ARAVALI COLLEGE OF ENGINEERING & MANAGEMENT

Jasana Tigoan Road Greater Faridabad Haryana, 121006

Object-oriented Programming Using C++ Lab File

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

(2024-2025)

FACULTY INCHARGE	**Submitted By:**
Mrs.Ekta Mam	**Name: Ankit kumar**
(Assistant Professor)	**Roll No: 24011312010**

Index

EXPERIMENT NO. 1

AIM: WAP to check if a Number is prime or not.

Code:

```
#include <iostream>
using namespace std;
int main() {
    int n, i, flag=0;
    cout << "Enter a number: ";
    cin >> n;
    if(n<=1) {
        cout << n << " is not prime.";
        return 0;
    }
    for(i=2; i<=n/2; ++i) {
        if(n % i == 0) {
            flag = 1;
            break;
        }
    }
    if(flag == 0)
        cout << n << " is prime.";
    else
        cout << n << " is not prime.";
    return 0;
}
```

Output:

```
 Enter a number: 13
13 is prime.
```

EXPERIMENT NO. 2

AIM: Write a program to find an element in list using binary search.

Code:

```
#include <iostream>
using namespace std;
int binarySearch(int arr[], int size, int key) {
    int left = 0, right = size-1, mid;
    while(left <= right) {
        mid = left + (right - left)/2;
        if(arr[mid] == key) return mid;
        else if(arr[mid] < key) left = mid+1;
        else right = mid-1;
    }
    return -1;
}
int main() {
    int arr[5] = {2, 5, 8, 12, 16}, key;
    cout << "Enter key to search: ";
    cin >> key;
    int res = binarySearch(arr, 5, key);
    if(res != -1)
        cout << "Element found at index " << res << endl;
    else
        cout << "Element not found." << endl;
    return 0;
}
```

Output:

```
 Enter key to search: 8
Element found at index 2
```

EXPERIMENT NO. 3

AIM: WAP to implement Student grade using Classes.

Code:

```
#include <iostream>
using namespace std;
class Student {
    string name;
    int marks;
    public:
    void getData() {
        cout << "Enter name and marks: ";
        cin >> name >> marks;
    }
    void showGrade() {
        cout << "Student: " << name << endl;
        if(marks >= 90)
            cout << "Grade: A" << endl;
        else if(marks >= 80)
            cout << "Grade: B" << endl;
        else if(marks >= 60)
            cout << "Grade: C" << endl;
        else
            cout << "Grade: D" << endl;
    }};
int main() {
    Student s;
    s.getData();
    s.showGrade();
    return 0;
}
```

Output:

```
 Enter name and marks: Rohit 85
 Student: Rohit
Grade: B
```

EXPERIMENT NO. 4

AIM: WAP to compute total salary of employees using containership.

Code:

```
#include <iostream>
using namespace std;
class Salary {
    int basic, hra, da;
    public:
    Salary(int b, int h, int d): basic(b), hra(h), da(d) {}
    int total() { return basic + hra + da; }
};
class Employee {
    Salary sal;
    string name;
    public:
    Employee(string n, int b, int h, int d): name(n), sal(b,h,d) {}
    void showTotalSalary() {
        cout << "Employee: " << name << endl;
        cout << "Total Salary: " << sal.total() << endl;
    }
};
int main() {
    Employee e("Anita", 25000, 4000, 5000);
    e.showTotalSalary();
    return 0;
}
```

Output:

```
Employee: Anita

Total Salary: 34000
```

EXPERIMENT NO. 5

AIM: WAP to calculate grade of students using array of objects.

