Kalló Bernát **1. beadandó / 3. feladat** 2010. október 5–6.
KABRABI.ELTE *Objektumelvű alkalmazások fejlesztése*
kallo.bernat@gmail.com Gregorics Tibor
3. csoport Szabóné Nacsa Rozália

Feladat

Valósítsuk meg a nagyon nagyszámok típusát! Ábrázoljuk a számokat számjegyeik sorozatával, amelyet egy dinamikus helyfoglalású tömbben helyezzünk el, és implementáljuk a hatékony összeadás és a szorzás műveleteit! Tegye lehetővé két nagyszám típusú változó közötti értékadást!

Bignum típus

Típusértékhalmaz

A természetes számok.

Típusműveletek

1. Összeadás

$e := a + b$

2. Szorzás

$e := a * b$

3. Létrehozás sztringből

A tízes számrendszerbeli alakja alapján hozza létre a számot.

4. Sztringgé alakítás

Állítsa elő a tízes számrendszerbeli alakját.

5. Beolvasás

Beolvassa a szám tízes számrendszerbeli alakját.

6. Kiírás

Írja ki a tízes számrendszerbeli alakját.

Reprezentáció

Egy tömbben (v) tároljuk a szám tízes számrendszerbeli számjegyeit. A 0-s index az egyes helyiértéknek felel meg. Külön mezőben (n) tároljuk a számjegyek számát. Az n-edik helyen nem állhat 0, ha a nullát akarjuk reprezentálni, $n = 0$ lesz.

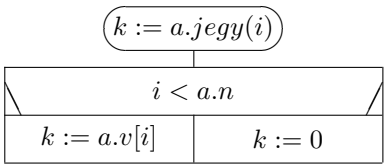
Például az 1234 reprezentációja $(n : 4, v : \langle 4, 3, 2, 1 \rangle)$. A 0 reprezentációja $(n : 0, v : \langle\rangle)$.

Implementáció

Itt a specifikációban szereplő függvényekhez hozzáveszünk még néhányat, amelyeket az összeadáshoz és a szorzáshoz felhasználunk.

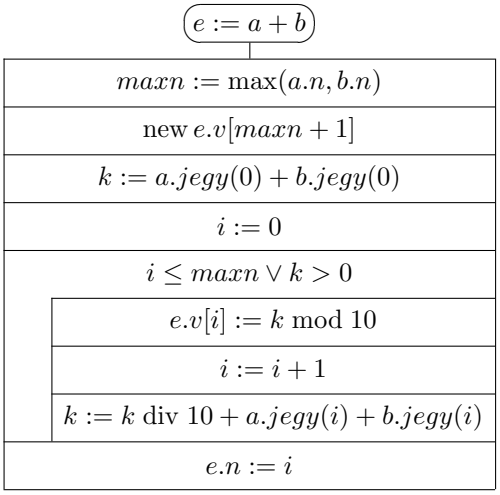
1. Egy jegy lekérdezése

Visszaadja a szám egy jegyét, vagy 0-t.



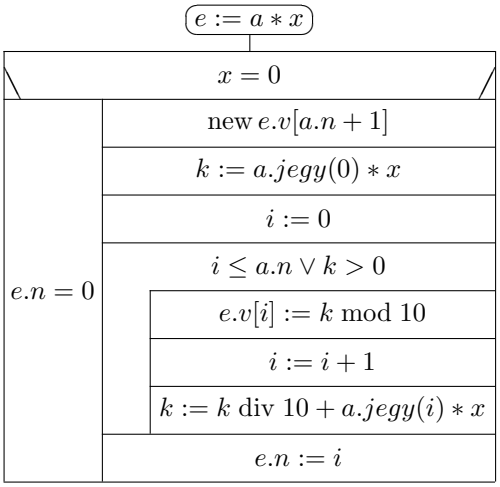
2. Összeadás

A nagyszámokat az írásbeli összeadásnak megfelelően adjuk össze a legkisebb helyiértéktől kezdve a legnagyobbig. Az eredmény leföljebb eggyel lesz hosszabb, mint a hosszabbik operandus, az egyszerűség kedvéért ennyit le is foglalunk a tömbnek.



