**Fonksiyon yapısı**

Fonksiyon, eğer varsa, kendisine geçirilen parametreleri de kullanarak kod bloğu içindeki işlem satırları ile bir takım işlemler yaparak bir sonuç elde eder. Elde edilen sonuç, ihtiyaca bağlı olarak, fonksiyon içinde ya da return komutu ile geri döndürüldükten sonra program içinde kullanılır.

**veri-türü** fonksiyon-adı (**parametreler**)

{

işlem satırı

.

.

işlem satırı

**return** **ifade**;

}

Fonksiyon yapısında 5 temel eleman vardır. Bu elemanlardan fonksiyon-adı ve işlem satırı mutlaka bulunmalıdır. Ancak, veri-türü, parametreler ve return ifadelerinin tanımlanması programcının ihtiyaçları doğrultusunda isteğe bağlıdır. Parametre yerine argüman ifadesi de kullanılmaktadır.

**veri-türü** : Fonksiyonun geri döndürdüğü veri türünü gösterir.

fonksiyon-adı : Fonksiyon adını gösterir.

**parametreler** : Fonksiyona geçirilen verileri gösterir. Sayısı birden fazla olabilir.

İşlem satırı : Fonksiyon içindeki işlem satırlarını gösterir.

**return** : Verileri geri döndürmeye yarar. Son satırda kullanılması şart değildir.

**ifade** : Değişken, sabit ve işlemciler kullanılarak oluşturulan veridir. Elde edilen veri türü fonksiyonun geri döndürdüğü veri türü ile aynı olmalıdır.

Fonksiyon, kendisine geçirilen parametrelerin değerlerini, işlem durumuna bağlı oarak, değiştirerek veya değiştirmeden kullanabilir.

Bir fonksiyonun geri verdiği veri türü **dizi** dışında herhangi bir veri olabilir. Fonksiyon dizi tipinde bir veri geri döndüremez, ancak farklı yöntemlerle bu işlemi gerçekleştiribilir.

Bir fonksiyon için geri dönen veri türü ve parametre tanımlanmadığında, bu ifadelerin yerine **void** ifadesini kullanarak bu durumu derleyiciye bildirebiliriz. main() fonksiyonundan önce de void ifadesini kullanılabilir.

**void** fonksiyon-adı (**void**)

{

işlem satırları

}

**veri-türü** ifadesi Eğer bir fonksiyonun adının başında veri türü tanımlanmazsa, fonksiyon int bir değer geri verir. Eğer bir fonksiyonun int değer dışında bir veri türü geri vermesi isteniyorsa, fonksiyon adının başında mutlaka bir veri türü tanımlanması gerekir.

Aşağıdaki bir ve ikinci fonksiyonlar int bir değer, üçüncü fonksiyon ise float bir değer geri verir:

fonk(int d1, int d2)

{

.

.

.

return id+id2;

}

int fonk(int d1, int d2)

{

.

.

.

return id+id2;

}

float fonk(float fd1, float fd2)

{

.

.

.

return fd1+fd2;

}

**Fonksiyon bildirimi**

Normal olarak, fonksiyonların kullanılmadan önce fonksiyon prototipi yöntemiyle bildirimi yapılmalıdır. Ffonksiyonların ana yapısı main() fonksiyonundan önce yer alırsa, fonksiyon bildirimi yapılmayabilir. Ancak, main() fonksiyonundan sonra yer alıyorsa, mutlaka main() fonksiyonunda önce fonksiyon bildirimi yapılmalıdır.

// Fonksiyon bildiriminin yapılması gereken durum

fonk(int d1, int d2); // Fonksiyon bildirimi

int main(void)

{

.

.

.

return 0;

}

fonk(int d1, int d2) // Fonksiyon ana yapı bildirimi

{

.

.

.

return id+id2;

}

// Fonksiyon bildiriminin yapılması şart olmayan durum

fonk(int d1, int d2) // Fonksiyon ana yapı bildirimi

{

.

.

.

return id+id2;

}

int main(void)

{

.

.

.

return 0;

}

Bir veri türü geri veren bir fonksiyon kullanılacağı zaman, bu fonksiyonu kullanmadan önce, fonksiyonun geri vereceği veri türü ile fonksiyona geçirilecek argümanların sayısını ve türlerini programın başında belirlenmesi gerekir. Bu işleme **Fonksiyon bildirimi** adı verilir.

Eğer fonksiyonun yapısı main() fonksiyonundan önce tanımlanırsa, ayrıca prototip yöntemiyle bir bildirim yapılmasına ihtiyaç duyulmaz.

