Barak Barclay

Leslie Tekamp

ECE-1021

4 November 2015

HWK7

//

// HWK7(1).cpp

//

// By: Barak Barclay

// Date: 4 Nov 2015

//

// Problem Statement: Write a program with two functions. The main function will just define and initialize the variables and call the user functions.

// All the work will be done by the user functions. The size of the array is 100.

// Function #1: this function will be passed a float array and its size and return nothing.The function will fill the array with random floats between 0 and 1 (exclusive).Use the formula below :

// x = (float)rand() / (float)(RAND\_MAX + 1);

// Function #2: Print the array to the screen.Print the numbers using %8.4f format no blank spaces between.

//

// %%%% Algorthim %%%%

//

// preprocessor directives

// start main

// init variables

// call uf1

// call uf2

// end main

//

// 1111 uf1 1111

//

// fill array with flosta b/w 0 and 1: x = (float)rand() / (float)(RAND\_MAX + 1);

//

// 2222 uf2 2222

//

// print arrat using %8.4f format with no blank space in b/w

//

#include <stdio.h>

#include <stdlib.h> //includes rand function

#define S 100

void uf1(float A[], size\_t i);

void uf2(float A[], size\_t i);

void main(void)

{

float A[S];

size\_t i;

for (i = 0; i < S; i++)

{

A[i] = 0;

}

uf1(A, i);

uf2(A, i);

}

void uf1(float A[], size\_t i)

{

float x;

for (i = 0; i < S; i++)

{

x = (float)rand() / (float)(RAND\_MAX + 1);

A[i] = x;

}

}

void uf2(float A[], size\_t i)

{

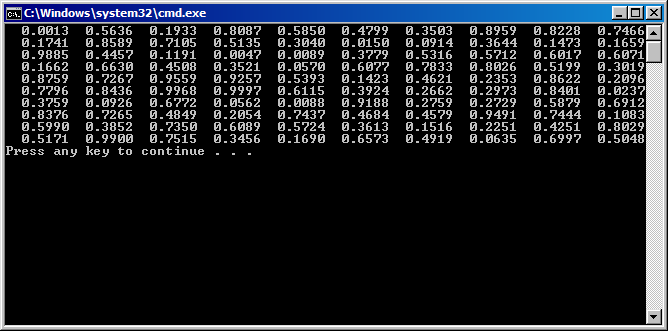
for (i = 0; i < S; i++)

{

printf("%8.4f", A[i]);

}

}



//

// HWK7(2).cpp

//

// By: Barak Barclay

// Date: 4 Nov 2015

//

// Problem Statement: Write an additional function to add to the above program (Program #1). This function

// will do a Histogram operation on the data in the array and print the results. The function will be passed

// the float array and its size and return nothing. Prompt the user to enter in the number of cells for the

// Histogram, this should be < the size of the array, do the error check. Then count the number of array data

// values in that histogram range and print the results to the screen, repeat until entire histogram range is

// covered. Range of data values is 0.0 to 1.0 exclusive.

// Example: user types in 5, histogram cell width is 1.0 / 5 = 0.2.First cell goes from 0.0 to <0.2, next cell

// goes 0.2 to <0.4, …….., last cell goes 0.8 to <1.0, function counts all the data values in the range of each cell.

//

// %%%% Algorthim %%%%

//

// preprocessor directives

// start main

// init variables

// call uf1

// call uf2

// end main

//

// 1111 uf1 1111

//

// fill array with flosta b/w 0 and 1: x = (float)rand() / (float)(RAND\_MAX + 1);

//

// 2222 uf2 2222

//

// prompt user to enter value and error check

// for loops to print cell# and amount of data values within each cell

//

#include <stdio.h>

#include <stdlib.h> //includes rand function

#define S 100

void uf1(float A[], size\_t i);

void uf2(float A[], size\_t i);

void main(void)

{

float A[S];

size\_t i;

for (i = 0; i < S; i++)

