北京郵電大學

实验报告

实验名称: 二进制转八进制

班级 : 2019211309 姓名 : 陈悦 学号 : 2019211413 分工 : 文档

班级 : 2019211309 姓名 : 马晓亮 学号 : 2019211400 分工 : 代码

2020 年 *10* 月 *18* 日

一　需求分析

本程序要完成的功能为，将一段二进制序列转化为八进制的序列。输入要求 01 序列，并且要以 # 结束。例如，101010011010#。如果有必要，可以读入多个二进制串。100110#1010101#。我们对输入做出以下限制:

1. 输入的串的长度不能超过 10^6
2. 输入的所有字符只能有 0,1,#

在正常情况下，本程序输出一段 8 进制串。在有多个二进制串输入的情况下，八进制串将按顺序输出。以上面两个输入样例为例。

5232

46 125

二　概要设计

main 创建并初始化 Solution 结构，并且循环调用 Solution 中的读入和输出。

Solution 中保存了两个栈,一个用于放置二进制串,一个用于放置八进制串。栈中实现了 push,pop,get_top 等栈的基本功能。Solution 结构可以调用 readBinary 进行读入，调用 writeOctal 写入。再在 Solution 中实现一个可以将二进制数转化为八进制的中间函数。

三　详细设计

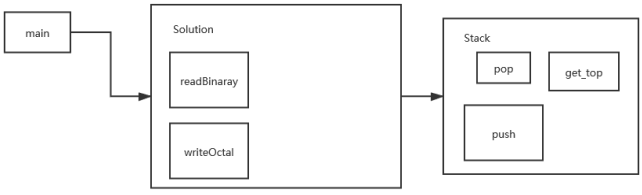

图 1: 函数调用关系图

Algorithm 1 Solution 结构定义

Input: 输入二进制串
Output: 输出八进制串

```
读入二进制串
function readBinary(void)
    读入字符 a
    while a 不是 # 也不是 EOF do
        if a 是错误字符 then
            输出错误
        end if
        将 a 加入到栈 binary 中
    end while
end function

输出八进制串
function writeOctal(void)
    while binary 栈中不为空 do
        temp = 在栈中取三个 bit 转化为八进制数
        将 temp 加入到 Octal 中
    end while
end function
```

四　调试分析报告

使用栈数据结构完成这个任务较为平凡。值得一提的是我们完成了对栈的封装，使得他可以在其他实验中复用。

在使用线性数据结构的操作下，算法复杂度几乎已经达到了下界，即 $O(N)$。但是空间复杂度上面还有许多可以优化的地方。我们人为定义的数据上限为 10^6。但是显然，在大多数情况下用户使用的空间会远远低于这个上界。所以一次申请 10^6 显然是过度浪费内存的。

所以我们将栈的数据存储到一个叫做 Vector 的动态数组中，这个数组开始时数据边界为 10^2 级别。在使用过程中，这个数组会自动调用 *resize*() 扩大边界。我们设一个系数 k，每次 *resize*() 后的边界为原来的 k 倍。接下来我们将测试不同 k 下由 *resize* 造成的浪费。

	N	10^4	10^5	10^6
1.6		29634	121688	121688
2.0		32704	65472	65472

2.4	20807	120056	120056

观察发现，在 $k = 1.6$ 与 $k = 2.4$ 时，数据表现较为极端，而 $k = 2.0$ 时较为平稳，所以我们选择 $k = 2.0$ 作为我们的 vector 的 resize 系数。

五　用户使用说明

用户可以使用 IDE 或者手动编译源代码 stack.cpp，获得可执行文件。
笔者使用的 gcc 版本为 8.1.0 运行可执行文件后，用户可以选择文件输入或者交互式输入。在文件输入下输出将会从定向到 output.in。结束程序可以输入 EOF

六　测试结果

用于测试的样例分为两部分，一部分为功能性样例，用于测试程序的正确性。另外一个为极端样例，用 generator.cpp 编译得到的可执行文件生成，用于测试程序的性能。
我们构造的功能测试样例为
101010011010 -> 5232
100110#1010101# -> 46 125
在程序中输入显示结果与手算相符