Az algoritmusokról általában

Az algoritmus fogalma

- Egy (rész)probléma megoldásának mikéntjét egy eljárási renddel, egy ún. algoritmussal adhatjuk meg.
- Az **algoritmus**, tehát **jól ismert, megengedett lépésekből, tevékenységekből álló véges utasítássorozat, amelyet követve mindig elérjük a kívánt célt**.

Reprezentációs technikák

- Hogyan adhatunk meg egy algoritmust?
- Ennek egyik legtermészetesebb módja az ún. **mondatszerű leírás**, ahol az egyes lépéseket egy természetes emberi nyelven (például magyarul) megfogalmazott mondatok sorozatával adjuk meg.

Előjel függvény

•Meg szeretnénk fogalmazni, hogyan működik a matematikában alkalmazott előjel függvény. Mit jelent tehát az `y=sign(x)` jelölés?

1. Ha `x` egyenlő nullával, `y` értéke legyen `0`!
2. Különben ha `x` nagyobb, mint nulla, `y` értéke legyen `+1`!
3. Minden más esetben a függvény adjon `-1` értéket!

•Természetesen megfogalmazhatjuk ezt másképpen is anélkül, hogy a jelentés megváltozna.

1. Ha a bemenet (x) nulla a kimenet (y) is ennyi legyen!
2. Negatív bemenet esetén a kimenet legyen -1!
3. Pozitív bemenet esetén a kimenet legyen +1!
4. Ezeket **alternatív algoritmus**-oknak hívjuk.

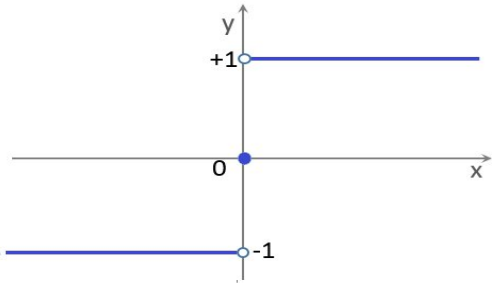
További módszerek

- A mondatszerű leíráson kívül számos további megjelenítési módszer, reprezentációs technika terjedt el.

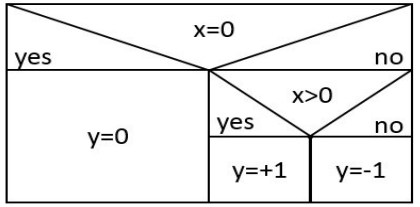
Algebra-szerű reprezentáció

$x \in \Re$
$y \in \{-1, 0, +1\}$
$\forall x, x>0 \Rightarrow y=+1$
$\forall x, x<0 \Rightarrow y=-1$
$x=0 \Rightarrow y=0$

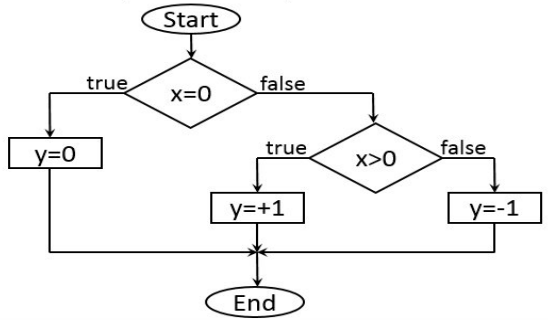
Grafikus reprezentáció



Struktogrammos reprezentáció



Folyamatábrás reprezentáció



Pszeudokódos reprezentáció

```
if x=0 then
  y=0
else
  if x>0 then
    y=+1
  else
    y=-1
  endif
endif
```

Programnyelven történő reprezentáció

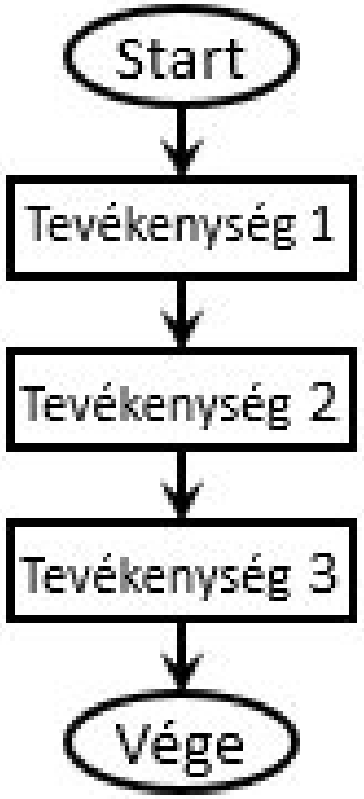
```
if (x == 0){
   y = 0
   }
else{
   if (x>0){
      y = +1
      }
   else{
      y = -1
      }
   }
```

