Σημειώσεις C

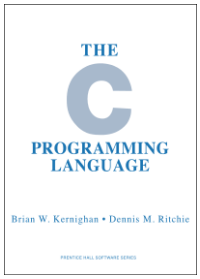
Βασισμένες στο υλικό του καθηγητή Ν. Βασιλά στο ΤΕΙ Αθηνών

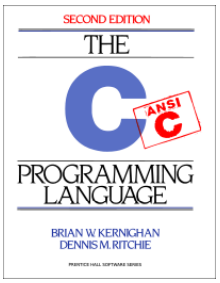
Η ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ C

- • Η C δημιουργήθηκε το 1972 στα Bell Labs από τον Dennis Ritchie καθώς, μαζί με τον Ken Thompson (είχε επινοήσει τη γλώσσα B), σχεδίαζαν το λειτουργικό σύστημα UNIX.
  - [Το **λειτουργικό σύστημα** (**operating system**) είναι ένα σύνολο προγραμμάτων για τη διαχείριση των πόρων ενός υπολογιστικού συστήματος].
- Αν και οι περισσότερες γλώσσες στοχεύουν στο να είναι χρήσιμες, συχνά επινοούνται και για άλλους λόγους. Για παράδειγμα, η Pascal σχεδιάσθηκε κυρίως για εκπαιδευτικούς σκοπούς (διδασκαλία αρχών προγραμματισμού) ενώ η BASIC σχεδιάσθηκε ως μια φιλική γλώσσα που μαθαίνεται εύκολα ακόμη και αν δεν υπάρχει εξοικείωση με τους Η/Υ.
- Αντιθέτως, η C σχεδιάσθηκε για να αποτελέσει ένα χρήσιμο εργαλείο για προγραμματιστές.

Classic book on C

- *zanasi.chem.unisa.it/download/**C**.pdf*





Η ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ C

- Η ραγδαία εξάπλωση της C και η καθιέρωσή της ως μιας από τις πιο σημαντικές γλώσσες προγραμματισμού οφείλεται στους εξής λόγους:
- Παρέχει δυνατότητες (π.χ. δομημένου προγραμματισμού) που η υπολογιστική επιστήμη θεωρεί ότι είναι επιθυμητές με αποτέλεσμα πιο αξιόπιστα και κατανοητά προγράμματα.
- Η C είναι μια αποτελεσματική γλώσσα (τα C-προγράμματα έχουν την τάση να είναι πιο συμπαγή και να εκτελούνται πιο γρήγορα).
- Η C είναι μια μεταφέρσιμη (portable) γλώσσα (ο C-κώδικας που έχει γραφεί σε κάποιο σύστημα μπορεί να χρησιμοποιηθεί αυτούσιος και σε άλλα συστήματα)
- Η C είναι ένα πανίσχυρο και ευέλικτο προγραμματιστικό εργαλείο (π.χ. το 94%, περίπου, του UNIX είναι γραμμένο σε C συμπεριλαμβανομένων και μεταγλωττιστών και διερμηνευτών για άλλες γλώσσες όπως, η FORTRAN, η Pascal, η LISP, η Logo και η BASIC).
- Η C παρέχει δυνατότητες λεπτομερούς ελέγχου της ροής του προγράμματος - κάτι που συνήθως συσχετίζεται με τη γλώσσα assembly - επιτρέποντας βελτιστοποίηση των προγραμμάτων για
- μέγιστη επίδοση.
- Η C είναι μια φιλική δομημένη γλώσσα που ενθαρρύνει τις καλές προγραμματιστικές συνήθειες.

