



AIR UNIVERSITY, ISLAMABAD

# Department of Creative Technologies

---

FACULTY OF COMPUTING AND ARTIFICIAL INTELLIGENCE

---

CS214L – Object Oriented Programming Lab

Class: BSGM-II (A)

Lab (no): 02

Topic: Structures

Date: February 15<sup>th</sup>, 2022

Instructor: Ms. Saira Qamar

Lab Instructor: Ms. Saira Qamar

---



## Instructions:

**Submission:** Combine all your work in one .cpp file. Save that file(.cpp) for your use because when I will check your assignment, I will ask you to show me the code while Viva of the .doc that you will submit. Take screenshot of the **output** and your **written code** and **create a word document(.doc)** with the name as Lab\_NO\_DEGREE\_ROLLNUM.doc. Submit .doc file on Google Classroom within the deadline. **Failure to submit according to the above format would result in deduction of 10% marks.** Submissions on the email will not be accepted.

**Plagiarism:** Plagiarism cases will be dealt with strictly. If found plagiarized, both the involved parties will be awarded zero marks in this assignment, all of the remaining assignments, or even an F grade in the course. Copying from the internet is the easiest way to get caught!

**Deadline:** The deadline to submit the assignment is **February 18<sup>th</sup>, 2022 at 11:59 PM**. Late submission with marks deduction will be accepted according to the course policy shared earlier. Correct and timely submission of the assignment is the responsibility of every student; hence no relaxation will be given to anyone.

**Comments:** Comment your code properly. Bonus marks (maximum 10%) will be awarded to well comment code. Write your name and roll number (as a block comment) at the beginning of the solution to each problem.

**Tip:** For timely completion of the assignment, start as early as possible. Furthermore, work smartly - as some of the problems can be solved using smarter logic.

1. Note: Follow the given instructions to the letter, failing to do so will result in a zero.

## Objectives:

In this lab, you will learn:

- To use Structures.
- Structure with Arrays
- Structure with Functions
- Structure with Pointers

### 1.1 What is Structures:

"A structure is a **collection of variables** under a single name. These variables can be of different types, and each has a name that is used to select it from the structure"

### 1.2 Example of Structures:

There is always **a requirement in most of our data processing applications that the relevant data should be grouped and handled as a group**. This is what the concept of structure is.

Suppose that you want to write a program to process **student data**. A student record consists of, among other things, the student's name, student ID, GPA, courses taken, and course grades. Thus, various components are associated with a student. However, these components are all of different types. For example, the student's name is a string, and the GPA is a floating-point number.

Because these components are of different types, you cannot use an array to group all of the items associated with a student. C++ provides a structured data type called struct to group items of different types. Grouping components that are related but of different types offers several advantages. For example, a single variable can pass all the components as parameters to a function.

### 1.3 General Syntax of Structures in C++:

The **components** of a **struct** are called the **members** of the **struct**. The general **syntax** of a struct in C++ is:

```
struct structName
{
    dataType1 identifier1;
    dataType2 identifier2;
    .
    .
    .
    dataTypeN identifierN;
};
```

**Figure 2: Shows the syntax of structure C++ program**

### 1.4 Definition of Structures:

In C++, struct is a **reserved** word. The members of a struct, even though they are enclosed in braces (that is, they form a block), are not considered to form a compound statement. Thus, a semicolon (after the right brace) is essential to end the struct statement. A **semicolon** at the **end of the struct definition** is, therefore, a **part** of the **syntax**.

```
struct studentType
{
    string firstName;
    string lastName;
    char courseGrade;
    int testScore;
    int programmingScore;
    double GPA;
};
```

