**BUKALEMUN**

Bukalemun Windows ve Mobil ortamda Türkçe kodlama yapısı ile uygulama oluşturmak için geliştirilmiş bir dildir.

Bukalemun ile yazılan uygulamalar bir kere derlendikten sonra Windows ve Android üzerinde çalıştırılabilir.

Kod yapısı olarak Bukalemun; C#, python, ruby gibi dillerin kodlama yapısından esinlenerek yazımı çok basit olan bir dil olarak hazırlanmıştır. Bu özelliğinden dolayı bukalemun dilinin asıl var oluş amacı programlama öğrenmek isteyenlere temel olmasıdır.

|  |
| --- |
| Örnek Bir Bukalemun Programı |
| sınıf faktöriyel  {  fonksiyon başla()  {  sayı a;  yaz("Faktöriyeli alınacak sayıyı giriniz..: ");  a = oku();  sayı sonuç = 1;  say (sayı x: 1...a)  {  sonuç \*= x;  }  yaz("Sonuç: ", sonuç);  }  } |

**başla()** fonksiyonu bukalemun dili için özel bir fonksiyondur. Parametre almaz. Program başlatılınca ilk olarak başla fonksiyonu çalıştırılır.

**Bukalemun dosya uzantıları**

Bukalemun dili için kullanılan uzantılar aşağıdaki gibidir.

.buk : Bukalemun kod dosyası.

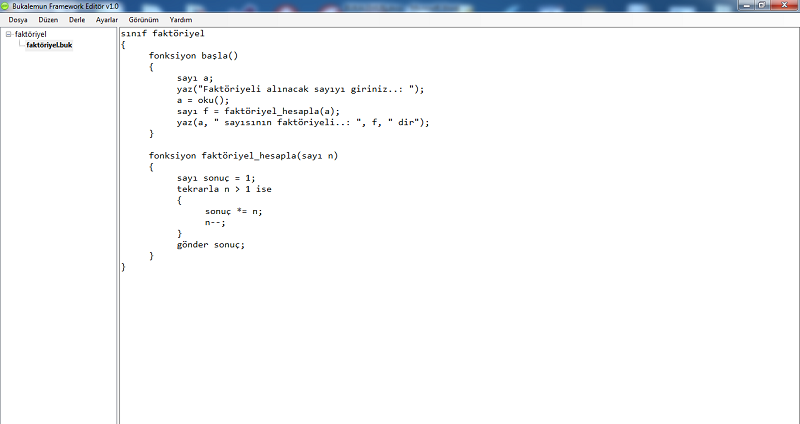
.ale : Deleyiciye birden fazla .buk dosyası gönderilecekse bunlar .ale dosyası içine satır satır yazılarak gönderilir.

.mun : Derlenmiş kod dosyası (Konsol uygulamaları, sadece windows)

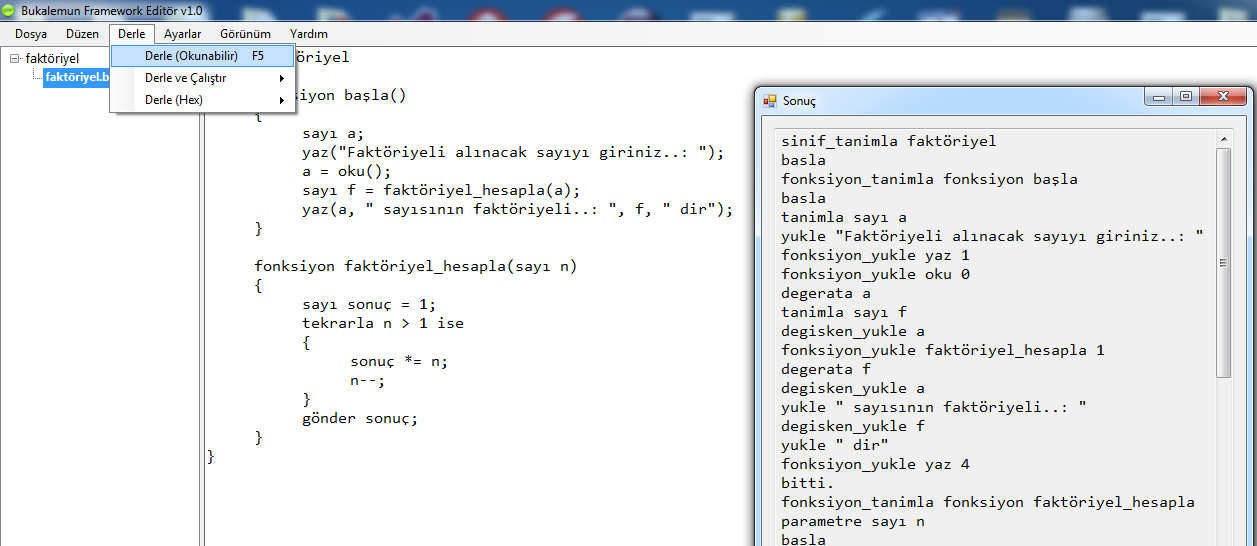
.munf : Derlenmiş kod dosyası (Form uygulamaları, Windows ve android)

**Bukalemun Editör**

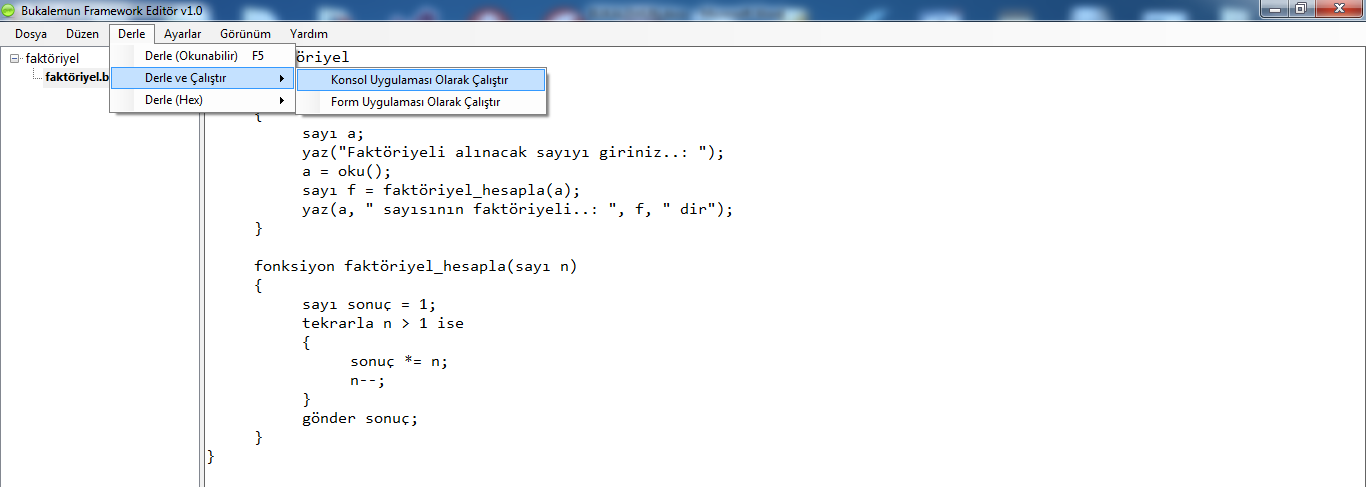
Bukalemun kod yazıp derleyiciye göndermenize olanak sağlayan basit bir editöre sahiptir.

