BİLİŞİM TEKNOLOJİLERİ VE YAZILIM DERSİ 9.SINIF 2. DERS NOTLARI

2. ÜNİTE ALGORİTMA İLE PROBLEM ÇÖZME VE AKIŞ DİYAGRAMI

Bir problemi çözmeden önce, bu problemin neden ortaya çıktığını bilmemiz gerekiyor. Eğer problemin nedenlerini bulursak, çözüm yolunu da daha kolay bulabiliriz.

Bu yaklaşım, algoritma yazarken de aynıdır: Problemi tam anlamıyla çözmek için sorunun kökenine inmek, yani nedenini bulmak çok önemli.

Problem çözme süreci üzerine geliştirilmiş kuramlar ve yaklaşımlar

1. Polya'nın Dört Adımlı Problem Çözme Yöntemi:

Matematikçi George Polya, problem çözme sürecini basitleştirmek için dört adım önermiştir:

1. Problemi Anlamak
2. Bir Plan Geliştirmek
3. Planı Uygulamak
4. Sonucu Gözden Geçirmek

2. Algoritmik Yaklaşım:

Algoritmalar, bir problemi çözmek için takip edilmesi gereken adım adım talimatlardır. Bu yaklaşım özellikle bilgisayar bilimlerinde yaygındır.

Algoritmik problem çözme şu adımları içerir:

•Girdileri tanımlama: Problemi çözmek için gerekli tüm bilgileri toplamak.
•Adımları belirleme: Adım adım izlenecek çözüm yolunu belirleme.
•Sonuç alma: Algoritmayı uygulayarak sonuca ulaşma.

3. Heuristik (Deneme-Yanılma) Yöntem:

Bu yöntemde insanlar, problemi çözmek için daha önce deneyip başarılı oldukları stratejileri kullanırlar.
Ancak bu yaklaşım her zaman kesin bir çözüm sunmaz. Heuristik yöntem, bazen çözüm için birkaç deneme ve yanılmasüreci gerektirebilir.

BIR PROBLEMIN ÇÖZÜMÜ IÇIN KULLANILABILECEK TEMEL YÖNTEM VE TEKNIKLEr

BÖL VE FETHET (DİVİDE AND CONQUER): Büyük bir problemi daha küçük parçalara bölmek ve her parçayı ayrı ayrı çözmek.

BENZER PROBLEMLERİ KULLANMA (ANALOGİLERDEN FAYDALANMA): Daha önce çözülmüş benzer bir problemi kullanarak yeni bir probleme yaklaşmak.

GÖRSEL DÜŞÜNME (DİYAGRAMLAR VE HARİTALAR KULLANMA): Problemi görselleştirerek çözüm yollarını daha açık hale getirmek. Diyagramlar, akış şemaları, zihin haritaları gibi görsel araçlar kullanılarak problem ve çözüm yolları organize edilir.

GERİYE DOĞRU ÇALIŞMA (TERSİNE MÜHENDİSLİK): Problemin çözümünü son aşamadan başlayarak geriye doğru gitmek.

BEYİN FIRTINASI: Farklı kişiler farklı çözüm önerileri sunar ve en iyi çözüm yolu seçilir.

PROBLEMIN GIRDI, ÇIKTI VE IŞLEM AŞAMALARI

■ Girdi (Input):

• Tanım: Girdi, bir problemin başında sisteme girilen veriler ya da işlemin başlaması içingerekli olan malzemelerdir.
• Örnekler: Bir hesap makinesine 5 + 3 yazdığımızda, 5 ve 3 bizim girdimizdir.
• Bir pizzacıya sipariş verirken seçtiğimiz pizza türü ve adresimiz de girdi olarak kabul
Edilebilir.

