Lists

Chapter 8

Specifying the ADT List

- Things you make lists of
  - Chores
  - Addresses
  - Groceries
- Lists contain items of the same type
- Operations
  - Count items
  - Add, remove items
  - Retrieve

Specifying the ADT List

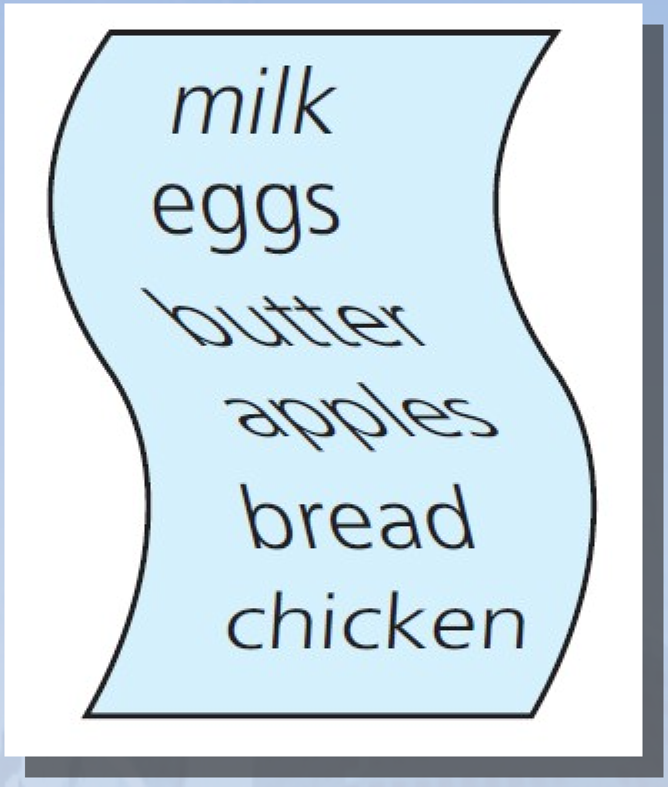
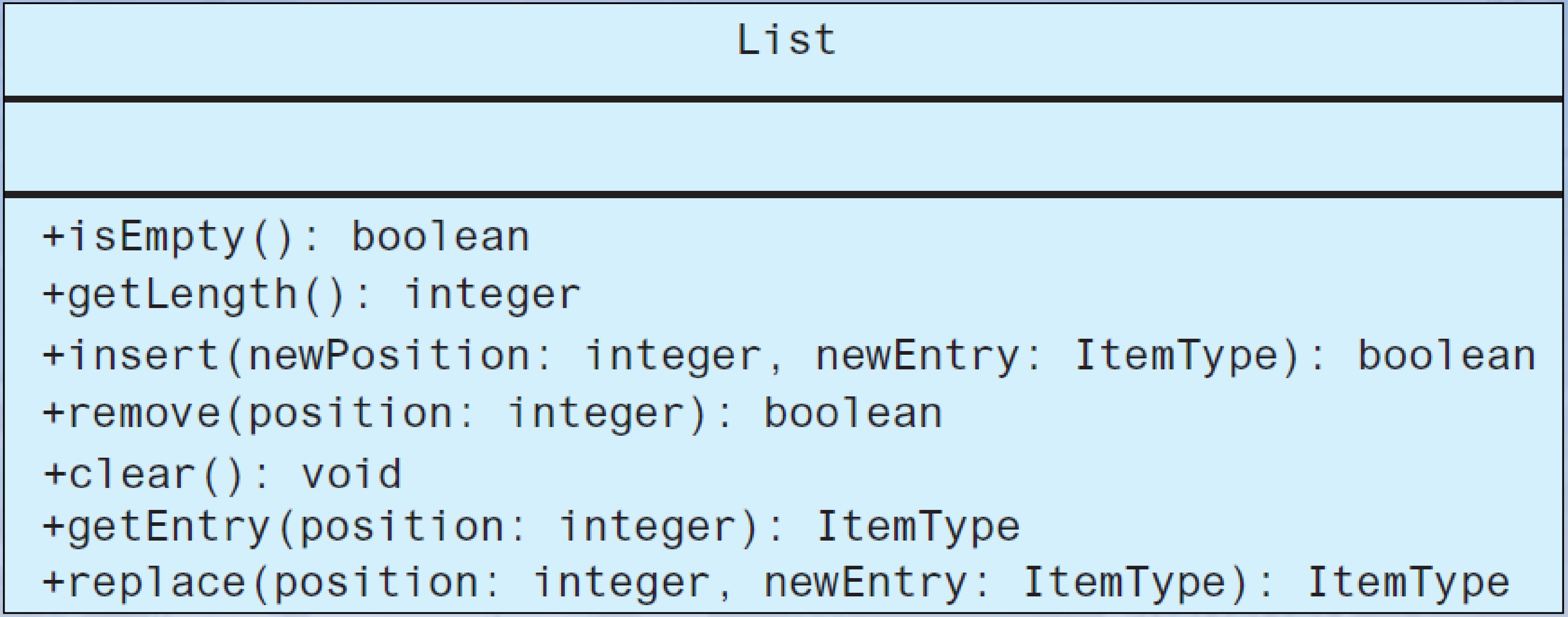


