Fonksiyonlar (Altprogram)

C Programlama Dili fonksiyon olarak adlandırılan alt programların birleştirilmesi kavramına dayanır. Bir C programı bir ya da daha çok fonksiyonun bir araya gelmesi ile oluşur. C Dilini öğrenmek için ilk önce fonksiyon oluşturmayı ve onların diğerleri ile birlikte kullanılmasını öğrenmek gerekir. Her fonksiyonun bir adı ve fonksiyona gelen değerleri gösteren argumanları (bağımsız değişkenleri) vardır. C dilinde hazırlanan bir fonksiyonun genel yapısı şöyledir:

```
FonksiyonTipi FonksiyonAdı(arguman listesi)
argumanların tip bildirimleri
{
  Yerel değişkenlerin bildirimi
  ...
  fonksiyon içindeki deyimler veya diğer fonksiyonlar
  ...
}
```

Örnek fonksiyon : iki tamsayıyı toplayıp sonucu `sonuc` adlı değişkene aktarır.

```
  /* Bu fonksiyon iki tamsayıyı toplar ve sonucu sonuc değişkenine aktarır
*/
  int tamsayi_topla( int x, int y )
  {
     int sonuc;
     sonuc = x + y;
     return sonuc;
  }
```

Yukarıdaki program parçası şu şekilde de yazılabilir:

```
  /* Bu fonksiyon iki tamsayıyı toplar */
  int tamsayi_topla( int x, int y )
  {
     return (x+y);
  }
```

Her iki program parçasında da `return` (geri dönüş) deyimi kullanılmaktadır. Bu deyim C programlama dilinin anahtar sözcüklerinden biridir ve fonksiyon içerisinde sonucu, kendisini çağıran yere göndemek için kullanılır. Yani `tamsay_topla()` fonksiyonu herhangi bir programın içerisinde kullanıldığında, fonksiyonun üreteceği sonuç `return` deyiminden sonra belirtilen değişken veya işlem olacaktır. Örneğin:

```
  ...
  int toplam;
  toplam = tamsayi_topla(1,2);
  printf("Toplam = %d dir",toplam);
  ...
```

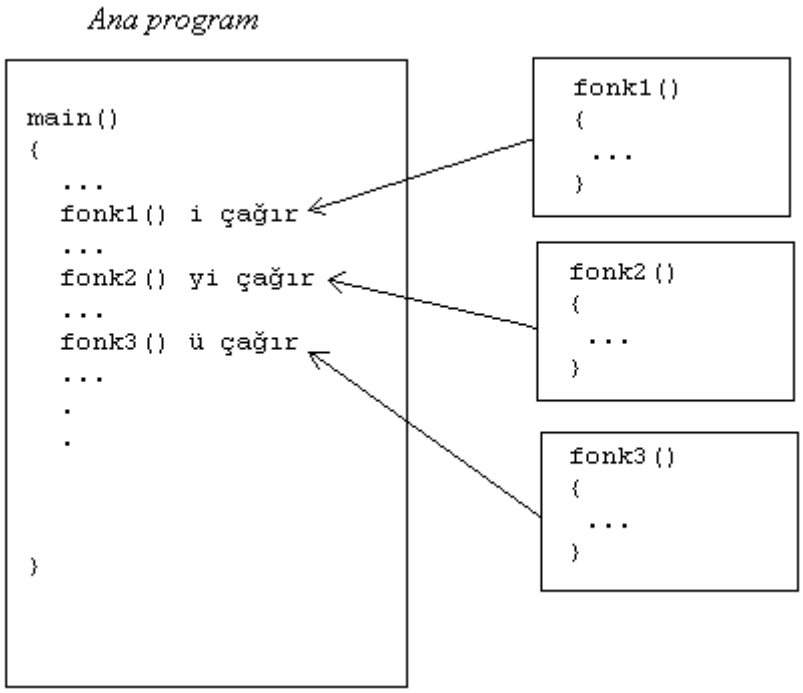
şeklinde kullanılırsa, `toplam` değişkenine 1+2=3 değeri atanır. Fonksiyon kullanımı Program.1 in üzerinde açıklanmıştır.

