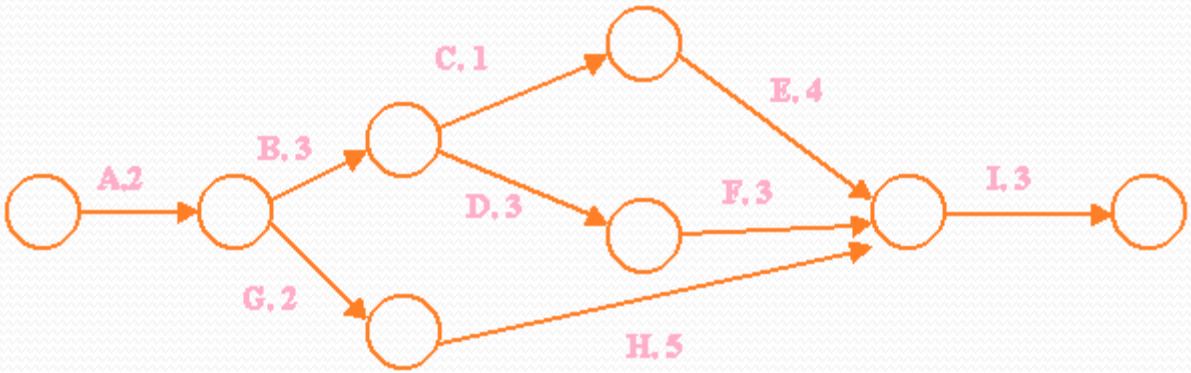
PROGRAMLAMA TEMELLERİ

Altyordamlar

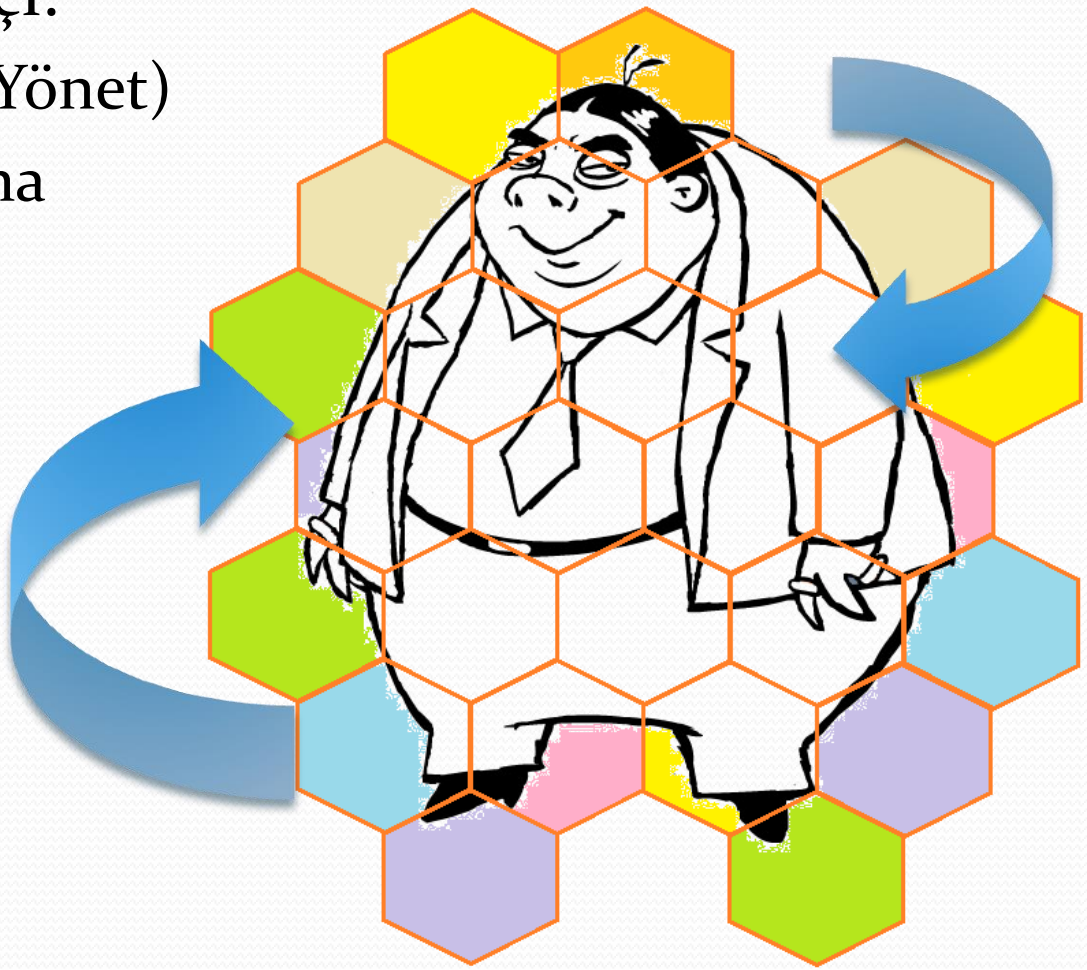
Algoritma Tasarım Teknikleri

- Bir problemin etkili çözümü için bir yapı oluşturmaya yönelik genel yaklaşımlar
- Bu teknikler geniş çapta, çeşitli problemlerin çözümü için bir şablon sunarlar.
- Üst seviye dillerin desteklediği kontrol deyimlerine ve veri yapılarına kolayca dönüştürülebilirler.
- Algoritmaların gereksinimleri kati olarak çözümlenebilir.



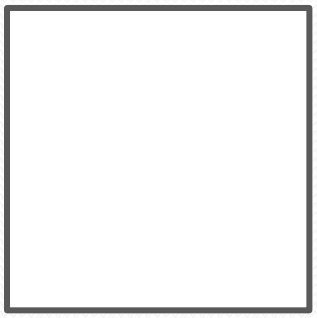
Algoritma Tasarım Teknikleri

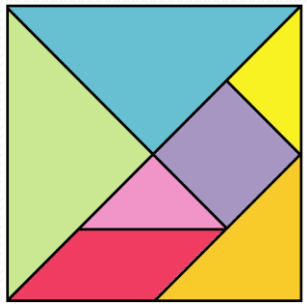
- Bu tekniklerden birkaçı:
  - Böl ve Fethet (Böl ve Yönet)
  - Dinamik Programlama
  - Obur Yöntemi
  - Geri izleme

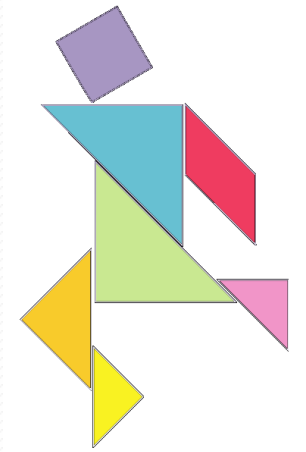


Böl ve Fethet: Yöntem

- Yöntem:
  - Verilen problemi bir çok küçük alt parçalara böl,
  - Her bir parçayı ayrı ayrı çöz,
  - Sonra her bir parçanın çözümünü –orijinal problemin çözümü olacak şekilde- birleştir.

