PA04 - Sorting Algorithms

Generated by Doxygen 1.8.11

Contents

1 Class Index

1.1 Class List

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

2 File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

3 Class Documentation

3.1 BubbleSort Class Reference

Public Member Functions

- BubbleSort ()

  The default constructor of a BubbleSort object.

- BubbleSort (int ∗data, int size)

  The parameterized constructor of a BubbleSort object.
- ∼BubbleSort ()

  The destructor of a BubbleSort object.
- int ∗ DoSort ()

  This function runs the sorting algorithm.
- int ∗ DoSort (int ∗data, int size)

  Runs the sorting algorithm with new parameters.
- void Swap (int ∗firstVal, int ∗secondVal)

  This function swaps two values.
- void PrintFinal (int swapCount, int compCount)

  Outputs what happened in the sort.
- int GetSwaps ()

  Gets the number of swaps.
- int GetComps ()

  Gets the number of comparisons.

Private Attributes

- int **size**
- int ∗ **data**
- int **lastSwap**
- int **lastComp**

3.1.1 Constructor & Destructor Documentation

3.1.1.1 BubbleSort::BubbleSort ( )

The default constructor of a BubbleSort object.

This constructor initializes values of a BubbleSort object to default values

Algorithm None.

Parameters

in	*None.*	
out	*None.*	

Returns

None.

Note

None.

3.1.1.2 BubbleSort::BubbleSort (int ∗ *sentData,* int *sentSize*)

The parameterized constructor of a BubbleSort object.

This constructor initializes values of a BubbleSort object to the sent values

Algorithm None.

Parameters

`in`	*None.*	
`out`	*None.*	

Returns

None.

Note

None.

3.1.1.3 BubbleSort::∼BubbleSort ( )

The destructor of a BubbleSort object.

This safely removes a BubbleSort object from memory

Algorithm None.

Parameters

`in`	*None.*	
`out`	*None.*	

Returns

None.

Note

None.

3.1.2 Member Function Documentation

3.1.2.1 int ∗ BubbleSort::DoSort ( )

This function runs the sorting algorithm.

The function takes the data and sorts it using the bubble sorting algorithm

Algorithm Bubbles values up to the top so that an array is sorted in ascending order

Parameters

in	*None.*	
out	*The*	array pointed at by data is now sorted

Returns

None.

Note

None.

3.1.2.2 int ∗ BubbleSort::DoSort (int ∗ *sentData,* int *sentSize*)

Runs the sorting algorithm with new parameters.

This function uses the sent size and data and sorts it instead of the original values

Algorithm None.

Parameters

in	*sentData*	A pointer to the new data array
in	*sentSize*	The size of the sent array
out	*None.*	

Returns

Returns a pointer to the sorted int array.

Note

None.

3.1.2.3 int BubbleSort::GetComps ( )

Gets the number of comparisons.

Gets the number of comparisons from the last run of the sort

Algorithm None.

Parameters

in	*None.*	
out	*None.*	

Returns

Returns the integer value of lastComp.

Note

None.

3.1.2.4 int BubbleSort::GetSwaps ( )

Gets the number of swaps.

Gets the number of swaps from the last run of the sort

Algorithm None.

Parameters

in	*None.*	
out	*None.*	

Returns

Returns the integer value of lastSwap.

Note

None.

3.1.2.5 void BubbleSort::PrintFinal (int *swapCount,* int *compCount*)

Outputs what happened in the sort.

Prints the data in the array as well as the number of swaps and comparisons

Algorithm None.

Parameters

`in`	*swapCount*	An integer representing the number of swaps performed
`in`	*compCount*	An integer representing the number of comparisons performed
`out`	*None.*	

Returns

None.

Note

None.

3.1.2.6 void BubbleSort::Swap (int $*$ *firstVal,* int $*$ *secondVal*)

This function swaps two values.

This function swaps the two values pointed at by the pointers

Algorithm None.

