《离散数学》课程实验报告

1 命题逻辑联接词、真值表、主范式

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。

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）。

详细说明：

1. 逻辑联接词的运算

本实验要求利用C/C＋＋语言，实现二元合取、析取、条件和双向条件表达式的计算。充分利用联接词和逻辑运算符之间的相似性来实现程序功能。

2. 求任意一个命题公式的真值表

本实验要求利用C/C＋＋语言，实现任意输入公式的真值表计算。一般将公式中的命题变元放在真值表的左边，将公式的结果放在真值表的右边。命题变元可用数值变量表示，合式公式的表示及求真值表转化为逻辑运算结果；可用一维数表示合式公式中所出现的n个命题变元，同时它也是一个二进制加法器的模拟器，每当在这个模拟器中产生一个二进制数时，就相当于给各个命题变元产生了一组真值指派。算法逻辑如下：

（1）将二进制加法模拟器赋初值0。

（2）计算模拟器中所对应的一组真值指派下合式公式的真值。

（3）输出真值表中对应于模拟器所给出的一组真值指派及这组真值指派所对应的一行真值。

（4）产生下一个二进制数值，若该数值等于2n-1，则结束，否则转（2）。

实验环境

本程序采用c++语言，使用VSCode作为编译器，g++进行编译。

实验原理和实现过程（算法描述）

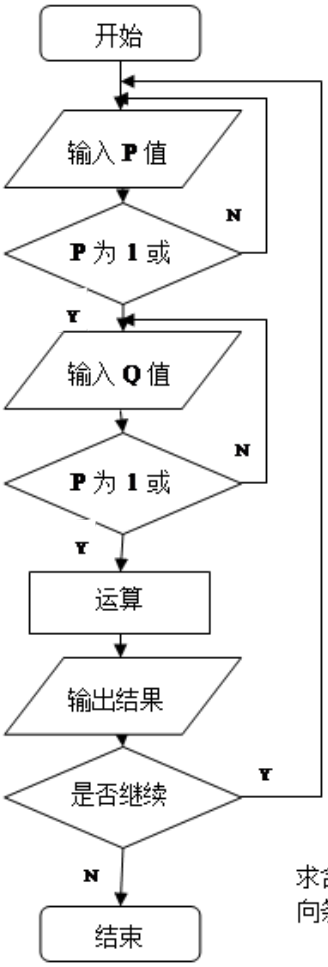
