**Sets**

**Set**: an unordered collection of an arbitrary number of distinct values of a common type.

**Characteristic array**: a bit vector whose length (in bits) is the number of distinct values of the base type. A one in the $k$th position in the bit vector indicated that the $k$th element of the base type is a memeber of the set; a zero indicated that it is not.

The type from which elements of a set are drawn is known as the **base** or **universe** type. In a language that uses ASCII, a set of characters would occupy 128 bits—16 bytes. Operations on bit-vector sets can make use of fast logical instructions on most machines. Union is bit-wise `or`; intersection is bit-wise `and`; difference is bit-wise `not`, followed by bit-wise `and`.

Unfortunately, bit vectors do not work well for large base types: a set of integers, represented as a bit vector, would consume some 500 megabytes on a 32-bit machine. With 64-bit integers, a bit-vector set would consume more memory than is currently contained on all the computers in the world.Because of this problem, some languages limited sets to base types of fewer than some fixed number of values.

For sets of elements drawn from a large universe, most modern languages use alternative implementations, whose size is proportional to the number of elements present, rather than to the number of values in the base type. Most languages also provide a built-in iterator to yield the elements of the set. A distinction is often made between **sorted lists**, whose base type must support some notion of ordering, and whose iterators yield the elements smallest-to-largest, and **unordered lists**, whose iterators yield the elements in arbitrary order. **Ordered sets** are commonly implemented with skip lists or various sorts of trees. **Unordered sets** are commonly implemented with hash tables.