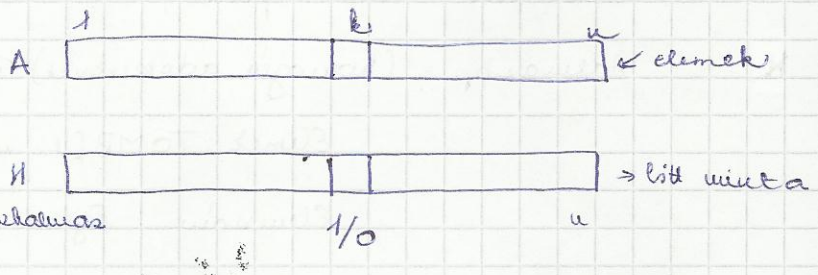
1. előadás

Halmazok

- Az elemek száma változhat, az egyes elemek közötti sorrendiséget nincs értelme megadni. Az egyes adatszerkezetek elemeit csak egyszer érdemes megadni. Csak egyszer szerepel.
- 2 féle megadási mód: – sorozat, amibe felvesszük az alaphalmaz elemeit

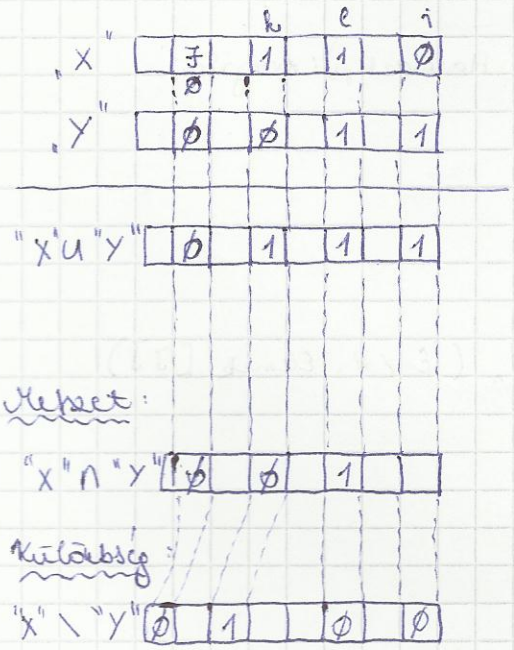


$H \subset A \qquad a_i \in A$

Egy elem meglétét 1 bit jelzi! ~ 1/0 aszerint, hogy k eleme-e A-nak.

$k_i = 1$, ha $a_i \in H$

Unióképzés



"X"∪"Y" = "X ∪ Y" – vagy

Metszet:

"X"∩"Y"

– és

Különbség:

"X" \ "Y"

– Hf: és részhalmaz-e.

- Meg tudjuk nézni, hogy üres-e a halmaz, $\emptyset$-e az értéke. Ha nem, lineáris kereséssel végigmegyünk rajta, vagy az első nem $\emptyset$-ig.
- Kéréri kell egy alkalmas adatszerkezetet. Konstrukció. Van egy alaphalmaz + több részhalmaz. Végtelen halmazokkal a számítógép nem tud mit kezdeni, ezért csak végest használunk.

Konstans Maxelemszám : elemek max. számai

TÍPUS: Elemtípus = elemek típusa;

Halmaztíp = (homogén adatszerkezet) Rekord

Elemek : TÖMB[1.. maxelemszám]: Elemtíp

Elemszám : Egész

R vége

Halmaz inicializálása:

Eljárás Halmazkesd (H: Halmaztíp);

H. Elemszám := $\emptyset$;

E vége

Függvény Eleme (E: Elemtíp; H: Halmaztíp): logikai

{eldöntés kérdése}

Változó I: egész;

I := 1;

Ciklus míg (I ≤ H. Elemszám), (E ≠ H. Elemek [I])

I := I + 1;

C vége;

Eleme := (I ≤ H. Elemszám);

F vége;

Üres-e a halmaz? logikai fgv.

Függvény Halmazüres (H : Halmaz): logikai;

Halmazüres = (H. Elemszám := ∅);

F vége

Megtelt-e a halmaz?

Függvény Halmazteli (H : Halmaz) : logikai

Halmazteli = (H. Elemszám = Maxelem);

F vége;

Halmazelméleti műveletek vetülete:

∩ ∪ \ $|A \cap B|$ elemszáma $\leq MIN(|A|; |B|)$ {2 halmaz elemszámai közül a legkisebb}

$(A \subset B \vee B \subset A \Rightarrow "=")$ Ha 2 halmaz diszjunkt, vagyis nincs közös elem, az elemszám ∅.

