《离散数学》课程实验报告 1 命题逻辑联接词、真值表、主范式

2253893 苗君文 软件工程

1. 题目简介

1.1. 背景与目的

在现代计算机科学中，逻辑是一项关键的概念，而命题逻辑作为逻辑中的一个重要部分，为问题建模和解决提供了强大的工具。命题逻辑主要关注命题之间的逻辑关系，通过联接词、真值表和主范式等概念，我们能够深入理解命题之间的复杂关系，并运用这些知识解决实际问题。

本实验旨在实现以下目标：理解联接词的概念、掌握真值表的制作和应用、熟悉主范式的使用、培养程序设计思维、撰写实验报告的能力。

1.2. 问题描述

（1）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，用户可以选择是否继续输入新的真值进行计算（A）。

（2）B,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）。

1.3. 程序输入输出

1.3.1. 输入

A 题：用户需要通过键盘输入两个逻辑命题变元 P 和 Q 的真值，即 0 或 1。

B 题：用户通过控制台输入命题公式，该公式包含逻辑运算符（如非!、与&、或|、蕴含^、等值~）和命题变量（字母 a-z、A-Z）。用户输入的公式需要符合逻辑运算的语法规则，否则程序会进行错误提示。

1.3.2. 输出

A 题：程序将输出经过逻辑运算后的结果，包括合取（P∧Q）、析取（P∨Q）、条件（P→Q）和双向条件（P↔Q）的真值。

BC 题：程序将根据用户输入的命题公式生成真值表。真值表的列包括命题变量和最终逻辑运算结果。还将计算生成输入命题公式的主析取范式和主合取范式。

1.4. 功能选择

用户可以选择是否继续运行程序。在每次计算完成后，程序会询问用户是否继续运行，用户可以选择输入 'y'（继续运行）或 'n'（退出程序）。

2. 解题思路

2.1. 明确命题逻辑的相关定义

2.1.1. 联结词

（1）非：表示为 ¬，它对命题的真假进行否定。例如，¬p 表示 "非 p"，如果 p 为真，则¬p 为假，反之亦然。

（2）合取：表示为∧，它对两个命题进行"与"操作。例如，p∧q 表示 "p 且 q"，只有当 p 和 q 都为真时， p∧q 才为真。

（3）析取： 表示为∨，它对两个命题进行"或"操作。例如， p∨q 表示 "p 或 q"，只要 p 或 q 有一个为真，p∨q 就为真。p∨q 为假当且仅当 p 与 q 同时为假。

（4）蕴含：表示为→，它描述了一种"如果...那么..."的关系。p→q 表示 "如果 p，则 q"，当 p 为假或者 q 为真时，p→q 为真；p→q 为假当且仅当 p 为真且 q 为假。

（5）等价：表示为↔，它描述了两个命题之间的等价关系。例如，p↔q 表示 "p 当且仅当 q"，p↔q 为真当且仅当 p 与 q 同时为真或同时为假。

各种联结词的优先级：( ),¬, ∧, ∨, →, ↔。

2.1.2. 真值表

真值表是列出了命题的所有可能组合及其结果的表格。通过真值表，我们可以确定复合命题在不同情况下的真假值。命题公式在所有可能的赋值下的取值的列表含 n 个变项的公式有 2^n 个赋值。下图为一个真值表的举例：

p q	¬p	¬q	p∨q	¬(p∨q)	¬p∧¬q	¬(p∨q)↔(¬p∧¬q)
0 0	1	1	0	1	1	1
0 1	1	0	1	0	0	1
1 0	0	1	1	0	0	1
1 1	0	0	1	0	0	1

2.1.3. 主范式

主析取范式：在含有 n 个命题变元的简单合取式中,若每个命题变元与其否定不同时存在,而两者之一出现一次且仅出现一次,则称该简单合取式为极小项。由若干个不同的极小项组成的析取式称为主析取范式;与 A 等价的主析取范式称为 A 的主析取范式。任意含 n 个命题变元的非永假命题公式 A 都存在与其等价的主析取范式,并且是惟一的。

主合取范式：在含有 n 个命题变元的简单析取式中，若每个命题变元与其否定不同时存在，而两者之一出现一次且仅出现一次，称该简单析取式为极大项。由若干个不同的极大项组成的合取式称为主合取范式；与 A 等价的主合取范式称为 A 的主合取范式。任意含 n 个命题变元的非永真命题公式 A 都存在与其等价的主合取范式，并且是惟一的。

