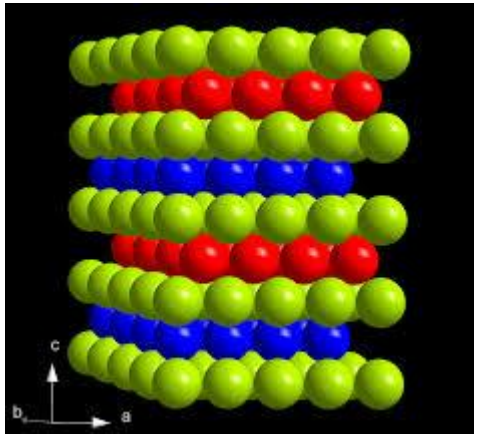
Δομημένος προγραμματισμός

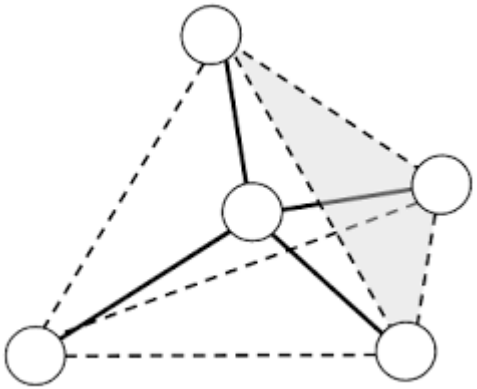


https://en.wikipedia.org/

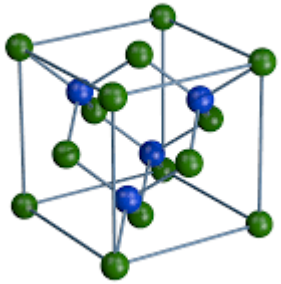
Δομές (structures) στη C

Περίγραμμα της ύλης που θα καλυφθεί

- Οι δομές στη C
  - Δήλωση
  - Προσπέλαση
  - Εκχώρηση τιμών
- Δείκτες σε δομές
- Ένθετες δομές
- Πεδία bit
- Ενώσεις



https://commons.wikimedia.org/



Γιατί δομές;

- Στη καθημερινή μας ζωή υπάρχουν δεδομένα που συσχετίζονται μεταξύ τους.
- Παράδειγμα: δεδομένα για ένα φοιτητή
  - Ονοματεπώνυμο (συμβολοσειρά)
  - Αριθμός Μητρώου (συμβολοσειρά ή ακέραιος)
  - Εξάμηνο σπουδών (ακέραια τιμή)
  - Βαθμοί σε μαθήματα (πίνακας δεκαδικών αριθμών)

  Εδώ έχουμε μια συλλογή στοιχείων τα οποία αφορούν στον ίδιο φοιτητή που όμως στη C αναπαρίστανται με διαφορετικούς τύπους δεδομένων.
- Μέχρι τώρα αφιερώναμε για κάθε στοιχείο έναν τύπο δεδομένων.
- Στη C όμως μπορούμε να δημιουργήσουμε και σύνθετους τύπους δεδομένων. Αυτό υλοποιείται με τις *δομές*.

Οι δομές στη C

- Η δομές είναι σύνθετοι τύποι δεδομένων που αποτελούνται από πολλές διαφορετικού τύπου σχετιζόμενες μεταβλητές.
- Πρόκειται για λογικές ενότητες που ορίζονται από τον προγραμματιστή.
- Για τη δήλωση μιας δομής χρησιμοποιείται η δεσμευμένη λέξη *struct* (structure).
- Η γενική μορφή δήλωσης/ορισμού μιας δομής έχει ως εξής:

```
struct [ονομασία_δομής] {
    <τύπος> ονομασία_μέλους_1;
    <τύπος> ονομασία_μέλους_2;
    ..................
    <τύπος> ονομασία_μέλους_N;
} [μεταβλητή ή λίστα μεταβλητών];
```

Είτε η *ονομασία της δομής* στην αρχή, είτε η *λίστα των μεταβλητών* στο τέλος μπορούν να λείπουν.

Όχι όμως να λείπουν και τα δυο μαζί ταυτόχρονα.

Η *ονομασία μέλους* μπορεί να είναι η ίδια σε διαφορετικές δομές.

Παραδείγματα δήλωσης δομών

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

Εδώ ορίζεται απλά μια δομή (πρότυπο).

