2. *Egyszerű osztályok II.*

1. Egy nxm-es négyzetrács alaprajzú labirintus i. sorának j. mezője vagy egy fal, vagy üres hely, vagy kincset tartalmaz, vagy szellemet. Kérdezhessük le a labirintus i. sorának j. mezőjéről, hogy megadott irányban (fel, le, jobbra, balra) tovább lépve falba ütközünk-e, szellemmel találkozunk-e, kincset tálálunk-e. Vezessük be a kincs begyűjtés műveletét is, amely a labirintus i. sorának j. mezőjéről törli a kincset, így az üres lesz.

Labirintus

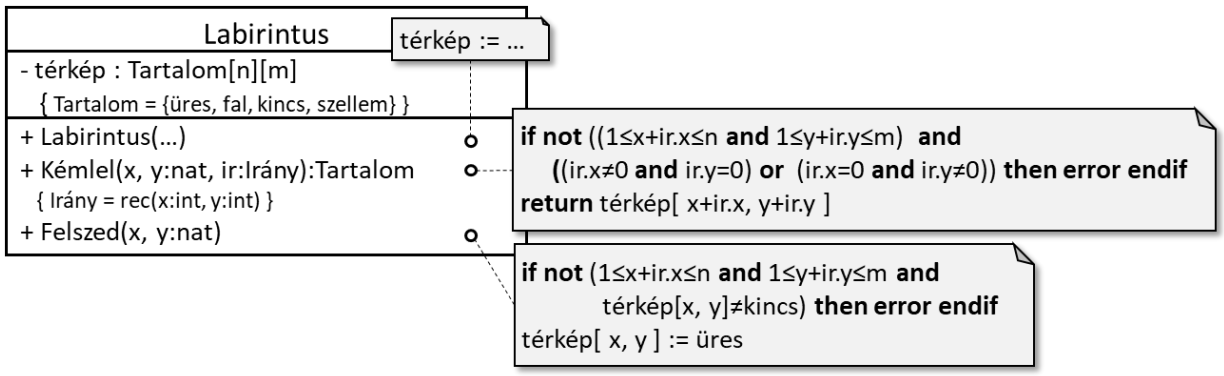
labirintusok	a) t := Kémlel(a, x, y, ir) (a:Labirintus, x, y:ℕ, ir:Irány, t:Tartalom) Irány = { fel, le, jobb, bal } Tartalom = { üres, fal, kincs, szellem }
	b) a := Felszed(a, x, y) (a:Labirintus, x, y:ℕ)
térkép : Tartalom$^{n\times m}$	a) **switch** (ir) **case** fel: **if** (x=1) **then error endif**; x := x-1; **case** le: **if** (x=n) **then error endif**; x := x+1; **case** bal: **if** (y=1) **then error endif**; y := y-1; **case** jobb: **if** (y=m) **then error endif**; y := y+1; **else error** **endswitch** t := térkép[x, y]
	b) **if not** (1≤x≤n **and** 1≤y≤m **and** térkép[x, y]≠kincs) **then error endif** térkép[x, y] := üres

A Kémlel() művelet egyszerűbb lesz, ha az irányokat koordinátapárok reprezentálják:

Irány=**rec**(x,y : [-1..+1]), ahol fel = (-1,0), le = (1,0), jobbra = (0,1), balra = (0-1).

Ekkor a paramétereket ellenőrző elágazás után elég a t := térkép[x+ir.x, y+ir.y] értékadás.

Nézzük meg ezt az osztálydiagramban:



A térkép adattag inicializálását a konstruktor végzi, amelynek paraméterként átadhatunk akár egy Tartalom[][] típusú mátrixot, akár annak a szöveges fájlnak a nevét, ahonnan beolvashatjuk a mátrix elemeit.

Másik modellt kapunk, ha önálló objektumokként tekintünk a labirintus pozícióira. Legyenek ezek olyan egészszám-párok, amelyekre értelmezzük a (koordinátánként vett) összeadást (Összead()). Ha az irányokat olyan speciális pozícióknak tekintjük, ahol -1 ≤ x, y ≤ 1, és az x , y közül pontosan az egyik 0 (ezt az Irány() művelettel ellenőrzihetjük), akkor egy pozícióhoz hozzáadva egy irányt a megfelelő szomszédos pozíciót kapjuk eredményül.[1]

Pozíció

pozíciók	c := Összead(a, b) (a,b,c : Pozíció)
	ok := Irány(p) (p:Pozíció, ok:𝕃)
x, y : ℤ	c.x, c.y := a.x+b.x, a.y+b.y
	ok := (-1 ≤ x ≤ 1) **and** (-1 ≤ y ≤ 1) **and** ((x=0 **and** y≠0) **or** (x≠0 **and** y=0))

Erre építve egyszerűsíthetjük a Labirintus definícióját. Cseréljük le a térkép típusát egy olyan gyűjteményre (úgynevezett asszociatív adatszerkezetre, jele: Map(Pozíció, Tartalom)), amelyben pozíciójukkal indexelve tároljuk a tartalmakat (Tartalom = {üres, fal, kincs, szellem}). A térkép[poz] a labirintus adott pozícióján levő tartalomra hivatkozik: az lekérdezhető, és megváltoztatható. Ki kell kötni azonban, hogy a térkép azokat és csak azokat a pozíciókat tartalmazza, amelyek koordinátái a labirintus határain belül, azaz 1..n és 1..m intervallumokba esnek.) Láthatjuk, hogy a műveletek programjai egyszerűbbek lesznek.

