Bevezetés a programozásba
INF-501

Algoritmusok és adatábrázolás

Kógelmann Gábor
főiskolai docens

Informatikai Intézet
112 - es szoba

1. Az algoritmusszerkesztés alapfogalmai

Az elektronikus számítógépek által végrehajtandó feladatokat a gép számára érthető nyelven kell megfogalmazni. Erre a feladatra számítógépi programokat használunk.

A számítástechnika történetének legjelentősebb programnyelvei, kialakulásuk időrendjében:

- ALGOL (**ALGO**rithmic **L**anguage)
- FORTRAN (**FOR**mula **TRAN**slator)
- COBOL (**CO**mmon **B**usiness **O**riented **L**anguage)
- PL/I (**P**rogramming **L**anguage **I**)
- PASCAL
- C, C++
- Modula stb...

A fenti, úgynevezett **magasszintű programnyelv**ek mellett minden számítógép architektúra (mikroprocesszor) számára létezik egy gépközeli programnyelv, amit **ASSEMBLY szintű programnyelv**nek hívnak.

A program utasítások sorozata, amelyeket az adott programnyelv **formai** (**szintaktikai**) szabályainak betartásával kell alkalmazni. A programok tartalmi ("működési") helyessége a **szemantikai** programhelyesség.

Az **algoritmikus gondolkodásmód**, a **programozási logika** elsajátításához nincs szükség konkrét programnyelv ismeretére.

Tananyagunk e fejezete, az algoritmuskészítés alapjainak megismertetésére vállalkozik.

1.1 Az algoritmus fogalma

A számítógép általános megfogalmazás szerint egy adatok feldolgozására készített eszköz. Ilyen megközelítés alapján az adatok két csoportba sorolhatók:

- A program
- A program által kezelt adat

1.1.1 A program és az algoritmus

A program a megoldáshoz vezető utat tartalmazza, azaz azt írja le, hogy milyen műveleteket kell elvégezni a **tényleges adatokkal**, hogy a kívánt eredményhez jussunk.
A **program** egy adott programnyelven megvalósított végrehajtási **algoritmus.** Az algoritmus általában nem kötődik semmilyen konkrét programnyelvhez. Az a mód-

szer vezet jó eredményre, ha előbb elkészítjük a feladat megoldásának algoritmusát, majd ehhez választunk megfelelő programnyelvet.

Definíció:

Algoritmusnak nevezzük az aritmetikai, logikai, stb... műveletek olyan célszerűen összeállított sorozatát, amely a kitűzött feladat egyértelmű megoldásához vezet.

A feladatok általában bonyolultak, azaz sok esetben több száz, sőt több ezer elemi lépést tartalmaznak. Ezért a feladatot célszerű részfeladatokra bontani.
Ebben az esetben mindig van egy **főprogram**, és több-kevesebb **alprogram**. Az alprogramokat szokás **eljárásnak, szubrutinnak** is nevezni. A feladat megoldásnak ezt a megközelítését **strukturált programozásnak** nevezzük.

A főprogram eljárást, vagy eljárásokat hív meg, melyek aztán további eljárásokat hívhatnak. Az eljárások befejeztével a vezérlés a hívás utáni utasításra kerül vissza. Az eljáráshíváskor **paraméter**ek adhatók át a hívottnak. Ennek segítségével lehet az általános feladatmegoldást az aktuális igényhez igazítani.

Követelmények az algoritmusokkal szemben:

- Legyen általános, amennyire ezt az adott feladat megengedi.
- Szélső esetekben is helyes eredményt adjon.
- Véges számú lépés (idő) után fejeződjön be.

Az algoritmusok fajtái:

- Verbális algoritmus
- Jel algoritmus

A **verbális algoritmus** szövegesen adja meg a feladatot. (Élőszóban, vagy írásban.)
Az írásban megadott feladatmegoldást szokás **pszeudó kód**nak nevezni.

A **jel algoritmus** a feladat megfogalmazására geometriai szimbólumokat, matematikai jelöléseket használ.

A leggyakrabban használt jel algoritmusok:

- Folyamatábra
- Struktogram
- Szerkezeti ábra

1.1.2 A program által kezelt adatok

Az adatok egyik nagy csoportját a **konstans**ok alkotják. **Értékük** a program futtatása során **nem változhat** meg.
A másik csoportot a **változó**k képezik. Ezek **értéke** a program futtatása során **megváltozhat**. A változó is kétféle lehet: **egyszerű** (skalár) és **összetett**.

Az egyszerű változók három jellemző tulajdonsága:

- Név
  Általában az angol ABC betűiből, a számokból, és néhány különleges jelből képezzük. Célszerű a tartalomra utaló elnevezést választani.
  Egy memóriaterületet szimbolizál.

- Típus
  Ez a jellemző utal arra, hogy a memóriában az adatot milyen formában tároljuk, illetve azt is behatárolja, hogy milyen jellegű műveletet

  végezhetünk vele.

- A változó címe a memóriában
  Azt írja le, hogy az adat milyen kezdőcímtől helyezkedik el a memóriában.

Az összetett változók jellemző tulajdonságai:

- Megegyezik az előző hárommal
- A kezelt adatcsoport jellemzői
  Eszerint beszélünk **tömb**ről és **halmaz**ról.
  - Tömb
    Jellemző tulajdonsága a **dimenzió**.
    Egy, két, három, esetleg több dimenzió.
    Az egydimenziós tömb neve: **vektor**
    A[3] ; B[12]
    A kétdimenziós tömb neve: **mátrix**
    C[3,2] ; D[5,5]
    A tömb elemeire az **indexel** hivatkozunk.
    C[1,1] - Az első sor, első oszlopa
    C[3,2] - A harmadik sor második oszlopa
  - Halmaz
    Jellemző tulajdonsága, hogy elemeit mely adatok közül választhatjuk ki.
    H1 = [12, 34, 7, 23, 12]
    H2 = [A, B, a, b]
    H3 = [ ] - Üres halmaz.

Az egyes adat típusokon értelmezett műveletek:

- Numerikus (valós, egész)
  - A négy alapművelet (+ - * /)
  - A hatványozás (^ vagy **)
  - Egyváltozós műveletnél az előjel (-)

- Csak egész típusú adatok esetén
  - Egészosztás (DIV) 7 DIV 2 → 3
  - Maradékképzés (MOD) 7 MOD 2 → 1

- Karakter és sztring típusú adatok
  - Egymásutánírás (+) (konkatenálás)
    "A" + "1" → "A1"
    "ABCD" + "EFG" → "ABCDEFG"

- Logikai adatok
  - ÉS művelet (AND)
  - VAGY művelet (OR)
  - KIZÁRÓ VAGY művelet (XOR)
  - TAGADÁS művelet (NOT)
  - Az igazságtáblázat:

a	b	a AND b	a OR b	a XOR b	NOT a
1	1	1	1	0	0
1	0	0	1	1	
0	1	0	1	1	1
0	0	0	0	0	

0 - Hamis (FALSE)
1 - Igaz (TRUE)

Relációs műveletek:
- Két azonos típusú konstans, illetve változó(k) között értelmezhető.
- = , > , < , >= , <= , <> (Nem egyenlő)
- A művelet eredménye egy logikai érték (1 , 0)

A = 5; B = 6; A > B → 0
A < B → 1
A = B → 0

A halmazokon értelmezett műveletek:
- Mindig kétoperandusúak.
- Halmazok összege (+)
  Az azonosak csak egyszer kerülnek az eredmény halmazba.
- Halmazok szorzata (*)
  Azok kerülnek az eredmény halmazba, melyek mindkét halmazban előfordulnak.
- Halmazok különbsége (-)
  Az első operandus elemeiből elmaradnak azok az elemek, amelyek a másodikban is megvannak.

A változók függvények segítségével is kaphatnak értéket. Példaként néhány, majdnem minden programnyelvben előforduló, beépített függvény:
- Matematikai
  - ABS(x) x numerikus
  - COS(x) x numerikus
  - SQRT(x) x nemnegatív numerikus

- Konverziós
  - CHR(x) x numerikus (0 - 255)
  - STR(x) x numerikus
  - VAL(x) x sztring

- Sztring-kezelő
  - LEFT(x,y) x sztring, y nemnegatív egész
  - LENGTH(x) x sztring

Értékadó kifejezések:
Azt, hogy milyen adatokon, milyen műveletet kell elvégezni **kifejezés**sel adjuk meg.

A kifejezés bal oldalán változónak kell lennie, a jobb oldalon előfordulhatnak változók, konstansok, függvényhívások (visszatérő értékkel) és műveleti jelek. A két oldal között az egyenlőségjel áll.
A jobboldali változó(k) értéke a műveletvégzés során nem változik meg.

A műveletek elvégzési sorrendje kötött. A sorrend **zárójel**ekkel befolyásolható. Több zárójel esetén a kiértékelés a belső zárójelpártól indul. A műveleteknek **precedenciá**ja (prioritása) van. A precedencia-szabály részben programnyelv függő, de általában az alábbi:

- előjel
- negáció
- hatványozás
- szorzás, osztás
- összeadás, kivonás
- reláció műveletek
- stb.

A felsorolás csökkenő prioritás szerinti.
Az azonos prioritásszinten a kiértékelés általában balról jobbra irányú.

