝΣΗ», για θερμοκρασία μικρότερη από 20^o C,
β. «ΨΥΞΗ», για θερμοκρασία μεγαλύτερη από 25^o C.

3. Να γραφεί πρόγραμμα το οποίο :
α. θα διαβάζει από το πληκτρολόγιο τις τιμές δύο αντιστάσεων R_1 και R_2 και τον τρόπο σύνδεσής τους, ως εξής :
1 για παράλληλη σύνδεση
2 για σύνδεση σε σειρά
β. θα υπολογίζει και θα εμφανίζει την ολική αντίσταση R.
Δίνονται οι σχέσεις υπολογισμού της ολικής αντίστασης :
Σύνδεση σε σειρά : $R = R_1 + R_2$

Παράλληλη σύνδεση : $R = \dfrac{R_1 R_2}{R_1 + R_2}$

4. Ένας πωλητής έχει βασικό μισθό 700 €, ενώ παίρνει προμήθεια ανά πώληση 1 € για ποσότητα πωλήσεων μεγαλύτερη από 500 Kg και 0,6 € για ποσότητα μικρότερη από 500 Kg. Να γραφεί πρόγραμμα το οποίο :
α. θα ζητά την ποσότητα προϊόντων σε Kg που πούλησε ο πωλητής και
β. θα υπολογίζει και θα εμφανίζει τις μηνιαίες αποδοχές του.

5. Να γράψετε πρόγραμμα το οποίο θα διαβάζει τις γωνίες Α, Β και Γ ενός τριγώνου και θα εμφανίζει στην οθόνη μήνυμα για το είδος του τριγώνου(οξυγώνιο – Α ή Β ή Γ $< 90^o$, αμβλυγώνιο – Α ή Β ή Γ $> 90^o$ ή ορθογώνιο Α ή Β ή Γ $= 90^o$).

6. Σε ένα Μικροβιολογικό Εργαστήριο χρειάζονται πρόγραμμα το οποίο :
α. θα διαβάζει την τιμή της χοληστερίνης HDL του αίματος σε mg/dl,
β. θα εμφανίζει ένα από τα παρακάτω μηνύματα ανάλογα με την τιμή HDL :

Μήνυμα	Τιμές HDL
Επιθυμητή	< 200
Οριακή	200 – 239
Υψηλή	240 – 299
Πολύ υψηλή	> 300

7. Μία εταιρία εισάγει οθόνες υπολογιστών και ακολουθεί την παρακάτω πολιτική χονδρικής πώλησης για τους πελάτες της

Ποσότητα	Τιμή μονάδος σε €
1 – 100	300
101 - 200	280
201 - 300	260
301 και πάνω	240

Να γραφεί πρόγραμμα το οποίο :
α. θα ζητάει από το πληκτρολόγιο την ποσότητα των οθονών,
β. θα υπολογίζει και θα τυπώνει την αξία, τον φόρο προστιθέμενης αξίας (ΦΠΑ) που αναλογεί και το συνολικό κόστος (αξία + ΦΠΑ).
Δίνεται ότι ο συντελεστής ΦΠΑ είναι 18%.

8. Ένας φορολογούμενος φορολογείται σύμφωνα με την παρακάτω κλίμακα :
⇒ για εισόδημα μέχρι 5000 € συντελεστής φόρου 0 %
⇒ για το τμήμα εισοδήματος 5001 – 10000 € συντελεστής φόρου 5 €
⇒ για το τμήμα εισοδήματος 10001 – 20000 € συντελεστής φόρου 15 %
⇒ για το τμήμα εισοδήματος πάνω από 20000 € συντελεστής φόρου 25 %

Να γραφεί πρόγραμμα το οποίο θα ζητάει το ετήσιο εισόδημα του υπαλλήλου και θα εμφανίζει στην οθόνη το φόρο που αναλογεί.

9. Μια εταιρία πουλάει ένα προϊόν με έκπτωση ως εξής :

Ποσότητα	Ποσοστό έκπτωσης
μέχρι 100	0 %
από 100 μέχρι 200	5 %
από 200 μέχρι 400	10 %
πάνω από 400	20 %

Να γραφεί πρόγραμμα το οποίο :
α. θα ζητάει από το πληκτρολόγιο την ποσότητα των προϊόντων και την τιμή μονάδας,
β. θα υπολογίζει και θα τυπώνει το ποσό της έκπτωσης και την τελική αξία των προϊόντων σε €.

Σημείωση ! Η έκπτωση είναι κλιμακωτή και δεν αφορά τη συνολική ποσότητα.

10. Μια βιομηχανία πληρώνει τους εργάτες της προς 5 € την ώρα. Να γραφεί πρόγραμμα το οποίο :
α. θα διαβάζει τον αριθμό ωρών εργασίας του εργάτη ανά εβδομάδα,
β. θα υπολογίζει και θα εμφανίζει τις εβδομαδιαίες αποδοχές του,
Δίνεται ότι κάθε ώρα εργασίας, πάνω από το υποχρεωτικό ωράριο των 40 ωρών, αμοίβεται το διπλάσιο του ωρομισθίου των 5 €.

11. Ένας πωλητής έχει σταθερές μηνιαίες αποδοχές 800 €. Όταν οι εισπράξεις των πωλήσεων που πραγματοποιεί το μήνα είναι κάτω από 10000 € δεν παίρνει προμήθεια. Για πωλήσεις από 10000 μέχρι 20000 παίρνει προμήθεια 5 % επί των εισπράξεων. Για πόσο πάνω από 20000 € η προμήθειά του είναι 8 %. Να γραφεί πρόγραμμα το οποίο :
α. θα ζητάει από το πληκτρολόγιο το ποσό των μηνιαίων πωλήσεων που πραγματοποίησε,
β. θα υπολογίζει την προμήθεια, τις ακαθάριστες αποδοχές, τον φόρο και τον μισθό του υπαλλήλου,
γ. θα εμφανίζει στην οθόνη τα παρακάτω στοιχεία του πωλητή ως εξής :
Βασικός μισθός :
Προμήθεια :
Ακαθάριστα :
Φόρος :
Μισθός :
Δίνεται ότι ο φόρος είναι 10% επί του συνολικού ποσού.

12. Η χρέωση λογαριασμού μια τηλεφωνικής εταιρίας γίνεται με βάση τον παρακάτω πίνακα :

Πάγιο 20 €		
Κόστος αστικής μονάδας	0,03 €	
Κόστος υπεραστικής μονάδας (κλιμακωτή χρέωση)	0 – 200 μονάδες	0,04 €
	201 – 400 μονάδες	0,05 €
	401 μονάδες και πάνω	0,06 €
ΦΠΑ (επί του συνόλου) : 18 %		

Να γραφεί πρόγραμμα το οποίο :
α. θα διαβάζει το πλήθος των αστικών και υπεραστικών μονάδων ενός συνδρομητή,
β. θα υπολογίζει και θα εμφανίζει την αναλυτική χρέωση του λογαριασμού ως εξής :
Πάγιο :
Αστικές κλήσεις :
Υπεραστικές κλήσεις :

ΦΠΑ :
Σύνολο :

13. Με το Διατραπεζικό Σύστημα Συναλλαγών (ΔΙΑΣ) μπορούμε από ένα μηχάνημα ΑΤΜ να κάνουμε συναλλαγές από μια Τράπεζα χρησιμοποιώντας την κάρτα ΑΤΜ μιας άλλης Τράπεζας. Κάθε συναλλαγή ανάληψης χρημάτων χρεώνεται με το ένα εκατοστό του ποσού ανάληψης, ενώ η χρέωση αυτή δεν πρέπει να είναι μικρότερη από 1 € και μεγαλύτερη από 3 €. Να γραφεί πρόγραμμα το οποίο :
α. θα δέχεται στην είσοδο το ποσό ανάληψης σε €,
β. θα υπολογίζει και θα εμφανίζει τη χρέωση του πελάτη της Τράπεζας.

14. Μια τράπεζα πρόκειται να αναβαθμίσει το σύστημα on-line. Για το σκοπό αυτό ζήτησε από μια εταιρία προσφορά για την αγορά νέων υπολογιστών. Η εταιρία έδωσε τον παρακάτω πίνακα :

Ποσότητα Η/Υ	Έκπτωση %
0 – 50	0
51 – 100	5
101 – 150	10
151 και άνω	15

Να γραφεί πρόγραμμα το οποίο :
α. θα διαβάζει την ποσότητα υπολογιστών που χρειάζεται η τράπεζα,
β. θα υπολογίζει και θα εμφανίζει το συνολικό κόστος αναβάθμισης του συστήματος.
Δίνεται ότι η αξία κάθε υπολογιστή είναι 1020 €.

15. Σε ένα βιντεοκλάμπ η ενοικίαση μιας βιντεοκασέτας χρεώνεται ως εξής :
⇒ ελάχιστη χρέωση 1,5 € για διάρκεια δύο ημερών,
⇒ πρόσθετη χρέωση 1 € για κάθε ημέρα καθυστέρησης στην επιστροφή.
Να γραφεί πρόγραμμα το οποίο :
α. θα διαβάζει τον αριθμό των ημερών που κράτησε τη βιντεοκασέτα ένας πελάτης,
β. θα υπολογίζει και να εμφανίζει τη συνολική χρέωση.

16. Να γραφεί πρόγραμμα το οποίο να υπολογίζει τον μηνιαίο λογαριασμό του νερού ως εξής :
α. θα διαβάζει από το πληκτρολόγιο τα κυβικά του νερού που καταναλώθηκαν,
β. θα υπολογίζει και θα τυπώνει την αξία του νερού (δίνεται ότι τα πρώτα 20 κυβικά χρεώνονται προς 2 €, ενώ τα υπόλοιπα προς 3 € το κυβικό),
γ. θα υπολογίζει και θα εμφανίζει το κόστος αποχέτευσης (κάθε κυβικό χρεώνεται με 0,5 €),
δ. θα υπολογίζει και θα εμφανίζει το συνολικό κόστος του λογαριασμού.
Δίνεται ότι το πάγιο ύδρευσης είναι 20 € και το πάγιο αποχέτευσης 15 €. Αν δεν υπάρχει κατανάλωση νερού, τότε χρεώνονται και τα δύο πάγια. Ο ΦΠΑ είναι 8 %.

17. Μια μεταλλοβιομηχανία αγοράζει ένα μεταλλικό εξάρτημα προς 3 € το ένα, για ποσότητα μέχρι 2000 τεμάχια και προς 2,7 € το ένα, για ποσότητα πάνω από 2000 τεμάχια. Μπορεί επίσης να κατασκευάσει το εξάρτημα αυτό με κόστος 1,5 € το ένα, αφού πρώτα προχωρήσει σε πρόσθετη επένδυση 4000 € για την επιμόρφωση των τεχνικών που θα εργαστούν για την παραγωγή του εξαρτήματος. Να γραφεί πρόγραμμα το οποίο :
α. θα ζητάει από το πληκτρολόγιο την ποσότητα των μεταλλικών εξαρτημάτων και θα υπολογίζει το κόστος αγοράς και το κόστος παραγωγής τους,
β. θα τυπώνει το μήνυμα «αγορά» ή «κατασκευή», ανάλογα με το αν συμφέρει την εταιρία να αγοράσει ή να κατασκευάσει το μεταλλικό εξάρτημα.

18. Υποθέστε ότι ο μισθός ενός εργαζομένου προσαυξάνεται κατά 10 € για κάθε παιδί, αν έχει μέχρι τρία παιδιά. Αν έχει περισσότερα από τρία παιδιά η προσαύξηση είναι 20 € για κάθε παιδί. Να γραφεί πρόγραμμα το οποίο :
α. θα διαβάζει το βασικό μισθό και τον αριθμό των παιδιών του εργαζόμενου,
β. θα υπολογίζει το οικογενειακό επίδομα και το μισθό και
γ. θα εμφανίζει το μήνυμα «Μισθός : » και το ποσό του τελικού μισθού.

19. Μια εταιρία εμπορίας αυτοκινήτων ακολουθεί την πολιτική τιμών που δίνεται στον παρακάτω πίνακα :

Αξία αυτοκινήτου	Αριθμός δόσεων	Προσαύξηση επί της αρχικής αξίας %
Από 8.000 € μέχρι 15.000 €	0	0
	6	10
	12	15
	24	20
Από 15.000 € μέχρι 20.000 €	0	0
	6	8
	12	12
	24	18
Πάνω από 20.000 €	0	0
	6	6
	12	10
	24	15

Να γραφεί πρόγραμμα το οποίο :
α. θα διαβάζει την αξία του αυτοκινήτου και τον επιθυμητό αριθμό δόσεων,
β. θα υπολογίζει και θα εμφανίζει το τελικό κόστο του αυτοκινήτου.

20. Ένας σύλλογος ζήτησε από ένα ταξιδιωτικό γραφείο προσφορές για το κόστος διαμονής της 5ήμερης εκδρομής τους στη Κέρκυρα. Το ταξιδιωτικό γραφείο έδωσε τρείς προσφορές για το κόστος του δίκλινου δωματίου ανά ημέρα :

Ξενοδοχείο	Α	Β	Γ
Κόστος	60 €	50 €	45 €

Για την επιλογή της προσφοράς, ο σύλλογος έφτιαξε πρόγραμμα το οποίο :
α. ζητάει τον αριθμό των ατόμων που θα συμμετάσχουν στην εκδρομή και το ξενοδοχείο διαμονής,
β. υπολογίζει και εμφανίζει το συνολικό κόστος διαμονής της εκδρομής.

Σημείωση . Σε κάθε δωμάτιο μένουν δύο άτομα. Οι ημέρες διανυκτέρευσης είναι 4.

6. ΕΠΑΝΑΛΗΠΤΙΚΕΣ ΔΟΜΕΣ

Οι επαναληπτικές δομές επιτρέπουν στον υπολογιστή να επαναλαμβάνει ένα σύνολο εντολών μέχρι να ικανοποιηθεί μια συγκεκριμένη συνθήκη. Οι επαναληπτικές δομές της C είναι οι **while**, **do - while** και **for**.

6.1 Η Επαναληπτική Δομή while

Η γενική μορφή της επαναληπτικής δομής **while** είναι:

while (*συνθήκη*) *εντολή*;

όπου η *εντολή* μπορεί να είναι η κενή εντολή, μία μόνο εντολή, η μία ομάδα εντολών που επαναλαμβάνεται. Η *συνθήκη* μπορεί να είναι οποιαδήποτε αποδεκτή παράσταση.

Λειτουργία της εντολής
Ο βρόχος επαναλαμβάνεται όσο η *συνθήκη* είναι ΑΛΗΘΗΣ (δηλαδή ≠ 0). Μόλις η *συνθήκη* γίνει ΨΕΥΔΗΣ (δηλαδή = 0), ο έλεγχος του προγράμματος περνάει στη γραμμή που βρίσκεται μετά το βρόχο.

Παράδειγμα
```
#include <stdio.h>
int main()
{
        int i = 1,sum = 0;
        while (i != 0) {
                printf("Δώσε έναν ακέραιο αριθμό (0 για τερματισμό) : ");
                scanf("%d",&i);
                sum += i;
        }
        printf("Το άθροισμα των αριθμών που έδωσε είναι %d.\n",sum);
        return 0;
}
```

Το παραπάνω πρόγραμμα ζητά συνεχώς αριθμούς από το πληκτρολόγιο και υπολογίζει το άθροισμά τους. Το πρόγραμμα τερματίζεται (και εμφανίζει το άθροισμα των αριθμών που δόθηκαν) όταν δοθεί ο αριθμός 0.

Οι βρόχοι **while** ελέγχουν την συνθήκη στην αρχή του βρόχου, και αυτό σημαίνει ότι ανάλογα με την τιμή της *συνθήκης* ο κώδικας του βρόχου μπορεί να μην εκτελεστεί ποτέ. Στο παραπάνω παράδειγμα αυτός είναι ο λόγος που χρειάστηκε να δώσουμε αρχική τιμή στην μεταβλητή i.

Παράδειγμα
```
#include <stdio.h>
int main()
{
```

```
        int i=1,n,average,sum = 0;
        printf("Δώσε το πλήθος των αριθμών : ");
        scanf("%d",&n);
        while (i <= n) {
                        sum += i;
                        i++;
        }
        average = sum / n;
        printf("Ο μέσος όρος των αριθμών που έδωσες είναι %d \n",average);
        return 0;
}
```

Το παραπάνω πρόγραμμα υπολογίζει τον μέσο όρο δοθέντος πλήθους αριθμών.

