



Τεχνολογικό Εκπαιδευτικό Ίδρυμα Αθήνας
Τμήμα Μηχανικών Πληροφορικής Τ.Ε
Προγραμματισμός Υπολογιστών
Εργασία 02

Στέφανος Στεφάνου
Αριθμός Μητρώου : 161118
Τμήμα Α3(Τετάρτη 8:00-10:00)

Ερώτημα 01)

(1.1)Τι είναι σταθερά και τι είναι μεταβλητή; (1.2)Ποια τα βασικά χαρακτηριστικά μίας μεταβλητής;(1.3) Πώς σχετίζονται οι τύποι δεδομένων με τις σταθερές και τις μεταβλητές; (1.4)Δώστε παραδείγματα σταθερών τιμών διαφόρων τύπων δεδομένων.

Απάντηση 01)

(1.1)Με τον όρο Μεταβλητή , αναφερόμαστε σε μια συγκεκριμένη θέση μνήμης στην οποία μπορούμε να αποθηκεύσουμε δεδομένα συγκεκριμένης μορφής και στην οποία έχουμε δώσει ένα συμβολικό όνομα,για να μπορούμε να χρησιμοποιούμε τα δεδομένα της κατά την διάρκεια του προγραμματισμού ενός προγράμματος σε οποιαδήποτε γλώσσα προγραμματισμού .

Με τον όρο σταθερά , αναφερόμαστε σε οποιαδήποτε σταθερή τιμή συγκεκριμένου τύπου δεδομένων . Η οποία μπορεί να συμμετέχει σε παραστάσεις ή σε αριθμητικές πράξεις . Οι σταθερές απο μόνες τους δεν αποθηκεύονται στην μνήμη παρα μόνο προσωρινά . Για να αποθηκεύσουμε μια σταθερά στην μνήμη χρειαζόμαστε μια μεταβλητή. Στην γλώσσα προγραμματισμού C οι σταθεροί χαρακτήρες γράφονται με μονά εισαγωγικά ('a' , 'b') οι σταθερές συμβολοσειρές με διπλά ("foo" , "bar") καθώς οι σταθερές τιμές με αριθμούς (5,123,4.32)

(1.2)Τα χαρακτηριστικά μιας μεταβλητής είναι τα εξής

1. Το όνομα της μεταβλητής .
2. Ο τύπος δεδομένων της μεταβλητής
3. Το περιεχόμενο της μεταβλητής

(1.3)Οι τύποι δεδομένων είναι άρρηκτα συνδεδεμένοι με τις μεταβλητές . Όπως γνωρίζουμε , ένας τύπος δεδομένων , Είναι ουσιαστικά ο τρόπος που θα αποθηκεύουν τα δεδομένα στην μνήμη , αναλόγα με την φύση τους , καθώς και ο τρόπος για να γίνονται εργασίες πάνω σε αυτά.Οι μεταβλητές όπως αναφέραμε , αποθηκεύουν δεδομένα συγκεκριμένου τύπου . Έτσι ένας τύπος δεδομένων "δειχνει" τον τρόπο αποθήκευσης των δεδομένων μιας μεταβλητής στην μνήμη.Όπως προείπαμε . Ο τύπος δεδομένων προδιαγράφει επίσης τον τρόπο που θα γίνονται εργασίες(πράξεις) πάνω σε δεδομένα συγκεκριμένου τύπου . Έτσι όταν οι σταθερές/μεταβλητές , συμμετέχουν σε πράξεις ή αριθμητικές/ λογικές παραστάσεις, ο υπολογιστής γνωρίζοντας τον τύπο δεδομένων της

εκάστοτε σταθεράς/μεταβλητής , είναι σε θέση να γνωρίζει πως να χειριστεί τα δεδομένα αυτά . Για αυτόν τον λόγο οι τύποι δεδομένων είναι άρρηκτα συνδεδεμένοι με τις μεταβλητές και τις σταθερές , διότι προδιαγράφουν τον τρόπο επεξεργασίας και αποθήκευσης των δεδομένων μιας μεταβλητής ή σταθεράς ανάλογα με την φύση των δεδομένων που περιέχουν

(1.4) Παραδείγματα σταθερών

1. 273 -> Ακέραια σταθερά(int)
2. '2' -> Αλφαριθμητική σταθερά(char)
3. 'a' -> Αλφαριθμητική σταθερά (char)
4. "the foo bar" -> Σταθερά συμβολοσειράς (η συμβολοσειρα περικλείεται αυστηρά σε διπλά εισαγωγικά)(array of char's)
5. 5.43 -> Δεκαδική σταθερά(float)

Ερώτηση 02) (2.1)Τι είναι η standard είσοδος και τι η standard έξοδος ενός προγράμματος;
(2.2) Τι γνωρίζετε για τις συναρτήσεις με τις οποίες χειριζόμαστε τη standard είσοδο και την standard έξοδο στον C προγραμματισμό;

