

DAILY ASSESSMENT FORMAT

Date:	25-06-2020	Name:	Yamuna Shree N
Course:	C++ programming	USN:	4AL17EC097
Topic:	Inheritance and Polymorphism	Semester & Section:	6 th sem 'B' sec
Github Repository:	yamunashree-course		

FORENOON SESSION DETAILS

Inheritance

Inheritance is one of the most important concepts of object-oriented programming. Inheritance allows us to define a class based on another class. This facilitates greater ease in creating and maintaining an application.

The class whose properties are inherited by another class is called the **Base** class. The class which inherits the properties is called the **Derived** class. For example, the **Daughter** class (derived) can be inherited from the **Mother** class (base). The derived class inherits all feature from the base class, and can have its own additional features.

Base Class

base class

Derived Class
(inherited from base class)

base class features

Inheritance

When inheriting classes, the base class' constructor and destructor are not inherited. However, they are being called when an object of the derived class is created or deleted.

To further explain this behavior, let's create a sample class that includes a constructor and a destructor:

```

class Mother {
public:
    Mother()
    {
        cout << "Mother ctor" << endl;
    }
    ~Mother()
    {
        cout << "Mother dtor" << endl;
    }
};
    
```

Creating an object in main results in the following output:



Polymorphism:

The word polymorphism means having many forms. Typically, polymorphism occurs when there is a hierarchy of classes and they are related by inheritance. C++ polymorphism means that a call to a member function will cause a different function to be executed depending on the type of object that invokes the function.

Difference between Inheritance and Polymorphism :

Inheritance:

Inheritance is one in which a new class is created that inherits the properties of the already exist class. It supports the concept of code reusability and reduces the length of the code in object-oriented programming.

Types of Inheritance are:

1. Single inheritance
2. Multi-level inheritance
3. Multiple inheritance
4. Hybrid inheritance
5. Hierarchical inheritance

Polymorphism:

Polymorphism is that in which we can perform a task in multiple forms or ways. It is applied to the functions or methods. Polymorphism allows the object to decide which form of the function to implement at compile-time as well as run-time.

Types of Polymorphism are:

1. Compile-time polymorphism (Method overloading)
2. Run-time polymorphism (Method Overriding)

Example of Polymorphism:

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 possesses different behavior in different situations. This is called polymorphism. Polymorphism is considered as one of the important features of Object Oriented Programming.

In C++ polymorphism is mainly divided into two types:

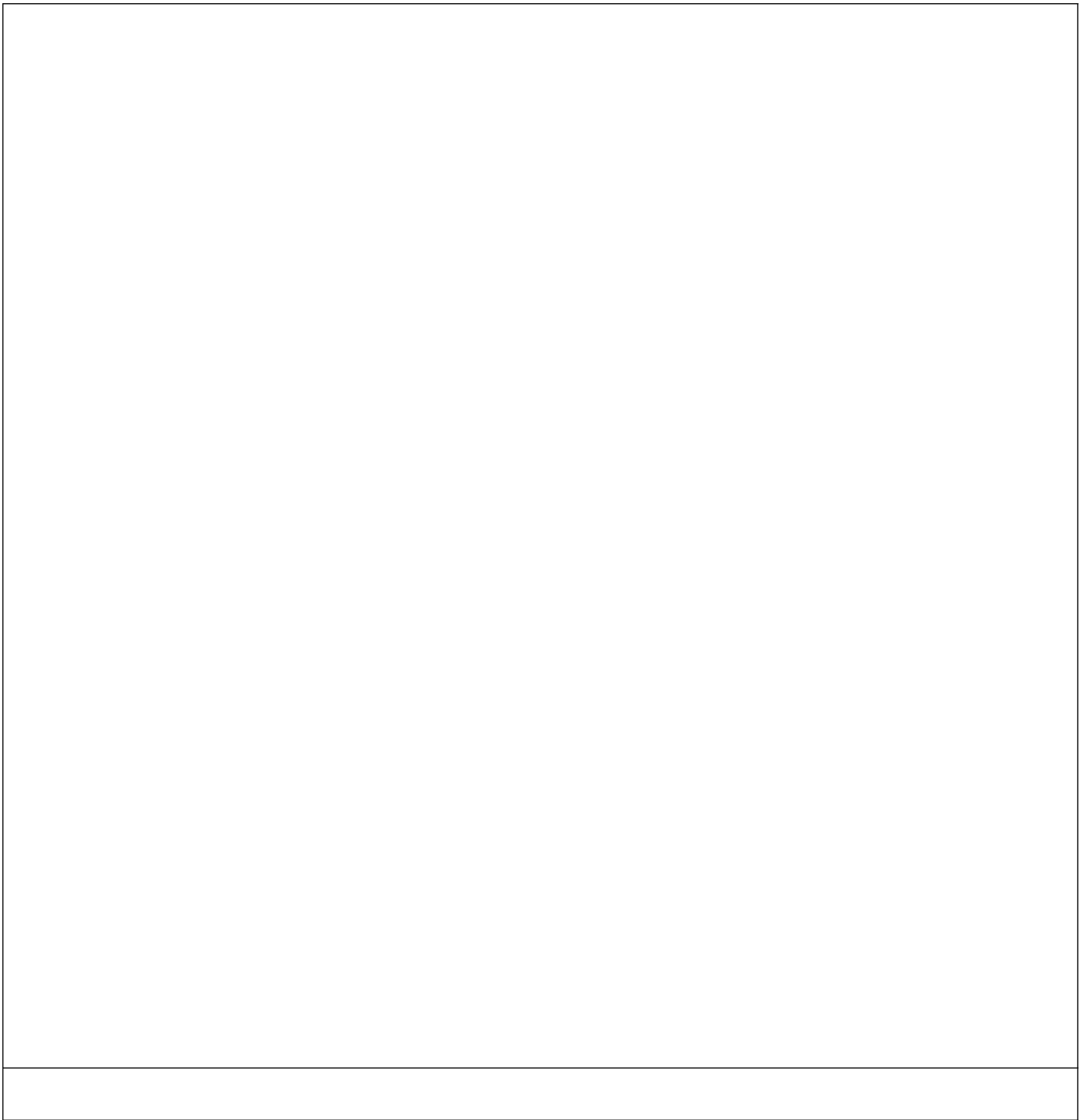
1. Compile time Polymorphism
2. Runtime Polymorphism

