Pszeudonyelv-definíció

Tartalomjegyzék

[1. Bevezetés 1](#_Toc496455114)

[2. Lefoglalt szavak 2](#_Toc496455115)

[3. Változóhasználat, típusok 3](#_Toc496455116)

[4. Típuskonverziók 5](#_Toc496455117)

[5. Operátorok 6](#_Toc496455118)

[6. Kifejezések 7](#_Toc496455119)

[7. Kommentezés 8](#_Toc496455120)

[8. Vezérlési szerkezetek 9](#_Toc496455121)

[9. I/O kezelés 10](#_Toc496455122)

[10. Tömbkezelés 11](#_Toc496455123)

[11. Nyelvtan definíció 12](#_Toc496455124)

# 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\_kezdsor,
* 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, osztályokat vagy más objektumorientált elvekben használt struktúrákat.

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 és kezdőértékét. A tömbök létrehozása és kezelése eltérő módon történik, ezt megtalálható a [Tömbkezelés](#_Tömbkezelés) fejezetben.

**Létrehozás módja:** típus név = kezdőérték

Példa:egész x = 5

**Értékadás:**

Az értékadás az egyenlőségjel operátorral történik:

x = 3

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 beilleszthetők 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 implicit típuskonverzió nincs. Explicit típuskonverzió a beépített konverziós függvények használatával végezhető el.

**Példa konverziós függvény használatára:**

egész y = törtből\_egészbe(5,4) // y értéke 5 lesz

Minden két alaptípus között (egész, tört, szöveg, logikai) értelmezett konverziós függvény, ezek elnevezési sémája a következő:

forrástípusból\_céltípusba

Azaz ha a forrástípus egész, a céltípus logikai, akkor a függvény neve:

egészből\_logikaiba

Így a következő függvények alakulnak ki:

* egészből\_logikaiba
* egészből\_törtbe
* egészből\_szövegbe
* törtből\_\_egészbe
* törtből\_logikaiba
* törtből\_szövegbe
* logikaiból\_egészbe
* logikaiból\_törtbe
* logikaiból\_szövegbe
* szövegből\_egészbe
* szövegből\_törtbe
* szövegből\_logikaiba

# Operátorok

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 |  | bármilyen |

|  |  |  |  |
| --- | --- | --- | --- |
| **Jel** | **Név** | **Példa** | **Használható típusok** |
| = | értékadás | x **=** 1 | bármilyen két azonos típus között |
| == | egyenlőség-vizsgálat | y **==** 4 |
| != | nem-egyenlőség vizsgálat | s **!=** 8 |
| és | feltételes és művelet | z **és** y | logikai |
| vagy | feltételes vagy művelet | k **vagy** 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 |

# Kifejezések

A nyelv (a szintaktikus elemző implementálásakor felmerült nehézségek miatt) korlátozott kifejezésfelismerési képességekket definiál. Ez a nyelv kifejezőerejét nem csökkenti, de a használat módját kényelmetlenebbé teszi.

Példák:

Ha a következő kifejezést szeretnénk leírni a nyelvben:

egész a = 2 \* (3 + 4) – 2

Akkor azt a fenti forma helyett így kell leírnunk:

egész a = 3 + 4

a = 2 \* a

a = a – 2

Ezek pontosabb szabályai kiolvashatóak a szintaktikus nyelvtan definícióból.

# 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 egyszerű 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 használhatók:

**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 futási idejű 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[] tömb = létrehoz[6]

A tömb elemeire való hivatkozás:

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

# Nyelvtan definíció

Itt található a nyelvtan szintaktikájának definíciója Backus-Naur formában.

Kezdő mondat szimbólum: <Program>

<Program> ::= "program\_kezd" "újsor" <Állítások> "program\_vége"

<Állítások> ::= <Állítás> "újsor" <Állítások>

| <Állítás> "újsor"

<Állítás> ::= <VáltozóDeklaráció>

| <Értékadás>

| <IoParancs>

| "kilép"

| "ha" <NemTömbLétrehozóKifejezés> "akkor" "újsor" <Állítások> "különben" "újsor" <Állítások> "elágazás\_vége"

| "ha" <NemTömbLétrehozóKifejezés> "akkor" "újsor" <Állítások> "elágazás\_vége"

| "ciklus\_amíg" <NemTömbLétrehozóKifejezés> "újsor" <Állítások> "ciklus\_vége"

<VáltozóDeklaráció> ::= <AlapTípus> "azonosító" "=" <NemTömbLétrehozóKifejezés>

| <TömbTípus> "azonosító" "=" "azonosító"

| <TömbTípus> "azonosító" "=" <TömbLétrehozóKifejezés>

| <AlapTípus> "azonosító" "=" <BelsőFüggvény> "(" <NemTömbLétrehozóKifejezés> ")"

<Értékadás> ::= "azonosító" "=" <NemTömbLétrehozóKifejezés>

| "azonosító" "=" <TömbLétrehozóKifejezés>

| "azonosító" "=" <BelsőFüggvény> "(" <NemTömbLétrehozóKifejezés> ")"

| "azonosító" "[" <NemTömbLétrehozóKifejezés> "]" "=" <NemTömbLétrehozóKifejezés>

<Operandus> ::= <UnárisOperátor> "azonosító"

| <UnárisOperátor> "literál"

| "azonosító" "[" <Operandus> "]"

| "azonosító"

| "literál"

<NemTömbLétrehozóKifejezés> ::= <BinárisKifejezés>

| <Operandus>

<TömbLétrehozóKifejezés> ::= "létrehoz" "[" <NemTömbLétrehozóKifejezés> "]"

<BinárisKifejezés> ::= <Operandus> <BinárisOperátor> <Operandus>

<BinárisOperátor> ::= "=="

| "!="

| "és"

| "vagy"

| ">"

| ">="

| "<"

| "<="

| "+"

| "-"

| "\*"

| "/"

| "mod"

| "."

<AlapTípus> ::= "egész"

| "tört"

| "szöveg"

| "logikai"

<TömbTípus> ::= "egész tömb"

| "tört tömb"

| "szöveg tömb"

| "logikai tömb"

<IoParancs> ::= "beolvas" "azonosító"

| "kiír" "azonosító"

<BelsőFüggvény> ::= "egészből\_logikaiba"

| "egészből\_törtbe"

| "egészből\_szövegbe"

| "törtből\_egészbe"

| "törtből\_logikaiba"

| "törtből\_szövegbe"

| "logikaiból\_egészbe"

| "logikaiból\_törtbe"

| "logikaiból\_szövegbe"

| "szövegből\_egészbe"

| "szövegből\_törtbe"

| "szövegből\_logikaiba"

<UnárisOperátor> ::= "-"

| "!"