Példa:

```
a = -b + c * 2      ;     a = -(b + c) * 2      ;      a = (-b + c) * 2
z = ABS(x) + y
i  = i + 1
```

Az utóbbi példa olvasása: Olvasd ki **i** értékét, növeld meg eggyel, és az eredményt helyezd vissza **i** - be.

1.2 Az algoritmusokban előforduló tevékenységek

Alaptevékenységek:

- Adatok beolvasása megadott változókba.
  (**Input** tevékenység)
- Adatok kiírása. Az adat lehet konstans, változó tartalom, és kifejezés értéke.
  (**Output** tevékenység)
- A kifejezések értékének kiszámítása.
  (**Értékadó** tevékenység)

Vezérlőtevékenységek:

- A számítógép az egyes műveleteket az utasítások leírásának sorrendjében hajtja végre.
  (**Szekvenciális** vezérlőtevékenység)
- Egy kifejezés értékétől függően kiválasztja a következő tevékenységet.
  (**Szelekciós** vezérlőtevékenység)
- Egy vagy több tevékenységet, egy feltételtől függően ismétel.
  (**Iterációs** vezérlőtevékenység)

1.2.1 Szekvenciális vezérlőtevékenység

Másnéven szekvenciális (soros) végrehajtási folyamat. A legtöbb adatfeldolgozási folyamat ezt a logikát követi.

```
.......
A - tevékenység
B - tevékenység   |  A végrehajtás sorrendje
C - tevékenység   ↓
.......
```

1.2.2 Szelekciós vezérlőtevékenység

Feltételtől függő végrehajtási folyamat. Olyan feltételt tartalmaz, amely több alternatív tevékenység közül egynek a kiválasztásához vezet.

Típusai:

- Egyszerű alternatíva

  IF feltétel **THEN**
  [A - modul]
  (Az IF szerkezet vége)

- Kettős alternatíva

```
IF      feltétel      THEN
        [A - modul]
ELSE
        [B - modul]
(Az IF szerkezet vége)
```

- Összetett alternatíva

```
IF      feltétel(1)           THEN
        [A(1) - modul]
ELSE  IF  feltétel(2)     THEN
        [A(2) - modul]
        ......
        ......
ELSE  IF  feltétel (n)    THEN
        [A(n) - modul]
ELSE
        [B - modul]
(Az IF szerkezet vége)
```

A modul egy, vagy több tevékenységet (utasítást) foglal magába.

1.2.3 Iterációs vezérlőtevékenység

Feltételtől függő, ismétlődő végrehajtási folyamat, másnéven **ciklus**. Az ismétlődő utasítás(ok) alkotják a **ciklustörzset (**ciklusmagot). Az iterációs vezérlőtevékenység másik része a **ciklusfej**.

Általános esetben a ciklusfej résztevékenységei:

- Ciklusváltozó kezdeti értékadása (inicializálás)
- A ciklusváltozó tesztelése (a ciklusból való kilépés érdekében)
- A ciklusváltozó értékének módosítása

Konkrét programnyelvtől függően az első, illetve harmadik résztevékenység el is maradhat.

Típusai:

- Előltesztelő - WHILE típusú ciklus

```
WHILE  feltétel
        [modul (ciklusmag)]
(A WHILE szerkezet vége)
```

- Előltesztelő - FOR típusú ciklus

```
FOR  i = k  TO  v  BY  m
        [modul (ciklusmag)]
(A FOR szerkezet vége)
```

A betűk (változók) jelentése:

i - Ciklusváltozó
k - A ciklusváltozó kezdőértéke
v - A ciklusváltozó végértéke
m - A ciklusváltozó módosító értéke

- Hátultesztelő - DO - WHILE típusú ciklus

DO
[modul (ciklusmag)]
WHILE feltétel
(A WHILE szerkezet vége)

A hátultesztelő ciklus esetén a ciklusmag legalább egyszer végrehajtódik.

Ha a ciklusváltozó tesztelése elmarad, (esetleg hibás) úgynevezett **végtelen ciklus**t kapunk.
A ciklusok a gyakorlati feladatok megoldásakor sok esetben egymásba ágyazottak.

```
x = 0
FOR  i = 1  TO  2  [BY  1]
    FOR  j = 1  TO  3  [BY  1]
        x = x + 1
    NEXT
NEXT
```

Az x értéke az egymásbaágyazott ciklusok végrehajtása után 6 lesz.
A [...] jelölés arra utal, hogy a módosító (lépésköz) megadása elmaradhat. Ekkor a feltételezett érték: +1.

2. Az algoritmusok megfogalmazására felhasználható segédeszközök

Ebben a fejezetben a leggyakrabban használt algoritmus leíró módszereket tekintjük át:

- Pszeudókód
- Folyamatábra
- Struktogram
- Szerkezeti ábra

2.1 Pszeudókód

Az algoritmus szövegszerű (mondatszerű) leírása. Hátránya, hogy nincs szabványosítva. Minden könyv, jegyzet szerzője az ízlésének legjobban megfelelő jelölésrendszert használ, ezért mindig definiálni kell az egyes alap és vezérlőtevékenységek éppen aktuálisan alkalmazott jelölésrendszerét.
A nyelve lehet angol, de adott esetben akár a szerző anyanyelvének megfelelő kulcsszavak is alkalmazhatók. E jegyzetben mi is ez utóbbit tesszük.

Operátorok:

- Aritmetikai operátorok: **+, -, *, /**
- Relációs (összehasonlító) operátorok**: <, <=, >, >=, =, <>**
- Logikai operátorok: **és, vagy, nem**

Tevékenységek (műveletek):

- Értékadás: változó = kifejezés
- Beolvasás: **Be:** <változó lista>
- Kiírás: **Ki:** <változó lista>

- Szelekciós tevékenység:
  - Egyszerű alternatíva

```
Ha <feltétel> akkor
     <műveletek>
ha vége.
```

  - Kettős alternatíva:

```
Ha <feltétel> akkor
     <műveletek1>
különben
     <műveletek2>
ha vége.
```

- Összetett alternatíva:

```
Ha <feltétel1> akkor
     <műveletek1>
különben  Ha <feltétel2> akkor
     <műveletek2>
különben  Ha <feltétel3> akkor
     <műveletek3>
[különben
     <műveletek4>]
ha vége.
```

- Iterációs tevékenység:
  - Előltesztelő - WHILE típusú ciklus:

```
Ciklus amíg <feltétel>
     <műveletek>
ciklus vége.
```

  - Előltesztelő - FOR típusú ciklus:

```
Ciklus <kezdőértéktől> <végértékig> [<lépésköz>]
     <műveletek>
ciklus vége.
```

  - Hátultesztelő - DO - WHILE típusú ciklus:

```
Ciklus
     <műveletek>
amíg <feltétel> ciklus vége.
```

Eljárás (szubrutin, függvény):

```
<az eljárás neve> (<formális paraméterlista>)
     <műveletek>
<az eljárás neve> vége.
```

Példa:

Számoljuk ki egy N elemű sorozat összegét!

Megoldás:

1. Olvassuk be az összegzendő elemek számát (N);
2. Olvassuk be az összegzendő elemeket (A[1] → A[N]);
3. Számoljuk ki az elemek összegét;
4. Irassuk ki az eredményt (S)!

```
Be: N
Be: A[1] –-> A[N]
S = 0
Ciklus I=1-től N-ig
   S = S + A[I]
ciklus vége.
Ki: S
```

2.2 Folyamatábra

A folyamatábrát szokás blokksémának is nevezni.
Két alaptípusa:

- Szervezői
- Programozói

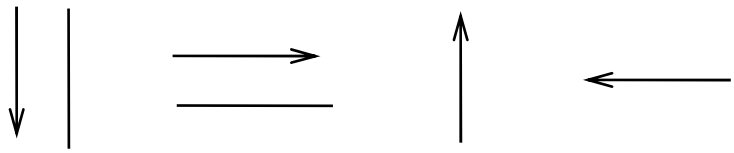
A **szervezői folyamatábra** a felhasznált adatállományok nevét, típusát, és az adatokat kezelő programok megnevezését tartalmazza.
A **programozói folyamatábra** (blokk-diagram) a megoldandó feladat részletekbe menő megfogalmazása.

A folyamatábra blokkokból áll, ezek tartalmazzák az egyes utasításokat.
Hogy egy feladatnak mekkora részét tekintjük egy blokknak, nincs előírás. Összetett, nagyobb méretű feladat esetén, mindenképpen alkalmazzunk eljárásokat.

2.2.1 Folyamatábra szimbólumok

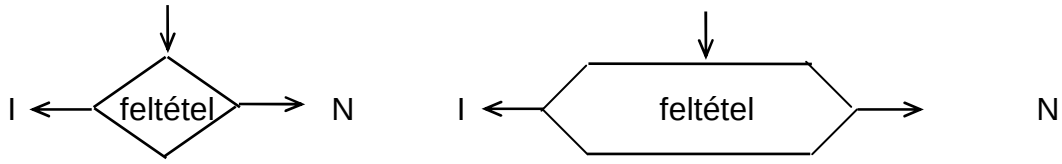
Az aritmetikai (általános) utasítás szimbóluma:
Elsősorban aritmetikai utasítások jelölésére használatos, de minden olyan esetben is használható, amelyre nincs külön szimbólum.

A művelet végrehajtási sorrend:
A sorrendet nyilakkal jelöljük. A fentről lefelé, illetve a balról jobbra mutató nyilat nem kell kitenni.
A folyamatvonalak nem keresztezhetik egymást.



