****

项目说明文档

**离散数学课程实验报告**

**——命题逻辑联接词、真值表、主范式**

**培养单位：软件学院**

**本 科 生：蓝 笙 聆**

**学 号：1951096**

**指导老师：唐 剑 锋**

二○二○年十二月

目录

[一、实验目的 1](#_Toc60056695)

[二、实验内容 1](#_Toc60056696)

[三、实验原理 1](#_Toc60056697)

[四、实验过程 3](#_Toc60056698)

[1、实验思路 3](#_Toc60056699)

[1、实验代码 5](#_Toc60056700)

[2、实验截屏 13](#_Toc60056701)

[五、实验小结 13](#_Toc60056702)

[1、解题思路 13](#_Toc60056703)

## 一、实验目的

本实验课程训练学生掌握命题逻辑中的联接词、真值表、主范式等，进一步能用它们来解决实际问题。通过实验提高学生编写实验报告、总结实验结果的能力；使学生具备程序设计的思想，能够独立完成简单的算法设计和分析。

## 二、实验内容

1. 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。（A）。

2. 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）。

## 三、实验原理

**（1）合取：**二元命题联结词。将两个命题P、Q联结起来，构成一个新的命题P∧Q, 读作P、Q的合取, 也可读作P与Q。这个新命题的真值与构成它的命题P、Q的真值间的关系为只有当两个命题变项P = T, Q = T时方可P∧Q =T, 而P、Q只要有一方为F则P∧Q = F。P∧Q可用来表示日常用语P与Q, 或P并且Q。

**（2）析取：**二元命题联结词。将两个命题P、Q联结起来，构成一个新的命题P∨Q, 读作P、Q的析取, 也可读作P或Q。这个新命题的真值与构成它的命题P、Q的真值间的关系为只有当两个命题变项P = F, Q = F时方可P∨Q =F, 而P、Q只要有一为T则P∨Q = T。P∨Q可用来表示日常用语P或者Q。

**（3）条件：**二元命题联结词。将两个命题P、Q联结起来，构成一个新的命题P→Q, 读作P条件Q, 也可读作如果P，那么Q。这个新命题的真值与构成它的命题P、Q的真值间的关系为只有当两个命题变项P = T, Q = F时方可P→Q =F, 其余均为T。

**（4）双向条件：**二元命题联结词。将两个命题P、Q联结起来，构成一个新的命题P←→Q, 读作P双条件于Q。这个新命题的真值与构成它的命题P、Q的真值间的关系为当两个命题变项P = T, Q =T时方可P←→Q =T, 其余均为F。

**（5）真值表：**表征逻辑事件输入和输出之间全部可能状态的表格。列出命题公式真假值的表。通常以1表示真，0 表示假。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决定，命题联结词的真值表给出了真假值的算法。 真值表是在逻辑中使用的一类数学表，用来确定一个表达式是否为真或有效。

