Privremeni objekti

Do C++17, rezultat ovog eksplicitnog poziva konstruktora, kao i bilo koje konverzije u klasni tip, jeste privremeni objekat tipa *complex*, baš kao i rezultat svake od ovih operatorskih funkcija + i * čiji je povratni tip *complex*

- Pre verzije C++17, privremeni objekti obavezno su se pravili i u sledećim situacijama, pored još nekih:
  - konverzija koja ne vraća lvrednost, uključujući i eksplicitan poziv konstruktora; na primer:

```
complex c1(3.0,0.0), c2(0.0,4.0), c3;
c3 = (c1 + c2) * (complex(1.,0.) + c2);
```

Do C++17, rezultat inicijalizatorskog izraza sa desne strane znaka = je privremeni objekat tipa *complex* kojim se inicijalizuje objekat *c4* pozivom konstruktora kopije

  - prilikom inicijalizacije kopiranjem (*copy initialization*, inicijalizacija sa znakom =):

```
comlex c4 = (c1 + c2) * (complex(1.,0.) + c2);
```

  - kada se referenca inicijalizuje izrazom drugog, ali konvertibilnog tipa

- Pritom, za navedeni drugi slučaj inicijalizacije kopiranjem, kao i na nekim drugim mestima, prevodiocu se dopušta (ali se ne obavezuje) optimizacija *izostavljanja kopiranja* (*copy elision*): umesto da se najpre napravi privremeni objekat kao rezultat inicijalizacionog izraza, a potom njime inicijalizuje deklarisani objekat konstruktorom kopije, prevodilac može odmah da inicijalizuje deklarisani objekat rezultatom inicijalizacionog izraza, izostavljajući kopiranje, čak i ako konstruktor kopije ima vidljive bočne efekte, ali samo pod uslovom da su sve odgovarajuće funkcije, uključujući i konstruktor kopije, dostupne na mestu deklaracije (proveravaju se prava pristupa kao da se kopiranje ne izostavlja, tzv. *as-if* pravilo)

Privremeni objekti

- Počev od verzije jezika C++17, ova semantika je značajno izmenjena: rezultat poziva funkcije (uključujući i operatorske funkcije), baš kao i rezultat bilo koje operacije i izraza jeste *vrednost* (*value*), a ne privremeni objekat
- Vrednost je poseban entitet u programu i odnosi se na rezultat izraza koji se dalje može koristiti kao argument poziva funkcije. Na taj način se između poziva funkcija u izrazima prosleđuju vrednosti
- Pravljenje privremenog objekta se sada maksimalno odlaže do trenutka kada postane zaista neophodno, i vrši se samo u nekim situacijama; ovo se naziva *materijalizacija privremenog objekta* (*temporary materialization*); na primer, navedeno izostavljanje kopiranja kod inicijalizacije, kao i na mnogim drugim mestima, gde god je moguće kopiranje izostaviti, nije više dozvoljena, ali neobavezna optimizacija prevodioca, već je definisana semantika: na navedenom mestu se privremeni objekat *nikada* ne pravi
- Referenca (na lvrednost) se može inicijalizovati izrazom koji nije lvrednost samo ako je ta referenca na konstantu; tada se (i pre, i od verzije C++17), pravi privremeni objekat za koji se vezuje ta referenca:

```
complex& r1 = complex(1.,0.);
const complex& r2 = complex(1.,0.);
```

- Zbog ovoga, ako neka funkcija ima parametar koji je referenca (na lvrednost), ona treba da bude referenca na konstantu, inače se ta funkcija ne bi mogla pozvati sa argumentima koji to nisu:

```
complex operator+ (const complex&, const complex&);
...c1+complex(1.,0.)
```