Η ΓΛΩΣΣΑ C

Μάθημα 4:

Τελεστές και η Δομή Ελέγχου (if…else)

Δημήτρης Ψούνης

www.psounis.gr

Περιεχόμενα Μαθήματος

Α. Τελεστές

1. Παραστάσεις στην C

1. Απλές Παραστάσεις

- Για την ώρα έχουμε δει 3 μορφές παραστάσεων που μπορούμε να χρησιμοποιήσουμε σε ένα πρόγραμμα:
  - Μεταβλητές (π.χ. x,y,z)
    - (όπου είδαμε ότι πρέπει να τις δηλώσουμε πρώτα ορίζοντας τον τύπο δεδομένων τους)
  - Αριθμητικές Σταθερές (π.χ. 5, 8.0, 3.2543)
    - (όπου χρησιμοποιούμε απ'ευθείας αριθμούς στο πρόγραμμά μας και ο διαχωρίσμος είναι ότι οι ακέραιοι δεν έχουν υποδιαστολή (.) ενώ οι δεκαδικοί έχουν υποδιαστολή)
  - Συμβολικές Σταθερές (π.χ. PI, N)
    - (που τις έχουμε ορίσει είτε με την οδηγία #define είτε με την λέξη κλειδί const)

Α. Τελεστές

1. Παραστάσεις στην C

2. Σύνθετες Παραστάσεις

- Μπορούμε να ορίσουμε με χρήση των απλών παραστάσεων, πιο σύνθετες παραστάσεις έτσι ώστε:
  - Να κάνουμε μαθηματικές πράξεις
    - Όπου χρησιμοποιούμε τελεστές που κάνουν πράξεις (όπως π.χ το + για την πρόσθεση και το / για την διαίρεση)
  - Να κάνουμε συγκρίσεις:
    - Όπου χρησιμοποιουμε τελεστές που κάνουν σύγκριση (όπως π.χ. το <= για το μικρότερο ή ίσο)
  - Να κάνουμε λογικές πράξεις:
    - Όπως π.χ. το || το οποίο θα εκτελεί την λογική πράξη OR μεταξύ των παραστάσεων στις οποίες εφαρμόζεται.
  - Να καταχωρούμε μία τιμή σε μία μεταβλητή:
    - Όπου χρησιμοποιούμε το =, για να αποθηκεύσουμε το αποτέλεσμα μιας παράστασης σε μια μεταβλητή

Α. Τελεστές
2. Ο Τελεστής Καταχώρησης

- Ο **τελεστής καταχώρησης (εκχώρησης)** είναι το **σύμβολο ίσον (=)**
- Η σύνταξη είναι:

```
όνομα_μεταβλητής = παράσταση;
```

  - και η λειτουργία είναι:
    - Υπολογίζεται η τιμή της παράστασης (που μπορεί να είναι ένας περίπλοκος υπολογισμός, όχι απαράιτητα μόνο μία αριθμητική σταθερά)
    - Αποθηκεύεται το αποτέλεσμα στην μεταβλητή που έχει όνομα `όνομα_μεταβλητής`
- Παραδείγματα:

```
x=1+4;
```

  - Θα έχει σαν αποτέλεσμα να αποθηκευτεί (καταχωρηθεί) στο x η τιμή 5.

```
x=16/(3+1);
```

  - Θα έχει σαν αποτέλεσμα να αποθηκευτεί (καταχωρηθεί) στο x η τιμή 4.

Α. Τελεστές
3. Μαθηματικοί Τελεστές
1. Διμελείς μαθηματικοί Τελεστές

- Οι **μαθηματικοί τελεστές** εκτελούν τις συνηθισμένες μαθηματικές πράξεις
- Είναι οι εξής:

Ονομα Τελεστή	Σύμβολο	Χρήση	Παράδειγμα
Πρόσθεση	+	ΠΑΡ+ΠΑΡ	x+y
Πολ/μός	*	ΠΑΡ*ΠΑΡ	x*y
Αφαίρεση	-	ΠΑΡ-ΠΑΡ	x-y
Διαίρεση	/	ΠΑΡ/ΠΑΡ	x/y
Υπόλοιπο Διαίρεσης	%	ΠΑΡ%ΠΑΡ	x%y

  - Όπου ΠΑΡ είναι μία μαθηματική παράσταση

- Οι τελεστές αυτοί λέγονται **διμελείς** διότι έχουν αριστερό και δεξί μέρος (δύο μέλη)!

Α. Τελεστές
3. Μαθηματικοί Τελεστές
1. Διμελείς μαθηματικοί Τελεστές

- Μεταγλωττίστε και εκτελέστε το ακόλουθο πρόγραμμα prakseis.c
  - Υπάρχουν ακόμη κάποια σημεία που θα μάθουμε στην διάρκεια του μαθήματος, όπως η σύνταξη της εντολής if…else if…else και ο τελεστής ισότητας (==).
  - Παρατηρήστε ιδιαίτερα πως λειτουργεί η πράξη της διαίρεσης, π.χ. Για την πράξη **7/3**.
  - Δείτε ότι **το αποτέλεσμα της πράξης είναι 2!!**
  - Αυτό συμβαίνει διότι αποθηκεύεται το αποτέλεσμα μιας διαίρεσης σε έναν ακέραιο αριθμό (όχι πραγματικό).
  - Στην περίπτωση αυτή θα αποθηκεύεται μόνο το ακέραιο μέρος του αριθμού

