

Προγραμματισμός Ι

Τύποι Δεδομένων - Δήλωση Μεταβλητών - Έξοδος Προγράμματος

Πανεπιστήμιο Πελοποννήσου
Τμήμα Πληροφορικής & Τηλεπικοινωνιών

Νικόλαος Δ. Τσελίκας

Μνήμη και Μεταβλητές

◆ Σχέση Μνήμης Υπολογιστή και Μεταβλητών

- Η μνήμη (RAM) ενός υπολογιστή αποτελείται από πολλά εκατομμύρια θέσεις αποθήκευσης δεδομένων που έχουν διαδοχική αρίθμηση
- Το μέγεθος κάθε θέσης μνήμης είναι μία οκτάδα (byte)
- Π.χ. σκεφτείτε ότι ένας παλιός υπολογιστής με μόνο 16MB μνήμης έχει μνήμη:
$$16 * 1.024 = 16.384 \text{ kbytes}$$
$$16.384 * 1.024 = 16.777.216 \text{ θέσεις μνήμης (bytes)}$$
- Κάθε θέση μνήμης μπορεί να έχει ένα όνομα και ένα περιεχόμενο
- **Μεταβλητή** ονομάζεται **μία θέση μνήμης** που έχει ένα **συγκεκριμένο όνομα**
- Η **τιμή μίας μεταβλητής** είναι **το περιεχόμενο** αυτής της **θέσης μνήμης** (ή των **θέσεων μνήμης**, όπως θα δούμε) και μπορεί να αλλάξει κατά τη διάρκεια εκτέλεσης του προγράμματος

Ονόματα Μεταβλητών

- ◆ **Απαράβατοι κανόνες κατά τη δήλωση του ονόματος μίας μεταβλητής**
 - Μπορεί να αποτελείται από **πεζά και κεφαλαία** γράμματα του λατινικού αλφαβήτου και **ψηφία**
 - Αποδεκτός είναι επίσης και ο **χαρακτήρας υπογράμμισης** `'_'` (underscore)
 - Ο **πρώτος χαρακτήρας** πρέπει να είναι γράμμα ή ο χαρακτήρας υπογράμμισης `'_'`
 - Η γλώσσα C είναι **case sensitive** (δηλ. κάνει διάκριση μεταξύ των πεζών και κεφαλαίων γραμμάτων)
 - ◆ Συνεπώς, η μεταβλητή με το όνομα `nick` είναι διαφορετική από τη μεταβλητή με το όνομα `Nick`
 - Οι **δεσμευμένες λέξεις** της C απαγορεύεται να χρησιμοποιηθούν ως ονόματα μεταβλητών

Δεσμευμένες Λέξεις της C

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

Παρατηρήσεις

- Το όνομα που επιλέγετε να δώσετε σε μία μεταβλητή είναι χρήσιμο να περιγράφει όσο το δυνατόν καλύτερα τον σκοπό της μεταβλητής μέσα στο πρόγραμμα, ώστε ο κώδικας να είναι πιο ευανάγνωστος
 - ◆ Π.χ. το όνομα μίας μεταβλητής που υπολογίζει το άθροισμα κάποιων αριθμών είναι προτιμότερο να είναι `sum` αντί για `var`
- Αν το όνομα που επιλέξατε για μία μεταβλητή αποτελείται από δύο ή και περισσότερες λέξεις, τότε προτείνεται να τις διαχωρίζετε μεταξύ τους με τον χαρακτήρα υπογράμμισης `'_'`, έτσι ώστε να διευκολύνεται η ερμηνεία τους
 - ◆ Π.χ. το όνομα μίας μεταβλητής που υπολογίζει τον αριθμό των βιβλίων σε μία βιβλιοθήκη είναι προτιμότερο να είναι `books_number` αντί για `booksnumber`
- Προτείνεται, τα ονόματα μεταβλητών να αποτελούνται μόνο από πεζά γράμματα

Δήλωση Μεταβλητών

- Για να χρησιμοποιήσετε μία μεταβλητή μέσα σε ένα πρόγραμμα πρέπει πρώτα να τη δηλώσετε
- Η δήλωση μίας μεταβλητής γίνεται με τον ακόλουθο τρόπο:

```
τύπος_δεδομένων όνομα_μεταβλητής;
```

- Το όνομα_μεταβλητής είναι το τυχαίο όνομα που επιλέγει ο προγραμματιστής σύμφωνα με τους κανόνες και τις παρατηρήσεις που είπαμε προηγουμένως
- Ο τύπος_δεδομένων είναι ένας από τους αριθμητικούς τύπους δεδομένων που υποστηρίζει η γλώσσα C
 - ♦ Π.χ. η δεσμευμένη λέξη `int` χρησιμοποιείται για τη δήλωση ακέραιων μεταβλητών, δηλαδή μεταβλητών που μπορούν να έχουν μόνο ακέραιες τιμές ενώ η δεσμευμένη λέξη `float` χρησιμοποιείται για τη δήλωση πραγματικών μεταβλητών, δηλαδή μεταβλητών που μπορούν να έχουν τιμές με κλασματικό μέρος

Τύποι Μεταβλητών

Τύπος	Συνηθισμένο μέγεθος (bytes)	Εύρος τιμών (min-max)	Ψηφία ακρίβειας
<code>char</code>	1	-128 ... 127	
<code>short int</code>	2	-32.768 ... 32.767	
<code>int</code>	4	-2.147.483.648...2.147.483.647	
<code>long int</code>	4	-2.147.483.648...2.147.483.647	
<code>float</code>	4	Μικρότερη θετική τιμή: $1.17 \cdot 10^{-38}$ Μεγαλύτερη θετική τιμή: $3.4 \cdot 10^{38}$	6
<code>double</code>	8	Μικρότερη θετική τιμή: $2.2 \cdot 10^{-308}$ Μεγαλύτερη θετική τιμή: $1.8 \cdot 10^{308}$	15
<code>long double</code>	8, 10, 12, 16		
<code>unsigned char</code>	1	0 ... 255	
<code>unsigned short int</code>	2	0 ... 65535	
<code>unsigned int</code>	4	0 ... 4.294.967.295	
<code>unsigned long int</code>	4	0 ... 4.294.967.295	

■ Π.χ.

```
int a; /* Δήλωση ακέραιας μεταβλητής με όνομα a. */
```

```
float b; /* Δήλωση πραγματικής μεταβλητής με όνομα b. */
```

Προγραμματισμός Ι

Παρατηρήσεις (1/3)

- Πολλές μεταβλητές του ίδιου τύπου μπορούν να δηλωθούν στην ίδια γραμμή, αρκεί να διαχωρίζονται μεταξύ τους με κόμμα (,)
 - ♦ Δηλαδή, αντί να δηλώσετε τις μεταβλητές `a`, `b` και `c` σε τρεις ξεχωριστές γραμμές:

```
int a;
```

```
int b;
```

```
int c;
```

