**Oops through c++**

**Step material**

|  |  |
| --- | --- |
| |  | | --- | | **1. List out the characteristics of FOP.** 1. Large programs are divided into smaller programs known as functions. 2. Most of the functions share global data 3. Functions transform data from one form to another. 4. It employs top-down approach.    **2. List out the characteristics of OOP.** 1. Programs are divided into objects. 2. Data is hidden. 3. Objects communicate with each other, by sending messages and receiving responses. 4. It follows bottom-up approach.    **3. List down the basic concepts of OOP.** 1. Objects 2. Classes 3. Data abstraction and encapsulation 4. Inheritance 5. Polymorphism 6. Dynamic binding 7. Message passing    **4. Define an object.** Objects are the basic run time entities in an object oriented system. They may represent a person, a place or any item that a program has to handle.    **5. Define Object Oriented Programming.** OOP is a method of implementation in which programs are organized as co-operative collection of objects, each of which represents an instance of some class and whose classes are all members of a hierarchy of classes united through the property called Inheritance.    **6. Define a class.** A class is a collection of objects with similar attributes and operations. Eg. Class – AccountIt will create an object savings\_account belonging to the class Account.  **7. Define Encapsulation.** The wrapping up of data and functions into a single unit is known as encapsulation. The data is kept safe from external interference and misuse.    **8. Define Data hiding?** The isolation of data from direct access by the program is called as data hiding or information hiding.    **9. Define Abstraction.** It refers to the act of representing essential features without including the background details.    **10. Define ADT?** The classes which are using the concept of data abstraction is known as abstract data types (ADT).  **11. Define data member and member function?** The attributes which are holding the information is known as data members. The functions that operate on data member are sometimes called as member function.    **12. Define Inheritance?** Inheritance is the process by which objects of one class acquire the properties of objects of another class. The new derived class inherits the members of the base class and also adds its own.    **13. Define Polymorphism?** It allows a single name/operator to be associated with different operations depending on the type of data passed to it. An operation may exhibit different behaviors in different instances.    **14. Define dynamic binding?** Dynamic binding means that the code associated with the given procedure call is not known until the time of call at runtime.    **15. What is message passing?** It is the process of invoking an operation on an object. In response to a message, the corresponding function is executed in the object.    **16. List down the benefits of OOP?** 1. The principle of data hiding helps to build secure programs. 2. Through inheritance, redundant code is eliminated. 3. Software complexity can be easily managed.    **17. List down the applications of OOP.** 1. Real time system 2. Simulation and modeling 3. AI and expert system 4. Neural network programming. 5. CAD/CAM systems.    **18. What is Implicit Type Conversion?** When an expression consists of data items of different types, the compiler performs type conversions automatically. This is referred as Implicit Type Conversion.    **19. What are the differences between ‘break’ and ‘continue’ statement.**  **Break**  Break statement takes the control to the outside of loop It is used in loop and also in switch statement  **continue** Continue statement takes the control to the beginning of loop It can be used only in loop statement.  **20. Distinguish ‘while’ and ‘do – while’ statements.**  **While**  This is the top tested loop Loop is not executed if the condition is false  **do while** This is the bottom tested loop Loop is executed at least once even though the condition is false.    **21. Give the syntax of Array Initialization with an example.** Data-type array-name[size] = {list of values separated by comma};  (eg.) int mark[5] = {96,45,66,74,82};    **22. Give four examples for String Manipulation functions.** strlen( ) – finds the length of string strcpy( ) – copies the contents of one string to another strcat( ) – concatenates two strings into one single string strupr( ) – converts a lower case string to upper case  **23. List the different types of parameter passing techniques.** (i) Pass by value (ii) Pass by Address (iii) Pass by Reference    **24. What is Dynamic memory allocation?** Allocation of memory space for any data structure during the course of the program execution is called as Dynamic memory allocation.    **25. How memory management is performed dynamically in C++?** Two operators are available for dynamic memory management. (i) new – for dynamic memory allocation (ii) delete – for dynamic memory deallocate  **Basics of C++ Language**  **1. Define class?** A class is a way to bind data and its associated functions together. It allows the data to be hidden if necessary.    **2. What are the parts of class specification?