**Logo, company name

Description automatically generated**

**Department of (Computer Science)**

**Pak-Austria** Fachhochschule**: Institute of Applied Sciences and Technology, Haripur, Pakistan**

**COMP-112L Object Oriented Programming Lab**

**Lab Journal**

**Class: BS Computer Science**

**Name: Ahmed Raza**

**Registration No.: B20F0436CS031**

**Semester: 3rd**

**Submission Date: 2nd December, 2021**

**Submitted to: Engr. Rafi-Ullah**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Instructor Signature**

**Lab No. 05**

**Class Constructers**

**Objectives:**

In this lab we will be discussing about Class Constructors in detail. This is one of the most important concepts in Object Orientation C++. A class constructor is a special function in a class that is called when a new object of the class is declared. It therefore provides the opportunity to initialize objects as they are created and to ensure that data members only contain valid values. d. The primary function of a class constructor is to assign initial values to the data elements of the class, and no return type is necessary or, indeed, permitted.

**Tools/Software Required:**

* All the tasks are implemented on DEV C++.

**Introduction:**

**Assigning Default Values in a Constructor**

If we put the definition of the member function inside the class definition, we can put the default values for the parameters in the function header. If we only include the prototype of a function in the class definition, the default parameter value should go in the prototype.

**Using an Initialization List in a Constructor**

We can demonstrate this with an alternative version of the constructor for the class Box:

// Constructor definition using an initialization list

Box(double lv=1.0, double bv=1.0, double hv=1.0): length(lv), breadth(bv), height(hv)

{

cout << endl << "Constructor called.";

}

The initializing list for the constructor is separated from the parameter list by a colon and that each of the initializers is separated by a comma. This technique for initializing parameters in a constructor is important, because, as we shall see later, it's the only way of setting values for certain types of data members of an object.

**Private Members of a Class**

We can get it by using the keyword private when we define the class members. Class members which are private can, in general, only be accessed by member functions of a class. There's one exception, but we'll worry about that later. A normal function has no direct means of accessing the private members of a class.

Using a constructor or a member function is now the only way to get a value into a private data member of an object. You have to make sure that all the ways in which you might want to set or modify private data members of a class are provided for through member functions.

The default access attribute which applies to members of a class is private. You could, therefore, put all your private members at the beginning of the class definition and let them default to private by omitting the keyword. However, it's better to take the trouble to explicitly state the access attribute in every case, so there can be no doubt about what you intend.

**Accessing private Class Members**

On reflection, declaring all the data members of a class as private might seem rather extreme. It's all very well protecting them from unauthorized modification, but that's no reason to keep their values a secret. What we need is a Freedom of Information Act for private members. You don't need to start writing to your state senator to get it - it's already available to you. All you need to do is to write a member function to return the value of a data member. Look at this member function for the class Box:

inline double Box::GetLength(void)

{

return length;

}

**The Default Copy Constructor**

Suppose we declare and initialize a Box object Box1 with this statement:

Box Box1(78.0, 24.0, 18.0);

We now want to create another Box object, identical to the first. We would like to initialize the second Box object with Box1.

**Copy constructer**

A copy constructor does exactly what we're doing here - it creates an object of a class by initializing it with an existing object of the same class. The default version of the copy constructor creates the new object by copying the existing object, member by member.

**Destructor:**

A destructor is a special member function of a class that is executed whenever an object of it's class goes out of scope or whenever the delete expression is applied to a pointer to the object of that class.

A destructor will have exact same name as the class prefixed with a tilde (~) and it can neither return a value nor can it take any parameters. Destructor can be very useful for releasing resources before coming out of the program like closing files, releasing memories etc.

**Lab Tasks:**

**Task # 01:**

Passing parameters to constructors is same as passing parameters to normal function. Design a C++ program for Entering Employee data using classes. Your Employee class must have 4 data members i.e. Employee id, first name, last name and salary.

