Η ΓΛΩΣΣΑ C

Μάθημα 13:

Δομές

Δημήτρης Ψούνης

Περιεχόμενα Μαθήματος

Α. Δομές

1. Τι είναι η δομή;

- Δομή είναι η ομαδοποίηση δύο ή περισσότερων μεταβλητών με ένα κοινό όνομα για ευκολότερο χειρισμό.
  - Σε μία δομή μπορούμε να έχουμε μεταβλητές που έχουν διαφορετικό τύπο, ακόμη και πινάκων ή και άλλων δομών.

- Ένα χαρακτηριστικό παράδειγμα είναι το εξής:
  - Έστω ότι θέλουμε να απεικονίσουμε στην μνήμη ένα σημείο του επιπέδου.
  - Ένα σημείο του επιπέδου καθορίζεται από τις δύο συντεταγμένες του. Έτσι για να το απεικονίσουμε στη μνήμη έχουμε πλέον δύο τρόπους:
    - Είτε θα έχουμε δύο ξεχωριστές μεταβλητές (ακέραιες) που αποθηκεύουν τις συντεταγμένες
    - Είτε θα κατασκευάσουμε μια δομή τύπου simeio (το όνομα το καθορίζουμε εμείς) που θα περικλείει τις δύο μεταβλητές
      - Ο 2ος τρόπος είναι καλύτερος γιατί πάντα ένα σημείο καθορίζεται από αυτές τις δύο πληροφορίες συνεπώς επιτυγχάνουμε αφαίρεση στα δεδομένα.

Α. Δομές

2. Πως ορίζουμε μια δομή

- Αφού αποφασίσουμε τι δεδομένα θα έχει η δομή μας, γράφουμε την εντολή δήλωσης δομής:

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

- Δηλαδή ακολουθούμε το πρότυπο:

```
struct ΟΝΟΜΑ{
   ΤΔ1 ον_μετ1;
   ΤΔ2 ον_μετ2;
   ...
   ΤΔn ον_μετ_n;
};
```

  - Το struct είναι λέξη κλειδί που ορίζει ότι δηλώνουμε δομή.
  - Το ΟΝΟΜΑ της δομής το επιλέγουμε εμείς.
  - Το ΤΔ1,ΤΔ2,...,ΤΔn είναι τύπος δεδομένων της αντίστοιχης μεταβλητής.
  - Τα ονόματα των μεταβλητών – μελών της δομής τα επιλέγουμε εμείς.

Α. Δομές

3. Πως χρησιμοποιούμε τη δομή

- Ο ορισμός της δομής γράφεται πριν από την main και πριν από τα πρωτότυπα των συναρτήσεων.
- Από την στιγμή που την ορίσουμε, η δομή συμπεριφέρεται σαν οποιοσδήποτε τύπος δεδομένων.
  - Αυτό σημαίνει ότι μπορώ να ορίσω μεταβλητές που έχουν τον τύπο δεδομένων της δομής (αναφέρονται και σαν στιγμιότυπα της δομής)
- Έτσι στις συναρτήσεις (και στην main) μπορώ να δηλώσω στιγμιότυπα της δομής με εντολή δήλωσης:

```
struct simeio a;
```

  - Ενώ έχω πρόσβαση στα μέλη της δομής με τον τελεστή τελεία (.) π.χ.:

```
a.x=5;
a.y=10;
```

- Μπορώ να δηλώσω πολλά στιγμιότυπα της ίδιας δομής ως ξεχωριστές μεταβλητές π.χ. με μια εντολή δήλωσης:

```
struct simeio a,b,c;
```

  - Και σε κάθε μία από αυτές έχω πρόσβαση στα μέλη της με τον τελεστή τελεία (.) π.χ.
  - a.x=10; a.y=5; b.x=4; b.y=9; c.x=12; c.y=18;

Α. Δομές

3. Πως χρησιμοποιούμε τη δομή

- Υπάρχει ένας ακόμη τρόπος να δηλώσουμε μια δομή και να δηλώσουμε αμέσως και κάποια στιγμιότυπα του τύπου της δομής. Π.χ.:

```
struct simeio{
   int x;
   int y;
} a,b,c;
```

  - Όπου δηλώνουμε και την δομή και ταυτόχρονα δηλώνουμε και τα στιγμιότυπα της δομής a,b,c τα οποία μπορούμε να τα χρησιμοποιήσουμε όπως είδαμε πριν.
- Οι δομές είναι μία συλλογή μεταβλητών, άρα μπορούμε να ορίσουμε σε αυτές οποιονδήποτε τύπο δεδομένων. Π.χ. ας δούμε την επόμενη δομή που θα μπορούσε να αποθηκεύσει τα δεδομένα ενός ατόμου:

```
struct atomo{
   char onoma[30];
   char eponimo[30];
   int ilikia;
   float misthos;
};
```

  - Μέσω της οποίας μπορώ να δηλώσω ένα ή περισσότερα στιγμιότυπα ατόμων, κάθε ένα από τα οποία θα έχει τις δικές του μεταβλητές στις οποίες έχω πρόσβαση με την (.)

