Software Engineering and Applications 软件工程与应用, 2018, 7(3), 168-176
Published Online June 2018 in Hans. http://www.hanspub.org/journal/sea
https://doi.org/10.12677/sea.2018.73020

Hans汉斯

A Method of Hierarchical Reconfiguration of Flow Chart Reversing from C Source Code

Fujun Ji, Zitao Wang

Information School, Capital University of Economics and Business, CUEB, Beijing
Email: jfj@cueb.edu.cn

Received: Jun. 7th, 2018; accepted: Jun. 21st, 2018; published: Jun. 29th, 2018

Abstract

Source-code understanding plays a vital role in white box testing and in correcting students' source code. However, it is very inefficient to manually build a corresponding process in the brain by testing a line of code that may be very different in style. In this paper, a method of hierarchical reconfiguration of flow graph reversing from C source code is proposed, which realizes the visualization of source code, reduces the complexity of source code and reduces the size of source code by stratified abstractness. According to the classification of source code, the definition of each figure of module is given, and the source is analyzed based on the thought of stratification. The automatic transformation program of C source code to multi-layer flow chart is developed. Finally, the automatic generation process from C source code to multi-layer program flow chart is practiced through specific instance operation. The program flow chart of the reconfiguration of the source code can show the business logic and process of the program well, and realize the visualization of the source code. This provides the convenience for the white box testers and the teachers of the C language who must correct programming assignments.

Keywords

Method of Hierarchical Reconfiguration, Flow Chart, Source Code, Understanding

一种C源代码逆向流程图分层重构的方法

冀付军，王子涛

首都经济贸易大学信息学院，北京
Email: jfj@cueb.edu.cn

收稿日期：2018年6月7日；录用日期：2018年6月21日；发布日期：2018年6月29日

文章引用: 冀付军, 王子涛. 一种 C 源代码逆向流程图分层重构的方法[J]. 软件工程与应用, 2018, 7(3): 168-176.
DOI: 10.12677/sea.2018.73020

摘　要

源程序理解在白盒测试以及高校教师批改学生源程序作业代码时发挥着至关重要的作用。然而，人工通过查看一行行风格可能迥异的代码，自主在脑中构筑相应流程再进行测试或评判，是非常低效的。本文提出了一种C源代码逆向流程图分层重构的方法，实现了源代码的可视化，利用分层的抽象性逐步降低源代码的复杂性和缩小源代码的规模，依据源代码结构分类给出了各模块图形的定义，并基于分层法的思想逆向分析源代码，开发了C源代码到多层流程图的自动转化程序，最后通过具体的实例操作，实践了从C源代码到多层程序流程图的自动生成过程。这种源代码逆向分层重构的程序流程图，能很好的显示出来程序的业务逻辑和流程，实现了源代码的可视化，这为白盒测试人员以及源程序作业批改教师等人员，进行高效的程序理解提供了方便。

关键词

分层法，流程图，源代码，程序理解

1. 引言

在实际教学过程中，学生在程序编程的题目中所呈现出来的格式，代码风格迥异。教师在批阅学生的答题情况时，在自己的头脑中已经展现出来了大致的程序流程，往往在批阅过程中从不自主的按照自己的惯性思维作为评判标准，效率低下，并且可能会出现差错。在较大项目的前期进行白盒测试代码走查过程中，虽然现在有各种技术来辅助测试人员，但是还是需要人工进行大量的阅读代码，而人工阅读代码并且能够分析出它的程序结构是非常麻烦的，而且费时费力，导致工作效率低下。程序流程图通过简单的平面二维平面图形和一些相应的文字构成，简单明了的为技术人员展现出算法的步骤和程序的结构，清晰的呈现出复杂的嵌套结构，远比测试人员去阅读源代码方便。

分层法研究的主要是面向过程的开发，通过读取源代码直接输出程序流程图，而不需要通过编译源代码来实现。通过对复杂的嵌套的提取实现主程序图的简单明了，实现了对程序结构的充分细致的了解。