2.1.4. 求主范式的步骤

（1）求主析取范式的步骤

设公式 A 含命题变项 $p_1, p_2, \cdots, p_n$

a. 求 A 的析取范式 $A'=B_1 \vee B_2 \vee \cdots \vee B_s$，其中 B_j 是简单合取式 $j=1,2,\cdots,s$

b. 若某个 B_j 既不含 p_i，又不含 $\neg p_i$，则将 B_j 展开成 $B_j \Leftrightarrow B_j \wedge (p_i \vee \neg p_i) \Leftrightarrow (B_j \wedge p_i) \vee (B_j \wedge \neg p_i)$；然后重复这个过程，直到所有简单合取式都是长度为 n 的极小项为止。

c. 消去重复出现的极小项，即用 m_i 代替 $m_i \vee m_i$

d. 将极小项按下标从小到大排列。

（2）求主合取范式的步骤

设公式 A 含命题变项 $p_1, p_2, \cdots, p_n$

a. 求 A 的合取范式 $A'=B_1 \wedge B_2 \wedge \cdots \wedge B_s$，其中 B_j 是简单析取式 $j=1,2,\cdots,s$

b. 若某个 B_j 既不含 p_i，又不含 $\neg p_i$，则将 B_j 展开成 $B_j \Leftrightarrow B_j \vee (p_i \wedge \neg p_i) \Leftrightarrow (B_j \vee p_i) \wedge (B_j \vee \neg p_i)$；然后重复这个过程，直到所有简单析取式都是长度为 n 的极大项为止。

c. 消去重复出现的极大项，即用 M_i 代替 $M_i \wedge M_i$

d. 将极大项按下标从小到大排列。

2.2. 实现步骤

2.2.1. A 题

（1）定义一个 dealInputError 函数，用于处理用户输入的错误情况，确保输入是合法的 0 或 1。

（2)利用循环读取用户输入的逻辑命题变元 P 和 Q 的真值,并进行逻辑运算。

（3）输出合取、析取、条件和双向条件的真值。

（4）根据用户的选择，决定是否继续运行程序。

2.2.2. BC 题

（1）**输入处理与验证**

首先，程序需要从用户处获取输入的命题公式。使用 getInput 函数，该函数通过循环等待用户输入，然后调用 isValidInput 函数来验证输入的命题公式是否符合语法规则。如果输入不合法，程序会提示用户重新输入。

（2）命题变量提取

使用 getProposition 函数从输入的命题公式中提取命题变量，并将其存储在 Map_ic 类型的映射中。该映射将命题变量与其在命题公式中的位置对应起来。

（3）**真值表的生成**

利用提取到的命题变量，通过二进制方式生成真值表。使用 toBinary 函数将一个整数转化为包含 n_proposition 个二进制位的映射，表示命题变量的所有可能取值情况。然后，通过循环计算每种组合下的逻辑运算结果，生成完整的真值表。

（4）**主析取范式和主合取范式的计算**

在生成真值表的基础上，程序计算主析取范式和主合取范式。遍历真值表，将满足条件的情况加入主析取范式或主合取范式。

主析取范式是由真值为 1 的情况通过逻辑或连接而成的。在程序中，通过遍历真值表，调用 calculate 函数计算每种情况的真值,判断每种情况的真值是否为 1，若为 1，则将该情况加入主析取范式。

主合取范式是由真值为 0 的情况通过逻辑与连接而成的。在程序中，通过遍历真值表，调用 calculate 函数计算每种情况的真值，判断每种情况的真值是否为 0，若为 0，则将该情况加入主合取范式。

输出时，根据数量的不同添加相应的连接符。

2.3. 界面设计

本程序为方便使用者调试与使用，增设了让用户判断是否需要继续运行程序的环节。使用循环与判断语句，并做了详尽的输入错误处理。

3. 所用数据结构

3.1. A 题

A 题中所使用的均为基本的数据结构。具体内容如下：

（1）bool：用于存储逻辑运算的结果。程序中定义了一个布尔数组 bool a[4]，用于存放合取、析取、条件和双向条件的真值结果。

（2）int：用于存储命题变元的真值。程序中定义了两个整型变量 `int i, j`，分别表示逻辑命题变元 P 和 Q 的真值。

（3）char： 用于存储用户选择是否继续运行程序的输入。程序中定义了字符变量 char answer，用户可以输入 'y'（继续运行）或 'n'（退出程序）。

（4）string： 用于存储输出提示信息。在 dealInputError 函数中，使用字符串类型 string 存储输出提示信息，以提高交互性。

3.2. BC 题

3.2.1. Map（映射）

映射是一个关联容器，提供了一种将键（key）和值（value）关联起来的方式。在本项目中，本项目中，映射用于存储命题变量和它们在真值表中的对应关系。在代码中，主要用了三种映射：

Map_ci: 用于存储字符到整数的映射，表示运算符的优先级。在项目中，通过这个映射，能够方便地查找运算符的优先级。

Map_ic: 用于存储整数到字符的映射，表示真值表中的命题变量。在真值表的计算中，需要将整数表示的真值表中的某一行映射到具体的命题变量，以便计算该行的真值。

Map_ii: 用于存储整数到整数的映射，表示真值表中命题变量的二进制表示。在真值表计算中，需要将当前行的整数表示转换为对应的二进制形式，以确定命题变量的取值情况。

这些 map 的应用使得程序能够更方便地处理与逻辑运算相关的信息，通过键值对的方式建立了字符到字符、字符到整数、整数到整数的映射关系，提高了程序的可读性和可维护性。

3.2.2. Stack（栈）

在这个项目中， stack 数据结构主要用于两个栈：stack<char> opter 和 stack<int> pvalue。这两个栈在逻辑运算的过程中发挥了关键作用，分别用于存储运算符和操作数。