Καθώς το πρόγραμμα "τρέχει". Είναι πιθανόν να ζητήσει κάποια δεδομένα , ή να χρειαστεί να εξάγει προς τα κάπου αποτελέσματα . Το πρόγραμμα το ίδιο,δεν συνδέεται απευθείας με τις συσκευές εισόδου εξόδου , αλλά παραδίδει τα δεδομένα προς εκτύπωση ή αναμένει τα δεδομένα προς εισαγωγή , απο τα γνωστά ρεύματα εισόδου και εξόδου (stdin,stdout και stderr).

Πώς λειτουργεί...

Στα περισσότερα λειτουργικά συστήματα , πριν το Unix , κάθε πρόγραμμα έπρεπε να συνδεθεί απευθείας με την συσκευή που ήθελε , για να ανταλλάξει δεδομένα. Το Unix έφερε τα ρεύματα εισόδου εξόδου(data streams) για πρώτη φορά , με την τεχνική των "Αφηρημένων Συσκευών"(abstract devices).Σύμφωνα με την τεχνική αυτή , οι συσκευές (Συμπεριλαμβανομένου του πληκτρολογίου και της οθόνης)απεικονίζονται σαν ειδικά αρχεία στον σκληρό δίσκο , τα οποία κάθε πρόγραμμα απλά διαβάζει και εγγράφει . Όλη η διαδικασία μεταφοράς των δεδομένων στην εκάστοτε συσκευή παραμένει θέμα καθαρά του λειτουργικού συστήματος και οι προγραμματιστές δεν μπλέκονται καθόλου με αυτό το κομμάτι . Τα προγράμματα τα οποία θέλουν να εξάγουν αποτελέσματα ,απλά εγγράφουν στο αρχείο της εκάστοτε συσκευής (απο οθόνη ως σειριακή θύρα) και το λειτουργικό(συγκεκριμένα οι drivers της κάθε συσκευής) αναλαμβάνει την υπόλοιπη δουλειά.

Χωρίς να κάνουμε απολύτως τίποτα . Το πρόγραμμα μας "συνδέεται" με τα *προεπιλεγμένα* ρεύματα εισόδου εξόδου (stdin,stdout) με την οθόνη και το πληκρολόγιο . Και όταν εμείς θελήσουμε να εξάγουμε αποτελέσματα ή να εκτυπώσουμε κάτι στην οθόνη . Απο

προεπιλογή τα δεδομένα θα πάνε σε αυτές τις συσκευές. Τα ειδικά αρχεία συσκευών βρίσκονται στον φάκελο /dev/ στα συστήματα Unix και Linux .

(2.2) Οι συναρτήσεις που έχουμε μάθει να χειριζόμαστε τα ρεύματα εισόδου εξόδου , είναι η scanf() για χειρισμό του stdin(προεπιλεγμένο ρεύμα εισόδου) και printf() για χειρισμό του stdout(Προεπιλεγμένο ρεύμα εξόδου) .

Ακολουθεί η σύνταξη των προαναφερόμενων συναρτήσεων

Η συνάρτηση printf()..

printf("string format",param_1,param_2,.....,param_n)

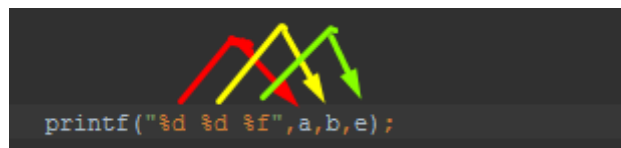
Όπου το string format είναι η Συμβολοσειρά(FormatString)που παρέχει οδηγίες εκτύπωσης του επιθυμητού αποτελέσματος στην stdout . Το FormatString περιέχει οδηγίες για την εκτύπωση μέσω ειδικών χαρακτήρων διαφυγής (Escape Characters) και ειδικών Flag που θα αναφερθούν παρακάτω...

Ας δούμε ένα παράδειγμα

```
#include <stdio.h>
int main() {
    int a=20,b=30;
    float e=2.71;
    printf(" %d %d %f \n",a,b,e);
}
```

Παρατηρούμε εδώ , η printf() χρησιμοποιεί 3 string modifiers (τον "%d"(για ακέραιους αριθμούς) 2 φορές και τον "%f"(Για αριθμούς κινητής υποδιαστολής) μία φορά) καθώς επίσης και τον ειδικό escape character '\n' ο οποίος αναγκάζει τον κέρσορα της οθόνης να γεμίσει την υπόλοιπη οθόνη (line feed) και να πάει στην απο κάτω γραμμή (carriage return).

