Egyszerű adattípusokra visszavezethető adattípusok, az összetett adattípusok

A sztring absztrakt adattípus

A sztring adattípus a karakter adattípusra támaszkodik. Karakterfüzért jelent, szöveget, ha úgy tetszik.

Definíció: Halmaz * halmaza és + halmaza

Egy A halmazhoz képezhetjük az A^* halmazt definíció szerint a következő módon. $A^* = A^0 \cup A^1 \cup A^2 \cup \ldots \cup A^n \cup \ldots$. Itt most $A^0 = \varepsilon$, ahol ε egy speciális szimbólum, az úgynevezett üres szimbólum. Az a szerepe, hogy ahol megjelenik, ott nem áll szimbólum. ($\varepsilon \notin A$). Az A halmaz A^+ halmaza az $A^+ = A^1 \cup A^2 \cup \ldots \cup A^n \cup \ldots$ halmaz.

Tulajdonképpen a * halmaz jelenti az A elemeiből képezhető összes, véges sok, egymás mellé írt szimbólumokból alkotott szimbólumláncok, szimbólumfüzérek, szimbólumsorozatok halmazát, megengedve azt a láncot is, amelyben nincs egyetlen szimbólum sem. A + halmaz annyiban kevesebb, mint a * halmaz, hogy az üres láncot nem tartalmazza.

Példa. Legyen B={0, 1}! Akkor $B^* = \{\varepsilon,0,1,00,01,10,11,000,001,010,100,011,101.110.111,\ldots\}$ az összes lehetséges véges sok bitből álló bináris jelsorozatok halmaza, beleértve az üres jelsorozatot is. $B^+ = \{0,1,00,01,10,11,000,001,010,100,011,101.110.111,\ldots\}$ ugyanaz, mint B^*, csak az üres jelsorozatot nem engedi meg.

Definíció: A sztring absztrakt adattípus

Legyen rögzítve egy X karakterek halmaza. A hozzátartozó S sztring absztrakt adattípus definíció szerint: $S = X^*$. A halmaz elemeit sztringnek nevezzük. Az ε neve üres sztring. A típus művelete a konkatenáció bináris művelet, amely a benne szereplő sztringek egymáshoz fűzését jelenti, az első végéhez hézagmentesen hozzáillesztjük a másodikat, így formálva az eredménysztringet. Jele: $\bullet$.

A sztring, mint összetett adattípus, rendelkezik *attribútum*mal (jellemzővel), amely az ő esetében a **Hossz** attribútum. A hossz attribútum egy szám, amely megadja, hogy a sztring hány karakterből áll. Az attribútumra hivatkozáskor az attribútum neve után szögletes zárójelbe tesszük az adatot, amelynek az attribútumáról szó van. Példa: Hossz[’vad’]=3, és Hossz[ε]=0. A sztring egyes karaktereire az indexükkel hivatkozhatunk. Amennyiben korlátozni szeretnénk a sztringet alkotó elemek számát, akkor az $A^{*,k}$, vagy $A^{+,k}$ formát használjuk jelezve, hogy tetszőleges hosszú lehet a sztring amennyiben *k*-nál nem hosszabb.

Példa. Legyen X=ASCII, legyen $S = X^*$, $x = 'vad' \in S$, $y = 'alma' \in S$, akkor $x \bullet y = 'vadalma' \in S$. Hossz['*vad*']=3, Hossz['*alma*']=5, Hossz['*vadalma*']=8, és Hossz[ε]=0. Az egyes karaktaerek: x_1=’v’, x_2=’a’, x_3=’d’.

