

Practical No. 1

Date: - / /

Q.1) Write a simple program (Without class) to use of operators in C++.

```
#include <iostream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    // Arithmetic operators
```

```
    int a = 10, b = 5;
```

```
    cout << "Arithmetic Operators:" << endl;
```

```
    cout << "a + b = " << a + b << endl;
```

```
    cout << "a - b = " << a - b << endl;
```

```
    cout << "a * b = " << a * b << endl;
```

```
    cout << "a / b = " << a / b << endl;
```

```
    cout << "a % b = " << a % b << endl;
```

```
    // Increment and Decrement operators
```

```
    int num = 5;
```

```
    cout << "\nIncrement and Decrement Operators:" << endl;
```

```
    cout << "Original value of num: " << num << endl;
```

```
    cout << "num++ is " << num++ << endl;
```

```
    cout << "After increment, num: " << num << endl;
```

```
    cout << "++num is " << ++num << endl;
```

```
cout << "After pre-increment, num: " << num << endl;

cout << "num-- is " << num-- << endl;

cout << "After decrement, num: " << num << endl;

cout << "--num is " << --num << endl;

cout << "After pre-decrement, num: " << num << endl;
```

```
// Assignment operators
```

```
int var = 10;

cout << "\nAssignment Operators:" << endl;

var += 5;

cout << "var += 5: " << var << endl;

var -= 3;

cout << "var -= 3: " << var << endl;

var *= 2;

cout << "var *= 2: " << var << endl;

var /= 4;

cout << "var /= 4: " << var << endl;

var %= 3;

cout << "var %= 3: " << var << endl;

return 0;

}
```

Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\91930\Desktop\cpp journal> g++ .\PractNo1.cpp
PS C:\Users\91930\Desktop\cpp journal> .\a.exe
Arithmetic Operators:
a + b = 15
a - b = 5
a * b = 50
a / b = 2
a % b = 0

Increment and Decrement Operators:
Original value of num: 5
num++ is 5
After increment, num: 6
++num is 7
After pre-increment, num: 7
num-- is 7
After decrement, num: 6
--num is 5
After pre-decrement, num: 5

Assignment Operators:
var += 5: 15
var -= 3: 12
var *= 2: 24
var /= 4: 6
var %= 3: 0
PS C:\Users\91930\Desktop\cpp journal> █
```

Practical No. 2

Date: - / /

Q.1) Illustration Control Structures.

```
#include <iostream>

using namespace std;

int main()
{
    // If-else statement

    int num = 10;

    cout << "If-else statement:" << endl;

    if (num > 0)
    {
        cout << num << " is positive." << endl;
    }
    else
    {
        cout << num << " is not positive." << endl;
    }

    // Switch statement

    char grade = 'B';

    cout << "\nSwitch statement:" << endl;

    switch (grade)
```

```
{  
case 'A':  
    cout << "Excellent!" << endl;  
    break;  
case 'B':  
    cout << "Well done!" << endl;  
    break;  
case 'C':  
    cout << "You passed." << endl;  
    break;  
default:  
    cout << "Invalid grade." << endl;  
}
```

// While loop

```
cout << "\nWhile loop:" << endl;  
int i = 0;  
while (i < 5)  
{  
    cout << i << " ";  
    i++;  
}  
cout << endl;
```

```
// For loop

cout << "\nFor loop:" << endl;

for (int j = 0; j < 5; j++)

{

    cout << j << " ";

}

cout << endl;


// Do-while loop

cout << "\nDo-while loop:" << endl;

int k = 0;

do

{

    cout << k << " ";

    k++;

} while (k < 5);

cout << endl;


return 0;

}
```

Output:

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\91930\Desktop\cpp journal> g++ .\PractNo2.cpp
```

```
PS C:\Users\91930\Desktop\cpp journal> .\a.exe
```

```
If-else statement:
```

```
10 is positive.
```

```
Switch statement:
```

```
Well done!
```

```
While loop:
```

```
0 1 2 3 4
```

```
For loop:
```

```
0 1 2 3 4
```

```
Do-while loop:
```

```
0 1 2 3 4
```

```
PS C:\Users\91930\Desktop\cpp journal> █
```

Practical No. 3

Date: - / /

Q.1) Write a program to create a class and creating an object.

```
#include <iostream>

using namespace std;
```

```
// Define a class
```

```
class MyClass
```

```
{
```

```
public:
```

```
    void setValue(int val)
```

```
    {
```

```
        myVariable = val;
```

```
    }
```

```
    int getValue()
```

```
    {
```

```
        return myVariable;
```

```
    }
```

```
    int myVariable;
```

