山东大学 计算机科学与技术 学院

数据结构与算法 课程实验报告

学号：202300130183	姓名： 宋浩宇	班级：23 级人工智能班
实验题目：2024 级数据结构--数据/智能 实验 6 栈		
实验学时：2	实验日期： 2024/10/23	

实验目的：

1. 掌握栈结构的定义与实现；
2. 掌握栈结构的使用。

软件开发工具：

1. visual studio code 2022（使用 C/C++、C/C++ Extension Pack、C/C++ Themes 插件）
2. mingw64 工具包

1. 实验内容

完成 2024 级数据结构--数据/智能 实验 6 栈 A ： 计算表达式

需要实现栈类，完成 push(value)，pop()，top()方法

并且借助自己写的 stack 类模板来解决计算表达式的问题

2. 数据结构与算法描述 （整体思路描述，所需要的数据结构与算法）

使用的数据结构为：栈

关于解题的算法：首先对于输入的表达式，有两种处理思路，一种是按照输入，也就是中缀表达式直接计算，另一种是先将中缀表达式转化为后缀表达式（波兰式）来对后缀表达式进行计算，这两种计算过程都需要使用栈数据结构来存储操作符和数据。

在此先阐述将中缀表达式转化为后缀表达式的方式。

首先我们需要一个栈用于临时存储数字和运算符。

然后我们遍历这个中缀表达式，只要遵循以下原则，就可以将中缀表达式转化为后缀表达式：

①如果是数字，直接加入后缀表达式

②如果是（，直接压入栈

③如果是），弹出栈到后缀表达式直到遇到（

④如果是操作符，如果栈顶运算符的优先级小于当前运算符的优先级，或者栈为空或者栈顶是（，则直接压入栈

⑤否则，弹出栈直到栈顶运算符优先级小于当前运算符优先级，或者遇到（，然后将当前运算符压入栈

在完成整个中缀表达式的遍历后，再将栈中剩余的表达式弹出到后缀表达式中。经过以上处理，我们可以得到一个存放着完整后缀表达式的栈。

之后我们只需要在对这个后缀表达式进行读取即可计算出答案。这个读取过程遵循以下原则：

①如果读到数字，直接压入结果栈

②如果读到运算符，则取出结果栈顶上的两个数字进行运算，再将运算结果压入结果栈

在按照以上方式遍历完这个后缀表达式之后，结果栈中唯一剩余的数值就是这个后缀表达式计算的结果，之后我们再把这个结果按照题目要求的保留两位小数输出出来即可。

另一种解法，即直接按照中缀表达式来计算。

这种解法的核心思路是，在读取时将括号里的数据直接计算出结果，最后再计算一个不带括号的中缀表达式。因此在遍历这个中缀表达式的时候，只要遵循以下原则：

①如果时数字，则压入数值栈

②如果是运算符，则压入运算符栈（包括括号“（”）

③如果是“）”，则弹出运算符栈直到弹出“（”，再依据弹出的运算符对数值栈进行计算，实际就是计算一个没有括号的中缀表达式（因为在此处有这项操作，因此这个算法需要输入是绝对合法的才可以）。
按照以上原则遍历完这个中缀表达式后，我们可以得到一个没有括号的中缀表达式，这个时候我们只要复用之前使用的计算括号内中缀表达式的算法即可。
所以在理解了上述的核心思路后，我们只要完成不带括号的中缀表达式的计算方法即可。
在计算时，我们的主要问题有两点：运算符优先级问题、减法变号问题。
我们将计算这个表达式进行原子化处理，我们定义一次计算，这次计算会取出栈顶的运算符，取出两个数值栈栈顶的数字，依照此运算符进行处理。但是因为我们上述两个问题的存在，我们需要检测第二个栈顶运算符。分为以下几种情况（这几种情况中的压入栈顶指的都是压入数值栈栈顶）：
①如果我们取出的是*或/，而此时栈顶是+或-，这个时候可以直接进行计算，再将结果压入栈顶
②如果我们取出的是*或/，而此时栈顶是*或/，这个时候我们需要再执行一次原子化的运算，即使用递归的方式在调用一次计算，等栈顶的*或/被处理完之后再处理当前的运算符。具体方式为：将取出的第二个数压入栈顶，调用下一次计算，在下一次计算完成之后再取出栈顶的数字，继续进行原本的计算过程
③如果我们取出的是+或-，而此时栈顶是+，则直接进行计算，并将结果压入栈顶
④如果我们取出的是+或-，而此时栈顶是-，则改变当前被取出的运算符，如果是+则改为进行-运算，如果是-则改为进行+运算
⑤如果我们取出的是+或-，而此时栈顶是*或/，则进行和②一样的操作
⑥重复上述操作直至栈中只有一个元素
按照以上原则计算，我们就可以计算出一个不带括号的中缀表达式的结果，再按照以上所说的调用方式即可通过原子化（或叫递归）的思路成功解决这个问题。
3. 测试结果（测试输入，测试输出）
测试输入为：
3
1+6/1*7+2*1*4+9/1+2*0*9+9+7/(9*5)-1*6-0*8-7-9*2+6-(0-5-2*8-7-9*5*(6-5*5*2*6-2-7-5+6*7+6*9-1*0*0+3*0+2/1-6/6+5))
0-4-1/6*(1-(6/7)-4+6+2+6*1)-1*7+2-8*2+0-(4+6-6*1+(3-8*6/4-6-5)*6/4/8+7-1*4/9*5)-0/6+1-0-2+7-2+6*4-3*6+2/8+6+1*6*2
5-3*9+5/1*5-9+1*8-6-8-4*1+5-2+9/3*2-2/5/(2-6)*2/7-9*0-2+4/6*6*7*8-8-8*6+8*9*(3+0*1/5/2*7*8+0-8*8-5+8/5*2-0)
输出为：
-9197.84
-3.47
-4362.57
附上以上三个中缀表达式转化出来的后缀表达式：
1 6 1 / 7 * + 2 1 * 4 * + 9 1 / + 2 0 * 9 * + 9 + 7 9 5 * / + 1 6 * - 0 8 * - 7 - 9 2 * - 6 + 0 5 - 2 8 * - 7 - 9 5 * 6 5 5 * 2 * 6 * - 2 - 7 - 5 - 6 7 * + 6 9 * + 1 0 * 0 * - 3 0 * + 2 1 / + 6 6 / - 5 + * - -

