Basic Definitions in C++

1) **C++:** C++ is an object oriented programming language that allows programmers to build large and complex applications in a useful and efficient way.
2) **Header files:** Header files are special pre-defined files in C++ for some specified purpose. It contains the definitions of functions as well as object that are used in a program.
3) **Character Set:** It is a set of valid characters that a language can recognize.
4) **Tokens:** It is the smallest element of a C++ program that is meaningful to the compiler.
5) **Identifiers:** An identifier is a sequence of letters and digits. The term identifiers are used to describe the name of constants, variables, functions etc.
6) **Keywords:** Keywords (also known as reserved or predefined words) have special meaning to the compiler and are always written or typed in lower case. These are reserved by the language for special purpose and cannot be redefined to use as identifiers. Example: auto, break, if, for, etc.
7) **Literals:** Literals or constants are those tokens whose value does not change during the program execution. The various literals are integer constants, character constants, floating constants and string constants.
8) **Operators:** Operator is the term (symbol) used to describe the action to be taken between one or more data expressions. An identifiers or a constants used with an operator is called *operand.*
9) **Arithmetic Operators:** These operators are used to perform mathematical calculations.
10) **Assignment Operators:** An assignment operator is used to assign the value to a variable.
11) **Increment/Decrement Operator:** Increment (++) and decrement (--) operators are called *unary operators* because they operate on single variable only. The increment operator (++) increments the value by 1 & the decrement operator (--) decrements the value by 1.
12) **Relational Operator:** The relational operators are used to test or compare the value of two operands.
13) **Logical Operator:** Logical operators are used to combine two or more relational operations or conditions. These are also called *Boolean operators* because the test between values are reduced to either true or false, with zero being false and one being true.
14) **Conditional Operator:** It is also called ternary operator. The conditional operator is condensed form of an if-then-else statement.

15) **Expressions:** An expression may consist of one or more operands and zero or more operators to produce a value.
16) **C++ Comments:** A comment is a text that the compiler ignores, but that is useful for programmers. Comments are normally used to annotate code for future reference. The compiler treats them as white space.
17) **Data Types:** Data types refer to the type of data used in the program. C++ supports a variety of data types and programmer can select the appropriate data type as per the need of the application.
18) **Built in Data Types:** These data types are known as basic or fundamental data types. These are the data types that are not composed of any other data type. These are already defined in compiler program. Example: char, int, float, double, void, etc.
19) **Derived Data Types:** These are the data types that are composed of fundamental data types. Example: array, function, pointer, reference, class, structure, union and enumeration.
20) **User Defined Data Types:** These are the data types that enables a programmer to invent his/her own data types and define it. Example: class, structure, union, etc.
21) **Variables:** Variables are identifiers, which can hold value. And the value of variable can be manipulated during program execution, i.e. in a program; we need to use values, which must be stored in memory cell.
22) **Errors:** Errors may be made during program creation even by the programmers. Such types of errors are detected by the compiler.
23) **Syntax Error:** This means that the compiler could not compile a line of code. Such as identifiers not declare, statement not terminated by semicolon etc.
24) **Runtime Error:** These errors may be detected at runtime due to division by zero, stack overflow etc.
25) **Logical Error:** This error would not be displayed on the screen. However, it will lead to display wrong results. Example: infinite loop.
26) **Linking Error:** These errors are occurring during the linking process.
27) **Statements:** A computer program is a set of instructions known as statements.
28) **Sequence Statements:** Sequence statements are being executed sequentially. Sequence structure is also known as straight line path.
29) **Selection or Branching Statements:** When programmers are required to execute a particular set of statements depending upon a particular test condition, a branching statement is required.
30) **Looping or Iteration Statements:** Sometimes, there is a need to repeat a group of statements a number of times. Iteration statements allow a set of instructions to

be performed repeatedly until a certain condition is fulfilled. Iteration continues till the loop's test condition becomes false.

31) **Jump Statements:** The jump Statements unconditionally transfers program control within a function.

32) **Functions:** A function is a self-contained block of statements, which are kept together to perform a specific task in related manner. A function has a name and it has a property i.e. reusable.

33) **Library Functions:** Library functions are the predefined functions in programming system. These are also known as *in-built functions.*

34) **User Defined Functions:** C++ provides facility for programmers to define their own functions according to their requirements. The function defined by programmers is known as user defined functions.

35) **Arrays:** An array is a collection of elements of the same type of variables that are referred by a common name, which are stored in memory sequentially.

