Lists - I

The List ADT

List ADT (expanded from Weiss)

- A list is a dynamic ordered tuple of homogeneous elements

  $A_1, A_2, A_3, \ldots, A_N$

  where A_i is the ith element of the list
- Definition: The *position* of element A_i is i; positions range from 1 to N inclusive
- Definition: The *size* of a list is N (a list of NO elements is called “an empty list”)

Other considerations

- What design considerations are there regarding this list.

1. Will the list hold an “infinite” number of elements, or will it have limited capacity? If so, what’s the maximum number of elements?

2. How will the list handle insertion of duplicate elements?

3. If the list allows insertion of duplicates, how does it handle deletion of a duplicated element?

Operations on a List

- List( ) -- construct an empty list
- List(const List &rhs) -- construct a list as a copy of rhs
- ~List( ) -- destroy the list
- const List &operator= (const List &rhs)
  - make this list contain copies of the elements of rhs in the same order
  - elements are deep copied from rhs, not used directly. If $L_1 = (A_1, A_2, A_3)$ and $L_2 = (B_1, B_2)$ before the assignment, then $L_2 = L_1$ causes $L_2 = (A_1, A_2, A_3)$

Operations on a List (cont)

- bool isEmpty( ) const -- returns true if the list size is zero
- void makeEmpty( ) -- causes the list to become empty
- void remove (const Object &x)
  - the first occurrence of x is removed from the list, if it is present. If x is not present, the list is unchanged.
  - an occurrence of x is an element A_i of the list such that $A_i == x$
- Also:
- insert
- find
- findPrevious

What's Missing?

- There is no **size( )** method that returns the size of the list
- There is no **retrieve(int i)** or **operator[int i]** method that access the i*th* element in the list

So, it's NOT possible to write code like this:

```
for (int i = 1; j < L.size( ); i++)
    cout << L.retrieve ( i );
```

How do we "scan" a list and look at all the elements, one at a time?

Iterators

- An *iterator* is an object that provides access to the elements of a collection (in a specified order) without exposing the underlying structure of the collection.
  - order dictated by the iterator
  - collection provides iterators on demand
  - each iterator on a collection is independent
  - iterator operations are generic

Iterator Operations

- bool isPastEnd( ) -- returns true if the iterator is past the end of the list
- void advance( ) -- advances the iterator to the next position in the list. If the iterator is already past the end, no change.
- const Object &retrieve( ) -- returns the element in the list at the current position of the iterator. It is an error to invoke retrieve( ) on an iterator that isPastEnd

List Operations

- ListIter<Object> first( ) -- returns an iterator representing the first element on the list

- ListIter<Object> zeroth( ) -- returns an iterator representing the header of a list

- ListIter<Object> find(const Object &x) -- returns an iterator representing the first occurrence of x in the list. If x not present, the iterator isPastEnd.

- ListIter<Object> findPrevious(const Object &x) -- returns an iterator representing the element before x in the list. If x is not in the list, the iterator represents the last element in the list. If x is first element (or list is empty), the iterator returned is equal to the one returned by zeroth( ).

“scanning” a Collection

```
Iterator iter = collection.first ( );
while (! iter.isPastEnd ( ) )
{
   Object x = iter.retrieve( ) ;
   // do something with x
   iter.advance ( );
}
```

List Operators (cont)

- void insert (const Object &x, const ListIter<Object> &p)
  - inserts a copy of x in the list ***after*** the element referred to by p
  - if p isPastEnd, the insertion fails without an indication of failure.

Ex: Building a List

```
List<int> list;  // empty list of int

ListIter<int> iter = list.zeroth( );

for (int i=0; i < 5; i++) {
  list.insert(i, iter);
  iter.advance( );
}
```

Ex: Building a List #2

```
List<int> list;  // empty list of int

ListIter<int> iter = list.zeroth( );

for (int i=0; i < 5; i++) {
  list.insert(i, iter);
}
```

More List Operations

- Find an element in the list and return it's position
- Return the value of the Nth element
- Determine if the list is full
- Determine if the list is empty
- Print the list