# Semantika vrednosti od C++17

- To se u mnogim slučajevima svodi na obavezu izbegavanja kopiranja. Na primer, sledeća inicijalizacija formalno se tumači ovako:

```
X f () { return X(1); }
X x = X(X(f()));
```

  - Funkcija *f* u naredbi *return* vraća vrednost koja je čdvrednost i koja "predstavlja" inicijalizaciju nekog objekta (ali tek kada se on odredi kontekstom i ne obavezno privremenog) pozivom konstruktora *X(1)*
  - Ova čdvrednost prosleđuje se kao "zahtev" za inicijalizaciju ponovo čdvrednosti *X(f())*, dakle čdvrednost koju vraća *X(f())*, predstavlja istu prethodno navedenu inicijalizaciju; slično važi za još jednu čdvrednost *X(X(f()))*
    - najzad, tom čdvrednošću *X(X(f()))* inicijalizuje se objekat *x*, tako da se navedena inicijalizacija pozivom konstuktora *X(1)* konačno "vezuje" za objekat *x*

- Implementaciono, sve se opet svodi na isti mehanizam sprovođenja izostavljanja kopiranja RVO tako što pozvana funkcija, ukoliko vraća čdvrednost, dobija adresu mesta (objekta) koji treba da inicijalizuje u naredbi *return*, i ta adresa prosleđuje se tranzitivno

- Međutim, formalna semantika jezika je promenjena i sada konstruktori kopije i premeštanja ne moraju uopšte da budu definisani da bi ovo bilo moguće:

```
struct X {
  X (int);
  X (const X&) = delete;
  X (X&&) = delete;
};

X f () { return X(1); }

X x = f();
```

Do C++17 ovo ne bi bilo moguće. U C++17 je sasvim ispravno i radi *X x(1)*

# Semantika vrednosti od C++17

- Situacije kada se vrši materijalizacija privremenog objekta zapravo predstavljaju slučajeve u kojima se mora obezbediti mesto za objekat koji će se na kraju ovakvog lanca inicijalizovati onim što predstavlja čdvrednost i čija će adresa biti prosleđivana svim funkcijama koje se pozivaju, a vraćaju čdvrednost. Pored još nekih, to su sledeće situacije:
  - kada se referenca vezuje za čdvrednost:

```
const X rl& = X(1);
X rr&& = X(1);
```

  - kada se pristupa članu čdvrednosti tipa klase:

```
int i = X(1).i;
```

  - kada se niz koji je čdvrednost (npr. kao objekat član izraza koji je čdvrednost) konvertuje u pokazivač ili kada se pristupa njegovom elementu
  - kada je operand operatora *sizeof* čdvrednost
  - kada je čdvrednost izraz čiji se rezultat odbacuje (levi operand operatora zarez i izraz kao naredba):

```
X f() { return X(1); }
f();
```