Α. Τελεστές
3. Μαθηματικοί Τελεστές
1. Διμελείς μαθηματικοί Τελεστές

```
/* prakseis.c: Anadeiknyei tin xrisi twn dimelwn mathimatikwn praksewn */

#include <stdio.h>

main()
{
   int x,y,z;
   int telestis;

   printf("\nDwste ton 1o akeraio: ");
   scanf("%d",&x);
   printf("Dwste ton 2o akeraio: ");
   scanf("%d",&y);
   printf("Dwste ton telesti\n 0 gia +\n 1 gia -\n 2 gia /\n 3 gia *\n 4 gia %");
   printf("\nEpilogi? ");
   scanf("%d",&telestis);

   if (telestis==0)
   {
      z=x+y;
      printf("%d+%d=%d",x,y,z);
   }
   else if (telestis==1)
   {
      z=x-y;
      printf("%d-%d=%d",x,y,z);
   }
```

A. Τελεστές
3. Μαθηματικοί Τελεστές
1. Διμελείς μαθηματικοί Τελεστές

```
   else if (telestis==2)
   {
      z=x*y;
      printf("%d*%d=%d",x,y,z);
   }
   else if (telestis==3)
   {
      z=x/y;
      printf("%d/%d=%d",x,y,z);
   }
   else if (telestis==4)
   {
      z=x%y;
      printf("%d\%%d=%d",x,y,z);
   }

}
```


A. Τελεστές
3. Μαθηματικοί Τελεστές
2. Μονομελείς μαθηματικοί Τελεστές

- Οι **μονομελείς μαθηματικοί τελεστές** είναι 2: το ++ και το --
- Εκτελούν μια πράξη που είναι εξαιρετικά συνηθισμένη όταν γράφουμε ένα πρόγραμμα, την αύξηση μιας μεταβλητής κατά 1 και τη μείωση μιας μεταβλητής κατά 1.

Ονομα Τελεστή	Σύμβολο	Χρήση	Παράδειγμα
Μοναδιαία Αύξηση	++	MET++ ή ++MET	i++ ++j
Μοναδιαία Μείωση	--	MET— ή --MET	i— --i --j

  - Όπου MET είναι το όνομα μιας μεταβλητής.
- Παρατηρήστε ότι υπάρχουν 2 χρήσεις:
  - Η προθεματική (πρώτα ο τελεστής και μετά η μεταβλητή)
  - Η μεταθεματική (πρώτα η μεταβλητή και μετά ο τελεστής)

A. Τελεστές
3. Μαθηματικοί Τελεστές
2. Μονομελείς μαθηματικοί Τελεστές

- Η μοναδιαία αύξηση ως εντολή:

```
++x;
```

  - Έχει το ίδιο αποτέλεσμα με την εντολή:

```
x=x+1;
```

  - Και ακριβώς το ίδιο αποτέλεσμα έχει και η εντολή x++;

```
x++;
```

- Η μοναδιαία μείωση ως εντολή:

```
++x;
```

  - Έχει το ίδιο αποτέλεσμα με την εντολή:

```
x=x+1;
```

  - Και ακριβώς το ίδιο αποτέλεσμα έχει και η εντολή x--;

```
x++;
```


A. Τελεστές
3. Μαθηματικοί Τελεστές
2. Μονομελείς μαθηματικοί Τελεστές

- Στην προθεματική μορφή
  - ΠΡΩΤΑ γίνεται η αύξηση κατά 1 και έπειτα υπολογίζεται η παράσταση.
  - Π.χ. Στο τμήμα κώδικά:

```
x=1;
y=++x;
```

  - Στην 2η γραμμή:
    - Πρώτα γίνεται η αύξηση του x κατά 1, άρα το x γίνεται 2.
    - Έπειτα υπολογίζεται η παράσταση άρα αποθηκεύεται στο y η τιμή 2.
- Στην μεταθεματική μορφή
  - ΠΡΩΤΑ υπολογίζεται η παράσταση (σαν να μην υπάρχει το ++), έπειτα αυξάνεται η τιμή της μεταβλητής.
  - Π.χ. Στο τμήμα κώδικά:

```
x=1;
y=x++;
```

  - Στην 2η γραμμή:
    - Πρώτα υπολογίζεται η παράσταση άρα αποθηκεύεται στο y η τιμή 1

A. Τελεστές
3. Μαθηματικοί Τελεστές
2. Μονομελείς μαθηματικοί Τελεστές

- Προσπαθήστε να βρείτε την εκτύπωση του προγράμματος, προτού το μεταγλωττίσετε και το τρέξετε!

```
/*monadiaioi.c: Anadeikniei tin leitoyrgia twn monadiaiwn telestwn*/

#include <stdio.h>

main()
{
   int x=1;
   int y=1;

   printf("%d,%d",x++,++y);
   printf("\n%d,%d",x++,++y);
   printf("\n%d,%d",x++,++y);
   printf("\n%d,%d",x,y);

}
```


A. Τελεστές
3. Μαθηματικοί Τελεστές
3. Προτεραιότητα Μαθηματικών Τελεστών

- Σε μία περίπλοκη παράσταση, είναι σημαντικό να γνωρίζουμε με ποια σειρά γίνονται οι πράξεις, δηλαδή να ορίσουμε την **προτεραιότητα των τελεστών**

Προτεραιότητα	Τελεστές
1.	++,--
2.	*, / , %
3.	+, -

  - Κατά σειρά προτεραιότητας πρώτα οι μοναδιαίοι, έπειτα πολλαπλασιασμός, διαίρεση και mod, και τέλος πρόσθεση αφαίρεση
- Αν σε μία παράσταση υπάρχουν σύμβολα με την ίδια προτεραίοτητα, τότε οι πράξεις εφαρμόζονται από αριστερά προς τα δεξιά!

