BURDUR
MEHMET AKİF ERSOY ÜNİVERSİTESİ
TEKNİK BİLİMLER MESLEK YÜKSEKOKULU

ALGORİTMA TEMELLERİ VE PROGRAMLAMAYA GİRİŞ

ÖĞR. GÖR. BUĞRA DAĞCI

2025-2026 GÜZ DÖNEMİ

PROGRAMLAMA NEDİR?

Programlama, bilgisayara belirli bir problemi çözmesi için verilen talimatların bütünüdür. Bu talimatlar kesin, net ve adım adım olmalıdır. İnsanlar günlük hayatta soyut düşünebilir, ancak bilgisayarlar yalnızca açık ve net komutlarla çalışır. Programlama sayesinde insanlar bilgisayarlara kendi düşüncelerini aktarır ve bilgisayar bu talimatları işleyerek sonuç üretir.

Önemli Nokta: Programlama, bir problemi çözmek için gerekli adımların bilgisayar diline dönüştürülmesidir.

PROGRAMCININ AMACI

Bir programcının temel amacı, bilgisayara doğru, eksiksiz ve anlaşılır talimatlar vererek problemi çözmektir.

Programcı:

- Problemi analiz eder,
- Çözüm adımlarını belirler,
- Bu adımları bilgisayarın anlayacağı biçimde kodlar.

Programcının başarısı, yazdığı programın hem doğru sonuç vermesine hem de verimli çalışmasına bağlıdır.

Önemli Nokta: Programcı, bilgisayarı belirlenen hedefe yönlendiren kişidir.

ALGORİTMA NEDİR?

Algoritma, bir problemin çözümünü sağlayan sonlu sayıdaki adımların mantıksal sırasıdır. Başka bir ifadeyle, algoritma bir işin nasıl yapılacağını adım adım tanımlar.

Algoritmanın Temel Özellikleri:

- Sonluluk: Algoritma mutlaka belirli adımlardan sonra sonuca ulaşmalıdır.
- Kesinlik: Her adım açık ve net olmalıdır, belirsizlik içermez.
- Doğruluk: Algoritma doğru şekilde uygulandığında istenilen sonucu vermelidir.

Önemli Nokta: Algoritma olmadan program yazmak mümkün değildir.

GÜNLÜK HAYATTAN ALGORİTMA ÖRNEĞİ

?

ÖZET

- Programlama: Bilgisayara problemi çözmesi için verilen talimatlar bütünüdür.
- Programcının amacı: Problemi çözmek için doğru, net ve verimli adımlar oluşturmaktır.
- Algoritma: Sonlu, kesin ve doğru adımlardan oluşan çözüm yöntemidir.
- Günlük yaşamda da algoritmalar vardır (çay demleme, yemek tarifi, alışveriş listesi).

Önemli Nokta: Programlama öğrenmeye başlamadan önce algoritma kavramını anlamak zorunludur.

DONANIM VE YAZILIM NEDİR?

- **Donanım (Hardware):** Bilgisayarın fiziksel parçalarıdır. Örneğin; işlemci (CPU), bellek (RAM), sabit disk, klavye, fare ve ekran gibi elemanlar donanıma girer. Donanım olmadan bilgisayarın çalışması mümkün değildir.

- **Yazılım (Software):** Bilgisayara hangi işlemleri yapacağını söyleyen komutlar bütünüdür. Programlar, işletim sistemleri ve uygulamalar yazılımı oluşturur.

Önemli Nokta: Bilgisayar = Donanım + Yazılım.

DONANIM TÜRLERİ

- **Giriş Birimleri (Input):** Bilgisayara veri girilmesini sağlar. Örneğin; klavye, fare, mikrofon.
- **İşlem Birimleri (Process):** Veriyi işler. Temel bileşen CPU'dur (işlemci). RAM geçici bellektir ve hızlı veri erişimi sağlar.
- **Çıkış Birimleri (Output):** İşlenen bilgiyi kullanıcıya sunar. Örneğin; ekran, yazıcı, hoparlör.
- **Depolama Birimleri:** Verileri kalıcı olarak saklar. Sabit disk, SSD, USB bellek örnekleridir.

Önemli Nokta: Donanım bileşenleri birlikte çalışarak bilgisayarın işlevselliğini sağlar.

YAZILIM TÜRLERİ

- **Sistem Yazılımları:** Bilgisayarın temel işleyişini sağlayan yazılımlardır. İşletim sistemleri (Windows, Linux, macOS) bu gruba girer. Donanım ile kullanıcı arasındaki köprüyü kurar.
- **Uygulama Yazılımları:** Belirli bir işi yapmak için kullanılan yazılımlardır. Örneğin; Word, Excel, oyunlar, hesap makineleri.
- **Programlama Dilleri:** Yeni yazılımların geliştirilmesi için kullanılan dillerdir (C, Python, Java).

