



Object Oriented Programming (CS02203)

Lab Report

Name: Noor Baho
Registration #: CSU-F12-116
Lab Report #: 03
Dated: 29-10-2018
Submitted To: Mr. Usman Ahmed

The University of Lahore, Islamabad Campus
Department of Computer Science & Information Technology

Experiment # 3

Data Structures in C++

Objective

The objective of this lab session is to understand structures, nested structures and how to use structures in functions.

Software Tool

1. Windows 10
2. DEV C++
3. C++

1 Theory

As we know that Array is collection of the elements of same type, but many time we have to store the elements of the different data types. Structure is composition of the different variables of different data types, grouped under same name.

Example

```
struct {  
    char name[68];  
    char course[128];  
    int age;  
    int year;  
} student;
```

Definitions of Structures

Each member declared in Structure is called member.

```
char name[68];  
char course[128];  
int age;  
int year;
```

Are some examples of members defined above? Name given to structure is called as tag student.

Structure Initialization

When we declare a structure then memory won't be allocated for the structure. i.e only writing below declaration statement will never allocate memory.

```
struct student
{
int mark1;
int mark2;
int mark3;
};
```

We need to initialize structure variable to allocate some memory to the structure.

```
struct student s1 = {89,54,65};
```

We can initialize structure variable in different ways

- **Declare and Initialize :**

```
struct student
{
char name[20];
int roll;
float marks;
}std1 = { "Pritesh",67,78.3 };
```

In the above code snippet, the structure is declared and as soon as after declaration we have initialized the structure variable.

```
std1 = {"Pritesh",67,78.3};
```

This is the code for initializing structure variable.

- **Declaring and Initializing Multiple Variables :**

```
struct student
{
```

Data Type	Default value if not initialized
Integer	0
Float	0.00
Char	NULL

```
char name[20];
int roll;
float marks;
}
std1 = {"Pritesh",67,78.3};
std2 = {"Don",62,71.3};
```

In this example, we have declared two structure variables in above code. After declaration of variable we have initialized two variables.

```
std1 = {"Pritesh",67,78.3};
std2 = {"Don",62,71.3};
```

- **Initializing Single Member :**

```
struct student
{
int mark1;
int mark2;
int mark3;
} sub1={67};
```

Though there are three members of structure, only one is initialized, and then remaining two members are initialized with Zero. If there are variables of other data type then their initial values will be

- **Initializing inside main :**

```
struct student
{
int mark1;
int mark2;
int mark3;
};
void main()
```

```

{
    struct student s1 = {89,54,65};
    _ _ _ _ _
};

```

Passing Structure to Function

It can be done in 3 ways.

- Passing structure to a function by value .
- Passing structure to a function by address(reference) .
- No need to pass a structure – Declare structure variable as global .

Passing structure to function value

```

struct student
{
    int id;
    char name[20];
    float percentage;
};
void func(struct student record);
int main()
{
    struct student record;
    record.id=1;
    strcpy(record.name, "Raju");
    record.percentage = 86.5;
    func(record);
    return 0;
}
void func(struct student record)
{
    cout<<" Id is: <<record.id<<endl;
    cout<<" Name is: "<<record.name<<endl;
}

```

```
cout<<" Percentage is: "<<record.percentage<<endl;
}
```

Passing structure to function by address

```
struct student
{
int id;
char name[20];
float percentage;
};
void func(struct student *record);
int main()
{
struct student record;
record.id=1;
strcpy(record.name, "Raju");
record.percentage = 86.5;
func(&record);
return 0;
}
void func(struct student *record)
{
cout<<" Id is: " <<record->id;
cout<<" Name is: "<< record->name;
cout<<" Percentage is: "<<record->percentage;
}
```

Declare a structure variable as global

```
struct student
{
int id;
char name[20];
float percentage;
};
struct student record; // Global declaration of structure
void structure_demo();
int main()
```

```

{
record.id=1;
strcpy(record.name, "Raju");
record.percentage = 86.5;
structure_demo();
return 0;
}
void structure_demo()
{
cout<<" Id is: "<< record.id;
cout<<" Name is: "<< record.name;
cout<<" Percentage is: "<< record.percentage;
}

