INSTITUTE OF TECHNOLOGY
& MANAGEMENT
GWALIOR • MP • INDIA

Laboratory Manual

OBJECT ORIENTED PROGRAMMING & METHODOLGY
(IT-304)

For
Second Year Students
Department: Information Technology

Department of Information Technology and Engineering

Vision of IT Department

The Department of Information Technology envisions preparing technically competent problem solvers, researchers, innovators, entrepreneurs, and skilled IT professionals for the development of rural and backward areas of the country for the modern computing challenges.

Mission of the CSE Department

- To offer valuable education through an effective pedagogical teaching-learning process.
- To shape technologically strong students for industry, research & higher studies.
- To stimulate the young brain entrenched with ethical values and professional behaviors for the progress of society.

Program Educational Objectives

Graduates will be able to

- Our graduates will show management skills and teamwork to attain employers' objectives in their careers.
- Our graduates will explore the opportunities to succeed in research and/or higher studies.
- Our graduates will apply technical knowledge of Information Technology for innovation and entrepreneurship.
- Our graduates will evolve ethical and professional practices for the betterment of society.

Program Outcomes (POs)

Engineering Graduates will be able to:

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering Fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: Identify, formulate, review research literature, and analyze complexengineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions**: Design solutions for complex engineering problemsand design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data,and synthesis of the information to provide valid conclusions.
5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society**: Apply reasoning informed by the contextual knowledge toassess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work**: Function effectively as an individual, and as a member orleader in diverse teams, and in multidisciplinary settings.
10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, andgive and receive clear instructions.
11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Course Outcomes

Object Oriented Programming & Methodology (IT-304)

CO1 :	Understand and Describe of object oriented programming and discuss advantages over procedure oriented programming
CO2 :	Demonstrate and apply C++ classes and Objects concepts using Function, constructors , arrays and polymorphism
CO3 :	Understand and Apply the concepts of inheritance and operator overloading using c++ programming..
CO4 :	Analyze memory management concepts and implement pointers and virtual function
CO5 :	Design application to solve real world problems.

Course	Course Outcomes	CO Attainment	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	Understand and Describe of object oriented programming and discuss advantages over procedure oriented programming		1	1	0	0	0	0	0	0	0	0	0	0	0	2	0
CO2	Demonstrate and apply C++ classes and Objects concepts using Function,constructors , arrays and polymorphism		1	1	1	1	0	0	0	0	0	0	0	0	1	2	0
CO3	.Understand and Apply the concepts of inheritance and operator overloading using c++ programming.		1	1	1	0	0	0	0	0	0	0	0	0	0	2	2
CO4	Analyse memory management concepts and implement pointers and virtual function		2	1	1	0	0	0	0	0	0	0	0	0	2	0	1
CO5	Design application to solve real world problems.		1	1	1	1	1	0	0	0	1	0	1	0	2	0	2

List of Program

S. No.	List	Course Outcome	Page No.
1	Write a program to find out the largest number using function.	CO1, CO2	1-2
2	Write a program to find the area of circle, rectangle and triangle using function overloading.	CO1, CO2	3-4
3	Write a program to implement complex numbers using operator overloading and type conversion.	CO1, CO2	5-6
4	. Write a C++ program to display Student details using classes	CO2, CO3	7-8
5	Write a program which defines a class with constructor and destructor which will count number of object created and destroyed.	CO2, CO3	9-9
6	Write a program to implement single and multiple inheritances taking student as the sample base class	CO2, CO3	10-12
7	Write a program to add two private data members using friend function.	CO2, CO3	13-13
8	Write a program using dynamic memory allocation to perform 2x2 matrix addition and subtraction.	CO3, CO4	14-16
9	Write a program to create a stack using virtual function.	CO3, CO4	17-18
10	Write a program that store five student records in a file.	CO4, CO5	19-21

