ΗΥ-150
Προγραμματισμός

Εντολές Ελέγχου Ροής

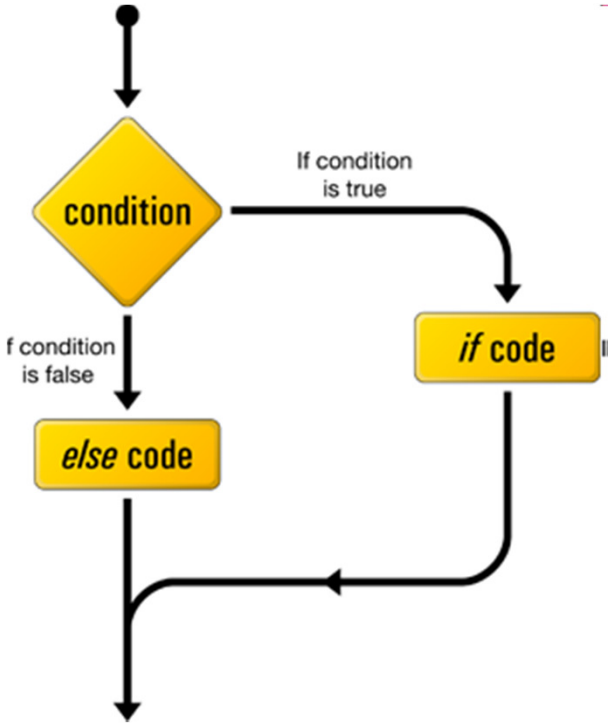
Σειριακή εκτέλεση εντολών

- Όλα τα προγράμματα «γράφονται» χρησιμοποιώντας 3 είδη εντολών:
  - Σειριακές εντολές (sequential – built in C)
  - Εντολές απόφασης (if, if/else, switch)
    - Περιλαμβάνει έκφραση ελέγχου σε μορφή λογικής πρότασης.
    - Ανάλογα με το εάν η πρόταση είναι αληθής ή ψευδής εκτελείται ένα, αντίστοιχο, σύνολο εντολών.
  - Εντολές επανάληψης (for, while, do/while)

if εντολή διακλάδωσης

Συντάσσεται ως εξής:

```
if (condition)
{
      ... // block A
}
else
{
      ... // block B
}
```



Εάν η συνθήκη (condition) είναι αληθής, εκτελείται το block των εντολών που περικλείεται μεταξύ των πρώτων {} (block A).

Αλλιώς, εκτελείται το block των εντολών μετά το **else** (block B).

if

Το κάθε block μπορεί να αποτελείται από καμμία, μία, ή περισσότερες εντολές. Στην περίπτωση που περιλαμβάνει μία μόνο εντολή μπορούν να παραληφθούν τα άγκιστρα ({}) που την περικλείουν.

```
if (val > max)
{
  max = val;
}
else
{
  max = 1000.0;
  i=i+1;
}
```

if

Στην περίπτωση που το **else** block είναι κενό μπορεί να παραληφθεί ολόκληρο:

```
if (condition)
{
   ...
}
```

Εάν το αποτέλεσμα της έκφρασης (*συνθήκης*) είναι αληθής τότε εκτελούνται οι εξαρτώμενες εντολές αλλιώς η εκτέλεση συνεχίζει μετά το if block (αγνοούνται οι εξαρτώμενες εντολές)

Παράδειγμα

```
int main()
{
        int a, b;
        printf("dwse to a\n");
        scanf("%d", &a);
        printf("dwse to b\n");
        scanf("%d", &b);
        if (a < b)
                printf("%d\n", a);
        else
                printf("%d\n", b);
        return 0;
}
```

if

- Κάθε block μπορεί να περιλαμβάνει οποιεσδήποτε άλλες εντολές και βέβαια άλλο **if**.
- Η ***if*** *(condition)* ***{…} else {…}*** είναι μία εντολή.
- Στην περίπτωση ενός εσωκλειόμενου **if** , θέλει ιδιαίτερη προσοχή το επόμενο **else** του κώδικα. Το κάθε **else** ταιριάζει με το αμέσως προηγούμενό του **if** στο ίδιο block.

