# 《离散数学》课程实验报告1 命题逻辑联接词、真值表、主范式

# 题目简介；

本题涉及命题逻辑中的命题变元、真值表、合取、析取、条件、双向条件、主范式等概念。第一部分要求输入两个命题变元P和Q的真值，计算它们的合取、析取、条件和双向条件的真值。第二部分要求求任意一个命题公式的真值表，并根据真值表求主范式。

# 二.解题思路；：

（A）

1. 输入命题变元P和Q的真值。

2. 根据输入的真值，计算P和Q的合取、析取、条件和双向条件的真值。

3. 合取的真值为P和Q同时为真时为真，否则为假；析取的真值为P和Q至少有一个为真时为真，否则为假；条件的真值为当P为真且Q为假时为假，否则为真；双向条件的真值为P和Q相同时为真或者同时为假时为真，否则为假。

（B）

1. 输入命题公式，并确定其中包含的命题变元。

2. 根据命题变元的不同真值组合，列出真值表，包括各命题变元的真值和整个命题公式的真值。

3. 根据真值表，确定主析取范式和主合取范式：

- 主析取范式是使命题公式为真的最小项的析取；

- 主合取范式是使命题公式为假的最大项的合取。

# 三.数据结构；

# 1. 使用一个名为`a`的整型数组来存储逻辑运算结果，数组的下标对应着逻辑运算的类型（如合取、析取、条件、双条件）。

# 2. 使用两个布尔型变量`i`和`j`来分别存储输入的P和Q的真值。

# 3. enum定义了一个枚举类型，用于表示不同的逻辑运算类型（如AND、OR、Implication、Equality）

# 四. 实验原理和实现过程（算法描述）；

## 1.实验原理

（1）合取：合取是二元命题联结词，用于将两个命题P和Q联结起来，构成一个新的命题P∧Q，读作P与Q的合取。合取的真值与构成它的命题P和Q的真值之间的关系为只有当P和Q都为真时，P∧Q才为真；只要P或者Q中有一个为假，P∧Q就为假。合取通常用来表示日常用语P与Q或P并且Q。

（2）析取：析取是二元命题联结词，用于将两个命题P和Q联结起来，构成一个新的命题P∨Q，读作P或Q的析取。析取的真值与构成它的命题P和Q的真值之间的关系为只有当P和Q都为假时，P∨Q才为假；只要P或者Q中有一个为真，P∨Q就为真。析取通常用来表示日常用语P或者Q。

（3）条件：条件是二元命题联结词，用于将两个命题P和Q联结起来，构成一个新的命题P→Q，读作P条件Q或者如果P，那么Q。条件的真值与构成它的命题P和Q的真值之间的关系为只有当P为真且Q为假时，P→Q才为假，其余情况下P→Q为真。

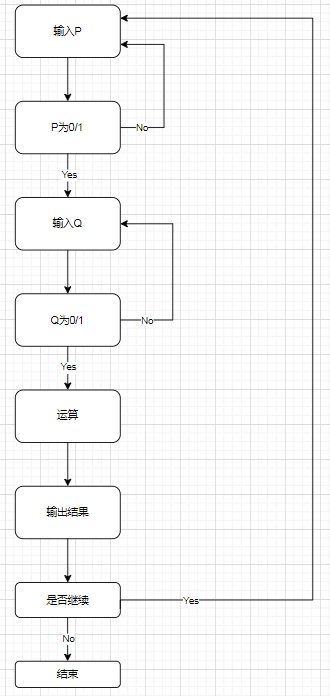
（4）双向条件：双向条件是二元命题联结词，用于将两个命题P和Q联结起来，构成一个新的命题P↔Q，读作P双条件于Q。双向条件的真值与构成它的命题P和Q的真值之间的关系为当P和Q同时为真或者同时为假时，P↔Q才为真，其余情况下P↔Q为假。

（5）真值表：真值表是用来表征逻辑事件输入和输出之间全部可能状态的表格，通常用1表示真，0表示假。真值表列出了命题公式的真假值，其中命题公式的取值由组成命题公式的命题变元的取值和命题联结词的真值表决定。

