Algoritmusleíró eszközök

Az algoritmusleíró eszközök használatának célja a feladatok megoldásának leírása programozási nyelvtől független nyelven. A programozási nyelvek ugyanis szigorú szintaxisúak, a tervezés szempontjából lényegtelen sallangokat tartalmaznak. A programozási nyelven történő tervezés esetén nehézzé válhat a program átírása más nyelvre, más gépre.

Mindegyik algoritmusleíró eszközzel leírjuk egy feladat megoldását:

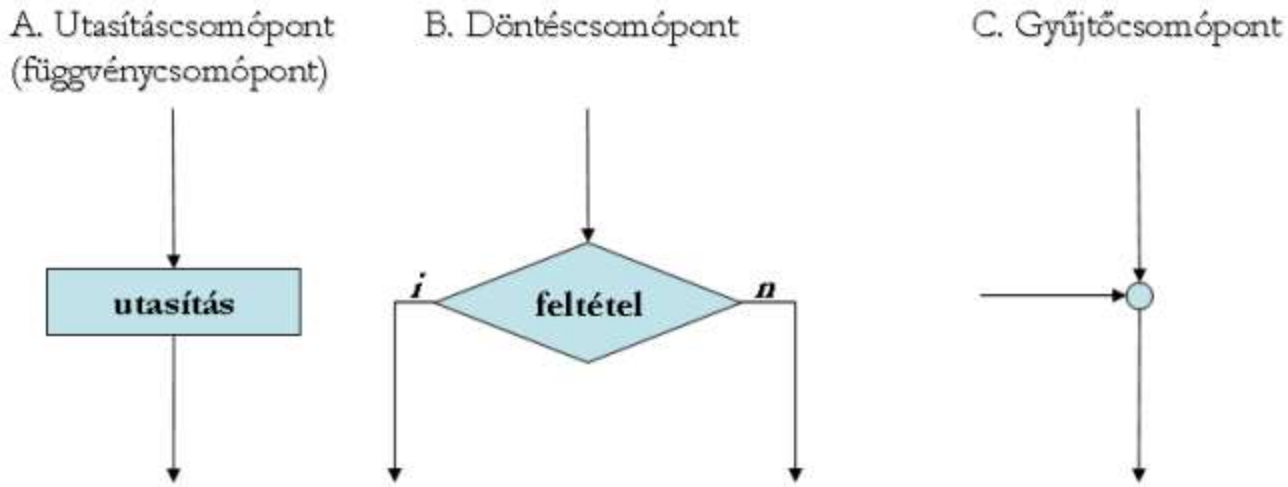
Feladat

N tanuló év végi átlagának ismeretében adjuk meg a jeles átlagú tanulók számát!

1. Folyamatábra

Az egyik legkorábban kialakult algoritmusleíró eszköz a programot *gráfként* írja le. A programgráf egy irányított gráf, amely csomópontokból és az azokat összekötő élekből áll, egyetlen induló és befejező éle van, az induló élből bármely csomópont elérhető, s bármely csomópontból el lehet jutni a befejező élre.

A folyamatábra alapesetben háromféle csomópontot tartalmaz.



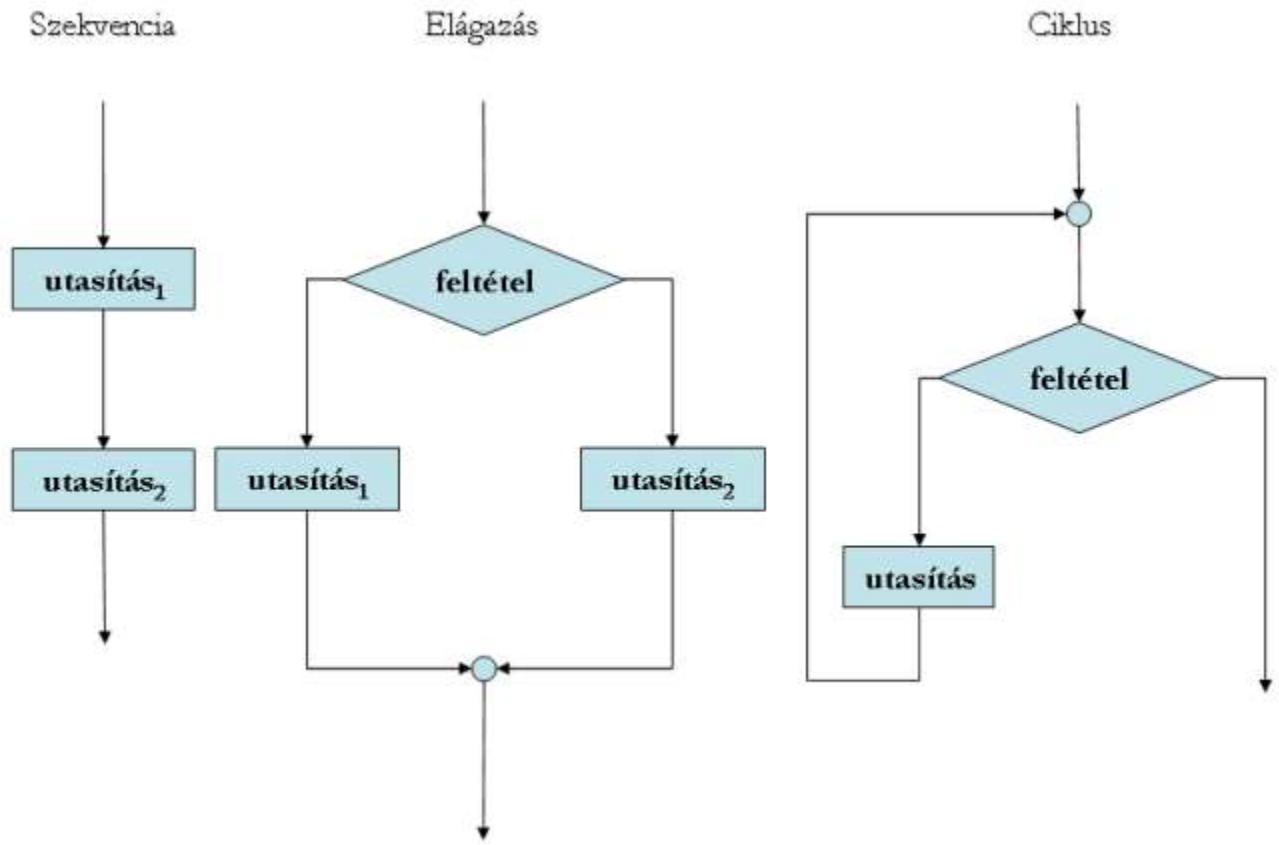
Folyamatábra-szimbólumok.