ΣΤΑΔΙΑ ΔΗΜΙΟΥΡΓΙΑΣ ΚΑΙ ΕΚΤΕΛΕΣΗΣ ΕΝΟΣ ΠΡΟΓΡΑΜΜΑΤΟΣ ΣΕ C

- Χρησιμοποιούμε έναν **συντάκτη** (**editor**) για να γράψουμε το πρόγραμμά μας σε ένα **'.c' αρχείο**.
- Υποβάλλουμε το πρόγραμμά μας σε μεταγλώττιση χρησιμοποιώντας τον **μεταγλωττιστή** (**compiler**) της C. Ο μεταγλωττιστής αρχικά θα ελέγξει το πρόγραμμά μας για πιθανά συντακτικά και σημασιολογικά σφάλματα. Αν δεν βρει σφάλματα, θα μεταφράσει το πρόγραμμα σε συμβολική γλώσσα (assembly) και ο συμβολομεταφραστής θα δημιουργήσει ένα **παράγωγο πρόγραμμα** (**object program**) σε **'.o' αρχείο**.
- Στη συνέχεια, γίνεται η σύνδεση του παράγωγου προγράμματός μας με άλλα παράγωγα προγράμματα καθώς και με τις βιβλιοθήκες του συστήματος μέσω του **συνδέτη** (**linker**) και καταλήγουμε στη μορφή ενός **εκτελέσιμου αρχείου** (**a.out**).

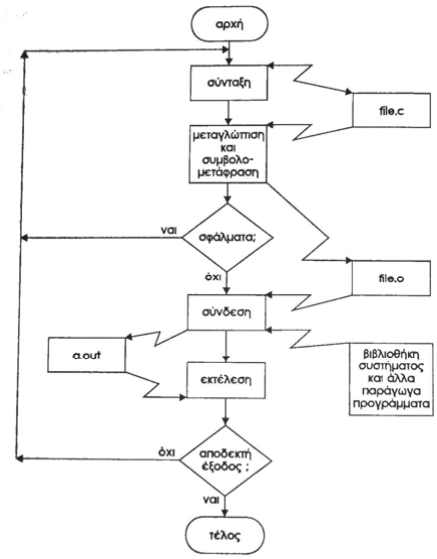
  gcc <filename>.c

  gcc –o <executable filename> <filename>.c
- Εκτελούμε / τρέχουμε (**execute** / **run**) το πρόγραμμά μας πληκτρολογώντας το όνομα του εκτελέσιμου αρχείου.

  ./a.out

  ./ <executable filename>

Διαδικασία παραγωγής εκτελέσιμου προγράμματος



ΕΝΑ ΑΠΛΟ ΠΡΟΓΡΑΜΜΑ

```
#include <stdio.h>

int main() {
        printf("Hello.\n");
        return 0;
}
```

ΣΗΜΕΙΩΣΗ: ***Όλα τα προγράμματα που θα χρησιμοποιηθούν στο μάθημα θα είναι σύμφωνα με το πρότυπο ANSI (American National Standard Institute) της C. Η ANSI C αναπτύχθηκε για να αντιμετωπιστούν προβλήματα ασυμβατότητας.***

```
#include<stdio.h>
int main()
{
long int N,r,q;
int a[100],i=1,j;
printf("Dose: ");
 scanf("%ld",&N);
q=N;
while (q!= 0){
a[i++]=q%2;
q=q/2;}
printf("Apot %d: ",N);
for(j=i-1;j> 0;j--)
rintf("%d",a[j]);
return 0;
}
```

τι κάνει το δίπλα πρόγραμμα;

Τώρα…
τι κάνει το δίπλα πρόγραμμα;

```
#include<stdio.h>

int main() {

        long int decimalNumber,remainder,quotient;
        int binaryNumber[100],i=1,j;

        printf("Enter any decimal number: ");
         scanf("%ld",&decimalNumber);

        quotient = decimalNumber;
        while ( quotient != 0) {
                binaryNumber[i++]= quotient % 2;
                quotient = quotient / 2;
        }

        printf("Equivalent binary value of decimal number %d: ",decimalNumber);
        for(j = i -1 ;j> 0;j--)
                printf("%d",binaryNumber[j]);
        return 0;
}
```

