

6. Fonksiyonların Aşırı Yüklenmesi(Function overloading) ve çok biçimlilik

C++ da isimleri aynı yaptıkları işler farklı fonksiyonlar yazılabilir. Buna göre iki farklı fonksiyon, parametrelerinin tipleri, sayıları veya sıraları farklı olmak koşulu ile aynı isme sahip olabilirler. Derleyici hangi fonksiyonun çağrıldığını parametre listesine bakarak karar verir. Aynı isimde birden fazla fonksiyonun tanımlanmasına fonksiyonların aşırı yüklenmesi (function overloading) denir. Aşırı yüklenmiş fonksiyonlar, birbiriyle alakalı işlemlerin aynı adla çağrılmasına izin vererek programın karmaşıklığını azaltabilirler.

Çok biçimlilik (polimorfizm), genel anlamda bir adın, birbirleriyle ilişkili fakat teknik açıdan farklı iki veya daha fazla amaç için kullanılabilmesi yeteneğidir.

C++'da çok biçimlilik özelliği, fonksiyonlara yeni görev yükleme veya bir başka değişle fonksiyonlara aşırı yükleme (function overloading) ile sağlanabilir. Aşağıdaki programı inceleyiniz.

```
#include <iostream.h>
```

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

```
void yaz(); //Bildirimler
```

```
void yaz(char);
```

```
void yaz(char, int);
```

```
int main()
```

```
{
```

```
yaz();
```

```
yaz('=');
```

```
yaz('+', 30);
```

```
getch();
```

```
return 0;
```

```
}
```

```
//-----
```

```
// 45 defa * yazar
```

```
void yaz()
```

```
{
```

```
for(int j=0; j<45; j++) // her zaman 45 kez döner
```

```
cout << '*';
```

```
cout << endl;
```

```
}
```

```
//-----
```

```
// belirtilen karakteri 45 kez yazar
```

```
void yaz(char ch)
```

```
{
```

```
for(int j=0; j<45; j++)
```

```
cout << ch;
```

```
cout << endl;
```

```
}
```

```
//-----
```

```
// Belirtilen karakteri n kez yazar
```

```
void yaz(char ch, int n)
```

```
{
```

```
for(int j=0; j<n; j++) // n defa döner
cout << ch;
cout << endl;
}
```



Operatörlerin Aşırı Yüklenmesi(Operator Overloading)

C++'da, +, -, *, !, ++ gibi operatörlere fonksiyonlar yazarak bu operatörlere yeni işlevler yüklenebilir. Sadece C++'da zaten var olan operatörler aşırı yüklenebilir. Örneğin C++'da var olmayan '^' operatörü için üs almak üzere bir fonksiyon yazılamaz. Fonksiyonlar tanımlanırken operatörün orijinalinde var olan operand sayısı değiştirilemez. Örneğin '+' operatörü iki değişken değerini toplar. Değişken sayısı tek olamaz. Ayrıca daha önceki konularda geçtiği gibi operatörlerin önceki sırası değiştirilemez. Bir 4, -, *, ! gibi bir işlem operatörlerine 'operator' sözcüğü ile yükleme yapılır. Örneğin toplama operatörü '+' operatörüne "operator+" şeklinde isim verilerek yükleme yapılır. Operatör aşırı yüklemesi aşağıda gösterildiği gibi yapılır.

Dönüş tipi operator operatörismi(parametre listesi)

```
{
// gerçekleştirilecek işlem
}
```


Örnek olarak aşağıda verile programda '+' operatörü iki karmaşık sayıyı toplamak için aşırı yüklenmektedir.

```
#include <iostream.h>
#include <conio.h>
struct KarmasikSayi{ // Karmaşık sayıları tanımlamak için bir yapı
float reel,sanal; // Reel ve sanal kısımlar
};
// + operatörünün karmaşık sayıları toplamak için yüklenmesi
KarmasikSayi operator+ (const KarmasikSayi &v1, const KarmasikSayi &v2)
{
KarmasikSayi sonuc; // sonucun yazılacağı yerel değişken
sonuc.reel=v1.reel+v2.reel; // reel kısımlar toplanıyor
sonuc.sanal=v1.sanal+v2.sanal; // sanal kısımlar toplanıyor
return sonuc; // sonuç çağırılan programa döndürülüyor
}
```

```
void yaz (KarmasikSayi c){ //kompleks sayıları yazan print fonksiyonu
cout<< "reel= " << c.reel << " sanal= " << c.sanal << endl;
}
```

```
int main()
{
KarmasikSayi c1,c2,c3; // Üç adet karmaşık sayı tanımlanıyor
```

```
c1.reel= 3;  
c1.sanal= -1;  
c2.reel= 2.5;  
c2.sanal= 0.7;  
c3= c1 + c2; // Burada + iki karmaşık sayıyı topluyor  
yaz(c3);  
getch();  
return 0;  
}
```



```
reel= 5.5 sanal= -0.3  
_
```