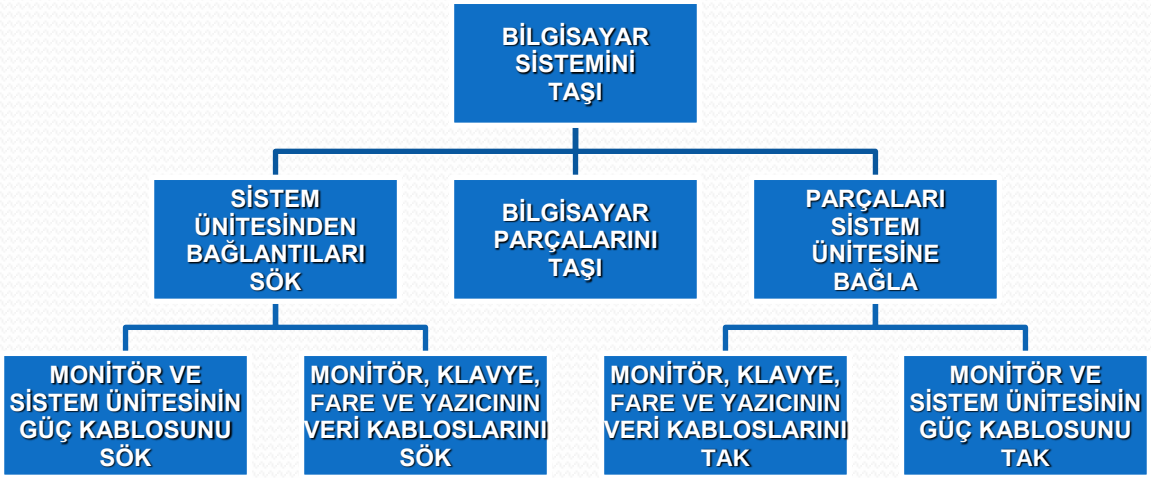






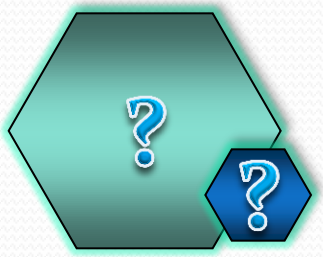
Böl ve Fethet: Yöntem

- Bir bilgisayar sistemini bir kerede taşıyamazsınız.
  - Önce bağlantıları kesin
  - Her bir parçayı ayrı ayrı taşıyın
  - Yeni yerinde sistemin bağlantılarını yeniden yapın.



Böl ve Fethet: Özellikler

- Bu yöntemde verilmesi gereken en önemli karar:
  - alt parçaların ne kadar küçük olmalı.
- Karar verirken her bir alt parçanın başka problemlerin de bir alt parçası olabileceğini akılda tutmak gerekir.
- Her bir alt parça, bir bütün olarak çözülebilmeli.
- Her bir parçanın çözümü bağımsız bir işlevi olmalı.
- Gerekirse bir alt parça aynı şekilde daha alt parçalara bölünebilir.



Böl ve Fethet: Örnek

- Problem: Bir miktar sayıyı büyükten küçüğe sıralama.
- Çözüm:
  1. "N" adet Sayı oku.
  2. "i" 1 olsun.
  3. Bir sayıdan fazla sayı olduğu sürece (i<N)
     a) Kalanlar arasında ("i" ve "N" arasındaki) en büyük sayıyı bul.
     b) En büyük sayıyı "i." sayı ile değiştir.
     c) Bir sonraki sıraya geç ("i"yi 1 artır).
  4. "N" adet sayı yazdır.

Böl ve Fethet: Örnek

- Çözümün alt parçaları
  - Sayıların okunması (1. adım)
  - “i-N” arasındaki en büyük sayının bulunması (3a. Adım)