****

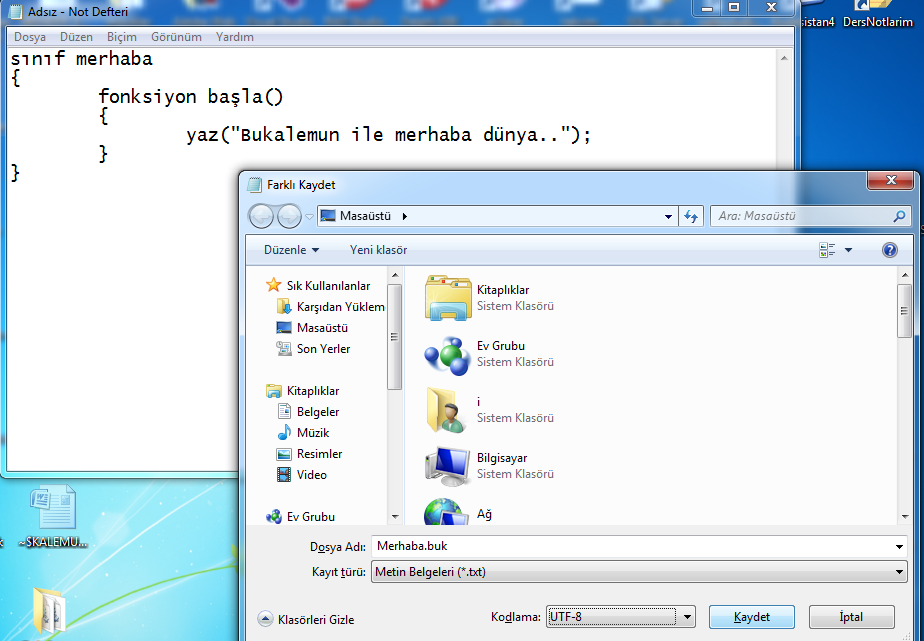
Yazdığınız kodların yorumlanabilir kodlara çevrildikten sonraki halini görmek isterseniz:



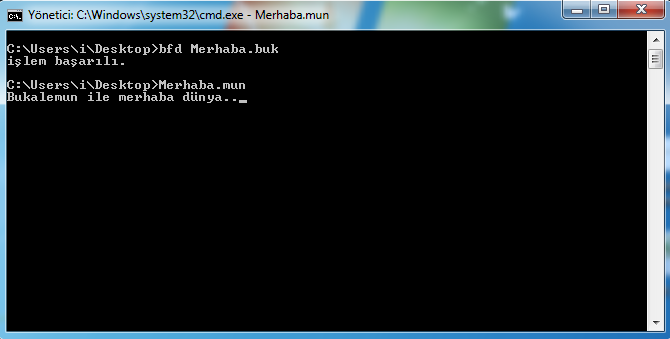
Hazırladığınız uygulamanın form uygulaması veya konsol uygulaması olmasına göre derleme işlemini aşağıdaki gibi yapabilirsiniz.



Bukalemun editör yerine başka bir program veya not defterini de kullanabilirsiniz. Aşağıda not defteri ile yazılan bir kodun nasıl derleneceği gösteriliyor.



Not defteri ile kodlar yazıldıktan sonra uzantısı “.buk” olarak UTF-8 formatında kaydediliyor.



Komut satırı çalıştırılarak aşağıdaki satır yazılıyor böylece kodlar derleniyor.

Konsol uygulamaları için:

**bfd Merhaba.buk**

form uygulamaları için:

**bfd dosyaadı.buk –form**

komutları çalıştırılabilir.

Derlenen dosyayı çalıştırmak için ise dosyaya çift tıklanabilir veya komut satırından dosyanın adı yazılır.

**Değişkenler**

Herhangi bir verinin depolanmasını sağlar. Değişken tanımlaması aşağıdaki şekilde yapılır.

*Değişken\_türü değişken\_adı;*

Örnekler:

sayı a;

Bellekte sayı türündeki veriyi saklayabilecek bir alan oluşturur.

metin b;

Bellekte metin türündeki veriyi saklayabilecek bir alan oluşturur.

doğruyanlış c;

Bellekte doğruyanlış türündeki veriyi saklayabilecek bir alan oluşturur.

**Değişken isimlendirme kuralları**

* Değişken adları alfabedeki karakterler, rakamlar ve \_ (alt çizgi) karakterinden oluşabilir.
* Değişken adları sayı ile başlayamaz.
* Büyük harf ve küçük harf ayrımına dikkat edilmelidir. “şifre” ile “Şifre” farklı değişkenlerdir.
* Bukalemun dili için kullanılan anahtar kelimeler değişken adı olarak kullanılmamalıdır.

**Değişken Türleri**

**sayı:** 32 bitlik veri depolayabilir.-2.147.483.648 ile 2.147.483.647 arasında değer alabilen değişken türüdür.

sayı a = 312;

sayı b = 234252341;

sayı c = -9595313;

**vsayı:** 64 bitlik veri depolayabilir. - 5\*10-324 : + ya da - 1,7\*10308 arasında değer alabilir.

vsayı a = 15.67;

vsayı b = 0.01;

vsayı c = -598.44121;

**metin:** Karakterlerden oluşan verileri depolamak için kullanılır.

metin a = “Antalya”;

metin b = “Bilişim Teknolojileri”;

metin c = “Bukalemun.”;

**karakter:** 8 bitliktir. 1 karakter depolayabilir.

karakter a = ‘i’;

karakter b = ‘R’;

karakter c = ‘#’;

**doğruyanlış:** 1 bitliktir. Alabileceği değerler “doğru” (evet, true), “yanlış” (hayır, false).

doğruyanlış a = doğru;

doğruyanlış b = yanlış;

doğruyanlış c = evet;

**Değişkenlere değer atama**

= operatörü, sağındaki verileri solundaki değişkene atar.

*Örnekler:*

sayı sayı1 = 50;

Örnekte sayı1 değişkeni tanımlanıp 50 sayısı atanmıştır.

metin m1, m2; //metin türündeki m1 ve m2 değişkenleri tanımlanıyor.

m1 = “Bilişim”; // m1 değişkenine “Bilişim” verisi atanıyor.

m2 = m1 + “Teknolojileri”; // m2 değişkenine (m1 + “Teknolojileri”) değeri atanıyor. Böylece m1 değişkeninin değeri “BilişimTeknolojileri” oluyor.

Not: Değişkene tanımlandığı anda değer atanabildiği gibi sonradan da değer atanabilir.

**Değişkenlerin değerlerini ekrana yazdırma**

Bukalemun dilinde değişkenlerin değerlerini veya sabit verileri konsol ekranına yazdırmak için aşağıdaki fonksiyonlar kullanılır.

**yaz(parametreler…)**

Parametre olarak verilen verilerin tamamını ekrana yazdırır.

*Örnek 1:*

sınıf merhaba

{

fonksiyon başla()

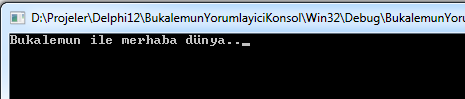
{

yaz("Bukalemun ile merhaba dünya..");

}

}

*Ekran Çıktısı:*



*Örnek 2:*

sınıf merhaba

{

fonksiyon başla()

{

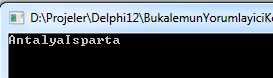
yaz("Antalya");

yaz("Isparta");

}

}

*Ekran Çıktısı:*



Not: yaz fonksiyonu verileri yan yana yazar.

**yazA(parametreler…)**

Parametre olarak verilen verilerin tamamı ekrana yazdırılır ve satır sonu eklenir. Böylece yeniden yazdırılan veriler alt satıra yazılır.

