**LAB SHEET 10**

1.SOURCE CODE:

#include<iostream>

//basic to class type with the help of constructor (meter to feet and inches)

using namespace std;

class height{

private:

int feet;

float inches;

public:

height(){

}

height(float m){

float f = m \* 2.2808;

feet = (int)f;

inches = (f - feet) \* 12;

}

void display(){

cout<<feet<<"foot "<<inches<<"inches"<<endl;

}

};

int main(){

height h;

float meter;

cout<<"Enter the value of height in meter:"<<endl;

cin>>meter;

//basic to class type

h = meter;

h.display();

return 0;

}

OUTPUT

Enter the value of height in meter:

4

9foot 1.4784inches

--------------------------------

Process exited after 7.864 seconds with return value 0

Press any key to continue . . .

2.SOURCE CODE:

#include<iostream>

//feet and inches to meter(class to basic) ...using casting operator function

using namespace std;

class height{

private:

int feet;

float inches;

public:

height(int f, float i){

feet = f;

inches = i;

}

void display(){

cout<<feet<<"feet "<<inches<<"inches"<<endl;

}//for float type meter

operator float(){

float f = inches/12;

f = f + feet;

return (f/3.2808);

}

};

int main(){

height h(5, 47.896);

float meter;

meter = h;

cout<<"The given height in meter is => "<<meter<<"m"<<endl;

return 0;

}

OUTPUT

The given height in meter is => 2.74059m

--------------------------------

Process exited after 0.05979 seconds with return value 0

Press any key to continue . . .

3.SOURCE CODE:

#include <iostream>

using namespace std;

class memory{

private:

int kilobites, megabites;

int bytes;

public:

//since it is basic to class type so just use casting operator function in

memory(){

}

memory(int b){

int n;

megabites = b / (1024\*1024);

n= b%(1024\*1024);

kilobites = n/1024;

bytes = n%1024;

}void display(){

cout<<megabites<<"Mb "<<kilobites<<"Kb "<<bytes<<"b"<<endl;

}

};

int main(){

memory m;

int bytes;

cout<<"Enter the amount of memory in bytes:"<<endl;

cin>>bytes;

m = bytes;//basic to class type

m.display();

return 0;

}

OUTPUT:

Enter the amount of memory in bytes:

4253554

4Mb 57Kb 882b

--------------------------------

Process exited after 7.34 seconds with return value 0

Press any key to continue . . .

4.SOURCE CODE:

#include <iostream>

#include <math.h>

using namespace std;

class rectangle{

private:

float xco, yco;

public:

rectangle(){

}

rectangle(float x,float y){

xco = x;

yco = y;

}

void display(){

cout<<"("<<xco<<","<<yco<<")"<<endl;

}

};

class polar{

private:

float radius;

float angle;

public:

polar(){

}

polar(float r, float a){

radius = r;

angle = a;

}

void display(){

cout<<"("<<radius<<","<<angle<<")"<<endl;

}

operator rectangle(){

float x=radius\*cos(angle);

float y=radius\*sin(angle);

return rectangle(x,y);

}

};

int main(){

polar p(5.587, 36.78);

rectangle r;

r = p;

cout<<"The polar coordinates:"<<endl;

p.display();

cout<<"The recatangular coordinates:"<<endl;

r.display();

return 0;

}

OUTPUT:

The polar coordinates:

(5.587,36.78)

The recatangular coordinates:

(3.38866,-4.44202)

--------------------------------

Process exited after 0.06154 seconds with return value 0

Press any key to continue . . .

5.SOURCE CODE:

#include <iostream>

#include <math.h>

using namespace std;

class polar{

private:

float radius;

float angle;

public:

polar(){

}

polar(float r, float a){

radius = r;

angle = a;

}

void display(){

cout<<"("<<radius<<","<<angle<<")"<<endl;

}

};

class rectangle{

private:

float xco, yco;

public:

rectangle(float x, float y){

xco = x;

yco = y;

}

void display(){

cout<<"("<<xco<<","<<yco<<")"<<endl;

}

operator polar(){

float r = sqrt((xco\*xco)+ (yco\*yco));

float a = tan(yco/xco);

return polar(r,a);

}

};

int main(){

rectangle r(36.38,12.98);

polar p;

p = r;

cout<<"The rectangular coordinates:"<<endl;

r.display();

cout<<"The polar coordinates:"<<endl;

p.display();

return 0;

}

OUTPUT:

The rectangular coordinates:

(36.38,12.98)

The polar coordinates:

(38.6262,0.372742)

--------------------------------

Process exited after 0.05485 seconds with return value 0

Press any key to continue . . .

6.SOURCE CODE:

#include<iostream>

#include<math.h>

using namespace std;

class rectangle{

private:

float xco, yco;

public:

rectangle(){

}

rectangle(float x, float y){

xco = x;

yco = y;

}

void display(){

cout<<"("<<xco<<","<<yco<<")"<<endl;

}

float getx(){

return xco;

}

float gety(){

return yco;

}

};

class polar{

private:

float radius ;

float angle;

public:

polar(){

}

polar(float r, float a){

radius = r;

angle = a;

}

void display(){

cout<<"("<<radius<<","<<angle<<")"<<endl;

}

polar(rectangle R){

float x = R.getx();

float y = R.gety();

radius = sqrt(x\*x + y\*y);

angle = tan(y/x);

}

};

int main(){

rectangle r(34.47, 67.38);

polar p;

p = r;

cout<<"Rectangular coordinates.."<<endl;

r.display();

cout<<"Polar coordinates...."<<endl;

p.display();

return 0;

}

OUTPUT:

Rectangular coordinates..

(34.47,67.38)

Polar coordinates....

(75.6852,-2.47527)

--------------------------------

Process exited after 0.05642 seconds with return value 0

Press any key to continue . . .

7.SOURCE CODE:

#include <iostream>

using namespace std;

class celsius{

private:

float deg\_cel;

public:

celsius(){

}

celsius(float c){

deg\_cel = c;

}

void display(){

cout<<deg\_cel<<" C"<<endl;

}

float getdc(){

return deg\_cel;

}

};

class ferhenheit{

private:

float deg\_ferh;

public:

ferhenheit(){

}

ferhenheit(float f){

deg\_ferh=f;

}

void display(){

cout<<deg\_ferh<<" F"<<endl;

}

ferhenheit(celsius C){

float c = C.getdc();

deg\_ferh= ((1.8\*c) + 32);

}

};

int main(){

celsius c(37.43);

ferhenheit f;

f = c;

cout<<"Temperature in degree celsius.... "<<endl;

c.display();

cout<<"Temperature in degree ferhenheit...."<<endl;

f.display();

return 0;

}

OUTPUT:

Temperature in degree celsius....

37.43 C

Temperature in degree ferhenheit....

99.374 F

--------------------------------

Process exited after 0.05513 seconds with return value 0

Press any key to continue . . .