  - En büyük sayıyı “i.” sayı ile değiştir. (3b. Adım)
  - Sayıların yazdırılması. (4. adım)
- Her bir parça
  - Küçük ama genel bir problemdir.
  - Çözümü bir işlevi yerine getirecektir.
  - Daha alt parçaya bölünemez.
  - Başka problem çözümünde de kullanılabilir.

Böl ve Fethet → Altyordam

- Böl ve fethet yönteminin her bir alt parçası altyordam olarak tasarlanır ve kodlanır.
- Altyordam veya yordam, alt program, modül, prosedür, fonksiyon gibi kelimeler eş anlamlı olarak kullanılır. Orijinali “subroutine”, “sub program”, “module”, “procedure”, “function”.

Altyordam

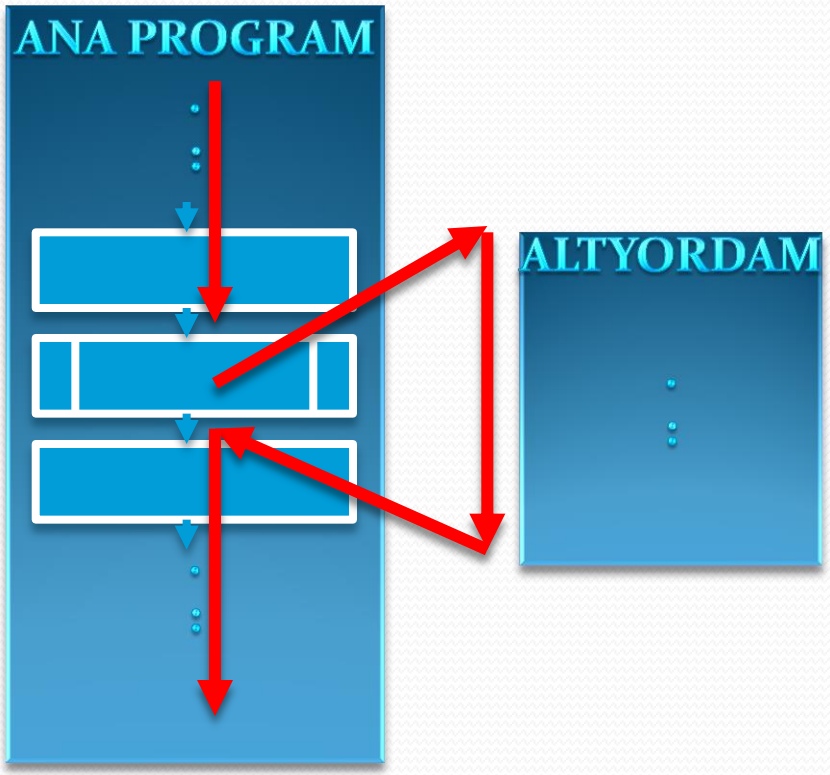
- altyordam İng. subroutine, subprogram
  - Bir izlencede gereksenen belli bir işlevi, parametreler aracılığıyla kendisine verilen değişkenlerden yararlanarak, bunlara uygun işlem dizilerini yürüterek gören bir yordam ya da izlence kesimi. Yordamlıkta saklanan bir altyordamı birçok izlence gerektikçe kullanabileceği gibi, bir izlence değişik yerlerde birden çok kez de çağrılabilir. İşlem sırasının denetimi, altyordamda öngörülen işlemler bitirildikten sonra, genel kural olarak altyordamı çağıran komutu izleyen ana izlence komutuna geçer.
- BSTS / Bilişim Terimleri Sözlüğü 1981