Α. Δομές

4. Αρχικοποίηση Δομής

- Η αρχικοποίηση μια δομής μπορεί να γίνει με τον τρόπο που μάθαμε, δηλ. με μία εντολή δήλωσης και εντολές καταχώρησης στην συνέχεια

```
struct simeio a;

a.x=5;
a.y=10;
```

- Υπάρχει και εντολή δήλωσης και αρχικοποίησης ως εξής:

```
struct simeio a = {5,10}
```

  - (υπάρχει μία ένα προς ένα απεικόνιση της αρχικοποίησης με τις σταθερές που γράφουμε στα άγκιστρα).
  - Έτσι ένα στιγμιότυπο τύπου atomo θα αρχικοποιείται π.χ. ως εξής:

```
struct atomo sb = {"John","Doe",34,890.44};
```

Α. Δομές

5. Δομές που περιέχουν Δομές

- Κάθε δομή, από την στιγμή που την ορίσουμε, είναι ένας νέος τύπος δεδομένων.
- Άρα από την στιγμή που μια δομή έχει ως μέλη της μεταβλητές οποιουδήποτε τύπου δεδομένων, μία δομή μπορεί να περιέχει ως μέλη της δομές!
  - Δείτε για παράδειγμα τον ορισμό της δομής τρίγωνο που ορίζεται από τρία σημεία (τις τρείς κορυφές του)

```
struct trigono{
   struct simeio A;
   struct simeio B;
   struct simeio C;
};
```

  - Πως θα δηλώσω ένα στιγμιότυπο τύπου τρίγωνο; Με την εντολή

```
struct trigono TR;
```

  - Ενώ θα έχω πρόσβαση π.χ. στη x συντεταγμένη του σημείου με την εντολή:
(δείτε την διπλή χρήση του τελεστή .)

```
TR.A.x=5;
```

Α. Δομές
6. Ένα πλήρες παράδειγμα

- Το ακόλουθο παράδειγμα δείχνει την βασική χρήση μιας δομής:

```
/* struct.c Deixnei tin vasiki leitourgia mias domis*/
#include <stdio.h>

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

struct person{
   char name[80];
   char surname[80];
   struct date gennisi;
};
```

(συνεχίζεται…)

Α. Δομές

6. Ένα πλήρες παράδειγμα

```
main()
{
   struct person p;
   printf("Dwse to onoma: ");
   scanf("%s", p.name);

   printf("Dwse to eponimo: ");
   scanf("%s", p.surname);

   printf("Dwse imera gennisis: ");
   scanf("%d",&p.gennisi.day);

   printf("Dwse mina gennisis: ");
   scanf("%d",&p.gennisi.month);

   printf("Dwse etos gennisis: ");
   scanf("%d",&p.gennisi.year);

   printf("\n%s %s (%d/%d/%d)", p.name, p.surname,
            p.gennisi.day,p.gennisi.month,p.gennisi.year);
}
```

Β. Προγραμματιστικές Τεχνικές στις Δομές

1. Ο τελεστής καταχώρησης

- Πρέπει να σημειώσουμε ότι οι μεταβλητές μιας δομής αποθηκεύονται σε διαδοχικές θέσεις της μνήμης.
- Έχουμε δικαίωμα να καταχωρήσουμε ένα στιγμιότυπο μιας δομής σε ένα άλλο στιγμιότυπο.
  - Πρέπει να ξέρουμε ότι δημιουργείται bit προς bit αντίγραφο!
- Έτσι π.χ. το ακόλουθο τμήμα κώδικα:

```
struct point a;
struct point b;

a.x=10;
a.y=5;

b=a;

printf("%d %d",b.x,b.y);
```

- Θα εκτυπώσει στην οθόνη 10 5 (δηλαδή θα δημιουργηθεί ακριβές αντίγραφο του a στην μεταβλητή b, σαν να γράφαμε τις εντολές:

```
b.x=a.x;
b.y=a.y;
```

Β. Προγραμματιστικές Τεχνικές στις Δομές

2. Δομές και Πίνακες

- Είδαμε ήδη ότι μία δομή μπορεί να περιέχει ως μέλη της πίνακες.

- Μπορούμε επίσης να ορίσουμε πίνακα του οποίου τα στοιχεία θα είναι στιγμιότυπα μιας δομής.
  - Π.χ. μπορούμε να ορίσουμε έναν πίνακα ατόμων ως εξής:

```
struct atomo pinakas[10];
```

  - Και έχουμε 10 μεταβλητές τύπου struct atomo στις οποίες έχουμε πρόσβαση ως pinakas[0], pinakas[1],…,pinakas[9], στα μέλη των οποίων έχουμε πρόσβαση με τον τελεστή τελεία (.)
  - Π.χ. Η εντολή

```
pinakas[1].ilikia=52;
```

    - Αποθηκεύει στο 2ο άτομο την ηλικία 52.
  - Ενώ με την εντολή

```
pinakas[3].onoma[4]='c';
```

