ELTE | IK

PROGRAMOZÁS

1. előadás

Horváth Győző, Horváth Gyula, Szlávi Péter

Bevezetés

Bevezetés

- Tárgy ismertetése
  - Módszeres feladatmegoldás számítógép segítségével
  - Eszközök bonyolultabb feladatok helyes megoldásához
    - Programozási tételek
  - Programozási alapok
- Követelmények
  - Folyamatos számonkérésű tárgy
  - http://progalap.elte.hu

A programkészítés folyamata

1. **Specifikálás** (miből?, mit?) → *specifikáció*
2. **Tervezés** (mivel?, hogyan?) → *adat- + algoritmus-leírás*
3. **Kódolás** (a gép hogyan?) → *kód* (reprezentáció + implementáció)
4. **Tesztelés** (hibás-e?) → *hibalista* (diagnózis)
5. **Hibakeresés** (hol a hiba?) → *hibahely*, *-ok*
6. **Hibajavítás** (hogyan jó?) → *helyes program*
7. **Minőségvizsgálat, hatékonyság** (jobbítható-e?, hogyan?) → *jó program*
8. **Dokumentálás** (hogyan működik, használható?) → *használható program*
9. **Használat, karbantartás** (még mindig jó?) → *időtálló program*

A specifikáció

Példa

Feladat:

Egy ötgyerekes nagycsalád nyaralni indul a Balatonra. Az autópályán a 11 éves, a számok iránt mindig nagy érdeklődést mutató Matyi nem látja a kilométerórát, de szeretné megtudni, mekkora sebességgel mennek. Így elkezdi megszámolni hány kilométer táblát hagytak el, és közben az időt is méri. Milyen eredményt kap?

Példa

Feladat:

Az út, idő ismeretében határozd meg a sebességet!

Példa adatok

Feladat:

Az út, idő ismeretében határozd meg a sebességet!

Példa:

120km; 1óra → 120km/óra

12km; 0,1óra → 120km/óra

10,5km; 0,1óra → 105km/óra

Adatok köre

Példa:

s=120; t=1 → v=120

s=10,5; t=0,1 → v=105

Megkap: s, t azonosítójú valós számok
Visszaad: v azonosítójú valós szám

- „Megkapni”
  - két számot
  - pontosabban: két valós számot
  - még pontosabban: s, t azonosítójú valós számokat
- „Visszaadni”
  - egy számot
  - pontosabban: valós számot
  - még pontosabban: v azonosítójú valós számot

Megszorítások

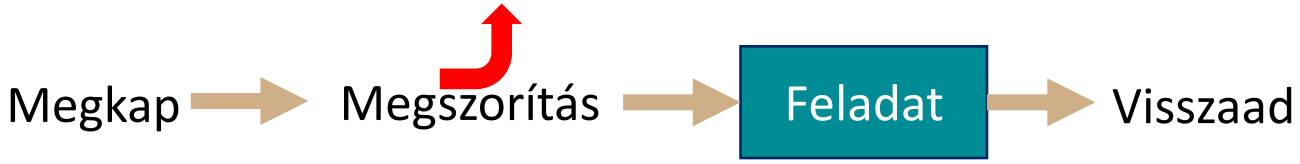
Mindent elfogadok?

s=-1; t=0 → ?

Megkap: s, t azonosítójú valós számok
Visszaad: v azonosítójú valós szám
Megszorítás: s nemnegatív, t pozitív

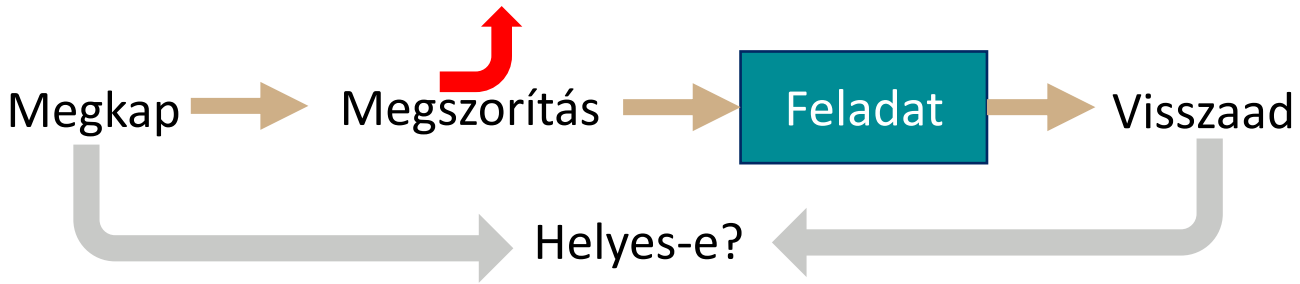
„Megkapni”:

s nemnegatív és t pozitív valós számokat.



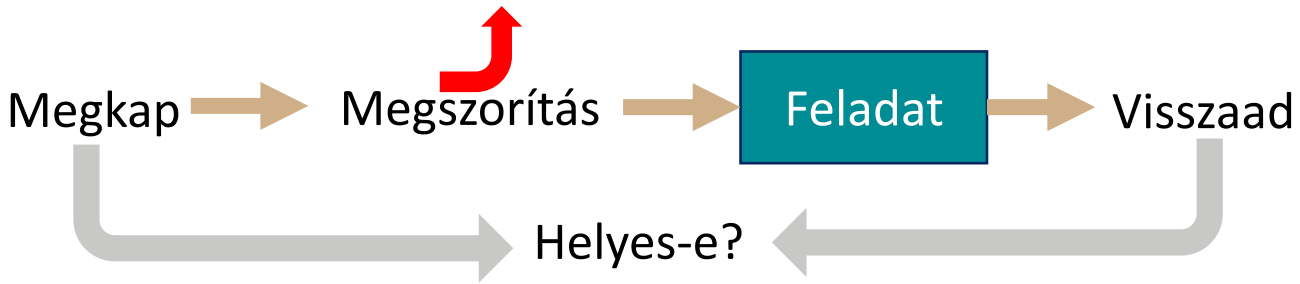
Helyes-e a feladat?