μπορείτε να τις δηλώσετε σε μία γραμμή ως εξής:

```
int a, b, c;
```

- Η δεσμευμένη λέξη `int` μπορεί να παραληφθεί στους ακέραιους τύπους
 - ♦ Π.χ., `short` αντί για `short int`
- Οι δεσμευμένες λέξεις μπορούν να βρίσκονται σε οποιαδήποτε σειρά κατά τη δήλωση μιας μεταβλητής
 - ♦ Π.χ., `unsigned long int a;`
είναι το ίδιο με:
`int long unsigned a;`

Παρατηρήσεις (2/3)

- Όταν δηλώνεται μία μεταβλητή, ο μεταγλωττιστής δεσμεύει τόσα bytes όσα χρειάζονται για να είναι σε θέση να αποθηκεύσει την τιμή της
- Το μέγεθος της μνήμης που δεσμεύει ένας τύπος δεδομένων μπορεί να διαφέρει από υπολογιστή σε υπολογιστή
 - ♦ Δηλαδή, ο τύπος `int` μπορεί να δεσμεύει 2 bytes σε κάποιον υπολογιστή και όχι 4 bytes
 - ♦ Για να μάθετε πόσες οκτάδες δεσμεύει ένας τύπος δεδομένων στον υπολογιστή που εργάζεστε πρέπει να χρησιμοποιήσετε τον τελεστή `sizeof` → Θα τον δούμε παρακάτω
- Με τη δήλωση μιας μεταβλητής, ο μεταγλωττιστής γνωρίζει το όνομά της και τη διεύθυνση μνήμης στην οποία βρίσκεται η μεταβλητή αυτή, έτσι, όταν η μεταβλητή χρησιμοποιείται στο πρόγραμμα ο μεταγλωττιστής χρησιμοποιεί το όνομά της και γνωρίζοντας την αντίστοιχη διεύθυνση μνήμης έχει πρόσβαση στο περιεχόμενο της μεταβλητής

Παρατηρήσεις (3/3)

- Να χρησιμοποιείτε τον τύπο `float` μόνο όταν η ακρίβεια των δεκαδικών ψηφίων δεν είναι τόσο σημαντική στο πρόγραμμά σας
- Σε περίπτωση που χρειάζεστε υψηλή ακρίβεια των δεκαδικών ψηφίων, να χρησιμοποιείτε τον τύπο `double`
- Π.χ., ποια πιστεύετε θα είναι η έξοδος του παρακάτω προγράμματος???

```
#include <stdio.h>
int main(void)
{
    float a = 3.1;

    if(a == 3.1)
        printf("Yes\n");
    else
        printf("No\n");
    return 0;
}
```

Μεγάλη ήττα...!!!
Περιμέναμε η έξοδος του προγράμματος να είναι Yes, αλλά εμφανίζει... No !!!!
Αν δηλώναμε τη μεταβλητή `a` ως `double` θα εμφάνιζε σίγουρα Yes

Εκχώρηση τιμών σε Μεταβλητές (I)

- Η εκχώρηση μίας τιμής σε μία μεταβλητή γίνεται είτε μαζί με τη δήλωση της μεταβλητής είτε αργότερα
- Π.χ. με την πρώτη εντολή δηλώνεται μία ακέραια μεταβλητή (`int`) με όνομα `a` και μετά της εκχωρείται η τιμή `100`

```
int a;  
a = 100;
```

- Εναλλακτικά, θα μπορούσαμε να γράψουμε την εκχώρηση τιμής μαζί με τη δήλωση:

```
int a = 100;
```

Εκχώρηση τιμών σε Μεταβλητές (II)

- Επίσης, επιτρέπεται η απόδοση αρχικών τιμών σε περισσότερες από μία μεταβλητές ίδιου τύπου μαζί με τη δήλωσή τους, π.χ.

```
int a = 100, b = 200, c = 300;
```

- Για την εκχώρηση μίας πραγματικής τιμής σε μία μεταβλητή τύπου `float` χρησιμοποιείται η τελεία (.) για το δεκαδικό μέρος και όχι το κόμμα (,) π.χ.

```
float a = 1.24;
```

- Αν μπροστά από μία ακέραια τιμή υπάρχει το ψηφίο 0, τότε αυτή η τιμή ερμηνεύεται σαν οκταδικός αριθμός
 - ◆ Π.χ. με την παρακάτω εντολή η τιμή που εκχωρείται στη μεταβλητή `a` δεν είναι 100, αλλά 64

```
int a = 0100;
```

- Παρομοίως, αν μπροστά από μία ακέραια τιμή υπάρχει το `0x` ή το `0X`, τότε αυτή η τιμή ερμηνεύεται σαν δεκαεξαδικός αριθμός
 - ◆ Π.χ. με την παρακάτω εντολή η τιμή της μεταβλητής `a` γίνεται 16.

```
int a = 0x10;
```

Παρατηρήσεις (I)

- Η τιμή μίας μεταβλητής μπορεί (προφανώς) να αλλάζει μέσα στο πρόγραμμα
- Όταν γίνεται χρήση μίας μεταβλητής στο πρόγραμμα χρησιμοποιείται πάντα η τελευταία τιμή της και όχι κάποια από τις προηγούμενες τιμές της

```
#include <stdio.h>
int main(void)
{
    int a;
    a = 1;
    a = 2;
    a = 3;
    printf("Val = %d\n",a);
    return 0;
}
```

- Ποια τιμή εκτυπώνεται στην οθόνη???

Παρατηρήσεις (II)

- Η τιμή που εκχωρείται σε μία μεταβλητή πρέπει να συμβαδίζει με τον τύπο της μεταβλητής

- ♦ Π.χ. με την εντολή:

```
int a = 10.9;
```

η τιμή της μεταβλητής `a` γίνεται `10`, γιατί η μεταβλητή `a` δηλώνεται σαν ακέραια μεταβλητή και όχι σαν πραγματική και το δεκαδικό μέρος αποκόπτεται (Προσοχή!! **Δεν στρογγυλοποιείται**)

- Η τιμή που εκχωρείται σε μία μεταβλητή πρέπει να είναι μέσα στο επιτρεπτό εύρος τιμών

- ♦ Π.χ. με την εντολή:

```
char ch = 130;
```

η τιμή της μεταβλητής `ch` δεν γίνεται `130`, γιατί το εύρος τιμών μίας μεταβλητής τύπου `char` είναι από `-128` έως `127`. Άρα, η τιμή `130` είναι μία τιμή εκτός των επιτρεπτών ορίων

Παρατηρήσεις (III)

- Η τιμή μίας πραγματικής μεταβλητής μπορεί να είναι και ακέραια
 - ♦ Π.χ. επιτρέπεται να γράψουμε:

```
float a = 50;
```

γιατί είναι ισοδύναμο με:

```
float a = 50.0;
```

- Η τιμή μίας πραγματικής μεταβλητής μπορεί να γραφεί και με επιστημονική σημειογραφία (συνήθως χρησιμοποιείται όταν η τιμή είναι πολύ μικρή ή πολύ μεγάλη)

- ♦ Π.χ. αντί για

```
a = 0.085;
```

μπορούμε να γράψουμε

```
a = 85E-3;
```

- Το γράμμα E ή e αναπαριστά το 10, ενώ ο αριθμός που το ακολουθεί είναι η θετική ή αρνητική δύναμη του 10.

