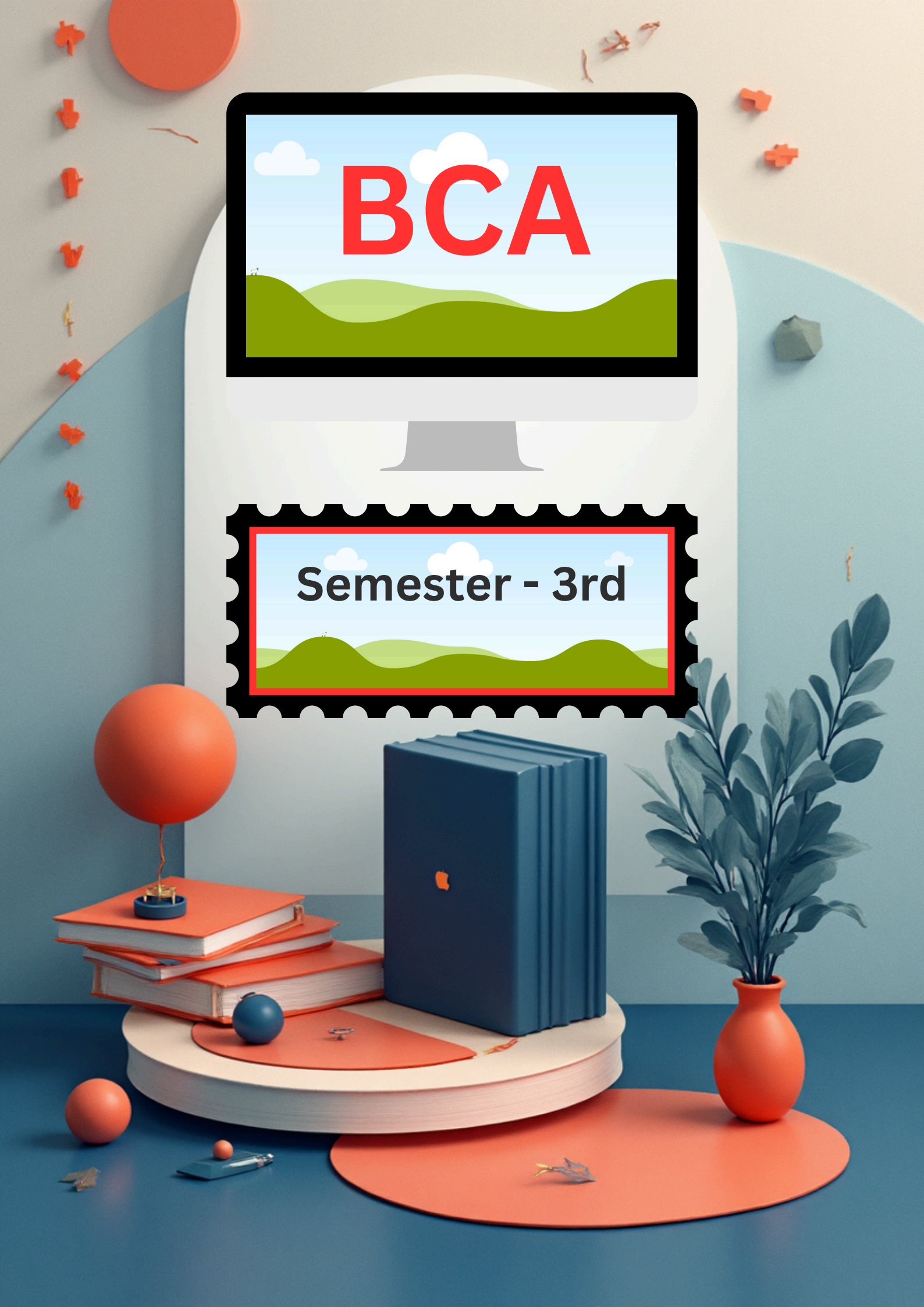
BCA
Semester - 3rd

Object Oriented Programming using C++

Notes - 1

Contents

Object oriented programming concepts Why do we need object oriented. C++ Programming basics: Output using cout. Directives. Input with cin. Type bool, The setw manipulator, Type conversions.

