BMB214 Programlama Dilleri Prensipleri

Ders 9. Subprograms (Alt Programlar)

Erdinç Uzun
Tekirdağ Namık Kemal Üniversitesi
Bilgisayar Mühendisliği Bölümü

Konular

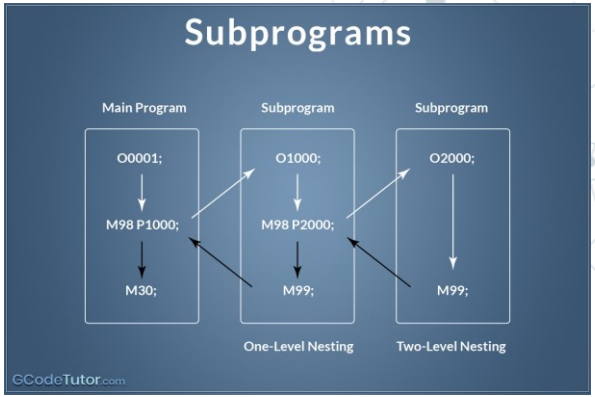
- Alt Programların (Subprograms) Temelleri
- Alt Programlar için Tasarım Sorunları
- Yerel Referans Ortamları
- Parametre Geçiş (Parameter-Passing) Yöntemleri
- Alt Program Olan Parametreler
- Alt Programları Dolaylı Olarak Çağırma
- Fonksiyonlar için Tasarım Sorunları
- Aşırı Yüklenmiş (Overloaded) Alt Programlar
- Generic Alt Programlar
- Kullanıcı Tanımlı Aşırı Yüklenmiş Operatörler
- Closures
- Coroutines

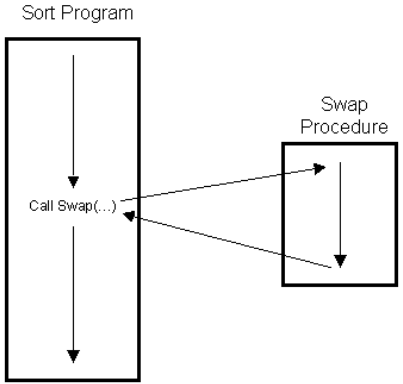
Giriş

- İki temel soyutlama yöntemi
  - İşlem soyutlaması (Process abstraction)
    - Bu derste işlenecek
  - Veri soyutlama (Data abstraction)
    - 1980'lerde vurgulanan
    - Ders 11’de ayrıntılı olarak anlatılacak

Alt Programların Temelleri (Fundamentals of Subprograms)

- Her alt programın tek bir giriş noktası vardır
- Çağıran program (calling program), çağrılan alt programın yürütülmesi sırasında askıya alınır
- Çağrılan alt programın yürütülmesi sona erdiğinde kontrol daima arayana döner





Temel tanımlar

- Bir alt program tanımı, alt program soyutlamasının arayüzünü ve eylemlerini açıklar.
  - Python'da fonksiyon tanımları çalıştırılabilir; diğer tüm dillerde çalıştırılamazlar
  - Ruby'de, fonksiyon tanımları sınıf tanımlarının içinde veya dışında görünebilir. Dışarıdaysa, bunlar Object metotlarıdır. Bir fonksyion gibi nesne olmadan çağrılabilirler.
  - Lua'da tüm fonksiyonlar anonimdir.
- Bir alt program çağrısı, alt programın çalıştırılmasına yönelik açık (explicit) bir istektir.
- Bir alt program başlığı, adı, alt programın türünü ve biçimsel parametreleri içeren tanımın ilk kısmıdır.
- Bir alt programın parametre profili (diğer adıyla imzası(signature)), parametrelerinin sayısı, sırası ve türleridir.
- Protokol, bir alt programın parametre profilidir ve dönüş türüdür (bir fonksiyonsa)

Python Tutorial: Functions def - 2020 (bogotobogo.com)

Temel Tanımlar...

- C ve C++'daki fonksiyon bildirimleri genellikle prototip (prototype) denir
- Bir alt program bildirimi (subprogram declaration), alt programın protokolünü sağlar, ancak gövdesini sağlamaz.
- Biçimsel bir parametre (formal parameter), alt program başlığında listelenen ve alt programda kullanılan bir kukla değişkendir
- Gerçek bir parametre (actual parameter), alt program call statement'ını kullanılan bir değeri veya adresi temsil eder

Gerçek / Biçimsel Parametre (Actual/Formal Parameter)