Program 1 : *tamsayi_topla(); fonksiyonunu ile basit bir fonksiyon uygulaması*

```
1:  /* iki sayıyı topla ve sonucu ekranda göster */
2:  #include <stdio.h>
3:
4:  int tamsayi_topla( int x,int y );  /* fonksiyon prototipi */
5:
6: int main()
7: {
8:    int toplam;
9: /*
10:    fonksiyon çağırılıp, 5 ve 12 değerleri parametre olarak aktarılıyor.
11:    tamsayi_topla(5,12) = 5 + 12 değeri toplam değişkenine atanması.
12: */
13:    toplam = tamsayi_topla(5,12);
14:
15:    printf("5 ve 12 nin toplamı  %d dir.\n", toplam);
16
17: return 0;
18: }
19:
20: /* fonksiyon tanımlanması */
21: /* Bu fonksiyon iki tamsayıyı toplar ve sonucu sonuc değişkenine
aktarır */
21:  int tamsayi_topla( int x, int y )
22:  {
23:     int sonuc;
24:     sonuc = x + y;
25:     return sonuc;
26:  }
```

Program 1 , basit bir fonksiyonun C programlama dilinde nasıl oluşturulduğunu ve kullanıldığını gösterir. 4. satırda kullanılacak olan fonksiyonun prototipi bildirilmektedir. Geleneksel olarak bütün fonksiyon kodları `main()` fonksiyonundan sonra bildirilir. (Gerçekte buna gerek yoktur. Fonksiyon bildirimleri `main()`den önce de tanımlanabilir.) Eğer geleneksel tercih yapılırsa, fonksiyonun adı `main()`den önce bildirilmelidir. Bu bildirime fonksiyon protipi(fonksiyon örneği) denir. 13. satırda toplam adlı tamsayı değişkenine `tamsayi_topla()` fonksiyonunun dönüş değeri, x + y değeri, atanır.

Ana programdan, `main()` fonksiyonundan, isteğe göre birden çok fonksiyon çağırmak mümkündür. Bir fonksiyonun çağırılması demek, o fonksiyonun geri dönüş değerinin ana programda kullanılması demektir. Birden çok fonksiyonun `main()` tarafından nasıl çağrıldığını temsil eden blok diyagram Şekil 1 de gösterilmiştir.



Şekil 1 : *Ana programdan fonksiyonların çağırılması. Fonksiyonu çağırmak için, fonksiyonun adını yazmak yeterlidir.*

Bir fonksiyonun her zaman geri dönüş değerinin olması gerekmez. Böyle fonksiyonların tipi `void` olarak belirtilmelidir. Bu tip fonksiyonlar başka bir yerde kullanılırken, herhangi bir değişkene atanması söz konusu değildir, çünkü geri dönüş değeri yoktur. Fakat `void` fonksiyonlara parametre aktarımı yapmak mümkündür. Örneğin Program 1 de kullanılan `tamsayi_topla()`: fonksiyonu `void` olarak şöyle bildirilir:

```
/* x,y,z global değişken olmalıdır !... */
void tamsayi_topla( int x,int y,int z )
{
  z = x + y;
}
```

Bu fonksiyon başka bir fonksiyonun içinde kullanılacaksa sadece isminin yazılması yeterli olacaktır. Mesela `main()` fonksiyonunda kullanmak istenirse:

```
main()
{
 ...
  tamsayi_topla(5,12,z);
  printf("5 ile 12 nin toplamı %d dir",z);
 ...
}
```

şeklinde olacaktır. Program 2 'de, ne parametresi ne de geri dönüş değerine sahip tipi `void` olan `mesaj_yaz()` fonksiyonu ile, ekrana basit bir mesajın, fonksiyon kullanarak, çıkarılması gösterilmiştir.

Program 2 : *mesaj_yaz() fonksiyonunu ile ekrana mesaj verme*

```
1:  /* void mesaj_yaz() ile ekrana mesaj yazar */
2:  #include <stdio.h>
3:
4:  void mesaj_yaz(void); /* fonksiyon prototipi */
5:
6:  main()
7:  {
8:     mesaj_yaz();  /* fonksiyonun çağırılması */
9:  }
10: /* fonksiyonun bildirimi */
11: void mesaj_yaz(void)
12: {
13:    puts("Merhaba...");
14: }
```

Program 2 'de 4. satırda `void mesaj_yaz()` fonksiyonunun prototipi bildirilmiştir. 8. satırda bu fonksiyonun sadece adı yazılmıştır. Derleyici bu satıra gelince hemen 11. satıra gidip `puts()` fonksiyonunu işleme koyar ve sonucu çağrıldığı yere, 8. satıra, gönderir.