- Έτσι π.χ. Η παράσταση: `z=x+y/x-y`
- Θα κάνει τις πράξεις με την εξής σειρά: `z=(x+(y/x))-y`

Συμβουλή: Χρησιμοποιούμε παρενθέσεις για να μην κάνουμε λάθος με την σειρά με την οποία θέλουμε να γίνουν οι πράξεις!

A. Τελεστές
4. Σχεσιακοί Τελεστές
1. Ορισμοί

- Οι σχεσιακοί τελεστές συγκρίνουν δύο τιμές και επιστρέφουν 1 (που αντιστοιχεί στο ΑΛΗΘΕΣ – αν ισχύει η σύγκριση) και 0 (που αντιστοιχεί στο ΨΕΥΔΕΣ – αν δεν ισχύει η συνθήκη)
- Είναι οι εξής:

Ονομα Τελεστή	Σύμβολο	Χρήση	Παράδειγμα
Ισον	==	ΠΑΡ==ΠΑΡ	x==y
Μεγαλύτερο	>	ΠΑΡ>ΠΑΡ	x>y
Μεγαλύτερο ή ίσο	>=	ΠΑΡ>=ΠΑΡ	x>=y
Μικρότερο	<	ΠΑΡ<ΠΑΡ	x<y
Μικρότερο ή ίσο	<=	ΠΑΡ<=ΠΑΡ	x<=y
Διάφορον	!=	ΠΑΡ!=ΠΑΡ	x!=y

  - Όπου ΠΑΡ είναι μία μαθηματική παράσταση

A. Τελεστές
4. Σχεσιακοί Τελεστές
1. Ορισμοί

- Παραδείγματα:

```
8+1==9
```
  - Αποτέλεσμα: 1 (ΑΛΗΘΕΣ)

```
6*2==5+4+3
```
  - Αποτέλεσμα: 1 (ΑΛΗΘΕΣ)

```
5%2 != 5/2
```
  - Αποτέλεσμα: 0 (ΨΕΥΔΕΣ)

```
7>9
```
  - Αποτέλεσμα: 0 (ΨΕΥΔΕΣ)

- Σημαντικό! Λέμε ότι το αποτέλεσμα **επιστρέφεται**
  - Αυτό σημαίνει ότι η παράσταση ενός σχεσιακού τελεστή, υπολογίζεται και έπειτα αντικαθίσταται με την τιμή 0 ή 1. Έτσι π.χ. η εντολή:

```
z=(1<2);
```
  - Έχει ως αποτέλεσμα να αποθηκευτεί στην z η τιμή 1.

A. Τελεστές
4. Σχεσιακοί Τελεστές
1. Ορισμοί

- Σχετικά με τους σχεσιακούς τελεστές:
  - Το συνηθέστερο είναι να γράφουμε μια παράσταση σύγκρισης δύο παραστάσεων μέσα σε μια εντολή συνθήκης if…else… όπου ανάλογα με το αν αληθεύει ή όχι η συνθήκη, εκτελούμε διαφορετικές εντολές.
  - Ωστόσο είναι σημαντικό να καταλάβουμε ότι το αποτέλεσμα ενός σχεσιακού τελεστή θα είναι είτε 0 είτε 1.

- Προσοχή!
  - Ο έλεγχος ισότητας γίνεται με το == (τελεστής ισότητας) και όχι με το = (που είδαμε ότι είναι ο τελεστής καταχώρησης)

  - Τι θα κάνει άραγε αυτός ο κώδικας;

```
x=(x!=x)<(x==x);
```

  - Μεταγλωττίστε και εκτελέστε το πρόγραμμα της επόμενης διαφάνειας για την εξάσκηση με τους σχεσιακούς τελεστές

A. Τελεστές
4. Σχεσιακοί Τελεστές
1. Ορισμοί

```
/*sxesiakoi.c: Anadeikniei tin leitoyrgia twn sxesiakwn telestwn*/
#include <stdio.h>

main()
{
   int x=1;
   int y=2;
   int z;

   z=(x>y);
   printf("\nx>y:%d",z);

   z=(x>=y);
   printf("\nx>=y:%d",z);

   z=(x==y);
   printf("\nx==y:%d",z);

   z=(x<=y);
   printf("\nx<=y:%d",z);

   z=(x<y);
   printf("\nx<y:%d",z);
}
```


A. Τελεστές
4. Σχεσιακοί Τελεστές
2. Προτεραιότητα Σχεσιακών Τελεστών

- Ακόμη και οι σχεσιακοί τελεστές έχουν **προτεραιότητα**. Αυτό γίνεται γιατί μπορεί να έχουμε εκκεντρικές παραστάσεις που χρησιμοποιούν ταυτόχρονα πολλούς σχεσιακούς τελεστές. Αν και δεν πρόκειται να το συναντήσουμε στην πράξη το αναφέρουμε για λόγους πληρότητας:

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

- Και πάλι ισχύει η προτεραιότητα από αριστερά προς τα δεξιά όταν έχουμε τελεστές ίδιας προτεραιότητας.