- Konumsal
  - Actual parametrelerin biçimsel parametrelere bağlanması konuma göredir: ilk gerçek parametre ilk biçimsel parametreye bağlıdır ve bu böyle devam eder.
  - Güvenli ve etkili
- Anahtar kelime
  - Gerçek bir parametrenin bağlanacağı biçimsel parametrenin adı, gerçek parametre ile belirtilir.
    - Avantaj: Parametreler herhangi bir sırada görünebilir, böylece parametre yazışma hatalarını önler
    - Dezavantaj: Kullanıcı resmi parametrenin adlarını bilmelidir

Biçimsel Parametre
Varsayılan Değerleri (Default Values)

- Bazı dillerde (örneğin, C ++, Python, Ruby, PHP), biçimsel parametreler varsayılan değerlere sahip olabilir (gerçek bir parametre verilmezse)
- C++ 'da, parametreler konumsal olarak ilişkilendirildiği için varsayılan parametreler en son görünmelidir (anahtar kelime parametresi yoktur)
- Değişken sayıda parametre C# metotları, aynı türde oldukları sürece değişken sayıda parametre kabul edebilir - karşılık gelen biçimsel parametre, **params** bulunan bir dizidir.
- Ruby'de, gerçek parametreler bir hash değişmezinin öğeleri olarak gönderilir ve karşılık gelen biçimsel parametrenin önünde bir yıldız işareti bulunur.

C++

```
// A function with default arguments, it can be called with
// 2 arguments or 3 arguments or 4 arguments.
int sum(int x, int y, int z=0, int w=0)
{
    return (x + y + z + w);
}
```

Parametrelerin Değişken Sayıları

```
def multiply(*args):
    z = 1
    for num in args:
        z *= num
    print(z)
```

- Python'da, gerçek, bir değerler listesidir ve karşılık gelen biçimsel parametre, yıldız işaretli bir addır.
- Lua'da değişken sayıda parametre, üç periyotlu biçimsel bir parametre olarak temsil edilir; bunlara bir for ifadesi ile veya üç dönemden çoklu bir atama ile erişilir

```
printResult = ""

function print (...)
  for i,v in ipairs(arg) do
    printResult = printResult .. tostring(v) .. "\t"
  end
  printResult = printResult .. "\n"
end
```

Prosedürler ve Fonksiyonlar (Procedures and Functions)

- İki alt program kategorisi vardır
  - Prosedürler, parametreli hesaplamaları tanımlayan ifadeler koleksiyonudur
  - Fonksiyonlar yapısal olarak prosedürlere benzer, ancak matematiksel fonksiyonlar üzerinde anlamsal olarak modellenmiştir.
    - Hiçbir yan etki yaratmaması bekleniyor
    - Uygulamada, program fonksiyonlarının yan etkileri vardır

Alt Programlar için Tasarım Sorunları

- Yerel değişkenler statik mi yoksa dinamik mi?
- Alt program tanımları diğer alt program tanımlarında görünebilir mi?
- Hangi parametre geçirme yöntemleri sağlanır?
- Parametre türleri kontrol ediliyor mu?
- Alt programlar parametre olarak geçirilebilir ve alt programlar iç içe yerleştirilebilirse, geçilen bir alt programın referans ortamı nedir?
- Fonksiyonel yan etkilere (side effects) izin veriliyor mu?
- Fonksiyonlardan ne tür değerler döndürülebilir?
- Fonksiyonlardan kaç değer döndürülebilir?
- Alt programlar aşırı yüklenebilir mi?
- Alt program generic olabilir mi?
- Dil iç içe geçmiş alt programlara (nested subprograms) izin veriyorsa, closures destekleniyor mu?

Yerel Referans Ortamları (Local Referencing Environments)

