ΠΡΟΧΕΙΡΕΣ ΣΗΜΕΙΩΣΕΙΣ ΣΤΗΝ ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ C

Χρήστος Αρβανίτης

Εισαγωγή

Στις σημειώσεις αυτές καταγράφεται το περιεχόμενο των διαλέξεων που δόθηκαν κατα το ακαδ. έτος 2008 στο Πανεπιστήμιο Κρήτης σχετικά με την την γλώσσα προγραμματισμού C, στα πλαίσια του μαθήματος "Γλώσσα Προγραμματισμού Η/Υ" του Τμήματος Εφαρμοσμένων Μαθηματικών.

ΠΑΡΑΤΗΡΗΣΗ: Οι σημειώσεις αυτές μεταβάλλονται με δυναμικό τρόπο κατα την διάρκεια του ακαδημαϊκού εξαμήνου. Αφενός επαυξάνονται με το υλικό των νέων διαλέξεων και αφετέρου μορφοποιούνται συνεχώς ώστε να περιέχουν τις ερωταπαντήσεις που σε κατοπινό χρόνο συζητώνται στο αμφιθέαρο ή τα εργαστήρια. Κάθε φορά που ο σπουδαστής ανακαλεί αυτό το έγγραφο απο την ιστοσελίδα του μαθήματος, πρέπει να κάνει μια σύντομη ανάγνωση σε όλη την έκταση του για να ενημερωθεί για τις πιθανές μεταβολές σε προηγούμενα τμήματα. Το μόνο τμήμα που θα παραμένει σταθερό μέχρι το τέλος των διαλέξεων, ειναι το περιεχόμενο αυτής της παρατήρησης...

ΣΥΣΤΗΜΑΤΑ ΑΝΑΠΑΡΑΣΤΑΣΗΣ ΑΡΙΘΜΩΝ

Για να γίνει πιο κατανοητό το περιεχόμενο των επόμενων κεφαλαίων, απαιτείται μια σύντομη επισκόπηση των **συστημάτων αναπαράστασης αριθμών**. Στην καθημερινότητα χρησιμοποιούμε τους φυσικούς αριθμούς για την αρίθμηση πραγμάτων και την εκτέλεση αριθμητικών πράξεων. Σε αυτές τις ενέργειες, πάντα υπεισέρχεται ένα βήμα που συνήθως περνάει απαρατήρητο. Προκειμένου να χρησιμοποιήσουμε έναν φυσικό αριθμό, τον μετατρέπουμε αρχικά απο "αφηρημένη οντότητα" σε κάτι συγκεκριμένο, αποδίδοντάς του μοναδικό όνομα και μοναδική μορφή με έναν εννιαίο τρόπο, όποιος και αν ήταν ο αριθμός. Πρόκειται για την **αναπαράσταση αριθμών στο δεκαδικό σύστημα**, ένα βήμα που εκτελείται μηχανικά γι' αυτό και συνήθως ταυτίζουμε τους αριθμούς με την δεκαδική αναπαράσταση τους (κάτι που ειναι λάθος). Σε αυτό το κεφάλαιο θα προσεγγίσουμε την αναπαράσταση αριθμών με πιό γενικό τρόπο, ώστε να κατανοήσουμε πώς "αντιλαμβάνεται" τους αριθμούς ένας Η/Υ.

Ως **k-δική αναπαράσταση ενός φυσικού αριθμού** x (όπου k ένας φυσικός > 1) εννοούμε μια διάταξη *ψηφίων* $x_m x_{m-1} \dots x_0$ με τις εξής ιδιότητες:

. Τα ψηφία $x_m, x_{m-1}, \dots, x_0$ επιλέγονται όλα απο ένα δοσμένο σύνολο συμβόλων $\{\Psi_0, \Psi_1, \dots, \Psi_{k-1}\}$ που εκπροσωπούν ακριβώς τους φυσικούς αριθμούς $\{0, 1, 2, \dots, k-1\}$. Ειδικά ο φυσικός μηδέν εκπροσωπείται πάντα με το σύμβολο 0, (δηλαδή σε κάθε σύστημα αναπαράστασης αριθμών, ώς Ψ_0 επιλέγεται το 0).

. Το άθροισμα $x_m \cdot k^m + x_{m-1} \cdot k^{m-1} + \dots + x_0 \cdot k^0$ ισούται αριθμητικά με τον x, δηλαδή $\sum_{i=0}^{m} x_i \cdot k^i = x$

Ουσιαστικά πρόκειται για κωδικοποίηση όπου καταγράφουμε με φθίνουσα σειρά μόνο τους συντελεστές των δυνάμεων του k που απαιτούνται για να παραχθεί ο x σαν άθροισμά τους (παραλείποντας όλα τα άλλα σύμβολα). Η πρώτη απαίτηση εξασφαλίζει την ύπαρξη και μοναδικότητα της k-δικής αναπαράστασης οποιουδήποτε φυσικού αριθμού (όταν το αριστερότερο ψηφίο x_m είναι $\neq 0$).

ΟΡΟΛΟΓΙΑ: Το k θα αναφέρεται ώς **βάση** και το σύνολο $\{\Psi_0, \Psi_1, \dots, \Psi_{k-1}\}$ ώς **σύνολο[1] επιλογής ψηφίων ($\Sigma E \Psi_k$)**, αυτού του συστήματος αναπαράστασης αριθμών.
Άν $x_m x_{m-1} \dots x_1 x_0$ ειναι η k-δική αναπαράσταση ενός φυσικού x τότε, το x_m θα αναφέρεται ώς το **πλέον σημαντικό ψηφίο** ενώ το x_0 ώς το **λιγότερο σημαντικό ψηφίο** αυτής της αναπαράστασης. Για να ξεχωρίζουμε τις αναπαραστάσεις ενός αριθμού σε διαφορετικά συστήματα, θα σημειώνουμε την βάση αναπαράστασης σαν κάτω-δεξιό δείκτη με εξαίρεση την δεκαδική αναπαράσταση όπου δεν θα βάζουμε δείκτη.

ΠΑΡΑΔΕΙΓΜΑΤΑ

i) Η βάση του δεκα-δικού συστήματος αναπαράστασης αριθμών, ειναι το $k = 10$ και ώς σύνολο επιλογής ψηφίων έχει καθιερωθεί να ειναι το $\Sigma E \Psi_{10} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Η δεκα-δική αναπαράσταση π.χ. του αριθμού 5^3 ειναι 125 αφού:

. Τα ψηφία $1, 2, 5$ ανήκουν όλα στο σύνολο επιλογής $\Sigma E \Psi_{10}$.

. Το άθροισμα $1 \cdot 10^2 + 2 \cdot 10^1 + 5 \cdot 10^0$ ισούται αριθμητικά με τον αριθμό 5^3 !
ΑΣΚΗΣΗ: Προσπαθήστε να δείξετε ότι το παραπάνω άθροισμα ισούται αριθμητικά με τον 5^3 χωρίς να χρησιμοποιήσετε την δεκαδική μορφή του. Αυτό ειναι εφικτό μόνο άν *αποταυτίσετε τους αριθμούς απο την δεκαδική μορφή τους!*

ii) Η βάση του οκτα-δικού συστήματος αναπαράστασης αριθμών, ειναι το $k = 8$ και ώς σύνολο επιλογής ψηφίων έχει καθιερωθεί να ειναι το $\Sigma E \Psi_8 = \{0, 1, 2, 3, 4, 5, 6, 7\}$. Η οκτα-δική αναπαράσταση του αριθμού 5^3 ειναι 175_8 αφού:

. Τα ψηφία $1, 7, 5$ ανήκουν όλα στο σύνολο επιλογής $\Sigma E \Psi_8$.

. Το άθροισμα $1 \cdot 8^2 + 7 \cdot 8^1 + 5 \cdot 8^0 = 64 + 56 + 5 = 125$ που "είδαμε πριν" ότι ισούται με τον αριθμό 5^3.

[1] Το $\Sigma E \Psi_k$ αναφέρεται ώς "σύνολο" επιλογής ψηφίων μόνο για λόγους γλωσσικής απλούστευσης. Η σειρά εμφάνισης των στοιχείων του $\Psi_0, \Psi_1, \dots \Psi_{k-1}$ ειναι καθοριστική για την εύρεση του φυσικού που το καθένα απο αυτά εκπροσωπεί. Ουσιαστικά δηλαδή το $\Sigma E \Psi_k$ ειναι μια διάταξη και όχι ένα απλό σύνολο.

iii) Η βάση του δυα-δικού συστήματος αναπαράστασης αριθμών, ειναι το $k = 2$ και ώς σύνολο επιλογής ψηφίων έχει καθιερωθεί να ειναι το $\Sigma E \Psi_2 = \{0, 1\}$. Η δυα-δική αναπαράσταση του αριθμού 5^3 ειναι 1111101_2 αφού:
. Τα ψηφία $1, 1, 1, 1, 1, 0, 1$ ανήκουν στο σύνολο επιλογής $\Sigma E \Psi_2$.
. Το άθροισμα $1 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 64 + 32 + 16 + 8 + 4 + 1 = 125$ που "είδαμε πριν" ότι ισούται με τον αριθμό 5^3.

iv) Η βάση του τετρα-δικού συστήματος αναπαράστασης αριθμών, ειναι το $k = 4$ και ώς σύνολο επιλογής ψηφίων έχει καθιερωθεί να ειναι το $\Sigma E \Psi_4 = \{0, 1, 2, 3\}$. Το κείμενο 160_4 δεν ειναι αναπαράσταση κάποιου αριθμού στο τετρα-δικό σύστημα αφού χρησιμοποιούνται ψηφία εκτός του συνόλου επιλογής $\Sigma E \Psi_4$.

ΠΑΡΑΤΗΡΗΣΕΙΣ

i) Γενικά η προσέγγιση που θα ακολουθούμε για να επαληθεύσουμε μιά k-δική αναπαράσταση αριθμού θα ειναι όπως στα παραδείγματα ii) και iii). Δηλαδή, πρώτα θα σχηματίζουμε το άθροισμα των σχετικών δυνάμεων, και έπειτα θα αντικαθιστούμε τα ψηφία με τον ακέραιο που εκπροσωπούν χρησιμοποιώντας όμως δεκαδική αναπαράσταση αριθμών όπου έχουμε πιό "ξεκάθαρη άποψη"(το παράδειγμα i) δείχνει ότι βρίσκουμε όλοι το ίδιο αποτέλεσμα λόγω της συνέπειας των αριθμητικών πράξεων και όχι επειδή γνωρίζουμε την πραγματική φύση των αριθμών).
ΠΑΡΑΔΕΙΓΜΑ: Οι 403_8 και 10003_4 ειναι αναπαραστάσεις του ίδιου αριθμού αφού, βρίσκοντας τις αντίστοιχες δεκαδικές αναπαράστασεις τους
$403_8 = 4 \cdot 8^2 + 0 \cdot 8^1 + 3 \cdot 8^0 = 256 + 3 = 259$ και $10003_4 = 1 \cdot 4^4 + 0 \cdot 4^3 + 0 \cdot 4^2 + 0 \cdot 4^1 + 3 \cdot 4^0 = 256 + 3 = 259$, μπορούμε να αποφανθούμε ότι και οι δύο παριστάνουν τον ίδιο αριθμό.

ii) Όπως φαίνεται στα παραδείγματα, για $k \leq 10$ έχει καθιερωθεί οι k-αδικές αναπαραστάσεις να έχουν σύνολο επιλογής ψηφίων το $\Sigma E \Psi_k = \{0, 1, .., k - 1\}$, αυτό όμως δεν ειναι εφικτό για $k > 10$.
ΠΑΡΑΔΕΙΓΜΑ: Η βάση του δεκαεξα-δικού συστήματος αναπαράστασης αριθμών, ειναι το $k = 16$. Άν καθιερώναμε ώς σύνολο επιλογής ψηφίων το $\Sigma E \Psi_{16} = \{0, 1, \ldots, 14, 15\}$, τότε η αναπαράσταση 15_{16} θα εκπροσωπούσε τον $15 \cdot 16^0 = 15$ ή τον αριθμό $1 \cdot 16^1 + 5 \cdot 16^0 = 21$;
Το πρόβλημα προέκυψε επειδή κάποια απο τα ψηφία του συνόλου επιλογής περιγράφονται με σύμβολο που περιέχεται σε σύμβολο άλλου ψηφίου, με αποτέλεσμα όταν τοποθετούνται σε μία σειρά χωρίς διαχωριστικά, να τίθεται το δίλημμα με ποιά δύναμη σχετίζονται.
Ορίζοντας για σύνολο επιλογής ψηφίων το $\Sigma E \Psi_{16} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, \mathrm{a}, \mathrm{b}, \mathrm{c}, \mathrm{d}, \mathrm{e}, \mathrm{f}\}$ (όπου ώς ψηφία χρησιμοποιούνται διαφορετικά σύμβολα) τότε η αναπαράσταση 15_{16} ειναι φανερό ότι απεικονίζει τον αριθμό $1 \cdot 16^1 + 5 \cdot 16^0 = 21$ ενώ ο 15 θα δινόνταν απο την αναπαράσταση f_{16} αφού το $\mathrm{f}_{16} \cdot 16^0$ ισούται αριθμητικά με το $15 \cdot 16^0 = 15$.

ΙΔΙΟΤΗΤΕΣ: Άν $(x_m x_{m-1} \ldots x_0)_k$ ειναι η k-δική αναπαράσταση ενός φυσικού x, τότε, λαμβάνοντας υπόψιν ότι στο $\Sigma E \Psi_k$ εκπροσωπούνται ακριβώς οι ακέραιοι $0, 1, \ldots, k - 1$, συμπεραίνουμε ότι:

i) Η k-δική αναπαράσταση του φυσικού $x \cdot k$ ειναι $(x_m x_{m-1} \ldots x_0 0)_k$ γιατί
$$\begin{aligned} x \cdot k &= (x_m x_{m-1} \ldots x_0)_k \cdot k = (x_m \cdot k^m + x_{m-1} \cdot k^{m-1} + \cdots + x_1 \cdot k^1 + x_0 \cdot k^0) \cdot k \\ &= x_m \cdot k^{m+1} + x_{m-1} \cdot k^m + \cdots + x_0 \cdot k^1 = x_m \cdot k^{m+1} + x_{m-1} \cdot k^m + \cdots + x_0 \cdot k^1 + 0 \cdot k^0 \\ &= (x_m x_{m-1} \ldots x_0 0)_k. \end{aligned}$$

ii) Η k-δική αναπαράσταση του φυσικού $[x/k]$ (πρόκειται για το πηλίκο της Ευκλείδιας διαίρεσης, δηλαδή το ακέραιο μέρος της διαίρεσης x/k) ειναι $(x_m x_{m-1} \ldots x_1)_k$ γιατί
$$\begin{aligned} [x/k] &= [(x_m x_{m-1} \ldots x_0)_k / k] = [(x_m \cdot k^m + x_{m-1} \cdot k^{m-1} + \cdots + x_1 \cdot k^1 + x_0 \cdot k^0)/k] \\ &= [x_m \cdot k^{m-1} + x_{m-1} \cdot k^{m-2} + \cdots + x_1 \cdot k^0 + x_0/k] \\ &= x_m \cdot k^{m-1} + x_{m-1} \cdot k^{m-2} + \cdots + x_1 \cdot k^0 + [x_0/k] \\ &= x_m \cdot k^{m-1} + x_{m-1} \cdot k^{m-2} + \cdots + x_1 \cdot k^0 \qquad (\text{επειδή } x_0 \in \Sigma E \Psi_k \text{ έπεται ότι } [x_0/k] = 0) \\ &= (x_m x_{m-1} \ldots x_1)_k. \end{aligned}$$

iii) Η k-δική αναπαράσταση του φυσικού $x \mod k$ (προκειται για το υπόλοιπο της Ευκλείδιας διαίρεσης x/k, δηλαδή τον φυσικό $x - [x/k]k$) ειναι $(x_0)_k$ γιατί
$$\begin{aligned} x \mod k &= x - [x/k]k = (x_m x_{m-1} \ldots x_0)_k - (x_m x_{m-1} \ldots x_1)_k \cdot k \qquad (\text{ απο ιδιότητα ii}) \\ &= (x_m x_{m-1} \ldots x_1 x_0)_k - (x_m x_{m-1} \ldots x_1 0)_k \qquad (\text{ απο ιδιότητα i}) \\ &= (00 \ldots 0 x_0)_k = (x_0)_k. \end{aligned}$$

ΕΦΑΡΜΟΓΕΣ

i) Με επανάληψη των ιδιοτήτων iii), ii) μπορούμε να προσδιορίσουμε όλα τα ψηφία $x_m, x_{m-1}, \ldots, x_0$ της k-δικής αναπαράστασης ενός φυσικού x, ξεκινώντας απο το λιγότερο σημαντικό και καταλήγοντας στο πλέον σημαντικό.

ΠΑΡΑΔΕΙΓΜΑ : Να βρεθεί η αναπαράσταση του φυσικού 67 στο δυαδικό, οκταδικό και δεκαεξαδικό σύστημα.

Άν η δυαδική αναπαράσταση του 67 ειναι $(b_m b_{m-1} \ldots b_0)_2$, τότε απο τις ιδιότητες ξέρουμε ότι

$b_0 = (67 \mod 2) = 1$ και ότι η δυαδική αναπαράσταση του $[67/2] = 33$ ειναι $(b_m b_{m-1} \ldots b_1)_2$ οπότε
$b_1 = (33 \mod 2) = 1$ και ότι η δυαδική αναπαράσταση του $[33/2] = 16$ ειναι $(b_m b_{m-1} \ldots b_2)_2$ οπότε
$b_2 = (16 \mod 2) = 0$ και ότι η δυαδική αναπαράσταση του $[16/2] = 8$ ειναι $(b_m b_{m-1} \ldots b_3)_2$ οπότε
$b_3 = (8 \mod 2) = 0$ και ότι η δυαδική αναπαράσταση του $[8/2] = 4$ ειναι $(b_m b_{m-1} \ldots b_4)_2$ οπότε
$b_4 = (4 \mod 2) = 0$ και ότι η δυαδική αναπαράσταση του $[4/2] = 2$ ειναι $(b_m b_{m-1} \ldots b_5)_2$ οπότε
$b_5 = (2 \mod 2) = 0$ και ότι η δυαδική αναπαράσταση του $[2/2] = 1$ ειναι $(b_m b_{m-1} \ldots b_6)_2$ οπότε
$b_6 = (1 \mod 2) = 1$ και ότι η δυαδική αναπαράσταση του $[1/2] = 0$ ειναι $(b_m b_{m-1} \ldots b_7)_2$

εδώ θα σταματήσουμε γιατί τα υπόλοιπα ψηφία θα ειναι πάντα το 0, οπότε $67 = 1000011_2$.

Κρατώντας μόνο τις στήλες των αριθμών που προκύπτουν σε κάθε βήμα, η παραπάνω διαδικασία περιγράφεται απλούστερα με τον πίνακα

Αριθμός	Αριθμός mod 2	[Αριθμός / 2]
67	1	33
33	1	16
16	0	8
8	0	4
4	0	2
2	0	1
1	1	0

Αντίστοιχη διαδικασία για το οκταδικό και δεκαεξαδικό σύστημα οδηγεί στους πίνακες:

Αριθμός	Αριθμός mod 8	[Αριθμός / 8]
67	3	8
8	0	1
1	1	0

,

Αριθμός	Αριθμός mod 16	[Αριθμός / 16]
67	3	4
4	4	0

οπότε συμπεραίνουμε ότι οι αντίστοιχες αναπαραστάσεις του 67 ειναι $103_8, 43_{16}$.

ii) Πόσα k-αδικά ψηφία απαιτούνται για την αναπαράσταση ενός δοσμένου φυσικού ;
Την απάντηση την παίρνουμε άμεσα απο την παραπάνω διαδικασία, δηλαδή, όσες επαναλήψεις απαιτούνται μέχρι να παραχθεί το 0 στην δεξιά στήλη του πίνακα, δηλαδή $1 + [\log_k x]$.

ΓΕΝΙΚΕΥΣΗ ΓΙΑ ΟΠΟΙΟΝΔΗΠΟΤΕ ΘΕΤΙΚΟ ΑΡΙΘΜΟ

Στην γενική περίπτωση όπου ένας θετικός αριθμός x δεν ειναι φυσικός, μιά διάταξη της μορφής $(x_m x_{m-1} \ldots x_0 \bullet x_{-1} \ldots x_{-n})_k$ θεωρείται ώς η k-αδική αναπαράστασή του άν και μόνο:

. Τα ψηφία $x_m, x_{m-1}, \ldots, x_0, x_{-1}, \ldots, x_{-n}$ επιλέγονται όλα απο το $\Sigma E \Psi_k$

. Το άθροισμα $x_m \cdot k^m + x_{m-1} \cdot k^{m-1} + \cdots + x_1 \cdot k^1 + x_0 \cdot k^0 + x_{-1} \cdot k^{-1} + \cdots + x_{-n} \cdot k^{-n}$ ισούται αριθμητικά με τον x, δηλαδή $\sum_{i=-n}^{m} x_i \cdot k^i = x$.

Η τελεία (υποδιαστολή) στην παραπάνω διάταξη ψηφίων, ειναι απαραίτητη γιατί δείχνει το ψηφίο όπου εμφανίζονται οι αρνητικές δυνάμεις στο αντίστοιχο άθροισμα.

Για τέτοιες αναπαραστάσεις μπορούμε να δούμε ότι ισχύει η δεύτερη ιδιότητα στην μορφή:

ii') Η k-δική αναπαράσταση του αριθμού x/k (όχι ακέραιο μέρος) είναι ίδια με του αριθμού x εκτός απο την υποδιαστολή που ειναι μετατοπισμένη μιά θέση αριστερά.

Χρησιμοποιώντας αυτήν την ιδιότητα μπορούμε να προσδιορίσουμε την k-αδική αναπαράστασή ενός αριθμού x ακόμα και άν δεν ειναι φυσικός. Πράγματι, άν αρχικά πολλαπλασιάσουμε τον x με κατάλληλη δύναμη της βάσης k^n ώστε ο $\tilde{x} = x \cdot k^n$ να είναι φυσικός και στην συνέχεια προσδιορίσουμε με την διαδικασία της πρώτης

εφαρμογής την k-αδική αναπαράστασή $(\tilde{x}_m\tilde{x}_{m-1}\dots\tilde{x}_0)_k$ του $\tilde{x}$, τότε θα ξέρουμε απο την ιδιότητα ii') ότι η k-δική αναπαράστασή του αριθμού x ειναι $(\tilde{x}_m\tilde{x}_{m-1}\dots\tilde{x}_n\ .\ \tilde{x}_{n-1}\dots\tilde{x}_0)_k$.

ΠΑΡΑΔΕΙΓΜΑ : Να βρεθεί η αναπαράσταση του 67.125 στο δυαδικό, οκταδικό και δεκαεξαδικό σύστημα.
Παρατηρούμε ότι $67.125 \cdot 8 = 537$ δηλαδή ότι το 2^3 πολλαπλάσιό του ειναι ακέραιος, οπότε αφού βρούμε την δυαδική αναπαράσταση αυτού

Αριθμός	Αριθμός mod 2	[Αριθμός / 2]
537	1	268
268	0	134
134	0	67
67	1	33
33	1	16
16	0	8
8	0	4
4	0	2
2	0	1
1	1	0

δηλαδή ότι $537 = 1000011001_2$, μπορούμε να συμπεράνουμε ότι $67.125 = 1000011.001_2$.
Ομοίως για το οκταδικό σύστημα, παρατηρούμε ότι $67.125 \cdot 8 = 537$ δηλαδή ότι το 8^1 πολλαπλάσιό του ειναι ακέραιος, οπότε αφού βρούμε την οκταδική αναπαράσταση αυτού

Αριθμός	Αριθμός mod 8	[Αριθμός / 8]
537	1	67
67	3	8
8	0	1
1	1	0

δηλαδή ότι $537 = 1031_8$, μπορούμε να συμπεράνουμε ότι $67.125 = 103.1_8$.
Τέλος για το δεκαεξαδικο σύστημα, παρατηρούμε ότι $67.125 \cdot 16 = 1074$ δηλαδή ότι το 16^1 πολλαπλάσιό του ειναι ακέραιος, οπότε αφού βρούμε την δεκαεξαδική αναπαράσταση αυτού

Αριθμός	Αριθμός mod 16	[Αριθμός / 16]
1074	2	67
67	3	4
4	4	0

δηλαδή ότι $1074 = 432_{16}$, μπορούμε να συμπεράνουμε ότι $67.125 = 43.2_{16}$.

ΑΝΑΠΑΡΑΣΤΑΣΕΙΣ ΑΡΝΗΤΙΚΩΝ ΑΡΙΘΜΩΝ ΣΕ ΣΥΣΤΗΜΑΤΑ ΠΕΠΕΡΑΣΜΕΝΟΥ ΠΛΗΘΟΥΣ ΨΗΦΙΩΝ

Γενικά για την αναπαράσταση αρνητικού αριθμού ισχύουν τα παραπάνω, αλλά, εφαρμοσμένα στην απόλυτη τιμή του και τοποθετώντας στο τέλος αρνητικό πρόσημο. Για παράδειγμα η αναπαράσταση του φυσικού -67 στο δυαδικό, οκταδικό και δεκαεξαδικό σύστημα ειναι $-1000011_2, -103_8$ και -43_{16} αντίστοιχα.

Ωστόσο στους Η/Υ τα ψηφία αναπαράστασης ειναι πεπερασμένου πλήθους m οπότε δημιουργείται μια περιοδικότητα στην εμφάνιση των αναπαράστασεων. Για παράδειγμα σε σύστημα όπου κρατώνται μόνο τα m πρώτα ψηφία, η αναπαράσταση $00\dots0_k$ εμφανίζεται για οποιονδήποτε αριθμό της μορφής $\lambda \cdot k^m,\ \ \lambda = 0, 1, ...$ Παρόμοια η k-δική αναπαράσταση $(x_{m-1}x_{m-2}\dots x_0)_k$ ενός αριθμού $x \in [0,\ k^m - 1]$, εμφανίζεται περιοδικά για κάθε αριθμό της μορφής $x + \lambda \cdot k^m,\ \ \lambda = 0, 1, ...$ Αυτή η περιοδικότητα που εμφανίζεται σε συστήματα πεπερασμένου πλήθους ψηφίων m, χρησιμοποιείται για να ορισθεί η k-δική αναπαράσταση αρνητικού αριθμού. Για παράδειγμα, λόγω περιοδικότητος ο -1 θεωρείται ότι έχει την ίδια αναπαράσταση με τον $-1 + k^m$.

Φυσικά με ένα πεπερασμένο πλήθος m ψηφίων αναπαράστασης μόνο k^m διαφορετικοί αριθμοί μπορούν να απεικονισθούν. Όταν χρησιμοποιούμε αυτό το σύστημα για απεικόνιση φυσικών αριθμών, μόνο οι αριθμοί $0, 1, ..., (k^m - 1)$ μπορούν να απεικονισθούν ενώ, για μεγαλύτερους αριθμούς x απεικονίζεται μόνο το τμήμα $(x \mod k^m)$ (υπόλοιπο της διαίρεσης του x με τον k^m) γιατί τα μεγαλύτερα ψηφία χάνονται. Όταν χρησιμοποιούμε αυτό το σύστημα για απεικόνιση ακέραιων αριθμών, τότε οι μισές περίπου απο τις αναπαραστάσεις αντιστοιχούν σε αριθμούς θετικού προσήμου ενώ οι υπόλοιπες σε αριθμούς αρνητικού προσήμου. Πιό συγκεκριμένα απεικονίζονται οι αριθμοί $0, \pm1, ..., \pm[\frac{k^m-1}{2}], -[\frac{k^m}{2}]$. Γενικότερα, **σε σύστημα πεπερασμένου πλήθους ψηφίων**

m**, ένας αρνητικός αριθμός** $x \in [-[\frac{k^m}{2}],\ 0]$ **παριστάνεται απο την** k**-δική αναπαράσταση του φυσικού αριθμού** $x + k^m$.

Στα παρακάτω αναφερόμαστε μόνο σε συστήματα δυαδικής αναπαράσταση ($k = 2$) πεπερασμένου αριθμού ψηφίων m, γιατί αυτά χρησιμοποιούνται ουσιαστικά στους Η/Υ. Σύμφωνα με τα προηγούμενα, σε αυτά τα συστήματα απεικονίζονται ακριβώς οι ακέραιοι $0, \pm 1, ..., \pm(2^{m-1} - 1), -2^{m-1}$.

Μπορούμε να μελετήσουμε την αναπαράσταση αριθμών σε ένα σύστημα 8 δυαδικών ψηφίων με έναν πίνακα σαν τον παρακάτω

00000000	00000001	00000010	...	01111111	10000000	10000001	...	11111110	11111111
0	1	2	...	127	128	129	...	254	255
0	1	2	...	127	ΕΔΩ ;	-127	...	-2	-1

Αυτός ο πίνακας περιέχει τόσες στήλες όσες είναι οι δυαδικές αναπαραστάσεις που μπορεί να δώσει το συγκεκριμένο σύστημα ($2^8 = 256$ σε πλήθος). Στην πρώτη γραμμή του καταγράφονται οι πιθανές δυαδικές αναπαραστάσεις. Στην δεύτερη γραμμή καταγράφονται οι αντίστοιχοι φυσικοί στο δεκαδικό σύστημα αναπαράστασης. Στην τρίτη γραμμή καταγράφονται στο δεκαδικό σύστημα αναπαράστασης, οι αντίστοιχοι ακέραιοι που προκύπτουν εφαρμόζοντας τον τρόπο αντιστοίχισης απο την περιοδικότητα. Η γραμμή αυτή πρέπει να συμπληρωθεί χρησιμοποιώντας την περιοδικότητα, δηλαδή απο τις ακραίες στήλες προς το μέσο.

Αρχικά τοποθετείται το 0, φροντίζουμε να αναπαρίσταται ΠΑΝΤΑ με ακολουθία ψηφίων της μορφής $00...0$ σε οποιοδήποτε k-δικό σύστημα με ή χωρίς δεκαδικά ψηφία.

Έπειτα τοποθετείται το 1, οπότε μπορούμε να αποφανθούμε ότι το -1 παριστάνεται απο την δυαδική αναπαράσταση του αριθμού 256-1=255.

Έπειτα τοποθετείται το 2, οπότε μπορούμε να αποφανθούμε ότι το -2 παριστάνεται απο την δυαδική αναπαράσταση του αριθμού 256-2=254.

... (συνεχίζοντας με αυτόν τον τρόπο) ...

Έπειτα τοποθετείται το 127, οπότε μπορούμε να αποφανθούμε ότι το -127 παριστάνεται απο την δυαδική αναπαράσταση του αριθμού 256-127=129.

Στην τελευταία κενή στήλη, αυτήν του 10000000, τίθεται ένα δίλημμα. Ποιόν ακέραιο απεικονίζει αυτή η αναπαράσταση, αφού και οι δύο επιλογές 128 και -128 οδηγούν σε αυτήν την στήλη ; Η συνήθης επιλογή ειναι -128 επειδή στα προηγούμενα βήματα όσοι αριθμοί αντιστοιχήθηκαν σε δυαδικό αριθμό με πλέον σημαντικό ψηφό 1, ήταν αρνητικοί. **Σε αυτές τις αναπαραστάσεις το πλέον σημαντικό ψηφίο απεικονίζει το πρόσημο, ειναι** 1 **μόνο για τους αρνητικούς αριθμούς.**

ΠΑΡΑΤΗΡΗΣΗ: Μιά άλλη πιθανή επιλογή που να μην αντιβαίνει στο ψηφίο του προσήμου, ειναι να θεωρηθεί ότι η αναπαράσταση 10000000 εικονίζει το -0. Αυτό το σύμβολο δεν πρέπει να θεωρηθεί ότι ταυτίζεται με το 0 αλλά με την τιμή της απροσδιόριστης μορφής $+\infty - \infty$, η οποία δεν ειναι εκ των προτέρων γνωστή. Ισως να μην ορίζεται, ή να ειναι κάποιος πραγματικός αριθμός, ή να ειναι κάποιο απο τα σύμβολα $-\infty, +\infty$, κάθε περίπτωση χρειάζεται ειδικότερη αντιμετώπιση.

ΠΑΡΑΔΕΙΓΜΑΤΑ :

i Να βρεθεί η δυαδική αναπαράσταση του -12 σε σύστημα αναπαράστασης 8 και 16 ψηφίων.
m=8) Λόγω περιοδικότητος, σε αυτό το σύστημα, ο -12 θα έχει την ίδια αναπαράσταση με τον $-12 + 2^8 = 256 - 12 = 244$, δηλαδή ο -12 αναπαρίσταται στην μορφή 11110100_2.
m=16) Λόγω περιοδικότητος, σε αυτό το σύστημα, ο -12 θα έχει την ίδια αναπαράσταση με τον $-12+2^{16} = 65536 - 12 = 65524$, δηλαδή ο -12 αναπαρίσταται στην μορφή $11111111\ 11110100_2$.

ii Να βρεθεί η δυαδική αναπαράσταση του 340 σε σύστημα αναπαράστασης 8 και 16 ψηφίων.
Επειδή $340 = 101010100_2$ έπεται ότι σ' ενα σύστημα 8 ψηφίων θα εμφανισθεί μόνο το τμήμα 01010100_2 δηλαδή ο αριθμός 84 ($= 340 \mod 2^8$). Σ' ενα σύστημα 16 ψηφίων θα εμφανισθεί ολόκληρος ο αριθμός στην μορφή $00000001\ 01010100_2$.

ΑΣΚΗΣΗ: Σε ένα σύστημα δυαδικής αναπαράστασης m ψηφίων, είδαμε ότι όταν $x \in \{0, 1, \ldots, 2^{m-1} - 1\}$ τότε ο $-x$ παριστάνεται απο την δυαδική αναπαράσταση του $2^m - x$. Δεδομένου ότι η αναπαράσταση του $[\frac{x}{2}]$ ειναι γνωστό ποιά θα ειναι (λόγω της ιδιότητος ii)), ποιά θα ειναι η αναπαράσταση του $[\frac{-x}{2}]$;

ΔΟΜΙΚΑ ΣΤΟΙΧΕΙΑ ΠΗΓΑΙΟΥ ΚΩΔΙΚΑ ΓΛΩΣΣΑΣ C

1) Ο **πηγαίος κώδικας** ειναι κείμενο με λέξεις απο την γλώσσα C, αποθηκευμένο σε αρχείο του συστήματος με όνομα "*.c". Αυτό το αρχείο μεταφράζεται απο τον **μεταγλωττιστή** της γλώσσας C σε **αντικειμενικό κώδικα** (ακολουθία εντολών που μπορούν να εκτελεστούν απο την κεντρική μονάδα επεξεργασίας του Η/Υ) και αποθηκεύεται σε κάποιο εκτελέσιμο αρχείο. Η εκτέλεση των εντολών του αντικειμενικού κώδικα γίνεται

τυπώνοντας το όνομα του εκτελέσιμου αρχείου, στην γραμμή εντολών μιάς κονσόλας κελύφους.

2) Για να μεταγλωττιστεί ο πηγαίος κώδικας που βρίσκεται στο αρχείο name.c έτσι ώστε ο αντικειμενικός κώδικας που παράγεται να αποθηκευτεί στο αρχείο name, καλούμε τον μεταγλωττιστή απο μιά κονσόλα με εντολή της μορφής:

gcc option$_1$ option$_2$ ... option$_k$ name.c -o name -lib$_1$ -lib$_2$ ... -lib$_k$

όπου, τα options καθορίζουν τις παραλλαγές των εργασιών του μεταγλωττιστή (με "man gcc"μπορούμε να πάρουμε την αναλυτική λειτουργία του gcc, περίπου 11000 γραμμές επεξηγήσεις...), ενώ, τα libs ειναι βιβλιοθήκες απο αντικειμενικούς κώδικες που πρέπει να ενσωματωθούν στον κώδικα που θα παραχθεί.
Στις δικές μας εφαρμογές χρειαζόμαστε μόνο την βιβλιοθήκη μαθηματικών συναρτήσεων, οπότε η μεταγλώττιση θα γίνεται με την εντολή:

gcc name.c -o name -lm

Η διαδικασία της μεταγλώττισης γίνεται εσωτερικά σε δύο φασεις, την φάση της **προμεταγλώττισης** και την φάση της **κυρίως μεταγλώττισης**. Κατά την προμεταγλώττιση, μέσω κατάλληλων οδηγιών είτε εισάγεται κείμενο απο αρχείο άλλου πηγαίου κώδικα (*ενσωμάτωση πηγαίου κώδικα*), είτε τμήματα του κειμένου αντικαθίσταται με άλλο κείμενο (*μακροαντικατάσταση κειμένου*), είτε τμήματα του κειμένου διαγράφονται (*σχόλια*). Το τελικό κείμενο που προκύπτει απο την πρώτη φάση χρησιμοποιείται στην κυρίως μεταγλώττιση για να παραχθεί ο αντικειμενικός κώδικας.

3) Βασική δομή ενός πηγαίου κώδικα C: Αποτελείται απο το **ενημερωτικό τμήμα** που περιέχει τις οδηγίες για τον προμεταγλωττιστή και το **κυρίως τμήμα** που περιέχει τις εντολές γλώσσας C που θα μεταγλωττιστούν. Το κυρίως τμήμα αποτελείται απο την **περιοχή ορισμού μεταβλητών** και την **περιοχή επεξεργασίας μεταβλητών**. Η περιοχή επεξεργασίας αποτελείται απο **blocks**

Ώς block εντολών θα εννοούμε οποιαδήποτε ακολουθία εντολών της C που περικλείεται απο άγκιστρα "{", "}".

Όπως θα δουμε λεπτομερέστερα παρακάτω, ένα block ίσως έχει όνομα που προηγείται απο το αριστερό άγκιστρο "{", δική του περιοχή ορισμού μεταβλητών, καθώς και εσωτερικά blocks. Ένα βασικό block εντολών που ΠΑΝΤΑ θα συναντάμε έχει το όνομα

int main(int argc, char **argv)

Στις αρχικές εφαρμογές θα χρησιμοποιούμε γι' αυτό το block, το απλούστερο όνομα **main( )** .

4) Το ενημερωτικό τμήμα περιέχει **οδηγίες** που ελέγχουν την φάση της προμεταγλώττισης.

Στην φάση αυτή, ο προμεταγλωττιστής σαρώνει τον πηγαίο κώδικα γραμμή προς γραμμή απο την πρώτη προς την τελευταία. Όταν κατα την σάρωση αυτή, συναντηθεί γραμμή κειμένου με πρώτο χαρακτήρα την δίεση "#", τότε αυτή η γραμμή ερμηνεύεται σαν κάποια απο τις οδηγίες μετατροπής κειμένου, όποτε στην συνέχεια εφαρμόζεται σε όλο το υπολοιπόμενο κείμενο του πηγαίου κώδικα. Έπειτα η συγκεκριμένη γραμμή διαγράφεται και η σάρωση προχωράει στην επόμενη γραμμή. Εξαίρεση αποτελούν τα σχόλια. Οποιοδήποτε κείμενο του πηγαίου κώδικα περιέχεται ανάμεσα στα σύμβολα "/*" και "*/", θεωρείται σχόλιο και διαγράφεται κατά την σάρωση στην προμεταγλώττιση.

Υπάρχουν πολλές οδηγίες μετατροπής κειμένου προς τον προμεταγλωττιστή, εμείς θα χρησιμοποιήσουμε το πολύ τις παρακάτω:

α) Η οδηγία #**define**. Στην απλή της μορφή έχει την σύνταξη:

#**define ΛΕΞΗ ΑΚΟΛΟΥΘΙΑ_ΛΕΞΕΩΝ**

Με αυτήν την οδηγία, ο προμεταγλωττιστής αντικαθιστά σε όλο το υπολοιπόμενο κείμενο του πηγαίου κώδικα, το κειμένο ΛΕΞΗ με το κείμενο ΑΚΟΛΟΥΘΙΑ_ΛΕΞΕΩΝ (*μακροαντικατάσταση κειμένου*).

Συντακτικοί κανόνες: Το κείμενο ΛΕΞΗ μπορεί να ειναι οποιαδήποτε ακολουθία απο αλφαριθμητικούς λατινικούς χαρακτήρες (λατινικά γράμματα πεζά ή κεφαλαία, τα αριθμητικά ψηφία 0-9) η του χαρακτήρα underscore "_". Περιορισμοί τίθενται στο μήκος (μικρότερο απο 32 χαρακτήρες), τον πρώτο χαρακτήρα (όχι αριθμητικό ψηφίο) και τέλος, ακριβώς το ίδιο κείμενο δεν πρέπει, να ειναι κωδική λέξη της C ή να έχει χρησιμοποιηθεί σε προηγούμενες οδηγίες #define.

Το κείμενο ΑΚΟΛΟΥΘΙΑ_ΛΕΞΕΩΝ ειναι είτε ακολουθία εντολών της γλωσσας C, είτε λέξεις που έχουν σημειωθεί για μακροαντικατάσταση σε προηγούμενες οδηγίες. Στο κείμενο ΑΚΟΛΟΥΘΙΑ_ΛΕΞΕΩΝ επιτρέπονται κενά, όχι όμως αλλαγή γραμμής αφού αυτή σημειώνει το τέλος της οδηγίας #define.

β) Η οδηγία #**include**. Έχει την σύνταξη:

#include ONOMA

Με αυτήν την οδηγία, ο προμεταγλωττιστής εισάγει στην τρέχουσα θέση σάρωσης, όλο το κείμενο που περιέχει το αρχείο ONOMA (*ενσωμάτωση πηγαίου κώδικα*). Το κείμενο ONOMA ειναι, είτε η διαδρομή εύρεσης του αρχείου μέσα στο δέντρο αρχείων του λειτουργικού συστήματος UNIX οπότε περικλείεται απο διπλά quotes " " (π.χ. "./dir/name.c"), είτε μιά διαδρομή κάτω απο το δέντρο αρχείων /usr/include, οπότε περικλείεται απο τα σύμβολα ανισώσεων "<" , ">". Στις δικές μας εφαρμογές θα βάζουμε πάντα, τις παρακάτω εντολές ενσωμάτωσης αρχείων:

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

ΠΑΡΑΔΕΙΓΜΑ: Ο πιό στοιχειώδης πηγαίος κώδικας που με αυτές τις γνώσεις μπορούμε να φτιάξουμε ειναι:

```
#include   <stdio.h>
#include   <stdlib.h>
#include   <math.h>
main( )
{ }
```

Αυτός ο κώδικας μεταγλωττίζεται και παράγει ένα αντικειμενικό κώδικα που όταν εκτελείται έχει μηδενική λειτουργικότητα, ΔΕΝ ΚΑΝΕΙ ΤΙΠΟΤΑ. Η χρησιμότητά του έγκειται στο ότι:

α) ΠΑΝΤΑ θα ξεκινούμε απο αυτόν και στην συνέχεια θα τον επαυξάνουμε με εντολές της C, προκειμένου να παραχθεί ένας πηγαίος κώδικας που θα παράγει αντικειμενικό κώδικα ζητούμενης λειτουργικότητας.

β) Κατανοούμε τις μορφολογικές μεταβολές που επιτρέπονται στην C, πχ εισάγοντας ή αφαιρώντας κενές γραμμές ή αποστάσεις μεταξύ των αγκίστρων ή των παρενθέσεων παρατηρούμε ότι οδηγούμαστε πάλι σε μεταγλωττίσιμο κώδικα που παράγει τον αντικειμενικό κώδικα μηδενικής λειτουργικότητας.

5) Στο κυρίως τμήμα του πηγαίου κώδικα, περιγράφεται **ο ορισμός και η επεξεργασία μεταβλητών**.
Ως μεταβλητή θα θεωρούμε μιά συνεχόμενη περιοχή μνήμης[2] που ο μεταγλωττιστής έχει αντιστοιχίσει σε κάποια λέξη δικής μας επιλογής. Η αρχική διεύθυνση αυτής της μνήμης δεν μας ειναι γνωστή (άν χρειαστεί μπορούμε να την μάθουμε), όμως, τόσο το μέγεθός της μετρημένο σε bytes, όσο και η ερμηνεία του περιεχομένου αυτής της μνήμης, καθορίζεται άμεσα απο τον **ορισμό** της. Ώς ορισμό της μεταβλητής εννοούμε την εντολή προς τον μεταγλωττιστή για να δημιουργήσει την απαιτούμενη αντιστοίχιση.
Τα βασικά χαρακτηριστικά μιάς μεταβλητής ειναι το **όνομα**, ο **τύπος**, η **εμβέλεια** ορισμού και τέλος, η **τιμή** της. Πιό αναλυτικά:

α) **ΟΝΟΜΑ**: Πρόκειται για την λέξη που επιλέξαμε για να αναφερόμαστε στην μεταβλητή. Οποιαδήποτε **αναφορά** στην μεταβλητή (χρήση του ονόματός της), πρέπει πάντα να έπεται του ορισμού της.
Στην επιλογή του ονόματος ισχύουν οι ίδιοι συντακτικοί κανόνες με της επιλογής του κειμένου ΛΕΞΗ, στην οδηγία μακροαντικατάστασης #define.

[2] Η μνήμη ενός Η/Υ συγκροτείται απο αριθμημένα κελιά μνήμης του ενός **byte**, δηλαδή καθένα απο αυτά μπορεί να αποθηκεύσει οποιαδήποτε απο τις $2^8 = 256$ διαφορετικές διατάξεις που σχηματίζουν 8 δυαδικά ψηφία (**bits**). Γενικά 1byte πληροφορίας αποτελείται απο 8bits. Προκειμένου να γίνει εγγραφή ή ανάσυρση του περιεχομένου σε ένα απο αυτά τα κελιά μνήμης, απαιτείται η **διεύθυνσή του**, δηλαδή ο αύξων αριθμός του στην αρίθμηση της μνήμης.

ΠΑΡΑΤΗΡΗΣΗ: Το όνομα μιάς μεταβλητής ίσως τερματίζει σε ζεύγος παρενθέσεων "(", ")", ή σε ζεύγος αγκύλων "[", "]". Αυτές ειναι ειδικότερες κατηγορίες μεταβλητών, πρόκειται για τις **συναρτήσεις** και τις **διανυσματικές μεταβλητές** αντίστοιχα, που θα δούμε αργότερα.

β) **ΤΥΠΟΣ**: Πρόκειται για τυποποιημένες λέξεις της C οι οποίες χρησιμοποιούνται για την κατηγοριοποίηση των μεταβλητών. Μέσω του τύπου μιας μεταβλητής, ο μεταγλωττιστής αντιστοιχεί στο όνομα της μεταβλητής τόσο το απαιτούμενο μέγεθος μνήμης, όσο και τον τρόπο ερμηνείας του περιεχομένου αυτής της μνήμης. Οι τυποποιημένοι τύποι μεταβλητών της C ειναι, είτε οι τύποι για ακέραιους αριθμούς:

char (τύπος "*χαρακτήρα*", μέγεθος 1 byte), **short** (τύπος "*μικρού ακέραιου*", μέγεθος 2 bytes),
int (τύπος "*ακέραιου*", μέγεθος 4 bytes), **long** (τύπος "*μεγάλου ακέραιου*", μέγεθος 4 ή 8 bytes αναλόγως του μεταγλωττιστή),

είτε οι τύποι για ρητούς αριθμούς:

float (τύπος "*πραγματικού απλής ακρίβειας*", μέγεθος 4 bytes),
double (τύπος "*πραγματικού διπλής ακρίβειας*", μέγεθος 8 bytes).

Το συντακτικό που χρησιμοποιείται για τον ορισμό μεταβλητών έχει την μορφή:

ΠΡΟΣΔΙΟΡΙΣΤΗΣ ΤΥΠΟΣ ΜΕΤΑΒΛΗΤΗ$_1$, ΜΕΤΑΒΛΗΤΗ$_2$, . . . , ΜΕΤΑΒΛΗΤΗ$_k$;

όπου

- Ο ΠΡΟΣΔΙΟΡΙΣΤΗΣ ειναι μιά απο τις κωδικές λέξεις:
unsigned (προσδιορίζει τον τύπο ώς απρόσημο), **signed** (προσδιορίζει τον τύπο ώς προσημασμένο), **const** (προσδιορίζει τον τύπο ώς σταθερής τιμής), **extern** (ταυτίζει την μεταβλητή με κάποια ίδιου ονόματος και τύπου που βρίσκεται σε εξωτερικό αρχείο που ενσωματώνεται), **static** (αποταυτίζει την μεταβλητή απο όσες έχουν το ίδιο όνομα). Εμείς θα χρησιμοποιήσουμε το πολύ τους προσδιοριστές signed, unsigned, οι οποίοι εφαρμόζονται μόνο σε τύπους ακεραίων (οι τύποι float, double ειναι πάντα προσημασμένοι). Άν κατα τον ορισμό μεταβλητής ακέραιου τύπου δεν υπάρχει προσδιοριστής προσήμου, θεωρείται προσημασμένη.
- Ο ΤΥΠΟΣ ειναι μιά απο τις αναφερθείσες κωδικές λέξεις τύπων.
- ΜΕΤΑΒΛΗΤΗ$_1$, ΜΕΤΑΒΛΗΤΗ$_2$, . . . , ΜΕΤΑΒΛΗΤΗ$_k$ ειναι τα ονόματα των μεταβλητών που ορίζονται χωρισμένα με κόμμα ",". Η εντολή ορισμού τερματίζει σε ερωτηματικό ";" .

ΠΑΡΑΤΗΡΗΣΗ: ΓΕΝΙΚΑ ΟΛΕΣ ΟΙ ΕΝΤΟΛΕΣ ΤΗΣ C ΤΕΡΜΑΤΙΖΟΥΝ ΣΕ ΕΡΩΤΗΜΑΤΙΚΟ.

ΠΑΡΑΔΕΙΓΜΑΤΑ ΟΡΙΣΜΟΥ ΜΕΤΑΒΛΗΤΩΝ (ΣΕ ΠΕΡΙΟΧΕΣ ΟΡΙΣΜΟΥ)

```
int i, j, metav1;            ( ορίζονται 3 μεταβλητές ακέραιου τύπου, με ονόματα i, j, metav1 )
unsigned char gf, meta;      ( ορίζονται 2 μεταβλητές απρόσημου τύπου χαρακτήρα, με ονόματα gf, meta )
double y_12;                 ( ορίζεται 1 μεταβλητή πραγματικού διπλής ακρίβειας, με όνομα y_12 )
```

γ) **ΕΜΒΕΛΕΙΑ**: Πρόκειται για το τμήμα του πηγαίου κώδικα στο οποίο έχει νόημα ο ορισμός της μεταβλητής, δηλαδή το όνομα της παραπέμπει σε κάποια περιοχή μνήμης. Στο τμήμα αυτό, οποιαδήποτε αναφορά στην μεταβλητή αντικαθίσταται με την τιμή που έχει σε εκείνο το σημείο.
ΓΕΝΙΚΟΣ ΚΑΝΟΝΑΣ: Η εμβέλεια οποιασδήποτε μεταβλητής ειναι όλο το επόμενο μέρος αμέσως μετά τον ορισμό της (οπότε σε οποιοδήποτε πηγαίο κώδικα, ΔΕΝ μπορούν να συνυπάρχουν διαφορετικές μεταβλητές ίδιου ονόματος και κοινής εμβέλειας).
ΠΕΡΙΟΡΙΣΜΟΣ ΤΗΣ ΕΜΒΕΛΕΙΑΣ: Οι μεταβλητές που ορίζονται μέσα σε block έχουν εμβέλεια εκεί, ενώ, έξω απο το block το όνομά τους δεν αντιστοιχεί σε περιοχή μνήμης. Στην περίπτωση που ένα όνομα εχει χρησιμοποιηθεί σε ορισμό μεταβλητής, τόσο πρίν όσο και μέσα σ' ένα block, τότε μέσα στο block ισχύει ο τοπικός ορισμός (σε αυτήν την περίπτωση αν και ίδιου ονόματος έχουν αντιστοιχηθεί διαφορετικές περιοχές μνήμης σε κάθε μεταβλητή, η εμβέλεια της έξωτερικής μεταβλητής δεν περιλαμβάνει το block).
Στην περίπτωση που ενσωματώνονται κώδικες απο εξωτερικά αρχεία (με την εντολή #include), την ταύτιση ή την αποταύτιση μεταβλητών με ίδιο όνομα, την καθορίζει ή χρήση των προσδιοριστών extern ή static αντίστοιχα.

δ) **ΤΙΜΗ**: Πρόκειται για το περιεχόμενο της περιοχής μνήμης που έχει αντιστοιχηθεί στο όνομα της, ύστερα όμως απο ερμηνεία που εξαρτάται απο τον τύπο της.

ΠΑΡΑΤΗΡΗΣΗ: Η περιοχή μνήμης που έχει αντιστοιχηθεί στο όνομα μίας μεταβλητής, δεν αλλάζει σε όλη την εμβέλειά της μεταβλητής ωστόσο το περιεχόμενο αυτής της μνήμης μπορεί να αλλάζει (εκτός απο τις μεταβλητές που έχουν προσδιορισθεί const οι οποίες έχουν μόνιμα την αρχική τιμή).
Το πεδίο τιμών μιάς μεταβλητής, έχει πληθυκότητα που καθορίζεται απο το μέγεθος της μνήμης που απαιτεί ο τύπος της. Ένα byte μνήμης μπορεί να αποθηκεύσει έναν απο τους 256 φυσικούς $0, 1, .., 255$, οπότε συνεχίζοντας συνδυαστικά, 2 bytes μνήμης αποθηκεύουν έναν απο τους $256^2 = 65536$ διαφορετικούς φυσικούς $0, 1, .., 65535$, 4 bytes μνήμης αποθηκεύουν έναν απο τους $256^4 = 4294967296$ διαφορετικούς φυσικούς $0, 1, .., 4294967295$ και τελος, 8 bytes μνήμης αποθηκεύουν έναν απο τους $256^8 = ...$ διαφορετικούς φυσικούς.
Η τιμή που τελικά θεωρείται ότι έχει μία μεταβλητή, αν ειναι απρόσημου ακεραίου τύπου (unsigned char, unsigned short, unsigned int, unsigned long) ταυτίζεται με τον αποθηκευμένο φυσικό στην αντίστοιχη μνήμη, όμως για τους υπόλοιπους τύπους βρίσκεται ερμηνεύοντας κατάλληλα αυτόν τον αποθηκευμένο φυσικό αριθμό.
Στους προσημασμένους τύπους ακεραίων αριθμών char, short, int, long (οι οποίοι έχουν ίδιο μέγεθος μνήμης με τους αντίστοιχους απρόσημους) το πλέον σημαντικό δυαδικό ψηφίο εκλαμβάνεται ώς πληροφορία προσήμου με αποτέλεσμα, στον τύπο char ο αποθηκευμένος φυσικός που ειναι ένας απο τους παραπάνω 256, να ερμηνεύεται ώς κάποιος απο τους ακεραίους $-128, .., -1, 0, 1, .., 127$. Στον τύπο short ο αποθηκευμένος φυσικός που ειναι ένας απο τους παραπάνω 65536, ερμηνεύεται ώς κάποιος απο τους ακεραίους $-32768, , .., -1, 0, 1, .., 32767$, κ.λ.π..
ΓΕΝΙΚΑ, στον τύπο char ο αποθηκευμένος φυσικός k ερμηνεύται ώς αρνητικός ακέραιος άν υπερβαίνει τον αριθμό 127 (γιατί αυτοί οι φυσικοί έχουν ώς πλέον σημαντικό δυαδικό ψηφίο το 1) οπότε σε αυτήν την περίπτωση συμβολίζει τον αρνητικό k-256, π.χ. οι αποθηκευμένοι φυσικοί $255, 254, .., 129, 128$ ερμηνεύονται ώς οι αρνητικοί ακέραιοι $-1, -2, ..., -127, -128$ αντίστοιχα. Ανάλογα ισχύουν για τους τύπους short, int, long.
Για τους τύπους ρητών αριθμών float, double, η ερμηνεία του αποθηκευμένου φυσικού ειναι πιό πολύπλοκη. Το τελικό αποτέλεσμα ειναι ένας ρητός αριθμός στο διάστημα πραγματικών $(10^{-128}, 10^{127})$ (το διάστημα αυτό ίσως ειναι διαφορετικό, εξαρτάται απο τον μεταγλωττιστή). Προφανώς λόγω του μεγέθους των τύπων float, double, το πλήθος των ρητών που μπορούν να εκπροσωπηθούν ειναι 256^4 και 256^8 αντίστοιχα.