实验原理

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。

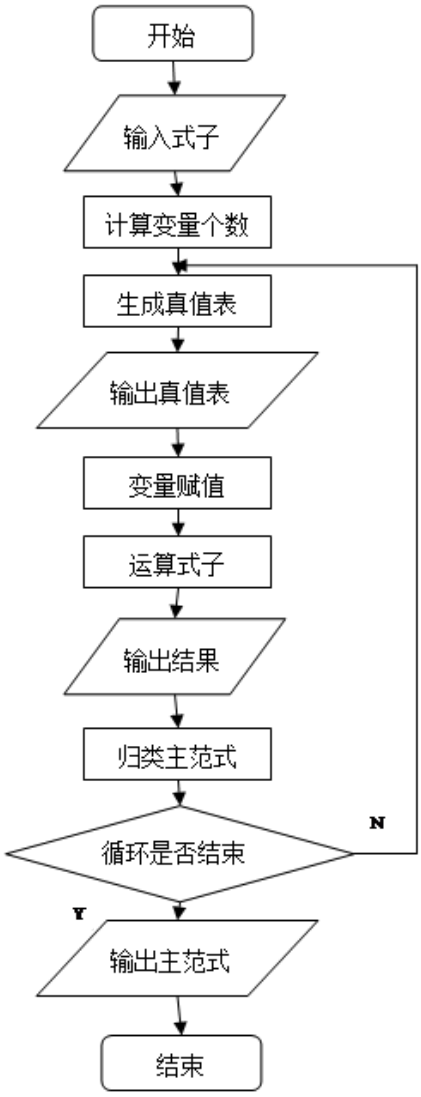
实验过程

（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，具体流程图如下：

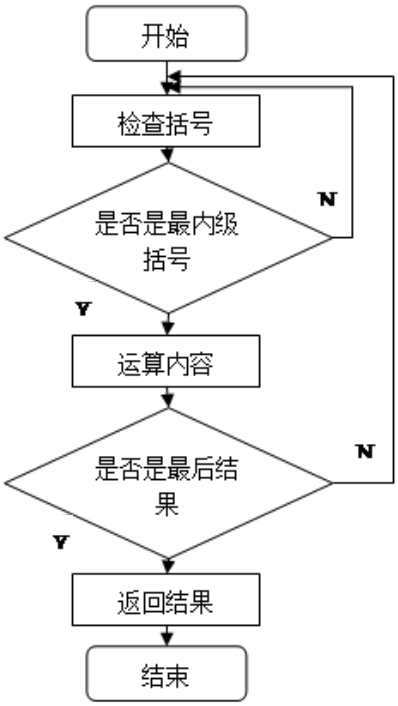


求合取、析取、条件和双向条件的真值流程图

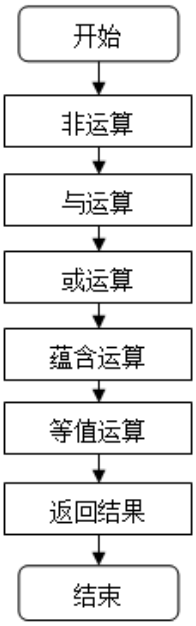
（2）B，C题部分，首先是输入一个合理的式子，然后从式子中查找出变量的个数，开辟一个二进制函数，用来生成真值表，然后用函数运算，输出结果，并根据结果归类给范式，最后输出范式。
函数部分，主要是3个函数，一个为真值表递加函数，通过二进制的加法原理递进产生，一个为分级运算函数，这个函数是通过判断括号，选出最内级括号的内容执行运算函数，这样一级级向外运算，最后得出最终结果，剩下一个为主运算函数，按照运算符号的优先级按顺序进行运算。


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
循环是否结束
N
Y
输出主范式
结束

主函数



分级运算函数



主运算函数

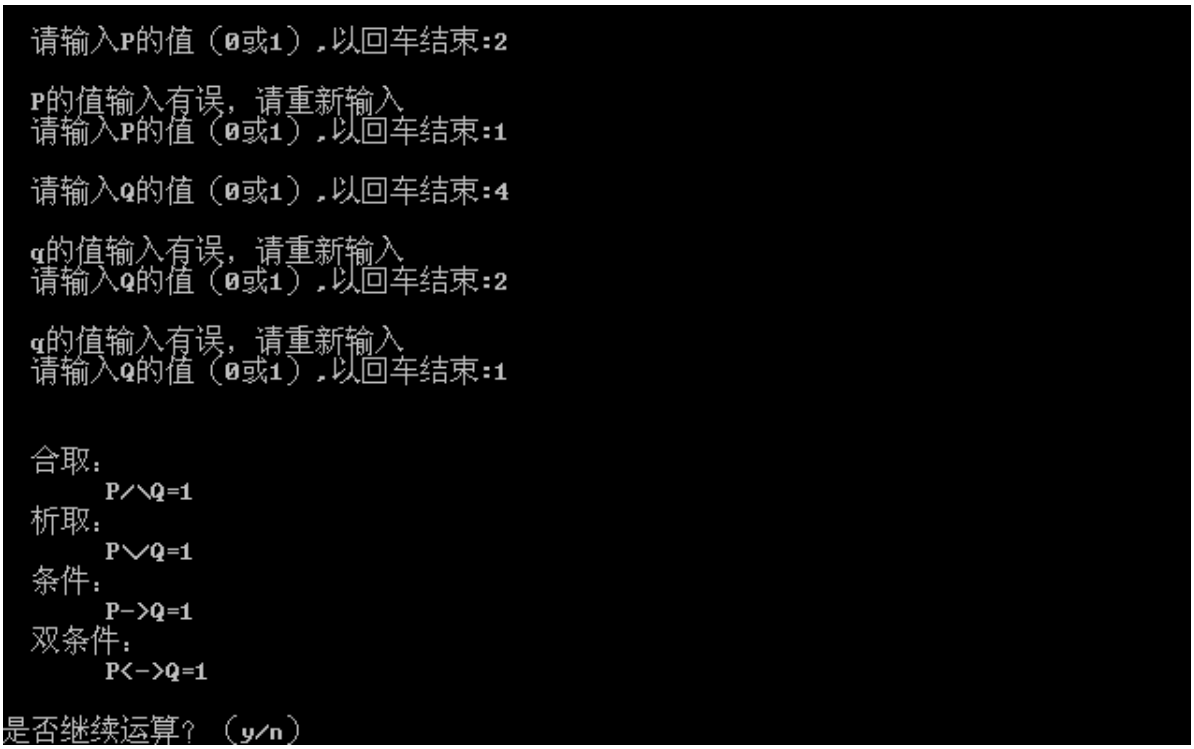
实验数据及结果分析

题目A:

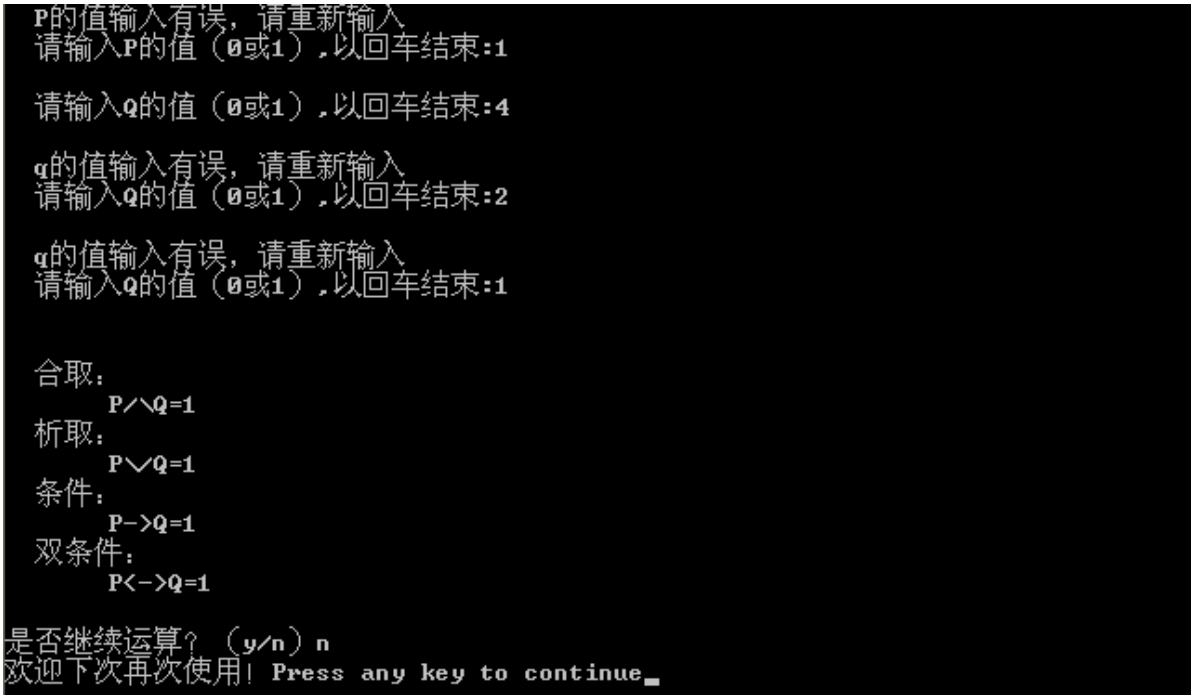
进入界面:

正确运算结果:

错误控制和输入正确后:



退出:



结果分析

这道题主要是读取数值并进行计算，输出两个命题变元(P和Q)的合取，析取，条件和双向条件的真值结果，同时要注意输入的值要必须0或1，如果不是，则进行错误提示，并进行重新输入。

B,C题：

欢迎界面：

输出非运算公式的真值表，主合取范式和主析取范式：

输出与运算公式的真值表，主合取范式和主析取范式：

输出或运算公式的真值表，主合取范式和主析取范式：

```
**                                      **
****************************************

请输入一个合法的命题公式:
a!b

d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

该命题公式的主合取范式:
        M(0)

该命题公式的主析取范式:
        m(1)/\m(2)/\m(3)

欢迎下次再次使用!
```

输出蕴含运算公式的真值表，主合取范式和主析取范式：

```
**                                      **
****************************************

请输入一个合法的命题公式:
a^b

d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

该命题公式的主合取范式:
        M(2)

该命题公式的主析取范式:
        m(0)/\m(1)/\m(3)

欢迎下次再次使用!
```

