



YAPILAR (STRUCTURES)

Dr. Zeynep Banu ÖZGER



İÇERİK



- 1. Yapılar
- 2. Yapıların Bildirimi
- 3. Yapı Değişkenlerinin Tanımlanması
- 4. Yapı Elemanlarına Erişme
- 5. Yapı Elemanı Olarak Diziler ve Göstericiler
- 6. Yapı Değişkenlerine İlk Değer Verilmesi
- 7. Yapı Göstericileri
- 8. Ok Operatörü
- 9. İç İçe Yapılar
- 10. Yapı Değişkenleri Arasında İşlemler
- 11. Parametre Olarak Yapılar
- 12. Typedef Kullanımı



1- YAPILAR



- Aralarında mantıksal bir ilişki bulunan, farklı türden bilgiler içeren veri yapılarıdır.
- Örneğin öğrenci kaydı;
 - Ad-soyad char, numara int, ort double
- Yapılar ve diziler benzerdirler.
- Yapılar
 - Diziler gibi belleğe bitişik konumlarda yerleşirler.
 - Diziler gibi başlangıç adresleri üzerinden erişilebilir ve fonksiyonlara gönderilebilir.







- Yapı bildirimi; yapının derleyiciye tanıtılmasıdır.
- Yapı değişkenlerinin tanımlanması için ilk önce yapı bildiriminin yapılması gerekir.
- Yapı bildirimi yapıldığında derleyici yapılar hakkında bilgi edinir. Bellekte yer ayırmaz.
- Yapı bildirimi;

```
Struct[yap1_ismi]
```

•

Yapının ismi

- <tür> <yapı_elemanı>;
- <tür> <yapı_elemanı>;
- ...
- };

Yapıyı oluşturan değişken isimleri







```
Struct[yap1_ismi]
   <tür> <yapı_elemanı>;
   <tür> <yapı_elemanı>;
 Struct TARIH
   int gun;
   int ay;
   int yil;
```

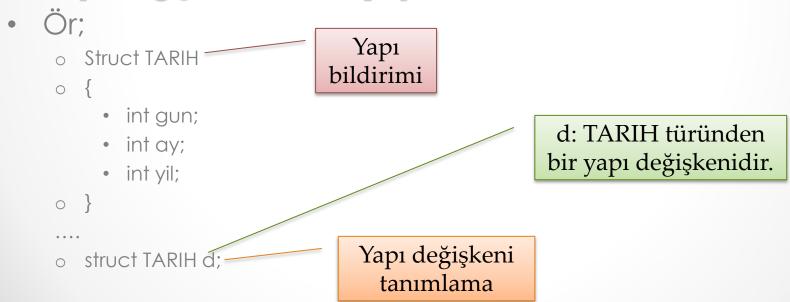
Tarih bilgisi 3 ayrı değişken yerine bir yapı ile tanımlanabilir.



3- YAPI DEĞİŞKENLERİNİN TANIMLANMASI



- Bellekte yer ayırma yapı değişkenlerinin tanımlanması ile olur.
- · Tanım;
 - o struct <yapi_ismi> <yapi_degiskeninin_ismi>
- Yapı değişkeni tanımı, yapı bildiriminden sonra olmalıdır.





3- YAPI DEĞİŞKENLERİNİN TANIMLANMASI



- Yapının bildirimi global biçimde yapıldıysa değişkenler global, yerel bir biçimde yapıldıysa değişkenler yerel faaliyet alanına sahip olur.
- d yapı türünde;

} x, y, z; -

- o 3 tane int türünde değişken var.
- Derleyici 3x4 byte yer tahsis eder.
- Yapının boyutu sizeof ile öğrenilebilir.
 - o sizeof(d) veya sizeof(struct TARIH)
- Tanımlama yapı bildirimi ile birlikte yapılabilir.

```
Struct <yapı_ismi>
{

<tür> <yapı_elemanı>;
} [değişken_listesi];

Struct TARIH
{

int gun, ay, yil;
```

TARIH yapı türünde x, y, z olarak 3 değişken tanımlandı.