Στην θέση του πρώτου modifier θα μπει το περιεχόμενο της μεταβλητής a (20) και την θέση του δεύτερου το περιεχόμενο της μεταβλητής (b). Τέλος στον τελευταίο modifier θα μπει η τιμή της μεταβλητής e (2.71) και ο κέρσορας θα προβεί στην επόμενη γραμμή .



```
printf("%d %d %f",a,b,e);
```

Έτσι το αποτέλεσμα μας θα είναι...

20 30 2.710000

Η printf() μπορεί να δειχθεί τους κάτωθι modifiers ανάλογα με το είδος των δεδομένων που πρόκειται να εξάγει, όπως..

modifier	Ιδιότητα Modifier	Παράδειγμα
d or i	Προσημασμένος ακεραίος Αριθμός	392

u	Μη-προσημασμένος ακέραιος Αριθμός	7235
o	Μη προσημασμένος αριθμός με βάση το 8 (οχταδικός)	610
x	Μη προσημασμένος αριθμός με βάση το 16 (Δεκαεξαδικός)	7fa
X	Μη προσημασμένος αριθμός με βάση το 16(Δεκαεξαδικός)(κεφαλαία)	7FA
f	Προσημασμένος Αριθμός κινητής υποδιαστολής	392 . 65
F	Προσημασμένος Αριθμός κινητής υποδιαστολή	392 . 65
e	Αριθμός γραμμένος με επιστημονική σημειογραφία(Maddissa)	3 . 9265e+2
E	Αριθμός γραμμένος με επιστημονική σημειογραφία(Maddissa)(Κεφ)	3 . 9265E+2
g	Εκτύπωση με την μικρότερη έκφραση(%f ή &e)	392 . 65
G	Εκτύπωση με την μικρότερη έκφραση(%F ή %E)	392 . 65
a	Προσημασμένος Δεκαδικός αριθμός με βάση το 16(Δεκαεξαδικό)	-0xc . 90fep -2
A	Προσημασμένος Δεκαδικός αριθμός με βάση το 16(Δεκαεξαδικό)(Κεφ)	-0XC . 90FEP -2
c	Χαρακτήρας	a
s	Αλφαριθμητική Συμβολοσειρά	sample
p	Διεύθυνση μνήμης RAM (χρησιμοποιείται σε συνδιασμό με τους δείκτες)	b8000000
n	Καμία εκτύπωση	
%	Ο χαρακτήρας % ακολουθούμενος απο εναν & εξάγει τον χαρακτήρα % στο προεπιλεγμένο ρεύμα εισόδου	%

Επίσης , μπορούν να χρησιμοποιηθούν και παρακάτω Escape Characters

Escape sequence	Character represented
\a	Ήχος βομβητή
\b	Backspace
\f	Form Feed(Γέμισμα γραμμής)
\n	Νέα γραμμή (Συνδιασμός \f και \r)
\r	Επιστροφή φορέα(Carriage Return)
\t	Οριζόντια επομενη στήλη (Horizontal Tab)
\v	Κάθετη επόμενη στήλη(Vertical Tab)
\\	Χαρακτήρας Πισωκάθετος(BackSlash)
\'	Απλό εισαγωγικό
\"	Διπλό εισαγωγικό
\?	Ερωτηματικό
\nnn	Το byte που η τιμή του δίνεται ως nnn και εκφράζεται ως αριθμός με βάση το οχταδικό Το byte που η τιμή του δίνεται ως nnn και εκφράζεται ως αριθμός με βάση το οχταδικό
\xhh...	Το byte που η τιμή του δίνεται ως nnn και εκφράζεται ως αριθμός με βάση το Δεκαεξαδικό

Επιπροσθέτως υπάρχουν και οι εξής άλλες συναρτήσεις για χειρισμό ρευμάτων .

1. Fscanf() (Εισαγωγή δεδομένων σε υπάρχων ανοιχτό ρεύμα)
2. fprintf() (Εξαγωγή δεδομένων σε υπάρχων ανοιχτό ρεύμα)
3. getchar() (Εισαγωγή χαρακτήρα απο προεπιλεγμένο ρεύμα)(**)
4. getch() (Εισαγωγή χαρακτήρα απο προεπιλεγμένο ρεύμα)(*)
5. putchar() (Εξαγωγή χαρακτήρα απο προεπιλεγμένο ρεύμα)(**)
6. putch() (Εξαγωγή χαρακτήρα απο προεπιλεγμένο ρεύμα)(*)
7. fopen() (Εγκαθίδρυση ρεύματος με άνοιγμα αρχείου)
8. fclose() (Κλείσιμο ρεύματος)
9. perror() (Εξαγωγή στο προεπιλεγμένο ρεύμα λαθών (stderr))

--> (*) και (**) έχουν μερικές διαφορές ως προς την λειτουργία<--

Η συνάρτηση scanf()...

```
scanf(char *formatString,*Address01,*Address02 ,.....,*AddressXX);
```

Η συνάρτηση scanf() διαβάζει μορφοποιημένα δεδομένα απο το προεπιλεγμένο ρεύμα εισόδου εξόδου . Και τα εναποθέτει στις διευθύνσεις μνήμης που δίνονται μετέπειτα απο το FormatString . Η scanf() είναι μια συνάρτηση η οποία αποτελεί πολυ ισχυρό εργαλείο στα χέρια ενός έμπειρου προγραμματιστή .