- Honnan tudjuk, hogy Matyi helyesen számolt?
- Honnan tudjuk, hogy a program **helyesen oldotta meg** a feladatot?
- Definiálni kell az összefüggést a kapott és a visszaadott adatok között!
  - v=s/t



Nem formálisan összegezve

- Megkapunk: s, t valós számokat
- Visszaadunk: v valós számot
- Megszorítás: s nemnegatív, t pozitív
- Megoldás: v = s / t

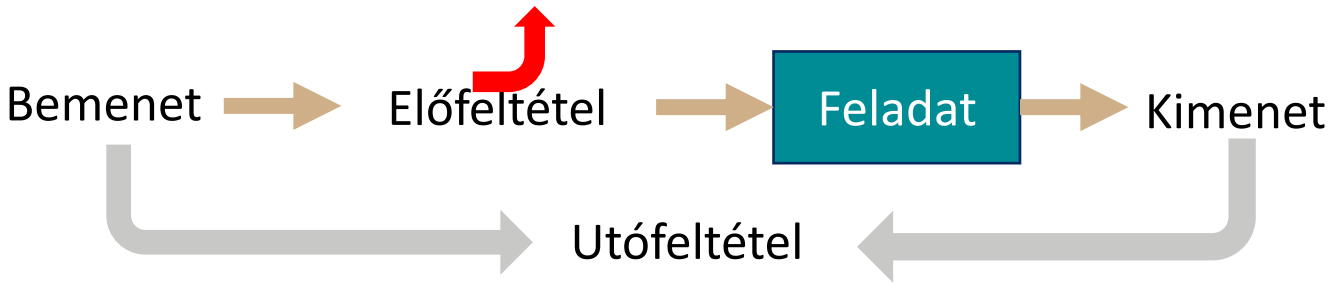


Példa: út-idő-sebesség

Feladat:

Az út, idő ismeretében határozd meg a sebességet!

```
Bemenet: s∈R, t∈R
Kimenet: v∈R
Előfeltétel: s>=0 és t>0
Utófeltétel: v = s / t
```



Példa: út-idő-sebesség

Feladat:

Az út, idő ismeretében határozd meg a sebességet!

Specifikáció:

```
Be: s∈R, t∈R
Ki: v∈R
Ef: s>=0 és t>0
Uf: v = s / t
```

R=Valós számok **halmaz**a

A specifikáció fogalma

Célja:

a feladat formális megragadása
szerződés a megbízó és a fejlesztő között

Kérdések:

- Mitől függ a megoldás? – *bemenet*
- Mi a megoldás? – *kimenet*
- Mit jelent: „megoldásnak lenni”? – *utófeltétel*
- Mindig/Mikor *van* megoldás? – *előfeltétel*

A specifikáció fogalma

Összetevői:

1. Bemenő adatok (azonosító, értékhalmaz [mértékegység])
2. Ismeretek a bemenetről (előfeltétel)
3. Eredmények (azonosító, értékhalmaz)
4. Az eredményt meghatározó logikai állítás (utófeltétel), amely a helyesen összetartozó adatokra igaz értéket ad
5. A használt fogalmak definíciói
6. A megoldással szembeni követelmények
7. Korlátozó tényezők

A specifikáció fogalma

Tulajdonságai:

1. „Egyértelmű”, pontos, teljes
2. Tömör (←formalizált)
3. Érthető, szemléletes (fogalmak)

A három szempont sokszor ellentmond egymásnak.

Specifikációs eszközök:

1. Szöveges leírás
2. Matematikai megadás

Jelölések

Megnevezés	Jelölés, halmaz	Példa
Egész szám	$\mathbb{Z}$	…;-2; -1; 0; 1; 2; …
Természetes szám	$\mathbb{N}$	0; 1; 2; 3; …
Valós szám	$\mathbb{R}$	10,234; π
Logikai érték	$\mathbb{L}$	igaz, hamis
Szöveg	$\mathbb{S}$	„alma”
Karakter	$\mathbb{C}$	„a”

Példa: út-idő-sebesség

Feladat:

Az út, idő ismeretében határozd meg a sebességet!

Helyes-e az alábbi utófeltétel?

```
Be: s∈R, t∈R
Ki: v∈R
Ef: s>=0 és t>0
Uf: s = v * t
```

Specifikáció mint függvény

Sebesség: R x R ⟶ R
Sebesség: R^2 ⟶ L
Sebesség(s, t):=v és v=s/t
Sebesség(s, t):=s/t

- **Bemenet**: `s∈R, t∈R`
  a függvény értelmezési tartománya: $\mathbb{R}\times\mathbb{R}=\mathbb{R}^2$ (amelynek egyes komponenseire lehet hivatkozni a specifikációban s-sel, t-vel)
- **Kimenet**: `v∈R`
  a függvény értékkészlete: $\mathbb{R}$ (amelyre hivatkozhatunk a specifikációban v-vel)
- **Előfeltétel**: `s≥0 és t>0`
  a függvény értelmezési tartományának ($\mathbb{R}^2$) szűkítése ($\mathbb{R}_{0,+}{}^2$)
- **Utófeltétel**: `v=s/t`
  mi igaz a végeredményre: a „kiszámítási szabály”

Az algoritmus

Az algoritmus

Hogyan oldjuk meg a feladatot?

A megoldás *elemi lépések*re bontása

Sebesség:

- s / t → v
- v ← s / t
- v := s / t

Az algoritmus fogalma

Elemi tevékenységek:

értékadás, beolvasás, kiírás.

