

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 (").

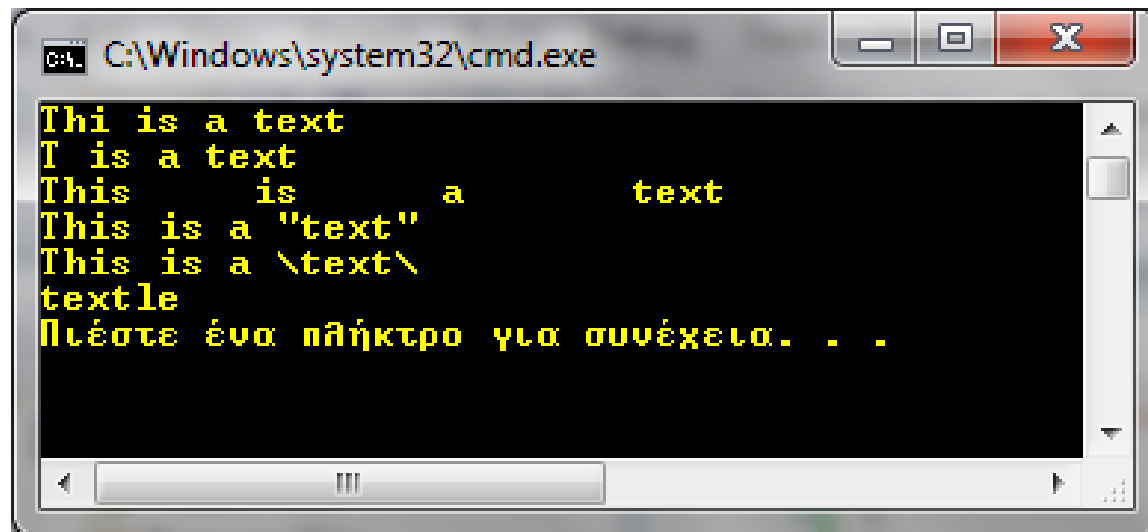
Προσδιοριστικό Μετατροπής

- 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()
{
    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;
}
```

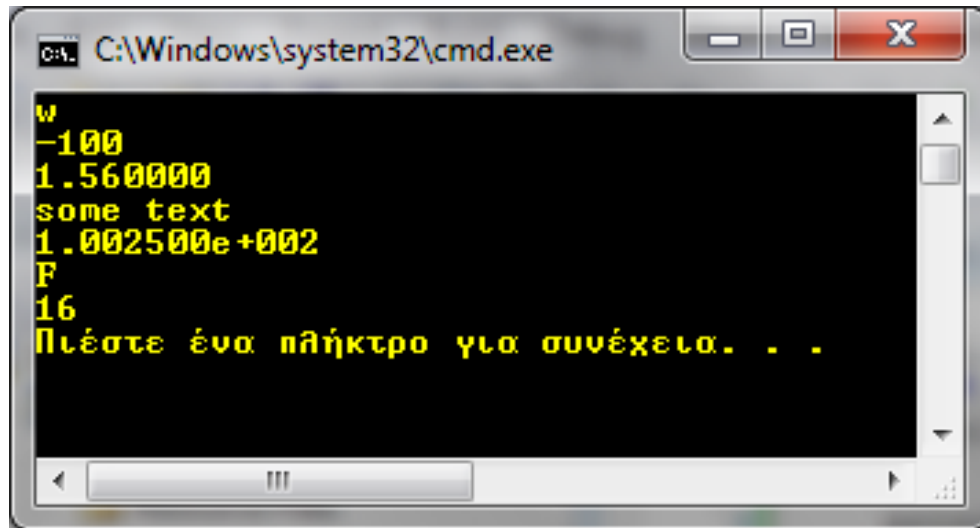


A screenshot of a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The window has standard Windows window controls (minimize, maximize, close). The output of the C program is displayed in yellow text on a black background. The output is as follows:

```
This is a text
I is a text
This   is       a       text
This is a "text"
This is a \text\
textle
Πιέστε ένα πλήκτρο για συνέχεια. . .
```

Παραδείγματα (II)

