Dinamikus adatstruktúrák

- Változók:
  - **Statikus változók**: helyigényük, méretük fordítási időben eldől.
  - **Dinamikus változók**: futási időben hozhatók létre ill. szüntethetők meg.

- Mutatók: memóriacímek, amelyek a központi memória egy adott helyét címzik, arra mutatnak.
  - **Adatra mutatók**: adatot tároló memóriarészre mutatnak.
  - **Szubrutinra mutatók**: programkódot tartalmazó memóriarészre mutatnak.

- Adatmutatók: egy adott **Típus** típusú adatra mutató mutatók.
  - Konstans: NIL
  - Műveletek:
    - Hasonlítások: Egyenlő (=), Nem egyenlő (<>)

Dinamikus adatstruktúrák

- Operátorok:
  - Címoperátor: &
  - Indirekció operátor: *

Funkció	Azonosító	Típus	Jelleg
Egy egész szám	A	Egész	M
Egy egész szám címe	B	Egészre mutató	M

```
A ← 2          /* A-ba 2-t teszünk */
B ← &A         /* B mutasson A-ra */
Ki: *B         /* 2-t ír ki */
*B ← *B+1      /* B által mutatott változó használata */
Ki: A          /* 3-t ír ki */
```

- Megjegyzés:
  - Rekordokra és tömbökre mutató mutatók esetén egy rekordmező ill. egy tömbelem hivatkozása: Pl. (*RM).MEZO, (*TM)[I], ahol RM egy rekordra, TM egy egydimenziós tömbre mutató mutatóváltozó.

Dinamikus adatstruktúrák

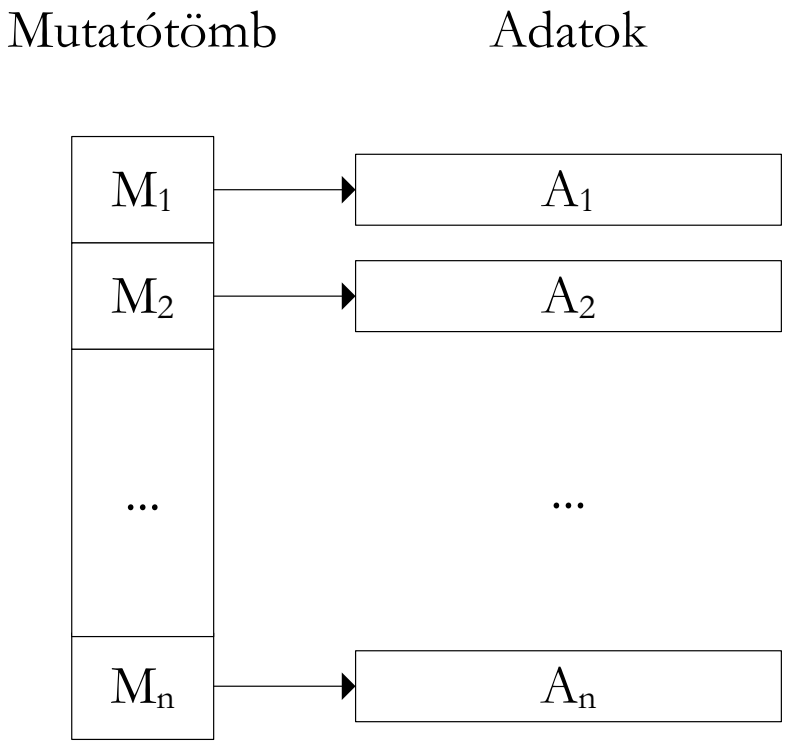
- Segédszubrutinok:
  - HELYFOGLAL(P)
    - **P** egy **Típus** típusú adatra mutató változó.
    - Az eljárás akkora helyet foglal a szabad memóriából, amely egy **Típus** típusú adat tárolásához kell.
    - Ha van elég szabad hely, akkor **P** értéke a lefoglalt terület kezdőcíme lesz, egyébként hiba lép fel.
  - VANHELY(Méret)
    - A függvény igaz értéket ad, ha van még **Méret** bájt, összefüggő szabad hely a dinamikus tárban, egyébként hamisat.
  - MERET(Azonosító)
    - A függvény eredménye egy változó ill. típus mérete, helyigénye bájtokban.
  - FELSZABADIT(P)
    - **P** egy **Típus** típusú adatra mutató változó.
    - Az eljárás felszabadítja a **P** által mutatott címen lévő, **Típus** típusú változó által elfoglalt helyet, azaz megszünteti a címen lévő dinamikus változót.
    - **P** értéke ezután definiálatlan lesz.

