证明A-(BUC)=(A-B)n(A-C)的过程一步一步实现，老师这是通过上一次作业改编而来的

代码：

#include<bits/stdc++.h>

typedef long long ll;

using namespace std;

int Conditional(char str[]);

int DeMorgan(char str[]);

bool subDeleSame(char );

string turnString(char str[]);

string distributivity(char str[]);

string turn\_s(const string & s);

string turn\_s2(const string & s);

string turn\_s3(const string & s);

string turn\_s4(const string & s);

string turn\_s5(const string & s);

string turn\_s6(const string & s);

string turn\_s7(const string & s);

int myReplace(string & s1,const string s2,const string s3)

{//把s1中粗线的第一个s2用s3替换

s1.replace(s1.find(s2),s2.size(),s3);

return 0;

}

int main() {

char s[1000];

Conditional(s);

string s1="x@A-(B|C)";

string s\_temp=s1;

DeMorgan(s);

string str2="(A-B)&(A-C)";

cout<<s1<<endl;

s1=turn\_s(s1);

cout<<s1<<endl;

s1=turn\_s2(s1);

cout<<s1<<endl;

DeMorgan(s);

s1=turn\_s3(s1);

cout<<s1<<endl;

s1=turn\_s4(s1);

cout<<s1<<endl;

s1=turn\_s5(s1);

cout<<s1<<endl;

s1=turn\_s6(s1);

cout<<s1<<endl;

s1=turn\_s7(s1);

cout<<s1<<endl;

for(int i=0;i<s1.size();i++){

if(s1.substr(i)==str2){

cout<<s\_temp.substr(2)<<"="<<str2<<endl;

}

}

return 0;

}

int Conditional(char str[]){

stack<char> stk;//处理前端 的stack

int bi/\*储存左指针\*/,ei;

char ans[10000];

for(int i=0; i<strlen(str); i++) { //遍历预处理字符

char fin[1000];//处理前端

char fin2[1000];

if(str[i]=='^') {

fin[0]='~';//在字符前加~和括号（

fin[1]='(';

int l=i,r=i;//l为左指针 r为右指针

//遇到括号

if(str[l-1]==')') {

int count=2;//作为fin的下标 fin[0]为~，fin[1]为（

l--;

int bracketNum=0;

while(str[l]!='('||bracketNum!=1) {//因为第一个')'要入stack所以 =1

if (str[l]==')')bracketNum++;

if(str[l]=='(')bracketNum--;//检查括号匹配

stk.push(str[l]);//把左括号之前的入栈

//cout<<stk.top()<<endl;

l--;//左移

}

bi=l;//储存左指针

while(!stk.empty()) {

fin[count++]=stk.top();

stk.pop();

}

} //无括号

else {

fin[1]=str[l-1];

bi=l-1;

}// 推的右边有括号

if(str[r+1]=='(') {

r++;

int count=0;

while(str[r]!=')') {//存在括号匹配问题

fin2[count++]=str[r];

r++;

}

ei=r;

fin2[count]=')';

} //右边无括号

else {

fin2[0]=str[r+1];

ei=r+1;

}

queue<char> que;//用于最后整合

for(int j=0; j<bi; j++)//将bi指针前面未处理部分入队列

que.push(str[j]);

for(int j=0; j<strlen(fin); j++)//将处理好的部分fin(运算符‘^’左边的运算数）入队列

que.push(fin[j]);

que.push('&');//插入‘|’运算符

for(int j=0; j<strlen(fin2); j++)//将'^' 右边的运算数入队列

que.push(fin2[j]);

for(int j=ei+1; j<strlen(str); j++)//将ei指针后面未处理部分入队列

que.push(str[j]);

int s=que.size();

for(int j=0; j<s; j++) {//推出所有队列元素附加给str

str[j]=que.front();

que.pop();

}

}

}

return 0;

}

int DeMorgan(char str[]){

int bi/\*储存左指针\*/,ei/\*储存右指针\*/;

int len=strlen(str);//获取长度

char str2[10000];

char a[]="";

queue<char> que;

char ans[10000];

strcpy(str2,str);//复制预处理字符储存到str2

for(int i=0;i<len;i++){

char fin[1000];

if(str2[i]=='~' && str2[i+1]=='('){//遇到~（

// cout<<1<<endl;

bi=i-1;//左指针

int j=i+2,cnt=0;//

int bracketNum=0;// 检查括号匹配

while(str2[j]!=')'||bracketNum!=0/\*j=j+2 '('不包含\*/){

// cout<<2<<endl;//没有遇到右括号

if (str[j]=='(')bracketNum++;

if(str[j]==')')bracketNum--;//检查括号匹配

if(str2[j]>='A' && str2[j]<='Z') {

// cout<<3<<endl;

fin[cnt++]='~';

fin[cnt++]=str[j];//在字母前面加'~'

// cout<<fin<<endl;

}

else if(str2[j]=='|') fin[cnt++]='&';

else if(str2[j]=='&')fin[cnt++]='|';

else fin[cnt++]=str2[j];//'~'照抄

j++;//指针向后扫

}

ei=j+1;//ei指针后面的未处理的部分

que.push('(');

for(int j=0;j<=bi;j++)//bi前面的未处理的部分入队列

que.push(str2[j]);

for(int j=0;j<strlen(fin);j++)//将处理好的部分入队列

que.push(fin[j]);

que.push(')');

for(int j=ei;j<len;j++){//ei指针后面的未处理的部分

que.push(str2[j]);

