|  |
| --- |
| 离散数学实验报告 |
| 学号：1120210526\_\_ 姓名：\_马煜\_\_\_ |

**目 录**

[1 求命题的主范式 1](#_Toc148558911)

[1.1 **概述** 1](#_Toc148558912)

[1.2 **步骤流程** 1](#_Toc148558913)

[1.3 **程序实现** 1](#_Toc148558914)

[1.3.1 **OperatorRank** 2](#_Toc148558915)

[1.3.2 **evalRPN** 2](#_Toc148558916)

[1.3.3 **GetElements** 3](#_Toc148558917)

[1.3.4 **assign** 3](#_Toc148558918)

[1.3.5 **find** 3](#_Toc148558919)

[1.3.6 **Calculate** 4](#_Toc148558920)

[1.3.7 **cal** 4](#_Toc148558921)

[2 消解算法 4](#_Toc148558922)

[2.1 **概述** 4](#_Toc148558923)

[2.2 **步骤流程** 4](#_Toc148558924)

[2.3 **程序实现** 5](#_Toc148558925)

[2.3.1 **init** 5](#_Toc148558926)

[2.3.2 **resolve** 6](#_Toc148558927)

**表目录**

[表格 1求命题的主范式函数说明表 1](#_Toc148558905)

[表格 2 消解算法函数说明表 5](#_Toc148558906)

**图目录**

[图 1 程序运行主逻辑 1](#_Toc148558900)

[图 2 evalRPN 函数执行流程 3](#_Toc148558901)

[图 3 消解逻辑图 6](#_Toc148558902)

# 求命题的主范式

## **概述**

输入命题公式的合式公式，求出公式的真值表，并输出该公式的主合取范式和主析取范式。

## **步骤流程**

程序运行主逻辑如图1所示。为单线程程序。

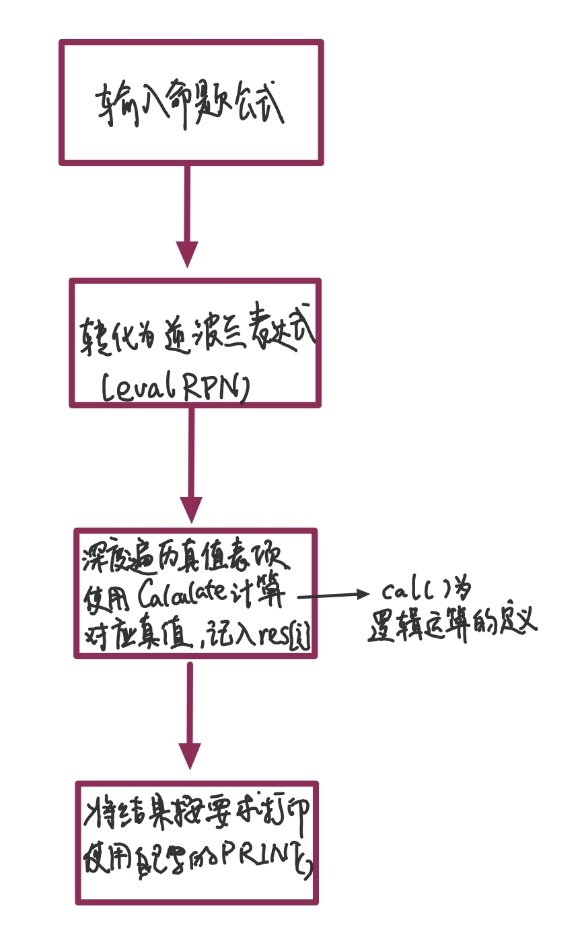


图 1 程序运行主逻辑

## **程序实现**

表格 1求命题的主范式函数说明表

| 序号 | 名称 | 说明 |
| --- | --- | --- |
|  | evalRPN | 用于将输入的字符串转成逆波兰表达式 |
|  | GetElements | 从逆波兰表达式中提取所有命题变量 |
|  | assign | 遍历得到真值表并且打印结果 |
|  | find | 在字母表中找到命题变量在二进制中的位数 |
|  | Calculate | 计算二进制真值结果，返回01 |
|  | cal | 布尔运算定义 |
|  | OperatorRank | 处理运算符的优先级帮助转换为逆波兰表达 |

### **OperatorRank**

用于处理运算符的优先级，其中 ）,！ 优先级最高，其次是&、|、-、+、（。

### **evalRPN**

用于将输入的字符串转成逆波兰表达式，要使用函数 OperatorRank作为辅助。 依次遍历整个字符串，如果为字母，则直接输出到存储结果的数组中，如果为运算符&、|、+、-、（、），则需要与栈顶运算符的优先级进行比较，具体实现请见图2

