Θεματική ενότητα 3:
Τελεστές – εκφράσεις

Τελεστές (operators) – Εκφράσεις (expressions)

- **Σύμβολα ή λέξεις που αναπαριστούν συγκεκριμένες διεργασίες, οι οποίες εκτελούνται πάνω σε ένα ή περισσότερα δεδομένα.**
- **Τα δεδομένα καλούνται τελεστέοι (operands) και μπορούν να είναι μεταβλητές, σταθερές ή ακόμη και κλήσεις συναρτήσεων.**
- **Οι τελεστές χρησιμοποιούνται για το σχηματισμό εκφράσεων. Μία έκφραση, εν γένει, αποτελείται από έναν ή περισσότερους τελεστέους και προαιρετικά από έναν ή περισσότερους τελεστές.**
- **Κάθε έκφραση έχει μία τιμή, η οποία υπολογίζεται με ορισμένους κανόνες.**

Σύμβολα συνηθισμένων τελεστών

Δυαδικός τελεστής	Μαθηματικό σύμβολο	C	Pascal
μικρότερο	<	<	<
μικρότερο ή ίσο	≤	<=	<=
ίσο	=	==	=
διάφορο	≠	!=	<>
μεγαλύτερο	>	>	>
μεγαλύτερο ή ίσο	≥	>=	>=
πρόσθεση	+	+	+
αφαίρεση	-	-	-
πολλαπλασιασμός	*	*	*
διαίρεση πραγματικών	/	/	/
διαίρεση ακεραίων	div	/	div
υπόλοιπο διαίρ. ακερ.	Mod	%	mod

Συμβολισμοί στο σχηματισμό εκφράσεων

Ένας δυαδικός (binary) τελεστής μπορεί να τοποθετηθεί:

- **Μεταξύ των δεδομένων στα οποία ενεργεί, όπως στην έκφραση x+y, οπότε έχουμε τη σημειογραφία *ένθεσης* ή ένθετου τελεστή (infix notation).**

- **Πριν από τους τελεστέους, όπως στην έκφραση +x y, οπότε έχουμε τη σημειογραφία *πρόθεσης* ή *προπορευόμενου τελεστή* (prefix notation).**

- **Μετά από τους τελεστέους, όπως στην έκφραση x y+, οπότε έχουμε τη σημειογραφία *επίθεσης* ή *παρελκόμενου τελεστή* (postfix notation).**

Προσοχή:

- **Καλύτερη πρακτική: να μη χρησιμοποιούνται τελεστές σε μεικτούς τύπους:**

```
out_int = my1_int + my2_int;        // καλό
out_float = my_double / my_int;     // κακό
```

- **!!ΚΙΝΔΥΝΟΣ!!: Όταν γίνεται διαίρεση ακεραίων το αποτέλεσμα είναι το πηλίκο, δηλαδή 5 / 2 = 2 κι όχι 2.5**

- **Για να ληφθεί ως αποτέλεσμα αριθμός κινητής υποδιαστολής, τουλάχιστον ένας από τους τελεστέους πρέπει να είναι αριθμός κινητής υποδιαστολής:**

5.0 / 2 υπολογίζεται ως 2.5

Οι εκφράσεις είναι συχνά φωλιασμένες (nested):

```
(((n+5)<=a) && q)
```

– **Η έκφραση αυτή**

Υπολογίζεται και γίνεται ένας όρος:

```
((n+5)<=a)
```

στη συνέχεια η έκφραση υπολογίζεται και γίνεται ένας όρος:

```
(((n+5)<=a) && q)
```

Προτεραιότητα (precedence) και προσεταιριστικότητα (associativity)

- **Πώς θα υπολογισθεί η έκφραση 17 * 8 - 2;**

Είναι 17*(8-2) ή (17*8)-2 ;

- **Οι ανωτέρω εκφράσεις οδηγούν σε διαφορετικά αποτελέσματα. Κατά συνέπεια απαιτούνται κανόνες.**

- **Η σειρά εφαρμογής των τελεστών ονομάζεται *εφαρμοστική σειρά* (applicative order).**

