总结报告

一 系统设计要求

能够实现加减乘除四则运算，可以实现加括号的运算，能实现三角函数，幂运算，取模运算，阶乘运算，对数运算等。三角函数运算等都支持括号里进行运算，再进行相应的运算，如sin(3\*5+8+log(10+5)),sin5,(4%3+10)!等，在输入时进行相应的判断，当遇到错误的输入时，不执行该操作

二 设计思路及关键程序代码

当输入时，我在运算符那儿做了一点特殊处理，二元运算符左右各留一个空格，一元运算符如果是数字在左边的则在左边留一个空格，如“log ”;如果是在右边的，则在右边留一个空格，如“ ！”，左括号为“（ ”右括号为“ ）”。这样有利于以后的读数，当读到一个空格时就得到一个CString类型的字符串，并将其放入相应的栈内。

首先将要计算的中缀式转为后缀式，这时候有两个栈s1和s2,如上所说，每次读到的字符串，要么是完整的一个数，要么是完整的运算符，因此只需判断是什么就行了。

如果是数字则直接进栈s1，如果是运算符则要与s2栈顶的运算符比较优先级，如果优先级较高，则直接入栈s2,否则将s2栈顶元素放入栈s1，然后从s2中抛出，并将该运算符入栈s2，如果遇到左括号则直接入栈s2，当遇到右括号时则将与之距离的最近的左括号之间的栈s2中的运算符依次放入栈s1.当读完以后，如果栈s2中还有运算符，则依次读入s1.

然后将后缀式颠倒一下顺序，将s1中的元素依次读入栈s3.并将其作为中转站。然后依次获得s3中的元素，如果是数字则直接入栈s4，如果是运算符，则判断，如果是二元运算符，则从栈s4中依次取两个元素进行运算，将运算结果放入栈s4,如果是一元运算符取s4的栈顶元素进行运算，并将运算结果入栈s4.最终s4中只剩一个数，即为最终运算结果。

class CCount : public CWnd

{

DECLARE\_DYNAMIC(CCount)

public:

CCount(CString);

virtual ~CCount();

void transform();//转为逆波兰后缀表达式

bool IsOprand1(CString);//判断运算符的函数

double calculate(void);//进行计算

int fac(int);//求阶乘的函数

bool Expression();//判断一个数里是否已经有小数点了

bool Judge();//判断左括号与右括号的数目

int Compare(char);//判断运算符优先级

protected:

DECLARE\_MESSAGE\_MAP()

CString m\_result;//输入的算术表达式

stack<CString>s1;//存储后缀表达式的栈

};

CCount.cpp文件中的部分代码

CCount::CCount(CString m\_test)

{

m\_test += " # "; //添加“#”作为转化为后缀表达式的停止点

m\_test.Replace("π","3.1415926"); //把输入的"π"用.1415926替换

m\_result = m\_test; //把m\_test的值传给m\_result

}

CCount::~CCount()

{

}

BEGIN\_MESSAGE\_MAP(CCount, CWnd)

END\_MESSAGE\_MAP()

// CCount 消息处理程序

bool CCount::IsOprand1(CString m\_a)

{

char p = m\_a.GetAt(0);

if(p=='+'||p=='-'||p=='\*'||p=='/'||p=='^'||p=='c'||p=='s'||p=='t'||p=='l'||p=='L'||p=='%')

{

return true;

}

else

{

return false;

}

}

int CCount::Compare(char m\_operator)

{

if(m\_operator=='('||m\_operator=='#')

{

return 1;

}

if(m\_operator=='+'||m\_operator=='-')

{

return 2;

}

if(m\_operator=='\*'||m\_operator=='/')

{

return 3;

}

if(m\_operator=='^'||m\_operator=='%'||m\_operator=='c'||m\_operator=='s'||m\_operator=='t'||m\_operator=='l'||m\_operator=='L')

{

return 4;

}

}

int CCount::fac(int n)

{

int i;

int j = 1;

for(i=1;i<=n;i++)

{

j\*=i;

}

return j;

}

void CCount::transform(void)

{

stack<CString>s2;

s2.push("#"); //用#压入栈底

s1.push("#"); //用#压入栈底

CString m\_string;

m\_string = m\_result.SpanExcluding(" "); //读出从开始到第一个空白断点处的字符串

m\_result.Delete(0,m\_string.GetLength()+1); //删除m\_string字符串并其之后的一个空白处

char pp = m\_string.GetAt(0); //读出m\_string的第一个字符

while(pp != 35) //判断是否读完

{

if(!IsOprand1(m\_string))

