Ozsvárt-féle pszeudonyelv definíció

Tartalomjegyzék

[1. Bevezetés 1](#_Toc480750854)

[2. Lefoglalt szavak 2](#_Toc480750855)

[3. Változóhasználat, típusok 2](#_Toc480750856)

[4. Típuskonverziók 4](#_Toc480750857)

[5. Operátorok 5](#_Toc480750858)

[6. Kommentezés 7](#_Toc480750859)

[7. Vezérlési szerkezetek 7](#_Toc480750860)

[8. I/O kezelés 8](#_Toc480750861)

[9. Tömbkezelés 9](#_Toc480750862)

[10. Nyelvtan definíció 11](#_Toc480750863)

A megjegyzések zöld színnel szerepelnek.

# Bevezetés

A nyelv a C# nyelvet veszi alapjául. Ami e dokumentumból nem derül ki egyértelműen, arra a C# nyelv szabályai érvényesek.

A nyelv **nem** kisbetű-nagybetű érzékeny (non-case-sensitive).

Minden program

* elejét a **program\_kezd** sor,
* végét a **program\_vége** sor jelzi.

**Definíció (whitespace karakter)**: Tabulátor vagy szóköz karakter.

Egy sorba csak egy utasítás írható. Minden kulcsszó előtt és után szerepelnie kell legalább egy whitespace karakternek.

A nyelv nem definiál függvényeket, eljárásokat.

A **kilép** illetve **kilépés** utasítás használható a program futásának megszakítására.

# Lefoglalt szavak

**Lefoglalt szavak listája:**

1. Kulcsszavak:

program\_kezd

program\_vége

kilép, kilépés

ha

akkor

különben

elágazás\_vége

ciklus\_amíg

ciklus\_vége

beolvas

beolvas:

kiír

kiír:

létrehoz

egész

tört

logikai

szöveg

1. Logikai literálok:

igaz

hamis

1. Operátorok

Lásd [Operátorok](#_Operátorok) fejezet.

# Változóhasználat, típusok

A változókat használat előtt deklarálni kell és minden változónak meg kell adni a típusát.  
(A tömbök létrehozása és kezelése eltérő módon történik, lásd [Tömbkezelés](#_Tömbkezelés) fejezet.)

**Deklarálás módja:** típus név = kezdőérték

*Példa deklarációra: egész x = 5*

**Értékadás:**

Az értékadás az egyenlőségjel operátorral történik:

x = 3

Ha egy változónak nem adnak kezdeti értéket, akkor az szintaktikai hibának minősül. Tömb típusok esetén létrehozáskor a tömb elemei az adott típus alapértelmezett értékét veszik fel, ez a következő táblázatban található.

**Típusok:**

|  |  |
| --- | --- |
| **Típus neve** | **Alapértelmezett érték** |
| egész | 0 |
| tört | 0,0 |
| logikai | hamis |
| szöveg | ”” (üres szöveg) |

**Literálok:**

és jelöljük az üres karaktersorozatot ε–nal.  
Jelentse + az 1 vagy több darab karaktert, \* a 0 vagy több darab karaktert, | a „vagy” kapcsolatot.

|  |  |  |
| --- | --- | --- |
| **Típus neve** | **Literál szabályok** | **Példa:** |
| egész |  | -64 |
| tört |  | 3,14 |
| logikai | hamis | igaz | hamis |
| szöveg | Két idézőjel (””) között tetszőleges, idézőjelet nem tartalmazó karaktersorozat. | ”Tetszőleges szöveg…” |

**Elnevezési konvenció:**

A lefoglalt szavak nem használhatók változónévként.

Egy változónév

* **első karaktere** magyar betű,
* **minden további karaktere** magyar betű, arab számjegy vagy aláhúzás karakter ( \_ ) lehet

**Elfogadott betűk listája (egykarakteres magyar betűk):** a, á, b, c, d, e, é, f, g, h, i, í, j, k, l, m, n, o, ó, ö, ő, p, q, r, s, t, u, ú, ü, ű, v, w, x, y, z, A, Á, B, C, D, E, É, F, G, H, I, Í, J, K, L, M, N, O, Ó, Ö, Ő, P, Q, R, S, T, U, Ú, Ü, Ű, V, W, X, Y, Z.

