ΔΟΜΕΣ

Πολλές φορές, μία σύνθετη οντότητα μπορεί να καθορισθείαπό ένα σύνολο δεδομένων, πιθανώς διαφορετικών τύπων,οπότε θα ήταν χρήσιμο να ομαδοποιούσαμε τα δεδομένααυτά κάτω από ένα γενικό όνομα, με σκοπό νααναφερόμαστε στην οντότητα αυτή με το όνομα αυτό.
Στην C, η δυνατότητα ομαδοποίησης δεδομένων, που αποτελούν κατά κάποιο τρόπο τα χαρακτηριστικά μίας
προγραμματιστικής οντότητας, παρέχεται από τις δομές.

Μία δομή ορίζεται με τον εξής τρόπο:
struct ετικέτα δομής

{τύπος1 μέλος1;
τύπος 2 μέλος2;
.....
τύπος *ν* μέλος*ν*;
};

Ο ορισμός μίας δομής αποτελείται από τους ορισμούς των μελών της, που περιγράφονται σαν μεταβλητές
συγκεκριμένων τύπων. Φυσικά, πίνακες, δείκτες ή ακόμα και δομές μπορούν να είναι μέλη μίας δομής.

• Παράδειγμα

```
Struct employee
{
    char firstname[10];
   char  lastname[18];
   int   id_number;
   float salary;
};
```

Στο παράδειγμα αυτόο ορίζουμε μία δομή struct employee για την αναπαράσταση πληροφοριών για ένα
υπάλληλο (μικρό όνομα, επώνυμο, αριθμός μητρώου και μισθός)
Ο ορισμός μίας δομής είναι μία δήλωση για την οποία δεν γίνεται κάποια δέσμευση μνήμης. Μπορούμε όμως να ορίσουμε συγκεκριμένες μεταβλητές που έχουν σαν τύπο μία δομή, ως εξής:
struct ετικέτα δομής μεταβλητή1, , μεταβλητή *ν*;
Τότε, δεσμεύεται η απαιτούμενη μνήμη για τις μεταβλητές, ανάλογα με τον χώρο που χρειάζεται για να φυλαχθούν τα μέλη της δομής.
Μεταβλητές με τύπο δομή μπορούν να ανατίθενται σαν τιμές σε άλλες μεταβλητές του ίδιου τύπου, μπορούν να δίνονται σαν ορίσματα κατά την κλήση συναρτήσεων και μπορούν να επιστρέφονται από συναρτήσεις στο όνομά τους.

Η αναφορά σ΄ ένα μέλος μίας μεταβλητής τύπου δομής γίνεται με την παράσταση

Μεταβλητή **.** μέλος

΄Οπως ορίζουμε μεταβλητές με τύπο κάποια δομή, έτσι μπορούμε να ορίσουμε και δείκτες σε δομές. Παράδειγμα:
struct point *ppa, *ppb;

Αν ένας δείκτης δείχνει σε κάποια δομή και θέλουμε να αναφερθούμε σ΄ ένα μέλος της, αντί να γράφουμε
(*δείκτης).μέλος η C μας δίνει τη δυνατότητα να γράψουμε πιο απλά
δείκτης->μέλος

΄Οπως συμβαίνει και με όλους τους τύπους στην C,μπορούμε να ορίσουμε και πίνακες δομών.

Αυτο-αναφορικές δομές

Τα μέλη μίας δομής μπορεί να είναι οποιουδήποτε τύπου,ακόμα και δείκτες σε δομές του ίδιου τύπου.
Χρησιμοποιώντας τέτοιου είδους δομές, που ονομάζονται αυτο-αναφορικές και είναι πολύ χρήσιμες στον
προγραμματισμό, μπορούμε να οργανώσουμε δεδομένα με τρόπους που διευκολύνουν εξαιρετικά τη διαχείριση και επεξεργασία τους.
Οι πλέον συνήθεις οργανώσεις δεδομένων μέσω αυτο-αναφορικών δομών είναι οι συνδεδεμένες λίστες (ή απλά λίστες) και τα δυαδικά δέντρα.

