Programlama Dillerinin Prensipleri

HAFTA 6

YAPISAL PROGRAMLAMA

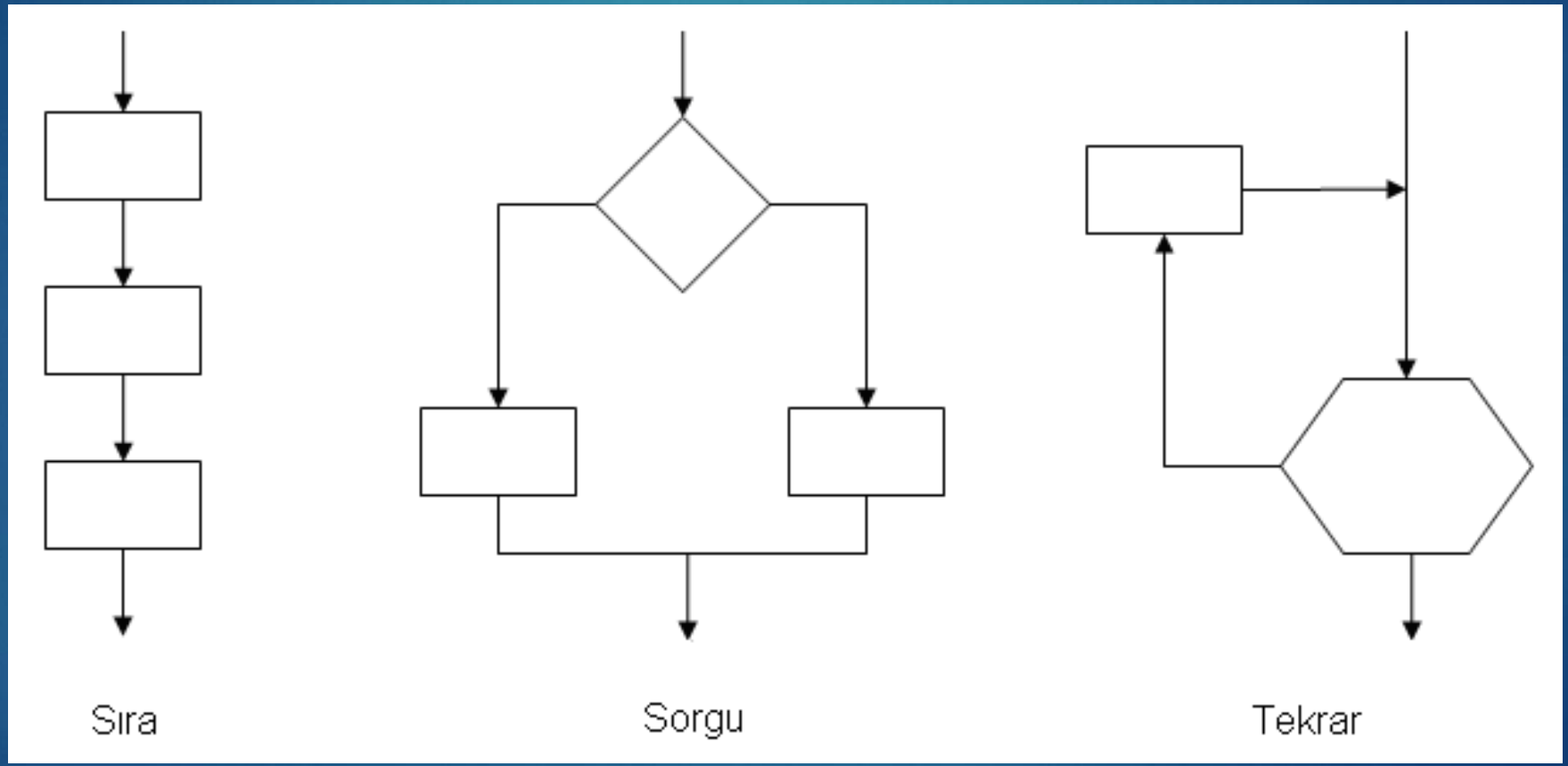
ALT PROGRAMLAR VE MODÜLASYON

DR. ÖĞR. ÜYESİ DENİZ BALTA

Yapısal Programlama

- Yapısal programlama, program tasarımı ve yazılmasını kurallara bağlayan ve disiplin altına alan bir yaklaşımdır.
- Yapısal programlama tekniğinde bir programın kolay yazılması, okunabilir olması ve hatalardan daha kısa sürede ayıklanması amaçlanmaktadır.
- Özellikle 70'li yıllar yapısal programlamanın ilke olarak yerleşmekte olduğu yıllardır.

Yapısal Programlama



Sıralı Yapılar

- Sıralı yapılar bir programdaki bir yada birden fazla deyimin göründükleri sırada yapılmasını sağlar.
- Sıralı işlemlerde en temel işlem atama işlemidir.
- En basit akış şeması ifadeleri, bir dizi işlemin birbiri ardından sırasıyla yapılmasını şeklinde olan akış ifadeleridir.