Προτεραιότητα και προσεταιριστικότητα τελεστών στη C

ΤΕΛΕΣΤΕΣ	ΠΡΟΣΕΤΑΙΡΙΣΤΙΚΟΤΗΤΑ
() [ ] ->	Από αριστερά προς τα δεξιά
! ~ ++ -- + - * & (τύπος) size of	Από δεξιά προς τα αριστερά
* / %	Από αριστερά προς τα δεξιά
+ -	Από αριστερά προς τα δεξιά
<< >>	Από αριστερά προς τα δεξιά
< <= > >=	Από αριστερά προς τα δεξιά
== ! =	Από αριστερά προς τα δεξιά
&	Από αριστερά προς τα δεξιά
^	Από αριστερά προς τα δεξιά
|	Από αριστερά προς τα δεξιά
&&	Από αριστερά προς τα δεξιά
| |	Από αριστερά προς τα δεξιά
?:	Από δεξιά προς τα αριστερά
= += -= *= /= %= &= ^= |= <<= >>=	Από δεξιά προς τα αριστερά
,	Από αριστερά προς τα δεξιά

Προτεραιότητα και προσεταιριστικότητα

Παραδείγματα:

X =17 - 2 * 8 **Απάντηση:** X=17-(2*8) , X=1

Y = 17 - 2 - 8 **Απάντηση:** Y = (17-2)-8, Y=7

Z = 10 + 9 * ((8 + 7) % 6) + 5 * 4 % 3 *2 + 1 (;)

Μπερδευτήκατε; Τότε χρησιμοποιείστε παρενθέσεις στον κώδικά σας.

Τελεστές αύξησης και μείωσης

- **Τελεστής αύξησης (increment operator): ++**

  Αντί για `num = num + 1;`

  γράφουμε `num++;`

- **Τελεστής μείωσης (decrement operator): --**

  Αντί για `num = num - 1;`

  Γράφουμε `num--;`

Μήπως γίνεται πλέον φανερό από πού έλαβε η C++ το όνομά της;

Παράδειγμα: **Στην ακολουθία εκφράσεων που παρατίθεται ακολούθως παρουσιάζεται ενδεικτικά η λειτουργία των προπορευόμενων και παρελκόμενων τελεστών μοναδιαίας αύξησης και μείωσης.**

πρόταση	τιμή x	τιμή y
int x = 10, y = 20;	10	20
++x;	11	20
y = --x;	10	10
y = x-- + y;	9	20
y = y - x++;	10	11

Παράδειγμα: **Να προσδιορισθεί η τιμή των x και z μετά την εκτέλεση κάθε μίας από τις παρακάτω προτάσεις, θεωρώντας ότι, πριν την εκτέλεση της κάθε πρότασης, οι τιμές των x και y είναι το 10 και 20 αντίστοιχα.**

α) `z = ++x + y;`

β) `z = --x + y;`

γ) `z = x++ + y;`

δ) `z = x-- + y;`

Λύση: **Στην περίπτωση του προπορευόμενου τελεστή, το σύστημα πρώτα εκτελεί την αύξηση ή μείωση και μετά χρησιμοποιεί τη νέα τιμή της μεταβλητής στον υπολογισμό της τιμής της έκφρασεις (προτάσεις α και β). Αντίθετα, στην περίπτωση του παρελκόμενου τελεστή το σύστημα πρώτα χρησιμοποιεί την τιμή της μεταβλητής για τον υπολογισμό της τιμής της έκφρασης και μετά εκτελεί την αύξηση ή μείωση της τιμής της μεταβλητής (προτάσεις γ και δ).**

Αποτελέσματα:

Πρόταση	Τιμή x	Τιμή z
z = ++x + y;	11	31
z = --x + y;	9	29
z = x++ + y;	11	30
z = x-- + y;	9	30

Τελεστές ανάθεσης (assignment)

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

- **x* = y + 1;** Αντιστοιχεί στην πρόταση **x = x * (y + 1);** κι **ΟΧΙ** στην πρόταση **x = x * y + 1;**