Parameters

`in`	*firstVal*	Pointer to the first integer value
`in`	*secondVal*	Pointer to the second integer value
`out`	*The*	values are now swapped.

Returns

None.

Note

None.

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

- PA04/BubbleSort.h
- PA04/BubbleSort.cpp

3.2 CountingSort Class Reference

Public Member Functions

- CountingSort ()

  The default constructor of a CountingSort object.
- CountingSort (int ∗data, int size, int max)

  The parameterized constructor of a CountingSort object.
- ∼CountingSort ()

  The destructor of a CountingSort object.
- int ∗ DoSort ()

  This function runs the sorting algorithm.
- int ∗ DoSort (int ∗data, int size, int max)

  This function runs the sorting algorithm.
- void PrintFinal (int swapCount, int compCount)

  Outputs what happened in the sort.
- int GetSwaps ()

  Gets the number of swaps.
- int GetComps ()

  Gets the number of comparisons.

Private Attributes

- int **size**
- int ∗ **data**
- int **max**
- int **lastSwap**
- int **lastComp**

3.2.1 Constructor & Destructor Documentation

3.2.1.1 CountingSort::CountingSort ( )

The default constructor of a CountingSort object.

This constructor initializes values of a CountingSort object to default values

Algorithm None.

Parameters

in	*None.*	
out	*None.*	

Returns

None.

Note

None.

3.2.1.2 CountingSort::CountingSort (int ∗ *sentData*, int *sentSize*, int *sentMax*)

The parameterized constructor of a CountingSort object.

This constructor initializes values of a CountingSort object to the sent values

Algorithm None.

Parameters

in	*None.*	
out	*None.*	

Returns

None.

Note

None.

3.2.1.3 CountingSort::∼CountingSort ( )

The destructor of a CountingSort object.

This safely removes a CountingSort object from memory

Algorithm None.

Parameters

in	*None.*	
out	*None.*	

Returns

None.

Note

None.

3.2.2 Member Function Documentation

3.2.2.1 int ∗ CountingSort::DoSort ( )

This function runs the sorting algorithm.

The function takes the data and sorts it with the counting sort algorithm

Algorithm Counts the frequency of each value in the data, then sorts it by the count

Parameters

`in`	*None.*	
`out`	*The*	array pointed at by data is now sorted

Returns

None.

Note

None.

3.2.2.2 int ∗ CountingSort::DoSort (int ∗ *sentData,* int *sentSize,* int *sentMax*)

This function runs the sorting algorithm.

The function takes the data and sorts it with the counting sort algorithm with the sent values

Algorithm Counts the frequency of each value in the data, then sorts it by the count

Parameters

`in`	*sentData*	Pointer to the integer array to be sorted
`in`	*sentSize*	The size of the array
`in`	*sentMax*	The maximum value in the data, constantly 1M for this project
`out`	*The*	array pointed at by data is now sorted

Returns

Returns a pointer to the sorted integer array

Note

None.

3.2.2.3 int CountingSort::GetComps ( )

Gets the number of comparisons.

Gets the number of comparisons from the last run of the sort

Algorithm None.

Parameters

`in`	*None.*	
`out`	*None.*	

Returns

Returns the integer value of lastComp.

Note

None.

3.2.2.4 int CountingSort::GetSwaps ( )

Gets the number of swaps.

Gets the number of swaps from the last run of the sort

Algorithm None.

Parameters

`in`	*None.*	
`out`	*None.*	

Returns

Returns the integer value of lastSwap.

Note

None.

3.2.2.5 void CountingSort::PrintFinal (int *swapCount,* int *compCount*)

Outputs what happened in the sort.

Prints the data in the array as well as the number of swaps and comparisons

Algorithm None.

Parameters

in	*swapCount*	An integer representing the number of swaps performed
in	*compCount*	An integer representing the number of comparisons performed
out	*None.*	

Returns

None.

Note

None.

