To:
Tenouk

C LAB WORKSHEET 14a_1
C Structures, struct Part 3

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

4. Write main(), which will call GetData() and store the structure in a local variable. Passing this local variable, main() will call GetData() and have this data printed.

```
#include <stdio.h>

void GetData(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 LocalVar = {"Jane Seymour"};

   // fill in the myEmp structure with data
   LocalVar.hired.dd = 12;
   LocalVar.hired.mm = 10;
   LocalVar.hired.yy = 2007;
   LocalVar.Salary = 99999;

   // send the structure to PrintData()
   GetData(LocalVar);
}

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

```
Mrs. Jane Seymour was hired on 12/10/2007
with $99999.00 salary per month.
Press any key to continue . . .
```

5. Write main(), which will define an array called a[5] of type struct Emp. main() will then call GetData() in a loop 5 times, each time getting the data in the proper slot of the array. Then main() will call PrintData() in a loop so that the data is printed in reverse order.

```
#include <stdio.h>

// function prototypes
struct Emp GetData();
void PrintData(struct Emp);

// structure definition
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)
{
   // declare an array of structure variable
   struct Emp a[5];
```

```
    // normal variable, a counter
    int i;
    // send the structure to GetData(), to get data from user
    for(i=0;i<=4;i++)
        a[i] = GetData();
    printf("\n");
    // print all the data stored in the structure reversely
    for(i=4;i>=0;i--)
        PrintData(a[i]);
}

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

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);
}
```

```
Enter employee name: Melinda
Enter salary: 45300
Enter hired date (dd) (mm) (yyyy): 21 03 2007
Enter employee name: Stephanie
Enter salary: 55010
Enter hired date (dd) (mm) (yyyy): 09 07 2006
Enter employee name: Obama
Enter salary: 89010
Enter hired date (dd) (mm) (yyyy): 17 11 2008
Enter employee name: Mikhail
Enter salary: 67500
Enter hired date (dd) (mm) (yyyy): 19 09 2005
Enter employee name: Kikuyo
Enter salary: 24000
Enter hired date (dd) (mm) (yyyy): 04 05 2008

Mr. Kikuyo was hired on 4/5/2008 with $24000.00 per month
Mr. Mikhail was hired on 19/9/2005 with $67500.00 per month
Mr. Obama was hired on 17/11/2008 with $89010.00 per month
Mr. Stephanie was hired on 9/7/2006 with $55010.00 per month
Mr. Melinda was hired on 21/3/2007 with $45300.00 per month
Press any key to continue . . .
```

Can you correct the employee's title using if-else statement?

6. Write a function that will receive one array of type struct Emp and an integer that gives the size of the array. Remember that, if the size is 5, then the largest index of the array should be 4. It will then return those employees' average salary.

```
#include <stdio.h>
#define SIZE 3

// function prototypes
struct Emp GetData();
double GetAverage(struct Emp [], int);

// structure definition
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)
{
    // declare an array of structure variable
    struct Emp a[SIZE];
    // normal variable, a counter
    int i;
    // hold the returned average value
    double Average;
    // send the structure to GetData()
    // to get data from user
    for(i=0;i<SIZE;i++)
    a[i] = GetData();
     printf("\n");
    // get the salary average
    Average = GetAverage(a,SIZE);
    printf("The average salary is = %.2f\n",Average);
}

struct Emp GetData()
```

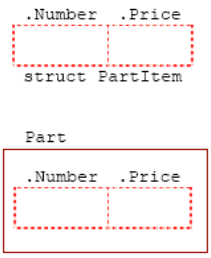
```
{
   struct Emp askEmp;
   // prompt user for inputs
   // and store them in structure askEmp
   printf("Enter employee name: ");
   scanf_s(" %s", &askEmp.EmpName, 25);
   printf("Enter salary: ");
   scanf_s("%f", &askEmp.Salary, sizeof(float));
   printf("Enter hired date (dd) (mm) (yyyy): ");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

double GetAverage(struct Emp myData[], int arrSize)
{
   int i;
   double sum = 0.00, Avg =0.00;

   // sum up all the salary values
   for(i=0;i<arrSize;i++)
      sum = sum + myData[i].Salary;

   // find the average
   Avg = sum/arrSize;
   // return the average
   return Avg;
}
```

```
Enter employee name: Kovinsky
Enter salary: 20000
Enter hired date (dd) (mm) (yyyy): 12 10 2007
Enter employee name: Catherine
Enter salary: 10000
Enter hired date (dd) (mm) (yyyy): 23 09 2006
Enter employee name: Jennifer
Enter salary: 30000
Enter hired date (dd) (mm) (yyyy): 25 12 2008

The average salary is = 20000.00
Press any key to continue . . .
```

More Structure Exercises

1. PartItem is a structure template made up of two members. Part is the variable of type struct PartItem. To access the part number, we use Part.Number. How would you set the price of part as 0.35?