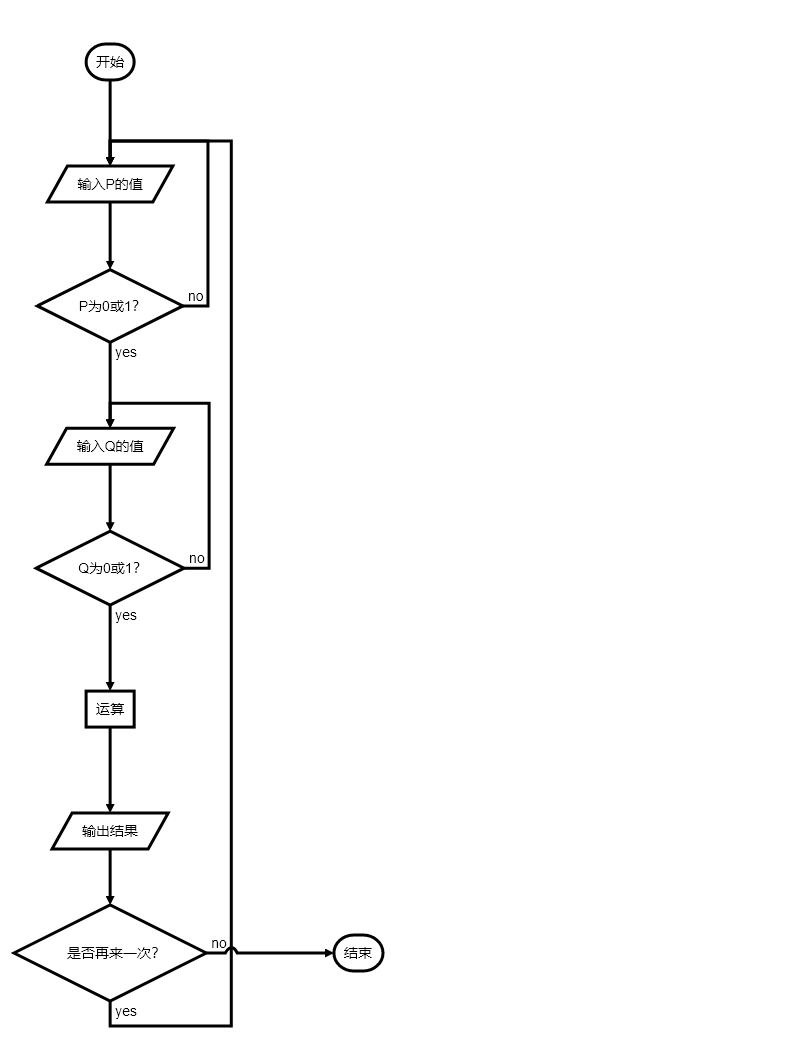
**（6）主范式：**

**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。

## 四、实验过程

### 1、实验思路

1. A题部分，首先是对各个输入量的处理，要确定输入的为0或1，否则则为出错，接下来就是运算处理。在C语言中本身支持的有与、或、非这三种，可以用!,&&,||来表示，而在这个实验中，不是与、或、非的可以通过转化变为与、或、非的形式。最后，枚举P和Q所有可能的取值并输出对应的结果。具体流程图如下：
2. B，C题部分，首先是输入一个合理的式子，然后从式子中查找出变量的个数，用来生成真值表，然后用函数运算，输出结果，并根据结果归类给范式，最后输出范式。

函数部分，主要是3个函数，一个为真值表递加函数，通过二进制的加法原理递增产生，一个为分级运算函数，这个函数是将中缀表达式转换成后缀表达式，并根据传入的二进制数枚举变量所有可能的取值，计算并得出结果。

开始

输入式子

计算变量个数

生成真值表

输出真值表

变量赋值

运算式子

输出结果

归类主范式

输出主范式

结束

循环是否结束

Y

N

### 1、实验代码

根据题意，写出实验代码如下所示：

#include <iostream>

#include <string>

using namespace std;

void opening() {

    cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"  //标语

         << "\*\*                                    \*\*\n"

         << "\*\*     Welcome to Logical Operator    \*\*\n"

         << "\*\*                                    \*\*\n"

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

}

bool loop() {

    int a[4];

    string i, j;

    char s;

    while (true) {

        cout << "Please enter a value for P(0/1): ";

        cin >> i;  //读取P的值

        if (i.size() != 1 || (i[0] != '0' && i[0] != '1')) {

            cout << "Error: Invalue input detected, please try again." << endl;

            continue;

        }

        break;

    }

    while (true) {

        cout << "Please enter a value for Q(0/1): ";

        cin >> j;  //读取Q的值

        if (j.size() != 1 || (j[0] != '0' && j[0] != '1')) {

            cout << "Error: Invalue input detected, please try again." << endl;

            continue;

        }

        break;

    }

    int p = i[0] - '0';

    int q = j[0] - '0';

    a[0] = p && q;     //与运算

    a[1] = p || q;     //或运算

    a[2] = (!p) || q;  //蕴含运算，将其转化为与或非形式

    a[3] = ((!p) || q) && ((!q) || p);  //等值运算，将其转化为与或非形式

    cout << "  P = " << i << "       Q = " << j << endl

         << endl                                             //输出结果

         << "Conjunction:\n       P/\\Q = " << a[0] << endl  //输出结果

         << "Disjunctive:\n       P\\/Q = " << a[1] << endl

         << "Conditional:\n       P->Q = " << a[2] << endl

         << "Biconditional:\n       P<->Q = " << a[3] << endl;

    cout << "Again? (y for yes, n for no)";  //询问是否继续操作

    cin >> s;

    if (s == 'y') return true;

    return false;

}

int main() {

    opening();

    while (loop())

        ;

    cout << "Thanks for using it. See you next time!" << endl;  //退出

    return 0;

}

#include <algorithm>

#include <cctype>

#include <cmath>

#include <iostream>

#include <list>

#include <sstream>

#include <stack>

#include <string>

#include <vector>

using namespace std;

double str2num(string s) {

    stringstream ss;

    double d = 0.0;

    ss << s;

    ss >> d;

    return d;

}

inline bool isVariable(char c) { return (c == '1' || (c >= 'a' && c <= 'z')); }

/\* 返回每个运算符对应的优先级 \*/

int priority(const char &ch) {

    switch (ch) {

        // 规定五个联结词优先级顺序是 ￢ ∧ ∨ -> <->

        // 等值式的优先级最低，我们设为1

        case '~':

            return 1;

        // 蕴含式的优先级其次，设为2

        case '^':

            return 2;

