Budapesti Műszaki és Gazdaságtudományi Egyetem

Közlekedés- és Járműirányítási Tanszék

Számítástechnika I.

BMEKOKAA152

BMEKOKAA119

Infokommunikáció I.

BMEKOKAA606

Dr. Bécsi Tamás

2. előadás

Console I/O bővebben

- Lásd mintaprogram

Számábrázolásról röviden

- Tábla, jegyzeteljetek, hátha jobban figyeltek :P

Szintaktikai alapok
Alapvető típusok, ismétlés

C# típus	.NET típus	Méret (byte)	Leírás
byte	System.Byte	1	Előjel nélküli 8 bites egész szám (0..255)
char	System.Char	2	Egy Unicode karakter
bool	System.Boolean	1	Logikai típus, értéke igaz(1) vagy hamis(0)
sbyte	System.SByte	1	Előjeles 8 bites egész szám (-128..127)
short	System.Int16	2	Előjeles 16 bites egész szám (-32768..32767)
ushort	System.UInt16	2	Előjel nélküli 16 bites egész szám (0..65535)
int	System.Int32	4	Előjeles 32 bites egész szám (–2147483647.. 2147483647).
uint	System.UInt32	4	Előjel nélküli 32 bites egész szám (0..4294967295)
float	System.Single	4	Egyszeres pontosságú lebegőpontos szám
double	System.Double	8	Kétszeres pontosság lebegőpontos szám
decimal	System.Decimal	8	Fix pontosságú 28+1 jegyű szám
long	System.Int64	8	Előjeles 64 bites egész szám
ulong	System.UInt64	8	Előjel nélküli 64 bites egész szám
string	System.String	N/A	Unicode karakterek szekvenciája
object	System.Object	N/A	Minden más típus őse

Szám-Szöveg konverzió

- Típus statikus Parse függvénye:

```
public static <típus> <típus>.Parse(string s);
int i=int.Parse(”31”);
```

- Típus statikus TryParse függvénye:

```
public static bool TryParse( string s, out int result )
```

- Convert osztály:

```
k=Convert.ToInt32("13");
```

- ToString() metódus:

```
string s = k.ToString();
```

Szintaktikai alapok
Alapvető operátorok és precedencia

Kategória	Operátor	Asszociativitás
Elsődleges	() [] -> . ++ - -	Left to right
	+ - ! ~ ++ - - (type)* & sizeof	Right to left
Multiplikatív	* / %	Left to right
Additív	+ -	Left to right
Shift	<< >>	Left to right
Relációs és típusvizsgálat	< <= > >= is as	Left to right
Egyenlőség	== !=	Left to right
Bitenkénti AND	&	Left to right
Bitenkénti XOR	^	Left to right
Bitenkénti OR	|	Left to right
Logikai AND	&&	Left to right
Logikai OR	||	Left to right
Feltételes	?:	Right to left
Értékadás	= += -= *= /= %=>>= <<= &= ^= |=	Right to left
Vessző	,	Left to right

Értékadás operátor

- Az egyik legáltalánosabb művelet, amit elvégezhetünk az, hogy egy változónak értéket adunk. A C# nyelvben ezt az egyenlőségjel segítségével tehetjük meg:
- `i=2;`
  Az olyan kifejezések esetén, ahol a bal oldali érték megjelenik a jobb oldalon, pld.:
- `i=i+2;`
  tömörebb formában is írhatjuk:
- `i+=2;`
  Az értékadásnak van visszatérő értéke!

Aritmetikai operátotok

A C# nyelv kétoperandusú aritmetikai operátorai a +, -, * és /, valamint a % modulus operátor. Az egészek osztásakor a törtrészt a rendszer levágja, ezért van szükség a % modulus operátorra. Az x % y kifejezés az x/y egészosztás (egész) maradékát adja, és értéke nulla, ha x osztható y-nal.

A % operátor nem alkalmazható float és double típusú adatokra. Negatív operandusok esetén az egészosztás hányadosának csonkítása, valamint a modulus előjele gépfüggő, és ugyanez igaz az esetlegesen előforduló túlcsordulásra és alácsordulásra is.

Az egyoperandusú (unáris) + és - operátorok precedenciája a legmagasabb. A kétoperandusú + és - operátorok precedenciája kisebb a *, / és % precedenciájánál. Az aritmetikai operátorok mindig balról jobbra haladva hajtódnak végre (a precedencia figyelembevételével).

Relációs operátorok

A C# nyelv relációs operátorai: >, >=, <, <=. Ez a sorrend egyben a precedenciájuk sorrendje is. Ezeknél eggyel alacsonyabb precedenciájúak az egyenlőség operátorok: ==, !=. Egy relációs vagy logikai kifejezés értéke definíció szerint **false**, ha a kifejezés hamis, és **true**, ha igaz.

x > y	x nagyobb, mint y
x >= y	x nagyobb vagy egyenlő, mint y
x < y	x kisebb, mint y
x <= y	x kisebb vagy egyenlő, mint y
x == y	x egyenlő y-nal
x != y	x nem egyenlő y-nal

Logikai operátorok

Az && és || (ÉS illetve VAGY) operátorokkal összekapcsolt kifejezések kiértékelése balról jobbra történik, és a kiértékelés azonnal félbeszakad, ha az eredmény igaz vagy hamis volta ismertté válik.
A ! unáris (egyoperandusú) negáló operátor a true értéket false-á, és fordítva alakítja.

A	B	A&&B	A||B
HAMIS	HAMIS	HAMIS	HAMIS
HAMIS	IGAZ	HAMIS	IGAZ
IGAZ	HAMIS	HAMIS	IGAZ
IGAZ	IGAZ	IGAZ	IGAZ

2.8. Inkrementáló és dekrementáló operátorok

A C# nyelv két szokatlan operátort használ a változók inkrementálására (eggyel való növelésére) és dekrementálására (eggyel való csökkentésére). A ++ inkrementáló operátor egyet ad az operandushoz, a -- dekrementáló operátor pedig egyet kivon belőle.