```
#include <stdio.h>
int main()
{
    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;
}
```



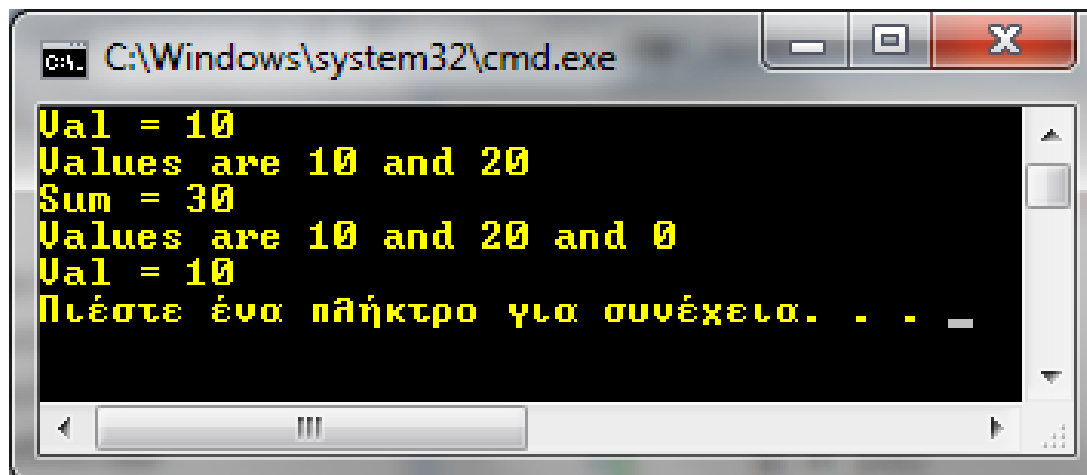
A screenshot of a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The window has a black background with yellow text. The output of the program is displayed line by line: 'w', -100, 1.560000, some text, 1.002500e+002, F, 16, and Πιέστε ένα πλήκτρο για συνέχεια. . . . The window includes standard Windows window controls (minimize, maximize, close) in the top right corner and a scroll bar on the right side.

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()
{
    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;
}
```



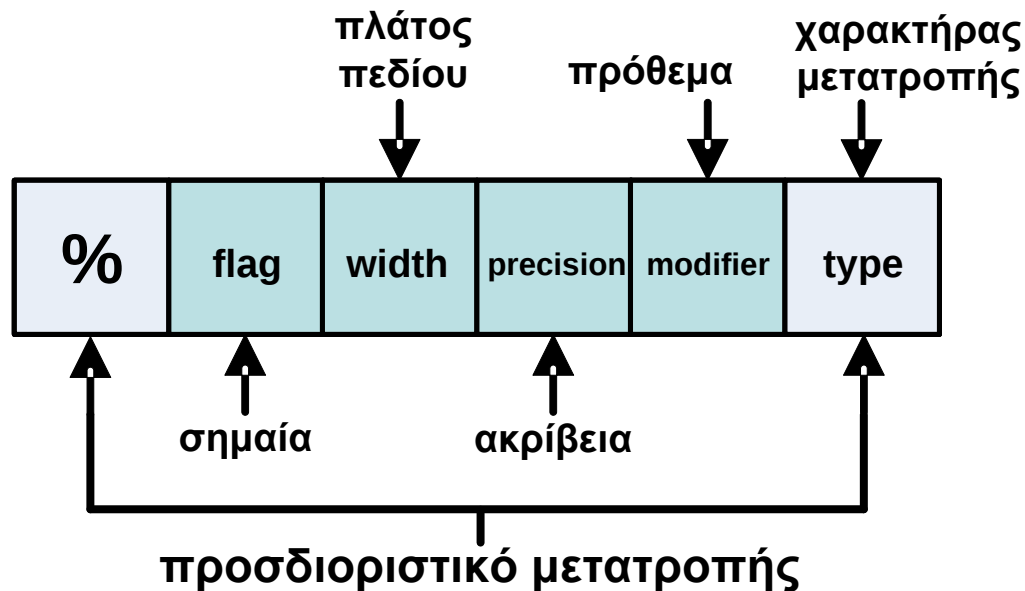
A screenshot of a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The window has a black background with yellow text. The output of the program is displayed as follows:

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

The window includes standard Windows window controls (minimize, maximize, close) and a scrollbar on the right side.

