3. PROBLEM ÇÖZME SÜRECİ

Bu bölümde;

- ✓ Problemlerin keşifsel ve algoritmik çözümleri arasındaki farkları belirtebilecek,
- ✓ Algoritmik bir çözümü olan problemleri çözmek için gereken 6 problem çözme adımını açıklayıp listeleyebilecek,
- ✓ Bir problemi çözebilmek için 6 problem çözme adımını kullanacak,
- ✓ Değişken ve sabit arasındaki farkı algılayacak,
- ✓ Karakter, sayısal ve mantıksal veri türlerini anlayacak,
- ✓ Operatör, işlemci ve sonucu ayırt edecek,
- ✓ Fonksiyon tanımlayıp kullanacak,
- ✓ İlişkisel ve mantıksal operatörleri kullanacak,
- ✓ İşlem önceliğini kavrayacak,
- ✓ İfade ve eşitlikleri kullanarak işlem yapacaksınız.

3.1. Problem Çözme Teknikleri

3.1.1. Her Zaman Bir Planınız Olsun

- Belirsiz bir durumu yaşamak yerine her zaman bir planınız olmalıdır.
- Belki oluşturduğunuz çözüm planı ilk denemelerde sonuç vermeyecek ama her seferinde sizi çözüme biraz daha yaklaştıracak ipuçları elde etmenizi sağlayacaktır.
- Küçük hedeflerden bile oluşan bir plan yaparsanız çözüme yönelik adımlar attığınızı ve zamanı etkili biçimde kullandığınızı göreceksiniz.

3.1. Problem Çözme Teknikleri

3.1.2. Problemi Tekrar İfade Edin

- Önceki problemlerde de gördüğümüz üzere bazen problemi tekrar ifade etmek, göremediğimiz bir ayrıntıyı görmemizi ya da problemi daha kolay çözmek adına bir ipucu yakalamamızı sağlayabilir.

3.1.3. Problemi Küçük Parçalara Ayırın

- Verilen problemi adımlara ya da bölümlere ayırmak, çözümü kolaylaştırır.

3.1. Problem Çözme Teknikleri

3.1.4. Önce Bildiklerinizden Yola Çıkın

- Programlama yaparken öncelikle bildiklerimiz ile başlamalı ve sonra yeni çözümler arayışına girmeliyiz.
- Problemi küçük parçalara bölerek çözebildiğiniz parçadan başlayınız.
- Bu parçaları çözerken diğer parçalarla ilgili olarak aklınıza yeni fikirler geldiğini ve aynı zamanda kendinize olan güvenin arttığını göreceksiniz.

3.1.5. Problemi Basitleştirin

- Çözmekte zorlandığınız bir problemle karşılaşırsanız problemin kapsamını daraltmayı deneyebilirsiniz
- Temel amacınız problemi basitçe ifade etmeye çalışmak olmalıdır.

3.1. Problem Çözme Teknikleri

3.1.6. Benzerlikleri Arayın

- Burada ele aldığımız benzerlik kavramı, çözülmesi istenen problemle önceden çözülen problem arasındaki olası örtüşme ya da yeni çözüme ilham verme olarak tanımlanabilir.
- Programlamaya yeni başlayanlar için bir problemi çözerken hazır yazılmış bir kodu bulmak ve onu güncelleyerek problemi çözmeye çalışmak son derece yanlıştır.
- Bir çözümü kendiniz üretmezseniz tamamen anlayamaz ve içselleştiremezsiniz