    - Αποθηκεύει στον 5ο χαρακτήρα του ονόματος του 4ου ατόμου το c

Β. Προγραμματιστικές Τεχνικές στις Δομές

2. Δομές και Πίνακες

- Επίσης μπορούμε να δεσμεύσουμε δυναμικά τον χώρο μνήμης για τον πίνακα όπως είδαμε στο προηγούμενο μάθημα.
- Δείτε ότι στον τελεστή sizeof διοχετεύουμε ως όρισμα το όνομα της δομής.
- Έτσι η προηγούμενη δήλωση μπορεί να γίνει με malloc και free ως εξής:

```
struct atomo *pinakas;

pinakas=malloc(sizeof(struct atomo)*10);

if (pinakas==NULL)
{
    printf("Adynamia desmeysis mnimis");
    exit(0);
}

...χρήση των pinakas[0],...,pinakas[9]...

free(pinakas);
```

Β. Προγραμματιστικές Τεχνικές στις Δομές

2. Δομές και Πίνακες

- Μεταγλωττίστε, εκτελέστε και μελετήστε το πρόγραμμα struct_array.c το οποίο
  - Ζητά από τον χρήστη να εισάγει το πλήθος των επαφών του.
  - Έπειτα δεσμεύει δυναμικά έναν πίνακα από επαφές.
  - Ζητά από τον χρήστη να εισάγει τις επαφές του και τις αποθηκεύει στον πίνακα επαφών.
  - Τυπώνει τις επαφές.
  - Απελευθερώνει την μνήμη.

Β. Προγραμματιστικές Τεχνικές στις Δομές

2. Δομές και Πίνακες

```
/* struct_array.c: Epeksergasia enos pinaka apo stigmiotupa domis */

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

#define STRING_SIZE 100

struct epafi {
        char name[STRING_SIZE];
        char phone[STRING_SIZE];
};

main()
{
        struct epafi *pinakas;
        int i,N;

        /* 1. Eisagwgi tis diastasis toy pinaka */
        printf("Dwse plithos epafwn: ");
        scanf("%d",&N);
```

(συνεχίζεται…)

Β. Προγραμματιστικές Τεχνικές στις Δομές

2. Δομές και Πίνακες

```
/* 2. Desmeusi Mnimis */
pinakas=malloc(sizeof(struct epafi)*N);
if (!pinakas)
{
        printf("Adynamia desmeusis mnimis");
        exit(0);
}

/*3. Diavasma epafwn */
fflush(stdin);
for (i=0; i<N; i++)
{
        printf("Dwse onomatepwnimo %d-ou atomou: ",i+1);
        gets(pinakas[i].name);
        printf("Dwse tilefwno %d-ou atomou: ",i+1);
        gets(pinakas[i].phone);
}
```

(συνεχίζεται…)

Β. Προγραμματιστικές Τεχνικές στις Δομές

2. Δομές και Πίνακες

```
        /*4. Ektypwsi epafwn */
        printf("\n\nEPAFES\n==============");
        for (i=0; i<N; i++)
        {
                printf("\n%d) %s (%s) ",i+1,
                        pinakas[i].name,
                        pinakas[i].phone);
        }

        /*5. Apodesmeusi Mnimis */
        free(pinakas);
}
```

Β. Προγραμματιστικές Τεχνικές στις Δομές

3. Δομές και Δείκτες

1. Δομές με Δείκτες

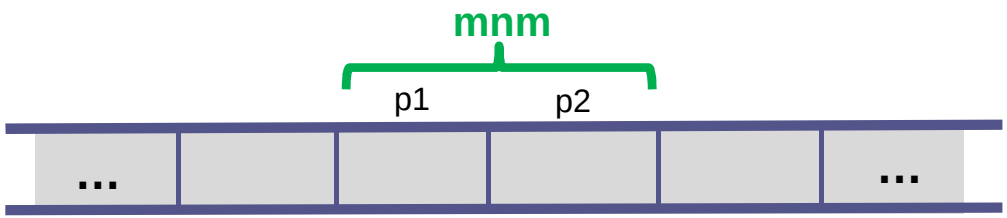
- Ήδη είδαμε ότι μία δομή που ορίζουμε είναι ένας τύπος δεδομένων, άρα μπορούμε να ορίσουμε και δείκτη που δείχνει σε στιγμιότυπο της δομής.
- Επίσης έχουμε κάθε δυνατότητα να ενσωματώσουμε ως μέλη μιας δομής και δείκτες.
- Για παράδειγμα μπορούμε να ορίσουμε την δομή:

```
struct minima{
   char *p1;
   char *p2;
};
```

- Ειδικά οι δομές που περιέχουν δείκτες θέλουν προσοχή στην διαχείριση.
- Αν ορίσουμε ένα στιγμιότυπο της δομής π.χ. με την εντολή

```
struct minima mnm1;
```

- τότε απλά δεσμεύεται χώρος για δύο δείκτες.



