Problem Statement: Distance Calculation Using Operator Overloading

You are required to implement a program that calculates distances using operator overloading in C++. The program should be able to perform the following operations on distances:

Addition of Distances:

Implement an addition operator (+) that adds two distances together.

The distance should be represented in feet and inches.

Subtraction of Distances:

Implement a subtraction operator (-) that subtracts one distance from another.

Ensure that the subtraction operation handles cases where the result may involve negative values or borrowing (like in subtraction of inches).

Comparison of Distances:

Implement comparison operators (==, !=, <, >, <=, >=) to compare distances based on their total length (combined feet and inches).

Use these operators to determine which distance is greater, less than, or equal to another.

Requirements:

Distance Class: Implement a Distance class with appropriate member variables (feet and inches).

Constructors: Implement constructors to initialize distances.

Member Functions: Implement member functions for display and any other necessary operations.

Operator Overloading: Overload the necessary operators (+, -, ==, !=, <, >, <=, >=) inside the Distance class to perform the specified operations.

Testing: Create a main() function to test the implemented Distance class and its operator overloading functionality. Test various scenarios including addition, subtraction, and comparison of distances.

#include <iostream>

class Distance {

private:

int feet;

float inches;

public:

Distance(int ft = 0, float in = 0) : feet(ft), inches(in) {}

Distance operator+(const Distance& d) const {

int newFeet = feet + d.feet;

float newInches = inches + d.inches;

if (newInches >= 12.0) {

newInches -= 12.0;

newFeet++;

}

return Distance(newFeet, newInches);

}

Distance operator-(const Distance& d) const {

int newFeet = feet - d.feet;

float newInches = inches - d.inches;

if (newInches < 0) {

newInches += 12.0;

newFeet--;

}

return Distance(newFeet, newInches);

}

bool operator==(const Distance& d) const {

return (feet == d.feet && inches == d.inches);

}

bool operator!=(const Distance& d) const {

return !(\*this == d);

}

bool operator<(const Distance& d) const {

float thisTotal = feet \* 12 + inches;

float dTotal = d.feet \* 12 + d.inches;

return thisTotal < dTotal;

}

bool operator>(const Distance& d) const {

float thisTotal = feet \* 12 + inches;

float dTotal = d.feet \* 12 + d.inches;

return thisTotal > dTotal;

}

bool operator<=(const Distance& d) const {

return (\*this < d || \*this == d);

}

bool operator>=(const Distance& d) const {

return (\*this > d || \*this == d);

}

void display() const {

std::cout << "Distance: " << feet << " feet " << inches << " inches" << std::endl;

}

};

int main() {

Distance d1(3, 9.5);

Distance d2(4, 7.25);

Distance d3(8, 3.0);

Distance sum = d1 + d2;

std::cout << "Sum of ";

d1.display();

std::cout << "and ";

d2.display();

std::cout << "is ";

sum.display();

std::cout << std::endl;

Distance diff = d3 - d1;

std::cout << "Difference of ";

d3.display();

std::cout << "and ";

d1.display();

std::cout << "is ";

diff.display();

std::cout << std::endl;

if (d1 == d2)

std::cout << "Distance d1 is equal to d2" << std::endl;

if (d1 != d3)

std::cout << "Distance d1 is not equal to d3" << std::endl;

if (d2 < d3)

std::cout << "Distance d2 is less than d3" << std::endl;

if (d3 > d1)

std::cout << "Distance d3 is greater than d1" << std::endl;

if (d2 <= d3)

std::cout << "Distance d2 is less than or equal to d3" << std::endl;

if (d3 >= d1)

std::cout << "Distance d3 is greater than or equal to d1" << std::endl;

return 0;

}

Output:

