Introduction to C++ : Classes and Objects

Note: Create a class with a "class_name" and save the program with the name

"class_name.cpp"

The main purpose of C++ programming is to add object orientation to the C programming language and classes are the central feature of C++ that supports object-oriented programming and are often called user-defined types. A class is used to specify the form of an object and it combines data representation and methods for manipulating that data into one neat package. The data and functions within a class are called members of the class.

C++ Class Definitions:

When you define a class, you define a blueprint for a data type. This doesn't actually define any data, but it does define what the class name means, that is, what an object of the class will consist of and what operations can be performed on such an object.

A class definition starts with the keyword class followed by the class name; and the class body, enclosed by a pair of curly braces. A class definition must be followed either by a semicolon or a list of declarations. For example, we defined the Box data type using the keyword class as follows:

```
class Box
{
public:
double length; // Length of a box
double breadth; // Breadth of a box
double height; // Height of a box
};
```

The keyword public determines the access attributes of the members of the class that follow it. A public member can be accessed from outside the class anywhere within the scope of the class object. You can also specify the members of a class as private or protected which we will discuss in a sub-section.

Define C++ Objects:

A class provides the blueprints for objects, so basically an object is created from a class. We declare objects of a class with exactly the same sort of declaration that we declare variables of basic types. Following statements declare two objects of class Box:

Box Box1; // Declare Box1 of type Box

Box Box2; // Declare Box2 of type Box

Both of the objects Box1 and Box2 will have their own copy of data members.

Accessing the Data Members:

The public data members of objects of a class can be accessed using the direct member access operator (.). Let us try the following example to make the things clear:

```
#include <iostream>

using namespace std;

class Box

{

public:

double length; // Length of a box

double breadth; // Breadth of a box

double height; // Height of a box

};

int main( )

{

Box Box1; // Declare Box1 of type Box

Box Box2; // Declare Box2 of type Box

double volume = 0.0; // Store the volume of a box here

// box 1 specification

Box1.height = 5.0;

Box1.length = 6.0;

Box1.breadth = 7.0;

// box 2 specification

Box2.height = 10.0;

Box2.length = 12.0;
```

```
Box2.breadth = 13.0;

// volume of box 1

volume = Box1.height * Box1.length * Box1.breadth;

cout << "Volume of Box1 : " << volume <<endl;

// volume of box 2

volume = Box2.height * Box2.length * Box2.breadth;

cout << "Volume of Box2 : " << volume <<endl;

return 0;

}
```

When the above code is compiled and executed, it produces the following result:

```
Volume of Box1 : 210

Volume of Box2 : 1560
```

It is important to note that private and protected members can not be accessed directly using direct member access operator (.). We will learn how private and protected members can be accessed.

Example of Class in C++ (using private data members and public member functions to access them)

```
class temp

{

private:

int data1;

float data2;

public:

void func1()

{ data1=2; }

float func2(){
```

```
data2=3.5;

return data;

}

};
```

Explanation

There are two keywords: private and public mentioned inside the body of class. Keywords: private and public Keyword private makes data and functions private and keyword public makes data and functions public. Private data and functions are accessible inside that class only whereas, public data and functions are accessible both inside and outside the class. This feature in OOPS is known as data hiding. If programmer mistakenly tries to access private data outside the class, compiler shows error which prevents the misuse of data. Generally, data are private and functions are public.

/* Program to illustrate working of Objects and Class in C++ Programming */

```
#include <iostream>

using namespace std;

class temp

{

private:

int data1;

float data2;

public:

void int_data(int d){

data1=d;

cout<<"Number: "<<data1;

}

float float_data(){

cout<<"\nEnter data: ";
```

```
cin>>data2;
return data2;
}
};
int main(){
temp obj1, obj2;
obj1.int_data(12);
cout<<"You entered "<<obj2.float_data();
return 0; }
```

Lab Assignment 9
Week 10 (8th April – 13th April 2019)

Object oriented programming

Q 1: Define a class to represent a bank account. Include the following members:

Data members

1. Name of the depositor
2. Account number
3. Type of account
4. Balance amount in the account.

Member functions

1. To assign initial values
2. To deposit an amount
3. To withdraw an amount after checking the balance
4. To display name and balance

Write a main program to test the program.

Q 2: Modify the class and the above program for handling 10 customers.

Q 3: Design appropriate C++ classes for the following Data structures:

a) Stack

b) Queue

Add suitable member functions to above mentioned classes and/ or new classes and call them using their objects.

Q 4: The class Rectangle contains the instance variables height, width, and colour. It implements the following methods:

1. Rectangle();
2. Rectangle(int size) ; // Creates a square where height = width = size
3. Rectangle(int height, int width);
4. Rectangle(int height, int width, Int colour);
5. Rectangle(Rectangle otherRectangle);
6. void set(Int colour);
7. void set(int size);
8. void set(int height, int width) ;
9. void set(int height, int width, Colour colour) ;

Q 5: Design a class called Message. The Message class models a simplified e-mail message, with the following member variables (with the obvious meanings):

string from;

string to;

string text; //the message body

int time_stamp;

Give the class the following member functions:

- A constructor that takes sender & recipient, creates an empty message time-stamped with the instant of creation.
- A function that appends a line of text (passed as a string) to the message body.

Q 6: Create a class student having name, roll no and semester.

- Create a default constructor
- Create a parameterized constructor with three parameters and default value 1 for semester.
- In main function create an object s1 with name Abc, Roll no. 221 and semester 3.
- Now change the semester of s1 from 3 to 4.
- Create an object s2 with values user defined.
- Create a copy constructor for the class
- Create a new object s3 with same values as s1.
- Create an array of n students where n is user defined.

