Course Hand Out

Subject Name: Object Oriented Programming Using C++

Prepared by: Mr. K. Vamshee Krishna, Assistant Professor, CSE

Year, Semester, Regulation: II Year- I SEM (R18)

UNIT –I

KEY POINTS:

- **Object oriented Programming**
  Object oriented Programming is defined as an approach that provides a way of modularizing programs by creating partitioned memory area for both data and functions that can be used as templates for creating copies of such modules on demand. Writing object-oriented programs involves creating classes, creating objects from those classes, and creating applications, which are stand-alone executable programs that use those objects
- **Class** A class is a user defined data type. A class is a logical abstraction. It is a template that defines the form of an object. A class specifies both code and data. It is not until an object of that class has been created that a physical representation of that class exists in memory. When you define a class, you declare the data that it contains and the code that operates on that data. Data is contained in instance variables defined by the class known as data members, and code is contained in functions known as member functions. The code and data that constitute a class are called members of the class.
- **Object** An object is an identifiable entity with specific characteristics and behavior. An object is said to be an instance of a class. Defining an object is similar to defining a variable of any data type. Space is set aside for it in memory.
- **Encapsulation** Encapsulation is a programming mechanism that binds together code and the data it manipulates, and that keeps both safe from outside interference and misuse. C++'s basic unit of encapsulation is the class. Within a class, code or data or both may be private to that object or public.
- **Data abstraction** In object oriented programming, each object will have external interfaces through which it can be made use of. There is no need to look into its inner details. The object itself may be made of many smaller objects again with proper interfaces. The user needs to know the external interfaces only to make use of an object.
- **Inheritance** Inheritance is the mechanism by which one class can inherit the properties of another. It allows a hierarchy of classes to be build, moving from the most general to the most specific. When one class is inherited by another, the class that is inherited is

called the base class. The inheriting class is called the derived class. In general, the process of inheritance begins with the definition of a base class. The base class defines all qualities that will be common to any derived class.

- **Polymorphism** Polymorphism (from the Greek, meaning "many forms") is a feature that allows one interface to be used for a general class of actions. The specific action is determined by the exact nature of the situation. The concept of polymorphism is often expressed by the phrase "one interface, multiple methods." This means that it is possible to design a generic interface to a group of related activities.
- **C++** : C++ is an object oriented programming language. It was developed by Bjarne Stroustrup in 1979 at Bell Laboratories in Murray Hill, New Jersey. He initially called the new language "C with Classes." However, in 1983 the name was changed to C++.
- **Data types :** Data type defines size and type of values that a variable can store along with the set of operations that can be performed on that variable. C++ provides built-in data types that correspond to integers, characters, floating-point values, and Boolean values.
- **Variable: A** variable is a named area in memory used to store values during program execution. Variables are run time entities. A variable has a symbolic name and can be given a variety of values. When a variable is given a value, that value is actually placed in the memory space assigned to the variable.
- **Operator**: An operator is a symbol that tells the compiler to perform specific mathematical or logical manipulations. C++ is rich in built-in operators. Generally, there are six type of operators: Arithmetical operators, Relational operators, Logical operators, Assignment operators, Conditional operators, Comma operator.
- **Typecasting**: Typecasting is the concept of converting the value of one type into another type. For example, you might have a float that you need to use in a function that requires an integer.
- **Four typecast operators** : The C++ language has four typecast operators:
    - static_cast
    - reinterpret_cast
    - const_cast
    - dynamic_cast.
- **Control Structures:** Control structures allows to control the flow of program's execution based on certain conditions C++ supports following basic control structures: 1) Selection Control structure 2) Loop Control structure.
- **C++ Functions** : A function groups a number of program statements into a unit and gives it a name. This unit can then be invoked from other parts of the program. The function's code is stored in only one place in memory, even though the function is executed many times in the course of the program's execution.
- **Inline Functions**: An inline function is a function that is expanded inline at the point at which it is invoked, instead of actually being called. The reason that inline functions are an important addition to C++ is that they allow you to create very efficient code.

- **Reference variable**: A reference variable is an alias, that is, another name for an already existing variable. Once a reference is initialized with a variable, either the variable name or the reference name may be used to refer to the variable. T
- **Call by Value:** - In this method the values of the actual parameters (appearing in the function call) are copied into the formal parameters (appearing in the function definition), i.e., the function creates its own copy of argument values and operates on them**.**
- **Call by Reference**: - A reference provides an alias – an alternate name – for the variable, i.e., the same variable's value can be used by two different names : the original name and the alias name

Short Questions

1) What is programming paradigm? list the different paradigms for problem solving?

2) Write the difference between OOP and procedure oriented programming?

3) Define OOP. List the different types of operator in C++?