0 4 - 1 6 / 1 6 7 / - 4 - 6 + 2 + 6 1 * + * - 1 7 * - 2 + 8 2 * - 0 + 4 6 + 6 1 * - 3 8 6 * 4 / - 6 - 5 - 6 * 4 / 8 / + 7 + 1 4 * 9 / 5 * - - 0 6 / - 1 + 0 - 2 - 7 + 2 - 6 4 * + 3 6 * - 2 8 / + 6 + 1 6 * 2 * +

5 3 9 * - 5 1 / 5 * + 9 - 1 8 * + 6 - 8 - 4 1 * - 5 + 2 - 9 3 / 2 * + 2 5 / 2 6 - / 2 * 7 / - 9 0 * - 2 - 4 6 / 6 * 7 * 8 * + 8 - 8 6 * - 8 9 * 3 0 1 * 5 / 2 / 7 * 8 * + 0 + 8 8 * - 5 - 8 5 / 2 * + 0 - * +

4. 分析与探讨（结果分析，若存在问题，探讨解决问题的途径）

从我们测试的输入的结果来看，我们的算法成功解决了我们的问题。

但这个仅是从算法题的角度来讲的，实际上我们实现的栈并不是一个完善的黑盒，为了使用 int 类型时的正常，我们在 pop 对象的时候并没有手动调用析构函数，这在内存管理上存在问题，而且因为是数组实现的，且在扩张这个栈的大小时我们是直接翻倍这个数组的长度，因此在申请这个块内存的时候，是有很大的概率为了能够多存储一个对象而直接翻倍这块内存空间的，这对内存资源有着很大的潜在的浪费的风险。另外我们并没有为这个代码加上合适的 throw-catch 结构，在实际使用时出了问题是无法即使修复的。关于内存管理的问题，如果访问频次不是很高，那么可以使用链表数据结构来代替数组来实现栈的存储，且在 pop 一个对象的时候，调用一次它的析构函数，并单独为没有显式析构函数存在的类单独写 pop 方法，且使用智能指针来管理内存，这样可以解决这个问题，再为这段代码加上合适的 throw-catch 结构，来及时处理代码运行中的问题。

5. 附录：实现源代码（本实验的全部源程序代码，程序风格清晰易理解，有充分的注释）

附录代码分为两部分，第一部分是直接计算中缀表达式的版本，第二部分是按后缀表达式计算的版本（在实际的测试中，这两种方式的计算效率是相差极大的（原因可能来自中缀表达式中多次递归调用使得程序进行了及其大量的进出函数的操作））。

