

Programming in C++ Overloading the < > + Operators

```
// shows overloaded '<' '>' operators to compare two Distance objects
#include <iostream>
using namespace std;
```

```
class Distance { // English Distance class
private:
    int nFeet;
    float flnches;

public:
    Distance() : nFeet(0), flnches(0.0f) { }
    Distance(int nFt, float fln) : nFeet(nFt), flnches(fln) { }
    void GetDistance()
    {
        cout << "\nEnter Feet: "; cin >> nFeet;
        cout << "Enter Inches: "; cin >> flnches;
    }
    void ShowDistance() const { cout << nFeet << "\'-" << flnches << "\'"; }
    bool operator< (Distance& rdRHS) const // inline style
    { // declare and define
        float fLHS = nFeet + flnches/12.0f;
        float fRHS = rdRHS.nFeet + rdRHS.flnches/12.0f;
        return (fLHS < fRHS) ? true : false;
    }
    bool operator> (Distance& rdRHS) const; // out-of-line style
    // declare only
};
```

```
bool Distance::operator> (Distance& rdRHS) const // define
{
    float fLHS = nFeet + flnches/12.0f;
    float fRHS = rdRHS.nFeet + rdRHS.flnches/12.0f;
    return (fLHS > fRHS) ? true : false;
}
```

```
void main()
{
    Distance dist1;
    dist1.GetDistance();

    Distance dist2(6, 2.5);

    cout << "\ndist1 = "; dist1.ShowDistance();
    cout << "\ndist2 = "; dist2.ShowDistance();

    if (dist1 < dist2)
        cout << "\ndist1 is less than dist2";
    else if (dist1 > dist2)
        cout << "\ndist1 is greater than dist2";
    else
        cout << "\ndist1 and dist2 must be equal";
    cout << endl;
}
```

Alternative Form (never used)
if (dist1.operator<(dist2))

```
// englplus.cpp -- overloaded '+' operator adds two Distances
#include <iostream>
using namespace std;
```

```
class Distance {
private:
    int nFeet;
    float flnches;

public:
    // constructor (no args)
    Distance() : nFeet(0), flnches(0.0f) { }
    Distance(int nFt, float fln) : nFeet(nFt), flnches(fln) { }
    void getdist() // get length from user
    {
        cout << "\nEnter nFeet: "; cin >> nFeet;
        cout << "Enter flnches: "; cin >> flnches;
    }
    void showdist() const { cout << nFeet << "-" << flnches << "\n"; }
    Distance operator+ (const Distance& rdRHS) const; // add 2 distances
};
```

```
Distance Distance::operator+ (const Distance& rdRHS) const
{ // return sum
    int nFt = nFeet + rdRHS.nFeet; // add the nFeet
    float fln = flnches + rdRHS.flnches; // add the flnches
    if (fln >= 12.0f) { // if total exceeds 12.0, decrease
        fln -= 12.0f; // flnches by 12.0 and
        nFt++; // increase nFeet by 1
    } // return a temporary Distance
    return Distance(nFt, fln); // initialized to sum
}
```

```
void main()
{
    Distance dist1, dist3, dist4; // define distances
    dist1.getdist(); // get dist1 from user
    Distance dist2(11, 6.25f); // define, initialize dist2
    dist3 = dist1 + dist2; // single '+' operator
    dist4 = dist1 + dist2 + dist3; // multiple '+' operators
    // display all lengths
    cout << "dist1 = "; dist1.showdist(); cout << endl;
    cout << "dist2 = "; dist2.showdist(); cout << endl;
    cout << "dist3 = "; dist3.showdist(); cout << endl;
    cout << "dist4 = "; dist4.showdist(); cout << endl;
}
```

Alternative Form (never used)
 dist3 = dist1.operator+(dist2)