



**Riphah International University**

**Islamabad**

**LAB MANUAL**

**Lab Manual for Programming Fundamentals**

**FACULTY OF COMPUTING (WISH)**

**Lab Manual for Programming Fundamentals**

**Lab 11: Functions pass by value and pass by references and Structure**

|  |  |
| --- | --- |
|  | Supervision and Coordination |

**Dr Sumera Saleem**

Lecturer

Faculty of Computing

|  |  |  |  |
| --- | --- | --- | --- |
|  | | |  |
|  | Lab Designers | | |

|  |
| --- |
| **Fareeha Ashraf**  Teaching Fellow  Faculty of Computing |

**Lab 11: Functions and Introductions of Structures**

**Lab Manual Lecture [Expected time = 30 minutes]**

**Objective:**

* Understand the concept of functions and their usage in C++.
* Learn how to use references to modify values within functions.
* Apply functions and references to perform various tasks.

1. **Function**

Function is a set of statements that take inputs, do some specific computation and produces output.

The idea is to put some commonly or repeatedly done task together and make a function, so

that instead of writing the same code again and again for different inputs, we can call the funct

1. **Pass by Value:**

How it works:

* In pass by value, a copy of the actual argument is passed to the function.
* Any changes made to the parameters within the function do not affect the original values outside the function.

**Example:**

#include <iostream>

using namespace std;

void incrementByValue(int x)

{ x++; }

int main()

{ int num = 5;

cout << "Before function call: " << num << endl;

incrementByValue(num);

cout << "After function call: " << num << endl;

return 0; }

**Output:**

Before function call: 5 After function call: 5

1. **Pass by Reference:**

How it works:

* In pass by reference, the memory address of the actual argument is passed to the function.
* Any changes made to the parameters within the function directly affect the original values outside the function.

**Example:**

#include <iostream>

using namespace std;

void incrementByReference(int &x) { x++; }

int main()

{ int num = 5;

cout << "Before function call: " << num << endl;

incrementByReference(num); cout << "After function call: " << num << endl;

return 0; }

**Output:**

Before function call: 5 After function call: 6

1. **Key Differences:**
2. **Memory Usage:**
   * Pass by value involves creating a copy of the actual argument, which may consume more memory.
   * Pass by reference directly works with the memory location of the original variable, which is more memory-efficient.
3. **Modification of Original Values:**
   * Pass by value does not modify the original values outside the function.
   * Pass by reference allows modification of the original values outside the function.
4. **Use Cases:**
   * Pass by value is suitable when you don't want the function to modify the original values.
   * Pass by reference is useful when you want the function to directly modify the original values.
5. **Syntax:**
   * Pass by value uses normal parameters in the function definition.
   * Pass by reference uses references in the function definition (denoted by **&**).

// Pass by value

void functionName(int parameter);

// Pass by reference void functionName(int &parameter);

In summary, pass by value creates a copy of the argument, while pass by reference allows direct manipulation of the original values. The choice between them depends on the specific requirements of the program. Pass by reference is often preferred when working with large data structures to avoid unnecessary copying.

**What is an object?**

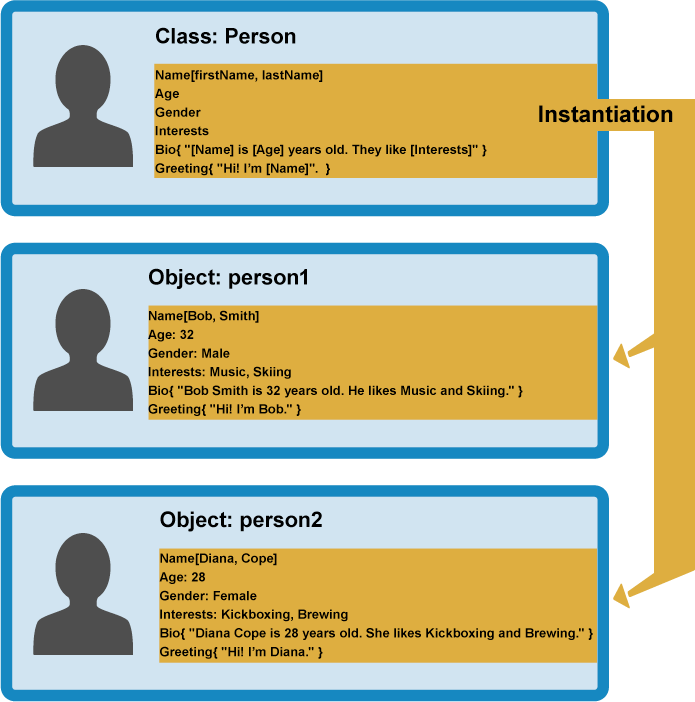
Objects have states and behaviours. Example: A dog has states - color, name, breed as well as behaviors – wagging the tail, barking, eating. An object is an instance of a class.

