PROGRAMOZÁSMÓDSZERTAN
1. ELŐADÁS'2005
(VÁZLAT)

Megjegyzés [SzP1]: Módszeres programozás – Programozási bevezető (*ulógia* 18) [55-67, 68-74]
Módszeres programozás – Adattípusok (*ulógia* 34) [10-16, 17-35]

1. ADATOK JELLEMZŐI

1.1. Azonosító

Az a ***jelsorozat***, *amellyel hivatkozhatunk a tartalmára, amely által módosíthatjuk tartalmát.*
Például:

i, j, n, a – *változó*k nevei (szimbolikus nevek)
π, -128, 3.14, "Eredmény=", IGAZ – *konstans*okat azonosító jelsorozat

Megjegyzés: **x := x+1**

*cím*hivatkozás *érték*hivatkozás

vagy

Feldolgoz(x) – *eljáráshívásnál nem látszik, hogy x az adat címét vagy értékét jelenti*
Eljárás Feldolgoz(**Konstans** x:TX)*:* – *Ez már világos beszéd!*

1.2. Hozzáférési jog

Adatokat ***módosítani***, illetve értéküket ***lekérdezni***, használni lehet; eszerint egy adat hozzáférés szempontjából négyféle lehet:

	módosítható	**lekérdezhető**	
Független	***Nem***	***Nem***	***Független***
Input	**Nem**	**Igen**	**Konstans**
Output	**Igen**	**Nem**	**Virgin**
I/O	**Igen**	**Igen**	**Változó**
Háttértár-adat			**Memória-adat**

1.3. Kezdőérték

A ***születéskor*** ***hozzárendelt érték***. Változóknál deklarációban kap*hat* értéket, vagy eleve *van* típushoz rendelt kezdőérték, esetleg speciális '*nem definiált*' érték, s így akkor mód van hivatkozás ellenőrzésre is (*virgin*)!

1.4. Hatáskör

A ***programszöveg*** ***azon tartománya, amelyben az adathoz a hozzáférés lehetséges***.

Gondoljunk a blokkstruktúrájú programozási nyelvek egymásba ágyazódó adatdeklarációira! Így beszélhetünk: *globális* és *lokális* (sőt *saját*) adatokról.
A hatáskört „színezi” a **láthatóság** kérdése: ha egy adat azonosítója megegyezik egy olyan adatéval, amelynek hatáskörébe esik, akkor azt elérhetetlenné teszi a saját hatáskörében, hiszen az adott név alatt neki van elsősége.

Példa:

```
…
Változó
  x:TX
Konstans
  y:TY(…)
…
  [itt az x TX, az y TY típusú]
…
Eljárás E1(Változó x:TX2):
  Változó
    y:TY2
  [itt az x TX2, az y TY2 típusú]
  …
Eljárás vége.
…
  [itt az x TX, az y TY típusú]
…
```

1.5. Élettartam

A ***futási idő*nek az az intervalluma, amelyben az adat azonosítója mindvégig ugyanazt az objektumot jelöli**.
Meggondolandó a hatáskörrel való kapcsolat! (Globális, lokális adatok hatásköre és élettartama, továbbá a „dinamikus”, azaz futásközben a programozó döntése szerint születő és megszűnő adatok. L. később!)

1.6. Értéktípus

Az adatoknak az a tulajdonsága, hogy ***értékei*** mely **halmaz***ból* származnak (*értékhalmaz*) és **tevékenységek***nek* (eljárások, függvények, operátorok) *mely* „***készlete***, amely *létrehozza*, *felépíti*, *lerombolja* és *részekre bontja*”, alkalmazható rá (*asszociált műveletek*).

2. AZ ÉRTÉKTÍPUS

2.1. Az értéktípusról általánosságban

Összetettség (strukturáltság) szempontjából:

- *skalár* (vagy *strukturálatlan*)
- *összetett* (más szóval *strukturált*)

A*datstrukturálási módok*:

Strukturálási mód		→	Adatszerkezet
Keresztszorzat	A×B…	→	Rekord
(Megkülönböztetett) egyesítés	A×B×…(C×D∪E×F∪…)	→	Alternatív rekord
Halmazképzés	2^A	→	Halmaz
Iterált-képzés	A^*	→	Sorozatfélék

2.2. Az asszociált műveletek osztályozása

Az asszociálható műveleteket csoportosítjuk:

- *értékadás* (azonos típusúak közötti adatmozgatás: másolatkészítés, értékmegosztás),