关于源代码与程序流程图之间相互转换已有很多研究[1]-[7]，大部分研究集中在根据流程图自动生成代码，如文献[2]等实现了由程序流程图自动生成源代码。而由源代码逆向生成流程图的研究却比较少，且存在一定的局限性。如朱云[7]等人实现了基于图文法的程序流程图与源代码之间的自动转换，其生成过程是对一维字符文法的二维扩展，同时有一组产生规则以及基于产生式规则的推导和规约操作，在该方法中从源程序到程序流程图的转化需要对源代码进行编译。又如文献[8]的逆向输出必须通过该系统产生的源代码才能生成程序流程图，该方法生成程序流程图的局限性比较大。另外，code2flow 是目前公认比较好的一款自动生成程序流程图的软件(http://www.code2flow.com/)，但它利用源代码生成的流程图，结构上不能很好的显示程序的意义，尤其在程序流程图分层抽象方面，无法提供程序作业阅卷者所需的关键信息，这在一定程度上辅助人类对于复杂大规模程序的理解，显得力不从心。

本文尝试提出一种基于分层法[9]的 C 语言源代码转化成程序流程图的方法，无需对源代码进行编译，无需使用特定源代码和伪代码，并能支持逆向流程图的抽象分层，以方便人类理解。同时将研究分析分

层法的分层依据以及相关的参数，为后续的研究提供基础；最终提出并实现基于分层法的 C 语言源代码转化成程序流程图的图形生成的基本方法；然后利用实例展示从源代码到程序流程图转化的过程；最后对全文进行总结和展望。

2. 模块流程设计图的提出

所谓层次分析法，是指将一个复杂的多目标决策问题作为一个系统，将目标分解为多个目标或准则，进而分解为多指标(或准则、约束)的若干层次，通过定性指标模糊量化方法算出。

本文提出的分层法是在结构复杂的程序中，定义部分复杂的结构进行简单化表示，从而形成一个较为简单的主体结构，可以看出程序的大体流程图。然后再进行对复杂的结构进行单独输出，这样可以较为清晰看到复杂结构里面的结构，从而可以较快的判断出是否程序是否正确。

如今图形界面的可视化发展迅速，分层法作为形式化的方法，可以清楚呈现出源程序的结构体系，图形化的表示避免的纯文字表述带来的歧义。下面是后文中用到的分层法中的一些定义。

传统的流程图(FC, N-S, PAD)等都比较容易理解，但是之间的关系不是很严格，绘制后需要大量的修改，我们基于 C 语言的特点，把几个复杂的结构分为几个单元块分别表示：If-else if 模块、for 模块、switch 模块和递归模块。

在 C 语言中，结构类型有顺序结构、循环结构、选择结构。下图将复杂的结构在主程序图中简单化进行表示，这里自定义了四个复杂的结构，分别是 If-else if 模块、for 模块、switch 模块和递归模块。在一般的流程图中，往往将递归内分解为具体的 If-else if 模块、for 模块、switch 模块，但是由于递归次数可能会比较大，而且递归往往完成一个相对明确的功能，与循环类似却又不同，因此将递归专门定义为一个模块。借鉴现有流程图经验，这四个模块的具体图形化流程提出如图 1。