Figure 8-1 A grocery list

Specifying the ADT List



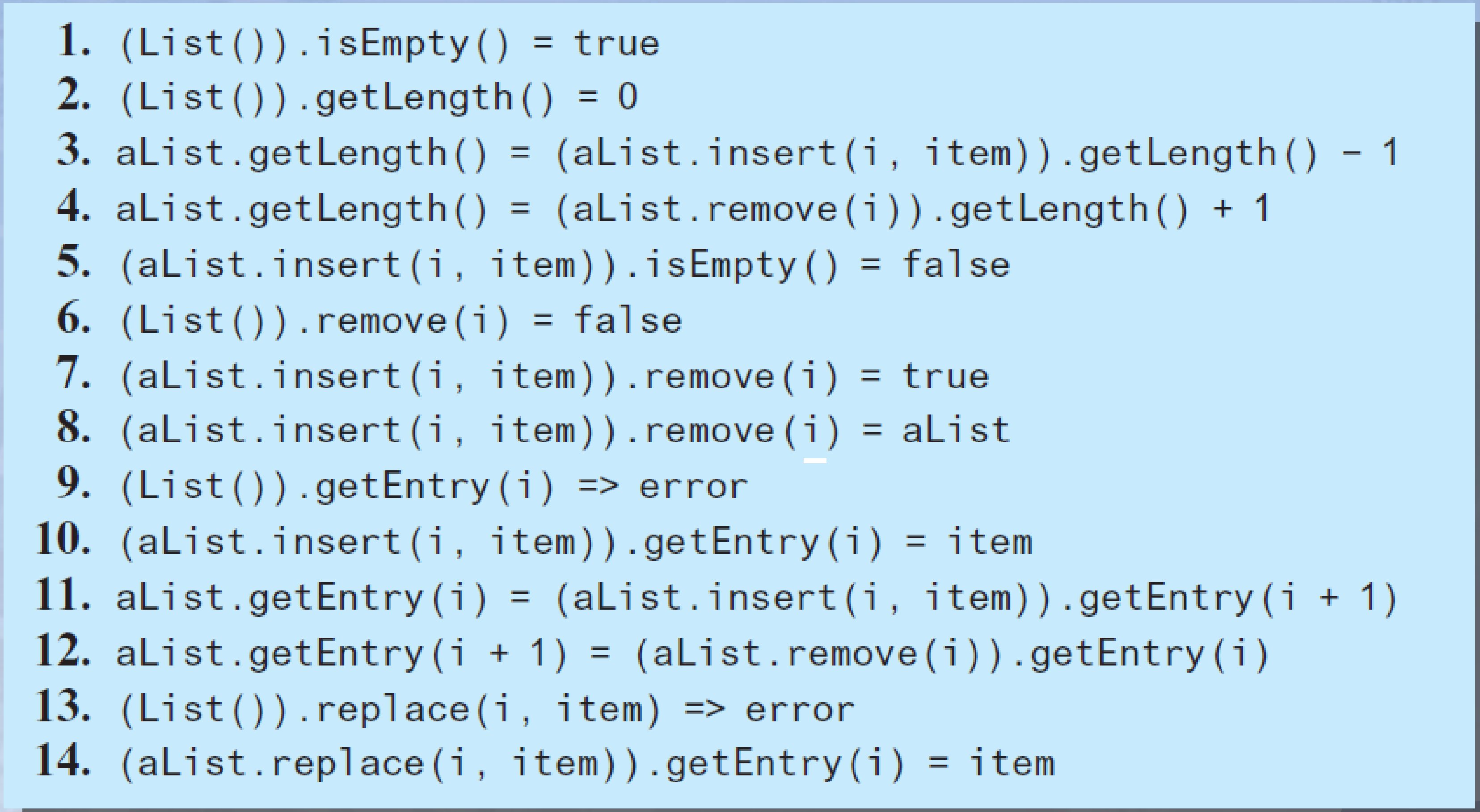
```
List
-------------------------------------------------------------------------

-------------------------------------------------------------------------
+isEmpty(): boolean
+getLength(): integer
+insert(newPosition: integer, newEntry: ItemType): boolean
+remove(position: integer): boolean
+clear(): void
+getEntry(position: integer): ItemType
+replace(position: integer, newEntry: ItemType): ItemType
```