```
struct PartItem
{
  char Number[10];
  float Price;
};

struct PartItem Part;
```



1. Part.Price = 0.35;

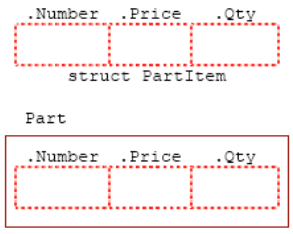
2. Use exercise 1 to answer the following questions.

   a. Name the variable.
   b. Name the structure template.
   c. What are the parts of a structure called?
   d. An array uses a set of brackets to specify the index. What do structures use to specify the item inside the structure?
   e. Once a structure template and its variables are defined, which name will be used less in the rest of the program?

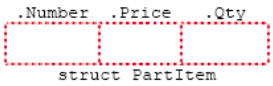
2. Answer...

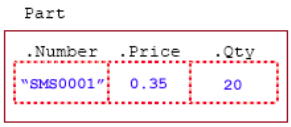
   a. The variable name is Part.
   b. The template name is struct PartItem.
   c. Parts of a structure are called members or member variables.
   d. Structures use the period (dot) to separate the structure variable and its member.
   e. The name of the structure template is used less in a program. In this case, Part would be used more than the PartItem or struct PartItem.

3. Add a third member to the structure template called Qty, which is an integer. It doesn't matter if it is the first, second or third member of the structure. Then set the part number to be "SMS0001", the price to be 0.35 and the quantity to be 20. Show the complete structure template and variable definitions and then fill up the following diagram.




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

// structure definition
struct PartItem
{
   char Number[10];
   float Price;
   int Qty;
};

void main(void)
{
   struct PartItem Part;
   // strcpy(Part.Number, "SM0001");
   strcpy_s(Part.Number, 10, "SM0001");
   Part.Price = 0.35F;
   Part.Qty = 20;
}
```





4. Remember that when we initialized **arrays** while defining them, we just placed the items for the array in order, enclosed in a set of braces. With **structures**, it works the same way. Try to show how to initialize the values of Part the same way.

```
struct PartItem Part = {"SMS0001", 0.35, 20};
```

5. To read in a string, we can use the scanf()/scanf_s() function. We can also use gets()/gets_s() function. gets_s(x); is the statement that reads a string into the character string variable called x[ ]. Define another variable of type struct PartItem and call it OldPart. Give the gets_s() and two scanf_s() calls that will read the values into OldPart. Your interactive screen should look like the following.

```
------Output---------
What is the part number? OLS001
The price? 0.80
And the quantity? 30
```

```
#include <stdio.h>

// structure definition
struct PartItem
{
   char Number[10];
   float Price;
   int Qty;
};

void main(void)
{
   struct PartItem OldPart;

   printf("What is the part number? ");
   // gets(OldPart.Number);
   gets_s(OldPart.Number,10);
   printf("The price? ");
   // scanf("%f", &OldPart.Price);
   scanf_s("%f", &OldPart.Price, sizeof(float));
   printf("And the quantity? ");
   // scanf("%d", &OldPart.Qty);
   scanf_s("%d", &OldPart.Qty, sizeof(int));
}
```

```
What is the part number? OLS0001
The price? 0.80
And the quantity? 30
Press any key to continue . . .
```

6. Next, code a program. Have the code simply print the total inventory value of each part. Your output should look like the following.

```
------Output-------
SMS0001    0.35  20    7.00
OLS0001     0,80  30    24.00
```

```
#include <stdio.h>

// structure definition
struct PartItem
{
   char Number[10];
   float Price;
   int Qty;
};

void main(void)
{
   struct PartItem OldPart, Part = {"SMS0001", 0.35f, 20};

   printf("What is the part number? ");
   // gets(OldPart.Number);
   gets_s(OldPart.Number,10);
   printf("The price? ");
   // scanf("%f", &OldPart.Price);
   scanf_s("%f", &OldPart.Price, sizeof(float));
   printf("And the quantity? ");
   // scanf("%d", &OldPart.Qty);
   scanf_s("%d", &OldPart.Qty, sizeof(int));

   printf("%s\t%.2f\t%d\t%.2f\n", Part.Number, Part.Price, Part.Qty, Part.Price*Part.Qty);
   printf("%s\t%.2f\t%d\t%.2f\n", OldPart.Number, OldPart.Price, OldPart.Qty, OldPart.
Price*OldPart.Qty);
}
```

```
What is the part number? OLS0001
The price? 0.80
And the quantity? 30
SMS0001 0.35    20      7.00
OLS0001 0.80    30      24.00
Press any key to continue . . .
```

