

PRACTICAL FILE
ON
OOPS (Object Oriented Programming)



Submitted To:

Name:—Sonam Kumari

Roll No:— 4125224

Class:— BTech (AI) 3rd sem

Submitted By:

Miss Mamtesh

Yamuna Institute Of Engineering And Technology (YIET),

Yamuna Group Of Institutions (YGI),

Vill. Gadholi, Distt. Yamuna Nagar (Haryana)

INDEX

S. No.	Name of Practical	Page No.	Date of Practical	Date of Submission	Remarks
1.	Raising a number n to a power p is the same as multiplying n by itself p times. Write a function called power() that takes a double value for n and an int value for p, and returns the result as double value. Use a default argument of 2 for p, so that if this argument is omitted, the number will be squared. Write a main () function that gets values from the user to test this function.				
2.	Write a C++ program that illustrates the order of execution of constructors and destructors when new class is derived from more than one base class. b) Write a Program to Invoking Derived Class Member Through Base Class Pointer				
3.	Create the equivalent of a four function calculator. The program should request the user to enter a number, an operator, and another number. It should then carry out the specified arithmetical operation: adding, subtracting, multiplying, or dividing the two numbers. (It should use a switch statement to select the operation). Finally it should display the result. When it finishes the calculation, the program should ask if the user wants to do another calculation. The response can be 'Y' or 'N'. Some sample interaction with the program might look like this.				
4.	A phone number, such as (212) 767-8900, can be thought of as having three parts: the area code (212), the exchange (767) and the number (8900). Write a program that uses a structure to store these three parts of a phone number separately. Call the structure phone. Create two structure variables of type phone Initialize one, and have the user input a number for the other one. Then display both numbers. The interchange might look like this:				

PRACTICAL 1.

Raising a number n to a power p is the same as multiplying n by itself p times. Write a function called `power()` that takes a double value for n and an int value for p , and returns the result as double value. Use a default argument of 2 for p , so that if this argument is omitted, the number will be squared. Write a main () function that gets values from the user to test this function.

```
#include<iostream>

using namespace std;

double power(double n,int p=2)
{
    double result=1;
    for (int i=1;i<=p;i++)
        result=result*n;
    return result;
}

int main()
{
    double n;
    int p;
    cout<<"Enter Number: ";
    cin>>n;
    cout<<"Enter Power: ";
    cin>>p;
    cout<<"Result = "<<power(n,p)<<endl;
    cout<<"Square = "<<power(n)<<endl;
    return 0;
}
```

Output

Enter Number: 2

Enter Power: 4

Result = 16

Square = 4

=== Code Execution Successful ===

downloaded from
StudentSuvidha.com

PRACTICAL 2.

Write a C++ program that illustrates the order of execution of constructors and destructors when new class is derived from more than one base class. b) Write a Program to Invoking Derived Class Member Through Base Class Pointer

(a) C++ Program to Illustrate the Order of Execution of Constructors and Destructors in Multiple Inheritance

Program:

```
#include <iostream>
using namespace std;
class A
{
public:
    A()
    {
        cout << "Constructor of A" << endl;
    }
    ~A()
    {
        cout << "Destructor of A" << endl;
    }
};
class B
{
public:
    B()
    {
        cout << "Constructor of B" << endl;
    }

    ~B()
    {
        cout << "Destructor of B" << endl;
    }
};
```

```

    }
};

class C : public A, public B
{
public:
    C()
    {
        cout << "Constructor of C" << endl;
    }
    ~C()
    {
        cout << "Destructor of C" << endl;
    }
};

int main()
{
    C obj;
    return 0;
}

```

Output
Constructor of A
Constructor of B
Constructor of C
Destructor of C
Destructor of B
Destructor of A

b)C++ Program to Invoke a Derived Class Member Through a Base Class Pointer .

program:

```
#include <iostream>

using namespace std;

class Base
{
public:
    virtual void show()
    {
        cout << "Base class" << endl;
    }
};

class Derived : public Base
{
public:
    void show() override
    {
        cout << "Derived class" << endl;
    }
};

int main()
{
    Base *ptr;
    Derived obj;
    ptr = &obj;
    ptr->show();

    return 0;
}
```

Output

Derived class

downloaded from
StudentSuvidha.com

PRACTICAL 3.

