

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

FONKSİYONLARIN ASIRI YÜKLENMESİ

Fonksiyonların Aşırı Yüklenmesi

- Bir bilim adamına soruyorlar, Sizce dünyanın en büyük icadı nedir?
- Bilim adamın cevap veriyor:
 - Termos
 - -İçerisine sıcak bir şey koyduğunuzda sıcak tutuyor.
 - -İçerisine soğuk bir şey koyduğunuzda soğuk tutuyor.
- İşte aşırı yükleme olayında böyle bir mantık söz konusu. Bir fonksiyonumuz var ve bu fonksiyon
 - Duruma göre bazen X görevini yerine getirirken.
 - Bazen Y görevini yerine getiriyor.

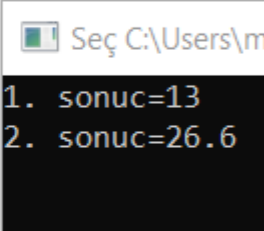
Fonksiyonların Aşırı Yüklenmesi

- Peki hangi görevi hangi durumda kullanıcağın nasıl anlıyor? Şimdi bunu açıklayalım.
 1. En az iki fonksiyon olmalı.
 2. Bu fonksiyonlar aynı isme sahip olmalı.
 3. Yine bu fonksiyonlar farklı argümanlara parametrelere sahip olmalılar.
- Yukarıdaki 3 maddeyi de sağlayan fonksiyonlar oluşturduğumuzda bu duruma **fonksiyonların aşırı yüklenmesi** deniliyor.

NOT: Aslında siz farkında olmadan yapıcı fonksiyonlar konusunda yapıcı fonksiyonları aşırı yükledik. Bir sınıf için bazı örneklerde birden çok yapıcı fonksiyon yazdık ve fonksiyon çağrıldığında parametrelere bakılıp uygun fonksiyon işletiliyordu. 3

ÖRNEK-1

```
1  #include <iostream>
2  using namespace std;
3
4  int topla(int sayi1, int sayi2){
5      return sayi1+sayi2;
6  }
7
8  double topla(double sayi1, double sayi2){
9      return sayi1+sayi2;
10 }
11
12 int main() {
13
14     int a1,a2;
15     double b1,b2;
16
17     a1=5; a2=8;
18     b1=10.8; b2=15.8;
19
20     int sonuc1=topla(a1,a2);
21     double sonuc2=topla(b1,b2);
22
23     cout<<"1. sonuc="<<sonuc1<<endl;
24     cout<<"2. sonuc="<<sonuc2<<endl;
25
26     return 0;
27 }
```



- İnt tipinde topla() adında bir fonksiyon oluşturduk ve int türünde 2 parametre kabul etti ve return ile bu parametreleri toplayıp geri döndürdü.
- Bir sonraki fonksiyonuda yine aynı isimde kullandık. Fakat parametreleri ve tipini double yaptık.
- Programda 20. satırda ilk çağırdığımız yerde topla(a1,a2) diyerek parametreleri int verdik ve derleyeci uygun topla fonksiyonuna gidip işlemi yaptı.
- Benzer şekilde 21. satırda parametreleri double verdik derleyici bu kez double için uygun olan topla fonksiyonuna gitti.

Bu örnekten çıkarılması gereken sonuç, aşırı yüklenmiş bir fonksiyon çağırılacağı zaman derleyici çağrıda belirtilen fonksiyonların argümanlarının (parametrelerin) özelliklerine (sayısı,tipi,sırasını) inceliyor ve ona göre uygun olan fonksiyonla işlem yapıyor.

