在《数据结构》教学中开展程序理解的实践探讨

张昱

（中国科学技术大学计算机科学技术系，合肥，230026，yuzhang@ustc.edu.cn）

摘　要：为提高学生对 C 语言知识的理解和编程能力，促进学生更好地学习数据结构课程，笔者在本系 2004 级学生的数据结构教学中，开展了 Gzip 程序理解的实践活动。本文阐述该活动的开展意义、方法和效果，总结实践中的经验教训，以供同行参考。

在 C 语言版数据结构的教学中，学生对 C 语言的掌握程度是影响学生对数据结构学习兴趣和理解程度的重要因素之一。C 语言中的指针、结构体、宏定义、复杂的带参函数等是 C 语言版数据结构表示和实现的基础，学生在学习 C 语言时对这部分知识的理解和运用往往比较欠缺。再加上学生是在学习 C 语言之后又间隔一个学期才学习数据结构，致使学生在开始学习数据结构时 C 语言能力普遍比较弱。

为增强学生对 C 语言的理解和运用，更好地促进对数据结构课程的学习，笔者在本系 2004 级本科生的数据结构教学中，开展了 Gzip 程序理解的实践活动。虽然学生在实践中遇到种种困难和挫折，但是最终或多或少都能有一些收获。

1 程序理解的引入与定位

根据笔者十年来在 C 语言及数据结构等课程的教学实践经验以及软件研发经验，笔者认为学生在学习 C 语言时要强调概念的正确理解和编程实践两个方面。强调前者是因为不论多复杂的 C 程序都跑不出其语言定义的范围；强调后者是因为实践性的课程要在实践中学习才能更深入地领会其内涵。

就编程实践而言，笔者鼓励学生从编写很小的程序入手，这样一方面降低编译出错的几率，另一方面便于出错时的错误定位，从而使学生在短时间内可以将程序调试成功，获得成就感，增强学习的自信心。但是，笔者强调学生所编写的小程序应尽可能多地蕴涵不同的知识点，这样便于学生能通过一个小程序领会更多的知识。例如下面这个小程序：

```
#include <stdio.h>
void main(){
   int (*pf)() = scanf, a;
   printf("%d\n", pf("%d", &a));
}
```

它包含文件包含预编译命令、指向函数的指针以及输入输出函数等知识点。

对 C 语言能力弱的学生来说，自己设计实现蕴涵多个知识点的小程序是比较困难的。那么怎样让学生更快更多地掌握 C 语言呢？笔者想：如果让学生阅读理解一些较大的 C 程序，这样学生就可

以在此过程中见识许多自己从未使用过的 C 语言知识点，试着理解其含义，进而可以照葫芦画瓢，编写类似的程序来消化对知识点的领会。为此，笔者决定将程序理解引入到数据结构教学实践中。

考虑到程序理解毕竟是数据结构教学中为提高学生 C 语言能力而引入的辅助实践活动，教师和学生都不可能对这一活动投入许多时间。因此，笔者对这一活动的定位是：教师给出一些阅读的线索，如重要的变量、宏、算法、数据结构等，同时介绍程序阅读的方法和工具(如 SourceInsight)；而学生通过小组合作和文献调研与实验来理解所给的线索，并进一步自主地挖掘更多的程序理解点。程序理解的任务是弹性的，但是它的目标是明确的，即促进学生：1) 进一步学习 C 语言；2) 学习如何从数据结构和流程出发阅读比较大的源代码；3) 熟练使用程序阅读和编译调试工具；4) 尝试与人合作，增强团队意识；5) 学做电子讲稿并演讲，展示自己的成果；6) 了解数据结构在实际中的使用；7) 自主地学习，主动询问，多思考。