3.1.7. Deneme Yapın

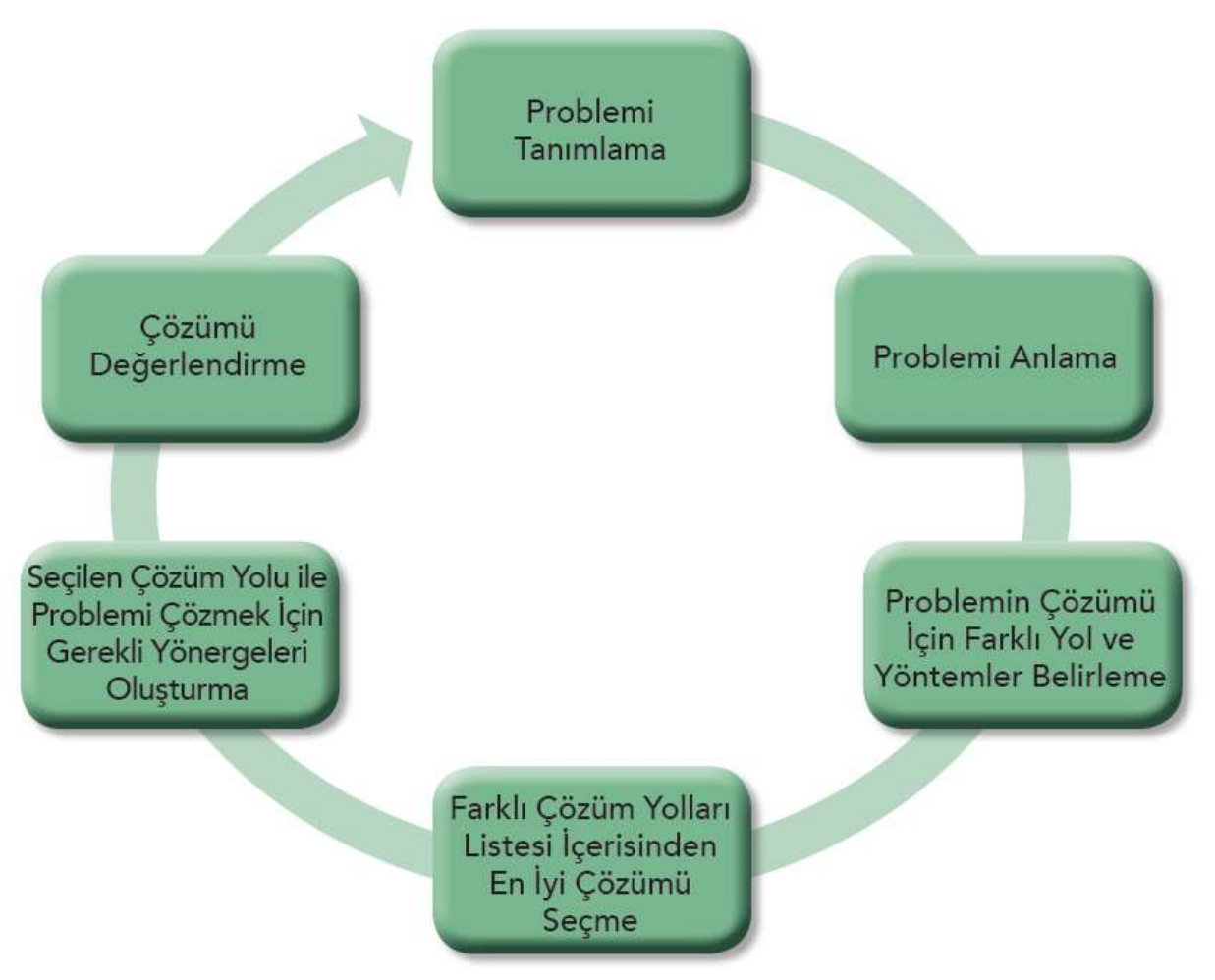
- Bazen bir problemi çözmenin en kolay yolu denemek ve sonuçlarını gözlemlemektir.

3.1.8. Asla Vazgeçmeyin

- Kararlılık, güven ve istek kaybolduğu zaman açık düşünemezsiniz, işlemler olması gerektiğinden uzun sürer ve gittikçe zorlaşır.

3.1. Problem Çözme Teknikleri

- 3.2. Problem Çözme Adımları



3.1. Problem Çözme Teknikleri

1. Problemi Tanımlama:	• Problemi çözmeye başlamadan önce problemin açık, anlaşılır ve çok doğru bir şekilde tanımlanmış olması gerekir. • Problemin ne olduğunu bilemezseniz onu çözemezsiniz.
2. Problemi Anlama:	• Çözüme doğru yol almadan önce problemi çok iyi anladığınızdan emin olmanız gerekir. • Problemi anlamak, problemi yarı yarıya çözmek demektir.
3. Problemin Çözümü İçin Farklı Yol ve Yöntemler Belirleme:	• Problemin çözümü için olabildiğince farklı yol ve yöntem belirlemeli ve bu listenin, tüm olasılıkları içerdiğinden emin olmalısınız.

