



SAKARYA ÜNİVERSİTESİ
Bilgisayar ve Bilişim Bilimleri Fakültesi
Bilgisayar Mühendisliği Bölümü

Veri Tipi Tanımlama



Prof. Dr. Cemil ÖZ
Prof. Dr. Celal ÇEKEN
Doç. Dr. Cüneyt BAYILMIŞ

Konular

- ✓ **Veri Tipi Tanımlama (Enum)**
- ✓ **Veri Tipi Tanımlama (Struct)**
- ✓ **Tipler İçin Kısa Ad (Typedef)**
- ✓ **Struct Veri Tipi**
- ✓ **Kayan Noktalı Sayılar**
- ✓ **Kaynaklar**

Enumerations (Birleştirme veri tipi)

- ✓ Her programlama dilinde tanımlı temel veri tipleri bulunmaktadır.
- ✓ Tanımlı olmayan veri türleri, kullanıcı bazlı tanımlama ile sağlanır.
- ✓ Topluluk ve birleşme gibi iki farklı şekilde yapılabilmektedir.
- ✓ Kendi veri tipinizi tanımlama imkanı sunar.
- ✓ Enum veri tipi, değişkenin alabileceği değerlerin belirli/sabit olduğu durumlarda programın daha anlaşılır olmasını sağlar.

enum *tipAdi* { *isim listesi (deger1, deger2, ...)* } *degiskenAdi*;

```
#include <iostream>
using namespace std;
enum bolumler {bilgisayar, bilisim, yazilim} bolum;
int main()
{
    bolum=bilgisayar;
    cout<<bolum;
    bolum=static_cast<bolumler>(bolum+1);
    cout<<bolum;
    return 0;
}
```

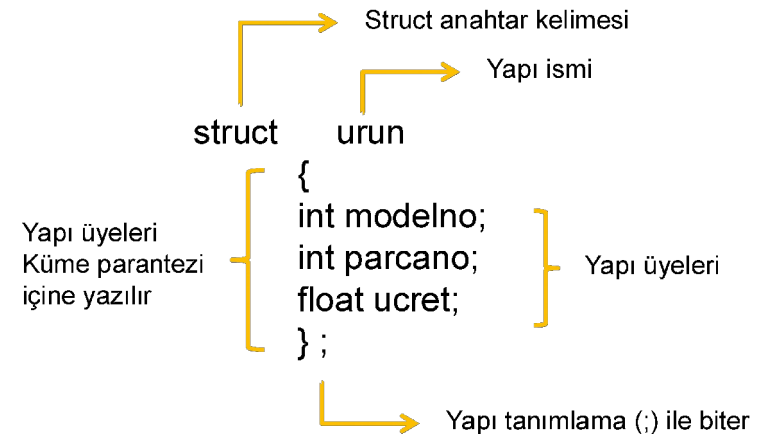
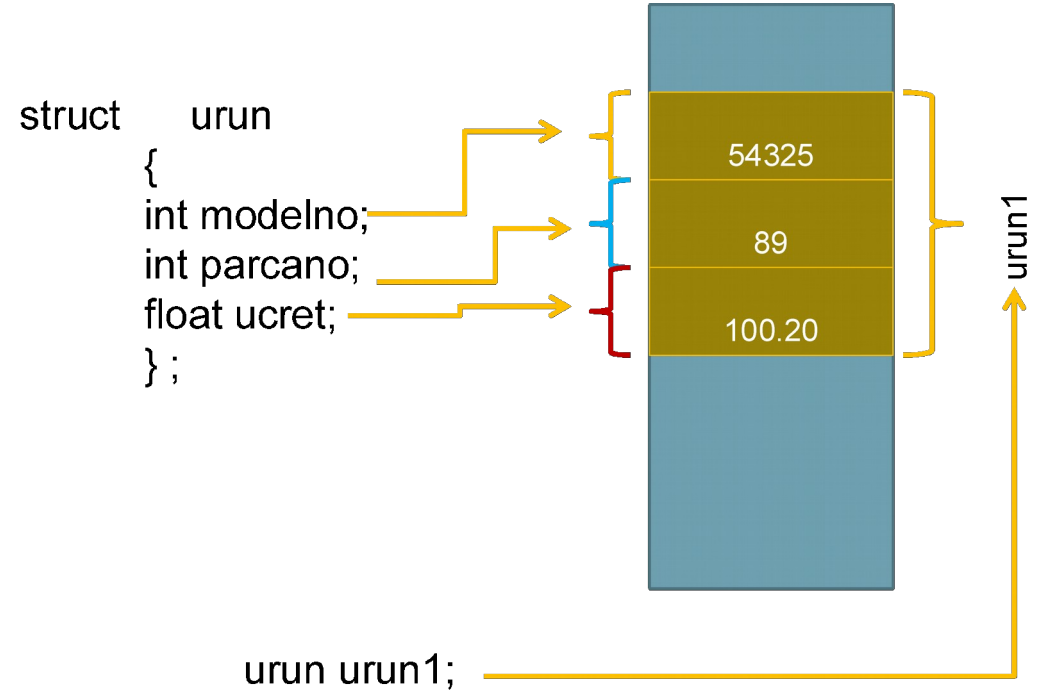
Veri Tipi Tanımlama (Enum)

```
#include <iostream>
using namespace std;
enum yonler {Guneş, Kuzey, Dogu, Bati};
int main()
{
    enum yonler yon;
    int secim;
    cout<<"Yon Giriniz (Guneş=0, Kuzey=1, Dogu=2, Bati=3) :";
    cin>>secim;
    yon=static_cast<yonler>(secim);
    switch(yon)
    {
        case Guneş: cout<<"Guneş";break;
        case Kuzey: cout<<"Kuzey";break;
        case Dogu: cout<<"Dogu";break;
        case Bati: cout<<"Bati";break;
        default: cout<<"hatali secim";
    }
    return 0;
}
```

