Κεφάλαιο V:

Δομές και ενώσεις.

5.1 Δομές.

Όπως αναφέραμε στο προηγούμενο κεφάλαιο η ομαδοποίηση της πληροφορίας στον προγραμματισμό είναι ιδιαίτερα σημαντική. Ένα παράδειγμα ομαδοποίησης της πληροφορίας αποτελούν οι πίνακες οι οποίοι αποτελούνται από στοιχεία του ιδίου τύπου. Είδαμε επίσης πόσο απλά μπορούμε να διαχειριστούμε τα δεδομένα ενός πίνακα.

Το ερώτημα το οποίο τίθεται είναι εάν υπάρχει η δυνατότητα στη C να ορίσουμε οντότητες οι οποίες όμως να αποτελούνται από στοιχεία διαφορετικών τύπων δεδομένων. Για παράδειγμα έστω πως έχουμε το ακόλουθο σύνολο μεταβλητών:

```
float a;        /* Μεταβλητή τύπου float */
int b;          /* Μεταβλητή τύπου int */
float c[3];     /* Πίνακας τύπου float με τρία στοιχεία */
char d[3];      /* Συμβολοσειρά */
a=7.1;
b=10;
c[0]=2.1;
c[1]=4.5;
c[2]=6.8;
d="Hi";
```

Μπορούμε να ορίσουμε μια οντότητα που να έχει μέλη τις παραπάνω μεταβλητές, παρ' ότι αυτές είναι διαφορετικού τύπου μεταξύ τους; Η απάντηση είναι ναι. Τέτοιες οντότητες στην C ονομάζονται **δομές**. Ας δούμε πως μπορούμε να κατασκευάσουμε μια δομή:

- Πρώτα οφείλουμε να ορίσουμε την *περιγραφή της δομής* που περιλαμβάνει τον σχεδιασμό της. Για το παραπάνω παράδειγμα έχουμε:

```
struct my_example{
    float a;
    int b;
    float c[3];
    char d[3];
};
```

Στον ορισμό της δομής οφείλουμε να χρησιμοποιήσουμε ένα όνομα της αρεσκείας μας (ακολουθώντας πάντα τους κανόνες ονοματολογίας που έχουμε αναφέρει). Στον παραπάνω ορισμό χρησιμοποιήσαμε το όνομα *my_example*. Στη συνέχεια μέσα σε αγκύλες αναγράφουμε τα στοιχεία της δομής τα οποία ονομάζονται *μέλη*.

- Το δεύτερο βήμα είναι η δήλωση (κατασκευή μια δομής) με βάση τον παραπάνω ορισμό. Αυτό γίνεται ως εξής:

  ```
  struct my_example s1;
  ```

  Στην παραπάνω γραμμή δηλώσαμε (κατασκευάσαμε) την δομή s1 με βάση τον ορισμό μας στο πρώτο βήμα. Οι δομές που δηλώνουμε μπορούν να έχουν ονόματα της αρεσκείας μας ακολουθώντας πάντα τους κανόνες ονοματολογίας που έχουμε αναφέρει.
- Το τρίτο βήμα είναι η αρχικοποίηση των μελών της δομής. Για το παραπάνω παράδειγμα έχουμε:

  ```
  s1.a=7.1;
  s1.b=10;
  s1.c[0]=2.1;
  s1.c[1]=4.5;
  s1.c[2]=6.8;
  s1.d=”Hi”;
  ```

  Όπως παρατηρούμε για να αναφερθούμε στα μέλη της δομής, αυτό γίνεται με το όνομα της δομής και το όνομα του μέλους διαχωριζόμενα από μια τελεία (δηλαδή *όνομα_δομής.όνομα_μέλους*). Για τα μέλη κάθε δομής ισχύουν ότι και για τις απλές μεταβλητές. Μπορούν να εμπλακούν σε υπολογισμούς, να εκτυπωθούν μέσω της printf(), να εισαχθούν σε αυτά δεδομένα μέσω της scanf(), να περάσουν ως ορίσματα σε συναρτήσεις κτλ.

