



SAKARYA ÜNİVERSİTESİ
Bilgisayar ve Bilişim Bilimleri Fakültesi
Bilgisayar Mühendisliği Bölümü

İşlevler (Functions)



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

Konular

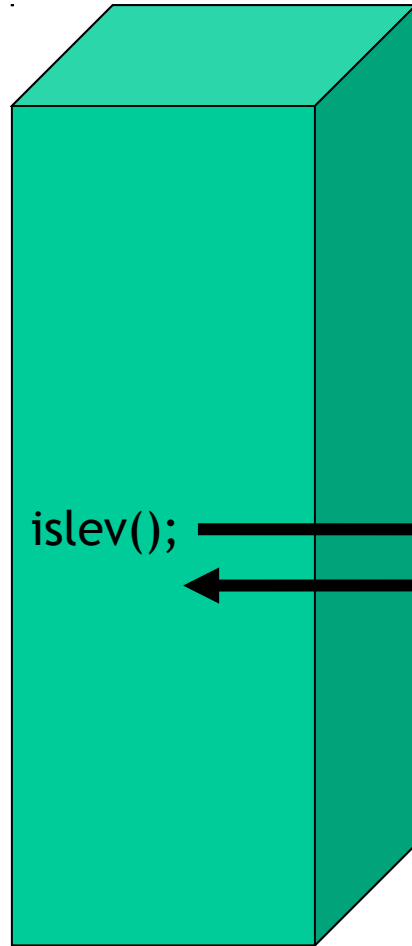
- ✓ İşlevler (Functions)
- ✓ Fonksiyon Tanımlama
- ✓ İşlevlerin Aşırı Yüklenmesi (Function Overloading)
- ✓ Scope (Kapsam)
- ✓ Inline Fonksiyonlar
- ✓ Yapılar - Fonksiyonlar
- ✓ Özyinelemeli (Recursive) Fonksiyonlar
- ✓ Fonksiyon Çağrılar
- ✓ Sorular
- ✓ Kaynaklar

İşlevler (Functions)

