Shri Shahu Shikshan Sanstha’s

Shri Shahaji Chhatrapati Mahavidyalaya, Kolhapur

JOURNAL 2023-2024

This is certifying that he/she has successfully completed the journal of
BCA II Sem IV

Name of Student	:	Samrat Krishnat Desai
University	:	Shivaji University
Class	:	BCA II (SEM III)
Roll NO	:	13
Exam Seat No	:	
Subject	:	C++
Date	:	

Internal Examiner External Examiner BCA Coordinator

Name : Samrat Krishnat Desai Class : BCA II (SEM IV)
Roll No : 13 PRN :

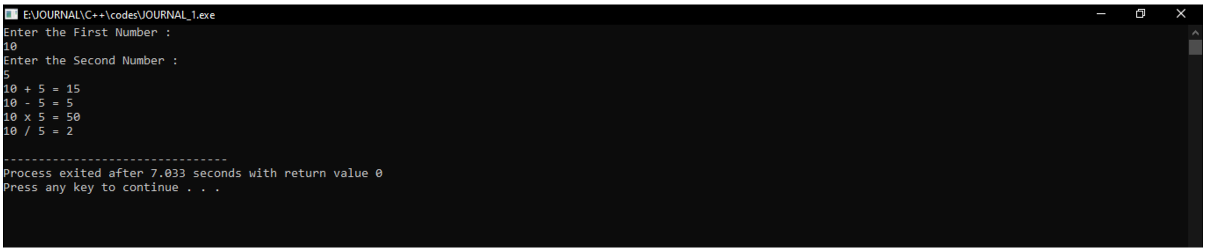
INDEX

13	Illustrating Inheritance – single, multiple and multilevel		
14	Perform static and dynamic polymorphism		
15	Demonstrate virtual & pure virtual function.		

1. Write a simple program (without Class) to use of operators in C++

```
#include<iostream>
using namespace std;
int main()
{
    int no1, no2;
    cout<<"Enter the First Number :"<<endl;
    cin>>no1;
    cout<<"Enter the Second Number :"<<endl;
    cin>>no2;
    // ADDITION
    cout<<no1<<" + "<<no2<<" = "<<no1+no2<<endl;
    // SUBTRACTION
    cout<<no1<<" - "<<no2<<" = "<<no1-no2<<endl;
    // MULTIPLICATION
    cout<<no1<<" x "<<no2<<" = "<<no1*no2<<endl;
    // DIVISION
    cout<<no1<<" / "<<no2<<" = "<<no1/no2<<endl;
    return 0;
}
```

OUTPUT:

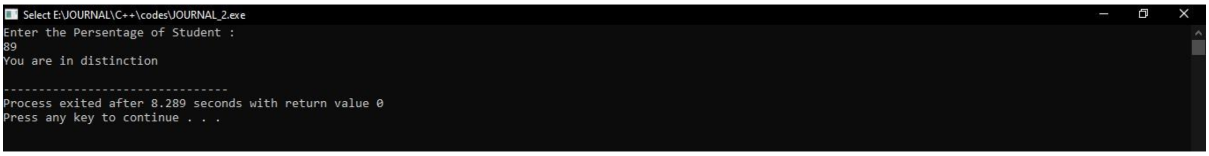


```
Enter the First Number :
10
Enter the Second Number :
5
10 + 5 = 15
10 - 5 = 5
10 x 5 = 50
10 / 5 = 2

--------------------------------
Process exited after 7.033 seconds with return value 0
Press any key to continue . . .
```

2. Illustrating Control Structures.

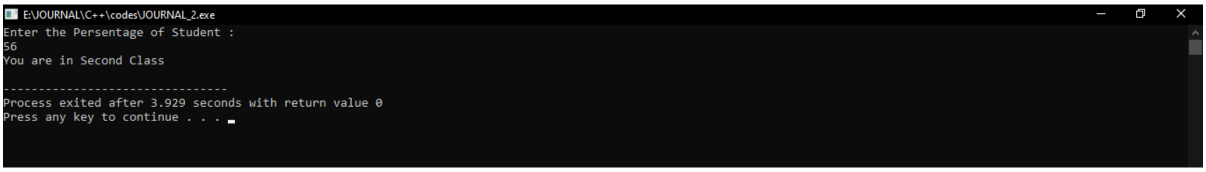
```
#include<iostream>
using namespace std;
int main()
{
   double per;
   cout<<"Enter the Persentage of Student :"<<endl;
   cin>>per;
   if (per>=75 && per< 100)
   {
      cout<<"You are in distinction"<<endl;
   }
   else if (per<75 && per >= 60)
   {
      cout<<"You are in First Class"<<endl;
   }
   else if (per<60 && per >= 50)
   {
      cout<<"You are in Second Class"<<endl;
   }
   else if (per<50 && per >= 35 && per >0)
   {
      cout<<"You are in Pass Class"<<endl;
   }
   else if (per<0 || per > 100)
   {
      cout<<"Enter right Persentage "<<endl;
   }
   else
   {
        cout<<"You are Fail "<<endl;
    }
    return 0;
}
```

OUTPUT:



Select E:\JOURNAL\C++\codes\JOURNAL_2.exe

```
Enter the Persentage of Student :
89
You are in distinction

--------------------------------
Process exited after 8.289 seconds with return value 0
Press any key to continue . . .
```