36) **One-Dimensional Array:** A one dimensional array (or single dimensional array) is a type of linear array. Accessing its elements involves a single subscript, which can either represent a row or a column index.

37) **Multi-Dimensional Array:** The single dimensional array is known as array of vectors or list. Array of arrays are called multi-dimensional array. The simplest form of multi-dimensional array is the 2D (two-dimensional) array.

38) **Structure:** A structure is a group of data elements grouped together under one name. These data elements, known as members, can have different types and different lengths.

39) **Classes:** A class is a group of objects that share common properties and behaviors. Classes are user defined data types and behave like the built-in types of a programming language. Objects are variables of the type class.

40) **Data Abstraction:** Abstraction refers to the act of representing essential features of real word concept without including the background details.

41) **Data Encapsulation:** The wrapping up of data (member variables) and functions into a single unit is known as data encapsulation. It is the most fundamental concept of OOPs. The data is not accessible to the outside world and only those functions which are wrapped in that class can access it. These functions are referred to as member functions and also provide the interface between the object's data and the program.

42) **Data Hiding:** When member data and member function are bind into a single unit then the data is not accessible to the outside world and only those functions, which are wrapped in that class, can access it.

43) **Genericity:** The software components of a program have more than one version depending on the data types of arguments. This feature allows declaration of

variables without specifying exact data type. The compiler identifies the data type at run time. The programmer can create a function that can be used for any type of data. The template feature in C++ allows generic programming.

44) **Inheritance:** Inheritance is the process by which objects of one class acquire the properties of objects of another class. It supports the concept of hierarchical classification. The principle behind inheritance is that each derived class shares common characteristics with the class from which it is derived.

45) **Abstract Class:** Abstract class is a class that defines an interface, but does not necessarily provide implementation for all its member functions. No object of an abstract class exists, i.e. it cannot be instantiated.

46) **Concrete Class:** A concrete class is a derived class of an abstract class that implements all the missing functionality. This class can be instantiated, i.e. its object can be created.

47) **Polymorphism:** Polymorphism means processing of data or messages in more than one form. It is the phenomenon, where different set of actions for the same message. It is implemented using function overloading and operator overloading.

48) **Operator Overloading:** The process of making an operator to exhibit different behavior in different instances is called operator overloading.

49) **Function Overloading:** Function overloading refers to the use of one function name to perform different tasks. C++ allows us to use the same name for different function, as long as they have different parameter type lists, the compiler will regard them as different functions.

50) **Data Members:** They have the syntax of variable definitions and specify the representation (properties) of class objects.

51) **Member Functions:** They have the syntax of function prototypes and specify the class operations. The member function mostly manipulates the internal data or data member of a class.

52) **Public Members:** They are accessible by all class members.

53) **Protected Members:** They are only accessible by the class members and the members of a derived class.

54) **Private Members:** They are only accessible by the class members within class.

55) **Objects:** Objects are the variables of a user defined data type called class. In other words, the class acts as a data type and the object act as its variable. When we define a class, it does not define or create objects of that class; rather it only specifies what type of information the object of this class type will be contained. Once a class has been defined, its object can be defined like any other variable.

56) **Pass by Value:** In this method, a copy of the object is passed to function. This means that any change made to the object inside do not affect the object used to call the function.

57) **Pass by Reference:** In this method, an address of an object is passed to function. This means that any changes made to the object inside the function will reflect on the actual object.

58) **Static Data Member:** Static data members are the data members that are common to all the objects of a class. It means whenever a static data member is declared, only one copy of that member is created and is shared by all objects. They exist only once in all objects of a class. By default, static data members are initialized to zero when first object is created. The static data member should be created and initialized before the main function control block begins.

59) **Static Member Function:** We can also declare member function as static by prefixing *static* before the function name. The properties of static member functions are same as the static data member. A static function can access and manipulate only static data members of the class. A static member function is independent of any object.

60) **Constructor:** A Constructor is a special member function of a class that is automatically executed or called whenever an object of that class is created. It will have exact same name as the class and it does not have any return type. The Primary job of a constructor is to set initial values for certain member variables.

61) **Default Constructor:** A constructor having no arguments is called a default constructor.

62) **Parameterized Constructor:** The constructor with arguments is known as parameterized constructor. Parameterized constructors are very helpful in a situation, where the programmer want to initialize various data elements of different objects with different values when they are created.