*Örnek1:*

sınıf merhaba

{

fonksiyon başla()

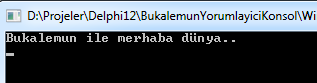
{

yazA("Bukalemun ile merhaba dünya..");

}

}

*Ekran Çıktısı:*



İmleç alt satırdadır.

*Örnek 2:*

sınıf merhaba

{

fonksiyon başla()

{

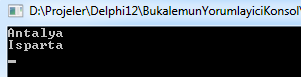
yazA("Antalya");

yazA("Isparta");

}

}

*Ekran Çıktısı:*



Not: Her yazA fonksiyonu verileri yazdıktan sonra alt satıra geçer bir sonraki veriler yeni satıra yazılır.

*Örnek 3:*

sınıf ekranayaz

{

fonksiyon başla()

{

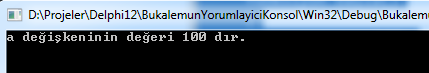
sayı a = 100;

yaz("a değişkeninin değeri ", a, " dır.");

}

}

*Ekran Çıktısı:*



*Örnek 4:*

sınıf ekranayaz

{

fonksiyon başla()

{

sayı a = 100;

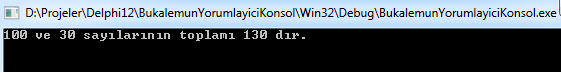
sayı b = 30;

yaz(a, " ve ", b, " sayılarının toplamı ", a+b, " dır.");

}

}

*Ekran Çıktısı:*



Not: yaz ve yazA fonksiyonları için istenildiği kadar parametre verilebilir. Bu parametreler virgül ile ayrılmalıdır.

**Klavyeden değişkenlere değer atama**

Klavyeden girilen bir bilgi istenilen bir değişkene atanabilir. Bukalemun tür dönüşümlerini kendisi yapar. Okunan veri atanan değişkenin türüne göre otomatik olarak dönüştürülür.

**oku()**: klavyeden veri almak için kullanılan fonksiyondur.

*Örnek 1:*

sınıf ekranayaz

{

fonksiyon başla()

{

yaz("Adınız..: ");

metin a;

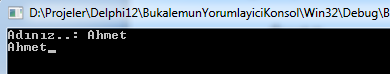
a = oku();

yaz(a);

}

}

*Ekran Çıktısı:*



Örnekte öncelikle ekrana “Adınız..: “ verisi yazılmıştır. Sonrasında tanımlanan a değişkenine oku() fonksiyonu ile kullanıcının yazdığı veri atanmıştır. Yaz fonksiyonu ile de değişkenin değeri ekrana yazılıyor.

*Örnek 2:*

sınıf klavyedenoku

{

fonksiyon başla()

{

sayı a;

yaz("Bir sayı giriniz..: ");

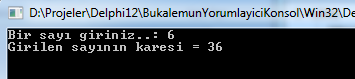
a = oku();

yaz("Girilen sayının karesi = ", a\*\*2);

}

}

*Ekran Çıktısı:*



Yukarıdaki örnekte sayı türünde bir değişken tanımlanıyor. Klavyeden okunan değer tür dönüşümü yapmadan doğrudan a değişkenine atanıyor. Sonrasında ekrana girilen sayının karesi yazdırılıyor.

\*\* operatörü kendisinden önceki sayının üssünü hesaplar.

*Örnek 3:*

sınıf klavyedenoku

{

fonksiyon başla()

{

yaz("Bir karakter giriniz..: ");

karakter c;

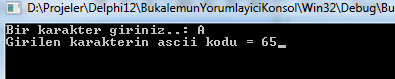
c = oku();

yaz("Girilen karakterin ascii kodu = ", ascii(c));

}

}

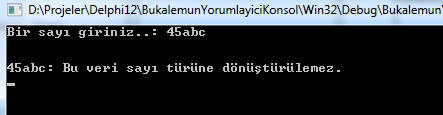
*Ekran Çıktısı:*



Örnekte oku() fonksiyonu bu sefer tek bir karakter okumuştur ve bu değer c değişkenine atanmıştır. Yaz fonksiyonu içindeki ascii(c) fonksiyonu ile de girilen karakterin ascii kodu ekrana yazdırılmıştır.

**Değişkenlere değer atarken oluşan hata mesajları**

Değişkenlere değer atarken oluşabilecek hatalarda aşağıdaki gibi bir hata mesajı ile karşılaşılıyorsa atanmak istenilen değer değişkenin türüne dönüştürülemez demektir.



Yukarıdaki ekran görüntüsünün elde edildiği program aşağıdadır. Programda sayı türündeki c değişkenine sayı dışında bir veri atanmak istenirse örnekteki hata mesajı ile karşılaşılır.

sınıf klavyedenoku

{

fonksiyon başla()

{

yaz("Bir sayı giriniz..: ");

sayı c = oku();

yaz("Girilen sayı ", c);

}

}

**Açıklama satırları**

Yazılan uygulamalarda kodlar arasında açıklama yazmak gerekebilir. Bazen yapılan işlemin açıklaması bazen ileride dikkat edilmesi gereken noktalar açıklama olarak yazılabilir. Bukalemun dilinde kodlar arasına açıklama yazmak için aşağıdaki yöntemler kullanılır.

1. Yöntem: //

İki slash karakteri yan yana yazılırsa, bu karakterlerden sonra satır sonuna kadar yazılanlar dikkate alınmaz.

*Örnek:*

sınıf açıklamalar

{

fonksiyon başla()

{

sayı sayı1 = 100; // bu satırlar dikkate alınmaz

// Bu satırlar da dikkate alınmaz.

}

}

1. Yöntem: /\*

\*/

Slash karakterinden sonra \* karakteri varsa, tekrar \* ve / karakterlerine kadar yazılanlar dikkate alınmaz.

*Örnek:*

sınıf açıklamalar

{

fonksiyon başla()

{

sayı sayı1 = 100; /\* bu araya

yazılan hiçbir şey

dikkate alınmaz. \*/

yaz(sayı1);

}

}

**Operatörler**

Değişkenler ve veriler üzerinde işlem yapılmasını sağlar. Bukalemun dilindeki operatörler aşağıdaki tabloda listelenmiştir.

|  |  |  |  |
| --- | --- | --- | --- |
| **Operatör** | **Açıklama** | **Operatör** | **Açıklama** |
| + | Toplama | += | Toplayarak ata |
| - | Çıkarma | -= | Çıkararak ata |
| \* | Çarpma | \*= | Çarparak ata |
| / | Bölme | /= | Bölerek ata |
| ++ | Bir arttır | -- | Bir azalt |
| = | Atama | % | Mod |
| \*\* | Üssü | %= | Mod alarak ata |
| == | Eşit mi? | != | Eşit değil mi? |
| < | Küçük | <= | Küçük ve eşit |
| > | Büyük | >= | Büyük ve eşit |
| ve | Şartların hepsi doğru | veya | Şartların en az 1 tanesi doğru |
| ! | Değil |  |  |

**= (Atama) Operatörü:** Operatörün sağındaki verileri solundaki değişkene atar.

**+ (Toplama) Operatörü:** Sayısal verileri üzerinde toplama işlemi yapmak için kullanılır.

sayı s = 2 + 5; // s değişkeninin değeri 7

**- (Çıkarma) Operatörü:** Sayısal veriler üzerinde çıkarma işlemi yapmak için kullanılır.

sayı s = 5 – 3; // s değişkeninin değeri 2

**\* (Çarpma) Operatörü:** Sayısal veriler üzerinde çarpma işlemi yapmak için kullanılır.

sayı s = 5 \* 3; // s değişkeninin değeri 15

**/ (Bölme) Operatörü:** Sayısal veriler üzerinde bölme işlemi yapmak için kullanılır.

sayı s = 10 / 2; // s değişkeninin değeri 5

Not:

10 / 3; // bu işlemin sonucu 3 tür. Çünkü 10 tamsayısının 3 tamsayısına bölündüğünde sonuç tamsayı olacaktır.

