

Dr Strauber Györgyi – Sóti Lászlóné – Johanné Dukai Klára

A számítástudomány alapjai. II.

Programozási feladatok, feladatsorok, megoldások

2009.

Tartalomjegyzék

1. A feladatmegoldás lépései	3
2. Programozási tételek	5
2.1. Egy sorozathoz egy érték hozzárendelése.....	5
(Összegzés, Eldöntés, Kiválasztás, Megszámlálás, Keresés)	
2.2. Egy sorozathoz egy sorozat hozzárendelése	7
(Kiválogatás, Rendezések, Metszetképzés, Unióképzés, Összefuttatás, Backtrack)	
2.3. Programozási tételek rekurzív megvalósítása	13
2.4. Feladatok a programozási tételek gyakorlásához.....	18
2.5 Megoldások	22
3. Adattípusok megvalósításai	27
3.1. Algoritmus típus konstrukció	27
3.2. Sorozattípusok ábrázolása a memóriában	29
3.3. Lista típuskonstrukció	30
Láncolt ábrázolású lista (dinamikus)	
Láncolt ábrázolású lista (statikus)	
Láncolt ábrázolású lista (szekvenciális)	
Kiválogatási tétel újrafogalmazása listára	
Beszúrásos rendezés	
3.4. Verem típuskonstrukció	40
Láncolt ábrázolású verem	
Folytonos ábrázolású verem	
3.5. Sor típuskonstrukció.....	44
Láncolt ábrázolású sor	
Folytonos ábrázolású sor	
3.6. A prioritási sor típuskonstrukció	48
3.7. Feladatok az adattípusok megvalósításához.....	51
4. Feladatok és megoldások az elméleti rész anyagához.....	55
5. Ellenőrző kérdések	57
6. Vizsga Minta feladatok	59
Irodalomjegyzék	63

1. A feladatmegoldás lépései.

Az informatikai vagy számítástechnikai feladatok megoldása során először pontosan meg kell fogalmaznunk, körül kell írunk mi is a megoldásra váró probléma. Egyértelműen, tömören le kell írni, milyen adatokkal, milyen feltételekkel, milyen környezetben, mi az, amit meg kell oldani. Ez rendszerint a rendszerszervező feladata. Ezt nevezzük specifikációnak.

Definíció:

A **specifikáció** tehát egy olyan leképezést határoz meg, amely a bemenő értékekhez hozzárendeli a jó eredmény értékeket. Megkülönböztetjük a feladat és a program specifikációját.

A **feladat specifikálása** a feladat pontos szövegének megfogalmazása, a bemenő és kimenő adatok meghatározása, a bemenő és kimenő adatok kialakítása.

A **program specifikálása** a program környezetének meghatározása, a programmal szemben támasztott követelmények megadása (minőség, hatékonyság, gyorsaság, elő- és utófeltételek definiálása).

Következő lépés a megoldás menetének kialakítása, ennek szemléletes leírása. Ez általában a programtervező feladata. A feladat megoldási menetét véges számú lépések sorozatával kell megadni. Ez az algoritmus. Az algoritmus nem kötődik semmilyen konkrét program nyelvhez. A számítógép általános megfogalmazása szerint egy adatok feldolgozására készített eszköz az algoritmus, ami megadja a műveletek tartalmát és sorrendjét. Csak olyan módszer vezet jó eredményre, amely számol az előfeltételekkel, a tervezett feladatokat oldja meg, a kialakított műveleti sorrendek minden eshetőségre gondolnak és nyújtanak megoldást (fontos hogy szélsőséges eseteknél is eredményt szolgáltatassanak!). Fontos még a végrehajthatóság, az egyértelműség és a végesség is!

Definíció:

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

A feladatok általában bonyolultak, ezért érdemes részfeladatokra bontani. Az algoritmus átírása valamely programnyelvre, majd ezek eljárásokra, modulokra; al- és főprogramra bontása, már a **programozás** része és a programozó feladata. A főprogram eljárást vagy eljárásokat hív meg, melyek további eljárásokat hívhatnak, és paramétereket adnak át egymásnak.

El kell végezni a program **tesztelését**, és ellenőrzését is. Meg kell vizsgálni elvégzi-e azokat a feladatokat, melyeket kijelöltek és ha igen mennyire helyesen. Szükség esetén végig kell vezetni a megfelelő javításokat a programon. Az elkészült program leírása – a **dokumentálás**, amely tartalmazza a specifikációs leírásokat, a program leírását és az üzemeltetési információkat, amely szintén fontos része a feladat megoldásának. Meg kell adni a használathoz szükséges információkat is, a karbantartási feladatok környezethez igazíthatósági feltételeket.

Az algoritmusok néhány jellemzője:

Az algoritmus vizsgálatának szempontjai:

- milyen előfeltételekkel működik az algoritmus
- tényleg a tervezett feladatot oldja-e meg
- befejeződik-e az algoritmus minden lehetséges esetben
- van-e összefüggés az adatok mennyisége és a végrehajtási idő közt?

Az algoritmusok fajtái:

- Verbális algoritmus: szóvegesen adja meg a feladatot (szóban vagy írásban)
- Jelalgoritmus: a feladat megfogalmazása geometriai szimbólumokkal, matematikai jelölésekkel történik. Ilyen a folyamatábra, a stuktogram, vagy a döntési tábla.

Az algoritmusokban előforduló tevékenységek:

Alaptevékenységek:

- Adatok beolvasása a megadott változókba (*input tevékenység*)
- Adatok kiírása. Az adat lehet konstans, változó tartalom és kifejezés értéke (*output tevékenység*)
- A kifejezés értékének kiszámítása (*értékkadó tevékenység*)

Vezérlőtevékenységek:

- A számítógép végrehajtási sorrendje az utasítások leírt sorrendje (*szekvenciális vezérlő tevékenység*)
- A tevékenységek sorrendje egy bizonyos logikai kifejezés értékétől függően változik (*szelekciós vezérlő tevékenység*)
- Egy vagy több tevékenység ismétlése egy feltételtől függően (*Iterációs vezérlő tevékenység*)

Az algoritmusok leírására használt eszközök közül a leggyakoribbak a

- *mondatszerű leírás* (amely jól definiált formalizmusokat használó egyértelmű szóveges leírás)
- *stuktogram* (amely egy rajzos forma, ami a strukturált programozást támogatja)
- *folyamatábra* (jól áttekinthető rajzos forma, blokkdiagramnak is nevezik)

2. Programozási tételek

2.1. Egy sorozathoz egy érték hozzárendelése

Összegzés

Adott egy N elemű számsorozat: $A(N)$. Számoljuk ki az elemek összegét!

Eljárás:

```
S:=0
Ciklus I=1-től N-ig
    S:=S+A(I)
Ciklus vége
Eljárás vége.
```

Eldöntés

Adott egy N elemű sorozat és egy - a sorozaton értelmezett - T tulajdonság. Van-e a sorozatnak legalább egy T tulajdonságú eleme?

Eljárás:

```
I:=1
Ciklus amíg  $I \leq N$  és  $A(I)$  nem  $T$  tulajdonságú
    I:=I+1
Ciklus vége
VAN:= $I \leq N$ 
Eljárás vége
```

(„VAN” egy logikai változó, amely akkor és csak akkor igaz, ha $I \leq N$)

Igaz-e, hogy a sorozat minden eleme T tulajdonságú?

Eljárás:

```
I:=1
Ciklus amíg  $I \leq N$  és  $A(I)$   $T$  tulajdonságú
    I:=I+1
Ciklus vége
IGAZ:= $I > N$ 
Eljárás vége
```

Kiválasztás

Adott egy N elemű sorozat, egy - a sorozat elemein értelmezett - T tulajdonság, és tudjuk, hogy a sorozatban van legalább egy T tulajdonságú elem. A feladat ezen elem sorszámának meghatározása.

Eljárás:

I:=1

Ciklus amíg A(I) nem T tulajdonságú

I:=I+1

Ciklus vége

SORSZ:=I

Eljárás vége

Megszámlálás

Adott egy N elemű sorozat és egy - a sorozat elemein értelmezett - T tulajdonság. Feladat a T tulajdonsággal rendelkező elemek megszámlálása.

Eljárás:

S:=0

Ciklus I=1-től N-ig

Ha A(I) T tulajdonságú akkor S:=S+1

Ciklus vége

Eljárás vége.

Keresés

-Lineáris keresés

Általános feladat: N elemű sorozat; sorozat elemein értelmezett T tulajdonság. Van-e T tulajdonságú elem és ha van, akkor mi a sorszáma. (Eldöntés és kiválasztás együtt.)

Eljárás:

I:=1

Ciklus amíg I<=N és A(I) nem T tulajdonságú

I:=I+1

Ciklus vége

VAN:=I<=N

Ha VAN akkor SORSZ:=I

Eljárás vége.

-Logaritmikus keresés

Általános feladat: N elemű *rendezett* sorozat; egy keresett elem (X). Szerepel-e a keresett elem a sorozatban és ha igen, akkor mi a sorszáma.

Kihasználjuk, hogy a sorozat rendezett, így el tudjuk dönteni, hogy a keresett elem az éppen vizsgált elemhez képest hol helyezkedik el.

A, F: intervallum alsó és felső végpontjai.

Eljárás:

A:=1

F:=N

Ciklus

K:=INT((A+F)/2)

Ha A(K)<X akkor A:=K+1

Ha A(K)>X akkor F:=K-1


```

amíg A<=F és A(K)≠ X
Ciklus vége
VAN:=A<=F
Ha VAN akkor SORSZ:=K
Eljárás vége.

```

Megjegyzések:

- azért hívják logaritmikus keresésnek, mert a ciklus lépésszáma legfeljebb: $\log_2 N$
- sokkal hatékonyabb rendezett sorozatra, mint a lineáris keresés
- a hátul tesztelés ciklus – a Pascal nyelvénél megszokottól eltérően – igaz feltétel esetén tér vissza a ciklusba

Maximumkiválasztás

Sorozat legnagyobb elemének indexe.

```

Eljárás:
INDEX:=1
Ciklus I=2-től N-ig
    Ha A(INDEX)<A(I) akkor INDEX:=I
Ciklus vége
Eljárás vége.

```

Minimum-kiválasztásnál a relációjel megfordul.

Mennyi a legnagyobb érték:

```

Eljárás:
ÉRTÉK:=A(1)
Ciklus I=2-től N-ig
    Ha ÉRTÉK<A(I) akkor ÉRTÉK:=A(I)
Ciklus vége
Eljárás vége.