Code:

```
#include <iostream>
using namespace std;
class Student {
    string name;
    int marks;
    public:
    void getData() {
        cout << "Enter name and marks: ";
        cin >> name >> marks;
    }
    void showGrade() {
        cout << name << ": ";
        if(marks >= 90) cout << "A";
        else if(marks >= 80) cout << "B";
        else if(marks >= 60) cout << "C";
        else cout << "D";
        cout << endl;
    }};
int main() {
    Student s[3];
    for(int i=0; i<3; i++)
        s[i].getData();
    for(int i=0; i<3; i++)
        s[i].showGrade();
    return 0;
}
```

Output:

```
Enter name and marks: Aman 92
Enter name and marks: Sara 75
Enter name and marks: Ravi 83
Aman: A   Sara: C   Ravi: B
```

EXPERIMENT NO. 5

AIM: Write a program to calculate area of different shapes using function overloading.

Code:

```
#include <iostream>
#define PI 3.14
using namespace std;
float area(int r) { return PI * r * r; }
float area(float side) { return side * side; }
float area(float r, float h) { return 2 * PI * r * h + 2 * PI * r * r; }
float area(float b, float h, int t) { return 0.5 * b * h; }
float cone(float r, float h) { return PI * r * (r + h); }
int main() {
    cout << "Circle area: " << area(3) << endl;
    cout << "Square area: " << area(4.0f) << endl;
    cout << "Cylinder area: " << area(3.0f, 5.0f) << endl;
    cout << "Triangle area: " << area(3.0f, 6.0f, 0) << endl;
    cout << "Cone area: " << cone(3.0f, 5.0f) << endl;
    return 0;
}
```

Output:

```
Circle area: 28.26
  Square area: 16
  Cylinder area: 150.72
  Triangle area: 9
  Cone area: 75.36
```

EXPERIMENT NO. 6

AIM: Write a program to find compound interest using default argument.

Code:

```
#include <iostream>
#include <cmath>
using namespace std;
float compoundInterest(float p, float r=6.5, int t=2) {
   return p * pow(1 + r/100, t) - p;
}
int main() {
   cout << "Compound Interest (default rate & time): " << compoundInterest(10000) << endl;
   cout << "Compound Interest (custom rate, time): " << compoundInterest(10000, 8.0, 3) << endl;
   return 0;
}
```

Output:

```
Compound Interest (default rate & time): 1340.88
  Compound Interest (custom rate, time): 2597.12
```

EXPERIMENT NO. 7

AIM: Write a program to do swapping of two numbers using:

(a) Call by value

(b) Call by reference

(c) Call by address

Code:

```
#include <iostream>
using namespace std;
void swapVal(int a, int b) { int t=a; a=b; b=t; cout << "Swapped (val): " << a << "," << b << endl;}
void swapRef(int &a, int &b) { int t=a; a=b; b=t; cout << "Swapped (ref): " << a << "," << b << endl;}
void swapAddr(int *a, int *b) { int t=*a; *a=*b; *b=t; cout << "Swapped (addr): " << *a << "," << *b <<
endl;}
int main() {
   int x=5, y=7;
   swapVal(x, y);
   swapRef(x, y);
   swapAddr(&x, &y);
   return 0;
}
```

Output:

```
Swapped (val): 7,5
  Swapped (ref): 5,7
  Swapped (addr): 7,5
```

EXPERIMENT NO. 8

AIM: Write a program to have 2 times addition using argument passing.

Code:

```
#include <iostream>
using namespace std;
int add(int a, int b) { return a + b; }
int twiceAdd(int a, int b) { return add(a, b) + add(a, b); }
int main() {
    int x=2, y=3;
    cout << "Twice addition: " << twiceAdd(x, y) << endl;
    return 0;
}
```

Output:

```
Twice addition: 10
```

EXPERIMENT NO. 9

AIM: Write a program to addition of two Matrix using argument passing.

Code:

```
#include <iostream>
using namespace std;
void addMatrix(int a[2][2], int b[2][2]) {
    int c[2][2];
    for(int i=0;i<2;i++)
        for(int j=0;j<2;j++)
            c[i][j]=a[i][j]+b[i][j];
    cout << "Sum matrix:\n";
    for(int i=0;i<2;i++){
        for(int j=0;j<2;j++)
            cout << c[i][j] << " ";
        cout << endl;
    }
}
int main() {
    int m1[2][2]={{1,2},{3,4}}, m2[2][2]={{5,6},{7,8}};
    addMatrix(m1, m2);
    return 0;
}
```

Output:

```
Sum matrix:
  6 8
  10 12
```

EXPERIMENT NO. 10

AIM: Write a program to add two complex number using constructor function.

