Objektumorientált Programozás II.

Adattípusok ismétlés
Operátorok
Hello C# World
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.

Objektumorientált Programozás II.

Adattípusok ismétlés
Operátorok
Hello C# World
Vezérlési szerkezetek
Gyakorlás

Egész (fixpontos) számok

Név	Leírás	Értéktartomány
sbyte	8 bites előjeles egész	-128 : 127
byte	8 bites előjel nélküli egész	0 : 255
short	16 bites előjeles egész	-32 768 : 32 767
ushort	16 bites előjel nélküli egész	0 : 65535
int	32 bites előjeles egész	-2 147 483 648 : 2 147 483 647
uint	32 bites előjel nélküli egész	0 : 4 294 967 295
long	64 bites előjeles egész	-9 223 372 036 854 775 808 : 9 223 372 036 854 775 807
ulong	64 bites előjel nélküli egész	0 : 18 446 744 073 709 551 615

Valós (lebegőpontos) számok

Név	Leírás	Értékes jegy	Értéktartomány
float	32 bites lebegőpontos	7	$\pm 1{,}5*10^{-45}$: $\pm 3{,}4*10^{38}$
double	64 bites lebegőpontos	15	$\pm 5{,}0*10^{-324}$: $\pm 1{,}7*10^{308}$
decimal	128 bites nagypontosságú	28	$\pm 1{,}0*10^{-28}$: $\pm 7{,}9*10^{28}$

	Méret	Előjel	Kitevő	Törtrész	Eltolás
Egyszeres IEEE-754 szabvány pontosság	32 bit	1 bit	8 bit	23 bit	127
Kétszeres pontosság	64 bit	1 bit	11 bit	52 bit	1023

Karakterek, karakterláncok

- Karakter: char (megadás: aposztróffal)
  - char karakter='ű';
- Karakterlánc: string (megadás: idézőjellel)
  - string karakterlanc="Árvíztűrő Tükörfúrógép";
- Speciális karakterek is megadhatóak (@ jellel kikapcsolható):

Jelölés	Karakter
\0	Null karakter
\a	Sípszó
\b	Visszatörlés
\f	Lapdobás
\n	Soremelés
\r	Kocsi vissza
\t	Vízszintes tabulátor

Jelölés	Karakter
\v	Függőleges tabulátor
\x....	Hexadecimális kód
\u....	Unicode karakter
\U....	Unicode karakter
\'	Aposztróf
\"	Idézőjel
\\	Backslash

Logikai típus

Név	Leírás	Értéktartomány
bool	**Logikai adattípus**	true **vagy** false **(igaz vagy hamis)**

- **Teljesítmény-okokból általában nem 1 biten ábrázoljuk, részletesebben lásd IEA**
- **Logikai műveletek:**

A	B	A AND B	A OR B	A XOR B	NOT(A)
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

Változók deklarálása és használata

```
int j = -10;
int x = 10, y = 20;
double pi = 3.14159;
const int száz = 100;
char d = 'x';
char UnicodePélda = '\u0170'; // "Ű" karakter
string jegy = "jeles";
string ElérésiÚt = "C:\\Program Files\\";
string ElérésiÚt2 = @"C:\Program Files\";
string vers = @"Hová merült el
  szép szemed világa";
bool igaz = true;
```

Fontos szabály: azonos névvel egy változót nem lehet kétszer deklarálni!

Típuskonverziók

- A számtípusok közötti konverzió miként történik attól függ, hogy történik –e értékvesztés a konverzió során
- Egyszerű értékadás használható, amennyiben biztos, hogy nincs értékvesztés:

```
byte a=5;       long c=5;       float f=3.2f;
int b=a;        float d=c;      double g=f;
```

- Amennyiben értékvesztés történhet, akkor mindenképp jelezni kell a konverziót, ez az ún. típuskényszerítés, „kasztolás” (typecasting):

```
int a=999;          double d=3.14;      int i1=-1;
byte b=(byte)a;     int c=(int)d;       uint i2=(uint)i1;
```

Típuskonverziók

- A stringgé történő konverzió a C# nyelven MINDEN változónál ugyanúgy történik:

```
byte b=250;                    float f=3.14f;
string s1=b.ToString();        string s2=f.ToString();
```

- Stringből számmá tudunk konvertálni:

```
string s="123";                string s2="123,456";
byte b=byte.Parse(s);          float f=float.Parse(s2);
```

- Typecasting esetén (ebben a félévben számok között):
  célváltozó = (céltípus)forrásváltozó;
- Stringgé konvertálásnál:
  célváltozó = forrásváltozó.ToString();
- Stringből konvertálásnál:
  célváltozó=céltípus.Parse(stringváltozó);

Objektumorientált Programozás II.

