Pokazivači na objekte

- korumpiran pokazivač:

```
int a[5]; // array of 5 integers in range [0..4]
Clock* p = new Clock(…);
```

p je verovatno alociran u memoriji odmah iza niza *a*

U jednom kontekstu:

```
for (int i = 0; i<=5; i++) // i:=0..5
  a[i]=i;
```

Za poslednju vrednost *i==5* pregaziće pokazivač *p*

A onda u nekom potpuno drugom kontekstu:

```
p->tick();
```

p najverovatnije ima invalidnu vrednost

- Zaštita — pažljivo programiranje:
  - obavezna inicijalizacija, posebno pokazivača
  - provera granica prilikom pristupa nizovima, posebno baferima u koje se učitavaju podaci sa potencijalno neograničenim ili prevelikim sekvencama (provera na prekoračenje niza ili bafera, *buffer overflow*)

Pokazivači na objekte

- Dereferenciranje *null* vrednosti pokazivača:
  - Uzrok: pristup do objekta preko pokazivača koji ima *null* vrednost; vrednost se ne proverava za vreme izvršavanja prilikom dereferenciranja:

```
Clock* pc = nullptr;
pc->tick();
```

  - Efekti:
    - po pravilu greška za vreme izvršavanja, npr. izuzetak koji podiže hardver i prosleđuje operativni sistem (zbog nelegalnog pristupa delu memorije, tipično završava gašenjem programa
  - Tipične situacije u kojima nastaje:
    - neka funkcija koja treba da vrati pokazivač na objekat koji se zahteva, može i da ne vrati to, jer ne može da uradi to što se od nje traži, što je regularna situacija, ili je neregularna, ali funkcija ne podiže izuzetak (a trebalo bi):

```
Clock* pc = ClockFactory::getClock();
…
pc->tick();
```