Programozás I.

1. előadás

Sergyán Szabolcs
sergyan.szabolcs@nik.uni-obuda.hu

Óbudai Egyetem
Neumann János Informatikai Kar

2012. szeptember 10.

Tartalom

Tartalom

Mi az az algoritmus?

- 1957 előtt nem lehetett megtalálni az *algorithm* szót a Webster's szótárban.
- Az *algorism* szó volt a szótárban, melynek jelentése: aritmetikai eljárások elvégzése arab számokkal.
- Az első algoritmus, amivel találkozhatunk, az Euklideszi algoritmus volt.

Mai definíció

Jól meghatározott utasítások véges sorozata, amelynek bemenete egy bizonyos érték vagy értékhalmaz, és amely létrehoz valamilyen értéket vagy értékhalmazt kimenetként.

Algoritmus készítésének lépései

- A folyamatot elemi lépésekre bontjuk
- Figyelembe vesszük az összes felmerülő lehetőséget
- Ügyelünk, hogy az algoritmus véges sok lépésben véget érjen

Példa (1)

Kávéautomata

1. Válaszd ki, amit inni akarsz!
2. Dobj be pénzt!
3. Ha nem elég a bedobott pént, akkor menj a 2. lépésre!
4. Várd, amíg elkészül a kávé!
5. Vedd el a kávét!
6. Vedd el a visszajáró pénzt!

Példa (2)

Nem indul a motor

1. Ellenőrizd a benzintankot!
2. Ha nincs benne benzin, akkor töltsd tele!
3. Ha elindul a motor, akkor VÉGE.
4. Ellenőrizd az akkumulátort!
5. Ha nincs megfelelően feltöltve, akkor töltsd fel!
6. Ha elindul a motor, akkor VÉGE.
7. Ellenőrizd a gyertyát!
8. Ha szükséges, cseréld ki!
9. Ha elindul a motor, akkor VÉGE.
10. Vidd el a szerelőhöz!

Példa (3)

Euklideszi algoritmus

Bemenet: Két pozitív egész szám: m és n.
Kimenet: A legnagyobb egész szám, amely m-nek és n-nek is osztója.
Algoritmus

1. Osszuk el m-et n-nel és legyen r az osztási maradék.
2. Ha $r = 0$, akkor véget ér az algoritmus és n a kimenet.
3. Egyébként $m \leftarrow n$ és $n \leftarrow r$.
4. Ugrás az 1. lépésre.

m	n	r
120	48	24
48	24	0

Példa (4)

Adjuk meg egy szó szótárban történő keresésének algoritmusát!

Az algoritmusokkal szembeni elvárások

- **Végesség**: Véges számú lépést követően véget kell érnie az algoritmusnak.
- **Meghatározottság**: Az algoritmus minden egyes lépését minden lehetséges esetre precízen definiálni kell.
- **Bemenet**: Egy algoritmusnak nulla vagy annál több bemenete lehet, amik az algoritmus kezdetekor ismertek.
- **Kimenet**: Egy algoritmusnak nulla vagy annál több kimenete van, amelyek meghatározott viszonyban vannak az algoritmus bemeneteivel.
- **Hatékonyság**: Egy számítógép által végrehajtott algoritmustól elvárt, hogy gyorsabban működjön annál, mintha számítógép nélkül hajtottuk volna végre.

Tartalom

Megengedett vezérlési szerkezetek

- **Szekvencia**: Egy utasítást közvetlenül egy másik után végzünk el.
- **Elágazás**: Adott (legalább) 2 darab feltétel-program páros. A teljesülő feltételhez tartozó programrész (utasítások) végrehajtása.
- **Ciklus**: Megadott feltétel teljesülése esetén egy programrész (ciklusmag) többszöri végrehajtása.