Altyordam

- altyordam yani ...
- Bir problemin belli bir işlevi, parametreler olarak gelen değerlerden yararlanarak, bunlara uygun komutları yürüterek gören program parçası.
- Bir altyordam birçok programda gerektikçe kullanabilir.
- Bir program, bir altyordamı değişik yerlerde birden çok kez çağırabilir.

Altyordam

- İşlem sırası, altyordamda öngörülen işlemler bitirildikten sonra, genel kural olarak altyordamı çağıran komutu izleyen ana program komutuna geçer.

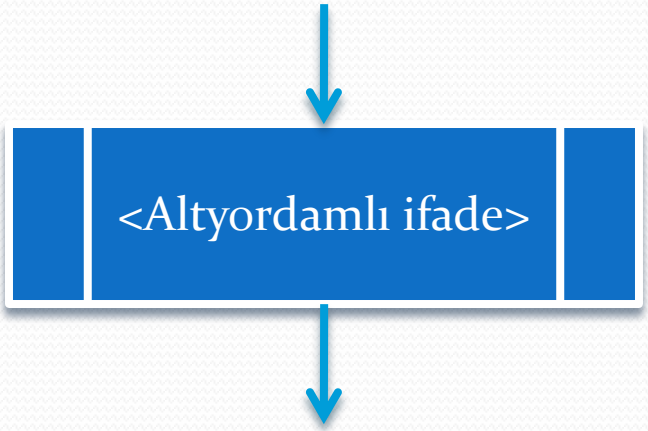


Altyordam

- Genel olarak ve gerekmedikçe
- **Girdiler**: altyordam içerisinde klavyeden okutulmaz, ana programdan parametreler aracılığıyla alınır.
- **Çıktılar**: altyordamda ekrana yazılmaz, dönüş değeri veya parametreler aracılığıyla ana programa gönderilir.

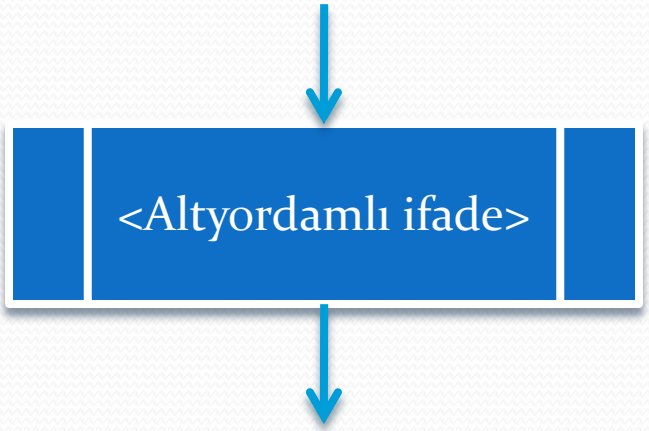
Altyordam: Akış Diyagramı

- Ana program içerisinden bir alt program çağırmak istiyorsanız;