Az utasítás-csomóponton áthaladva végre kell hajtani a beleírt utasítást. A döntéscsomópontban levő feltétel igaz értéke esetén az **i** betűvel jelölt élen, hamis értéke esetén pedig az **n** betűvel jelölt élen kell továbbhaladni. A gyűjtőcsomóponton való áthaladás nem változtatja meg a programállapotát.

A programe háromfajta csomópontból felépített gráf, bemenő éle a „semmiből” jön, kimenő éle a „semmibe” megy. A szokásos algoritmikusstruktúrák ezekből a közvetkezőképpen építhetők fel (csak a struktúrát jelöljük, a tartalmat nem).

Az ilyen algoritmusleíró eszköz jól használható az algoritmusok végrehajtásának követésére, hiszen a végrehajtás a programgráf csomópontjainak bejárása az élek mentén.

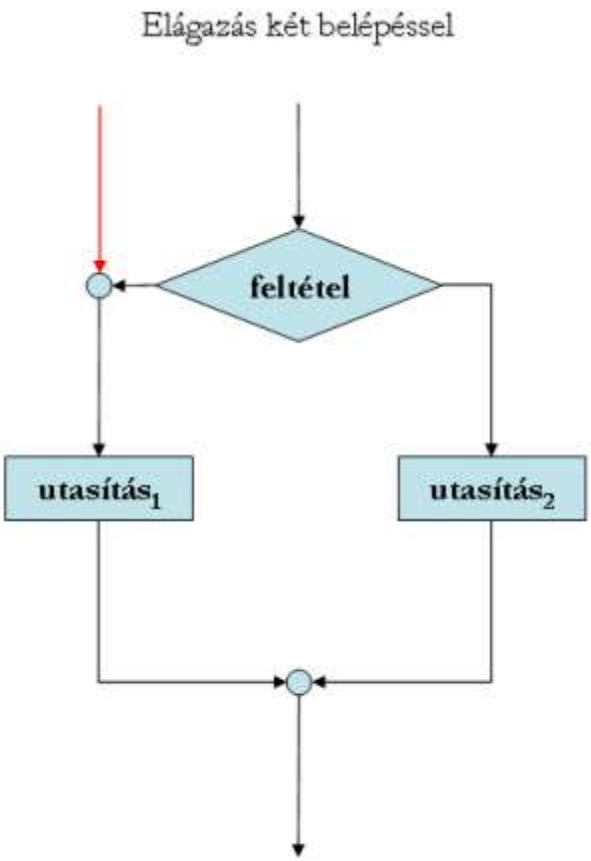
Előnye mellett azonban több nagyon hátrányos jellemzője van. Terjedelmes, szövegszerkesztővel nehezen készíthető, javítása rendkívül nehézkes. Nagy programok leírása könnyen áttekinthetetlen ábrákhoz vezethet (nem fér ki egy lapra, a gráf élei többszörösen keresztezik egymást stb.). Alapproblémája, hogy strukturális alapelemei nem azonosak a szokásos algoritmikus szerkezetekkel, sőt segítségükkel más struktúrák is létrehozhatók.

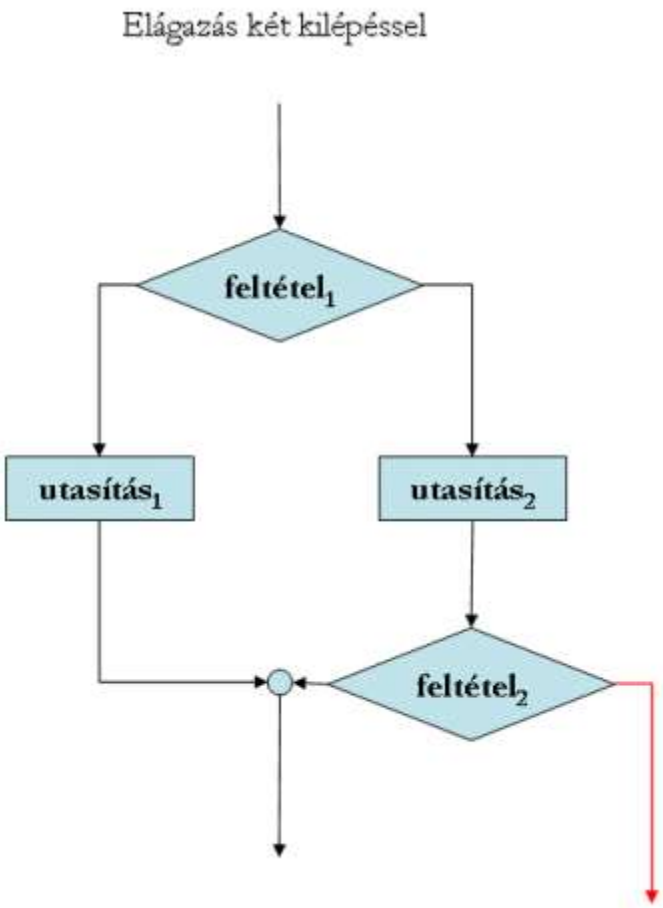


Elemi strukturált programok folyamatábra-szimbólumokkal.