10 / 3.0 // bu işlemin sonucu ise 3.333333 tür. İşleme giren sayılardan en az birinin virgüllü sayı olması sonucu da virgüllü sayı yapar.

**% (Mod) Operatörü:** Sayısal bir verinin modunu almak için kullanılır.

sayı a = 20 % 3; // a değişkeninin değeri 2

**++ Operatörü:** Sayısal değişkenin değerini 1 arttırır.

sayı a = 5;

a++; // a değişkeninin değeri 6

**-- Operatörü:** Sayısal değişkenin değerini 1 azaltır.

sayı a = 5;

a--; // a değişkeninin değeri 4

**+= Operatörü:** Toplayarak ata.

sayı a = 10;

a += 5; // a değişkeninin değeri 15

**-= Operatörü:** Çıkararak ata.

sayı a = 10;

a -= 3; // a değişkeninin değeri 7

**\*= Operatörü:** Çarparak ata.

sayı a = 10;

a \*= 5; // a değişkeninin değeri 50

**/= Operatörü:** Bölerek ata.

sayı a = 10;

a /= 5; // a değişkeninin değeri 2

**%= Operatörü:** Mod alarak ata.

sayı a = 10;

a %= 3; // a değişkeninin değeri 1

**\*\* Operatörü:** Üssünü al.

sayı a = 5 \*\* 2; // a değişkeninin değeri 25

sayı b = 4 \*\* 3; // b değişkeninin değeri 64

**Karşılaştırma Operatörleri**

**== (Eşit mi?) Operatörü**: İki verinin birbirine eşit olup olmadığının kontrolünü yapar. Eşit ise **doğru** değilse **yanlış** değerini döndürür.

sayı a = 5, b = 7;

a == 5 // doğru

b == 5 // yanlış

**!= (Eşit değil mi?)** **Operatörü:** İki verinin birbirinden farklı olup olmadığının kontrolünü yapar. Eşit değil ise **doğru** değilse **yanlış** değerini döndürür.

sayı a = 6, b = 8;

a != 6 // yanlış

b != 10 // doğru

**< (Küçük) Operatörü:** Birinci verinin ikinci veriden küçük olup olmadığını kontrol eder. Küçük ise **doğru** değilse **yanlış** değerini döndürür.

sayı a = 6, b = 8;

a < 10 // doğru

b < 5 // yanlış

**> (Büyük) Operatörü:** Birinci verinin ikinci veriden büyük olup olmadığını kontrol eder. Büyük ise **doğru** değilse **yanlış** değerini döndürür.

sayı a = 6, b = 8;

a > 10 // yanlış

b > 5 // doğru

**<= (Küçük ve eşit) Operatörü:** Birinci verinin ikinci veriden küçük ve eşit olup olmadığını kontrol eder. Küçük ve eşit ise **doğru** değilse **yanlış** değerini döndürür.

sayı a = 6, b = 8;

a <= 6 // doğru

b <= 5 // yanlış

**>= (Büyük ve eşit) Operatörü:** Birinci verinin ikinci veriden büyük ve eşit olup olmadığını kontrol eder. Büyük ve eşit ise **doğru** değilse **yanlış** değerini döndürür.

sayı a = 6, b = 8;

a >= 6 // doğru

b >= 10 // yanlış

**Mantıksal operatörler**

**ve, && Operatörü:** ve, && sorgulanan tüm şartların doğru olması durumunda **doğru** değilse **yanlış** değerini döndürür.

sayı a = 5, b = 9;

a == b ve a < 10 ve b > 5 // yanlış (a değeri b den farklı olduğu için)

b == 9 ve b > 1 ve a > 2 // doğru

**veya, || Operatörü:** veya, || sorgulanan şartlardan en az bir tanesinin doğru olması durumunda **doğru** değerini döndürür. Aksi halde **yanlış** değerini döndürür.

sayı a = 5, b = 9;

a != b veya a < 3 veya b > 10 // doğru (sadece a!=b şartı sağlanıyor)

a > b veya a > 6 veya b < 8 // yanlış (Hiçbir şart sağlanmıyor)

a < b veya a < 10 veya b < 20 // doğru (tüm şartlar sağlanıyor)

**! (Değil) Operatörü:** Değeri tersine çevirir. Doğru ise **yanlış**, yanlış ise **doğru** değerini döndürür.

!doğru // yanlış

!yanlış // doğru

**Kontrol İfadeleri**

Bazı kodların uygun şartlar sağlanması durumunda çalışması istenebilir. Bu durumda kontrol ifadeleri kullanılır.

**eğer**

eğer ifadesi ile belirli bir şartın sağlanması durumunda yapılacak işlemler belirlenir. Bu işlemler şartlar sağlanırsa çalıştırılır aksi durumda çalıştırılmaz.

*Kullanımı:*

eğer şart ise

{

// şart doğru olması durumunda çalışacak kodlar

}

*Örnek:*

sınıf ehliyet

{

fonksiyon başla()

{

yaz("Yaşınızı giriniz..: ");

sayı yaş;

yaş = oku();

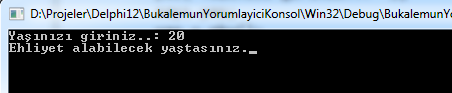
eğer yaş >= 18 ise

yaz("Ehliyet alabilecek yaştasınız.");

}

}

*Ekran Çıktısı:*



Yukarıdaki örnekte kullanıcının girdiği yaş bilgisi kontrol edilerek 18 den büyük ve eşit ise ekrana “Ehliyet alabilecek yaştasınız.” bilgisi yazdırılıyor.

**değilse**

eğer ifadesi ile birlikte kullanılır. Eğer ifadesi şart sağlanmadığı için çalışmazsa **değilse** bölümü çalışır.

*Kullanımı:*

eğer şart ise

{

// şart doğru ise çalışacak kodlar

}

değilse

{

// şart yanlış ise çalışacak kodlar

}

*Örnek 1:*

sınıf ehliyet

{

fonksiyon başla()

{

yaz("Yaşınızı giriniz..: ");

sayı yaş;

yaş = oku();

eğer yaş >= 18 ise

yaz("Ehliyet alabilecek yaştasınız.");

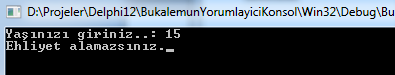
değilse

yaz("Ehliyet alamazsınız.");

}

}

*Ekran Çıktısı:*



Örnekte kullanıcının girdiği 15 değeri eğer ifadesindeki şartı sağlamadığı için **değilse** bölümü çalışmıştır.

*Örnek 2:*

sınıf pozitifnegatif

{

fonksiyon başla()

{

yaz("Bir sayı giriniz..: ");

sayı a;

a = oku();

eğer a < 0 ise

yaz("Girilen sayı negatif.");

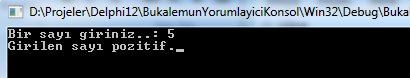
değilse

yaz("Girilen sayı pozitif.");

}

}

*Ekran Çıktısı:*



Örnekte kullanıcının girdiği sayının 0 (sıfırdan) küçük olup olmadığı kontrol ediliyor. 5 değeri 0 (sıfır) dan küçük olmadığı için değilse bölümü çalışmaktadır.

**eğer – değilse eğer**

Bazı durumlarda eğerlerden bir tanesinin şartlarının sağlanması durumunda diğer eğerlerin atlanması gerekebilir. Bu durumda eğer ifadelerini aşağıdaki gibi kullanabilirsiniz.

