Objectives:

· Learn how to create and use classes and objects in C++.

· Use constructors to set up objects when they are created.

· Understand destructors and how they clean up when objects are destroyed.

· Write simple programs using classes, constructors, and destructors.

Tools and Libraries Used :

· Programming Language: C++

· IDE: Visual Studio Code

· Libraries: #include<iostream>, #include<string>

Theory

Classes and Objects in C++

In C++, a class is a user-defined data type that acts as a blueprint for creating objects. A class can contain data members (variables) and member functions (methods) which define the behavior of the class.

An object is an instance of a class. When you create an object, you're creating a variable that has its own copy of the class's data and access to the class's functions.

Syntax of a Class:

```
1. class ClassName {
2. public:
3. // data members
4. // member functions
5. };
```

By default, members of a class are private, but using public:, we make them accessible from outside the class.Example:

```
1. #include <iostream>
2. using namespace std;
3.
4. class Car {
5. public:
6. string brand;
7. int year;
8.
9. void display() {
10. cout << "Brand: " << brand << ", Year: " << year << endl;
11. }
12. };
13.
14. int main() {
15. Car myCar; // Creating an object of Car
16. myCar.brand = "Toyota";
17. myCar.year = 2021;
18. myCar.display(); // Accessing member function
19. return 0;
20. }
```

· Car is a class with two variables (brand and year) and a function display().

· myCar is an object created from the Car class.

· We assign values to the object and call its method.

Constructors in C++

A constructor is a special member function that is automatically called when an object is created. Its main job is to initialize the object.

Rules and Features:

· The constructor has the same name as the class.

· It has no return type, not even void.

· You can have multiple constructors (constructor overloading).

Syntax:

```
1. class ClassName {
2. public:
3. ClassName(parameters) {
4. // constructor body
5. }
6. };
```
7.Example with Constructor:
```
1. #include <iostream>
2. using namespace std;
3.
4. class Car {
5. public:
6. string brand;
7. int year;
8.
9. // Constructor
10. Car(string b, int y) {
11. brand = b;
12. year = y;
13. }
14.
15. void display() {
16. cout << "Brand: " << brand << ", Year: " << year << endl;
17. }
18. };
19.
20. int main() {
21. Car myCar("Honda", 2019); // Constructor is automatically called
22. myCar.display();
23. return 0;
24. }
```
Explanation:

· The constructor Car(string b, int y) sets the initial values of brand and year when the object is created.

· No need to assign values manually later in main().

Destructors in C++

A destructor is a special function that is automatically called when an object goes out of scope or is explicitly deleted. It is used to clean up resources such as memory, files, or database connections.

Rules:

· It has the same name as the class, but preceded by a tilde (~).

· It has no parameters and no return type.

· A class can have only one destructor.Syntax:
```
1. class ClassName {
2. public:
3. ~ClassName() {
4. // cleanup code
5. }
6. };
```
Example with Destructor:
```
1. #include <iostream>
2. using namespace std;
3.
4. class Car {
5. public:
6. string brand;
7.
8. // Constructor
9. Car(string b) {
10. brand = b;
11. cout << brand << " is created." << endl;
12. }
13.
14. // Destructor
```

```
15. ~Car() {
16. cout << brand << " is destroyed." << endl;
17. }
18. };
19.
20. int main() {
21. Car myCar("BMW");
22. // Destructor will be called automatically when main ends
23. return 0;
24. }
25.
```

Explanation:

· When myCar is created, the constructor runs.

· When main() ends and myCar goes out of scope, the destructor runs automatically.

Qn: Define a class Car with private members brand, model, and year. Include public member functions to set and get these private members. Ensure that only member functions can access these private members.