- Bu tip akışlar oldukça yalın ve basittir.
- Bu tarz akışlar genelde bir problemin bir parçasını çözümlemek ve ifade etmek için kullanılır.
- Sorgu ve tekrar gerektirmeyen bazı basit ardışık problemler de bu tip akış kullanabilir.

Seçimlik Yapılar

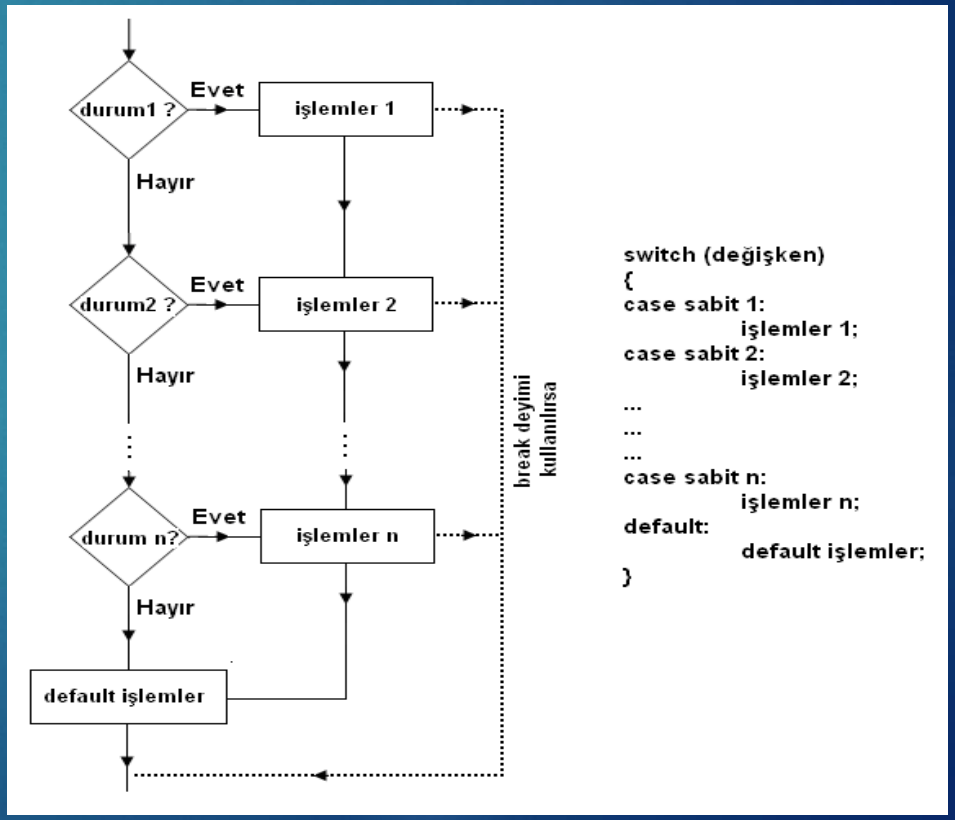
- Algoritma içerisinde verilen işlem adımları genel de sıralı adımlardan oluşur.
- Fakat bazı koşullarda bu işlem sıralarının değiştirilmesi ve diğer bir işlem sırasının seçilmesi gerekebilir.
- Sorgu (seçme) işlemi akış diyagramında baklava dilimi şeklindeki karşılaştırma simgesi ile ifade edilir.
- Simgenin içerisine koşul yazılır. Koşulun sonucuna göre iki yönden birisi seçilir.
- Program akışı, koşul olumlu ise “evet” olumsuz ise “hayır” olarak etiketlenen yöne dallanma yapar.

