Log book

Tempreture code, converting by implicit & explicit and cast

#include <iostream>

using namespace std;

// Declaration of class

class kelvin;

class fahrenheit;

class celsius {

protected:

double degrees;

public:

friend ostream & operator<<(ostream & os, const celsius & cel);

friend istream & operator>>(istream & is, celsius & cel);

celsius();

celsius(double);

celsius( const fahrenheit & );

celsius(const kelvin & );

double getDegrees()const ;

};

class kelvin {

protected:

double degrees;

public:

friend ostream & operator<<(ostream & os, const kelvin & kel);

friend istream & operator>>(istream & is, kelvin & kel);

kelvin();

kelvin(double val);

kelvin( celsius & val);

kelvin( fahrenheit & val);

double getkelvin()const;

};

class fahrenheit {

protected:

double degrees;

public:

friend ostream & operator<< (ostream &, const fahrenheit & fah );

friend istream & operator>> (istream &, fahrenheit & fah);

fahrenheit();

fahrenheit(double);

fahrenheit( const celsius & val);

fahrenheit(const kelvin & val);

double getfah()const ;

};

ostream & operator<<(ostream & os, const fahrenheit & fah ) {

os << fah.degrees << " degrees F";

return os;

}

istream & operator>> (istream & is, fahrenheit &fah) {

is >> fah.degrees;

return is;

}

fahrenheit::fahrenheit() { degrees =0.0; }

fahrenheit::fahrenheit(double val) { degrees=val; }

fahrenheit::fahrenheit(const celsius & val ) {

degrees = ( val.getDegrees() /5)\*9 +32;

}

fahrenheit::fahrenheit(const kelvin & val ){

degrees = ( ( val.getkelvin() - 273.15) \* 9/5 ) + 32;

}

double fahrenheit::getfah()const { return degrees;}

//////////////////////////////////////////////////////////////

ostream & operator<<(ostream & os, const celsius & cel) {

os << cel.degrees << " degrees C";

return os;

}

istream & operator>>(istream & is, celsius & cel) {

is >> cel.degrees;

return is;

}

celsius::celsius() {degrees=0.0;}

celsius::celsius(double val) { degrees=val; }

celsius::celsius( const fahrenheit & val ) {

degrees = ( val.getfah() -32)/9 \*5;

}

celsius::celsius(const kelvin & val) {

degrees = val.getkelvin() - 273.15;

}

double celsius::getDegrees()const {return degrees;}

///////////////////////////////////////////////////////

ostream & operator<<(ostream & os, const kelvin & kel) {

os << kel.degrees<< " degrees K";

return os;

}

istream & operator>> (istream & is, kelvin & kel) {

is >> kel.degrees;

return is;

}

kelvin::kelvin() {degrees =0.0; }

kelvin::kelvin(double val) { degrees = val; }

kelvin::kelvin( celsius & val ) {

degrees = val.getDegrees() + 273.15;

}

kelvin::kelvin( fahrenheit & val) {

degrees = ( ( val.getfah() - 273.15) \* 9/5 ) + 32;

}

double kelvin::getkelvin()const {return degrees;}

//////////////////////////////////////////////////////////////

void main (){

celsius cel(2.0);

cout << cel << " " <<endl ;

fahrenheit fah(2.0);

cout << fah << " " <<endl ;

kelvin Kelvin(3.0);

cout << Kelvin << " " << endl;

cout << "\nConversion by cast =>";

fahrenheit F2(3.0);

cout << F2 << " = " << (celsius) F2 << endl;

cout << "\nConversion by cast =>";

celsius C2(7.0);

cout << C2 << " = " << (fahrenheit) C2 << endl;

cout << "\nConversion by cast =>";

kelvin K2(7.0);

cout << K2 << " = " << (celsius) K2 << endl;

cout << "\nConversion by explicit constructor call => ";

fahrenheit F3(cel);

cout << F3 << " = " << cel << endl;

cout << "\nConversion by explicit constructor call => ";

kelvin K3(Kelvin);

cout << K3<< " = " << Kelvin << endl;

cout << "\nConversion by explicit constructor call => ";

celsius C4(fah);

cout << C4<< " = " << fah << endl;

cout << "\nConversion by implicit constructor call => ";

fahrenheit F4;

F4=C2;

cout << F4 << " = " << C2 << endl;

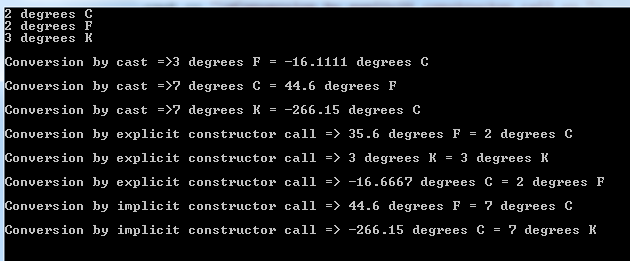
cout << "\nConversion by implicit constructor call => ";

celsius C5;

C5 = K2;

cout << C5 << " = " << K2 << endl;

}



Validation of the last three calculations:

