**数据结构课程设计**

**班级：1615403**

**学号：161540205**

**姓名：张佳钰**

**指导老师：孙涵**

2、算术表达式求值 (必做) （栈）

[问题描述]

　　一个算术表达式是由操作数(operand)、运算符(operator)和界限符(delimiter)组成的。假设操作数是正实数，运算符只含加减乘除等四种运算符，左右括号和表达式起始、结束符“#”，如：#（35-25）\*（12+8/4）#。引入表达式起始、结束符是为了方便。编程利用“算符优先法”求算术表达式的值。

[基本要求]

（1） 从键盘或文件读入一个合法的算术表达式，输出正确的结果。

（2） 显示输入序列和栈的变化过程。

（3） 考虑算法的健壮性，当表达式错误时，要给出错误原因的提示。

采用的数据结构：

栈

算法思想：

现将表达式转化为后缀表达式然后对波兰表达式求值

源程序：

#include<iostream>

#include<string>

#include<fstream>

using namespace std;

#include<stack>

#include<map>

#define EXPRESSION\_FILE "exp.txt"

char int\_divide='\_';

int count(int left,int right,char op)

{

switch(op)

{

case '+':

return left+right;

case '-':

return left-right;

case '\*':

return left\*right;

case '/':

return left/right;

default:

return -1;

}

}

int priority(char a)

{

switch(a)

{

case '(':

return 1;

case '+':

return 2;

case '-':

return 2;

case '\*':

return 3;

case '/':

return 3;

default:

return -1;

}

}

int nipolan(char \*a,char \*&post)

{

stack<char>p;

int i=0,j=0;

int flag=0;

// memset(post,' ',sizeof(post));

i++;

cout<<"origin expression: "<<a<<endl;

while(a[i]!='#')

{

if(a[i]=='(')

{

p.push(a[i]);

}

else if(a[i]>='0'&&a[i]<='9')

{

if(flag==0)

{

post[j++]=a[i];

post[j]=int\_divide;

j++;

flag=1;

}

else

{

post[j-1]=a[i];

post[j]=int\_divide;

j++;

}

}

else if(a[i]=='+'||a[i]=='-'||a[i]=='\*'||a[i]=='/'||a[i]=='('||a[i]==')')

{

flag=0;

if(a[i]==')')

{

while(!p.empty()&&p.top()!='(')

{

post[j]=p.top();

j++;

p.pop();

}

if(p.empty())

{

return 0;

}

else

{

if(p.top()=='#')

{

cout<<"wrong expresion"<<endl;

return 0;

}

if(p.top()=='(')

{

p.pop();

}

}

}

else if(p.empty()||priority(a[i])>priority(p.top()))

{

p.push(a[i]);

}

else if(a[i]=='(')

{

p.push(a[i]);

}

else

{

while(priority(a[i])<=priority(p.top()))

{

post[j++]=p.top();

p.pop();

if(p.empty())

{

break;

}

}

p.push(a[i]);

}

}

i++;

}

while(!p.empty())

{

post[j++]=p.top();

p.pop();

}

post[j]='#';

post[++j]='\0';

cout<<"post: "<<post<<endl;

return 1;

}

int count\_expression(char post[],char \*a)

{

char \*exp;

memset(post,' ',sizeof(post));

if(nipolan(a,post))

{

exp=post;

stack<int>p;

stack<char>op;

while(\*exp!='#')

{

if(\*exp=='(')

{

return 0;

}

if(\*exp>='0'&&\*exp<='9')

{

int tmp=0;

while(\*(exp)!=int\_divide)

{

tmp=tmp\*10+(\*exp-'0');

exp++;

}

p.push(tmp);

exp++;

}

else if(\*exp!='#')

{

int left,right;

char chr;

right=p.top();

p.pop();

left=p.top();

p.pop();

p.push(count(left,right,\*exp));

exp++;

}

}

cout<<endl<<p.top()<<endl<<endl;

}

else

{

return 0;

}

return 1;

}

int main()

{

fstream exp\_file;

int i=1;

exp\_file.open(EXPRESSION\_FILE,ios::in);

if(exp\_file.fail())

{

cout<<"can't open file"<<endl;

exit(1);

}

while(!exp\_file.eof())

{

char post[30];

char exp[30];

exp\_file>>exp;

cout<<i++<<" :　"<<endl;

if(!(count\_expression(post,exp)))

{

cout<<"wrong expression. :"<<exp<<endl<<endl;

}

}

exp\_file.close();

return 0;

}

测试数据：

exp.txt

#12+90\*(90+99)-29+12/2#

#13+90\*(90+99)-29+12/2#

#14+90\*(90+99)-29+12/2#

#15+90\*(90+99-29+12/2#

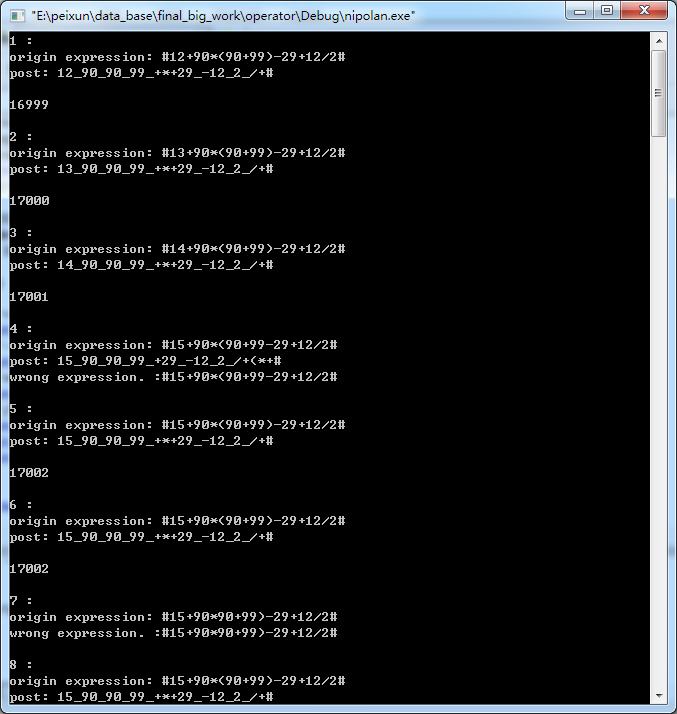
#15+90\*(90+99)-29+12/2#

#15+90\*(90+99)-29+12/2#

#15+90\*90+99)-29+12/2#

#15+90\*(90+99)-29+12/2#

测试结果：



代码行数：

209