# 数据结构进制转换实验报告

班级：2015211306 姓名：李航 学号：2015211284 分工：

班级：2015211306 姓名：魏晓 学号：2015211301 分工：

1. **需求分析**

**这道题的目的是利用栈的先进后出的特点实现栈操作，强调的是通过操作栈实现进制转换，输入一串二进制数，并且用#结尾，输出相应的八进制形式，程序的功能是实现进制转换，如果输入中有一个数字大于1，就会报错**

1. **概要设计**

**首先需要定义一个栈，用来实现栈操作，栈结构中必须存在一个指向int的指针，一个栈底指针，主程序首先需要用字符的形式读取二进制数，并且转化为二进制数，并且保存在栈中，然后只要每到栈底就一直转换，并且保存到一个sum中，并且把他最后用八进制打出来**

1. **详细设计**

**struct stak{**

**int \*base;**

**int \*top;**

**int stacksize;//粘当前的存储空间**

**};**

**这就是这个栈结构，用来实现栈操作，栈结构中必须存在一个指向int的指针，一个栈底指针，其中stacksize用来存储栈的大小。**

**while(bbs!='#')**

**{**

**\*s.top=bbs-'0';**

**if(\*s.top>1)**

**{**

**cout<<"error";**

**return 0;**

**}**

**s.top++;**

**scanf("%c",&bbs);//执行完最后一遍后s.top并没有被赋值**

**}**

**这是这次的数据输入部分，首先用字符的形式将数据读取，然后根据阿斯玛值转换为数字，最后把这些数保存到栈中，**

while(s.base!=s.top)

{

sum=sum+(\*s.top)\*num;

num=num\*2;;

free(s.top);

s.top--;

}

这是运算部分用来存储每一项转换产生的值的总和，最后用八进制输出。

1. **调试分析报告**

**因为这个输入端的代码中，循环是在赋值后面，所以在退出循环的时候，最后一个s.top并没有被赋值，指向一个随机数，在转换的时候你就会发现这个数实际上转换为一个根本不熟悉的数，如果你此时在malloc之后没有使用free你会发现他指向一个上一次运算的数，然后就很难发现错误。这个算法的时间复杂度是O(n)的空间复杂度是一个Ω（n）的。改进的地方就是很多函数内置了没有封装成不同的函数包。可改进性比较低**

1. **用户使用说明**

**使用这个代码，不需要使用编译器，直接点击exe文件运行即可，点击输入一个二进制数并且用#结尾，就可以看到转换的八进制数**

1. **测试结果**

**输入：123412341234# 输出：error**

**输入：10101011101# 输出：2535**