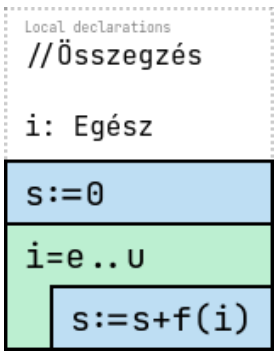
Típus | Speci | Stuki | C#

Type

Specifikáció	Algoritmus Stílus	C# Kód Snipet	Magyarázat
a∈N	a: Természetes	int a;	a egy darab természetes szám.
b∈Z	b: Egész	int b;	b egy darab egész szám.
c∈R	c: Valós	double c;	c egy valós szám (tizedestört).
d∈L	d: Logikai	bool d;	d egy logikai érték (igaz/hamis).
e∈S	e: Szöveg	string e;	e egy szöveg (karakterlánc).
f∈C	f: Karakter	char f;	f egy darab karakter.
iv1∈[1..3]	iv1: [1..3]	int iv1;	iv1 egy egész szám (az [1..3] korlátot kódban kell ellenőrizni).
x1∈N[1..3]	x1: Tömb(1..3, Természetes)	int[] x1 = new int[3];	x1 egy 3 elemű tömb (fix méretű).
n∈N, x∈Z[1..n]	n: Természetesx: Tömb(1..n, Egész)	int n;int[] x;x = new int[n];	n egy szám, x pedig egy n elemű tömb (futásidőben dől el).
x3∈N[1..]	x3: Tömb(1.., Természetes)	List<int> x3;x3 = new List<int>();	x3 egy dinamikus méretű tömb (lista), aminek nem tudjuk előre a méretét.
m1∈Z[1..8,1..8]	m1: Tömb(1..8, 1..8, Egész)	int[,] m1;m1 = new int[8, 8];	m1 egy 8×8-as mátrix (fix méretű).
m2∈Z[1..n,1..n]	m2: Tömb(1..n, 1..n, Egész)	int[,] m2;m2 = new int[n, n];	m2 egy n x n-es mátrix (futásidőben dől el).
r1∈név:S x jegy:N	r1: Rekord(név: Szöveg, jegy: Term.)	EgyediRekord r1;	r1 egy rekord/struktúra típusú változó.
Hallgató=(...)	Típus Hallgató = Rekord(...)	struct Hallgato { ... }	Hallgató néven egy struktúra típust definiálunk.
h2∈Hallgató[1..]	h2: Tömb(1.., Hallgató)	List<Hallgato> h2;h2 = new List<Hallgato>();	h2 egy dinamikus lista, ami Hallgato struktúrákat tárol.

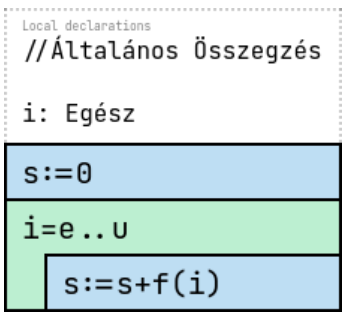
☝ Összegzés

Be: e∈Z, u∈Z
Ki: s∈H
Ef: -
Uf: s=SZUMMA(i=e..u, f(i))



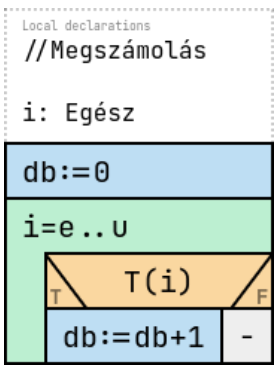
☝ Általános Összegzés

Be: e∈Z, u∈Z
Ki: s∈H
Ef: -
Uf: s=SZUMMA(i=e..u,f(i),0,+)



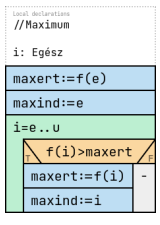
☝ Megszámolás

Be: e∈Z, u∈Z
Ki: db∈N
Ef: -
Uf: db=SZUMMA(i=e..u, 1, T(i))
Rövidítve:
Uf: db=DARAB(i=e..u, T(i))



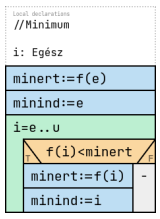
☝ Maximum

Be: e∈Z, u∈Z
Ki: maxind∈Z, maxért∈H
Ef: e<=u
Uf: maxind∈[e..u] és
∀i∈[e..u]:(f(maxind)>=f(i)) és
maxért=f(maxind)
Rövidítve:
Uf: (maxind,maxért)=MAX(i=e..u, f(i)



