ΔΟΜΕΣ (structures)

Ο τρόπος αναπαράστασης πολύπλοκων ή/και σύνθετων δεδομένων στη C είναι μέσω μιας ευέλικτης μορφής που ονομάζεται δομή. Οι δομές είναι πεπερασμένες συλλογές μεταβλητών, διαφόρων τύπων στη γενική περίπτωση, και οι οποίες ομαδοποιούνται υπό ένα και μοναδικό όνομα. Η πρόσβαση στις μεταβλητές αυτές γίνεται μέσω του ονόματος της δομής. Η γενική μορφή μιας δομής στη C είναι η εξής:

```
struct όνομα δομής {
  τύπος   στοιχείο1;
  τύπος   στοιχείο2;
  . . .
  τύπος   στοιχείοN;
};
```

όπου τα "στοιχεία" μπορεί να είναι απλές μεταβλητές, κατασκευές (π.χ. πίνακες, συμβολοσειρές), δείκτες ή δομές.

ΔΟΜΕΣ . . .

Στο παρακάτω παράδειγμα, η δομή book,

```
struct book {
    int   serial_nr;      /* 1ο πεδίο */
    char  title[SIZE];    /* 2ο πεδίο */
    char  author[SIZE];   /* 3ο πεδίο */
    float value;          /* 4ο πεδίο */
};
```

προσδιορίζει τέσσερα πεδία για τις μεταβλητές και συμβολοσειρές: serial_nr (τύπου int), title και author (τύπου πίνακα από char) και value (τύπου float). Το πρώτο πεδίο είναι για τον αύξοντα αριθμό, τα επόμενα δύο πεδία είναι για τον τίτλο του βιβλίου και το όνομα του πρώτου συγγραφέα αντίστοιχα και το τέταρτο πεδίο είναι για την τιμή κοστολόγησης.

Μπορούμε να ερμηνεύσουμε τον ορισμό μιας δομής με τον καθορισμό ενός δικού μας (σύνθετου ή μικτού) τύπου δεδομένων. Έτσι, μπορούμε να ορίσουμε και μεταβλητές αυτού του τύπου, όπως, για παράδειγμα, με τη διακήρυξη:

```
struct book    book1, book2,  oldbooks[100],  *bkptr;
```

τύπος — *μεταβλητές δομής* — *πίνακας δομών* — *pointer σε δομή*

ΔΕΣΜΕΥΣΗ ΜΝΗΜΗΣ

Κατά τη διακήρυξη των μεταβλητών, δεσμεύεται και κατάλληλος χώρος στη μνήμη του υπολογιστή ανάλογα με τον ορισμό των πεδίων της δομής. Έτσι, υποθέτοντας ότι ο τύπος int απαιτεί 4 bytes μνήμης, για κάθε μια από τις `book1` και `book2`, θα δεσμευθούν

1 x sizeof (int) + 2 x SIZE x sizeof (char) + 1 x sizeof (float) = 2 x (SIZE + 4) bytes.

Αντίστοιχα, για τον πίνακα `oldbooks` θα δεσμευθούν **200 x (SIZE + 4) bytes** και για τον pointer `bkptr` θα δεσμευθούν **4 bytes**.

Επίσης, μπορείτε να δείτε τον χώρο μνήμης που δεσμεύεται μέσω του τελεστή `sizeof` (π.χ., η `sizeof book1` δίνει το μέγεθος της `book1` σε bytes). Αν δεν συμφωνεί με τον παραπάνω αναλυτικό υπολογισμό, αυτό θα οφείλεται στο ότι εμείς υποθέσαμε byte-addressable σύστημα ενώ στην ουσία πρόκειται για word-addressable σύστημα. Π.χ. για words των 4 bytes και SIZE = 27, στη μεν περίπτωση byte addressable συστήματος οι book1 και oldbooks δεσμεύουν 62 bytes και 6200 bytes αντίστοιχα ενώ στην περίπτωση word addressable συστήματος δεσμεύουν 64 bytes και 6400 bytes αντίστοιχα (η διεύθυνση του πρώτου byte και ο συνολικός χώρος που δεσμεύεται για κάθε μεταβλητή ή πίνακα πρέπει να είναι σε πολλαπλάσιο των 4 bytes).

