The Shirpur Education Society's

R. C. Patel College of Engineering & Polytechnic, Shirpur

Department of Computer Engineering

COMPUTER ENGINEERING

Course Title - Object Oriented Programming Using C++

Course Code - 313304

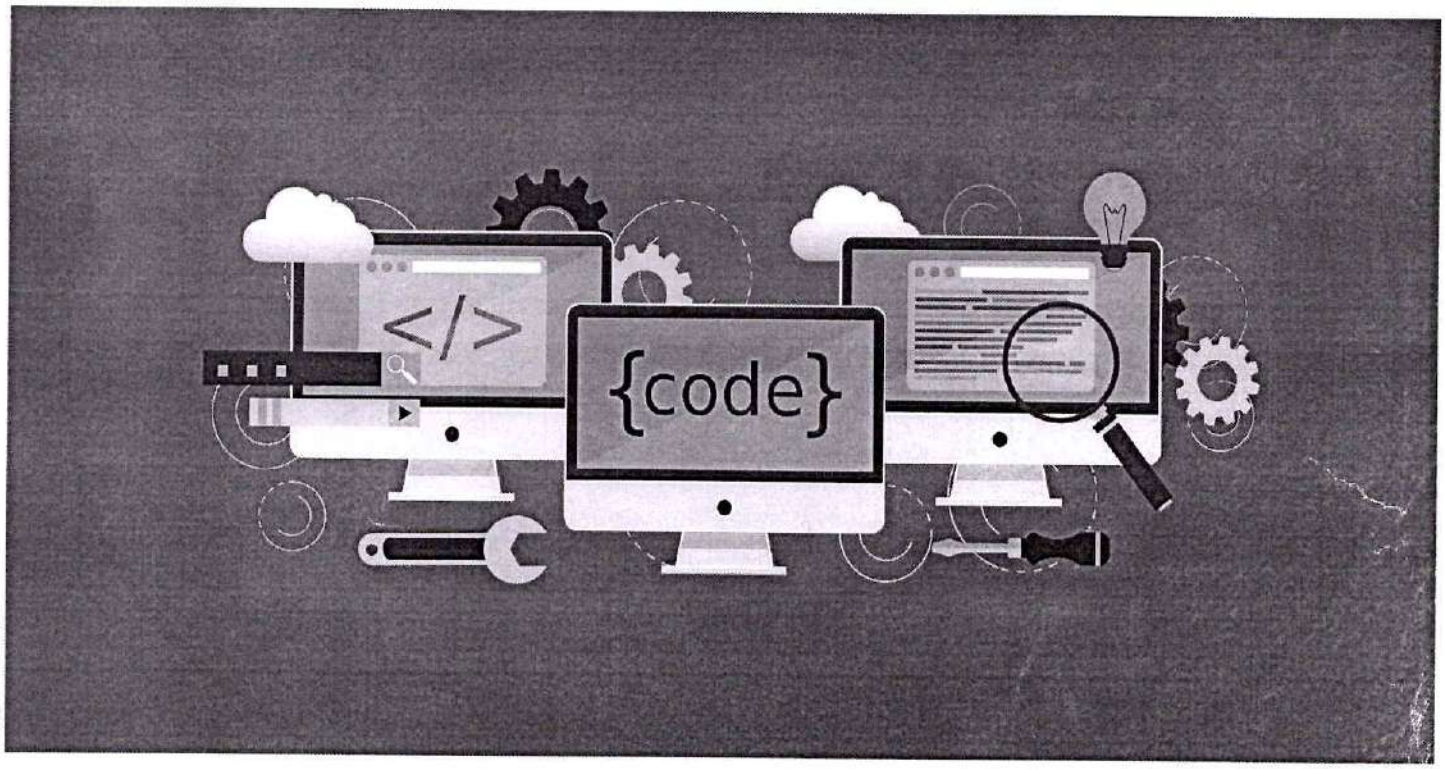
Programme Name - Computer Engineering

Semester - Third

Unit - I Principles of Object Oriented Programming

Total Marks:12

Prepared By: Mrs.Pratiksha S. Patil



Subject: oop using c++

Unit 1.

Principles of object oriented programming

oop is a method of programming in which data (variables) and functions (methods) are combined into a single unit called an object.

eg. Car

- Data (Attributes) → color, speed, model
- Functions (methods) → start(), stop()

* Basic concept / Features of oop :-

1. Class
2. object
3. Data abstraction
4. Data Encapsulation
5. Inheritance
6. Polymorphism
7. Dynamic Binding
8. message Passing.