Az algoritmusok **összeállítási mód**jai:

- Szekvencia (egymás utáni végrehajtás)
- Elágazás (választás 2 vagy több tevékenységből)
- Ciklus (ismétlés adott darabszámszor vagy adott feltételtől függően)
- Alprogram (egy összetett tevékenység, egyedi néven – absztrakció)

Algoritmusleíró nyelvek - pszeudokód

Szekvencia:

```
utasítás1
utasítás2
utasítás3
```

Elágazások:

```
Ha feltétel akkor
  utasítások igaz esetén
különben
  utasítások hamis esetén
Elágazás vége
```

kétirányú

```
Elágazás
  feltétel1 esetén utasítások1
  feltétel2 esetén utasítások1
  …
  különben utasítások
Elágazás vége
```

többirányú

Ciklusok:

```
Ciklus amíg feltétel
  utasítások
Ciklus vége
```

elöltesztelő

```
Ciklus
  utasítások
amíg feltétel
Ciklus vége
```

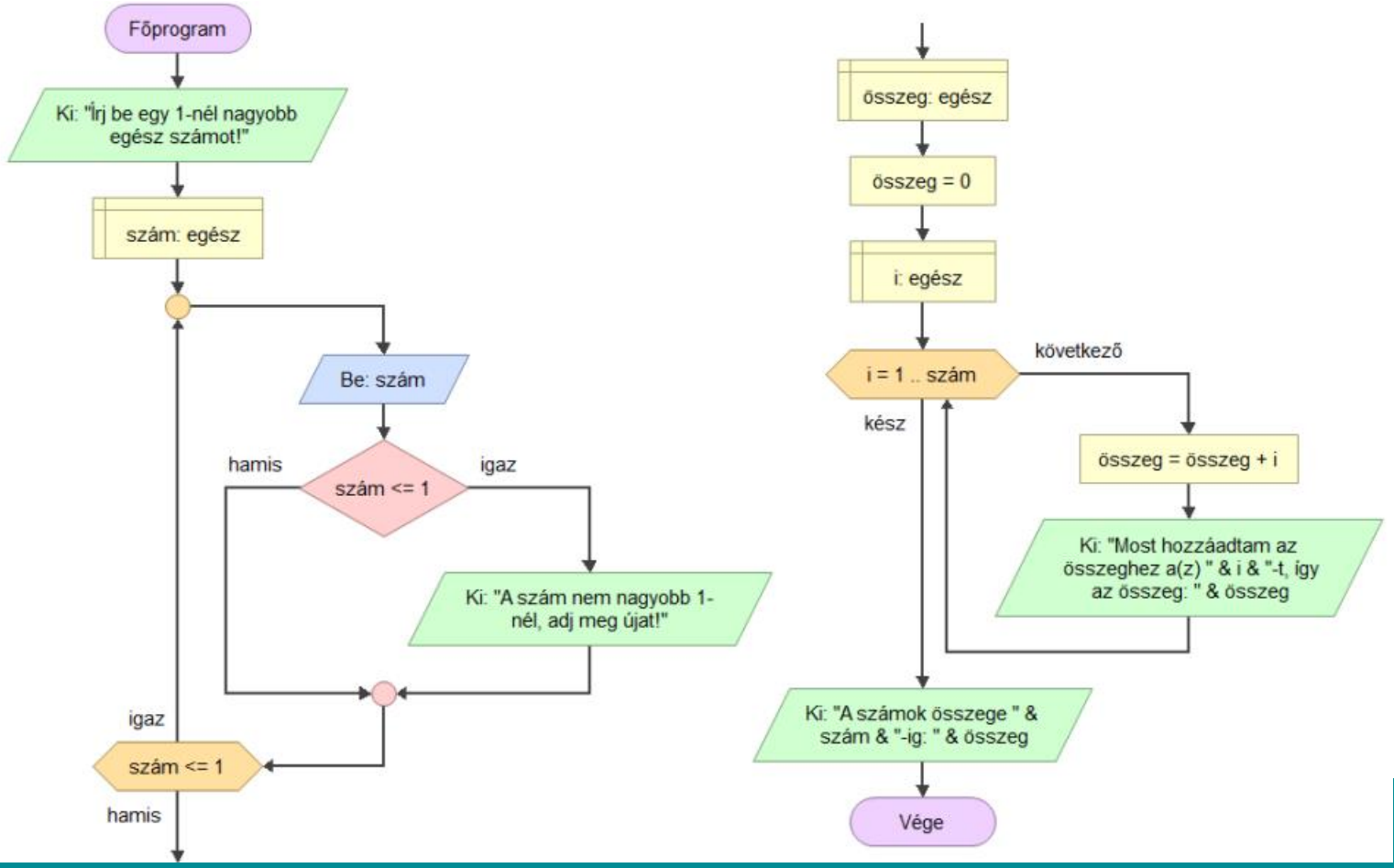
hátultesztelő

```
Ciklus i=e-től u-ig
  utasítások
Ciklus vége
```

számlálós

Algoritmusleíró nyelvek - folyamatábra

Szekvencia, elágazás, ciklus

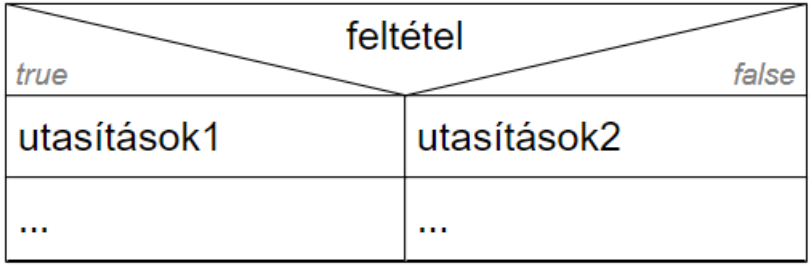


Algoritmusleíró nyelvek - Struktogram

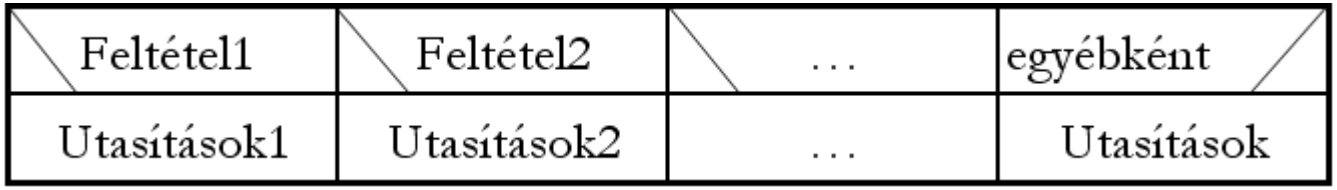
Szekvencia:

utasítás1
utasítás2
utasítás3

Elágazások:



kétirányú



többirányú

Algoritmusleíró nyelvek - Struktogram

Ciklusok:

számlálós

elöltesztelős

hátultesztelős

Algoritmusleíró nyelvek - Struktogram

- Struktogramszerkesztés
  - Táblázatkezelő/Szövegszerkesztő
  - Structorizer
  - CodeBlocks NSD szerkesztő
  - Online szerkesztő (speciális nyelv)