```

2.2. Egy sorozathoz egy sorozat hozzárendelése

Kiválogatás

Egy N elemű sorozat összes T tulajdonságú elemét kell meghatározni. A kiválogatott elemek sorszámait egy B() vektorban gyűjtjük.

```

Eljárás:
J:=0
Ciklus I=1-től N-ig
    Ha A(I) T tulajdonságú, akkor J:=J+1
                                     B(J):=I
Ciklus vége
Eljárás vége.

```

Sorozat elemeinek permutálása

Rendezések

Rendezési eljárás kiválasztásánál szempontok:

- tárigény
- végrehajtási idő
- összehasonlítások, mozgatások, cserék száma
- adott gépi környezet.

- Rendezés közvetlen kiválasztással

Módszer lényege: Rendezendő számok az A vektor elemei. Első menetben az A(1)-et összehasonlítjuk az összes elemmel és ha kisebbet találunk nála, akkor felcseréljük. Így az első menet végére a legkisebb elem lesz az első helyen. Ezután ezt ismételjük az A(2)-es elemmel, stb. N-1 menet után rendezett lesz a sorozat.

Eljárás:

```
Ciklus I=1-től N-1-ig
  Ciklus J=I+1-től N-ig
    Ha A(J)<A(I) akkor A:=A(J)
                                A(J):=A(I)
                                A(I):=A
  Ciklus vége
Ciklus vége
Eljárás vége.
```

- Rendezés minimum-kiválasztással

Módszer lényege: Felesleges cserék kiküszöbölése érdekében két segédváltozót vezetünk be (legkisebb elem értékének és indexének).

Eljárás:

```
Ciklus I=1-től N-1-ig
  INDEX:=I
  ÉRTÉK:=A(I)
  Ciklus J=I+1-től N-ig
    Ha A(J)<ÉRTÉK akkor ÉRTÉK:=A(J)
                                INDEX:=J
  Ciklus vége
  A(INDEX):=A(I)
  A(I):=ÉRTÉK
Ciklus vége
Eljárás vége.
```

- Buborékos rendezés:

Módszer lényege: Vektor végéről indulva minden elemet összehasonlítunk az előtte lévővel. Ha rossz a sorrend, akkor csere. Az első menet végére az első helyen a megfelelő elem áll. Ezt az elvet folytatjuk egyre kevesebb elemmel. (N-1 menet)

Eljárás:

```
Ciklus I=2-től N-ig
  Ciklus J=N-től I-ig -1-esével
    Ha  $A(J-1) > A(J)$  akkor  $A := A(J-1)$ 
       $A(J-1) := A(J)$ 
       $A(J) := A$ 
    Ciklus vége
  Ciklus vége
Eljárás vége.
```

- Egyszerű beillesztéses rendezés

Módszer lényege: Mintha kártyáinkat egyesével felvéve sorba raknánk. (N-1 menet)

Eljárás:

```
Ciklus J=2-től N-ig
  I:=J-1
  A:=A(J)
  Ciklus amíg  $I > 0$  és  $A < A(I)$ 
     $A(I+1) := A(I)$ 
    I:=I-1
  Ciklus vége
   $A(I+1) := A$ 
  Ciklus vége
Eljárás vége.
```

Metszetképzés

Általános feladat: Rendelkezésünkre áll egy N és egy M elemű halmaz az A() és a B() vektorban ábrázolva. Készítsük el a két halmaz metszetét a C() vektorba!

Eljárás:

```
CN:=0
Ciklus I=1-től N-ig
  J:=1
  Ciklus amíg  $J \leq M$  és  $A(I) \neq B(J)$ 
    J:=J+1
  Ciklus vége
  Ha  $J \leq M$  akkor  $CN := CN + 1$ 
     $C(CN) := A(I)$ 
  Ciklus vége
Eljárás vége.
```

Unióképzés

Általános feladat: Rendelkezésünkre áll egy N és egy M elemű halmaz az A() és a B() vektorban ábrázolva. Készítsük el a két halmaz egyesítését a C() vektorba!

Eljárás:

```
Ciklus I=1-től N-ig
  C(I):=A(I)
Ciklus vége
CN:=N
Ciklus J=1-től M-ig
  I:=1
  Ciklus amíg I<=N és A(I)<>B(J)
    I:=I+1
  Ciklus vége
  Ha I>N akkor CN:=CN+1
    C(CN):=B(J)
  Ciklus vége
Eljárás vége.
```

Összefuttatás

Általános feladat: Két rendezett sorozat uniója úgy, hogy a rendezettség megmaradjon.

Eljárás:

```
I:=1
J:=1
K:=0
Ciklus amíg I<=N és J<=M
  K:=K+1
  Elágazás
    A(I)<B(J) esetén C(K):=A(I)
      I:=I+1
    A(I)=B(J) esetén C(K):=A(I)
      I:=I+1
      J:=J+1
    A(I)>B(J) esetén C(K):=B(J)
      J:=J+1
  Elágazás vége
Ciklus vége
Ciklus amíg I<=N
  K:=K+1
  C(K):=A(I)
  I:=I+1
Ciklus vége
Ciklus amíg J<=M
  K:=K+1
  C(K):=B(J)
  J:=J+1
Ciklus vége
Eljárás vége.
```

Backtrack – Visszalépéses keresés

Feladat: Adott N sorozat, amelyek rendre $M(1), M(2), \dots$ elemszámúak. Ki kell választani mindegyikből egy-egy elemet úgy, hogy egyes sorozatokból való választások másokat befolyásolnak. A megoldást úgy kell elkészíteni, hogy ne kelljen az összes lehetőséget végignézni.

Először megpróbálunk az első sorozatból választani egy elemet, ezután a következőből, s ezt mindaddig csináljuk, amíg a választás lehetséges.

$X(I)$ jelöli az I . sorozat kiválasztott eleme sorszámát, ha még nem választottunk, akkor az értéke 0.

Ha nincs jó választás, akkor visszalépünk az előző sorozathoz, s megpróbálunk abból egy másik elemet választani. Ez az egész eljárás vagy úgy ér véget, hogy minden sorozatból sikerült választani, vagy pedig úgy, hogy a visszalépések sokasága után már az első sorozatból sem lehet újabb elemet választani.

Eljárás:

```
I:=1
Ciklus amíg I>=1 és I<=N
  Ha VAN_JÓ_ESET(I) akkor I:=I+1
  X(I):=0
  különben I:=I-1
Ciklus vége
VAN:=I>N
Eljárás vége.
```

Az I . sorozatból úgy választunk elemet, hogy próbálunk mindaddig új elemet venni, amíg egyáltalán van további elem, és az éppen vizsgált nem felel meg a feladatnak. Ha a keresgélés közben a sorozat elemei nem fogytak el, akkor az előző szintnek válaszolhatjuk azt, hogy sikeres volt a választás. Ha pedig az utolsó sem felelt meg, akkor azt, hogy vissza kell lépni az előző sorozathoz.

```
VAN_JÓ_ESET(I):
Ciklus
  X(I):=X(I)+1
  amíg X(I)<=M(I) és ROSSZ_ESET(I, X(I))
Ciklus vége
VAN_JÓ_ESET:=X(I)<=M(I)
Eljárás vége.
```

Rossz választásnak nevezzük azt, amelyet a korábbi választások közül valamelyik megakadályoz.

```
ROSSZ_ESET(I, X(I)):
J:=1
Ciklus amíg J<I és (J, X(J)) nem zárja ki (I, X(I))-t
  J:=J+1
Ciklus vége
ROSSZ_ESET:=J<I
Eljárás vége.
```

Példa: Készítsünk algoritmust, amely elhelyez egy sakktablán 8 vezért úgy, hogy egyik sem üti a másikat!

2.3. Programozási tételek rekurzív megvalósítása

Ciklusok megvalósítása rekurzióval

Elöltesztelő ciklus R eljárás(x): Ciklus amíg p(x) S eljárás(x) Ciklus vége Eljárás vége.	és	rekurzív változata: R eljárás(x). Ha p(x) akkor S eljárás(x) R eljárás(x) Elágazás vége Eljárás vége.
Hátultesztelő ciklus R eljárás(x): Ciklus S eljárás(x) amíg p(x) Ciklus vége Eljárás vége.	és	rekurzív változata: R eljárás(x): S eljárás(x) Ha p(x) akkor R eljárás(x) Eljárás vége.
Számlálós ciklus R eljárás: Ciklus I:=kezd-től vég-ig növ-vel S eljárás Ciklus vége Eljárás vége.	és	rekurzív változata: R eljárás: I:=kezd Q eljárás(I) Eljárás vége. Q eljárás(I): Ha I<=vég akkor S eljárás(I) I:=I+növ Q eljárás(I) Elágazás vége Eljárás vége.

Néhány programozási tétel rekurzív megvalósítása:

Összegzés:

S:=0
Ciklus I=1-től N-ig
 S:=S+A(I)
Ciklus vége
Eljárás vége.

Összegzés(N):

 Ha $N > 0$ akkor
 Összegzés:=Összegzés(N-1)+A(N)
 különben Összegzés:=0

Eljárás vége.

Eldöntés:

I:=1
Ciklus amíg $I \leq N$ és A(I) nem T tulajdonságú
 I:=I+1
Ciklus vége
VAN:= $I \leq N$
Eljárás vége

Eldöntés:

 I:=1
 Eld(I)
 VAN:= $I \leq N$
Eljárás vége.

Eld(I):

 Ha $I \leq N$ és A(I) nem T tulajdonságú akkor I:=I+1
 Eld(I)

Eljárás vége.

Kiválasztás:

I:=1
Ciklus amíg A(I) nem T tulajdonságú
 I:=I+1
Ciklus vége
SORSZ:=I
Eljárás vége

Kiválasztás:

 I:=1
 Kiv(I)
 SORSZ:=I
Eljárás vége.

