Δομημένος Προγραμματισμός

Τμήμα Επιχειρηματικού Σχεδιασμού και Πληροφοριακών Συστημάτων

www.bpis.teicrete.gr

Ορισμός & Χρήση Συναρτήσεων

```
#include <stdio.h>

int SquareIt(int a)
{
   int result;
   result = a*a;
   return result;
}

void main()
{
  int num, square;
  printf("Enter a number:");
  scanf(" %d", &num );
  square = SquareIt( num );
  printf("The square of %d is: %d", num,square);
}
```

Παράμετρος Συνάρτησης

Ορισμός Συνάρτησης

Σώμα Συνάρτησης

Τιμή Συνάρτησης

Επιστρεφόμενη τιμή Συνάρτησης

Όρισμα Συνάρτησης

Κλήση Συνάρτησης

Πίνακες ως ορίσματα συναρτήσεων

Εάν κατά την κλήση μίας συνάρτησης το όρισμα είναι όνομα πίνακα, δεν αποστέλλει στη συνάρτηση ολόκληρο τον πίνακα αλλά τη διεύθυνση του πρώτου byte του πίνακα.

Η παράμετρος στη δήλωση της συνάρτησης είναι ένα όνομα τοπικού πίνακα (π.χ. **number**). Κρατά ένα αντίγραφο της ίδιας διεύθυνσης αλλά χρησιμοποιεί διαφορετικό όνομα.

Επιπρόσθετα, στη δήλωση δεν αναφέρεται το ακριβές μέγεθός του αλλά μόνο το γεγονός ότι είναι πίνακας καθώς και ο τύπος στοιχείων του.

Επομένως ουσιαστικά γνωστοποιείται στο μεταγλωττιστή η διεύθυνση του πρώτου byte και η απόσταση (αριθμός bytes) μεταξύ των στοιχείων:

Παράδειγμα με όρισμα πίνακα

```
#include <stdio.h>

display( int num[ ] )
{
   int i;
   for (i=0; i<=9; i++)
      printf( "%d ",num[i] );
}

void main()
{
   int table[10], i;
   for (i=0; i<=9; i++)
      table[i]=i;
   display( table );
}
```

Νέο Πρόγραμμα

Δημιουργήστε ένα νέο project με τίτλο

function3

Άσκηση

Ορίστε μια συνάρτηση με 2 ορίσματα: ένα πίνακα και ένα αριθμό που πρόκειται να αναζητηθεί στον πίνακα. Η συνάρτηση επιστρέφει 1 αν ο αριθμός βρεθεί στον πίνακα και 0 αν δεν βρεθεί.

Χρησιμοποιήστε τη συνάρτηση σε συνδυασμό με τον αριθμό που εισάγει ο χρήστης και τυπώστε
FOUND ή NOT FOUND
αν βρεθεί ή όχι ο αριθμός στον πίνακα

Νέο Πρόγραμμα

Δημιουργήστε ένα νέο project με τίτλο

function4

Άσκηση

Γράψτε ένα πρόγραμμα που θα τυπώνει 5 τυχαίους αριθμούς από το 1 έως το 45 και 1 τυχαίο αριθμό από το 1 έως το 20.

Χρησιμοποιήστε τη συνάρτηση rand και τον τελεστή % για να δημιουργήσετε τυχαίες τιμές

```
int r;
srand(time());
r = 1 + (rand() % 45);
```

Νέο Πρόγραμμα

Δημιουργήστε ένα νέο project με τίτλο

function5

Άσκηση

Ορίστε ένα πίνακα 3 θέσεων, και εισάγετε πραγματικούς αριθμούς από το πληκτρολόγιο

Γράψτε μία συνάρτηση που να υπολογίζει το άθροισμα των 3 αριθμών και να επιστρέφει αυτή τη τιμή.

