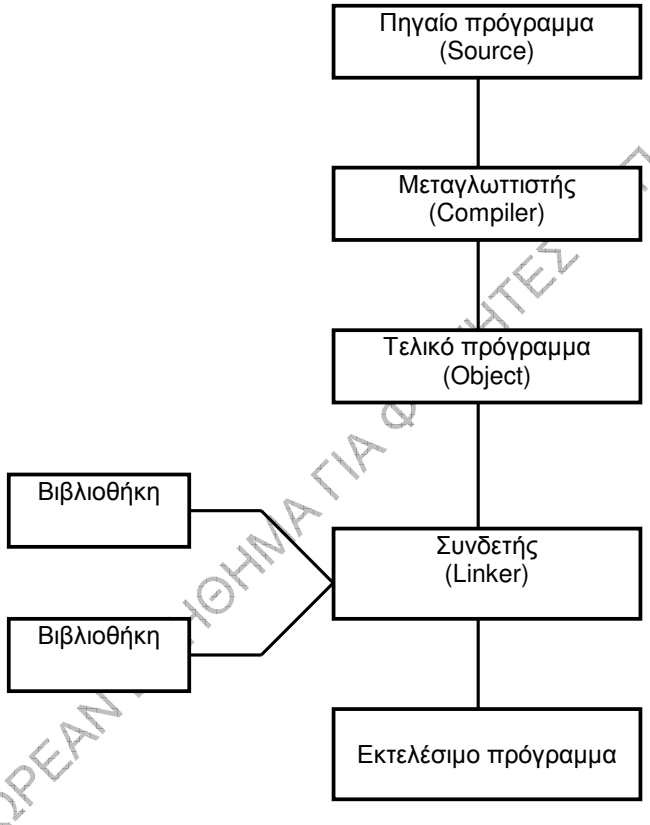
ΓΕΝΙΚΑ

- Πρόγραμμα
- Γλώσσες προγραμματισμού
- Μεταγλώττιση / compiler
- Βιβλιοθήκες / linker

ΒΗΜΑΤΑ ΔΗΜΙΟΥΡΓΙΑΣ ΠΡΟΓΡΑΜΜΑΤΟΣ



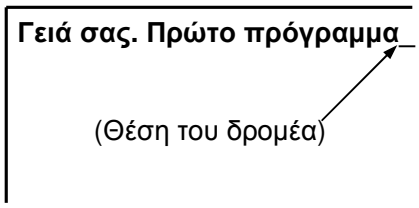
ΜΟΡΦΗ ΠΡΟΓΡΑΜΜΑΤΟΣ C

```
/* Πρώτο πρόγραμμα */

#include  <stdio.h>

main()
{
  printf(‘’ Γειά σας. Πρώτο πρόγραμμα ‘’);
}
```

- Η C είναι γλώσσα **ελευθέρου format**.
- **Κάνει διάκριση** μικρών/κεφαλαίων.
- **Σχόλια**: ό,τι βρίσκεται μεταξύ των /* …….. */ . Μπαίνουν οπουδήποτε στο πρόγραμμα.
- #include : **οδηγία** προς τον **προεπεξεργαστή**
- main() : **κύρια συνάρτηση**, υπάρχει πάντα
- Άγκιστρα { }
- **Συναρτήσεις**. Είναι αυτοτελή τμήματα κώδικα. Κάθε συνάρτηση έχει μια τιμή.
- Η printf **εμφανίζει στην οθόνη** ό,τι είναι ανάμεσα στα ‘’ ‘’. Το πιο πάνω πρόγραμμα θα γράψει δηλαδή στην οθόνη:



ΕΝΑ ΠΙΟ ΣΥΝΘΕΤΟ ΠΡΟΓΡΑΜΜΑ

```
#include  <stdio.h>

main()
{
  int num, art;

  num = 1;
  printf (‘’ Ο αριθμός ισούται με %d\n‘’, num);
  art = num + 1;
  printf (‘’ Προσθέτοντας 1 έχουμε %d\n’’,  art);
}
```

Σχόλια:

- num, art : **μεταβλητές.** Οι μεταβλητές παίζουν τον ρόλο που ξέρουμε από τα Μαθηματικά, «κρατάνε» δηλαδή τιμές. Τις χειριζόμαστε και τις χρησιμοποιούμε σαν να ήταν συγκεκριμένοι αριθμοί (θα δούμε βέβαια πιο κάτω ότι οι μεταβλητές δεν έχουν πάντα αριθμητική τιμή).
- Στη φάση της εκτέλεσης του προγράμματος, όταν περάσουμε από δήλωση μεταβλητής, ΟΠΟΥ κι αν συμβαίνει η δήλωση αυτή μέσα στο πρόγραμμα, γίνονται τα εξής:
  - **Δεσμεύεται χώρος στη μνήμη** για να κρατηθεί η τιμή της. Το πόσος χώρος (ποσα byte) δεσμεύεται εξαρτάται από το είδος της μεταβλητής.
  - Τίθεται ένας **δείκτης** (ένα ιδεατό βελάκι) που δείχνει τη θέση μνήμης, στην οποία έχει αποθηκευτεί η μεταβλητή.
- Οι εντολές του προγράμματος (όχι οι γραμμές) **χωρίζονται** με το **;**
- Οι εντολές **εκτελούνται** η μία μετά την άλλη με τη σειρά που τις συναντούμε.
- num = 1; : **εκχώρηση τιμής**. Ο τελεστής **= δεν είναι ο τελεστής ισότητας**. Σημαίνει: βρές την τιμή του δεξιού μέλους και μετά καταχώρησέ την στο αριστερό μέλος. Έτσι, αν το num είχε τιμή 18, τότε μετά τις εντολές:

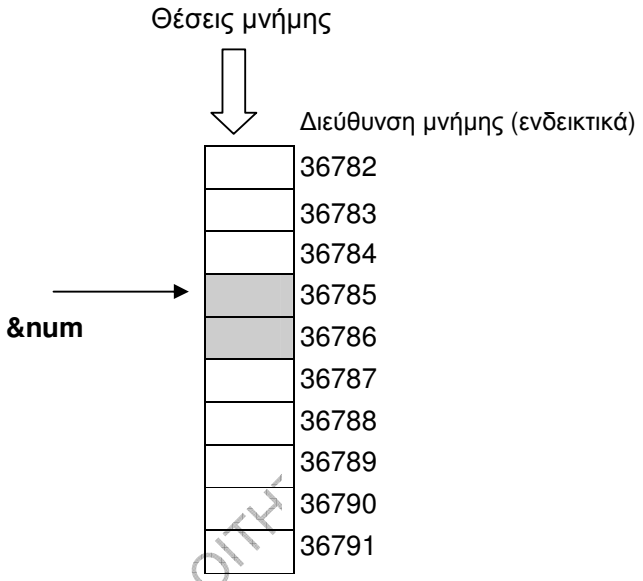
```
num = 52 + 3 * 6;
num = num + 1;
```

το num έχει τιμή 71.

Δυό εισαγωγικά λόγια για τους δείκτες:

Ο δείκτης δείχνει πάντα σε ένα byte στη μνήμη και συγκεκριμένα, στο πρώτο από τα byte που έχουν δεσμευτεί για την αποθήκευση της μεταβλητής.

Οι δείκτες έχουν ονόματα. Π.χ. ο δείκτης ο οποίος δείχνει εκεί που έχει αποθηκευτεί η μεταβλητή num, λέγεται **&num**. Το σκιασμένο τμήμα στο σχήμα δείχνει τον χώρο αποθήκευσης του num (2 byte).



Η ΣΥΝΑΡΤΗΣΗ printf()

(Μια πρώτη ματιά)

Εμφανίζει στην οθόνη ό,τι βρίσκεται ανάμεσα στα ‘’ ‘’.

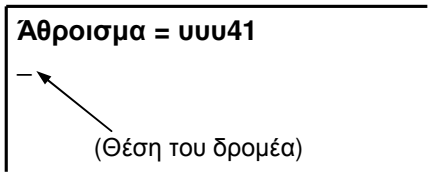
Π.χ.

printf (‘’ Άθροισμα = %5d\n ’’, num+art);

\n : **χαρακτήρας νέας γραμμής** (θεωρείται ένας χαρακτήρας στη C). Κατά την εκτέλεση του προγράμματος, όταν τον συναντήσουμε μέσα σε μια printf( ), έχει σαν αποτέλεσμα να κατεβεί ο δρομέας στην αρχή της αποκάτω γραμμής από αυτήν στην οποία βρίσκεται.

%d : **προσδιοριστής μορφής**. Δηλαδή πού και με ποια μορφή θα εμφανιστεί ένας ακέραιος. Π.χ. το %3d σημαίνει να εμφανιστεί ένας ακέραιος στην οθόνη σε χώρο 3 διαστημάτων.

Στην πιο πάνω εντολή, αν το num έχει τιμή 23 και το art έχει τιμή 18, στην οθόνη θα εμφανιστεί:

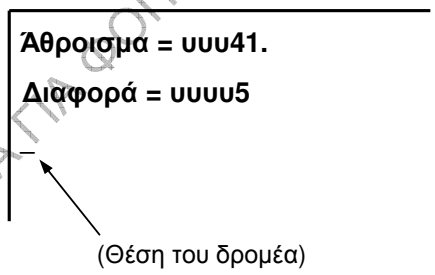


(Τα υυυ σημαίνουν τρία κενά διαστήματα).

Για τις ίδιες τιμές των num και art, η παρακάτω εντολή:

printf (‘’ Άθροισμα = %5d.\nΔιαφορά = %5d\n ’’, num+art, num-art);

έχει σαν αποτέλεσμα να εμφανιστεί στην οθόνη:



ΔΕΔΟΜΕΝΑ

Κύριοι τύποι δεδομένων που χρησιμοποιούνται στην C:

- Ακέραιοι (int)
- Μακρείς ακέραιοι (long)
- Μη προσημασμένοι ακέραιοι (unsigned)
- Χαρακτήρες (char)
- Κινητής υποδιαστολής (float)
- Διπλής ακρίβειας (double)

ΑΚΕΡΑΙΟΙ (int)

- Δεν έχουν κλασματικό μέρος.
- Χρειάζονται χώρο **2 byte** στη μνήμη ο καθένας.
- Περιοχή τιμών : **-32768 έως +32767.** Ο προγραμματιστής οφείλει να φροντίζει ώστε τόσο οι τιμές των μεταβλητών, όσο και τα αποτελέσματα των τυχόν πράξεων να μη παραβιάζουν τα όρια (τη μέγιστη και την ελάχιστη τιμή).
- Οι ακέραιες μεταβλητές δηλώνονται ως εξής:

```
int  num, a, atr;
```

- Το παρακάτω είναι επιτρεπτό:

```
int num = 20;
```

και αποτελεί απόδοση αρχικής τιμής στη μεταβλητή num.

- Αν το ak έχει τιμή 8652, στην παρακάτω printf( ):

```
printf (''%2d'', ak);
```

το 2 στο %2d αγνοείται και ο αριθμός εμφανίζεται σε όσο χώρο είναι αναγκαίος.