Logaritmiikus keresés:

```
A:=1
F:=N
Ciklus
  K:=INT((A+F)/2)
  Ha A(K)<X akkor A:=K+1
  Ha A(K)>X akkor F:=K-1
amíg A<=F és A(K)≠X
Ciklus vége
VAN:=A<=F
Ha VAN akkor SORSZ:=K
Eljárás vége.
```

Logaritmiikus keresés:

```
E:=1
V:=N
Logker(E,V,K)
VAN:=E<=V
Ha VAN akkor SORSZ:=K
Eljárás vége.
```

Logker(E,V,K):

```
K:=INT((E+V)/2)
Ha A(K)<X akkor E:=K+1
Ha A(K)>X akkor V:=K-1
Ha E<=V és A(K)≠X akkor Logker(E,V,K)
Eljárás vége.
```

Kiválogatás:

```
J:=0
Ciklus I=1-től N-ig
  Ha A(I) T tulajdonságú, akkor J:=J+1
  B(J):=I
Ciklus vége
Eljárás vége.
```

Kiválogat:

```
J:=0
I:=1
Kiv(A(),I,J,B())
Eljárás vége.
```

Kiv(A(),I,J,B()):

```
Ha I<=N akkor Ha A(I) T tulajdonságú akkor J:=J+1
  B(J):=I
  I:=I+1
  Kiv(A(),I,J,B())
Elágazás vége.
Eljárás vége.
```

Rendezés közvetlen kiválasztással:

```
Ciklus I=1-től N-1-ig
  Ciklus J=I+1-től N-ig
    Ha  $A(J) < A(I)$  akkor csere( $A(I), A(J)$ )
  Ciklus vége
Ciklus vége
Eljárás vége.
```

```
Rendez:
  I:=1
  Rend1(I)
Eljárás vége.
```

```
Rend1(I):
  Ha  $I \leq N$  akkor J:=I+1
    Rend2(I, J)
    I:=I+1
  Rend1(I)
Eljárás vége.
```

```
Rend2(I, J):
  Ha  $J \leq N$  akkor Ha  $A(J) < A(I)$  akkor csere( $A(I), A(J)$ )
    J:=J+1
  Rend2(I, J)
  Elágazás vége
Eljárás vége.
```

2.4. Feladatok a programozási tételek gyakorlására.

1. Szorozzunk össze N db szomszédos egész számot A -tól kezdődően!
2. Egy horgászverseny adatait egy mátrixban tároljuk: $M(i,j)$ jelenti, hogy az i . horgász a j . halfajtából mennyit fogott. Írjunk programot, amely kiszámítja, hogy a horgászok összesen hányat fogtak az egyes fajtákból!
3. Egy horgászverseny adatait egy mátrixban tároljuk: $M(i,j)$ jelenti, hogy az i . horgász a j . halfajtából mennyit fogott. Írjunk programot, amely kiszámítja, hogy az egyes horgászok hány halat fogtak összesen!
4. Adott egy 12. osztály tanulójának nyilvántartása, valamint az aznapi dátum. Írjunk programot, amely megállapítja, hogy az adott osztály tanulói között van-e nagykorú!
5. Egy halgazdaság próbafogást végzett. Minden kifogott halról tároljuk a súlyát és a hosszát. Egy hal méretes, ha adott súlynál többet nyom, és ha adott hosszánál nagyobb. Írjunk programot, amely meghatározza, hogy van-e olyan hal, amely nem méretes!
6. Az előző feladat adatai alapján írjunk programot, amely meghatározza, hogy minden hal méretes-e!
7. Egy bűnügyi nyilvántartásban a zsebtolvajokról négy adatot tartanak nyilván: magasság, szemszín, hajszín, eddig letöltött büntetés.
 - Döntsük el, van-e két olyan zsebtolvaj, akiket ez a nyilvántartás nem különböztet meg!
 - Döntsük el van-e két olyan zsebtolvaj, akinek legalább két nyilvántartott adata megegyezik!
 - Döntsük el van-e két olyan zsebtolvaj, akinek két nyilvántartott adata pontosan megegyezik!
 - Döntsük el, van-e olyan zsebtolvaj, akit ez a négy adat a nyilvántartásban szereplő minden más személytől megkülönböztet!
8. Addig kötünk fogadásokat, amíg először nem veszítünk. Állapítsuk meg, hányszor kötöttünk fogadást. s összesen mennyit nyertünk!
9. Ismerjük egy labdarugó bajnokság hétvégi fordulójának párosítását, adjuk meg, hogy hol játszik a hazai csapat!
10. Írjunk programot, amely egy szabászat személyi nyilvántartásából kideríti az egyik dolgozó születési adatait!
11. A vasúti nyilvántartás tartalmazza a Savaria expresszre kiadott helyjegyeket. Írjunk programot, amely meghatároz egy szabad helyet a vonaton!
12. Egy házi telefonkönyv a nevek szerint rendezett. Keressük meg benne egy adott telefonszámhoz tartozó nevet!
13. Nyelvvizsgán a nyelvtani tesztek pontszámait ülési sorrendben jegyezték fel. Keressünk olyan vizsgázót, aki ugyanannyi pontot kapott, mint a szomszédja!
14. Határozzuk meg az $A(N)$ vektor azon pozitív elemeinek számát, amelyek közvetlenül egy negatív elem után állnak!
15. Adott a valós értékeket tartalmazó $X(N)$ vektor és az A pozitív szám. Állapítsuk meg, hogy a vektor elemei közül hány esik a nulla A sugarú környezetébe!

16. Állapítsuk meg, hogy a P érték szerepel-e az $A(N)$ vektorban, és ha igen, hány nála nagyobb elem előzi meg!
17. Adott az $A(N)$ valós számokból álló vektor. Adjuk meg a P és Z változóban a vektor pozitív, illetve nulla értékű elemeinek számát!
18. Állapítsuk meg, hány olyan érték van a növekvően rendezett $A(N)$ vektorban, ami legalább K -szor előfordul!
19. Az $A(N)$ vektor egy évfolyam vizsgajegyeit tartalmazza. Számoljuk meg, hogy az 1, 2, 3, 4, és 5-ös jegyekből külön-külön hány db van!
20. Állapítsuk meg, hogy az $A(N)$ vektorban hány elem van nagyság szerint a helyén (növekvő sorrendet tekintve)!
21. Az $A(M,N)$ mátrix egy évfolyam vizsgaeredményeit tartalmazza.
 - Állapítsuk meg, hányan buktak meg!
 - Állapítsuk meg a kitűnő tanulók számát!
22. Dobókockával 100-szor dobunk. Számoljuk meg, hogy hányszor dobtunk páros számot!
23. Adjuk meg az N elemű sorozatnak azt a legnagyobb elemét, amely nagyobb az előtte levő elemnél, de kisebb az őt követőnél!
24. Ismerjük N gyerek születési idejét. Iskolaköteles az, aki X év június 30. előtt született. Határozzuk meg a legfiatalabb iskolaköteles gyereket, s a nem iskolakötelesek közül a legidősebbet! Hány nap eltérés van köztük?
25. Egy műkorcsolyázó teljesítményét N bíró pontozza. Összpontszámát úgy számolják ki, hogy elhagyva a legkisebb és legnagyobb pontszámot, a maradék pontok átlagát képzik. Határozzuk meg a korcsolyázó teljesítményét!
26. Ismerjük egy osztály tagjainak tárgyankénti osztályzatát. $A(I,J)$ jelenti az I . tanuló J . tárgyból szerzett jegyét. Állapítsuk meg, hogy
 - tantárgyanként ki a legjobb;
 - melyik diák melyik tárgyból a legjobb;
 - ki a legjobb átlagú tanuló!
27. Adott egy $A(N)$ vektor. Elemei közül a negatívok helyére írjunk 0-t!
28. Adottak egy futóverseny eredményei, valamint az első osztályú szintetár. Kik érték el az első osztályú eredményt?
29. A BKV utasszámlálást végzett. Ismerjük a felmérés adatait. Zsúfolt az a busz, ahol a férőhelyek legalább 80%-a foglalt. Üres, ha 20%-nál kisebb a kihasználtság. Készítsünk programot a zsúfolt és az üres buszok kiválogatására!
30. Ismerjük egy étterem étlapját (levesek, készételek, frissensültek, saláták, stb.). Válogassuk ki az X forintnál olcsóbb frissensülteket!
31. Írja le a megszámlálás tételének algoritmusát! (Adott egy N elemű sorozat és egy $-$ a sorozat elemein értelmezett T tulajdonság. Feladat a T tulajdonsággal rendelkező elemek megszámlálása.)
32. Írja le az eldöntés tételének algoritmusát! (N elemű sorozat és egy a sorozaton értelmezett T tulajdonság. Van-e a sorozatnak legalább egy T tulajdonságú eleme?)

33. Írja le a kiválasztás tételének algoritmusát! (Adott egy N elemű sorozat, egy - a sorozat elemein értelmezett - T tulajdonság, és tudjuk, hogy a sorozatban van legalább egy T tulajdonságú elem. A feladat ezen elem sorszámának meghatározása.)
34. Írja le a maximum kiválasztás tételének algoritmusát! (Adott egy N elemű sorozat, a sorozat legnagyobb elemének indexét kell meghatározni.)

35. Egészítse ki a következő programrészleteket:

- a. Buborékos rendezés:

Eljárás:

Ciklus I=2-től N-ig

Ciklus J=.....

Ha A(J-1) .. A(J) **akkor** A:=A(J-1)

A(J-1) :=A(J)

A(J) :=A

Ciklus vége

Ciklus vége

Eljárás vége.

- b. Kiválogatás

(Egy N elemű sorozat összes T tulajdonságú elemét kell meghatározni. A kiválogatott elemek sorszámait egy B() vektorban gyűjtjük.)

Eljárás:

J:=0

Ciklus I=1-től N-ig

Ha A(I) T tulajdonságú, **akkor**

.....

Ciklus vége

Eljárás vége.

- c. Metszetképzés

(Általános feladat: Rendelkezésünkre áll egy N és egy M elemű halmaz az A() és a B() vektorban ábrázolva. Készítsük el a két halmaz metszetét a C() vektorba!)

Eljárás:

CN:=0

Ciklus I=1-től N-ig

J:=1

Ciklus amíg J<=M és

J:=J+1

Ciklus vége

Ha J<=M **akkor** CN:=CN+1

C(CN) :=A(I)

Ciklus vége

Eljárás vége.

d. Unióképzés

(Általános feladat: Rendelkezésünkre áll egy N és egy M elemű halmaz az A() és a B() vektorban ábrázolva. Készítsük el a két halmaz egyesítését a C() vektorba!)

Eljárás:

Ciklus I=1-től N-ig

 C(I) :=A(I)

Ciklus vége

CN:=N

Ciklus J=1-től M-ig

 I:=1

Ciklus amíg I<=N és

 I:=I+1

 Ciklus vége

Ha I>N **akkor** CN:=CN+1

 C(CN) :=B(J)

Ciklus vége

Eljárás vége.

3. Adattípusok megvalósítása.

A fejezet az ELTE TTK Informatikai Tanszéke által 1990-es években a hallgatók részére kiadott Mikrológia füzetek sorozat programjainak a felhasználásával készült.[4], [5], [7], [8], [9]

3.1. Algoritmikus típus-specifikáció

modul: a típus megvalósítás kerete

exportmodul: adattípus állandó ismérvei (mit és miként lehet egy adattal tenni, mitől tipikus)

megvalósítási modul: reprezentáció és implementáció megfogalmazása (az adat hogyan nézzen ki a memóriában, műveleteit hogyan kell elvégezni)

Exportmodul

Célja: a kívülről látható fogalmak felsorolása és felhasználási módjának definiálása.

