Προγραμματισμός Υπολογιστών

Δομές

Νικόλαος Ζ. Ζάχαρης
Καθηγητής

Πανεπιστήμιο Δυτικής Αττικής
Τμήμα Μηχανικών Πληροφορικής και Υπολογιστών

ΔΟΜΕΣ (structures)

Η C όπως και οι άλλες γλώσσες προγραμματισμού μας επιτρέπει να δημιουργήσουμε μια δομή η οποία αποτελείται από βασικούς τύπους δεδομένων.

Για παράδειγμα αν σε ένα πρόγραμμα θέλαμε να αποθηκεύσουμε την ημερομηνία γέννησης ενός υπαλλήλου. Τότε θα έπρεπε να χρησιμοποιήσουμε τρεις ακεραίους αριθμούς, ένα για την ημέρα, ένα για το μήνα και ένα για το χρόνο γέννησης.

```
int day, month, year;
```

Με την δημιουργία μιας δομής πάλι θα χρησιμοποιήσουμε τους τρεις βασικούς τύπους αλλά ενοποιημένους κάτω από το περίβλημα ενός κοινού ονόματος.

```
struct Birthday {
    int day;
    int month;
    int year;
    };
```

Η λέξη struct είναι δεσμευμένη στην C και μέσα στα άγκιστρα μπορεί να περιέχει βασικούς τύπους δεδομένων ή άλλες δομές. Η λέξη Birthday είναι το όνομα της δομής. Προσοχή το ερωτηματικό στο τέλος.

```
struct  Birthday {
      int day;
      int month;
      int year;
      };

int main(int argc, char *argv[]) {
  struct Birthday myBirthday;

  myBirthday.day = 2;
  myBirthday.month = 4;
  myBirthday.year = 1967;


  printf("%d", myBirthday.day);
  printf("%d", myBirthday.month);
  printf("%d", myBirthday.year);

  return 0;
}
```

Η δομή περιέχει τους βασικούς τύπους δεδομένων και τους ενώνει κάτω από ένα κοινό όνομα, το Birthday.

Η myBirthday είναι μια μεταβλητή τύπου struct Birthday.

Η πρόσβαση στα στοιχεία μιας δομής γίνεται με χρήση της τελείας. Προηγείται το όνομα της μεταβλητής, εν συνεχεία η τελεία και έπειτα το όνομα του πεδίου. Π.χ Για να δώσουμε την τιμή 2 στο πεδίο day της δομής myBirthday γράφουμε

myBirthday.day = 2;

Εναλλακτικός τρόπος γραφής

Διαφορετικά μπορούμε να δημιουργήσουμε ένα νέο ορισμό, τον Birthday, ο οποίος είναι μια δομή. Δεν χρειάζεται να προηγείται το struct στη δήλωση μεταβλητής.

```
typedef struct {
      int day;
      int month;
      int year;
      } Birthday;

int main(int argc, char *argv[]) {
  Birthday myBirthday;

  myBirthday.day = 2;
  myBirthday.month = 4;
  myBirthday.year = 1967;
  return 0;
}
```

Εναλλακτικός τρόπος γραφής

Διαφορετικά μπορούμε να δημιουργήσουμε μια επώνυμη ή ανώνυμη δομή και μια ή περισσότερες μεταβλητές της.

Ανώνυμη	Επώνυμη
struct { int day; int month; int year; } maryBirthday;	struct Birthday { int day; int month; int year; } tomBirthday;

```
int main(int argc, char *argv[]) {

  myBirthday.day = 2;
  myBirthday.month = 4;
  myBirthday.year = 1967;
  return 0;
}
```

Αν τα πλεονεκτήματα της δομής δεν είναι αμέσως ορατά μπορούμε να σκεφτούμε τι θα χρειαζόταν στην περίπτωση ενός πίνακα από ημερομηνίες. Θα μπορούσαμε να είχαμε είτε τρεις πίνακες ακεραίων είτε ένα πίνακα όπου η κάθε θέση είναι μια δομή.

```
struct Birthday {
    int day;
    int month;
    int year;
    } eDays[100];
```

Η δήλωση ενός πίνακα 100 θέσεων με το όνομα eDays όπου η κάθε μία είναι μια δομή τύπου Birthday.

