Programming Fundamentals – ENSF 337

Lab 5

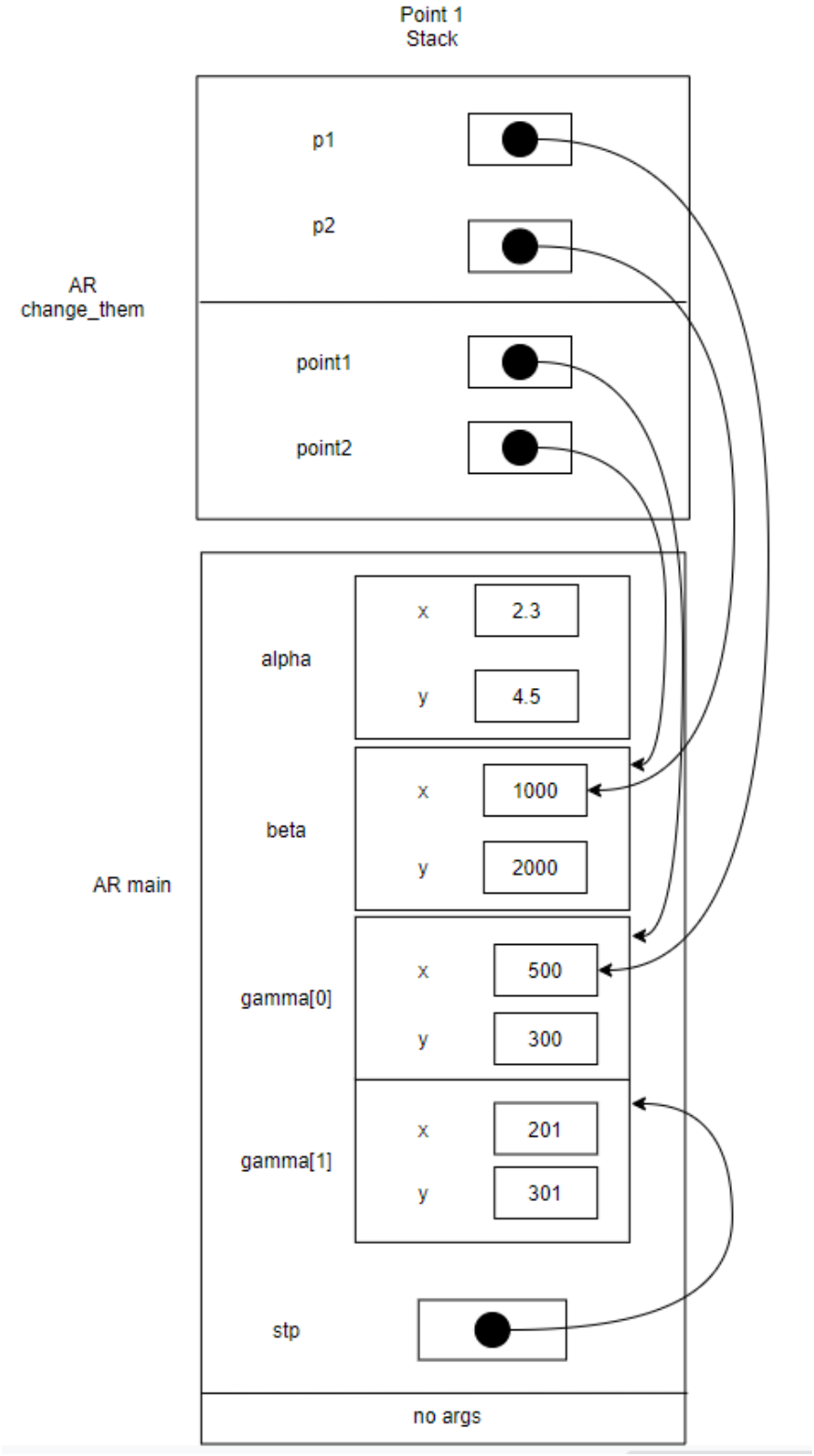
M. Moussavi

Jaxon Braun

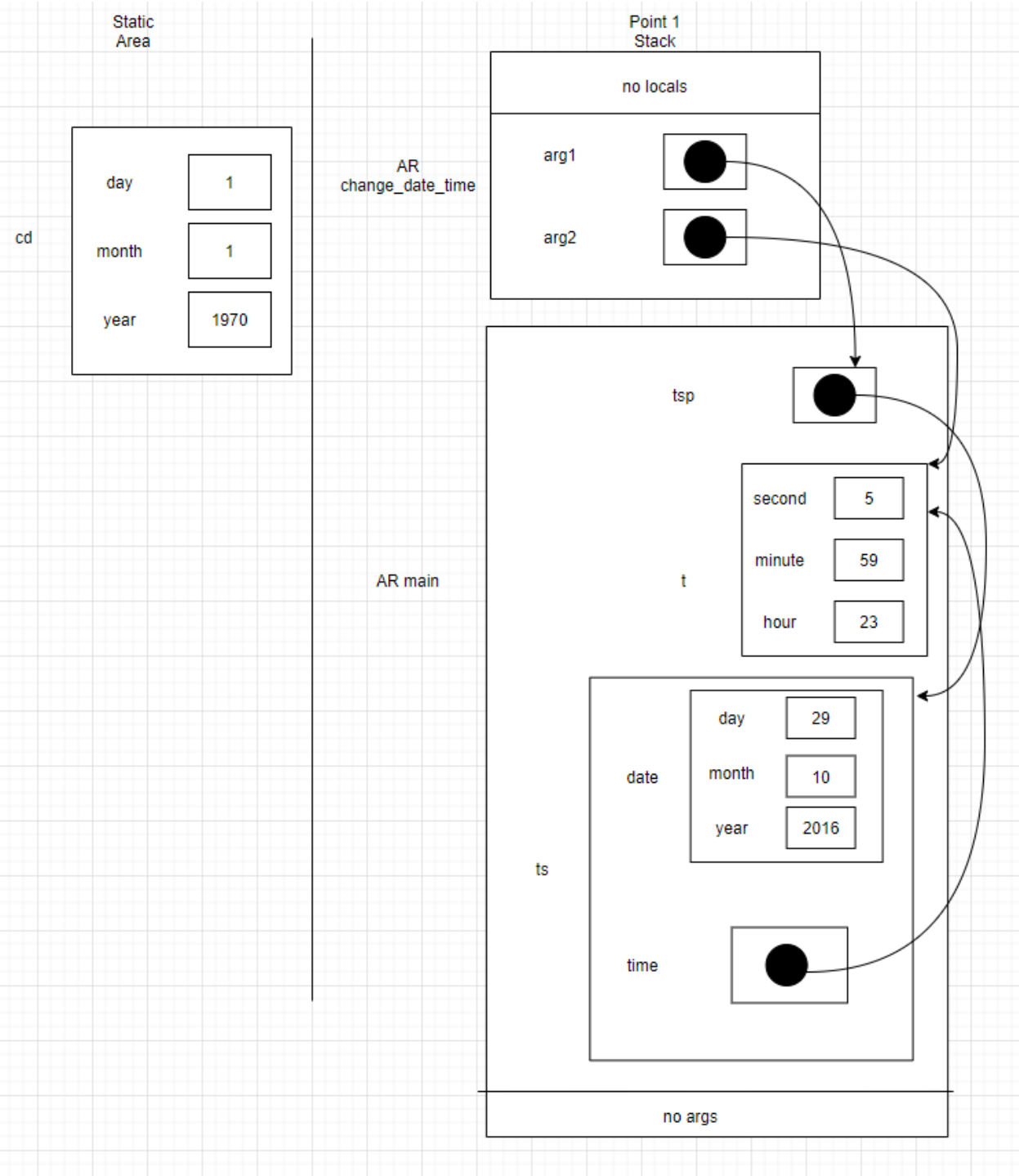
B01

Submitted on October 30, 2021

Exefcise A: C struct Objects on the Computer Memory



Exercise B: Nested Structure

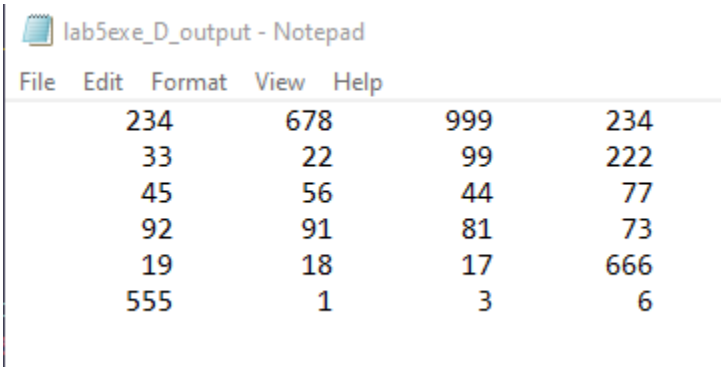


Exercise D: Writing into a Text File

```
void display_multiple_column(const IntVector *intV, int col, const char* output_filename)
{
    FILE * fPtr = fopen(output_filename, "w");

    int i;
    int j;
    int counter = 0;
    if (col >= intV ->number_of_data){
        for (i = 0; i < intV ->number_of_data; i++)
            fprintf(fPtr, "%10d", intV ->storage[i]);
    }
    else{
        for (j = 0; j < (intV ->number_of_data)/(col); j++){
            int k = 0;
            while (k < col){
                fprintf(fPtr, "%10d", intV ->storage[counter]);
                counter++;
                k++;
            }
            fprintf(fPtr, "\n");
        }
    }

    fclose(fPtr);
}
```