Β. Προγραμματιστικές Τεχνικές στις Δομές

3. Δομές και Δείκτες

1. Δομές με Δείκτες

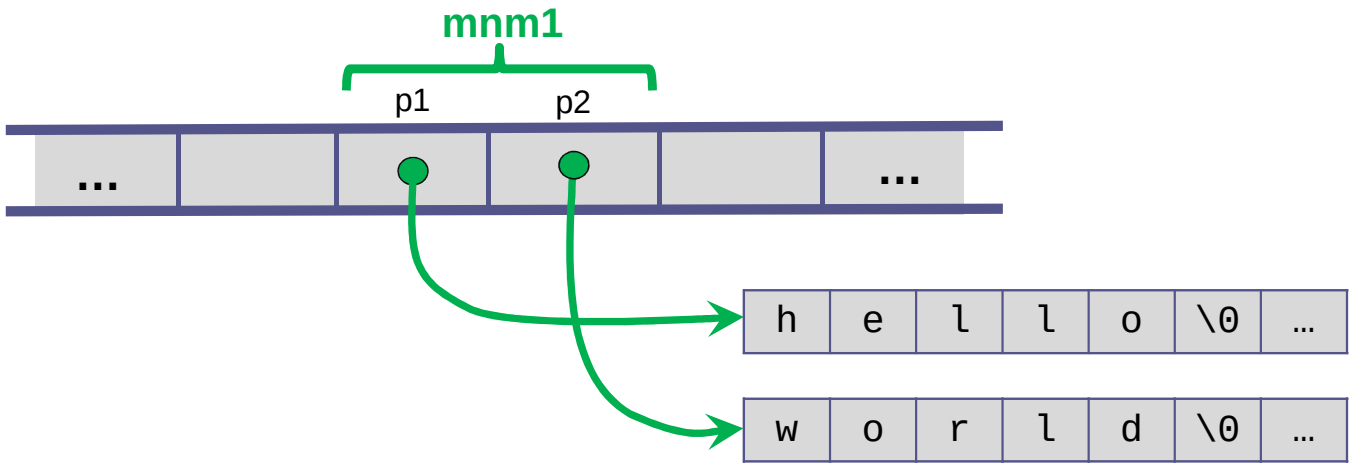
- Είναι δική μας προγραμματιστική υποχρέωση να δεσμεύσουμε χώρο για τους δείκτες, με κατάλληλες εντολές malloc:

```
mnm1.p1=malloc(10*sizeof(char));
mnm1.p2=malloc(20*sizeof(char));
```

- Τις οποίες μπορούμε να αρχικοποιήσουμε κατάλληλα π.χ.:

```
strcpy(mnm1.p1, "hello");
strcpy(mnm1.p2, "world");
```

- Θα δεσμευτεί και θα αρχικοποιηθεί ο κατάλληλος χώρος και η εικόνα της μνήμης θα είναι:



Β. Προγραμματιστικές Τεχνικές στις Δομές

3. Δομές και Δείκτες

1. Δομές με Δείκτες

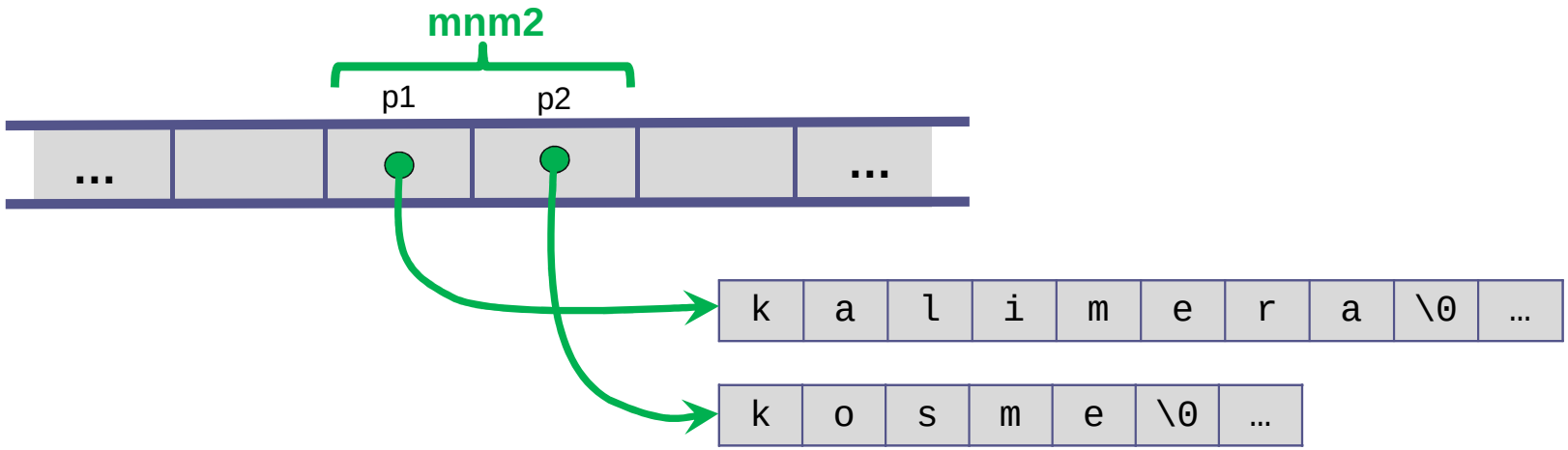
- Αν δηλώσουμε 2ο στιγμιότυπο με τον ίδιο τρόπο και την αρχικοποιήσουμε κατάλληλα με αντίστοιχες εντολές, π.χ.:

```
struct minima mnm2;

mnm2.p1=malloc(10*sizeof(char));
mnm2.p2=malloc(20*sizeof(char));

strcpy(mnm2.p1, "kalimera");
strcpy(mnm2.p2, "kosme");
```