Ένας άλλος τρόπος αρχικοποίησης των μελών της δομής μπορεί να γίνει κατά την δήλωσή της και εικονίζεται στην παρακάτω γραμμή:

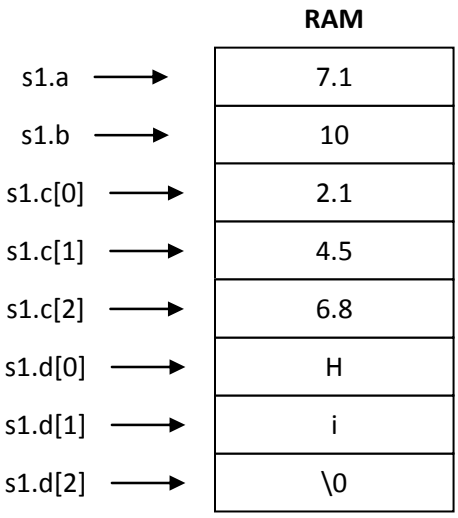
```
struct my_example s1={7.1, 10, 2.1, 4.5, 6.8, “Hi” };
```

Όπως παρατηρούμε τα δεδομένα μπαίνουν μέσα σε αγκύλες και αρχικοποιούν τα μέλη της δομής με την σειρά ακριβώς που αυτά εμφανίζονται στην περιγραφή της. Σημειώνουμε επίσης πως ο ορισμός και η δήλωση μιας δομής μπορεί να γίνει ταυτόχρονα με τον ακόλουθο τρόπο:

```
struct my_example{
    float a;
    int b;
    float c[3];
    char d[3];
}s1;
```

Οι δομές αποτελούν σημαντικές οντότητες στον προγραμματισμό σε C. Με τις δομές ο προγραμματιστής έχει την δυνατότητα να ομαδοποιεί την πληροφορία και ουσιαστικά να ορίζει και να δημιουργεί τους δικούς του τύπους δεδομένων που χρειάζεται στο πρόγραμμά του.

Στο παρακάτω σχήμα εικονίζεται η κατάταξη της πληροφορίας στην μνήμη του υπολογιστή για την συγκεκριμένη δομή s1 την οποία δημιουργήσαμε και αρχικοποιήσαμε παραπάνω. Η πληροφορία κατατάσσεται ακριβώς σειριακά.



Παράδειγμα: Να αναπτύξετε ένα πρόγραμμα στο οποίο να ορίσετε μια δομή που να περιγράφει σημεία στο επίπεδο. Με βάση τον ορισμό να δηλώσετε δύο δομές και να τις αρχικοποιήσετε για τα σημεία Α(2,3) και Β(4,5). Στη συνέχεια να υπολογίσετε και να εκτυπώσετε την απόσταση κάθε σημείου από την αρχή των αξόνων και την απόσταση μεταξύ τους.

Ένα σημείο στο επίπεδο ορίζεται με δύο πραγματικούς αριθμούς που αποτελούν τις συντεταγμένες του. Άρα ο ορισμός της δομής που περιγράφει σημεία στο επίπεδο μπορεί να είναι ο ακόλουθος:

```
struct simeio{
    double x;
    double y;
};
```

Μία λύση του παραπάνω προβλήματος είναι η ακόλουθη:

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

int main(void){

    struct simeio{        /* Ορισμός δομής σημείου στο επίπεδο */
        double x;
        double y;
    };

    struct simeio A={2., 3.};      /* Δήλωση-αρχικοποίηση σημείου Α */
    struct simeio B={4.,5.};       /* Δήλωση-αρχικοποίηση σημείου Β */
    double ra, rb, rab;            /* Δήλωση μεταβλητών ra, rb, rab */
```

```
    ra=sqrt(pow(A.x,2.)+pow(B.y,2.));          /* Υπολογισμός απόστασης σημείου Α */
    rb=sqrt(pow(B.x,2.)+pow(B.y,2.));          /* Υπολογισμός απόστασης σημείου Β */
    rab=sqrt(pow(B.x-A.x,2.)+pow(B.y-A.y,2.));     /* Υπολογισμός απόστασης σημείων Α-Β */

    printf("Apostasi simeiou A apo arxi=%f \n",ra);     /* Εκτύπωση αποτελεσμάτων */
    printf("Apostasi simeiou B apo arxi=%f \n",rb);
    printf("Apostasi simeion A-B =%f \n",rab);

    return 0;
}
```

Παράδειγμα: Να ορίσετε μια δομή που να περιγράφει μιγαδικούς αριθμούς. Με βάση τον ορισμό να δηλώσετε δύο μιγαδικούς αριθμούς τους a=3+5i και b=5-8i.