FIGURE 8-2 UML diagram for the ADT list

Specifying the ADT List

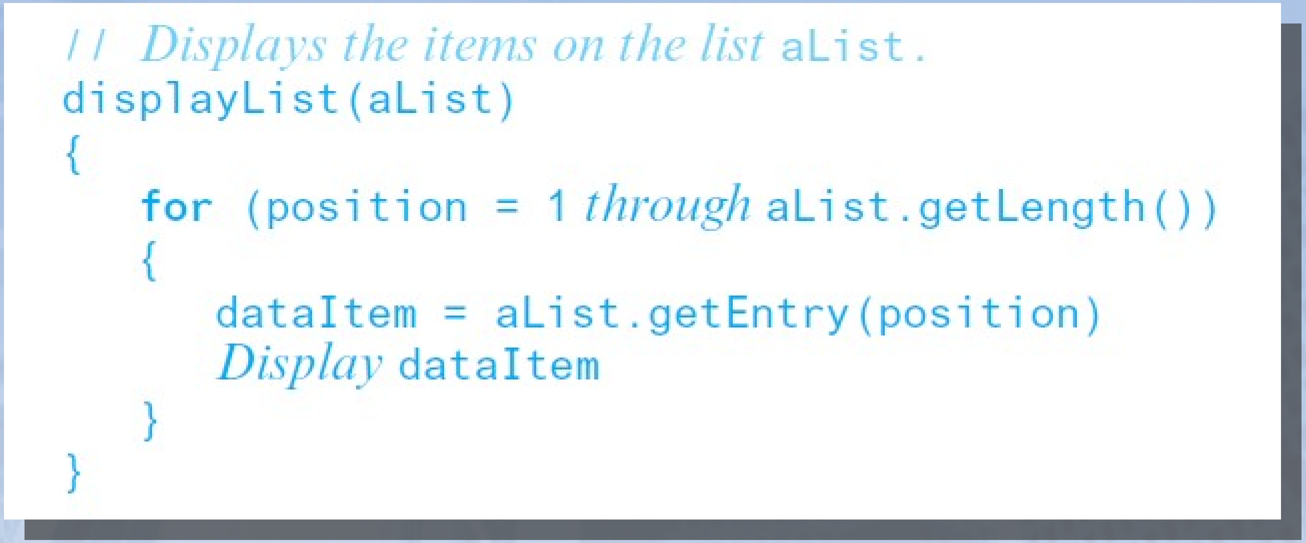
- Definition: ADT List
  - Finite number of objects
  - Not necessarily distinct
  - Same data type
  - Ordered by position as determined by client