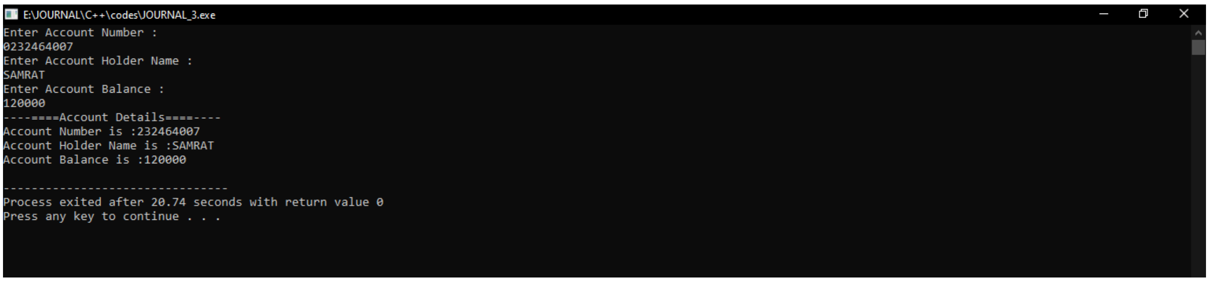
E:\JOURNAL\C++\codes\JOURNAL_2.exe

```
Enter the Persentage of Student :
56
You are in Second Class

--------------------------------
Process exited after 3.929 seconds with return value 0
Press any key to continue . . .
```

3. Write a program to create a class and creating an object

```
#include<iostream>
using namespace std;
class bank{
    int ac_no;
    string name;
    double balance;
    public :
    void input()
    {
        cout<<"Enter Account Number :"<<endl;
        cin>>ac_no;
        cout<<"Enter Account Holder Name :"<<endl;
        cin>>name;
        cout<<"Enter Account Balance :"<<endl;
        cin>>balance;
    }
    void display()
    {
        cout<<"----====Account Details==== --- "<<endl;
        cout<<"Account Number is :"<<ac_no<<endl;
        cout<<"Account Holder Name is :"<<name<<endl;
        cout<<"Account Balance is :"<<balance<<endl;
    }
};
int main()
{
    bank b1;
    b1.input();
    b1.display();
    return 0;
}
```

OUTPUT:



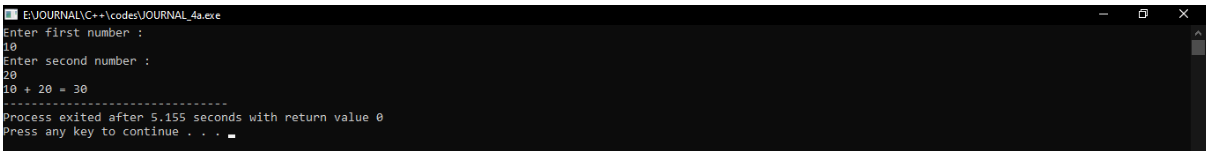
```
Enter Account Number :
0232464007
Enter Account Holder Name :
SAMRAT
Enter Account Balance :
120000
----====Account Details====----
Account Number is :232464007
Account Holder Name is :SAMRAT
Account Balance is :120000

--------------------------------
Process exited after 20.74 seconds with return value 0
Press any key to continue . . .
```

4. Illustrating different Access Specifiers

PRIVATE:

```
#include<iostream>
using namespace std;
class addition{
private:
    int a, b;
    public :
    void add(){
        cout<<"Enter first number :"<<endl;
        cin>>a;
        cout<<"Enter second number :"<<endl;
        cin>>b;
    }
    void display(){
        cout<<a<<" + "<<b<<" = "<<a+b;
    }
};
int main()
{
    addition a;
    a.add();
    a.display();
    return 0;
}
```

OUTPUT:

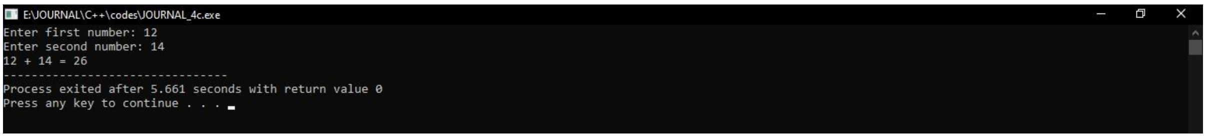


```
Enter first number :
10
Enter second number :
20
10 + 20 = 30
--------------------------------
Process exited after 5.155 seconds with return value 0
Press any key to continue . . .
```

PUBLIC:

```
#include<iostream>
using namespace std;
class Addition {
public:
    int a, b;
    void add(){
        cout << "Enter first number: ";
        cin >> a;
        cout << "Enter second number: ";
        cin >> b;
    }
    void display(){
        cout << a << " + " << b << " = " << a + b;
    }
};
int main() {
    Addition a;
    a.add();
    a.display();
    return 0;
}
```

OUTPUT:

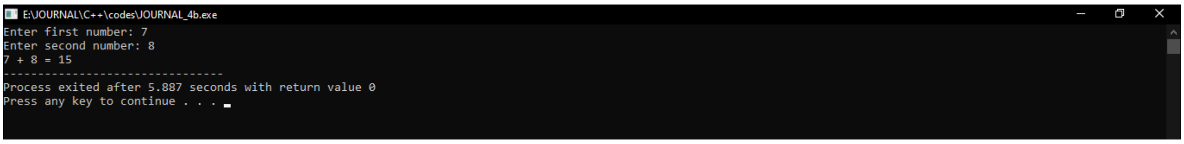


```
Enter first number: 12
Enter second number: 14
12 + 14 = 26
--------------------------------
Process exited after 5.661 seconds with return value 0
Press any key to continue . . .
```

PUBLIC:

```
#include<iostream>
using namespace std;
class Addition {
protected:
    int a, b;
public:
    void add(){
        cout << "Enter first number: ";
        cin >> a;
        cout << "Enter second number: ";
        cin >> b;
    }
    void display(){
        cout << a << " + " << b << " = " << a + b;
    }
};
int main() {
    Addition a;
    a.add();
    a.display();
    return 0;
}
```