Seçimlik Yapılar – İç içe (Nested) seçimlik yapılar

- İki-yollu seçim deyimleri içiçe yuvalandığında, else deyiminin hangi if deyimine ait olduğunun belirlenmesi güçleşir ve sallanan-else (dangling else) problemi oluşabilir.
- Bu problem dillerde, ikinci if-then yapısı bir deyimin gurubunun tek bir deyim gibi algılanmasını sağlayan birleşik deyim yapılması yoluyla çözülmüştür.
- Birleşik deyimler, ALGOL60, Pascal'da begin ... end yapısı, C'de ise { … } yapısıyla sunulmaktadır.

Seçimlik Yapılar- Çoklu seçim yapıları

- "switch" – "case" seçme yapısı bir değişkenin içeriğine bakarak programın akışını birçok seçenekten birisine yönlendiren bir karşılaştırma deyimidir.
- Değişkenin içeriği hangi sabit ile uyuşursa ona ait işlem kümesi ve arkasındaki bütün işlem kümeleri yürütülür.
- Ancak küme deyimleri arasında break kullanılırsa, daha sonraki tüm işlem kümeleri atlanarak "switch" bloğunun sonuna gidilir.



Seçimlik Yapılar- Kısa devre değerlendirme

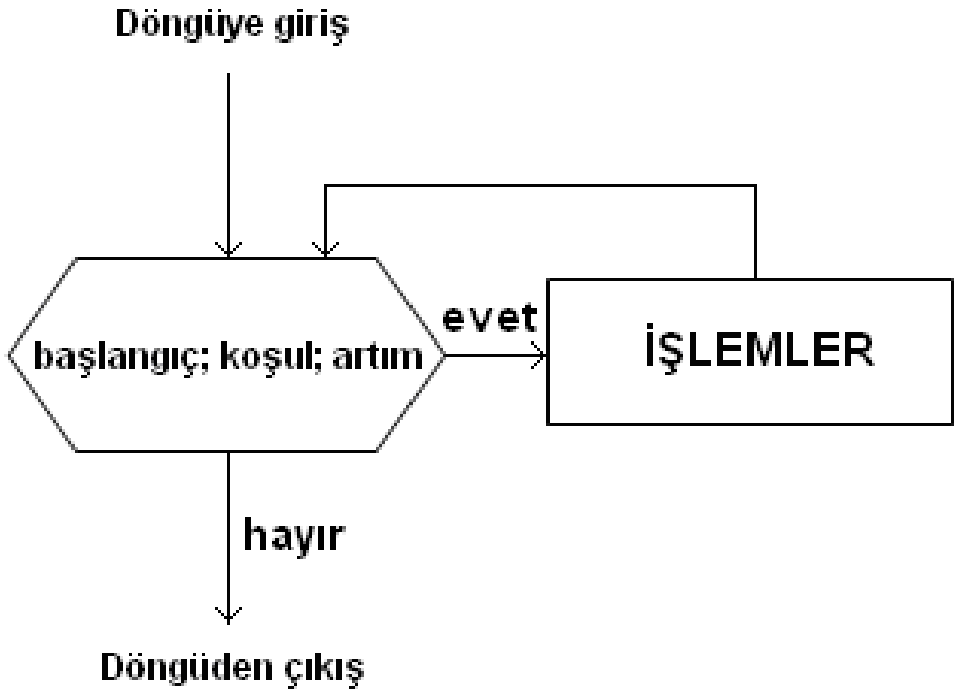
- Bir ifadenin sonucunun elde edilmesinde ifadede bulunan bütün operand yada operatörlerin değerlendirilmesine gerek yok ise burada kısa devre değerlendirme söz konusudur.
- Kısa devre değerlendirmede, ifadenin sonucunun elde edilmesi için tek bir bileşenin sonucu yeterli olabilir.
- Eğer sonucun belirlenmesinde bütün operand yada operatörlerin değerlendirilmesi gerekiyorsa buna **tam değerlendirme** denir. Tam değerlendirmede, ifadedeki her bileşen ayrı ayrı değerlendirilir.
- Bir programlama dilinin tam değerlendirme veya kısa devre değerlendirmeyi destekleyip desteklemediği dilin tasarımı sırasında bağlanır.