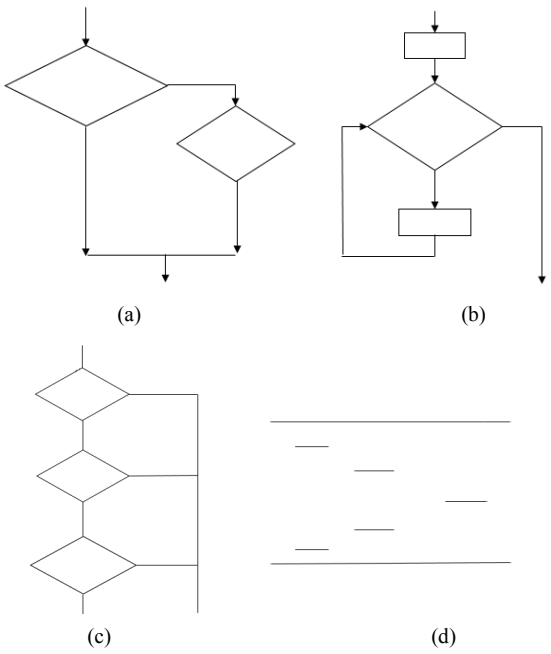

Figure 1. Design of flow chart for complicated module. (a) Flow chart of if-else if; (b) Flow chart of For; (c) Flow Chart of Switch; (d) Flow chart of recursion

图 1. 复杂模块流程设计图。(a) If-else if 流程图；(b) For 流程图；(c) Switch 流程图；(d) 递归流程图

本研究所提的这四个模块的流程设计图，即 If-else if 模块流程设计图、for 模块流程设计图、switch 模块流程设计图和递归模块流程设计图，具有全覆盖、可伸缩分层的特点，满足了程序读者或理解者的层次化抽象理解的需要。全覆盖特点是指，本研究所提四种模块流程设计图配合国家标准学会规定的流程图符号，可以保证所有的结构化 C 源代码都可以覆盖到，即都可以图形化。可伸缩分层是指每个模块的流程设计图都是可以逐层细化呈现的，这使得源代码可视化既可以实现从大的方面理解也可以从细节方面查看，大大方便了不同尺度宏观、微观理解能力的人群。

3. 图形的生成

当程序源代码含有 If-else if 模块、for 模块、switch 模块和递归模块时，其可视化将采用本研究所提出的流程设计图，其他情况将采用国国家标准学会(American National Standards Institute, ANSI)所规定一些常用的流程图符号。

3.1. 文件的读取

文件的读取采用按行读取的方式，如果遇到带有{}的模块，将使用正则表达式进行匹配。

```
public class Read{
    public static void Read(){
File file = new File("g:\\d.txt");//Text 文件
            BufferedReaderbr = null;
            br = new BufferedReader(new FileReader(file));
            String s = null;
            while((s = br.readLine())!=null){
String r = s.replaceAll("^.*\\[", "").replaceAll("].*", "");//截取{}之间的内容
…}
…}
```

这里采用对{}的配对来读取大括号({})的关键字，然后根据匹配语句中的关键字来确定所要输出的图形和相应的语句表示。

3.2. 图形的生成

本研究团队采用的是 Drawing Panel 进行图形的输出。主要输出的图形有起止框、菱形、平行四边形、直线、if-else if 图形、for 图形、switch 图形、递归图形等。给与图形输出的起始点，然后各个图形将会根据所输出时坐标点的位置进行输出。

```
package tiqu;
import java.awt.Graphics;
public class a11 {
    Juxing(Graphics g,intx,inty,intlength,int width );//矩形
    Yuanjiaojuxing(Graphics g,intx,inty,intlength,intwidth,int h);//开始
    Yuanjiaojuxing1(Graphics g,intx,inty,intlength,intwidth,int h);//结束
    lingxing(Graphics g,intx,inty,intlength,intwidth,int h);//画一个菱形
    jiaotouzhixian(Graphics g,intx,inty,intlength,int width ,int h);//画一个带箭头直线的方法//在多边形方面
```

```
        jiaotouzhixian2(Graphics g,intx,inty,intlength,int width ,int h);//一个是开始结束框的直线
        jiaotouzhixian1(Graphics g,intx,inty,intlength,int width ,int h);//画一个带箭头的直线//在具体函数方面
        guaiwanzhixianjiantou(Graphics g);//画一个拐弯直线箭头
        pingxingsibianxing(Graphics g ,int x, int y, int length, int width );//画一个平行四边形
        digui(Graphics g ,int x, int y, int length, int width );//递归模块
        mokuai_for(Graphics g ,int x, int y, int length, int width ); //for 模块
        mokuai_if_else(Graphics g ,int x, int y, int length, int width );//if_else 模块
        mokuai_switch(Graphics g ,int x, int y, int length, int width );//switch 模块
    }
```

