《数据结构》课程实践报告

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 院、系 | 计算机学院 | | 年级专业 | 大二 软工 | 姓名 | 丁彦君 | 学号 | 2329403126 |
| 实验布置日期 | | 2024/10/07 | | 提交  日期 | 2024/10/16 | | 成绩 |  |

课程实践实验3：栈的实现及应用

## 一、问题描述及要求

**1. 实现自己的顺序栈类**

**2. 完成中缀表达式求值**

**要求中缀表达式以一个字符串的形式读入，可含有加减乘除运算和左右括号，并假设该表达式以“#”作为输入结束符。**

**3. 完成后缀表达式求值**

**（1）输入带#引导符的后缀形式**

**（2）输入带空格的后缀表达式**

**4.完成前缀表达式求值**

**从右到左扫描表达式**

## 二、概要设计

**（1）对实验内容的理解。**

**实验内容是通过编写程序来实现对前缀、中缀和后缀表达式的求值。中缀表达式通常是人类书写数学表达式的方式，而后缀表达式（逆波兰表达式）则更适合计算机处理。通过这个实验，可以加深对栈结构的理解。**

**（2）系统的功能列表**

**系统的主要功能如下：**

**用户输入表达式，系统能够根据选择处理中缀或后缀表达式。**

**中缀表达式的求值（支持整数与浮点数）。**

**后缀表达式的求值（支持整数与浮点数）。**

**前缀表达式的求值（支持整数与浮点数）**

**运算符优先级判断功能。**

**提示用户选择计算表达式的类型。**

**（3）程序运行的界面设计**

**1.提示用户选择表达式类型：用户在命令行界面看到提示，并选择输入"1"或"2"。**

****

**2.输入表达式：用户根据选择输入中缀或后缀表达式，按 # 结束输入。**

****

****

****

**3.显示结果：程序计算并显示表达式的结果。**

****

****

****

**（4）总体设计思路**

**数据结构：**

**栈：用于存储操作数和操作符，栈是后进先出的数据结构，适合处理表达式求值。