Δομή για τη περιγραφή της πληροφορίας

Η δομή μας επιτρέπει να σκεφτόμαστε το πρόβλημα που επιλύουμε με φυσικό τρόπο. Για παράδειγμα θα μπορούσαμε να ορίσουμε μια δομή με το όνομα employee η οποία θα περιέχει το όνομα (name), την διεύθυνση (address), την ημερομηνία πρόσληψης ενός υπαλλήλου (hireDate) καθώς και άλλα στοιχεία. Η ημερομηνία πρόσληψης θα μπορούσε με την σειρά της να είναι και αυτή μια δομή, όπως ακριβώς την περιγράψαμε προηγουμένως. Τότε η απόδοση της τιμής 7 στο πεδίο day της δομής hireDate θα γίνει όπως παρακάτω :

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

struct Birthday {
      int day;
      int month;
      int year;
      };

struct employee {
       char name[20];
       char address[20];
       struct Birthday hireDate;
};
```

```
int main(int argc, char *argv[]) {
   struct employee Jim;

  strcpy(Jim.name, "Jim Page");
  strcpy(Jim.address, "Cold Mountain");
  Jim.hireDate.month = 11;
  Jim.hireDate.day = 7;
  Jim.hireDate.year = 2001;

  printf("%s\n", Jim.name);
  printf("%d\n", Jim.hireDate.month);
  printf("%d\n", Jim.hireDate.day);
  return 0;
}
```

Αρχικοποίηση μιας δομής (1-2)

Κατά τη δημιουργία μιας μεταβλητής τύπου δομής, μπορούμε να δώσουμε τιμές και σε όλα τα πεδία της.

```
#include <stdio.h>
#include <stdlib.h>

struct  Birthday {
      int day;
      int month;
      int year;
      };

int main(int argc, char *argv[]) {
  struct Birthday oneDate = {12,3,2001};

  printf("%d\n", oneDate.day);
  printf("%d\n", oneDate.month);
  printf("%d\n", oneDate.year);

  return 0;
}
```

Αρχικοποίηση μιας δομής (2-2)

```
#include <stdio.h>
#include <stdlib.h>

struct Birthday {
      int day;
      int month;
      int year;
      };

struct employee {
       char name[20];
       char address[20];
       struct Birthday hireDate;
};

int main(int argc, char *argv[]) {
  struct employee Mary = {"Mary Jones", "Petrou Ralli 280", {12, 5, 2014}};

  printf("%s\n", Mary.name);
  printf("%s\n", Mary.address);
  printf("%d\n", Mary.hireDate.month);
  return 0;
}
```

Η αρχικοποίηση μιας δομής μπορεί να γίνει και στην περίπτωση που περιέχει άλλη δομή στα πεδία της.

Δομές σαν παράμετροι σε συναρτήσεις

Στις παραμέτρους μιας συνάρτησης εκτός από τους βασικούς τύπους μπορούμε να συμπεριλάβουμε και δομές και να διαχειριστούμε τα δεδομένα της δομής μέσα στο σώμα της συνάρτησης.

Για παράδειγμα στην δομή Birthday περιλαμβάνονται τα δεδομένα μιας ημερομηνίας. Επίσης γνωρίζουμε ότι τα ίδια στοιχεία – ημέρα, μήνα, έτος - διατηρούμε είτε πρόκειται για την Ελλάδα είτε για την Αγγλία. Η μόνη διαφορά είναι ότι οι Έλληνες χρησιμοποιούν την μορφή Ημέρα/Μήνας/Χρόνος ενώ οι Άγγλοι χρησιμοποιούν την μορφή Μήνας/Ημέρα/Χρόνος.

Μπορούμε να κατασκευάσουμε μια συνάρτηση η οποία δέχεται σαν όρισμα μια δομή τύπου Birthday και ένα ακέραιο αριθμό, ο οποίος καθορίζει το πως θα εκτυπωθούν οι τιμές της δομής. Πιο συγκεκριμένα αν ο αριθμός έχει τιμή 0 τότε η συνάρτηση θα εκτυπώσει την ημερομηνία βάση του αγγλικού προτύπου ενώ με οποιαδήποτε τιμή εκτός 0, θα εκτυπώσει την ημερομηνία βάση του ελληνικού προτύπου

```
struct Birthday {
      int day;
      int month;
      int year;
};
```

```
void showDate(struct Birthday t, int option){
   if(option)   {
        printf("%d/%d/%d",t.day ,t.month, t.year);
   }
   else    {
          printf("%d/%d/%d",t.month,t.day , t.year);
   }
}
```

```
int main(int argc, char *argv[])  {
  struct Birthday myBirthday;

  myBirthday.day = 2;
  myBirthday.month = 4;
  myBirthday.year = 1967;

  showDate(myBirthday, 1);

  return 0;
}
```

Το πέρασμα μιας δομής σαν παράμετρο σε μια συνάρτηση γίνεται με τον ίδιο τρόπο όπως και σε κάθε άλλη μεταβλητή.

Δείκτες σε Δομές και ο τελεστής -> (1-2)

Όπως και στους βασικούς τύπους έτσι και στις δομές μπορούμε να χρησιμοποιήσουμε τους δείκτες.