{

if(pp !='(' && pp !=')')

{

s1.push(m\_string); //如果是数字或者是“！”则入栈s1

}

else

{

if(pp =='(')

{

s2.push(m\_string); //如果是"("则入栈s2

}

else

{

char n = s2.top().GetAt(0);

while( n != '(')

{

s1.push(s2.top());

s2.pop();

n = s2.top().GetAt(0);

} //如果碰到")"则把s2中最后一个"("之前的符号依次入栈s1

s2.pop(); //并把该左括号抛出

}

}

}

else

{

char m = s2.top().GetAt(0);

CString m\_cstr;

m\_cstr= m\_string.GetAt(0);

if(Compare(m)<Compare(pp))

{

s2.push(m\_cstr);

}

else

{

s1.push(s2.top());

s2.pop();

s2.push(m\_cstr);

}

//当是运算符的时候，判断栈s2顶端的运算符优先级低于该运算符，如果是则

} //将该运算符直接入栈s2，否则使s2栈顶运算符进入s1;并将该运算符直接入栈s2

m\_string = m\_result.SpanExcluding(" ");

m\_result.Delete(0,m\_string.GetLength()+1);

pp = m\_string.GetAt(0);

}

char q ;

q = s2.top().GetAt(0);

while(q!='#')

{

s1.push(s2.top());

s2.pop();

q = s2.top().GetAt(0); //将s2栈中剩下的运算符放入s1

}

}

double CCount::calculate(void)

{

stack<CString>s3; //存储顺序相反的后缀表达式

stack<double>s4; //存储中间计算结果以及最终计算结果

s3.push("#");

char a;

a = s1.top().GetAt(0);

while(a !='#')

{

s3.push(s1.top());

s1.pop();

a =s1.top().GetAt(0);

} //将栈s1中的字符依次读入栈s3

char b = s3.top().GetAt(0);

while(b!='#')//

{

if(!IsOprand1(s3.top()))

{

CString m\_pp = s3.top();

double m\_p = atof(m\_pp.GetBuffer(m\_pp.GetLength()+1));

s4.push(m\_p);

s3.pop();

} //如果读到的是数字，则将其转成double型，并放入栈s4

else

{

double number1;

double number2;

double number3;

if(b =='+'||b=='-'||b=='\*'||b=='/'||b=='^')

{

s3.pop();

number1 = s4.top();

s4.pop();

number2 = s4.top();

s4.pop();

switch(b)

{

case'+':

{number3 = number2+number1;

break;

}

case'-':

{

number3 = number2-number1;

break;

}

case'\*':

{

number3 = number2\*number1;

break;

}

case'/':

{

number3 = number2/number1;

break;

}

case'^':

{

number3 = pow(number2,number1);

break;

}

}

s4.push(number3);

}

if(b =='%')

{

s3.pop();

int number5;

number5 = int(s4.top());

s4.pop();

int number6;

number6 = int(s4.top());

s4.pop();

int number7 = number5%number6;

s4.push(number7);

}

if(b=='!')

{

s3.pop();

int number4;

number4 = int(s4.top());

s4.pop();

int number8 = fac(number4);

s4.push(number8);

}

if( b =='c'||b =='s'||b =='t'||b =='l'||b =='L')

{

s3.pop();

number1 = s4.top();

s4.pop();

switch(b)

{

case'c':

{

s4.push(cos(number1));

break;

}

case's':

{

s4.push(sin(number1));

break;

}

case't':

{

s4.push(tan(number1));

break;

}

case'l':

{

s4.push(log10(number1));

break;

}

case'L':

{

s4.push(log(number1));

break;

}

}

} //如果读入的是运算符，则依次进行相应的运算

}

b = s3.top().GetAt(0);

}

return s4.top();

}

当输入时会进行相应的判断，如当输入是数字时，会判断当前m\_show的最后一个字符，如果是")"或者是"!"时，因此操作是错误的，所以不会读入m\_show中，即不执行该按键操作，否则读入m\_show

void CCalculatoraDlg::OnBnClickednum1()

{

UpdateData(TRUE);

if(m\_show.GetLength()>=1)

{

char m\_p;

m\_p = m\_show.GetAt(m\_show.GetLength()-1);

if(m\_p == ')'||m\_p =='!')

