格式

1. 符号使用 +-*/%() 的形式，包括“（”也要修改。
2. 统一使用半角字符，区分(和（
3. 不要使用\\刻意分段，要分段就空一行（而且不要一句话就一段）
4. 从一级标题（6个=）、二级标题依次排列
5. 代码格式完全不对

内容

1. 合法的判定似乎不够严谨，例如是否考虑了符号两侧无数字的情况？2+
2. 计算方法中，队列存储的是什么？为什么存储的是运算符，却有数字入栈？
3. 传统的实现应当是使用两个栈，而非一个栈和一个队列。
4. 能否找一些例题？
5. 供参考：有兴趣可学习编译原理中的递归下降分析法。

表达式的计算

给出一个以=结尾的包含+-*/%()的式子，按运算规则计算其值。
挺模板的一个问题
我们一般的数学语言给出的式子是表达式的中缀表达式，而为了计算，我们需要将其转为后缀表达式。

中缀表达式的合法性判定

一般来说给的数据是合法的，但不排除会有题目让你判断是否合法。
这时候我们需要做一个合法性判断。
因为中缀表达式就是我们的自然语言，所以判断起来也比较容易，主要有以下几点：

1. 开头若有字符，则只能是+或-。
2. 括号需要匹配，即有“（”必有“）”。

中缀转后缀的方法

用队列存储运算符，栈辅助处理。对每个字符，规则如下：

1. 若为数字，直接入队（当然，多位数字要提前处理一下）
2. 若为运算符：
    1. 栈为空直接入栈
    2. 优先级大于栈顶元素入栈
    3. 优先级小于等于栈顶元素，则栈顶元素出栈入队，直到优先级大于栈顶元素。
3. 若为（，直接入栈
4. 若为），栈顶元素出栈入队，直到（，两个括号不入队。

计算方法

对于一个后缀表达式，用栈辅助计算
从左至右扫描，若为数字，直接入栈，若为运算符，则将栈顶两个元素计算，计算结果入栈。
直到最后留下一个数字，即为结果。

#include <stdio.h> #include <string.h> #include <stdlib.h> char stack[500]; int queue[500]; void calcu(int r,int top) { switch (stack[top]){

```
      case '+':queue[r-1]=queue[r]+queue[r-1];break;
  case '-':queue[r-1]=queue[r-1]-queue[r];break;
  case '*':queue[r-1]=queue[r]*queue[r-1];break;
  case '/':queue[r-1]=queue[r-1]/queue[r];break;
  case '%':queue[r-1]=queue[r-1]%queue[r];break;
  }
```

} int main() {

```
      char s[500];
int pr[300]={0};
pr[(int)'+']=pr[(int)'-']=1;
pr[(int)'*']=pr[(int)'/']=pr[(int)'%']=2;
pr[(int)'(']=0;
gets(s);
int top=0,r=0,i=0;
while (s[i]!='=')
{
      if (s[i]>='0'&&s[i]<='9')
  {
      int x=(int)s[i]-48;
      i++;
      while (s[i]>='0'&&s[i]<='9')
      {
          x=x*10+(int)s[i]-48;
          i++;
      }
      queue[++r]=x;
      i--;
  }
  else if (s[i]!='('&&s[i]!=')'&&s[i]!=' '){
      if (top==0) stack[++top]=s[i];
      else {
          if (pr[(int)s[i]]>pr[(int)stack[top]]) stack[++top]=s[i];
          else {
              while (top!=0&&pr[(int)s[i]]<=pr[(int)stack[top]]) calcu(r--
,top--);
              stack[++top]=s[i];
          }
      }
  }
  else if (s[i]=='(') stack[++top]=s[i];
  else if (s[i]==')') {
          while (stack[top]!='(') calcu(r--,top--);
          top--;
```

```
  }
  i++;
}
for (int i=top;i>=1;i--) calcu(r--,top--);
printf("%d",queue[1]);
return 0;
```

} </codedoc>