7. Suppose we had 100 parts. Coding them would be very inefficient. Write a function called PrintInv() that will receive a structure variable called Prt of type struct PartItem and print the appropriate report line as shown in exercise 6. The function doesn't return anything.

```
#include <stdio.h>

void PrintInv(struct PartItem);

// structure definition
struct PartItem
{
   char Number[10];
   float Price;
   int Qty;
};

void main(void)
{
```

```
    struct PartItem OldPart, Part = {"SMS0001", 0.35f, 20};

    printf("What is the part number? ");
    // gets(OldPart.Number);
    gets_s(OldPart.Number,10);
    printf("The price? ");
    // scanf("%f", &OldPart.Price);
    scanf_s("%f", &OldPart.Price, sizeof(float));
    printf("And the quantity? ");
    // scanf("%d", &OldPart.Qty);
    scanf_s("%d", &OldPart.Qty, sizeof(int));

    PrintInv(Part);
    PrintInv(OldPart);
}

void PrintInv(struct PartItem Prt)
{
    printf("%s\t%.2f\t%d\t%.2f\n", Prt.Number, Prt.Price, Prt.Qty, Prt.Price*Prt.Qty);
}
```

```
What is the part number? OLS0001
The price? 0.80
And the quantity? 30
SMS0001 0.35    20      7.00
OLS0001 0.80    30      24.00
Press any key to continue . . .
```

8. Now add a local variable in PrintInv() called Inventory. Have PrintInv() pass Prt to a new function called FindInv(), which will simply find and return the product of Price and Qty. In FindInv(), no new variable is needed. PrintInv() should give the same output as before, but now it uses FindInv() to find the product. Show the coding for both functions.

```
#include <stdio.h>

void PrintInv(struct PartItem);
float FindInv(struct PartItem);

// structure definition
struct PartItem
{
    char Number[10];
    float Price;
    int Qty;
};

void main(void)
{
    struct PartItem OldPart, Part = {"SMS0001", 0.35f, 20};

    printf("What is the part number? ");
    // gets(OldPart.Number);
    gets_s(OldPart.Number,10);
    printf("The price? ");
    // scanf("%f", &OldPart.Price);
    scanf_s("%f", &OldPart.Price, sizeof(float));
    printf("And the quantity? ");
    // scanf("%d", &OldPart.Qty);
    scanf_s("%d", &OldPart.Qty, sizeof(int));

    PrintInv(Part);
    PrintInv(OldPart);
}

void PrintInv(struct PartItem Prt)
{
    float Inventory;

    Inventory = FindInv(Prt);
    printf("%s\t%.2f\t%d\t%.2f\n", Prt.Number, Prt.Price, Prt.Qty, Inventory);
}

float FindInv(struct PartItem AnotherPrt)
{
    return (AnotherPrt.Price*AnotherPrt.Qty);
}
```

```
What is the part number? OLS0001
The price? 0.80
And the quantity? 30
SMS0001 0.35    20      7.00
OLS0001 0.80    30      24.00
Press any key to continue . . .
```

