Geszti Bence
WXNDSZ

6. FELADAT:

Valósítsa meg az egész számokat tartalmazó zsák típust! A zsákot dinamikusan lefoglalt tömb segítségével ábrázolja! Implementálja a szokásos műveleteket (elem betevése, kivétele, üres-e a halmaz, egy elem hányszor van a zsákban), valamint két zsák unióját (a közös elemek előfordulása összegződik), továbbá egy zsák kiírását, és végül a másoló konstruktort és az értékadás operátort! Törekedjen az unióképzés műveletigényének minimalizálására, a dokumentációban mutasson rá a saját megoldásának műveletigényére!

TÍPUS MŰVELETEK

1.Elem betevése

A = zsak(Z) x Z
a s
Q = (a=a' $\wedge$ s=s')
R = (Q $\wedge$ a:= a unio s)

2.Elem kivétele

A = zsak(Z) x Z
a s
Q = (a=a' $\wedge$ s=s' $\wedge$ s eleme a-nak)
R = (Q $\wedge$ a:= a \ e)

3.Két zsák uniója

A = zsak(Z) x zsak(Z) x zsak(Z)
a b c
Q = (a=a' $\wedge$ b=b')
R = (Q $\wedge$ c := a unio b)

IMPLEMENTÁCIO

1.Elem betevése:

_darab != _meret	
_t[_meret] = s _darab := _darab + 1	kivétel

2. Elem kivétele:

l,i := hamis,0	
¬l ∧ i<_darab	
l := _t[i] = s ind := i i: = i+1	
l	
_darab := _darab -1	kivétel
_t[ind] := t[_darab]	
_t[_darab] := 0	

3 Két zsák uniója

i:= 0…a._darab	
c._t[i]:=a._t[i]	
i:= 0…b._darab	
c._t[i+a._darab]:=b._t[i]	
c._darab := a._darab + b._darab	

C++

ZSÁK OSZTÁLY:

```
#ifndef zsak_h
#define zsak_h
#include <iostream>
class zsak{

public:
    enum Exceptions{TELI,NINCS};
    zsak(int n=0);
    ~zsak(){delete[] _t;}
    zsak(const zsak &s);
    zsak& operator= (const zsak &s);
    void be(const int &s);
    void ki(const int &s);
    bool ures() {return _darab==0;}
    int db(const int &s);
    void meretez(const int &new_s);
    friend std::istream& operator>>(std::istream& s, zsak& a);
    friend std::ostream& operator<<(std::ostream& s, const zsak& a);
    friend zsak operator+ (const zsak& a, const zsak& b);
private:
    int* _t;
    int _meret;
    int _darab;

};



#endif /* zsak_h */
```

MENU OSZTÁLY:

```
#include "menu.hpp"
#include "zsak.h"
#include <iostream>

void menu::run()
{
    int x=0;
    menuszoveg();
    do{
        std::cout << "Menupont valasztas: ";
        std::cin >>x;
       switch (x) {
            case 1:   berak();  break;
            case 2:   kivesz();  break;
            case 3:   darabszam();  break;
            case 4:   ures();  break;
            case 5:   unio();  break;
            case 6:   kiir();  break;
            case 7:   meretez(); break;
            }
        std::cout<< std::endl;
    }while (x!=0) ;
}

void menu::menuszoveg()
{
    std::cout << "1. Elem berakasa.\n";
    std::cout << "2. Elem kivetele.\n";
    std::cout << "3. Elem darabszama.\n";
    std::cout << "4. Ures a zsak?.\n";
    std::cout << "5. Unio.\n";
    std::cout << "6. Zsak kiirasa.\n";
    std::cout << "0. Kilep.\n";

}

void menu::berak()
{
    int n;
    std::cout << "Elem: ";
    std::cin >>n;
    try {
        a.be(n);
    } catch (zsak::Exceptions ex) {
        if (ex==zsak::TELI) {
            std::cout<< "A zsak tele van."<< std::endl;;
        }
    }


}
```

```
void menu::kivesz()
{
    int n;
    std::cout << "Elem: ";
    std::cin >>n;
    try {
        a.ki(n);
    } catch (zsak::Exceptions ex) {
        if (ex==zsak::NINCS) {
            std::cout<<"Nincs a zsakban ilyen elem."<< std::endl;;
        }
    }
}

void menu::darabszam()
{
    int n;
    std::cout << "Elem: ";
    std::cin >>n;
    std::cout << a.db(n)<<" darab ilyen szam van a zsakban"<< std::endl;;
}

void menu::ures()
{
    if (a.ures()) {
        std::cout << "Üres."<< std::endl;;
    }
    else
    {
        std::cout<< "Nem üres."<< std::endl;;
    }
}

void menu::kiir()
{
    if (a.ures()) {
        std::cout<<"A zsak ures"<< std::endl;
    }
    else{
    std::cout <<"A zsak elemei: "<< std::endl << a << std::endl;
    }
}

void menu::unio()
{
    zsak b(5);
    std::cout << "Uj zsak elemei: "<< std::endl;
    std::cin >> b;
    std::cout << "Elso zsak elemei:" << a<< std::endl;
    std::cout << "Ket zsak unioja: " << std::endl << a + b;
}

void menu::meretez()
{
    int n;
    std::cout << "Uj meret: ";
    std::cin >>n;
    std::cout<< std::endl;
    a.meretez(n);
}
```

TESZTELÉSI TERV

Fekete doboz tesztelés

1. Különböző méretű zsákok létrehozása, feltöltése és kiírása.
2. Új zsák létrehozása meglevő zsák alapján, majd kiírása.
3. Zsák-értékadás kipróbálása.
4. a és b zsák unió képzés kipróbálása.

Fehér doboz tetsztelés

1. Furcsa méretú zsákok létrehozása (-1, 0, 1, 99)
2. Kivételek