Fonksiyon bildiriminin genel yapısı aşağıda gösterilmektedir:

**veri-türü** **fonksiyon-adı** (veri-türü **parametre1**, veri-türü **parametre2**, ...);

Fonksiyon bildirimi yapıldığında, eğer fonksiyonu çağırırken kullandığınız argümanların sayısı ve argüman veri türü, bildirim yaparken tanımladığınızdan farklı olursa, derleyici bir hata mesajı verir. Program, fonksiyonların her çağrılmasında tanımlanan argüman sayısının ve argüman veri türlerinin fonksiyon bildirimine uygun olup olmadığını kontrol eder. Fonksiyon çağrısı ile fonksiyon bildirimi arasında uyuşmazlık olduğunda, derleyici programı derlemez. Örneğin, 3 parametreli olarak bildirimi yapılmış bir fonksiyonu 4 parametre ile çağırırsanız derleyici bir hata mesajı vererek çalışmasını sona erdirir.

Fonksiyon bildirimini bir örnek üzerinde incelemeye çalışalım:

Örnek

#include <stdio.h>

int kare\_al(int id); // 1

int main(void)

{

int id;

id = 21;

return 0 printf("%d sayısının karesi: %d", id, kare\_al(id));

;

}

int kare\_al(int id)

{

return id\*id;

}

Yukarıdaki programı derleyip çalıştırdığımızda, aşağıdaki ifadeleri ekrana yazar:

21 sayısının karesi: 441

Program, id adlı int bir değişken tanımladıktan sonra değişkene 21 değerini atar. id değişkenini parametre olarak geçirerek kare\_al() fonksiyonunu çağırır. Fonksiyon kendisine paramter olarak geçirilen değişkenin karesini alarak elde ettiği değeri geri döndürür. Program, fonksiyon tarafından geri döndürülen değeri ekrana yazar. Başlangıçta 1 sayısı ile gösterilen işlem satırında fonksiyon bildirimi yapar. Yapılan fonksiyon bildirimi, kare\_al() fonksiyonunun geri verdiği veri türünün int olacağını ve bir adet int değişkenin parametre olarak tanımlandığını derleyiciye bildirir.

Parametre kullanmayan bir fonksiyon için yapılan bildirimin parametre bölümünde **void** ifadesi kullanılır:

**void** **fonksiyon-adı**(**void**);

**Fonksiyonlar ve return ifadesi kullanımı**

return ifadesi bir fonksiyondan çıkış yapmak ve program içinde fonksiyon çağrısının yapıldığı işlem satırından bir sonraki işlem satırına geçiş yapmak için kullanılır.

Eğer bir fonksiyon void olmayan bir değer geri döndürecek şekilde tanımlanmış ise, return ifadesi fonksiyonun geri döndürdüğü değerle birlikte dönüş yapar. Eğer fonksiyon void bir değer geri döndürecek şekilde tanımlanmış ise, return değeri de herhangi bir değer geri döndürmez.

return ifadesi fonksiyonun geri döndürmesi gereken değerle dönüş yapar (void veya void olmayan bir veri türü).

return ifadesinin genel kullanım şekli aşağıdaki şekildedir:

**veri-türü** fonksiyon-adı (**parametreler**)

{

işlem satırı

.

.

işlem satırı

**return** **ifade**;

}

Return satırında yer alan ifade kavramı sadece fonksiyon bir değer döndürecek şekilde tanımlanmış ise eklenir. Bu durumda, ifade ile elde edilen değer fonksiyonun geri döndüreceği değer olacaktır.

Bir fonksiyon içinde istediğimiz kadar dönüş ifadesi kullanabiliriz. Fonksiyon içindeki return ifadeleri doğrudan veya bir koşula bağlı olarak eklenebilir. Ancak, program ilk return ifadesi ile karşılaştığında, fonksiyon sona erer ve programda fonksiyon çağrısının yapıldığı işlem satırından sonraki satırdan çalışmaya devam eder.

Fonksiyonun sonunda bulunan **}** değerine gelindiğinde de fonksiyon tanımlanmamış bir değer döndürür. Eğer bu durum, void olmayan bir değer döndürecek şekilde tanımlanan bir fonksiyonda meydana gelirse, fonksiyonun geri döndürdüğü değer tanımsız olur.

Fonksiyon void bir değer döndürecek şekilde tanımlandığında, bir değer içeren return ifadesi kullanılamaz.

