

# **Bilgisayar Bilimlerine Giriş-II**

**-5-**

**BIL 1002**

**Dokuz Eylül Üniversitesi, Fen Fakültesi,  
Bilgisayar Bilimleri Bölümü**

# Yapı (Struct) Tanımlamaları

## ► Yapılar

- Diğer tipte nesneler kullanılarak oluşturulan, türetilmiş veri tipleridir.

```
struct kart{  
    char *taraf;  
    char *takim;  
};
```

- **struct** anahtar kelimesi yapı tanımını başlatır.
- Kart tanıttıcısı yapı etiketidir (structure tag). Yapı etiketleri, tanımına isim verir.
- Parantezler içinde bildirilen değişkenler yapı elemanlarıdır.

# Yapı (Struct) Tanımlamaları

```
struct isci{  
    char adi[20];  
    char soyadi[20];  
    int yas;  
    char cinsiyet;  
    double saatlikUcreti;  
};
```

```
struct isci2{  
    char adi[20];  
    char soyadi[20];  
    int yas;  
    char cinsiyet;  
    double saatlikUcreti;  
    struct isci2 kisi; //hata x  
    struct isci2 *ePtr; //gösterici ✓  
};
```

Yukarıda ki yapı tanımlaması hafızada yer ayırmaz, bunun yerine değişkenler bildirmek için kullanılacak yeni bir veri tipi oluşturur. Yapı değişkenleri diğer tiplerdeki değişkenler gibi bildirilirler.

# Yapı (Struct) Tanımlamaları

```
struct kart{  
    char *taraf;  
    char *takim;  
} a, deste[52], *cPtr;
```

**struct kart a, deste[52], \*cPtr;**

bildirimi, **struct kart** tipinde bir **a** değişkeni, **struct kart** tipinde 52 elemana sahip bir **deste** dizisi ve **struct kart**'ı gösteren bir gösterici değişkeni bildirir. Verilen bir yapı tipindeki değişkenler, değişken isimleri yapı tanımının sonundaki parantez ile yapı tanımlamasını sonlandıran noktalı virgül arasına, virgüllerle ayrılmış bir biçimde yazılarak bildirilebilir.

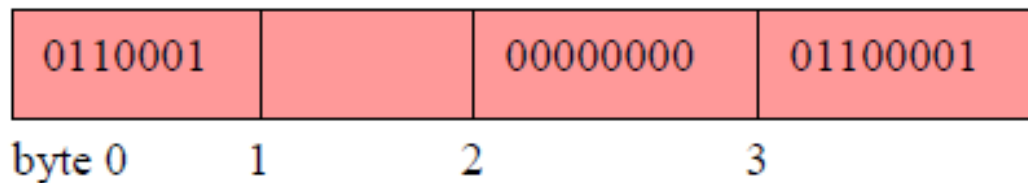
# Yapı (Struct) Tanımlamaları

- ▶ Yapılar, `=` ve `!=` operatörleri kullanılarak karşılaştırılmaz çünkü yapı elemanları hafızada ardışık byte'lar içinde bulunmak zorunda değildir.
- ▶ Bazen yapılar içinde boşluklar olabilir çünkü bilgisayarlar belirli veri tiplerini özel sınırlar içerisinde depolarlar.
- ▶ Bu sınırlar, verileri bilgisayarda tutmak için kullanılan standart hafıza birimleri olarak düşünülebilir.
- ▶ Bu standart birim 1 byte (halfword) 2 byte (word) veya 4 byte (double word) uzunluğunda olabilir.

# Yapı (Struct) Tanımlamaları

```
struct ornek{  
    char c;  
    int i;  
} numune1,numune2;
```

Bu yapı tanımlaması ile **struct ornek** tipinde iki değişken; **numune1** ve **numune2** bildirilmiştir. 2-byte sınırlar kullanan bir bilgisayar, **struct ornek** yapısının her elemanını bir sınıra hizalayabilir. Yani her elemanı bir sınırın başlangıcına yerleştirir (bu, her makinede değişebilir).



