7.9.4 创建子视图

多维数组的操作是比较复杂的，multi_array 允许用户为多维数组创建一个只查看其中一部分数据的子视图（view），子视图既可以与原数组拥有相同的维数，也可以少于原数组的维数。后一种情况被称为多维数组的“切片”（slice），它可以降低数组的维数，使之更易处理。

以二维数组为例，假设我们有一个 3×4 的矩阵：

```
typedef multi_array<int, 2> ma_type;
multi_array<int, 2> ma(extents[3][4]) ;
```

该矩阵里面的数据如下：

```
(0, 1, 2, 3)
(4, 5, 6, 7)
(8, 9,10,11)
```

我们需要使用 index_gen 类的预定义实例 indices 来定义子视图的索引范围。

indices 用法类似 extents 对象，它重载了 operator[]来限定范围，但它的参数不是单个的整数，而是一个 multi_array<T, N>::index_range 对象，它用来指定从多维数组中抽取的维度范围。例如：

```
typedef ma_type::index_range range;                    //简化类型定义
indices[range(0,2)][range(0,2)];                       //创建一个临时 indices 对象
```

multi_array 的 operator[]接收 indices 对象返回子视图，子视图的类型是 multi_array<T, N>::array_view<N>::type，也可以直接用 auto，从而无须写出这个烦琐的类型声明。所以，获取一个 2×2 的子视图可以这样：

```
auto view = ma[indices[range(0,2)][range(0,2)] ];  //自动推导类型
ma_type::array_view<2>::type view =                //等价的 array_view 类型
      ma[indices[range(0,2)][range(0,2)] ];
```

它的数据是 ma 的一部分：

```
(0,1)(4,5)
```

multi_array 的子视图也是个 multi_array，拥有相同的接口，可以用 operator[]、num_dimensions()等成员函数操作，但不能改变子视图的形状和大小。例如：

```
cout << view.num_elements() << endl;
for (int i = 0; i < 2; ++i)
```