1] Class :-

1) A class is a collection of objects of similar type.

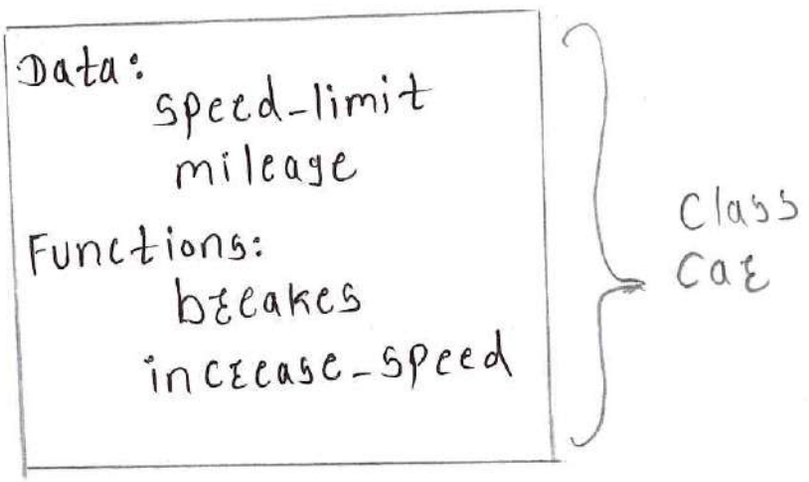
2) A class is like a Blueprint for an object.

3) classes are the user defined data types, which hold it's own data members & member function, which can be accessed & used by creating an instance (object) of a class.

4) Once the class has been defined, We can create any number of objects belonging to that class.

5) class is also known as collection of data members & member function.

eg. class of cars will share some common properties Wheels, speed limit, mileage etc.



2] object :-

1) objects are basic run time entities in an object oriented system.

2) They may represent a person, a place, a bank account, a table of data or any item that the program has to handle.

3) objects is a real World entity
eg. chair, car, pen, mobile etc.

4) object is the instance of a class. All members of a class can be accessed through objects.

5) eg. Student it has peopeeties oe attebutes like eoll no, name, addeess & functions aee total, aveeage, etc.

Object: student
Attebutes: Rollno name addeess
Functions: total() aveeage() display()

3] Data Abstraction

1) Hiding complex implementation details & exposing only essential featuees to the usee, simplifying code & eeducing complexity.

2) 'Abstraction' woed indicates that showing the necessaey data. Anothee meaning is hiding unnecessaey data.

4] Data Encapsulation

1) The weapping up of data & functions togethee into a single unit (called class) is known as encapsulation.

2) By this attribute the data is not accessible to the outside world & only those functions which are wrapped in the class can access it.

5] Inheritance

1) Inheritance is a process of creating new class from existing class.

2) Inheritance is the process by which objects of one class acquire the properties of objects of another class.

3) It supports the concept of hierarchical classification. It also provides the idea of reusability.

6] Polymorphism

1) Polymorphism means the ability to take more than one form.

2) In oop, it refers to the ability of different objects to respond to the same method call in different ways.

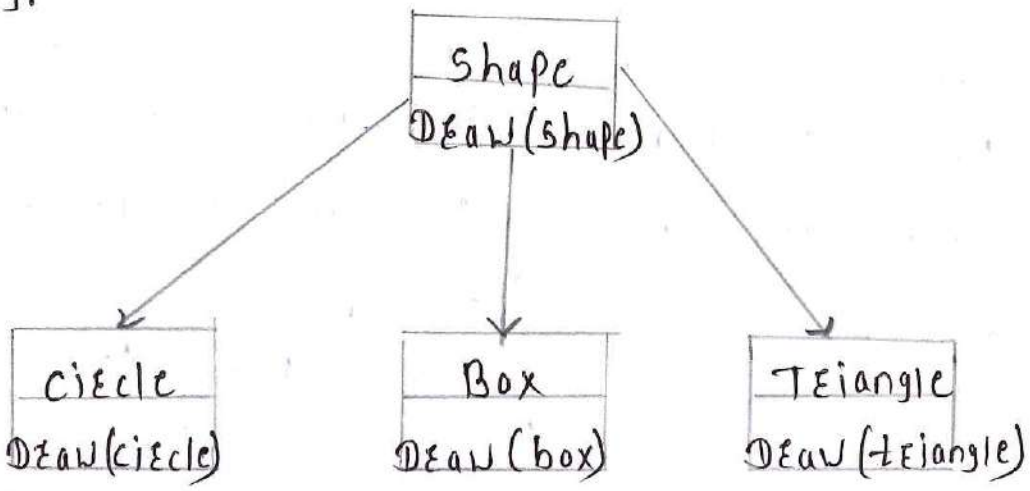
Poly + morphism

↓ ↓

many Acquiring forms

3) Consider a person who performs many roles in daily life, such as he can be father, husband, son, brother at the same time he can be employee of an organization.