Ας δούμε ένα παράδειγμα

```
#include <stdio.h>
int main() {
    int a,b,c;
    float d;
    scanf("%d%d%d%f",a,b,c,d);
    printf("|%d|%d|%d|%d|%f",a,b,c,d);
}
```

Όπως παρατηρούμε εδώ , αναλύοντας το FormatString. Η scanf() περιμένει να δεχθεί τα εξής δεδομένα

1. 3 Αριθμούς Τύπου ακεραίου (int)
2. 1 Αριθμό κινητής υποδιαστολής Float(float)

Αξίζει να σημειώσουμε οτι η scanf() θεωρεί τους χαρακτήρες λευκού διαστήματος (Whitespace characters) ως διαχωριστικό μεταξύ των δεδομένων .έτσι διαβάζει έναν αριθμό και τον καταχωρεί στην αντιστοιχη θέση μνήμης , μόνο όταν ανιχνεύσει χαρακτήρα κενού,επόμενης γραμμής ή επόμενης στηλοθέτησης (\t ή \n ή “ “)

Οι StringModifiers και οι EscapeCharacters που η scanf μπορεί να δεχθεί είναι οι ίδιοι με αυτούς της printf() . Μόνο που η scanf() έχει και κάποιες επιπλέον λειτουργίες , που ξεφεύγουν απο το εύρος της εργασίας και του ερωτήματος αυτού .

Ερώτημα 03)(1.1)Τι είναι αριθμητικοί,(1.2) τι είναι σχεσιακοί και τι είναι(1.3) λογικοί τελεστές;(1.4) Αναφέρετε τη λειτουργία των βασικότερων τελεστών και από τις τρεις κατηγορίες. (1.5)Εξηγήστε την διαφορά του τελεστή προαύξησης από τον τελεστή μετααύξησης.

Οι τελεστές γενικά , επιτελούν εργασίες (Πράξεις) Πάνω σε δεδομένα συγκεκριμένου τύπου .Οι τελεστές γενικά διακρίνονται σε 3 κατηγορίες , ανάλογα με το είδος επεξεργασίας που επιτελούν.Έτσι έχουμε

1. Τους Αριθμητικούς τελεστές
2. Τους σχεσιακούς τελεστές
3. Λογικούς τελεστές

Αριθμητικοί τελεστές

Οι αριθμητικοί τελεστές , επιτελούν πάνω σε αριθμητικά (και όχι μόνο *)δεδομένα αριθμητικές πράξεις.Αριθμητικοί τελεστές μπορούν να συνηπάρχουν μέσα σε μια σύνθετη παράσταση σε συνδιασμό με μεταβλητές και σταθερές.Σε αυτήν την περίπτωση υπάρχει συγκεκριμένη προτεραιότητα πράξεων που αναφέρεται παρακάτω.Στην C οι αριθμητικοί τελεστές είναι οι ακόλουθοι.

Όνομα Τελεστή		Σύνταξη
Τελεστής εκχώρησης		<code>a = b</code>
Τελεστής πρόσθεσης		<code>a + b</code>
Τελεστής Αφαίρεσης		<code>a - b</code>
Τελεστής θετικού προσήμου		<code>+a</code>
Τελεστής αρνητικού προσήμου		<code>-a</code>
Τελεστής Πολλαπλασιασμού		<code>a * b</code>
Τελεστής Διαίρεσης		<code>a / b</code>
Υπόλοιπο Ακαίρεας διαίρεσης		<code>a % b</code>
Τελεστής Αύξησης	Με πρόθεμα	<code>++a</code>
	Με Επίθεμα	<code>a++</code>
Τελεστής Μείωσης	Με πρόθεμα	<code>--a</code>
	Με Επίθεμα	<code>a--</code>

Προτεραιότητα Αριθμητικών τελεστών....

Οι αριθμητικοί τελεστές όταν υπάρχουν σε συνδιασμό σε μια αριθμητική παράσταση τότε για την σωστή διενέργηση των πράξεων και την διεξαγωγή σωστού αποτελέσματος , ακολουθείται μια προτεραιότητα η οποία είναι ως εξής .

