Sort

primera

Generated by Doxygen 1.8.15

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

Class Documentation

2.1 ArrayList< T > Class Template Reference

Public Member Functions

- ArrayList ()
- ArrayList (int theSize)
- ArrayList (const ArrayList< T > &list)
- ArrayList (const ArrayList< T > &list, int theCapacity)
- ~ArrayList ()
- void add (const T &element)
- void add (const T &element, int index)
- bool agrega_ordena (const T &element)
- void add (const ArrayList< T > &list, int index)
- void add (const ArrayList< T > &list)
- bool eliminar_elemento (const T &element)
- const T ∗ eliminar (int index)
- void **eliminar_todo** ()
- void **eliminar_todo** (int newCapacity)
- bool **move** (int index1, int index2)
- const T ∗ **get** (int index) const
- const ArrayList< T > & **subList** (int index1, int index2) const
- T ∗ **getArray** () const
- T ∗ **getSubArray** (int index1, int index2) const
- int **binarySearch** (const T &element) const
- void **quickSort** ()
- void **SelectionSort** ()
- void **mergeSort** ()
- void **shuffle** ()
- int **Obt_tamano** () const
- int **Obt_capacidad** () const
- std::string **toString** () const

2.1.1 Constructor & Destructor Documentation

2.1.1.1 ArrayList() [1/4]

```
template<class T >
ArrayList< T >::ArrayList ( )
```

Default constructor, creates a 20 element ArrayList, of type T.

2.1.1.2 ArrayList() [2/4]

```
template<class T >
ArrayList< T >::ArrayList (
             int theSize )
```

Creates an ArrayList of type T of size theSize.

Parameters

theSize	the size to initialize the array to.

2.1.1.3 ArrayList() [3/4]

```
template<class T >
ArrayList< T >::ArrayList (
             const ArrayList< T > & list )
```

Creates an ArrayList of type T that is twice the size of the passed in ArrayList, and adds all elements from the passed ArrayList<T> list, to this ArrayList.

Runs in O(n) time, where n = the size of the passed list.

Parameters

list	the ArrayList to use as a seed for this ArrayList.

2.1.1.4 ArrayList() [4/4]

```
template<class T >
ArrayList< T >::ArrayList (
             const ArrayList< T > & list,
             int theCapacity )
```

Creates an ArrayList of type T that has a capacity equal to the passed in theCapacity parameter. This ArrayList starts with the passed ArrayList.

Note: If the passed in capacity is smaller than the size of the passed in ArrayList, then the capacity is set to twice the size of the passed ArrayList.

Runs in O(n) time where n is the size of the passed list.

Parameters

list	the ArrayList to use as a seed for this ArrayList.
theCapacity	the capacity for this ArrayList.

2.1.1.5 ∼ArrayList()

```
template<class T >
ArrayList< T >::∼ArrayList ( )
```

General destructor, deallocates the array.

2.1.2 Member Function Documentation

2.1.2.1 add() [1/4]

```
template<class T >
void ArrayList< T >::add (
             const T & element )
```

Adds the passed in element to the end of the ArrayList.

Runs in O(n) in worst case, where reallocate is called. O(1) for most cases.

Parameters

element	the element to add to the array.

2.1.2.2 add() [2/4]

```
template<class T >
void ArrayList< T >::add (
             const T & element,
             int index )
```

Adds the passed in element to the specified index. Provided that the index is valid.

A valid index is: 0 <= index <= size

Runs in O(n) where n is either the number of elements that must be shifted to fit element in index, or the size of the array if the array has to be reallocated.

Parameters

element	the element to add to the ArrayList.
index	the index to add the element to.

2.1.2.3 add() [3/4]

```
template<class T >
void ArrayList< T >::add (
            const ArrayList< T > & list,
            int index )
```

Adds an ArrayList<T> to this ArrayList<T> at the specified index.

Note: If index is larger than the capacity of the ArrayList, the ArrayList is enlarged to a size that is two times the index plus the size of the arrayList. If you want a check on the index size, use Obt_tamano() to check.

Runs in O(n) time where n is the size of the passed in list, or the number of elements which are shifted in this list. Whichever one is larger.

Parameters

list	the ArrayList<T> to add to this one.
index	the index to add the passed in ArrayList<T> at.

2.1.2.4 add() [4/4]

```
template<class T >
void ArrayList< T >::add (
            const ArrayList< T > & list )
```

Adds an ArrayList<T> to the end of this ArrayList.

Runs in O(n) time, where n = the size of the passed list.

Parameters

list	the ArrayList<T> to add to this one.

2.1.2.5 agrega_ordena()

```
template<class T >
bool ArrayList< T >::agrega_ordena (
            const T & element )
```

Takes in an element of type T and adds it to the correct point in the ArrayList such that the sort is preserved.

NOTE: If the ArrayList is NOT sorted, the element is not inserted, and false is returned.

Runs in O(n) time where n is the number of elements that must be shifted to accommodate the inserted element.

Parameters

element	the element to insert.

Returns

true if ArrayList was sorted (meaning the element was inserted), false if the ArrayList was NOT sorted (the element was NOT inserted)

2.1.2.6 eliminar()

```
template<class T >
const T * ArrayList< T >::eliminar (
             int index )
```

Removes the element at the passed in index.

Runs in O(n) time where n is the number of elements that must be shifted.

Parameters

index	the index of the element to eliminar.

Returns

eliminard a pointer to the eliminard element, or NULL/0 if the index is not valid.

2.1.2.7 eliminar_elemento()

```
template<class T >
bool ArrayList< T >::eliminar_elemento (
             const T & element )
```

Removes the passed in element from the ArrayList if it's present. Returns a boolean value indicating if the element was present in the array.

Runs in O(n) time where n is the number of elements that must be shifted to fill the hole.

Parameters

element	the element to eliminar from the array.

Returns

value true if the element was present, false otherwise.

The documentation for this class was generated from the following files:

- ArrayList.h
- binarysearch.h
- mergesort.h
- quicksort.h

Index