数据结构需要接口interface，即ADT抽象数据类型：定义能做什么，而不是怎么做

- [ ] LIFO后进先出队列：栈

> 进push，出pop
> 判断括号匹配算法

CheckParen(str):

```
Stack s
int i=1
while (str[i]!=NULL)
  if (str[i] is '(' or '[' or '{')
    s.push(str[i])
  if (str[i] is ')' or ']' or '}')
    if (s.empty())
      return false
    if (s.pop() and str[i] mismatch)
      return false
  i++
return s.empty()
```

- [ ] FIFO Queue

> 增删add和remove时间复杂度取决于位置

数组

- 数组适合栈，只适合在尾部操作（push，pop）。
- 不适合FIFO、Deque两端队列
- 循环数组：保留优点，在头尾修改快，但中间位置插入删除慢，查询快
- 估计数组容量限制，分配空间

链表

- 链表内存空间不需要连续，一小块小块的，存储灵活快，没有容量限制
- 链表适合栈，但不适合在尾部pop，链表适合头尾操作
- 高级链表可以克服这个缺点

后缀表达式（postfix expression）

- 中缀表达式和后缀表达式互换

逆波兰式（RPN）

不需要括号来突出优先级

EvalRPN(str):

```
Stack s
while ((token=NextToken(str))!=NULL)
  if (token is an operand)
    s.push(token)
  else
    res=PopOperandAndCalc(s,token)
    s.push(res)
return s.pop()
```

> token每次取出一个操作数（operand）或操作符。
> 根据操作符几目就弹出几个操作数，结果存在res里压回栈中
> 最终栈只剩一个元素，即结果

函数调用

> 汇编层面不会调用函数

递归效率低，代码简单