Παράδειγμα

```
#include <stdio.h>
int main()
{
        int grade,max = 0;
        printf("Δώσε ένα ακέραιο αριθμό (αρνητικό για τερματισμό) : ");
        scanf("%d",&grade);
        while (grade > 0) {
                if (grade > max) max = grade;
                printf("Δώσε ένα ακέραιο αριθμό (αρνητικό για τερματισμό) : ");
                scanf("%d",&grade);
        }
        printf("Ο μεγαλύτερος από τους αριθμούς που έδωσες είναι ο %d\n",max);
        return 0;
}
```

Το παραπάνω πρόγραμμα διαβάζει συνεχώς αριθμούς από το πληκτρολόγιο (μέχρι να δοθεί αρνητικός αριθμός, οπότε και τερματίζει) και υπολογίζει και εμφανίζει τον μεγαλύτερο από αυτούς.

Παράδειγμα

```
#include <stdio.h>
int main()
{
        int i=1,j,col,row;
        printf("Δώσε τις γραμμές και τις στήλες : ");
        scanf("%d %d",&row,&col);
        while (i <= row) {
                        j = 1;
while (j <= col) {
        // πρώτη ή τελευταία γραμμή
                                if (i == 1 || i == row) printf("%c",'*');
                                else
                                        // πρώτη ή τελευταία στήλη
        if (j == 1 || j == col) printf("%c",'*');
                                        else printf("%c",' ');
```

```
                        j++;
                }
                printf("\n");
                i++;
        }
        return 0;
}
```

Το παραπάνω πρόγραμμα σχεδιάζει ένα μεταβλητό ορθογώνιο στην οθόνη (ανάλογα με τις τιμές που δίνει ο χρήστης για τις γραμμές και τις στήλες), χρησιμοποιώντας τον χαρακτήρα *. Για να το πετύχουμε αυτό χρησιμοποιούμε δύο εμφωλευμένες δομές επανάληψης όπου πρώτα ολοκληρώνει την εκτέλεσή της η εσωτερική και κατόπιν η εξωτερική αυξάνει την τιμή της κατά 1.

*Ένα σημείο που αξίζει ιδιαίτερη προσοχή, είναι ότι η πρόταση **while** απο μόνη της, ακόμα και αν χρησιμοποιεί σύνθετες προτάσεις, μετράει, από άποψη συντακτικού σαν μία πρόταση. Η πρόταση υπολογίζεται από το **while** μέχρι το πρώτο ελληνικό ερωτηματικό, ή στην περίπτωση σύνθετης πρότασης, μέχρι το τελευταίο άγκιστρο.* Αυτό ακριβώς φαίνεται στο επόμενο παράδειγμα.

Παράδειγμα

```
#include <stdio.h>
int main()
{
        int i = 0;
        while (i++ < 5) ;
        printf("%d\n",i);
        return 0 ;
}
```

Όπως αναφέρθηκε προηγουμένως, όταν δεν υπάρχει σύνθετη πρόταση ο βρόχος τελειώνει στο πρώτο ελληνικό ερωτηματικό. Αφού το ελληνικό ερωτηματικό «ακολουθεί» την **while**, ο βρόχος σταματάει εκεί και άρα η πρόταση **printf( )** δεν είναι μέρος του βρόχου. Επομένως το **i** αυξάνεται σε κάθε επανάληψη αλλά τυπώνεται μόνο μετά την έξοδο από το βρόχο.

6.2 Η Επαναληπτική Δομή do – while

Σε αντίθεση με το βρόχο **while** που ελέγχει τη συνθήκη στην αρχή του, ο βρόχος **do – while** ελέγχει τη συνθήκη στο τέλος του. Αυτό το χαρακτηριστικό προκαλεί πάντοτε την εκτέλεση του βρόχου **do – while** τουλάχιστον μία φορά. Η γενική μορφή της επαναληπτικής δομής **do – while** είναι:

do {
        εντολή;
} **while** (*συνθήκη*);

Παρ' όλο που τα άγκιστρα δεν είναι απαραίτητα όταν υπάρχει μία μόνο εντολή, συνήθως τα χρησιμοποιούμε για να κάνουμε το πρόγραμμα πιο ευανάγνωστο. Η πιο

κοινή ίσως χρήση του **do – while** είναι η ανάγνωση της επιλογής του χρήστη από ένα σύνολο διαθέσιμων επιλογών (μενού επιλογών).Αυτή η ομάδα εντολών πρέπει να εκτελεστεί τουλάχιστον μία φορά. Όταν πληκτρολογήσουμε μια αποδεκτή απάντηση ο βρόχος τερματίζει και επεξεργαζόμαστε την απόκριση του χρήστη, αλλιώς ο βρόχος πρέπει να συνεχίζεται και να ζητάει πάλι την απόκριση του χρήστη. Αυτό ακριβώς φαίνεται στο ακόλουθο παράδειγμα.

Παράδειγμα

```
#include <stdio.h>
int main()
{
	int choise;
	do {
		printf("1. Εισαγωγή\n");
		printf("2. Διόρθωση\n");
		printf("3. Διαγραφή\n");
		printf("4. Έξοδος\n");
		printf("Δώσε την επιλογή σου (1 - 4) : ");
		scanf("%d",&choise);
	} while (choise < 1 || choise > 4);
	printf("Επέλεξες την περίπτωση %d\n",choise);
	return 0;
}
```

Άσκηση 4.2.1

Να γραφεί πρόγραμμα το οποίο θα ελέγχει την εγκυρότητα εισαγωγής ενός ακεραίου αριθμού a > 0.

```
#include <stdio.h>
int main()
 {
	int a;
	do {
		printf(“Δώσε έναν αριθμό > 0 : ”);
		scanf(“%d”,&a);
	} while (a <= 0);
	return 0;
}
```

Στο παραπάνω πρόγραμμα χρησιμοποιήσαμε την επαναληπτική δομή **do – while**. Το ίδιο ακριβώς αποτέλεσμα μπορούμε να επιτύχουμε αν χρησιμοποιήσουμε την επαναληπτική δομή **while**.

```
#include <stdio.h>
int main()
 {
	int a;
	printf(“Δώσε έναν αριθμό > 0 : ”);
	scanf(“%d”,&a);
	while (a <= 0)
	{
```

```
            printf("Δώσε έναν αριθμό > 0 : ");
            scanf("%d",&a);
        }
        return 0;
}
```

Άσκηση 4.2.2

Να γραφεί πρόγραμμα το οποίο θα εξασφαλίζει την εισαγωγή έγκυρου βαθμού ενός φοιτητή (εύρος τιμών εισόδου από 1 μέχρι 10).

```
#include <stdio.h>
int main()
{
        int grade;
        do {
            printf("Δώσε την βαθμολογία του φοιτητή :");
            scanf("%d",&grade);
        } while (grade < 1 || grade > 10);
        return 0;
}
```

Στο παραπάνω πρόγραμμα χρησιμοποιήσαμε την επαναληπτική δομή **do – while**. Το ίδιο ακριβώς αποτέλεσμα μπορούμε να επιτύχουμε αν χρησιμοποιήσουμε την επαναληπτική δομή **while**.

```
#include <stdio.h>
int main()
{
        int grade;
        printf("Δώσε την βαθμολογία του φοιτητή :");
        scanf("%d",&grade);
        while (grade < 1 || grade > 10)
        {
            printf("Δώσε την βαθμολογία του φοιτητή :");
            scanf("%d",&grade);
        }
        return 0;
}
```

Άσκηση 4.2.3

Να γραφεί πρόγραμμα το οποίο θα ζητάει από το πληκτρολόγιο την τιμή του θετικού ακεραίου αριθμού n και θα υπολογίζει το άθροισμα των τετραγώνων των n πρώτων ακεραίων $S = 1^2 + 2^2 + 3^2 + \ldots + n^2$.

```
#include <stdio.h>
#include <math.h>
int main()
{
        int n, i = 1;
        long s = 0;
        do {
            printf("Δώσε την τιμή του n (αριθμός ≥ 1) : ");
```

```
            scanf("%d",&n);
        } while (n < 1);
        while (i <= n)
        {
            s += pow(i,2);
            i++;
        }
        printf("Το άθροισμα είναι %ld\n",s);
        return 0;
}
```

Άσκηση

Να γραφεί πρόγραμμα το οποίο θα ζητάει την τιμή του άρτιου ακεραίου αριθμού n και θα υπολογίζει το άθροισμα S = 2 + 4 + 6 + 8 + … + n.

```
#include <stdio.h>
int main()
 {
        int i = 2, n;
        long s = 0;
        do {
            printf("Δώσε την τιμή του n (αριθμός ≥ 2) : ");
            scanf("%d",&n);
        } while (n < 2);
        while (i <= n)  {
            s += i;
            i += 2;
        }
        printf("Το άθροισμα είναι %ld\n",s);
        return 0;
}
```

Άσκηση 4.2.5

Να γραφεί πρόγραμμα το οποίο θα υπολογίζει το άθροισμα

$$S = \frac{1}{2} + \frac{2}{3} + \frac{3}{4} + \frac{4}{5} + ... + \frac{99}{100}$$

```
#include <stdio.h>
int main()
{
        int i = 1;
        float s = 0.0;
        while (i <= 99) {
            s += (float) i / (i + 1);
            i++;
        }
        printf("Το άθροισμα είναι %.2f\n",s);
        return 0;
}
```

Άσκηση 4.2.6

Να γραφεί πρόγραμμα το οποίο θα ζητάει την τιμή του θετικού ακεραίου Ν και θα υπολογίζει την τιμή του αθροίσματος

$$S = \frac{1}{4} + \frac{2}{6} + \frac{3}{8} + \frac{4}{10} + ... + \frac{N}{K}$$

όπου Κ η τιμή του ακεραίου που αντιστοιχεί στο Ν.

Αν ο αριθμητής κάθε κλάσματος θεωρήσουμε ότι ακολουθεί τις τιμές της μεταβλητής i = 1, 2, 3, 4, . . . , N τότε ο κάθε παρονομαστής εξαρτάται από τον αριθμητή σύμφωνα με τη σχέση 2 * i + 2.

```
#include <stdio.h>
int main()
 {
        int i = 1, n;
        float s = 0.0;
        do {
                printf("Δώσε την τιμή του N (αριθμός ≥ 1) : ");
                scanf("%d",&n);
        } while (n < 1);
        while (i <= n) {
                s +=  (float) i / ( 2 * i + 2);
                i++;
        }
        printf("Το άθροισμα είναι %.2f\n",s);
        return 0;
}
```

Άσκηση 4.2.7

Δίνεται από το πληκτρολόγιο ακέραιος αριθμός α μεγαλύτερος του 1. Να γραφεί πρόγραμμα που θα υπολογίζει τον μικρότερο ακέραιο αριθμό k, για τον οποίο ισχύει

$$1 + 2 + 3 + 4 + . . . + k > a$$

```
#include <stdio.h>
int main()
{
        int a, i = 0, s = 0;

        do {
                printf("Δώσε ακέραιο μεγαλύτερο του 1 : ");
                scanf("%d",&a);
        } while (a <= 1);
        while (s <= a)
                s += ++i;
        printf("Μικρότερος ακέραιος είναι ο %d\n",i);
        return 0;
}
```

Άσκηση 4.2.8

Να γραφεί πρόγραμμα το οποίο θα εμφανίζει τους παρακάτω αριθμούς

1 2 3 4 5 4 3 2 1

χρησιμοποιώντας την επαναληπτική δομή while.

Όπως παρατηρούμε οι αριθμοί είναι 9. Αρκεί λοιπόν να χρησιμοποιήσουμε μία μεταβλητή i η οποία θα πάρει διαδοχικά τις τιμές 1, 2, 3, 4, 5, 6, 7, 8, 9. Όταν όμως πάρει την τιμή 6 πρέπει να εμφανίσει το 4, όταν πάρει την τιμή 7 το 3, . . . , όταν πάρει την τιμή 9 το 1. Η σχέση λοιπόν στην οποία καταλήγουμε για τιμές του i > 5 είναι : τιμή εμφάνισης = 10 – i.

```
#include <stdio.h>
int main()
{
        int i = 1;
        while (i <= 9) {
                if (i <= 5)
                        printf("%3d",i);
                else
                        printf("%3d",10 – i);
                i++;
        }
        return 0;
}
```

Άσκηση 4.2.9

Να γραφεί πρόγραμμα το οποίο θα διαβάζει από το πληκτρολόγιο μια σειρά θετικών ακεραίων τιμών και θα υπολογίζει το πλήθος, το άθροισμα και το μέσο όρο των τιμών εισόδου. Η διαδικασία εισαγωγής τιμών θα τελειώνει δίνοντας την τιμή 0, ενώ έχει διαβαστεί όμως τουλάχιστον ένας έγκυρος αριθμός.

```
#include <stdio.h>
int main()
{
        int a, pl = 0, sum = 0, mo;
        // a -> τιμή, pl -> πλήθος, sum -> άθροισμα, mo -> μέσος όρος
        do {
        printf("Δώσε ακέραιο αριθμό : ");
                scanf("%d",&a);
        } while (a <= 0);
        while (a != 0) {
                pl++;
                sum += a;
                do {
        printf("Δώσε ακέραιο αριθμό (0 για έξοδος) : ");
                        scanf("%d",&a);
                } while (a < 0);
        }
        mo = sum / pl;
        printf("Το πλήθος των αριθμών είναι : %d\n",pl);
        printf("Το άθροισμα των αριθμών είναι : %d\n",sum);
        printf("Ο μέσος όρος είναι : %d\n",mo);
        return 0;
}
```

Άσκηση 4.2.10

Να γραφεί πρόγραμμα που θα υπολογίζει τον μικρότερο ακέραιο N > 1 για τον οποίο το άθροισμα $1^2 + 2^2 + \ldots + N^2$ είναι τέλειο τετράγωνο ακεραίου αριθμού.

Ένας αριθμός α είναι τέλειο τετράγωνο ακεραίου αριθμού, όταν η τετραγωνική του ρίζα είναι ακέραιος αριθμός.

```
#include <stdio.h>
#include <math.h>
int main()
{
        int i = 2, sum;
        double x;
        // Το άθροισμα θα πρέπει να έχει τουλάχιστον δύο όρους, αφού N > 1.
        sum = 1 + pow(2,2);
        x = sqrt(sum);
        while (floor(x) != x) {
                i++;
                sum += pow(i,2);
                x = sqrt(sum);
        }
        printf("Ο ζητούμενος αριθμός είναι %d\n",i);
        return 0;
}
```

Στην παραπάνω άσκηση χρησιμοποιήθηκαν οι μαθηματικές συναρτήσεις **pow( )** και **floor( )**. Όλες οι μαθηματικές συναρτήσεις προϋποθέτουν την ύπαρξη του αρχείου επικεφαλίδας **math.h** στο πρόγραμμα μας.
Η σύνταξη των εντολών είναι:

double pow(double base,double exp)

Η συνάρτηση **pow( )** επιστρέφει το base υψωμένο στη δύναμη exp ($base^{exp}$).

double floor(double num)

Η συνάρτηση **floor( )** επιστρέφει το μεγαλύτερο ακέραιο, που αναπαρίσταται σαν double και ο οποίος δεν είναι μεγαλύτερος από την num.

Άσκηση 4.2.11

Να γραφεί πρόγραμμα το οποίο θα ζητάει από το πληκτρολόγιο έναν αριθμό και θα τυπώνει την τάξη μεγέθους του.

Η τάξη μεγέθους ενός αριθμού είναι το πλήθος των ακεραίων ψηφίων του μείον 1. Δηλαδή η τάξη μεγέθους του αριθμού 5432,45 είναι τρία (αφού έχουμε χιλιάδες -> 10^3). Άρα θα μπορούσαμε να λύσουμε αυτό το πρόβλημα αν πάρουμε το ακέραιο μέρος του αριθμού και καταγράψουμε το πλήθος των διαδοχικών διαρέσεων του ακεραίου μέρους με το 10 μέχρι να πάρουμε πηλίκο 0.

```
#include <stdio.h>
#include <math.h>
int main()
{
        float a;
        int x, pl = 0;
```

```
        printf("Δώσε τον αριθμό : ");
        scanf("%f",&a);
        x = floor(a);
        while (x != 0) {          // μπορεί να γραφεί και  while (x)
                x = x / 10;
                pl++;
        }
        printf("Το πλήθος των ψηφίων του αριθμού είναι %d\n",pl);
        printf("Η τάξη μεγέθους του είναι %d\n",pl – 1);
        return 0;
}
```

6.3 Η Επαναληπτική Δομή for

Η γενική μορφή της επαναληπτικής δομής **for** είναι:

***for** (αρχική τιμή; έλεγχος; ανανέωση)*

Η επαναληπτική δομή **for** "ενσωματώνει» τρείς ενέργειες που πρέπει να γίνουν προκειμένου να εκτελεστεί ένας συγκεκριμένος αριθμός επαναλήψεων. Ένας αριθμός-μετρητής πρέπει να πάρει μια αρχική τιμή, πρέπει να συγκριθεί με κάποιον πεπερασμένο αριθμό και τρίτον, πρέπει να αυξάνεται κάθε φορά που η επαναληπτική δομή εκτελείται.

