Project2 实验报告

22336044 陈圳煌

1、 程序功能简要说明。

输入一个不含变量的表达式，利用栈的功能实现对表达式求值，表达式应该包含运算符号（+、-、*、/、^）和左右括号，以及最后以#作为结束标志，能够实现对表达式求值。（若除数为 0 则显示结果为 0，若终端输入不符合要求则显示 error，结果为-1）

2、 程序的部分关键代码及其说明

首先以字符串的形式输入表达式，并将表达式传入函数 operation（）中。（使用 getline 输入确保表达式中有空格时不会出错）

```
string op;
getline(cin ,op);
double result=operation(op);
```

operation（）函数:

```
stack<char> op_sta;//运算符栈
stack<double> num_sta;//数字栈
char c='#';
op_sta.push(c);
```

首先先创建两个栈，一个用来存放数字，一个用来存放字符，并把“#”存入运算符栈的栈底。接着遍历栈:

（1）空格的情况:

```
for(int i=0;i<len;){
    if(a[i]==' '){//如果是空格，则跳到下一位
        i++;
        continue;
    }
```

（2）数字的情况:

```
bool isnum(char a){//判断是否为数字
    if(a>='0' && a<='9'){
        return true;
    }else{
        return false;
    }
}
```

先判断是否为数字，并且将整数和小数分情况入栈。

```
if(isnum(a[i])==1){//如果是数字，判断是几位数，并插入到数字栈中
    double num=0;
    while(i<len && isnum(a[i])==1){//整数的情况
        num=num*10+(a[i]-'0');
        i++;
    }
```

```
if(a[i]=='.'){//出现小数的情况
    i++;
    int con=0;
    while(i<len && isnum(a[i])==1){
        num=num*10+(a[i]-'0');
        i++;
        con++;
    }
    while(con){
        con--;
        num=num/10;
    }
}
```

（3）运算符的情况：

```
}else{//不是数字的情况
    if(a[i]=='('){//如果为（ ，可以忽略
        op_sta.push(a[i]);
        i++;
        while(a[i]==' '){
            i++;
        }
        if(a[i]=='-'){//处理负数情况
            num_sta.push(0);
        }
```

```
}else{
    int pril=get_pri(a[i]);
    int pri2=get_pri(op_sta.top());
    if(pril<=pri2){//进行计算的情况
        char top_op=op_sta.top();
        op_sta.pop();
        double num2=num_sta.top();
        num_sta.pop();
        double num1;
        if(num_sta.empty()){//处理出现负数情况
            num1=0;
        }else{
            num1=num_sta.top();
            num_sta.pop();
        }
        double resl=calculate(num1,top_op,num2);
        num_sta.push(resl);
    }else{
        op_sta.push(a[i]);
        i++;
    }
}
```

```
}else if(a[i]==')'){//碰到），分情况运算
    if(op_sta.top()=='('){//如果（）内运算已完成，则直接消除括号
        op_sta.pop();
        i++;
    }else if(op_sta.top()=='#'){//出错的情况1
        cout << "error!" << endl;
        return -1;
    }else{
        char top_op=op_sta.top();
        op_sta.pop();
        double num2=num_sta.top();
        num_sta.pop();
        double num1;
        if(num_sta.empty()){
            num1=0;
        }else{
            num1=num_sta.top();
            num_sta.pop();
        }
        double resl=calculate(num1,top_op,num2);
        num_sta.push(resl);
    }
```

```
}else if(a[i]=='#'){//如果碰到#，则结束运算过程
    if(op_sta.top()=='#'){//如果两个#之间无其他运算符
        if(num_sta.empty()){
            cout << "error!" << endl;
            return -1;
        }else{
            return num_sta.top();
        }
    }else if(op_sta.top()=='('){//（后出现#的特殊情况
        cout << "error!" << endl;
        return -1;
    }else{//正常运算情况下的最后一步
        while(op_sta.top()!='#'){
            char top_op=op_sta.top();
            op_sta.pop();
            double num2=num_sta.top();
            num_sta.pop();
            double num1;
            if(num_sta.empty()){
                num1=0;
            }else{
                num1=num_sta.top();
                num_sta.pop();
            }
            double res1=calculate(num1,top_op,num2);
            num_sta.push(res1);
        }
        return num_sta.top();
    }
}
```