Alapszerkezetek

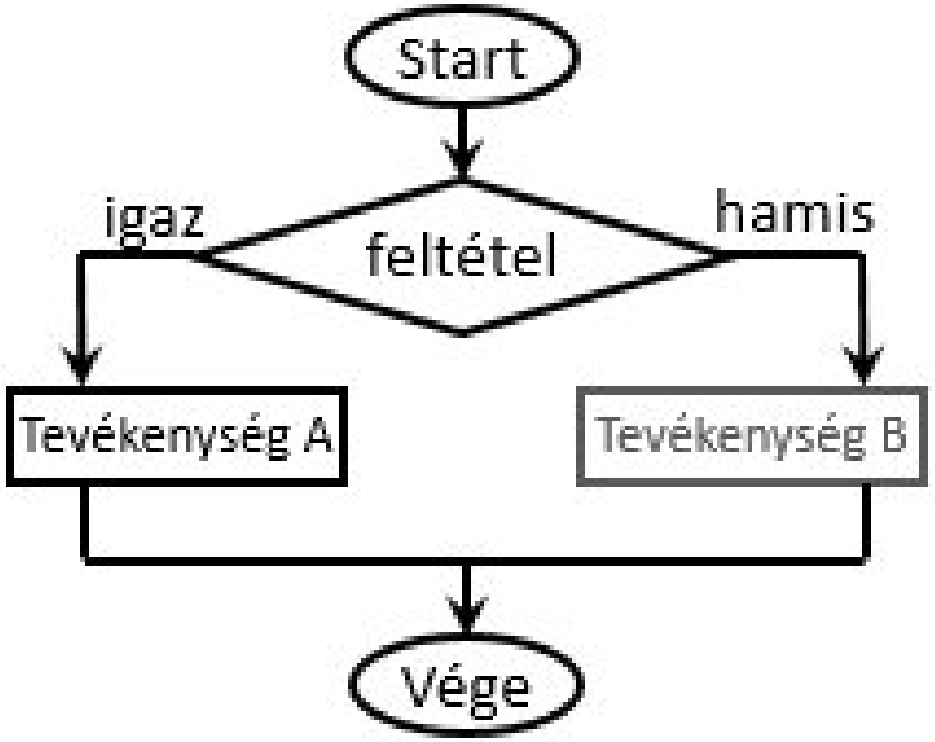
Az algoritmusok háromféle alapszerkezetből épülnek fel:

- **Szekvencia**
  - Az egyes jól ismert elemi lépések egymás utáni sorrendjének megadását jelenti.
  - egyértelmű melyik utasítással kell kezdenünk a végrehajtást,
  - melyik lesz a második, harmadik, stb. lépés.
  - Az `i+1.` lépés csak akkor kezdhető meg, amikor az `i.` befejeződött.
- **Elágazás**
  - Akkor használjuk, ha választanunk kell lehetőségek közül
- **Ismétlés**
  - Egyes problémák megkövetelik azt, hogy egy lépést vagy lépéssorozatot egymás után többször is végrehajtsunk, azaz ciklikusan ismételjünk.