- • Το πρόγραμμα είναι γνωστό και ως **“πηγαίος κώδικας”** (**source code**) ενώ το αρχείο στο οποίο αποθηκεύεται ονομάζεται **“πηγαίο αρχείο”** (**source file**).
- Για να δημιουργήσουμε εκτελέσιμο αρχείο με χρήση του μεταγλωττιστή cc του UNIX, πληκτρολογούμε:
  - **cc test.c**
- Αν δεν υπήρχαν σφάλματα στο πρόγραμμά μας τότε το αποτέλεσμα είναι να δημιουργηθεί ένα εκτελέσιμο αρχείο
- με όνομα **a.out**.
- Πληκτρολογώντας τώρα **a.out** θα εμφανισθεί στην οθόνη του υπολογιστή μας το:
  - **Hello.**

Επεξηγήσεις [1/3]

- **#include <stdio.h>**
- Πριν από κάθε C-πρόγραμμα εμφανίζονται οδηγίες προς τον μεταγλωττιστή μέσω του **προεπεξεργαστή** της C (**Cpreprocessor**).
- Η παραπάνω οδηγία υποχρεώνει τον μεταγλωττιστή της C να διαβάσει και να μεταγλωττίσει το **αρχείο επικεφαλίδας** (**header file**) **stdio.h** που έχει σχέση με τη βιβλιοθήκη Εισόδου/Εξόδου στην οποία βρίσκεται η συνάρτηση **printf()**.
- Το αρχείο **stdio.h** (**st**andar**d i**nput **o**utput) περιέχει δηλώσεις συναρτήσεων για εισαγωγή δεδομένων από το πληκτρολόγιο και εμφάνιση δεδομένων στην οθόνη και το συμπεριλαμβάνουμε σε όλα τα προγράμματά μας.

Επεξηγήσεις [2/3]

- **int main() {**
  }
- Ένα C-πρόγραμμα αποτελείται από μια ή περισσότερες **συναρτήσεις** (**functions**). Η βασική συνάρτηση που υπάρχει σε κάθε C-πρόγραμμα είναι η **main()**.
- Μέσα στις παρενθέσεις τοποθετούνται τα **ορίσματα** (**arguments**) της συνάρτησης. Οι παρενθέσεις πρέπει πάντα να υπάρχουν ακόμη και αν δεν υπάρχουν ορίσματα, όπως σ'αυτό το παράδειγμα.
- Το **σώμα** (**body**) της συνάρτησης, δηλαδή ο κώδικας που αφορά τη συνάρτηση, περικλείεται στη C ανάμεσα από άγκιστρα **{ }**.
- Η δεσμευμένη λέξη **int** υποδηλώνει ότι η **main()** πρέπει να επιστρέψει ακέραιο αριθμό στο λειτουργικό σύστημα.

Επεξηγήσεις [3/3]

- **printf(“Hello.\n");**
- Η εντολή αυτή είναι στην ουσία μια κλήση της συνάρτησης βιβλιοθήκης **printf()** η οποία, όπως άλλωστε όλες οι δηλώσεις ή εντολές της C, καταλήγει στο σύμβολο **';'**.
- Το όρισμα της συνάρτησης αυτής είναι η **συμβολοσειρά** (δηλαδή, η ακολουθία χαρακτήρων) **“Hello.\n"**. Οι συμβολοσειρές στη C περικλείονται από διπλά εισαγωγικά.
- Ο ειδικός χαρακτήρας **'\n'** ανήκει στις λεγόμενες **ακολουθίες διαφυγής** (**escape sequences**) και ονομάζεται **χαρακτήρας νέας γραμμής** (**newline character**).
- Το αποτέλεσμα αυτής της εντολής είναι να εμφανισθεί στην οθόνη το:

  Hello. με τον **δρομέα** (**cursor**) να τοποθετείται στην αρχή της επόμενης γραμμής.