enum {Guneş= 10, Kuzey, Dogu= 0, Bati};

Veri Tipi Tanımlama (Struct)

- ✓ Yapılar ve sınıflar ilişkili fakat farklı tipe de sahip olabilen verileri tutmak için kullanılırlar.
- ✓ Yapılar, sınıflar ve diziler statik elemanlardır. Programın çalışma süresi boyunca sabit boyuttadırlar.
- ✓ Aynı tipe sahip veri elemanlarının oluşturduğu veri yapılarına (ilişkili veri elemanları topluluğu) dizi denir. Yapılar, diziler ile karıştırılmamalıdır.
- ✓ Yapı içerisindeki bileşenlere ÜYE denir.
- ✓ Her yapı farklı bir isme sahiptir, ancak aynı isimli üyeleri olabilir.

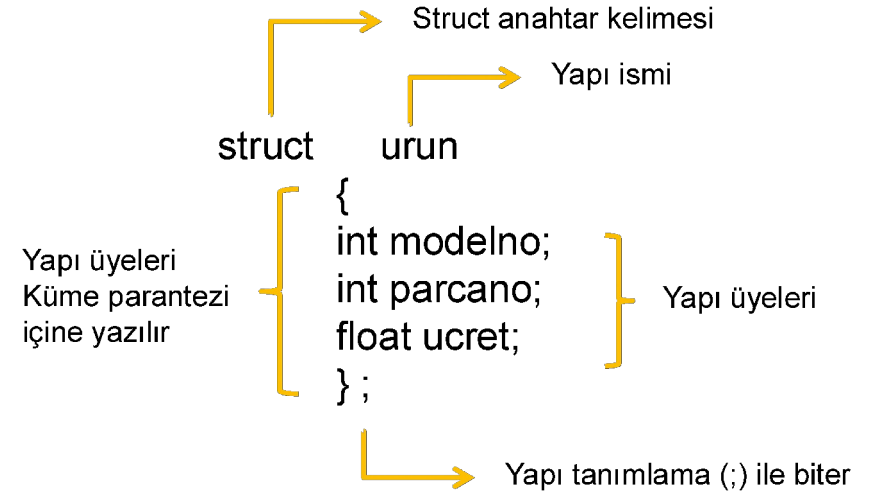


Veri Tipi Tanımlama (Struct)

✓ Yapı tanımı

Yapı tanımı bellekte yer ayırmaz. Yapı tanımı ile yapı değişkeni tanımlanmış olmaz.