3.1. Problem Çözme Teknikleri

4. Farklı Çözüm Yolları Listesi İçerisinden En İyi Çözümü Seçme:	• Bu adımda her bir çözümün olumlu ve olumsuz yönlerini ortaya koymalısınız. • Bu nedenle değerlendirme yapabilmek için ölçütler oluşturmalısınız.
5. Seçilen Çözüm Yolu ile Problemi Çözmek İçin Gerekli Yönergeleri Oluşturma:	• Bu adımda numaralandırılmış ve adım adım yönergeler oluşturmanız gerekir.
6. Çözümü Değerlendirme:	• Çözümü test etmek ya da değerlendirmek, sonucun doğruluğunu kontrol etmek anlamına gelir. • Sonucun doğru olması ve problemi olan bireyin beklentilerini karşılama düzeyi önemlidir.

3.1. Problem Çözme Teknikleri

- **3.3. Problem Türleri**
- Kek yapmak ya da araba kullanmak gibi problemleri çözmek için bir dizi eylem gerekir.
- Adım adım yönergelere dayalı olan bu çözümlere "**algoritmik çözümler**" denir.

- En iyi yolu seçtikten sonra sonuca, ilgili adımları izleyerek ulaşılır. Bu adımlardan oluşan yapıya "**algoritma**" denir.

- Doğrudan işlem adımları ile ulaşılamayan sonuçlara "**keşfe dayalı çözümler**" denir.

3.1. Problem Çözme Teknikleri

- **3.4. Bilgisayarlar ile Problem Çözme**
- Bu ders kapsamında "çözüm" demek problem çözme surecinin 5. adımında yer alan işlem adımları ya da yönergeler anlamına gelmektedir.

5. Seçilen Çözüm Yolu ile Problemi Çözmek İçin Gerekli Yönergeleri Oluşturma:	• Bu adımda numaralandırılmış ve adım adım yönergeler oluşturmanız gerekir.

3.1. Problem Çözme Teknikleri

3.4. Bilgisayarlar ile Problem Çözme

- **"Sonuç"** demek, çıktı ya da tamamlanmış bilgisayar destekli yanıt demektir.

- **"Program"** ise herhangi bir bilgisayar dilinde kodlanmış, çözümü oluşturan işlem adımlarının tamamını ifade etmektedir.
- Bilgisayarlar, zor ve zaman alıcı olabilen algoritmik çözümler ile ilgilenmek üzere tasarlanmıştır.
- İnsanlar, keşifsel çözümleri bulma konusunda daha iyidirler ancak bilgisayarların çözebildiği ileri düzey hesaplama ve karmaşık problemleri çözme konusunda bilgisayarların hızlarına ulaşamazlar.

3.1. Problem Çözme Teknikleri

3.4. Bilgisayarlar ile Problem Çözme

- Keşifsel problem türleri ile ilgilenen bilgisayar dalına "**yapay zeka**" adı verilmektedir.
- Yapay zeka uygulamaları, bilgisayarlara mevcut bilgileri kullanarak yeni bilgiler inşa etmesini sağlamaktadır.
- Böylece bilgisayarın problem çözme becerileri insanların yeteneklerine daha çok benzemektedir.
- Bilgisayarlar insanlar gibi düşünmeye başlayana kadar daha çok algoritmik problemlerin çözüm süreçlerinde kullanılacaktır.
- *Bu nedenle bu derste ağırlıklı olarak algoritmik çözümler üzerinde durulacaktır.*

3.1. Problem Çözme Teknikleri

3.5. Problem Çözme Kavramları

- Bilgisayara ilişkin temel kavramlar ve belirtilen türdeki problemleri çözmek için kullanılan ifade ve eşitlikler anlatılmaktadır.
- "**Sabit**" ve "**değişken**" önemli iki kavramdır.
- Programcı işlenmemiş halde veriyi alır, işlenmiş hale yani bilgiye dönüştürür.

3.1. Problem Çözme Teknikleri

3.5. Problem Çözme Kavramları

- Diğer önemli kavramlar ise operatör ve fonksiyonlardır.
- "**Operatör**", sabit ve değişkenler arasındaki
- ilişkileri gösteren, eşitlik ve ifadelerde kullanılan işaret ve sembolleri ifade eder.
- Operatörlerin belirli bir hiyerarşik yapı içerisinde kullanılması gerekir.
- Operatörler sabit ve değişkenlerle birlikte kullanıldığında "**eşitlik**" ve "**ifade**" olarak adlandırılan yapılar oluşur.
- Eşitlik ve ifadeler ise çözüm surecinin yapı taşları olan işlemlerdir.
- "**Fonksiyonlar**" bir dizi işlem seti olarak tanımlanabilir.

