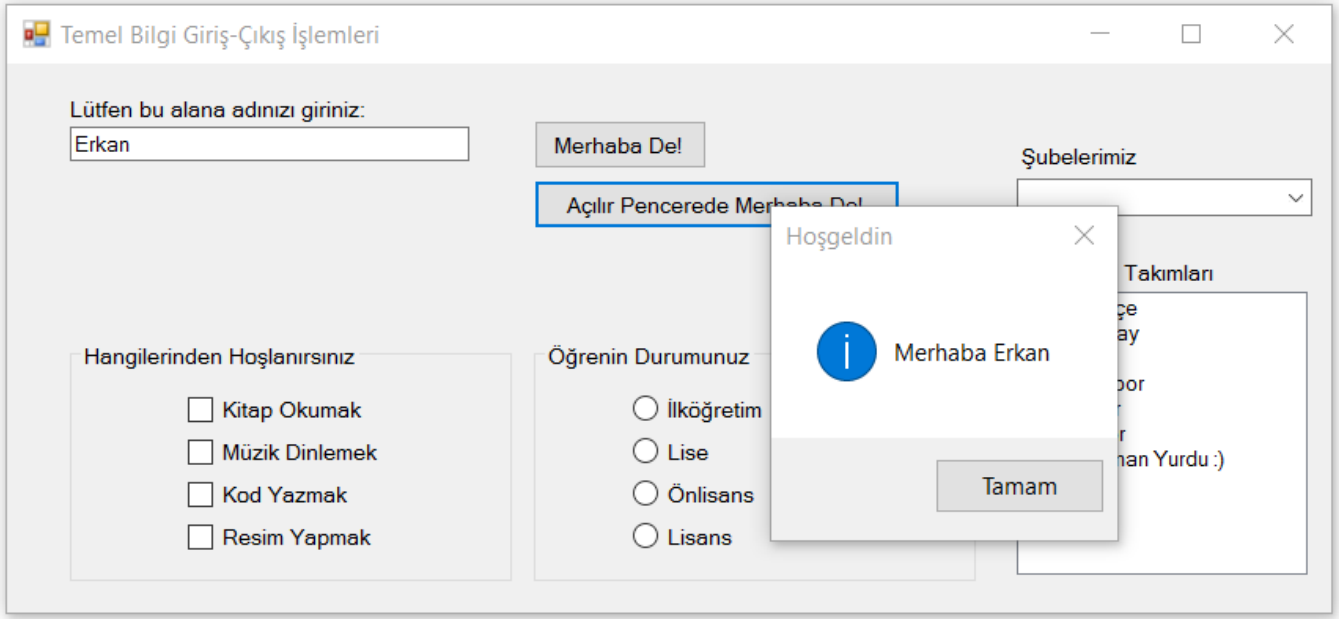
Değişken Tanımlama ve Veri Türleri*

*https://tr.wikibooks.org/wiki/C_Sharp_Programlama_Dili kitabından derlenmiştir.

Öğr. Gör. Erkan HÜRNALI

Temel Bilgi Giriş-Çıkış İşlemleri (Hatırlatma)



- Nesneler
  - Button, TextBox, CheckBox, RadioButton, Panel …
- Temel İşlemler
  - Bilgi Girişi, Bilgi Çıkışı
  - Mesaj Pencereleri
- Olay Tabanlı Programlama
  - Click, KeyPress, MouseEnter, MouseMove, MouseLeave …
- Temel VS Panelleri
  - Toolbox, Properties, Solutuin Explorer …

