AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

Szoftver laboratórium 2. 3. ellenőrző dolgozat. 2012.03.27. Kurz/**Terem**: L2/	**15 perc**
Név: **Neptun:**	Összpont:

1.feladat 4.5+1.5 pont

Valósítson meg C++ nyelven egy olyan osztályt (***Stringek***), ami tetszőleges számú *String* típusú objektumot képes tárolni! Elvárás az osztállyal szemben, hogy másolható legyen és értékadás bal és jobb oldalán is szerepelhessen. Legyen az osztálynak egy ***keres()*** tagfüggvénye, ami a tárolóban megkeres egy, a paraméterként megkapott betűvel kezdődő stringet. Ha több ilyen van, akkor az első találatot adja, ha nincs ilyen, úgy üres stringet (`String("")`) adjon vissza! (Feltételezzük, hogy üres stringet nem tárolunk a *Stringek* osztályban.) Tételezze fel, hogy a *String* osztály létezik és helyesen működik!

Megadtuk a ***Stringek*** osztály deklarációját, és a tagfüggvények definícióját, de ez több helyen hiányos. A **pontozott** részekre írva **egészítse ki** az alábbi **kódrészletet** úgy, hogy a megadott főprogram az elvárásoknak megfelelően működjön, és ne lépjen fel memóriakezelési hiba! Nem kell minden pontozott részre írnia! Feltételezheti, hogy az **értékadás operátor,** amely külön állományban van megvalósítva, **helyesen működik,** ezért azt **nem kell** megvalósítania!

A feladatlap hátoldalán **adja meg**, hogy **megoldása** minimálisan **milyen elvárásokat támaszt** a *String* osztállyal szemben! (pl: van olyan konstruktora, ami C stringet kap paraméterként; van … operátora; stb.) (1.5p)

```
class Stringek {
    String* pStr;      // dinamikus adatterület kezdőcíme, ahol tárolunk
    int db;            // tárolt Stringek száma (pontosan ennyi string van)
public:
// Konstruktor: n darab C stringet kap, amiből n darab String-et készít és eltárolja
    Stringek(const char *sv[], int n = 0) :db(n) :db(n) {
        pStr = new String[db];
        for (int i = 0; i < db; i++)
            pStr[i] = String(sv[i]);
    }
    Stringek(const Stringek&) ................;
    Stringek& operator=(const Stringek&);
    String keres(char) const;
    ........~Stringek() { delete [] pStr; }
};
Stringek::Stringek(const Stringek& s) {
    db = s.db;
    pStr = new String[db];
    for (int i = 0; i < db; i++)
       pStr[i] = s.pStr[i];
}
// Adott betűvel kezdődő stringet keres.
String Stringek::keres(char ch) const {
    for (int i = 0; i < db; i++)
        if (pStr[i][0] == ch)
            return pStr[i];
    return String("");
}
int main(int argc, const char *argv[]) {
    Stringek s1(argv, argc);  // indítási paraméterként kapott Stringek
    Stringek s2 = s1;
    s2 = s1 = s1;
    cout << s1.keres('C');    // 'C' betűvel kezdődő stringet keresünk.
}
```

String osztállyal szemben támasztott elvárások:

- legyen létrehozható C stringből
- legyen default konstruktora
- legyen másoló konstruktora
- legyen értékadás (operator=) operátora
- legyen index (operator[]) operátora, ami visszaadja az adott indexű karaktert
- kiírható legyen egy ostream objektumra