        // 或的优先级其次，设为3

        case '|':

            return 3;

        // 与的优先级其次，设为4

        case '&':

            return 4;

        // 非的优先级最高，设为5

        case '-':

            return 5;

        // 括号优先级另行处理，这里可以不写

        case '(':

        case ')':

            return 0;

        // 防止意外情况导致RE，default可写可不写

        default:

            return -1;

    }

}

/\* 将中缀表达式转换为后缀表达式 \*/

string toSuffix(const string &s) {

    string ret = "";                      // 用于返回的后缀表达式

    stack<char> st;                       // 用于实现转换的栈

    for (int i = 0; i < s.size(); i++) {  // 枚举每一位

        if (isVariable(s[i])) {  // 如果为数字或小数点，直接输出到结果中

            ret += s[i];

        } else if (s[i] == '(') {  // 如果为左括号，直接加入栈中

            st.push(s[i]);

        } else if (s[i] == ')') {  // 如果为右括号，

            while (st.top() != '(')  // 一直弹出并输出直到遇到左括号，

                ret += ' ', ret += st.top(), st.pop();  // 并弹出左括号

            st.pop();

        } else {  // 否则即为普通的运算符，

            ret += ' ';

            while (!st.empty() && priority(st.top()) >= priority(s[i]))

                ret += st.top(), ret += ' ',

                    st.pop();  // 一直弹出并输出直到遇到比当前优先级低的运算符，

            st.push(s[i]);  // 最后把当前运算符压入栈中

        }

    }

    while (!st.empty())  // 最后如果栈中还有剩余的字符，直接弹出并输出

        ret += ' ', ret += st.top(), st.pop();

    return ret;  // 最后返回结果

}

/\* 根据运算符计算相应的运算结果 \*/

int calcNum(const int &a, const int &b, const int &symbol) {

    switch (symbol) {

        case '&':

            return a & b;

        case '|':

            return a | b;

        case '-':

            return a - b;

        case '^':

            return (1 - a) | b;

        case '~':

            return 1 - (a ^ b);

        default:  // 与上面同理，default可写可不写

            return -1;

    }

}

/\* 计算后缀表达式并输出计算过程 \*/

int calc(string s, vector<char> v, vector<int> num) {

    /\*\*

     \* 实现计算的数字栈，为了输出方便(不需要将每个元素弹出)

     \* 这里使用vector

     \*/

    while (num.size() != v.size()) num.push\_back(0);

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

        for (auto &&j : s)

            if (j == v[v.size() - i - 1]) j = num[i] + '0';

    vector<int> st;

    // cout << s << endl;  // 先输出后缀表达式

    string str;

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

        if (i != s.size() && s[i] != ' ')

            str += s[i];

        else {

            int temp = -1;

            if (isdigit(str[0])) {  // 如果遇到数字直接压入栈

                temp = str2num(str);

                st.push\_back(temp);

            } else {       // 否则计算结果并压入栈

                int a, b;  //取出栈顶的两个元素

                a = st.back();

                st.pop\_back();

                b = st.back();

                st.pop\_back();

                st.push\_back(calcNum(b, a, str[0]));  // 注意计算顺序

                // 每计算一次输出一次计算过程

                /\*

                for (auto &&i : st) cout << i << ' ';  // 输出栈中已计算过的数

                for (int j = i + 1; j < s.size(); j++)

                    cout << s[j];  // 再输出还未计算过的数

                cout << endl;

                \*/

            }

            str = "";

        }

    }

    return st[0];

}

bool effective(string s) {

    if (s.size() == 0 || (s.size() == 1 && !isVariable(s[0]))) return false;

    int brackets = 0;

    if (s[0] == '(') brackets++;

    for (int i = 1; i < s.size(); i++) {

        if (s[i] == '(')

            brackets++;

        else if (s[i] == ')')

            brackets--;

        if (brackets < 0) return false;

        if (!isVariable(s[i]) && !isVariable(s[i - 1])) {

            if (s[i] != '!' && s[i] != '(' && s[i - 1] != ')') return false;

        }

    }

    if (brackets != 0) return false;

    return true;

}

bool loop() {

    string s;

    string ss = "";

    vector<char> v;

    cout << "Please enter an expression. " << endl;

    cin >> s;

    while (!effective(s)) {

        cout << "An invalid expression was detected. Please try again." << endl;

        cin >> s;

    }

    for (auto &&i : s) {

        bool fflag = true;

        if (isVariable(i)) {

            for (auto &&j : v)

                if (i == j) {

                    fflag = false;

                    break;

                }

            if (fflag) v.push\_back(i);