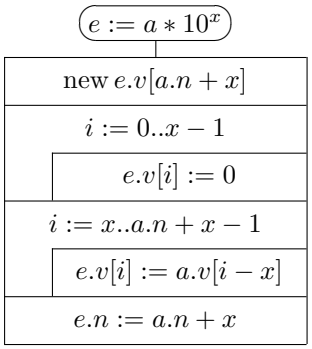
3. Egész számmal szorzás

A nagyszámokat (rendes) egész számmal az írásbeli szorzásnak megfelelően szorzunk, számjegyenként a legkisebb helyiértéktől kezdve, nagyon hasonlóan az összeadáshoz. Itt is megtehetjük, hogy rögtön a szám hosszánál eggyel nagyobb tömböt foglalunk le, mert ennél hosszabb nem lehet a szorzat. Viszont a szám hosszánál lehet rövidebb, akkor, ha a szorzó 0, ezért ezt külön meg kell nézni.



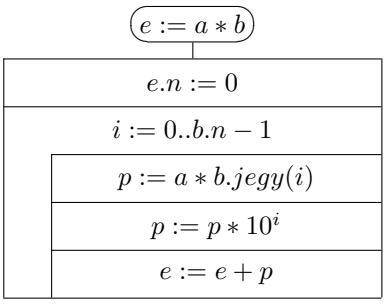
4. Tízhatvánnyal szorzás

Egy nagyszámot tíz hatványával szorozni hatékonyabban is lehet, mint az előző algoitmussal:

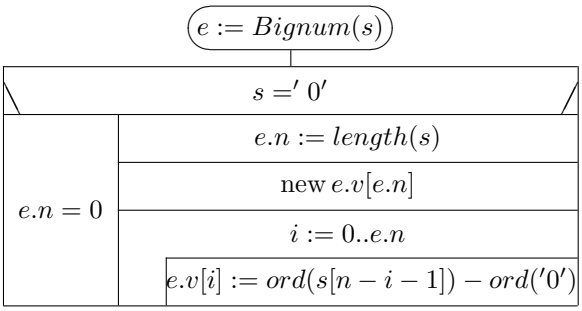


5. Szorzás

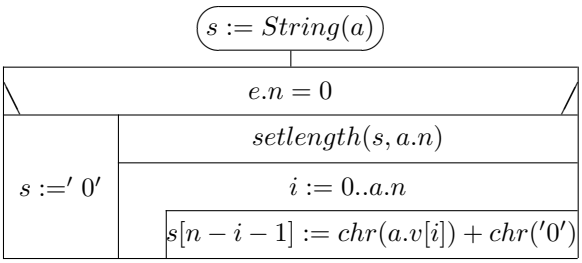
A szorzást vissza tudjuk vezetni a számjeggyel való szorzásra, a tízhatvánnyal való szorzásra és az összeadásra, az írásbeli szorzásnak megfelelően.



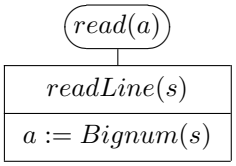
6. Létrehozás sztringből



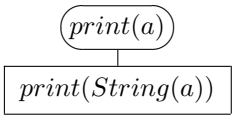
7. Sztringgé alakítás



8. Beolvasás



9. Kiírás



C++ megvalósítás

A `Bignum` osztályt a `bignum.cpp`/`bignum.h` fájlokban valósítjuk meg. A műveletek nagy részét operátor-felüldefiniálással implementáljuk. A további dokumentációt lásd a forráskódban.

A kódban megvalósítjuk az említetteken kívül az értékadást, a copy konstruktort, ill. a destruktort.

Tesztelési terv

A teszteléshez a CxxTest keretrendszert használjuk. Ez a `cxxtest/` mappában található. A tesztek a `test/test.h` fájlban vannak, ebből generáltuk a `test/runner.cpp` fájlt, amit lefordítva kaptuk meg a `test/runner` futtatható állományt. A generáláshoz és fordításhoz szükséges parancsok a Makefile-ban láthatók.

Feketedoboz tesztesetek

1. Számok összeadása átvitel nélkül
2. Számok összeadása átvitellel
3. Nulla összeadása más számmal, nullával
4. Számok szorzása
5. Számok szorzása 1-gyel, 0-val
6. Nagyon nagy számok ($> 2^{32}, 2^{64}$) összeadása, szorzása

Ezek egyben a sztring és Bignum közötti konverziókat is tesztelik.

Fehérdoboz tesztesetek

1. Szorzás kommutativitása az előző esetekben (mivel az algoritmusban nem szimmetrikusan kezeltük a-t és b-t)
2. Szorzáshoz használt segéd metódusok tesztelése
3. Copy konstruktor tesztelése
4. Értékadás tesztelése

5. Destruktor tesztelése
6. Beolvasás és kiírás tesztelése