{

A[i] = 0;

}

uf1(A, i);

uf2(A, i);

}

void uf1(float A[], size\_t i)

{

float x;

for (i = 0; i < S; i++)

{

x = (float)rand() / (float)(RAND\_MAX + 1);

A[i] = x;

}

}

void uf2(float A[], size\_t i)

{

float f = 0.0;

int cell = 1;

int c = 0;

size\_t j;

printf("Enter in the number of cells for the Histogram (less than 100):");

scanf\_s("%f", &f);

while (f <= 0.0 || f >= 100)

{

printf("\tERROR: Enter a number between 0 and 100(exclusive):");

printf("Enter in the number of cells for the Histogram (less than 100):");

scanf\_s("%f", &f);

}

for (j = 0; j < f; j++)

{

for (i = 0; i < S; i++)

{

if (A[i] <= 1.0 / f \* cell && A[i] >= 1.0 / f \* (cell - 1))

{

c++;

}

}

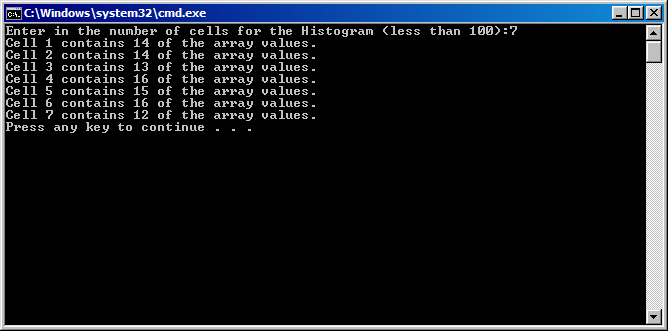
printf ("Cell %d contains %d of the array values.\n", cell, c);

cell++;

c = 0;

}

}



//

// HWK7(3).cpp

//

// By: Barak Barclay

// Date: 4 Nov 2015

//

// Problem Statement: Write a program with two functions. The main function will just define and initialize

// the variables and call the user functions.All the work will be done by the user functions.

// In the main function, define an integer array of 55 elements and initialize the array to

// zero.

// Function #1: this function will be passed the integer array and its size and return nothing.

// .This function will fill the array with random numbers between 1 and 10. Print the array

// to the screen with 5 blank spaces between numbers.

// Function #2: this function will be passed the integer array and its size and return nothing.

// This function will prompt the user to enter in a number to search for between 1 and 10,

// error check to make sure the number is in range.Search the array and print the element

// number for each match to the screen.

//

// %%%% Algorthim %%%%

//

// preprocessor directives

// start main

// init variables

// call uf1

// call uf2

// end main

//

// 1111 uf1 1111

//

// same as last problem including printing and changing to int

//

// 2222 uf2 2222

//

// change up to search for a number (outer for loop doesn’t need to exist

//

#include <stdio.h>

#include <stdlib.h> //includes rand function

#define S 55

void uf1(int A[], size\_t i);

void uf2(int A[], size\_t i);

void main(void)

{

int A[S];

size\_t i;

for (i = 0; i < S; i++)

{

A[i] = 0;

}

uf1(A, i);

uf2(A, i);

}

void uf1(int A[], size\_t i)

{

int x;

for (i = 0; i < S; i++)

{

x = rand() % 9 + 1;

A[i] = x;

}

for (i = 0; i < S; i++)

{

printf("%d ", A[i]);

}

}

void uf2(int A[], size\_t i)

{

int f = 0;

printf("\nEnter to search for between 1 and 10:");

scanf\_s("%d", &f);

while (f < 1 || f > 10)

{

printf("\tERROR: Enter a number between 1 and 10:");

scanf\_s("%d", &f);

}

for (i = 0; i < S; i++)

{

if (A[i] <= f)