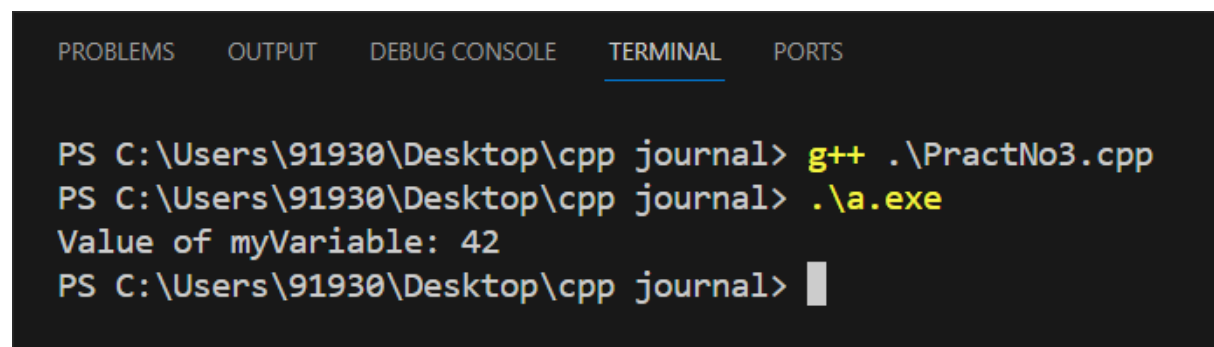
```
};
```

```
int main()
```



```
{  
  
    // Create an object of MyClass  
  
    MyClass myObject;  
  
  
    myObject.setValue(42);  
  
  
    cout << "Value of myVariable: " << myObject.getValue() << endl;  
  
  
    return 0;  
}
```

Output:



The screenshot shows a terminal window with a dark background and light-colored text. At the top, there are tabs labeled 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (which is selected and underlined), and 'PORTS'. The terminal content shows the following commands and output:

```
PS C:\Users\91930\Desktop\cpp journal> g++ .\PractNo3.cpp  
PS C:\Users\91930\Desktop\cpp journal> .\a.exe  
Value of myVariable: 42  
PS C:\Users\91930\Desktop\cpp journal> 
```

Practical No. 4

Date: - / /

Q.1) Illustrating different access specifiers.

```
// access modifier

#include <iostream>

using namespace std;

class Circle

{

private:

    double radius;

public:

    void compute_area(double r)

    {

        radius = r;

        double area = 3.14 * radius * radius;

        cout << "Radius is: " << radius << endl;

        cout << "Area is: " << area;

    }

};

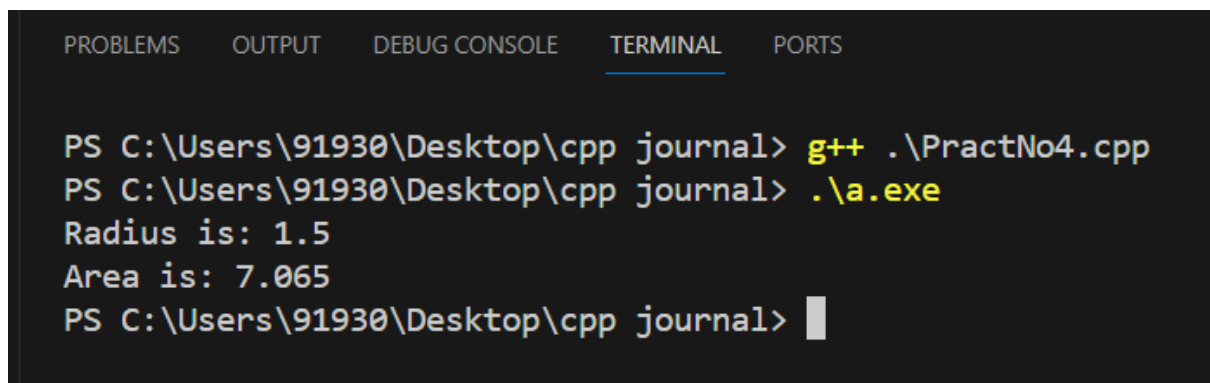
int main()

{

    Circle obj;
```

```
    obj.compute_area(1.5);  
  
    return 0;  
  
}
```

Output:



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  
  
PS C:\Users\91930\Desktop\cpp journal> g++ .\PractNo4.cpp  
PS C:\Users\91930\Desktop\cpp journal> .\a.exe  
Radius is: 1.5  
Area is: 7.065  
PS C:\Users\91930\Desktop\cpp journal> 
```

Practical No. 5

Date: - / /

Q.1) Write a OOP program to demonstrate Static data member.

```
#include <iostream>

using namespace std;

class Employee
{
    int id;

    static int count;

public:
    void setData(void)
    {
        cout << "Enter the id" << endl;

        cin >> id;

        count++;
    }

    void getData(void)
    {
        cout << "The id of this employee is " << id << " And this is employee number " <<
count << endl;
    }
};

int Employee::count; // Initialize the static data member
```

```
int main()
{
    Employee e1, e2, e3;

    e1.setData();

    e1.getData();


    e2.setData();