{

m\_show+="";

}

else

{

m\_show+="1";

}

}

else

{

m\_show+="1";

}

UpdateData(FALSE);//当上一次输入的是")"或者是"!"时，因此操作是错误的，所以不会读入m\_show中，以下数字类似

}

当输入的是运算符时，判断m\_show的最后一个字符是否为空格，如果是则上一个是运算符，因为不允许两个运算符挨着，所以不执行其操作，并且如果该m\_show是空的，该操作也是错误的，也不执行，其他情况则正常执行。如

void CCalculatoraDlg::OnBnClickedadd()

{

UpdateData(TRUE);

if(m\_show.GetLength()>=1) //不支持两个二元运算符连着输入，以下类似

{ char m\_pp;

m\_pp = m\_show.GetAt(m\_show.GetLength()-1);

if (m\_pp == ' ')

{

m\_show+="";

}

else

{

m\_show+=" + ";

}

}

else

{

m\_show+="";

}

UpdateData(FALSE);// TODO: 在此添加控件通知处理程序代码

}

执行删除操作时，因不同的运算符会有不同的长度，所以要进行判断，并执行相应的操作，数字和小数点则每次删去一位。

void CCalculatoraDlg::OnBnClickeddelete()//对不同的字符串删除时，执行不同的动作

{

if(m\_show.GetLength()>=2)

{

char m\_pp;

m\_pp = m\_show.GetAt(m\_show.GetLength()-1);

if(m\_pp ==')'||m\_pp =='!')

{

m\_show.Delete(m\_show.GetLength()-2,2);

}

else

{

char m\_p;

m\_p = m\_show.GetAt(m\_show.GetLength()-2);

if(m\_pp ==' ')

{

if(m\_p=='('||m\_p=='n'||m\_p=='s'||m\_p=='g')

{

if(m\_p =='(')

{

m\_show.Delete(m\_show.GetLength()-2,2);

}

else

{

m\_show.Delete(m\_show.GetLength()-4,4);

}

}

else

{

m\_show.Delete(m\_show.GetLength()-3,3);

}

}

else

{

m\_show.Delete(m\_show.GetLength()-1,1);

}

}

}

else

{

m\_show.Delete(m\_show.GetLength()-1,1);

}

UpdateData(FALSE);

// TODO: 在此添加控件通知处理程序代码

}

当输入小数点时，不仅要判断紧挨着的是否是数字，还要判断这个数中是否已有一个小数点，然后执行相应的动作。为此在CCount类中又增加了一个函数来判断是否已有小数点

void CCalculatoraDlg::OnBnClickedcomma()//当输入小数点时，判断上一次输入的字符串，不支持错误的操作

{

UpdateData(TRUE);

CCount mycal(m\_show);

if(m\_show.GetLength()>1)

{ char m\_pp;

m\_pp = m\_show.GetAt(m\_show.GetLength()-2);

if (m\_pp == '+'||m\_pp == '-'||m\_pp == '\*'||m\_pp == '/'||m\_pp=='^'||m\_pp=='c'||m\_pp=='s'||m\_pp=='t'||m\_pp=='l'||m\_pp=='L'||m\_pp=='%'||m\_pp=='!'||m\_pp=='('||m\_pp==')')

{

m\_show+="";

}

else

{

char m\_p;

m\_p = m\_show.GetAt(m\_show.GetLength()-1);

if (m\_p == '.')

{

m\_show+="";

}

else

{

if(mycal.Expression())

{

m\_show+="";

}

else

{

m\_show+=".";

}

}

}

}

else

{

if(m\_show.GetLength()==0)

{

m\_show+="";

}

else

{

m\_show+=".";

}

}

UpdateData(FALSE);// TODO: 在此添加控件通知处理程序代码

}

bool CCount::Expression()//遍历上一个运算符之后的数字串，判断是否已有小数点

{

char m\_str;

m\_str = m\_result.GetAt(m\_result.GetLength()-4);

int i=4;

while(m\_str !=' ' && m\_result.GetLength()-i>0)

{

if(m\_str =='.')

