**EVEREST ENGINEERING COLLEGE**

  **SANEPA, LALITPUR**

(AFFILIATED TO POKHARA UNIVERSITY)

AN

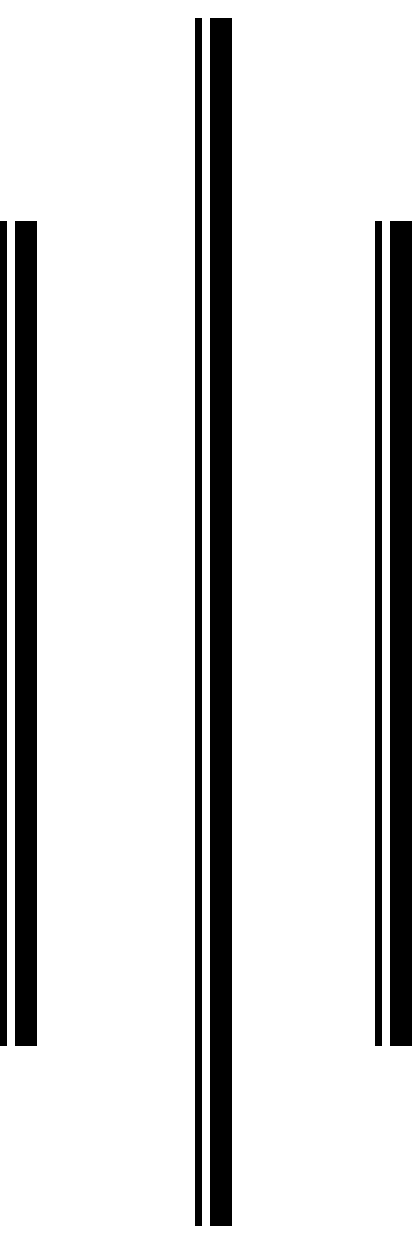
LAB REPORT

ON

**Object Oriented Programming In C++**

***[Classes & Object]***

**Lab Sheet: 2**



**SUBMITTED BY SUBMITTED TO**

Name:-**SAKCHYAM ACHARYA** Er.Pradip Paudel

Roll No:-40 Everest Engineering College

Department of Computer Engineering

Faculty:-BE CMP

Year/Semester: - 2nd SEM

Batch: 2021

Verified By:-

Student's Signature

**Index Page:-**

Topics Page No

* Title, Objective....................................................................................2
* Theory……………………………………………………………………………….2-3
* Introduction to Object Oriented Programming
* Class & Object
* Defining member function inside & outside the function
* Data Hiding
* Abstraction
* Access Specifier
* Array of object
* Memory Allocation Of Object
* Structure vs Class
* Questions with Source Code & Output…………………………………………...3-11
* Discussion & Conclusion ...................................................................................11

**Title:-**Class & Object.

**Objective:-**

* Explaining Classes and Object.

• Understanding the difference between Structures and Classes.

• Performing tasks based on classes.

**Theory:-**

**Introduction to Object Oriented Programming:-** Object-Oriented Programming or OOPs refers to languages that use objects in programming. Object-oriented programming aims to implement real-world entities like inheritance, hiding, polymorphism, etc in programming. The main aim of OOP is to bind together the data and the functions that operate on them so that no other part of the code can access this data except that function.

**OOPs Concepts:**

* Class
* Objects
* Data Abstraction
* Encapsulation
* Inheritance
* Polymorphism
* Dynamic Binding
* Message Passing

**Class:**A class is a user-defined data type. It consists of data members and member functions, which can be accessed and used by creating an instance of that class. It represents the set of properties or methods that are common to all objects of one type. A class is like a blueprint for an object.

**Object:**

It is a basic unit of Object-Oriented Programming and represents the real-life entities. An Object is an instance of a Class. When a class is defined, no memory is allocated but when it is instantiated (i.e. an object is created) memory is allocated. An object has an identity, state, and behavior. Each object contains data and code to manipulate the data. Objects can interact without having to know details of each other’s data or code, it is sufficient to know the type of message accepted and type of response returned by the objects.