# Yapılara İlk Değer Atama

Yapılara, dizilerde olduğu gibi atama listeleri ile atama yapılır. Yapıya değer atamak için, yapı değişkeninin adından sonra eşittir işareti ve küme parantezleri içinde virgüllerle ayrılmış atama değerleri kullanılır.

```
struct kart a={"İki","Kupa"};
```

bildirimi daha önceden tanımlanmış **struct kart** tipinde bir **a** değişkeni yaratır ve bu değişkenin **taraf** elemanına “İki” ve **takim** elemanına “Kupa” değerini atar. Eğer atama listesinde yapı elemanlarından daha az sayıda atama değeri varsa, kalan elemanlar otomatik olarak **0**’a (ya da eleman gösterici ise **NULL**’a) atanır.

# Yapı Elemanlarına Ulaşmak

- ▶ Yapı elemanlarına ulaşmak için iki operatör kullanılır.
  - Yapı elemanı operatörü (.)  
**printf(“%s”, a.takim);**
  - Yapı gösterici operatörü  
**printf(“%s”, aPtr -> takim);**

**aPtr-> takim** deyimi, **(\*aPtr).takim** ile eşdeğerdir. Burada parantezler gereklidir çünkü yapı elemanı operatörü ( . ), gösterici operatöründen (\*) daha yüksek önceliğe sahiptir. Yapı gösterici operatörü ve yapı elemanı operatörü, parantez ve dizilerde kullanılan köşeli parantez operatörüyle ( [ ] ) birlikte en yüksek önceliğe sahiptir ve soldan sağa doğru işler.

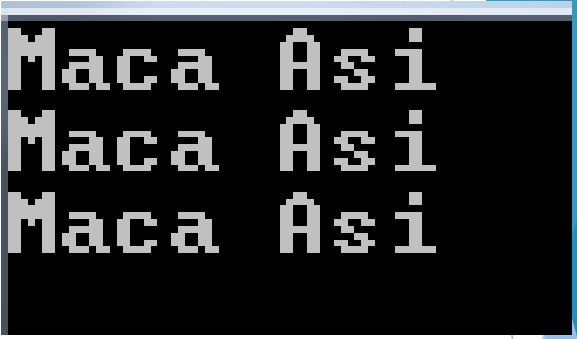


```
#include <stdio.h>
#include <conio.h>
```

```
struct kart {
    char *taraf;
    char *takim;
};
```

```
int main()
{
    struct kart a;
    struct kart *aPtr;

    a.taraf = "As";
    a.takim = "Maca";
    aPtr = &a;
    printf( "%s %s%s\n%s %s%s\n%s %s%s\n",
            a.takim, a.taraf,"i",
            aPtr->takim, aPtr->taraf,"i",
            ( *aPtr ).takim, ( *aPtr ).taraf ,"i");
    return 0;
}
```



```
Maca Asi
Maca Asi
Maca Asi
```

# Yapıların Fonksiyonlarda Kullanılması

- ▶ Yapılar fonksiyonlara
  - ▶ Yapı elemanlarının bağımsız bir şekilde geçirilmesiyle,
  - ▶ tüm yapının geçirilmesiyle ya da
  - ▶ yapıyı gösteren bir göstericinin geçirilmesiyle geçirilirler.
  - ▶ Değere göre çağırma ile geçirilirler. Bu sebepten, çağırıcının yapı elemanları çağrılan fonksiyonla değiştirilemez. (Yapı elemanları da dahil)
  - ▶ Bir yapıyı referansa göre çağırmak için yapı değişkeninin adresi geçirilir. Yapı dizileri, diğer tüm diziler gibi, otomatik olarak referansa göre geçirilir.