Exercise E: Writing Functions that Use C struct

```
// ENSF 337 - lab 5 - Exercise E
// Created by: M. Moussavi
// lab5exe_E.c
// Finished by: Jaxon Braun
// Submission Date: October 30, 2021

#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;
}

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)
{
```

```
    double x_m = (p1->x + p2->x)/2;
    double y_m = (p1->y + p2->y)/2;
    double z_m = (p1->z + p2->z)/2;

    Point middle = {};
    for (int i = 0; i < 10; i++)
        middle.label[i] = *(label+i);
    middle.x = x_m;
    middle.y = y_m;
    middle.z = z_m;
    return middle;
}

void swap(Point* p1, Point *p2)
{
    Point temp = *p1;

    *(p1)->label = *(p2)->label;
    p1->x = p2->x;
    p1->y = p2->y;
    p1->z = p2->z;

    for (int i = 0; i < 10; i++)
        p2->label[i] = temp.label[i];
    p2->x = temp.x;
    p2->y = temp.y;
    p2->z = temp.z;
}
```

```
double distance(const Point* p1, const Point* p2)
{
    Point p1_d = *p1;
    Point p2_d = *p2;

    double d_x = pow(p1_d.x - p2_d.x, 2);
    double d_y = pow(p1_d.y - p2_d.y, 2);
    double d_z = pow(p1_d.z - p2_d.z, 2);

    double distance = sqrt(d_x + d_y + d_z);
    return distance;
}
```

```
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

```
// lab5exF.c
// Created by: M. Moussavi
// lab5exeF.c
// Finished by: Jaxon Braun
// Submission Date: October 30, 2021

#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;

        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 counter = 0;
        for (int j = 0; j < 2; j++){
            if (struct_array[i].label[j] == label[j])
                counter++;
        }
        if (counter == 2)
            return i;
    }
    return -1;
}

void reverse (Point *a, int n)
{
    Point a_temp[n];
    for (int i = 0; i < n; i++)
        a_temp[i] = a[i];
    for (int j = 0; j < n; j++){
        a[j] = a_temp[n-1-j];
    }
}
```

```
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
Found: struct_array[8] contains E1
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>
```