C
Fonksiyonlar

Dr. Öğr. Üyesi M. Ozan AKI

Rev 1.0

fonksiyonlar

Fonksiyonlar, C Dilinin “Yapısal” bir dil olmasını sağlayan temel yapılardır.

Karmaşık bir problem,

Böl ve Yönet (Divide & Conquer)

yaklaşımı ile çözümü daha basit küçük parçalara ayrılabilir.

Bu parçaların her biri, ayrı birer fonksiyon ile çözülebilir.

fonksiyon tanımlama

Tipik bir fonksiyon tanımı şu şekildedir;

[dönüş değeri] **fonksiyon_adı(**[tip değişken adı, ...]**)**

{

... // fonksiyon gövdesi

}

Değişken isimlerinde geçerli olan adlandırma kuralları fonksiyon isimleri içinde geçerlidir.

Fonksiyon adını takiben () parantezleri kullanılır

fonksiyon dönüş değeri – void

Eğer fonksiyon bir değer döndürmüyor ise **void** anahtar kelimesi kullanılır.

Aksi halde dönüş değeri için veri tipi belirtilmelidir.

Herhangi tip bildirimi yapılmamış ise, bazı derleyiciler bunu varsayılan olarak dönüş tipini **int** kabul eder.

```
void mesaj_yaz();
int topla(int a, int b);
```

fonksiyon dönüş değeri - return

Fonksiyon kodundan bir değer ile dönmek için **return** anahtar kelimesi kullanılır.

return, yanına yazılan sabit ya da değişken değeri geri döndür.

return kullanıldığı an fonksiyon bloğu sonlandırılır.

Bir dönüş değeri olmasa bile, return sadece fonksiyonu sonlandırmak için yanında değer yazılmadan da kullanılabilir.

fonksiyon parametreleri

Fonksiyona geçilen parametreler, fonksiyon adını takip eden parantezler içerisine yazılır.

Her bir parametre için ayrı tip tanımlayıcısı olmalıdır.

Hiçbir parametre geçilmeyecek ise, parantezler boş bırakılır. (ya da void anahtar kelimesi kullanılabilir.)

float ortalama(int a, int b, int c)

char buyuk_harf(char c);

fonksiyon deklarasyonu ve tanımı

Eğer bir fonksiyonun gövdesi { } blokları ile tanımlanmamış ise, bu bir fonksiyon deklarasyonudur.

Derleyici dönüş ve parametre tiplerini öğrenir.

Ancak fonksiyon gövdesi Bağlama (Linking) aşamasında hazır olmalıdır.

Bir fonksiyonun deklarasyonu ve tanımı, aynı kaynak dosyası içerisinde olabileceği gibi, farklı kaynak dosyalarında da olabilir.

Hatta bazı fonksiyonların hiç tanımı olmadığı halde, bağlama aşamasında kütüphaneden bağlanıyor olabilir.

fonksiyon deklarasyonu ve tanımı

```
int asalmi(int sayi); // → deklarasyon

...

int asal_mi(int sayi)
{
   int i;
   for(i=2; i<sayi; i++)
      if(sayi%i == 0) return 0;
   return 1;
}
```

fonksiyonun çağırılması (call & return)

Bir fonksiyon çağırıldığında, komut işletimi fonksiyon bloğuna atlamadan önce, STACK belleğe dönüş adresi ve geçilen tüm parametreler kaydedilir.

Fonksiyon kod bloğu içerisinde return komutu ile karşılaştığında ya da blok sona erdiğinde STACK bellekten dönüş değerini okur ve yerine dönüş değerini koyarak kaldığı noktaya geri döner.

fonksiyonun çağırılması (inline)

Bir fonksiyon inline anahtar kelimesi ile tanımlanmış ise, bu fonksiyonun adı geçen her noktaya kod bloğu ayrı ayrı yerleştirilir. (MAKRO gibi)

Call – Return söz konusu olmaz. Çalıştırılabilir kod büyür ancak STACK bellek kullanımı azaltılmış olur.

Gerektiği durumlarda (?) kullanılır.

Recursive (Özyinelemeli) Fonksiyonlar

Kendi kod bloğu içerisinde kendisini çağıran fonksiyonlardır.

Yinelenen fonksiyonlar, bazı algoritmların daha anlaşılır ve daha basit kod yazımını sağlar.

Yinelenen fonksiyon içerisinde mutlaka işlemi belirli bir noktada sonrandıran bir if olmalıdır.

Recursive (Özyinelemeli) Fonksiyonlar

Yinelenen bir fonksiyonun her kopyasında, yerel değişkenler ve parametreler yığın (stack) bellekte tutulur. Böylece her bir çağrıya ait değişkenler ayrı ayrı korunmuş olur.

Bu nedenle fazla sayıda yinelenen bir fonksiyon yığın (stack) belleği kolayca tüketebilir. Dolayısıyla, belirsiz iterasyonlar için kullanımı sakıncalı olabilir.

Recursive (Özyinelemeli) Faktöryel

Yinelenen fonksiyonlar için en basit örnek faktöryel hesabıdır.

```
int faktoryel(int n)
{
        int sonuc;
        if(n==1) return 1;
        sonuc = n * faktoryel(n-1);
        return sonuc;
}
```

Recursive (Özyinelemeli) Fibonacci

Fibonacci dizi elemanlarını döndüren yinelemeli fonksiyon:

```
int fibonacci(int n)
{
  return n<=1?n:fibonacci(n-1)+fibonacci(n-2);
}
```