ÖRNEK-2

```
1  #include <iostream>
2  using namespace std;
3
4  void kare_al(int sayi){
5      cout<<"integer sayinin karesi="<<sayi*sayi<<endl;
6  }
7
8  void kare_al(float sayi){
9      cout<<"float sayinin karesi="<<sayi*sayi<<endl;
10 }
11
12 int main() {
13
14     int a;
15     float b;
16
17     cout<<"integer bir sayi girin:";
18     cin>>a;
19
20     cout<<"float bir sayi girin:";
21     cin>>b;
22
23     kare_al(a);
24     kare_al(b);
25
26     return 0;
27 }
```

C:\Users\monster\Desktop\ders sl

```
integer bir sayi girin:5
float bir sayi girin:2.5
integer sayinin karesi=25
float sayinin karesi=6.25
```

```
-----
Process exited after 8.753 se
Press any key to continue . .
```

ÖRNEK-3

```
1  #include <iostream>
2  using namespace std;
3
4  void fonk(int a){
5      cout<<"tek parametrelili int alan fonk. icindeyiz"<<endl;
6      cout<<"a="<<a<<endl;
7  }
8
9  void fonk(int a,int b){
10     cout<<"iki parametrelili int alan fonk. icindeyiz"<<endl;
11     cout<<"a="<<a<<endl;
12     cout<<"b="<<b<<endl;
13 }
14
15 void fonk(double a){
16     cout<<"tek parametrelili double alan fonk. icindeyiz"<<endl;
17     cout<<"a="<<a<<endl;
18 }
19
20 int main() {
21     fonk(3);
22     fonk(5, 8);
23     fonk(3.14);
24
25     return 0;
26 }
27
```

C:\Users\monster\Desktop\ders slaytlar²\algoritma2\de

```
tek parametrelili int alan fonk. icindeyiz
a=3
iki parametrelili int alan fonk. icindeyiz
a=5
b=8
tek parametrelili double alan fonk. icindeyiz
a=3.14
```

Varsayılan argüman değerleri ile işlem yapma

- İstenildiği takdirde varsayılan argümanlarla da işlem yapılabilir.
- 3 parametrelili bir fonksiyon düşünelim.
- Biz bu fonksiyonu çağırırken 2 adet parametre değerini girersek program bu 2 parametreyi bizim girdiklerimize göre yazacak.
- Ancak üçüncü argümanı varsayılan olarak fonksiyonu tanımladığımız yerdeki argümanın değerinden alacak.

ÖRNEK-4

```
1  #include <iostream>
2  using namespace std;
3
4  void fonk(int a=0, float b=3.14, char c='m', string d="MTU"){
5      cout<<a<<"", "<<b<<"", "<<c<<"", "<<d<<endl;
6  }
7
8
9  int main() {
10
11      fonk();
12      fonk(5);
13      fonk(5,2.3);
14      fonk(5,2.3,'z');
15      fonk(5,2.3,'z',"C++");
16
17      return 0;
18  }
```

C:\Users\monster\Desktop\de

```
0, 3.14, m, MTU
5, 3.14, m, MTU
5, 2.3, m, MTU
5, 2.3, z, MTU
5, 2.3, z, C++
```

```
-----
Process exited after 0.041
Press any key to continue
```


ÖRNEK-5 Yapıcı fonksiyonlarda aşırı yüklenme

Önceki derslerde aslında siz farkında olmadan aşırı yükleme yapıyorduk. Hatırlatma amaçlı 1 örnek yapalım.

```
1  #include <iostream>
2  using namespace std;
3
4  class sinifim{
5      int sayi1,sayi2;
6
7      public:
8
9      sinifim(){
10         sayi1=1;
11         sayi2=2;
12     }
13
14     sinifim (int a,int b){
15         sayi1=a+3;
16         sayi2=b+3;
17     }
18
19     void goruntule(){
20         cout<<"sayi1="<<sayi1<<endl;
21         cout<<"sayi2="<<sayi2<<endl<<endl;
22     }
23 };
24
25 int main() {
26
27     sinifim nesne1;
28     sinifim nesne2(10,20);
29
30     nesne1.goruntule();
31     nesne2.goruntule();
32
33     return 0;
34 }
```



C:\Users\monst

sayi1=1
sayi2=2

sayi1=13
sayi2=23

Process exited
Press any key to