Axioms for ADT List



1. `(List()).isEmpty() = true`
2. `(List()).getLength() = 0`
3. `aList.getLength() = (aList.insert(i, item)).getLength() - 1`
4. `aList.getLength() = (aList.remove(i)).getLength() + 1`
5. `(aList.insert(i, item)).isEmpty() = false`
6. `(List()).remove(i) = false`
7. `(aList.insert(i, item)).remove(i) = true`
8. `(aList.insert(i, item)).remove(i) = aList`
9. `(List()).getEntry(i) => error`
10. `(aList.insert(i, item)).getEntry(i) = item`
11. `aList.getEntry(i) = (aList.insert(i, item)).getEntry(i + 1)`
12. `aList.getEntry(i + 1) = (aList.remove(i)).getEntry(i)`
13. `(List()).replace(i, item) => error`
14. `(aList.replace(i, item)).getEntry(i) = item`

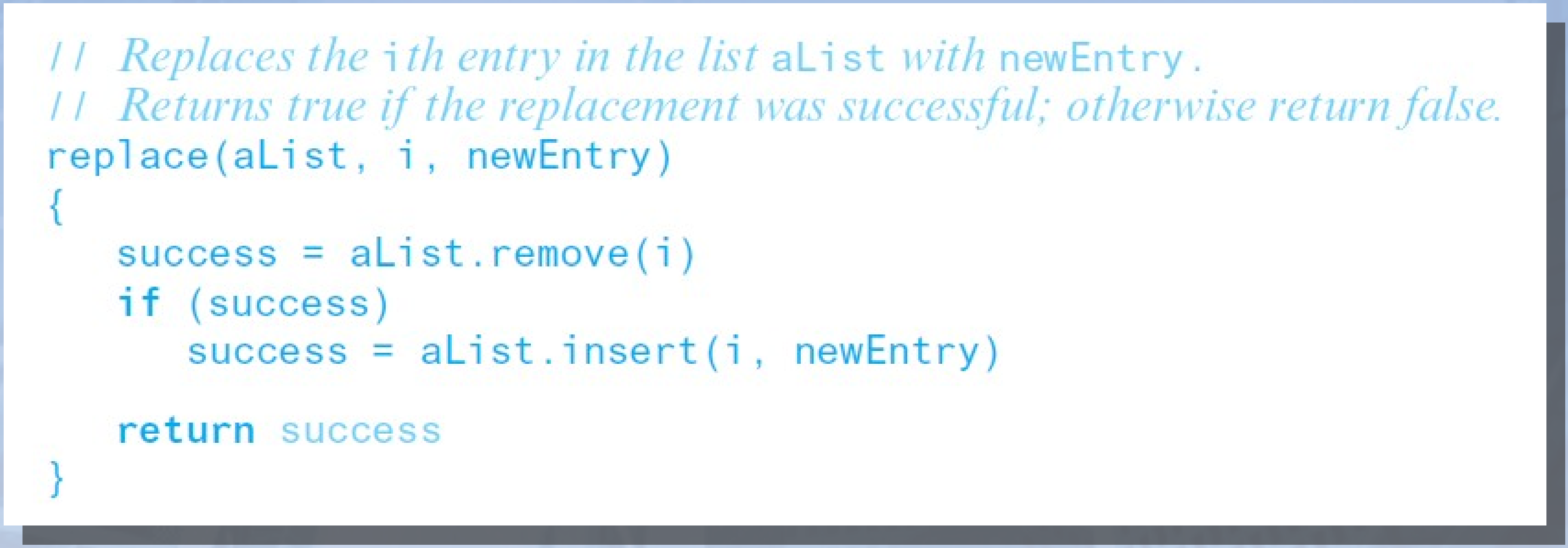
Using the List Operations



```
// Displays the items on the list aList.
displayList(aList)
{
   for (position = 1 through aList.getLength())
   {
      dataItem = aList.getEntry(position)
      Display dataItem
   }
}
```

Displaying the items on a list.