Δεν δημιουργείται μεταβλητή.

Μια μεταβλητή τύπου date μπορεί να δημιουργηθεί (δηλωθεί) μετέπειτα στο πρόγραμμα ως εξής:

struct date birthday;

```
struct date {
    int  day;
    int  month;
    int  year;
} start, end;
```

Δήλωση (και δημιουργία) των μεταβλητών (start, end) τύπου date, ταυτόχρονα με τον ορισμό τη δομής. Αργότερα στο πρόγραμμα, μπορούμε αν θέλουμε να δηλώσουμε και άλλη μεταβλητή τύπου date ως εξής:

struct date birthday;

Και άλλα παραδείγματα δήλωσης δομών

```
struct person {
    char fname [15];
    char lname [20];
    char address [25];
    char tel[15];
    int age;
} friend, customer;
```

Δομή με ονομασία person

Μέλη (πεδία) της δομής

Μεταβλητές τύπου person

```
struct student {
    int   id;
    char name [30];
    int semester;
    float mark;
} stud_rec[100];
```

Πίνακας δομών τύπου student

```
struct {
    int  day;
    int  month;
    int  year;
} start_d, finish_d;
```

Εδώ δηλώνονται άμεσα οι μεταβλητές *start_d* και *finish_d*.

Όμως δεν μπορούμε να δηλώσουμε άλλη μεταβλητή ίδιου τύπου στη συνέχεια του προγράμματος.

Θα πρέπει να επαναλάβουμε την περιγραφή της δομής.

Αναφορά/προσπέλαση σε μέλη των δομών

- Πρέπει να καθορίζεται το όνομα της μεταβλητής της δομής, καθώς και το όνομα του μέλους. Χωρίζονται με μια τελεία:

  ονομασία_δομής.ονομασία_μέλους;

- Εκχώρηση τιμών:

```
student.id = 4345;
strcpy(student.name, "MANOLIS");
student.semester=5;
student.mark =67.8;
```

Δομή

Μέλος

```
struct stud {
      int   id;
      char name [30];
      int semester;
      float mark;
 } student, pupil;
```

- Δίνοντας τιμές από το πληκτρολόγιο:

```
printf("Student's ID:"); scanf("%d",&student.id);
printf("Student's name:"); gets(student.name);
printf("Student's semester:"); scanf("%d",&student.semester);
printf("Student's mark:"); scanf("%f",&student .mark);
```

- Αλλά και: *pupil=student; //Η pupil περιέχει τα ίδια με την student.*

Πρόγραμμα καταχώρισης τιμών σε δομή

Δημιουργία μιας δομής στη οποία δίνουμε τιμές από το πληκτρολόγιο

```
#include <stdio.h>
main() {
    struct student{ //δημιουργία της δομής
        char name[50];
        int id;
        float mark;
    } SD;   //Μεταβλητή τύπου struct student
    printf("Enter student info:\n");
    printf("Enter name:"); scanf("%s", SD.name);
    printf("Enter id: "); scanf("%d", &SD.id);
    printf("Enter mark:"); scanf("%f", &SD.mark);
    printf("\nDisplaying Information\n");
    printf("Name: %s\n", SD.name);
    printf("ID: %d\n", SD.id);
    printf("Mark: %.2f\n", SD.mark);
}
```

Εδώ η δομή δηλώνεται έξω από το πρόγραμμα και έχει καθολική εμβέλεια.

```
#include <stdio.h>
struct student {
        char name[50]; //δημιουργία
         int id;                // του τύπου
        float mark;        // της δομής
 };   //χωρίς μεταβλητές
main()
{
    struct student  SD; //μεταβλητή
    printf("Enter student info\n");
    ……..
    ……..
```

Παραδείγματα καταχώρισης τιμών

```
//ορισμός της δομής
struct stud {
     int   id;
     char name [30];
     float mark;
 };
```

```
//Δήλωση της μεταβλητής student
struct stud student ;
.......
//ανάθεση τιμών
 student.id=1;
 strcpy(student.name, "Nikolaou");
 student.mark = 86.5;
```

```
//Δίνουμε αρχικές τιμές κατά τη δήλωση της μεταβλητής
struct stud student = {15, "Athanasakis", 80.5};
```

```
struct stud student ;
......
//δίνουμε τιμές από το πληκτρολόγιο
printf("Student's ID:"); scanf("%d", &student.id);
printf("Student's name:"); scanf("%s",student.name);
printf("Student's mark:"); scanf("%f", &student.mark);
```