  Példa:

```
…
Típus
  TPont=Rekord(x,y,z:Valós)
Változó
  Origó,Egys:TPont
…
  Eltol(Origó,Egys)
…
Eljárás Eltol(Változó p:TPont;Konstans Δ:TPont):
  [értékmegosztás jön létre a p és az aktuális
   paraméterpárja között, Δ-ba az aktuális para-
   méterpár másolata kerül]
  p.x:+Δ.x; p.y:+Δ.y; p.z:+Δ.z
  [„tiszta” értékmásolások]
Eljárás vége.
```

- *típusátviteli függvények:*
  - *Konstrukciós műveletek* (strukturált érték létrehozása)

    Példa:

```
…
Típus
  TPont=Rekord(x,y,z:Valós)
  TVárakozók=Sor(TEmber)   [l. később]
  TPMut=TPont’Mutató   [l. később]
Változó
  p,q:TPont
  v:TVárakozók
  pm:TPMut
…
  p:=TPont(1.0,0.0,0.0)
  Üres(v)   [l. később]
  Létrehoz(pm,p); q:=TPMut(pm)   [l. később]
…
```

- *Szelekciós operációk*

  Példa:

```
…
  hossz:=SQRT(p.x^2+p.y^2+ p.z^2)
  vevő:=Sorból(v)   [l. később]
…
```

- *Azonosság* és más *relációk*

  Példa:

```
…
Típus
  TNap=(hétfő,kedd,szerda,csütörtök,péntek,
        szombat,vasárnap) [l. később]
Konstans
  napSzám:Egész(Számosság'TNap) [l. később]
Típus
  TNapNév=Tömb(1..napSzám:Szöveg)
Konstans
  SNapHu:TNapNév=
         ('hétfő','kedd','szerda','csütörtök',
          'péntek','szombat','vasárnap')
  SNapEn=
         ('Monday','Tuesday','Wednesday',
          'Thursday','Friday',
          'Saturday','Sunday')
Változó
  nN:TNapNév
  mn:TNap
  c:Karakter
…
  Ha c='E' és nN=SNapHu akkor nN:=SNapEn
…
   Ciklus amíg mn≤péntek
…
```

- *Számosság-függvény*

  Példa:

```
…
Típus
  TNap=(hétfő,kedd,szerda,csütörtök,péntek,
        szombat,vasárnap)
Konstans
  napSzám:Egész(Számosság'TNap)
  SNap:Tömb(1..napSzám:Szöveg)=
       ('hétfő','kedd','szerda','csütörtök',
        'péntek','szombat','vasárnap')
…
```

- *Min- és Max-függvény*, amelyek értelemszerűen csak *rendezett típusok* esetén léteznek.

  Példa:

```
…
Típus
  TNap=(hétfő,kedd,szerda,csütörtök,péntek,
        szombat,vasárnap)
Konstans
  SNap:Tömb(Min'TNap..Max'TNap:Szöveg)=
       ('hétfő','kedd','szerda','csütörtök',
        'péntek','szombat','vasárnap')
…
```

- *Sorszám- (vagy Rend-) függvény*

  Példa:

```
…
  első:=Sorszám(hétfő) [első:=0]
  utsó:=Sorszám(vasárnap) [utsó:=6]
  A_Kód:=Sorszám('A') [A_Kód:=65]
…
```

- *Be/Ki műveletek* (konverzió az input/output és belsőábrázolás között)

  Példa:

```
…
Változó
  nap:TNap
…
  Be: nap   [nap<Max'TNap]
  Ki: Következő(nap)
…
```

- *Transzformációs* (a típuson értelmezett, a típusra képező függvények)

  Példa:

```
…
Típus
  TPont=Rekord(x,y,z:Valós)
Változó
  kövNap,nap:TNap
  p,q:TPont
…
  kövNap:=Következő(nap)
  táv:=Hossz(p)-Hossz(q)  [Hossz a programozó
                           által definiált fv.]
…
```

- *egyéb függvények.*

3. EGYSZERŰ ADATTÍPUSOK

Alábbiakban definiáljuk az „eredendően” szerkezetnélküli ún. ***skalár*** típusokat, megadva ezek *értékhalmaz*át, a hozzátapadó *művelet*ek, *reláció*k halmazát. Két típus „lóg ki” ezek közül. A szöveg (string) és a mutató típus. A ***szöveg típus***t azért soroljuk ide, mert a programozási nyelvek majd mindegyike fölkínálja, s így hozzátartozik az ún. *elemi* (natív) *típusok*hoz, másrészt abban jócskán eltér a későbbiekben tárgyalt összetettektől (s így oda még kevésbé illik), hogy elemeinek típusa nem választható meg szabadon. A ***mutató*** típus valóban szerkezet nélküli, hisz értékhalmaza a memóriacímek egy részhalmaza (ezért tárgyaljuk itt), de az is kétségtelen, hogy oly szorosan tapad valamely összetett típushoz, amely címét tartalmazza, hogy anélkül értelme sem lenne bevezetni.