- Έτσι π.χ. στην παράσταση:

```
(1<2)<(3>=2);
```

- Θα υπολογιστούν πρώτα στην παράσταση οι παρενθέσεις:

```
1 < 1
```

- Άρα τελικά η τιμή της παράστασης είναι:

```
0
```


A. Τελεστές
5. Λογικοί Τελεστές
1. Ορισμοί

- Οι **Λογικοί τελεστές** αποτελούν την εφαρμογή των λογικών πράξεων **AND**, **OR** και **NOT**
- Είναι οι εξής:

Ονομα Τελεστή	Σύμβολο	Χρήση	Αληθεύει (=1) όταν	Ψευδής (=0) όταν
AND	&&	ΠΑΡ1 && ΠΑΡ2	Η ΠΑΡ1 είναι αληθής (1) **ΚΑΙ** Η ΠΑΡ2 είναι αληθής (1)	Η ΠΑΡ1 είναι ψευδής (1) **Ή** Η ΠΑΡ2 είναι ψευδής (1)
OR	||	ΠΑΡ1 || ΠΑΡ2	Η ΠΑΡ1 είναι αληθής (1) **Ή** Η ΠΑΡ2 είναι αληθής (1)	Η ΠΑΡ1 είναι ψευδής (0) **ΚΑΙ** Η ΠΑΡ2 είναι ψευδής (0)
NOT	!	!ΠΑΡ	Η ΠΑΡ είναι ψευδής (0)	Η ΠΑΡ είναι αληθής (1)

  - Όπου ΠΑΡ είναι μία παράσταση που είναι 0 ή 1 (συνήθως είναι μια παράσταση που χρησιμοποιεί σχεσιακούς τελεστές)

Στην πραγματικότητα: Η C ερμηνεύει ως ψευδές ΜΟΝΟ το 0, και ως αληθές ΟΠΟΙΑΔΗΠΟΤΕ τιμή διάφορη του 0.

A. Τελεστές
5. Λογικοί Τελεστές
1. Ορισμοί

- Παραδείγματα:

```
(5<3) && (3<5)
```

- Αποτέλεσμα: 0 (ΨΕΥΔΕΣ)

```
(7+3==10) || (4<2)
```

- Αποτέλεσμα: 1 (ΑΛΗΘΕΣ)

```
((5+2!=3)&&(1<4)) || (2<4)
```

- Αποτέλεσμα: 0 (ΨΕΥΔΕΣ)

```
!(7<7)
```

- Αποτέλεσμα: 1 (ΑΛΗΘΕΣ)

- Σημαντικό! Λέμε ότι το αποτέλεσμα **επιστρέφεται**
  - Αυτό σημαίνει ότι η παράσταση ενός σχεσιακού τελεστή, υπολογίζεται και έπειτα αντικαθίσταται με την τιμή 0 ή 1. Έτσι π.χ. η εντολή:

```
z=(1<2)&&(2>3);
```

  - Έχει ως αποτέλεσμα να αποθηκευτεί στην z η τιμή 0.

A. Τελεστές
5. Λογικοί Τελεστές
2. Προτεραιότητα Λογικών Τελεστών

- Αν και οι λογικοί τελεστές έχουν προτεραιότητα, καλό θα είναι να την καθορίζουμε και με παρενθέσεις. Η προτεραιότητα (εφόσον αυτή δεν καθορίζεται με παρενθέσεις) είναι:

Προτεραιότητα	Τελεστές
1.	!
2.	&&
3.	||

- Με προτεραιότητα από αριστερά προς τα δεξιά, αν έχουμε δύο φορές εμφάνιση του ίδιου τελεστή.

A. Τελεστές
5. Λογικοί Τελεστές
2. Προτεραιότητα Λογικών Τελεστών

- Προσπαθήστε να βρείτε την εκτύπωση του προγράμματος, προτού το μεταγλωττίσετε και το τρέξετε!

```
/* logikoi.c: Anadeikniei tin leitoyrgia twn logikwn telestwn */

#include <stdio.h>

main()
{
   int x=1;
   int y=2;
   int z;

   z=(y>x) && (x<x);
   printf("%d",z);
   z=(x=x) && (y==y);
   printf("%d",z);
}
```


A. Τελεστές
6. Τελεστές Συντομογραφίας Πράξεων
1. Ορισμοί

- Επειδή κάποιες πράξεις γίνονται πολύ συχνά όταν γράφουμε ένα πρόγραμμα, υπάρχουν κάποιοι τελεστές που είναι συντομογραφίες πιο σύνθετων παραστάσεων:

Ονομα Τελεστή	Σύμβολο	Χρήση	Παράδειγμα	Είναι συντομογραφία της παραστασης:
Αύξηση μεταβλητής	+=	ΜΕΤ+=ΠΑΡ	x+=5;	x=x+5;
Μειωση μεταβλητής	-=	ΜΕΤ-=ΠΑΡ	x-=2;	x=x-2;
Διαίρεση μεταβλητής	/=	ΜΕΤ/=ΠΑΡ	x/=y;	x=x/y;
Πολ/μος μεταβλητής	*=	ΜΕΤ*=ΠΑΡ	x*=a+b;	x=x*(a+b)
Modulo μεταβλητής	%=	ΜΕΤ%=ΠΑΡ	x%=8;	x=x%8;

- Όπου ΜΕΤ είναι μία μεταβλητή και
- ΠΑΡ είναι μια παράσταση που υπολογίζεται σε μία τιμή

Α. Τελεστές
7. Συγκεντρωτικός Πίνακας Προτεραιότητας Τελεστών

- Υπάρχουν ακόμη μερικοί τελεστές που θα τους μάθουμε σε επόμενα μαθήματα:
  - Τελεστές δεικτών: ***, ->,*** και ***&***
  - Τελεστές λογικών πράξεων σε bytes: ***&, |*** και ***^***
  - Τελεστές ολίσθησης: ***>>*** και ***<<***
  - Ο τριαδικός(!) τελεστής ***?:*** (αυτόν θα τον μάθουμε σήμερα)
  - Και μερικοί ακόμη…
- Ωστόσο, ο πίνακας της ακόλουθης διαφάνειας είναι πολύτιμη αναφορά κατά τη συγγραφή των προγραμμάτων μας.