OUTPUT:

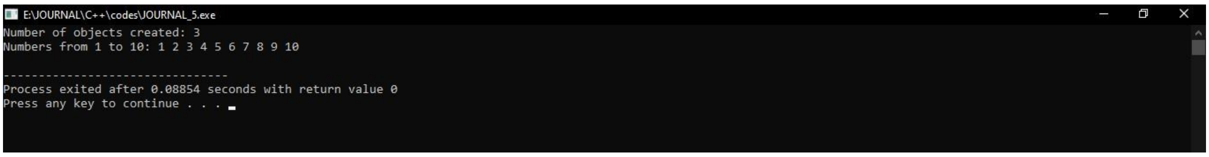


```
Enter first number: 7
Enter second number: 8
7 + 8 = 15
--------------------------------
Process exited after 5.887 seconds with return value 0
Press any key to continue . . .
```

5. Write a OOP program to demonstrate static data member

```
#include <iostream>
using namespace std;
class Counter {
private:
    static int count;
public:
    Counter() {
        count++;
    }
    static void printNumbers() {
        for (int i = 1; i <= 10; ++i) {
            cout << i << " ";
        }
        cout << endl;
    }
    static int getCount() {
        return count;
    }
};
// Initializing static data member
int Counter::count = 0;
int main() {
    // Creating objects of Counter
    Counter obj1;
    Counter obj2;
    Counter obj3;
    // Displaying the count of objects created
    cout << "Number of objects created: " << Counter::getCount() << endl;
    // Printing numbers 1 to 10
    cout << "Numbers from 1 to 10: ";
    Counter::printNumbers();
    return 0;
}
```

OUTPUT:

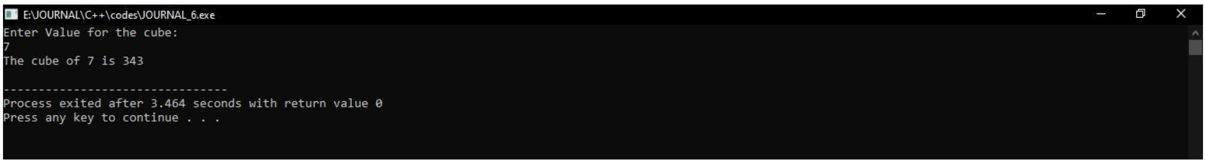


```
Number of objects created: 3
Numbers from 1 to 10: 1 2 3 4 5 6 7 8 9 10

--------------------------------
Process exited after 0.08854 seconds with return value 0
Press any key to continue . . .
```

6. Demonstrate arguments to the function.

```
#include<iostream>
using namespace std;
int cube(int a){
    return a*a*a;
}
int main()
{
    int b,c;
    cout<<"Enter Value for the cube:"<<endl;
    cin>>b;
    c = cube(b);
    cout<<"The cube of "<<b<<" is "<<c<<endl;
    return 0;
}
```

OUTPUT:

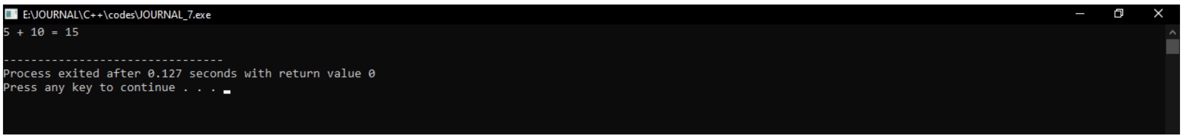


```
Enter Value for the cube:
7
The cube of 7 is 343

--------------------------------
Process exited after 3.464 seconds with return value 0
Press any key to continue . . .
```

7. Illustrating inline function.

```
#include <iostream>
using namespace std;
inline int add(int a, int b) {
    return a + b;
}
int main() {
    int num1 = 5, num2 = 10;
    int result = add(num1, num2);
    cout <<num1 << " + " << num2 << " = " << result << endl;
    return 0;
}
```

OUTPUT:

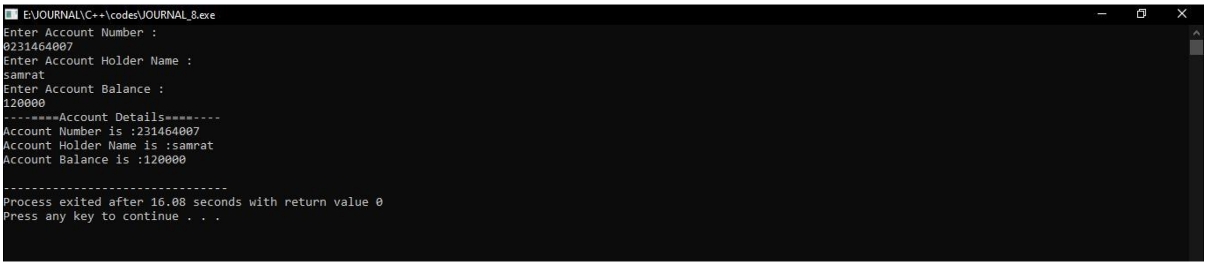


```
5 + 10 = 15

--------------------------------
Process exited after 0.127 seconds with return value 0
Press any key to continue . . .
```

8. Define Member function-outside the class using Scope Resolution Operator

```
#include<iostream>
using namespace std;
class bank{
    int ac_no;
    string name;
    double balance;
    public :
    void input();
    void display();
};
    void bank :: input()
    {
        cout<<"Enter Account Number :"<<endl;
        cin>>ac_no;
        cout<<"Enter Account Holder Name :"<<endl;
        cin>>name;
        cout<<"Enter Account Balance :"<<endl;
        cin>>balance;
    }
    void bank :: display()
    {
        cout<<"----====Account Details==== --- "<<endl;
        cout<<"Account Number is :"<<ac_no<<endl;
        cout<<"Account Holder Name is :"<<name<<endl;
        cout<<"Account Balance is :"<<balance<<endl;
    }
int main()
{
    bank b1;
    b1.input();
    b1.display();
    return 0;
}
```