A tárgyalt típusokról a következőket adjuk meg:

- ***Értékhalmaz***
- ***Kezdőérték***, amit az ilyen típusú objektum létrejövetelekor kap.[1]
- ***Asszociált műveletek***
- (s esetleg) tipikus ***ábrázolása***(i)
- (s néhány esetben) ***példa***.

A rövid leírás kedvéért időnként alkalmazni fogjuk a ***Típ*** jelölést az éppen tárgyalt típusra való hivatkozásra. (Értelemszerűen akkor, amikor a típus nevét a programozó feladata megadni.) Származtatott típus esetén pedig gyakorta a ***TB*** az ún. *bázistípus*ra (amelyből kiindulunk) utal.

3.1. Elemi adattípusok

3.1.1. Egész

Értékhalmaz	-32768**..**32767 $\subset$ **Z** (*Min'Egész***..***Max'Egész*)
Kezdőérték	**0**
Műveletek	**+**, **-**, *****, **Div** (egészosztás), **Mod**, **-** (unáris mínusz), **^** (pozitív egészkitevős hatványozás) **:=** **=**, **<**, **≤**, **≥**, **>**, **≠** **Be:**, **Ki:**
Ábrázolás	ún. *kettes komplemens kódú*

Megjegyzés:

A programozási nyelvek többféle értékhalmazzal is felkínálnak efféle típusokat. Pl. a Turbo Pascal megkülönböztet BYTE, SHORTINT (8 bites); INTEGER, WORD (16-bites); LONGINT (24 bites)… Ennek ellenére nem tartjuk fontosnak az algoritmikus

[1] És most legkevésbé sem hagyjuk magunkat befolyásolni olyan programozási nyelvektől, amelyben a változók létrejöttét nem kíséri automatikus kezdőértékadás.

nyelvben ezeket megkülönböztetni, annál is inkább szükségtelennek gondoljuk, mert ha kell a „szélsőséges értékeire” tudunk hivatkozni a Min’Egész és Max’Egész típusfüggvénnyel anélkül, hogy letennénk a voksunkat bármelyik konkrét értékhalmaz mellett.

3.1.2. Valós

Értékhalmaz	???..??? ⊂ **R** (*Min’Valós*..*Max’Valós*)
Kezdőérték	**0.0**
Műveletek	**+** , **-** , ***** , **/**, **-** (unáris mínusz), **^** **:=** **=**, **<**, ≤, ≥, **>**, ≠ **Be**:, **Ki**:
Ábrázolás	ún. *lebegőpontos ábrázolás* (pontosabb lenne, ha e típust *racionális*nak nevezzénk, mert csak racionális számot képes ábrázolni), vagy ún. *pakolt decimális* ábrázolás

Megjegyzések:

Vegyük észre, hogy ugyanazok a műveleti jelek most –ha hasonló jelentéssel is, de– mégsem egészen ugyanazzal a jelentéssel bírnak. (Polimorfizmus.)
Itt már föl sem vállaltuk az értékhalmaz pontosítását, mivel ez sokkal inkább implementáció-függő, mint az egész számoké.

3.1.3. Logikai

Értékhalmaz	*Hamis*..*Igaz* ⊂ **L** (*Min’Logikai*..*Max’Logikai*)
Kezdőérték	**Hamis**
Műveletek	**nem**, **és**, **vagy** **:=** =, <, ≤, ≥, >, ≠ **Be**:, **Ki**:
Ábrázolás	0 ≡ *Hamis*, –1 ≡ *Igaz* – azaz (valamely) egész típusra visszavezetés; néha 1 ≡ *Igaz* – ekkor 1 bites az ábrázolás

3.1.4. Karakter

Értékhalmaz	0**..**255 -*kód*ú jelek (*Min'Karakter*..*Max'Karakter*)
Kezdőérték	**Min'Karakter**
Műveletek	**Karakter(.)** – **Karakter**: Egész→Karakter **Sorszám(.) – Sorszám**: Karakter→Egész (belső kód) **:=** =, <, ≤, ≥, >, ≠ **Be**:, **Ki**:
Ábrázolás	Valamely kódrendszer szerinti kód, mint előjelnélküli szám. (*Fix bitszámú kód*.)