A modul szintaxisa:

```
Exportmodul TípusNév (InputParaméterek) :  
    [most következnek az exportálandó fogalmak csoportosított felsorolása]  
Típus      [Ritkán van ilyenre szükség]  
    Tip1=TipDef1  
    Tip2=TipDef2  
    .....  
Konstans  [Ritkán van ilyenre szükség]  
    Kons1:KonSTip1  
    Kons2:KonSTip2  
    .....  
Változó   [Ritkán van ilyenre szükség, sőt kerülendő!]  
    Vált1:VáltTip1  
    Vált2:VáltTip2  
    .....  
Függvény Fv1 (FormParam) :FvTip1  
    Fv2 (FormParam) :FvTip2  
    .....  
Eljárás   Elj1 (FormParam)  
    Elj2 (FormParam)  
    .....  
Modul vége.
```

Megjegyzések:

- A csoportosításra nincs megkötés. (Tetszőleges sorrend, csoportszám.)
- Kerülni célszerű a változók, konstansok exportálását, mivel veszélyes, ha a felhasználó programnak lehetősége van az adatokkal közvetlen kapcsolatba kerülni. (Ha szükséges inkább külön erre a célra definiáljunk eljárásokat, függvényeket.)

Megvalósítási (vagy Reprezentációs-Implementációs) modul

Célja: az exportmodulban felsorolt fogalmak definiálása, részletezése. (adatok reprezentálása, tevékenységek implementálása)

A modul szintaxisa:

Modul TípusNév (InputParaméterek) :

Import

ModulNév1, vagy ModulNév1⇒fogalomfölsorolás

ModulNév2, vagy ModulNév2⇒fogalomfölsorolás

....

Reprezentáció

[a szokásos adatleíró rész, amely nemcsak az exportált adatfogalmakat, hanem más, csak saját maga számára szükséges lokális fogalmakat is tartalmaz]

Implementáció

[a szokásos tevékenységdefiníciók]

Inicializálás

[az adat megszületésekor milyen kezdőértékkel rendelkezzen]

Modul vége.

3.2. Sorozattípusok ábrázolása a memóriában:

1. Szekvenciális

- elemek memóriabeli sorrendje azonos a sorozatbeli sorrenddel
- az elemek a kezdőcímtől kezdve folytonosan töltik ki a memóriát
- elemeket könnyű elérni (elemek memóriacíme könnyen számolható)
- elemek helyét nehéz változtatni (beszúrás, törlés sok adatmozgatással jár)

2. Láncolt

Dinamikus:

- elemek memóriabeli sorrendje nem azonos a sorozatbeli sorrenddel
- elemeket kiegészítjük egy cím-mezővel, amely a sorozat következő elemének címét tartalmazza (ez a memóriánövekedés gazdaságtalan, ha az elemek mérete kicsi)
- elemek elérése nehéz (végig kell járni)
- elemek helyváltoztatása egyszerű, mindig a szükséges memóriaterület foglalt

Statikus:

- elemeket egy tömbben helyezük el
- valódi memóriacím helyett tömbindexet alkalmazunk a láncolásra
- rögzítjük a maximális méretet (nem dinamikus)

3. Blokkolt

- elemek memóriabeli sorrendje nem azonos a sorozatbeli sorrenddel
- elemek csoportjait blokkokba foglaljuk és a blokkokat láncoljuk
- szekvenciális és láncolt érényeit egyesíti:
 - blokkon belül az elemek címezhetők, könnyen elérhetők
 - blokkok között láncoltan lehet haladni
 - elem törlésekor csak a blokkon belül kell mozgatni az elemeket
 - elem beszúrásakor: ha elfér a blokkban, akkor csak ott mozgatjuk az elemeket
ha megtelt a blokk, akkor egy új blokkot fűzünk a láncba

3.3 A lista típuskonstrukció

A lista exportmodulja:

```
ExportModul Lista(Típus ElemTip):  
  Eljárás Üres(Változó l:Lista) [lista létrehozása]  
  Függvény Üres?(Konstans l:Lista):Logikai [üres-e]  
  Függvény ElemÉrték(Változó l:Lista):ElemTip [az aktuális elem értéke]  
  Eljárás ElemMódosít(Változó l:Lista, Konstans e:ElemTip)  
  [az aktuális elemet módosítja]  
  Eljárás Elsőre(Változó l:Lista) [az első legyen az aktuális elem]  
  Eljárás Következőre(Változó l:Lista) [a következő legyen az aktuális elem]  
  Eljárás BeszúrMögé(Változó l:Lista, Konstans e:ElemTip)  
  [beszúrás az aktuális elem mögé]  
  Eljárás BeszúrElejére(Változó l:Lista, Konstans e:ElemTip)  
  [beszúrás az első elem elé]  
  Eljárás BeszúrElé(Változó l:Lista, Konstans e:ElemTip)  
  [beszúrás az aktuális elem elé]  
  Eljárás Kihagy(Változó l:Lista) [az aktuális elem kihagyása]  
  Függvény UtolsóE?(Változó l:Lista):Logikai  
  [az aktuális elem az utolsó-e]  
  Függvény Elemszám(Konstans l:Lista):Egész [listaelemek száma]  
  Függvény Hibás?(Változó l:Lista):Logikai  
  [történt-e hiba a listára hivatkozás során]
```

Modul vége.

A lista megvalósítási moduljai

Lefoglal és Felszabadít eljárások felhasználásával

Láncolt ábrázolású lista (dinamikus)

```
Modul Lista(Típus ElemTip):  
  Reprezentáció  
    Típus  
      ListaElem=Rekord  
      (érték:ElemTip  
       köv:ListaElem'Mutató)  
    Változó  
      fej,akt:ListaElem'Mutató  
      hossz:Egész  
      hiba:Logikai  
  
  Implementáció  
  
    Eljárás Üres(Változó l:Lista):  
      fej:=sehova  
      akt:=sehova  
      hossz:=0
```

hiba:=Hamis
Eljárás vége.

Függvény Üres?(Konstans l:Lista):Logikai
Üres?:=hossz=0 [vagy fej=sehova]
Függvény vége.

Függvény ElemÉrték(Változó l:Lista):ElemTip
Ha akt≠sehova **akkor** ElemÉrték:=ListaElem(akt).érték
különben hiba:=Igaz
Függvény vége.

Eljárás ElemMódosít(Változó l:Lista, Konstans e:ElemTip):
Ha akt≠sehova **akkor** ListaElem(akt).érték:=e
különben hiba:=Igaz
Eljárás vége.

Eljárás Elsőre(Változó l:Lista):
akt:=fej
Eljárás vége.

Eljárás Következőre(Változó l:Lista):
Ha akt≠sehova **akkor** akt:=ListaElem(akt).köv
különben hiba:=Igaz
Eljárás vége.

Eljárás BeszúrMögé(Változó l:Lista, Konstans e:ElemTip):
Változó új:ListaElem'Mutató
Elágazás
fej=sehova **esetén** BeszúrElejére(l,e)
akt≠sehova **esetén**
Lefoglal(új,ListaElem(e,ListaElem(akt).köv))
hiba:=új=sehova
Ha nem hiba **akkor** ListaElem(akt).köv:=új
akt:=új
hossz:=hossz+1
Elágazás vége.
egyéb esetben hiba:=Igaz
Elágazás vége
Eljárás vége.

Eljárás BeszúrElejére(Változó l:Lista, Konstans e:ElemTip):
Változó új:ListaElem'Mutató
új:=fej
Lefoglal(fej,ListaElem(e,új))
hiba:=fej=sehova
Ha nem hiba **akkor** akt:=fej
hossz:=hossz+1
Eljárás vége.

Eljárás BeszúrElé(**Változó** l:Lista, **Konstans** e:ElemTip):
Változó új:ListaElem'Mutató

Elágazás

fej=sehova **esetén** BeszúrElejére(l,e)

akt≠sehova **esetén** Lefoglal(új,ListaElem(akt))

hiba:=új=sehova

Ha nem hiba **akkor**

ListaElem(akt).köv:=új

ListaElem(akt).érték:=e

akt:=új

hossz:=hossz+1

Elágazás vége

egyéb esetben hiba:=Igaz

Elágazás vége

Eljárás vége.

Eljárás Kihagy(**Változó** l:Lista):

Változó el:ListaElem'Mutató

Ha akt≠sehova **akkor**

Ha akt=fej **akkor** fej:=ListaElem(akt).köv

Felszabadít(akt)

akt:=fej

különben el:=fej

Ciklus amíg ListaElem(el).köv≠akt

el:=ListaElem(el).köv

Ciklus vége

ListaElem(el).köv:=ListaElem(akt).köv

Felszabadít(akt)

akt:=ListaElem(el).köv

Elágazás vége

hossz:=hossz-1

különben hiba:=Igaz

Elágazás vége

Eljárás vége.

Függvény UtolsóE?(**Változó** l:Lista):Logikai

Ha akt≠sehova **akkor** UtolsóE?:=ListaElem(akt).köv=sehova

különben hiba:=Igaz

Függvény vége.

Függvény Elemszám(**Konstans** l:Lista):Egész

Elemszám:=hossz

Függvény vége.

Függvény Hibás?(**Változó** l:Lista):Logikai

Hibás?:=hiba

hiba:=Hamis

Függvény vége.

Inicializálás

```
fej:=sehova
akt:=sehova
hossz:=0
hiba:=Hamis
```

Modul vége.

Láncolt ábrázolású lista (statikus)

- tömbön belüli láncolás (mutatók helyett tömbindex)
- 0 tömbindex = sehova
- lista elemszáma rögzített (egyszerűbb a memórialefoglalás ellenőrzése)
- szabad elem: - negatív indexhivatkozás
- láncolt ábrázolás

Felhasznált eljárások:

- Helyürítés: -1 értékkel feltölti a tömb minden elemének *köv* részét
- Helykiválasztás(új): keres egy -1 indexű elemet
- HelyFelszabadít(akt): t(akt).köv értékét -1-re állítja

Modul lista(**Típus** ElemTip, **Konstans** max: Egész):

Reprezentáció

```
Típus IndexTip=-1..Max
      ListaElem=Rekord
          (érték: ElemTip
           köv: IndexTip)
Változó [az alábbiak alkotják a Listát]
  t: Tömb(1..Max:ListaElem)
  fej, akt: IndexTip
  hossz: Egész
  hiba: Logikai
```

Implementáció

```
Eljárás Üres(Változó l:Lista):
  fej:=0
  akt:=0
  hossz:=0
  hiba:=Hamis
  Helyürítés
Eljárás vége.
```

```
Függvény Üres?(Konstans l:Lista):Logikai
  Üres?:=hossz=0 [vagy fej=0]
Függvény vége.
```

Függvény ElemÉrték(**Változó** l:Lista):ElemTip

Ha akt≠0 **akkor** ElemÉrték:=t(akt).érték
különben hiba:=Igaz

Függvény vége.