Code:

```
#include <iostream>
using namespace std;
class Complex {
    float re, im;
    public:
    Complex(float r=0, float i=0): re(r), im(i) {}
    Complex add(Complex c) {
        return Complex(re + c.re, im + c.im);
    }
    void display() {
        cout << re << " + " << im << "i" << endl;
    }
};
int main() {
    Complex c1(1.5,2.5), c2(2.5,3.5);
    Complex sum = c1.add(c2);
    cout << "Sum: ";
    sum.display();
    return 0;
}
```

Output:

```
Sum:
4 + 6i
```

EXPERIMENT NO. 11

AIM: WAP to implement friend function to add two complex numbers.

Code:

```
#include <iostream>
using namespace std;
class Complex {
    float re, im;
    public:
    Complex(float r=0, float i=0): re(r), im(i) {}
    friend Complex add(Complex, Complex);
    void display() {
        cout << re << " + " << im << "i" << endl;
    }
};
Complex add(Complex a, Complex b) {
    return Complex(a.re+b.re, a.im+b.im);
}
int main() {
    Complex c1(2.1,3.3), c2(1.2,4.1);
    Complex sum = add(c1, c2);
    cout << "Sum: ";
    sum.display();
    return 0;
}
```

Output:

```
Sum:
3.3 + 7.4i
```

EXPERIMENT NO. 12

AIM: Write a program to add two complex numbers using overloading binary + operator.

Code:

```
#include <iostream>
using namespace std;
class Complex {
    float re, im;
    public:
    Complex(float r=0, float i=0): re(r), im(i) {}
    Complex operator+(Complex c) {
        return Complex(re + c.re, im + c.im);
    }
    void display() {
        cout << re << " + " << im << "i" << endl;
    }
};
int main() {
    Complex c1(3,2), c2(1,7);
    Complex sum = c1 + c2;
    cout << "Sum: ";
    sum.display();
    return 0;
}
```

Output:

```
Sum:
4 + 9i
```

EXPERIMENT NO. 13

AIM: Write a program to implement overloading unary - operator using point class.

Code:

```
#include <iostream>
using namespace std;
class Point {
    int x, y;
    public:
    Point(int xa=0, int ya=0): x(xa), y(ya) {}
    Point operator-() {
        return Point(-x, -y);
    }
    void show() {
        cout << "(" << x << "," << y << ")" << endl;
    }
};
int main() {
    Point p(2,3);
    Point neg = -p;
    neg.show();
    return 0;
}
```

Output:

```
(-2,-3)
```

EXPERIMENT NO. 14

AIM: Write a program to compare two length object by using == operator.

Code:

```
#include <iostream>
using namespace std;
class Length {
    int meter;
    public:
    Length(int m=0): meter(m) {}
    bool operator==(Length l) {
        return meter == l.meter;
    }
};
int main() {
    Length l1(5), l2(5);
    if(l1 == l2)
        cout << "Lengths are equal." << endl;
    else
        cout << "Lengths are not equal." << endl;
    return 0;
}
```

Output:

```
Lengths are equal.
```

EXPERIMENT NO. 15

AIM: Write a program to implement increment/decrement operator on times class object using overloading.

Code:

```
#include <iostream>
using namespace std;
class Times {
    int t;
    public:
    Times(int v=0): t(v) {}
    Times operator++() { t++; return *this; }
    Times operator--() { t--; return *this; }
    void show() { cout << "Value: " << t << endl; }
};
int main() {
    Times obj(5);
    ++obj;
    obj.show();
    --obj;
    obj.show();
    return 0;
}
```

Output:

```
    Value: 6
    Value: 5
```