Megjegyzés:

Sokfajta kód rendszer létezik (legismertebbek: ASCII, UNICODE). Többségük fix bitszámú (ASCII 8 bites, UNICODE 16 bites), de elképzelhető változó bitszámmal dolgozó is. (L. Huffman-kódolás.)

3.1.5. (Absztrakt)Felsorolástípus

Értékhalmaz	**(**konstans$_1$**,** konstans$_2$**, ... ,** konstans$_N$**)** (*Típ* jelölje a típus azonosítóját) (*Min'Típ*=konstans$_1$**..***Max'Típ*=konstans$_N$)
Kezdőérték	**Min'Típ** vagy **NemDef**
Műveletek	**Következő**(Típ-*típusbeli kifejezés*)**, – Következő**: Típ→Típ **U** {NemDef} **Előző(**Típ-*típusbeli kifejezés*)**, – Előző**: Típ→Típ **U** {NemDef} **Sorszám(**Típ-*típusbeli kifejezés*)**, – Sorszám**: Típ→Egész **Típ(***egész típusú kifejezés*)**, – Típ**: [0..**Sorszám(Max'**Típ)]→Tip **:=** =, <, ≤, ≥, >, ≠ (a *felsorolás sorrendje* egyben *rendezés* is) **Be**:, **Ki**:
Ábrázolás	A minimálisan szükséges bitszámú (≈log$_2$(**Számosság'**Típ)) kód, mint előjelnélküli szám.

Megjegyzések:

Az értékhalmazban szereplő ismétlődés nélküli, szimbolikus nevek (a típus konstansai) ***nem lehetnek más típus*** (pl. egy másik felsorolástípus) ***értékhalmazában***, ez a feltétel amiatt szükséges, mert a fordítóprogram így mindig egyértelműen el tudja dönteni a konstans „hovatartozását". Természetesen „…" nem szerepelhet a felsorolásban, mivel a fordítónak nincsenek „előzetes elképzelései" arról, hogy mik lehetnének ott a felsorolásban.

Mindazon típusokat, amelyek értékkészletét konstansainak egyszerű fölsorolásával adhatunk vagy adhatnánk meg ***diszkrét típus***nak hívjuk. Tehát ilyen az `Egész`, a `Logi-`

`kai`, a `Karakter`, de lehet bármilyen –a program írója által kreált– *absztrakt konstans*okat tartalmazó fenti *absztrakt felsorolástípus* is.

Példa:

```
Típus
  TNap=(hétfő, kedd, szerda, csütörtök, péntek, szombat,
        vasárnap)
Változó
  tegnap,ma,holnap: TNap
Konstans
  ünnepnap: TNap(vasárnap)
…
  Ha ma=Min'TNap akkor  tegnap:=Max'TNap
                 különben holnap:=Előző(ma)
  Ha ma=Max'TNap akkor  holnap:=Min'TNap
                 különben holnap:=Következő(ma)
  első:=Sorszám(hétfő) [első:=0]
  utsó:=Sorszám(vasárnap) [utsó:=6]
…
```

3.1.6. (Rész)Intervallumtípus

Csak ***diszkrét*** típusból származtatható egyszerű típus. Jelöljük a bázistípust TB-vel.

Értékhalmaz	$konstans_1$..$konstans_2$ $\subseteq$ TB (*Min'Típ*=$konstans_1$..*Max'Típ*=$konstans_2$)
Kezdőérték	**Min'Típ** vagy **NemDef**
Műveletek	TB-vel megegyező műveletek (korlátozva persze az értékhalmazra)
Ábrázolás	TB-vel megegyező ábrázolás

Példa:

```
Típus
  TNap=(hétfő, kedd, szerda, csütörtök, péntek, szombat,
        vasárnap)
  TMunkanap=hétfő..péntek
  THétvége=szombat..vasárnap
Változó
  tegnap,ma,holnap: TNap
Konstans
  ünnepnap: TNap(vasárnap)
…
  Ha ma=Min'TNap akkor  tegnap:=Max'TNap
                 különben holnap:=Előző(ma)
  Ha ma=Max'TNap akkor  holnap:=Min'TNap
                 különben holnap:=Következő(ma)
  első:=Sorszám(hétfő) [első:=0]
  utsó:=Sorszám(vasárnap) [utsó:=6???]
…
```