Eljárás ElemMódosít(**Változó** l:Lista, **Konstans** e:ElemTip):

Ha akt≠0 **akkor** t(akt).érték:=e
különben hiba:=Igaz

Eljárás vége.

Eljárás Elsőre(**Változó** l:Lista):

akt:=fej

Eljárás vége.

Eljárás Következőre(**Változó** l:Lista):

Ha akt≠0 **akkor** akt:=t(akt).köv
különben hiba:=Igaz

Eljárás vége.

Eljárás BeszúrMögé(**Változó** l:Lista, **Konstans** e:ElemTip):

Változó új:IndexTip

Elágazás

fej=0 **esetén**

fej:=1

akt:=fej

hossz:=1

t(akt).érték:=e

t(akt).köv:=0

akt≠0 és hossz<Max **esetén**

Helykiválasztás(új)

t(új).érték:=e

t(új).köv:=t(akt).köv

t(akt).köv:=új

akt:=új

hossz:=hossz+1

egyéb esetben hiba:=Igaz

Elágazás vége

Eljárás vége.

Eljárás BeszúrElejére(**Változó** l:Lista, **Konstans** e:ElemTip):

Változó új:IndexTip

Ha hossz<Max **akkor** Helykiválasztás(új)

t(új).érték:=e

t(új).köv:=fej

fej:=új

akt:=fej

hossz:=hossz+1

különben hiba:=Igaz

Elágazás vége

Eljárás vége.

Eljárás BeszúrElé(**Változó** l:Lista, **Konstans** e:ElemTip):
Változó új:IndexTip

Elágazás

fej=0 **esetén** BeszúrElejére(l,e)

akt≠0 és hossz<Max **esetén**

Helykiválasztás(új)

t(új).köv:=t(akt).köv

t(új).érték:=t(akt).érték

t(akt).köv:=új

t(akt).érték:=e

akt:=új

hossz:=hossz+1

egyéb esetben hiba:=Igaz

Elágazás vége

Eljárás vége.

Eljárás Kihagy(**Változó** l:Lista):

Változó el:Indextip

Ha akt≠0 **akkor**

Ha akt=fej **akkor** fej:=t(akt).köv

HelyFelszabadít(akt)

akt:=fej

különben el:=fej

Ciklus amíg t(el).köv≠akt

el:=t(el).köv

Ciklus vége

t(el).köv:=t(akt).köv

HelyFelszabadít(akt)

akt:=t(el).köv

Elágazás vége

hossz:=hossz-1

különben hiba:=Igaz

Elágazás vége

Eljárás vége.

Függvény UtolsóE?(**Változó** l:Lista):Logikai

Ha akt≠0 **akkor** UtolsóE?:=t(akt).köv=0

különben hiba:=Igaz

Függvény vége.

Függvény Elemszám(**Konstans** l:Lista):Egész

Elemszám:=hossz

Függvény vége.

Függvény Hibás?(**Változó** l:Lista):Logikai

Hibás?:=hiba

hiba:=Hamis

Függvény vége.

Inicializálás

```
fej:=0
akt:=0
hossz:=0
hiba:=Hamis
```

Modul vége.

Folytonos (szekvenciális) ábrázolású lista

- tömbön belüli ábrázolás
- elemek fizikai sorrendje megegyezik a logikai sorrenddel
- módosításkor (beszúrás, törlés) a lista elemeit tologatjuk

Modul lista(**Típus** ElemTip, **Konstans** MaxSzám: Egész):

Reprezentáció

```
Típus ListaElemek=Tömb(1..MaxSzám: ElemTip)
Változó [az alábbiak alkotják a Listát]
  le: ListaElemek
  akt, hossz: 0..MaxSzám+1
  hiba: Logikai
```

Implementáció

```
Eljárás Üres(Változó l:Lista):
  akt:=0
  hossz:=0
  hiba:=Hamis
Eljárás vége.
```

```
Függvény Üres?(Konstans l:Lista):Logikai
  Üres?:=hossz=0
Függvény vége.
```

```
Függvény ElemÉrték(Változó l:Lista):ElemTip
  Ha akt∈[1..hossz] akkor ElemÉrték:=le(akt)
  különben hiba:=Igaz
Függvény vége.
```

```
Eljárás ElemMódosít(Változó l:Lista, Konstans e:ElemTip):
  Ha akt∈[1..hossz] akkor le(akt):=e
  különben hiba:=Igaz
Eljárás vége.
```

Eljárás Elsőre (**Változó** l:Lista):
 akt:=1
Eljárás vége.

Eljárás Következőre (**Változó** l:Lista):
 Ha akt∈[1..hossz] **akkor** akt:=akt+1
 különben hiba:=Igaz
Eljárás vége.

Eljárás BeszúrMögé (**Változó** l:Lista, **Konstans** e:ElemTip):
 Változó i: 1..MaxSzám
 Elágazás
 hossz=0 **esetén**
 le(1):=e
 akt:=1
 hossz:=1
 akt=hossz és hossz<MaxSzám **esetén**
 akt:=akt+1
 le(akt):=e
 hossz:=hossz+1
 akt∈[1..hossz-1] és hossz<MaxSzám **esetén**
 akt:=akt+1
 Ciklus i=hossz-tól akt-ig -1-esével
 le(i+1):=le(i)
 Ciklus vége
 le(akt):=e
 hossz:=hossz+1
 egyéb esetben hiba:=Igaz
 Elágazás vége
Eljárás vége.

Eljárás BeszúrElejére (**Változó** l:Lista, **Konstans** e:ElemTip):
 Változó i: 1..MaxSzám
 Ha hossz<MaxSzám **akkor** **Ciklus** i=hossz-tól 1-ig -1-esével
 le(i+1):=le(i)
 Ciklus vége
 le(1):=e
 akt:=1
 hossz:=hossz+1
 különben hiba:=Igaz
 Elágazás vége
Eljárás vége.

Eljárás BeszúrElé (**Változó** l:Lista, **Konstans** e:ElemTip):
 Változó i: 1..MaxSzám
 Elágazás
 hossz=0 **esetén** le(1):=e
 akt:=1
 hossz:=1
 akt=hossz+1 és hossz<MaxSzám **esetén** le(akt):=e
 akt:=akt+1


```

                                hossz:=hossz+1
    akt∈[1..hossz] és hossz<MaxSzám esetén
        Ciklus i:=hossz-tól akt-ig -1-esével
            le(i+1):=le(i)
        Ciklus vége
        le(akt):=e
        hossz:=hossz+1
        akt:=akt+1
    egyéb esetben hiba:=Igaz
Elágazás vége
Eljárás vége.

```

```

Eljárás Kihagy(Változó l:Lista):
    Változó i: 1..MaxSzám
    Ha akt∈[1..hossz] akkor Ciklus i=akt-tól hossz-1-ig
        le(i):=le(i+1)
    Ciklus vége
    hossz:=hossz-1
    különben hiba:=Igaz

    Elágazás vége
Eljárás vége.

```

```

Függvény UtolsóE?(Változó l:Lista):Logikai
    Ha akt∈[1..hossz] akkor UtolsóE?:=akt=hossz
    különben hiba:=Igaz

Függvény vége.

```

```

Függvény Elemszám(Konstans l:Lista):Egész
    Elemszám:=hossz
Függvény vége.

```

```

Függvény Hibás?(Változó l:Lista):Logikai
    Hibás?:=hiba
    hiba:=Hamis
Függvény vége.

```

Inicializálás

```

    akt:=0
    hossz:=0
    hiba:=Hamis

```

Modul vége.

Kiválogatás tétel újrafogalmazása listára

Válogassuk külön azokat az elemeket a listának, amelyek eleget tesznek egy T tulajdonságnak.

Változó l: Lista(ElemTip) [bemenő paraméter]

tl: Lista(ElemTip) [kimenő paraméter]
Függvény Ttul(Konstans e:ElemTip):Logikai
az e adat T tulajdonságú
Függvény vége.

Kiválogatás(Konstans l:Lista, Változó tl:Lista)
Üres(tl)
Elsőre(l)
Ciklus amíg nem UtolsóE?(l)
Ha Ttul(ElemÉrték(l)) **akkor** BeszúrMögé(tl,ElemÉrték(l))
Következőre(l)
Ciklus vége
Ha Ttul(ElemÉrték(l)) **akkor** BeszúrMögé(tl,ElemÉrték(l))
Eljárás vége.

Beszúrásos rendezés

Típus Rendezendő=Tömb(1..Max: ElemTip)

Rendezés(Konstans t:Rendezendő, Változó L:Lista(ElemTip))
Üres(l)
Ciklus i=1-től N-ig
Elsőre(l)
e:=t(i)
Ciklus amíg nem UtolsóE?(l) és e>ElemÉrték(l)
Következőre(l)
Ciklus vége
Ha e>ElemÉrték(l) **akkor** BeszúrMögé(l,e)
különben BeszúrElé(l,e)
Ciklus vége
Eljárás vége

3.4. A verem típuskonstrukció

A sorozat egyik végével tudunk műveletet végezni (új elem "tetejére", kivenni elemet a "tetejéről")

LIFO lista: Last In First Out

alapvető műveletei: -ráhelyezés, leemelés

A verem exportmodulja

```
ExportModul Verem(Típus ElemTip) :  
  Eljárás Üres (Változó v:Verem) [veremlétrehozás a memóriában]  
  Függvény Üres? (Konstans v:Verem) :Logikai [üres-e a verem]  
  Függvény Tele? (Konstans v:Verem) :Logikai [a verem megtelt-e]  
  Függvény Tető (Változó v:Verem) :ElemTip [a verem tetőelemének értéke]  
  Eljárás Verembe (Változó v:Verem, Konstans e:ElemTip) [verembetétel]  
  
  Eljárás Veremből (Változó v:Verem, e:ElemÉrték) [tetőelem kivétele]  
  Függvény Hibás? (Változó v:Verem) :Logikai [történt-e hiba a veremre hivatkozásnál]
```

Modul vége.

A verem megvalósítási moduljai

Láncolt ábrázolású verem

Teteje: a verem tetején lévő elem címe (mutató).