**

**一个栈 values 用来存储操作数。**

**一个栈 ops 用来存储操作符。**

**设计方法：**

**对于中缀表达式，程序逐字符扫描表达式：**

**1.遇到数字将其存入操作数栈。**

**2.遇到左括号(将其入栈) 遇到右括号则出栈并进行计算直到匹配左括号。**

**3.遇到操作符时，比较优先级决定是否出栈计算或直接入栈。**

**后缀表达式：**

**逐个扫描并直接进行操作数和操作符的计算。**

**前缀表达式：**

**从右到左扫描表达式然后和后缀表达式处理方式相同。**

**（5）程序结构设计**

**主程序文件：main.cpp**

**核心算法文件：处理中缀和后缀表达式的求值方法在 func1和func2 函数中实现，处理前缀表达式的求值方法在prefix中实现。**

**类与对象：Stack.h**

## 三、详细设计

**主函数设计**

**1.提示用户选择表达式类型。**

**2.根据用户输入调用不同的函数进行计算：**

**func1 用于中缀表达式求值。**

**func2 用于后缀表达式求值。**

**Prefix用于前缀表达式求值**

**3.显示计算结果。**

**类设计**

**SeqStack类:**

**SeqStack(); //初始化**

**~SeqStack(); //析构释放**

**isEmpty(); //判断栈是否为空**

**isFull(); //判断栈是否已满**

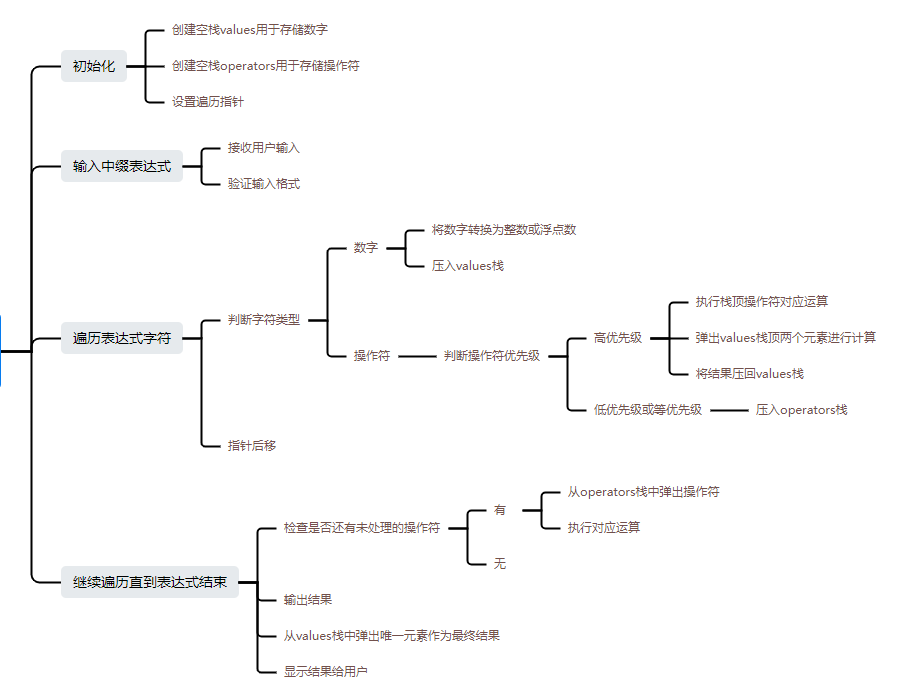
**push(); //入栈**

**pop(); //出栈**

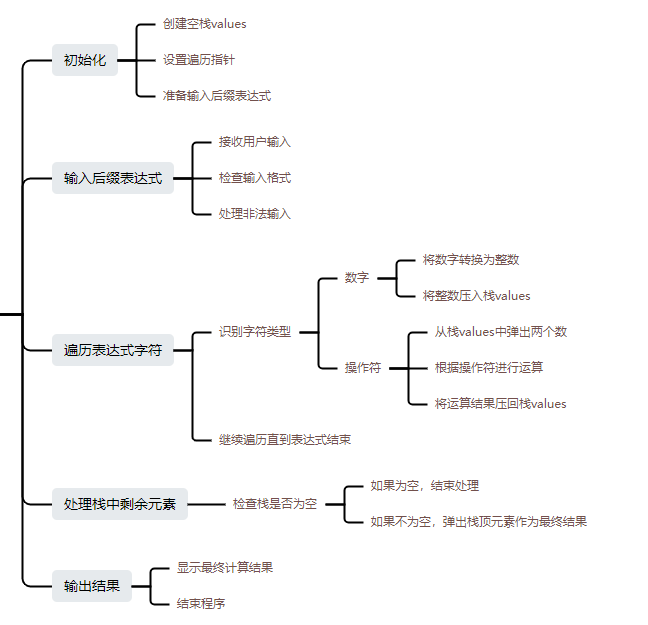
**peek(); //找出栈顶值**

**关键算法设计**

**中缀表达式求值算法流程图：**

****

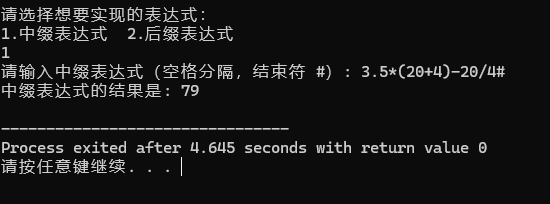
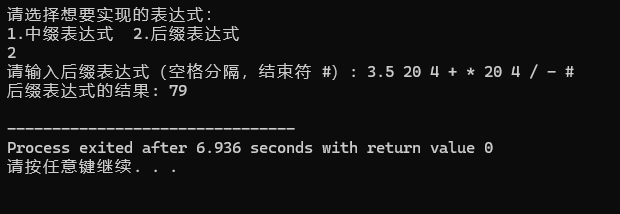
**后缀表达式求值算法流程图：**