（1）stack<char> opter - 运算符栈

用途：存储逻辑运算中的运算符，帮助维护运算符的优先级和正确的运算顺序。

操作：入栈（push）、出栈（pop）、获取栈顶元素（top）等。

作用：在逻辑表达式中，通过比较运算符的优先级，将运算符按正确的顺序加入栈中，确保正确的逻辑运算顺序。

在遇到右括号时，从运算符栈中弹出运算符，直到遇到左括号，以确保括号内的运算顺序正确。

通过比较栈顶运算符的优先级，决定是否继续弹出运算符或将当前运算符压入栈中。

（2）stack<int> pvalue - 值栈

用途：存储逻辑运算中的操作数，用于计算逻辑表达式的值。

操作：入栈（push）、出栈（pop）、获取栈顶元素（top）等。

作用：在遇到命题变量或已计算出的部分结果时，将其加入值栈，以便后续的逻辑运算使用。

在进行逻辑运算时，从值栈中弹出操作数，进行相应的计算，并将计算结果压入值栈。

这两个栈的交互，配合逻辑运算的算法，实现了对逻辑表达式的正确计算。在遍历逻辑表达式的过程中，根据运算符的优先级和括号的情况，动态地调整这两个栈的状态，确保逻辑运算的正确性。这种栈的设计使得程序能够灵活地处理各种逻辑运算，并最终生成真值表和主析取/合取范式。

4. 核心算法

4.1. A 题

本逻辑运算程序主要通过用户输入两个命题变元 P 和 Q 的真值，然后进行逻辑运算得到合取、析取、条件和双向条件的真值。核心算法主要包括以下步骤：

（1）输入处理：使用 dealInputError 函数处理用户输入，确保输入的是单个整数，且在指定的范围内（0 或 1）。用户需要输入两个整数，分别表示命题变元 P 和 Q 的真值。

（2）逻辑运算：根据用户输入的 P 和 Q 的真值，进行四种基本的逻辑运算：

合取（&&）：a[0] = i && j;

析取（||）：a[1] = i || j;

条件（->，转化为与或非形式）：a[2] = (!i) || j;

双向条件（<->，转化为与或非形式）：a[3] = ((!i) || j) && ((!j) || i);

```
//读取P的值
dealInputError(i, "请输入P的值（0或1），以回车结束:");
//读取Q的值
dealInputError(j, "请输入Q的值（0或1），以回车结束:");

a[0] = i && j;  //合取
a[1] = i || j;  //析取
a[2] = (!i) || j;  //条件，转化为与或非形式
a[3] = ((!i) || j) && ((!j) || i);  //双向条件，转化为与或非形式

//输出结果
cout << endl << endl << "合取: P/\\Q = " << a[0] << endl;
cout << "析取: P\\/Q = " << a[1] << endl;
cout << "条件: P->Q = " << a[2] << endl;
cout << "双向条件: P<->Q = " << a[3] << endl << endl;
```