Megjegyzés [SzP2]: A Sorszám(vasárnap) melyik típushoz asszociált függvény a vasárnap helyen? (a TNap-é vagy a THétvégé-é)

Megjegyzések:

1. Érdekes anomáliára vezet a `Sorszám` és `Típ` típuskonstrukciós függvény következetes bevezetése. Miért?
2. Ha a diszkrétségtől eltekintünk, kiterjeszthető a `Valós`akra is az intervallumtípusképzés. Ez persze csak azzal a többlettel képzelhető el, hogy egy lépésközt is megadunk a származtatáshoz (amelyre vannak persze elvárások).
3. További általánosítás lehetséges: nem első és utolsó elem által meghatározott része egy értékhalmaznak, hanem valamilyen (*predikátum*mal definiált) tulajdonságnak eleget tevő elemeinek részhalmaza.

Példa:

```
Típus
  TTermSzám=Egész
            [Típusinvariáns: i:TTermSzám : i>0]
  TPrímek=TTermSzám
          [Típusinvariáns: n:TTermSzám : Prím?(n)]
          Függvény Prím?(Konstans x:Egész):Logikai
            …
          Függvény vége.
…
```

3.1.7. Szövegtípus

Értékhalmaz	MaxHossz darabnyi jelből álló karakterláncok halmaza $\subset$ **Karakter*** (*Min'Szöveg*..~~*Max'Szöveg*~~) – *alfabetikus* rendezés szerinti első (=üres szöveg); az utolsó erősen reprezentáció-függő, ezért nem definiáljuk.
Kezdőérték	**''** azaz üres-szöveg (**Min'Szöveg**)
Műveletek	**+**, **Hossz(.)**, **Balrész(.)**, **Jobbrész(.)**, **Jele(.)** **:=** =, <, ≤, ≥, >, ≠ **Be**:, **Ki**:
Ábrázolás	**Rekord**(hossz:Egész, jel:**Tömb**(1..MaxHossz:Karakter)) – Pascal-stílusú Karakter* × SzövegVégJel – C-stílusú

3.2. Mutató típusok

Az alcímbeli többes szám jogos, mert valójában tetszőleges (többnyire összetett) típushoz, mint bázistípushoz (`TB`) szervesen tartozhat egy-egy ilyen típus. Egy konkrét mutató típusú objektum ***csak egyfajta*** (nevezetesen `TB`-típusú) ***elemek kezdőcímei***t hordozhatja. E szigorúság oka: az ellenőrizhetőség, biztonságosság.

Értékhalmaz	Memóriacím, amely valamely TB-típusú elem kezdőcíme, vagy **Sehova** (rendezésnek nincs értelme ⇒ *Min'Típ, Max'Típ*)
Kezdőérték	**Sehova**
Műveletek	**Lefoglal(Változó** m:Típ, **Konstans** e:TB**)** – az eljárás létrehoz a memóriában egy TB-elemet, amelynek értéke éppen 'e', és a *cím*ét teszi 'm'-be; ha nincs elegendő hely, akkor 'm'-be **Sehova** érték kerül. Elhagyható a kezdőértéket definiáló paraméter, ekkor a TB-beli *iniciális* értékű elem keletkezik. **Típ(Konstans** m:Típ**)**:TB – a függvény az 'm'-beli címnél kezdődő TB-elemet adja vissza értékként **Felszabadít(Változó** m:Típ**)** – az eljárás felszabadítja a memória 'm'-ben lévő címtől kezdődő TB-elemnyi tartományát, majd 'm'-be a **Sehova** érték kerül. **:=** **=, ≠, <, >, ≤, ≥** **~~Be:, Ki:~~**
Ábrázolás	Memóriacím

Megjegyzések:

A `Lefoglal` művelet fent taglalt szemantikája is indokolja a szigorú típusosság kívánalmát!

A Pascal-beli `Lefoglal` művelet, a `New`, nem foglalkozik kezdőértékadással, sőt a helyfoglalás sikertelenségét sem közli a `Sehova`, vagyis a `Nil` értékkel. A `Felszabadít`, `Dispose` művelet sem törli a mutatót.