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



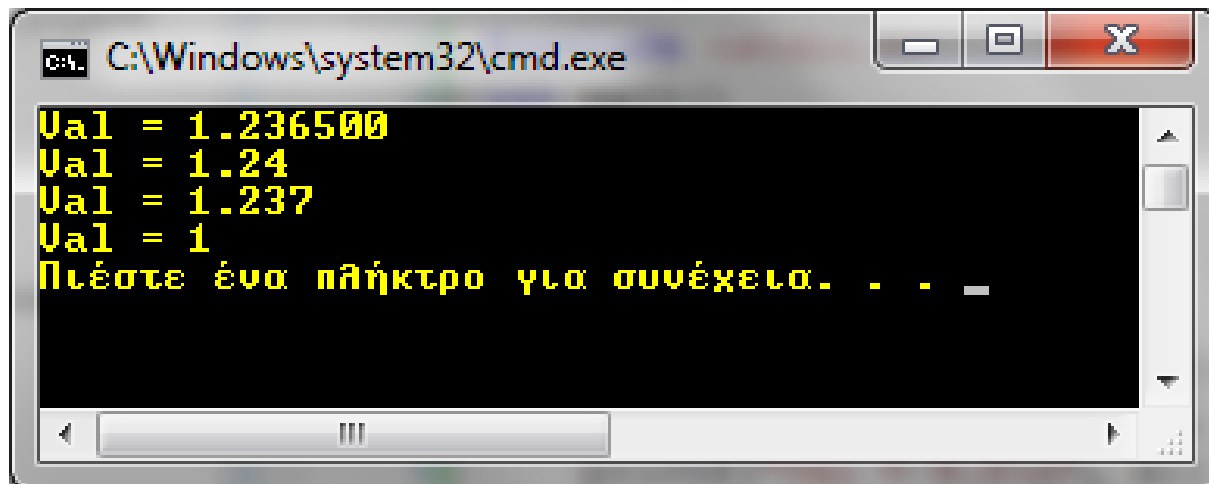
Ακρίβεια

- 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()
{
    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. . . _
```

Πλάτος Πεδίου

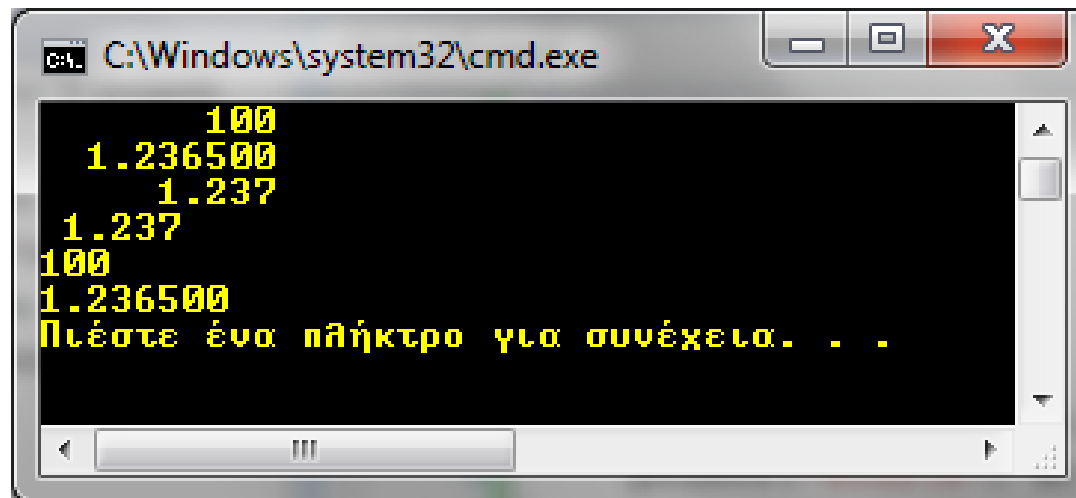
- Όταν εμφανίζουμε την τιμή μίας ακέραιας ή πραγματικής μεταβλητής μπορούμε να καθορίσουμε το συνολικό πλήθος των χαρακτήρων που θα εμφανιστούν στην οθόνη, μαζί με τα ψηφία ακρίβειας και την υποδιαστολή
- Για να καθορίσουμε το συνολικό πλήθος εισάγουμε:
 - ♦ είτε έναν ακέραιο αριθμό αμέσως μετά από τον 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()
{
    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 of formatted numbers, followed by a prompt for user input.