ΠΡΟΣΠΕΛΑΣΗ ΣΤΑ ΠΕΔΙΑ ΔΟΜΗΣ

Η προσπέλαση στα πεδία μιας δομής γίνεται μέσω του τελεστή μέλους δομής ' . '. Για παράδειγμα, η εκχώρηση τιμών στα πεδία της `book1` με βάση τα στοιχεία του βιβλίου

1. A. Tomaras, "C – Theory and Practice", €14.67

μπορεί να γίνει με τον εξής τρόπο:

```
book1.serial_nr = 1;
strcpy(book1.title, "C - Theory and Practice");
  /* Αν όμως δοθεί από το πληκτρολόγιο τότε: gets(book1.title); */
strcpy(book1.author, "A. Tomaras");
book1.value = 14.67;
```

ενώ το διάβασμα των στοιχείων 100 βιβλίων στον πίνακα oldbooks θα μπορούσε, για παράδειγμα, να γίνει με το παρακάτω πρόγραμμα:

```
#include <stdio.h>
struct book {
    int   serial_nr;
    char  title[81];
    char  author[81];
    float value;
};    /* Global ορισμός δομής */
```

```
int main(void)
{
  int count = 0;
  char answer[5], junk[80];
  struct book oldbooks[100];

  printf("Do you want to enter a new book? (yes/no) :\n");
  while( gets(answer) != NULL && !strcmp(answer,"yes")
         && count < 100 )
  {
    oldbooks[count].serial_nr = count + 1;
    printf("Titlos bibliou?\n");
    gets(oldbooks[count].title);
    printf("Prwtos syggrafeas?\n");
    gets(oldbooks[count].author);
    printf("Axia bibliou se Euro?\n");
    scanf("%f",&oldbooks[count].value);
    gets(junk);        /* Αδειασε τον buffer */
    count++;
    printf("Do you have another book? (yes/no) :\n");
  }
  return 0;
}
```

Σχολιασμός προγράμματος

Στο παραπάνω πρόγραμμα η συμβολοσειρά **`junk`** (σκουπίδια) χρησιμοποιείται για να αδειάζουμε τον buffer από χαρακτήρες που άφησε η scanf() (ό,τι ακολουθεί στη γραμμή εισόδου μέχρι και τον χαρακτήρα '\n') πριν διαβάσουμε την επόμενη είσοδο με gets(). Επίσης, παρατηρήστε ότι αφού η strcmp() επιστρέφει την τιμή 0 όταν οι συμβολοσειρές είναι ίδιες, η !strcmp(answer,"yes") θα επιστρέψει 1 αν πληκτρολογήσαμε "yes".

Ο πίνακας **`oldbooks`** έχει σαν στοιχεία τα **`oldbooks[i]`** (i = 0, 1, ..., 99). Το κάθε ένα από αυτά τα στοιχεία είναι μια δομή τύπου **`struct book`**. Κατά συνέπεια, μπορούμε να προσπελάσουμε τα πεδία (ή μέλη) της **`oldbooks[i]`** με τον τελεστή '.' όπως και για τις απλές μεταβλητές δομής. Έτσι, για παράδειγμα, η αξία του 15ου βιβλίου θα βρίσκεται στο **`oldbooks[14].value`** (***Προσοχή***: *όχι oldbooks.value[14]*) με διεύθυνση στη μνήμη την &oldbooks[14].value. Αντίστοιχα, η διεύθυνση μνήμης στην οποία βρίσκεται αποθηκευμένη η δομή oldbooks[14] θα είναι η &oldbooks[14]. Οι ομοιότητες με τους κλασικούς πίνακες είναι προφανείς. Ο πίνακας δομών είναι απλώς ένας γενικευμένος πίνακας στον οποίο αποθηκεύουμε διαφόρων τύπων δεδομένα.

