**Vezérlési szerkezetek**

**Szelekció (elágazás)**

**Elágazás utasításblokkal**

Amennyiben a paraméterként megadott kifejezés igaz, akkor a megadott utasítások végrehajtódnak. Megadási formátuma a következő:

if (feltétel)

{

utasítás1;

utasítás2;

}

int a = 6;

int b = 5;

if (a>b) {

Console.Write("Az a nagyobb: ");

Console.WriteLine(a);

}

Egyetlen utasításnál nem kötelezőek a blokk zárójelek:

if (feltétel) utasítás;

int a = 6;

int b = 5;

if (a>b) Console.Write("Az a nagyobb: ");

**Elágazás else ággal**

Az **if** szerkezet kibővíthető egy **else (különben)** ággal. Az **else** ágba a vezérlés akkor fog kerülni, ha az **if** feltétele hamis. Megadási formája:

if (feltétel) utasítás;

else utasítás;

int a = 6;

int b = 5;

if (a>b) Console.WriteLine("Az a nagyobb: "+a);

else Console.WriteLine("Az a nem nagyobb: "+a);

if (a>b) {

Console.Write("Az a nagyobb: ");   
Console.WriteLine(a);

}

else {

Console.Write("A b nagyobb vagy egyenlő:");   
Console.WriteLine(b);

}

**Egymásba ágyazott elágazások**

Több **if** utasítás is egymásba ágyazható **else** ágak segítségéhez. Ezt hívjuk **else if** szerkezetnek:

if (feltétel) utasítás;

else if (másik\_feltétel) utasítás;

else utasítás;

if (a==b) {

Console.Write("Egyenlő: ");   
Console.WriteLine(a);

}

else if (a>b) {

Console.Write("a nagyobb:");   
Console.WriteLine(a);

}

else {

Console.Write("b nagyobb:");   
Console.WriteLine(b);

}

Ezen szerkezetben, ha az első feltétel igaz volt, akkor a hozzárendelt utasítások fognak végrehajtódni. Ha az első feltétel nem volt igaz, akkor megnézi a második feltételt. Ha az igaz volt, akkor a hozzá társított utasításokkal folytatódik a program végrehajtása. Ha egyik feltétel sem volt igaz, akkor az utolsó **else** ág utasításai futnak le.

**Többirányú elágazás**

Ezen szerkezet akkor hasznos, ha egy változó konstans, előre tudott értékeire szeretnénk reagálni megadott módon. Például, ha tudjuk, hogy egy változó értéke 1, 2 vagy 3 lehet és nem egy 200 és 30000 közötti szám.

Ezen esetben sokkal átláthatóbb és némileg gyorsabb futást eredményez a programunkban a switch-case vezérlési szerkezet alkalmazása. Ezen felül tisztább kódot produkál, mint rengeteg if-else ág. Megadási formája:

switch ( változó )

{

case konstans kifejezés:

utasítás;

break;

default:

utasítás;

break;

}

A **case** szerkezetek végén lévő break utasítás azért kell, mert ha nem lenne ott, akkor a vezérlés folytatódna a következő **case** ágban is. Hiányáért általában panaszkodik a fordítóprogram.

A **default** ágba a vezérlés akkor kerül, ha a fentebb megadott egyik case vezérlési eset sem teljesül. Megadása nem kötelező. Vezérléshez egész szám típusú változókat, karaktereket, szövegeket használhatunk fel, valamint felsorolást is.

int szam = 3;

switch(szam) {

case 1 : Console.WriteLine("Egy"); break;

case 2 : Console.WriteLine("Kettő"); break;

case 3 : Console.WriteLine("Három"); break;

default : Console.WriteLine("Sok"); break;

}

**Iteráció**

Utasítássorozatot ismétlő programszerkezet.

A ciklus részei:

ciklusfej: meghatározza azokat a feltételeket, amelyek teljesülése esetén a program végrehajtja a ciklusmagot

ciklusmag: az a szekvencia, amelyet a program ismételten végrehajt.

**Elöltesztelős ciklus**

Az elöltesztelős ciklus először megvizsgálja, hogy a feltétel teljesül-e. Ha igen, akkor végrehajtja a ciklusmagot, majd a folyamat az újból kezdődik. Ha nem, akkor a program a ciklus utáni ponton folytatódik tovább, azaz a ciklusmag kimarad.

while ( feltétel )

{

//ciklusmag

}

int i = 1;

while (i<10){

Console.WriteLine(i);

i++;

}

**Hátultesztelős ciklus**

A hátultesztelős ciklus először végrehajtja a ciklusmagot utána megvizsgálja, hogy a feltétel teljesül-e.

A hátultesztelős ciklus ciklusmagja egyszer mindenképpen végrehajtásra kerül.

do

{

//ciklusmag

}

while ( ismétlés feltétele );

int i = 1;

do {

Console.WriteLine(i);

i++;

} while (i<10);

**Növekményes ciklus**

Az ismétlések száma előre ismert. A ciklus a ciklusváltozó előre megadott értékétől, előre megadott értékig fut. A ciklusváltozó egész típusú.

ciklus n-től m-ig

ciklusmag utasításai

ciklus vége

Ha a ciklusváltozó nem egyesével növekszik, akkor megadjuk a számlálás irányát és lépésközét is.

ciklus n-től m-ig lefelé egyesével

ciklusmag utasításai

ciklus vége

For ciklus C# formája:

for (utasítás1; feltétel; utasítás2) {

// Ciklusmag

...

}

utasítás1: ciklusváltozó deklarálása

feltétel: amíg a feltétel igaz addig fut a ciklus

utasítás2: ciklusváltozó növelése/csökkentése

Mindaddig amíg a feltétel teljesül a feltételvizsgálatot követően megtörténik a ciklusmag végrehajtása, majd a ciklusváltozó csökkentése és ismét kezdődik előröl. Ha a feltétel nem teljesül, akkor ciklust követő utasítással folytatódik a program végrehajtása.

for (int i = 0; i < 10; i++){

Console.WriteLine(i)

}

for (int i = 9; i >= 0; i--){

Console.WriteLine(i);

}

Bejáró ciklus, a **foreach** ciklus egy speciális ciklus, amelyet direkt listák, tömbök és minden olyan osztály elemeinek bejárására találtak ki. A ciklus annyiszor fog lefutni, ahány elemű a lista.

Egy tömb elemein végigmehetünk az indexeik segítségével. Erre többnyire a for ciklust alkalmazzuk úgy, hogy a ciklusváltozót a tömb indexelésére használjuk fel.

int[] tömb = {1,3,5,6,7,9}

for(int i = 0; i<tömb.Length; i++){

Console.WriteLine(’A tömb {0}. eleme: {1}’, i, tömb[i]);

}

A **valtozo** nevű ciklusváltozó pedig minden egyes ciklus futáskor a lista, aktuális elemét tárolja.

foreach (var valtozo in lista)

{

//ciklusmag

}