Map Class Prototypes

Maps support associative array operations (insertion, deletion, and membership of records based on an associated key). They require the specification of two types, the key type and the contents type.

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] accessing (via op [], contains), [d] deleting elements).

AVLMaps — implement ordered Maps via threaded AVL trees ([a O(log n)], [d O(log n)]).

RAVLMaps — Similar, but also maintain ranking information, used via **ranktoPix(int r)**, that returns the **Pix** of the item at rank r, and **rank(key)** that returns the rank of the corresponding item. ([a O(log n)], [d O(log n)]).

SplayMaps — implement ordered Maps 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)], [d O(log n)]).

VHMaps — implement unordered Maps via hash tables. The tables are automatically resized when their capacity is exhausted. ([a O(1)/O(n)], [d O(1)/O(n)]).

CHMaps — implement unordered Maps via chained hash tables. ([a O(1)/O(n)], [d O(1)/O(n)]).

The different implementations differ in whether their constructors require an argument specifying their initial

capacity. Initial capacities are required for hash table based Maps. If none is given **DEFAULT_INITIAL_CAPACITY** (from **<T>defs.h**) is used.

All Map classes share the following operations (for some Map class, **Map** instance **d**, **Pix ind** and key variable **k**, and contents variable **x**).

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

Map d(x); Map d(x, int initial_capacity) Declare d to be an empty Map. The required argument, x, specifies the default contents, i.e., the contents of an otherwise uninitialized location. The second version, specifying initial capacity is allowed for Maps with an initial capacity argument.

d.empty() returns true if d contains no items.

d.length() returns the number of items in d.

d[k] returns a reference to the contents of item with key k. If no such item exists, it is installed with the default contents. Thus d[k] = x installs x, and x = d[k] retrieves it.

d.contains(k) returns true if an item with key field k exists in d.

d.del(k) deletes the item with key k.

d.clear() deletes all items from the table.

x = d.dflt() returns the default contents.

k = d.key(ind)	returns a reference to the key at Pix ind.
x = d.contents(ind)	returns a reference to the contents at Pix ind.
ind = d.first()	returns the Pix of the first element in d, or 0 if d is empty.
d.next(ind)	advances ind to the next element, or 0 if there are no more.
ind = d.seek(k)	returns the Pix of element with key k, or 0 if k is not in d.