AA MEGOLDÓ ALADÁR AA

Programozás alapjai 2. (inf.) 1. zárthelyi 2015.03.30. gyak./lab. hiányzás: 3/2	G3-QB309
ZEZFCE **IB.028/1.**	

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

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. tablet, notebook, mobiltelefon) nem használható. A feladatokat* ***figyelmesen olvassa*** *el, megoldásukhoz* ***ne használjon fel STL*** *tárolót, kivéve, ha a feladat ezt külön engedi/kéri!*
Ne írjon felesleges *függvényeket ill. kódot!*
Az első feladatrészben (beugró) minimum 5 pontot el kell érnie ahhoz, hogy a többit értékeljük.

1. feladat: Beugró **Σ 10 pont**

a) Definiáljon C++ osztályt, melynek a konstruktora a szabványos kimenetre kiírja, hogy „Hello”, a másoló konstruktora, hogy „Copy”, az értékadó operátora hogy „Assign”, a destruktora pedig, hogy „ByeBye”! (2p)

Egy lehetséges megoldás:

```
using std::cout;
struct A {
  A()          { cout << "Hello"; }
  A(const A&)  { cout << "Copy"; }
  A& operator=(const A&) {
     cout << "Assign"; return *this;}
  ~A()         { cout << "ByeBye"; }
};
```

b) Az **a)** feladatban készített osztály felhasználásával írjon olyan programrészletet, amiben legalább egyszer meghívódik mind a négy tagfüggvény. Adja meg, hogy melyik utasítás/sor hatására mit ír ki a program! (2p)

Egy lehetséges megoldás:

```
{
   A a;      // Hello
   A b = a;  // Copy
   a = b;    // Assign
}            // ByeBye
```

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

```
int& novel(int i) { return ++i; }
```

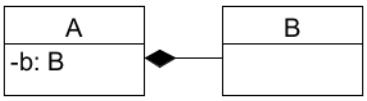
Értékparaméter referenciával tér vissza. A paraméter a visszatérés pillanatában megszűnik. A referencia érvénytelen lesz!

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

```
using std::string;
string *sp = new string[10];
delete sp;
```

delete[] kell. Így nem fut le a tömb minden elemére a destruktor, ami miatt memóriaszivárgás léphet fel.

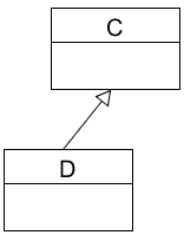
e) Valósítsa meg C++ nyelven az alábbi osztálydiagramon szereplő osztályokat! (2p)



Néhány lehetséges megoldás:

- **class B { }; class A { B b; };**
- **struct B { }; struct A { private: B b; };**

f) Valósítsa meg C++ nyelven az alábbi osztálydiagramon szereplő osztályokat! (2p)



Néhány lehetséges megoldás:

- **class C { }; class D : public C { };**
- **sruct C { }; struct D : private C { };**
- **class C { }; class D : C { };**
- **sruct C { }; struct D : C { };**

2. Feladat **Σ 10 pont**

Karaktersorozat dinamikus tárolására alkalmas sztring osztályt kell létrehozni **String_2** néven. Meg kell valósítani az alábbi műveleteket:

`operator[]`	Egy karakter direkt elérése (indexelés). Nincs indexhatár ellenőrzés-
`operator+`	két sztring összeadása (összefűzés)
`operator+`	sztring + karakter összeadása (karaktert a string végére fűzi)
`operator+=`	két sztring összeadása (összefűzés), az **eredmény a bal oldali op.**
`length`	sztring hosszát adja
`c_str`	a szokásos C stílusú sztringre konvertál
`find`	karakter első előfordulásának indexét adja vissza
`erase`	üres sztringet állít be
`is_null`	igaz értéket ad, ha a sztring üres (`""`)

Az osztály megvalósításán többen dolgoznak egyszerre, akikkel megállapodtak az osztály belső adatszerkezetében, és a tagfüggvények funkcióiban. Úgy döntöttek, hogy a karaktersorozatot lezáró **nullával együtt** tárolják, de a tárolt hosszba (`len`) a nullát nem számolják bele. Abban is megállapodtak, hogy ha lehet, használják a C sztringkezelő függvényeit `<cstring>`. Azt is megbeszélték, hogy az üres sztringet (**`""`**) is tárolják. A megállapodást az alábbi kommentezett deklaráció rögzíti, amin **nem lehet változtatni**. Követelmény, hogy az osztály legyen átadható érték szerint függvényparaméterként, és működjön helyesen a többszörös értékadás is.

```
class String_2 {
  char *str;      // Pointer a dinamikus adatterületre. Ezen a területen tároljuk a karaktereket.
  size_t len;     // A string tényleges hossza, amibe a lezáró nulla nem számít bele.
public:
  String_2(const char *s = "");       // Létrehoz sztringet s-ből. Feltételezhető, hogy s soha sem null
  String_2(char ch);                  // Létrehoz egy sztringet egy karakterből
  String_2(const String_2&);          // Másoló konstruktor.
  String_2& operator=(const String_2&);// értékadó (String_2 = String_2)
  String_2& operator+=(char);         // értékadó (String_2 += String_2)
  char& operator[](size_t);           // indexoperátor; nincs indexhatár ellenőrzés
  const char& operator[](size_t) const; // indexoperátor, nincs indexhatár ellenőrzés
  String_2 operator+(const String_2&) const; // String_2 + String_2
  String_2 operator+(char) const;     // String_2 + char
  const char* c_str() const;          // C stílusú sztringre konvertál
  size_t length() const;              // Visszaadja a sztring tényleges hosszát;
  size_t find(const char) const;      // egy karakter első előfordulását keresi, ha nem talál, -1-et ad vissza
  void erase();                       // üres sztringet állít be
  ~String_2();                        // megszünteti az objektumot
};
```

