Sa proceduralnog na OO programiranje: polimorfizam

- Drugi primer: pravimo program (na jeziku C) za igranje šaha na računaru. Skica delova:

```
enum FigureKind { pawn, bishop, knight, rook, queen, king };
enum FigureColor { white, black };

struct Figure {
  FigureKind kind;
  FigureColor color;
  unsigned posCol, posRow; // Current position
  …
};

int canMoveTo (Figure* fig, unsigned col, unsigned row);
```

- Kako implementirati fukciju *canMoveTo* (da li data figura može da se pomeri na dato polje)?

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