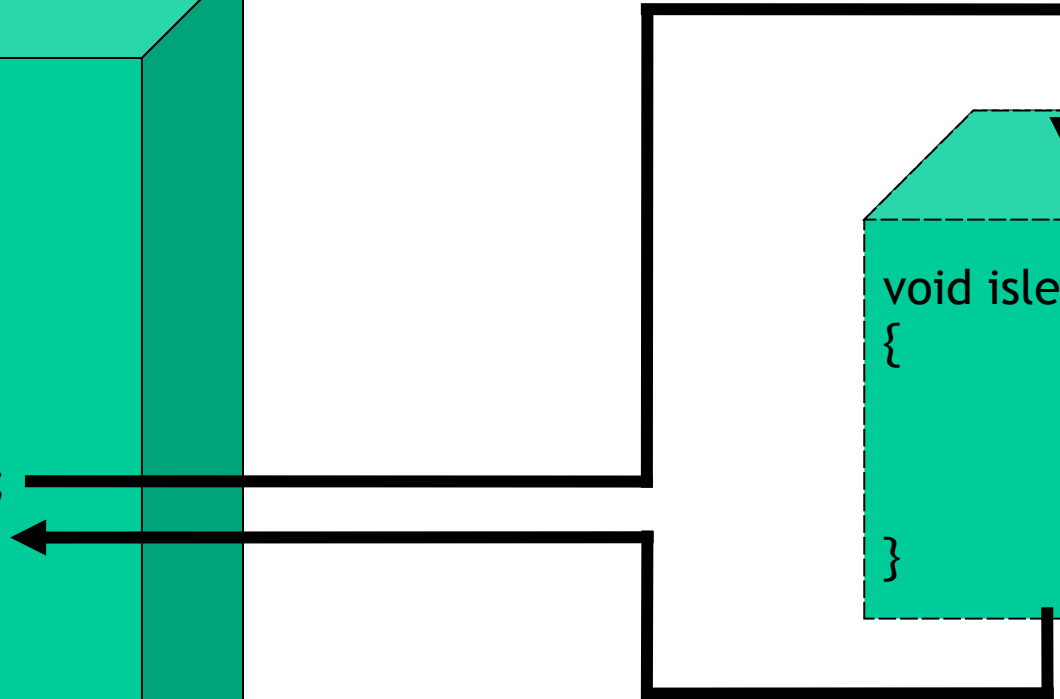
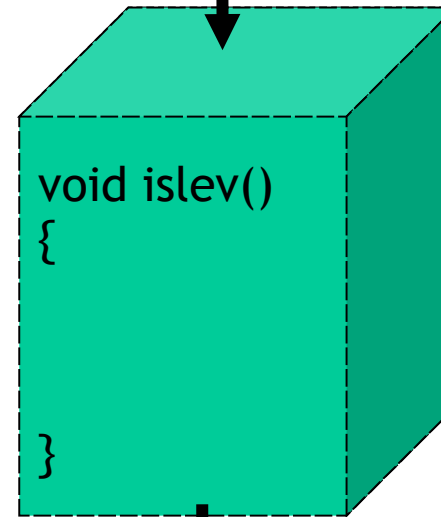
- ✓Büyük problemler, küçük parçalara bölünerek kolay çözülür.
- ✓Bu parçalara işlev, alt program, yöntem ya da procedure denilmektedir.
- ✓Yazılımlar benzer modüller içerirler. Dolayısıyla program içerisinde tekrar eden işlemlerin her defasında yeniden yazılması engellenerek program geliştirme kolaylaştırılır.
- ✓Hata ayıklama küçük ölçekte daha kolaydır.
- ✓Küçük parçalara yoğunlaşmak kolaydır.
- ✓Parçalara ayrılan problem çok sayıda insan tarafından paralel olarak çözülebilir.
- ✓Yapısal programlama fonksiyonlarla mümkündür.
- ✓Daha güvenli ve verimli kod üretimi sağlanır.
- ✓Ayrıca program boyutları da nispeten küçülür.

İşlevler (Functions)

İşlevi Çağırın Program



İşlev



Fonksiyon Tanımlama

```
dönüş_tipi  fonksiyon_ismi (parametre_listesi){  
    gerekli değişken tanımlamaları;  
    fonksiyon gövdesi;  
    return geriDonusDegeri/Degiskeni;  
}
```

Dönüş değeri. Yöntemden geriye dönen değerın tipi yazılır. Geriye değeri dönmeyecek ise **void** yazılmalıdır. Void kullanımı durumunda fonksiyon return komut satırı içermez. Fonksiyonda geri dönüş değeri/değişkeni **return** komut satırı ile belirtilir.

Fonksiyon adı. Yöntemin adını belirler. Değişken isimlendirme kuralları aynen geçerlidir.

Parametre listesi. Yöntem içerisinde kullanılacak giriş parametreleri tanımlanır.

Örnek fonksiyon tanımlamaları:

float aylıkMaas(float mesaiSaati, float saatUcreti)

float bol (int x, float f)

```
int kareAl (int x) {  
    return x*x;  
}
```

```
void mesaj (int mesajNo) {  
    switch(mesajNo){  
        case 0: cout<<"Mesaj 0";break;  
        case 1: cout<<"Mesaj 1";break;  
        default:cout<<"Tanımsız Durum";  
    }  
}
```

İşlevler (Functions)