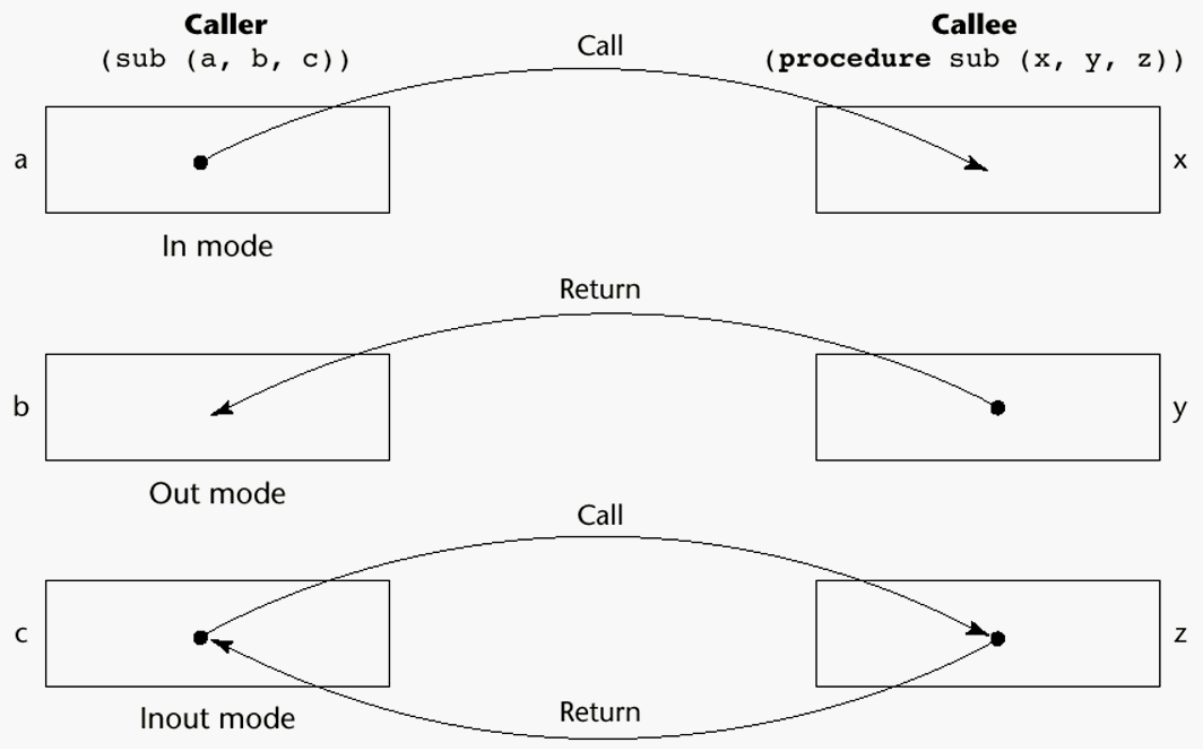
- Yerel değişkenler stack-dynamic olabilir
  - Avantajlar
    - Özyineleme desteği
    - Yereller için depolama bazı alt programlar arasında paylaşılır
  - Dezavantajları
    - Tahsis / tekrar tahsis, başlatma süresi
    - Dolaylı adresleme
    - Alt programlar geçmişe duyarlı olamaz
- Yerel değişkenler statik olabilir
  - Avantajlar ve dezavantajlar, stack-dynamic yerel değişkenler için olanların tam tersidir.

Yerel Referans Ortamları...
Örnekler

- Çoğu çağdaş dilde, yerel değişkenler stack-dynamic'tir
- C tabanlı dillerde yereller varsayılan (default) stack-dynamic, ancak static olarak bildirilebilir
- C++, Java, Python ve C# metotları yalnızca stack-dynamic yerellere sahiptir
- Lua'da, örtük olarak bildirilen tüm değişkenler (implicitly declared variables) globaldir; yerel değişkenler local olarak bildirilir ve stack-dynamic'tir.

Parametre Geçişinin Anlamsal Modelleri (Semantic Models of Parameter Passing)

- In mode
- Out mode
- Inout mode



Kavramsal Transfer Modelleri (Conceptual Models of Transfer)

- Bir değeri fiziksel olarak taşıma
- Erişim yolunu bir değere taşıma

Pass-by-Value (In Mode)

- Gerçek parametrenin değeri, karşılık gelen biçimsel parametreyi başlatmak için kullanılır.
  - Normalde kopyalayarak uygulanır
  - Bir erişim yolu iletilerek uygulanabilir, ancak önerilmez (yazma korumasını zorlamak kolay değildir)
  - Dezavantajlar (fiziksel taşıma ile ise): ek depolama gereklidir (iki kez depolanır) ve gerçek taşıma maliyetli olabilir (büyük parametreler için)
  - Dezavantajlar (erişim yolu yöntemiyle ise): çağrılan alt programda yazmaya karşı korumalı olmalı ve daha fazla maliyete erişmelidir (dolaylı adresleme)

Pass-by-Result (Out Mode)

- Sonuç (Result) olarak bir parametre geçildiğinde, alt programa hiçbir değer iletilmez; karşılık gelen biçimsel parametre yerel bir değişken olarak hareket eder; değeri, fiziksel hareketle kontrol arayana döndürüldüğünde arayanın gerçek parametresine iletilir
  - Ekstra depolama konumu ve kopyalama işlemi gerektir
- Olası sorunlar:
  - sub(p1, p1); hangi biçimsel parametre geri kopyalanırsa, p1'in mevcut değerini temsil eder.
  - sub(liste[sub], sub); Alt programın başındaki veya sonundaki [sub] listesinin adresini hesaplayın.