Dinamikus adatstruktúrák

Funkció	Azonosító	Típus	Jelleg
Egy egész szám címe	A	Egészre mutató	M

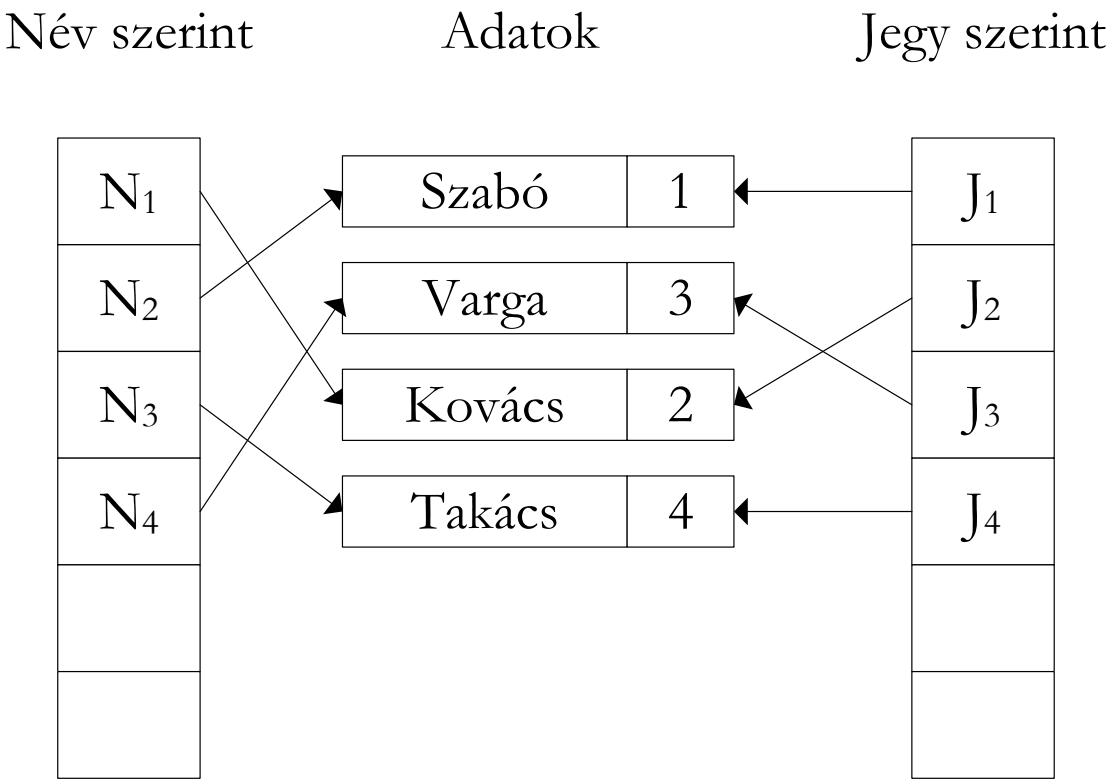
```
                          /*  *A még nem használható  */
HELYFOGLAL(A)             /*  Az A változó értéket kap  */
*A ← 2                    /*  *A dinamikus változó használata  */
Ki: *A                    /*  2-t ír ki  */
FELSZABADIT(A)            /*  *A dinamikus változó megszüntetése  */
                          /*  *A már nem használható  */
                          /*  Az A változó értéke definiálatlan  */
```

Dinamikus adatstruktúrák



Kollekció

Dinamikus adatstruktúrák



Azonos adatrészű kollekciók

Dinamikus adatstruktúrák

- **Feladat**: Készítsünk szubrutinokat lottószelvények kiértékelésére. Az adatokat a háttértáron szövegfájlban, a memóriában egy kollekcióban tároljuk.

