例如，求和可以被认为是具有 n 个输入和 1 个输出的函数。该函数实现 n 个输入值相加得到总和并最终输出求和结果。函数式语言主要实现下面的功能：

- 函数式语言预定义一系列可供任何程序员调用的原始（原子）函数。
- 函数式语言允许程序员通过若干原始函数的组合创建新的函数。

图 9-7　函数式语言中的函数

例如，定义一个称为 first 的原始函数，由它来完成从一个数据列表中抽取第一个元素的功能。再定义另一个函数 rest，由它完成从一个数据列表中抽取出除第一个元素以外的所有元素。通过两个函数的组合使用，可以在一个程序中定义一个函数来完成对第三个元素的抽取，如图 9-8 所示。

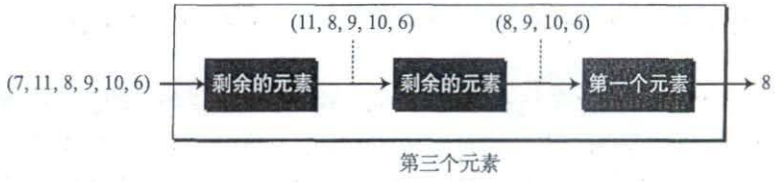

图 9-8　提取列表中的第三个元素

函数式语言相对过程式语言具有两方面优势：它支持模块化编程并且允许程序员使用已经存在的函数来开发新的函数。这两个因素使得程序员能够编写出庞大而且不易出错的程序。

一些函数式语言

我们以 LISP 和 Scheme 为例来简要介绍函数式语言。

LISP

表处理解释语言（LISt Programming，LISP）是 20 世纪 60 年代早期由麻省理工学院科研小组设计开发的。它是一种把表作为处理对象的语言。

Scheme

表处理解释语言没有统一标准化。不久之后，就有许多不同的版本流传于世。实际使用的标准是由麻省理工学院在 20 世纪 70 年代早期开发的，称为 Scheme。

Scheme 语言定义了一系列原始函数来解决问题。函数名和函数的输入列表写在括号内，结果是一个可用于其他函数输入的列表。例如，有一个函数 car，用来从列表中取出第一个元素。第二个函数 cdr 用来从列表中取出除第一个元素以外的所有元素。两个函数如下：

```
(car 2 3 7 8 11 17 20)→2
(cdr 2 3 7 8 11 17 20)→3 7 8 11 17 20
```

现在可以通过组合这两个函数来完成从列表中取出第三个元素的函数。

```
(car ( cdr ( cdr list ) ) )
```

如果将上面的函数应用于列表 2 3 7 8 11 17 20，结果是取出 7。我们来分析一下，最里面的括号取出列表 3 7 8 11 17 20。中间一层括号取出列表 7 8 11 17 20。再通过函数 car 取出该列表的第一个元素 7。