03016417 张家齐

三、设计工作小结

Programming is about managing complexity in a way that facilitates change. There are two powerful mechanisms available for accomplishing this: decomposition and abstraction`

Apply abstraction and decomposition to solve more complex problems

- decompose a large problem into parts and design algorithms to solve them
- recognise similar problems, and apply generic solutions and abstractions
- creating algorithms to obtain the generic solution results

The set of problem-solving methods with computer is also called Computational Thinking.

【译文】:

编程是促进管理复杂性变更的一种方式。它有两种强大的机制可用于实现此目的：分解和抽象

应用抽象和分解来解决更复杂的问题：

✓将大问题分解成部分并设计算法来解决它们

✓识别类似问题，并应用通用解决方案和抽象

✓创建算法以获得通用解决方案结果

计算机问题解决方法的集合也称为计算思维。

这次作业要求我们编写一个通用朗肯循环模拟器，在作业二的基础上又有了提升。使其不仅仅局限于一个特定的朗肯循环，而是具有了普遍性和一般性。只要将朗肯循环的节点和设备参数以 json 文件的形式输入，便可输出结果。

在这次作业中，我也深刻领会了用分解与抽象的方法来解决复杂的问题：

对于一个朗肯循环，其节点之多，设备之庞大，我们很难从整体角度去分析。那么，我们可以通过分解的方法，将整个朗肯循环分解成各个设备部件，通过计算各个设备的参数、能量，进而得到整个朗肯循环的能量信息。

在一个朗肯循环中，有很多的节点；不同额朗肯循环也有具有不同的设备。如此，我们可以通过抽象的方法，建立节点类和设备类，将具体的问题抽象化，使抽象的模型能够使用于各个不同的具体事例。

这次作业使我有了进一步的提升，通过参考老师所给的案例，做起来也比较轻松。其实，这次作业给我们提供的是解决问题的一种方法——利用分解和抽象的方法将具体的问题模型化、算法化，这样，就能起到“举一反三”的效果。我们还是要多思考、多编码，程序的开发过程，远比结果重要。

03016403 刘祎璇

1. 设计工作小结

Programming is about managing complexity in a way that facilitates change. There are two powerful mechanisms available for accomplishing this: decomposition and abstraction.

Apply abstraction and decomposition to solve more complex problems.

- ✓ decompose a large problem into parts and design algorithms to solve them
- ✓ recognise similar problems, and apply generic solutions and abstractions
- ✓ creating algorithms to obtain the generic solution results

The set of problem-solving methods with computer is also called Computational Thinking.

【翻译】

编程是关于以促进变更的方式管理复杂性。有两种强大的机制可用于实现此目的：分解和抽象。

应用抽象和分解来解决更复杂的问题。

- ✓ 将大问题分解成部分并设计算法来解决它们
- ✓ 识别类似问题，并应用通用解决方案和抽象
- ✓ 创建算法以获得通用解决方案结果

计算机问题解决方法的集合也称为计算思维。

这次的程序主要还是在老师的例程的基础上进行改进的。整个程序看上去一目了然，十分的清晰。整个程序是在找到设备的相似点之后抽象成一个个的类，每需要进行一项功能的时候，就循环该设备的相关函数。程序的后续升级维护都非常的方便，如果需要，只需要自己继续添加相关设备的类就可以了。

相比于练习 2 中的程序，可以看得出来，应用类之后，程序的抽象化更加的明显了，同样的，程序也更加的通用了。练习 2 中的程序，更符合我们解题时的思路，将步骤规划好以后，设计一个一个的函数。而类的加入，则将这一个个的函数剥离出来，放到每一个设备的类中，将程序打散，也提高了程序的健壮性。

这样抽象化的程序里面类里套着类，函数一层层调用的思路能够使得程序更加的通用，更加的容易升级维护。不过，这并不是自己一朝一夕能够写的出来的，后面还需要更多的练习！

03016301 张佳钰

四、 遇到的问题：

开始的过程都很顺利，按部就班的添加了 rankine86 的 json 文件，新增了 reheater，trap，openheater，closedheater 四个设备的类，在 rankine_cycle 文件中引入了新增的设备类模块和 comdict 字典。