- Δηλαδή, η έκφραση 85E-3 αντιστοιχεί στον αριθμό $85 \cdot 10^{-3}$

Σταθερές (I)

- Σταθερά ονομάζεται μία μεταβλητή που η τιμή της δεν μπορεί να αλλάξει μέσα στο πρόγραμμα
- Για να δηλωθεί μία μεταβλητή σαν σταθερά, πρέπει να προηγηθεί η λέξη `const` πριν (ή μετά) από τον τύπο της μεταβλητής
- Επίσης, μαζί με τη δήλωση της σταθεράς, πρέπει να της εκχωρηθεί και μία αρχική τιμή, η οποία δεν θα μπορεί να αλλάξει μέσα στο πρόγραμμα

- ◆ Π.χ. με την επόμενη εντολή η ακέραια μεταβλητή `a` δηλώνεται σαν σταθερά και της εκχωρείται (μόνιμα) η τιμή 10

```
const int a = 10;
```

- ◆ Αν σε κάποιο σημείο του προγράμματος επιχειρήσουμε να της αλλάξουμε τιμή, π.χ. να γράψουμε:

```
a = 100;
```

τότε ο μεταγλωττιστής θα εμφανίσει μήνυμα λάθους για μη επιτρεπτή ενέργεια

Σταθερές (II)

- Εναλλακτικός τρόπος για τη δήλωση μίας σταθεράς είναι η χρήση της οδηγίας `#define`, η οποία χρησιμοποιείται για τη δήλωση μακροεντολών
- Συνήθως, μία μακροεντολή αντιστοιχίζει ένα συμβολικό όνομα με κάποια αριθμητική τιμή
- Για τη δήλωση μακροεντολών, η οδηγία `#define` χρησιμοποιείται ως εξής:

```
#define όνομα_μακροεντολής τιμή
```


- Π.χ. η εντολή:

```
#define NUM 100
```

δηλώνει τη μακροεντολή με όνομα `NUM` και τιμή `100`

- Η `NUM` μπορεί να χρησιμοποιηθεί οπουδήποτε μέσα στο πρόγραμμα
- Ο μεταγλωττιστής όταν συναντάει τη `NUM` μέσα στο πρόγραμμα την αντικαθιστά με την τιμή `100`

Παρατηρήσεις

- Οι δηλώσεις των μακροεντολών με την οδηγία `#define` είναι προτιμότερο να γίνονται πριν από τη συνάρτηση `main()`
 - Τα ονόματα των μακροεντολών με την οδηγία `#define` συνηθίζεται να δηλώνονται με κεφαλαία γράμματα
-  Στο τέλος της οδηγίας `#define` δεν μπαίνει ελληνικό ερωτηματικό (;)
- Π.χ.

```
#include <stdio.h>

#define NUM 100

int main(void)
{
    int a, b, c;

    a = 20 - NUM;
    b = 20 + NUM;
    c = 3 * NUM;
    return 0;
}
```

vs.

```
#include <stdio.h>

int main(void)
{
    int a, b, c;

    a = 20 - 100;
    b = 20 + 100;
    c = 3 * 100;
    return 0;
}
```



const vs. #define

- Μπορούμε να δημιουργήσουμε ονόματα σταθερών, χρησιμοποιώντας είτε τη λέξη-κλειδί `const` είτε με τον ορισμό μακροεντολής (`#define`), αλλά υπάρχουν αρκετές σημαντικές διαφορές μεταξύ των δύο τρόπων δημιουργίας σταθερών
- Με την `#define` μπορούμε να δημιουργήσουμε ένα όνομα αριθμητικής σταθεράς, σταθεράς χαρακτήρα ή αλφαριθμητικού
- Με χρήση της λέξης-κλειδί `const` δημιουργούμε σταθερές οποιουδήποτε τύπου, π.χ. δεικτών, πινάκων, δομών, ενώσεων κτλ.
- Οι σταθερές με χρήση της λέξης-κλειδί `const` υπόκεινται τους ίδιους κανόνες εμβέλειας, όπως και οι μεταβλητές (θα δούμε περισσότερα στο Κεφ. 11), ενώ οι σταθερές που δηλώνονται ως μακροεντολές με την `#define` δεν ακολουθούν τους ίδιους κανόνες
- Οι τιμές των σταθερών που έχουν δηλωθεί με τη λέξη-κλειδί `const` φαίνονται στον debugger, σε αντίθεση με αυτές που έχουν δηλωθεί με την `#define`

Η συνάρτηση printf()

- Η συνάρτηση `printf()` χρησιμοποιείται για την εμφάνιση δεδομένων στο αρχείο εξόδου `stdout` (*standard output stream*), το οποίο εξ' ορισμού συνδέεται με την οθόνη
- Η συνάρτηση `printf()` δέχεται μία μεταβλητή λίστα παραμέτρων
 - ♦ Η πρώτη παράμετρος είναι ένα **αλφαριθμητικό μορφοποίησης** (*format string*), δηλαδή **μία ακολουθία χαρακτήρων μέσα σε διπλά εισαγωγικά** (" ") η οποία καθορίζει τον τρόπο με τον οποίο θα εμφανιστούν τα δεδομένα στην οθόνη
 - ♦ Οι επόμενες παράμετροι είναι **προαιρετικές** και, αν υπάρχουν, η `printf()` εμφανίζει τις τιμές τους στην οθόνη
- Το αλφαριθμητικό μορφοποίησης (*format string*) μπορεί να περιέχει:
 - ♦ **Απλούς χαρακτήρες** (οι οποίοι εμφανίζονται όπως είναι στην οθόνη)
 - ♦ **Ακολουθίες Διαφυγής**
 - ♦ **Προσδιοριστικά Μετατροπής**

Ακολουθία Διαφυγής

- Μία ακολουθία διαφυγής (escape sequence) χρησιμοποιείται είτε για να μετακινηθεί ο δρομέας (cursor) σε κάποια θέση της οθόνης είτε για την εμφάνιση κάποιων ειδικών χαρακτήρων
- Μία ακολουθία διαφυγής αποτελείται από μία ανάστροφη κεκλιμένη (\) (backslash) και έναν ειδικό χαρακτήρα

Ακολουθία διαφυγής	Σημασία
<code>\a</code>	Χρησιμοποιείται για τη δημιουργία ηχητικού σήματος.
<code>\b</code>	Χρησιμοποιείται για τη διαγραφή του τελευταίου χαρακτήρα, όπως το πλήκτρο backspace.
<code>\n</code>	Χρησιμοποιείται για την αλλαγή γραμμής, όπως το πλήκτρο Enter.
<code>\r</code>	Χρησιμοποιείται για την επαναφορά του δρομέα στην αρχή της τρέχουσας γραμμής.
<code>\t</code>	Χρησιμοποιείται για τη μετακίνηση του δρομέα σε μία απόσταση ίση με το μήκος του tab, όπως το πλήκτρο tab.
<code>\\</code>	Χρησιμοποιείται για την εμφάνιση της ανάστροφης κεκλιμένης (\).
<code>\"</code>	Χρησιμοποιείται για την εμφάνιση των διπλών εισαγωγικών (").