Şimdi, int bir değer döndüren ve return ifadesinin sadece en son satırda kullanıldığı bir fonksiyonun çalışmasını bir örnek üzerinde inceleyelim:

Örnek

#include <stdio.h>

int deger\_yaz\_topla(int limit);

int main(void)

{

printf("Sayıların toplamı: %d", deger\_yaz\_topla(10));

return 0;

}

int deger\_yaz\_topla(int limit)

{

int id1, id2;

for (id1=1, id2=0; id1<=limit; id1++) {

printf("%d ", id1);

id2 += id1;

}

printf("\n");

return id2;

}

Yukarıdaki programı derleyip çalıştırdığımızda, aşağıdaki ifadeleri ekrana yazar:

1 2 3 4 5 6 7 8 9 10

Sayıların toplamı: 55

Program, deger\_yaz\_topla() fonksiyonu yoluyla, 1'den başlayarak fonksiyona parametre olarak geçirdiği 10 değerine kadar olan sayıları ekrana yazar ve sayıları toplayarak bir değişkene aktarır. Fonksiyon sayıların toplamını içeren değişken değerini fonksiyonun son satırında bulunan return ifadesi ile geri döndürür. Geri döndürülen değer main() fonksiyonu içine ekrana yazılır.

**Fonksiyon parametrelerine argüman geçirme**

Bir fonksiyona argüman geçirmenin 2 farklı yöntemi vardır:

1. Değer yoluyla

2. Referans yoluyla

**Değer yoluyla argüman geçirme**

Değer yoluyla argüman geçirme yöntemi kullanıldığında, argüman olarak kullanılan **değişkenler** çağrılan fonksiyonun parametrelerine geçirilir. Bu durumda, çağrılan fonksiyon içindeki işlem satırları tarafından değerleri değiştirilen parametre değişkenleri, fonksiyon çağırırken argüman olarak kullanılan değişken değerlerini etkilemez.

Bu özelliği bir örnek üzerinde incelemeye çalışalım:

Örnek

#include <stdio.h>

void fonk(int id);

int main(void)

{

int id = 21;

fonk(id);

printf("main() fonksiyonu içinde id değişken değeri: %d", id);

return 0;

}

void fonk(int id)

{

id = id + 35;

printf("Fonksiyon içinde id değişken değeri: %d\n", id);

}

Yukarıdaki programı derleyip çalıştırdığımızda, aşağıdaki ifadeleri ekrana yazar:

Fonksiyon içinde id değişken değeri: 56

main() fonksiyonu içinde id değişken değeri: 21

Program, int bir değişken parametresi olan fonk() adlı bir fonksiyon bildirimi yapar. 21 değerini atadığı id değişkenini argüman olarak kullanıp, fonk() fonksiyonunu çağırır. fonk() fonksiyonu kendisine parametre olarak geçirilen id değişkenine, 35 sayısını ekleyerek, elde ettiği değeri ekrana yazar. Fonksiyondan çıktıktan sonra, program id değişken değerini ekrana yazar. Bu değer, fonksiyon çağrıldığında id değişkeninin sahip olduğu değer ile aynı olduğundan, fonksiyon içinde id değişkenine yapılan değişiklik, id değişkeninin program içindeki değerini etkilememiş olur.

**Referans yoluyla argüman geçirme**

Referans yoluyla argüman geçirme metodunu kullanıldığında, argüman olarak kullanılan değerlerin bellek adresleri çağrılan fonksiyonun parametrelerine geçirilir. Bu durumda, çağrılan fonksiyon içinde değeri değiştirilen parametreler, fonksiyonu çağırırken argüman olarak kullanılan değişken değerlerini değiştirir. Çünkü, fonksiyon parametrelerine geçirilen değerler argümanların bellek adresleridir.

Bu özelliği bir önceki örneği değiştirerek incelemeye çalışalım:

Örnek

#include <stdio.h>

void fonk(int \*ip);

int main(void)

{

int id = 21;

fonk(&id);

printf("main() fonksiyonu içinde id değişken değeri: %d", id);

return 0;

}

void fonk(int \*ip)

{

\*ip = (\*ip) + 35;

printf("Fonksiyon içinde id değişken değeri: %d\n", \*ip);

}

Yukarıdaki programı derleyip çalıştırdığımızda, aşağıdaki ifadeleri ekrana yazar:

Fonksiyon içinde id değişken değeri:56

main() fonksiyonu içinde id değişken değeri:56

Programda, fonk() fonksiyonuna argüman olarak geçirilen id değişken değerinin değişmesinin tek nedeni, değişken adresinin fonksiyona geçirilmiş olmasıdır. Bu durumda, fonk() fonksiyonu içinde değeri değiştirilen id değişken değeri, fonksiyondan çıkıldıktan sonra da, program içinde aynı değeri korumaktadır.

Kısaca ifade etmek gerekirse, bir fonksiyona argüman olarak geçirilen değerlere, fonksiyon içinde yapılan değişikliklerin, fonksiyonun çalışması sona erdikten sonra, program içinde de geçerli olması istendiğinde **Referans Yoluyla Argüman Geçirme yöntemi**, aksi takdirde **Değer Yoluyla Argüman Geçirme yöntemi** kullanılmalıdır.