Άλλοι προσδιοριστές ακεραίων (%x και %o) :

```
printf (''%3d x %3d = %5x\n '', num, 3, 3*num);

printf (''  %3d + %3d = %5o\n '', num, 2, num+2);
```

Ο %x λέγεται **προσδιοριστής μορφής των ακεραίων του δεκαεξαδικού αριθμητικού συστήματος,** ενώ ο %o **λέγεται προσδιοριστής μορφής των ακεραίων του οκταδικού αριθμητικού συστήματος.** Αντίστοιχα, ο %d αποτελεί τον **προσδιοριστή μορφής των ακεραίων του δεκαδικού αριθμητικού συστήματος.** Δηλαδή το %x μέσα σε μια printf( ) έχει σαν αποτέλεσμα να εμφανιστεί στην οθόνη ένας ακέραιος εκφρασμένος στο δεκαεξαδικό, ενώ με το %o εμφανίζεται ένας ακέραιος εκφρασμένος στο οκταδικό. Αν το num έχει τιμή 20, οι δυό παραπάνω printf( ) θα εμφανίσουν στην οθόνη τα εξής:

```
υ20 x υυ3 = υυυ3C
υ20 + υυ2 = υυυ26
_
```

(Θέση του δρομέα)

ΚΙΝΗΤΗΣ ΥΠΟΔΙΑΣΤΟΛΗΣ ΚΑΙ ΔΙΠΛΗΣ ΑΚΡΙΒΕΙΑΣ
(float και double)

- Έχουν ακέραιο και κλασματικό μέρος.
- Χρειάζονται χώρο **4 byte** στη μνήμη οι float και **8 byte** οι double.
- Δηλώνονται :

```
float  pr, tel;
double art;
```

- Καταχώρηση τιμών:

```
pr = 9.356;
tel = 5.6e-5;
```

Στο πιο πάνω το tel παίρνει τιμή 5.6 **επί 10 εις την** -5.

Απεικόνιση στην οθόνη:

```
float  val = 397.86;
printf(" TIMH %f\n", val);
```

Ο **%f** αποτελεί τον **προσδιοριστή μορφής και των float και των double**.

Στην παρακάτω printf( ) το val απεικονίζεται σε χώρο 5 διαστημάτων, από τα οποία το ένα για το κλασματικό μέρος:

```
printf(" TIMH %5.1f\n", val);
```

Μετά την εκτέλεση της πιο πάνω printf( ), αλλά και των:

```
printf(" TIMH %8.2f\n", val);
printf(" TIMH %9.3f\n", val);
```

στην οθόνη θα εμφανιστούν τα παρακάτω:

```
TIMH 397.9
TIMH uu397.86
TIMH uu397.860
_
```

(Θέση του δρομέα)

Η printf( ) με προσδιοριστή %4.2f θα εμφανίσει 397.86. Δηλαδή **όταν το ακέραιο μέρος δεν χωράει στο πλήθος των διαστημάτων που καθορίζονται, τότε το ακέραιο μέρος καταλαμβάνει τον χώρο που απαιτείται.**

ΧΑΡΑΚΤΗΡΕΣ (char)

- Στην πραγματικότητα είναι ακέραιοι τύποι.
- Χρειάζονται χώρο **1 byte** ο καθένας στη μνήμη.
- Δηλώνονται:

char ch, grm;

Μια μεταβλητή τύπου χαρακτήρα παίρνει ως τιμή ένα από τους 256 χαρακτήρες που χρησιμοποιεί ο υπολογιστής (οι χαρακτήρες του πληκτρολογίου και όχι μόνο). Με τις εντολές που ακολουθούν, το ch παίρνει τιμή τον χαρακτήρα K, δηλαδή το σύμβολο K, ενώ το grm παίρνει τιμή ίση με θαυμαστικό:

ch = 'K';

grm = '!';

Ένας συγκεκριμένος χαρακτήρας στην C τίθεται μέσα σε μονά εισαγωγικά.

Αν καταγράψουμε σε ένα κατάλογο τους χαρακτήρες και αποδώσουμε σε καθένα ένα αύξοντα αριθμό, δημιουργείται μία αντιστοίχιση ένα προς ένα μεταξύ χαρακτήρων και αυξόντων αριθμών. Μια τέτοια αντιστοίχιση λέγεται **κώδικας**, ένας δε από τους πολλούς δυνατούς κώδικες αποτελεί τον **κώδικα ASCII**, τον οποίο χρησιμοποιεί ο υπολογιστής. Οι παρακάτω είναι πραγματικές τιμές του κώδικα ASCII:

<space>	20
……	…..
!	33
……	…..
#	35
……	…..
0	48
1	49
2	50
……	…..
A	65
B	66
C	67
D	68
……	…..
a	97
b	98
c	99
d	100
……	…..

Οι χαρακτήρες, όπως είναι φανερό, έχουν **διάταξη**, άρα ισχύει για παράδειγμα: 'A' < 'B'. Επίσης είναι ακέραιοι τύποι, αφού μπορούμε **ισοδύναμα** να μιλήσουμε για τον χαρακτήρα C ή για τον 67^{ο} χαρακτήρα του κώδικα ASCII. Έτσι, εν γένει **μπορούμε, όπου χρειάζεται χαρακτήρας να χρησιμοποιούμε ισοδύναμα τον αύξοντα αριθμό του στον ASCII**.