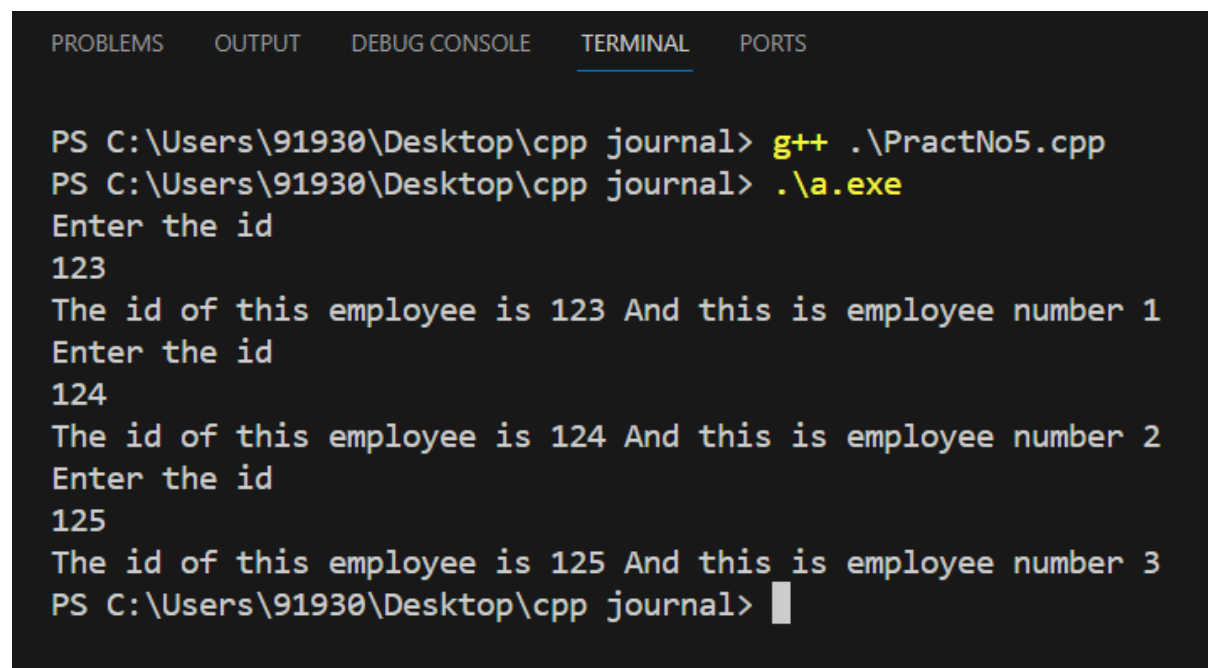
    e2.getData();


    e3.setData();

    e3.getData();

    return 0;
}
```

Output:



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\91930\Desktop\cpp journal> g++ .\PractNo5.cpp
PS C:\Users\91930\Desktop\cpp journal> .\a.exe
Enter the id
123
The id of this employee is 123 And this is employee number 1
Enter the id
124
The id of this employee is 124 And this is employee number 2
Enter the id
125
The id of this employee is 125 And this is employee number 3
PS C:\Users\91930\Desktop\cpp journal> █
```

Practical No. 6

Date: - / /

Q.1) Demonstrate arguments to the function.

```
#include <iostream>
```

```
using namespace std;
```

```
// call by Value
```

```
int add(int a, int b)
```

```
{
```

```
    return a + b;
```

```
}
```

```
// Here we modify the original value
```

```
void square(int &num)
```

```
{
```

```
    num *= num;
```

```
}
```

```
// Function to swap two integers using pass by reference
```

```
void swap(int &x, int &y)
```

```
{
```

```
    int temp = x;
```

```
    x = y;
```

```
    y = temp;
```

```
}
```

```
int main()

{
    // example 1

    int num1 = 5, num2 = 10;

    int sum = add(num1, num2);

    cout << "Sum of " << num1 << " and " << num2 << " is " << sum << endl

        << endl;


    // example 2

    int num = 7;

    cout << "Original value: " << num << endl;

    square(num);

    cout << "Square value: " << num << endl

        << endl;


    // example 3

    int a = 20, b = 30;

    cout << "Before swap: a = " << a << ", b = " << b << endl;

    swap(a, b);

    cout << "After swap: a = " << a << ", b = " << b << endl;


    return 0;

}
```

Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\91930\Desktop\cpp journal> g++ .\PractNo6.cpp
PS C:\Users\91930\Desktop\cpp journal> .\a.exe
Sum of 5 and 10 is 15

Original value: 7
Square value: 49

Before swap: a = 20, b = 30
After swap: a = 30, b = 20
PS C:\Users\91930\Desktop\cpp journal> 
```


Practical No. 7

Date: - / /

Q.1) Illustrating inline function.

```
#include <iostream>

using namespace std;

// Inline function

inline int square(int num)

{
    return num * num;
}

int main()

{
    int num = 5;

    // Example 1 - Calling inline function directly

    cout << "Square of " << num << " is " << square(num) << endl;

    // Example 2 - Using inline function in an expression

    int result = square(num) + square(3);

    cout << "Result: " << result << endl;

    return 0;
}
```

Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\91930\Desktop\cpp journal> g++ .\PractNo7.cpp
PS C:\Users\91930\Desktop\cpp journal> .\a.exe
Square of 5 is 25
Result: 34
PS C:\Users\91930\Desktop\cpp journal> █
```