ΑΚΟΛΟΥΘΙΕΣ ΔΙΑΦΥΓΗΣ

`\n`	τοποθέτηση δρομέα στην αρχή νέας γραμμής (newline)
`\t`	χαρακτήρας στηλοθέτη (tab character)
`\b`	χαρακτήρας οπισθοδρόμησης (backspace character)
`\r`	επαναφορά δρομέα στην αρχή της τρέχουσας γραμμής (carriage return)
`\f`	τοποθέτηση δρομέα στην αρχή νέας σελίδας (form feed)
`\\`	ανάποδη κάθετος ή πισωκάθετος ' \ ' (backslash)
`\'`	απλό εισαγωγικό ' ' ' (single quote)
`\"`	διπλό εισαγωγικό ' " ' (double quote)

ΕΝΑ ΠΙΟ ΣΥΝΘΕΤΟ ΠΡΟΓΡΑΜΜΑ 1/2

```
#include <stdio.h>
int main()  /* a simple program */
{
  int num;

  num = 1;
  myprint();
  printf("O arithmos poy brisketai sth ");
  printf("num einai o %d.\n",num);
  return 0;
}

void myprint()
{
  printf("Paradeigma emfanishs arithmou.\n\n");
}
```

Σχόλια

Δήλωση ακέραιας μεταβλητής

Εκχώρηση της τιμής 1 *στην* num

Κλήση δικής μας συνάρτησης

Ορισμός δικής μας συνάρτησης, στο ίδιο αρχείο, μετά την main()

Έξοδος:
```
Paradeigma emfanishs arithmou.
O arithmos poy brisketai sth num einai o 1.
```

ΕΝΑ ΠΙΟ ΣΥΝΘΕΤΟ ΠΡΟΓΡΑΜΜΑ 2/2

```
#include <stdio.h>
int main()  /* a simple program */
{
  int s;

  s = 52;
  printf("There are %d weeks in a year.\n",s);
  return 0;
}
```

Έξοδος προγράμματος:

```
There are 52 weeks in a year.
```

Δεδομένα: μεταβλητές και σταθερές

- Μια **μεταβλητή** είναι μια **θέση μνήμης** με συγκεκριμένο όνομα. Το περιεχόμενο της θέσης μνήμης είναι η **τιμή** της μεταβλητής.
- Σε μια μεταβλητή μπορούμε να εκχωρήσουμε αρχικά μια τιμή ενώ η τιμή αυτή μπορεί να αλλάξει (να μεταβληθεί) στη συνέχεια κατά την εκτέλεση του προγράμματος.
- Η προσπέλαση της τιμής μιας μεταβλητής γίνεται με άμεσο τρόπο καθώς γνωρίζουμε εκ των προτέρων τη διεύθυνση της αντίστοιχης θέσης μνήμης και ονομάζεται **άμεση προσπέλαση μνήμης**.
- Σε αντίθεση, οι **σταθερές** προκαθορίζονται και δεν αλλάζουν τιμή κατά την εκτέλεση του προγράμματος.

Ονόματα μεταβλητών

- Το όνομα μιας μεταβλητής περιλαμβάνει πεζά [a, b, …, z] ή κεφαλαία [A, B, …, Z] γράμματα του αγγλικού αλφαβήτου, ψηφία [0, 1, …, 9] ή τον χαρακτήρα υπογράμμισης '_'.
- Ο πρώτος χαρακτήρας πρέπει να είναι είτε γράμμα ή ο χαρακτήρας υπογράμμισης (δεν επιτρέπεται να ξεκινάει από ψηφίο).
- Στην C, οι πεζοί χαρακτήρες εκλαμβάνονται ως διαφορετικοί από τους κεφαλαίους κατά την ονοματοθεσία των μεταβλητών (π.χ. οι μεταβλητές **X1** και **x1** ή οι **area** και **Area** είναι διαφορετικές).
- Δεν επιτρέπεται μια μεταβλητή να έχει το ίδιο όνομα με κάποια **δεσμευμένη** λέξη της C.
- Είναι καλό να δίνονται μνημονικά ονόματα στις μεταβλητές μας.