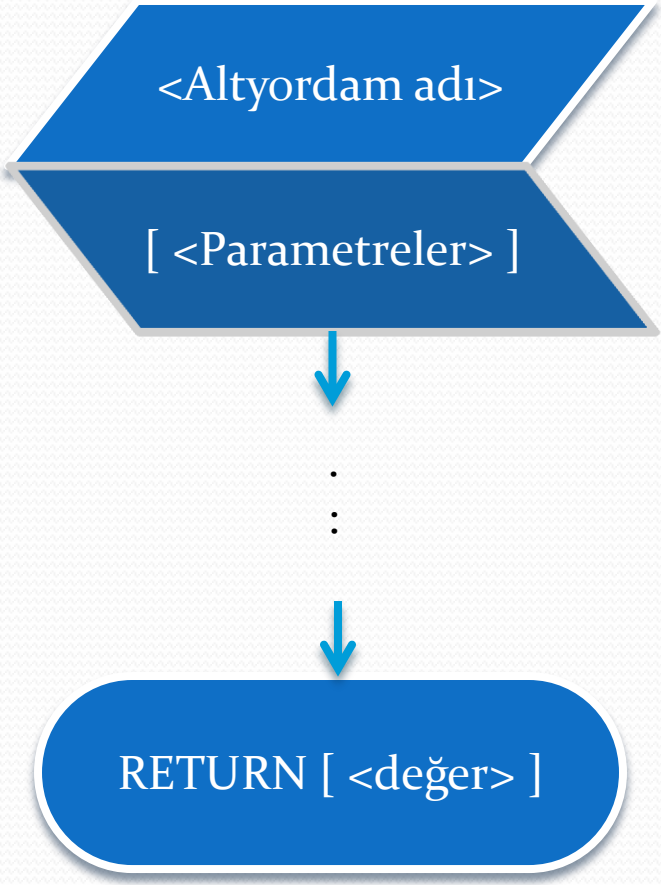
Altyordam: Akış Diyagramı

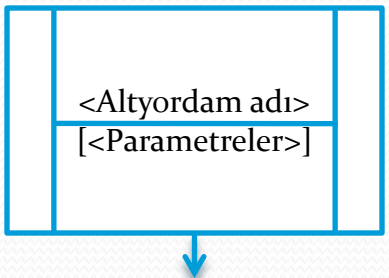
- Bazı altyordamlar sadece ismi ile çağrılır
  `RastgeleSayi()`
- Bazı altyordamlara sadece parametre girilir
  `DiziyiYaz(Dizi, N)`
- Bazı altyordamların ürettiği sonuçlar ana programda işlenir
  `x1← (-b-karekok(d))/(2*a)`
- Veya bir değişkene atanır
  `a2←karekok(a)`



Altyordam: Akış Diyagramı

- Altyordam tanımlamak için yüz yüze iki paralel kenar kullanılır. Üstteki paralel kenara altyordamın adı, alttakine ise –varsa- parametre listesi yazılır.

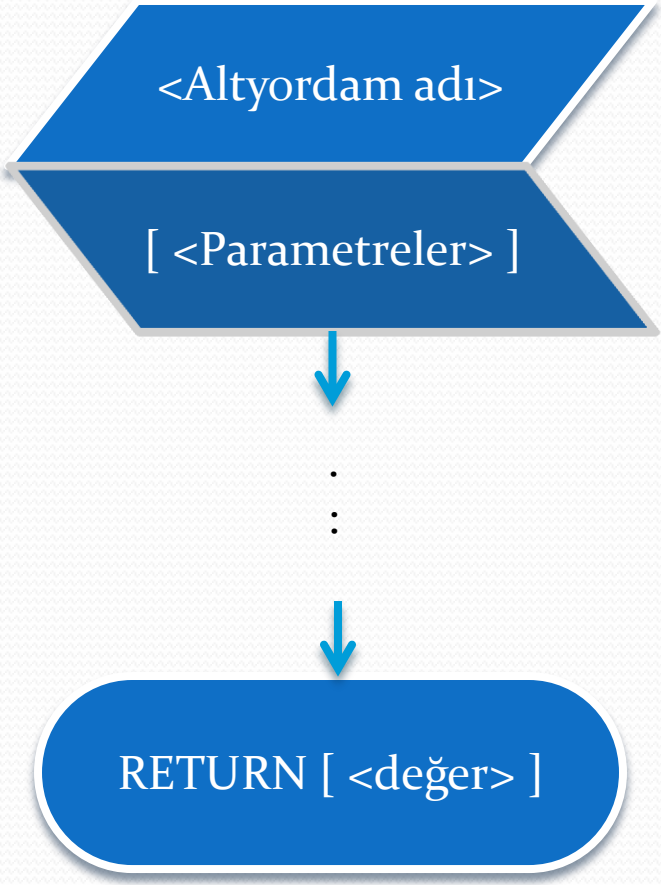




!
Bazı kaynaklarda soldaki sembole rastlayabilirsiniz.

