# 编译技术课程设计申优文章

**16061109 马振亚**

历时2个多月的编译课设终于接近了尾声，总结反思两个多月过来，压力很大，收获也很大。在中间遇到过很多困难，也遇到过很多障碍。每次遇到障碍额时候就总是想放弃，但是自己也会发现再多坚持一下就能够找到折中或者是彻底解决的办法。下面我将就完成课设的过程中遇到的典型问题和一些启发式想法简要描述一下。

### 早期准备—分析题意，确定宏观架构与编程语言

C0文法的作业要求是先将高级语言设计成中间代码，然后中间对中间代码进行多多轮次迭代优化，然后将中间代码翻译成最终的汇编代码。本部分内容看似跟项目没有直接的关系，但是隐式地影响了整个项目编写的效率。

拿到文法以及作业要求之后，首先第一步要做的就是确定编程语言以及编程结构。由于我们大一以来学院一直强调c语言基础，所以可能第一反应想到的就是使用c语言完成相应的项目任务。但是仔细思考一下，C语言虽然有运行速度快的先天优势，但是灵活性较差的特点也会给项目的编写带来不小的阻力。比如c语言的字符串是依托字符数组实现，和其他数组一样会有长度不可变的情况产生，在编程的时候需要时刻警惕溢出的情况。而且c语言本身内置的数据结构以及方法并不是很充足，比如在符号表经常需要的查表以及插入新元素的操作，如果采用链表或者数组的时候不能保证两个方面都能去的很好的性能。如果采用C++的话，map数据结构就提供了简单而且高效的接口。再如优化部分的数据流分析记录的是变量的集合，如果采用C++的set结构管理起来要比c语言自主实现速度快而且性能更高。

所以我认为花一小部分时间学习一下c++ STL类库里面相关数据结构的能后节约特别多的编程以及调试时间。

确定了编程语言之后，下一步就要决定程序的文件组织形式，个人认为编译课设的情景特点并不适合全篇使用面向对象编程，更多的部分还是强调过程，但是因为过程比较繁杂需要我们分文件书写，这时候头文件就需要好好加以利用。进过前期调研发现有头文件编程以及头文件声明主文件定义这两种写法，头文件编程的好处是直接在头文件中像正常在C语言编程那样书写代码即可。但是头文件编程的缺点到了后期也很明显，后期随着代码量增加，单独进行一次编译可能要花费相当长的时间，头文件编程方法只要有一个头文件更改会触发整个编译重新进行。头文件只声明的好处在后期编译的时候会根据make的依赖关系，只重新编译依赖修改项的文件，大大缩短了重新编译的时间。但是缺点就是需要在头文件声明，然后cpp文件中定义。

在整个项目完成过程中IDE也是我的痛，早期一直使用数据结构课程使用的codeblocks，但是到优化阶段发现codeblock的调试功能远远满足不了需求，最后不得不转向使用VS。

以上三个因素因人而异，但是我觉得会在隐形中对工程设计编写实现的效率产生极大的影响，希望下一届的学弟学妹能认真考虑清楚这些因素再编程。

### 阶段1—词法分析部分

词法分析部分难度比较小，主要就是按照Pascal-S的例子以及我们课本上所讲的算法进行实现，这里不在赘述。

### 阶段2—语法分析阶段

### 问题2.1 文法改写与函数名/变量名识别

可以发现我们所给的文法是不符合LL文法的，FIRST集合之间有较多的相交，解决方法就是按照课程所讲的转换为FIRST集合不相交的若干子句，这一步是应用不带回溯的递归下降子程序法的前提条件。

在我的文法中出现了int/char 后可以跟[无参函数名] [变量名] [有参数函数名]这三种成分，所以，所以这个部分如果直接使用文法改写会出现对于原文法成分的太多的修改，所以对于这种特殊情况采用向后跳读两个词素，即读到标识符后面那个单词，如果是分号或者逗号或者左方括号就是变量名，如果是左圆括号就是有参数函数名，如果是左大括号就是无参函数名。这种跳读只是为了判断一下读到int/char之后需要走向哪个子程序分支，因此向后偷偷读几个单词之后还需要将文件指针回退到读完int的位置。

### 阶段3—语义分析阶段

### 问题3.1—DO FOR的翻译

抽到这个文法我感到非常悲伤，因为do-for结构可以说是这几个循环中最复杂的结构了，翻译起来的难度最大(因为do-while的先要做的do体出现在前面，但是do-for的是判断体出现在前面，但是不能先判断)。我们看一下这个结构

for(p1;p2;p3){p4},

正确的执行顺序是

p1

for:p4

p3

bnz p2

所以在处理这一部分的时候需要将for中的条件判断和while或者if的区分开，因为for读到判断条件p2的时候不能直接生成中间代码，需要将代码暂时存储起来再处理。

### 问题3.2—表达式值类型是char还是int

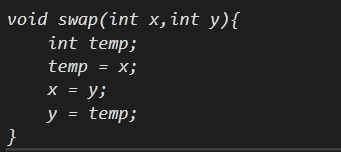
由于最后做了要求我们要判断一个成分是不是字符，这个决定了是否能进行赋值以及输出的时候输出形态。重点在于判断表达式的类型，为此我们给a\_exp()子程序中传入了一个引用变量is\_char,这个变量会随着表达式-.>项->因子逐层向下分析，最后决定因素在因子段，但是即使因子是char出现了因子之间的乘除或者是项之间的加减也认为时发生了char到int的类型转换。通过层层分析得到的值向上传递就得到了最后的分析结果。

### 阶段4—代码优化阶段

优化这部分并没有像之前基础部分那么规范全面的算法描述，我觉得这也是老师有意在锻炼我们的创新能力。的确这个过程中收获了很多新科研方法以及新的想法。

### 问题4.1—拓展优化之死代码删除

在程序中经常会遇到许多代码无法执行，或者执行后对程序运行结果不产生任何影响，比如一个c语言初学者经常会犯错误的swap例子:



这个代码执行完后其实不会产生任何影响,因为我们的传参是值传递。编译程序需要对基本块进行活跃变量分析才能确定是否能作为死代码进行删除。具体做法是在基本块中按照每条指令从后向前做活跃变量分析(最后一条指令的out为基本块的out)，到达当条指令时如果out中不包含当条指令定义的变量且当条不是scanf语句(这是一个坑点，虽然输入是无用的，但是不能删掉),则删除这条语句，并且迭代多次重新分析，直到不能再删除新的代码为止。

### 问题4.2 如何构建冲突图

书上并没有给出构建冲突图的算法，都是形式化描述，经过思考之后发现可以借鉴上面的删除死代码的分析流程，具体算法如下:

**对于基本块B，首先令live = out[B]**

**然后对于B中的指令I，从后往前，开始遍历**

**对于任意变量x属于def(I)，对于任意变量y属于live**

**冲突图添加边(x,y)**

**然后更新live = use(I) U (live-def(I))**

总结一下自己的编译课设之旅，说句实话在我看来这个过程的难受程度是高于计算机组成原理课程的。我认为两者的最根本区别是计组更像是初级课程，虽然难度也大，但是方法是固定的，老师已经准备了非常详尽的教程，我么需要做的更多的比葫芦画瓢。但是编译有一部分算法并不是很完备的，需要我们独立思考和调整。我认为作为一个高年级学生更应该培养自己的创新能力，课程组刚好也提供给我们这样的训练，这些都以后即将走向科研之路的我们打下一个良好的基础。