|  |
| --- |
| **Course**: Programming Fundamental – ENSF 337  **Lab #**: Lab 5  **Instructor**: M. Moussavi  **Student Name**: Aarushi Roy Choudhury  **Lab Section**: B01  **Date submitted**: Nov 1,2021 |

**Exercise A**

Diagram, schematic

Description automatically generated

**Exercise B**

Diagram, schematic

Description automatically generated

**Exercise D**

#include <stdio.h>

#include <stdlib.h>

#include "lab5exe\_D.h"

int main(void) {

    char input\_filename[30] = "lab5exe\_D.txt";

    char output\_filename[30] = "lab5exe\_D\_output.txt";

    IntVector intVec;

    intVec.number\_of\_data = 0;

    read\_text\_file( & intVec, input\_filename);

    display\_single\_column( & intVec);

    display\_multiple\_column( & intVec, 4, output\_filename);

    return 0;

}

void read\_text\_file(IntVector \* vec,

    const char \* input\_filename) {

    int nscan;

    FILE \* fp = fopen(input\_filename, "r");

    if (fp == NULL) {

        fprintf(stdout, "Sorry cannot open the text file %s.\n", input\_filename);

        exit(1);

    }

    do {

        nscan = fscanf(fp, "%d", & vec -> storage[vec -> number\_of\_data]);

        if (nscan == 1)

            (vec -> number\_of\_data) ++;

        else if (nscan != EOF) {

            fprintf(stderr, "Invalid data in %s.\n", input\_filename);

            exit(1);

        }

    } while ((nscan != EOF) & (vec -> number\_of\_data < MAX\_CAPACITY));

    fclose(fp);

}

void display\_single\_column(const IntVector \* intV) {

    int i;

    for (i = 0; i < intV -> number\_of\_data; i++)

        printf("%10d\n", intV -> storage[i]);

}

void display\_multiple\_column(const IntVector \* intV, int col,const char \* output\_filename)

{

        int i;

        FILE \*fptr;

        fptr = fopen(output\_filename,"w");

        for (i = 0; i < intV -> number\_of\_data; i++) {

                if(i>0 && i%col==0) fprintf(fptr,"\n");

        fprintf(fptr, "%10d\t", intV -> storage[i]);

        }

}

A screenshot of a computer screen

Description automatically generated with medium confidence

**Exercise E**

// ENSF 337

//Lab5exe\_E.c

//Completed By: Aarushi Roy Choudhury

#include "lab5exE.h"

#include <stdio.h>

#include <math.h>

#include <string.h>

int main(void){

    Point alpha = {"A1", 2.3, 4.5, 56.00};

    Point beta = {"B1", 25.9, 30.0, 97.00};

    printf ("Display the values in alpha, and beta: ");

    display\_struct\_point(alpha);

    display\_struct\_point(beta);

    Point \*stp = &alpha;

    printf ("\n\nDisplay the values in \*stp: ");

    display\_struct\_point(\*stp);

    Point gamma = mid\_point(stp, &beta, "M1");

    printf ("\n\nDisplay the values in gamma after calling mid\_point function...");

    printf ("\nExpected result is: M1 <14.10, 17.25, 76.50>");

    printf("\n\nThe actual result of calling mid\_point function is: ");

    display\_struct\_point(gamma);

    swap (stp, &beta);

    printf ("\n\nDisplay the values in \*stp, and beta after calling swap function... ");

    printf ("Expected to be: \nB1 = <25.90, 30.00, 97.00> \nA1 = <2.30, 4.50, 56.00>");

    printf("\n\nThe actual result of calling swap function is: ");

    display\_struct\_point(\*stp);

    display\_struct\_point(beta);

    printf("\n\nThe distance between alpha and beta is: %.2f. (Expected to be: 53.74)", distance(&alpha, &beta));

    printf("\nThe distance between gamma and beta is: %.2f. (Expected to be: 26.87) \n", distance(&gamma, &beta));

    return 0;

}

void display\_struct\_point(const Point p){

    printf("\n%s <%.2lf, %.2lf, %.2lf>", p.label, p.x, p.y, p.z);

}

Point mid\_point(const Point \*p1, const Point \*p2, const char \*label){

    //YOU ARE NOT ALLOWED TO USE ANY STRING LIBRARY FUNCTIONS IN THIS FUNCTION

        //Create new point named label that is in the middle of p1 and p2

        int i;

        Point middle;

        double x=((\*p1).x+(\*p2).x)/2;

        double y=((\*p1).y+(\*p2).y)/2;

        double z=((\*p1).z+(\*p2).z)/2;

        for(i=0; label[i] != '\0'; i++){

                middle.label[i]=label[i];

        }

        middle.label[i]='\0';

        middle.x=x;

        middle.y=y;

        middle.z=z;

    return middle;

}

void swap(Point \*p1, Point \*p2){

        //Swaps the values of p1 and p2

        Point temp=\*p1;

        \*p1=\*p2;

        \*p2=temp;

}

double distance(const Point \*p1, const Point \*p2){

    //YOU ARE NOT ALLOWED TO USE THE ARROW OPERATOR (->)

        //Finds the distance between p1 and p2

        double d=sqrt(pow(((\*p1).x-(\*p2).x), 2) + pow(((\*p1).y-(\*p2).y), 2) + pow(((\*p1).z-(\*p2).z), 2));

    return d;