Software objects also have a state and a behavior. A software object's state is stored in fields and behavior is shown via methods.

For example;

**Int dollars=0;**

Here we have defined a single instance of type int and given the name dollars.Variable dollars is an object of type int.

****

**Creating our own Data Types**

In C++ we can define our own data types and specify the operations that can be applied to them. The types that we defined are referred to as user defined data types. These are structures and unions. We use keywords struct to create them.

# 

# Structure

A structure is a collection of elements of various types. A *data structure* is a group of data elements grouped together under one name. Each of the elements in a structure is called a data member. Data member can have different types and different lengths Each member is accessed through the use of the member selector operator. In C++, a structure is commonly declared according to the following syntax:

**struct TypeName**

**{**

**DataType MemberName;**

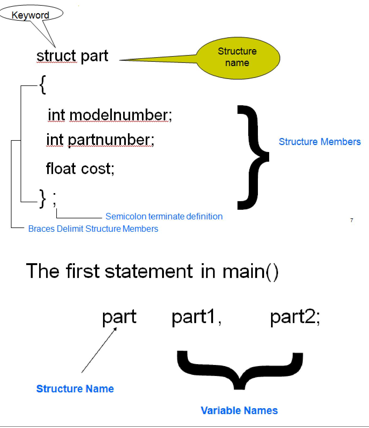
**DataType MemberName;**

**};**

Sample Code

|  |  |
| --- | --- |
| ***struct* product {**  ***int* weight;**  ***double* price;**  **} ;**  **product apple;**  **product banana, melon;** |  |

This declares a structure type, called product, and defines it having two members: weight and price, each of a different fundamental type. This declaration creates a new type (product), which is then used to declare three objects (variables) of this type: apple, banana, and melon. Note how once product is declared, it is used just like any other type.



**Example :**

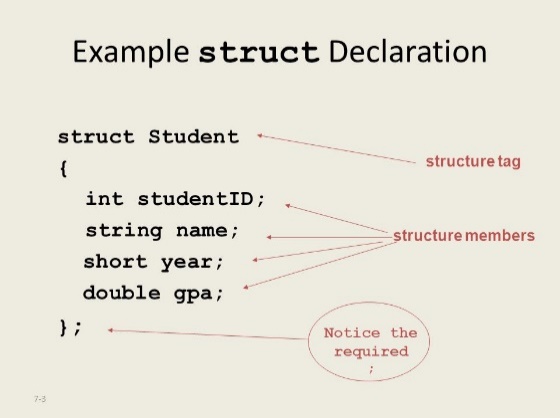
The member\_list is where you describe the types of data that are to be associated with the object that you are defining.

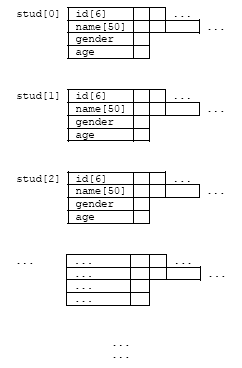
For example, the member list for Student is:

**string name;**

**int id;**

**int mark[3]**



****

This code just defines the format of the structure. In order to start using this particular structure you need to declare an instance of it. This is similar to defining a variable of a predefined type such as int.

**Declaring a Structure Instance**

To create an instance of a structure -you need to declare it, just as you would declare an instance of a primitive data type. Following the example referred to in the previous section, the statement

**Student stu;**

Declares an instance called stu of the structure called Student. Note that the general syntax would be:

**datatype variable\_name;**

Think back to how you declare an instance of an integer .