2 Gzip 程序的选择

该选择什么样的 C 程序让学生阅读理解呢？笔者认为所选择的程序应：

1） 有一定的规模。代码量不能很大，也不能过小，可以选择几十至几百 K 字节的。

2） 包含 C 中各种知识点，尤其是学生理解上比较薄弱的知识点。如宏定义、条件编译、指针、结构体、带参的 main 等。

3） 具有比较简单的功能。

4） 涉及比较少的算法，且易于理解。

5） 使用尽可能多的数据结构种类。

6） 具有良好的编码风格和适当的注释。

这样，可以使学生将程序理解的重点放在领会各种 C 知识点及数据结构的运用上，尽可能地降低因程序本身的功能和算法的复杂性所带来的理解上的困难。

基于上述原则，笔者选取开源的GNU项目Gzip程序[1]来开展程序理解活动。Gzip是一个提供压缩(编码)和解压缩(译码)功能的程序，其 1.2.4 版本的源代码大小约为 250K。Gzip程序的核心算法是 LZ77 数据压缩算法[2][3]的一个变种以及静态或动态Huffman编码方法，算法种类少且易于理解。另外，Gzip程序使用C语言中较为复杂的宏、条件编译、结构体、指针、文件操作等，并且含有线性表、串、Huffman树、Hash表等数据结构。代码风格较好，且有一定的注释。

3 Gzip 程序的算法思想

Gzip 对要压缩的文件，首先使用 LZ77 算法的变种进行压缩，再对得到的结果使用静态或动态 Huffman 编码进行压缩。

LZ77 算法是 Ziv 和 Lempel 于 1977 年提出的，它是基于字符串匹配的第一个具有实用价值的数据压缩算法。LZ77 算法是建立在如下朴素认识基础上：待编码的数据符号串可能包含在已编码的信息结构中，这样根据已编码的数据完全可以构造出后续与之相同的数据。从而压缩问题就转变成在已编码的历史数据(保存于历史表)中寻求与待编码的数据相同的符号串，然后，在压缩码中仅存储

在历史表中获得成功匹配的数据段的起点位置及其长度。LZ77 算法是建立在工作缓冲区 buffer[0..n-1]这个数据结构上，该缓冲区划分为历史表 buffer[0..p-1]和输入超前缓冲区 buffer[p..n-1]两部分。在该缓冲区上，编码过程概述为：

1) 初始化历史表，并从输入中读取 q 个字节的数据装入超前缓冲区中；
2) 在历史表中寻找与超前缓冲区中字符串的最大匹配(设 addr 记录匹配串在历史表中的起始位置，len 表示匹配串的长度)；
3) 按照某种约定将 addr、len 及匹配串的最后一个字符一起形成一个压缩代码送入输出位流中；
4) 将 buffer 中的数据左移 len 个元素，并从输入中截取 len 个字节的数据填补到超前缓冲区的尾部；
5) 重复 2 直至无输入数据。

译码是编码的逆过程，此时可以不使用超前缓冲区。译码主要是遵循编码约定从压缩码中正确解释 addr 和 len，再从历史表中 addr 处开始顺序向输出数据队列中复制 len 个字符。

Gzip 对 LZ77 算法进行了改进，其压缩码为二元组(distance，length)，distance 表示当前串和前一匹配串的距离(不超过 32K)，length 表示串的长度(不超过 258)。它用一个 32K 的缓冲区 window 和一个 hash 表代替工作缓冲区，初始时先输入 32K 字节填充 window，再用 hash 表来寻找匹配串，所有长度超过 3 的串都插入 hash 表(实际记录该串在 window 中的起始位置)。Hash 表是以**链地址法**解决冲突。Hash 链上的第一个元素是最近插入该位置的串，链上的其他元素按插入时间依次存放。Gzip 提供 9 种级别的压缩。在级别为 4~9 的压缩中，采用惰性计算机制寻找匹配，而在级别为 1~3 的压缩则采用快速压缩方法。压缩后输出的 literals、length 和 distance 将用 Huffman 树再次压缩，其中，literals 和 length 用一棵 Huffman 树压缩，distance 用另一棵 Huffman 树压缩。

