整数集合(intset)是集合键的底层实现之一，当一个集合只包含整数值元素，并且这个集合的元素数量不多时，Redis就会使用整数集合作为集合键的底层实现。

intset可以保存类型为int16\_t，int32\_t，int64\_t的整数值，并且保证集合中不会出现重复元素。

整数集合的结构体定义在intset.h中：

**typedef** **struct** intset {

uint32\_t encoding;

uint32\_t length;

int8\_t contents[];

} intset;

encoding表示intset中保存的整数的编码，共有三种编码，分别是：INTSET\_ENC\_INT16、INTSET\_ENC\_INT32和INTSET\_ENC\_INT64。三种编码的定义如下：

#define INTSET\_ENC\_INT16 (sizeof(int16\_t))

#define INTSET\_ENC\_INT32 (sizeof(int32\_t))

#define INTSET\_ENC\_INT64 (sizeof(int64\_t))

注意，编码在保存时是按照小端的方式保存的，也就是说在大端系统中，还需要将实际的编码值翻转之后，才能存储到encoding中，比如：

is->encoding = intrev32ifbe(INTSET\_ENC\_INT16); 其中intrev32ifbe表示将32位整数按照字节进行翻转。

length表示intset中保存的整数个数。与encoding一样，它也是按照小端的方式保存的，在大端系统中，也需要将实际值翻转之后，才能存储到length中，比如：

is->length = intrev32ifbe(len-1)。

contents是柔性数组成员，它是整数集合的底层实现：整数集合中的每个元素都是contents数组的一个数组项，各个项在数组中按值从小到大有序地排列，并且数组中不包含任何重复项。

虽然contents的类型是int8\_t，但它可以保存类型为int16\_t，int32\_t，int64\_t的整数值。可以将contents想象为一段连续的内存空间，该空间的实际大小是encoding\*length。保存在contents中的整数，也是按照小端的方式保存的。

如果encoding属性的值为INTSET\_ENC\_INT16，那contents中保存的整数就是int16\_t类型，范围是[-32768, 32767]。如果encoding属性的值为INTSET\_ENC\_INT32，那contents中保存的整数就是int32\_t类型，范围是[-2147483648, 2147483647]。如果encoding属性的值为INTSET\_ENC\_INT64，那contents中保存的整数就是int64\_t类型，范围是[-9223372036854775808, 9223372036854775807]。

将一个新元素添加到intset，且新元素的类型比整数集合现有所有元素的类型都要大时，整数集合需要先进行升级，然后才能将新元素添加到整数集合里面。升级intset并添加新元素的代码实现如下：

static intset **\***intsetUpgradeAndAdd**(**intset **\***is**,** int64\_t value**)** **{**

uint8\_t curenc **=** intrev32ifbe**(**is**->**encoding**);**

uint8\_t newenc **=** \_intsetValueEncoding**(**value**);**

int length **=** intrev32ifbe**(**is**->**length**);**

int prepend **=** value **<** 0 **?** 1 **:** 0**;**

*/\* First set new encoding and resize \*/*

is**->**encoding **=** intrev32ifbe**(**newenc**);**

is **=** intsetResize**(**is**,**intrev32ifbe**(**is**->**length**)+**1**);**

*/\* Upgrade back-to-front so we don't overwrite values.*

*\* Note that the "prepend" variable is used to make sure we have an empty*

*\* space at either the beginning or the end of the intset. \*/*

**while(**length**--)**

\_intsetSet**(**is**,**length**+**prepend**,**\_intsetGetEncoded**(**is**,**length**,**curenc**));**

*/\* Set the value at the beginning or the end. \*/*

**if** **(**prepend**)**

\_intsetSet**(**is**,**0**,**value**);**

**else**

\_intsetSet**(**is**,**intrev32ifbe**(**is**->**length**),**value**);**

is**->**length **=** intrev32ifbe**(**intrev32ifbe**(**is**->**length**)+**1**);**

**return** is**;**

**}**

该函数只有在升级插入时才会调用，也就是说新值value超出了is现有的编码范围，因此，value要么插入在索引0，要么插入在索引is->length。

首先获取is当前的编码curenc，当前长度length，并根据value得到其合适的编码newenc，newenc成为is在升级插入value后的新编码；

prepend用于升级is时，定位contents中原有元素的新索引。如果value为正数，则value需要插入到索引is->length中，因此prepend为0，表明contents原有元素的索引值保持不变。注意这并不意味着contents的内容保持不变，因为编码升级了，每个元素所占的内存空间也发生了变化，因此还需要重新设置每个索引的元素；

若value为负数，则说明value要插入到索引0中，因此prepend为1，表明contents原有元素需要后移一个索引；

然后更新is的encoding为newenc，并调用intsetResize重新申请is的空间；根据新的编码newenc，将is的contents中的原有元素，根据prepend的值重新放置；

最后根据prepend值，将value插入到0或length处；并更新is的length属性。

注意，intset不支持降级操作，一旦对intset进行了升级，编码就会一直保持升级后的状态。比如一个intset保存了1,2,3,4四个数，这四个数使用编码INTSET\_ENC\_INT16即可，后插入了4294967295，intset升级编码为INTSET\_ENC\_INT64。现在如果再把4294967295删除了，则intset中，虽然还是1,2,3,4四个数，但是编码依然保持为INTSET\_ENC\_INT64。

有关intset其他的代码实现，可以参阅：

https://github.com/gqtc/redis-3.0.5/blob/master/redis-3.0.5/src/intset.c