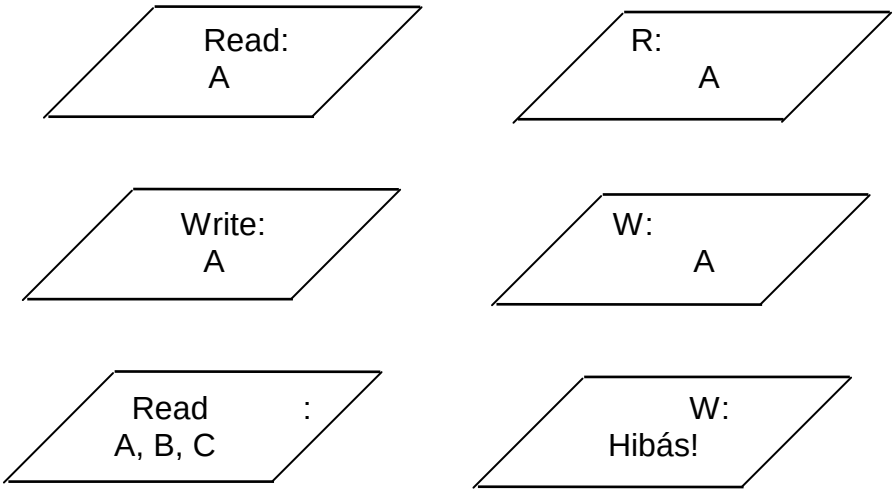
A döntés, elágaztatás (szelekció) szimbóluma:

A szimbólum belsejébe írt feltétel (kifejezés) teljesülésekor az **Igen**, ellenkező esetben a **Nem** ágon folytatódik a feladat végrehajtása.



A beolvasó (input) és kiíró (output) művelet szimbóluma:

A szimbólum nem határozza meg az adott periféria típusát, csak az adat-mozgás irányát.
A beolvasás speciális értékadásnak tekinthető. A kiíró művelet esetén konstans érték is megadható.



Az eljárás hívás szimbóluma:

Amennyiben a feladat mérete és bonyolultsága megkívánja használható.

Az algoritmus kezdés és befejezés szimbóluma:

Minden folyamatábrának egy kezdő és egy végpontja lehet. Az eljárásoknál a szimbólumok az eljárás nevét is tartalmazzák.

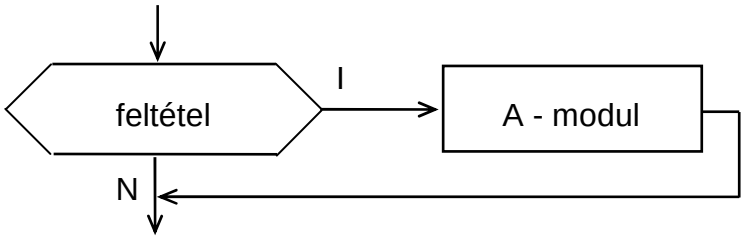
A csatlakozópont szimbóluma:

Amennyiben a folyamatábra nem fér el egy lapon,vagy a folyamatvonalak kereszteznék egymást, akkor kell alkalmazni.

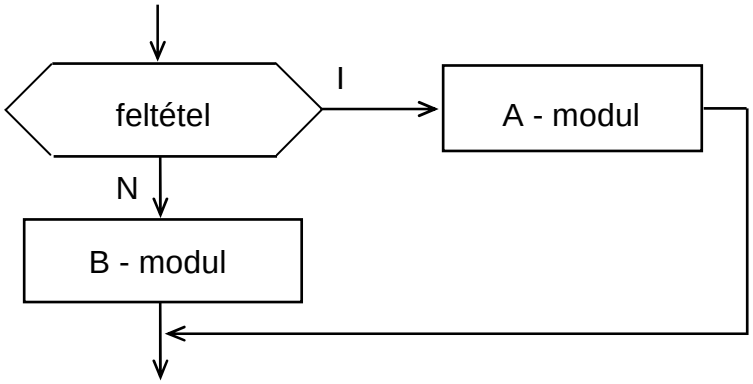
2.2.2 A vezérlőtevékenységek megvalósítása folyamatábrával

Szelekciós vezérlőtevékenység:

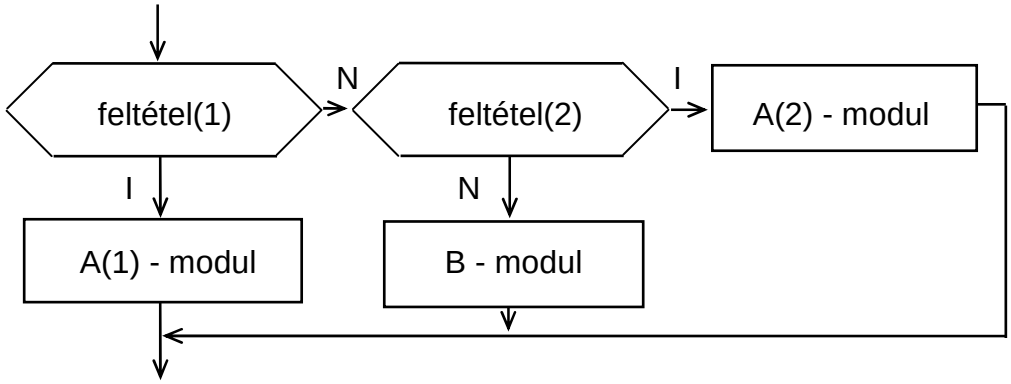
- Egyszerű alternatíva



- Kettős alternatíva

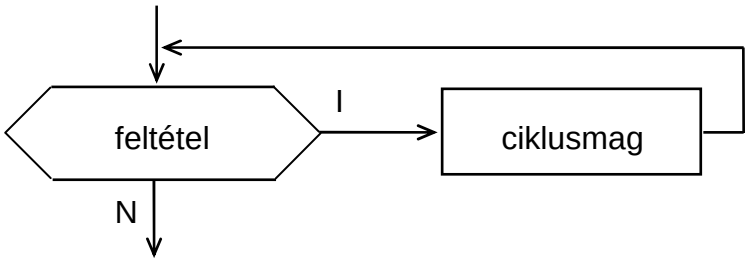


- Összetett alternatíva

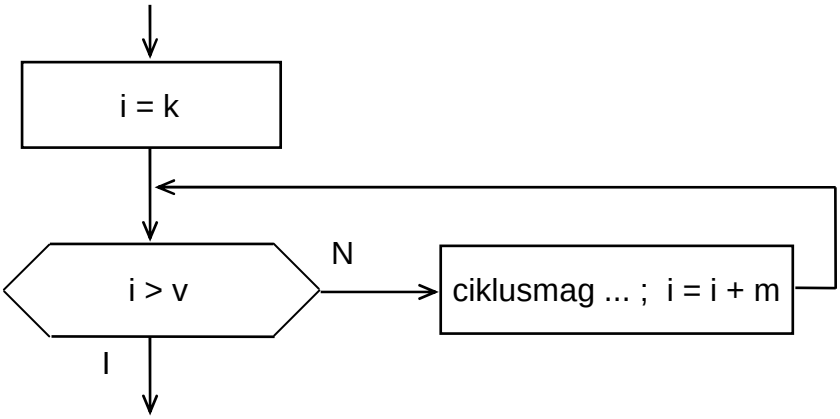


Iterációs vezérlőtevékenység:

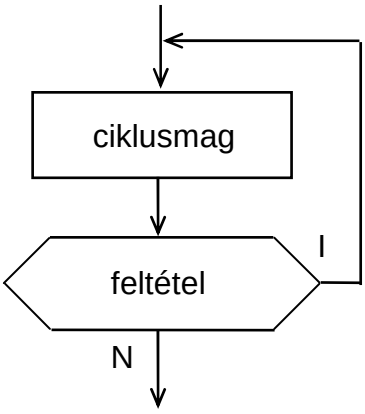
- Előltesztelő - WHILE típusú ciklus



- Előltesztelő - FOR típusú ciklus



- Hátultesztelő - DO - WHILE típusú ciklus



2.2.3 Példák folyamatábrára

1. Példa

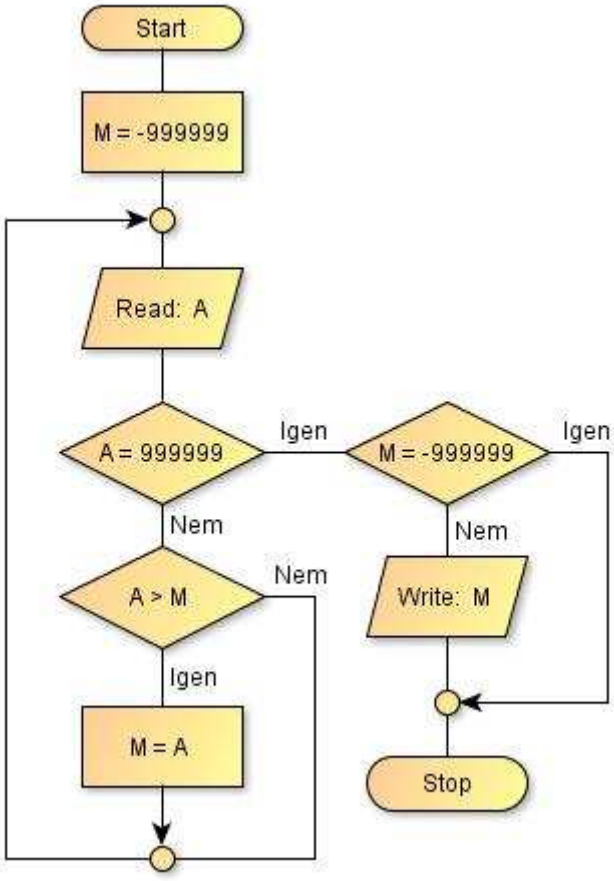
Olvasson (kérjen) be számokat és válassza ki közülük a maximális értékűt! A maximumot jelenítse meg!