Specifikáció és megvalósítás

Specifikáció és megvalósítás:

A feladat specifikációja valós világbeli objektumokhoz rendel valamilyen valós világbeli eredményt. Emiatt valós világbeli dolgokkal (pontosabban azok absztrakciójával, pl. valós számok halmaza) foglalkozik.

A feladat számítógépes megoldása emiatt több részből áll

- a valós világbeli objektumokat leíró adatokat be kell juttatni a számítógépbe, annak memóriájában tárolni kell – ezek lesznek a megoldásbeli változók, amelyek típusa a számítógépes világ által elfogadott/megvalósított típusokból állhat (azaz pl. a számítógépes valós számok halmaza a matematika valós számhalmazának egy véges része lehet) – a specifikációban szereplő neveket (egyelőre) azonos nevű memóriabeli változókkal azonosítjuk;

Specifikáció és megvalósítás

- a memóriában megjelenő változókból valamilyen függvénnyel kiszámítjuk az eredményt, amit szintén a memóriában tároljuk – ezek neve (egyelőre) szintén megegyezik a specifikációban szereplő elnevezésekkel;
- végül az eredményt tartalmazó változók értékeit valahogyan kijuttatjuk a külvilágba.

Megjegyzés: lehetnek olyan változók is (látni fogjuk), amelyek a specifikációban nem jelennek meg.

Ebből alakul ki a klasszikus programok három fő lépése (= 3 algoritmus szekvenciája):

- az adatok beolvasása;
- az eredmény kiszámítása;
- az eredmény kiírása.

Specifikáció és algoritmus

- Specifikáció
  - Adathoz adatot rendel, adat → adat
  - Például: 10,5; 0,1 → 105
  - Az adatokra címkéken keresztül hivatkozunk
  - Például: s∈R, t∈R → v∈R

Specifikáció és algoritmus

- Algoritmus
  - A specifikációbeli címkékhez ugyanolyan nevű változókat hozunk létre
    - Például: s∈R → s: Valós
  - Az algoritmus végrehajtása előtt a bemeneti változók felveszik a bemeneti adatok értékeit (beolvasás)
  - Az algoritmus a megoldás során módosíthatja a változók értékét (feldolgozás)
  - Az algoritmus végrehajtása után a kimeneti változók a kimeneti adatok értékét kell tartalmazzák (kiírás)

Példa: sebesség

Valós: Valós számok **típusa**

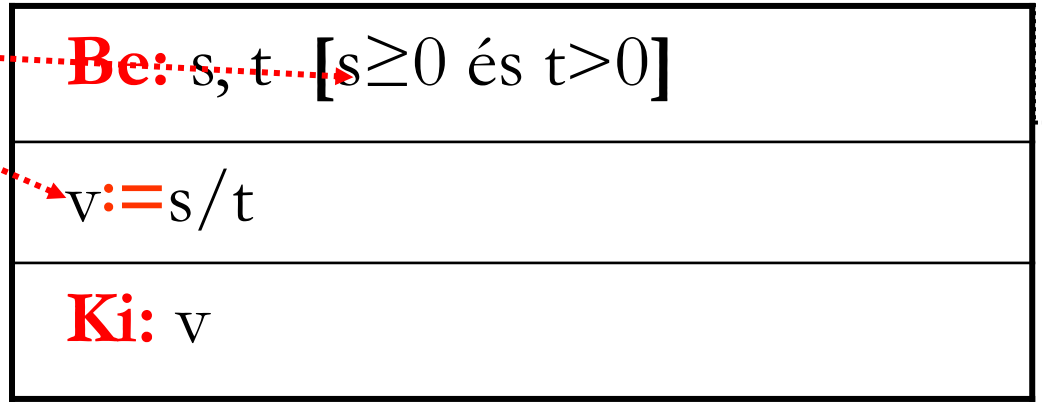
Algoritmus:

A programunk 4 fő részből áll: az adatok deklarálása, beolvasása, az eredmény kiszámítása, az eredmény kiírása:

Specifikáció:

```
Be: s∈R, t∈R
Ki: v∈R
Ef: s>=0 és t>0
Uf: v = s / t
```

Változó
s, t:**Valós**
v:**Valós**

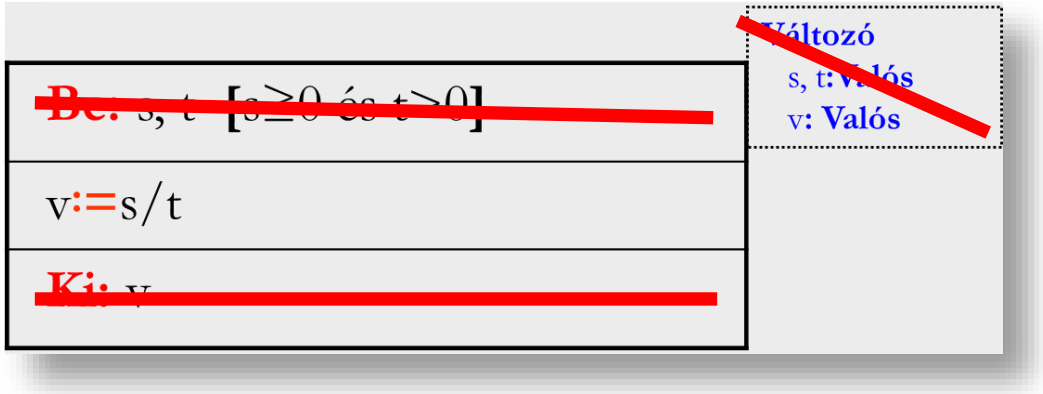


A deklarációt, az „elemi” utasításokat egy-egy „dobozba” írjuk.