Μετά τις παρακάτω δηλώσεις και εντολές:

```
char br = 68;
char ch;
……..
ch = 'K';
printf('' ΧΑΡΑΚΤΗΡΕΣ %3c%3c\n '', br, ch);
……..
```

στην οθόνη θα εμφανιστούν τα παρακάτω:

```
ΧΑΡΑΚΤΗΡΕΣ υυDυυK
_
```

(Θέση του δρομέα)

Ο **%c** αποτελεί τον **προσδιοριστή μορφής των χαρακτήρων**. Η printf( ) εμφανίζει τον χαρακτήρα είτε ως ακέραια τιμή είτε ως χαρακτήρα, ανάλογα με τον προσδιοριστή που χρησιμοποιούμε, όπως στο παρακάτω:

```
char ch = 'C';
……..
printf('' Ο ch έχει τιμή %3c\n'', ch);
printf('' Στον ASCII ο ch είναι ο %5d\n '', ch);
……..
```

```
Ο ch έχει τιμή υυC
Στον ASCII, ο ch είναι ο υυυ67
_
```

(Θέση του δρομέα)

Ειδικοί χαρακτήρες:

Κάποιοι ειδικοί χαρακτήρες στη C είναι οι **ακολουθίες διαφυγής**, δύο χαρακτήρες δηλαδή, οι οιποίοι στη C θεωρούνται ένας:

b : Μετακίνηση του δρομέα μια θέση προς τα πίσω

n : Αλλαγή γραμμής

r : Μετακίνηση του δρομέα στην αρχή της γραμμής, στην οποία ήδη βρίσκεται

t : Μετακίνηση του δρομέα κατά TAB

Π.χ. :

α)

```
printf (''ΚΑΛΗΜΕΡΑ\b\b\b'');
printf (''ΚΑΛΗΜΕΡΑ\r'');
```

Στην πρώτη printf( ), ο δρομέας κάτω από το Ε

Στη δεύτερη printf( ), ο δρομέας κάτω από το Κ

β)

Με τις παρακάτω εντολές:

```
printf (''ΚΑΛΗΜΕΡΑ\b\b'');
printf (''ΓΙΑΝΝΗ\r'');
```

στην οθόνη θα γραφεί:

ΚΑΛΗΜΕΓΙΑΝΝΗ

και ο δρομέας θα σταματήσει κάτω από το Κ.

Η ΣΥΝΑΡΤΗΣΗ scanf( )

Διαβάζει την τιμή μιάς μεταβλητής από το πληκτρολόγιο.

Π.χ.

```
int   age;
float hgt;
……..
printf (''Δώστε το ύψος σας σε μέτρα ''
        '' και την ηλικία σας\n'');
scanf(''%f %d'', &hgt, &age);
printf (''Είστε %5d ετών. Ύψος %5.2f m\n'', age, hgt);
……..
```

Σχόλια:

- Η scanf ( ) χρησιμοποιεί **ίδιους προσδιοριστές με την printf ( )**.
- Αν θέλαμε να γράψουμε στην οθόνη τις age και hgt θα είχαμε:

  printf (‘%f%d’’, hgt, age);

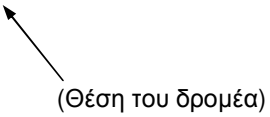
  και θα διαβάζαμε την εντολή ως εξής: «Γράψε στην οθόνη ένα ακέραιο, τον age και ένα float, τον hgt».
- Η scanf ( ) μετά τα “ “ έχει δείκτες:

  scanf(‘’%f %d’’, &hgt, &age);

  και λέει: «Διάβασε ένα float και **τοποθέτησέ τον** στη μνήμη **εκεί που δείχνει ο δείκτης &hgt** και ένα ακέραιο και **τοποθέτησέ τον** στη μνήμη **εκεί που δείχνει ο δείκτης &age**».

Όταν εκτελείται η scanf( ) το πρόγραμμα **περιμένει** να δοθούν οι τιμές που καθορίζονται από τους προσδιοριστές. Οι τιμές χωρίζονται μεταξύ τους με <space> ή με <Enter> και στο τέλος των τιμών δίνουμε <Enter>. Δηλαδή, αν δίναμε για την παραπάνω scanf( ) τις τιμές 1.85 και 25 αντίστοιχα, μετά την εκτέλεση του πιο πάνω προγράμματος, στην οθόνη θα είχαμε:

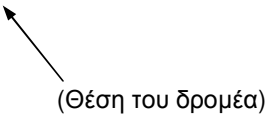
Δώστε το ύψος σας σε μέτρα και την ηλικία σας
1.85υ25
Είστε υυυ25 ετών. Ύψος υ1.85m
–



(Θέση του δρομέα)

ή το παρακάτω:

Δώστε το ύψος σας σε μέτρα και την ηλικία σας
1.85
25
Είστε υυυ25 ετών. Ύψος υ1.85m
–



(Θέση του δρομέα)

ΣΤΑΘΕΡΕΣ

- Είναι ονόματα, τα οποία **διατηρούν την ίδια τιμή σε όλη τη διάρκεια εκτέλεσης του προγράμματος**.
- Ορίζονται με **εντολές προς τον προεπεξεργαστή**.

  Π.χ.

  #define CNT 99.3

  Ο προεπεξεργαστής είναι μια ιδεατή «μηχανή» η οποία ενεργεί πάνω στο πρόγραμμα **πριν** την μεταγλώττιση, δηλαδή τα ονόματα των σταθερών αντικαθιστώνται με την τιμή τους πρίν την μεταγλώττιση
- Οι σταθερές γράφονται συνήθως με **κεφαλαία γράμματα**.
- Μετά την τιμή **δεν υπάρχει το ;**

Παράδειγμα:

Στο παράδειγμα που ακολουθεί δίδονται από το πληκτρολόγιο οι τιμές τεσσάρων μεταβλητών, των tel, dvd, pr και hd, οι οποίες αντιπροσωπεύουν τις τιμές μιάς τηλεόρασης, μιάς συσκευής DVD, ενός εκτυπωτή και ενός σκληρού δίσκου. Οι τιμές που δίνονται από το πληκτρολόγιο είναι χωρίς ΦΠΑ. Το πρόγραμμα υπολογίζει και γράφει στην οθόνη τις τιμές με ΦΠΑ (θεωρούμε τον ΦΠΑ ίσο με 19%).

```
#include <stdio.h>
main()
{
    float tel, dvd, pr, hd;

    printf (''\nΤιμή τηλεόρασης\n'');
    scanf (''%f'', &tel);
    printf (''Τιμή DVD\n'');
    scanf (''%f'', &dvd);
    printf (''Τιμή εκτυπωτή\n'');
    scanf (''%f'', &pr);
    printf (''Τιμή σκληρού δίσκου\n'');
    scanf (''%f'', &hd);
    tel = tel + 0.19 * tel;
    dvd = dvd + 0.19 * dvd;
    pr = pr + 0.19 * pr;
    hd = hd + 0.19 * hd;
    printf (''\n\n'');
    printf (''Τιμές με ΦΠΑ κατά σειρά\n'');
    printf (''%f %f %f %f \n'', tel, dvd, pr, hd);
}
```


Για οποιαδήποτε αλλαγή στο ΦΠΑ πρέπει να αλλάξουμε παντού μέσα στο πρόγραμμα το 0.19. Το παραπάνω παράδειγμα με την χρήση μιας σταθεράς της FPA μπορεί να γραφεί όπως παρακάτω.

```
#include <stdio.h>
#define FPA 0.19
main()
{
    float tel, dvd, pr, hd;

    printf (''\nΤιμή τηλεόρασης\n'');
    scanf (''%f'', &tel);
    printf (''Τιμή DVD\n'');
    scanf (''%f'', &dvd);
    printf (''Τιμή εκτυπωτή\n'');
    scanf (''%f'', &pr);
    printf (''Τιμή σκληρού δίσκου\n'');
    scanf (''%f'', &hd);
    tel = tel + FPA * tel;
    dvd = dvd + FPA * dvd;
    pr = pr + FPA * pr;
    hd = hd + FPA * hd;
    printf (''\n\n'');
    printf (''Τιμές με ΦΠΑ κατά σειρά\n'');
    printf (''%f %f %f %f \n'', tel, dvd, pr, hd);
}
```


Τώρα, για οποιαδήποτε αλλαγή στο ΦΠΑ αρκεί να αλλάξουμε την τιμή της σταθεράς FPA και να μεταγλωττίσουμε ξανά το πρόγραμμα, οπότε οι αλλαγές γίνονται παντού όπου υπάρχει το FPA.

ΠΙΝΑΚΕΣ

Πίνακας είναι **συλλογή στοιχείων κάποιου τύπου** (του ίδιου τύπου όλα, δηλαδή όλα ακέραιοι ή όλα float ή όλα χαρακτήρες κλπ), τα οποία **βρίσκονται αποθηκευμένα σε διαδοχικές (συνεχόμενες) θέσεις μνήμης**.

Δήλωση πίνακα:

```
char name[40];
```

Με την δήλωση αυτή λέμε στο πρόγραμμά μας (άρα στον υπολογιστή και το λειτουργικό μας σύστημα): «Πήγαινε στη μνήμη, βρές 40 συνεχόμενες ελεύθερες θέσεις, δέσμευσέ τις για μας, πές τις και τις 40 με το όνομα name και σε κάθε θέση να ξέρεις ότι θα μπεί ένας χαρακτήρας».

Ο αριθμός μέσα στις αγκύλες δεν σημαίνει byte, αλλά θέσεις μνήμης, δηλαδή πόσους χαρακτήρες θα αποθηκεύσουμε στον πίνακα. Σε κάθε θέση θα αποθηκευτεί ένας χαρακτήρας ή για να είμαστε πιο ακριβείς μια μεταβλητή τύπου χαρακτήρα. Δεδομένου ότι κάθε χαρακτήρας χρειάζεται χώρο 1 byte, ο πίνακας καταλαμβάνει 40 byte στην μνήμη. Δηλαδή ο πίνακας name είναι μια «συλλογή» από 40 μεταβλητές τύπου χαρακτήρα, οι οποίες δεν βρίσκονται όμως «σκόρπιες» στη μνήμη, αλλά αποτελούν ένα **διατεταγμένο σύνολο**, μια **δομή δεδομένων**.

Στα παρακάτω δηλώνουμε ένα πίνακα ακεραίων και ένα πίνακα float:

```
int   arf[100];
float pro[89];
```

Σύμφωνα με όσα είπαμε ήδη, **ο arf είναι πίνακας ακεραίων 100 θέσεων, δηλαδή 100 ακεραίων μεταβλητών.** Αντίστοιχα, **ο pro είναι πίνακας float 89 θέσεων, δηλαδή 89 float μεταβλητών**. Δεδομένου ότι κάθε ακέραιος χρειάζεται 2 byte στη μνήμη και κάθε float χρειάζεται 4 byte, **ο πίνακας arf καταλαμβάνει 200 byte στη μνήμη, ενώ ο pro καταλαμβάνει 356 byte**.

Κατά τα γνωστά, μια δήλωση συνοδεύεται και από τη «δημιουργία» ενός δείκτη στη μνήμη. **Ο δείκτης, ο οποίος δημιουργείται ταυτόχρονα με την δημιουργία του πίνακα, δείχνει στην πρώτη θέση του πίνακα και έχει το ίδιο όνομα με το όνομα του πίνακα**. Δηλαδή ο δείκτης στην πρώτη θέση του name λέγεται και αυτός name και όχι &name, ενώ ο δείκτης στην πρώτη θέση του arf λέγεται και αυτός arf και όχι &arf.