Λειτουργία της εντολής
Η πρόταση αρχικών τιμών εκτελείται μόνο μία φορά πρίν εκτελεστεί κάποια από τις προτάσεις του βρόχου. Αν η συνθήκη ελέγχου είναι αληθής (ή μη μηδενική), ο βρόχος εκτελείται μία φορά. Μετά υπολογίζεται η έκφραση ανανέωσης, και η συνθήκη ελέγχου ελέγχεται ακόμα μια φορά. Στην **for** η απόφαση για μια ακόμη εκτέλεση του βρόχου, παίρνεται πρίν την εκτέλεση της πρότασης. Έτσι, ο βρόχος είναι πιθανόν να μην εκτελεστεί ποτέ. Το μέρος των προτάσεων μπορεί να αποτελείται από μια απλή πρόταση ή από μια σύνθετη πρόταση.

Παράδειγμα 4.3.1

```
#include <stdio.h>
#define COUNTER 10
int main()
{
        int count;
        for (count = 1;count <= COUNTER;count++)
                printf("Αριθμός επανάληψης %d\n",count);
        return 0;
}
```

Οι παρενθέσεις που ακολουθούν την **for** περιέχουν τρείς εκφράσεις χωρισμένες με ελληνικά ερωτηματικά. Η πρώτη έκφραση δίνει αρχικές τιμές και εκτελείται μια φορά όταν ο βρόχος **for** ξεκινάει. Η δεύτερη έκφραση είναι μια συνθήκη ελέγχου, και υπολογίζεται πριν από κάθε εκτέλεση του βρόχου. Όταν η συνθήκη γίνει ψευδής (όταν η count γίνει μεγαλύτερη από την COUNTER), ο βρόχος τερματίζεται. Η τρίτη έκφραση υπολογίζεται στο τέλος κάθε βρόχου.

Παράδειγμα 4.3.2

```
#include <stdio.h>
int main()
{
        int number;
        printf("%5s %15s\n","n","n τετράγωνο");
        for (number = 1;number <= 10;number++)
                printf("%5d %15d\n",number,number * number);
        return 0;
}
```

Το παραπάνω παράδειγμα χρησιμοποιεί ένα βρόχο **for** σε ένα πρόγραμμα που εμφανίζει τους αριθμούς από 1 έως 10 και τα τετράγωνά τους. Ο βρόχος **for** περίεχει την αρχική τιμή της **num**, την τελική τιμή της **num** καθώς και το ποσό αύξησης της **num** σε κάθε βρόχο.

Παράδειγμα

```
#include <stdio.h>
int main()
{
        int i;
        for (i = 1;i <= 1000;i++)
                ;
        printf("Εδώ τελειώνει η εκτέλεση του βρόχου με i = %d\n",i);
        return 0;
}
```

Το παραπάνω παράδειγμα θα εκτελέσει 10000 φορές την κενή εντολή, χωρίς στην πραγματικότητα να κάνει τίποτα άλλο από το να καθυστερήσει τον υπολογιστή στην εκτέλεση της εντολής **printf( )** που θα εμφανίσει τελικά για το **i** την τιμή 10001 (είναι η τιμή που έχει το **i** μετά το πέρας του βρόχου).

▫ Επίσης η επαναληπτική δομή **for** μπορεί να χρησιμοποιηθεί για να μετράμε πρός τα κάτω με χρήση του τελεστή μείωσης, όπως φαίνεται στο ακόλουθο παράδειγμα.

Παράδειγμα

```
#include <stdio.h>
int main()
{
        int i;
        for (i = 10; i  > 0; i--)
                printf("%d\n",i);
        return 0;
}
```

Ο τελεστής αύξησης (ή μείωσης αντίστοιχα) μπορεί να αυξάνεται (ή να μειώνεται) κατά οποιονδήποτε ακέραιο αριθμό, όπως φαίνεται στο ακόλουθο παράδειγμα που υπολογίζει το άθροισμα όλων των άρτιων αριθμών από 2 έως 100.

Παράδειγμα

```
#include <stdio.h>
int main()
{
        int i, sum = 0;
        for (i = 2; i <= 100; i += 2)
                sum += i;
        printf("Το άθροισμα είναι %d\n",sum);
        return 0;
}
```

Η επαναληπτική δομή **for** μπορεί να χρησιμοποιεί χαρακτήρες αντί για αριθμούς.

Παράδειγμα

```
#include <stdio.h>
int main()
{
        char ch;
        for (ch = 'A'; ch <= 'Z'; ch++)
                printf("Η ASCII τιμή του χαρακτήρα %c είναι %d\n",ch,ch);
        return 0;
}
```

Το παραπάνω παράδειγμα εμφανίζει την ASCII τιμή όλων των χαρακτήρων από Α έως Z.

Στην επαναληπτική δομή **for** μία ποσότητα μπορεί να αυξάνει γεωμετρικά παρά αριθμητικά. Αυτό σημαίνει ότι αντί να προσθέτουμε ένα σταθερό ποσό κάθε φορά μπορούμε να πολλαπλασιάζουμε με ένα σταθερό ποσό.

Παράδειγμα

```
#include <stdio.h>
int main()
{
        int i = 0;
        float d;
        for(d = 1000.0;d < 3100.0; d *= 1.05)
        printf("%4d %10.2f\n",++i,d);
        return 0 ;
}
```

Το παραπάνω παράδειγμα υπολογίζει σε πόσα χρόνια (μέσω της μεταβλητής **i**) ένα αρχικό ποσό 1000 €, όταν αυξάνεται κάθε χρόνο κατά 5% (d *= 1.05), θα ξεπεράσει τα 3000 €.

Μία η περισσότερες εκφράσεις μπορούν να παραμείνουν κενές (χωρίς να παραλείψουμε όμως τα ελληνικά ερωτηματικά). Σε αυτήν την περίπτωση πρέπει να βεβαιωθούμε ότι περιλαμβάνονται στο βρόχο μερικές προτάσεις που θα τον κάνουν να τερματίσει.

Παράδειγμα

```
#include <stdio.h>
int main()
{
        int i;
        for(i = 0;i < 10; )
        printf("%4d %4d\n",i,2 * (++i + 1));
        return 0 ;
}
```

Το παραπάνω παράδειγμα υπολογίζει την αλγεβρική παράσταση 2 • (i+1) για τιμές του i από 1 μέχρι 10.

Παράδειγμα

```
#include <stdio.h>
int main()
{
        for( ; ; )
                printf("Βλέπεις συνέχεια αυτό το μήνυμα ;\n");
        return 0;
}
```

Ο παραπάνω βρόχος επαναλαμβάνεται για πάντα αφού ένας κενός έλεγχος θεωρείται αληθής.

Η πρώτη έκφραση δεν χρειάζεται να δώσει αρχική τιμή σε μια μεταβλητή. Μπορεί να είναι απλά μια πρόταση **printf( )**. Πρέπει να θυμόμαστε όμως ότι η πρώτη έκφραση υπολογίζεται ή εκτελείται μόνο μία φορά, πριν εκτελεστεί οποιοδήποτε μέρος του βρόχου.

Παράδειγμα

```
#include <stdio.h>
int main()
{
        int num = 1;
        for(printf("Δώσε συνεχώς αριθμούς. 0 για τερματισμό.\n"); num != 0; )
                scanf("%d",&num);
        printf("Πληκτρολόγησες το 0.\n");
        return 0;
}
```

Στο παραπάνω παράδειγμα, το πρόγραμμα ζητά από τον χρήστη να πληκτρολογεί συνεχώς αριθμούς, μέχρι να δοθεί ο αριθμός 0, οπότε το πρόγραμμα τερματίζει.

Άσκηση 4.3.1

Διαβάζονται δέκα τριάδες ακεραίων αριθμών. Να γραφεί πρόγραμμα που να τυπώνει σε πόσες τριάδες υπάρχει τουλάχιστον ένας αριθμός που να ισούται με το άθροισμα των άλλων δύο.

```
#include <stdio.h>
int main()
```

```
{
        int i,a,b,c,count = 0;
        for (i = 1; i <= 10; i++) {
                printf("Δώσε τρείς αριθμούς : ");
                scanf("%d %d %d",&a,&b,&c);
                 if (a + b == c || a + c == b || c + b == a)
                        count++;
        }
        printf("Το πλήθος των τριάδων είναι %d.\n",count);
        return 0;
}
```

Άσκηση 4.3.2

Να γραφεί πρόγραμμα που να υπολογίζει το Ν παραγοντικό. Δίνεται ότι

$$N! = \begin{cases} 1, \alpha\nu & \text{N} = 0 \\ 1*2*3*...*(N-1)*N, \alpha\nu & \text{N} > 0 \end{cases}$$

```
#include <stdio.h>
int main()
{
        long N = 1;
        int n,i;
        do {
                printf("Δώσε έναν αριθμό : ");
                scanf("%d",&n);
        while (n < 0);
        for (i = 2; i <= n; i++)
                N *= i;
        printf("Το %d παραγοντικό είναι %ld.\n",n,N);
        return 0;
}
```

Άσκηση 4.3.3

Ένας αριθμός ονομάζεται πρώτος αν οι μόνοι διαιρέτες του (δηλαδή οι μόνοι αριθμοί που τον διαιρούν ακριβώς) είναι ο ίδιος ο αριθμός και η μονάδα. Να γραφεί πρόγραμμα το οποίο να διαβάζει έναν αριθμό και να εξετάζει αν είναι πρώτος ή όχι.

```
#include <stdio.h>
int main()
{
        int i,n;
        do {
                printf("Δώσε έναν ακέραιο αριθμό : ");
                scanf("%d",&n);
        } while (n <= 0);
        for (i = 2; i < n; i++)
                if (n % i == 0) break;
        printf("o αριθμός %s\n",(i < n) ? "είναι πρώτος" : "δεν είναι πρώτος");
        return 0;
}
```

Άσκηση 4.3.4

Ένα αρχείο που περιέχει χρήσιμες πληροφορίες είναι «κλειδωμένο» με κωδικό. Ο κωδικός είναι τριψήφιος αριθμός από 111 μέχρι 999 και ισχύει ότι x < y < z, καθώς και ότι ο x είναι άρτιος και ο z περιττός. Να γραφεί πρόγραμμα το οποίο θα εμφανίζει όλους τους πιθανούς κωδικούς.

```
#include <stdio.h>
int main()
{
        int x,y,z,code;
        for (x = 1; x <= 9; x++)
                if (x % 2 == 0)
                        for (y = x + 1; y <= 9; y++)
                                for (z = y + 1; z <= 9; z++)
                                        if (z % 2 != 0) {
                                                code = 100 * x + 10 * y + z;
                                                printf("Κωδικός = %d\n",code);
                                        }
        return 0;
}
```

Άσκηση 4.3.5

Να γραφεί πρόγραμμα που θα εμφανίζει όλους τους τέλειους αριθμούς από το 2 έως το 1000. Τέλειος είναι ένας αριθμός που το άθροισμα των διαιρετών του είναι ίσο με το διπλάσιο του αριθμού.

Για παράδειγμα, το 6 είναι τέλειος αριθμός, αφού οι διαιρέτες του είναι οι αριθμοί 1, 2, 3, 6 και έχουν ως άθροισμα : 1 + 2 + 3 + 6 = 12 = 2 * 6.

```
#include <stdio.h>
int main()
{
        int i,j,sum;
        for (i = 2; i <= 1000; i++) {
                sum = 1 + i;
                for (j = 2; j < i; j++)
                        if (i % j == 0)
                                sum += j;
                if (sum == 2 * i)
                        printf("Τέλειος αριθμός %d\n",i);
        }
        return 0;
}
```

Άσκηση 4.3.6

Να γραφεί πρόγραμμα το οποίο θα διαβάζει έναν θετικό ακέραιο αριθμό N και θα υπολογίζει και θα εμφανίζει το άθροισμα

$$S = \frac{1}{2} + \frac{1}{4} + \frac{1}{6} + ... + \frac{1}{N}$$

αν ο αριθμός N είναι άρτιος, ή το άθροισμα

$$S = \frac{1}{1} + \frac{1}{3} + \frac{1}{5} + ... + \frac{1}{N}$$

αν ο αριθμός N είναι περιττός.

```
#include <stdio.h>
int main()
{
        float S = 0.0;
        int i, N, start;
        do {
                printf("Δώσε έναν θετικό ακέραιο αριθμό : ");
                scanf("%d",&N);
        } while (N <= 0);
        if (N % 2 == 0)
                start = 2;
        else
                start = 1;
        for (i = start; i <= N; i += 2)
                S += (float) 1 / i;
        printf("Το άθροισμα είναι %.2f\n",S);
        return 0;
}
```

Άσκηση 4.3.7

Να γραφεί πρόγραμμα το οποίο θα βρίσκει και θα εμφανίζει τους τριψήφιους θετικούς ακέραιους αριθμούς, οι οποίοι έχουν την ιδιότητα να ισούνται με το άθροισμα των κύβων των ψηφίων τους.

(για παράδειγμα $371 = 3^3 + 7^3 + 1^3$)

```
#include <stdio.h>
#include <math.h>
int main()
{
        int i, m, d, e;
        for (i = 100; i <= 999; i++) {
                e = i / 100;
                d = (i – e * 100) / 10;
                m = i – e * 100 – d * 10;
                if (i == pow(e,3) + pow(d,3) + pow(m,3))
                        printf("%5d",i);
        }
        printf("\n");
        return 0;
}
```

6.4 Ο Τελεστής κόμμα

Ο τελεστής κόμμα χρησιμοποιείται για την παράθεση πολλών παραστάσεων στη σειρά. Για παράδειγμα, στην

```
x = (y = 3, y + 1);
```

αποδίδεται πρώτα η τιμή 3 στην **y** και κατόπιν η τιμή 4 στην **x**. Οι παρενθέσεις είναι αναγκαίες, γιατί ο τελεστής κόμμα έχει χαμηλότερη προτεραιότητα από τον τελεστή

εκχώρησης τιμής. Ουσιαστικά, αυτό που κάνει το κόμμα είναι να προκαλεί την εκτέλεση μιας σειράς από πράξεις. Όταν χρησιμοποιείται στο δεξιό μέλος μιας πρότασης εκχώρησης τιμής, η τιμή που αποδίδεται στο αριστερό μέλος είναι η τιμή της τελευταίας παράστασης της χωριζόμενης με κόμματα λίστας. Για παράδειγμα, αφού εκτελεστούν οι εντολές

```
y = 10;
x = (y = y – 5, 25 / y);
```

το **x** θα έχει την τιμή 5 αφού η αρχική τιμή του **y**, η 10, μειώθηκε κατά 5, και στη συνέχεια η διαίρεση του 25 με το **y** δίνει αποτέλεσμα 5.

Ο τελεστής κόμμα επεκτείνει την ελαστικότητα του βρόχου **for**, επιτρέποντάς μας να περιλαμβάνουμε περισσότερους προσδιορισμούς αρχικής τιμής ή ανανεώσεις στον ορισμό του βρόχου **for**.

Παράδειγμα

```
#include <stdio.h>
int main()
{
        int up, down, target;
        printf("Μέχρι που θέλεις να μετρήσω ; ");
        scanf("%d",&target);
        for (up = 1, down = target; (up <= target) && (down >= 1); up++, down--)
                printf("%5d  %5d\n",up, down);
        return 0;
}
```

Στο παραπάνω παράδειγμα ό βρόχος **for** χρησιμοποιεί τον τελεστή κόμμα για να «μετρήσει» δύο φορές, μία από ***1*** μέχρι ***target*** και συγχρόνως αντίστροφα από ***target*** μέχρι ***1***.

6.5 Η εντολή break

Στο προηγούμενο κεφάλαιο, είδαμε την χρήση της **break** με την εντολή **switch**. Η **break** μπορεί να χρησιμοποιηθεί επίσης και με τις τρείς επαναληπτικές δομές **for**, **while** και **do – while**, και έχει ως αποτέλεσμα την διακοπή εκτέλεσης της επαναληπτικής δομής, και τη μετάβαση του ελέγχου του προγράμματος στην εντολή που βρίσκεται αμέσως μετά το βρόχο. Αν η **break** βρίσκεται μέσα σε εμφωλευμένες δομές, επηρεάζει μόνο την πιο εσωτερική δομή.

