实验三 算术表达式求值

#include <iostream>

using namespace std;

#define Maxsize 100

typedef int dataType;

typedef struct Stack

{

dataType \*top;

dataType \*base;

int stacksize;

}sqstack;

void create(sqstack \*s)

{

s->base=(dataType \*)malloc(Maxsize\*sizeof(dataType));

s->top=s->base;

s->stacksize=Maxsize;

return;

}

int push\_in(sqstack \*s,dataType value)

{

if(s->top-s->base==s->stacksize)

return 0;

\*s->top=value;

s->top++;

return 0;

}

int pop\_out(sqstack \*s,dataType \*elem)

{

if(s->base==s->top)

return 0;

s->top--;

\*elem=\*s->top;

return 0;

}

dataType GetTop(sqstack \*s)

{

if(s->base==s->top)

return 0;

return \*(s->top-1);

}

char Precede(char theta1,char theta2)

{

int i,j;

char pre[7][7]={

{'>','>','<','<','<','>','>'},

{'>','>','<','<','<','>','>'},

{'>','>','>','>','<','>','>'},

{'>','>','>','>','<','>','>'},

{'<','<','<','<','<','=','0'},

{'>','>','>','>','0','>','>'},

{'<','<','<','<','<','0','='}};

switch(theta1){

case '+': i=0; break;

case '-': i=1; break;

case '\*': i=2; break;

case '/': i=3; break;

case '(': i=4; break;

case ')': i=5; break;

case '=': i=6; break;

}

switch(theta2){

case '+': j=0; break;

case '-': j=1; break;

case '\*': j=2; break;

case '/': j=3; break;

case '(': j=4; break;

case ')': j=5; break;

case '=': j=6; break;

}

return(pre[i][j]);

}

int Operate(int a,char theta,int b)

{

switch(theta){

case'+':return a+b;

case'-':return a-b;

case'\*':return a\*b;

case'/':

if(b!=0)

return a/b;

else

{

printf("Divisor can not Be zero!\n");

exit(0);

}

}

return 0;

}

int In(char c)

{

switch(c){

case '+':

case '-':

case '\*':

case '/':

case '(':

case ')':

case '=':

return 1;

default:

return 0;

}

}

int evaluateExpression(char exp[])

{

sqstack OPND,OPTR;

dataType a,b,theta,x,X1,X2; /\* a，b，theta用于Operate函数

X用于存放多余的出栈字符

X1,X2用于归并 \*/

char ch; //读取字符的变量

int i=0; //指向存放表达式数组的下标指针（其实不是真正的指针，而是数组下标）

create(&OPND); //建立并初始化运算数栈OPND

create(&OPTR); //建立并初始化运算符栈OPTR

push\_in(&OPTR,'='); //先将“=”号压入OPTR栈（表达式也必须以“=”结束）

ch=exp[i++]; //ch读取字符数组第一个元素，并将指针i后移一位

while(ch!='='||GetTop(&OPTR)!='=') //表达式没有扫描完毕或OPTR的栈顶不为“=”

{

if(In(ch)) //判断ch是否为操作符

{

switch(Precede(GetTop(&OPTR),ch))//比较ch的与OPTR栈顶元素的优先级

{

case'<':

push\_in(&OPTR,ch);

ch=exp[i++]; //读取下一位字符并将指针向后偏移一位

break;

case'>':

pop\_out(&OPTR,&theta);

pop\_out(&OPND,&b);

pop\_out(&OPND,&a);

push\_in(&OPND,Operate(a,theta,b));

break;

case'=':

pop\_out(&OPTR,&x);

ch=exp[i++]; //读取下一位字符并将指针向后偏移一位

break;

}

}

else if(isdigit(ch)) //判断是否为数字

{

X1=ch-'0'; //将字符形式转化为数字

push\_in(&OPND,X1);

X2=X1;

ch=exp[i++]; //读取下一位字符并将指针向后偏移一位

while(isdigit(ch)) //判断下一位是否还是数字

{

X1=ch-'0';

X2=10\*X2+X1; //归并至X2

pop\_out(&OPND,&x);

push\_in(&OPND,X2);

ch=exp[i++]; //读取下一位字符并将指针向后偏移一位

}

}

else if(ch==' ') //判断是否为空格

{

while(ch==' ') //跳过若干个空格

{

ch=exp[i++]; //忽略空格，直接取下一位

}

}

else //若不是上述三种情况之一，则为非法字符

{

cout<<"输入错误:"<<endl;

exit(0); //返回错误提示

}

}

return(GetTop(&OPND)); //最后返回操作数栈顶为运算结果

}

int main()

{

char exp[100]; //定义一个字符数组用于存储表达式

int result;

cout<<"输入一个算术表达式:"<<endl;

gets(exp); //换成scanf（“%s”，exp）；也可以

result=evaluateExpression(exp);

cout<<exp<<result<<endl;

return 0;

}