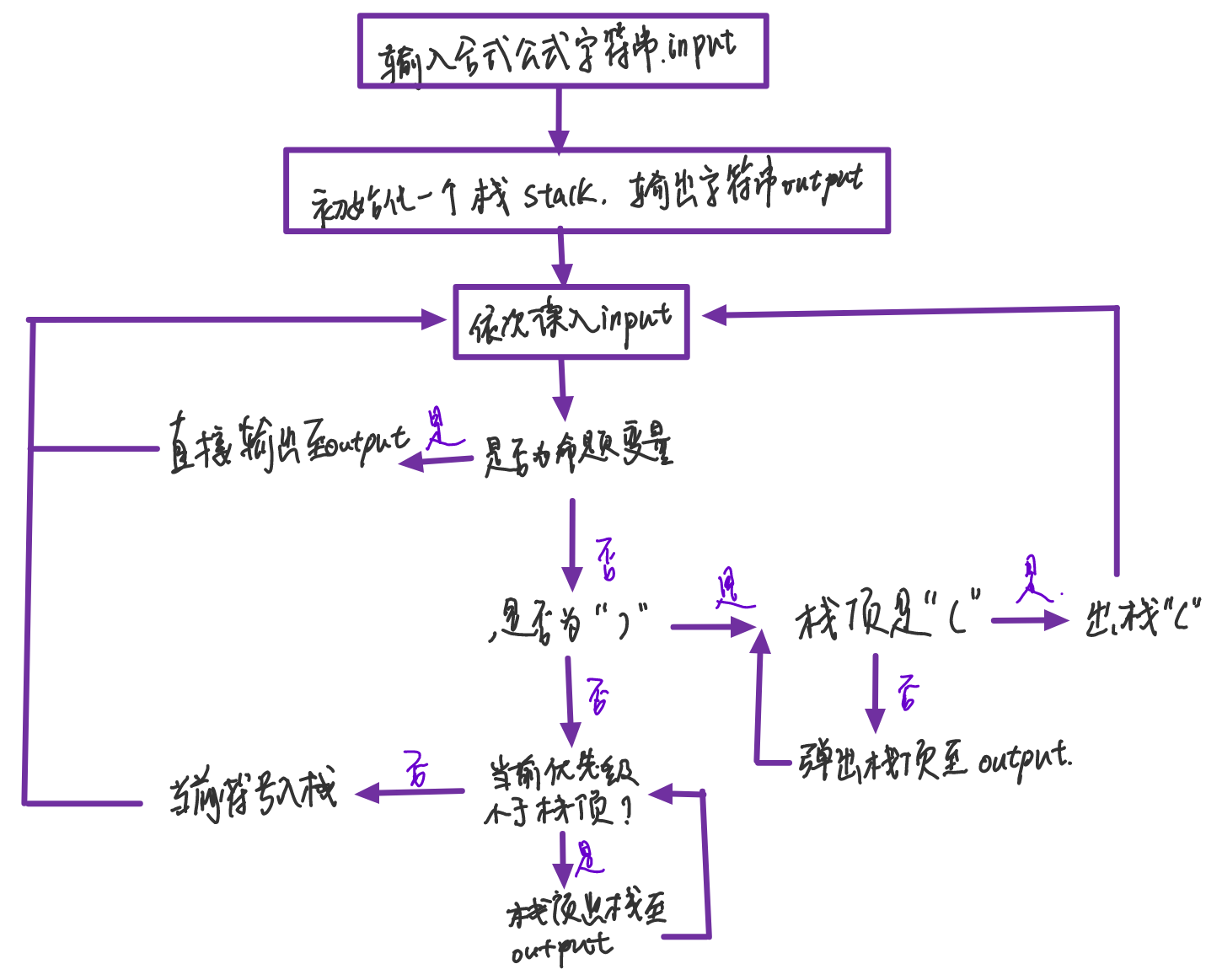


图 evalRPN 函数执行流程

### **GetElements**

从逆波兰表达式中提取所有命题变量遍历字符串，将出现的命题变量提取并且按顺序排序。返回值的长度就是命题变量的数量。

### **assign**

记命题变量数量为n，建立2^n的循环，每次循环将计数的二进制赋值给（赋值过程中用到了find函数，在变量中找到逆波兰表达式中该位变量的索引，将对应的二进制数放进num\_binary中。num\_binary[i] = binary[find(variables, num\_binary[i])];）逆波兰表达式，并使用Calculate（）进行计算，得到真值表，最后使用PRINT（）进行结果打印。

### **find**

返回变量位置的索引

int find(string variables, char a)

{

for (int i = 0; i < variables.size(); i++)

{

if (variables[i] == a)

return i;

}

}

### **Calculate**

计算二进制的真值

for (unsigned int i = 0; i < a.**size**(); i++)

        {

            if (**is\_in**(a[i], "+|&-!"))

            {

                if (a[i] != '!')