```
      100
    1.236500
      1.237
    1.237
    100
    1.236500
Πιέστε ένα πλήκτρο για συνέχεια. . .
```

Σημαίες

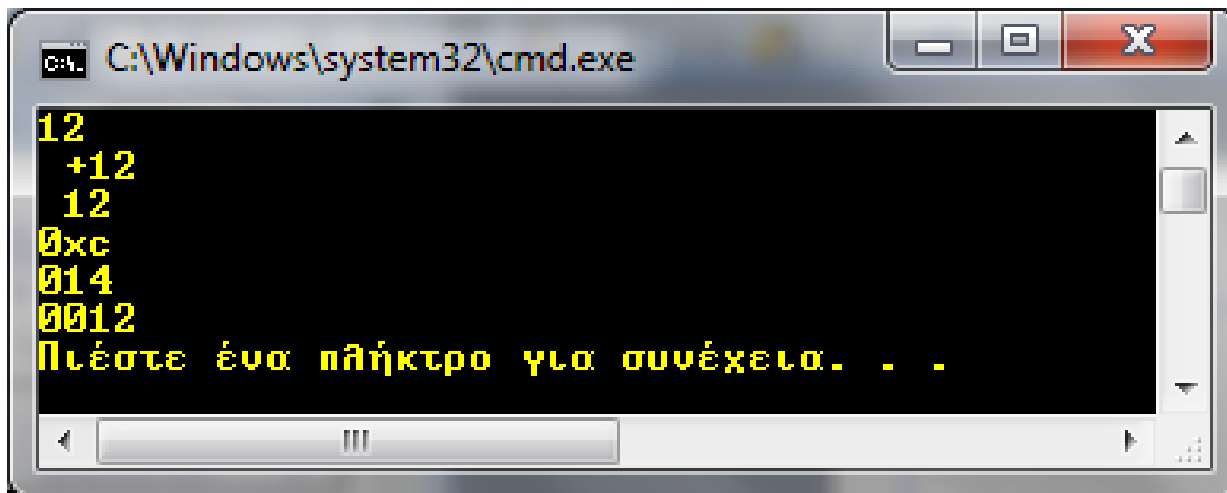
- Οι σημαίες χρησιμοποιούνται για περαιτέρω μορφοποίηση των εμφανιζόμενων τιμών, όπως φαίνεται στον παρακάτω πίνακα

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()
{
    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;
}
```



A screenshot of a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The window has a black background and yellow text. The output of the program is displayed as follows:

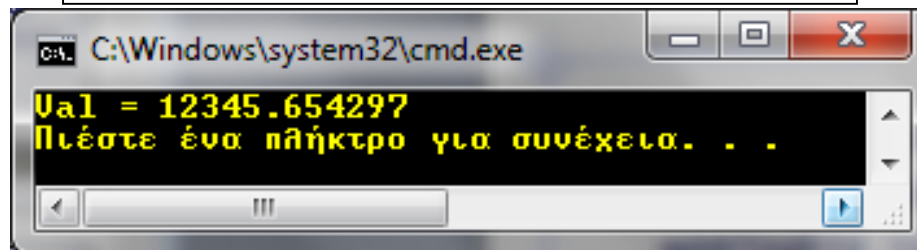
```
12
+12
 12
0xc
014
0012
Πιέστε ένα πλήκτρο για συνέχεια. . .
```

The output shows the integer value 12 in various formats: left-aligned, right-aligned with a sign, right-aligned with a space, hexadecimal, octal, and zero-padded decimal. The prompt then asks the user to press a key to continue.