4) eg.



7] Dynamic Binding

1) Process of deciding which method to call at runtime rather than at compile time.

2) When a program runs, the system be executed based on the actual object type, not the reference type.

8] message passing

1) objects communicate with one another by sending & receiving information much the same way as people pass message to one another.

2) When one object wants another object to perform some action, it sends a message

* Procedure Oriented Programming (POP) Verses Object Oriented Programming (OOP)

POP	OOP
① POP is a structure oriented.	① OOP is a object oriented
② Program divided into small parts called function	② Program is divided into parts called object
③ Top-Down approach	③ Bottom-up approach
④ POP does not have any access specifier	④ OOP has access specifier public, private, protected.
⑤ Global data accessible by all functions.	⑤ Data is hidden & accessed using member function.
⑥ Overloading is not possible	⑥ Overloading is available in form of operator overloading & function overloading
⑦ Data can move freely from function to function	⑦ object can move & communicate with each other
⑧ Less secure - data can be freely accessed.	⑧ more secure - access specifier used.
⑨ Inheritance, Polymorphism not supported	⑨ Inheritance, Polymorphism are supported.
⑩ Limited code reusability	⑩ High reusability via inheritance & Polymorphism
⑪ eg. C, VB, Pascal	⑪ eg. C++, Java, VB.Net

* Examples of object oriented languages

1. C++
2. Java
3. Python
4. VB.net
5. C#.net
6. Object Pascal
7. Simula
8. Ada
9. Turbo Pascal
10. Eiffel

* Applications of OOP

1. Real-time system
2. Simulation & modeling
3. object oriented databases
4. Hypertext, Hypermedia & expertext
5. AI & expert system
6. Neural networking & Parallel Programming
7. Stimulation & modeling system
8. client-server system
9. CIM/CAD/CAM system

* Some of striking features of OOP

1. Emphasis is on data rather than procedure.
2. Program are divided into parts called objects.
3. Data structure are designed such that they characterize the objects.
4. Data is hidden & cannot be accessed by external function.
5. objects communicate with each other through function.

⑥ New data & function can be easily added whenever necessary.

⑦ Follow bottom-up approach in program design.

* Data types →

In c++, data types are classifed into following categories.

1) Basic / Built-in data type :-
   int, float, double, char, void

2) Derived Data types :-
   Array, Pointer, Reference, function

3) User defined data types :-
   union, class, structure, enumeration, typedef, using.

* Type compatibility →

- Type compatibility refers to whether different data types can be used together in operations or assignments.
- c++ allows conversion between related types
  eg. int → double
  float → int
- No actual conversion
- checked by compiler

* Declaration of Variables

- Variable names in c++ can range from 1 to 255 characters.
- All variables names must begin with a letter of the alphabet or underscore (_).
- After the first initial letters, variable names can also contain letters & numbers.
- Variable names are case sensitive
- No space or special characters are allowed.
- You cannot use a c++ keyword (reserved word) as a variable name.

Syntax:-

```
datatype variable-name;
```

for eg:-

```
int age;  // declare variable

float salary=30000;  // declare & initialize
                        variable.

int x=10, y=5, z=15;  // declares & initialize
                         three integer variables.
```

* Type casting / Type Conversion

- Type conversion → also called implicit type conversion. Automatically done by compiler.

```
eg. int a=10;
    double b=a;
```

- Type casting → also called explicity type conversion. User manually changes data from one type to another type.

eg. double Pi = 3.14;
int Pi = (int) Pi;

* Dynamic Initialization of Variable

- Dynamic initialization means initializing a variable at runtime using an expression or value calculated during program execution.
- The value is not fixed at compile time. It may depend on user input or calculations.

eg.
```
float average = sum / i; // initialize dynamically at run time
```

* Reference Variable

A Reference variable in c++ is an alias for another variable. It allows you to create a second name for the same memory location. Any change made to the reference are actually performed on the original variable.