A sztring adatstruktúra implementálása

Maradunk az X=ASCII esetnél, nem foglalkozunk a Unicode UTF-32, UTF-16, vagy UTF-8 realizálásával és annak nehézségeivel

A PASCAL implementáció. Egy lehetséges implemenráció a PASCAL nyelvben megtalálható változat, amelyben az ASCII karakterek mindegyike egy-egy byte-ot foglal el a memóriában és a sztring karakterei egymást közvetlenül követik a memóriabyte-okon. A sztring legelső byte-ja elé egy byte-ot helyezünk el, amelyben előjel nélküli egész számként tároljuk a sztring hosszát megadó számot. Ez a byte a sztring nulladik byte-ja. A sztring tényleges karaktereinek indexelése, sorszámozása 1-gyel kezdődik.
Példa:

Byte index	0	1	2	3	4	5
sztring	5	z	e	b	r	a
tartalom	05	7A	65	62	72	61

Az implementáció előnye: a sztring hossza a zérus indexű byte-ból azonnal kiolvasható.
Hátránya, hogy legfeljebb 255 karakterből állhat a sztring, mivel a hosszattribútum számára fenntartott byte-on nagyobb szám nem fér el.

Az ASCIIZ implementáció. Ilyen található a C nyelvben. A sztring egymást követő karaktereit a memória egymást követő byte-jain helyezzük el, majd a legutolsót követően egy zérusbyte-ot helyezünk el. A sztring karakterei között a zérusbyte tiltott. Az ASCIIZ megnevezésben a Z betű a zérusra utal.

Példa:

Byte index	0	1	2	3	4	5
sztring	Z	e	b	r	a	0x00
tartalom	7A	65	62	72	61	00

Az implementáció előnye, hogy a sztring tetszőleges hosszúságú lehet a memória adta korláton belül. Hátránya, hogy a zérusbyte ki van zárva a sztring karakterkészletéből, és időigényes lehet a sztring hosszának a meghatározása, mert meg kell keresni a zérusbyt-ot a sztring elejétől indulva. A karakterek indexelése zérussal kezdődik, erre a sztringet kezelő szoftver elkészítésekor oda kell figyelni.

FELADATOK

1. Ellenőrizze le, hogy a konkatenáció művelete kommutatív-e és asszociatív-e!

A sorozat absztrakt adattípus

Definíció: Sorozat absztrakt adattípus
Legyen adott egy A absztrakt adattípus. Az A típusra épülő sorozat adattípusnak nevezzük a $T=(A^n, M)$ adattípust, ahol $n \in \boldsymbol{N}$, rögzített szám. A sorozat adattípus esetében minden elemnek (az utolsót kivéve) van közvetlen rákövetkezője Köv(.) és (az elsőt kivéve) közvetlen megelőzője Elő(.). Az egyes elemekre az index megadásával (1,2,3,…) hivatkozunk. A sorozat attribútuma a Hossz, amely tulajdonképpen az *n* szám.

Példa: Legyen A=$\boldsymbol{Z}$, akkor az S=$\boldsymbol{Z}^{10}$ jelentése 10 darab egész szám, amely egymást követi elhelyezkedés szempontjából. Legyen az $\boldsymbol{x}$=(6, -2, 3, 0, 5, 11, -21, 43, 3, -7). Akkor x_1=6, x_2=-2, x_3=3, stb. Köv(x_5)=11=x_6, és Elő(x_5)=0=x_4. Látható, hogy a következő és az előző elem az index eggyel történő megnövelésével illetve csökkentésével elérhető. Köv(x_{10}), és Elő(x_1) nem létezik, meghatározatlan, nincs értelmezve. Az $\boldsymbol{x}$ neve gyakran *vektor*, ami utal arra, hogy az elemei megkülönböztetésére elegendő egyetlen indexet használni. Hossz[$\boldsymbol{x}$]=10. A

struktúra és az alapelem keveredése elkerülése végett célszerű a struktúrát félövér betűkkel jelölni. Tehát: $\boldsymbol{x}$=(6,-2, 3,0,5,11,-21,43,3, -7), Hossz[$\boldsymbol{x}$]=10, de x_1=6. Szokás ábrázolni a vektort elemeinek vízszintes irányú írásával (sorvektor), vagy függőleges irányú írásával (oszlopvektor). Az oszlopvektort tekintjük alapesetnek, a sorvektort transzponáltnak nevezzük és egy kis T betűt írunk felső indexbe a neve fölé. Példa:

$$\boldsymbol{x}^{\mathrm{T}} = \begin{pmatrix} -2, & 3, & 8 \end{pmatrix} \qquad \boldsymbol{x} = \begin{pmatrix} -2 \\ 3 \\ 8 \end{pmatrix}$$