Χρησιμοποιήστε μία συνάρτηση με όρισμα ένα πίνακα πραγματικών (που θα περιέχει τους αριθμούς)

Τοπικές Μεταβλητές

Στο σώμα μίας συνάρτησης μπορούμε να ορίσουμε μεταβλητές

Οι μεταβλητές αυτές έχουν περίοδο ζωής όσο διαρκεί η λειτουργία της συνάρτησης

Οι μεταβλητές αυτές έχουν εμβέλεια και δικαίωμα χρήσης μόνο στο σώμα της συγκεκριμένης συνάρτησης που ορίζονται

Είναι γενικά κακό προγραμματιστικό στυλ να χρησιμοποιούμε το ίδιο όνομα για μία τοπική μεταβλητή (local variable) και μία καθολική (global variable)

Εμβέλεια Μεταβλητών

```
int num;

void main()
{
      int square;

      printf(“Enter a number:”);
      scanf(“ %d”, &num );
      square = SquareIt( num );

      printf(“The square of %d is: %d”, num, square);
}

int SquareIt( int a )
{
     int result;
     result = a*a;
     return result;
}
```

Ολική εμβέλεια
Global Variable
μεταβλητή num

Τοπική εμβέλεια
Local Variable
μεταβλητή square

Τοπική εμβέλεια
Local Variable
μεταβλητή square

Διάρκεια μεταβλητών

Η διάρκεια ορίζει το χρόνο κατά τον οποίο το όνομα της μεταβλητής είναι συνδεδεμένο με τη θέση μνήμης που περιέχει την τιμή της μεταβλητής.

Ορίζονται ως **χρόνοι δέσμευσης** και **αποδέσμευσης** οι χρόνοι που το όνομα συνδέεται με τη μνήμη και αποσυνδέεται από αυτή, αντίστοιχα.

Καθολικές Μεταβλητές

δεσμεύεται χώρος με την έναρξη εκτέλεσης του προγράμματος και η μεταβλητή συσχετίζεται με την ίδια θέση μνήμης έως το τέλος του προγράμματος. Είναι **πλήρους διάρκειας**.

Τοπικές Μεταβλητές

Η ανάθεση της μνήμης σε τοπική μεταβλητή γίνεται με τη είσοδο στο χώρο εμβέλειάς της και η αποδέσμευσή της με την έξοδο από αυτόν. Δηλαδή η τοπική μεταβλητή δε διατηρεί την τιμή της από τη μία κλήση της συνάρτησης στην επόμενη. Είναι **περιορισμένης διάρκειας**

Νέο Πρόγραμμα

Δημιουργήστε ένα νέο project με τίτλο

function6

Άσκηση

Γράψτε μία συνάρτηση που να υπολογίζει το άθροισμα 3 αριθμών.

Καλέστε τη συνάρτηση σας 3 φορές για τις παρακάτω τριάδες αριθμών

10 20 30

40 50 60

70 80 90

υπολογίστε το γενικό άθροισμα και των 9 αριθμών με χρήση καθολικής μεταβλητής

Πρωτότυπα Συναρτήσεων (Function Prototypes)

Επιστρεφόμενος-τύπος όνομα-συνάρτησης (λίστα-τύπων-παραμέτρων);

- Δηλώσεις των συναρτήσεων στην αρχή του αρχείου
- Χρειάζεται οπωσδήποτε αν χρησιμοποιούμε τη συνάρτηση πριν τον ορισμό της

Πλεονεκτήματα

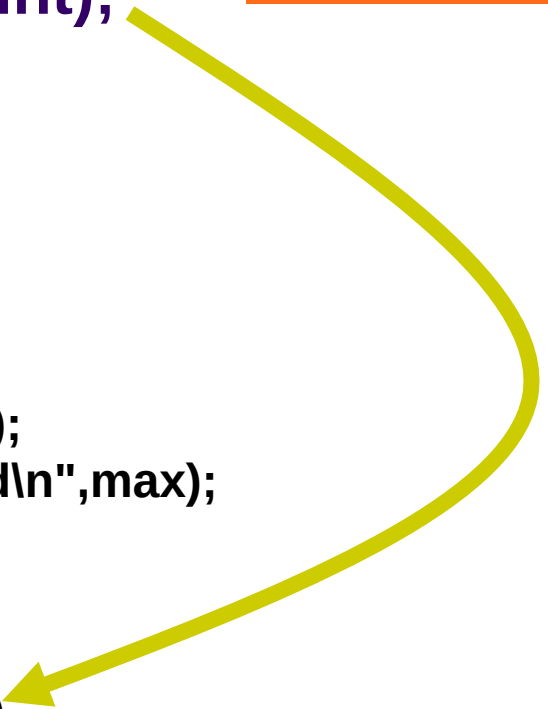
- Ελέγχουμε την ορθή χρήση των συναρτήσεων
- Μπορούμε να δούμε ποιες είναι οι συναρτήσεις που έχουν υλοποιηθεί στο πρόγραμμα
- Μπορούμε να καλέσουμε όλες τις συναρτήσεις που έχουν υλοποιηθεί στο πρόγραμμα χωρίς να μας ενδιαφέρει η σειρά που έχουν γραφτεί

```
#include <stdio.h>

int maximum( int, int, int);

int main()
{
        int a, b, c;
        int max;
        a=10; b=20; c=30;
        max = maximum(a,b,c);
        printf("Maximum is %d\n",max);
        return 0;
}

int maximum(int x, int y, int z)
{
        int max=x;
        if (y>max)
          max=y;
        if(z>max)
          max=z;
        return max;
}
```



```
#include <stdio.h>

int maximum(int x, int y, int z)
{
        int max=x;
        if (y>max)
          max=y;
        if(z>max)
          max=z;
        return max;
}

int main()
{
        int a, b, c;
        int max;
        a=10; b=20; c=30;
        max = maximum(a,b,c);
        printf("Maximum is %d\n",max);
        return 0;
}
```