*Kullanımı:*

eğer şart1 ise

{

// şart1 doğru ise çalışacak kodlar

}

değilse eğer şart2 ise

{

// şart2 doğru ise çalışacak kodlar

}

değilse eğer şart3 ise

{

// şart3 doğru ise çalışacak kodlar

}

değilse

{

//şartların hiçbiri sağlanmazsa çalışacak kodlar

}

*Örnek:*

sınıf şartlıifadeler

{

fonksiyon başla()

{

yaz("Matematik notunuzu giriniz..: ");

sayı not = oku();

eğer not <= 100 ve not >= 85 ise

yaz("Pek iyi");

değilse eğer not < 85 ve not >= 70 ise

yaz("İyi");

değilse eğer not < 70 ve not >= 55 ise

yaz("Orta");

değilse eğer not < 55 ve not >= 45 ise

yaz("Geçer");

değilse eğer not < 45 ve not >= 0 ise

yaz("Başarısız");

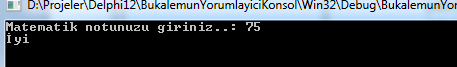
değilse

yaz("Girilen not 0 - 100 arasında olmalıdır.");

}

}

*Ekran Çıktısı:*



Örnekte girilen not bilgisi için tüm durumlar kontrol edilerek sonuç yazdırılmıştır. Eğer ifadelerine bağlı olan şartlardan herhangi birinin sağlanması durumunda diğer eğer ifadeleri kontrol edilmez.

**İç içe eğer ifadesi**

Bazı şartların çalışması başka şartlara bağlı olduğu durumlarda iç içe eğer ifadeleri kullanılır.

*Kullanımı:*

eğer şart1 ise

{

eğer şart2 ise

{

// şart1 ve şart2 doğru ise çalışacak kodlar

}

değilse

{

// şart1 ve şart2 yanlış ise çalışacak kodlar

}

}

değilse

{

//şart1 sağlanmazsa çalışacak kodlar

}

*Örnek:*

sınıf şartlıifadeler

{

fonksiyon başla()

{

karakter cinsiyet;

yaz("Cinsiyetiniz (e:erkek, b: bayan) ");

cinsiyet = oku();

eğer cinsiyet == 'e' veya cinsiyet == 'E' ise

{

sayı yaş;

yaz("Yaşınızı giriniz ");

yaş = oku();

eğer yaş >= 20 ise

yaz("Askerlik çağınız gelmiş.");

değilse yaz("Askerlik için küçüksünüz.");

}

değilse

{

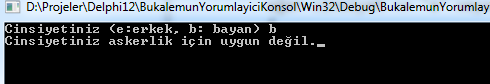
yaz("Cinsiyetiniz askerlik için uygun değil.");

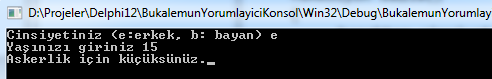
}

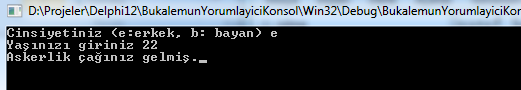
}

}

*Ekran Görüntüleri:*







Yukarıdaki örnekte askerlik için yaş kontrolü cinsiyete bağlı olarak çalışmaktadır. Cinsiyet olarak e (erkek) girilirse yaş ile ilgili veriler kullanıcıdan alınıyor ve tekrar karşılaştırma yapılıyor.

**Döngü deyimleri**

Belirli bir kod veya kod grubunun birden fazla çalışması istenildiği durumlarda döngüler kullanılır.

Bukalemun içerisinde 2 çeşit döngü vardır. Bunlar;

* say
* tekrarla

**say döngüsü**

Başlangıç, bitiş ve artış değerinin belli olduğu durumlarda say döngüsü kullanılabilir.

*Kullanımı:*

say (döngü değişkeni: başlangıç .. bitiş, artış)

{

//çalışacak kodlar

}

**Döngü değişkeni:** Döngüde sayılan değeri tutacak olan değişken.

**Başlangıç:** Döngünün sayma işlemine başlayacağı sayı.

**.. :** iki nokta yan yana ile oluşturulan döngülerde alt sınır dahil üst sınır dahil değildir.

**… :** üç nokta yan yana ile oluşturulan döngülerde alt sınır ve üst sınır sayma işlemine dahildir.

**Bitiş:** Döngünün hangi sayıya kadar sayacağı

**Artış:** kaçar kaçar sayılacağı. Yazmak zorunlu değildir. Yazılmazsa varsayılan olarak 1 er sayar.

*Örnek1: 1 den 10 a kadar (10 dahil değil) sayıları ekrana yazan program*

sınıf döngü

{

fonksiyon başla()

{

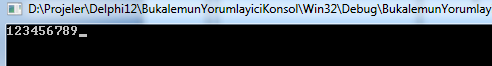
say (sayı a: 1..10)

yaz(a);

}

}

*Ekran Çıktısı:*



Yukarıdaki örnek incelendiğinde;

sayı a döngü değişkeni olarak a tanımlanıyor

1. döngünün başlangıç değeri

.. başlangıç değeri dahil, bitiş değeri dahil değil

10 döngünün bitiş değeri

yaz fonksiyonuyla a değişkeni ekrana yan yana yazdırılıyor.

*Örnek 2: 2 den 14’e kadar 3 er ekrana yazan program*

sınıf döngü

{

fonksiyon başla()

{

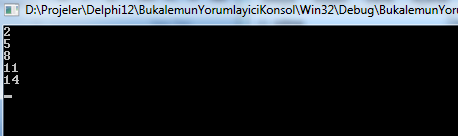
say (sayı a: 2...14, 3)

yazA(a);

}

}

*Ekran Çıktısı:*



Örnekte başlangıç değeri 2, … ile bitiş değerinin dahil olduğu, 3 ile de sayma işleminin 3 er yapılacağı belirleniyor.

*Örnek 3:20 den geriye 10 a kadar ekrana yazan program*

sınıf döngü

{

fonksiyon başla()

{

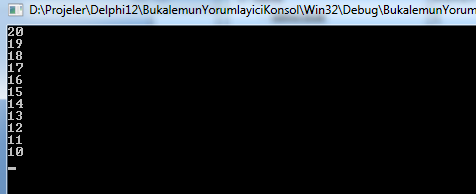
say (sayı a: 20...10, -1)

yazA(a);

}

}

Ekran Çıktısı:



Bu örnekte ise 20 den geriye 10’a kadar (10 dahil) 1 er sayılıyor.

*Örnek 4: 2 - 100 arasındaki asal sayıları ekrana yazan program*

sınıf asalsayılar

{

fonksiyon başla()

{

say (sayı a: 2...100)

{

doğruyanlış asal = doğru;

say (sayı b: 2..a)

{

sayı kalan = a % b;

eğer kalan == 0 ise

asal = yanlış;

}

eğer asal ise

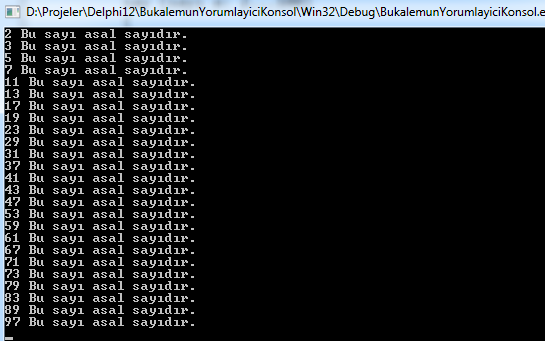
yazA(a, " Bu sayı asal sayıdır.");

}

}

}

*Ekran Çıktısı:*



Örnekte iç içe 2 adet **say** döngüsü oluşturularak asal sayıları bulan program kodları verilmiştir.