Altyordam: Akış Diyagramı

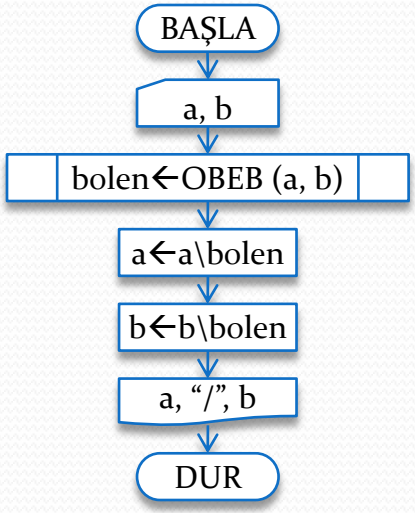
- Altyordamı sonlandırmak için sonlandırıcı kullanılır. Sembol içine “RETURN”, “GERİ DÖN” yazılır.
- Bazı altyordamlar değer döndürebilir. Böyle durumlarda sonlandırıcı sembole dönüş değeri veya dönüş değerini tutan değişken yazılır.



Altyordam: Örnek

- Klavyeden girilen “a/b” şeklindeki bir rasyonel sayının sadeleştirilmesi

1. READ a, b
2. Bolen ← OBEB (a, b)
3. a←a \ bolen
4. b←b \ bolen
5. WRITE a, “ / “, b

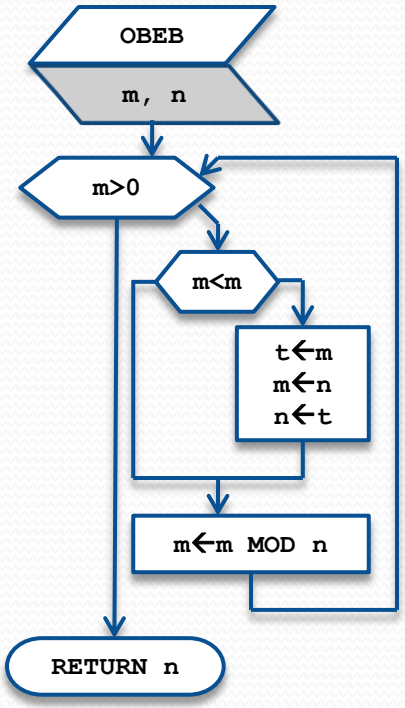


- Burada OBEB bizim yazacağımız bir altyordamdır.
- Alt problemimiz iki tamsayınıın OBEBini bulmaktır.