Αρχεία Κεφαλίδων

ομαδοποίηση συναρτήσεων σε αρχεία κεφαλίδων

Χρήση των αρχείων κεφαλίδων του χρήστη, όπως και αυτών τους συστήματος

αλλά με " " και όχι < >
π.χ. #include "mymath.h"

Νέο Πρόγραμμα

Δημιουργήστε ένα νέο project με τίτλο

header1

Άσκηση

Στο προηγούμενο παράδειγμα, δημιουργήστε ένα πρότυπο συνάρτησης για την **maximum()**

Δημιουργήστε το αρχείο κεφαλίδας myheader.h και ορίστε εκεί τη συνάρτηση maximum()

Καλέστε την στο πρόγραμμα σας χρησιμοποιώντας κατάλληλη πρόταση **include**

Static Μεταβλητές

Όλες οι μεταβλητές είναι αυτόματα δηλωμένες σαν **automatic**

→όταν η εμβέλεια τους περάσει τότε ο χώρος μνήμης που καταλαμβάνουν γίνεται διαθέσιμος για άλλες μεταβλητές που θα χρειαστούν κατά την εκτέλεση του προγράμματος.

Static Μεταβλητές

Συνεπώς όταν μια συνάρτηση χρησιμοποιεί για παράδειγμα κάποια τοπική μεταβλητή k, και αυτή η μεταβλητή λαμβάνει κάποια τιμή μέσα στη συνάρτηση π.χ. 5 τότε με το τελείωμα της συνάρτησης η εμβέλεια της μεταβλητής k λήγει.

Εάν η συνάρτηση κληθεί ξανά δεν μπορούμε να είμαστε σίγουροι ότι η τιμή της μεταβλητής k θα είναι ίση με 5 κατά την αρχή της εκτέλεσης της νέας κλήσης της συνάρτησης.

Εάν για κάποιο λόγο χρειάζεται να κρατάμε τη τιμή μιας «τοπικής» μεταβλητής ανάμεσα σε διαδοχικές κλήσεις τότε τη δηλώνουμε σαν **static**.

Νέο Πρόγραμμα

Δημιουργήστε ένα νέο project με τίτλο

static

Άσκηση

Τι πρέπει να αλλάξετε στον κώδικα που σας δόθηκε, ώστε να τυπωθεί το μήνυμα στην οθόνη μόνο μία φορά

Δομές Δεδομένων

- Μέχρι τώρα έχουμε δει μεταβλητές που ο τύπος τους είναι ένας από τους βασικούς τύπους που υποστηρίζει η C (π.χ. int, char, float κλπ.)