（3）输出结果：将四种逻辑运算的结果输出，显示在屏幕上，提供用户清晰的运算结果。

（4）用户选择：用户可以选择是否继续运行程序。使用 while 循环和 if 语句判断用户的选择，若选择继续，则回到输入处理步骤；若选择退出，则结束程序。

整体来说，这个程序的核心算法是基于用户输入的两个命题变元的真值，通

过逻辑运算得到相应的结果，然后输出给用户。程序主要使用了基本的逻辑运算符和控制结构，没有复杂的算法。

4.2. BC 题

4.2.1. 输入命题公式

（1）用户交互

用户通过键盘输入一个逻辑表达式，该表达式可以包含字母表示的命题变量、逻辑运算符和括号。用户输入的过程需要进行逐字符检查，以确保输入的表达式是合法的。

（2）合法性检查

在用户输入命题公式时，需要进行详细的合法性检查，以确保输入的逻辑表达式是符合规范的。

a. 字符合法性检查

检查每个字符是否为合法字符，包括字母表示的命题变量、逻辑运算符和括号。

字母合法性：确保字母只包含大写或小写字母，用于表示命题变量。不允许其他字符混入命题变量。

运算符合法性：验证逻辑运算符的合法性，包括与&、或|、非!、蕴含^、等值~。

括号合法性：确保只包含左括号(和右括号)，且左右括号数量相等。

b. 字母相对顺序检查

检查相邻字符之间的相对顺序，确保满足逻辑表达式的语法。

字母相对顺序：字母之间不能有其他字母，确保命题变量的正确使用。

运算符相对顺序：确保运算符前后有字母或右括号等，避免运算符的错误使用。

c. 括号匹配检查

用一个括号计数器维护，确保左右括号数量相等且顺序正确。

括号数量匹配：左右括号数量应当相等，不得有多余或缺失的括号。

括号顺序匹配：左括号必须在相应的右括号之前。

通过详细的合法性检查，确保用户输入的逻辑表达式在语法上是正确的，提

高了程序的容错性和用户体验。

4.2.2. 逻辑表达式的计算

（1）**遍历逻辑表达式：** 从左到右遍历逻辑表达式的每个字符。

（2）**处理命题变量：**当遇到字母（命题变量）时，将其对应的值（从真值表中取得）入操作数栈。

（3）**处理运算符：**

如果运算符栈为空，或者当前运算符为左括号，直接将当前运算符入栈。

如果当前运算符的优先级高于栈顶运算符，将当前运算符入栈。

否则，弹出栈顶运算符直到满足上述条件，然后将当前运算符入栈。

（4）**处理右括号：**遇到右括号时，不断弹出运算符栈的元素并计算，直到遇到左括号。左括号出栈，继续处理逻辑表达式。

（5）**处理最后的运算符：**最后，如果运算符栈不为空，依次弹出栈中剩余运算符并进行计算。

（6）**计算过程：**在处理过程中，每当有新的操作数入栈或运算符进行计算，都会立即更新操作数栈和运算符栈。这样通过栈的先入后出的性质，保证了正确的计算顺序。

通过这样的设计，实现了对逻辑表达式的动态计算，确保了正确的运算顺序和优先级。整个计算过程模拟了中缀表达式转后缀表达式的思想，通过栈的操作实现了逻辑运算符的正确计算。

4.2.3. 生成真值表

（1）获取命题变量

函数名: Map_ic getProposition(string formula)

功能: 从给定的逻辑公式 formula 中提取命题变量，并将其存储在一个映射 proposition 中，最终返回该映射。

实现细节：

定义映射 Map_ic proposition 用于存储命题变量。

遍历输入的逻辑公式中的每个字符。

如果当前字符是字母（命题变量），则检查其是否已经在 proposition 中存在。如果存在，说明该命题变量已经被遍历过，不重复计数。

如果不存在,将该命题变量存储在 proposition 映射中,键为 n_proposition,即当前的命题变量的个数。

增加命题变量的个数 n_proposition。

如果当前字符是运算符，直接忽略。

返回值: 返回存储了命题变量的映射 proposition。

（2）计算逻辑表达式

函数名: int calculate(string formula, Map_ic pSet, Map_ii value)

功能: 计算给定命题变量的值组合 value 下的逻辑运算结果。

实现细节:

使用两个栈，stack<char> opter 用于存储运算符，stack<int> pvalue 用于存储操作数。

遍历输入的逻辑表达式 formula 中的每个字符。

对于命题变量，将其对应的值压入操作数栈 pvalue。

对于运算符，通过比较优先级和栈顶运算符，决定是将当前运算符压入栈还是弹出栈顶运算符并进行计算。

对于括号，确保括号内的逻辑表达式得到正确计算。

最终，操作数栈中存储的即为逻辑表达式的计算结果。

返回值: 返回逻辑表达式的计算结果。

（3）生成真值表

函数名: void generateTruthTable(string formula)

功能: 根据给定的逻辑公式生成真值表，并输出结果。

实现细节:

获取命题变量集合 pSet 和逻辑公式 formula。

遍历命题变量的所有可能取值组合，使用二进制表示。

对每个取值组合，调用逻辑表达式计算的算法 calculate 得到结果。

输出每行的命题变量取值和对应的逻辑运算结果。

整体上，生成真值表的过程主要包括获取命题变量、计算逻辑表达式，通过遍历命题变量的所有可能取值组合，得到逻辑表达式的真值表。

（4）主析取/合取范式的计算

遍历命题变量的所有可能取值组合，计算逻辑表达式的值。记录使逻辑表达式为真的情况，得到主析取范式。记录使逻辑表达式为假的情况，得到主合取范式。

最后，输出命题变量及其对应的真值表及主析取范式和主合取范式。

4.3. 输入错误处理相关函数

4.3.1. 单个参数的输入错误处理

函数 dealInputError 使用 while 循环不断尝试获取用户输入，直到输入满足要求。使用 cin.fail() 及 min、max 来判断输入是否出错，检查输入是否在有效范围内。如果输入无效，则输出错误信息，清除输入缓冲区并忽略之后的字符，并要求重新输入，重新输入的提示内容也由函数所传的参数 str 决定。通过一个字符来获取输入一个数之后的字符以检查输入字符的个数，若不正确，也输出错误信息，执行相应清除操作。如果输入有效，则跳出循环。

此函数可以用于 A 题中判断输入的 P，Q 的值的正误。

```
void dealInputError(int& n, string str, int min = 0, int max = 1)  //处理输入错误
{
    while (1) {
        char c;  //用来获取输入一个数之后的一个字符，若获取一个字符且非换行符则视为输入错误。
        cout << str << endl;
        cin >> n;
        if (cin.fail() || n<min || n>max) {
            cout << "输入错误，请重新输入。" << endl;
            cin.clear();
            cin.ignore(9999, '\n');
            continue;
        } //内容错误
        else if (cin.get(c) && c != '\n') {
            cout << "输入错误，请重新输入。" << endl;
            cin.clear();
            cin.ignore(9999, '\n');
            continue;
        } //个数错误
        else
            break;
    }
    return;
}
```

4.3.2. 判断是否继续运行程序的错误处理

此处输入的内容是字符，程序将大小写的 y/n 输入都作为正确输入，而其他情况需要重新输入。类似之前，分为内容错误和个数错误两种，并根据输入正确的内容需不需要继续运行。Y 则 continue，N 则 break。