Οι δεσμευμένες λέξεις της C

asm	auto	break	case	char
const	continue	default	do	double
else	enum	extern	float	for
goto	if	int	long	register
return	short	signed	sizeof	static
struct	switch	typedef	union	unsigned
void	volatile	while		

Τύποι Δεδομένων 1/2

```
#include <stdio.h>
int main()  /* data types */
{
  int i,j,hours,months;  /* integer */
  short int years = 12;  /* short integer */
  short s1;              /* short integer */
  long int seconds;      /* long integer */
  long l1,l2;            /* long integer */
  unsigned int u,v;      /* unsigned integer */
  unsigned u1,v1;        /* unsigned integer */
  unsigned short us1;    /* unsigned short */
  unsigned long ul1;     /* unsigned long */
  char ch = 'A';         /* character */
  float pi = 3.14;       /* floating point */
  double x,y,z;          /* double precision */

  printf("int: %d bytes\n",sizeof (int));
  printf("short: %d bytes\n",sizeof (short));
  printf("long: %d bytes\n",sizeof (long));
  printf("float: %d bytes\n",sizeof (float));
  return 0;
}
```

Τύποι Δεδομένων 2/2

Η C χρησιμοποιεί τους εξής βασικούς τύπους δεδομένων: **char**, **int**, **float**, **double**. Με χρήση όμως των **προσδιοριστών (qualifiers)** **short**, **long**, **signed**, **unsigned** οι τύποι της C είναι όπως στον παρακάτω πίνακα.

Τύπος Δεδομένων	Σύνηθες όνομα τύπου	Μέγεθος σε bytes	Εύρος τιμών
short int	short	2	- 32.768 έως 32.767
unsigned short int	unsigned short	2	0 έως 65.535
unsigned int	unsigned int	4	0 έως 4.294.967.295
int	int	4	-2.147.483.648 έως 2.147.483.647
long int	long	4	-2.147.483.648 έως 2.147.483.647
unsigned long int	unsigned long	4	0 έως 4.294.967.295
signed char	char	1	-128 έως 127
unsigned char	unsigned char	1	0 έως 255
float	float	4	Θετικοί: 1.17e-38 έως 3.4e38 Αρνητικοί: -3.4e38 έως -1.17e-38
double	double	8	περίπου: $-1.8*10^{308}$ έως $1.8*10^{308}$
long double	long double	12	εκτεταμένη ακρίβεια ...

Εκχώρηση τιμών σε μεταβλητές

- Η **εκχώρηση τιμής** σε μια μεταβλητή αρχικοποιεί την μεταβλητή. Μπορεί να γίνει είτε κατά τη δήλωση της μεταβλητής είτε και αργότερα. Παράδειγμα:

```
. . .
int a, b, c = 5;
float r = 2.0;

a = 0;
b = 1;
. . .
```

Υπερχείλιση (overflow)

```
/* overflow example */
#include <stdio.h>
int main() {
        short int i = 32767;
        unsigned char c = 255;
        printf("i+1 = %d, c+2 = %d\n", i+1, c+2);
        return 0;
}
```

Έξοδος προγράμματος:

i+1 = -32768, c+2 = 1

Η printf()

- Χρησιμοποιείται για την εμφάνιση αριθμητικών δεδομένων και χαρακτήρων στην οθόνη.
- Το πρώτο όρισμα της **printf()** είναι μια ακολουθία χαρακτήρων (συμβολοσειρά) η οποία καθορίζει τον τρόπο εμφάνισης των δεδομένων στην οθόνη.
- Τα επόμενα ορίσματα (αν υπάρχουν) είναι τα δεδομένα (λίστα μεταβλητών) που θέλουμε να εμφανίσουμε.
- Η θέση εμφάνισης των δεδομένων καθορίζεται από τα λεγόμενα **προσδιοριστικά μετατροπής** (**modifiers**) δηλαδή, τον χαρακτήρα '**%**' ακολουθούμενο από κάποιο συγκεκριμένο χαρακτήρα που προσδιορίζει τον τύπο των δεδομένων προς εμφάνιση.