OUTPUT:

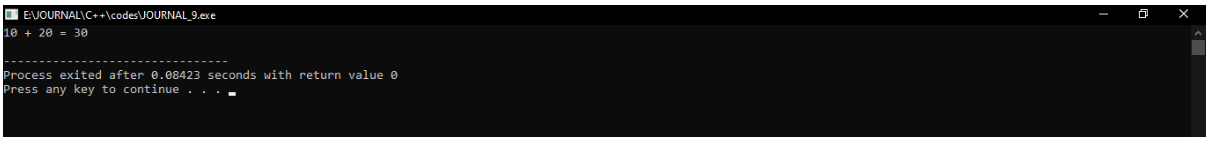


```
E:\JOURNAL\C++\codes\JOURNAL_8.exe
Enter Account Number :
0231464007
Enter Account Holder Name :
samrat
Enter Account Balance :
120000
----====Account Details====----
Account Number is :231464007
Account Holder Name is :samrat
Account Balance is :120000

--------------------------------
Process exited after 16.08 seconds with return value 0
Press any key to continue . . .
```

9. Illustrating friend class and friend function.

```
#include  <iostream>
using namespace std;
class Adder {
public:
    int num1;
public:
    Adder(int n) : num1(n) {}
    // Declare friend function
    friend int add(const Adder& obj, int num2);
};
// Definition of friend function
int add(const Adder& obj, int num2) {
    return obj.num1 + num2;
}
int main() {
    Adder obj(10);
    int num2 = 20;
    // Adding num1 and num2 using friend function
    int result = add(obj, num2);
    cout <<obj.num1 << " + " << num2 << " = " << result << endl;
    return 0;
}
```

OUTPUT:



```
10 + 20 = 30

--------------------------------
Process exited after 0.08423 seconds with return value 0
Press any key to continue . . .
```

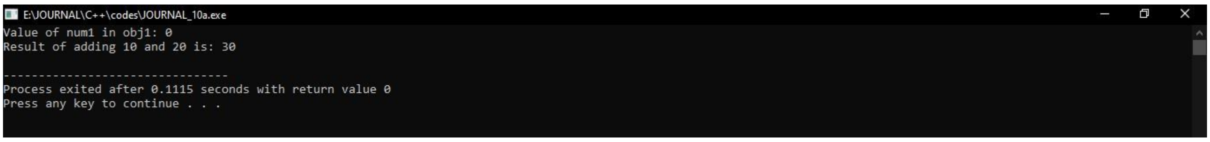
10. Create constructors – default, parameterized, copy.

DEFAULT CONSTRUCTER:

```
#include  <iostream>
using namespace std;
class Adder {
public:
    int num1;
public:
    // Default constructor
    Adder() : num1(0) {}
    // Parameterized constructor
    Adder(int n) : num1(n) {}
    // Declare friend function
    friend int add(const Adder& obj, int num2);
};
// Definition of friend function
int add(const Adder& obj, int num2) {
    return obj.num1 + num2;
}
int main() {
    // Creating an Adder object using default constructor
    Adder obj1;
    cout << "Value of num1 in obj1: " << obj1.num1 << endl;
    // Creating an Adder object using parameterized constructor
    Adder obj2(10);
    int num2 = 20;
    // Adding num1 and num2 using friend function
    int result = add(obj2, num2);
    cout << "Result of adding " << obj2.num1 << " and " << num2 << " is: " << result <<
endl;
```

```
    return 0;
}
```

OUTPUT:

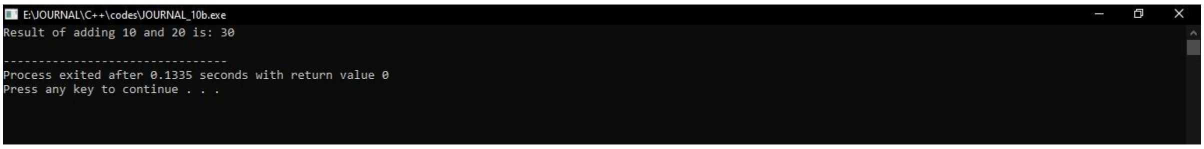


```
Value of num1 in obj1: 0
Result of adding 10 and 20 is: 30

--------------------------------
Process exited after 0.1115 seconds with return value 0
Press any key to continue . . .
```

PARAMETERIZED CONSTRUCTER:

```
#include  <iostream>
using namespace std;
class Adder {
public:
    int num1;
public:
    // Parameterized constructor
    Adder(int n) : num1(n) {}
    // Declare friend function
    friend int add(const Adder& obj, int num2);
};
// Definition of friend function
int add(const Adder& obj, int num2) {
    return obj.num1 + num2;
}
int main() {
    // Creating an Adder object using parameterized constructor
    Adder obj(10);
    int num2 = 20;
    // Adding num1 and num2 using friend function
    int result = add(obj, num2);
    cout << "Result of adding " << obj.num1 << " and " << num2 << " is: " << result << endl;
    return 0;
}
```

OUTPUT:

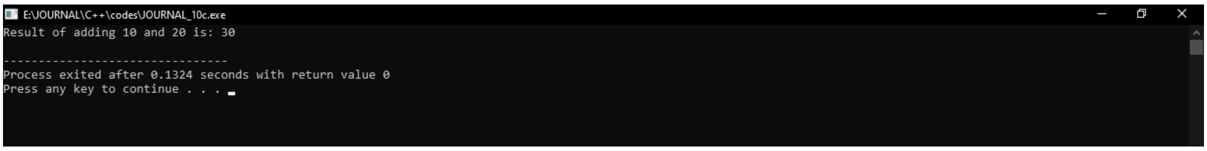


```
Result of adding 10 and 20 is: 30

--------------------------------
Process exited after 0.1335 seconds with return value 0
Press any key to continue . . .
```

