Object Oriented Programming through C++

UNIT-2

Classes and Objects:

- Classes: A class is a user-defined data type that encapsulates data and functions into a single unit. It serves as a blueprint for creating objects. It defines the structure and behavior of objects that belong to it. The data members represent the attributes of the class, and the member functions define the operations that can be performed on the objects of the class.
- Objects: Objects are instances of a class. They represent a particular entity or instance of the class. Each object has its own set of data members and can invoke the member functions defined in the class.

Defining Member Functions:

- Member Functions: Member functions are functions that are declared and defined inside a class. They operate on objects of the class and have access to the data members of the class. Member functions can be defined within the class definition or outside the class using the scope resolution operator (::).

Array of Class Objects:

- Array of Class Objects: It is possible to create an array of objects of a class. Each element of the array represents an object, and you can access the individual objects using array indexing. This allows you to work with multiple objects of the same class in a convenient manner.

Pointers and Classes:

- Pointers and Classes: Pointers can be used to create and manipulate objects of a class dynamically. You can create a pointer to a class and allocate memory for an object using the `new` operator. Pointers to objects can then be used to access the member functions and data members of the class.

Passing Objects:

- Passing Objects: Objects can be passed as arguments to functions. When an object is passed as a function argument, a copy of the object is created. You can pass objects by value, by reference, or by constant reference. Passing objects by reference allows the function to modify the original object.

Constructors:

- Constructors: Constructors are special member functions that are automatically called when an object is created. They initialize the object's data members and perform any necessary setup. Constructors have the same name as the class and do not have a return type. They can be overloaded to provide different ways of initializing objects.

Types of Constructors:

- Default Constructor: A default constructor is a constructor that takes no arguments. It is called when an object is created without any initializer.
- Parameterized Constructor: A parameterized constructor is a constructor that takes one or more parameters. It allows you to initialize the object's data members with specific values.
- Copy Constructor: A copy constructor is a constructor that creates a new object by making a copy of an existing object. It is called when an object is initialized with another object of the same class.

Destructors:

- Destructors: Destructors are special member functions that are called when an object is destroyed or goes out of scope. They are used to perform cleanup operations, such as releasing dynamically allocated memory or closing open files. Destructors have the same name as the class preceded by a tilde (~) and do not take any arguments.

this Pointer:

- this Pointer: The `this` pointer is a special pointer that is available within the member functions of a class. It points to the object that invoked the member function. It can be used to access the data members and member functions of the current object.

Access Specifiers:

- Access Specifiers: Access specifiers are keywords used to control the accessibility of the class members. There are three access specifiers in C++:
    - Public: Public members are accessible from anywhere in the program. They can be accessed by objects of the class, derived classes, and external functions.
    - Private: Private members are only accessible within the class itself. They cannot be accessed by objects of the class or external functions.
    - Protected: Protected members are similar to private members, but they can also be accessed by derived classes.

Friend Functions:

- Friend Functions: Friend functions are non-member functions that are granted access to the private and protected members of a class. They are declared as

friends inside the class. Friend functions can be useful when you need to allow external functions to access private or protected members of a class.

Static Member of Class:

- Static Member of Class: A static member of a class is a member that belongs to the class itself rather than to individual objects of the class. It is shared among all objects of the class and can be accessed using the class name. Static members can be data members or member functions. They are not associated with any particular object and can be accessed even when no objects of the class are created.

Exam Type Questions

Question 1: What are the types of constructors in C++? Explain each type with an example.

Answer:

There are three types of constructors in C++:

1. Default Constructor:
    - A constructor with no arguments.
    - It is automatically invoked when an object is created.
    - Example:
    - vbnet
    - Copy code
    - class MyClass { public: MyClass() { // Default constructor implementation } };

2. Parameterized Constructor:
    - A constructor with one or more arguments.
    - It allows initialization of data members with specific values.
    - Example:

```
class Rectangle { private: int length; int width; public: Rectangle(int l, int w) { length = l; width = w; } };
```

3. Copy Constructor:
    - A constructor that creates a new object by copying the values from an existing object.
    - It is invoked when an object is initialized with another object of the same class.
    - Example:

```
class Point { private: int x; int y; public: Point(const Point&
```


2. Question: What is the role of the "this" pointer in C++? Explain with an example.

Answer:

The "this" pointer in C++ is a special pointer that holds the memory address of the current object. Its primary role is to refer to the object on which a member function is called.

Example:

```
class Circle { private: double radius; public: void setRadius(double r) { this->radius = r; } };
```

In the above example, the "this" pointer is used to assign the value of the "r" parameter to the "radius" data member of the current object.

3. Question: What are access specifiers in C++? Explain the three types of access specifiers.

Answer:

Access specifiers in C++ are keywords used to define the visibility and accessibility of class members. There are three types of access specifiers:

1. Private:
    - Members declared as private are only accessible within the class.
    - They cannot be accessed directly from outside the class.
    - Example:

```
class MyClass { private: int privateVar; };
```

2. Protected:
    - Members declared as protected have limited accessibility.
    - They can be accessed by derived classes and within the class itself.
    - They are not accessible outside the class hierarchy.
    - Example:

```
class BaseClass { protected: int protectedVar; };
```

3. Public:
    - Members declared as public are accessible from anywhere.
    - They can be accessed by objects of the class and outside the class.

```
class MyClass { public: int publicVar; };
```

4. Question: What are friend functions in C++? How are they different from member functions?

Answer:

Friend functions in C++ are functions that are not members of a class but have access to its private and protected members. They are declared inside the class with the keyword "friend" and can be defined outside the class.

Differences between friend functions and member functions:

1. Scope: Friend functions are not members of the class, while member functions belong to the class.
2. Access to private members: Friend functions can access private members of the class, whereas member functions have direct access to them.
3. Invocation: Friend functions can be invoked without the need for an object, while member functions require an object to be called.

5. Question: What is a static member of a class? Explain its characteristics and usage.

Answer:

A static member of a class in C++ is a member that belongs to the class rather than individual objects. It is shared by all objects of the class and exists independently of any object. Some characteristics and usage of static members are:

1. Shared memory: Static members are stored only once in memory and shared among all objects of the class.
2. Accessibility: Static members can be accessed using the class name, without the need for an object.
3. Initialization: Static members need to be explicitly defined and initialized outside the class.

4. Visibility: Static members can access only other static members and cannot access non-static members.
5. Common usage: Static members are often used to represent class-wide variables, constants, or utility functions that are not tied to specific objects.