A ++ és -- szokatlan vonatkozása, hogy prefix formában (a változó előtt elhelyezve, pl. ++n) és postfix formában (a változó után elhelyezve, pl. n++) egyaránt létezik. A kétféle változat egyaránt növeli (vagy csökkenti) a változót, de a ++n a felhasználás előtt, az n++ pedig utána növeli az n értékét (a -- operátor hasonlóan működik). Ebből következően minden olyan esetben, amikor a változó értékét is felhasználjuk (nem csak a növelésre vagy csökkentésre, azaz számlálásra van szükség), a ++n és az n++ különbözik.

Ha pl. n értéke 5, akkor

```
x = n++;
```

hatására x értéke 5 lesz, amíg az

```
x = ++n;
```

hatására x értéke 6 lesz.

2.9. Bitenkénti logikai operátorok

A C# nyelvben hat operátor van a bitenkénti műveletekre. Ezek az operátorok csak egész típusú adatokra használhatóak, akár előjeles, akár előjel nélküli változatban. Az egyes operátorok és értelmezésük a következő:

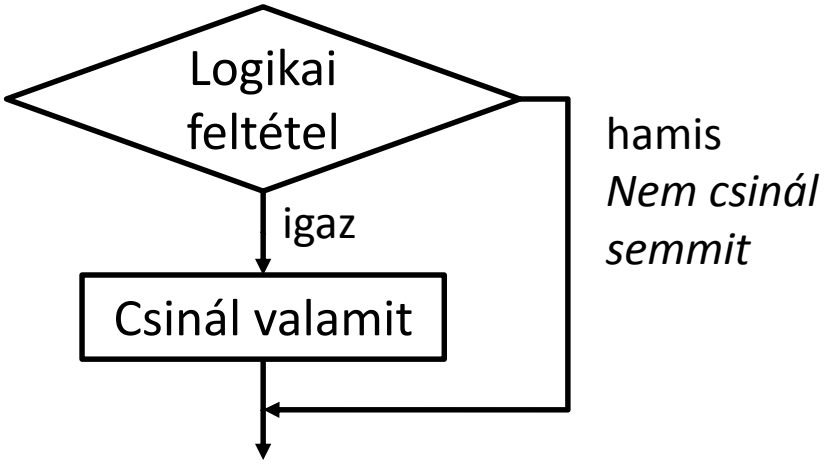
&	bitenkénti ÉS-kapcsolat
|	bitenkénti megengedő (inkluzív) VAGY-kapcsolat
^	bitenkénti kizáró (exkluzív) VAGY-kapcsolat
<<	balra léptetés
>>	jobbra léptetés
~	egyes komplemens képzés (unáris)

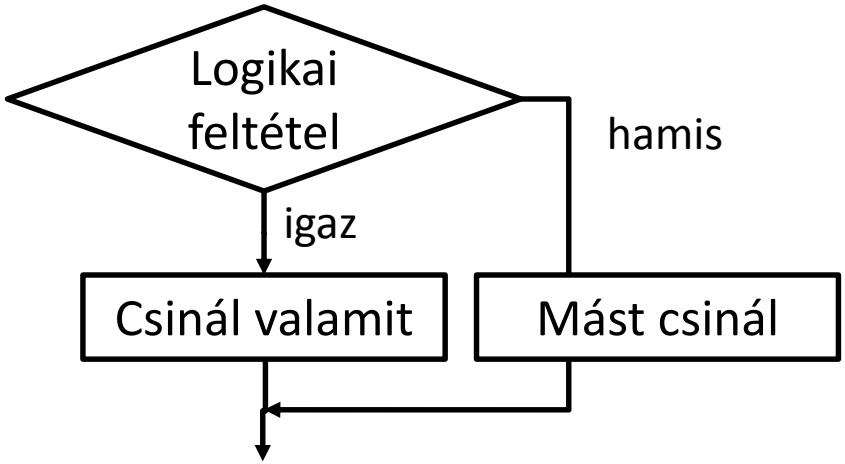
A Math osztály (System.Math)

Metódus	Művelet
Math.Sin(x)	sin(x), ahol az x szög értékét radiánban kell megadni
Math.Cos(x)	cos(x)
Math.Tan(x)	tg(x)
Math.Exp(x)	e^x
Math.Log(x)	ln(x)
Math.Sqrt(x)	x négyzetgyöke
Math.Abs(x)	x abszolút értéke
Math.Round(x)	kerekítés a matematikai szabályok szerint
Math.Ceiling(x)	felfelé kerekítés
Math.Floor(x)	lefelé kerekítés
Math.Pow(x,y)	hatványozás, x^y
Math.PI	a PI konstans (3.14159265358979323846)
Math.E	az e konstans (2.71828182845904523 54)

Vezérlési szerkezetek
Elágazás

- Gyakran előfordul, hogy meg kell vizsgálnunk egy állítást, és attól függően, hogy igaz vagy hamis, a programnak más-más utasítást kell végrehajtania.





Vezérlési szerkezetek
Elágazás szintaktika


```
if (<log. Kif.>)
<utasítás>
```

```
if (<log.kif.>)
<utasítás>
else
<utasítás>
```

Vezérlési szerkezetek
Elágazás


```
int i=1;
if (i < 10)
   i = 10;
else
   i -= 1;
```

```
int i=1,a;
if (i < 10)
{
    i = 10;
    a = 2;
}
else
{
    i -= 1;
    a = 21;
}
```