COPY CONSTRUCTER:

```
#include  <iostream>
using namespace std;
class Adder {
public:
    int num1;
public:
    // Parameterized constructor
    Adder(int n) : num1(n) {}
    // Copy constructor
    Adder(const Adder& other) : num1(other.num1) {}
    // Declare friend function
    friend int add(const Adder& obj, int num2);
};
// Definition of friend function
int add(const Adder& obj, int num2) {
    return obj.num1 + num2;
}
int main() {
    // Creating an Adder object using parameterized constructor
    Adder obj1(10);
    // Creating another Adder object using copy constructor
    Adder obj2(obj1);
    // Adding num1 and num2 using friend function
    int num2 = 20;
    int result = add(obj2, num2);
    cout << "Result of adding " << obj2.num1 << " and " << num2 << " is: " << result << 
endl;
    return 0;
}
```

OUTPUT:



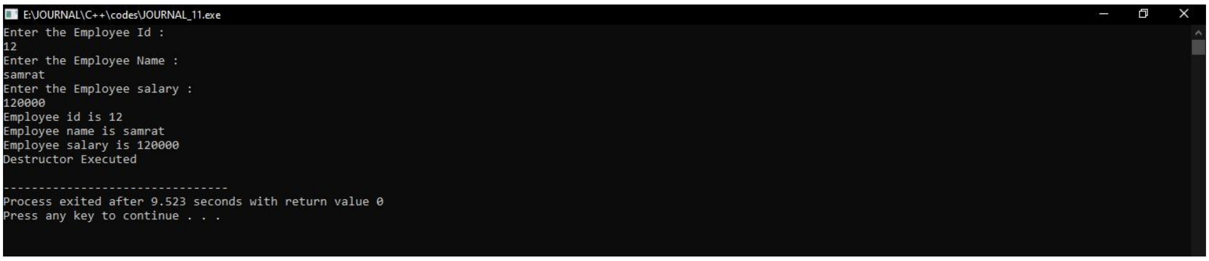
```
Result of adding 10 and 20 is: 30

--------------------------------
Process exited after 0.1324 seconds with return value 0
Press any key to continue . . .
```

11. Destructor

```
#include<iostream>
using namespace std;
class employee{
   int id;
   string name;
   double salary;
   public:
   employee(){
      cout<<"Enter the Employee Id :"<<endl;
      cin>>id;
      cout<<"Enter the Employee Name :"<<endl;
      cin>>name;
      cout<<"Enter the Employee salary :"<<endl;
      cin>>salary;
   }
   void display()
{
   cout<<"Employee id is "<<id<<endl;
   cout<<"Employee name is "<<name<<endl;
   cout<<"Employee salary is "<<salary<<endl;
}
   ~employee(){
      cout<<"Destructor Executed"<<endl;
   }
};
int main()
{
   employee e;
   e.display();
   return 0;
```

}

OUTPUT:

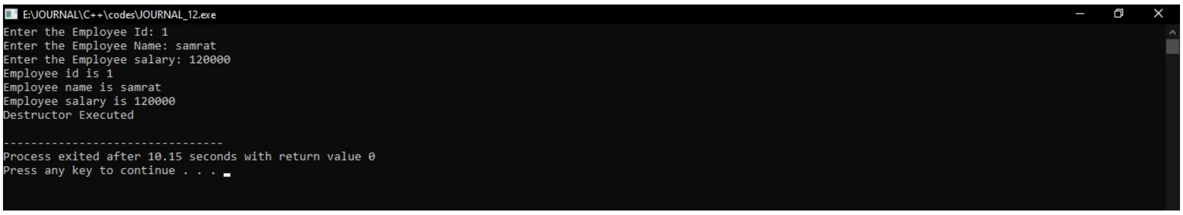


```
Enter the Employee Id :
12
Enter the Employee Name :
samrat
Enter the Employee salary :
120000
Employee id is 12
Employee name is samrat
Employee salary is 120000
Destructor Executed

--------------------------------
Process exited after 9.523 seconds with return value 0
Press any key to continue . . .
```

12. Dynamic Initialization of Object.

```
#include<iostream>
using namespace std;
class Employee {
private:
    int id;
    string name;
    double salary;
public:
    Employee() {
        cout << "Enter the Employee Id: ";
        cin >> id;
        cout << "Enter the Employee Name: ";
        cin >> name;
        cout << "Enter the Employee salary: ";
        cin >> salary;
    }
    void display() {
        cout << "Employee id is " << id << endl;
        cout << "Employee name is " << name << endl;
        cout << "Employee salary is " << salary << endl;
    }
    ~Employee() {
        cout << "Destructor Executed" << endl;
    }
};
int main() {
    Employee *e = new Employee();
    e->display();
    delete e;
    return 0;

}
```

OUTPUT:



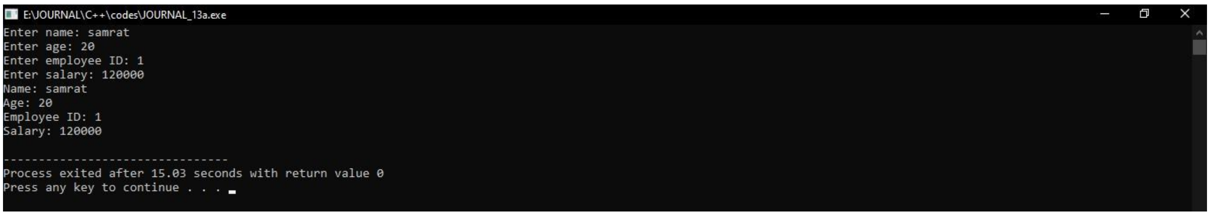
```
Enter the Employee Id: 1
Enter the Employee Name: samrat
Enter the Employee salary: 120000
Employee id is 1
Employee name is samrat
Employee salary is 120000
Destructor Executed

--------------------------------
Process exited after 10.15 seconds with return value 0
Press any key to continue . . .
```

