## 3.2 栈的应用示例

- 在较复杂的数据处理过程中，通常需要保存多个临时产生的数据。
- 如果先产生的数据后进行处理，那么需要用栈来保存这些数据。
- 后面示例算法中均采用顺序栈，实际上采用链栈完全相同。

【示例-1】假设以I和O分别表示进栈和出栈操作，栈的初态和终态均为空，进栈和出栈的操作序列可表示为仅由I和O组成的序列。

① 下面所示的序列中哪些是合法的？

A. IOIIOIOO　　B. IOOIOIIO

C. IIIOIOIO　　D. IIIOOIOO

② 通过对①的分析，设计一个算法利用链栈的基本运算判定操作序列str是否合法。若合法返回1；否则返回0。

解：

A. IOIIOIOO　　B. IOOIOIIO

C. IIIOIOIO　　D. IIIOOIOO

① A、D均合法，而B、C不合法。

② 用一个顺序栈st来判断操作序列是否合法。

```
#include "Sqstack.cpp"  //包含前面的顺序栈基本运算函数
int Judge(char str[], int n)
{
   int i;
   ElemType x;
   SqStack st;          //定义一个顺序栈st
   InitStack(st);       //栈初始化
```

```
    for (i=0;i<n;i++)           //遍历str的所有字符
    {  if (str[i]=='I')         //为'I'时进栈
          Push(st,str[i]);
       else if (str[i]=='O')    //为'O'时出栈
          if (!Pop(st,x))       //若栈下溢出，则返回0
          {  DestroyStack(st);
             return 0;
          }
    }
    if (StackEmpty(st))          //栈为空时返回1；否则返回0
    {   DestroyStack(st); return 1; }
    else
    {   DestroyStack(st); return 0; }
}
```

【示例-2】 回文指的是一个字符串从前面读和从后面读都一样，如"abcba"、"123454321"都是回文。

设计一个算法利用顺序栈的基本运算判断一个字符串是否为回文。

```
int Palindrome(char str[],  int n)
{
   SqStack st;                    //定义一个顺序栈st
   InitStack(st);                 //栈初始化
   int i;
   char ch;
    for (i=0;i<n;i++)             //所有字符依次进栈
          Push(st,str[i]);
    i=0;                                  //从头开始遍历str
    while (!StackEmpty(st))       //栈不空循环
    {  Pop(st,ch);                //出栈元素ch
       if (ch!=str[i++])          //两字符不相同时返回0
           return 0;
    }
    return 1;                     //所有相应字符都相同时返回1
}
```

【示例-3】 设计一个算法，判断一个可能含有小括号（'('与')'）、中括号（'['与']'）和大括号（'{'与'}'）的表达式中各类括号是否匹配。若匹配，则返回1；否则返回0。

✓ Exp = 5*（7+4*（6+8*（1+2））） 

**解：思路：**

- 设置一个顺序栈st，定义一个整型flag变量（初始为1）。
- 用*i*扫描表达式exp，当i<n并且flag=1时循环：
- 当遇到左括号“(”、“[”、“{”时，将其进栈；
- 遇到“}”、“]”、“)”时，出栈字符ch，若出栈失败（下溢出）或者ch不匹配，则置flag=0退出循环；
- 否则直到exp扫描完毕为止。
- 若栈空并且flag为1则返回1，否则返回0。

```
#include "SqStack.cpp"     //包含前面的顺序栈基本运算函数
int Match(char exp[],int n)      //exp存放表达式
{
   SqStack st;               //定义一个顺序栈st
   InitStack(st);            //栈初始化
   int flag=1,i=0;
   char ch;
```

```
while (i<n && flag==1)    //遍历表达式exp
{  switch(exp[i])
   {
   case '(': case '[': case '{':  //各种左括号进栈
      Push(st,exp[i]); break;
   case ')':                  //判断栈顶是否为'('
      if (!Pop(st,ch) || ch!='(')   //出栈操作失败或不匹配
         flag=0;
      break;
   case ']':                  //判断栈顶是否为'['
      if (!Pop(st,ch) || ch!='[')   //出栈操作失败或不匹配
         flag=0;
      break;
   case '}':                  //判断栈顶是否为'{'
      if (!Pop(st,ch) || ch!='{')   //出栈操作失败或不匹配
      flag=0;
      break;
   }
   i++;
}
```

```
    if (StackEmpty(st) && flag==1) //栈空且符号匹配则返回1
        return 1;
    else
        return 0;                  //否则返回0
}
```

END