# Structlar ve Pointerlar

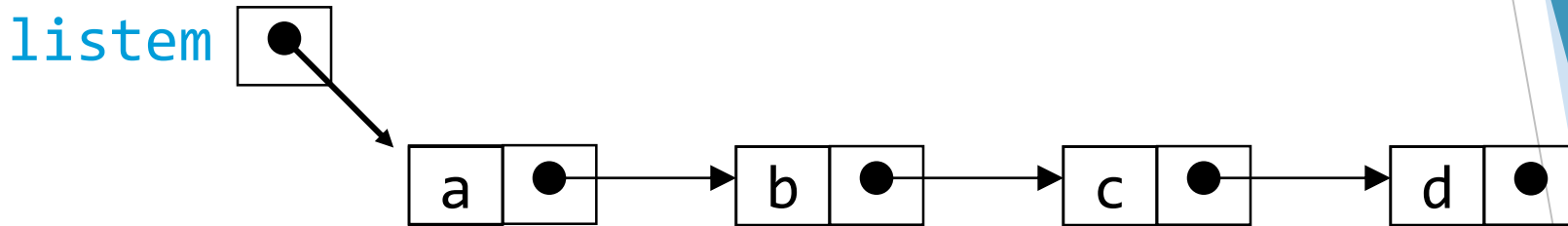
```
struct item {  
    char *s;  
    struct item *next;  
}
```

Yasak değil!!

- Örneğin bir item başka bir itemi gösteriyor olabilir.  
Böylece öğelerin bir listesi oluşturulabilir.

# Listeler (Bağlı Liste)

- Bir bağlı liste bir dizi düğümden oluşur.



- Her düğümden en azından bir **değer** bulunur.
- Ayrıca başka bir düğüme bağlanan bir **bağlantı (pointer)** vardır.
- Son düğümün bağlantısı **NULL** olmalıdır.
- Bir listenin başlangıç düğümün (**head**) tutulması gerekebilir.

# Structlar ve Pointerlar

- Aşağıdaki ifade ise uygun değildir:

```
struct motor {  
    float volts;  
    float amps;  
    float rpm;  
    unsigned int phases;  
};    //struct motor
```

```
motor m;  
motor *p;
```

Aşağıdaki şekilde yazılmalı;  
**struct** motor m;  
**struct** motor \*p;

# typedef

- **typedef** anahtar kelimesi, daha önceden tanımlanmış veri tipleri için eşanlamlı sözcükler (ya da takma isimler) yaratan bir mekanizma sağlar. Yapı tipi isimleri genellikle **typedef** ile tanımlanarak daha kısa tip isimleri oluşturulur.

**typedef struct kart Kart;**

ifadesi **struct kart** tipi ile eş anlamda kullanılan, **Kart** isminde yeni bir tip yaratır.

# typedef

- ▶ C programcıları **typedef** anahtar kelimesini, yapı tipi tanımlarken kullanırlar böylece yapı etiketi kullanmaya gerek kalmaz.

```
typedef struct{  
    char *taraf;  
    char *takim;  
} Kart;
```

tanım1, **Kart** yapı tipini ayrı bir **typedef** ifadesi kullanmaya gerek kalmadan yaratır.

# typedef

- ▶ Artık **Kart**, **struct kart** tipinde değişkenler bildirmek için kullanılabilir.

**Kart deste[52];**

bildirimi, 52 **Kart** yapısından (yani **struct kart** tipinde değişkenlerden) oluşan bir dizi bildirir. **typedef** ile yeni bir isim yaratmak yeni bir tip yaratmaz; **typedef** daha önceden var olan bir tip ismi için, eş anlamlı olarak kullanılabilecek yeni tip isimleri yaratır. Anlamlı bir isim, programın daha anlaşılır olmasını sağlar.



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

```
int main()
```

```
{
```

```
    struct complex {
```

```
        float real, imag;
```

```
    };
```

```
    struct complex z1, z2, z3;
```

```
    printf("1. sayinin reel ve imajiner kisimlari gir ");
```

```
    scanf("%f %f", &z1.real, &z1.imag);
```

```
    printf("2. sayinin reel ve imajiner kisimlari gir ");
```

```
    scanf("%f %f", &z2.real, &z2.imag);
```

```
    z3.real = z1.real + z2.real;
```

```
    z3.imag = z1.imag + z2.imag;
```

```
    printf("%.2f + %.2fjn", z3.real, z3.imag);
```

```
    getch();
```

```
    return 0;
```

```
}
```

```
1. sayinin reel ve imajiner kisimlari gir 23 56
2. sayinin reel ve imajiner kisimlari gir 56 48
79 + 104jn
```

```

#include <stdio.h>
#include <conio.h>
#define OGRSAY 3
int main()
{
    struct ogrenci{
        char no[10];
        int notu;
    };
    struct ogrenci ogr[OGRSAY];
    int i, j;
    float t, ort;
    for (i=0; i<OGRSAY; i++)
    {
        printf("%d. ogrencinin nosu ve notu : ",
i+1);
        scanf("%s%d",ogr[i].no, &ogr[i].notu);
    }
    t = 0;
    for (i=0; i<OGRSAY; i++)
        t = t + ogr[i].notu;
    ort = t / OGRSAY;