Adattípusok ismétlés
Operátorok
Hello C# World
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)

Operátorok és precedenciájuk

- **Hozzáférési célú operátorok**

Operátor	Kifejezés	Precedencia	Jelentés
.	x.y	1	Taghozzáférés (összetett típusoknál és felsorolásoknál)
( )	f(x)	1	Metódushívás (tagfüggvények végrehajtása egyes összetett típusoknál)
[ ]	a[x]	1	Stringből karakter kiszedése (később) Tömbelem-hozzáférés (később) Hozzáférés indexelt tulajdonsághoz *(ebben a félévben ez nem kell*

- **Egyéb operátorok**

Operátor	Kifejezés	Precedencia	Jelentés
? :	x ? y : z	13	Ha az „x” feltétel igaz, akkor a kifejezés értéke „y”, ellenkező esetben „z” lesz

Utasítások

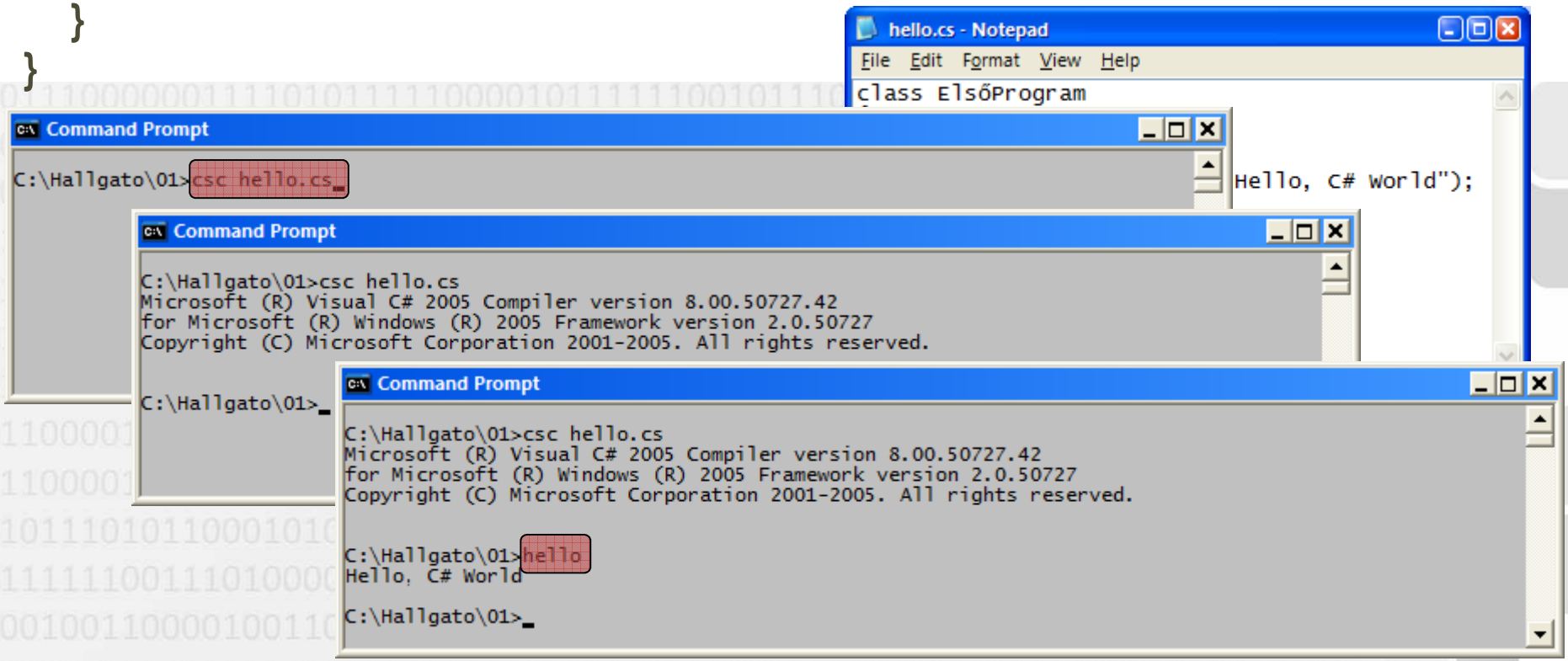
- Egy program alapvetően utasítások sorozatából áll
- Egyszerű utasítások („statement”)
  - Az egyszerű utasítások lehetnek deklarációk, kifejezések vagy előre definiált utasítások
  - Az egyszerű utasításokat „ ; ” karakter zárja le
- Összetett utasítások („compound statement”)
  - Több utasítás sorozata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

Objektumorientált Programozás II.