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

- PA04/CountingSort.h
- PA04/CountingSort.cpp

3.3 MergeSort Class Reference

Public Member Functions

- MergeSort ()

  The default constructor of a MergeSort object.
- MergeSort (int ∗data, int size)

  The parameterized constructor of a MergeSort object.
- ∼MergeSort ()

  The destructor of a MergeSort object.
- void DoSort (int first, int last)

  This function runs the sorting algorithm.
- int ∗ DoSort (int first, int mid, int last)

  This function runs the sorting algorithm.
- void **DoSort** (int ∗data, int size)
- void **Swap** (int ∗firstVal, int ∗secondVal)
- void PrintFinal ()

  Outputs what happened in the sort.
- void **ResetCounts** ()
- int GetSwaps ()

  Gets the number of swaps.
- int GetComps ()

  Gets the number of comparisons.

Private Attributes

- int **size**
- int ∗ **data**
- int **lastComparisonCount**
- int **lastSwapCount**

3.3.1 Constructor & Destructor Documentation

3.3.1.1 MergeSort::MergeSort ( )

The default constructor of a MergeSort object.

This constructor initializes values of a MergeSort object to default values

Algorithm None.

Parameters

`in`	*None.*	
`out`	*None.*	

Returns

None.

Note

None.

3.3.1.2 MergeSort::MergeSort (int ∗ *sentData,* int *sentSize*)

The parameterized constructor of a MergeSort object.

This constructor initializes values of a MergeSort object to the sent values

Algorithm None.

Parameters

`in`	*None.*	
`out`	*None.*	

Returns

None.

Note

None.

3.3.1.3 MergeSort::∼MergeSort ( )

The destructor of a MergeSort object.

This safely removes a MergeSort object from memory

Algorithm None.

Parameters

in	*None.*	
out	*None.*	

Returns

None.

Note

None.

3.3.2 Member Function Documentation

3.3.2.1 void MergeSort::DoSort (int *first,* int *last*)

This function runs the sorting algorithm.

The function takes the data and sorts it by splitting the array into smaller arrays and sorting those then merging it all back together

Algorithm Splits the main array into smaller arrays and sorts them then merges them together into one sorted array

Parameters

in	*first*	The index of the first value of the array in scope
in	*last*	The index of the last value of the array in scope
out	*The*	array pointed at by data is now sorted

Returns

None.

Note

None.

3.3.2.2 int * MergeSort::DoSort (int *first*, int *mid*, int *last*)

This function runs the sorting algorithm.

The function takes the data and sorts it by splitting the array into smaller arrays and sorting those then merging it all back together

Algorithm Splits the main array into smaller arrays and sorts them then merges them together into one sorted array

Parameters

`in`	*first*	The index of the first value of the array in scope
`in`	*last*	The index of the last value of the array in scope
`in`	*mid*	The index of the mid point of the array in scope
`out`	*The*	array pointed at by data is now sorted

Returns

None.

Note

None.

3.3.2.3 int MergeSort::GetComps ( )

Gets the number of comparisons.

Gets the number of comparisons from the last run of the sort

Algorithm None.

Parameters

`in`	*None.*	
`out`	*None.*	

Returns

Returns the integer value of lastComp.

Note

None.

3.3.2.4 int MergeSort::GetSwaps ( )

Gets the number of swaps.

Gets the number of swaps from the last run of the sort

Algorithm None.

Parameters

`in`	*None.*	
`out`	*None.*	

Returns

Returns the integer value of lastSwap.

Note

None.

3.3.2.5 void MergeSort::PrintFinal ( )

Outputs what happened in the sort.

Prints the data in the array as well as the number of swaps and comparisons

Algorithm None.

Parameters

`in`	*swapCount*	An integer representing the number of swaps performed
`in`	*compCount*	An integer representing the number of comparisons performed
`out`	*None.*	

Returns

None.

Note

None.