**tekrarla**

Sayma işleminin belirli bir şarta göre yapılacağı durumlarda kullanılır. Şart doğru olduğu sürece döngü tekrarlanır.

*Kullanımı:*

tekrarla şart ise

{

// kodlar

}

*Örnek 1:1 den 10’a kadar olan sayıları ekrana yazan program*

sınıf döngü

{

fonksiyon başla()

{

sayı a = 1;

tekrarla a <= 10 ise

{

yazA(a);

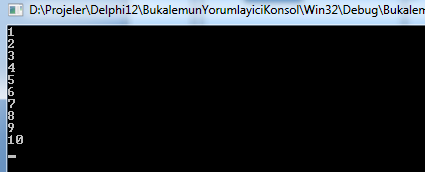
a++;

}

}

}

*Ekran Çıktısı:*



Örnekte şart a değişkeninin 10 dan küçük ve eşit olması durumudur. Döngünün her çalışmasında a değişkeninin değeri 1 arttırılarak şartın sağlanmadığı duruma kadar devam eder. Böylece 1 den 10 a kadar sayan bir döngü oluşturulabilir.

*Örnek 2: Klavyeden girilen sayının faktöriyelini hesaplayan program*

sınıf faktöriyel

{

fonksiyon başla()

{

sayı a;

yaz("Faktöriyeli alınacak sayıyı giriniz..: ");

a = oku();

sayı sonuç = 1;

tekrarla a >= 1 ise

{

sonuç \*= a;

a--;

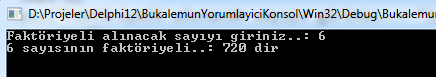
}

yaz(“Sayının faktöriyeli..: ", sonuç, " dir");

}

}

*Ekran Çıktısı:*



Faktöriyel örneğinde klavyeden girilen sayının 1 den büyük olduğu durumlar için döngü oluşturuluyor. Döngünün her elemanı sonuç değeriyle çarpılarak tekrar sonuç değişkenine atanıyor. Döngü bitince klavyeden girilen sayıdan 1’e kadar tüm sayılar çarpılarak sonuç değişkenine atanıyor.

**Önemli Not:** Yukarıdaki **tekrarla** döngüsü örneklerinde önce şart kontrol edilir sonrasında şart doğru ise kodlar çalıştırılır. Önce kod çalışıp sonrasında şart kontrol edilsin istenirse kodlama aşağıdaki gibi yapılmalıdır.

*Kullanım:*

tekrarla

{

// kodlar

} *şart* ise;

*Örnek 1: 15den 10’a kadar sayıları ekrana yazan program*

sınıf sayıyaz

{

fonksiyon başla()

{

sayı a = 15;

tekrarla

{

yazA(a);

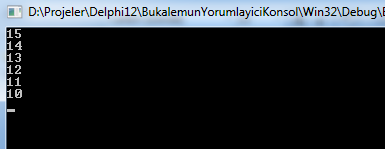
a--;

}a >= 10 ise;

}

}

*Ekran Çıktısı:*



Örnekte öncelikle kodlar çalıştırılıyor sonrasında şart kontrol ediliyor. Bunun kullanım amacı kodların en az bir defa çalışması istendiği durumlar olabilir.

*Örnek 2: 1 den 10'a kadar olan sayıların toplamı*

sınıf sayıtoplam

{

fonksiyon başla()

{

sayı toplam = 0;

sayı i = 1;

tekrarla i <= 10 ise

{

toplam += i;

i++;

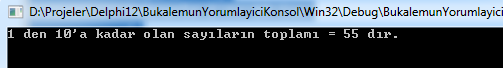
}

yazA("1 den 10'a kadar olan sayıların toplamı = ", toplam, " dır.");

}

}

*Ekran Çıktısı:*



*Örnek 3: Şifre kontrol programı*

sınıf şifrekontrol

{

fonksiyon başla()

{

metin şifre;

tekrarla

{

yaz("Programın açılış şifresini giriniz..: ");

şifre = oku();

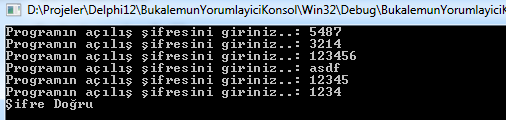
} şifre != "1234" ise;

yaz("Şifre Doğru");

}

}

*Ekran Çıktısı:*



Örnekte şifre girişi yapılmadan şifrenin doğruluk kontrolü yapılamayacağı için öncelikle şifre girişi yapılıyor sonrasında kontrol ediliyor. Şifre değişkeninin tekrarla döngüsü dışında tanımlanmasının nedeni kontrol işleminin kod bloğunun dışında yapılmasıdır.

**bitir, devam**

Döngüler ile kullanılabilen 2 adet faydalı anahtar kelime vardır. Bunlar bitir ve devam. Bitir döngüyü sonlandırmak için kullanılır. Devam ise döngünün sonraki adıma geçmesini sağlar.

*Örnek:*

sınıf döngüler

{

fonksiyon başla()

{

sayı a = 0;

tekrarla a < 20 ise

{

a++;

eğer a >= 5 ve a <= 10 ise devam;

eğer a == 15 ise bitir;

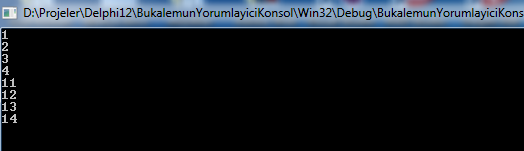
yazA(a);

}

}

}

*Ekran Çıktısı:*



Örnekte a değişkeninin değeri 5 ile 10 arasında ise sonraki adıma geçiliyor. Yani devam komutundan sonraki kodlar çalıştırılmadan döngünün bir sonraki adımına geçiliyor. A değişkeninin değeri 15 olduğunda ise döngü tamamen bitiriliyor.

**Fonksiyonlar**

Fonksiyonlar bukalemun uygulamalarında belirli görevleri yerine getirmek için yazılan kodlardır. Fonksiyon kullanmanın amacı, bazı program parçalarına ana program içerisinde birden fazla yerde ihtiyaç duyulabilir. Bu gibi durumda kodu tekrar tekrar yazmaktansa fonksiyon olarak yazılır ve istenildiği kadar kullanılabilir.

**Fonksiyon tanımlama**

Bukalemun dilinde fonksiyon aşağıdaki gibi tanımlanır.

fonksiyon fonksiyon\_adı(değişken\_türü değişken\_adı…)

{

//kodlar

}

Not: Fonksiyonun parametresi olmayacak ise fonksiyon adından sonra parantez açılır ve kapatılır.

*Örnek 1:*

sınıf fonksiyondenemesi

{

fonksiyon başla()

{

merhaba\_yaz(); // fonksiyonun kullanıldığı yer

}

// fonksiyonun tanımlandığı yer

fonksiyon merhaba\_yaz()

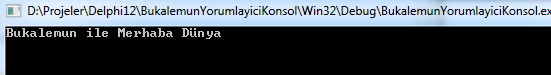
{

yazA("Bukalemun ile Merhaba Dünya");

}

}

*Ekran Çıktısı:*



Örnek 1 de parametresi olmayan merhaba\_yaz adında bir fonksiyon tanımlanıyor. Fonksiyonun herhangi bir parametresi olmadığı için fonksiyon adından sonra parantez açılıp kapanıyor. Fonksiyonu kullanmak için ise fonksiyon adını yazıp parantez açıp kapatmak yeterli.