Adattípusok ismétlés
Operátorok
Hello C# World
Vezérlési szerkezetek
Gyakorlás

Hello, C# World

```
// Első programunk C# nyelven
class ElsőProgram
{
    static void Main()
    {
        System.Console.WriteLine("Hello, C# World");
    }
}
```



Hello, C# World

Készítsünk programot, amely kiírja a konzolra a „Szervusz, hallgató!” szöveget!

```
class Program
{
    static void Main()
    {
        System.Console.WriteLine("Szervusz, hallgató!");
        System.Console.ReadLine();
    }
}
```

Hello, C# World

Készítsünk programot, amely a konzolról beolvas egy nevet, majd név szerint üdvözli az illetőt!

```
class Program
{
    static void Main()
    {
        string név;
        System.Console.WriteLine("Hogy hívnak?");
        név = System.Console.ReadLine();
        System.Console.WriteLine("Szervusz, " + név + "!");
    }
}
```

Objektumorientált Programozás II.

Adattípusok ismétlés
Operátorok
Hello C# World
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
Megjegyzés */
```

Az if utasítás (példa)

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

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

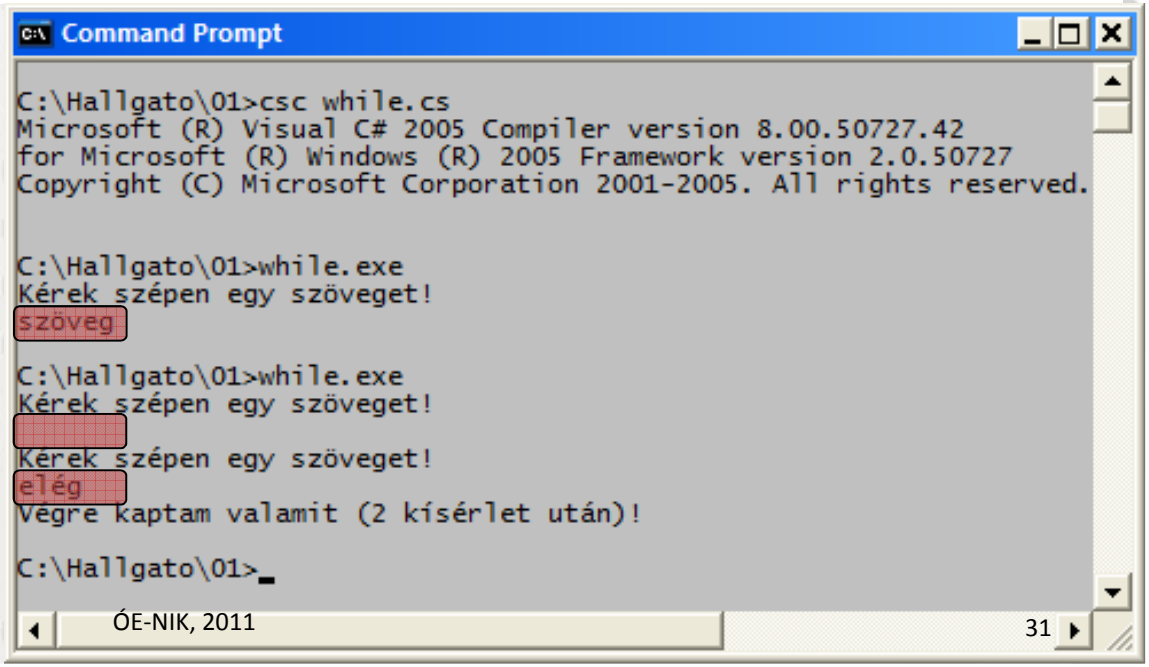
```
while (feltétel)
   utasítás
```

- **Szokványos elnevezése: elöltesztelő ciklus („loop”)**
- **Ha a feltétel mindig teljesül, végtelen ciklusról beszélünk („infinite loop”)**
  - **A végtelen ciklus gyakori programozói hiba forrása**
- **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 == "")
{
    System.Console.WriteLine("Kérek szépen egy szöveget!");
    s = System.Console.ReadLine();
    számláló++;
    if ( (s != "") && (számláló > 1) )
        System.Console.WriteLine("Végre kaptam valamit (" + számláló + " kísérlet után)!");
}
```



A do...while utasítás

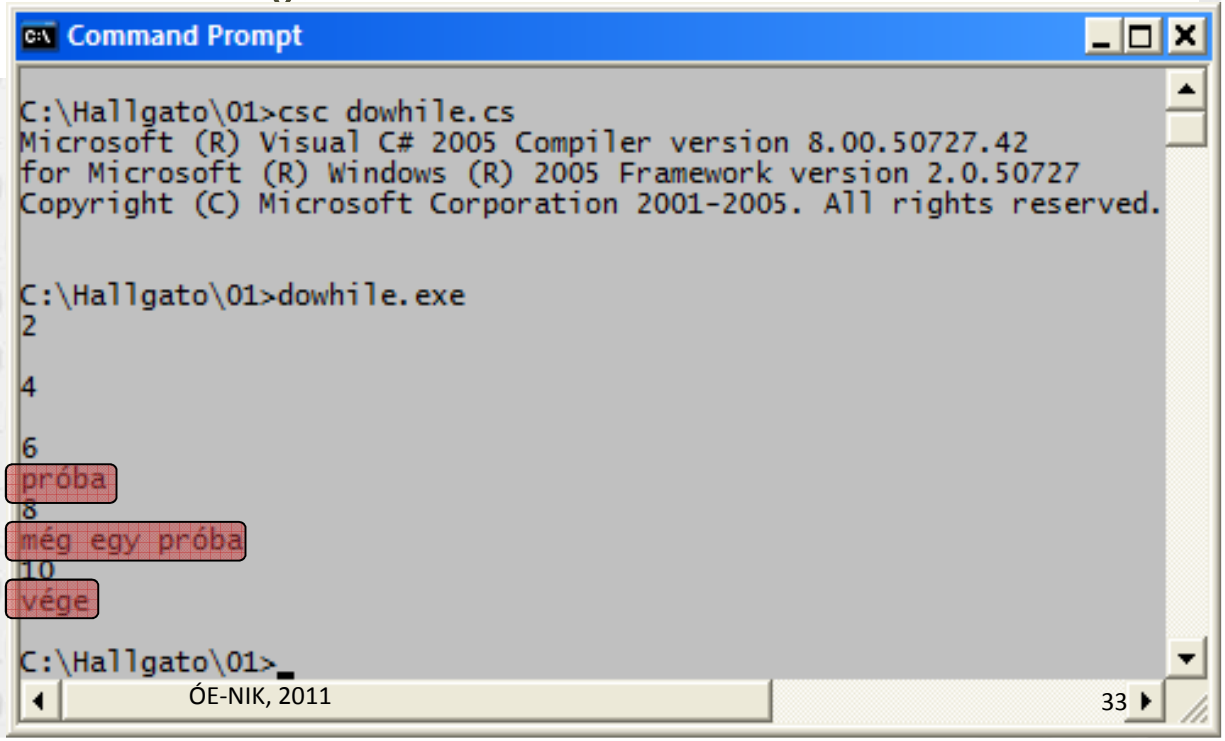
```
do
   utasítás
while (feltétel)
```

- **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;
    System.Console.WriteLine(i);
    válasz = System.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

