“Design by Contracts”

- Eiffel and Bertrand Meyer
- Requirements/Constraints
- Typing introduces constraints on a programs behaviour
- Constrained parameter in Ada
- Preconditions in C/C++
- Design by Contracts in C/C++
- The `assert` function in C/C++
- Invariants for an Object or a Class
- Defensive Programming
- Summary
- Some Related Web Links
Eiffel and Bertrand Meyer

• Bertrand Meyer and his associated designed the advanced programming language Eiffel in the mid 1990’s.
• Eiffel has a number of advanced compile time and run time checking mechanisms.
• Design by Contracts is included in Eiffel but can also be used in adapted forms in other programming languages like Ada, C/C++, Java, …
Requirements/Constraints

- Regular requirements state what the system shall be able to do
- Constraint requirements state what the system must not be able to do
  - Constraints are one of the most important tools in design for safety and reliability
  - Constraints are used to prevent unwanted behaviour
Typing introduces constraints on a program's behaviour

- **A C++ example:**
  ```cpp
  int count; // restricts count to be an integer
  ```

- **An Ada example:**
  ```ada
  count : is range 1 .. 10;
  -- restricts count to be an integer in range 1 .. 10
  ```

*In Ada it is possible to define constraints in variables by means of the programming language in a way that cannot be done in e.g. C/C++.*
Constrained parameter in Ada

procedure F ( P is range 1 .. 10 );
   -- restricts P to be an integer in range 1 .. 10
begin
   ...
   end F;

In Ada it is possible to define constraints in parameters by means of the programming language in a way that cannot be done in e.g. C/C++.
Preconditions in C/C++

```c
int f ( int p)
{
    require ( p >= 1 and p <= 10 );
    // restricts p to be in range 1 .. 10
    ...
}
```

By means of ”require” preconditions in C/C++ it is possible to define constraints in variables and parameters also in C/C++. 
Design by Contracts in C/C++

```c
int f ( int p )
{
    require ( p >= 1 and p <= 10 );
    . . .
    check ( x > 15 );
    . . .
    ensure ( r > 0 and r < 100 );
    return r
}
```
The *assert* function in C/C++

- The contract clauses *require*, *check* and *ensure* all interrupt the normal execution and direct the execution to some exception handling code and they usually have the same underlying code.
- In C/C++ an *assert* function is used in all three cases.
Invariants for an Object or a Class

An invariant is a condition that is maintained consistently.
An example in Eiffel syntax:

```eiffel
invariant
  0 <= count
  count <= capacity
```
Defensive Programming

• Design by Contracts is a way to implement a consistent strategy for "Defensive Programming"
Summary

• Specifying and implementing constraint requirements (what the system shall not do) is one of the most important strategies in building reliable and safe systems.

• At the code level, constraints can be implemented by:
  – Strong typing if the programming language supports strong typing
  – "Design by Contracts" for constraints that otherwise could not be implemented just by programming language constructs
Some Related Web Links

• Building bug-free O-O software:
  An introduction to Design by Contract™

• Design by Contract Framework for C++

• Design by Contract™ for Java™ Using JMSAssert™
  – http://www.mmsindia.com/DBCForJava.html