ΣΗΜΕΙΩΣΗ: Ο αποθηκευμένος φυσικός αριθμός σε μιά μεταβλητή, ταυτίζεται σίγουρα με την τιμή της μόνο άν αυτή ειναι απρόσημου ακέραιου τύπου (unsigned char, unsigned short, unsigned int, unsigned long). Στην γενική περίπτωση δηλαδή,

ΠΕΡΙΕΧΟΜΕΝΟ ΜΕΤΑΒΛΗΤΗΣ $\neq$ ΤΙΜΗ ΜΕΤΑΒΛΗΤΗΣ.

Ωστόσο, σε όλους τους τύπους μεταβλητών η τιμή μηδέν αναπαρίσταται στην μνήμη ΠΑΝΤΑ με τον ίδιο φυσικό αριθμό, το μηδέν .

ΠΕΡΙΟΧΗ ΕΠΕΞΕΡΓΑΣΙΑΣ ΜΕΤΑΒΛΗΤΩΝ

Σε αυτήν την περιοχή οι τιμές των μεταβλητών αλλάζουν μέ κάποιον απο τους παρακάτω πιθανούς τρόπους

1) **ΑΠΟΔΟΣΗ ΤΙΜΗΣ :** Σε μιά μεταβλητή μπορεί να **αποδοθεί** τιμή ακόμα και κατά τον ορίσμό της, χρησιμοποιώντας σύνταξη της μορφής

ΜΕΤΑΒΛΗΤΗ = ΤΙΜΗ ;

για πολλές αποδόσεις τιμών σε μεταβλητές μπορεί να χρησιμοποιηθεί η σύνταξη

ΜΕΤΑΒΛΗΤΗ$_1$ = ΤΙΜΗ$_1$, ΜΕΤΑΒΛΗΤΗ$_2$ = ΤΙΜΗ$_2$, ..., ΜΕΤΑΒΛΗΤΗ$_k$ = ΤΙΜΗ$_k$;

Στο δεξιό μέρος του "=" αναγράφεται είτε άμεσα η τιμή που θα αποδοθεί, είτε το όνομα μιάς μεταβλητής που έχει ορισθεί (οπότε η τιμή που έχει σε αυτό το σημείο ειναι και αυτή που αποδίδεται στη αριστερή μεταβλητή), είτε τέλος μιά **παράσταση** όπου συσχετίζονται πολλές τιμές.

ΠΑΡΑΤΗΡΗΣΗ: **Η απόδοση τιμής σε μεταβλητή συνοδεύεται πάντα απο την προσαρμογή της στον τύπο της μεταβλητής**. Αυτό σημαίνει ότι, όταν ο τύπος της τιμής έχει μέγεθος μεγαλύτερο απο αυτό του τύπου της μεταβλητής, τότε αποκόπτονται όσα bytes δεν χωράνε ξεκινώντας απο τα πιό σημαντικά. Ειδικότερα, όταν αποδίδεται ρητή τιμή σε μεταβλητή ακέραιου τύπου, τότε αποκόπτονται και τα δεκαδικά ψηφία. Τέλος, ακόμα και η απόδοση ρητής τιμής σε μεταβλητή ρητού τύπου, συνοδεύεται απο προσαρμογή που οφείλεται στην πληθυκότητα των ρητών αριθμών. Στο διάστημα των αριθμών που χρησιμοποιεί ο μεταγλωττιστής περιέχονται άπειρες σε πλήθος ρητές τιμές ωστόσο το πολύ 256^8 απο αυτές μπορούν να αποθηκευτούν σε μεταβλητή ρητού τύπου. Ώς εκ τούτου, κατα την απόδοση ρητής τιμής σε μεταβλητή ρητού τύπου, η τιμή αυτή στρογγύλευεται στον κοντινότερο απο τους ρητούς που χρησιμοποιεί ο μεταγλωττιστής.

Πολλές φορές χρησιμοποιείται η πολλαπλή απόδοση τιμής:

ΜΕΤΑΒΛΗΤΗ$_1$ = ΜΕΤΑΒΛΗΤΗ$_2$ = ... = ΜΕΤΑΒΛΗΤΗ$_k$ = ΤΙΜΗ

Εδώ, ο δεξιότερος όρος ειναι η τιμή η οποία αποδίδεται στην ΜΕΤΑΒΛΗΤΗ$_k$, έπειτα η προασαρμοσμένη τιμή στον τύπο της ΜΕΤΑΒΛΗΤΗ$_k$ αποδίδεται στην ΜΕΤΑΒΛΗΤΗ$_{k-1}$, ... (συνεχίζοντας με αυτόν τον τρόπο), ... η προασαρμοσμένη τιμή στον τύπο της ΜΕΤΑΒΛΗΤΗ$_2$ αποδίδεται στην ΜΕΤΑΒΛΗΤΗ$_1$.
ΠΑΡΑΤΗΡΗΣΗ: Πολλαπλή απόδοση μπορεί να γίνει μόνο μετά τον ορισμό των εμπλεκομένων μεταβλητών.

ΠΑΡΑΔΕΙΓΜΑΤΑ

i) int y=3, i, j; (η μεταβλητή y παίρνει την τιμή 3 κατά τον ορισμό της)
int x=6.53; (λόγω προσαρμογής τύπου αποκόπτονται τα δεκαδικά ψηφία, άρα η x παίρνει την τιμή 6)
unsigned short k=120000; (λόγω προσαρμογής τύπου μόνο τα μικρότερα 2 bytes της τιμής θα αποδοθούν, άρα η k παίρνει την τιμή (120000 mod 65536)=54464)

ii) double met ;
int k;
met=k=-34.5; (και οι δύο μεταβλητές met, k παίρνουν την τιμή -34 λόγω προσαρμογής τύπου)
k=met=-34.5; (η μεταβλητή met παίρνει την τιμή -34.5, ενώ η μεταβλητή k παίρνει την τιμή -34)

iii) double x=5.8;
int i=x, j=350; (η μεταβλητή i λόγω προσαρμογής τύπου παίρνει την τιμή 5)
char m=j; (λόγω προσαρμογής τύπου μόνο το μικρότερο byte θα αποδοθεί, άρα η m παίρνει την τιμή (350 mod 256)=94)

iv) Μιά ρητή τιμή που αποδίδεται άμεσα, περιγράφεται είτε σε δεκαδική μορφή π.χ.
double x=-5.8;
float a=100.327;
είτε σε επιστημονική μορφή MeD η οποία συμβολίζει τον αριθμό M·10^{D}, π.χ. οι προηγούμενες αποδόσεις τιμών μπορούν να δοθούν και στην μορφή
double x=-58.e-1;
float a=1.00327e2;
Η a παίρνει την ίδια τιμή και με αποδόσεις της μορφής a=0.100327e3, a=1003.27e-1, κ.λ.π.

ΥΠΕΝΘΥΜΙΣΗ: Η τιμές που τελικά παίρνουν οι μεταβλητές x, a ίσως ειναι διαφορετικές απο τους αριθμούς -5.8 και 100.327 αντίστοιχα λόγω προσαρμογής στους ρητούς του μεταγλωττιστή.

v) Οι μεταβλητές τύπου char χρησιμοποιούνται βασικά για την καταχώρηση **χαρακτήρων κειμένου**.

> Σύστημα ASCII: Πρόκειται για τυποποιημένη αντιστοίχιση των γραμμάτων του λατινικού αλφαβήτου, των αριθμητικών ψηφίων, ειδικών συμβόλων (! $, . ; { } ...), καθώς και κάποιων λειτουργιών εκτύπωσης ("αλλαγή γραμμής", "απόσταση 8 κενών"(tab), ...), στους φυσικούς 0, 1, ..., 255.

Προκειμένου να αποθηκεύσουμε έναν χαρακτήρα κειμένου σε μεταβλητή τύπου char, αποδίδουμε στην μεταβλητή τον κωδικό-ascii αυτού του χαρακτήρα. Η απόδοση αυτή γίνεται κλείνοντας τον χαρακτήρα σε ' ' (απλά quotes). Ειδικότερα με '\n' και '\t' συμβολίζονται οι κωδικοί-ascii των λειτουργιών "αλλαγή γραμμής" και "tab" αντίστοιχα.

ΠΑΡΑΔΕΙΓΜΑ

```
char k='a', lar1=31, m=']', r='\n' ;
```

(η μεταβλητή k παίρνει την τιμή του κωδικού-ascii του χαρακτήρα "a", η μεταβλητή lar1 παίρνει την τιμή 31, η m παίρνει την τιμή του κωδικού-ascii του χαρακτήρα "]" και η r παίρνει την τιμή του κωδικού-ascii της λειτουργίας "αλλαγή γραμμής").

ΠΑΡΑΣΤΑΣΕΙΣ

Παράσταση θεωρείται οποιαδήποτε ακολουθία απο παρενθέσεις (ανα ζεύγη), αριθμούς (σε δεκαδική ή επιστημονική μορφή), ονόματα μεταβλητών (που έχουν ήδη ορισθεί), τα ' ' (απλά quotes), το "," (κόμμα), το "=" (ίσον), το "?" (questionmark), το ":" (άνω-κάτω τελεία), τα σύμβολα των πράξεων

αριθμητικής		bitwise λογικής		ολίσθησης		συντομογραφίες
+	πρόσθεση	&	και	>>	προς τα δεξιά	$+=$, $-=$, $*=$
-	αφαίρεση	|	η	<<	προς τα αριστερά	$/=$, $\%=$
*	πολ/σμός	^	αποκλειστικό η			$\&=$, $\|=$, $\hat{}=$
/	διαίρεση	~	συμπλήρωμα			$>>=$, $<<=$
%	υπόλοιπο διαίρεσης					$++$, $--$

,

καθώς και τα σύμβολα

σύκρισης αριθμητικών τιμών		λογικής συσχέτισης τιμών	
>	η αριστερή ειναι μεγαλύτερη απο την δεξιά τιμή	!	αποτέλεσμα της άρνησης
<	η αριστερή ειναι μικρότερη απο την δεξιά τιμή	&&	αποτέλεσμα του λογικού και
>=	η αριστερή ειναι μεγαλύτερη ή ίση απο την δεξιά τιμή	||	αποτέλεσμα του λογικού η
<=	η αριστερή ειναι μικρότερη ή ίση απο την δεξιά τιμή		
==	η αριστερή ειναι ίση με την δεξιά τιμή		
!=	η αριστερή ειναι διάφορη με την δεξιά τιμή		

.

ΠΑΡΑΔΕΙΓΜΑΤΑ: (met+3*5.3e-12/pi)+12, (b=5*a, b+3), (met<=8) & (5+3*i) .

ΤΙΜΗ ΠΑΡΑΣΤΑΣΗΣ

Ως τιμή μιάς παράστασης θεωρείται το αποτέλεσμα όλων των πράξεων μεταξύ των μεταβλητών και των αριθμών που περιέχει και αποδίδεται είτε ώς τύπου double είτε ώς τύπου long (δηλαδή με το μέγιστο μέγεθος). Πιο συγκεκριμένα, μόνο στις αριθμητικές πράξεις +, -, *, / και τις αντίστοιχες συντομογραφίες +=, -=, *=, /=, η υπολογισθείσα τιμή ειναι τύπου double και αυτό, μόνο άν εμπλέκεται τουλαχιστον ένα όρισμα τύπου double. Σε όλες τις άλλες πράξεις η υπολογισθείσα τιμή ειναι τύπου long.

Ειδικότερα, μιά παράσταση για την οποία μας ενδιαφέρει μόνο άν η τιμή της ειναι ίση ή διάφορετική απο μηδέν, θα αναφέρεται ώς **συνθήκη**.

Μιά συνθήκη θεωρείται **αληθής όταν έχει τιμή διάφορετική απο το μηδέν** και **ψευδής όταν έχει τιμή ίση με το μηδέν.**

Ο υπολογισμός της τιμής μιάς παράστασης ακολουθεί τους παρακάτω κανόνες:

i) Κάθε παράσταση υπολογίζεται με επαναλαμβανόμενα εσωτερικά βήματα. Σε κάθε βήμα **η παράσταση σαρώνεται απο αριστερά προς τα δεξιά** και εκτελούνται οι πράξεις μεγαλύτερης προτεραιότητος. Στο τέλος κάθε βήματος ο υπολογισμός της τιμής έχει αναχθεί στον υπολογισμό απλούστερης παράστασης. Η διαδικασία συνεχίζεται μέχρι να προκύψει μία τιμή αλλά το πολύ για έναν πεπερασμένο αριθμό βημάτων, κάποιες χιλιάδες αλλά όχι άπειρα...

ii) Γενικά ειναι καθορισμένη η προτεραιότητα όλων των πράξεων, πχ προηγούνται οι πολ/αστικές πράξεις *, /, % έναντι των προσθετικών +, -, η άρνηση και το συμπλήρωμα έναντι όλων των άλλων λογικών πράξεων στις οποίες το "λογικό και" έχει προτεραιότητα εναντι του "λογικού η", κλπ.
ΠΑΡΑΔΕΙΓΜΑ: Η παράσταση 3+5*4/2-7 υπολογίζεται εσωτερικά με τα εξης βήματα:

$$3+20/2-7 \rightarrow 3+10-7 \rightarrow 13-7 \rightarrow 6$$

iii) Άν και η σειρά προτεραιότητος των πράξεων ειναι αυστηρά καθορισμένη, πρακτικά ειναι δύσκολη αλλά και αχρείαστη η απομνημόνευση της. Στην γενική περίπτωση οι πράξεις ειναι απαραίτητο να γίνουν με άλλη σειρά. Για την αλλαγή σειράς στην προτεραιότητα χρησιμοποιούνται τα ζεύγη των παρενθέσεων. **Οι πράξεις που περιέχονται σε παρενθέσεις έχουν μεγαλύτερη προτεραιότητα απο αυτές που ειναι έξω απο τις παρενθέσεις.**
ΠΑΡΑΔΕΙΓΜΑ: Η παράσταση (3+5)*4/(2-7) υπολογίζεται εσωτερικά με τα εξης βήματα:

$$8*4/(2-7) \rightarrow 8*4/-5 \rightarrow 32/-5 \rightarrow -6$$

οπότε η τιμή της ειναι ο αριθμός -6, τύπου long.

ΠΑΡΑΤΗΡΗΣΗ: Το γεγονός ότι σε κάθε βήμα υπολογισμού, δεν υπήρχαν ορίσματα πραγματικού τύπου (όλα ηταν ακέραιοι) οδήγησε σε αποτέλεσμα τύπου long, οπότε τα δεκαδικά ψηφία έχουν αποκοπεί.

iv) Μέσα σε παρενθέσεις μπορεί να υπάρχουν περισσότερες απο μία παραστάσεις χωρισμένες με κόμμα, ή ακόμα και απόδοση τιμής σε μεταβλητές. Σε αυτές τις περιπτώσεις, βρίσκονται οι τιμές όλων των παραστάσεων που περιέχονται ξεκινώντας απο αριστερά προς τα δεξιά και ώς τιμή της συνολικής παράστασης λαμβάνεται η τιμή της δεξιότερης παράστασης.

ΠΑΡΑΔΕΙΓΜΑΤΑ : Έστω ότι προηγείται ο ορισμός int j, l, m; τότε:

i) Η παράσταση (j=2, l=2*j, 5.2*l+j) έχει την τιμή της παράστασης 5.2*l+j η οποία λόγω των προηγούμενων εσωτερικών παραστάσεων j=2, l=4 έχει τιμή 5.2*4+2, δηλαδή έχει τιμή 22.8, τύπου double.

ii) Η παράσταση (j=2, l=2*j, m=5.2*l+j) έχει την τιμή της παράστασης m=5.2*l+j, η οποία λόγω των προηγούμενων εσωτερικών παραστάσεων j=2, l=4 έχει την τιμή της παράστασης m=22.8, δηλαδή την τιμή 22.8 αφού αποδοθεί στην μεταβλητή m, οπότε λόγω της υφιστάμενης προσαρμογής τύπου έχει τιμή 22 τύπου int.

v) Οι αριθμητικές πράξεις +, -, *, / απαιτούν κατα τα γνωστά 2 ορίσματα και όπως αναφέρθηκε επιστρέφουν μία τιμή που ειναι τύπου double μόνο άν τουλάχιστον ένα απο τα ορίσματα ειναι τύπου double αλλιώς ειναι τύπου long. Αυτό στην περίπτωση της διαίρεσης ίσως ειναι καταστροφικό πχ η παράσταση $5/2$ έχει τιμή 2 λόγω της προσαρμογής τύπου (παρακάτω θα δούμε πώς παρακάμπτεται η δυσκολία). Παράσταση της μορφής A % B (όπου A, B ειναι ακέραιοι αριθμοί ή παραστάσεις ακέραιας τιμής) έχει ώς τιμή **το υπόλοιπο της Ευκλείδιας διαίρεσης του A δια του B** .