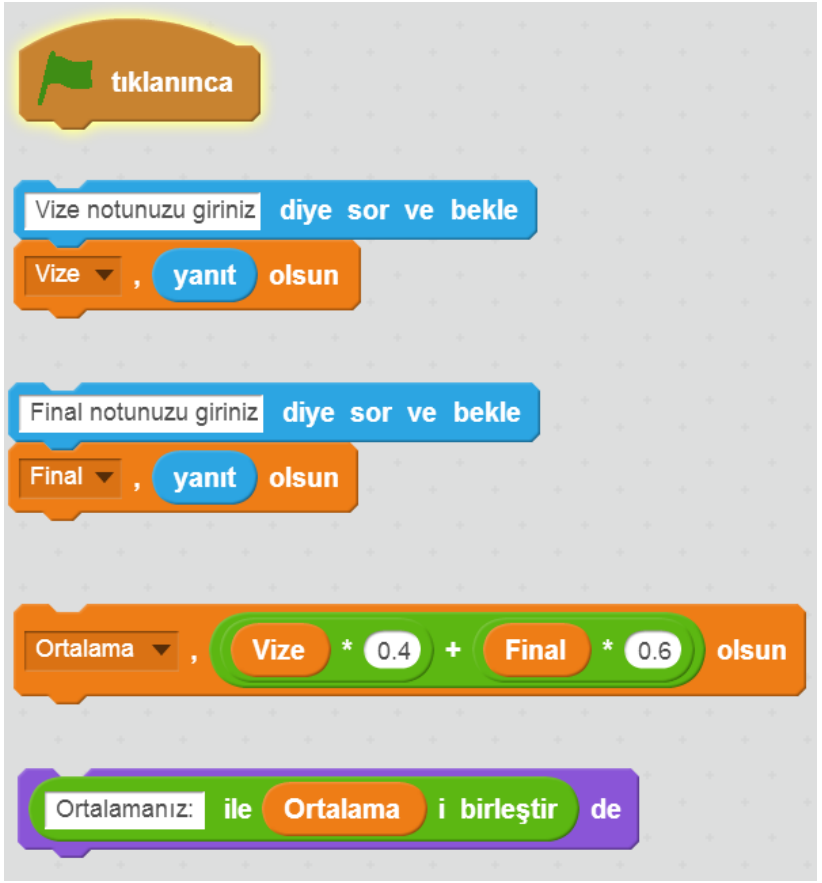
Saklama İhtiyacı

?

İçerik

- Değişken Tanımlama
- Değer Atama
- Veri Türleri
- Değişkenleri Program İçinde Kullanma
- Değişkenlerin Ömürleri
- Değişken Adlandırma Kuralları
- Sık Yapılan Hatalar
- Sabit Değişkenler ☺

Değişken



Scratch programındaki bazı özel yapılar, kullanıcının Vize ve Final olarak girmiş olduğu değerleri (yanıtlarını) saklayarak Ortalama hesaplama formülüne kadar iletilmelerini sağlamışlardı.

Değişken Tanımlama

Çoğu programlama dilinde değişkenler kullanılmaya başlanmadan önce tanımlanırlar. Aşağıdaki şekli inceleyiniz.

Yukarıdaki şekilde C#'ta değişken tanımlamanın nasıl yapıldığı anlatılmıştır. Değişken türü bellekte ayrılan gözeneğin büyüklüğünü belirtmemizi sağlar. Değişken adı da bu gözeneğe verdiğimiz adı belirtir. Doğal olarak bu gözenekteki veriye erişmek istediğimizde veya bu gözenekteki veriyi değiştirmek istediğimizde bu adı kullanacağız. Yukarıdaki şekilde -2,147,483,648 ile 2,147,483,647 arasında (sınırlar dâhil) bir değer tutabilen ve adı "ad" olan bir bellek gözeneği oluşturduk.

Değişkenlere Değer Atama

Çoğu programlama dilinde değişkenler tanımlandıktan sonra direkt olarak programda kullanılabilirler. Ancak C#'ta değişkeni tanımladıktan sonra ayrıca bir de ilk değer atamak zorundayız. Aksi bir durumda değişkeni programımız içinde kullanamayız. Değişkenlere değer atama şöyle yapılır:

```
ad=5;
```

Burada ad değişkenine 5 değerini atadık. Bu en basit değer atama yöntemidir. Ayrıca şunlar da mümkündür:

```
int a=5;
int b, c, d, e;
int f=10, g, m=70;
```

Birinci satırda tanımlama ve değer vermeyi aynı satırda yaptık. İkincisinde aynı türden birden fazla değişken tanımladık. Üçüncü satırda ise tanımladığımız değişkenlerin bazılarına değer verirken bazılarına vermedik.

Sayısal Veri Türleri

Tür	Boyut	Kapasite	Örnek
byte	1 bayt	0, ..., 255 (tam sayı)	`byte a=5;`
sbyte	1 bayt	-128, ..., 127 (tam sayı)	`sbyte a=5;`
short	2 bayt	-32768, ..., 32767 (tam sayı)	`short a=5;`
ushort	2 bayt	0, ..., 65535 (tam sayı)	`ushort a=5;`
int	4 bayt	-2147483648, ..., 2147483647 (tam sayı)	`int a=5;`
uint	4 bayt	0, ..., 4294967295 (tam sayı)	`uint a=5;`
long	8 bayt	-9223372036854775808, ..., 9223372036854775807 (tam sayı)	`long a=5;`
ulong	8 bayt	0, ..., 18446744073709551615 (tam sayı)	`ulong a=5;`
float	4 bayt	$\pm 1.5*10^{-45}$, ..., $\pm 3.4*10^{38}$ (reel sayı)	`float a=5F;` veya `float a=5f;`
double	8 bayt	$\pm 5.0*10^{-324}$, ..., $\pm 1.7*10^{308}$ (reel sayı)	`double a=5;` veya `double a=5d;` veya `double a=5D;`
decimal	16 bayt	$\pm 1.5*10^{-28}$, ..., $\pm 7.9*10^{28}$ (reel sayı)	`decimal a=5M;` veya `decimal a=5m;`

Alfasayısal Veri Türleri

Tür	Boyut	Açıklama	Örnek
char	2 bayt	Tek bir karakteri tutar.	`char a='h';`
string	Sınırsız	Metin tutar.	`string a="Ben bir zaman kaybıyım, beni boşver hocam";`

String türüne ayrıca char ve/veya string sabit ya da değişkenler + işaretiyle eklenip atanabilir. Örnekler:

```
char a='g';
string b="deneme";
string c=a+b+"Viki"+'m';
```

Diğer Bazı Veri Türleri

bool [değiştir]

Koşullu yapılarda kullanılır. Bool türünden değerlere `true`, `false` veya `2<1` gibi ifadeler örnek verilebilir. Örnekler:

```
bool b1=true;
bool b2=false;
bool b3=5>4;
```

object [değiştir]

Bu değişken türüne her türden veri atanabilir. Örnekler:

```
object a=5;
object b='k';
object c="metin";
object d=12.7f;
```

Değişkeni Programımız İçinde Kullanma

```
using System;
class degiskenler
{
   static void Main()
   {
      byte a=5;
      Console.WriteLine(a);
   }
}
```

Burada a değişkeninin değerini ekrana yazdırdık. Başka bir örnek:

```
using System;
class degiskenler
{
   static void Main()
   {
      byte a=5;
      byte b=8;
      Console.WriteLine(a+b);
   }
}
```

Bu programda iki değişkenimizin değerlerinin toplamını ekrana yazdırdık.

Değişkeni Programımız İçinde Kullanma

```
using System;
class degiskenler
{
   static void Main()
   {
      string a="Viki", b="kitap";
      Console.WriteLine(a+b);
   }
}
```

Bu programda aynı satırda iki tane `string` değişkeni tanımladık ve değer verdik. Bu değişkenlerin değerlerini `WriteLine` metoduyla ekrana yan yana yazdırdık. `WriteLine` metodu + işaretini gördüğünde sayısal değişken ve değerleri toplar, string türünden değişken ve değerleri yan yana yazar, `char` türünden değişken ve değerlerin Unicode karşılıklarını toplar. Ancak tabii ki + karakterinin ayırdığı değişken veya değerler `char` ile `string`se char karakterle string metni yan yana yazar.