4) Define operator. List different types of operators in C++?

5) Define type conversion?

6) Discuss in brief about strings?

7) Define flow control statements?

8) Define function .list the components of c++ function?

9) What is an inline function?

10) Define preproceesor directives?

Long Questions

1) What is programming paradigms? Explain about different paradigms for problem solving?

2) Illustrate the structure of a C++ Program?

3) What is an operator? list and explain the types of operators in c++?

4) Describe about type conversions?

5) Discuss in brief about multidimensional array?

6) list and explain string manipulation functions?

7) Discuss about flow control statements?

8) Discuss about functions in detail?

9) Discuss the different parameter passing techniques?

10) what is an inline function? explain how to implement it?

Fill in the Blanks

1) **Abstract** is a process of showing relevant details and hiding irrelevant details from the user.

2) Object oriented programming allows code **Reusability**

3) Comments in C++ are represented by **2types (single line and multiline)**

4) By default, all members functions defined inside the class are treated as **Inline function**

5) **Data type** is used to indicate the type of data value stored in a variable.

6) A variable that stores address of another variable is called a **pointer**

7) **Stremp()** function is used to compare two strings.

8) **Switch** is a multiway decision making statement.

9) **Type conversion** refers to the process of converting the data of one type into another type.

10) An **array** is a variable which is capable of holding fixed values of same type in contiguous memory location.

Multiple Choice Questions

1) A simple C++ program execute_____. [a]
   a) Without preprocessor directives b) Object code
   c) With pre-processor directives d) Standard library function

2) Which one of the below things is not considered as a key component of object oriented programming _____. [d]
   a) Inheritance b) Encapsulation c) Polymorphism d) parallelism

3) In C++ the function main has default return type _____. [b]
   a) Float b) Integer
   b) Character d) None of the above

4) Default arguments are defined when functions are,_____. [b]

a) Defined b) Declared

b) Both(a)and(b) d) None of the above

5) The scope resolution operator (::) is used to access_____. [c]

a) Objects b)Local variables

c)) Global variables d)Botha (a) and (b)

6)Identify which of the following functions calls are allowed [b]

1. sum(2)
2. sum(2,3)
3. sum()
4. sum(3,4,5)

a) 1,4 b)1,2 and 4

c)2 and 4 d) 2 and 1

7) Inline function_____ [c]

a) Executes Slowly b) wastes memory

c) Saves memory d) Saves more time

8) _____ operator is used for deallocation [b]

a) new b) delete

c) deallocate d) allocate

9) The memory allocated dynamically can be accesses only using [a]

a) Pointers b)Structures

c)Unions d)Arrays

10)Which function is used to concatenate two strings? [c]

a)strcmp b)strlen

c)strcat d)strcmpi

UNIT-2

Key Points:

Application of C++

C++ is a versatile language for handling very large programs; it is suitable for virtually any programming task including development of editors, compilers, databases, communication systems and any complex real life applications systems.

• Since C++ allow us to create hierarchy related objects, we can build special object-oriented libraries which can be used later by many programmers.

• While C++ is able to map the real-world problem properly, the C part of C++ gives the language the ability to get closed to the machine-level details.

• C++ programs are easily maintainable and expandable. When a new feature needs to be implemented, it is very easy to add to the existing structure of an object.

• It is expected that C++ will replace C as a general-purpose language in the near future

Simple C++ Program

Let us begin with a simple example of a C++ program that prints a string on the screen. Printing A String

```
#include Using namespace std;

int main()

{

 cout< <<" c++ is better than c \n";

return 0;

}
```

STATIC CLASS MEMBERS

Data members and member functions of a class in C++, may be qualified as static. We can have static data members and static member function in a class.

Static Data Member:

It is generally used to store value common to the whole class

Static Member Function:

A static member function can access only the static members of a class.

FRIEND CLASSES

In C++ , a class can be made a friend to another class

A **constructor** (having the same name as that of the class) is a member function which is automatically used to initialize the objects of the class type with legal initial values.

Destructors are the functions that are complimentary to constructors. These are used to deinitialize objects when they are destroyed. A destructor is called when an object of the class goes out of scope, or when the memory space used by it is de allocated with the help of delete operator.

TYPE CONVERSIONS

We have overloaded several kinds of operators but we haven't considered the assignment operator (=). It is a very special operator having complex properties. We know that = operator assigns values form one variable to another or assigns the value of user defined object to another of the same type

Essay Questions:

1. Define a class. Explain how classes are defined.
2. Write in detail about class objects.
3. Explain about the scope of class.
4. Discuss about this pointer.
5. Write about friends to class.
6. What are static members of a class? Explain.
7. Discuss about constant member functions.
8. Define constructor. List and explain the different types of constructor in c++
9. What is a Destructor? Explain.
10. Explain about the dynamic creation and destruction of objects.