ΠΑΡΑΔΕΙΓΜΑ :

int x=16, y=3, z;

z = x % y ;

Εδώ η μεταβλητή z πήρε την τιμή 16 mod 3 =1 .

vi) Οι παραστάσεις που περιέχουν λογικές πράξεις "&, $|$, $\hat{}$, $\sim$" ή ολισθήσεις ">>, <<" εφαρμόζονται μόνο σε ακέραιους. **Για να προσδιορίσουμε την τιμή αυτών των παραστάσεων ειναι χρήσιμο να γνωρίζουμε τα δυαδικά ψηφία των εμπλεκομένων ορισμάτων τουλάχιστον στο μέγεθος του τύπου των. (υπενθύμιση: οι char έχουν μέγεθος 8 bit, οι short 16 bit, οι int 32 bit και οι long 32 ή 64 bit)**.

Απο αυτές τις πράξεις, το "λογικό συμπλήρωμα"$\sim$ απαιτεί μόνο έναν ακέραιο και επιστρέφει εκείνον τον ακέραιο που έχει συμπληρωματικά δυαδικά ψηφία παντού.

ΠΑΡΑΔΕΙΓΜΑΤΑ : Έστω ότι προηγούνται οι ορισμοί

unsigned char j=136; και unsigned short k=136;

i) Για να υπολογίσουμε την τιμή της παράστασης $\sim$ j, πρέπει πρώτα να βρούμε την δυαδική αναπαράσταση της τιμής που έχει η j σε όλο το μέγεθος του τύπου της δηλαδή $136 = 10001000_2$, οπότε συμπεραίνουμε ότι $\sim$ j=$01110111_2 = 119$.

ii) Για να υπολογίσουμε την τιμή της παράστασης $\sim$ k, πρέπει πρώτα να βρούμε την δυαδική αναπαράσταση της τιμής που έχει η k σε όλο το μέγεθος του τύπου της δηλαδή $136 = 0000000010001000_2$, οπότε συμπεραίνουμε ότι $\sim$ k=$1111111101110111_2 = 65399$.

ΠΑΡΑΤΗΡΗΣΗ: Η συμπληρωματική τιμή μίας τιμής A μεγέθους m bit ειναι $(2^{\mathbf{m}}$ **-1)-A** πχ στο παράδειγμα i) όπου το μέγεθος ειναι 8 bit βρέθηκε η συμπληρωματική τιμή 255-136=119, ενώ στο παράδειγμα ii) βρέθηκε η συμπληρωματική τιμή 65535-136=65399.

ΕΡΩΤΗΣΗ: Άν δεν ήταν απρόσημου τύπου οι παραπάνω μεταβλητές, τελικά τί τιμή θα είχαν;

Οι υπόλοιπες λογικές πράξεις και οι ολισθήσεις απαιτούν δύο ακεραίους και δίνουν ώς αποτέλεσμα έναν ακέραιο ώς ακολούθως:

. Το "λογικό και" A & B, μεταξύ των τιμών A, B επιστρέφει τον ακέραιο που έχει δυαδικά ψηφία τύπου 1 ΜΟΝΟ στις θέσεις όπου και οι δύο ακέραιοι έχουν δυαδικό ψηφίο 1.

. Το "λογικό ή" A $|$ B, μεταξύ των τιμών A, B επιστρέφει τον ακέραιο που έχει δυαδικά ψηφία τύπου 1 ΜΟΝΟ στις θέσεις όπου τουλάχιστον ένας απο τους δύο ακεραίους έχει δυαδικό ψηφίο 1.

. Το "αποκλειστικό λογικό ή" A^B, μεταξύ των τιμών A, B επιστρέφει τον ακέραιο που έχει δυαδικά ψηφία τύπου 1 ΜΟΝΟ στις θέσεις όπου ακριβώς ένας απο τους δύο ακεραίους έχει δυαδικό ψηφίο 1.

. Η προς τα "αριστερά ολίσθηση" A << B, επιστρέφει έναν ακέραιο που έχει ίδια δυαδικά ψηφία με τον A αλλά μετατοπισμένα προς τα αριστερά τόσες θέσεις όση ειναι η τιμή B, συμπληρώνοντας με 0 τις κενές θέσεις που προκύπτουν στα δεξιά.
ΥΠΕΝΘΥΜΙΣΗ: Κάθε μετατόπιση των δυαδικών ψηφίων της A προς τα αριστερά συμπληρώνοντας με 0 τα κενά, διπλασιάζει την τιμή A.

. Η προς τα "δεξιά ολίσθηση" A >> B, επιστρέφει έναν ακέραιο που έχει ίδια δυαδικά ψηφία με τον A αλλά μετατοπισμένα προς τα δεξιά τόσες θέσεις όση ειναι η τιμή B. Οι κενές θέσεις που προκύπτουν στα αριστερά συμπληρώνονται με 0 άν η τιμή A ειναι απρόσημου τύπου, Όταν άν η τιμή A ειναι προσημασμένου τύπου τότε οι κενές θέσεις που προκύπτουν στα αριστερά συμπληρώνονται με την ίδια τιμή που έχει το μέγιστα σημαντικό ψηφίο.
ΥΠΕΝΘΥΜΙΣΗ: Κάθε μετατόπιση των δυαδικών ψηφίων των δυαδικών ψηφίων της A προς τα δεξιά, οδηγεί στο ακέραιο πηλίκο της τιμής A δια 2. Η συμπλήρωση με τους παραπάνω κανόνες γίνεται για να διατηρείται το πρόσημο της τιμής A.

ΠΑΡΑΔΕΙΓΜΑΤΑ :

i)Έστω ότι προηγούνται οι ορισμοί unsigned char j=136, m=10; τότε j=10001000_2, m=00001010_2 οπότε:

η παράσταση j & m έχει τιμή 00001000_2=8,

η παράσταση j | m έχει τιμή 10001010_2=138,

η παράσταση j^m έχει τιμή 10000010_2=130.

Φυσικά τα ορίσματα δεν ειναι πάντα μεταβλητές, μπορεί να ειναι άμεσες τιμές ή παραστάσεις πχ η παράσταση j & 3 έχει τιμή 2 και η παράσταση (j | m) & (j^m) έχει τιμή 138 & 130 δηλαδή 130.

ii)Έστω ότι προηγούνται οι ορισμοί unsigned char j=69, k1; char m=-10; unsigned int k2; τότε j=01000101_2, m=11110110_2 οπότε:

η παράσταση j << 1 έχει τιμή 10001010_2=138

η παράσταση j << 2 έχει τιμή 00010100_2=20 άν αποδοθεί στην μεταβλητή $k1$ (λόγω προσαρμογής στον τύπο της) αλλά έχει τιμή 100010100_2=276 άν αποδοθεί στην μεταβλητή $k2$ (ΓΙΑΤΙ?)

η παράσταση j >> 1 έχει τιμή 00100010_2=34

η παράσταση m >> 2 έχει τιμή 11111101_2=-3

vii) Οι συγκρίσεις αριθμητικών τιμών "$==, !=, >, >=, <, <=$" και λογικής συσχέτισης τιμών "!, &&, ||" δίνουν παραστάσεις που έχουν ώς τιμή κάποιον απο τους αριθμούς 0 ή 1 (δηλαδή είτε ειναι αληθής είτε ψευδής). Βασικά χρησιμοποιούνται στην σύνθεση συνθηκών. Πιο συγκεκριμένα άν A, B ειναι παραστάσεις τότε:

. Η τιμή της παράστασης A == B ειναι αληθής αν και μόνο οι A, B έχουν την ίδια τιμή.

. Η τιμή της παράστασης A != B ειναι αληθής αν και μόνο οι A, B έχουν διαφορετική τιμή.

. Η τιμή της παράστασης A > B ειναι αληθής αν και μόνο η A έχει τιμή μεγαλύτερη απο την B.

. Η τιμή της παράστασης A >= B ειναι αληθής αν και μόνο η A έχει τιμή μεγαλύτερη ή ίση απο την B.

. Η τιμή της παράστασης A < B ειναι αληθής αν και μόνο οι A έχει τιμή μεγαλύτερη απο την B.

. Η τιμή της παράστασης A <=B ειναι αληθής αν και μόνο οι A έχει τιμή μικρότερη ή ίση απο την B.

. Η τιμή της παράστασης A && B ειναι αληθής αν και μόνο οι A και B ειναι αληθείς .

. Η τιμή της παράστασης A || B ειναι αληθής αν και μόνο τουλάχιστον μία απο τις A, B ειναι αληθής.

. Η τιμή της παράστασης !A ειναι αληθής αν και μόνο η A ειναι ψευδής.

ΠΑΡΑΔΕΙΓΜΑΤΑ :

i) Η παράσταση (A > B) && !Γ αληθεύει όταν οι παραστάση A έχει τιμή μεγαλύτερη απο την B και η παράσταση Γ ειναι ψευδής.

ii) Η απόδοση τιμής j= (A > B) && !C ; δίνει στην μεταβλητή j την τιμή 1 ή 0.

iii) Εκμεταλευόμενοι την αριθμητική τιμή που έχουν οι έννοιες αληθές-ψευδές στην C, μπορούμε να περιγράψουμε μιά απόδοση τιμής της μορφής

$$f = \begin{cases} 5.2 & \text{άν } x \in [2,3), \\ 6.3 & \text{αλλιώς,} \end{cases}$$

(που εξαρτάται απο την τιμή της μεταβλητής x), με την σύνταξη

$$f = ((2 <= x)\&\&(x < 3)) * 5.2 + ((x < 2)||(3 <= x)) * 6.3;$$

viii) Άν Σ και Α, Β ειναι παραστάσεις, τότε μιά παράσταση της μορφής **(Σ)** ? **A : B** έχει την τιμή της παράστασης Α όταν η συνθήκη Σ ειναι αληθής, αλλιώς έχει την τιμή της παράστασης Β.
ΠΑΡΑΔΕΙΓΜΑ : Η απόδοση τιμής στο προηγούμενο παράδειγμα περιγράφεται και με την σύνταξη

$$f = ((2 <= x)\&\&(x < 3))?5.2 : 6.3;$$

ix) Όταν αποδίδουμε σε μια μεταβλητή την τιμή που ήδη έχει αφού πραχθεί με μια παράσταση, (δηλαδή αποδόσεις της μορφής

ΜΕΤΑΒΛΗΤΗ = ΜΕΤΑΒΛΗΤΗ ΠΡΑΞΗ ΠΑΡΑΣΤΑΣΗ

όπου ΠΡΑΞΗ ειναι οποιαδήποτε Αριθμητική πράξη ή Λογική πράξη ή πράξη ολίσθησης), τότε μπορούμε να χρησιμοποιήσουμε συντομογραφίες. Πιο συγκεκριμένα :

Απόδοση τιμής	Συντομογραφία
ΜΕΤΑΒΛΗΤΗ = ΜΕΤΑΒΛΗΤΗ + ΠΑΡΑΣΤΑΣΗ	ΜΕΤΑΒΛΗΤΗ += ΠΑΡΑΣΤΑΣΗ
ΜΕΤΑΒΛΗΤΗ = ΜΕΤΑΒΛΗΤΗ − ΠΑΡΑΣΤΑΣΗ	ΜΕΤΑΒΛΗΤΗ −= ΠΑΡΑΣΤΑΣΗ
ΜΕΤΑΒΛΗΤΗ = ΜΕΤΑΒΛΗΤΗ * ΠΑΡΑΣΤΑΣΗ	ΜΕΤΑΒΛΗΤΗ *= ΠΑΡΑΣΤΑΣΗ
ΜΕΤΑΒΛΗΤΗ = ΜΕΤΑΒΛΗΤΗ / ΠΑΡΑΣΤΑΣΗ	ΜΕΤΑΒΛΗΤΗ /= ΠΑΡΑΣΤΑΣΗ
ΜΕΤΑΒΛΗΤΗ = ΜΕΤΑΒΛΗΤΗ % ΠΑΡΑΣΤΑΣΗ	ΜΕΤΑΒΛΗΤΗ %= ΠΑΡΑΣΤΑΣΗ
ΜΕΤΑΒΛΗΤΗ = ΜΕΤΑΒΛΗΤΗ & ΠΑΡΑΣΤΑΣΗ	ΜΕΤΑΒΛΗΤΗ &= ΠΑΡΑΣΤΑΣΗ
ΜΕΤΑΒΛΗΤΗ = ΜΕΤΑΒΛΗΤΗ | ΠΑΡΑΣΤΑΣΗ	ΜΕΤΑΒΛΗΤΗ |= ΠΑΡΑΣΤΑΣΗ
ΜΕΤΑΒΛΗΤΗ = ΜΕΤΑΒΛΗΤΗ ^ ΠΑΡΑΣΤΑΣΗ	ΜΕΤΑΒΛΗΤΗ ^= ΠΑΡΑΣΤΑΣΗ
ΜΕΤΑΒΛΗΤΗ = ΜΕΤΑΒΛΗΤΗ >> ΠΑΡΑΣΤΑΣΗ	ΜΕΤΑΒΛΗΤΗ >>= ΠΑΡΑΣΤΑΣΗ
ΜΕΤΑΒΛΗΤΗ = ΜΕΤΑΒΛΗΤΗ << ΠΑΡΑΣΤΑΣΗ	ΜΕΤΑΒΛΗΤΗ <<= ΠΑΡΑΣΤΑΣΗ

Τέλος θα δούμε αργότερα ότι ειναι πολύ εύχρηστες και οι συντομογραφίες

. "ΜΕΤΑΒΛΗΤΗ ++ " που περιγράφει την μοναδιαία αύξηση της τιμής που ήδη έχει η ΜΕΤΑΒΛΗΤΗ, δηλαδή ειναι συντομογραφία της απόδοσης "ΜΕΤΑΒΛΗΤΗ = ΜΕΤΑΒΛΗΤΗ + 1".

. "ΜΕΤΑΒΛΗΤΗ - - " που περιγράφει την μοναδιαία μείωση της τιμής που ήδη έχει η ΜΕΤΑΒΛΗΤΗ, δηλαδή ειναι συντομογραφία της απόδοσης "ΜΕΤΑΒΛΗΤΗ = ΜΕΤΑΒΛΗΤΗ - 1".

ΠΑΡΑΤΗΡΗΣΗ: Οι συντομογραφίες ++, - - όταν περιέχονται σε αποδόση τιμής έχουν πιο πολύπλοκη ερμηνεία π.χ. αν j, k ειναι μεταβλητές, τότε :
η απόδοση τιμής j = k++ ειναι συντομογραφία της παράστασης (j=k, k=k+1) (δηλαδή πρώτα η τιμή της k αποδίδεται στην j και έπειτα αυξάνει κατα ένα),
ενώ η απόδοση τιμής j = ++k ειναι συντομογραφία της παράστασης j=k=k+1 (δηλαδή πρώτα η τιμή της k αυξάνει κατα ένα και έπειτα αποδίδεται στην j).

x) Εξαναγκασμένη προσαρμογή τύπου : Υπάρχουν περιπτώσεις όπου η τελική τιμή μιάς παράστασης απαιτείται να ειναι συγκεκριμένου τύπου. Προκειμένου να μετατρέψουμε τον τύπο μιάς τιμής χρησιμοποιούμε το συντακτικό

(ΤΥΠΟΣ) ΤΙΜΗ

όπου, ΤΥΠΟΣ ειναι όνομα τύπου ίσως με προσδιοριστή (π.χ. char, unsigned int, unsigned char).

ΠΑΡΑΔΕΙΓΜΑΤΑ :

i) Άν η μεταβλητή x ειναι τύπου double τότε :
Η παράσταση (int) x έχει τιμή τον ακέραιο που προκύπτει απο την τιμή της x άν αποκοπούν τα δεκαδικά ψηφία. Ειναι τύπου int.
Η παράσταση (int)(x+0.5) έχει τιμή τον κοντινότερο ακέραιο στην τιμή που έχει η x. Ειναι τύπου int.
ii) Το σύμβολο "/" χρησιμοποιείται για τον υπολογισμό του πηλίκου δύο αριθμών. Αν και οι δύο αριθμοί ειναι ακέραιου τύπου, τότε η τιμή αυτής της πράξης θα ειναι επίσης ακέραιου τύπου, ουσιασικά δηλαδή θα ειναι **το πηλίκο της Ευκλείδιας διαίρεσης** των δύο αριθμών. Στην περίπτωση που θέλουμε να πάρουμε το πηλίκο της διαίρεσης των αντίστοιχων πραγματικών αριθμών, πρέπει να κάνουμε εξαναγκασμένη προσαρμογή τύπου, τουλάχιστον σε έναν απο αυτούς (συνήθως στον παρανομαστή).

ΠΑΡΑΔΕΙΓΜΑ :

```
double  x,  y;
int i=4,  j=8;
x=i/j, y=i/3;
```

Σ' αυτό το παράδειγμα, επειδή οι παραστάσεις i/j και i/3 έχουν τιμή ακέραιου τύπου (γιατί στις εμφανιζόμενες πράξεις εμπλέκονται μόνο ακέραιοι), αποκόπτονται τα δεκαδικά ψηφία απο τα παραγόμενα πηλίκα 0.5 και 1.3.. και έπειτα το αποτέλεσμα προσαρμόζεται στον τύπο των μεταβλητών κατα την απόδοση της τιμής. Άρα οι τιμές που παίρνουν οι μεταβλητές x, y ειναι 0 και 1.0 αντίστοιχα!

Άν θέλαμε να αποδώσουμε το πηλίκο της διαίρεσης των αντίστοιχων πραγματικών, έπρέπε να χρησιμοποιήσουμε απόδοση τιμής στην μορφή `x=i/(double)j, y=i/(double)3;` (στην δεύτερη περίπτωση μπορούμε απλά να χρησιμοποιήσουμε παράσταση σταθεράς με δεκαδικό μέρος δηλαδή y=i/3.0).

2) **ΑΠΟΔΟΣΗ ΤΙΜΗΣ ΑΠΟ ΤΟ ΠΛΗΚΤΡΟΛΟΓΙΟ - ΕΚΤΥΠΩΣΗ ΤΙΜΗΣ ΣΤΗΝ ΟΘΟΝΗ**

(Υπο διαμόρφωση η περιγραφή των εντολών printf, scanf.....)