Μέγεθος μιας δομής

- Όταν δηλώνεται μια μεταβλητή τύπου δομής, ο compiler δεσμεύει την αντίστοιχη μνήμη.
- Το μέγεθος της δομής σε Bytes υπολογίζεται από τα μεγέθη των μελών της δομής τα οποία εξαρτώνται από τον compiler.
- Για να έχετε την δυνατότητα να μεταφέρετε τον κώδικα σας σε διαφορετικά περιβάλλοντα, είναι ασφαλέστερο να χρησιμοποιείτε τον τελεστή *sizeof* για να εξακριβώνετε το μέγεθος μιας δομής:

length= sizeof(struct student);

- Σε διαφορετικά περιβάλλοντα (λειτουργικά συστήματα, compilers, hardware) τα μεγέθη αλλάζουν.
- Το μήκος της μνήμης (σε Bytes) που δεσμεύεται για μια δομή είναι τουλάχιστον ίσο με το άθροισμα της μνήμης που δεσμεύει κάθε μέλος της δομής.

Ποιο είναι το αποτέλεσμα;

```
#include <stdio.h>
struct student{
        int id;
        char name[50];
        float mark;
    };
main()  {
      struct student s;
      int length;
      length = sizeof (struct student);
      printf("Length of struct student:%d\n", length);
      printf("Length of name in s:%d\n", sizeof(s.name));
      printf("Length of id in s:%d\n", sizeof(s.id));
      printf("Length of mark in s:%d\n", sizeof(s.mark));
}
```

Το πρόγραμμα θα τυπώσει:

Length of struct student:60
Length of name in s:50
Length of id in s:4
Length of mark in s:4

Όμως το άθροισμα των μεγεθών των μελών της δομής είναι:

50+4+4 =58

και όχι *60* όπως δίνει η πρώτη σειρά του αποτελέσματος.

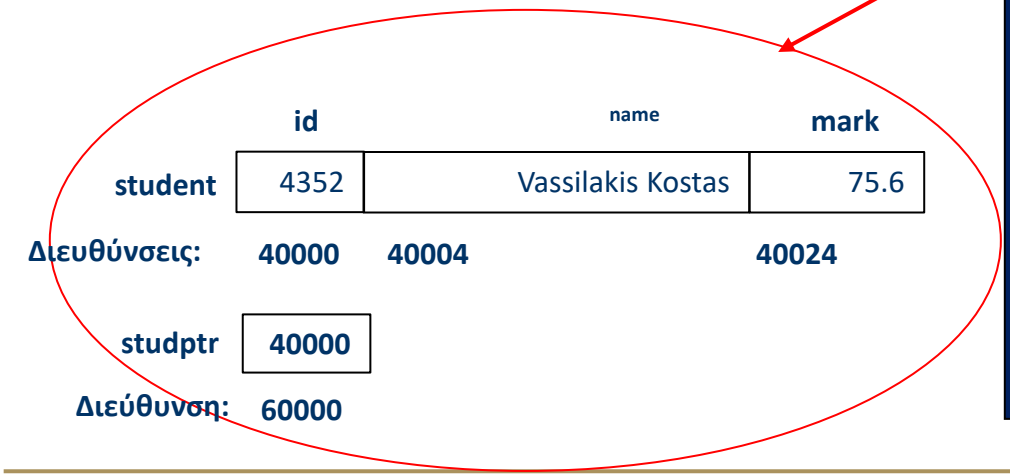
Δηλώσεις δεικτών προς δομές

- Ένας δείκτης σε μια δομή δηλώνεται με τον ίδιο ακριβώς τρόπο όπως σε οποιαδήποτε άλλο τύπο.
- Η προσπέλαση σε μέλος της δομής μέσω δείκτη γίνεται με 2 τρόπους:

  *(*studptr).mark =5.3;*

  studptr->mark =5.3;

- Η αύξηση κατά 1 ενός δείκτη σε δομή, θα αυξήσει τη τιμή του δείκτη όσο το μέγεθος της δομής σε Bytes.




