Δομές και Ενώσεις

- Η γλώσσα C επιτρέπει να δημιουργήσουμε καινούργιους τύπους δεδομένων με δύο τρόπους:
- πρώτον, ενώνοντας πολλές μεταβλητές σε μία συμπυκνωμένη μεταβλητή που λέγεται δομή (Structure) και
- δεύτερον, χρησιμοποιώντας την ένωση (union) για να μοιρασθούν πολλές μεταβλητές την ίδια περιοχή της μνήμης.

Δομές (Structures)

- Λέμε *δομή* τη συλλογή πολλών μεταβλητών οι οποίες θα αναφέρονται με το ίδιο όνομα και δεν θα αποτελούν συνεχόμενες θέσεις μνήμης, όπως οι πίνακες.
- Ο ορισμός μιας δομής αποτελεί ένα πρότυπο, το οποίο μπορεί να χρησιμοποιηθεί προκειμένου να δημιουργηθούν κάποιες μεταβλητές που να ανήκουν στη δομή.
- Κάθε δομή αποτελείται από μία ή περισσότερες μεταβλητές οι οποίες είναι λογικά συνδεδεμένες μεταξύ τους. Αυτές οι μεταβλητές λέγονται στοιχεία της δομής.

Ορισμός

Ο γενικός τύπος ορισμού μιας δομής είναι :

```
struct  όνομα-δομής
{
   τύπος-όνομα μεταβλητής;
   τύπος-όνομα μεταβλητής;
   τύπος-όνομα μεταβλητής;
    .
    .
    .
} μεταβλητές-δομής;
```

Παράδειγμα

Το όνομα και η διεύθυνση σε έναν κατάλογο είναι μία συνηθισμένη ομάδα συγγενικών πληροφοριών.

Η δομή η οποία περιέχει το όνομα, τη διεύθυνση, την οδό, την πόλη και τον κωδικό της πόλης, μπορεί να γραφτεί :

```
struct address
  {
      char name[30];
      char street[40];
      char city[20];
      unsigned int zip;
  };
```

Με τον τρόπο αυτό, δεν έχει δηλωθεί καμία μεταβλητή.

Μόνο ο τύπος των δεδομένων έχει ορισθεί.

Για να δηλώσουμε μια συγκεκριμένη μεταβλητή η οποία θα ανήκει σ' αυτή τη δομή (address) θα πρέπει να γράψουμε :

```
struct address info1;
```

Όταν ορίζουμε μια δομή, στην ουσία δηλώνουμε ένα σύνθετο τύπο μεταβλητής που αποτελείται από τα στοιχεία της δομής.

Επεξεργασία των στοιχείων μιας δομής

Ο γενικός τύπος αναφοράς σ' ένα στοιχείο μιας μεταβλητής κάποιας δομής είναι:

όνομα-μεταβλητής-δομής . όνομα-στοιχείου-δομής

Η τελεία συνήθως ονομάζεται τελεστής τελεία (dot operator) και σημαίνει ότι ακολουθεί ένα στοιχείο της μεταβλητής που αναφέρεται πριν την τελεία.

Π.χ. γράφουμε :

info1.zip = 12345;

Αν θέλουμε να προσεγγίσουμε κάθε ένα από τα στοιχεία του πίνακα name ξεχωριστά, θα μπορούσαμε να χρησιμοποιήσουμε μια εντολή ανακύκλωσης.

Για παράδειγμα, θα μπορούσαμε να συμπληρώσουμε το περιεχόμενο του πίνακα name χρησιμοποιώντας τις εντολές:

```
int t;
for(t=0; info1.name[t]; ++t )
putchar(info1.name[t]);
```

Πίνακες δομών

Προκειμένου να δηλώσουμε έναν πίνακα δομών πρέπει πρώτα να ορίσουμε μία δομή και μετά να δηλώσουμε μία μεταβλητή πίνακα, αυτού του τύπου.

Για παράδειγμα,

αν θέλουμε να δηλώσουμε ένα πίνακα δομών με 300 στοιχεία του τύπου address,

θα πρέπει να γράψουμε:

```
struct address info[300];
```

<u>Υπενθύμιση</u>. Όπως όλες οι μεταβλητές που είναι πίνακες έτσι και οι πίνακες δομών αρχίζουν την αρίθμηση από το 0.