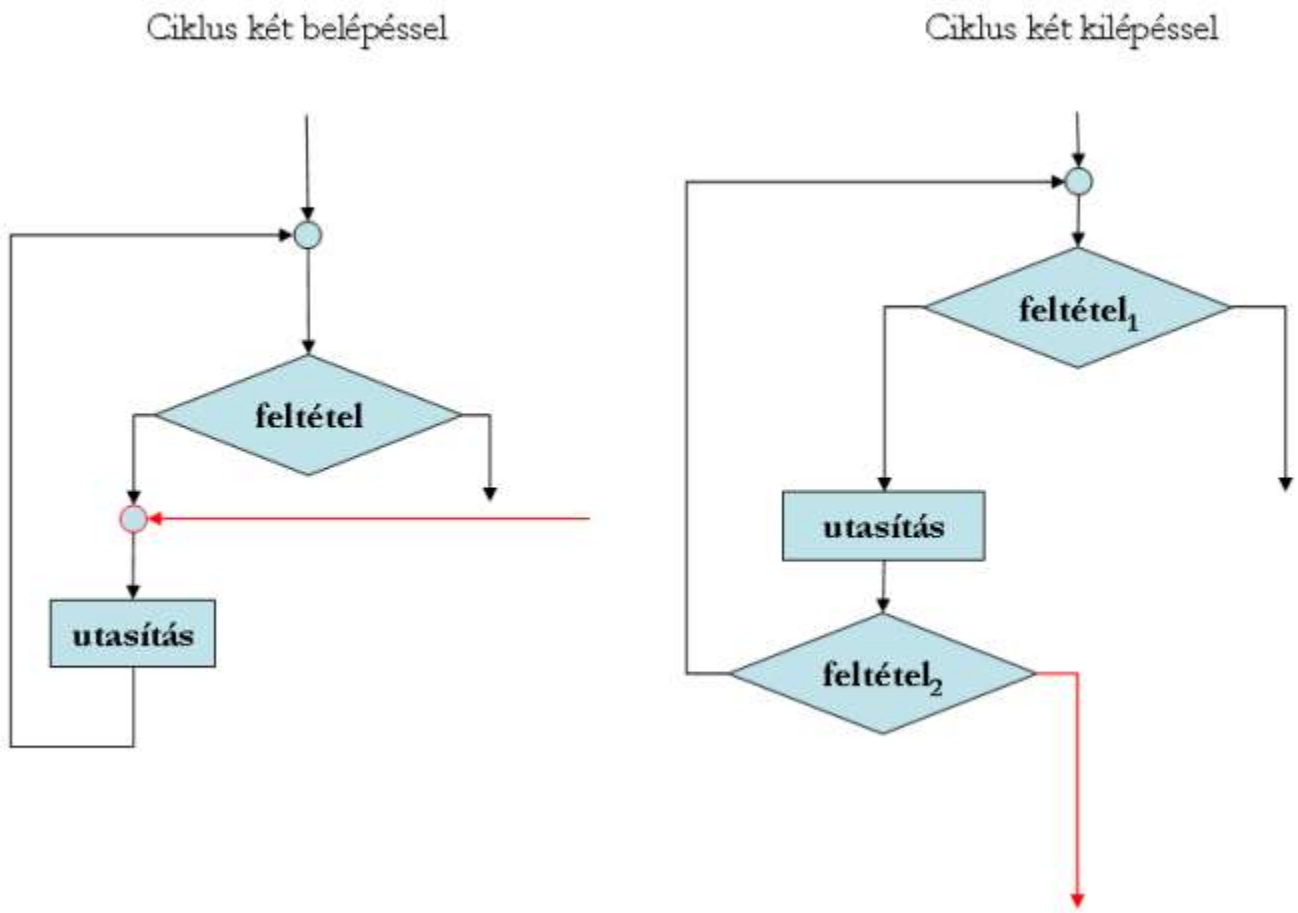
Definíció: **Strukturált programnak** nevezzük azt a programot, amely csak a fenti három algoritmikus szerkezetet (szekvencia, elágazás, ciklus) tartalmazza.

Vannak olyan programok, amelyek nemcsak ezeket az ún. strukturált alapszerkezeteket tartalmazzák, hanem másokat. E mások a következőfélék lehetnek*:*





