Nesneye Dayalı Programlama

Sakarya Üniversitesi
Bilgisayar ve Bilişim Bilimleri Fakültesi
Bilgisayar Mühendisliği

Prof. Dr. Ümit KOCABIÇAK
Prof. Dr. Cemil Öz
Öğr. Gör. Nevzat TAŞBAŞI
Öğr. Gör. Sinan İLYAS

2017
2. HAFTA

DEĞİŞKENLER

Değişkenler bir bilginin bellekteki konumunu temsil eden sembolik isimlerdir.
Program çalıştırıldığında değişken ve bu değişkenin türüne göre bellekte yer ayrılır.
Değişken isimlendirmesinde bazı kurallara uyulmalıdır.

1. Değişken mutlaka bir harf ile başlamalıdır. Rakam ve alt çizgi (_) karakterleri de değişken isminde kullanılabilir.
2. C# komutları ve fonksiyonları değişken adı olarak kullanılamaz.
3. Değişken adı arasında boşluk bulundurmamalıdır.
4. C#'ta değişken adlandırmada küçük-büyük harf ayırımı vardır. **AD**, **ad** ve **Ad** farklı değişkenleri ifade etmektedir.
5. C#'ta Türkçe karakterler isimlendirmede kullanılabilir, ancak tavsiye edilmez.

Geçerli değişken isimleri : başlamaZamanı, ad_soyad, x5
Geçersiz değişken isimleri : 3x, while, data?in, data in

DEĞİŞKEN TANIMLAMA

Tip adı ile değişken tanımlama

```
int sayi1;
int sayi2 = 10;
```

var anahtar kelimesi ile değişken tanımlama

```
var sayi1 = 10;     // int
var sayi2 = 10.0;   // double
var sayi3 = 10.0f; // float

var sayi4; // HATA: derleyici tipi bilemez
```

SABİT TANIMLAMA

```
const int yukseklik = 10;
const string dil = “Visual C#”;

const int genislik; // HATA: değer belirtilmemiş
```

VALUE VE REFERENCE TİPLER

Value tipler	bool, byte, sbyte, char, decimal, double, float, int, uint, long, ulong, short, ushort, struct ve enum
Reference tipler	class, interface, delegate, object ve string

int x=5;

Stack	Heap
x=5	

object y=1;

Stack	Heap
x=5	
&y	y=1

BOXING VE UNBOXING

Boxing: Stack alanından Heap alanına taşıma

```
object x;
int i=10;
x=i;
```

Unboxing: Heap alanından Stack alanına taşıma

```
object x;
int i=10, j=20;
x=j;
i=(int)x;
```

VERİ TİPLERİ

STANDART VERİ TİPLERİ

	Veri Tipi	Byte	Aralık
Tamsayı	byte	1	0 ... 255
	short	2	-32,768 ... 32,767
	int	4	-2,147,483,648 ... 2,147,483,647
	long	8	-9,223,372,036,854,775,808 ... 9,223,372,036,854,775,807
Reel Sayı	float	4	-3.4028235E38 ... 3.4028235E38
	double	8	-1.7976931348623 1E308 ... 1.7976931348623 1E308
	decimal	16	$-79,228 \times 10^{24}$ ... $79,228 \times 10^{24}$
Karakter	char	2	0 ... 65,535
	string	2	0 ... 2 milyar karakter
Lojik	bool	2	True veya False (False durumunda 0 değeri döndürülür)
Tarih/zaman	DateTime	8	Tarih ve zaman
	TimeSpan	8	Tarih aritmetik işlemlerinde (mesela 2 tarih arasındaki farkı bulmak için) kullanılır.
Genel	object	4	Harhangi bir tip. C# nesnelerini (buton veya form gibi) tanımlamak için kullanılı.

IMPLICIT VE EXPLICIT TİP DÖNÜŞÜMLERİ

```
int sayi1 = 10;
float sayi2 = sayi1;

float sayi3 = 20.0f;
int sayi4 = sayi3; // HATA: float int'ten büyüktür

float sayi5 = 30.0f;
int sayi6 = (int)sayi5;
```

byte VERİ TİPİ

```
byte a = 1;
byte b = 2;
byte c = a + b; // Derleme hatası
```

Yukarıdaki kod derleme hatası verecektir çünkü byte tipi sayı olarak kabul edilmez, 8 bitlik grup olarak kabul edilir. Bu yüzden de toplama işlemi yapılmadan önce otomatik olarak int tipine dönüştürülür. Bu sebeple de a+b işleminin sonucu int olur. Sonucu byte tipindeki bir değişkene atayabilmek için tip dönüşümü yapılmalıdır.

```
byte a = 1;
byte b = 2;
byte c = (byte)(a + b);
```

İŞLEM SONUÇ TİPLERİ

```
int sayi1 = 5, sayi2 = 2;
float sonuc = sayi1 / sayi2; // sonuc=2.0f
```

İşleme giren tüm elemanlar int olduğu için sonucun da int olacağı kabul edilir ve işlem sonucundaki virgülden sonraki kısım atılır.

```
int sayi1 = 5, sayi2 = 2;
float sonuc = (float)sayi1 / sayi2; // sonuc=2.5f
```

İşleme giren elemanlardan biri float biri int olduğu için sonucun float (büyük olan) olacağı kabul edilir.

