

Maths Library Report

by David James

FILE	4073_DAVID_JAMES_MATHS_LIBRARY_REPORT_112721_1692561855.D OCX		
TIME SUBMITTED	25-APR-2018 11:05PM (UTC+0100)	WORD COUNT	791
SUBMISSION ID	87484936	CHARACTER COUNT	5186

Maths for Software Development.

Student Name: David James

Student ID: 1704050

Lecturer: Gordon Dickers

Module No: M4X01015

Table of Contents

Introduction.....	3
1. Vectors.....	4
1.1. Cross Product.....	4
1.2. Dot Product.....	6
1.3. Length.....	7
1.4. Distance.....	8
1.5. Normalize.....	9
2. Matrices.....	10
2.1. Multiply 4x4 by 4x1 Matrix.....	10
3. Sets.....	13

Introduction

As outlined in the brief the task is to create a maths library that performs mathematical operations in the following topics:

- Vectors
 - Cross Product
 - Dot Product
 - Length
 - Distance
 - Normalize
- Matrices
 - Multiply 4x4 by 4x1
- Sets
 - Mean
 - Mode
 - Median
 - Standard Deviation
 - Union
 - Intersection

Each subject has its own class and each mathematical operation is a method with the class. The decision to do this meant that each class could also be used as a user defined data type, this simplified the approach of programming the methods as it gave a lot of control when needing to return results from calculations.

For usability a menu class was also created which provided a way to program functions that could create user defined types to leverage the methods from each class.

1. Vectors.

1.1. Cross Product.

The function for the cross-product can be seen in Fig 1. This is within the menu class and allows the user to input the required data that is needed to perform the calculations required to retrieve the cross product.

```
152 void FindCrossProduct()
153 {
154     Vector vector;
155
156     Vector a;
157     Vector b;
158
159     WriteText("Please enter the coordinates of your first vector\r\n");
160     a = ReadVector();
161     a.Length();
162     WriteText("\r\n");
163
164     WriteText("Please enter the coordinates of your second vector\r\n");
165     b = ReadVector();
166     b.Length();
167     WriteText("\r\n");
168
169     vector = vector.CrossProduct(a, b);
170
171     printf_s("The cross product of the two vectors is: x: %0.2lf, y: %0.2lf, z: %0.2lf\r\n", vector.x, vector.y, vector.z);
172
173     WriteText("\r\n");
174 }
```

Fig 1.

Two Vectors variables are created, the read vector function allows the user to provide coordinates for each axis as seen in Fig 2.

```
23 Vector ReadVector()
24 {
25     Vector vector;
26
27     WriteText("Enter x value: ");
28     scanf_s("%lf", &vector.x);
29
30     WriteText("Enter y value: ");
31     scanf_s("%lf", &vector.y);
32
33     WriteText("Enter z value: ");
34     scanf_s("%lf", &vector.z);
35
36     return vector;
37 }
```

Fig 2.

The length method of the vector class is then called to perform its calculations (Fig 3).

```
45 double Vector::Length()  
46 {  
47     // Each coordinate is squared and added with the final result as the square root  
48     _magnitude = sqrt((x * x) + (y * y) + (z * z));  
49  
50     return GetLength();  
51 }
```

Fig 3

This is repeated for the second vector variable. Once the length of both vectors has been found the cross-product method is then called on the variable vector this can be seen in Fig 4.

```
74 Vector Vector::CrossProduct(Vector a, Vector b)  
75 {  
76     Vector c;  
77  
78     // Multiply both vectors coordinates then subtract the result from each other  
79     c.x = (a.y * b.z) - (a.z * b.y);  
80     c.y = (a.z * b.x) - (a.x * b.z);  
81     c.z = (a.x * b.y) - (a.y * b.x);  
82  
83     return c;  
84 }
```

Fig 4.

The method takes two arguments which are of type vector, the two vectors that were created by the user in the cross-product function within the menu class are passed as the arguments. Within the method a new vector variable is created, and its members are used to store the result of the calculations.

The results are returned to the vector variable within the cross-product function from the menu class which is then displayed on the console window (see Fig. 5).

```
Please enter the coordinates of your first vector  
Enter x value: 3  
Enter y value: -1  
Enter z value: 5  
  
Please enter the coordinates of your second vector  
Enter x value: -4  
Enter y value: 0  
Enter z value: 2  
  
The cross product of the two vectors is: x: -2.00, y: -26.00, z: -4.00
```

Fig 5.