*Örnek 2:*

sınıf kare\_fonksiyonu

{

fonksiyon başla()

{

kare\_yaz(3);

kare\_yaz(5);

}

fonksiyon kare\_yaz(sayı a)

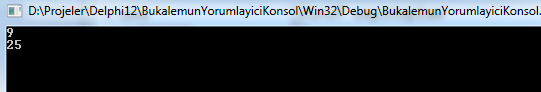
{

yazA(a\*\*2);

}

}

*Ekran Çıktısı:*



Örnek 2 de kare\_yaz adında bir fonksiyon tanımlanıyor. Bu fonksiyonun **sayı** türünde **a** adında bir parametresi var. Fonksiyon içinde bu **a** değişkeninin karesi alınıyor ve ekrana yazdırılıyor. Fonksiyonu çağırırken fonksiyon adı yazılarak parametrelere istenilen değer veriliyor.

**Geriye değer döndüren ve döndürmeyen fonksiyonlar**

Fonksiyonlar yazılış amaçlarına göre yaptığı işlemin sonucunu çağrıldığı yere gönderebilir veya kendisi ilgili yerlere bildirebilir.

*Örnek:*

|  |  |
| --- | --- |
| sınıf fonksiyonlar  {  fonksiyon başla()  {  sayı s = kare(4);  yazA(s);    küp(3);  }    fonksiyon kare(sayı a)  {  gönder a \* a;  }    fonksiyon küp(sayı a)  {  yazA(a \* a \* a);  }  } | Yandaki örnekte **kare** ve **küp** adında 2 fonksiyon tanımlanmıştır.  kare fonksiyonu yaptığı işlemin sonucunu çağrıldığı yere geri gönderiyor. **sayı s = kare(4);** satırı ile de geri gönderilen değer **s** değişkenine alınıyor. Böylece fonksiyondan gelen sonuç farklı işlemlerde kullanılabilir. Örnekte ise ekrana yazdırılıyor.  küp fonksiyonu ise işlemin sonucunu doğrudan ekrana yazdırıyor. O yüzden fonksiyonu kullanırken sadece **küp(3);** yazıyoruz. Herhangi bir değer göndermediği için değişkene değer ataması şeklinde kullanmıyoruz. |

**Fonksiyonlarla ilgili örnekler**

*Örnek:* Daire alan ve çevre heaplaması

sınıf fonksiyonlar

{

fonksiyon başla()

{

sayı r = 7;

vsayı çevre = 2 \* pi\_sayısı() \* r;

vsayı alan = pi\_sayısı() \* r \* r;

yaz("Dairenin çevresi: ", çevre, " alanı: ", alan);

}

fonksiyon pi\_sayısı()

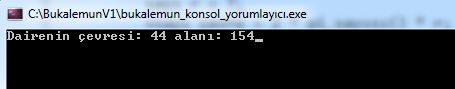
{

gönder 22.0 / 7;

}

}

*Ekran Çıktısı:*



Örnekte sabit olan pi sayısı bir fonksiyona hesaplatılarak (22/7) alınıyor.

*Örnek:* Aşağıdaki denklem için bilinmeyenleri parametre olarak alıp işlem sonucunu ekrana yazan program.

sınıf fonksiyonlar

{

fonksiyon başla()

{

hesapla(4, 1);

hesapla(6, 2);

}

fonksiyon hesapla(sayı x, sayı y)

{

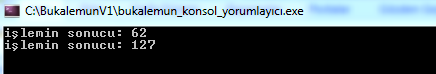
sayı sonuç = 3 \* x \*\* 2 + 5 \* y + 9;

yazA("işlemin sonucu: ", sonuç);

}

}

*Ekran Çıktısı:*



Örnekte fonksiyona parametre olarak gönderilen 4 sayısı x değişkenine, 1 sayısı y değişkenine atanır ve işlem hesaplanarak sonuç ekrana yazdırılır. Aynı işlem 6 ve 2 sayıları için tekrarlanır.

**Dahili fonksiyonlar**

Bukalemun içerisinde daha önceden tanımlı bazı fonksiyonlar bulunur. Bu fonksiyonları kullanarak uygulamalarınıza işlevsellik kazandırabilirsiniz.

**uzunluk**

Metinsel verilerin veya değişkenlerin kaç karakterden oluştuğunu döndürür.

*Kullanımı:*

sayı a = uzunluk(metinsel değişken veya veri);

*Örnek:*

sınıf fonksiyonlar

{

fonksiyon başla()

{

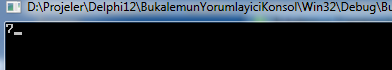
sayı b = uzunluk("Bilişim");

yaz(b);

}

}

*Ekran Çıktısı:*



Örnekte 7 karakter uzunluğundaki “Bilişim” kelimesi uzunluk fonksiyonuyla hesaplanarak b değişkenine atanıyor. Sonrasında b değişkeninin değeri ekrana yazılıyor.

**rastgele**

Rastgele sayı üretmek için kullanılan fonksiyondur. 0 (sıfır) ile parametre olarak verilen maksimum değer arasında rastgele sayı üreterek geri gönderir.

*Kullanımı:*

sayı x = rastgele(maksimum değer);

maksimum değer: Rastgele üretilecek olan sayı 0 (sıfır) ile maksimum değer arasında olacaktır. 0 dahil maksimum değer dahil değildir.

*Örnek 1*: 10 adet rastgele sayı üretip ekrana yazan program.

sınıf fonksiyonlar

{

fonksiyon başla()

{

say (sayı i: 1...10)

{

sayı b = rastgele(50);

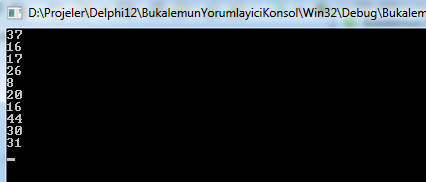
yazA(b);

}

}

}

*Ekran Çıktısı:*



Örnekte 1 den 10’a kadar döngü oluşturuluyor. Döngünün her adımında 0 ile 50 arasında rastgele bir sayı üretilerek ekrana yazdırılıyor.

*Örnek 2:* 20 ile 30 arasında rastgele 20 adet sayı üreten program.

sınıf fonksiyonlar

{

fonksiyon başla()

{

say(sayı x:1...20)

{

sayı a = rastgele(10) + 20;

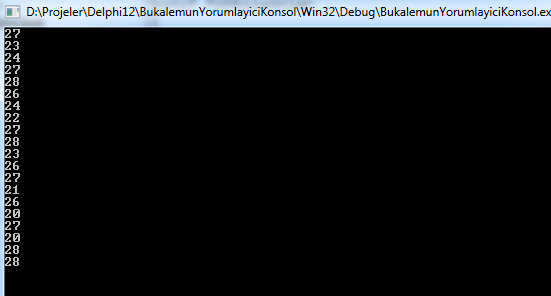
yazA(a);

}

}

}

*Ekran Çıktısı:*



**rastgeleV**

0 ile 1 arasında rastgele virgüllü sayı üretmek için kullanılır.

Kullanımı:

vsayı a = rastgeleV();

Örnek:

sınıf fonksiyonlar

{

fonksiyon başla()

{

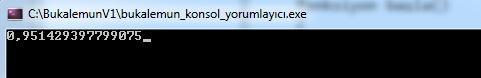
vsayı a = rastgeleV();

yaz(a);

}

}

Ekran Çıktısı:



**ondalık**

Parametre olarak verilen virgüllü sayının tam kısmını atarak virgüllü kısmını geri gönderir.