Nem strukturált alapszerkezetek

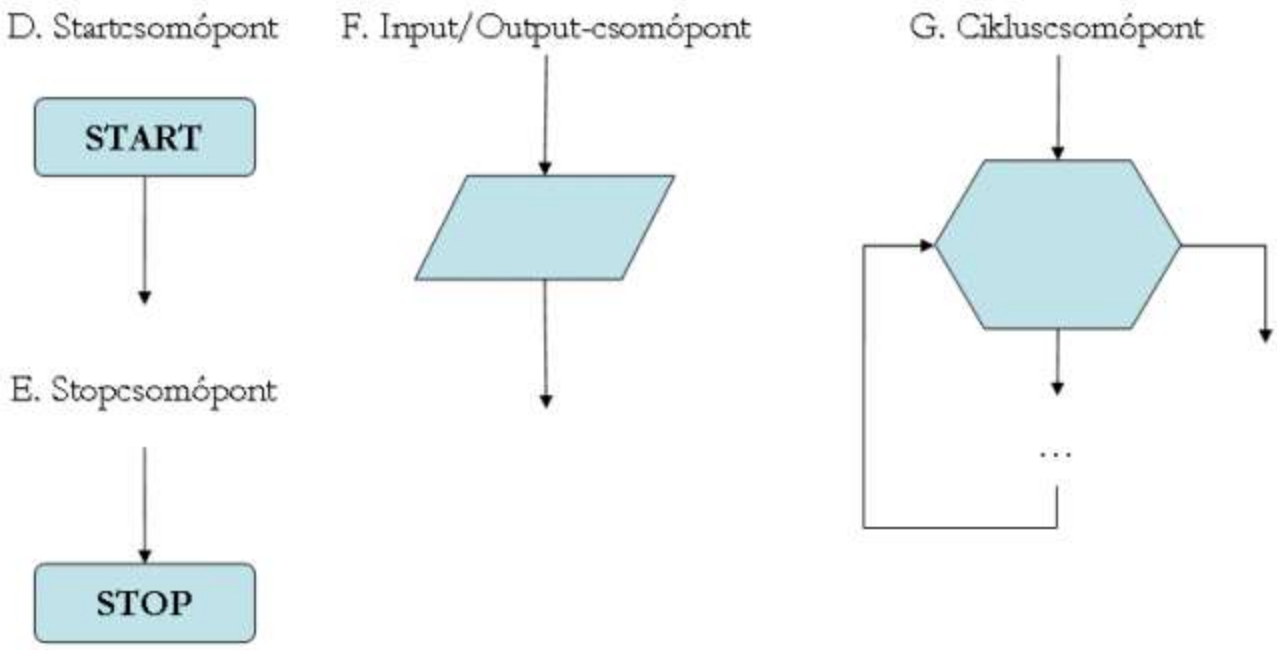


Nem strukturált alapszerkezetek

Mind a négy nem strukturált alapszerkezetről látható, hogy bonyolultabbak a strukturált alapszerkezeteknél, alkalmazásuk nehezebb, bonyolultabb programot eredményez.

Szigorú szabályok önkéntes betartásával persze ezt az eszközt is lehet használni strukturált programok írására, de a sok más lehetőség túlságosan csábító lehet, s az ettől való eltérésért sokszor nagyon nagy árat kell fizetni.

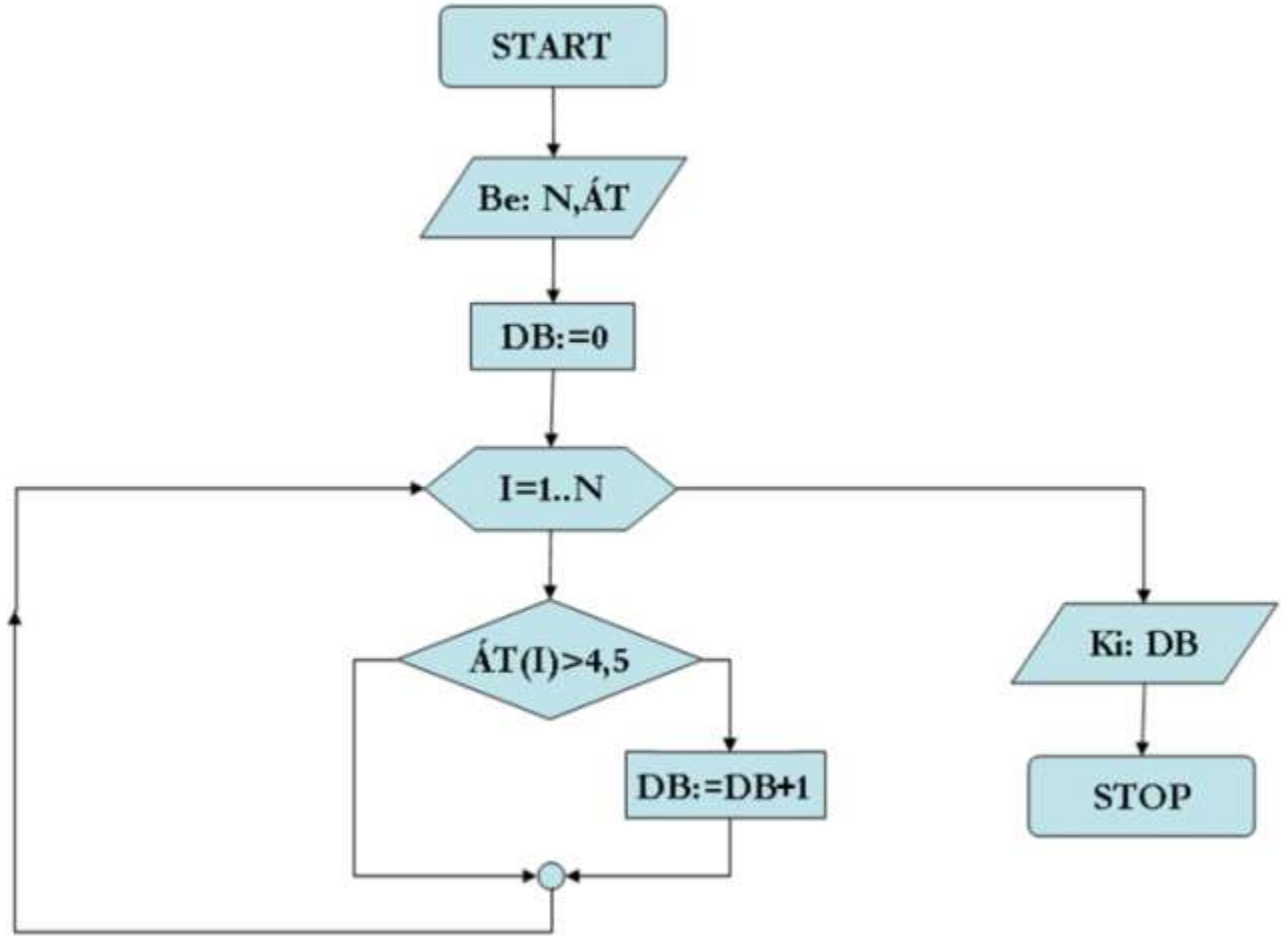
E minimális csomópontkészletet a folyamatábráknál néhány újabb elemmel szokás bővíteni – a leírást nem teszik lényegileg bővebbé, csupán kényelmességi, olvashatósági szempontok miatt definiálták ezeket. Közülük néhány, a korábban elkezdett betűjelölést folytatva az alábbi ábrán látható.



További folyamatábra-szimbólumok.

Start és Stop csomópont a programban egyetlen lehet, az eddigi „semmiből” jövő, illetve „semmibe” menő élhez helyezzük el őket. Az input-output csomópontba beolvasást, illetve kiírást tehetünk, jelölve, hogy éppen melyikről van is szó. A ciklusCsomópontot számlálós ciklusoknál alkalmazzuk, a csomópontban szerepelnie kell a ciklusváltozónak, a ciklusváltozó kezdeti és végértékének.

A kitűzött példa ezen eszközökkel készült megoldását láthatjuk a következő ábrán.



