Assignment-1

Date: 05/02/24

Q1. Create a menu driven program to work with Array ADT.

Q2. Create a program to check whether a given matrix is sparse or not.

Q3. Create a program to transform a given sparse matrix into its corresponding Three row triplet representation.

Q1. Create a menu driven program to work with Array ADT.

#include<stdio.h>

#include <stdlib.h>

// global array

int arr[10] = {1,2,3,4,5};

int length=5; // 0 for custom array

// function to take values from user

void makeArray(){

printf("Enter size of array (1-10): ");

scanf("%d", &length);

for(int i=0; i<length; i++){

printf("\nEnter value at index %d: ",i);

scanf("%d", &arr[i]);

}

}

// function to traverse array

void traverse(){

for(int i=0; i<length; i++){

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

}

printf("\n");

}

// function to insert into array

int insert(int loc, int val){

if(loc < 0 || loc > length){

printf("Invalid index\n");

return -1;

}

for(int i=length+1; i>loc; i--){

arr[i] = arr[i-1];

}

arr[loc] = val;

length++;

return 0;

}

// function to Delete from array (index based delete)

int deleteItem(int loc){

if(loc <0 || loc >=length){

printf("Invalid index\n");

return -1;

}

int temp = arr[loc];

for(int i=loc; i<length-1; i++){

arr[i] = arr[i+1];

}

length--;

return temp;

}

// function to perform linear search (return first occurance)

int lsearch(int val){

for(int i=0; i<length; i++){

if(val == arr[i]) return i;

}

return -1;

}

// function to find count of occurances using linear search

int count(int val){

int count = 0;

for(int i=0; i<length; i++){

if(val == arr[i]) count++;

}

return count;

}

// function to search an element using binary search

// NOTE: array must be sorted

int binarySearch(int val){

int low = 0, high=length, mid;

while(low<=high){

mid=(low+high)/2;

if(arr[mid] == val) return mid;

if(arr[mid] > val) high = mid-1;

if(arr[mid] < val) low = mid+1;

}

return -1;

}

//Bubble Sort

void bubbleSort(int\* arr, int len){

// outer loop: no. of passes (n-1, observation)

for(int i=0; i<len; i++){

// inner loop: individual pass, compare adjacent elements and swap.

// (0 <= j < n-1)

for(int j=0; j<len-1; j++){

if(arr[j] > arr[j+1]){

// swap operation

int temp = arr[j+1];

arr[j+1] = arr[j];

arr[j] = temp;

}

}

}

}

// Q3. Modified bubble sort

/\* Modification:

1. no. of iterations in each pass only n-i-1 as by each pass i element is sorted from last index.

2. remove redundant passes if array gets sorted already (using flag to check if swaping is done or not)

\*/

void modifiedBubbleSort(int\* arr, int len){

bool flag;

for(int i=0; i<len; i++){

flag = false;

for(int j=0; j<len-1-i; i++){

if(arr[j] > arr[j+1]){

// swap ops

int temp = arr[j+1];

arr[j+1] = arr[j];

arr[j] = temp;

flag = true;

}

}

if(flag == false) break; // array is sorted already

}

}

// Insertion sort

void insertionSort(int\* arr, int len){

for(int i=0; i<len-1; i++){

int j = i+1;

int temp = arr[j];

while(j>0){

if(temp < arr[j-1]){

arr[j] = arr[j-1]; // shifting greater values

j--;

}

else{

break;

}

}

// inserting the element

arr[j] = temp;

}

}

// function for menu

void showMenu(){

int choice;

printf("Array operations menu\n");

printf("1. Create your own array.\n");

printf("2. Traverse\n");

printf("3. Insert\n");

printf("4. Delete\n");

printf("5. Linear Search (first occurance)\n");

printf("6. Count (multiple occurance)\n");

printf("7. Binary Search (ONLY IF SORTED ARRAY!)\n");

printf("8. Sort Array \n");

printf("9. Exit\n");

printf("\nYour choice: ");

scanf("%d", &choice);

switch(choice){

case 1: makeArray(); break;

case 2: traverse(); break;

case 3: {

int loc,val;

printf("\nEnter insert index:");

scanf("%d", &loc);

printf("\nEnter value to insert:");

scanf("%d", &val);

insert(loc, val); break;

}

case 4: {

int loc;

printf("\nEnter delete index:");

scanf("%d", &loc);

deleteItem(loc);

break;

}

case 5: {

int val;

printf("\nEnter value to search:");

scanf("%d", &val);

printf("Index of searched item: %d\n", lsearch(val));

break;

}

case 6: {

int val;

printf("\nEnter value to count:");

scanf("%d", &val);

printf("Count of given value: %d\n", count(val));

break;

}

case 7:{

int val;

printf("\nEnter value to search:");

scanf("%d", &val);

printf("Index of searched item: %d\n", binarySearch(val));

break;

}

case 8: {

bubbleSort(arr);

printf(“Array is sorted successfully!\n”);

}

case 9: exit(1);

default: printf("Invalid option!"); break;