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

struct stud {
        int id;
        char name[20];
        float mark;
    };

main() {
        struct stud student, *studptr;
        student.id=4352;
        strcpy(student.name, "Vassilakis");
        student.mark = 75.6;
        studptr= &student;
        printf("\nDisplaying Information\n");
        printf("Name:%s\n", student.name);
         printf("ID:%d\n", studptr->id);
         printf("Mark:%.2f\n", (*studptr).mark);
}
```

Δήλωση δείκτη σε δομή

Δομές και συναρτήσεις

- Η μεταβίβαση των μελών μιας δομής σε συνάρτηση γίνεται με τον ίδιο τρόπο όπως με οποιονδήποτε τύπο δεδομένων.
- Μπορούμε να μεταβιβάσουμε ολόκληρη τη δομή σε κάποια συνάρτηση σαν τυπική παράμετρο ή όρισμα (κλήση). Με αυτό τον τρόπο μεταβιβάζονται όλα τα μέλη της συνάρτησης.
- Η μεταβίβαση μπορεί να είναι *κατ' αξία* (δεν γίνονται αλλαγές από τη συνάρτηση στα μέλη της δομής) ή *κατ' αναφορά* (αν θέλουμε ν' αλλάξουμε τις τιμές των μελών της δομής).
- Μια συνάρτηση μπορεί να επιστρέψει μια μεταβλητή τύπου δομής.
- Η δήλωση μιας μεταβλητής τύπου δομής, έξω από κάθε συνάρτηση (καθολική μεταβλητή) είναι μια καλή πρακτική, καθώς δεν είναι ανάγκη να μεταβιβάζεται σε συναρτήσεις.
- Μεταβλητές δομών μπορούν να δηλώνονται και τοπικά.

Μεταβίβαση δομής σε συνάρτηση κατ' αξία

```c
#include <stdio.h>
struct student{ //καθολικό αρχέτυπο
        int id;
        char name[30];
        float mark;
    };
//δήλωση συναρτήσεων
struct student read_student();
void print_s (struct student ) ;
main()
{
     struct student student1;
     student1 = read_student();
     print_s (student1);
}
```

Επιστρέφει δομή

```c
struct student read_student ()
{ //διαβάζει μια δομή τύπου student
     struct student student2; //δήλωση δομής
     printf("Student's ID:");
     scanf("%d", &student2.id);
     printf("Student's name:");
     scanf("%s", student2.name);
     printf("Student's mark:");
     scanf("%f", &student2.mark);
     return (student2); //επιστροφή δομής
}

void print_s (struct student student2)
{ //τυπώνει μια δομή τύπου student2
   printf( "ID is: %d\n", student2.id);
   printf( "name is: %s\n", student2.name);
   printf( "mark is: %.2f\n", student2.mark);
}
```

κατ' αξία

Μεταβίβαση δομής σε συνάρτηση κατ' αναφορά

```
#include <stdio.h>
struct student{
        int id;
        char name[30];
        float mark;
    };
void read_student(struct student *);
void print_s (struct student ) ;
main()
{
     struct student student1;
     read_student(&student1);
     print_s (student1);
}
```

Δείκτης σε δομή

```
void read_student(struct student *student2)
{
      printf("Student's ID:");
      scanf("%d", &student2->id);
      printf("Student's name:");
      scanf("%s", student2->name);
      printf("Student's mark:");
      scanf("%f", &student2->mark);
}

void print_s (struct student student2) {
     printf( "ID is: %d\n", student2.id);
     printf( "name is: %s\n", student2.name);
     printf( "mark is: %.2f\n", student2.mark);
}
```

Ένθετες δομές

- Μια δομή μπορεί να είναι μέλος σε μια άλλη δομή.
- Σε αυτή τη περίπτωση καλείται ένθετη δομή

```
struct person {
        char fname [15];
        char lname [20];
        char address [25];
        char tel [15];
        int age;
 };
```

Ένθετη δομή

```
struct student {
        int   id;
        stuct person sperson;
        int semester;
        float mark;
 } stud_rec;
```

Ένθετη δομή

```
struct student {
     int   id;
     struct person {
          char fname [15];
          char lname [20];
          char address [25];
          char tel [15];
          int age;
     } sperson;
     int semester;
     float mark;
 } stud_rec;
```

Η αναφορά/προσπέλαση μελών ένθετης δομής γίνεται από τη εξωτερική δομή:

gets(stud_rec.sperson.fname);

stud_rec.sperson.age=20;

Πρόγραμμα με ένθετες δομές