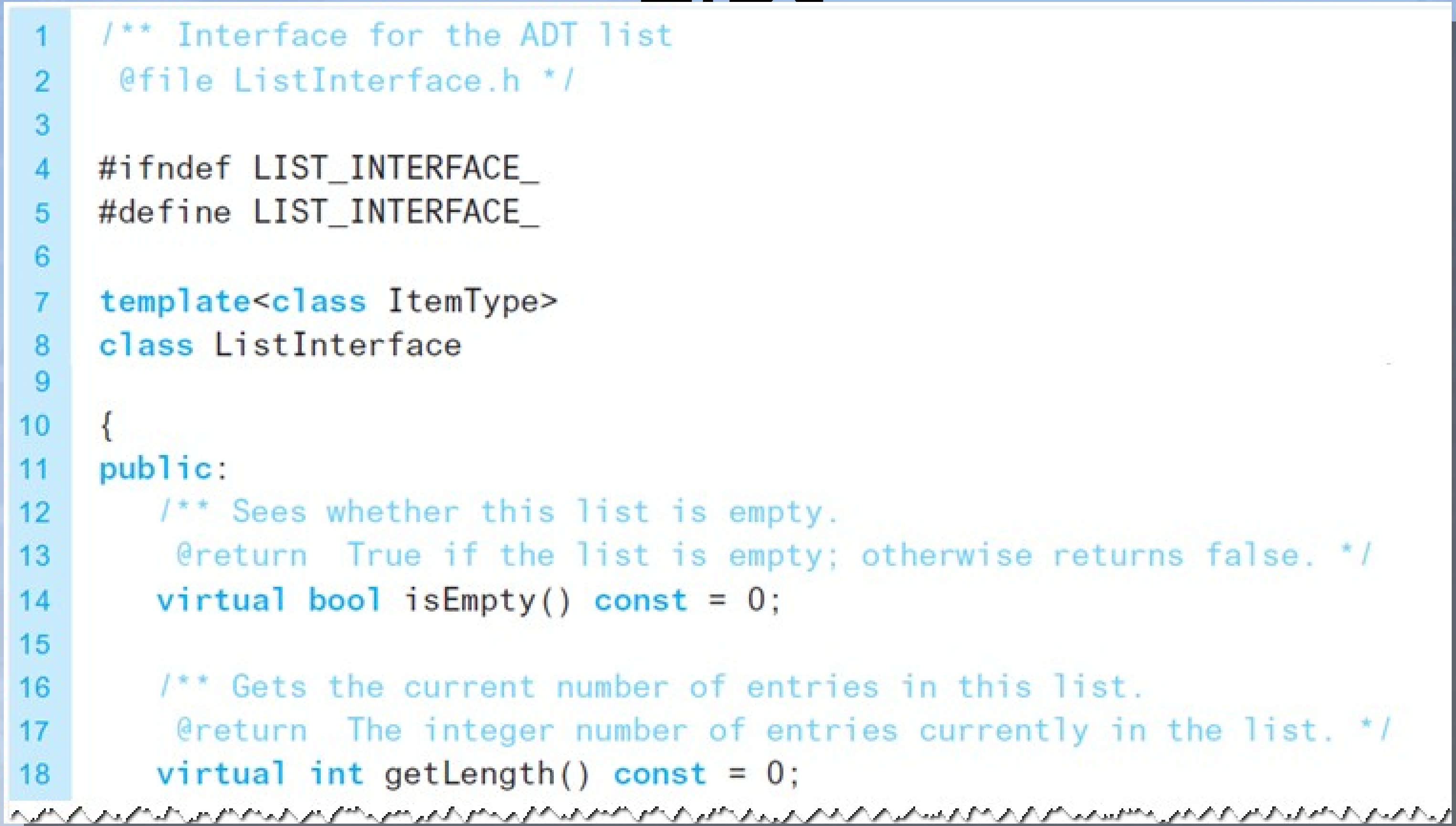
Using the List Operations



```
// Replaces the ith entry in the list aList with newEntry.
// Returns true if the replacement was successful; otherwise return false.
replace(aList, i, newEntry)
{
   success = aList.remove(i)
   if (success)
      success = aList.insert(i, newEntry)

   return success
}
```

Replacing an item.