Değişken Adlandırma Kuralları

Şimdiye kadar değişkenlere ad, a veya b gibi basit adlar verdik. Ancak aşağıdaki kuralları ihlal etmemek şartıyla değişkenlere istediğiniz adı verebilirsiniz.

- Değişken adları boşluk, simge içeremez.
- Değişkenler bir numerik karakterle başlayamaz.
- C#'ın diğer bütün komut, metot ve benzerlerinde olduğu gibi değişken adlarında büyük-küçük harf duyarlılığı vardır. Yani `degisken` isimli bir değişkenle `Degisken` isimli bir değişken birbirinden farklıdır.
- Değişken adları Türkçe karakterlerden(ğ,ü,ş,ö,ç,ı) oluşamaz. (Yeni versiyonlarda Türkçe karakterler kullanılabilir.)

Sık Yapılan Hatalar

C#'ta değişkenlerle ilgili sık yapılan hatalar şunlardır:

- Aynı satırda farklı türden değişkenler tanımlamaya çalışma. Örneğin aşağıdaki örnek hatalıdır:

```
int a, string b;
```

- Değişkene uygunsuz değer vermeye çalışma. Örnek:

```
int a;
a="metin";
```

Sık Yapılan Hatalar

- Değişkeni tanımlamadan ve/veya değişkene ilk değer vermeden değişkeni kullanmaya çalışma. Aşağıdaki örnekte iki değişkenin de kullanımı hatalıdır.

```
using System;
class degiskenler
{
   static void Main()
   {
      int b;
      Console.WriteLine(a);
      Console.WriteLine(b);
   }
}
```

Sık Yapılan Hatalar

- Bazı değişken türlerindeki değişkenlere değer verirken eklenmesi gereken karakteri eklememek. Örnek:

```
using System;
class degiskenler
{
   static void Main()
   {
      float a=12.5;
      Console.WriteLine(a);
   }
}
```

Sık Yapılan Hatalar

- Ondalık sayıların ondalık kısmını ayırırken nokta (.) yerine virgül (,) kullanmak. Örnek:

```
using System;
class degiskenler
{
    static void Main()
    {
        float a=12,5f;
        Console.WriteLine(a);
    }
}
```

Sık Yapılan Hatalar

- Metinsel değişkenlerle matematiksel işlem yapmaya çalışmak. Örnek:

```
using System;
class degiskenler
{
    static void Main()
    {
        string a="1", b="2";
        int c=a+b;
        Console.WriteLine(a);
    }
}
```

Sık Yapılan Hatalar

- Bir değişkeni birden fazla kez tanımlamak. Örnek:

```
using System;
class degiskenler
{
    static void Main()
    {
        string a;
        string a="deneme";
        Console.WriteLine(a);
    }
}
```

Sık Yapılan Hatalar

- Değişkenlere değer verirken yanlış şekilde değer vermek. Örnek:

```
using System;
class degiskenler
{
   static void Main()
   {
      string a=deneme;
      Console.WriteLine(a);
   }
}
```

Sabit Değişkenler

Programımızda bazen değeri hiç değişmeyecek değişkenler tanımlamak isteyebiliriz. Örneğin `pi` isimli float türünden bir değişken tanımlayıp buna 3.14 değerini verip programımızda pi sayısına ihtiyaç duyulduğunda bu değişkeni kullanabiliriz. Sabit değişkenlerin normal değişkenlerden farkı değişkeni değiştirmek istediğimizde ortaya çıkar, sabit olarak belirtilen değişkeni değiştirirsek derleyici hata verip programımızı derlemez. Bu daha çok uzun program kodlarında işe yarayabilir. Ayrıca sabit değişkenlere tanımlandığı satırda değer vermeliyiz. Herhangi bir değişkeni sabit olarak belirtmemiz için değişken türünden önce `const` anahtar sözcüğü kullanılır. Örnekler:

```
const int a = 5;
const string b ="deneme";
const char c ='s';
```

Uygulama Ödevleri