** The class specification has two parts 1. Class declaration – To describe the type and scope of its members 2. Class function definition – To describe how the class functions are implemented.    **3. Write the syntax of class declaration** class classname {  Private: Variable declaration; Function declaration; Public: Variable declaration; Function declaration; };    **4. Where will you define a member function?**  Member functions can be defined in two places. 1. Outside the class definition. 2. Inside the class definition.    **5. Give the syntax for member function definition outside the class.** Return type class name:: function name (argument name declaration) { function body }    **6. List the characteristics of member function.** 1. Member function can access the private data of class. 2. A member function can call another member function directly without using the dot operator.  **7. Define nesting of member function.** A member function can be called by using its name inside another member function of the same class. This is known as nesting of member functions.    **8. List the properties of static members.** A data member of a class can be qualified as static properties. 1. It is always initialized to zero when the first object of its class is created. 2. It is visible only within the class.    **9. Write the properties of static member function.** 1. A static function can have access to only other static members declared in the same class. 2. A static member function can be called using the class name (Instead of objects) as follows: class name:: function name;    **10. Define constructor?** A constructor is a special member function whose task is to initialize the object of its class. It is called constructor because it constructs the value of data members of the class.    **11. What are the characteristics of constructor?** 1. Constructors should be declared in public section. 2. They are involved automatically when the objects are created. 3. They do not have return types. 4. They can not be inherited.    **12. Define parameterized constructor.** Arguments can be passed to the constructor function when the objects are created. The constructors that can take arguments are called as parameterized constructor.    **13. What is an implicit constructor?** C++ compiler has an implicit constructor which creates objects even though it was not defined in the class.    **14. What is the use of copy constructor?** Copy constructor is used to declare and initialize an object from another object. E.g. Class name object2 (object1); Will define the object2 and at the same time initialize it the values of object1.  **15. What do you mean by dynamic construction?** Allocation of memory to objects at the time of their contraction Is known as dynamic contraction of objects.    **16. What is the use of destructor?** It is used to destroy the objects that have been created by a constructor. It releases the memory space for future use.    **17. What are the characteristics of destructor?** 1. A destructor is a member function whose name is the same as the class name but it is preceded by a tilde. 2. It neither takes any argument nor returns any value. 3. It will be invoked implicitly by the compiler to cleanup the storage.    **18. Define operator overloading?** The process of making an operator to exhibit different behaviors in different instances is known as operator overloading.    **19. Define function overloading?** Performing different types of task using single function name is referred as function overloading.    **20. Define Virtual function.** When the form of a member function is changed at run time, that member function is referred to as virtual function.  **Unit-I**  **1.What are the characteristics of procedure oriented programming language?**   1. Large programs are divided into smaller programs known as functions 2. Most of the functions share global data 3. Data move openly around the system from function to function 4. Functions transform data from one form to another 5. Uses top-down approach in program design. 6. Concentration is on doing things( algorithms)     **2.What are the features of object-oriented programming languages?**   1. Programs are divided into objects 2. Data structures designed such that they characterize the objects. 3. Functions that operate on the data of an object are tied together in the data structure 4. Data is hidden and cannot be accessed by external functions 5. Objects may communicate with each other through functions 6. New data and functions can be easily added whenever necessary 7. IT follows bottom-up approach in program design 8. Concentration is on data rather than procedure   **3.Define Object Oriented Programming**  It is an approach that provides a way of modularizing programs by creating partitioned memory area for both data and functions that can be used as templates for creating copies of such modules on demand.   4. **What is meant by an object?**   * Objects are the basic run-time entities in an object-oriented system. * They may represent a person, a place, , a table of data or any item that the program must handle. * They may also represent user defined data such as vectors, time and lists. * When a program is executed, the objects interact by sending message to one another. * Each object contains data and code to manipulate the data. * Objects can interact without knowing having to know details of each other’s data or code.   **5. What is meant by Classes?**   1. A class is a collection of objects of same type. 2. Once the class has been defined , we can create any number of objects belonging to that class. 3. Each object is associated with the data of type class with which they are created.   **6.What is meant by Encapsulation?**   * The wrapping up of data and functions into a single unit ( called class) is known as encapsulation * Data encapsulation is the most striking feature of a class. * The data is not accessible to the outside world and only those functions which are wrapped in the class can access it. * These functions provide the interface between the object’s data and the program * This insulation of the data from direct access by the program is called data hiding.    7. **What is meant by Abstraction**   * => It refers to the act of representing essential features without including the background details or explanation * Classes use the concept of abstraction and are defined as a list of abstraction and are defined as a list of abstract attributes and functions to operate on these attributes. * They encapsulate all the essential properties of the objects that are to be created   **8. Define Inheritance.**   * Inheritance is the process by which objects of one class acquire the properties of objects of another class. * It provides the idea of reusability. * This is possible by deriving a new class from the existing class . The new class will have the combined features of both the classes.   **9.Define polymorphism**   * It means the ability to take more than one form. * For ex, consider the operation of addition. * If the operands are strings, then the operation would produce a third string by concatenation. * If the operands are numbers, it will generate a sum.   **10.Define dynamic binding**   * It refers to the linking of a procedure call to the code to be executed in response to the call. * Dynamic binging means that the code associated with a given procedure call is not known until the time of the call at run time. * It is associated with polymorphism and inheritance   **11. What r the benefits of OOP**   * Through inheritance, we can eliminate redundant code and extend the use of existing classes. * It saves the program development time and higher productivity. * The use 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 an objects to co-exist without any interference. * It is possible to map objects in the program domain to those objects in the program. * It is easy to partition the work in a project based on objects. * It can be upgraded from small to large systems. * It makes the interface descriptions with external systems much simpler by using message passing techniques. * Software complexity can be easily managed.   **12. What are the 2 types of OOP?**   1. Object-Based Programming Languages 2. Object-Oriented Programming Languages   Object- Based Programming Languages   * It is the style of programming the primarily supports encapsulation and object identity. * Major features are:  1. Data encapsulation 2. Data hiding and access mechanisms 3. Automatic initialization and clear-up of objects 4. Operator Overloading  * It doesn’t support inheritance and dynamic binding * Ex: Ada   Object-Oriented Programming Languages   * It combines Object- based programming features along with 2 additional features, inheritance and dynamic binding * Ex: C++, Smalltalk     **13. What are the applications of OOP**   * Real time systems * Simulation and modeling * Object oriented data bases * Hypertext, hypermedia and expertext * AI and expert systems * Neural networks and parallel programming * Decision support and office automation systems * CIM\CAM\CAD systems     **14. What is C++?**   * C++ is an object-oriented programming language * Also known as C with classes , combination of C and simula 67 * Invented by Bjarne Stroustrup at AT &T Bell Lab in New Jersey, USA. * It is an extension of C with a major addition of the class construct feature of simula 67. * The 3 most important features That C++ adds on to C are classes, function overloading and operator overloading. * It allows the programmer to build large programs with clarity, extensibility and ease of maintenance, incorporating the sprit and efficiency of C     **15. What are the applications of C++?**   * It is a versatile language for handling very large programs * It is suitable for virtually any programming task including development of editors, compilers, databases, communication systems and any complex real life application systems * It allows us to create hierarchy-related objects, so we can build special object-oriented libraries which can be used later by many programmers * While C++ is able to map the real-world problem properly, the C part of C++ gives the language the ability to get close to the machine – level details * C++ programs are easily maintainable and expandable     **16. Why we need the preprocessor directive #include?**   * This directive causes the preprocessor to add the content of the iostream file to the program. It contains declarations for the identifier cout and the operator <<. * It should be included at the beginning of all programs that use i/o statements.     **17. How does a main() function in C++ differ from main() in C?