Yapı tanımı ise sadece, yapı değişkenlerinin tanımlandıkları zaman nasıl görüneceklerini gösteren bir modeldir.



✓ Yapı değişkenin tanımlanması

`urun urun1; // urun1 için bellekte yer ayrılır`

✓ Yapı üyelerine değer atanması

Yapı içerisindeki üyeye '.' (üye erişim operatörü) ile erişilir.

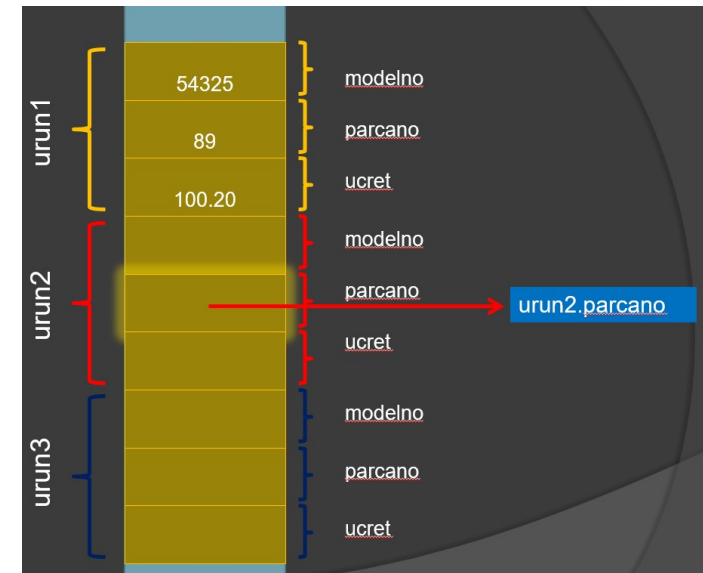
`yapidegiskeni.uyeadı`

`urun1.modelno=2016`

`urun urun1={5235, 98, 153.15f};`

`urun urun2; // urun2 yapı değişkeni tanımlanır.`

`urun2= urun1; //ilk yapı değişkenindeki değerleri ikinci değişkene ata`



Struct Veri Tipi

```
#include <iostream>
using namespace std;
struct ogrenci {           // yapı tanımlama
    string ad;
    string soyad;
    int vize;
    int final;
};
int main()
{
    struct ogrenci ogrNot;           // yapı değişkeni tanımlanır
    double basari;
    cout<<"Ogrencinin adini giriniz"<<endl;
    cin>>ogrNot.ad;                  // yapı üyesine değer atama
    cout<<endl<<"Ogrencinin soyadini giriniz \n";
    cin>>ogrNot.soyad;
    cout<<endl<<"Vize notunu giriniz \n";
    cin>>ogrNot.vize;
    cout<<endl<<"Final notunu giriniz \n";
    cin>>ogrNot.final;
    cout<<endl<<"Ogrenci Bilgileri\n";
    cout<<ogrNot.ad<<"\t"<<ogrNot.soyad<<"\t"<<ogrNot.vize<<"\t"<<ogrNot.final<<endl;
    basari=0.4*ogrNot.vize + 0.6*ogrNot.final;
    if (basari >50 )
        cout<<"basari notunuz:"<<basari<<"\t"<<"Gectiniz \n";
    else
        cout<<"basari notunuz:"<<basari<<"\t"<<"Seneye \n";
    system("pause");
    return 0;
}
```