Program-1

Write a program to find out the largest number using function.

```
 #include<stdio.h>

int biggest(int, int, int);

// function prototype
int main()
{
int a, b, c;

printf("Enter 3 integer numbers\n");

scanf("%d%d%d", &a, &b, &c);

//function call biggest(a, b, c)

printf("Biggest of %d, %d and %d is %d\n", a, b, c, biggest(a, b, c));
return 0;
}
// function definition
int biggest(int x, int y, int z)
{
if(x > y && x > z)
{
return x;
}
else
{
if(y > z)
return y;
else
return z ;
}
}
```

Explanation

Start

Input four numbers a, b, c, d

```
If a > b then
If a > c then
If a > d then
A is the greatest
Else
D is the greatest
Else if b > c then
If b > d then
B is the greatest
Else
D is the greatest
Else if c > d then
C is the greatest
Else
D is the greatest
```

Output

```
Enter 3 integer numbers
2
4
6
Biggest of 2,4,and 6 is 6
```

Program-2

Write a program to find the area of circle, rectangle and triangle using function overloading.

```
#include<iostream.h>
#include<conio.h>
const float pi=3.14;
float area(float n,float b,float h)
{
float ar;
ar=n*b*h;
return ar;
}
float area(float r)
{
float ar;
ar=pi*r*r;
return ar;
}
float area(float l,float b)
{
float ar;
ar=l*b;
return ar;
}
void main()
{
float b,h,r,l;
float result;
clrscr();
cout<<"\nEnter the Base & Hieght of Triangle: \n";
cin>>b>>h;
result=area(0.5,b,h);
cout<<"\nArea of Triangle: "<<result<<endl;
cout<<"\nEnter the Radius of Circle: \n";
cin>>r;
result=area(r);
cout<<"\nArea of Circle: "<<result<<endl;
cout<<"\nEnter the Length & Bredth of Rectangle: \n";
cin>>l>>b;
result=area(l,b);
cout<<"\nArea of Rectangle: "<<result<<endl;
getch();
}
```

Explanation:

Algorithm:
Step 1: start the program.
Step 2: declare the class name as fn with data members and member functions.
Step 3: read the choice from the user.
Step 4: choice=1 then go to the step 5.
Step 5: the function area() to find area of circle with one integer argument.
Step 6: choice=2 then go to the step 7.
Step 7: the function area() to find area of rectangle with two integer argument.
Step 8: choice=3 then go to the step 9.
Step 9: the function area() to find area of triangle with three arguments, two as integer and one as float.
Step 10: choice=4 then stop the program.

Output

Enter the Base & Height of triangle :

12
15

Area of Triangle : 90
Enter the Radius of Circle :

Program-3

Write a program to implement complex numbers using operator overloading and type conversion

```
#include <iostream>
using namespace std;
class Complex
{
private:
float real;
float imag;
public:
Complex(): real(0), imag(0){ }
void input()
{
cout << "Enter real and imaginary parts respectively: ";
cin >> real;
cin >> imag;
}
// Operator overloading
Complex operator - (Complex c2)
{
Complex temp;
temp.real = real - c2.real;
temp.imag = imag - c2.imag;
return temp;
}
void output()
{
if(imag < 0)
cout << "Output Complex number: "<< real << imag << "i";
else
cout << "Output Complex number: " << real << "+" << imag << "i";}};
int main()
{Complex c1, c2, result;
cout<<"Enter first complex number:\n";
c1.input();
cout<<"Enter second complex number:\n";
c2.input();
result = c1 - c2;
result.output();
return 0;
}
```

Explanation:

STEP 1: Call the header file iostream.

STEP 2: Use the namespace std.

STEP 3: Create a class complex with float variables real and imag;

STEP 4: Create a constructor complex( ); set the value of real and imag to 0

STEP 5: Define the function for reading the real and imaginary parts of the numbers from the user.

STEP 6: Define a function for operator overloading.

STEP 7: Define a function to display the real and imaginary parts of the complex number.