Az első megoldás:

A legegyszerűbb, de csak bizonyos feltételek mellett működik helyesen. Ezek a feltételek:

- A számok maximum hat jegyűek
- A számsorozat végét a 999999 jelzi
- A számok között nem lehet a -999999 és a 999999

A változók: A – A beolvasott szám
M – A maximum

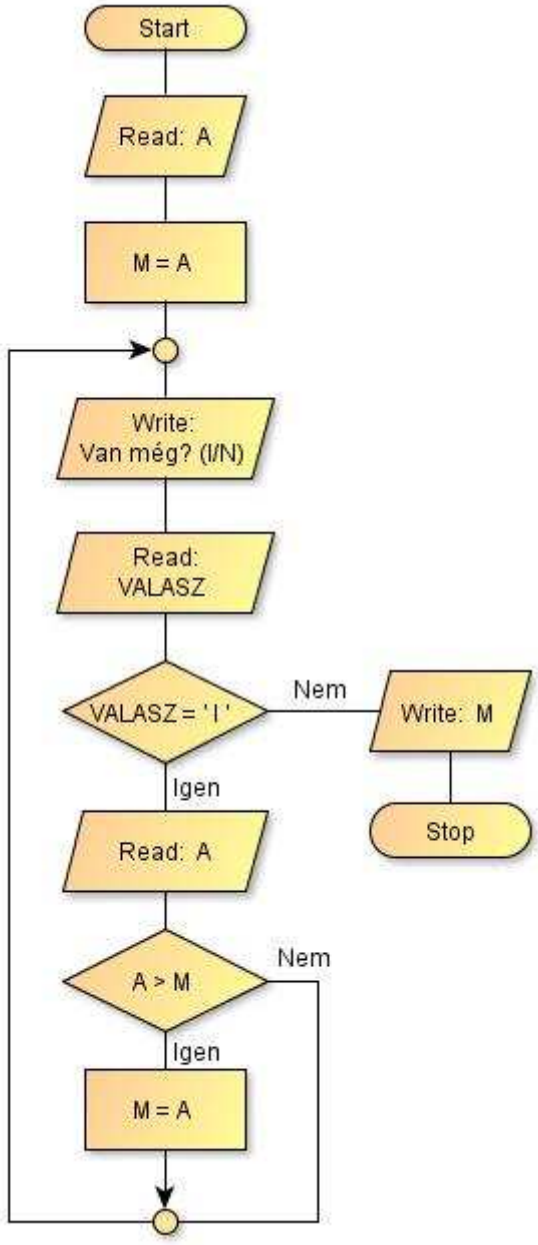


1. példa / 1. megoldás

A második megoldás:

Ebben a továbbfejlesztett megoldásban minden újabb adatbekérése előtt megkérdezzük, van-e még további szám?

A változók: A – A beolvasott szám
M – A maximum
VALASZ – A feltett kérdésre adott válasz



1. példa / 2. megoldás

2. Példa

Olvasson (kérjen) be (**N**) darab számot, és gyűjtse a párosakat a (**B**) vektorba, a páratlanokat a (**C**) vektorba! A két tömb elemeit írassa ki!

Miután először a páros, majd a páratlan számokat szeretnénk kiíratni, a vektorokba való ideiglenes tárolás elkerülhetetlen.

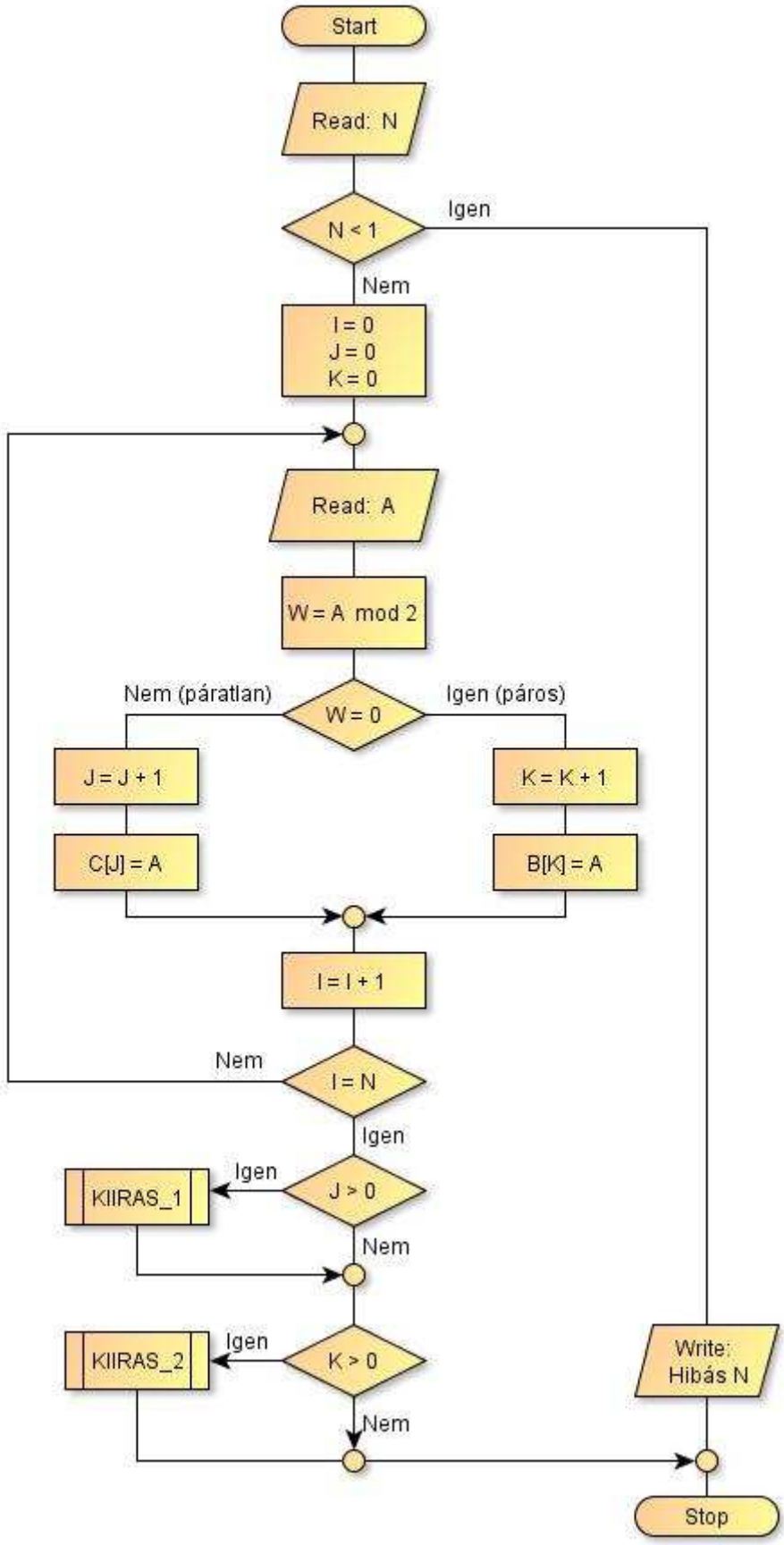
A szám páros, illetve páratlan voltának eldöntésére két módszert is választhatunk:

- Alkalmazzuk a moduló (**MOD**) utasítást, vagy függvényt.
- Egészosztást (**DIV**) végzünk kettővel, és a kapott eredményt kettővel visszaszorozva hasonlítjuk az eredeti értékhez.

Az algoritmusban a páros, illetve páratlan számok kiírására eljárásokat használunk.