```

```

printf("-----\n");
printf("Ortalama = %f\n", ort);
printf("-----\n");
printf("Ortalamayi gecen
ogrencilerin\nNosu\tNotu\n");
for (i=0; i<OGRSAY; i++)
    if (ogr[i].notu > ort)
        printf("%s\t%d\n", ogr[i].no,
ogr[i].notu);
getch();
return 0;
}

```

```

1. ogrencinin nosu ve notu : 01 75
2. ogrencinin nosu ve notu : 02 45
3. ogrencinin nosu ve notu : 03 35

```

```

Ortalama = 51.666668

```

```

Ortalamayi gecen ogrencilerin
Nosu      Notu
01        75

```

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

```
#include <conio.h>
```

```
typedef struct {
```

```
    float re, im;
```

```
} complex;
```

```
complex toplu(complex, complex);
```

```
complex cikart(complex, complex);
```

```
complex carp(complex, complex);
```

```
complex toplu(complex a, complex b)
```

```
{
```

```
    complex z;
```

```
    z.re = a.re + b.re;
```

```
    z.im = a.im + b.im;
```

```
    return z;
```

```
}
```

```
complex cikart(complex a, complex b)
```

```
{
```

```
    complex z;
```

```
    z.re = a.re - b.re;
```

```
    z.im = a.im - b.im;
```

```
    return z;
```

```
}
```

```
complex carp(complex a, complex b)
```

```
{
```

```
    complex z;
```

```
    z.re = a.re * b.re - a.im * b.im;
```

```
    z.im = a.re * b.im + a.im * b.re;
```

```
    return z;
```

```
}
```

```

int main()
{
    complex z1={ 1,1}, z2={2,2};
    complex z3;
    printf("Karmasik Sayi 1:%3.0f %3.0f\n"
        "Karmasik Sayi 2:%3.0f %3.0f\n"
        ,z1.re,z1.im,z2.re,z2.im);
    printf("Toplama:\n");
    z3 = topla(z1, z2);
    printf("%3.0f %3.0f\n",z3.re,z3.im);
    printf("Cikartma:\n");
    z3 = cikart(z1, z2);
    printf("%3.0f %3.0f\n",z3.re,z3.im);
    printf("Carpma:\n");
    z3 = carp(z1, z2);
    printf("%3.0f %3.0f\n",z3.re,z3.im);
    getch();
    return 0;
}

```

```

Karmasik Sayi 1: 1 1
Karmasik Sayi 2: 2 2
Toplama:
 3 3
Cikartma:
-1 -1
Carpma:
0 4

```

# ÖRNEKLER

**Soru1:** Bir tanesi tam sayı diğeri string olan iki elemanlı bir struct'ı bir başka struct'a direk kopyalama, memcpy kullanarak kopyalama ve tek tek elemanlarını kopyalarak kopyalama metotları ile kopyalayan bir program yazınız.

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

struct ogr
{
    int id;
    char name[30];
};

int main()
{
    int i;
    struct ogr rd1 = {1, "Ahmet"};
    struct ogr rd2, *rd3, *ptr1, rd4;

    printf(" Id : %d \n Name : %s\n",
           rd1.id, rd1.name);

    rd2=rd1;

    printf("\ndirek kopyalama\n");
    printf(" Id : %d \n Name : %s\n",
           rd2.id, rd2.name);
```