Tekrar Yapıları

- Uygulamalarda sıklıkla döngü kurulması gerekmektedir.
- Bu döngü kurma işlemi, ya döngü deyimleriyle yada yapısal programlamada ilke olarak kullanılması istenilmeyen goto deyimiyle gerçekleştirilir.
- Döngü deyimleri uygulamalarda yazılan kod uzunluğunu azaltır.
- Tekrar yapıları temel olarak Sayaç kontrollü ve mantıksal kontrollü döngüler olmak üzere iki sınıfa ayırabiliriz.

Sayaç kontrollü döngü yapıları

FOR Döngüsü:



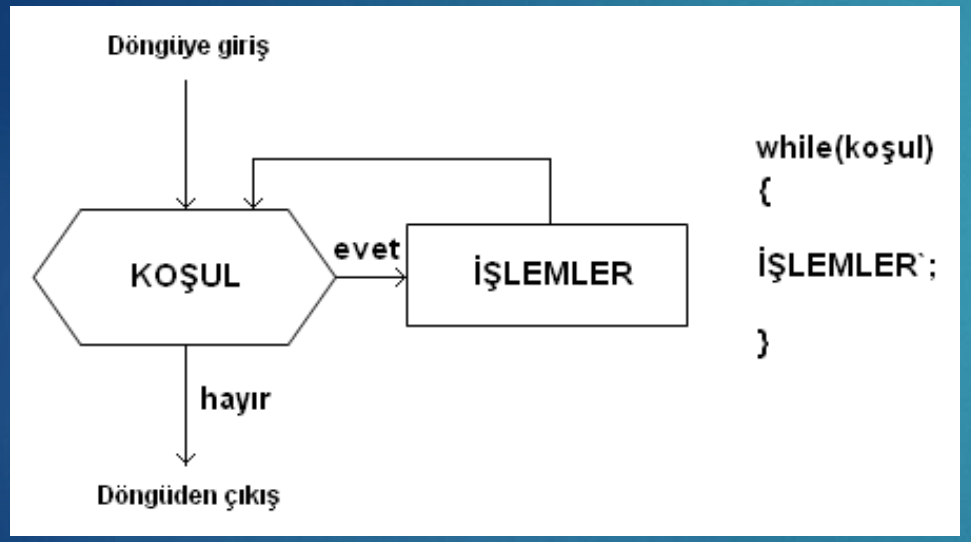

```
for(sayaç başlangıcı; koşul; artım)
{

İŞLEMLER`;

}
```