一、中缀表达式版本：

```
1.  /*2024 级数据结构--数据智能 实验 6 栈 A 计算表达式.cpp*/
2.  #include <iostream>
3.  #define debug cout << "debug" << endl;
4.  using namespace std;
5.
6.  template<class T>
7.  class Stack {
8.  private:
9.      T* data;
10.     int top;
11.     int size;
12. public:
13.     Stack(int size) :size(size), top(-1), data(new T[size]) {}
14.     Stack(): size(2500), top(-1) { data = new T[size]; }
15.     ~Stack() { delete[] data; }
16.     T Top() { return data[top]; }
17.     int getSize() { return size; }
18.     void push(T val) {
19.         if (full()) { return; }
20.         data[++top] = val;
21.     }
22.     void pop() {
23.         if (empty()) { return; }
24.         data[top--].~T();
25.     }
26.     void display() {
```

```
            for (int i = 0; i <= top; i++) {
                cout << data[i] << " ";
            }
            cout << endl;
        }
        bool empty() { return top == -1; }
        bool full() { return top == size - 1; }
    };

    class Solution {
    public:
        void solute();
    };



    size_t strlen(const char* str) {
        size_t len = 0;
        while (str[len]) { len++; }
        return len;
    }

    double coculate(char op, double a, double b)
    {
        switch (op)
        {
            case '+': return a + b;
            case '-': return a - b;
            case '*': return a * b;
            case '/': return a / b;
            default: return 0;
        }
        // switch (op)
        // {
        // case '+': cout<< a + b << endl; return a + b;
        // case '-': cout<< a - b << endl; return a - b;
        // case '*': cout<< a * b << endl; return a * b;
        // case '/': cout<< a / b << endl; return a / b;
        // default: return 0;
        // }
    }

    void one_shoot(Stack<char>& exception, Stack<double>& value)
    {
        if (exception.empty()) { return; }
        if (value.empty()) { return; }
```

```
73.       if (exception.Top() == '(') { return; }
74.       double b = value.Top();
75.       value.pop();
76.       double a = value.Top();
77.       value.pop();
78.       char op = exception.Top();
79.       exception.pop();
80.       if (op == '*' || op == '/')
81.       {
82.           if (exception.empty() || exception.Top() == '(' || exceptio
   n.Top() == '+' || exception.Top() == '-')
83.           {
84.               value.push(coculate(op, a, b));
85.               return;
86.           }
87.           else
88.           {
89.               value.push(a);
90.               one_shoot(exception, value);
91.               double ans = coculate(op, value.Top(), b);
92.               value.pop();
93.               value.push(ans);
94.           }
95.       }
96.       else
97.       {
98.           if (exception.Top() == '+' || exception.Top() == '-' || exc
   eption.Top() == '(' || exception.empty())
99.           {
100.              if (op == '+')
101.              {
102.                  if (exception.Top() == '+' || exception.Top() == '(
   ' || exception.empty())
103.                  {
104.                      value.push(a + b);
105.                  }
106.                  else if (exception.Top() == '-')
107.                  {
108.                      value.push(a - b);
109.                  }
110.              }
111.              else if (op == '-')
112.              {
113.                  if (exception.Top() == '+' || exception.Top() == '(
   ' || exception.empty())
114.                  {
```

```
                    value.push(a - b);
                }
                else if (exception.Top() == '-')
                {
                    value.push(a + b);
                }
            }
        }
        else
        {
            value.push(a);
            one_shoot(exception, value);
            if (op == '+')
            {
                if (exception.Top() == '+' || exception.Top() == '(
' || exception.empty())
                {
                    double ans = value.Top() + b;
                    value.pop();
                    value.push(ans);
                }
                else if (exception.Top() == '-')
                {
                    double ans = value.Top() - b;
                    value.pop();
                    value.push(ans);
                }
            }
            else if (op == '-')
            {
                if (exception.Top() == '+' || exception.Top() == '(
' || exception.empty())
                {
                    double ans = value.Top() - b;
                    value.pop();
                    value.push(ans);
                }
                else if (exception.Top() == '-')
                {
                    double ans = value.Top() + b;
                    value.pop();
                    value.push(ans);
                }
            }
        }
    }
```

```
159. }
160.
161. void Solution::solute()
162. {
163.     char exp[2000];
164.     int n;
165.     cin >> n;
166.     for (int i = 0; i < n; i++)
167.     {
168.         cin >> exp;
169.         Stack<char> exception;
170.         Stack<double> value;
171.         for (int j = 0; j < strlen(exp); j++)
172.         {
173.             if (exp[j] == '(') { exception.push(exp[j]); }
174.             else if (exp[j] == ')')
175.             {
176.                 while (exception.Top() != '(')
177.                 {
178.                     one_shoot(exception, value);
179.                 }
180.                 exception.pop();
181.             }
182.             else if (exp[j] == '+' || exp[j] == '-' || exp[j] == '*
    ' || exp[j] == '/')
183.             {
184.                 exception.push(exp[j]);
185.             }
186.
187.             else if (exp[j] >= '0' && exp[j] <= '9')
188.             {
189.                 value.push(double(exp[j] - '0'));
190.             }
191.         }
192.         while (!exception.empty()&&value.getSize()>=2)
193.         {
194.             one_shoot(exception, value);
195.         }
196.         printf("%.2f\n", value.Top());
197.     }
198. }
199.
200. int main()
201. {
202.     Solution solution;
203.     solution.solute();
```

```
    return 0;
}
```