Ξεκινώντας απο τα αριστερά προς τα δεξιά

1. Πράξεις στις παρενθέσεις
2. Τελεστές Αύξησης και μείωσης με επιθεμα (a++)
3. Τελεστές Αύξησης και μείωσης με πρόθεμα (++a)
4. Πολλαπλασιασμοί , διαίρεσεις και υπόλοιπα διαίρεσης (* , / , %)
5. Προσθέσεις και αφαιρέσεις (+ , - , -- , ++)
6. Εκχωρήσεις (=)

Σχεσιακοί τελεστές ...

Οι σχεσιακοί τελεστές επιτελούν πάνω σε δεδομένα συγκρίσεις . Και επιστρέφουν την ακέραια τιμή 1 σε περίπτωση που η σύγκριση είναι αληθής , και 0 σε περίπτωση που η σύγκριση δεν είναι αληθής. Όταν παραπάνω απο ένας σχεσιακοί τελεστές συνηπάρχουν σε μία παράσταση . Τότε κρατείται προτεραιότητα , για την διεξαγωγή σωστών αποτελεσμάτων .

Σε αυτό το σημείο θα ήταν σωστό να τονίσουμε , ότι η C θεωρεί αληθή , οποιαδήποτε παράσταση η οποία είναι διάφορη του μηδενός , έτσι η παράσταση 5+10 θεωρείται αληθής .

Ας δούμε τους βασικούς σχεσιακούς τελεστές

Όνομα Τελεστή	Σύνταξη
Τελεστής ίσου	a == b
Τελεστής διάφορο	a != b a not_eq b
Μεγαλύτερο απο	a > b
Μικρότερο απο	a < b
Μεγαλύτερο ή ίσο	a >= b
Μικρότερο ή ίσο	a <= b

Προτεραιότητα Σχεσιακών τελεστών.

Όπως προείπαμε, όταν έχουμε παραπάνω απο έναν σχεσιακό τελεστή σε μια παράσταση , κρατείται αυστηρά προτεραιότητα τελεστών , όπως ορίζεται πιο κάτω .

1. Τελεστές < , <= , > , >=
2. Τελεστές == , !=

Λογικοί Τελεστές ...

Οι λογικοί τελεστές , εφαρμόζουν πράξεις της άλγευρας boole σε δεδομένα . Όπως όλοι οι τελεστές ,κρατούν συγκεκριμένη προτεραιότητα σε περίπτωση παράστασης .

Όνομα Τελεστή	Σύνταξη
Τελεστής Άρνησης(NOT)	<ul style="list-style-type: none">• <code>!a</code>◦ <code>not a</code>
Λογικό ΚΑΙ(AND)	<ul style="list-style-type: none">• <code>a && b</code>◦ <code>a and b</code>
Λογικό Η(OR)	<code>a b</code> <code>a or b</code>

Ο λογικός τελεστής ΚΑΙ (AND)

Ο Λογικός τελεστής ΚΑΙ , επιστρέφει αληθές αποτέλεσμα , μόνο όταν και οι δύο παραστάσεις ή λογικές εκφράσεις δεξιά και αριστερά του , είναι αληθείς (στην πραγματικότητα διάφορες του μηδενος)

Ας δούμε ένα παράδειγμα

```
#include <stdio.h>
int main() {
    int expression = 5>4 and 2 - !2 - !0;
    printf("%d",expression);
}
```

Ας αναλύσουμε την *expression*

```
int expression = 5>4 and 2 - !2 - !0;
```

Γνωρίζοντας την συνολική προτεραιότητα των πράξεων η οποία αναφέρεται αμέσως πιο κάτω ,εκτελώντας διαδοχικά έχουμε...

*Σημείωση , ο τελεστής NOT (!) θα αναλυθεί πιο κάτω.

1. `int expression = 5>4 and 2 - 0 - !0;`
2. `int expression = 5>4 and 2 - 0 - 1;`
3. `int expression = 5>4 and 1;`
4. `int expression = 1 and 1; //και οι δύο τιμες είναι != 0 !!`
5. `int expression = 1;`

Όπως παρατηρήσαμε στο βήμα 4 , Και οι δύο τιμες δεξιά και αριστερά του τελεστή `and` είναι αληθείς , έτσι το αποτέλεσμα είναι 1 (Δηλαδή αληθές)

Ο πίνακας αληθείας του λογικού τελεστή ΚΑΙ είναι ο εξής

Τιμές x	Τιμές y	Αποτέλεσμα x AND y
0	0	0
0	1	0
1	0	0
1	1	1

Λογικός Τελεστής OR

Ο λογικός τελεστής OR επιστρέφει τιμή αληθείας , Στην περίπτωση που *τουλάχιστον* μία απο τις τιμές ή τις παραστάσεις που έχει δεξιά ή αριστερά του είναι διάφορες του μηδέν(Αληθής)