ΟΡΙΣΜΟΣ ΔΟΜΗΣ: ΜΕ ΟΝΟΜΑ Ή ΧΩΡΙΣ ΟΝΟΜΑ

Συνήθως, παράλληλα με τη δήλωση μιας δομής της δίνουμε και ένα όνομα (βλ. για παράδειγμα τη δήλωση της δομής `struct book`). Έχοντας δώσει όνομα στη δομή, μπορούμε στη συνέχεια να διακηρύξουμε μεταβλητές δομής με δηλώσεις της μορφής:

```
struct book  book1;
```

Πολλές φορές, δηλώνουμε μια δομή εξωτερικά ώστε να φαίνεται από όλες τις συναρτήσεις ενώ ορίζουμε τοπικές μεταβλητές αυτής της δομής στις συναρτήσεις που τις χρησιμοποιούν.

Αν ο ορισμός της δομής και των μεταβλητών γίνεται στο ίδιο σημείο του προγράμματος, τότε μπορούμε να αποφύγουμε να δώσουμε όνομα στη δομή. Στην περίπτωση αυτή έχουμε ταυτόχρονα ορισμό δομής (χωρίς όνομα) και μεταβλητών. Παράδειγμα:

```
 struct {                /* Δεν υπάρχει όνομα δομής */
      int   serial_nr;
      char  title[SIZE];
      char  author[SIZE];
      float value;
  } oldbooks[100], book1, book2, *bkptr;
```

ΑΡΧΙΚΟΠΟΙΗΣΗ ΔΟΜΗΣ

Παράδειγμα αρχικοποίησης δομής με όνομα:

```
struct book book1 = {
   1,
   "C – Theory and Practice",
   "A. Tomaras",
   14.67
};
```

Παράδειγμα αρχικοποίησης δομής χωρίς όνομα:

```
struct {
    int   serial_nr;
    char  title[SIZE];
    char  author[SIZE];
    float value;
}  book1 = {1, "C – Theory and Practice", "A. Tomaras", 14.67};
```

ΑΡΧΙΚΟΠΟΙΗΣΗ ΠΙΝΑΚΑ ΔΟΜΩΝ

Παράδειγμα αρχικοποίησης πίνακα δομών:

```
struct book  C_books[] = {
    { 1,
      "C – Theory and Practice",
      "A. Tomaras",
      14.67  },
    { 2,
      "H glwssa C se bathos",
      "N. Hatzigiannakis",
      16.50 },
    { 3,
      "C Programmaing Language",
      "B. Kernighan",
      15.25 }
};
```

ΕΜΦΩΛΕΥΜΕΝΕΣ ΔΟΜΕΣ

```
struct onomatepwnymo {      /* Η πρώτη δομή */
      char  onoma[30];
      char  epwnymo[30];
};
struct stoixeia {     /* Δεύτερη δομή που περιέχει την πρώτη */
      struct onomatepwnymo names;
      char  favfood[30];
      char  job[80];
      float income;
};
```

Έστω τώρα ότι έχουμε διακηρύξει την ακόλουθη μεταβλητή:

```
struct stoixeia atomo;
```

Η πρόσβαση στα πεδία και υποπεδία της μεταβλητής μπορεί να γίνει σύμφωνα με το εξής παράδειγμα:

```
strcpy(atomo.names.onoma,"Bill");     ή  gets(atomo.names.onoma);
strcpy(atomo.names.epwnymo,"Gates"); ή  gets(atomo.names.epwnymo);
...
atomo.income = 1.0e+09;
```

ΔΕΙΚΤΕΣ ΣΕ ΔΟΜΕΣ

```
#include <stdio.h>

#define LEN 80

struct CD {
    char  band[LEN];
    char  album[LEN];
    int   year;
};

int main()
{
  struct CD rock[2] = {
    { "Joe Cocker", "Greatest Hits", 1998 },
    { "Bruce Springsteen", "The Rising", 2002 }
  };
  struct CD *cdptr;    /* Pointer σε δομή */

  printf("&rock[0] = %u, &rock[1] = %u\n",&rock[0],&rock[1]);
```