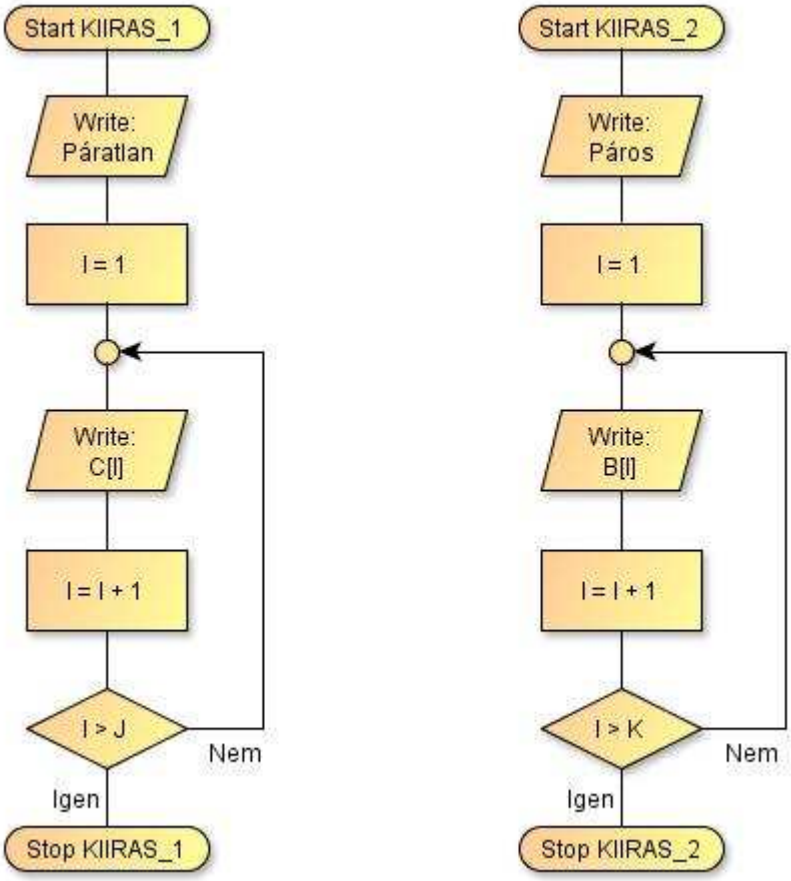
A változók: N – Az elemek száma
A – A beolvasott szám
I – A futó index
J – A páratlan számok tömbjének indexe
K – A páros számok tömbjének indexe
W – Munkaváltozó

A következő oldalon láthatják a főprogram („főalgoritmus”) folyamatábráját:


Start
Read: N
N < 1
Igen
Nem
I = 0
J = 0
K = 0
Read: A
W = A mod 2
Nem (páratlan)
W = 0
Igen (páros)
J = J + 1
K = K + 1
C[J] = A
B[K] = A
I = I + 1
Nem
I = N
Igen
Igen
KIIRAS_1
J > 0
Nem
Igen
KIIRAS_2
K > 0
Write:
Hibás N
Nem
Stop

2. példa / 1. lap

Az eljárások:



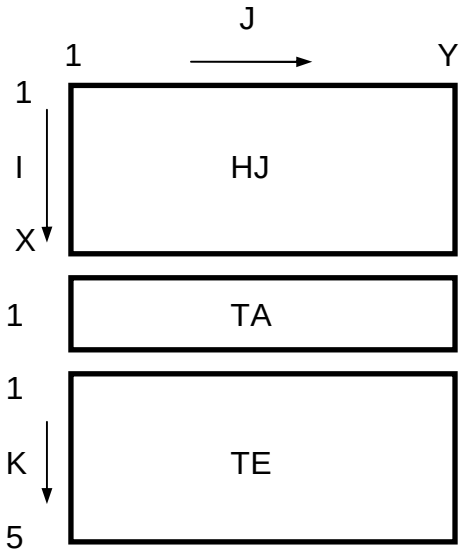
2. példa / 2. lap

3. Példa
Adott (**X**) hallgató (**Y**) tantárgyának érdemjegye.
Határozza meg a tantárgyankénti jegyeloszlást, illetve átlagot!

A feladat eléggé összetett ahhoz, hogy egy főprogram mellett több eljárásra bontsuk.

A legegyszerűbb (és egyben legáltalánosabb) megoldást akkor kapjuk, ha tömböket használunk a feladat megoldására.

A felhasznált tömböket és ciklusváltozókat az alábbi ábrán szemléltetjük:

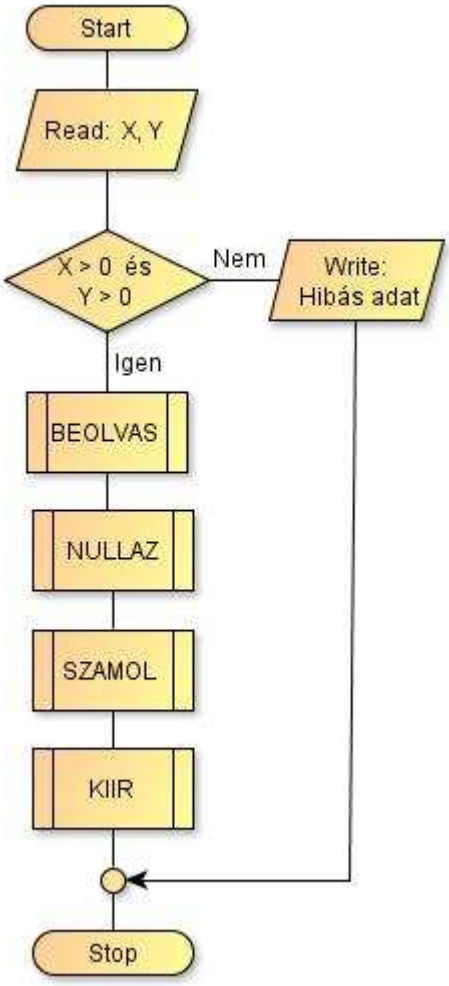


HJ - Hallgatók jegyei mátrix

TA - Tantárgy átlag vektor

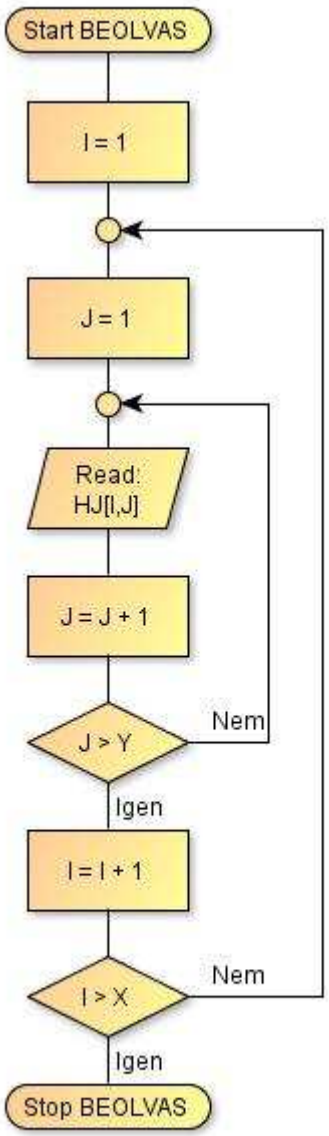
TE - Tantárgy jegyeloszlás mátrix

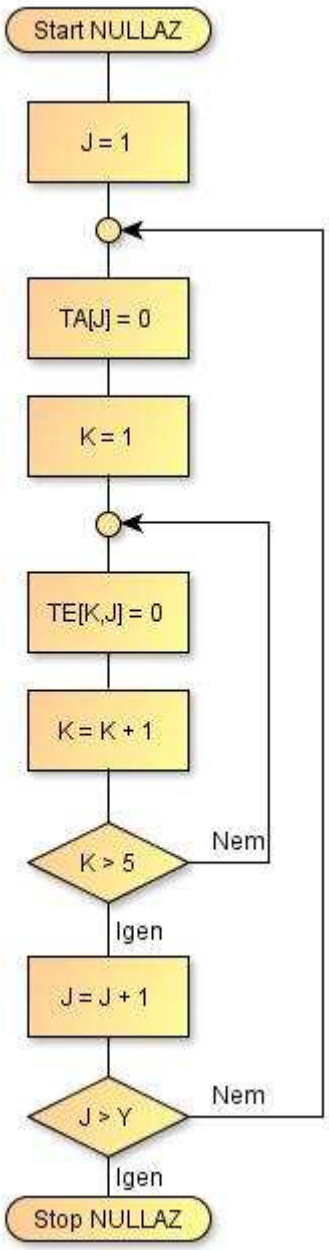
A főprogram („főalgoritmus”):



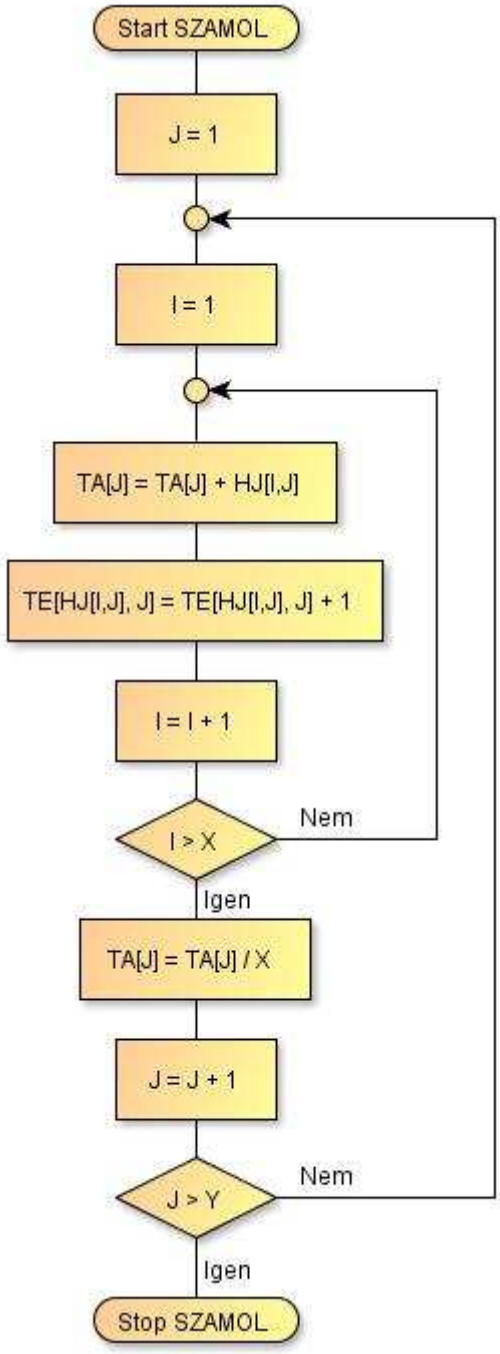
3. példa / 1. lap

Az eljárások:

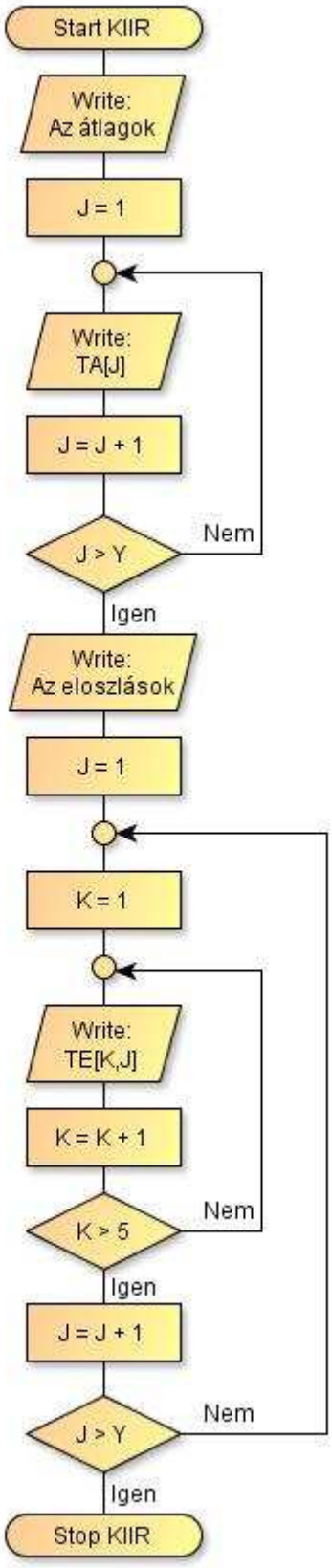




3. példa / 2. lap


Start SZAMOL
J = 1
I = 1
TA[J] = TA[J] + HJ[I,J]
TE[HJ[I,J], J] = TE[HJ[I,J], J] + 1
I = I + 1
I > X
Nem
Igen
TA[J] = TA[J] / X
J = J + 1
J > Y
Nem
Igen
Stop SZAMOL

3. példa / 3. lap


Start KIIR
Write:
Az átlagok
J = 1
Write:
TA[J]
J = J + 1
J > Y
Nem
Igen
Write:
Az eloszlások
J = 1
K = 1
Write:
TE[K,J]
K = K + 1
K > 5
Nem
Igen
J = J + 1
J > Y
Nem
Igen
Stop KIIR

3. példa / 4. lap

2.3 Struktogram

A jel algoritmus másik gyakran használt formája. Jellemzője, hogy az algoritmus megtervezésére és megfogalmazására csak téglalapokat használ. Egy feladat megoldását az őt jelképező téglalapba rajzolt - szintén téglalappal jelképezett - részfeladatok megoldásával, vagy alaptevékenységek végrehajtásával kapjuk meg.

2.3.1 A vezérlőtevékenységek megvalósítása struktogrammal

Szelekciós vezérlőtevékenység:

- Egyszerű alternatíva

// feltétel //	
igen	nem
A - modul	-

- Kettős alternatíva

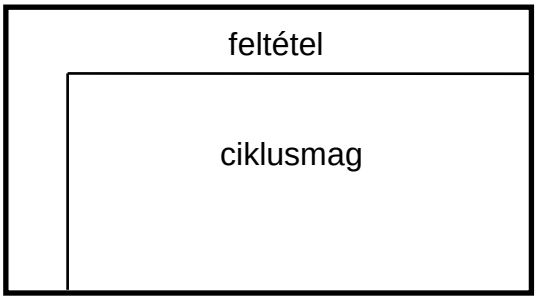
// feltétel //	
igen	nem
A - modul	B - modul

- Összetett alternatíva

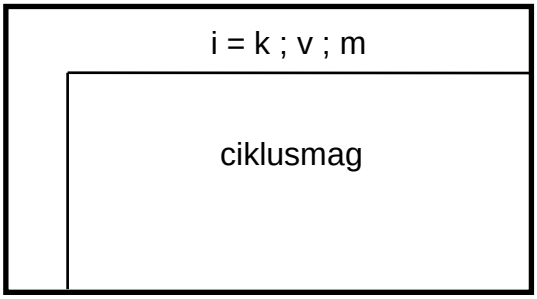
<table>
<tr><td colspan="3">// feltétel(1) //</td></tr>
<tr><td>igen</td><td colspan="2">nem</td></tr>
<tr><td rowspan="3">A(1) - modul</td><td colspan="2">// feltétel(2) //</td></tr>
<tr><td>igen</td><td>nem</td></tr>
<tr><td>A(2) - modul</td><td>A(3) - modul</td></tr>
</table>

Iterációs vezérlőtevékenység:

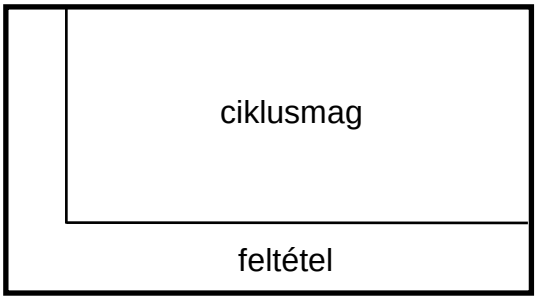
- Előltesztelő - WHILE típusú ciklus



- Előltesztelő - FOR típusú ciklus



- Hátultesztelő - DO - WHILE típusú ciklus



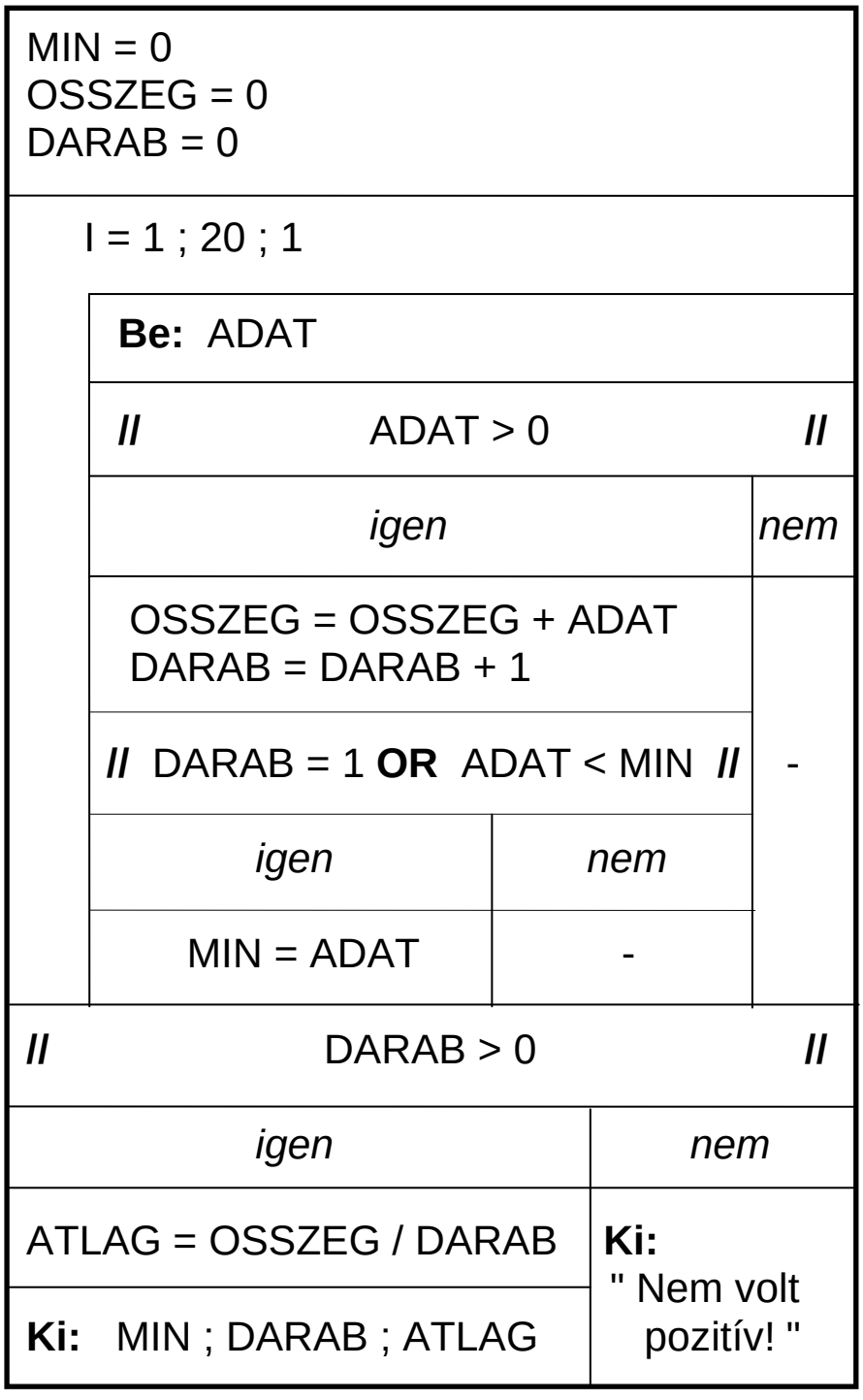
2.3.2 Példa struktogramra

Olvassunk (kérjünk) be 20 számot! Keressük ki a számsorozat pozitív elemeinek minimumát, számítsuk ki ezek összegét és átlagát!

