OBJECTIVE

- To understand the working process of classes and objects.
- To initialize constructors and destructors for their own purpose.
- To enhance the concept of object and classes along with their uses.
- To promote code reusability using OOP features.

BACKGROUND THEORY

Object-Oriented Programming was invented with the purpose of solving the problems caused by previous programming language. It contains various concepts and building blocks that made the coders and developers more easier to code. Objects, classes, constructor, destructor, encapsulation, polymorphism, etc are the essential keys for OOP.

Object

An object is a real-world or logical entity that contains both data (attributes) and operations (methods). Objects are the basic runtime entities in OOP and are created using classes. Multiple objects can be created by a single class Objects are the basic runtime entities which may be created or destroyed at run time.

Class:

A class is a blueprint or template from which objects are created. It defines the data and the operations that can be performed on that data. Class has three access specifiers: public, private and protected. So, class promotes the concept of data hiding. For example, car is a class and color and drive are its members.

Syntax Example in C++:

```
class Car {

private:

string color;

public:

void driver( )

{

//driving logic

}
```

Constructors:

A constructor is a specialized function of a class that has the same name as the class and no return type. It is automatically invoked at the time of object creation to assign initial values to the data members of the class.

Function of constructor:

- To initialize object automatically when they are created.
- To avoid manual initialization using a separate

Types of constructors:

a. Default constructor
   It is a constructor that takes no arguments. It is used to initialize object with default values when no specific values are passed.

Sytanx:

```
class  ClassName{

public:

ClassName( );

};
```

b. Parameterized constructor:
   It is a constructor that takes arguments/parameters. It is used to initialize objects with specific values at the time of creation.

Syntax:

```
class ClassName {

public:

ClassName(data_type parameter1, data_type parameter2, ...);

};
```

c. Copy constructor:
   A copy constructor in C++ is a special constructor that creates a new object by initializing it with an existing object of the same class. It takes a reference to a constant object of the same class as its parameter and is used to perform member-wise copying of data.

Syntax:

```
class ClassName {

public:

ClassName(const ClassName &obj);

};
```

Destructor:

In C++, a destructor is a special member function that is automatically called when an object goes out of scope or is explicitly deleted. Its primary purpose is to free resources that the object may have acquired during its lifetime, such as memory, file handles, or network connections. Its name is as same as the class and takes no argument and return type.

Functions of destructor:

- To perform clean-up tasks (e.g., releasing memory, closing files).
- To avoid memory leaks in programs using dynamic memory.

Syntax:

```
~ClassName() {
// required code
}
```