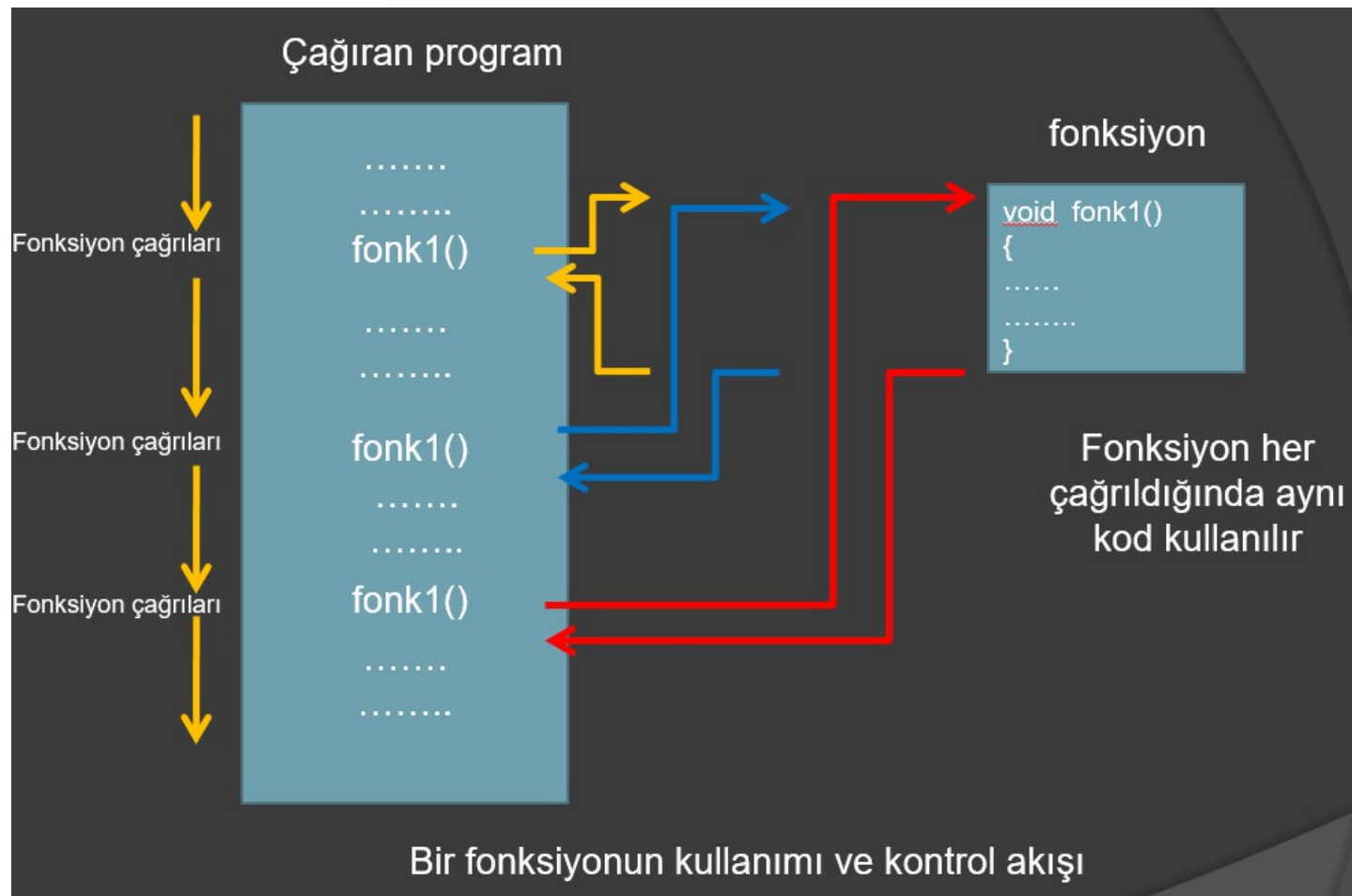
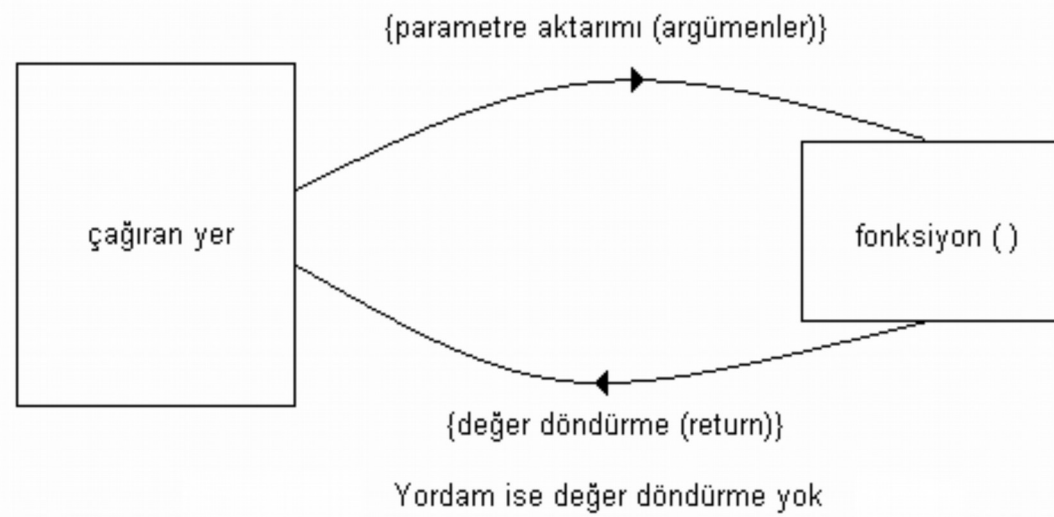
- ✓Fonksiyon tanımı çağıran fonksiyondan önce yapılmalıdır.
- ✓Eğer sonra yapılacaksa, tanıtım (prototip) mutlaka çağıran fonksiyondan önce yapılmalıdır.