然而，在我以为大功告成的时候，运行时得出现了错误

报错如下：

```
Traceback (most recent call last):
  File "e:/pythonha/P3/rankine.py", line 39, in <module>
    cycle[i].CycleSimulator()
  File "e:\pythonha\P3\rankine_cycle.py", line 235, in CycleSimulator
    self.cycle.cycleSimulator()
  File "e:\pythonha\P3\rankine_cycle.py", line 141, in cycleSimulator
    self.Comps[key].simulate(self.nodes)
  File "e:\pythonha\P3\components\condenser.py", line 61, in simulate
    (nodes[self.inNode].h - nodes[self.outNode].h)
TypeError: unsupported operand type(s) for *: 'NoneType' and 'float'
```

在 turbine 设备调用 simulate（）函数时，它的节点的 fdot 值出现了错误。

一遍又一遍的检查了自己的代码，却找不出问题所在，花费了许多时间。

然后我在 RankineCycle 的 cycleFdot 函数中添加了

```
for key in self.nodes:
    print(key.fdot)
```

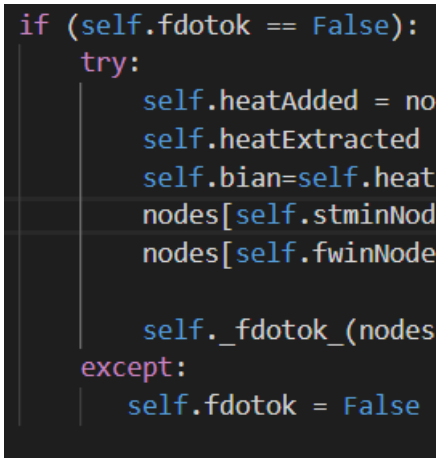
得到

```
1.0
0.14970428650697784
0.8502957134930221
0.8502957134930221
None
None
None
None
1.0
1.0
1.0
0.14970428650697784
0.14970428650697784
```

发现流经 openheater 的节点没有值，再次查看了 openheater 的 fdot（）函数的代码，

```
(Nodes[self.trinNode].h-nodes[self.fwinNode].h)
```

终于发现，有一个 nodes 被我写成了 Nodes，正是一个字母的大小写弄错导致我得不出结果。而编译器没有指出这个错误，因为代码在”try except”结构里，发生了错误后直接跳到了 except 的代码块中。



通过这个很幼稚的错误，我发现写代码时认真书写重要，但如何发现问题出在哪里更加重要，一开始我没有目的的检查真的浪费了很多时间，所以今后我要多多练习 debug 的方法和技能。

03016327-张崇辉

3 设计工作小结

Programming is about managing complexity in a way that **facilitates** change. There are two powerful mechanisms available for accomplishing this: decomposition and abstraction
编程就是以一种**方便**拓展的方式来管理复杂性。实现这一点有两种强大的机制：分解和抽象

Apply abstraction and decomposition to solve more complex problems
应用抽象和分解来解决更复杂的问题

decompose a large problem into parts and design algorithms to solve them
将一个大问题分解成若干部分，并设计解决这些问题的算法

recognise similar problems, and apply generic solutions and abstractions
识别类似的问题，并应用通用的解决方案和抽象

creating algorithms to obtain the generic solution results
创建算法以获得通用解结果

The set of problem-solving methods with computer is also called Computational Thinking.
用计算机解决问题的一套方法又称计算机思维

这次作业是第二次作业的进一步实现，作业的目的是计算出一个非常复杂的带有再热以及二级抽气的朗肯循环，同时还希望该程序可以解决 8.1，8.5。这就需要我们对问题进行抽象和分解，以实现针对不同的问题，程序调用不同的模块解决。分解是把问题分解成一个个小问题，把不同的设备分解到不同的类。抽象：是把不同的问题抽象为相同或者相似的描述方式，比如高压缸，低压缸都属于 TurbineEx1。

我们将整个庞大的计算过程分解为：设备部分，节点部分，循环计算部分，以及最后总的函数部分。同时输入的 json 文件中包含识别所属类的信息，这样一来复杂的问题就变成了针对各个设备的计算和最后的汇总。同时针对不同的问题只需要自动识别不同的设备计算即可。

这样这个复杂的问题就被解决了，事实上，针对更加复杂的朗肯循环我们只需要在程序中加入未被包含的部分，就可以解决更多的问题。这就体现了程序的可扩展性和通用性。