程序设计基础Calculator自主作业

姓名：诸俊豪 学号：2306010422 班级：计算机23\_4

科学计算器的实现

Calculator.h

#ifndef CALCULATOR\_H  
#define CALCULATOR\_H  
  
#include <stack>  
#include <string>  
  
using namespace std;  
  
// 计算器类  
class Calculator {  
public:  
 int Calculator::getnums(); //获取合法数字  
 bool Valid(const string &exp); //用于判断getstr的后缀表达式是否合法(目前仅实现判断非法字符)  
 string getstr(); //获取一个字符串  
 bool isOperator(char c); //判断是否是运算符，用于转换后缀表达式的重要逻辑  
 int precedence(char c); //定义优先级  
 string format(const std::string &infix); //格式化字符串，将空格删除，将负数添加负号，替换单元函数  
 string Postfix(const std::string &infix); //转换为后缀表达式  
 double calPostfix(const std::string &postfix); //计算后缀表达式  
 void menu(); //主菜单  
 void menu1(); //基础计算器菜单  
private:  
};  
  
#endif // CALCULATOR\_H

Calculator.cpp

//  
// Created by Walker on 2023/10/30.  
//  
#include <iostream>  
#include <stack>  
#include <string>  
#include <cstdlib> // atof 将字符串转化为浮点数  
#include <sstream>  
#include <regex> //正则表达式  
#include <bitset>  
#include "Calculator.h"  
  
double NaN = nan("NaN");  
  
bool Calculator::isOperator(char c) {  
 return c == '+' || c == '-' || c == '\*' || c == '/' || c == 'z' || c == 'y' || c == 'x' || c == 'w' || c == 'v' ||  
 c == 'u' || c == 't' || c == 's' || c == 'r';  
}  
  
int Calculator::precedence(char c) {  
 if (c == 'z' || c == 'y' || c == 'x' || c == 'w' || c == 'v' || c == 'u' || c == 't' || c == 's' || c == 'r')  
 return 3;  
 if (c == '+' || c == '-') {  
 return 1;  
 } else if (c == '\*' || c == '/') {  
 return 2; //乘除优先级更高  
 }  
 return 0;  
}  
  
int Calculator::getnums() {  
 int input;  
 if (scanf("%d", &input) != 1) {  
 printf("%s\n", "Invalid input, please re-enter.\n");  
 while (getchar() != '\n'); // 清除输入缓冲区  
 return getnums(); // 递归调用自身，重新获取输入  
 }  
 return input;  
}  
  
string Calculator::getstr() {  
// cout<<"begin"<<endl; //重要测试  
 string str;  
 getline(cin, str);  
// cout << "这是得到的字符串:" << str << endl; //重要测试  
// cout<<"下面是Valid的返回值"<<endl; //重要测试  
// cout<<Valid(str)<<endl; //重要测试  
 if (!Valid(str)) {  
 cout << "Invalid characters, please re-enter.\n" << endl;  
// cout<<2<<endl; //重要测试  
 return getstr(); // 递归调用自身，重新获取输入  
 }  
// cout<<"end"<<endl; //重要测试  
 return str;  
}  
  
bool Calculator::Valid(const string &exp) {  
 // 空字符串不是合法输入，直接返回 false  
 if (exp.empty()) {  
 return false;  
 }  
 bool Valid = true;  
// 判定字符是否定义  
// 给定字符数组  
 char charArray[] = {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '+', '-', '\*', '/', '(', ')', '.', ' ', 's',  
 'i', 'n', 'c', 'o', 'a', 't', 'l', 'q', 'r', 'g'};  
 // 创建一个位向量，用于存储字符数组中的字符  
 bitset<256> charVector;  
 // 设置位向量中的位，表示字符数组中的字符存在  
 for (char c: charArray) { charVector.set(static\_cast<unsigned char>(c)); }  
 // 遍历字符串中的每个字符，并检查位向量中的相应位  
 for (char c: exp) {  
 if (!charVector.test(static\_cast<unsigned char>(c))) {  
 Valid = false;  
 break;  
 }  
  