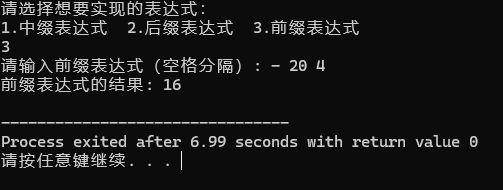
****

**前缀表达式求值算法流程图：**

****

## 四、实验结果





## 五、实验分析与探讨

**1. 测试结果分析**

**针对前缀，中缀和后缀表达式的求值功能，进行了多个测试用例，验证了程序在处理整数、小数、四则运算和括号优先级时的正确性。**

**时间复杂度分析：**

**对于中缀表达式求值，程序需要遍历每个字符，对操作数入栈以及根据操作符优先级进行相应计算。每个操作符会触发从栈中弹出相应的操作数，进行计算后再入栈。因此整个求值过程的时间复杂度为 O(n)，其中 n 是表达式的长度。同样，对于前后缀表达式的求值，由于无需判断操作符的优先级，直接遍历表达式并进行操作数的弹栈和计算，时间复杂度也是 O(n)。**

**空间复杂度分析：**

**空间复杂度主要由栈的数据存储决定。对于每个操作数和操作符都需要使用栈暂存，因此空间复杂度也是 O(n)。**

**2. 问题与解决方法**

**（1）小数运算问题：原始程序仅支持整数运算，无法处理浮点数。通过将数值处理部分修改为 double 类型，以及改进数值解析逻辑，支持带小数点的数字处理，在读取浮点数时增加了对小数点的判断和处理，使得程序能够正确地解析和运算诸如 3.5 \* 15 之类的表达式。**

**（2）操作符优先级处理问题：在中缀表达式中，操作符的优先级需要严格处理，乘法和除法优先于加法和减法。通过定义一个 judge() 函数，明确了不同操作符的优先级，并在解析过程中根据优先级决定是否立即进行运算，确保了运算顺序的正确性。**

**（3）括号优先级处理问题：对于中缀表达式中的括号，程序在遇到右括号时，需要依次将栈中的操作符弹出并进行计算，直到遇到左括号为止。这一过程确保了括号内部的优先计算。该部分逻辑通过栈结构得到了有效的实现。**

**（4）前后缀表达式的转换问题：在将后缀表达式转换成前缀表达式时，反转字符串时没有先将其转化成一个一个的小块然后再逆序，这样才不会使多位数字转换后值发生改变，使得结果能正确输出。**

**3. 优化与扩展探讨**

**错误处理机制改进：当前程序对于用户输入的非法表达式没有完善的处理机制。可以进一步加强错误处理，例如检测括号是否匹配、非法字符的输入等，提高程序的鲁棒性。**

**4. 实验设计中的难点**

**（1）浮点数处理：在处理中缀表达式时，如何正确解析带小数点的数字并将其存入栈中是设计的一个难点。尤其是在解析过程中，必须小心处理小数点后的部分，确保在读取和运算时精度不丢失。通过改进数值读取逻辑，解决了这一问题。**

**（2）操作符优先级的处理：另一个难点是如何在扫描过程中动态判断操作符的优先级，并确保在适当时机进行运算。通过使用栈来存储操作符，并在适当时根据优先级出栈进行计算，有效解决了这一问题。**

## 六、小结

**通过本次实验，掌握了使用栈结构实现表达式求值的技巧，解决了浮点数处理和操作符优先级等难点问题。实验过程中不断测试和调整算法，最终实现了稳定、正确的前缀、中缀和后缀表达式求值功能。未来可以进一步扩展表达式的处理能力，完善错误处理机制，提高程序的健壮性和实用性。**

## 附录：源代码

1. **实验环境：Dev-C++**
2. **代码见附件**