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()
{
    float a;
    a = 12345.65432;
    printf("Val = %f\n",a);
    return 0;
}
```



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 \nmessage 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);
```

Η συνάρτηση `scanf()`

- Η συνάρτηση `scanf()` χρησιμοποιείται για την είσοδο δεδομένων από ένα αρχείο εισόδου, το οποίο ονομάζεται `stdin` (*standard input stream*) και εξ' ορισμού συνδέεται με το πληκτρολόγιο
- Η `scanf()` δέχεται μία μεταβλητή λίστα παραμέτρων, παρόμοια με την `printf()`, δηλαδή:
 - ♦ Η πρώτη παράμετρος είναι ένα **αλφαριθμητικό μορφοποίησης** (*format string*), το οποίο, συνήθως, περιέχει μόνο απλά προσδιοριστικά μετατροπής (π.χ. `%d` για μεταβλητές τύπου `int`, `%f` για μεταβλητές τύπου `float` κτλ...)
 - ♦ Οι επόμενες προαιρετικές παράμετροι είναι οι **διευθύνσεις μνήμης** των μεταβλητών στις οποίες θα εκχωρηθούν τα δεδομένα που θα εισάγει ο χρήστης από το πληκτρολόγιο



Κάθε προσδιοριστικό μετατροπής πρέπει να αντιστοιχεί σε μία διεύθυνση μεταβλητής και η αντιστοίχιση γίνεται ένα προς ένα



Για μεταβλητές τύπου `double`, χρησιμοποιείται το προσδιοριστικό μετατροπής `%lf` και όχι το `%f`, το οποίο χρησιμοποιείται μόνο για μεταβλητές τύπου `float`

Παράδειγμα 1

- Π.χ.

```
int i;  
scanf("%d", &i);
```

- Ο χαρακτήρας &, που μπαίνει πριν από το όνομα της μεταβλητής, ονομάζεται **τελεστής διεύθυνσης** και χρησιμοποιείται για να αποθηκευτεί ο αριθμός που θα εισάγει ο χρήστης στη διεύθυνση μνήμης της μεταβλητής `i`
- Περισσότερες λεπτομέρειες για τη σημασία του τελεστή διεύθυνσης & θα δούμε αναλυτικά στους "δείκτες της C"

Παράδειγμα 2 (I)

- Μπορούμε να χρησιμοποιήσουμε τη συνάρτηση `scanf()` για να διαβάσουμε περισσότερες από μία τιμές από το πληκτρολόγιο και να τις αποθηκεύσουμε ως τιμές σε κάποιες μεταβλητές του προγράμματος
- Π.χ.

```
int i;  
float j;  
scanf("%d%f", &i, &j);
```

- Η πρώτη παράμετρος της `scanf()` είναι το αλφαριθμητικό μορφοποίησης `%d%f`, ενώ οι επόμενες παράμετροι είναι οι διευθύνσεις μνήμης των μεταβλητών `i` και `j` αντίστοιχα
 - ♦ Το `%d` αντιστοιχεί στη διεύθυνση της μεταβλητής `i`
 - ♦ Το `%f` αντιστοιχεί στη διεύθυνση της μεταβλητής `j`
 - ♦ Δηλαδή η αντιστοίχιση γίνεται ένα προς ένα και από αριστερά προς τα δεξιά
- Για την είσοδο των δεδομένων χρησιμοποιείται συνήθως το «κενό διάστημα» (space) μεταξύ των διαφορετικών τιμών που εισάγονται, δεδομένου ότι κατά το διάβασμα αριθμητικών τιμών, η `scanf()` αγνοεί όλα τα λευκά διαστήματα (π.χ. κενά διαστήματα, tab, χαρακτήρα νέας γραμμής) που μπορεί να υπάρχουν πριν από κάθε αριθμητική τιμή

Παράδειγμα 2 (II)

- Στο προηγούμενο παράδειγμα, αν ο χρήστης εισάγει π.χ. τις τιμές 10 και 4.65, αυτές θα πρέπει να απέχουν μεταξύ τους ένα ή περισσότερα κενά
- Για να ληφθούν από τη συνάρτηση `scanf()` πρέπει μετά ο χρήστης να πατήσει Enter
- Τότε, η τιμή 10 αποθηκεύεται στη μεταβλητή `i` και η τιμή 4.65 στη μεταβλητή `j`, αντίστοιχα

Παρατηρήσεις



Να θυμάστε ότι η `scanf()` απαιτεί τον τελεστή διεύθυνσης & πριν από το όνομα κάθε αριθμητικής μεταβλητής (π.χ. `int`, `double`, `char`, `float`, ...)

