1. **将下面不完整的程序进行调试，需要自己编写部分函数，如入栈push()、出栈pop()、取头元素top\_stack。**

// 算术表达式求值的运算符优先数算法。假定从终端输入的表达式无语法错误

#include <iostream>

using namespace std;

const int StackSize = 10; //10是示例性的数据，根据实际问题具体定义

template <typename DataType> //定义模板类SeqStack

class SeqStack

{

public:

SeqStack(); //构造函数，初始化一个空栈

~SeqStack(); //析构函数

void Push(DataType x); //入栈操作，将元素x入栈

DataType Pop(); //出栈操作，将栈顶元素弹出

DataType GetTop(); //取栈顶元素（并不删除）

int Empty(); //判断栈是否为空

private:

DataType data[StackSize]; //存放栈元素的数组

int top; //游标，栈顶指针，为栈顶元素在数组中的下标

};

template <typename DataType>

SeqStack<DataType> ::SeqStack()

{

top = -1;

}

template <typename DataType>

SeqStack<DataType> :: ~SeqStack()

{

}

template <typename DataType>

int SeqStack<DataType> ::Empty()

{

if (top == -1)

return 1;

else

return 0;

}

template <typename DataType>

int SeqStack<DataType> ::Precedence(char ch)

{int z = 0;

switch (ch)

{

case '+':z = 1; break;

case '-':z = 1; break;

case '\*':z = 2; break;

case '/':z = 2; break;

case '#':z = 0; break;

default:printf("error!\n");

}

return z;

}

template <typename DataType>

int SeqStack<DataType> ::Operate(int x, char ch, int y)

{

int z = 0;

switch (ch)

{

case'+':z = x + y; break;

case'-':z = x - y; break;

case'\*':z = x \* y; break;

case'/':z = x / y; break;

default:printf("error!\n");

}

return z;

}

int main()

{

SeqStack<int> OPTR{ },OPND{}; //定义顺序栈变量S

char w;

int x, y, z;

char op[6];

strcpy\_s(op, "+-\*/#"); // op为运算符集合

//init(OPTR);

OPTR.Push( '#');

//init(OPND);

//栈初始化，并在运算符栈的栈底压入表达式左边虚设的字符“#”

w = getchar(); //从终端读入一个字符

while (!((w == '#') && (OPTR.GetTop() == '#')))

{

if (!strchr(op, w))

{

OPND.Push(atoi(&w));

w = getchar(); //操作数进操作数栈

}

else if (OPTR.precedence(w) > OPTR.precedence(OPTR.GetTop())) //比较优先数

{

OPTR.Push(w);

w = getchar();

}

else

{

char theta;

theta = OPTR.Pop(); //弹出栈顶运算符

y = OPND.Pop();

x = OPND.Pop(); //连续弹出两个操作数

z = OPND.operate(x, theta, y); //进行运算xθy

OPND.Push(z); //将运算结果压入操作数栈

}

}

cout << z << endl;

return(OPND.GetTop()); //从操作数栈顶取出表达式运算结果返回

}

1. **队列的应用—门诊看病问题**
   1. 在医院门诊某诊室看病的时候,护士给每位病人一个编号
   2. 病人按编号进行排队
   3. 一个病人看完后,护士安排下一个病人看病
   4. 编程:对此过程进行模拟

提示：先录入病人的编号，然后呼叫当前病人的号并提示下一号准备，在一定的时间间隔后（或通过屏幕提示）再呼叫当前病人看病并提示下一号准备，直到看完所有的病人。