```

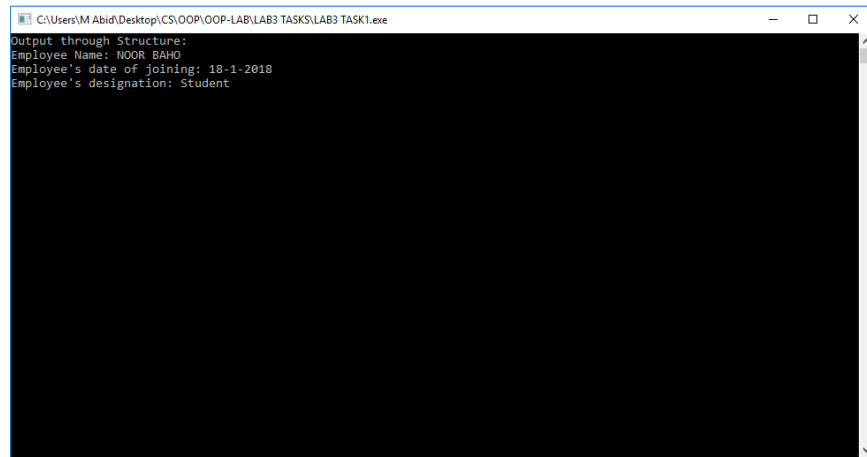


Figure 1: Record of Employee

2 Task

2.1 Procedure: Task 1

Define a Structure of Employee with following data (Employee-Name, Date-of-joining, and Designation).

```
\textbf{Solution: }\\
#include <iostream>
#include <conio.h>
#include <cstring>
//task1
using namespace std;
struct EmpRecord{
    char employee_name[10];
    int date, month, year;
    char designation[10];
};
int main()
{
    EmpRecord emp;
    strcpy(emp.employee_name,"NOOR BAHO");
```

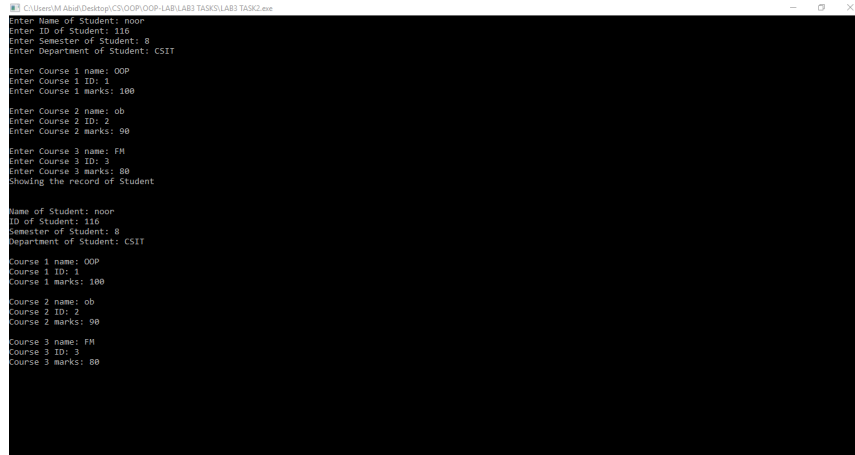


```

    emp.date=18;
    emp.month=01;
    emp.year=2018;
    strcpy(emp.designation,"Student");
    cout<<"Output through Structure:"<<endl;
    cout<<"Employee Name: "<<emp.employee_name;
    cout<<endl<<"Employee's date of joining: "<<emp.date<<"-"<<
    emp.month<<"-"<<emp.year;
    cout<<endl<<"Employee's designation: "<<emp.designation;
    getch();
    return 0;
}