此错误处理在 A 题和 BC 题中均有使用。

5. 心得体会

5.1. A 题

通过完成 A 题逻辑运算程序的编写，我深刻理解了逻辑运算的基本概念和原理。通过代码的实现，我加深了对合取、析取、条件和双向条件等逻辑运算的理解。此外，通过编写输入错误处理的代码，我学会了如何提高程序的健壮性，使其在用户输入错误时能够进行适当的处理，提高用户体验。这次编程任务使我更加熟悉了 C++ 语言的基本语法和面向对象编程的思想，对我的编程能力提升有很大帮助，也为我后续离散数学的学习打下深厚的基础。

5.2. BC 题

在项目中，通过合理设计数据结构和模块化的函数，使得代码具有良好的可读性和可维护性。合理利用函数、类等代码结构，实现了逻辑运算软件的核心功能，同时也为未来的功能扩展提供了可靠的基础。

为了能够使程序判断每一种输入错误的情况，我特地在草稿纸上列出了各种可能出现的输入错误，并将其归结为以下这几类：首字符错误、多个字母连续、非法字符出现、括号前是字母、最后一个字符非法、括号不匹配，并在程序中专门设计了 isValidInput 来实现。并且花了大量的时间去调试这一部分的输入错误处理，保证所有情况都被考虑。

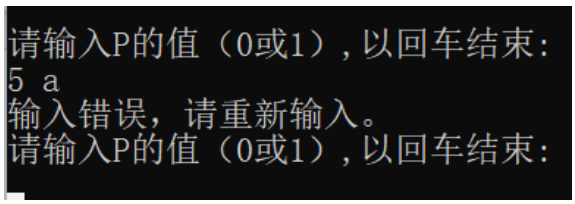
在原程序中，有一个“#”用来判断命题公式有没有完全计算。为了删去这个“#”运算符，也就是要设定一个不存在“#”时的计算终止条件。因此，我修改了 calculate 函数的结束运算条件。在 calculate 函数中，结束运算的条件是遍历完整个逻辑公式字符串 formula 并且运算符栈 opter 为空。这意味着当函数处理完最后一个字符时，它会检查运算符栈是否为空。如果运算符栈中仍有运算符未处理，就会继续进行计算，直到运算符栈为空为止。这是因为可能存在一些运算符尚未与其对应的操作数进行计算。最终，函数返回值栈 pvalue 的栈顶元素，即最终的逻辑运算结果。