C++ Inheritance. In C++, inheritance is a process in which one object acquires all the properties and behaviors of its parent object automatically. ... In C++, the class which inherits the members of another class is called derived class and the class whose members are inherited is called base class. Inheritance in C++ 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.



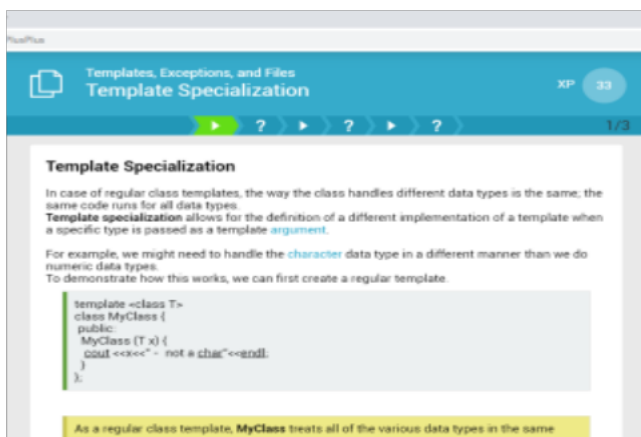
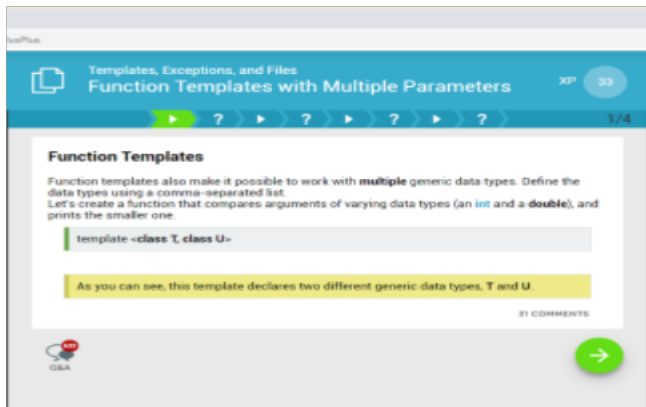


DAILY ASSESSMENT FORMAT

Date:	25-06-2020	Name:	Yamunashree N
Course:	C++ programming	USN:	4AL17EC097
Topic:	Templates, exceptions and files	Semester & Section:	6 th sem 'B' sec

AFTERNOON SESSION DETAILS

Image of session



Templates in C++

Templates are the foundation of generic programming, which involves writing code in a way that is independent of any particular type. A template is a blueprint or formula for creating a generic class or a function. The library containers like iterators and algorithms are examples of generic programming and have been developed using template concept. There is a single definition of each container, such as vector, but we can define many different kinds of vectors for example, vector or vector. 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.

