

AlgorithmAnalysis

1.1

Generated by Doxygen 1.7.5.1

Sun Dec 16 2012 15:55:05

Contents

1	File Index	1
1.1	File List	1
2	File Documentation	3
2.1	include/aux.h File Reference	3
2.1.1	Function Documentation	3
2.1.1.1	checkArrayEquality	3
2.1.1.2	checkIsOrdered	3
2.1.1.3	copyArray	3
2.1.1.4	createRandomArray	3
2.1.1.5	printArrayContents	3
2.2	include/generators.h File Reference	4
2.2.1	Function Documentation	4
2.2.1.1	generateAllFiles	4
2.2.1.2	generateBubbleSortDataFile	4
2.2.1.3	generateBubbleSortTime	4
2.2.1.4	generateInsertionSortDataFile	4
2.2.1.5	generateInsertionSortTime	4
2.2.1.6	generateQuickSortDataFile	4
2.2.1.7	generateQuickSortTime	4
2.2.1.8	generateSelectionSortDataFile	4
2.2.1.9	generateSelectionSortTime	4
2.3	include/interaction.h File Reference	4
2.3.1	Function Documentation	5
2.3.1.1	chooseAlgorithm	5
2.3.1.2	getGap	5

2.3.1.3	getMaximumInteger	5
2.3.1.4	getProblemSize	5
2.4	include/ordenacion.h File Reference	5
2.4.1	Function Documentation	5
2.4.1.1	bubbleSort	5
2.4.1.2	insertionSort	5
2.4.1.3	quickSort	6
2.4.1.4	selectionSort	6
2.5	include/test.h File Reference	6
2.5.1	Function Documentation	6
2.5.1.1	generateArray	6
2.5.1.2	makeFourCopies	7
2.6	src/aux.cpp File Reference	7
2.6.1	Function Documentation	7
2.6.1.1	checkArrayEquality	7
2.6.1.2	checkIsOrdered	7
2.6.1.3	checkSameLength	7
2.6.1.4	copyArray	7
2.6.1.5	createRandomArray	7
2.6.1.6	printArrayContents	7
2.7	src/generators.cpp File Reference	7
2.7.1	Function Documentation	8
2.7.1.1	generateAllFiles	8
2.7.1.2	generateBubbleSortDataFile	8
2.7.1.3	generateBubbleSortTime	8
2.7.1.4	generateInsertionSortDataFile	8
2.7.1.5	generateInsertionSortTime	8
2.7.1.6	generateQuickSortDataFile	8
2.7.1.7	generateQuickSortTime	8
2.7.1.8	generateSelectionSortDataFile	8
2.7.1.9	generateSelectionSortTime	8
2.8	src/interaction.cpp File Reference	8
2.8.1	Function Documentation	9
2.8.1.1	chooseAlgorithm	9

2.8.1.2	getGap	9
2.8.1.3	getMaximumInteger	9
2.8.1.4	getProblemSize	9
2.9	src/main.cpp File Reference	9
2.9.1	Function Documentation	9
2.9.1.1	main	9
2.10	src/ordenacion.cpp File Reference	9
2.10.1	Function Documentation	10
2.10.1.1	bubbleSort	10
2.10.1.2	insertionSort	10
2.10.1.3	merge	10
2.10.1.4	quickSort	10
2.10.1.5	selectionSort	10
2.11	src/test.cpp File Reference	11
2.11.1	Function Documentation	11
2.11.1.1	generateArray	11
2.11.1.2	makeFourCopies	11

Chapter 1

File Index

1.1 File List

Here is a list of all files with brief descriptions:

include/ aux.h	3	include/aux.h File Reference	section.2.1
include/ generators.h	4	include/generators.h File Reference	section.2.2
include/ interaction.h	4	include/interaction.h File Reference	section.2.3
include/ ordenacion.h	5	include/ordenacion.h File Reference	section.2.4
include/ test.h	6	include/test.h File Reference	section.2.5
src/ aux.cpp	7	src/aux.cpp File Reference	section.2.6
src/ generators.cpp	7	src/generators.cpp File Reference	section.2.7
src/ interaction.cpp	8	src/interaction.cpp File Reference	section.2.8
src/ main.cpp	9	src/main.cpp File Reference	section.2.9
src/ ordenacion.cpp	9	src/ordenacion.cpp File Reference	section.2.10
src/ test.cpp	11	src/test.cpp File Reference	section.2.11

Chapter 2

File Documentation

2.1 include/aux.h File Reference

```
#include <time.h> #include <cstdlib> #include <iostream> ×  
#include "../include/ordenacion.h" #include <fstream> ×  
#include <iomanip>
```

Functions

- void **createRandomArray** (int V[], int arrayLength, int maxInteger)
Creates a random integer array between 0 and MaxInteger.
- void **copyArray** (int V[], int W[], int n)
Copies an array contents into another array with the same length.
- bool **checkArrayEquality** (int V[], int W[], int n)
Checks equality between two given arrays (both arrays have same content in same order)
- bool **checkIsOrdered** (int V[], int n)
Checks if an array is ordered.
- void **printArrayContents** (int V[], int n)
Prints array content on console.

2.1.1 Function Documentation

2.1.1.1 bool checkArrayEquality (int V[], int W[], int n)

Checks equality between two given arrays (both arrays have same content in same order)

Parameters

V	the original array
W	the other array
n	the array's length

2.1.1.2 `bool checksOrdered (int $V[]$, int n)`

Checks if an array is ordered.

Parameters

V	the array
n	the array's length

2.1.1.3 `void copyArray (int $V[]$, int $W[]$, int n)`

Copies an array contents into another array with the same length.

Parameters

V	the original array
W	the cloned array
n	the array's length

2.1.1.4 `void createRandomArray (int $V[]$, int $arrayLength$, int $maxInteger$)`

Creates a random integer array between 0 and `MaxInteger`.

Parameters

V	an array already set to a length of $arrayLength$
$arrayLength$	the array's length
$maxInteger$	the maximum integer to generate

2.1.1.5 `void printArrayContents (int $V[]$, int n)`

Prints array content on console.

Parameters

V	the array
n	the array's length

2.2 include/generators.h File Reference

```
#include "aux.h"
```

Functions

- void **generateInsertionSortTime** (ofstream &file, int V[], int n)
- void **generateSelectionSortTime** (ofstream &file, int V[], int n)
- void **generateBubbleSortTime** (ofstream &file, int V[], int n)
- void **generateQuickSortTime** (ofstream &file, int V[], int n)
- void **generateInsertionSortDataFile** (int problemSize, int V[], int GAP)
- void **generateSelectionSortDataFile** (int problemSize, int V[], int GAP)
- void **generateBubbleSortDataFile** (int problemSize, int V[], int GAP)
- void **generateQuickSortDataFile** (int problemSize, int V[], int GAP)
- void **generateAllFiles** (int problemSize, int V[], int GAP)

2.2.1 Function Documentation

2.2.1.1 void **generateAllFiles** (int *problemSize*, int *V[]*, int *GAP*)

2.2.1.2 void **generateBubbleSortDataFile** (int *problemSize*, int *V[]*, int *GAP*)

2.2.1.3 void **generateBubbleSortTime** (ofstream & *file*, int *V[]*, int *n*)

2.2.1.4 void **generateInsertionSortDataFile** (int *problemSize*, int *V[]*, int *GAP*)

2.2.1.5 void **generateInsertionSortTime** (ofstream & *file*, int *V[]*, int *n*)

2.2.1.6 void **generateQuickSortDataFile** (int *problemSize*, int *V[]*, int *GAP*)

2.2.1.7 void **generateQuickSortTime** (ofstream & *file*, int *V[]*, int *n*)

2.2.1.8 void **generateSelectionSortDataFile** (int *problemSize*, int *V[]*, int *GAP*)

2.2.1.9 void **generateSelectionSortTime** (ofstream & *file*, int *V[]*, int *n*)

2.3 include/interaction.h File Reference

Functions

- int **getProblemSize** ()
Sets problem's size (random array's size)
- int **getMaximumInteger** ()
Sets the maximum integer to generate.
- int **getGap** ()

Sets the gap between sort iteration.

- char **chooseAlgorithm** ()

Shows the algorithms menu.

2.3.1 Function Documentation

2.3.1.1 char chooseAlgorithm ()

Shows the algorithms menu.

Returns

a *char* to be used in the main switch case

2.3.1.2 int getGap ()

Sets the gap between sort iteration.

Returns

incremental value between iterations

2.3.1.3 int getMaximumInteger ()

Sets the maximum integer to generate.

Returns

a positive integer that will represent the maximum integer to generate

2.3.1.4 int getProblemSize ()

Sets problem's size (random array's size)

Returns

a positive integer that will represent problem's size hereinafter

2.4 include/ordenacion.h File Reference

Functions

- void **insertionSort** (int V[], int num)

Performs an insertion sort algorithm on a vector of positive integers.

- void **selectionSort** (int V[], int num)

Performs a selection sort algorithm on a vector of positive integers.

- void **bubbleSort** (int V[], int num)

Performs a bubble sort algorithm on a vector of positive integers.

- void **quickSort** (int V[], int left, int right)

Performs a quick sort algorithm on a vector of positive integers.

2.4.1 Function Documentation

2.4.1.1 void bubbleSort (int V[], int num)

Performs a bubble sort algorithm on a vector of positive integers.

Parameters

<i>V</i>	a vector of positive integers
<i>num</i>	The array length

2.4.1.2 void insertionSort (int V[], int num)

Performs an insertion sort algorithm on a vector of positive integers.

Parameters

<i>V</i>	a vector of positive integers
<i>num</i>	The array length

2.4.1.3 void quickSort (int V[], int left, int right)

Performs a quick sort algorithm on a vector of positive integers.

Parameters

<i>V</i>	a vector of positive integers
<i>left</i>	the left index for the divide and conquer strategy (initially 0)
<i>right</i>	the right index for the divide and conquer strategy (initially the array length)

2.4.1.4 void selectionSort (int V[], int num)

Performs a selection sort algorithm on a vector of positive integers.

Parameters

<i>V</i>	a vector of positive integers
<i>num</i>	The array length

2.5 include/test.h File Reference

```
#include "../include/aux.h"
```

Functions

- void **generateArray** (int *V*[], int *n*, int *maxInt*)
Generates an array with n random integers between 0 and maxInt.
- void **makeFourCopies** (int *V*[], int *first*[], int *second*[], int *third*[], int *fourth*[], int *n*)

2.5.1 Function Documentation

2.5.1.1 void generateArray (int *V*[], int *n*, int *maxInt*)

Generates an array with n random integers between 0 and maxInt.

Parameters

<i>V</i>	the container array with a size of n
<i>n</i>	The array length
<i>maxInt</i>	maximum integer to generate array to

2.5.1.2 void makeFourCopies (int *V*[], int *first*[], int *second*[], int *third*[], int *fourth*[], int *n*)

2.6 src/aux.cpp File Reference

```
#include "../include/aux.h"
```

Functions

- void **createRandomArray** (int *V*[], int *n*, int *maxInteger*)
Creates a random integer array between 0 and MaxInteger.
- void **copyArray** (int *V*[], int *W*[], int *n*)
Copies an array contents into another array with the same length.
- bool **checkSameLength** (int *V*[], int *W*[])
- bool **checkArrayEquality** (int *V*[], int *W*[], int *n*)
Checks equality between two given arrays (both arrays have same content in same order)

- bool **checkIsOrdered** (int V[], int n)
Checks if an array is ordered.
- void **printArrayContents** (int V[], int n)
Prints array content on console.

2.6.1 Function Documentation

2.6.1.1 bool checkArrayEquality (int V[], int W[], int n)

Checks equality between two given arrays (both arrays have same content in same order)

Parameters

<i>V</i>	the original array
<i>W</i>	the other array
<i>n</i>	the array's length

2.6.1.2 bool checkIsOrdered (int V[], int n)

Checks if an array is ordered.

Parameters

<i>V</i>	the array
<i>n</i>	the array's length

2.6.1.3 bool checkSameLength (int V[], int W[])

2.6.1.4 void copyArray (int V[], int W[], int n)

Copies an array contents into another array with the same length.

Parameters

<i>V</i>	the original array
<i>W</i>	the cloned array
<i>n</i>	the array's length

2.6.1.5 void createRandomArray (int V[], int arrayLength, int maxInteger)

Creates a random integer array between 0 and MaxInteger.

Parameters

<i>V</i>	an array already set to a length of <i>arrayLength</i>
<i>arrayLength</i>	the array's length
<i>maxInteger</i>	the maximum integer to generate

2.6.1.6 void printArrayContents (int *V*[], int *n*)

Prints array content on console.

Parameters

<i>V</i>	the array
<i>n</i>	the array's length

2.7 src/generators.cpp File Reference

```
#include "../include/generators.h"
```

Functions

- void **generateInsertionSortTime** (ofstream &file, int *V*[], int *n*)
- void **generateSelectionSortTime** (ofstream &file, int *V*[], int *n*)
- void **generateBubbleSortTime** (ofstream &file, int *V*[], int *n*)
- void **generateQuickSortTime** (ofstream &file, int *V*[], int *n*)
- void **generateInsertionSortDataFile** (int *problemSize*, int *V*[], int *GAP*)
- void **generateSelectionSortDataFile** (int *problemSize*, int *V*[], int *GAP*)
- void **generateBubbleSortDataFile** (int *problemSize*, int *V*[], int *GAP*)
- void **generateQuickSortDataFile** (int *problemSize*, int *V*[], int *GAP*)
- void **generateAllFiles** (int *problemSize*, int *V*[], int *GAP*)

