Set class prototypes

Set classes maintain unbounded collections of items containing no duplicate elements.

These are currently implemented in several ways, differing in representation strategy, algorithmic efficiency, and appropriateness for various tasks. (Listed next to each are average (followed by worst-case, if different) time complexities for [a] adding, [f] finding (via seek, contains), [d] deleting, elements, and [c] comparing (via ==, <=) and [m] merging (via |=, -=, &=) sets).

XPSets	implement unordered sets via XPlexes. ([a O(n)], [f O(n)], [d O(n)], [c O(n^2)] [m O(n^2)]).
OXPSets	implement ordered sets via XPlexes. ([a O(n)], [f O(log n)], [d O(n)], [c O(n)] [m O(n)]).
SLSets	implement unordered sets via linked lists ([a O(n)], [f O(n)], [d O(n)], [c O(n^2)] [m O(n^2)]).
OSLSets	implement ordered sets via linked lists ([a O(n)], [f O(n)], [d O(n)], [c O(n)] [m O(n)]).
AVLSets	implement ordered sets via threaded AVL trees ([a O(log n)], [f O(log n)], [d O(log n)], [c O(n)] [m O(n)]).
BSTSets	implement ordered sets via binary search trees. The trees may be manually rebalanced via the O(n) **balance()** member function. ([a O(log n)/O(n)], [f O(log n)/O(n)], [d O(log n)/O(n)], [c O(n)] [m O(n)]).
SplaySets	implement ordered sets via Sleater and Tarjan's (JACM 1985) splay

trees. The algorithms use a version of ``simple top-down splaying'' (described on page 669 of the article). (Amortized: [a O(log n)], [f O(log n)], [d O(log n)], [c O(n)] [m O(n log n)]).

VHSets implement unordered sets via hash tables. The tables are automatically resized when their capacity is exhausted. ([a O(1)/O(n)], [f O(1)/O(n)], [d O(1)/O(n)], [c O(n)/O(n^2)] [m O(n)/O(n^2)]).

VOHSets implement unordered sets via ordered hash tables The tables are automatically resized when their capacity is exhausted. ([a O(1)/O(n)], [f O(1)/O(n)], [d O(1)/O(n)], [c O(n)/O(n^2)] [m O(n)/O(n^2)]).

CHSets implement unordered sets via chained hash tables. ([a O(1)/O(n)], [f O(1)/O(n)], [d O(1)/O(n)], [c O(n)/O(n^2)] [m O(n)/O(n^2)]).

The different implementations differ in whether their constructors require an argument specifying their initial capacity. Initial capacities are required for plex and hash table based Sets. If none is given **DEFAULT_INITIAL_CAPACITY** (from **<T>defs.h**) is used.

Sets support the following operations, for some class **Set**, instances **a** and **b**, **ix ind**, and base element **x**. Since all implementations are virtual derived classes of the **<T>Set** class, it is possible to mix and match operations across different implementations, although, as usual, operations are generally faster when the particular classes are specified in functions operating on Sets.

Pix-based operations are more fully described in the section on Pixes. See *Pseudo-indexes* in **/NextLibrary/Documentation/GNU/libg++/Intro.rtf**.

Set a; or **Set a(int initial_size);** Declares a to be an empty Set. The second version is allowed in set classes that require initial capacity or sizing specifications.

a.empty()	returns true if a is empty.
a.length()	returns the number of elements in a.
Pix ind = a.add(x)	inserts x into a, returning its index.
a.del(x)	deletes x from a.
a.clear()	deletes all elements from a;
a.contains(x)	returns true if x is in a.
a(ind)	returns a reference to the item indexed by ind.
ind = a.first()	returns the Pix of first item in the set or 0 if the Set is empty. For ordered Sets, this is the Pix of the least element.
a.next(ind)	advances ind to the Pix of next element, or 0 if there are no more.
ind = a.seek(x)	Sets ind to the Pix of x, or 0 if x is not in a.
a == b	returns true if a and b contain all the same elements.
a != b	returns true if a and b do not contain all the same elements.
a <= b	returns true if a is a subset of b.
a |= b	Adds all elements of b to a.
a -= b	Deletes all elements of b from a.

a &= b	Deletes all elements of a not occurring in b.