Practical No. 8

Date: - / /

Q.1) Define member function – outside the class using Scope Resolution operator.

```
#include <iostream>

using namespace std;

class student
{
    string name;

    int rollno;

public:

    void getdata();

    void display();

};

void student::getdata()
{
    cout << "Enter the name : ";

    getline(cin, name);

    cout << "Enter ROLL Number : ";

    cin >> rollno;

}
```

```
void student::display()

{

    cout << "\nName is : " << name;

    cout << "\nRoll number is : " << rollno;

}
```

```
int main()

{

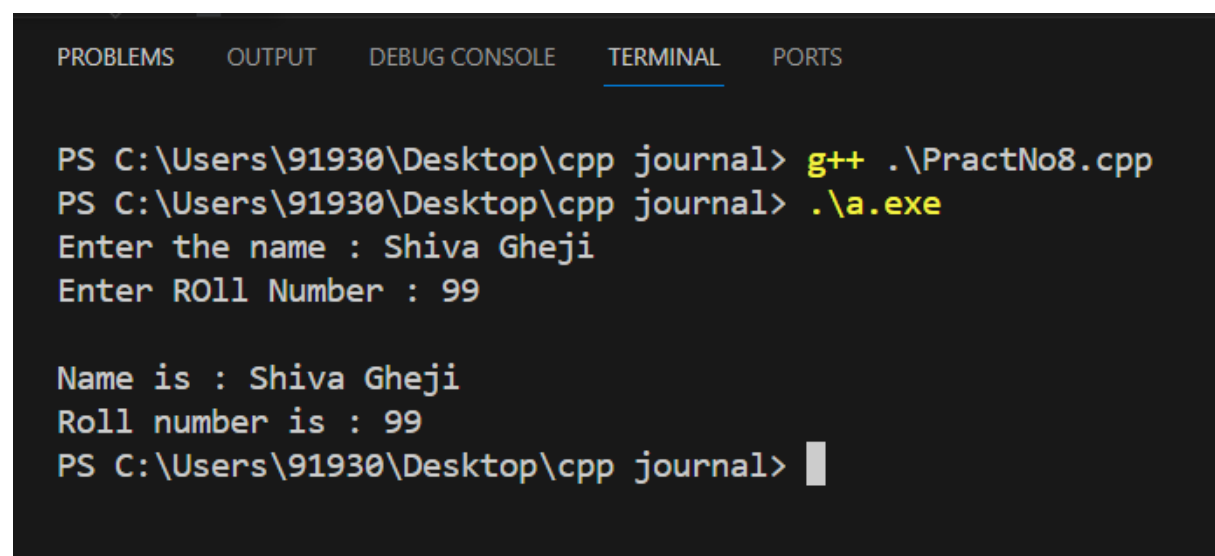
    student obj;

    obj.getdata();

    obj.display();

}
```

Output:

A screenshot of a terminal window with a dark background. At the top, there are tabs labeled 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (which is selected and underlined), and 'PORTS'. The terminal shows the following text:

```
PS C:\Users\91930\Desktop\cpp journal> g++ .\PractNo8.cpp
PS C:\Users\91930\Desktop\cpp journal> .\a.exe
Enter the name : Shiva Gheji
Enter Roll Number : 99

Name is : Shiva Gheji
Roll number is : 99
PS C:\Users\91930\Desktop\cpp journal> |
```

Practical No. 9

Date: - / /

Q.1) Illustrating Friend class and Friend function.

- **Friend function**

```
#include <iostream>

using namespace std;

class Distance
{
private:
    int meter;

    // friend function
    friend int addFive(Distance);

public:
    Distance() : meter(0) {}
};

// friend function definition
int addFive(Distance d)
{

    // accessing private members from the friend function
    d.meter += 5;
```

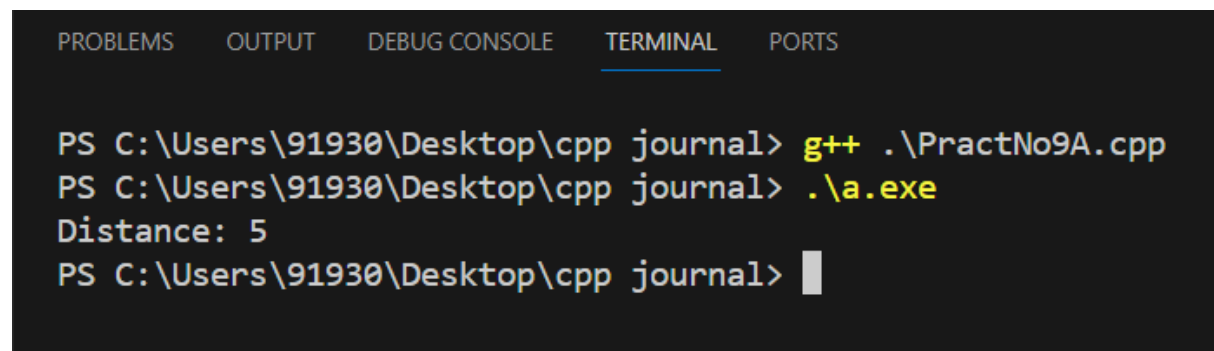
```
        return d.meter;
    }

    int main()
    {
        Distance D;

        cout << "Distance: " << addFive(D);

        return 0;
    }
```

Output:



The screenshot shows a terminal window with a dark background and light-colored text. At the top, there are five tabs: 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (which is selected and underlined), and 'PORTS'. Below the tabs, the terminal shows the following commands and output:

```
PS C:\Users\91930\Desktop\cpp journal> g++ .\PractNo9A.cpp
PS C:\Users\91930\Desktop\cpp journal> .\a.exe
Distance: 5
PS C:\Users\91930\Desktop\cpp journal> 
```