```
struct migadikos{          /* Ορισμός δομής μιγαδικών */
    double re;
    double im;
};
struct migadikos a={3., 5.};       /* Δήλωση-αρχικοποίηση μιγαδικού a */
struct migadikos b={5., -8};       /* Δήλωση-αρχικοποίηση μιγαδικού b */
```

Παράδειγμα: Να ορίσετε μια δομή που να περιγράφει τους φοιτητές του Τμήματος Φυσικής. Δημιουργείστε έναν φοιτητή με στοιχεία "Anagnostou", "Dimitrios" με αριθμό μητρώου 3241.

Κάθε φοιτητής έχει επώνυμο, όνομα και αριθμό μητρώου. Άρα ο ορισμός της δομής που περιγράφει φοιτητές μπορεί να είναι ο ακόλουθος:

```
struct foititis{                    /* Ορισμός δομής φοιτητών */
    char eponimo[20];
    char onoma[20];
    int AM;
};
struct foititis a={ "Anagnostou", "Dimitrios", 3241};
```

Στον παραπάνω κώδικα χρησιμοποιήσαμε για το επώνυμο και για το όνομα συμβολοσειρές με μέγεθος 20, ενώ για τον αριθμό μητρώου μία μεταβλητή τύπου int.

5.4 Πίνακες Δομών.

Στη C έχουμε την δυνατότητα να ορίσουμε πίνακες των οποίων τα στοιχεία είναι δομές. Ως παράδειγμα ας θεωρήσουμε την παραπάνω δομή η οποία περιγράφει φοιτητές. Με βάση αυτόν τον ορισμό μπορούμε να δηλώσουμε έναν πίνακα δομών ως εξής:

```
struct foititis{                    /* Ορισμός δομής φοιτητών */
    char eponimo[20];
    char onoma[20];
    int AM;
};
struct foititis A[200];
```

Ο παραπάνω πίνακας A[200] είναι ένας πίνακας δομών με διακόσια στοιχεία. Κάθε στοιχείο είναι δομή και αντιπροσωπεύει έναν φοιτητή. Για παράδειγμα ο πρώτος φοιτητής αντιστοιχεί στα μέλη:

A[0].eponymo
A[0].onoma
A[0].AM

Ο δεύτερος φοιτητής στα μέλη:

A[1].eponymo
A[2].onoma
A[3].AM

και ου το καθεξής. Όπως καταλαβαίνουμε ένας πίνακας δομών σαν τον παραπάνω μπορεί να θεωρηθεί ως μια πολύ απλή βάση δεδομένων.

5.3 Δομές και συναρτήσεις.

Ας δούμε με ένα απλό παράδειγμα πως μπορούμε να περνάμε δομές ως ορίσματα σε μία συνάρτηση. Στο παρακάτω πρόγραμμα χρησιμοποιούμε την δομή migadikos που αναπτύξαμε σε προηγούμενη παράγραφο, με βάση την οποία μπορούμε να ορίσουμε μιγαδικούς αριθμούς στα προγράμματά μας. Στη συνέχεια αναπτύσσουμε μία συνάρτηση η οποία απλά τυπώνει έναν μιγαδικό αριθμό που παίρνει ως όρισμα. Την συνάρτηση αυτή την καλούμε μέσα από την main().