Dinamikus adatstruktúrák

- **Konstans**

```
MAXSZELV  16000        /* A betölthető szelvények max. száma */
DB  6                  /* A lottószámok db száma egy szelvényen */
MAX  45                /* Egy lottószám maximális értéke */
MINTAL  3              /* A minimális találatszám a nyeréshez */
```

- **Típus**

```
SZELVENY Egydimenziós egész tömb[DB]                    /* Lottószelvény */
TETEL Rekord                                            /* A kollekció tétele */
        SZ          SZELVENY                            /* A számok */
        TAL         Egész                               /* Találatok száma */
KOLLEKCIO Egydimenziós TETEL-re mutató tömb[MAXSZELV]    /* A kollekció */
STAT Egydimenziós egész tömb                            /* Statisztika */
```

Dinamikus adatstruktúrák

- **Feladat**: Töltsünk be adatokat egy szövegfájlból a memóriába kollekció segítségével! A fájl soronként egy szabályos lottószelvény számait tartalmazza.

Dinamikus adatstruktúrák

- **Feladat**: Töltsünk be adatokat egy szövegfájlból a memóriába kollekció segítségével! A fájl soronként egy szabályos lottószelvény számait tartalmazza.

Funkció	Azonosító	Típus	Jelleg
A fájl azonosítója	FNEV	Sztring	I
Az eredmény kollekció	K	KOLLEKCIO	O
A kollekció elemszáma	KDB	Egész	O
Betöltöttük-e az összes adatot	OK	Logikai	O
A fájlváltozó	F	Szövegfájl	M
Segédváltozó	I	Egész	M

Dinamikus adatstruktúrák

```
/* Az FNEV nevű szövegfájl szelvényeinek betöltése */
BETOLT(FNEV,K,KDB)
NYIT(F,FNEV,"I")
KDB ← 0
while NOT FAJLVEGE(F) AND VANHELY(MERET(TETEL)) AND (KDB<MAXSZELV)
      KDB ← KDB+1
      HELYFOGLAL(K[KDB])
      for I ← 1,DB
            Be F: (*K[KDB]).SZ[I]
OK ← FAJLVEGE(F)
ZAR(F)
return OK
```

Dinamikus adatstruktúrák

- **Feladat**: Értékeljünk ki egy, az előzőekben deklarált kollekcióban lévő szelvényeket, adott nyerőszámok alapján!
  - A nyertes szelvényeket (amelyeken legalább MINTAL találat volt), írjuk ki egy szövegfájlba úgy, hogy először a telitalálatos szelvényeket, legvégül a MINTAL találatos szelvényeket listázzuk a darabszámuk feltüntetésével!

Dinamikus adatstruktúrák

- **Megoldás**:
  - A könnyebb kiértékeléshez a nyerőszámokból halmazt készítünk.
  - Az egyes szelvények találatainak számát, magában a kollekcióban tároljuk (TAL mező).
  - Az egyes találatokhoz (0, 1, …, DB) tartozó, ilyen találatos szelvények számát egy (STAT típusú) tömbben gyűjtjük.

Dinamikus adatstruktúrák

Funkció	Azonosító	Típus	Jelleg
Az eredmény fájl azonosítója	FNEV	Sztring	I
A kiértékelendő szelvények	K	KOLLEKCIO	I, M, O
A szelvények száma	KDB	Egész	I
A nyerőszámok tömbje	NY	SZELVENY	I
A nyerőszámok halmaza	H	Egészhalmaz	M
A találat statisztika	S	STAT	M
A fájlváltozó	F	Szövegfájl	M
Segédváltozók	I, J, L	Egész	M