Write a function check() which is global function and not a member function of class to check whether the data members of s1 and s3 are same or not.

Q 7: Define a class ratio with numerator and denominator as member variables. Write the constructors, destructors and copy constructor. Overload the following operators for the same:

• Arithmetic operators : +, -, *, /

• Relational operators: <=, >=, ==, !=

• Assignment operator : =

• Arithmetic assignment operator: +=, -=

• Pre and post increment operators: ++

• Stream input and output operators: << and >>

Lab Assignment 10
Week 11 (15th April – 20th April 2019)

Object oriented programming

1. What will happen if you execute following programs:

(a)

```
#include <iostream.h>
#include <string.h>
class Book
{
public:
Book(char *title) { strcpy(Book::title, title);
};
void show_title(void) { cout << title << endl;
};
protected:
float cost;
void show_cost(void) { cout << cost << endl;
};
private:
char title[64];
};
class LibraryCard : public Book
{
public:
LibraryCard(char *title, char *author, char
*publisher) : Book(title)
{
strcpy(LibraryCard::author, author);
strcpy(LibraryCard::publisher, publisher);
cost = 49.95;
};
void show_library(void)
{
show_title();
show_cost();
cout << author << ' ' << publisher;
};
private:
char author[64];
char publisher[64];
};
int main(void)
{
LibraryCard card("A", "B", "C");
card.show_library();
}
```

(b)

```
#include <iostream.h>
class B {
protected:
int i, j;
public:
void se(int a, int b) {
i = a;
j = b;
}
void sh() {
cout << i << " " << j << endl;
int m;
public:
void sem() {
m = i_j;
}
void shm() {
cout << m << endl;}};
int main()
{
B obj1;
B obj2;
obj1.se(2, 3);
obj1.sh();
obj1.sek();
obj1.shk();
obj2.se(3, 4);
obj2.sh();
obj2.sek();
obj2.sem();
obj2.shk();
obj2.shm();
return 0;}
```

(c)

```
#include <iostream.h>
class ViBase
{
public:
ViBase(const char* s): m_s(s)
{
cout << "_ ViBase constructor called: 0x"
<< hex << this
<< " s=" << s << dec << endl << endl;
};
const char* GetS() const
{
return m_s;
};
private:
const char* m_s;
};
class Base : virtual public ViBase
{
public:
Base(const int k): ViBase("bad"),
m_k(k)
{
cout << "_ Base constructor called: 0x" <<
hex << this
<< dec << endl << endl;
};
const int GetK() const
{
return m_k;
};
hex << this
<< dec << endl << endl;
};
void Print()
{
cout << endl << "Base : S = " << GetS() <<
endl
<< " K = " << GetK() << endl << endl;
};
private:
int m_a;
};
private:
const int m_k;
};
class Final: public Base
{
public:
Final(const char *s,
const int k)
: Base(k),
ViBase(s)
{
cout << "_ Final constructor called: 0x" <<

int main(int argc, char* argv[])
{
Final *x = new Final("ObjectX", 2007);
if (x == NULL)
cout << endl << "Out of memory." << endl
<< endl;
x_>Print();
cout << "GetS() = " << x_>GetS() << endl;
cout << "GetK() = " << x_>GetK() << endl;
delete x;
return 0;
}
```

(d)

```
#include <iostream.h>
class Base
{
public:
Base() {cout << "In Base::Base()\n";}
virtual ~Base() {}
virtual void virtualFunc() {cout << "In
Base::virtualFunc()\n";}
};
class Derived1 : virtual public Base
{
public:
Derived1() {cout << "In
Derived1::Derived1()\n";}
virtual ~Derived1() {}
virtual void virtualFunc() {cout << "In
Derived1::virtualFunc()\n";}
};
class Derived2 : virtual public Base
{
public:
Derived2()
{
cout << "In Derived2::Derived2()\n";
virtualFunc();
}
virtual ~Derived2() {}
};
class MostDerived : public Derived1, public
Derived2
{
public:
MostDerived() {cout << "In
MostDerived::MostDerived()\n";}
virtual ~MostDerived() {}
};
int main()
{
int wait;
MostDerived md;
md.virtualFunc();
cin >> wait;
}
```

2. Write a program to read and print student information with department and grade information using hierarchical inheritance, multiple inheritance and multilevel inheritance.

3. Assume that a bank maintains two kinds of accounts for customers, one called as savings account and other as current account. The savings account provides compound interest and withdrawal facilities but no cheque book facility. The current account provides cheque book facility but no interest. Current account holders also maintain a minimum balance and if the balance falls below this level, a service charge is imposed. Create a class account that stores customer name, account number and type of account. From this derive the classes cur_acct and sav_acct to make them more specific into their requirements. Include necessary member functions in order to achieve the following tasks:

a) Accept deposit from a customer and update the balance

b) Display the balance.

c) Compute and deposit interest.

d) Permit withdrawal and update the balance.

e) Check for minimum balance, impose penalty, necessary and update the balance.

4. class employee

{

private:

char name[LEN]; //employee name

unsigned long number; //employee number

}

Derive a class called employee2 from the employee class in the above EMPLOY program. This new class should add a type double data item called compensation, and also an enum type called period to indicate whether the employee is paid hourly, weekly, or monthly. For simplicity you can change the manager, scientist, and labourer classes so they are derived from employee2 instead of employee. However, note that in many circumstances it might be more in the spirit of OOP to create a separate base class called compensation and three new classes manager2, scientist2, and laborer2, and use multiple inheritance to derive these three classes from the original manager, scientist, and labourer classes and from compensation. This way none of the original classes needs to be modified.