Pass-by-Value-Result (inout Mode)

- Pass-by-value ve pass-by-result bir kombinasyonu
- Bazen pass-by-copy olarak ta adlandırılır
- Biçimsel parametrelerin yerel depolama alanı vardır
- Dezavantajları:
  - Pass-by-result'a göre
  - Pass-by-value'e göre

Pass-by-Reference (Inout Mode)

- Bir erişim yolu (Access path) aktarın
- Pass-by-sharing olarak da adlandırılır
- Avantaj: Geçiş süreci verimlidir (kopyalama ve çoğaltılmış depolama yoktur)
- Dezavantajları
  - Biçimsel parametrelere daha yavaş erişim (pass-by-value'a kıyasla)
  - İstenmeyen yan etkilerin potansiyeli (çarpışmalar - collisions)
  - İstenmeyen takma adlar (genişletilmiş erişim)

```
fun(total, total);
fun(list[i], list[j];
fun(list[i], i);
```

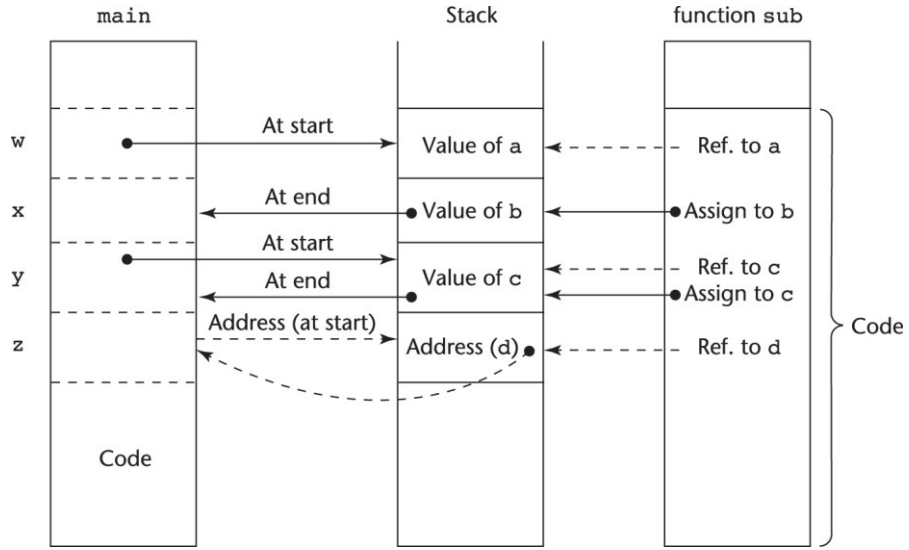
Pass-by-Name (Inout Mode)

- Metinsel Yerdeğiştirme (Textual Substitution)
- Formaller, arama sırasında bir erişim yöntemine bağlıdır, ancak bir değer veya adrese gerçek bağlanma, bir referans veya atama sırasında gerçekleşir.
- Geç bağlamada esneklik sağlar
- Uygulama, arayanın referans ortamının parametre ile geçirilmesini gerektirir, böylece gerçek parametre adresi hesaplanabilir

Parametre Geçiş Yöntemlerini Uygulama

- Çoğu dilde parametre iletişimi çalışma zamanı (run-time) stack aracılığıyla gerçekleşir
- Pass-by-reference, uygulanması en basit olanıdır; stack'e yalnızca bir adres yerleştirilir

Parametre Geçiş Yöntemlerini Uygulama…



- Fonsiyon başlığı: `void sub(int a, int b, int c, int d)`
- Main'den fonksiyonu çağırma: `sub(w, x, y, z)`
- (pass `w` by value, `x` by result, `y` by value-result, `z` by reference)

Parametre Geçiş Yöntemlerini Uygulama... Programlama Dili Örnekleri

- C
  - Pass-by-value
  - Pass-by-reference, işaretçilerin (pointer) parametre olarak kullanılmasıyla elde edilir
- C++
  - Pass-by-reference: Referans türü (reference type) adı verilen özel bir işaretçi türü (special pointer type)
- Java
  - Tüm parametreler değere göre geçirilir (pass-by-value)
  - Object parametreleri referansla aktarılır (pass-by-reference)

Parametre Geçiş Yöntemlerini Uygulama... Programlama Dili Örnekleri

- Fortran 95+
  - Parametreler in, out, veya inout modunda ilan edilebilir
- C#
  - Varsayılan yöntem: pass-by-value
  - Pass-by-reference, hem biçimsel bir parametrenin hem de gerçek parametresinin ref ile önce gelmesiyle belirtilir.
- PHP: C # 'a çok benzer, tek fark, gerçek veya biçimsel parametrenin ref belirtebilmesi dışında (Değişkenin başına & işareti eklenir)
- Perl: tüm gerçek parametreler örtük olarak @_ adında önceden tanımlanmış bir diziye yerleştirilir.
- Python ve Ruby pass-by-assignment (tüm veri değerleri nesnelerdir) kullanır

Tür Denetim Parametreleri (Type Checking Parameters)