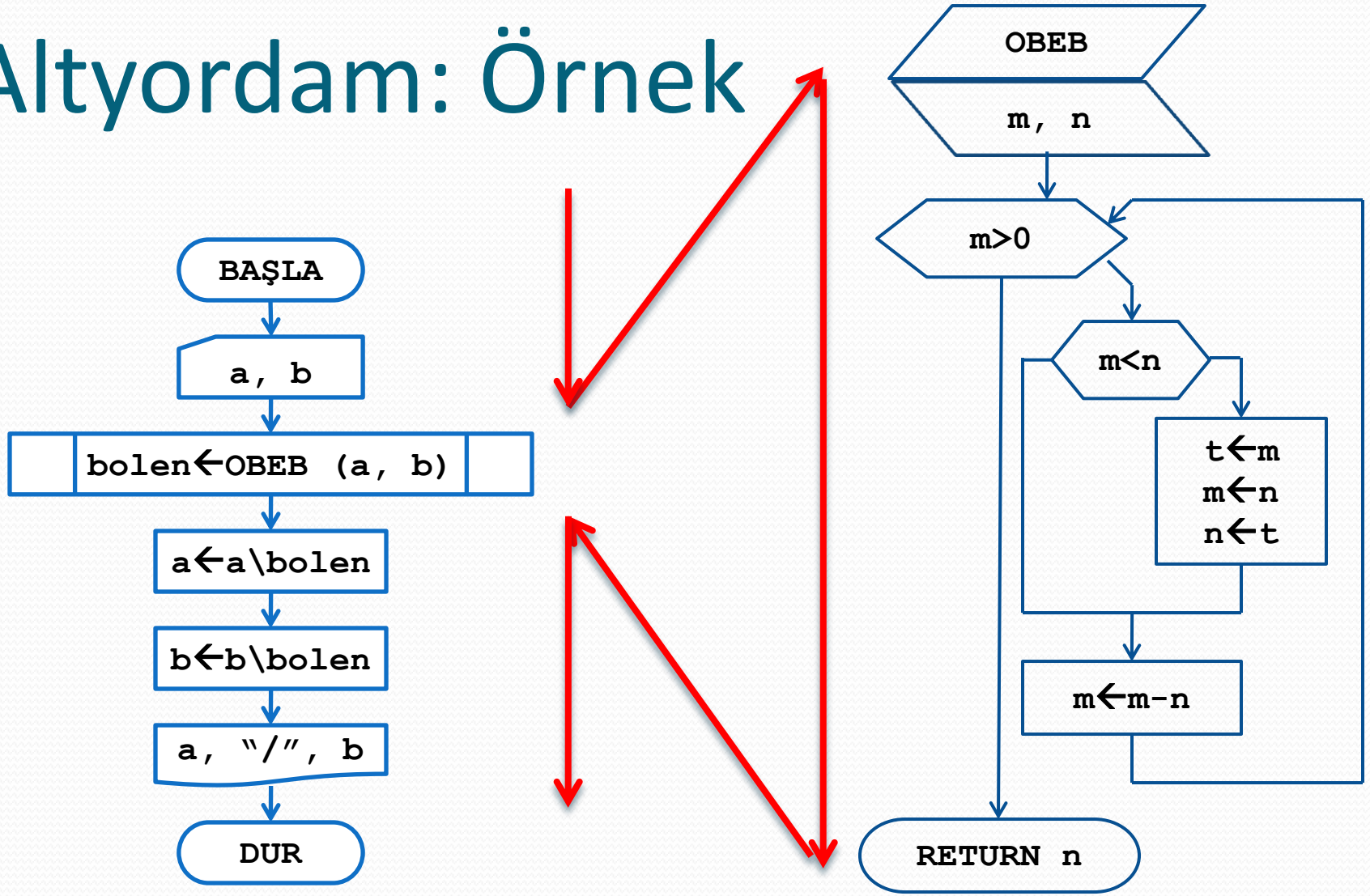
Altyordam: Örnek

- Yeni problem: iki tamsayının OBEBi
- Bu problemi Euclid'in yaklaşımı ile çözelim.

1. WHILE m>o
   a) IF m<n
      i. t←m
      ii. m←n
      iii. n←t
   b) m←m MOD n
2. Obeb←n



Altyordam: Örnek



Neden Altyordam?

- Programlar kısalır
  - Bir çok programda bazı işlemler tekrarlanabilir. Aynı kod parçalarını tekrar tekrar yazmak yerine alt program olarak bir kere yazıp defalarca çağırabiliriz. Böylece uzun ve tekrar eden kod parçalarının yerine artık tek satırlık altyordam çağrıları kullanabiliriz.
- Programlar daha kolay anlaşılır
- Daha verimlidir
- Programlarda hata olasılığı azalır
- Kodlar paylaşılabilir
- Daha az tasarım zamanı

Neden Altyordam?

- Programlar kısalır
- Programlar daha kolay anlaşılır
  - Kod uzadıkta takip etmek zorlaşır. Altyordam kullanarak kod kısaltmış oluruz. Programımız daha düzenli görünür. Ayrıca Programa modül olarak bakarsak, blokların görevini tek tek anladıktan sonra ana programda bunu birleştirmek analizi ve tasarımı da kolaylaştıracaktır.
- Daha verimlidir
- Programlarda hata olasılığı azalır
- Kodlar paylaşılabilir
- Daha az tasarım zamanı

Neden Altyordam?

- Programlar kısalır
- Programlar daha kolay anlaşılır
- Daha verimlidir
  - Altyordam kullanmak işlemcinin fazladan bazı saklama işlemleri yapmasına neden olur ama bu faydalarının yanında sayılamayacak kadar faydası vardır. Kod yazmayı kolaylaştırması, kodlama zamanını azaltması, hata oranını azaltması gibi etkenler verimliliği artırır.
- Programlarda hata olasılığı azalır
- Kodlar paylaşılabilir
- Daha az tasarım zamanı

Neden Altyordam?