Ας δούμε ένα παράδειγμα

```
#include <stdio.h>
int main() {
    int expression = 1<2 or 'A'!=65;
    printf("%d",expression);
}
```

Ας αναλύσουμε λίγο την expression,

```
int expression = 1<2 or 'A'!=65;
```

Γνωρίζοντας την προτεραιότητα των πράξεων , Απο δεξιά προς τα αριστερά , εκτελώντας διαδοχικά έχουμε

1. `int expression = 1 or 'A'!=65;`
2. `int expression = 1 or 0;`
3. `int expression = 1;`

Έτσι , ο πίνακας αληθείας του τελεστή OR ορίζεται ως εξής

Τιμές x	Τιμές y	Αποτέλεσμα x OR y
0	0	0
0	1	1
1	0	1
1	1	1

Ο λογικός τελεστής NOT

Ο λογικός τελεστής NOT , επιστρέφει αντεστραμμένη την λογική τιμή στα δεξιά του . Σε περίπτωση που δεξιά του υπάρχει τιμή 0 , επιστρέφει 1 , σε οποιαδήποτε άλλη περίπτωση επιστρέφει 0 .

Ο λογικός τελεστής NOT έχει τον εξής πίνακα τιμών

Τιμές x	Αποτέλεσμα NOT x
0	1
1	0

Προτεραιότητα λογικών τελεστών

Όπως προείπαμε , οι λογικοί τελεστές κρατούν και αυτοί μια προτεραιότητα όσο αναφορά τις παραστάσεις με παραπάνω απο εναν τελεστές . Η προτεραιότητα ορίζεται ως εξής

Ξεκινώντας απο τα αριστερα στα δεξιά

1. Τελεστής \rightarrow NOT
2. Τελεστής \rightarrow AND
3. Τελεστής \rightarrow H

Συνολική Προτεραιότητα Τελεστών

Στην γλώσσα προγραμματισμού C , είναι εφικτό να υπάρχουν και παραστάσεις με παραπάνω απο ένα είδος τελεστή . Έτσι ξεκινώντας απο αριστερά στα δεξιά, η συνολική προτεραιότητα των πράξεων ορίζεται στον παρακάτω πίνακα

1. Τελεστές μεταάυξησης(++ και αντιστοίχως --), παρενθέσεις
2. Τελεστές προάυξησης(++ και αντιστοίχως --) , τελεστής θετικού και αρνητικού προσήμου (+ και -) , Λογικό OXI , Bitwise OXI .
3. Πολλαπλασιασμος (*), Διαίρεση (/), Υπόλοιπο διαίρεσης
4. Πρόσθεση (+) , Αφαίρεση (-)
5. Μεγαλύτερο (>) , Μικρότερο (<) , Μεγαλύτερο και ίσον (>=), Μικρότερο και ίσον(<=)
6. Σχεσιακή Ισοδυναμία (==), Σχεσιακό Διάφορο (!=)
7. Bitwise AND(&) , Bitwise OR(|)
8. Λογικό AND(&&)

9. Λογικό OR (||)

10. Τελεστής Ανάθεσεως (=)

Τελεστής προαύξησης με μεταάυξησης

Οι τελεστές προαύξησης και μεταάυξησης , παρόλο που είναι αρκετά χρήσιμοι , μπορούν να γίνουν η αιτία αρκετών bugs , αν δεν ξεκαθαριστεί η λειτουργία τους .

Τελεστής προαύξησης (++ και αντιστοίχως --)

Ο τελεστής προαύξησης , μπαίνει πριν την μεταβλητή που θέλουμε να προσαυξήσουμε , και αφού αυξήσει την τιμή της , επιστρέφει ως αποτέλεσμα παράστασης την αυξημένη τιμή

Ας δούμε ένα παράδειγμα

```
#include <stdio.h>
int main() {
    int b=10;
    printf("%d", ++b);
}
```

Ο τελεστής ++ αφού αύξησε κατά 1 το περιεχόμενο της μεταβλητης b , επέστρεψε την αυξημένη τιμή στην printf(). Έτσι το αποτέλεσμα μας είναι 11.

Τελεστής μεταάυξησης (++ και αντιστοίχως --)

Ο τελεστής μεταάυξησης , μπαίνει μετά απο την μεταβλητη ,Και αφού επιστρέφει την τρέχουσα τιμή , αυξάνει το περιεχόμενο της κατά 1.

Ας δούμε ένα παράδειγμα

```
#include <stdio.h>
int main() {
    int b=10;
    printf("%d\n", b++);
    printf("%d\n", b);
}
```

Εδώ , ο τελεστής Μεταάυξησης αφού επιστρέψει την προηγούμενη τιμή ως τιμή της παράστασης , αυξάνει το περιεχόμενο της μεταβλητής b κατά 1!