Προσδιοριστικά μετατροπής της printf()

%d ή %i	εμφάνιση ακεραίου
%f	εμφάνιση πραγματικού αριθμού
%u	εμφάνιση θετικού (μη προσημασμένου) ακεραίου
%c	εμφάνιση χαρακτήρα
%s	εμφάνιση συμβολοσειράς
%e ή %E	εμφάνιση πραγματικού σε εκθετική μορφή
%g	χρήση της πιο σύντομης μορφής από %f ή %e
%p	εμφάνιση διεύθυνσης μνήμης μιας μεταβλητής
%x ή %X	εμφάνιση ακεραίου σε δεκαεξαδική μορφή
%o	εμφάνιση ακεραίου σε οκταδική μορφή

Η scanf()

- Χρησιμοποιείται για την είσοδο δεδομένων από το πληκτρολόγιο.
- Η **scanf()** χρησιμοποιεί μια ακολουθία ειδικών χαρακτήρων μετατροπής (%d, %f, κ.λπ.) για το διάβασμα των δεδομένων και στη συνέχεια εκχωρεί τα δεδομένα σε μια ή περισσότερες μεταβλητές.
- Για την εκχώρηση των δεδομένων στις μεταβλητές, η **scanf()** χρειάζεται (στα επόμενα ορίσματα) τις διευθύνσεις των μεταβλητών στη μνήμη του συστήματος.
- π.χ. **scanf("%f %d",&r, &a);** *Διάβασμα ενός πραγματικού και ενός ακεραίου από το πληκτρολόγιο και αποθήκευσή τους στη διεύθυνση της μεταβλητής r και στη διεύθυνση της μεταβλητής a αντίστοιχα.*

Παράδειγμα scanf()

```
#include <stdio.h>
#define PI 3.14159        /* Ορισμός σταθεράς */

int main()
{
  float r,perim,area;
  char  name[40];         /* character string*/

  printf("Geia sou! Poio einai to onoma sou?\n");
  scanf("%s",name);       /* Εισαγωγή ονόματος */

  printf("%s, dosmou mia aktina kuklou \n",name);
  scanf("%f",&r); /* Εισαγωγή πραγματικού αριθμού */

  perim = 2.0*3.14*r;
  area = PI*r*r;

  printf("Perimetros kuklou: %f\n",perim);
  printf("Epifaneia kuklou: %f\n",area);

  return 0;
}
```

Αριθμητικοί Τελεστές,

- **Τελεστής εκχώρησης τιμής:** =
  c = 5; x = y = z = 0.0;
- **Αριθμητικοί τελεστές:** +, -, *, /, %, τελεστής προσήμου –
  c = a + 3; w = w – 2*x; y = -w; f = 12.0/3.0;
  i = 5/4; /* ακεραία διαίρεση 5/4 --> 1 */
  rem = 5%4 /* υπόλοιπο ακεραίας διαίρεσης */
- **Οι τελεστές μοναδιαίας αύξησης και μείωσης:** ++, --
  i++; j--; /* i = i+1; j = j-1; */
  j = i++; /* j = i; i = i+1; */
  j = ++i; /* i = i+1; j = i; */
- **Συνδυαστικοί τελεστές εκχώρησης:** +=, -=, *=, /=, %=
  x += a; /* x = x + a */
  y -= z + 1; /* y = y – (z + 1); */
  a %= 3; /* a = a % 3; */

Αριθμητικές Εκφράσεις

- Μια έκφραση αποτελείται από συνδυασμό τελεστών και τελεστέων

  -6
  a * (b + c / d) / 20
  q = 5 * 2
  x = ++q % 3
  6 + (c = 3 + 8)
  x = 30 – (2 * (10 – (7 % 4)))
  y = 1 + 3 / 5 – 2 * 10 – 7 % 4

- Για την αποτίμηση μιας αριθμητικής έκφρασης χρειάζεται να γνωρίζουμε τις προτεραιότητες των τελεστών.