 }  
 return Valid;  
}  
  
  
string Calculator::format(const string &ori\_infix) {  
 string ori\_infix\_0 = ori\_infix;  
 // 去除空格  
 regex reg\_0("\\s+"); // 匹配一个或多个连续的空格  
 ori\_infix\_0 = regex\_replace(ori\_infix\_0, reg\_0, "");  
  
 // 函数替换逻辑  
 string Funcs[9] = {"sin", "cos", "tan", "asin", "acos", "atan", "ln", "lg", "sqrt"};  
 char Replacements[9] = {'z', 'y', 'x', 'w', 'v', 'u', 't', 's', 'r'};  
 for (size\_t i = 0; i < 9; i++) {  
 string temp = Funcs[i];  
 regex regex(temp);  
 ori\_infix\_0 = regex\_replace(ori\_infix\_0, regex, string(1, Replacements[i]));  
 }  
  
// 负数的处理  
 if (ori\_infix\_0[0] == '-') { ori\_infix\_0 = ori\_infix\_0.insert(0, 1, '0'); }//处理首位是负号  
 regex reg("\\((-[0-9]+([.][0-9]+)?)"); //处理左括号后是负号  
 string ori\_infix\_1 = regex\_replace(ori\_infix\_0, reg, "((0$1)");  
 string infix = ori\_infix\_1;  
  
 return infix; //等待更多实现  
}  
  
string Calculator::Postfix(const string &ori\_infix) {  
 string infix = format(ori\_infix);  
 stack<char> s; // 操作符栈  
 string postfix; // 后缀表达式存放  
 for (char c: infix) {  
 if (isdigit(c) || c == '.') { // 读取字母或数字  
 postfix += c;  
 continue;  
 }  
 postfix += ' '; // 添加空格便于后期计算  
 if (isOperator(c)) {  
 // 栈非空而且栈顶操作符优先级高（先判断栈是否空，否则不存在栈顶）  
 while (!s.empty() && s.top() != '(' && precedence(s.top()) >= precedence(c)) {  
 postfix += s.top();  
 s.pop();  
 postfix += ' '; // 添加空格便于后期计算  
 }  
 s.push(c);  
  
 } else if (c == '(') { // 左括号直接压入栈中  
 s.push(c);  
// flag2 = -1;  
 } else if (c == ')') { // 收取中间所有操作符  
 while (!s.empty() && s.top() != '(') {  
 postfix += s.top();  
 s.pop();  
 postfix += ' '; // 添加空格便于后期计算  
 }  
 if (!s.empty()) s.pop(); // 弹出左括号  
 }  
 }  
 postfix += ' '; // 添加空格便于后期计算  
 // 弹出剩余优先级低的操作符  
 while (!s.empty()) {  
 postfix += s.top();  
 s.pop();  
 if (!s.empty()) { // 仅当栈中还有元素的时候添加空格  
 postfix += ' ';  
 }  
 }  
 return postfix;  
}  
  