Bir fonksiyon istenilen yerde istenildiği kadar kullanılabilir. Program 3 de `topla()` fonksiyonu, `ortalama()` fonksiyonunun içinden çağırılmıştır.

Program 3 : *İki sayının toplamı ve ortalamasını fonksiyonlar kullanarak hesaplanması*

```
1:   /* iki sayının toplamını ve ortalamasını hesaplar */
2:  #include <stdio.h>
3:
4:  int topla( int x,int y );
5:  float ortalama( int x, ,int y );
6:
7:  int x,y;  /* x ve y global olarak bildiriliyor */
8:
9:  main()
10: {
11:    puts("Toplamı ve ortalaması hesaplanacak iki sayı girin:");
12:    scanf("%d %d",&x,&y);
13:    printf( "\nToplamları : %d", topla(x,y) );
14:    printf( "\nOrtalaması : %f", ortalama(x,y) );
15: }
16:
17: /* iki sayının ortalamasını hesaplar */
18: float ortalama(x,y)
19: {
20:     float ort;
21:     ort = topla(x,y)/2.0;  /* topla(x,y) = x+y değeri çağırılıyor */
22:     return ort;
23: }
24:
25: /* iki sayının toplamını hesaplar */
26: int topla(x,y)
27: {
28:     return x+y;
29: }
```

Program 3 de, klavyeden girilen iki sayı 12. satırda okunmuştur. 13. ve 14. satırlarda fonksiyonlar sırasıyla çağırılmış ve sonuçları ekrana yazdırılmıştır. 21. satırda `topla` fonksiyounu, bir kez daha kullanılmıştır.

Başlık dosyalarında da bolca olan makro fonksiyon tanımlaması `#define` önişlemci komutu kullanılarak yapılır. Örneğin aşağıdaki makro fonksiyonlar geçerlidir.

```
#define kare(x) (x)*(x)
#define delta(a,b,c) (b)*(b)-4*(a)(c)
#define yaz() puts("Devam etmek için bir tuşa basın...")
#define PI 4*atan(1.0)
```

Bu şekilde tanımlanan fonksiyonların kullanımı, diğerleri gibidir. Yalnızca programın başında tanımlanır.

Matematiksel Fonksiyonlar

Matematiksel fonksiyonların tipleri `double` dır. Bu fonksiyonlardan biri program içinde kullanılacaksa `math.h` başlık dosyası program içine eklenmelidir. Bu kısımda bu fonksiyonlardan en sık kullanılanlar (`sin(),cos(),tan,pow(),sqrt()`) anlatılacaktır.

`sin() - cos() - tan()` Fonksiyonları

Bu fonksiyonlar en temel trigonometrik fonksiyonlardır. Kullanımları Tablo 1 de gösterilmiştir. Örnek program Program 4 de sunulmuştur.

Tablo 1 : *Bazı Trigonometrik fonksiyonlar*

Fonksiyon	Açıklama	Örnek
`double sin(double x)`	x sayısının sinüs değerini radyan cinsinden hesaplar	`y = sin(0.22)`
`double cos(double x)`	x sayısının kosinüs değerini radyan cinsinden hesaplar	`y = cos(0.14)`
`double tan(double x)`	x sayısının tanjant değerini radyan cinsinden hesaplar	`y = tan(0.82)`

Bu fonksiyonlar kendisine parametre olarak gelen değeri radyan olarak kabul eder ve sonucu hesaplar. Eğer derece cinsinden açıların hesaplanması gerekiyorsa su dönüşüm kullanılmalıdır:

```
radyan = (3.141593/180.0)*derece;
```

Program 4 : *45 dercelik bir açının sinüs, kosinüs ve tanjant değerleri*

```
1:  /* sin(), cos(), and tan() funksiyonlarının kullanımı */
2:  #include <stdio.h>
3:  #include <math.h>
4:
5:  main()
6:  {
7:     double x;
8:
9:     x = 45.0;                 /* 45 derece */
10:    x *= 3.141593 / 180.0;    /* radyana çeveir */
11:    printf("45 derecenin sinüsü   : %f.\n", sin(x));
12:    printf("45 derecenin kosinüsü : %f.\n", cos(x));
13:    printf("45 derecenin tanjantı : %f.\n", tan(x));
14: return 0;
15: }
```

`pow() - sqrt() - log() - log10()` Fonksiyonları

Bu fonksiyonların kullanımı Tablo 2 de, örnek program Program 5 de verilmiştir.