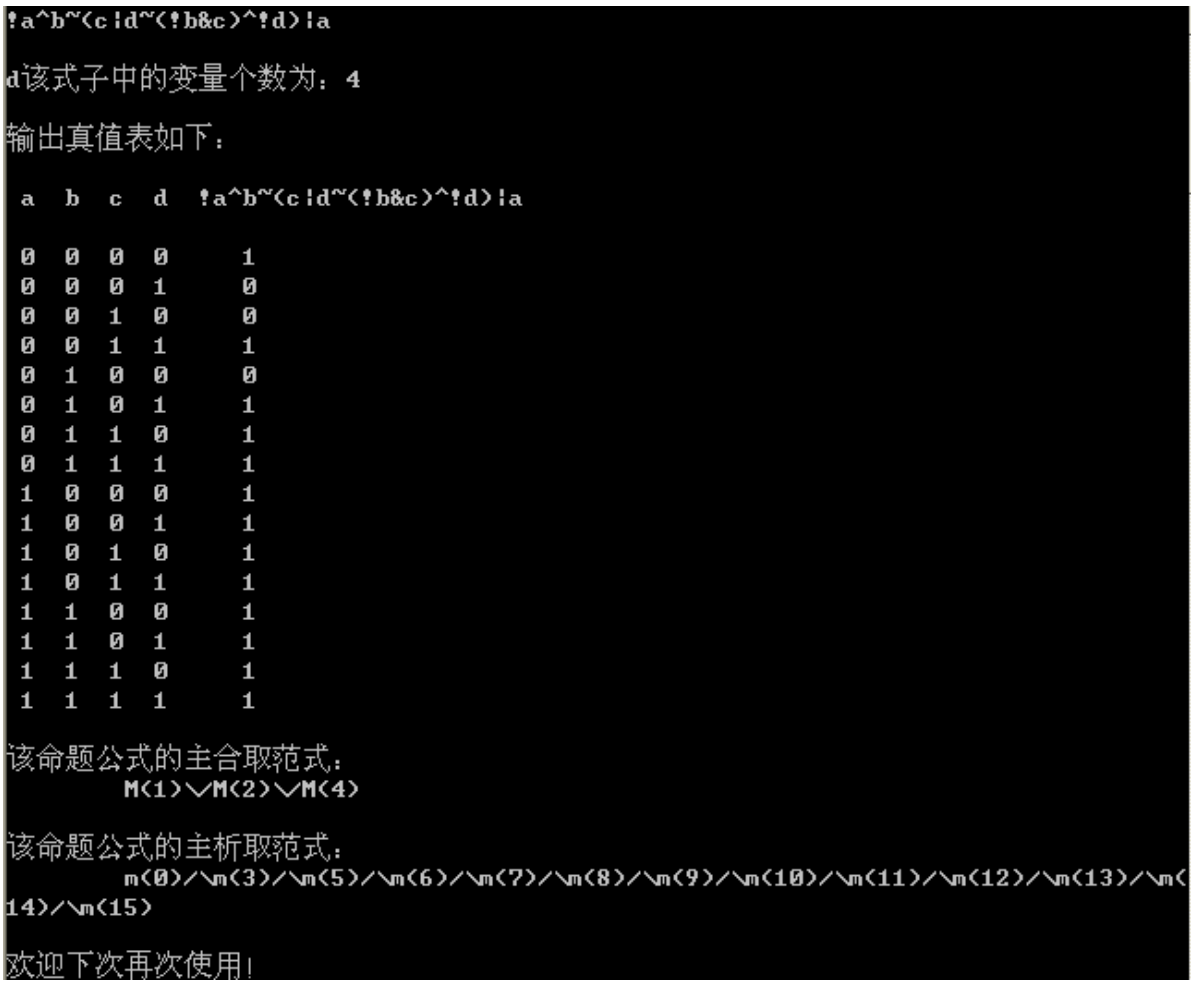
输出等值运算公式的真值表，主合取范式和主析取范式：

```
**                                      **
****************************************

请输入一个合法的命题公式:
a^b

d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

该命题公式的主合取范式:
        M(1)∨M(2)

该命题公式的主析取范式:
        m(0)∧m(3)

欢迎下次再次使用!
```

