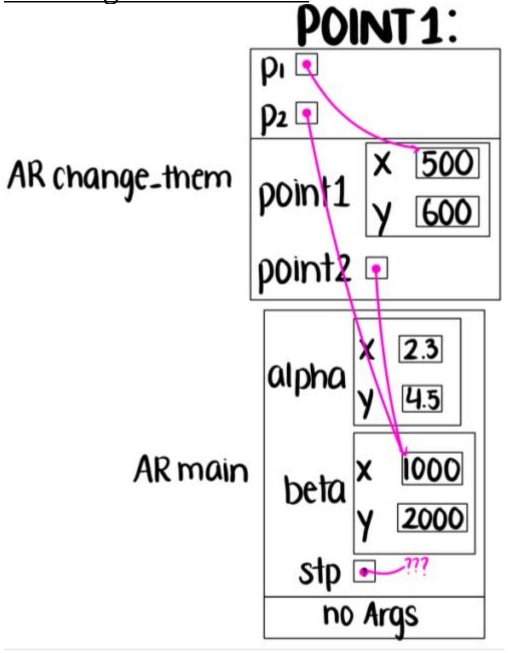
Lab Report

Course: ENSF 337 – Programming Fundamentals for Software and Computer
Lab #: 5
Instructor: Dr. Maan Khedr
Student Name: Sadia Khandaker
Lab Section: B04
Date Submitted: November 1, 2021

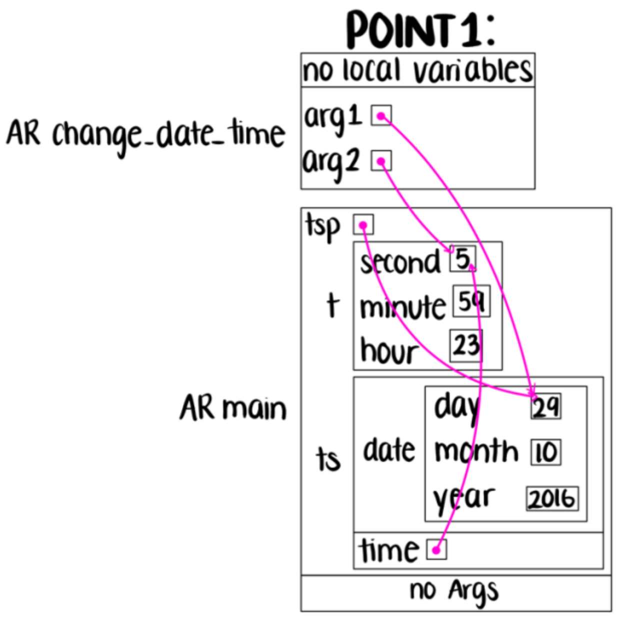
Exercise A: C Struct Objects on the Computer Memory

AR Diagram at Point 1:



Exercise B: Nested Structure

AR Diagram at Point 1:



Exercise D: Writing Into a Text File

Function Definition of `display_multiple_column`:

```
void display_multiple_column(const IntVector *intV, int col, const char*
output_filename)
{
    int i;
    FILE *fp;
    fp = fopen(output_filename, "w");
    if(fp == NULL) {
        printf("Sorry, cannot open text file %s.\n", output_filename);
        exit(1);
    }

    for (i = 0; i < intV ->number_of_data; i++ ) {
        if(i % col == 0)
            fprintf(fp, "\n");
        fprintf(fp, "%10d", intV->storage[i]);
    }
    fclose(fp);
}
```

Preview of `lab6exe_D_output.txt`:

```
 234       678       999       234
  33        22        99       222
  45        56        44        77
  92        91        81        73
  19        18        17       666
 555         1         3         6
```

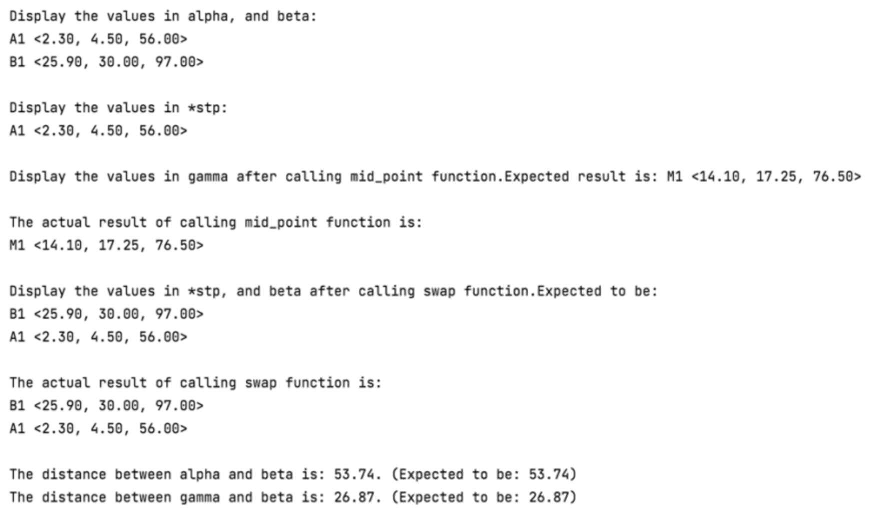
Exercise E: Writing Functions That Use C Struct