$|A \setminus B| \leq A$

A : = {1..10}
B := {5..100}
A\B : = {1, 2, 3, 4}

$|A \cup B| \leq |A| + |B|$ – Ez az érték lehet nagyobb, mint a maxelemszám. Ez hibát eredményezhet.

Eljárás Metszet (A, B, C : Halmaztip);

Változó J: egész;

C. Elemszám : = ∅;

Ciklus J : = 1.. A. Elemszám

Ha Eleme (A. elemek [J], B) akkor (B-nek is eleme)

C. Elemszám := C. Elemszám + 1

C. Elemek [C. Elemszám] := A. Elemek [J];

```
    H vége
C vége;

Eljárás Unio (A, B, C: Halmaztip);
    Változó J: Egész;
    C := A;
    Ciklus J := 1 .. B.Elemszám;
        Ha NemEleme (B.Elemek[J], A) akkor
            C.Elemszám := C.Elemszám + 1
            C.Elemek[C.Elemszám] := B.Elemek[J];
        H vége
    C vége

Eljárás Különbség (A, B, C: Halmaztip);
    Változó J: Egész;
    Ciklus J := 1 .. A.Elemszám;
        Ha NemEleme (A.Elemek[J], B) akkor
            C.Elemszám := C.Elemszám + 1;
            C.Elemek[C.Elemszám] := A.Elemek[J];
        H vége;
    C vége;

Hf.: eljárás részhalmaz eldöntés létrehozása
    Ha |A| > |B| ⇒ A ⊄ B
```

2. előadás

Lista

$\{a_n\} \wedge \{b_n\}$ → különbségnél

~: Egy homogén, dinamikus adatszerkezet.
↓
véges, az adatszerkezet elemeinek a számára mindig annyi helyet foglalunk le, amennyire szükség van. Ha nincs eleme, akkor a lefoglalt hely minimális.
nem üres, h. ez ∅.

Művelet: pl.: töröljük egy elemét → az adatterület egy adat tárigényével csökken (felszabadul memóriaterület)

- A hozzájuk rendelt tárigény → folytonos reprezentációtól beszélünk (ha minden elem memóriaterülete 1 tartományon belül van, itt mást nem tárolunk).
- szétszórt repr. az adatok szétszórtan helyezkednek el.
  Ha találunk 1 olyan memóriaterületet, ahol 1 adat elfér, ott lefoglal neki helyet.

- Folytonos repr. esetén, ha tudjuk ∀ adatelem méretét (tudjuk, h. a többi is akkora) → az első adatelem címének ismeretében számíthatjuk a többi adatelem címét.

  A[1.. n]

  LOC(A) első tömbelem címe a memóriában

ha tudjuk, h. 1 elem tárigénye c · (3), igaz az, h a $LOC(A[k]) = LOC(A) + (k-1) \cdot 3$ CÍMFÜGGVÉNY

- a rendszer a címfgv alapján tudja a folyt. repr. egyes elemeit címét elérni

– Szétszórt repr:

Két részből áll · a; tárolandó adatot

b; azzal kapcsolatban tárt információt, h. az adatelem hol található meg a memóriában

- Minden adatelemhez valamilyen valamilyen információ.
- A olyan típust, amely adatcímet mutat meg, MUTATÓ TÍPUSNAK nevezzük. (Pascalban → Pointer)
- A általa mutatott adattal végzett műveletek. Nem az adatot adjuk meg, hanem a helyét, így programot is azonosíthat. Ez a közvetett hivatkozás. (INDIREKCIÓ)

k → sorszám

A[k] → az indexelés is indirekció → az adott tömbön van értelmezve.

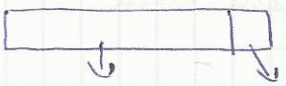
sőt tudjuk kezelni, mert van olyan elem, ahol le van írva, hol található, ez alapján lehet megtalálni.

Törlés esetén az adatszerkezeten belül nincs olyan mutató, ami rámutat.

- Így elveszett az információ, hol található. Ezért adminisztrációt kell vezetni, hogy melyik helyet szabadok. Létrehozunk egy dinamikus adatszerkezetet, amelyben tároljuk a törölt elemeket.

Láncolt lista

A mutatók értékei között van egy, kitüntetett. Szokás NIL-nek nevezni, v. VÉGJEL-nek. [jele: Ø.]



