### OOP第三周作业文档

2019010175 孔瑞阳 土木92

### 功能说明

不断进行以下操作：输入一个0-5的自然数表示操作类型。

1) 输入一个三角形的边长。

2) 输入一个正方形的边长。

3) 输入一个正五边形的边长。

4) 输入一个正六边形的边长。

5) 输入一个圆形的半径。

0) 结束程序，输出结果。

结束后输出输入的图形总个数、所有图形的总周长、所有图形的总面积。

### 模型

##### 输入与输出

首输入和输出采用C++标准库中的输入输出流cin和cout进行。

在实现过程中，发现课上所讲的输入一个正整数中的如下语句：

while (m\_data <= 0)

{

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cin >> m\_data;

}

并不能通过直接把<=0改成<0来完成输入一个自然数的程序，因为在输入一个不是数字的数据后，m\_data的值依然是0，这在正整数中算是非法输入，但是在自然数中是合法的。

经过查阅资料后，采用以下方式实现：

while (m\_data < 0 || cin.rdstate() == ios\_base::failbit)

##### 正多边形的周长和面积计算

记正多边形的边数为n，边长为l。

则周长：。

对于面积，我们考虑把正n边形分成n个相邻顶点为底角、中心点为顶角的三角形。



则顶角的角度是，则可以得到高。

一个三角形的面积是，则总面积。

l

h

运用cmath库中的三角函数实现，其中cot使用1/tan实现。

##### 圆形的周长和面积计算

记圆形的半径为r。

则周长：。

面积：。

##### 图形总个数、总周长和总面积的统计

记使用三个计数器polygonNumber，totalPerimetre，totalSquare记录如上信息。

每次输入一个图形之后即进行相应的修改。

### 软件构件介绍

|  |  |
| --- | --- |
| **文件** | **功能介绍** |
| CP\_NaturalInteger.h/cpp | 实现自然数的输入和操作 |
| CP\_PositiveReal.h/cpp | 实现正实数的输入和操作 |
| CP\_Circle.h/cpp | 有关于圆的信息、操作和计算 |
| CP\_RegularPolygon.h/cpp | 有关于正多边形的信息、操作和计算 |
| regularPolygonTest.h/cpp | 对于圆、正多边形的计算的测试 |
| regularPolygonMain.cpp | 主程序 |

### 单元测试

##### 输入单元的测试

**对于输入一个自然数操作符opt：( CP\_NaturalInteger.h/cpp )**

1. 等价类划分
2. 输入的不是0-5的某个自然数。
3. 输入的是0-5的某个自然数。
4. 测试：

|  |  |  |
| --- | --- | --- |
| **等价类** | **选取案例** | **输出结果（下一步操作）** |
| 等价类① | HelloWorld! | 您输入的不是自然数，请输入一个自然数： |
| 1919.810 | 您输入的不是自然数，请输入一个自然数： |
| -10 | 您输入的不是自然数，请输入一个自然数： |
| 12 | 您输入的自然数超过了5，请输入一个自然数： |
| 等价类② | 1 | 请输入正三角形的边长，请输入一个正实数： |
| 2 | 请输入正方形的边长，请输入一个正实数： |
| 3 | 请输入正五边形的边长，请输入一个正实数： |
| 4 | 请输入正六边形的边长，请输入一个正实数： |
| 5 | 请输入圆形的半径，请输入一个正实数： |
| 0 | 输入的图形总个数是……（程序结束） |

**对于输入一个正实数边长/半径：( CP\_PositiveReal.h/cpp )**

1. 等价类划分

① 输入的不是正实数。

② 输入的是正实数。

1. 测试：

|  |  |  |
| --- | --- | --- |
| **等价类** | **选取案例** | **输出结果（下一步操作）** |
| 等价类① | HelloWorld! | 您输入的不是正实数，请输入一个正实数： |
| -114.514 | 您输入的不是正实数，请输入一个正实数： |
| 等价类② | 1 | 请输入一个0-5……（进入下一步操作） |
| 1919.810 | 请输入一个0-5……（进入下一步操作） |