■ Çıktı (Output):

• Tanım: Bir sistemin veya işlemin sonunda elde ettiğimiz sonuçtur. Yani, girdileri işledikten sonra aldığımız sonuç ya da bilgi çıktıdır.
• Örnekler:Hesap makinesi, 5 + 3 işlemi sonrası bize 8 sonucunu verir, bu çıktıdır.
• Pizzacı siparişin sonunda pizzayı teslim eder, bu da işlemin çıktısıdır.

Örnek problem durumlarının girdi ve çıktıları

■ Örnek : Bir Metin Belgesini Yazdırmak

• Problem: Bilgisayarda yazdığımız bir metni yazıcıdan yazdırmak istiyoruz.
• Girdiler: Bilgisayardaki metin dosyası, yazıcı, kağıt.
• İşlem: Yazdırma komutu vermek ve yazıcının belgeyi yazdırması.
• Çıktı: Kağıda basılmış metin.

ALGORİTMA

Algoritma, bir problemin çözümü için izlenen adım adım işlemlerin tanımıdır. Bir işi yapmak için gereken talimatlar dizisidir.

Günlük rutin işlerinizin mantığı ve sırası değiştiğinde sonuçlar da değişir
Örnek: Diş fırçalamak

■ Diş macunu sürmeden fırçalamaya başlarsak, dişlerimiz temizlenmez.

■ Önce macun sürülmeli, sonra fırçalanmalı.
Örnek: Yemek yapmak
Yemeği pişirmeden malzemeleri tabağa koyarsak çiğ olur ve yenmez.
Önce malzemeler hazırlanır, sonra pişirilir ve en son servis edilir.

Pasta Yapma Algoritması

■ Algoritma:
ADIM 1. Gerekli malzemeleri topla (un, şeker, yumurta, süt, yağ, kabartma tozu, vanilin).
ADIM 2. Fırını 180°C'de önceden ısıt.
ADIM 3. Büyük bir kapta yumurtaları ve şekeri çırp.
ADIM 4. Süt ve yağı ekle, karıştır.
ADIM 5. Ayrı bir kapta un, kabartma tozu ve vanilini karıştır.
ADIM 6. Kuru malzemeleri sıvı karışıma ekle ve iyice karıştır.
ADIM 7. Karışımı yağlanmış kek kalıbına dök.
ADIM 8. Önceden ısıtılmış fırına yerleştir.
ADIM 9. 30-35 dakika pişir. (Pişip pişmediğini kontrol etmek için bir kürdan batır.)
ADIM 10.Pişen pastayı fırından çıkar ve soğumaya bırak.
ADIM 11.İsteğe bağlı olarak üzerine krema veya meyve ekle ve servis et.

Algoritma Kelimesinin Kökeni

• Algoritma Terimi: El-Harezmi'nin adı, Latince'de "Algoritmi" veya "Algoritmus" olarak geçmektedir.
• Bu terim, zamanla "algoritma" biçimine evrilmiştir.
• Günümüzde "algoritma", belirli bir problemi çözmek için izlenilen sistematik adım dizisi anlamına gelir.

EN DOĞRU ALGORİTMA İÇİN KULLANLAN KRİTERLER;

1. **Sorun Tespiti ve Çözümleme:**
2. **Tutarlılık ve Tekrar Edilebilirlik:**
3. **Yanlış veya Eksik Adım Tanımlama:**
4. **Yanlış Mantık Yapısı:**
5. **Yanlış Girdi Tanımlama:**
6. **Sınır Durumlarını Göz Ardı Etme:**
7. **Optimizasyon Eksikliği:** .
8. **Hatalı Veri Yapıları Kullanma: Döngüler:**
9. **Test Eksikliği:**

ALGORITMA ÖRNEKLERI:

1. Çay demleme algoritması: **Adım1 : başla** **Adım2: suyu ısıt** **Adım3: çayı demle** **Adım4: demlenmesini bekle** **Adım5: çay hazır** **Adım6: bitir**	**2. Ehliyet alabilme algoritması:** **Adım1: başla** **Adım2: yaşınızı girin** **Adım 3: yaş eşit veya büyük mü 18'den evet ise adım4 e git; hayır ise adım 5 e git** **Adım4:"ehliyet alabilirsin" yaz ve adım 6 a git** **Adım5:"ehliyet alamazsın" yaz vr adım 6 a git** **Adım6: bitir**
3. ATM'den para çekme işlemi Algoritma Adımları: **Adım 1.Başla** **Adım 2.Kartı yerleştir ve PIN kodunu gir** **Adım 3.PIN doğru mu?** **Adım 4. Evet: Para çekme işlemi başlat** **Adım 5. Hayır: Tekrar PIN gir veya işlemi sonlandır** **Adım 6.Çekilecek miktarı gir** **Adım 7.Hesapta yeterli bakiye var mı?** **Adım 8. Evet: Parayı ver, bakiye düş** **Adım 9. Hayır: Yetersiz bakiye uyarısı ver** **Adım 10.Kartı iade et** **Adım 11.Bitir**	**4. Sayı tahmin oyunu algoritması:** **Adım1:başla** **Adım2: bir sayı tut** **Adım3: tahmin et** **Adım4: tahmin edilen sayı eşit mi tutulan sayıya evet ise adım5 e git; hayır ise adım 6 a git.** **Adım5: "doğru tahmin" yaz ve adım7 e git** **Adım6: "yanlış tahmin" yaz ve adım 3 e git** **Adım7: bitir.**

PROGRAMLAMA DİLİ NEDİR?

Tüm yazılımlar programlama dilleriyle üretilir.

• Cep telefonunuzdaki uygulamalar, masaüstü uygulamaları, oyun ve ağ uygulamaları üretmek için programlama diline ihtiyacınız vardır.
• Günümüz itibariyle dünya üzerindeki tüm programlama dillerinin toplam sayısının 750'nin üstünde olduğu tahmin edilmektedir.

Programlama Neden Önemlidir?

1. Günlük Hayatı Kolaylaştırır
2. Problem Çözme Becerisi Kazandırır
3. Yaratıcılığı Teşvik Eder
4. Geleceğin Meslekleri için Temeldir
5. İşbirliği ve Paylaşımı Destekler

BAZI PROGRAMLAMA DİLLERİ: Pascal, Basic, C, C#, C++, Java, JavaScript, Cobol, Perl, PHP, Python, Ada, Fortran, Delphi ve Swift'tir.

AKIŞ ŞEMASI

Bir sürecin adımlarını **görsel** ya da **sembolik** olarak gösterir.

Akış diyagramının kullanım amaçları ve avantajları:

• Problemi Görselleştirir
• Mantıksal Düşünme Becerisini Geliştirir
• Hataları Önceden Görme İmkanı Sunar
• Çözüm Sürecini Optimize Etme Fırsatı Verir
• İletişimi Kolaylaştırır
• Adım Adım Kılavuz Görevi Görür
• Değişiklik Yapmayı Kolaylaştırır

TEMEL AKIŞ DIYAGRAMI ŞEKILLERI VE ELEMANLARI

Simge	İşlev
	Başla/Bitir
	Giriş
	Atama/İşlem
	Denetim (Karar)
	Çıkış
	Döngü
	Akış Yönü
	Bağlaç
	Önceden Tanımlı İşlem/Fonksiyon

Akış diyagramının elektronik ortamdaki çizimi için kullanılan kelime işlemci programları veya diğer çizim programları: Microsoft Word, Flowchart.com, Google Docs, Microsoft PowerPoint, Google Slides, Microsoft VisioL, Lucidchart, Diagrams.net (Draw.io), Google Drawings, Mermaid.js VB.

2.9. Akış diyagramının kullanım amaçları ve avantajları

1. Doğrusal Akış Diyagramları

Doğrusal akış diyagramları, basit ve sıralı bir iş akışını gösterir. Adımların birbirini takip ettiği ve herhangi bir karar veya döngü içermeyen işlemleri temsil eder. Genellikle tek bir başlangıç ve bitiş noktası vardır.

