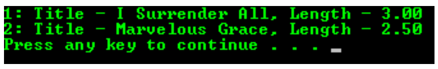
To:
Tenouk

C LAB WORKSHEET 14a
C Structures, struct Part 2

1. More on using C structure data type.
2. More questions and activities on structure, array and function.
3. Tutorial reference that should be used together with this worksheet is: C/C++ type specifiers – struct, union, typedef.

4. Show the output and answer the questions for the following program.

```
#include <stdio.h>

void main()
{
   struct Song
   {
         char Name[25];
         float Length;
   };

   struct Song
         Title1 = {"Marvelous Grace", 2.50},
         Title2 = {"I Surrender All", 3.0},
         Title3;

   Title3 = Title1;
   Title1 = Title2;
   Title2 = Title3;

   printf("1: Title - %s, Length - %.2f\n", Title1.Name, Title1.Length);
   printf("2: Title - %s, Length - %.2f\n", Title2.Name, Title2.Length);
}
```



```
1: Title - I Surrender All, Length - 3.00
2: Title - Marvelous Grace, Length - 2.50
Press any key to continue . . .
```

a. When initializing structures, do you have to provide the value of the first member first or can it be placed second?
b. Can you leave out the value of one of the members as shown in the following code? Does this work?

   struct Song Title1 = { , 2.50}, Title2 = {"I Surrender All", },
   Title3;

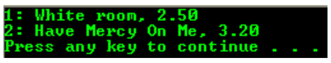
c. How many variables of type struct Song were created?
d. What was done to the original values of Title1 and Title2?

a. We need to provide the value in order, similar to the declaration.
b. No we can't.
c. Three. That are Title1, Title2 and Title3.
d. The original value of Title1 was overwritten by Title2 and the original value of Title2 was overwritten by Title3 which was copied from Title1.

5. Show the output and answer the questions for the following program.

```
#include <stdio.h>
#include <string.h>

// start of structure definition
struct Song
{
   char Name[25];
   float Length;
}; // end of structure definition

// function prototype, receives and returns structure data type
struct Song ZeroOut(struct Song);

void main()
{
   struct Song Title1 = {"White room", 2.50}, Title2;
   Title2 = ZeroOut(Title1);
   printf("1: %s, %.2f\n", Title1.Name, Title1.Length);
   printf("2: %s, %.2f\n", Title2.Name, Title2.Length);
}

// beginning of function definition
struct Song ZeroOut(struct Song x)
{
   struct Song y = {"Have Mercy On Me", 3.20F};
   strcpy_s(x.Name, 25, " ");
   x.Length = 0;
   return y;
} // end of function definition
```

a. The structure is not defined inside any function. If it were defined inside the main() function, as was done in the previous exercises, will this program work? Why or why not? Try it.
b. Let us first concentrate on how functions receive arguments that are structures. As shown in the program, the definition of the structure is global, that is, it is available to all functions. Title1 in main() becomes what variable in ZeroOut()?
c. ZeroOut() erased the contents of what was provided through the variable Title1. The contents of Title1 also erased when control went back to main()?
d. When passing a structured variable to a function, can that function



```
1: White room, 2.50
2: Have Mercy On Me, 3.20
Press any key to continue . . .
```

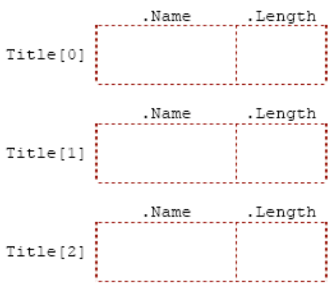
change its contents?
e. Do functions make copies of structures like they make copies of scalars?
f. Do functions receive only addresses of structures, as they do of arrays?
g. Let us now consider how functions return structures. What is the data type that ZeroOut() returns?
h. What variable is returned by ZeroOut()? What is its data type? What are its values?
i. Which variables in main() accepts the returned structure from ZeroOut()?
j. Were the values returned by ZeroOut() received in main()? If so, then into which variable?