6. 测试结果

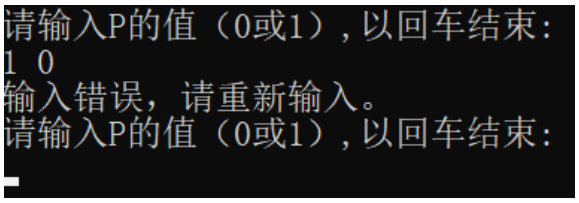
6.1. A 题

6.1.1. 输入 P 的值

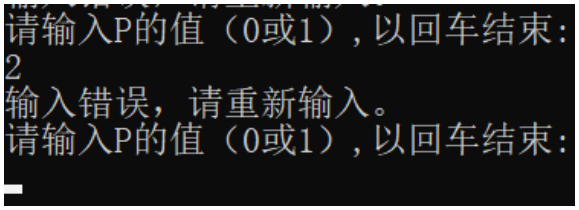
（1）输入个数及内容错误



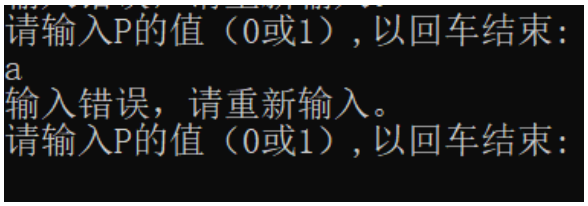
（2）输入个数错误



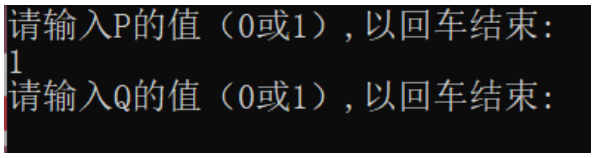
（3）输入的值错误



（4）输入的类型错误

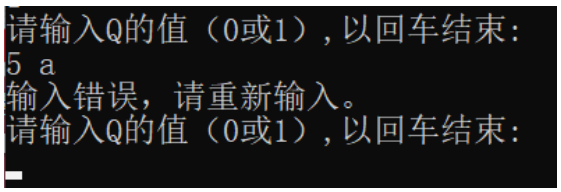


（5）输入正确即可进入下一环节

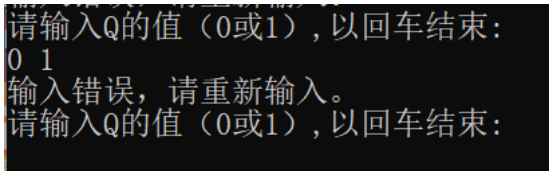


6.1.2. 输入 Q 的值

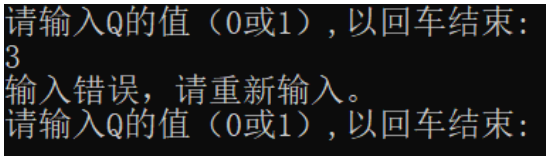
（1）输入个数及内容错误



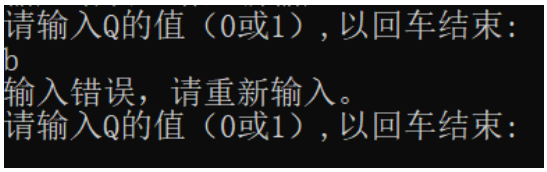
（2）输入个数错误



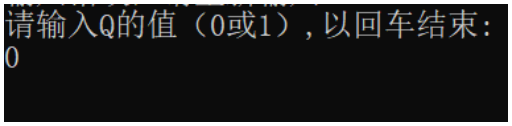
（3）输入的值错误



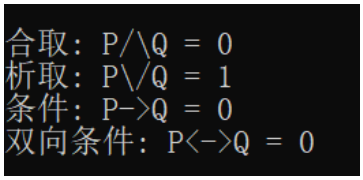
（4）输入的类型错误



（5）输入正确即可进入下一环节

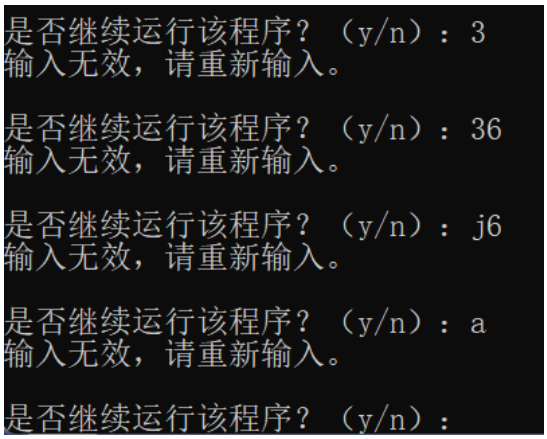


6.1.3. 输出逻辑运算结果

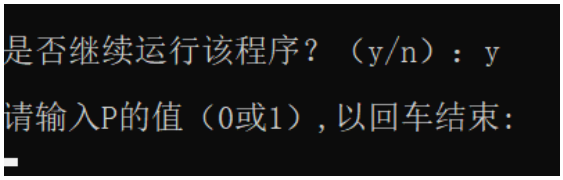


6.1.4. 选择是否继续运行该程序

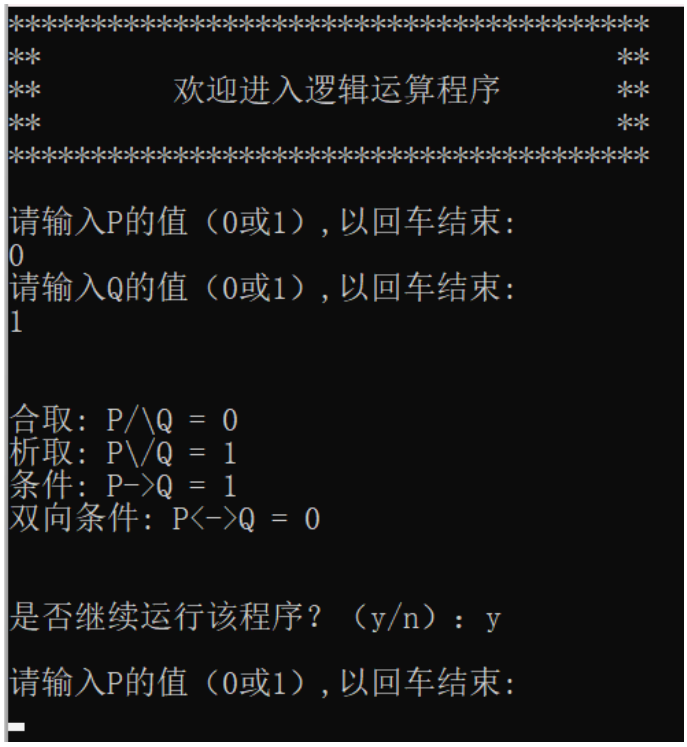
（1）输入错误



（2）输入 y/Y/n/N 均输入正确（下图以 y 为例）

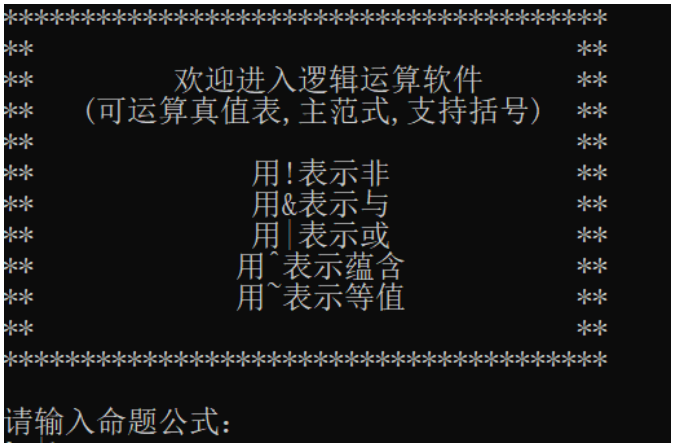


6.1.5. A 题输入正确下的程序总览



6.2. B，C 两题

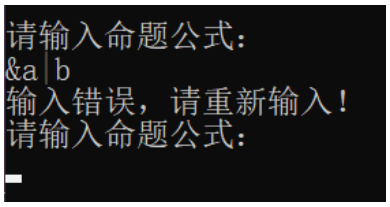
6.2.1. 欢迎界面



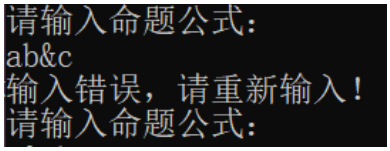
6.2.2. 输入命题公式错误的情况

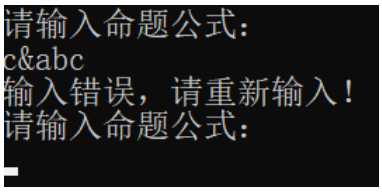
（1）输入的首字符错误

首字符只能为字母字符或 “(” 或 “!”