double Calculator::calPostfix(const string &postfix) {  
 istringstream input(postfix); //使用输入流读取后缀表达式的所有字符  
 stack<double> s;  
 string token;  
 while (input >> token) { //从输入流读取字符  
 if (isOperator(token[0])) {//判断读取到的这个字符是否为操作符  
 if (s.empty()) return NaN;  
 double opn2 = s.top();  
 s.pop();  
 if (token == "+") {  
 if (s.empty()) return NaN;  
 double opn1 = s.top();  
 s.pop();  
 s.push(opn1 + opn2);  
 } else if (token == "-") {  
 if (s.empty()) return NaN;  
 double opn1 = s.top();  
 s.pop();  
 s.push(opn1 - opn2);  
 } else if (token == "\*") {  
 if (s.empty()) return NaN;  
 double opn1 = s.top();  
 s.pop();  
 s.push(opn1 \* opn2);  
 } else if (token == "/") {  
 if (s.empty()) return NaN;  
 double opn1 = s.top();  
 s.pop();  
 s.push(opn1 / opn2);  
 } else if (token == "z") {  
 s.push(sin(opn2));  
 } else if (token == "y") {  
 s.push(cos(opn2));  
 } else if (token == "x") {  
 s.push(tan(opn2));  
 } else if (token == "w") {  
 s.push(asin(opn2));  
 } else if (token == "v") {  
 s.push(acos(opn2));  
 } else if (token == "u") {  
 s.push(atan(opn2));  
 } else if (token == "t") {  
 s.push(log(opn2));  
 } else if (token == "s") {  
 s.push(log10(opn2));  
 } else if (token == "r") {  
 s.push(sqrt(opn2));  
 }  
 } else {  
 s.push(atof(token.c\_str())); //将字符串转化为浮点数，并压入栈中。  
 }  
 }  
 return s.top();  
}  
  
void Calculator::menu1() {  
  
// string exp;  
 cout << "支持数据类型，整数，负数，小数\n暂时支持二元操作符+-\*/,\n"  
 "暂时支持单元操作符{\"sin\", \"cos\", \"tan\", \"asin\", \"acos\", \"atan\", \"ln\", \"lg\", \"sqrt\"}\n"  
 "单元操作符需要以op()的格式输入，例如sin(1),不可以是sin1.(三角函数以弧度制输入)\n"  
 "本次输入支持非法字符检测，空格自动删除\n";  
 cout << "输入中缀表达式: " << endl;  
 cin.ignore(); // 添加此行以清除换行符!!!!非常重要，否则会访问空指针。  
 string exp = getstr();  
 clock\_t time\_0 = clock();  
 string postfixexp = Postfix(exp);  
 clock\_t time\_1 = clock();  
 cout << "中缀表达式中支持的单元运算符号在后缀表达式中存在这样的映射关系:\n"  
 "{\"sin\t\", \"cos\t\", \"tan\t\", \"asin\", \"acos\", \"atan\", \"ln\t\", \"lg\t\", \"sqrt\"}\n"  
 "{'z\t', 'y\t', 'x\t', 'w\t', 'v\t', 'u\t', 't\t', 's\t', 'r\t'}\n"  
 "后缀表达式是: " << postfixexp << endl;  
 system("pause"); //测试  
  
 clock\_t time\_2 = clock();  
 double result = calPostfix(postfixexp);  
 clock\_t time\_3 = clock();  
 if (isnan(result))  
 cout << "Input format error.\n";  
 else {  
 cout << "计算结果是: " << result << endl;  
 cout << "转化后缀表达式花费时间为: " << (double) (time\_1 - time\_0) / CLOCKS\_PER\_SEC << "\n后缀表达式计算花费时间为:"  
 << (double) (time\_3 - time\_2) / CLOCKS\_PER\_SEC << "\n\n\n";  
 }  
 system("pause");  
}  
  
void Calculator::menu() {  
 setbuf(stdout, NULL);  
 int flag = 1;  
 while (flag) {  
 cout << "欢迎使用这个简陋的计算器!\n请用英文字符输入";  
 cout << "你可以选择\n0.退出\n1.基础计算器\n";  
 cout << "你的选择是(支持大多数非法输入处理):\n";  
 int choice = getnums();  
 switch (choice) {  
 case 0 :  
 flag = 0;  
 cout << "感谢使用.\n已退出..." << endl;  
 system("pause");  
 break;  
 case 1:  
 menu1();  
 break;  
 default:  
 cout << "Function not available.\n" << endl;  
 break;  
 }  
 }  
  
}

main.cpp

#include "Calculator.h"  
  
int main() {  
 Calculator cal;  
 cal.menu();  
 return 0;  
}

输出结果如下 ：





