**a、需求分析**

    通过单元测试代码，测试程序加减乘除功能是否正确工作，对于格式错误、数值溢出、除零异常是否能检测的到。

**b、设计测试框架，模拟测试数据**

* 测试加减乘除功能

TEST\_METHOD(TestAdd) //测试加法

{

Test.CheckInput("1/2+4");

Assert::AreEqual(test11,Test.add());

}

TEST\_METHOD(TestAdd1)

{

Test.CheckInput("3+1/3");

Assert::AreEqual(test12, Test.add());

}

TEST\_METHOD(TestAdd2)

{

Test.CheckInput("2+4");

Assert::AreEqual(test13, Test.add());

}

TEST\_METHOD(TestAdd3)

{

Test.CheckInput("1/2+1/4");

Assert::AreEqual(test14, Test.add());

}

TEST\_METHOD(TestSub) //测试减法

{

Test.CheckInput("1/2-1/3");

Assert::AreEqual(test21, Test.sub());

}

TEST\_METHOD(TestSub1)

{

Test.CheckInput("1/3-2");

Assert::AreEqual(test22, Test.sub());

}

TEST\_METHOD(TestSub2)

{

Test.CheckInput("2-1/3");

Assert::AreEqual(test23, Test.sub());

}

TEST\_METHOD(TestSub3)

{

Test.CheckInput("5-3");

Assert::AreEqual(test24, Test.sub());

}

TEST\_METHOD(TestMul) //测试乘法

{

Test.CheckInput("1×2");

Assert::AreEqual(test31, Test.mul());

}

TEST\_METHOD(TestMul1)

{

Test.CheckInput("1/2×2");

Assert::AreEqual(test32, Test.mul());

}

TEST\_METHOD(TestMul2)

{

Test.CheckInput("2×1/3");

Assert::AreEqual(test33, Test.mul());

}

TEST\_METHOD(TestMul3)

{

Test.CheckInput("1/3×1/2");

Assert::AreEqual(test34, Test.mul());

}

TEST\_METHOD(TestDiv) //测试除法

{

Test.CheckInput("1/2÷2");

Assert::AreEqual(test41, Test.div());

}

TEST\_METHOD(TestDiv1)

{

Test.CheckInput("4÷1/2");

Assert::AreEqual(test42, Test.div());

}

TEST\_METHOD(TestDiv2)

{

Test.CheckInput("1/2÷2/3");

Assert::AreEqual(test43, Test.div());

}

TEST\_METHOD(TestDiv3)

{

Test.CheckInput("1÷2");

Assert::AreEqual(test44, Test.div());

}

* 测试格式错误、数值溢出、除零异常功能

TEST\_METHOD(TestCheckFormat1) //测试格式错误

{

Test.CheckInput("2--2");

Assert::AreEqual(test5, Test.g\_szErrIn);

}

TEST\_METHOD(TestCheckFormat2)

{

Test.CheckInput("2+÷2");

Assert::AreEqual(test5, Test.g\_szErrIn);

}

TEST\_METHOD(TestCheckFormat3)

{

Test.CheckInput("2+×2");

Assert::AreEqual(test5, Test.g\_szErrIn);

}

TEST\_METHOD(TestCheckFormat4)

{

Test.CheckInput("2+×");

Assert::AreEqual(test5, Test.g\_szErrIn);

}

TEST\_METHOD(TestCheckFormat5)

{

Test.CheckInput("2+/2");

Assert::AreEqual(test5, Test.g\_szErrIn);

}

TEST\_METHOD(TestCheckNum61) //测试非法数值

{

Test.CheckInput("1÷0");

Test.div();

Assert::AreEqual(test6, Test.g\_szErrNum);

}

TEST\_METHOD(TestCheckNum62)

{

Test.CheckInput("1/0");

Test.div();

Assert::AreEqual(test6, Test.g\_szErrNum);

}

TEST\_METHOD(TestCheckNum63)

{

Test.CheckInput("1/2+2/0");

Test.add();

Assert::AreEqual(test6, Test.g\_szErrNum);

}

TEST\_METHOD(TestCheckBey71) //测试时数值溢出

{

Test.CheckInput("-2000+2");

Test.add();

Assert::AreEqual(test7, Test.g\_szErrBey);

}

TEST\_METHOD(TestCheckBey72)

{

Test.CheckInput("-2000");

Test.add();

Assert::AreEqual(test7, Test.g\_szErrBey);

}

TEST\_METHOD(TestCheckBey73)

{

Test.CheckInput("-200+2/3000");

Test.add();