Mantıksal Kontrollü Döngü yapıları

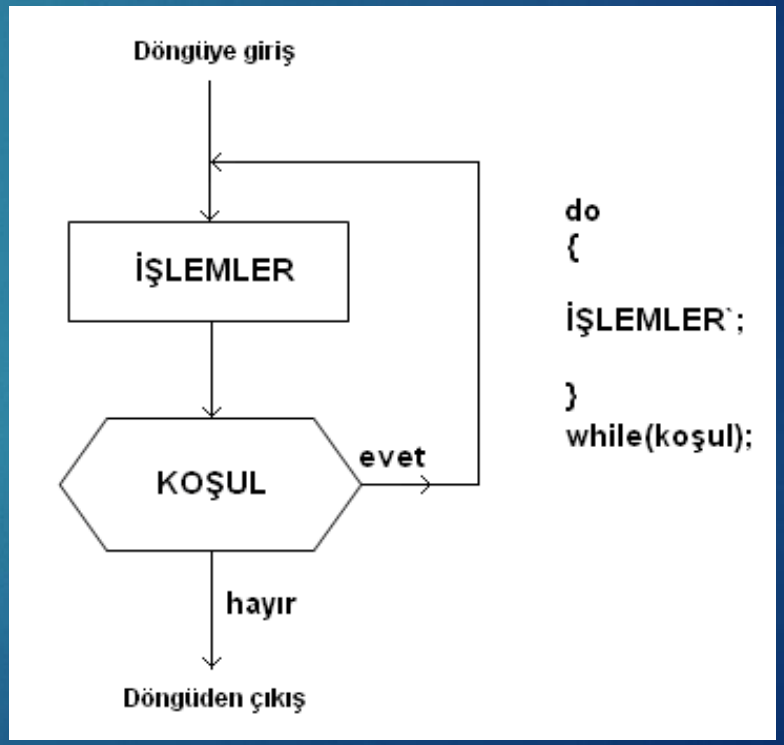
- Mantıksal kontrolü Tekrar Yapıları kendi içerisinde koşulu başta sınayan (pre-test), diğeri koşulu sonda sınayan (post-test) olmak üzere iki sınıfa ayrılır.
- Koşulu başa sınaması, daha çevrime girmeden döngü koşuluna bakılması ve koşul olumlu ise çevrime girilmesi, koşul olumsuz ise çevrime girilmeden sonraki adımlara geçilmesi anlamına gelir.
- Koşulun sonda sınanması ise, çevrim içerisinde kodun en az bir kere işletilmesi ve eğer koşul sağlanıyorsa çevrime devam edilmesi sağlanmıyor ise çevrimden çıkılması anlamına gelir.

Mantıksal Kontrollü Döngü yapıları

- WHILE döngüsü



DO-WHILE döngüsü



Döngü kontrol mekanizmaları

- Bir program içinde deyimlerin akışı kullanıcı tarafından yönlendirilebilir.
- Dil döngünün bir tekrarının normalden önce tamamlanması için deyimler kullanılmasına izin verebilir.
- Bir döngünün tek başlangıç ve çıkış noktası olmalıdır. Oysa zaman zaman bir döngüden normalden önce çıkmak yada başka bir döngüye girmek gerekebilir.
- Java, C ve C++ dillerinde bir döngüden normalden önce çıkış için break deyimi, Modula-2, QuickBASIC ve Fortran 90 dillerinde ise ADA'da olduğu gibi exit deyimi kullanılır.
- C' de döngülerde akışı değiştirmek için ayrıca denetimi en içteki döngünün sınama deyimine aktaran continue deyimi kullanılabilmektedir.

Alt Programlar ve Modülasyon

- Çoğu problemin çözümünde genellikle uzun programlara ihtiyaç duyulmaktadır.
- Binlerce ifadeden oluşan bir programların yazılması ve anlaşılması oldukça zor bir işlemdir.
- Bu sebeple problemin daha kolay çözülebilen alt parçalarına (modül) ayrıştırılması işlemi yapısal programlamanın temel yaklaşımlarındandır.