- **Friend Class**

```
#include <iostream>

using namespace std;

// forward declaration
class ClassB;

class ClassA
{
private:
    int numA;

    // friend class declaration
    friend class ClassB;

public:
    // constructor to initialize numA to 12
    ClassA() : numA(12) {}
};

class ClassB
{
private:
    int numB;

public:
    // constructor to initialize numB to 1
    ClassB() : numB(1) {}

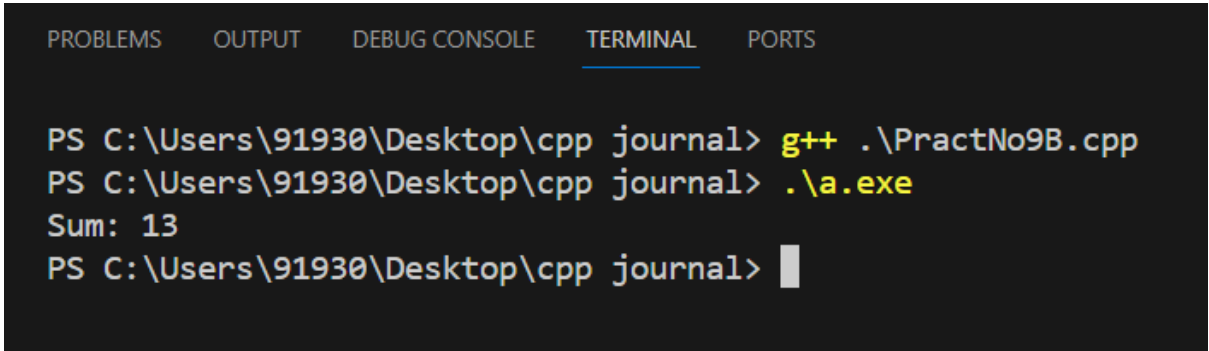
    // member function to add numA
    // from ClassA and numB from ClassB
```

```
int add()
{
    ClassA objectA;
    return objectA.numA + numB;
}

};

int main()
{
    ClassB objectB;
    cout << "Sum: " << objectB.add();
    return 0;
}
```

Output:



The screenshot shows a Visual Studio interface with the 'TERMINAL' tab selected. The terminal displays the following commands and output:

```
PS C:\Users\91930\Desktop\cpp journal> g++ .\PractNo9B.cpp
PS C:\Users\91930\Desktop\cpp journal> .\a.exe
Sum: 13
PS C:\Users\91930\Desktop\cpp journal> █
```


Practical No. 10

Date: - / /

Q.1) Create constructors – default, parameterized, copy.

```
#include <iostream>

using namespace std;

class MyClass
{
private:
    int data;

public:
    // Default constructor
    MyClass()
    {
        data = 0;
        cout << "Default constructor called. Data set to 0." << endl;
    }

    // Parameterized constructor
    MyClass(int value)
    {
        data = value;
        cout << "Parameterized constructor called. Data set to " << value << "." << endl;
    }
}
```

```
// Copy constructor

MyClass(const MyClass &obj)
{
    data = obj.data;

    cout << "Copy constructor called. Data copied from another object." << endl;
}

void display()
{
    cout << "Data: " << data << endl;
}

};

int main()
{
    cout << "Creating object using default constructor:" << endl;

    MyClass obj1; // Default constructor called

    cout << "\nCreating object using parameterized constructor:" << endl;

    MyClass obj2(100); // Parameterized constructor called

    cout << "\nCreating object using copy constructor:" << endl;

    MyClass obj3(obj2); // Copy constructor called

    // Displaying data of objects

    cout << "\nData in obj1:" << endl;

    obj1.display();
```

```
cout << "\nData in obj2:" << endl;

obj2.display();

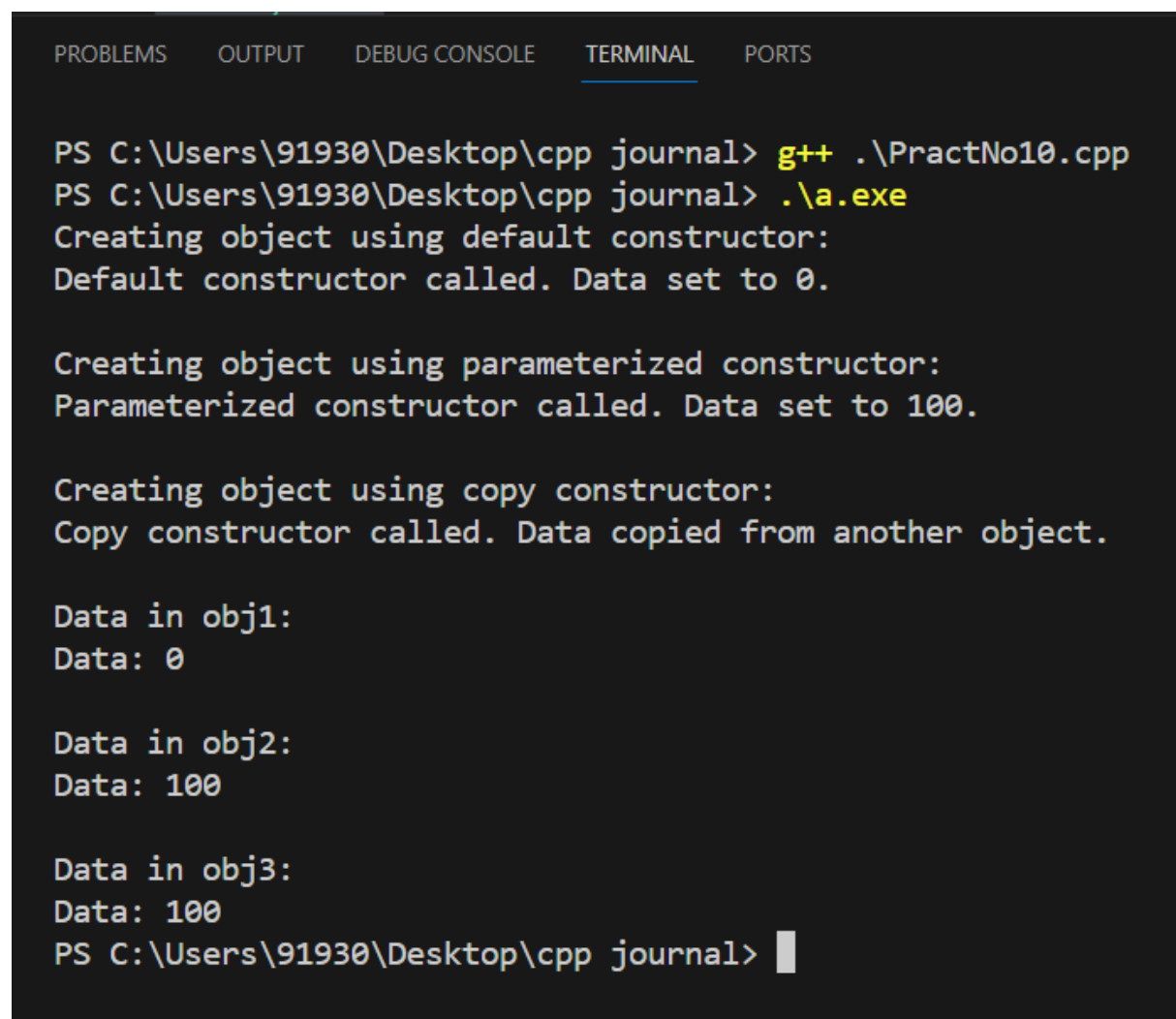
cout << "\nData in obj3:" << endl;

obj3.display();


return 0;

}
```