- Η εικόνα της μνήμης για την νέα μεταβλητή θα είναι:



Β. Προγραμματιστικές Τεχνικές στις Δομές

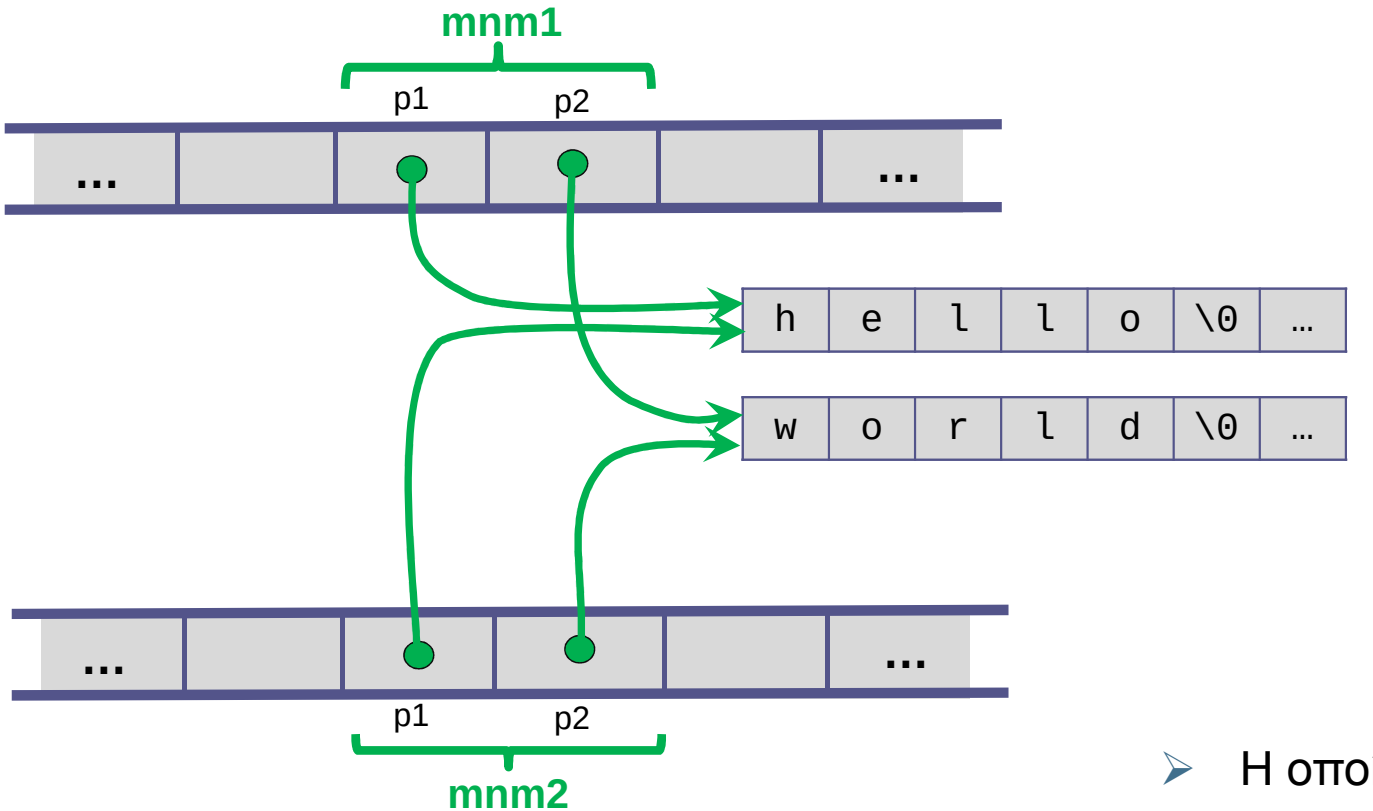
3. Δομές και Δείκτες

1. Δομές με Δείκτες

- Προσοχή όμως. Τι θα συμβεί αν γράψουμε την καταχώρηση:

```
mnm2=mnm1;
```

- Αν και θα θέλαμε να αντιγραφούν τα δεδομένα των συμβολοσειρών της mnm1 στην mnm2, γίνεται bit προς bit αντιγραφή, άρα απλά οι αντίστοιχοι δείκτες αντιγράφουν τις διευθύνσεις τους. Έτσι η εικόνα της μνήμης θα είναι:



- Η οποία δεν είναι επιθυμητή!