Dinamikus adatstruktúrák

```
/* Kiértékelés, a nyerőszelvények kiírása szövegfájlba */
ERTEKEL(FNEV,K,KDB,NY)
/* Nyerőszámok halmaza */
H ← []
for I ← 1,DB
    H ← H+[NY[I]]
/* Statisztika kezdőértéke */
for I ← 0,DB
    S[I] ← 0
/* A szelvények kiértékelése */
for I ← 1,KDB
    (*K[I]).TAL ← 0
    for J ← 1,DB
        if (*K[I]).SZ[J] IN H
            (*K[I]).TAL ← (*K[I]).TAL+1
    /* Statisztika */
    S[(*K[I]).TAL] ← S[(*K[I]).TAL]+1
/* Adatkiírás */
...
```

Dinamikus adatstruktúrák

```
...
/* Adatkiírás */
NYIT(F,FNEV,"O")
Ki F:"A nyerőszámok"
for I ← 1,DB
      Ki F: NY[I]
/* A nyerőszelvények */
for L ← DB,MINTAL,−1
      Ki F: L," találatos szelvények száma:",S[L]
      for I ← 1,KDB
            if (*K[I]).TAL=L
                  for J ← 1,DB
                        Ki F: (*K[I]).SZ[J]
ZAR(F)
```

