项目说明文档

数据结构课程设计

——算数表达式求解

作 者 姓 名：安江涛

学　　号：1952560

指 导 教 师：张颖

学院、 专业：软件学院 软件工程

同济大学

Tongji University

目　　录

1　分析.....1
1.1 项目简介.....1
2 设计.....1
2.1 数据结构设计.....1
2.2 类结构设计.....1
2.3 成员与操作设计.....1
3 实现.....3
3.1 中缀表达式转后缀表达式.....3
3.1.1 流程图.....3
3.1.2 核心代码.....3
3.2 计算后缀表达式.....6
3.2.1 流程图.....6
3.2.2 核心代码.....6
3.4 总体系统的实现.....10
3.4.1 总体系统流程图.....10
3.4.2 总体系统核心代码.....10
3.4.3 总体系统截屏示例.....13

1 分析

1.1 项目简介

从键盘上输入中缀算数表达式，包括括号，计算出表达式的值。

程序对所有输入的表达式作简单的判断，如表达式有错，能给出适当的提示。支持包括加减，乘除取余，乘方和括号等操作符，其中优先级是等于<括号<加减<乘除取余<乘方

能处理单目运算符：+或-。

2 设计

2.1 数据结构设计

算数表达式求值，常规方法为将中缀表达式转后缀表达式（用 vector 来储存），进而用 stack 来求值。本题用到的 stack 以及 vector 均为手写，封装在头文件。

2.2 类结构设计

本项目只有一个类，eval 类，实现中缀表达式转后缀表达式、后缀表达式求值等操作。

2.3 成员与操作设计

eval 类

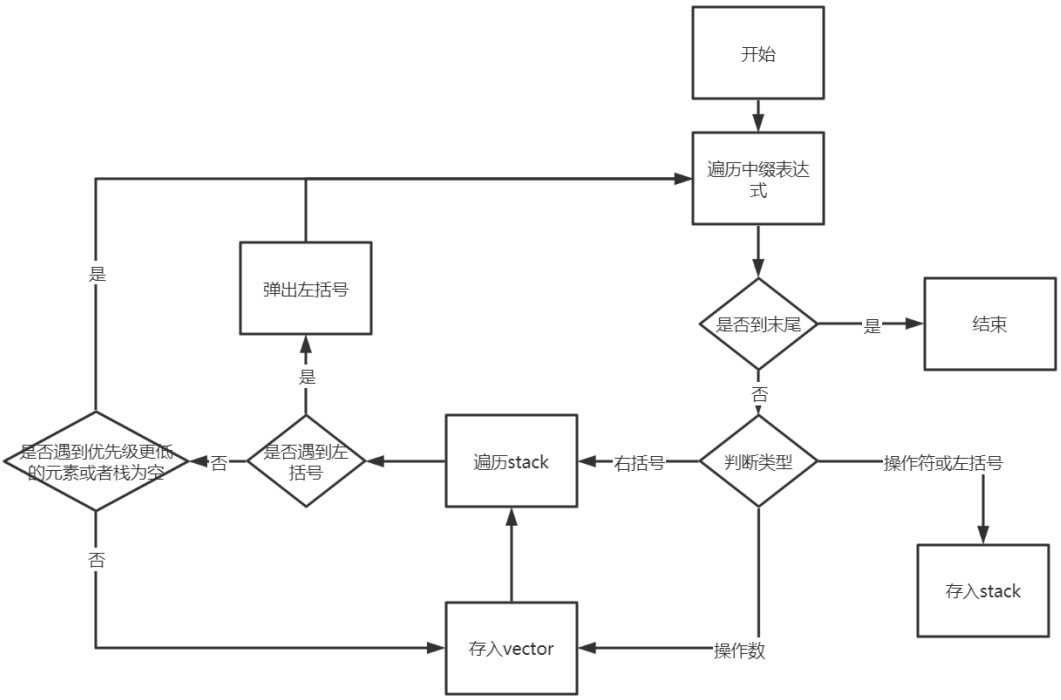
```
struct expre {
    bool isnum;//1为数字
    char oper;
    int num;
};

class eval {
public:
    eval(std::string& str) {
        flag = false;
        infix = str;
        change();
        calculate();
    }
private:
    void change();//中缀转后缀
    int priority(char x);//优先级
    void calculate();//计算
    friend std::ostream& operator<<(std::ostream& os, const eval& e) {
        if (e.flag) {
            os << '\n' << '\n';
            return os;
        }
        os << e.ans << '\n' << '\n';
        return os;
    }
private:
    std::string infix;//存中缀表达式
    Vector<expre> suffix;//存后缀表达式
    int ans;
    bool flag;//表示表达式有无错误
};
```