输出综合运算公式的真值表，主合取范式和主析取范式：

```
a^b~c¦b&a^c

d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

该命题公式的主合取范式:
        M(4)∨M(5)∨M(6)

该命题公式的主析取范式:
        m(0)∧m(1)∧m(2)∧m(3)∧m(7)

欢迎下次再次使用!
```

输出带括号的综合运算运算公式的真值表，主合取范式和主析取范式：



结果分析

B，C题目由于本身关系比较密切，所以将两个做在了一起。这个程序达到了题目要求的各个功能，可以运算与，或，非，蕴含，等值条件组成的表达式，并且支持括号运算。

C++源程序清单

A题部分源代码：

```
#include <iostream>
using namespace std;

int main() {
  int a[4];
  int i = -1, j = -1;
  char s;
  cout << "****************************************\n";  //标语
  cout << "**                                    **\n";
  cout << "**         欢迎进入逻辑运算程序         **\n";
  cout << "**                                    **\n";
  cout << "****************************************\n\n";
  while (true) {
    cout << "\n  请输入P的值（0或1）,以回车结束:";
    cin >> i;  //读取P的值
    if (i != 0 && i != 1) {
      cout << "\n  P的值输入有误,请重新输入!";
      continue;
    }
    cout << "\n  请输入Q的值（0或1）,以回车结束:";
```

```
    cin >> j;  //读取Q的值
    if (j != 0 && j != 1) {
      cout << "\n  Q的值输入有误,请重新输入!";
      continue;
    }
    a[0] = i && j;      //与运算
    a[1] = i || j;      //或运算
    a[2] = (!i) || j;  //蕴含运算，将其转化为与或非形式
    a[3] = ((!i) || j) && ((!j) || i);  //等值运算，将其转化为与或非形式
    cout << "\n\n  合取:\n       P/\\Q = " << a[0] << endl;  //输出结果
    cout << "  析取:\n       P\\/Q = " << a[1] << endl;
    cout << "  条件:\n       P->Q = " << a[2] << endl;
    cout << "  双条件:\n       P<->Q = " << a[3] << endl;
    cout << "\n是否继续运算?（y/n）";  //询问是否继续操作
    cin >> s;
    while (true) {
      if (s == 'y' || s == 'n') {
        if (s == 'y') {
          break;
        } else {
          printf("欢迎下次再次使用!\n");  //退出
          exit(0);
        }
      } else {
        printf("输入错误,请重新输入!\n");  //错误校验
        cin >> s;
      }
    }
  }
  return 0;
}
```