                {

                    a[i - 2] = char(**cal**(int(a[i - 2] - '0'), int(a[i - 1] - '0'), a[i]) + '0');

                    a.**erase**(a.**begin**() + i - 1);

                    a.**erase**(a.**begin**() + i - 1);

                    break;

                }

                else

                {

                    a[i - 1] = char(**cal**(int(a[i - 1] - '0'), 0, a[i]) + '0');

                    a.**erase**(a.**begin**() + i);

                    break;

                }

            }

        }

### **cal**

里面定义了所有的逻辑运算符的计算方法

# 消解算法

## **概述**

输入合式公式 A 的合取范式，当 A 是可满足时，回答“YES ”。

## **步骤流程**

程序运行主逻辑如下：

* + - 1. 求A的合取范式S
      2. 令 S0和S2为不含任何元素的集合S1为S的所有简单析取式组成的集合
      3. 对S0中的每一个简单析取式 C1与S1中的每一个简单析取式C2
      4. 如果 C1, C2,可以消解,则
      5. 计算 C=Res(C1,C2)
      6. 如果 C=空,则
      7. 输出“no”,计算结束
      8. 如果 S0与 S2都不包含 C,则
      9. 把C加入S2
      10. 对S中的每一对子句 C1,C2
      11. 如果 C1,C2,可以消解,则
      12. 计算 C=Res(C1,C2)
      13. 如果 C=空,则
      14. 输出“no”,计算结束
      15. 如果 S0,与 S1，都不包含 C,则
      16. 把C加入 S2
      17. 如果 S2中没有任何元素,则
      18. 输出“yes”,计算结束
      19. 否则把 S1加入S0,令 S1,等于 S2,,清空 S2,返回 3

## **程序实现**

表格 2 消解算法函数说明表

| 序号 | 名称 | 说明 |
| --- | --- | --- |
| 1. | init | 用于将输入的格式化，存取每个简单析取式 |
| 2. | not\_in | 判断消解后的结果在不在S1和S2中 |
| 3. | same | 判断两个简单析取式是否相同 |
| 4. | resolve | 两两消解 |

### **init**

每行存储一个简单析取式，第二维下标0~25代表命题变项a~z

取值 0: 该变项没有出现，1: 该变项出现，2: 该变项出现且为否定

### **resolve**

用于两个简单析取式的消解。具体逻辑见下图

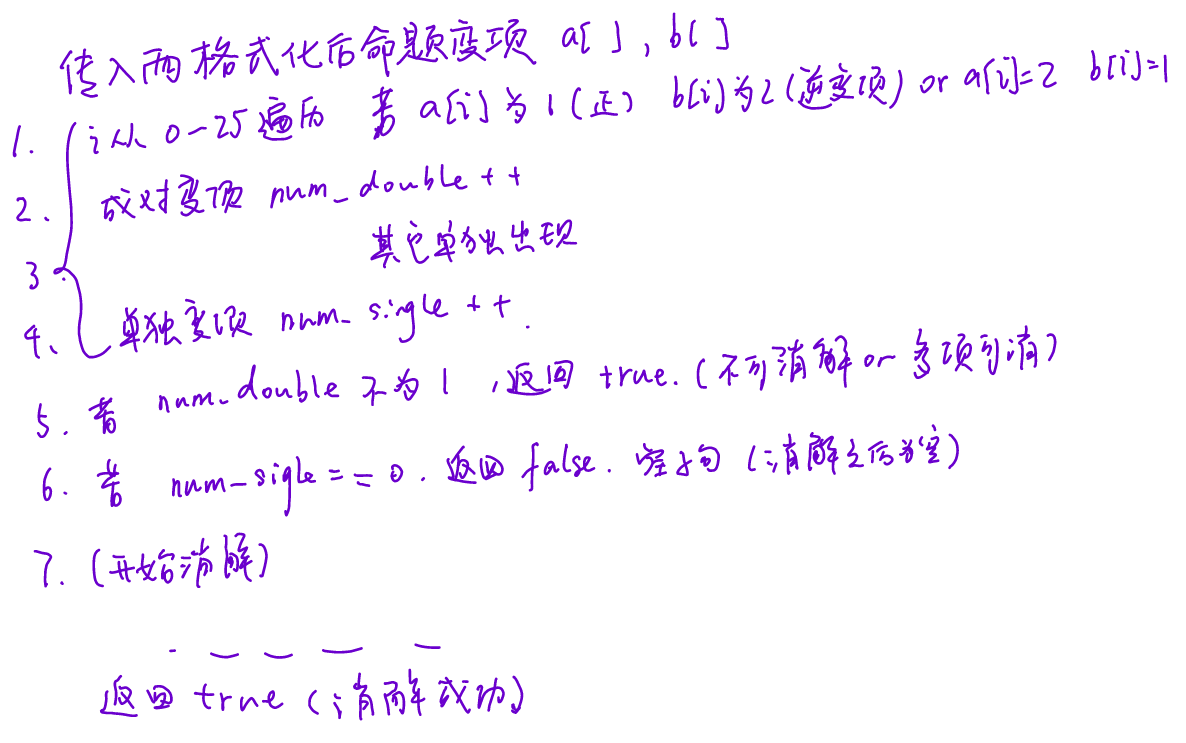


图 3 消解逻辑图