Példa:

```
…
Típus
  TBlokk=Tömb(1..MaxM:TElem)
  TBMut=TBlokk'Mutató
  TBlokkok=Tömb(1..MaxN:TBMut)
Változó
  e:TElem;    i:Egész
  b:TBlokk;   bk:TBlokkok;   bm:TBMut
…
  Ciklus i=1-től MaxN Div 2-ig
    Lefoglal(bm)  [bm-be kerül a lefoglalt TBlokk-nyi
                   terület címe, és a bm-nél kezdődő
                   tömb elemei TElem-kezdőértékűek]
    bk(i):=bm     [bk i. eleme a dinamikusan lefoglalt
                   tömb címe; értékmegosztás]
    … [a bk(i) tömbbe értékek kerülnek] …
  Ciklus vége
  b:=TBlokk(bk(1))  [az első blokk b-be]
  bm:=bk(1)         [az első blokk címe bm-be]
```

```
  Ciklus i=1-től MaxM-ig
    TBlokk(bm)(i):=e  [TBlokk(bm) bm-nél kezdődő blokk
                       mint tömb,
                       TBlokk(bm)(i) a tömb i. eleme]
  Ciklus vége
  [vajon milyen értékek vannak a bk(2..MaxN) elemekben;
   és mit mondhatunk a TBlokk(bk(2..MaxN)) értékéről?]
…
```

4. ÖSSZETETT ADATTÍPUSOK – TÍPUSKONSTRUKCIÓK

Helyesebb összetett típusok helyett ***típuskonstrukciók***ról beszélni, mivel valahány összetevő típusból hozunk létre, konstruálunk egy újat; s az ilyeneket létrehozni képes eszközöket ***típuskonstrukciós eszközök***nek hívni.

Többnyire *nem* magától értetődő az ilyen struktúrák *rendezése*, ezért sem a `Min'`, sem a `Max'` típusfüggvényt nem jelezzük. (Megjegyezzük: természetesen nem elképzelhetetlen – sőt!–, hogy utólag valamilyen rendezést rájuk is definiáljuk. Erről később még szó esik.)

A korábbi szokásos mondanivalók mellé bekerül a „konstrukció”, amelyben tisztázzuk az alkalmazás szintaxisát.

Rövidítés miatt a bázistípusokat sorszámozzuk és így jelöljük: TB_1, TB_2, …

4.1. Összetett típusok osztályozásai

A bázistípusok sokfélesége szerint –

- *Heterogén* (rekord, alternatív rekord)
- *Homogén* (sorozatfélék: tömb, lista…, halmaz; rekurzív típusok: lista, fa; gráf…)

Rákövetkezési reláció az elemei között –

- *Nincs* (értelmetlen: rekord, alternatív rekord; lehetne, de nincs: halmaz)
- *Egyértelmű*, kivéve a szélső elemeket (sorozatfélék: tömb, lista,…)
- *Többszörös* (rekurzív típusok közül a fa; gráf)

4.2. Rekord

Konstrukció	**Rekord(**mező$_1$**:** TB$_1$**,** mező$_2$**:** TB$_2$ …**)**
Értékhalmaz	TB$_1$×TB$_2$×…
Kezdőérték	Az egyes komponensekhez tartozó kezdőérték.
Műveletek	**Típ(Konstans** m$_1$**:** TB$_1$**,** m$_2$**:** TB$_2$ …**)** – létrejön egy Típ típusú konstans m$_1$, m$_2$… mezőértékekből (konstrukciós operáció) objTip**.mező$_i$**:TB$_i$ – a „szokatlan” szintaxisú művelet értéke az adott Típ típusú objektum (objTip) mező$_i$ mezőjének értéke… (szelekciós operáció) objTip**.mező$_i$**:TB$_i$**:=**e$_i$ – a „szokatlan” szintaxisú értékadás eredménye az adott Típ típusú objektum (objTip) mező$_i$ mezőjének értékül adja e$_i$-t… **:=** **=**, **≠** **Be**:, **Ki**:
Ábrázolás	A mezők folytonos memóriaterületre képezve, a felsorolás sorrendjében.

4.3. Alternatív rekord

Olyan rekordféléről van szó, amelynek valamely mezőjétől (mezőitől) függ további mezőinek típusbesorolása.