3) **ΣΥΝΑΡΤΗΣΕΙΣ :** Πρόκειται για μιά ειδική κατηγορία μεταβλητών που περιληπτικά, διαφέρουν απο τις απλές μεταβλητές στα εξής σημεία:

α) Οι απλές μεταβλητές μπορούν να ορίζονται μέσα σε block ενώ οι συναρτήσεις ορίζονται ΜΟΝΟ εξω απο κάθε block.

β) Ο ορισμός τους γίνεται με ένα **ονοματισμένο block**, δηλαδή ειναι πιό σύνθετος απο των απλών μεταβλητών.

γ) Στις μεταβλητές αποδίδεται τιμή μέσω παραστάσεων και του συμβολου "=", ενώ στις συναρτήσεις η τιμή παράγεται απο τις εντολές που περιέχει το ονοματισμένο block του ορισμού των.

Ουσιαστικά οι συναρτήσεις ειναι μεταβλητές στις οποίες ο τρόπος ερμηνείας του περιεχομένου των (η τιμή τους δηλαδή) δεν ειναι κάποιος απο τους τυποποιημένους, απρόσημου ακεραίου, προσημασμένου ακεραίου, πραγματικού αριθμού, αλλά περιγράφεται απο τον χρήστη.

Τα χαρακτηριστικά των συναρτήσεων αναλυτικότερα έχουν ώς εξής:

ΟΡΙΣΜΟΣ : Μια συνάρτηση ορίζεται με ένα ονοματισμένο block που βρίσκεται στο τμήμα επεξεργασίας έξω απο κάθε block. Η γενική μορφή αυτών των block ειναι:

ΠΡΟΣΔΙΟΡΙΣΤΗΣ ΤΥΠΟΣ ΟΝΟΜΑ(ΤΥΠΟΣ$_1$ ΜΕΤΑΒΛΗΤΗ$_1$, ... , ΤΥΠΟΣ$_k$ ΜΕΤΑΒΛΗΤΗ$_k$)

{
....
....
}

Block εντολών.

ΟΝΟΜΑ : Η επιλογή του ΟΝΟΜΑΤΟΣ στον ορισμό της συνάρτησης, υπόκειται στους ίδιους συντακτικούς κανόνες με των μεταβλητών.

ΠΑΡΑΔΕΙΓΜΑΤΑ: i) Η βιβλιοθήκη μαθηματικών συναρτήσεων της C, περιέχει ενα μεγάλο πλήθος των γνωστών μαθηματικών συναρτήσεων δίνοντάς τους παρεμφερή ονόματα, πχ cos, sin, tan, sqrt, pow, ...

ii) Η συνάρτηση main υπάρχει σε όλα τα προγράμματα της C γιατί έχει τυποποιηθεί να ορίζει το αρχικό σημείο εκτέλεσης σε οποιοδήποτε πρόγραμμα.

ΤΥΠΟΣ : Ο ΠΡΟΣΔΙΟΡΙΣΤΗΣ και ο ΤΥΠΟΣ στον ορισμό της συνάρτησης, ειναι κάποιες απο τις κωδικές λέξεις περιγραφής τύπων που είδαμε στις απλές μεταβλητές. Η σκοπιμότητα αυτών ειναι ίδια, δηλαδή τόσο ο προσδιορισμός του μέγεθους μνήμης που απαιτείται για την αποθήκευση της τιμής όσο και η ερμηνεία του περιεχομένου αυτής της μνήμης προκειμένου να ανακτάται η τιμή της. Επιπλέον για τις συναρτήσεις υπάρχει και ο τύπος **void**. Αυτός χρησιμοποιείται για τον ορισμό συναρτήσεων που δεν επιστρέφουν τιμή! Η κλήση αυτών των συναρτήσεων γίνεται για το λειτουργικό μέρος που περιγράφει ο ορισμός τους πχ ενώ η κλήση της συνάρτησης $\cos(x)$ γίνεται για να υπολογισθεί το συνημίτονο της γωνίας x ακτινίων και αυτή η τιμή να αφεθεί στην μεταβλητή cos, η κλήση της συνάρτησης printf ουσιαστικά γίνεται για να εκτυπωθεί κάτι στην οθόνη και όχι για να πάρει τιμή η μεταβλητή printf. Παρολαυτά να σημειωθεί ότι το μέγεθος μνήμης που κρατιέται για μιά τιμή τύπου **void** ειναι ίση με το μέγιστο μέγεθος απο όλους τους τύπους!

ΠΑΡΑΤΗΡΗΣΗ: Στον ορισμό μιάς συνάρτησης επιτρέπεται η παράληψη των ΠΡΟΣΔΙΟΡΙΣΤΗΣ ΤΥΠΟΣ ! Σε αυτήν την περίπτωση η συνάρτηση θεωρείται αυτόματα ότι ειναι τύπου void.

ΠΑΡΑΔΕΙΓΜΑ: Είδαμε ότι η συνάρτηση main μπορεί να οριστεί απλά στην μορφή main( ), οπότε σε αυτήν την περίπτωση θεωρείται απο τον μεταγλωττιστή ώς τύπου void.

ΕΜΒΕΛΕΙΑ : Η εμβέλεια μιάς συνάρτησης καθορίζεται όπως και στις απλές μεταβλητές, δηλαδή ειναι το τμήμα του κώδικα αμέσως μετά απο την επικεφαλίδα του ορισμού της. Ωστόσο στις συναρτήσεις δεν υπάρχει περιορισμός της εμβέλειας αφού αυτές ορίζονται μόνο έξω απο κάθε block. Άρα ΔΕΝ ειναι δυνατόν να υπάρχουν συναρτήσεις με ίδιο όνομα. (*Υπενθύμιση: μεταβλητές με ίδιο όνομα μπορούν να υπάρχουν και ίσως να ειναι και διαφορετικού τύπου, αρκεί να βρίσκονται σε διαφορετικά block*).

Προκειμένου να επεκτείνουμε την εμβέλεια μιάς συνάρτησης σε τμήμα του κώδικα που βρίσκεται πρίν απο τον ορισμό της, μπορούμε να χρησιμοποιήσουμε στο αρχικό τμήμα ορισμού μεταβλητών μιά **δήλωση** της μορφής

ΠΡΟΣΔΙΟΡΙΣΤΗΣ ΤΥΠΟΣ ΟΝΟΜΑ(ΤΥΠΟΣ$_1$, ΤΥΠΟΣ$_2$, ... , ΤΥΠΟΣ$_k$);

γράφοντας δηλαδή την επικεφαλίδα του ονοματισμένου block ορισμού της, και μάλιστα για τις μεταβλητές απο τις οποίες εξαρτάται κρατούμε μόνο τον τύπο τους, όχι τα ονόματα τους (τέτοιες δηλώσεις ονομάζονται **αρχέτυπα** (*prototypes*)) .
ΠΑΡΑΤΗΡΗΣΕΙΣ : i) Η δήλωση μιάς συνάρτησης τερματίζει σε ερωτηματικό, ειναι αυτοτελής εντολή προς τον μεταγλωττιστή! Αντίθετα, η επικεφαλίδα στον ορισμό μιάς συνάρτησης δεν τερματίζει σε ερωτηματικό, δεν ειναι εντολή αλλά τμήμα των αγκύλων { } του block ορισμού της.
ΠΑΡΑΔΕΙΓΜΑ:Τα αρχέτυπα όλων των μαθηματικών συναρτήσεων της C, βρίσκονται στο αρχείο /usr/include/math.h πχ εκεί υπάρχουν δηλώσεις της μορφής:

```
double cos( double);
double sin( double);
double sqrt(double);
double pow(double, double);
.......
```

Προκειμένου να χρησιμοποιήσουμε αυτές τις συναρτήσεις, ειναι απαραίτητο να συμπεριλάβουμε τα αρχέτυπά τους στο τμήμα ορισμού μεταβλητών, ώστε να επεκταθεί η εμβέλειά τους και στον δικό μας κώδικα. Αυτό επιτυγχάνεται με την οδηγία του προμεταγλωττιστή #include <math.h>, μέσω της οποίας όλο το αρχείο /usr/include/math.h ενσωματώνεται στον δικό μας κώδικα κατα την φάση της προμεταγλώττισης.
Οι ορισμοί αυτών των συναρτήσεων, βρίσκονται σε μορφή αντικειμενικού κώδικα στο αρχείο /usr/lib/libm.so το οποίο για να ενσωματωθεί απαιτείται το option -lm στην εντολή μεταγλώττισης.
ii) *Η εμβέλεια μιάς συνάρτησης ειναι το τμήμα του κώδικα αμέσως μετά απο την επικεφαλίδα του ορισμού της* σημαίνει ότι **επιτρέπεται να γίνει αναφορά της ίδιας της συνάρτησης μέσα στο block ορισμού της** (οπότε επιτρεπονται ορισμοί **αναδρομικού** τύπου)!

ΤΙΜΗ: Μιά συνάρτηση παίρνει τιμή με αναφορά του ονόματός της συνοδευόμενη απο παρενθέσεις που περιέχουν τιμές για όλες τις ΜΕΤΑΒΛΗΤΗ$_1$, ... , ΜΕΤΑΒΛΗΤΗ$_k$ του ορισμού της.
ΠΑΡΑΤΗΡΗΣΗ: Επιτρέπεται η αναφορά συναρτήσεων μέσα σε παραστάσεις αφού αυτές ειναι μεταβλητές. Σε αυτές τις περιπτώσεις, προηγείται ο υπολογισμός των τιμών των συναρτήσεων έναντι όλων των υπολοίπων πράξεων της παράστασης .
ΠΑΡΑΔΕΙΓΜΑΤΑ:

```
i) double y=cos( 3.1415*2-0.5);  ( Στην μεταβλητή y αποδίδεται η τιμή της μεταβλητής cos. Επειδή η cos
                                   ειναι συνάρτηση, πρώτα θα αποδοθεί στην μεταβλητή απο την οποία
                                   εξαρτάται η τιμή της παράστασης 3.1415*2-0.5 και στην συνέχεια
                                   θα υπολογισθεί η τιμή της με το block εντολών του ορισμού της.)
ii) int j=1;                     ( Όπως πρίν αλλά πρώτα θα αποδοθεί η τιμή 2 στην μεταβλητή απο την
    double y=cos( 2*j);            οποία εξαρτάται η cos, οπότε πρώτα θα προσαρμοστεί στον τύπο της !
                                   ΠΡΟΣΟΧΗ, ειναι προτιμότερο να ελέγχουμε τέτοιες προσαρμογές τύπου,
                                   πχ ειναι προτιμότερο να δίναμε την εντολή y=cos( (double) 2*j) )
iii) double t=1.5, r=pow( 3*t);  ( Μή αποδεκτή αναφορά της pow, απαιτείται να δίνουμε τιμές σε όλες τις
                                   μεταβλητές εξάρτησής της, εδώ απαιτούνται 2 τιμές x, y για να υπολογιστεί
                                   απο την pow η τιμή $x^y$.)
```

Η τελική τιμή της συνάρτησης ειναι του ΤΥΠΟΥ που περιγράφει ο ορισμός της και υπολογίζεται με το block εντολών του ορισμού της, αφού πάρουν τιμές η ΜΕΤΑΒΛΗΤΗ$_1$, ... , ΜΕΤΑΒΛΗΤΗ$_k$. Μέσα στο block ορισμού της, ο υπολογισμός της τιμής τερματίζει στην πρώτη εντολή της μορφής

return ΠΑΡΑΣΤΑΣΗ;

οπότε, ώς τιμή της συνάρτησης θεωρείται η τιμή αυτής της ΠΑΡΑΣΤΑΣΗΣ (αφού προσαρμοστεί στον τύπο της συνάρτησης). Στις συναρτήσεις τύπου void όπου δεν ενδιαφερόμαστε για την τιμή της συνάρτησης, αρκεί η εντολή *return;* ή και καθόλου ύπαρξη της στο block ορισμού της.
ΠΑΡΑΤΗΡΗΣΗ: Συντακτικά επιτρέπεται μιά εντολή της μορφής *return;* για συνάρτηση οποιουδήποτε τύπου αλλά σε αυτήν την περίπτωση η τιμή της συνάρτησης ειναι απροσδόριστη!

ΠΑΡΑΔΕΙΓΜΑΤΑ:

i) double NormComplex(double x, double y)
{ double x2=x*x, y2=y*y;
return sqrt(x2+y2);
printf("Ayto den typwthke!! "); }

(Ορίζεται η συνάρτηση NormComplex τύπου double, η οποία εξαρτάται απο δύο τιμές τύπου double (αποδίδονται αρχικά στις μεταβλητές x, y). Η τιμή της συνάρτησης που παράγει αυτό το block εντολών ειναι $\sqrt{x^2+y^2}$.

Όλες οι εντολές μετά την *return* δεν θα εκτελεστούν. Οποιαδήποτε αναφορά στην NormComplex μπορεί να γίνει μετά απο την επικεφαλίδα του ορισμού της. Άν χρειαστεί επέκταση της εμβέλειάς της, πρέπει στο αρχικό τμήμα ορισμού μεταβλητών να δηλωθεί το αρχέτυπό της *double NormComplex(double, double);*

ii) int factorial(int m)
{ return m*factorial(m-1); }

Εδώ ορίζεται η συνάρτηση factorial τύπου int, η οποία εξαρτάται απο μία τιμή τύπου int που αποδίδεται αρχικά στην μεταβλητή m. Η τιμή της συνάρτησης που παράγει αυτό το block εντολών, προκύπτει ύστερα απο μιά κλήση του εαυτού της αλλά με τιμή παραμέτρου κατα 1 μικρότερη, δηλαδή αναδρομικά.
Θα μελετήσουμε την συμπεριφορά αυτής της συνάρτησης με ένα παράδειγμα.
Έστω ότι σε κάποιο σημείο του κώδικα μέσα στην εμβέλεια της factorial, επιχειρούμε την απόδοση τιμής j=factorial(5);
Προκειμένου να βρεθεί η τιμή της δεξιάς παράστασης, θα εκτελεστεί πρώτα ο ορισμός της factorial με τιμή παραμέτρου 5, ουσιαστικά δηλαδή πρέπει να βρεθεί η τιμή της παράστασης 5*factorial(4) .
Άρα, προκειμένου να βρεθεί η τιμή της δεξιάς παράστασης, θα εκτελεστεί πρώτα ο ορισμός της factorial με τιμή παραμέτρου 4, ουσιαστικά δηλαδή πρέπει να βρεθεί η τιμή της παράστασης 5*(4*factorial(3)) .
Άρα, προκειμένου να βρεθεί η τιμή της δεξιάς παράστασης, θα εκτελεστεί πρώτα ο ορισμός της factorial με τιμή παραμέτρου 3, ουσιαστικά δηλαδή πρέπει να βρεθεί η τιμή της παράστασης 5*(4*(3*factorial(2))) .
.....
Η διαδικασία αυτή θα επαναλαμβάνεται συνεχώς. Σε κάθε βήμα θα μεγενθύνεται εσωτερικά η παράσταση με έναν επιπλέον πολλαπλασιασμό και μια κλήση της factorial, όμως κανένας πολλαπλασιασμός δεν θα εκτελείται αφού προτεραιότητα έχει πάντα ο υπολογισμός της συνάρτησης. Σε κάποιο βήμα η παράσταση που θα έχει δημιουργηθεί, θα υπερβεί τον μέγιστο αριθμό πράξεων που επιτρέπεται να περιέχει, οπότε η εκτέλεση του προγράμματος θα διακοπεί.

ΣΥΜΠΕΡΑΣΜΑ : Ένας αναδρομικός ορισμός συνάρτησης πρέπει πάντα να περιέχει μιά συνθήκη τερματισμού της ανάκλησης.

ΠΑΡΑΤΗΡΗΣΗ: Στο παραπάνω παράδειγμα, έγινε προσπάθεια να υπολογιστεί το $n!$ ("παραγοντικό του φυσικού n"), δηλαδή η τιμή $1 \cdot 2 \cdot \ldots n$ όταν $n > 0$, ή 1 όταν $n = 0$. Άμεσα παρατηρούμε ότι αυτή η συνάρτηση μπορεί να ορισθεί αναδρομικά απο τον τύπο:

$$n! = \begin{cases} 1, & \text{όταν } n = 0 \\ n \cdot (n-1)!, & \text{αλλιώς} \end{cases}$$

οπότε, στην υλοποίηση της σε υπολογιστικό κώδικα απαιτείται κάποια εντολή που να αποδίδει τιμή υπο συνθήκη, πχ ο σωστός ορισμός θα ήταν return (m)?m*factorial(m-1):1;
Θα δούμε αργότερα πιό ευανάγνωστες εντολές που ελέγχουν την ροή επεξεργασίας.

4) **ΔΙΑΝΥΣΜΑΤΙΚΕΣ ΜΕΤΑΒΛΗΤΕΣ :** Στα Μαθηματικά αλλά και σε όλες τις επιστήμες, μελετώνται μεγέθη που προσδιορίζονται με περισσότερους απο έναν αριθμό. Για παράδειγμα, ενώ η θερμοκρασία θ ενός σώματος προσδιορίζεται με έναν αριθμό, η θέση x ενός σημείου στον χώρο προσδιορίζεται με τρείς αριθμούς. Αυτή η συσχέτιση διατυπώνεται στην μορφή $x = (x_1, x_2, x_3)$, δηλαδή ταυτίζουμε την θέση x με ενα **διάνυσμα** 3 αριθμών. Η θέση στον χώρο ειναι **διανυσματικό μέγεθος διάστασης** 3.
ΠΑΡΑΤΗΡΗΣΗ: Για λόγους οικονομίας στα ονόματα, όλα τα **στοιχεία** (ή **συνιστώσες**) ενός διανυσματικού μεγέθους, έχουν ίδιο όνομα με του μεγέθους. Για να ξεχωρίζουν, τα δεικτοθετούμε με τον αριθμό της σειράς που εμφανίζονται στο διάνυσμα.
Ανάλογα ορίζονται οι **διανυσματικές μεταβλητές** στην γλώσσα C.

ΣΥΝΤΑΚΤΙΚΟΣ ΚΑΝΟΝΑΣ : Όταν στον ορισμό μιάς μεταβλητής, τοποθετήσουμε στα δεξιά του ονόματος της έναν φυσικό n>0 κλεισμένο μέσα σε τετραγωνικές αγκύλες "[", "]", τότε αυτή θεωρείται *διανυσματική μεταβλητή διάστασης n*. Με αυτήν την σύνταξη ουσιαστικά ορίζονται n διαδοχικές μεταβλητές ίδιου τύπου, μία για κάθε συνιστώσα της διανυσματικής μεταβλητής. Όλες έχουν το ίδιο όνομα, η αναφορά σε μία απο αυτές γινεται βάζοντας μέσα σε τετράγωνικές αγκύλες "[", "]"τον αριθμό θέσης της, αρχίζοντας την αρίθμηση απο το 0.

