Google Testing Blog

TotT: Floating-Point Comparison

Thursday, October 16, 2008

If your code manipulates **floating-point** values, your tests will probably involve **floating-point** values as well.

When comparing floating-point values, checking for equality might lead to unexpected results. Rounding errors can lead to a result that is close to the expected one, but not equal. As a consequence, an assertion might fail when checking for equality of two floating-point quantities even if the program is implemented correctly.

The Google C++ Testing Framework provides functions for comparing two floating-point quantities up to a given precision.

In C++, you can use the following macros:

```
ASSERT_FLOAT_EQ(expected, actual);
ASSERT_DOUBLE_EQ(expected, actual);
EXPECT_FLOAT_EQ(expected, actual);
EXPECT_DOUBLE_EQ(expected, actual);
```

In Java, JUnit overloads Assert.assertEquals for floating-point types:

```
assertEquals(float expected, float actual, float
delta);
assertEquals(double expected, double actual, double
delta);
```

An example (in C++):

```
TEST(SquareRootTest, CorrectResultForPositiveNumbers)
{
  EXPECT_FLOAT_EQ(2.0f, FloatSquareRoot(4.0f));
  EXPECT_FLOAT_EQ(23.3333f,
FloatSquareRoot(544.44444f));
  EXPECT_DOUBLE_EQ(2.0, DoubleSquareRoot(4.0));
  EXPECT_DOUBLE_EQ(23.33333333333333,
DoubleSquareRoot(544.44444444444444));
    // the above succeeds
  EXPECT_EQ(2.0, DoubleSquareRoot(4.0));
    // the above fails
  EXPECT_EQ(23.33333333333333,
DoubleSquareRoot(544.44444444444444));
}
```

Remember to download this episode of Testing on the Toilet and post it in your office.

Labels: C++ , TotT

No comments :

Post a Comment

The comments you read and contribute here belong only to the person who posted them. We reserve the right to remove off-topic comments.

Enter your comment...

Comment as: Unknown (Goo ▼ Sign out

Publish Preview Notify me