{

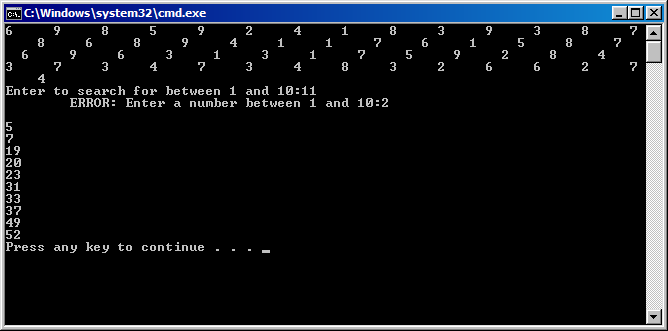
printf("\n%d", i);

}

}

printf("\n");

}



//

// HWK7(4).cpp

//

// By: Barak Barclay

// Date: 4 Nov 2015

//

// Problem Statement: Write a program with three functions. The main function will just define and initialize

// the variables and call the user functions.All the work will be done by the user functions.

// In the main function, define an integer array of 20 elements and initialize the array to zero.

// Function #1: this function will be passed the integer array and its size and return nothing.

// This function will fill the array with random numbers between 1 and 10.

// Function #2: this function will be passed the integer array and its size and return nothing.

// This function will print the array elements to the screen with one tab space between(\t).

// Function #3: this function will be passed the integer array and its size and return nothing.

// This function will do a bubble sort, with the largest data value in element zero.

// print the sorted array – use function 2 again.

//

// %%%% Algorthim %%%%

//

// preprocessor directives

// start main

// init variables

// call uf1

// call uf2

// call uf3

// call uf2

// end main

//

// 1111 uf1 1111

//

// fill array like last problem

//

// 2222 uf2 2222

//

// print array like last problem

//

// 3333 uf3 3333

//

// fill array by resetting values and holding values putting them in order with nested for loops (bubble sort)

//

#include <stdio.h>

#include <stdlib.h> //includes rand function

#define S 20

void uf1(int A[], size\_t i);

void uf2(int A[], size\_t i);

void uf3(int A[], size\_t size);

void main(void)

{

int A[S];

size\_t i;

for (i = 0; i < S; i++)

{

A[i] = 0;

}

uf1(A, i);

uf2(A, i);

puts("\n");

uf3(A, i);

uf2(A, i);

}

void uf1(int A[], size\_t i)

{

int x;

for (i = 0; i < S; i++)

{

x = rand() % 9 + 1;

A[i] = x;

}

}

void uf2(int A[], size\_t i)

{

for (i = 0; i < S; i++)

{

printf("%d ", A[i]);

}

}

void uf3(int A[], size\_t size)

{

int count = 1;

int pass;

int hold;

int i;

for (pass = 1; (unsigned)pass < size; pass++)

{

for (i = 0; i < (size - count); i++)

{

if (A[i] < A[i + 1])

{

hold = A[i];

A[i] = A[i + 1];

A[i + 1] = hold;

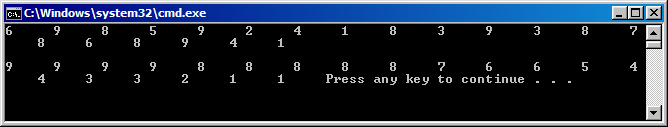
}

}

count += 1;

}

}



//

// HWK7(5).cpp

//

// By: Barak Barclay

// Date: 4 Nov 2015

//

// Problem Statement: Write a program with two functions. The main function will just define and initialize the

// variables and call the user functions. All the work will be done by the user functions. In the main function,

// define a char array of 256 elements and initialize the array to blanks.

// Function#1: This function will be passed the char array and its size and return nothing.Fill each element with

// its element number – i.e.mycharArray[i] = i.You are storing an integer number in a char, this works for char

// variables.

// Function#2: This function will be passed the char array and its size and return nothing.Print out each element

// from the array with 2 blank spaces between characters.Shows what symbol you get from the ASCII character set.