13. Illustrating Inheritance – single, multiple and multilevel

SINGLE INHERITANCE:

```
#include<iostream>
using namespace std;
// Base class
class Person {
protected:
    string name;
    int age;
public:
    Person() {
        cout << "Enter name: ";
        cin >> name;
        cout << "Enter age: ";
        cin >> age;
    }
    void displayInfo() {
        cout << "Name: " << name << endl;
        cout << "Age: " << age << endl;
    }
};
// Derived class
class Employee : public Person {
private:
    int employeeId;
    double salary;
public:
    Employee() {
        cout << "Enter employee ID: ";
        cin >> employeeId;
        cout << "Enter salary: ";
        cin >> salary;
    }
    void displayEmployee() {
        Person::displayInfo();
        cout << "Employee ID: " << employeeId << endl;
        cout << "Salary: " << salary << endl;
    }
};
int main() {
    // Creating object of derived class
    Employee emp;
    // Accessing member functions of derived class
    emp.displayEmployee();
    return 0;
}
```

OUTPUT:



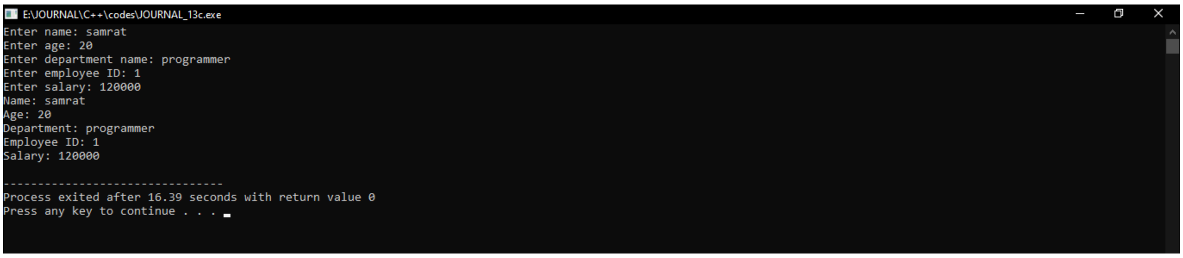
E:\JOURNAL\C++\codes\JOURNAL_13a.exe

```
Enter name: samrat
Enter age: 20
Enter employee ID: 1
Enter salary: 120000
Name: samrat
Age: 20
Employee ID: 1
Salary: 120000

--------------------------------
Process exited after 15.03 seconds with return value 0
Press any key to continue . . .
```

MULTIPLE INHERITANCE:

```
#include<iostream>
using namespace std;
// Base class
class Person {
protected:
    string name;
    int age;
public:
    Person() {
        cout << "Enter name: ";
        cin >> name;
        cout << "Enter age: ";
        cin >> age;
    }
    void displayInfo() {
        cout << "Name: " << name << endl;
        cout << "Age: " << age << endl;
    }
};
// Intermediate class inheriting from Person
class Department : public Person {
protected:
    string departmentName;
public:
    Department() {
        cout << "Enter department name: ";
        cin >> departmentName;
    }
    void displayDepartment() {
        cout << "Department: " << departmentName << endl;
    }
};
// Derived class inheriting from Department
class Employee : public Department {
private:
    int employeeId;
    double salary;
public:
    Employee() {
        cout << "Enter employee ID: ";
        cin >> employeeId;
        cout << "Enter salary: ";
        cin >> salary;
    }
    void displayEmployee() {
        displayInfo(); // Accessing base class method
        displayDepartment(); // Accessing intermediate class method
        cout << "Employee ID: " << employeeId << endl;
        cout << "Salary: " << salary << endl;
    }
};
int main() {
    // Creating object of derived class
    Employee emp;
    // Accessing member functions of derived class
    emp.displayEmployee();
    return 0;
}
```

OUTPUT:

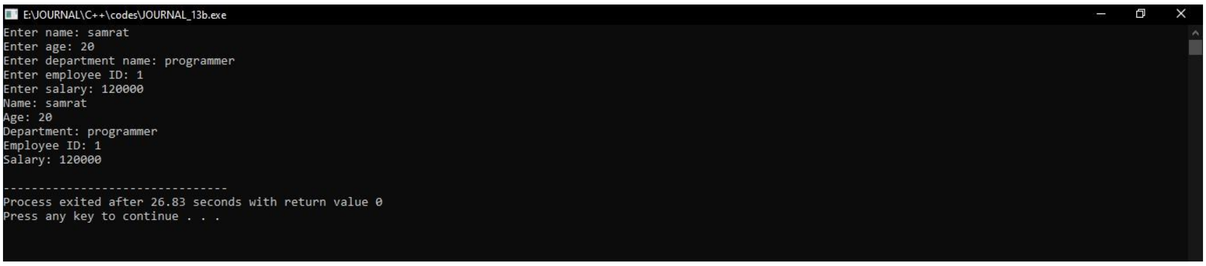


```
Enter name: samrat
Enter age: 20
Enter department name: programmer
Enter employee ID: 1
Enter salary: 120000
Name: samrat
Age: 20
Department: programmer
Employee ID: 1
Salary: 120000

--------------------------------
Process exited after 16.39 seconds with return value 0
Press any key to continue . . .
```

