Szoftvertervezés és -fejlesztés I.

Operátorok
Vezérlési szerkezetek
Gyakorlás

Hallgatói Tájékoztató

A jelen bemutatóban található adatok, tudnivalók és információk a számonkérendő anyag vázlatát képezik. Ismeretük szükséges, de nem elégséges feltétele a sikeres zárthelyinek, illetve vizsgának.

Sikeres zárthelyihez, illetve vizsgához a jelen bemutató tartalmán felül a kötelező irodalomként megjelölt anyag, a gyakorlatokon szóban, illetve a táblán átadott tudnivalók ismerete, valamint a gyakorlatokon megoldott példák és az otthoni feldolgozás céljából kiadott feladatok önálló megoldásának képessége is szükséges.

Szoftvertervezés és -fejlesztés I.

Operátorok

Vezérlési szerkezetek

Gyakorlás

Kifejezések

- A kifejezések („expression”) adatokat szolgáltató operandusokból és rajtuk valamilyen műveletet végző operátorokból állnak
  - Operandus: pl. bármely változó vagy konkrét megadott érték
  - Operátor: pl. + - / *
- A kifejezések egymásba is ágyazhatók
  - Egy kifejezés operandusa maga is lehet kifejezés
- Több operátor esetén ezek fontossági sorrendje (precedenciája) határozza meg a kiértékelés sorrendjét
  - Példa: az „x + y * z” kifejezés kiértékelés szempontjából „x + (y * z)”
  - A sorrend zárójelezéssel explicit módon is meghatározható

Operátorok és precedenciájuk

- **Aritmetikai operátorok**

Operátor	Kifejezés	Precedencia	Jelentés
+	+x	2	Előjelképzés
	x + y	4	Összeadás vagy kombináció *(szám/string)*
-	-x	2	Előjelképzés
	x – y	4	Kivonás
*	x * y	3	Szorzás
/	x / y	3	Osztás *(egész/tört osztás, nullával osztás!)*
%	x % y	3	Maradékképzés
++	x++	1	Növelés eggyel x kiértékelése után
	++x	2	Növelés eggyel x kiértékelése előtt
--	x--	1	Csökkentés eggyel x kiértékelése után
	--x	2	Csökkentés eggyel x kiértékelése előtt

Operátorok és precedenciájuk

- **Relációs (összehasonlító) operátorok**

Operátor	Kifejezés	Precedencia	Jelentés
==	x == y	7	Egyenlő
!=	x != y	7	Nem egyenlő
<	x < y	6	Kisebb
>	x > y	6	Nagyobb
<=	x <= y	6	Kisebb vagy egyenlő
>=	x >= y	6	Nagyobb vagy egyenlő

Operátorok és precedenciájuk

- **Bináris logikai (bitenkénti műveletvégző) operátorok**

Operátor	Kifejezés	Precedencia	Jelentés
~	~x	2	Bitenkénti NEM művelet
&	x & y	8	Bitenkénti ÉS művelet
^	x ^ y	9	Bitenkénti KVAGY (kizáró VAGY) művelet
|	x | y	10	Bitenkénti VAGY művelet
<<	x << y	5	Eltolás balra (x eltolása y helyiértékkel)
>>	x >> y	5	Eltolás jobbra (x eltolása y helyiértékkel)

Operátorok és precedenciájuk

- **Logikai (feltételvizsgáló) operátorok**

Operátor	Kifejezés	Precedencia	Jelentés
!	!x	2	A kifejezés értéke x ellentettje
&&	x && y	11	A kifejezés akkor igaz, ha x és y is igaz
||	x || y	12	A kifejezés akkor igaz, ha x vagy y igaz

