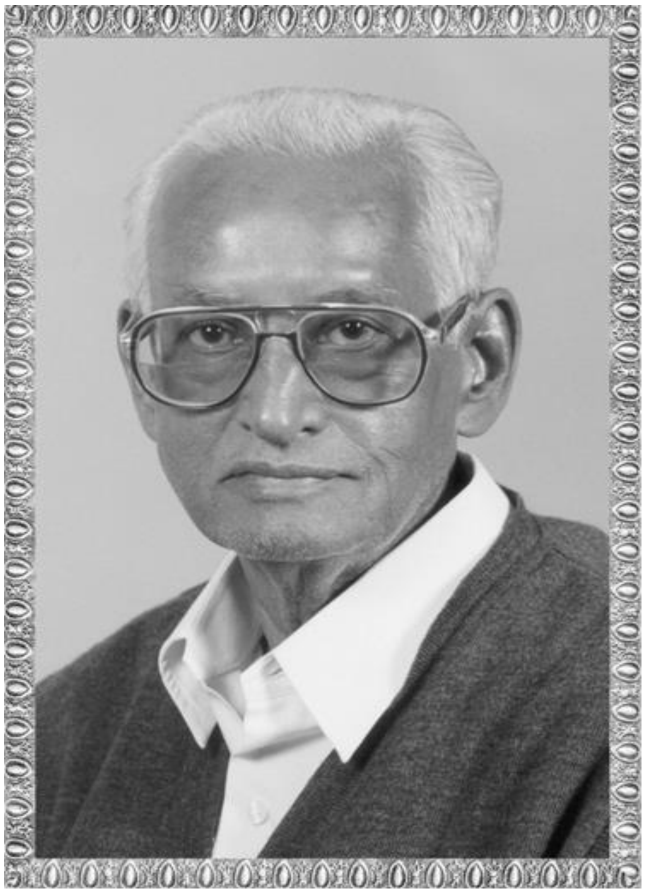


Lt. Shree Chimanbhai Shukla

BCA SEM 3 C++

Website: [www. hnsgroupofcolleges.org](www.hnsgroupofcolleges.org)
Email : hnsinfo@hnshukla.com

Sr No.	Topic	Details
1	**Principles of object oriented programming Tokens, expressions and control statements**	Procedure – oriented programming Object oriented programming paradigm Basic concepts of object oriented Programming Benefits of object oriented programming Application of object oriented programming What is c++? Application of c++ Input/output operators Structure of c++ program Introduction of namespace Tokens : keywords, identifiers, basic data types, user- defined types, derived data types, symbolic constants, type compatibility, declaration of variables, dynamic initialization of variables, reference variables Operators in C++: scope resolution operator, member referencing operator, memory management operator, manipulators, type cast operator. Expression : Expression and their types, special assignment operator, implicit conversions, operator precedence Control structures Conditional control structure :- • simple if, if…else , nested if else, switch etc. Looping control structure:- • for, while , do…while
	Functions in C++	The main function Function prototype Call by reference Return by reference Inline function Default arguments Const arguments

CH -1 Principles of object oriented programming tokens, expressions and control statements

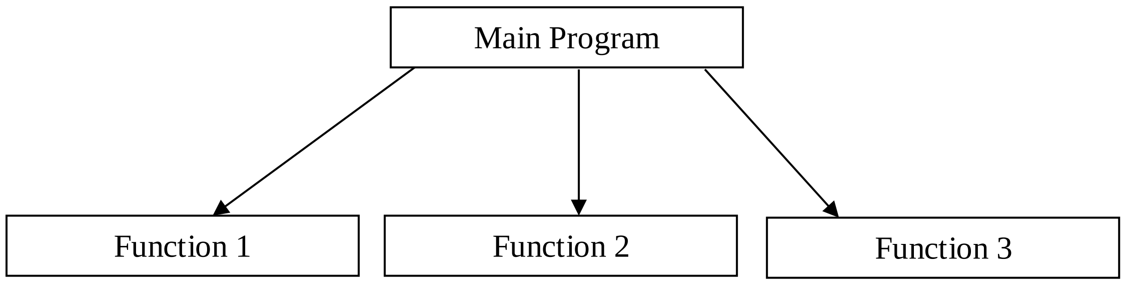
Topic: Procedure-Oriented Programming (POP)

Trailer: **(Only for Understanding)**

- function pe kaise focus kare?
- Kaise pura program function me divide kare?
- Data ko kaise manage kare?

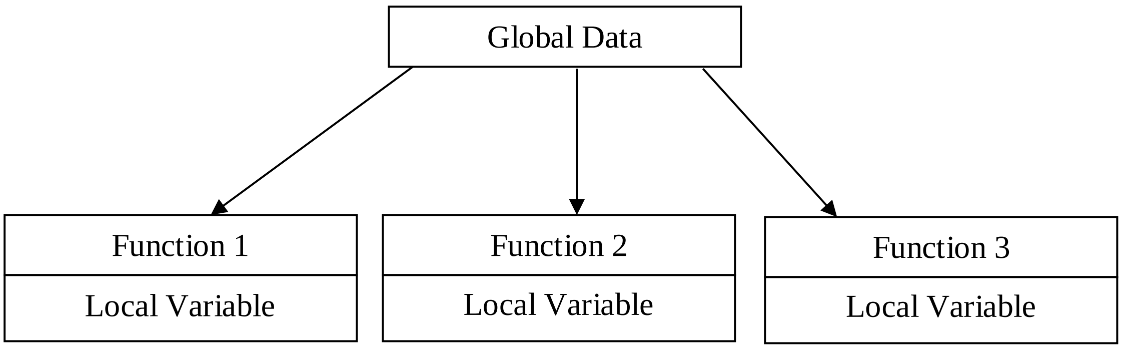
Details : **(for Exam Content)**

- Conventional programming using high level languages such as COBOL, FORTRAN and C, is commonly known as **procedure-oriented programming (POP)**.
- In the procedure-oriented approach, the problem is viewed as a sequence of things to be done such as reading, calculating and printing.
- A number of functions are written to complete these tasks.
- The primary focus is on functions. A typical program structure for procedural programming is shown in fig.



Procedure-oriented programming basically consists of writing of list of instructions for the computer to follow

- This instructions groups known as functions.
- We normally use a flow chart to organize these actions and represent the flow of control from one action to another.
- In a multi –function program, many important data items are placed as global so that they may be accessed by all the functions.
- Each function may have its own local data.



Learning Outcome: **(Summary)**

- ❖ Procedure means functions and procedure oriented programming means set of function.

1 word Question Answer

Sr No.	Question	Answer
1.	POP Stands for?	procedure-oriented programming

Topic: Object -Oriented Programming (OOP)

Trailer: **(Only for Understanding)**

- what is object?
- What is object oriented programming and why we need?

Details: **(for Exam Content)**

- Object Oriented programming (OOP) is a programming paradigm that relies on the concept of classes and objects.
- It is used to structure a software program into simple, reusable pieces of code blueprints (usually called classes), which are used to create individual instances of objects.
- Object Oriented Programming is a programming in which we design and develop our application or program based of object. Objects are instances (variables) of class.

- Object oriented programming does not allow data to flow freely around the system. It binds data more closely to the functions that operate on it, and protects it from accidental modifications from outside functions.

Learning Outcome: (Summary)

- ❖ In this topic we learn with the use of object and class we can reuse our block of code easily & reduce our line of code.

1 word Question Answer

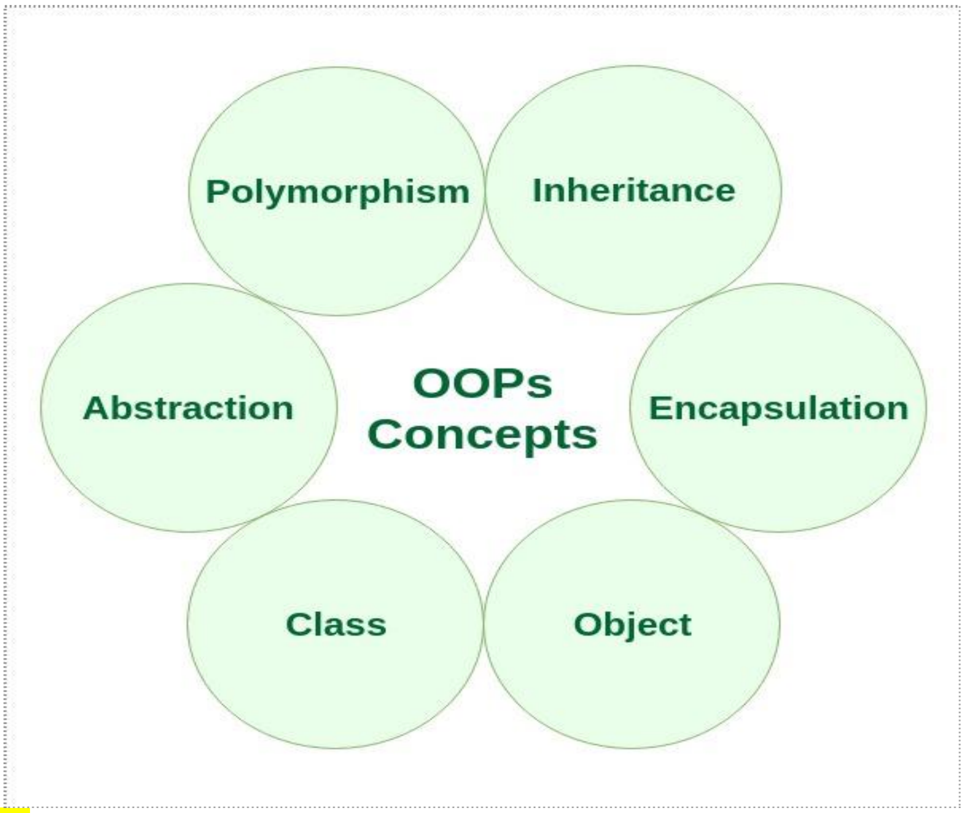
Sr No.	Question	Answer
1.	OOP Stands for?	Object-oriented programming

Topic: Basic Concept of OOP

Trailer: **(Only for Understanding)**

- How many concept of OOP? And what is the meaning of all the concept.

Details: **(for Exam Content)**



1. Class :

- Classes are an expanded version of structures. Structure can contain multiple variables.
- Classes can contain multiple variables, even more, classes can also contain functions as class member.
- Variables available in class are called Data Members.
- Functions available in class are called Member Functions.

2. Object :

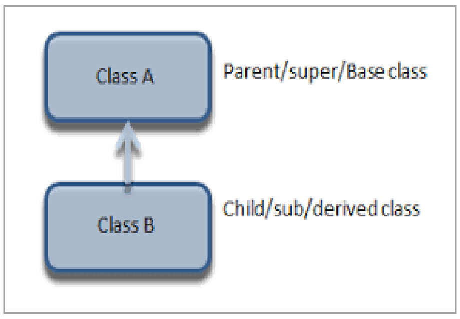
- Class is a user-defined data type and object is a variable of class type.
- Object is used to access class members.

- Ex Of Class & Object
-
```
class MyClass {       // The class
  public:             // Access specifier
    int myNum;        // Attribute (int variable)
    string myString;  // Attribute (string variable)
};
```

```
int main() {
  MyClass myObj;  // Create an object of MyClass
  // Access attributes and set values
  myObj.myNum = 15;
  myObj.myString = "Some text";

}
```

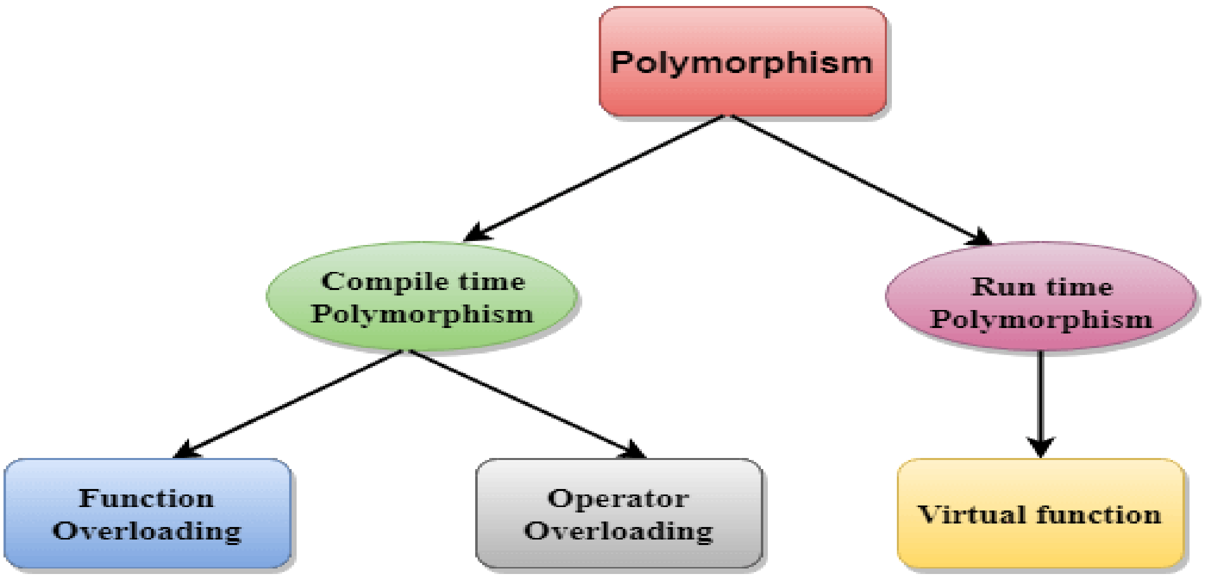
3. Inheritance :

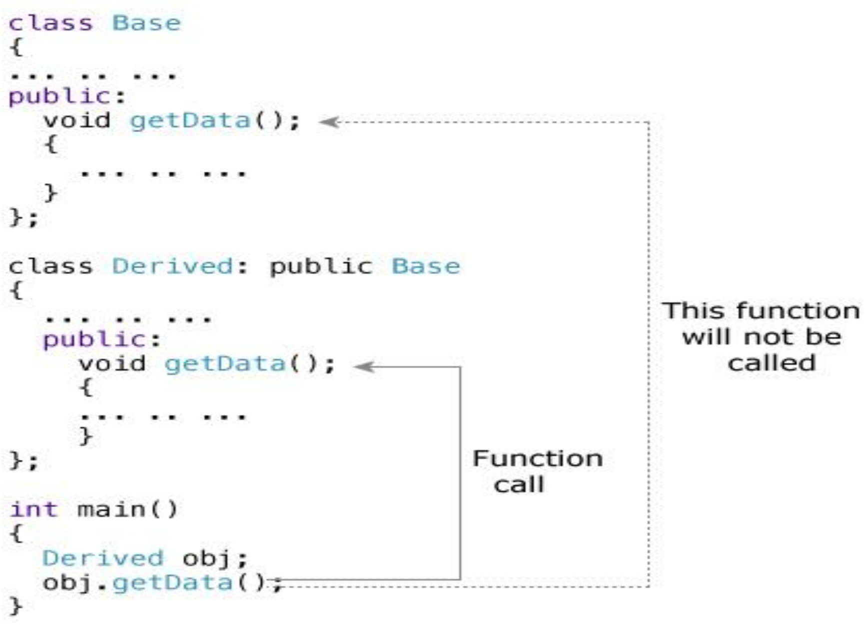
- Inheritance means access the properties and features of one class into another class.
- The class who is going to provide its features to another class will be called base class and the class who is using the properties and features of another class will be called derived class.



4. Polymorphism :

- Polymorphism means more than one function with same name, with different working. It can be static or dynamic.
- In static polymorphism memory will be allocated at compile time. In dynamic polymorphism memory will be allocated at runtime.
- Both function overloading and operator overloading are an examples of static polymorphism.
- Virtual function is an example of dynamic polymorphism.





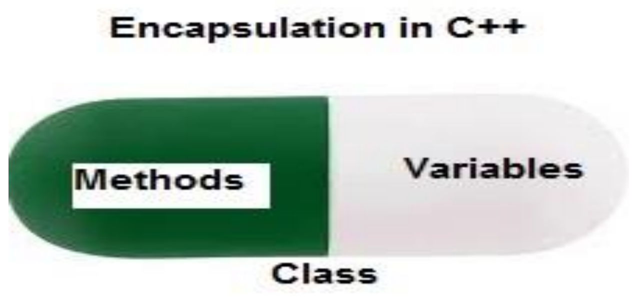
```
class Base
{
... .. ...
public:
   void getData();
   {
      ... .. ...
   }
};

class Derived: public Base
{
   ... .. ...
   public:
      void getData();
      {
      ... .. ...
      }
};

int main()
{
   Derived obj;
   obj.getData();
}
```

5. Data Abstraction :

- The basic idea of data abstraction is to visible only the necessary information, unnecessary information will be hidden from the outside world.
- This can be done by making class members as private members of class.
- Private members can be accessed only within the same class where they are declared.

6. Encapsulation :

- Encapsulation is a process of wrapping data members and member functions in a single unit called class.
- Using the method of encapsulation, the programmer cannot directly access the data.
- Data is only accessible through the object of the class.



Learning Outcome: (Summary)

❖ In this topic we learn meaning of class, object ,abstraction, polymorphism, inheritance, encapsulation.

1 word Question Answer

Sr No.	Question	Answer
1.	________ are an expanded version of structures. Structure can contain multiple variables.	Class
2.	____________ is used to access class members.	Object
3.	___________ access the properties and features of one class into another class.	Inheritance
4.	________ more than one function with same name, with different working. It can be static or dynamic.	Polymorphism
5.	_______________is to visible only the necessary information, unnecessary	data abstraction

	information will be hidden from the outside world.	
6.	_____________ is a process of wrapping data members and member functions in a single unit called class	Encapsulation

Topic: Benefits of OOP's

Trailer: **(Only for Understanding)**

- How to reduce line of code
- What is benefits of object oriented programming?

Details: **(for Exam Content)**

- Through inheritance, we can eliminate redundant code and extend the use of existing classes which is not possible in procedure oriented approach.
- We can build programs from the standard working modules that communicate with one another, rather than having to start writing the code from scratch which happens procedure oriented approach. This leads to saving of development time and higher productivity.
- The principle of data hiding helps the programmer to build secure programs that cannot be invaded by code in other parts of the program.
- It is possible to have multiple instances of object to co-exist without any interference.
- It is possible to map objects in the problem domain to those in the program.
- It is easy to partition the work in a project based on objects.
- The data-centered design approach enables us to capture more details of a model in implementable from.
- Object oriented systems can be easily upgraded from small to large systems.
- Message passing techniques for communication between objects makes the interface descriptions with external systems much simpler.
- Software complexity can be easily managed.

Topic :Object Oriented vs Procedure Oriented Programming

Procedure Oriented Programming	Object Oriented Programming
In POP, program is divided into small parts called functions.	OOP, program is divided into parts called objects.
In POP, Importance is not given to data but to functions as well as sequence of actions to be done.	In OOP, Importance is given to the data rather than procedures or functions because it works as a real world.
POP follows Top Down approach.	OOP follows Bottom Up approach.
POP does not have any access specifier.	OOP has access specifiers named Public, Private, Protected, etc.
In POP, Data can move freely from function to function in the system.	In OOP, objects can move and communicate with each other through member functions.
To add new data and function in POP is not so easy.	OOP provides an easy way to add new data and function.
In POP, Most function uses Global data for sharing that can be accessed freely from function to function.	In OOP, data can not move easily from function to function,it can be kept public or private so we can control the access of data.
POP does not have any proper way for hiding data so it is less secure.	OOP provides Data Hiding so provides more security.
In POP, Overloading is not possible.	In OOP, overloading is possible in the form of Function Overloading and Operator Overloading.
Example of POP are : C, VB, FORTRAN, Pascal.	Example of OOP are : C++, JAVA, VB.NET, C#.NET.

Topic: Application of OOP & C++

The promising areas for application of oop include:

1. Client-Server Systems
2. Object-Oriented Databases
3. Real-Time System Design
4. Simulation And Modelling System
5. AI Expert Systems
6. Hypertext, hypermedia and expertext
7. Neural networks and parallel programming
8. CAM/CAD systems.

➢ ***Real-World Applications of C++***

1. Games
2. Graphics User Interface
3. Web Browswes
4. Advance Computations and Graphics
5. Database Software
6. Operating Systems
7. Enter prose Software

Topic: What is c++? And explain features of C++.

Trailer: **(Only for Understanding)**

- In this topic we will learn when was the c++ language developed?
- How many versions of C++?

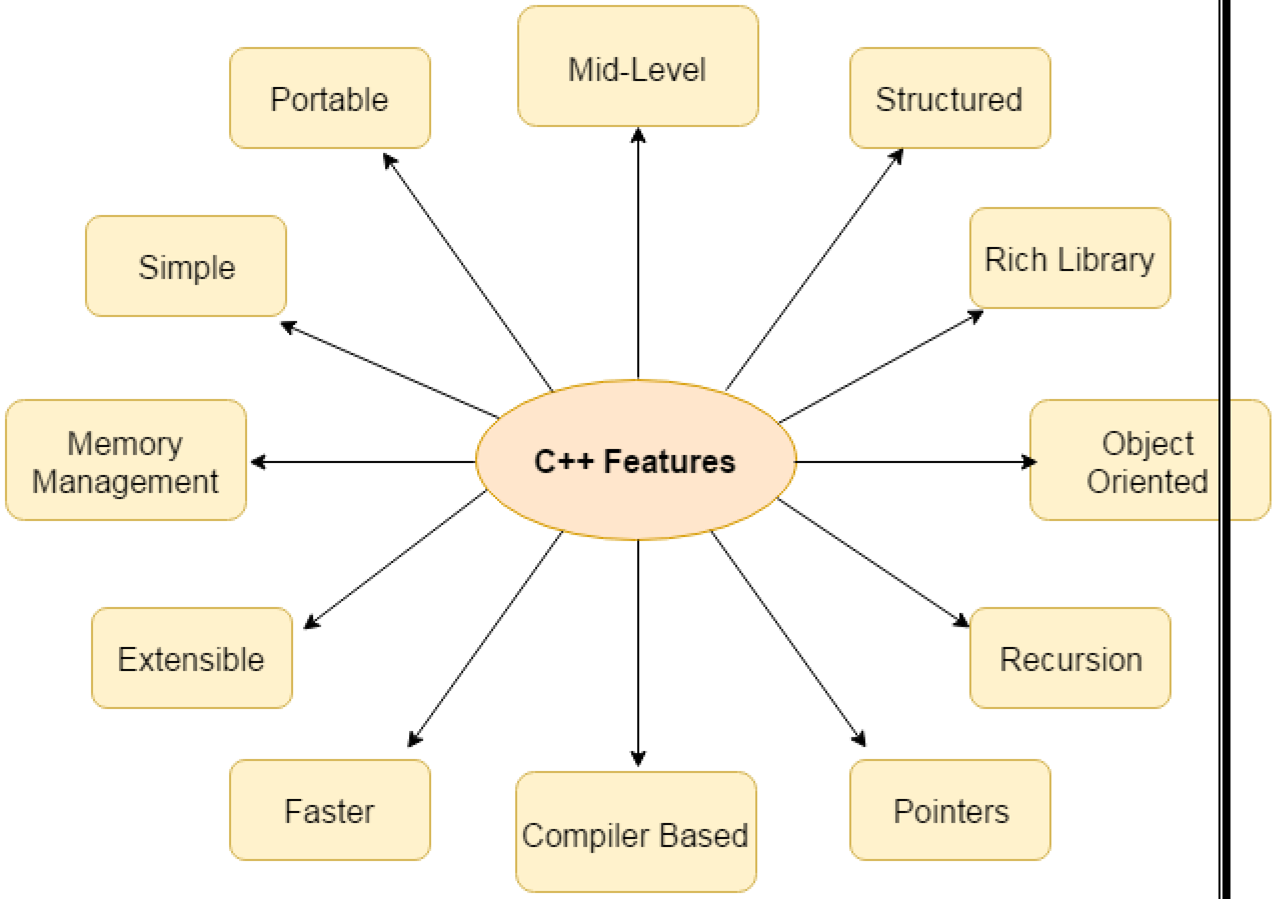
Details: **(for Exam Content)**

- Over the years, computer programs have become larger and more complex.
- Even though C is an excellent programming language, it has its limits. In C, once a program exceeds from thousands lines of code, it becomes so complex that it is difficult to maintain as a totality.
- The purpose of C++ is to allow this barrier to be broken. The essence of C++ is to allow the programmer to comprehend and manage larger, more complex programs.

