

MALATYA
TURGUT ÖZAL ÜNİVERSİTESİ
MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ
ALGORİTMA VE PROGRAMLAMA 2

YAPILAR (Struct)

Yapılar (struct)

- Birbirleri ile ilişki içinde bulunan veri kümelerine **yapı (struct)** adı verilir.
- Dizilere çok benzemektedir. Çünkü diziler gibi birden fazla nesneyi içinde tutarlar.
- Ayrıca diziler gibi bellekte sıralı bir şekilde bulunurlar.
- Fakat diziler sadece aynı tür veriler tutabiliyorken (int dizisi, char dizisi gibi), yapılar farklı türden nesneleri içinde barındırabilirler.

Yapılar (struct)

- Veri, bir nesne için gerekli bilgilerin derlendiği topluluk, grup veya küme, sınıf şeklinde ifade edilebilir.
- Burada nesne ile ifade edilmek istenilen şey, bilgisayar ortamında değerlendirilecek bilgileri veya özellikleri olan bir cisim için uygun yazılımın geliştirilmesidir.
- Nesne (hedef) bir malzemenin üretimi ile bilgileri içerebilir.
- Bu ürünü tanımlayabilmek için gerekli bilgiler: ürünün adı, üretim tarihi, üretim seri nosu, rengi, boyutları, kaçınıcı parça olduğu, fiyatı, katma değer vergisi ve benzeri bilgileri içerebilir.

Yapılar (struct)

- Nesne (bilgisayar yazılımı hazırlanacak) bir vatandaşa ait bilgiler olabilir:
 - vatandaşın adı, soyadı, annesinin adı, babasının adı, doğum yeri, doğum yılı, mahallesi, sokağı vs.
- Bir işyerindeki çalışan bir nesne olabilir :
 - Çalışanın adı, soyadı, sicil numarası, doğum tarihi, işe başladığı tarih, izinleri, evli olup olmadığı vs.
- Nesne tatil için bir otelde yer ayırtma işlemi olabilir :
 - tatilcinin adı soyadı, otele giriş çıkış tarihleri, ücreti, konaklama ücretini ödeme şekli, cep telefonu, adresi vs.
- Nesne bir otobüs/uçak firmasından alınan bilet olabilir :
 - hareket günü, saati, hareket yeri gibi bilgiler (veri).

Yapılar (struct)

- Kısaca (kendisi için program yazılacak) nesneye ait bilgiler veridir. Bu verilerin uygun bir şekilde saklanması ve kullanılması gerekmektedir.
- struct, farklı uzunluklardaki çeşitli bilgilerden (veya tiplerden) oluşturulmuş bir veri grubunun yekpare olarak nasıl saklanacağını tanımlanmasıdır.

Yapılar (struct)

- C++ programlama dilinde struct genel yazım şekli aşağıdaki gibidir:

```
struct yapı_ismi {  
    tip1 eleman1;  
    tip2 eleman2;  
    tip3 eleman3;  
    .  
    .  
} nesne_ismi;
```

- Buradaki yapı_ismi yapının ismi için ve nesne_ismi ise program içinde kullanılacak veri yapısının (yani değişkenler topluluğunun) ismidir.
- **Nesne ismi** veri yapısının tanımlanması esnasında **isteğe bağlı** olarak yazılabilir veya yazılmayabilir.

Yapılar (struct)

```
struct yapi_ismi {  
    tip1 eleman1;  
    tip2 eleman2;  
    tip3 eleman3;  
    .  
    .  
};
```

- **Nesne ismi** veri yapısının tanımlanması esnasında **isteğe bağlı** olarak yazılabilir veya yazılmayabilir.
- Yukardaki gibi nesne ismi yazılmaz ise } işareti noktalı virgül ile sonlandırılmalıdır.
- Oluşturulacak nesne daha sonra kod satırı içindede oluşturulabilir.