B，C题部分源代码：

```
#include <cmath>
#include <iostream>
#include <string.h>

#define N 50

using namespace std;

void panduan(int b[N], int f);                          //赋值函数
int tkh(char sz[N], char ccu[N], int icu[N], int h0);  //分级运算函数
int fkh(char sz[N], char ccu[N], int icu[N], int h0);  //主运算函数

int main() {
  int i1, i2, d = 1, icu[N], kh = 0, jg, j = 0,
              h0;  // icu[N]用于存放变量值,kh用于括号计数,jg存放结果
  int bj = 0, hq[N], h = 0, x = 0, xq[N];  // hq[N]存放合取结果,xq[N]存放析取结果
  char sz[N], ccu[N],
      sz0[N];  // sz[N]存放式子,ccu[N]存放变量,sz0[N]也是用于存放式子
  hq[0] = -1;
  xq[0] = -1;
  cout << "**************************************\n";  //标语
  cout << "**                                  **\n";
```

```
cout << "**          欢迎进入逻辑运算软件       **\n";
cout << "**    (可运算真值表,主范式,支持括号)  **\n";
cout << "**                                     **\n";
cout << "**               用!表示非             **\n";
cout << "**               用&表示与             **\n";
cout << "**               用|表示或             **\n";
cout << "**              用^表示蕴含            **\n";
cout << "**              用~表示等值            **\n";
cout << "**                                     **\n";
cout << "*****************************************\n\n";
cout << "请输入一个合法的命题公式:\n";  //输入式子
cin >> sz;                                  //读取式子
strcpy(sz0, sz);                            //复制式子
for (unsigned i1 = 0; i1 < strlen(sz); i1++) {
  if (sz[i1] == ')' || sz[i1] == '(')  //存储括号数量
    kh++;
  if (sz[i1] >= 'a' && sz[i1] <= 'z' || sz[i1] >= 'A' && sz[i1] <= 'Z') {
    for (i2 = 0; i2 < j; i2++)  //判断并储存变量。
      if (ccu[i2] == sz[i1])    //去除重复变量
        d = 0;
    if (d == 1) {
      ccu[j] = sz[i1];
      j++;
    }
    d = 1;
  }
}
cout << "\n该式子中的变量个数为:" << j << endl;  //输出变量个数
h0 = j;
if (j == 1 && strlen(sz) == 1) {
  cout << "\n输出真值表如下:\n \n";  //输出真值表表头
  for (i1 = 0; i1 < h0; i1++) cout << ccu[i1] << " ";
  cout << "    ";
  cout << sz;
  cout << "\n";
  cout<<"0     0"<<endl;
  cout<<"1     1"<<endl;
  cout << "\n该命题公式的主合取范式:\n\t";
  cout << "M(" << 0 << ")";
  cout << "\n\n该命题公式的主析取范式:\n\t";
  cout << "m(" << 1 << ")";  //输出主析取范式
  return 0;
}
cout << "\n输出真值表如下:\n \n";  //输出真值表表头
for (i1 = 0; i1 < h0; i1++) cout << ccu[i1] << " ";
cout << "    ";
cout << sz;
cout << "\n";
for (i1 = 0; i1 < j; i1++)  //先将所有的变量赋值为零。
  icu[i1] = 0;
for (i2 = 0; i2 < j; i2++)  //输出真值表前项
  cout << icu[i2] << " ";
jg = tkh(sz, ccu, icu, h0);  //用函数求结果
if (jg == 0)                 //结果为0，合取加1
  hq[h++] = bj;
else  //否则，析取加1
  xq[x++] = bj;
cout << "    " << jg << endl;  //输出运算结果
```

```
  strcpy(sz, sz0);
  for (i1 = 0; i1 < (int)pow(2.0, j) - 1; i1++) {
    ++bj;
    panduan(icu, j - 1);  //赋值变量
    jg = tkh(sz, ccu, icu, h0);
    if (jg == 0)  //结果为0，合取加1
      hq[h++] = bj;
    else  //否则，析取加1
      xq[x++] = bj;
    strcpy(sz, sz0);  //恢复被修改的数组
    for (i2 = 0; i2 < j; i2++) cout << icu[i2] << " ";  //输出真值表前项
    cout << "    " << jg << endl;                       //输出运算结果
  }
  if (hq[0] == -1)  //不存在合取范式时
    cout << "\n该命题公式不存在主合取范式.\n";
  else {
    cout << "\n该命题公式的主合取范式:\n\t";
    for (i1 = 0; i1 < h; i1++) {
      if (i1 > 0)  //判断并添加符号
        cout << "\\/";
      cout << "M(" << hq[i1] << ")";  //输出主合取范式
