Skorka Bence
DOMJ1R
skorka.bence@gmail.com

Feladat

Készítsen egy halmaz típust! A halmazt rendezett láncolt listával ábrázolja! Implementálja a szokásos műveleteket (elem betétele, kivétele, benne van-e egy adott elem, üres-e), egészítse ki az osztályt a halmaz tartalmát kiíró operátor << -ral! Definiáljon olyan barát-operátorokat is, amely kiszámítja két halmaz szimmetrikus differenciáját és metszetét! A metszet műveletigénye: Ơ(m+n), ahol m és n a két halmaz elemszáma.

Lista típus

A feladatot egy láncolt listával fogjuk ábrázolni. Ez a lista fejelem nélküli, egyirányú láncolt lista lesz. Egy elem értékei: $value$ – az elemben tárolt érték, $next$ – a következő listaelem. Ebben a listában az elemek mindig növekvő sorrendben helyezkednek el, a könnyű beszúrás és keresés érdekében.

Amennyiben a halmaz üres, úgy az első elemet jelölő mutató 0.

Implementáció

IsEmpty

Ez a függvény meghatározza, hogy a halmaz üres e. Itt csak annyit kell csinálni, hogy megnézzük, hogy az első elemre mutató mutató egy nulla mutató e.

Beszúrás

Új elem beszúrásánál meg kell keresni az elem helyét. Két eset lehetséges: Első esetben még üres a lista, vagy a lista első eleme nagyobb, mint a baszúrni kívánt elem. Ezt a két lehetőséget azért veszem egy esetbe, mert ugyanazt kell csinálni: Új elemet az első helyre, és akármi volt az első elem, arra mutasson az új első elem. (Ha nullptr volt eddig, akkor a next nullptr lesz, ami helyes, ha pedig igazi elem volt, akkor az az elem lesz a next)

A második esetben egy belső elemhez kell helyezni. Ekkor elkezdjük bejárni a listát, és ha megtaláltuk az első olyan elemet, amely nagyobb, mint a beszúrni kívánt elem, az elé kell beszúrnunk. Ehhez folyamatosan tárolunk egy előző elem pointert is, mivel visszafele lépni nem tudunk.

Keresés

Elem keresésénél csak elkezdjük bejárni a listát, és akkor állunk meg, ha megtaláltuk a keresett elemet, vagy találunk egy olyan elemet, amely nagyobb, mint a keresett elem.

Törlés

Törlésnél, mint beszúrásnál, szintén két eset lehetséges: Ha az első elem a törlendő elem, vagy egy belső elem. A beszúráshoz hasonlóan járunk el, csak töröljük a keresett elemet, és nem beszúrjuk.

Kiíró operátor

Kiírás esetén először egy „Set:” nyitással kezdünk, majd ezután felsoroljuk az elemeket. A legvégére kapcsos zárójelek között megjelenítjük a set elemeinek számát.

Metszet

Metszet esetén két set -en kell végig mennünk, és az egyező elemeket kigyűjteni egy új setbe. Szerencsére tudjuk, hogy mindkét set elemei növekvő sorrendben helyezkednek el a listájukban.

```
Set operator & (const Set& set1, const Set& set2) {
    Set::SetItem* p1 = set1.first;
    Set::SetItem* p2 = set2.first;

    Set newset;
    Set::SetItem* np = nullptr;

    while (p1 != nullptr && p2 != nullptr) {
        if (p1->value < p2->value) {
            p1 = p1->next;
        } else if (p1->value == p2->value) {
            if (np == nullptr) {
                newset.first = newset.newItem(p1->value);
                np = newset.first;
            } else {
                np->next = newset.newItem(p1->value);
                np = np->next;
            }
            p1 = p1->next;
            p2 = p2->next;
        } else if (p1->value > p2->value) {
            p2 = p2->next;
        }
    }

    return newset;
}
```

Szimmetrikus differencia

```
Set operator | (const Set& set1, const Set& set2) {
    Set::SetItem* p1 = set1.first;
    Set::SetItem* p2 = set2.first;

    Set newset;
    Set::SetItem* np = nullptr;

    while (p1 != nullptr || p2 != nullptr) {
        if ((p2 == nullptr) || (p1 != nullptr && p1->value < p2->value)) {
            if (np == nullptr) {
                np = newset.newItem(p1->value);
                newset.first = np;
            } else {
                np->next = newset.newItem(p1->value);
                np = np->next;
            }
            p1 = p1->next;
        } else if (p1 != nullptr && p2 != nullptr && p1->value == p2->value) {
            p1 = p1->next;
            p2 = p2->next;
        } else if ((p1 == nullptr) || (p2 != nullptr && p1->value > p2->value)) {
            if (np == nullptr) {
                np = newset.newItem(p2->value);
                newset.first = np;
            } else {
                np->next = newset.newItem(p2->value);
                np = np->next;
            }
            p2 = p2->next;
        }
    }

    return newset;
}
```

Skorka Bence
DOMJ1R
skorka.bence@gmail.com

Megoldás

Set osztály

A Set osztály egy olyan osztály, amely kezeli a láncolt listára vonatkozó műveleteket. Az osztály deklarációját a set.h -ban helyezzük el, a metódusok implementációját pedig a set.cpp forrásban.

Tesztelési terv

1. Üresség tesztje
    a. Kezdetben üres e
    b. Egy elem beszúrva üres e
    c. Az elem kivétele után üres e
2. Keresés & beszúrás & törlés tesztje
    a. Kezdetben ne találjon semmit
    b. Beszúrjuk az elemet, utána megtalálja e
    c. Kivesszük az elemet, most megtalálja e
3. Tisztítás tesztje
    a. Beszúrunk több elemet
    b. Tisztítjuk a listát
    c. Ezután talál e valamit?
4. Metszet
    a. Beszúrunk elemeket
        i. 2-vel osztható számok az egyik setben
        ii. 3-al osztható számok a másik serben
    b. Metszetet számítunk
    c. A végső listában helyes metszet esetén csak 6-al osztható számok maradhatnak
5. Szimmetrikus differencia
    a. Beszúrunk elemeket (a metszethez hasonlóan)
    b. Szimmetrikus differenciát számítunk
    c. A végső listában helyes szimdiff esetén csak ((3-al osztható) XOR (2-vel osztható)) számok maradhatnak.
6. Másolás tesztje
    a. Beszúrunk elemeket
    b. Lemásoljuk a listát
    c. Megegyeznek az elemek?
7. Ön-Másolás tesztje
    a. Beszúrunk elemeket
    b. Saját magába másoljuk a listát
    c. Történt változás?