**Experiment: 11**

PART A

(PART A: TO BE REFFERED BY STUDENTS)

**Aim:** **To study polymorphism in C++.**

**Learning Outcomes: Learner would be able to**

Implement function overloading and operator overloading.

**Task 1:** Create a class named “Shapes” with integer variable “Area”. Write a member function “calArea” with two float parameters to calculate the area of rectangle and overload the same function with having one float parameter to calculate the area of square.

**Task 2:** Write a program to overload + operator to add two instances of time (having hours and minutes).

**Task 3:** write a program using class distance that creates an object and gets value from user in feet and inches. It then adds these values with the values of another object by overloading of + operator.

**Task 4:** Write a program to overload sum function to perform addition of two integers, 3 integers and two floating point numbers.

**Theory:**

In C++, two functions can have the same name if the number and/or type of arguments passed is different.

These functions having the same name but different arguments are known as overloaded functions. For example:

// same name different arguments

int test() { }

int test(int a) { }

float test(double a) { }

int test(int a, double b) { }

Here, all 4 functions are overloaded functions.

Notice that the return types of all these 4 functions are not the same. Overloaded functions may or may not have different return types but they must have different arguments. For example,

// Error code

int test(int a) { }

double test(int b){ }

Here, both functions have the same name, the same type, and the same number of arguments. Hence, the compiler will throw an error.

In C++, we can make operators to work for user defined classes. This means C++ has the ability to provide the operators with a special meaning for a data type, this ability is known as operator overloading.

For example, we can overload an operator ‘+’ in a class like String so that we can concatenate two strings by just using +.

Other example classes where arithmetic operators may be overloaded are Complex Numbers.

#include<iostream>

using namespace std;

class Complex {

private:

int real, imag;

public:

Complex(int r = 0, int i =0) {real = r; imag = i;}

// This is automatically called when '+' is used with

// between two Complex objects

Complex operator + (Complex const &obj) {

Complex res;

res.real = real + obj.real;

res.imag = imag + obj.imag;

return res;

}

void print() { cout << real << " + i" << imag << endl; }

};

int main()

{

Complex c1(10, 5), c2(2, 4);

Complex c3 = c1 + c2; // An example call to "operator+"

c3.print();

}

PART B

(PART B: TO BE COMPLETED BY STUDENTS)

Students must submit the soft copy as per following segments within two hours of the practical. The soft copy must be uploaded on the portal at the end of the practical. The filename should be **PPS\_batch\_rollno\_experimentno Example: PPS\_B2\_B001\_Exp1**

|  |  |
| --- | --- |
| **Roll No.:** | **Name:** |
| **Prog/Yr/Sem:** | **Batch:** |
| **Date of Experiment:** | **Date of Submission:** |

**Task 1:**

**Task 2:**

**Task 3:**

**Task 4:**

**Conclusion (Learning Outcomes):** Reflect on the questions answered by you jot down your learnings about the Topic: Structure and Union

**Home Work Questions:**

1. What is constructor overloading?

2. What are the operators which cannot be overloaded?