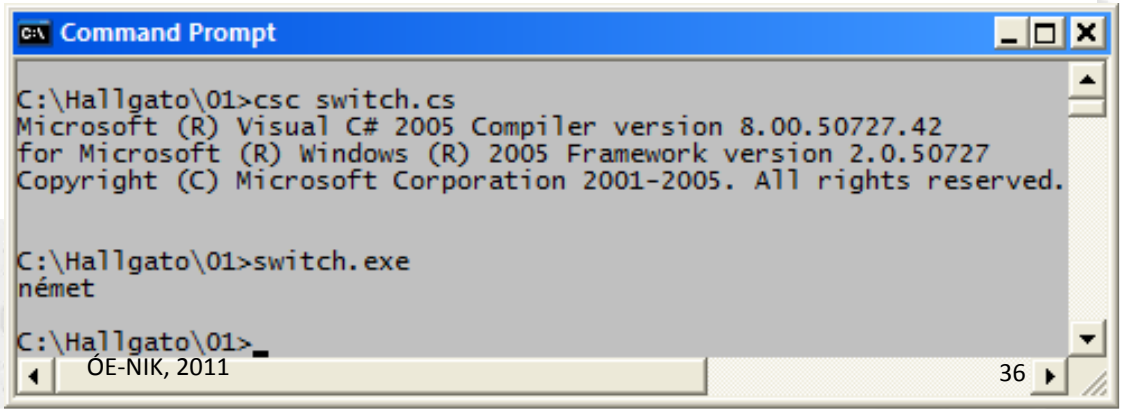
```
switch (kifejezés)
{
    case címkekonstans1:
        utasítássorozat
        break;
    case címkekonstans2:
        utasítássorozat
        break;
    …
    case címkekonstansN:
        utasítássorozat
        break;
    [default:
        utasítássorozat
        break;]
}
```

- **Minden címkekonstans értéke egyszer szerepelhet**
- **A címkekonstansok sorrendje tetszőleges**
  - **Ez a** default **ágra is vonatkozik**

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;
}
System.Console.WriteLine(nyelv);
```



A goto utasítás

```
goto címke;
goto case címkekonstans;
goto default;
```

- **Közvetlen ugrás a megadott címkéhez**
  - **Ciklus belsejébe nem lehet ilyen módon belépni**
- **switch utasításnál ugrás a megadott konkrét (case), illetve alapértelmezett (default) címkéhez**
  - **Ezzel az is elérhető, hogy a switch utasításnál több különböző esetben is végrehajtódjon ugyanaz az utasítássorozat (a megoldás neve „átesés”)**
- **Használata SOHA nem javasolt**

A continue utasítás

```
continue;
```

- **Cikluson belül a hátra lévő kódsorok kihagyása, folytatás a ciklusfeltétel kiértékelésével**
  - **Segítségével átléphetjük az aktuális switch, while, do…while, for, illetve foreach ciklus hátralévő részét**

```
string válasz;
int i = 0;
do
{
    i++;
    if (i%2 == 0)
        continue;
    System.Console.WriteLine(i);
}
while (i<20);
```

A break utasítás

break ;

- **A végrehajtás megszakítása, folytatás a következő utasítással**
  - **Segítségével kiléphetünk az aktuális switch, while, do…while, for, illetve foreach utasítás belsejéből**

```
string válasz;
int i = 0;
do
{
    i += 2;
    if (i > 20)
        break;
    System.Console.WriteLine(i);
    válasz = System.Console.ReadLine();
}
while (válasz != "vége");
```

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

- 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 exists
```

Változók hatóköre

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

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

Objektumorientált Programozás II.

Adattípusok ismétlés
Operátorok
Hello C# World
Vezérlési szerkezetek
Gyakorlás

Gyakorló feladat

Készítsük el a következő feladat struktogramját és 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, 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.

Objektumorientált Programozás II.

- ✓Adattípusok ismétlés
- ✓Operátorok
- ✓Hello C# World
- ✓Vezérlési szerkezetek
- ✓Gyakorlás

Otthoni gyakorló feladatok

Készítsük el az alábbi feladatok megoldásának struktogramját és programját:

1. Olvassunk be egy sugár értéket és számítsuk ki kör kerületét és területét, valamint a gömb felszínét és térfogatát!

2. Olvasson be három számot, majd írassa ki őket csökkenő sorrendben!

3. Egy háromszög oldalainak (a, b, c) hosszát olvassa be a billentyűzetről, majd megmondja, hogy a háromszög szerkeszthető-e! (A háromszög szerkeszthető, ha az (a+b>c) és (a+c>b) és (b+c>a) feltétel teljesül.)

4. Olvassa be egy hónap nevét, majd írja ki, hogy melyik évszakban van az adott hónap!

Irodalom, feladatok

- **Kotsis-Légrádi-Nagy-Szénási: Többnyelvű programozástechnika, PANEM, Budapest, 2007**
- **Faraz Rasheed: C# School, Synchron Data, 2006 http://www.programmersheaven.com/2/CSharpBook**
- **Reiter István: C# jegyzet, DevPortal, 2010, http://devportal.hu/content/CSharpjegyzet.aspx**