Website - prepfolio.co.in

Object-Oriented Programming (OOP) Concepts

Object-Oriented Programming (OOP) is a programming approach that organizes a program into objects, which represent real-world entities like a person, car, bank account, or student. Each object contains data (attributes) and functions (behaviors) that operate on that data.

In simple words, OOP focuses on “objects” rather than just functions or logic. It helps make the program more structured, reusable, and easier to manage.

Main Concepts of OOP

Class

A class is a blueprint or template for creating objects.
It defines the data members (variables) and member functions (methods).
Example:

```
class Student {
   int roll;
   string name;
   void showData() {
      cout << roll << " " << name;
   }
};
```

Here, Student is a class that defines properties and behavior of a student

Object

An object is a real instance of a class.
It uses the structure and behavior defined by the class.

Example:

```
Student s1;  // s1 is an object of class Student
```

Encapsulation

It means binding data and functions together in a single unit (class) and protecting data from outside interference.

In OOP, data is hidden using private access specifiers, and accessed using public functions.

Example:

```
class Account {
private:
   int balance;
public:
   void setBalance(int b) { balance = b; }
   int getBalance() { return balance; }
};
```

Abstraction

It means showing only essential features and hiding unnecessary details.

For example, when you drive a car, you use the steering and pedals but don’t see the complex engine details.

In programming, we achieve abstraction using classes and functions.

Inheritance

Inheritance allows a class to use the properties and functions of another class.

It supports code reusability and reduces redundancy.

Example:

```
class Animal {
   void eat() { cout << "Eating"; }
};
class Dog : public Animal {
   void bark() { cout << "Barking"; }
};
```

Polymorphism

Polymorphism means one name with many forms.
It allows the same function to behave differently based on the object.
It can be achieved through function overloading or function overriding.

Example:

```
void area(int r);       // circle
void area(int l, int b); // rectangle
```

Why Do We Need Object-Oriented Programming?

- Real-World Representation

OOP models real-world entities as objects, making programs more understandable and closer to real-life systems.

- Reusability

Once a class is written, it can be reused in other programs without rewriting the same code.

- Data Security

Encapsulation and access specifiers keep data safe from unauthorized access or modification.

- Easy Maintenance and Modification

Since data and functions are combined in objects, making changes in one part of the program doesn't affect others easily.

- Extensibility

OOP allows new features to be added easily using inheritance and polymorphism without changing existing code.

- Better Organization

The object-based structure makes large programs easier to manage, debug, and update.

C++ Programming Basics

C++ is an object-oriented programming language that combines the features of C language with additional concepts like classes and objects. Before learning advanced concepts, it's important to understand the basic input-output operations, directives, data types, and type conversions in C++.

1. Output Using `cout`

`cout` (pronounced as "see-out") is used in C++ to display output on the screen.
It is part of the **iostream** library and works with the **insertion operator** (`<<`).

Syntax:

```cpp
cout << "Message or data";
```

Example:

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

int main() {
    cout << "Hello, World!" << endl;
    cout << "The sum is: " << 10 + 20;
    return 0;
}
```

Explanation (in simple words):

`cout` likhne ka matlab hota hai output dikhana. `<<` operator se hum screen pe text ya variable value show karte hain.

2. Directives

A **directive** gives instructions to the compiler before the actual program starts.
They begin with the symbol # and are known as **preprocessor directives**.

Common Directives:

- `#include` → Used to include header files.
  Example: `#include <iostream>`
- `#define` → Used to define constants or macros.
  Example: `#define PI 3.14`

Example Program:

cpp Copy code