**Elfogadott számjegyek listája**: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

**Escape-szekvenciák**:

Minden **szöveg** típusú értékbe beilleszthetőek ún. escape-szekvenciák, amelyeknek speciális jelentése van az adott szövegen belül. Ezek az alábbi táblázatban találhatóak:

|  |  |
| --- | --- |
| **Jel** | **Név** |
| \n | Új sor |
| \t | Vízszintes tabulátor |
| \” | Idézőjel |
| \\ | Fordított perjel |

# Típuskonverziók

A nyelv erősen típusos. Értékadás esetén az implicit típuskonverzió automatikusan megtörténik.

**Implicit konverziók:**

1. Bármiből szövegbe
2. Egészből törtbe

**Explicit konverziók:**

A konverzió elvégezhető a beépített konverziós függvények használatával is, ezeket mutatja be a következő táblázat:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Forrás ↓ |  | | | |
| Cél → |  | egész | tört | logikai | szöveg |
|  | egész |  | (implicit) | egészből\_logikaiba() | (implicit) |
| tört | törtből\_egészbe() |  | törtből\_logikaiba() | (implicit) |
| logikai | logikaiból\_egészbe() | logikaiból\_törtbe() |  | (implicit) |
| szöveg | szövegből\_egészbe() | szövegből\_törtbe() | szövegből\_logikaiba() |  |

**Példa implicit konverzióra:**

egész a = 3

tört b = a //b értéke 3,0 lesz

**Példa explicit konverzióra (konverziós függvény használatára):**

tört x = 5,4

egész y = törtből\_egészbe(x) //y értéke 5 lesz

**Típuskonverziós függvények:**

* **egészből\_logikaiba():** 0 bemenet esetén hamis a visszaadott érték, egyébként igaz.
* **törtből\_egészbe():** a kimenet a bemeneti szám lefelé, egészre kerekítve.
* **törtből\_logikaiba():** 0,0 bemenet esetén hamis a visszaadott érték, egyébként igaz
* **logikaiból\_egészbe():** hamis bemenet esetén 0 a kimenet, igaz bemenet esetén 1.
* **logikaiból\_törtbe():** hamis bemenet esetén 0,0 a kimenet, igaz bemenet esetén 1,0.
* **szövegből\_egészbe():** megadja a szöveg egészbeli reprezentációját decimális formában. A szám előtt a negatív előjelet jelezheti a kötőjel (-) karakter.
* **szövegből\_törtbe():** megadja a szöveg törtbeli reprezentációját decimális formában. A szám előtt a negatív előjelet jelezheti a kötőjel (-) karakter. Az egész és tizedes jegyek elkülönítésére a vessző karaktert (,) használja.
* **szövegből\_logikaiba():** megadja a szöveg logikai reprezentációját:  
  ”hamis” → hamis  
  ”igaz” → igaz  
  A függvény **nem** kisbetű-nagybetű érzékeny, azaz pl. ”hAmiS” bemeneti szövegre is hamis a kimenet.  
  Helytelen bemenet esetén futásidejű hiba keletkezik.

# Operátorok

A nyelv az alábbi operátorokat definiálja.

A használható típusok oszlopban ha egy adott operátornál egy típust jelöltem meg, akkor adott típusú literálon és változón (illetve többoperandusú operátorok között azok között) használható, vagy azok egy kifejezésén.