4 实践活动的开展

2005-2006 学年第一学期，笔者在本系 2004 级本科生《数据结构》课程教学中开展了 Gzip 程序理解实践，共 158 名学生参加了这次活动。为便于管理与考核，同时培养学生的团队合作精神，所有学生自由组成 25 个小组，每组有 5~8 名学生。

为引导学生循序渐进地阅读程序，笔者将程序理解分成以下四个阶段：

第一阶段的任务是：1)了解 Gzip 的功能及命令行参数。2)熟悉 VC++集成环境下 Gzip 工程的创建、编译和运行，包括：①熟悉断点的设置与调试(debug)；②学会在 debug 状态下，查看变量的值、内存的状态、函数调用栈等信息；③学会输入、输出的重定向；④对于能力强的学生，可以进一步学习 makefile 的编写，进行命令行方式下的编译和运行；等等。3)阅读 Gzip 源代码中对命令行参数的处理，掌握带参的 main 的编程与运行。4)学习使用 SourceInsight 工具阅读源代码，该工具通过多视窗及上下文相关的方式使学生能从多个维度理解程序。在这一阶段开始前，由助教给学生做了关于 Gzip 的编译、使用和主要流程以及 SourceInsight 等工具的使用的讲座。

第二阶段的任务是：1)理解一些标识符的含义和意义，包括 OF、EXTERN、DECLARE、ALLOC、

FREE 等宏以及 work。例如带参的宏 OF 可以使程序既能适应编译器要求函数声明带参数原型的情况，又能适应不带参数原型的情况；“int (***work**) OF((int infile, int outfile)) = zip;”中的 work 是指向函数的指针变量，这样就可以统一处理对文件的压缩或解压缩(见 gzip.c 中的 treat_file 函数)。2)理解文件及其操作，如变量 ifname、ofname 表示输入输出文件名，函数 fileno、setmode、perror、OPEN、fflush、fgets、fprintf、read、opendir、closedir 等提供文件的基本操作，函数 treat_file、treat_stdin 等提供对文件或 stdin 的压缩或解压缩等。3)理解输入缓冲区 inbuf 及相关变量 inptr、insize 和相关操作 clear_bufs、get_byte、fill_inbuf、try_byte 等。4)进一步理解其他缓冲区 outbuf、d_buf 和 window。这一阶段的主要目的是理解 Gzip 的处理基础(即文件和缓冲区)，并且理解 main 函数的总体处理流程。

第三阶段的重点是理解 trees.c，其中包含 huffman 树的定义 tree_desc、一些全局量如 l_desc 等、以及对树的处理函数如 build_tree 等。在这一阶段，需要借助调研和教师提供的资料来消化 Gzip 的原理和实现。

第四阶段的重点是理解以LZ77 和Huffman编码为基础的deflate压缩算法 (见deflate.c) [4][5]，并理解其中的hash表及其处理。这一阶段的目标是较清晰地了解Gzip的压缩以及解压缩的处理过程，并对整个程序理解工作进行总结。

学生在 9、10 月份进行第一、二阶段的理解，并在 10 月中下旬接受对这两阶段工作的检查。10 月下旬到 12 月上旬，学生进行第三、四阶段的理解，并在 12 月中旬接受检查。

由于学时有限而学生又多，教师不可能一个个检查学生的理解情况，也不可能让每一组在课堂上汇报本组的理解状况。那么，怎样才能相对客观地考核学生在这一活动中的成绩呢？针对此，笔者采取的做法如下：

1) 以小组为单位，提交 2 次理解报告(可以是 ppt)，一次是对第一、二阶段的总结，一次是最后对整个程序理解的总结。

2) 每位学生在活动结束时提交个人总结并给出对此活动的看法和建议。

3) 规定各小组提交报告的截止时间。在收到各小组报告后，及时将报告整理放在课程主页上，并利用课堂教学时间随机抽几组同学进行汇报和回答问题。

4) 主讲教师、助教和学生根据报告以及平时的了解对学生进行评分。评分的比例是：30%是各组学生互评得分，同一组中的各学生在这部分的得分上是相同的；30%是组内成员互评得分，通过这部分分数能相对客观地了解每个学生对小组的贡献；40%是主讲老师和助教对各学生的评分。

