8.- Ghost entities

- Specifications and ghost constructs are used only during verification; the compiler omits them from the executable code.
- A field x of some type T can be declared to be a ghost field as:

```
ghost var x: T;
```

- Also parameters and results of methods can be declared to be a ghost by preceding the declaration with the keyword **ghost**.

- **lemma** is equivalent to **ghost method**.
- By default, a function (i.p. a predicate) is ghost, and cannot be called from non-ghost code.
- To make a function be non-ghost, replace the keyword **function** (respec. **predicate**) with the two keywords **function method** (respec. **predicate method**).
- Not every **predicate** can be converted into a **predicate method**. Problem: quantifiers (see file: predicate-method-problems.dfy)

- A ghost variable is useful for computing a value which allows to specify some interesting property, but that value is not really needed in the real code. For example:
  - a ghost variable could allows us to (easily) specify and prove a property.
  - termination proofs
  - to specify class invariants in OO programming
  - etc.
- Non-ghost variables cannot be calculated in terms of a ghost variable.
- In general, real code (boolean conditions of if, while, ...) cannot depend on ghost variables.