PA02 - Recursion

Generated by Doxygen 1.8.6

Mon Oct 3 2016 18:28:39

Contents

1 File Index

1.1 File List

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

main.cpp
File containing source code for kthSmall function 1

2 File Documentation

2.1 main.cpp File Reference

File containing source code for kthSmall function.

```
#include <iostream>
#include <cstddef>
#include <fstream>
```

Functions

- int kSmall (int k, int first, int last, int *data)
  Function finds the kth smallest value in a set of numbers.
- void swap (int a, int b, int *data)
  Function swaps two values.
- void logToFile (int k, int pivot, int pivotPos, int firstPos, int lastPos, int *data)
  Function logs the recursive steps of kSmall.
- int **main** (int argc, char *argv[])

2.1.1 Detailed Description

File containing source code for kthSmall function.

Author

Alex Kastanek

Contains kSmall function, swap function, and logToFile function

Version

1.00 C.S. Student (14 September 2016) Initial development and testing of kSmall

Note

File's pseudocode created by Shehryar Khattak for CS302 Spring 2016 class.
Program must be run with a command line argument to determine filename

2.1.2 Function Documentation

2.1.2.1 int kSmall (int *k,* int *first,* int *last,* int * *data*)

Function finds the kth smallest value in a set of numbers.

Function finds the kth smallest value in a set of numbers by recursively dividing the set into 2 using a pivot and only evaluating one half of the set depending on what the relationship between k and pivot is

Precondition

int pivot is intialized to the first value in the set
int pivotIndex is intialized to address of pivot
int lowCount is initialized to address of value after first value in the set
int highCount is initialized to address of last value in the set

Postcondition

pivot and its index change depending on what values in the set are less than it or greater than it
lowCount is incremented and highCount is decremented until their positions are the same

Algorithm

compares the values at lowCount and pivot, swaps and decrements highCount if lowCount is greater than or equal to pivot, increments lowCount if not OR swaps and increments lowCount if highCount is less than pivot, decremets highCount if not

Algorithm

swap pivot's position with the next value until all values to the left of pivot are less than it and all value to the right of pivot are greater than it

Algorithm

compares k to pivotIndex - first + 1 and returns pivot if they are equal, returns a call to the function with the same parameters if k is k is less than the expression, or returns a call to the function with k being subtracted by the expression if k is greater than the expression

Parameters

in	*k*	holds the magnitude of the value function looks for, function could search for 4th smallest value or 8th smallest depending user input
in	*first*	holds position of first value in the array
in	*last*	holds position of last value in the array
in	*data*	array of integers that holds the set of values the function searches through

out	*pivot*	pivot is the only output of this function as it will contain the kth smallest value

Exceptions

None	

Returns

pivot

Note

: None

2.1.2.2 void logToFile (int *k,* int *pivot,* int *pivotPos,* int *firstPos,* int *lastPos,* int * *data*)

Function logs the recursive steps of kSmall.

Function logs the partitions in kSmall as well as changes to k and pivot

Precondition

None

Postcondition

None

None

Parameters

in	*k*	k in kSmall
in	*pivot*	pivot in kSmall
in	*pivotPos*	holds position of pivot
in	*firstPos*	holds position of first value in section of array
in	*lastPos*	holds position of last value in section of array
in	*data*	array that holds all integers in the set
out	*None*	

Exceptions

None	

Returns

None

Note

: None

2.1.2.3 void swap (int *a,* int *b,* int * *data*)

Function swaps two values.

Function swaps two values by storing one in a temp location

Precondition

int temp is intialized to b

Postcondition

temp gets stored in a

None

Parameters

in	*a*	gets b, which was stored in temp
in	*b*	gets a
in	*data*	contains a and b
out	*data*	a and b get swapped in this array

Exceptions

None	

Returns

None

Note

: None

Index