Tartalom

Algoritmus leíró eszközök

- Mondatok
- Blokkdiagram
- Struktogram
- Jackson ábra
- Pszeudokód
- Programnyelv

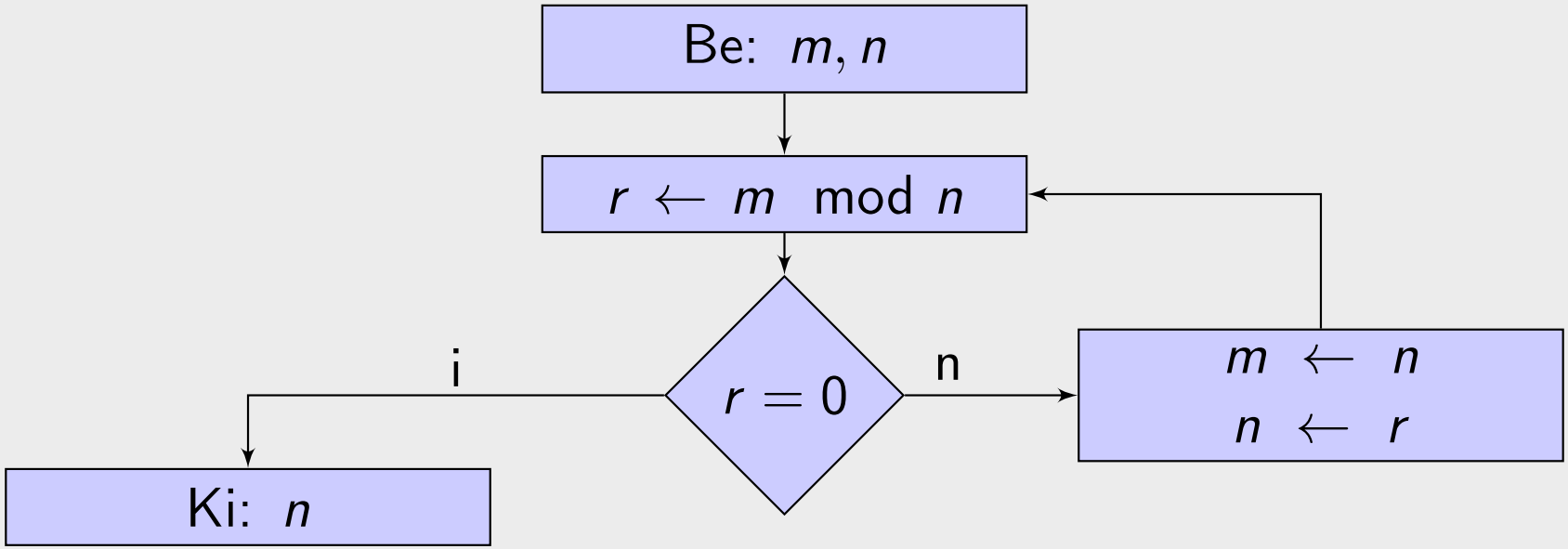
Algoritmus leíró eszközök

- Mondatok
- Blokkdiagram
- Struktogram
- Jackson ábra
- Pszeudokód
- Programnyelv