dönüş_tipi fonksiyon_ismi (parametre veri tipleri);

Örnek: int kareAl (int);

- ✓Diğer bir deyişle; bir C/C++ programında fonksiyonlar ana fonksiyon olan main() fonksiyonundan önce veya sonra tanımlanabilir. Kullanılacak fonksiyon main() fonksiyonundan önce tanımlanır ise bu durumda fonksiyonun ön bildirimine ihtiyaç yoktur.
- ✓Fakat özellikle büyük boyutlu programlar oluştururken programın okunulabilirliği açısından main() fonksiyonundan önce ön bildirim yapıp fonksiyonu main() fonksiyonundan sonra tanımlamak daha uygundur.
- ✓Argümanlar yerel değişkenlere benzerler ve sadece fonksiyon içerisinde tanınırlar.

İşlevlerin Çalışması



İşlevlere Parametre Aktarımı

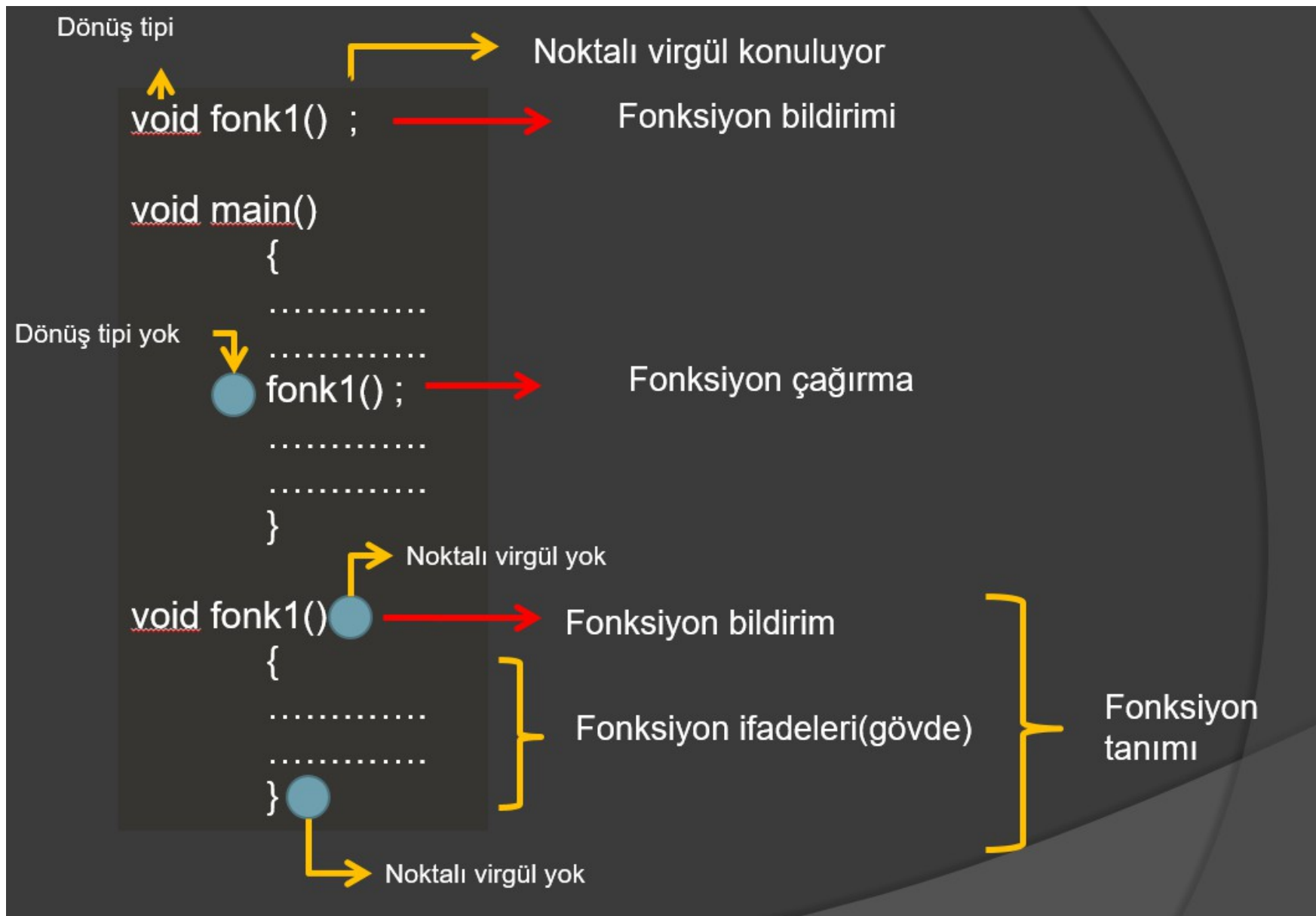
- ✓ 'C/C++' da bir fonksiyonuna parametre aktarımı (argüman kullanımı) **iki** şekilde yapılabilir;
 - ✓ Verinin değeri doğrudan aktarılabilir yani değer doğrudan parametre olarak kullanılabilir (**değer ile çağırma - calling by value**).
 - ✓ Verinin adresi (değişken) aktarılabilir yani değişken parametre olarak kullanılabilir (**adres ile çağırma - calling by reference**).
- ✓ Bunun ile birlikte fonksiyon argümanlarının bir kısmı **değer** bir kısmı da **değişken** şeklinde olabilir.

İşlevler (Functions)

✓ Prototip tanımlamamak için void ciz () fonksiyon bloğu main'den önce yazılmalıdır.

```
#include <iostream>
using namespace std;
void ciz();           // prototip tanımlama
int main()           // ana fonksiyon
{
    ciz();           // fonksiyon çağırma
    cout << "Veri Tipi   Aralık" << endl;
    ciz();           // fonksiyon çağırma
    cout << "char       -128 -- 127" << endl
         << "short      -32,768 -- 32,767" << endl
         << "int         Sisteme Bagli" << endl
         << "long        -2,147,483,648 -- 2,147,483,647" << endl;
    ciz(); // fonksiyon çağırma
    system("pause");
    return 0;
}
void ciz()           // fonksiyon tanımlama
{
    for(int j=0; j<45; j++)
        cout << '*';
    cout << endl;
}
```

İşlevler (Functions)