Β. Προγραμματιστικές Τεχνικές στις Δομές

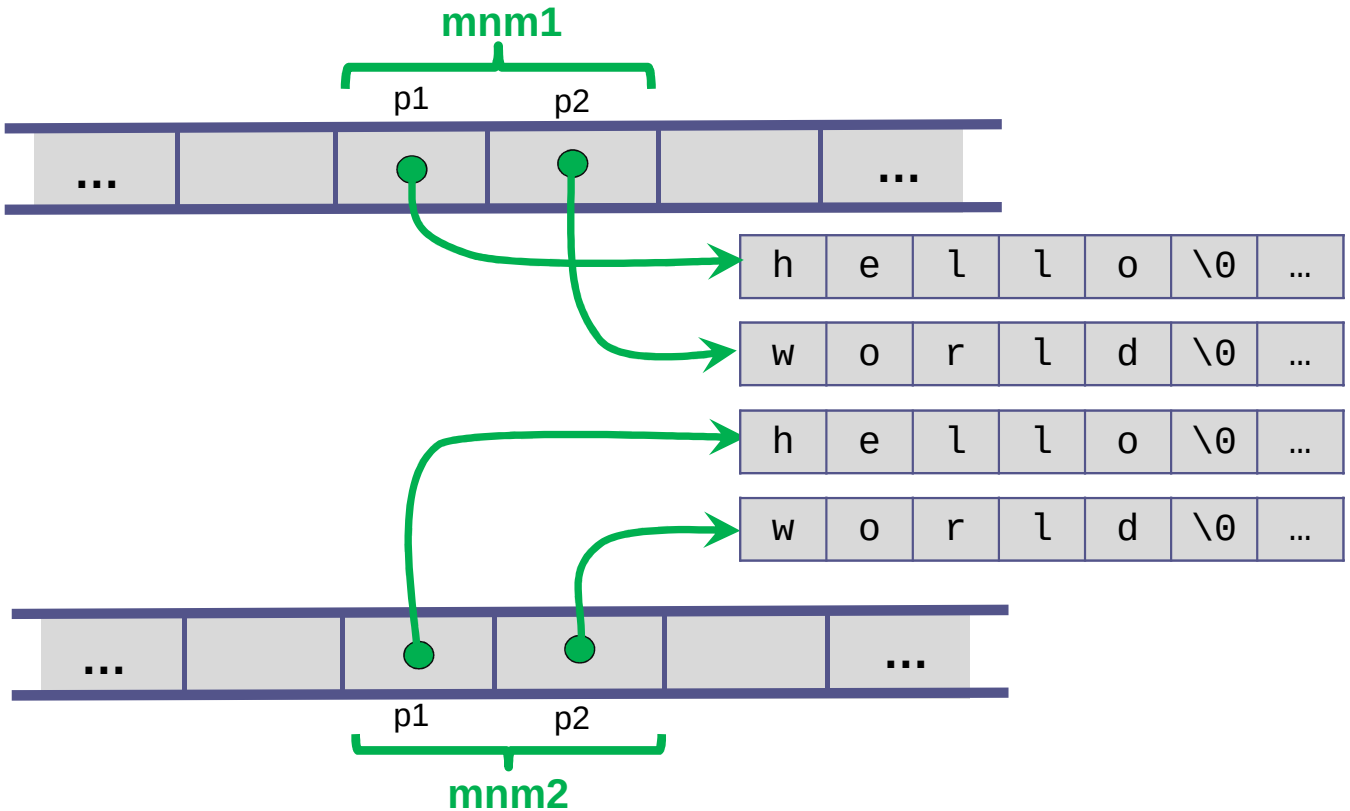
3. Δομές και Δείκτες

1. Δομές με Δείκτες

- Ο σωστός τρόπος για να κάνουμε την αντιγραφή είναι να κάνουμε ξεχωριστά αντιγραφή των μελών της δομής ως εξής:

```
strcpy(mnm2.p1,mnm1.p1);
strcpy(mnm2.p2,mnm1.p2);
```

- οπότε και η εικόνα της μνήμης θα είναι αυτή που πρέπει.



Β. Προγραμματιστικές Τεχνικές στις Δομές

3. Δομές και Δείκτες

1. Δομές με Δείκτες

Συνεπώς

- Για να κάνουμε αντιγραφή μέσω του τελεστή εκχώρησης, πρέπει να προσέχουμε αν η δομή μας περιέχει δείκτες.
- **Όταν η δομή περιέχει δείκτες, η αντιγραφή των δεδομένων τους πρέπει να γίνει αυτόνομα για να μην δείχνουν και οι δύο δείκτες στον ίδιο χώρο μνήμης**

Β. Προγραμματιστικές Τεχνικές στις Δομές

3. Δομές και Δείκτες

2. Δείκτες που δείχνουν σε Δομές

- Όταν δηλώνουμε έναν δείκτη που να δείχνει σε ένα στιγμιότυπο μιας δομής, π.χ. με τις εντολές:

```
struct atomo x;
struct atomo *p;

p=&x;
```

