# 1 题目简介

A题:

从键盘输入两个命题变元P和Q的真值，求它们的合取、析取、条件和双向条件的真值。

1. C题：

求任意一个命题公式的真值表（B），并根据真值表求主范式（C）。

# 2 解题思路即算法分析

A题：

事实上，合取运算对应于程序中的逻辑与运算：P&&Q ；析取对应于程序的逻辑或运算 P||Q ；条件运算对应于 (!p) || q ；等值运算对应于 ((!p) || q) && ((!q) || p) 。

因此，根据用户输入的P、Q值直接通过程序进行上述运算即可得到结果。

此外，需要注意的一点是：用户在输入P、Q值时，可能意外输入了非数字字符或者0、1之外的无效字符，这时程序会提示用户输入非法并重新输入，以增强程序健壮性与稳定性。

B、C题：

本程序中，用栈来模拟表达式的运算顺序并利用循环结构模拟真值表，对变元依次赋值，进行运算，并根据真假结果添加合取项、析取项到最终结果，从而得到最终的主析取范式、主合取范式。

利用栈对运算进行模拟是本程序解题的核心，具体来看：分为数字栈、符号栈。其执行逻辑如下：从左往右顺序遍历表达式，如遇到0、1数字，则进数字栈，若遇到符号，则比较当前符号和符号栈栈顶符号的优先级，若符号栈栈顶优先级低，则当前符号入栈，否则先执行符号栈顶运算符对应的运算。此外，如果遇到的符号是左括号，则设定左括号的优先级最高，左括号直接进栈，如果遇到有括号，则栈定左括号出栈。这样一来，随着对表达式从左到有的遍历，就利用数字栈与符号栈完成了对中缀式运算顺序的模拟。

# 3 数据结构

栈Stack类:



## 接下来对类函数进行说明：

## Push函数：功能：将值为item的节点入栈

事实上，这个函数功能的实现只有一句话：top = new StackNode<T>(item, top)；

按照头结点插入法，新插入节点永远是在已有节点之前，并将top指针在插入之后指向新节点，从而做到后进先出。

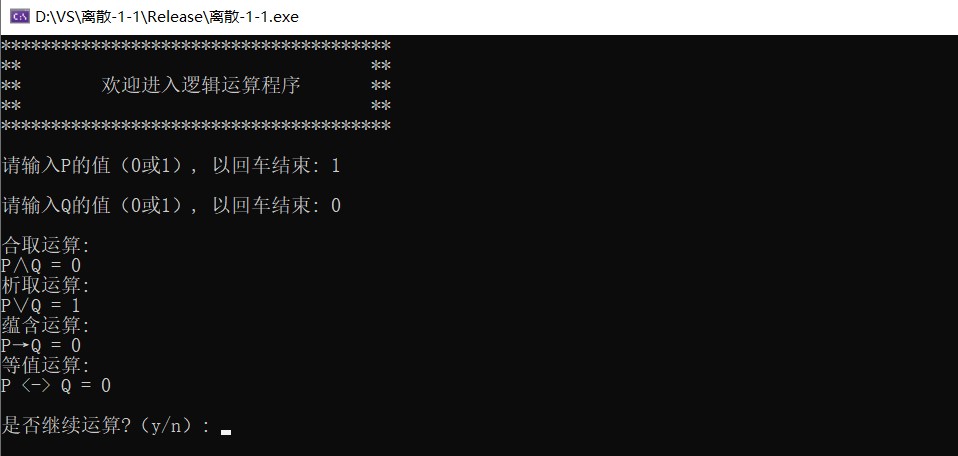
1. Pop函数：功能：将栈顶元素弹出栈

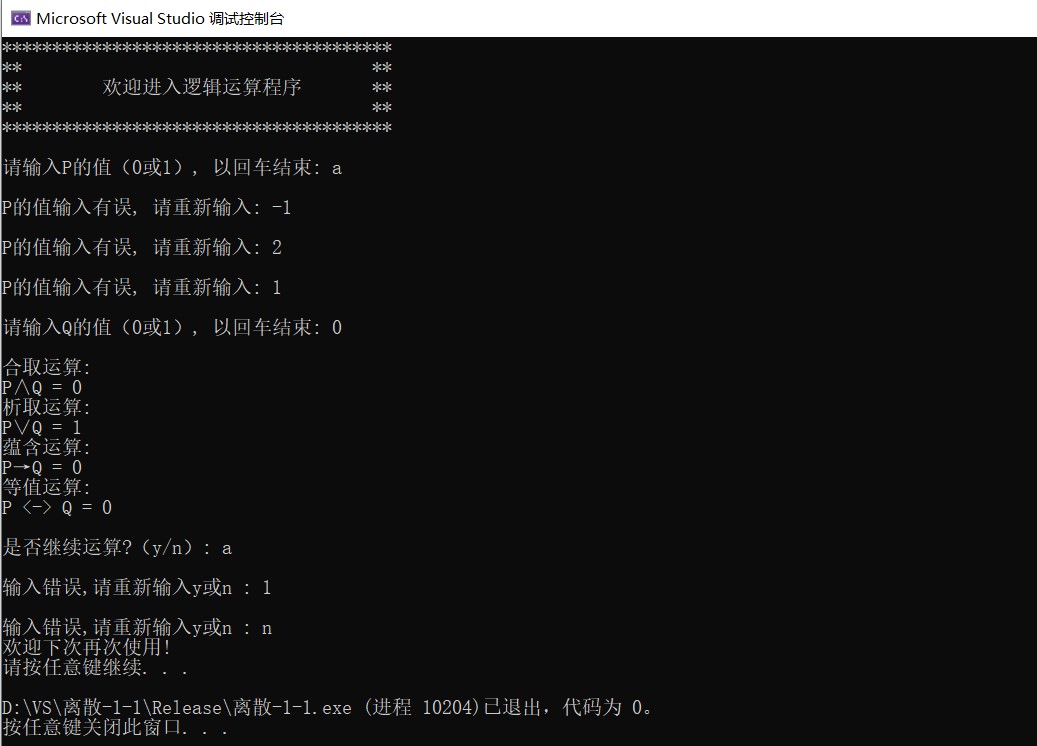
在弹出元素之前先判断栈是否为空。正常来说，本程序中表达式的计算不会导致栈空（在栈里额外添加了结束标记符的前提下），所以，若表达式导致栈空，那么说明存在表达式不规范的问题此时利用exit函数结束程序。若栈不空，则先暂存栈顶元素的值，之后删除栈顶元素，把栈顶指针指向新的栈顶节点，并返回原来栈顶元素的值即可。