（6）主范式：主析取范式和主合取范式分别用于表示命题公式的最小项和最大项的析取和合取形式。主析取范式由若干个不同的极小项组成的析取式，主合取范式由若干个不同的极大项组成的合取式。任意含n个命题变元的非永假命题公式都存在与其等价的主析取范式，以及任意含n个命题变元的非永真命题公式都存在与其等价的主合取范式，而且是唯一的。

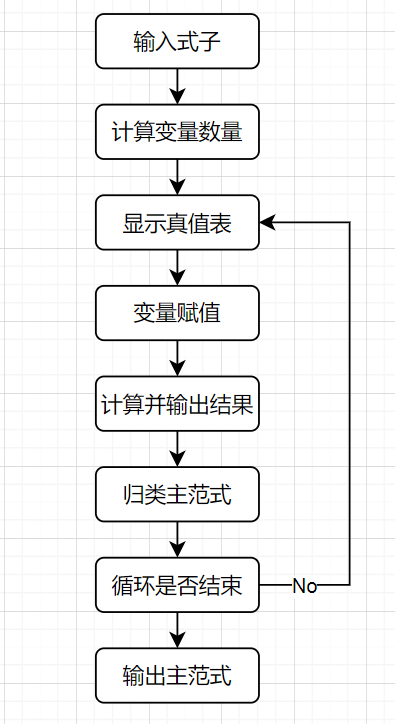
## 2.实验过程

（1）A题部分涉及对输入变量的处理，需要确保输入的值为0或1，否则将被视为错误。随后进行逻辑运算处理，在C语言中，可以使用逻辑非（!）、逻辑与（&&）、逻辑或（||）来表示与、或、非的逻辑运算。对于不是与、或、非的情况，在这个实验中可以通过转化为与、或、非的形式来进行处理。具体流程图如下：



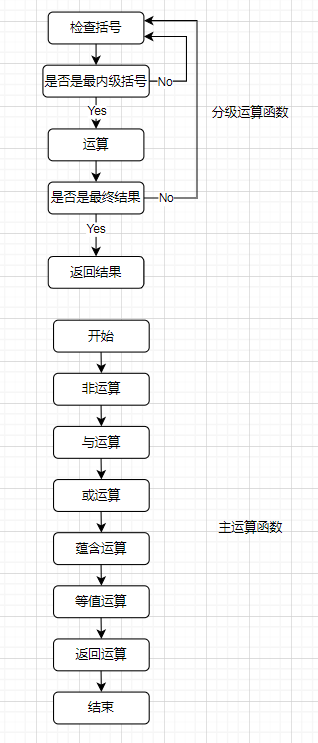
（2）B题部分要求首先输入一个合理的命题公式，然后从式子中查找出变量的个数，开辟一个二进制函数来生成真值表，并利用该函数计算结果。随后，根据计算结果归类给出主析取范式和主合取范式，并最终输出范式。

在函数部分，主要包括三个函数。第一个函数是真值表递加函数，通过二进制加法原理递进产生真值表。第二个函数是分级运算函数，通过判断括号，选出最内级括号的内容执行运算函数，逐级向外运算，得出最终结果。第三个函数是主运算函数，按照运算符号的优先级按顺序进行运算。



**主函数**

。



# 五.部分核心代码；

for (unsigned i1 = 0; i1 < strlen(sz); i1++) // 统计命题公式中的变量个数

{

if (sz[i1] == ')' || sz[i1] == '(')

kh++;

if (sz[i1] >= 'a' && sz[i1] <= 'z' || sz[i1] >= 'A' && sz[i1] <= 'Z') // 统计命题公式中的变量

{

for (i2 = 0; i2 < j; i2++)

if (ccu[i2] == sz[i1])

d = 0;

if (d == 1)

{

ccu[j] = sz[i1];

j++;

}

d = 1;