Προτεραιότητες τελεστών

(κατά φθίνουσα προτεραιότητα)

Τελεστές	Προσεταιριστικότητα
( )	από αριστερά προς τα δεξιά
– (μοναδιαίο πλην)	από αριστερά προς τα δεξιά
++ --	από δεξιά προς τα αριστερά
* /	από αριστερά προς τα δεξιά
%	από αριστερά προς τα δεξιά
= += -= %= *= /=	από δεξιά προς τα αριστερά

Σχεσιακοί Τελεστές

- Μορφή έκφρασης: **τελεστέος1 τελεστής τελεστέος2**
- Οι σχεσιακοί τελεστές χρησιμοποιούνται για συγκρίσεις και οι εκφράσεις αποτιμούνται σε True (αληθές ή ισοδύναμα 1) ή False (ψευδές ή ισοδύναμα 0)
- **Ισότητα**: == (είναι οι δύο τελεσταίοι ίσοι?)
- **Μεγαλύτερο ή μικρότερο από**: **>, <**
- **Μεγαλύτερο ή ίσο**: **>=**
- **Μικρότερο ή ίσο**: **<=**
- **Ανισότητα**: **!=**
- Παραδείγματα
  - **1 == 2 (false)**
  - **1 != 2 (true)**
  - **x+1 > x–5 (true)**

Λογικοί Τελεστές

- **&&** **(and)**
- **||** **(or)**
- **!** **(not)**

Λογικές Εκφράσεις

- **έκφραση1 && έκφραση2** (*αληθής (true) αν και μόνο αν και οι δύο εκφράσεις είναι αληθείς*)
- **έκφραση1 || έκφραση2** (*αληθής αν τουλάχιστον μία από τις δύο εκφράσεις είναι αληθής*)
- **!έκφραση** (*αληθής αν η έκφραση είναι ψευδής (false) και αντίστροφα*)
- Σειρά αποτίμησης: από αριστερά προς τα δεξιά. Η αποτίμηση σταματάει όταν εξασφαλισθεί η ΑΛΗΘΕΙΑ ή το ΨΕΥΔΟΣ της έκφρασης.
- Παραδείγματα:

  6 > 4 && 3 == 3 *αληθής*

  (6 > 3) || (x == 2) *αληθής ή ψευδής ανάλογα με την τιμή του x*

  !(6 > 3) || (x = 2) *αληθής (γίνεται εκχώρηση τιμής)*

  (6 > 3) || (x = 2) *αληθής (δεν γίνεται εκχώρηση τιμής)*

Τελεστές

Αριθμητικοί

Αριθμητικοί Τελεστές,

- Τελεστής εκχώρησης τιμής: =
  c = 5; x = y = z = 0.0;
- Αριθμητικοί τελεστές: +, -, *, /, %, τελεστής προσήμου –
  c = a + 3; w = w – 2*x; y = -w; f = 12.0/3.0;
  i = 5/4; /* ακέραιος διαίρεση 5/4 --> 1 */
  rem – 5%4 /* υπόλοιπο ακεραίου διαίρεσης */
- Οι τελεστές μοναδαίας αύξησης και μείωσης: ++, --
  i++; j--; /* i = i+1; j = j-1; */
  j = i++; /* i = i; i = i+1; */
  j = ++i; /* i = i+1; j = i; */
- Συνδυαστικοί τελεστές εκχώρησης: +=, -=, *=, /=, %=
  x += a; /* x = x + a */
  y -= z + 1; /* y = y – (z + 1); */
  a %= 3; /* a = a % 3; */

Λογικοί

Λογικοί Τελεστές

- && (and)
- || (or)
- ! (not)

Σχεσιακοί

Σχεσιακοί Τελεστές

- Μορφή έκφρασης: τελεστέος1 τελεστής τελεστέος2
- Οι σχεσιακοί τελεστές χρησιμοποιούνται για συγκρίσεις και οι εκφράσεις αποτιμώνται σε True (αληθές ή ισοδύναμα 1) ή False (ψευδές ή ισοδύναμα 0)
- **Ισότητα:** == (είναι οι δύο τελεσταίοι ίσοι?)
- **Μεγαλύτερο ή μικρότερο από:** >, <
- **Μεγαλύτερο ή ίσο:** >=
- **Μικρότερο ή ίσο:** <=
- **Ανισότητα:** !=
- Παραδείγματα
  1 == 2 (false)
  1 != 2 (true)
  x+1 > x–5(true)