Η εντολή max = 10; εκτελείται όταν το i είναι 0 και δεν ισχύει το (val > max) και όχι όταν δεν ισχύει το (i == 0). Η χρήση των αγκίστρων επιβάλει την πρόθεση του προγραμματιστή:

```
if (i == 0)
  if (val > max)
     max = val;
  else
     max = 10;
```

```
if (i == 0)
{
  if (val > max)
     max = val;
}
else
  max = 10;
```

Εντολές υπό Συνθήκη - if

εντολή <- έκφραση;
εντολή <- { σειρά εντολών }

```
if (έκφραση)                    //Αν η έκφραση έχει τιμή διάφορη του 0 (!0),
    εντολή1                     //εκτελείται η εντολή 1

if (έκφραση)                    //Αν η έκφραση έχει τιμή διάφορη του 0 (!0),
    εντολή1                     //εκτελείται η εντολή 1 αλλιώς η εντολή2
else
    εντολή2

if (έκφραση1)                   //Αν η έκφραση1 έχει τιμή διάφορη του 0 (!0),
    εντολή1                     //εκτελείται η εντολή 1
else if (έκφραση2)              //αλλιώς αν η έκφραση2 έχει τιμή διάφορη του 0
    εντολή2                     //εκτελείται η εντολή 2
else                            //αλλιώς εκτελείται η εντολή 3
    εντολή3
```

if

διάβασε τους ακέραιους *α, β*

αν (*α* < *β*) **τότε**

τύπωσε *α*

αλλιώς

τύπωσε *β*

if

αν (*συνθήκη*) **τότε**

εντολές Α

αλλιώς

εντολές Β

συνθήκη : **λογική παράσταση** που αποτιμάται σε *true-αλήθεια* (**1**) ή *false-λάθος* (**0**)

Λογικές παραστάσεις

- Μια **λογική παράσταση (ΛΠ)** είναι ανάλογη μιας μαθηματικής παράστασης, με τη διαφορά ότι το αποτέλεσμα μπορεί να είναι μόνο **αλήθεια (1)** ή **λάθος (0)**
- Λογικές παραστάσεις συνθέτονται χρησιμοποιώντας σχεσιακούς τελεστές
- Δυο λογικές παραστάσεις μπορούν να συνδυαστούν με ένα **λογικό τελεστή**

Αληθής / Ψευδής λογική πρόταση στη C

Η C αποτιμά οποιοδήποτε αποτέλεσμα έκφρασης σε συνθήκη που έχει τιμή διάφορη του 0 ως αληθή (TRUE)

```
if (x)
{
    printf("TRUE");   /* x= ..., -3, -2, -1, 1, 2, 3, ... */
}
else
{
    printf("FALSE"); /* x= 0 */
}
```

Σχεσιακοί Τελεστές

- Δυαδικοί Τελεστές
  - **< μικρότερο από**
  - **> μεγαλύτερο από**
  - **<= μικρότερο ή ίσο με**
  - **>= μεγαλύτερο ή ίσο με**
  - **== ίσο με**
  - **!= διάφορο του**
- Αποτιμούνται σε 0 (ψευδής) ή 1 (αληθής)
- Τύποι τελεσταίων int, char, float, double

Παραδείγματα

- (x < y)
- t = (x < y); Αν (t) τότε printf("true\n");
- k = (i >= 8);
- a = (b != c);
- (f == 2.3456)
  - ΠΡΟΣΟΧΗ ποτέ δεν συγκρίνουμε δυο float ή double με ισότητα
  - Εναλλακτικά: ((f1-f2) < .00001)
- ('a' >= 'd')
  - 'a'<'b'<..<'z'
  - 'A'<'B'<..<'Z'
  - '0'<'1'<..<'9'

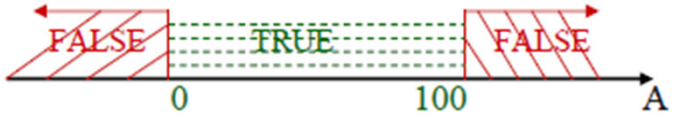
Θυμηθείτε πως οι χαρακτήρες είναι αριθμοί και ερμηνεύονται σύμφωνα με τον πίνακα ASCII. Στον πίνακα αυτό, οι χαρακτήρες είναι οργανωμένοι με αλφαβητική σειρά.

