KULLANICI TANIMLI FONKSİYONLAR(ALT PROGRAMLAR)

Önceki bölümlerde C dilinde tanımlanmış fonksiyonlardan bahsedilmişti. Burada ise bu temel fonksiyonlar dışında farklı bir fonksiyona ihtiyaç duyulması halinde nasıl tanımlanabileceğinden bahsedilecektir.

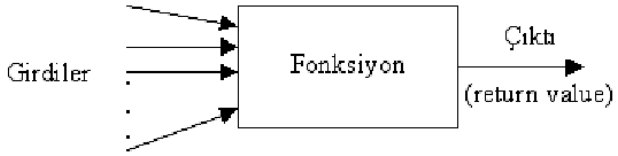
C programlama dili fonksiyon olarak adlandırılan alt programların birleştirilmesi kavramına dayanır. Bir C programı bir yada daha çok fonksiyonun bir araya gelmesi ile oluşur. Bu özellik bütün Yapısal Diller'in (C, Fortran, Pascal, ...) temelini oluşturur.

Bazı uygulamalarda aynı tür işlemler farklı durumlarda tekrar tekrar kullanılır. Örneğin bir denklem sistemi çözüm yönteminin farklı denklem sistemlerinin çözümü için kullanılması gibi. Bu durumda aynı işi gören birçok deyimden oluşan bir program parçasını programın çeşitli yerlerinde bir kereden çok tekrar tekrar kullanmak yerine, bu deyimlerden oluşan programları ayrı bir kısımda kullanmak çeşitli yararlar sağlar. Örneğin kodlama tekrarını önler ve programın anlaşılabilirliğini artırır. Diğer taraftan bir program içinde ard arda yapılan çeşitli işlemler vardır. Bunları da farklı gruplar (modüller) halinde kullanmak, programın yapılması ve anlaşılabilmesi açısından yararlı olur.

Bu tür işlemler alt programlar tarafından sağlanır. Bir alt program, bir ana program veya başka bir alt program tarafından "çağrılan" ve kendi içinde bir bütün oluşturan program parçasıdır. Bir alt programı kullanan program ise "çağıran" programdır.

Fonksiyon Kavramı

Fonksiyon, belirli sayıda verileri kullanarak bunları işleyen ve bir sonuç üreten komut grubudur. Her fonksiyonun bir adı ve fonksiyona gelen değerleri gösteren argümanları (bağımsız değişkenleri) vardır. Genel olarak bir fonksiyon Şekil 7.1.1'deki gibi bir kutu ile temsil edilir:

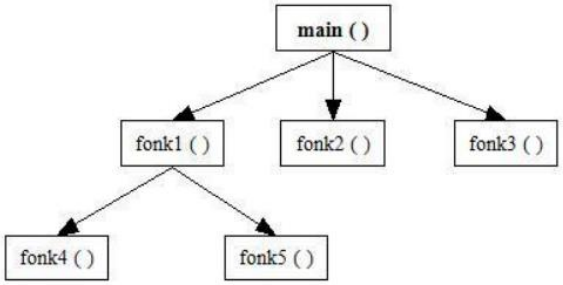


Şekil 0.1 Bir fonksiyonun kutu gösterimi

Fonksiyonların girdilerine parametreler yada argümanlar denir. Bir fonksiyon bu parametreleri alıp bir işleme tabi tutar ve bir değer hesaplar. Bu değer, çıktı veya geri dönüş değeri (return value) olarak adlandırılır. Bir fonksiyonun kaç girişi olursa olsun sadece bir çıkışı vardır.

Gerçek hayattaki problemlerin çözümü için geliştirilen programlar çok büyük boyutlardadır. Daha büyük programlar yazmanın en kolay yolu onları küçük parçalar halinde yazıp sonra birleştirmektir. Böylece çok büyük boyutlardaki program kodlarını yönetmek daha kolay olacaktır.