Code:

```cpp
#include <iostream>
#include <string>
using namespace std;
class Car {
private:
    string brand;
    string model;
    int year;
public:
   void setBrand(const string& b) {
        brand = b;
    }
void setModel(const string& m) {
        model = m;
    }
void setYear(int y) {
        year = y;
    }
string getBrand() const {
        return brand;
    }
string getModel() const {
        return model;
    }
int getYear() const {
        return year;
    }
};

int main() {
    Car myCar;
    myCar.setBrand("Toyota");
    myCar.setModel("Corolla");
    myCar.setYear(2020);

    cout << "Car Details:\n";
    cout << "Brand: " << myCar.getBrand() << endl;
    cout << "Model: " << myCar.getModel() << endl;
    cout << "Year: " << myCar.getYear() << endl;
    return 0;
}
```

Output:

```
Car Details:
Brand: Toyota
Model: Corolla
Year: 2020
Program ended with exit code: 0
```

Qn: Define a class Book with private members title, author, and year. Implement both default and parameterized constructors. The default constructor should initialize the

members with default values, and the parameterized constructor should set these values based on user input. Provide a method to display the details of a book.

Code:

```cpp
#include <iostream>
#include <string>
using namespace std;

class Book {
private:
    string title;
    string author;
    int year;

public:
    Book() {
        title = "Unknown";
        author = "Unknown";
        year = 0;
    }
    Book(string t, string a, int y) {
        title = t;
        author = a;
        year = y;
    }
     void displayDetails() const {
        cout << "Book Title : " << title << endl;
        cout << "Author     : " << author << endl;
        cout << "Year       : " << year << endl;
    }
};
int main() {
    Book book1;
    cout << "Default Book Details:\n";
    book1.displayDetails();
    cout << "\n";
    Book book2("1984", "George Orwell", 1949);
    cout << "Parameterized Book Details:\n";
    book2.displayDetails();

    return 0;
}
```

Output:

```
Default Book Details:
Book Title : Unknown
Author     : Unknown
Year       : 0

Parameterized Book Details:
Book Title : 1984
Author     : George Orwell
Year       : 1949
Program ended with exit code: 0
```

Qn: Create a class Employee with private members name and age. Implement a copy constructor to create a deep copy of an existing Employee object. Provide a method to display the details of an employee.

Code:

```cpp
#include <iostream>
#include <string>
using namespace std;
class Employee {
private:
    string name;
    int age;
public:
    Employee() {
        name = "Unknown";
        age = 0;
    }
    Employee(string n, int a) {
        name = n;
        age = a;
    }
    Employee(const Employee& other) {
        name = other.name;
        age = other.age;
    }
    void display() const {
        cout << "Employee Name: " << name << endl;
        cout << "Employee Age : " << age << endl;
    }
};

int main() {
    Employee emp1("John Doe", 30);
    Employee emp2 = emp1;
    cout << "Original Employee:\n";
    emp1.display();
    cout << "\nCopied Employee:\n";
    emp2.display();
    return 0;
}
```

Output:

```
Original Employee:
Employee Name: John Doe
Employee Age : 30

Copied Employee:
Employee Name: John Doe
Employee Age : 30
Program ended with exit code: 0
```

Qn: Define a class Book with a private member title. Implement a constructor that initializes title and a destructor that prints a message when the Book object is destroyed. Create instances of Book objects in main() to demonstrate the usage of the destructor.

Code:

```cpp
#include <iostream>
#include <string>
using namespace std;

class Book {
private:
    string title;

public:
    Book(string t) {
        title = t;
        cout << "Book \"" << title << "\" is created." << endl;
    }
    ~Book() {
        cout << "Book \"" << title << "\" is destroyed." << endl;
    }
};

int main() {
    {
        Book book1("The Alchemist");
        Book book2("1984");
    }
    cout << "End of main function." << endl;
    return 0;
}
```

Output:

Qn: Define a class Rectangle with private members length and width. Implement a constructor to initialize these members. Write a function calculateArea() that calculates and returns the area of the rectangle. Define another function doubleDimensions(Rectangle rect) that takes a Rectangle object as an argument and doubles its length and width. Demonstrate the usage of both functions in main().

```
Book "The Alchemist" is created.
Book "1984" is created.
Book "1984" is destroyed.
Book "The Alchemist" is destroyed.
End of main function.
Program ended with exit code: 0
```

