**DAILY ASSESSMENT FORMAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **24-06-2020** | **Name:** | **Dhanya Shetty** |
| **Course:** | **C PROGRAMMING** | **USN:** | **4AL17EC026** |
| **Topic:** | **1.CLASSES AND OBJECTS**  **2.MORE ON CLASSES** | **Semester & Section:** | **6th A** |
| **Github Repository:** | **Dhanya Shetty\_026** |  |  |

|  |
| --- |
| **FORENOON SESSION DETAILS** |
| C:\Users\Hp\Desktop\report\24june111.PNG  C:\Users\Hp\Desktop\report\24june222.PNG  **C:\Users\Hp\Desktop\report\24june333.PNG**  **C:\Users\Hp\Desktop\report\24june444.PNG**  **C:\Users\Hp\Desktop\report\24june11111.PNG**  **C:\Users\Hp\Desktop\report\24june22222.PNG**  **C:\Users\Hp\Desktop\report\24june33333.PNG**  **C Programming :**  **C** is highly portable and is **used for** scripting system applications which form a major part of Windows, UNIX, and Linux operating system. **C** is a general-purpose **programming language** and can efficiently work on enterprise applications, games, graphics, and applications requiring calculations, etc.  **C Classes :**  A **class** consists of an instance type and a **class** object: An instance type is a struct containing variable members called instance variables and function members called instance methods. A variable of the instance type is called an instance.  A *class* consists of an instance type and a class object:  An *instance type* is a struct containing variable members called *instance variables* and function members called [instance methods](https://www.pvv.ntnu.no/~hakonhal/main.cgi/c/classes/#instance_methods). A variable of the instance type is called an *instance*.  A *class object* is a global const struct variable containing *class variables* and [class methods](https://www.pvv.ntnu.no/~hakonhal/main.cgi/c/classes/#class_methods). These members belong the whole class without any references to any instances.  A class named "Complex" should name the instance type struct Complex and the class object Complex, and put the interface definitions in "Complex.h" and the implementation in "Complex.c".  Complex.h:  struct Complex {  ...  };  extern const struct ComplexClass {  ...  } Complex;    Complex.c:  #include "Complex.h"  const struct ComplexClass Complex={...};  What is a class in C programming?  In object-oriented **programming**, a **class** is an extensible **program**-code-template for creating objects, providing initial values for state (member variables) and implementations of behavior (member functions or methods). ... In these languages, a **class** that creates **classes** is called a metaclass.  Storage classes in C language?  A **storage class** defines the scope (visibility) and life-time of variables and/or functions within a **C Program**. They precede the type that they modify. We have four different **storage classes** in a **C program** − auto. Register.  A **Class** is a user defined data-type which has data members and member functions. Data members are the data variables and member functions are the functions used to manipulate these variables and together these data members and member functions defines the properties and behavior of the objects in a **Class**.  **Objects :**  **Objects in C**. Redland uses **objects** and is written in **C**, which has no built in support for **object** construction, destruction, copying etc. ... A class FOO is defined as a **C** typedef librdf\_foo , and its a public interface defined in rdf\_foo. h along with any public or private types, enumerations or constants.  Real programmers can write **object**-oriented code in **ANY** language. But no, **C** is not an '**object**-oriented' language. It has no concept of classes, **objects**, polymorphism, inheritance.  In **C**++, **objects are created** as soon as they're defined. You do not need to use the new keyword. If you simply define an **object**, as in your first and third lines, then it is **created** with automatic storage duration: that is, the **object** exists until it goes out of scope.  In **object-oriented programming** (**OOP**), **objects** are the things you think about first in designing a program and they are also the units of code that are eventually derived from the process. ... Each **object** is an instance of a particular class or subclass with the class's own methods or procedures and data variables.  **Object** − **Objects** have states and behaviors. **Example**: A dog has states - color, name, breed as well as behaviors – wagging the tail, barking, eating. An **object** is an instance of a class. Class − A class can be defined as a template/blueprint that describes the behavior/state that the **object** of its type support.  **C++** supports **object**-**oriented** programming, but OO is not intrinsic to the language. In fact, the main function isn't a member of an **object**. ... (Of course, one can argue about Java being a completely **object**-**oriented** language too, because its primitives (say, int) are not **objects**.)  An **object** is a noun (or pronoun) that is governed by a verb or a preposition. There are three kinds of **object**: Direct **Object** (e.g., I know him.) Indirect **Object** (e.g., Give her the prize.) **Object** of a Preposition (e.g., Sit with them.)  **Classes and Objects** are basic concepts of **Object** Oriented Programming which revolve around the real life entities. **Class**. A **class** is a user defined blueprint or prototype from which **objects** are created. It represents the set of properties or methods that are common to all **objects** of one type.  In **object**-oriented programming (OOP), **objects** are **the** things you think about first in designing a program and they are also **the** units of code that are eventually derived from **the** process. ... Each **object** is an instance of a particular class or subclass with **the** class's own methods or procedures and data variables  **More on Classes**  This section covers more aspects of classes that depend on using object references and the dot operator that you learned about in the preceding sections on objects:   * Returning values from methods. * The this keyword. * Class vs. instance members. * Access control   The classes are the most important feature of C++ that leads to Object Oriented programming. Class is a user defined data type, which holds its own data members and member functions, which can be accessed and used by creating instance of that class.  The variables inside class definition are called as data members and the functions are called member functions.  **For example:** Class of birds, all birds can fly and they all have wings and beaks. So here flying is a behavior and wings and beaks are part of their characteristics. And there are many different birds in this class with different names but they all posses this behavior and characteristics.  Similarly, class is just a blue print, which declares and defines characteristics and behavior, namely data members and member functions respectively. And all objects of this class will share these characteristics and behavior  The classes are the most important feature of C++ that leads to Object Oriented programming. Class is a user defined data type, which holds its own data members and member functions, which can be accessed and used by creating instance of that class.  The variables inside class definition are called as data members and the functions are called member functions.  **For example:** Class of birds, all birds can fly and they all have wings and beaks. So here flying is a behavior and wings and beaks are part of their characteristics. And there are many different birds in this class with different names but they all posses this behavior and characteristics.  Similarly, class is just a blue print, which declares and defines characteristics and behavior, namely data members and member functions respectively. And all objects of this class will share these characteristics and behavior.  **More about Classes**   1. Class name must start with an uppercase letter(Although this is not mandatory). If class name is made of more than one word, then first letter of each word must be in uppercase. *Example*, 2. Class Study, class StudyTonight etc. 3. Classes contain, data members and member functions, and the access of these data members and variable depends on the access specifiers (discussed in next section). 4. Class's member functions can be defined inside the class definition or outside the class definition. 5. Class in C++ are similar to structures in C, the only difference being, class defaults to private access control, whereas structure defaults to public. 6. All the features of OOPS, revolve around classes in C++. Inheritance, Encapsulation, Abstraction etc. 7. Objects of class holds separate copies of data members. We can create as many objects of a class as we need. 8. Classes do possess more characteristics, like we can create abstract classes, immutable classes, all this we will study later.   Objects of Classes  Class is mere a blueprint or a template. No storage is assigned when we define a class. Objects are instances of class, which holds the data variables declared in class and the member functions work on these class objects.  Each object has different data variables. Objects are initialised using special class functions called **Constructors**. We will study about constructors later.  And whenever the object is out of its scope, another special class member function called **Destructor** is called, to release the memory reserved by the object. C++ doesn't have Automatic Garbage Collector like in JAVA, in C++ Destructor performs this task. |

|  |
| --- |
|  |
|  |
|  |