申优文章

18231047王肇凯

以下是一些关于如何完成课程设计的分享，包括我所遇到的问题和自己的解决方法

C++编程语言不熟悉

解决方法：

看菜鸟教程自学C++语法：由于大二下学期OO课学过另一门面向对象语言Java，可以从两门语言的不同之处入手进行自学；

此外在实际上手过程中，一个好的开发环境也能帮助我们边用边学，e.g.我强推Clion（Jetbrains，yyds！），由于mac端VS不能开发C++而从Clion上手，实际发现体验相当不错，特别是在语言特性不熟悉的时候可以通过实时的warning（Clang-Tidy检查）和suggestions来了解一些效率高、安全、简洁的写法，以完成对于新语言的入门。举例：函数参数不变时可以变为`const string &`，只使用一次时可以使用`std::move`来节省内存拷贝等等。

调试困难

实际上从测试数据来说，编译相对于之前的两门系统类课程计组和OS来说调试难度已经是比较低了，而且课程组提供了相当多的测试库，具有很高的覆盖范围，其他课程的盲debug现象很少会再出现。

然而这门课比较复杂的地方在于整体步骤比较繁琐：以往的课程是编译你的源代码为目标程序——目标程序读入输入产生输出——比对输出，而这门课是：编译你的编译器源代码——你的编译器编译测试数据为目标代码（其中还经过了中间代码的环节）——Mars运行目标代码读入输入产生输出——比对输出

所以导致发现一个bug时，需要先从错误输出定位到错误的mips代码，再回去查生成的mips是否有误，再回去查询中间代码是否有误，整体上比较花费时间。

如下是一些可以尝试的方法：

- 实现自动化评测：编写脚本以命令行形式完成编译、调用mars的过程
- 对拍：找已经AC的同学要来程序进行对拍，进行黑箱测试（比对输出的步骤也可以写在评测脚本里面）
- 多去讨论区发言，每个人分享遇到的问题和解决方法，就可以塑造一个比较良好的（反内卷的）学习氛围
- 注重单元测试等测试方法，例如先实现最基本功能保证没有bug再进行优化等
- 造数据：特别是做优化时，例如加完函数内联应该主动构造一些函数调用的样例进行测试，比大规模黑箱测试再定位问题要节省时间
- 手写调试方法：如panic、assert、全局DEBUG开关等

编译器实现原理不熟悉

解决方法：

还是要注重理论课的学习，这门课的理论课和实验课联系最为紧密，尤其是词法分析语法分析可以直接用书上代码的实现方式来完成，后面的优化算法更是和课上内容紧密相连。因此平时上课和作业都要认真对待。

整体设计没有头绪，经常推倒重构

解决方法：

在动手写代码之前进行设计的时候多和其他人讨论设计方案，以免到了实现时才发现设计上的缺陷，以至推倒重来。

也可以参考往届学长的设计文档进行整体架构设计。

以下有一些设计实现上的建议：

- 解耦合：设计成多遍处理，源程序=词法分析=>词法分析结果=语法分析语义分析=>中间代码=目标代码生成=>目标代码
- 面向对象：OO不能白学，不能再一main到底了，从可读性、调试难度、可扩展性来说都不好
- 多封装：实际上也就是面向对象的主要思想（毕竟编译器也用不到继承和多态），把常用的方法封装起来，好处如下：
    - 节省时间：少写很多代码
    - 防止出错：很多错误是复制粘贴的地方忘了改，封装起来再改参数就直观多了
    - 方便调试：直接注释掉函数调用语句就可以删去整个行为逻辑；也方便增加cout等调试语句
- 多用STL：C++比C好的一个方面就是有丰富的数据结构，比静态数组高到不知道哪里去了

版本混乱，改着改着就改乱了

解决方法：

使用git进行版本控制，建立private的github仓库，同步到远程进行备份；有新的想法可以新建分支编写，写乱了可以直接`git reset --hard`回退到之前的版本。