63) **Copy Constructor:** The initialization of an object when done by another object is called copy constructor. The copy constructor is defined in the class as a parameterized constructor receiving an object as argument passed-by-reference.

64) **Destructor:** A destructor is a member function that is used to destroy the object that has been created by a constructor. Destructor will have same name as the class and tilde sign (~) precede their name.

65) **Single Inheritance:** When a derived class inherits only from one base class, it is known as single inheritance.

66) **Multiple Inheritances:** When a derived class inherits from multiple base classes, it is known as multiple inheritances.

67) **Multilevel Inheritance:** In multilevel Inheritance, a derived class can be used as a base class for another derived class, creating a multilevel class hierarchy. In this case, the original base class is said to be an *indirect base class of the second derived class.*

68) **Hierarchical Inheritance:** When many derived classes inherit from a single base class, it is known as hierarchical inheritance.

69) **Hybrid Inheritance:** Hybrid Inheritance combines two or more forms of inheritance. When a derived class inherits from multiple base classes and its entire base classes inherit from a single base class, this form of inheritance is known as hybrid inheritance.

70) **Data File:** A data file is a bunch or stream of bytes stored on some secondary storage devices. These files store data concerning to a specific application for later use.

71) **Text File:** It is a file that stores information in ASCII characters. In text files, each line of text is terminated with a special character known as EOL (*End of Line*) character or delimiter character. When this EOL character is read or written, certain internal translation take place. The file extension of text file is .txt.

72) **Binary File:** It is a file that contains information in the same format, as it is held in memory. In binary files, no delimiters are used for a line and no translations occur here.

73) **File Stream:** A stream is a sequence of bytes. A stream acts as an interface between program and the input/output device.

74) **Pointer:** A pointer is a variable, which holds a memory address. This address is the location of another object (typically another variable) in memory.

75) **Self-Referential Structures:** The self-referential structures are structures that include a member which is a pointer to another structure of the same type.

76) **Data Structure:** Data Structure is group similar or different types of data elements that can be processed as a single unit. A data structure is an arrangement of data in a computer's memory or disk storage. It specifies not only to organize data but also how to access those data.

77) **Primitive Data Structure:** These are directly operated by machine-level instructions. The integer, real, character data, pointer and reference are primitive data structures.

78) **Non-Primitive Data Structure:** These are more complex data structures. These data structures are derived from the primitive data structures. They stress on formation of sets of homogenous and heterogeneous data elements, i.e. array, linked list, stack, queue, tree, graph, etc.

79) **Linear Data Structure:** These are single level data structures, having their elements in a sequence. Example: array, stack, queue and linked list.

80) **Non-Linear Data Structure:** These are multilevel data structures. Example: tree and graph.

81) **Static Data Structure:** These data structures are those whose size, structure and associated memory locations are fixed at compile time. Example: array.

82) **Dynamic Data Structure:** These data structures are ones which can shrink or expand as required during the program execution and change their associated memory locations. Example: Linked list.

83) **Linked List:** Linked lists are among the simplest and most common data structures. A linked list is a list in which address/location of next element is embedded in the current element.

84) **Stack:** A stack is a Last In First Out (LIFO) abstract data type and linear or user-defined data structure. A stack is a list like structure, where stack items can only be added or removed from the one end (i.e. Top) of the stack. It can be implemented using array and using linked list structure, depending upon the requirement.

85) **Queue:** A queue is an ordered collection of items from which items may be deleted at one end (called the front end of the queue) and into which items may be inserted at the other end (called the rear end of the queue).

86) **Database Management System (DBMS):** A Database Management System is a set of programs that enables users to define, create and maintain the database and provides controlled access to this database.

87) **Virtual Functions:** Virtual functions of the base class should be redefined in the derived classes. The programmer can define a virtual function in a base class and can then use the same function name in any derived class, even if the number and type of argument are matching. The matching function rides the base class function of a similar name.

88) **Pure Virtual Functions:** A pure virtual function declared in the base class cannot be used for any operation. The class containing the pure virtual function cannot be used to declare objects. A pure virtual function is similar to an unfilled container that the derived class is made to fill.

89) **Template:** The template is one of the most useful characteristics of the C++. Templates are a part of ANSI C++ standard. The template provides generic programming by defining generic class. Templates help the programmer declare groups of functions or classes.

90) **Object Slicing:** Virtual functions permit us to manipulate both base and derived objects using similar member functions with no modifications. Virtual functions can be invoked using pointer or reference. If we do so, object slicing takes place.