Παράδειγμα

```
#include <stdio.h>
int main()
{
        int i;
        for (i = 0; i < 100; i++) {
                printf("%5d",i);
                if (i == 10) break;
```

```
        }
        return 0;
}
```

Το παραπάνω πρόγραμμα θα εμφανίσει στην οθόνη τους αριθμούς 1 έως 10 και στην συνέχεια θα τερματιστεί γιατί η **break** προκαλεί άμεση έξοδο από το βρόχο.

Παράδειγμα

```
#include <stdio.h>
#define MAX_COUNT 10
int main()
{
        int val, count = 0, sum = 0;
        printf("Δώσε έναν αριθμό : ");
        scanf("%d",&val);
        while (val >= 0) {
                count++;
                if (count > MAX_COUNT) {
                        printf("Δεν επιτρέπονται άλλοι αριθμοί.\n");
                        count--;
                        break;
                }
                sum += val;
                printf("Δώσε έναν αριθμό : ");
                scanf("%d",&val);
        }
        if (count > 0)
        printf("Ο μέσος όρος των %d αριθμών είναι %.2f\n",count, sum/count);
        else
                printf("Δέν έχεις δώσει αριθμούς.\n");
        return 0;
}
```

Στο παραπάνω παράδειγμα, αν προσπαθήσουμε να εισάγουμε περισσότερους από 10 αριθμούς, η **break** θα διακόψει τον βρόχο **while** και ο έλεγχος του προγράμματος θα μεταβιβαστεί στην εντολή που βρίσκεται αμέσως μετά το βρόχο **while** .

6.6 Η εντολή continue

Αυτή η εντολή μπορεί να χρησιμοποιηθεί και με τις τρείς επαναληπτικές δομές, αλλά όχι με την εντολή **switch**. Η **continue** όπως και η **break** διακόπτει τη ροή ενός προγράμματος. Όμως αντί να τερματίσει ολόκληρο το βρόχο, η **continue** έχει ως αποτέλεσμα τη μη εκτέλεση του υπόλοιπου τμήματος του βρόχου και την επανάληψή του από την αρχή.

Παράδειγμα

```
#include <stdio.h>
int main()
{
```

```
        int i;
        for (i = 0; i < 100; i++)
                if (i % 2) continue;
        printf("%5d",i);
        return 0;
}
```

Το παραπάνω πρόγραμμα θα εμφανίσει μόνο τους άρτιους αριθμούς. Κάθε φορά που παράγεται από το πρόγραμμα ένας περιττός αριθμός, εκτελείται η εντολή **if**, γιατί το υπόλοιπο της διαίρεσης ενός περιττού αριθμού με το 2 είναι ίσο πάντοτε με 1, δηλαδή ΑΛΗΘΗΣ τιμή. Έτσι, ένας περιττός αριθμός προκαλεί την εκτέλεση της εντολής **continue**, οπότε εκτελείται η επόμενη επανάληψη του βρόχου, ενώ η εντολή **printf( )** παρακάμπτεται.

Παράδειγμα

```
#include <stdio.h>
#define TARGET 1000
int main()
{
        int i;
        for (i = 1; i <= TARGET; count++)
                if (i % 2) continue;
                else if (i % 4) continue;
                else if (i % 6) continue;
                else if (i % 8) continue;
                else if (i % 10) continue;
                else if (i % 12) continue;
                else
        printf("%d\n",i);
        }
        return 0;
}
```

Το παραπάνω παράδειγμα θα εμφανίσει όλους τους αριθμούς που είναι μεγαλύτεροι ή ίσοι του 1 και μικρότεροι ή ίσοι του 1000 και που διαιρούνται ακριβώς με το 2, 4, 6, 8, 10, 12.

6.7 Ο Τύπος Δεδομένων char (Χαρακτήρες)

Ένας σημαντικός τύπος δεδομένων στην C είναι ο **char** από την λέξη character (χαρακτήρας). Ένας χαρακτήρας είναι μια τιμή του ενός byte την οποία μπορούμε να χρησιμοποιήσουμε για να κρατήσουμε εκτυπώσιμους χαρακτήρες ή ακεραίους από -128 έως 127. Σε ένα πρόγραμμα, μια σταθερά ενός χαρακτήρα περιβάλλεται από μονά εισαγωγικά.

Παράδειγμα

```
#include <stdio.h>
int main()
{
```

```
        char ch;
        do {
                printf("Δώσε έναν χαρακτήρα (q ή Q για έξοδος) : ");
                scanf("%c",&ch);
                printf("Ο χαρακτήρας που έδωσες είναι ο %c\n",ch);
        } while (ch != 'q' && ch != 'Q');
        return 0;
}
```

Παρ' όλο που μπορούμε να χρησιμοποιούμε την **scanf( )** για να διαβάσουμε ένα χαρακτήρα από το πληκτρολόγιο, μια πιο συνηθισμένη μέθοδος είναι να χρησιμοποιούμε τη συνάρτηση βιβλιοθήκης **getchar( )** (εναλλακτικά μπορεί να χρησιμοποιηθεί η **getc(stdin)**). Επίσης για την εμφάνιση του χαρακτήρα στην οθόνη μπορούμε να χρησιμοποιούμε την **putchar( )** (εναλλακτικά μπορεί να χρησιμοποιηθεί η **putc(ch, stdout)**). Τα πρώτυπα των **getchar( )** και **putchar( )** βρίσκονται στο **stdio.h** και «δηλώνονται» ως :

```
int getchar(void);
int putchar(int ch);
```

(Τυπικά, είναι μάκρο - συναρτήσεις του προεπεξεργαστή). Δεν πρέπει να μας ενοχλεί που η συνάρτηση **getchar( )** επιστρέφει έναν ακέραιο. Το byte χαμηλής τάξης περιέχει τον χαρακτήρα. Επίσης, αν και η συνάρτηση **putchar( )** έχει δηλωθεί ότι δέχεται ακέραια παράμετρο, στην οθόνη εμφανίζεται μόνο το byte χαμηλής τάξης.

Παράδειγμα

```
#include <stdio.h>
int main()
{
        char ch;
        printf("Δώσε έναν χαρακτήρα  : ");
        ch = getchar();
        putchar(ch);
        return 0;
}
```

Το πρόβλημα με την **getchar( )** είναι ότι κρατάει την είσοδο στην περιοχή προσωρινής αποθήκευσης μέχρι να πατήσουμε το <Enter>. Το αποτέλεσμα αυτού του γεγονότος είναι να «περιμένουν» ένας ή περισσότεροι χαρακτήρες στην ουρά εισόδου μετά την επιστροφή της **getchar( )**.

Παράδειγμα

```
#include <stdio.h>
int main()
{
        char ch1,ch2;
        printf("Δώσε έναν χαρακτήρα : ");
        ch1 = getchar();
        ch2 = getchar();
        printf("Οι χαρακτήρες είναι οι %c και %c.\n",ch1,ch2);
```

```
        return 0;
}
```

Στο παραπάνω πρόγραμμα, αν μετά την εμφάνιση του μηνύματος *«Δώσε έναν χαρακτήρα :»* στην οθόνη πληκτρολογήσουμε *abcdefg* και πατήσουμε το <Enter>, τότε η μεταβλητή **ch1** θα «περιέχει» τον χαρακτήρα 'a' και η μεταβλητή **ch2** τον χαρακτήρα 'b'. Αυτό προέκυψε γιατί όπως αναφέρθηκε και προηγουμένως η είσοδος κρατήθηκε στην περιοχή προσωρινής αποθήκευσης μέχρι να πατηθεί το <Enter>. Η πρώτη «κλήση» της **getchar( )** διάβασε τον πρώτο χαρακτήρα της περιοχής προσωρινής αποθήκευσης και η δεύτερη «κλήση» διάβασε τον επόμενο χαρακτήρα.

Παράδειγμα

```
#include <stdio.h>
#define SPACE ' '
int main()
{
        char ch;
        printf("Δώσε αγγλικούς χαρακτήρες με κεφαλαία γράμματα\n");
        while ((ch = getchar()) != '\n') {
                if (ch == SPACE)
                        putchar(ch);
                else
                        putchar(ch + 32);
        }
        return 0;
}
```

Το παραπάνω πρόγραμμα διαβάζει κεφαλαίους αγγλικούς χαρακτήρες και όταν πατήσουμε το <Enter> τους εμφανίζει με πεζούς αγγλικούς (Το Α στον κώδικα ASCII έχει τιμή 65 ενώ το a έχει τιμή 97. Άρα η διαφορά τους είναι 32).

Η γραμμή ***while ((ch = getchar()) != '\n')*** συνδιάζει δύο ενέργειες σε μία πρόταση. Αυτές είναι : εκχώρηση μιάς τιμής στη **ch** και σύγκριση αυτής της τιμής με το χαρακτήρα νέας γραμμής. Οι παρενθέσεις που περικλείουν το ***ch = getchar()*** καθορίζουν το αριστερό μέλος του τελεστή **!=** . Για να αξιολογήσει αυτήν την παράσταση ο υπολογιστής πρέπει να κάνει «κλήση» της συνάρτησης **getchar( )** και να εκχωρήσει την τιμή της στην **ch**. Μετά την ανάγνωση της **ch** η συνθήκη ελέγχου έχει πλέον διαμορφωθεί σε **ch != '\n'**.

4.8 Αλφαριθμητικά

Στην C ένα αλφαριθμητικό (string) είναι ένας πίνακας χαρακτήρων που τερματίζεται με ένα μηδενικό (null). Η C δέν έχει τύπο «αλφαριθμητικού». Γι' αυτό πρέπει να δηλώνουμε έναν πίνακα χαρακτήρων και να χρησιμοποιούμε τις διάφορες συναρτήσεις αλφαριθμητικών της βιβλιοθήκης για να τον χειριστούμε. Μπορούμε να «δηλώσουμε» ένα αλφαριθμητικό με τον ακόλουθο τρόπο:

```
char str[80];
```

η παραπάνω πρόταση δηλώνει ένα αλφαριθμητικό 80 χαρακτήρων, που έχει το όνομα **str**. Για να αναφερθούμε σε ένα συγκεκριμένο στοιχείο, τοποθετούμε το δείκτη (index) του στοιχείου μέσα σε αγκύλες μετά το όνομα του πίνακα. Οι δείκτες όλων των πινάκων της C αρχίζουν από το 0. Έτσι, το str[0] είναι το πρώτο στοιχείο, το str[1] το δεύτερο στοιχείο, μέχρι το str[79] που είναι το τελευταίο στοιχείο. Όπως αναφέρθηκε, όλα τα αλφαριθμητικά τελειώνουν με ένα μηδενικό. Το μηδενικό στη C είναι ένα 0 και καθορίζεται σε ένα πρόγραμμα σαν η σταθερά τύπου χαρακτήρα '\0'. Επομένως για να είναι ένας πίνακας χαρακτήρων αρκετά μεγάλος ώστε να περιέχει τη λέξη «Γιώργος» θα πρέπει να έχει μήκος τουλάχιστον οκτώ χαρακτήρες, επτά χαρακτήρες για το αλφαριθμητικό και ένα χαρακτήρα για το μηδενικό τέλους, όπως φαίνεται παρακάτω.

char str[] = "Γιώργος";

'Γ'	'ι'	'ώ'	'ρ'	'γ'	'ο'	'ς'	'\0'

Για το διάβασμα ενός αλφαριθμητικού από το πληκτρολόγιο, μπορούμε να χρησιμοποιήσουμε τη συνάρτηση βιβλιοθήκης **gets( )**. Η συνάρτηση **gets( )** παίρνει σαν όρισμα το όνομα του αλφαριθμητικού και διαβάζει χαρακτήρες από το πληκτρολόγιο μέχρι να πατηθεί το <Enter>. Το <Enter> δεν αποθηκεύεται αλλά αντικαθίσταται από το μηδενικό τερματισμού.

Παράδειγμα

```
#include <stdio.h>
int main()
{
        char name[20];
        printf("Δώσε το όνομά σου : ");
        scanf("%s",name);
        printf("Το ονομά σου είναι %s\n",name);
        return 0;
}
```

Στο παραπάνω παράδειγμα χρησιμοποιούμε την **scanf( )** για να διαβάσουμε ένα αλφαριθμητικό και την **printf( )** για να το εμφανίσουμε στην οθόνη.

Παράδειγμα

```
#include <stdio.h>
int main()
{
        char name[20];
        printf("Δώσε το όνομά σου : ");
        gets(name);
        printf("Το ονομά σου είναι ");
        puts(name);
        return 0;
}
```

Στο παραπάνω παράδειγμα χρησιμοποιούμε την **gets( )** για να διαβάσουμε ένα αλφαριθμητικό και την **puts( )** για να το εμφανίσουμε στην οθόνη. Πρέπει να έχουμε υπόψιν μας ότι η gets( ) δεν εκτελεί έλεγχο ορίων στον πίνακα με τον οποίο

«καλείται». Επομένως, αν δώσουμε ένα αλφαριθμητικό μεγαλύτερο από το μέγεθος του πίνακα, ο πίνακας θα ξεπεραστεί.

6.9 Συναρτήσεις Βιβλιοθήκης για Αλφαριθμητικά

Η C υποστηρίζει μια μεγάλη ποικιλία από συναρτήσεις χειρισμού αλφαριθμητικών. Όλες οι συναρτήσεις χειρισμού αλφαριθμητικών προϋποθέτουν την ύπαρξη του αρχείου επικεφαλίδας **string.h** στο πρόγραμμα μας.

Η συνάρτηση strlen( )

Η «δήλωση» της συνάρτησης είναι:

```
int strlen(char *str)
```

Η συνάρτηση **strlen( )** επιστρέφει το μήκος, του τερματιζόμενου με μηδενικό χαρακτήρα αλφαριθμητικού, στο οποίο δείχνει ο **str**. Η συνάρτηση δεν μετράει το μηδενικό χαρακτήρα.

Παράδειγμα

```
#include <stdio.h>
#include <string.h>
int main()
{
        char str[80];
        printf("Δώσε ένα αλφαριθμητικό : ");
        gets(str);
        printf("Το μήκος του αλφαριθμητικού που έδωσες είναι %d\n",strlen(str));
        return 0;
}
```

Παράδειγμα

```
#include <stdio.h>
#include <string.h>
int main()
{
        int i;
        char str[80];
        printf("Δώσε ένα αλφαριθμητικό : ");
        gets(str);
        for (i = strlen(str);i >= 0; i--)
                printf("%c",str[i]);
        printf("\n");
        return 0;
}
```

Η συνάρτηση strcpy( )

Η «δήλωση» της συνάρτησης είναι:

char *strcpy(char *str1,char *str2)

Η συνάρτηση **strcpy( )** χρησιμοποιείται για την αντιγραφή των περιεχομένων του αλφαριθμητκού **str2** στο αλφαριθμητικό **str1**. Ο πίνακας **str1** πρέπει να είναι μεγάλος ώστε να χωρέσει το αλφαριθμητικό **str2**. Το αλφαριθμητικό **str2** πρέπει να είναι αλφαριθμητικό τερματιζόμενο με μηδενικό χαρακτήρα.

Παράδειγμα

```
#include <stdio.h>
#include <string.h>
int main()
{
        char str1[80];
        strcpy(str1,"Αυτό είναι το νέο αλφαριθμητικό.");
        printf("%s\n",str1);
        return 0;
}
```

Η συνάρτηση strcat( )

Η «δήλωση» της συνάρτησης είναι:

char *strcat(char *str1,char *str2)

Η συνάρτηση **strcat( )** προσθέτει ένα αντίγραφο του **str2** στο τέλος του **str1**, χωρίς να επηρεάζει το αλφαριθμητικό **str2**. Και τα δύο αλφαριθμητικά πρέπει να τελειώνουν σε μηδενικό χαρακτήρα και το αποτέλεσμα θα τελειώνει σε μηδενικό χαρακτήρα.

Παράδειγμα

```
#include <stdio.h>
#include <string.h>
int main()
{
        char str2[20],str1[80] = "Το ονομά σου είναι, ";
        printf("Ποιο είναι το ονομά σου ; ");
        gets(str2);
        strcat(s1,str2);
        printf("%\n",str1);
        return 0;
}
```

Η συνάρτηση strcmp( )

Η «δήλωση» της συνάρτησης είναι:

int strcmp(char *str1,char *str2)

Η συνάρτηση **strcmp( )** συγκρίνει λεξικογραφικά δύο τερματιζόμενα με NULL αλφαριθμητικά, και επιστρέφει:
α. τιμή = 0, αν τα δύο αλφαριθμητικά είναι ίσα,
β. τιμή > 0, αν το str1 είναι μεγαλύτερο από το str2,
γ. τιμή < 0, αν το str1 είναι μικρότερο από το str2.