Source Code:

```
/* File: lab5exE.c
 * Assignment: Lab 5 Exercise E
 * Lab Section: B04
 * Completed By: Sadia Khandaker
 */

#include "lab5exE.h"
#include <stdio.h>
#include <math.h>
#include <string.h>

int main(void)
{
    Point alpha = { "A1", 2.3, 4.5, 56.0} ;
    Point beta = { "B1", 25.9, 30.0, 97.0 } ;
    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 ("Expected 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. ",
distance(&alpha, &beta));
    printf ("(Expected to be: 53.74)");
    printf("\nThe distance between gamma and beta is: %.2f. ",
distance(&gamma, &beta));
    printf ("(Expected to be: 26.87)");
    return 0;
}
```

```c
void display_struct_point(const Point x)
{
    printf("\n%s <%.2lf, %.2lf, %.2lf>", x.label, x.x, x.y, x.z);
}


Point mid_point(const Point* p1, const Point* p2, const char* label)
{
    Point middle = {"?", 0, 0,0};
    double x=((*p1).x+(*p2).x)/2.0, y=((*p1).y+(*p2).y)/2.0,
z=((*p1).z+(*p2).z)/2.0;
    int i;
    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)
{
    Point temp = *p1;
    *p1 =*p2;
    *p2 = temp;
}

double distance(const Point* p1, const Point* p2)
{
    double d, x1 = (*p1).x, x2 = (*p2).x,y1 = (*p1).y,y2 = (*p2).y,z1 =
(*p1).z,z2 = (*p2).z;
    d = sqrt(pow((x2-x1),2) + pow((y2-y1),2) + pow((z2-z1),2));
    return d;
}
```

<u>Output:</u>



```
Display the values in alpha, and beta:
A1 <2.30, 4.50, 56.00>
B1 <25.90, 30.00, 97.00>

Display the values in *stp:
A1 <2.30, 4.50, 56.00>

Display the values in gamma after calling mid_point function.Expected result is: M1 <14.10, 17.25, 76.50>

The actual result of calling mid_point function is:
M1 <14.10, 17.25, 76.50>

Display the values in *stp, and beta after calling swap function.Expected to be:
B1 <25.90, 30.00, 97.00>
A1 <2.30, 4.50, 56.00>

The actual result of calling swap function is:
B1 <25.90, 30.00, 97.00>
A1 <2.30, 4.50, 56.00>

The distance between alpha and beta is: 53.74. (Expected to be: 53.74)
The distance between gamma and beta is: 26.87. (Expected to be: 26.87)
```

Exercise F: Using Array of Structures

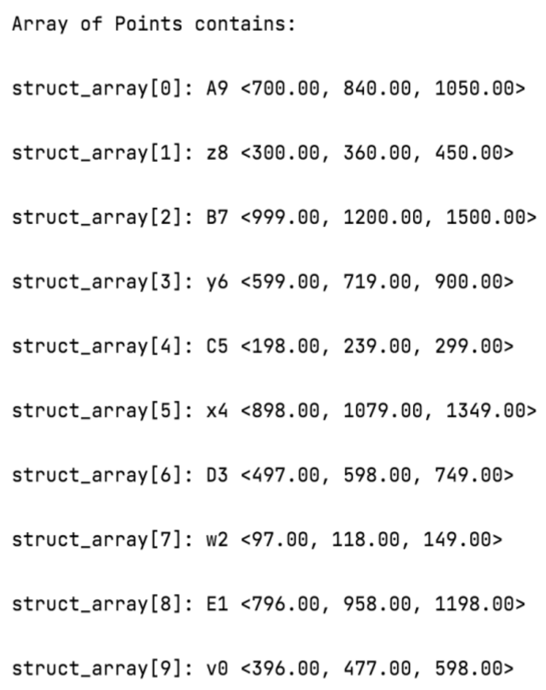
Source Code:

```
/* File Name: lab5exF.c
 * Assignment: Lab 5 Exercise F
 * Lab Section: B04
 * Completed by: Sadia Khandaker
 */

#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("\nArray of Points contains: \n");

    for(i=0; i < 10; i++)
        display_struct_point(struct_array[i], i);


    printf("\nTest the search function");

    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)
```

