**COP 3331 Week 5 Example:**

1. Student Test Scores

Class File:

#ifndef STUDENTTESTSCORES\_H

#define STUDENTTESTSCORES\_H

#include <string>

using namespace std;

const double DEFAULT\_SCORE = 0.0;

class StudentTestScores

{

private:

string studentName; // The student's name

double \*testScores; // Points to array of test scores

int numTestScores; // Number of test scores

// Private member function to create an

// array of test scores.

void createTestScoresArray(int size)

{ numTestScores = size;

testScores = new double[size];

for (int i = 0; i < size; i++)

testScores[i] = DEFAULT\_SCORE; }

public:

// Constructor

StudentTestScores(string name, int numScores)

{ studentName = name;

createTestScoresArray(numScores); }

// Copy constructor

StudentTestScores(const StudentTestScores &obj)

{ studentName = obj.studentName;

numTestScores = obj. numTestScores;

testScores = new double[numTestScores];

for (int i = 0; i < numTestScores; i++)

testScores[i] = obj.testScores[i]; }

// Destructor

~StudentTestScores()

{ delete [] testScores; }

// The setTestScore function sets a specific

// test score's value.

void setTestScore(double score, int index)

{ testScores[index] = score; }

// Set the student's name.

void setStudentName(string name)

{ studentName = name; }

// Get the student's name.

string getStudentName() const

{ return studentName; }

// Get the number of test scores.

int getNumTestScores()

{ return numTestScores; }

// Get a specific test score.

double getTestScore(int index) const

{ return testScores[index]; }

// Overloaded = operator

const StudentTestScores operator=(const StudentTestScores &right)

{ delete [] testScores;

studentName = right.studentName;

numTestScores = right.numTestScores;

testScores = new double[numTestScores];

for (int i = 0; i < numTestScores; i++)

testScores[i] = right.testScores[i];

return \*this; }

};

#endif

Driver Program:

// This program demonstrates the overloaded = operator returning a value.

#include <iostream>

#include "StudentTestScores.h"

using namespace std;

// Function prototype

void displayStudent(StudentTestScores);

int main()

{

// Create a StudentTestScores object.

StudentTestScores student1("Kelly Thorton", 3);

student1.setTestScore(100.0, 0);

student1.setTestScore(95.0, 1);

student1.setTestScore(80, 2);

// Create two more StudentTestScores objects.

StudentTestScores student2("Jimmy Griffin", 5);

StudentTestScores student3("Kristen Lee", 10);

// Assign student1 to student2 and student3.

student3 = student2 = student1;

// Display the objects.

displayStudent(student1);

displayStudent(student2);

displayStudent(student3);

return 0;

}

// displayStudent function

void displayStudent(StudentTestScores s)

{

cout << "Name: " << s.getStudentName() << endl;

cout << "Test Scores: ";

for (int i = 0; i < s.getNumTestScores(); i++)

cout << s.getTestScore(i) << " ";

cout << endl;

}

1. Feet and Inches

Class File:

#ifndef FEETINCHES\_H

#define FEETINCHES\_H

// The FeetInches class holds distances or measurements

// expressed in feet and inches.

class FeetInches

{

private:

int feet; // To hold a number of feet

int inches; // To hold a number of inches

void simplify(); // Defined in FeetInches.cpp

public:

// Constructor

FeetInches(int f = 0, int i = 0)

{ feet = f;

inches = i;

simplify(); }

// Mutator functions

void setFeet(int f)

{ feet = f; }

void setInches(int i)

{ inches = i;

simplify(); }

// Accessor functions

int getFeet() const

{ return feet; }

int getInches() const

{ return inches; }

// Overloaded operator functions

FeetInches operator + (const FeetInches &); // Overloaded +

FeetInches operator - (const FeetInches &); // Overloaded -

};

#endif

Implementation File:

// Implementation file for the FeetInches class

#include <cstdlib> // Needed for abs()

#include "FeetInches.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Definition of member function simplify. This function checks for

values in the inches member greater than twelve or less than zero.

If such a value is found, the numbers in feet and inches are adjusted

to conform to a standard feet & inches expression. For example,

3 feet 14 inches would be adjusted to 4 feet 2 inches and

5 feet -2 inches would be adjusted to 4 feet 10 inches.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void FeetInches::simplify()

{

if (inches >= 12)

{

feet += (inches / 12);

inches = inches % 12;

}

else if (inches < 0)

{

feet -= ((abs(inches) / 12) + 1);

inches = 12 - (abs(inches) % 12);

}

}

// Overloaded binary + operator.

FeetInches FeetInches::operator + (const FeetInches &right)

{

FeetInches temp;

temp.inches = inches + right.inches;

temp.feet = feet + right.feet;

temp.simplify();

return temp;

}

// Overloaded binary - operator.

FeetInches FeetInches::operator - (const FeetInches &right)

{

FeetInches temp;

temp.inches = inches - right.inches;

temp.feet = feet - right.feet;

temp.simplify();

return temp;

}

Driver Program:

// This program demonstrates the FeetInches class's overloaded

// + and - operators.