Blokkdiagram

Teendők és kérdések összekötése nyilakkal

Euklideszi algoritmus

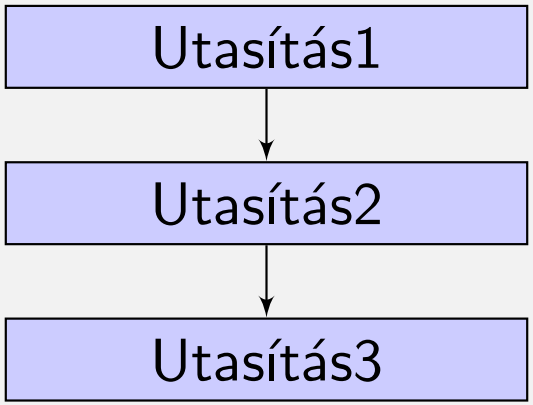


Megjegyzés

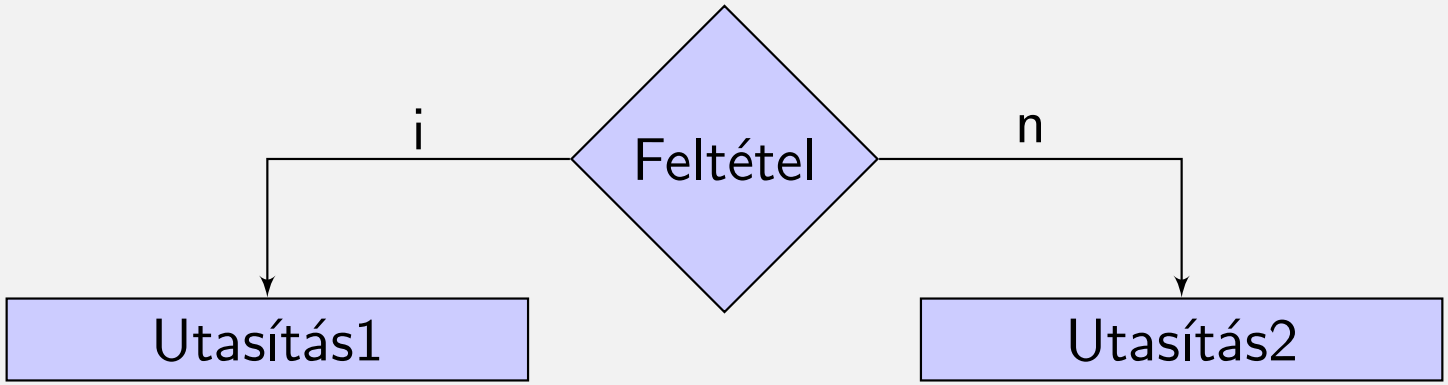
Fontos különbség van az értékadás jele és az összehasonlító egyenlőség jele között

- $\leftarrow$ Értékadást jelöljük vele. A bal oldalon mindig egy változó áll, ami értékül kapja a jobb oldali kifejezés aktuális értékét.
- $=$ Összehasonlító egyenlőséget jelölünk vele. Igaz ha a két oldalán kifejezés megegyezik egymással, egyébként pedig hamis.

Szekvencia ábrázolása blokkdiagramon

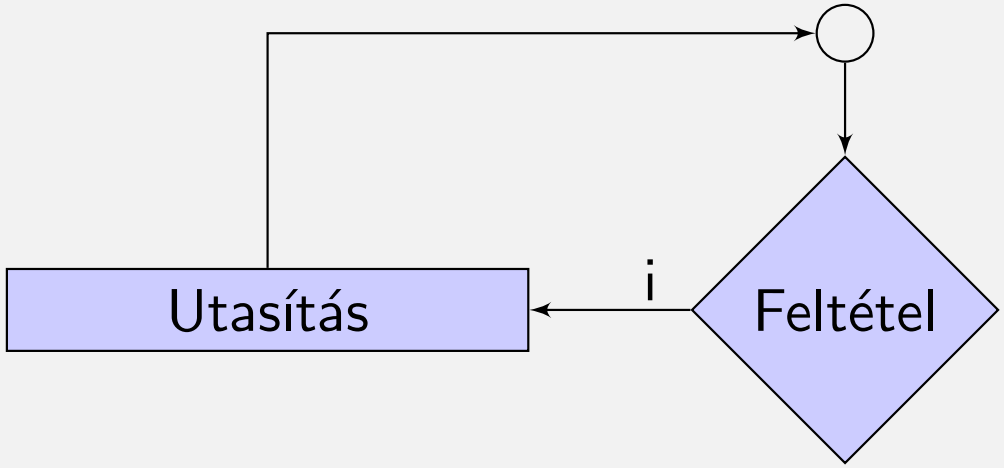


Elágazás ábrázolása blokkdiagramon



Ciklus ábrázolása blokkdiagramon

Nincs külön jelölés a ciklusra, elágazással viszont ábrázolható.



Problémák a blokkdiagrammal

- Nyilak és vonalak kesze-kusza rendszere
- Teljesen ad-hoc elrendezésű, két ugyanolyan algoritmus leírása a rajzoló kénye-kedve szerint akár teljesen más elrendezésű is lehet

Struktogram

Teendők és kérdések strukturáltan kötött, mindig téglalap alakú képi reprezentációja

Euklideszi algoritmus

Be: m, n
$r \leftarrow m \mod n$
Amíg $r \neq 0$
$m \leftarrow n$
$n \leftarrow r$
$r \leftarrow m \mod n$
Ki: n

Szekvencia ábrázolása struktogramon

Utasítás1
Utasítás2
Utasítás3

Elágazás ábrázolása struktogramon

Ciklus ábrázolása struktogramon

Problémák a struktogrammal

- Nehezen módosítható
- Az elkészítése és az értelmezése olykor nehézkes

Pszeudokód

- Teendők és kérdések kötött kifejezésekkel való szöveges leírása
- Nagy mértékű hasonlóságot mutat a programnyelvek sturktúrájához

Euklideszi algoritmus

Be: m, n
$r \leftarrow m \mod n$
Ciklus amíg $r \neq 0$
    $m \leftarrow n$
    $n \leftarrow r$
    $r \leftarrow m \mod n$
Ciklus vége
Ki: n

Szekvencia megadása pszeudokóddal

Utasítás1
Utasítás2
Utasítás3

Elágazás megadása pszeudokóddal

Ha feltétel **akkor**
    Utasítás1
különben
    Utasítás2
Elágazás vége

Többirányú elágazás megadása pszeudokóddal

Elágazás
    Feltétel1 **esetén**
        Utasítás1
    Feltétel2 **esetén**
        Utasítás2
    Feltétel3 **esetén**
        Utasítás3
    különben
        Utasítás4
Elágazás vége

Ciklus megadása pszeudokóddal

Előltesztelős ciklus

Ciklus amíg feltétel
    Utasítás
Ciklus vége

Akkor lépünk be, illetve addig maradunk a ciklusban, amíg a feltétel igaz

Hátultesztelős ciklus

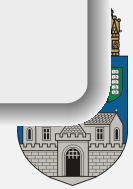
Ciklus
    Utasítás
Ciklus amíg feltétel

Addig maradunk a ciklusban, amíg a feltétel igaz

Számlálós ciklus

Ciklus $i \leftarrow i_0$**-tól** i_1**-ig**
    Utasítás
Ciklus vége

Az i ciklusváltozó értéke minden ciklusban 1-gyel növekszik. Akkor lépünk ki a ciklusból, ha i értéke már meghaladja az i_1 értéket.



Tartalom

Strukturált programozás

- Strukturált programnak tekintjük azokat a programokat, amelyek csak a megengedett elemi programokat tartalmazzák a megengedett programkonstrukciók (vezérlési szerkezetek) alkalmazásával.
- Elemi programok
  - üres program
  - értékadás (állapot változtatás)
    jele: $\leftarrow$
- Megengedett konstrukciók
  - szekvencia
  - elágazás
  - ciklus
- Bizonyítható, hogy a fenti szabályok megtartásával minden algoritmussal megoldható feladatra adható is megoldás.

