500.CC Megoldó Cecil 500.C

Programozás alapjai 2. (inf.) 1. PZH 2016.05.27. gyak./lab. hiányzás: 0/4	G4-E405/L2-R4B
ABC123 **Q-I/15.**	

Minden beadandó megoldását a feladatlapra, a feladat után írja! Készíthet piszkozatot, de csak a feladatlapra írt megoldásokat értékeljük! *A megoldások során feltételezheti, hogy minden szükséges input adat az előírt formátumban rendelkezésre áll. A feladatok megoldásához csak a letölthető C, C++ és STL összefoglaló használható. Elektronikus eszköz (pl. mobiltelefon) nem használható. A feladatokat* ***figyelmesen olvassa*** *el, megoldásukhoz csak akkor használjon STL tárolót, ha a feladat ezt külön engedi/kéri!* ***Ne írjon felesleges*** *függvényeket ill. kódot! A feleletválasztós feladatoknál a hibás válaszért pontlevonás jár. A részfeladatokra kapott pontok a feladaton belül előjelesen összeadódnak. negatív összeg esetén a feladatra kapott pontszám 0. Az első feladatrészben (beugró) minimum 5 pontot el kell érnie ahhoz, hogy a többit értékeljük.*

f.	max.	elért	jav.
1.	10		
2.	10		
3.	10		
4.	10		
Σ	**40**		

1. feladat: Beugró — Σ 10 pont

a) Jelölje, hogy igaz (I), vagy hamis (H)! (2p)

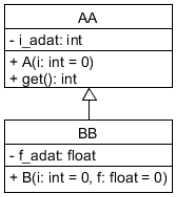
A privát adattagok a scope (::) operátorral mindig elérhetők.	I	**H**
Referencia típusú tagváltozót konstruktor törzsében lehet inicializálni.	I	**H**
A logikai típus egész értéket is kaphat értékül.	**I**	H
B osztály kompatibilis A-val, ha B mindenütt használható, ahol A használható.	**I**	H

b) Mi a hiba az alábbi programrészletben? (1p)

```
{ struct A {
    double* adat;
    A(int i) { adat = new double[i]; }
  };
  A a_obj(20); }
```

Memória fogyás

c) Valósítsa meg C++ nyelven az alábbi osztálydiagramon szereplő osztályokat! A tagváltozók kezdeti értékét a konstruktorparaméterek adják. (2p)



Egy lehetséges megoldás:

```
class AA {
    int i_adat;
public:
    AA(int i = 0) :i_adat(i) {}
    int get() { return i_adat; }
};
class BB : public AA {
    int f_adat;
public:
    BB(int i = 0, float f = 0)
              : AA(i), f_adat(f) {}
};
```

d) Mi a hiba az alábbi programrészletben? (1p)

```
int sum(int i = 0, int j) { return i + j };
```

Default paraméter után más csak default paraméter következhet.

e) (2p)

A c) részfeladat *AA* osztályát egészítse ki úgy, hogy működjön az alábbi kódrészlet:

```
AA a1(1), a2(2);
AA a3 = a1 + a2;
std::cout << a3.get();
```

Valósítsa meg a működéshez szükséges összes tagfüggvényt!

Egy lehetséges megoldás:

```
class AA {
    int i_adat;
public:
    AA(int i = 0) :i_adat(i) {}
    int get() { return i_adat; }
    AA operator+(const AA& rhs) {
       return AA(this->i_adat + rhs.i_adat);
    }
};
```

f) Készítsen inline függvényt (*swap*), ami felcseréli a paraméterként kapott két double típusú adatot! Egy kódrészlettel mutassa be a függvény használatát! (2p)

Egy lehetséges megoldás:

```
inline void swap(int& a, int& b) {
    int t = a;
    a = b;
    b = t;
}

int a = 1, b = 2;
swap(a, b);
```

2. Feladat **Σ 10 pont**

Tételezze fel, hogy egy olyan csapatban dolgozik, melynek egy olyan osztályt kell létrehoznia, ami egész számokból álló véges halmaz dinamikus tárolására alkalmas! A halmaz a dinamikus memóriában mindig csak az elemszámnak megfelelő helyet foglalhatja el. Meg kell valósítani az alábbi műveleteket:

size	Megadja a halmaz elemszámát (számosságát).
isElement	Megvizsgálja, hogy egy elem eleme-e a halmaznak.
empty	Megvizsgálja, hogy a halmaz üres-e.
operator==	Összehasonlít két halmazt.
insert	Egy halmazelem hozzáadása a halmazhoz.
erase	Egy halmazelem eltávolítása a halmazból.
operator+	Eredménye egy új halmaz, ami a két halmaz unióját tartalmazza.
operator-	Eredménye egy új halmaz, ami a két halmaz különbségét tartalmazza.