  - (ακριβώς όπως είδαμε ότι κάνουμε τις ενέργειες αυτές στο Μάθημα 9)
- Τότε μπορούμε να έχουμε πρόσβαση μέσω δείκτη στα μέλη της δομής με τις μέχρι τώρα γνώσεις μας ως εξής:

```
(*p).ilikia=35;
```

- Αλλά και με έναν νέο τελεστή το βελάκι (γράφεται με έμμεσο τρόπο με παύλα (-) και μεγαλύτερο (>), δηλαδή (->) στην θέση της τελείας ως εξής:

```
p->ilikia=35;
```

Συνεπώς πρόσβαση στα μέλη της δομής κάνω:
- Με τον τελεστή τελεία (.) αν έχω στιγμιότυπο της δομής
- Με αστεράκι στον δείκτη (*) και τελεία (.) αν έχω δείκτη σε στιγμιότυπο της δομής.
- Με τον τελεστή βελάκι (->) αν έχω δείκτη σε στιγμιότυπο της δομής

Β. Προγραμματιστικές Τεχνικές στις Δομές

4. Δομές και Συναρτήσεις

- Βεβαίως είναι συνηθισμένο να διοχετεύουμε ως ορίσματα σε συναρτήσεις, στιγμιότυπα δομών.
- Έτσι μπορούμε να ορίσουμε μια συνάρτηση που παίρνει ως όρισμα ένα στιγμιότυπο της struct atomo ως εξής:

```
void func(struct atomo x)
{
   ...
}
```

- Ενώ μπορούμε να έχουμε και επιστροφή στιγμιότυπου δομής από συνάρτηση όπου θα καθορίζουμε ότι ο τύπος επιστροφής είναι struct atomo.

```
struct atomo func(int x, int y)
{
   ...
}
```

- Καθώς και συναρτήσεις που συνδυάζουν τα παραπάνω.
- Επίσης δεν ξεχνάμε ότι αν η συνάρτηση θέλει να αλλάζει τα μέλη της δομής θα πρέπει να διοχετεύουμε τα ορίσματα μέσω αναφοράς.

Β. Προγραμματιστικές Τεχνικές στις Δομές

5. Δομές και η εντολή typedef

- Είδαμε ότι όποτε θέλουμε να αναφερθούμε στην δομή που χρησιμοποιούμε πρέπει να αναφέρουμε και την λέξη κλειδί struct.
- Ωστόσο με χρήση της εντολής typedef μπορούμε να κατασκευάσουμε ένα συνώνυμο για να κάνουμε την προγραμματιστική δουλειά πιο γρήγορη.
- Π.χ. Με τις παρακάτω εντολές πριν την main:

```
struct atomo{
   char onoma[30];
   char eponimo[30];
   int ilikia;
   float misthos;
};

typedef struct atomo person;
```

  - Μπορούμε να γράφουμε παντού person αντί για struct atomo. Π.χ. Στην δήλωση μεταβλητής μπορούμε να γράψουμε:

```
person x,y;
```

Β. Προγραμματιστικές Τεχνικές στις Δομές

5. Δομές και η εντολή typedef

- Υπάρχει και ένας πιο «τεχνικός τρόπος» να γράψουμε μια δομή με χρήση της εντολής typedef ο οποίος χρησιμοποιείται συχνά.
- Γράφουμε κατευθείαν το συνώνυμο αφού ορίσουμε την δομή χωρίς όνομα μέσα στην εντολή typedef:

```
typedef struct{
   char onoma[30];
   char eponimo[30];
   int ilikia;
   float misthos;
} person;
```

- Με την ίδια χρήση με πριν. Έπειτα η χρήση της δομής γίνεται και πάλι με το συνωνυμο που κατασκευάσαμε. Π.χ.:

```
person x,y;
```

Γ. Ενώσεις

1. Τι είναι η ένωση

- Οι ενώσεις (unions) είναι το ίδιο με τις δομές, δηλώνονται και χρησιμοποιούνται με τον ίδιο τρόπο με τις δομές, με την διαφορά ότι μόνο ένα από τα μέλη της μπορεί να χρησιμοποιηθεί κάθε φορά.
- Είναι ευθύνη του προγραμματιστή να παρακολουθεί ποιο μέλος της ένωσης χρησιμοποιεί κάθε φορά.
- Το πρόγραμμα της επόμενης διαφάνειας δείχνει ότι ανάλογα με την χρήση που έχει κάνει ο προγραμματιστής αποθηκεύεται η σωστή πληροφορία στο χώρο μνήμης.

- Οι ενώσεις ήταν ιδιαίτερα χρήσιμες παλιότερα, όταν θέλαμε να επιτύχουμε οικονομία μνήμης και ανάλογα με την περίσταση να χρησιμοποιείται η μνήμη με τον τρόπο που θέλουμε.
- Σήμερα δεν θεωρείται ιδιαίτερα χρήσιμη.

- Δείτε στο παράδειγμα ότι ανάλογα με το τελευταίο μέλος που έχουμε χρησιμοποιήσει, αποθηκεύεται σωστά η πληροφορία που έχουμε καταχωρήσει.