Maga az alaptípus lehet szintén összetett. Például Legyen A=$\boldsymbol{Z}$, S=$\boldsymbol{Z}^4$, és $\boldsymbol{B}$= S^3. Akkor képezhetjük a T=($\boldsymbol{B}$,M) absztrakt adarttípust, amelyben az elemek maguk is összetettek. Ábrázolás szempontjából itt szerencsésebb a táblázatos forma. Legyen $\boldsymbol{b}$=($\boldsymbol{y}_1$, $\boldsymbol{y}_2$, $\boldsymbol{y}_3$), ahol $\boldsymbol{y}_1$, $\boldsymbol{y}_2$, $\boldsymbol{y}_3 \in$S, azaz például $\boldsymbol{y}_1$=(3,5,1,0), $\boldsymbol{y}_2$=(-1,10,2,1), y_3=(-8,2,1,1). Kényelmesebb megjelenítése $\boldsymbol{b}$-nek a táblázatos, amely táblázatot *mátrix*nak nevezünk. A táblázat olyan vektor, amelynek az elemei is vektorok. A mátrixot célszerű félkövér nagybetűkkel jelölni, az elemeit kicsikkel.

3	5	1	0
-1	10	2	1
-8	2	1	1

Ennek a táblázatnak lehet a neve $\boldsymbol{B}$ mátrix. Ilyenkor az egyes táblázatbeli elemekre két indexszel hivatkozunk. Például b_{11}=3, b_{12}=5, b_{21}=–1, stb. A kettős indexben az első a sorindex, a második az oszlopindex. A két indexet egymástól vesszővel (pontosvesszővel) választjuk el, de az elválasztás el is maradhat, ha nem okoz félreértést. Itt most Hossz[$\boldsymbol{B}$]=3, de Hossz[$\boldsymbol{y}_1$]=4. A mátrixhoz is hozzárendelhetünk egy kettős Hossz attribútumot, amely megadja a sorok és az oszlopok számát, azaz Hossz[$\boldsymbol{B}$]=(3, 4). Szokásos matematikai megjelenítése a mátrixnak a

$$\boldsymbol{B} = \begin{pmatrix} 3 & 5 & 1 & 0 \\ -1 & 10 & 2 & 1 \\ -8 & 2 & 1 & 1 \end{pmatrix}.$$

A sorozat adattípus egy lehetséges implementációja a *tömb*. A tömbben azonos típusú elemek követik egymást lineáris sorrendben. A memória kiválóan alkalmas olyan matematikai objektumok tárolására, mint a vektor vagy a mátrix. Ahogyan a vektor koordinátáit egymást követően soroljuk fel, úgy ezeket a tömbelemeket a memóriában is tárolhatjuk azonos tulajdonságú egymást követő rekeszekben. A vektor minket érdeklő koordinátájára az indexe alapján hivatkozunk. Az index és a vektor kezdő elemének címe alapján a keresett indexű elem *címe* (= a kezdőelem kezdőbyte-jának memóriabeli sorszáma) meghatározható. Feltesszük, hogy egy $\boldsymbol{v}$ vektor kezdőelemének indexe 1. Ha a a $\boldsymbol{v}$ vektor kezdőcíme, k a keresett elem indexe és h a rekeszméret byte-ban, akkor a keresett elem címe, jelölésben @(v_k), vagy, ha nem okoz félreértést, akkor röviden csak @v_k:

$$@(\mathrm{v}_k) = a + (k-1)\cdot h\,.$$

A cím @ jele (at jel) az angol address szóból származik, ami címet jelent.

Mátrixokat úgy szokás tárolni, hogy vagy a sorait (*sorfolytonos tárolási mód*), vagy az oszlopait (*oszlopfolytonos tárolási mód*) helyezzük egymás után.