#include <iostream>

#include "FeetInches.h"

using namespace std;

int main()

{

int feet, inches; // To hold input for feet and inches

// Create three FeetInches objects. The default arguments

// for the constructor will be used.

FeetInches first, second, third;

// Get a distance from the user.

cout << "Enter a distance in feet and inches: ";

cin >> feet >> inches;

// Store the distance in the first object.

first.setFeet(feet);

first.setInches(inches);

// Get another distance from the user.

cout << "Enter another distance in feet and inches: ";

cin >> feet >> inches;

// Store the distance in second.

second.setFeet(feet);

second.setInches(inches);

// Assign first + second to third.

third = first + second;

// Display the result.

cout << "first + second = ";

cout << third.getFeet() << " feet, ";

cout << third.getInches() << " inches.\n";

// Assign first - second to third.

third = first - second;

// Display the result.

cout << "first - second = ";

cout << third.getFeet() << " feet, ";

cout << third.getInches() << " inches.\n";

return 0;

}

1. Feet and Inches (Version 2)

Class File:

#ifndef FEETINCHES\_H

#define FEETINCHES\_H

// The FeetInches class holds distances or measurements

// expressed in feet and inches.

class FeetInches

{

private:

int feet; // To hold a number of feet

int inches; // To hold a number of inches

void simplify(); // Defined in FeetInches.cpp

public:

// Constructor

FeetInches(int f = 0, int i = 0)

{ feet = f;

inches = i;

simplify(); }

// Mutator functions

void setFeet(int f)

{ feet = f; }

void setInches(int i)

{ inches = i;

simplify(); }

// Accessor functions

int getFeet() const

{ return feet; }

int getInches() const

{ return inches; }

// Overloaded operator functions

FeetInches operator + (const FeetInches &); // Overloaded +

FeetInches operator - (const FeetInches &); // Overloaded -

FeetInches operator ++ (); // Prefix ++

FeetInches operator ++ (int); // Postfix ++

};

#endif

Implementation File:

// Implementation file for the FeetInches class

#include <cstdlib> // Needed for abs()

#include "FeetInches.h"

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Definition of member function simplify. This function \*

// checks for values in the inches member greater than \*

// twelve or less than zero. If such a value is found, \*

// the numbers in feet and inches are adjusted to conform \*

// to a standard feet & inches expression. For example, \*

// 3 feet 14 inches would be adjusted to 4 feet 2 inches and \*

// 5 feet -2 inches would be adjusted to 4 feet 10 inches. \*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void FeetInches::simplify()

{

if (inches >= 12)

{

feet += (inches / 12);

inches = inches % 12;

}

else if (inches < 0)

{

feet -= ((abs(inches) / 12) + 1);

inches = 12 - (abs(inches) % 12);

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Overloaded binary + operator. \*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

FeetInches FeetInches::operator + (const FeetInches &right)

{

FeetInches temp;

temp.inches = inches + right.inches;

temp.feet = feet + right.feet;

temp.simplify();

return temp;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Overloaded binary - operator. \*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

FeetInches FeetInches::operator - (const FeetInches &right)

{

FeetInches temp;

temp.inches = inches - right.inches;

temp.feet = feet - right.feet;

temp.simplify();

return temp;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Overloaded prefix ++ operator. Causes the inches member to \*

// be incremented. Returns the incremented object. \*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

FeetInches FeetInches::operator++()

{

++inches;

simplify();

return \*this;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Overloaded postfix ++ operator. Causes the inches member to \*

// be incremented. Returns the value of the object before the \*

// increment. \*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

FeetInches FeetInches::operator++(int)

{

FeetInches temp(feet, inches);

inches++;

simplify();

return temp;

}

Driver Program:

// This program demonstrates the FeetInches class' overloaded

// prefix and postfix ++ operators.

#include <iostream>

#include "FeetInches.h"

using namespace std;

int main()

{

int count; // Loop counter

// Define a FeetInches object with the default

// value of 0 feet, 0 inches.

FeetInches first;

// Define a FeetInches object with 1 foot 5 inches.

FeetInches second(1, 5);

// Use the prefix ++ operator.

cout << "Demonstrating prefix ++ operator.\n";

for (count = 0; count < 12; count++)

{

first = ++second;

cout << "first: " << first.getFeet() << " feet, ";

cout << first.getInches() << " inches. ";

cout << "second: " << second.getFeet() << " feet, ";

cout << second.getInches() << " inches.\n";

}

// Use the postfix ++ operator.

cout << "\nDemonstrating postfix ++ operator.\n";

for (count = 0; count < 12; count++)

{

first = second++;

cout << "first: " << first.getFeet() << " feet, ";

cout << first.getInches() << " inches. ";

cout << "second: " << second.getFeet() << " feet, ";

cout << second.getInches() << " inches.\n";

}

return 0;

}