Παράδειγμα

Αν θέλουμε να δημιουργήσουμε ένα πίνακα δομών για να αποθηκεύσει και να επεξεργαστεί τις πληροφορίες των πελατών-συνδρομητών μιας εταιρείας, μπορούμε να δημιουργήσουμε μια δομή δεδομένων, (client), η οποία θα κρατήσει αυτές τις πληροφορίες :

```
struct client
{
    char name[30];
    char street[40];
    char city[20];
    char state[3];
    unsigned  int zip;
} info[300];
```

```
main( )              // Κυρίως πρόγραμμα (Μενού επιλογών)
{
    char s[80], choice;
    init_list( );
    do {
       choice=menu_select( );
        switch(choice)
      {
        case 1: enter( );
                 break;
        case 2: delete( );
                 break;
        case 3:  list( );
                 break;
        case 4: exit(0);
       }
    } while(1);
}
```

```
menu_select( )
{
      char s[80];
      int c;
      printf("1. Enter a name\n");
      printf("2. Delete a name\n");
      printf("3. List the file\n");
      printf("4. Quit\n");
      do {
            printf("\nEnter your choice: ");
            gets(s);
            c=atoi(s);
          } while(c<1 ¦ ¦ c>4)
       return c;
 }
```

```
 enter( )
{
      int slot;
      slot = find_free( );
      if(slot = = -1)
   {
            printf("\n list full");
            return;
    }
      printf("enter name ");
      gets(info[slot].name);
      printf("enter street:");
      gets(info[slot].street);
      printf("enter city: ");
      gets(info[slot].city);
      printf("enter state: ");
      gets(info [slot].state);
      printf("enter zip: ");
      scanf("%d",&info[slot].zip);
}
```

```
find_free( )
{
 int t;
for(t=0; info[t].name[0] &&  t<300; ++t)
      if(t = = 300) return -1;
return t;
}
```

```
delete( )
{
        int slot;
        char s[80];
        printf("enter record #: ");
         gets( );
         slot = atoi(s);
         if(slot>0 && slot < 300)
           info[slot].name[0]='\0';
}
```

```
list( )
{
    int t;
   for(t=0; t<300; ++t)
  {
    if (info[t].name[0])
          {
            printf("%s\n",info[t].name);
            printf("%s\n",info[t].street);
            printf("%s\n",info[t].city);
            printf("%s\n",info [t].state);
            printf( "%u\n",info[t].zip) ;
          }
  }
       printf("\n");
}
```

Μεταφορά στοιχείων δομής σε συνάρτηση

Όταν περνά ένα στοιχείο μίας δομής σε μία συνάρτηση, στην πραγματικότητα περνά η τιμή αυτού του στοιχείου στη συνάρτηση, όπως ακριβώς περνά μία απλή μεταβλητή.

Αν είχαμε τη δήλωση:

```
struct test
{
  int x;
  char s[15];
} image;
```

Τότε η κλήση : `function1(image.x);`

σημαίνει ότι περνά στη συνάρτηση function1 η τιμή του στοιχείου image.x της μεταβλητής image

ενώ η κλήση :

```
function2(image.s[2]);
```

σημαίνει ότι περνά στη συνάρτηση function2 η τιμή της τρίτης κατά σειρά θέσης του στοιχείου image.s, δηλαδή το τρίτο στοιχείο του πίνακα s.

Αν θέλουμε να περάσουμε τη διεύθυνση μεμονωμένων στοιχείων της δομής ξεχωριστά σε μια συνάρτηση, θα πρέπει να χρησιμοποιήσουμε τον τελεστή &, μπροστά από το όνομα της δομής.

Προκειμένου να περάσουμε τη διεύθυνση των δύο στοιχείων της μεταβλητής image, θα πρέπει να γράψουμε:

```
function1(&image.x);
function2(&image.s[2]);
```

Μεταφορά ολόκληρης δομής σε συνάρτηση

Όταν μία δομή περνά σε μία συνάρτηση, μόνο η διεύθυνση του πρώτου χαρακτήρα (πρώτο byte) της δομής μεταφέρεται στη συνάρτηση.

Αυτός ο τρόπος είναι παρόμοιος με τον τρόπο που περνούν οι πίνακες σε συναρτήσεις.

