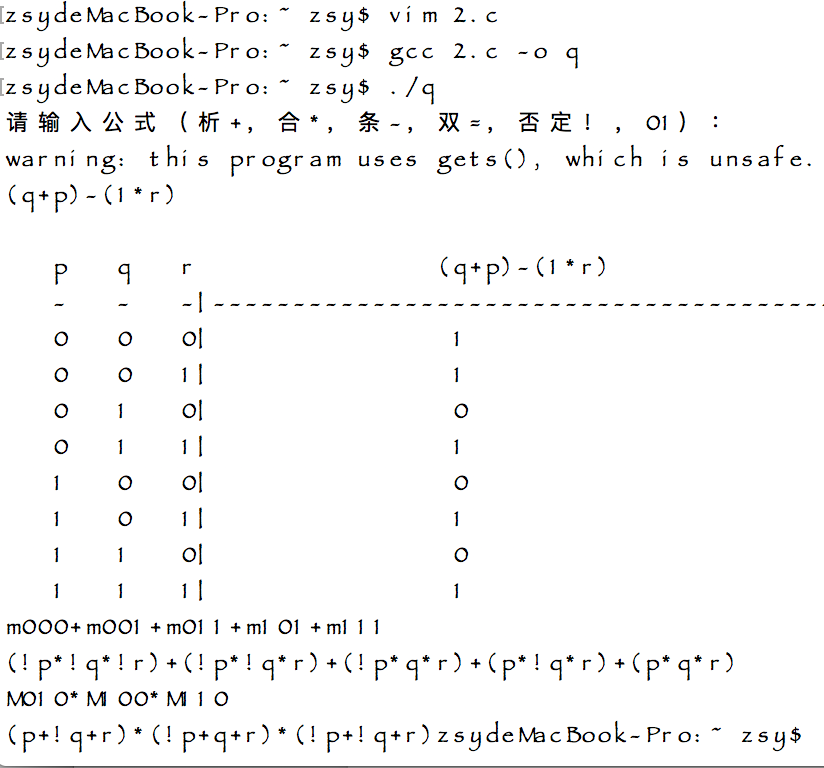
w4真值表生成代码

1. 运行结果：



1. 代码改进与注释（红字为此段伪代码解释，红字加下划线为修改过的代码解释，//黑字为单句解释）

#include<string.h>

#include<stdio.h>

int getalpha(char a[],char b[])

{

char tmpc=' ';

int n=strlen(a),i=0,j=0,k=0;

for(i=0;i<n;i++)//是字母

{

if(((a[i]>='a')&&(a[i]<='z'))||((a[i]>='A')&&(a[i]<='Z')))

{

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

{

if(b[k]==a[i])

break;

}

if(k>=j)

{

b[j]=a[i];

j++;

}

}

}

//从低到高排序

for(i=0;i<j-1;i++)

{

for(k=0;k<j-i-1;k++)

{

if(b[k]>b[k+1])

{

tmpc=b[k];

b[k]=b[k+1];

b[k+1]=tmpc;

}

}

}

b[j]='\0';//加上结束的标志

return j;//小写字母的个数

}

void fillValue(char a[],char varchar[],int nvar,char valchar[],char resultchar[])

{

//a是原始公式，varchar是变元列表如pqrs，

//valchar是变元的某次取值

int nLen=strlen(a),i=0,j=0,k=0;

for(i=0;i<nLen;i++)

resultchar[i]=a[i];

resultchar[i]='\0';

for(i=0;i<nLen;i++)

{

//原公式中的每个字符

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

{

//公式中的字符是第几个变元

if(resultchar[i]==varchar[j])

{

//是第j变元，其值换成第j个值

resultchar[i]=valchar[j];

break;

}

}

}

}

void negatecal(char a[])

{

int \_result=0,i=0,j=0;

while(i<strlen(a))

{

//如果当前位置起形如"!1"则换成"0"

j=i;

\_result=0;

if((j+1<strlen(a))&&(a[j]=='!')&&(a[j+1]=='1'))

{

a[j]='0';

\_result=1;

}

else if((j+1<strlen(a))&&(a[j]=='!')&&(a[j+1]=='0'))

{

a[j]='1';

\_result=1;

}

else if((j+2<strlen(a))&&(a[j]=='(')&&(a[j+1]=='1')&&(a[j+2]==')'))

{

a[j]='1';

\_result=2;

}

这里有修改：我的result记录的是我修改的个数，也就是到时候需要前移的位数。

else if((j+2<strlen(a))&&(a[j]=='(')&&(a[j+1]=='0')&&(a[j+2]==')'))

{

a[j]='0';

\_result=2;

}

if(\_result!=0)//如果有运算则后面的往前移

不用分着写移动1个或2个，根据result的值来移位。

{

j++;

while(a[j+\_result]!='\0')//后面的字符串往前移1格

{

a[j]=a[j+\_result];

j++;