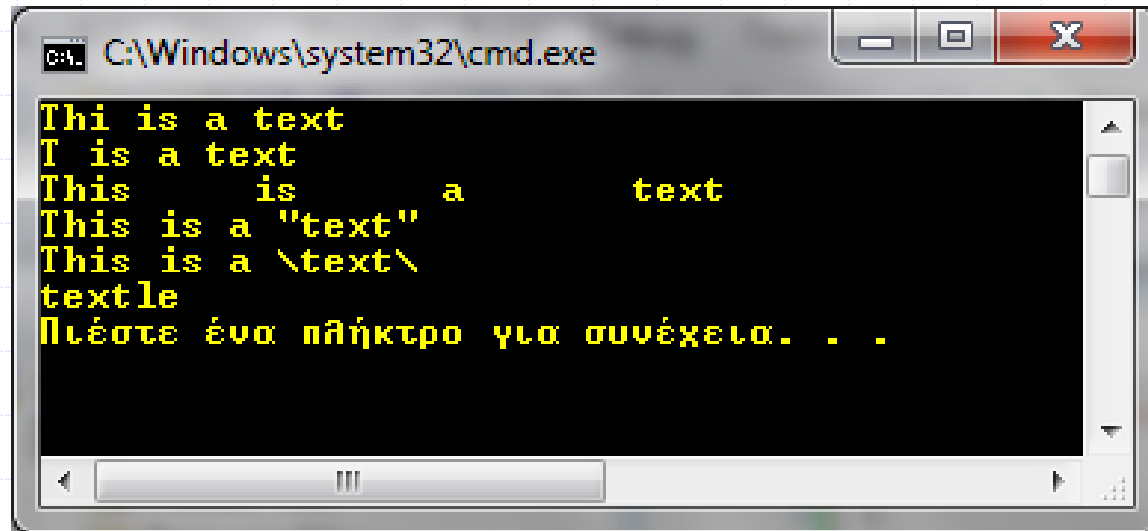
Προσδιοριστικό Μετατροπής

- Ένα προσδιοριστικό μετατροπής (conversion specification) αρχίζει με τον χαρακτήρα % και ακολουθείται από έναν ή περισσότερους **χαρακτήρες μετατροπής** που έχουν ειδική σημασία

Χαρακτήρας μετατροπής	Σημασία
c	Χρησιμοποιείται για την εμφάνιση του χαρακτήρα που αντιστοιχεί σε μία ακέραια τιμή.
d ή i	Χρησιμοποιείται για την εμφάνιση ενός ακεραίου.
u	Χρησιμοποιείται για την εμφάνιση ενός μη-προσημασμένου ακεραίου.
f	Χρησιμοποιείται για την εμφάνιση ενός πραγματικού αριθμού. Η εξ' ορισμού ακρίβεια είναι έξι δεκαδικά ψηφία.
s	Χρησιμοποιείται για την εμφάνιση των χαρακτήρων ενός αλφαριθμητικού.
e ή E	Χρησιμοποιείται για την εμφάνιση ενός πραγματικού αριθμού σε επιστημονική μορφή. Ανάλογα με την επιλογή, εμφανίζεται το γράμμα e ή E πριν από τον εκθέτη.
g ή G	Χρησιμοποιείται για την εμφάνιση ενός πραγματικού αριθμού σε κανονική ή επιστημονική μορφή.
p	Χρησιμοποιείται για την εμφάνιση μίας διεύθυνσης μνήμης σε δεκαεξαδική μορφή.
x ή X	Χρησιμοποιείται για την εμφάνιση ενός μη-προσημασμένου ακεραίου σε δεκαεξαδική μορφή. Με το %x τα δεκαεξαδικά ψηφία (a-f) εμφανίζονται πεζά, ενώ με το %X εμφανίζονται ως κεφαλαία (A-F).
o	Χρησιμοποιείται για την εμφάνιση ενός μη-προσημασμένου ακεραίου σε οκταδική μορφή.
%	Χρησιμοποιείται για την εμφάνιση του χαρακτήρα %.

Παραδείγματα (I)

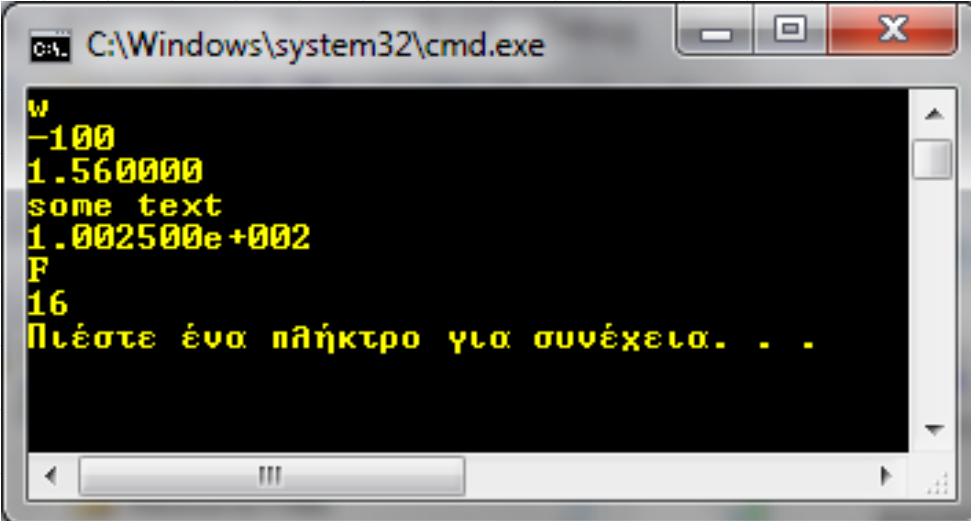
```
#include <stdio.h>
int main(void)
{
    printf("\a");
    printf("This\b is a text\n");
    printf("This\b\b\b is a text\n");
    printf("This\t is\t a\t text\n");
    printf("This is a \"text\"\n");
    printf("This is a \\text\\\n");
    printf("Sample\rtext\n");
    return 0;
}
```



```
C:\Windows\system32\cmd.exe
Thi is a text
T is a text
This is a text
This is a "text"
This is a \text\
textle
Πέστε ένα πλήκτρο για συνέχεια. . .
```

Παραδείγματα (II)

```
#include <stdio.h>
int main(void)
{
    printf("%c\n", 'w');
    printf("%d\n", -100);
    printf("%f\n", 1.56);
    printf("%s\n", "some text");
    printf("%e\n", 100.25);
    printf("%X\n", 15);
    printf("%o\n", 14);
    return 0;
}
```