```
#include <stdio.h>
#include <stdlib.h>

struct Birthday {
      int day;
      int month;
      int year;
      };

int main(int argc, char *argv[]) {
  struct Birthday myBirthday;
  struct Birthday *pointer;

  pointer = & myBirthday;
```

Η δήλωση (*pointer).year είναι ακριβώς η ίδια με την δήλωση

pointer->year

```
  myBirthday.day = 2;
  myBirthday.month = 4;
  myBirthday.year = 1967;

   printf("%d\n",(*pointer).day);
  printf("%d\n",(*pointer).month);
  printf("%d\n",pointer->year);

  return 0;
}
```

Δείκτες σε Δομές και ο τελεστής -> (2-2)

Η δομή μπορεί να χρησιμοποιηθεί για κλήση μιας συνάρτησης με αναφορά στη διεύθυνση της δομής.

```
#include <stdio.h>
#include <stdlib.h>

struct Birthday {
     int day;
     int month;
     int year;
     };

void readDate(struct Birthday *oneBirthday) {
   printf("Type day : ");
   scanf("%d", &oneBirthday->day);
   printf("Type month : ");
   scanf("%d", &(*oneBirthday).month);
   printf("Type year : ");
   scanf("%d", &oneBirthday->year);
}
```

```
int main(int argc, char *argv[]) {
  struct Birthday myBirthday;

  readDate(&myBirthday);
  showDate(myBirthday, 1);
  return 0;
}
```

Η λειτουργία ShowDate έχει υλοποιηθεί στις προηγούμενες διαφάνειες.

Επιστροφή δομής από συνάρτηση

Μια συνάρτηση μπορεί να επιστρέφει σαν τιμή μια δομή.

```
#include <stdio.h>
#include <stdlib.h>

struct Birthday {
      int day;
      int month;
      int year;
      } ;

struct Birthday Initialize (int day, int month, int year) {
            struct Birthday test;
            test.day = day;
            test.month = month;
            test.year = year;

            return test;
}
```

```
int main(int argc, char *argv[]) {

  struct Birthday x;
  x  = Initialize(23, 9, 2016);
  showDate(x, 1);
  return 0;
}
```

Η λειτουργία ShowDate έχει υλοποιηθεί στις προηγούμενες διαφάνειες.

Συγκρίσεις με δομές (1-2)

Μεταξύ δύο δομών δεν μπορεί να γίνει σύγκριση με τη χρήση των τελεστών σύγκρισης, αλλά θα πρέπει να δημιουργήσουμε μία συνάρτηση η οποία θα συγκρίνει τις δομές είτε ως προς ένα πεδίο τους (π.χ. μόνο το year) είτε προς κάποιο συνδυασμό τους.

```
#include <stdio.h>
#include <stdlib.h>

struct Birthday {
      int day;
      int month;
      int year;
      }  ;

const int monthDays[12] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

int getMonthDays(int month) {
                int i, n = 0;
       for (i=0; i< month - 1; i++)
              n += monthDays[i];
    return n;
}
```

Συγκρίσεις με δομές (2-2)

```
int Compare(struct Birthday aDate, struct Birthday bDate) {
    int adays = aDate.year * 365 + getMonthDays(aDate.month) + aDate.day;
    int bdays = bDate.year * 365 + getMonthDays(bDate.month) + bDate.day;

   if(adays > bdays) {
            return -1;
    }
    else if(adays < bdays) {
            return 1;
     }
     else {
          return 0;
     }
}

int main(int argc, char *argv[]) {
   struct Birthday x = Initialize(23, 9, 2016);
   struct Birthday y = Initialize(23, 9, 2016);

   if( Compare(x, y) == 0)  {
     printf("The two days are the same.\n");
  }
   return 0;
}
```

Η Compare υπολογίζει τα πλήθη των ημερών για δύο ημερομηνίες και τα συγκρίνει μεταξύ τους