**4.实验结果**

A题：

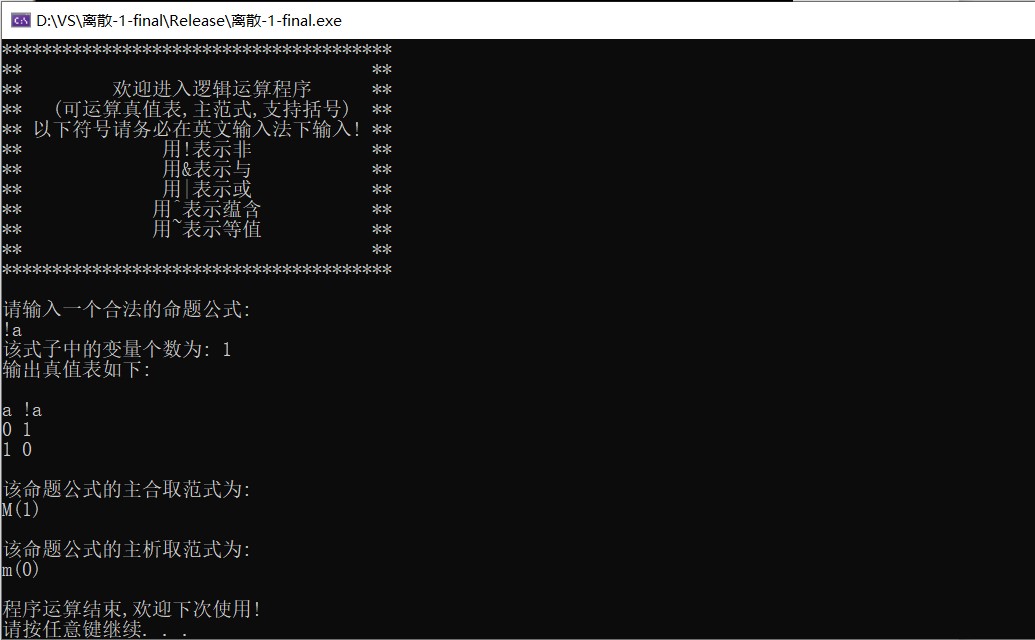
正确运算结果：

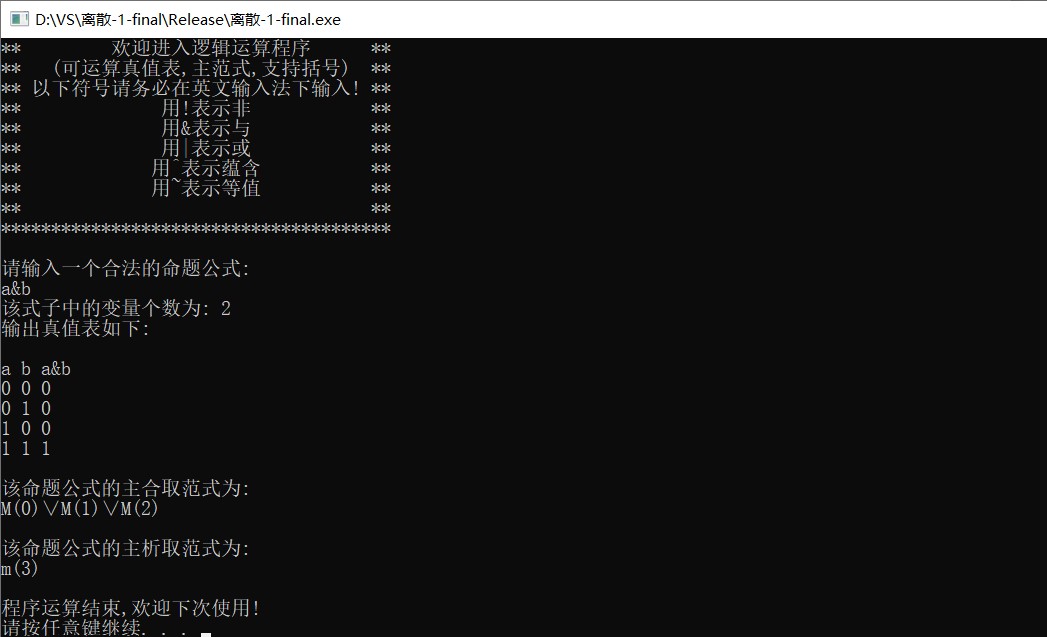


错误输入处理：

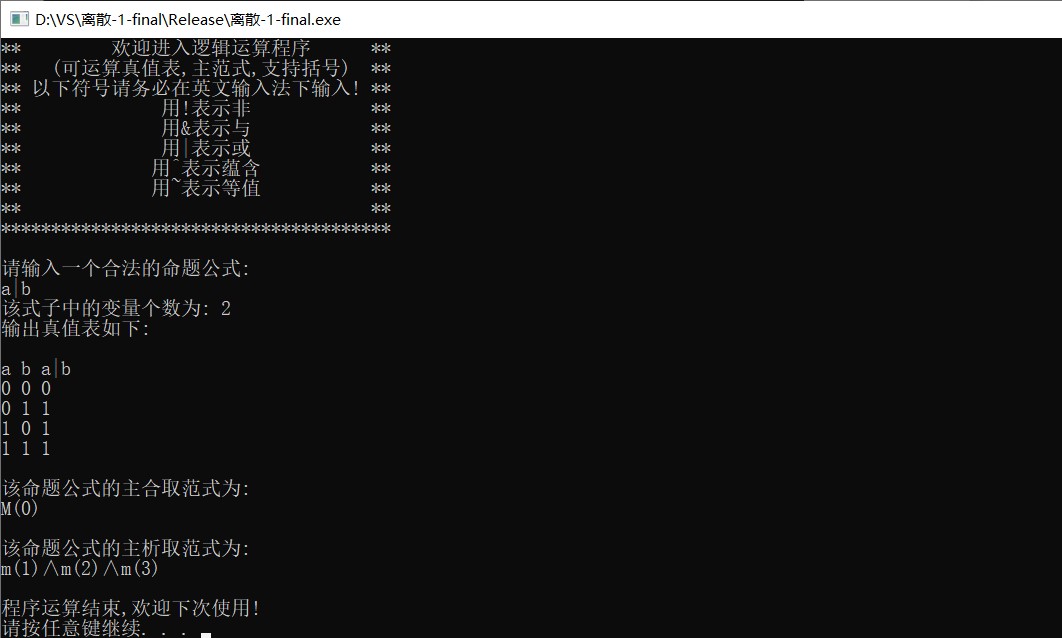
B、C题：

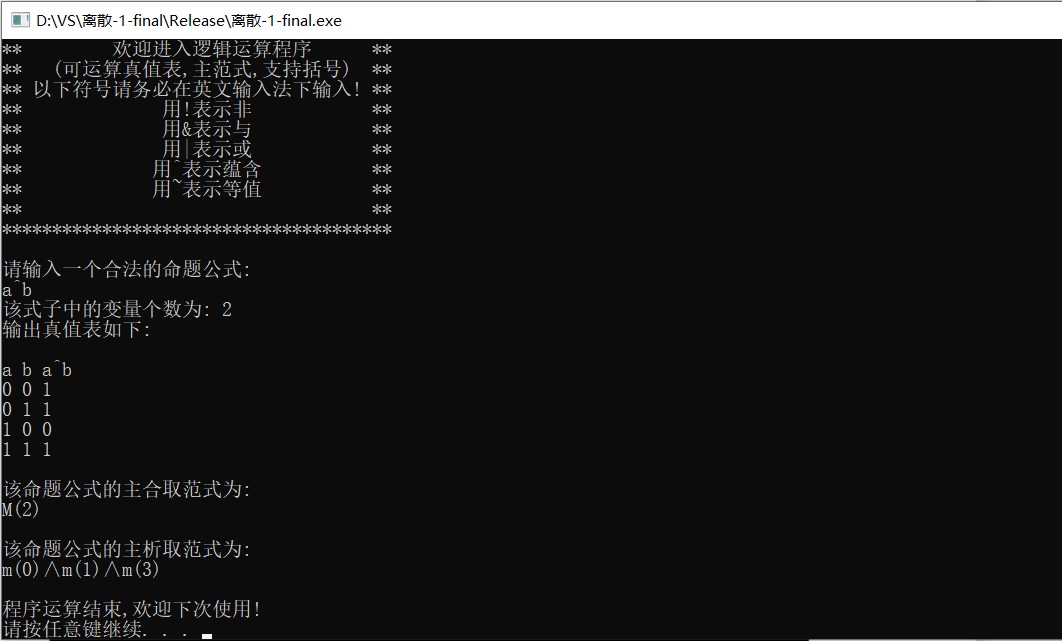
输出非运算公式的真值表，主合取范式和主析取范式：



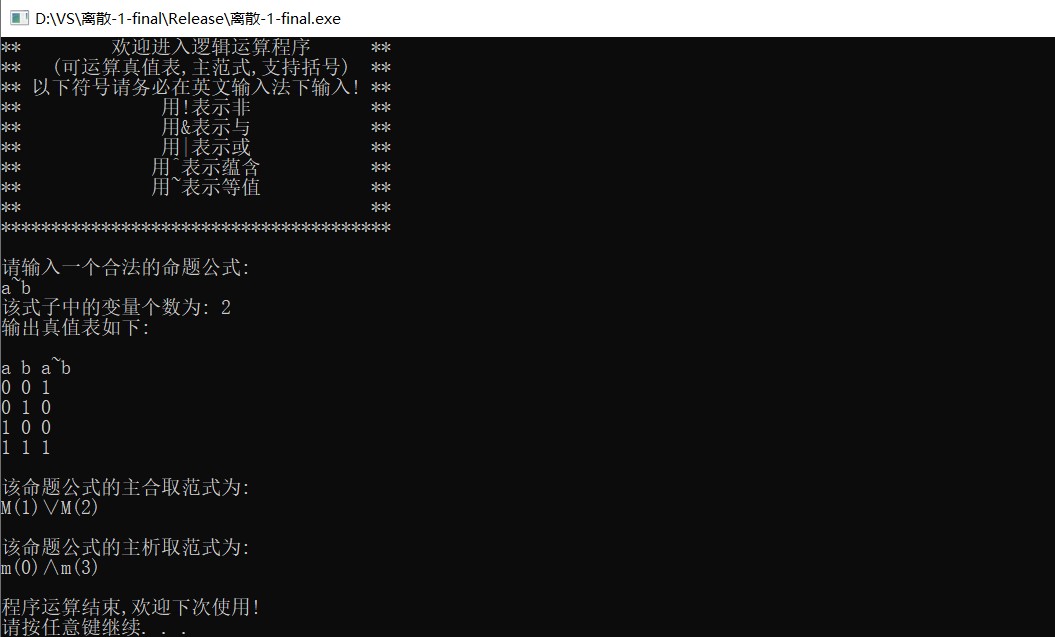
输出与运算公式的真值表，主合取范式和主析取范式：

