# 一、书上P100页程序有误

中缀表达式变后缀表达式，书上100页所给程序如下：

void transfer(char\* temp,char\* out)

{

linkedstack<char> opst;

opst.push('#'); //将#先压入栈

char topch;//用于取栈顶元素

char outch;//用于取退栈元素

int i1=0,i2=0;

char ch=temp[i1++]; //利用ch从原中缀表达式中读取字符

while(opst.isempty()==false&&ch!='#') //逐个处理

{

if(isnum(ch)) //操作数

{

out[i2++]=ch;//输出到后缀表达式

ch=temp[i1++]; //读取下一字符

}

else //操作符

{

opst.gettop(topch); //取栈顶操作符给topch

if(isp(topch)<icp(ch)) //新操作符优先级高

{

opst.push(ch);//新操作符入栈

ch=temp[i1++]; //取下一操作符

}

else if (isp(topch)>icp(ch)) //新操作符优先级低

{

opst.pop(outch);//退栈

out[i2++]=outch;//输出退栈元素

}

else //栈顶元素和当前操作符优先级相等，即括号或井号

{

opst.pop(outch); //出栈

if(outch=='(')

ch=temp[i1++]; //跳过括号，读取下一字符

}

}

}

out[i2]='\0'; //输出字符串添加结尾

}

考虑一种极端情况，如只有一个操作符+。则分析程序运行过程。首先#作为开始标志入栈，从中缀表达式中读取一个字符，即+，+优先级比栈内操作符#优先级高，进栈，读取中缀表达式下一操作符。下一操作符是结尾标志#，while循环判断调件为false，不通过跳出。可以发现+根本没有输出到后缀表达式字符串。由于这样的原因，所有前缀表达式按书上程序转化后，都会丢失最后一个操作符。为了改正这个问题，应当在最后加一收尾工作，将栈内最后一个操作符退出并输出。改正后程序如下：

void transfer(char\* temp,char\* out)

{

linkedstack<char> opst;

opst.push('#'); //将#先压入栈

char topch;//用于取栈顶元素

char outch;//用于取退栈元素

int i1=0,i2=0;

char ch=temp[i1++]; //利用ch从原中缀表达式中读取字符

**while(opst.isempty()==false)** //逐个处理

{

if(isnum(ch)) //操作数

{

out[i2++]=ch;//输出到后缀表达式

ch=temp[i1++];

}

else //操作符

{

opst.gettop(topch); //取栈顶操作符给topch

if(isp(topch)<icp(ch)) //新操作符优先级高

{

opst.push(ch);//新操作符入栈

ch=temp[i1++]; //取下一操作符

}

else if (isp(topch)>icp(ch)) //新操作符优先级低

{

opst.pop(outch);//退栈

out[i2++]=outch;//输出退栈元素

}

else //栈顶元素和当前操作符优先级相等，即括号或井号

{

opst.pop(outch);

if(outch=='(')

ch=temp[i1++];

}

}

**if(ch=='#')** //收尾工作，把最后一个操作符输出,并且跳出循环

**{**

**opst.pop(outch);**//退栈

**out[i2++]=outch;**//输出退栈元素

break**;**

**}**

**}**

out[i2]='\0'; //输出字符串添加结尾

}

红色为经过修改的部分