Önemli Nokta: Yazılım olmadan donanım tek başına işlevsizdir.

İŞLETİM SİSTEMİ İLE PROGRAM ARASINDAKİ FARK

- **İşletim Sistemi (OS):**
  - Bilgisayarın açılmasını ve donanımın kullanılmasını sağlar.
  - Donanım ve yazılım arasında arayüzdür.
  - Örnek: Windows, Linux, macOS.
- **Program (Uygulama Yazılımı):**
  - Belirli bir görevi yerine getirmek için çalıştırılan yazılımdır.
  - İşletim sistemi üzerinde çalışır.
  - Örnek: Word, tarayıcı, oyunlar.

Önemli Nokta: İşletim sistemi olmadan programlar çalışmaz.

ÖZET

- **Donanım:** Bilgisayarın fiziksel parçaları (CPU, RAM, giriş-çıkış birimleri).
- **Yazılım:** Donanımı çalıştıran ve kullanıcıya hizmet sunan komutlar bütünü.
- **Donanım türleri:** Giriş, işlem, çıkış ve depolama.
- **Yazılım türleri:** Sistem yazılımları, uygulama yazılımları, programlama dilleri.
- İşletim sistemi, donanım ile uygulamalar arasında köprü görevi görür.

Önemli Nokta: Donanım ve yazılım birlikte çalıştığında bilgisayar işlevsel hale gelir.

PROGRAMLAMA DİLİ NEDİR?

- Programlama dili, bilgisayara yapılacak işlemleri anlatmak için kullanılan yapay bir dildir. İnsanların düşüncelerini bilgisayara aktarabilmesi için özel sözdizimi (syntax) ve kuralları vardır.
  - İnsan dili → esnek ve belirsiz olabilir.
  - Bilgisayar dili → kesin ve net olmalıdır.
  - Programlama dili sayesinde algoritmalar bilgisayarın anlayacağı hale gelir.

Önemli Nokta: Programlama dilleri olmadan bilgisayarlar yalnızca 0 ve 1 ile çalışabilir.

MAKİNE DİLİ

- Bilgisayarların doğrudan anlayabildiği tek dildir.
- 0 ve 1'lerden oluşur (ikili sayı sistemi, binary).
- Her komut işlemciye doğrudan etki eder.
- Yazması ve okuması çok zordur, hataya açıktır.

```
10110000 01100001
10110000 01100010
```

Önemli Nokta: Makine dili hızlıdır ama insanlar için anlaşılması zordur.

ASSEMBLY DİLİ

- Makine diline çok yakın, düşük seviyeli programlama dilidir.
- Komutlar kısa kelimelerden oluşur (ör. MOV, ADD, SUB).
- Her işlemci için farklı assembly dili olabilir.
- Daha anlaşılırdır ancak taşınabilirlik (portability) azdır.

```
MOV AX, 1
MOV BX, 2
ADD AX, BX
```

Önemli Nokta: Assembly, makine diline göre daha anlaşılır ama yine de karmaşıktır.

YÜKSEK SEVİYELİ PROGRAMLAMA DİLLERİ

- İnsan diline daha yakın yapıya sahiptir.
- Öğrenmesi ve yazması daha kolaydır.
- Derleyici veya yorumlayıcı tarafından makine diline dönüştürülür.
- Taşınabilirlik yüksektir, aynı kod farklı bilgisayarlarda çalışabilir.

Önemli Nokta: Günümüzde en yaygın kullanılan diller yüksek seviyeli dillerdir.

DERLEYİCİ VE YORUMLAYICI

- Derleyici (Compiler): Programın tamamını makine diline çevirir ve çalıştırılabilir dosya üretir.
  - Örnek: C, C++.
- Yorumlayıcı (Interpreter): Programı satır satır çevirerek çalıştırır.
  - Örnek: Python.
- Karşılaştırma:
  - Derleyici → Hızlı çalışır ama hata bulmak zordur.
  - Yorumlayıcı → Daha yavaş ama hata ayıklamak kolaydır.

Önemli Nokta: Her dil bir derleyici veya yorumlayıcı aracılığıyla çalışır.

BASİT KOD ÖRNEKLERİ

- C DİLİNDE “MERHABA DÜNYA”

```c
#include <stdio.h>
int main() {
    printf("Merhaba Dünya");
    return 0;
}
```

- PYTHON DİLİNDE “MERHABA DÜNYA”

```python
print("Merhaba Dünya")
```

ÖZET

- Programlama dilleri → bilgisayara talimat vermek için kullanılır.
- Makine dili: 0 ve 1'lerden oluşur, zor anlaşılır.
- Assembly: Makineye yakın ama biraz daha okunabilir.
- Yüksek seviyeli diller: İnsan diline yakın, taşınabilir.
- Derleyici ve yorumlayıcı, yüksek seviyeli dilleri makine diline dönüştürür.