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

- PA04/MergeSort.h
- PA04/MergeSort.cpp

4 File Documentation

4.1 PA04/BubbleSort.cpp File Reference

This is the implementation of the BubbleSort class.

```
#include "BubbleSort.h"
```

4.1.1 Detailed Description

This is the implementation of the BubbleSort class.

Author

Bryce Monaco

This file contains the implementation of the BubbleSort class

Version

1.0

Note

None.

4.2 PA04/BubbleSort.h File Reference

This is the header of the BubbleSort class.

```
#include <iostream>
#include <ctime>
```

Classes

- class BubbleSort

4.2.1 Detailed Description

This is the header of the BubbleSort class.

Author

Bryce Monaco

This file contains the header of the BubbleSort class

Version

1.0

Note

None.

4.3 PA04/CountingSort.cpp File Reference

This is the implementation of the CountingSort class.

```
#include "CountingSort.h"
```

4.3.1 Detailed Description

This is the implementation of the CountingSort class.

Author

Bryce Monaco

This file contains the implementation of the CountingSort class

Version

1.0

Note

None.

4.4 PA04/CountingSort.h File Reference

This is the header of the CountingSort class.

```
#include <iostream>
#include <ctime>
```

Classes

- class CountingSort

4.4.1 Detailed Description

This is the header of the CountingSort class.

Author

Bryce Monaco

This file contains the header of the CountingSort class

Version

1.0

Note

None.

4.5 PA04/MergeSort.cpp File Reference

This is the implementation of the MergeSort class.

```
#include "MergeSort.h"
```

4.5.1 Detailed Description

This is the implementation of the MergeSort class.

Author

Bryce Monaco

This file contains the implementation of the MergeSort class

Version

1.0

Note

None.

4.6 PA04/MergeSort.h File Reference

This is the header of the MergeSort class.

```
#include <iostream>
#include <ctime>
```

Classes

- class MergeSort

4.6.1 Detailed Description

This is the header of the MergeSort class.

Author

Bryce Monaco

This file contains the header of the MergeSort class

Version

1.0

Note

None.

4.7 PA04/PA04.cpp File Reference

This is the main driver file for Programming Assignment 04.

```
#include <iostream>
#include <fstream>
#include <vector>
#include <string>
#include "BubbleSort.h"
#include "MergeSort.h"
#include "CountingSort.h"
#include <cstdlib>
#include <time.h>
```

Functions

- void GenerateValues (int ∗valuesStart, int amount)

  Generates a certain amount of random values and stores them in a file.
- void ReadValuesFromFile (int ∗valuesStart, int amount, int fileNumber)

  This function reads the values in from a file.
- int **main** ()

4.7.1 Detailed Description

This is the main driver file for Programming Assignment 04.

Author

Bryce Monaco

This file runs through the data with each sorting algorithm and finds average times and counts for each to compare.

Version

1.0

Note

None.

4.7.2 Function Documentation

4.7.2.1 void GenerateValues (int ∗ *valuesStart*, int *amount*)

Generates a certain amount of random values and stores them in a file.

This function generates a certain amount of random values and them dumps them into a file for easy reference later

Algorithm Generates random values into an array, then traverses the array and outputs the values to a file.

Parameters

`in`	*valuesStart*	A pointer to an integer array. The argument is never used in the current version and can just be called with NULL
`in`	*amount*	The amount of values to generate.
`out`	*Creates*	ten files each populated with a certain amount of random values.

Returns

None.

Note

The pointer valuesStart is not used in the current implementation, so the argument can just be sent as NULL

4.7.2.2 void ReadValuesFromFile (int ∗ *valuesStart*, int *amount*, int *fileNumber*)

This function reads the values in from a file.

This function reads the values in from a file created by GenerateValues() and stores them in an array

Algorithm None.

Parameters

`in`	*valuesStart*	The pointer to the values array in main()
`in`	*amount*	The number of values to read in
`in`	*fileNumber*	The number corresponding to the file to be opened
`out`	*the*	valuesStart pointer now holds the numbers inside the file.

Returns

None.

Note

None.

Index