Λογικοί Τελεστές

- Συνδυάζουν δύο λογικές παραστάσεις σε μια ***σύνθετη λογική παράσταση***
  - **&&** σύζευξη, δυαδικός τελεστής (***AND***)
  - || διάζευξη, δυαδικός τελεστής (***OR***)
  - **!** άρνηση, μοναδιαίος τελεστής (***NOT***)
- Αποτιμούνται σε **0** ή **1**
  - **0** (δεν ισχύει, ψευδής ή **false**)
  - **1** (ισχύει, αληθής ή **true**)

Εκφράσεις με Λογικούς και Σχεσιακούς Τελεστές

(A>=0 && A<=100)



(s<10 || s>100)

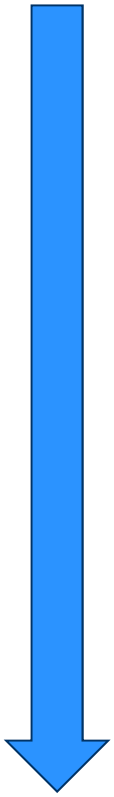
!(s<10 || s>100)

(s>=10 && s<=100)

(i<10 && j==1)

Προτεραιότητες Τελεστών

υψηλότερη



χαμηλότερη

()
! | &
*** / %**
+ -
< <= >= >
== !=
&&
| |
=

Αποτίμηση Λογικών Εκφράσεων

Η αποτίμηση αρχίζει από τα αριστερά και προχωρεί μέχρι το σημείο που χρειάζεται να προχωρήσει (lazy evaluation), π.χ.

- 0 && E *αποτιμάται σε 0*
  - η αποτίμηση της έκφρασης E δεν χρειάζεται
- 1 || E *αποτιμάται σε 1*
  - η αποτίμηση της έκφρασης E δεν χρειάζεται

Αποτίμηση Λογικών Εκφράσεων

- Για λόγους αποδοτικότητας
- Αποφυγή λαθών
  - ((y / x) > 2 && (x != 0))
  - Εάν x ισούται με 0, run time error
    - (λάθος διότι απαγορεύεται η διαίρεση με το 0).
- Όμως η ακόλουθη διατύπωση δεν έχει πρόβλημα (υπερβαίνει το πρόβλημα της διαίρεσης με 0)
  - (x != 0 && (y / x) > 2)

Σειρά Συνθηκών σε μία Λογική Έκφραση

- Γενικά σε μία σύζευξη, *E1 && E2*, η συνθήκη E1 πρέπει να αποτελεί τον αριστερό τελεστέο, εάν στην περίπτωση που αποτιμείται σε 0, τυχόν αποτίμηση της E2 θα οδηγήσει σε πρόβλημα.
- Με αυτή την σειρά αποτρέπεται η αποτίμηση της E2.
- Ανάλογο για διάζευξη E1||E2
  - ((x == 0) || (y / x) > 2)

Εκφράσεις στη C

- x και y μεγαλύτερα του z
  - (x>z && y>z)
- x είναι ίσο με το 2 ή με το 10
  - (x==2 || x==10)
- a είναι στο πεδίο από b μέχρι και c
  - (a>=b && a<=c)
- α είναι έξω από το πεδίο b μέχρι και c
  - !(a>=b && a<=c) ή (a<b || a>c)
- x είναι μικρότερο του 0 ή μεταξύ 10 και 1000
  - (x<0) || (x>=10 && x<=1000)
- z είναι αγγλικός χαρακτήρας
  - (z>='a' && z<='z') || (z>='A' && z<='Z')

Παράδειγμα

Γράψετε κώδικα που παίρνει παραμέτρους δυο ακέραιους αριθμούς και τυπώνει τον μικρότερο.

Παράδειγμα

```
int main()
{
        int a, b;
        printf("dwse to a\n");
        scanf("%d", &a);
        printf("dwse to b\n");
        scanf("%d", &b);
        if (a < b)
                printf("%d\n", a);
        else
                printf("%d\n", b);
        return 0;
}
```

Παράδειγμα

```
int main()
{
        int a, b, minimum;
        printf("dwse ta a kai to b\n");
        scanf("%d%d", &a, &b);
        if (a < b)
                minimum = a;
        else
                minimum = b;
        printf("%d\n", minimum);
        return 0;
}
```

Παράδειγμα

```
int main()
{
      int a, b, minimum;
      printf("dwse ta a kai to b\n");
      scanf("%d%d", &a, &b);
      minimum = b;
      if (a < b)
            minimum = a;
      printf("%d\n", minimum);
      return 0;
}
```

Παράδειγμα

- *Γράψετε κώδικα που τρεις ακέραιους αριθμούς και επιστρέφει τον μικρότερο.*

Παράδειγμα

```
int minimum;
if (a < b)
     minimum = a;
else
     minimum = b;
if (c < minimum)
     minimum = c;
else
{
}
printf("The minimum is %d\n", minimum);
```

Παράδειγμα

```
int minimum;
if (a < b)
      if (a < c)
            minimum = a;
      else
            minimum = c;
else
      if (b < c)
            minimum = b;
      else
            minimum = c;
printf("The minimum is %d\n", minimum);
```

Το κάθε else ταιριάζει με το προηγούμενο if (εκτός και αν χρησιμοποιήσουμε {})

Nested if-else

Γράψετε κώδικα που ελέγχει ένα ακέραιο αριθμό και τυπώνει

- 1 εαν είναι θετικός ,
- 0 εαν είναι 0 και
- -1 εαν είναι αρνητικός

Nested if-else

```
int number, code;
scanf("%d", &number);
if (number > 0)
    code = 1;
else
    if (number < 0)
        code = -1;
    else /* number einai 0 */
        code = 0;
printf("%d\n",code);
```

Nested if-else

```
int number, code;
scanf("%d", &number);
if (number > 0)
    code = 1;
else
{
    if (number < 0)
        code = -1;
    else /* number einai 0 */
        code = 0;
}
printf("%d\n",code);
```

Nested if-else

```
int number, code;
scanf("%d", &number);
if (number > 0)
    code = 1;
else if (number < 0)
    code = -1;
else /* number einai 0 */
    code = 0;

printf("%d\n",code);
```

Χρήση λογικών τελεστών

Γράψετε κώδικα που παίρνει τρεις ακέραιους αριθμούς και επιστρέφει τον μικρότερο.

```
int minimum;
if ( (a < b) && (a < c) )
     minimum = a;
else if ( (b < a) && (b < c) )
     minimum = b;
else
     minimum = c;
printf("The minimum is %d\n", minimum);
```

if για έλεγχο

Εντολές ελέγχου για να έλεγχο της ορθότητας δεδομένων π.χ.

```
if(size<=0)
{
    printf("Error: size is not
        positive\n");
    exit(-1);
    /*termatizei το programma*/
}
```

if για έλεγχο

```
printf("Enter length and width in  meters:");
scanf("%f%f",&length,&width);
if (length<0)
{
     printf("Error: Length is not
          positive!\n");
     exit(-1);
}
```

Παράδειγμα

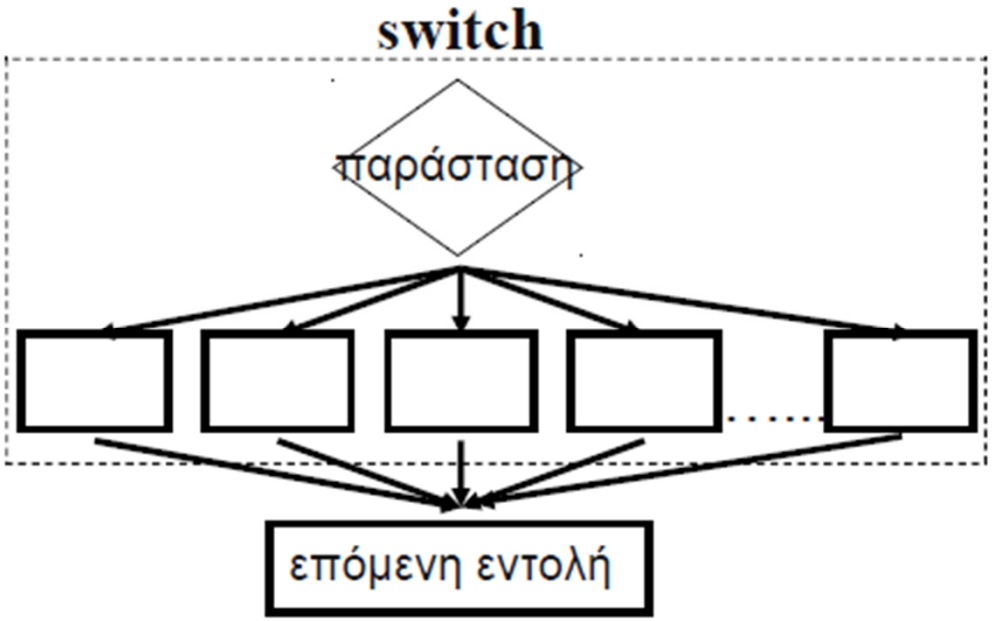
```
int main()
{
   char p;
   float x,y,res;

   scanf("%f %c %f",&x,&p,&y);
   if (p == '+')
   {
        res = x+y;
   }
   else if (p == '-')
   {
        res = x-y;
   }
   else if (p == '*')
   {
        res = x*y;
   }
   else if (p == '/');
   {
        res = x/y;
   }
   else
   {
        printf("La8os\n");
        return -1;
   }
   printf("%f%c%f = %f\n",
        x,p,y,res);
   return 0;
}
```

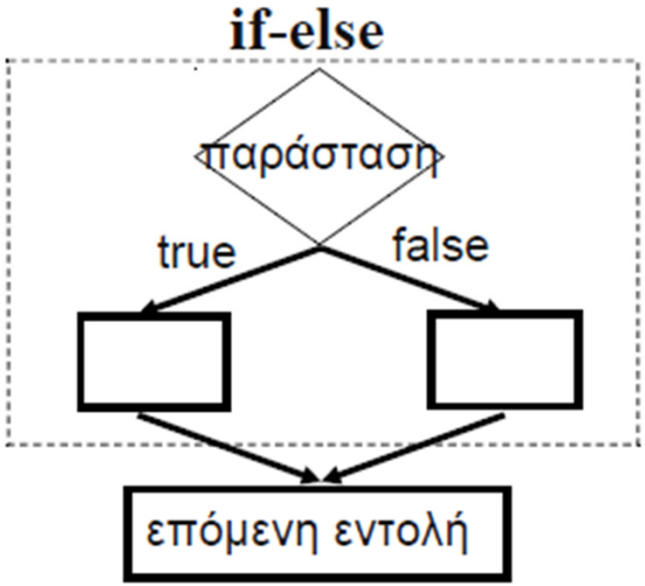
Εντολές υπό Συνθήκη - switch

```
switch (έκφραση)
{
    case σταθερή_παράσταση_1 :
                    statements;
                    break;
    case σταθερή_παράσταση_2 :
                    statements;
                    break;
    default : statements;
}
```

- Το default είναι προαιρετικό και εκτελείται αν καμία παράσταση δεν βρεθεί ίση με την έκφραση
- Αν ένα break λείπει τότε συνεχίζεται η εκτέλεση των εντολών

switch


switch
παράσταση
επόμενη εντολή


if-else
παράσταση
true
false
επόμενη εντολή

Εντολές υπό Συνθήκη - switch

- Η τιμή ελέγχου i πρέπει να είναι ακέραιου τύπου (**char** , **int** , **short int**, **long int**) με τις **signed** και **unsigned** παραλλαγές τους. Η τιμή μπορεί να προέρχεται από μεταβλητή ή να είναι η τιμή συνάρτησης που επιστρέφει τέτοιο τύπο ή κάποια μεταβλητή. Οι τιμές value1, value2,..., valueN είναι σταθερές ακέραιου τύπου και διακριτές μεταξύ τους.
- Κατά την εκτέλεση, η τιμή ελέγχου i συγκρίνεται με κάθε μία από τις value1, value2,...,valueN. Αν η τιμή της περιλαμβάνεται σε αυτές, τότε εκτελούνται οι εντολές που ακολουθούν το αντίστοιχο **case** . Η εκτέλεση συνεχίζει με τις εντολές των επόμενων **case** ή/και του **default** αν έπεται, έως ότου διακοπεί με **break** (ή τυχόν π.χ. **goto**, **return**).
- Μετά το **break** η εκτέλεση συνεχίζει με την πρώτη εντολή μετά το καταληκτικό άγκιστρο της δομής **switch**.
- Αν δεν βρεθεί η τιμή της στις value1, value2,...,valueN εκτελείται το block των εντολών του **default** , αν υπάρχει. Αλλιώς, η εκτέλεση συνεχίζει μετά το } του **switch** .
- Οι σχετικές θέσεις των **case** και του **default** μπορούν να είναι οποιεσδήποτε.

Παράδειγμα με switch

```
int main()
{
	char p;
	float x, y, res;

	scanf("%f %c %f",&x,&p,&y);
	switch (p)
	{
		case ‘+’ :
				res = x+y;
				break;
		case ‘-’ :
				res = x-y;
				break;
		case ‘*’ :
				res = x*y;
				break;
		case ‘/’ :
				res = x/y;
				break;

		default : printf(“La8os eisodos\n”);
	}

	printf(“%f%c%f = %f\n”,x,c,y,res);
}
```

Τυπικά Λάθη

- if(x=5) αντί για if(x==5)
- Ξεχάσατε το break σε εντολή switch
- Ξεχάσατε {, }
- Λάθος στις προτεραιότητες: ( )
- ***Λύση: Μορφοποίηση και χρήση σχολίων***

Παραστάσεις υπό Συνθήκη - ?

Ο τελεστής (?:) είναι ένας ιδιαίτερα διαδεδομένος ιδιωματισμός της C++. Η εντολή

```
if (condition)
  val = value1;
else
  val = value2;
```

ισοδυναμεί με val = (condition ? value1 : value2);

Γενικότερα, η έκφραση

(condition ? έκφρασηΑ : έκφρασηΒ)

έχει την τιμή "έκφρασηΑ" όταν η συνθήκη condition είναι αληθής, ενώ έχει την τιμή "έκφρασηΒ" όταν η συνθήκη είναι ψευδής.

Παραστάσεις υπό Συνθήκη - ?

Οι παρενθέσεις που περιβάλουν τον τελεστή (?:) με τα ορίσματά του στο συγκεκριμένο παράδειγμα δεν είναι απαραίτητες, βοηθούν όμως στην κατανόηση του κώδικα.

Καθώς για το συγκεκριμένο τελεστή δεν μπορεί να καθοριστεί μονοσήμαντα η προτεραιότητά του σε σχέση με τον (=) πρέπει να τις χρησιμοποιούμε για να εκτελείται η επιδιωκόμενη πράξη.

Εναλλακτικά, μπορούμε να εφαρμόζουμε τον εξής κανόνα: Οι τελεστές (?:) και (=) έχουν ίδια προτεραιότητα, δεχόμενοι ότι οι πράξεις εκτελούνται από τα δεξιά προς τα αριστερά. Επομένως, η έκφραση

- `a = b ? c : d` ισοδυναμεί με `a = (b ? c : d)`,
- ενώ η έκφραση `a ? b : c = d` εκτελείται ως `a ? b : (c = d)`

Παραστάσεις υπό Συνθήκη - ?

```
if ( x > y)
{
   x = a;
}
else
{
   x = b;
}
```

$\Leftrightarrow$

```
x = (x>y) ? a : b;
```

Το πιο σύνηθες λάθος

printf(“%d\n”,(x>y?a:b));