|  |  |  |  |
| --- | --- | --- | --- |
| **Jel** | **Név** | **Példa** | **Használható típusok** |
| [ ] | tömbindexelő | t**[**x**]**  (t tömb x. elemére való hivatkozás) | Tömbindex csak egész lehet. |
| - | numerikus negáció | **-**x |  |
| ! | logikai negáció | **!**d | logikai |
| ( ) | kerek zárójelpár (precendencia jelölésére) | 2 \* **(**4+3**)** | bármilyen |

|  |  |  |  |
| --- | --- | --- | --- |
| **Jel** | **Név** | **Példa** | **Használható típusok** |
| = | értékadás | x **=** 1 | bármilyen |
| == | egyenlőség-vizsgálat | y **==** 4 |
| != | nem-egyenlőség vizsgálat | s **!=** 8 |
| és | feltételes és művelet | z > 3 **és** y < 4 | logikai |
| vagy | feltételes vagy művelet | k != 9 **vagy** s > s | logikai |
| >  >=  <  <= | relációs operátorok | x **>** 3  y **>=** 3  x **<** 4  y **<=** 4 | szám típusok: egész, tört |
| + | összeadás | g **+** h |
| - | kivonás | i **–** j |
| \* | szorzás | k **\*** l |
| / | osztás | m **/** n |
| mod | maradékos osztás | o **mod** p |
| . | szöveg összefűzés | ”alma”**.**”körte” | szöveg |

# Kommentezés

A nyelv kommentezési lehetőségei megegyeznek a C# nyelvben lévőkkel:

|  |  |  |
| --- | --- | --- |
| **Jel** | **Név** | **Példa** |
| // | egysoros komment | egész x = 4 //x értéke 4 |
| /\* \*/ | többsoros komment | /\* A sorozatszámítás egy egyszerűen megérthető programozási tétel. \*/ |

// után az adott sorban minden karaktert kommentként értelmez.

/\* és \*/ karakterpárok között minden karaktert kommentként értelmez.

# Vezérlési szerkezetek

**Szelekció:**

**ha** feltétel **akkor**

utasítás(ok)

**elágazás\_vége**

**ha** feltétel **akkor**

utasítás(ok)

**különben**

utasítás(ok)

**elágazás\_vége**

**Iteráció:**

**ciklus\_amíg** bennmaradási\_feltétel

utasítás(ok)

**ciklus\_vége**

Vezérlés folyamata:

1. Bennmaradási feltétel megvizsgálása. Ha hamis, ugorj a ciklus\_vége utáni sorra.
2. Ciklusmag utasításainak végrehajtása. Ugorj az 1. pontra.

# I/O kezelés

Az input és output kezelésére a következő utasítások használhatóak:

**Beolvasás:**

Utasítás: beolvas

Példa: beolvas x

Leírás: beolvassa az aktuális input streamről a következő sort, majd a beolvasott szöveget megpróbálja átalakítani x típusára. Ha sikertelen, akkor futásidejű hiba keletkezik.

Megjegyzés: x-et a használat (beolvasás) előtt deklarálni kell!

**Kiírás:**

Utasítás: kiír

Példa: kiír x

Leírás: kiírja az aktuális output streamre az x változó tartalmát.

**„Összetett” kiírás:**

Egy változó értékét hozzáfűzhetjük a kiírandó szöveghez.

**Példa:** kiír ”x értéke: ”.x

**Magyarázat:** Összefűzi az ”x értéke: ” szöveget az x változó értékének szövegbeli reprezentációjával, majd kiírja a konzolra.

Működési példa: ha x értéke 12, akkor ez kerül az aktuális output streamre:

x értéke: 12

# Tömbkezelés

A tömbindexelés nullától történik. Tömbindexként csak egész literálok illetve egész változók használhatók.

A nyelv jelenlegi verziójában csak az egydimenziós tömbök támogatottak.

**Tömblétrehozás:**

egész[] x = létrehoz[N]

ahol:

N a tömb mérete (pozitív egész szám)

Egy tömb deklarációja és definíciója egy sorba is vonható:

**Példa:** egész[] tomb = létrehoz[6]

A tömb elemeire való hivatkozás:

x[0] = 5 Beállítja az x tömb 0. elemének értékét az 5 értékre.

# Nyelvtan definíció

Alant a nyelvtan definiálom a C#-hoz az MSDN-en is használt nyelvtanmegadásos formában.

Kezdő/mondat szimbólum: ***program***

***program***: program\_kezd újsor ***utasítások*** program\_vége