Példa: sebesség

Algoritmus:
A be- és kimenetet nem algoritmizáljuk!
A specifikációból egyértelműen származtatott deklarációkat nem tüntetjük fel.

Specifikáció:
```
Be: s∈R, t∈R
Ki: v∈R
Ef: s>=0 és t>0
Uf: v = s / t
```



A kód

Kód keret

```
namespace sebesseg {
    internal class Program {
        static void Main(string[] args) {
            // Deklaráció

            // Beolvasás

            // Feldolgozás

            // Kiírás

        }
    }
}
```

Kód megoldás

```
// Deklaráció
double s, t;
double v;
// Beolvasás
Console.Write("s = ");
double.TryParse(Console.ReadLine(), out s);

Console.Write("t = ");
double.TryParse(Console.ReadLine(), out t);

// Feldolgozás
v = s / t;

// Kiírás
Console.WriteLine("A sebesség: {0}", v);
```

Kód: deklaráció

Specifikáció	Algoritmus	Kód
$\mathbb{Z}$	Egész	sbyte, short, **int**, long
$\mathbb{N}$	Természetes	byte, ushort, **uint**, ulong
$\mathbb{R}$	Valós	float, **double**
$\mathbb{L}$	Logikai	bool
$\mathbb{S}$	Szöveg	string
$\mathbb{C}$	Karakter	char

Változó s:Valós → `double s;`

Kód: beolvasás, kiírás

- Beolvasás

```
Console.Write("s = ");
double.TryParse(Console.ReadLine(), out s);
```

- Kiírás

```
Console.WriteLine("A sebesség: {0}", v);
```

- Feldolgozás
  - Értékadás: `=` operátor

```
v = s / t;
```

Összefoglalva – specifikáció

Feladat:

Egy ötgyerekes nagycsalád nyaralni indul a Balatonra. Az autópályán a 11 éves, a számok iránt mindig nagy érdeklődést mutató Matyi nem látja a kilométerórát, de szeretné megtudni, mekkora sebességgel mennek. Így elkezdi megszámolni hány kilométer táblát hagytak el, és közben az időt is méri. Milyen eredményt kap?

Feladat:

Az út, idő ismeretében határozd meg a sebességet!

Specifikáció:

```
Be: s∈R, t∈R
Ki: v∈R
Ef: s>=0 és t>0
Uf: v = s / t
```

Összefoglalva – algoritmus

Specifikáció:

```
Be: s∈R, t∈R
Ki: v∈R
Ef: s>=0 és t>0
Uf: v = s / t
```

Változó
s, t: **Valós**
v: **Valós**

Be: s, t [s≥0 és t>0]
v:=s/t
Ki: v

v:=s/t

Összefoglalva – kód

Specifikáció:

```
Be: s∈R, t∈R
Ki: v∈R
Ef: s>=0 és t>0
Uf: v = s / t
```

Változó

s, t: **Valós**

v: **Valós**

Be: s, t [s≥0 és t>0]

v:=s/t

Ki: v

```
// Deklaráció
double s, t;
double v;
// Beolvasás
Console.Write("s = ");
double.TryParse(Console.ReadLine(), out s);

Console.Write("t = ");
double.TryParse(Console.ReadLine(), out t);

// Feldolgozás
v = s / t;

// Kiírás
Console.WriteLine("A sebesség: {0}", v);
```

Adatokkal kapcsolatos fogalmak

Adatokkal kapcsolatos fogalmak

- **Konstans**
  Az az adat, amely a műveletvégzés során nem változtathatja meg értékét, mindvégig ugyanabban az „állapotban” marad.
- **Változó**
  Az ilyen adatféleségnek lényegéhez tartozik a „változékony-ság”, más szóval: vonatkozhatnak rá olyan műveletek is, amelyek új értékkel látják el.
  Tudományosabban fogalmazva: végrehajtás során megváltozhat az állapothalmaza.

Adatokkal kapcsolatos fogalmak

- **Változók fajtái céljuk szerint**
  - bemeneti változó: bemenetkor kap értéket
  - eredmény: kiszámítás tartozik hozzá
  - részeredmény: kiszámítás tartozik hozzá, belőle további kiszámítások indulnak
  - … *(lesznek még továbbiak)*

Adatokkal kapcsolatos fogalmak

- **Értékadás**
  Az az utasítás, amely révén a pillanatnyi állapotból egy meghatározott állapotba kerül a változó. (Nyilvánvaló, hogy konstans adatra nem vonatkozhat értékadás, az egy, kezdő-értéket meghatározón kívül.)

- **Típus**
  Olyan „megállapodás” (absztrakt kategória), amely adatok egy lehetséges körét jelöli ki az által, hogy rögzíti azok állapothalmazát és az elvégezhető műveletek készletét.

Az adatjellemzők összefoglalása

Azonosító

Az a jelsorozat, amellyel hivatkozhatunk a tartalmára, amely által módosíthatjuk tartalmát.

Kezdőérték

A születéskor hozzárendelt érték.
Konstansoknál nyilvánvaló, hogy deklarációban kapja; változóknál akár deklarációban, akár futáskor szerez értéket magának.

A típus

Összetettség (strukturáltság) szempontjából beszélhetünk

- strukturálatlan (vagy skalár, elemi) típusról, ha (az adott szinten) szerkezetet nem tulajdonítunk neki; vagy
- strukturált (más szóval: összetett) típusról, ha (elemibb) összetevőkre bontjuk.