A feladatban felhasznált változók:

A változók:	MIN	– A pozitív számok minimum értéke
	OSSZEG	– A pozitív számok összege
	DARAB	– A pozitív számok darabszáma
	ATLAG	– A pozitív számok átlaga
	I	– Ciklusváltozó

A feladat struktogramja:

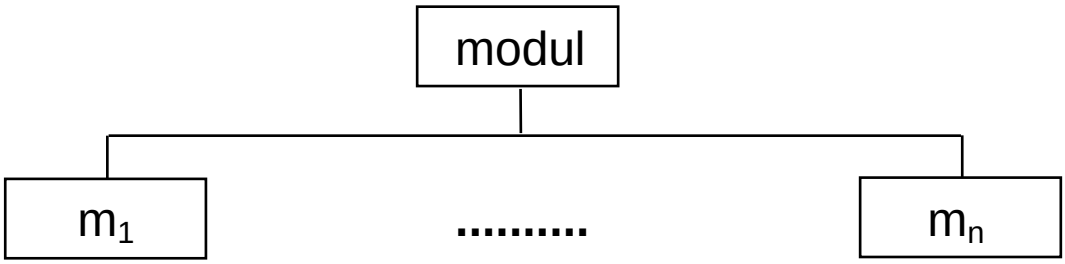


2.4 A szerkezeti ábra (Jackson-diagram)

A szerkezeti ábra modulokból épül fel. A modulok jelentik az elvégzendő feladatokat. Jelölésére téglalapokat használunk, amelybe a modul nevét, vagy az elvégzendő feladatot írjuk.

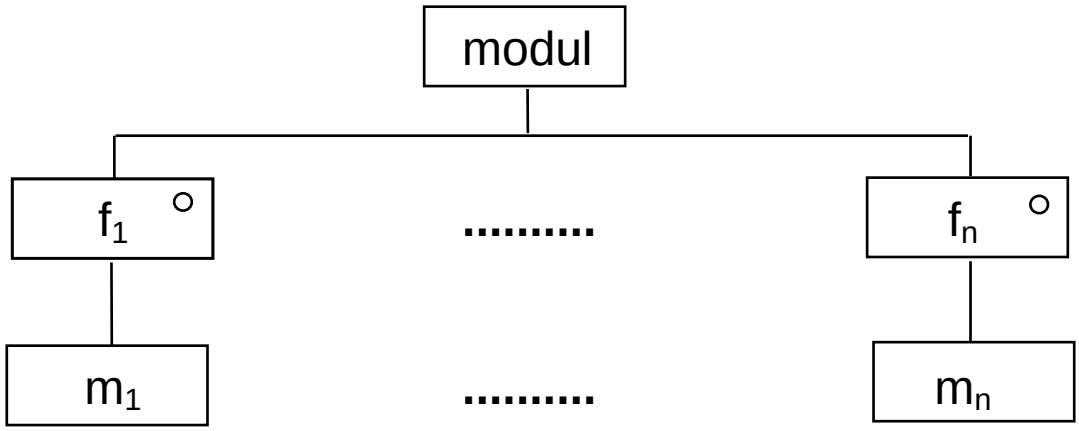
2.4.1 A vezérlőtevékenységek megvalósítása szerkezeti ábrával

Szekvenciális vezérlőtevékenység:



Jelentése:
A **modul** nevű feladat úgy oldandó meg, hogy egymás után megoldjuk az **m_1**..... **m_n** nevű feladatokat, vagy ha valamelyik **m_i** modul alaptevékenység, azt végrehajtjuk.

Szelekciós vezérlőtevékenység:

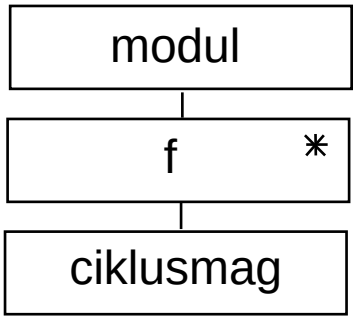


Jelentése:

A **modul** nevű feladatot úgy kell megoldani, hogy a feltételeket megvizsgálva, megoldjuk, vagy végrehajtjuk azt a modult, melynél az $\mathbf{f_i}$ feltétel igaz.
Kikötés, hogy a feltételek közül csak egy, vagy egyik sem lehet igaz.

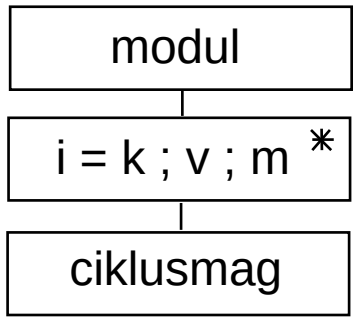
Iterációs vezérlőtevékenység:

- Előltesztelő - WHILE típusú ciklus



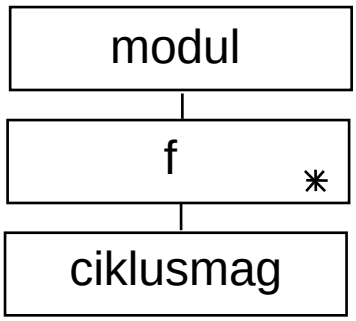
Jelentése:
A **modul** nevű feladatot úgy oldjuk meg, hogy a **ciklusmag**ot addig ismételjük, míg az **f** feltétel igaz.

- Előltesztelő - FOR típusú ciklus



Jelentése:
A **modul** nevű feladat megoldása **i = k** értékkel kezdődik **v** értékig tart, és **m** értékkel módosul minden ciklusmag végrehajtás után.

- Hátultesztelő - DO - WHILE típusú ciklus



Jelentése:
A **modul** nevű feladatot úgy oldjuk meg, hogy a **ciklusmag**ot addig ismételjük, míg az **f** feltétel igaz.
A feltétel vizsgálat a ciklusmag végrehajtása után történik.

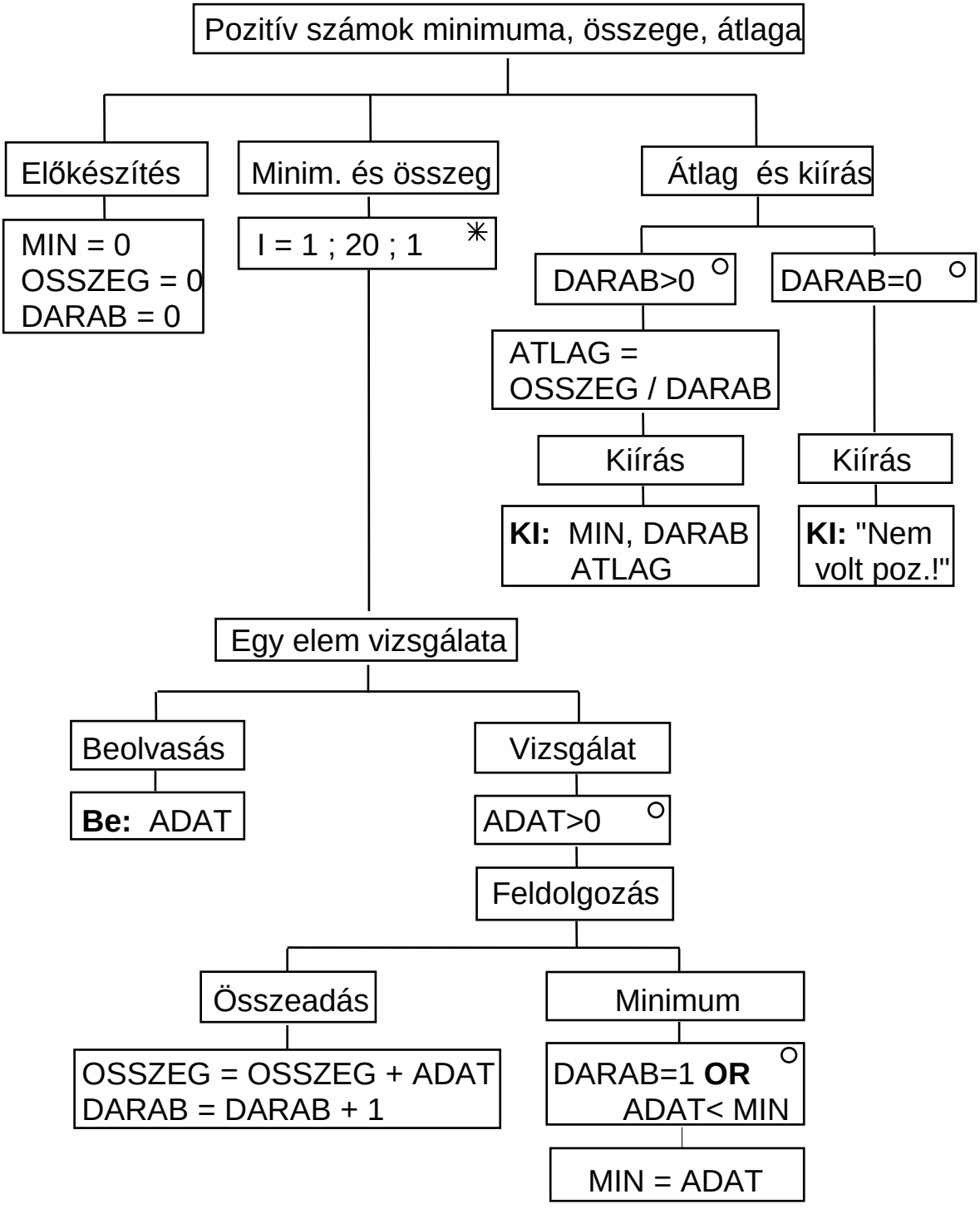
2.4.2 Példa szerkezeti ábrára

Olvassunk (kérjünk) be 20 számot! Keressük ki a számsorozat pozitív elemeinek minimumát, számítsuk ki ezek összegét és átlagát!