分层流程图的关键技术思路为，在进行具体层次抽象输出时，首先判断是否涉及递归、循环、判断和分支，如果是的话，将同时生成两层图形，上层图形为相应的模块图，下层图形则为该模块内部代码的程序流程图，而在生成下层图形时，会再次判断是否涉及递归、循环、判断和分支，如果是的话，将再次生成两层图形，如此循环，直到底层不存在递归、循环、判断和分支为止。这样，即可实现程序流程图的多级伸缩，它表示了一定的抽象性。

4. 实例分析

下面，本课题组将以 C 语言递归解决分鱼问题作为实例来分析分层流程图的自动生成。

```
1. #include <time.h>
2.int main()
3. {
4. Int d1,d2,c1,c2,i,j;
5. c1 = c2 = 0;
6. rand();
7. for(i=1; i<=100; i++)
8. {
9. d1 = d2 = 0;
10. for(j=1; j<=6; j++)
11. {
12. d1 = d1+rand()+1;
13. d2 = d2+rand()+1;
14. }
15. if(d1>d2)
16. c1++;
17. else if(d1<d2)
18. c2++;
19.
20. if(c1>c2)
21.printf("\nThis first win.\n");
22. else
```

```
23. if (c1<c2)
24. printf("\nThis second win.\n");
25. else
26. printf("This tie.\n");
27. return 0;
28. }
29. }
```

读取到第 7 行时，发现关键字 for，然后采用正则表达式的方法进行花括号({})的匹配，当匹配到 28 行时结束，将其作为一个 for 循环模块在主流程图中输出图 2 中的 for 图(b)。然后再从第 7 行开始输出 for 图中的 for 循环和 if-else if 判断图形，直至第 28 行结束。具体 for 循环之内流程图如图 3 所示。

通过 code2flow 来画出同样代码的流程图，结果如图 4 所示，它无法展示程序作业阅卷者所需关键信息，且其流程图主要以线性为主，在不同结构模块分类呈现方面，不如本流程图更易于人类理解。通过实例应用可以看出，本文在没有编译源代码的情况下，使用了非自身特定的通用 C 源代码，该代码系真实源代码(非伪代码)，实现了程序流程图易于人类理解的分层结构化展示，这是前面调研中所提转换流程图的三种方法都做不到的，由此可见该方法存在较大创新性，同时本实例也表明，本研究所开发系统在一定程度上是成功的。