3 实现

3.1 中缀表达式转后缀表达式

3.1.1 流程图



3.1.2 核心代码

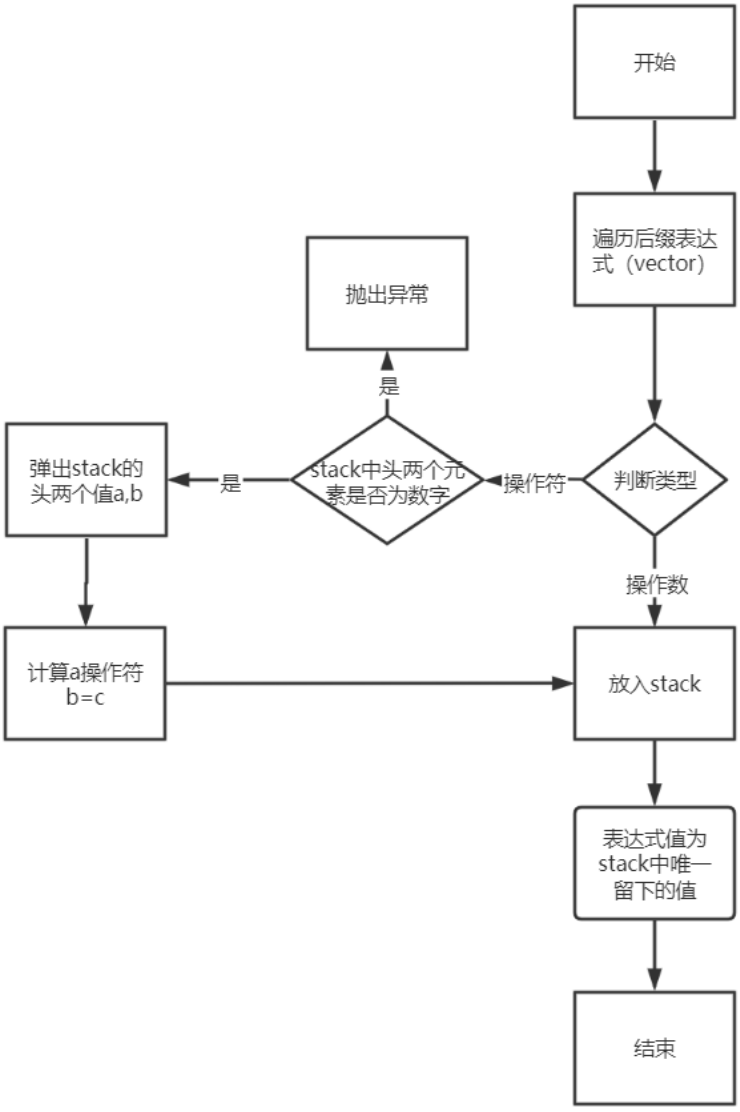
```
int eval::priority(char x) {
    if (x == ')' || x == '(')
        return 5;
    if (x == '+' || x == '-')
        return 1;
    if (x == '*' || x == '/')
        return 2;
    if (x == '%')
        return 3;
    if (x == '^')
        return 4;
    return -1;
}

void eval::change() {
    Stack<char> oper;
    bool flag = false;
    if (infix[0] == '+' || infix[0] == '-')
        flag = true;
    for (int i = 0; i < infix.size(); ) {
        if (infix[i] >= '0' && infix[i] <= '9') {
            flag = false;
            std::string str;
            while (infix[i] >= '0' && infix[i] <= '9') {
                str.push_back(infix[i]);
                i++;
            }
            int k = 1;
            int num = 0;
            for (int i = str.size() - 1; i >= 0; i--) {
                num += k * (int)(str[i] - '0');
                k *= 10;
            }
            expre e;
            e.isnum = true; e.num = num; e.oper = '?';
            suffix.push_back(e);
        }
        else {
            if (infix[i] == ')') {//如果是‘）’
                while (oper.top() != '(') {
                    expre e;
                    e.isnum = false; e.num = 0; e.oper = oper.top();
                    oper.pop();
                    suffix.push_back(e);
```

```
            }
            oper.pop();
            i++;
        }
        else {//如果不是‘）’
            if ((infix[i] == '+' || infix[i] == '-') && flag) {
                flag = false;
                expre e;
                e.isnum = true; e.num = 0;
                suffix.push_back(e);
            }
            if (infix[i] == '(')
                flag = true;
            while ((!oper.empty()) && (oper.top() != '(') && (priority(infix[i]) <= priority(oper.t
op()))) {
                expre e;
                e.isnum = false; e.num = 0, e.oper = oper.top();
                oper.pop();
                suffix.push_back(e);
            }
            oper.push(infix[i]);
            i++;
        }
    }
  }
  while (!oper.empty()) {
      expre e;
      e.isnum = false; e.num = 0, e.oper = oper.top();
      oper.pop();
      suffix.push_back(e);
  }
}
```