- Güvenilirlik (reliability) için çok önemlidir
- FORTRAN 77 ve orijinal C: yok
- Pascal ve Java: her zaman gereklidir
- ANSI C ve C ++: seçim kullanıcı tarafından yapılır
  - Prototypes
- Nispeten yeni diller Perl, JavaScript ve PHP tür denetimi gerektirmez
- Python ve Ruby'de değişkenlerin türleri yoktur (nesneler vardır), bu nedenle parametre türü denetimi mümkün değildir

Parametreler Olarak Çok Boyutlu Diziler (Multidimensional Arrays as Parameters)

- Çok boyutlu bir dizi bir alt programa aktarılırsa ve alt program ayrı olarak derlenirse, derleyicinin depolama eşleme fonksiyonu (storage mapping function) oluşturmak için bu dizinin bildirilen boyutunu bilmesi gerekir.

Parametreler Olarak Çok Boyutlu Diziler…
C ve C++

- Programcı, gerçek parametrede ilk alt simge hariç tümünün beyan edilen boyutlarını dahil etmek zorundadır.
- Esnek alt programlar yazmaya izin vermez
- Çözüm: diziye ve boyutların boyutlarına diğer parametreler olarak bir işaretçi (pointer) iletin; kullanıcı, boyut parametreleri açısından depolama haritalama fonksiyonuna dahil etmelidir

```
// n must be passed before the 2D array
void print(int m, int n, int arr[][n])
{
    int i, j;
    for (i = 0; i < m; i++)
      for (j = 0; j < n; j++)
        printf("%d ", arr[i][j]);
}
```

```
void print(int *arr, int m, int n)
{
    int i, j;
    for (i = 0; i < m; i++)
      for (j = 0; j < n; j++)
        printf("%d ", *((arr+i*n) + j));
}
```

Parametreler Olarak Çok Boyutlu Diziler… Java ve C#

- Ada diline benzer
- Diziler nesnelerdir; hepsi tek boyutludur, ancak öğeler diziler olabilir
- Her dizi, dizi nesnesi oluşturulduğunda dizinin uzunluğuna ayarlanan adlandırılmış bir sabiti (Java'da length, C#'da Length) kullanır.

Parametre Geçişi için Tasarımla İlgili Hususlar

- İki önemli husus
  - Verimlilik
  - Tek yönlü (one-way) veya iki yönlü (two-way) veri aktarımı
- Ancak yukarıdaki hususlar çelişkili
  - İyi programlama, değişkenlere sınırlı erişim sağlar, bu da mümkün olduğunda tek yönlü anlamına gelir
  - Ancak pass-by-reference, önemli boyuttaki yapıları geçmek için daha etkilidir

Alt Program Adları olan Parametreler

- Bazen alt program adlarını parametre olarak geçirmek uygundur.
- Sorunlar:
  - Parametre türleri kontrol ediliyor mu?
  - Parametre olarak gönderilen bir alt program için doğru referans ortamı nedir?

Alt Program Adları olan Parametreler: Referans Ortamı

- Gölge bağlama (Shallow binding): Geçilen alt programı etkinleştiren çağrı ifadesinin ortamı
  - Dinamik kapsamlı (dynmamic-scoped) diller için en doğalı
- Derin bağlama (Deep binding): Geçilen alt programın tanımının ortamı
  - Statik kapsamlı (static-scoped) diller için en doğalı
- Ad-hoc bağlama: Alt programı geçen çağrı ifadesinin ortamı

Alt Program Adları olan Parametreler: Referans Ortamı...

```
function sub1() {
  var x;
  x = 1;
  function sub2() {  // this defines sub2
    print(x);
    };
  function sub3() {
    var x;
    x = 3;
    sub4(sub2);  // this passes sub2
    };
  function sub4(subx) {
    var x;
    x = 4;
    subx();  // this enacts sub2
    };
  sub3();
};
```

shallow - output is 4
deep - output is 1
ad hoc - output is 3

Alt Programları Dolaylı Olarak Çağırma

- Genellikle çağrılması gereken birkaç olası alt program olduğunda ve programın belirli bir çalışmasında doğru olanı yürütülene kadar bilinmediğinde (örneğin, olay işleme (event handling) ve GUI'ler)
- C ve C++ 'da, bu tür çağrılar fonksiyon işaretçileri (function pointers) aracılığıyla yapılır

Alt Programları Dolaylı Olarak Çağırma...