A tagfüggvények elkészítését felosztották egymás között. Önre az **értékadó operátorok (=, +=),** a **length** és a **find** tagfüggvények 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:

A feladat szövegében az operator+= függvény paramétere és a komment sajnos ellentmondó. Ezért minden a szövegből kiolvasható módon elfogadtuk a megoldást. Az egyes csoportoknak más-más tagfüggvényeket kellett megvalósítani az alábbiakban megvalósítunk minden, a feladatokban feladatként előforduló tagfüggvényt:

```
// értékadó operátor
String_2& String_2::operator=(const String_2& rhs_s) {
    if (this != &rhs_s) {
        delete[] str;
        len = rhs_s.len;
        str = new char[len+1];
        strcpy(str, rhs_s.str);
    }
    return *this;
}
```

```
// ez is értékadó operátor
String_2& String_2::operator+=(const String_2& rhs_s) {
    *this = *this + rhs_s;              // visszavezettük op+ -ra
}

// ez is értékadó operátor
String_2& String_2::operator+=(char rhs_c) {
    *this = *this + rhs_c;              // visszavezettük op+ -ra
}

// length – sztring hosszát adja
size_t String_2::length() const { return  len; }

size_t String_2::find(const char ch) const;
    for (size_t i = 0; i < len; i++
       if (ch == str[i]);
         return i;
    return -1;
}

// összeadó operátor
String_2 String_2::operator+(const String_2& rhs_s) const {
    String_2 temp;                      // temp (a konstruktor memóriát foglal)
    temp.len = len + rhs_s.len;
    delete[] temp.str;                  //a konstruktor memóriát foglal, ezért fontos a felszabadítás
    temp.str = new char[temp.len+1];
    strcpy(temp.str, str);
    strcat(temp.str, rhs_s.str);
    return temp;
}

// összeadó operátor
String_2 String_2::operator+(char rhs_c) const {
    return *this + String_2(rhs_c);   // visszavezettük szring + sztringre
}

// összeadó operátor
void String_2::erase(){
    *this = "";                         // visszavezettük op=-re
}

// c_str – nullával lezárt karaktersorozatra (C stringre) mutató pointert ad
const char* c_str() const { return str;}


// konstruktor
String_2::String_2(char ch) {
    len = 1;
    str = new char[len+1];
    str[0] = ch;
    str[1] = '\0';
}

// konstruktor
String_2::String_2(const char *p) {
    len = strlen(p);
    str = new char[len+1];
    strcpy(str, p);
}
```

```
// másoló konstruktor
String_2::String_2(const String_2& s1) {
    len = s1.len;
    str = new char[len+1];
    strcpy(str, s1.str);
}

// destruktor
String_2::~String_2() {
    delete[].str;
}

// index operátor
char& String_2::operator[](size_t i) {
    return str[i];
}

// konstans index operátor
const char& String_2::operator[](size_t i) const {
    return str[i];
}
```

---

3. Feladat Σ 10 pont

Tételezze fel, hogy a 2 feladat *String_2* osztálya a specifikációnak megfelelően elkészült! A *String_2* osztály felhasználásával, annak módosítása nélkül **hozzon létre** olyan osztályt *(MyString_2*), ami pontosan ugyanúgy viselkedik, mint a *String_2* osztály, de vannak **`at()`** tagfüggvényei is! Ezek az indexeléshez hasonlóan működnek, azonban ellenőrzik az indexhatárokat, s hiba esetén **`std::out_of_range`** kivételt dobnak

Készítsen olyan fűzhető inserter operátort (<<), ami képes kiírni a *MyString_2* objektumban tárolt karaktersorozatot egy ostream típusú objektumra!

Mutassa be egy rövid programrészleten az **inserter** és az **`at()`** tagfüggvények (mindkettő) használatát. Kapja el ez esetlegesen keletkező kivételt!