}

a[j]='\0';

}

else//没有！0或！1则看下一个指针

i++;

}

}

void conYsh(char a[],int j,int r)

{

int \_result=0,i=j;

while(i<r)

{

//\*

j=i;

\_result=0;

if((j+2<strlen(a))&&(a[j]=='0')&&(a[j+1]=='\*')&&(a[j+2]=='0'))

{

a[j]='0';

\_result=1;

}

else if((j+2<strlen(a))&&(a[j]=='0')&&(a[j+1]=='\*')&&(a[j+2]=='1'))

{

a[j]='0';

\_result=1;

}

else if((j+2<strlen(a))&&(a[j]=='1')&&(a[j+1]=='\*')&&(a[j+2]=='0'))

{

a[j]='0';

\_result=1;

}

else if((j+2<strlen(a))&&(a[j]=='1')&&(a[j+1]=='\*')&&(a[j+2]=='1'))

{

a[j]='1';

\_result=1;

}

if(\_result==1)//如果有运算则后面的往前移

{

j++;

while(a[j+2]!='\0')//后面的字符串往前移1格

{

a[j]=a[j+2];

j++;

}

a[j]='\0';

r-=2;

}

else//没有！0或！1则看下一个指针

i++;

}

}

就是把a\*b的各种可能列举出来

void biCondYsh(char a[],int j,int r)

{

int \_result=0,i=j;

while(i<r)

{

//=

j=i;

\_result=0;

if((j+2<strlen(a))&&(a[j]=='0')&&(a[j+1]=='=')&&(a[j+2]=='0'))

{

a[j]='1';

\_result=1;

}

else if((j+2<strlen(a))&&(a[j]=='0')&&(a[j+1]=='=')&&(a[j+2]=='1'))

{

a[j]='0';

\_result=1;

}

else if((j+2<strlen(a))&&(a[j]=='1')&&(a[j+1]=='=')&&(a[j+2]=='0'))

{

a[j]='0';

\_result=1;

}

else if((j+2<strlen(a))&&(a[j]=='1')&&(a[j+1]=='=')&&(a[j+2]=='1'))

{

a[j]='1';

\_result=1;

}

if(\_result==1)//如果有运算则后面的往前移

{

j++;

while(a[j+2]!='\0')//后面的字符串往前移1格

{

a[j]=a[j+2];

j++;

}

a[j]='\0';

r-=2;

}

else//没有！0或！1则看下一个指针

i++;

}

}

就是把a=b的各种可能列举出来

void condYsh(char a[],int j,int r)

{

int \_result=0,i=j;

while(i<r)

{

//-》

j=i;

\_result=0;

if((j+2<strlen(a))&&(a[j]=='0')&&(a[j+1]=='-')&&(a[j+2]=='0'))

{

a[j]='1';

\_result=1;

}

else if((j+2<strlen(a))&&(a[j]=='0')&&(a[j+1]=='-')&&(a[j+2]=='1'))

{

a[j]='1';

\_result=1;

}

else if((j+2<strlen(a))&&(a[j]=='1')&&(a[j+1]=='-')&&(a[j+2]=='0'))

{

a[j]='0';

\_result=1;

}

else if((j+2<strlen(a))&&(a[j]=='1')&&(a[j+1]=='-')&&(a[j+2]=='1'))

{

a[j]='1';

\_result=1;

}

if(\_result==1)//如果有运算则后面的往前移

{

j++;

while(a[j+2]!='\0')//后面的字符串往前移1格

{

a[j]=a[j+2];

j++;

}

a[j]='\0';

r-=2;

}

else//没有！0或！1则看下一个指针

i++;

}

}

就是把a-b的各种可能列举出来

void disConjYsh(char a[],int j,int r)

{

int \_result=0,i=j;

while(i<r)

{

//+

j=i;

\_result=0;

if((j+2<strlen(a))&&(a[j]=='0')&&(a[j+1]=='+')&&(a[j+2]=='0'))

{

a[j]='0';

\_result=1;

}

else if((j+2<strlen(a))&&(a[j]=='0')&&(a[j+1]=='+')&&(a[j+2]=='1'))

{

a[j]='1';

\_result=1;

}

else if((j+2<strlen(a))&&(a[j]=='1')&&(a[j+1]=='+')&&(a[j+2]=='0'))

{

a[j]='1';

\_result=1;

}

else if((j+2<strlen(a))&&(a[j]=='1')&&(a[j+1]=='+')&&(a[j+2]=='1'))

{

a[j]='1';

\_result=1;

}

if(\_result==1)//如果有运算则后面的往前移

{

j++;

while(a[j+2]!='\0')//后面的字符串往前移1格

{

a[j]=a[j+2];

j++;

}

a[j]='\0';

r-=2;

}

else

i++;