Önemli Nokta: Programlama dillerinin evrimi insanın bilgisayara daha kolay hükmetmesini sağlamıştır.

ALGORİTMA TASARIMININ ÖNEMİ

- Algoritma tasarımı, program yazmadan önce problemin çözümünü adım adım planlama sürecidir. İyi bir algoritma olmadan yazılan programlar hatalı, anlaşılmaz ve verimsiz olur. Algoritmalar, karmaşık problemleri küçük ve yönetilebilir parçalara ayırarak çözmeyi kolaylaştırır.

Önemli Nokta: Algoritma tasarımı, programlamanın temelini oluşturur.

PROBLEMİN KÜÇÜK PARÇALARA AYRILMASI

- Büyük problemler genellikle doğrudan çözülemez.
- Çözüm için problem alt adımlara ayrılır.
- Her alt adım ayrı ayrı çözüldüğünde, tüm problem de çözülmüş olur.

Örnek: Bir sınıftaki öğrencilerin not ortalamasını hesaplama:

- Notları oku,
- Topla,
- Öğrenci sayısına böl,
- Ortalamayı yazdır.

Önemli Nokta: Adım adım düşünmek, karmaşık problemleri basitleştirir.

ADIM ADIM DÜŞÜNME (STEPWISE REFINEMENT)

- Stepwise refinement (adım adım geliştirme), algoritmanın önce genel hatlarının belirlenmesi ve sonra detaylandırılması yöntemidir.
  - Önce problemi kabaca tanımla,
  - Daha sonra her adımı detaylandır.
- Örnek:
  - Genel: "İki sayıyı topla ve sonucu göster."
  - Detaylı: "1. Kullanıcıdan iki sayı al. 2. Topla. 3. Sonucu ekrana yazdır."

Önemli Nokta: Küçük adımlar hataları azaltır ve algoritmayı daha anlaşılır kılar.

DOĞRULUK VE VERİMLİLİK

- Doğruluk: Algoritma doğru sonucu üretmelidir. Yanlış adımlar doğru sonucu engeller.
- Verimlilik: Algoritma gereksiz adımlardan arındırılmış, en kısa sürede doğru sonuca ulaştırmalıdır.
- Algoritmalar hem doğru hem verimli olmalıdır.

Örnek:

- Doğru ama verimsiz → Binlerce gereksiz hesaplama yapan yöntem.
- Doğru ve verimli → Daha az adımda aynı sonucu bulan yöntem.

Önemli Nokta: İyi bir algoritma hem doğru hem verimlidir.

ALGORİTMA TASARIMI ADIMLARI

1. Problemin tanımlanması
2. Girdilerin ve çıktının belirlenmesi
3. Çözüm adımlarının planlanması
4. Adımların mantıksal sıraya konması
5. Gereksiz işlemlerin ayıklanması
6. Algoritmanın test edilmesi

Önemli Nokta: Algoritma tasarımı, kodlamadan önce yapılmalıdır.

ÖZET

- Algoritma tasarımı → programlamanın temel aşamasıdır.
- Problemler küçük parçalara bölünerek çözülür.
- Adım adım düşünme (stepwise refinement) yöntemi ile algoritmalar geliştirilir.
- Doğruluk ve verimlilik algoritmanın en önemli iki özelliğidir.
- Algoritma yazmadan kod yazmak büyük hatalara yol açar.

PROGRAMLAMA SÜRECİNE GİRİŞ

- Program geliştirme, bir problemin bilgisayar tarafından çözülebilmesi için izlenen aşamalar bütünüdür. Başarılı bir program, yalnızca kod yazmakla değil; problem analizi, algoritma tasarımı, test ve hata ayıklama gibi adımların dikkatle uygulanmasıyla ortaya çıkar.

Önemli Nokta: Programlama bir süreçtir, yalnızca kod yazmak değildir.

PROGRAMLAMA SÜRECİNİN AŞAMALARI

1. **Problemin Tanımlanması:** Çözülecek sorunun anlaşılması.
2. **Algoritma Geliştirme:** Çözüm adımlarının belirlenmesi.
3. **Akış Diyagramı Hazırlama:** Çözümün görselleştirilmesi.
4. **Kodlama:** Algoritmanın programlama diliyle yazılması.
5. **Derleme ve Çalıştırma:** Programın makine tarafından çalıştırılabilir hale getirilmesi.
6. **Test ve Hata Ayıklama (Debug):** Hataların bulunması ve düzeltilmesi.
7. **Sonuçların Değerlendirilmesi:** Programın amacına uygun çalışıp çalışmadığının kontrolü.

Önemli Nokta: Her adım birbirini tamamlar, eksik bir adım süreci aksatır.

PROBLEMİN TANIMLANMASI

- Bir program yazmaya başlamadan önce problem net şekilde tanımlanmalıdır.
- Problemin amacı belirlenir.
- Girdiler ve çıktılar tanımlanır.
- Çözümün sınırları belirlenir.

Örnek: "Bir sınıftaki öğrencilerin ortalamasını bulan program"

→ Girdi: notlar, Çıktı: ortalama.

Önemli Nokta: Doğru tanımlanmayan problem çözülemez.

AKIŞ DİYAGRAMI VE KODLAMA

- **Akış Diyagramı:** Algoritmanın görsel ifadesidir. Karar kutuları, işlem kutuları ve akış okları kullanılır.
- **Kodlama:** Akış diyagramındaki adımlar, seçilen programlama diline uygun olarak yazılır.

Örnek:

- Akış diyagramı:
  - "Başla → Sayı al → Sayı çift mi? → Evet → Çift yazdır → Hayır → Tek yazdır → Bitir."
- Kodlama:
  - Python'da if number % 2 == 0: print("Çift") else: print("Tek").

Önemli Nokta: Akış diyagramı kodlamaya rehberlik eder.

DERLEME (COMPILE) VE ÇALIŞTIRMA

- **Derleme (Compile):** Programın yüksek seviyeli dilden makine diline çevrilmesidir. Bu aşamada sözdizimi (syntax) hataları bulunur.
- **Çalıştırma (Run):** Derlenen programın bilgisayar tarafından yürütülmesidir.

Önemli Nokta: Derleme aşamasında bulunan hatalara derleme hatası, çalıştırma sırasında çıkan hatalara çalışma zamanı hatası denir.

TEST VE HATA AYIKLAMA (DEBUG)

- **Test:** Programın farklı girdilerle denenmesi.
- **Hata Ayıklama (Debug):** Programın hatalarının bulunup düzeltilmesi.
- Hatalar:
  - Sözdizimi hataları (syntax errors)
  - Mantık hataları (logic errors)
  - Çalışma zamanı hataları (runtime errors)

Önemli Nokta: Hata ayıklama programlama sürecinin en çok zaman alan aşamasıdır.

ÖZET

- Programlama süreci = problem → algoritma → akış diyagramı → kodlama → derleme → test → sonuç.
- Problemin doğru tanımlanması en kritik adımdır.
- Akış diyagramı programı görsel olarak planlamayı sağlar.
- Derleme hataları ve çalışma zamanı hataları farklıdır.
- Test ve hata ayıklama süreci, programın güvenilirliğini artırır.

PROGRAMLAMANIN HAYATTAKİ ROLÜ

- Programlama yalnızca bilgisayar mühendislerine özgü değildir. Günlük hayatta kullandığımız birçok araç, cihaz ve hizmet programlama sayesinde çalışır. Telefonlarımızdaki uygulamalar, internet siteleri, bankacılık işlemleri, sağlık cihazları, oyunlar ve hatta ulaşım sistemleri programlarla yönetilmektedir.

Önemli Nokta: Programlama, modern yaşamın görünmez altyapısını oluşturur.

MÜHENDİSLİK ALANINDA PROGRAMLAMA

- Elektrik ve elektronik mühendisliği: Devre simülasyonları, otomasyon sistemleri.
- İnşaat mühendisliği: Yapıların bilgisayar ortamında modellenmesi.
- Makine mühendisliği: 3B tasarımlar, üretim planlaması.

Önemli Nokta: Mühendislikte programlama verimliliği ve doğruluğu artırır.

SAĞLIK ALANINDA PROGRAMLAMA

- MR, tomografi ve ultrason cihazlarının çalışması.
- Hasta kayıt ve takip sistemleri.
- Yapay zekâ ile hastalık teşhisi.
- İlaç geliştirme süreçlerinde veri analizi.

Önemli Nokta: Sağlık alanındaki birçok yenilik, yazılım ve programlama sayesinde gerçekleşmektedir.

FİNANS VE İŞ DÜNYASINDA PROGRAMLAMA

- Bankacılık işlemleri (ATM, internet bankacılığı).
- E-ticaret siteleri ve mobil ödeme sistemleri.
- Büyük veri analizi ile yatırım kararları.
- Otomatik işlem yapan algoritmik ticaret sistemleri.

Önemli Nokta: Finans dünyasında güvenlik ve hız programlamayla sağlanır.

EĞLENCE VE OYUN SEKTÖRÜNDE PROGRAMLAMA

- Bilgisayar ve mobil oyunları.
- Animasyon filmleri.
- Sanal gerçeklik (VR) ve artırılmış gerçeklik (AR) uygulamaları.
- Müzik düzenleme ve video montaj yazılımları.

Önemli Nokta: Eğlence sektörünün gelişimi programlama ve yazılım teknolojilerine bağlıdır.