}

Text

Description automatically generated

**Exercise F**

#include "lab5exF.h"

#include <stdio.h>

#include <math.h>

#include<string.h>

int main(void){

    Point struct\_array[10];

    int i;

    int position;

    populate\_struct\_array(struct\_array, 10);

    printf("Array of Points contains: \n");

    for(i=0; i < 10; i++){

        display\_struct\_point(struct\_array[i], i);

        }

    printf("\nTesting the search function... \n");

    position = search(struct\_array, "v0", 10);

    if(position != -1){

        printf("\nFound: struct\_array[%d] contains %s", position, struct\_array[position].label);

        }

    else{

        printf("\nstruct\_array doesn't have label: %s.", "v0");

        }

    position = search(struct\_array, "E1", 10);

    if(position != -1){

        printf("\nFound: struct\_array[%d] contains %s", position, struct\_array[position].label);

        }

    else{

        printf("\nstruct\_array doesn't have label: %s.", "E1");

        }

    position = search(struct\_array, "C5", 10);

    if(position != -1){

        printf("\nFound: struct\_array[%d] contains %s", position, struct\_array[position].label);

        }

    else{

        printf("\nstruct\_array doesn't have label: %s.", "C5");

        }

    position = search(struct\_array, "B7", 10);

    if(position != -1){

        printf("\nFound: struct\_array[%d] contains %s", position, struct\_array[position].label);

        }

    else{

        printf("\nstruct\_array doesn't have label: %s.", "B7");

        }

    position = search(struct\_array, "A9", 10);

    if(position != -1){

        printf("\nFound: struct\_array[%d] contains %s", position, struct\_array[position].label);

        }

    else{

        printf("\nstruct\_array doesn't have label: %s.", "A9");

        }

    position = search(struct\_array, "E11", 10);

    if(position != -1){

        printf("\nFound: struct\_array[%d] contains %s", position, struct\_array[position].label);

        }

    else{

        printf("\nstruct\_array doesn't have label: %s.", "E11");

        }

    position = search(struct\_array, "M1", 10);

    if(position != -1){

        printf("\nFound: struct\_array[%d] contains %s \n", position, struct\_array[position].label);

        }

    else{

        printf("\nstruct\_array doesn't have label: %s. \n", "M1");

    }

    printf("\nTesting the reverse function... \n");

    reverse(struct\_array, 10);

    printf("\nThe reversed array is: \n");

    for(i=0; i < 10; i++)

        display\_struct\_point(struct\_array[i], i);

    return 0;

}

void display\_struct\_point(const Point x , int i){

    printf("struct\_array[%d]: %s <%.2lf, %.2lf, %.2lf> \n", i, x.label, x.x, x.y, x.z);

}

void populate\_struct\_array(Point\* array, int n){

    int i;

    char ch1 = 'A';

    char ch2 = '9';

    char ch3 = 'z';

    for(i = 0; i < n; i++){

        /\* generating some random values to fill them elements of the array: \*/

        array[i].x = (7 \* (i + 1) % 11) \* 100 - i / 2;

        array[i].y = (7 \* (i + 1) % 11) \* 120 - i / 3;

        array[i].z = (7 \* (i + 1) % 11) \* 150 - i / 4;

        if(i % 2 == 0){

            array[i].label[0] = ch1++;

                }

        else{

            array[i].label[0] = ch3--;

                }

        array[i].label[1] = ch2--;

        array[i].label[2] = '\0';

    }

}

int search(const Point\* struct\_array, const char\* target, int n){

    //YOU ARE NOT ALLOWED TO USE ANY C LIBRARY FUNCTION IN YOUR SOLUTION

        //Returns index of first occurence of target in struct\_array

        int i=0, j=0, index=0;

        if(struct\_array[0].label[0]=='\0'){

                return -1;

        }

        while((struct\_array[i].label[j] != '\0' || target[j] != '\0') && i<n){       //Cycles through struct\_array to check if label matches target

                if(struct\_array[i].label[j]==target[j]){

                        index=i;

                        j++;

                }

                else if((struct\_array[i].label[j] != '\0' && target[j]=='\0') || (struct\_array[i].label[j]=='\0' && target[j] != '\0') || (struct\_array[i].label[j] != target[j])){

                        index=-1;

                        i++;

                }

        }

    return index;

}

void reverse (Point \*a, int n){

    //Reverses elements of array a with length n

        Point temp[n];

        int i, j, k, l;

        if(a!=NULL){                            //if a is not empty...

                for(i=0; i < n; i++){        //Copying a into temp

                        temp[i].x=a[i].x;

                        temp[i].y=a[i].y;

                        temp[i].z=a[i].z;

                        for(j=0; a[i].label[j] != '\0'; j++){

                                temp[i].label[j]=a[i].label[j];

                        }

                        temp[i].label[j]='\0';

                }

                for(k=0; n >= 1; k++){               //Reversing a

                        a[k].x=temp[n-1].x;

                        a[k].y=temp[n-1].y;

                        a[k].z=temp[n-1].z;

                        for(l=0; a[k].label[l] != '\0'; l++){

                                a[k].label[l]=temp[n-1].label[l];

                        }

                        a[k].label[l]='\0';

                        n--;

                      }

                }

        else{

                printf("Array could not be reversed");

        }

}

Text

Description automatically generated