Short Questions:

1. Define class.
2. Write a short note on object.
3. Write a short note on scope of class.
4. Write about pointer.
5. What are static members of a class
6. Define constructor and destructor.

7. List the details to declare static member variables outside the class.
8. Write a short note on data abstraction.
9. What is ADT?.
10. Discuss about information Hiding.

Unit -3

KEY POINTS

BASE CLASS AND DERIVED CLASS

Let us take the classes, Employee and Manager. A Manager is an Employee with some additional information. when we are declaring the classes Employee and Manager without applying the concept of inheritance, they will look as follows:

```
class Employee

{

public: char* name;

int age;

char* address;

int salary;

char*department;

int id;

};
```

Now, the class Manager is as follows:

```
Class Manager

{

public: char* name;

int age; char* address;

int salary;
```

```
char*department;

int id;

 employee* team_members; //He heads a group of employees

 int level; // his position in hierarchy of the organisation

 };
```

SINGLE INHERITANCE ‘

In this Section, you will learn the ways of deriving a class from single class. So, there will be only one base class for the derived class.

New Operator

In C++, the pointer support dynamic memory allocation (allocation of memory during runtime). While studying arrays we declared the array size approximately. In this case if the array is less than the amount of data we cannot increase it at runtime. So, if we wish to allocate memory as and when required new operator helps in this context. The syntax of the new operator is given below :

```
 pinter_variable = new data_type;
```

Delete Operator

It is used to release or deallocate memory. The syntax of delete operator is : delete_pointer_variable;

VIRTUAL FUNCTIONS :Polymorphism is a mechanism that enables same interface functions to work with the whole class hierarchy. Polymorphism mechanism is supported in C++ by the use of virtual functions. The concept of virtual function is related to the concept of dynamic binding. The term Binding refers to binding of actual code to a function call. Dynamic binding also called late binding is a binding mechanism in which the actual function call is bound at run-time and it is dependent on the contents of function pointer at run time.

Polymorphim

Polymorphism means ‘one name multiple forms’. Runtime polymorphism can be achieved by using virtual functions.

STATIC POLYMORPHISM OR COMPILE TIME POLYMORPHISM

It means existence of an entity in various physical forms simultaneously. Static polymorphism refers to the binding of functions on the basis of their signature (number, type and sequence of parameters). It is also called early binding because the calls are type and sequence of parameters). It

is also called early binding because the calls are already bound to the proper type of functions during the compilation of the program.

DYNAMIC POLYMORPHISM

It means change of form by entity depending on the situation. A function is said to exhibit dynamic polymorphism if it exists in various forms, and the resolution to different function calls are made dyanamically during execution time. This feature makes the program more flexible as a function can be called, depending on the context.

Essay Questions:

1. Discuss in brief about class hierarchies.
2. What is Inheritance? List and Explain the different types of Inheritance.
3. Define briefly about (a) Base class (b) Derived class
4. Write an example program to illustrate the concept of constructor and destructor in inheritance
5. Write short note on virtual Base Class.
6. Explain in detail about the static and dynamic binding.
7. Explain in detail about virtual functions.
8. Write about Abstract Classes.
9. What is polymorphism? Explain the types of polymorphism.
10. Write a note on virtual destructors.

Short Questions:

1. What is Inheritance?
2. Define base class and Derived class.
3. What are Destructors?
4. Explain about virtual Base class.
5. Write about static and dynamic binding.
6. Write a short note on virtual functions.
7. Define pure virtual functions.
8. What are Abstract classes?
9. What is polymorphism?
10. Write short note on virtual destructor.

UNIT –IV

KEY POINTS

The C language did not build the input/output facilities into the language. In other words, there is no keyword like read or write. Instead, it left the IO to the compiler as external library functions (such as printf and scanf in stdio library). The ANSI C standard formalized these IO functions into Standard IO package (stdio.h). C++ continues this approach and formalizes IO in libraries such as iostream and fstream.