Minden elemnek tartalmaznia kell az alatta lévő elem címét.

(Hasonlít a listához, csak más műveletek.)

Modul Verem(**Típus** ElemTip) :

Reprezentáció

Típus

```
VeremElem=Rekord(érték:ElemTip,  
                 alatta:VeremElem'Mutató)
```

Változó

```
teteje: VeremElem'Mutató  
hiba:Logikai
```

Implementáció

```
Eljárás Üres (Változó v:Verem)
```

```
  teteje:=sehova
```

```
  hiba:=Hamis
```

```
Eljárás vége.
```

```
Függvény Üres? (Konstans v:Verem) :Logikai
```

```
  Üres?:=teteje=sehova
```

Függvény vége.

Függvény Tele?(**Konstans** v:Verem):Logikai

Változó sv:VeremElem'Mutató

Lefoglal(sv)

Ha sv=sehova **akkor** Tele?:=Igaz

különben Tele?:=Hamis

Felszabadít(sv)

Elágazás vége

Függvény vége.

Függvény Tető(**Változó** v:Verem):ElemTip

Ha teteje≠sehova **akkor** Tető:=VeremElem(teteje).érték

különben hiba:=Igaz

Elágazás vége

Függvény vége.

Eljárás Verembe(**Változó** v:Verem, **Konstans** e:ElemTip)

Változó új:VeremElem'Mutató

Lefoglal(új)

Ha új≠sehova **akkor** VeremElem(új):=VeremElem(e,teteje)

teteje:=új

különben hiba:=Igaz

Elágazás vége

Eljárás vége.

Eljárás Veremből(**Változó** v:Verem,e:ElemÉrték)

Változó új:teteje: VeremElem'Mutató

Ha teteje≠sehova **akkor** e:=VeremElem(teteje).érték

új:teteje:=VeremElem(teteje).alatta

Felszabadít(teteje)

teteje:=új:teteje

különben hiba:=Igaz

Elágazás vége

Eljárás vége.

Függvény Hibás?(**Változó** v:Verem):Logikai

Hibás?:=hiba

hiba:=Hamis

Függvény vége.

Inicializálás

teteje:=sehova

hiba:=hamis

Modul vége.

Folytonos ábrázolású verem

Egy tömbben ábrázoljuk.

Modul Verem(**Típus** ElemTip):

Reprezentáció

Konstans MaxMélység:Egész(??)

Típus

VeremElemek=Tömb(1..MaxMélység:ElemTip)

Változó

ve:VeremElemek

teteje:0..MaxMélység

hiba:Logikai

Implementáció

Eljárás Üres(**Változó** v:Verem)

teteje:=0

hiba:=Hamis

Eljárás vége.

Függvény Üres?(**Konstans** v:Verem):Logikai

Üres?:=teteje=0

Függvény vége.

Függvény Tele?(**Konstans** v:Verem):Logikai

Tele?:=teteje=MaxMélység

Függvény vége.

Függvény Tető(**Változó** v:Verem):ElemTip

Ha teteje≠0 **akkor** Tető:=ve(teteje)

különben hiba:=Igaz

Elágazás vége

Függvény vége.

Eljárás Verembe(**Változó** v:Verem, **Konstans** e:ElemTip)

Ha teteje=MaxMélység **akkor** hiba:=Igaz

különben teteje:=teteje+1

ve(teteje):=e

Elágazás vége

Eljárás vége.

Eljárás Veremből(**Változó** v:Verem, e:ElemÉrték)

Ha teteje≠0 **akkor** e:=ve(teteje)

teteje:=teteje-1

különben hiba:=Igaz

Elágazás vége

Eljárás vége.

```
Függvény Hibás?(Változó v:Verem):Logikai  
    Hibás?:=hiba  
    hiba:=Hamis  
Függvény vége.
```

Inicializálás

```
teteje:=0  
hiba:=hamis
```

Modul vége.

3.5. A sor típuskonstrukció

Új elemet a végére tehetjük, elvenni elemet az elejéről lehet.

FIFO lista: First In First Out

A sor exportmodulja

ExportModul Sor(**Típus** ElemTip):

Eljárás Üres(Változó s: Sor)	[sorrétrehozás a memóriában]
Függvény Üres?(Konstans s: Sor): Logikai	[üres-e a sor]
Függvény Tele?(Konstans s: Sor): Logikai	[a sor megtelt-e]
Függvény Első(Változó s: Sor): ElemTip	[a sor első elemének értéke]
Eljárás Sorba(Változó s: Sor, Konstans e: ElemTip)	[sorbatétel]
Eljárás Sorból(Változó s: Sor, e: ElemÉrték)	[az első elem kivétele]
Függvény Hibás?(Változó s: Sor): Logikai	[történt-e hiba a sorra hivatkozásnál]

Modul vége.

A sor megvalósítási moduljai

Láncolt ábrázolású sor

Modul Sor(**Típus** ElemTip):

Reprezentáció

Típus

SorElem=Rekord(érték: ElemTip,
köv: SorElem' Mutató)

Változó

eleje, vége: SorElem' Mutató
hiba: Logikai

Implementáció

Eljárás Üres(**Változó** s: Sor)

eleje:=sehova
vége:=sehova
hiba:=Hamis

Eljárás vége.

Függvény Üres?(**Konstans** s: Sor): Logikai

Üres?:=eleje=sehova

Függvény vége.

Függvény Tele?(**Konstans** s: Sor): Logikai

Változó ss: SorElem' Mutató

Lefoglal(ss)

Ha ss=sehova **akkor** Tele?:=Igaz

különben Tele?:=Hamis
Felszabadít(ss)

Függvény vége.

Függvény Első(Változó s: Sor):ElemTip
Ha eleje≠sehova **akkor** Első:=SorElem(eleje).érték
különben hiba:=Igaz

Függvény vége.

Eljárás Sorba(Változó s: Sor, Konstans e:ElemTip)
Változó új: SorElem' Mutató
Lefoglal(új)
Ha új≠sehova **akkor** SorElem(új) := SorElem(e, sehova)
Ha vége≠sehova **akkor** SorElem(vége).köv:=új
különben eleje:=új
Elágazás vége
vége:=új
különben hiba:=Igaz

Elágazás vége

Eljárás vége.

Eljárás Sorból(Változó s: Sor, e:ElemÉrték)
Változó újeleje: SorElem' Mutató
Ha eleje≠sehova **akkor** e:=SorElem(eleje).érték
újeleje:=SorElem(eleje).köv
Felszabadít(eleje)
eleje:=újeleje
különben hiba :=Igaz

Eljárás vége.

Függvény Hibás?(Változó s: Sor):Logikai
Hibás?=hiba
hiba:=Hamis
Függvény vége.

Inicializálás

eleje:=sehova
vége:=sehova
hiba:=Hamis

Modul vége.

Folytonos ábrázolású sor (ciklikus)

Megvalósítási ötletek:

- sor eleje: tömb 1. eleme
új elem felvételekor a tömb vége növekedjen
Sorból művelet sok mozgatással jár.

- Sorból műveletnél az eleje indexe növekszik
a vége betelt, de az elején maradt hely ...
- a tömböt ciklikussá tesszük, így az utolsó elem után az első következik
Üresség vizsgálata: hossz-számláló bevezetése

Modul Sor(**Típus** ElemTip):

Reprezentáció

Konstans MaxHossz: Egész

Típus

SorElemek=Tömb(1..MaxHossz:ElemTip)

Változó

se: SorElemek

eleje, vége: 0..MaxHossz

hiba:Logikai

Implementáció

Eljárás Üres(**Változó** s: Sor)

eleje:=1

vége:=1

hossz:=0

hiba:=Hamis

Eljárás vége.

Függvény Üres?(**Konstans** s: Sor): Logikai

Üres?:=hossz=0

Függvény vége.

Függvény Tele?(**Konstans** s: Sor): Logikai

Tele?:=hossz=MaxHossz

Függvény vége.

Függvény Első(**Változó** s: Sor): ElemTip

Ha hossz≠0 **akkor** Első:=se(eleje)

különben hiba:=Igaz

Függvény vége.

Eljárás Sorba(**Változó** s: Sor, **Konstans** e: ElemTip)

Ha hossz<MaxHossz **akkor** se(vége):=e

hossz:=hossz+1

vége:=(vége Mod Maxhossz)+1

különben hiba:=Igaz

Eljárás vége.

Eljárás Sorból(**Változó** s: Sor, e: ElemÉrték)

Ha hossz>0 **akkor** e:=se(eleje)

hossz:=hossz-1

```
        eleje:=(eleje Mod Maxhossz)+1
különben hiba :=Igaz
```

Eljárás vége.

Függvény Hibás?(Változó s: Sor):Logikai

```
Hibás?=hiba
```

```
hiba:=Hamis
```

Függvény vége.

Inicializálás

```
eleje:=1
```

```
vége:=1
```

```
hossz:=0
```

```
hiba:=Hamis
```

Modul vége.

3.6. A prioritási sor típuskonstrukció

Prioritás: egy „rendezése” az elemeknek.

A prioritási sor exportmodulja

```
ExportModul PrSor(Típus ElemTip: Típus Prioritás):  
  Eljárás Üres(Változó s:PrSor) [sorrétrehozás a memóriában]  
  Függvény Üres?(Konstans s:PrSor):Logikai [üres-e a sor]  
  Függvény Tele?(Konstans s:PrSor):Logikai [a sor megtelt-e]  
  Függvény ElsőElem(Változó s:PrSor):ElemTip  
  [a sor első elemének értéke]  
  Függvény ElsőPrioritás(Változó s:PrSor):Prioritás  
  [a sor első elemének prioritási értéke]  
  Eljárás Sorba(Változó s:PrSor, Konstans e:ElemTip, p:Prioritás)  
  [sorbatétel]  
  Eljárás Sorból(Változó s:PrSor, e:ElemTip, p:Prioritás)  
  [az első elem kivétele prioritással együtt]  
  Függvény Hibás?(Változó s:PrSor):Logikai  
  [történt-e hiba a sorra hivatkozásnál]
```

Modul vége.

A prioritási sor megvalósítási moduljai

Elejéről vesszünk ki és a prioritásának megfelelő helyre tesszük az újat.