}

strcpy(str2,a);

int qLen=que.size();

for(int j=0;j<qLen;j++){

// cout<<que.front();

ans[j]=que.front();

que.pop();

}

// cout<<endl;

// cout<<ans<<endl;

strcpy(str2,a);

strcpy(str2,ans);

}

}

int len1=strlen(ans);

for(int i=0;i<len1;i++){

if(ans[i]=='~' && ans[i+1]=='~'){

len1=len1-2;

for(int j=i;j<len1;j++) {

ans[j]=ans[j+2];

}

}

}

ans[len1]='\0';//加上结尾标识符 /0

//cout<<ans<<endl;

strcpy(str,ans);

}

string turnString(char str[]){

string ans;

for(int i=0;i<strlen(str);i++)

ans+=str[i];

return ans;

}

string absorbChar(char str1[]){

int begini,endi;

string str,ans="";

str=turnString(str1);

stack<char> stk[27];

queue<char> que;

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

if(str[i]=='('){

begini=i;

i++;

while(str[i]!=')'){

if(str[i]>='a' && str[i]<='z'){

stk[str[i]-'a'].push(str[i]);

stk[str[i]-'a'].push(str[i+1]);

}

i++;

endi=i;

}

}

if(str[i]>='a' && str[i]<='z'){

if(!stk[str[i]-'a'].empty()){

if((stk[str[i]-'a'].top()=='&' && str[i-1]=='|')|| (stk[str[i]-'a'].top()=='|' && str[i-1]=='&') ){

for(int j=begini;j<=endi;j++) str[j]=' ';

str[endi]=str[i];

str[i]=' ';

str[i-1]=' ';

}

}

}

}

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

if(str[i]!=' ') {

ans+=str[i];

}

}

// cout<<ans<<endl;

return ans;

}

bool subDeleSame(char s){

switch(s){

case '^':case '&':case '|':

return true;

default:

return false;

}

}

string distributivity(char str[10000]){

string fin;

int len=strlen(str),si,bk,ek;

queue<char> qu1;

queue<char> qu2;

qu1.push('(');

qu2.push('(');

for(int i=0;i<len;i++){

if(str[i]=='(') bk=i;

if(str[i]==')') ek=i;

}

// cout<<bk<<" "<<ek<<endl;

for(int i=bk+1;subDeleSame(str[i])!= true;i++){

qu1.push(str[i]);

si=i;

// cout<<si<<endl;

}

for(int i=ek+1;i<len;i++)

qu1.push(str[i]);

qu1.push(')');

for(int i=si+2;subDeleSame(str[i])!= true && str[i]!=')';i++){

qu2.push(str[i]);

}

for(int i=ek+1;i<len;i++)

qu2.push(str[i]);

qu2.push(')');

while(!qu1.empty()){

fin=fin+qu1.front();

qu1.pop();

}

fin=fin+str[si+1];

while(!qu2.empty()){

fin=fin+qu2.front();

qu2.pop();

}

return fin;

}

string turn\_s(const string & s) {

string s1=s;

int pos1;

int i;//找到第一个大写字母

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

if(s1[i]>='A'&&s[i]<='Z')

break;

}

s1.replace(i+1,s1.size()-i-1,"&x#B|C");

return s1;

}

string turn\_s2(const string & s){

int i=0;

string s1=s;

for(i=0;i<s1.size();i++){

if(s[i]=='&')

break;

}

s1.replace(i+1,s1.size()-i-1,"!(x@B|x@C)");

return s1;

}

string turn\_s3(const string & s){

int i=0;

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

if(s[i]=='&')

break ;

}

string s1=s;

s1.replace(i+1,s1.size()-i-1,"(!x@B&!x@C)");

return s1;

}

string turn\_s4(const string & s){

int i=0;

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

if(s[i]=='&')

break ;

}

string s1=s;

s1.replace(i+1,s1.size()-i-1,"x#B&x#C");

return s1;

}

string turn\_s5(const string & s){

int i=0;

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

if(s[i]=='B')

break ;

}

string s1=s;

s1.insert(0,"(");

s1.insert(i+2,")");

s1.insert(i+4,"(");

s1.replace(s1.size()-2,2,"@A&x#C)");

return s1;

}

string turn\_s6(const string & s){

string s1=s;

for(int i=0;i<s1.size()-7;i++){

if(s1.substr(i,7)=="x@A&x#B"){

s1.replace(i,7,"x@A-B");

}

else if(s1.substr(i,7)=="x@A&x#C"){

s1.replace(i,7,"x@A-C");

}

}

return s1;

}

string turn\_s7(const string & s){

string s1=s;

int i=0;

s1.erase(0,1);

for(i=0;i<s1.size();i++){

if(s[i]=='A')

break;

}

s1.insert(i-1,"(");

for(;i<s1.size();i++){

if(s[i]=='x')

break;

}

s1.erase(i,2);

return s1;

}

运行图：