Syntax:
datatype & referenceName = original variable;

eg. int a = 10;
int &ref = a;

* operators in c++ :

1. Arithmetic operators (*, +, /, -, %, ++, --)
2. Relational operator (==, !=, <, >, <=, >=)
3. logical operator (&&, !)
4. BitWise operator (&, ^, ~, <<, >>)
5. Increment & decrement operator (++, --)
6. conditional operator (?)
7. Pointer operator (*, &)
8. Assignment operator (=, +=, -=, *=, /=, %=, <<=, >>=, &=, ^=)

* Special operator in c++

Scope Resolution operator (::) → Scope resolution operator is used for:

1) Accessing a global variable when there is a local variable with same name.
2) Defining a function outside a class.
3) Accessing a class's static variables.
4) Referring to a class inside another class.
5) In multiple inheritance
6) Namespace

* Manipulators

Manipulators are operators that are used to format the data display. most commonly used manipulators are endl & setw. iomanip header file are used.

1) endl → insert a newline character.
2) setw → set width of next output field.
3) setprecision → control decimal precision.
4) fixed → display fixed decimal format.
5) ws → Remove White space from input.

* Structure of c++ program

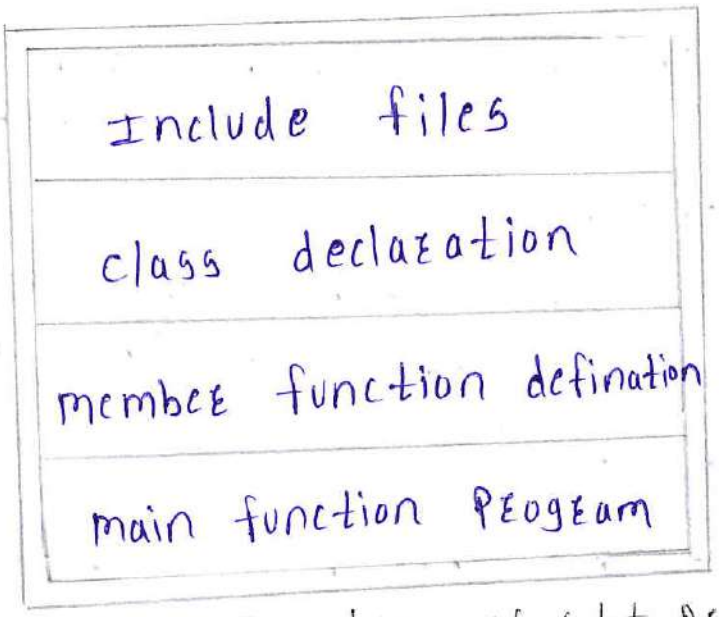


fig. structure of c++ program

① Include files →

In this section a programmer include all header files which are require to execute given program. The most important file is

iostream.h header file. This file defines most of the c++ statements like cout & cin. Without this file cannot load c++ program.

② class declaration →

In this section a programmer declare all classes which are necessary for given program. The programmer uses general syntax of creating class.

③ member function definition →

This section allows programmer to design member function of a class. The programmer can have inside declaration of a function or outside declaration of a function.

④ main function program →

In this section programmer creates objects & calls various functions written within various class.

* Input operator (>> cin) → Extraction operator

- Used to take input from user
- Work with std :: cin

* Output Operator (<< cout)

<< → Insertion operators

- Used to display output to the screen
- Work with std :: cout

```
eg. int age;
    std :: cin >> age;
    std :: cout << age;
```

* Input function

① cin.get() → Read a single character

```
char ch;
ch = cin.get();
```

② getline () → reads full line (string)

```
string str;
getline (cin.str);
```

* output function

① cout.put() → Prints a single character

```
cout.put('A');
```

② cout.Write → Prints string with length

```
cout.Write("HelloWorld", 5);
```

* Simple c++ program →

```
#include<iostream>
using namespace std;
int main()
{
  int number;
  cout<<"Enter a number;"; //ask input
  cin >> number ;          //Read i/p from user
 cout<<" Number is: "<<number<<endl;
 return 0;
}
```

* Without using namespace std;

```
#include<iostream.h>
  int main()
  {
    int number;
    std:: cout<<" Enter no";
    std:: cin >> number;
    std:: cout<<" Number is: "<<number;
    return 0;
  }
```

* Class & objects →

Class → A blueprint or template for creating objects. It defines properties (data) & behaviors (functions).