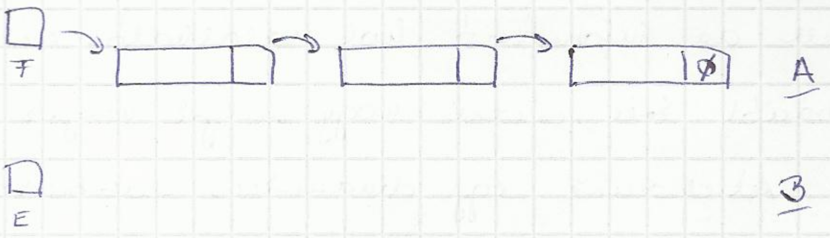
állhatnak több adatból, v. mutatóból.

A mutató mutatja meg, hol található az őt követő elem.

A elemet ismerjük, ha tudjuk az előtte álló helyét. Nincs leírva a lista első elemének a címe, ide kell egy mutató, amely megmutatja az első elem helyét. (FEJ)
↳ tárolni címét

Elegendő a fejet ismerni az adatok megismeréséhez.

- A mutató tárigénye 4 Byte. → címei, és 1 címét azonosít.
- Ha megváltoztatjuk a fej értékét, máshová mutat.



B := A;

ahol nem engedi meg a program, ott ciklust írni (pl.: BASIC)

De itt elegendő E:=F, és lehet az egész 100KB is, mégis 4 Byte-ot kell megmozgatni hozzá.

- Ha paraméterátadást akarunk csinálni, paraméterként a fejet kell megadni, nem paraméter szerint 4 Byte lesz a tárigény.

- Vannak olyan nyelvek, amelyekkel dinamikus változókat lehet létrehozni, ahol van mutató típus.

Pl.: PASCAL

nem ilyen pl.: a BASIC

<u>Dinamikus tárkezeléshez kell:</u>

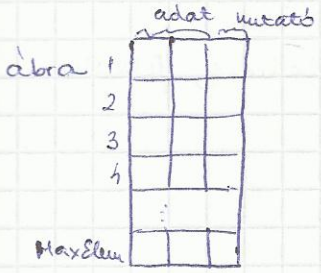
NEW → bemenete egy típus, kimenete egy cím.

Egy adott területen az adott típusú változót el lehet tárolni.

DISPOSE: a pointerben megadott tárterületet felszabadítja

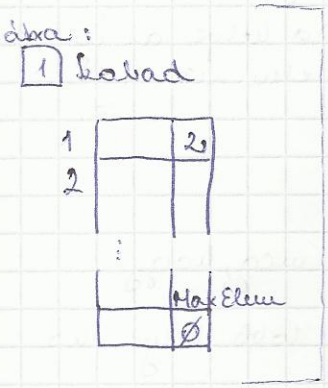
MODELL:

```
Típus   MutatóTípus = Egész;
        ÉrtékTípus = tárolandó_értékek_típusa; (adatrész típusa)
        ListaElemTípus = Rekord
                Érték : ÉrtékTípus;
                Köv : MutatóTípus;
        RVége
Konstans  MaxElem = maximális_elemszám;
          Végjel := 0;
Változó   Elemek : Tömb[1.. MaxElem] ListaElemTípus;
```



Szabad helyek kezelése:

```
Változó   Szabad : MutatóTípus
          (↳ a szabad lista első elemére mutat)
```




```
Eljárás SzabadLista Kezd;
    Változó I: Egész;
    Algoritmus
        Szabad := 1;
        Ciklus I := 1 .. MaxElem - 1
            Elem[I].Köv := I+1;
```

Cvége

Elem[MaxElem].Kov := Végjel;

Evége;

- A lista üres, ha a FEJ-ben Végjel van.
- Ha az i-dik elemet ímni-olvasni akarjuk, az (i-1)-diket végig kell olvasni, ímni.

Célszerű a lista elejére láncolni, helyfoglaláskor a lista elemét foglaljuk le. (Ha törölni akarjuk, a szabad listából törölni kell.)

Lefoglaláskor az első elem elé fogalmunk, felszabadításkor az első elem helyére.

Függvény Lefoglal (Hely: Mutató Típus): Logikai

{Ha valaki meghívja a lefoglalt elem címét, az a lefoglalt elem helyével tér vissza. Beírjuk a végjelet a mutatójába, és a végjel helyére a P-t.
↳ a helye az új elemnek.

Lefoglal (P).

→ hová mutat a mutató
<--- értékadás

Azért fgv és logikai, mert az adja vissza, hogy sikerült-e lefoglalni helyet. Nincs több hely, ha a szabad listafejben végjel van.}

Algoritmus

Ha Szabad <> Végjel Akkor

Hely := Szabad; (az első listaelemet foglaljuk le.)