Példaprogram folyamatábrával.

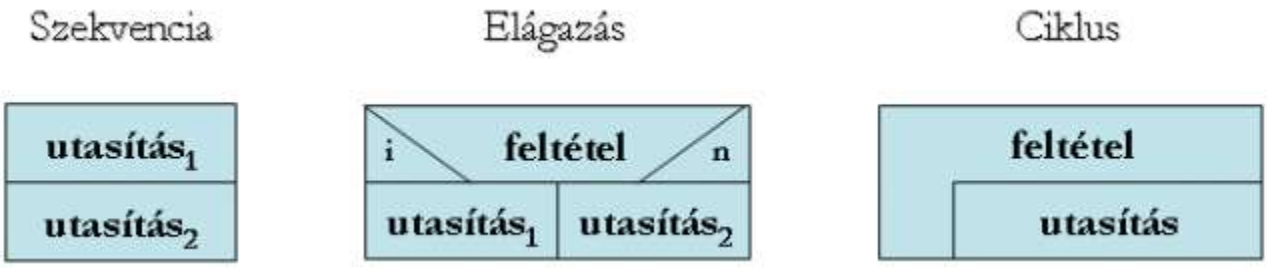
Vissza a tartalomjegyzékhez

2. Struktogram

A stuktogram az előző eszköz hibáit próbálja kiküszöbölni azzal, hogy a programgráfot élek nélkül ábrázolja. Így egyetlen egy alapelem marad, a téglalap.

A struktogram alapeleme.

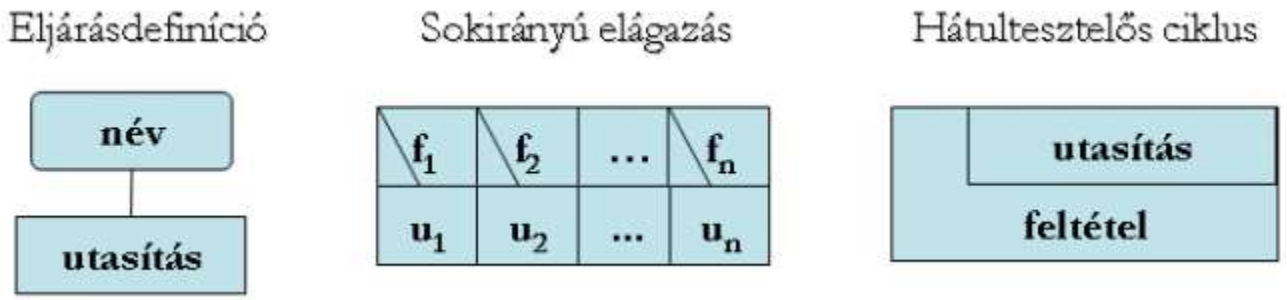
Ezzel az alapelemmel építhetjük fel a szokásos strukturált alapszerkezeteket (és csak azokat).



A struktogram összetett alapszerkezetei.

Szekvenciánál a téglalapok egymás alatti sorrendje dönti el a végrehajtás sorrendjét. Az elágazásfeltétel igaz értéke esetén az **i** betűvel jelölt bal oldali téglalap utasítását kell végrehajtani, hamis értéke esetén pedig az **n** betűvel jelölt jobb oldali téglalapét. Ha az elágazás valamelyik ága üres, akkor a neki megfelelő téglalap is üres marad. A ciklus elöltesztelős, azaz a benne levő utasítást mindaddig végre kell hajtani, amíg a feltétel igaz.

Az utasítások helyén lehet egyetlen elemi utasítás, lehet a három algoritmikus szerkezet valamelyike, és lehet egy eljáráshívás. Ezt a leíróeszközt még többféle elemmel szokták bővíteni: az eljárásdefinícióval, a sokirányú elágazással, illetve a hátultesztelős ciklussal.



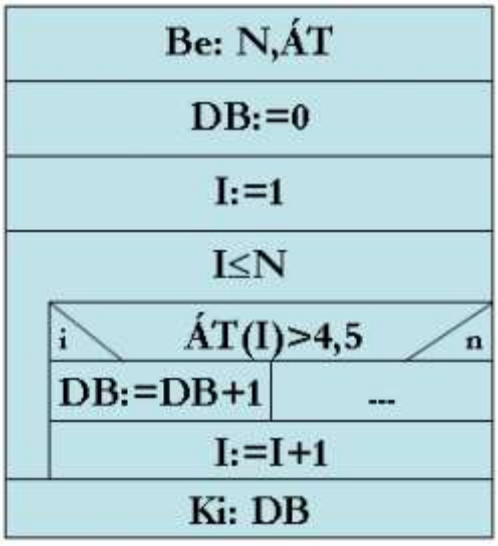
A struktogram további összetett alapszerkezetei.

Sokirányú elágazásnál azt az ágat kell végrehajtani, amelynek igaz értékű a feltétele (közülük minden esetben pontosan egy teljesülhet).

Ez az eszköz egyértelműen csak strukturált programok írására alkalmas, így kiküszöböli az előző hiányosságainak egy részét. Megmarad azonban a „rajzosság” miatt a terjedelmesség, nehéz javíthatóság. Veszítünk viszont a gráf éleinek elhagyása miatt a követhetőségből.

A lokális adatokat az eljárások téglalapjai mellett, az eljárásnév után sorolhatjuk fel.

Nézzük meg ezzel az eszközzel leírva az előző példát!



Példaprogram struktogrammal.