Output:

A screenshot of a terminal window with a dark background. At the top, there are tabs labeled 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (which is selected and underlined), and 'PORTS'. The terminal shows the execution of a C++ program. It starts with a command prompt 'PS C:\Users\91930\Desktop\cpp journal>' followed by 'g++ .\PractNo10.cpp' and then './a.exe'. The output consists of several lines: 'Creating object using default constructor:', 'Default constructor called. Data set to 0.', 'Creating object using parameterized constructor:', 'Parameterized constructor called. Data set to 100.', 'Creating object using copy constructor:', 'Copy constructor called. Data copied from another object.', followed by three blocks of output: 'Data in obj1: Data: 0', 'Data in obj2: Data: 100', and 'Data in obj3: Data: 100'. The prompt 'PS C:\Users\91930\Desktop\cpp journal>' is shown again at the bottom with a cursor.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\91930\Desktop\cpp journal> g++ .\PractNo10.cpp
PS C:\Users\91930\Desktop\cpp journal> .\a.exe
Creating object using default constructor:
Default constructor called. Data set to 0.

Creating object using parameterized constructor:
Parameterized constructor called. Data set to 100.

Creating object using copy constructor:
Copy constructor called. Data copied from another object.

Data in obj1:
Data: 0

Data in obj2:
Data: 100

Data in obj3:
Data: 100
PS C:\Users\91930\Desktop\cpp journal> █
```

Practical No. 11

Date: - / /

Q.1) Destructor.

```
#include <iostream>

using namespace std;

class Employee
{
public:
    Employee()
    {
        cout << "Constructor Invoked" << endl;
    }
    ~Employee()
    {
        cout << "Destructor Invoked" << endl;
    }
};

int main(void)
{
    Employee e1; // creating an object of Employee
    Employee e2; // creating an object of Employee
    return 0;
}
```

Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\91930\Desktop\cpp journal> g++ PractNo11.cpp
PS C:\Users\91930\Desktop\cpp journal> .\a.exe
Constructor Invoked
Constructor Invoked
Destructor Invoked
Destructor Invoked
PS C:\Users\91930\Desktop\cpp journal> █
```

Practical No. 12

Date: - / /

Q.1) Dynamic initialisation of object.

```
#include <iostream>

using namespace std;

class MyClass {

public:

    MyClass(int val) : value(val) {

        cout << "Constructor called with value: " << value << endl;

    }

    ~MyClass() {

        cout << "Destructor called for value: " << value << endl;

    }

    void display() {

        cout << "Value: " << value << endl;

    }

private:

    int value;

};

int main() {

    // Dynamic initialization of object

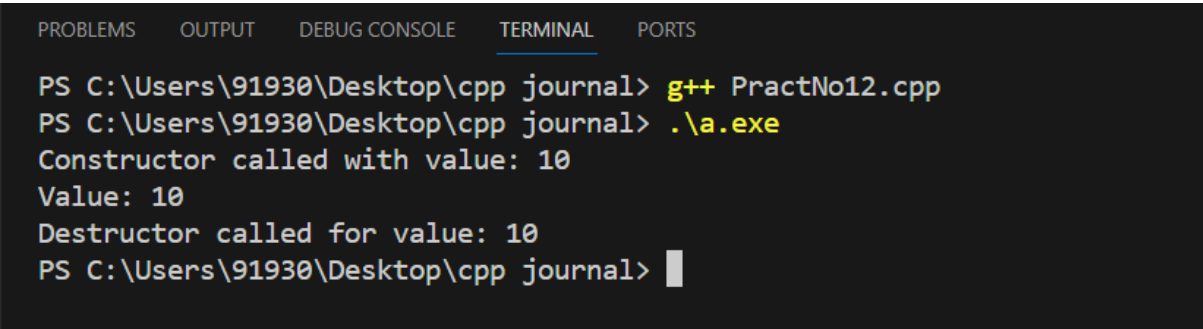
    MyClass *objPtr = new MyClass(10);

    // Accessing member function

    objPtr->display();
```

```
// Deallocating memory  
  
delete objPtr;  
  
return 0;  
  
}
```

