

主函数：

void main()

{

ElemType str1, e1, e2, str2;

char str[100];

stack S1, S2;

InitStack(&S1);

InitStack(&S2);

gets\_s(str);

for (ElemType i = 0; str[i] != '\0'; i++)

{

if (str[i] == 42 || str[i] == 43 || str[i] == 45 || str[i] == 47)

{

str2 = str[i];

Pop(&S1, &e1);

Pop(&S1, &e2);

switch (str2)

{

case 43: Push(&S1, e2 + e1); break;

case 45: Push(&S1, e2 - e1); break;

case 42: Push(&S1, e2 \* e1); break;

case 47: Push(&S1, e2 / e1); break;

}

Push(&S2, str[i]);

}

if (str[i] >= 48 && str[i] <= 57)

{

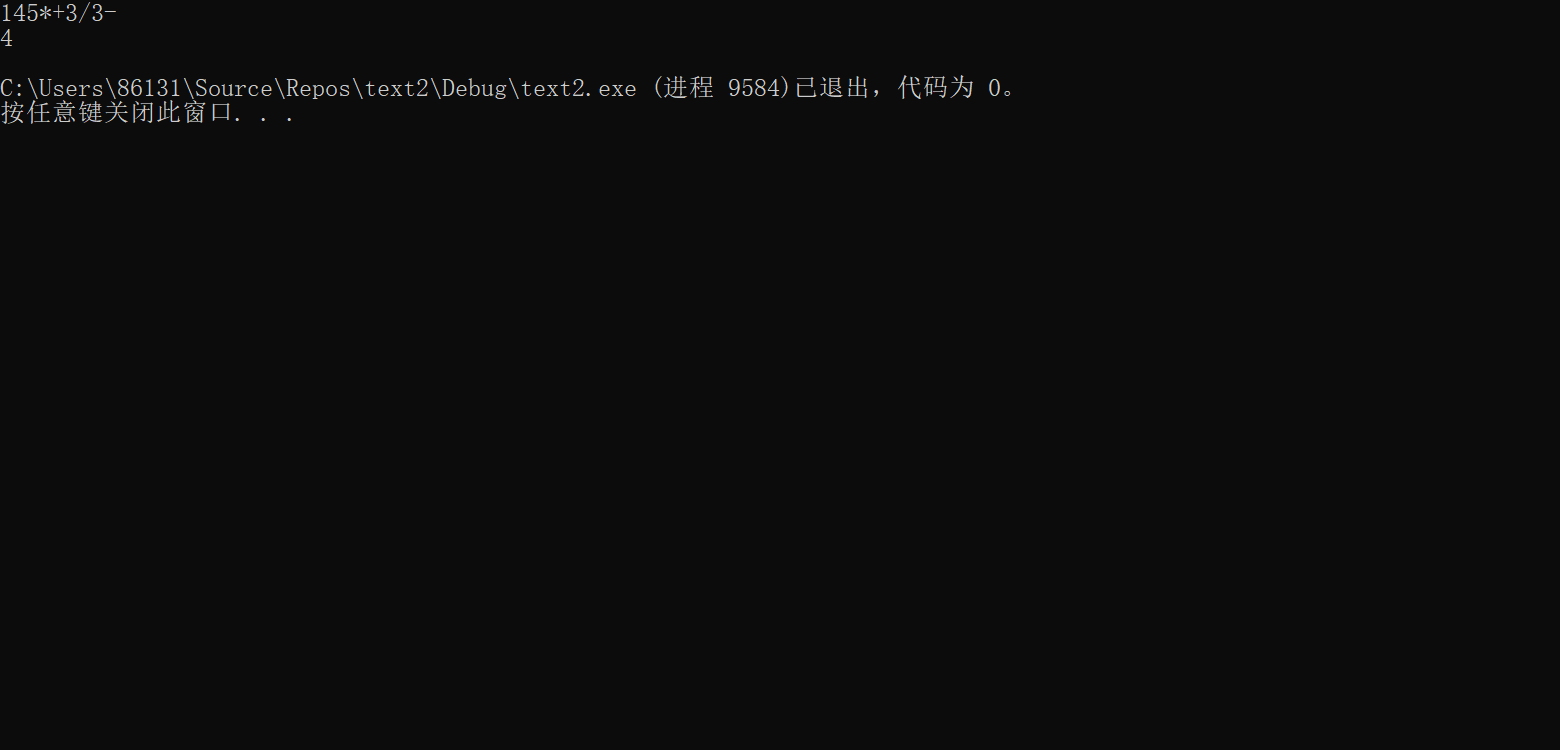
Push(&S1, str[i]-48);

}

}

ListTraverse(S1);

}



总结：

从栈的特点先入后出考虑，通过循环操作遇到运算符就实现s1表中两个数字的运算，然后将结果压栈，依次执行直到完成全部操作。编写代码时遇到的第一个问题就是输出的结果不对，仔细分析代码发现存在运算符的优先级顺序问题，经多次修改实现代码。