A screenshot of a Windows command prompt window. The title bar shows the path "C:\Windows\system32\cmd.exe". The window contains the following output from the C program:

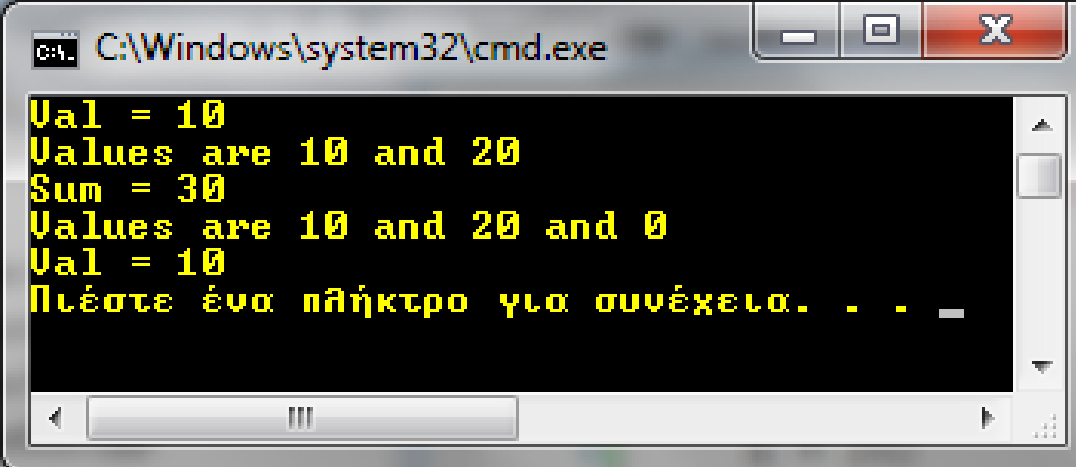
```
w
-100
1.560000
some text
1.002500e+002
F
16
Πιέστε ένα πλήκτρο για συνέχεια. . .
```


Εμφάνιση μεταβλητών

- Τα ονόματα των μεταβλητών εισάγονται στην `printf()` μετά τα διπλά εισαγωγικά (" ") με τη χρήση κόμματος (,) και αν οι μεταβλητές είναι περισσότερες από μία πρέπει και αυτές να διαχωρίζονται μεταξύ τους με κόμμα (,)
- Ο μεταγλωττιστής αντιστοιχίζει ένα-προς-ένα, από αριστερά προς τα δεξιά, τα ονόματα των μεταβλητών με τα προσδιοριστικά μετατροπής
- Αν τα προσδιοριστικά μετατροπής είναι περισσότερα από τις μεταβλητές, τότε για τα πρόσθετα προσδιοριστικά εμφανίζονται τυχαίες τιμές (σκουπίδια)
- Αντίστοιχα, αν τα προσδιοριστικά μετατροπής είναι λιγότερα από τις μεταβλητές, τότε δεν εμφανίζονται οι τιμές των πρόσθετων μεταβλητών

Παράδειγμα

```
#include <stdio.h>
int main(void)
{
    int a,b;
    a = 10;
    b = 20;
    printf("Val = %d\n",a);
    printf("Values are %d and %d\n",a,b);
    printf("Sum = %d\n",a+b);
    printf("Values are %d and %d and %d\n",a,b);
    printf("Val = %d\n",a,b);
    return 0;
}
```

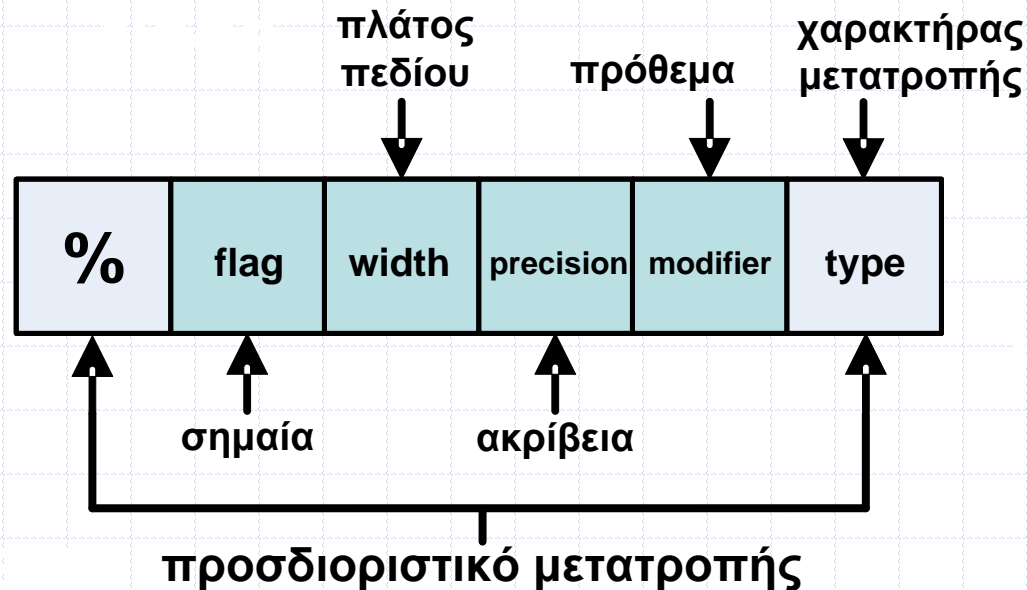


The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The output of the program is displayed in yellow text on a black background:

```
Val = 10
Values are 10 and 20
Sum = 30
Values are 10 and 20 and 0
Val = 10
Πιέστε ένα πλήκτρο για συνέχεια. . . -
```

Προαιρετικά Πεδία

- Ένα προσδιοριστικό μετατροπής, στην απλή μορφή του, αρχίζει με τον χαρακτήρα % και ακολουθείται από τον κατάλληλο χαρακτήρα μετατροπής
- Όμως, ανάμεσά τους, μπορεί να περιέχονται έως και τέσσερα επιπλέον προαιρετικά πεδία, όπως φαίνεται στο σχήμα