Labirintus

labirintusok	a) t := Kémlel(a, poz, ir) (a:Labirintus, poz:Pozíció, ir:Pozíció, t:Tartalom)
	b) a := Felszed(a, poz) (a:Labirintus, poz:Pozíció)
n, m : ℕ térkép : Map(Pozíció, Tartalom) Tartalom={üres, fal, kincs, szellem} Inv: a térkép pozíciói az összes 1..n×1..m-be eső számpár	a) **if not** Irány(ir) **then error endif** t := térkép[Add(poz, ir)]
	b) **if** térkép[poz]≠kincs **then error endif** térkép[poz] := üres

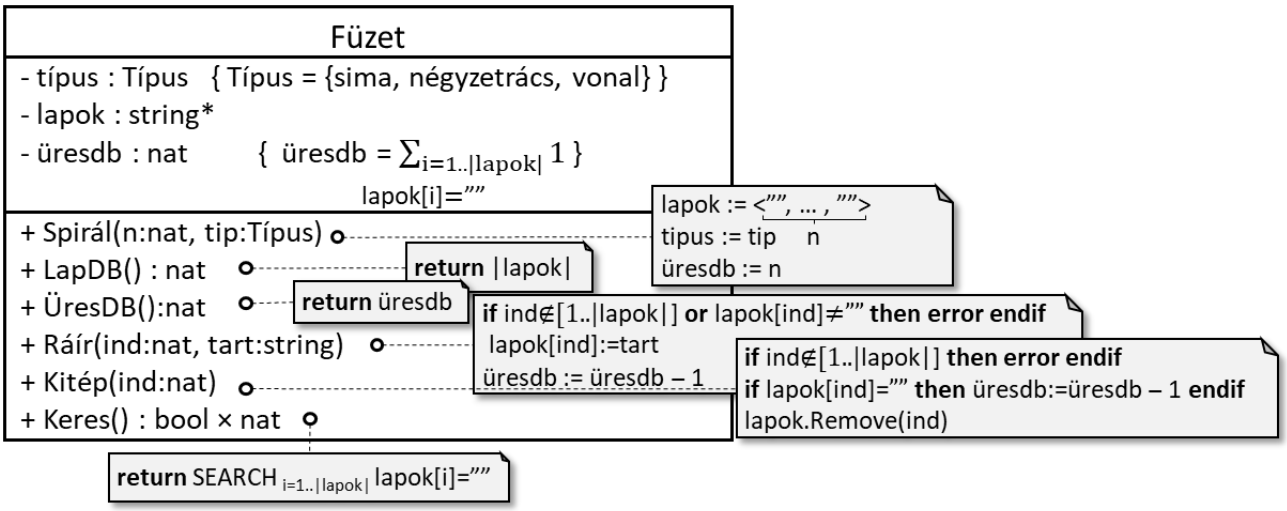
[1] Amikor majd az öröklődés fogalmát megismerjük, akkor a Pozíció osztályból származtathatjuk az Irány osztályt.

2. Spirál füzet típusa. Egy füzet azonos típusú (vagy négyzetrácsos, vagy sima, vagy vonalas) lapokból áll; bizonyos lapjaira már írtak, a többi még üres. Le lehet kérdezni, hogy hány üres lap van még a füzetben; ki lehet tépni valahányadik lapot; rá lehet írni egy kiválasztott lapra, ha az még üres; megkereshetjük az első üres lap sorszámát.

Füzet

füzetek	a) db := LapDB(f) (f:Füzet, db:ℕ) b) db := ÜresLapDB(f) (f:Füzet, db:ℕ) c) f := Ráír(f, ind, tart) (f:Füzet, ind:ℕ, tart:𝕊) d) f := Kitép(f, ind) (f:Füzet, ind:ℕ) e) l, ind := KeresÜres(f) (f:Füzet, l:𝕃, ind:ℕ)
típus : Típus lapok : 𝕊* üresdb : ℕ // üresdb = $\sum_{i=1..\|lapok\|,\ lapok[i]=""} 1$ Típus = {sima, négyzetrács, vonal}	a) db := |lapok| b) db := üresdb c) **if** ind∉[1 .. lapok] **or** lapok[ind]≠"" **then error endif** lapok[ind]:=tart üresdb := üresdb – 1 d) **if** ind∉[1 .. |lapok|] **then error endif** **if** lapok[ind]="" **then** üresdb:=üresdb –1 **endif** lapok.Remove(ind) e) l, ind := **SEARCH**$_{i=1..\|lapok\|}$ (lapok[i] = "")