- Μια σχετική εξαίρεση ήταν οι Πίνακες που μπορούμε να αποθηκεύσουμε δεδομένα (ενός συγκεκριμένου τύπου) σε μια ακολουθιακή μορφή

- Δομή – Structure
  νέα μορφή μεταβλητής
  - Επιτρέπει να αποθηκευτούν δεδομένα διαφορετικών τύπων που έχουν κάποια «λογική» σχέση μεταξύ τους
  - Είναι πιο δομημένη από την απλή ακολουθιακή μορφή Πινάκων

Παράδειγμα

```
struct όνομα_προτύπου
{
        τύπος_πεδίου_1    όνομα_πεδίου_1;
        τύπος_πεδίου_2    όνομα_πεδίου_2;
        ...
        τύπος_πεδίου_ν    όνομα_πεδίου_ν;
};
```

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

```
struct person
{
        char name[50];
        int age;
        float height;
};
```

```
struct company
{
        char name[50];
        int start_year;
        int field;
        int tax_num;
        int num_empl;
        char addr[50];
        float balance;
};
```

Παράδειγμα

Δήλωση Δομής:

```
struct όνομα_προτύπου όνομα_δομής_1,..., όνομα_δομής_ν;
```

Παραδείγματα:

```
struct person person_1;
```

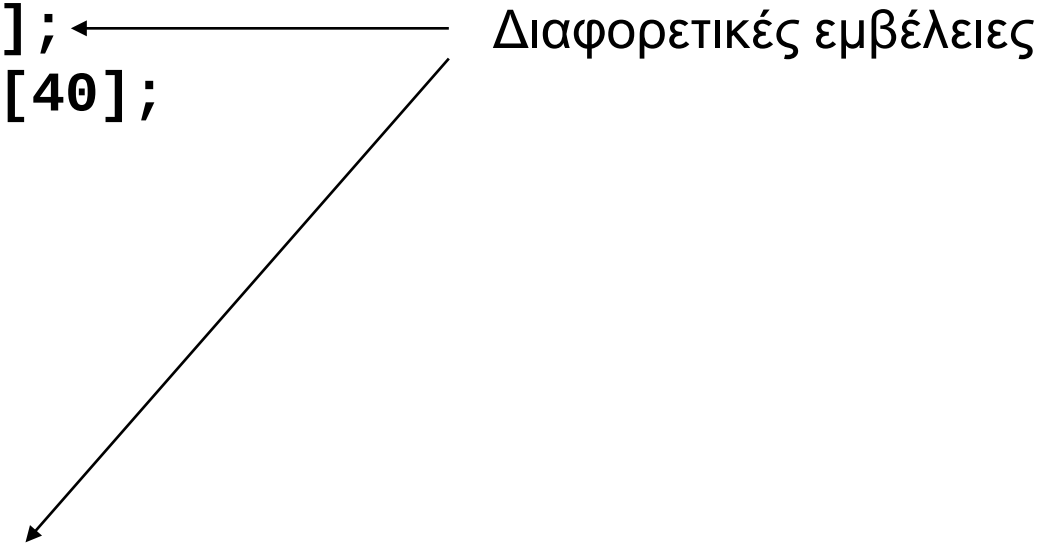
```
struct company comp_1, comp_2;
```

```
struct book
{
        char title[100];
        int year;
        float price;
} book_1, book_2;
```

Παράδειγμα

```
struct {
     int number;
     int sex;
     char name[10];
     char address[40];
     int age;
} person1;


struct {
     int number;
     int name[10];
     int yearsOfEmployment;
} employee1;
```



Αρχικές τιμές σε Δομές

Όπως και στους πίνακες, έτσι και μία δομή μπορεί να πάρει αρχικές τιμές κατά τη δήλωση της π.χ.

```
struct {
    int number;
    int sex;
    char name[11];
  } person1 = {154, 1, “John Smith”},
    person2 = {180, 2, “Mary Jones”};
```