**Figure 3: Structure Definition of Student**

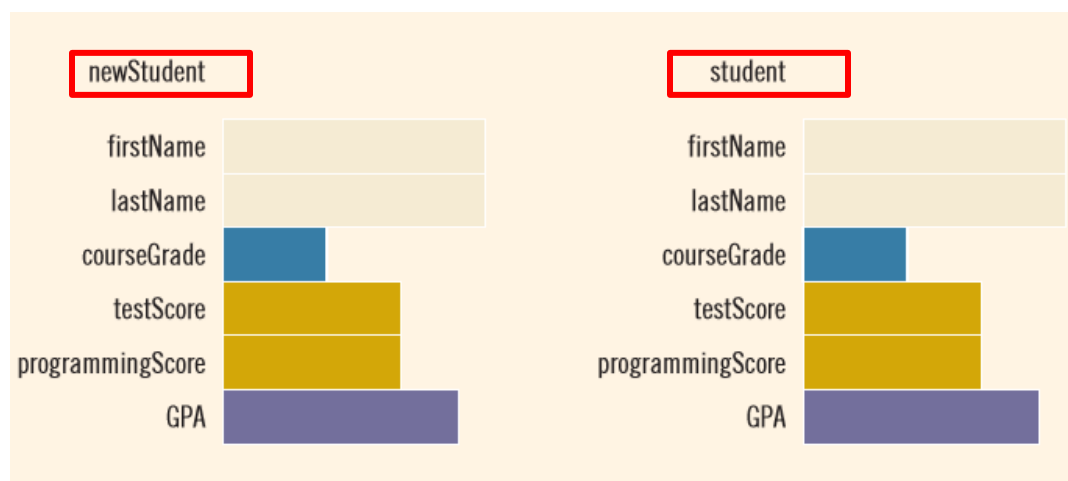
### 1.5 Declaration of Structures:

Once a data type is defined, you can declare variables of that type. As we defined a struct type, `studentType` in Section 1.5, and then declare variables of that type.

```
studentType newStudent;
studentType student;
```

**Figure 4: Structure Declaration of Student**

These statements declare two struct variables, **newStudent** and **student**, of type `studentType`. The memory allocated is large enough to store `firstName`, `lastName`, `courseGrade`, `testScore`, `programmingScore`, and `GPA`. The following Figures visualize the declared variables:



**Figure 5: Visualization of members of Structure**

### 1.6 Another way to define and declare the Structures in C++:

You can also declare struct variables when you define the struct. For example, consider the following statements:

```
struct studentType
{
    string firstName;
    string lastName;
    char courseGrade;
    int testScore;
    int programmingScore;
    double GPA;
} tempStudent;
```

Figure 6: Define and Declare Structure of Student together

These statements **define** the struct **studentType** and also **declare tempStudent** to be a variable of type **studentType**.

Typically, in a program, a struct is defined before the definitions of all of the functions in the program so that the struct can be used throughout the program. **Therefore, if you define a struct and also simultaneously declare a struct variable (as in the preceding statements), then that struct variable becomes a global variable and thus can be accessed anywhere in the program.**

### 1.7 Initializing and Accessing Structure Members:

We have so far learnt how to define a structure and declare its variables. Let's see how we can **put the values in its data members**. The following example can help us understand the phenomenon further.

#### Example:

Here is a simple example showing the initialization and displaying (accessing) the structure members.

```
#include <iostream>
using namespace std;

struct Student
{
    char name[50];
    int semester;
};

int main()
{
    // Creating and initializing Student object
    Student std={"Saima", 2};

    // Accessing members of Student object
    cout << "\nStudent Information." << endl;
    cout << "Name: " << std.name << endl;
    cout << "Semester: " << std.semester << endl;

    return 0;
}
```

In the above example, we have **declared a variable `std`** of data type **student** structure and initialize its data member. The values of **data members of `std`** are **comma separated in curly braces**. “Saima” will be assigned to *name*, “2” will be assigned to the *semester*. **So far we have not touched these data members directly**. To **access the data members of structure, dot operator (`.`) is used**. Therefore while manipulating name of `std`, we will say `std.name`. This is a **way of referring to a data member of a structure**. We can also initialize that way.

**Output will be:**

```
Student Information.
Name: Saima
Semester: 2
```

### 1.8 Array as a member of Structure

A structure may consist of different types of data. The members can be of simple type such as int, float, etc. But it can also be complex as arrays. The following example shows:

```
#include <iostream>
using namespace std;
struct Student
{
    int marks[2];
};
int main()
{
    Student std1;
    std1.marks[0]=12;
    std1.marks[1]=30;

    cout << "1st subject Marks " << std1.marks[0] << endl;
    cout << "2nd subject Marks " << std1.marks[1] << endl;

    return 0;
}
```

Output will be:

```
D:\Air University\OOP Lab\OOP Lab-2\arraysWithStructures.exe
1st subject Marks 12
2nd subject Marks 30
```

### 1.9 Structure with array

An array is a collection of **data items of the same type**. Each element of the array can be int, char, float, double, or even a **structure**. We have seen that a structure allows elements of different data types to be grouped together under a single name. This structure can then be thought of as a new data type in itself. So, **an array can comprise elements of this new data type**. An array of structures finds its applications in grouping the records together and provides for fast accessing.

