Retirement Calculator: developed by Axcess Enterprise,

Programmed by: Brayton Huckstep, and Jace Regal

Debbugged by: Jace Regal

GitHub Repository: <https://github.com/Jregal2023/RetirementCalculator>

main.cpp

#include <iostream>

#include <cmath> // For pow function

#include <iomanip> // For formatting output

#include “tools.h””

Using namespace std;

// Function prototypes

void getUserInput(double &annualContribution, double &interestRate, int &yearsToRetirement, int &yearsDuringRetirement);

double calculateFutureValue(double annualContribution, double interestRate, int yearsToRetirement);

double calculateMonthlyRate(double futureValue, int yearsDuringRetirement);

void displayResults(double futureValue, double monthlyRate);

void testApplication();

// Main function

/\* main function here is meant to run the code it uses\*/

int main() {

// Variables to hold user inputs

double annualContribution = 0.0; //used on line 16

double interestRate = 0.0;

int yearsToRetirement = 0;

int yearsDuringRetirement = 0;

// Welcome message

cout << "Welcome to the Retirement Income Calculator!\n";

// Get user input

getUserInput(annualContribution, interestRate, yearsToRetirement, yearsDuringRetirement);

// Calculate future value of the fund

double futureValue = calculateFutureValue(annualContribution, interestRate, yearsToRetirement);

// Calculate monthly withdrawal rate

double monthlyRate = calculateMonthlyRate(futureValue, yearsDuringRetirement);

// Display the results

displayResults(futureValue, monthlyRate);

// Optional: Test application with sample inputs

cout << "\nRunning automated tests...\n";

testApplication();

return 0;

}

tools.h

/\*Tools.h is meant to create classes with functions in them to make the code more readable.\*/

#ifndef TOOLS\_H

#define TOOLS\_H

#include <iostream>

#include <cmath>

#include <iomanip>

/\*This file is used to define the classes that will be used in main.cpp and tools.cpp. This helps the program

maintain cleanliness, and makes it more readable. The program originally didn't need it, however this makes the program

more efficient than before.\*/

//This class get the user input that will then get the values which will be used by RetirementCalculator

class UserInput

{

public:

void getUserInput(double &annualContribution, double &interestRate, int &yearsToRetirement, int &yearsDuringRetirement);

double calculateFutureValue(double annualContribution, double interestRate, int yearsToRetirement);

};

/\*This holds all the functions from the previous code, in which it helps make the code cleaner, and efficient.

Additionally,\*/

class RetirementCalculator

{

public:

double calculateFutureValue(double annualContribution, double interestRate, int yearsToRetirement);

double calculateMonthlyRate(double futureValue, int yearsDuringRetirement);

void displayResults(double futureValue, double monthlyRate);

void testApplication();

};

#endif

Tools.cpp:

#include <iostream>

#include <cmath> // For pow function

#include <iomanip> // For formatting output

#include "tools.h"

using namespace std;

/\*Using tools.h, tools.cpp defines the functions within tools.h.

Thankfully we already have the functions pre-defined, due to the original code.

The only difference is that it now derives off of classes, and not the main function.

\*/

// Define UserInput class member function

void UserInput::getUserInput(double &annualContribution, double &interestRate, int &yearsToRetirement, int &yearsDuringRetirement) {

cout << "Enter annual contribution amount (in dollars): ";

cin >> annualContribution;

cout << "Enter expected annual interest rate (as a percentage): ";

cin >> interestRate;

interestRate /= 100.0; // Convert percentage to decimal

cout << "Enter number of years until retirement: ";

cin >> yearsToRetirement;

cout << "Enter number of years during retirement: ";

cin >> yearsDuringRetirement;

}

// Calculates the future value of your input

double RetirementCalculator::calculateFutureValue(double annualContribution, double interestRate, int yearsToRetirement) {

if (interestRate == 0) {

return annualContribution \* yearsToRetirement;

}

return annualContribution \* (pow(1 + interestRate, yearsToRetirement) - 1) / interestRate;

}

// Calculates the monthly rate using futureValue, and yearsDuringRetirement

double RetirementCalculator::calculateMonthlyRate(double futureValue, int yearsDuringRetirement) {

if (yearsDuringRetirement == 0) {

return 0.0; // Avoid division by zero

}

return futureValue / (yearsDuringRetirement \* 12);

}

//displays the results of the future value.

void RetirementCalculator::displayResults(double futureValue, double monthlyRate) {

cout << fixed << setprecision(2); // Format output to 2 decimal places

cout << "\n=== Retirement Income Summary ===\n";

cout << "Future value of the fund: $" << futureValue << "\n";

cout << "Monthly withdrawal amount during retirement: $" << monthlyRate << "\n";

cout << "==================================\n";

}

/\*This function was used to test the program and display results.

Although it is useful, it has no genuine purpose but to display that the functions work.\*/

void RetirementCalculator::testApplication() {

double annualContributionArray[3] = {5000.0, 0.0, 10000.0};

double interestRateArray[3] = {5.0 / 100.0, 4.0 / 100.0, 0.0}; // ran at

int yearsToRetirementArray[3] = {30, 25, 20}; //declared at line 64

int yearsDuringRetirementArray[3] = {20, 15, 10}; //declared at line 65

for (int i = 0; i < 3; ++i) {

double futureValue = calculateFutureValue(annualContributionArray[i], interestRateArray[i], yearsToRetirementArray[i]);

double monthlyRate = calculateMonthlyRate(futureValue, yearsDuringRetirementArray[i]);

cout << "Test Case " << i + 1 << ": Future Value = $" << futureValue << ", Monthly Rate = $" << monthlyRate << "\n";

}

}

END OF SOURCE CODE

Screenshots: The screenshots taken were used for validation and testing, this ensure th

Screenshot 1( checks the code for how it works, the validation testing is printed after the user has imputed their values.)