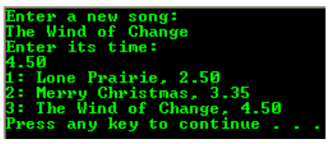
a. The program won't work because the structure was used in the function prototype. We need to declare it before using it.
b. Title1 becomes x in ZeroOut().
c. No.
d. No. The function cannot change its contents. In the function we try to assign an empty string to x.Name and 0 to the x.Length. However, when we return to main() the original value still intact.
e. Yes function makes copy of structure like a scalar.
f. No, it is like a scalar.
g. struct Song type.
h. y variable of type struct Song.
i. Title2 variable of type struct Song.
j. Yes, those values received in main() and assigned to Title2 of type struct Song.

6. How about an array of structure. Show the output and answer the questions for the following program.

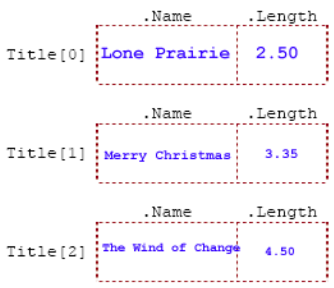
```
#include <stdio.h>
#include <string.h>

struct Song
{
   char Name[25];
   float Length;
};

void main()
{
   struct Song Title[3] =
   {{"Lone Prairie", 2.50F},
   {"Merry Christmas", 3.35F}};

   int i;

   puts("Enter a new song:");
   gets_s(Title[2].Name, 25);
   puts("Enter its time:");
   scanf_s("%f", &Title[2].Length, 4);

   for(i = 0; i <= 2; ++i)
         printf("%d: %s, %.2f\n", i + 1, Title[i].Name, Title[i].Length);
}
```

a. Remember that indexing starts at 0 and enter the initialized values of the array in the diagram.
b. Show the values in the following diagram that you provided in the array of structures.





```
Enter a new song:
The Wind of Change
Enter its time:
4.50
1: Lone Prairie, 2.50
2: Merry Christmas, 3.35
3: The Wind of Change, 4.50
Press any key to continue . . .
```

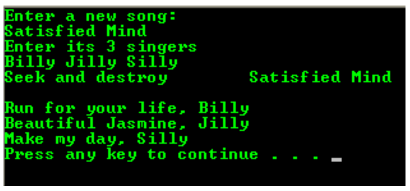


a. ..
b. ..
c. Title, with three slots.
d. Song. Name and Length.
e. Don't be confused. The index is for the Song structure not for the structures' elements. The &Title.Length[2] is an array for the structure's element. So, there are array of structures and array of arrays' elements. Both are valid and must be differentiated clearly.
f. Array name, index and then member name.
g. Brackets.
h. A period.

c. What is the name of the array? How many slots does it have?
d. What is the name of the structure? What are the names of its members?
e. Why is the index provided next to the array name and not the member name, in all the instances of this exercise? Why the following isn't correct: &Title.Length[2]?
f. When accessing any item in this array of structures, what must be specified first, second and last? Choose your answers from: member name, array name, index.
g. An array slot is specified by using brackets or a period?
h. A structure member is specified by using brackets or a period?

7. Next, let us have an array of names within a structure. Enter “Satisfied Mind”, “Billy”, “Jilly” and “Silly” for the sample input data. Show the output and answer the questions.

```
#include <stdio.h>

struct Song
{
   char Name[25];
   char Singer[3][25];
   float Length;
};

void main(void)
{
   int i;
   struct Song Title1 =
   {"Seek and destroy", "Run for your life",
   "Beautiful Jasmine", "Make my day", 2.50}, Title2;

   puts("Enter a new song:");
   gets_s(Title2.Name, 25);
   puts("Enter its 3 singers");
   scanf_s("%s", &Title2.Singer[0], 25);
```