- Programlar kısalır
- Programlar daha kolay anlaşılır
- Daha verimlidir
- Programlarda hata olasılığı azalır
  - Altyordam olarak yazılan kodlar ayrı ayrı test edilebilir. Kısa kod bloklarının yani altyordamların hatalarını gidermek, uzun kod blokları içerisinde hatanın kaynağını bile bulmaktan daha kolaydır. Ayrıca daha önceden kullanılmış ve test edilmiş altyordamları da kullanabileceğimize göre bu altyordamları kullanan programlar yazmak daha sağlıklı olacaktır.
- Kodlar paylaşılabilir
- Daha az tasarım zamanı

Neden Altyordam?

- Programlar kısalır
- Programlar daha kolay anlaşılır
- Daha verimlidir
- Programlarda hata olasılığı azalır
- Kodlar paylaşılabilir
  - Bazı altyordamları başka programlarda da kullanmak mümkündür. Böylece kodları paylaşabiliriz. Bu bize kodlama zamanını azaltmak, hata oranını azaltmak gibi bazı avantajlar da sağlayacaktır.
- Daha az tasarım zamanı

Neden Altyordam?

- Programlar kısalır
- Programlar daha kolay anlaşılır
- Daha verimlidir
- Programlarda hata olasılığı azalır
- Kodlar paylaşılabilir
- Daha az tasarım zamanı
  - Zor bir problemin tüm girdi, çıktı ve işlem bölümlerini tasarlamak zor olabilir. Ama altyordamlara bölünmüş bir problemle başa çıkmak daha kolay olacaktır. Ayrıca gerektiğinde daha önceden kodlanmış altyordamları programa dahil etmek kodlamadan zaman kazanmamızı sağlayacaktır. Hem aynı kodu tekrar tekrar neden yazalım ki, daha önceden yazılmışı varsa?

Neden Altyordam?

```
Program Prosedursuz;
Uses winCrt;
Var Counter : Integer;
Begin
GotoXy(10,5);
For Counter := 1 to 10 do
 Begin                {Step [1]}
  write(chr(196)); {Step [2]}
 End;                 {Step [3]}
GotoXy(10,6);
For Counter := 1 to 10 do
Begin                {Step [1]}
write(chr(196)); {Step [2]}
 End;                {Step [3]}
 GotoXy(10,7);
 For Counter := 1 to 10 do
 Begin                {Step [1]}
 write(chr(196)); {Step [2]}
 End;                 {Step [3]}
 GotoXy(10,10);
 For Counter := 1 to 10 do
 Begin                {Step [1]}
 write(chr(196)); {Step [2]}
 End;                 {Step [3]}
 Readkey;
 End.
```

```
Program Prosedurle;
Uses WinCrt;,
Procedure DrawLine;
{Bu prosedür 1,2,3 adımlarının tekrarını önlüyor.}
Var Counter : Integer;
Begin
  For Counter := 1 to 10 do
    Begin                  {Step [1]}
       write(chr(196)); {Step [2]}
     End;                  {Step [3]}
 End;

Begin
 GotoXy(10,5);
 DrawLine;
 GotoXy(10,6);
 DrawLine;
 GotoXy(10,7);
 DrawLine;
 GotoXy(10,10);
 DrawLine;
 Readkey;
End.
```

Neden Altyordam?

```
Program Prosedursuz;
Uses winCrt;
Var Counter : Integer;
Begin
GotoXy(10,5);          {Step [1]}
For Counter := 1 to 10 do
 Begin                 {Step [2]}
  write(chr(196));     {Step [3]}
 End;                  {Step [4]}
GotoXy(10,6 );         {Step [1]}
For Counter := 1 to 10 do
Begin                  {Step [2]}
write(chr(196));       {Step [3]}
 End;                  {Step [4]}
 GotoXy(10,7 );        {Step [1]}
 For Counter := 1 to 10 do
 Begin                 {Step [2]}
 write(chr(196));      {Step [3]}
 End;                  {Step [4]}
 GotoXy(10,10 );       {Step [1]}
 For Counter := 1 to 10 do
 Begin                 {Step [2]}
 write(chr(196));      {Step [3]}
 End;                  {Step [4]}
 Readkey;
End.
```

```
Program ProsedurVeParametreyle;
Uses WinCrt;,
Procedure DrawLine(x, y);
{Bu prosedür 1-4 adımlarının tekrarını önlüyor.}
Var Counter : Integer;
Begin
  GotoXY(x, y)             {Step [1]}
  For Counter := 1 to 10 do
    Begin                  {Step [2]}
       write(chr(196));    {Step [3]}
     End;                  {Step [4]}
 End;

Begin
  DrawLine(10,5);
  DrawLine(10,6);
  DrawLine(10,7);
  DrawLine(10,10);
 Readkey;
End.
```