Yapılar (struct)

```
struct urunler {  
    char isim[30];  
    float fiyat;  
} Ayakkabi, Cantan, Tshort;
```

Eğer nesne ismini yapıyı oluştururken tanımlamak istemiyorsak:

```
struct urunler {  
    char isim[30];  
    float fiyat;  
};
```

Daha sonra kod içinde

```
urunler Ayakkabi, Cantan, Tshort;
```

Şeklinde oluşturabiliriz.

Yapılar elemanlarına erişim

Nesne_ismi.eleman_adı şeklinde erişilmektedir.

urunler Ayakkabi, Canta, Tshort;

Ayakkabi.isim =...; //Ayakkabı adı ile ilgili bilgi girilebilir;

Ayakkabi.fiyat =...; //Ayakkabının fiyatı ile ilgili bilgi girilebilir;

Canta.isim= ...; //Canta adı ile ilgili bilgi girilebilir;

Canta.fiyat=...; // Canta fiyatı ile ilgili bilgi girilebilir;

Tshort.isim=...; //Tshort adı ile ilgili bilgi girilebilir;

Tshort.fiyat=...; // Tshort fiyatı ile ilgili bilgi girilebilir;

Bunların herbiri ile şu veri tipleri ortaya çıkar: **Ayakkabi.isim**, **Canta.isim** ve **Tshort.isim** elemanları **char[30]** tipinde, ve **Ayakkabi.fiyat**, **Canta.fiyat** ve **Tshort.fiyat** değişkenleri ise tek duyarlı değişken tanımlayıcı olan **float** tipindedir.

Örnek 1:

```
1  #include <iostream>
2  #include <string.h>
3  using namespace std;
4
5  struct calisan{
6      char ad[20],soyad[20];
7      int yas;
8  }kisi;
9
10 int main() {
11     strcpy(kisi.ad,"Ahmet");
12     strcpy(kisi.soyad,"Demir");
13     kisi.yas=25;
14     cout<<"Calisan adi="<<kisi.ad<<endl;
15     cout<<"Calisan soyad="<<kisi.soyad<<endl;
16     cout<<"yasi="<<kisi.yas<<endl;
17     return 0;
18 }
```

C:\Users\monster\Desktop

```
Calisan adi=Ahmet
Calisan soyad=Demir
yasi=25

-----
Process exited after
Press any key to cont
```

Örnek 1a: strcpy komutu kullanmamak için char değişkenleri pointer (*) olarak kullanmamız daha mantıklı olur.

```
1  #include <iostream>
2  #include <string.h>
3  using namespace std;
4
5  struct calisan{
6      char *ad,*soyad;
7      int yas;
8  }kisi;
9
10 int main() {
11     kisi.ad="Ahmet";
12     kisi.soyad="Demir";
13     kisi.yas=25;
14     cout<<"Calisan adi="<<kisi.ad<<endl;
15     cout<<"Calisan soyad="<<kisi.soyad<<endl;
16     cout<<"yasi="<<kisi.yas<<endl;
17     return 0;
18 }
```

C:\Users\monster\Desktop

Calisan adi=Ahmet
Calisan soyad=Demir
yasi=25

Process exited after 0
Press any key to conti

Örnek 1b: struct sonunda nesne ismi vermeden aynı örnek.

O halde kod içinde yapı ismini kullanarak nesneyi tanımlamamız gerekiyor.

calisan kisi; gibi

```
1  #include <iostream>
2  #include <string.h>
3  using namespace std;
4
5  struct calisan{
6      char *ad,*soyad;
7      int yas;
8  };
9
10 int main() {
11     calisan kisi;
12     kisi.ad="Ahmet";
13     kisi.soyad="Demir";
14     kisi.yas=25;
15     cout<<"Calisan adi="<<kisi.ad<<endl;
16     cout<<"Calisan soyad="<<kisi.soyad<<endl;
17     cout<<"yasi="<<kisi.yas<<endl;
18     return 0;
19 }
```

```
C:\Users\monster\Desktop\
Calisan adi=Ahmet
Calisan soyad=Demir
yasi=25

-----
Process exited after 0.3
Press any key to continu
```