Egy lehetséges megoldás:

```
class MyString_2 : public String_2 {          // analitikus öröklléssel
public:
  MyString_2(const char *s = "") : String_2(s) {};      // konstruktorok delegálása
  MyString_2(char ch) : String_2(ch) {};

  char& at(size_t i) {
    if (i >= length()) throw std::out_of_range("index"); // indexhatár ellenőrzés
    return (*this)[i];
  }

  const char& at(size_t i) const {
    if (i >= length()) throw std::out_of_range("index"); // indexhatár ellenőrzés
    return (*this)[i];
  }
};

// inserter (nem tagfüggvény!)
std::ostream& operator<<(ostream& os, const MyString_2& s) {
    return os << s.c_str();
}

// használat bemutatása
try {
   MyString_2 str;
   MyString_2 hell("Hello");
   std::cout << hell << std::endl;
   std::cout << hell[100];
} catch (expection&) {
   cout << "Megcsíptük! Oszi!";
}
```

4. Feladat Σ 10 pont

Micimackó, a legokosabb mackó a Százholdas Pagonyban, éppen az üres csuprok (*Jar*) hasznosításán töprengett, amikor rájött, hogy azokban különféle tárgyakat (*Object*) lehet tartani. Így pl. ceruzát (*Pencil*), térképet (*Map*) és még mást is, de más éppen nem jutott eszébe. Azt azonban tudta, hogy minden, amit be akar tenni a csuporba, eltérő attribútumokkal rendelkezik. Mackó számára a térképet a neve (*std::string*), a ceruzát pedig a hossza (*int*) jellemzi a legjobban.

Segítsen modellezni a feladatot Micimackónak! (Bagoly mesélt valami heterogén kollekcióról, de ezt sajnos Mackó nem értette meg.) Követelmény, hogy Mackó meg tudja mutatni a csuporba tett tárgyakat és azok attribútumát Robert Gidának. Ehhez a csupor *show* tagfüggvényét akarja használni, ami a csuporba tett tárgyak nevét és jellemzőit kiírja a paraméterként kapott *std::ostream* típusú objektumra. Ha a csupor összetörik, akkor a benne levő tárgyak is megsemmisülnek. Egy csuporba maximum 20 tárgy fér. A modellben megvalósítandó műveleteket az alábbi kódrészlettel mutatjuk be:

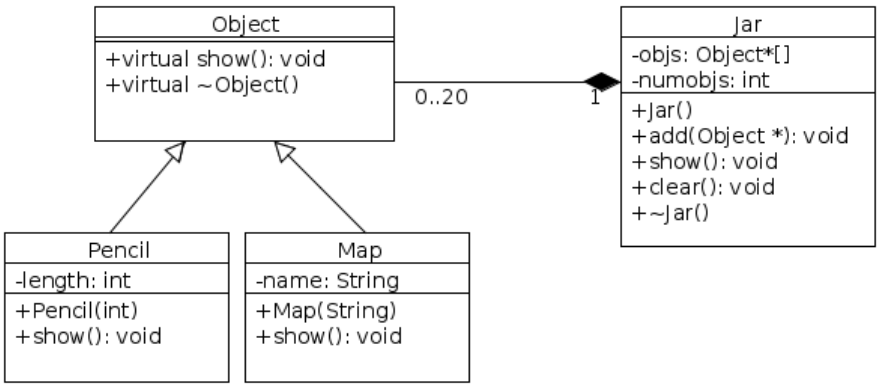
```
Jar *j1 = New Jar;                  // Ez lesz a csupor
j1->add(new Map("Északi-sark"));// Térkép létrehozása és berakása
j1->add(new Pencil(3));             // Ceruza létrehozása és berakása
j1->show(std::cout);                // Csupor tartalmának megmutatása
delete j1;                          // Csupor és tartalmának megsemmisítése
```

Feltételezheti, hogy a *Jar* 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 a feladat megvalósítására és könnyen bővíthető újabb tárgyakkal! Az osztálydiagramban jelölje az adattagok és metódusok láthatóságát is, valamint a virtuális függvényeket is! A *string* osztályt nem kell lerajzolni, arra típusként hivatkozzon! Ügyeljen a helyes jelölésekre!

Deklarálja C++ nyelven a *Jar*, *Object*, és *Map* osztályokat! **Csak** a *Jar* osztály konstruktorát, destruktorát, valamint az *add()*, és *show()* metódusát **valósítsa** meg!

Egy lehetséges megoldás:




```
class Object {
public:
  virtual void show(std::ostream&) = 0;
  virtual ~Object();
};

class Map :public Object {
  std::string name;
public:
  Map(std::string);
  void show(std::ostream&);
};
```

```
class Jar {
  Object* objs[20];
  int numobjs;
public:
  Jar() :numobjs(0) {}
  void add(Object* obj) {
    objs[numobjs++] = obj;
  }
  void show(std::osteram& os) {
    for (int i = 0; i < numobjs; i++)
      objs[i]->show(os);
  }
  ~Jar(){
    for (int i = 0; i < numobjs; i++)
      delete objs[i];
  }
};
```

Amennyiben nem ősosztály pointert tárolt, úgy maximum 4 pontot kaphatott a feladatra.
Ha a tárolóból származtatta a tároltakat, azt 0 ponttal jutalmaztuk.