

数据结构与算法 课程实验报告

学号：202000130143	姓名： 郑凯饶	班级： 计科 20.1
实验题目：栈		
实验学时：2	实验日期：1103	
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。		
软件开发环境： Windows 10 家庭中文版 64 位 (10.0, 版本 18363) Dev-C++ IDE		
1. 实验内容 创建栈类，采用数组描述，计算数学表达式的值。数学表达式由单个数字和运算符“+”，“-”，“*”，“/”，“(”，“)”构成。		
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）		
<pre> template&lt;class T&gt; class arrayStack { // : public stack&lt;T&gt; public:     arrayStack(int iniCap = 1000);     ~arrayStack() { delete [] stack; }     bool empty() const { return Top == -1; }     int size() const { return Top + 1; }     T&amp; top() {         if (Top == -1) {         }         return stack[Top];     }     void pop() {         stack[Top--].~T(); // 3221226356         Top--;     }     void push(const T&amp; tar);     void clear() {         while (!this-&gt;empty()) {             this-&gt;pop();         }     } private:     int Top;     int Length;     T* stack; };           </pre>		
以上为栈的定义与实现。 Top: 栈顶元素的索引 Length: 栈的容量 Stack: 栈元素的储存空间		
表达式计算的思路： 维护两个栈 Dig 和 Sym，前者储存数字，后者储存运算符。扫描一遍表达式，遇到数字直接入栈 (Dig)；遇到运算符 (加减乘除) 将其优先级和栈顶 (Sym) 的元素进行比较，若 Top 的优先级大于等于当前运算符，进行一次 Cal ()，直到栈空或者 Top 优先级小于当前运算符。若遇到 “(”，直接入栈；遇到 “)”，不断 Cal () 直到遇到 “(”。 Cal () 函数就是进行一次运算，将 Dig 栈顶的两个元素弹出，将 Sym 栈顶的一个元素弹出，进行运算，结果入栈 (Dig)。		

3. 测试结果（测试输入，测试输出）  
在 OJ 平台上成功提交。

4. 分析与探讨（结果分析，若存在问题，探讨解决问题的途径）

一开始设计算法的时候，忽略了栈顶和当前运算符是同级别时也要进行 Cal ()，像  $3/2*5$ ，计算顺序应该是  $3/2=1.5 \rightarrow 1.5*5=7.5$ ，而不是  $2*5=10 \rightarrow 3/10=0.3$ 。

另一方面，在实现算法时，写错了一些程序逻辑和细节。如将 pop 误输入为 top，导致最后未将栈清空，而题目有多组样例输入，引发越界错误，程序返回异常值 3221225477。

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）

```
6. #include<bits/stdc++.h>
7. using namespace std;
8.
9. template<class T>
10. class arrayStack { // : public stack<T>
11. public:
12.     arrayStack(int iniCap = 1000);
13.     ~arrayStack() { delete [] stack; }
14.     bool empty() const { return Top == -1; }
15.     int size() const { return Top + 1; }
16.     T& top() {
17.         if (Top == -1) {
18.
19.         }
20.         return stack[Top];
21.     }
22.     void pop() {
23.         stack[Top--].~T(); // 3221226356
24.         // Top--;
25.     }
26.     void push(const T& tar);
27.     void clear() {
28.         while (!(this -> empty())) {
29.             this -> pop();
30.         }
31.     }
32. private:
33.     int Top;
34.     int Length;
35.     T* stack;
36. };
37.
38. template<class T>
39. arrayStack<T>::arrayStack(int iniCap) {
```

```

40. Length = iniCap;
41. stack = new T[Length];
42. Top = -1;
43.}
44.
45.template<class T>
46.void arrayStack<T>::push(const T& tar) {
47. // changeLength1D()
48. stack[++Top] = tar;
49.}
50.
51.arrayStack<double> Dig;
52.arrayStack<int> Sym;
53.
54.string operand[] = {"+", "-", "*", "/"};
55.
56.void Cal() {
57. double b = Dig.top(); Dig.pop();
58. double a = Dig.top(); Dig.pop();
59. int op = Sym.top(); Sym.pop();
60.
61. double Res;
62. switch(op) {
63. case 0:
64.     Res = a + b;
65.     break;
66. case 1:
67.     Res = a - b;
68.     break;
69. case 2:
70.     Res = a * b;
71.     break;
72. case 3:
73.     Res = a / b;
74.     break;
75. }
76. // cout << a << " " << operand[op] << " " << b << " = " << Res << '\n';
77. Dig.push(Res);
78.}
79.
80.void solve() {
81.
82. int n; cin >> n;
83. string Exp;
84. for (int i = 1; i <= n; i++) {
85. // Dig.clear(); Sym.clear(); // 3221225477

```

```
86.  cin >> Exp;
87.  int len = Exp.length();
88.  // Cal
89.  for (int j = 0; j < len; j++) {
90.      if (Exp[j] <= '9' && Exp[j] >= '0') Dig.push(Exp[j] - '0');
91.      else {
92.          if (Exp[j] == '+') {
93.              while (!Sym.empty() && (Sym.top() <= 3)) {
94.                  Cal();
95.              }
96.              Sym.push(0);
97.          }
98.          else if (Exp[j] == '-') {
99.              while (!Sym.empty() && (Sym.top() <= 3)) {
100.                  Cal();
101.              }
102.              Sym.push(1);
103.          }
104.          else if (Exp[j] == '*') {
105.              while (!Sym.empty() && (Sym.top() == 2 || Sym.top() == 3)) {
106.                  Cal();
107.              }
108.              Sym.push(2);
109.          }
110.          else if (Exp[j] == '/' ) {
111.              while (!Sym.empty() && (Sym.top() == 2 || Sym.top() == 3)) {
112.                  Cal();
113.              }
114.              Sym.push(3);
115.          }
116.          else if (Exp[j] == '(') {
117.              Sym.push(4);
118.          }
119.          else if (Exp[j] == ')') { // 5
120.              while (Sym.top() != 4) {
121.                  Cal();
122.              }
123.              Sym.pop();
124.          }
125.      }
126.  }
127.  // More
128.  while (Dig.size() > 1) {
129.      //  cout << "Cal" << '\n';
130.      Cal();
131.
```

```
132.     double Ans = Dig.top(); Dig.pop();
133.     cout << Ans << '\n';
134. }
135. }
136. int main(){
137.     cout << setiosflags(ios::fixed) << setprecision(2);
138.     solve();
139.     return 0;
140. }
```