Αν τον ξεχάσετε, το πρόγραμμά σας δεν θα εκτελεστεί σωστά

Αντίθετα, όταν ο τύπος της μεταβλητής είναι δείκτης, ο τελεστής διεύθυνσης δεν χρειάζεται



Για να διαβάσετε με τη `scanf()` ένα αλφαριθμητικό που μπορεί να αποτελείται από πολλές λέξεις (π.χ. "Text with multiple words"), πρέπει να χρησιμοποιήσετε μία πιο σύνθετη μορφή της,

π.χ.: `scanf("%[^\\n]", str);`

γιατί η `scanf()` εξ' ορισμού σταματάει το διάβασμα του αλφαριθμητικού όταν συναντήσει έναν κενό χαρακτήρα

Τί επιστρέφει η συνάρτηση `scanf()` ???

- Η συνάρτηση `scanf()` επιστρέφει έναν ακέραιο αριθμό που δηλώνει πόσα από τα δεδομένα εισόδου διαβάστηκαν και εκχωρήθηκαν στις μεταβλητές του προγράμματος, ενώ οι τιμές που δεν διαβάστηκαν παραμένουν στο `stdin`
- Π.χ. στο παράδειγμα:

```
int i;  
scanf("%d", &i);
```

αν ο χρήστης εισάγει έναν ακέραιο, η συνάρτηση `scanf()` επιστρέφει την τιμή 1

- Ενώ στο παράδειγμα:

```
int i;  
float j;  
scanf("%d%f", &i, &j);
```

αν ο χρήστης εισάγει έναν ακέραιο και έναν πραγματικό αριθμό, η συνάρτηση `scanf()` επιστρέφει την τιμή 2

Παρατηρήσεις

- Η συνάρτηση `scanf()` δεν είναι μία ασφαλής συνάρτηση και πρέπει να τη χρησιμοποιείτε με μεγάλη προσοχή
 - ♦ Σε επαγγελματικές εφαρμογές πρέπει να ελέγχετε την τιμή επιστροφής της

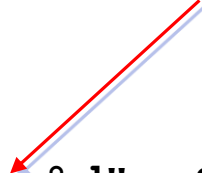
```
#include <stdio.h>
int main()
{
    int num;
    printf("Enter number: ");
    while(scanf("%d", &num) != 1)
    {
        printf("Enter number: ");
        while(getchar() != '\n'); /* Καθαρισμός μνήμης
πληκτρολογίου. */
    }
    printf("Inserted value: %d\n", num);
    return 0;
}
```

- ♦ Σχεδόν σε όλα τα προγράμματα που θα υλοποιήσουμε, όταν χρησιμοποιούμε τη συνάρτηση `scanf()`, για λόγους απλότητας, δεν θα ελέγχουμε την τιμή επιστροφής της και θα θεωρούμε ότι οι τιμές που εισάγει ο χρήστης θα είναι σε συμφωνία με τα προσδιοριστικά μετατροπής

Παρεμβολή απλών χαρακτήρων στη scanf() (I)

- Στην πιο συνηθισμένη χρήση της, το αλφαριθμητικό μορφοποίησης της scanf() δεν περιέχει απλούς χαρακτήρες παρά μόνο τα προσδιοριστικά μετατροπής (π.χ. %d, %f,...)
- Ωστόσο, αν παρεμβληθούν κάποιοι απλοί χαρακτήρες, τότε πρέπει οι αντίστοιχοι χαρακτήρες να εισαχθούν και από το πληκτρολόγιο
- Π.χ. στην επόμενη scanf() παρεμβάλλεται ο χαρακτήρας κόμμα (,)

```
#include <stdio.h>
int main()
{
    int a, b;
    scanf("%d , %d", &a, &b);
    printf("%d %d\n", a, b);
    return 0;
}
```



- Για να λειτουργήσει σωστά αυτό το πρόγραμμα πρέπει οι ακέραιες τιμές που θα εισάγει ο χρήστης να διαχωρίζονται μεταξύ τους με κόμμα (,)

Παρεμβολή απλών χαρακτήρων στη scanf() (II)