3.2 计算后缀表达式

3.2.1 流程图



3.2.2 核心代码

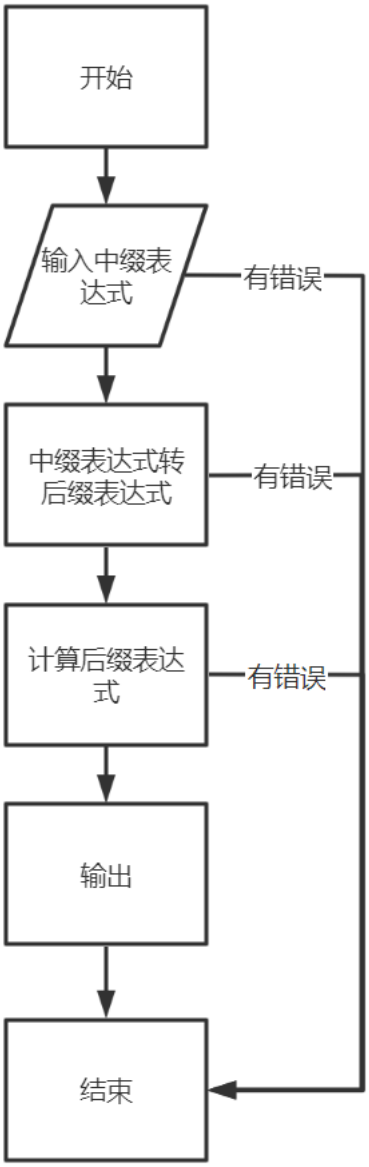
```
void eval::calculate() {
    Stack<int> cal;
    try {
        for (int i = 0; i < suffix.size(); i++) {
            if (suffix[i].isnum)
                cal.push(suffix[i].num);
            else {
                char oper = suffix[i].oper;
                switch (oper) {
                case '+': {
                    if (cal.size() >= 2) {
                        int x = cal.top(); cal.pop();
                        int y = cal.top(); cal.pop();
                        cal.push(x + y);
                    }
                    else {
                        this->flag = true;
                        throw std::string("The expression is wrong!");
                    }
                    break;
                }
                case'-': {
                    if (cal.size() >= 2) {
                        int x = cal.top(); cal.pop();
                        int y = cal.top(); cal.pop();
                        cal.push(y - x);
                    }
                    else {
                        this->flag = true;
                        throw std::string("The expression is wrong!");
                    }
                    break;
                }
                case'*': {
                    if (cal.size() >= 2) {
                        int x = cal.top(); cal.pop();
                        int y = cal.top(); cal.pop();
                        cal.push(x * y);
                    }
                    else {
                        this->flag = true;
                        throw std::string("The expression is wrong!");
                    }
                    break;
```

```
}
case'/': {
    if (cal.size() >= 2) {
        int x = cal.top(); cal.pop();
        int y = cal.top(); cal.pop();
        cal.push(y / x);
    }
    else {
        this->flag = true;
        throw std::string("The expression is wrong!");
    }
    break;
}
case'%': {
    if (cal.size() >= 2) {
        int x = cal.top(); cal.pop();
        int y = cal.top(); cal.pop();{
            if(x==0){
                this->flag=true;
                throw std::string("0 cannot be used as dividend!");
            }
        }
        cal.push(y % x);
    }
    else {
        this->flag = true;
        throw std::string("The expression is wrong!");
    }
    break;
}
case'^': {
    if (cal.size() >= 2) {
        int x = cal.top(); cal.pop();
        int y = cal.top(); cal.pop();
        if(y==0&&x<0){
            this->flag==true;
            throw std::string("The expression is wrong!");
        }
        cal.push(pow(y, x));
    }
    else {
        this->flag = true;
        throw std::string("The expression is wrong!");
```

```
                }
                break;
            }
            }
        }
    }
    ans = 0;
    if (cal.size() == 1) {
        ans += cal.top();
        cal.pop();
    }
    else {
        throw std::string("The expression is wrong!");
    }
  }
  catch (std::string str) {
    std::cout << str;
  }
}
```

3.3 总体系统的实现

3.3.1 总体系统流程图



3.3.2 总体系统核心代码

```
int main() {
    std::string str;
    std::string ch;
    while (true) {
        std::cout << "Please input expression：" << '\n';
        std::cin >> str;
        if (str[str.size() - 1] != '=') {
            std::cout << "The expression is missing '=', please re-enter！" << '\n' << '\n';
            continue;
        }
        std::string str_copy;
        for(int i=0;i<str.size();){
            if((str[i]=='-'||str[i]=='+')&&i>0&&(str[i-1]=='*'||str[i-1]=='^'||str[i-1]=='/'||str[i-
1]=='%')){
                int num=1;
                str_copy.push_back('(');
                str_copy.push_back('0');
                str_copy.push_back(str[i]);
                while(i+num<str.size()&&str[i+num]>='0'&&str[i+num]<='9'){
                    str_copy.push_back(str[i+num]);
                    num++;
                }
                if(num==1)
                    str_copy.push_back('0');
                str_copy.push_back(')');
                i+=num;
            }
            else{
                str_copy.push_back(str[i]);
                i++;
            }
        }
        str=str_copy;
        //std::cout<<str<<'\n';
        str.pop_back();
        std::cout << eval(str);
        std::cout << "Whether to continue（y,n）？";
        std::cin >> ch;
        while (std::cin.fail() || (ch != "n" && ch != "y")) {
            if (std::cin.fail()) {
                std::cin.ignore(INT_MAX, '\n');
                std::cin.clear();
            }
```

```
            std::cout << "Input errors, please re-enter:";
            std::cin >> ch;
        }
        if (ch == 'n')
            break;
    }
}
```

3.3.3 总体系统截屏示例