C programlarında main() ana fonksiyon olduğu için programın çalışması main() fonksiyonundan başlar. Main fonksiyonu diğer fonksiyonları çağırarak çalıştırır. Bir fonksiyon içerisinden başka bir veya birkaç fonksiyonda çağrılabilir(Şekil 7.1.2).



Şekil 0.2 Main() fonksiyonunun yapısı [4]

Fonksiyonların Bildirimi

C dilinde hazırlanan bir fonksiyonun genel yapısı şöyledir:

```
FonksiyonTipi FonksiyonAdı (argüman listesi)
argumanların tip bildirimleri
{
        Yerel değişkenlerin bildirimi
        ...
        fonksiyon içindeki deyimler veya diğer fonksiyonlar
        ...
        return geri dönüş değeri;
}
```

Bir fonksiyonun bildirimi iki şekilde yapılabilir:

- Ana programdan önce:
  Bu durumda fonksiyon yukarıda belirtildiği gibi yazılır ve ana programın önüne eklenir.

- Ana programdan sonra:
  Bu durumda fonksiyon yine yukarıda belirtildiği gibi yazılır ve ana programın arkasına eklenir. Ancak farklı olarak fonksiyon örneği (function prototype) ana programdan önce bildirilmelidir. Basit bir örnek ile açıklayalım:

  Ana programdan önce:

```
int topla (int x, int y)  /* fonksiyon */
{
```

```
    int sonuc;
    sonuc = x + y;
    return sonuc;
}
    ...
main()
{
    …
    topla(a,b);
    …
}
```

Ana programdan sonra:

```
...
int topla(int x, int y); /* fonksiyon örneği(prototipi) */
...
main()
{
    …
    topla(a,b);
    …
}
...
int topla (int x, int y)  /* fonksiyon */
{
    int sonuc;
    sonuc = x + y;
    return sonuc;
}
```

Bir C programı içinde, yazılmış fonksiyonlar genellikle bu iki tipte kullanılır. İkinci kullanımda fonksiyon prototipi mutlaka bildirilmelidir. Aksi halde bir hata mesajı ile karşılaşılır. Fonksiyon prototipinde arguman isimlerinin yazılması zorunlu değildir. Sadece argüman tiplerini belirtmek de yeterlidir.

Yukarıdaki topla fonksiyona ait prototip:

- `int topla (int x, int y);`

şeklinde yazılabileği gibi

- `int topla(int, int);`

şeklinde de yazılabilir. Fonksiyonların kullanımı ileride verilen örnekler ile daha iyi anlaşılacaktır.

Geri Dönüş Değerleri

Return anahtar sözcüğünün iki önemli işlevi vardır:

1. fonksiyonun geri dönüş değerini oluşturur

2. fonksiyonu sonlandırır

Bu deyiminden sonra bir değişken, işlem, sabit veya başka bir fonksiyon yazılabilir. Örneğin;

- return (a+b/c); /* parantez kullanmak zorunlu değil */
- return 10; /* değişken kullanmak mecbur değil */
- return topla(a,b)/2.0; /* önce topla fonksiyonu çalışır */

Bir fonksiyonda birden çok geri dönüş değeri kullanılabilir. Fakat, ilk karşılaşılan return deyiminden sonra fonksiyon sonlanır ve çağrılan yere bu değer gönderilir. Örneğin aşağıdaki harf fonksiyonunda beş tane return deyimi kullanılmıştır.

```
char harf(int not)
{
    if( not>=0 && not<50 ) return 'F';
    if( not>=50 && not<70 ) return 'D';
    if( not>=70 && not<80 ) return 'C';
    if( not>=80 && not<90 ) return 'B';
    if( not>=90  ) return 'A';
}
```

Bu fonksiyon kendisine parametre olarak gelen 0-100 arasındaki bir notun harf karşılığını gönderir. Aslında geri gönderilen değer bir tanedir. Eğer bu fonksiyon aşağıdaki gibi çağrılırsa:

```
char harfim;
...
harfim = harf(78);
...
```

harfim değişkenine 'C' değeri (karakteri) atanır.