Logic Coverage CppUTest

Presented By

Ahmed Saleh
Omar Osama
Mohamed Hamada
Mina Magdy

Agenda

- About CppUTest
- Logic Coverage
- Project Demo
- References

About CppUTest

- C/C++ based unit xUnit test framework
- Simple to use and small
- Portable to old and new platforms

1. Test Macros

```
TEST_GROUP(FirstTestGroup)
{
};

TEST(FirstTestGroup, Test1)
{
    FAIL("Fail me!");
}

TEST(FirstTestGroup, Test2)
{
    STRCMP_EQUAL("hello", "world");
}
```

```
TEST_GROUP(SecondTestGroup) {
    int x;
    void setup() {
        x = 5;
    }
};

TEST(SecondTestGroup, Test1) {
    CHECK(x == 5);
    x++;
}

TEST(SecondTestGroup, Test2) {
    CHECK(x == 5);
    x++;
}
```

2 tests passed

2. Assertions (some macros)

- CHECK(boolean condition)
- CHECK_TEXT(boolean condition, text)
- CHECK_FALSE(condition)
- CHECK_EQUAL(expected, actual)
- CHECK_THROWS(expected_exception, expression)
- LONGS_EQUAL(expected, actual)
- UNSIGNED_LONGS_EQUAL(expected, actual)
- STRCMP_EQUAL(expected, actual)
- STRNCMP_EQUAL(expected, actual, length)
- STRCMP_NOCASE_EQUAL(expected, actual)
- STRCMP_CONTAINS(expected, actual)
- POINTERS_EQUAL(expected, actual)
- DOUBLES_EQUAL(expected, actual, tolerance)

Project Demo

- We worked on some classes from our competitive programming library
- Tested classes
  - Some data structures
    - Fenwick Tree
  - Math utilities

Example 1

Testing math function indicates happy numbers.
Happy number in his definition that has 2 or more features (even, has perfect square root or lucky i.e. all digits are either ‘4’ or ‘7’).

```
if ((hasSqrt && isEven) || (isLucky && (hasSqrt || isEven))) return true;
else return false;
```

#	isLucky	isEven	hasSqrt	P	P(isLucky)	P(isEven)	P(hasSqrt)	Input
1	True	True	True	True	False	False	False	4
2	True	True	False	True	True	True	False	74
3	True	False	True	True	True	False	True	-
4	True	False	False	False	False	True	True	7
5	False	True	True	True	False	True	True	16
6	False	True	False	False	True	False	True	2
7	False	False	True	False	True	True	False	81
8	False	False	False	False	False	False	False	3

Example 1 (cont.)

```
TEST_GROUP(Math_Happy) {

};

TEST(Math_Happy, Test1) {
    CHECK(checkHappy(74));
}

TEST(Math_Happy, Test2) {
    CHECK_FALSE(checkHappy(7));
}

TEST(Math_Happy, Test3) {
    CHECK(checkHappy(16));
}

TEST(Math_Happy, Test4) {
    CHECK_FALSE(checkHappy(2));
}
```

Example 2

Testing operator[] in multiset implemented using Fenwick tree.

```
if (idx < 1 || idx > cnt) {
    throw out_of_range("ERROR :: trying to access an out of range element");
}
```

#	idx < 1	idx > cnt	P	P(idx<1)	P(idx>cnt)	idx
1	True	True	True	False	False	-
2	True	False	True	True	False	0
3	False	True	True	False	True	cnt + 1
4	False	False	False	True	True	1

Example 2 (cont.)

```
TEST_GROUP(FenwickTree_Operator) {
    fenwick_multiset fm;
    int n;
    void setup() {
        n = MAX_N;
        for (int i = 0; i < n; ++i)
            fm.insert(rand() % MAX_V + 1);
    }
    void teardown() {

    }
};

TEST(FenwickTree_Operator, Test1) {
    CHECK_THROWS(out_of_range, fm[0]);
}
```

```
TEST(FenwickTree_Operator, Test2) {
    int ret;
    try {
        ret = fm[1];
    }
    catch (exception e) {
        FAIL("test fail");
        return;
    }
    LONGS_EQUAL(ret, fm[1]);
}


TEST(FenwickTree_Operator, Test3) {
    CHECK_THROWS(out_of_range, fm[n + 1]
}
```

Example 3

Testing insert function in multiset implemented using Fenwick tree.

```
if (val < 1 || val > N) {
    throw exception("ERROR :: invalid value to insert");
}
```

#	val < 1	val > N	P	P(val<1)	P(val>N)	val
1	True	True	True	False	False	-
2	True	False	True	True	False	0
3	False	True	True	False	True	N + 1
4	False	False	False	True	True	1

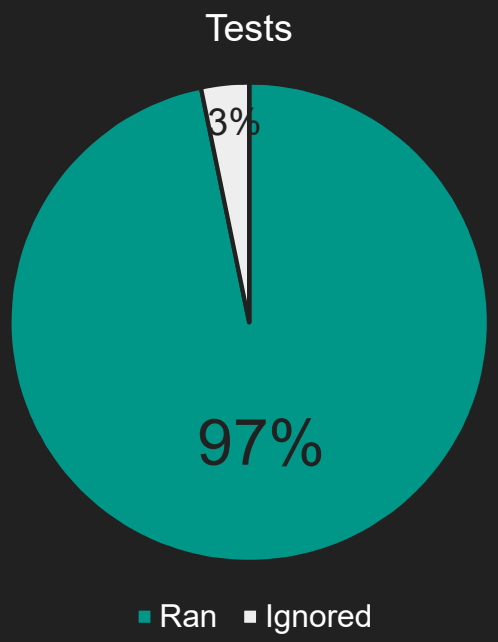
Example 3 (cont.)

```
TEST_GROUP(FenwickTree_Insert) {
    fenwick_multiset fm;
    int n;
    void setup() {
        n = MAX_N;
        for (int i = 0; i < n; ++i)
            fm.insert(rand() % MAX_V + 1);
    }
    void teardown() {

    }
};

TEST(FenwickTree_Insert, Test1) {
    CHECK_THROWS(out_of_range, fm[0]);
}
```

```
TEST(FenwickTree_Insert, Test2) {
    int ret;
    try {
        ret = fm[1];
    }
    catch (exception e) {
        FAIL("test fail");
        return;
    }
    LONGS_EQUAL(ret, fm[1]);
}
```

```
TEST(FenwickTree_Insert, Test3) {
    CHECK_THROWS(out_of_range, fm[n + 1]);
}
```

Statistics



References

https://cpputest.github.io/
https://cpputest.github.io/manual.html

Any Question ?

Thank You