ΔΕΙΚΤΕΣ ΣΕ ΔΟΜΕΣ . . .

```
  cdptr = &rock[0];
  printf("cdptr =    %u, cdptr+1 =  %u\n",cdptr,cdptr+1);
  printf("cdptr->year is %d, (*cdptr).year is %d\n",
           cdptr->year, (*cdptr).year);
  cdptr++;
  printf("BAND: %s - ALBUM: %s – YEAR: %d\n",
           cdptr->band, cdptr->album, cdptr->year);
  return 0;
}
```

Έξοδος προγράμματος:

```
&rock[0] = 4235424, &rock[1] = 4235588
cdptr =    4235424, cdptr+1 =  4235588
cdptr->year is 1998, (*cdptr).year is 1998
BAND: Bruce Springsteen - ALBUM: The Rising – YEAR: 2002
```

Σχόλια

α) Όταν αυξάνουμε κατά ένα τον δείκτη, η διεύθυνσή του αυξάνεται κατά το πλήθος των bytes που απαιτούνται για μια δομή. Έτσι, επειδή τα στοιχεία του πίνακα είναι δομές που η κάθε μία απαιτεί 80 + 80 + 4 + 4 = 168 bytes, η διεύθυνση του cdptr+1 θα είναι: 4235424 + 168 = 4235592.

β) Η C παρέχει δυνατότητα προσπέλασης των μελών μια δομής απ'ευθείας από δείκτη στη δομή. Αυτό γίνεται μέσω του τελεστή `'->'`. Με άλλα λόγια, ένας δείκτης σε δομή που ακολουθείται από τον τελεστή '->' λειτουργεί με τον ίδιο ακριβώς τρόπο που μια μεταβλητή δομής ακολουθείται από τον τελεστή `'.'`. Στο παραπάνω πρόγραμμα διαπιστώσαμε ότι
`cdptr->year` == **`(*cdptr).year`**
κάτι που μπορεί να εξηγηθεί από το γεγονός ότι ενώ ο cdptr είναι ένας δείκτης σε δομή, το *cdptr θα είναι στην ουσία η ίδια η (σύνθετη) θέση μνήμης των 168 bytes που αντιστοιχεί στη δομή, κατ'αναλογία με τους δείκτες σε μεταβλητές. Οι παρενθέσεις στο **`(*cdptr).year`** είναι απαραίτητες καθώς ο τελεστής '.' έχει μεγαλύτερη προτεραιότητα από τον '*'. Συνεπώς, το **`*cdptr.year`** ερμηνεύεται ως **`*(cdptr.year)`** με το τελευταίο να μην έχει νόημα.

ΔΟΜΕΣ ΚΑΙ ΣΥΝΑΡΤΗΣΕΙΣ

Γνωρίζουμε ήδη ότι από τα ορίσματα των συναρτήσεων περνάμε τιμές στη συνάρτηση. Οι τιμές αυτές μπορεί να είναι, για παράδειγμα, τύπου int, float ή ακόμη και διευθύνσεις. Λόγω, όμως, της πολυπλοκότητας μιας δομής, σε κάποιους μεταγλωττιστές δεν επιτρέπεται οι δομές να χρησιμοποιούνται ως ορίσματα σε συναρτήσεις (πάντως, στην ANSI C επιτρέπεται να περνάμε δομές ως ορίσματα συναρτήσεων). Ο Dev-C++ επιτρέπει ορίσματα συναρτήσεων τύπου δομής.

Καθώς μια δομή μπορεί να περιλαμβάνει πεδία μεγάλων διαστάσεων (π.χ. εικόνες, μουσικά κομμάτια, κ.λπ.), είναι προφανές ότι μπορεί να καταλαμβάνει μεγάλη έκταση στη μνήμη. Για τον λόγο αυτό, συνιστάται εντόνως η χρήση δεικτών σε δομές ως ο ενδεδειγμένος τρόπος επικοινωνίας με συναρτήσεις.