C++ Problemi na koje treba obratiti pažnju

1. Poziv metoda preko nevalidnih pokazivača (2007.)

`pok->metodaKojaNeZavisiOdVrednostiAtributa()`
se može izvršiti čak iako `pok` ima slučajnu vrednost, ali se neće izvršiti ako je 0 == pok (ne verujem
da ce se ponoviti ista slicno, pošto je prošle godine bila baš takva greška u jednom zadatku, pa je bilo svašta na žalbama, ali vredi znati).

Nisam siguran da li je ovo možda implementaciono zavisno. Svakako bih dodao da ovo ima smisla razmatrati ako je u pitanju nevirtuelna metoda. Prema logici stvari, `this` je skriveni parametar metode, pa ako se ne koriste atributi – logično je pretpostaviti da se metoda može izvršiti čak iako pokazivač pokazuje na pogrešnu adresu u memoriji. U suštini, očekujem da se ponaša kao statička metoda. Postavlja se pitanje – zašto je to "netačno" ako `pok` ima vrednost 0? Zbog toga mislim da je ponašanje u ovom slučaju implementaciono zavisno, i verovatno se Visual C++ ovako ponaša.

Evo još jednog primera koji je krajnje diskutabilan:

```
void neka_funkcija(Tip &a)
{
      if( &a == 0 ) cout << "1";
      else cout << "2";
}
...
Tip *t = 0;
neka_funkcija(*t);
```

Šta će prevodilac uraditi kada se dereferencira t prilikom poziva? Kada sam ovo pokušao, delovalo mi je logično da može da se uradi, jer je u pitanju petljanje sa adresama (referenca je ništa drugo do posebna vrsta indirektnog adresiranja) i ne postoji stvarno dereferenciranje da bi se pristupilo nekom atributu ili pozvala neka metoda. Visual C++ prihvata ovakvu akrobaciju i izvršava je bez ikakvog problema: ponaša se kao da postoji "validan" objekat u memoriji sa adresom 0, na koji referiše formalni parametar funkcije, kao što sam očekivao. Ali – ako standard ne propisuje ovakvo ponašanje, to je onda implementaciono zavisno i ne bi smelo da se koristi.

2. Hvatanje izuzetaka, kada formalni parametar rukovaoca nije tipa reference

U ovakvim situacijama se poziva konstruktor kopije, kao da se radi o pozivu metode.
`try { ... throw T(); } catch(Tip t) { ... }`

3. Virtuelno izvođenje klasa

U sledećem primeru, klasa D je odgovorna za instanciranje (tj. pozivanje konstruktora) klase A u inicijalizatoru svojih konstruktora. Ako se ne poziva eksplicitno konstruktor klase A, onda se automatski poziva podrazumevani konstruktor klase A (koji onda mora da postoji).

```
class A;
class B : public virtual A;
class C : public virtual A;
class D : public B, public C;
```

4. Metode sa modifikatorom const

Modifikator ulazi u sastav potpisa metode. Ove metode se pozivaju kada je instanca klase (za koju se pozivaju) const objekat. Ako nije const objekat, a pozvaće se ako postoje obe varijante metode (const i ne-const), a objekat nije l-vrednost.