```
rd3=(struct ogr*)malloc(sizeof(rd1));
memcpy(rd3, &rd1, sizeof(rd1));

printf("\nmemcpy kullanarak kopyalama\n");
printf(" Id : %d \n Name : %s\n",
       rd3->id, rd3->name);

printf("\ntek tek elemanlari kullanarak kopyalama
kopyalama\n");
rd4.id=rd1.id;
strcpy(rd4.name, rd1.name);

printf(" Id : %d \n Name : %s\n",
       rd4.id, rd4.name);

return 0;
}
```

```
Id : 1
Name : Ahmet

direk kopyalama
Id : 1
Name : Ahmet

memcpy kullanarak kopyalama
Id : 1
Name : Ahmet

tek tek elemanlari kullanarak kopyalama kopyalama
Id : 1
Name : Ahmet

-----
Process exited with return value 0
Press any key to continue . . . _
```

**Soru2:** Bir tanesi tam sayı diğeri string olan iki elemanlı bir struct'ı kendisini ekrana yazdırmaya yarayan bir fonksiyona değeri ile, referansı ile ve bir global değişken yardımı ile taşıyan C programını yazınız.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
struct ogr
{
    int id;
    char name[30];
};
void func(struct ogr record);
void func1(struct ogr*);
void func2();
struct ogr ogr1;
int main()
{
    struct ogr rd;
    rd.id=1;
    strcpy(rd.name, "ahmet");
    printf("dogrudan struct'i fonksiyona vererek\n");
    func(rd);
    printf("gosterici struct'i fonksiyona vererek\n");
    func1(&rd);
    printf("global struct'i fonksiyona vererek\n");
    ogr1=rd;
    func2();
    return 0;
}
```

```
void func(struct ogr rd)
{
    printf(" Id : %d \n", rd.id);
    printf(" Ad : %s \n", rd.name);
}
void func1(struct ogr *rd)
{
    printf(" Id : %d \n", rd->id);
    printf(" Ad : %s \n", rd->name);
}
void func2()
{
    printf(" Id : %d \n", ogr1.id);
    printf(" Ad : %s \n", ogr1.name);
}
```

```
dogrudan struct'i fonksiyona vererek
Id : 1
Ad : ahmet
gosterici struct'i fonksiyona vererek
Id : 1
Ad : ahmet
global struct'i fonksiyona vererek
Id : 1
Ad : ahmet
```

```
-----
Process exited with return value 0
Press any key to continue . . . _
```

**Soru3:** Struct kullanarak bir sınıftaki öğrencilerin nosu adı ve notunu no yerine -1 girene kadar alıp ekrana yazdıran programı yazınız.

```
#include<stdio.h>
#include<stdlib.h>
typedef struct ogr {
    float notu;
    int no;
    char ad[30];
    ogr *next;
} ogr;
ogr *head=NULL;
ogr *cur=NULL;
int main()
{
    ogr *ptr = (ogr*)malloc(sizeof(ogr));
    cur=head=ptr;
    char c;
    printf("No:");
    scanf("%d",&cur->no);
    if(cur->no != -1)
    {
        while(1)
        {
            ogr *ptr1 = (ogr*)malloc(sizeof(ogr));
            printf("Ad:");
            scanf("%s",cur->ad);
            printf("Not:");
            scanf("%f",&cur->notu);
            printf("No:");
            scanf("%d",&ptr1->no);
```

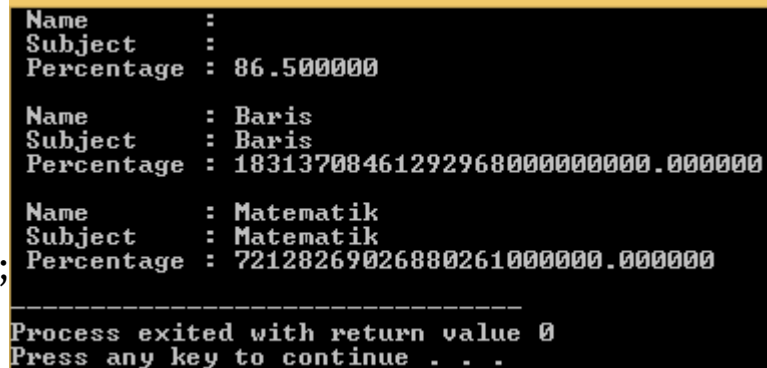
```
            if(ptr1->no==-1)
            {
                cur->next=NULL;
                break;
            }
            cur->next=ptr1;
            cur=ptr1;
        }
        printf("Sinif Listesi\n");
        printf("-----\n");
        ogr *ptr2=head;
        while(ptr2 != NULL)
        {
            printf("No:%5d\t",ptr2->no);
            printf("Ad:%10s\t",ptr2->ad);
            printf("Not:%5.2f\n",ptr2->notu);
            ptr2 = ptr2->next;
        }
    }
    return 0;
}
```

```
No:7138
Ad:Baris
Not:89
No:6155
Ad:Ahmet
Not:92
No:-1
Sinif Listesi
-----
No: 7138      Ad:   Baris   Not:89.00
No: 6155      Ad:   Ahmet   Not:92.00
-----
Process exited with return value 0
Press any key to continue . . .
```