**Encapsulation:**

Encapsulation is defined as the wrapping up of data under a single unit. It is the mechanism that binds together code and the data it manipulates. In Encapsulation, the variables or data of a class are hidden from any other class and can be accessed only through any member function of their class in which they are declared. As in encapsulation, the data in a class is hidden from other classes, so it is also known as **data-hiding**.

**Data Abstraction:**

Data abstraction is one of the most essential and important features of object-oriented programming. Data abstraction refers to providing only essential information about the data to the outside world, hiding the background details or implementation.

**Array of object:** An array in C/C++ or be it in any programming language is a collection of similar data items stored at contiguous memory locations and elements can be accessed randomly using indices of an array.

**Memory Allocation Of Object:** The way memory is allocated to variables and functions of the class is different even though they both are from the same class. The memory is only allocated to the variables of the class when the object is created. The memory is not allocated to the variables when the class is declared.

* **Problems with Source Code & Output:-**

**Problem No:1**

1. **Create a class named Employee with data members (EID, name, address, salary). Create member function named getdata() to take input from user and display () to display the information of employee. In main() create two objects of class Employee and call member function getdata() and display() for each object.**

*Source Code:*

#include<iostream>

using namespace std;

class employee{

private:

string name,address;

unsigned long int emp\_id;

float salary;

public:

void getdata(){

cout<<endl<<" Input employee detail";

cout<<endl<<"Enter the Employee Id:";

cin>>emp\_id;

cout<<"Enter the Name of the employee:";

getline(cin>>ws,name);

cout<<"Enter the Address of the employee:";

getline(cin>>ws,address);

cout<<"Enter the salary:";

cin>>salary;

};

void display(){

cout<<endl<<" Employee Detail";

cout<<endl<<"Employee No="<<emp\_id<<endl;

cout<<"Full Name="<<name<<endl;

cout<<"Monthly Salary="<<salary<<endl;

cout<<"Address="<<address<<endl;

};

};

int main(){

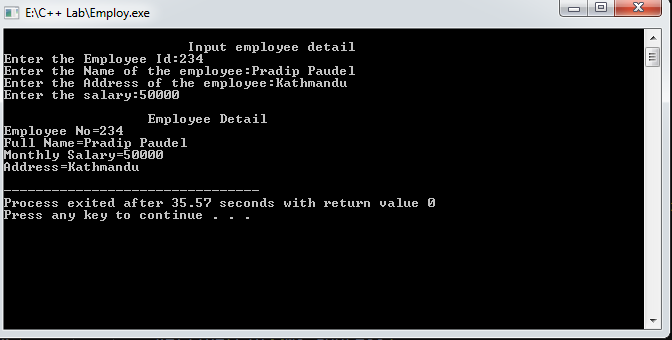
employee E1;

E1.getdata();

E1.display();

return 0;

}

****

**Problem No:-2**

1. ***Modify the above program for 50 Employees.***

*Source Code:-*

 #include<iostream>

using namespace std;

class employee

{

private:

int emp\_id;

string name;

int salary;

string address;

public:

void getempdetail();

void showempdetail();

};

void employee::getempdetail()

{

cout << "Enter the employee id:";

cin>>emp\_id;

cout << "Enter the name of employee:";

getline(cin>>ws,name);

cout << "Enter the salary:";

cin>>salary;

cout << "Enter the address:";

getline(cin>>ws,address);

cout<<endl;

}

void employee::showempdetail()

{

cout<<"Employee No="<<emp\_id<<endl;

cout<<"Full Name="<<name<<endl;

cout<<"Monthly Salary="<<salary<<endl;

cout<<"Address="<<address<<endl;

}

int main(){

employee e[50];

cout <<endl<<" Input Employee Detail"<<endl;

for(int i=0;i<50;i++)

{

e[i].getempdetail();

