SOR:

Def.: Homogén adatelemek olyan sorozata, amelyen két alapművelet értelmezett.
- új elem elhelyezése a sor végére
- elem kivétele a sor elejéről.

FIFO (First in first out)

1., Megvalósítása:

```
konstans  maxelem = max. elemszám

típus     elemtip : tárolandó elem típusa

típus     sortip : rekord (

              elem : tömb [ 1 .. maxelem ] elemtip
              első, utolsó : egész
          )
```

2., Kezelése:

a, inicializálása:

```
eljárás sorkezd (sor : sortip)

  sor. első := 1 ;
  sor. utolsó := 0;
vége
```

b, állapot lekérdezése:

```
függvény sorüres (sor: sortip) : logikai

  sorüres := (sor. első > sor. utolsó) ;

vége

függvény sortele (sor : sortip) : logikai

    sortele := ( sor. utolsó = maxelem);

vége
```

c, Új elem elhelyezése:

```
függvény sorba (sor: sortip; adat: elemtip): logikai
  ha nem sortele (sor) akkor
      sor.utolsó := sor.utolsó +1;
      sor.elem[sor.utolsó] := adat;
      sorba := igaz
  különben
      sorba := hamis;
  ha vége
vége.
```

d, Elem kivétele:

```
függvény sorból (sor: sortip; adat: elemtip): logikai.
  ha nem sorüres (sor) akkor
      adat := sor.elem[sor.első];
      sor.első := sor.első +1;
      sorból := igaz;
  különben
      sorból := hamis;
  ha vége;
vége.
```

e, Elem kivétele léptetéssel:

```
függvény lépsorból (sor: sortip; adat: elemtip): logikai
  változó i: egész
  ha nem sorüres (sor) akkor
      adat := sor.elem[sor.első];
      ciklus i := sor.első .. sor.utolsó - 1 ismm;
          sor.elem[i] := sor.elem[i+1];
      ciklus vége;
      sor.utolsó := sor.utolsó - 1;
      lépsorból := igaz
  különben
      lépsorból := hamis;
  ha vége
vége.
```

3, Üllsos sor:

a, inicializálás

sorkezd eljárás

b, sor állapotának vizsgálata:

függvény üllsorures (sor: sortip): logikai

üllsorures := (sor.utolsó = 0);

vége

függvény üllsortele (sor: sortip): logikai

üllsortele := ((sor.első = 1 és sor.utolsó = maxelem)

vagy (sor.utolsó > 0 és sor.utolsó = sor.első - 1))

vége

c, Új elem elhelyezése:

függvény üllsorba (sor: sortip; adat: elemtip): logikai

ha nem üllsortele (sor) akkor

ha sor.utolsó < maxelem akkor

sor.utolsó := sor.utolsó + 1;

különben

sor.utolsó := 1;

ha vége

sor.elem [sor.utolsó] := adat;

üllsorba := igaz

különben

üllsorba := hamis;

ha vége

vége.

d.; elem kivétele:

```
függvény allsorból(sor: sortip; adat: elemtip): logikai
ha nem sorüres(sor) akkor
    adat := sor.elem[sor.első]
    ha sor.első = sor.utolsó akkor
        sorkezd(sor)
    különben
     ha sor.első < maxelem akkor
        sor.első := sor.első + 1
     különben
        sor.első := 1;
     ha vége;
    ha vége;
    allsorból := igaz;
  különben
    allsorból := hamis;
 ha vége;
vége.
```

HALMAZ:

1.; Megvalósítása:

```
konstans    maxelem = max elemszám
típus       elemtip : tárolandó elemek típusa
típus       halmaxtip : rekord(
                elem : tömb[1..maxelem] elemtip
                elemszám : egész
            )
```

2., Kezelése:

a.; inicializálása:

```
eljárás halmaxkezd(halmaz: halmaxtip);
 halmaz.elemszám := 0;
vége.
```

b, állapot lekérdezése:

```
függvény halmazüres (halmaz : halmaztip) : logikai
  halmazüres := (halmaz.elemszám = 0)
vége.

függvény halmazteli (halmaz : halmaztip) : logikai
  halmazteli := (halmaz.elemszám = maxelem),
vége

függvény eleme (halmaz : halmaztip; adat : elemtip) : logikai;
  változó i : egész;
  i := 1;
  amíg (i <= halmaz.elemszám) és
       (halmaz.elem[i] <> adat) ism
    i := i+1;
  a vége
  eleme := (i <= halmaz.elemszám)
vége
```