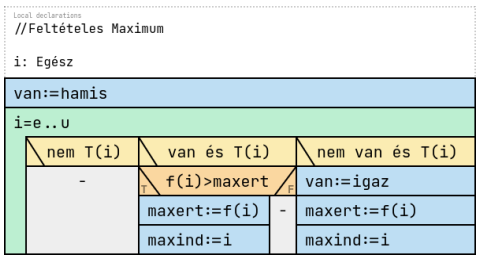
☝ Minimum

Be: e∈Z, u∈Z
Ki: minind∈Z, minért∈H
Ef: e<=u
Uf: minind∈[e..u] és
∀i∈[e..u]:(f(minind)<=f(i)) és
minért=f(minind)
Uf: (minind,minért)=MIN(i=e..u,f(i))



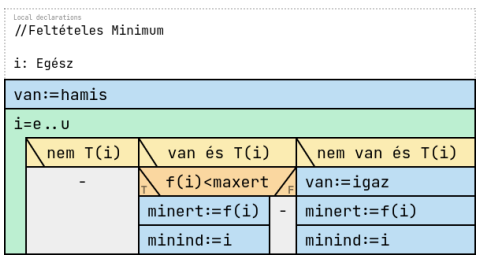
☝ Feltételes Maximum

Be: e∈Z, u∈Z
Ki: van∈L, maxind∈Z, maxért∈H
Ef: -
Uf: van = ∃i∈[e..u]:(T(i)) és
van → (maxind∈[e..u] és
maxért=f(maxind) és T(maxind) és
∀i∈[e..u]:(T(i) → maxért>=f(i))))
Uf: (van,maxind,maxért)=FELTMAX(i=e..u,f(i),T(i))



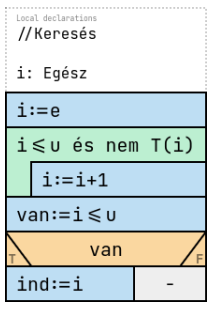
☝ Feltételes Minimum

Be: e∈Z, u∈Z
Ki: van∈L, minind∈Z, minert∈H
Ef: -
Uf: van = ∃i∈[e..u]:(T(i)) és
van → (minind∈[e..u] és
maxért=f(minind) és T(minert) és
∀i∈[e..u]:(T(i) → minert>=f(i))))
Uf: (van,minind,minert)=FELTMIN(i=e..u,f(i),T(i))



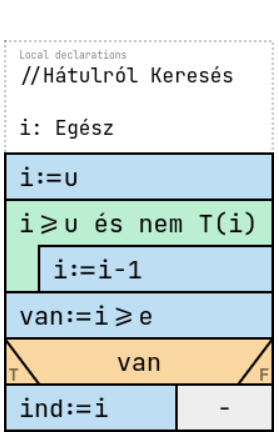
☝ Keresés

Be: e∈Z, u∈Z
Ki: van∈L, ind∈Z
Ef: -
Uf: van=∃i∈[e..u]:(T(i)) és
van→(ind∈[e..u] és T(ind) és
∀i∈[e..ind-1]:(nem T(i)))
Rövidítve:
Uf: (van,ind)=KERES(i=e..u,T(i))



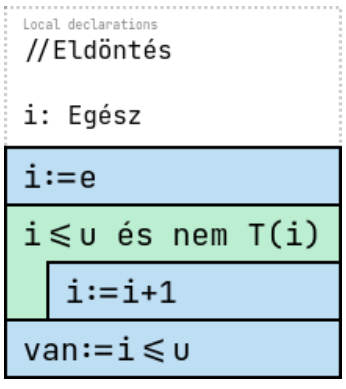
☝ Hátulról keresés

Be: e∈Z, u∈Z
Ki: van∈L, ind∈Z
Ef: -
Uf: (van,-ind)=KERES(i=u..-e,T(-i))



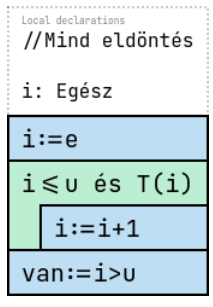
☝ Eldöntés

Be: e∈Z, u∈Z
Ki: van∈L
Ef: -
Uf: van=∃i∈[e..u]:(T(i))
Rövidítve:
Uf: van=VAN(i=e..u,T(i))