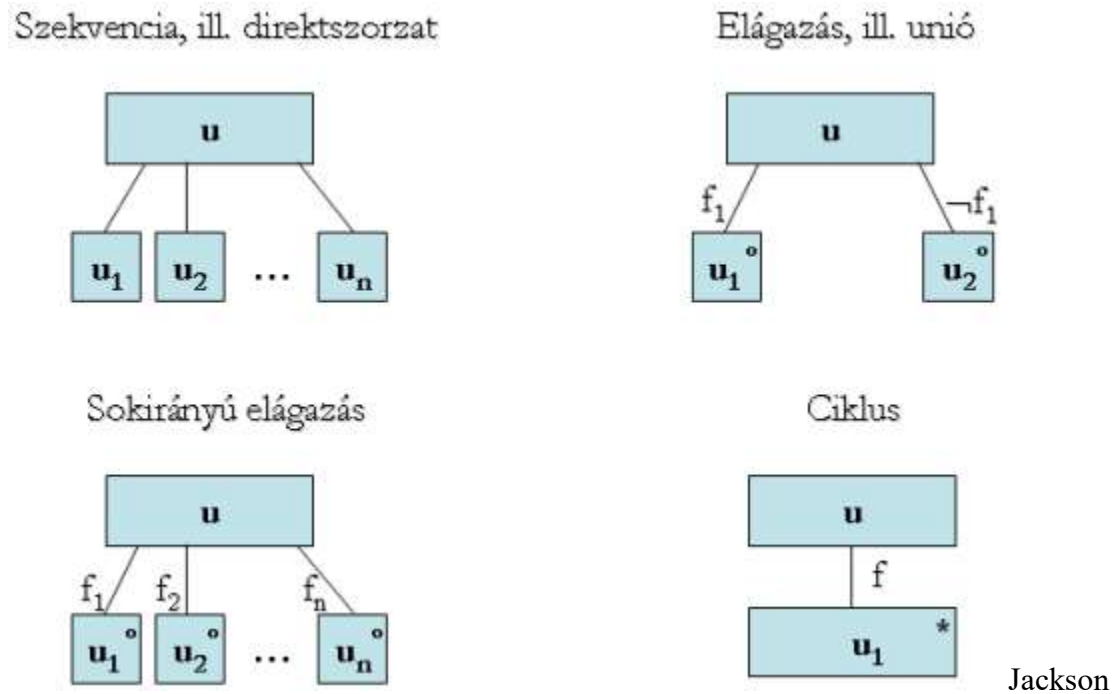
Vissza a tartalomjegyzékhez

3. Jackson-diagramok

A harmadik rajzos eszköz az adat- és az algoritmikus szerkezetek leírására egységes ábrakészletet definiál. Ahogyan az algoritmikus alapszerkezetek a szekvencia, az elágazás és a ciklus, úgy a nekik megfelelő adatszerkezetek a direktszorzat, az unió, illetve a sokaság. Ezeknek a Jackson-diagramban háromféle struktúra felel meg.

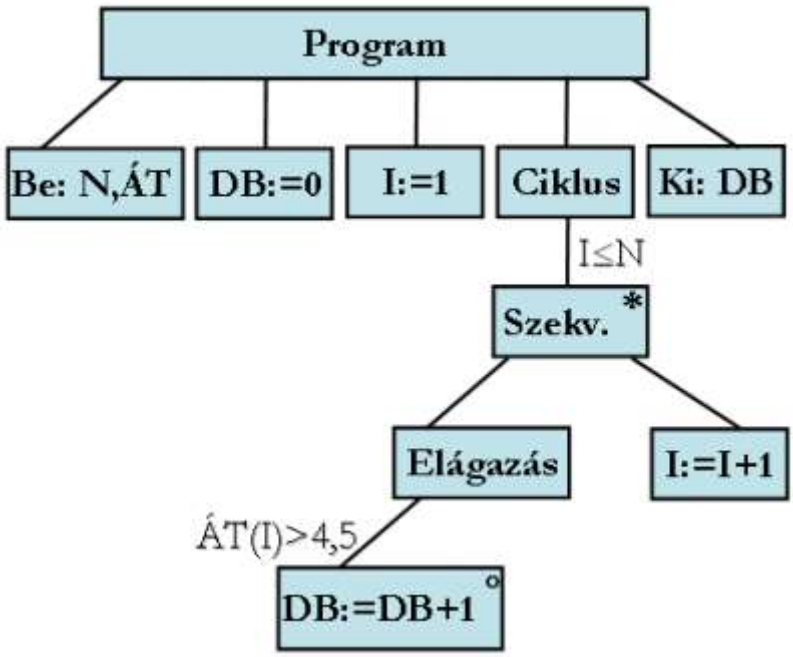
Szekvencia esetén az utasításokat balról jobbra haladva kell végrehajtani, a kétféle elágazásnál az igaz feltételű, jobb felső sarkában kis karikával (o) jelölt téglalap tartalmát, ciklusnál pedig mindaddig, amíg a feltétel teljesül, a jobb felső sarkában csillaggal (*) jelöltét.

Az új eszköz egyértelmű előnye az algoritmus- és adatleírás egységessége, áttekinthetősége azonban az eddigieknél is rosszabb. A szekvenciát itt azonnal, mint sok utasítás szekvenciáját definiáltuk, a többi alapelem a szokásossal megegyező.



Jackson diagramok.

Nézzük ezzel is a példánkat!



Példaprogram Jackson diagrammal.

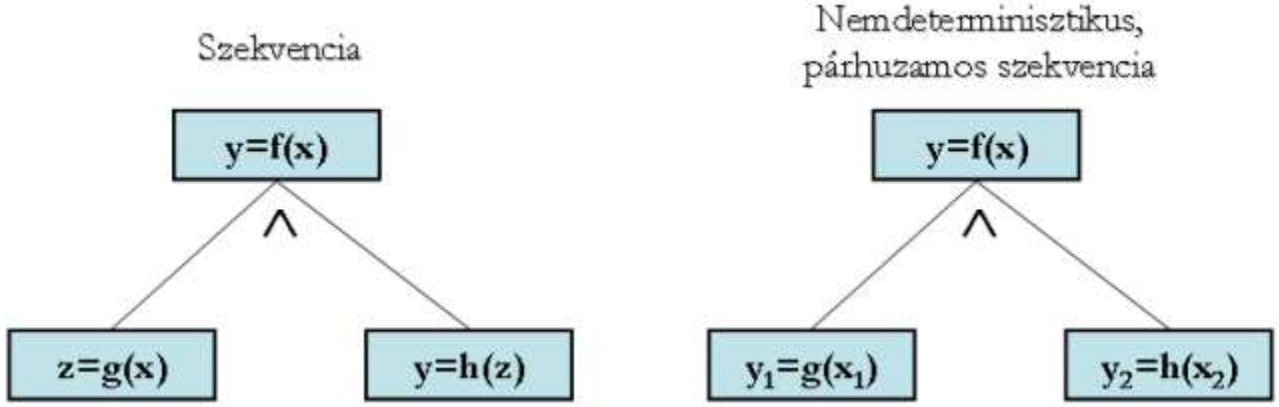
A példában szereplő Program, Ciklus, Szekv., Elágazás elnevezésű csomópontok helyett konkrét feladatokban valami „értelmes” elnevezést célszerű alkalmazni.