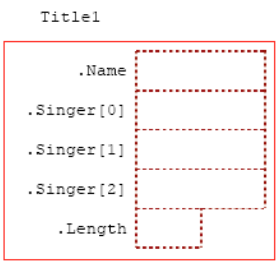


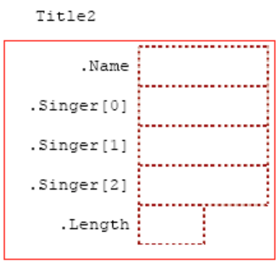
```
Enter a new song:
Satisfied Mind
Enter its 3 singers
Billy Jilly Silly
Seek and destroy         Satisfied Mind

Run for your life, Billy
Beautiful Jasmine, Jilly
Make my day, Silly
Press any key to continue . . . _
```

a. A usual, it is based on the array's index. Index 0 will store string 1 as in Title2.Singer[0], index 1 will store string 2 as in Title2.Singer[1] and index 2 will store string 3 as in Title2.

```
    scanf_s("%s", &Title2.Singer[1], 25);
    scanf_s("%s", &Title2.Singer[2], 25);
    printf("%s\t%s\n", Title1.Name, Title2.Name);
    printf("\n");
    for(i = 0; i <= 2; ++i)
        printf("%s, %s\n", Title1.Singer[i], Title2.Singer[i]);
}
```

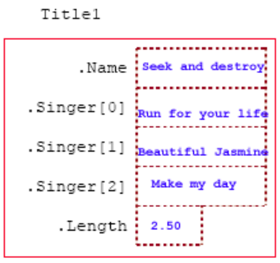
a. So far our structure has had one member that was an array. However, we treated it as a single unit or a string. Here, we have a 3-slot array of strings that hold the names of singers. When initializing, how do we know which string goes into which slot of the structure?
b. Show the contents of all locations in the following Figure.

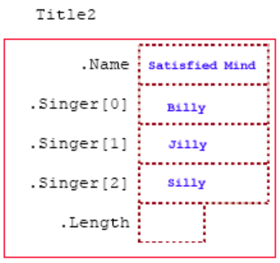




c. When reading in the name member of Title2, why is no array index given?
d. When reading in the singers for Title2, why is an index specified?
e. When printing the song names, are array indexes specified? Why or why not?
f. When printing out the singers, are array indexes specified? Why and why not?
g. By looking at the structure definition, how can you tell which member needs an index specified?
h. In exercise #6, was the variable name or the member name specified with an index?
i. By looking at the structure and/or variable definitions, how can you tell if the variable name or the structure member name gets an index?
j. In this exercise, was the variable name or the member name specified with an index? How can you tell which one should be?
k. Show how you would have to change the definition of the structure and/or the variable so that both indexes would have to be given when accessing a singer.

Singer[2].





b. ...

c. The name string is considered as one element. The array has been used to store the individual character and the index of the array is one dimensional.
d. In this case we have three strings to be read and stored into three different slots and the arrays are two dimensional.
e. Not specified because the song names have been considered one element and the index is one dimensional.
f. The array indexes are specified because every singer name has been considered a different string.
g. Based on the dimensional of the index. If it is two dimensional, then we need to specify an index.
h. In exercise #6, the structure member and the structure variable specified with an index and this is a structure that contains an array element and then declared as an array variable.
i. The use of the square brackets ([ ]). Which one or both that have the brackets get an index.
j. In this exercise the structure member names specified with an index as char Name[25]; and char Singer[3][25];
k. We need to declare the structure member and the structure variables as an array type as shown below.

```
    struct Song
    {
        char Name[25];
        char Singer[3][25];
        float Length;
    };
...
struct Song Title3[3];
...
Title3[0].Singer[0] = { "May day"};
```

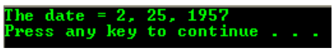
More Structure Questions

For the following questions, use the definitions of the given structure.

```
struct Date
{
    int yy, mm, dd;
};

struct Emp
{
    char EmpName[25];
    float Salary;
    struct Date hired;
};

struct Dep
{
    struct Emp manager;
    struct Emp worker[25];
    float Profits;
};
```

