2/8进制转换器实验报告

目录

[一：需求分析 2](#_Toc86412534)

[二: 概要设计 2](#_Toc86412535)

[问题解决思路 2](#_Toc86412536)

[主函数伪代码 3](#_Toc86412537)

[stack\_input 类 3](#_Toc86412538)

[stack\_output 类 4](#_Toc86412539)

[函数调用关系 4](#_Toc86412540)

[输入检测 5](#_Toc86412541)

[三：调试分析报告 5](#_Toc86412542)

[四：设计实验的回顾讨论 6](#_Toc86412543)

[五：用户使用说明 6](#_Toc86412544)

[六：测试结果 7](#_Toc86412545)

[七：实验结果 7](#_Toc86412546)

题目：2/8进制转换器

班级：2020211306 姓名：马天成 学号：2020211376 分工：撰写代码+实验报告

班级：2020211306 姓名：黄洪建 学号：2020211371 分工：撰写代码

班级：2020211306 姓名：马紫薇 学号：2020211392 分工：撰写实验报告

# 一：需求分析

题目的目的为：

从终端输入一串0/1表示二进制数，以“#”结束，输出它的8进制表示形式。

此程序输入输出的形式是：输入一个任意的非负的二进制整数，输出与其相等的八进制。

根据题意，我们知道：

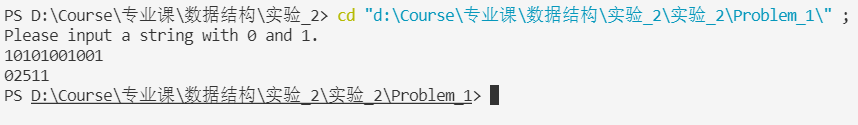
1. 输入的应该是无符号数，所以不需要考虑补码负数首位为0的情况。

2. 而且，不能出现小数，所以会进行相应检测。

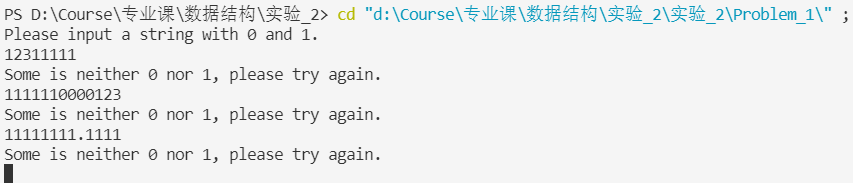
3. 此外，输入数据要清除前导0.

4. 不能只有一个回车。

输入输出的正确形式是：



输入输出的错误形式是：



# 二: 概要设计

## 问题解决思路

1. 在运行过程中各位是按照从低到高这个顺序产生的，但输出时却要由高到低位输出，此 顺序恰好与栈的进出顺序一致，此时将二进制的数按顺序入栈，算出数位，再按照顺序， 计算到3的倍数时出栈，得到的便是8进制的数。

2. 检测输入，不能执行非法输入。

3. 最后要在8进制数输出前加上0表示为8进制数。

## 主函数伪代码

int main()

{

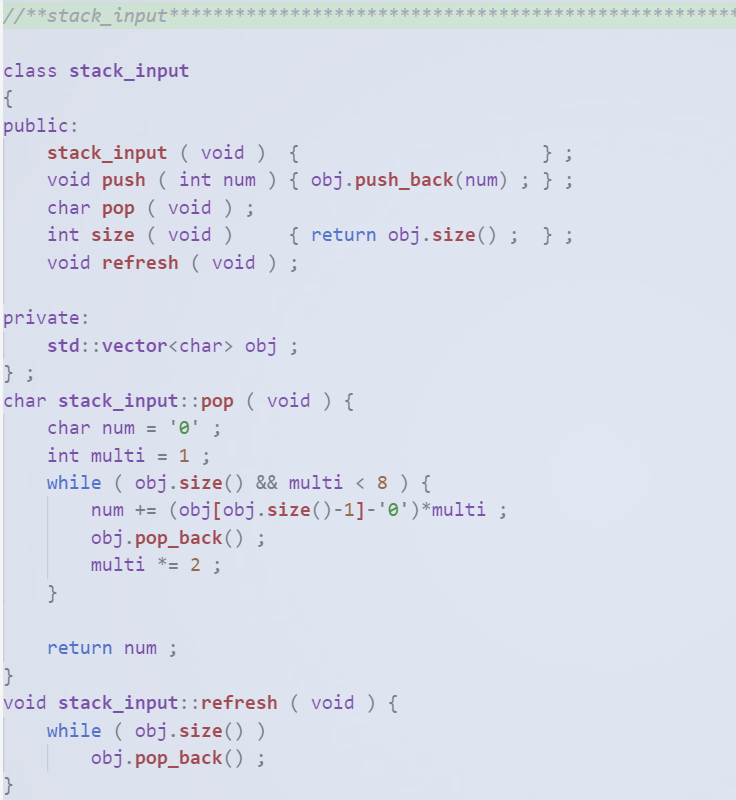
清除前导0 ;