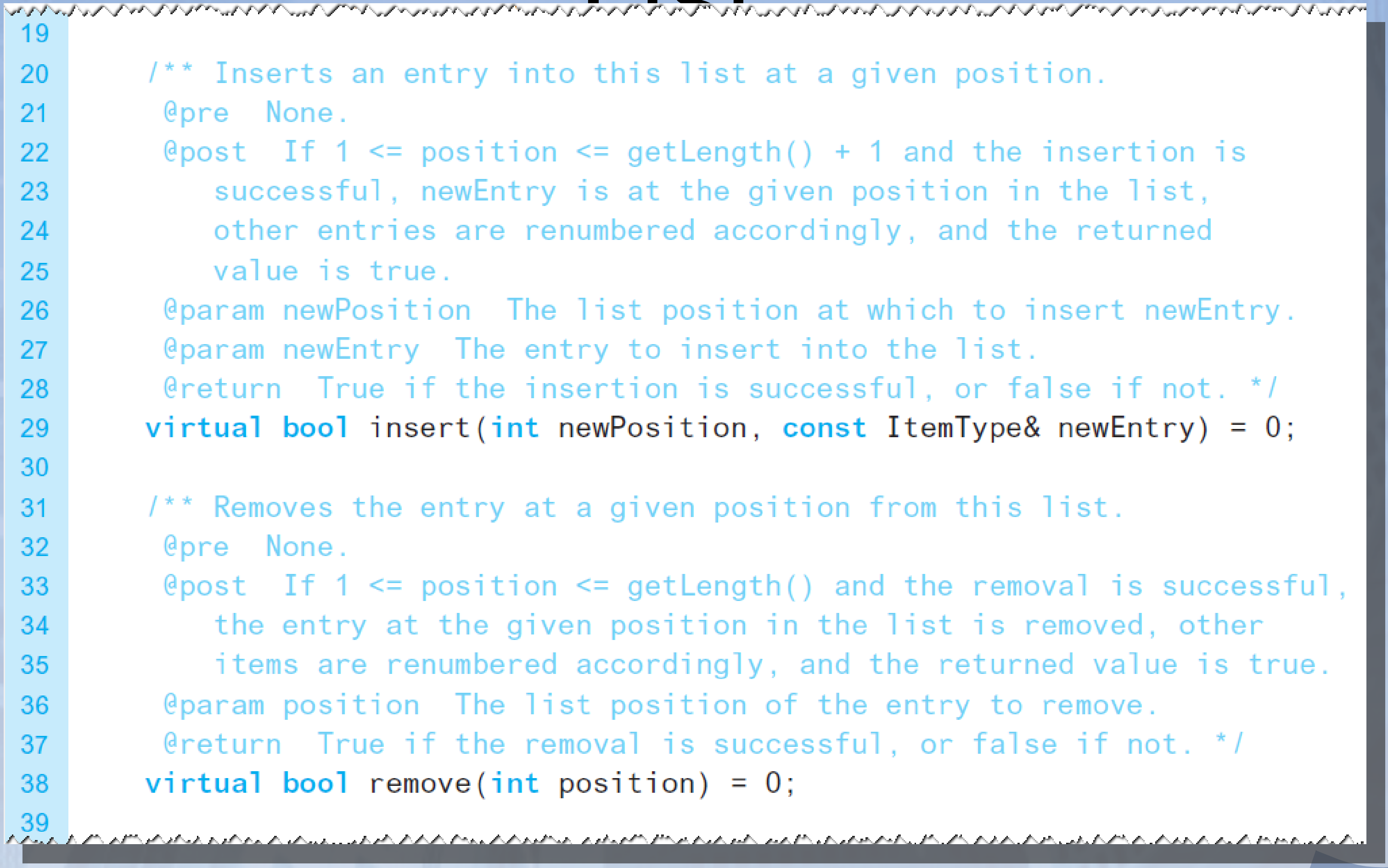
Interface Template for ADT List



```
/** Interface for the ADT list
 @file ListInterface.h */

#ifndef LIST_INTERFACE_
#define LIST_INTERFACE_

template<class ItemType>
class ListInterface

{
public:
   /** Sees whether this list is empty.
    @return  True if the list is empty; otherwise returns false. */
   virtual bool isEmpty() const = 0;

   /** Gets the current number of entries in this list.
    @return  The integer number of entries currently in the list. */
   virtual int getLength() const = 0;
```