Az osztály megvalósításán a csapatban többen dolgoznak egyszerre, akikkel megállapodtak az osztály belső adatszerkezetében, a tagfüggvények funkcióiban, valamint abban, hogy nem használnak *friend* függvényt. A megállapodást egy kommentezett deklaráció rögzíti, aminek a vége sajnos elveszett. **Egészítse** ki a deklarációt a tagfüggvények hiányzó deklarációival! Ügyeljen a konstans tagfüggvények helyes jelölésére! Követelmény, hogy az osztály legyen átadható érték szerint függvényparaméterként, és **működjön** helyesen **a többszörös értékadás** is.

```
typedef unsigned int size_t;
class Halmaz {
  int *pData;                              // pointer a tárolóra (itt vannak az adatok)
  size_t siz;                              // tároló mérete
public:
  Halmaz();                                // konstruktor
  size_t size() const;                     // megadja a halmaz számosságát
  bool isElement(int) const;               // megvizsgálja, hogy egy elem eleme-e a halmaznak
  bool empty() const;                      // megvizsgálja, hogy a halmaz üres-e
  bool operator==(const Halmaz&) const;    // összehasonlít két halmazt
  void insert(int);                        // egy halmazelem hozzáadása a halmazhoz
  ~Halmaz();                               // destruktor
  void erase(int);                         // egy halmazelem eltávolítása halmazból
  Halmaz_2 operator+(const Halmaz_2&) const;    // két halmaz uniója
  Halmaz_2 operator-(const Halmaz_2&) const;    // két halmaz különbsége
  Halmaz_2& operator=(const Halmaz_2&);  // értékadó operator
  Halmaz_2(const Halmaz_2&);             // másoló konstruktor
};
```

A tagfüggvények elkészítését felosztották egymás között. Önre a **destruktor,** a **konstruktorok**, az **értékadó operátor,** valamint az **insert** megírása jutott. **Valósítsa** meg a **feladatul kapott** tagfüggvényeket! Vegye figyelembe, hogy a mások által írt függvények belső megoldásait nem ismeri, azaz nem használhat ki olyan működést, ami a fenti kódrészletből vagy annak megjegyzéseiből nem olvasható ki.

Egy lehetséges megoldás:

```
Halmaz::~Halmaz() { delete[] pData; }
Halmaz:: Halmaz() : pData(NULL), siz(0) {}
Halmaz::Halmaz(const Halmaz& h) {
    pData = NULL;
    *this = h;
}
Halmaz& Halmaz::operator=(const Halmaz& rhs) {
    if (this != &rhs) {
        delete[] pData;
        siz = rhs.siz;
        pData = new int[siz];
        for (size_t i = 0; i < siz; i++)
            pData[i] = rhs.pData[i];
    }
    return *this;
}
```

```
void Halmaz::insert(int e) {
    if (isElement(e))
        return;
    int *tmp = new int[siz+1];
    for (size_t i = 0; i < siz; i++)
        tmp[i] = pData[i];
    delete[] pData;
    pData = tmp;
    pData[siz] = e;
    ++siz;
}
```

3. Feladat Σ 10 pont

Tételezze fel, hogy a 2. feladat *Halmaz* osztálya a specifikációnak megfelelően elkészült! A *Halmaz* osztály felhasználásával, annak módosítása nélkül **hozzon létre** egy olyan osztályt (*Halmazom*), ami kompatibilis *Halmaz* osztállyal és van *operator+=* tagfüggvénye is, ami az alaptípusoknál megszokott módon végzi el += műveletet, azaz az unió eredménye a bal oldali operandusban keletkezik.