stack录入数据;

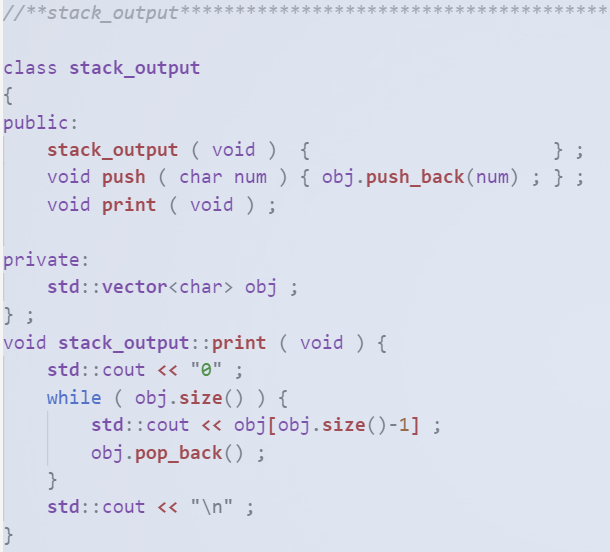
stack转换并输出8进制; // 同时进行，节省时间。

}

## stack\_input 类



## stack\_output 类



## 函数调用关系

Main

Stack\_Char::pop()

Stack\_Char::push()

Stack\_Int::pop()

Stack\_Int::push()