Παράδειγμα

```
#include <stdio.h>
#include <string.h>
int main()
{
        char password[] = "1212";
        char code[15];
        do {
                printf("Δώσε τον κωδικό : ");
                gets(code);
        } while (strcmp(code,password));
        return 0;
}
```

Παράδειγμα

```
#include <stdio.h>
#include <string.h>
int main()
{
        char str[80];
        for ( ; ; ) {
                printf("Πληκτρολόγησε ένα αλφαριθμητικό : ");
                gets(str);
                if (! strcmp(str,"έξοδος")) break;
        }
        return 0;
}
```

Η συνάρτηση **strcmp( )** επιστρέφει τιμή ΨΕΥΔΗΣ(τιμή = 0) όταν τα αλφαριθμητικά είναι ίδια. Επομένως πρέπει να χρησιμοποιούμε τον τελεστή NOT (!), αν θέλουμε να συμβαίνει κάτι όταν τα αλφαριθμητικά είναι ίσα. Το παραπάνω πρόγραμμα λοιπόν, ζητάει συνεχώς από τον χρήστη να πληκτρολογήσει ένα αλφαριθμητικό, μέχρι να πληκτρολογήσει το αλφαριθμητικό «έξοδος», οπότε και το πρόγραμμα τελειώνει.

Η συνάρτηση strchr( )

Η «δήλωση» της συνάρτησης είναι:

```
char *strchr(char *str,int ch)
```

Η συνάρτηση **strchr( )** επιστρέφει ένα δείκτη στην πρώτη εμφάνιση του byte χαμηλής τάξης του ch στο αλφαριθμητικό στο οποίο δείχνει ο str. Αν η συνάρτηση δεν βρεί το χαρακτήρα σε καμία θέση, επιστρέφει ένα μηδενικό δείκτη.

Παράδειγμα

```
#include <stdio.h>
#include <string.h>
int main()
{
        char *p;
        p = strchr("Η C είναι μια συμπαγής γλώσσα προγραμματισμού",(int) ' ');
        printf(p);
        return 0;
}
```

Το αποτέλεσμα της εκτέλεσης του παραπάνω προγράμματος είναι η εμφάνιση στην οθόνη του αλφαριθμητικού « είναι μια συμπαγής γλώσσα προγραμματισμού».

6.10 Ασκήσεις

1. Να γραφεί πρόγραμμα το οποίο να εμφανίζει στην οθόνη τους 20 πρώτους ακέραιους θετικούς αριθμούς και τις τετραγωνικές τους ρίζες.
2. Να γραφεί πρόγραμμα το οποίο να υπολογίζει τη μέγιστη και την ελάχιστη ημερήσια θερμοκρασία που εμφανίστηκε κατά τη διάρκεια του έτους 2003 (365 ημέρες) στη Λάρισα.
3. Να γραφεί πρόγραμμα το οποίο :
α. θα διαβάζει από το πληκτρολόγιο μια σειρά θετικών ακέραιων τιμών. Η διαδικασία εισαγωγής τιμών θα τερματίζεται όταν το άθροισμα τους γίνει μεγαλύτερο από 1000 ή όταν δοθεί τιμή αρνητική ή μηδέν.
β. θα εμφανίζει το άθροισμα των τιμών εισόδου.
4. Να γραφεί πρόγραμμα το οποίο :
α. θα διαβάζει από το πληκτρολόγιο μια σειρά θετικών ακέραιων τιμών. Η διαδικασία εισαγωγής τιμών θα τελειώνει δίνοντας την τιμή 0,
β. θα υπολογίζει και θα εμφανίζει το πλήθος, το άθροισμα και το μέσο όρο των τιμών εισόδου.
5. Να γραφεί πρόγραμμα, το οποίο :
α. θα διαβάζει από το πληκτρολόγιο μια σειρά θετικών αριθμών και
β. θα υπολογίζει και θα εμφανίζει το πλήθος τους σε καθεμιά από τις παρακάτω κατηγορίες τιμών 1 – 20, 21 – 50, 51 – 80.
Η διαδικασία εισαγωγής τιμών θα τελειώνει εισάγοντας την τιμή 0.
6. Μια εταιρεία δημοσκοπήσεων έκανε μια έρευνα στην Αθήνα με στόχο να καταγράψει τη χρήση του μετρό από τους πολίτες κατά τις μετακινήσεις τους. Το ερώτημα που υποβλήθηκε σε 2000 άτομα συνολικά ήταν : 'Πόσο συχνά χρησιμοποιείται το μετρό στις μετακινήσεις σας ; '. Καθένας/μία είχε τη δυνατότητα να δώσει μια από τις παρακάτω απαντήσεις : 'Καθημερινά', 'Μερικές φορές', 'Καθόλου'. Να γραφεί πρόγραμμα το οποίο :
α. θα διαβάζει την απάντηση κάθε ερωτώμενου (1 = Καθημερινά, 2 = Μερικές φορές, 3 = Καθόλου),
β. θα εμφανίζει το ποσοστό % για κάθε απάντηση,
γ. θα εμφανίζει την απάντηση της πλειονότητας των ερωτηθέντων και το αντίστοιχο ποσοστό.
7. Να γραφεί πρόγραμμα το οποίο :
α) θα διαβάζει μέχρι 50 το πολύ ακέραιους αριθμούς διάφορους του 100,

β) θα υπολογίζει και θα εμφανίζει τον μέσο όρο τους,
γ) θα υπολογίζει και θα εμφανίζει πόσοι απ' αυτούς είναι άρτιοι, πόσοι περιττοί και πόσοι μηδέν.
Να γίνεται τερματισμός της επανάληψης αν δοθεί στην είσοδο η τιμή 100.

8. Να γραφεί πρόγραμμα που θα υπολογίζει και θα εμφανίζει το άθροισμα των πρώτων αριθμών που είναι μικρότεροι του 1000. Πρώτος είναι ένας ακέραιος αριθμός όταν διαιρείται μόνο με τον εαυτό του και τη μονάδα.

9. Να γραφεί πρόγραμμα το οποίο θα υπολογίζει τον μικρότερο ακέραιο, ο οποίος είναι περιττός και το γινόμενο των ψηφίων του είναι διπλάσιο από το άθροισμά τους.

10. Σύμφωνα με τον κώδικα οδικής κυκλοφορίας, όταν ένας οδηγός κάνει μια παράβαση η τροχαία του επιβάλει ποινή απο 5 μέχρι 40 βαθμούς,ανάλογα με το είδος της παράβασης. Για κάθε νέα παράβαση οι βαθμοί ποινής αθροίζονται με τους προηγούμενους. Αν το άθροισμα ποινών υπερβεί τους 40 βαθμούς, αφαιρείται το δίπλωμα για ένα τρίμηνο, ενώ αν υπερβεί τους 60 αφαιρείται για ένα έτος.
Να γραφεί πρόγραμμα το οποίο :
α. θα διαβάζει το πλήθος των οδηγών που συμπλήρωσαν 5 παραβάσεις,
β. θα διαβάζει για κάθε οδηγό τους βαθμούς ποινής κάθε παράβασης, ελέγχοντας την εγκυρότητά τους,
γ. θα βρίσκει το άθροισμα των βαθμών ποινής για κάθε οδηγό και θα εμφανίζει ανάλογα το μήνυμα 'Σύνολο βαθμών ποινής κάτω από 40' ή 'Αφαιρείται το δίπλωμα για 3 μήνες' ή 'Αφαιρείται το δίπλωμα για 1 χρόνο'.

11. Μία ελαστική σφαίρα αφήνεται να πέσει ελεύθερη από ύψος 1000 μέτρων. Μετά από κάθε πρόσκρουση στο έδαφος χάνει ενέργεια, με αποτέλεσμα να φτάνει σε ύψος μικρότερο από το προηγούμενο κατά 10 %. Να γραφεί πρόγραμμα το οποίο θα υπολογίζει και θα εμφανίζει τον αριθμό των προσκρούσεων της σφαίρας στο έδαφος, μετά από τις οποίες το ύψος της αναπήδησης θα είναι μικρότερο από 1 μέτρο.

12. Για την πρόσβαση στο σύστημα ATM μιας τράπεζας ζητείται από τον χρήστη, αφού εισάγει την κάρτα του στο σχετικό μηχάνημα, να πληκτρολογήσει τον κωδικό PIN. Αν ο χρήστης δώσει τον σωστό κωδικό επιτρέπεται η πρόσβαση του στο σύστημα και δίνεται το μήνυμα «Είδος συναλλαγής». Αν δεν δοθεί ο σωστός κωδικός το σύστημα ζητάει νέα προσπάθεια εμφανίζοντας σχετικό μήνυμα «Δώσε κωδικό». Η διαδικασία αυτή επαναλαμβάνεται και δεύτερη φορά, ενώ αν δοθεί και τρίτη φορά λανθασμένος κωδικός, το πρόγραμμα σταματά την εκτέλεσή του και εμφανίζει το μήνυμα «Μη αποδεκτός κωδικός». Να γραφεί πρόγραμμα που να προσομειώνει τη διαδικασία πρόσβασης στο σύστημα ATM της τράπεζας.
Σημείωση : Η τιμή του κωδικού αριθμού PIN είναι συνήθως τετραψήφιος ακέραιος αριθμός. Μπορείτε να θεωρήσετε ως κωδικό μια τιμή της επιλογής σας.

13. Ο πληθωρισμός σε μία χώρα της Ευρωπαϊκής Ένωσης είναι 5 %. Ένας εργαζόμενος είχε το 2002 μηνιαίο μισθό 1000 € και παίρνει αύξηση κάθε χρόνο 2 % επί του μισθού του. Να γραφεί πρόγραμμα που θα υπολογίζει και θα εμφανίζει σε πόσα χρόνια η αγοραστική δύναμη του εργαζόμενου θα μειωθεί στο μισό (θα γίνει δηλαδή 500 €). Να θεωρήσετε ότι το ποσό του πληθωρισμού και της ετήσιας αύξησης μισθού παραμένουν σταθερά.
Σημείωση : Το 2003 ο μισθός του θα αυξηθεί κατά 2 % δηλαδή θα γίνει 1000 + 1000 * 2 % = 1020 € αλλά συγχρόνως θα μειωθεί και κατά 5 % που είναι ο πληθωρισμός δηλαδή θα γίνει 1020 – 1020 * 5 % = 969 €.

14. Σε μια πόλη κυκλοφορούν 40.000 αυτοκίνητα. Σύμφωνα με σχετική μελέτη του Υπουργείου Συγκοινωνιών αναμένεται αύξηση των αυτοκινήτων με μέσο ρυθμό 6 % το χρόνο. Να γραφεί πρόγραμμα το οποίο θα υπολογίζει και θα εμφανίζει τον αριθμό των αυτοκινήτων της πόλης αυτής μετά από 10 χρόνια.

15. Δύο παίκτες ρίχνουν διαδοχικά ένα ζάρι. Αν ο αριθμός της ζαριάς είναι μονός, τότε ο πρώτος παίκτης κερδίζει έναν πόντο. Αν συμβεί να είναι ζυγός, κερδίζει ένα πόντο ο δεύτερος παίκτης. Να γραφεί πρόγραμμα το οποίο θα διαβάζει τον αριθμό της ζαριάς κάθε παίκτη μετά από 50 προσπάθειες του καθενός και θα βρίσκει και θα εμφανίζει ποιος απο τους δύο παίκτες είναι ο νικητής.

16. Στο ράφι ενός super market βρίσκονται τέσσερις διαφορετικές συσκευασίες ενός απορρυπαντικού πλυντηρίου. Να γραφεί πρόγραμμα το οποίο :

α. θα διαβάζει το βάρος σε Kg και την τιμή κάθε συσκευασίας (σε €), για 5 συσκευασίες συνολικά,

β. θα υπολογίζει και θα εμφανίζει ποιά συσκευασία είναι οικονομικά συμφέρουσα.

Ως κριτήριο να χρησιμοποιηθεί η τιμή ανά Kg απορρυπαντικού.

17. Ένας φοιτητής θέλει να αγοράσει ένα υπολογιστικό σύστημα αξίας 1200 €. Τον υπολογιστή θα τον εξωφλήσει σταδιακά, δίνοντας κάθε εβδομάδα ποσό διπλάσιο από την προηγούμενη, αρχίζοντας την πρώτη εβδομάδα με 120 €. Να γραφεί πρόγραμμα το οποίο :

α. θα υπολογίζει και θα εμφανίζει μετά από πόσες εβδομάδες θα εξωφλήσει το υπολογιστικό σύστημα.

β. θα υπολογίζει, θα ελέγχει και θα εμφανίζει πιθανό περίσσευμα χρημάτων.

18. Σε ένα υποκατάστημα μιας τράπεζας λειτουργούν τέσσερα ταμεία (1, 2, 3 και 4). Για την καλύτερη οργάνωσης της λειτουργίας της, η τράπεζα χρειάζεται στατιστικά στοιχεία για τα άτομα που εξυπηρετεί κάθε ταμείο την ημέρα. Να γραφεί πρόγραμμα το οποίο :

α. θα διαβάζει τον αριθμό του ταμείου που εξυπηρετεί κάθε πελάτη. Το τέλος της ημέρας θα δηλώνεται όταν δοθεί ως αριθμός ταμείου το 0,

β. θα υπολογίζει και θα εμφανίζει τον αριθμό των ατόμων που εξυπηρετήθηκαν από κάθε ταμείο και από το υποκατάστημα συνολικά,

γ. θα υπολογίζει και θα εμφανίζει το ποσοστό ατόμων που εξυπηρετήθηκαν από κάθε ταμείο ξεχωριστά.

19. Να γραφεί πρόγραμμα το οποίο :

α. θα ζητάει από το πληκτρολόγιο την τιμή ενός θετικού ακεραίου αριθμού N,

β. θα υπολογίζει και θα εμφανίζει την τιμή των αθροισμάτων :

$$S_1 = 1 - 2 + 3 - 4 + 5 - \ldots + (2N+1)$$

$$S_2 = 5^2 + 10^2 + 15^2 + 20^2 + (5N)^2$$

7. ΠΙΝΑΚΕΣ

Ένας πίνακας (array) είναι μια συλλογή από μεταβλητές του ιδίου τύπου οι οποίες αναφέρονται με ένα κοινό όνομα. Ένας πίνακας αποτελείται από διαδοχικές θέσεις μνήμης. Η χαμηλότερη διεύθυνση αντιστοιχεί στο πρώτο στοιχείο, ενώ η υψηλότερη διεύθυνση αντιστοιχεί στο τελευταίο στοιχείο. Ένας πίνακας μπορεί να έχει απο μια διάσταση μέχρι πολλές διαστάσεις. Η προσπέλαση σε ένα συγκεκριμένο στοιχείο του πίνακα γίνεται μέσω δεικτών.

7.1 Μονοδιάστατοι Πίνακες

Η γενική μορφή μιας δήλωσης μονοδιάστατου πίνακα είναι:

τύπος μεταβλητή[μέγεθος];

Ο *τύπος* δηλώνει το βασικό τύπο του πίνακα. Ο βασικός τύπος καθορίζει τον τύπο δεδομένου για κάθε στοιχείο του πίνακα. Η *μεταβλητή* καθορίζει το όνομα του πίνακα. Το *μέγεθος* καθορίζει τον αριθμό των στοιχείων που θα περιέχει ο πίνακας.

Παράδειγμα

```
#include <stdio.h>
int main()
{
        int table[10], i;
        for (i = 0;i < 10; i++)
                table[i] = i;
        return 0;
}
```

Στην C, όλοι οι πίνακες χρησιμοποιούν το μηδέν σαν δείκτη για το πρώτο τους στοιχείο. Έτσι το παραπάνω παράδειγμα, δηλώνει έναν ακέραιο πίνακα με 10 στοιχεία, από **table[0]** έως **table[9]**.

Για ένα μονοδιάστατο πίνακα, το συνολικό μέγεθος υπολογίζεται ως εξής:

Συνολικό μέγεθος σε bytes = sizeof(τύπος) * μήκος πίνακα

Παράδειγμα

```
#include <stdio.h>
int main()
{
        int table[10], size;
        size = sizeof(int) * 10;
        printf("Το συνολικό μέγεθος σε bytes είναι %d.\n",size);
        return 0;
}
```

Οι πίνακες χρησιμοποιούνται πολύ στον προγραμματισμό, γιατί μας επιτρέπουν να χειριζόμαστε εύκολα πολλές μεταβλητές σχετικές μεταξύ τους. Για παράδειγμα, η χρήση των πινάκων διευκολύνει τον υπολογισμό της μέσης τιμής μιας σειράς αριθμών.