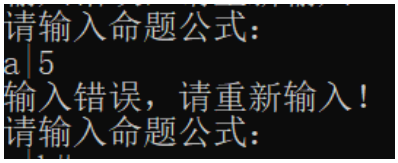
（2）输入内容中有两个及以上字母连续

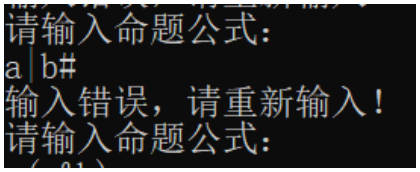




（3）输入内容存在非法字符

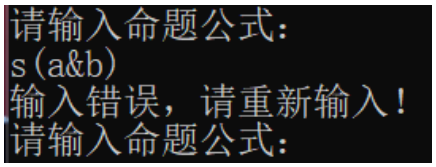
本程序中仅有字母字符与规定的逻辑运算字符是合法的。





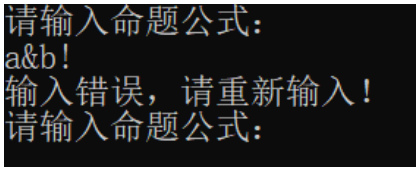
（4）输入的命题公式中括号前是字母

按照公式的规律，括号前只能是逻辑运算符号。

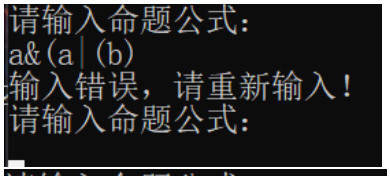


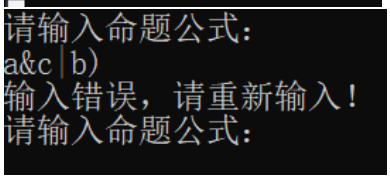
（5）输入的命题公式中最后一个字符非法

命题公式中的最后一个字符只能是字母字符或 “)”。



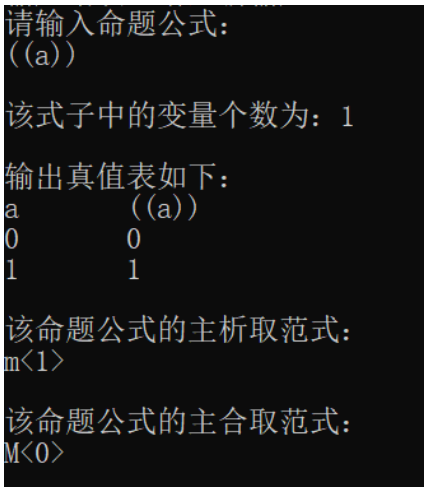
（6）输入的命题公式中括号不匹配



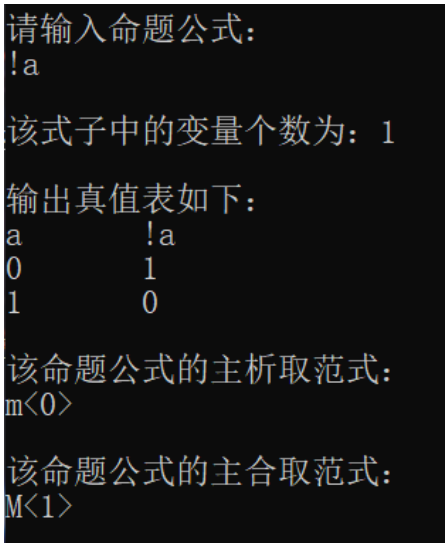


6.2.3. 输入的命题公式正确的情况，并输出对应的真值表、主析取范式、主合取范式

（1）测试内容：((a))（命题公式中变量带多层括号）



（2）测试内容：!a（非运算）



（3）测试内容：a&b（与运算）

```
请输入命题公式:
a&b

该式子中的变量个数为: 2

输出真值表如下:
a         b         a&b
0         0         0
0         1         0
1         0         0
1         1         1

该命题公式的主析取范式:
m<3>

该命题公式的主合取范式:
M<0> /\ M<1>  /\ M<2>
```

（4）测试内容：a|b（或运算）

```
请输入命题公式:
a|b

该式子中的变量个数为: 2

输出真值表如下:
a         b         a|b
0         0         0
0         1         1
1         0         1
1         1         1

该命题公式的主析取范式:
m<1> \/ m<2>  \/ m<3>

该命题公式的主合取范式:
M<0>
```

