200. Megoldó Alfréd 200.

Programozás alapjai 2. (inf.) zárthelyi	2009.05.15. gyakorlat:	Érdemjegy:
	Ka.51/4.	Hftest:

F.	Max.	Elért
1	3	
2	4	
3	5	
4	3	
5	5	
Σ	**20**	

Minden beadandó megoldását a feladatlapra, a feladat után írja! *A megoldások során feltételezheti, hogy minden szükséges input adat az előírt formátumban rendelkezésre áll.*
A feladatok megoldásához csak a letölthető C és C++ összefoglaló használható. Számítógép, notebook, menedzser kalkulátor, organiser, rádiótelefon nem használható.

*A feladatokat **figyelmesen olvassa** el, megoldásukhoz **ne használjon fel STL** tárolót, kivéve, a feladat ezt külön engedi! **Ne írjon felesleges** függvényeket ill. kódot, a feladatra koncentráljon!*
Jó munkát! *Értékelés: 0-8:1, 9-11:2, 12-14:3, 15-17:4, 18-20:5*

1. Feladat 6*0.5=3 pont

Mit ír ki a szabványos kimenetre az alábbi program? Válaszához használja a négyzetrácsos területet!

```
#include <iostream>
using namespace std;

class Alap {
  int alap;
public:
  Alap(int i) :alap(i)        { cout << alap << 'k';}
  Alap(const Alap&) :alap(100) { cout << alap << 'c';}
  virtual ~Alap()             { cout << 'd'; }
};

class Uj :public Alap {
  const char *s;
public:
  Uj(const char *s = "B", int al = 0)
      :s(s), Alap(al)          { cout << s << "Uk"; }
  virtual ~Uj()                { cout << "Ud"; }
};

int main() {
  Uj e45("E4ESB1"); cout << endl;
  Uj b1 = e45;      cout << endl;
  Alap *uj = new Uj;cout << endl;
  delete uj;        cout << endl;
  uj = new Alap(5); cout << endl;
  return(0);
}
```

0	k	E	4	E	S	B	1	U	k					
1	0	0	c											
0	k	B	U	k										
U	d	d												
5	k													
U	d	d	U	d	d									

2. Feladat Σ 4 pont

Írjon függvénysablont (**my_unique**), ami az STL *unique* sablonjához hasonlóan egy tároló elemeit úgy „rendezi” át helyben, hogy az egymás után ismétlődő adatokból csak egy maradjon! A függvény két előrehaladó iterátort kap paraméterként, ami kijelöli a jobbról nyílt intervallum kezdetét és végét (pl: x.begin(), x.end()). A függvény visszatérési értéke szintén egy iterátor, ami az átalakított adatsorozat végére mutat. (A kihagyott elemek miatt rövidebb lehet az adatsorozat, ezért ettől az iterátortól a tároló végéig szemét maradhat!) Feltételezheti, hogy a tárolóban levő generikus adatnak van egyenlőségvizsgálat (==) operátora. (2 pont)

Amennyiben helyesen oldja meg a feladatot, akkor az alábbi kódrészlet a következőt írja ki: **`1, 2, 7, 8, 2,`**

```
int v[] = { 1, 2, 2, 7, 8, 8, 2, 2, 2, 2 };
int *pe = my_unique(v, v+10);
for (int *p = v; p != pe; ++p)
   cout << *p << ", ";
```

Adott a következő deklaráció: **`Storage store;`** Tételezze fel, hogy a **`Storage`** osztály egész számokat tárol, és van iterátora, valamint létezik a szokásos begin(), end() és erase() tagfüggvénye is! (Index operátora nincs!). Az **`erase()`** tagfüggvény törli a tárolóból a paraméterként kapott két iterátor közötti elemeket.

Írjon olyan programrészletet, amely az elkészített **my_unique** felhasználásával törli a **`store`** tárolóból az egymás után ismétlődő egész számokat! (2 pont)

Egy lehetséges megoldás:

```
template <class I>
I my_unique(I first, I last) {
    I res = first;
    while (++first != last)
        if (!(*res == *first))
            *(++res) = *first;
    return ++res;
}
Tarolo::iterator i1 = my_unique(tar.begin(), tar.end());
tar.erase(i1, tar.end());
```

3. Feladat Σ 5 pont

Egy *String5* osztályt kell létrehozni, ami karaktersorozatok dinamikus tárolására alkalmas. Meg kell valósítani az alábbi műveleteket:

`operator[]`	Egy karakter direkt elérése (indexelés). hibás indexelés esetén `range_error` kivételt dob.
`operator+`	két string összeadása (összefűzés)
`operator+`	string + karakter összeadása (karaktert a string végére fűzi)
`operator+=`	két string összeadása (összefűzés), az **eredmény a bal oldali op.**
`length`	string hosszát adja
`operator(const char*)`	a szokásos C stílusú stringre konvertál
`Find`	karaktersorozat első előfordulásának indexét adja vissza
`erase`	törli a stringet
`is_null`	igaz értéket ad, ha a string 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 stringkezelő függvényeit `<cstring>`. Azt is megbeszélték, hogy az üres stringet (**`""`**) 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 String5 {
protected:
  char *str;              // Pointer a dinamikus adatterületre. Ezen a területen tároljuk a karaktereket.
  int len;                // A string tényleges hossza, amibe a lezáró nulla nem számít bele.
public:
  String5(const char *s = "");// Létrehoz stringet s-ből. Feltételezhető, hogy s soha sem null
  String5(const String5&);    // Másoló konstruktor.
  String5& operator=(const String5&);       // értékadó (string = string)
  String5& operator+=(const String5&);    // értékadó (string += string)
  char& operator[](int);      // indexoperátor. Hibát dob, ha az index rossz
  String5 operator+(const String5&) const;// string + string
  String5 operator+(char) const;            // string + char
  operator const char*() const;// C stílusú stringre konvertál
  int length() const;          // Visszaadja a string tényleges hosszát;
  int find(const char*) const; // karaktersorozat első előfordulását keresi, ha nem talál, -1-et ad vissza
  void erase();                                // törli a stringet
  ~String5();                                  // megszünteti az objektumot
};
```

A tagfüggvények elkészítését felosztották egymás között. Önre ..... 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 (4 pont)!

Az osztály tesztelése közben kiderült, hogy nincs beolvasó és kiíró tagfüggvénye se az osztálynak. Kollegája azt mondja, hogy nem is kell, hiszen kipróbálta és működött a cout << String**5** (”Hello”); utasítás.

Hogyan lehetséges ez? Esetleg tévedett? Működik-e beolvasás? Röviden válaszoljon! (1p)

Válasz:

Nem tévedett a kollega, mert a kiírás a char*-ra való konverzió miatt megy. A beolvasást viszont meg kell írni!

Lehetséges megoldások:

Konsruktorok és a destruktor:

```
String5::String5(const char *s) {
    len = strlen(s);
    str = new char[len+1];
    strcpy(str, s);
}
String5::String5(const String5& s) {
    str = NULL;
    *this = s;

}
String5::~String5() {
    delete[] str;
}
```

Két értékadó operátor, és az erase:

```
String5& String5::operator=(const String5& s) {
    if (this != &s) {
        delete[] str;
        len = s.len;
        str = new char[len+1];
        strcpy(str, s);
    }
    return *this;
}
String5& String5::operator+=(const String5& s) {
    return *this = *this + s;

}
void String5::erase() {
    len = 0; str[0] = 0; // üres stringet mindig tárolunk
}
```

Az operator+=, az operator+ és a lenght:

```
String5& String5::operator+=(const String5& s) {
    return *this = *this + s;
}
int String5::length() const {
    return len;
}
String5 String5::operator+(const String5& s) const {
    char *tmp = new char[len+s.len+1];
    strcpy(tmp, str);
    strcat(tmp, s.str);
    String5 s5tmp(tmp);
    delete[] tmp;
    return s5tmp;
}
String5 String5::operator+(char ch) const {
    char tmp[2];
    tmp[0] = ch; tmp[1] = 0;
    return *this + String5(tmp);
}
```

Index operátor, a C stringre konvertáló, és a find():

```
char& String5::operator[](int i) {
    if (i >= len || i < 0)
        throw range_error("indexelesi hiba");
    return str[i];
}
String5::operator const char*() const {
    return str;
}
int String5::find(const char *s) const {
      char *p = strstr(str, s);
      if (p == NULL)
            return -1;
      else
            return p-str;
}
```

4. Feladat Σ 3 pont

Tételezze fel, hogy **a 3. feladat** *String5* osztály elkészült és hibátlanul működik! Ezen osztály **felhasználásával készítsen** egy olyan *MyString5* osztályt, ami a *String5* osztállyal azonos funkcionalitású, de a kostruktorokon/ destruktoron kívül csak értékadás és egyenlőtlenségvizsgáló (!=) operátora van! Más elérhető tagfüggvénye nincs! (Ügyeljen az alapértelmezett tagfüggvényekre is, hogy azok se legyenek elérhetők!) A megoldást nem értékeljük, amennyiben nem használja fel a *Srting5* osztályt!

Egy lehetséges megoldás:

```
class MyString5: private String5 {
  MyString5* operator&();      // a feladat szerint ez is kell, de nem vettük hibának, ha valaki elhagyta
  void operator,(const MyString5&); // és ez is ...
public:
  MyString5(const char *s = ""): String5(s) {}
  bool operator!=(const MyString5&) const;
  MyString5& operator+=(const MyString5&);// a feladat szerint ez is kell, de nem vettük hibának ...
};
bool MyString5::operator!=(const MyString5& s) const {
    return strcmp(str, s.str) != 0;
}
MyString5& MyString5::operator+=(const MyString5& s) {
    String5::operator+=(s);
    return *this;
}
```

Alternatív megoldás1: tartalmazott objektummal.

```
class MyString5 {
  String5 data;
  MyString5* operator&();      // a feladat szerint ez is kell, de nem vettük hibának, ha valaki elhagyta
  void operator,(const MyString5&); // és ez is ...
public:
  MyString5(const char *s = ""): data(s) {}
  bool operator!=(const MyString5&) const;
  MyString5& operator+=(const MyString5&); // a feladat szerint ez is kell, de nem vettük hibának ...
};

bool MyString5::operator!=(const MyString5& s) const {
    return strcmp(data, s.data) != 0;
}
MyString5& MyString5::operator+=(const MyString5& s) {
    data += s.data;
    return *this;
}
```

Alternatív megoldás2: **Publikus örökléssel:**
Minden nem kívánt tagfüggvényt privváttá kell tenni. (Sokat kell írni hozzá!)

---

5. Feladat Σ 5 pont

Egy zsebszámológép kétoperandusú algebrai kifejezéseiben műveletek (összeadás, kivonás de ez a lista bővülhet) és valós számok szerepelnek. Minden **művelethez** hozzá lehet rendelni **két kifejezést** (a művelet két operandusát). **Kifejezésnek** számítanak mind a **műveletek**, mind maguk a **számok** is. Minden kifejezésnek meg kell valósítania a *evaluate()* metódust. Ez számok esetén a szám értékét, műveletek esetén a művelet eredményét adja vissza, ha azt a két beállított operandusra végrehajtjuk.

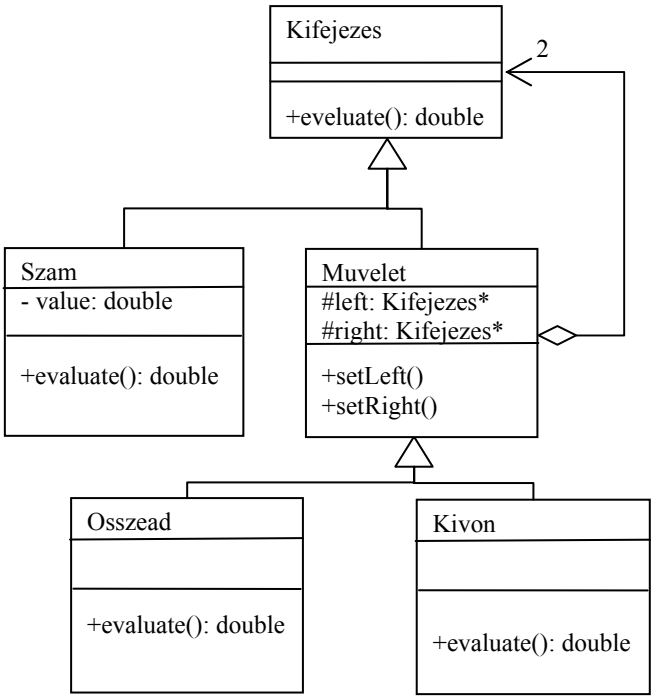
A rendszerben a következő funkciókat kell megvalósítani:

- Művelethez operandus rendelése (*setLeft, setRight*)
- Számhoz a szám értékének módosítása (*setValue*)
- Kifejezés értékének rekurzív kiértékelése (*evaluate*)

Feladatok:

- **Tervezzen** meg egy olyan OO modellt, mely a fenti követelményeket kielégíti! Rajzolja fel a modell osztálydiagramját!
- **Definiálja** az *Kifejezes, Osszeadas, Szam* osztályokat és az elvárt a funkciók ellátásához szükséges tagfüggvényeket! Használja a dőlt betűs neveket!
- **Implementálja** a fenti osztályokat és azok tagfüggvényeit! Az osztályokat olyan módon készítse el, hogy újabb művelet- vagy kifejezéstípus felvételekor a már meglevő kódot ne kelljen módosítani! (2 pont)
- **Írjon** egy egyszerű programrészletet, ami kiszámolja a 3+4+5 kifejezés értékét! (0.5 pont)

Lehetséges megoldás:




```
class Kifejezes {
  public:
      virtual ~Kifejezes(){}
      virtual double evaluate() = 0;
};

class Szam : public Kifejezes {
      double value;
  public:
      Szam(double d) : value(d) {}
      ~Szam() {}
      double evaluate() { return value; }
      void setValue(double d) { value = d;
  }
};
```

```
class Muvelet : public Kifejezes {
  protected:
      Kifejezes* leftOp;
      Kifejezes* rightOp;
  public:
      Muvelet(Kifejezes *l = 0, Kifejezes *r = 0) : leftOp(l), rightOp(r) {}
      ~Muvelet() { delete leftOp; delete rightOp; }
      void setLeft(Kifejezes* k) { delete leftOp; leftOp = k; }
      void setRight(Kifejezes* k) { delete rightOp; rightOp = k; }
};
class Osszeadas : public Muvelet {
  public:
      Osszeadas(Kifejezes *l = NULL, Kifejezes *r = NULL) : Muvelet(l, r) {}
      double evaluate() { return (leftOp->evaluate() + rightOp->evaluate()); }
};

int main() {
   Kifejezes *kif = new Osszeadas(new Szam(3),new Osszeadas(new Szam(4),new Szam(5)));
   cout << kif->evaluate() << endl;
   return 0;
}
```