Kullanımı:

vsayı a = ondalık(virgüllü sayı veya değişken);

**yuvarla**

Virgüllü sayıları yuvarlamak için kullanılır. Buçuk ve üzeri sayılar yukarı, diğer sayılar aşağı yuvarlanır.

Kullanımı:

sayı a = yuvarla(virgüllü sayı veya değişken);

Örnek:

sınıf fonksiyonlar

{

fonksiyon başla()

{

vsayı sayı1 = 60.15;

vsayı sayı2 = 70.50;

vsayı sayı3 = 80.65;

yazA(sayı1, " yuvarla -> ", yuvarla(sayı1));

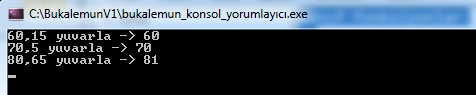
yazA(sayı2, " yuvarla -> ", yuvarla(sayı2));

yazA(sayı3, " yuvarla -> ", yuvarla(sayı3));

}

}

Ekran Çıktısı:



**tamsayı**

Virgüllü sayıların tam kısmını almak için kullanılır.

Kullanımı:

sayı a = tamsayı(virgüllü sayı veya değişken);

Örnek:

sınıf fonksiyonlar

{

fonksiyon başla()

{

vsayı sayı1 = 60.15;

vsayı sayı2 = 70.50;

vsayı sayı3 = 80.65;

yazA(sayı1, " tamsayı -> ", tamsayı(sayı1));

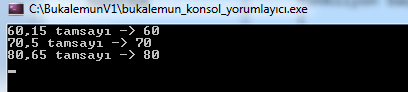
yazA(sayı2, " tamsayı -> ", tamsayı(sayı2));

yazA(sayı3, " tamsayı -> ", tamsayı(sayı3));

}

}

Ekran Çıktısı:



**tarihsaat**

Sistemin tarih ve saatini geri gönderir.

Kullanımı:

metin m = tarihsaat();

Örnek:

sınıf fonksiyonlar

{

fonksiyon başla()

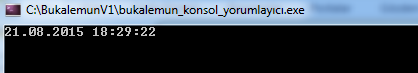
{

yaz(tarihsaat());

}

}

Ekran Çıktısı:



**ascii**

Parametre olarak gönderilen karakterin ascii kodunu geri gönderir.

Kullanımı:

sayı a = ascii(karakter veya karakter değişkeni);

Örnek:

sınıf fonksiyonlar

{

fonksiyon başla()

{

sayı x = ascii('A');

yaz(x);

}

}

Ekran Çıktısı:

40.png

**karakter**

Parametre olarak gönderilen sayının karakter karşılığını geri gönderir.

Kullanımı:

karakter k = karakter(sayısal veri veya değişken);

Örnek:

sınıf fonksiyonlar

{

fonksiyon başla()

{

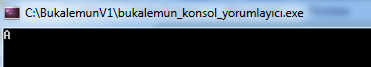
karakter x = karakter(65);

yaz(x);

}

}

Ekran Çıktısı:



Örnek 2: Büyük küçük harf

sınıf fonksiyonlar

{

fonksiyon başla()

{

say (sayı a: 65..75)

{

karakter k = karakter(a);

yaz(k);

karakter m = karakter(a+32);

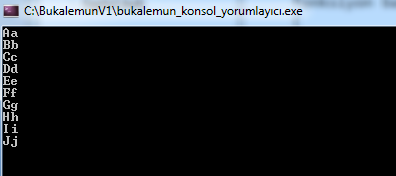
yazA(m);

}

}

}

*Ekran Çıktısı:*



**metinböl**

metin içerisinden istenen sayıda karakter almak için kullanılır.

Kullanımı:

metin m = metinböl(metinsel ifade veya değişken, başlangıç);

metin m = metinböl(metinsel ifade veya değişken, başlangıç, adet);

*Örnek:*

sınıf fonksiyonlar

{

fonksiyon başla()

{

metin m = "abcçdef";

metin m1 = metinböl(m, 1, 3);

metin m2 = metinböl(m, 2);

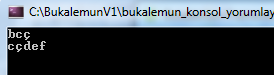
yazA(m1);

yazA(m2);

}

}

*Ekran Çıktısı:*



**karekök**

Parametre olarak verilen sayının karekökünü alarak geri gönderir.

Kullanımı:

vsayı a = karekök(karekökü alınacak sayı);

*Örnek:*

sınıf fonksiyonlar

{

fonksiyon başla()

{

yaz("Karekökü alınacak sayıyı giriniz ");

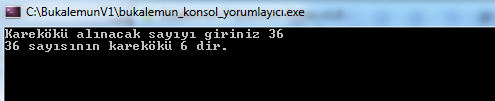
sayı s = oku();

yaz(s, " sayısının karekökü ", karekök(s), " dir.");

}

}

*Ekran Çıktısı:*



**karakteral**

Metin içerisinden 1 karakter almak için kullanılır.

Kullanımı:

karakter k = karakteral(metinsel ifade veya değişken, indeks);

Örnek:

sınıf fonksiyonlar

{

fonksiyon başla()

{

yaz("Bir metin giriniz..: ");

metin m = oku();

sayı u = uzunluk(m);

say (sayı s:0..u)

{

karakter k = karakteral(m,s);

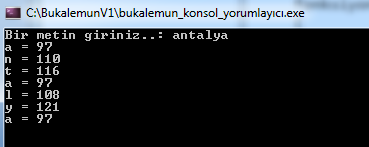
yazA(k, " = ", ascii(k));

}

}

}

*Ekran Çıktısı:*



**Tür Dönüştürme Fonksiyonları**

Bukalemun tür dönüşüm işlemlerini otomatik olarak yapar ancak istenirse aşağıdaki fonksiyonlarda kullanılabilir.

**metin**

Parametre olarak verilen ifadeyi metin türüne dönüştürür.

Kullanımı

metin m = metin(1500);

**sayı**

Parametre olarak verilen ifadeyi sayı türüne dönüştürür.

Kullanımı:

sayı s = sayı(“1500”);

Örnek:

|  |  |
| --- | --- |
| sınıf fonksiyonlar  {  fonksiyon başla()  {  sayı a = 9;  eğer a == "9" ise  yaz("şart doğru");  değilse yaz("şart yanlış");  }  } | sınıf fonksiyonlar  {  fonksiyon başla()  {  sayı a = 9;  eğer a == sayı("9") ise  yaz("şart doğru");  değilse yaz("şart yanlış");  }  } |
| 47.png | 48.png |
| Metinsel ve sayısal ifadeler karşılaştırıldığında eşit olmadıkları görülür. | Sayısal bir ifade ile, önce metinsel olup sayı türüne çevrilen bir ifade karşılaştırıldığında ise karşılaştırma işlemi doğru bir şekilde yapılır. |

**vsayı**

Parametre olarak verilen ifadeyi vsayı türüne dönüştürür.

Kullanımı:

vsayı v = vsayı(“150,5”);

Örnek:

sınıf fonksiyonlar

{

fonksiyon başla()

{

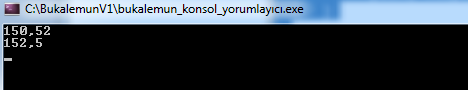
yazA( "150,5" + 2 );

yazA( vsayı("150,5") + 2 );

}

}

Ekran Çıktısı:



Örnekte **yazA( "150,5" + 2 );** satırı ile metinsel ve sayısal ifade toplanamayacağından dolayı ekrana yan yana yazılıyor. **yazA( vsayı("150,5") + 2 );** satırı ile ise metinsel ifade önce virgüllü sayıya çevrildiği için 2 ile toplanıp öyle ekrana yazılıyor.