Γ. Ενώσεις
2. Παράδειγμα

```
/* union.c: Deixnei pos xrisimopoiw mia enosi */
#include <stdio.h>
union shared{
      char c;
      int i;
      double d;
};
main()
{
      union shared s;

      s.c='a';
      printf("\nXaraktiras: %c", s.c);
      printf("\nAkeraios  : %d", s.i);
      printf("\nDouble    : %f", s.d);

      s.d=10.55;
      printf("\nXaraktiras: %c", s.c);
      printf("\nAkeraios  : %d", s.i);
      printf("\nDouble    : %f", s.d);
}
```

Γ. Ασκήσεις

1. Υπολογισμός Ευκλείδιας Απόστασης

- Η απόσταση των σημείων A(x_1, y_1) και B(x_2, y_2) δίνεται από τον τύπο:

$$E = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

1. Ορίστε μία δομή με όνομα point με μέλη τις συντεταγμένες ενός σημείου.
2. Κατασκευάστε συνάρτηση που παίρνει ως όρισμα ένα point, διαβάζει τις συντεταγμένες του σημείου και τις επιστρέφει.
3. Κατασκευάστε συνάρτηση που δέχεται ως ορίσματα δύο σημεία, υπολογίζει την ευκλείδια απόστασή τους και την επιστρέφει.
4. Κατασκευάστε συνάρτηση main που με χρήση των παραπάνω συναρτήσεων, διαβάζει τις συντεταγμένες δύο σημείων και τυπώνει την ευκλείδια απόστασή τους.

- Για τον υπολογισμό της τετραγωνικής ρίζας, χρησιμοποιήστε την συνάρτηση sqrt(x) ενσωματώνοντας τη βιβλιοθήκη συναρτήσεων math.h
- Για τον υπολογισμό του τετραγώνου, χρησιμοποιήστε την συνάρτηση pow(a,n) που επιστρέφει την ποσότητα a^n

Γ. Ασκήσεις
2. Δομές και Συναρτήσεις

1. Δηλώστε μία δομή με μέλη 4 συμβολοσειρές onoma, diefthinsi, arithmos, nomos. Οι συμβολοσειρές να έχουν σταθερό μέγεθος ίσο με 80

2. Με την εντολή typedef ορίστε ως RECORD το συνώνυμο της δομής που ορίσατε.

3. Γράψτε μια συνάρτηση έστω με όνομα read_record η οποία:
   - Παίρνει ως όρισμα ένα στιγμιότυπο της RECORD δια αναφοράς και πληκτρολογώντας κατάλληλα μηνύματα αρχικοποιεί την δομή με είσοδο του χρήστη.

4. Γράψτε μια συνάρτηση έστω με όνομα print_record η οποία
   - Παίρνει ως όρισμα ένα στιγμιότυπο της RECORD δια τιμής και τυπώνει τα περιεχόμενα του στην οθόνη.

5. Γράψτε μια main που αναδεικνύει τις παραπάνω συναρτήσεις δηλώνοντας και αρχικοποιώντας από την είσοδο δύο στιγμιότυπα της δομής και τυπώνοντας τα.

Γ. Ασκήσεις
3. Δομές που περιέχουν Δείκτες

- Τροποποιήστε το πρόγραμμα της προηγούμενης εφαρμογής, έτσι ώστε να χρησιμοποιεί δυναμική (αντί για στατική) δέσμευση μνήμης στις συμβολοσειρές ενός στιγμιοτύπου της δομής.
  - (Μην ξεχάσετε να απελευθερώσετε τη μνήμη στο τέλος)
  - (πιθανόν να είναι χρήσιμο να ορίσετε συναρτήσεις για την δέσμευση και την αποδέσμευση της μνήμης)

Γ. Ασκήσεις
4. Πίνακες που περιέχουν Δομές

- Τροποποιήστε το προηγούμενο πρόγραμμα ώστε η διαχείριση να γίνεται με έναν στατικό πίνακα από Ν στιγμιότυπα της δομής. Ορίστε οι ενέργειες να γίνονται για Ν=3 εγγραφές.

Γ. Ασκήσεις
5. Δυναμικοί Πίνακες Δομών

- Τροποποιήστε το προηγούμενο πρόγραμμα ώστε ο πίνακας εγγραφών να δεσμεύεται δυναμικά. Το πλήθος των εγγραφών να εισάγεται από το χρήστη.

Γ. Ασκήσεις
6. Αντιγραφή Στιγμιοτύπων Δομής

- Τροποποιήστε το προηγούμενο πρόγραμμα ώστε
  1. Να αντιγράφεται μία εγγραφή του πίνακα (ο χρήστης να επιλέγει την εγγραφή) σε ένα αυτόνομο στιγμιότυπο της δομής (ονομάστε το x).
     1. Αυτό να γίνεται μέσω μίας συναρτησης (ονομάστε την copy_record) με κατάλληλα ορίσματα.
  2. Το πρόγραμμα να τυπώνει και τη x.
  3. Προσοχή να απελευθερώνονται όλοι οι χώροι μνήμης που δεσμεύτηκαν δυναμικά.