List Processing

Lecture 30
Sections 8.5, 8.6

Robb T. Koether

Hampden-Sydney College

Wed, Nov 13, 2019

Outline

List Processing

- There is a variety of things we might want to do with a list.
  - Find the largest or smallest element.
  - Verify that the elements are in order.
  - Delete an element from the list.
  - Insert an element into an ordered list.
  - Search the list for a specified value.
  - Sort the list, i.e., arrange, the elements into increasing or decreasing order.

Outline

Largest Element

Find the Largest Element

- Find the largest element in an array.
- The function should return
  - The largest value.
  - Or, the index of the largest value.
- Which is the better choice?
- At least one, but preferably all, of the order operators <, >, <=, >= must be defined on the data type.
- What if the array is empty?

Find the Maximum in an Array

Find the Maximum in an Array

- The Problem
  - Find the largest element in an array.
- The Algorithm
  - Initialize `max_so_far` to the first element.
  - Make 1 pass down the list, starting with the second element.
  - Compare `max_so_far` to each element from positions `1` to $n - 1$.
  - Replace `max_so_far` with the larger value whenever one is found.

Outline

Verify Order

Verify the Order

- Verify that the elements of the array are in order (ascending or descending).
- The function should return **`true`** or **`false`**, indicating whether the elements are in order.
- At least one of the order operators <, >, <=, >= must be defined on the data type.

Verify that an Array is in Order

Verify that an Array is in Order

- The Problem
  - Verify that the elements of an array are in ascending order.
- The Algorithm
  - Make 1 pass down the list.
  - Compare each member, except the last, to its immediate successor.
  - If a pair is found out of order, return **`false`** (immediately).
  - If no pair is found out of order, return **`true`**.

Outline

Delete

Delete an Element

- Delete an element from an array.
- The function should return **`true`** or **`false`**, indicating whether the value was successfully deleted, or it could be a **`void`** function.
  - What if the item was not found?
  - What if the array is empty?
  - What if the value occurs more than once in the array?

Delete an Element from an Array

Delete an Element from an Array

- The Problem
  - Delete an value from an unordered array.
- The Algorithm
  - Start at the beginning of the list (position `0`).
  - Compare the value to each element (indexes `0` through `n - 1`).
  - If they do not match, then move to the next element.
  - If they match, then continue down the list, copying each subsequent element to the previous position to plug up the hole left by the deletion.

Delete an Element from an Array

Delete an Element from an Array

- If the array is ordered, then the search may stop as soon as the value to be deleted is smaller than the element to which it is compared.

Outline

Insert

Insert an Element

- Insert an element into an ordered array.
- The function should return **`true`** or **`false`**, indicating whether the value was successfully inserted, or it could be a **`void`** function.
  - What if the value is already in the list?
  - What if the array if full?
- At least one of the order operators <, >, <=, >= must be defined on the data type.

Insert an Element into an Array

Insert an Element into an Array

- The Problem
  - Insert a value into its proper place in an ordered array.
- The Algorithm
  - Start at the end (position $n - 1$).
  - Moving towards the beginning, compare the value to be inserted to each element in the list.
  - If the value is smaller, shift the element up one position in the array.
  - If the value is larger, put it in the previous position.

Insert an Element into an Array

Insert an Element into an Array

- If the list is unordered, then the value may be inserted in the last open position (position n).

Outline

Search

Search an Array

- Search a list for a value.
- The function should return
  - `true` or `false`, indicating *whether* the value was found.
  - Or, the index of the position *where* the value was found.
  - Or, both.
- The list may be sorted or unsorted.

Outline

Sort

Sort an Array

- Arrange the elements in order.
- The function should be **`void`**.
- At least one of the order operators <, >, <=, >= must be defined on the data type.

Outline

Example of Array Processing

- Examples
  - `AverageAge.cpp`
  - `Date.cpp`

Outline

Assignment

Assignment

- Read Sections 8.5, 8.6.