}

cout<<" Employee Details:"<<endl;

for (int i=0;i<50;i++)

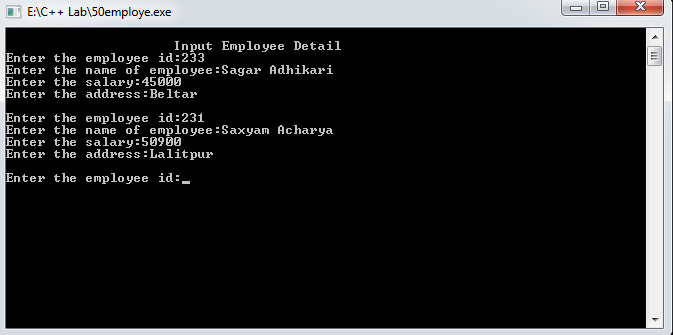
{

e[i].showempdetail();

}

return 0;

}

****

**Problem No 3:-**

**b.Modify Qno.(a) to display the information of employee whose salary is greater than 50000.**

Source Code:

#include<iostream>

using namespace std;

class employee

{

private:

int emp\_id;

string name;

int salary;

string address;

public:

void getempdetail(){

cout << "Enter the employee id:";

cin>>emp\_id;

cout << "Enter the name of employee:";

getline(cin>>ws,name);

cout << "Enter the salary:";

cin>>salary;

cout << "Enter the address:";

getline(cin>>ws,address);

cout<<endl;

}

void showempdetail(){

if (salary>=50000){

cout<<"Employee No="<<emp\_id<<endl;

cout<<"Full Name="<<name<<endl;

cout<<"Monthly Salary="<<salary<<endl;

cout<<"Address="<<address<<endl;

cout<<endl;

}

}

};

int main(){

employee e[3];

cout <<endl<<" Input Employee Detail"<<endl;

for(int i=0;i<3;i++)

{

e[i].getempdetail();

}

cout<<" Employee Details:"<<endl;

for (int i=0;i<3;i++)

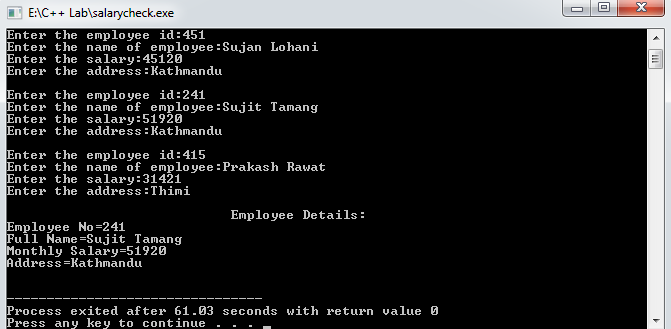
{

e[i].showempdetail();

}

return 0;

}

****

**Problem No:4**

***Create a structure named Employee with members (EID, name, address, salary) now read the information of employee and display them.***

*Source Code:-*

//Create a structure

#include<iostream>

using namespace std;

struct employee{

int emp\_id;

string name,address;

int salary;

}e1;

int main(){

cout<<" Input employee details";

cout<<endl<<"Enter the employee id:";

cin>>e1.emp\_id;

cout<<"Enter the Name:";

getline(cin>>ws,e1.name);

cout<<"Enter the Address:";

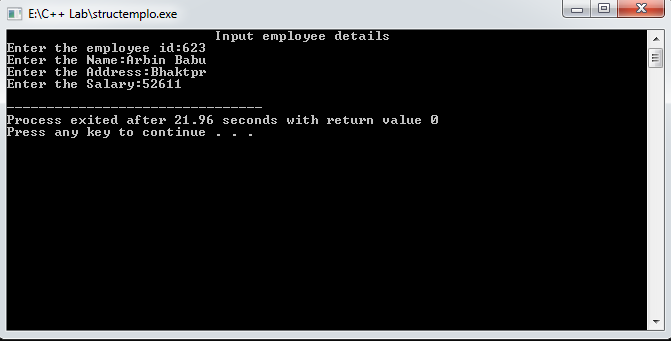
getline(cin>>ws,e1.address);

cout<<"Enter the Salary:";

cin>>e1.salary;

return 0;

}



**Problem No:-5**

**How classes are different from structure?**

* **Class** is defined as a collection of related variables and functions contained within a single structure. **Structure** is a grouping of variables of various data types referenced by the same name. In **Structure**, the default accessibility is **public**, whereas in **class** the default is **private**.