Konstrukció	**Rekord(** mező$_1$**:** TB$_1$**,** … mező$_K$**:** TB$_K$**,** **Alternatívák** felt$_1$(mező$_1$,…) **esetén (**mező-k sorozata**),** … felt$_m$(mező$_1$,…) **esetén (**mező-k sorozata**)** **Alternatívák vége)**
Értékhalmaz	TB$_1$×…TB$_K$×…∪$_{(i=1..m)}$TB$_{i,1}$×…TB$_{i,k_i}$
Kezdőérték	A nem egyértelműsége miatt **NemDef**.
Műveletek	A rekordhoz hasonlóan.
Ábrázolás	A mezők folytonos memóriaterületre képezve, a felsorolás sorrendjében, a hosszat a leghosszabb alternatívával számolva.

Példa:

```
…
Konstans
  ffi: Egész(1)
  nő : Egész(2)
Típus
  TDátum=...
  TNem=ffi..nő
```

```
  TKontroll=Függvény(TNem,Dátum,0..999):0..9
  Függvény Kontroll(Konstans n:TNem
                            d:TDátum
                            x:0..999):0..9
    [Kontroll: egy TKontroll típusú konkrét függvény]
    …
  Függvény vége.
Típus
  TSzemSzám=Rekord(
    nem:TNem
    szülidő:TDátum
    sorszám:Egész
    ellenőr:TKontroll [itt csak olyan fv-típus képzelhető
                       el, amely értelmezési tartománya:
                       TNem×TDátum×Egész; értékkészlete
                       most nincs megkötve])
  TSzemély=Rekord(
    szsz:TSzemSzám
    név:Szöveg
    Alternatívák
      nem=nő  esetén (lnév :Szöveg)
      nem=ffi esetén (katsz:Egész)
    Alternatívák vége)
Konstans
  Jézus:TSzemély((ffi,
                  1.12.25,
                  123,
                  Kontroll) [szsz mező tartalma],
                  'Jézus Krisztus' [név mező],
                  (123456) [nem=ffi⇒katsz mező])
…
```

4.4. (Hatvány-)Halmaz

Csak *diszkrét* (többnyire nagyon is korlátozott számosságú) típusnak definiálhatjuk a hatvány-halmaz-típusát.

Konstrukció	**Halmaz(**TB**)** (*Min'Típ*=∅..*Max'Típ*=TB-értékhalmaz, a tartalmazás alapján rendezve)
Értékhalmaz	TB*
Kezdőérték	**Üres** halmaz
Műveletek	∈ (eleme), ∩ (metszet), ∪ (egyesítés), \ (különbség), ∅ vagy **Üres** (üres halmaz létrehozás), **Üres?** (logikai értékű függvény) **:=** =, < (⊂), ≤ (⊆), ≥ (⊇), > (⊃), ≠ **Be**:, **Ki**:
Ábrázolás	**Tömb**(TB:Logikai) – azaz bitvektor az adott elem tartalmazása vagy nem tartalmazása szerint; **Lista**(TB) – tartalmazott elemeinek felsorolása [l. később]

Példa:

```
…
Típus
  TÓra=0..23
  TNap=Halmaz(TÓra)
Változó
  ma,holnap:TNap
Konstans
  munkanap:TNap(7..12,14..20)
…
  Üres(ma)  [ma nincs kötött program ⇒ ma „szabad”]
  Ha Üres?(holnap) akkor ma:=munkanap
…
```

4.5. Tömb

Konstrukció	**Tömb(**TIndex**:** TElem**)**
Értékhalmaz	$\text{TElem}^{\text{Számosság(TIndex)}}$
Kezdőérték	TElem típus kezdőértékei
Műveletek	objTip**(**i**)** (a Típ típusú objTip tömb i. TElem típusú eleme), objTip**(**i**):=**e (a Típ típusú objTip tömb i. eleme legyen 'e' értékű) **:=** **=**, ≠ **Be**:, **Ki**:
Ábrázolás	Az elemek folytonos memóriaterületre képezve, növekvő index sorrendben. (L. később)

5. SOROZATTÍPUSOK

Sorozattípusnak hívjuk azt a típust, amely

1. ***homogén*** (azaz elemei egyetlen bázistípushoz tartoznak),
2. ***egyértelmű rákövetkezési reláció*** van elemei között (az adott elemét legfeljebb egy elem követheti),
3. ***egyetlen*** olyan eleme van, amelyet ***nem előz meg*** másik, s ***egyetlen*** olyan, amelyet ***nem követ***.