3.1. Problem Çözme Teknikleri

3.6. Veri Türleri

- Ham veriler, bilgisayar tarafından **"girdi"** olarak algılanır ve program aracılığı ile işlenir.
- Kullanıcıya geri donen değer, işlenmiş veridir; **"çıktı"** ya da **"bilgi"** olarak adlandırılır.
- **Bilgisayara hangi veri türüyle çalışıyor olduğu mutlaka belirtilmelidir.**
- Bir programda farklı veri türleriyle işlem yapılabilir.
- Örneğin *tam sayılar*, *kesirli sayılar*, karakterler, simgeler, *metinler* ve mantıksal değerler, **veri türlerini** oluşturur.

3.1. Problem Çözme Teknikleri

3.6.1. Sayısal Veri

Veri Türü	Veri Seti	Örnek
Sayısal: Tam sayı	Tüm sayılar	66578 -2356
Sayısal: Reel sayı	Tüm reel sayılar ve ondalık sayılar	-56.23 8695.235 0.005

3.6.2. Alfanümerik/Karakter Veri

Veri Türü	Veri Seti	Örnek
Karakter	Tüm rakamlar, harfler ve özel semboller	"A", "Y", "k", "i", "6", "0", "+", "%"
Dizi	Birden fazla karakterden oluşan kombinasyon	"Bilgisayar", "532-5556633"

3.1. Problem Çözme Teknikleri

3.6.3. Mantıksal Veri

Veri Türü	Veri Seti	Örnek
Mantıksal	Doğru Yanlış True False	Doğru Yanlış True False

Örnekler

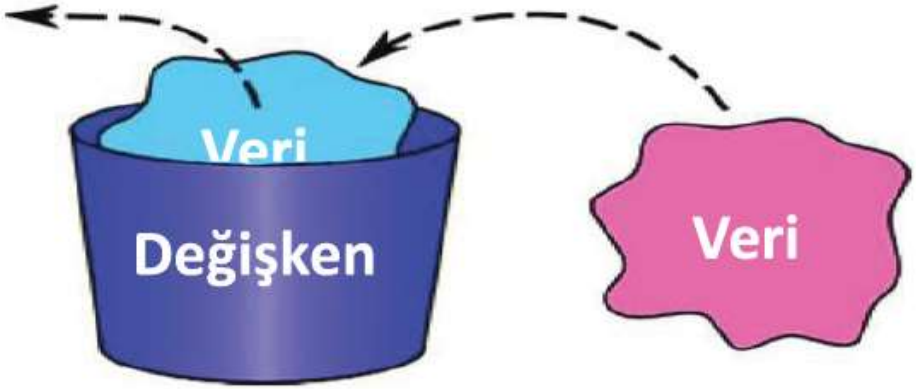
Veri	Veri Türü	Açıklama
Ürün satış bedeli: 49.99, 101.50	Sayısal: Reel	Bir ürünün satış bedeli hesaplama işlemlerinde kullanılır.
T.C. Kimlik No.: 10654876542	Karakter dizisi	Kimlik bilgileri hesaplama amaçlı kullanılmaz.
Ağırlık: 67	Sayısal: Tam sayı	Kilo cinsinden tam sayı olabilir ve hesaplamalarda kullanılır.
Şirket İsmi: ABC Firması	Karakter dizisi	Tamamen karakterlerden oluşur.
Kredi Onayı: Var, Yok	Mantıksal	Bu durumda onay ya vardır "Doğru" ya da yoktur "Yanlış".
Posta Kodu: 06110, 34217	Karakter dizisi	Posta kodları işlem yapmak için kullanılmaz.
Tarih: 21042017	Karakter dizisi, Sayısal Tam sayı	İşlem yapmak için tam sayı biçiminde tanımlanabilir; aksi takdirde dizi olarak tanımlanması daha uygundur.
IBAN: TR0600006543000012	Karakter dizisi	Para transferi için bankaya verilen kodlar hesaplama amaçlı kullanılmaz.