（5）测试内容：a^b（蕴含式运算）

```
请输入命题公式:
a^b

该式子中的变量个数为: 2

输出真值表如下:
a       b       a^b
0       0       1
0       1       1
1       0       0
1       1       1

该命题公式的主析取范式:
m<0> \/ m<1>  \/ m<3>

该命题公式的主合取范式:
M<2>
```

（6）测试内容：a~b（等值式运算）

```
请输入命题公式:
a^b

该式子中的变量个数为: 2

输出真值表如下:
a       b       a~b
0       0       1
0       1       0
1       0       0
1       1       1

该命题公式的主析取范式:
m<0> \/ m<3>

该命题公式的主合取范式:
M<1> /\ M<2>
```

（7）测试内容：!(a&b)（含有括号的简单运算）

```
请输入命题公式:
!(a&b)

该式子中的变量个数为: 2

输出真值表如下:
a       b       !(a&b)
0       0       1
0       1       1
1       0       1
1       1       0

该命题公式的主析取范式:
m<0> \/ m<1>  \/ m<2>

该命题公式的主合取范式:
M<3>
```

（8）测试内容：a&(b)（命题公式中单个变量带有括号）

```
请输入命题公式：
a&(b)

该式子中的变量个数为：2

输出真值表如下：
a       b       a&(b)
0       0       0
0       1       0
1       0       0
1       1       1

该命题公式的主析取范式：
m<3>

该命题公式的主合取范式：
M<0> /\ M<1>  /\ M<2>
```

（9）测试内容：a^b~c|b&a^c（不带括号的综合命题公式）

```
请输入命题公式：
a^b~c|b&a^c

该式子中的变量个数为：3

输出真值表如下：
a       b       c       a^b~c|b&a^c
0       0       0       1
0       0       1       1
0       1       0       1
0       1       1       1
1       0       0       0
1       0       1       0
1       1       0       0
1       1       1       1

该命题公式的主析取范式：
m<0> \/ m<1>  \/ m<2>  \/ m<3>  \/ m<7>

该命题公式的主合取范式：
M<4> /\ M<5>  /\ M<6>
```

（10）测试内容：!a^b~(c|d~(!b&c)^!d)|a（带括号的综合命题公式）

```
请输入命题公式:
!a^b~(c|d~(!b&c)^!d)|a

该式子中的变量个数为：4

输出真值表如下:
a       b       c       d       !a^b~(c|d~(!b&c)^!d)|a
0       0       0       0       1
0       0       0       1       0
0       0       1       0       0
0       0       1       1       1
0       1       0       0       0
0       1       0       1       1
0       1       1       0       1
0       1       1       1       1
1       0       0       0       1
1       0       0       1       1
1       0       1       0       1
1       0       1       1       1
1       1       0       0       1
1       1       0       1       1
1       1       1       0       1
1       1       1       1       1

该命题公式的主析取范式:
m<0> \/ m<3>  \/ m<5>  \/ m<6>  \/ m<7>  \/ m<8>  \/ m<9>  \/ m<10>  \/ m<11>  \/ m<12>  \/ m<13>  \/ m<14>  \/ m<15>

该命题公式的主合取范式:
M<1> /\ M<2>  /\ M<4>
```

6.2.4. 选择是否继续运行该程序

（1）输入错误

```
是否继续运行该程序？（y/n）：3
输入无效，请重新输入。

是否继续运行该程序？（y/n）：36
输入无效，请重新输入。

是否继续运行该程序？（y/n）：j6
输入无效，请重新输入。

是否继续运行该程序？（y/n）：a
输入无效，请重新输入。

是否继续运行该程序？（y/n）：
```

（2）输入 y/Y/n/N 均输入正确（下图以 y 为例）

```
是否继续运行该程序？（y/n）：y

请输入命题公式:
```

6.2.5. B,C 题输入正确下的程序总览

```
****************************************
**                                    **
**          欢迎进入逻辑运算软件      **
**    (可运算真值表,主范式,支持括号)  **
**                                    **
**                用!表示非           **
**                用&表示与           **
**                用|表示或           **
**                用^表示蕴含         **
**                用~表示等值         **
**                                    **
****************************************

请输入命题公式:
!a^b~(c|d~(!b&c)^!d)|a

该式子中的变量个数为: 4

输出真值表如下:
a       b       c       d       !a^b~(c|d~(!b&c)^!d)|a
0       0       0       0       1
0       0       0       1       0
0       0       1       0       0
0       0       1       1       1
0       1       0       0       0
0       1       0       1       1
0       1       1       0       1
0       1       1       1       1
1       0       0       0       1
1       0       0       1       1
1       0       1       0       1
1       0       1       1       1
1       1       0       0       1
1       1       0       1       1
1       1       1       0       1
1       1       1       1       1

该命题公式的主析取范式:
m<0> \/ m<3>  \/ m<5>  \/ m<6>  \/ m<7>  \/ m<8>  \/ m<9>  \/ m<10>  \/ m<11>  \/ m<12>  \/ m<13>  \/ m<14>  \/ m<15>

该命题公式的主合取范式:
M<1> /\ M<2>  /\ M<4>

是否继续运行该程序? (y/n): _
```