LISTING 8-1 A C++ interface for lists

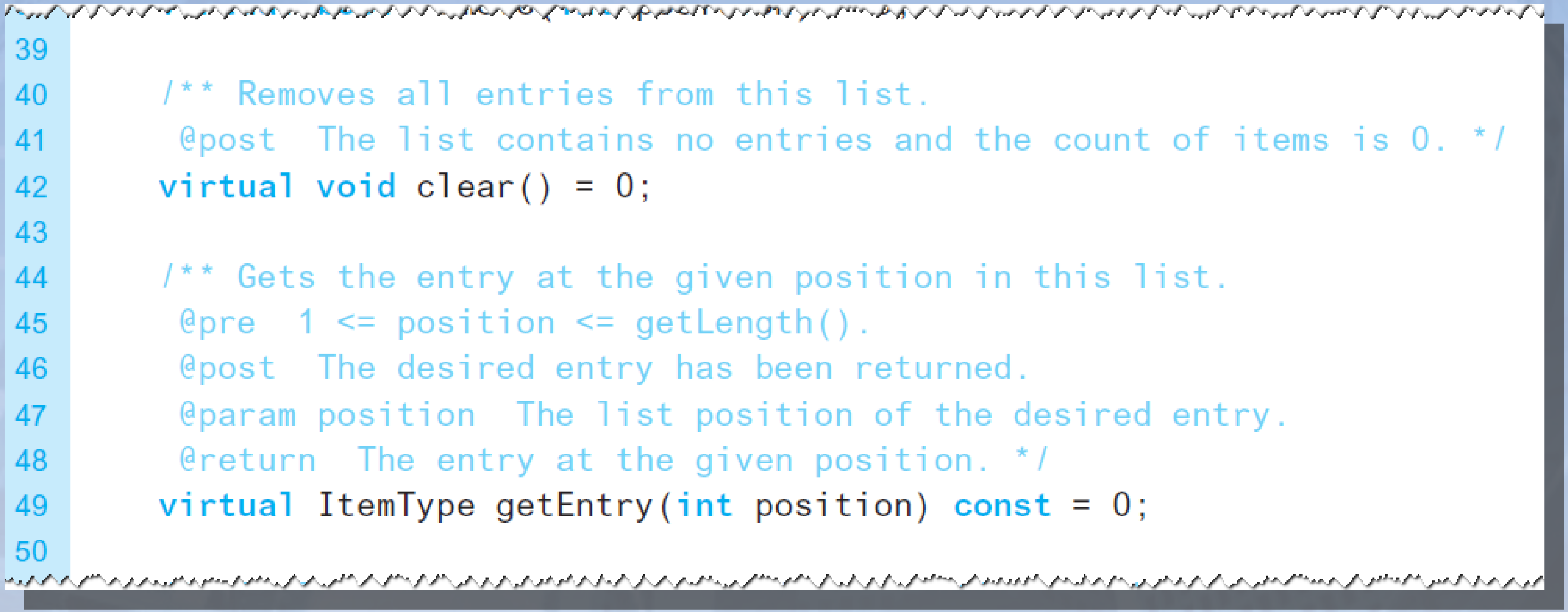
Interface Template for ADT List



```
    /** Inserts an entry into this list at a given position.
     @pre  None.
     @post  If 1 <= position <= getLength() + 1 and the insertion is
        successful, newEntry is at the given position in the list,
        other entries are renumbered accordingly, and the returned
        value is true.
     @param newPosition  The list position at which to insert newEntry.
     @param newEntry  The entry to insert into the list.
     @return  True if the insertion is successful, or false if not. */
    virtual bool insert(int newPosition, const ItemType& newEntry) = 0;

    /** Removes the entry at a given position from this list.
     @pre  None.
     @post  If 1 <= position <= getLength() and the removal is successful,
        the entry at the given position in the list is removed, other
        items are renumbered accordingly, and the returned value is true.
     @param position  The list position of the entry to remove.
     @return  True if the removal is successful, or false if not. */
    virtual bool remove(int position) = 0;
```