Készítsen az *std::istream* adatfolyamhoz egy olyan extractor (>>) operátort, ami pontosan 10 egész számot olvas be az adatfolyamról egy *Halmazom* típusú objektumba! Amennyiben nem olvasható be 10 szám, úgy dobjon *std::out_of_range* kivételt! **Mutassa** be egy programrészlettel az osztály használatát: olvasson be a standard inputról 2*10 egész számot egy-egy halmazba. Képezze a halmazok különbségét, és írja ki, hogy az eredmény üres halmaz-e! Kapja el az esetlegesen keletkező kivételt!

```
struct Halmazom : public Halmaz {
    Halmazom() {}
    Halmazom(const Halmaz& h) :Halmaz(h) { }
    Halmazom& operator+=(const Halmazom& h) {
        return *this = *this + h;
    }
};

std::istream& operator>>(std::istream& is, Halmazom h) {
    int j;
    for (int i = 0; i < 10; i++)
        if (cin >> j) h.insert(j);
        else throw std::out_of_range("is >> Halmaz");
    return is;
}

int main() {
    Halmazom h1, h2;
    try {
        cin >> h1;
        cin >> h2;
        if ((h1-h2).empty())
            cout << "ures" << endl;
        else
            cout << "nem ures" << endl;
    } catch (std::exception& e) {
      cerr << e.what() << endl;
    }
    return 0;
}
```

4. Feladat Σ 10 pont

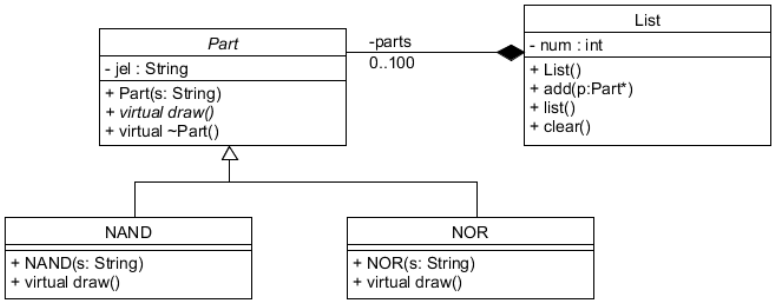
Digitális áramkörök tervezését támogató rendszerben az **alkatrészlistát** úgy szeretnénk elkészíteni, hogy minden alkatrész mellé kirajzoljuk az áramkör jelét is. Minden alkatrésznek van egy szöveges pozíciójele is (pl: G12). Tudjuk, hogy az alkatrészek száma soha nem haladja meg a 100-at. Először csak *NAND* és *NOR* kapukat akarunk használni, de később újabb alkatrészekkel akarjuk bővíteni a rendszert anélkül, hogy a lista implementációját megváltoztatnánk. Az alkatrészek (*Part*) pozíciójelét *String*-ként kell tárolni, az áramköri jel kirajzolásáért pedig a *draw()* tagfüggvény a felelős. A rendszerben megvalósítandó műveleteket egy kódrészlettel mutatjuk be:

```
List parts;                    // Ez lesz az alkatrészlista
parts.add(new NAND("G12"));    // G12 jelű NAND kapu hozzáadása a listához
parts.add(new NOR("G45"));     // G45 jelű NOR kapu hozzáadása a listához
parts.list();                  // alkatrészlista készítése a jelek kirajzolásával
parts.clear();                 // alkatrészlista törlése
```

Feltételezheti, hogy a *List* osztályból példányosított objektumot nem akarjuk paraméterként átadni, és értékadás jobb, ill. bal oldalán sem szerepel.

Tervezzen meg és **rajzoljon** fel egy olyan osztályhierarchiát, ami alkalmas az alkatrészek tárolására, és könnyen bővíthető. Az osztálydiagramban jelölje az adattagok és metódusok láthatóságát is! A *String* osztályt nem kell lerajzolni, arra típusként hivatkozzon!
Deklarálja a *List, Part* és *NOR* osztályokat! **Csak** a *List* és a *NOR* osztályok konstruktorát, valamint a *List* osztály *add()* és *list()* metódusát **valósítsa** meg!

Egy lehetséges megoldás:




```
class Part {
        String jel;
public:
        Part(String);
        virtual void draw();
        virtual ~Part();
};
class NOR : public Part {
public:
        NOR(String s)
                :Part(s) {}
        void draw();
};
class List {
        Part* parts[100];
        int numParts;
public:
        List() :numParts(0) {}
        void add(Part* p) {
                parts[numParts++] = p;
        }
        void list() {
                for (int i = 0; i < numParts; i++)
                        parts[i]->draw();
        }
        void clear();
        ~List();
};
```