Παράδειγμα

```
#include <stdio.h>
int main()
{
        int table[30], i, sum = 0;
        for (i = 0; i < 30; i++) {
                printf("Δώσε το table[%d] στοιχείο : ",i + 1);
                scanf("%d",&table[i]);
                sum += table[i];
        }
        printf("Το άθροισμα των στοιχείων του πίνακα είναι %d.\n",sum);
        return 0;
}
```

Η γλώσσα C, δεν εκτελεί έλεγχο ορίων στους πίνακες, έτσι είναι πολύ εύκολο να ξεπεράσουμε το τέλος ενός πίνακα. Αν ξεπεράσουμε το τέλος ενός πίνακα κατά τη λειτουργία εκχώρησης τιμής, τότε οι τιμές που θα εκχωρήσουμε θα αποτελούν δεδομένα κάποιας άλλης μεταβλητής ή ακόμη και ενός τμήματος του κώδικα του προγράμματος. Μπορούμε λοιπόν, να τοποθετήσουμε το δείκτη ενός πίνακα μεγέθους N πέρα από το N, χωρίς να προκαλέσουμε μηνύματα λάθους χρόνου μεταγλώττισης ή χρόνου εκτέλεσης, ακόμη και αν αυτή η ενέργεια είναι πιθανόν να «κρεμάσει» τον υπολογιστή μας. Σαν προγραμματιστές, έχουμε την ευθύνη να εξασφαλίζουμε ότι όλοι οι πίνακες είναι αρκετά μεγάλοι ώστε να κρατάνε τα στοιχεία που θα τοποθετηθούν σ' αυτούς από το πρόγραμμα, όπως είμαστε και υπεύθυνοι για τον έλεγχο των ορίων όταν είναι απαραίτητος.

Παράδειγμα

```
#include <stdio.h>
int main()
{
        int table[10], i;
        for (i = 0; i < 100; i++) {
                printf("Δώσε το table[%d] στοιχείο : ",i + 1);
                scanf("%d",&table[i]);
        }
        return 0;
}
```

Στο παραπάνω παράδειγμα, ό βρόχος θα επαναληφθεί 100 φορές παρόλο που ο πίνακας table έχει μήκος δέκα μόνο στοιχείων. Η γλώσσα C δεν διαθέτει έλεγχο ορίων στους πίνακες, γιατί σχεδιάστηκε για να αντικαθιστά τη γλώσσα assembly στις περισσότερες περιπτώσεις. Για να το πετύχει αυτό, η C δεν διαθέτει σχεδόν κανένα έλεγχο λαθών, γιατί έτσι επιβραδύνεται η εκτέλεση του προγράμματος.

Άσκηση 5.1.1

Να γραφεί πρόγραμμα το οποίο :
α) θα καταχωρεί σ' έναν πίνακα 50 ακέραιες τιμές που δίνονται από το πληκτρολόγιο,
β) θα υπολογίζει και θα εμφανίζει τον μέσο όρο των τιμών του πίνακα,
γ) θα υπολογίζει και θα εμφανίζει το πλήθος τιμών που είναι μικρότερες από το μέσο όρο, καθώς και τις θέσεις αυτών στον πίνακα,
δ) θα βρίσκει και θα εμφανίζει τη μικρότερη τιμή και τη θέση της στον πίνακα.

```
#include <stdio.h>
int main()
{
	int table[50], i, sum = 0, count = 0, min, pos = 0;
	float average;
	for (i = 0; i < 50; i++) {
		printf("Δώσε το table[%d] στοιχείο : ",i + 1);
		scanf("%d",&table[i]);
		sum += table[i];
	}
	average = sum / 50.0;
	printf("Ο μέσος όρος των τιμών του πίνακα είναι %.2f\n",average);
	for (i = 0; i < 50; i++)
		if (table[i] < average) {
			count++;
			printf("Θέση στοιχείου : %d\n", i + 1);
		}
	printf("Πλήθος στοιχείων μικρότερων του μέσου όρου : %d\n",count);
	min = table[0];
	for (i = 1; i < 50; i++)
		if (table[i] < min) {
			min = table[i];
			pos = i;
		}
	printf("Μικρότερο στοιχείο του πίνακα : %d\n",min);
	printf("Θέση μικρότερου στοιχείου : %d\n",pos + 1);
	return 0;
}
```

Άσκηση 5.1.2

Να γραφεί πρόγραμμα το οποίο :
α) θα διαβάζει από το πληκτρολόγιο 20 ακέραιες τιμές τις οποίες θα καταχωρεί σε έναν πίνακα,
β) θα βρίσκει και θα εμφανίζει τα στοιχεία που είναι μικρότερα από 40 και μεγαλύτερα από 5,
γ) θα υπολογίζει και θα εμφανίζει το άθροισμα των αριθμών του προηγούμενου ερωτήματος.

```
#include <stdio.h>
int main()
{
	int table[20], i, sum = 0;
	for (i = 0; i < 20; i++) {
		printf("Δώσε το table[%d] στοιχείο του πίνακα : ",i + 1);
		scanf("%d",&table[i]);
```

```
        }
        for (i = 0; i < 20; i++)
                if (table[i] > 5 && table[i] < 40) {
                        sum += table[i];
                        printf("%4d",table[i]);
                }
        printf("\nΤο άθροισμα των στοιχείων είναι %d\n",sum);
        return 0;
}
```

Άσκηση 5.1.3

Να γραφεί πρόγραμμα το οποίο :
α) θα καταχωρεί σε έναν πίνακα 40 ακέραιες τιμές από 20 έως 30, οι οποίες δίνονται από το πληκτρολόγιο,
β) θα βρίσκει και θα εμφανίζει την τιμή που εμφανίζεται με τη μεγαλύτερη συχνότητα.

```
#include <stdio.h>
int main()
{
        int table[40], s[11], i, pos, max;
        for (i = 0; i < 40; i++) {
                printf("Δώσε το table[%d] στοιχείο (≥ 20 και ≤ 30) : ",i + 1);
                do {
        scanf("%d",&table[i]);
                } while (table[i] < 20 || table[i] > 30);
        }
        for (i = 0; i < 11; i++) s[i] = 0;
        for (i = 0; i < 40; i++) {
                pos = table[i] – 20;
                s[pos]++;
        }
        max = s[0];
        for (i = 1; i < 11; i++)
                if (s[i] > max)
                                max = s[i];
        printf("Η τιμή με την μεγαλύτερη συχνότητα είναι η %d\n",max + 20);
        return 0;
}
```

Άσκηση 5.1.4

Δίνεται ο πίνακας table που περιέχει τα ακόλουθα στοιχεία :

5, 3, 1, 27, 15, 4, 9, 8, 7, 12, 32, 10, 5, 4, 2

Να γραφεί πρόγραμμα το οποίο θα εμφανίζει στην οθόνη τα στοιχεία του πίνακα κατά αύξουσα και φθίνουσα σειρά.

```
#include <stdio.h>
int main()
{
        int table[15] = {5,3,1,27,15,4,9,8,7,12,32,10,5,4,2}, i, j, temp;
        //αλγόριθμος ταξινόμησης bubble sort – αύξουσα ταξινόμηση
        for (i = 0; i < 15; i++)
```

```
                for (j = 14; j > i; j--)
                        if (table[j–1] > table[j]) {
                                temp = table[j];
                                table[j] = table[j-1];
                                table[j-1] = temp;
                        }
        for (i = 0; i < 15; i++)
                printf("%4d",table[i]);
        printf("\n");
        //αλγόριθμος ταξινόμησης bubble sort – φθίνουσα ταξινόμηση
        for (i = 0; i < 15; i++)
                for (j = 14; j > i; j--)
                        if (table[j–1] < table[j]) {
                                temp = table[j];
                                table[j] = table[j-1];
                                table[j-1] = temp;
                        }
        for (i = 0; i < 15; i++)
                printf("%4d",table[i]);
        printf("\n");
        return 0;
}
```

Άσκηση 5.1.5

Να γραφεί πρόγραμμα το οποίο :
α) θα δημιουργεί έναν πίνακα 20 ακεραίων αριθμών,
β) θα ζητάει μία ακέραια τιμή από το πληκτρολόγιο,
γ) σε περίπτωση που αυτή η τιμή υπάρχει στον πίνακα θα εμφανίζει τη θέση που βρέθηκε.

```
#include <stdio.h>
int main()
{
        int table[20], key, i, position = -1;
        for (i = 0; i < 20; i++) {
                printf("Δώσε το table[%d] στοιχείο του πίνακα : ",i + 1);
                scanf("%d",&table[i]);
        }
        printf("Δώσε την τιμή που αναζητάς : ");
        scanf("%d",&key);
        i = 0;
        // Σειριακή αναζήτηση, τερματίζει όταν βρεθεί το στοιχείο
        while (i < 20 && position == -1)
                if (table[i] == key)
                        position = i;
                else
                        i++;
        if (position != -1)
                printf("Το στοιχείο βρέθηκε στη θέση %d\n",position + 1);
        return 0;
}
```

Άσκηση 5.1.6

Δίνεται ο ταξινομημένος μονοδιάστατος πίνακας source, 20 θέσεων, που περιέχει τα ακόλουθα στοιχεία :

1, 6, 7, 8, 9, 11, 12, 14, 15, 16, 17, 22, 25, 28, 30, 33, 34, 65, 67, 78

Να γραφεί πρόγραμμα το οποίο :

α) θα διαβάζει ένα αριθμό n,

β) θα δημιουργεί έναν νέο μονοδιάστατο πίνακα target, 21 θέσεων, ο οποίος θα περιέχει όλα τα στοιχεία του αρχικού πίνακα και τον αριθμό n στη σωστή θέση, ώστε ο νέος πίνακας να είναι ταξινομημένος.

```
#include <stdio.h>
int main()
{
	int source[20] = {1,6,7,8,9,11,12,14,15,16,17,22,25,28,30,33,34,65,67,78};
	int n, i = 0, j, complete = 0, target[21];
	printf("Δώσε τον αριθμό που θέλεις να εισάγεις : ");
	scanf("%d",&n);
	while (i < 20 && !complete)
		if (source[i] < n) {
			target[i] = source[i];
			i++;
		}
		else {
				target[i] = n;
				complete = 1;
		}
	if (!complete)
		target[i] = n;
	else
		for (j = i; j < 20; j++)
			target[j+1] = source[j];
	for (i = 0; i < 21; i++)
		printf("%4d",target[i]);
	printf("\n");
	return 0;
}
```

Άσκηση 5.1.7

Δίνεται ο ταξινομημένος πίνακας Α, 6 θέσεων, που περιέχει τα στοιχεία 1, 3, 5, 7, 9, 11 και ο ταξινομημένος πίνακας Β, 7 θέσεων, που περιέχει τα στοιχεία 2, 4, 6, 8, 10, 12, 14. Να γραφεί πρόγραμμα, το οποίο θα δημιουργεί έναν ταξινομημένο πίνακα C, 13 θέσεων, ο οποίος θα περιέχει όλα τα στοιχεία των πινάκων Α και Β.

```
#include <stdio.h>
#define N 6
#define M 7
int main()
{
	int A[N] = {1,3,5,7,9,11}, B[M]={2,4,6,8,10,12,14}, C[M+N],i=0, j=0, k=0,
n;
	 while (i < N && j < M)
```

```
        if (A[i] < B[j]) {
                C[k] = A[i];
                k++;
                i++;
        }
        else {
                C[k] = B[j];
                k++;
                j++;
        }
        if (i >= N)
                for (n = k; n < N+M; n++) {
                        C[n] = B[j];
                        j++;
                }
        else
                for (n = k; n < N+M; n++) {
                        C[n] = A[i];
                        i++;
                }
        for (i = 0; i < N+M; i++)
                printf("%4d",C[i]);
        printf("\n");
        return 0;
}
```

7.2 Δισδιάστατοι Πίνακες

Η C επιτρέπει τη χρήση πολυδιάστατων πινάκων. Η απλούστερη μορφή πολυδιάστατου πίνακα είναι ο δισδιάστατος πίνακας. Ουσιαστικά, ένας δισδιάστατος πίνακας είναι ένας πίνακας που αποτελείται από μονοδιάστατους πίνακες. Για να δηλώσουμε ένα δισδιάστατο πίνακα ακεραίων, μεγέθους 5 ,10 θα γράψουμε:

```
int table[5][10];
```

Σε αντίθεση με άλλες γλώσσες προγραμματισμού, οι οποίες χρησιμοποιούν κόμματα για να χωρίζουν τις διαστάσεις των πινάκων, η C τοποθετεί την κάθε διάσταση σε δικό της σετ αγκυλών.

Παράδειγμα

```
#include <stdio.h>
int main()
{
        int i, j, table[4][5];
        // Ανάγνωση στοιχείων
        for(i = 0; i < 4; i++)
                for (j = 0; j < 5; j++)
                        table[i][j] = i * j;
        // Εμφάνιση στοιχείων
```

```
        for(i = 0; i < 4; i++) {
                for (j = 0; j < 5; j++)
                        printf("%4d",table[i][j]);
                printf("\n");
        }
        return 0;
}
```

Η γλώσσα C αποθηκεύει τους δισδιάστατους πίνακες σε μια μήτρα με γραμμές και στήλες, όπου ο πρώτος δείκτης δηλώνει τη γραμμή και ο δεύτερος δείκτης δηλώνει τη στήλη. Αυτή η δομή σημαίνει ότι ο δεξιότερος δείκτης αλλάζει ταχύτερα από τον αριστερότερο όταν προσπελαύνουμε τα στοιχεία του πίνακα με τη σειρά που η C τα αποθηκεύει στη μνήμη.

Για τους δισδιάστατους πίνακες, μπορούμε να χρησιμοποιήσουμε τον παρακάτω τύπο για να βρούμε τον αριθμό των bytes μνήμης:

Συνολικός αριθμός bytes = γραμμή * στήλη * sizeof(τύπος δεδομένων)

Παράδειγμα

```
#include <stdio.h>
int main()
{
        int size, table[4][5];
        size = 4 * 5 * sizeof(int);
        printf("Το συνολικό μεγεθος bytes είναι %d.\n",size);
        return 0;
}
```

Άσκηση 5.2.1

Να γραφεί πρόγραμμα το οποίο :
α) θα διαβάζει από το πληκτρολόγιο τις τιμές ενός τετραγωνικού πίνακα ακεραίων A[100,100],
β) θα υπολογίζει και θα εμφανίζει το άθροισμα των περιφερειακών στοιχείων του πίνακα (δηλαδή των στοιχείων της $1^{ης}$ γραμμής, της $100^{ης}$ στήλης, της $100^{ης}$ γραμμής και της $1^{ης}$ στήλης).

```
#include <stdio.h>
int main()
{
        int i, j, A[100][100], sum = 0;
        for (i = 0; i < 100; i++)
                for (j = 0; j < 100; j++) {
                        printf("Δώσε ακέραιο αριθμό : ");
                        scanf("%d",&A[i][j]);
                }
        for (i = 0; i < 100; i++)
                sum += A[0][i] + A[99][i] + A[i][0] + A[i][99];
        printf("Το άθροισμα των περιφερειακών στοιχείων είναι %d\n",sum);
        return 0;
}
```

Άσκηση 5.2.2

Να γραφεί πρόγραμμα το οποίο :
α) θα διαβάζει 100 αριθμούς και θα τους καταχωρεί σε έναν μονοδιάστατο πίνακα A[100],
β) θα αντιγράφει τα στοιχεία του πίνακα Α σε έναν πίνακα Β[10,10] γραμμή πρός γραμμή (δηλαδή τα πρώτα 10 στοιχεία του πίνακα Α θα καταχωρηθούν στην $1^η$ γραμμή του πίνακα Β, τα επόμενα 10 στην $2^η$ γραμμή κ.ο.κ.).

```
# include <stdio.h>
 int main()
{
        int i, j, A[100], B[10][10];
        for (i = 0; i < 100;i++) {
                printf("Δώσε αριθμό : ");
                scanf("%d",&A[i]);
        }
        for (i = 0; i < 10; i++) {
                for (j = 0; j < 10; j++) {
                        B[i][j] = A[i * 10 + j];
                        printf("%4d",B[i][j]);
                }
                printf("\n");
        }
        return 0;
}
```

Άσκηση 5.2.3

Να γραφεί πρόγραμμα το οποίο θα δημιουργεί και θα εμφανίζει στην οθόνη τον παρακάτω δισδιάστατο πίνακα :

1	2	3	4	5
2	4	6	8	10
3	6	9	12	15
4	8	12	16	20
5	10	15	20	25

```
# include <stdio.h>
int main()
{
        int i, j, A[5][5];
        for (i = 0; i < 5; i++) {
                for (j = 0; j < 5; j++) {
                        A[i][j] = (i + 1) * (j + 1);
                        printf("%4d",A[i][j]);
                }
                printf("\n");
        }
        return 0;
}
```

Άσκηση 5.2.4

Να γραφεί πρόγραμμα το οποίο :

α) θα δέχεται τις τιμές των στοιχείων A[i,j] ενός τετραγωνικού πίνακα 100 x 100,
β) θα υπολογίζει και θα τυπώνει το άθροισμα των στοιχείων της κύριας διαγωνίου,
γ) θα υπολογίζει και θα τυπώνει το άθροισμα των στοιχείων που βρίσκονται πάνω και κάτω από την κύρια διαγώνιο.