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

Τελεστές ανάθεσης (συνέχεια)

- **Οι τελεστές ανάθεσης μαζί με τους τελεστές αύξησης/μείωσης γίνονται αιτία δημιουργίας παρενεργειών (side effects), για το λόγο αυτό αναφέρονται και ως παρενεργοί τελεστές (side effect operators).**

- **Οι παρενέργειες αυτές έχουν ως αποτέλεσμα την απροσδιόριστη συμπεριφορά του συστήματος ως προς τον τρόπο υπολογισμού της τιμής της μεταβλητής i σε εκφράσεις όπως: i = n[i++]; ή i = ++i + 1;**

Συσχετιστικοί τελεστές (relational operators)

Τελεστής	Δράση
<	Μικρότερο από
>	Μεγαλύτερο από
<=	Μικρότερο ή ίσον από
>=	Μεγαλύτερο ή ίσον από
==	Ίσο
!=	διάφορο

Το αποτέλεσμα είναι πάντοτε είτε **ΑΛΗΘΕΣ** (TRUE) είτε **ΨΕΥΔΕΣ** (FALSE)

Συσχετιστικοί τελεστές

- ***Παράδειγμα:***
  - **Η τιμή της έκφρασης (3 < 2) είναι ΨΕΥΔΗΣ**
  - **Η τιμή της έκφρασης (2 = = 2) είναι ΑΛΗΘΗΣ**
- **Στη C (και σε πολλές άλλες γλώσσες),**
  - **Η τιμή ΑΛΗΘΗΣ αντιστοιχεί στον ακέραιο 1**
  - **Η τιμή ΨΕΥΔΗΣ αντιστοιχεί στον ακέραιο 0**

Συσχετιστικοί – Αριθμητικοί Τελεστές:

- **ΚΑΙ ΟΙ ΔΥΟ χρησιμοποιούν αριθμητικές εισόδους:**

Παράδειγμα: **(num < 10) και (num + 10)**
(όπου num είναι μία ακέραια μεταβλητή)

- **Ωστόσο, οι συσχετιστικές έξοδοι είναι *μόνο* TRUE/FALSE.**
  - **(number < 10) δίνει TRUE ή FALSE (0/1)**
  - **(number + 10) δίνει οποιοδήποτε αριθμό**

Λογικοί τελεστές

Διαφορετικοί από τους συσχετιστικούς τελεστές καθώς έχουν εισόδους True/False και εξόδους *True/False*.

Τελεστής	Δράση
&&	AND
||	OR
!	NOT

Πίνακας αληθείας

p	q	p&&q (and)	p||q (or)	!p (not)
T	T	T	T	F
T	F	F	T	F
F	T	F	T	T
F	F	F	F	T

Λογικοί τελεστές

Παραδείγματα:

```
int x,y;
x=10;
y=-8;
```

Υπολογισμός των παρακάτω εκφράσεων:

(x+5) < (12-y) (10+5) < (12- (-8))

15< 20 → **TRUE**

(x>5) || (y>10) (10>5) || (-8>10)

(**TRUE**) || (**FALSE**) → **TRUE**

Τελεστής μετατροπής τύπου (typecasting)

Ο τελεστής μετατροπής τύπου ή *cast* τελεστής, όπως αποκαλείται, είναι μοναδιαίος κι έχει τη μορφή (τύπος δεδομένων), π.χ. (float). Τοποθετείται μπροστά από μία έκφραση για να μετατρέψει την τιμή της στον περικλειόμενο σε παρενθέσεις τύπο. Η μετατροπή ισχύει αποκλειστικά στο σημείο εφαρμογής της, όπως φαίνεται στο ακόλουθο παράδειγμα:

Παράδειγμα:

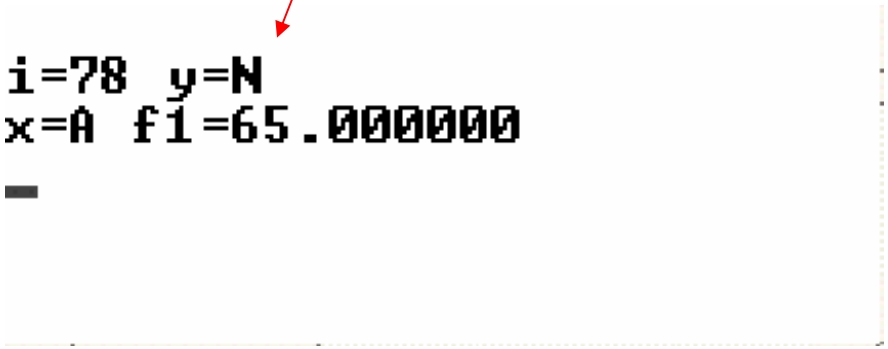
```
int i,j;
float f1,f2,f3;
i=5; j=2;

f1 = i/j + 0.5;          /* αποτέλεσμα: 2.5 */
f2 = (float)i/(float)j + 0.5;   /* αποτέλεσμα : 3.0 */
f3 = i/j + 0.5;          /* αποτέλεσμα : 2.5 */
```

μετατροπή των i και j σε float

Παράδειγμα: **Στον κώδικα που ακολουθεί αποδεικνύεται ότι η μετατροπή τύπου ισχύει για όλους τους τύπους δεδομένων.**

```
#include <stdio.h>
main(){
  char x='A',y;
  int i=78;
  float f1;
  y=(char)i;
  printf( "\ni=%d y=%c\n",i,y );
  f1=(float)x;
  printf( "x=%c f1=%f\n",x,f1 );
}
```

Ο ASCII χαρακτήρας με δεκαδικό ισοδύναμο 78



Τελεστής sizeof

Ο τελεστής *sizeof* είναι μοναδιαίος και δρα:

α) **σε έκφραση, π.χ. sizeof(x+y) και**

β) **σε τύπο δεδομένων, πχ. sizeof(int)**

Σε κάθε περίπτωση, επιστρέφει τον αριθμό των bytes που η τιμή τής έκφρασης ή ο τύπος των δεδομένων καταλαμβάνει στη μνήμη. Προσοχή θα πρέπει να δοθεί στο γεγονός ότι το σύστημα δεν υπολογίζει την τιμή της έκφρασης κι έτσι πιθανή ύπαρξη παρενεργειών τελεστών δε δημιουργεί παρενέργειες. Μπορεί να βρεθεί το μέγεθος σε bytes ενός πίνακα χρησιμοποιώντας τον τελεστή *sizeof*. Για παράδειγμα, αν θεωρηθεί ο πίνακας int ar[5]; η έκφραση sizeof(ar) δίνει τιμή 20 επειδή ο πίνακας αποτελείται από 5 ακεραίους των 4 bytes.

Τελεστής sizeof (συνέχεια)

Στη *sizeof* θα πρέπει να περιλαμβάνεται μόνο το όνομα του πίνακα. Αν περιληφθεί δείκτης ενός στοιχείου, τότε θα εξαχθεί το μέγεθος του στοιχείου. Για παράδειγμα, η έκφραση sizeof(ar[0]) δίνει τιμή 4. Χρησιμοποιώντας ένα συνδυασμό των παραπάνω μπορεί να βρεθεί ο αριθμός των στοιχείων του πίνακα. Η έκφραση sizeof(ar)/sizeof(ar[0]) δίνει 5, τον αριθμό δηλαδή των στοιχείων του πίνακα ar.

Παράδειγμα: **Ο κώδικας που ακολουθεί δίνει το μέγεθος των 4 βασικών τύπων δεδομένων της C.**

```
#include <stdio.h>
void main()  {
  printf( "\nsize of char = %d",sizeof(char) );
  printf( "\nsize of int = %d",sizeof(int) );
  printf( "\nsize of float = %d",sizeof(float) );
  printf( "\nsize of double = %d",sizeof(double) );
}
```

```
size of char = 1
size of int = 4
size of float = 4
size of double = 8
```