CS2263 System Software Lab2

Graham Hill 3587614

Vasili Osipau 3552807

Github Address: https://github.com/grahamhillUNB/CS2263_Summer2019_L2

ArraySort.c:

```
/*********************************
 *
 * ArraySort.c
 *
 * Created by Jean-Philippe Legault
 * Modified by Graham Hill and Vasili Osipau
 *
 * Your task is to implement in place sorting using the two available functions
 * swapAdjacent, and compareAdjacent.
 *
 * Some bug might have been introduced... you will have to find out if there are any!
 * if so, you will have to correct it
 *
 *********************************/
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>

void printArray(int *array, int size)
{
	for(int i=0; i<size; i++)
	{
```

```
            if(i != 0)
            {
                printf(", ");
            }
            printf("%d", array[i]);
        }
        printf("\n");
}

void swapAdjacent(int *a, int index)
{
        int temp = *(a + index);
        *(a + index) = *(a + index + 1);
        *(a + index + 1) = temp;
}

int compareAdjacent(int *a, int index)
{
        return *(a + index) - *(a + index + 1);
}

/**
  * Author Graham & Vasili
  * by using the two functions swapAdjacent and compareAdjacent
  */
void inPlaceSort(int *array, int size)
{
        for(int i = 0; i < size-1; i++){
            for(int j = 0; j < size - i -1; j++){
```

```
                if(compareAdjacent(array, j) > 0){
                        swapAdjacent(array, j);
                }
            }
        }
}

int main(void)
{
        int array_size = 0;
        printf("Enter the array size (>0) and the numbers to fill the array with: ");
        if(!scanf("%d", &array_size))
        {
                printf("ERROR. Must enter an integer.\n");
                return EXIT_FAILURE;
        }
        else if(array_size < 1)
        {
                printf("ERROR. array size must be at least 1.\n");
                return EXIT_FAILURE;
        }

        int a[array_size];

        /**********************
          *    Authored by Graham and Vasili
          * it should parse user input with scanf to fill the array with values
          **********************/
```

```
	int temp = 0;
	for(int count = 0; count < array_size; count++){
		if(scanf("%d", &temp) == 1){
			a[count] = temp;
		}
		else{
			printf("Not a correct integer!");
			return EXIT_FAILURE;
		}
	}
	printArray(a, array_size);
	inPlaceSort(a, array_size);
	printArray(a, array_size);

}
```

Makefile:

```
##########################################
# Created by Jean-Philippe Legault
#
# This is a comment, a comment always start with `#`
# Indentation is primordial in a Makefile.
# the steps for a target are always indented
#
##########################################

# compile with gcc, change this to clang if you prefer
COMPILER = gcc
```

```
# The C flags to pass to gcc
C_FLAGS = -Wall -Wextra

# prepend the command with '@' so that Make does not print the command before running it
help:
   @printf "available command:\n"
   @printf "   make help                    (this command)\n"
   @printf "   make ArraySort               (to build your C program)\n"
   @printf "   make test                    (to run every test case)\n"

# link our .o files to make an executable
ArraySort: ArraySort.o
   $(COMPILER) $(C_FLAGS) -o ArraySort ArraySort.o

# compile the `Stack.o` file
ArraySort.o: ArraySort.c
   $(COMPILER) $(C_FLAGS) -c ArraySort.c

#####################################################################
# Test Cases
test: test1 test2 test3

# run our executable by passing in the text file via stdin with `<` and passing stdout to a file with `>`
# then use a scrit to verify that the result are the same one as the one expected
test1: ArraySort Data/test1.input Data/test1.expected
   ./ArraySort < Data/test1.input > test1.result
   ./TestPassed.sh test1.result Data/test1.expected

test2: ArraySort Data/test2.input Data/test2.expected
```

./ArraySort < Data/test2.input > test2.result

./TestPassed.sh test2.result Data/test2.expected

test3: ArraySort Data/test3.input Data/test3.expected

./ArraySort < Data/test3.input > test3.result

./TestPassed.sh test3.result Data/test3.expected

GDB Debugging:

```
user@LAPTOP-L25MBUSV:/mnt/c/users/user/desktop/Lab 2/CS2263_Summer2019_L2$ gdb ArraySort
GNU gdb (Ubuntu 8.1-0ubuntu3) 8.1.0.20180409-git
Copyright (C) 2018 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ArraySort...done.
(gdb) b swapAdjacent
Breakpoint 1 at 0x826: file ArraySort.c, line 33.
(gdb) r
Starting program: /mnt/c/users/user/desktop/Lab 2/CS2263_Summer2019_L2/ArraySort
Enter the array size (>0) and the numbers to fill the array with: 5
Please input ints for array:
3
5
1
4
2
=== Array before Sorting = 3, 5, 1, 4, 2

Breakpoint 1, swapAdjacent (a=0x7fffffffee240, index=1) at ArraySort.c:33
33          int temp = *(a + index);
(gdb) bt
#0  swapAdjacent (a=0x7fffffffee240, index=1) at ArraySort.c:33
#1  0x0000000008000918 in inPlaceSort (array=0x7fffffffee240, size=5) at ArraySort.c:52
#2  0x0000000008000ace in main () at ArraySort.c:95
(gdb)
```

Running Make tests: