**实验三实验报告**

**2112066 于成俊**

**一、实验题目：**

**使用数组实现栈的出栈、入栈、获取栈顶元素、判断空栈四个方法。并使用该栈进行表达式运算。表达式包含“+”、 “-”、"\*”、“/”、 "("、 “")"和数字的字符串，输出该表达式运算结果。**

**输入: ((1+2)\*3-5)/2**

**输出: 2**

**实验要求:不能使用调用外部库实现栈。**

**二、核心代码**

**①出栈：**

void pop() {

if (top == 0) return;

else { top--; }

}

**②入栈：**

void push(T n) {

top++;

stack[top] = n;

}

**③获取栈顶元素**

T get() {return stack[top];}

**④判断空栈：**

bool empty() {

if (top == 0) { return true; }

else { return false; }

}

**⑤表达式运算实现：**

for (int i = 0;; i++) {

if (A[i] == '\0') { break;}**//读取字符，到字符串末尾则跳出循环**

if (A[i] == '+' || A[i] == '-' || A[i] == '\*' || A[i] == '/' || A[i] == '(' || A[i] == ')') {

if (Symbol.empty()) {

//考虑负数

if ((A[i] == '-' && i == 0) || (i > 0 && A[i - 1] == '(')) {

//满足条件即为负数，然后按正常插入数字的代码执行就可以了

for (int j = i + 1;; j++) {

if (A[j] >= 48 && A[j] <= 57) {

y = y \* 10 + A[j] - 48;

}

else {

if (A[j] == '.') {

while (1) {

j++;

if (A[j] >= 48 && A[j] <= 57) {

double key = A[j] - 48;//防止先去尾再转换成double

z = z / 10 + key / 10;

}

else { break; }

}

}

number.push(-(y + z));

i = j - 1;

z = 0;

y = 0;

break;

}

}

continue;

}

Symbol.push(A[i]); continue;

}**//符号栈为空，则直接进栈**

if(A[i] == '('){ Symbol.push(A[i]); continue;}//左括号直接进栈

//因为出现右括号，所以要把左括号之前的操作符全部弹出

if (A[i] == ')') {

for (int k =Symbol.top;; k--) {

if (A[k] == '(') {

Symbol.pop();

break;

}//不断寻找左括号，然后跳出循环

if (Symbol.stack[k] == '+' ) {

Symbol.pop();

x = number.stack[number.top - 1] + number.stack[number.top];//将操作数栈前两个进行运算

number.pop();number.pop();//因为两个数运算，所以弹出两次

number.push(x);

}//减法、乘法、除法代码类似

if (A[i] == '\*') {

if (Symbol.stack[Symbol.top ] == '/') {

x = number.stack[number.top - 1] / number.stack[number.top ];

number.pop(); number.pop();

number.push(x);

Symbol.pop();

Symbol.push('\*');

continue;

}

if (Symbol.stack[Symbol.top] == '\*') {

x = number.stack[number.top - 1] \* number.stack[number.top];

number.pop(); number.pop();

number.push(x);

Symbol.pop();

Symbol.push('\*');

continue;

}

else{ Symbol.push('\*'); }

continue;

}//乘号，除号类似

if (A[i] == '+') {

if (Symbol.stack[Symbol.top] == '\*') {

x = number.stack[number.top - 1] \* number.stack[number.top];

number.pop();number.pop();

number.push(x);

Symbol.pop();

Symbol.push('+');

continue;

}

if (Symbol.stack[Symbol.top] == '/') {

x = number.stack[number.top - 1] / number.stack[number.top];

number.pop();number.pop();

number.push(x);

Symbol.pop();

Symbol.push('+');

continue; }

if(Symbol.stack[Symbol.top]=='+'||Symbol.stack[Symbol.top] == '-') {

Symbol.push('+');continue;

}

Symbol.push('+'); continue;

}//加号,减号类似（减号要进行负号判断）

}

//若是数字，则直接进入操作数栈

else {

for (int j=i;; j++) {

if (A[j] >= 48 && A[j]<= 57) {

y = y \* 10 + A[j] - 48;

}

else {

//小数实现

if (A[j] == '.') {

while (1) {

j++;

if (A[j] >= 48 && A[j] <= 57) {

double key = A[j] - 48;//防止先去尾再转换成double

z = z / 10 + key/10;

}

else { break; }

}

}

number.push(y+z);

i = j - 1;

z = 0;

y = 0;

break;

}

}}

if (Symbol.empty()) {

x = number.stack[number.top];

cout << x; return 0;}//若为空，直接返回栈底元素

若不为空，依次弹出操作符栈，代码与之前类似

**三、算法分析**

**①时间复杂度分析 O(n)**

**②空间复杂度分析 O(n)**

**四、心得总结**

**1.要先把逻辑结构理清，思路要清晰。**

**2.当进行两个运算数运算时，要注意操作数栈要弹出两次，还要把得数进栈**

**3.注意栈的top位置，是要只向栈顶元素，不要指向栈顶元素的上一个**

**4.要注意操作数栈的栈顶元素在表达式中在右边**