MULTILEVEL INHERITANCE:

```
#include<iostream>
using namespace std;
// Base class
class Person {
protected:
    string name;
    int age;
public:
    Person() {
        cout << "Enter name: ";
        cin >> name;
        cout << "Enter age: ";
        cin >> age;
    }
    void displayInfo() {
        cout << "Name: " << name << endl;
        cout << "Age: " << age << endl;
    }
};
// Another base class
class Department {
protected:
    string departmentName;
public:
    Department() {
        cout << "Enter department name: ";
        cin >> departmentName;
    }
    void displayDepartment() {
        cout << "Department: " << departmentName << endl;
    }
};
// Derived class inheriting from both Person and Department
class Employee : public Person, public Department {
private:
    int employeeId;
    double salary;
public:
    Employee() {
        cout << "Enter employee ID: ";
        cin >> employeeId;
        cout << "Enter salary: ";
        cin >> salary;
    }
    void displayEmployee() {
        Person::displayInfo();
        Department::displayDepartment();
        cout << "Employee ID: " << employeeId << endl;
        cout << "Salary: " << salary << endl;
    }
};
int main() {
    // Creating object of derived class
    Employee emp;
    // Accessing member functions of derived class
    emp.displayEmployee();
    return 0;
}
```

OUTPUT:



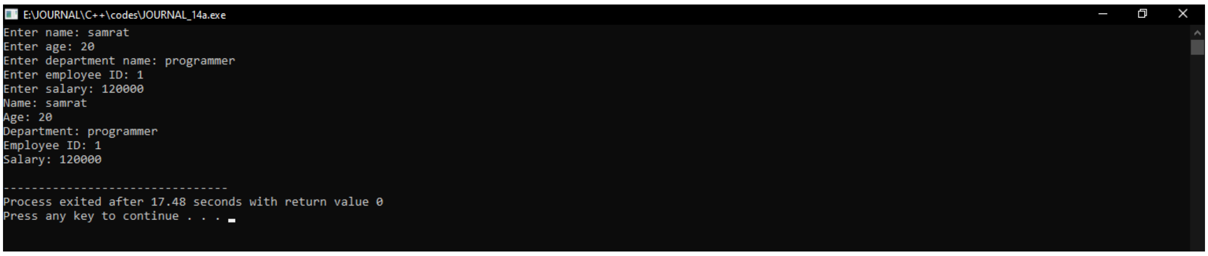
```
Enter name: samrat
Enter age: 20
Enter department name: programmer
Enter employee ID: 1
Enter salary: 120000
Name: samrat
Age: 20
Department: programmer
Employee ID: 1
Salary: 120000

--------------------------------
Process exited after 26.83 seconds with return value 0
Press any key to continue . . .
```

14. Perform static and dynamic polymorphism

STATIC POLYMORPHISM:

```
#include<iostream>
using namespace std;
// Base class
class Person {
protected:
    string name;
    int age;
public:
    Person() {
        cout << "Enter name: ";
        cin >> name;
        cout << "Enter age: ";
        cin >> age;
    }
    void displayInfo() {
        cout << "Name: " << name << endl;
        cout << "Age: " << age << endl;
    }
};
// Derived class inheriting from Person
class Department : public Person {
protected:
    string departmentName;
public:
    Department() {
        cout << "Enter department name: ";
        cin >> departmentName;
    }
    void displayInfo() { // Overridden function
        Person::displayInfo(); // Call base class function
        cout << "Department: " << departmentName << endl;
    }
};
// Derived class inheriting from Department
class Employee : public Department {
private:
    int employeeId;
    double salary;
public:
    Employee() {
        cout << "Enter employee ID: ";
        cin >> employeeId;
        cout << "Enter salary: ";
        cin >> salary;
    }
    void displayInfo() { // Overridden function
        Department::displayInfo(); // Call intermediate class function
        cout << "Employee ID: " << employeeId << endl;
        cout << "Salary: " << salary << endl;
    }
};
int main() {
    // Creating object of derived class
    Employee emp;
    // Accessing member functions of derived class
    emp.displayInfo();
    return 0;
}
```

OUTPUT:



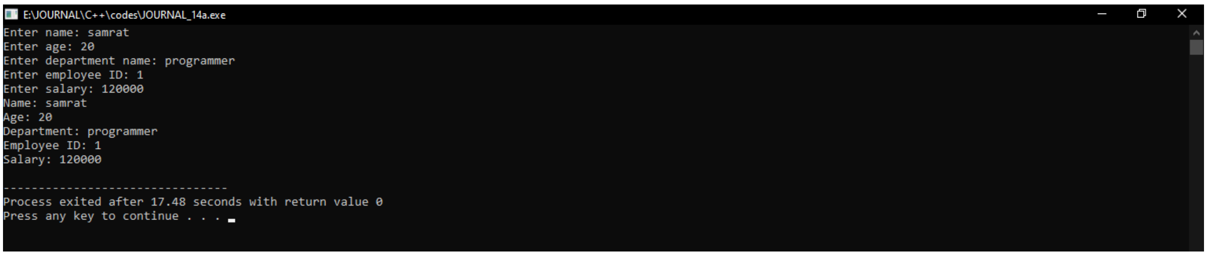
E:\JOURNAL\C++\codes\JOURNAL_14a.exe

```
Enter name: samrat
Enter age: 20
Enter department name: programmer
Enter employee ID: 1
Enter salary: 120000
Name: samrat
Age: 20
Department: programmer
Employee ID: 1
Salary: 120000

--------------------------------
Process exited after 17.48 seconds with return value 0
Press any key to continue . . .
```

