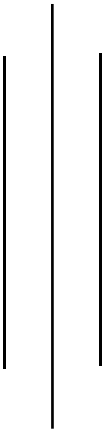
TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PULCHOWK CAMPUS



LAB REPORT ON

OBJECT ORIENTED PROGRAMMING [CT 451]

LAB 4

DATA CONVERSION IN C++

Submitted by:

Rujal Acharya

PUL076BEI029

Submitted to:

Department of Electronics and Computer Engineering, Pulchowk Campus

Institute of Engineering, Tribhuvan University

Lalitpur, Nepal

November, 2020

Task 1

Problem:

Write a program in CPP to convert the distance in meters entered by the user into distance in feet and inch using the concept of basic to user defined data conversion.

Program:

```
#include <iostream>

class Distance {
	public:
		Distance();
		Distance(float);
		void showdata();
	private:
		float feet;
		float inch;
};

Distance::Distance() {
	feet = 0;
	inch = 0;
}
Distance::Distance(float meters) {
	float tfeet = meters * 3.2808;
	feet = int(tfeet);
	inch = (tfeet - feet) * 12;
}
void Distance::showdata() {
	std::cout << "The distance in feet and inch is: " << std::endl;
	std::cout << feet << "'" << inch << "\"" << std::endl;
}

int main() {
	Distance d;
	float meters;
	std::cout << "Enter distance in meters: " ;
	std::cin >> meters;
	d = meters;
	d.showdata();
	return 0;
}
```

Task 2

Problem:

Write a program in CPP to convert the distance in feet and inch entered by the user into distance in meters using the concept o user defined to basic data conversion.

Program:

```
#include <iostream>

class Distance {
    public:
        Distance();
        Distance(float, float);
        operator float();
        void getdata();

    private:
        float feet;
        float inch;
};

Distance::Distance() {
    feet = 0;
    inch = 0;
}

Distance::Distance(float f, float i) {
    feet = f;
    inch = i;
}

Distance::operator float() {
    return (feet + inch / 12) / 3.2808;
}

void Distance::getdata() {
    std::cout << "Enter the distance in feet and inch: " << std::endl;
    std::cin >> feet >> inch;
}

int main() {
```

```
    Distance d;
    float meters;
    d.getdata();
    meters = d;
    std::cout << "Distance in meters: " << meters << "m" << std::endl;
    return 0;
}
```

Task 3

Problem:

WAP in CPP to convert the co-ordinates from Cartesian system to polar system using the concept of user defined to user defined conversion by writing conversion routine in source class.

Program:

```
#include <iostream>
#include <cmath>

class Polar {
	public:
		Polar();
		Polar(float, float);
		void showdata();

	private:
		float r;
		float theta;
};

class Cartesian {
	public:
		Cartesian();
		Cartesian(float, float);
		operator Polar();
		void getdata();

	private:
		float x;
		float y;
};

Polar::Polar() {
	r = 0;
	theta = 0;
}

Polar::Polar(float ar, float th) {
	r = ar;
	theta = th;
```

```
}

void Polar::showdata() {
	std::cout << "The coordinate in polar form is (" << r << "," << theta << ")" << std::endl;
}

Cartesian::Cartesian() {
	x = 0;
	y = 0;
}

Cartesian::Cartesian(float xx, float yy) {
	x = xx;
	y = yy;
}

void Cartesian::getdata() {
	std::cout << "Enter the coordinate in cartesian form: " ;
	std::cin >> x >> y;
}

Cartesian::operator Polar() {
	return Polar(sqrt(pow(x, 2) + pow(y, 2)), atan(y / x));
}

int main() {
	Cartesian cartesian;
	Polar polar;
	cartesian.getdata();
	polar = cartesian;
	polar.showdata();
	return 0;
}
```

Task 4

Problem:

WAP in CPP to convert the co-ordinates from Cartesian system to polar system using the concept of user defined to user defined conversion by writing conversion routine in destination class.

Program:

```
#include <iostream>
#include <cmath>

class Cartesian {
    public:
        Cartesian();
        Cartesian(float, float);
        void getdata();

    private:
        float x;
        float y;

        friend class Polar;

};

class Polar {
    public:
        Polar();
        Polar(float, float);
        Polar(Cartesian);
        void showdata();

    private:
        float r;
        float theta;
};

Cartesian::Cartesian() {
    x = 0;
    y = 0;
}
```

```
Cartesian::Cartesian(float xx, float yy) {
	x = xx;
	y = yy;
}

void Cartesian::getdata() {
	std::cout << "Enter the coordinate in cartesian form: " ;
	std::cin >> x >> y;
}

Polar::Polar() {
	r = 0;
	theta = 0;
}

Polar::Polar(float ar, float th) {
	r = ar;
	theta = th;
}

void Polar::showdata() {
	std::cout << "The coordinate in polar form is (" << r << "," << theta << ")" << std::endl;
}


Polar::Polar(Cartesian c) {
	r = sqrt(pow(c.x, 2) + pow(c.y, 2));
	theta = atan(c.y / c.x);
}


int main() {
	Cartesian cartesian;
	cartesian.getdata();
	Polar polar;
	polar = cartesian;
	polar.showdata();
	return 0;
}
```