15. Példa: Legyen egy 3 sor és négy oszlopból álló A mátrixunk

a_{11}	a_{12}	a_{13}	a_{14}
a_{21}	a_{22}	a_{23}	a_{24}
a_{31}	a_{32}	a_{33}	a_{34}

.

Sorfolytonos

a_{11}	a_{12}	a_{13}	a_{14}	a_{21}	a_{22}	a_{23}	a_{24}	a_{31}	a_{32}	a_{33}	a_{34}

Oszlopfolytonos

a_{11}	a_{21}	a_{31}	a_{12}	a_{22}	a_{23}	a_{13}	a_{23}	a_{33}	a_{14}	a_{24}	a_{34}

Legyen a mátrix mérete $m \times n$ (m sor és n oszlop) és az $(1,1)$ indexű elem (bal-felső sarokelem) címe legyen $@(a_{11})=a$. Sorfolytonos tárolási módot választva ekkor az (i,j) indexű elem címe a memóriában

$$@(a_{11}) = a + \left[(i-1)\cdot n + j - 1\right]\cdot h\,.$$

Ugyanez oszlopfolytonos tárolási mód mellett:

$$@(a_{11}) = a + \left[(j-1)\cdot m + i - 1\right]\cdot h\,.$$

A tömb implementálása során egy Méret attribútumot is rendelhetünk a struktúrához, ami azt mutatja meg, hogy az elemet tartalmazó rekeszben, mint tárolóegységben számolva mennyi memóriát foglaltunk le a tömb számára. Az egyes programozási nyelvek ezt meg is követelik. Amikor tömböt (indexes változót) deklarálunk, akkor meg kell adni azt a maximális memóriaterület méretet, amely megadja, hogy mekkora a maximális adatmennyiség esetére lefoglalt memóriaterület. Ez az érték nem lehet kisebb, mint a Hossz. Ha nagyobb, mint a Hossz, akkor a tömböt még lehet bővíteni további elemekkel a feldolgozás során.

Mi a különbség a sztring és a karaktertömb (karakter-vektor) között?

A sorozat adattípusnak létezik más realizációja és így implementációja is például a *láncolt lista*, amelyre azonban itt nem térünk ki.

FELADATOK

1. Adott egy tömb a memóriában, melynek elemei 80 byte-osak. A 32. tömbelem 17. byte-jának 6-os bitje hanyadik bit lesz a tömb elejétől számítva. (Ebben a feladatban a tömbelemeket is és a tömbelemeken belül a byte-okat is egytől indítva számláljuk balról-jobbra, a bitindex szokványos.) Adjon formulát ha az elemek *m* byte-osak, a tömbelem indexe *i*, az elemen belüli byte index *j*, és a byte-on belül a bitindex *k*!

2. Adott egy előjel nélküli egész számokat tároló, duplaszavas elemekből álló tömb a memóriában, amely egy alsó háromszög mátrixot tárol. Alsó háromszög mátrixnak nevezzük azt a négyzetes mátrixot, amelynek a főátlója feletti elemei garantáltan zérusok. A mátrixot sorfolytonosan tároljuk. A főátló feletti elemeket nem tároljuk, hiszen tudjuk, hogy azok zérusok. Adjon formulát az a_{ij} elem kezdő byte-ja tömbkezdethez képest relatív indexének kiszámítására! A relatív indexeket zérustól kezdve számláljuk. Ha a tömbelem nincs tárolva, az index legyen negatív (mondjuk -1). Feltételezzük, hogy az (i, j) indexpár az $n \times n$-es tömb valódi elemére mutat, azaz $1 \le i, j \le n$ teljesül.

3. Az előző feladatot oldjuk meg oszlopfolytonos tárolás esetére!

4. A 2.. feladatban szereplő mátrix legyen felső háromszög mátrix, azaz olyan négyzetes mátrix, amelynek a főátlója alatti elemek zérusok!

5. Az előző feladatot oldjuk meg oszlopfolytonos tárolás esetére!

6. A 2. feladatban szereplő mátrix legyen diagonális mátrix, azaz olyan mátrix, amelynek a főátlón kívüli elemei zérusok!