- Αν αντί για κόμμα (όπως είδαμε στο προηγούμενο παράδειγμα), μεταξύ των ειδικών χαρακτήρων μέσα στην scanf() χρησιμοποιούνται ο χαρακτήρας 'm', (όπως φαίνεται παρακάτω), θα έπρεπε να εισαχθεί ο χαρακτήρας 'm' μεταξύ των τιμών που θα έδινε ο χρήστης από το πληκτρολόγιο

```
#include <stdio.h>
int main()
{
    int a, b;
    scanf("%dm%d", &a, &b);
    printf("%d %d\n", a, b);
    return 0;
}
```

- Αν δηλαδή ο χρήστης επιθυμούσε να εισάγει τις τιμές 12 και 43, θα έπρεπε να πληκτρολογήσει: 12m43



Γενικά, προτείνουμε να μην παρεμβάλλεται κάποιος χαρακτήρας μεταξύ των προσδιοριστικών μετατροπής, έτσι ώστε να μην χρειάζεται κάποια επιπλέον εισαγωγή χαρακτήρων από τον χρήστη (σκεφτείτε επίσης ότι ο τελικός χρήστης του προγράμματος δεν είναι απαραίτητα και ο δημιουργός του προγράμματος)

Καθαρισμός Μνήμης

- Η συνάρτηση `scanf()` δεν θα λειτουργήσει σωστά, αν ο χρήστης δεν εισάγει τα δεδομένα σύμφωνα με την ακολουθία των προσδιοριστικών μετατροπής που ορίζονται σε αυτήν
- Στο επόμενο παράδειγμα η `scanf()` αναμένει μία ακέραια και μία πραγματική τιμή

```
#include <stdio.h>
int main()
{
    int i;
    double j;
    printf("Enter numbers: ");
    scanf("%d", &i);
    scanf("%lf", &j);
    printf("Num1 = %d, Num2 = %f\n", i, j);
    return 0;
}
```

- Τι θα συμβεί αν ο χρήστης εισαγάγει - έστω κατά λάθος - ως ακέραια τιμή την τιμή 5.65 ?

Παράδειγμα (I)

- Η συνάρτηση `scanf()` δεν διαβάζει τον χαρακτήρα νέας γραμμής που πληκτρολογεί ο χρήστης στο τέλος της εισαγωγής δεδομένων
- Αυτός ο χαρακτήρας θα διαβαστεί στην επόμενη κλήση της `scanf()`
- Αν όμως, στην επόμενη κλήση της, η `scanf()` χρησιμοποιείται για το διάβασμα χαρακτήρων, τότε θα διαβαστεί μόνο αυτός ο χαρακτήρας (της νέας γραμμής) και οι υπόλοιποι θα αγνοηθούν
- Π.χ. το επόμενο πρόγραμμα δεν θα εκτελεστεί σωστά

```
#include <stdio.h>
int main()
{
    char ch;
    int i;

    printf("Enter number: ");
    scanf("%d",&i);

    printf("Enter character: ");
    scanf("%c",&ch);

    printf("Int = %d and Char = %c\n",i,ch);
    return 0;
}
```

Παράδειγμα (II)

- Αν αντιστρέψουμε στο προηγούμενο παράδειγμα τη σειρά του διαβάσματος, τότε το πρόγραμμα θα εκτελεστεί σωστά, αφού – σύμφωνα με προηγούμενη παρατήρηση – ο χαρακτήρας νέας γραμμής που υπάρχει πριν από την αριθμητική τιμή αγνοείται

```
#include <stdio.h>
int main()
{
    char ch;
    int i;

    printf("Enter character: ");
    scanf("%c",&ch);

    printf("Enter number: ");
    scanf("%d",&i);

    printf("Int = %d and Char = %c\n",i,ch);
    return 0;
}
```

Τρόποι Καθαρισμού Μνήμης

- Ένας τρόπος για να αδειάσουμε τη μνήμη του πληκτρολογίου από τα δεδομένα που έχουν παραμείνει είναι με τη χρήση της συνάρτησης `getchar()` χρησιμοποιώντας τον παρακάτω επαναληπτικό βρόχο

```
while (getchar() != '\n');
```

- Ωστόσο, πολλοί προγραμματιστές χρησιμοποιούν τη συνάρτηση `fflush()`

```
fflush(stdin);
```



Προσέξτε όμως, σύμφωνα με το πρότυπο της C, η συμπεριφορά της `fflush()` όταν χρησιμοποιείται με όρισμα το `stdin` **είναι ακαθόριστη**