    }
  }
  if (xq[0] == -1)  //不存在析取范式时
    cout << "\n该命题公式不存在主析取范式.\n";
  else {
    cout << "\n\n该命题公式的主析取范式:\n\t";
    for (i1 = 0; i1 < x; i1++) {
      if (i1 > 0)  //判断并添加符号
        cout << "/\\";
      cout << "m(" << xq[i1] << ")";  //输出主析取范式
    }
  }
  cout << "\n";
  cout << "\n欢迎下次再次使用!\n";  //结束
  return 0;
}

void panduan(int b[N], int f)  //二进制赋值
{
  int i;
  i = f;
  if (b[f] == 0)  //加1
    b[f] = 1;
  else  //进位
  {
    b[f] = 0;
    panduan(b, --i);
  }
}

int tkh(char sz[N], char ccu[N], int icu[N], int h0)  //分级运算函数
{
  int i, j, h, s, kh = 0, wz[N], a;
  char xs1[N], ckh[N];  // xs1用来保存括号内的字符 ckh用来保存括号。
  s = strlen(sz);
  for (i = 0; i < s; i++)
    if (sz[i] == '(' || sz[i] == ')')  //判断括号
```

```
    {
      wz[kh] = i;        //存储括号位置
      ckh[kh] = sz[i];  //存储括号类型
      kh++;
    }
  if (kh == 0)
    return fkh(sz, ccu, icu, h0);  //如果无括号，直接运行
  else {
    for (i = 0; i < kh; i++)
      if (ckh[i] == ')')  //找到第一个）
        break;
    for (j = wz[i - 1] + 1, h = 0; j < wz[i];
         j++, h++)  //存储最内级括号中的内容
      xs1[h] = sz[j];
    xs1[h] = '\0';
    a = fkh(xs1, ccu, icu, h0);  //运行最内级括号的式子，得到结果
    if (a == 1)                  //判断并存储结果
      sz[wz[i - 1]] = 1;
    else
      sz[wz[i - 1]] = -2;
    for (j = wz[i - 1] + 1; j < s + wz[i - 1] - wz[i]; j++)  //将括号后内容前移
      sz[j] = sz[j + wz[i] - wz[i - 1]];
    sz[j] = '\0';
    return tkh(sz, ccu, icu, h0);  //循环执行
  }
}

int fkh(char sz[N], char ccu[N], int icu[N], int h0)  //主运算函数
{
  int i, h = 0, j = 0, j1 = 0, j2 = 0, j3 = 0, j4 = 0, j5 = 0, i1, i2, p1 = -1,
         p2 = -1, s;
  char dt[N];
  s = strlen(sz);
  if (s == 1)
    if (sz[0] == -2)  //判断是否是最后一项
      return 0;
    else
      return 1;  // 1就是sz[0]的值
  else {
    for (i = 0; i < s - j; i++)  //先处理非
      if (sz[i] == '!') {
        for (i1 = 0; i1 < h0; i1++)
          if (sz[i + 1] == ccu[i1])  //将变量赋值并给P1
            p1 = icu[i1];
        if (sz[i + 1] == -2)  //如果是前运算结果的0，则P1等于0
          p1 = 0;
        if (p1 == -1)  //如果是数字，直接给P1
          p1 = sz[i + 1];
        dt[j + 2] = !p1;  //非运算
        sz[i] = j + 2;
        j++;
        p1 = 0;
        for (i1 = i + 1; i1 < s - j; i1++)
          sz[i1] = sz[i1 + 1];  //将后续式子前移一项
      }
    p1 = -1;
    j1 = j;
    for (i = 0; i < s - j1 - 2 * j2; i++)  //处理与
```

```
      if (sz[i] == '&') {
        for (i1 = 0; i1 < h0; i1++) {
          if (sz[i - 1] == ccu[i1])  //将变量赋值并给P1
            p1 = icu[i1];
          if (sz[i + 1] == ccu[i1])  //将变量赋值并给P2
            p2 = icu[i1];
        }
        for (i2 = 2; i2 < j + 2; i2++) {
          if (sz[i - 1] == i2)  //如果为前计算结果，将结果赋值并给P1
            p1 = dt[i2];
          if (sz[i + 1] == i2)  //如果为前计算结果，将结果赋值并给P2
            p2 = dt[i2];
        }
        if (sz[i - 1] == -2)  //如果是前运算结果的0，则P1等于0
          p1 = 0;
        if (sz[i + 1] == -2)  //如果是前运算结果的0，则P2等于0
          p2 = 0;
        if (p1 == -1)  //如果是数字，直接给P1
          p1 = (int)(sz[i - 1]);
        if (p2 == -1)  //如果是数字，直接给P2
          p2 = (int)(sz[i + 1]);
        dt[j + 2] = p1 && p2;  //与运算
        sz[i - 1] = j + 2;
        j++;
        j2++;
        p1 = -1;
        p2 = -1;
        for (i1 = i; i1 < s - j1 - 2 * j2; i1++)  //将后续式子前移两项