Α. Τελεστές
7. Συγκεντρωτικός Πίνακας Προτεραιότητας Τελεστών

Προτεραιότητα	Τελεστές
1.	`() [] -> .`
2.	`! ++ -- * & (type) sizeof + -`
3.	`* / %`
4.	`+ -`
5.	`<< >>`
6.	`< <= > >=`
7.	`== !=`
8.	`&`
9.	`^`
10.	`|`
11.	`&&`
12.	`||`
13.	`?:`
14.	`= += -= *= /= %= &= ^= |= <<= >>=`
15.	`,`

Β. Η Δομή Ελέγχου
1. Εισαγωγή

- Όπως είδαμε, οι εντολές ενός προγράμματος εκτελούνται σειριακά η μία μετά την άλλη.
- Μέσω της **δομής ελέγχου (if)** έχουμε δικαίωμα να εκτελούμε διαφορετικές εντολές ανάλογα με το αν ικανοποιείται μια συνθήκη.
  - Η συνθήκη συντάσσεται συνήθως με χρήση των λογικών και των σχεσιακών τελεστών που είδαμε νωρίτερα.
- Έτσι έχουμε την δυνατότητα μέσω της δομής ελέγχου, να κάνουμε διαφορετικές ενέργειες ανάλογα με το αν ικανοποιούνται συνθήκες που ορίζουμε εμείς
- Αυτό είναι πάρα πολύ σημαντικό και είναι ένα από τα κύρια χαρακτηριστικά του προγραμματισμού.
  - Άλλο κύριο χαρακτηριστικό είναι η επανάληψη που εκτελούμε πολλές φορές την ίδια ενέργεια.
  - Θα την δούμε σε επόμενο μάθημα.

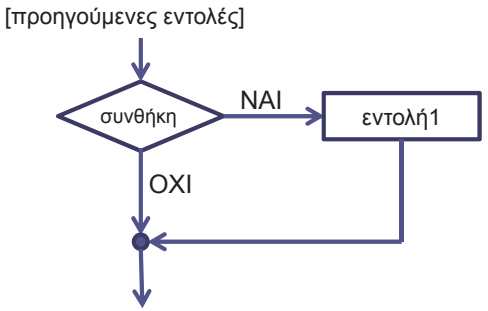
Β. Η Δομή Ελέγχου
2. Πρώτη Χρήση Δομής Ελέγχου: Απλή if

- Η εντολή συνθήκης στην 1η της χρήση συντάσσεται ως εξής:

```
[προηγουμενες εντολές]

if (συνθήκη)
   εντολή1;

[επόμενες εντολές]
```

  - Αφού εκτελεστούν οι προηγούμενες εντολές
  - Ελέγχεται η συνθήκη (συνήθως θα είναι μια παράσταση που θα χρησιμοποιεί κάποιον σχεσιακό/λογικό τελεστή)
    - Αν η συνθήκη είναι ΑΛΗΘΗΣ εκτελείται η εντολή1 και έπειτα οι (επόμενες εντολές)
    - Αν η συνθήκη είναι ΨΕΥΔΗΣ εκτελούνται απευθείας οι (επόμενες εντολές)

ΔΙΑΓΡΑΜΜΑ ΡΟΗΣ ΠΡΟΓΡΑΜΜΑΤΟΣ



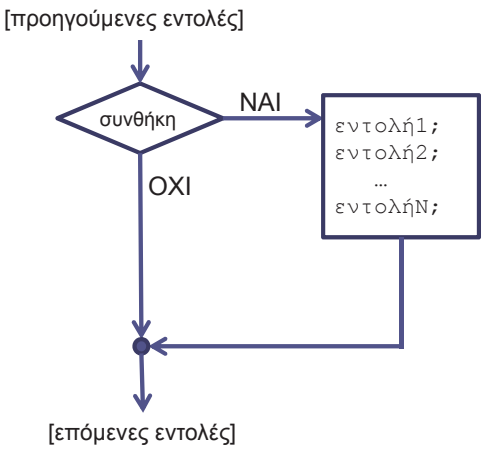
Β. Η Δομή Ελέγχου
2. Πρώτη Χρήση Δομής Ελέγχου: Απλή if

- Αν θέλουμε να εκτελέσουμε περισσότερες από μία εντολές, εφόσον ικανοποιείται η συνθήκη, τότε θα πρέπει να τις βάλουμε μεταξύ αριστερού και δεξιού άγκιστρου

```
[προηγουμενες εντολές]
if (συνθήκη)
{
   εντολή1;
   εντολή2;
   …
   εντολήN;
}
[επόμενες εντολές]
```

Συνοπτικά!
- Αν θέλουμε να τρέξουμε μία εντολή, δεν οφείλουμε να βάλουμε άγκιστρα
- Αν θέλουμε να τρέξουμε περισσότερες από μία εντολές ΟΦΕΙΛΟΥΜΕ να βάλουμε άγκιστρα! (σύνολο εντολών με άγκιστρα λέγεται και **μπλοκ κώδικα**)

ΔΙΑΓΡΑΜΜΑ ΡΟΗΣ ΠΡΟΓΡΑΜΜΑΤΟΣ



Β. Η Δομή Ελέγχου
2. Πρώτη Χρήση Δομής Ελέγχου: Απλή if

Συμβουλή 1: Παρατηρήστε ότι στο συντακτικό της if, γράφουμε την συνθήκη σε παρένθεση και ΔΕΝ ακολουθείται από ερωτηματικό!

Έτσι αν κατά λάθος γράψουμε μια if της μορφής

```
if (x>y);
{
   x=x+1;
   y=y+2;
}
```

Τότε το πρόγραμμα μας δεν θα έχει την επιθυμητή συμπεριφορά!

