# 20220908 数据结构与算法 解题报告

## 后缀表达式计算

### 题目大意

给定一条包含小写字母的后缀表达式，计算他的值。其中

### 字符—数值转换

对于

的转换，我们可以利用C++的字符对应ASCII码运算

double tmp = ch - 'a' + 1;

### 表达式计算

分析后缀表达式计算的特点，考虑使用栈结构。步骤如下：

1⃣️ 若读入字母，将其转换为数值，压入栈中。

2⃣️ 若读入操作符，将栈顶两元素弹出并进行计算，再将结果压入栈中。

3⃣️ 待到表达式串遍历完毕，栈顶剩下的唯一元素即为结果。

由于给定表达式必定合法，所以不需要考虑爆栈的问题。

### 代码

#include <bits/stdc++.h>  
using namespace std;  
  
int main(int argc, char const \*argv[]) {  
 string s;  
 for (; cin >> s;) {  
 stack<double> stk;  
 for (int i = 0; i < s.length(); i++) {  
 if (s[i] >= 'a' && s[i] <= 'z') stk.push(s[i] - 'a' + 1);  
 else if (s[i] == '+') {double tmp = stk.top(); stk.pop(); tmp = stk.top() + tmp; stk.pop(); stk.push(tmp);}  
 else if (s[i] == '-') {double tmp = stk.top(); stk.pop(); tmp = stk.top() - tmp; stk.pop(); stk.push(tmp);}  
 else if (s[i] == '\*') {double tmp = stk.top(); stk.pop(); tmp = stk.top() \* tmp; stk.pop(); stk.push(tmp);}  
 else if (s[i] == '/') {double tmp = stk.top(); stk.pop(); tmp = stk.top() / tmp; stk.pop(); stk.push(tmp);}  
 }  
 printf("%.2lf\n", stk.top());  
 }  
 return 0;  
}

## 中缀表达式转后缀表达

### 题目大意

给定包含字母的中缀表达式，将其转换为后缀表达式。

### 转换方法

分析中缀表达式，一共有三个优先级：括号>乘除>加减，并且按照从左往右的顺序计算。又由后缀表达式的特点，单个子后缀表达式自成一体，故可作为整体参与到其余后缀表达式计算。

### 代码

#include <bits/stdc++.h>  
using namespace std;  
  
int pos;  
string s;  
  
string Transfer() {  
 stack<string> stk, opt;  
 if (s[pos] == '(') {  
 pos++; stk.push(Transfer());  
 }  
 else {  
 stk.push(s.substr(pos, 1)); pos++;  
 }  
 for (; s[pos] != ')';) {  
 opt.push(s.substr(pos, 1));  
 if (s[pos + 1] == '(') {  
 pos += 2;  
 stk.push(Transfer());  
 }  
 else {stk.push(s.substr(pos + 1, 1)); pos += 2;}  
 for (; !opt.empty() && (opt.top() == "\*" || opt.top() == "/"); opt.pop()) {  
 string tmp = stk.top(); stk.pop();  
 tmp = stk.top() + tmp + opt.top(); stk.pop();  
 stk.push(tmp);  
 }  
 }  
 pos++;  
 string tmp = "";  
 for (; !opt.empty(); opt.pop(), stk.pop())  
 tmp = stk.top() + opt.top() + tmp;  
 return stk.top() + tmp;  
}  
  
int main(int argc, char const \*argv[]) {  
 cin >> s; s += ")";  
 // s = "a+((b-c)\*d+e)/f)";  
 cout << Transfer() << endl;  
 return 0;  
}

## Bracket Matching

### 题目大意

给定一个括号字符串，判断其是否合法。

### 匹配过程

由于只需要判断括号是否相互匹配，故可忽略除括号外的所有字符。

将左括号视作压栈操作，右括号视作查询/弹栈操作。

### 代码

#include <bits/stdc++.h>  
using namespace std;  
  
bool Judge(string s) {  
 stack<char> stk;  
 for (int i = 0; i < s.length(); i++) {  
 if (s[i] == '(' || s[i] == '[' || s[i] == '{')  
 stk.push(s[i]);  
 if (s[i] == ')')  
 if (!stk.empty() && stk.top() == '(')  
 stk.pop();  
 else  
 return false;  
 if (s[i] == ']')  
 if (!stk.empty() && stk.top() == '[')  
 stk.pop();  
 else  
 return false;  
 if (s[i] == '}')  
 if (!stk.empty() && stk.top() == '{')  
 stk.pop();  
 else  
 return false;  
 }  
 if (!stk.empty()) return false;  
 return true;  
}  
  
int main(int argc, char const \*argv[]) {  
 string s;  
 for (; cin >> s;)  
 cout << (Judge(s) ? "Yes" : "No") << endl;  
 return 0;  
}