**i.e. int i; // datatype variable\_name;**

# Sample Program

**Struct person**

**{**

**Char name [20];**

**Int age,**

**Char degree [20]**

**};**

**Int `main ()**

**{**

**Person sara = {“sara khan”,20,“BSSE”};**

**Return 0;**

**}**

**Sample Program**

**//Using a BOOK Structure**

**#include <iostream>**

**using namespace std;**

**//Structure to Represent a Box**

**struct Book**

**{**

**char title[80];**

**char author[80];**

**char publisher[80];**

**int publishingYear;**

**};**

**//Prototype of function to calculate the volume of a box**

**int main()**

**{**

**Book oopBook1 = { "Beginning C++","Ivor Horton’s","Wrox",1998};**

**cout<<endl<<"Title of oopBook1 = "<<oopBook1.title<<endl**

**<<"Author of oopBook1 = "<<oopBook1.author<<endl**

**<<"Publisher of oopBook1 = "<<oopBook1.publisher<<endl**

**<<"Publishing Year of oopBook1 = "<<oopBook1.publishingYear<<endl;**

**Book oopBook2= { "OOP in C++",**

**"Robert Lafore",**

**"SAMS",**

**1998**

**};**

**oopBook2.publishingYear += 2;**

**cout<<endl**

**<<"Title of oopBook2 = "<<oopBook2.title<<endl**

**<<"Author of oopBook2 = "<<oopBook2.author<<endl**

**<<"Publisher of oopBook2 = "<<oopBook2.publisher<<endl**

**<<"Publishing Year of oopBook2 = "<<oopBook2.publishingYear<<endl;**

**return 0;**

**}**

# Evaluation criteria

The evaluation criteria for this lab will be based on the completion of the following tasks. Each task is assigned the marks percentage which will be evaluated by the instructor in the lab whether the student has finished the complete/partial task(s).

**Lab Tasks**

**Task 1:**

* Write a C++ function named **square** that takes an integer as a parameter and
* Returns the square of that number. Use pass by value.

**Task 2:**

* Implement a program that finds the maximum and minimum values in an array.
* Utilize a function with references to update maximum and minimum values.
* Display the results.

**Task 3:**

* Write a program that calculates the sum and average of elements in an array.
* Define a function that takes the array, its size, and references for sum and average.
* Display the sum and average of the array elements.

**Task 4:**

* The **Book** structure includes members for the book's title, author, year of publication, and number of pages.
* Functions are defined to input information and display information about a book.
* The **main** function creates an array of **5** **Book** structures, inputs information for each book, and then displays the information for each book.

**Task 5:**

* The **User** structure has a member variable name to store the username.
* The program prompts the user to enter their username, simulates a login process (you should replace this with actual authentication logic), and then displays a personalized welcome message upon successful login.

**Task 6:**

* Create a Structure named **StudentGrading** for collecting information about a student, such as name, SAP ID, address, department, and marks for two subjects.
* The **calculateMaxMarks** function calculates the maximum marks for a student based on their performance in the two subjects.
* **The displayStudent** function showcases the student's information, including their name, SAP ID, address, department, marks for two subjects, and the calculated maximum marks.
* The main function acts as the program's entry point. It creates an array of Student structures to manage details for five students. It utilizes loops to input information, calculate maximum marks, and then display the overall information for each student.

Table 3: Evaluation of the Lab

|  |  |  |
| --- | --- | --- |
| **Task No** | **Description** | **Marks** |
| 1 | Task 1 | 1.5 |
| 2 | Task 2 | 1.5 |
| 3 | Task 3 | 1.5 |
| 4 | Task 4 | 1.5 |
| 5 | Task 5 | 2 |
| 6 | Task 6 | 2 |
|  | **Total** | **10** |

# Further Reading

## 6.6 Books

1. The slides and reading material can be accessed from the folder of the class instructor available at Moellim.
   1. **Out comes**

The outcomes of this lab were:

* + - Functions
    - Structures