《离散数学》课程实验报告1 真值表、主范式

**1.项目要求**

从键盘输入任意一个命题公式，求其真值表，并根据真值表求主范式。

**2.数据结构**

**2.1.映射**

本项目的实现中采用了map库用以建立整型数和字符之间的映射.比如Map\_ci是代表map<char, int>, Map\_ci代表map<int, char>, Map\_ii代表map<int, int>。

**2.2.栈**

本项目使用了stack库用以在逻辑运算中建立运算符栈stack<char> opter和运算数栈stack<int> pvalue。

**3.实现思路及重要函数**

在主函数开始时先定义好运算符优先级（priority）。首先从键盘中得到命题公式字符串。通过getProposition函数得到输入公式中的命题变项，具体实现方式是遍历命题公式中的每一个字符，若该字符是字母则通过findProposition函数确定该命题变项是否被遍历过，此举是为了避免当某些相同命题变项重复出现时造成重复计数的情况，若该字母并未被遍历过（findProposition函数返回-1），则将该命题变项添加进入proposition中，同时变量n\_proposition加一（计数变量）。在proposition中，保证每个命题变项对应一个整型数。若该字符是运算符，则再priority中寻找该运算符，若能找到(priority.count()方法返回非零正数)则说明该运算符合法，反之非法，给出错误提示，程序退出。该函数返回的Map\_ic型变量储存在主函数的proposition\_set中。proposition\_set的规模即该式子中的变量个数。以上的步骤通过遍历一次命题公式得到了全部命题变项。下面要做的就是计算该命题公式的真值表。

动态申请一段内存m，用于依次存放命题公式各行运算结果的值，便于求主析取范式和主合取范式。（此处调用pow2函数，该函数通过递归方式返回指定数字的二次方的值，因为对于一个有n个命题变项的逻辑表达式而言，其真值表有2^n行）。下一步就是给各个命题变项赋值。

通过循环计数变量，将0~2^n-1转化为相应的二进制数，通过toBinary函数返回各个命题变项的二进制取值（toBinary函数返回一个Map\_ii类型的变量,而proposition\_set本身类型为Map\_ic。借助这两个类型可以实现命题变项和赋值的一一对应）函数返回的结果存储在变量bina\_set中。得到命题变项的赋值后，便可以开始运算（调用calculate函数）。

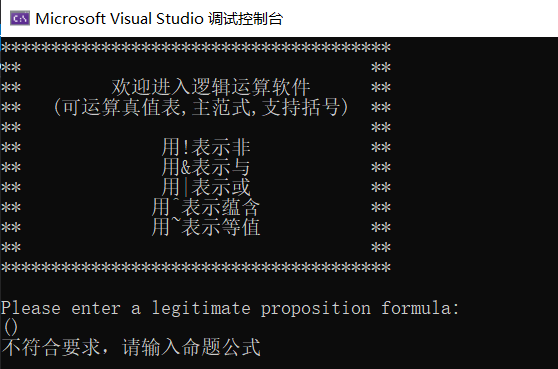
calculate函数有3个参数，string formula是原有命题公式，Map\_ic pSet是该命题公式中的命题变项，Map\_ii value是各个命题变项当前的取值。首先建立运算符栈和运算数栈，先令标记’#’入运算符栈作为运算结束的标志。随后遍历命题公式的各个字符，若遍历到的是命题变项，则令当前命题变项对应的值入运算数栈；若遍历到的字符是运算符，则取目前栈顶的运算符进行比较，若当前栈顶运算符优先级比该运算符优先级高且当前栈顶运算符不是左括号，则通过check函数进行运算，运算结束后对检查运算符栈栈顶，若栈顶元素为’#’，则说明运算结束，直接返回运算数栈栈顶元素（运算最终结果）即可，若遍历到的运算符优先级高于运算符栈栈顶的运算符或当前栈顶运算符是左括号，则将运算符压入栈中。下面介绍真正进行单个运算的check函数。

check函数有两个参数，stack<int>& value用于访问运算数栈，stack<char>& opter用于访问运算符栈。当要调用check函数时即说明需要进行栈顶运算符所代表的运算，switch-case语句就是很好的实现方式。对于不同的运算符，在运算数栈的栈顶取运算数，完成相应运算后将运算结果压入栈顶。值得一提的是蕴含、左括号和右括号三个运算符。由于蕴含运算符不具备交换律，在取运算数时要注意顺序问题，先取到的栈顶元素是蕴含运算的第二个运算数，后取到的栈顶元素是蕴含运算的第一个运算数；取到左括号是不用管，继续向下遍历即可；取到右括号时，则要求将右括号和距离该右括号最近的左括号之间的运算全部处理。

calculate函数的返回值存储在变量result中，代表该取值下的运算结果，随后将该结果存入数组m中（m存储的即为不同取值下的真值）。将上述运算过程循环2^n次，分别将0~2^n-1转化为相应的二进制数，给各个命题变项赋值。整个程序的最后一步就是求主析取范式和主合取范式。遍历数组m，若遍历到1，则说明该下标所对应的简单合取式为极小项，若遍历到0，则说明该下标所对应的简单析取式为极大项，按格式输出主合取范式和主析取范式即可。

**4.心得体会**

这个程序实现起来比较复杂，但引用现有的库可以使程序可读性变得更强。在本项目中就使用了map库来建立映射，便于管理不同命题变项的取值；使用stack库来建立运算符栈和运算数栈，便于更好的模拟逻辑运算。整个程序的逻辑虽然复杂，但通过阅读代码，我进一步加深了对命题公式各种运算的理解，也知道了使用栈这种数据结构能更好地模拟运算，这对我以后编写更复杂的程序提供了更多思路。

以下是效果展示：