İşlevlere Parametre Aktarımı

```
#include <iostream>
using namespace std;
void ciz (char,int);          // prototip tanımlama
int main()                   // ana fonksiyon
{
    int sayi;
    char karakter;
    cout<<"basilacak karakter sayisi";
    cin>>sayi;
    cout<<"karakteri giriniz";
    cin>>karakter;
    ciz (karakter,sayi);      // fonksiyon çağırma
    cout << "Veri Tipi      Aralık" << endl;
    ciz('&',40);              // fonksiyon çağırma
    cout << "char          -128 -- 127" << endl
        << "short          -32,768 -- 32,767" << endl
        << "int            Sisteme Bagli" << endl
        << "long           -2,147,483,648 -- 2,147,483,647" << endl;
    ciz('-',30);
    system("pause");
    return 0;
}
```

```
void ciz (char ch, int n) // fonksiyon tanımlama
{
    for(int j=0; j<n; j++)
        cout << ch;
    cout << endl;
}
```

2ciz1.cpp 3ciz2.cpp

İşlevlerin Aşırı Yüklenmesi (Function Overloading)

Aynı isme sahip fonksiyonun aldığı parametrelere göre farklı şekilde çalışmasıdır.

```
float aylikMaas(float mesaiSaati, float saatUcreti)
```



İşlevin imzası: Aynı sınıf içerisindeki yöntemlerin imzası farklı olmalı

```
double aylikMaas(double mesai, float saatUcreti)
```

```
double aylikMaas(double m)
```

```
double aylikMaas()
```

İşlevlere Parametre (Argüman) Olarak Yapıların Aktarımı

```
#include <iostream>
using namespace std;
struct Olcu
{
    int metre;
    int cmetre;
};
void olcuGoster(Olcu);
int main()                // ana fonksiyon
{
    int sayi;
    Olcu d1,d2;
    cout << "\n uzunluk(metre) giriniz: ";
    cin >> d1.metre;
    cout << "uzunluk(cmetre) giriniz:: ";
    cin >> d1.cmetre;
    olcuGoster(d1);
    cout << "\nuzunluk(metre) giriniz: ";
    cin >> d2.metre;
    cout << "uzunluk(cmetre) giriniz:: ";
    cin >> d2.cmetre;
    olcuGoster(d1);
    olcuGoster(d2);
    system("pause");
    return 0;
}
```

```
void olcuGoster (Olcu a) // fonksiyon tanımlama
{
    cout << a.metre << "m, " << a.cmetre << "cm \n";
}
```

5Olcu.cpp 6Olcu1.cpp

İşlevlerden Değer Döndürmek

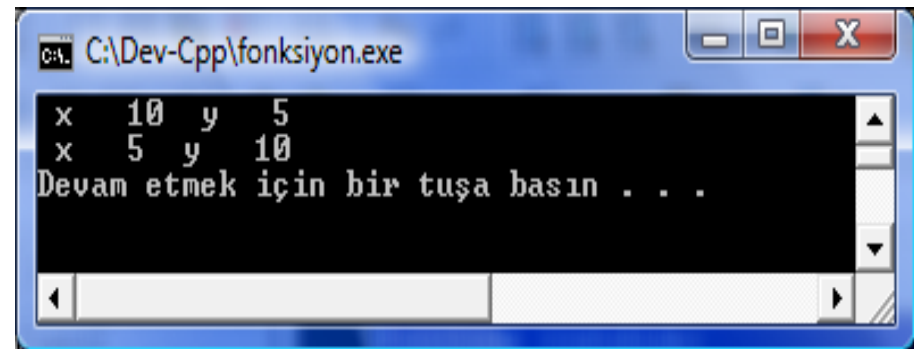
Bir fonksiyon çalışmasını tamamladıktan sonra kendisini çağıran programa tek bir değer döndürebilir. Genellikle döndürülen bu değer, Fonksiyonun çözdüğü problemin cevabını içerir.

```
#include <iostream>
using namespace std;
float birimcevir (float);
int main()
{
    float libre, kilo;
    cout<< "\n    kilonuzu kg olarak giriniz ..";
    cin>>kilo;
    libre=birimcevir(kilo);
    cout<<"  libre cinsinden ağırlığınız
    ..="<<libre<<endl;
    return 0;
}
float birimcevir (float agirlikKg)
{
    float agirlikLb=agirlikKg/0.453592;
    return agirlikLb;
```