Ο κανόνας αυτός μπορεί να εφαρμοστεί επανειλημμένες φορές προκειμένου να παραχθούν ακόμα πιό σύνθετες μεταβλητές!

ΠΑΡΑΔΕΙΓΜΑΤΑ :

i) int x, sd[3];
Εδώ ορίζεται μιά μεταβλητή τύπου int με όνομα x και μιά διανυσματική μεταβλητή διάστασης 3, με όνομα sd. Άρα, για την sd θα κρατηθεί συνεχόμενος χώρος μνήμης για 3 συνιστώσες μεταβλητές τύπου int, στις οποίες μπορούμε να αναφερόμαστε χρησιμοποιώντας τα ονόματα sd[0], sd[1], sd[2].

ii) char ar[5][2];
Εδώ ορίζεται μια διανυσματική μεταβλητή διάστασης 5 με όνομα ar. Γι' αυτήν θα κρατηθεί συνεχόμενος χώρος μνήμης για 5 συνιστώσες μεταβλητές στις οποίες μπορούμε να αναφερόμαστε χρησιμοποιώντας τα ονόματα ar[0], ar[1], ..., ar[4]. Καθεμιά απο αυτές ειναι διάνυσματική μεταβλητή διάστασης 2, επειδή έπεται και άλλη αγκύλη στον ορισμό. Οπότε, για κάθεμιά απο αυτές θα κρατηθεί συνεχόμενος χώρος μνήμης για 2 συνιστώσες μεταβλητές τύπου char στις μπορούμε να αναφερόμαστε χρησιμοποιώντας τα ονόματα ar[0][0], ar[0][1], ar[1][0], ar[1][1], ..., ar[4][0], ar[4][1]. Σε αυτό το παράδειγμα, η ar ήταν διάνυσμα απο διανύσματα μεταβλητών τύπου char, για συντομία θα αναφέρεται ώς **πίνακάς 5 γραμμών επι 2 στηλών**, μεταβλητών τύπου char.

iii) double tr[5][2][3]; Εδώ έχουμε ένα διάνυσμα απο διανύσματα διανυσμάτων .... Για συντομία θα αναφερόνταν ώς διάνυσμα διάστασης 5, απο πίνακες 2 γραμμών επι 3 στηλών, μεταβλητών τύπου double ! Συνολικά θα κρατηθεί χώρος για $5 \cdot 2 \cdot 3 = 30$ μεταβλητές τύπου double . Για να αναφερθούμε σε μία απο αυτές πρέπει να δώσουμε τους 3 αριθμούς που προσδιορίζουν την θέση της μέσα στην tr.

ΠΑΡΑΤΗΡΗΣΕΙΣ :

1) **Μιά εντολή της μορφής "ΜΕΤΑΒΛΗΤΗ=ΤΙΜΗ" περιγράφει απόδοση τιμής σε απλή ΜΕΤΑΒΛΗΤΗ, ποτέ σε διανυσματική.**
ΠΑΡΑΔΕΙΓΜΑ :
int x[3], y[3]; (Ορίζονται δύο διανύσματα διάστασης 3, με συνιστώσες μεταβλητές τύπου int)
x[0]=0, x[1]=1, x[2]=2; (Αποδίδονται τιμές στα στοιχεία της πρώτης, ειναι σωστή σύνταξη αφου τα στοιχεία αυτά ειναι απλές μεταβλητές)
y=x; (Επιχειρείται μια απόδοση διανυσματικής τιμής ΕΙΝΑΙ ΛΑΘΟΣ ! Η μεταφορά των τιμών πρέπει να γίνει με ξεχωριστή απόδοση τιμής για κάθε συνιστώσα, δηλαδή y[0]=x[0], y[1]=x[1], y[2]=x[2].))
Ο μόνος τρόπος για να αποδόσουμε τιμές ταυτόχρονα στις συνιστώσες μεταβλητές, μιάς διανυσματικής μεταβλητής, ειναι στον ορισμό της (**αρχικές τιμές**). **Για να γίνει αυτό δίνουμε μια λίστα με τις τιμές χωρισμένες με κόμμα και κλεισμένες μέσα σε άγκιστρα. ΠΡΟΣΟΧΗ: Κάθε λίστα αρχικών τιμών πρέπει να ειναι μικρότερης ή ίσης διάστασης με του διανύσματος που θα αποδοθούν.**

ΠΑΡΑΔΕΙΓΜΑΤΑ :

i) int y=3, x[4]={ 2, 2*y, 2.5}; (Ορίζεται το διάνυμα x διάστασης 4, μεταβλητών τύπου int. Κάποιες συνιστώσες μεταβλητές παίρνουν τις αρχικές τιμές x[0]=2, x[1]=6, x[2]=2, ενώ η x[3] έχει άγνωστη αρχική τιμή.)

ii) double x[2]={ 0, -1, 1}; (Επιχειρείται μια ΛΑΘΟΣ απόδοση αρχικών τιμών! Η λίστα τιμών ειναι μεγαλύτερης διάστασης απο του διανύσματος!)

iii) int m[5][2]={ { 0, 1 }, { 2 }, { 4, 5 } } ;

(Εδώ ορίζεται ένας πίνακας 5 γραμμών επι 2 στηλών, μεταβλητών τύπου int. Θέλοντας να αποδόσουμε αρχικές τιμές, πρέπει να προσέξουμε και την διάσταση των συνιστωσών μεταβλητών. Επειδή ουσιαστικά η διανυσματική μεταβλητή m περιέχει διανύσματα διάστασης 2, οι αρχικές τιμές για αυτά πρέπει να ειναι λίστες των 2 τιμών το πολύ. Στο παράδειγμα αυτό δόθηκαν μόνο οι εξής αρχικές τιμές
m[0][0]=0, m[0][1]=1, m[1][0]=2, m[2][0]=4, m[2][1]=5.)

Εκμεταλευόμενοι την ελευθερία έκφρασης του συντακτικού της C, η παραπάνω αρχικοποίηση του πίνακα m, ίσως ειναι πιο ευανάγνωστη στην μορφή :

```
int m[5][2]={ { 0, 1 },
              { 2    },
              { 4, 5 } };
```

γιατί μας θυμίζει τις γεωμετρικές δαστάσεις του πίνακα.

iv) Ειδικά τα διανύσματα μεταβλητών τύπου char μπορούν να χρησιμοποιηθούν για την αποθήκευση διαδοχικών γραμμάτων πχ για την αποθήκευση λέξεων, φράσεων κλπ. Για παράδειγμα

```
char Lexh[4]={ 'A', 'L', 'F', 'A' };
```

(Εδώ ορίζεται ένα διάνυμα διάστασης 4, απο μεταβλητές τύπου char. Οι συνιστώσες μεταβλητές πήραν ώς αρχικές τιμές τους κώδικες ASCII των γραμμάτων της λέξης ALFA.)

Επειδή διαννύσματα χαρακτήρων συναντώνται συχνά στις εφαρμογές, η αρχικοποίηση αυτών των διανυσματικών μεταβλητών διατυπώνεται πιό απλά κλείνοντας την λέξη ή φράση σε διπλά quotes. Στο παράδειγμα αυτό, το ίδιο αποτέλεσμα παράγεται και με την σύνταξη

```
char Lexh[4]="ALFA" ;
```

v) char HMERES[7][9]={ "KYRIAKH", "DEYTERA", "TRITH", "TETARTH", "PEMPTH", "PARASKEYH", "SABBATO" };

Εδώ σε κάθε γραμμή του πίνακα χαρακτήρων HMERES, αποθηκεύονται οι κώδικες ASCII των γραμμάτων της αντίστοιχης ημέρας της εβδομάδος.

ΠΑΡΑΤΗΡΗΣΗ: Ο πίνακας αυτός ορίστηκε να έχει 9 στήλες για να χωρέσουν όλα τα γράμματα της μεγαλύτερου μήκους λέξης (της PARASKEYH), ενώ για τις υπόλοιπες λέξεις ο περισσευόμενος χώρος μένει ανεκμετάλευτος.

2) Ανάλογα με τον τελεστή "=" ισχύουν και για όλες τις πράξεις που συναντήσαμε στην σύνθεση παραστάσεων. **Οι παραστάσεις στην C έχουν πάντα απλή τιμή, ποτέ διανυσματική.**

ΠΑΡΑΔΕΙΓΜΑ :

```
int x[3], y[3];
x+y;
```

(Εδώ, επιχειρείται μια διανυσματική άθροιση (έμμεσα υπονοούνται οι αθροίσεις y[0]+x[0], y[1]+x[1], y[2]+x[2]), ειναι ΛΑΘΟΣ !)

Κατα συνέπεια, επειδή επιτρέπεται η αναφορά συναρτήσεων μέσα σε παραστάσεις, **οι συναρτήσεις δεν γίνεται να ειναι διανυσματικού τύπου**, πχ δεν υπάρχει ορισμός συνάρτησης με επικεφαλίδα της μορφής double func[2][3](ΤΥΠΟΣ$_1$ ΜΕΤΑΒΛΗΤΗ$_1$, ..., ΤΥΠΟΣ$_k$ ΜΕΤΑΒΛΗΤΗ$_k$).

Ωστόσο, οι μεταβλητές απο τις οποίες εξαρτάται μια συνάρτηση, μπορούν να ειναι διανυσματικού τύπου!

ΠΑΡΑΔΕΙΓΜΑΤΑ :

i) void func(int m, int r[1])

(Επικέφαλίδα συνάρτησης που εξαρτάται απο μία μεταβλητή τύπου int και απο μία διανυσματική μεταβλητή **διάστασης 1**, στοιχείων τύπου int. Το αρχέτυπο της ειναι func(int, int [1]);)

ii) void CequalAplusB(double A[2], double B[2], double C[2])

(Επικέφαλίδα συνάρτησης που εξαρτάται απο 3 διανύσματα διάστασης 2, στοιχείων τύπου double . Το αρχέτυπο της ειναι void CequalAplusB(double [2], double [2], double [2]);)

ΣΗΜΕΙΩΣΗ: **Στις διανυσματικές μεταβλητές απο τις οποίες εξαρτάται μία συνάρτηση, δεν μεταβιβάζονται οι τιμές αλλά οι ίδιες οι μεταβλητές**, γι' αυτό και απαιτείται **οι διαστάσεις αυτών των μεταβλητών στον ορισμό να συμφωνούν με τις διαστάσεις των αντίστοιχων μεταβλητών κατα την κλήση της συνάρτησης**.

ΠΑΡΑΔΕΙΓΜΑΤΑ :

i) Για να κατανοήσουμε τί είδους πληροφορία μεταφέρεται σε μια συνάρτηση όταν αυτή εξαρτάται απο διανυσματικές μεταβλητές, θα μελετήσουμε το παρακάτω πρόγραμμα:

```
void func( int m, int r[1] )
{  m=10, r[0]=10; }

main( )
{ int k=5, p[1]={  5};
  printf( "PRIN APO THN KLHSH THS func, k=%d  p[0]=%d\n",  k,  p[0]);
  func( k, p);
  printf( "META APO THN KLHSH THS func, k=%d  p[0]=%d\n",  k,  p[0]); }
```

Παρατηρήστε ότι η διανυσματική μεταβλητή p περιέχει μόνο ένα στοιχείο το p[0], που αρχικά παίρνει την τιμή 5. Ακριβώς την ίδια αρχική τιμή έχει και η απλή μεταβλητή k και αυτό ακριβώς θα τυπώσει η πρώτη printf για αυτές τις μεταβλητές. Μετά όμως απο την κλήση της συνάρτησης μέσω της αναφοράς func(k, p), η δεύτερη printf θα τυπώσει την τιμή 5 για την μεταβλητή k ενώ για την μεταβλητή p[0] θα τυπώσει την τιμή 10 ! Γιατί στην μία μεταβλητή άλλαξε η τιμή ενώ στην άλλη όχι; Αυτό έγινε γιατί στον ορισμό της συνάρτησης func, η m ειναι απλή μεταβλητή, οπότε κατα την κλήση func(k, p) **μεταφέρεται η τιμή της μεταβλητής k στην m** για να χρησιμοποιηθεί εκεί. Αντίθετα επειδή η r ειναι διανυσματική μεταβλητή, κατα την κλήση func(k, p) **μεταφέρεται η διεύθυνση της μεταβλητής p σαν διεύθυνση της μεταβλητής r**, οπότε οποιεσδήποτε αποδόσεις τιμής σε στοιχεία της r ουσιαστικά αλλοιώνουν τα αντίστοιχα στοιχεία της p.

ii)

```
void CequalAplusB( double A[2], double B[2], double C[2])
{  C[0]=A[0]+ B[0],  C[1]= A[1]+B[1]; }

main( )
{ double PIN1[2]={-1, 1 }, PIN2[2]={ 2, 1 }, PIN3[2];
  CequalAplusB( PIN1,  PIN2,  PIN3); }
```

Εδώ η συνάρτηση CequalAplusB εξαρτάται απο τρείς διανυσματικές μεταβλητές A, B, C. Ουσιαστικά παράγει το διανυσματικό άθροισμα των δύο πρώτων και το αποθηκεύει στην τρίτη.
Μέσα στην main το διάνυσμα PIN3 δεν έχει αρχικοποιηθεί με τιμές, όμως μετά απο την κλήση CequalAplusB(PIN1, PIN2, PIN3) τα στοιχεία της PIN3 έχουν πάρει τις τιμές PIN3[0]=1, PIN3[1]=2.

iii)

```
main( )
{ double PIN[3][2]={ {-1, 1 },
                     { 2, 1 } };
  CequalAplusB( PIN[0], PIN[1],  PIN[2]); }
```

Εδώ η PIN ειναι πίνακας 3 γραμμών επι 2 στηλών, δηλαδή περιέχει 3 διανύσματα διάστασης 2, τα PIN[0], PIN[1], PIN[2]. Επειδή οι διαστάσεις αυτών των διανυσμάτων συμφωνούν με των διανυσματικών μεταβλητών A, B, C του ορισμού της συνάρτησης CequalAplusB, έπεται ότι η κλήση CequalAplusB(PIN[0], PIN[1], PIN[2]) ειναι επιτρεπτή. Μετά απο αυτήν την κλήση η τρίτη γραμμή της PIN θα περιέχει το άθροισμα των πρώτων δύο.
Αυτή η διατύπωση ειναι προτιμότερη απο του προηγούμενου παραδείγματος, γιατί χρειάστηκε ένα όνομα μεταβλητής αντί για τρία διαφορετικά, οπότε πιθανές αλλαγές στο πλήθος αυτών των διανυσμάτων μπορούν να περιγραφούν με κατάλληλη #define (κάτι που δεν μπορεί να γίνει άν κάθε μεταβλητή έχει διαφορετικό όνομα).

5) ΕΛΕΓΧΟΣ ΤΗΣ ΡΟΗΣ ΕΠΕΞΕΡΓΑΣΙΑΣ

ΠΑΡΑΤΗΡΗΣΗ: Στα παρακάτω θα μελετήσουμε εντολές που ελέγχουν τη ροής της επεξεργασίας. Όσες απο αυτές έχουν την μορφή

```
   ΕΝΤΟΛΗ
  {
    ....
    ....
  }
```
Block εντολών.

ειναι δυνατόν να αναδιατυπωθούν σε πιο απλή μορφή, εφαρμόζοντας τον παρακάτω κανόνα:
Όταν το Block εντολών περιέχει μία εντολή τότε τα άγκιστρα του Block μπορούν να παραληφθούν, όπου σε γενικές γραμμές, "μία εντολή" σημαίνει ότι περιέχεται μόνο ένα ερωτηματικό.

ΥΠΟ ΣΥΝΘΗΚΗ BLOCK ΕΝΤΟΛΩΝ: Στις αναδρομικές συναρτήσεις, είδαμε ότι στο block ορισμού των, πρέπει να τους αποδίδεται τιμή που να εξαρτάται απο μία συνθήκη. Όταν αληθεύει η συνθήκη τότε ο υπολογισμός της τιμής απαιτεί μια κλήση της ίδιας της συνάρτησης, αλλιώς ειναι ένας σταθερός αριθμός. Γενικότερα, σε οποιοδήποτε σημείο ενός προγράμματος μπορεί να απαιτηθεί έλεγχος της ροής επεξεργασίας αναλόγως του άν αληθεύει μιά συνθήκη Α. Μιά εντολή της μορφής:

```
if( A)
{
  ....
  ....
}
```
Block εντολών.

εξαναγκάζει την ροή επεξεργασίας να παρακάμψει αυτό το Block εντολών άν η συνθήκη Α δεν αληθεύει. Δηλαδή, πρώτα θα βρεθεί η τιμή της παράστασης Α και άν αυτή ειναι $\neq 0$ τότε θα εκτελεστεί αυτό το Block εντολών, αλλιώς η επεξεργασία θα συνεχίσει μετά απο αυτό. Μια πιο σύνθετη περίπτωση περιγράφει η παρακάτω εντολή.

```
if( A)
{
  ....
  ....
}
```
Block1 εντολών.
```
else
{
  ....
  ....
}
```
Block2 εντολών.

Εδώ πρώτα θα βρεθεί η τιμή της παράστασης Α και άν ειναι αληθής θα εκτελεστεί το Block1 εντολών, ενώ άν ειναι ψευδής θα εκτελεστεί το Block2 εντολών (σε καμμία περίπτωση δεν θα εκτελεστούν και τα δύο Block). Επειτα η επεξεργασία θα προχωρήσει μετά απο το Block2.

ΠΑΡΑΔΕΙΓΜΑ: Ο αναδρομικός ορισμός της συνάρτησης factorial μπορεί να διατυπωθεί με εντολή if-else σε μία απο τις παρακάτω μορφές:

ΤΥΠΙΚΑ	ΠΙΟ ΣΥΝΤΟΜΑ

int factorial(int m)
{
if(m)
{ return m*factorial(m-1); }
else
{ return 1; }
}
	int factorial(int m)
	{ if(m) return m*factorial(m-1);
	else return 1; }

Στην αριστερή στήλη εμφανίζεται η τυπική διατύπωση της ζητούμενης ροής επεξεργασίας, ενώ στην δεξιά στήλη εμφανίζεται η ισοδύναμη διατύπωση που προκύπτει ύστερα απο εφαρμογή του απλοποιητικού κανόνα.

