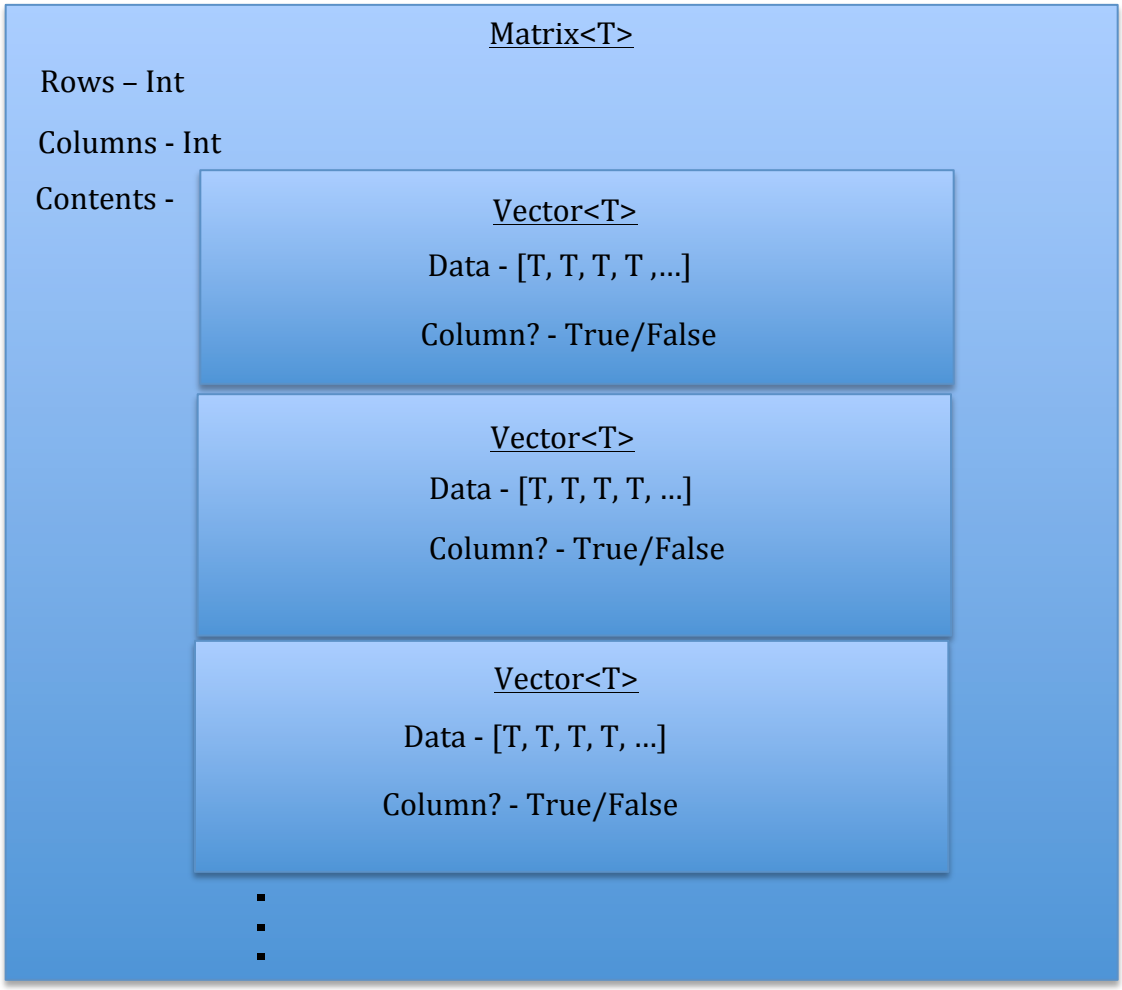
Benjaman Dutton ID: 247060

Assignment 1 report

Matrix and Vector Library –

As a matrix is just a series of vector objects I chose to have the matrix object store references to vector objects to represent each row of the matrix. Each vector object contains the data. Having matrix objects composed of vector objects makes it so the vectors do all the work with the individual data values and all the matrices need to do it specify the ordering for the vectors do this work. All of the matrix arithmetic is done by arranging vector objects to match the logic of the operation and then calling the maths functions on the vector objects.