4- YAPI ELEMANLARINA ERİŞME



- Yapı elemanlarına nokta operatörü ile erişilir.
- Nokta;
 - o 2 operandlidir.
 - Sol taraf bir yapı değişkeni ve sağ taraf ilgili yapının bir elemanıdır.
- Örnek;
 - o d.gun=10;
 - o d.ay=1;
 - o d.yil=2021;
 - o d yapı elema

d yapı elemanının gün değişkenine 10, ay değişkenine 1, ve yıl değişkenine 2021 değeri atandı.





'TARİH' yapısı ile tanımlanmış günün tarihi ve doğum tarihine göre birinin yaşını hesaplayalım.



5-YAPI ELEMANI OLARAK DİZİLER VE GÖSTERİCİLER



- Dizi ve göstericiler birer yapı elemanı olabilir.
- struct INSAN

```
char ad_soyad[30];
char dogum_yeri[30];
int yas;
};
struct INSAN x;
```

- Yapıdaki dizilerin isimleri, yapıdaki dizilerin bellekteki başlangıç adreslerini gösterir.
 - o x.ad_soyad ve x.dogum_yeri ilgili dizilerin başlangıç adresleridir.
- Yapıdaki bir dizinin n. elemanına erişmek için;
 - o x.adi_soyadi[n]



5-YAPI ELEMANI OLARAK DİZİLER VE GÖSTERİCİLER



- Göstericiler de birer değişken gibi tanımlanır.
- struct SAMPLE

```
 { o int a; char *c;
```

- };
- struct SAMPLE smp;
- smp.c; char türünde bir değişkenin tutulduğu bir göstericidir.



6- YAPI DEĞİŞKENLERİNE İŞK DEĞER **VERİLMESİ**



- İlk değer ataması dizilerdeki gibi yapılır.
- Derleyici yapı elemanlarını tanımlandığı sıra ile yerleştirir.
- Or:

```
o struct TARIH
     int gun, ay, yil;
```

o struct TARIH d={10,1,2021};

Yapı içinde dizi varsa, küme parantezleri arasında

tanımlanabilir.

Ör:

```
o struct NO
```

int x[5];

charp;

struct NO n={{10,20,30},"s"};

Dizinin ilk 3 elemanına ilk değer verildi.



7- YAPI GÖSTERİCİLERİ



- Bir yapının ve yapı içindeki değişkenlerin adreslerine erişilebilir.
- Ör:
 - struct DENEME
 - o {
- · charch;
- int n;
- 0 };
- o struct DENEME x;
- &x.ch > ch nesnesinin adresini dönderir. ch char tipinde olduğundan elde edilen adresin türü de char dır.
- &x > x nesnesinin adresini dönderir. Elde edilen adres struct DENEME türündendir.







- Yapı göstericisi tanımlarken;
 - o struct <yapi_ismi> *gosterici_ismi;
 - o Ör:
 - struct DENEME x; →x adında DENEME türünde yapı nesnesi tanımlandı
 - struct DENEME *p; →p adında DENEME türünde yapı gösterici tanımlandı
 - p=&x;→yapının adresi göstericiye atandı
 - (*p).n=5;

. operatörü *
operatöründen öncelikli
olduğundan, Parantez ile
önceliklendirilmelidir.



8- OK (->) OPERATÖRÜ



- Yapı göstericileri ile yapı nesnelerine erişmek için kullanılır.
- 2 operandlidir;
 - Sol taraftaki eleman yapı göstericisi sağ taraftaki yapı elemanıdır.
- (*p).a ile p->a aynı anlamdadır.
- Ör:

```
void tarihGoruntule(struct TARIH *x)
{
printf("Gun:%d\n",x->gun);
printf("Ay:%d\n",x->yil);
printf("Yıl:%d\n",x->yil);
```

 -> operatörü en öncelikli operatör grubundadır ve önceliği (), . ve [n] ile aynıdır.