Output:



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  
  
PS C:\Users\91930\Desktop\cpp journal> g++ PractNo12.cpp  
PS C:\Users\91930\Desktop\cpp journal> .\a.exe  
Constructor called with value: 10  
Value: 10  
Destructor called for value: 10  
PS C:\Users\91930\Desktop\cpp journal> █
```

Practical No. 13

Date: - / /

Q.1) Illustrating inheritance.

- **single inheritance.**

```
#include <iostream>

using namespace std;

class Account
{
public:
    float salary = 60000;
};

class Programmer : public Account
{
public:
    float bonus = 5000;
};

int main(void)
{
    Programmer p1;

    cout << "Salary: " << p1.salary << endl;

    cout << "Bonus: " << p1.bonus << endl;

    return 0;
}
```


Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  
PS C:\Users\91930\Desktop\cpp journal> g++ PractNo13.cpp  
PS C:\Users\91930\Desktop\cpp journal> .\a.exe  
Salary: 60000  
Bonus: 5000  
PS C:\Users\91930\Desktop\cpp journal> █
```

- **Multilevel inheritance.**

```
#include <iostream>

using namespace std;

class Animal
{
public:
    void eat()
    {
        cout << "Eating..." << endl;
    }
};

class Dog : public Animal
{
public:
    void bark()
    {
        cout << "Barking..." << endl;
    }
};

class BabyDog : public Dog
{
public:
```

```
void weep()

{
    cout << "Weeping...";
}

};

int main(void)
{
    BabyDog d1;

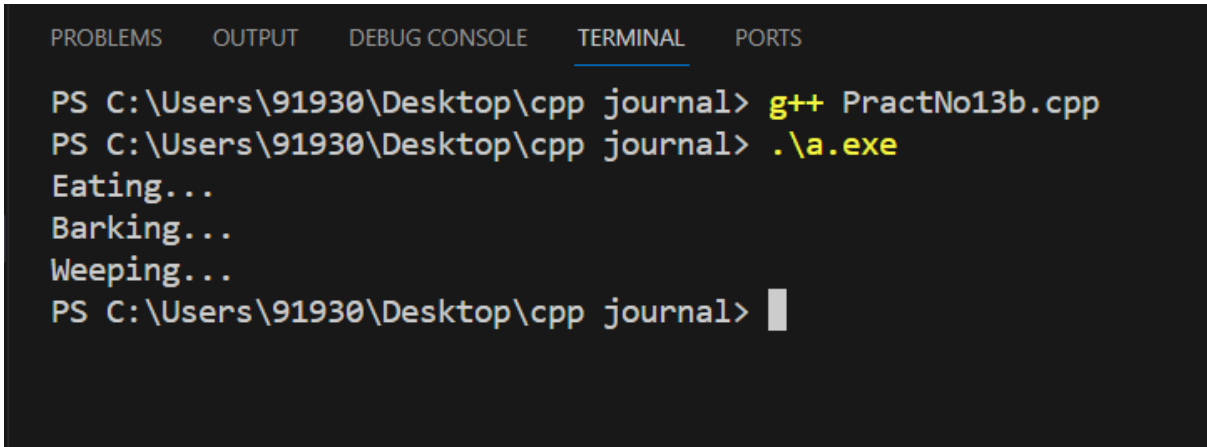
    d1.eat();

    d1.bark();

    d1.weep();

    return 0;
}
```

Output:



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\91930\Desktop\cpp journal> g++ PractNo13b.cpp
PS C:\Users\91930\Desktop\cpp journal> .\a.exe
Eating...
Barking...
Weeping...
PS C:\Users\91930\Desktop\cpp journal> █
```

Practical No. 14

Date: - / /

Q.1) Perform static and dynamic polymorphism.