Συμβουλή 2: Δείτε ότι οι εντολές που είναι μέσα σε if στοιχίζονται λίγο «πιο δεξιά». Αυτό είναι πάγια προγραμματιστική τακτική για να έχει το πρόγραμμα μας μια καλαίσθητη μορφή.

Έχουμε ήδη δει ότι και ο κώδικας μιας συνάρτησης μετατοπίζεται πιο «δεξιά».

Β. Η Δομή Ελέγχου

- Μεταγλωττίστε, εκτελέστε και μελετήστε το παρακάτω πρόγραμμα που δείχνει την 1η χρήση της εντολής if
- Πόσες συγκρίσεις θα γίνουν κατά την εκτέλεση του προγράμματος;

```
/* if_01.c: Deixnei tin xrisi tis if */
#include <stdio.h>

main()
{
   int x;
   int y;

   printf("Dwste ton akeraio x: ");
   scanf("%d",&x);
   printf("Dwste ton akeraio y: ");
   scanf("%d",&y);

   if (x<y)
      printf("\n%d<%d",x,y);

   if (x<=y)
      printf("\n%d<=%d",x,y);

   if (x==y)
      printf("\n%d==%d",x,y);

   if (x>y)
      printf("\n%d>%d",x,y);

   if (x>=y)
      printf("\n%d>=%d",x,y);
}
```

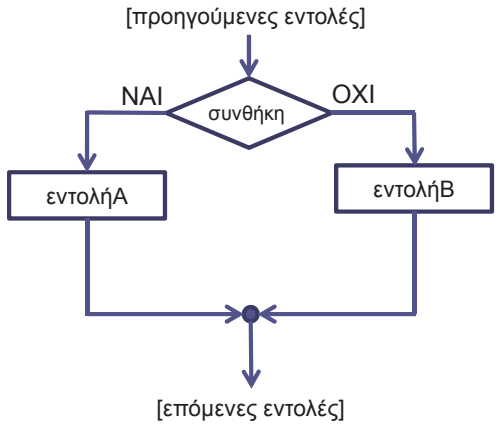

Β. Η Δομή Ελέγχου
3. Δεύτερη Χρήση Δομής Ελέγχου: if…else

- Η δομή ελέγχου, στην 2η της χρήση συντάσσεται ως εξής:

```
[προηγουμενες εντολές]

if (συνθήκη)
   εντολήΑ;
else
   εντολήΒ;

[επόμενες εντολές]
```

- Αφού εκτελεστούν οι προηγούμενες εντολές
- Ελέγχεται η συνθήκη (συνήθως θα είναι μια παράσταση που θα χρησιμοποιεί κάποιον σχεσιακό/λογικό τελεστή)
  - Αν η συνθήκη είναι ΑΛΗΘΗΣ εκτελείται η εντολή Α και έπειτα οι (επόμενες εντολές)
  - Αν η συνθήκη είναι ΨΕΥΔΗΣ εκτελείται η εντολή Β και έπειτα οι (επόμενες εντολές)

ΔΙΑΓΡΑΜΜΑ ΡΟΗΣ ΠΡΟΓΡΑΜΜΑΤΟΣ

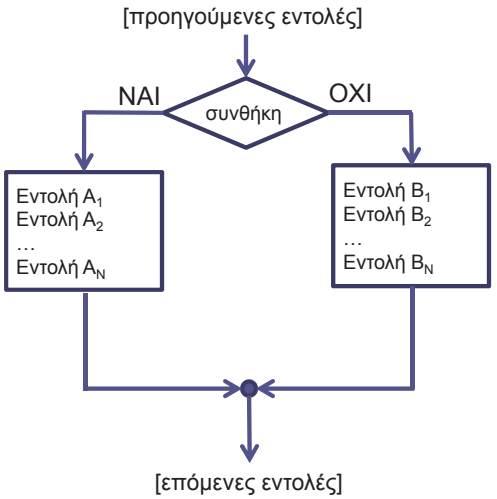


Β. Η Δομή Ελέγχου
3. Δεύτερη Χρήση Δομής Ελέγχου: if…else

- Ομοίως με την πρώτη χρήση μπορούμε να έχουμε πολλές εντολές που εκτελούνται είτε στην if είτε στην else, βάζοντας τις ανάμεσα στα άγκιστρα:

```
[προηγουμενες εντολές]

if (συνθήκη)
{
   εντολήΑ_1;
   ...
   εντολήΑ_N;
}
else
{
   εντολήΒ_1;
   ...
   εντολήΒ_N;
}
[επόμενες εντολές]
```

ΔΙΑΓΡΑΜΜΑ ΡΟΗΣ ΠΡΟΓΡΑΜΜΑΤΟΣ



Β. Η Δομή Ελέγχου
3. Δεύτερη Χρήση Δομής Ελέγχου: if…else

- Μεταγλωττίστε, εκτελέστε και μελετήστε το παρακάτω πρόγραμμα που δείχνει την 1η χρήση της εντολής if
- Πόσες συγκρίσεις θα γίνουν κατά την εκτέλεση του προγράμματος;

```
/* if_02.c: Deixnei tin xrisi tis if..else */
#include <stdio.h>

main()
{
   int x;

   printf("Dwste enan akeraio: ");
   scanf("%d",&x);

   if(x%2==0)
     printf("O arithmos einai artios!");
   else
     printf("O arithmos einai perittos!");

}
```


Β. Η Δομή Ελέγχου
3. Δεύτερη Χρήση Δομής Ελέγχου: if…else

- Το απολύτως τυπικό συντακτικό της εντολής if…else είναι το ακόλουθο (και μόνον αυτό):

```
if (συνθήκη)
   μπλοκ εντολώνΑ
[else
   μπλοκ εντολώνΒ]
```