Örnek 1c: verileri yapı değişkenin tanımlandığı yerde tek seferde girme.

yapi_ismi degiskenismi={eleman1,eleman2,....};
//Elemanları tanımlandığı sıra ile giriyoruz.

```
1  #include <iostream>
2  #include <string.h>
3  using namespace std;
4
5  struct calisan{
6      char *ad,*soyad;
7      int yas;
8  };
9
10 int main() {
11     calisan kisi={"Mehmet","Akman",27};
12     cout<<"Calisan adi="<<kisi.ad<<endl;
13     cout<<"Calisan soyad="<<kisi.soyad<<endl;
14     cout<<"yasi="<<kisi.yas<<endl;
15     return 0;
16 }
```

C:\Users\monster\Desktop

Calisan adi=Mehmet
Calisan soyad=Akman
yasi=27

Process exited after 0
Press any key to conti

Örnek 1d: verileri yapının sonundatek seferde girme.

```
1  #include <iostream>
2  #include <string.h>
3  using namespace std;
4
5  struct calisan{
6      char *ad,*soyad;
7      int yas;
8  }kisi={"Samet","Dogan",33};
9
10 int main() {
11     cout<<"Calisan adi="<<kisi.ad<<endl;
12     cout<<"Calisan soyad="<<kisi.soyad<<endl;
13     cout<<"yasi="<<kisi.yas<<endl;
14     return 0;
15 }
```

```
C:\Users\monster\Desktop>
Calisan adi=Samet
Calisan soyad=Dogan
yasi=33
-----
Process exited after 0
Press any key to continue
```

Nesne Dizileri

Yapılar da bir veri türü olduğu için yapılarla da diziler tanımlanabilir.

Dizi tanımını aşağıdaki gibi yapılır:

```
Yapı_ismi dizi_adi[];
```

Diziden yapının elemanlarına erişmek ise aşağıdaki gibi olur:

```
dizi_adi[0].eleman1=....  
dizi_adi[0].eleman2=....  
dizi_adi[1].eleman1=....  
dizi_adi[2].eleman2=.... Vb gibi.
```


Örnek 2: Öğrenci adında bir yapı oluşturalım. Yapının elemanları ise char tipinde ad ve int tipinde numara olsun. Bu yapıdan 3 boyutlu bir dizi oluşturup 3 farklı öğrencinin bilgilerini girelim.

```
1  #include <iostream>
2  #include <string.h>
3  using namespace std;
4
5  struct ogrenci{
6      char *ad;
7      int numara;
8  };
9
10 int main() {
11     ogrenci dizi[3];
12     dizi[0].ad="ali";
13     dizi[0].numara=10;
14     dizi[1].ad="veli";
15     dizi[1].numara=20;
16     dizi[2].ad="Ahmet";
17     dizi[2].numara=30;
18
19     int i;
20     for (i=0;i<3;i++){
21         cout<<"Ogrenci "<<i<<endl<<"-----"<<endl;
22         cout<<"Adi="<<dizi[i].ad<<endl;
23         cout<<"Numarasi="<<dizi[i].numara<<endl<<endl<<endl;
24     }
25
26     return 0;
27 }
28
```

C:\Users\monster\Desktop\ders slay

Ogrenci 0

Adi=ali
Numarasi=10

Ogrenci 1

Adi=veli
Numarasi=20

Ogrenci 2

Adi=Ahmet
Numarasi=30

Process exited after 0.2471 se
Press any key to continue . .

YAPILARIN FONKSİYON İLE KULLANIMI

- Bir fonksiyonun geri dönüş değeri veya parametre olarak aldığı değer bir yapı olabilir.

```
struct kisi{  
    char *ad,*soyad;  
}
```

Kisi yapısını kullanarak bir fonksiyon tanımlayabiliriz.

```
struct kisi fonkadi(){  
    struct kisi a;  
    ....  
    return a;  
}
```

```
struct kisi fonk2(struct kisi x){  
    x.ad=...;  
    x.soyad=...;  
    struct kisi a;  
    .....  
    return a;  
}
```

Örnek 3: Kullanıcıdan bugunun tarihini ve doğum tarihini isteyelim. Kaç yıl kaç gün kaç aydır yaşadığını hesaplayan programı yapı dizileri ile oluşturalım.