1. Define a struct Date variable called Date1 and initialize it to February 25, 1957, in the correct format.

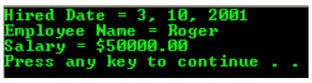
```
struct Date
{
    int yy, mm, dd;
};

void main(void)
{
    struct Date Date1;

    Date1.dd = 25;
    Date1.mm = 2;
    Date1.yy = 1957;

    printf("The date = %d, %d, %d\n", Date1.mm, Date1.dd, Date1.yy);
}
```



```
The date = 2, 25, 1957
Press any key to continue . . .
```

2. Define a struct Emp variable called Person1 and initialize it to “Roger”, with a salary of $50,000, who was hired on March 10, 2001.

```
#include <stdio.h>

struct Date
{
   int yy, mm, dd;
};

struct Emp
{
   char EmpName[25];
   float Salary;
   struct Date hired;
};

void main(void)
{
   struct Emp Person1 = {"Roger"};

   Person1.Salary = 50000;
   Person1.hired.mm = 3;
   Person1.hired.dd = 10;
   Person1.hired.yy = 2001;

   printf("Hired Date = %d, %d, %d\n", Person1.hired.mm, Person1.hired.dd, Person1.hired.yy);
   printf("Employee Name = %s\n", Person1.EmpName);
   printf("Salary = $%.2f\n", Person1.Salary);
}
```

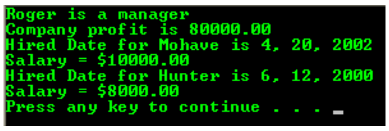


```
Hired Date = 3, 10, 2001
Employee Name = Roger
Salary = $50000.00
Press any key to continue . .
```

3. Define a struct Dep variable called Toys whose manager is “Roger”, from question 2 above and the profits are $80,000. This department has only two employees as shown below:

   “Mohave” with a salary of 10,000 and the date of hire being April 20, 2002
   “Hunter” with a salary of 8,000 and the date of hire being June 12, 2000

```
#include <stdio.h>

struct Date
{
   int yy, mm, dd;
};

struct Emp
{
   char EmpName[25];
   float Salary;
   struct Date hired;
};

struct Dep
{
   struct Emp manager;
   struct Emp worker[25];
   float Profits;
};

void main(void)
{
   // note the order of the arrangement
   // {EmpName,Salary,{yy,mm,dd}},{{EmpName[0], Salary,{yy,mm,dd}},
   //{EmpName[1],Salary,{yy,mm,dd}},..., {EmpName[24],Salary,{yy,mm,dd}}},
   //{Profits}}
   struct Dep Toys = {{"Roger"}, {{"Mohave"},{"Hunter"}}};

   Toys.Profits = 80000;
   Toys.worker[0].hired.mm = 4;
   Toys.worker[0].hired.dd = 20;
   Toys.worker[0].hired.yy = 2002;
   Toys.worker[0].Salary = 10000;

   Toys.worker[1].hired.mm = 6;
   Toys.worker[1].hired.dd = 12;
   Toys.worker[1].hired.yy = 2000;
   Toys.worker[1].Salary = 8000;

   printf("%s is a manager\n", Toys.manager.EmpName);
   printf("Company profit is %.2f\n", Toys.Profits);
   printf("Hired Date for %s is %d, %d, %d\n", Toys.worker[0].EmpName, Toys.worker[0].hired.
mm = 4, Toys.worker[0].hired.dd = 20, Toys.worker[0].hired.yy = 2002);
   printf("Salary = $%.2f\n", Toys.worker[0].Salary);
   printf("Hired Date for %s is %d, %d, %d\n", Toys.worker[1].EmpName, Toys.worker[1].hired.
mm = 6, Toys.worker[1].hired.dd = 12, Toys.worker[1].hired.yy = 2000);
   printf("Salary = $%.2f\n", Toys.worker[1].Salary);
}
```



```
Roger is a manager
Company profit is 80000.00
Hired Date for Mohave is 4, 20, 2002
Salary = $10000.00
Hired Date for Hunter is 6, 12, 2000
Salary = $8000.00
Press any key to continue . . .
```