5) 明确规定对迟交、不参加阅读和不参加评分等情况的考核方法。

5 实践活动的总结

毋庸质疑，开展程序理解活动会给学生增加负担，但是这个负担到底有多大呢？为此，笔者在实践中要求学生在报告中说明每一学生个人的投入时间以及小组讨论时间。以学生在第一、二阶段投入的时间为例，共有 16 组在其报告中陈述了所投入的时间(如图 1，横轴为组号，纵轴为小时数)，大多数学生所花的时间不到 10 小时，有 5 组的小组讨论时间不足 1 小时。由此可见，学生不太习惯小组讨论，团队合作意识薄弱。

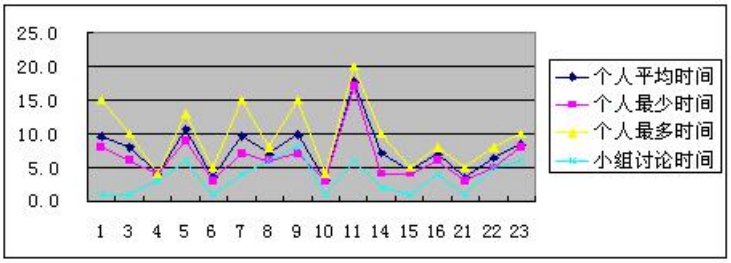

图 1　各组学生在第一、二阶段投入的小时数

根据学生所写的报告，很多学生在初看 Gzip 程序时有些懵了，因为他们从未读过这么大的程序。通过笔者和助教给的提示和讲座，一些学生可以较顺利地开展阅读。但是，对 C 语言能力很弱的学生来说，往往只能局部地理解标识符和数据结构的含义，不能全局地理解 Gzip 程序。他们希望老师能给予更具体的指导，带着他们循序渐进地阅读，而这一点因课时数的限制是难以做到的。

学生希望安排稍微有点难度但不至于无从下手的程序来开展阅读。笔者同意这个观点，但遗憾的是笔者尚不能找到更合适的程序，因此在本学期仍然继续给 2005 级的学生开展 Gzip 程序理解。

尽管一些学生认为自己的收获不大，但是都指出通过这个活动让他们学到了以前所不知道或不会的一些C语言知识(如库函数、条件编译等)和工具，也让他们认识到自己知识的局限性和工作的辛苦，认识到工作是需要有耐心和毅力的，认识到团队合作的意义，认识到英语的重要性，……。因篇幅所限，这里不能一一说明。读者可以通过http://staff.ustc.edu.cn/~yuzhang/ds/2005/gzip.html查看这次实践活动的情况，其中给出了学生提交的报告和总结等。

程序理解活动目前尚处在尝试阶段，需要在实践中找到不足并改进。笔者希望以此文来引起国内同行对这一活动的讨论，从而能促进我国计算机基础教育的发展。

参考文献

[1] Gzip主页, http://www.gzip.org/.

[2] Jacob Ziv, Abraham Lempel. A Universal Algorithm for Sequential Data Compression. IEEE Transactions on Information Theory, Vol. 23, No. 3: 337-343. http://citeseer.ist.psu.edu/ziv77universal.html.

[3] 王忠效, 姜丹. 关于 Lempel—Ziv77 压缩算法及其实现的研究. 计算机研究与发展, 1996, 33(5):329-340.

[4] P. Deutsch. DEFLATE Compressed Data Format Specification version 1.3. Network Working Group, May 1996. http://www.ietf.org/rfc/rfc1951.txt.

[5] Antaeus Feldspar. An Explanation of the DEFLATE Algorithm. August 1997. http://www.gzip.org/deflate.html.