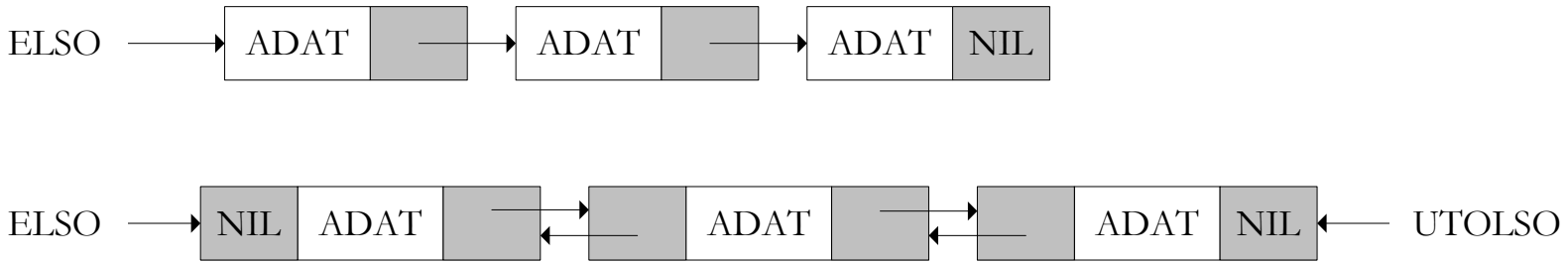
Dinamikus adatstruktúrák

LANCELEM1 Rekord

ADAT	A tárolandó adat típusa
KOVETO	LANCELEM1 rekordra mutató

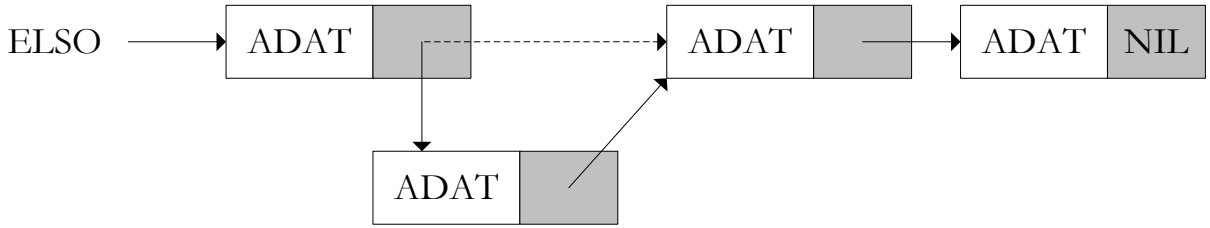
LANCELEM2 Rekord

ADAT	A tárolandó adat típusa
ELOZO, KOVETO	LANCELEM2 rekordra mutató

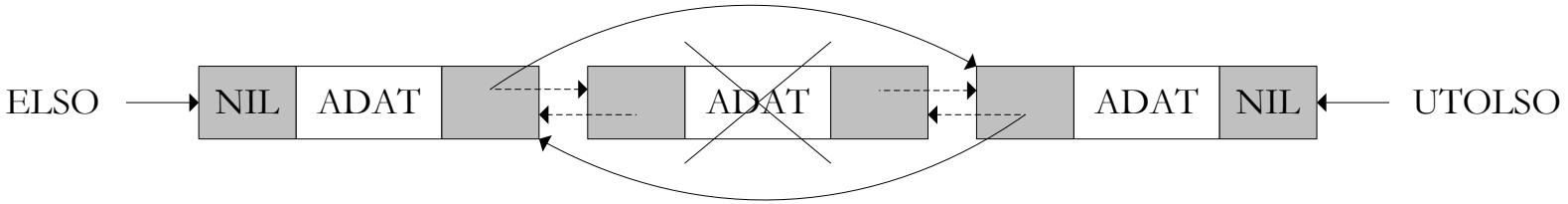


Egy- és kétirányban láncolt lista

Dinamikus adatstruktúrák



Elem beszúrása egy egyirányban láncolt listába



Elem törlése egy kétirányban láncolt listából

Dinamikus adatstruktúrák

- **Feladat**: Keressünk meg egy adott értéket egy egyirányban láncolt listában!

Dinamikus adatstruktúrák

- **Feladat**: Keressünk meg egy adott értéket egy egyirányban láncolt listában!

- **Típus**

LANCELEM1 Rekord

ADAT	Egész
KOVETO	LANCELEM1 rekordra mutató

Funkció	Azonosító	Típus	Jelleg
A lista első eleme	ELSO	LANCELEM1-re mutató	I
A keresett érték	X	Egész	I
Az aktuálisan vizsgált elem	AKT	LANCELEM1-re mutató	M, O

Dinamikus adatstruktúrák

```
/* Keresés egy egyirányban láncolt, rendezetlen listában */
KERESES1(ELSO,X)
AKT ← ELSO
while (AKT<>NIL) AND ((*AKT).ADAT<>X)
      AKT ← (*AKT).KOVETO
return AKT
```

Dinamikus adatstruktúrák

```
/* Keresés egy egyirányban láncolt, rendezett listában */
KERESESREND1(ELSO,X)
AKT ← ELSO
while (AKT<>NIL) AND ((*AKT).ADAT<X)
      AKT ← (*AKT).KOVETO
if (AKT<>NIL) AND ((*AKT).ADAT>X)
      AKT ← NIL
return AKT
```

Dinamikus adatstruktúrák

- **Feladat**: Bővítsünk egy egyirányban láncolt rendezett listát egy új listaelemmel!

Dinamikus adatstruktúrák

- **Feladat**: Bővítsünk egy egyirányban láncolt rendezett listát egy új listaelemmel!

Funkció	Azonosító	Típus	Jelleg
A lista első eleme	ELSO	LANCELEM1-re mutató	I, O
A beszúrandó listaelem	UJ	LANCELEM1-re mutató	I
Az aktuálisan vizsgált elem	AKT	LANCELEM1-re mutató	M
A listaelem ami után beillesztjük az új listaelemet	MIUTAN	LANCELEM1-re mutató	M

Dinamikus adatstruktúrák

```
/* Rendezett lista bővítése */
LISTARAREND1(ELSO,UJ)
/* Keresés */
AKT ← ELSO
MIUTAN ← NIL
while (AKT<>NIL) AND ((*AKT).ADAT<(*UJ).ADAT)
      MIUTAN ← AKT
      AKT ← (*AKT).KOVETO
/* Beillesztés */
...
```

Dinamikus adatstruktúrák

```
...
/* Beillesztés */
if ELSO=NIL
    /* Üres listára */
    (*UJ).KOVETO ← NIL
    ELSO ← UJ
else if MIUTAN=NIL
    /* Nem üres lista elejére */
    (*UJ).KOVETO ← ELSO
    ELSO ← UJ
else if AKT=NIL
    /* A lista végére, a MIUTAN után */
    (*UJ).KOVETO ← NIL
    (*MIUTAN).KOVETO ← UJ
else
    /* A MIUTAN és az AKT közé */
    (*UJ).KOVETO ← (*MIUTAN).KOVETO
    (*MIUTAN).KOVETO ← UJ
```

Dinamikus adatstruktúrák

- **Feladat**: Vegyünk fel egy új listaelemet egy kétirányban láncolt lista legvégére!