Az üres lapot kereső művelet tipikus példája a végrehajtható specifikáció alkalmazásának. Ezt a specifikációt egyértelműen visszavezethetjük a lineáris keresés algoritmus mintájára. Ennek használhatnánk a felsorolós változatát is, amely nem az első üres lap indexét, hanem magát az üres lapot adja vissza: l, lap := SEARCH$_{e∈lapok}$ (e="") (lásd előadás).



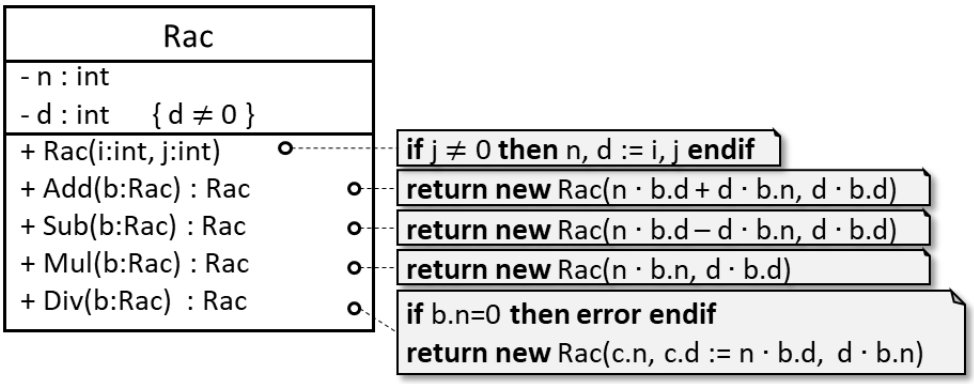
3. Racionális számok. (Ábrázoljuk a racionális számokat egész számpárokkal.)

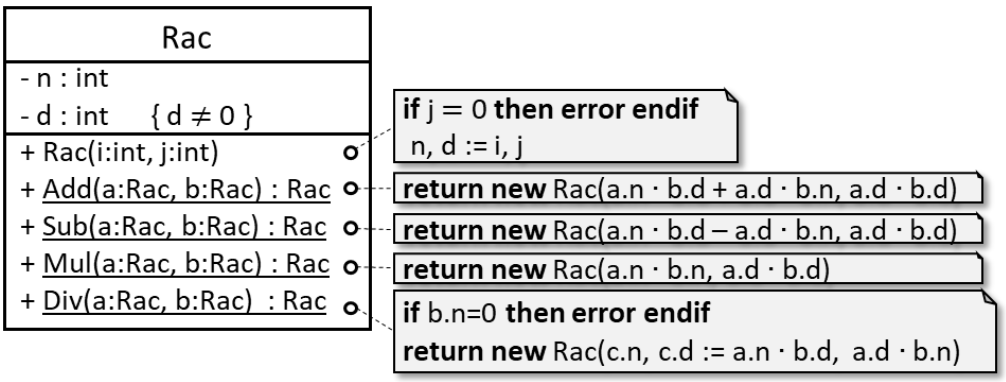
Rac ($\mathbb{Q}$)

racionális számok	c := a ± b (a, b, c: $\mathbb{Q}$)
	c := a · b (a, b, c: $\mathbb{Q}$)
	c := a / b (b≠0) (a, b, c: $\mathbb{Q}$)
n, d: $\mathbb{Z}$ Inv: d≠0	c.n, c.d := a.n · b.d ± a.d · b.n, a.d · b.d
	c.n, c.d := a.n · b.n, a.d · b.d
	if b.n=0 **then error endif** c.n, c.d := a.n · b.d, a.d · b.n (b.n≠0)

A típusinvariáns lehetne a d>0 is, vagy „n és d relatív prím” is.

A műveletek nem egyetlen racionális számhoz kapcsolódnak: két racionális számból állítanak elő egy harmadikat. Ezért nem lenne elegáns (bár megtehetnénk), ha ezeket a műveleteket egyetlen Rac típusú objektum műveleteiként vezetnénk be. Ehelyett ezek a műveletek a Rac osztály (osztályszintű) metódusai lesznek. Emiatt ezeket nem egy racionális szám objektumra kell meghívni úgy, hogy paraméterként adjuk meg a másik számot, hanem olyan metódusként, amelynek két racionális szám partamétere van.





A C# (C++) nyelven lehetőségünk van operátorok felüldefiniálására. Ilyenkor például az összeadást az Add() metódusnév helyett az operator+() metódusnévvel definiáljuk, és ezután c=Add(a,b) helyett használhatjuk a c=a+b kifejezést.