Defensive Programming

Assertion and Exception Handling

Indranil Saha

Department of Computer Science and Engineering
Indian Institute of Technology Kanpur

Dealing with Garbage Inputs

- Check the values of all data from external sources
- Check the values of all routine input parameters
- Decide how to handle bad inputs

Assertion

- An assertion is code that is used during development – usually a routine or macro – that allows a program to check itself as it runs
- When an assertion is true, that means everything is operating as expected
- When it is false, that means it has detected an unexpected error in the code
- **Example:** If the system assumes that a customer information file will never have more than 50,000 records, the program might contain an assertion that the number of records is less than or equal to 50,000
    - As long as the number of records is less than or equal to 50,000, the assertion will be silent
    - If it encounters more than 50,000 records, however, it will loudly “assert” that an error is in the program

Assertion

An assertion usually takes two arguments

- A boolean expression that describes the assumption that is supposed to be true
- A message to display if it is not

Example: Java Assertion

Java Assertion

```
assert denominator != 0 : "denominator is unexpectedly equal to 0.";
```

Example: C++ Assertion

C++ Assertion

```
#include <stdio.h>
#include <assert.h>

int main()
{
    int x = 7;

    /*  Some big code in between and let's say x
       is accidentally changed to 9  */
    x = 9;

    // Programmer assumes x to be 7 in rest of the code
    assert(x==7);

    /* Rest of the code */

    return 0;
}
```

Output

```
Assertion failed: (x==7), function main, file assertion1.cpp, line 14.
```

Exception and Error Handling

- Use exceptions to notify other parts of the program about errors that should not be ignored
- Throw an exception only for conditions that are truly exceptional
- Avoid empty `catch` block
- Know the exceptions your library code throws
- Handle error condition locally without throwing an exception if possible
- Shutdown the system if appropriate

Robustness Vs. Correctness

- **Correctness:** Never return an inaccurate result
  - returning no result is better than returning an inaccurate result
- **Robustness:** Always try to do something that will allow the software to keep operating, even if that leads to results that are inaccurate sometimes
- Depends on the kind of software
  - Safety-critical applications vs consumers applications

Assertion Vs. Exception

- Assertions check for conditions that should never occur
- Error-handling code checks for off-nominal circumstances that might not occur very often, but that have been anticipated by the programmer who wrote the code and that need to be handled by the production code
- Error handling typically checks for bad input data; assertions check for bugs in the code

Defensive Programming

Assertion and Exception Handling

Indranil Saha

Department of Computer Science and Engineering
Indian Institute of Technology Kanpur