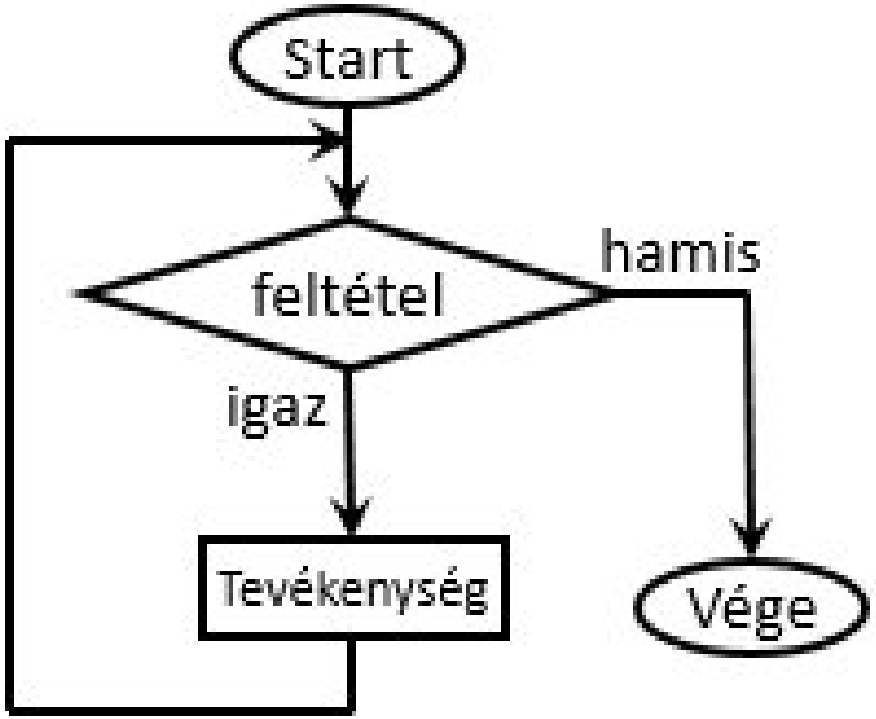
Szekvencia



Elágazás



Ismétlés



Az algoritmus tulajdonságai

Teljesség

Egy probléma megoldása során minden eshetőségre rendelkeznünk kell akciótervvel, minden lehetőséget kezelnünk kell.

Hogyan jussunk el a 2. emeletről az 5. emeletre lifttel?

1. Nyomd meg a lift hívógombját!
2. Ha megérkezett a lift, szállj be!
3. Nyomd meg az ‘5’ gombot!
4. Várj!
5. Ha az ajtó kinyílik megérkeztél. Szállj ki!

Végesség

Már az algoritmus definíciójában is megjelenik, a „véges utasítássorozat” kifejezés. Egy algoritmus nem tartalmazhat a végtelenségig ismétlődő ciklust.

Hogyan juthatunk el villamossal a klinikára?

1. Várj a villamosra!
2. Ha megjött, szállj fel!
3. Vegyél jegyet!
4. Ülj le!
5. Ha megérkeztél, szállj le!

Félreérthetetlenség

Egy algoritmus jól ismert elemi lépésekből kell álljon.

Hogyan készítsünk sült csirkét?

1. Tedd be a csirkét a sütőbe!
2. Állítsd be a hőmérsékletet!
3. Várj, amíg kész lesz!
4. Szolgáld fel!

Determinisztikusság

Egy algoritmusnak azonos feltételek mellett mindig ugyanazt az eredményt kell szolgáltatnia.

Hogyan legyünk milliomosok?

1. Vegyél egy lottót!
2. Húzd le a kívánt számokat!
3. Várj a nyereményre (vagy szomorkodj)!

Egy algoritmus módosításának lehetőségei

Általánosítás

Egy speciális eset kezelése helyett egy szélesebb körű megoldás létrehozására való.

1. Deríts ki a nettó árat!
2. Vedd ennek a 127%-át (27%-os ÁFA kulcsot feltételezve)!
3. A kapott érték a bruttó ár.

Ez az algoritmus csak 27%-os ÁFA esetén használható, ami elég speciális.

Általánosítsuk változtatható ÁFA kulcsot alkalmazva!

1. Deríts ki a termék nettó árát és ÁFA kulcsát!
2. Növeld az értéket az aktuális adókulcsnak megfelelően!
3. A kapott érték a bruttó ár.

Kiterjesztés

Néha rájövünk, hogy bizonyos feltételek eseten a problémák egy teljesen új körét is kezelnünk kell.

1. Add meg a ledolgozott órák számát és az órabért!
2. Szorozd össze az előbbi két értéket!
3. Vond le az adókat/illetékeket!
4. Ami marad az a fizetés.