Ogrenci.cpp

İç İçe Yapılar

✓ İç içe yapı değişkenine başlangıç değeri atama

takvim t = { 2016, 10, 20, {09, 33} };

```
#include <iostream>
using namespace std;
```

```
struct sure {                // yapı tanımlama
    int saat;
    int dakika;
};
```

```
struct takvim {              // yapı tanımlama
    int yıl;
    int ay;
    int gün;
    struct sure tarih;        // içice yapı k
};
```

```
main(){
```

```
    struct takvim t;
    cout<<"yıl ay gün saat dakika";
    cin>>t.yıl>>t.ay>>t.gün;
    cin>>t.tarih.saat>>t.tarih.dakika;
    cout<<t.gün<<t.ay<<t.yıl<<"\t"<<t.tarih.saat<<":"<<t.tarih.dakika;
    system("Pause");
    return 0;
```

```
} Etkinlik.cpp
```

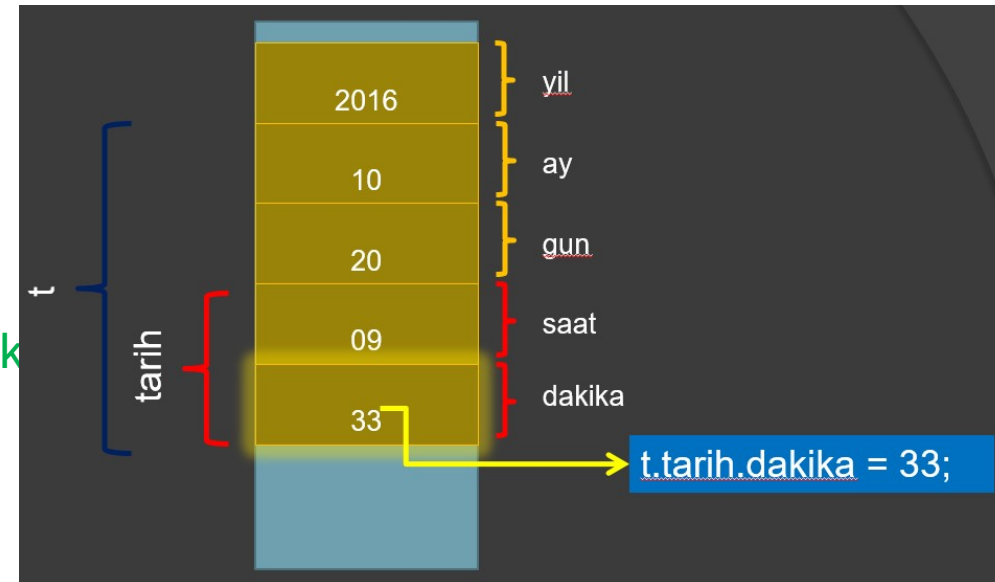
EnumStruct.cpp

Olcu.cpp

Olcu1.cpp

Olcu2.cpp

8



Tipler İçin Kısa Ad (typedef)

- ✓ Yazılan kodların genelleştirilmesinde kullanılır. (Farklı tipler için aynı kodların kullanılabilmesi)
- ✓ Yazımı karmaşık, uzun ve zor olan tip tanımlarının daha basit ve anlaşılır olmasını sağlamak amacıyla kullanılabilir.

```
typedef   veritipi   yeniTipAdi;
```

```
typedef   int   tamsayi;
```

```
tamsayi   x, y;   // x ve y int tipindedir
```

```
typedef double YigitVeriTipi;  
  
enum HataKodu { basarili, bos, dolu};  
  
const int mAXeLEMANsAYISI = 10;  
  
class Stack  
{  
    private:  
        int elemanSayisi;  
        YigitVeriTipi *depolamaBirimi;  
    public:  
        Stack();  
        ~Stack();  
        bool empty() const;  
        HataKodu pop();  
        HataKodu top(YigitVeriTipi &deger) const;  
        HataKodu push(const YigitVeriTipi &deger);  
};
```