- C++ began as an expanded version of C.
- The C++ were first invented by Bjarne Stroustrup in 1979 at Bell Laboratories in Murray Hill, New Jersey. This new language was initially called "C with Classes".

C++ Features

- C++ is object oriented programming language. It provides a lot of **features** that are given below.



1) Simple

- C++ is a simple language in the sense that it provides structured approach (to break the problem into parts), rich set of library functions, data types etc.

2) Machine Independent or Portable

- Unlike assembly language, c programs can be executed in many machines with little bit or no change. But it is not platform-independent.

3) Mid-level programming language

- C++ is also used to do low level programming. It is used to develop system applications such as kernel, driver etc. It also supports the feature of high level language. That is why it is known as mid-level language.

4) Structured programming language

- C++ is a structured programming language in the sense that we can break the program into parts using functions. So, it is easy to understand and modify.

5) Rich Library

- C++ provides a lot of inbuilt functions that makes the development fast.

6) Memory Management

- It supports the feature of dynamic memory allocation. In C++ language, we can free the allocated memory at any time by calling the free() function.

7) Speed

- The compilation and execution time of C++ language is fast.

8) Pointer

- C++ provides the feature of pointers. We can directly interact with the memory by using the pointers. We can use pointers for memory, structures, functions, array etc.

9) Recursion

- In C++, we can call the function within the function. It provides code reusability for every function.

10) Extensible

- C++ language is extensible because it can easily adopt new features.

11) Object Oriented

C++ is object oriented programming language. OOPs makes development and maintenance easier where as in Procedure-oriented programming language it is not easy to manage if code grows as project size grows.

12) Compiler based

C++ is a compiler based programming language, it means without compilation no C++ program can be executed. First we need to compile our program using compiler and then we can execute our program.

1 word Question Answer

Sr No.	Question	Answer
1.	C++ developed at	Bell Laboratories
2.	C++ was developed in which year?	1979
3.	C++ was developed by?	Bjarne Stroustrup
4.	When c++ developed - C++ was initially called?	C with Classes

Topic: Input/Output operator

Trailer: **(Only for Understanding)**

- What is input operator? & what is the name of input operator?
- What is output operator? & what is the name of output operator?
- How to take input from user?

Details: **(for Exam Content)**

- In C++, input and output (I/O) operators are used to take input and display output.
- The operator used for taking the input is known as the extraction or get from operator (>>), while the operator used for displaying the output is known as the insertion or put to operator (<<).
- **Input Operator**
- The input operator, commonly known as the extraction operator (>>), is used with the standard input stream, cin. As stated earlier, cin treats data as a stream of characters. These characters flow from cin to the program through the input operator. The input operator works on two operands, namely, the c in stream on its left and a variable on its right.

Thus, the input operator takes (extracts) the value through cin and stores it in the variable.

- To understand the concept of an input operator, consider this example.

```
int main ()
{
int a;
cin>>a;
a = a+1;
return 0;
}
```

- In this example, the statement cin>> a takes an input from the user and stores it in the variable a.

- **Output Operator**

- The output operator, commonly known as the insertion operator (<<), is used. The standard output stream cout Like cin, cout also treats data as a stream of characters. These characters flow from the program to cout through the output operator. The output operator works on two operands, namely, the cout stream on its left and the expression to be displayed on its right. The output operator directs (inserts) the value to cout.
- To understand the concept of output operator, consider this example.
- #include<iostream>
- int main ()
- {
- int a;
- cin>>a;
- a=a+1;
- cout<<a;
- return 0;
- }

- This example is similar to Example 1. The only difference is that the value of the variable a is displayed through the instruction cout << a .

Learning Outcome: **(Summary)**

- ❖ In this topic we learn input and output operator. Program of c++ for print the output of screen & take input from user.

1 word Question Answer

Sr No.	Question	Answer
1.	Give the Name & Symbol of input operator.	Name: expression Symbol: >>
2.	Give the Name & Symbol of Output operator.	Name: insertion Symbol: <<
3.	Which object is used for input operator.	Cin – standard input stream
4.	Which object is used for output operator.	Cout - standard output stream

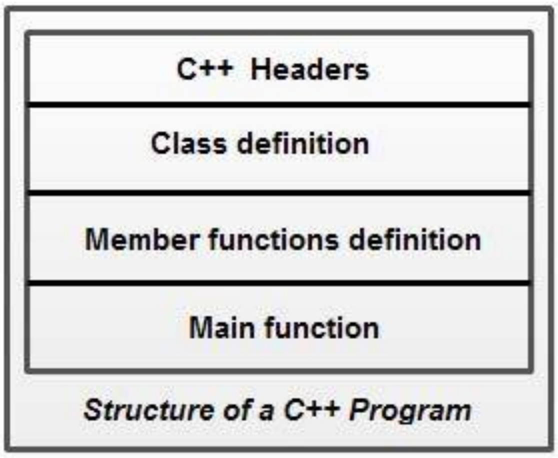
Topic: Structure of a C+ + Program

Trailer: **(Only for Understanding)**

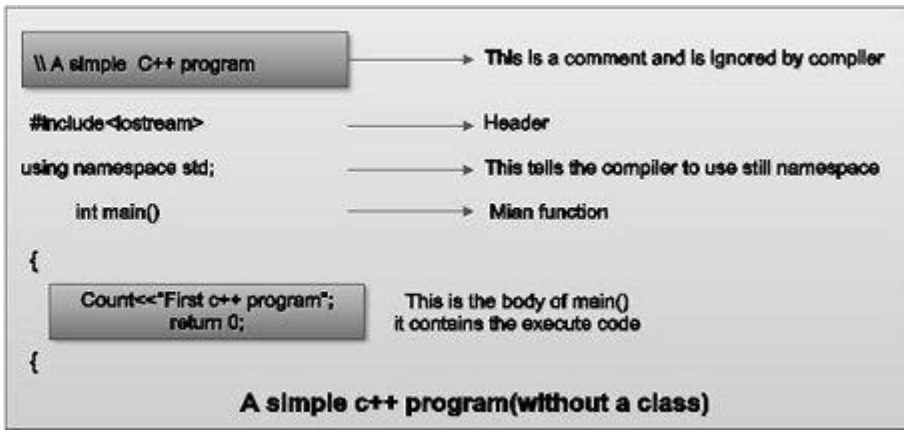
- What is the structure of C++?
- Which flow follow for run the C++ program?
- How many block in structure of C++ program?

Details: **(for Exam Content)**

- Programs are a sequence of instructions or statements. These statements form the structure of a C++ program. C++ program structure is divided into various sections, namely, headers, class definition, member functions definitions and main function.



- Note that C++ provides the flexibility of writing a program with or without a class and its member functions definitions. A simple C++ program (without a class) includes comments, headers, namespace, main() and input/output statements.
- Comments are a vital element of a program that is used to increase the readability of a program and to describe its functioning. Comments are not executable statements and hence, do not increase the size of a file.



```
\\ A simple  C++ program
#include<iostream>
using namespace std;
      int main()
{
    Count<<"First c++ program";
    return 0;
{
```

This is a comment and is ignored by compiler
Header
This tells the compiler to use still namespace
Mian function
This is the body of main()
it contains the execute code
A simple c++ program(without a class)

- C++ supports two comment styles: single line comment and multiline comment. Single line comments are used to define line-by-line descriptions. Double slash (//) is used to represent single line comments. To understand the concept of single line comment, consider this statement.
- / / An example to demonstrate single line comment It can also be written as
- / / An example to demonstrate
- / / single line comment

- Multiline comments are used to define multiple lines descriptions and are represented as / * * /. For example, consider this statement.

- /* An example to demonstrate
- multiline comment */
- Generally, multiline comments are not used in C++ as they require more space on the line. However, they are useful within the program statements where single line comments cannot be used.
- **Headers**: Generally, a program includes various programming elements like built-in functions, classes, keywords, constants, operators, etc., that are already defined in the standard C++ library. In order to use such pre-defined elements in a program, an appropriate header must be included in the program. The standard headers contain the information like prototype, definition and return type of library functions, data type of constants, etc. As a result, programmers do not need to explicitly declare (or define) the predefined programming elements.
- Standard headers are specified in a program through the preprocessor directive" #include. In Figure, the iostream header is used. When the compiler processes the instruction #inc1ude<iostream>, it includes the contents of iostream in the program. This enables the programmer to use standard input, output and error facilities that are provided only through the standard streams defined in <iostream>. These standard streams process data as a stream of characters, that is, data is read and displayed in a continuous flow. The standard streams defined in <iostream> are listed here.
- cin (pronounced "see in") : It is the standard input stream that is associated with the standard input device (keyboard) and is used to take the input from users.
- cout (pronounced "see out") : It is the standard output stream that is associated with the standard output device (monitor) and is used to display the output to users.
- **Namespace:** Since its creation, C++ has gone through many changes by the C++ Standards Committee. One of the new features added to this language is namespace. A namespace permits grouping of various entities like classes, objects, functions and various C++ tokens, etc.,

under a single name. Different users can create separate namespaces and thus can use similar names of the entities. This avoids compile-time error that may exist due to identical-name conflicts.

- The C++ Standards Committee has rearranged the entities of the standard library under a namespace called std. In Figure, the statement using namespace std informs the compiler to include all the entities present in the namespace std. The entities of a namespace can be accessed in different ways which are listed here.

- By specifying the using directive
- **using namespace std;**
- **cout<<"Hello World";**
- By specifying the full member name
- **std: :cout<<"Hello World";**
- By specifying the using declaration
- **using std:: cout;**
- **cout<<"Hello World";**

- As soon as the new-style header is included, its contents are included in the std namespace. Thus, all the modern C++ compilers support these statements.
- **#include<iostream>**
- **using namespace std;**
- However, some old compilers may not support these statements. In that case, the statements are replaced by this single statement.
- **#include<iostream.h>**

- **Main Function**: The main () is a startup function that starts the execution of a c++ program. All C++ statements that need to be executed are written within main ( ). The compiler executes all the instructions written within the opening and closing curly braces' {}' that enclose the body of main ( ). Once all the instructions in main () are executed, the control passes out of main ( ), terminating the entire program and returning a value to the operating system.

- By default, main () in C++ returns an int value to the operating system. Therefore, main () should end with the return 0 statement. A return value zero indicates success and a non-zero value indicates failure or error.

Learning Outcome: **(Summary)**

❖ In this topic we learn comments, namespace and main function.

1 word Question Answer

Sr No.	Question	Answer
1.	C++ program structure is divided into How many sections?	Four
2.	C++ supports How many types of comment?	Two
3.	Single line comment specified by?	//
4.	Multiline comment specified by?	/* */
5.	A ___________ permits grouping of various entities like classes, objects, functions and various C++ tokens.	Namespace
6.	C++ Starts execution of program from which function?	Main()

Topic: What is Namespace?

Trailer: **(Only for Understanding)**

- What is Namespace? How to include **namespace**?
-

Details: **(for Exam Content)**

- Consider following C++ program.
- // A program to demonstrate need of namespace

int main()

```
{
   int value;
   value = 0;
   double value; // Error here
   value = 0.0;
}
```

- Output :
- Compiler Error:
- 'value' has a previous declaration as 'int value'
- In each scope, a name can only represent one entity. So, there cannot be two variables with the same name in the same scope. Using namespaces, we can create two variables or member functions having the same name.

```
// Here we can see that more than one variables
// are being used without reporting any error.
// That is because they are declared in the
// different namespaces and scopes.
#include <iostream>
using namespace std;
 // Variable created inside namespace
namespace first
{
   int val = 500;
}
 // Global variable
int val = 100;
int main()
{
   // Local variable
   int val = 200;
     // These variables can be accessed from
   // outside the namespace using the scope
   // operator ::
   cout << first::val << '\n';
     return 0;
}
```

- Output:
- 500

- Definition and Creation:
- Namespaces allow us to group named entities that otherwise would have global scope into limited scopes, giving them namespace scope. This allows organizing the elements of programs into different logical scopes referred to by names.
- Namespace is a feature added in C++ and not present in C.
- A namespace is a declarative section that provides a scope to the identifiers (names of the types, function, variables etc) inside it.
- Multiple namespace blocks with the same name are allowed. All declarations within those blocks are declared in the named scope.

Learning Outcome: We learn namespace allow group name entities which we can use in any program of c++ and easily use the namespace code.

1 word Question Answer

Sr No.	Question	Answer
1.	Which operator is used to signify the namespace?	scope operator
2.	What is the use of Namespace?	To structure a program into logical units
3.	What is the general syntax for accessing the namespace variable?	namespace::operator

Topic: Tokens in C++

Trailer: **(Only for Understanding)**

- we will learn what is tokens and how we can use in the program.
- A small unit which is usefull in program is a tokens like variable name, keywords, data types etc.

Details: **(for Exam Content)**

- Tokens is the smallest individual unit in C++ language. In fact, every unit that makes a sentence in C++ is a Token. C++ has ten types of Tokens as given below.
  1) Keywords
  2) Identifiers
  3) Basic data types
  4) User-defined types
  5) Derived data types
  6) Symbolic constants
  7) Type compatibility
  8) Declaration of variables
  9) Dynamic initialization of variables
  10) Reference variables

1) Keywords:

- Keywords are those words who has special meaning for compiler. We can't use keywords as variable name.
- C++ has 32 Keywords as follows:

Keywords			
Auto	Double	int	struct
Break	Else	long	switch
Case	enum	register	typedef
Char	extern	return	union
Const	float	short	unsigned
Continue	For	signed	void

Default	Goto	sizeof	volatile
Do	If	static	while

2) Identifiers

- Identifiers are fundamental building blocks of a program and are used as the general terminology for the names given to different parts of program viz. variables, functions array,structures etc.

3) Basic data types

Type	Typical Bit Width	Typical Range
Char	1byte	-127 to 127 or 0 to 255
unsigned char	1byte	0 to 255
signed char	1byte	-127 to 127
Int	4bytes	-2147483648 to 2147483647
unsigned int	4bytes	0 to 4294967295
signed int	4bytes	-2147483648 to 2147483647
short int	2bytes	-32768 to 32767
unsigned short int	Range	0 to 65,535
signed short int	Range	-32768 to 32767
long int	4bytes	-2,147,483,648 to 2,147,483,647
signed long int	4bytes	same as long int
unsigned long int	4bytes	0 to 4,294,967,295
Float	4bytes	+/- 3.4e +/- 38 (~7 digits)
Double	8bytes	+/- 1.7e +/- 308 (~15 digits)
long double	8bytes	+/- 1.7e +/- 308 (~15 digits)

4) User-defined types

- The data types that are defined by the user are called the derived datatype or user-defined derived data type.
  These types include:

1.) Class

2.) Structure

3.) Union

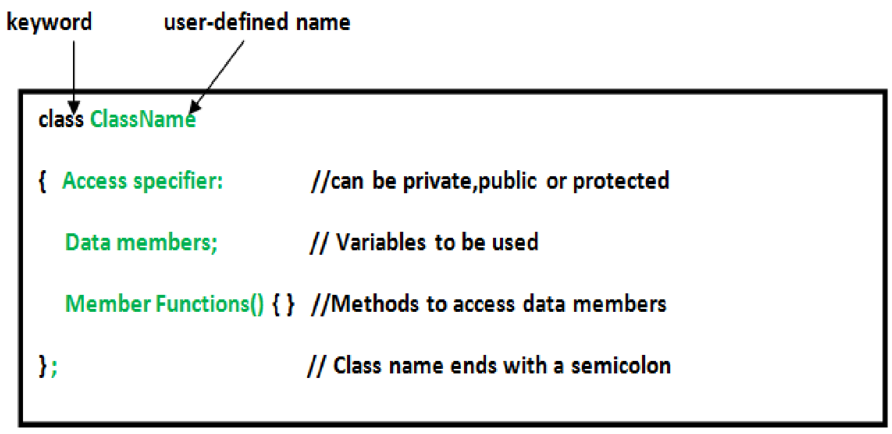
4.) Enumeration

5.) Typedef defined DataType

Class:

- The building block of C++ that leads to Object-Oriented programming is a Class. It is a user-defined data type, which holds its own data members and member functions, which can be accessed and used by creating an instance of that class. A class is like a blueprint for an object.

- **Syntax**:



```
class ClassName
{  Access specifier:          //can be private,public or protected
   Data members;              // Variables to be used
   Member Functions() { }     //Methods to access data members
};                            // Class name ends with a semicolon
```

Srtucture:

- A structure is a user defined data type in C/C++. A structure creates a data type that can be used to group items of possibly different types into a single type.

Syntax:

```
struct address {
   char name[50];
   char street[100];
   char city[50];
```

```
    char state[20];
    int pin;
};
```

Union:

- Like Structures, union is a user defined data type. In union, all members share the same memory location. For example in the following C program, both x and y share the same location. If we change x, we can see the changes being reflected in y.

Enumeration:

- Enumeration (or enum) is a user defined data type in C. It is mainly used to assign names to integral constants, the names make a program easy to read and maintain.

- **Syntax:**
- enum State {Working = 1, Failed = 0};

Typedef :

- C++ allows you to define explicitly new data type names by using the keyword typedef. Using typedef does not actually create a new data class, rather it defines a name for an existing type. This can increase the portability(the ability of a program to be used across different types of machines; i.e., mini, mainframe, micro, etc; without much changes into the code)of a program as only the typedef statements would have to be changed. Using typedef one can also aid in self-documenting code by allowing descriptive names for the standard data types.
- **Syntax:**

  typedef type name;
- where *type* is any C++ data type and *name* is the new name for this data type.
  This defines another name for the standard *type* of C++.
- **Example:**

```
// C++ program to demonstrate typedef
#include <iostream>
using namespace std;

// After this line BYTE can be used
// in place of unsigned char
typedef unsigned char BYTE;

int main()
```

```
{
   BYTE b1, b2;
   b1 = 'c';
   cout << " " << b1;
   return 0;
}
```

- **Output:**

```
C
```

5) Derived data types

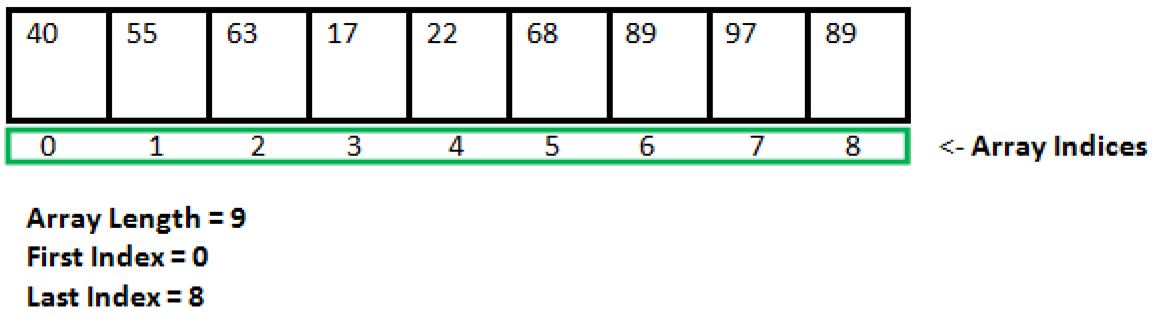
- The data-types that are derived from the primitive or built-in datatypes are referred to as Derived Data Types. These can be of four types namely:
  1.) Function
  2.) Array
  3.) Pointers
- Let's briefly understand each of the following derived datatypes:

1. **Function:**

- A function is a block of code or program-segment that is defined to perform a specific well-defined task. A function is generally defined to save the user from writing the same lines of code again and again for the same input. All the lines of code are put together inside a single function and this can be called anywhere required. main() is a default function that is defined in every program of C++.
- **Syntax:**
  FunctionType FunctionName(parameters)

2. Array:

- An array is a collection of items stored at continuous memory locations. The idea of array is to represent many instances in one variable.



Array Length = 9
First Index = 0
Last Index = 8

- **Syntax:**

DataType ArrayName[size_of_array];

3. Pointers:

- Pointers are symbolic representation of addresses. They enable programs to simulate call-by-reference as well as to create and manipulate dynamic data structures. It's general declaration in C/C++ has the format:
- **Syntax:**
  datatype *var_name;

6) Symbolic constants

- Any value declared as a const can not be modified by the program in any way.
- Syntax: const variable-name=value;

```
Ex: #include<iostream.h>
    #include<conio.h>
    #include<iomanip.h>
   Void main()
   {
        Const i=10;
        Cout<<i;
        I=i+1; //generate an error because of constant value can't modify
        getch();
   }
```

7) Type compatibility

- C++ is very strict with regard to type compatibility as compared to C. Type compatibility is very close to automatic or implicit type conversion.
- The type compatibility is being able to use two types together without modification and being able to substitute one for the other without modification.
- **Implicit Type Conversion** Also known as 'automatic type conversion'.
- Done by the compiler on its own, without any external trigger from the user.
- Generally takes place when in an expression more than one data type is present. In such condition type conversion (type promotion) takes place to avoid lose of data.

- All the data types of the variables are upgraded to the data type of the variable with largest data type.
- bool -> char -> short int -> int ->
- unsigned int -> long -> unsigned ->
- long long -> float -> double -> long double
- It is possible for implicit conversions to lose information, signs can be lost (when signed is implicitly converted to unsigned), and overflow can occur (when long long is implicitly converted to float).

  Example of Type Implicit Conversion:

```
#include <iostream>
using namespace std;

int main()
{
   int x = 10; // integer x
   char y = 'a'; // character c

   // y implicitly converted to int. ASCII
   // value of 'a' is 97
   x = x + y;

   // x is implicitly converted to float
   float z = x + 1.0;

   cout << "x = " << x << endl
       << "y = " << y << endl
       << "z = " << z << endl;

   return 0;
}
```

  Output:

```
x = 107
y = a
z = 108
```

- **Explicit Type Conversion**:
- This process is also called type casting and it is user-defined. Here the user can typecast the result to make it of a particular data type.
- In C++, it can be done by two ways:
- Converting by assignment: This is done by explicitly defining the

required type in front of the expression in parenthesis. This can be also considered as forceful casting.

- Syntax:
- (type) expression

Example:

```
// C++ program to demonstrate
// explicit type casting

#include <iostream>
using namespace std;

int main()
{
   double x = 1.2;

   // Explicit conversion from double to int
   int sum = (int)x + 1;

   cout << "Sum = " << sum;

   return 0;
}
```

Output:

```
Sum = 2
```

- **Advantages of Type Conversion:**
- This is done to take advantage of certain features of type hierarchies or type representations.
- It helps to compute expressions containing variables of different data types.

8) Declaration of variables

- Variables are used to store values. variable name is the name of memory location where value is stored. It must be alphanumeric, only underscore is allowed in a variable name. It is composed of letters, digits and only underscore. It must begin with alphabet or underscore. It can not be begin with numeric.
- Declaration of Variable
- Declaration will allocate memory for specified variable with garbage

value.

- Syntax :

  ```
  Data-Type Variable-name;
  ```
- Examples :

  ```
  int a;
  float b;
  char c;
  ```
- Initialization means assigning value to declared variable. Every value will overwrite the previous value.
- Examples :

  ```
  a = 10;
  b = 4.5;
  c = 'a';
  ```
- Character value must be enclosed with single quotes.

  ```
  a = 4.5;
  ```
- if we assign decimal value to integer variable, it will accept only integer portion of value. In the above example variable a will accept 4 only.

9) Dynamic initialization of variables

- Here, the variable is assigned a value at the run time. The value of this variable can be altered every time the program is being run.

10) Reference variables

- Think of a variable name as a label attached to the variable's location in memory. You can then think of a reference as a second label attached to that memory location. Therefore, you can access the contents of the variable through either the original variable name or the reference. For example, suppose we have the following example −

  ```
  int i = 17;
  ```
- We can declare reference variables for i as follows.

  ```
  int& r = i;
  ```
- Read the & in these declarations as reference. Thus, read the first declaration as "r is an integer reference initialized to i" and read the second declaration as "s is a double reference initialized to d."

Learning Outcome: **(Summary)**

- keywords is the word with the special meaning.
- Identifier is identify the variable, structure, function etc.

- Data types is used for which type of data we can store in variable.
- User define data type means enumeration , typedef.
- Derived data type means function, array, pointers.
- Type conversion means change the data type of variable.
- Dynamic initialization means we can store the data at run time.

1 word Question Answer

Sr No.	Question	Answer
1.	C++ provides various types of …………………… that includes keywords, identifiers, constants, strings and operators.	Tokens
2.	………………… refer to the names of variables, functions, arrays, classes etc. created by programmer.	Identifiers
3.	…………………. are explicitly reserved identifiers and cannot be used as names for the program variables or other user defined program elements.	Keywords
4.	In C++, …………………. refer to fixed values that do not change during the execution of a program.	Constants
5.	C++ provides an additional use of ……………………., for declaration of generic pointers	Void
6.	In the case of ……………………. in C++, we can not modify the address that the pointer is initialized.	constant pointer
7.	……………….. are widely used in C++ for memory management and to achieve polymorphism.	Pointers
8.	C++ permits initialization of the variables at run time which is referred to as ……………… initialization.	Dynamic

9.	…………………… used in C++ provides an alias (alternative name) for a previously defined variables.	Reference
10.	A reference variable must be initialized at the time of ……………..	declaration

Topic: What is Operators? Explain types of Operators.

Trailer: **(Only for Understanding)**

- Operator means which is operate something.
- For ex: we want to perform addition then we have to use + "Plus" operator which operate addition.
- We have many types of operator which provide different different facility for operations.

Details: **(for Exam Content)**

- An operator is a symbol that tells the compiler to perform specific mathematical or logical calculations on operands(variables).
- **Scope Resolution Operator**
- C++ is also a Block-Structured Language. The Scope of a variable extends from the point of its Declaration till the end of the code block, containing the declarations. A Variable declared inside a code block is said to be local to that code block. ::(Scope Resolution Operator) Operator allows access to the global version of a Variable.

  1) Global Scope Resolution Operator :

  Let us see an program Example illustrating the use of Scope Resolution Operator

  (::) used with the Gloal Variable is given below :

```
#include <iostream>
using namespace std;

int Sum = 10;
int main ()
{
   int Sum = 50;
   cout << "Value of local Sum variable in the main function = " << Sum ;
   cout << "\nValue of Sum using Scope Resolution Operator = " << ::Sum ;
   return 0 ;
}
```

2) Class Scope Resolution Operator :

In the below example we are using Scope Resolution Operator to define the class functions outside the class :

```
#include <iostream>
using namespace std;

class Scope_Resolution_Operator
{
   public :
            void Display();
};
void Scope_Resolution_Operator :: Display()
{
   cout << "We are in the Display Function." ;
}
int main ()
{
   Scope_Resolution_Operator Scopeobj;
   Scopeobj.Display();
   return 0 ;
}
```

- **Member Dereferencing Operators**
- Once a class is defined, its members can be accessed using two Operators :-
  1) (.) Dot Operator, And
  2) (->)Arrow Operator

  While (.) Dot Operator takes class or struct type Variable as Operand, (-

>) Arrow Operator takes a Pointer or Reference Variable as its Operand.

Let us see an simple example of using (.) Dot operator to make access to the members defination of structure Student :

```
#include <iostream>
using namespace std;

struct Student{
   string name , state;
   int rollno , houseno;
};
int main ()
{
   Student amit;
   amit.name = "Amit" ;
   amit.rollno = 26 ;
   amit.houseno = 24 ;
   amit.state = "Delhi" ;

      cout << "Name of the Student is = " << amit.name ;
   cout << "\nRoll no of the Student is = " << amit.rollno ;
   cout << "\nHouse of the Student is = " << amit.houseno ;
   cout << "\nState of the Student is = " << amit.state ;
   return 0 ;
}
```

- *Let us see an simple example of using (->) Arrow operator to make access to the members defination of structure Student :*

```
#include <iostream>
using namespace std;

struct Student{
   string name , state;
   int rollno , houseno;
};
int main ()
{
   Student amit;
```

```
    Student *amitptr;
    // Here, *amitptr points to the object of Structure Student i.e amit

    amitptr = &amit;
    // Here, amitptr is equal to the address of the object amit

    amitptr -> name = "Amit" ;
    amitptr -> rollno = 26 ;
    amitptr -> houseno = 24 ;
    amitptr -> state = "Delhi" ;

    cout << "Name of the Student is = " << amit.name ;
    cout << "\nRoll no of the Student is = " << amit.rollno ;
    cout << "\nHouse of the Student is = " << amit.houseno ;
    cout << "\nState of the Student is = " << amit.state ;
    return 0 ;
}
```

- **Memory Management Operators**

  C++ also define two Unary Operators :-
  1) New, and
  2) Delete.

  - New and Delete Operators performs the task of allocating and freeing the memory. Since, these Operators manipulates memory on the Free Store, They are also known as **Free Store Operators.**
  - A Data Object created inside a block with New Operator will remain in existence Until it is explicitly destroyed using Delete Operator.
  - When An Object Is No Longer Needed, It Is Destroyed To Release The Memory Space For Reuse by another variables by using Delete Operator.
  - *The General Syntax of using New Operator is :*
    Pointer_variable = New Data_type(Value);
    Let us see an simple example of using new operator :

```
#include <iostream>
```

```
using namespace std;

int main ()
{
        int *pointer;
        pointer = new int;
      // Now new operator dynamically allocates the
memory space to the pointer variable

        *pointer = 12;

        cout << *pointer ;
        return 0 ;
}
```

- The General Syntax of using Delete Operator is :
  Delete Pointer_variable;

  Let us see an simple example of using delete operator :

```
#include <iostream>
using namespace std;

int main ()
{
        int *pointer;
        pointer = new int;
        *pointer = 12;

        cout << *pointer ;
        // This returns the value 12 instead of the address
value of pointer

        delete pointer;
        // delete operator deallocates the memory space of
pointer and now pointer variable does not contain any
value

        cout << "\n" << *pointer ;
        // Now this will return the memory address of the
pointer

        return 0 ;
}
```

- These operators are used to format the data display. Some of the manipulators are as follows :

- **1) endl manipulator -**
  - endl manipulator when used causes a linefeed. endl is a manipulator which is manipulating the actual nature of program by printing the next specified code in the next new line. endl does not get any parameters.
    For Example :-

```
#include<iostream>
using namespace std;

int main ()
{
        cout << "This is text of first line" << endl;
        cout << "This is text display in second line";
        // If you try to print this without endl the both lines
will come in oneline.

        return 0;
}
```

2. setw() manipulator -

- setw(value) is a manipulator which sets the width before the character or can say that gives the number of space set in the parameter before the text is displayed.
- We can also fill the setted space by some character by the manipulator "setfill('character or value').
- This manipulator takes the parameter which is a int datatype. To use this manipulator you must add "#include<iomanip>".
- This manipulator will only allow one character after its declaration.

For Example :-

```
#include <iostream>
#include <iomanip>
using namespace std;

int main ()
{
        int a = 10;   // value to be putten in setw must be declared
in a variable first

        cout << setw(a) << "Setw" << endl;
        cout << setw(a) << setfill('#') << "Setfill";

        return 0;
}
```

- **3. hex manipulator -**
- hex is a manipulator which manipulates the written expression and print the hexadecimal value of that expression. hex manipulator also does not take any parameters.
  For Example :-

```
#include<iostream>
//#include <iomanip>
using namespace std;

int main ()
{

        cout << hex << 50 << endl;
        cout << hex << 100 << endl;
        cout << hex << 150 << endl;
        cout << hex << 200;
        // All values will be printed in their hexadecimal
values by hex manipulator

        return 0;
```

```
                    }
```

Learning Outcome: **(Summary)**

- ❖ In this topic we learn global scope resolution operation with this operator we can use the global data in any where in the program.
- ❖ We learn memory management operator which is manage the memory for storing our data.
- ❖ We learn manipulators which is used for designing purpose.

1 word Question Answer

Sr No.	Question	Answer
1.	Which operator is having the right to left associativity in the following?	Type cast
2.	Which operator is having the highest precedence?	Postfix
3.	What is this operator called ?:?	Conditional
4.	What is the use of dynamic_cast operator?	it converts virtual base class to derived class
5.	_____ is a manipulator which manipulates the written expression and print the hexadecimal value of that expression.	Hex
6.	________ is a manipulator which sets the width before the character or can say that gives the number of space set in the parameter before the text is displayed.	Setw
7.	_______ manipulator when used causes a linefeed.	Endl
8.	___________Opeartors performs the task of allocating and freeing the memory.	New and Delete

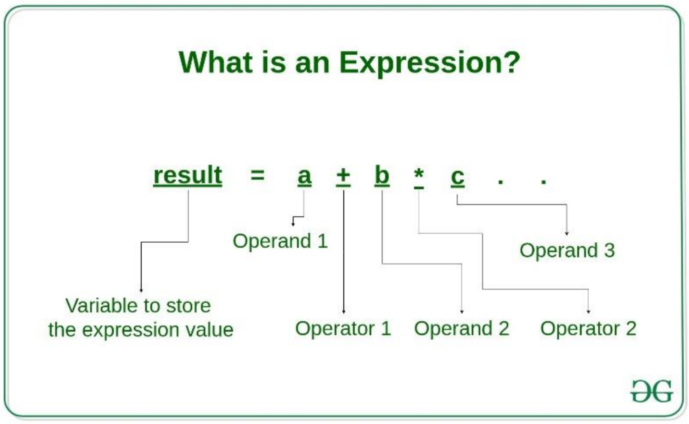
Topic: Expression and its Types

Trailer: **(Only for Understanding)**

- ❖ Expression means to express something for ex: res= A + B is express the addition and this will store the addition in the res variable.

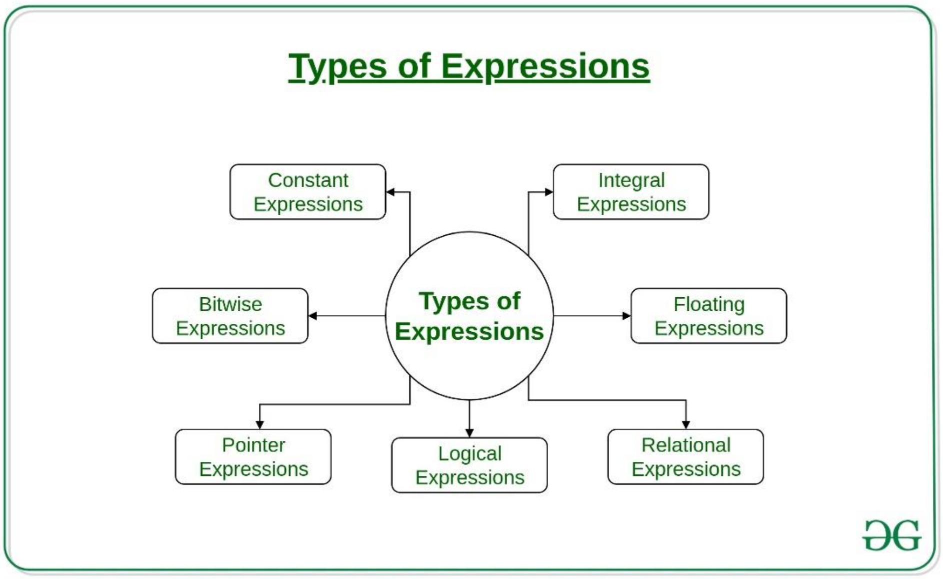
Details: **(for Exam Content)**

- Expression: An expression is a combination of operators, constants and variables. An expression may consist of one or more operands, and zero or more operators to produce a value.



Types of Expressions:

Expressions may be of the following types:



- **Constant expressions:** Constant Expressions consists of only constant values. A constant value is one that doesn't change.
  Examples:

  5, 10 + 5 / 6.0, 'x'

- **Integral expressions:** Integral Expressions are those which produce integer results after implementing all the automatic and explicit type conversions.
  Examples:

  x, x * y, x + int(5.0)

  where x and y are integer variables.

- **Floating expressions**: Float Expressions are which produce floating point results after implementing all the automatic and explicit type conversions.
  Examples:
  x + y, 10.75
  where x and y are floating point variables.
- **Relational expressions**: Relational Expressions yield results of type bool which takes a value true or false. When arithmetic expressions are used on either side of a relational operator, they will be evaluated first and then the results compared. Relational expressions are also known as

Boolean expressions.
Examples:
x <= y, x + y > 2

- **Logical expressions**: Logical Expressions combine two or more relational expressions and produces bool type results.
  Examples:
  x > y && x == 10, x == 10 || y == 5
- **Pointer expressions**: Pointer Expressions produce address values.
  Examples:
  &x, ptr, ptr++
  where x is a variable and ptr is a pointer.
- **Bitwise expressions**: Bitwise Expressions are used to manipulate data at bit level. They are basically used for testing or shifting bits.
  Examples:
  x << 3
  shifts three bit position to left
  y >> 1
  shifts one bit position to right.
  Shift operators are often used for multiplication and division by powers of two.

Learning Outcome: **(Summary)**

❖ In this topic we learn expression. An expression is a combination of operators, constants and variables.

Topic: Conditional Structure

- C++ has the following conditional statements:
  - Use if to specify a block of code to be executed, if a specified condition is true
  - Use else to specify a block of code to be executed, if the same condition is false
  - Use else if to specify a new condition to test, if the first condition is false
  - Use switch to specify many alternative blocks of code to be executed

The if Statement

- Use the if statement to specify a block of C++ code to be executed if a condition is true.

- Syntax

```
if (condition) {
  // block of code to be executed if the condition is true
}
```

The else Statement

- Use the else statement to specify a block of code to be executed if the condition is false.
- Syntax

```
if (condition) {
  // block of code to be executed if the condition is true
} else {
  // block of code to be executed if the condition is false
}
```

The else if Statement

- Use the else if statement to specify a new condition if the first condition is false.
- Syntax

```
if (condition1) {
  // block of code to be executed if condition1 is true
} else if (condition2) {
  // block of code to be executed if the condition1 is false and condition2
is true
} else {
  // block of code to be executed if the condition1 is false and condition2
is false
}
```

C++ Switch Statements

- Use the switch statement to select one of many code blocks to be executed.
- Syntax

```
switch(expression)
{
  case x:
    // code block
    break;
  case y:
    // code block
    break;
  default:
    // code block
  }
```

Topic: Looping Structure

Trailer: **(Only for Understanding)**

- ❖ If we want to execute some line of code for multiple time than we don't write this code for multiple time we can use the loop for that.

Details: **(for Exam Content)**

- A Loop is defined as a block of processing steps repeated a certain no of times.
- **For Loop:**
- The For loop statement is used to repeat a statement or a block of statement specified number of times.
- **Syntax:**

```
For(initialization; condition; updation)
{
                Body of loop
}
```

- According to the above syntax the initialization part is initialized the variable.
- The condition part tested the loop variable if the condition is true, the body of the loop get executed else the loop is terminated and the next statement after the loop executed.
- In the updation part increments or decrements the loop variable. After testing condition update part is executed.
- The condition is check again and the whole process is repeated till the condition will false.

- **While**
- The while loop is used when the no of iterations to be performed are not known in advance.
- The statement in the loop is executed if the test condition is true and the execution continuous as long as it remains true.
    - **Syntax:**

```
While(condition)
```

```
          {
                    Body of the loop
          }
```

- According to the above syntax first execute the condition, if condition is true execute the body of loop, else loop will terminate.

- **Do….While**
- Sometimes it is required to execute the body of the loop at least once even if the test expression execute false during the first iteration.
- This requires testing termination expression at the end of loop rather than beginning.
- For that do….while loop is used where the test condition is at the bottom of the loop.
- This means that the program always execute the statements at least once.
- **Syntax:**

```
Do
{
          Body of the loop
} while (condition);
```

- According to the syntax first execute the body of the loop and at last execute the condition and if condition is true then repeat body of loop once again, else control come out from the loop.

Learning Outcome: **(Summary)**

❖ With loop we can easily execute the block of code multiple time.

❖ While loop is entry control loop because this will check the condition first than after will execute the code.

❖ Do – while loop is exit control loop because this will execute the code first than after check the condition.

1 word Question Answer

Sr No.	Question	Answer
1.	How many loops are there in C++	3
2.	Give the correct syntax of for loop?	for(initialization;condition;

		increment/decrement)
3.	Give the name of entry control loop	While
4.	Give the name of exit control loop	Do…while

Topic: Explain the main Function.

Trailer: **(Only for Understanding)**

- ❖ This function is the main function for C++. And in this topic we will learn the syntax of the main function.

Details: **(for Exam Content)**

- The execution of each and every c++ program is start from the main() function. It is the entry point of a program execution.
- The general format of main() function is as follow.
- **Syntax:**
- Return type main([int argc, char *argv[ ],[char ** envp]])
- {
    - Body of the main function
- }
- According to the above format the **return type** of the main() function must be either void or int.
- Here, return type specifies the status of the program termination.
- The main() function can also takes arguments from command prompt to. It is known as command line arguments.
- **argc** specifies the total number at argument. It is the argument counter. Its value is always positive.
- **Argv** represents the argument vector(array). It holds pointer to the argument passed from the command line.
- **argv[]** is one kind of an array so it holds the data in following manner:
    - argv[0] = pointer to the name of the executable program.
    - Argv[1], argv[2]….. argv[n] = pointers to argument strings
- **envp** represents an environment parameter. It is optional.

    - **For example**

```
#include<iostream.h>
#include<conio.h>
int main(int argc, char *argv[])
```

```
{
        int i;
        cout<<endl<<"Total arguments="<<argc;
        cout<<endl<<"Program name is="<<argv[0];
        cout<<endl<<"Other Arguments are";
        for(i=1;i<argc;i++)
        {
                cout<<endl<<argv[i];
        }
        getch();
        return(0);
}
```

- run the above program from dos shell and enter following arguments:
  - C:\TC\BIN\SOURCE> ARGS.EXE I like c++ very much.

Learning Outcome: **(Summary)**

❖ With the main function we can pass the argument to the main function and first argument is the program name by default.

1 word Question Answer

Sr No.	Question	Answer
1.	How many arguments in main function?	3
2.	Give the syntax of main function.	Return type main([int argc, char *argv[ ],[char ** envp]])

Topic: Explain the call by reference.

Trailer: **(Only for Understanding)**

❖ Call by reference means we can call the function by its reference means address of variable.

Details: **(for Exam Content)**

- A reference as its name, is like alias.

- It refer to the same entity. A variable and its reference are tightly attached with each other.
- So, change in one it will also change in the other. When call any function by its reference any modifications made through the formal pointer parameter is also reflected in the actual parameter.
- It has functionality of pass-by-pointer and the syntax of call-by-value.
- In the function declaration parameter are to be received by reference must be preceded by the **& operator** and arguments or parameters pass same as call by value.
- However any modification in the variable in function body directly reflected to the actual parameter.
  - Ex.
    - Void swapref(int &x, int &y);
  - In above ex we can see that the reference of the variable will passed in the function parameters.
  - When we want to call this type of function we can call same as call by value functions. Like **swapref(x,y).**

Learning Outcome: **(Summary)**

- ❖ With the call by reference we can directly change the value of variable in the user define function because we use the reference of variable.
- ❖ In general case we can not change the value of variable permanently.

1 word Question Answer

Sr No.	Question	Answer
1.	In the call by value function declaration parameter are to be received by ___________.	reference
2.	In the call by value function which symbol is used for pass reference?	&

Topic: Explain the return by reference.

Trailer: **(Only for Understanding)**

❖ Return by reference means we return the reference means address of variable.

Details : **(for Exam Content)**

- We can pass a reference to a parameter same we can also return a reference from a function.
- A function that returns a reference variable is actually an alias for the referred variable.
- Here. A function can be called on the receiving side of an assignment.
- For ex.
    - Int & max(int &x,int &y)
- In above ex you can see that the return type of function is int reference.

Learning Outcome: **(Summary)**

❖ In this type of function we can return the reference means address of variable means our value of variable is permanently change.

1 word Question Answer

Sr No.	Question	**Answer**
1.	In return by reference function we have to return __________.	Reference

Topic: Explain the inline function.

Trailer: **(Only for Understanding)**

❖ Inline function means the code of function is in the line at the time of function calling.

Details: **(for Exam Content)**

- C++ provides an inline functions to reduce the function call overhead. Inline function is a function that is expanded in line when it is called. When the inline function is called whole code of the inline function gets inserted or substituted at the point of inline function call.

- When we call the function, control of program jumps to the code of the function, the CPU stores memory address of the instruction and copies arguments onto the stack, and finally transfers control to the specified function.
- CPU executes the function code, stores the return value in memory location or register and returns control to the calling function.
- If we call function 5 times, then the control of the program jumps to the set of instruction 5 times and perform same task to store argument in stock and return value to register again and again.
- C++ provides a solution to overcome such a problem. For that declare a function with the keyword inline. It is known as inline function.
- Here, the compiler does not create a one copy of real function, but copies the code where the function call have been made.
- If the function is called 5 times, the code of the function copied into the calling function each of the 5 times.
- It will improve the speed at the program but increase the size of executable program.
- The inline functions are defined as follow.
- Syntax:
    - Inline returntype functionname()
    - {
        - //function body
    - }
- According to above format the keyword inline is placed with function header at the time of function definition and also with function prototype.
- Some of the situation where inline function may not work are:
    - If a loop, switch or goto exists.
    - If returns statement exists but function not return a value.
    - If a function contains static variables.
    - If inline functions are recursive.

Learning Outcome: **(Summary)**

❖ Inline function is a function that is expanded in line when it is called. It will easy for the compiler.

1 word Question Answer

Sr No.	Question	Answer
1.	Name the function whose definition can be substituted at a place where its function call is made __________	inline function
2.	loop, switch or goto exists in inline function the function will work or not.	No
3.	If returns statement exists but function not return a value exists in inline function the function will work or not.	No
4.	If a function contains static variables in inline function the function will work or not.	No
5.	inline functions are recursive or not	no

Topic: Explain default Argument.

Trailer: **(Only for Understanding)**

❖ We can provide default values for function parameters. If a function with default arguments is called without passing arguments, then the default parameters are used.

Details: **(for Exam Content)**

- In c it a function is defined to receive 3 arguments, so whenever call a function need to pass value those 3 arguments to the function.
- It assigns a garbage value for argument for the last argument.
- In c++ functions have an ability to define default values for arguments that are not passed when the function is call.
- The default arguments can be specified by following the arguments is name with = default value in the function argument list.
- The default value must be specified from right-to-left specify a value to a particular argument in the middle of an argument list is not possible.
- A function can be declare with a default argument as follows:
  - Int sum(int x=5, int y=10, int z=15);

- Int sum(int x, int y=10, int z=15);
- Int sum(int x, int y, int z=15);

- Here, default arguments are specified from left to right.
- If specifies default argument value of any argument then also pass value for it, the new passed value is consider as a value of argument instead of default argument.

Learning Outcome: **(Summary)**

❖ Functions have an ability to define default values for arguments that are not passed when the function is call.

1 word Question Answer

Sr No.	Question	Answer
1.	the arguments is name with = default value in the which type of function argument list ?	Default argument function
2.	The default value must be specified from?	left-to-right
3.	If specifies default argument value of any argument then also pass value for it, which value is consider as a value of argument?	new passed value

Topic: Explain Const argument.

Trailer: **(Only for Understanding)**

❖ When you put "const" in front of a parameter, it means that it cannot be modified in the function.

Details: **(for Exam Content)**

- The keyword const specifies that the value of variable will not change throughout the program.
- If anyone attempt to after the value of variable defined with this qualifier an error can be created.
- A function can also take an argument as a const. which is specifies no any modification on the value.

Learning Outcome: **(Summary)**

❖ Const Arguments means value of arguments can not change.

1 word Question Answer

Sr No.	Question	Answer
1.	A function can also take an argument as a ________ which is specifies no any modification on the value.	const

Topic: Explain function overloading.

Trailer: **(Only for Understanding)**

❖ In c language we can not declare the multiple function with the same name but in c++ we can do this with Function overloading.

Details: **(for Exam Content)**

- Overloading refers to the use of a same thing for different purpose.
- It is the polymorphism feature of a object oriented concept.
- Function overloading or function polymorphism is a concept that allows multiple functions to use the same name with different number of argument and types of argument.
- In c language that can not possible to make any function with same name while in c++ it can be possible with function overloading or function name overloading.
- For ex:
  - Char * copy(char *);
  - int copy(int,int);
  - Float copy(int);
- Here, the above three function use the same function name copy() but work with different data types.
- Here, return type doesn't mean.
  - int copy(int,int);
  - Void copy(int,int);
- As shown in above two functions, it can not work,it will create ambiguity.

- Return type does not differentiate any function.
- When any function is called, the selection of the particular function is made from given forms at function and it done through the process of argument matching.
- The actual argument of the function call gets matched with the formal argument of each forms of the function.
- One of the following choice will occur when a call for the overloaded function is made.
- A match
- No match
- Ambiguous match

- Match
- Here, the compiler first tries to find exact match in which types of arguments are the same, and use that function.
- No match
- A no match occurs when the actual argument cannot be match with an argument of the defined function and will cause the error message.
- Ambiguous match
- If an exact match is not found. C++ compiler tries to match the arguments by two ways.
- A match through promotion.
- For ex:
  - Void ff(int);
  - Void ff(char *);
- Here, if function called as ff('a') . if is not exact match but 'a' is of type char.
- It is promoted to type int after no exact match is found.
- The other promotion such as.
- Float to double
- Enumeration to int
- Char to int
- A match by standard conversion.
- Here, compiler tries to use the built-in conversion to the actual argument and then uses the function whose match is unique.
- For ex:
- Void ff(char *c);
- Void ff(double d);

- Void ff(void);
- If any function called as
- ff(10); // Matches ff(double)
- ff("a");// match ff(char *)
- If the conversion have multiple matches, then the compiler will generate an error message.

  Learning Outcome: **(Summary)**
- With the function overloading we can use same function name multiple time with different argument list.

1 word Question Answer

Sr No.	Question	Answer
1.	____________ is a concept that allows multiple functions to use the same name with different number of argument and types of argument.	Function overloading or function polymorphism
2.	How many choice will occur when a call for the overloaded function is made?	3

Question Bank

1 Marks

1. What is Procedure Oriented Programming? (jan-2021)
2. What is C++? (jan-2021)
3. What is a reference variable? (jan-2021)
4. What is a symbolic constant? (jan-2021)
5. What is main function? (jan-2021)
6. What is namespace? (jan-2021)
7. What is derived data- type? (jan-2021)
8. What is Keywords? (dec-2018)
9. C++ is an "object oriented" programming language created by _____. (dec-2018)
10. Give name and symbol of input operator. (dec-2018)
11. Write how to declare user define function. (dec-2018)
12. C++ language was developed by ________. (oct-2017)
13. Define the term Data Encapsulation. (oct-2017)

14. ADT stands for _______. (oct-2017)
15. Define the term Data Abstraction. (oct-2017)

2 Marks

1. What is reference variable? Explain with example. (oct-2017)
2. What is inline function? Explain with example. (oct-2017)
3. Explain memory management operator. (dec-2018) (jan-2021)

3 Marks

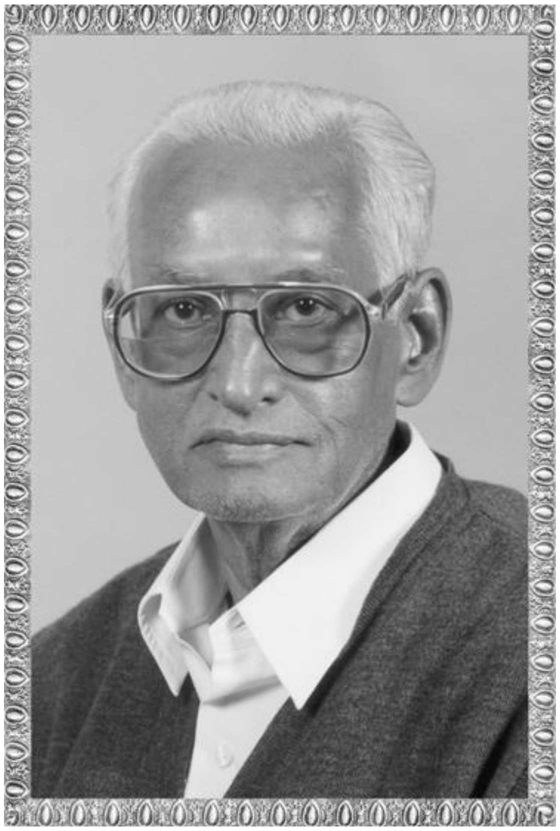
1. Explain scope resolution operator with suitable ex. (oct-2017) (dec-2018) (jan-2021)
2. Differentiate: OOP vs POP. (oct-2017)
3. Explain inline function with example. (dec-2018)
4. Explain function overloading. (dec-2018)
5. What is call by reference? Explain in details. (jan-2021)
6. What is Function Prototype? Explain in detail. (jan-2021)

5 Marks

1. List out operators available in C++. Explain memory management operators with syntax & Ex. (oct-2017)
2. Explain function overloading with suitable ex. (oct-2017)
3. Explain data type in details. (dec-2018)
4. What is OOP? Explain basic concept of OOP. (dec-2018)
5. What is Conditional Structure? Explain any ine in details. (jan-2021)
6. What is Looping Control Structure? Explain any one in detail. (jan-2021)

SHREE H. N. SHUKLA GROUP OF COLLEGES

(Affiliated to Saurashtra University & Gujarat Technological University)



Lt. Shree Chimanbhai Shukla

COMPUTER FUNDAMENTAL
BCA & BSC IT SEM 1

Shree H.N.Shukla College Campus,
Street No. 2, Vaishali Nagar,
Nr. Amrapali Railway Crossing,
Raiya Road, Rajkot.

Shree H.N.Shukla College
Street No. 3, Vaishali Nagar,
Nr. Amrapali Railway Crossing,
Raiya Road, Rajkot.
Ph. (0281)2471645

CH -2 classes and objects, constructor and destructor

Topic: Explain structure in detail.

- ✓ A structure combines logically related data items into a single unit.
- ✓ It is a user define data types.
- ✓ It can be used to create a variables, which can be used in the same way as variables of standard data types.
- ✓ Syntax:

```
 struct structure name
 {
      Data type member1;
      Data type member2;
      -----
 };
```

- ✓ According to syntax the structure declaration starts with struct keyword with structure name.
- ✓ The data type of each variable is specified in the individual member declaration.
- ✓ The closing bracket is terminated with a semicolon.

Ex:

- ➢ Write a c++ program for enter the student roll no and name from user using structure.

```
#include<iostream.h>
#include<conio.h>
struct std
{
    int no;
    char name[40];
    void input()
    {
          cout<<"Enter no:";
          cin>>no;
          cout<<"Enter name:";
          cin>>name;
    }
    void display()
    {
      cout<<endl<<"No is:"<<no;
      cout<<"\n Name is:"<<name;
    }
};
void main()
{
    clrscr();
    std s1;
```

```
        s1.input();
        s1.display();
        getch();
}
```

1 word Question Answer

Sr No.	Question	Answer
1.	A _________ combines logically related data items into a single unit.	Structure
2.	Structure is _________ data type.	User define

Topic: Explain Access Specifiers.

- ✓ Access specifiers define how the members (attributes and methods) of a class can be accessed.
- ✓ The public keyword is an **access specifier.**
- ✓ the members are public - which means that they can be accessed and modified from outside the code.
- ✓ In C++, there are three access specifiers:
  1. public - members are accessible from outside the class.
  2. private - members cannot be accessed (or viewed) from outside the class.
  3. protected - members cannot be accessed from outside the class, however, they can be accessed in inherited classes. You will learn more about Inheritance later.

Topic: Explain class in detail.

- ✓ A class is an entity which binds the data and its associated functions together defining variables of class data type is known as a class instance and such a variables are called objects.
- ✓ Syntax:

```
class classname
{
    private:
            variable declaration;
            function declaration;
            public:

                    variable declaration;
                    function declaration;

        };
```

- ✓ According to syntax, the keyword class indicates that the name which follows class_name.
- ✓ The body at the class is enclosed within the bracket and closing bracket is terminated with the semicolon.
- ✓ The body of class having declaration of variable and function, known as members.
- ✓ Variables of class is known as **data member** and function known as **member function.**
- ✓ The member specifies the visibility mode either private or public.
- ✓ The private members are accessible only inside the class or their own, class's member.
- ✓ The public members accessible inside the class and also outside the class also.
- ✓ By default all the class members in class are private.
- ✓ It hides the data from externals. So, data hiding is the key features of the OOP.

❖ Creating object:

- ✓ A class specification only declares the structure of the object.
- ✓ To use the members of the class need of instance of class.
- ✓ **Class_name object list;**
- ✓ In above syntax class_name is the name of the class and object list contains the different object name.
- ✓ Ex:

    Data d1;

    Data d1,d2,d3;

- ✓ Objects can also be created by placing their name immediately after the closing bracket of the class.
- ✓ Ex:

```
Class Data
{
    _______
    _______
}d1,d2,d3;
```

❖ Defining a member functions:

- ✓ A member functions of the class can be defined in any of the following two methods:
    1. Inside the class specification
    2. Out side the class specification

1. Inside the class specification

- ✓ When specify the member function inside the class it follows the same syntax as a normal function definition .
- ✓ But here the function definition is enclosed within the body of the class.

- ✓ Member functions defined insides a class are considered as inline functions by default.

2. Outside the class specification

- ✓ The function defined outside the class specification have the same syntax as normal functions.
- ✓ Only the difference is that a member function specifying with a **'identity-label'** to inform the compiler, the class to which the function belongs by using the scope resolution operator (::).
- ✓ Syntax:

```
Class class_name
{
     Return_type member_function(arguments);
          };

          Return_type class_name :: member_function(srguments)

          {

                    //Function Body

          }
```

- ✓ As shown in above syntax the label class_name informs the compiler that the member function belongs to the class_name.
- ✓ Ex:
- ➢ Write a c++ program for enter the student roll no and name from user using class and its member function.

```
#include<iostream.h>

#include<conio.h>

#include<stdio.h>

class std

{

        int no;

        char name[40];

        public:

        void input()

        {

                cout<<"Enter no:";

                cin>>no;
                cout<<"Enter name:";
                gets(name);
        }
        void display()
        {
           cout<<endl<<"No is:"<<no;
           cout<<"\n Name is:"<<name;
        }
};
void main()
{
        clrscr();
        std s1;
        s1.input();
        s1.display();
        getch();
}
```

- Write a c++ program for member function with arguments.

```
#include<iostream.h>
#include<conio.h>
#include<stdio.h>
#include<string.h>
class data
{
    int no;
    char name[20];
    public:
    void input(int,char *);
    void output();
};
void data::input(int n,char nm[])
```

```
    {
        no=n;
        strcpy(name,nm);
    }
    void data::output()
    {
        cout<<"\n number is:"<<no;
        cout<<"\n Name is :"<<name;
    }
    int main()
    {
        int no;
        char name[20];
        data d1;
        clrscr();
        cout<<"Enter the number: ";
        cin>>no;
        cout<<"Enter the Name: ";
        gets(name);
        d1.input(no,name);
        d1.output();
        getch();
        return 0;
    }
```

- Write a c++ program for nesting of member function.

```
#include<iostream.h>
#include<conio.h>
class number
{
    int no1,no2;
    public:
    void input();
    void output();
    int add();
};
void number::input()
{
    cout<<"\n Enter any two Numbers: ";
    cin>>no1>>no2;
}
int number::add()
{
    return(no1+no2);
}
void number::output()
{
     cout<<endl<<"No1 is: "<<no1;
     cout<<endl<<"No2 is: "<<no2;
     cout<<endl<<"Addition of two number is: "<<add();
}
void main()
{
     clrscr();
     number n1;
     n1.input();
     n1.output();
     getch();
}
```

- Write a c++ program from access the private member function.

```
#include<iostream.h>
#include<conio.h>
#include<stdio.h>
class std
{
     int no;
     char name[50];
     void input()
     {
       cout<<"\n Enter number:";
       cin>>no;
     }
     public:
     void display();
}s1;
void std::display()
{
     input();
     cout<<"\n Square of number is: "<<no*no;
}
void main()
{
     clrscr();
     std s2;
     s1.display();
     s2.display();
     getch();
}
```

1 word Question Answer

Sr No.	Question	Answer
1.	Data members and member functions of a class in C++ program are by default	Private
2.	The classes in c++ are used to manipulate ____________ .	data and functions
3.	Variables of class is known as __________ and function known as __________.	data member, member function
4.	The _________members are accessible only inside the class or their own, class's member.	Private
5.	The _________ members accessible inside the class and also outside the class also.	Public

Topic: How to make outside function inline?

- ✓ An inline function is working as a macro, any call to this function in program is replaced by the function calling itself.
- ✓ In c++ all the member functions that are defined within the class specification is inline by default.
- ✓ member function declare outside the class specification can be made inline by prefixing inline keyword to its definition.

➢ **Write a c++ program for inline outside member function.**

```
#include<iostream.h>
#include<conio.h>
class number
{
    int no1,no2;
    public:
    void input(int,int);
    void output();
};
inline void number::input(int n1,int n2)
{
    no1=n1;
    no2=n2;
}
inline void number::output()
{
    cout<<endl<<"No1 is: "<<no1;
    cout<<endl<<"No2 is: "<<no2;
}
void main()
{
    clrscr();
    number n1,n2;
    n1.input(10,20);
    n2.input(20,30);
    cout<<"The Numbers are: ";
    n1.output();
    n2.output();
    cout<<endl<<"Size of operator 1 is: "<<sizeof(n1);
    cout<<endl<<"Size of operator 2 is: "<<sizeof(n2);
    getch();
}
```

1 word Question Answer

Sr No.	Question	Answer
1.	An _________function is working as a macro, any call to this function in program is replaced by the function calling itself.	Inline
2.	Member function declare outside the class specification can be made inline by prefixing ________ keyword to its definition .	inline

Topic: Explain array within a class.

✓ An array is derived data type. Array is a group of related data items of same type which share a common name. The array can also be used as a member variable.

➢ Write a c++ program for array with in a class.

```
#include<iostream.h>
```

```
#include<conio.h>
class sum
{
        int no[10];
        int s;
        public:
                void input();
                void output();
};
void sum::input()
{
        for(int i=0;i<10;i++)
        {
                cout<<"\n Enter the value of no ["<<i+1<<"]: ";
                cin>>no[i];
        }
}
void sum::output()
{
        int tot=0;
        for(int i=0;i<10;i++)
        {
                tot=tot+no[i];
        }
        cout<<"\n Total of all Number is: "<<tot;
}
```

```
void main()
{
        sum s1;
        clrscr();
        s1.input();
        s1.output();
        getch();
}
```

1 word Question Answer

Sr No.	Question	Answer
1.	An array is ________ data type	Derived
2.	________is a group of related data items of same type which share a common name	Array

Topic: Explain Memory allocation for objects.

- ✓ The object is a variable of class. A class contain both member variable and member function.
- ✓ An object occupies number of bytes in a memory.
- ✓ The member function are created and placed in memory only one when, they are defined in the class declaration because all the objects belonging to that class use the same member functions. When the objects are created.
- ✓ The data members are placed in memory when each object is defined because all object have the separate data member and allocate separate memory of each data member for each object.
- ✓ The sizeof operator is used to find out size of standard data type.
- ✓ The size of any class object can also be find using it.

1 word Question Answer

Sr No.	Question	Answer

1.	The ______ operator is used to find out size of standard data type	Sizeof

Topic: Explain Static data Member.

- ✓ Sometimes situation occurs when need one or more common data member, which are accessible to all the objects of the class.
- ✓ Syntax:

```
Class Class_Name
{
    Static data_type  Data_member;
};
```

- ✓ According to syntax, the static data member can be created by static keyword .
- ✓ It is initialized during its definition outside all the member functions.
- ✓ The static data member is stored separately rather than as a part of an object.
- ✓ it is associated with the class itself, rather than with any class object.
- ✓ It is also known as **class variable.**
- ✓ Like:

```
        Datatype class_name :: Datamember= value;
```

- ✓ Consider following characteristics of static data member as follows:
  1. Only one copy of that member is created for the entire class and is shared by all objects of that class . Al l the changes made to that member at same memory location. So when display the value of that always get last value.
  2. When first object of class is created it is initialized to zero bydefault.
  3. It is visible only within the class, but its life is entire the program.

➢ Write a c++ program for static data member.

```
#include<iostream.h>
#include<conio.h>
class test
{
    static int count;
    int num;
    public:
    void setdata(int n)
    {
        num=n;
        ++count;
    }
    void display()
    {
        cout<<endl<<"Value of number: "<<num;
        cout<<endl<<"Value of static variable: "<<count;
    }
};
int test::count;
void main()
{
    clrscr();
    test t1,t2,t3;
    t1.display();
    t1.setdata(10);
    t1.display();
    t2.setdata(20);
    t2.display();
    t3.setdata(30);
    t3.display();
    getch();
}
```

1 word Question Answer

Sr No.	Question	Answer
1.	The static data member can be created by _______ keyword.	Static
2.	The static data member is stored separately rather than as a part of an________ .	Object

Topic: Explain Static member Function.

- ✓ Same as static member variable, class can also have static member function.
- ✓ Consider following points about static member function.
  1. Only one copy of static member function exists for all instance of class.
  2. A static member function can access only static data member.
  3. A static member function can be called using the class_name instead of its object. Such as:

     Class_name :: Function_name

- ➢ Write a c++ program for static member function.

```
#include<iostream.h>

#include<conio.h>
```

```
class test
{
	static int count;
	int num;
	public:
	void setdata(int n)
	{
		num=n;
		++count;
	}
	static void display()
	{
		cout<<endl<<"Value of static variable: "<<count;
	}
	void dispnum()
	{
		cout<<endl<<"Value of number is: "<<num;
	}
};
int test::count;
void main()
{
	clrscr();
	test t1,t2,t3;
	test::display();
	t1.setdata(10);
```

```
        test::display();

        t1.dispnum();

        t2.setdata(20);

        test::display();

        t2.dispnum();

        t3.setdata(30);

        test::display();

        t3.dispnum();

        getch();

}
```

1 word Question Answer

Sr No.	Question	Answer
1.	A static member function can access only _______ data member.	Static

Topic: Explain array of objects.

- ✓ An array is a group of same data type and stored in adjacent memory location.
- ✓ As an array of any other standard data type, the array of class object can also be created.
- ✓ It is known as an array of objects.
- ✓ Consider the following format to create array of object:

```
Class class_name

{

        Private:

                Variable declaration;

                Function declaration;

        Public:
                Variable declaration;

                Function declaration;

        };

        Class_name  object[size];
```

- ✓ Here, size defines the size of the array of class object.

➢ Write a c++ program for static member function.

```
#include<iostream.h>
#include<conio.h>
#include<stdio.h>
class std
{
    int rlno;
    char stnm[30];
    public:
    void input()
    {
        cout<<"Enter rono:";
        cin>>rlno;
        cout<<"Enter Std name:";
        gets(stnm);
    }
    void output()
    {
        cout<<"\n Rollno is: "<<rlno;
        cout<<"\n Name is: "<<stnm;
    }
};
void main()
{
    clrscr();
    std s[3];
    for(int i=0;i<3;i++)
    {
        s[i].input();
    }
    for(i=0;i<3;i++)
        s[i].output();
    getch();
}
```

Topic: Explain object as function argument.

- ✓ Like any other data type, an object can be passed as an argument to a function.
- ✓ This can be achieved by following ways:
    1. Pass – by – value
    2. Pass - by – reference
    3. Pass – by – pointer

1. Pass – by – value
    - ✓ In the case of pass – by – value a copy of object is passed to the function.
    - ✓ It modification made to the object inside function is not reflected to the object that is used to call the function.

- ➢ Write a c++ program for passing the object by value.

```
//Write a c++ program for pass argument object by value.....

#include<iostream.h>

#include<conio.h>

class calc

{

        int no1,no2;

        public:

        void input(int n1,int n2)

        {

                no1=n1;

                no2=n2;

        }

        void output()

        {

                cout<<endl<<"No1= "<<no1;

                cout<<endl<<"No2= "<<no2;

        }

        void add(calc c1,calc c2)

        {

                no1=c1.no1+c2.no1;
                no2=c1.no2+c2.no2;
        }
};
int main()
{
        clrscr();
        calc c1,c2,c3;
        c1.input(5,10);
        c2.input(20,30);
        c1.output();
        c2.output();
        c3.add(c1,c2);
        cout<<endl<<"Addition of two object is: ";
        c3.output();
        getch();
        return 0;
}
```

2. Pass – by – reference
    - ✓ In the case of pass – by – reference any changes made to the object inside the function is reflected in the actual object.
    - ✓ Here, object is passed same as pass by value but received in the function body with its reference.

3. Pass – by – pointer
    - ✓ In the case of pass – by – pointer an address of object passed to function.
    - ✓ Any changes made in value of object function it will also affect the original value.
    - ✓ The member of object passed by pointer is accessed using the operator.

1 word Question Answer

Sr No.	Question	Answer
1.	An _______ can be passed as an argument to a function.	Object

Topic: Explain returning object from function.

- ✓ Similar to sending objects as arguments to the function, it is also possible to return object from the function.
- ➢ Write a c++ program for returning object from function.

```
#include<iostream.h>
#include<conio.h>
class a
{
    int x,y,z;
    public:
    void input()
    {
    cout<<endl<<"Enter Three Value..";
    cin>>x>>y>>z;
    }
    void output()
    {
    cout<<endl<<"x: "<<x<<"\ty:  "<<y<<"\tz:  "<<z;
    }
    a add(a a1)
    {
    a a2;
    a2.x=a1.x+a1.y+a1.z;
    return a2;
    }
};
void main()
{
    a a1,a2;
    clrscr();
    a1.input();
    a1.output();
    a2=a2.add(a1);
    a2.output();
    getch();
}
```

Topic: Explain friend function.

- ✓ As the object oriented concept the private member of class can not be accessed from outside the class.
- ✓ However, under certain situations, sometimes requirement to use a common function with two classes.
- ✓ In c++ this is achieved by using the concept of **friends.** It permits a function of another class to access different class's private members.
- ✓ The friendly function must be prefixed by keyword **friend**.
- ✓ The function calling is same as ordinary function without scope of class.
- ✓ A function calling is same as normal function without the dot(.) operator.
- ✓ The special characteristics of a friend function are as follows:
  1. It is not declared as a member of any class. So the scope of friend function is not limited to the class in which it has been declared as a friend.
  2. It can only be invoked like a normal function without the help of any object because it is not in the scope of class.
  3. It can access the private members of a class using object name and dot(.) operator.
  4. It can be declared either in the public or private section without affecting its accessibility.
  5. It has the objects as arguments.
- ✓ Format of friend function.

```
    Class sample
    {
        Int x,y;
        Public:
                friend void add();//declaration of friend function
        };

//definition of friend function like normal function

        Void add()

        {

                //body of function;

        }

        int main()
    {
        sample obj;
        add(obj);
                }
```

- Write a c++ program for friend function.

```
//write a c++ program for friend function.....
#include<iostream.h>
#include<conio.h>
class data2;
class data1
{
    int x;
    public:
    void input()
    {
        cout<<endl<<"Enter Value of x:";
        cin>>x;
    }
    void output()
    {
        cout<<endl<<"Value of x is: "<<x;
    }
    friend int max(data1,data2);
};
class data2
{
    int y;
    public:
    void input()
    {
        cout<<endl<<"Enter value of y:";
        cin>>y;
    }
    void output()
    {
        cout<<endl<<"Value of y is: "<<y;
    }
    friend int max(data1,data2);
};
int max(data1 d1,data2 d2)
{
    if(d1.x>d2.y)
    {
        return (d1.x);
    }
    else
    {
        return (d2.y);
    }
}
int main()
{
    clrscr();
    data1 d1;
    data2 d2;
    d1.input();
    d2.input();
    d1.output();
    d2.output();
    cout<<endl<<"Maximum value is: "<<max(d1,d2);
    getch();
    return(0);
}
```

1 word Question Answer

Sr No.	Question	Answer
1.	________ permits a function of another class to access different class's private members.	Friend function
2.	The friendly function must be prefixed by keyword_______.	Friend
3.	_______ can only be invoked like a normal function without the help of any object because it is not in the scope of class	Friend function

Topic: Explain const member function.

- ✓ If the member function of a class access the class data member without modifying them, then declare such function as **const** (constant) function.
- ✓ syntax :

return type function name(argument)const.

- ✓ According to above syntax the keyword **const** is placed at the end of function header, at the time of declaring its prototype and its definition.
- ✓ The compiler will generate an error message if function try to change the data values.
- ➢ Write a c++ program for const member function.

```
//write a c++ program for const function.....
#include<iostream.h>
#include<conio.h>
class data
{
    int i;
    public:
    void set(int x)
    {
        i=x;
    }
    void change()const
    {
        //i=30;
        cout<<endl<<"you can not change the value of const function:";
    }
    void show()
    {
        cout<<endl<<"i: "<<i;
    }
};
void main()
{
    clrscr();
    data d;
    d.set(10);
    d.show();
    d.change();
    d.show();
    getch();
}
```

Topic: Explain pointer to member function.

- ✓ pointer variable holds the address of another variable.
- ✓ It is also possible to take the address of member of a class and assign it to a pointer.

- ✓ The address of a member can be obtained with using the address of operator (&) to a class member name.
- ✓ C++ provides special operator pointer to member (:: *) with the class_name to declare a class member pointer.
- ✓ For ex:

```
class sample
{
    int x;
    public:
        Void display();
        };
```

- ✓ Pointer to member can be created as follows:
  int sample :: *p=&sample :: x;
- ✓ Here, the p is pointer to member and the portion **sample :: *** means **"pointer-to-member of sample class"** and the portion **&sample :: x** means **"address of the x member of class sample".**
- ✓ int *p=&x;
- ✓ here, this statement is invalid.
- ✓ It indicate pointer of normal variable.
- ✓ The x is not only int variable but a member variable of class sample.
- ✓ The scope resolution operator must be attached to both the pointer and the member.
- ✓ A pointer can be used to access the member variable x is as follow.

```
    sample s;
    cout<<s.*p;         //display value of x
    cout<<s.x; //display value of x
```

- ✓ The dereferencing operator(->*) is used to access a member when use pointer to both object and member.
- ✓ The dereferencing operator(.*) is used when the object use only member pointer.
- ✓ For ex:

```
sample *s;
s=&x;
cout<<s->*p;
cout<<s->x;
```

- ✓ The pointer member function can be invoked using the dereferencing operators in the main() as follows:

(object name.*pointer_to_member function)(value);
(pointer-to-object ->*pointer-to-member function)(value);

- ➢ Wrire a c++ program for pointer to member function.

```
//write a c++ program for pointer to class member....
#include<iostream.h>
#include<conio.h>
class data
{
	int a,b;
	public:
	void setdata(int x,int y)
	{
	a=x;
	b=y;
	}
	  friend int add(data);
};
int add(data d1)
{
	//*p1=&a;
	int data::*p1=&data::a;
	int data::*p2=&data::b;
	//data d1;
	data *tmp=&d1;
	int sum;
	//sum=d1.a+d1.b;
	//sum=d1.*p1+d1.*p2;
	//sum=tmp->*p1+tmp->*p2;
	//sum=tmp->a+tmp->b;
	sum=d1.*p1+tmp->*p2;
	return(sum);
}
void main()
{
	clrscr();
	data d;
	d.setdata(5,10);
	cout<<endl<<"Addition of two number is: "<<add(d);
	getch();
}
```

1 word Question Answer

Sr No.	Question	Answer
1.	pointer variable holds the ________ of another variable.	Address
2.	The _________ operator must be attached to both the pointer and the member.	scope resolution

3.	The __________ is used to access a member when use pointer to both object and member.	dereferencing operator(->*)
4.	The _________ is used when the object use only member pointer.	dereferencing operator(.*)

Topic: Explain nested class and local class.

- ✓ When specifies any class inside another class, then its known as nested class.
- ✓ For ex:

```
class outer
{
    class inner
    {
    -------
    -------
    };
    inner obj;
    public:
    --------
    --------
};
```

- ✓ As shown in above ex the class outer has one class inside it called inner.
- ✓ It will create an object of class inner and all member function of class outer can access member variable of inner class by its object.
- ✓ It is also possible to access the class outside the other class, but then use the inner class with the scope of outer such as outer :: inner.
- ✓ When classes defined and used inside function block then it is known as local class.
- ✓ For ex:

```
void demo()
{
    class inner
    {
            --------
            --------
```

```
        };
        ------
        ------
        Inner obj;
                };
```

- ✓ Here, the function sample can has a class_name inner local class can use global.
- ✓ Variables and static variable declared inside the function but cannot use automatic local variables.
- ✓ .
- ✓ The function demo can not access the private member of a local class.

➢ Write a c++ program for nested class.

```
#include<iostream.h>
#include<conio.h>
class A
{
    public:
    class B
    {
        private:
        int num;
        public:
        void getdata(int n)
        {
              num = n;
        }
        void putdata()
        {
              cout<<"The number is "<<num;
       }
    };
};
int main()
{
  clrscr();
  cout<<"Nested classes in C++"<< endl;
  A :: B obj;
  obj.getdata(9);
  obj.putdata();
  getch();
  return 0;
}
```

- Write a c++ program for local class.

```
#include<iostream.h>
#include<conio.h>
void demo()
{
    class inner
    {
        int x;
        public:
        void set(int y)
        {
            x=y;
        }
        void disp()
        {
            cout<<"Value of x is: "<<x;
        }
    };
    inner obj;
    obj.set(10);
    obj.disp();
}
void main()
{
    clrscr();
    demo();
    getch();
        }
```

1 word Question Answer

Sr No.	Question	Answer
1.	When specifies any class inside another class, then its known as ______class.	nested
2.	When classes defined and used inside function block then it is known as _________ class.	Local
3.	Local class can not have ________ data members	static

Topic: What is Constructor?

- ✓ There is a need to initialize the value of data member before accessing it.
- ✓ Normally member function are used to initialize value of variable.
- ✓ For Ex:
```
    Obj.input(10)
```
- ✓ Here, member function input() passes the initialize value as arguments these values are assigned to the private data member of object obj.
- ✓ C++ provides a special member function called the constructor that enables an object to be initialized when it is created.
- ✓ A constructor is different than other member function of the class it has the same name as its class without return type even not void.
- ✓ Similar to other member function the constructor can be defined either within, or outside the body of a class.
- ✓ It can access an data member like all other member function.
- ✓ It must be declared in public part.
- ✓ Syntax:
```
class class_name
{
    private:
        data member;
    public:
        class_name()   //constructor
        {
                -------
                -------
        }
        Other member functions;
};
```
- ✓ Here, constructor is defined in a class body by the name of class in public section.
- ✓ Constructor defining outside of class
- ✓ Ex:
```
class class_name
{
    private:
        data members;
    public:
        classname();    //constructor declaration
};
classname :: classname()
{
    //body of constructor
}
```

- ✓ Here, constructor is defined outside the class with the scope of class.
- ✓ The constructor of a class is the first member function to be executed automatically when object of class is created.
- ✓ It is executed every time an object is created it can be used to assign initial, value to the data members of the object.

1 word Question Answer

Sr No.	Question	Answer
1.	C++ provides a special member function called the constructor that enables an ________to be initialized when it is created.	Object
2.	A ___________is different than other member function of the class it has the same name as its class without return type even not void.	Constructor
3.	The __________of a class is the first member function to be executed automatically when object of class is created.	constructor

Topic:Give the Characteristics of Constructor?

- ✓ It has the same name as a class name.
- ✓ It is executed automatically whenever the class object is created.
- ✓ It does not have any return type not even void.
- ✓ It has also default argument as other functions.
- ✓ It is normally used to initialize the data member of a class.
- ✓ It is also used to allocate resources such as memory, to dynamic data members of a class.
- ✓ The address at constructor cannot be referred.
- ✓ It cannot be virtual.
- ✓ It cannot be inherited, through derived class call the base class constructor.

- ✓ It makes use of new and delete operator as implicit call when memory allocation is required.

1 word Question Answer

Sr No.	Question	Answer
1.	______ has the same name as a class name.	**Constructor**
2.	_______ is executed automatically whenever the class object is created.	**Constructor**

Topic:What Default Constructor?

- ✓ The default constructor is a special member function with no arguments which initialize the data members.
- ✓ The default constructor accepts no parameters.
- ✓ For ex the default constructor for class data is data() constructor.
- ✓ If no constructor is defined for a class then the compiler supplies the default constructor.
- ➢ Write a c++ program for default constructor.

```
#include<conio.h>
#include<iostream.h>
class demo
{
    int x;
    public:
    demo()
    {
    cout<<"Demo for constructor....";
    x=10;
    }
    void disp()
    {
    cout<<endl<<"Value of x is: "<<x;
    }
};
void main()
{
    clrscr();
    demo d;
    d.disp();
```

```
	getch();
}
```

1 word Question Answer

Sr No.	Question	Answer
1.	The ______is a special member function with no arguments which initialize the data members.	default constructor
2.	The _______accepts no parameters.	default constructor

Topic:What is Parameterized Constructor?

- ✓ The constructor with arguments is called parameterized constructors.
- ✓ It can be invoked same as a function argument by specifying arguments list in brackets.
- ✓ When the constructor is parameterized, must be provide appropriate argument to the constructor.
- ✓ Syntax:

```
Class sample
{
	---------
	---------
	Public:
		sample(int x)
		{
			------
			------
		}
	--------
};
```

- ✓ sample s(5);

```
	or
sample s = sample(5);
```

- ✓ here, the constructor has a one argument which passed when object of class is created.
- ✓ The passing of initial values to the constructor can be done by two way.
  1) Implicit call – sample s(5)
  2) Explicit call – sample s = sample(5)

➢ Write a c++ program for parameterized constructor.

```
// Write a c++ program to find area of rectangle with parameterized constructor..
#include<iostream.h>
#include<conio.h>
```

```
class ABC
{
  private:
     int length,breadth,x;
  public:

     ABC (int a,int b)
     {
       length = a;
       breadth = b;
      }
      void area()
      {
       x = length * breadth;

      }
      void display()
      {
        cout << "Area = " << x << endl;
      }
};

int main()
{
     clrscr();
     int len,brth;
     cout<<endl<<"Enter length of rect: ";
     cin>>len;
     cout<<endl<<"Enter breadth of rect: ";
     cin>>brth;
     ABC obj(len,brth);
     obj.area();
     obj.display();
     getch();
     return 0;
 }
```

1 word Question Answer

Sr No.	Question	Answer

1.	The constructor with arguments is called_______.	parameterized constructors
2.	When the constructor is parameterized, must be provide appropriate _______ to the constructor.	argument

Topic: Explain multiple constructor in class.

- ✓ When the class has multiple constructors, is called **constructor overloading.**
- ✓ All the constructors have the same name as the class in which they belongs to. Same as function overloading.
- ✓ Each constructor should be differ from their number of arguments and types of arguments.

1 word Question Answer

Sr No.	Question	Answer
1.	When the class has multiple constructors, is called ________.	**constructor overloading**
2.	Each constructor should be differ from their _______of arguments and _____of arguments.	Number, types

Topic: Explain constructor with default argument.

- ✓ It is possible to define constructors with arguments having default value same as any other function in c++.
- ✓ If any arguments are passed during the creation of an object, the compiler selects the suitable constructor with default argument.
- ✓ For ex: sample (int i,int j=0);
- ✓ Here in above constructor the default value of argument j is 0 when create an object.
- ✓ Sample s(10)
- ✓ It assigns the value 10 to i and 0 to y by default.
- ✓ When call the constructor with new value of j it will replaced by new value. Such as
- ✓ Sample s(10,20)
- ✓ Now the value of j is consider as 20.
- ➢ Write a c++ program for constructor with default argument.

```
#include<iostream.h>
#include<conio.h>
```

```
class demo
{
      int i,j;
      public:
      demo(int a,int b=10)
      {
      i=a;
      j=b;
      }
      void disp()
      {
      cout<<endl<<"Value of i is: "<<i;
      cout<<endl<<"Value of j is: "<<j;
      }
};
void main()
{
      int a;
      clrscr();
      cout<<endl<<"Enter value: ";
      cin>>a;
      demo d(a);
      d.disp();
      getch();
}
```

Topic: Explain copy constructor.

- ✓ A constructor can have argument of any data type.
- ✓ When constructor has object of its own class is called copy constructor.
- ✓ The object of own class must be passed as a reference parameter.
- ✓ For ex:

```
Class test
{
      ------
      ------
      public :
      test(test &obj)
      {
              -------
              -------
         }
         ------
};
```

- ✓ Such constructor having a reference to an instance of its own class as an argument is known as copy constructor.
- ✓ A compiler copies all the members of the user define source object to the destination object in the assignment statement.
- ✓ For ex:
- ✓ test t1(5),t2(10);

t1=t2;

- ✓ here the copy constructor will not invoke. It just assigns the value of t1 to t2, member by member.
- ✓ This is the task of the overloaded assignment operator (=).
- ✓ Because both objects are predefined object.
- ✓ A copy constructor can be called like this way:

```
test t1(5);
test t2(t1);
    or
test t2=t1;
```

- ✓ Here, in both cases it will invoke copy constructor.
- ✓ The initialization of one object to another object is performed during object definition.
- ✓ The data member of t1 is copied to t2 member by member. It is the default action performed by copy constructor.

➢ Write a c++ program for constructor with default argument.

```
#include<iostream.h>
#include<conio.h>
class demo
{
    int i,j;
    public:
    demo(int a,int b=10)
    {
      i=a;
      j=b;
    }
    void disp()
    {
      cout<<endl<<"Value of i is: "<<i;
      cout<<endl<<"Value of j is: "<<j;
    }
};
void main()
{
    int a;
    clrscr();
    cout<<endl<<"Enter value: ";
    cin>>a;
    demo d(a);
    d.disp();
    getch();
}
```

1 word Question Answer

Sr No.	Question	Answer
1.	When constructor has object of its own class is called_______ .	copy constructor

Topic: Explain Dynamic Initialization of Object.

- ✓ A class object can also be initializing at run time. It is known as dynamic initialization of object.
- ✓ One advantage of dynamic initialization is that with the multiple constructors various initialization techniques can be provided.

➢ Write a c++ program for dynamic initialization of object.

```
#include<iostream.h>
#include<conio.h>
class test
{
    int x;
    public:
    test(int a)
    {
      x=a;
    }
    int add()
    {
      return (x*x);
    }
    void disp()
    {
      cout<<endl<<"Value of x is: "<<x;
    }
};
void main()
{
    clrscr();
    int a;
    cout<<endl<<"Enter Value of a:";
    cin>>a;
    test obj1(a);
    obj1.disp();
    test obj2(obj1.add());
    cout<<endl<<"New Object after Square:";
    obj2.disp();
    getch();
}
```

1 word Question Answer

Sr No.	Question	Answer
1.	A class object can also be initializing at run time. It is known as________.	dynamic initialization of object

Topic: Explain Dynamic Constructor.

- ✓ A Constructor normally used for the management of memory allocation during runtime.
- ✓ It provides a way to allocate the right amount of memory during execution for each object when the object's data member size is not same.
- ✓ Allocation of memory to objects at the time of their construction is known as dynamic constructor.
- ✓ The constructor which performs such memory allocation is called dynamic constructor.
- ✓ The dynamic memory allocation operator new is used to allocate memory dynamically.

➢ Write a c++ program for dynamic constructor.

```
#include<iostream.h>
#include<conio.h>
class sample
{
    int *no,size;
    public:
    sample(int s)
    {
      size=s;
      no=new int[size];
    }
    void input();
    void disp();
};
void sample::input()
{
      for(int i=0;i<size;i++)
      {
        cout<<endl<<"Enter Value for "<<i<<": ";
        cin>>no[i];
      }
}
void sample::disp()
{
      for(int i=0;i<size;i++)
      {
        cout<<endl<<no[i];
      }
}
void main()
{
      clrscr();
      int s;
      cout<<endl<<"Enter Size of Array: ";
      cin>>s;
      sample s1(s);
      s1.input();
      s1.disp();
      getch();
}
```

1 word Question Answer

Sr No.	Question	Answer
1.	Allocation of memory to objects at the time of their construction is known as_________.	dynamic constructor
2.	The dynamic memory allocation operator ______is used to allocate memory dynamically.	New

Topic: What is Destructor?

- ✓ The constructor is called to initialize data member and allocate memory for object at the time of creation.
- ✓ When an object is no longer needed it can be destroyed.
- ✓ Class can have special member function is called destructor.
- ✓ As similar to constructor it automatically gets call when object is destroyed.
- ✓ The destructor has the same name as the class but it specified with ~ complements.
- ✓ Syntax:

```
class clasname
{
    private:
        Member variable;
                public:

                -------

                -------

                ~classname()

                {

        }

};
```

- ✓ Here, shown in above syntax destructor is defined same name with classname.
- ✓ **Rules for defining destructor:**
- ✓ The destructor function has same name as class but prefixed with tiled(~). It makes difference of a constructor and destructor.
- ✓ It has no argument and no return type.
- ✓ Destructor is invoked automatically whenever an object goes out of scope.
- ✓ It is also declare in public section of class .
- ✓ Class can not have more than one destructor.
- ✓ **Difference b'ween constructor and destructor.**
- ✓ Arguments cannot be passed to destructor as constructor.
- ✓ Only one destructor can be declared for a given class while more than one constructor can work in same class.
- ✓ Destructor can not be overloaded as constructor.
- ✓ Destructor can be virtual, while constructor can not be virtual.

➢ Write a c++ program for destructor....

```
#include<iostream.h>
#include<conio.h>
class demo
{
        public:
        demo()
        {
        cout<<endl<<"Constructor is called...";
        }
        ~demo()
        {
        cout<<endl<<"Destructor is called....";
        }
};
int main()
{
        clrscr();
        demo d;
        getch();
        return 0;
}
```

1 word Question Answer

Sr No.	Question	Answer
1.	When an object is no longer needed it can be destroyed. Class can have special member function is called ________.	Destructor
2.	The destructor function has same name as class but prefixed with______.	tiled(~)
3.	________has no argument and no return type.	Destructor
4.	Destructor is invoked automatically whenever an object_________.	goes out of scope
5.	Destructor is also declare in ______section of class .	Public
6.	Class can not have _____destructor.	more than one

Topic: What is MIL?

✓ MIL stands for Multiple Initialize list.

- ✓ Initialize List is used in initializing the data members of a class.
- ✓ The list of members to be initialized is indicated with constructor as a comma-separated list followed by a colon.
- ✓ Syntax:

```
class Point
{
            private:
    int x;
    int y;
            public:
              Point(int i, int j):x(i), y(j) {}
};
```

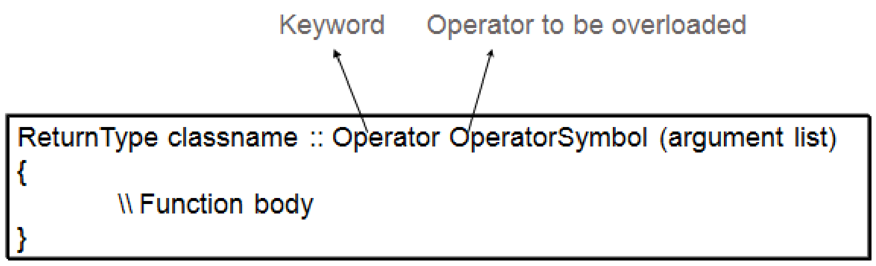
1 word Question Answer

Sr No.	Question	Answer
1.	MIL stands for_______.	Multiple Initialize list

Ch – 3 Operator overloading and type conversion, Inheritance

Q -1 What is operator overloading?

- ✓ Operator overloading is an important concept in C++. It is a type of polymorphism in which an operator is overloaded to give user defined meaning to it.
- ✓ You can redefine or overload most of the built-in operators available in C++. Thus, a programmer can use operators with user-defined types as well.
- ✓ Overloaded operator is used to perform operation on user-defined data type. For example '+' operator can be overloaded to perform addition on various data types, like for Integer, String(concatenation) etc.
- ✓ Operator Overloading Syntax:



```
ReturnType classname :: Operator OperatorSymbol (argument list)
{
        \\ Function body
}
```

- ✓ Return type specifies the type of value returned by the specified operation.
- ✓ Operator is the keyword. Operator should be preceded which want to be overloaded.
- ✓ Operator function must be either member function or friend function.
- ✓ A difference between them is that a friend function will have only one argument for unary operators and two for binary operators, while member function has no argument for unary operator and one argument for binary operator.
- ✓ Arguments may be passed either by value or by reference.
- ✓ The operator can be classified into unary and binary operators based on number of arguments on which they operate.
- ✓ C++ allow almost all operators to be overloaded but at least one operand must be an instance of a class(object).

1 word Question Answer

Sr No.	Question	Answer
1.	Operator function must be either _____ or________.	member function, friend function
2.	_________ is a type of polymorphism.	Operator overload
3.	The operator can be classified into _____ and ________ operators.	Unary, binary

Q-2 Give the rules for operator overloading.

1) Overloaded operators must perform operation similar to those defined for basic data types. Meaning of operator can not be change.
2) Only existing operator can be overloaded. New operator can not be created like $.
3) At least one operand should be class object.
4) Operator function can be either member function or friend function.
5) In case of member function, it will take no explicit argument and return no explicit values for unary operator. It will take one explicit argument and may be return explicit values for binary argument.
6) In case of friend function, it will take one explicit argument which is the reference of the relevant class for unary operator, it will take two explicit argument for binary operator.
7) When binary operators overloaded, through a member function, the left hand operand must be an object of the relevant class.
8) There are some operators cannot be overloaded. They are listed as follow:

.	member accessing operator.
.*	pointer to member operator
::	scope resolution operator
?:	conditional operator
Sizeof()	sizeof operator

9) Some of the operators can not be overloaded by friend function but member function can overloaded it. They are listed as follows:

=	assignment operator
( )	function call operator
[ ]	subscripting operator
→	Class member access operator

10) Binary arithmetic operators such as +,-,*,/ must explicitly return a value. They must not attempt to change their own arguments.

Q-3 Explain Unary Operator.

- ✓ Overloading without explicit arguments to an operator function is known as unary operator overloading.
- ✓ The unary operators like unary +, unary -, increment and decrement operator.
- ✓ Syntax:

```
            Returntype operator OP()
{
      //body of operator function
      }
```

- ✓ Where, function return type can be either from following : primitive, void, or user defined operator is the keyword OP specifies the operator symbol to be overloaded.
- ✓ For Ex:

Int operator – ();
Void operator ++ ();

1 word Question Answer

Sr No.	Question	Answer
1.	Overloading without explicit arguments to an operator function is known as ________.	unary operator overloading

Q-4 Explain Binary operator.

- ✓ The concept of overloading unary operators applies also to the binary operator.
- ✓ Syntax:

```
Returntype operator OP(arg)
{
      //body of function
}
```

- ✓ Here the function takes the first object as an implicit operand and the second operand must pass explicitly.
- ✓ The data members of first object are passed without using dot operator while second argument members can be accessed using the object and dot operator.
- ✓ For Ex:

Void operator / (data d1)

1 word Question Answer

Sr No.	Question	Answer
1.	In the binary operator the function takes the first object as an _______ operand and the second operand must pass________.	Implicit, explicitly

Q-5 Explain operator overloading with friend function.

- ✓ The operator function can also be used to overload operator.
- ✓ It provides a flexibility to use object of different class.
- ✓ The difference between a friend function and member function is that friend function require the arguments to be passed explicitly to the operator function while member function consider the first argument implicitly.
- ✓ Friend function can be used for both unary and binary operators.
- ✓ Syntax:

```
Friend returntype operator OP(arg1, arg2)
{
    //body of function
}
```

- ✓ According to above syntax the friend function must be prefixed with keyword friend.
- ✓ The body of the friend function can be either inside class or outside the class.
- ✓ When the definition of friend function is outside class then it defined as normal function and not prefixed with the friend keyword.
- ✓ It can return and data type wither or following: void, primitive or user defined.
- ✓ OP is the operator symbol to be overloaded and prefixed with operator keyword.
- ✓ For Ex:

```
Friend data operator +  (data d1,data d2);
```

1 word Question Answer

Sr No.	Question	Answer
1.	OP is the operator symbol to be overloaded and prefixed with ________ keyword.	Operator
2.	_______ function can be used for both unary and binary operators.	Friend

Q-6 Explain this pointer (*this).

- ✓ Normally the member function of a class is invoked, by some object of the class.
- ✓ C++ has keyword this pointer to represent an object that invokes a member function. Thus, member function of every object has access to a pointer named this, which points to the object itself. It represented as *this.
- ✓ This pointer can be treated like any other pointer to an object.
- ✓ Using this pointer, any member function can find out the address of the object of which it is a member.

1 word Question Answer

Sr No.	Question	Answer
1.	C++ has keyword ______ pointer to represent an object that invokes a member function.	This
2.	________ pointer can be treated like any other pointer to an object.	This

Q-7 Explain Type conversion.

- ✓ The assignment operator assigns the contents of a variable, the results of an expression, or a constant to other variable.
- ✓ There are three types of data conversion
  1) Basic type to class type
  2) Class type to basic type
  3) One type class to another class type

1) Basic type to class type

- ✓ To convert data from a basic type to user defined type.
- ✓ The conversion function should be defined in user defined object class in the form of constructor.
- ✓ The constructor can take a single argument whose type is to be converted.
- ✓ The syntax:

```
Constructor (basic type)
{
    //body for conversion
}
```

- ✓ Where constructor specifies the same name as class in which it belongs.

- ✓ It has only one argument of basic type.
- ✓ Conversion from basic type to class type is possible in following two ways.
- ✓ 1. Using Constructor
- ✓ 2. Using Operator

1. **Using Constructor**

- ✓ We can use constructor to perform type conversion during the object creation.
- ✓ To achieve that we have implemented one constructor function which accepts
- ✓ one argument.

2. **Using Operator Overloading**

- ✓ We can also achieve type conversion by operator overloading. We can overload assignment operator for this purpose.
- ✓ By using overloaded assignment operator we can perform the type conversion at any place in program.

2) class type to basic type

- ✓ Constructor function does not support the operation of class to basic type.
- ✓ In case of user defined data type conversion.
- ✓ The operator function is defined as an overloaded basic data type which has no any arguments.
- ✓ It returns converted data member object to basic data type.
- ✓ Syntax:

```
 Operator basictype ()
{
     //code for conversion
 }
```

- ✓ According to above syntax the operator is the keyword.
- ✓ Basic type specifies any primitive data type such as int, char, float double etc.

➢ The conversion function must satisfy the following conditions :-

- ✓ It must be a class member.
- ✓ It must not specify the return value even though it returns the value.
- ✓ It must not have any argument.

3) One class to another class type

- ✓ The c++ compiler does not support data conversion between object to two user define classes.
- ✓ However the conversion of class type to basic type and basic type to class type can be performed either by operator conversion function or constructor.
- ✓ Ex:

```
Classa obja;
Classb objb;
Obja=objb;
```

- ✓ Where obja is the object of classa and objb is the object classb.
- ✓ The class classa type is converted to the class classb and converted value is assigned to the obja.
- ✓ In this type of conversion both the type that is source type and the destination type are of class type. Means the source type is of class type and the destination type is also of the class type. In other words, one class data type is converted into the another class type.
- ✓ Conversion from one class to another class can be performed in following two ways

1. Using constructor
2. Using type conversion function

1. Using constructor

- ✓ Following is the example to show how constructor is used for conversion from one class type to another class type.

2. Using type conversion function

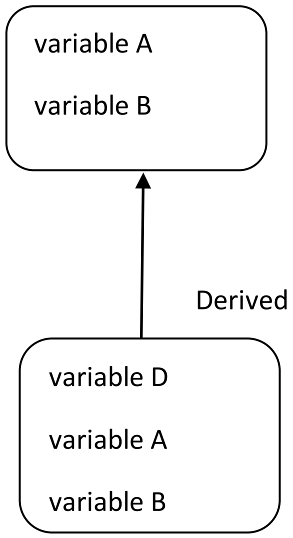
- ✓ Following is the example to show how type conversion is used for conversion from one class type to another class type.

1 word Question Answer

Sr No.	Question	Answer
1.	The conversion function should be defined in user defined object class in the form of________.	Constructor
2.	Constructor function does not support the operation of_________.	class to basic type
3.	The c++ compiler does not support data conversion between _______.	object to two user define classes

Q-8 What is inheritance?

- ✓ Inheritance is the process of creating new classes, called derived classes, from existing classes, which are often called base classes.
- ✓ The derived class inherits all the capabilities of the base class and it can also add new features of its own. But here base class remain unchanged.



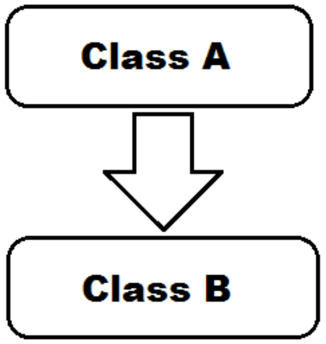
- ✓ As shown in above figure the base class has there features (A,B,C). The derived class has three features of base class and adds its own features (D). The arrow represents the direction of derivation.
- ✓ Here, its direction from the derived class towards the base class. It represents that derived class access features of the base class not vice-versa.
- ✓ This inheritance mode is used mostly. In this the protected member of Base class becomes protected members of Derived class and public becomes public.
- ✓ class derivedclass: public baseclass

Access specifier in base class	Access specifier when inherited publicly	Access specifier when inherited privately	Access specifier when inherited protectedly
Public	Public	Private	Protected
Protected	Protected	Private	Protected
Private	Inaccessible	Inaccessible	Inaccessible

- ✓ Derived class of Derived Classes: If we are inheriting a derived class using a public inheritance as shown below
- ✓ class B : public A
- ✓ class C : public B
- ✓ then public and protected members of class A will be accessible in class C as public and protected respectively.
- ✓ There are different types of inheritance:
  1) Single Inheritance
  2) Multiple Inheritance
  3) Multilevel Inheritance
  4) Hierarchical Inheritance
  5) Hybrid (Virtual) Inheritance

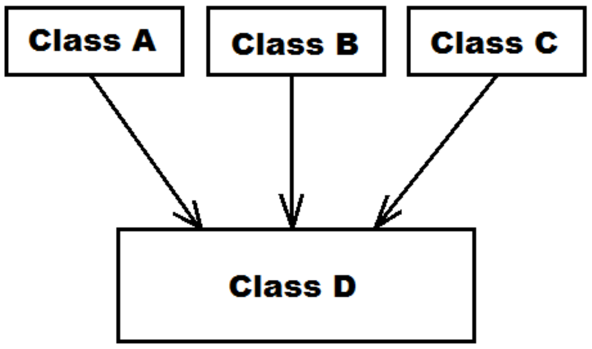
1. **Single Inheritance**

- ✓ Single inheritance represents a form of inheritance when there is only one base class and one derived class. For example, a class describes a Person**:**



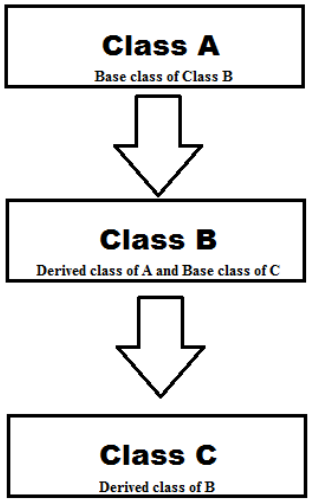
2. Multiple Inheritance

- ✓ Multiple inheritance represents a kind of inheritance when a derived class inherits properties of multiple classes. For example, there are three classes A, B and C and derived class is D as shown below:



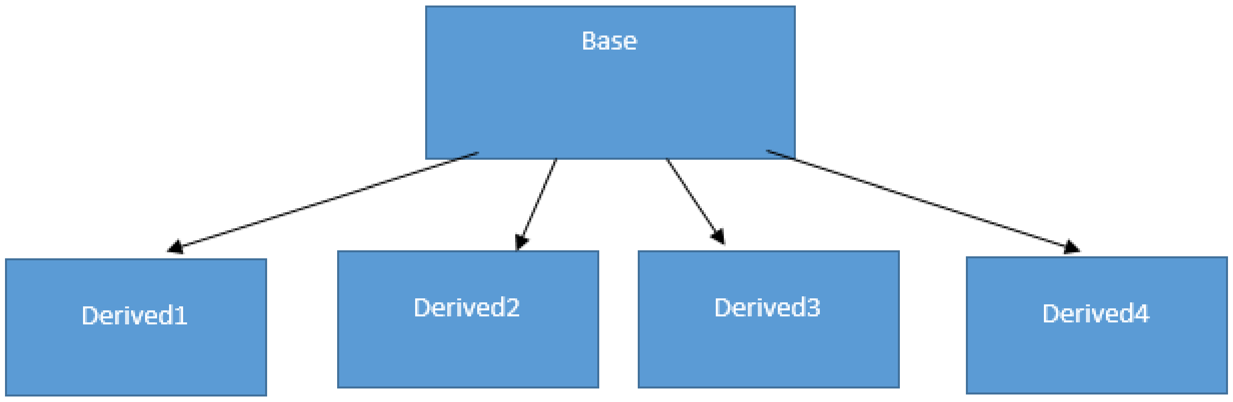
3. Multilevel Inheritance

- ✓ Multilevel inheritance represents a type of inheritance when a Derived class is a base class for another class. In other words, deriving a class from a derived class is known as multi-level inheritance. Simple multi-level inheritance is shown in below image where Class A is a parent of Class B and Class B is a parent of Class C



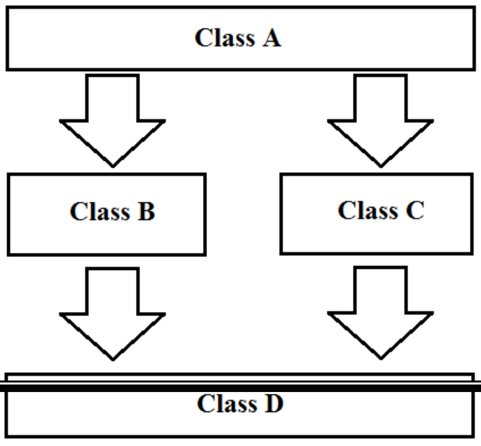
4. Hierarchical Inheritance

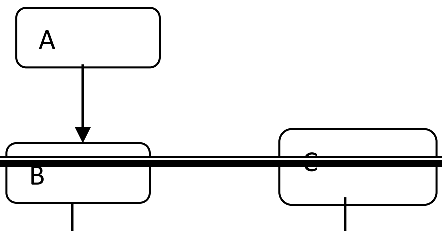
- ✓ When there is a need to create multiple Derived classes that inherit properties of the same Base class is known as Hierarchical inheritance

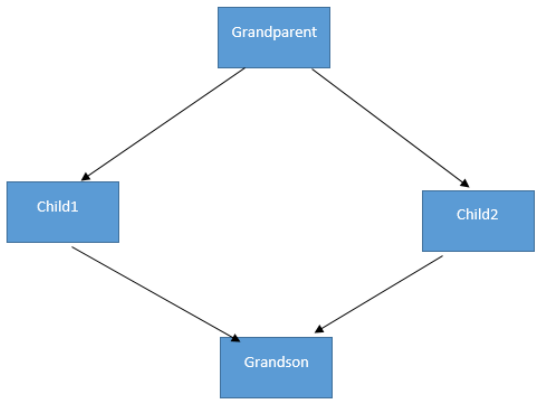


5.Hybrid Inheritance

- ✓ Combination of Multi-level and Hierarchical inheritance will give you Hybrid inheritance.







- ✓ **Virtual Inheritance**
- ✓ We can avoid Diamond problem easily with Virtual Inheritance. Child classes in this case should inherit Grandparent class by using virtual inheritance:
- ✓ Constructor in derived class
- ✓ When a default or parameterized constructor of a derived class is called, the default constructor of a base class is called automatically. As you create an object of a derived class, first the default constructor of a base class is called after that constructor of a derived class is called.
- ✓ When multiple inheritance is used, default constructors of base classes are called in the order as they are in inheritance list. For example, when a constructor of derived class is called:
- ✓ Destructor in derived class
- ✓ Deleting a derived class object using a pointer to a base class that has a non-virtual destructor results in undefined behaviour. To correct this situation, the base class should be defined with a virtual destructor. For example, following program results in undefined behaviour.

- ✓ Making base class destructor virtual guarantees that the object of derived class is destructed properly, i.e., both base class and derived class destructors are called.

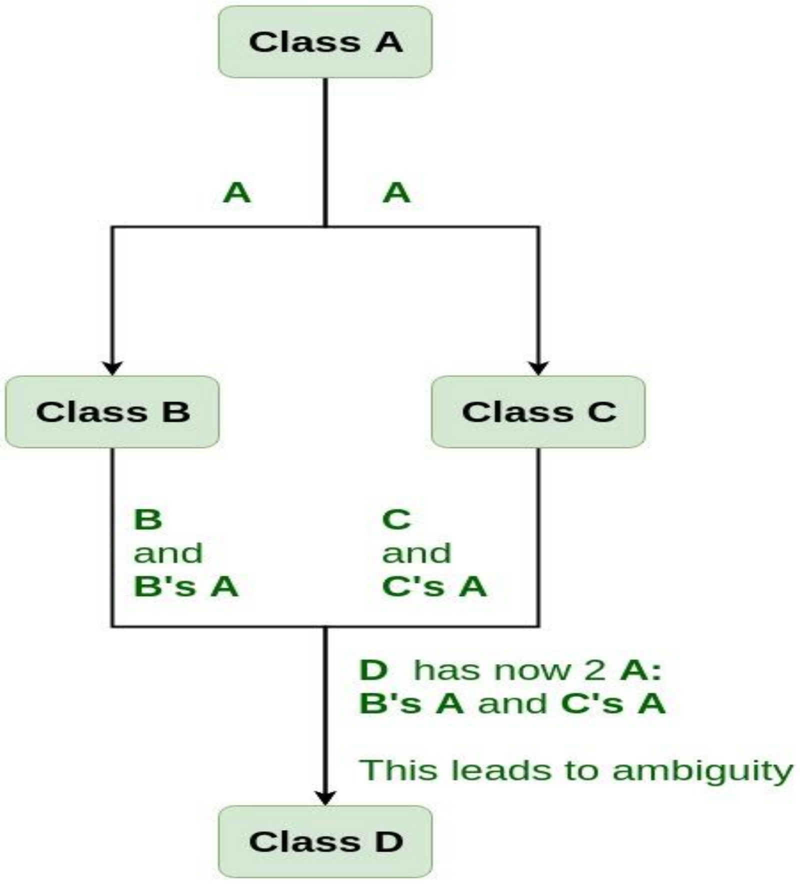
1 word Question Answer

Sr No.	Question	Answer
1.	________is the process of creating new classes, called derived classes, from existing classes, which are often called base classes.	Inheritance
2.	________ inheritance represents a form of inheritance when there is only one base class and one derived class.	Single
3.	________inheritance represents a kind of inheritance when a derived class inherits properties of multiple classes.	Multiple
4.	________inheritance represents a type of inheritance when a Derived class is a base class for another class.	Multilevel
5.	When there is a need to create multiple Derived classes that inherit properties of the same Base class is known as ________ inheritance.	Hierarchical
6.	Combination of Multi-level and Hierarchical inheritance will give you ________ inheritance.	Hybrid

Topic: Virtual base class

- ✓ Virtual base classes are used in virtual inheritance in a way of preventing multiple "instances" of a given class appearing in an inheritance hierarchy when using multiple inheritances.
- ✓ **Need for Virtual Base Classes:**
  Consider the situation where we have one class A .This class is A is inherited by two other

classes B and C. Both these class are inherited into another in a new class D as shown in figure below.



- ✓ As we can see from the figure that data members/function of class A are inherited twice to class D.
- ✓ One through class B and second through class C. When any data / function member of class A is accessed by an object of class D, ambiguity arises as to which data/function member would be called?
- ✓ One inherited through B or the other inherited through C. This confuses compiler and it displays error.

Example: To show the need of Virtual Base Class in C++

```
#include <iostream>
using namespace std;

class A {
public:
  void show()
  {
    cout << "Hello form A \n";
  }
};

class B : public A {
};

class C : public A {
};

class D : public B, public C {
};

int main()
{
  D object;
  object.show();
}
```

Compile Errors:

```
prog.cpp: In function 'int main()':

prog.cpp:29:9: error: request for member 'show' is ambiguous

  object.show();

         ^

prog.cpp:8:8: note: candidates are: void A::show()

   void show()

        ^

prog.cpp:8:8: note:                 void A::show()
```

- ✓ **How to resolve this issue?**
  To resolve this ambiguity when class A is inherited in both class B and class C, it is declared as virtual base class by placing a keyword virtual as :
- ✓ **Syntax for Virtual Base Classes:**

Syntax 1:

```
class B : virtual public A
```

```
{
};
```

Syntax 2:

```
class C : public virtual A
{
};
```

- ✓ **Note**: virtual can be written before or after the public. Now only one copy of data/function member will be copied to class C and class B and class A becomes the virtual base class.
- ✓ Virtual base classes offer a way to save space and avoid ambiguities in class hierarchies that use multiple inheritances.
- ✓ When a base class is specified as a virtual base, it can act as an indirect base more than once without duplication of its data members.
- ✓ A single copy of its data members is shared by all the base classes that use virtual base.

1 word Question Answer

Sr No.	Question	Answer
1.	_______ classes are used in virtual inheritance in a way of preventing multiple "instances" of a given class appearing in an inheritance hierarchy when using multiple inheritances.	Virtual base

Topic: Abstract Classes

- ✓ Abstract classes act as expressions of general concepts from which more specific classes can be derived. You cannot create an object of an abstract class type; however, you can use pointers and references to abstract class types.

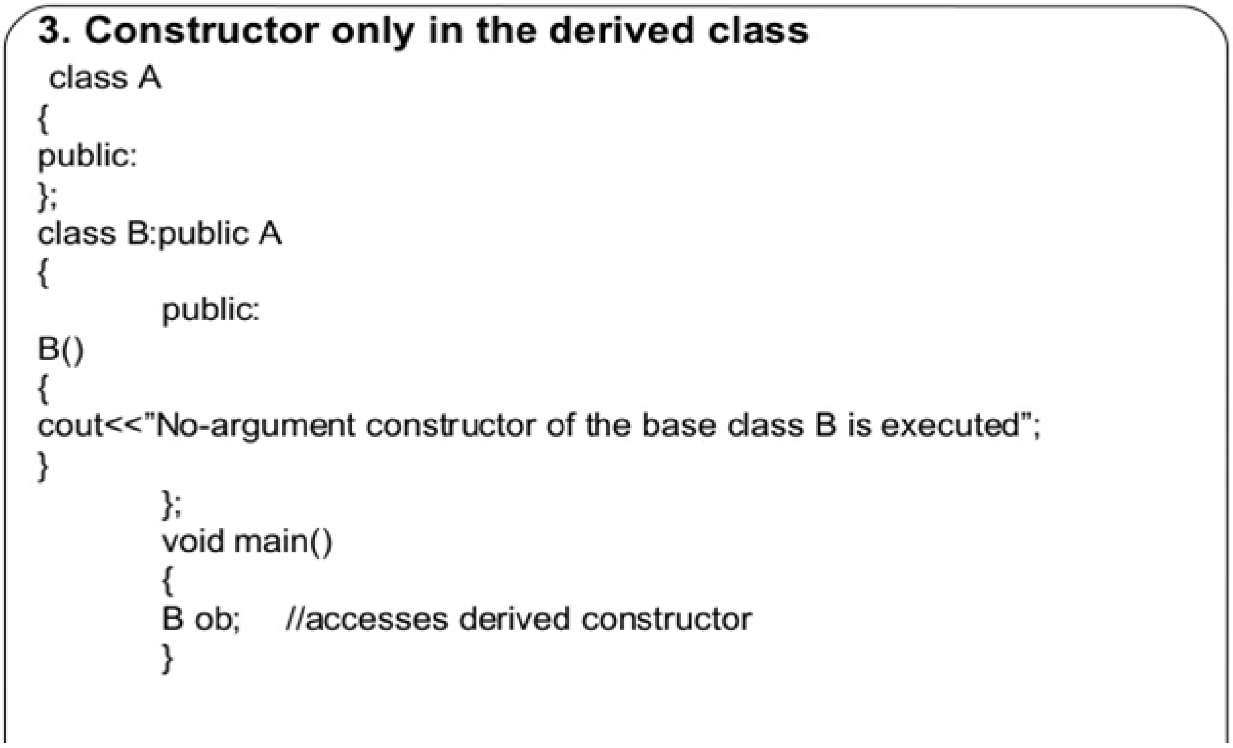
- ✓ A class that contains at least one pure virtual function is considered an abstract class. Classes derived from the abstract class must implement the pure virtual function or they, too, are abstract classes.
- ✓ Consider the example presented in Virtual Functions. The intent of class Account is to provide general functionality, but objects of type Account are too general to be useful. Therefore, Account is a good candidate for an abstract class:

```
// deriv_AbstractClasses.cpp
// compile with: /LD
class Account {
public:
   Account( double d );   // Constructor.
   virtual double GetBalance();   // Obtain balance.
   virtual void PrintBalance() = 0;   // Pure virtual function.
private:
    double _balance;
};
```

- ✓ The only difference between this declaration and the previous one is that Print Balance is declared with the pure specifier (= 0).

Topic: Constructor in derived class

- ✓ The derived class need not have a constructor as long as the base class has no-argument constructor.
- ✓ However, if the base class has constructors with arguments (one or more), then it is mandatory for the derived class to have a constructor and pass the arguments to the base class constructor.
- ✓ When an object of a base class is created, the constructor of the base class is executed first and later the constructor of the derived class.



3. Constructor only in the derived class

```
 class A
{
public:
};
class B:public A
{
          public:
B()
{
cout<<"No-argument constructor of the base class B is executed";
}
          };
          void main()
          {
          B ob;    //accesses derived constructor
          }
```

1 word Question Answer

Sr No.	Question	Answer
1.	You cannot create an object of an abstract class type; however, you can use pointers and references to _______ types.	abstract class

Topic: Containership

- ✓ We can create an object of one class into another and that object will be a member of the class.
- ✓ This type of relationship between classes is known as containership or has_a relationship as one class contain the object of another class.
- ✓ And the class which contains the object and members of another class in this kind of relationship is called a container class.
- ✓ The object that is part of another object is called contained object, whereas object that contains another object as its part or attribute is called container object.
- ✓ **Difference between containership and inheritance**
- ✓ **Containership**
  -> When features of existing class are wanted inside your new class, but, not its interface
  for eg->
  1) computer system has a hard disk
  2) car has an Engine, chassis, steering wheels.
- ✓ **Inheritance**
  -> When you want to force the new type to be the same type as the base class.
  for eg->
  1)computer system is an electronic device
  2)Car is a vehicle

1 word Question Answer

Sr No.	Question	Answer
1.	class which contains the object and members of another class in this kind of relationship is called a _________.	container class
2.	We can create an object of one class into another and that object will be a member of the class.This type of relationship between classes is known as _________.	containership

Ch – 4 Pointer, Virtual functions and Polymorphism, RTTI Console I/O operations

TOPIC: POINTER TO OBJECTS:

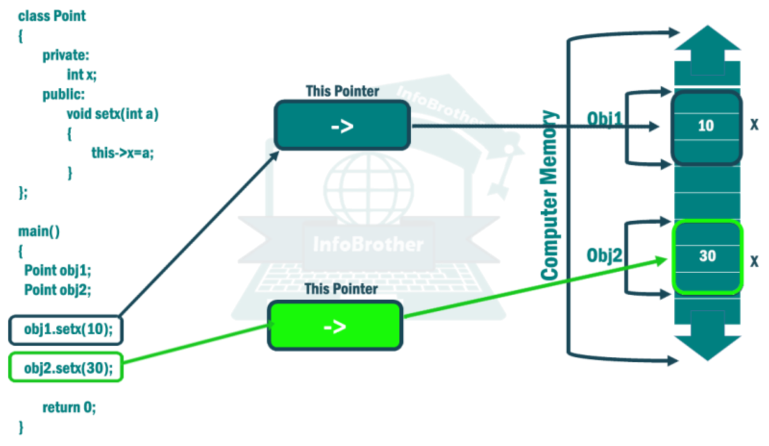
- ✓ A Variable That Holds an Address value is called a Pointer variable or simply pointer.
- ✓ We already discussed about pointer that's point to Simple data types likes int, char, float etc. So similar to these type of data type, Objects can also have an address, so there is also a pointer that can point to the address of an Object, This Pointer is Known as **This Pointer.**

1 word Question Answer

Sr No.	Question	Answer
1.	A Variable That Holds an Address value is called a___________.	Pointer variable or simply pointer

THIS POINTER:

- ✓ Every Object in C++ has access to its own address through an important pointer called T h i s Pointer.
- ✓ The This Pointer is an implicit parameter to all member functions. Therefore, inside a member function, this may be used to refer to the invoking object. Whenever a member function is called, it is automatically passed an implicit arguments that is This pointer to the invoking object *(i.e. The object on which the function is invoked).*
- ✓ The This pointer is passed as a hidden argument to all Nonstatic member function calls and is available as a local variable within the body of all Nonstatic functions. This Pointer is a constant pointer that holds the memory address of the current object. This pointer is not available in static member functions as static member functions can be called without any object (with class name).



```
class Point
{
    private:
        int x;
    public:
        void setx(int a)
        {
            this->x=a;
        }
};

main()
{
 Point obj1;
 Point obj2;
 obj1.setx(10);
 obj2.setx(30);

    return 0;
}
```

1 word Question Answer

Sr No.	Question	Answer
1.	Every Object in C++ has access to its own address through an important pointer called ________.	This Pointer

TOPIC: VIRTUAL FUNCTION:

- ✓ A Virtual Function is a function that is declared as Virtual in a base class and redefined in one or more derived classes.
- ✓ Thus, each derived class can have its own version of a Virtual Function. If there is Member functions with same name in base class and derived class, Virtual Functions gives Programmer Capability to call member function of different class by a same function call depending upon different context.
- ✓ This feature in C++ Programming is known as Polymorphism which is one of the important features of Object Oriented Programming.
- ✓ If a base Class and derived Class have same function and if we write code to access that function using pointer of base class then, the function in the base class is executed even, if the object of derived class is referenced with that pointer variable.
- ✓ SYNTAX:
    - o Virtual returntype funname(args);
- ✓ Ex:
    - o Virtual void input();

TOPIC: PURE VIRTUAL FUNCTION:

- ✓ In C++ Programming, sometimes inheritance is used only for the better visualization of data and we don't need to create any object of base class.
- ✓ For example: if we want to calculate area of different objects like Circle and Square then, we can inherit these classes from a shape, because it helps to visualize the problem but, we don't need to create any object of Shape.
- ✓ in such case, we can declare Shape as abstract class if we try to create object of a Abstract class, compiler shows error.
- ✓ A pure Virtual function or abstract function in C++ is a virtual function for which we don't have implementation, we only declare it.
- ✓ If Expression =0 is added to a virtual function then, that function is becomes pure Virtual function.
- ✓ Syntax:
  - Virtual returntype funname()=0;

Ex:

Virtual void area()=0;

1 word Question Answer

Sr No.	Question	Answer
1.	If Expression =0 is added to a virtual function then, that function is becomes _______.	Pure Virtual function

TOPIC: RULES FOR PURE VIRTUAL FUNCTION

1. When a virtual function in base class is created, there must be definition of the virtual function in base class even it base class version of the function is never actually called.
2. The virtual function must be members of some class.
3. They cannot be static member.
4. They can be a friend function to another class.
5. They are accessed using object pointers.
6. A base pointer can serve as a pointer to a derived object because it is type compatible whereas the derived object pointer variable cannot serve as a pointer to base objects.
7. Its prototype in base class and derived class must be same for the virtual function to work properly.
8. The class cannot have virtual constructor.

9. When a base pointer points to derived class, incrementing or decrementing it will not make it to point to the next object of derived class.
10. They should be declared in the public section of a class.

TOPIC: RTTI (RUN-TIME TYPE INFORMATION) IN C++

- ✓ In C++, RTTI (Run-time type information) is a mechanism that exposes information about an object's data type at runtime and is available only for the classes which have at least one virtual function.
- ✓ It allows the type of an object to be determined during program execution.

Adding a virtual function to the base class B makes it working.

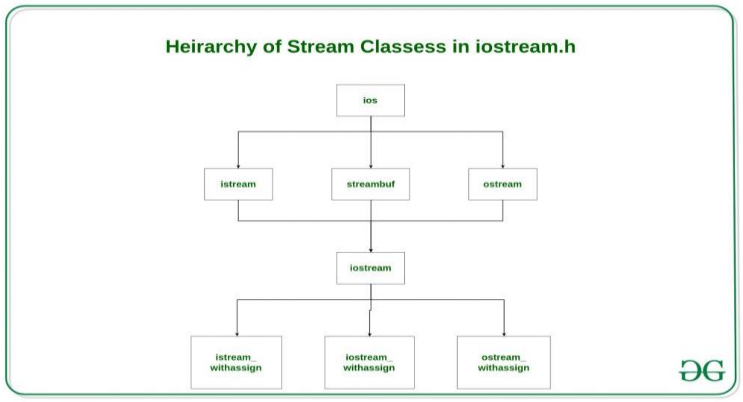
```
// CPP program to illustrate
// Run Time Type Identification
#include<iostream>
using namespace std;
class B { virtual void fun() {} };
class D: public B { };

int main()
{
  B *b = new D;
  D *d = dynamic_cast<D*>(b);
  if(d != NULL)
    cout << "works";
  else
    cout << "cannot cast B* to D*";
  getchar();
  return 0;
}
```

1 word Question Answer

Sr No.	Question	Answer
1.	**RTTI STANDS FOR?**	**RUN-TIME TYPE INFORMATION**

TOPIC: C++ STREAM CLASSES

- ✓ In C++ there are number of stream classes for defining various streams related with files and for doing input-output operations.
- ✓ All these classes are defined in the file iostream.h. Figure given below shows the hierarchy of these classes.



1. ios class is topmost class in the stream classes hierarchy. It is the base class for istream, ostream, and streambuf class.
2. istream and ostream serves the base classes for iostream class. The class istream is used for input and ostream for the output.
3. Class ios is indirectly inherited to iostream class using istream and ostream.
4. The _withassign classes are provided with extra functionality for the assignment operations that's why _withassign classes.
   1. **The ios class**: The ios class is responsible for providing all input and output facilities to all other stream classes.
   2. **The istream class**: This class is responsible for handling input stream. It provides number of function for handling chars, strings and objects such as get, getline, read etc.
   3. **The ostream class:** This class is responsible for handling output stream. It provides number of function for handling chars, strings and objects such as write, put etc.
   4. **The iostream**: This class is responsible for handling both input and output stream as both istream class and ostream class is inherited into it. It provides function of both

istream class and ostream class for handling chars, strings and objects such as get, getline, read, ignore, putback, put, write etc..

5. **istream_withassign class**: This class is variant of istream that allows object assigment. The predefined object cin is an object of this class and thus may be reassigned at run time to a different istream object.
6. **ostream_withassign class**: This class is variant of ostream that allows object assigment. The predefined objects cout, cerr, clog are objects of this class and thus may be reassigned at run time to a different ostream object.
7. **Streambuf class:** A *stream buffer* is an object in charge of performing the reading and writing operations of the stream object it is associated with: the stream delegates all such operations to its associated stream buffer object, which is an intermediary between the stream and its controlled input and output sequences. All stream objects, no matter whether buffered or unbuffered, have an associated stream buffer: Some stream buffer types may then be set to either use an intermediate buffer or not.

1 word Question Answer

Sr No.	Question	Answer
1.	The ______ is responsible for providing all input and output facilities to all other stream classes.	The ios class
2.	_______ class is responsible for handling input stream.	The istream class
3.	_______ class is responsible for handling output stream.	The ostream class
4.	_______ class is responsible for handling both input and output stream	The iostream class
5.	_____ class is variant of istream that allows object assigment.	istream_withassign class
6.	______ class is variant of ostream that allows object assigment.	ostream_withassign class

TOPIC: UNFORMATTED AND FORMATTED I/O OPERATIONS

UNFORMATTED I/O OPERATIONS

- ✓ The classes istream and ostream defines two member functions get() and put() respectively to handle single character input/output operations.

  1. get(char *)
  2. put(void)

  - ✓ Both of them can be used to fetch a character including blank space, tab or new-line character.

```
cin.get(ch);
 while(ch != '\n')
 {
   cout<<ch;
   cin.get(ch);
 }
```

- ✓ Similarly, the function put(), a member of a stream class can be used to output a line of text character by character.

```
cout.put ('g');
char ch;
cout.put(ch);
```

getline() and write()

- ✓ You can read and display lines of text more efficiently using the lie oriented input/output functions. They are:
- ✓ getline()
- ✓ write()
- ✓ The getline() function reads the entire line of texts that ends with a newline character. The general form of getline() is:

```
cin.getline (line, size);
```

- ✓ The write() function displays the entire line of text, and the general form of writing this function is:

```
cout.write (line, size);
```

FORMATTED I/O OPERATIONS

Width()

- ✓ The default width of your output will be just enough space to print the number, character, or string in the output buffer. You can change this by using width(). Because width() is a member function, it must be invoked with a cout object.
- ✓ Ex:

```
cout << "Start :";
cout.width(25);
 cout << 123
```

precision()

- ✓ Sets the decimal precision to be used to format floating-point values on output operations.
- ✓ Behaves as if member precision were called with n as argument on the stream on which it is inserted/extracted as a manipulator (it can be inserted/extracted on input streams or output streams).
- ✓ Ex:

```
Float f=12.234556
cout.precision(3);
cout<<f;
```

fill()

- ✓ fill() function is a library function of algorithm header, it is used to assign a value to the all elements within a given range of a container, it accepts iterators pointing to the starting and ending position in the container and a value to be assigned to the elements within the given range, and assigns the value.

- ✓ Ex:

```
Cout.fill(‘*’);
Cout.width(5);
Cout<<50;
```

Manipulators

- ✓ Manipulators are used along with the extraction >> and insertion << operators for stream input and output formatting.
- ✓ According to the number of arguments to be supplied manipulators are categorized in the following types.

1. Non-parameterized Manipulators
2. Parameterized Manipulators

- ✓ Non parameterized Manipulators do no take argument to control the formatting of input/output where as parameterized manipulators take argument for formatting.
- ✓ The Following table shows different non parameterized manipulators.

Manipulator	**Effect produced**
left	sets ios::left flag of ios::adjustfield
right	sets ios::right flag of ios::adjustfield
dec	sets ios::dec flag of ios::adjustfield
hex	sets ios::hex flag of ios::adjustfield
oct	sets ios::oct flag of ios::adjustfield
endl	output newline and flush
ends	Insert null character - ouput ‘’ character

- ✓ Following table shows different parameterized manipulators

Manipulators	**Effect produced**

setw(int n)	equivalent to ios function width()
setprecision (int n)	equivalent to ios function precision()
setfill(int c)	equivalent to ios::function fill()

1 word Question Answer

Sr No.	Question	Answer
1.	Which function is Sets the decimal precision to be used to format floating-point values on output operations?	precision()
2.	_______ function is a library function of algorithm header.	fill()
3.	_________are used along with the extraction >> and insertion << operators for stream input and output formatting.	Manipulators

Ch – 5 Working with Files, Exception handling, Introduction to Template STL

TOPIC: FILE HANDLING THROUGH C++ CLASSES

- ✓ In C++, files are mainly dealt by using three classes fstream, ifstream, ofstream available in fstream headerfile.
  ofstream: Stream class to write on files
  ifstream: Stream class to read from files
  fstream: Stream class to both read and write from/to files.
- ✓ Now the first step to open the particular file for read or write operation. We can open file by
  1. passing file name in constructor at the time of object creation
  2. using the open method
  For e.g.
- ✓ Open File by using constructor
  ifstream (const char* filename, ios_base::openmode mode = ios_base::in);
  ifstream fin(filename, openmode) by default openmode = ios::in
  ifstream fin(“filename”);
- ✓ Open File by using open method
  Calling of default constructor
  ifstream fin;
- ✓ fin.open(filename, openmode)
  fin.open(“filename”);

Modes :

MEMBER CONSTANT	STANDS FOR	ACCESS
in	input	File open for reading: the internal stream buffer supports input operations.
Out	output	File open for writing: the internal stream buffer supports output operations.
binary	binary	Operations are performed in binary mode rather than text.
Ate	at end	The output position starts at the end of the file.
App	append	All output operations happen at the end of the file, appending to its existing contents.
Trunc	truncate	Any contents that existed in the file before it is open are discarded.

Default Open Modes :

Ifstream	ios::in
Ofstream	ios::out
Fstream	ios::in | ios::out

1 word Question Answer

Sr No.	Question	Answer
1.	________ Stream class to write on files	Ofstream
2.	________ Stream class to read from files	Ifstream

TOPIC: OPENING A FILE

- ✓ A file must be opened before you can read from it or write to it. Either ofstream or fstream object may be used to open a file for writing. And ifstream object is used to open a file for reading purpose only.
- ✓ Following is the standard syntax for open() function, which is a member of fstream, ifstream, and ofstream objects.
- ✓ void open(const char *filename, ios::openmode mode);
- ✓ Here, the first argument specifies the name and location of the file to be opened and the second argument of the open() member function defines the mode in which the file should be opened.
- ✓ You can combine two or more of these values. For example if you want to open a file in write mode and want to truncate it in case that already exists, following will be the syntax –
- ✓ ofstream outfile;
- ✓ outfile.open("file.dat", ios::out | ios::trunc);

- ✓ Similar way, you can open a file for reading and writing purpose as follows −
- ✓ fstream afile;
- ✓ afile.open("file.dat", ios::out | ios::in);

1 word Question Answer

Sr No.	Question	Answer
1.	Either _______ or _______ object may be used to open a file for writing.	Ofstream, fstream
2.	_________ object is used to open a file for reading purpose only.	Ifstream

TOPIC: CLOSING A FILE

- ✓ When a C++ program terminates it automatically flushes all the streams, release all the allocated memory and close all the opened files. But it is always a good practice that a programmer should close all the opened files before program termination.
- ✓ Following is the standard syntax for close() function, which is a member of fstream, ifstream, and ofstream objects.
- ✓ void close();

1 word Question Answer

Sr No.	Question	Answer
1.	C++ program terminates it automatically flushes all the _________.	Streams

TOPIC: WRITING TO A FILE

- ✓ While doing C++ programming, you write information to a file from your program using the stream insertion operator (<<) just as you use that operator to output information to the screen. The only difference is that you use an ofstream or fstream object instead of the cout object.

Reading from a File

- ✓ You read information from a file into your program using the stream extraction operator (>>) just as you use that operator to input information from the keyboard. The only difference is that you use an ifstream or fstream object instead of the cin object.

File Position Pointers

- ✓ we have a get pointer and a put pointer for getting (i.e. reading) data from a file and putting(i.e. writing) data on the file respectively.
- ✓ seekg() is used to move the get pointer to a desired location with respect to a reference point.
- ✓ Syntax: file_pointer.seekg (number of bytes ,Reference point);
- ✓ Example: fin.seekg(10,ios::beg);
- ✓ tellg() is used to know where the get pointer is in a file.
- ✓ Syntax: file_pointer.tellg();
- ✓ Example: int posn = fin.tellg();
- ✓ seekp() is used to move the put pointer to a desired location with respect to a reference point.
- ✓ Syntax: file_pointer.seekp(number of bytes ,Reference point);
- ✓ Example: fout.seekp(10,ios::beg);
- ✓ tellp() is used to know where the put pointer is in a file.
- ✓ Syntax: file_pointer.tellp();
- ✓ Example: int posn=fout.tellp();
- ✓ The reference points are:
- ✓ ios::beg – from beginning of file
- ✓ ios::end – from end of file
- ✓ ios::cur – from current position in the file.

- ✓ In seekg() and seekp() if we put – sign in front of number of bytes then we can move backwards.

1 word Question Answer

Sr No.	Question	Answer
1.	______ is used to move the get pointer to a desired location with respect to a reference point.	seekg()
2.	______is used to know where the get pointer is in a file.	tellg()
3.	_______is used to move the put pointer to a desired location with respect to a reference point.	seekp()
4.	_______ is used to know where the put pointer is in a file.	tellp()

TOPIC: SEQUENTIAL I/O OPERATIONS

- ✓ The file stream classes support a number of member functions for performing the input and output operations on files.
- ✓ The functions get() and put() are capable of handling a single character at a time. The function getline() lets you handle multiple characters at a time.
- ✓ Another pair of functions i.e., read() and write() are capable of reading and writing blocks of binary data.

The get(), getline() and put() Functions

- ✓ The functions get() and put() are byte-oriented. That is, get() will read a byte of data and put() will write a byte of data. The get() has many forms, but the most commonly used version is shown here, along with put() :
- ✓ istream & get(char & ch) ;
- ✓ ostream & put(char ch) ;

- ✓ The get() function reads a single character from the associated stream and puts that value in ch. It returns a reference to the stream. The put() writes the value of ch to the stream and returns a reference to the stream.

The read() and write() Functions

- ✓ Another way of reading and writing blocks of binary data is to use C++'s read() and write() functions. Their prototypes are :
- ✓ istream & read((char *) & buf, int sizeof(buf)) ;
- ✓ ostream & write((char *) & buf, int sizeof(buf)) ;
- ✓ The read() function reads sizeof(buf) (it can be any other integer value also) bytes from the associated stream and puts them in the buffer pointed to by buf. The write() function writes sizeof(buf) (it can be any other integer value also) bytes to the associated stream from the buffer pointed to by buf.
- ✓ As you see, these functions take two arguments. The first is the address of variable buf, and the second is the length of that variable in bytes. The address of the variable must be type cast to typ
- ✓ e char * (i.e., a pointer to character type).

1) Write a c++ program for enter file name and copy one file content to another file.

```
#include<iostream.h>
#include<conio.h>
#include<fstream.h>
#include<stdio.h>
#include<stdlib.h>
void main()
{
    clrscr();
    ifstream fs;
    ofstream ft;
    char ch, fname1[20], fname2[20];
    cout<<"Enter source file name with extension (like files.txt) : ";
    gets(fname1);
    fs.open(fname1);
    if(!fs)
    {
        cout<<"Error in opening source file..!!";
        getch();
```

```
            exit(1);
        }
        cout<<"Enter target file name with extension (like filet.txt) : ";
        gets(fname2);
        ft.open(fname2);
        if(!ft)
        {
            cout<<"Error in opening target file..!!";
            fs.close();
            getch();
            exit(2);
        }
        while(fs.eof()==0)
        {
            fs>>ch;
            ft<<ch;
        }
        cout<<"File copied successfully..!!";
        fs.close();
        ft.close();
        getch();
}
```

2) Write a c++ program for enter the file name from command line argument and write the content in that file.

```
#include<iostream.h>
#include<conio.h>
#include<fstream.h>
int main(int argc, char *argv[])
{
        int i;
        char ch[50];
        cout<<endl<<"Total arguments="<<argc;
        cout<<endl<<"Program name is="<<argv[0];
        ofstream out;
        out.open(argv[1],ios::out);
        out<<"This is demo file for Command line Arguments: ";
        out.close();
        getch();
        return(0);
}
```

how to run this file:

Step 1: open dos shell from file menu...

Step 2: set the path c:\turboc3\source folder with cd for enter directory and cd.. for came out for last directory....

Step 3: give the file for run the program without extension and give the another text file name

for open and write the file....
Like c:\turboc3\source> cmdfile demo.txt

3) Write a c++ program for merge two file in third file.

```
/* C++ Program - Merge Two Files */


#include<iostream.h>

#include<conio.h>

#include<fstream.h>

#include<stdio.h>

#include<stdlib.h>

void main()

{

        clrscr();

        ifstream ifiles1, ifiles2;

        ofstream ifilet;

        char ch, fname1[20], fname2[20], fname3[30];

        cout<<"Enter first file name (with extension like file1.txt) : ";

        gets(fname1);

        cout<<"Enter second file name (with extension like file2.txt) : ";

        gets(fname2);

        cout<<"Enter name of file (with extension like file3.txt) which will store the contents of
the two files (fname1 and fname2) : ";

        gets(fname3);

        ifiles1.open(fname1);

        ifiles2.open(fname2);

        if(ifiles1==NULL || ifiles2==NULL)

        {
```

```
        perror("Error Message ");
        cout<<"Press any key to exit...\n";
        getch();
        exit(EXIT_FAILURE);
}
ifilet.open(fname3);
if(!ifilet)
{
        perror("Error Message ");
        cout<<"Press any key to exit...\n";
        getch();
        exit(EXIT_FAILURE);
}
while(ifiles1.eof()==0)
{
        ifiles1>>ch;
        ifilet<<ch;
}
while(ifiles2.eof()==0)
{
        ifiles2>>ch;
        ifilet<<ch;
}
cout<<"The two files were merged into "<<fname3<<" file successfully..!!";
ifiles1.close();
ifiles2.close();
```

```
        ifilet.close();

        getch();

    }
```

1 word Question Answer

Sr No.	Question	Answer
1.	_____ will read a byte of data and _____will write a byte of data.	get(),put()

TOPIC: EXCEPTION HANDLING...

- ✓ An exception is a problem that arises during the execution of a program. A C++ exception is a response to an exceptional circumstance that arises while a program is running, such as an attempt to divide by zero.
- ✓ Exceptions provide a way to transfer control from one part of a program to another. C++ exception handling is built upon three keywords: try, catch, and throw.
- ✓ throw − A program throws an exception when a problem shows up. This is done using a throw keyword.
- ✓ catch − A program catches an exception with an exception handler at the place in a program where you want to handle the problem. The catch keyword indicates the catching of an exception.
- ✓ try − A try block identifies a block of code for which particular exceptions will be activated. It's followed by one or more catch blocks.
- ✓ Assuming a block will raise an exception, a method catches an exception using a combination of the try and catch keywords. A try/catch block is placed around the code that might generate an exception. Code within a try/catch block is referred to as protected code, and the syntax for using try/catch as follows −

```
        try {
```

```
    // protected code
  } catch( ExceptionName e1 ) {
    // catch block
  } catch( ExceptionName e2 ) {
    // catch block
  } catch( ExceptionName eN ) {
    // catch block
  }
```

- ✓ You can list down multiple catch statements to catch different type of exceptions in case your try block raises more than one exception in different situations.

1 word Question Answer

Sr No.	Question	Answer
1.	An ________is a problem that arises during the execution of a program.	Exception
2.	A program throws an exception when a problem shows up. This is done using a _______ keyword.	throw
3.	The ______ keyword indicates the catching of an exception.	catch

TOPIC: THROWING EXCEPTIONS

- ✓ Exceptions can be thrown anywhere within a code block using throw statement. The operand of the throw statement determines a type for the exception and can be any expression and the type of the result of the expression determines the type of exception thrown.
- ✓ Following is an example of throwing an exception when dividing by zero condition occurs –

```
double division(int a, int b)
{
   if( b == 0 ) {
      throw "Division by zero condition!";
   }
   return (a/b);
}
```

Catching Exceptions

- ✓ The catch block following the try block catches any exception. You can specify what type of exception you want to catch and this is determined by the exception declaration that appears in parentheses following the keyword catch.

```
try {
   // protected code
} catch( ExceptionName e ) {
  // code to handle ExceptionName exception
}
```

- ✓ Above code will catch an exception of ExceptionName type. If you want to specify that a catch block should handle any type of exception that is thrown in a try block, you must put an ellipsis, ..., between the parentheses enclosing the exception declaration as follows −

```
try {
   // protected code
} catch(...) {
  // code to handle any exception
}
```

- ✓ The following is an example, which throws a division by zero exception and we catch it in catch block.

```
#include <iostream>
using namespace std;

double division(int a, int b) {
   if( b == 0 ) {
      throw "Division by zero condition!";
   }
   return (a/b);
}
```

```
int main () {
  int x = 50;
  int y = 0;
  double z = 0;

  try {
    z = division(x, y);
    cout << z << endl;
  } catch (const char* msg) {
    cerr << msg << endl;
  }

  return 0;
}
```

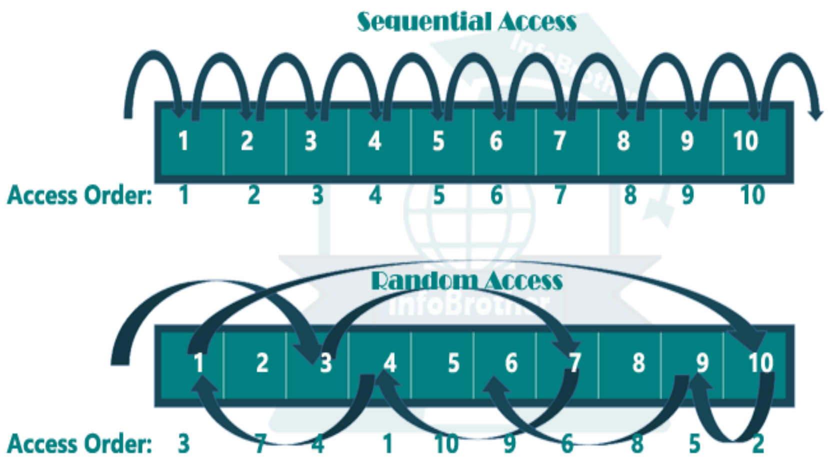
1 word Question Answer

Sr No.	Question	Answer
1.	Exceptions can be thrown anywhere within a code block using _______ statement.	throw

TOPIC: RANDOM-ACCESS-FILE:

- ✓ In This Tutorial, We are Introduction the concept of Random file Access. Unlike Sequential files, we can access Random Access files in any order we want.
- ✓ Think of data in a Random Access file as we would songs on a compact disc or record, we can go directly to any song we want without having to play or fast-forward over the other songs.
- ✓ If we want to play the first song, the sixth song, and then the fourth song, we can do so. The order of play has nothing to do with the order in which the songs were originally recorded.
- ✓ Random-file access sometimes takes more Programming but rewards our effort with a more flexible file-access method .
- ✓ Random file access enables us to read or write any data in our disk file without having to read or write every piece of data before it.

- ✓ We can Quickly search for data, Modify data, delete data in a random-access file.
- ✓ We can open and close Random access file same like Sequential files with same opening mode, but we need a few new functions to access files randomly, we find that the extra effort pays off in flexibility, power, and speed of disk access:



- ✓ The process of randomly accessing data in a file is simple. Think about the data files of a large credit card organization.
- ✓ When we make a purchase, the store calls the credit card company to receive authorization.
- ✓ Millions of names are in the credit card company's files. There is no quick way the credit card company could read every record sequentially from the disk that comes before Ours. Sequential files do not lend themselves to quick access.
- ✓ It is not feasible, in many situations, to look up individual records in a data file with sequential access.
- ✓ The credit card companies must use a random file access so their computers can go directly to our record, just as we go directly to a song on a compact disk or record album.
- ✓ The functions we use are different from the sequential functions, but the power that results from learning the added functions is worth the effort.
- ✓ When our program reads and writes files Randomly, it treats the file like a big Array. with Arrays, we know we can add, print, or remove values in any order.

- ✓ we don't have to start at the first array element, Sequentially looking at the Next one, Until we get the Element we need.
- ✓ We can view our Random-access-file in the same way, accessing the data in any order. In Random Access file, we can:
- ✓ » We Can Read from Anywhere from the file.
  » We Can Write to Anywhere in the file.
  » We can Modify our record:
  » We can Search any Record from file
  » And we can Delete Any record from file:
- ✓ For doing this all operation, we have just one Important things that we need to know about before Move on.
- ✓ The thing is file-pointer, this File-Pointer move sequentially but we can also make it to move Randomly using some function.
- ✓ C++ provide us four function, using these function can help us to set this file-pointer to any where in the file:

Special function to move File-Pointer within the File:

Function	**Syntax**	**Explanation**
seekg()	fileObject.seekg(long_num, origin);	We can move input pointer to a specified location for reading by using this function. fileObject is the pointer to the file that we want to access. long_num is the number of bytes in the file we want to skip. and origin is a value that tells to compiler, where to begin the skipping of bytes.
seekp()	fileObject.seekp(long_num, origin);	We can move Output pointer to a specified location by using this function. it's same like seekg() Function but in Case of writing:

tellg()	fileObject.tellg();	This function return the current position of Input pointer. this function don't need any argument.
tellp()	fileObject.tellp();	This function return the current position of Output pointer. this function don't need any argument.

1 word Question Answer

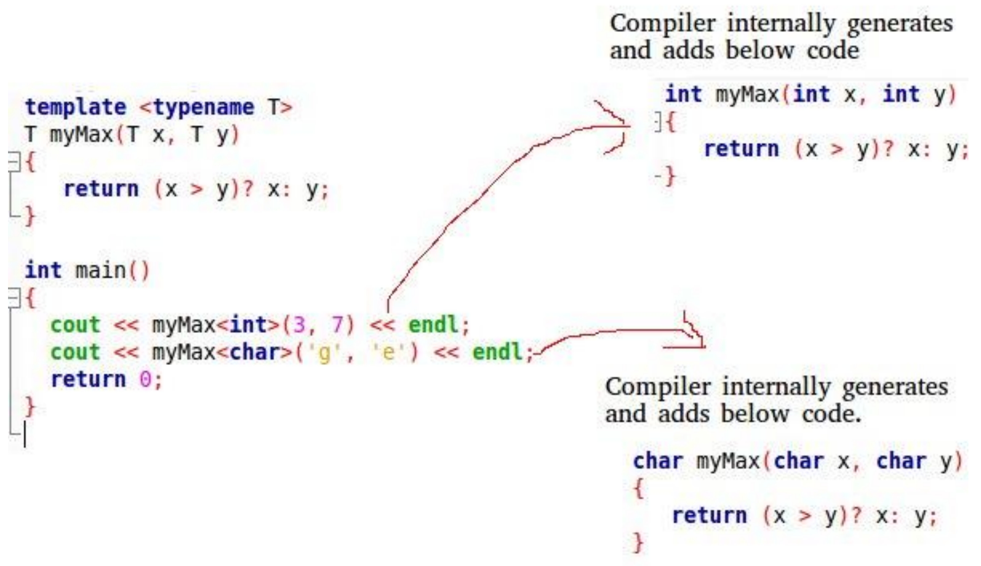
Sr No.	Question	Answer
1.	________enables us to read or write any data in our disk file without having to read or write every piece of data before it.	Random file access

TOPIC: TEMPLATES IN C++

- ✓ A template is a simple and yet very powerful tool in C++.
- ✓ The simple idea is to pass data type as a parameter so that we don’t need to write the same code for different data types.
- ✓ For example, a software company may need sort() for different data types. Rather than writing and maintaining the multiple codes, we can write one sort() and pass data type as a parameter.
- ✓ C++ adds two new keywords to support templates: ‘template’ and ‘typename’. The second keyword can always be replaced by keyword ‘class’.
- ✓ How templates work?

  Templates are expanded at compiler time. This is like macros. The difference is, compiler does type checking before template expansion. The idea is simple, source code contains only function/class, but compiled code may contain multiple copies of same function/class.

Function template



```
template <typename T>
T myMax(T x, T y)
{
    return (x > y)? x: y;
}

int main()
{
    cout << myMax<int>(3, 7) << endl;
    cout << myMax<char>('g', 'e') << endl;
    return 0;
}
```

Compiler internally generates and adds below code

```
int myMax(int x, int y)
{
    return (x > y)? x: y;
}
```

Compiler internally generates and adds below code.

```
char myMax(char x, char y)
{
    return (x > y)? x: y;
}
```

- ✓ Function Templates We write a generic function that can be used for different data types. Examples of function templates are sort(), max(), min(), printArray(). Know more on Generics in C++

```
#include <iostream>
using namespace std;
// One function works for all data types.  This would work
// even for user defined types if operator '>' is overloaded
template <typename T>
T myMax(T x, T y)
{
  return (x > y)? x: y;
}
int main()
```

```
{
  cout << myMax<int>(3, 7) << endl;  // Call myMax for int
  cout << myMax<double>(3.0, 7.0) << endl; // call myMax for double
  cout << myMax<char>('g', 'e') << endl;   // call myMax for char


  return 0;
}
```

Function template overloading

- ✓ Function templates and non-template functions may be overloaded.
- ✓ A non-template function is always distinct from a template specialization with the same type. Specializations of different function templates are always distinct from each other even if they have the same type.
- ✓ Two function templates with the same return type and the same parameter list are distinct and can be distinguished with explicit template argument list.
- ✓ When an expression that uses type or non-type template parameters appears in the function parameter list or in the return type, that expression remains a part of the function template signature for the purpose of overloading:
- ✓ For ex:

```
template<int I, int J> A<I+J> f(A<I>, A<J>); // overload #1

template<int K, int L> A<K+L> f(A<K>, A<L>); // same as #1

template<int I, int J> A<I-J> f(A<I>, A<J>); // overload #2
```

Class Templates

- ✓ Class Templates Like function templates, class templates are useful when a class defines something that is independent of the data type. Can be useful for classes like LinkedList, BinaryTree, Stack, Queue, Array, etc.
- ✓ Following is a simple example of template Array class.

```
#include <iostream>

using namespace std;
```

```
template <typename T>
class Array {
private:
   T *ptr;
   int size;
public:
   Array(T arr[], int s);
   void print();
};

template <typename T>
Array<T>::Array(T arr[], int s) {
   ptr = new T[s];
   size = s;
   for(int i = 0; i < size; i++)
      ptr[i] = arr[i];
}

template <typename T>
void Array<T>::print() {
   for (int i = 0; i < size; i++)
      cout<<" "<<*(ptr + i);
   cout<<endl;
}

int main() {
   int arr[5] = {1, 2, 3, 4, 5};
   Array<int> a(arr, 5);
   a.print();
   return 0;
}
```

Non-type template argument

- ✓ A template can have multiple arguments. In the template specification for a generic class, it is also possible to specify non-type arguments.
- ✓ It is addition to the type argument T, another non-type argument may be either of following: integers, strings, function names, constant expression and built-in types.

```
template<class T, int size>
class Myfilebuf
{
    T filepos;
    static int array[size];
          public:
    Myfilebuf() { /* ... */ }
    ~Myfilebuf();
    advance(); // function defined elsewhere in program
};
```

Partial and Primary template specialization

- ✓ Allows customizing class and variable templates for a given category of template arguments.

```
template<class T1, class T2, int I>
class A {};          // primary template
```

```
        template<class T, int I>

        class A<T, T*, I> {};  // #1: partial specialization where T2 is a pointer to T1


        template<class T, class T2, int I>

class A<T*, T2, I> {}; // #2: partial specialization where T1 is a pointe
```

1 word Question Answer

Sr No.	Question	Answer
1.	_______are expanded at compiler time.	Templates

TOPIC: CONTAINERS

- ✓ Containers or container classes store objects and data. There are in total seven standard “first-class” container classes and three container adaptor classes and only seven header files that provide access to these containers or container adaptors.
- ✓ Sequence Containers: implement data structures which can be accessed in a sequential manner.
- ✓ vector
- ✓ list
- ✓ deque
- ✓ arrays
- ✓ forward_list(Introduced in C++11)
- ✓ Container Adaptors : provide a different interface for sequential containers.
- ✓ queue
- ✓ priority_queue

- ✓ stack
- ✓ Associative Containers : implement sorted data structures that can be quickly searched (O(log n) complexity).
- ✓ set
- ✓ multiset
- ✓ map
- ✓ multimap

1 word Question Answer

Sr No.	Question	Answer
1.	_________ classes store objects and data.	Containers
2.	_________ provide a different interface for sequential containers.	Container Adaptors
3.	_________ implement sorted data structures that can be quickly searched (O(log n) complexity).	Associative Containers

TOPIC: ITERATORS IN C++ STL

- ✓ Prerequisite : Introduction to Iterators
  Iterators are used to point at the memory addresses of STL containers. They are primarily used in sequence of numbers, characters etc. They reduce the complexity and execution time of program.

Operations of iterators :-

1. begin() :- This function is used to return the **beginning position** of the container.

2. end() :- This function is used to return the **after end position** of the container.

3. advance() :- This function is used to **increment the iterator position** till the specified number mentioned in its arguments.

4. next() :- This function **returns the new iterator** that the iterator would point after **advancing the positions** mentioned in its arguments.

5. prev() :- This function **returns the new iterator** that the iterator would point **after decrementing the positions** mentioned in its arguments.

6. inserter() :- This function is used to **insert the elements at any position** in the container. It accepts **2 arguments, the container and iterator to position where the elements have to be inserted**.

1 word Question Answer

Sr No.	Question	Answer
1.	______ function is used to return the **beginning position** of the container.	**begin()**
2.	_______ function is used to return the **after end position** of the container.	**end()**
3.	________ function is used to **increment the iterator position** till the specified number mentioned in its arguments	**advance()**
4.	_______ function **returns the new iterator** that the iterator would point after **advancing the positions** mentioned in its arguments.	**next()**
5.	_______ function **returns the new iterator** that the iterator would point **after decrementing the positions** mentioned in its arguments.	**prev()**
6.	_______ function is used to **insert the elements at any position** in the container.	**inserter()**

TOPIC: NAMESPACES IN C++

- ✓ Same situation can arise in your C++ applications. For example, you might be writing some code that has a function called xyz() and there is another library available which is also having same function xyz(). Now the compiler has no way of knowing which version of xyz() function you are referring to within your code.
- ✓ A namespace is designed to overcome this difficulty and is used as additional information to differentiate similar functions, classes, variables etc. with the same name available in different libraries.
- ✓ Using namespace, you can define the context in which names are defined. In essence, a namespace defines a scope.

TOPIC: DEFINING A NAMESPACE

A namespace definition begins with the keyword namespace followed by the namespace name as follows −

```
namespace namespace_name {
  // code declarations
}
```

- ✓ To call the namespace-enabled version of either function or variable, prepend (::) the namespace name as follows −
- ✓ name::code; // code could be variable or function.
- ✓ Let us see how namespace scope the entities including variable and functions −

```
#include <iostream>

using namespace std;


// first name space

namespace first_space {

  void func() {

    cout << "Inside first_space" << endl;

  }
}

// second name space
namespace second_space {
  void func() {
    cout << "Inside second_space" << endl;
  }
}

int main () {
  // Calls function from first name space.
  first_space::func();

  // Calls function from second name space.
  second_space::func();

  return 0;
}
```

1 word Question Answer

Sr No.	Question	Answer
1.	Using ________ you can define the context in which names are defined.	namespace

TOPIC: What is STL? Explain in brief.

- ✓ The Standard Template Library (STL) is a set of C++ template classes to provide common programming data structures and functions such as lists, stacks, arrays, etc.
- ✓ It is a library of container classes, algorithms, and iterators.
- ✓ It is a generalized library and so, its components are parameterized.
- ✓ A working knowledge of template classes is a prerequisite for working with STL.
- ✓ **STL has four components**

1. Algorithms
2. Containers
3. Functions
4. Iterators