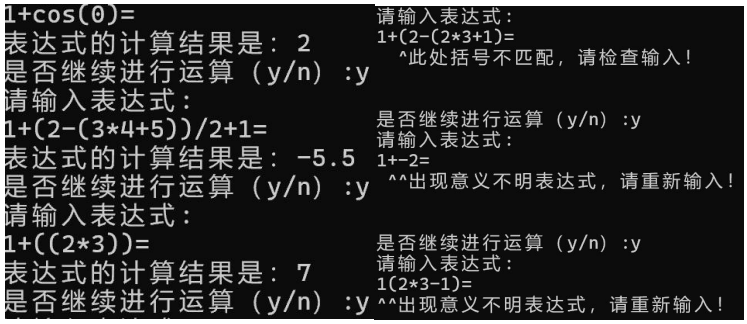
四川大学计算机学院（软件学院、智能科学与技术学院）

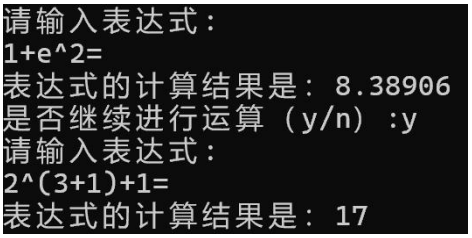
实验报告

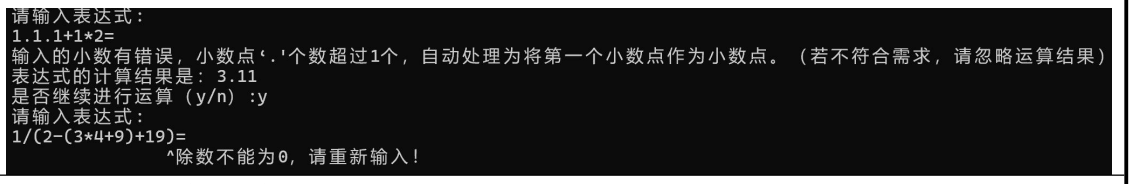
学号：2023141460321 姓名：孙谦昊 专业：计算机科学与技术 班级：行政七班 第 6 周

课程名称	数据结构与算法分析课程设计	实验课时	第 1 次
实验题目	算数表达式求值	实验时间	2024 年 10 月 8 日
实验目的和要求	实验基本要求： 1. 采用算符优先级算法，能正确求表达式的值; 2. 熟练掌握栈的应用; 3 .熟练掌握计算机系统的基本操作方法，了解如何编辑、编译、链接和运行一个 C++ 程序; 4. 上机调试程序，掌握查错、排错使程序能正确运行 实验扩展： 1.多位数字 2.浮点数输入 3.支持单目运算符 4.多重、多种括号括号匹配 5.连续运算符检测 6.表达式合法性判断 7.非数字输入检测 8.科学计算		
实验环境	12th Gen Intel(R) Core(TM) i7-1260P 2.10 GHz Visual Studio 2022 Debug x64 本地 Windows 调试器		
算法描述	本程序采用堆栈的方式实现了四则混合运算计算器的功能，具体实现方式如下： 进入程序，输入表达式。 完成表达式输入后，在 main.cpp 中的 bool isRight(string s)函数中，借助堆栈的方式首先对输入的表达式进行括号匹配运算。若匹配则进入 Calculator::expRun()函数中进行相关运算，反之则输出错误并要求用户重新输入表达式。 在 Calculator 类中，初始化私有成员 NumStack 和 OpeStack 堆栈，随后调用构造函数 Calculator()或 Calculator(string exp)。接下来调用Calculator::expRun()函数进行四则混合运算以及错误情况判断。 Calculator::expRun()函数中，确保 NumStack 堆栈遵循“ANANA…”规则——A 代表除括号以外的运算符，N 代表 char 类型的数字，以便区分运算符与数字之间的位置关系。确保 OpeStack 堆栈遵循“不用则存，用则取”的规则，确保运算符在正确时机出栈参与运算。进行四则混合运算时，遵循“优先级高或相同级的立即进行运算”的原则，先定义 Calculator::opeGrade()运算符等级函数并根据运算符返回对应等级，随后判断优先级后决定是否立即进行运算。若前一运算符 opeBefore 等级小于当前运算符 opeCh 等级，则 opeBefore 入栈，更新 opeBefore=opeCh。若前一运算符 opeBefore 等级大于等于当前运算符 opeCh 等级，NumStack 堆栈中出两个 N，完成运算后放入 NumStack，更新运算符。在 while(opeGrade(opeCh) <= opeGrade(opeBefore) && !(opeBefore == '=' && opeBefore == opeCh) && !(opeBefore == '(' && opeCh == ')'))循环中重复这一步，直至若前一运算符 opeBefore 等级小于当前运算符 opeCh 等级或其他情况。 注意，因为本程序的设计思路在调用 Calculator::expRun()函数前已经完成了括号匹配操		