        }

    }

    sort(v.begin(), v.end());

    int flag = 0;

    for (size\_t i = 0; i < s.size(); i++) {

        if ((s[i] == '!') &&

            (i == 0 || (s[i - 1] != ')' && !isVariable(s[i - 1])))) {

            ss += '(';

            ss += '1';

            ss += '-';

            flag++;

            continue;

        }

        ss += s[i];

        if (i != s.size() - 1 && isVariable(s[i + 1])) continue;

        for (size\_t i = 0; i < flag; i++) ss += ')';

        flag = 0;

    }

    auto sss = toSuffix(ss);

    int len = pow(2, v.size());

    vector<int> mdisjunctive;  //主析取范式

    vector<int> mconjunctive;  //主合取范式

    vector<int> num(v.size());

    for (auto &&i : v) cout << i << "    ";

    cout << s << endl;

    for (int i = 0; i < len; i++) {

        int n = i;

        int j = 0;

        while (n >= 2) {

            num[j] = n % 2;

            n /= 2;

            j++;

        }

        num[j] = n;

        j++;

        for (j; j < v.size(); j++) num[j] = 0;

        int temp = calc(sss, v, num);

        for (auto &&k : num) cout << k << "    ";

        cout << temp << endl;

        if (temp == 1)

            mdisjunctive.push\_back(i);

        else

            mconjunctive.push\_back(i);

    }

    if (mconjunctive.size() == 0)

        cout << "There is no principal conjunctive normal form in this proposition formula." << endl;

    else {

        cout << "The main conjunctive normal form of the proposition formula is as follows:" << endl;

        cout << "    M(" << mconjunctive[0] << ")";

        for (int i = 1; i < mconjunctive.size(); i++)

            cout << "\\/M(" << mconjunctive[i] << ")";

        cout << endl;

    }

    if (mdisjunctive.size() == 0)

        cout << "There is no principal disjunctive normal form in the propositional formula." << endl;

    else {

        cout << "The main disjunctive normal form of the proposition formula is as follows:" << endl;

        cout << "    m(" << mdisjunctive[0] << ")";

        for (int i = 1; i < mdisjunctive.size(); i++)

            cout << "/\\m(" << mdisjunctive[i] << ")";

        cout << endl;

    }

    cout << "Again? (y for yes, n for no)";

    char c;

    cin >> c;

    if (c == 'y')

        return true;

    else

        return false;

}

int main() {

    while (loop())

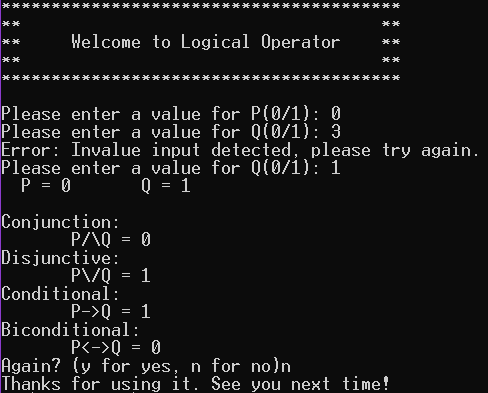
        ;

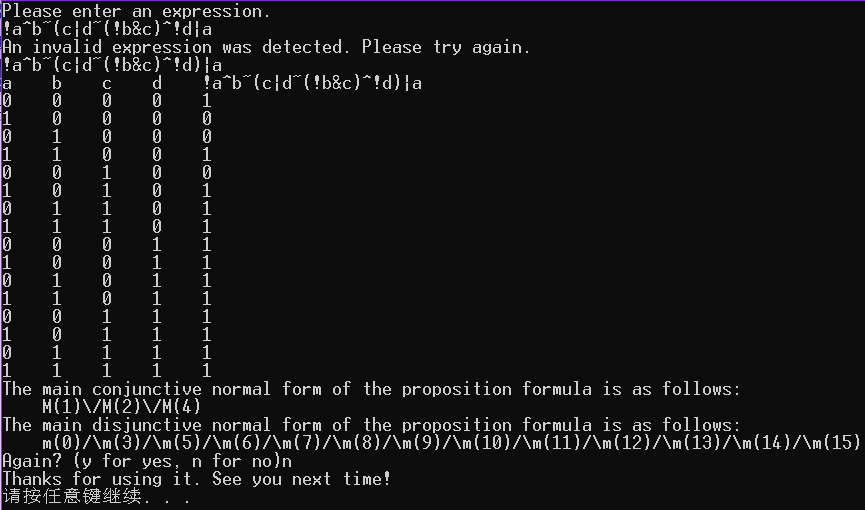
    std::cout << "Thanks for using it. See you next time! " << std::endl;

    return 0;

}

### 2、实验截屏





## 五、实验小结

1、掌握了命题逻辑中的联接词、真值表、主范式等，能运用它们解决问题。知道了如何求逻辑表达式的主合取范式和主析取范式。