```
#include <stdio.h>
struct student {
    int   id;
    struct person {
        char lname[20];
        char tel [15];
        int age;
    } sperson;
    int semester;
    float mark;
}  stud_rec = {4356, {"Vassilakis","2810-379803",25}, 5, 7.8};

main() {
     printf("\nStudent ID : %d", stud_rec.id);
     printf("\nStudent Name : %s", stud_rec.sperson.lname);
     printf("\nStudent Mark : %.2f", stud_rec.mark);
}
```

Επίδειξη προγράμματος που χρησιμοποιεί ένθετες δομές.

Ένθετη δομή

Αρχικές τιμές

Πρόγραμμα με διαχείριση πίνακα δομών

```
#include <stdio.h>
#define N 5
struct student {
   int id;
   char name[30];
   float mark;
};
int main() {
   struct student SD[N];
   int i;
   printf("Enter data for students:\n");
   for(i=0;i<N;++i) {
         printf("Enter ID for %d student:", i+1);
         scanf("%d", &SD[i].id);
         printf("Enter name: ");
         scanf("%s", SD[i].name);
         printf("Enter mark: ");
         scanf("%f", &SD[i].mark);
}
```

Πίνακας δομών

Επίδειξη προγράμματος που χρησιμοποιεί πίνακα δομών:

- Ανάθεση τιμών στα πεδία του πίνακα δομών
- Εκτύπωση περιεχομένων των μελών πίνακα δομών

```
printf("\n Displaying data of students:\n");
 for(i=0;i<N;++i) {
         printf("ID: %.d\n", SD[i].id);
         printf("Name: ");  puts(SD[i].name);
         printf("Mark: %.1f\n\n",SD[i].mark);
  }
}
```

Πεδία bit

- Στην C υπάρχει η δυνατότητα να χειριστούμε τα μέλη μιας δομής σαν **bits**.
- Ένα πεδίο (μέλος) bit μπορεί ν' αποτελείται από 1 ή περισσότερα bits.
- Χρήσιμη δυνατότητα για εξοικονόμηση μνήμης και για προγραμματισμό «χαμηλού» επιπέδου.
- Ορισμός μέλους bit:

  <Τύπος> <ονομασία_μέλους>: <μέγεθος>;
- Ο τύπος πρέπει να είναι *int* ή *unsigned*.
- Το μέγεθος καθορίζει το πλήθος των bits.

Το μέγεθος ενός μέλους bit εξαρτάται από τον compiler. Συνήθως οι compilers προσπαθούν ν' αποθηκεύσουν ένα μέλος bit στο μικρότερο δυνατόν χώρο.

```
struct type_byte {
        unsigned first:1
        unsigned mid:6;
        unsigned last:1;
}
```

Δεν υποστηρίζονται πίνακες από bits.

Πρόγραμμα με χρήση των πεδίων bits

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
struct p_status  {
        unsigned  LowInk     : 1;     // 1 Low ink, 0 OK
        unsigned  PaperJam   : 1;   // 1 Paper Jam, 0 OK
        unsigned  EmptyTrayA  : 1;  // 1 Tray A empty, 0 OK
        unsigned  EmptyTrayB  : 1;  // 1 Tray B empty, 0 OK
        unsigned  NeedService : 1;  // 1 needs service, 0 OK
        unsigned  JobsInQueue : 1; // 1 there are waiting jobs, 0 no job
        unsigned  Printing    : 1;      // 1 printing now, 0 not printing
        unsigned  A4_A3       : 1;      // 1 A4 is set on, A3 is set on
    } ;
main(){
   struct p_status P;
   int length;
   srand(time(NULL));
   P.LowInk = rand()%2;
   P.PaperJam = rand()%2;
   P.EmptyTrayA= rand()%2;
   P.EmptyTrayB= rand()%2;
   P.NeedService = rand()%2;
   P.JobsInQueue  = rand()%2;
   P.Printing = rand()%2;
   P.A4_A3 = rand()%2;
```

Έλεγχος εκτυπωτή.
Κάθε πεδίο bit έχει κάποια ένδειξη 0 ή 1.
Τα 0 και 1 παράγονται τυχαία με χρήση της συνάρτησης rand().
Ο έλεγχος γίνεται με το τελεστή ?