1.2. Dot Product

The dot product function within the menu class can be seen in Fig. 6

```
128 void FindDotProduct()
129 {
130     Vector vector;
131
132     Vector a;
133     Vector b;
134
135     WriteText("Please enter the length of your first vector\r\n");
136     a = ReadVector();
137     a.Length();
138     WriteText("\r\n");
139
140     WriteText("Please enter the length of your second vector\r\n");
141     b = ReadVector();
142     b.Length();
143     WriteText("\r\n");
144
145     vector.DotProduct(a, b);
146
147     printf_s("The dot product of the two vectors is: %0.21f\r\n", vector.GetDotProduct());
148
149     WriteText("\r\n");
150 }
```

Fig. 6

Just like the cross-product function two variables of type vector are created in order for the user to supply the values for each vectors coordinates.

The dot product method is then called as seen in Fig. 7.

```
66 double Vector::DotProduct(Vector a, Vector b)
67 {
68     // Multiply both vectors coordinates and add them together
69     _dotProduct = (a.x * b.x) + (a.y * b.y) + (a.z * b.z);
70
71     return GetDotProduct();
72 }
```

Fig. 7

1.3. Length

The function for finding the length of a vector can be seen in Fig. 8

```
94 void LengthOfVector()  
95 {  
96     Vector vector;  
97  
98     double magnitude = 0.0;  
99  
100    vector = ReadVector();  
101    WriteText("\r\n");  
102  
103    magnitude = vector.Length();  
104  
105    printf_s("The length of the vector is: %0.2lf\r\n", magnitude);  
106  
107    WriteText("\r\n");  
108 }
```

Fig. 8

The length method is called which returns the result of the calculation and assigns it to the variable magnitude.

The results are then printed to the console window (see Fig. 9)

```
Please select an option: 1  
  
Enter x value: 4  
Enter y value: 1  
Enter z value: 0  
  
The length of the vector is: 4.12
```

Fig. 9

1.4. Distance

The function for distance can be seen in Fig. 10

```
176 void DistanceBetweenVectors()
177 {
178     Vector vector;
179
180     Vector a;
181     Vector b;
182
183     double distance = 0.0;
184
185     WriteText("Please enter the coordinates of your first vector\r\n");
186     a = ReadVector();
187     WriteText("\r\n");
188
189     WriteText("Please enter the coordinates of your second vector\r\n");
190     b = ReadVector();
191     WriteText("\r\n");
192
193     distance = vector.Distance(a, b);
194
195     printf_s("The distance between the two points is: %0.2lf\r\n", distance);
196
197     WriteText("\r\n");
198 }
```

Fig. 10

Following the same structure as the previous functions the user provides the values for both vectors and a variable distance is created to store the returned value from the distance method.

The distance method can be seen in Fig. 11

```
86 double Vector::Distance(Vector a, Vector b)
87 {
88     // Subtract each vectors coordinates then multiply the results together
89     _distance = sqrt((b.x - a.x) * (b.x - a.x) + (b.y - a.y) * (b.y - a.y) + (b.z - a.z) * (b.z - a.z));
90
91     return GetDistance();
92 }
```

Fig. 11

1.5. Normalize

The function for normalizing a vector can be seen in Fig. 12

```
110 void NormalizeVector()
111 {
112     Vector vector;
113
114     double magnitude = 0.0;
115
116     vector = ReadVector();
117     WriteText("\r\n");
118
119     magnitude = vector.Length();
120
121     vector = vector.Normalize(magnitude);
122
123     printf_s("The normalized vector is: x: %0.2lf, y: %0.2lf, z: %0.2lf\r\n", vector.x, vector.y, vector.z);
124
125     WriteText("\r\n");
126 }
```

Fig. 12

To normalize a vector, the length of a vector is needed. The length method is called and returns the value into the variable magnitude. The magnitude variable is passed as an argument into the normalize method which can be seen in Fig. 13.

```
53 Vector Vector::Normalize(double magnitude)
54 {
55     // Find the length of the vector
56     magnitude = Length();
57
58     // Divide each coordinate by the length of the vector
59     x = x / GetLength();
60     y = y / GetLength();
61     z = z / GetLength();
62
63     return Vector(x, y, z);
64 }
```

Fig. 13

The magnitude variable that is passed as an argument is used within the method to store the returned value from the length method. The getter for the length attribute of the vector class is called on each coordinate. A constructor is used to return a new vector with the coordinates from the calculations.