ΠΑΡΑΤΗΡΗΣΕΙΣ :

i) Το τμήμα else δεν ειναι απαραίτητο να υπάρχει. αλλά δεν μπορεί να βρίσκεται και μόνο του. **Όταν υπάρχει τότε συμπληρώνει την αμέσως προηγούμενη if** (που πρέπει να υπάρχει). Δηλαδή ένα if(A) Block1 - else Block2 **συγκροτεί συνολικά μια εντολή** (αυτό να λαμβάνεται υπόψιν στον αρχικό απλοποιητικό κανόνα για τα Block αυτών των εντολών).
ΠΑΡΑΔΕΙΓΜΑ : Στην αριστερή στήλη του παρακάτω πίνακα, περιγράφουμε τυπικά μια ζητούμενη ροή επεξεργασίας και έπειτα, στις δεξιότερες στήλες εφαρμόζουμε τον απλοποιητικό κανόνα για να αναχθούμε σε απλούστερες αλλά ισοδύναμες περιγραφές.

ΤΥΠΙΚΑ	ΠΙΟ ΣΥΝΤΟΜΑ	ΠΙΟ ΣΥΝΤΟΜΑ	ΔΙΑΦΟΡΕΤΙΚΗ ΡΟΗ

if(x>3)
{ if(y<2)
{ z=2;
u=3; }
else
{ z=3;
u=2; }
}
	if(x>3)
	{ if(y<2)
	z=2, u=3;
	else
	z=3, u=2; }
		if(x>3)
		if(y<2) z=2, u=3;
		else z=3, u=2;
			if(x>3)
			{ if(y<2) z=2, u=3; }
			else z=3, u=2;

Στο πρώτο απλοποιητικό βήμα χρησιμοποιήσαμε το κόμμα σαν διαχωριστή ώστε τα εσωτερικά Block να περιέχουν μία εντολή οπότε να απλοποιηθούν τα άγκιστρα. Δεδομένου ότι η εσωτερική if-else συγκροτεί μία εντολή, μπορούμε να απλοποιήσουμε πάλι τα άγκιστρα, οπότε να καταλήξουμε στην ισοδύναμη διατύπωση της τρίτης στήλης. Αυτή η διατύπωση δεν οδηγεί σε σύγχιση τον μεταγλωττιστη, αφού η else συμπληρώνει την αμέσως προηγούμενη if.
Άν η else έπρεπε να συμπληρώσει την πρώτη if, τότε απαιτούνται τα άγκιστρα όπως στην διατύπωση της 4 στήλης.

ii) Όταν υπάρχουν πολλές πιθανές επιλογές στην ροή επεξεργασίας, τότε απαιτείται μια ακολουθία απο εντολές if-else ώστε να καλύπτονται όλες οι περιπτώσεις.
ΠΑΡΑΔΕΙΓΜΑ : Θέλοντας να περιγράψουμε την απόδοση τιμής:

$$z = \begin{cases} 1 & , \text{άν } x < 3 \\ 2 & , \text{άν } x \in [3,5] \\ 3 & , \text{άν } x \in (5,6) \\ 4 & , \text{αλλιώς} \end{cases}$$

πρέπει να χρησιμοποιήσουμε πολλές if-else εντολές, όπως στην αριστερή στήλη του παρακάτω πίνακα:

ΤΥΠΙΚΑ	ΠΙΟ ΣΥΝΤΟΜΑ	ΠΙΟ ΣΥΝΤΟΜΑ	ΠΙΟ ΣΥΝΤΟΜΑ	ΤΕΛΙΚΑ

if(x<3)
{ z=1; }
else
{ if(x<=5)
{ z=2; }
else
{ if(x<6)
{ z=3; }
else
{ z=4; }
}
}
	if(x<3)
	{ z=1; }
	else
	{ if(x<=5)
	{ z=2; }
	else
	{ if(x<6) z=3;
	else z=4; }
	}
		if(x<3)
		{ z=1; }
		else
		{ if(x<=5) z=2;
		else if(x<6) z=3;
		else z=4;
		}
			if(x<3) z=1;
			else if(x<=5) z=2;
			else if(x<6) z=3;
			else z=4;
				if(x<3) z=1;
				else if(x<=5) z=2;
				else if(x<6) z=3;
				else z=4;

Ύστερα απο επαναλαμβανόμενες εφαρμογές του απλοποιητικού κανόνα, παρατηρούμε ότι οδηγούμαστε σε μια περιγραφή που περιέχει διατυπώσεις της μορφής **else if(ΣΥΝΘΗΚΗ) Block** . Αυτές αν και δεν ειναι εντολές της C ειναι συνηθισμένα τμήματα σε ροές επεξεργασίας που έχουν περισσοτέρους απο δύο υποσυνθήκη κλάδους, προκύπτουν λόγω του απλοποιητικού κανόνα.

ΕΠΑΝΑΛΗΠΤΙΚΑ BLOCK ΕΝΤΟΛΩΝ : Όπως είδαμε στις διανυσματικές μεταβλητές, επειδή ο τελεστής "="δεν έχει διανυσματικό χαρακτήρα, οι τιμές των συνιστωσών μεταβλητών πρέπει να αποδίδονται σε κάθε μία χωριστά.

ΠΑΡΑΔΕΙΓΜΑ :

```
double x[100], y[100], z[100];
....Διάφορες εντολές.....
z[0]=x[0]+y[0],  z[1]=x[1]+y[1], . . .  z[99]=x[99]+y[99];
```

Σε αυτό το παράδειγμα επαναλαμβάνεται συνεχώς η απόδοση κάποιου αθροίσματος σε στοιχείο διαφορετικής θέσης της διανυσματικής μεταβλητής z, εως ότου όλα τα στοιχεία της να πάρουν τιμές. Γενικότερα, σε οποιοδήποτε σημείο ενός προγράμματος ίσως απαιτηθεί η επανειλημμένη εκτέλεση μιάς ακολουθίας εντολών όσο αληθεύει μιά συνθήκη A. Μιά εντολή της μορφής:

```
while( A)
{
  ....
  ....
}
```

Block εντολών.

περιγράφει μια ροή επεξεργασίας όπου, όσο η συνθήκη A ειναι αληθής επαναλαμβάνονται οι εντολές του Block. Σε κάθε κύκλο επανάληψης, πρώτα βρίσκεται η τιμή της παράστασης A και άν αυτή ειναι $\neq 0$ τότε εκτελείται το Block εντολών. Ο μόνος τρόπος για να διαφύγει η ροή επεξεργασίας προς τον παρακάτω κώδικα, ειναι σε κάποιον απο τους κύκλους επανάληψης η συνθήκη A να πάρει την τιμή 0.

ΠΑΡΑΔΕΙΓΜΑΤΑ :

i) Όλες οι αποδόσεις τιμών του προηγούμενου παράδειγματος, μπορούν να γίνουν με το επαναληπτικό block που δίνεται στην αριστερή στήλη του παρακάτω πίνακα:

ΤΥΠΙΚΑ	ΠΙΟ ΣΥΝΤΟΜΑ	ΠΙΟ ΣΥΝΤΟΜΑ

{ int i=0;
while(i<100)
{ z[i]=x[i]+y[i];
i++; }
}
	{ int i=0;
	while(i<100)
	z[i]=x[i]+y[i], i++;
	}
		{ int i=0;
		while(i<100) z[i]=x[i]+y[i], i++; }

Οι υπόλοιπες στήλες δίνουν ισοδύναμες περιγραφές που προκύπτουν μέσω του απλοποιητικού κανόνα.
ΠΑΡΑΤΗΡΗΣΗ: Απαιτήθηκε μια μεταβλητή ακέραιου τύπου (η i) η οποία χρησιμοποιήθηκε για να σαρωθούν όλα τα στοιχεία της διανυσματικής μεταβλητής z. **Αρχικά έπρεπε να μηδενιστεί** και σε κάθε κύκλο επανάληψης **αρχικά να ελέγχεται άν έχει τιμή στο επιτρεπόμενο εύρος** και ύστερα απο την εκτέλεση των ουσιωδών εντολών **να αυξηθεί κατά ενα**. Αυτα τα 3 βήματα ειναι απαραίτητα σε κάθε επαναληπτική διαδικασία που εφαρμόζεται σε στοιχεία διανυσματικών μεταβλητών.

ii) Επαναληπτικά block χρησιμοποιούνται και για άλλους σκοπούς εκτός απο την απόδοση τιμών σε διανυσματικές μεταβλητές. Με το παρακάτω επαναληπτικό block προσδιορίζεται ο μικρότερος θετικός ρητός, της αριθμητικής του συστήματος στο οποίο εκτελείται αυτός ο κώδικας

```
{ double e=2, x;
  while( e/=2) x=e; }
```

Πράγματι σε κάθε κύκλο επανάληψης, αρχικά η τιμή της e υποδιπλασιάζεται και άν ειναι $\neq 0$ τότε μεταφέρεται στην μεταβλητή x. Με αριθμητική άπειρων ψηφίων αυτό θα σήμαινε επ'απειρο επανάληψη, όμως στην αριθμητική πεπερασμένων ψηφίων του H/Y, σε κάποια επανάληψη θα προκύψει ένας πολύ μικρός ρητός που το υποδιπλάσιό του θα στρογγυλευτεί στον αριθμό 0, οπότε και θα σταματήσει η εκτέλεση του επαναληπτικού Block. Σε αυτό το βήμα η μεταβλητή x θα έχει παραμείνει στην προηγούμενη τιμή της e, δηλαδή τον μικρότερο θετικό ρητό του συστήματος.

Το παραπάνω παράδειγμα i) καταδεικνύει τα 3 τυπικά βήματα ενός επαναληπτικού block για την σάρωση όλων των στοιχείων μιάς διανυσματικής μεταβλητής (αρχικοποίηση μιάς μεταβλητής θέσης, έλεγχο ορίων και τέλος ύστερα απο την εκτέλεση του Block των ουσιωδών εντολών, μεταβολή της μεταβλητής θέσης). Λόγω της ευρείας εφαρμογής αυτών των επαναλήψεων σε διανυσματικές μεταβλητές, έχει τυποποιηθεί και άλλου τύπου επαναληπτικό block. Μιά εντολή της μορφής:

for(A; B; Γ)
{
....
}
Block εντολών.

περιγράφει μια ροή επεξεργασίας όπου, αρχικά υπολογίζεται η παράσταση Α και έπειτα, όσο η συνθήκη Β ειναι αληθής επαναλαμβάνονται πρώτα οι εντολές του Block και μετά ο υπολογισμός της παράστασης Γ.

ΠΡΟΣΟΧΗ: Τα δύο ερωτηματικά ειναι μέρος του συντακτικού, απαιτούνται για να διαχωρίζονται οι τρείς παραστάσεις!

ΠΑΡΑΔΕΙΓΜΑΤΑ:

i') Το παράδειγμα i), υλοποιείται και με την παρακάτω εντολή

ΤΥΠΙΚΑ	ΠΙΟ ΣΥΝΤΟΜΑ

{ int i;
for(i=0; i<100; i++)
{ z[i]=x[i]+y[i];}
}
	{ int i;
	for(i=0; i<100; i++)
	z[i]=x[i]+y[i];
	}

ΣΗΜΕΙΩΣΗ: Το Block εντολών μιάς for εντολής, μπορεί να ενσωματωθεί στην παράσταση Γ αρκεί να διατηρηθεί η σωστή σειρά. Το παραπάνω for διατυπώνεται και στην ισοδύναμη μορφή

for(i=0; i<100; z[i]=x[i]+y[i], i++);

το ερωτηματικό της επαναλαμβανόμενης εντολής παρέμεινε έξω απο την παρένθεση, γιατί το συντακτικό της for απαιτεί ακριβώς δύο ερωτηματικά μέσα στην παρένθεση.

ii') Ανάλογα με τα διανύσματα ισχύουν για τους πίνακες, τα διανύσματα πινάκων, ... Για την σάρωση των στοιχείων πιό σύνθετων διανυσματικών μεταβλητών απαιτούνται περισσότερες μεταβλητές θέσης και αντίστοιχος αριθμός απο επαναλήψεις τύπου for.

Ο παρακάτω κώδικας τυπώνει τις τιμές των στοιχείων του πίνακα double mat[5][7] ανα γραμή και χωρισμένες με απόσταση.

ΤΥΠΙΚΑ	ΠΙΟ ΣΥΝΤΟΜΑ	ΠΙΟ ΣΥΝΤΟΜΑ

{ int i, j;
for(i=0; i<5; i++)
{ for(j=0; j<7; j++)
{ printf("%f ", mat[i][j]);}
printf("\n");
}
}
	{ int i, j;
	for(i=0; i<5; i++)
	{ for(j=0; j<7; j++)
	printf("%f ", mat[i][j]);
	printf("\n"); }
	}
		{ int i, j;
		for(i=0; i<5; printf("\n"), i++)
		for(j=0; j<7; j++)
		printf("%f ", mat[i][j]);
		}

ΣΗΜΕΙΩΣΗ: Ενσωματώνοντας την τελική εντολη μέσα στο δεύτερο for, προκύπτει η διατύπωση

for(i=0; i<5; printf("\n"), i++)
for(j=0; j<7; printf("%f ", mat[i][j]), j++);

Κλείνουμε την παράγραφο αναφέροντας μία παραλλαγή του επαναληπτικού while. Μια εντολή της μορφής:

do
{
....
}
Block εντολών.
while(A);

περιγράφει μια ροή επεξεργασίας όπου ξανα-επαναλαμβάνεται το Block εντολών άν η συνθήκη Α ειναι αληθής, δηλαδή παρόμοια με το επαναληπτικό while με την διαφορά ότι πρώτα εκτελείται το Block εντολών.

ΠΑΡΑΤΗΡΗΣΗ: **Τα άγκιστρα του Block εντολών δεν απλοποιούνται, ειναι μερος του συντακτικού της εντολής όπως και το ερωτηματικό στο τέλος αυτής.**

6) **ΔΙΑΧΕΙΡΙΣΗ ΔΙΕΥΘΥΝΣΕΩΝ ΜΝΗΜΗΣ (ΔΕΙΚΤΕΣ) ΚΑΙ ΚΑΤΑΣΚΕΥΗ ΝΕΩΝ ΤΥΠΩΝ**
Στο κεφάλαιο αυτό διδάχτηκαν οι παρακάτω έννοιες:

ΔΕΙΚΤΕΣ ΜΕΤΑΒΛΗΤΩΝ: Ορισμός, Αριθμητική διευθύνσεων, Σχέση δεικτών με τα διανύσματα μεταβλητών, Δυναμική παραχώρηση μνήμης, Δείκτες χαρακτήρων strings και τυποποιημένες συναρτήσεις διαχείρισης αυτών (strcmp, strlen, sprintf, atoi, atof).

ΔΕΙΚΤΕΣ ΣΥΝΑΡΤΗΣΕΩΝ: Ορισμός και χρήση της typedef, διανύσματα και πίνακες συναρτήσεων.

ΔΕΙΚΤΕΣ ΔΙΑΝΥΣΜΑΤΙΚΩΝ ΜΕΤΑΒΛΗΤΩΝ: Ορισμός, Σχέση τους με τους πίνακες μεταβλητών, Πίνακες μεταβλητού αριθμού στηλών ανα γραμμή.

ΔΙΑΣΥΝΔΕΣΗ ΜΕ ΤΟ ΛΕΙΤΟΥΡΓΙΚΟ ΣΥΣΤΗΜΑ: Γραμμή εντολών, Αρχεία συστήματος και τυποποιημένες συναρτήσεις διαχείρισης αυτών (fopen, fclose, fprintf, fscanf.

ΝΕΟΙ ΤΥΠΟΙ ΠΟΥ ΠΡΟΚΥΠΤΟΥΝ ΑΠΟ ΤΥΠΟΠΟΙΗΜΕΝΟΥΣ ΤΥΠΟΥΣ: Δομές (structures) και ενώσεις (unions) απλούστερων τύπων (ορισμός και χρήση της typedef, διαφορά μεταξύ των 2 κατηγοριών), Επιτρεπτές πράξεις, Μεταφορά τιμής μεταβλητών νέων τύπων απο/σε συνάρτηση.

Για περισσότερες πληροφορίες ο σπουδαστής πρέπει να ανατρέξει στις παρακάτω πηγές:

i. Βιβλιογραφία μαθήματος με ιδιαίτερη έμφαση στο βιβλίο "Η ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ C", Brian Kernighan, Κλειδάριθμος.
ii. Προσωπικές σημειώσεις απο τις διαλέξεις του μαθήματος.
iii. Τα εργαστήρια 3, 4, .., 8 (οι κώδικες αυτών υπάρχουν στην ιστοσελίδα του μαθήματος).
iv. Την άσκηση για την 1η Εργαστηριακή εξέταση (ιστοσελίδα του μαθήματος).
v. Τα θέματα της 2ης Προόδου (έχει επιστραφεί στους εξετασθέντες).

7) **ΤΥΠΟΠΟΙΗΜΕΝΑ ΣΧΗΜΑΤΑ ΑΠΟΘΗΚΕΥΣΗΣ ΚΑΙ ΔΙΑΧΕΙΡΙΣΗ ΑΥΤΩΝ**
Στο κεφάλαιο αυτό διδάχτηκαν οι παρακάτω έννοιες:

ΔΙΑΝΥΣΜΑΤΑ ΚΑΙ ΛΙΣΤΕΣ ΔΕΔΟΜΕΝΩΝ: Ορισμοι, επεξεργασία αυτών (Τακτοποίηση συνόλου δεδομένων με δοθείσα διάταξη, χρησιμοποιώντας τυποποιημένες συναρτήσεις Αναζήτησης (Σειριακή η Δυαδική για Διανύσματα, Σειριακή για Λίστες) και Ενημέρωσης . Για περισσότερες πληροφορίες ο σπουδαστής πρέπει να ανατρέξει στις παρακάτω πηγές:

i. Βιβλιογραφία μαθήματος με ιδιαίτερη έμφαση στο βιβλίο "Η ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ C", Brian Kernighan, Κλειδάριθμος.
ii. Προσωπικές σημειώσεις απο τις διαλέξεις του μαθήματος.
iii. Τα εργαστήρια 9, 10 (οι κώδικες αυτών υπάρχουν στην ιστοσελίδα του μαθήματος).
iv. Την άσκηση για την 2η Εργαστηριακή εξέταση (ιστοσελίδα του μαθήματος).