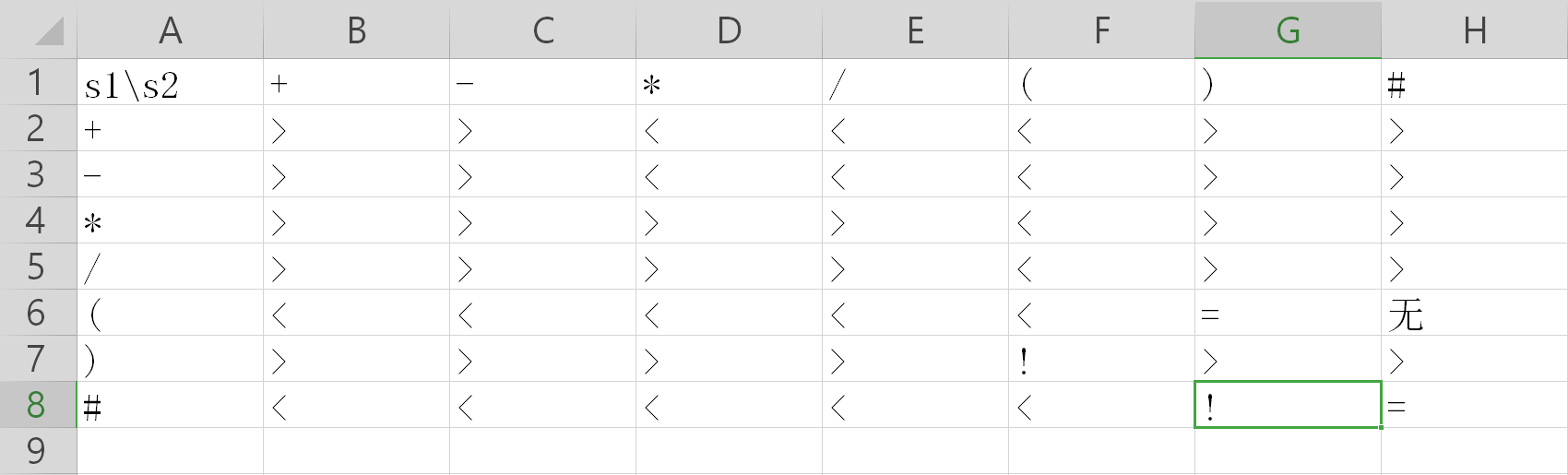
**表达式求值**

1. **直接上代码**
2. 我们可以构建这样的一张表格



代码

#include "SQStack.h"

using namespace std;

char op\_prior\_table[][7]{

{'>', '>', '<', '<', '<', '>', '>'},

{'>', '>', '<', '<', '<', '>', '>'},

{'>', '>', '>', '>', '<', '>', '>'},

{'>', '>', '>', '>', '<', '>', '>'},

{'<', '<', '<', '<', '<', '=', '!'},

{'>', '>', '>', '>', '!', '>', '>'},

{'<', '<', '<', '<', '<', '!', '='}

};

int index\_of\_op(const char op){

if(op == '+') return 0;

if(op == '-') return 1;

if(op == '\*') return 2;

if(op == '/') return 3;

if(op == '(') return 4;

if(op == ')') return 5;

if(op == '#') return 6;

}

int match\_pripr(const char op1, const char op2){

int index1 = index\_of\_op(op1);

int index2 = index\_of\_op(op2);

if(op\_prior\_table[index1][index2] == '>') return 1;

if(op\_prior\_table[index1][index2] == '=') return 0;

if(op\_prior\_table[index1][index2] == '<') return -1;

return 1000; //没有的事情

}

//计算函数

int calc\_num(const char op, const int num1, const int num2){

if(op=='+') return num2+num1;

if(op=='-') return num2-num1;

if(op=='\*') return num2\*num1;

if(op=='/') return num2/num1;

}

int main(){

SQStack<int> num;

SQStack<char> Op;

Op.push('#'); //嘿嘿，我们需要加入#才能进行匹配

string exp = "1+2\*3/(5-4)#";

for(int i = 0 ; i < exp.size(); i++){

if(exp[i]>='0' && exp[i]<='9'){ //如果是数字

num.push(exp[i]-'0');

}else{

//是运算符了

char op = Op.top();

int cmp = match\_pripr(op, exp[i]);

if(cmp < 0){ // 如果表达式中的运算符优先级大于栈中的，我们进入栈

Op.push(exp[i]);

}else if(cmp > 0){ //遇到的运算符比栈中的低，那我们就直接先运算入栈

Op.pop();

int num1 = num.top(); num.pop();

int num2 = num.top(); num.pop();

int res =calc\_num(op, num1, num2);

num.push(res);

cout << res << endl;

i--;

}else{ // 相等的话，我们就出栈

Op.pop();

}

}

}

cout << num.top();

}