MSF 575: C++ with Financial Applications

PROJECTS

1.) Create a program that asks the user to enter 4 numbers: **a**, **b**, **c**, **d**. Next, if (a, b) and (c, d) are ordered pairs, calculate the slope of the line connecting the two points according to the formula:

$$m = (b - d) / (a - c)$$

Print out m. Next, create references: **aRef**, **bRef**, **cRef** and **dRef**. Calculate the value of the slope again so that:

$$m = (bRef - dRef) / (aRef + cRef)$$

Print out the values and addresses to prove that *a* and *aRef* are the same location and likewise for *b* and *bRef*, *c* and *cRef*. Lastly, create five pointers: **aPtr, bPtr, cPtr, dPtr** and **mPtr**, that point to a, b, c, d and m. Calculate and print out the value of the slope again using only dereferenced pointers.

2.) Create a program that calculates a time period using the rule of 72.

Input:	Annual interest rate
Output:	Years.

3.) Create a program that computes the mix of coins returned by a vending machine.

Input:	Amount of money put into the machine
	Cost of purchase
Output:	Total change returned
	Count of each coin returned.

4.) Given a one-dimensional array with elements:

1, 2, 3, 4, 4, 2, 3, 1, 2, 1, 1, 3, 4, 2, 2,
2, 1, 4, 1, 2, 1, 3, 4, 2, 3, 4, 1, 1, 1, 2

Create a program that will ***calculate*** the mean and the sum of the elements, and find the minimum and maximum values.

5.) According to the definition of the Fibonacci series:

$$\text{Fibonacci}(n) = \text{Fibonacci}(n - 1) + \text{Fibonacci}(n-2)$$

Create a program that will place the first 100 elements in the Fibonacci series into a one dimensional array and print them out.

6.) Create a C++ program where the user tries to guess a letter in memory. If the user guesses the letter within four tries, the user wins. Given each guess, the computer should tell the user whether the letter in memory is higher or lower in the alphabet. (Hint: Casting a character as an int will return its ASCII character code.)

7.) Given the following two-dimensional arrays (i.e. matrices):

$$\begin{matrix} 1 & 2 & 7 & 6 \\ 8 & 5 & 6 & 4 \\ 5 & 1 & 4 & 9 \end{matrix} \qquad\qquad \begin{matrix} 3 & 4 & 5 & 2 \\ 1 & 7 & 9 & 3 \\ 5 & 3 & 5 & 4 \end{matrix}$$

Create a program that will add the two matrices together and print out the solution array.

8.) Create a Point structure. Each structure should have an x and y value. Create a function that returns the distance between two Points.

9.) Create an array of 10 numbers. Also, create a function called ArrayMax that will return the highest value in the array. Then print that value to the screen.

10.) Create a function that adds two numbers together. This function should accept two pointers to doubles and return a pointer to a double.

11.) Create a two functions (overloaded), both named NumCompare that accept two pointers to integer and doubles respectively. If the two numbers are the same, the function should return true. If the two numbers are different, the function should return false.

12.) Create function that returns the index number of a particular character in an array. If the character is not in the array, it should return 999.

13.) Create a function that accepts a pointer to a double as well as an array of doubles. If one of the elements of the array is the number, return true, otherwise, return false.

14.) Create a program that calls a function that returns the ID number of the student with the highest GPA. (Hint: Create an array of Student structures.)

15.) Create three functions that illustrate the pass-by-value, pass-by-reference, and pass-by-pointer methods. These functions should use the simple interest formula to calculate the future value of an asset and return the value by-value, by-reference and by-pointer respectively.

16.) Create a function that accepts a pointer to a character (i.e. an array of chars or a string), loops through the array and counts the number of elements in the array. Then, the function should return the number of elements as a reference to an integer.

17.) Create a function that accepts a pointer to a function as an input argument. Call this function from within the function.

18.) Create a Stock class that contains a char * member called Symbol, and a double price data member. Add a constructor and a destructor. Require that the symbol be set via the constructor. Include gets and sets for the data members as appropriate. Create both a value-type and reference-type instance of the Stock class.

19.) Create a function that accepts a reference (i.e. pass by reference) to your value-type instance of the Stock class. Within the function call the objects get_symbol() method.

20.) Create a function that accepts a pointer (i.e. pass by pointer) to your reference-type instance of the Stock class. Again, call the get_symbol() method from within the function.

21.) Create a Person object. First, create a Person class with data members and methods of your own choosing. Second, create an instance of your Person class.

22.) Create a structure called Performance, with double members High, Average, Low. Next, create a class called Strategy that has as a data member an instance of the Performance structure called m_Performance. Add a get_Performance() method to the class. Via the constructor, set the value of the Strategy's Performance. Create an instance of the Strategy class and print out its Performance.

23.) Create a Bond class and a MuniBond class that inherits from Bond. Create a simple program and create an instance of the MuniBond class as a reference type.

24.) Create a Stock class. Also, create a Preferred Stock class that inherits from both the Stock class and the Bond class. Create a simple program that creates a value type instance of the Preferred Stock class.

25.) Create an abstract class (i.e. one that contains a pure virtual function) called ITradable. This class should prototype functions to SendOrder() and CancelOrder(). Now, amend your classes from problems 1 and 2 to all inherit from the base ITradable class. That is, make all your classes implement the ITradable interface.

26.) Create a program that asks the user to enter the price of an option, its delta and its gamma. Add the price of the underlying stock. If the price of the stock moves, use Taylor's theorem to estimate the change in the price of the option.

27.) Use the matrix class to solve a system of linear equations.

28.) Use the matrix class to estimate the variance of a portfolio of stocks.

29.) Value a portfolio of three options.

30.) Given price data in .txt files for a stock and for the S&P 500, build a model to calculate the Beta of the stock.

31.) Given three bonds, calculate the portfolio duration and convexity.

32.) Calculate the ATM volatility using Whaley's method.

33.) Create a static library to hold the following classes from previous homeworks:

1. Bond and MuniBond
2. Instrument
3. SystemManager

Also, add a smart_ptr class to your library. Prove that your library works by creating a separate executable that instantiates each of these classes and tests their methods.

34.) Create a Random class that implements the linear congruent generator algorithm. I suggest Googling: *linear congruent generator C++*. Prove that your random number generator works by writing values to a .csv file and then opening that file in Excel. If you use the Tools->Data Analysis->Histogram wizard to get a look at the distribution of the data.

35.) For those of you are looking for extra things to do, try downloading and installing boost and QuantLib following the directions in the book. There is plenty of sample code on the QuantLib website to look into.

36.) Use the string class and the ofstream class to stream out information about a stock object to a file named StockInfo.dat. This information should be the name of the issuer, the price of the stock and the dividend yield. Use the string class and the ifstream class to stream in information about a stock. This information should be used to construct a new stock object.

37.) Create a Futures class and a Spread class. Use operator overloading to construct a Spread as being equal to the sum of two Futures objects. The Spread object should contain two pointers to the two Futures that comprise the legs of the Spread.

38.) Create an StockOption class. Create a hash_map of StockOptions. In the main, find the portfolio delta.

39.) Create an hours worth of random tick data—time interval (exponential distribution), price (Bernoulli trials using Mersenne Twister), and volume (Empirical). Next, create a time series of 60 bars—open, high, low, close, and total volume. Write your bar data to a file and graph your bars in Excel.

40.) In a C++ application, calculate the Betas of 3 stocks to an index using data from a file. Add to your program three stocks in a portfolio. Using the Betas and random number generation for the returns of the index, calculate a random portfolio price path for 10 days.