Η Βασική διαφορά λοιπόν

Μεταξύ του τελεστή προαύξησης και μεταάυξησης είναι το γεγονός ότι ο ένας πρώτα αυξάνει και μετά επιστρέφει , ενώ ο τελεστής μεταάυξησης επιστρέφει την προηγούμενη τιμή , και μετέπειτα αυξάνει την τιμή κατά 1!

Ερώτηση 04) Να γραφεί πρόγραμμα το οποίο θα επιβεβαιώνει προγραμματιστικά τον αριθμό των bytes μνήμης που καταλαμβάνουν οι μεταβλητές από κάθε ένα από τους τέσσερις βασικούς τύπους δεδομένων οι οποίοι αναφέρονται στη θεωρία.

```
#include <stdio.h>
int main() {
    printf("Short Int -> %d\n", sizeof(short int));
    printf("Unsigned Short Int-> %d\n", sizeof(unsigned short int));
}
```

```

printf("Unsigned Int->%d\n",sizeof(unsigned int));
printf("Int ->%d\n",sizeof(int));
printf("Long Int->%d\n",sizeof(long int));
printf("Unsigned Long Int->%d\n",sizeof(unsigned long int));
printf("Unsigned Char-> %d\n",sizeof(unsigned char));
printf("Float-> %d\n",sizeof(float));
printf("Double-> %d\n",sizeof(double));
printf("Long Double->%d\n",sizeof(long double));
}

```

Ερώτηση 5) Να γραφεί πρόγραμμα το οποίο θα διαβάσει από την standard είσοδο δύο ακέραιους αριθμούς και θα υπολογίζει:

- Το άθροισμα, τη διαφορά, το γινόμενο, το πηλίκο και το υπόλοιπο της διαίρεσης των δύο αριθμών.
- Το πραγματικό πηλίκο των δύο αριθμών.
- Το τετράγωνο του πρώτου αριθμού και την τετραγωνική ρίζα του δεύτερου.

Το πρόγραμμα θα τυπώνει στην standard έξοδο όλα τα παραπάνω αποτελέσματα όσο πιο... όμορφα γίνεται. Δεν χρειάζεται να λάβετε υπόψη την πιθανότητα διαίρεσης με το 0.

Κώδικας άσκησης 5)

```

#include <stdio.h>
#include <math.h>
int main() {
    int a,b;
    printf("Insert two values! ==> ");
    scanf("%d%d",&a,&b);
    printf("\n-----*\n");
    printf("Add->%d\r\n",a+b);
    printf("Rev->%d\n",a-b);
    printf("Mul->%d\n",a*b);
    printf("Div->%d\n",a/b);
    printf("Rem Div->%d\n",a%b);
    printf("Real Div->%4.2f\n", (float(a)/b));
    printf("Sqr->%d\n",a*a);
    printf("Sqrt->%f\n",sqrt(b));
    printf("-----*");
    scanf("%d",&a);
}

```

Ερώτηση 6) Να γραφεί πρόγραμμα το οποίο θα διαβάσει το μήκος, σε μέτρα, της ακμής ενός κύβου και να υπολογίζει το εμβαδόν και τον όγκο του κύβου. Να υπολογίζει, επίσης, το εμβαδόν και τον όγκο σφαίρας με ακτίνα ίδιου μήκους με την ακμή του προηγούμενου κύβου.

Κώδικας Άσκησης 6)

```

#include <stdio.h>
#include <math.h>
#define pi 3.14159
//Forward Declarations
void PrintSurfaceOfCube(double Meters);
void PrintVolumeOfCube(double Meters);
void PrintSurfaceOfSphere(double Meters);
void PrintVolumeOfSphere(double Meters);

```

```

int main() {
    double Meters;
    scanf("%lf", &Meters);
    PrintSurfaceofCube(Meters);
    PrintVolumeOfSphere(Meters);
    printf("*-----*\n");
    PrintSurfaceofSphere(Meters);
    PrintVolumeOfSphere(Meters);
}

void PrintSurfaceofCube(double Meters) {
    printf("Surface Cube-> %4.2lf \n", 6*pow(Meters, 2));
}

void PrintVolumeOfCube(double Meters) {
    printf("Volume Cube %4.2lf \n", pow(Meters, 3));
}

void PrintSurfaceofSphere(double Meters) {
    printf("Surface Sphere-> %4.2lf \n", (4*pi*pow(Meters, 2)));
}

void PrintVolumeOfSphere(double Meters) {
    printf("Volume Sphere->%4.2lf \n", (4./3)*pi*pow(Meters, 3));
}

```

Ερώτημα 07) Ποια η λειτουργία των τελεστών “|” και “&” και ποια η διαφορά τους από τους “||” και “&&” αντίστοιχα; Δώστε παραδείγματα.....