          sz[i1] = sz[i1 + 2];
        i = i - 1;
      }
    for (i = 0; i < s - j1 - 2 * j2 - 2 * j3; i++)  //处理或
      if (sz[i] == '|') {
        for (i1 = 0; i1 < h0; i1++) {
          if (sz[i - 1] == ccu[i1])  //将变量赋值并给P1
            p1 = icu[i1];
          if (sz[i + 1] == ccu[i1])  //将变量赋值并给P2
            p2 = icu[i1];
        }
        for (i2 = 2; i2 < j + 2; i2++) {
          if (sz[i - 1] == i2)  //如果为前计算结果，将结果赋值并给P1
            p1 = dt[i2];
          if (sz[i + 1] == i2)  //如果为前计算结果，将结果赋值并给P2
            p2 = dt[i2];
        }
        if (sz[i - 1] == -2)  //如果是前运算结果的0，则P1等于0
          p1 = 0;
        if (sz[i + 1] == -2)  //如果是前运算结果的0，则P2等于0
          p2 = 0;
        if (p1 == -1)  //如果是数字，直接给P1
          p1 = sz[i - 1];
        if (p2 == -1)  //如果是数字，直接给P2
          p2 = sz[i + 1];
        dt[j + 2] = p1 || p2;  //或运算
        sz[i - 1] = j + 2;
        j++;
        j3++;
        p1 = -1;
```

```
        p2 = -1;
        for (i1 = i; i1 < s - j1 - 2 * j2 - 2 * j3; i1++)  //将后续式子前移两项
          sz[i1] = sz[i1 + 2];
        i--;
      }
    for (i = 0; i < s - j1 - 2 * j2 - 2 * j3 - 2 * j4; i++)  //处理蕴含
      if (sz[i] == '^') {
        for (i1 = 0; i1 < h0; i1++) {
          if (sz[i - 1] == ccu[i1])  //将变量赋值并给P1
            p1 = icu[i1];
          if (sz[i + 1] == ccu[i1])  //将变量赋值并给P2
            p2 = icu[i1];
        }
        for (i2 = 2; i2 < j + 2; i2++) {
          if (sz[i - 1] == i2)  //如果为前计算结果，将结果赋值并给P1
            p1 = dt[i2];
          if (sz[i + 1] == i2)  //如果为前计算结果，将结果赋值并给P2
            p2 = dt[i2];
        }
        if (sz[i - 1] == -2)  //如果是前运算结果的0，则P1等于0
          p1 = 0;
        if (sz[i + 1] == -2)  //如果是前运算结果的0，则P2等于0
          p2 = 0;
        if (p1 == -1)  //如果是数字，直接给P1
          p1 = sz[i - 1];
        if (p2 == -1)  //如果是数字，直接给P2
          p2 = sz[i + 1];
        dt[j + 2] = !p1 || p2;  //蕴含运算
        sz[i - 1] = j + 2;
        j++;
        j4++;
        p1 = -1;
        p2 = -1;
        for (i1 = i; i1 < s - j1 - 2 * j2 - 2 * j3 - 2 * j4;
             i1++)  //将后续式子前移两项
          sz[i1] = sz[i1 + 2];
        i--;
      }
    for (i = 0; i < s - j1 - 2 * j2 - 2 * j3 - 2 * j4 - 2 * j5; i++)  //处理等值
      if (sz[i] == '~') {
        for (i1 = 0; i1 < h0; i1++) {
          if (sz[i - 1] == ccu[i1])  //将变量赋值并给P1
            p1 = icu[i1];
          if (sz[i + 1] == ccu[i1])  //将变量赋值并给P2
            p2 = icu[i1];
        }
        for (i2 = 2; i2 < j + 2; i2++) {
          if (sz[i - 1] == i2)  //如果为前计算结果，将结果赋值并给P1
            p1 = dt[i2];
          if (sz[i + 1] == i2)  //如果为前计算结果，将结果赋值并给P2
            p2 = dt[i2];
        }
        if (sz[i - 1] == -2)  //如果是前运算结果的0，则P1等于0
          p1 = 0;
        if (sz[i + 1] == -2)  //如果是前运算结果的0，则P2等于0
          p2 = 0;
        if (p1 == -1)  //如果是数字，直接给P1
          p1 = sz[i - 1];
```

```
        if (p2 == -1)  //如果是数字，直接给P2
          p2 = sz[i + 1];
        dt[j + 2] = (!p1 || p2) && (!p2 || p1);  //等值运算
        sz[i - 1] = j + 2;
        j++;
        j5++;
        p1 = -1;
        p2 = -1;
        for (i1 = i; i1 < s - j1 - 2 * j2 - 2 * j3 - 2 * j4 - 2 * j5;
             i1++)  //将后续式子前移两项
          sz[i1] = sz[i1 + 2];
        i--;
      }
    return dt[j + 1];  //返回结果
  }
}
```