Birlik (union)

- ✓ Birlikler yapılara benzer
- ✓ Yapıların kullanılmayan üyelerinden doğan verimsizliği ortadan kaldırmak için kullanılır.
- ✓ Aynı depo alanını farklı değişkenler ortak kullanırlar.
- ✓ Üyeler herhangi bir veri türü olabilir.
- ✓ Birlik en az, en büyük alana sahip üyeyi içerecek kadar kapasiteye sahip olmalıdır
- ✓ Bir anda sadece bir üyeye erişilebilir.
- ✓ Başlangıç değeri ilk üyenin türü ile olmalıdır.
- ✓ Başlangıçta sadece ilk üye için atama yapılabilir.

```
union sayi {  
    int x;  
    int y;  
};
```



```
union sayi deger={10};           //geçerli  
union sayi deger={1.23};         //geçersiz
```

Kayan Noktalı Sayılar

Kayan noktalı sayılar, hassasiyet belirtilmemişse, varsayılan olarak 6 basamak hassasiyetle gösterilirler(32 bitlik işletim sistemleri için)

Kaç basamak gösterileceğini belirtmek için **fixed** ve **setprecision** manipulatorleri kullanılır. **fixed** ifadesi kullanılmaz ise toplam basamak sayısı, kullanılır ise virgülden sonraki basamak sayısı belirtilmiş olur.

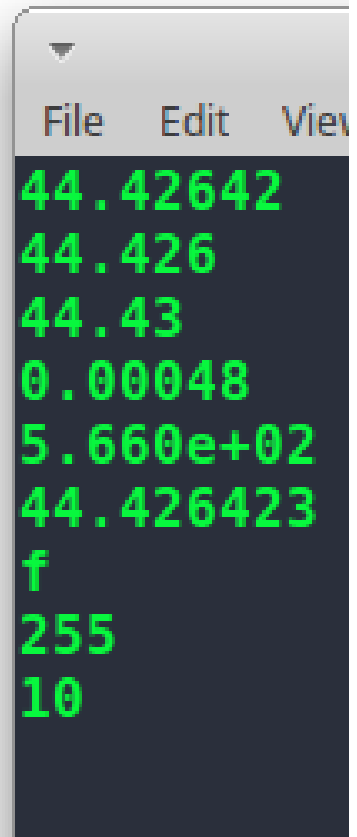
endl gibi bu manipulatorde iomanip kitaplığına ihtiyaç duyar.

Bilimsel gösterim çok büyük ya da çok küçük sayıların gösteriminde tercih edilir. Sayılar 10 üssü olarak ifade edilirler.

Kayan Noktalı Sayılar

```
int main()
{
    double sayi1,sayi2,sayi3;
    sayi1=44.426423423;
    sayi2=48e-5;
    sayi3=566.01;

    cout << setprecision(7)<<sayi1<<endl;
    cout << setprecision(5)<<sayi1<<endl;
    cout << fixed<<setprecision(2)<<sayi1<<endl;
    cout << fixed<<setprecision(5)<<sayi2<<endl;
    cout << scientific<<setprecision(3)<<sayi3<<endl;
    cout << fixed<<setprecision(6)<<sayi1<<endl;
    cout << hex << 15 <<endl;
    cout << dec << 0xff <<endl;
    cout << oct << 8 <<endl;
    return 0;
}
```



```
File Edit View
44.42642
44.426
44.43
0.00048
5.660e+02
44.426423
f
255
10
```

Kaynaklar

- ✓ Robert Lafore, Object Oriented Programming in C++, Macmillan Computer Publishing
- ✓ Deitel, C++ How To Program, Prentice Hall
- ✓ Prof. Dr. Cemil ÖZ, Programlamaya Giriş Ders Notları
- ✓ Prof. Dr. Celal ÇEKEN, Programlamaya Giriş Ders Notları