4cevir.cpp

İşlevlerden Return Kullanmadan Değer Döndürmek

```
#include <cstdlib>
#include <iostream>
using namespace std;
void swap(int , int );
int x=10; int y=5;
int main(int argc, char *argv[])
{
    cout<<" x "<< x<<" y "<< y<< endl;
    swap(x,y);
    cout<<" x "<< x<<" y "<< y<< endl;
    system("PAUSE");
    return EXIT_SUCCESS;
}
void swap(int first, int second)
{
    x=second;
    y=first;
}
```



Değişken Kapsamları

Yerel (local) değişkenler

Global değişkenler

Static değişkenler

Extern

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

```
int x=4,y=4;
```

```
//global değişkenler
```

```
int f1(int a, int b){
```

```
    int x=3;
```

```
//f1 in yerel x değişkeni
```

```
    int y;
```

```
    .....
```

```
}
```

```
void f2(void){
```

```
    int x=4;
```

```
//f2 nin yerel x değişkeni
```

```
    static int y=2;
```

```
//f2 nin statik değişkeni
```

```
    .....
```

```
}
```

```
main(){
```

```
    int x=5;
```

```
//main fonk. yerel x değişkeni
```

```
    y=10;
```

```
    .....
```

```
}
```

Kapsam.cpp UnaryScopeResulation.cpp

Değişken Kapsamları

Static Değişken Kullanımı

Not: Aynı örneği, static ifadelerini kaldırarak çalıştırınız.

```
#include <iostream.h>
float ortal (int sayi)
{
    static float toplam=0;
    static int say=0;
    say++;
    toplam+=sayi;
    return toplam/say;
}
int main()
{
    int sayi=1;
    float ort;
    while (sayi !=0)
    {
        cout<<" sayi giriniz: ";
        cin>>sayi;
        cout<<"Ortalama : "<<ortal(sayi)<<endl;
    }
    system("pause");
    return 0;
}
```

Inline Fonksiyonlar

- ✓ Program hızlanır, fakat boyutu artar...
- ✓ Normal fonksiyon tanımının başına inline ifadesi yazılması yeterli
- ✓ Özellikle sık kullanılan ve küçük fonksiyonlar için tercih edilir.

```
inline void ciz(char ch, int n)  
//function declarator
```

```
{  
    for(int j=0; j<n; j++)  
body    cout << ch;           //function  
    cout << endl;  
}
```

Özyinelemeli (Recursive) Fonksiyonlar

- ✓ Fonksiyonun kendi kendini çağırması ile döngüsel yapı kurulur.
- ✓ Seri hesaplamalar, döngüsel hesaplama zorluklarında kullanılır.
- ✓ Özyineli yöntemler problemi daha gerçekçi ifade ettiği için tercih edilir.
- ✓ Aynı problemler döngü kullanılarak da (iteratif) çözülebilir.
- ✓ Özyinelemeli çağrılar zaman alır ve ek bellek tüketimine neden olur. Bu nedenle performans durumunda özyinelemeli fonksiyon kullanımından kaçınılmalıdır.

```
Fonksiyon (int k)
{
    ....
    if (k<0)
        return a;      // dönüş noktası
    Fonksiyon(k-1);    //öz yineleme
    ....
}
```

Not: Dönüş noktası kullanılmazsa sonsuz döngü kurulmuş olur

Özyinelemeli (Recursive) Fonksiyonlar

✓ Faktöriyel Hesabı $n! = n * (n-1)!$

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

```
long fakto (int x){
```

```
    if(x<=1)
```

```
        return 1;
```

