**PROBLEM STATEMENT:**

Define a class DISTANCE with feet and inches as its data members. Using the concept of operator overloading, convert the distance given in metres to an object of type DISTANCE and vice-versa.

**PROGRAM CODE:**

#include<iostream.h>

#include<conio.h>

#define M 3.280839895 //To convert metres to feet

#define F 0.3048 //To convert feet to metres

#define I 0.0254 //To convert inches to metres

class distance

{

private:

float feet,inches;

public:

distance()

{

feet=0.0;

inches=0.0;

}

void get(float f,float i)

{

feet=f;

inches=i;

}

void show()

{

cout<<"\nDistance = "<<feet<<" feet"<<" and "<<inches<<" inches";

}

void operator=(float a)

{

float frac=a\*M;

feet=int(frac);

inches=(frac-feet)\*12;

}

operator float()

{

float mtrs=0;

mtrs=feet\*F;

mtrs+=inches\*I;

return mtrs;

}

};

int main()

{

int ch;

float metres,f,i;

distance d1,d2;

x:

cout<<"\n\nChoose:\n1. Basic datatype (metres) --> Object (feet & inches)\n2. Object (feet & inches) --> Basic datatype (metres)\n3. Exit\n";

cin>>ch;

switch(ch)

{

case 1:

cout<<"\nEnter metres: ";

cin>>metres;

d1=metres;

d1.show();

getch();

goto x;

case 2:

float result;

cout<<"\nEnter feet and inches: ";

cin>>f>>i;

d2.get(f,i);

result=d2;

cout<<"\nDistance = "<<result<<" metres";

getch();

goto x;

case 3:

break;

default:

cout<<"Enter a valid choice!";

getch();

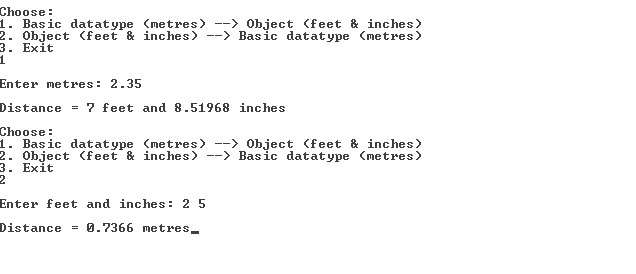
goto x;

}

return 0;

}

**OUTPUT:**



**RESULT:**

Hence operator overloading is employed to enable data conversion from user-defined type (class object) to built-in type (float) and vice versa.