**  In C++, main() returns an integer type value to the operating system. Therefore , every main() in C++ should end with a return (0) statement. Otherwise a warning or an error occurs. Since main() returns an integer type value, return type for main() is explicitly specified as int.  In C, it does not specify any return value for the main() function. So there is no need for explicitly use the return statement.    **18. Describe major parts of a C++ program.**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Include File | | ——————————– | | Class Declaration | | ——————————-|  | Member Function Definition | | ——————————-| | Main Function Program | |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|  The class declarations are placed in a header file and the definitions of member functions go into another file. The main program that uses the class is placed in a third file which “ includes” the previous two files as well as any other files required.    **19. Describe I/O operator.**  Input Operator  The statement               **cin>> num;**  is an input statement and causes the program to wait for the user to type in a number. The operator >> is known as extraction or get from operator. It takes the value from the keyboard and assigns it to the variable on its right.  Output Operator  The statement               **Cout<< “ the numbers”;**  uses the cout identifier that represents the standard output stream ( screen) in C++. The operator <    **20. Describe tokens**  The smallest individual units in a program are known as tokens. C++ has the following tokens:   1. Keywords 2. Identifiers 3. Constants 4. Strings 5. Operators.     **21. Classifications of Data Types.**  1. User defined Type  a. Structure  b. Union  c. Class  d. Enumeration  2. Built in type  a. integral type  i. int  ii. char  b. void  c. floating type  i. float  ii. double  d. Derived type  i. array  ii. function  iii. pointer  iv. reference    **22.Describe with example the uses of enumeration data type**  (i) It provides a way for attaching names to numbers.  (ii) The enum keyword gives the list of words by assigning them values 0,1,2 and so on. The Syntax is  **enum identifier { enumerated constants };**  Ex: enum shape{ circle, square, triangle};  enum color { red, blue, green };  colur background; /\* background is of type color \*/  (iii) Here each enumerated data type retains its own separate type. Ex:  color background = blue;  color background = ( color) 7;  (iv) By default, the enumerators are assigned integer values starting with 0 for the first enumerator. But this can be over-ride. For ex,  enum color { red, blue=4, green =8};  [ Write one example c++ program ]    **23. Explain Symbolic Constants**  There are two ways of creating symbolic constants  (i) Using the qualifier Constants  => Any value declared as const can’t be changed.  => ex: const int size =10;  char name[ size];  (ii) Defining a set of integer constants using enum keyword  For ex, enum { X, Y, Z];  This defines X, Y,Z as integer constants with values 0,1,2 respectively.  This can be also assigned explicitly. For ex,  enum { X=100, Y=200, Z=300 };    **24. What do u mean by dynamic initialization of a variables?**  Initialization of variable at run time is known as dynamic initialization of a variables. That is the variables can be initialized at run time using expression at the place of declaration.   Ex:  ………  ………  int n = strlen (string);  ……..   for( int i=0;i<5;i++)  {  ……  ……  }  By using this, we can create exactly the type of object needed, using the information that is known only at the run time.  **25. What are the operators available in C++?**   * Arithmetic operator * Relational Operator * Assignment Operator * Logical Operator * Increment/ decrement Operator * Conditional operator * Bitwise operator * Special operator * Scope resolution operator * Pointer to member declaration * Pointer to member operator * Memory release operator * Line feed operator * Filed width operator     **26.Explain about Type Casting Operator**  This is used for conversion of variables or expression explicitly  Syntax:  (type – name) expression  or  Type –name ( expression)  Ex: avg= sum/ float(i);  Alternatively, we can use typedef to create an identifier of the required type and use it in the functional notation  typedef int \* int\_pt;  p= int\_p;  **27.What are the types of expression?**   * Constant expression * Integral expression * Float expression * Pointer Expression * Relational Expression * Logical expression * Bitwise expression     **28. What is meant by operator overloading?**  Overloading means assigning different meaning to an operation, depending on the context. That is it is used to assign multiple meanings to operators.  Ex:  The operator \* when applied to a pointer variable, gives the value pointed to by the pointer. But it is used for multiplying two numbers.    **29. What are the types of control strucuture**   * sequence * selection * iteration     **30.What are the advantages of new operator**   * It automatically computes the size of the data object. So there is no need to use sizeof operator * It automatically returns the correct pointer type, so that here is no need to use a type cast. * It is possible to initialize the object while creating the memory space * Like any other operator, operator new and delete can be overloaded.     **Unit-II**   1. **What is a class ?