- C# 'da, metot işaretçileri (method pointers) delegates adı verilen nesneler olarak uygulanır.
  - Delegate bildirimi:
    - public delegate int Alter(int x);
  - Alter adlı bu delegate türü, int parametresi alan ve int değeri döndüren herhangi bir metotla başlatılabilir
  - Bir metot:
    - static int fun1(int x) {…}
  - Örnekleme (Instantiate):
    - Alter altfun1 = new Alter(fun1);
  - Şununla çağrılabilir:
    - altfun1(12);
- Bir delegate, çok noktaya yayın delege (multicast delegate) adı verilen birden fazla adresi depolayabilir

Fonksiyonlar için Tasarım Sorunları

- Yan etkilere izin veriliyor mu?
  - Yan etkiyi azaltmak için parametreler her zaman in-mode olmalıdır (Ada gibi)
- Ne tür dönüş değerlerine izin verilir?
  - Emir esaslı dillerin çoğu dönüş türlerini kısıtlar
  - C, diziler ve fonksiyonlar dışında herhangi bir türe izin verir
  - C++, C gibidir ancak kullanıcı tanımlı türlere de izin verir
  - Java ve C# metotları herhangi bir türü döndürebilir (ancak metotlar tür olmadığından döndürülemezler)
  - Python ve Ruby, yöntemleri birinci sınıf nesneler olarak ele alır, böylece diğer sınıfların yanı sıra geri döndürülebilirler.
  - Lua, fonksiyonların birden çok değer döndürmesine izin verir

Aşırı Yüklenmiş Alt Programlar (Overloaded Subprograms)

- Aşırı yüklenmiş bir alt program, aynı referans ortamında başka bir alt programla aynı adı taşıyan bir alt programdır.
  - Aşırı yüklenmiş bir alt programın her sürümünün benzersiz bir protokolü vardır
- C++, Java, C # ve Ada önceden tanımlanmış (predefined) aşırı yüklenmiş alt programları içerir
- Ada'da, aşırı yüklenmiş bir fonksiyon dönüş türü, çağrıları netleştirmek için kullanılabilir (böylece iki aşırı yüklenmiş fonksiyon aynı parametrelere sahip olabilir)
- Ada, Java, C++ ve C#, kullanıcıların aynı ada sahip birden çok alt program sürümü yazmasına olanak tanır

Generic Alt Programlar

- Generic veya polimorfik (polymorphic) bir alt program, farklı etkinleştirmelerde farklı türlerdeki parametreleri alır
- Aşırı yüklenmiş alt programlar geçici (ad hoc) polimorfizm sağlar
- Alt tür polimorfizmi (Subtype polymorphism), T türü bir değişkenin, T türündeki herhangi bir nesneye veya T'den türetilen herhangi bir türe erişebileceği anlamına gelir (OOP dilleri)
- Alt programın parametrelerinin türünü tanımlayan bir tür ifadesinde kullanılan genel bir parametreyi alan bir alt program, parametrik polimorfizm (parametric polymorphism) sağlar
  - Dinamik bağlama için ucuz bir derleme zamanı

Generic Alt Programlar
C++

- Generic bir alt programın sürümleri, alt program bir çağrıda adlandırıldığında veya adresi & operatörüyle alındığında dolaylı olarak oluşturulur.
- Generic alt programlardan önce, tür adları veya sınıf adları olabilen genel değişkenleri listeleyen bir template clause bulunur.

```
template <class Type>
    Type max(Type first, Type second) {
    return first > second ? first : second;
    }
```

Generic Alt Programlar
Java 5.0

- Java 5.0 - Java 5.0'daki generic'ler ile C++'kiler arasındaki farklar:
  - Java 5.0'daki genel parametreler sınıflar olmalıdır
  - Java 5.0 generic metotlar, yalnızca bir kez örneklenir
  - Kısıtlamalar (Restrictions), generic metoda generic parametreler olarak aktarılabilen sınıf aralığı üzerinde belirtilebilir
  - Genel parametrelerin joker türleri (wildcard types)

Generic Alt Programlar
Java 5.0...

```
public static <T> T doIt(T[] list) { … }
```

- Parametre, generic öğeler dizisidir (T, türün adıdır)
- Bir çağırım:

```
doIt<String>(myList);
```

- Genel parametrelerin sınırları olabilir:

```
public static <T extends Comparable> T
    doIt(T[] list) { … }
```

- Generic tür, Comparable arabirimini uygulayan bir sınıf olmalıdır

Generic Alt Programlar
Java 5.0...