Files :

C++ provides the following classes to perform output and input of characters to/from files:

1. `ofstream`: Stream class to write on files
2. `ifstream`: Stream class to read from files
3. `fstream`: Stream class to both read and write from/to files.

These classes are derived directly or indirectly from the classes `istream` and `ostream`. We have already used objects whose types were these classes: `cin` is an object of class `istream` and `cout` is an object of class `ostream`. Therefore, we have already been using classes that are related to our file streams. And in fact, we can use our file streams the same way we are already used to use `cin` and `cout`, with the only difference that we have to associate these streams with physical files.

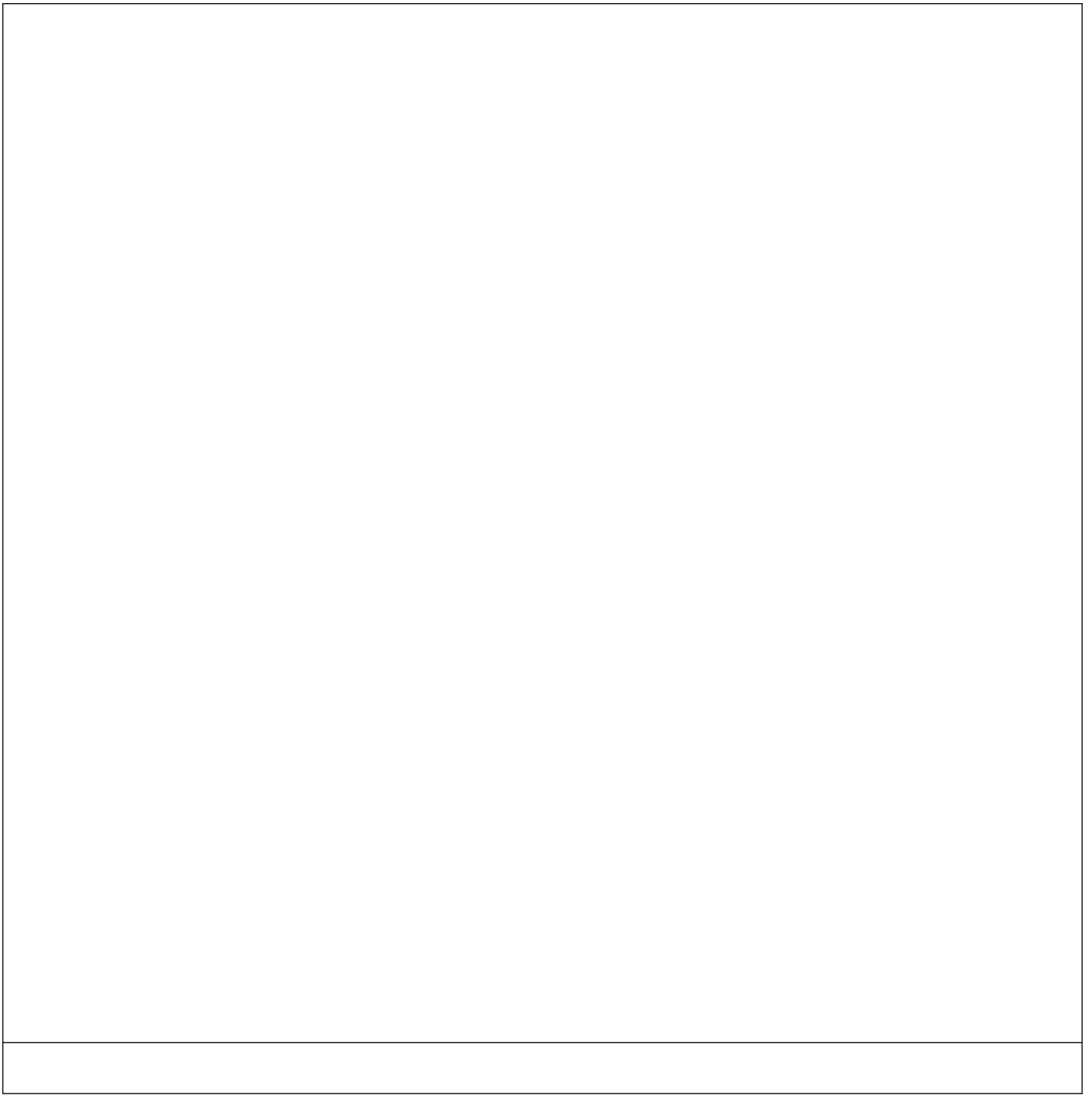
Exceptions :

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`.

1. `throw` – A program throws an exception when a problem shows up. This is done using a `throw` keyword.
2. `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.
3. `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.







Edit with WPS Office