```

Explanation: Structures are required to be made of given fields; Employee name, Date of Joining and Designation of Employee. So we defined a structure `EmpRecord` with members: `char` type array `"employee_name[10]"`, `int` type `"date"`, `"month"` and `"year"` and a `char` type `"designation"`. Then we declared `"emp"` a constructor in `main()` through which we can access the members of structure. Then we put record to the members of structure accessing them via constructor and shown through `cout` to the user.



```
C:\Users\TM\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.IE5\...
Enter Name of Student: noor
Enter ID of Student: 116
Enter Semester of Student: 8
Enter Department of Student: CSIT

Enter Course 1 name: OOP
Enter Course 1 ID: 1
Enter Course 1 marks: 100

Enter Course 2 name: ob
Enter Course 2 ID: 2
Enter Course 2 marks: 90

Enter Course 3 name: FM
Enter Course 3 ID: 3
Enter Course 3 marks: 80
Showing the record of Student

Name of Student: noor
ID of Student: 116
Semester of Student: 8
Department of Student: CSIT

Course 1 name: OOP
Course 1 ID: 1
Course 1 marks: 100

Course 2 name: ob
Course 2 ID: 2
Course 2 marks: 90

Course 3 name: FM
Course 3 ID: 3
Course 3 marks: 80
```

Figure 2: Grading System of Student

2.2 Procedure: Task 2

We are developing a grading system for students which have the following information. Student-Id, Student-Name, Department, Semester, Courses and Marks in Courses. A course has following information (Course-ID, Course-Name, Marks). 2a) Define a function that takes input from user to set student information. 2b) Define a function that displays the output of student information.

Solution:

```
#include <iostream>
#include <conio.h>
#include <cstring>
//grading system for student simple input and print
using namespace std;
struct Student{
    int std_id;
    char std_name[10];
    char department[10];
    int semester;
};
struct Course{
    char course_name[10];
```

```

        int course_id;
        int course_mark;
    };
    Student stdn;
    Course crs1;
    Course crs2;
    Course crs3;
    void inputstd()
    {
        cout<<"Enter Name of Student: ";
        cin>>stdn.std_name;
        cout<<"Enter ID of Student: ";
        cin>>stdn.std_id;
        cout<<"Enter Semester of Student: ";
        cin>>stdn.semester;
        cout<<"Enter Department of Student: ";
        cin>>stdn.department;

        cout<<endl<<"Enter Course 1 name: ";
        cin>>crs1.course_name;
        cout<<"Enter Course 1 ID: ";
        cin>>crs1.course_id;
        cout<<"Enter Course 1 marks: ";
        cin>>crs1.course_mark;

        cout<<endl<<"Enter Course 2 name: ";
        cin>>crs2.course_name;
        cout<<"Enter Course 2 ID: ";
        cin>>crs2.course_id;
        cout<<"Enter Course 2 marks: ";
        cin>>crs2.course_mark;

        cout<<endl<<"Enter Course 3 name: ";
        cin>>crs3.course_name;
        cout<<"Enter Course 3 ID: ";
        cin>>crs3.course_id;
        cout<<"Enter Course 3 marks: ";
        cin>>crs3.course_mark;
    }

```

```

    }
    void outputstd()
    {
        cout<<"Showing the record of Student"<<endl;
        cout<<endl<<endl<<"Name of Student: "<<stdn.std_name;
        cout<<endl<<"ID of Student: "<<stdn.std_id;
        cout<<endl<<"Semester of Student: "<<stdn.semester;
        cout<<endl<<"Department of Student: "<<stdn.department;

        cout<<endl<<endl<<"Course 1 name: "<<crs1.course_name;
        cout<<endl<<"Course 1 ID: "<<crs1.course_id;
        cout<<endl<<"Course 1 marks: "<<crs1.course_mark;

        cout<<endl<<endl<<"Course 2 name: "<<crs2.course_name;
        cout<<endl<<"Course 2 ID: "<<crs2.course_id;
        cout<<endl<<"Course 2 marks: "<<crs2.course_mark;

        cout<<endl<<endl<<"Course 3 name: "<<crs3.course_name;
        cout<<endl<<"Course 3 ID: "<<crs3.course_id;
        cout<<endl<<"Course 3 marks: "<<crs3.course_mark;
    }
}

int main()
{
    inputstd();
    outputstd();
    getch();
    return 0;
}