Π.χ. Έστω η μεταβλητή t της δομής xronos

```
struct xronos
{
  int hours;
  int minutes;
  int seconds;
} t;
```

Όταν η δομή αυτή μεταφερθεί σε μια συνάρτηση, θα πρέπει να χρησιμοποιείται με τον ακόλουθο τρόπο :

```
display(struct xronos *t ){
printf("%d:", (*t).hours);
printf("%d:", (*t).minutes);
printf("%d\n", (*t).seconds);
}
```

Σημείωση. Οι παρενθέσεις γύρω από το *t είναι απαραίτητες επειδή ο τελεστής τελεία έχει υψηλότερη προτεραιότητα απ' ό,τι ο τελεστής *

Τελεστής βέλος

Επειδή οι αναφορές σε μια δομή που έχει περάσει σε μία συνάρτηση συμβαίνουν συχνά, γι αυτό η γλώσσα C έχει ένα ειδικό τελεστή, που λέγεται «τελεστής βέλος» και γράφεται - >

0 τελεστής -> χρησιμοποιείται εναλλακτικά στη θέση του τελεστή-τελεία όταν προσεγγίζουμε ένα στοιχείο δομής μέσα σε μία συνάρτηση.

Για παράδειγμα, η γραφή :

(*t).hours

είναι το ίδιο με τη γραφή :

t->hours

Πίνακες και Δομές

Κάθε στοιχείο της δομής μπορεί να είναι απλό ή σύνθετο.

Μια δομή μπορεί να είναι στοιχείο μιας άλλης δομής, όπως στο ακόλουθο παράδειγμα:

```
struct boy
{
  struct address newaddress[2];
  char ch;
} tom;
```

Π.χ. μπορούμε να γράψουμε :

```
tom.newaddress[1].zip = 67100;
```

Δυαδικά πεδία (πεδία Bits)

Σαν μια γνήσια γλώσσα μέσου επιπέδου η γλώσσα C, μπορεί να προσεγγίζει εύκολα **μεμονωμένα bits** μέσα σε ένα **byte**.

Αυτό μπορεί να αποδειχθεί χρήσιμο:

- Όταν έχουμε μεγάλο όγκο δεδομένων και ο αποθηκευτικός χώρος είναι περιορισμένος,
- Ορισμένες συσκευές επικοινωνιών (modem, scanner κτλ.) μεταφέρουν τις πληροφορίες κωδικοποιημένες σε μια διαδοχική σειρά από bits αποθηκευμένα μέσα σε ένα byte.
- Αντί να χρησιμοποιήσουμε τους τελεστές bitwise, η χρησιμοποίηση ενός πεδίου από bits μπορεί να προσθέσει περισσότερη ευελιξία και ταχύτητα στο πρόγραμμα.

Παράδειγμα

Μπορούμε να ορίσουμε τέσσερα μέλη μιας δομής και το κάθε ένα να έχει μέγεθος ενός bit με τη δομή:

```
struct newdevice
{
    unsigned active : 1;
    unsigned ready : 1;
    unsigned error : 1;
    unsigned  stop  : 1;
} code;
```

ΑΣΚΗΣΕΙΣ

1. Ορίστε μία δομή η οποία θα μπορεί ν' αποθηκεύει τις παρακάτω πληροφορίες:

όνομα ομάδας

όνομα παίκτη

ηλικία

αριθμός φανέλας

Απάντηση

```
struct player
{
    char player_team[20];
    char player_ name[30];
    int player_ age;
    int player_ number;
};
```

ΑΣΚΗΣΕΙΣ

2. **Χρησιμοποιώντας τη δομή της προηγούμενης άσκησης, δηλώστε ένα πίνακα δομών για τις ανάγκες ενός προβλήματος όπου θα επεξεργαστούμε 4 παίκτες μιας ομάδας.**

Απάντηση :

. **struct player group[4];**

ΑΣΚΗΣΕΙΣ

3. Να γράψετε μία συνάρτηση η οποία με τη βοήθεια ερωτήσεων στην οθόνη να ζητά να πληκτρολογηθούν:

όνομα ομάδας

όνομα παίκτη

ηλικία

αριθμός φανέλας

χρησιμοποιώντας τον πίνακα δομών της προηγούμενης άσκησης.

Η συνάρτησή να έχει μια παράμετρο η οποία θα είναι ο δείκτης ενός στοιχείου του πίνακα.

ΑΣΚΗΣΕΙΣ

4. Να γράψετε και ένα πλήρες πρόγραμμα το οποίο με τη βοήθεια των ασκήσεων 1., 2., 3. θα συμπληρώνει στη μνήμη τα στοιχεία 4 παικτών και στη συνέχεια θα υπολογίζει και να εμφανίζει στην οθόνη τη μέση τιμή της ηλικίας τους