Dinamikus adatstruktúrák

- **Feladat**: Vegyünk fel egy új listaelemet egy kétirányban láncolt lista legvégére!

- **Típus**

LANCELEM2 Rekord

ADAT	Egész
ELOZO, KOVETO	LANCELEM2 rekordra mutató

Funkció	Azonosító	Típus	Jelleg
A lista első eleme	ELSO	LANCELEM2-re mutató	I, O
A lista utolsó eleme	UTOLSO	LANCELEM2-re mutató	I, O
A felveendő listaelem	UJ	LANCELEM2-re mutató	I

Dinamikus adatstruktúrák

```
/* Beillesztés egy kétirányban láncolt lista végére */
LISTARA2(ELSO,UTOLSO,UJ)
if ELSO=NIL
    /* Üres listára */
    (*UJ).ELOZO ← NIL
    (*UJ).KOVETO ← NIL
    ELSO ← UJ
    UTOLSO ← UJ
else
    /* Az UTOLSO után */
    (*UJ).ELOZO ← UTOLSO
    (*UJ).KOVETO ← NIL
    (*UTOLSO).KOVETO ← UJ
    UTOLSO ← UJ
```

Dinamikus adatstruktúrák

- **Feladat**: Töröljünk egy adott mutatójú listaelemet egy kétirányban láncolt listáról!

Dinamikus adatstruktúrák

- **Feladat**: Töröljünk egy adott mutatójú listaelemet egy kétirányban láncolt listáról!

Funkció	Azonosító	Típus	Jelleg
A lista első eleme	ELSO	LANCELEM2-re mutató	I, O
A lista utolsó eleme	UTOLSO	LANCELEM2-re mutató	I, O
A törlendő listaelem	MIT	LANCELEM2-re mutató	I

Dinamikus adatstruktúrák

```
/* Elem törlése egy kétirányban láncolt listából */
LISTAROL2(ELSO,UTOLSO,MIT)
/* Kikapcsolás */
if (MIT=ELSO) AND (MIT=UTOLSO)
      /* Egyetlen elem */
      ELSO ← NIL
      UTOLSO ← NIL
else if MIT=ELSO
      /* Első, de nem egyetlen */
      (*(*MIT).KOVETO).ELOZO ← NIL
      ELSO ← (*MIT).KOVETO
...
```

Dinamikus adatstruktúrák

```
...
else if MIT=UTOLSO
     /* Utolsó, de nem egyetlen */
     (*(*MIT).ELOZO).KOVETO ← NIL
     UTOLSO ← (*MIT).ELOZO
else
     /* Belső elem */
     (*(*MIT).ELOZO).KOVETO ← (*MIT).KOVETO
     (*(*MIT).KOVETO).ELOZO ← (*MIT).ELOZO
/* Megszüntetés */
FELSZABADIT(MIT)
```

Dinamikus adatstruktúrák

- **Feladat**: Tervezzünk adatstruktúrát egy tárgymutató memóriában való tárolására!

Dinamikus adatstruktúrák

- **Feladat**: Tervezzünk adatstruktúrát egy tárgymutató memóriában való tárolására!