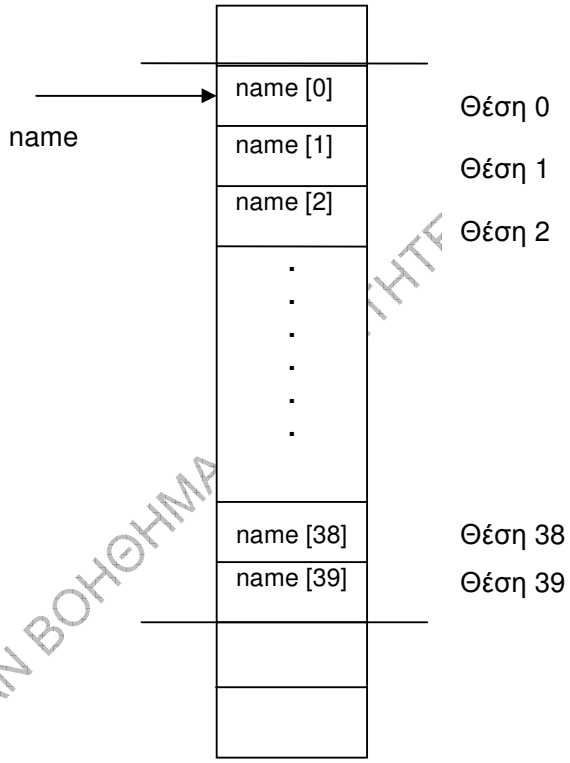
Οι θέσεις κάθε πίνακα στην C αριθμούνται, ξεκινώντας από το μηδέν. Έτσι, η θέση 0 είναι η πρώτη θέση του πίνακα name, ενώ η θέση 39 είναι η τελευταία. Αντίστοιχα, η θέση 0 είναι η πρώτη θέση του πίνακα arf, ενώ η θέση 99 είναι η τελευταία. Όπως αναφέρθηκε, σε κάθε θέση του πίνακα name υπάρχει ένας χαρακτήρας. **Ο χαρακτήρας που υπάρχει στην θέση 0 λέγεται name[0]**, **αυτός που υπάρχει στην θέση 8 λέγεται name[8], ενώ ο τελευταίος χαρακτήρας του name λέγεται name[39].** Αντίστοιχα, **ο πρώτος ακέραιος του πίνακα arf λέγεται arf[0], αυτός που υπάρχει στην θέση 32 λέγεται arf[32], ενώ ο τελευταίος χαρακτήρας του arf λέγεται arf[99]**. Άρα είναι σωστές οι εντολές:

name [5] = 'F';

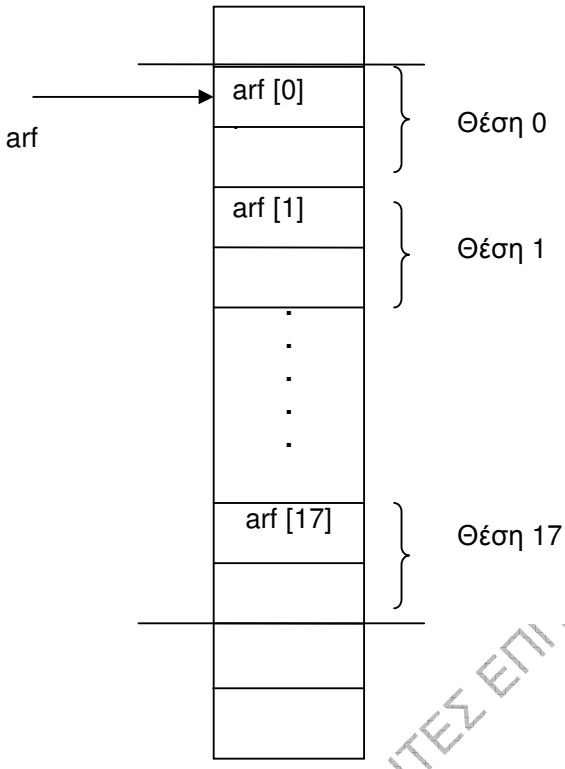
arf [18] = -5;

Σύμφωνα με αυτές, το στοιχείο 5 του name (το 6^{ο} στη σειρά, αφού η αρίθμηση έχει αρχίσει από το μηδέν) γίνεται ίσο με τον χαρακτήρα F, ενώ το το στοιχείο 18 του arf (το 19^{ο} στη σειρά) γίνεται ίσο με -5. Δηλαδή το όνομα ενός πίνακα με αγκύλες και ένα αριθμό μέσα σε αυτές μπορούμε να το συναντήσουμε σε δύο σημεία: στις δηλώσεις, όπου και δημιουργείται ο πίνακας, αλλά και σε κάποια εντολή του προγράμματος. Στην δεύτερη αυτή περίπτωση δεν δημιουργείται πίνακας, απλώς μιλάμε για το συγκεκριμένο στοιχείο του πίνακα.

Σχηματικά για τον πίνακα name:



Αντίστοιχα, για τον πίνακα arf θα είχαμε:



ΣΥΜΒΟΛΟΣΕΙΡΕΣ

Είναι **σειρές από χαρακτήρες**. Για παράδειγμα, μια σταθερή, μια συγκεκριμένη συμβολοσειρά είναι η:

"ΕΞΕΤΑΣΤΙΚΗ ΠΕΡΙΟΔΟΣ"

Τα **εισαγωγικά** δεν αποτελούν μέρος της συμβολοσειράς, αλλά απλώς καθορίζουν τα όριά της (όπως αντίστοιχα τα μονά εισαγωγικά, όταν μιλούμε π.χ. για τον χαρακτήρα 'Α' δεν αποτελούν μέρος του χαρακτήρα).