##### 2、周长与面积的验证 **( CP\_RegularPolygon.h/cpp , CP\_Circle.h/cpp )**

测试单元：**regularPolygonTest.h/cpp**

按照图形的种类来划分等价类。

**第一步：选取边长/半径为1的图形进行计算，并与理论值进行比较。**

正三角形：

正方形：

正五边形：

正六边形：

圆形：

测试：

采用如下语句进行测试：

CP\_RegularPolygon rp(i, 1);

cout << rp.perimeter() << ' ' << rp.square() << endl;

CP\_Circle cc(1);

cout << cc.perimeter() << ' ' << cc.square() << endl;

测试结果：

|  |  |  |
| --- | --- | --- |
| **图形** | **边长/半径** | **周长 面积** |
| 正三角形 | 1 | 3 0.433013 |
| 正方形 | 1 | 4 1 |
| 正五边形 | 1 | 5 1.72048 |
| 正六边形 | 1 | 6 2.59808 |
| 圆形 | 1 | 6.28319 3.14159 |

与预测值相同，初步测试成功。

**第二步：利用第一步的结果进行对拍。**

由于边长/半径为1的情况我们已经验证正确，并且根据理论，边长/半径为a的图形的周长应该是其的a倍，面积是其的倍，那么可以每次随机出一个图形和相应的边长/半径，计算其的周长和面积是否是相应边长/半径为1的图形的相应倍数，当两个值的相对误差小于时我们即认为是相等的。

运行regularPolygonTest2()，在一分钟之内程序没有发现错误暂停。事实上，在一分钟内计算机可以进行上亿次验证，基本可以验证程序的正确性。

##### 3、计数器的测试**(** regularPolygonMain.cpp **)**

为了考虑测试正确性以及非法输入对于程序的可能影响，设计一组输入数据如下：

其中，红色表示非法输入，运用测试2.1中的数据进行测试，理论输出应该为：

个数：5 总周长：24.2832 总面积：8.89316

|  |  |
| --- | --- |
| **输入** | **输出** |
| HelloWorld!  3  1  2  1  4  HelloWorld!  -10  1  203  1  0  1  5  1  12  0 | 输入的图形总个数是：5  所有图形的总周长是：24.2832  所有图形的总面积是：8.89316 |

与理论输出相同，由于计算的过程已经在2中得到验证，那么可以认为计数器实现没有问题，并且不会受到非法输入的影响。

### 文档其他部分

##### 如何充分进行单元测试

1. 保证测试的全面性，不能出现某个单元没有测试到的情况。确保所有的基本功能 都可以正确实现。
2. 保证测试数据覆盖了全部等价类，考虑特殊情况，保证程序没有考虑某些情况。

甚至在可以的情况下，测试数据覆盖所有情况。

1. 对于边界值进行测试，防止边界情况出现溢出。
2. 对于某些输入的非法情况进行测试，防止程序因此而崩溃。

##### C语言与C++在本例中的程序扩展性差异

首先，如果要直接扩展，由于在C语言中数据和处理的过程是独立的，所以在扩展的过程中对于原来的过程的依赖性很高，从而使得程序的耦合性变得很高，难以维护和扩展。而C++进行扩展时，对于对象本身的扩展是非常简便的，并且如果仅是要扩展出一个对于本对象进行数据处理的程序，不需要再去调用其他的过程，直接对于对象进行数据处理即可，耦合性较低。

其次，在测试的过程中，使用C++面向对象的方法进行测试，可以直接利用过程来检测对象的特性，但如果用C语言来实现，则会出现：测试过程（Test）->测试对象（多边形）->调用测试对象的过程（计算）->调用测试对象的另一个数据（多边形的数据）->……这样的情况，使得扩展之后的单元测试更加麻烦，也从另一方面降低了C语言程序的可扩展性。