Data Structures and Algorithms with D

Jens K. Mueller

jkm@informatik.uni-jena.de

Department of Mathematics and Computer Science
Friedrich-Schiller-University Jena

Monday 6th May, 2013
Today’s lecture: Contract Programming
Contract Programming

Design by Contract
Function Signature and Contracts

- Type and number of arguments and return value
  
  \[
  \text{double sqrt(const (double) x)}
  \]

- Purity and Throwing
  
  \[
  \text{pure nothrow double sqrt(double x)}
  \]

The function signature imposes contracts on the caller/callee and verified at compile time.
Concepts in Contract Programming
Assertions, Preconditions, Postconditions, and Invariants

- **Assertion**
  Check against a condition (run time and compile time)
- **Precondition**
  Condition has to be fulfilled by the caller (parameters, resources)
- **Postcondition**
  Condition is guaranteed by the callee (assuming normal return)
- **Invariant**
  Condition that holds during a computation
Contracts in D

```d
pure noexcept double sqrt(double x) in {
    assert(x >= 0, "x has to be greater equal than 0");
}
out(result) {
    assert(x == result * result ,
        text("Square of ", result, " does not equal ", x));
}
body {
    // ...
}
```
Contracts in D (cont.)

// invariant in a class/struct
struct MyStruct {
    invariant() {
        // ...
        // ...
    }
}
Enforcement of Contracts

- Optionally checked (enabled by compiler switch)
- Contracts needs to be written such that you can remove them without changing the logic of the program
  Don’t write

```c
assert(++x > y)
```

- Contracts in non-machine readable form are of less use (e.g. in form of documentation)
Enforcement of Contracts (cont.)

- **Execution**
  - Precondition validated prior to calling the function
  - Postcondition validated after successful execution of the called function
  - Invariant validated before and after any public interface function call. Explicitly via

```c
MyStruct myStruct;
assert(myStruct);
```
Contracts and Inheritance

- Precondition in subtype is only allowed to widen (weaken) the contract
  Preconditions are \( \|\)-ed. Empty precondition is considered true (accept all).

- Postcondition in subtype is only allowed to restrict (strengthen) the contract
  Postconditions are \&\&-ed. Empty postcondition is considered true (no restriction by default).

- All invariants must be fulfilled
  Invariants are \&\&-ed. Empty invariant means true.

Consequence of Liskov substitution principle: Objects of a subtype \( S \) of type \( T \) can be substituted anywhere a \( T \) is expected while preserving the program’s properties (introduced by Barbara Liskov).
References