Nizovi

- Kao podrška konvenciji nasleđenoj iz jezika C da se nizovi znakova završavaju terminalnim znakom *'\0'*, a na koji se oslanjaju sve bibliotečne funkcije standardnih bilioteka koje rade sa nizovima znakova, svaki string literal ima još jedan, implicitan znak - taj terminalni znak *'\0'*:

```
"Hello"
```

String literal tipa *const char[6]* sa 6 znakova: *'H', 'e', 'l', 'l', 'o', '\0'*

- String literali imaju statičko trajanje skladištenja: postoje za sve vreme izvršavanja programa (alociraju se i inicijalizuju za vreme prevođenja)
- Pokušaj upisa u neki znak string literala, kao u element niza konstantnih znakova, ima nedefinsan ishod (taj niz znakova može biti smešten u deo memorije sa zabranjenim upisom, pa upis može izazvati hardverski izuzetak)
- String literali mogu da se koriste za inicijalizaciju nizova znakova; tada dati niz znakova ima poznatu dimenziju određenu dužinom znakovnog literala (uključujući i terminalni znak) i sadrži kopiju datog stringa:

```
char str[] = "Hello";
```

str je tipa *char[6]* čiji su elementi redom znakovi: *'H', 'e', 'l', 'l', 'o', '\0'*

Nizovi

- Dve potpuno nezavisne implicitne konverzije mogu da dovedu do problema ukoliko se nizovi objekata ne koriste na korektan način:
  - konverzija *T[]* u *T**, koja potiče iz jezika C sa ciljem efikasnosti
  - konverzija pokazivača na izvedenu u pokazivač na dostupnu osnovnu klasu (*Derived** u *Base**), koja je uvedena u jezik C++ kao podrška principu supstitucije
- Ove dve konverzije se mogu raditi i povezano, obe, u lancu implicitnih konverzija, i mogu da dovedu do sledećeg problema: funkcija koja kao parametar ima niz objekata osnovne klase (*Base[]*, odnosno *Base**), može se pozvati sa argumentom koji je niz objekata izvedene klase (*Derived[]*):

Derived[] ⇒ *Derived** ⇒ *Base**

```
class Base {
  public: int bi;
};

class Derived : public Base {
  public: int di;
};

void f (Base b[]) { cout<<b[2].bi; }

int main () {
  Derived d[5];
  d[2].bi=77;

  f(d);
}
```