- **Típus**

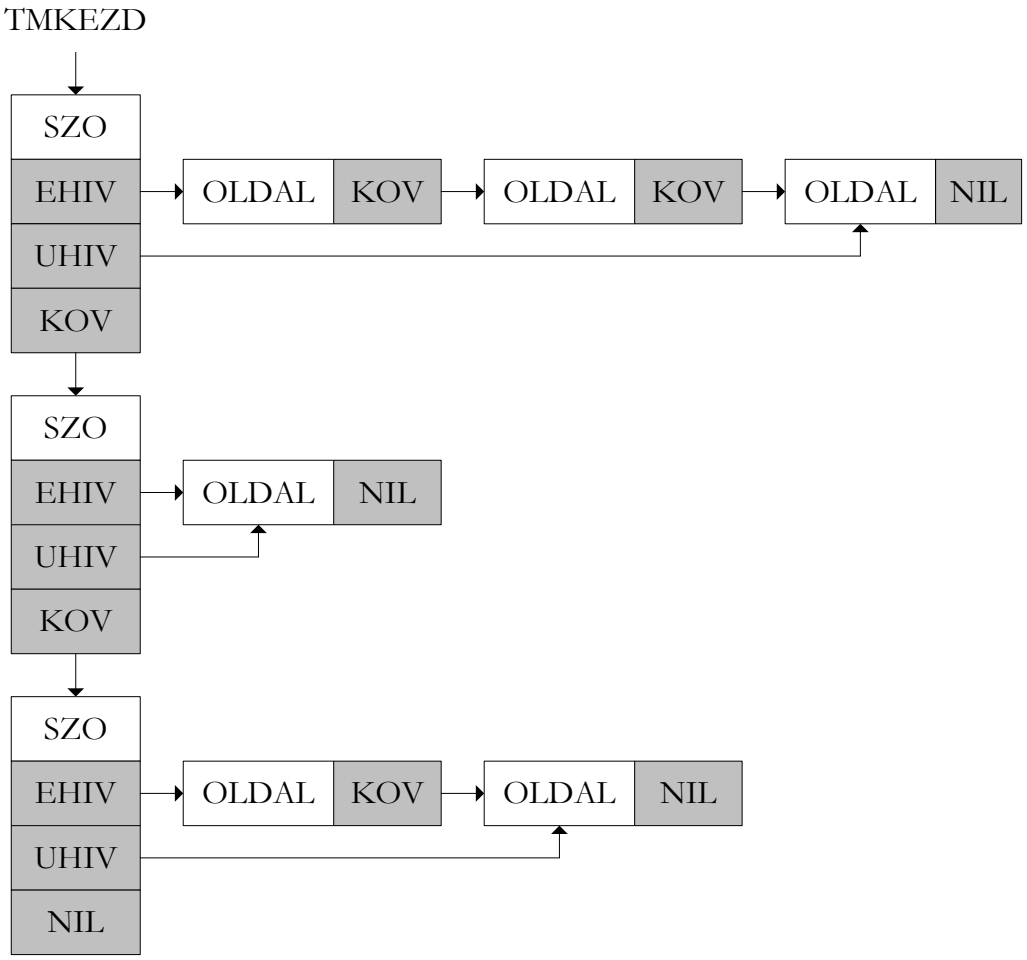
HIVREK Rekord

OLDAL	Egész
KOV	HIVREK rekordra mutató

SZOREK Rekord

SZO	Sztring
EHIV, UHIV	HIVREK rekordra mutató
KOV	SZOREK rekordra mutató

Dinamikus adatstruktúrák



Tárgymutató felépítése

Dinamikus adatstruktúrák

- **Feladat**: Írjuk ki a tárgymutató adatait a képernyőre!

Funkció	Azonosító	Típus	Jelleg
A tárgymutató kezdete	TMKEZD	SZOREK-re mutató	I
Az aktuális szó rekord	AKT	SZOREK-re mutató	M
Az aktuális hivatkozási rekord	AKTHIV	HIVREK-re mutató	M

Dinamikus adatstruktúrák

```
/* Tárgymutató kiírása a képernyőre */
KIIR(TMKEZD)
AKT ← TMKEZD
while AKT<>NIL
    Ki: (*AKT).SZO
    AKTHIV ← (*AKT).EHIV
    while AKTHIV<>NIL
        Ki: (*AKTHIV).OLDAL
        AKTHIV ← (*AKTHIV).KOV
    /* Esetleges soremelés, billentyűleütésre várás */
    AKT ← (*AKT).KOV
```