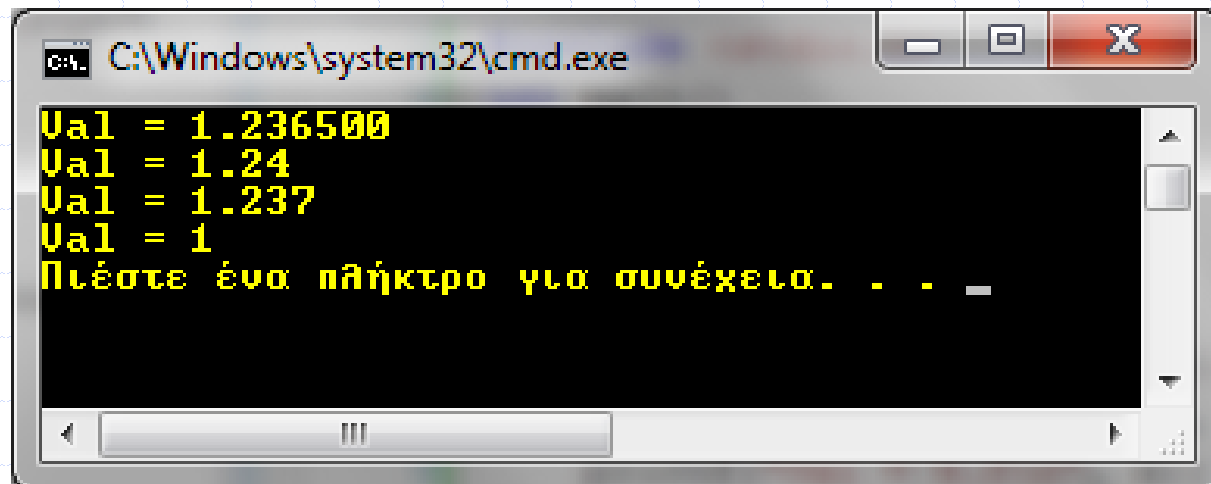
Ακρίβεια

- Όταν εμφανίζουμε την τιμή μίας πραγματικής μεταβλητής (τύπου `float` ή `double`) μπορούμε να καθορίσουμε πόσα ψηφία ακρίβειας θα εμφανιστούν στην οθόνη
- Εξ' ορισμού (by default) εμφανίζονται έξι δεκαδικά ψηφία
- Αν δεν επιθυμούμε να εμφανιστούν έξι ψηφία, τότε μετά τον χαρακτήρα % πρέπει να προσθέσουμε την τελεία (.) και
 - ◆ είτε έναν ακέραιο αριθμό που να δηλώνει το επιθυμητό πλήθος των δεκαδικών ψηφίων
 - ◆ είτε τον χαρακτήρα * και έναν ακέραιο στη λίστα των μεταβλητών που να δηλώνει το επιθυμητό πλήθος των δεκαδικών ψηφίων
- Η τελική τιμή του πραγματικού αριθμού που εμφανίζεται με την `printf()` στρογγυλοποιείται προς τα πάνω ή προς τα κάτω, ανάλογα με το αν η τιμή του πρώτου ψηφίου που αποκόπτεται είναι μεγαλύτερη ή όχι από το 4, αντίστοιχα
- Αν δεν θέλουμε να εμφανιστούν δεκαδικά ψηφία, τότε προσθέτουμε μόνο την τελεία (.), χωρίς αυτή να ακολουθείται από κάποιον αριθμό

Παράδειγμα

```
#include <stdio.h>
int main(void)
{
    float a = 1.2365;

    printf("Val = %f\n", a);
    printf("Val = %.2f\n", a);
    printf("Val = %.*f\n", 3, a);
    printf("Val = %.f\n", a);
    return 0;
}
```



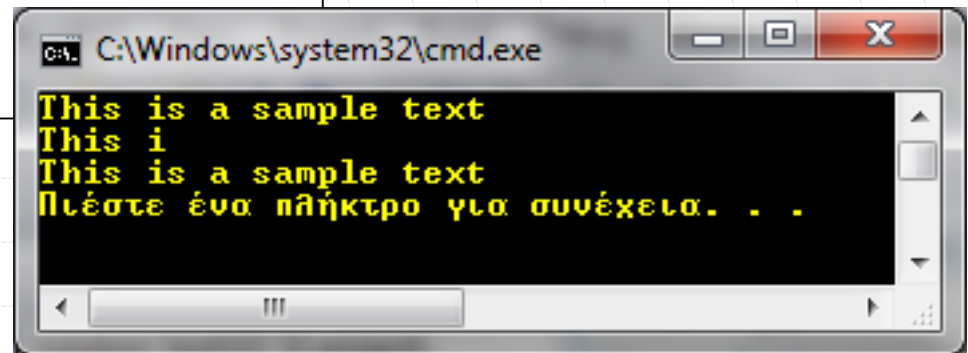
The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The output of the program is displayed in yellow text on a black background:

```
Val = 1.236500
Val = 1.24
Val = 1.237
Val = 1
Πιέστε ένα πλήκτρο για συνέχεια. . .
```

Εμφάνιση χαρακτήρων σε Αλφαριθμητικό

- Όταν θέλουμε να εμφανίσουμε ένα αλφαριθμητικό, δηλαδή μία ακολουθία χαρακτήρων, μπορούμε να καθορίσουμε πόσοι χαρακτήρες του θα εμφανιστούν, ακολουθώντας την ίδια τεχνική με προηγουμένως
- Αν ο αριθμός που θα δηλωθεί υπερβαίνει το πλήθος των χαρακτήρων, τότε εμφανίζονται όλοι οι χαρακτήρες του αλφαριθμητικού και δεν προστίθεται κανένας άλλος

```
#include <stdio.h>
int main(void)
{
    char msg[] = "This is a sample text";
    printf("%s\n",msg);
    printf("%.6s\n",msg);
    printf("%.30s\n",msg);
    return 0;
}
```



```
C:\Windows\system32\cmd.exe
This is a sample text
This i
This is a sample text
Πιέστε ένα πλήκτρο για συνέχεια. . .
```

Πλάτος Πεδίου

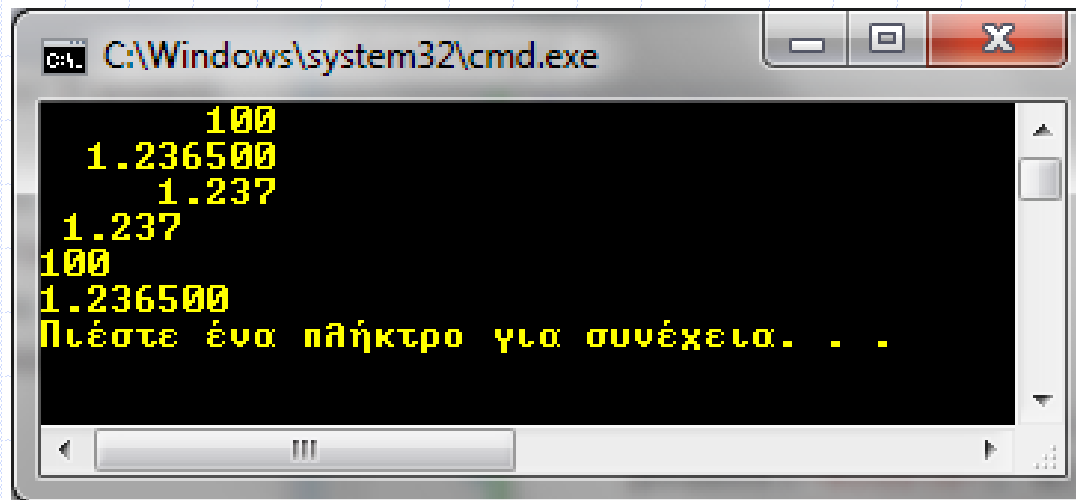
- Όταν εμφανίζουμε την τιμή μίας ακέραιας ή πραγματικής μεταβλητής μπορούμε να καθορίσουμε το συνολικό πλήθος των χαρακτήρων που θα εμφανιστούν στην οθόνη, μαζί με τα ψηφία ακρίβειας και την υποδιαστολή
- Για να καθορίσουμε το συνολικό πλήθος εισάγουμε:
 - ◆ είτε έναν ακέραιο αριθμό αμέσως μετά από τον χαρακτήρα $\%$, ο οποίος ονομάζεται πλάτος πεδίου
 - ◆ είτε τον χαρακτήρα $*$ και έναν ακέραιο στη λίστα μεταβλητών, ο οποίος δηλώνει το πλάτος πεδίου
- Αν η τιμή της μεταβλητής χρειάζεται λιγότερους χαρακτήρες από το δηλωμένο πλάτος, τότε στην έξοδο προστίθενται κενοί χαρακτήρες από αριστερά προς τα δεξιά μέχρι να συμπληρωθεί το συνολικό πλήθος των χαρακτήρων
- Αν η τιμή της μεταβλητής χρειάζεται περισσότερους χαρακτήρες από το δηλωμένο πλάτος, τότε ο αριθμός αυτός δεν λαμβάνεται υπόψη και η τιμή της μεταβλητής εμφανίζεται με όσους χαρακτήρες απαιτείται