}

}

就是把a+b的各种可能列举出来

void jinwei(int nvar,char charVal[])

{

int flagsum=1;

for(int j=nvar-1;j>=0;j--)

{

if(charVal[j]=='1')

{

if(flagsum==1)//1+1=10//进位

{

charVal[j]='0';

flagsum=1;

}

else//1+0=1 不变

break;

}

else if(charVal[j]=='0')

{

if(flagsum==1)//0+1=1//没有进位

{

charVal[j]='1';

flagsum=0;

}

else//0+0结束

break;

}

}

}

如果有进位，根据flagsum的值以及charVal[j]的值相加比较来设定进位。

int main()

{

char pstate[120],pstate0[120],charList[120],charVal[120];

char minItem[1024][52],maxItem[1024][52];//最多10个变量

int k,i=0,nold=0,nnew=0,nvar=1,nRow=1,j=0,iMinItem=0,iMaxItem=0,lold;

printf("请输入公式（析+，合\*，条-，双=，否定！，01）：\n");

gets(pstate0);

fflush(stdin);

nvar=getalpha(pstate0,charList);

//真值表各个变元的值

nRow=1;

for(i=0;i<nvar;i++)

{

charVal[i]='0';

nRow=nRow\*2;

}

charVal[i]='\0';

//真值表的首行

printf("\n");

for(i=0;i<nvar;i++)

printf("%4c",charList[i]);

printf("%15c%s\n",' ',pstate0);

for(i=0;i<nvar;i++)

printf("%4c",'-');

printf("|");

for(i=0;i<60;i++)

printf("%c",'-');

printf("\n");

for(i=0;i<nRow;i++)

{

//真值表的各行

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

printf("%4c",charVal[j]);

//将值填入到公式中

pstate[0]='\0';

fillValue(pstate0,charList,nvar,charVal,pstate);

//计算公式的值

nold=strlen(pstate0)+1;

nnew=strlen(pstate0);

while(nnew<nold)

{

nold=strlen(pstate);

negatecal(pstate);

for(k=0;k<strlen(pstate);k++)

{

while(1)

{

nold=strlen(pstate);

negatecal(pstate);

nnew=strlen(pstate);

if(nold<=nnew)

break;

}//((a+b)-(c\*d))

if(pstate[k]=='(')

lold=k+1;

if(pstate[k]==')')

{

conYsh(pstate,lold,k);

disConjYsh(pstate,lold,k);

condYsh(pstate,lold,k);

biCondYsh(pstate,lold,k);

k=0;

negatecal(pstate);

括号内的内容用上面函数定义的法则操作化简

}

}

conYsh(pstate,0,strlen(pstate)+1);

disConjYsh(pstate,0,strlen(pstate)+1);

condYsh(pstate,0,strlen(pstate)+1);

biCondYsh(pstate,0,strlen(pstate)+1);

nnew=strlen(pstate);

最后的公式再操作

}

if(strlen(pstate)==1)

{

if(pstate[0]=='1')

{

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

minItem[iMinItem][j]=charVal[j];

minItem[iMinItem][j]='\0';

iMinItem++;

}

if(pstate[0]=='0')

{

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

maxItem[iMaxItem][j]=charVal[j];

maxItem[iMaxItem][j]='\0';

iMaxItem++;

}

}

printf("|");

printf("%15c%s",' ',pstate);

printf("\n");

//值+1

jinwei(nvar,charVal);

进位直接用函数，不用在用if语句判断了

}

for(i=0;i<iMinItem;i++)

{

if(i==0)

printf("m%s",minItem[i]);

else

printf("+m%s",minItem[i]);

}

printf("\n");

for(i=0;i<iMinItem;i++)

{

if(i==0)

printf("(");

else

printf("+(");

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

{

if(j==0)

{

if(minItem[i][j]=='1')

printf("%c",charList[j]);

else

printf("!%c",charList[j]);

}

else

{

if(minItem[i][j]=='1')

printf("\*%c",charList[j]);

else

printf("\*!%c",charList[j]);

}

}

printf(")");

}

printf("\n");

for(i=0;i<iMaxItem;i++)

{

if(i==0)

printf("M%s",maxItem[i]);

else

printf("\*M%s",maxItem[i]);

}

printf("\n");

for(i=0;i<iMaxItem;i++)

{

if(i==0)

printf("(");

else

printf("\*(");

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

{

if(j==0)

{

if(maxItem[i][j]=='0')

printf("%c",charList[j]);

else

printf("!%c",charList[j]);

}

else

{

if(maxItem[i][j]=='0')

printf("+%c",charList[j]);

else

printf("+!%c",charList[j]);

}

}

printf(")");

}

}

总体是把函数定义了一个范围，输入变量的时候要同时写进去范围，这样我认为更方便后面运算—比如说要先针对括号内化简，在针对上层括号化简……