Tablo 2 : *pow() - sqrt() ve logaritmik fonksiyonlar*

Fonksiyon	Açıklama	Örnek
`double pow(double x, double y)`	x^y değerini hesaplar	`pow(2.0,3.0) =` 2^3 `= 8`
`double sqrt(double x)`	pozitif x sayısının karekökünü hesaplar	`sqrt(4.0) =` $4^{1/2}$ `= 2`
`double log(double x)`	pozitif x sayısının doğal logaritmasını hesaplar, ln(x)	`log(4.0) = 1.386294`
`double log10(double x)`	pozitif x sayısının logaritmasını hesaplar	`log(4.0) = 0.602060`

Program 5 : *pow(), sqrt(), log() ve log10() fonksiyonlarının kullanımı*

```
1:  /* pow(), sqrt(), log() ve log10() fonksiyonlarının kullanımı */
2:  #include <stdio.h>
3:  #include <math.h>
4:
5:  main()
6:  {
7:     double x, y, z;
8:
9:     x = 64.0;
10:    y = 3.0;
11:    z = 0.5;
12:    printf("pow(64.0, 3.0) = %7.0f\n", pow(x,y) );
13:    printf("sqrt(64.0)     = %2.0f\n", sqrt(x)  );
14:    printf("pow(64.0, 0.5) = %2.0f\n", pow(x,z) );
15:    printf("ln(3.0)        = %f   \n", log(y)   );
16:    printf("log(3.0)       = %f   \n", log10(y) );
17:
18: return 0;
19: }
```

Bazı String Fonksiyonları

Bu fonksiyonlar standart C dilinde iki stringi karşılaştırmak, bir stringin içeriğini diğerine kopyalamak ve stringin uzunluğunu bulmak vb işlemler için tanımlı fonksiyonlardır. Bu ve benzeri fonksiyonlar kullanılırken `string.h` kütüphanesi programın başına ilave edilmelidir. Burada sadece bunlardan bir kaçı Tablo 3 de verilmiştir.

`str`, `str1` ve `str2` birer string ve `kr` bir karakter olmak üzere:

Tablo 3 : *string.h kütüphanesine ait, bazı string fonksiyonları*

`strcmp( str1,str2 );`	str1 ve str2 yi karşlaştırır
`strcpy( str1,str2 );`	str2 yi str1 e kopyalar
`strcat( str1,str2 );`	str2 yi str1 e ekler
`strrev( str );`	str yi ters çevirir
`strlen( str );`	str nin kaç karakterden oluştuğunu hesaplar
`strchr( str,kr );`	kr ile verilen karakterin str içindeki soldan itibaren yerini verir

Program 6 : *İki stringin karşılaştırılması*

```
1:  #include <string.h>
2:  main(){
3:     char ktr1[10],ktr2[10];
4:     int sonuc;
5:     printf("1. katar:");gets(ktr1);
6:     printf("2. katar:");gets(ktr2);
7:     sonuc = strcmp(ktr1,ktr2);
8:     if(sonuc>0)        puts("2. 1.den büyük");
9:     else if(sonuc<0)   puts("2. 1.den küçük");
10:    else               puts("2. 1. eşit");
11: }
```

Program 7 : *basit bir şifre programı*

```
1:  #include <string.h>
2:  main(){
3:     char sifre[20];
4:     printf("SIFRE : ");
5:     scanf("%s",sifre);
6:    if( strcmp(sifre,"deneme")==0 )
7:        puts("sifre dogru girildi");
8:     else
9:        puts("sifre yanlis!");
10: }
```

Program 8 : *Bir stringi diğerine kopyalama*

```
1:  #include <string.h>
2:  main(){
2:     char *str1="ahmet", str2[10];
4:  /* once */
5:     puts(str2);
6:     strcpy(str2,str1);
7: /* sonra */
8:     puts(str2);
9: }
```

Program 9 : *bir stringi diğerine ekleme*

```
1:  #include <string.h>
2:  main(){
3:    char *a="ahmet           ";
4:    char *b="bingul";
5:    strcat(a,b);
6:    printf(a);
7:  }
```

Program 10 : *strinin uzunluğunu hesaplar*

```
1:  #include <string.h>
2:  main(){
3:     char ktr[100];
4:     puts("Birseyler yazin:");
5:     gets(ktr);
6:     printf("%s %d karakterden oluşmuştur.",ktr,strlen(ktr));
7:  }
```