	Array within a Structure	Array of Structures
<b>Basic idea</b>	A structure contains an array as its member variable	An array in which each element is of type structure
<b>Access</b>	Can be accessed using the dot operator just as we access other elements of the structure	Can be accessed by indexing just as we access an array
<b>Syntax</b>	<pre>struct class {     int ar[10]; } a1, a2, a3;</pre>	<pre>struct class {     int a, b, c; } students[10];</pre>



# Air University Islamabad

## FACULTY OF COMPUTING & ARTIFICIAL INTELLIGENCE

Department of Creative Technologies  
Object Oriented Programming Lab

```
#include <iostream>
using namespace std;
struct Student
{
    char name[50];
    int semester;
};
int main()
{
    Student std1, std2;

    //Student 1: std record input
    cout << "Enter Full name: ";
    cin.get(std1.name, 50);
    cout << "Enter semester: ";
    cin >> std1.semester;

    //Student 2: std2 record input
    cout << "Enter Full name: ";
    cin.get(std2.name, 50);
    cout << "Enter semester: ";
    cin >> std2.semester;

    cout << "\nStudent 1 Information." << endl;
    cout << "Name: " << std1.name << endl;
    cout << "Semester: " << std1.semester << endl;

    cout << "\nStudent 2 Information." << endl;
    cout << "Name: " << std2.name << endl;
    cout << "Semester: " << std2.semester << endl;

    return 0;
}
```

Output will be:

```
Enter Full name: Saima
Enter semester: 1
Enter Full name: Malik
Enter semester: 2

Student 1 Information.
Name: Saima
Semester: 1

Student 2 Information.
Name: Malik
Semester: 2
```

As you can notice, for each Student you have to create new object and feed their respective values. It is time consuming. Therefore, we can create array of Student object as we create array of some built-in data types. The following **example** shows:



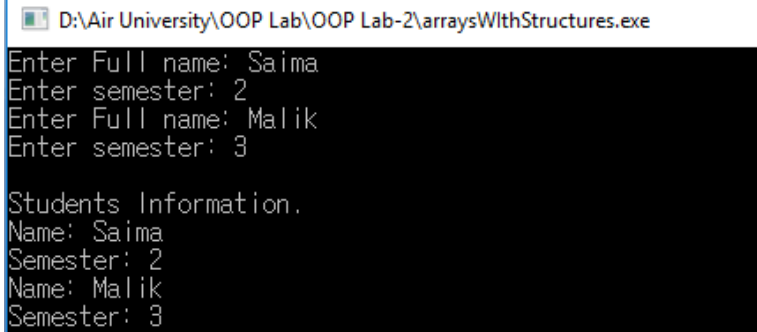
```
#include <iostream>
using namespace std;
struct Student
{
    char name[50];
    int semester;
};
int main()
{
    Student std[2];

    //Students record input
    for(int i=0; i<2; i++)
    {
        cout << "Enter Full name: ";
        cin.get(std[i].name, 50);
        cout << "Enter semester: ";
        cin >> std[i].semester;
    }

    //Students record input
    cout << "\nStudents Information." << endl;
    for(int i=0; i<2; i++)
    {
        cout << "Name: " << std[i].name << endl;
        cout << "Semester: " << std[i].semester << endl;
    }

    return 0;
}
```

Output will be:



D:\Air University\OOP Lab\OOP Lab-2\arraysWithStructures.exe

```
Enter Full name: Saima
Enter semester: 2
Enter Full name: Malik
Enter semester: 3

Students Information.
Name: Saima
Semester: 2
Name: Malik
Semester: 3
```

#### 1.10 Structure with function

Structure variables can be passed to a function and returned in a similar way as normal arguments.



**Example:**

```
#include <iostream>
using namespace std;

struct Student
{
    char name[50];
    int semester;
};

void displayData(Student); // Function declaration where you will pass Student obj

int main()
{
    Student std;

    cout << "Enter Full name: ";
    cin.get(std.name, 50);
    cout << "Enter semester: ";
    cin >> std.semester;

    // Function call with structure variable as an argument
    displayData(std);

    return 0;
}

void displayData(Student s)
{
    cout << "\nStudent Information." << endl;
    cout << "Name: " << s.name << endl;
    cout << "Semester: " << s.semester << endl;
}
```

**Output will be:**

```
D:\Air University\OOP Lab\OOP Lab-2\functionWithStructures.exe
Enter Full name: Saima
Enter semester: 2

Student Information.
Name: Saima
Semester: 2
```