Operátorok és precedenciájuk

- **Értékadó operátorok**

Operátor	Kifejezés	Precedencia	Értékadás típusa
=	x = y	14	Egyszerű (x értéke legyen egyenlő y-nal)
+=	x += y	14	Összeadással (x = x + y)
-=	x -= y	14	Kivonással (x = x – y)
***=**	x *= y	14	Szorzással (x = x * y)
/=	x /= y	14	Osztással (x = x / y)
%=	x %= y	14	Maradékképzéssel (x = x % y)
&=	x &= y	14	Bitenkénti ÉS művelettel (x = x & y)
^=	x ^= y	14	Bitenkénti KVAGY művelettel (x = x ^ y)
|=	x |= y	14	Bitenkénti VAGY művelettel (x = x | y)
<<=	x <<= y	14	Bitenkénti eltolással balra (x = x << y)
>>=	x >>= y	14	Bitenkénti eltolással jobbra (x = x >> y)

Utasítások

- Egy program alapvetően utasítások sorozatából áll
- Egyszerű utasítások („statement”)
  - Az egyszerű utasítások lehetnek deklarációk, kifejezések vagy előre definiált utasítások
  - Az egyszerű utasításokat „ ; ” karakter zárja le
- Összetett utasítások („compound statement”)
  - Több utasítás sorozata összefogható egy összetett utasítássá
  - Az összetett utasítások végén nem szerepel „ ; ” karakter
  - Az összetett utasítás másik neve: „blokk” vagy „kódblokk”

Szoftvertervezés és -fejlesztés I.

Operátorok

Vezérlési szerkezetek

Gyakorlás

Az if utasítás

```
if (feltétel)
    utasítás
else
    utasítás
```

- Az if utasítások egymásba is ágyazhatók
  - Minden feltételhez kapcsolódhat else ág, de jelenléte nem kötelező
  - Minden else ág az utolsó (őt közvetlenül megelőző) if utasításra vonatkozik
- Egyenlőségvizsgálat az „==” (és nem az „=”) operátorral
- Végrehajtható: 1 utasítás, vagy kódblokk {} karakterekkel

Rövidzár-kiértékelés

- „Short-circuit evaluation”
- Akkor fordul elő, amikor egy logikai kifejezésben több logikai kifejezést csatolunk össze az ÉS / VAGY (&& / ||) operátorok segítségével
- ÉS operátornál ha az első kifejezés hamis, a másodikkal nem érdemes foglalkozni, az eredmény mindenképp hamis lesz
- VAGY operátornál ha az első kifejezés igaz, a másodikkal nem érdemes foglalkozni, az eredmény mindenképp igaz lesz
- Fontos: C# esetén feltételek, ciklusok kiértékelésénél!

Az üres utasítás / Megjegyzés

```
;
```

- Szintaktikai szerepe van
  - Egyszerű utasítások lezárására szolgál
  - Olyan helyeken használjuk, ahol nincs teendő, de a C# nyelv megköveteli, hogy ott utasítás szerepeljen
  - Hibás használata veszélyes!

```
// Megjegyzés
/* Több
soros (vagy soron belüli)
megjegyzés */
```

Az if utasítás (példa)

```
int i = 12;
if (i == 10)
    Console.WriteLine("Ez bizony pontosan 10");

bool állítás;
if (i > 15)
{
    állítás = true;
    Console.WriteLine("Az állítás igaz, i értéke nagyobb, mint 15");
}
else
{
    állítás = false;
    Console.WriteLine("Az állítás hamis, i értéke nem nagyobb, mint 15");
}
Console.WriteLine(állítás);
```

Gyakorló feladat

Egészítsük ki a Hello, C# World alkalmazásunkat:

Ha a hallgató neve Béla, akkor írjuk ki neki, hogy „SZIA”. Egyébként, írjuk ki, hogy „HELLO”!