- Όπου
  - Μπλοκ Εντολών είναι είτε μία εντολή είτε περισσότερες εντολές.
  - Ο συμβολισμός […] δηλώνει ότι το κομμάτι του else είναι προαιρετικό, δηλαδή μπορεί να υπάρχει, μπορεί και να μην υπάρχει.
- Τυπικά:
  - Αυτή η μορφή συγκροτεί ΜΙΑ εντολή (απόφασης).
  - Ωστόσο πρέπει να είμαστε προσεκτικοί ότι η εντολή απόφασης δεν παίρνει ερωτηματικό στο τέλος της σε αντίθεση με τις υπόλοιπες εντολές που έχουμε μάθει

Β. Η Δομή Ελέγχου
4. Εμφωλιασμένες Δομές Ελέγχου

- Η δομή ελέγχου είναι απλά μία εντολή! Άρα μπορούμε να έχουμε π.χ. μία εντολή if μέσα σε μία εντολή if όσες φορές θέλουμε
  - Αυτό αναφέρεται και σαν εμφωλιασμένες εντολές συνθήκης
- Μελετήστε την παραλλαγή προηγούμενου προγράμματος.
- Πόσες συγκρίσεις θα γίνουν στο πρόγραμμα αυτό;

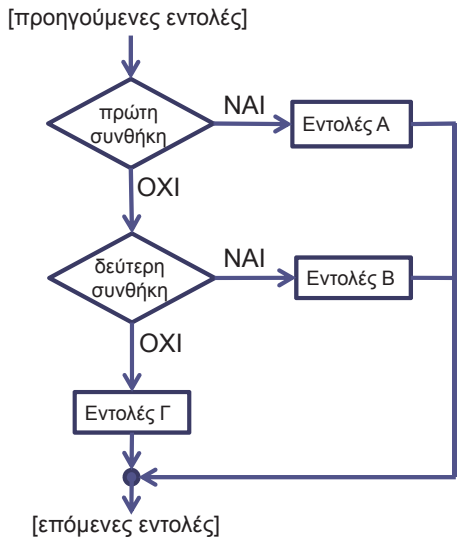
```
/* if_03.c: Deixnei tin xrisi tis if */
#include <stdio.h>

main()
{
   int x;
   int y;

   printf("Dwste ton akeraio x: ");
   scanf("%d",&x);
   printf("Dwste ton akeraio y: ");
   scanf("%d",&y);

   if(x<y)
     printf("Isxyei x<y");
   else
     if(x==y)
       printf("Isxyei x=y");
     else
       printf("Isxyei x>y");

}
```

Β. Η Δομή Ελέγχου
5. Τρίτη Χρήση Δομής Ελέγχου: if…else if … else

- Σε προγράμματα όπως το προηγούμενο υπάρχουν 3 περιπτώσεις που μπορούμε να έχουμε στο πρόγραμμα μας.
- Συνηθίζεται η σύνταξη να γίνεται με τον εξής τρόπο:

```
[προηγούμενες εντολές]
if (πρώτη συνθήκη)
{
   (εντολέςΑ)
}
else if (δεύτερη συνθήκη)
{
   (εντολέςΒ)
}
else
{
   (εντολέςΓ)
}
[επόμενες εντολές]
```

ΔΙΑΓΡΑΜΜΑ ΡΟΗΣ ΠΡΟΓΡΑΜΜΑΤΟΣ



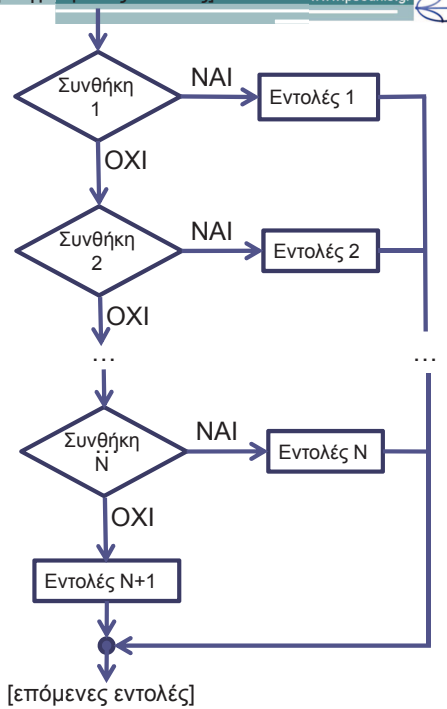
Β. Η Δομή Ελέγχου
6. Γενίκευση για διαδοχικούς ελέγχους

- Ο συλλογισμός για να κάνουμε διαδοχικούς ελέγχους μπορεί να γενικευτεί όπως φαίνεται στο σχήμα π.χ. για να κάνουμε έναν έλεγχο Ν περπτώσεων.
- Ωστόσο για να κάνουμε έλεγχο για ακόμη περισσότερες περιπτώσεις, έχουμε ένα ακόμη εργαλείο, την εντολή switch που θα δούμε σε επόμενο μάθημα.
- Στην επόμενη διαφάνεια το διάγραμμα ροής προγράμματος της χρήσης αυτής.

```
[προηγούμενες εντολές]
if (πρώτη συνθήκη)
{
   (εντολές1)
}
else if (δεύτερη συνθήκη)
{
   (εντολές2)
}
…
else if (συνθήκηΝ)
{
   (εντολέςΝ)
}
else
{
   (εντολέςΝ+1)
}

[επόμενες εντολές]
```