Elemi típusok

Egész típus

- Értékhalmaz: $-2^{31}..+2^{31}-1$
  (Min'Egész..Max'Egész)
- Műveletek: +, – , *, Div (egészosztás) ,
  Mod (osztási maradék), – (unáris mínusz),
  ^ (pozitív egészkitevős hatványozás)
- Relációk: =, <, ≤, ≠, ≥, >
- Ábrázolás: kettes komplemens kódú
- Változatai: méret és előjel szerint sokfélék

Példaként: 4-bájtos ábrázolást feltételezve.

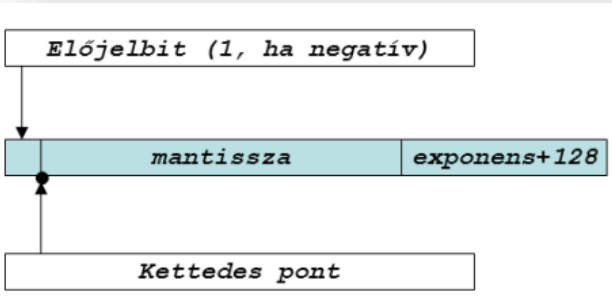
A beolvasáson, a kiíráson és értékadáson túliakkal foglalkozunk csak.

Pl. 3-bites 2-es komplemens kódú egész számok:
$+0=0|00_2$, $+1=0|01_2$, $+2=0|10_2$, $+3=0|11_2$,
$-1=1|11_2$, $-2=1|10_2$, $-3=1|01_2$, $-4=1|00_2$,
Vegye észre a „szabályszerűségeket”!

Elemi típusok

Valós típus

- Értékhalmaz: ????..????
  (Min'Valós..Max'Valós nem definiáltak, vagy reprezentáció-függőek)
- Műveletek: +, –, *, /, ^, – (unáris mínusz)
- Relációk: =, <, ≤, ≠, ≥, >
- Ábrázolás: lebegőpontos ábrázolás (pontosabb lenne, ha e típust racionálisnak neveznénk, mert csak racionális számot képes ábrázolni)



Pl.: 0 = (0000… 000…00) = $0*2^0$
-0.5 = (1100… 100…00) = $-0.5*2^0$
8 = (0100… 10…100) = $0.5*2^4$

Mivel $m*2^e$ alakra hozott számot ábrázol, ahol $m \in [0.5,1)$, ezért a 0.5-hez tartozó bit mindig 1, vagyis ekkor elhagyható; mi a példában megtartottuk a könnyebb ellenőrizhetőség miatt. A 0 számnál m=0 és e=0.

Elemi típusok

Logikai típus

- Értékhalmaz: Hamis..Igaz
  (Min'Logikai..Max'Logikai: Hamis, illetve Igaz)
- Műveletek: nem, és, vagy (a szokásos logikai műveletek)
- Relációk: =, <, ≤, ≠, ≥, >
- Ábrázolás: 0B = Hamis, –1B = Igaz
  (esetleg: 1B = Igaz)... ahol xB = x érték „bináris egészként” ábrázolva
- Megjegyzés: a rendezésnek nem nagy a gyakorlati jelentősége.

Elemi típusok

Karakter típus

- Értékhalmaz: 0..255 - kódú jelek – ASCII
  (Min'Karakter..Max'Karakter: a 0, illetve a 255 kódú karakter)
- Műveletek: karakter-specifikus nincs
  (esetleg a Kód:Karakter→Egész függvény, és inverze a Karakter:Egész→Karakter függvény, amelyek a belső ábrázolással hozza kapcsolatba)
- Relációk: =, <, ≤, ≠, ≥, >
  (a belső ábrázolásuk alapján → nem ABC-sorrend!)

Elágazás

Háromszög

Feladat:

3 szám lehet-e egy derékszögű háromszög 3 oldala?

Megoldás:

- Akkor lehet, ha $a^2 + b^2 = c^2$
- Akkor nem lehet, ha ez nem teljesül

Példa: háromszög

Tessék kipróbálni az alábbi adatokkal!

```
a: 3
b: 4
c: 5.000000000000001
lehet: false
```

Feladat:

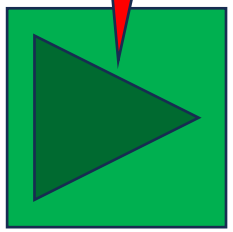
3 szám lehet-e egy derékszögű háromszög 3 oldala?

Specifikáció:

```
Be: a∈R, b∈R, c∈R
Ki: lehet∈L
Ef: a>0 és b>0 és c>0
Uf: (a*a+b*b=c*c és lehet=igaz) vagy
    (a*a+b*b≠c*c és lehet=hamis)
```

R=Valós számok **halmaza**

L=Logikai értékek **halmaza**



Megjegyzés: a 3 szám sorrendjét ezek szerint implicite rögzítettük – c az átfogó hossza!

Példa: háromszög

Feladat:

3 szám lehet-e egy derékszögű háromszög 3 oldala?

Specifikáció$_2$:

```
Be: a∈R, b∈R, c∈R
Ki: lehet∈L
Ef: 0<a és a<=b és b<=c
Uf: (a*a+b*b=c*c és lehet=igaz) vagy
    (a*a+b*b≠c*c és lehet=hamis)
```

Megjegyzés: a 3 szám sorrendjét ezek szerint explicite rögzítettük
– c az átfogó hossza!