**Problem No:-6**

**Demonstrate the program to define member function inside the class and outside the class.**

***£.Member function inside the class:***

**#Source Code:**

#include<iostream>

using namespace std;

class Honda{

private:

int salno,r1,r2;

char dualsport[20],nakedstreet[30];

public:

void getdata()

cout<<" Welcome! to HONDA MOTORS PVT.LTD.";

cout<<endl<<endl<<"Your Favourite DualSport?: ";

cin>>dualsport;

cout<<"Your rating on this bike out of 10:"<<endl;

cin>>r1;

cout<<"Your Favourite NakedStreet?: ";

cin>>nakedstreet;

cout<<"Your rating on this bike out of 10:"<<endl;

cin>>r2;

}

void display(){

cout<<endl<<" Customer Feedback";

cout<<endl<<"Dual Sport Segment:"<<dualsport<<endl;

cout<<"Rating on this bike out of 10:"<<r1<<endl;

cout<<endl<<"Naked Street Segment:"<<nakedstreet<<endl;

cout<<"Rating on this bike out of 10:"<<r2<<endl;

cout<<endl<<"Thanks for participating in our survey";

}

};

int main(){

Honda H1;

H1.getdata();

H1.display();

return 0;

}

***¥.Member function outside the class:***

**#Source Code:**

#include<iostream>

using namespace std;

class Honda{

private:

int salno,r1,r2;

char dualsport[20],nakedstreet[30];

public:

void getdata(void);

void display(void);

};

void Honda :: getdata(){

cout<<" Welcome! to HONDA MOTORS PVT.LTD.";

cout<<endl<<endl<<"Your Favourite DualSport?: ";

cin>>dualsport;

cout<<"Your rating on this bike out of 10:"<<endl;

cin>>r1;

cout<<"Your Favourite NakedStreet?: ";

cin>>nakedstreet;

cout<<"Your rating on this bike out of 10:"<<endl;

cin>>r2;

}

void Honda ::display(){

cout<<endl<<" Customer Feedback";

cout<<endl<<"Dual Sport Segment:"<<dualsport<<endl;

cout<<"Rating on this bike out of 10:"<<r1<<endl;

cout<<endl<<"Naked Street Segment:"<<nakedstreet<<endl;

cout<<"Rating on this bike out of 10:"<<r2<<endl;

cout<<endl<<"Thanks for participating in our survey";

}

int main(){

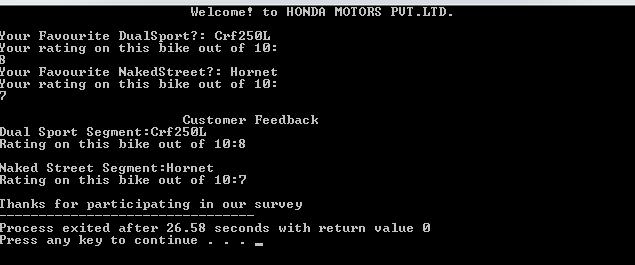
Honda H1;

H1.getdata();

H1.display();

return 0;

}



***#What is the role of access specifier?***

* **Access modifiers** (or **access specifiers**) are keywords in object-oriented languages that set the accessibility of classes, methods, and other members. **Access modifiers** are a specific part of programming language syntax used to facilitate the encapsulation of components.

**Discussion & Conclusion:-**

The program is focused on various task on Class & Object .From this program I understood the difference of class & structure, how to define member function inside & outside the class.

**Thank You**<SAKWheels>