Vissza a tartalomjegyzékhez

4. Leírás fával

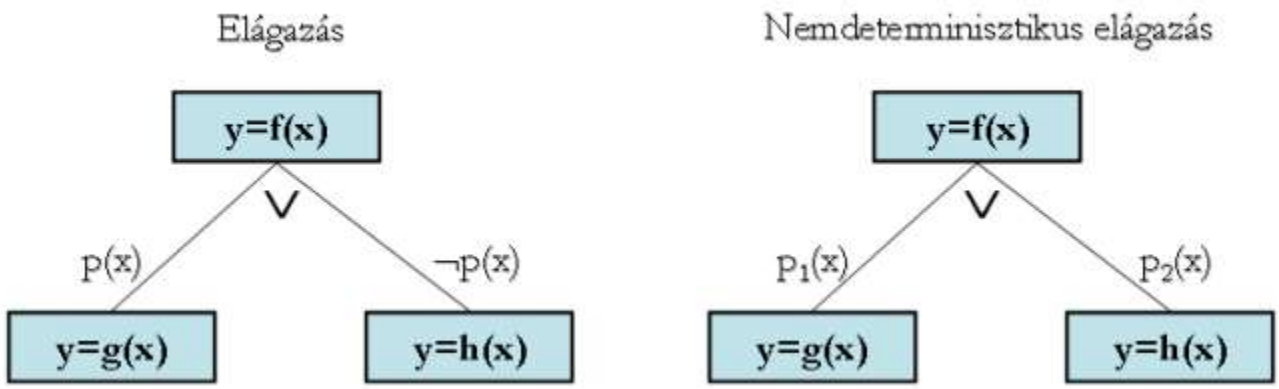
Ez a leíróeszköz nagyon hasonló az előzőhöz, a programgráfot **„és-vagy” fával** írja le. A fa elágazásai **„és”** típusúak, ha mindegyik ágat végre kell hajtani; **„vagy”** típusúak, ha az ágak egyikét kell végrehajtani. A fa ciklust nem tartalmaz, a ciklusokat rekurzívan írhatjuk le. Az eszköz jellemzője, hogy alkalmas nemdeterminisztikus, illetve párhuzamos algoritmusok leírására is.

A fa csomópontjaiban ennél a módszernél mindig a szükséges input→output leképezés függvényformája szerepel, a fában lefelé haladva egyre inkább konkrétabb függvényekkel. Nézzük végig a szokásos algoritmikus szerkezeteket ezzel az eszközzel!



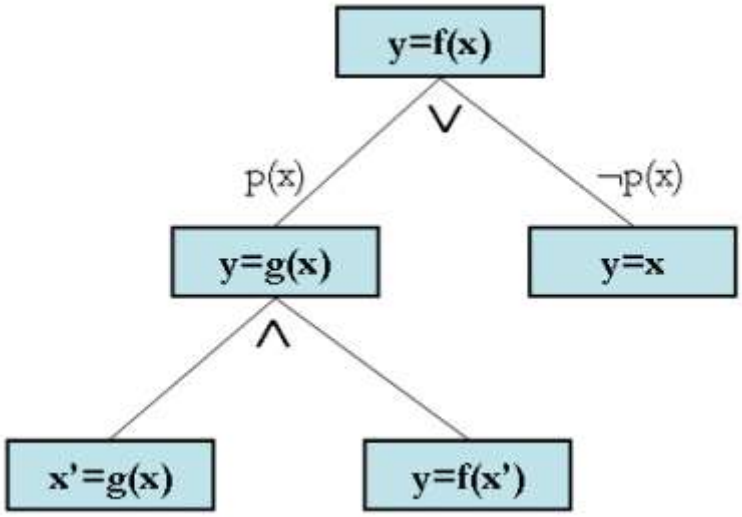
És-fa szerkezetek.

Mindkét fa lehet többágú is. A felírási formájuk teljesen egyforma, a determinisztikus sorrendet csak a leképezések input-output hozzárendelése rögzíti a bal oldali fában. A jobb oldali ágait tetszőleges sorrendben, sőt akár párhuzamosan is végre lehet hajtani.



Vagy-fa szerkezetek.

A jobb oldali fa lehet többágú is. A determinisztikus esetben a feltételek kötelezően kizáróak, és közülük az igaz feltételű ág hajtandó végre, a jobb oldali, nemdeterminisztikus elágazásban egyszerre több feltétel is lehet igaz értékű, és ilyenkor a nekik megfelelő ágak közül bármelyik végrehajtható.



Példaprogram És-Vagy-fával.

Itt az x' változó bevezetése jelenti a ciklusmag előbbi végrehajtását és utána az y=f(x) leképezés újbóli végrehajtását.

Beolvasás, illetve kiírás ennél az eszköznél nem létezik, helyette az input→output leképezést kell megadnunk. Elemi tevékenységként pedig a függvénydefiníció helyetti értékadás jel (:=) szerepel.

Ez az eszköz nyilvánvalóan szélesebb alkalmazási lehetőségű az eddigieknél, áttekinthetőnek azonban semmiképpen nem mondható.