Példa: háromszög

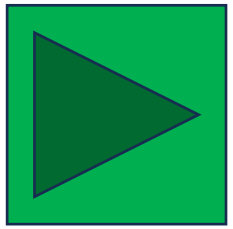
Implikáció:

a „ha-akkor” logikai kifejezése

a	b	a->b
igaz	igaz	igaz
igaz	hamis	hamis
hamis	igaz	igaz
hamis	hamis	igaz

```
Uf: (a*a+b*b=c*c és lehet=igaz) vagy
    (a*a+b*b≠c*c és lehet=hamis)

Uf: (a*a+b*b=c*c -> lehet=igaz) és
    (a*a+b*b≠c*c -> lehet=hamis)
```

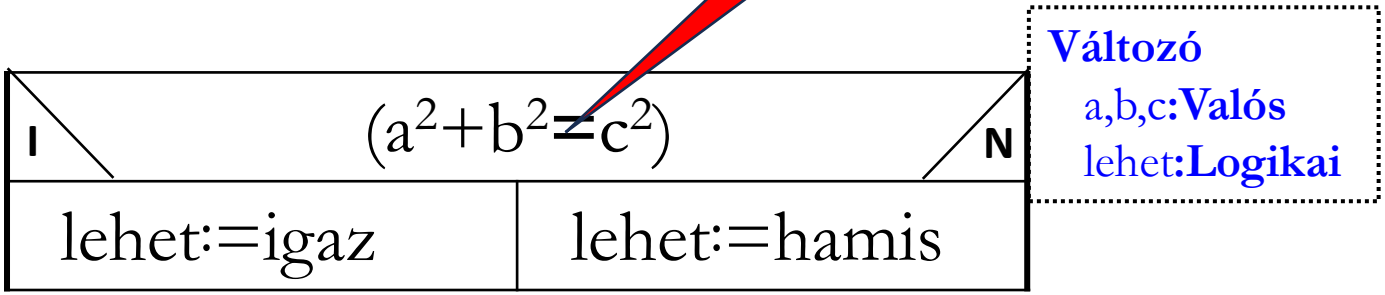


Példa: háromszög

A valós típusú azonosság körül adódhatnak problémák a valós típusú adatok ábrázolása miatt.
A precíz megoldás:
$|a^2+b^2-c^2|<epszilon$

```
Be: a∈R, b∈R, c∈R
Ki: lehet∈L
Ef: 0<a és a<=b és b<=
Uf: (a*a+b*b=c*c ->      c=igaz) és
    (a*a+b*b≠c*c      lehet=hamis)
```

Algoritmus:



Változó
a,b,c:**Valós**
lehet:**Logikai**

Kód megoldás

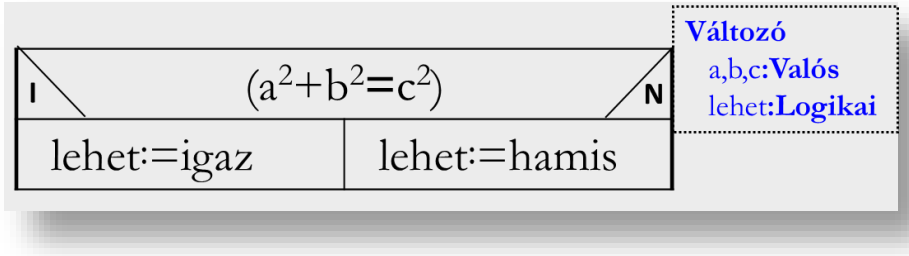
A valós típusú azonosság körül adódhatnak problémák a valós típusú adatok ábrázolása miatt.
A precíz megoldás:
Abs(a*a+b*b-c*c)<epszilon

```
// Deklaráció
double a, b, c;
bool lehet;
// Beolvasás
Console.Write("a = ");
double.TryParse(Console.ReadLine(), out a);

Console.Write("b = ");
double.TryParse(Console.ReadLine(), out b);

Console.Write("c = ");
double.TryParse(Console.ReadLine(), out c);

// Feldolgozás
if (a * a + b * b == c * c) {
  lehet = true;
}
else {
  lehet = false;
}
// Kiírás
Console.WriteLine("Lehet derékszögű háromszög: {0}", lehet);
```



Példa: háromszög

Specifikáció$_3$:

```
Be: a∈R, b∈R, c∈R
Ki: lehet∈L
Ef: 0<a és a<=b és b<=c
Uf: lehet=(a*a+b*b=c*c)
```

Algoritmus:

$$\text{lehet} := (a^2 + b^2 = c^2)$$

Példa: háromszög

Egy másik **algoritmus** a lényegi részre:

Segéd változók deklarálása „széljegyzetként”

Változó
aa,bb,cc:**Valós**

$aa:=a^2$
$bb:=b^2$
$cc:=c^2$
$lehet:=(aa+bb=cc)$

Bevezethetők/-endők segéd (belső, saját) változók.

Kód megoldás

```
// Deklaráció
double a, b, c;
bool lehet;
// Beolvasás
Console.Write("a = ");
double.TryParse(Console.ReadLine(), out a);

Console.Write("b = ");
double.TryParse(Console.ReadLine(), out b);

Console.Write("c = ");
double.TryParse(Console.ReadLine(), out c);

// Feldolgozás
lehet = (a * a + b * b == c * c);

// Kiírás
Console.WriteLine("Lehet derékszögű háromszög: {0}", lehet);
```

Kód megoldás

```
// Deklaráció és beolvasás ld. korábban
// Előfeltétel ellenőrzés
if (0 < a && a <= b && b <= c) {
    // Feldolgozás
    lehet = (a * a + b * b == c * c);

    // Kiírás
    if (lehet) {
        Console.WriteLine("Lehet derékszögű háromszög");
    }
    else {
        Console.WriteLine("Nem lehet derékszögű háromszög");
    }
}
else {
    Console.WriteLine("Nem megfelelő értékek! Futtassa újra!");
}
```

Másodfokú egyenlet

Feladat: *Adjuk meg a másodfokú egyenlet egy megoldását!*
Az egyenlet: $ax^2+bx+c=0$.

Specifikáció$_1$:

```
Be: a∈R, b∈R, c∈R
Ki: x∈R
Ef: -
Uf: ax²+bx+c=0
```

Megjegyzés: az uf. nem ad algoritmizálható információt. Nem baj, sőt tipikus, de … próbálkozzunk még!