}

}

std::cout << "\n该式子中的变量个数为:" << j << "\n"; // 输出变量个数

h0 = j; // 将变量个数保存为h0

std::cout << "\n输出真值表如下:\n \n"; // 输出真值表的表头

for (i1 = 0; i1 < h0; i1++)

std::cout << " " << ccu[i1] << " "; // 输出变量名

std::cout << " ";

std::cout << sz << "\n\n"; // 输出命题公式

for (i1 = 0; i1 < j; i1++)

icu[i1] = 0; // 初始化真值表

for (i2 = 0; i2 < j; i2++)

std::cout << " " << icu[i2] << " "; // 输出真值表初始状态

jg = tkh(sz, ccu, icu, h0); // 计算命题公式的真值

if (jg == 0)

hq[h++] = bj; // 将真值为0的行数存入hq数组

else

xq[x++] = bj; // 将真值为1的行数存入xq数组

std::cout << " " << jg << "\n"; // 输出计算结果

strcpy\_s(sz, sz0); // 恢复原始命题公式

for (i1 = 0; i1 < (int)pow(2.0, j) - 1; i1++) // 对真值表的每一行进行计算

{

++bj; // 行数加1

panduan(icu, j - 1); // 生成下一行真值

jg = tkh(sz, ccu, icu, h0); // 计算命题公式的真值

if (jg == 0)

hq[h++] = bj; // 将真值为0的行数存入hq数组

else

xq[x++] = bj; // 将真值为1的行数存入xq数组

strcpy\_s(sz, sz0); // 恢复原始命题公式

for (i2 = 0; i2 < j; i2++)

std::cout << " " << icu[i2] << " "; // 输出当前行的真值

std::cout << " " << jg << "\n"; // 输出计算结果

}

if (hq[0] == -1)

std::cout << "\n该命题公式不存在主合取范式.\n"; // 输出结果

else

{

std::cout << "\n该命题公式的主合取范式:\n\t"; // 输出结果

for (i1 = 0; i1 < h; i1++)

{

if (i1 > 0)

std::cout << "\\/";

std::cout << "M(" << hq[i1] << ")";

}

}

if (xq[0] == -1)

std::cout << "\n该命题公式不存在主析取范式.\n"; // 输出结果

else

{

std::cout << "\n\n该命题公式的主析取范式:\n\t"; // 输出结果

for (i1 = 0; i1 < x; i1++)

{

if (i1 > 0)

std::cout << "/\\";

std::cout << "m(" << xq[i1] << ")";

}

}

std::cout << "\n"; // 输出结果

std::cout << "\n欢迎下次再次使用!\n"; // 输出结果

return 0;

int a[4]; // 存储逻辑运算结果

bool i, j; // 存储输入的逻辑值

while (true) { // 无限循环，直到用户选择退出

// 输出欢迎标语

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

cout << "\*\* \*\*\n";

cout << "\*\* 欢迎进入逻辑运算程序 \*\*\n";

cout << "\*\* \*\*\n";

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n";

// 输入逻辑值 i

cout << "请输入P的值（0或1）,以回车结束:" << endl;

while (true) {

cin >> i;

if (cin.fail() || (i != false && i != true)) { // 检查输入是否合法

std::cerr << "P的值输入有误,请重新输入!" << endl;

std::cin.clear();

std::cin.ignore(1234, '\n');

}

else

break;

}

// 输入逻辑值 j

cout << "请输入Q的值（0或1）,以回车结束:" << endl;

while (true) {

cin >> j;

if (cin.fail() || (j != false && j != true)) { // 检查输入是否合法

std::cerr << "Q的值输入有误,请重新输入!" << endl;

std::cin.clear();

std::cin.ignore(1234, '\n');

}

else

break;

}

// 进行逻辑运算

a[AND] = (i && j); // 与运算

a[OR] = (i || j); // 或运算

a[Implication] = ((!i) || j); // 蕴含运算，将其转化为与或非形式

a[Equality] = (((!i) || j) && ((!j) || i)); // 等值运算，将其转化为与或非形式

// 输出逻辑运算结果

printf("\n\n 合取:\n P/\\Q = %d\n", a[AND]); // 输出结果

printf(" 析取:\n P\\/Q = %d\n", a[OR]);

printf(" 条件:\n P->Q = %d\n", a[Implication]);

printf(" 双条件:\n P<->Q = %d\n", a[Equality]);

// 询问是否继续运算

cout << "\n是否继续运算?（y/n）";

char s;

while (true) {

s = \_getch();

if (s == 'y' || s == 'n') {

cout << s << endl;

Sleep(100);

break;

}

else

continue;

}

if (s == 'y')

{

system("cls"); // 清屏

continue; // 返回顶层

}

else

{

printf("欢迎下次再次使用!\n"); // 退出

exit(0);

}

}

return 0;

}