Vissza a tartalomjegyzékhez

5. Leírás programozási nyelven

A gráfot alkalmazó, rajzos algoritmusleíró eszközök után áttérünk a szöveges eszközökre. Az egyik legelső szöveges algoritmusleíró nyelv egy egyszerű, de mégis általános célú programozási nyelv volt: az ALGOL60. Gyakorlati célú programozási nyelvként viszonylag szűkebb körben terjedt el, algoritmusleíró nyelvként azonban nagyon sok, az 1960-as, ’70-es években megjelent könyvben használták. Ugyanezt a célt szolgálja napjainkban az egyszerűsített Pascal, a C, az ELAN stb.

A bevezetőben említett negatív hatások mellett a programozási nyelv használatának még egy rossz tulajdonsága van: a szigorú szabályain könnyíteni kell, ha jó algoritmusleíró nyelvként szeretnénk használni, de ezt a könnyítést nehéz meghatározni, ráadásul kódolásnál csak zavarokat okozhat.

Vissza a tartalomjegyzékhez

6. Leírás mondatokkal

Sorszámozott utasítású programozási nyelvekhez (mint a BASIC, az assembly nyelvek, esetleg a Fortran) illeszkedik ez az eszköz. Utasításai sorszámozott mondatok.

A mondatok leírhatnak értékadást, beolvasást, kiírást, adott sorszámú mondattal folyatatást, valamint feltételes szerkezetet.

Értékadás: 5. Legyen I értéke I+1!

Beolvasás: 6. Olvasd be a K-t!

Kiírás: 7. Írd ki I*I-t!

Folytatás: 8. Folytasd a 15. soron!

Feltétel: 9. Ha I<N akkor ... ← itt, az előző négy sorbeli bármelyike lehet.

Ennél az eszköznél a követést a sorszámok teszik lehetővé. Az így leírt programnak azonban semmi szerkezete nem fog látszani, áttekinthetetlen, olvashatatlan lesz.

Szinte semmi előnye és rengeteg hátránya van. Csak a történelmi szerepe miatt írtuk róla e néhány sort.

Elrettentés kedvéért nézzük azonban meg ezzel az eszközzel is a példánkat!

1. Olvasd be N-t és ÁT-ot!

2. Legyen DB értéke 0!

3. Legyen I értéke 1!

4. Ha I>N, akkor Folytasd a 10. sornál!

5. Ha ÁT(1)>4,5, akkor Folytasd a 7. sornál!

6. Folytasd a 8. sornál!

7. Legyen DN értéke DB+1!

8. Legyen I értéke I+1!

9. Folytasd a 4. sornál!

10. Írd ki DB értékét!

11. Vége.

Vissza a tartalomjegyzékhez

7. Leírás mondatszerű elemekkel

A mondatszerű elemekkel való leírás szerepel ebben a füzetben, és a sorozat összes többi tagjában is. Az előző részekben az adatok és az algoritmikus elemek megismerésekor ezt használtuk, emiatt úgy gondoljuk, hogy újbóli részletes ismertetésére itt már nincs szükség.

Elemei nem teljes mondatok, hanem mondatszerű elemek (befejezetlen, hiányos mondatok).

Alapvető előnye, hogy struktúrái megfelelnek a szokásos Neumann-elvű programozási nyelv i struktúráknak, ezért kódolása rendkívül egyszerű. Egyik legalkalmasabb például az ELAN nyelv, amelyben a kódolás gyakorlatilag angolra fordítást jelent.

Csak a strukturált programozás alapszerkezeteit tartalmazza, így ettől eltérni nem lehetséges. Megfelelő programkészítési elvek használata esetén egy optimális algoritmusleíró eszközt kapunk.

Bevezető példánk ezzel az eszközzel:

Program:
Be: N, ÁT [N természetes szám, I≤ÁT[I]≤N I=1..N]
DB:=0
Ciklus I=1**-től** N**-ig**
Ha ÁT[I]>4,5 **akkor** DB:=DB+1
Ciklus vége
Ki: DB
Program vége.

Vissza a tartalomjegyzékhez

8. Leírás absztrakt függvényekkel

A szöveges algoritmusleíró eszközök után áttérünk a matematikai eszközökkel való algoritmusleírásra. Egy ilyen eszközzel fogunk megismerkedni, amely formulákkal írja le a programot. A legegyszerűbb formulák a szokásos algoritmikus szerkezetekhez tartoznak:

Szekvencia: P=**SEQ**(Q,R)

Elágazás: P=**IF**(f;Q,R)

Ciklus: P=**DO**(f;Q), illetve P=**UNTIL**(Q;f)

A kisbetűs jelölés logikai formulát, a nagybetűs pedig utasítást jelent. Az utasításokat a formulán belül vessző, a feltételeket az utasításoktól pontosvessző választja el. Szekvencián belül kettőnél több utasítás is lehet.

Kicsit bonyolultabbak már a *Programozási tételek*kel foglalkozó leckékben szereplő típusalgoritmusok formulái. Nézzünk meg ezek közül néhányat.

Az Elemtípus típusú elemeket tartalmazó, N elemű X vektorban van-e „t” tulajdonságú elem:

P=**Eldöntés**(*Elemtípus*;t;N,X,VAN)

Az Elemtípus típusú elemeket tartalmazó, N elemű X vektor maximális értékű eleme sorszámának a meghatározása:

P=**Maximumkiválasztás**(*Elemtípus*;N,X,MAX)

Az Elemtípus típusú elemeket tartalmazó, N elemű X vektor „t” tulajdonságú elemei sorszámainak kigyűjtése a DB elemű Y vektorba:

P=**Kiválogatás**(*Elemtípus*;t;N,X,DB,Y)

Nyilvánvaló, hogy ez az eszköz sem eredményez áttekinthető programot, de ennek nem is az a célja. Elsősorban arra a célra szolgál, hogy automatikusan, programmal elemezzünk algoritmusokat, s matematikai eszközökkel állapítsuk meg jellemzőiket.

Nézzük meg ezzel az eszközzel szokásos példánkat!

Program=**SEQ**(Be:N, Be:ÁT, DB:=0, I:=1,
DO(I≤N; **SEQ**(**IF**(ÁT[I]>4.5; DB:=DB+1,),
I:=I+1)),
Ki: DB)