3, Halmazműveletek:

a, metszet: (A∩B)

```
eljárás metszet (A, B, C : halmaztip)
  változó i : egész;
  C.elemszám := 0
  ciklus i := 1 .. A.elemszám ism.
    ha eleme (B, A.elem[i]) akkor
      C.elemszám := C.elemszám + 1;
      C.elem[C.elemszám] := A.elem[i]
    h vége
  c vége
vége
```

b., unió : A∪B

```
eljárás unio (A,B,C : halmaztip);
változó i : egész
ciklus i := 1.. A.elemszám ism
  C.elem[i] := A.elem[i]
c vége
C.elemszám := A.elemszám
ciklus i := 1.. B.elemszám ism.
ha nem eleme (A, B.elem[i]) akkor
  C.elemszám := C.elemszám +1
  C.elem [C.elemszám] := B.elem[i];
h vége;
c vége;
vége.
```

c., különbség : (A\B)

```
eljárás különbség (A,B,C:halmaztip);
változó i:egész;
C.elemszám := 0;
ciklus i := 1.. A.elemszám ism
  ha nem eleme (B,A.elem[i]) akkor
    C.elemszám := C.elemszám+1
    C.elem [C.elemszám] := A.elem[i]
  h vége;
c vége;
vége.
```

d., részhalmazvizsgálat: (BCA)

függvény részhalmaz (A, B : halmaztip) : logikai

változó i : egész;

ha A.elemszám < B.elemszám akkor

részhalmaz := hamis;

különben

i := 1;

amíg (i <= B.elemszám) és

elem (A, B.elem[i]) is

i := i + 1;

a vége

részhalmaz := (i > B.elemszám)

ha vége;

vége;

e., új elem felvétele:

függvény halmazba (halmaz : halmaztip; adat : elemtip) : logikai

ha nem halmazelle (halmaz) és

nem eleme (halmaz, adat) akkor

halmaz.elemszám := halmaz.elemszám + 1;

halmaz.elem[halmaz.elemszám] := adat;

halmazba := igaz.

különben

halmazba := hamis;

ha vége;

vége.

LISTA:

Olyan szekvenciális adatszerkezet, melynél mindegyik eleme az adatelemen kívül arra vonatkozólag is tartalmaz információkat, hogy melyik a sorou következő elem. Ez az információ a következő elem címe.

Szekvenciális adatszerkezet: az adatelemek egymás után, egymással következnek, továbbá egy elem feldolgozására csak a megelőző elemek feldolgozása után van lehetőség.

Minden listához tartozik egy változó, a LISTAFEJ.

A listaelemek nem közvetlenül, hanem csak a láncon végighaladva érhetők el, a listafej az egyetlen „szilárd pont".

Hogy felismerhessük a lista végét, szükséges egy speciális mutatóérték, a VÉGJEL.

A listát olyan problémák esetén érdemes használni, ahol: - nem ismerjük a tárolandó elemek számát

- jelentős a „mozgás" az elemek között (folyamatosan kerülnek új elemek a listába, vagy törlődnek belőle.)

1, Megvalósítása:

típus mutatótip: egész

típus értéktip: tárolandó értékek típusa

típus listaelemtip: rekord (

érték: értéktip

köv: mutatótip

)

```
konstans   maxelem = max. elemszám ;
változó    elem : tömb [ 1 .. maxelem ] listaelemtip ;
konstans   végjel = 0 ;
```

2, Szabad helyek kezelése:

változó szabad : mutatótip

- helyfoglalás új elem számára (kivesz egy elemet a szabad listából és átadja a címét a hívónak.)
- egy törölt elem helyének felszabadítása (az elemet visszafűzi a szabad listába)

a, inicializálás:

```
eljárás szabadrekd;
  változó i:egész;
  szabad := 1 ;
  ciklus i := 1 .. maxelem - 1  ism.
    elem [i] . köv := i+1 ;
  c vége ;
  elem [ maxelem ] . köv := végjel ;
vége.
```

b, elem lefoglalása:

```
fgv  lefoglal ( hely : mutatótip ) : logikai;
ha szabad <> végjel akkor
        hely := szabad ;
        szabad := elem [szabad]. köv ;
        lefoglal := igaz
különben
        lefoglal := hamis ;
h vége;
vége.
```

c., elem felszabadítása:

```
eljárás felszabadít (hely: mutatótíp);
ha hely <> végjel akkor
    elem[hely].köv := szabad;
    szabad := hely;
ha vége;
vége.
```

3; Listakezelő alapműveletek:

a., lista inicializálása:

```
eljárás listarend (fej: mutatótíp);
    fej := végjel;
vége.
```

b., lista feldolgozása:

```
eljárás listafeldolg (fej: mutatótíp);
változó P: mutatótíp;
p := fej;
amíg p <> végjel ism
    elem[P].érték feldolgozása;
    p := elem[p].köv;
a vége;
vége.
```