• KİŞİ KAYITLARI OLUŞTURALIM







- Bir yapı içinde başka bir yapı tanımlanmasıdır.
- 2 şekilde yapılabilir.
 - o İçerdeki yapının bildirimi daha yukarda yapılır.
 - o İçerdeki yapı dıştaki yapının içinde bildirimi yapılır.

```
struct TARIH
{
  int gun, ay, yil;
};
struct INSAN
{
  char name[30];
  struct TARIH dTarih;
};
```

```
struct INSAN

{
   char name[30];
   struct TARIH
   {
    int gun, ay, yil;
   }dTarih;
};
```



10- YAPI DEĞİŞKENLERİ ARASINDAKİ İŞLEMLER



Sadece aynı türden 2 yapı değişkeni birbirine atanabilir.

```
    struct DENEME
    char ch;
    int n;
    };
    struct DENEME n1,n2;
    n1=n2→ n1 yapı değişkeninin elemanları karşılıklı olarak n2 yapı değişkenine atandı.
```

- Toplama, çıkarma gibi aritmetik veya mantıksal işlemlerde kullanılamazlar.
- İsimleri farklı olan 2 yapının elemanları, içerikleri aynı bile olsa birbirine aktarılamaz.

```
    struct X1
    {

            int n;
            struct X1 n1;
            struct X2
            {

                    int n;
                    struct X2 n2;
                    n1=n2 → GEÇERSİZ
```





 Yapılar, direk bir yapı veya adresleri ile parametre olarak fonksiyonlara geçirilebilir.

```
#include <stdio.h>
struct TARIH
    int gun, ay, yil;
void tarihGoruntule (struct TARIH x)
    printf("Gun: %d\n", x.gun);
    printf("Ay: %d\n", x.ay);
    printf("Yıl: %d\n", x.yil);
int main (void)
    struct TARIH n;
    n.qun=30.;
    n.ay=12;
    n.yil=2021;
    tarihGoruntule (n);
```





 Yapılar, direk bir yapı veya adresleri ile parametre olarak fonksiyonlara geçirilebilir.

```
#include <stdio.h>
struct TARIH
    int gun, ay, yil;
void tarihGoruntule (struct TARIH *x)
    printf("Gun: %d\n", (*x).gun);
    printf("Ay: %d\n", (*x).ay);
    printf("Y11: %d\n", (*x).yil);
int main (void)
    struct TARIH n;
    n.qun=30.;
    n.ay=12;
    n.yil=2021;
    tarihGoruntule (&n) :
```





- Bir yapı fonksiyona parametre olarak geçirildiğinde, yapının uzunluğunu da geçirmeye gerek yoktur.
- Derleyici yapı bildiriminden yapının uzunluğunu ve yapı elemanlarının başlangıç adreslerinden ne kadar uzakta olduğunu anlayabilir.





- Yapı ismi bir yer işaretçisi olduğundan fonksiyondan parametre olarak dönderilebilir.
- Bu durumda;
 - o Fonksiyonun geri dönüş tipi dönderilecek yapı türünden tanımlanır.
 - o Değer karşılandığı yerde aynı yapı türünden bir değişkene atanır.



12- TYPEDEF



- Typedef; C'de var olan değişken türlerine yenisini eklemek için kullanılır.
- Ör:
 - typedef int LENGTH;
 - o C'de tanımlı int türünün aynısından 'LENGTH' ismiyle oluşturur.
- Struct tanımlamada da, tanım işlemlerini kolaylaştırmak için kullanılabilir.
- Aşağıdaki 2 kullanım eş değerdir.

```
struct TARIH

{
  int gun, ay, yil;
};
int main

{
  struct TARIH d1,d2;
}
```

```
typedef struct TARIH
{
  int gun, ay, yil;
}tarihler;
int main
{
  tarihler d1,d2;
}
```