Kétféle sorrendű ábrázolás:

- elemek sorrendje a prioritás szerinti sorrend
- elemek sorrendje a sorba bekerülési sorrend

Jelölés:

ciklikus növelés	$x:=x\oplus 1$	$x:=(x \text{ Mod } \text{MaxHossz}) + 1$
ciklikus csökkentés	$x:=x\ominus 1$	ha $x>1$ esetén $x:=x-1$ ha $x=1$ esetén $x:=\text{MaxHossz}$

Szekvenciális ábrázolás

Prioritás szerint rendezett ábrázolás

```
Modul PrSor(Típus ElemTip, Prioritás):
```

Reprezentáció

```
Konstans MaxHossz: Egész
```

```
Típus
```

```
  PrElemTip= Rekord(elem: ElemTip  
                    pr: Prioritás)
```

```
  SorElemek=Tömb(1..MaxHossz:PrElemTip)
```

Változó

se: SorElemek
eleje, vége: 0..MaxHossz
hiba:Logikai

Implementáció

Eljárás Üres(Változó s:PrSor)

eleje:=1
vége:=1
hossz:=0
hiba:=Hamis

Eljárás vége.

Függvény Üres?(Konstans s:PrSor):Logikai

Üres?:=hossz=0

Függvény vége.

Függvény Tele?(Konstans s:PrSor):Logikai

Tele?:=hossz=MaxHossz

Függvény vége.

Függvény ElsőElem(Változó s:PrSor):ElemTip

Ha hossz≠0 **akkor** ElsőElem:=se(eleje).elem
különben hiba:=Igaz

Függvény vége.

Függvény ElsőPrioritás(Változó s:PrSor):Prioritás

Ha hossz≠0 **akkor** ElsőPrioritás:=se(eleje).pr
különben hiba:=Igaz

Függvény vége.

Eljárás Sorba(Változó s:PrSor, Konstans e:ElemTip, p:Prioritás)

Ha hossz<MaxHossz
akkor **Ha** hossz=0 **akkor** hely:=eleje
különben Helykeresés(eleje, vége, hely, e, p)
Eltolás(hely, vége)

Elágazás vége

se(hely).elem:=e
se(hely).pr:=p
vége:=vége⊕1
hossz:=hossz+1

különben hiba:=Igaz

Eljárás vége.

Eljárás Helykeresés(Konstans eleje, vége: 0..MaxHossz; Változó hely: 0..MaxHossz; Konstans e:ElemTip, p:Prioritás):

hely:=eleje
Ciklus amíg hely≠vége⊕1 és se(hely).pr≥p
hely:=hely⊕1

Ciklus vége
Eljárás vége.

Eljárás Eltolás(**Konstans** hely, vége: 0..MaxHossz):

i:=vége

Ciklus amíg i≠hely⊕1

se(i⊕1):=se(i)

i:=i⊕1

Ciklus vége

Eljárás vége.

Eljárás Sorból(**Változó** s:PrSor, e:ElemTip, p:Prioritás):

Ha hossz>0 **akkor** p:=se(eleje).pr

e:=se(eleje).elem

eleje:=eleje⊕1

hossz:=hossz-1

különben hiba :=Igaz

Eljárás vége.

Függvény Hibás?(**Változó** s:PrSor):Logikai

Hibás?=hiba

hiba:=Hamis

Függvény vége.

Inicializálás

eleje:=1

vége:=1

hossz:=0

hiba:=Hamis

Modul vége.

3.7. Feladatok az adattípusok megvalósításához

Egészítse ki a következő algoritmusokat!

1. Dinamikusan láncolt ábrázolású lista:

Típus

ListaElem=Rekord
(érték:ElemTip
köv:ListaElem'Mutató)

Változó

fej,akt:ListaElem'Mutató
hossz:Egész
hiba:Logikai

Függvény ElemÉrték(Változó l:Lista):ElemTip

Ha akt≠sehova akkor ElemÉrték:=
különben hiba:=Igaz

Függvény vége.

2. Szekvenciálisan ábrázolt verem:

Konstans MaxMélység:Egész(??)

Típus

VeremElemek=Tömb(1..MaxMélység:ElemTip)

Változó

ve:VeremElemek
teteje:0..MaxMélység
hiba:Logikai

Függvény Üres?(Konstans v:Verem):Logikai

.....

Függvény vége.

3. Láncolt ábrázolású sor

Modul Sor(Típus ElemTip):

Reprezentáció

Típus

SorElem=Rekord(érték:ElemTip,
köv:SortElem'Mutató)

Változó

eleje, vége: SortElem'Mutató
hiba:Logikai

Függvény Első(Változó s:Sort):ElemTip

Ha eleje≠sehova akkor Első:=
különben hiba:=Igaz

Függvény vége.

4. Folytonos (szekvenciális) ábrázolású lista

Modul lista(Típus ElemTip, Konstans MaxSzám: Egész):

Reprezentáció**Típus** ListaElemek=Tömb(1..MaxSzám: ElemTip)**Változó**

le: ListaElemek
 akt, hossz: 0..MaxSzám+1
 hiba: Logikai

Eljárás Következőre (**Változó** l:Lista):

Ha akt∈[1..hossz] **akkor** akt:=.....
különben hiba:=Igaz

Eljárás vége.5. Láncolt ábrázolású lista (dinamikus)**Modul** Lista(**Típus** ElemTip):**Reprezentáció****Típus**

ListaElem=Rekord
 (érték:ElemTip
 köv:ListaElem'Mutató)

Változó

fej, akt:ListaElem'Mutató
 hossz:Egész
 hiba:Logikai

Eljárás ElemMódosít (**Változó** l:Lista, **Konstans** e:ElemTip):

Ha akt≠sehova **akkor**:=e
különben hiba:=Igaz

Eljárás vége.6. Folytonos ábrázolású verem**Modul** Verem(**Típus** ElemTip):**Reprezentáció****Konstans** MaxMélység:Egész(??)**Típus**

VeremElemek=Tömb(1..MaxMélység:ElemTip)

Változó

ve:VeremElemek
 teteje:0..MaxMélység
 hiba:Logikai

Függvény Tele? (**Konstans** v:Verem):Logikai

Tele?:=teteje=.....

Függvény vége.**Függvény** Tető (**Változó** v:Verem):ElemTip

Ha teteje≠0 **akkor** Tető:=.....)
különben hiba:=Igaz

Elágazás vége
Függvény vége.

7. Dinamikusan láncolt ábrázolású lista:

Típus

ListaElem=Rekord
(érték:ElemTip
köv:ListaElem'Mutató)

Változó

fej,akt:ListaElem'Mutató
hossz:Egész
hiba:Logikai

Függvény Következőre (**Változó** l:Lista):

Ha akt≠sehova **akkor** akt:=
különben hiba:=Igaz

Függvény vége.

8. Szekvenciálisan ábrázolt verem:

Konstans MaxMélység:Egész(??)

Típus

VeremElemek=Tömb(1..MaxMélység:ElemTip)

Változó

ve:VeremElemek
teteje:0..MaxMélység
hiba:Logikai

Eljárás Verembe (**Változó** v:Verem, **Konstans** e:ElemTip)

Változó új:VeremElem'Mutató

Lefoglal(új)

Ha új≠sehova **akkor** VeremElem(új):=VeremElem(e,teteje)
teteje:=

különben hiba:=Igaz

9. Folytonos (szekvenciális) ábrázolású lista

Modul lista (**Típus** ElemTip, **Konstans** MaxSzám: Egész):

Reprezentáció

Típus ListaElemek=Tömb(1..MaxSzám: ElemTip)

Változó

le: ListaElemek
akt, hossz: 0..MaxSzám+1
hiba: Logikai

Eljárás ElemÉrték (**Változó** l:Lista): ElemTip

Ha akt∈[1..hossz] **akkor** ElemÉrték:=.....
különben hiba:=Igaz

Eljárás vége.

10. Láncolt ábrázolású sor

Modul Sor (**Típus** ElemTip):

Reprezentáció

Típus

SorElem=Rekord(érték:ElemTip,
köv: SorElem'Mutató)

Változó

eleje, vége: SorElem'Mutató
hiba:Logikai

Eljárás Sorba(**Változó** s: Sor, **Konstans** e: ElemTip)

Változó új: SorElem'Mutató

Lefoglal(új)

Ha új≠sehova **akkor** SorElem(új) := SorElem(e, sehova)

Ha vége≠sehova **akkor** SorElem(vége).köv:=...
különben eleje:=...

Elágazás vége

vége:=új

különben hiba:=Igaz

Elágazás vége

Eljárás vége.

A fenti feladatok kipontozott részei megtalálhatóak a 3. fejezet megfelelő részeiben, ezért nincs külön részletezve a megoldás.

4. KÉRDÉSEK ÉS MEGOLDÁSOK AZ ELMÉLETI RÉSZ ANYAGÁBÓL:

1./ Adott az alábbi nyelvtan:

$G := \langle \{1, 0\}, \{S, A, B\}, S, P \rangle$, ahol $P: S \rightarrow 1A \mid 0B \mid 0 \mid \varepsilon \quad A \rightarrow 0 \mid 1B$

$B \rightarrow 1 \mid 0A \mid 1B$

Milyen típusú ez a nyelvtan?

Reguláris (vagy 3. Típusú) (2 pont)

Szerkesszen olyan automatát, mely felismeri az ezzel a nyelvtannal generált nyelvet!

Mi az automata kezdőállapota?

S (1 pont)

Mi(k) a végállapot(ok)?

S, E (1 pont)

Adja meg az átmenetfüggvényt táblázattal!

δ	0	1
S	{B,E}	{A}
A	{E}	{B}
B	{A}	{B,E}
E	\emptyset	\emptyset

(4 pont: a cellatartalmakért, ha teljesen jók, 0,5-0,5 pont.)

Megjegyzés: A táblázat sorainak és oszlopainak sorrendje nem számít, csak a megfelelő cellában a megfelelő érték kell legyen. 'E' állapot a végállapot, ha itt más nevet használ, de azt következetesen, az jó.

2./ Adott az alábbi súlymátrix:

	v_1	v_2	v_3
v_1	∞	1	4
v_2	∞	∞	2
v_3	3	∞	∞

Készítse el a legrövidebb utak mátrixát a Warshall algoritmus segítségével!

Töltse ki a lépésenként kapott mátrixokat:

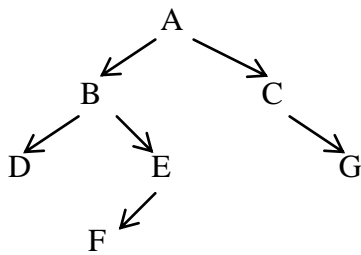
	v_1	v_2	v_3
v_1	∞	1	4
v_2	∞	∞	2
v_3	3	4	7

	v_1	v_2	v_3
v_1	∞	1	3
v_2	∞	∞	2
v_3	3	4	6

	v_1	v_2	v_3
v_1	6	1	3
v_2	5	6	2
v_3	3	4	6

(1-2. mátrix 2 pont, 3. mátrix 3 pont +1pont, az alábbi bontásban: minden olyan érték, amit kiemeltem (vastag, aláhúzott) ér 1 pontot (ezek térnek el az előző mátrixtól) és a maradék egyet.)