// dönüş değeri

```
    else
```

```
        return (x*fakto(x-1)); // özyineleme
```

```
}
```

```
main()
```

```
{
```

```
    int gir;
```

```
    cout<<"Faktoriyeli hesaplanacak sayiyi giriniz \n";
```

```
    cin>>gir;
```

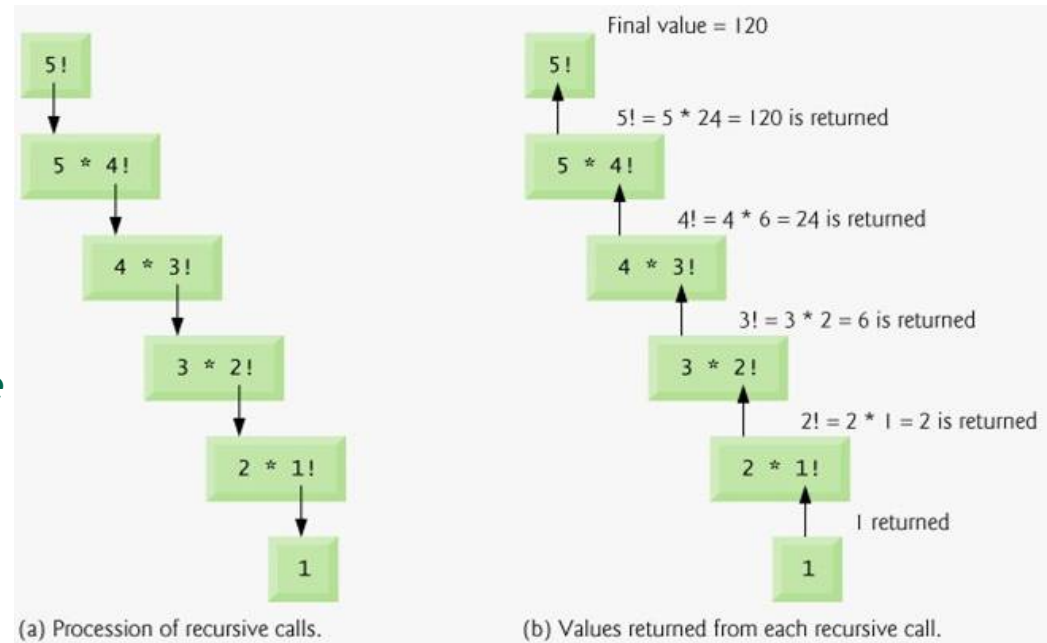
```
    cout<<endl;
```

```
    cout<<gir<<" Faktoriyeli "<<fakto(gir);
```

```
    system("pause");
```

```
}
```

Klavyeden girilen sayıya karşılık gelen Fibonacci sayısını bulan yöntemi özyinelemeli olarak geliştiriniz.
Fibonacci serisi: 0 1 1 2 3 5 8 13 21...



Sorular

Parametre olarak aldığı tam sayının 2 nin kuvveti olup olmadığını bulan ve geriye boolean olarak sonucu döndüren *bool tam(unsigned int)* fonksiyonunu yazınız.

Giriş parametresi olarak aldığı gerçel ve sanal değerlerini kullanarak karmaşık sayının kutupsal koordinatlarını bulan ve ekrana yazdıran fonksiyonu yazınız.

Kendisine gönderilen sayının asal olup olmadığını kontrol ederek geriye bool bir değer döndüren fonksiyonu tanımlayınız.

Aldığı ortalama not bilgisini harfe dönüştürerek geriye döndüren harfeDonustur fonksiyonunu tanımlayınız.

İkinci dereceden bir denklemin a, b ve c katsayılarını giriş parametresi olarak alan ve bu denklemin köklerini ekrana yazan fonksiyonu tanımlayınız.

Giriş parametresi olarak aldığı sayının mutlak değerini döndüren fonksiyonu yazınız.

Sorular

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