**Daily Assessment Report**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **25-06-20** | **Name:** | **Anand kumar k** |
| **Course:** | **C++** | **USN:** | **4AL16EC002** |
| **Topic:** |  | **Semester & Section:** | **8TH & A** |
| **Github Repository:** | **Anand-courses** |  |  |

|  |
| --- |
| **FORENOON SESSION DETAILS** |
| **Image of session** |
| Inheritance  The capability of a class to derive properties and characteristics from another class is called **Inheritance**. Inheritance is one of the most important feature of Object Oriented Programming. **Sub Class:** The class that inherits properties from another class is called Sub class or Derived Class. **Super Class:**The class whose properties are inherited by sub class is called Base Class or Super class.  Implementing inheritance in C++: For creating a sub-class which is inherited from the base class we have to follow the below syntax. Syntax:  class subclass\_name : access\_mode base\_class\_name  {  //body of subclass  };  Modes of Inheritance   1. Public mode: If we derive a sub class from a public base class. Then the public member of the base class will become public in the derived class and protected members of the base class will become protected in derived class. 2. Protected mode: If we derive a sub class from a Protected base class. Then both public member and protected members of the base class will become protected in derived class. 3. Private mode: If we derive a sub class from a Private base class. Then both public member and protected members of the base class will become Private in derived class.   Derived class constructors and destructors  Constructors and Destructors for Derived Classes. When an object of a derived class is created, both its base and derived parts might need initialization. The base part of the derived class object and its derived part are created in a rigid sequence. Polymorphism in C++ The word polymorphism means having many forms. In simple words, we can define polymorphism as the ability of a message to be displayed in more than one form. Real life example of polymorphism, a person at the same time can have different characteristic. Like a man at the same time is a father, a husband, an employee. So the same person posses different behavior in different situations. This is called polymorphism. Polymorphism is considered as one of the important features of Object Oriented Programming.  Virtual functions  A virtual function is a member function which is declared within a base class and is re-defined(Overriden) by a derived class. When you refer to a derived class object using a pointer or a reference to the base class, you can call a virtual function for that object and execute the derived class’s version of the function.   * Virtual functions ensure that the correct function is called for an object, regardless of the type of reference (or pointer) used for function call. * They are mainly used to achieve[Runtime polymorphism](https://www.geeksforgeeks.org/polymorphism-in-c/) * Functions are declared with a **virtual**keyword in base class. * The resolving of function call is done at Run-time.   Absract classes  Abstract Class is a class which contains atleast one Pure Virtual function in it. Abstract classes are used to provide an Interface for its sub classes. Classes inheriting an Abstract Class must provide definition to the pure virtual function, otherwise they will also become abstract class. |
| Daily Assessment Report   |  |  |  |  | | --- | --- | --- | --- | | Date: | 25-06-20 | Name: | Anand kumar k | | Course: | C++ | USN: | 4AL16EC002 | | Topic: |  | Semester & Section: | 8TH & A | | Github Repository: | Anand-courses |  |  |  |  | | --- | | Afternoon SESSION DETAILS | | Image of session |  Function template Function templates are special functions that can operate with generic types. This allows us to create a function template whose functionality can be adapted to more than one type or class without repeating the entire code for each type. In C++ this can be achieved using template parameters.  Class template  Templates are powerful features of C++ which allows you to write generic programs. In simple terms, you can create a single function or a class to work with different data types using templates. Templates are often used in larger codebase for the purpose of code reusability and flexibility of the programs. Template Specialization In many cases when working with templates, you'll write one generic version for all possible data types and leave it at that--every vector may be implemented in exactly the same way. The idea of template specialization is to override the default template implementation to handle a particular type in a different way.  For instance, while most vectors might be implemented as arrays of the given type, you might decide to save some memory and implement vectors of bools as a vector of integers with each bit corresponding to one entry in the vector. So you might have two separate vector classes. The first class would look like this.   |  |  | | --- | --- | |  | template <typename T>  class vector  {      // accessor functions and so forth      private:      T\* vec\_data;   // we'll store the data as block of dynamically allocated                     // memory      int length;    // number of elements used      int vec\_size;  // actual size of vec\_data  }; |   Exception  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.  *try*: represents a block of code that can throw an exception.  *catch*: represents a block of code that is executed when a particular exception is thrown.  *throw*: Used to throw an exception. Also used to list the exceptions that a function throws, but doesn’t handle itself.  More on files  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. |