```

Explanation: In addition to previous task, here is asked not only to initialize a structure but also to take random input from user and save that record to the members of structure accordingly. As from statement it's clear there will be two structures; one for Student's information and one for the courses and marks. So members are simply defined in the statement so we created two structures where course structure will contain the members for a course. Now we will create 4 global constructors as we are required to input and output through functions, so if constructors will be created in the scope of main() then they will not be responding to these functions defined out of main or we should have to pass structure to the functions;

one for Student record and reamaining three for three subjects relatively. So through inputstd() functions we will get input from user to all members of structures and also output the values to the users on his console screen through another function outputstd()).

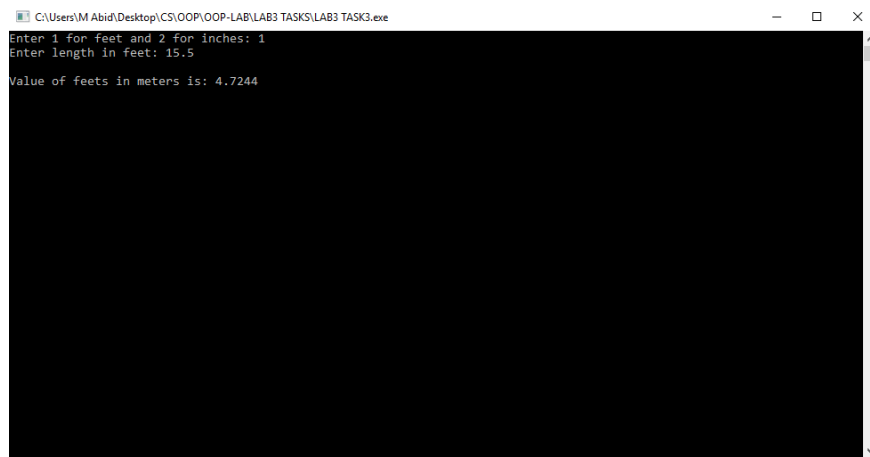


Figure 3: Feet to Meter Converter

2.3 Procedure: Task 3

Write a C++ program to convert length from feet and inches to meters by Using structure for length in feet and inches. Ask user for input feet and inches and display output in meters.

Solution:

```
#include<iostream>
#include<conio.h>
#include<cstring>
//converter feet inches to meters
using namespace std;

struct parameter{
    float feet;
    float inch;
    float meter;
};

int main()
{
    int choice;
    parameter obj;
```

```

        cout<<"Enter 1 for feet and 2 for inches: ";
        cin>>choice;

        switch(choice)
        {
            case 1:
            {
                cout<<"Enter length in feet: ";
                cin>>obj.feet;
                obj.meter= obj.feet*0.3048;
                cout<<endl<<"Value of feets in meters is: ";
                <<obj.meter;
                break;
            }

            case 2:
            {
                cout<<"Enter length in inch: ";
                cin>>obj.inch;
                obj.meter= obj.inch*0.0254;
                cout<<endl<<"Value of inches in meters is: ";
                <<obj.meter;
                break;
            }
        }

        getch();
        return 0;
    }

```

Explanation: An converter is required that converts length from feet to meters or inches to meters through structures. So we defined structure of parameters with the members feet, inch and meter. Defined a constructor obj for this structure in main(). Used switch statement on a variable "choice" to ask user; he wants to convert feets into meters or inches into meters. Then control to the relevant case, if feet to meters then case 1 will operate and get value from user in feets and will convert through a simple formula and show him the result in meters. Whereas if he wants to convert inches to meters then case 2 will proceed and will take input in inches to the

inch member of structure and will calculate it into meters through a simple formula and display the result in meters.