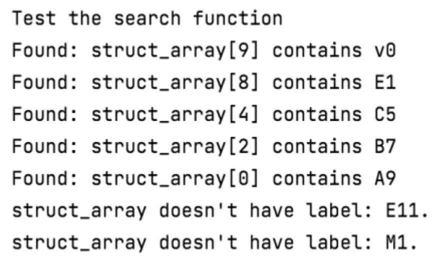
```
        printf("\nFound: struct_array[%d] contains %s", position,
               struct_array[position].label);
    else
        printf("\nstruct_array doesn't have label: %s.", "B7");
    position = search(struct_array, "A9", 10);
    if(position != -1)
        printf("\nFound: struct_array[%d] 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", position,
               struct_array[position].label);
    else
        printf("\nstruct_array doesn't have label: %s.", "M1");

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

    reverse(struct_array, 10);

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

    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("\nstruct_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 < 10; 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;
```

```c
        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* label, int n)
{
    for(int i = 0; i < n; i++) {
        int j = 0, temp = 0;
        while(label[j]!='\0' && struct_array[i].label[j]!='\0') {
            if(label[j]!=struct_array[i].label[j]) {
                temp=1;
                break;
            }
            j++;
        }
        if(temp==0 && label[j]=='\0' && struct_array[i].label[j]=='\0') {
            return i;
        }
    }
    return -1;
}

void reverse (Point *a, int n)
{
    int i=0,j=n-1;
    while (i<j){
        Point temp = a[i];
        a[i]=a[j];
        a[j]=temp;
        i++;
        j--;

    }
}
```

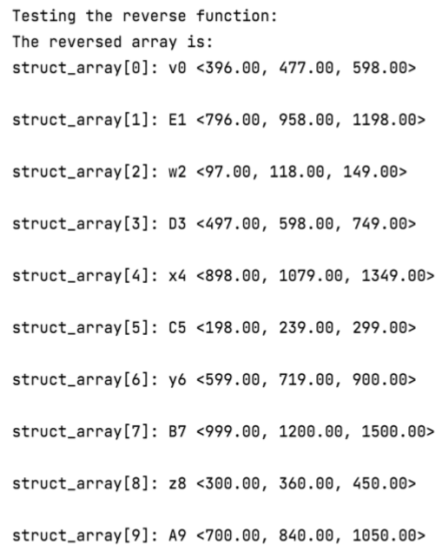
Output:



```
Array of Points contains:

struct_array[0]: A9 <700.00, 840.00, 1050.00>

struct_array[1]: z8 <300.00, 360.00, 450.00>

struct_array[2]: B7 <999.00, 1200.00, 1500.00>

struct_array[3]: y6 <599.00, 719.00, 900.00>

struct_array[4]: C5 <198.00, 239.00, 299.00>

struct_array[5]: x4 <898.00, 1079.00, 1349.00>

struct_array[6]: D3 <497.00, 598.00, 749.00>

struct_array[7]: w2 <97.00, 118.00, 149.00>

struct_array[8]: E1 <796.00, 958.00, 1198.00>

struct_array[9]: v0 <396.00, 477.00, 598.00>
```



```
Test the search function
Found: struct_array[9] contains v0
Found: struct_array[8] contains E1
Found: struct_array[4] contains C5
Found: struct_array[2] contains B7
Found: struct_array[0] contains A9
struct_array doesn't have label: E11.
struct_array doesn't have label: M1.
```



```
Testing the reverse function:
The reversed array is:
struct_array[0]: v0 <396.00, 477.00, 598.00>

struct_array[1]: E1 <796.00, 958.00, 1198.00>

struct_array[2]: w2 <97.00, 118.00, 149.00>

struct_array[3]: D3 <497.00, 598.00, 749.00>

struct_array[4]: x4 <898.00, 1079.00, 1349.00>

struct_array[5]: C5 <198.00, 239.00, 299.00>

struct_array[6]: y6 <599.00, 719.00, 900.00>

struct_array[7]: B7 <999.00, 1200.00, 1500.00>

struct_array[8]: z8 <300.00, 360.00, 450.00>

struct_array[9]: A9 <700.00, 840.00, 1050.00>
```