}

}

int main(){

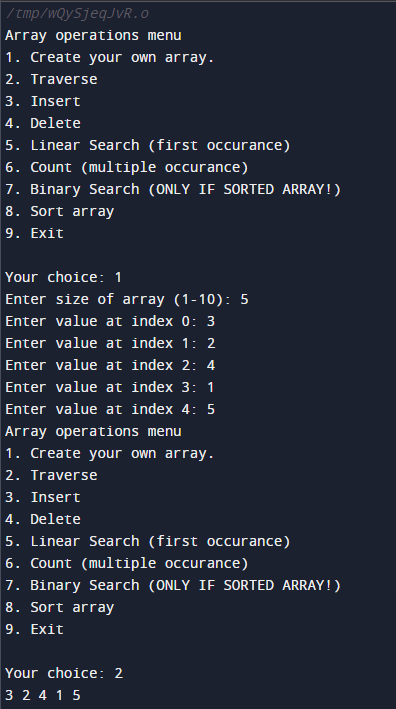
while(1){

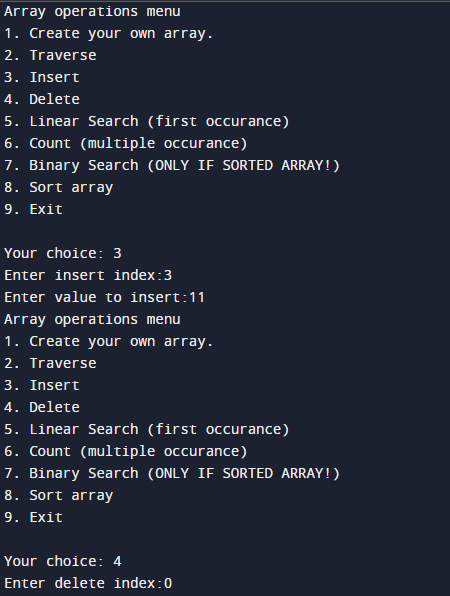
showMenu();

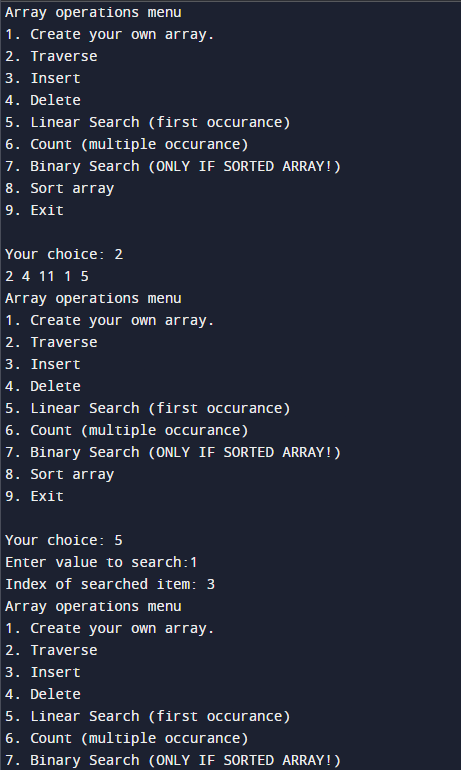
}

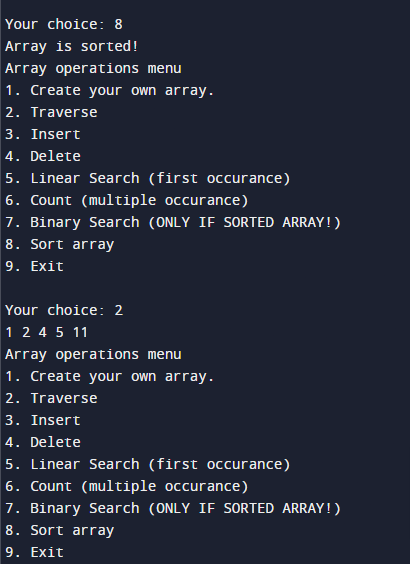
return 0;

}









Q2. Create a program to check whether a given matrix is sparse or not.

#include <stdio.h>

void printMatrix(int n, int arr[][n]){

for(int i=0; i<n; i++){

for(int j=0; j<n; j++){

printf("%d", arr[i][j]);

}

printf("\n");

}

}

int isSparseMatrix(int n, int arr[][n]){

int countZero = 0;

for(int i=0; i<n; i++){

for(int j=0; j<n; j++){

if(arr[i][j] == 0) countZero++;

}

}

return (countZero > (n\*n)/2) ? 1 : 0;

}

int elementCount(int n, int arr[][n]){

int count =0;

for(int i=0; i<n\*n; i++){

if(arr[i%n][i/5] != 0) count++;

}

return count;

}

int main() {

int arr[5][5] = {{5,0,8,0,0}, {0,0,0,9,0}, {0,2,0,0,6}, {0,0,0,1,0}, {0,0,7,0,0}};

printMatrix(5, arr);

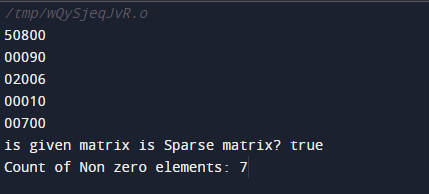
printf("is given matrix is Sparse matrix? %s", isSparseMatrix(5, arr) ? "true" : "false");

int nonZeroCount = elementCount(5, arr);

printf("Count of Non zero elements: %d", nonZeroCount);

return 0;

}



Q3. Create a program to transform a given sparse matrix into its corresponding Three row triplet representation.

#include <stdio.h>

int main() {

int arr[5][5] = {{5,0,8,0,0}, {0,0,0,9,0}, {0,2,0,0,6}, {0,0,0,1,0}, {0,0,7,0,0}};

int threeRowRep[3][7] = {0};

int elementCount = 0;

for(int i=0; i<5; i++){

for(int j=0; j<5; j++){

if(arr[i][j] != 0) {

threeRowRep[0][elementCount] = i; //row value

threeRowRep[1][elementCount] = j; //column value

threeRowRep[2][elementCount] = arr[i][j]; //element

elementCount++;

}

}

}

for(int i=0; i<3; i++){

for(int j=0;j<elementCount;j++){

printf("%d ", threeRowRep[i][j]);

}

printf("\n");

}

return 0;

}