A while utasítás

```
while (feltétel)
    utasítás
```

- Szokványos elnevezése: elöltesztelő ciklus („loop”)
- Ha a feltétel mindig teljesül, végtelen ciklusról beszélünk („infinite loop”)
  - A végtelen ciklus gyakori programozói hiba
- Akkor használjuk, ha valamely utasítást kizárólag bizonyos feltétel fennállása esetén kell ismételten többször végrehajtani
- Végrehajtható: 1 utasítás, vagy kódblokk { } karakterekkel

A while utasítás (példa)

```
string s = " ";
int számláló = 0;

while (s == " ")
{
    Console.WriteLine("Kérek szépen egy szöveget! ");
    s = Console.ReadLine();
    számláló++;
    if ( (s != " ") && (számláló > 1) )
        Console.WriteLine("Végre kaptam valamit (" + számláló + " kísérlet után)! ");
}
```

A do...while utasítás

```
do
    utasítás
while (feltétel)
```

- Szokványos elnevezése: hátultesztelő ciklus
- Ha a feltétel mindig teljesül, végtelen ciklusról beszélünk
- Akkor használjuk, ha valamely utasítást legalább egyszer biztosan végre kell hajtani, majd ezek után kizárólag bizonyos feltétel fennállása esetén kell ismételten végrehajtani őket
- Végrehajtható: 1 utasítás, vagy kódblokk { } karakterekkel

A do…while utasítás (példa)

```
string válasz;
int i = 0;

do
{
    i += 2;
    Console.WriteLine(i);
    válasz = Console.ReadLine();
}
while (válasz != "vége");
```

Gyakorló feladat

Egészítsük ki a Hello, C# World alkalmazásunkat:

A hallgató nevét addig kérjük be, amíg be nem ír valamit!

Ne fogadjuk el névnek, hogy „Shakespeare”!

A switch utasítás

```
switch (kifejezés)
{
    case címkekonstans1:
        utasítássorozat
        break;
    case címkekonstans2:
        utasítássorozat
        break;
    ...
    case címkekonstansN:
        utasítássorozat
        break;
    default:
        utasítássorozat
        break;
}
```

- Minden címkekonstans értéke egyszer szerepelhet
- A címkekonstansok sorrendje tetszőleges
  - Ez a default ágra is vonatkozik
- Break helyett más ugrási utasítás is szerepelhet (később)

A switch utasítás (példa)

```
string nyelv;
string országkód = "de";

switch (országkód)
{
    case "hu":
        nyelv = "magyar";
        break;
    case "en":
        nyelv = "angol";
        break;
    case "ch":
    case "de":
        nyelv = "német";
        break;
    default:
        nyelv = "ismeretlen nyelv";
        break;
}
Console.WriteLine(nyelv);
```

Gyakorló feladat

Egészítsük ki a Hello, C# World alkalmazásunkat:

Írjunk külön-külön köszönést a következő nevekre:
Béla – Szia!
Bill – A király!
Joe – Szevasz!
Maldini – Ciao!
Mindenki más – Hello!

Változók hatóköre

- **Alapszabály (egyszerűsítés): a változók azok között a kapcsos zárójelek között élnek, amelyek közt deklaráltuk őket**
- **Később ezt a szabályt pontosítjuk**

```
static void Main(string[] args)
{                                       //<--  i innentől él.
    int i = 10;
    while (i > 0)
    {                                   //<-- fele innentől él.
        double fele = i / 2.0;
        Console.WriteLine(fele);
        i--;
    }                                   //<-- fele idáig él.
    Console.ReadLine();
}                                       //<-- i idáig él.
```

Változók hatóköre

- **Cikluson belül definiált változó csak a cikluson belül látszik**
- **Egy ilyen változó nem használható while esetén a ciklusfeltételben! (akár elöl- akár hátultesztelő)**

```
do
{
    int x = 0;
    x = 6;
} while (x < 5); // ‘x’ does not exist
```

Változók hatóköre

- **Elágazáson belül definiált változó csak az elágazáson belül látszik**

```
if (true)
{
    int x = 5;
}
else
{
    int x = 5; // Nem számít újradeklarálásnak!
}
Console.WriteLine(x); // ‘x’ does not exist
```

Szoftvertervezés és -fejlesztés I.