Alt program modül tanımı

- Küçük program parçacıkları olarak adlandırılabilecek bu yapı alt program, modül, fonksiyon veya prosedüre şeklinde kendini gösterebilir.
- Altprogram kullanmanın faydaları aşağıdaki gibi sıralanabilir:
  - Belirli bir işi yapan program parçasının, birden çok yerde değişik veriler için ayrı ayrı yazılmasını önlenerek program kodlarının gereksiz yere uzaması önlenmiş olur.
  - Düşük düzeydeki ayrıntıları, program mantığının ana akışından uzaklaştırıp ayrı modüllere yerleştirerek, programın tasarlanmasını kolaylaştırır ve okunabilirliğini artırır.
  - Büyük bir programın, daha küçük ve yazılması daha kolay fonksiyonel parçalara bölünür.
  - Program yazma işinin birden çok programcı tarafından paylaşılmasına olanak sağlar ve altprogramlar birden çok uygulama arasında paylaşılabilir.

Fonksiyonlar

- Alt program olarak isimlendirilebilecek fonksiyon kendi kendine işlem görebilecek modülerliğe sahip bir işi gerçekleştirmek için yazılmış kod bloğudur.
- Altprogram (fonksiyon) çağrıldığı zaman formal parametreleri ve lokal değişkenleri ile birlikte çağrılır.
- Her altprogramın tek bir başlangıç noktası vardır.
- Bellek yeri ataması da çağrıldığı zaman gerçekleşir.
- Denetim çağrıldığı noktaya geçtiğinde ise bu bellek yeri boşaltılır.

Fonksiyonlar

- Yığıt bellek, yerel değişkenler ve altprogramın çalışması süresince kullanılan parametreler gibi bir altprogram çağrımıyla ilişkili bilgiyi saklamak için kullanılır.
- Yığıt, LIFO mantığında çalışan bir veri yapısı olduğu için ve aynı anda tek bir altprogram etkin olabildiği için, altprogramların çalışma tekniğine uygundur.
- Etkin olan her altprogram için yığıt bellekte bir etkinlik kaydı (activation record) oluşturulur.
- Altprogramlarda kullanılan yerel değişkenler için her yordama ilişkin etkinlik kaydı kapsamında, yığıt bellekte bellek atanır.

Fonksiyonlar

Program kodu Altprogram çağrım deyimi Program kodu	
Altprogram Başlığı	Altprogram ismini, parametreler listesini (formal parametreler) ve varsa altprogramın döndürdüğü değer tipini bildirir.
Lokal değişkenler	Altprogramda geçerli değişkenler
Deyimler	Altprogramda tanımlı deyimler
Altprogram sonu Program kodu	Altprogramın çalışması bittikten sonra kontrol çağrıldığı noktaya geçer.
call prog1 (a,b,c)	call:çağrım deyimi prog1:altprogram ismi a,b,c: gerçek parametre

Fonksiyonlarda çağrı kısmı

- Altprogramlar gerekli parametrelerle birlikte çağrılınca etkinleşir ve çalışması tamamlanıncaya kadar, diğer etkin olan programlar durdurulur.
- Eğer alt program bir fonksiyon ise fonksiyonun çalışması bitince, sonuç olarak tek bir değer üretilir ve bu değer, fonksiyonu çağıran ifadeye döndürülerek fonksiyon çağrısının yerini alır.
- Böylece etkinlik yeniden ilk program birimine geçer.

Fonksiyonlarda dönüş kısmı

- Bir alt program çağrıldığında ve dönüş kısmına geldiğinde bellekte yapılan işlemlere bakıldığında yerel değişkenler çıkarılır.
- İlgili parametreler karşılık gelen gerçek parametrelere aktarılır.
- Dönüş adresi yığıttan alınır ve bu adresteki komutlar çalıştırılır.
- Kontrol çağrılan noktadan devam eder.