5.1. Sorozatműveletek

Nagyon sokféle sorozattípus alkotható meg, mégha nem különböztetjük meg a különböző bázistípusúakat egymástól (azaz a *strukturálisan azonosak*at). Jellegzetes művelet-együttest rendelve hozzájuk kapjuk a későbbiekben tárgyalt egyes, fontos alosztályait.
A legfontosabb műveletei, halmozva, a következők:

Művelet	Tevékenység-leírás
Üres	Létrehoz, elemek nélkül.
Létrehoz	Létrehoz, struktúrától függő elemekkel.
Üres?/Teli?	Ellenőrzi, hogy van-e eleme, ill. hogy bővíthető lenne-e?
ElemSzám	Hány eleme van?
Beilleszt	Struktúrától függő helyre új elemet illeszt.
Kihagy	Struktúrától függő helyről elemet hagy el.
Első/Utolsó	Első, utolsó elemének értékét adja.
Elejéről/Végéről	Leválasztja a sorozat első, utolsó elemét, értékét is visszaadja.
ElsőUtániak/UtolsóElőttiek	Eldobja az első, utolsó elemet.
Elejére/Végére	A sorozat első eleme elé, utolsó eleme mögé illeszt egy újat.
Elem	Struktúrától függően meghatározott elemének értékét adja vissza.

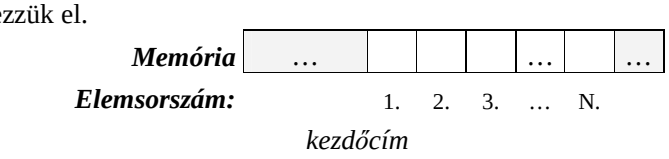
ElemMódosít	Struktúrától függően meghatározott elemének új értéket ad.
Elsőre/Utolsóra	A struktúra első, utolsó elem lesz az aktuális (ha volt ilyen).
Előzőre/Következőre	A struktúra aktuális eleme (ha volt ilyen) legyen az eddigit megelőző/követő.

Aszerint, hogy mely műveleteket engedjük meg egy sorozatfélénél beszélhetünk az alábbi típuskonstrukciókról. Nem jelöljük az „altípus-osztályokat”, továbbá a tevékenységeket nem a „hagyományos” nevükön említjük.

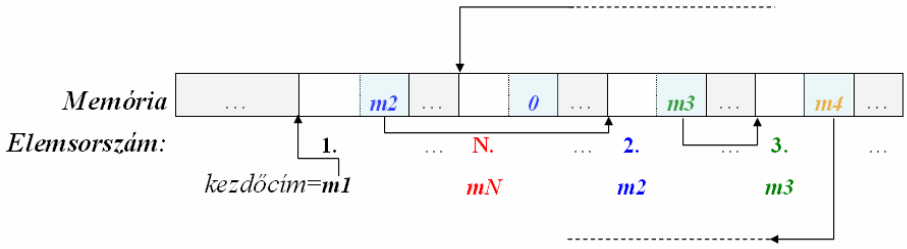
Típuskonstrukció	**Tevékenységhalmaz**
Tömb	(Létrehoz, ElemSzám,) Elem, ElemMódosít
Lista	Üres, Üres?, Teli?, Beilleszt, Kihagy, Elsőre, Utolsóra, Előzőre, Következőre, Elem, ElemMódosít
Sor	Üres, Üres?, Teli?, ElemSzám, Első, Elejéről, Végére
Verem	Üres, Üres?, Teli?, ElemSzám, Első, Elejére, Elejéről
Táblázat	Üres, Üres?, Teli?, ElemSzám, Elem, ElemMódosít…
InputSzekvenciálisFile	Létrehoz, Üres?, Elejéről
OutputSzekvenciálisFile	Üres, Üres?, Teli?, Végére
DirektFile	Üres, Létrehoz, Üres?, Teli?, ElemSzám, Elem, ElemMódosít…
AsszociatívFile	Üres, Üres?, Teli?, ElemSzám, Elem, ElemMódosít…

5.2. Sorozatok ábrázolásának lehetőségei

Folytonos ábrázolás, amikor az elemeket a memóriában a *kezdőcímtől szorosan egymásután* helyezzük el.



Láncolt ábrázolás, amelynél az elemek a memóriában elszórva helyezkednek el, a rákövetkezést mutatóval biztosítjuk.



Blokkolt ábrázolás ötvözi az előbbi kettőt úgy, hogy láncot hoz létre az elemek egy adott számú elemet tartalmazó tömbjéből.

TARTALOM