Εκτός των stdio.h και stdlib.h, χρειάζεται και η time.h

```
printf("Printer status:\n");
   P.LowInk ? printf("- Ink status LOW!\n"): printf("- Ink status: OK\n");
   P.PaperJam ? printf("- Paper Jam!\n"):printf("- No Paper Jam\n");
   P.EmptyTrayA ? printf("- Tray A: OK\n"): printf("- Tray A empty!\n");
   P.EmptyTrayB ? printf("- Tray B: OK\n"): printf("- Tray B empty!\n");
   P.NeedService ? printf("- Printer needs service\n") : printf("- No service yet\n");
   P.JobsInQueue ?  printf("- There are waiting jobs\n") : printf("- No job in queue\n");
   P.Printing ? printf("- Printing job now\n") : printf("- Waiting for job!\n");
   P.A4_A3 ? printf("- A4 format on\n") : printf("- A3 format on!\n");
   length = sizeof (struct p_status);
   printf("\nLength of struc p_status:%d\n", length);
}
```

Το μέγεθος της δομής

Ενώσεις (unions)

- Μια ένωση (union) είναι ένα κομμάτι μνήμης το οποίο μοιράζονται διαφορετικές μεταβλητές, ακόμα και διαφορετικού τύπου.
- Η μορφή που ορίζεται είναι παρόμοια με αυτή της δομής:

```
union [ονομασία_ένωσης] {
        τύπος  ονομασία_μέλους_1;
        τύπος  ονομασία_μέλους_2;
        ……….
        τύπος  ονομασία_μέλους_N;
} [λίστα μεταβλητών];
```

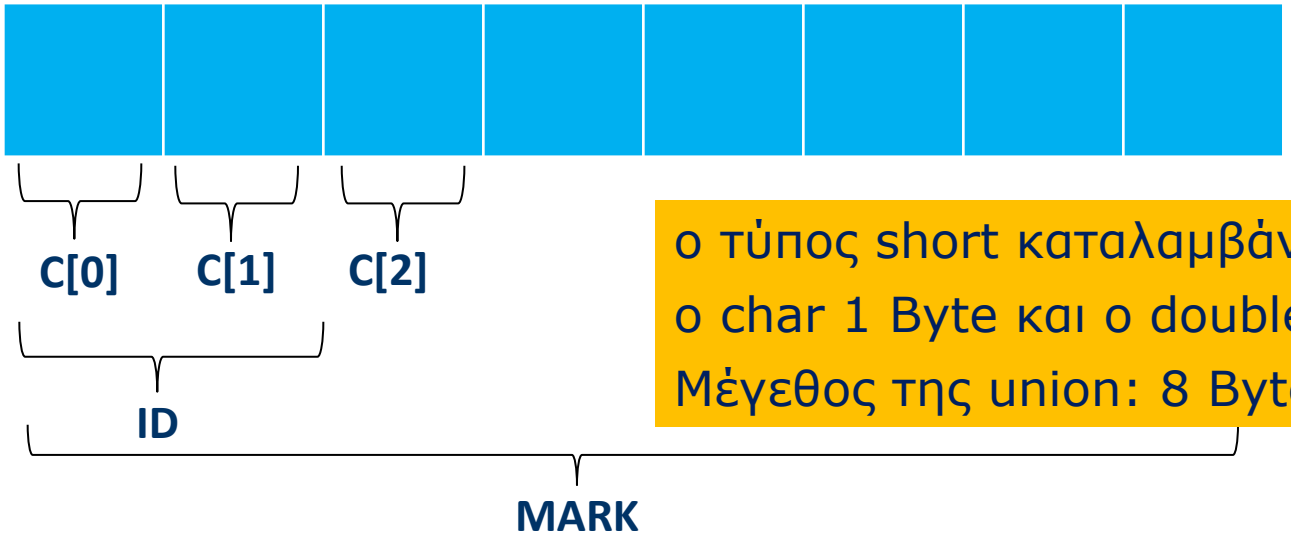
- Το μέγεθος της μνήμης μιας ένωσης είναι σταθερό και αρκετά μεγάλο για να χωρά το μεγαλύτερο μέλος της ένωσης.
- Κάθε φορά χρησιμοποιείται 1 μόνο μέλος της ένωσης.