Assert::AreEqual(test7, Test.g\_szErrBey);

}

* 单元测试加减乘除功能

string Calc::add() //测试加法

{

size\_t iPos = g\_szInput.find('+');

g\_szX = g\_szInput.substr(0, iPos);

g\_szY = g\_szInput.substr(iPos+1, g\_szInput.length()-1-iPos);

size\_t iPosInX = g\_szX.find('/');

g\_iFirNer = stoi(g\_szX.substr(0, iPosInX));

if (iPosInX == -1)

{

g\_iFirDer = 1;

}

else

{

g\_iFirDer = stoi(g\_szX.substr(iPosInX+1, g\_szInput.length() - 1 - iPosInX));

}

size\_t iPosInY = g\_szY.find('/');

g\_iSecNer = stoi(g\_szY.substr(0, iPosInY));

if (iPosInY == -1)

{

g\_iSecDer = 1;

}

else

{

g\_iSecDer = stoi(g\_szY.substr(iPosInY+1, g\_szInput.length() - 1 - iPosInY));

}

simplify(g\_iFirNer, g\_iFirDer, g\_iSecNer, g\_iSecDer);

g\_iRusNer = g\_iFirNer\*g\_iSecDer + g\_iFirDer\*g\_iSecNer;

g\_iRusDer = g\_iFirDer\*g\_iSecDer;

init(g\_iRusNer, g\_iRusDer);

g\_szRusDer = "";

g\_szRusNer = "";

if (g\_iRusDer == 1)

{

g\_szRusNer += to\_string(g\_iRusNer);

return g\_szRusNer;

}

else{

g\_szRusDer += to\_string(g\_iRusDer);

g\_szRusNer += to\_string(g\_iRusNer);

return g\_szRusNer + "/" + g\_szRusDer;

}

}

string Calc::sub() //测试加法

{

size\_t iPos = g\_szInput.find('-');

g\_szX = g\_szInput.substr(0, iPos);

g\_szY = g\_szInput.substr(iPos + 1, g\_szInput.length() - 1 - iPos);

size\_t iPosInX = g\_szX.find('/');

g\_iFirNer = stoi(g\_szX.substr(0, iPosInX));

if (iPosInX == -1)

{

g\_iFirDer = 1;

}

else

{

g\_iFirDer = stoi(g\_szX.substr(iPosInX + 1, g\_szInput.length() - 1 - iPosInX));

}

size\_t iPosInY = g\_szY.find('/');

g\_iSecNer = stoi(g\_szY.substr(0, iPosInY));

if (iPosInY == -1)

{

g\_iSecDer = 1;

}

else

{

g\_iSecDer = stoi(g\_szY.substr(iPosInY + 1, g\_szInput.length() - 1 - iPosInY));

}

simplify(g\_iFirNer, g\_iFirDer, g\_iSecNer, g\_iSecDer);

g\_iRusNer = g\_iFirNer\*g\_iSecDer - g\_iFirDer\*g\_iSecNer;

g\_iRusDer = g\_iFirDer\*g\_iSecDer;

init(g\_iRusNer, g\_iRusDer);

g\_szRusDer = "";

g\_szRusNer = "";

if (g\_iRusDer == 1)

{

g\_szRusNer += to\_string(g\_iRusNer);

return g\_szRusNer;

}

else{

g\_szRusDer += to\_string(g\_iRusDer);

g\_szRusNer += to\_string(g\_iRusNer);

return g\_szRusNer + "/" + g\_szRusDer;

}

}

string Calc::mul() //测试乘法

{

size\_t iPos = g\_szInput.find('×');

g\_szX = g\_szInput.substr(0, iPos);

g\_szY = g\_szInput.substr(iPos + 1, g\_szInput.length() - 1 - iPos);

size\_t iPosInX = g\_szX.find('/');

g\_iFirNer = stoi(g\_szX.substr(0, iPosInX));

if (iPosInX == -1)

{

g\_iFirDer = 1;

}

else

{

g\_iFirDer = stoi(g\_szX.substr(iPosInX + 1, g\_szInput.length() - 1 - iPosInX));

}

size\_t iPosInY = g\_szY.find('/');

g\_iSecNer = stoi(g\_szY.substr(0, iPosInY));

if (iPosInY == -1)

{

g\_iSecDer = 1;

}

else

{

g\_iSecDer = stoi(g\_szY.substr(iPosInY + 1, g\_szInput.length() - 1 - iPosInY));