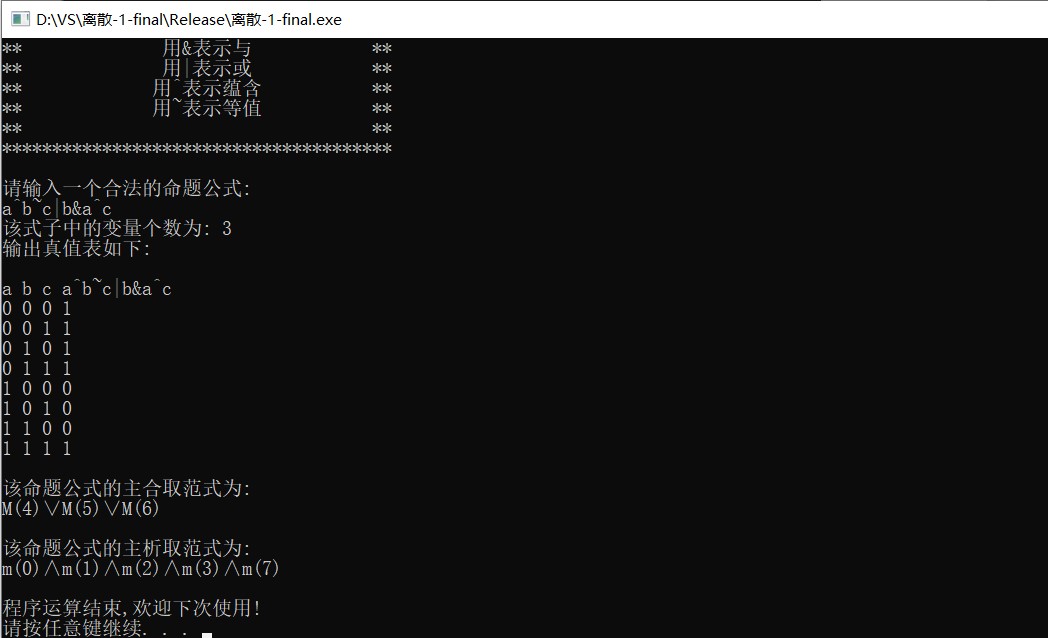
输出或运算公式的真值表，主合取范式和主析取范式：



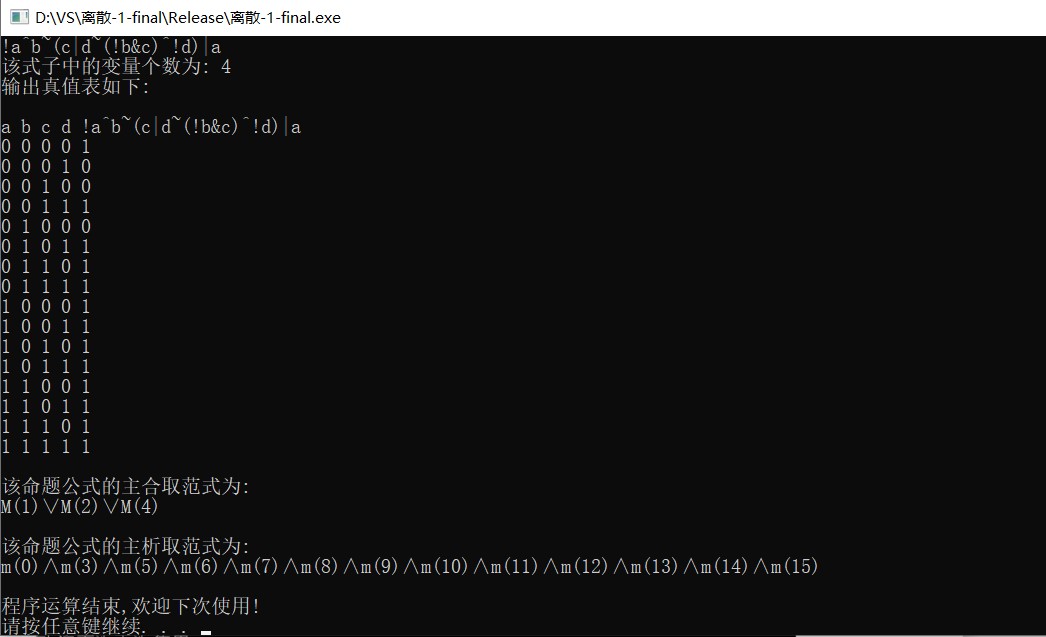
输出蕴含运算公式的真值表，主合取范式和主析取范式：