De mi a helyzet a vezetőkkel?

1. Ha a vezetőről van szó, akkor folytasd a 5. lépésnél, különben menj tovább!
2. Add meg a ledolgozott órák számát és az órabért!
3. Szorozd össze az előbbi két értéket!
4. Folytasd a 6. lépéssel!
5. A profit felét rendeld a főnökhöz!
6. Vond le az adókat/illetékeket!
7. Ami marad az a fizetés.

Ezzel az eredeti algoritmus ki lett terjesztve egy újabb részalgoritmussal.

Beágyazás

Előfordul sokszor, hogy egy nagyobb probléma kisebb részproblémákból áll, amelyeknek a megoldására viszont már korábban hoztunk létre egy jól működő algoritmust.

1. Ha az érték egyenlő nullával, akkor az eredmény legyen 0!
2. Különben ha az érték nagyobb, mint nulla, az eredmény legyen +1!
3. Minden más esetben -1 legyen az eredmény!

Ha már tudjuk, hogyan kell egy szám abszolút értékét meghatározni, akkor azt felhasználva (beágyazva) egyszerűsíthetjük a fenti algoritmust.

1. Ha az érték egyenlő nullával, akkor az eredmény legyen 0!
2. Különben az eredmény legyen az eredeti érték és a saját abszolút értékének hányadosa!

Bolondbiztossá tétel

Néha azért kell módosítani egy algoritmust (programot), hogy reagálni tudjon a felhasználó esetleges irreális aktivitására.

1. Add meg a területet!
2. Vedd ennek a négyzetgyökét!
3. A kapott szám 4-szerese lesz a négyzet kerülte. Írd ki ezt!

Egy negatív szám négyzetgyöke csak komplex szám lehet, amit viszont nehezen értelmezhetünk kerületként.

1. Add meg a területet!
2. Ha a megadott érték nem negatív, akkor folytasd az 5. lépéssel, különben menj tovább!
3. Írd ki az alábbi üzenetet: „Hibás adat. Adj meg újat!”
4. Folytasd az 1. lépésnél!
5. Vedd ennek a négyzetgyökét!
6. A kapott szám 4-szerese lesz a négyzet kerülte. Írd ki ezt!

Szójegyzék

algoritmus	**Egy probléma megoldását szolgáltató jól ismert, elemi tevékenységek véges sorozata, amelyet végrehajtva mindig egyértelműen elérjük a kívánt célt.**
alternatív algoritmus	Azonos módon viselkedő, de különböző felépítésű algoritmusok.
általánosítás	Valami szükségtelenül specifikus dolog (mint egy konstans érték) lecserélése valami megfelelően általánosra (mint egy változó vagy egy paraméter). Az általánosítás sokoldalúvá teszi a programot a valószínű újrahasznosításhoz, absztrakciót biztosít.
beágyazás	Elszeparált (rész)algoritmusok felhasználása egy komplexebb problémát megoldó algoritmusban.
bolondbiztossá tétel	Az algoritmus felkészítése speciális esetekre, amelyek főként a helytelen felhasználói interakciókból származnak
determinisztikusság	Egy algoritmusokra jellemző tulajdonság, mely biztosítja a nem sztochasztikus viselkedést, azaz a véletlenszerűségtől mentességet.

Folyt.

elágazás	**Egy feltételtől függően kiválasztott utasításcsoport végrehajtás, miközben más utasítások végrehajtását esetleg kihagyjuk.**
félreérthetetlenség	Az algoritmusok egyik tulajdonsága, miszerint minden lépésnek egyértelműnek kell lennie a félreértések elkerülése végett.
ismétlés	Tevékenységek többszöri végrehajtása bizonyos körülmények esetén.
kiterjesztés	Az algoritmus kibővítése, mellyel problémák egy újabb körét is meg lehet oldani.
szekvencia	Utasítások egymás után (lépésről-lépésre történő) végrehajtása.
teljesség	Az algoritmusok azon tulajdonsága, mely révén biztosak lehetünk abban, hogy a végrehajtás során előfordulható lehetőségeket kezeltünk.
végesség	Az algoritmusok azon jellemzője, miszerint a lépéssorozat biztosan befejeződik véges időn belül.