STEP 8:Create three objects for the class complex, c1, c2, and result;

STEP 9: Read the numbers from the user and store them in the objects c1 and c2. C1.step 5 and c2.step5

STEP 10: Invoke step 6 and store the resultant number in the object result;

STEP 11: Call step 7 with the object result. result.step7;

STEP 12: exit

Output

Enter first complex number :
Enter real and imaginary parts respectively : 11
2
Enter real and imaginary parts respectively : 32
4
Output Complex number : -21-2i

Program-4

Write a C++ program to display Student details using classes.

```
#include<iostream>
using namespace std;
class student
{
private:
char name[20],regd[10],branch[10];
int sem;
public:
void input();
void display();
};
void student::input()
{
cout<<"Enter Name:";
cin>>name;
cout<<"Enter Regdno.:";
cin>>regd;
cout<<"Enter Branch:";
cin>>branch;
cout<<"Enter Sem:";
cin>>sem;
}
void student::display()
{
cout<<"\nName:"<<name;
cout<<"\nRegdno.:"<<regd;
cout<<"\nBranch:"<<branch;
cout<<"\nSem:"<<sem;
int main()
{
student s;

s.input();

s.display();
}
```

Explanation

Algorithm

Step 1: start the program.

Step 2: declare the data members.

Step 3: define the data members outside of the class.

Step 4: read the student details ie. name, regd, Sem, branch

Step 5: calculate average of marks using

Avg = (m1+m2+m3)/3

Step 6: display the student details.

Step 7: stop the program.

Output

Enter Name : Prakhar
Enter Regdno : 14
Enter Branch : it
Enter Sem : 3rd

Program-5

Write a program which defines a class with constructor and destructor which will count number of object and created and destroyed.

```
#include<iostream.h>
#include<conio.h>
class ObjectCounter {
private:
static int objectCount; // Static member variable to track the object count
public:
ObjectCounter() {
objectCount++; // Increment the object count when an object is created
}
~ObjectCounter() {
objectCount--; // Decrement the object count when an object is destroyed
}
static int getCount() {
return objectCount; // Return the current object count
}
};
int ObjectCounter::objectCount = 0; // Initialize the static member variable
int main() {
ObjectCounter obj1; // Object created
ObjectCounter obj2; // Object created
int count = ObjectCounter::getCount(); // Get the current object count
std::cout << "Number of objects: " << count << std::endl;
return 0;}
```

Explanation

The Object Counter class has a static member variable count to keep track of the number of objects created.

The constructor increments the count variable whenever a new object is created.

The destructor decrements the count variable when an object is destroyed.

The getCount() function allows us to retrieve the current count of objects.

In the main() function, we create three objects of the Object Counter class and then print out the count of objects created**.**

Output: Number of Object is : 2

Program-6

Write a program to implement single and multiple inheritances taking student as the sample base class.

Single Inheritance

```
// Example: define member function without argument within
// the class

#include <iostream>
using namespace std;
class Person {
int id;
char name[100];
public:
void set_p()
{
cout << "Enter the Id:";
cin >> id;
cout << "Enter the Name:";
cin >> name;
}
void display_p()
{
cout << endl <<"Id: "<< id << "\nName: " << name <<endl;
}
};
class Student : private Person {
char course[50];
int fee;
public
void set_s()
{
set_p();
cout << "Enter the Course Name:";
cin >> course;
cout << "Enter the Course Fee:";
cin >> fee;
}

void display_s()
{
display_p();
cout <<"Course: "<< course << "\nFee: " << fee << endl;
}
};
```

```
int main()
{
Student s;
s.set_s();
s.display_s();
return 0;
}
```

Output

```
Enter the ID : 123
Enter the Name : Prakhar
Enter the course Name : it
Enter the course fee : free
ID : 123
Name : Prakhar
Course : it
fee : 0
```

Multiple Inheritance

```
#include<iostream.h>
#include<conio.h>
class student {
protected:
int rno, m1, m2;
public:
void get() {
cout << "Enter the Roll no :";
cin>>rno;
cout << "Enter the two marks :";
cin >> m1>>m2;
}
};
class sports {
 protected:
int sm; // sm = Sports mark
public:
void getsm() {
cout << "\nEnter the sports mark :";
cin>>sm;
}
};
class statement : public student, public sports {
```

```
int tot, avg;
public:
void display() {
tot = (m1 + m2 + sm);
avg = tot / 3;
cout << "\n\n\tRoll No : " << rno << "\n\tTotal : " << tot;
cout << "\n\tAverage : " << avg;
}
};
void main() {
clrscr();
statement obj;
obj.get();
obj.getsm();
obj.display();
getch();
}
```

Explanation

Step 1: start the program.
Step 2: declare the base class student.
Step 3: declare and define the function get() to get the student details.
Step 4: declare the other class sports.
Step 5: declare and define the function getsm() to read the sports mark.
Step 6: create the class statement derived from student and sports.
Step 7: declare and define the function display() to find out the total and average.
Step 8: declare the derived class object,call the functions get(),getsm() and display().
Step 9: stop the program.