二、后缀表达式版

```
/*2024级数据结构--数据智能 实验6 栈 A 计算表达式 优先级版.cpp*/
#include <iostream>
#define debug cout << "debug" << endl;
using namespace std;

template<class T>
class Stack {
private:
    T* data;
    int top;
    int size;
public:
    Stack(int size) :size(size), top(-1), data(new T[size]) {}
    Stack(): size(2000), top(-1) { data = new T[size]; }
    ~Stack() { delete[] data; }
    T Top() { return data[top]; }
    int getSize() { return size; }
    void push(T val) {
        if (full()) { return; }
        data[++top] = val;
    }
    void pop() {
        if (empty()) { return; }
        data[top--].~T();
    }
    void display() {
        for (int i = 0; i <= top; i++) {
            cout << data[i] << " ";
        }
        cout << endl;
    }
    bool empty() { return top == -1; }
    bool full() { return top == size - 1; }

};

class Solution {
public:
    void solute();
};

int priority(char op) {
    switch (op) {
```

```
44.         case '+':
45.         case '-':
46.             return 1;
47.         case '*':
48.         case '/':
49.             return 2;
50.         case '^':
51.             return 3;
52.         default:
53.             return 0;
54.         }
55.     }
56.
57.     size_t strlen(const char* str) {
58.         size_t len = 0;
59.         while (str[len]) { len++; }
60.         return len;
61.     }
62.
63.     void Solution::solute()
64.     {
65.         int n;
66.         cin >> n;
67.         for (int i = 0; i < n; i++)
68.         {
69.             Stack<char> op;
70.             Stack<char> back;
71.             char c[2500];
72.             char back_char[2500];
73.             int index = 0;
74.             cin >> c;
75.             for (int i = 0; i < strlen(c); i++)
76.             {
77.                 //如果是数字，直接加入后缀表达式
78.                 if (c[i] <= '9' && c[i] >= '0')
79.                 {
80.                     back.push(c[i]);
81.                     back_char[index++] = c[i];
82.                     continue;
83.                 }
84.                 else
85.                 {
86.                     //如果是（，直接压入栈
87.                     if (c[i] == '(')
88.                     {
89.                         op.push(c[i]);
```

```
                    }
                    //如果是），弹出栈直到遇到（
                    else if (c[i] == ')')
                    {
                        while (op.Top() != '(')
                        {
                            char op_char = op.Top();
                            op.pop();
                            back.push(op_char);
                            back_char[index++] = op_char;
                        }
                        op.pop();
                    }
                    else
                    {
                        //如果是操作符，如果栈顶运算符的优先级小于当前运算符的优先级，或者栈为空或者栈顶是（，则直接压入栈
                        if (op.empty() || priority(c[i]) > priority(op.Top()) || op.Top() == '(')
                        {
                            op.push(c[i]);
                        }
                        //否则，弹出栈直到栈顶运算符优先级小于当前运算符优先级，或者遇到（，然后将当前运算符压入栈
                        else
                        {
                            while (!op.empty() && priority(c[i]) <= priority(op.Top()))
                            {
                                char op_char = op.Top();
                                op.pop();
                                back.push(op_char);
                                back_char[index++] = op_char;
                            }
                            op.push(c[i]);

                        }
                    }
                }
            }
            //弹出栈剩余的运算符
            while (!op.empty())
            {
                char op_char = op.Top();
                op.pop();
                back.push(op_char);
```

```
            back_char[index++] = op_char;
        }
        back.display();
        // for (int i = 0; i < strlen(back_char); i++)
        // {
        //     cout << back_char[i] <<' ';
        // }

        Stack<double> ans;

        for (int i = 0; i < index; i++)
        {
            if (back_char[i] <= '9' && back_char[i] >= '0')
            {
                ans.push(back_char[i] - '0');
            }
            else
            {
                double num1 = ans.Top();
                ans.pop();
                double num2 = ans.Top();
                ans.pop();
                switch (back_char[i])
                {
                case '+':
                    ans.push(num2 + num1);
                    break;
                case '-':
                    ans.push(num2 - num1);
                    break;
                case '*':
                    ans.push(num2 * num1);
                    break;
                case '/':
                    ans.push(num2 / num1);
                    break;
                    // case '^':
                        // ans.push(pow(num2, num1));
                        // break;
                default:
                    break;
                }
            }
        }
        printf("%.2f\n", ans.Top());
    }
```

```
178.
179. }
180.
181. int main()
182. {
183.     Solution solute;
184.     solute.solute();
185.     return 0;
186. }
```