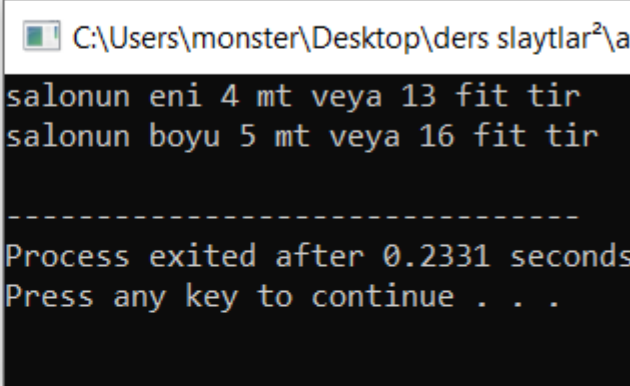
```
1  #include <iostream>
2  using namespace std;
3
4  struct tarih{
5      int gun,ay,yil;
6  };
7
8  struct tarih yashesapla(struct tarih, struct tarih); //protatip
9
10 int main() {
11     tarih bugun,dogum,sonuc;
12     cout<<"Bugunun tarihini gun,ay,yil giriniz:"<<endl;
13     cin>>bugun.gun>>bugun.ay>>bugun.yil;
14     cout<<"Dogum tarihini gun,ay,yil giriniz:"<<endl;
15     cin>>dogum.gun>>dogum.ay>>dogum.yil;
16     sonuc=yashesapla(bugun,dogum);
17     cout<<sonuc.yil<<" yil "<<sonuc.ay<<" ay "<<sonuc.gun<<" gun gecmis";
18     return 0;
19 }
20
21 struct tarih yashesapla(struct tarih x, struct tarih y){
22     tarih sonuc;
23     if(x.gun<y.gun){
24         x.ay=x.ay-1;
25         sonuc.gun=30+x.gun-y.gun;
26     }else{
27         sonuc.gun=x.gun-y.gun;
28     }
29     if(x.ay<y.ay){
30         x.yil=x.yil-1;
31         sonuc.ay=12+x.ay-y.ay;
32     }else{
33         sonuc.ay=x.ay-y.ay;
34     }
35     sonuc.yil=x.yil-y.yil;
36     return sonuc;
37 }
```

```
C:\Users\monster\Desktop\ders slaytlar²\alg
Bugunun tarihini gun,ay,yil giriniz:
19
2
2021
Dogum tarihini gun,ay,yil giriniz:
4
8
2005
15 yil 6 ay 15 gun gecmis
-----
Process exited after 29.26 seconds w
Press any key to continue . . .
```

YAPI İÇİNDE YAPI

Bir yapı içinde başka bir yapı türünde değişkenlerde oluşturabiliriz. Aşağıdaki örnekte oda adındaki yapının içinde uzunluk tipindeki yapı türünden değişkenler kullanılmıştır.

```
1  #include <iostream>
2  using namespace std;
3
4  struct uzunluk{
5      int metre;
6      int fit;
7  };
8
9  struct oda{
10     uzunluk en;
11     uzunluk boy;
12 };
13
14 int main() {
15     oda salon;
16     salon.en.metre=4;
17     salon.en.fit=13;
18     salon.boy.metre=5;
19     salon.boy.fit=16;
20     cout<<"salonun eni "<<salon.en.metre<<" mt veya "<<salon.en.fit<<" fit tir"<<endl;
21     cout<<"salonun boyu "<<salon.boy.metre<<" mt veya "<<salon.boy.fit<<" fit tir"<<endl;
22
23     return 0;
24 }
```



C:\Users\monster\Desktop\ders slaytlar²\al

salonun eni 4 mt veya 13 fit tir
salonun boyu 5 mt veya 16 fit tir

Process exited after 0.2331 seconds
Press any key to continue . . .