From the example above, you can notice **the type of function is void**. You can **also return structure from a function**.

```
#include <iostream>
using namespace std;

struct Student
{
    char name[50];
    int semester;
};

Student getInput(Student);    // Function declaration where you take Student input
void displayData(Student);    // Function declaration where you will pass Student obj
int main()
{
    Student std;

    // Receiving Student record
    std=getInput(std);
    // Function call with structure variable as an argument
    displayData(std);

    return 0;
}

Student getInput(Student s)
{
    cout << "Enter Full name: ";
    cin.get(s.name, 50);
    cout << "Enter semester: ";
    cin >> s.semester;

    return s;
}

void displayData(Student s)
{
    cout << "\nStudent Information." << endl;
    cout << "Name: " << s.name << endl;
    cout << "Semester: " << s.semester << endl;
}
```

Output will be:

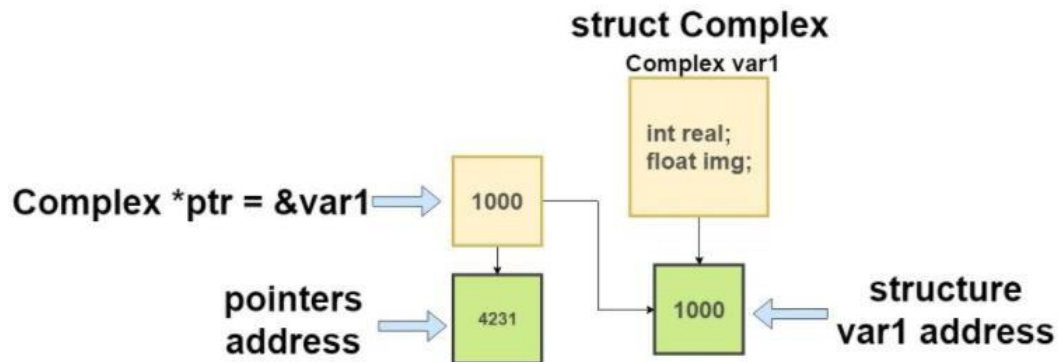
D:\Air University\OOP Lab\OOP Lab-2\functionWithStructures.exe

```
Enter Full name: Saima Majeed
Enter semester: 2

Student Information.
Name: Saima Majeed
Semester: 2
```

### 1.11 Structure with pointer

A **pointer** variable can be created not only for native types like (int, float, double etc.) but they can also be created for user defined types like **structure**.



A **structure pointer** is a type of pointer that **stores the address of a structure typed variable**. As you can see in the above diagram we have a structure named **Complex** with 2 data members (one integer type and one float type). When we **create a variable** of this structure (**Complex var1**), it is allotted a memory space. This **memory** can now be **accessed by creating a pointer of the same structure type** as shown in the diagram (**Complex \*ptr**). Now this pointer can point the actual memory address of the structure variable var1 and can access its values.

Using structure pointers, the members of structures are accessed using arrow operator `->`. Following is an example program for the same:

#### Example:

```
#include <iostream>
using namespace std;
struct Complex
{
    int real;
    float img;
};

int main()
{
    // creating a Complex structure variable
    Complex var1;

    /* creating a pointer of Complex type &
       assigning address of var1 to this pointer */
    Complex* ptr = &var1;

    /* assigning values to
       Complex variable var1 */
    var1.real = 5;
    var1.img = 0.33;

    // accessing values of var1 using pointer
    cout<<"Real part: "<<ptr->real<<endl;
    cout<<"Imaginary part: "<<ptr->img;

    return 0;
}
```



# Air University Islamabad

## FACULTY OF COMPUTING & ARTIFICIAL INTELLIGENCE

Department of Creative Technologies  
Object Oriented Programming Lab

---

### Output will be:

```
Real part: 5  
Imaginary part: 0.33
```

---

### Perform the following Tasks:

**Lab Task 1:** Write a program that will receive user data of birth e.g. year, day, and month. Display date of birth in a format as: *dd/mm/yy*. Use **functions** to get and display the date of birth.

**Lab Task 2:** Write a program that declares a structure to store id, name, and prices of a book. It defines an array of structures to store the record of three books. Input the records of three books and display the record of most costly book.

**Lab Task 3:** Perform [Section 1.11](#) example and change the struct and its members e.g. Student.

---

### Lab Task Submission Format:

- Implement all tasks in a single .cpp file and write the following things at the top of code file by using multi line comments.

```
/*  
 * @author Student Name (roll No)  
 * @date DATE  
 * @ Lab No  
 */
```

- Submit only the “.doc” files and not the whole project (**.rar or any other format will not be marked**).