Output

```
Enter the Roll no : 92
Enter the two marks : 100
Enter the sports marks : 200

Roll No :  92
Total      : 400
Average : 133
```

Program-7

Write a program to add two private data members using friend function.

```
#include <iostream>
class MyClass {
private:
int num1;
int num2;
public:
MyClass() : num1(0), num2(0) {}
// Declare friend function
friend void addNumbers(MyClass obj);
};
// Definition of the friend function
void addNumbers(MyClass obj) {
int sum = obj.num1 + obj.num2;
std::cout << "Sum of private members: " << sum << std::endl;
}
int main() {
MyClass myObject;
std::cout << "Enter first number: ";
std::cin >> myObject.num1;
std::cout << "Enter second number: ";
std::cin>>myObject.num2;
//Call the friend function to add private members
addNumbers(myObject);
return 0;
}
```

Explanation

Step 1: start the program.
Step 2: declare the class name as base with data members and member functions.
Step 3: the function get() is used to read the 2 inputs from the user.
Step 4: declare the friend function mean(base ob) inside the class.
Step 5: outside the class to define the friend function and do the following.
Step 6: return the mean value (ob.val1+ob.val2)/2 as a float.
Step 7: stop the program.

Output

Sum is : 3

Program-8

Write a program using dynamic memory allocation to perform 2*2 matrix addition and subtraction.

```
#include <iostream>
using namespace std;
int main() {
int rows = 2, cols = 2;
int **matrix1, **matrix2, **result;
// Dynamically allocate memory for matrices
matrix1 = new int*[rows];
matrix2 = new int*[rows];
result = new int*[rows];
for (int i = 0; i < rows; ++i) {
matrix1[i] = new int[cols];
matrix2[i] = new int[cols];
result[i] = new int[cols];
}
// Input for the first matrix
cout << "Enter elements of first matrix:" << endl;
for (int i = 0; i < rows; ++i) {
for (int j = 0; j < cols; ++j) {
cin >> matrix1[i][j];
} }
// Input for the second matrix
cout << "Enter elements of second matrix:" << endl;
for (int i = 0; i < rows; ++i) {
for (int j = 0; j < cols; ++j) {
cin >> matrix2[i][j];
} }
// Addition of matrices
for (int i = 0; i < rows; ++i) {
for (int j = 0; j < cols; ++j) {
result[i][j] = matrix1[i][j] + matrix2[i][j];
} }
// Output the addition result
cout << "Addition Result:" << endl;
for (int i = 0; i < rows; ++i) {
for (int j = 0; j < cols; ++j) {
cout << result[i][j] << " ";
} cout << endl;
}
// Subtraction of matrices
for (int i = 0; i < rows; ++i) {
for (int j = 0; j < cols; ++j) {
```

```
result[i][j] = matrix1[i][j] - matrix2[i][j];
} }
// Output the subtraction result
cout << "Subtraction Result:" << endl;
for (int i = 0; i < rows; ++i) {
for (int j = 0; j < cols; ++j) {
cout << result[i][j] << " ";
} cout << endl;
}
// Deallocate memory
for (int i = 0; i < rows; ++i) {
delete[] matrix1[i];
delete[] matrix2[i];
delete[] result[i];
}
delete[] matrix1;
delete[] matrix2;
delete[] result;
return 0;
}
```

Explanation

Initialize a resultant matrix res[n][m].

- Run a for loop for counter i as each row and in each iteration:
- Run a for loop for counter j as each column and in each iteration:
- Add values of the two matrices for index i, j and store in res[i][j].
- Return res.

