```
1       /**
2        * Find an item in the hash table.
3        * @param x the item to search for.
4        * @return the matching item.
5        */
6       public boolean contains( AnyType x )
7       {
8           int currentPos = findPos( x );
9           return isActive( currentPos );
10      }
11
12      /**
13       * Method that performs quadratic probing resolution.
14       * @param x the item to search for.
15       * @return the position where the search terminates.
16       */
17      private int findPos( AnyType x )
18      {
19          int offset = 1;
20          int currentPos = myhash( x );
21
22          while( array[ currentPos ] != null &&
23                  !array[ currentPos ].element.equals( x ) )
24          {
25              currentPos += offset;  // Compute ith probe
26              offset += 2;
27              if( currentPos >= array.length )
28                  currentPos -= array.length;
29          }
30
31          return currentPos;
32      }
33
34      /**
35       * Return true if currentPos exists and is active.
36       * @param currentPos the result of a call to findPos.
37       * @return true if currentPos is active.
38       */
39      private boolean isActive( int currentPos )
40      {
41          return array[ currentPos ] != null && array[ currentPos ].isActive;
42      }
```

图 5-16　使用平方探测进行散列的 contains 例程(及两个 private 型支撑方法)

第 25 行到第 28 行为进行平方探测的快速方法。由平方解决函数的定义可知，$f(i) = f(i-1) + 2i - 1$，因此，下一个要探测的单元离上一个被探测过的单元有一段距离，而这个距离在连续探测中增 2。如果新的定位越过数组，那么可以通过减去 *TableSize* 把它拉回到数组范围内。这比通常的方法要快，因为它避免了看似需要的乘法和除法。注意一条重要的警告：第 22 行和第 23 行的测试顺序很重要，切勿改变它！