A feladatban felhasznált változók:

MIN	- A pozitív számok minimum értéke
OSSZEG	- A pozitív számok összege
DARAB	- A pozitív számok darabszáma
ATLAG	- A pozitív számok átlaga
I	- Ciklusváltozó

A feladat szerkezeti ábrája:



3. Adatábrázolás

A számítógépi programok utasításai döntően valamilyen adatra vonatkozó műveletet hajtanak végre. Az adatnak a műveletvégzés idején a memóriában kell lennie, azaz valamilyen **"szabványos" formátumban a memóriában kell tárolni.**

Az adatok **felhasználási területük** szerint a következő csoportokba sorolhatók:

- Numerikus
- Logikai
- Karakteres

Az informatikában a megszokott tizes számrendszer mellett még további három számrendszerrel találkozhatnak:

Kettes (bináris)
Számjegyek: 0 , 1
Jelölés: 10101110_2

Tizenhatos (hexadecimális)
Számjegyek: 0 - 9 , A , B , C , D , E , F
Jelölés: $1AF3_{16}$

Nyolcas (oktális)
Számjegyek: 0 , 1 , 2 , 3 , 4 , 5 , 6 , 7
Jelölés: 1732_8

3.1 Alapműveletek különböző számrendszerekben

Átalakítás decimálisból binárisba:

51.75_{10} ⟶ 110011.11_2

51 / 2	
25	1
12	1
6	0
3	0
1	1
0	1

	75 * 2
1	50
1	00

Átalakítás binárisból hexadecimálisba:

$1\ 0110\ 0010.001_2 \longrightarrow 162.2_{16}$

Összeadás (Bin):

1011101101	749
101110	46
+ 11011	+ 27
1100110110_2	822_{10}

Kivonás (Bin):

1011101101	749
- 11011111	- 223
1000001110_2	526_{10}

Kivonás komplemens szám segítségével (Bin):

1011101101 - 11011111 = ?

0011011111 Egyes komplemense: 1100100000

Kettes komplemense: + 1

1100100001

1011101101

+ 1100100001

~~1~~1000001110$_2$

Átalakítás decimálisból hexadecimálisba:

$51.75_{10} \longrightarrow 33.C_{16}$

51 / 16			75 * 16
3	3	↓	12 | 00
0	3		

Átalakítás hexadecimálisból binárisba:

$33.C_{16} \longrightarrow 11\ 0011.11_2$

Összeadás (Hex):

1AF	431
+ 3A	+ 58
$1E9_{16}$	489_{10}

Kivonás (Hex):

2BA	698
- 1E2	- 482
$D8_{16}$	216_{10}

Kivonás komplemens szám segítségével (Hex):

2BA - 1E2 = ?

1E2 Komplemense: E1D

\+ 1

E1E

2BA

+ E1E

~~1~~0D8$_{16}$

3.2 Adattípusok

A konkrét adattípusok ismertetése előtt érdemes még néhány fogalommal megismerkedni.

- Bájt (Byte)

A számítógépek legkisebb címezhető egysége.
A memóriában nyolc egymás melletti bit.

- Félszó (Half Word)

Két egymás melletti bájt.

- Szó (Word)

Négy egymás melletti bájt.

- Duplaszó (Double Word)

Nyolc egymás melletti bájt.

- Zónarész, Magas (High) bitek

Egy bájt balról számolt négy bitje. (Tetrád)

- Számrész, Alacsony (Low) bitek

Egy bájt jobbról számolt négy bitje.

Egy bájt értéke tehát a következő formában adható meg:

1110 1010_2 = EA_{16}

Zónarész Számrész
High Low

Karakterek ábrázolása

- Az **ASCII** kódrendszert használja
  (**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange)
- 7 bites, 8 bites változat
- Nemzeti karakterkészletek
  (437, 852, 1250 kódlapok)
- Magyar szabvány: MSZ 7794-3
- Nemzetközi szabvány: ISO-8859-x
  x = 2 Kelet Európa
- Több bájtos változat: (UNICODE, UTF-8)

Egész típusú adatok ábrázolása

- Fixpontos bináris típus
- A kettedespont a szám után képzelendő (Csak egész érték)
- A negatív számok kettes komplemens formában
- A bináris aritmetika számol vele

- A hossz alapján:
  - Szavas egész (2 bájt)
    A számábrázolási tartomány:
    -2^{15} - $+2^{15}-1$
    - 32768 - + 32767

A szám (Decimális)	**Memória tartalom** (Hexadecimális)
1000	03E8
3	0003
-1	FFFF
32767	7FFF
-32768	8000
0	0000

  - Dupla egész (4 bájt)
    A számábrázolási tartomány:
    -2^{31} - $+2^{31}-1$
    Több mint ±2 milliárd

A szám (Decimális)	**Memória tartalom** (Hexadecimális)
-1	FFFFFFFF
1	00000001

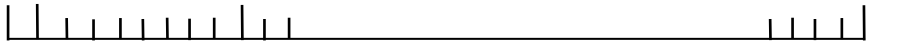
- Hosszú egész (8 bájt)
  A számábrázolási tartomány:
  -2^{63} - $+2^{63}-1$

A szám (Decimális)	Memória tartalom (Hexadecimális)
-1	FFFFFFFF FFFFFFFF
5	00000000 00000005

Valós típusú adatok ábrázolása

- Minden szám felírható a következő formában:
  $\pm \mathbf{m} * \mathbf{r}^{\pm \mathbf{e}}$
- Ahol **m** a mantissza, **r** a számrendszer alapszáma (radixa), **e** a kitevő (exponens)
- Bináris számrendszerben tehát:
  $\pm \mathbf{m} * \mathbf{2}^{\pm \mathbf{e}}$
- A törtszámok is ábrázolhatók
- Az exponens 8, 11, 15 bit, típustól függően
- A tárolás normalizált formában: $\pm \mathbf{1.m} * \mathbf{2}^{\pm \mathbf{e}}$
- IEEE 754 szabvány
- Tudományos, gazdasági számításokhoz
- A lebegőpontos aritmetika használja

- A hossz alapján:
  - Egyszeres pontosságú lebegőpontos szám
    (Single precision, 4 bájt)
    - A számábrázolási tartomány:
      $\pm 10^{-38}$ – $\pm 10^{+38}$
    - A pontossága: 6 - 7 decimális jegy
  - Felépítése:



Me Exponens (e_k) Mantissza (m)

Me - A mantissza előjele: + esetén 0
- esetén 1

Az átalakítás lépései példán bemutatva:

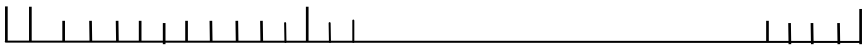
a. A decimális szám átalakítása binárissá
   $26.75_{10} \longrightarrow 11010.11_2$
b. Normalizálás
   $11010.11 \longrightarrow 1.101011 * 2^4$

c. Az exponens korrigálása (nulla pont eltolás)
$e_k = e + 127 = 4 + 127 = 131_{10} = 83_{16}$

d. Ábrázolás
0 10000011 10101100000000000000000
4 1 D 6 0 0 0 0

- Dupla pontosságú lebegőpontos szám
(Double precision, 8 bájt)
  - A számábrázolási tartomány:
$\pm 10^{-308}$ – $\pm 10^{+308}$
  - A pontossága:
15 - 16 decimális jegy
  - A korrigált exponens:
$e_k = e + 1023$

- Felépítése:



63 - 52 51 - 0 bit
Me Exponens (e_k) Mantissza (m)

Me - A mantissza előjele: + esetén 0
- esetén 1

- Kiterjesztett pontosságú lebegőpontos szám
(Extended precision, 10 bájt)

A teljes hossz 80 bit, amiből 1bit mantissza előjel, 15 bit az exponens, és 64 bit a mantissza.
  - A számábrázolási tartomány:
$\pm 10^{-4932}$ – $\pm 10^{+4932}$
  - A pontossága:
18 - 19 decimális jegy
  - A korrigált exponens:
$e_k = e + 16385$

A felhasznált irodalom

1. N. Wirth: Algoritmusok + adatstruktúrák = programok
könyv
Műszaki Könyvkiadó, Budapest
2. Lipschutz: Adatszerkezetek
könyv
Panem Kft, Budapest
3. Dr. Marton László - Pukler Antal - Pusztai Pál: Bevezetés a programozásba
könyv
NOVADAT BT, Győr