{

return true;

break;

}

else

{

i+=1;

m\_str = m\_result.GetAt(m\_result.GetLength()-i);

continue;

}

}

return false;

}

输入左括号和右括号时

void CCalculatoraDlg::OnBnClickedopenp()//输入左括号时，判断上一次输入的字符串，是否输入是合理的，若是，则支持

{

UpdateData(TRUE);

if(m\_show.GetLength()>=1)

{ char m\_pp;

m\_pp = m\_show.GetAt(m\_show.GetLength()-1);

if (m\_pp == ' ')

{

m\_show+="( ";

}

else

{

m\_show+="";

}

}

else

{

m\_show+="( ";

}

UpdateData(FALSE);

// TODO: 在此添加控件通知处理程序代码

}

void CCalculatoraDlg::OnBnClickedclosedp()//输入右括号时，判断上一次输入的字符串，以及m\_show中的左括号与右括号的数目，判断是否支持该操作，并执行相应的动作

{

UpdateData(TRUE);

if(m\_show.GetLength()>=1)

{ char m\_pp;

m\_pp = m\_show.GetAt(m\_show.GetLength()-1);

if (m\_pp == ' ')

{

m\_show+="";

}

else

{

CCount myc(m\_show);

if(myc.Judge())

{

m\_show+=" )";

}

else

{

m\_show+="";

}

}

}

else

{

m\_show+="";

}

UpdateData(FALSE);

// TODO: 在此添加控件通知处理程序代码

}

对应的在CCount 类中增加的函数

bool CCount::Judge()//遍历整个字符串判断左括号个数是否大于右括号个数

{

int m\_left=0;

int m\_right=0;

for(int i=0;i<m\_result.GetLength();i++)

{

char m\_c;

m\_c = m\_result.GetAt(i);

if(m\_c == '(')

{

m\_left+=1;

}

if(m\_c == ')')

{

m\_right+=1;

}

}

if(m\_left>m\_right)

{

return true;