4. Using one printf(), print the content of Date1.

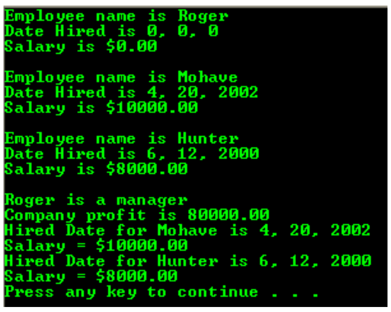
See #1

5. Using two printf()’s, print the contents of Person1.

See #2

6. Write a function called PrintEmp() that will receive a structure of type struct Emp and print its EmpName, Salary and Date hired. Calling this function three times and using an extra printf(), print the contents of Toys. You may want to draw a diagram to help you see the components of Toys.

```
#include <stdio.h>

void PrintEmp(struct Emp);

struct Date
{
    int yy, mm, dd;
};

struct Emp
{
    char EmpName[25];
    float Salary;
    struct Date hired;
};

struct Dep
{
    struct Emp manager;
    struct Emp worker[25];
    float Profits;
};

void main(void)
{
    // note the order of the arrangement
    // {EmpName,Salary,{yy,mm,dd}}
    // Let make it something like this for easiness
    // struct Emp Names0 = {"Roger"};
    // struct Emp Names1 = {"Mohave", 10000, {4, 20, 2002}};
    // struct Emp Names2 = {"Hunter", 8000, {6,12,2000}};
    // Then convert it to an array
    struct Emp Names[3] = {{"Roger"}, {"Mohave", 10000, {2002, 4, 20}}, {"Hunter", 8000, {2000,6,12}}};
    struct Dep Toys = {{"Roger"}, {{"Mohave"},{"Hunter"}}};

    PrintEmp(Names[0]);
    printf("\n");
    PrintEmp(Names[1]);
    printf("\n");
    PrintEmp(Names[2]);
    printf("\n");

    Toys.Profits = 80000;
    Toys.worker[0].hired.mm = 4;
    Toys.worker[0].hired.dd = 20;
    Toys.worker[0].hired.yy = 2002;
    Toys.worker[0].Salary = 10000;

    Toys.worker[1].hired.mm = 6;
    Toys.worker[1].hired.dd = 12;
    Toys.worker[1].hired.yy = 2000;
    Toys.worker[1].Salary = 8000;

    printf("%s is a manager\n", Toys.manager.EmpName);
    printf("Company profit is %.2f\n", Toys.Profits);
    printf("Hired Date for %s is %d, %d, %d\n", Toys.worker[0].EmpName, Toys.worker[0].hired.mm = 4, Toys.worker[0].hired.dd = 20, Toys.worker[0].hired.yy = 2002);
    printf("Salary = $%.2f\n", Toys.worker[0].Salary);
    printf("Hired Date for %s is %d, %d, %d\n", Toys.worker[1].EmpName, Toys.worker[1].hired.mm = 6, Toys.worker[1].hired.dd = 12, Toys.worker[1].hired.yy = 2000);
    printf("Salary = $%.2f\n", Toys.worker[1].Salary);
}

void PrintEmp(struct Emp Names)
{
    printf("Employee name is %s\n", Names.EmpName);
    printf("Date Hired is %d, %d, %d\n", Names.hired.mm, Names.hired.dd, Names.hired.yy);
    printf("Salary is $%.2f\n", Names.Salary);
}
```



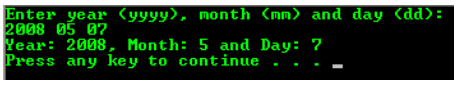
```
Employee name is Roger
Date Hired is 0, 0, 0
Salary is $0.00

Employee name is Mohave
Date Hired is 4, 20, 2002
Salary is $10000.00

Employee name is Hunter
Date Hired is 6, 12, 2000
Salary is $8000.00

Roger is a manager
Company profit is 80000.00
Hired Date for Mohave is 4, 20, 2002
Salary = $10000.00
Hired Date for Hunter is 6, 12, 2000
Salary = $8000.00
Press any key to continue . . .
```

Do A Structure Programming

Use the same structure definitions as given in the question section above.

1. Write a function called GetDate() that will receive no arguments and ask for the date in numerical format. Then return the date as struct Date.