154

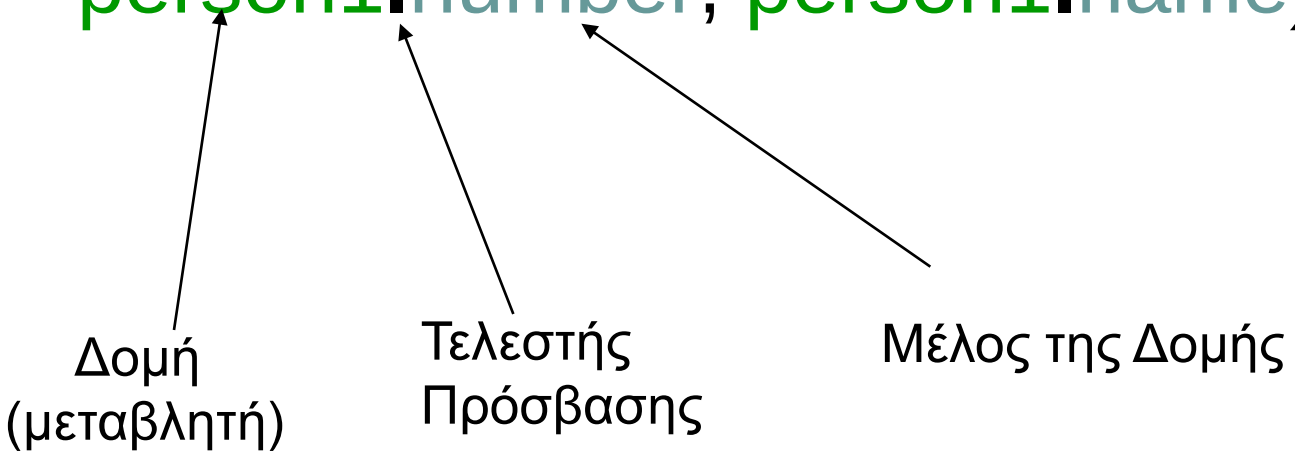
1

John Smith

Λειτουργίες σε Δομές

Όπως και στους πίνακες, έτσι και στις δομές, η πιο κοινή λειτουργία είναι να βρούμε κάποιο στοιχείο της π.χ.

```
printf("The name of person %d is %s",
        person1.number, person1.name);
```



Παράδειγμα

scanf(“%d”, &pesron1.number);

Διαβάζουμε το μέλος number

person1 = person2;

Αποθέτουμε τις τιμές των μελών της person2, στην person1

Σημείωση: Αυτό δεν ήταν εφικτό στους πίνακες

Δήλωση Τύπου Δομής

Θα θέλαμε να έχουμε την ευχέρεια να δηλώσουμε τον δικό μας τύπο και απλά να χρησιμοποιούμε το όνομά του όπως θα χρησιμοποιούσαμε int, float, κλπ

Αυτό επιτυγχάνεται με την εντολή **typedef**

Ένα πρότυπο δομής μπορεί να δηλωθεί εναλλακτικά με χρήση του προσδιοριστικού **typedef**

Σε αυτή την περίπτωση το όνομα του προτύπου εισάγεται μετά το δεξί άγκιστρο, π.χ.

```
typedef struct
{
        char title[100];
        int year;
        float price;
} book;
```

Αν το πρότυπο μίας δομής έχει δηλωθεί με χρήση της εντολής **typedef**, τότε δεν προσθέτουμε τη λέξη **struct** πριν από τη δήλωση μίας δομής

Δηλ. αν είχαμε δηλώσει την παραπάνω δομή (book), θα μπορούσαμε να δηλώσουμε τις μεταβλητές book_1 και book_2 ως εξής:

```
book book_1, book_2;
```

Νέο Πρόγραμμα

Δημιουργήστε ένα νέο project με τίτλο

struct1

Άσκηση

Στον κώδικα που σας δίνεται, ορίστε μία μεταβλητή **person,** τύπου δομής με τρία πεδία:

- **id : κωδικός**
- **name : όνομα**
- **age : ηλικία**

Αρχικοποιήστε την κατά την δήλωση της (με χρήση των { }) με τα στοιχεία σας και τον κωδικό 10 και τυπώστε την στη οθόνη

Άσκηση

Στο προηγούμενο παράδειγμα ορίστε ένα νέο τύπο δεδομένων (με **typedef**) για την δομή σας με όνομα customer.

Χρησιμοποιήστε τον τύπο customer, για να αρχικοποιήσετε 2 μεταβλητές **person1** και **person2** και τυπώστε τις μεταβλητές.

Καταχωρήστε τις τιμές της person2 στην person1 και τυπώστε την person1

Πίνακες Δομών

Προφανώς μπορούμε να έχουμε διανύσματα που τα στοιχεία τους είναι δομές !! Π.χ.

```
typedef struct {
        char firstName[10];
        char lastName[10];
} personType;

personType aDataBase[100];
```

Διάνυσμα 100 Δομών Τύπου PersonType

Οπότε μπορούμε να γράψουμε

```
aDataBase[30].firstName[0] = '\0'    /* Άδεια συμβολοσειρά */
```