ÖRNEK

2. Mantıksal Akış Diyagramları

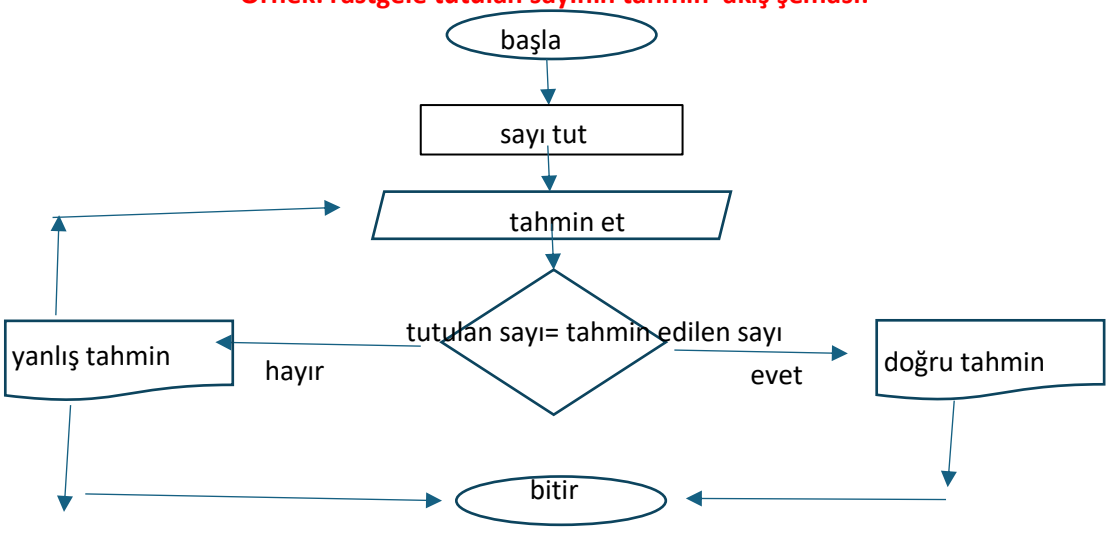
Mantıksal akış diyagramlarında, karar noktaları bulunur ve bu noktalar belirli koşullara göre farklı akışlar sağlar. Bu tür akış diyagramları, karar yapıları içerir (örneğin, "evet" veya "hayır" gibi durumlar).

ÖRNEK: 18 VE ÜSTÜ OLAN KİŞİLERİN EHLİYET ALABİLME ALGORİTMASI

3. Döngü İçeren Akış Diyagramları

Döngü içeren akış diyagramları, belirli bir işlem tekrar tekrar yapılırken kullanılır. Döngülerin bulunduğu yerlerde bir koşul karşılanana kadar işlemler devam eder.

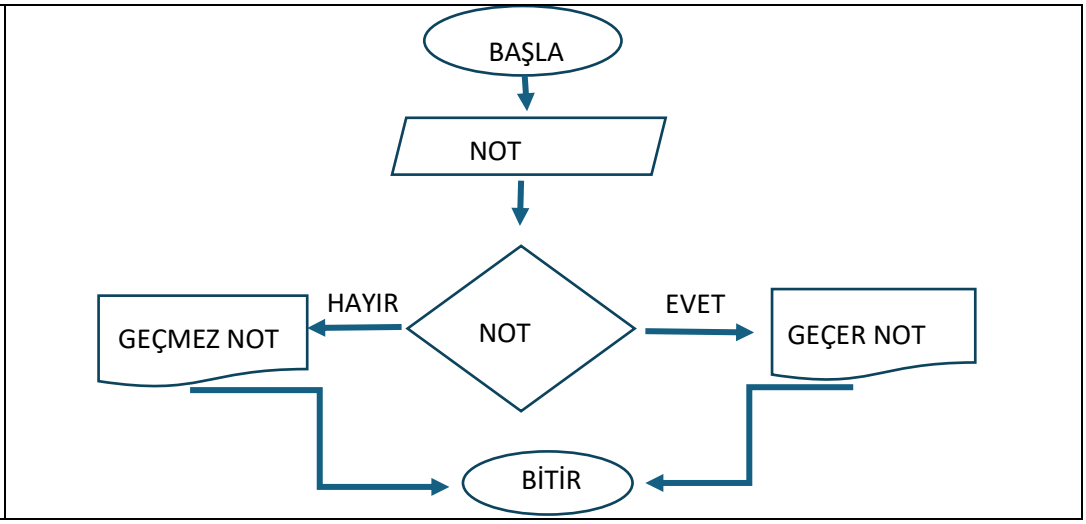
Örnek: rastgele tutulan sayının tahmin akış şeması:



ÖRNEK ALGORITMA VE AKIŞ ŞEMALARI:

ÖRNEK1: Bir öğrencinin sınavdan geçti mi kaldı mı olduğunun hesaplanması. Şartlar ,Öğrencinin sınav notu 50 veya üzeriyse geçti, 50'nin altındaysa kaldı.

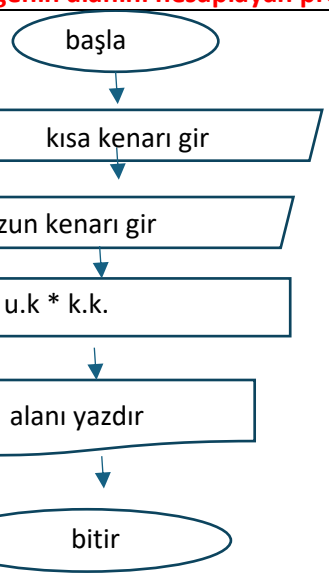
Algoritma Adımları:
1.Başla
2.Öğrencinin sınav notunu al
3.Eğer not 50 veya üzeriyse, "Geçti" yazdır
4.Eğer not 50'den küçükse, "Kaldı" yazdır
5.Bitir



Örnek:2 kullanıcının girdiği kısa kenar ve uzun kenara göre dikdörtgenin alanını hesaplayan programın;

ALGORİTMA:
ADIM1 BAŞLA
ADIM2 kısa kenarı gir
ADIM3 uzun kenarı gir
ADIM4 k.k ve u.k. çarp
ADIM5 sonucu yaz
Adım6: bitir

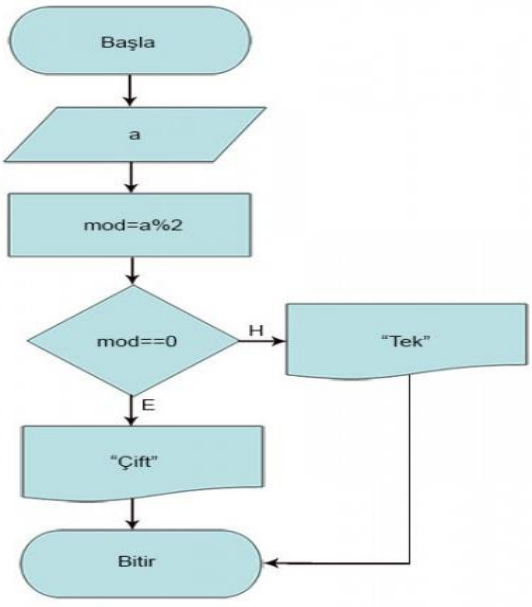
AKIŞ ŞEMASI:



Örnek3: girilen tam sayının çift mi tek mi olduğunu yazdırma algoritma ve akış şeması:

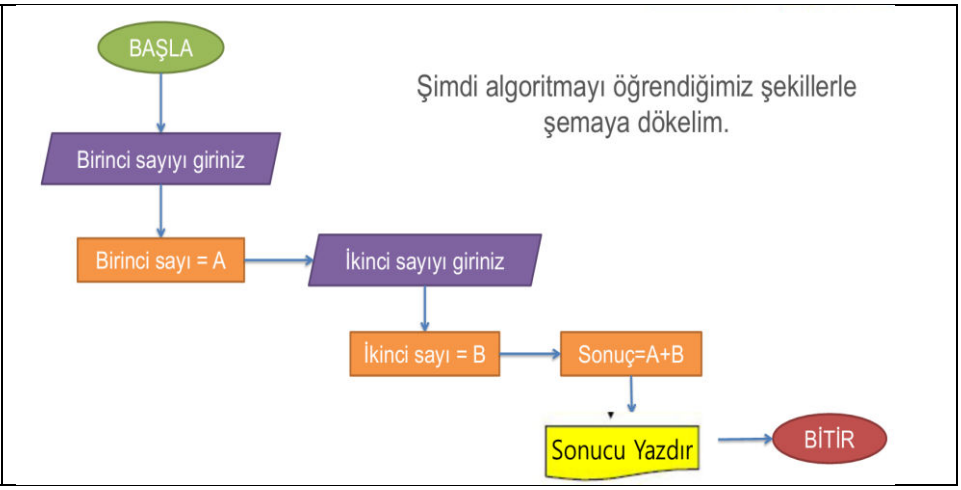
ALGORİTMASI

1-Başla
2-Herhangi bir sayı giriniz.(x)
3-Eğer x mod 2 =0 ise ekrana çift yaz değilse tek yaz.
4-Dur



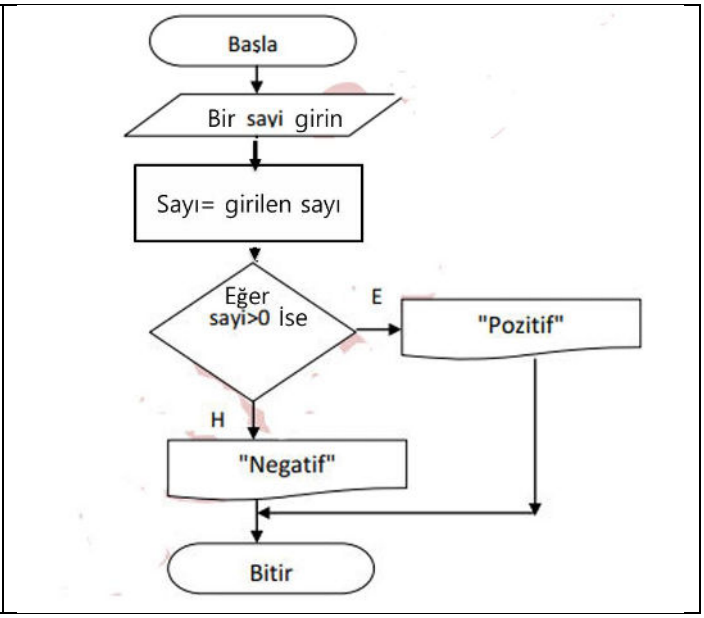
Örnek4. Klavyeden girilen iki sayıyı toplayıp ekrana yazdıran programın akış şemasını çizeceğiz. Önce algoritmasını yazalım.

Adım 1: Başla
Adım 2: İlk sayıyı gir.
Adım 3: A = İlk sayı Adım 4: İkinci sayıyı gir.
Adım 5: B = İkinci sayı
Adım 6: toplam=A+B
Adım 7: toplamı ekranda göster.
Adım 8: Bitir.