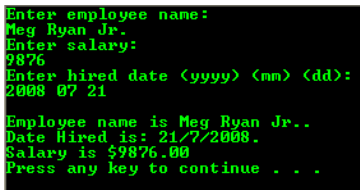
```
#include <stdio.h>

struct Date GetDate();

struct Date
{
    int yy, mm, dd;
};

void main(void)
{
    struct Date newDate;

    newDate = GetDate();
    printf("Year: %d, Month: %d and Day: %d\n", newDate.yy, newDate.mm, newDate.dd);
}

struct Date GetDate()
{
    struct Date myDate;
    puts("Enter year (yyyy), month (mm) and day (dd):");
    scanf_s("%d%d%d", &myDate.yy, &myDate.mm, &myDate.dd, 4, 2, 2);

    return myDate;
}
```



```
Enter year (yyyy), month (mm) and day (dd):
2008 05 07
Year: 2008, Month: 5 and Day: 7
Press any key to continue . . .
```

2. Write a function called GetData(). This function will not receive any arguments, but it will ask for the employee's name, salary and date of hire. Then it will return this data as a structure of type struct Emp.

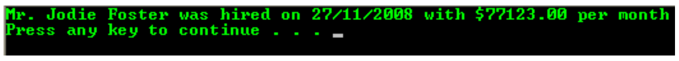
```
#include <stdio.h>

struct Emp GetData();

struct Date
{
    int yy, mm, dd;
};

struct Emp
{
    char EmpName[25];
    float Salary;
    struct Date hired;
};

struct Dep
{
    struct Emp manager;
    struct Emp worker[25];
    float Profits;
};

void main(void)
{
    struct Emp myEmp;

    myEmp = GetData();
    printf("\nEmployee name is %s.\nDate Hired is: %d/%d/%d.\nSalary is $%.2f\n", myEmp.
EmpName, myEmp.hired.dd, myEmp.hired.mm, myEmp.hired.yy, myEmp.Salary);
}

struct Emp GetData()
{
    struct Emp askEmp;

    // prompt user for inputs and store them in structure askEmp
    puts("Enter employee name:");
    gets_s(askEmp.EmpName, 25);
    puts("Enter salary: ");
    scanf_s("%f", &askEmp.Salary, sizeof(float));
    puts("Enter hired date (yyyy) (mm) (dd):");
    scanf_s("%d%d%d", &askEmp.hired.yy, &askEmp.hired.mm, &askEmp.hired.dd, 1,1,1);
    // return a structure with the stored data
    return askEmp;
}
```



```
Enter employee name:
Meg Ryan Jr.
Enter salary:
9876
Enter hired date (yyyy) (mm) (dd):
2008 07 21

Employee name is Meg Ryan Jr..
Date Hired is: 21/7/2008.
Salary is $9876.00
Press any key to continue . . .
```

3. Write a function called PrintData(). This function will receive one variable of type struct Emp and will print all three members of it on one line. Nothing will be returned.

```
#include <stdio.h>

void PrintData(struct Emp);

struct Date
{
    int yy, mm, dd;
};

struct Emp
{
    char EmpName[25];
    float Salary;
    struct Date hired;
};

struct Dep
{
    struct Emp manager;
    struct Emp worker[25];
    float Profits;
};

void main(void)
{
    struct Emp myEmp = {"Jodie Foster"};
    // fill in the myEmp structure with data
    myEmp.hired.dd = 27;
    myEmp.hired.mm = 11;
    myEmp.hired.yy = 2008;
    myEmp.Salary = 77123;

    // send the structure to PrintData()
    PrintData(myEmp);
}

void PrintData(struct Emp myData)
{
    // print all the data from the copy, copied into myData
    printf("Mr. %s was hired on %d/%d/%d with $%.2f per month\n", myData.EmpName, myData.
hired.dd, myData.hired.mm,myData.hired.yy,myData.Salary);
}
```



```
Mr. Jodie Foster was hired on 27/11/2008 with $77123.00 per month
Press any key to continue . . .
```