}

simplify(g\_iFirNer, g\_iFirDer, g\_iSecNer, g\_iSecDer);

g\_iRusNer = g\_iFirNer\*g\_iSecNer;

g\_iRusDer = g\_iFirDer\*g\_iSecDer;

init(g\_iRusNer, g\_iRusDer);

g\_szRusDer = "";

g\_szRusNer = "";

if (g\_iRusDer == 1)

{

g\_szRusNer += to\_string(g\_iRusNer);

return g\_szRusNer;

}

else{

g\_szRusDer += to\_string(g\_iRusDer);

g\_szRusNer += to\_string(g\_iRusNer);

return g\_szRusNer + "/" + g\_szRusDer;

}

}

string Calc::div() //测试除法

{

size\_t iPos = g\_szInput.find('÷');

g\_szX = g\_szInput.substr(0, iPos);

g\_szY = g\_szInput.substr(iPos + 1, g\_szInput.length() - 1 - iPos);

size\_t iPosInX = g\_szX.find('/');

g\_iFirNer = stoi(g\_szX.substr(0, iPosInX));

if (iPosInX == -1)

{

g\_iFirDer = 1;

}

else

{

g\_iFirDer = stoi(g\_szX.substr(iPosInX + 1, g\_szInput.length() - 1 - iPosInX));

}

size\_t iPosInY = g\_szY.find('/');

g\_iSecNer = stoi(g\_szY.substr(0, iPosInY));

if (iPosInY == -1)

{

g\_iSecDer = 1;

}

else

{

g\_iSecDer = stoi(g\_szY.substr(iPosInY + 1, g\_szInput.length() - 1 - iPosInY));

}

simplify(g\_iFirNer, g\_iFirDer, g\_iSecNer, g\_iSecDer);

g\_iRusNer = g\_iFirNer\*g\_iSecDer;

g\_iRusDer = g\_iFirDer\*g\_iSecNer;

init(g\_iRusNer, g\_iRusDer);

g\_szRusDer = "";

g\_szRusNer = "";

if (g\_iRusDer == 1)

{

g\_szRusNer += to\_string(g\_iRusNer);

return g\_szRusNer;

}

else{

g\_szRusDer += to\_string(g\_iRusDer);

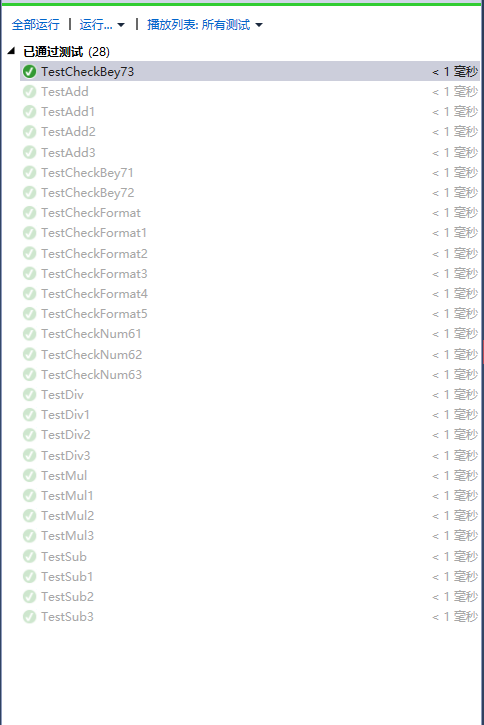
g\_szRusNer += to\_string(g\_iRusNer);

return g\_szRusNer + "/" + g\_szRusDer;

}

}

**c、测试结果**



**PSP**

|  |  |  |  |
| --- | --- | --- | --- |
| PSP2.1 | Personal Software Process Stages | Estimated time（h） | actual time（h） |
| Planning | 计划 | 1 | 1.5 |
| · Estimate | 估计这个任务需要多少时间 | 10 | 15 |
| Development | 开发 | 8 | 10 |
| · Analysis | 需求分析 (包括学习新技术) | 0.5 | 1 |
| · Design Spec | 生成设计文档 | 2 | 2 |
| · Design Review | 设计复审 | 0.3 | 0.5 |
| · Coding Standard | 代码规范 | 0.5 | 0.4 |
| · Design | 具体设计 | 1.5 | 1.3 |
| · Coding | 具体编码 | 8 | 7 |
| · Code Review | 代码复审 | 0.5 | 1 |
| · Test | 测试（自我测试，修改代码，提交修改） | 0.2 | 0.5 |
| Reporting | 报告 | 2 | 2 |
| · | 测试报告 | 0.5 | 0.5 |
| · | 计算工作量 | 0.5 | 1 |
| · | 并提出过程改进计划 | 2 | 4 |