TİP DÖNÜŞÜMÜ: Parse

```
string sayi1 = “1234”;
int x = int.Parse(sayi1);

string sayi2=“12.345”;
int y = int.Parse(sayi2); // çalışma zamanı hatası

long s = long.Parse(“123456”);
```

TİP DÖNÜŞÜMÜ: Convert

```
int sayi = Convert.ToInt32(“123”);
```

Dönüşüm metodu	Açıklama
Convert.ToBoolean (ifade)	Sayısal ifadeyi **bool** tipine dönüştürür.
Convert.ToChar (ifade)	Tek karakteri **char** tipine dönüştürür.
Convert.DateTime (ifade)	Geçeri bir tarih veya zamanı **DateTime** tipine dönüştürür.
Convert.ToSingle (ifade)	Bir ifadeyi **float** tipine dönüştürür.
Convert.ToDouble (ifade)	İfadeyi **double** tipine dönüştürür.
Convert.ToInt16 (ifade)	Bir ifadeyi **short** tipine dönüştürür.
Convert.ToInt32 (ifade)	Bir ifadeyi **int** tipine dönüştürür.
Convert.ToInt64 (ifade)	Bir ifadeyi **long** tipine dönüştürür.

KONSOLA YAZDIRMA

```
var ad = “Ahmet”;
var soyad = “Öztürk”;

Console.WriteLine(ad);
Console.WriteLine(soyad);
```

Ekran Çıktısı

```
Ahmet
Öztürk
```

KONSOLA YAZDIRMA

```
var sayi1 = 100;
var sayi2 = 200;

Console.Write(sayi1);
Console.Write(sayi2);
```

Ekran Çıktısı

```
100200
```

ARİTMETİK OPERATÖRLER

İŞLEM	OPERATÖR	ÖRNEK
Toplama	+	x + 3
Çıkarma	-	x - 3
Çarpma	*	x * 3
Bölme	/	x / 3
Mod Alma	%	x % 3

KARŞILAŞTIRMA OPERATÖRLERİ

İŞLEM	OPERATÖR	ÖRNEK
Büyüktür	>	x > 3
Küçüktür	<	x < 3
Büyük veya eşittir	>=	x >= 3
Küçük veya eşittir	<=	x <= 3
Eşittir	==	x == 3
Farklıdır	!=	x != 3

MANTIKSAL OPERATÖRLER

İŞLEM	OPERATÖR	ÖRNEK
Ve	&&	(x > 5) && (y < 4)
Veya	||	(x > 5) || (y < 4)
Değil	!	!(x > 5)

BİT İŞLEM OPERATÖRLERİ

İŞLEM	OPERATÖR	ÖRNEK
Ve	&	5 & 3 (sonuç = 1)
Veya	|	5 | 3 (sonuç = 7)
Exor	^	5 ^ 3 (sonuç = 6)
Değil	~	~5 (sonuç = -6)

~5 ifadesinin sonucunun -6 çıkmasının sebebi işaretli tamsayıların ikiye tümleyen formunda saklanmasıdır.

ATAMA OPERATÖRLERİ

İŞLEM	OPERATÖR	ÖRNEK
Değer atama	=	Sayi = 5
Toplayarak atama	+=	Sayi += 5
Çıkararak atama	-=	Sayi -= 5
Çarparak atama	*=	Sayi *= 5
Bölerek atama	/=	Sayi /= 5
Değeri 1 arttırma	++	Sayi++ veya ++Sayi
Değeri 1 azaltma	--	Sayi-- veya --Sayi

DİZİLER

```
int [] dizi = new  int[10];

int [] dizi = {1,2,3};

int [] dizi;
dizi=new int[10];

string [] kelimeler = {"Sakarya" , "Üniversitesi"};

char [] s = {'M','e','r','h','a','b','a'};
// string s = "Merhaba"; ile aynı değil

int sayi = Convert.ToInt32(elemanSayisiTextBox.Text);
int[] dizi = new int[sayi];
// Diziler dinamik olarak başlatılabilir
```

DİZİLER

```
string [] renkler = {"Kırmızı", "Yeşil", "Mavi", "Sarı"};

for (var i = 0; i < 4; i++)
{
      var renk = renkler[i];
      System.Console.WriteLine(renk);
}
```

Kod parçası çalıştırıldığında ekran çıktısı aşağıdaki gibi olur.

```
Kırmızı
Yeşil
Mavi
Sarı
```

ÇOK BOYUTLU DİZİLER

```
int[,] dizi1 = new int[3, 3];

int[,] dizi2 = {{1, 2}, {3, 4}, {10, 11}};

int[,,] dizi3 = new int[5, 5, 5];
```

veya

```
var dizi1 = new int[3, 3];

var dizi2 = {{1, 2}, {3, 4}, {10, 11}};

var dizi3 = new int[5, 5, 5];
```