**Code:**

**#include<iostream>**

**#include<string>**

**using namespace std;**

**class Employee**

**{**

**private:**

**int age;**

**public:**

**int id;**

**string firstname;**

**string lastname;**

**int salary;**

**Employee(int i, string fn, string ln, int s)**

**{**

**cout << endl << "Constructor called."<<endl;**

**id=i;**

**firstname=fn;**

**lastname=ln;**

**salary=s;**

**}**

**Employee(int ag)**

**{**

**age = ag;**

**}**

**void set\_age(int a)**

**{**

**age=0;**

**age = a;**

**}**

**int get\_age()**

**{**

**return age;**

**}**

**void display()**

**{**

**cout<<endl<<"Employee id: "<<id;**

**cout<<endl<<"Employee First\_Name: "<<firstname;**

**cout<<endl<<"Employee Last\_Name: "<<lastname;**

**cout<<endl<<"Employee salary: "<<salary;**

**cout<<endl<<"Employee age: "<<age;**

**}**

**};**

**int main()**

**{**

**int Age;**

**Employee Employee1= Employee(1,"Muhammad","Abdullah",10000);**

**cout<<"Enter age for Abdullah: ";**

**cin>>Age;**

**Employee1.set\_age(Age);**

**Employee1.display();**

**Employee1.get\_age();**

**cout<<endl;**

**Employee Employee2= Employee(2,"Muhammad","Hassan",8000);**

**cout<<"Enter age for Hassan: ";**

**cin>>Age;**

**Employee2.set\_age(Age);**

**Employee2.display();**

**cout<< endl;**

**cout<<endl;**

**Employee Employee3= Employee(3,"Muhammad","Hanan",6000);**

**cout<<"Enter age for Hanan: ";**

**cin>>Age;**

**Employee3.set\_age(Age);**

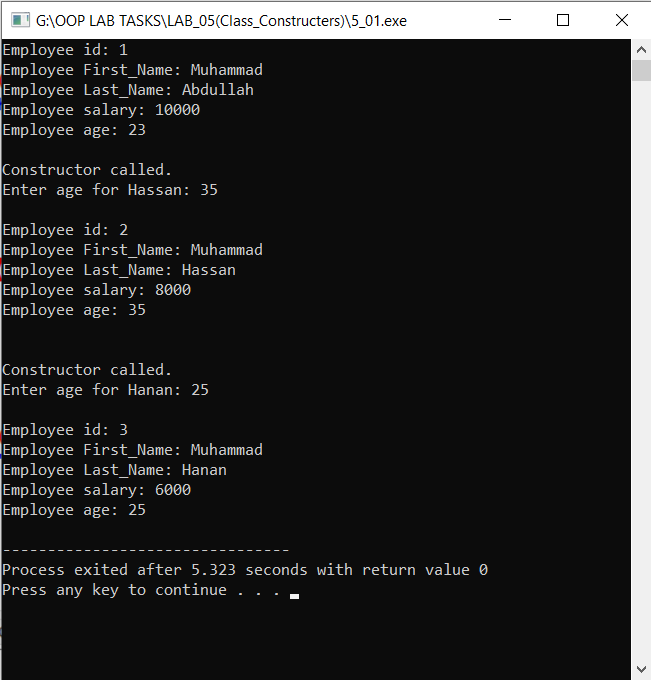
**Employee3.display();**

**cout<<endl;**

**return 0;**

**}**

**Output:**

****

**Task #02:**

Passing parameters to constructors is same as passing parameters to normal function. Design a C++ program for Entering Employee data using classes. Your Employee class must have 4 data members i.e. Employee id, first name, last name and salary.

**Code:**

**// Using a constructor**

**#include <iostream>**

**#include <string>**

**using namespace std;**

**class Student**

**{**

**public :**

**Student(int sr,string n, int m, char g)**

**{**

**serial=sr;**

**name=n;**

**marks=m;**

**grade=g;**

**}**

**void display()**

**{**

**cout<<serial<<"\t\t"<<name<<"\t\t"<<marks<<"\t\t"<<grade;**

**}**

**private:**

**int serial;**

**string name;**

**int marks=0;**

**char grade;**

**};**

**int main()**

**{**

**cout<<"No.\t\tName\t\t\tMarks\t\tGrade"<<endl;**

**Student Student1= Student(1,"Abdullah ",800, 'A');**

**Student1.display();**

**cout<<endl;**

**Student Student2= Student(2,"Ahmed \t", 600, 'B');**

**Student2.display();**

**cout<<endl;**

**Student Student3= Student(3,"Usman \t", 700, 'D');**

**Student3.display();**

**cout<<endl;**

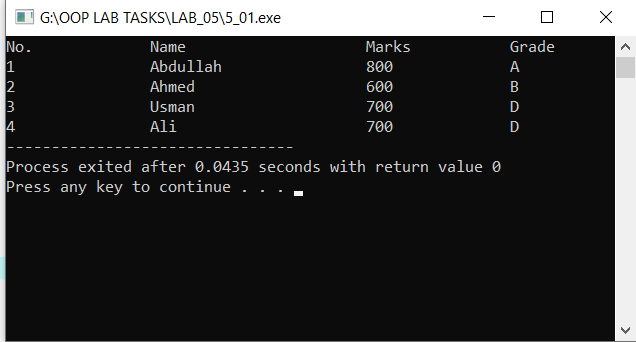
**Student Student4= Student(4,"Ali \t", 700, 'D');**

**Student4.display();**

**return 0;**

**}**

**Output:**

****

**Results & Observations:**

In this Lab I’ve learned about the concept of Classes and its objects & also I understand the concept of constructers that how can we use constructer in the class. As well as I learned about destructor. In the first task, I’ve used a Class of student in which I’ve used Constructer (that’s having same name as the class) and a display function for printing the data through the main function calling it by the class object and in the second task I’ve used constructers overloading where I’ve used two constructers in which one of them is taking variables like Employee name, id, salary but in the second constructer I’ve passed age variable that’s actually a private so that’s why I’m accessing it through set(), get() functions. Here I’m taking only age for every Employee from user to put to use the constructer overloading methodology.