ΣΗΜΕΙΩΣΗ: Προφανώς, σε περίπτωση πραγματικού αριθμού, μετά το πλάτος του πεδίου μπορεί να καθοριστεί και η ακρίβεια του πραγματικού αριθμού (με χρήση της τελείας και ενός αριθμού που έπεται αυτής και δηλώνει το πλήθος των δεκαδικών ψηφίων, όπως δείξαμε προηγουμένως)

Παράδειγμα

```
#include <stdio.h>
int main(void)
{
    int a = 100;
    float b = 1.2365;

    printf("%10d\n", a);
    printf("%10f\n", b);
    printf("%10.3f\n", b);
    printf("%*.3f\n", 6, b);
    printf("%2d\n", a);
    printf("%6f\n", b);
    return 0;
}
```



The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The output of the program is displayed in yellow text on a black background. The output consists of six lines: "100", "1.236500", "1.237", "1.237", "100", and "1.236500". Below the output, the prompt "Πιέστε ένα πλήκτρο για συνέχεια. . ." is visible.

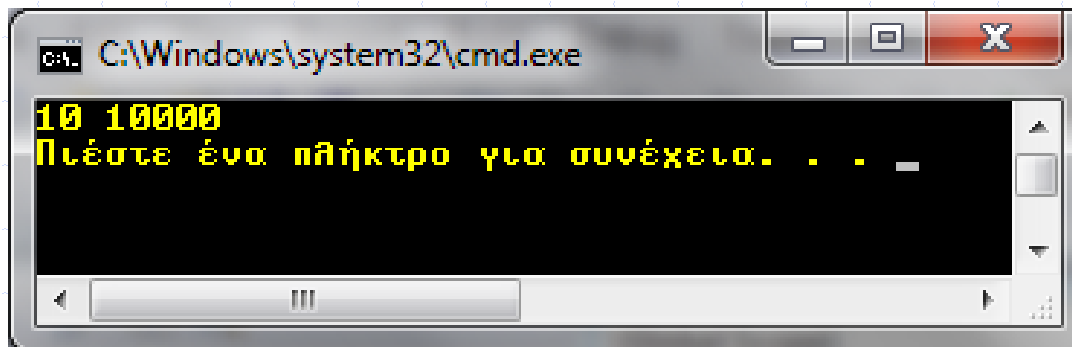
```
C:\Windows\system32\cmd.exe
100
1.236500
1.237
1.237
100
1.236500
Πιέστε ένα πλήκτρο για συνέχεια. . .
```


Πρόθεμα

- Για την εμφάνιση ενός `short` ακεραίου μπορεί να χρησιμοποιηθεί προαιρετικά το γράμμα `h`, ενώ για την εμφάνιση ενός `long` ακεραίου μπορεί να χρησιμοποιηθεί το γράμμα `l` ή `L`, όπως φαίνεται στο παρακάτω παράδειγμα

```
#include <stdio.h>
int main(void)
{
    short a = 10;
    long b = 10000;

    printf("%hd %ld\n", a, b);
    return 0;
}
```



The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The output of the program is displayed in yellow text on a black background: "10 10000" followed by a prompt "Πιέστε ένα πλήκτρο για συνέχεια. . . -".

Σημαίες

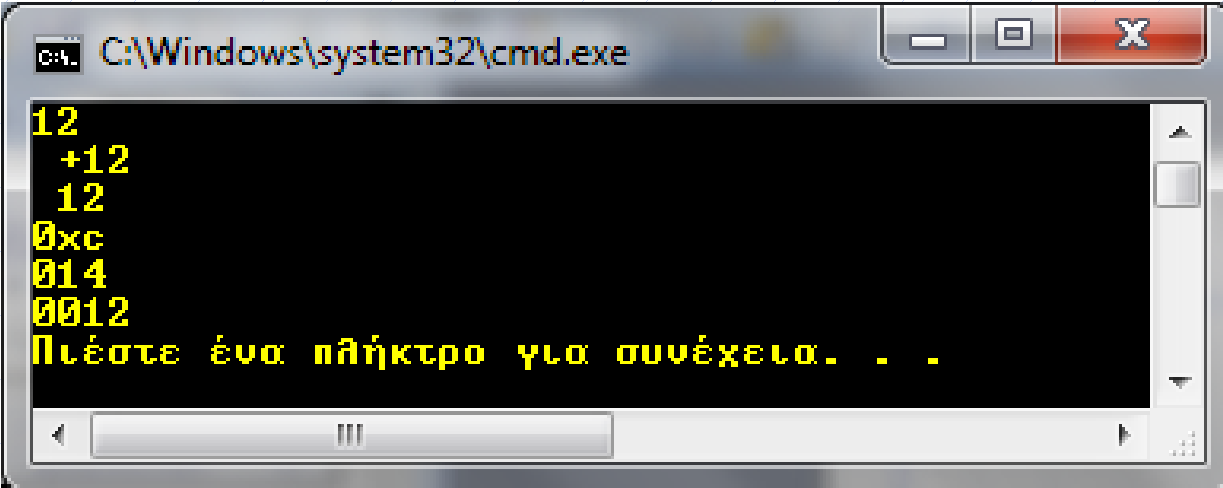
- Οι σημαίες χρησιμοποιούνται για περαιτέρω μορφοποίηση των εμφανιζόμενων τιμών, όπως φαίνεται στον παρακάτω πίνακα

Σημαία	Σημασία
-	Η έξοδος στο πεδίο πλάτους στοιχίζεται αριστερά (εξ' ορισμού η έξοδος στοιχίζεται δεξιά).
+	Προσθέτει το πρόσημο + μπροστά από τις θετικές τιμές.
κενός χαρακτήρας	Προσθέτει τον κενό χαρακτήρα μπροστά από τις θετικές τιμές.
#	Προσθέτει το 0 μπροστά από τις οκταδικές τιμές και το 0x ή το 0X μπροστά από δεκαεξαδικές τιμές.
0	Προσθέτει όσα μηδενικά χρειάζονται μπροστά από την εμφανιζόμενη τιμή, ώστε να καλυφθεί το πεδίο πλάτους.