分别讨论遍历过程中出现（、四则运算、^、）和 # 的情况。（图片中附说明）

其中，get_pri（）函数用来获取四则运算符的优先级

```
int get_pri(char op){//判断符号的优先级
    switch(op){
        case '+':
            return 1;
        case '-':
            return 1;
        case '*':
            return 2;
        case '/':
            return 2;
        case '^':
            return 3;
        case '#':
            return 0;
    }
}
```

calculate（）函数用来进行判断后的数字、运算符、数字的计算：

```
double calculate(double num1,char ope, double num2){//进行数字间的计算
    switch (ope){
        case '+':
            return num1+num2;
        case '-':
            return num1-num2;
        case '*':
            return num1*num2;
        case '/':
            if(num2==0){
                return 0;
            }else{
                return num1/num2;
            }
        case '^':
            return pow(num1,num2);
        default:
            return 0;
    }
}
```

3、部分实验案例

```
欢迎使用表达式求值计算器
请输入表达式:
[请按照正确的格式: '(' 需要有')'配对, 结尾请加一个'#'表示完成输入]
3*(7-2)#
表达式结果为: 15
```

```
欢迎使用表达式求值计算器
请输入表达式:
[请按照正确的格式: '(' 需要有')'配对, 结尾请加一个'#'表示完成输入]
8#
表达式结果为: 8

--------------------------------
```

```
欢迎使用表达式求值计算器
请输入表达式:
[请按照正确的格式: '(' 需要有')'配对, 结尾请加一个'#'表示完成输入]
1+2+3+4#
表达式结果为: 10

--------------------------------
```

```
欢迎使用表达式求值计算器
请输入表达式:
[请按照正确的格式: '(' 需要有')'配对, 结尾请加一个'#'表示完成输入]
1024/4*8#
表达式结果为: 2048

--------------------------------
```

```
欢迎使用表达式求值计算器
请输入表达式:
[请按照正确的格式: '(' 需要有')'配对, 结尾请加一个'#'表示完成输入]
1024/(2*4)#
表达式结果为: 128
```

```
请输入表达式:
[请按照正确的格式: '(' 需要有')'配对, 结尾请加一个'#'表示完成输入]
2*(6+2*(3+6*(6+6)))#
表达式结果为: 312
```

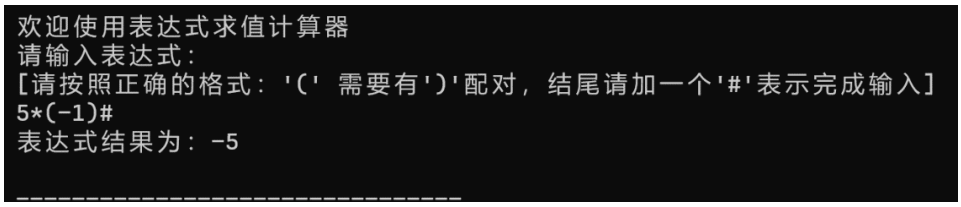
```
欢迎使用表达式求值计算器
请输入表达式:
[请按照正确的格式: '(' 需要有')'配对, 结尾请加一个'#'表示完成输入]
8/(9-9)#
表达式结果为: 0

--------------------------------
```

（除数为 0 时结果显示 0）

```
欢迎使用表达式求值计算器
请输入表达式:
[请按照正确的格式: '(' 需要有')'配对, 结尾请加一个'#'表示完成输入]
81+1*5#
表达式结果为: 86

--------------------------------
```



出现负数的情况

4、 程序运行方式简要说明

点开可执行文件“算法表达式求值.exe”后，按正确的格式输入表达式即可输出结果，若出现输入格式问题，则会显示“error”结果为-1。