DYNAMIC POLYMORPHISM:

```
#include<iostream>
using namespace std;
// Base class
class Person {
protected:
    string name;
    int age;
public:
    Person() {
        cout << "Enter name: ";
        cin >> name;
        cout << "Enter age: ";
        cin >> age;
    }
    virtual void displayInfo() { // Virtual function
        cout << "Name: " << name << endl;
        cout << "Age: " << age << endl;
    }
};
// Derived class inheriting from Person
class Department : public Person {
protected:
    string departmentName;
public:
    Department() {
        cout << "Enter department name: ";
        cin >> departmentName;
    }
    void displayInfo() override { // Overridden virtual function
        Person::displayInfo(); // Call base class function
        cout << "Department: " << departmentName << endl;
    }
};
// Derived class inheriting from Department
class Employee : public Department {
private:
    int employeeId;
    double salary;
public:
    Employee() {
        cout << "Enter employee ID: ";
        cin >> employeeId;
        cout << "Enter salary: ";
        cin >> salary;
    }
    void displayInfo() override { // Overridden virtual function
        Department::displayInfo(); // Call intermediate class function
        cout << "Employee ID: " << employeeId << endl;
        cout << "Salary: " << salary << endl;
    }
};
int main() {
    // Creating pointer to base class
    Person* p;
    // Creating objects of derived classes
    Department dep;
    Employee emp;
    // Polymorphic behavior
    p = &dep;
    p->displayInfo();
    p = &emp;
    p->displayInfo();
    return 0;
}
```

OUTPUT:



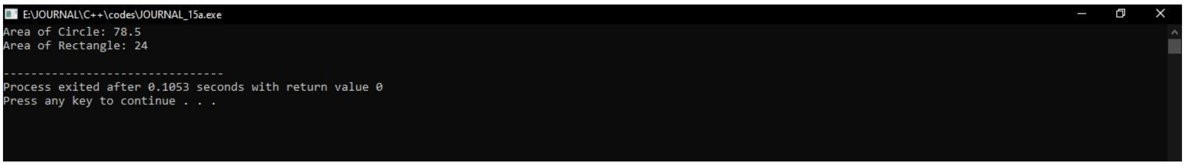
```
Enter name: samrat
Enter age: 20
Enter department name: programmer
Enter employee ID: 1
Enter salary: 120000
Name: samrat
Age: 20
Department: programmer
Employee ID: 1
Salary: 120000

--------------------------------
Process exited after 17.48 seconds with return value 0
Press any key to continue . . .
```

15. Demonstrate virtual & pure virtual function.

VIRTUAL FUNCTION:

```
#include <iostream>
using namespace std;
// Base class
class Shape {
public:
    // Virtual function to calculate area
    virtual double area() {
        cout << "Area of Shape" << endl;
        return 0;
    }
};
// Derived class: Circle
class Circle : public Shape {
private:
    double radius;
public:
    Circle(double r) : radius(r) {}
    // Override base class virtual function
    double area() override {
        return 3.14 * radius * radius;
    }
};
// Derived class: Rectangle
class Rectangle : public Shape {
private:
    double length;
    double width;
public:
    Rectangle(double l, double w) : length(l), width(w) {}
    // Override base class virtual function
    double area() override {
        return length * width;
    }
};
int main() {
    // Pointer to base class
    Shape* shape;
    // Creating objects of derived classes
    Circle circle(5);
    Rectangle rectangle(4, 6);
    // Pointing to objects and calling area function
    shape = &circle;
    cout << "Area of Circle: " << shape->area() << endl;
    shape = &rectangle;
    cout << "Area of Rectangle: " << shape->area() << endl;
    return 0;
}
```

OUTPUT:

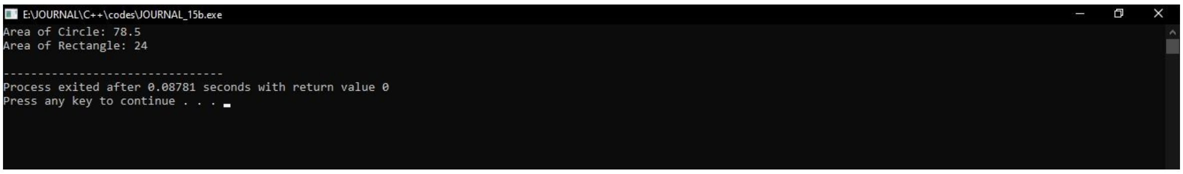


```
Area of Circle: 78.5
Area of Rectangle: 24

--------------------------------
Process exited after 0.1053 seconds with return value 0
Press any key to continue . . .
```

PURE VIRTUAL FUNCTION:

```
#include <iostream>
using namespace std;
// Abstract Base class
class Shape {
public:
    // Pure virtual function to calculate area
    virtual double area() = 0;
};
// Derived class: Circle
class Circle : public Shape {
private:
    double radius;
public:
    Circle(double r) : radius(r) {}
    // Override base class pure virtual function
    double area() override {
        return 3.14 * radius * radius;
    }
};
// Derived class: Rectangle
class Rectangle : public Shape {
private:
    double length;
    double width;
public:
    Rectangle(double l, double w) : length(l), width(w) {}
    // Override base class pure virtual function
    double area() override {
        return length * width;
    }
};
int main() {
    // Pointer to base class
    Shape* shape;
    // Creating objects of derived classes
    Circle circle(5);
    Rectangle rectangle(4, 6);
    // Pointing to objects and calling area function
    shape = &circle;
    cout << "Area of Circle: " << shape->area() << endl;
    shape = &rectangle;
    cout << "Area of Rectangle: " << shape->area() << endl;
    return 0;
}
```

OUTPUT:



```
Area of Circle: 78.5
Area of Rectangle: 24

--------------------------------
Process exited after 0.08781 seconds with return value 0
Press any key to continue . . .
```