Σε κάθε τετραγωνικό πίνακα ισχύουν τα εξής :

- τα στοιχεία της κύριας διαγωνίου έχουν ίσους δείκτες (i = j).
- για τα στοιχεία κάτω από την κύρια διαγώνιο, ο δείκτης γραμμής είναι μεγαλύτερος από τον δείκτη στήλης (i > j).
- για τα στοιχεία πάνω από την κύρια διαγώνιο, ο δείκτης στήλης είναι μεγαλύτερος από τον δείκτη γραμμής (i < j).

```
#include <stdio.h>
int main()
{
        int i, j, A[100][100], sumd = 0, sumup = 0, sumdown = 0;
        for (i = 0; i < 100; i++)
                for (j = 0; j < 100; j++) {
                        printf("Δώσε το A[%d][%d] στοιχείο : ",i, j);
                        scanf("%d",&A[i][j]);
                }
        for (i = 0; i < 100; i++)
                for (j = 0; j < 100; j++)
                        if (i == j)
                                sumd += A[i][j];
                        else if (i > j)
                                sumdown += A[i][j];
                        else
                                sumup += A[i][j];
        printf("Άθροισμα στοιχείων κύριας διαγωνίου %d\n",sumd);
        printf("Άθροισμα στοιχείων κάτω από την κύρια διαγώνιο %d\n",sumdown);
        printf("Άθροισμα στοιχείων πάνω από την κύρια διαγώνιο %d\n",sumup);
        return 0;
}
```

7.3 Πολυδιάστατοι Πίνακες

Όπως αναφέραμε και προηγουμένως, η C επιτρέπει τη χρήση πινάκων με περισσότερες από μια διαστάσεις. Η γενική μορφή της δήλωσης ενός πολυδιάστατου πίνακα είναι:

τύπος όνομα[μέγεθος 1] [μέγεθος 2] ... [μέγεθος N];

Η παρακάτω εντολή δημιουργεί έναν ακέραιο πίνακα 4 x 3 x 2:

int table[4][3][2];

Οι πίνακες τριών ή περισσότερων διαστάσεων δεν χρησιμοποιούνται συχνά λόγω της ποσότητας μνήμης που απαιτούν. Όπως αναφέραμε, η C κρατάει χώρο στη μνήμη για όλα τα στοιχεία ενός πίνακα.

7.4 Απόδοση Αρχικών Τιμών σε Πίνακες

Η C επιτρέπει την απόδοση αρχικών τιμών σε πίνακες. Η γενική μορφή της απόδοσης αρχικών τιμών σε ένα πίνακα μοιάζει με των άλλων μεταβλητών, όπως φαίνεται παρακάτω:

τύπος όνομα[μέγεθος 1][μέγεθος 2] ... [μέγεθος N] = {λίστα τιμών};

Η λίστα τιμών είναι μια λίστα σταθερών που χωρίζονται μεταξύ τους με κόμματα, ο τύπος τους είναι συμβατός με το βασικό τύπο του πίνακα. Αυτή η εντολή θα τοποθετήσει την πρώτη σταθερά στην πρώτη θέση του πίνακα, τη δεύτερη σταθερά στη δεύτερη θέση, κ.λ.π.

Παράδειγμα

```
#include <stdio.h>
int main()
{
        int i, table[10] = {1,2,3,4,5,6,7,8,9,10};
        for (i = 0; i < 10; i++)
                printf("%4d",table[i]);
        printf("\n");
        return 0;
}
```

Στο παραπάνω παράδειγμα το table[0] θα πάρει την τιμή 1 και το table[9] θα πάρει την τιμή 10.

Οι πίνακες χαρακτήρων που περιέχουν αλφαριθμητικά επιτρέπουν μια πιο σύντομη απόδοση αρχικών τιμών με τη μορφή:

char όνομα[μέγεθος] = «αλφαριθμητικό»;

Παράδειγμα

```
#include <stdio.h>
int main()
{
        char str[8] = "Γιώργος";
        puts(str);
        return 0;
}
```

Η παραπάνω δήλωση του αλφαριθμητικού **str** είναι ισοδύναμη με την ακόλουθη:

```
char str[8] = {'Γ', 'ι', 'ώ', 'ρ', 'γ', 'ο', 'ς', '\0'};
```

Επειδή τα αλφαριθμητικά στη C πρέπει να τερματίζουν σε μηδενικό (null), θα πρέπει να είμαστε βέβαιοι ότι ο πίνακας που δηλώνουμε έχει το κατάλληλο μέγεθος. Έτσι εξηγείται γιατί το **str** πρέπει να έχει μήκος 8 χαρακτήρες, παρ' όλο που το «Γιώργος»

έχει μήκος 7 χαρακτήρες. Όταν χρησιμοποιούμε την αλφαριθμητική σταθερά (Παράδειγμα 5.1.4.2), ο μεταγλωττιστής παρέχει αυτόματα το τελικό μηδενικό.

Οι πολυδιάστατοι πίνακες παίρνουν αρχικές τιμές κατά τον ίδιο τρόπο που παίρνουν αρχικές τιμές και οι μονοδιάστατοι πίνακες.

Παράδειγμα

```
#include <stdio.h>
int main()
{
        int table[10][2] = {
                                        1, 1,
                                        2, 4,
                                        3, 9,
                                        4, 16,
                                        5, 25,
                                        6, 36,
                                        7, 49,
                                        8, 64,
                                        9, 81,
                                        10, 100
        };
        int i, j;
        for (i = 0; i < 10; i++) {
                for (j = 0;j < 2; j++)
                        printf("%5d",table[i][j]);
                printf("\n");
        }
        return 0;
}
```

Όπως φαίνεται από τα προηγούμενα (ιδίως για τα αλφαριθμητικά), η μέτρηση των στοιχείων του πίνακα, η οποία απαιτείται για τον καθορισμό της σωστής διάστασης του πίνακα, είναι κουραστική δουλειά. Μπορούμε να βάλουμε την C να διαστασιολογεί αυτόματα τους πίνακες, χρησιμοποιώντας *πίνακες χωρίς μέγεθος*. Σε μια εντολή απόδοσης αρχικών τιμών σε πίνακα, αν δεν καθορίσουμε το μέγεθος του πίνακα, η C θα δημιουργήσει αυτόματα έναν πίνακα αρκετά μεγάλο ώστε να χωράει όλες τις υπάρχουσες τιμές.

Παράδειγμα

```
#include <stdio.h>
int main()
{
        char str[] = "Αυτό είναι ένα μήνυμα";
        printf("Το μέγεθος του %s μηνύματος είναι %d\n",str, sizeof str);
        return 0;
}
```

Η C δεν περιορίζει την απόδοση αρχικών τιμών σε πίνακες χωρίς μέγεθος, μόνο στους μονοδιάστατους πίνακες. Για τους πολυδιάστατους πίνακες θα πρέπει να

καθορίσουμε όλες τις διαστάσεις εκτός από την αριστερότερη, για να μπορέσει η C να εκτελέσει τους κατάλληλους υπολογισμούς. Με αυτόν τον τρόπο, μπορούμε να δημιουργούμε πίνακες με διάφορα μήκη ενώ ο μεταγλωττιστής θα τους παρέχει τον κατάλληλο χώρο αποθήκευσης.

Παράδειγμα

```
#include <stdio.h>
int main()
{
        int table[][2] = {
                                1, 1,
                                2, 4,
                                3, 9,
                                4, 16,
                                5, 25,
                                6, 36,
                                7, 49,
                                8, 64,
                                9, 81,
                                10, 100
        };
        int i, j;
        for (i = 0; i < 10; i++) {
                for (j = 0;j < 2; j++)
                        printf("%5d",table[i][j]);
                printf("\n");
        }
        return 0;
}
```

7.5 Ασκήσεις

1. Να γραφεί πρόγραμμα το οποίο :
α. θα δημιουργεί έναν μονοδιάστατο πίνακα 30 ακεραίων αριθμών,
β. θα εμφανίζει τα στοιχεία του πίνακα,
γ. θα υπολογίζει και θα εκτυπώνει το άθροισμα των στοιχείων του πίνακα.
2. Να γραφεί πρόγραμμα το οποίο :
α. θα δημιουργεί έναν μονοδιάστατο πίνακα 30 ακεραίων αριθμών,
β. θα εμφανίζει τα στοιχεία του πίνακα,
γ. θα βρίσκει και θα τυπώνει το μέγιστο και το ελάχιστο στοιχείο του πίνακα.
3. Να γραφεί πρόγραμμα το οποίο :
α. θα εισάγει τις τιμές των στοιχείων ενός μονοδιάστατου πίνακα 30 ακεραίων αριθμών,
β. θα υπολογίζει και θα τυπώνει τη μέση τιμή των στοιχείων του πίνακα,
γ. θα υπολογίζει και θα τυπώνει το πλήθος των στοιχείων του πίνακα που είναι μεγαλύτερα από τη μέση τιμή.
4. Να γραφεί πρόγραμμα το οποίο :
α. θα δημιουργεί έναν μονοδιάστατο πίνακα 30 ακεραίων αριθμών,
β. θα υπολογίζει και θα τυπώνει το μέγιστο στοιχείο του πίνακα,

γ. θα υπολογίζει και θα τυπώνει τη θέση του μέγιστου στοιχείου.

5. Σ' έναν μετεωρολογικό σταθμό πρόκειται να γίνει επεξεργασία των θερμοκρασιακών δεδομένων. Να γραφεί πρόγραμμα το οποίο :

α. θα δημιουργεί έναν μονοδιάστατο πίνακα 31 θέσεων, όπου θα καταχωρείται η πρωϊνή ημερήσια θερμοκρασία (ώρα 8 π.μ.) του Ιανουαρίου 2004,

β. θα υπολογίζει και θα τυπώνει στην οθόνη τη μέση πρωϊνή θερμοκρασία του μήνα Ιανουαρίου,

γ. θα εμφανίζει στην οθόνη τις ημερομηνίες με θερμοκρασία μικρότερη των 0^o C.

6. Ένας φυσικός εκτελεί ένα πείραμα και επαναλαμβάνει τη μέτρηση ενός μεγέθους 200 φορές. Να γραφεί πρόγραμμα το οποίο :

α. θα διαβάζει την τιμή κάθε μέτρησης,

β. θα υπολογίζει και θα τυπώνει την μέση τιμή του μεγέθους,

γ. θα υπολογίζει και θα τυπώνει την τυπική απόκλιση των μετρήσεων,

Η τυπική απόκλιση δίνεται από τη σχέση $\sigma = \dfrac{\sqrt{\Sigma(x - x_\mu)^2}}{n}$

7. Να γραφεί πρόγραμμα το οποίο :

α. θα καταχωρεί τιμές σε έναν πίνακα 100 θέσεων ως εξής : στις θέσεις που είναι πολλαπλάσιες του 5 να καταχωρείται ο αριθμός 1, ενώ στις υπόλοιπες ο αριθμός 0,

β. θα εμφανίζει τα στοιχεία του πίνακα στην οθόνη.

8. Στο άθλημα της γυμναστικής κάθε αθλήτρια βαθμολογείται από 10 κριτές με ακέραιο βαθμό της κλίμακας 0 – 10. Ο τελικός βαθμός της προσπάθειας της αθλήτριας προκύπτει από το μέσο όρο βαθμών των κριτών, αφού εξαιρεθούν ο μεγαλύτερος και ο μικρότερος βαθμός. Να γραφεί πρόγραμμα το οποίο :

α. θα καταχωρεί σε πίνακα του βαθμούς των 10 κριτών για την προσπάθεια μιας αθλήτριας στο δίζυγο,

β. θα υπολογίζει και θα εμφανίζει τον τελικό μέσο όρο της βαθμολογίας της αθλήτριας σύμφωνα με την παραπάνω διαδικασία.

9. Ένα μουσείο έχει 30 αίθουσες. Να γραφεί πρόγραμμα το οποίο :

α. θα διαβάζει τον αριθμό των επισκεπτών κάθε αίθουσας σε μια μέρα,

β. θα βρίσκει και θα εμφανίζει ποιά αίθουσα είχε τους περισσότερους επισκέπτες και πόσοι ήταν αυτοί,

γ. θα υπολογίζει και θα εμφανίζει το συνολικό πλήθος επισκεπτών του μουσείου σε μία μέρα.

10. Δίνεται μονοδιάστατος πίνακας Α,Ν στοιχείων, που είναι ακέραιοι αριθμοί. Να γραφεί πρόγραμμα, το οποίο να ταξινομεί με τη μέθοδο της φυσαλίδας τα στοιχεία του πίνακα Α.

11. Δίνονται η έκταση, ο πληθυσμός και το όνομα καθεμιάς από τις 15 χώρες της Ευρωπαϊκής Ένωσης. Να γραφεί πρόγραμμα το οποίο :

α. θα διαβάζει τα παραπάνω δεδομένα,

β. θα εμφανίζει τη χώρα με τη μεγαλύτερη έκταση

γ. θα εμφανίζει τη χώρα με το μικρότερο πληθυσμό και

δ. θα εμφανίζει το μέσο όρο του πληθυσμού των 15 χωρών της Ευρωπαϊκής Ένωσης.

12. Να γραφεί πρόγραμμα το οποίο :

α. θα τοποθετεί σε έναν μονοδιάστατο πίνακα Α τους 10 πρώτους περιττούς αριθμούς σε αύξουσα σειρά, κάνοντας χρήση μιας επαναληπτικής δομής,

β. θα σχηματίζει έναν δεύτερο πίνακα Β του οποίου το στοιχείο Β[i] είναι το άθροισμα των στοιχείων Α[1], Α[2], ... , Α[i] του πρώτου πίνακα (για παράδειγμα το στοιχείο Β[4] είναι το άθροισμα των 4 πρώτων στοιχείων του πίνακα Α),

γ. θα εμφανίζει τα στοιχεία και των δύο πινάκων.

13. Να γραφεί πρόγραμμα το οποίο :
α. θα καταχωρεί σε έναν πίνακα Α, 100 ακέραιες τιμές που δίνονται από το πληκτρολόγιο,
β. θα εμφανίζει τις διαδοχικές τριάδες αριθμών στις οποίες ο μεσαίος αριθμός ισούται με το άθροισμα των άλλων δύο,
γ. θα υπολογίζει και θα εμφανίζει το πλήθος των τριάδων αριθμών που έχουν την παραπάνω ιδιότητα.