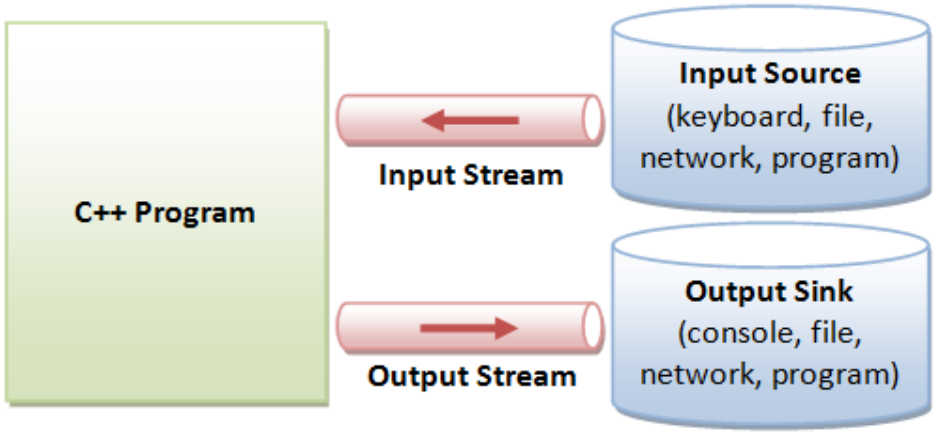
Features

C++ IO is type safe. IO operations are defined for each of the type. If IO operations are not defined for a particular type, compiler will generate an error.

C++ IO operations are based on streams of bytes and are device independent. The same set of operations can be applied to different types of IO devices.

1. Stream IO

C/C++ IO are based on *streams*, which are sequence of bytes flowing in and out of the programs (just like water and oil flowing through a pipe). In input operations, data bytes flow from an *input source* (such as keyboard, file, network or another program) into the program. In output operations, data bytes flow from the program to an *output sink* (such as console, file, network or another program). Streams acts as an intermediaries between the programs and the actual IO devices, in such the way that frees the programmers from handling the actual devices, so as to archive device independent IO operations.



Internal Data Formats:
- Text: char, wchar_t
- int, float, double, etc.

External Data Formats:
- Text in various encodings (US-ASCII, ISO-8859-1, UCS-2, UTF-8, UTF-16, UTF-16BE, UTF16-LE, etc.)
- Binary (raw bytes)

Header files available in C++ for Input/Output operations are:

1. **iostream**: iostream stands for standard input-output stream. This header file contains definitions to objects like cin, cout, cerr etc.
2. **iomanip**: iomanip stands for input output manipulators. The methods declared in this files are used for manipulating streams. This file contains definitions of setw, setprecision etc.
3. **fstream**: This header file mainly describes the file stream. This header file is used to handle the data being read from a file as input or data being written into the file as output.

SHORT TYPE

1 What is a stream? Explain with a neat diagram the C++ class hierarchy for stream handling.

2. Explain text made & binary made LIP w.r.t to

i) Character data

ii) Numeric data

3. Different between text files & binary file.

4. Explain the various functions available for text ILO in C++.

5. Explain the read () and write () function.

6. Explain the Open () function along with the various modes supported by it.

ESSAY TYPE

7. What is a file pointer? How can the file pointers can be explicitly manipulators.

8. Write a short notes on

i) SeeKP() ii) tellP()

iii) Seekg() iv) tellp()

9. How can a file be opened for both reading & writings?

10. What is the difference between opening a file using the constructor of stream class and open () function.

11. Explain with an example, how a file be randomly accessed C++.

12. Write short note on

i)eof() ii) fail() iii) bad() iv)Clear().

13.Explain with suitable examples the pre-defined manipulators available in C++

UNIT –V

KEY POINTS

Why Exception Handling?

Following are main advantages of exception handling over traditional error handling.

1) Separation of Error Handling code from Normal Code: In traditional error handling codes, there are always if else conditions to handle errors. These conditions and the code to handle errors get mixed up with the normal flow. This makes the code less readable and maintainable. With try catch blocks, the code for error handling becomes separate from the normal flow.

2) Functions/Methods can handle any exceptions they choose: A function can throw many exceptions, but may choose to handle some of them. The other exceptions which are thrown, but not caught can be handled by caller. If the caller chooses not to catch them, then the exceptions are handled by caller of the caller.
In C++, a function can specify the exceptions that it throws using the throw keyword. The caller of this function must handle the exception in some way (either by specifying it again or catching it)

3) Grouping of Error Types: In C++, both basic types and objects can be thrown as exception. We can create a hierarchy of exception objects, group exceptions in namespaces or classes, categorize them according to types.

SHORT TYPE

1.What is a function template? Explain with its syntax.

2.WAP to swap two numbers using template.

3.What is a class template? Explain with an example.

4.Write a brief on STL.

5.What a list class? Explain the various functions available this class.

6.Write short notes on
i) Vector class ii) Pair class iii) Map class iv) Set class

7.How does multi class and multiuse class duffer from map class and set class respectively?

ESSAY TYPE

8 Give a situation where exception handling is applied.

9.Define exception.

10.What are the options to handle an exception.

11Write a program to show exception handling.

12.Define and brief on STL.

13.What are containers? Give its application.

14.Give the usage of vectors.

15.Define list. Write a program to construct a list with various options.