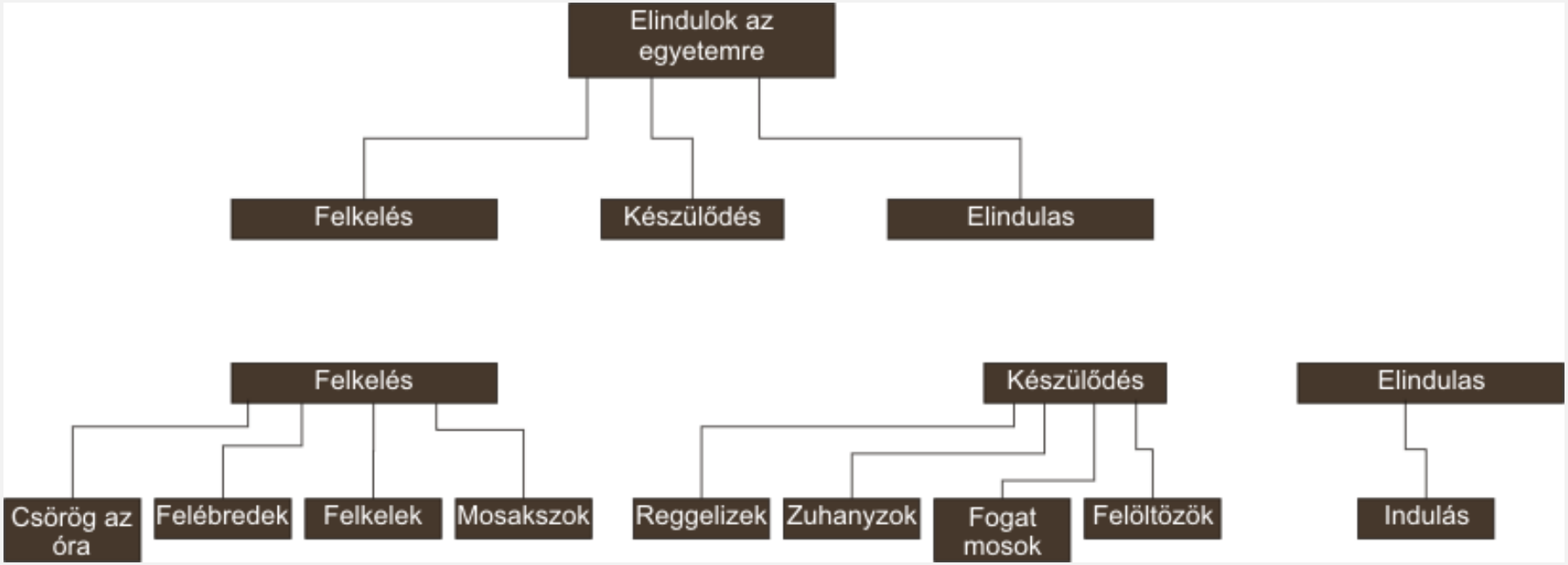
Moduláris programozás

- Az elkészítendő programot egyetlen utasítással szeretnénk megoldani.
- Ha ez nem sikerül, akkor több utasítással próbálkozunk, melyek egyes részfeladatokat valósítanak meg.
- Addig folytatjuk ezt a részfeladatokra bontást, míg minden részfeladatot sikerül megvalósítani elemi utasítások felhasználásával.

Összefoglalva: A teljes feladatot részekre bontjuk, majd ezeket a visszavezetés módszerével megoldjuk.

Moduláris programozás

A moduláris programozást az alábbi Jackson ábra szemlélteti



Moduláris programozás

A moduláris programozás pl. függvények (illetve eljárások) hívásával tudjuk megvalósítani.

- A függvények önálló részprogramok, melyeknek
- lehetnek bemenetei és
- visszatérési értékei.

Változók

- Az algoritmusok megvalósításához változókra van szükségünk (ld. Euklideszi algoritmus).
- Az egyes változóknak az algoritmusok számítógépes implementációjánál típust kell megfeleltetni.
- Sok esetben az alkalmazott változó típustól függ, hogy milyen algoritmussal oldható meg egy probléma.

Elemi változó típusok

- Egész típusok
  - Hány bájton ábrázolva?
  - Előjeles/előjel nélküli?
- Lebegőpontos típusok
  - Hány bájton ábrázolva?
  - Vigyázat! Nem ábrázolható minden valós szám!
- Karakter típus(ok)
- Logikai típus

Összetett típusok

- Tömbök
- Stringek
- Struktúrák
- Halmazok
- Objektumok
- ⋮

Változók jellemzői

- Élettartam
- Statikus/dinamikus
- Konstans/változtatható
- Hatókör (lokális/globális)
- Függvény paramétereként érték/cím szerint átadott

Tartalom

Előadásra járó hallgatók számának meghatározása

- Hányan vannak itt a teremben?
- Hogyan tudnánk ezt hatékonyan megszámolni?

Algoritmus lépésszáma

- Egy algoritmus lépésszáma alatt azt értjük, hogy hány elemi műveletet (értékadás, összehasonlítás, beolvasás, kiiratás, stb.) kell végrehajtani adott bemenet mellett.
- Egy algoritmus futási ideje függ az algoritmust megvalósító programtól, a programozási nyelvtől, a számítógéptől, stb.
- Algoritmuselméletben a futási időt a lépésszámmal jellemezzük.