2.7.1 Function Documentation

2.7.1.1 void generateAllFiles (int *problemSize*, int *V*[], int *GAP*)2.7.1.2 void generateBubbleSortDataFile (int *problemSize*, int *V*[], int *GAP*)2.7.1.3 void generateBubbleSortTime (ofstream & *file*, int *V*[], int *n*)2.7.1.4 void generateInsertionSortDataFile (int *problemSize*, int *V*[], int *GAP*)2.7.1.5 void generateInsertionSortTime (ofstream & *file*, int *V*[], int *n*)2.7.1.6 void generateQuickSortDataFile (int *problemSize*, int *V*[], int *GAP*)

2.7.1.7 void generateQuickSortTime (ofstream & file, int V[], int n)

2.7.1.8 void generateSelectionSortDataFile (int problemSize, int V[], int GAP)

2.7.1.9 void generateSelectionSortTime (ofstream & file, int V[], int n)

2.8 src/interaction.cpp File Reference

```
#include "../include/interaction.h" #include <iostream>
```

Functions

- int **getProblemSize** ()
Sets problem's size (random array's size)
- int **getMaximumInteger** ()
Sets the maximum integer to generate.
- int **getGap** ()
Sets the gap between sort iteration.
- char **chooseAlgorithm** ()
Shows the algorithms menu.

2.8.1 Function Documentation

2.8.1.1 char chooseAlgorithm ()

Shows the algorithms menu.

Returns

a *char* to be used in the main switch case

2.8.1.2 int getGap ()

Sets the gap between sort iteration.

Returns

incremental value between iterations

2.8.1.3 int getMaximumInteger ()

Sets the maximum integer to generate.

Returns

a positive integer that will represent the maximum integer to generate

2.8.1.4 int getProblemSize ()

Sets problem's size (random array's size)

Returns

a positive integer that will represent problem's size hereinafter

2.9 src/main.cpp File Reference

```
#include "../include/test.h" #include "../include/interaction.-  
h" #include "../include/generators.h"
```

Functions

- **int main ()**

2.9.1 Function Documentation**2.9.1.1 int main ()****2.10 src/ordenacion.cpp File Reference**

```
#include "../include/ordenacion.h"
```

Functions

- void **insertionSort** (int V[], int num)
Performs an insertion sort algorithm on a vector of positive integers.
- void **selectionSort** (int V[], int num)
Performs a selection sort algorithm on a vector of positive integers.
- void **bubbleSort** (int V[], int num)
Performs a bubble sort algorithm on a vector of positive integers.
- void **quickSort** (int V[], int left, int right)
Performs a quick sort algorithm on a vector of positive integers.
- void **merge** (int *a, int *b, int low, int pivot, int high)

2.10.1 Function Documentation

2.10.1.1 void bubbleSort (int *V*[], int *num*)

Performs a bubble sort algorithm on a vector of positive integers.

Parameters

<i>V</i>	a vector of positive integers
<i>num</i>	The array length

2.10.1.2 void insertionSort (int *V*[], int *num*)

Performs an insertion sort algorithm on a vector of positive integers.

Parameters

<i>V</i>	a vector of positive integers
<i>num</i>	The array length

2.10.1.3 void merge (int * *a*, int * *b*, int *low*, int *pivot*, int *high*)

2.10.1.4 void quickSort (int *V*[], int *left*, int *right*)

Performs a quick sort algorithm on a vector of positive integers.

Parameters

<i>V</i>	a vector of positive integers
<i>left</i>	the left index for the divide and conquer strategy (initially 0)
<i>right</i>	the right index for the divide and conquer strategy (initially the array length)

2.10.1.5 void selectionSort (int *V*[], int *num*)

Performs a selection sort algorithm on a vector of positive integers.

Parameters

<i>V</i>	a vector of positive integers
<i>num</i>	The array length

2.11 src/test.cpp File Reference

```
#include "../include/test.h"
```

Functions

- void **makeFourCopies** (int V[], int first[], int second[], int third[], int fourth[], int n)
- void **generateArray** (int V[], int n, int maxInt)

Generates an array with n random integers between 0 and maxInt.

2.11.1 Function Documentation

2.11.1.1 void generateArray (int V[], int n, int maxInt)

Generates an array with n random integers between 0 and maxInt.

Parameters

<i>V</i>	the container array with a size of n
<i>n</i>	The array length
<i>maxInt</i>	maximum integer to generate array to

2.11.1.2 void makeFourCopies (int V[], int first[], int second[], int third[], int fourth[], int n)