Code:

```cpp
#include <iostream>
using namespace std;
class Rectangle {
private:
    double length;
    double width;
public:
    Rectangle(double l, double w) {
        length = l;
        width = w;
    }
    double calculateArea() const {
        return length * width;
    }
    void doubleDimensions() {
        length *= 2;
        width *= 2;
    }
    void display() const {
        cout << "Length: " << length << ", Width: " << width << endl;
    }
};
void doubleDimensions(Rectangle& rect) {
    rect.doubleDimensions();
}
int main() {
    Rectangle rect(4.0, 3.0);
    cout << "Original Rectangle:\n";
    rect.display();
    cout << "Area: " << rect.calculateArea() << endl;
    doubleDimensions(rect);
    cout << "\nAfter Doubling Dimensions:\n";
    rect.display();
    cout << "Area: " << rect.calculateArea() << endl;
    return 0;
}
```

Output:

Qn: Define a class Student with private members name and age. Implement a constructor to initialize these members. Create an array studentArray of size 3 to store objects of type Student. Populate the array with student details and display the details using a method displayStudentDetails().

Code:

```cpp
#include <iostream>
#include <string>
using namespace std;
class Student {
private:
    string name;
    int age;
public:
    Student(string n, int a) {
        name = n;
        age = a;
    }
    Student() {
        name = "Unknown";
        age = 0;
    }
    void setStudentDetails(string n, int a) {
        name = n;
        age = a;
    }
    void displayStudentDetails() const {
        cout << "Name: " << name << ", Age: " << age << endl;
    }
};
int main() {
    const int SIZE = 3;
    Student studentArray[SIZE];
    for (int i = 0; i < SIZE; i++) {
        string name;
        int age;
        cout << "Enter name of student " << (i + 1) << ": ";
        getline(cin, name);
        cout << "Enter age of student " << (i + 1) << ": ";
        cin >> age;
        cin.ignore();
        studentArray[i].setStudentDetails(name, age);
    }
    cout << "\nStudent Details:\n";
    for (int i = 0; i < SIZE; i++) {
        studentArray[i].displayStudentDetails();
    }
return 0;
}
```

Output:

Qn: Define a class Math with two overloaded methods add(). The first method should take two integers as parameters and return their sum. The second method should take three integers as parameters and return their sum. Demonstrate the usage of both methods in main().

Code:

```cpp
#include <iostream>
using namespace std;

class Math {
public:
    int add(int a, int b) {
        return a + b;
    }

    int add(int a, int b, int c) {
        return a + b + c;
    }
};

int main() {
    Math math;

    int sum1 = math.add(5, 10);
    int sum2 = math.add(3, 7, 2);

    cout << "Sum of 5 and 10 is: " << sum1 << endl;
    cout << "Sum of 3, 7 and 2 is: " << sum2 << endl;

    return 0;
}
```

Output:

```
Sum of 5 and 10 is: 15
Sum of 3, 7 and 2 is: 12
Program ended with exit code: 0
```

Qn: Implement a class Area with overloaded methods calculate(). The first method should take the radius of a circle as a parameter and return the area of the circle. The second method should take the length and width of a rectangle as parameters and return the area of the rectangle. Demonstrate the usage of both methods in main().

Code:

```cpp
#include <iostream>
#include <cmath>
using namespace std;

class Area {
public:
    double calculate(double radius) {
        return M_PI * radius * radius;
    }
    double calculate(double length, double width) {
        return length * width;
    }
};

int main() {
    Area area;
    double circleRadius = 5.0;
    double rectangleLength = 4.0;
    double rectangleWidth = 6.0;
    double circleArea = area.calculate(circleRadius);
    double rectangleArea = area.calculate(rectangleLength, rectangleWidth);
    cout << "Area of Circle (radius = " << circleRadius << "): " << circleArea << endl;
    cout << "Area of Rectangle (" << rectangleLength << "x" << rectangleWidth << "): " << rectangleArea
        << endl;
    return 0;
}
```

Output:

```
Area of Circle (radius = 5): 78.5398
Area of Rectangle (4x6): 24
Program ended with exit code: 0
```