7. 设有一个 100*100 的稀疏矩阵，其中约有 1%的非 0 元素。每个非 0 元素以一个三元组表示（行号，列号，元素值）。欲将此矩阵中的非 0 元素存放在一个如代码清单 9-2 所示的闭散列表中。试设计散列表的长度、散列表中元素的类型、以及元素到关键字值的转换函数。假设练习 3 中的稀疏矩阵的元素值的范围为 1 到 1000 之间的整型数。试利用随机函数按 1%的非 0 元素的概率生成这个矩阵，并将矩阵存入闭散列表中。并检验最后生成的非 0 元素的个数是否为 1%。

【解】这个矩阵有 1%的非 0 元素，就是约有 100 个非 0 元素。我们选择一个大于等于 100 的素数 131 作为闭散列表的长度。散列表的每个元素存放一个矩阵的非 0 元素，表示一个非 0 元素要用三个属性：行号、列号和元素值。唯一标识一个元素的属性是行号和列号。也就是说，关键字是（行号，列号）。为此还需要定义一个关键词到整型数的转换函数 Get_Key。散列表元素的类型定义见代码清单 9-22。

代码清单 9-22 散列表元素类型

```
1.  template<class T>
2.  class matrix_entry{
3.        friend int Get_Key(const matrix_entry<T> &entry)
4.        {     return entry.row * 100 + entry.column; }
5.        friend bool operator==(const matrix_entry &a, const matrix_entry &b){
6.              if(a.row == b.row && a.column == b.column) return true;
7.              else return false;
8.        }
9.
10. public:
11.    matrix_entry(int r, int c, T value): row(r),column(c),data(value) {}
12.       matrix_entry() {}
13.
14. private:
15.       int row, column;
16.       T data;
17. };
```

Get_Key 函数用矩阵元素在按行序排列的序号作为它在散列表中的唯一标识。当要保存一个整型的矩阵时，可以定义一个散列表类的对象

```
closeHashTable<matrix_entry<int> > myhash(131, Get_Key);
```

如果要插入一个在第 20 行 30 列值为 10 的的矩阵元素，需要定义一个 matrix_entry 的对象

```
matrix_entry p(20,30,10);
```

然后对 myhash 对象调用 insert 函数插入 p

```
myhash.insert(p);
```