```
#include <iostream>

using namespace std;

// Base class

class Animal {

public:

    virtual void sound() {

        cout << "Animal makes a sound" << endl;

    }

    void eat() {

        cout << "Animal eats food" << endl;

    }

};

// Derived class

class Dog : public Animal {

public:

    void sound() override {

        cout << "Dog barks" << endl;

    }

    void eat() {

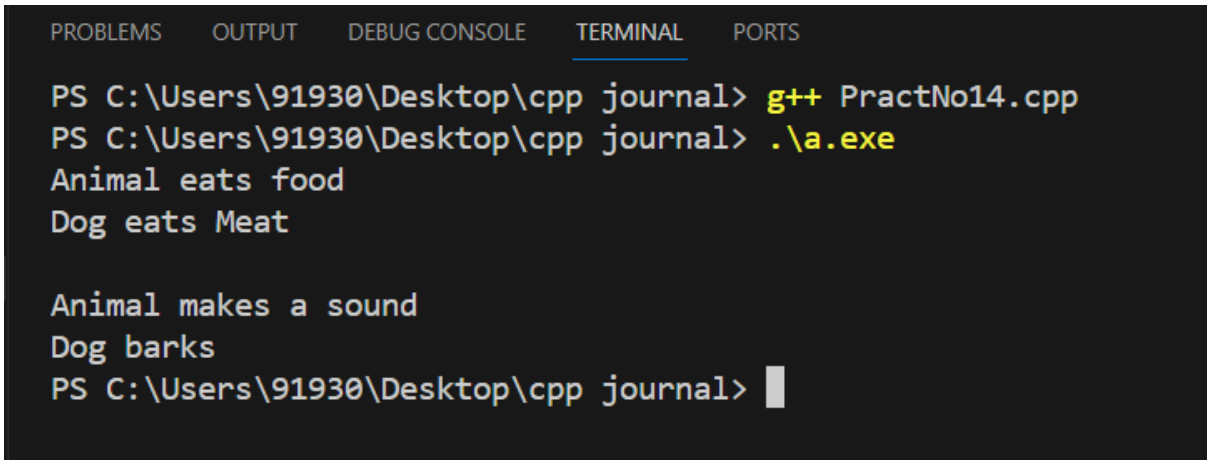
        cout << "Dog eats Meat" << endl;

    }

}
```

```
};  
  
int main() {  
    // Static polymorphism  
  
    Animal animal;  
  
    Dog dog;  
  
    animal.eat();  
  
    dog.eat();  
  
    std::cout << std::endl;  
  
    // Dynamic polymorphism  
  
    Animal* ptr = &animal;  
  
    ptr->sound();  
  
    ptr = &dog;  
  
    ptr->sound();  
  
    return 0;  
}
```

Output:



The screenshot shows a terminal window with tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The TERMINAL tab is active. The command prompt shows the user running 'g++ PractNo14.cpp' and then './a.exe'. The output of the program is displayed below the command prompt, showing the results of static and dynamic polymorphism.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  
PS C:\Users\91930\Desktop\cpp journal> g++ PractNo14.cpp  
PS C:\Users\91930\Desktop\cpp journal> .\a.exe  
Animal eats food  
Dog eats Meat  
  
Animal makes a sound  
Dog barks  
PS C:\Users\91930\Desktop\cpp journal> 
```

Practical No. 15

Date: - / /

Q.1) Demonstrate virtual and pure virtual function.

```
#include <iostream>

using namespace std;

class Shape
{
public:
    virtual double calculateArea() = 0; // Pure virtual function
    virtual ~Shape() {}
};

class Circle : public Shape
{
private:
    double radius;
public:
    Circle(double r) : radius(r) {}

    double calculateArea() override
    {
        return 3.14 * radius * radius;
    }
};

class Rectangle : public Shape
{
```

private:

double length;

double width;

public:

Rectangle(double l, double w) : length(l), width(w) {}

double calculateArea() override

{

return length * width;

}

};

int main()

{

Circle circle(5);

Rectangle rectangle(4, 6);

// Using virtual function to calculate area

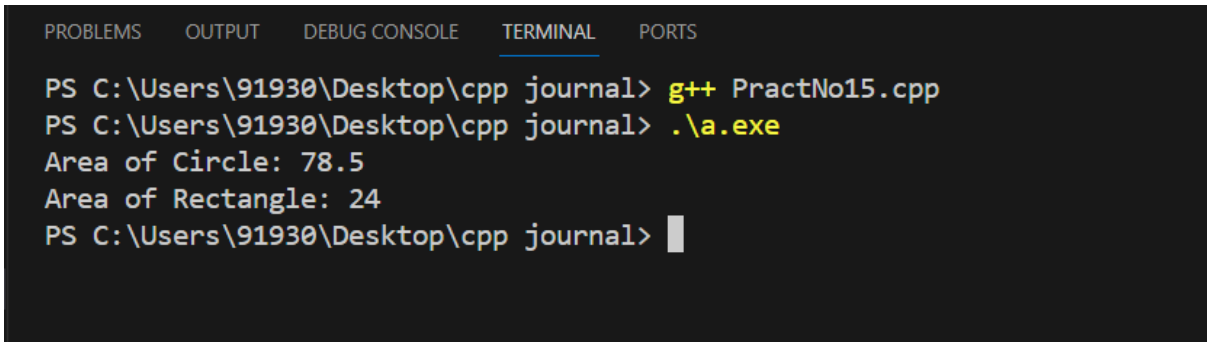
cout << "Area of Circle: " << circle.calculateArea() << endl;

cout << "Area of Rectangle: " << rectangle.calculateArea() << endl;

return 0;

}

Output:



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS C:\Users\91930\Desktop\cpp journal> g++ PractNo15.cpp
PS C:\Users\91930\Desktop\cpp journal> .\a.exe
Area of Circle: 78.5
Area of Rectangle: 24
PS C:\Users\91930\Desktop\cpp journal>
```