Here is a diagram of the library and how the matrices are composed of vectors, which store the values.



Benjaman Dutton ID: 247060

Matrix<T> -

Each matrix object is made up of a series of vector objects. Calling the initializers for Matrix<T> sets 3 data fields within the object. One for the number of rows, one for the number of columns and the third is an array of Vector<T> objects which store the data for each row.

The description function for matrix calls the description method of its constituent vector objects and prints them out line by line. Each row starts on a new line and there is no new line after the last row.

The transpose function creates a series of vectors with the transposed data and creates a new matrix object with them.

Vector view works by checking first is there is one row or one column and then returns either the vector stored in the matrix objects 'contents' data field or if the matrix has one column it creates a new vector using each value stored in each of the vector objects and calls an initializer that sets the new vector objects 'iscol' data field to true in order to store that it is a column.

The row function simply returns the vector at the corresponding position in the matrix objects 'contents' field. Changing this vector will alter the vector in the original matrix too, so to get a new vector use the copy() method first.

The column function creates a new vector using each value stored at the index position of each vector object in the matrix.

Subscript uses the first index to select the vector representing the row, the second index is used in a call to vector's subscript function which is called on the selected vector.

I made a secondary subscript function for matrix that takes a single index and calls the row function on itself to conveniently reference a vector contained in the matrix.

Copy creates a new array and fills it with copies of the originals vector objects by calling the copy method each of the vectors.

All of the maths function that matrix implements uses calls to the maths functions for the vectors so the vector object does all of the logic for adding individual values. All the matrix function needs to do is call these function in the right order. The + function for adding two matrices calls the + function for each vector object of both matrices and it does this for each row in the matrix.

Benjaman Dutton ID: 247060

Vector<T> -

The vector class has two data fields. One is an array of values of type T the second is a boolean which represents if the vector is supposed to be a column, this is false by default. I made two extra initializers, one makes it convenient for creating a vector representing a column and takes as parameters the numbers to store and a boolean representing if its a column. The other initializer creates a vector with the values specified at creation rather than just creating a zero value vector object.

The description function prints out each value with a space between each and no space at either end.

Subscript gets the value stored at the specified index in the numbers array. It sets the value with the same kind of logic.

Size calls the count method on the array of data values and returns it.

The matrixview function checks if the vector is a column or not, if not it just returns a matrix with a single row containing itself. If it is a column it creates a matrix with a series of vector objects each with one value.

Copy makes each value into a new generic and assigns it to a new numbers array which is then passed into the constructor for a new Vector<T>.

Dot uses the * and + methods provided by the matrix data values and does this to conform to the logic of the dot product operation.

The maths functions in the vector object do all the arithmetic between each data value.

Benjaman Dutton ID: 247060

TestMatrixVector –

My test class is made up of a series of functions. Each specify a certain test. I've made it so that the main method creates an instance of the class and then just calls each test function on that class object. The reason for having an instance of the class and calling functions on that instance rather than just having static functions is that a variable of the class can be declared with a short variable name so you don't need to type out TestMatrixVector.test…() and can instead type something like t.test…().

The way the test functions work is that the user defines a string which represents the expected output of the program. The string representation of a vector has one space in between each value with no space before or after and no other characters. The matrix is represented with call to each of its constituent vector rows separated by a newline character (\n) with no new line before or after. The tests then create a matrix/vector object with test values. The matrix/vector object is then altered with calls to its functions. The string representation of the matrix is then compared to the expected string specified by the user.
The comparison is carried out by a function called cc (check correctness) which takes the expected string and the string of the matrix/vector created from their description methods and compares them. If the strings are the same the function returns true. If they are different the function prints out the expected and actual matrix/vector and returns false.
The test functions then return the result of this test and the main program uses a simple function to print out pass or fail based on whether the tests return true or false.

Functions-

Init() – initializes the class object.

There are multiple test functions with the same generic layout that is like this .

test() -> Bool – The test functions are called on the class object from the main file. Each function returns a Boolean, if the test passes it returns true, if it doesn't it returns false.

cc(e: String, m: String) -> Bool – this function takes two strings one that represents the expected output, the other string from calling the description method of the matrix. If the two strings match the function returns true. If they don't, both strings are printed out and the function returns false.