	作。所以遇到 opeBefore == '(' && opeCh == ')'时直接将括号配对删除即可。 对于在四则运算外额外实现的功能，针对自然对数 e，监测到表达式 expressioon[i] == 'e' 后将 e 的近似值 2.718281828459045 压入 NumStack 中作为替代。针对三角函数，借助字符串匹配提取的方式提出三角函数括号中的表达式，并将其定义为一个 string b，例如在 sin(1+9)中 b 提取为 0+1+9=，在 sin(9)中 b 提取为 0+9=，随后将 b 作为输入嵌套调用 Calculator<char> CalTri(b);进行运算，返回运算结果后借助#include <cmath>中的 sin()等函数计算出对应数值后立即压入 NumStack 中作为一个数。 至于错误检测，本程序采用的策略是“变读取变错误判断”，借助表达式 expression[i]，expression[i+1]和 expression[i-1]之间的关系进行综合判断。如果有错误立即终止运算并标记处错误位置，随后返回。
源程序清单	1.main.cpp，用于主程序的调用 2.Stack.h，用于创建栈 3.Calculator.h，表达式计算器
运行结果	一、程序可运行功能 程序功能：1.可以完成正常的四则运算混合运算，例如:1+(1-(2*3+1))= 2.支持乘方运算，例如：2^3= 3.支持支持双精度浮点数运算，例如：1.1+1.1= 4.支持自然对数 e 运算，例如 e^2= 5.支持三角函数 sin,cos,tan,cot 运算,但不能嵌套。例如 sin(9)+cos(9)+tan(9) 6.支持括号匹配检测和指出错误功能,例如： 10+(10*8+1))= ^ 此处括号不匹配，请检查输入！ 1+((2))3= ^^出现意义不明表达式，请重新输入！ 1 + 2(2 + 1) = ^^出现意义不明表达式，请重新输入！ 7.非法除 0 运算检测，例如： 1+1/0= ^ 除数不能为 0，请重新输入！ 1+2/(1-1)= ^ 除数不能为 0，请重新输入！ 8.支持小数点运算检测。若超过一个则启动默认运算 1.1.1*2+1=3.11 输入的小数有错误，小数点‘.'个数超过 1 个 二、程序实际运行结果







实验运行情况分析（包括算法、运行结果、运行环境等问题的讨论）

一、程序运行情况分析

本程序采用的算法运行出的结果在输入正确的表达式时能够正确完成运算，在输入错误表达式时能够准确指出相关错误在哪里以及错误原因，在输入意义不明表达式时若能进行默认处理则进行正常运算，反之则指出相关错误在哪里以及错误原因。

不过本程序也有相关不足，例如在求解三角函数时无法进行例如 sin(sin(9))这样的嵌套类型的三角函数运算或在使用 tan()时无法精确判断 π/2，仅能输出一很大的正数或很小的负数；在计算-1+1=这种运算时无法根据人类书写与阅读习惯直接计算，程序会输出错误，必须要以(-1)+1=这样的输入才能运算。

二、运行环境讨论

本程序的编写、编译、运行和调试皆是借助 Visual Studio 2022 Debug x64 本地 Windows 调试器。若将本程序使用 DEV C++等软件上编译、运行和调试时可能报错，而这是因为各软件对于支持的 C++版本不同所导致的。例如在 Visual Studio 2022 中可以使用 nullptr，而在部分 DEV C++中仅能使用 NULL。

本程序因为采取的写作思路与事例程序有一定差别，因此代码长度显著长于事例代码。借助 Visual Studio 2022 Debug x64 本地 Windows 调试器强大的功能能够快速编译运行，而在部分 DEV C++上则速度较慢。表明程序仍有一定优化空间。

三、个人心得

在课上听老师讲述运算过程时以为自己听明白了，可是实际上手时发现问题很多。最开始时我想根据事例代码进行简单调整后就完成本次实验，不过发现事例代码存在一定的问题并且设计思路是直接开始进行运算，若有错误直接终止但不具体表明错误位置和错误原因，于是我打算自己从零完成这样一个程序。

在最开始编写程序时，我首先确定了设计思路：先判断括号匹配再进行运算。随后，我开始考虑 1+2=和 1+2*3+1=这样的简单四则混合运算。不过由于最开始采用的设计思路问题，在运算 1+2*3+1=时输出的结果始终是 7，这是因为我没有添加如现在代码所示的 Calculator::expRun()中的 while (opeGrade(opeCh) <= opeGrade(opeBefore) && !(opeBefore == '=' && opeBefore == opeCh) && !(opeBefore == '(' && opeCh == ')'))循环，即没办法回头运算表

	达式最开头的 1+。添加上 while 循环后，我开始考虑括号情况。从 1+(2*3+1)=一步步到 1+(2-(3*4+5)-10)/2+3=这样复杂的，多重括号进行嵌套的情况。能够解决四则运算后，开始对可能的错误输入的情况进行检测和输出，最后再加上关于自然对数 e 和三角函数 sin，cos，tan 和 cot 的运算。 整体而言自己编写这样一个程序是富有挑战性的，一共花了大概三个下午和晚上进行编写、调试。不过最终的结果是非常符合人的期待的。 四、其他 在编写完成程序后，邀请了同学们帮忙测试程序。期间也发现了许多 Bug，后面都逐一进行了相关修正。 在编写程序过程中的相关版本代码发布已经在 Github 中，链接为： https://github.com/StrayerSQH/Learning/tree/main/%E6%95%B0%E6%8D%AE%E7%BB%93%E6%9E%84%E4%B8%8E%E7%AE%97%E6%B3%95%E5%88%86%E6%9E%90%E5%AE%9E%E9%AA%8C%E8%AF%BE/%E5%AE%9E%E9%AA%8C1
指导老师评议	成绩评定：　　指导教师签名：