存01串 输出01串到字符栈 输入整型串的输出 转换并输出答案

因为我们通常进制转换时是要考虑到大于十进制的字符型的，所以我用字符栈类来存答案和输出答案，以便修改为不同的进制时只需要修改输出分支而不需要修改存储类型。

## 输入检测



因为本实验比较简单，所以我在输入检测时加了一些料：

1. 当检测出任何非01的字符时，清空输入并要求重新输入。

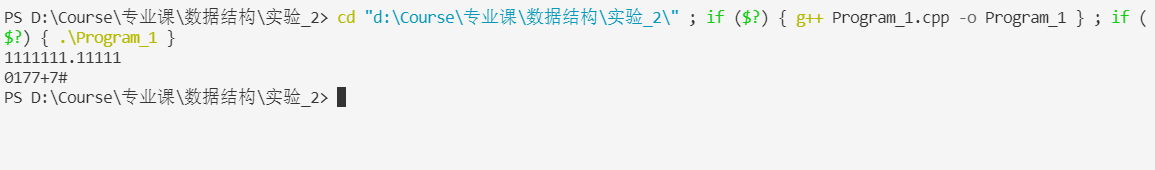
2. 输入会自动清除前导0。

3. 中文编码不可检测，因为中文编码至少2个字节，(UTF-8 2bytes)。不可用于回文检测。

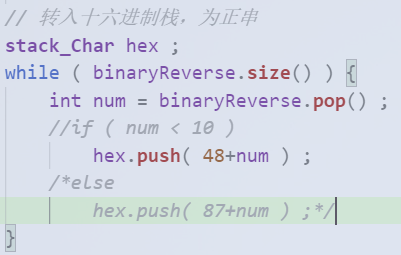
# 三：调试分析报告

1. 输入数据格式处理：在上述输入格式时已经阐述。

2. 在测试数据字符串出现小数点时，会出现乱码，是开始程序并没有进行输入检测所导致。



4. 审题不仔细，发现最初代码输出的是十六进制数。



5. 在处理每三个数的时候，一开始乘法计算算反了，导致高位的乘数比低位小，答案错误。

即在每一次计算进制转换时，最高位乘1，第二位乘2，最低为乘4，显然不对。

# 四：设计实验的回顾讨论

1. 输入数据格式处理。

2. 写class时栈初始化时不对导致不能使用。

3. 在测试数据字符串出现小数点时，会出现乱码，是开始程序并没有进行输入检测所导致。

4. 审题不仔细，发现最初代码输出的是十六进制数。

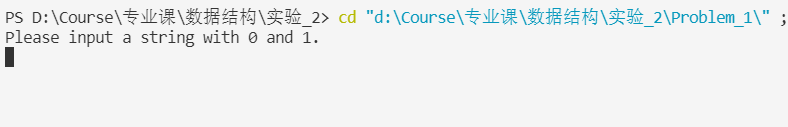
5. 在处理每三个数的时候，一开始乘法计算算反了，导致高位的乘数比低位小，答案错误。

6. 该程序的时间复杂度为O(n)。

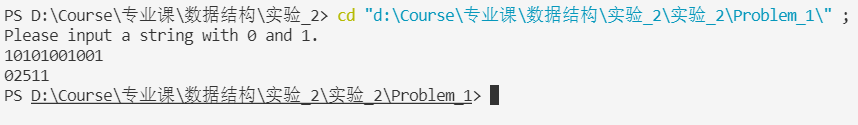
7. 该程序的空间复杂度为O(n)。

# 五：用户使用说明

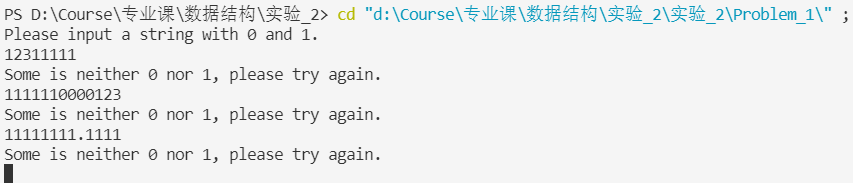
1. 打开程序，首先会出现如下界面：



2. 用户输入合法的01串后将会继续显示：

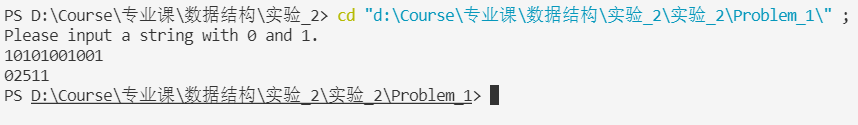


3. 当输入不合法时，会出现：

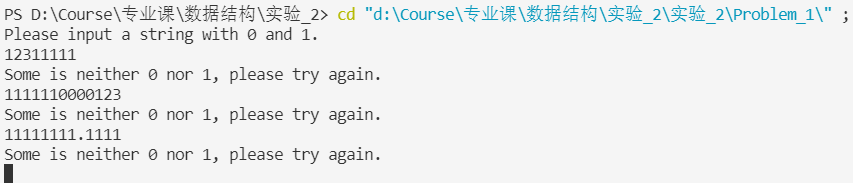
用户可根据提示，重新输入相应数值并回车即可。

# 六：测试结果

1. 正确输入时，输出正确答案：



2. 有错误输入会报错并要求重新输入：



用户可根据提示，重新输入相应数值并回车即可。

# 七：实验结果

**实验结论：**

1. 进制计算需要考虑字符型，用char来存因为超过十进制要用字符表示。
2. 输入要用字符串，这样灵活性会更高。

**实验效果：**

通过这次实验，我们将理论和实践相结合解决了实际问题，学会了将栈灵活使用，利用它来实现数据存储与转换，巧妙地不占用很多算力就解决了反序问题，在后续的小组讨论中，通过彼此思路的交换，拓展了彼此的视野，提供了从不同角度形成的代码，讨论彼此的利弊，找出了比较共性的错误，使程序得到了进一步的优化，还了解到了审题的重要性 ,这对我们以后在实验的审题仔细程度上敲响了警钟，应严格遵守题目要求，通过解决以上问题，本组同学的能力得到了进一步的提高。