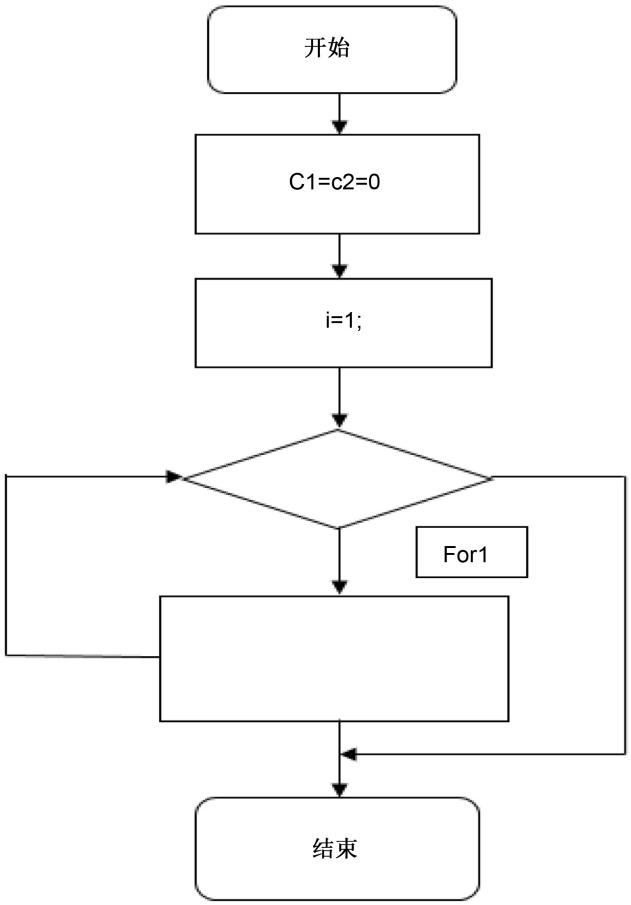

Figure 2. Entire flow chart of the program
图 2. 程序整体流程图

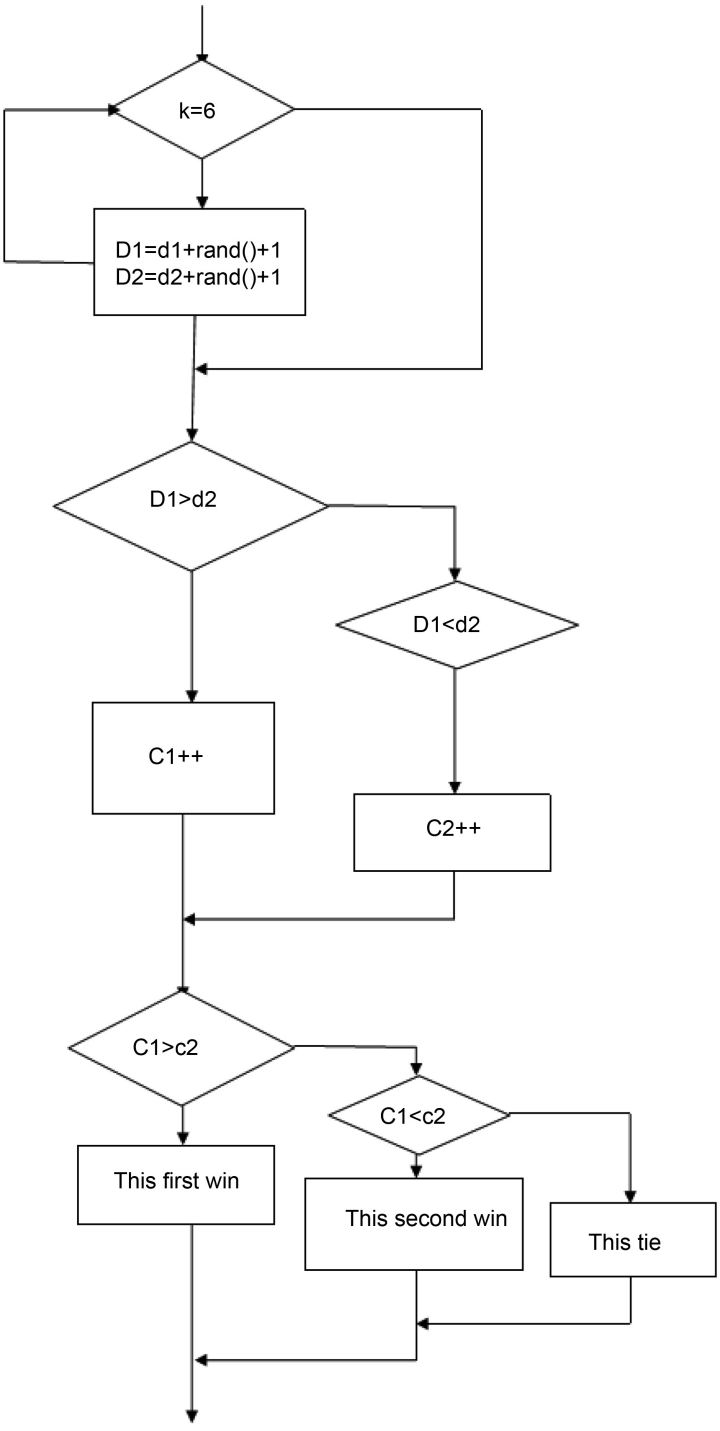

Figure 3. Flow chart of For
图 3. For 循环图

5. 总结及展望

基于分层法的 C 语言源代码转化成程序流程图可以清晰的展现出结构复杂程序的逻辑结构。对于一级图形可以看出程序的大体的结构，二级或者三级图形可以详细看出各个单元块的结构。使得教师在教学过程中可以看出学生思路是否清晰，是否很好理解题目；也为前期白盒测试人员来走查程序、检