- Wildcard types (Joker Türleri)
  `Collection<?>` koleksiyon sınıfları için bir joker karakter türüdür

```
void printCollection(Collection<?> c) {
    for (Object e: c) {
      System.out.println(e);
    }
  }
```

- Herhangi bir koleksiyon sınıfı için çalışır

Generic Alt Programlar
C# 2005

- Java 5.0'dakilere benzer generic metotları destekler
- Bir fark: derleyici belirtilmemiş türü çıkarabiliyorsa, çağrıda gerçek tür parametreleri (actual type parameters) göz ardı edilebilir
- Diğer taraftan C# 2005 joker karakterleri desteklemiyor

Generic Alt Programlar
F#

- Bir parametrenin türünü veya bir fonksiyonun dönüş türünü belirleyemiyorsa generic bir tür çıkarır - otomatik genelleme (automatic generalization)
- Bu türler bir kesme işareti ve tek bir harfle gösterilir, örneğin, ′a gibi
- Fonksiyonlar, generic parametrelere sahip olacak şekilde tanımlanabilir

```
let printPair (x: 'a) (y: 'a) =
    printfn "%A %A" x y
```

  - %A, her tür için bir biçim kodudur
  - Bu parametreler tür kısıtlamalı değildir

Generic Alt Programlar
F#

- Bir fonksiyonun parametreleri aritmetik operatörlerle kullanılıyorsa, parametreler generic olarak belirtilse bile bunlar tür kısıtlıdır.
- Tür çıkarımı ve tür zorlamalarının olmaması nedeniyle, F# generic fonksiyonlar C++, Java 5.0+ ve C# 2005+ fonksiyonlarına göre çok daha az kullanışlıdır.

Kullanıcı Tanımlı Aşırı Yüklenmiş Operatörler (User-Defined Overloaded Operators)

- Ada, C++, Python ve Ruby'de operatörler aşırı yüklenebilir
- Python örneği

```
def __add__ (self, second) :
  return Complex(self.real + second.real,
                 self.imag + second.imag)
```

  - Hesaplama: x + y, x.__add__(7)

Closures

- Closure, bir alt program ve tanımlandığı referans ortamdır.
  - Alt program, programdaki herhangi bir rastgele yerden çağrılabiliyorsa, referanslama ortamı gereklidir.
  - İç içe geçmiş alt programlara (nested subprograms) izin vermeyen statik kapsamlı bir dil, closure'lara ihtiyaç duymaz
  - Closure'lar yalnızca, bir alt program iç içe yerleştirme kapsamlarındaki değişkenlere erişebiliyorsa ve herhangi bir yerden çağrılabiliyorsa gereklidir.
  - Closure desteklemek için, bir uygulamanın bazı değişkenlere sınırsız kapsam sağlaması gerekebilir (çünkü bir alt program normalde artık canlı olmayan yerel olmayan bir değişkene erişebilir)

Closures
Javascript

```
function makeAdder(x) {
  return function(y) {return x + y;}
}
...
var add10 = makeAdder(10);
var add5 = makeAdder(5);
document.write("add 10 to 20: " + add10(20) +
               "<br />");
document.write("add 5 to 20: " + add5(20) +
               "<br />");
```

Closure `makeAdder` tarafından döndürülen anonim fonksiyondur.

Closure
C#

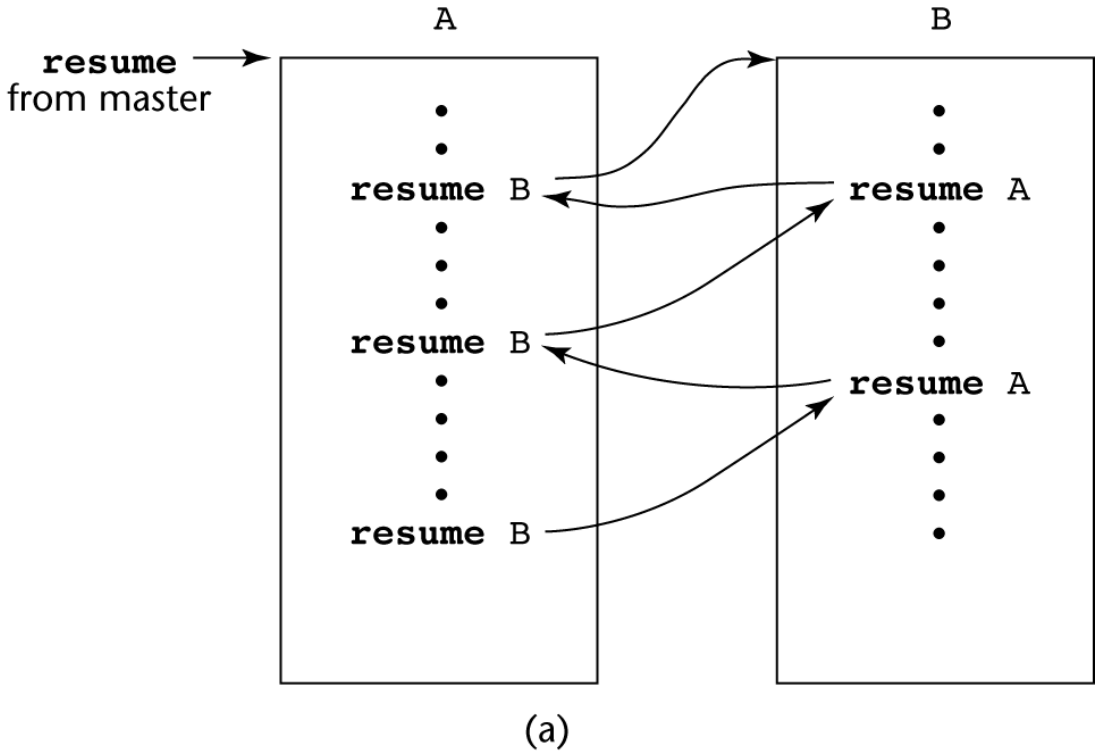
- İç içe geçmiş anonim bir delegate kullanarak aynı closure'u C#'da yazabiliriz
- func<int, int> (dönüş türü), parametre olarak bir int alan ve int ile dönen bir delegate belirtir

```
static Func<int, int> makeAdder(int x) {
   return delegate(int y) {return x + y;};
}
...
Func<int, int> Add10 = makeAdder(10);
Func<int, int> Add5 = makeAdder(5);
Console.WriteLine("Add 10 to 20: {0}", Add10(20));
Console.WriteLine("Add 5 to 20: {0}", Add5(20));
```