//

// %%%% Algorthim %%%%

//

// preprocessor directives

// start main

// init variables

// call uf1

// call uf2

// end main

//

// 1111 uf1 1111

//

// fill array like last problem filling with element number instead of random number

//

// 2222 uf2 2222

//

// print array like last problem chagning %d to %c

//

#include <stdio.h>

#include <stdlib.h> //includes rand function

#define S 256

void uf1(char A[], size\_t i);

void uf2(char A[], size\_t i);

void main(void)

{

char A[S];

size\_t i;

for (i = 0; i < S; i++)

{

A[i] = ' ';

}

uf1(A, i);

uf2(A, i);

puts("");

}

void uf1(char A[], size\_t i)

{

for (i = 0; i < S; i++)

{

A[i] = i;

}

}

void uf2(char A[], size\_t i)

{

for (i = 0; i < S; i++)

{

printf("%c ", A[i]);

}

}



//

// HWK7(6).cpp

//

// By: Barak Barclay

// Date: 4 Nov 2015

//

// Problem Statement: Write a binary search program with 4 functions. Create an integer array of 101 elements

// (0 to 100). The main function will use a sentinel loop; enter in a data value between 0 and 100 to search for

// or a negative number to quit, error check the entered number.

// Function #1: Fill the array.Set each array element value to its element number.So array will be sorted as filled.

// Pass the function the array and its size and return nothing.

// Function #2: Print out the array values with 2 blank spaces between numbers.Pass the function the array and its

// size and return nothing.

// Function #3: Do a binary search.Pass the function the array, its size and the value to search for - return the

// element number(integer) where the match was found.For each iteration of the binary search, print out the low,

// middle and high element numbers.Function #4: Pass the value you searched for and the returned element number

// where the match was found, return nothing.This function prints out the value you searched for and the element it

// was found in.

//

// %%%% Algorthim %%%%

//

// copy and paste last problem changing array to int

//

// preprocessor directives

// start main

// init variables

// call uf1

// call uf2

// call uf3

// end main

//

// 1111 uf1 1111

//

// fill array like last problem

//

// 2222 uf2 2222

//

// print array like last problem changing %c to %d

//

// 3333 uf3 3333

//

// init variables

// while loop

// set middle

// print variables

// if statements to either return middle or reset high and low values

// end while

//

// 44444 uf4 4444

//

// print values

//

#include <stdio.h>

#include <stdlib.h> //includes rand function

#define S 101

void uf1(int A[], size\_t i);

void uf2(int A[], size\_t i);

int uf3(int A[], size\_t i, int n);

void uf4(int n, int n2);

void main(void)

{

int A[S];

size\_t i;

int n = 0;

int n2 = 0;

for (i = 0; i < S; i++)

{

A[i] = 0;

}

uf1(A, i);

uf2(A, i);

puts("\nEnter in a data value between 0 and 100 to search for (negative # to quit):");

scanf\_s("%d", &n);

while (n >= 0)

{

n2 = uf3(A, i, n);

uf4(n, n2);

puts("\nEnter in a data value between 0 and 100 to search for (negative # to quit):");

scanf\_s("%d", &n);

}

puts("");

}

void uf1(int A[], size\_t i)

{

for (i = 0; i < S; i++)

{

A[i] = i;

}

}

void uf2(int A[], size\_t i)

{

for (i = 0; i < S; i++)

{

printf("%d ", A[i]);

}

}

int uf3(int A[], size\_t i, int n)

{

int middle = -1;

int low = 0;

int high = (i - 1);

while (low <= high)

{

middle = (low + high) / 2;

printf("\n Low = %d, middle = %d high = %d \n", low, middle, high);

if (n == A[middle])

{

return middle;

}

if (n < A[middle])

{

high = middle - 1;

}

else

{

low = middle + 1;

}

}

return -1;

}

void uf4(int n, int n2)

{

printf("The number %d, you searched for is found in element %d\n", n, n2);

}