Οι συμβολοσειρές αποθηκεύονται **σε πίνακες τύπου char** (όπως αντίστοιχα ένας χαρακτήρας αποθηκεύεται σε μια μεταβλητή τύπου χαρακτήρα). Με την scanf( ) στο πρόγραμμα που ακολουθεί διαβάζεται από το πληκτρολόγιο μια συμβολοσειρά, δηλαδή μια ομάδα χαρακτήρων:

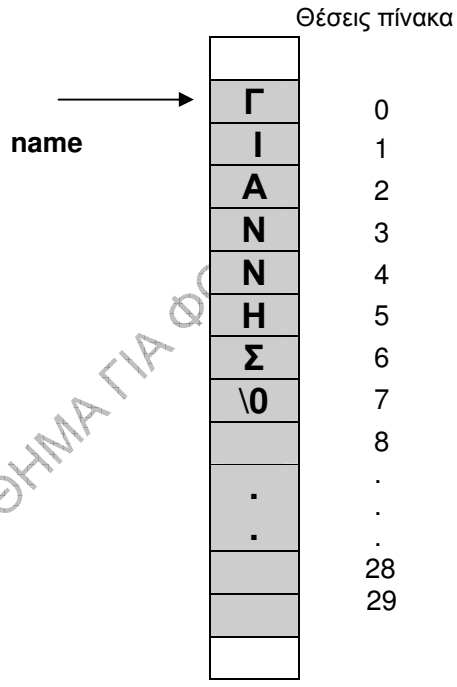
```
main()
{
    char name[30];
    ……….
    printf (''Πώς σε λένε;\n'');
    scanf(''%s'', name);
    printf (''Γειά σου %s\n'', name);
    ……….
}
```

Σχόλια:

- Η scanf( ) χρησιμοποιεί τον **προσδιοριστή μορφής %s**, ο οποίος σημαίνει ότι η scanf περιμένει να διαβάσει μια συμβολοσειρά, δηλαδή μια ομάδα χαρακτήρων. Η scanf θα περιμένει μέχρι να πληκτρολογήσουμε τους χαρακτήρες και να δώσουμε Enter.
- Η scanf( ) διαβάζει μόνο **απλές λέξεις** και **όχι φράσεις,** δηλαδή **όχι ομάδα χαρακτήρων μέσα στην οποία υπάρχει το κενό** (space).
- Η scanf( ) μετά τα διπλά εισαγωγικά χρειάζεται δείκτη. **Εδώ το name έχει τον ρόλο δείκτη**. Θυμίζουμε ότι ο πίνακας και ο δείκτης στο πρώτο στοιχείο του έχουν το ίδιο όνομα. Λέμε δηλαδή στην scanf( ): «Διάβασε μια συμβολοσειρά και τοποθέτησέ την στην μνήμη εκεί που δείχνει ο δείκτης name».
- Στον πίνακα τοποθετείται **ένας χαρακτήρας σε κάθε θέση του**. **Κάτω από τον τελευταίο χαρακτήρα τοποθετείται** «από μόνος του» (όχι δηλαδή από τον χρήστη), **ένας ειδικός χαρακτήρας, ο χαρακτήρας '\0'** (πιο σωστά ο χαρακτήρας '\x0', ο οποίος είναι ο μηδενικός, ο πρώτος στη σειρά χαρακτήρας του κώδικα ASCII). Έστω ότι δινουμε από το πληκτρολόγιο την συμβολοσειρά ΓΙΑΝΝΗΣ. Τότε, στη μνήμη έχουμε την εικόνα που ακολουθεί. **Το σκιασμένο τμήμα παριστάνει τον πίνακα χαρακτήρων name, ενώ η συμβολοσειρά name είναι το μέρος εκείνο του πίνακα μέχρι και το '\0'**, δηλαδή από τη θέση 0 έως και τη θέση 7 του πίνακα. Οι θέσεις του πίνακα μετά την 7 περιέχουν τους τυχόντες χαρακτήρες που υπήρχαν στον πίνακα κατά την δημιουργία του.
- **Ο χαρακτήρας '\0' χρειάζεται για να σηματοδοτήσει το τέλος της συμβολοσειράς.** Η δεύτερη printf( ) στις πιο πάνω εντολές χρησιμοποιεί και αυτή τον προσδιοριστή μορφής %s. **Η printf( ) γράφει στην οθόνη τα περιεχόμενα του πίνακα name από την αρχή μέχρι να συναντήσει το '\0'**, οπότε και σταματάει. **Αν σβήσουμε το '\0' από τον πίνακα** (για παράδειγμα βάζοντας στη

θέση του τον χαρακτήρα 'Β' με την εντολή **name[7] = 'B';**), **τότε η printf( ) γράφει από την αρχή μέχρι να συναντήσει κάπου μέσα στις επόμενες θέσεις του πίνακα ή στις επόμενες θέσεις της μνήμης το '\0'** και να σταματήσει. Το ότι έχουμε δηλώσει τον πίνακα name να έχει 30 θέσεις, δεν έχει σε τίποτα να κάνει με την printf( ). Η δέσμευση μνήμης έχει ως μόνο σκοπό κανένα άλλο πρόγραμμα που «τρέχει» παράλληλα με το δικό μας να μη γράφει στον δεσμευμένο χώρο και χαλάσει έτσι τα δεδομένα μας. Η printf( ) δεν «ξέρει» τίποτα γι' αυτή τη δέσμευση μνήμης και ούτε την «αφορά» η δέσμευση αυτή.

- Είναι προφανές ότι για να μην έχουμε προβλήματα, **κατά την εκτέλεση της scanf( ) πρέπει να δοθεί συμβολοσειρά η οποία να έχει το πολύ 29 χαρακτήρες,** άρα 30 μαζί με το '\0'.



- **Αν κατά την εκτέλεση της scanf( ) δοθεί συμβολοσειρά με περισσότερους από 29 χαρακτήρες,** τότε, εάν ο χώρος μετά τις 30 θέσεις είναι δεσμευμένος από το δικό μας ή άλλο πρόγραμμα, κατά την εκτέλεση θα πάρουμε μήνυμα λάθους. Εάν ο χώρος είναι ελεύθερος, τότε οι επιπλέον χαρακτήρες που δώσαμε θα αποθηκευτούν εκεί. Επειδή όμως ο επιπλέον αυτός χώρος δεν είναι δεσμευμένος, άρα θεωρείται ελεύθερος, μπορεί ανά πάσα στιγμή να χαθούν τα δεδομένα μας και το πρόγραμμά μας να «κρεμάσει». Η μόνη σωστή ενέργεια όταν

διαπιστώσουμε ότι ο δεσμευμένος χώρος δεν μας επαρκεί είναι να διορθώσουμε το πρόγραμμα, δηλώνοντας μεγαλύτερο μέγεθος πίνακα.

ΜΗΚΟΣ ΣΥΜΒΟΛΟΣΕΙΡΑΣ – ΤΙΜΗ ΕΠΙΣΤΡΟΦΗΣ ΣΥΝΑΡΤΗΣΗΣ – ΚΑΠΟΙΕΣ ΗΔΗ ΓΝΩΣΤΕΣ ΣΥΝΑΡΤΗΣΕΙΣ

Η C παρέχει τη δυνατότητα να μετρήσουμε το μήκος μιας συμβολοσειράς με την χρήση μιας «εντολής». Λέγοντας **«μήκος συμβολοσειράς»** εννοούμε **το πλήθος των χαρακτήρων της συμβολοσειράς** (η οποία είναι αποθηκευμένη σε ένα πίνακα τύπου char) **από την αρχή έως το '\0', χωρίς το '\0'**. Προκειμένου να γίνει αυτό, η C ελέγχει εάν ο 1ος χαρακτήρας του πίνακα είναι το '\0', εάν είναι ο 2ος κλπ, μέχρι τελικά να τον βρεί. Για να ολοκληρωθεί αυτή η διαδιακασία **εκτελείται ένα πλήθος από εντολές** και όχι μόνο μία, δηλαδή ένα ολόκληρο μικρό πρόγραμμα ή όπως λέγεται στην C **μια συνάρτηση**.

Κάθε συνάρτηση στην C έχει όνομα. Αυτή που μετράει τους χαρακτήρες μιάς συμβολοσειράς λέγεται **strlen( )**. **Η strlen( ) δέχεται ως όρισμα το όνομα της συμβολοσειράς, της οποίας αναζητεί το μήκος**. Αυτό σημαίνει ότι το όνομα της συμβολοσειράς βρίσκεται μέσα στις παρανθέσεις που ακολουθούν την λέξη strlen κατά την κλήση της. Αργότερα θα διατυπώσουμε αυστηρότερα το παραπάνω, αλλά προς το παρόν καλύπτει τις ανάγκες μας.

Επιπλέον, **κάθε συνάρτηση στην C όταν τελειώσει την δουλειά της γίνεται ίση με κάτι**, δηλαδή ένα ακέραιο, ένα float, ένα χαρακτήρα κλπ. Αυτό με το οποίο γίνεται ίση μια συνάρτηση λέγεται **«τιμή επιστροφής της συνάρτησης»**. **Η συνάρτηση strlen( ) έχει τιμή επιστροφής ένα ακέραιο** και συγκεκριμένα το μήκος της συμβολοσειράς, την οποία έχουμε δώσει ως όρισμα, δηλαδή **το πλήθος των χαρακτήρων της συμβολοσειράς** από την αρχή έως το '\0' , **χωρίς να υπολογίζεται το'\0'**.

Προκειμένου να γνωρίζει το πρόγραμμά σας πώς η strlen( ) θα δουλέψει σωστά, πρέπει να δώσουμε οδηγίες ως προς το πού υπάρχουν στοιχεία για την συνάρτηση αυτή. Αυτά τα στοιχεία βρίσκονται στο αρχείο string.h, το οποίο πρέπει να περιλάβουμε στο πρόγραμμά μας με την εντολή προς τον προεπεξεργαστή:

```
#include <string.h>
```

Στο αρχείο αυτό βρίσκονται πληροφορίες για όλες τις συναρτήσεις χειρισμού συμβολοσειρών.

Άλλες ήδη γνωστές συναρτήσεις:

Όπως η strlen( ) δεν είναι μια απλή εντολή, έτσι και άλλες από αυτές που μέχρι τώρα ονομάζαμε «εντολές» είναι ολόκληρα μικρά προγράμματα, δηλαδή συναρτήσεις. Για παράδειγμα **η printf( ) είναι μια συνάρτηση** που λέει τι αλλά και πώς θα γραφεί στην οθόνη, ενώ **η scanf( ) είναι μια συνάρτηση** που λέει τι αλλά και πώς θα διαβαστεί από το πληκτρολόγιο.

Όπως για την strlen( ) χρειάζεται να καθορίσουμε πού θα την βρούμε, πού δηλαδή θα βρούμε πληροφορίες γι' αυτήν, έτσι και για τις printf( ) και scanf( ) βρίσκουμε πληροφορίες στο αρχείο stdio.h, το οποίο έχουμε συνηθίσει μέχρι τώρα να το περιλαμβάνουμε πάντα και χωρίς να έχουμε αναλύσει το γιατί, σε ένα πρόγραμμα της C.