Παράδειγμα δήλωσης union

```
union  stype {
        short   ID;
        char C[2];
        double MARK;
} sample;
…..
union stype sample1;
```

Η ένωση περιέχει 3 στοιχεία:

- 1 τύπου short (ID)
- 1 τύπου char (ο πίνακας C, 3 χαρακτήρων) και
- 1 τύπου double (MARK)



ο τύπος short καταλαμβάνει 2 Bytes,
ο char 1 Byte και ο double 8 Bytes.
Μέγεθος της union: 8 Bytes

Πρόγραμμα με ένωση

Εμφανίζει την δυαδική αναπαράσταση του χαρακτήρα που δίνεται από το πληκτρολόγιο

```
#include <stdio.h>
#include <stdlib.h>
struct a_byte {
        unsigned  b0 : 1;
        unsigned  b1 : 1;
        unsigned  b2 : 1;
        unsigned  b3 : 1;
        unsigned  b4 : 1;
        unsigned  b5 : 1;
        unsigned  b6 : 1;
        unsigned  b7 : 1;
    } ;
union key_t {
     char ch;
     struct a_byte char_b;
} key1;
```

```
main(){
    int length;
    printf("Press any key...");
    key1.ch = getchar();
    printf("\nInternal representation is:");
    key1.char_b.b7 ? printf("1 ") : printf("0 ");
    key1.char_b.b6 ? printf("1 ") : printf("0 ");
    key1.char_b.b5 ? printf("1 ") : printf("0 ");
    key1.char_b.b4 ? printf("1 ") : printf("0 ");
    key1.char_b.b3 ? printf("1 ") : printf("0 ");
    key1.char_b.b2 ? printf("1 ") : printf("0 ");
    key1.char_b.b1 ? printf("1 ") : printf("0 ");
    key1.char_b.b0 ? printf("1 ") : printf("0 ");
    length = sizeof (union key_t);
    printf("\nLength of union key1:%d\n", length);
}
```

Το μέγεθος της ένωσης

Η typedef

- Με τη λέξη *typdef* μπορούμε στη C να δηλώνουμε ένα *συνώνυμο* σε κάποιο υπάρχοντα τύπο δεδομένων.
- Σύνταξη:

typedef παλαιό_όνομα νέο_όνομα;

- Παράδειγμα:

typedef int akeraios;

- Η *typedef* *δεν προκαλεί απενεργοποίηση* του παλαιού ονόματος.
- Μπορούμε να δημιουργήσουμε πολλά συνώνυμα για ένα τύπο δεδομένων που ήδη υπάρχει (δεν δημιουργούμε νέο!).
- Μας επιτρέπει την εύκολη μετατροπή των προγραμμάτων, όταν αυτά μεταφέρονται σε άλλα περιβάλλοντα με διαφορετικά μεγέθη τύπων δεδομένων.
- Επίσης, δημιουργούμε πιο κατανοητά προγράμματα.

Πρόγραμμα με typedef

```
#include <stdio.h>
typedef struct complex {
    float real;
    float imag;
} migadikos;
complex add(complex n1,complex n2);
int main(){
    migadikos C1, C2, NC;
    printf("1st complex number - real part: ");
    scanf("%f",&C1.real);
    printf("1st complex number - imaginary: ");
    scanf("%f",&C1.imag);
    printf("\n2nd complex number - real part: ");
    scanf("%f",&C2.real);
    printf("2nd complex number - imaginary: ");
    scanf("%f",&C2.imag);
    NC=add(C1, C2);
   printf("\nSum=%.1f+%.1fi",NC.real,NC.imag);
}
```

Προσθέτει 2 μιγαδικούς

```
complex add(complex n1, complex n2)
{
        complex temp;
        temp.real=n1.real+n2.real;
        temp.imag=n1.imag+n2.imag;
        return(temp);
}
```

Δομές που αναφέρονται στον εαυτό τους

- Δεν μπορεί μια δομή ν' αναφέρεται στον εαυτό της, διότι δυνητικά θα είχαμε άπειρες ένθετες δομές (compiler error)

```
struct person {
    short ID;
    char C[2];
    struct person dhead;
};
```

- Η αυτό-αναφορά όμως είναι δυνατή, αν αυτή γίνεται από δείκτη στη δομή.

```
struct person {
    short ID;
    char C[2];
    struct person *dhead;
};
```

Περισσότερη εξάσκηση

- Λύστε τη σειρά των ασκήσεων που έχουν αναρτηθεί στο δικτυακό τόπο του μαθήματος.