14.Να γραφεί πρόγραμμα το οποίο :
α. θα διαβάζει έναν ακέραιο αριθμό Ν από 10 μέχρι 100 (να γίνει έλεγχος εγκυρότητας, ώστε $10 \le N \le 100$),
β. θα διαβάζει Ν ακέραιους αριθμούς και θα τους καταχωρεί σε πίνακα Α[100],
γ. θα δημιουργεί ένα δεύτερο πίνακα Β, ο οποίος θα περιέχει τα στοιχεία του πίνακα Α με την ίδια σειρά έχοντας μετατοπίσει τα μηδενικά στοιχεία στο τέλος του.
Για παράδειγμα, αν ο πίνακας Α είναι της μορφής :

A = [1 0 0 9 0 0 6 4 0 0]

ο πίνακας Β θα πρέπει να έχει τη μορφή :

B = [1 9 6 4 0 0 0 0 0 0]

15. Να γραφεί πρόγραμμα το οποίο :
α. θα διαβάζει έναν ακέραιο θετικό αριθμό N <= 1000 (να γίνεται έλεγχος εγκυρότητας, ώστε $0 < N \le 1000$),
β. θα διαβάζει Ν αριθμούς και θα τους καταχωρεί σε μονοδιάστατο πίνακα Α,
γ. θα δημιουργεί ένα μονοδιάστατο αριθμητικό πίνακα Β με στοιχεία :
A[N], A[N - 1], ... , A[3], A[2], A[1]
Για παράδειγμα αν ο αρχικός πίνακας Α είναι ο :
A = [2 4 5 7 1 0 9]
τότε ο πίνακας Β θα είναι :
B = [9 0 1 7 5 4 2]

16. Να γραφεί πρόγραμμα το οποίο :
α. θα διαβάζει έναν ακέραιο αριθμό Ν από 1 μέχρι 100 (να γίνεται έλεγχος εγκυρότητας, ώστε $1 \le N \le 100$),
β. θα διαβάζει και θα καταχωρεί σε έναν μονοδιάστατο πίνακα Α, Ν αριθμούς, κάθε ένας από τους οποίους μπορεί να είναι ένα από τα ψηφία (να γίνει έλεγχος εγκυρότητας) :
0, 1, 2, 3, ..., 9
γ. θα υπολογίζει και θα εμφανίζει πόσες φορές επαναλαμβάνεται το κάθε ψηφίο στον πίνακα Α.
Για παράδειγμα αν ο πίνακας Α είναι :
A = [1 2 2 3 3 3 4]
τότε το πρόγραμμα θα πρέπει να εμφανίσει :
Το 1 εμφανίζεται 1 φορές
Το 2 εμφανίζεται 2 φορές
Το 3 εμφανίζεται 3 φορές
Το 4 εμφανίζεται 1 φορές

17. Δίνεται ο πίνακας A = [1,4,2,3,1,6,8,2,3,7,9,11,13,10,12,8,9,4,3, 1], ο οποίος περιέχει άρτιους και περιτούς αριθμούς. Να γραφεί πρόγραμμα το οποίο :
α. θα διαχωρίζει τις άρτιες από τις περιτές τιμές και θα τις καταχωρεί σε δύο πίνακες Β[20] (οι άρτιες τιμές) και C[20] (οι περιτές τιμές). Στις κενές θέσεις κάθε πίνακα θα καταχωρείται η τιμή 0,
β. θα υπολογίζει και θα εμφανίζει το πλήθος των άρτιων και περιτών τιμών,
γ. θα υπολογίζει και θα εμφανίζει το άθροισμα των άρτιων και περιτών τιμών.

18. Σε μία αποθήκη βρίσκονται 150 κιβώτια και κάθε ένα από αυτά έχει μοναδικό αριθμό από το 1 μέχρι και το 150 που του αντιστοιχεί. Να γραφεί πρόγραμμα το οποίο :
α. θα διαβάζει τα βάρη σε κιλά των 150 κιβωτίων (τα βάρη των κιβωτίων είναι πραγματικοί αριθμοί διαφορετικοί μεταξύ τους οι οποίοι δεν πρέπει να είναι μικρότεροι των 5 κιλών αλλά ούτε και μεγαλύτεροι των 100 κιλών) και να τα αποθηκεύει σε μονοδιάστατο πίνακα,
β. να βρίσκει και να τυπώνει τον αριθμό του κιβωτίου με το μεγαλύτερο βάρος,
γ. σε περίπτωση που ο αριθμός του κιβωτίου που βρέθηκε στο ερώτημα (β) ξεπερνά το 75, να υπολογίζει και να τυπώνει το μέσο όρο βάρους των κιβωτίων, ο αριθμός των οποίων ξεπερνά το 75, διαφορετικά να βρίσκει και να τυπώνει το μικρότερο βάρος μεταξύ όλων των κιβωτίων ο αριθμός των οποίων είναι μεταξύ του 1 και του 75.

19. Δίνεται πίνακας Α δύο διαστάσεων, που τα στοιχεία του είναι ακέραιοι αριθμοί με Ν γραμμές και Μ στήλες. Να γραφεί πρόγραμμα που να υπολογίζει το ελάχιστο στοιχείο του πίνακα.

20. Κατά τη διάρκεια Διεθνών Αγώνων Στίβου στον ακοντισμό έλαβαν μέρος δέκα (10) αθλητές. Κάθε αθλητής έκανε έξι (6) έγκυρες ρίψεις που καταχωρούνται ως επιδόσεις σε μέτρα. Να γραφεί πρόγραμμα το οποίο :
α. θα εισάγει σε πίνακα δύο διαστάσεων τις επιδόσεις όλων των αθλητών,
β. θα υπολογίζει και καταχωρεί σε μονοδιάστατο πίνακα την καλύτερη από τις επιδόσεις κάθε αθλητή,
γ. θα ταξινομεί τις καλύτερες επιδόσεις των αθλητών που καταχωρήθηκαν στον μονοδιάστατο πίνακα,
δ. θα βρίσκει την καλύτερη επίδοση του αθλητή που πήρε το χάλκινο μετάλλιο (τρίτη θέση).

21. Μια αλυσίδα ξενοδοχείων έχει 5 ξενοδοχεία. Σε έναν μονοδιάστατο πίνακα Α[5] καταχωρούνται τα ονόματα των ξενοδοχείων. Σε ένα άλλο δισδιάστατο πίνακα C[5,12] καταχωρούνται οι εισπράξεις κάθε ξενοδοχείου για κάθε μήνα του έτους 2004, έτσι, ώστε στην i γραμμή καταχωρούνται οι εισπράξεις του i ξενοδοχείου.
Να γραφεί πρόγραμμα, το οποίο :
α. διαβάζει τα στοιχεία των δύο πινάκων,
β. εκτυπώνει το όνομα κάθε ξενοδοχείου και τις ετήσιες εισπράξεις του για το έτος 2004,
γ. εκτυπώνει το όνομα του ξενοδοχείου με τις μεγαλύτερες εισπράξεις για το έτος 2004.

22. Μία αλυσίδα κινηματογράφων έχει δέκα αίθουσες. Τα ονόματα των αιθουσών καταχωρούνται σε ένα μονοδιάστατο πίνακα και οι μηνιαίες εισπράξεις κάθε αίθουσας για ένα έτος καταχωρούνται σε πίνακα δύο διαστάσεων. Να γραφεί πρόγραμμα το οποίο :
α. θα διαβάζει τα ονόματα των αιθουσών,
β. θα διαβάζει τις μηνιαίες εισπράξεις των αιθουσών αυτού του έτους,
γ. θα υπολογίζει τη μέση μηνιαία τιμή των εισπράξεων για κάθε αίθουσα,
δ. θα βρίσκει και θα εμφανίζει τη μικρότερη μέση μηνιαία τιμή,
ε. θα βρίσκει και θα εμφανίζει το όνομα ή τα ονόματα των αιθουσών που έχουν την ανωτέρω μικρότερη μέση μηνιαία τιμή.

23. Σε έναν διαγωνισμό του ΑΣΕΠ έλαβαν μέρος 500 υποψήφιοι. Οι υποψήφιοι διαγωνίστηκαν σε τρείς γραπτές εξετάσεις και θα βαθμολογηθούν με ακέραιους αριθμούς στη βαθμολογική κλίμακα από 0 έως και 100.
Να γραφεί πρόγραμμα το οποίο :

α. θα διαβάζει τα ονόματα των υποψηφίων και θα τα αποθηκεύει σε μονοδιάστατο πίνακα,
β. θα διαβάζει τους τρείς βαθμούς που έλαβε κάθε υποψήφιος και θα τους αποθηκεύει σε δισδιάστατο πίνακα,
γ. θα υπολογίζει το μέσο όρο των βαθμών του κάθε υποψήφιου,
δ. θα εκτυπώνει τα ονόματα των υποψηφίων και δίπλα τους το μέσο όρο των βαθμών τους ταξινομημένα με βάση τον μέσο όρο κατά φθίνουσα σειρά. Σε περίπτωση ισοβαθμίας η σειρά ταξινόμησης των ονομάτων να είναι αλφαβητική.
ε. θα υπολογίζει και να εκτυπώνει το πλήθος των υποψηφίων με το μεγαλύτερο μέσο όρο.

24. Να γραφεί πρόγραμμα, το οποίο :
α. θα διαβάζει την πρωϊνή (8 π.μ.) θερμοκρασία της Λάρισας για όλες τις ημέρες του έτους 2004 (365 ημέρες). Θεωρείστε ότι οι θερμοκρασίες στην πόλη αυτή κυμαίνονται από -20 ^{o}C μέχρι 45 ^{o}C,
β. θα βρίσκει τη συχνότητα κάθε θερμοκρασίας για το έτος 2004 (δηλαδή πόσες φορές έχει καταγραφεί κάθε θερμοκρασία),
γ. θα εμφανίζει τη συχνότητα των θερμοκρασιών που έχουν καταγραφεί τουλάχιστον μία φορά.

25. Ένας τετραγωνικός ακέραιος πίνακας 10 x 10 περιέχει τον αριθμό 99. Να γραφεί πρόγραμμα, το οποίο θα βρίσκει και θα εμφανίζει τη θέση του πίνακα στην οποία βρίσκεται ο αριθμός 99 (δηλαδή τον αριθμό γραμμής και στήλης). Το πρόβλημα να λυθεί σε δύο περιπτώσεις :
α. η διαδικασία της αναζήτησης θα σταματά μόλις γίνει ο εντοπισμός του αριθμού,
β. θα βρίσκονται και θα εμφανίζονται όλες οι θέσεις του πίνακα στις οποίες υπάρχει ο αριθμός.

26. Κατά την απογραφή του έτους 2001 απογράφηκαν σε ένα χωριό 1200 άτομα. Να γραφεί πρόγραμμα, το οποίο :
α. θα καταχωρεί το έτος γέννησης κάθε κατοίκου σε έναν μονοδιάστατο πίνακα και το ονοματεπώνυμό του σε έναν δεύτερο πίνακα (με αντιστοιχία θέσεων),
β. θα υπολογίζει και θα εμφανίζει το πλήθος των ατόμων κατά κατηγορία ηλικίας με ανάλογο μήνυμα. Οι κατηγορίες ηλικίας είναι οι εξής :
⇒ 0 έως και 25 ετών
⇒ 26 έως και 45 ετών
⇒ 46 έως και 65 ετών
⇒ άνω των 65 ετών
γ. θα εμφανίζει τις ηλικίες των 3 γηραιότερων ατόμων,
δ. θα εμφανίζει τα ονόματα των 3 γηραιότερων ατόμων.

27. Να γραφεί πρόγραμμα, το οποίο :
α. θα διαβάζει τα ψηφία ενός πενταψήφιου αριθμού,
β. θα καταχωρεί τα ψηφία σε έναν πίνακα ακεραίων Α,
γ. θα κρυπτογραφεί τον αριθμό με τη μέθοδο της δεξιάς ολίσθησης κατά δύο ψηφία,
δ. θα εμφανίζει τον κρυπτογραφημένο αριθμό.
Κατά τη μέθοδο της δεξιάς ολίσθησης τα αριθμητικά ψηφία αντικαθίστανται ως εξής:
0 → 2, 1 → 3, 2 → 4, 3 → 5, 4 → 6, 5 → 7, 6 → 8, 7 → 9, 8 → 0, 9 → 1.

28. Μια ασφαλιστική εταιρεία έχει 50 ασφαλιστές. Κάθε ασφαλιστής παίρνει προμήθεια ανά τρίμηνο ανάλογα με το ποσό των ασφαλειών ως εξής :
⇒ 20% αν το ποσό των ασφαλειών είναι μεγαλύτερο ή ίσο με 10.000 €.
⇒ 10% αν το ποσό των ασφαλειών είναι μικρότερο από 10.000 €.
Να γραφεί πρόγραμμα, το οποίο :

α. θα διαβάζει τον κωδικό για κάθε ασφαλιστή και το ύψος των ασφαλειών που έκανε για κάθε μήνα του τριμήνου,
β. θα εμφανίζει το συνολικό ποσό των ασφαλειών του ασφαλιστή και την προμήθεια που αντιστοιχεί στο τρίμηνο,
γ. θα εμφανίζει το συνολικό ποσό των ασφαλειών και το συνολικό ποσό των προμηθειών του τριμήνου για όλους τους ασφαλιστές της εταιρείας.

29. Στον διαγωνισμό πρόσληψης προσωπικού μιας Τράπεζας έλαβαν μέρος 500 υποψήφιοι οι οποίοι εξετάστηκαν σε πέντε μαθήματα. Η βαθμολογία σε κάθε μάθημα ήταν ακέραια στο διάστημα 1 έως 50. Η απόφαση της Τράπεζας ήταν να προσληφθούν οι υποψήφιοι με μέσο όρο βαθμολογίας μεγαλύτερο από 40. Να γραφεί πρόγραμμα, το οποίο :
α. θα διαβάζει από το πληκτρολόγιο τους βαθμούς των διαγωνιζομένων σε κάθε μάθημα και θα τους καταχωρεί σε δισδιάστατο πίνακα,
β. θα υπολογίζει και θα εμφανίζει τον μέσο όρο βαθμολογίας κάθε διαγωνιζομένου,
γ. θα υπολογίζει και θα εμφανίζει το πλήθος των υποψηφίων που προσλαμβάνονται στην Τράπεζα.

30. Ένα ξενοδοχείο έχει 15 ορόφους των 30 δωματίων ο καθένας. Να γραφεί πρόγραμμα, το οποίο :
α. θα εισάγει σε έναν δισδιάστατο πίνακα 15 x 30 μια πληροφορία για κάθε δωμάτιο ως εξής : αν το δωμάτιο είναι κατελειμμένο το στοιχείο εισαγωγής θα είναι 1, διαφορετικά θα είναι 0,
β. θα υπολογίζει και θα εμφανίζει πόσα δωμάτια είναι ελεύθερα σε κάθε όροφο,
γ. θα υπολογίζει και θα εμφανίζει πόσα δωμάτια είναι ελεύθερα στο ξενοδοχείο,
δ. θα εμφανίζει τα δωμάτια του 10 ορόφου που είναι κατελειμμένα.

Βιβλιογραφία

1 Β. Kernigham, D. Ritchie, "*Η Γλώσσα Προγραμματισμού C*", Εκδόσεις Κλειδάριθμος, Αθήνα 1986

2 Σ. Ζάρκος, *"Δομές Δεδομένων και Αρχεία στην C"*, Αθήνα, 1993

3 Κ. Λάζος, *"C++: Θεωρία και πράξη"*, Θεσσαλονίκη, 2003

4 Ν. Μισυρλής, "*Εισαγωγή στον Προγραμματισμό με την C*", Τμήμα Πληροφορικής και Τηλεπικοινωνιών, Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών, Αθήνα 2003.

5 Β. Σεφερίδης, "*C για Αρχάριους*", Εκδόσεις Κλειδάριθμος, Αθήνα 1995, Ανατύπωση 2001

6 Aho, J. Hopcroft, J. Ullman, "*Data Structures And Algorithms*", Addison-Wesley Publishing Company, USA, 1985 (Reprinted with corrections)

7 L. Ammeraal, "*C for Programmers"*, 2nd Edition, John Willey & Sons, England, 1991

8 L. Atkinson, M. Atkinson, *"Using C"*, Que Corporation, Carmel, USA, 1990

9 H. M. Deitel, P. J. Deitel, "*C: How To Program*", Prentice-Hall, Inc., New Jersey, USA, 1998

10 W. Feibel, *"Using ANSI C in Unix"*, Osborne McGraw-Hill, USA, 1990

11 L. Hancock, M. Krieger, and S. Zamir, *"The C Primer"*, 3rd Edition, McGraw-Hill, New York, USA, 1991

12 W. Hunt, *"The C Toolbox"*, Addison-Wesley, 1989

13 Kelley, and I. Pohl, "*A Book on C, An Introduction To Programming In C*", The Benjamin/Cummings Publishing Company, Inc California, USA, 1984

14 Kelley, and I. Pohl, "*C by Dissection*", The Benjamin/Cummings Publishing Company, Inc California, USA, 1986

15 H. Schildt, "*C: The Complete Reference*", Fourth Edition, McGraw Hill, 2000

16 M. G. Sobell, "*A Practical Guide to the UNIX* System"*, The Benjamin/Cummings Publishing Company, Inc California, USA, 1984

17 Tenenbaum, Y. Langsam, M. Augenstein, "*Data Structures Using C*", Prentice Hall International Editions, USA, 1990

18 T. Zhang, *"Μάθετε τη C σε 24 Ώρες"*, Δεύτερη Έκδοση, Γκιούρδας, 2000

19 "*C/C++ Users Journal, Advanced Solutions For C/C++ Programmers*", Boulder Co., USA

20 *"Computer Language"*, Freeman Miller Publications, USA

21 *"The C Users Journal"*, R & D Publications, USA