Sa proceduralnog na OO programiranje: polimorfizam

- Jedan, očigledan i jednostavan način je sledeći:

```
int canMoveTo (Figure* fig, unsigned col, unsigned row) {
  switch (fig->kind) {

    case pawn:
      if (fig->color==white) …
      else …
      break;

    case bishop:
      …
      break;

    …
  }
}
```

- Problemi:
  - nepregledno, jer je obrada svih slučajeva smeštena u jedan glomazan *switch*, pa je podložno greškama
  - u nekom drugom slučaju, kada skup podvrsta objekata nije konačan i može se menjati i proširivati, nije lako dodavati nov slučaj, mora se menajti i ponovo prevoditi kod, i to na svim ovakvim mestima gde se radi grananje na osnovu vrste objekta

Sa proceduralnog na OO programiranje: polimorfizam

- Nešto pregledniji pristup — razdvojiti obradu svake situacije u zaseban potprogram:

```
int canMoveTo (Figure* fig, unsigned col, unsigned row) {
  switch (fig->kind) {

    case pawn: return canPawnMoveTo(fig,col,row);
    case bishop: return canBishopMoveTo(fig,col,row);
    …
  }

int canPawnMoveTo (Figure* fig, unsigned col, unsigned row) {
     if (fig->color==white) …
     else …
}

int canBishopMoveTo (Figure* fig, unsigned col, unsigned row) {
  …
}

…
```

- I dalje moramo menjati *switch* u slučaju da dodajemo novu vrstu objekata, i to na svim ovakvim mestima (polimorfnim operacijama)