7. A 2. feladatban szereplő mátrix legyen tridiagonális mátrix, azaz olyan mátrix, amelynek a főátlón és a kétoldalról hozzá csatlakozó két mellékátlón kívüli elemei zérusok!

8. A 7. feladatban szereplő tridiagonális mátrixot ne sor vagy oszlopfolytonosan tároljuk, hanem először a főátló elemeit, majd a felső mellékátló elemeit és azt követően az alsó mellékátló elemeit rakjuk le egymást követően a bal felső végükön kezdve.

9. A 2. feladatban szereplő mátrix legyen szimmetrikus mátrix, azaz olyan mátrix, amelynél az (i, j) indexű elem megegyezik a (j, i) indexűvel! (Csak az alsó háromszög részt tároljuk! Hogyan változik a formula, ha csak a felső háromszög részt tároljuk?)

10. Legyen adott egy $m \times n$ -es mátrix számára lefoglalt terület. A mátrixot sorfolytonosan tároljuk. Ennek a mátrixnak a bal felső sarkában van egy $k \times l$-es méretű részmátrix. A részmátrix sorainak tárolása során a memóriában minden két sor között rekeszek maradnak ki, mert a terület egy nagyobb mátrix számára lett lefoglalva. Adjunk módszert a részmátrix elemeinek az eredeti mátrix területe elejéhez történő feltömörítésére úgy, hogy a részmátrix elemek sorfolytonosan, hézagmentesen helyezkedjenek el és ezáltal az eredeti mátrix korábbi területének a vége felszabadulhat, más célra használható legyen.

11. Oldjuk meg az előző feladat megfordításának a problémáját! Legyen a memóriában elhelyezve egy $k \times l$ -es mátrix sorfolytonosan és hézagmentesen. Bővítsük ki a mátrix által elfoglalt területet egy $m \times n$ -es mátrix számára megfelelő méretű területté! Ezek után szeretnénk az eredeti mátrixunkat az új mátrix bal felső sarkában lévő részmátrixnak tekinteni. Hogyan kellene az elemeket átmozgatni ezen célból?

A rekord absztrakt adattípus

Definíció: Rekord absztrakt adattípus
Legyenek adottak A_1, A_2,…,A_k absztrakt adattípusok. A rájuk épülő T=($A_1 \times A_2 \times \ldots \times A_k$, M) adattípust rekord absztrakt adattípusnak nevezzük, ahol $k \in \boldsymbol{N}$, rögzített szám. Itt olyan összetett adattípusról van szó, amelyben rögzített sorrendben általában különböző típusú adatelemek követik egymást. Ezen adatelemeket a rekord mezőinek nevezzük. Az egyes mezőkre hivatkozás a mező nevének leírásával és mögötte szögletes zárójelben a rekord nevének megadásával történik. A műveletek halmaza sokféle lehet az egyes mezőkkel végezhető műveletek sokfélesége miatt.

A rekordokban általában megkülönböztetünk *kulcsmező*(ke)t, amelyek alapján sok rekord között a minket érdeklő rekordot keressük. A kulcsmezőről gyakran feltesszük, hogy a rekordra nézve a tartalma egyedi, az a többitől megkülönbözteti, hogy az alapján lehessen keresni. A kulcsmező tartalmára, típusára nincsen komoly megkötés.

Példa. Legyen Név=ASCII^{*,20}, Születési_év=$\boldsymbol{N}$, és Neme={férfi, nő}. Legyen A=Név×Születési_év×Neme. Akkor T=(A,M) rekord típus. Ha $x \in A$, akkor Név[x] jelenti az x rekord elem Név mezejét, ami legfeljebb húsz ASCII karakterből álló sztring lehet, Születési_év[x] jelenti az x elem Születési_év mezejét, ami nemnegatív egész szám, Neme[x] jelenti a Neme mezőt, amely vagy a *férfi*, vagy a *nő* bejegyzést tartalmazhatja. Írhatjuk, hogy például az x személy rekordja x=(’Remek Elek’, 1965, *férfi*). A mezőket vesszővel választottuk el egymástól. De írható az is, hogy Név[x]= ’Remek Elek’, Születési_év[x]=1965, Neme[x]=*férfi*. Az előző formánál tudni kell a mezők neveit, pontosabban azok sorrendjét, ám az a típus megadásakor úgyis szerepel.

A rekord implementálása a mezőinek a memóriában történő egymást követő elhelyezésével valósítható meg a legegyszerűbben. A rekord tulajdonképpen általában egy objektumra (pl. személy) vonatkozó különféle típusú mezők felsorolása, ahol a mezőknek neve van. A rekordnak is lehet Hossz, vagy Méret attribútuma, de itt a típustarkaság miatt inkább byte-ban célszerű azt megadni. Rögzítenünk kell egy adott szituációban, hogy milyen egységben adjuk meg ezen attribútumokat.

A mutató absztrakt típus

Definíció: Mutató (pointer) absztrakt adattípus
A mutató absztrakt adattípus valamely adatra mutat a memóriában azáltal, hogy az adat memóriabeli címét (byte-sorszám) tárolja. Adattal együtt van értelme használni. Ha nem mutat adatra, akkor NIL mutatónak nevezzük. A mutatóhoz hozzátartozik az a típus is, amilyen típusú adatra mutat.

Vigyázni kell a mutatók használatával, mert ha ugyanazon memóriaterületre két mutató is mutat, és az egyik mutatón keresztül törlöm a terület tartalmát, netán fel is szabadítom azt a területet, akkor a másik mutató még oda mutat a semmire, amiből nagy programhibák származhatnak. Ha x is és y is mutató, és x mutat egy konkrét adatra, akkor elegendő annyit írni, hogy $y \leftarrow x$ és már y is ugyanarra az adatra mutat. Itt a ← jel arra utal, hogy az egyik memóriaterület tartalma átmásolódik egy másik területre. (Ha nevet adok neki, akkor a mutató is foglal helyet!) Ha van egy a-val jelzett adat, akkor az x mutatót az $x \leftarrow @a$ révén állíthatom rá az a adatra.

A mutató adattípust általában összetett adattípus esetén használjuk, mert általa nagy memóriaterületeket tudunk elérni. Persze lehet egyszerű típusokra is alkalmazni. Legyen például $\boldsymbol{A}$ valamilyen adattípus és $a \in \boldsymbol{A}$ ennek egy konkrét adata. Akkor $@a$ tulajdonképpen az a által elfoglalt adatterület kezdőbyte-jának a címe és a mutató ismeri ezen adattípus szerkezetét is (például tömb, vagy rekord) és ezáltal az attribútumait is. Ha az $@a$ által mutatott adatra vagyunk kíváncsiak, akkor arra, $[@a]$ révén hivatkozunk, ahol a szögletes zárójel a mutató által mutatott terület tartalmát jelenti. Ha ez összetett, akkor még indexelni is lehet (tömb esete), vagy mezőnevekre hivatkozni (rekord esete).

Például a rekordnál ismertetett példa esetében írhatjuk, hogy Neme[[@*x*]], ami eredményét tekintve megegyezik a Neme[*x*] által megadottal, de szerkezetét tekintve nem, hiszen az első esetben az *x*-hez előbb meg kell határoznunk az $@x$ mutatót, amely a $[@x]$ rekordra fog mutatni és ennek ismeretében a megnevezett mezőt kell kiemelnünk.

Tömb esetén írhatjuk, hogy $[@\boldsymbol{v}]_3$, ha a tömb hármas indexű elemére vagyunk kíváncsiak, egyébként ez azonos a v_3-mal.

Hogyan jelölnénk egy mezőt, amely egy rekordokból felépülő tömbnek egyik rekordjához tartozik?

Hogyan jelölnénk azt a tömbelemet, amely tömb egy rekord egyik mezeje?

És ha a rekordokból felépülő tömb egyik rekordjának adott mezeje egy tömb és annak valamely indexű elemére vagyunk kíváncsiak?