**   It is an extension to the structure data type. A class can have both variables and functions as members     1. **What is the difference between structure and a class ?**   The only difference between a structure and a class in C++ is that , by default , the members of a class are private, while , by default the members of a structure are public.     1. **What is the specification for a class ?**    * Class declaration    * Class function definitions   4.       **What are data members and member functions?**  The variables declared inside the class are known as data members and the functions are known as member functions. The data members are usually private and member functions as public.   1. **Give a simple class example.**   class item  {  int number;  float cost;  public:  void getdata(int a, float b);  void putdata(void);  };  Here class name is item  Data : number, cost  Functions: getdata(), putdata()     1. **What are objects ?**   The class variables are called objects. With objects we can access the public members using dot operator     1. **How is amember function of a class is defined?**   It can be defined either inside or outside the class     1. **What are the characteristics of member functions ?**    * Several different classes can use the same function name. the membership label will resolve their scope    * Member functions can access the private data of the class. Anon member function cannot do so    * A member function can call another function directly ,without using dot operator   9.       **When a function is defined inside a class ?**   * + it is treated as a inline function   + only small functions are defined inside the class definition   10.   **What is nesting of member functions ?**  Amember function can be called by using its name inside another member function of the same class, is known as member function     1. **How the space is allocated for the objects?**   The memory space is allocated when they are declared . space for the member variables is allocated separately for each object, but no separate space is allocated for the member functions   1. **When do we declare a member of a class static ?**   When it is used to maintain values common to the entire class. The static member variables defined outside the class     1. **What is a friend function?**   The functions that are declared with the keyword friend are known as friend functions. A function can be declared as a friend in any number of classes, it has full access rights to the private members of the class.     1. **What are the special characteristics of friend function ?**    * Can be invoked like a normal function, with the help of the object    * It has the objects as arguments    * It is not in the scope of the class to which is has been declared has friend   15.   **What is const member function ?**  If a member function does not alter any data in the class, then we declare it as const member function. The keyword const is appended to the function prototype.     1. **What is a constructor ?**   It is a special member function whose task is to initialize the objects of its class. It is special because its name is the same name as the class name.     1. **How do we invoke constructor function?**   It is invoked whenever an object of an associated class is created. It is called constructor because it constructs the values of data members of the class.  18.**List some special properties of constructor functions**.   * + They should be declared in the public section   + They are invoked automatically when the objects are created   + They do not have return types, therefore they cannot return values   + They cannot be inherited   + They can have default arguments   + Cannot refer to addresses   19.   **what is parameterized constructor ?**  It is nothing but passing arguments to the constructor function when the objects are created. The constructor can take arguments are called parameterized constructor.   1. **What is copy constructor ?**   The constructor that creates a new class object from an existing object of the same class.   1. **What is dynamic initialization of objects ?**   The initial value of an object provided at the run time. The advantage is that we can provide various initialization formats ,using overloaded constructors.   1. **What is dynamic constructor ?**   Allocation of memory to objects at the time of their construction is known as dynamic construction of objects. The memory is allocated with the help of new operator.   1. **What is a destructor?**   It is used to destroy the objects that have been created by a constructor, when they no longer required.    **Unit-III**  **1.What is meant by reusability**  Reusability is a feature which is supported in object-oriented programming. This allows the reuse of existing classes without redefinition.  **2.Define Inheritance**   * In object-oriented programming, **inheritance** is a way to form new classes (instances of which are called objects) using classes that have already been defined. * The former, known as **derived classes**, take over (or **inherit**) attributes and behavior of the latter, which are referred to as **base classes**. * It is intended to help reuse of existing code with little or no modification. * **Inheritance** is also called **generalization**, because the **is-a** relationships capture a hierarchal relationship between classes of objects     **3.What are the applications of Inheritance**  There are many different aspects to inheritance. Different uses focus on different properties, such as the external behavior of objects, internal structure of the object, structure of the inheritance hierarchy, or software engineering properties of inheritance. (i)Specialization One common reason to use inheritance is to create specializations of existing classes or objects. This is often called *subtyping* when applied to classes. In specialization, the new class or object has data or behavior aspects that are not part of the inherited class.  Another form of specialization occurs when an inherited class specifies that it has a particular behavior but does not actually implement the behavior. Each non-abstract, concrete class which inherits from that abstract class must provide an implementation of that behavior. This providing of actual behavior by a subclass is sometimes known as *implementation* or *reification*. (ii)Overriding Many object-oriented programming languages permit a class or object to replace the implementation of an aspect—typically a behavior—that it has inherited. This process is usually called *overriding*. (iii)Extension Another reason to use inheritance is to provide additional data or behavior features. This practice is sometimes called *extension* or *subclassing*.  Extension is often used when incorporating the new features into the inherited class is either not possible or not appropriate. (iv)Code re-use One of the earliest motivations for using inheritance was to allow a new class to re-use code which already existed in another class. This practice is usually called *implementation inheritance*. 4.What are the Constraints of inheritance-based design?  * **Singleness**: using single inheritance, a subclass can inherit from only one superclass. Continuing the example given above, Person can be either a Student or an Employee, but not both. Using multiple inheritance partially solves this problem, as a StudentEmployee class can be defined that inherits from both Student and Employee. However, it can still inherit from each superclass only once; this scheme does not support cases in which a student has two jobs or attends two institutions. * **Staticness**: the inheritance hierarchy of an object is fixed at instantiation when the object’s type is selected and does not change with time. For example, the inheritance graph does not allow a Student object to become a Employee object while retaining the state of its Person superclass. * **Visibility**: whenever client code has access to an object, it generally has access to all the object’s superclass data. Even if the superclass is not a public one, the client can still cast the object to its superclass type. For example, there is no way to give a function a pointer to a Student’s grade point average and transcript without also giving that function access to all of the personal data stored in the student’s Person superclass.   **5. What are the types of inheritance?**  1. Single inheritance: A derived class with only one base class is called single inheritance  2. Multiple inheritance: A class can inherit properties from more than one class which is known as multiple inheritance  3. Multilevel inheritance: A class can be derived from another derived class which is known as multilevel inheritance.  4. Hierarchical inheritance: When the properties of one class are inherited by more than one class, it is called hierarchical inheritance.  **6. How to define derived classes?**  A derived class can be defined by specifying its relational ship with the base class in addition to its own details.  The syntax is:  class derived-class-name : visibility-mode base-class-name  {  ……  …….};  Here the visibility mode is optional and if present, may be either private or public. The default mode is private.  When a base class is privately inherited by a derived class, public members of the base class become private members of the derived class and therefore the public members of the base class can only be accessed by the member functions of the derived class.  When the base class is publicly inherited, public members of the base class become public members of the derived class and therefore they are accessible to the objects of the derived class. 7. What is inherited from the base class? In principle, a derived class inherits every member of a base class except:   * its constructor and its destructor * its operator=() members * its friends   Although the constructors and destructors of the base class are not inherited themselves, its default constructor (i.e., its constructor with no parameters) and its destructor are always called when a new object of a derived class is created or destroyed.  **8.Define virtual base class**   * A base class that has been qualified as virtual in the inheritance definition. * In multiple inheritance, a derived class can inherit the members of a base class via two or more inheritance paths. * If the base class is not virtual, the derived class will inherit more than one copy of the members of the base class.     **9. Define virtual function**   * It is a function qualified by the virtual keyword. When a virtual function is called via a pointer, the class of the object pointed to determines which function definition will be used. * Virtual functions implement polymorphism, whereby objects belonging to different classes can respond to the same message in different ways.   **10. What is meant by pure virtual function**   * A virtual function that is declared in a base class but not defined there. The responsibility for defining the function falls on the derived classes, each of which generally provides a different definitions. * It is illegal to create instances of a class that declares a pure virtual function. So such a class is necessarily an abstract base class.     **11. What is meant by subclass and superclass**  Subclass: a class which has link to a more general class  Superclass: a class which has one or more members which are classes themselves.  **12. Define abstract class**  Abstract class is one that is not used to create objects. An abstract class is designed only to act as a base class. It is a design concept in program development and provides a base upon which other classes may be built.  **13.What are the forms of inheritance**  <!--[if mso &amp; !supportInlineShapes &amp; supportFields]&gt; SHAPE  \\* MERGEFORMAT <![endif]–>The virtual function must be members of some function.  •  They can’t be static members  •  They are accessed by using object pointers.  •  A virtual function can be a friend of another class  •  A virtual function in a base class must be defined, even though it may not be used.  •  The prototypes of the base class version of a virtual function and all the derived class versions must be identical. If two functions with the same name have different prototypes, C++ considers them as overloaded functions, and the virtual function mechanism is ignored.  •  We cannot have virtual constructors, but we can have virtual destructors.  •  While a base pointer can point to any type of the derived object, the reverse is not true.  •  When a base pointer to a derived class, incrementing or decrementing it will not make it to point to the next object of the derived class. It is incremented or decremented only relative to its base type. Therefore we should not use this method to move the pointer to the next object.  •  If a virtual function is defined in the base class, it need not be necessarily redefined in the derived class. In such cases, calls will invoke the base function.    **14. What is meant by this pointer?**   * C++ uses the unique keyword called **this** to represent an object that invokes a member function. * **this** is a pointer that points to the object for which this function was called. * This unique pointer is automatically passed to a member function when it is called. The pointer this acts as an implicit argument to all the member functions. * One important application of the **this** pointer this is to return the object it points to.     **15. What is meant by pointer and null pointer?**  **Pointer** = pointer is a data type that holds the address of a location in memory.  **Null Pointer** = is a pointer that does not point to any data object. In C++, the null pointer can be represented by the constant 0.  **Unit-V**  **1. What are C++ Streams?**  The C++ language offers a mechanism, which permits the creation of an extensible and consistent input-output system in the form of streams library. It is a collection of classes and objects which can be used to build a powerful system or it can be modified and extended to handle user defined data types  **2. List the predefined console streams.**a) cin – standard input b)cout – standard output c) cerr – standard error output d)clog – fully buffered version of cerr  **3. Give the usage of ios class.**  The ios class provides operations common to both input and output. It contains a pointer to a buffer object. It has constants and member functions that are useful in handling formatted I/O operations. Following are the derived classes of ios class,  a) istream – input stream b)ostream – output stream c)iostream – input-output stream  **4. What are the types of formatted console i/o operations?** a) ios stream class member functions and class b) standard manipulators c) user defined manipulators  **5. Give the flag value and bit field for (a) Left justified output and (b) Decimal** **conversion.**  Flag value Bit field (a) Left justified output ios::left ios::adjustfield (b) Decimal conversion ios::dec ios::basefield  **6. What are the types of manipulators? Give example.**  Two types of manipulators are available in C++.  a) **Parameterized manipulators**  Eg: setw(int width) – sets the field width setprecision(int prec) – sets the floating point precision  b) **Non-parameterized manipulator**  Eg: dec – sets the conversion base to 10  Endl – outputs a new line and flushes stream  **7. What is a custom manipulator? Give its syntax.** Designing of customized manipulators to control the appearance of the output is referred as custom manipulator.  Syntax:  ostream & manipulator(ostream & output, arguments\_if\_any)  {//manipulator code  return output;  }  **8. Write a note on File.**  A file is a collection of related information normally representing programs, both source and object forms and data. Data may be numeric, alphabetic or alphanumeric. A file can also be defined as a sequence of bits, bytes, lines or records whose meaning is defined by the programmer. Operations such as create, open, read, write and close are performed on files  **9. Define ifstream & fstream.