Β. Η Δομή Ελέγχου
6. Γενίκευση

```
[προηγούμενες εντολές]
if (συνθήκη1)
{
   (εντολές1)
}
else if (συνθήκη2)
{
   (εντολές2)
}
…
else if (συνθήκηΝ)
{
   (εντολέςΝ)
}
else
{
   (εντολέςΝ+1)
}

[επόμενες εντολές]
```



Β. Η Δομή Ελέγχου
7. … και ο τελεστής ?:

- Για πιο προχωρημένους (και ελαφρώς πειραγμένους) προγραμματιστές προσφέρεται και μια συντομογραφία της εντολής if…else…
- Έτσι αντί να γράφουμε:

```
if (X)
   A
else
   B
```

- Μπορούμε να γράψουμε με χρήση του τριαδικού τελεστή ?: την εξής εντολή:

```
X ? A : B
```

- Το ακόλουθο πρόγραμμα επιδεικνύει (πολύ προχωρημένη) χρήση του τελεστή ?:
  - Μόνο για τους πολύ φιλόδοξους!

Β. Η Δομή Ελέγχου
7. … και ο τελεστής ?:

```
/* ternary.c:Eksereuna ton telesti ?: */
#include <stdio.h>

main()
{
   int x,y,z;

   /* Kai i if epistrefei mia timi */
   x=0; y=1;
   if(x<y) z=0;
   else z=1;
   printf("%d",z);

   /* Isodynamos tropos me ton telesi ?: */
   x=1; y=0;
   z = x<y ? 0 : 1;
   printf("%d",z);

   /* kai mporoyme na to xrisimopoiisoyme ws orisma. */
   printf("%d",x<y?0:1);

}
```


Γ. Ασκήσεις
Εφαρμογή 1

Γράψτε ένα πρόγραμμα C το οποίο θα ελέγχει και θα εκτυπώνει κατάλληλο μήνυμα αν κάποιος είναι ανήλικος (<18), ενήλικος (18…65), συνταξιούχος (>65)

Γ. Ασκήσεις
Εφαρμογή 2

Γράψτε ένα πρόγραμμα C το οποίο θα λαμβάνει ως είσοδο δύο ακέραιους αριθμούς και θα τυπώνει τον μεγαλύτερο από τους 2.

Γ. Ασκήσεις
Εφαρμογή 3

Γράψτε ένα πρόγραμμα C το οποίο θα λαμβάνει ως είσοδο τρεις ακέραιους αριθμούς και θα τυπώνει τον μεγαλύτερο από τους 3.

Γ. Ασκήσεις
Εφαρμογή 4

- Γράψτε ένα πρόγραμμα C το οποίο θα λαμβάνει ως είσοδο τρεις ακέραιους αριθμούς και θα τους τυπώνει σε αύξουσα σειρά.
- Παραθέτουμε ένα παράδειγμα εκτέλεσης του προγράμματός σας:

```
Eisagete ton 1o arithmo: 8
Eisagete ton 2o arithmo: 5
Eisagete ton 3o arithmo: 9
H diataksi einai: 5,8,9
```


Γ. Ασκήσεις
Εφαρμογή 5

- Γράψτε ένα πρόγραμμα C το οποίο
  - Να δέχεται από τον χρήστη σαν είσοδο έναν ακέραιο αριθμό, που θα απεικονίζει δευτερόλεπτα.
  - Να υπολογίζει πόσες ώρες, λεπτά και δευτερόλεπτα είναι η είσοδος του χρήστη.
- Η παρακάτω είναι η επιθυμητή έξοδος αν ο χρήστης εισάγει το 1000:

```
Eisagete plithos deyteroleptwn: 5000
Wres: 1
Lepta: 23
Deyterolepta: 20
```

- Υπόδειξη:
  - Μελετήστε πως μπορείτε να χρησιμοποιήσετε τους τελεστές / και % για να επιτύχετε το επιθυμητό αποτέλεσμα!

Γ. Ασκήσεις
Εφαρμογή 6

Σε μια (εσφαλμένη) εκδοχή του μπαρμπουτιού δύο παίκτες ρίχνουν δύο ζάρια (ο καθένας) και νικάει ο παίκτης με μεγαλύτερο άθροισμα στα ζάρια.
Γράψτε ένα πρόγραμμα σε C που θα ζητάει από το χρήστη να εισάγει το αποτέλεσμα κάθε ζαριού κάθε παίκτη (έστω Α και Β) και να τυπώνει στην οθόνη τον παίκτη που νίκησε.

Γ. Ασκήσεις
Εφαρμογή 7

- Ένα έτος είναι δίσεκτο αν είναι:
  - Πολλαπλάσιο του 4 με την ειδική περίπτωση ότι
    - Αν είναι πολλαπλάσιο του 100 και όχι του 400 τότε δεν είναι δίσεκτο.
- Γράψτε ένα πρόγραμμα C που θα δέχεται σαν είσοδο χρήστη το έτος και θα τυπώνει αν είναι δίσεκτο ή όχι.
- Παραθέτουμε 4 εκτελέσεις που το πρόγραμμα σας πρέπει να απαντά με τον ίδιο τρόπο:

```
Eisagete etos: 2018
Einai Disekto!
```

```
Eisagete etos: 2000
Einai Disekto!
```

```
Eisagete etos: 2100
Den einai Disekto!
```

```
Eisagete etos: 2021
Den einai Disekto!
```

- Υπόδειξη:
  - Για να ελέγξουμε αν ένας αριθμός x είναι π.χ. πολλαπλάσιο του 4, θα πρέπει το υπόλοιπο της διαίρεσης του x με το 4 να είναι 0.