Object → An instance of a class that holds actual values.

* specifying a class

class specification has two parts:

① class declaration → describe the type & scope of its members.

② class function → describe how the class functions are implemented.

The general form of a class declaration is:

```
class class-name
{
  private:
        variable declarations;
        function declarations;

  public:
        variable declaration;
        function declaration;

};
```

eg.

```
class student
{
public: int id;
        string name;
void display()
{cout<<id<<" "<<name;
}
};
```

*Create a object :-

Syntax :-

```
ClassName objectName;
```

eg :-

```
class student
{
  .....
};
int main()
{
  student S1;   //create object S1
  student S2;   // create object S2

}
```

*member function declaration in C++ :-

Syntax :-

1) Inside a class

```
class className
{
  Public :
    returntype functionName(Parameters);

};
```

eg. class student
```
{
  public: void getdata()
          {
            cout<<"Inside class";
          }
};
```
} Write function inside a class

2) outside a class

Syntax:-
```
class className
{
  public: returnType functionname();

};
returntype className :: functionname()
{
}
```
eg. class student
```
{ public: void getdata();
};
void student :: getdata()
{
 cout<<"outside class";
}
```
} Write function outside using scope resolution operator (::)

* Access specifiers

In c++, there are 3 access specifie namely as public, private, protected.

① public → accessible from anywhere

② private → accessible only within class.

③ protected → accessible within class and derived class.

* Defining member function :-

① Inside the class

② outside the class

1) Inside the class

```
class Box
{
public:
    int l;
void showlength()
{
cout<<"length"<<l;
}
};
```

2) outside the class

```
class Box
{
public:
    int l;
   void showlength();
};//definition outside
void Box :: showlength()
{
cout<<"length"<<l;
}
```

* Creating objects :-

```
int main()
{
    student s1;  // object of class student
    s1.id = 101;
    s1.display();
}
```