Create the equivalent of a four function calculator. The program should request the user to enter a number, an operator, and another number. It should then carry out the specified arithmetical operation: adding, subtracting, multiplying, or dividing the two numbers. (It should use a switch statement to select the operation). Finally it should display the result. When it finishes the calculation, the program should ask if the user wants to do another calculation. The response can be 'Y' or 'N'. Some sample interaction with the program might look like this.

```
#include <iostream>
using namespace std;
int main()
{
    double num1, num2, result;
    char op, choice;
    do
    {
        cout << "Enter first number: ";
        cin >> num1;
        cout << "Enter Operator: ";
        cin >> op;
        cout << "Enter second number: ";
        cin >> num2;
        switch(op)
        {
            case '+':
                result = num1 + num2;
                break;
            case '-':
                result = num1 - num2;
                break;
```

```
case '*':
    result = num1 * num2;
    break;
case '/':
    result = num1 / num2;
    break;
default:
    cout << "Invalid operator!";
    continue;
}
cout << "Answer = " << result << endl;

cout << "Do another (Y/N)? ";
cin >> choice;
} while(choice == 'Y' || choice == 'y');
return 0;
}
```

```
Output
Enter first number: 8
Enter Operator: +
Enter second number: 9
Answer = 17
Do another (Y/N)? Y
Enter first number: 9
Enter Operator: *
Enter second number: 5
Answer = 45
Do another (Y/N)? n
```

PRACTICAL 4.

A phone number, such as (212) 767-8900, can be thought of as having three parts: the area code (212), the exchange (767) and the number (8900). Write a program that uses a structure to store these three parts of a phone number separately. Call the structure phone. Create two structure variables of type phone Initialize one, and have the user input a number for the other one. Then display both numbers. The interchange might look like this:

```
#include <iostream>

using namespace std;

struct phone
{
    int areaCode;
    int exchange;
    int number;
};

int main()
{
    phone myNumber = {212, 767, 8900};
    phone yourNumber;

    cout << "Enter your area code, exchange, and number: ";
    cin >> yourNumber.areaCode >> yourNumber.exchange >> yourNumber.number;

    cout << "My number is (" << myNumber.areaCode << ") "
         << myNumber.exchange << "-" << myNumber.number << endl;

    cout << "Your number is (" << yourNumber.areaCode << ") "
         << yourNumber.exchange << "-" << yourNumber.number << endl;
```

```
return 0;
```

Output

```
Enter your area code, exchange, and number: 200 199 600  
My number is (212) 767-8900  
Your number is (200) 199-600
```

downloaded from
StudentSuvidha.com

PRACTICAL 5.

Create two classes DM and DB which store the value of distances. DM stores distances in metres and centimeters and DB in feet and inches. Write a program that can read values for the class objects and add one object of DM with another object of DB. Use a friend function to carry out the addition operation. The object that stores the results maybe a DM object or DB objects, depending on the units in which the results are required. The display should be in the format of feet and inches or meters and centimeters depending on the object on display.

Program:

```
#include <iostream>

using namespace std;

class DB; // Forward declaration

class DM
{
    float meter;
    float centimeter;
public:
    void getData()
    {
        cout << "Enter distance in meter and centimeter: ";
        cin >> meter >> centimeter;
    }
    void display()
    {
        cout << "Distance = " << meter << " meter "
            << centimeter << " centimeter" << endl;
    }
    friend DM add(DM, DB);
```

```

};

class DB
{
    float feet;
    float inch;
public:
    void getData()
    {
        cout << "Enter distance in feet and inches: ";
        cin >> feet >> inch;
    }
    void display()
    {
        cout << "Distance = " << feet << " feet "
            << inch << " inches" << endl;
    }
    friend DM add(DM, DB);
};

DM add(DM d1, DB d2)
{
    DM result;
    // Convert feet and inches into meters
    float totalMeter = d2.feet * 0.3048 + d2.inch * 0.0254;
    // Add with DM object
    float total = d1.meter + d1.centimeter / 100 + totalMeter;
    result.meter = (int)total;
}

```

```
    result.centimeter = (total - result.meter) * 100;
    return result;
}

int main()
{
    DM d1, result;
    DB d2;
    d1.getData();
    d2.getData();
    result = add(d1, d2);
    cout << "\nAfter Addition:\n";
    result.display();
    return 0;
}
```

Output

```
Enter distance in meter and centimeter: 2 5
Enter distance in feet and inches: 10 50

After Addition:
Distance = 6 meter 36.8 centimeter
```