```cpp
#include <iostream>  // include directive
#define PI 3.1416    // define directive

using namespace std;

int main() {
    cout << "Value of PI: " << PI;
    return 0;
}
```

3. Input with `cin`

`cin` (pronounced "see-in") is used to take input from the user.
It works with the **extraction operator** (`>>`).

Syntax:

cpp Copy code

```cpp
cin >> variable;
```

Example:

cpp Copy code

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

int main() {
    int age;
    cout << "Enter your age: ";
    cin >> age;
    cout << "Your age is " << age;
    return 0;
}
```

4. Type `bool`

The `bool` data type is used to store **Boolean values** — `true` (1) or `false` (0).

Example:

cpp Copy code

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

int main() {
    bool isRaining = true;
    bool isSunny = false;
    cout << "Is it raining? " << isRaining << endl;
    cout << "Is it sunny? " << isSunny;
    return 0;
}
```

Output:

vbnet Copy code

```vbnet
Is it raining? 1
Is it sunny? 0
```

5. The `setw` Manipulator

`setw()` stands for **set width**.

It is used to **format the output** by setting the width of the output field.

It is part of the **iomanip** library.

Syntax:

cpp Copy code

```cpp
cout << setw(width) << value;
```

Example:

cpp Copy code

```cpp
#include <iostream>
#include <iomanip>   // required for setw
using namespace std;

int main() {
    cout << setw(10) << "Roll" << setw(10) << "Name" << endl;
    cout << setw(10) << 1 << setw(10) << "Amit" << endl;
    cout << setw(10) << 2 << setw(10) << "Rahul";
    return 0;
}
```

Output:

```markdown
Copy code

      Roll      Name
         1      Amit
         2     Rahul
```

Explanation:

`setw()` output ko properly align karne ke liye use hota hai, jaise table format me data show karna.

---

6. Type Conversions

Type conversion means changing the **data type of a variable** from one type to another.

There are two types:

(a) Implicit Conversion (Type Promotion)

Done **automatically by the compiler**.

Example:

```cpp
Copy code

int a = 5;
float b = 6.5;
float c = a + b;  // int 'a' is automatically converted to float
```

(b) Explicit Conversion (Type Casting)

Done **manually by the programmer** using type casting.

Syntax:

```cpp
Copy code

(data_type) variable;
```

Example:

```cpp
Copy code

#include <iostream>
using namespace std;

int main() {
    float a = 5.75;
    int b = (int)a;   // explicit conversion
    cout << "Float value: " << a << endl;
    cout << "Integer value: " << b;
    return 0;
}
```

Output:

```sql
Float value: 5.75
Integer value: 5
```

Explanation:

Type conversion ek type ka data dusre type me badal deta hai. Implicit me compiler khud karta hai, explicit me hum manually karte hain `(int)` ya `(float)` likh ke.

✅ In Summary

Concept	Description
`cout`	Displays output using `<<`
`cin`	Takes input from user using `>>`
Directives	Instructions to compiler (`#include` , `#define`)
`bool` type	Stores `true` (1) or `false` (0) values
`setw()`	Used for formatted (aligned) output
Type Conversion	Converts one data type into another (implicit or explicit)

Practice Questions

1. What is Object-Oriented Programming (OOP)? Explain its main concepts.
2. Why do we need Object-Oriented Programming?
3. Define Class and Object in C++ with examples.
4. Explain the concept of Encapsulation and Abstraction with suitable examples.
5. What is Inheritance? Explain its importance with an example.
6. What is Polymorphism in C++? Explain types of polymorphism with examples.
7. Explain the use of cout and cin in C++ with examples.
8. What are Preprocessor Directives in C++? Explain with suitable examples.
9. What is the purpose of the setw() manipulator in C++? Explain with example.
10. What do you mean by Type Conversion in C++? Explain implicit and explicit conversion with examples.

Check the answer in the Practice Questions section on our website.

Visit Website