- 3./ - Nevezzen meg három olyan algoritmust, amelyek vermet használ működése során!
 - Írjon egy példát rekurzív algoritmusra!
 - Milyen bejárési (keresési) algoritmusokat ismer általános irányított gráfok bejárására?
 - Milyen bejárési (keresési) algoritmusokat ismer bináris fák bejárására?
 - Az utóbbiak közül az egyik (szabadon választott) algoritmussal járja be az alábbi fát (a bejárt csúcsok sorrendjét és a választott algoritmust írja fel!):



- 4./ Adott az alábbi rendezetlen sorozat:
2, 3, 12, 10, 23, 25, 15, 8
 Rendezze ezt a sorozatot
 a./ QUICK sort algoritmussal (írja fel, hogy hogyan alakul a sorozat a rendezés egyes lépései során)
 b./ rendezőfával (rajzolja fel a rendezőfát és mondja meg, hogy milyen bejárással kell ezt a fát bejárni ahhoz, hogy rendezett sorozathoz jussunk.)
- 5./ Adott az alábbi infix kifejezés: **$(2 * (5 + 2)) / 2 + 5 * 6$**
 Alakítsa át ezt a kifejezést postfix kifejezéssé majd értékelje is ki! Mindkét esetben vermet használjon az algoritmus során és írja le lépésenként, hogyan alakul a verem tartalma!

5. ELLENŐRZŐ KÉRDÉSEK, TESZTKÉRDÉSEK

- 1) Milyen elemi adattípusokat ismer?
- 2) Milyen összetett adattípusokat ismer?
- 3) Mi a szekvenciális ábrázolás, mi az előnye és mi a hátránya?
- 4) Mi a dinamikusán láncolt ábrázolás, mi az előnye és mi a hátránya?
- 5) Mi a statikusán láncolt ábrázolás, mi az előnye és mi a hátránya?
- 6) Mi a jellemzője a SOR adattípusnak?
- 7) Mi a jellemzője a VEREM adattípusnak?
- 8) Mi a jellemzője a LISTA adattípusnak?
- 9) Mi a különbség a gráfok mélységi és szélességi bejárása között? A tanult algoritmus során milyen adatszerkezetek segítségével valósítottuk meg ezt a kétféle bejárást? Írja le a szélességi bejárás esetén a tanult algoritmust!
- 10) Milyen típusú nyelvtanokat ismer? Mi a Chomsky – féle hierarchia? Írjon tetszőleges példát a reguláris és a környezetfüggetlen nyelvtanra! Milyen típusú automaták ismerik fel ezeket a nyelvtanokat?
- 11) Mi az az 5 jellemző, amelyek segítségével egy véges determinisztikus vagy nem determinisztikus automata leírható? Mi a különbség egy véges determinisztikus és egy nem determinisztikus automata működése között? Adjon tetszőleges példát a kétféle automatára! (rajzban vagy átmenetfüggvény táblázat magadásával.)
- 12) Adja meg a verem adatszerkezet szintaktikáját. Melyek a konstrukciós és melyek a szelekciós műveletek? Soroljon fel három olyan algoritmust amelyek során vermet használunk. Mondjon példát rekurzív algoritmusra!
- 13) Melyik átalakító típusú automata az alábbiak közül?
 - a. véges determinisztikus automata
 - b. véges nem determinisztikus automata
 - c. Mealy automata
 - d. veremautomata
- 14) Melyik automaták ismerik fel ugyanazt a nyelvosztályt, mint a véges determinisztikus automaták?
 - a. véges nem determinisztikus automaták
 - b. Mealy automaták
 - c. veremautomaták
 - d. Moore automaták

15) Mely állítás igaz a sor adatszerkezet szemantikájára vonatkozólag?

- a. $\text{front}(\text{new}) = \text{new}$
- b. $\text{front}(\text{add}(s,e)) = \text{front}(s)$
- c. $\text{front}(\text{add}(s,e)) = e$ ha $s = \text{new}$, különben $\text{front}(s)$
- d. $\text{front}(\text{add}(s,e)) = s$ ha $s = \text{new}$, különben e

16) Mi jellemez egy halmot?

- a. Minden eleme nagyobb vagy egyenlő az elem gyerekeinél
- b. Preorder módon bejárva rendezett sorozatot kapunk
- c. Inorder módon bejárva rendezett sorozatot kapunk
- d. Minden elem esetén a bal gyerek nagyobb, mint a jobboldali gyerek.

6. MINTA VIZSGA FELADATOK.

Számítástudomány alapjai. II. Gyakorlat	VIZSGA	Név: Kód: Dátum:
--	---------------	--

1. Milyen elemi adattípusokat ismer? (3 pont)
2. Mi a jellemzője a SOR adattípusnak? (4 pont)
3. Mi a szekvenciális ábrázolás, mi az előnye és mi a hátránya? (6 pont)

4. Egészítse ki a következő algoritmusokat!

Dinamikusan láncolt ábrázolású lista:

Típus

```
ListaElem=Rekord  
                (érték:ElemTip  
                köv:ListaElem'Mutató)
```

Változó

```
fej, akt:ListaElem'Mutató  
hossz:Egész  
hiba:Logikai
```

Függvény ElemÉrték(**Változó** l:Lista):ElemTip

```
Ha akt≠sehova akkor ElemÉrték:= .....  
különben hiba:=Igaz
```

Függvény vége.

Szekvenciálisan ábrázolt verem:

Konstans MaxMélység:Egész(??)

Típus

```
VeremElemek=Tömb(1..MaxMélység:ElemTip)
```

Változó

```
ve:VeremElemek  
teteje:0..MaxMélység  
hiba:Logikai
```

Függvény Üres?(**Konstans** v:Verem):Logikai

```
.....  
.....
```

Függvény vége.

5. Rendezze a következő számsorozatot beillesztéses rendezéssel. (7 pont)
43, 21, 7, 56, 15

6. Rajzolja fel a - - - **a b c d** prefix kifejezéshez tartozó fát! (5 pont)
7. Ha egy csúcs fokszáma **0** akkor csúcsról beszélünk. (2 pont)
8. Ha két gráf akkor biztosan ugyanannyi csúcsuk és élük van. (2 pont)
9. Melyik adattípusnál nincs jelentősége az elemek sorrendjének? (2 pont)
10. Milyen adattípusok tartoznak a sokaság típushoz? (3 pont)
11. A vasúti nyilvántartás tartalmazza a Savaria expresszre kiadott helyjegyeket.
Írjon programot, amely meghatároz egy szabad helyet a vonaton! (8 pont)

Töltse ki a kipontozott részeket a megfelelő szóval vagy betűvel, számmal.

- 1./ A 2. típusú nyelvek felismerői a automaták.
- 2./ A legszűkebb nyelvosztály a nyelvek osztálya.
- 3./ Az alábbi szabályok: $A \rightarrow \omega$ ($\omega \in (N \cup T)^*$) a nyelvtanokat jellemzik.
- 4./ A felismerő automata átmenetfüggvénye az alábbi alakú: $\delta: A^*V \rightarrow A$. Ekkor véges automatáról beszélünk.
- 5./ A BNF-fel típusú nyelvek szabályait lehet leírni.
- 6./ A verem szelekciós műveletei a és a
- 7./ A buborékrendezés műveletigénye n elem esetén:
- 8./ Adott az alábbi élmátrix-szal rendelkező gráf:

	V1	V2	V3
V1	0	1	1
V2	1	0	0
V3	1	0	0

Mi lesz a Warshall-algoritmus első lépése után kapott mátrix, azaz a V1 csúcs felhasználásával kapható utak mátrixa?(Csak az útmátrixot keressük, legrövidebb utakat nem.)

	V1	V2	V3
V1			
V2			
V3			

- 9./ Adott az alábbi nemdeterminisztikus automata: $A = \langle \{A,B\}, \{0,1\}, A, \{B\}, \delta \rangle$

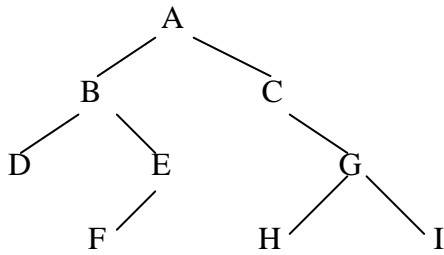
δ	0	1
A	A	B,A

B	B,A	B
---	-----	---

A vele ekvivalens determinisztikus automata átmenetfüggvénye: (Töltse ki a táblázatot!)

δ'		

10./



A fenti fát preorder módon végigolvasva a csúcsok sorrendje a következő:

.....

11./ A fenti fát inorder módon végigolvasva a csúcsok sorrendje a következő:

.....

12./ A fenti fát posztorder módon végigolvasva a csúcsok sorrendje a következő:

.....

13./ Adott az alábbi sorozat: 17, 8, 26, 23, 15, 40

Rendezze a sorozatot összefésüléssel. Az első 3 lépés eredményét írja le!

.....

.....

.....

Irodalomjegyzék

- [1] ***Demetrovics – Denev – Pavlov***: A számítástudomány matematikai alapjai.
Nemzeti Tankönyvkiadó
- [2] ***Katona – Recski - Szabó***: A számítástudomány alapjai.
Typotext kiadó
- [3] ***T.H.Cormen, C. E. Leiserson, R. L. Rivest***: ALGORITMUSOK
Szerk:Iványi Antal
Műszaki Könyvkiadó
- [4] ***µlogia34: Pappné – Szlávi - Zsakó***: Módszeres programozás: Adattípusok
ELTE TTK Informatikai Tanszékcsoport 1998
- [5] ***µlogia2: Szlávi – Zsakó***: Programozási forgácsok.
ELTE TTK Informatikai Tanszékcsoport
- [6] ***Lipschutz***: Adatszerkezetek.
Schaum könyvek. Panem Kft 1993.
- [7] ***µlogia18: Szlávi - Zsakó***: Módszeres programozás. Programozási bevezető
ELTE TTK Informatikai Tanszékcsoport 1993
- [8] ***µlogia27: Pappné – Szlávi - Zsakó***: Módszeres programozás: Rekurzív típusok
ELTE TTK Informatikai Tanszékcsoport 1994
- [9] ***µlogia27: Szlávi***: Előadás a sorozattípusokról
ELTE TTK Informatikai Tanszékcsoport 1992