The result is displayed in the console window (see Fig. 14)

```
Enter x value: 1
Enter y value: 4
Enter z value: -2

The normalized vector is: x: 0.22, y: 0.87, z: -0.44
```

Fig. 14

2. Matrices

2.1. Multiply 4x4 by 4x1 Matrix

There are three functions within the menu class that allow the user to input values for each element of each matrix. Two read matrix functions can be seen in Fig. 15 and Fig. 16.

```
39 Matrix Read4x4Matrix()
40 {
41     Matrix matrix;
42
43     int counter = 1;
44     int rows = 4;
45     int columns = 4;
46
47     double temp = 0.0;
48
49     WriteText("Enter values for the 4x4 matrix\r\n");
50     WriteText("\r\n");
51
52     for (int i = 0; i < rows; i++)
53     {
54         for (int j = 0; j < columns; j++)
55         {
56             printf_s("Please enter the %d element of the matrix: ", counter);
57             scanf_s("%lf", &matrix.matrix4x4[i][j]);
58
59             counter++;
60         }
61     }
62
63     WriteText("\r\n");
64
65     return matrix;
66 }
```

Fig. 15

```

68  Matrix Read4x1Matrix()
69  {
70      Matrix matrix;
71
72      int counter = 1;
73      int rows = 4;
74      int columns = 4;
75
76      double temp = 0.0;
77
78      WriteText("Enter values for the 4x1 matrix\r\n");
79      WriteText("\r\n");
80
81      for (int i = 0; i < rows; i++)
82      {
83          printf_s("Please enter the %d element of the matrix: ", counter);
84          scanf_s("%lf", &matrix.matrix4x1[i]);
85
86          counter++;
87      }
88
89      WriteText("\r\n");
90
91      return matrix;
92  }

```

Fig. 16

Both functions return a matrix which the user has input values for. The multiply matrix function utilises the matrices that are returned from the read functions, the multiply matrix function can be seen in Fig. 17

```
200 void MultiplyMatrices()
201 {
202     Matrix matrix;
203
204     Matrix a;
205     Matrix b;
206
207     int rows = 4;
208     int columns = 4;
209
210     double temp = 0.0;
211
212     a = Read4x4Matrix();
213     b = Read4x1Matrix();
214
215     for (int i = 0; i < rows; i++)
216     {
217         for (int j = 0; j < columns; j++)
218         {
219             temp = temp + (a.matrix4x4[i][j] * b.matrix4x1[j]);
220         }
221
222         printf_s("%0.21f\n", matrix.matrix4x1[i] = temp);
223
224         temp = 0.0;
225     }
226
227     WriteText("\r\n");
228 }
```

Fig. 17



3. Sets

Due to time limitations sets have not been implemented.



Maths Library Report

GRADEMARK REPORT

FINAL GRADE

GENERAL COMMENTS

Instructor

41 /100

PAGE 1

PAGE 2

PAGE 3



Comment 1

put mean median and mode in a separate statistics section

PAGE 4

PAGE 5

PAGE 6



Comment 2

no need for this

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12



Comment 3

Test results/screen shots for this please.

PAGE 13



Comment 4

I assume you mean the stats problems too.

DAVID JAMES

41

RESEARCH-INFORMED LITERATURE

-  Evidence of investigation into requisite concepts in evidence but several tasks not completed. No references! Include refs. please. **2**

KNOWLEDGE AND UNDERSTANDING

-  For those tasks completed there is a clear understanding of the underlying principles and methods. **6**

ANALYSIS

-  For those tasks completed you have shown appropriate analytical skills necessary to interpret the problem and synthesise a solution and to also generate valid test schedules. **7**

PRACTICAL APPLICATION AND DEP.

-  Not all tasks completed but you have shown some essential SE and maths practices. **24**

SKILLS FOR PROFESSIONAL PRAC.



Please include references in your work. Report structured - good. Testing in evidence -good. Some section missing.

2