LISTING 8-1 A C++ interface for lists

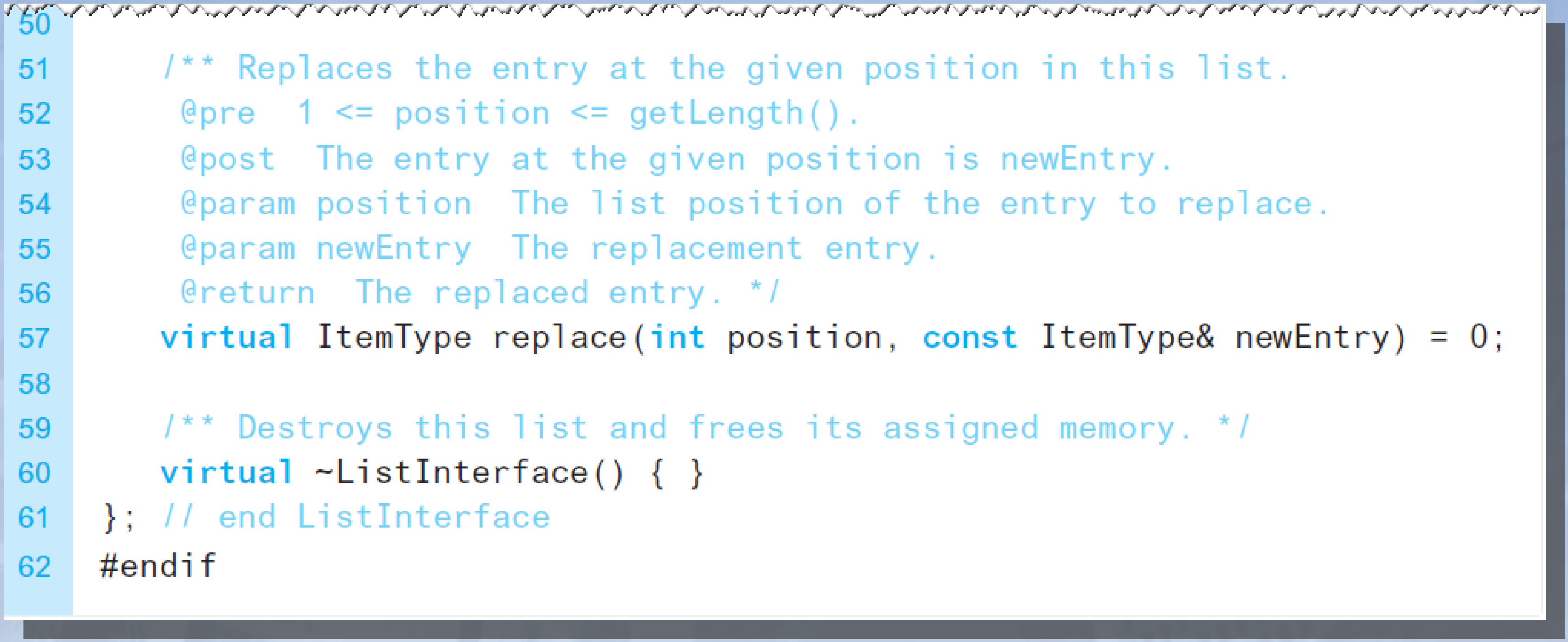
Interface Template for ADT List



```
    /** Removes all entries from this list.
     @post  The list contains no entries and the count of items is 0. */
    virtual void clear() = 0;

    /** Gets the entry at the given position in this list.
     @pre  1 <= position <= getLength().
     @post  The desired entry has been returned.
     @param position  The list position of the desired entry.
     @return  The entry at the given position. */
    virtual ItemType getEntry(int position) const = 0;

```

LISTING 8-1 A C++ interface for lists

Interface Template for ADT List



```cpp

   /** Replaces the entry at the given position in this list.
    @pre  1 <= position <= getLength().
    @post  The entry at the given position is newEntry.
    @param position  The list position of the entry to replace.
    @param newEntry  The replacement entry.
    @return  The replaced entry. */
   virtual ItemType replace(int position, const ItemType& newEntry) = 0;

   /** Destroys this list and frees its assigned memory. */
   virtual ~ListInterface() { }
}; // end ListInterface
#endif
```

LISTING 8-1 A C++ interface for lists

End

Chapter 8