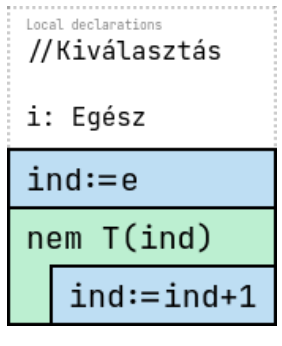
☝ Mind/optimista eldöntés

Be: e∈Z, u∈Z
Ki: van∈L
Ef: -
Uf: van=∀i∈[e..u]:(T(i))
Rövidítve:
Uf: van=VAN(i=e..u,T(i))



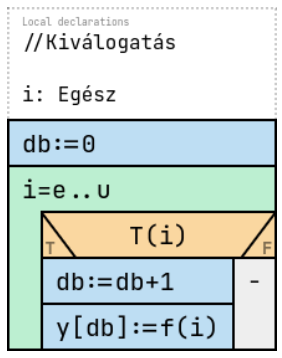
☝ Kiválasztás

Be: e∈Z
Ki: ind∈Z
Ef: ∃i∈[e..∞]:(T(i))
Uf: ind>=e és T(ind) és
∀i∈[e..ind-1]:(nem T(i)))
Rövidítve:
Uf: ind=KIVÁLASZT(i>=e,T(i))



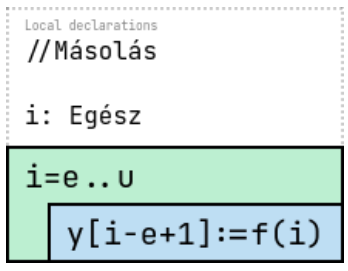
☝ Kiválogatás

Be: e∈Z, u∈Z
Ki: db∈N, y∈H[1..db]
Ef: -
Uf: db=DARAB(i=e..u,T(i)) és
∀i∈[1..db]:(
∃j∈[e..u]:T(j) és y[i]=f(j))
és y⊆(f(e),f(e+1),...,f(u))
Rövidítve:
Uf: (db,y)=KIVÁLOGAT(i=e..u,T(i),f(i))



☝ Másolás

Be: e∈Z, u∈Z
Ki: y∈H[1..u-e+1]
Ef: -
Uf: ∀i∈[e..u]:(y[i-e+1]=f(i))
Rövidítve:
Uf: y=MÁSOL(i=e..u, f(i))




```
// Név: Szász Roland
// Neptun kód: UW0FDO
// Inf-es e-mail: UW0FDO@inf.elte.hu

using System;
using System.Collections.Generic; // A Dynamic List-hez kell

namespace Puska
{
   // (Hallgató=(név:S x jegy:N))
   public struct Hallgato
   {
      public string nev;
      public int jegy;
   }

   class Program
   {
      static void Main(string[] args)
      {
         #region Deklaráció

         // 1. Elemi típusok (pl. a∈N, e∈S)
         int a;
         string e;

         // 2. Tömb (fix méretű, pl. x∈Z[1..n])
         int n; // A tömb mérete
         int[] x;
```

```
// 3. Lista (dinamikus méretű, pl. x3∈N[1..])
List<int> x3;

// 4. Struktúra (egyedi adat, pl. r1∈név:S x jegy:N)
Hallgato r1; // Létrehozunk egy 'r1' változót a 'Hallgato' tervrajz alapján

// 5. Struktúrák listája (pl. h2∈Hallgató[1..])
List<Hallgato> h2;

#endregion

#region Beolvasás (és Inicializálás)

// Értékadás
a = 10;
e = "Hello";

// 'n' beolvasása (most csak szimuláljuk)
n = 5;

// Tömb létrehozása 'n' alapján
x = new int[n];

// Listák létrehozása (üresen)
x3 = new List<int>();
        h2 = new List<Hallgato>();

        // Struktúra mezőinek feltöltése
        r1.nev = "Minta Béla";
        r1.jegy = 5;

        #endregion

        #region Feldgozás
        // Adunk a listákhoz 1-1 elemet
        x3.Add(100);

        // Létrehozunk egy új hallgatót és hozzáadjuk a listához
        Hallgato ujHallgato;
        ujHallgato.nev = "Nagy Anna";
        ujHallgato.jegy = 4;
        h2.Add(ujHallgato);

        #endregion

        #region Kiírás
        Console.WriteLine($"'a' (int) értéke: {a}");
        Console.WriteLine($"'e' (string) értéke: {e}");
        Console.WriteLine($"'r1' (struct) neve: {r1.nev}");
        Console.WriteLine($"'h2' (lista) első elemének neve: {h2[0].nev}");
        #endregion
    }
  }
}
```