Prototip Tanımlama

- C ve C++ dilinde metot ve fonksiyonlar değişkenler gibidir.
- Çağrıldıkları yerden daha yukarıda tanımlanmış olmaları gerekir.
- Bu tanımlama metodun tamamı olabileceği gibi sadece değişken tanımı gibi tanımlama yapılabilir.
- Bu tanımlama işlemine prototip tanımlama denir.
- Prototip tanımlamada metodun içeriği yazılmaz sadece parametre türleri ve dönüş türü yazılır.
- C dilinde, C++'tan farklı olarak dönüş türü int olan fonksiyonların prototipi tanımlanması zorunlu değildir.
- Derleyici bir fonksiyonu çağrıldığı yerden daha önce bulamaz ise onun dönüş türünün int olduğunu varsayacak ve dosyada bu fonksiyonu arayacaktır.

Parametre Aktarma Yöntemleri

- C dili için alt programın (metot ya da fonksiyon) veriye erişmesinin iki yolu vardır.
- Bunlardan biri lokal olmayan değişken tanımlayıp erişmek diğeri ise parametreler.
- Lokal olmayan değişken yani global değişken tanımı önerilmeyen bir yöntemdir.
- Dolayısıyla en iyi yol parametre ile alt yordamların iletişime geçmesidir.
- Parametreler lokal değişkenlerdir ve çalışma anı yığınında tutulurlar. Fonksiyon çağrısı bittiğinde bellekten silinirler.

Parametre Aktarma Yöntemleri

- Formal parametre, fonksiyon ya da altprogramın tanımlandığı yerdeki parametrelerdir.
- Gerçek parametre ise fonksiyonun çağrıldığı noktada karşılaşılan parametrelerdir.
- Fonksiyon çağrıldığında bu gerçek parameterler ile formal parametreler arasında bir aktarım ilişkisi kurulacaktır.

Parametre Aktarma Yöntemleri- Değer ile çağırma – Pass by value

- Çağıran, çağırılana değeri direk gönderir.
- Formal parametre, asıl parametre ile ilklenmiş olur.
- Sadece gerçek parametrelerden formal parametrelere doğru bir aktarım söz konusudur.
- Bütün dillerin desteklediği güvenli bir yöntemdir.
- Aktarım sırasında formal parametre için bellekte yer ayrılır.

Parametre Aktarma Yöntemleri- Referans ile çağırma – Pass by reference

- C ve Java dilinde desteklenmeyen bu çağırım şekli C++ dilinde desteklenmektedir.
- Aktarma işlemi sırasında formal parametre için ayrı bir bellek tahsisi yapılmaz.
- Kullanılan gerçek parametrenin belleğidir.

Parametre Aktarma Yöntemleri- Gösterici (adres) ile çağırma

- Bu çağrım türünde parametre türü bir göstericidir ve çağırırken değer yerine adres gönderilir.
- Dolayısıyla adres ile çağırma olarak ta isimlendirilir.
- Gerçek ve formal parametreler ayrı ve birer gösterici olarak bellekte yer işgal eder.

- Referans ile gösterici ile çağırma yöntemi karşılaştırıldığında referans ile çağrılma tercih edilmedilir.
- Çünkü formal parametrenin gösterdiği adresin değişme durumu olabilir. Bu nedenle referans ile çağırma daha güvenlidir.

Parametre Aktarma Yöntemleri- Sonuç ile çağırma

- sonuç (out) parametresi alt programa (metot ya da fonksiyon) bir değer olarak gönderilmez bilakis fonksiyon içerisinde değer alıp gelir.
- Formal parametre lokal değişken gibi davranır ve çağırana değer olarak döner.
- C++'ta referans çağırma yöntemi kullanılarak yapılabilecek bu işlem C dilinde referans ile çağırma desteklenmediği için C dilinde sonuç ile çağırma yöntemi uygulanamaz .

Parametre Aktarma Yöntemleri- isim ile çağırma

- Popüler dillerde pek rastlanmayan aktarma yöntemidir.
- Algol dilinde destek vardır.
- Fonksiyon çağrıldığında parametre olarak değer değil de fonksiyonun adı gider.