ÇOK BOYUTLU DİZİLER

```
var matris = {
    {10, 12, 20, 22},
    {17, 22, 19, 13},
    {10, 12, 20, 22},
    {17, 22, 19, 13}
};

Console.WriteLine("Matris );

for (var i = 0; i < 4; i++)
{
    for (var j = 0; j < 4; j++)
    {
        Console.Write(" " + matris[i, j] + " ");
    }

    Console.WriteLine();
}
```

Ekran Çıktısı

```
Matris
10 12 20 22
17 22 19 13
10 12 20 22
17 22 19 13
```

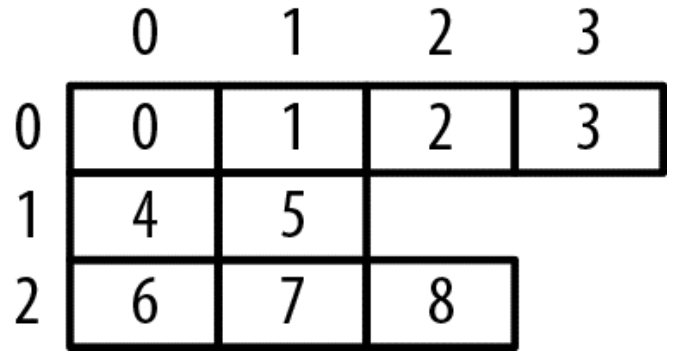
DÜZENSİZ DİZİLER (JAGGED ARRAYS)

Elemanları dizi olan dizilere düzensiz dizi adı verilir.

Düzensiz dizilerin her bir elemanı birbirinden bağımsız başka bir dizidir.

Her bir eleman birbirinden bağımsız olduğu için, eleman sayısı da birbirinden farklı olabilir.

```
int[][] jagged = new int[3][];
jagged[0] = new int[4] {0, 1, 2, 3};
jagged[1] = new int[2] {4, 5};
jagged[2] = new int[3] {6, 7, 8};
```



KARAR YAPILARI: if

```
if (ogrNot >= 40) sonuc = “geçti”;

if (ogrNot >= 40) sonuc = “geçti”;
else sonuc = “kaldı”;

if (ogrNot < 40) harfNotu = “FF”;
else if (ogrNot < 50) harfNotu = “DD”;
else if (ogrNot < 60) harfNotu = “DC”;
else if (ogrNot < 70) harfNotu = “CC”;
else if (ogrNot < 80) harfNotu = “CB”;
else if (ogrNot < 90) harfNotu = “BB”;
else if (ogrNot < 95) harfNotu = “BA”;
else harfNotu = “AA”;
```

KARAR YAPILARI: switch

```
switch (gun)
{
      case 1:
            ad = "pazartesi";
            break;
      case 2:
            ad = "salı";
            break;
      case 3:
            ad = "çarşamba";
            break;
      case 4:
            ad = "perşembe";
            break;
      case 5:
            ad = "cuma";
            break;
      case 6:
            ad = "cumartesi";
            break;
      case 7:
            ad = "pazar";
            break;
      default:
            ad = "HATA";
            break;
}
```

DÖNGÜLER: for

```
var sayi = 0;

for (var i = 0; i < 5; i++)
{
      sayi++;
}
```

Kod parçası çalıştırıldığında sayi = 5 olur.

DÖNGÜLER: for

```
var sayi = 0;

for (var i = 0; i < 5; i++)
{
      if (i == 2) break;
      sayi++;
}
```

Kod parçası çalıştırıldığında sayi = 3 olur.

DÖNGÜLER: for

```
var sayi = 0;

for (var i = 0; i < 5; i++)
{
      if (i == 2) continue;
      sayi++;
}
```

Kod parçası çalıştırıldığında sayi = 4 olur.

DÖNGÜLER: while

```
var sayi = 0;

while (sayi < 5)
{
      sayi++;
}
```

Kod parçası çalıştırıldığında sayi = 5 olur.

DÖNGÜLER: while

```
var sayi = 100;

while (sayi < 5)
{
      sayi++;
}
```

Kod parçası çalıştırıldığında sayi = 100 olur.

DÖNGÜLER: do-while

```
var sayi = 0;

do
{
      sayi++;
} while (sayi < 5);
```

Kod parçası çalıştırıldığında sayi = 5 olur.

DÖNGÜLER: do-while

```
var sayi = 100;

do
{
      sayi++;
} while (sayi < 5);
```

Kod parçası çalıştırıldığında sayi = 101 olur.

DÖNGÜLER: foreach

```
var dizi = new int[] { 1, 2, 3, 4 };

foreach (var sayi in dizi)
{
      Console.WriteLine(sayi);
}
```

Kod parçası çalıştırıldığında ekran çıktısı aşağıdaki gibi olur.

```
1
2
3
4
```