Παράδειγμα

```
#include <stdio.h>
int main(void)
{
    int a = 12;

    printf("%-4d\n", a);
    printf("%+4d\n", a);
    printf("% d\n", a);
    printf("%#0x\n", a);
    printf("%#o\n", a);
    printf("%04d\n", a);
    return 0;
}
```



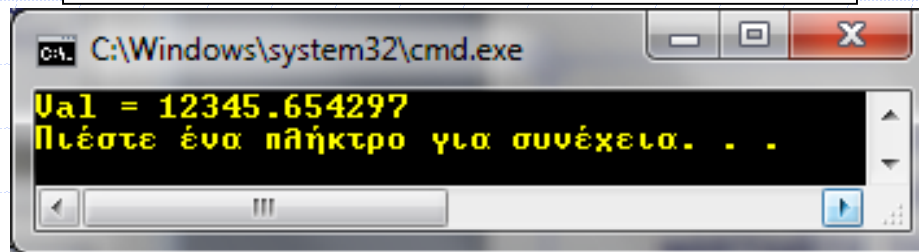
C:\Windows\system32\cmd.exe

```
12
+12
 12
0xc
014
0012
Πιέστε ένα πλήκτρο για συνέχεια. . .
```

Παρατηρήσεις (I)

- Όταν χρησιμοποιείτε συχνά μία πραγματική μεταβλητή σε διάφορες εκφράσεις μέσα στο πρόγραμμα (π.χ. συγκρίσεις, πράξεις, ...), τότε να προτιμάτε τον τύπο `double` και όχι τον τύπο `float`, γιατί είναι πιθανό να μην γίνει η διαχείριση των δεκαδικών ψηφίων με τον τρόπο που θα αναμένετε
- Π.χ. το επόμενο πρόγραμμα μπορεί να μην εμφανίσει την τιμή 12345.65432, αλλά μία τιμή παραπλήσια σε αυτή.

```
#include <stdio.h>
int main(void)
{
    float a;
    a = 12345.65432;
    printf("Val = %f\n",a);
    return 0;
}
```



```
C:\Windows\system32\cmd.exe
Val = 12345.654297
Πιέστε ένα πλήκτρο για συνέχεια. . .
```

Παρατηρήσεις (II)

- Αν θέλετε η ακολουθία χαρακτήρων της `printf()`, για λόγους εμφάνισης, να εκτείνεται σε περισσότερες από μία γραμμές στον κώδικά σας, τότε να χρησιμοποιείτε τον χαρακτήρα της ανάστροφης κεκλιμένης `'\'` (backslash)

- Π.χ. ο κώδικας της παρακάτω `printf()` εκτείνεται σε δύο γραμμές

```
printf("This printf takes two lines. However, the \
message is shown in the same line ");
```

- Όμως, σαν αποτέλεσμα στην οθόνη, όλοι οι χαρακτήρες του μηνύματος θα εμφανίζονται στην ίδια γραμμή
- Λόγω της ειδικής σημασίας του χαρακτήρα `%`, για την εμφάνιση του χαρακτήρα `'%'` πρέπει να γραφούν δύο χαρακτήρες `%`
- Π.χ. η επόμενη `printf()` εμφανίζει το μήνυμα `100%` στην οθόνη

```
printf("%d%%\n",100);
```

Μετατροπή Τύπου (type cast)

- Υπάρχουν περιπτώσεις όπου ένας τύπος δεδομένων πρέπει να μετατραπεί **προσωρινά** σε κάποιον άλλο τύπο δεδομένων
- Π.χ. είναι πιθανό σε ένα σημείο του προγράμματος μία ακέραια μεταβλητή που έχει δηλωθεί σαν `int` να πρέπει να μετατραπεί προσωρινά σε μία πραγματική τύπου `float`, ή και το αντίστροφο
- Η γενική μορφή μίας τέτοιας μετατροπής είναι:

(τύπος_δεδομένων) (παράσταση)

- Π.χ. αν η μεταβλητή `a` έχει δηλωθεί:

```
int a;
```

τότε η έκφραση:

```
(float) a;
```

προσαρμόζει προσωρινά τον τύπο της `a` από `int` σε `float`

- Λέγοντας προσωρινά, εννοούμε ότι στη συνέχεια του προγράμματος ο τύπος της μεταβλητής `a` συνεχίζει να είναι `int`

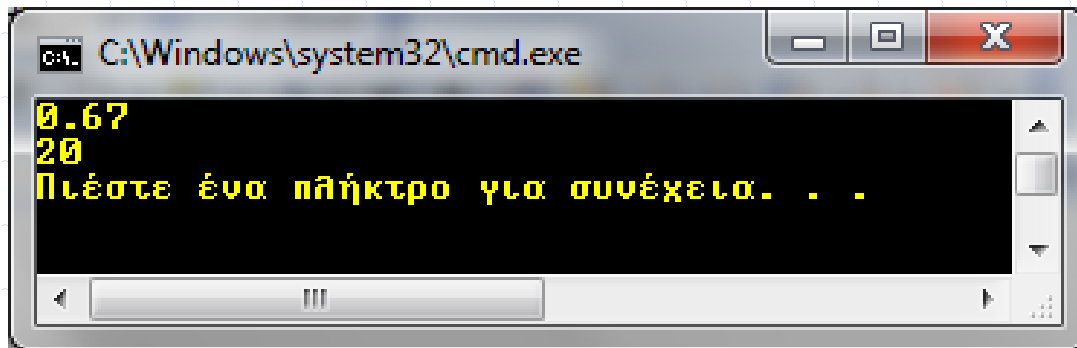
Παράδειγμα

```
#include <stdio.h>
int main(void)
{
    int i = 20, j = 30;
    float k;

    k = (float)i/j;

    printf("%.2f\n", k);
    printf("%d\n", i);
    return 0;
}
```

- ♦ Η έκφραση `(float)i` προσαρμόζει προσωρινά τον τύπο της μεταβλητής `i` από `int` σε `float`, έτσι ώστε το αποτέλεσμα της διαίρεσης να είναι πραγματικός αριθμός
- ♦ Αν γράφαμε `k = i/j`, τότε η τιμή του `k` θα ισούνταν με το αποτέλεσμα της ακέραιας διαίρεσης `20/30`, άρα υπό μορφή `float` με ακρίβεια 2 δεκαδικών ψηφίων (λόγω της συγκεκριμένης `printf()`) η τιμή του `k` θα ήταν `0.00`



```
C:\Windows\system32\cmd.exe
0.67
20
Πιέστε ένα πλήκτρο για συνέχεια. . .
```