΄Εστω η δήλωση:

```
struct  listnode
{
int       value;
struct   listnode *next;
};
```

Αυτή η δομή ορίζει έναν κόμβο λίστας στον οποίο φυλάσσεται ένας ακέραιος και ένας δείκτης σε κόμβο λίστας.Αν θέλουμε να αποθηκεύσουμε μία ακολουθία από ακεραίους, απροσδιορίστου πλήθους, αλλά και
μεταβαλλόμενου κατά τη διάρκεια της εκτέλεσης ενός προγράμματος, μπορούμε να το κάνουμε βάζοντας κάθε ακέραιο σ΄ ένα κόμβο λίστας και δείχνοντας από κάθε κόμβοστον επόμενό του.

Η αναφορά στη λίστα γίνεται με ένα δείκτη στον πρώτο κόμβο της. Ο τελευταίος κόμβος της λίστας έχει σαν δείκτη σε επόμενο κόμβο το NULL.

Οι κόμβοι μίας λίστας δεν φυλάσσονται σε διαδοχικές θέσεις μνήμης, αφού η μνήμη γι΄ αυτούς δεσμεύεται με διαφορετικές malloc. Αυτό σημαίνει ότι δεν μπορούμε να έχουμε άμεση πρόσβαση στο *ν*-οστό στοιχείο μίας λίστας, όπως στους πίνακες, αλλά μόνο ακολουθώντας την αλυσίδα των στοιχείων από το πρώτο έως το *ν*-οστό.

Πράξεις σε λίστες
Η παρεμβολή και η διαγραφή ενός κόμβου σε/από συγκεκριμένη θέση της λίστας

ΠΑΡΑΔΕΙΓΜΑΤΑ ΜΕ ΛΙΣΤΕΣ ΣΕ C

Ginete eisagogh stoixeiou sthn arxh thw listas,eisagogh stoixeiou sto telos thw listas, eisagogh stoixeiou se tyxaia thesh thw listas,diagrafh stoixeiou apo tin lista, anazhthsh stoixeiou sth lista, .taxinomhsh listas

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

struct node
{
  int value;
  node *next;
};

char ans;
node* listbase;//Η μεταβλητή listbase ορίζεται ως δείκτης σε κόμβο της λίστας
int x,i,epilogi,thesi,f,plithos_stoixeion=0;

//Η διαδικασία list_print χρησιμοποείται για να εκτυπώσει τη λίστα
void list_print()
{
 node *v=listbase;

 if (v==NULL)
          printf("H lista einai kenh\n");
 else
 {
        printf("\nH lista einai\n");
        while (v!=NULL)
        {
                printf("%d\t",v->value);
                v=v->next;
                plithos_stoixeion++;
        }
 }
 printf("\n");
}

void list_count()
{
 node *v=listbase;
plithos_stoixeion=0;
        while (v!=NULL)
        {
                v=v->next;
                plithos_stoixeion++;
                printf("Plithos stoixeion %d = \n",plithos_stoixeion);
        }

}


//Η διαδικασία list_push_infront χρησιμοποείται για να προσθέσει ένα νέο στοιχείο
στην αρχή της λίστας
void list_push_infront()
```

```
{
        node *v=listbase;
        node *w;

        w=(node*)malloc(1*sizeof(node));

        printf("Diabase to neo stoixeio pou tha mpei sthn arxh tis listas\n");
        scanf("%d",&w->value);

        w->next=v;
        listbase=w;
        list_print();
}

//Η διαδικασία list_push_atend χρησιμοποείται για να προσθέσει ένα νέο στοιχείο στο
τέλος της λίστας
void list_push_atend()
{
 node *w,*v;
 int t;

 w=(node*)malloc(1*sizeof(node));
 printf("Diabase to neo stoixeio pou tha mpei sto telos tis listas\n");
 scanf("%d",&t);
 w->value=t;

 v=listbase;

 if (v==NULL)
 {
          listbase=w;
    w->next=NULL;
 }
 else
 {
        while (v->next!=NULL)
                v=v->next;

        v->next=w;
        w->next=NULL;
 }
 list_print();
}

/*Η διαδικασία list_push_inside χρησιμοποείται για να προσθέσει ένα νέο στοιχείο σε
μια τυχαία θέση της
λίστας. Η θέση αυτή μπορεί να είναι από την πρώτη μέχρι και την τελευταία*/
void list_push_inside()
{
 node *v,*w;
```

```
 i=1;
 v=listbase;

 w=(node*)malloc(1*sizeof(node));

 printf("Dose thn timh tou neou kombou ths listas\n");
 scanf("%d",&w->value);

 printf("Se poia thesh ths listas theleis na to eisageis \n");
 scanf("%d",&thesi);
 if (thesi==1)
 {
     w->next=v;
     listbase=w;
 }
 else
 {
     while (v->next!=NULL && i<thesi-1)
     {
        v=v->next;
        i++;
           }
     w->next=v->next;
     v->next=w;
 }
 list_print();
}

//Η διαδικασία list_delete χρησιμοποιείται για να διαγράψει ένα στοιχείο από τη
λίστα
void list_delete()
{
          node *v,*w;

          if (listbase!=NULL)
   {
             printf("Dose to stoixeio ths listas pou tha diagrafei\n");
     scanf("%d",&x);

     v=listbase;
     f=0; //έστω ότι το στοιχείο που θα διαγραφεί δεν υπάρχει

     if (v->value==x)
           {
        v=v->next;
        listbase=v;
        f=1;
           }
```

```
      if (f==0)
            {
          w=v->next;
          while (w!=NULL && f==0)
                              {
             if (w->value!=x)
                                    {
                 v=v->next;
                 w=w->next;
                                    }
             else
               f=1;
                              }

          if (f==1)
                              {
             v->next=w->next;
             free(w);
                              }
          else
            printf("To zhtoymeno stoixeio den brethike sth lista \n");
            }
      list_print();
          }
   else
      printf("H lista einai kenh \n");
}

//Η διαδικασία list_search χρησιμοποιείται για να διαγράψει ένα στοιχείο από τη
λίστα
void list_search()
{
          node *v,*w;
          int i=1;

          if (listbase!=NULL)
    {
             printf("Dose to stoixeio ths listas pou tha anazhthsoume\n");
      scanf("%d",&x);

      v=listbase;
      f=0; //έστω ότι το στοιχείο που θα αναζητηθεί δεν υπάρχει

      if (v->value==x)
            {
        printf("To zhtoymeno stoixeio brethike sth thesh 1 ths listas\n");
                    f=1;
                    return;
            }
```

```
    if (f==0)//αν το στοιχείο δεν έχει βρεθεί στην 1η θέση
          {
                i++;//πήγαινε στην επόμενη
        w=v->next;
        while (w!=NULL && f==0)
                    {
          if (w->value!=x)
                              {
             v=v->next;
             w=w->next;
                                    i++;
                              }
          else
            f=1;
                    }

        if (f==1)
                    {
         printf("To zhtoymeno stoixeio brethike sth thesh %d \n",i);
                    }
        else
          printf("To zhtoymeno stoixeio den brethike sth lista \n");
          }
        }
  else
      printf("H lista einai kenh \n");
}

void list_sort()
{
 int temp,k,i;
 node *p,*q;

 list_count();
 printf("\n plithos = %d\n",plithos_stoixeion);
 for (k=1;k<plithos_stoixeion;k++)
 {
      p=listbase;
      q=p->next;
      for (i=1;i<=plithos_stoixeion-k;i++)
      {
         if (p->value>q->value)
                    {
            temp=p->value;
            p->value=q->value;
            q->value=temp;
                    }
         p=p->next;
         q=q->next;
              }
```

```
  }
  list_print();
}

void main()
{
        listbase=NULL;
        do
        {
                printf("1.eisagogh stoixeiou sthn arxh thw listas\n");
                printf("2.eisagogh stoixeiou sto telos thw listas\n");
                printf("3.eisagogh stoixeiou se tyxaia thesh thw listas\n");
                printf("4.diagrafh stoixeiou apo thw lista\n");
                printf("5.anazhthsh stoixeiou sth lista\n");
                printf("6.taxinomhsh listas\n");

                printf("Dose thn epilogh sou\n");
           scanf("%d",&epilogi);
           switch(epilogi)
                {
                case 1:list_push_infront();
                                break;

                case 2:list_push_atend();
                                break;

                case 3:list_push_inside();
                                break;

                case 4:list_delete();
                                break;

                case 5:list_search();
                                break;

                case 6:list_sort();
                                break;

                default:printf("Wrong choice\n");
                }

                getchar();
                printf("Continue y/n \n");
           scanf("%c",&ans);
        }
        while (ans=='y');
}
```