}

else

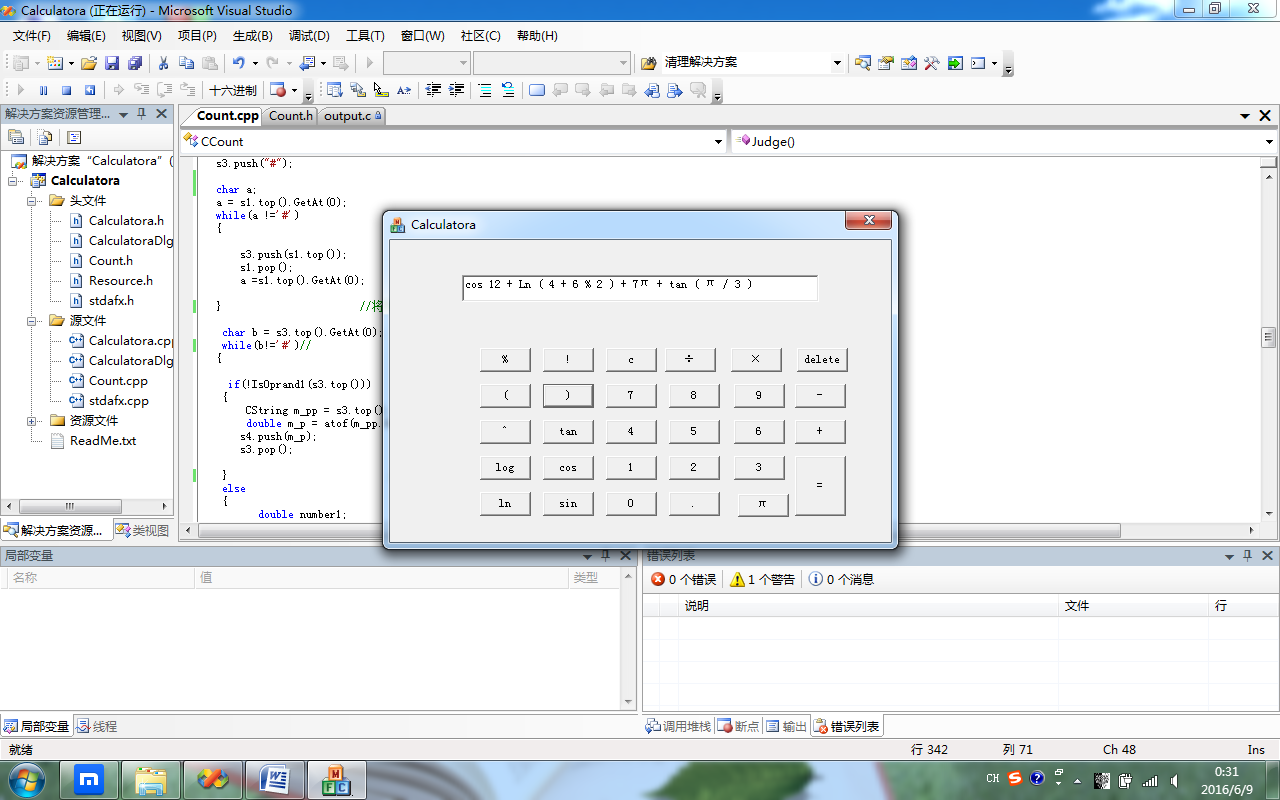
{

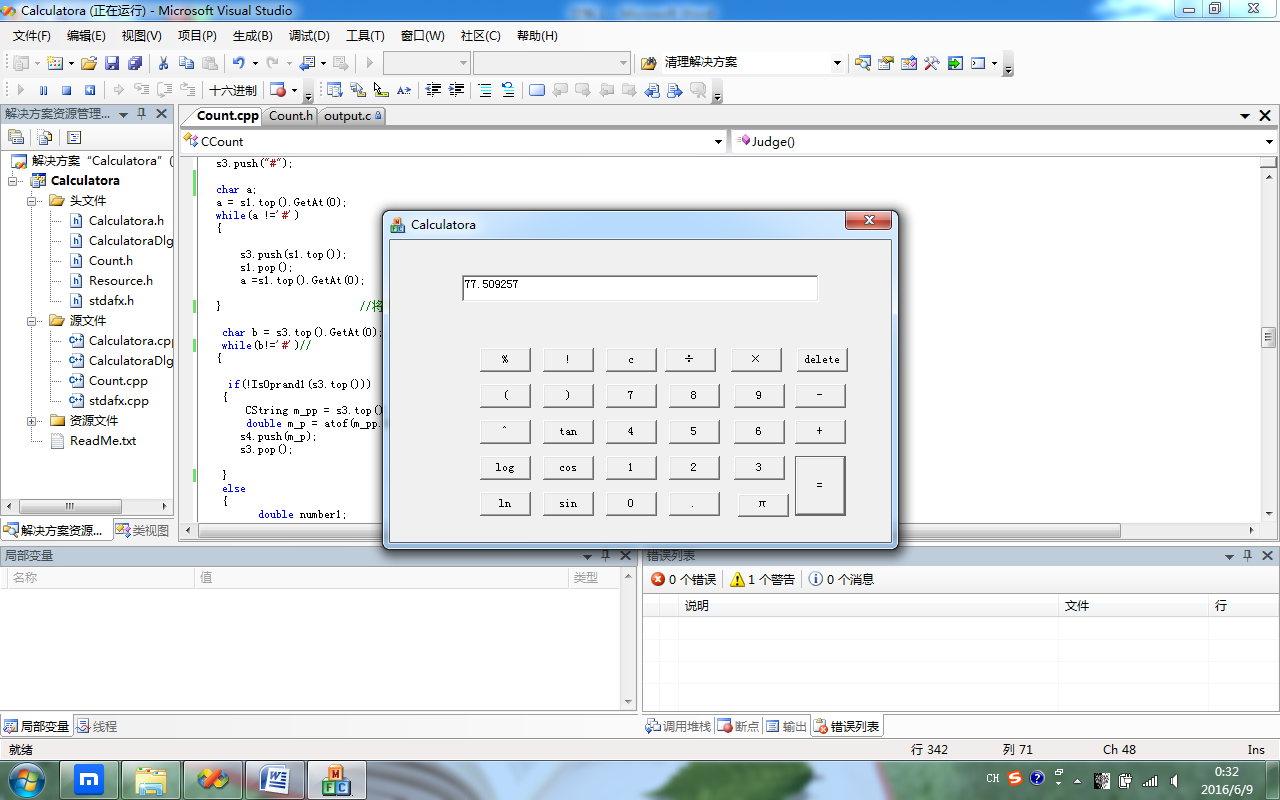
return false;

}

}

示例





总结

这个计算器已有二元运算符，一元左运算，一元右运算符的运算规则，按照类似的处理，一般的运算都能进行