```
#include<stdio.h>
struct migadikos{          /* Ορισμός δομής μιγαδικών*/
    double re;
    double im;
};
void fx(struct migadikos b);       /* Δήλωση πρωτότυπο της συνάρτησης */

int main(void){
    struct migadikos a={2., 3.};          /* Δήλωση-αρχικοποίηση δομής a */
    fx(a);                                /* Κλήση της συνάρτησης fx με όρισμα τον a*/
    return 0;
}

void fx(struct migadikos b){              /* Γενική συνάρτηση εκτύπωσης μιγαδικού */
    printf("(%f)+i(%f)",b.re, b.im);
}
```

Όπως παρατηρούμε στο παραπάνω πρόγραμμα ορίζουμε την δομή που περιγράφει μιγαδικούς αριθμούς ψηλά-ψηλά στο πρόγραμμά μας και πριν από κάθε συνάρτηση. Με αυτό τον τρόπο κοινοποιούμε τον ορισμό της σε όλες τις συναρτήσεις που ακολουθούν. Στην συνέχεια αναφερόμαστε στη δήλωση πρωτότυπο της συνάρτησης μας. Στην συνάρτηση main() δηλώνουμε και αρχικοποιούμε έναν μιγαδικό τον a. Στο επόμενο βήμα καλούμε την συνάρτηση με όρισμα τον συγκεκριμένο μιγαδικό a. Η εκτέλεση του προγράμματος μεταφέρεται τώρα στην συνάρτηση:

```
void fx(struct migadikos b)
```

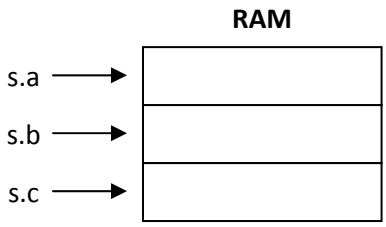
Η συνάρτηση αυτή παίρνει ένα όρισμα το οποίο είναι μιγαδικός. Είναι τύπου void γιατί δεν επιστρέφει κάποια τιμή αλλά απλά τυπώνει τον μιγαδικό με την συνάρτηση printf() που περιέχει.

5.4 Ενώσεις.

Οι ενώσεις είναι οντότητες οι οποίες μοιάζουν σε όλα με τις δομές εκτός από την διαχείριση μνήμης που κάνουν για την αποθήκευση των μελών τους. Ας δώσουμε ένα από παράδειγμα δομής και ένωσης και ας δούμε την διαφορά τους:

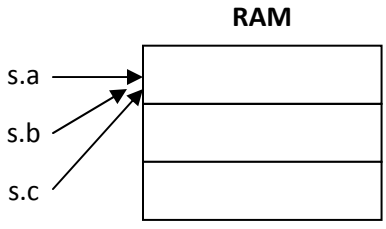
Ορισμός Δομής

```
struct various{
    float a;
    int b;
    char c;
};
struct various s;
```



Ορισμός Ένωσης

```
union various{
    float a;
    int b;
    char c;
};
union various s;
```



Όπως παρατηρούμε μία ένωση ορίζεται με το όνομα **union** αντί του struct. Το κάθε μέλος της δομής καταλαμβάνει τον ξεχωριστό χώρο στη μνήμη του υπολογιστή που του αναλογεί. Σε αντίθεση τα μέλη μιας ένωσης καταλαμβάνουν κοινό χώρο (δεσμεύεται χώρος κατάλληλος ώστε να είναι δυνατόν να αποθηκευθεί το μεγαλύτερο μέλος). Αυτό έχει ως αποτέλεσμα αφ ενός οικονομία στην χρήση της μνήμης του υπολογιστή αφετέρου όμως κάθε φορά μόνο ένα μέλος της ένωσης κάθε φορά περιέχει

δεδομένα. Οι ενώσεις γενικά είναι πολύ πιο δύσχρηστες από τις δομές και χρησιμοποιούνται σπάνια μόνο σε περιπτώσεις όπου ο προγραμματιστής θέλει να κάνει οικονομία στην μνήμη του υπολογιστή. Ειδικότερα σήμερα δεν χρησιμοποιούνται πολύ μιας και οι σύγχρονοι υπολογιστές διαθέτουν πολύ μεγάλη μνήμη.