Output

Enter elements of first matrix :

12

12

12

12

Enter elements of Second matrix :

23

23

23

23

Additional result:

35 35

35 35

Substraction Result:

-11 -11

-11 -11

Program-9

Write a program to create a stack using virtual function.

```
#include <iostream>
using namespace std;
const int MAX_SIZE = 100; // Maximum size of the stack
class Stack {
public:
virtual void push(int element) = 0; // Virtual function for pushing an element onto the stack
virtual int pop() = 0; // Virtual function for popping an element from the stack
virtual bool isEmpty() const = 0; // Virtual function to check if the stack is empty
virtual bool isFull() const = 0; // Virtual function to check if the stack is full
};
class DynamicStack : public Stack {
private:
int top;
int stackArray[MAX_SIZE];
public:
DynamicStack() : top(-1) {}
if (!isFull()) {
stackArray[++top] = element;
cout << "Pushed element: " << element << endl;
} else {
cout << "Stack Overflow: Cannot push element " << element << ". Stack is full." << endl;
}
}
int pop() override {
if (!isEmpty()) {
int poppedElement = stackArray[top--];
cout << "Popped element: " << poppedElement << endl;
return poppedElement;
} else {
cout << "Stack Underflow: Cannot pop element. Stack is empty." << endl;
return -1;
}
}
bool isEmpty() const override {
return (top == -1);
}
bool isFull() const override {
return (top == MAX_SIZE - 1);
}};
int main() {
DynamicStack stack;
stack.push(10);
stack.push(20);
```

```
stack.push(30);
stack.pop();
stack.pop();
stack.pop();
stack.pop(); // Trying to pop when stack is empty
return 0;
}
```

Explanation

```
Push(item)
Begin
   Increase the top pointer by 1
   Insert item into the location top
End

Pop()
Begin
   Item = top element from stack
   Reduce top pointer by 1
   Return item
End

Peek()
Begin
   Item = top element from stack
   Return item
End
```

Output

```
Pushed element : 10
Pushed element:  20
Pushed element : 30
Pushed element : 30
Pushed element : 20
Pushed element : 10
Stack Underflow : Cannot pop element. Stack is empty .
```

Program-10

Write a program that store five student records in a file.

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
// Structure to hold student information
struct Student {
string name;
int rollNumber;
float marks;
};
int main() {
// Create an array to hold five student records
Student students[5];
// Get student information from the user
for (int i = 0; i < 5; ++i) {
cout << "Enter details for student " << i + 1 << ":\n";
cout << "Name: ";
getline(cin, students[i].name);
cout << "Roll Number: ";
cin >> students[i].rollNumber;
cout << "Marks: ";
cin >> students[i].marks;
cin.ignore(); // Ignore newline in buffer
}
}
// Open a file to write student records
ofstream outFile("student_records.txt");
if (!outFile) {
cerr << "Error: Unable to create file!" << endl;
return 1;
}
// Write student records to the file
for (int i = 0; i < 5; ++i) {
outFile << "Student " << i + 1 << " Details:\n";
outFile << "Name: " << students[i].name << "\n";
outFile << "Roll Number: " << students[i].rollNumber << "\n";
outFile << "Marks: " << students[i].marks << "\n\n";
}
// Close the file
outFile.close();
cout << "Student records saved to file successfully!\n";
return 0; }
```

Explanation

Step 1: call the header file iostream.

Step 2: use the using namespace std.

Step 3: create struct **student**

Step 4: create structure members' **name, roll**, and **marks**.

Step 5: create a structure array of size 5. **S[5]**

Step 6: open the integer type main function: int main().

Step 7: ask the user to enter the details of the student.

Step 8: store the entered details in the array by using a for loop.

Step 9: display the details on the screen

Step10: Exit

Output

Enter details for student 1:
Name: Prakhar
Roll Number : 92
Marks :100
Enter details student 2:
Name : special
Roll Number:00
Marks : 1000
Enter details students 3 :

Name : abc
Roll Number :0909
Marks : 000
Enetr details students 4:
Name: Xyz
Roll Number:22
Marks : 34
Students Record save to file successfully!