Operátorok

Vezérlési szerkezetek

Gyakorlás

Gyakorló feladat

Mit látunk a konzolon, ha futtatjuk az alábbi programot?
(A megoldáshoz ne használjon Visual Studio fejlesztői környezetet!)

```
using System;

namespace Feladat
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine((3 < 4 || false) && (false || true));
            Console.WriteLine(!true && (!true || 100 != 5 >> 2));
            Console.WriteLine(true || !(true || false));
            Console.WriteLine(!!!!!!false || true);
            Console.WriteLine(false && (5 > 1 * 1 || 7 - 10 > -100) &&
(!!!!!!false || ((3 < 4 || false)) && (false || true)));
        }
    }
}
```

Gyakorló feladat

Mit látunk a konzolon, ha futtatjuk az alábbi programot?
(A megoldáshoz ne használjon Visual Studio fejlesztői környezetet!)

```
using System;

namespace Feladat
{
    class Program
    {
        static void Main(string[] args)
        {
            int a = 10;
            int b = 3;
            float c = a / b;

            Console.WriteLine(c);
        }
    }
}
```

Gyakorló feladat

Írjon programot, amelynek kezdetén adott egy pozitív egész szám, a „gondolt szám”. A felhasználónak ki kell találnia, hogy mi a gondolt szám. Ehhez a felhasználó megadhat számokat, melyekről a program megmondja, hogy a gondolt számnál nagyobbak, vagy kisebbek-e. A program akkor ér véget, ha a felhasználó kitalálta a gondolt számot. A program jelenítse meg a felhasználó próbálkozásainak számát is.

Gyakorló feladat

Készítsünk programot, mely beolvas a billentyűzetről két számot és egy műveleti jelet, majd kiírja a két számmal elvégzett művelet eredményét. A műveleti jelek megkülönböztetéséhez használjunk többágú (switch, case) elágaztatást.

Gyakorló feladat

Írjon programot, amely egy pozitív egész számnak kiszámítja valamely pozitív egész kitevőjű hatványát, illetve a faktoriálisát! Az aktuális értékeket a felhasználó adhatja meg.

Gyakorló feladat

Készítsen programot, amely két bekért pozitív egész számnak meghatározza a legnagyobb közös osztóját és a legkisebb közös többszörösét!

Gyakorló feladat

Írjon programot, amely a Fibonacci sorozatnak meghatározza valamely elemét!

A Fibonacci sorozat a következő formulák szerint definiált:

$$a_0=a_1=1$$

$$a_n=a_{n-1}+a_{n-2}$$

Gyakorló feladat

Készítsen programot, mely egy pozitív egész számnak kiírja az összes osztóját!

Gyakorló feladat

Írjon programot, mely egy pozitív egész számról megadja, hogy prím-e vagy sem!

Megjegyzés: Azok a pozitív egész számok prímek, melyeknek pontosan kettő darab osztója van. (Az 1 nem prím!)

Gyakorló feladat

Készítsük el a következő feladat C# kódját:

Kérjünk be a felhasználótól pozitív egész számokat, nempozitív szám jelentse a bekérés végét.

Írjuk ki a beírt számok átlagát, de úgy, hogy az átlagból hagyjuk ki a legkisebb és a legnagyobb számot!

Gyakorló feladat

Készítsünk programot, amely meghatározza a bemenetként megadott egynél nagyobb lebegőpontos szám gyökénél nem nagyobb, legnagyobb értékű pozitív egész számot!

Segítség:

Be: *x*

a ← 1

Ciklus amíg *a* * *a* <= *x*

    a ← *a* + 1

Ciklus vége

Ki: *a* - 1

Gyakorló feladat

Készítsünk programot, amely megmondja, hogy a bementként megadott év szökőév-e vagy sem!

Segítség:

Szökőévek a 4-gyel osztható évek, kivéve a százzal oszthatóak, viszont a 400-zal is oszthatóak szökőévek.