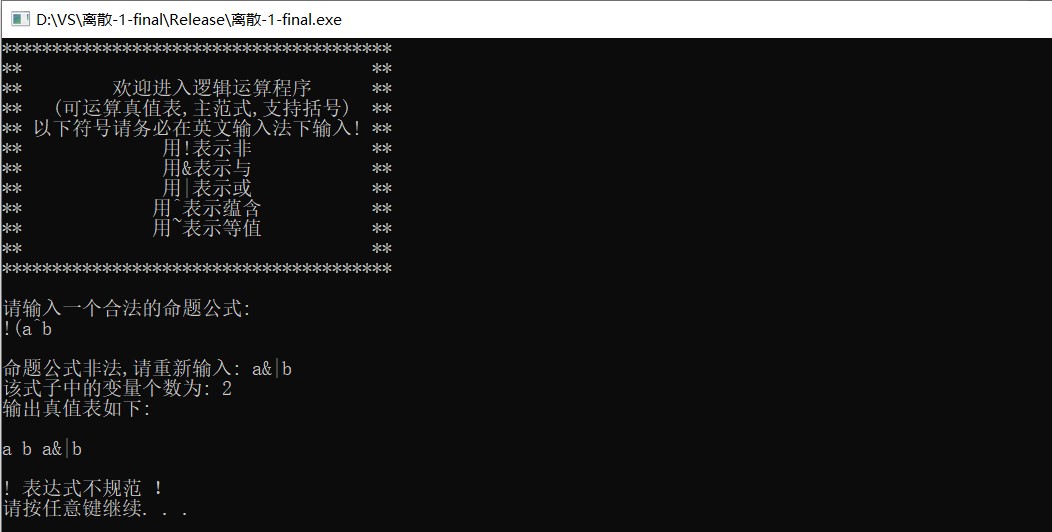
输出等值运算公式的真值表，主合取范式和主析取范式：



****输出综合运算公式的真值表，主合取范式和主析取范式：

输出带括号的综合运算运算公式的真值表，主合取范式和主析取范式：

****

对不规范的运算式的两种处理方式：