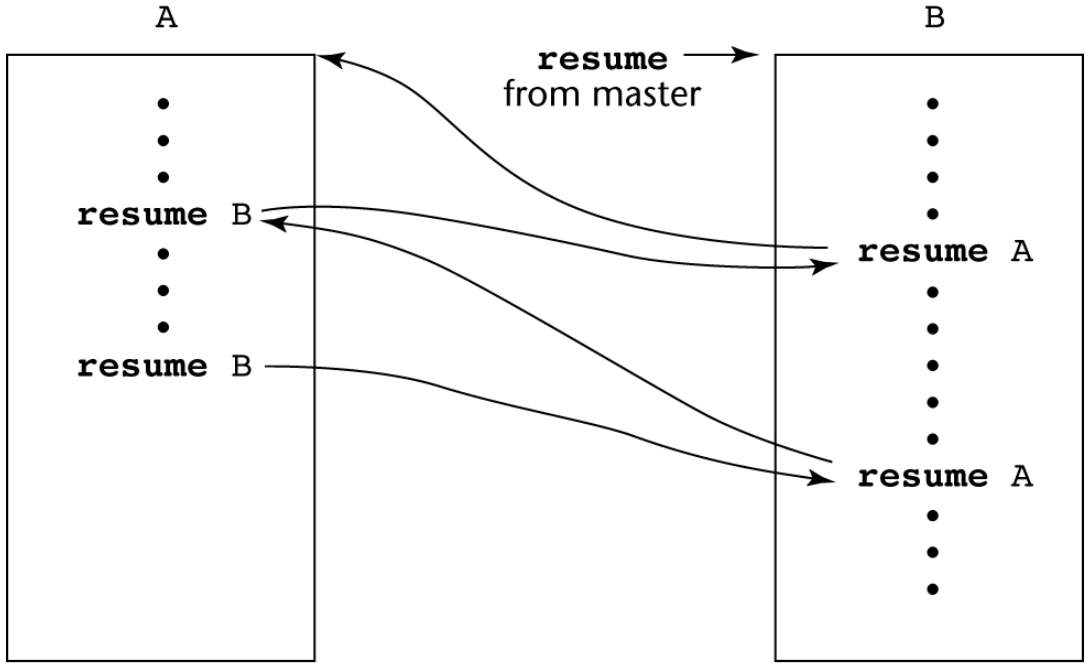
Coroutines

- Bir coroutine, birden çok girdiye sahip olan ve bunları kendisi kontrol eden bir alt programdır - doğrudan Lua'da desteklenir
- Simetrik kontrol (*symmetric control)* olarak da adlandırılır: arayan (caller) ve çağrılan coroutine'ler daha eşit bir temeldedir
- Bir coroutine çağrısı özgeçmiş (resume) olarak adlandırılır
- Bir coroutine'nin ilk özgeçmişi başlangıcıdır, ancak sonraki çağrılar, coroutine'de son çalıştırılan ifadeden hemen sonraki noktaya girer.
- Coroutine'ler, muhtemelen sonsuza kadar birbirlerini tekrar tekrar devam ettirir
- Coroutine'ler, program birimlerinin yarı eşzamanlı yürütülmesini (*quasi-concurrent execution)* sağlar; yürütmeleri aralıklıdır, ancak örtüşmez

Coroutine
Olası Yürütme Denetimleri (Possible Execution Controls)



Coroutine
Olası Yürütme Denetimleri (Possible Execution Controls)



(b)

Coroutine
Olası Yürütme Denetimleri (Possible Execution Controls)