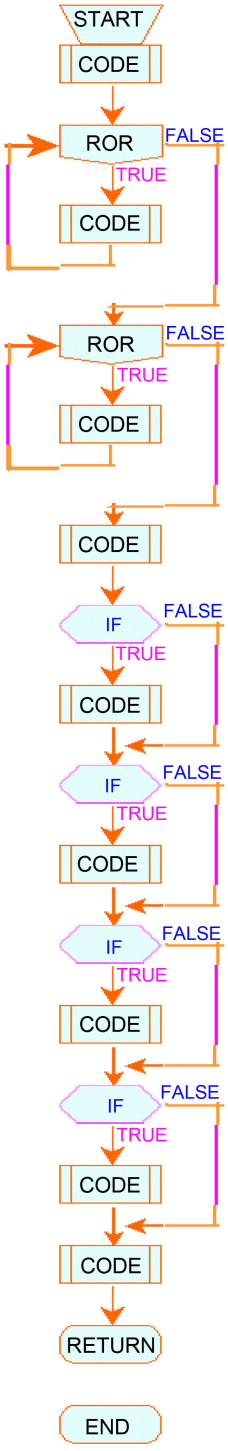

Figure 4. Corresponding flow chart in Code2 flow
图 4. 同样代码对应的 Code2 flow 中的流程图

查系统是否严格遵守软件概要设计和详细设计提供了方便。

本文的主要贡献是：

1) 定义了递归在程序流程图中的表示；

2) 采用了分层的方法使得程序流程图更加清晰。

本文方法对于多种形式的嵌套图形表示，还不够完善，进一步研究可以深入探讨多种形式嵌套在一起如何进行图形化；对于大型项目，输出点较多时，需要研究图形分裂问题的解决途径。

致　谢

感谢清华大学闻立杰副教授和本刊审阅人提供了非常有价值的论文修改意见和调研相关资料。

参考文献

[1] Yue, Y.X. (2012) Developing a Comprehensive Teaching Evaluation System for Foundation Courses with Enhanced Validity and Reliability. *Educational Technology Research and Development*, **605**, 55-60.

[2] Wang, L.-M., Wang, G.-N., Zhou, M.-Y., *et al.* (2012) Research and Implementation of Algorithm from Program Flowchart to Code. *Journal of Xidian University*, **39**, 80-92.

[3] Martin, C.C., *et al.* (2004) RAPTOR: Introducing Programming to Non-Majors with Flowcharts. *Journal of Computing Science in Colleges*, **19**, 52-60.

[4] Kains, C. and Somkiat, W. (2006) Visual Programming Using Flowchart. Communication Technologies, Bangkok, ISCIT'06, 1062-1065.

[5] Drazen, L. and Ivan, F. (2011) A Visual Programming Language for Drawing and Executing Flowcharts. *Proceedings of the* 34*th International Convention*, Opatija, 23-27 May 2011, 1679-1684.

[6] 严代彪, 王树宗. 一种源程序到程序流程图的自动生成算法[J]. 微计算机信息, 2003(7): 83-65.

[7] 朱云, 曾晓勤, 朱宁, 刘禹锋. 基于图文法的程序流程图与源代码自动转换[J]. 计算机工程与科学, 2015, 37(5): 937-945.

[8] 许秀林, 李蕴华. 基于图元装接模式由程序流程图自动生成源代码[J]. 软件工程, 2016, 19(11): 4-10.

[9] 许晓春, 杜晓晨, 梅琳, 徐永森. 一种改进的程序流程图——层次流程图 HFG [J]. 南京大学学报(自然科学版), 2002, 38(2): 158-165.