* Memory Allocation for objects :-

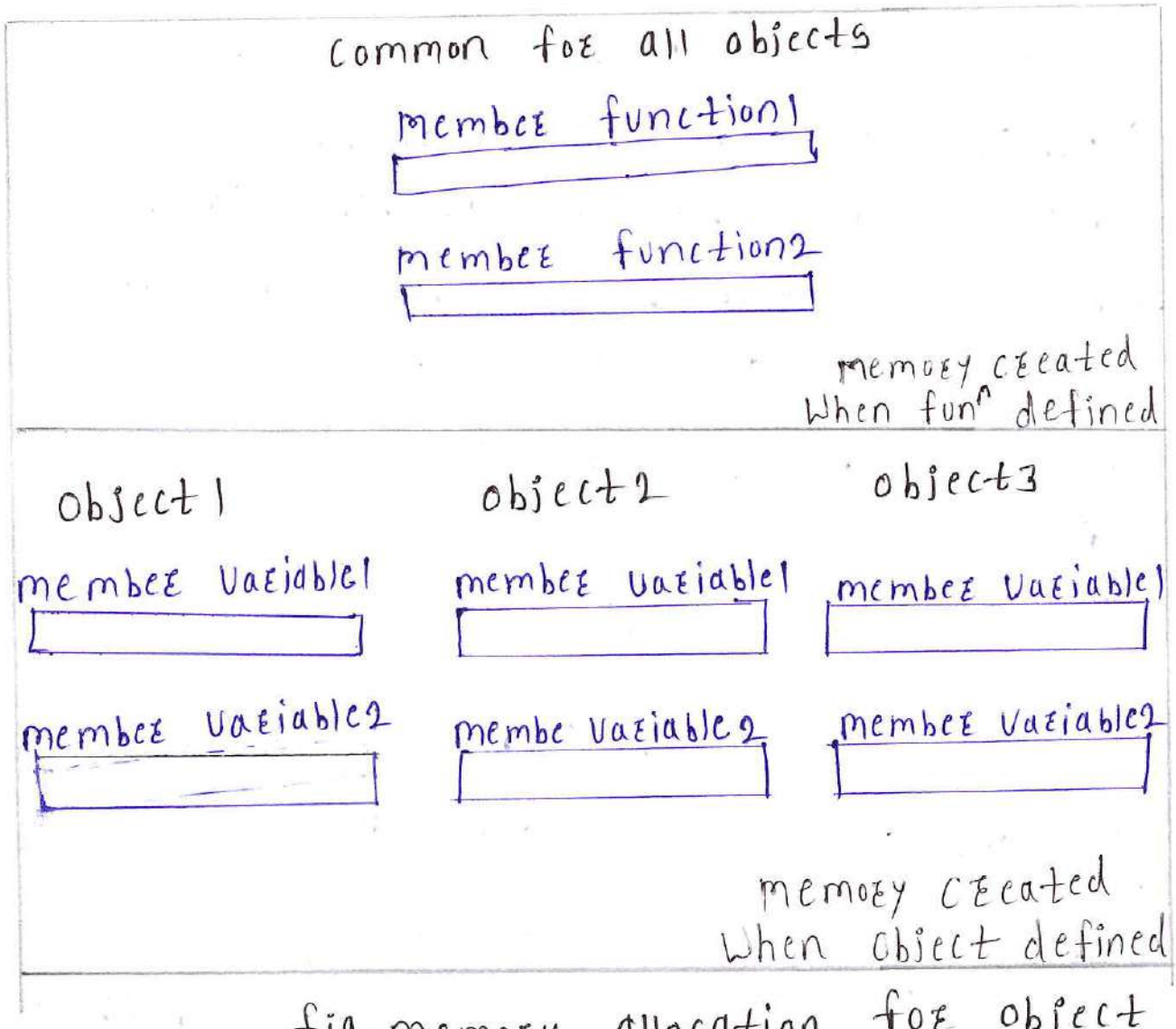


fig. memory allocation for object

① The memory space for object is allocated when they are declared & not when the class is specified.

② Actually, the member function are created & placed in memory space only once when they are defined as a part of a class definition.

③ All the objects belonging to that class use the same member function, no seperate space is allocated for member function.

④ When the objects are created only space for member variable is allocated seperately for each object.

⑤ Seperate memory location for the objects are essential because the member variables will hold different data values for different objects this is shown in above fig.

* Difference between class & structure :-

Class	Structure
① Class is a reference type & its object is created on the heap memory.	① structure is a value type that is why its variable is created on the stack memory.
② class can have the all types of constructor & destructor	② structure doesn't have constructor or destructor.
③ class can inherit the another class.	③ structure does not support the inheritance
④ The member variable of class can be initialized directly.	④ The member variable of structure cannot be initialized directly.

* Difference between C & C++.

C	C++
① C supports the procedural style programming.	① C++ supports both procedural & object oriented.
② C follows the top-down approach.	② C++ follow the bottom-up approach.
③ develop by Dennis Ritchie.	③ develop by Bjarne stroustrup.
④ Appeared in 1972	④ Appeared in 1985.
⑤ C does not support function overloading.	⑤ C++ support function overloading.
⑥ C does not support reference variable.	⑥ C++ support reference variable.
⑦ In C, scanf() & printf() are mainly used for input/output.	⑦ C++ mainly uses stream cin & cout to perform i/p & o/p operation
⑧ operator overloading is not possible.	⑧ operator overloading is possible.
⑨ C program are divided into procedures & modules.	⑨ C++ program are divided into functions & classes.
⑩ C does not support inheritance.	⑩ C++ supports inheritance.

P Patil
2015/26

2015/26.

Subject: oop using c++

Unit 1.

Principles of object oriented Programming

* Question Bank

1. State application of oop (W19, W22, S25)
2. State the features of oop (S23, W25)
3. Describe use of scope resolution operator with example (W19, S19, W22, W24)
4. Define class & object with syntax (W19, S19, W25)
5. Write the syntax for declaration of class (S23, W24, W25)
6. Describe structure of c++ program (S19, W18, S23, S22)
7. Difference between oop & pop (W18, W19, W24)
8. Difference between class & structure (W25, W24)
9. Describe the process of memory allocation for object with suitable diagram (W19, S19, W25)
10. Explain insertion & extration operator with suitable example (W24, S19)
11. Write any four benefit, object oriented languages (W18, W19)

12. Demostrate the static & dynamic initialization of variables (S23).

13. Write a program to declare a class 'circle' with data members as radius & area. declare a function getdata to accept radius & putdata to calculate & display area of circle. (W18)

14. Write a program to create a class STUDENT The data members of student class,

Roll_No, Name, marks (W18, S19)

15. Write a process to define member function outside the class (W25)

BPatil
2015/26

2015/26