**Soru4:** iki adet string bir adet float değişkeni olan bir union tanımlayınız daha sonra sırası ile tüm değişkenlere ilk değer atayarak, sadece string 'lerin birine değer atayarak , diğer string'e değer atayarak ekrana yazdıran bir program yazınız

```
#include <stdio.h>
#include <string.h>
union ogr
{
    char name[20];
    char subject[20];
    float percentage;
}rd;
int main()
{
    strcpy(rd.name, "Baris");
    strcpy(rd.subject, "Matematik");
    rd.percentage = 86.50;
    printf(" Name      : %s \n", rd.name);
    printf(" Subject   : %s \n", rd.subject);
    printf(" Percentage : %f \n", rd.percentage);
    printf("\n");
    strcpy(rd.name, "Baris");
    printf(" Name      : %s \n", rd.name);
    printf(" Subject   : %s \n", rd.subject);
    printf(" Percentage : %f \n", rd.percentage);
    printf("\n");
    strcpy(rd.subject, "Matematik");
    printf(" Name      : %s \n", rd.name);
    printf(" Subject   : %s \n", rd.subject);
    printf(" Percentage : %f \n", rd.percentage);
    return 0;
}
```



```
Name      :
Subject   :
Percentage : 86.500000

Name      : Baris
Subject   : Baris
Percentage : 1831370846129296800000000000.000000

Name      : Matematik
Subject   : Matematik
Percentage : 721282690268802610000000.000000

-----
Process exited with return value 0
Press any key to continue . . .
```

**Soru5:** Bir süpermarkette iki adet ürün için markette ne kadar kaldığını söyleyen bir program yazınız. Bu programda ürünün ismini, fiyatını, tipini ve ne kadar kaldığını içeren bir struct yapısı tanımlayınız. Ne kadar kaldığını belirten değişken union yapısında olsun mesela farklı birimler için(kg ve adet gibi)

```
#include <string.h>
#include <stdio.h>
typedef union {
    int units;
    float kgs;
} amount ;

typedef struct {
    char selling[15];
    float unitprice;
    int unittype;
    amount howmuch;
} product;

int main() {
    product motosiklet;
    product elma;
    product * myebaystore[2];

    int nitems = 2; int i;

    strcpy(motosiklet.selling, "Dizel Motosiklet");
    motosiklet.unitprice = 5488.00;
    motosiklet.unittype = 1;
    motosiklet.howmuch.units = 4;

    strcpy(elma.selling, "Granny");
    elma.unitprice = 0.78;
    elma.unittype = 2;
    elma.howmuch.kgs = 0.5;

    myebaystore[0] = &motosiklet;
    myebaystore[1] = &elma;

    for (i=0; i<nitems; i++) {
        printf("\n%s\n", myebaystore[i]->selling);
        switch (myebaystore[i]->unittype) {
            case 1:
                printf("Elimizde %d adet var.\n",
                    myebaystore[i]-
                    >howmuch.units);
                break;
            case 2:
                printf("Elimizde %f kg var.\n",
                    myebaystore[i]->howmuch.kgs);
                break;
        }
    }
    return 0;
}
```

```
Dizel Motosiklet
Elimizde 4 adet var.
```

```
Granny
Elimizde 0.500000 kg var.
```

```
-----
Process exited with return value 0
Press any key to continue . . .
```

SON