3.1. Problem Çözme Teknikleri

3.7. Bilgisayar Veriyi Nasıl Saklar?

- Bilgisayar veriyi hafızada saklar.
- Programın çalışması bittiğinde ya da bilgisayar kapatıldığında bu veriler silinir.
- Verilerin daha sonra tekrar kullanılması gerekiyorsa sabit disk gibi kalıcı bir konuma kaydedilmeleri gerekir.
- Bu şekilde kaydedilen verilere "dosya" adı verilir.

3.8. Sabit ve Değişkenler



Şekil 1.8: Değişken - Veri İlişkisi

3.1. Problem Çözme Teknikleri

3.8. Sabit ve Değişkenler

Değişkene içerdiği değer ile tutarlı isimler veriniz.

Değişkenlere isim verirken boşluk kullanmayınız.

Değişkenlere isim verirken bir karakter ile başlayınız.

Matematiksel semboller kullanmamaya dikkat ediniz.

3.1. Problem Çözme Teknikleri

3.8. Sabit ve Değişkenler

Değişken Kullanımında Doğru Yanlış Örnekler

Yanlış	Doğru
1 sayı	sayi1
Okul No.	okulNo
Soru?	soru

3.1. Problem Çözme Teknikleri

3.8. Sabit ve Değişkenler Kullanırken Uyulması Gerekenler

Bazı platformlar desteklemediği için Türkçe karakter kullanımı tavsiye edilmez.

- Ş, Ç, Ğ, İ, Ü, Ö

Programlama dillerinde kullanılan komut isimleri değişken olarak kullanılamaz.

- Çok bilinenleri; if, for, while, else, do, int, vb.

Değişken isimlendirmelerinde boşluk karakteri yerine alt çizgi (_) karakteri kullanılabilir

Genellikle küçük harfle başlanır ve ikinci bir kelime yazılacaksa ilk kelimenin hemen ardından büyük harfle devam edilir.

- Buna "Camel Karakter" kullanımı denir. Örnek: tcKimlikNo

Özel karakterler değişken isimlerinde kullanılamaz

- (*,/, -,+, #,%,&,(,=,?,$,[,{ gibi…).

3.1. Problem Çözme Teknikleri

3.9. Fonksiyonlar

- Fonksiyonlar, belirli işlemleri yürüten ve sonuçları döndüren bir işlem kümesidir.
- Problem çözme surecinde tekrarlanan işlemler için kullanılır ve böylece programcının, hem problemi daha hızlı çözmesini hem de programın daha anlaşılır olmasını sağlar.
- Her programlama dili, içerisinde kendine özgü fonksiyonlar barındırır.
- Fonksiyonlar, kendilerine verilen isim ve ayraç içerisinde gönderilen veri ile tanımlanır.
- Fonksiyonlara veri gönderilir. Fonksiyona gönderilen verilere "**parametre**" denir.

Fonksiyon İsmi (Veri)

3.1. Problem Çözme Teknikleri

3.9. Fonksiyonlar

Fonksiyonlar gruplara ayrılır:

1. Matematiksel Fonksiyonlar: Matematiksel işlemler için kullanılır.

2. Dizi Fonksiyonlar: Dizi ve karakterlerle ilgili işlemleri gerçekleştirmek için kullanılır.

3. Dönüştürme Fonksiyonları: Veriyi bir türden diğerine dönüştürmek için kullanılır.

4. İstatistiksel Fonksiyonlar: Maksimum değer, ortalama gibi değerleri hesaplamak için kullanılır.

5. Yardımcı Fonksiyonlar: Program dışındaki verilere erişerek işlem yapmak için kullanılır.

3.1. Problem Çözme Teknikleri

3.9. Fonksiyonlar

Tablo 1: Fonksiyon türleri ve örnekler

Fonksiyon	Tanım	Örnek	Sonuç
Matematiksel Fonksiyonlar			
Sqrt (N)	N değerinin karekökünü döndürür.	Sqrt(16)	4
Abs (N)	N değerinin mutlak değerini döndürür.	Abs(-6)	6
Integer (N)	N değerine en yakın ya da eşit tam sayıyı döndürür.	Integer(6.7689)	6
Random	0 ile 1 arasında rastgele bir sayı döndürür.	Random	0.6783456
Dizi Fonksiyonlar			
Mid (S, n1, n2)	Dizinin n1 pozisyonundan başlayan n2 kadar karakteri döndürür.	Mid(S, 3, 3) S= "Yasemin"	"sem"
Left (S, n)	Dizinin sol tarafındaki n kadar karakteri döndürür.	Left(S, 3) S= "Yasemin"	"yas"
Right (S, n)	Dizinin sağ tarafındaki n kadar karakteri döndürür.	Right(S, 4) S= "Yasemin"	"emin"
Length (S)	Dizideki karakter sayısını döndürür.	Length(S) S= "Yasemin"	7

