Nizovi

- Osim toga, kada se objekti, pa i nizovi objekata, koriste po vrednosti, nema nikakve supstitucije:

```
Base b[5];
Derived d;
b[0]=d;
```

b[0] je objekat tipa *Base* i sadrži kopiju podobjekta osnovne klase iz objekta *d*

- Zbog ovoga, ukoliko se računa sa nasleđivanjem i supstitucijom, ne treba praviti nizove objekata (po vrednosti), već nizove pokazivača na objekte:

```
void f (Base* b[], int i) { cout<<b[i]->bi; }
...
Base b1,b2; Derived d1,d2,d3; d2.bi=77;
Base* b[5]; b[0]=&d1; b[1]=&b1; b[2]=&d2; ...
f(b,2);
```

Sada se ovo korektno računa, pošto su elementi niza pokazivači (čija veličina ne zavisi od tipa objekta na koji ukazuju)

- Sada navedena greška nije više moguća, jer se niz tipa *Derived*[]* može konvertovati u *Derived***, ali se to ne može konvertovati implicitno u *Base*** (što bi funkcija sa parametrom tipa *Base*[]* prihvatila):

```
Derived* d[5]; d[0]=&d1; d[1]=&d2; ...
f(d,2);
```

Greška u prevođenju: *Derived*[]* se ne može implcitno konvertovati u *Base***

Reference

- Motivacija:
  - Zbog orijentacije na to da se i objekti klasa i objekti neklasnih tipova koriste što sličnije, objekti klasa mogu se koristiti *po vrednosti*: kreirati u svim kategorijama životnog veka, kopirati, prenositi kao argumenti
  - Zbog toga postoji opisani problem beskonačne rekurzije za prenos argumenta konstruktora kopije
  - U skladu sa istim opredeljenjem, mogu se preklapati operatori za objekte klasa; kako postići istu notaciju kao za prenos po vrednosti, za operatore koji treba da menjaju neki od svojih operanada, ili ne žele da kopiraju operande, na primer:

```
complex c1, c2;
…
c1 += c2;
```

- Zato je na jeziku C++ moguć prenos argumenata *po referenci* (*call by reference*):

```
void f (int i, int &j) {
  i++;
  j++;
}

int main () {
  int si=0,sj=0;
  f(si,sj);
  cout<<"si="<<si<<", sj=“<<sj<<"\n";
}
```