Megoldóképlet: $$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4*a*c}}{2*a}$$

Másodfokú egyenlet

Specifikáció$_2$:

```
Be: a∈R, b∈R, c∈R
Ki: x∈R
Ef: a≠0
Uf: x=(-b+sqrt(b*b-4*a*c))/(2*a)
```

Nyitott kérdések:

- *Mindig/Mikor van megoldás?*
- *Egy megoldás van?*

Másodfokú egyenlet

Specifikáció:

```
Be: a∈R, b∈R, c∈R
Ki: x∈R, van∈L
Ef: a≠0
Uf: van=(b*b-4*a*c>=0) és
    van->x=(-b+sqrt(b*b-4*a*c))
           /(2*a)
```

Másodfokú egyenlet

Kimenetbővítés:

```
Ki: x∈R, van∈L
Uf: van=(b*b-4*a*c>=0) és
    van -> x=(-b+sqrt(b*b-4*a*c))/(2*a)
```

Nyitott kérdés:

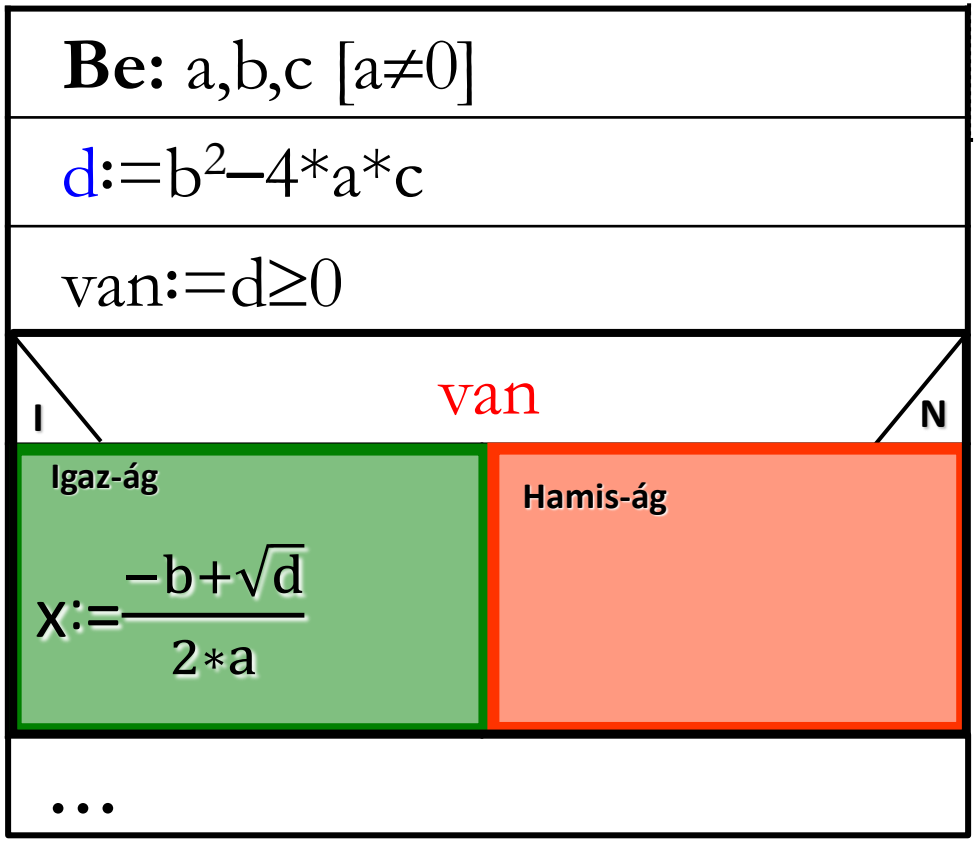
- *Egy megoldás van? – hf .*

Másodfokú egyenlet

Algoritmus:

Specifikáció:

```
Be: a∈R, b∈R, c∈R
Ki: x∈R, van∈L
Ef: a≠0
Uf: van=(b*b-4*a*c>=0) és
    van->x=(-b+sqrt(b*b-4*a*c))
           /(2*a)
```



Változó
a,b,c,x:**Valós**
van:**Logikai**
d:**Valós**

Összefoglalás

Adatok, típusok, változók

Specifikáció	Algoritmus	Kód
$\mathbb{Z}$	Egész	sbyte, short, **int**, long
$\mathbb{N}$	Természetes	byte, ushort, **uint**, ulong
$\mathbb{R}$	Valós	float, **double**
$\mathbb{L}$	Logikai	bool
$\mathbb{S}$	Szöveg	string
$\mathbb{C}$	Karakter	char

Specifikáció	Algoritmus	Kód
Be: a∈R	a: Valós	`double a;`

Szekvencia

Algoritmus:

utasítás1
utasítás2
utasítás3

Kód:

```
utasítás1;
utasítás2;
utasítás3;
```

Elágazás

Algoritmus:

feltétel	
true	*false*
utasítások1	utasítások2
...	...

kétirányú

Feltétel1	Feltétel2	...	egyébként
Utasítások1	Utasítások2	...	Utasítások

többirányú

Kód:

```
if (feltétel) {
  utasítások1;
} else {
  utasítások2;
}
```

```
if (feltétel1) {
  utasítások1;
}
else if (feltétel2) {
  utasítások2;
}
// ...
else {
  utasítások;
}
```

Megfelelések

Algoritmus	Kód
:=	=
=	==
és	&&
vagy	||
nem	!
Be: a	`Console.Write("a = ");` `double.TryParse(Console.ReadLine(), out a);`
Ki: a	`Console.WriteLine(a);` `Console.WriteLine("a = {0}", a);`

Ellenőrző kérdések

Kérdések

- Milyen lépésekből áll a programkészítés folyamata?
- Mi a specifikáció? Milyen részei vannak? Mi a célja?
- Mi a szerepe a specifikáció egyes részeinek?
- Mi az algoritmus, milyen elemi tevékenységeket tartalmaz?
- Milyen összeállítási módjai vannak az algoritmusnak?
- Hogyan néznek ki a különböző vezérlési szerkezetek struktogrammal írva?
- Hogyan lesz a specifikációból megvalósítás? Hogyan függ össze a specifikáció és az algoritmus?
- Az út, idő ismeretében határozd meg a sebességet! Írd le a feladat specifikációját!
- Számítsuk ki az oldalhosszak ismeretében egy téglalap területét! Írd le a feladat specifikációját!