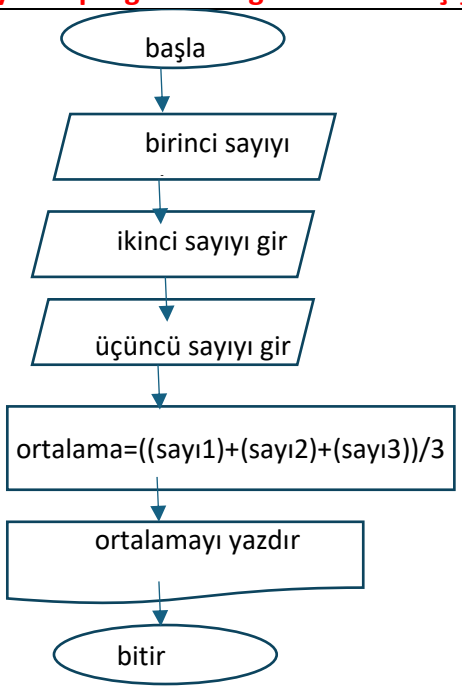
Örnek5: Girilen sayının negatif olup olmadığını bulan programın algoritma ve akış şemasını yazınız. Negatif ise ekrana ''negatif'', pozitifse "pozitif" yazmalı.

Algoritma
1. Başla
2. Sayıyı gir
3. sayı=girilen sayı
4. Eğer sayı<0 ise 6. Adıma git
5. Pozitif yaz 7. adıma git
6. Negatif Yaz
7. Bitir



Örnek 6: Girilen 3 sayının ortalamasını bulup ekrana yazan programın algoritma ve akış şemasını yazınız.

Algoritma
1. Başla
2. Birinci sayıyı gir
3. Girilen sayı= sayi1
4. İkinci sayıyı gir
5. Girilen sayı=sayi2
6. Üçüncü sayıyı gir
7. Girilen sayı=sayi3
8. ortalama= ((sayi1+sayi2+sayi3) /3)
9. Yaz ortalama
10. Bitir



3. ÜNİTE
PROGRAMLAMANIN TEMEL KAVRAMLARI
Programlama Temelleri

Programlamada sıkça kullanılan temel kavramlar:
- Operatörler - Değişkenler- Koşullar- Döngüler
• Örnek: x = 5, if, while

Veri: Bilgisayarların kullanmak üzere depoladığı her şeye veri denir. Örnekler: Boyunuz 1.70 cm uzunluğundadır. Saçınız sarı renklidir. Burcunuz Koç'tur. Hava yağmurludur. Hava 28 derecedir.

Sabit Veri: Problemin sürecinde asla değişmeyen değerlerdir. Ör: Müşterinin adı soyadı

Değişken Veri: Program sürecinde değişebilen değerlerdir. Ör: Müşterinin aylık alışveriş miktarı

Veri Türleri:

● **Sayısal Veri:** Yaş, uzaklık, nüfus, ücret, yarıçap gibi hesaplama sürecinde gerekli olan rakamsal verilerdir. Pozitif veya negatif olabilirler. Tam sayı veya reel sayı olabilirler. Önemli not: Posta kodu, telefon numarası veya öğrenci numarası gibi değerler de sayısal rakamlar içerirler ama bunlarla matematiksel işlemler yapılmadığı için sayısal veri değildirler.

● **KarakterVeri (string):** Tüm karakterleri içerir ancak tırnak içerisinde belirtilmesi gereken verilerdir. Harfleri ("a"..."z", "A"..."Z") , Özel karakterleri ("#", "&", "*","."," !") , Tüm tek haneli sayıları ("0".. "9") kapsarlar. Önemli Not: Karakter verilerde büyük ve küçük harfler birbirlerinden farklıdır. Örneğin b harfi B harfinden farklıdır. Önemli Not: Karakter verileri ile toplama yapılamaz. Ama birleştirme yapılabilir. Örnekler: '6'+'6'="66" 'uyur'+''gezer''= 'uyurgezer' Örnek soru: 12+'pc'=? Bu işlemin sonucu hatalıdır, çünkü sayısal veri ile karakter veri toplanmaz veya birleşmez.

● **MantıksalVeri:** Mantıksal veri yalnızca iki kelime barındırır: DOĞRU VE YANLIŞ. (True, False) Yani cevabı doğru veya yanlış olan tüm sorular mantıksal veri sınıfına girer. Örnek: Evli misin? Araban var mı? Öğrenci misin?