

```

// Name: James Small
// Program: 4A
// Class: CSE455
// Description: LinearRegression class implementation file

#include "LinearRegression.h"
#include <fstream>
#include <iostream>

// Constructor that takes in both file names
LinearRegression::LinearRegression()
{
    fileNamesGood = true;
    b0Value = 0;
    b1Value = 0;

    cout << "Enter the x-axis values filename: ";
    cin >> fileName1;

    while (!fileCheck.fileExist(fileName1))
        if (!enterNewFileName(fileName1)) {
            fileNamesGood = false;
            return;
        }

    cout << "\nEnter the y-axis values filename: ";
    cin >> fileName2;

    while (!fileCheck.fileExist(fileName2))
        if (!enterNewFileName(fileName2)) {
            fileNamesGood = false;
            return;
        }

    calculate();
}

// This method returns true if the file names were good
bool LinearRegression::getGood()
{
    return fileNamesGood;
}

// This method calculates the linear regression
void LinearRegression::calculate()
{
    readInValues(fileName1, vector1);
    readInValues(fileName2, vector2);
}

```

```

float topValue = 0;
float bottomValue = 0;
float topValueLeft = 0;
float topValueRight = 0;
float bottomValueLeft = 0;
float bottomValueRight = 0;

topValueLeft = sumValues(multiplyValues(vector1,vector2));
topValueRight = vector1.size() * averageOfVector(vector1) *
averageOfVector(vector2);
topValue = topValueLeft - topValueRight;

bottomValueLeft = sumValues(multiplyValues(vector1,vector1));
bottomValueRight = vector1.size() * averageOfVector(vector1)
* averageOfVector(vector1);
bottomValue = bottomValueLeft - bottomValueRight;

b1Value = topValue / bottomValue;
b0Value = averageOfVector(vector2) - b1Value *
averageOfVector(vector1);
}

// This method reads the numbers from a file into a vector

void LinearRegression::readInValues(string filename, vector
<float> &vector)
{
    ifstream infile;

    infile.open(filename.c_str());

    float currentValue = 0;

    while (!infile.eof()) {
        infile >> currentValue;
        vector.push_back(currentValue);
    }

    infile.close();
}

// This method calculates the average value of the vector

float LinearRegression::averageOfVector(vector<float> vector)
{
    return sumValues(vector) / vector.size();
}

// This method multiplies parallel vectors and returns a vector as
result

vector<float> LinearRegression::multiplyValues(vector<float>

```

```

vector1, vector<float> vector2)
{
    vector<float> multiplyVector;

    if (vector1.size() != vector2.size())
        return multiplyVector;

    for (int i = 0; i < vector1.size(); i++)
        multiplyVector.push_back(vector1[i] * vector2[i]);

    return multiplyVector;
}

// This method displays a report of the results

void LinearRegression::report()
{
    cout << "\nB0 = " << b0Value << endl;
    cout << "B1 = " << b1Value << endl << endl;
}

// This method sums all values in the vector

float LinearRegression::sumValues(vector<float> vector)
{
    float sum = 0;

    for (int i = 0; i < vector.size(); i++)
        sum += vector[i];

    return sum;
}

// This method asks the user to enter a new filename

bool LinearRegression::enterNewFileName(string &fileName)
{
    cout << "\nThe filename doesn't exist\n";

    char choice = 0;
    bool choiceGood = false;

    do {
        cout << "What would you like to enter a new filename?\n";
        cout << "Enter 1 to enter another filename.\n";
        cout << "Enter 0 to quit.\n";
        cout << "Choice: ";

        cin >> choice;

        if (isdigit(choice)) {
            if (atoi(&choice) >= 0 && atoi(&choice) < 2)

```

```

        choiceGood = true;
    else
        cout << "\nInvalid Choice, Try again\n\n";
    } else
        cout << "\nInvalid Choice, Try again\n\n";

    cin.ignore(INT_MAX, '\n');

} while (!choiceGood);

if (choice == '1') {
    cout << "Enter the file name to access: ";
    cin >> fileName;
    return true;
} else
    return false;
}

```