Sorozat rendezése

- Bemenet: egy számsorozat: $< a_1, a_2, \ldots, a_n >$
- Kimenet: növekvő módon rendezett sorozat: $< a'_1, a'_2, \ldots, a'_n >$, ahol $a'_1 \leq a'_2 \leq \ldots \leq a'_n$
- Javított beszúró rendezés pszeudokódja:

Ciklus $j \leftarrow 2$-**től** n-**ig**
    $k \leftarrow A[j]$
    $i \leftarrow j - 1$
    Ciklus amíg $i > 0$ **és** $A[i] > k$
        $A[i + 1] \leftarrow A[i]$
        $i \leftarrow i - 1$
    Ciklus vége
    $A[i + 1] \leftarrow k$
Ciklus vége

- Szükséges lépésszám egy sorozat rendezéséhez?

Sorozat rendezése

- A futási idő függ a bemeneti sorozattól (rendezett, majdnem rendezett, fordítva rendezett, stb.)
- A futási idő függ a bemeneti sorozat méretétől
- A futási időt felülről szeretnénk becsülni, tehát felső korlátot akarunk meghatározni

Futási idő analízisének fajtái

- Legrosszab eset analízis
  - $T(n)$: a maximális futási idő, amely bármely n elemű sorozat esetén a rendezéshez legfeljebb szükséges
- Átlagos eset analízis
  - $T(n)$: a várható futási idő, amely az n elemű sorozatok rendezéséhez szükséges
- Legjobb eset analízis
  - Magunkat verjük át, ha ezzel foglalkozunk

Lépésszám meghatározása

- Rendezett eset: $3 \cdot (n-1)$ darab értékadás és $n-2$ darab összehasonlítás. Így a lépésszám ($n \geq 2$ esetén):

$$T(n) = 3 \cdot (n-1) + (n-2) = 4n - 5$$

- Fordítva rendezett eset: $\frac{(n+2)(n-1)}{2} + \frac{n(n-1)}{2}$ darab összehasonlítás és $3 \cdot (n-1) + 2 \cdot \frac{n(n-1)}{2}$ értékadás. Így a lépésszám:

$$T(n) = \frac{(2n+2)(n-1)}{2} + (n+3)(n-1)$$

Sorozat rendezés

Megjegyzés

A pontos lépésszám meghatározás persze függ attól, hogy

- a gép minden értékadást ugyanannyi idő alatt (ugyanannyi elemi lépéssel) hajt végre? (pl. $i \leftarrow j - 1$ vs. $k \leftarrow A[j]$)
- a gép minden összehasonlítást ugyanannyi elemi lépéssel valósít meg? (pl. $i > 0$ vs. $A[i] > k$)
- egy értékadás és egy összehasonlítás ugyanannyi elemi lépés?

Ezek viszont csak valamilyen konstans együtthatóval való szorzást jelentenek az egyes tagoknál. (Az együttható nem függ n-től!)

Nagy ordó jelölés

Legyenek $f(x)$ és $g(x)$ egyváltozós függvények.

$$f(x) = O\left(g(x)\right)$$

akkor és csak akkor, ha léteznek olyan M és x_0 pozitív valós számok, hogy minden $x > x_0$ esetén

$$|f(x)| \leq M \cdot |g(x)|$$

Szó keresése szótárban

Hogyan tudunk hatékonyan szót keresni egy szótárban?

Tartalom

Feladatok

1. Írjuk le a másodfokú egyenlet megoldási algoritmusát folyamatábrával, struktogrammal és pszeudokóddal!
2. Adott két síkbeli pont: $P_1(x_1, y_1)$ és $P_2(x_2, y_2)$. Keressük a P_1-en és P_2-n áthaladó egyenesen az x_0 abszcisszájú pont y_0 koordinátáját. Adjon algoritmust a feladat megoldására!
3. Készítsen algoritmust, mely eldönti, hogy egy adott év szőkőév-e vagy sem!
4. Készítsen algoritmust, mely megadja, hogy egy adott év adott hónapja hány napból áll.
5. Készítsen algoritmust, amely egy pozitív egész számról eldönti, hogy prím-e vagy sem!
6. Készítsen algoritmust, amely bekéri egy tankör zh eredményeit, majd kiszámítja azok átlagát.