**  ifstream:  It is used for handling input files. It contains open() with default input mode and inherits get(), getline(), read(), seekg(), tellg() functions from istream.  fstream:  Used for handling files on which both i/o operations can be performed. It supports simultaneous i/o operations. It contains open() with default input mode and inherits all the functions from istream and ostream classes through iostream.  **10. Give the prototypes of file stream class constructors.**  a) ifstream class constructor ifstream(const char \*path, int mode=ios::in, int prot=filebuf::openprot);  b) ofstream class constructor  ofstream(const char \*path, int mode=ios::out int prot=filebuf::openprot);  c) fstream class constructor  fstream(const char \*path, int mode=ios::in/ios::out, int prot=filebuf::openprot);  **11. List any four file modes and their purpose** a) ios::in – open for reading b) ios::ate – seek to the end of file at opening time c) ios::nocreate – open fails if file does not exist d) ios::binary – opens a binary file  **12. List the file pointer control functions.** a) seekg() – moves get file pointer to a specific location b) seekp() – moves put file pointer to a specific location c) tellg() – returns the current position of the get pointer d) tellp() – returns the current position of the put pointer  **13. What are the types of file accessing?**  **a) Sequential access** This type of file is to be accessed sequentially that is to access a particular data all the preceding data items have to be read and discarded.  **b) Random access** This type of file allows access to the specific data directly with out accessing its preceding data items  **14. Give any two error handling functions and their purpose**  a) eof() – TRUE(nonzero) if eof encountered while reading; otherwise FALSE(zero) b) rdstate() – returns the status state data member of the class ios  **15. Define fault avoidance and fault tolerance**  **Fault avoidance:**  It deals with the prevention of fault occurrence by construction. It emphasizes on techniques to be applied during system development that ensures the system satisfies all reliability criteria  **Fault tolerance:**  This deals with the method of providing services complying with the specification in spite of false occurring by redundancy.  **16. What is exception handling? How it is classified?**  The error handling mechanism of C++ is referred to as exception handling. Exception refers to some unexpected condition in a program. It is classified as synchronous and asynchronous exception.  **Synchronous exception :**  The exceptions, which occur during the program execution, due to some fault in input data, within the program, is known as Synchronous exception.  **Asynchronous exception :**  The exceptions caused by events or a fault unrelated to the program and beyond the control of the program is known as asynchronous exception.  **17. Define the exception handling constructs.**  **a) try**  This keyword defines a boundary within which an exception can occur. A block of code in which an exception may occur must be prefixed by this keyword.  **b) throw**  Throw is used to raise an exception when an error is generated in the computation. It initializes a temporary object to be used in throw.  **c) catch** This keyword represents exception handler. It must be compulsorily used immediately after the statements marked by try keyword. It can also occur immediately after catch keyword. Each handler will only evaluate an exception that matches or can be converted to the type specified in the argument list  **18. What are the tasks to be performed by error handling code?** a) Hit the exception – detect the problem causing exception b) Throw the exception – inform that an error has occurred c) Catch the exception – receive the error information d) Handle the exceptions – take corrective actions  **19. List the functions for handling uncaught exceptions.** a) **terminate()** – it is invoked when an exception is raised and the handler is not found. b) **set\_terminate()** – allows the user to install a function that defines the program’s actions to be taken to terminate the program when a handler for the exception cannot be found c) **unexpected()** – this function is called when a function throws an exception not listed in its exception specification d) **set\_unexpected()** – it allows the user to install a function that defines the program’s actions to be taken when a function throws an exception not listed in its exception specification  **20. How fault tolerant s/w design techniques are classified?** a) **N-version programming** – in this technique, N-programmers develop N algorithms for the same problem with out interacting with each other. All these algorithms are executed simultaneously on a multiprocessor system and the majority solution is taken as the correct answer.  b) **Recovery block** – this structure represents the dynamic redundancy approach to s/w fault tolerance. It consists of (i) Primary routine– executes critical s/w function, (ii) acceptance test – tests the output of primary routine after each execution and (iii) alternate routine – performs the same function as primary routine but is invoked by an acceptance test after reduction of a fault. | |