**Lists**

**Real-Life Examples of Lists**

* Consider a list in real life that you might encounter, such as
  + a list of chores,
  + a list of important dates,
  + a list of addresses,
  + or a grocery list
* We will use the example of a grocery list to illustrate how lists work.
* As you write a grocery list, where do you put new items?
  + - You could add new items to the end (bottom) of the list
    - You could add new items to the beginning of the list
    - You could write down items in alphabetical order
    - You could write them randomly for all you care! (but this wouldn’t be very coherent or easy to understand)

Text

Description automatically generated

**Sequence Containers**

* Regardless of the order, the items on a list appear in some sort of an organized **sequence**.
* For a **sequence container**, the
  + **position** (index) of an element

depends on

* + the **order** **(time and place)** of insertion.
* An element’s position is **independent of the** **value** of the element.
* This means that elements are inserted into a container based on **time** (i.e., item1 was inserted before item2) or **place** (i.e., front or end of container).
* The six items in the Figure 8-1 list have some sort of **sequential order**
* The kind of sequence they are organized by, is dependent on how you want to order the items.
  + Are items inserted at the front of the list?
  + Are items inserted at the back of the list?
  + Etc…

**Lists Items**

* A list has
  + A single first item
  + A single last item
  + Items in between the first and last items
* The first item—the **head** or **front** of the list—does not have a predecessor
* The last item—the **tail** or end of the list—does not have a successor
* Each item in the middle of the list (between the first and last items) has a unique **predecessor** and a unique **successor**.

**List Operations**

* Lists contain items that are all of the same type
* We can perform a number of actions on the items in the list.
  + You could **count** the items on the list,
  + You could **add** an item to the list,
  + You could **remove** an item from the list,
  + You could **get** (retrieve) an item,
  + Etc.

**ADT List**

* An Abstract Data Type (ADT): **ADT** **List** is formed by
  + The items in a list
  + The operations that you can perform on the items

**ADT List Operations**

* Test whether a list is empty.
* Get the number of entries on a list.
* Insert an entry at a given position on the list.
* Remove the entry at a given position from the list.
* Remove all entries from the list.
* Look at (get) the entry at a given position on the list.
* Replace (set) the entry at a given position on the list.

Here is the **interface** for the previous operations we specified for the ADT List:

Text

Description automatically generated with medium confidence

**List insert()**

We have 3 options when inserting data into a **List**.

1. Insert an item at the beginning of a List
2. Insert an item at the end of a List
3. Insert an item in the middle of a List (with some specified index of where to insert it)

**Example:**

* Let’s try to insert the following grocery items:

*milk, eggs, butter, apples, bread, chicken*

* Milk will be the first item on the list
* Chicken will be the last item on the list
* To begin, consider how you can construct this list by using the ADT list operations.
* One way is first to

1. Declare an empty List
2. Use a series of insertion operations to append items to the List one at a time

aList = *a new empty list*

aList.insert(1, milk)

aList.insert(2, eggs)

aList.insert(3, butter)

aList.insert(4, apples)

aList.insert(5, bread)

aList.insert(6, chicken)

* We assume for this simple example that the list can contain all of the items we add to it.
* In the previous example, we in effect have inserted each new item at the end of the list.
* Nevertheless, the list’s insertion operation can place new items into any position of the list, not just at its front or end.
* The effect of an insertion between existing items, however, is not apparent from the previous example.
* For instance, if you start with the previous grocery list and you perform the operation aList.insert(4, nuts)