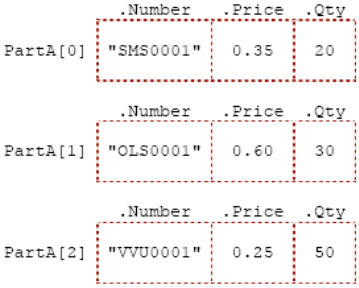
9. Now rewrite only PrintInv() and the changes in main(). PrintInv() will also return Inventory to main() and main() will add the inventories from the two parts and print their sum. Use the variable Sum in main().

```
#include <stdio.h>

float PrintInv(struct PartItem);
float FindInv(struct PartItem);

// structure definition
struct PartItem
{
    char Number[10];
    float Price;
    int Qty;
};

void main(void)
{
    struct PartItem OldPart, Part = {"SMS0001", 0.35f, 20};
    float Sum = 0.0;

    printf("What is the part number? ");
    // gets(OldPart.Number);
```

```
    gets_s(OldPart.Number,10);
    printf("The price? ");
    // scanf("%f", &OldPart.Price);
    scanf_s("%f", &OldPart.Price, sizeof(float));
    printf("And the quantity? ");
    // scanf("%d", &OldPart.Qty);
    scanf_s("%d", &OldPart.Qty, sizeof(int));

    Sum = PrintInv(Part);
    Sum = Sum + PrintInv(OldPart);

    printf("The total inventory is %.2f\n", Sum);
}

float PrintInv(struct PartItem Prt)
{
    float Inventory;

    Inventory = FindInv(Prt);
    printf("%s\t%.2f\t%d\t%.2f\n", Prt.Number, Prt.Price, Prt.Qty, Inventory);
    return Inventory;
}

float FindInv(struct PartItem AnotherPrt)
{
    return (AnotherPrt.Price*AnotherPrt.Qty);
}
```

```
What is the part number? OLS0001
The price? 0.80
And the quantity? 30
SMS0001 0.35    20      7.00
OLS0001 0.80    30      24.00
The total inventory is 31.00
Press any key to continue . . .
```

10. We can create an array of structures. Define a structure called PartA[3] that is an array of struct PartItem. Initialize the array so that the members in each cell are as shown in the accompanying Figure. When you do that, a nested pair of braces should be used for each array slot and an outer pair of braces should be used for the entire array.




```
struct PartItem PartA[3] = { {"SMS0001", 0.35, 20}, {"OLS0001", 0.60, 30}, {"VVU0001", 0.25, 50} };
```

11. For the array above, PartA[2].Qty is equal to 50. First, find the name of the variable, PartA[ ]; then give its index, PartA[2]; then provide the name of the structure member, PartA[2].Qty. Now write a for loop to add all the prices in the array.

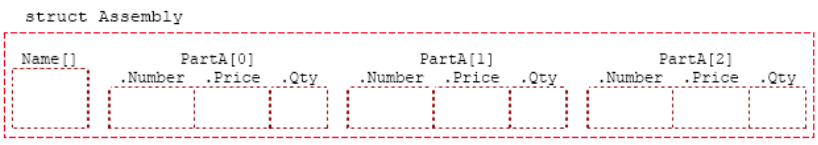
```
int i;
float Sum = 0.0f;
   for(i=0; i <= 2; ++i)
Sum = Sum + PartA[i].Price;
```

12. Write a for loop that sets all the quantities of each array slot to zero.

```
for(i = 0; i <= 2; i++)
   PartA[i].Qty = 0;
```

13. Let us define a new structure by using a pre-existing structure. For example, define a structure called struct Assembly, which is made up of the parts shown in the following diagram. The name of the Assembly has 10 characters and uses a 3-element array of struct PartItem.

```
struct PartItem
{
   char Number[10];
   float Price;
   int Qty;
};
struct Assembly
{
   char Name[10];
   struct PartItem PartA[3];
};
```



14. Define a structure variable called Motor. Assign its part number as “SMR0001”, its price as 3.57 and the quantity needed as 7. Then read in the three members for the second part. Do not assign any values to the third part.

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

// structure definitions
struct PartItem
{
    char Number[10];
    float Price;
    int Qty;
};

struct Assembly
{
    char Name[10];
    struct PartItem PartA[3];
};

void main(void)
{
    struct Assembly Motor;

    // strcpy(Motor.PartA[0].Number, "SMR0001");
    strcpy_s(Motor.PartA[0].Number, 10, "SMR0001");
    Motor.PartA[0].Price = 3.57f;
    Motor.PartA[0].Qty = 7;

    printf("Enter a part's name, price and quantity\n");

    // scanf("%s", &Motor.Name);
    scanf_s("%s", &Motor.Name, sizeof(Motor.Name));
    // scanf("%f", &Motor.PartA[1].Price);
    scanf_s("%f", &Motor.PartA[1].Price, sizeof(Motor.PartA[1].Price));
    // scanf("%d", &Motor.PartA[1].Qty);
    scanf_s("%d", &Motor.PartA[1].Qty, sizeof(Motor.PartA[1].Qty));
}
```

```
Enter a part's name, price and quantity
VMV0001 0.70 200
Press any key to continue . . .
```