Οι Bitwise Τελεστές....

Οι τελεστές “|” και “&” είναι δυαδικοί τελεστές, που εφαρμόζουν την λογική πράξη AND ανάμεσα στους αριθμούς που περιέχουν δεξιά και αριστερά τους. Ονομάζονται και Bitwise Τελεστές, καθώς εφαρμόζουν πράξεις στην εσωτερική απεικόνιση των αριθμών αυτών. Δηλαδή σε επίπεδο bit

Η διαφορά με τους λογικούς τελεστές AND(&&) και OR (||) είναι το γεγονός ότι οι λογικοί τελεστές, επιστρέφουν τιμή 1 ή 0, ανάλογα με το αποτέλεσμα της λογικής πράξης που κάνουν. Ενώ οι bitwise τελεστές, κάνουν την πράξη, bit προς bit στην εσωτερική απεικόνιση των αριθμών δεξιά και αριστερά τους.

Ας δούμε αναλυτικά ένα παράδειγμα.

```

#include <stdio.h>
int main() {
    printf("%d", 201&&9);
}

```

Εδώ, είχαμε 2 αριθμητικές τιμές, διάφορες του μηδενός, έτσι το αποτέλεσμα μας θα είναι 1

****->** Ας θυμηθούμε ότι οποιαδήποτε παράσταση διάφορη του μηδενός αντιπροσωπεύει την τιμή αληθείας (True).

Ας δούμε τώρα ένα παράδειγμα με χρήση των Bitwise τελεστών.

```

#include <stdio.h>
int main() {
    printf("%d", 201&9);
}

```

Εδώ παρατηρούμε ότι το εξαγόμενο αποτέλεσμα μας είναι 9! ας δούμε το γιατί...

Όπως προείπαμε , οι Bitwise Τελεστές κάνουν πράξεις στο εσωτερικό των bit. Έτσι η πράξη AND μεταξύ των αριθμών 201 και 9 εκτελείται ως εξής

Εσωτερική απεικόνιση των αριθμών .									
0	1	1	0	0	1	0	0	1	201
0	0	0	0	0	1	0	0	1	9
0	0	0	0	0	1	0	0	1	AND

Έτσι , η πράξη μας δίνει τον αριθμο 000001001 που μεταφράζεται στον δεκαδικό αριθμό 9 .

Ερώτημα 08)Προσπαθήσετε να... μαντέψετε την τιμή της ακέραιας μεταβλητής x μετά την εκτέλεση του ακόλουθου snippet. Εκτελέστε τον κώδικα και ελέγξτε την... μαντεψιά σας. Εξηγήστε το αποτέλεσμα.

```
x = 5;
```

```
x = x ++;
```

Απάντηση8)Ο τελεστής μεταύξησης θα αυξήσει την τιμή της μεταβλητής x και θα επιστρέψει ως αποτέλεσμα την προηγούμενη τιμή του x , πριν την αύξηση.Έτσι , η αύξηση θα συμβεί στιγμιαία στην τιμή του x , καθώς ως αποτέλεσμα της παράστασης αυτής, θα είναι η προηγούμενη τιμή του x. πριν την αύξηση . Έτσι στο x θα εκχωρηθεί η τιμή 5 αντί της τιμής 6

Ερώτημα 09)Ποια συντακτικά προβλήματα βλέπετε στην ακόλουθη αριθμητική παράσταση;

$$x = 3 * 'A' + 2 * 'D' / (2 + 4 > 5);$$

Συμφωνεί ο compiler μαζί σας;

Απάντηση9) Στην παραπάνω πρόταση , παρόλο που πολλαπλασιάζουμε χαρακτήρες , και διαιρούμε με αποτελέσματα συγκρίσεων(Σχεσιακών τελεστών),Παραδόξως , δεν υπάρχει κανένα απολύτως συντακτικό λάθος . Αυτό οφείλεται στην ελευθερία που μας δίνει η C σε σχέση με άλλες γλώσσες προγραμματισμού .

Γνωρίζοντας την προτεραιότητα των τελεστών , απο αριστερα στα δεξιά , εκτελώντας διαδοχικά , έχουμε...

*Το 'A' και το 'D' μετατράπηκαν στους αντίστοιχους αριθμούς τους στο ASCII. Καθώς ήδη γνωρίζουμε απο το μάθημα 1 ότι οι χαρακτήρες δεν είναι τίποτε άλλο απο αριθμοί .

1. $x = 3 * 65 + 2 * 68 / (2 + 4 > 5);$

2. $x = 3 * 65 + 2 * 68 / (6 > 5);$

3. $x = 3 * 65 + 2 * 68 / 1;$

4. $x = 195 + 136;$

5. $x = 331$