$$\begin{array}{ll} 1 \text{ meter} = 39.37 \text{ inches} & \Rightarrow 1 \text{ inch} = 1/39.37 \text{ meter} \\ \Rightarrow 1 \text{ inch} = 0.0254 \text{ meter} & \\ 1 \text{ meter} = 3.28 \text{ feet} & \Rightarrow 1 \text{ foot} = 1/3.28 \text{ meter} \\ \Rightarrow 1 \text{ foot} = 0.3048 \text{ meter} & \end{array}$$


```
Enter Record of 1st Employee
Enter Name of Employee: MOON
Enter ID of Employee: 1
Enter Designation of Employee: MANAGER

Enter Record of 2nd Employee
Enter Name of Employee: SAAD
Enter ID of Employee: 2
Enter Designation of Employee: ASSISTANT

Enter Record of 3rd Employee
Enter Name of Employee: ADEEL
Enter ID of Employee: 3
Enter Designation of Employee: OPERATOR

Enter Record of 4th Employee
Enter Name of Employee: MANSOOR
Enter ID of Employee: 4
Enter Designation of Employee: ASSISTANT

Enter Record of 5th Employee
Enter Name of Employee: MDAD
Enter ID of Employee: 5
Enter Designation of Employee: INTERNEE

Showing Records of Employees
Record of 1st Employees is as below:
Name: MOON
ID: 1
Designation: MANAGER

Record of 2nd Employees is as below:
Name: SAAD
ID: 2
Designation: ASSISTANT

Record of 3rd Employees is as below:
Name: ADEEL
ID: 3
Designation: OPERATOR

Record of 4th Employees is as below:
Name: MANSOOR
ID: 4
Designation: ASSISTANT

Record of 5th Employees is as below:
Name: MDAD
ID: 5
Designation: INTERNEE
```

Figure 4: Input & Output Record of 5 Employees

2.4 Procedure: Task 4

Write a function that take employee structure as parameter and display the data of employee. Create array of 5 employees and display the data of each employee.

Solution:

```
#include<iostream>
#include<conio.h>
#include<cstring>
//task 4 enter 5 employee data
using namespace std;

struct Employee{
    char emp_name[10];
    int emp_id;
    char designation[10];
};

void inputoutput(Employee)
```

```

Employee emp[5];

//input
for(int i=0; i<5; i++)
{
    cout<<"Enter Record of Employee "<<i<<":"<<endl;
    cout<<"Enter Name of Employee: ";
    cin>>emp[i].emp_name;
    cout<<"Enter ID of Employee: ";
    cin>>emp[i].emp_id;
    cout<<"Enter Designation of Employee: ";
    cin>>emp[i].designation;

    cout<<endl<<endl;
}

cout<<endl;
//output
for(int j=0; j<5; j++)
{
    cout<<"Showing Records of Employees"<<endl;
    cout<<"Record of Employee "<<j<<" is as below: ";
    cout<<endl<<"Name: "<<emp[j].emp_name;
    cout<<endl<<"ID: "<<emp[j].emp_id;
    cout<<endl<<"Designation: "<<emp[j].designation;
    cout<<endl<<endl;
}

}

int main()
{
    Employee emp;
    inputoutput(emp);

    getch();
    return 0;
}

```

Explanation: We are required to access structure through passing it to a function. So first of all defined structure and then a function inputoutput() which deals getting enteries from user and then prints that data by accessing structure memeber through a function. In main() we will define a constructor for structure and when we call the structure we will pass constructor as argument to the function, also while defining function we will declare the argument as structure type. Then with in the function we are now able to define constructor for the structure to access members of the structure. Remaining part is same as last part, by accessing members through constructor we will store data into them and then access them to print same data to screen to be shown to user.

3 Conclusion

We learned about structures used in programming by which we declare our own desired data type. In structure we include desired heterogeneous data types and group them into one. Only disadvantage is that we can't perform functions within these structures.