3.1. Problem Çözme Teknikleri

3.9. Fonksiyonlar

Dönüştürme Fonksiyonları			
Value (S)	Dizi olarak tanımlanan değişkeni sayısal değere çevirir.	Value("65.21")	+65.21
String (N)	Sayısal değeri dizi değerine çevirir.	String(+65.21)	"65.21"
İstatistiksel Fonksiyonlar			
Average (list)	Birkaç sayı için ortalama değeri döndürür.	Average(12, 24, 6)	14
Sum (list)	Birkaç sayının toplam değerini döndürür.	Sum(3, 5, 8)	16
Yardımcı Fonksiyonlar			
Date	Sistemin andaki tarih değerini döndürür.	Date	04/23/2017
Time	Sistemin şu andaki zaman değerini döndürür.	Time	20.57.36

3.1. Problem Çözme Teknikleri

3.10. Operatörler

- Bilgisayara, verileri nasıl işleyeceğini belirtmek gerekir. Bu işlem için operatörler kullanılır.
- "Operatörler" verileri, ifade ve eşitlikler ile birleştirir.
- "**İşlemci**" ve "**sonuç**", operatorlere ilişkin iki kavramdır.
- **İşlemci**, verileri bağlayan ve işleme alan yapı;
- **Sonuç** ise yapılan işlemin yanıtıdır.

3.1. Problem Çözme Teknikleri

3.10. Operatörler

Tablo 2: Operatör türleri ve örnekler

Operatör	Bilgisayar Sembolü	Örnek	
Matematiksel		**İşlem**	**Sonuç**
Toplama	+	6.7 + 2.1	8.8
Çıkarma	-	5.6-3.4	2.2
Çarpma	*	3.0*4.0	12.0
Bölme	/	40.0/8.0	5
Modül Alma	MOD	9 MOD 3	3
İlişkisel**		**İşlem**	**Sonuç**
Eşit	==	6 == 8	False
Küçüktür	<	6 < 8	True
Büyüktür	>	6 > 8	False
Küçük ya da eşittir	<=	6 <= 8	True
Büyük ya da eşittir	>=	6 >= 8	False
Eşit değildir	<>	6 <> 8	True
Mantıksal		**İşlem**	**Sonuç**
Değil	NOT	NOT True	False
Ve	AND	True AND True	True
Veya	OR	True OR False	True

3.1. Problem Çözme Teknikleri

3.11. İşlem Önceliği

Matematiksel, mantıksal ve ilişkisel operatörlerin bir hiyerarşisi yani öncelikleri vardır.

En içteki ayraçtan en dıştakine doğru işlem yapılmalı,

Ayraç içerisinde ise işlem önceliklerine dikkat edilmelidir.

3.1. Problem Çözme Teknikleri

3.12. İfade ve Eşitlikler

Şu ana kadar gördüğümüz tüm bileşenler, ifade ya da eşitlik biçiminde kullanılmadığı surece bir anlam ifade etmez.

Tablo 4: İfade ve eşitlikler

İfadeler	Eşitlikler
A + B A ve B sayısal veridir. Sonuç sayısaldır ve hafızada korunmaz.	**C = A + B** A, B ve C sayısal veridir. Sonuç sayısaldır ve C değişkenine atanarak korunur.
A < B A ve B sayısal, karakter ya da dizi olabilir. Sonuç mantıksal değerdir ve hafızada korunmaz.	**C = A < B** A, B ve C sayısal, karakter ya da dizi olabilir. Sonuç mantıksal değerdir ve C değişkenine atanarak korunur.
A OR B A ve B mantıksal veridir. Sonuç mantıksaldır ve hafızada korunmaz.	**C = A OR B** A, B ve C mantıksal veridir. Sonuç mantıksaldır ve C değişkenine atanarak korunur.