

Application of Maths for Software Development

by Dominik Kania

FILE	16923_DOMINIK_KANIA_APPLICATION_OF_MATHS_FOR_SOFTWARE_DEVELOPMENT_250776_913920366.DOCX (1.29M)		
TIME SUBMITTED	24-APR-2020 02:57PM (UTC+0100)	WORD COUNT	2643
SUBMISSION ID	125003695	CHARACTER COUNT	11845

Application of Maths for Software Development

Prepared by: Dominik Tomasz Kania

Student Number: 1702026

Preparation for University of Wales Trinity Saint David

Lecturer: Gordon Dickers

Module: Mathematics (ACDA4003)

Course: Computing (Games Development)

7th April 2020

Table of Contents

Vector length.....	3
Vector normalization	6
Distance between the two points.....	8
Dot product of the two vectors.....	12
Cross product of the two vectors	15
Rotation about the y axis transformation	17
Mean, mode and standard deviation	20
Union and intersection of the two sets.....	23

Vector length

Length of a vector can be described as the length of the directed segment which determines the numerical value of the vector. [1]

Formula for the two-dimensional vector length:

$$|a| = \sqrt{a_x^2 + a_y^2}$$

Formula for the three-dimensional vector length:

$$|a| = \sqrt{a_x^2 + a_y^2 + a_z^2}$$

Formula for the arbitrary dimensions vector length:

$$|a| = \left(\sum_{i=1}^n a_i^2 \right)^{1/2}$$

Example of plane task:

Find the length of the vector $a = \{4, 6\}$

$$|a| = \sqrt{4^2 + 6^2} = \sqrt{16 + 36} = \sqrt{52} = 2\sqrt{13}$$

Example of spatial task:

Find the length of the vector $a = \{8, 6, 4\}$

$$|a| = \sqrt{8^2 + 6^2 + 4^2} = \sqrt{64 + 36 + 16} = \sqrt{116} = 2\sqrt{29} = 10.7703$$

Example of n dimensional space task:

Find the length of the vector $a = \{1, 3, 3, 1\}$

$$|a| = \sqrt{1^2 + 3^2 + 3^2 + 1^2} = \sqrt{1 + 9 + 9 + 1} = \sqrt{20} = 2\sqrt{5}$$

Figure 1 – Completed function which calculates the length of a vector.

```
Please enter your choice from the menu above: 1
You have selected option 1.
Enter x for Vector A: 4
Enter y for Vector A: 6
Enter z for Vector A: 8
Vector A entered = (4, 6, 8)
Vector length = 10.7703
```

Figure 2 – Result for the same vector from the online calculator.

Solution:

Calculate [magnitude of a vector](#):

$$|\vec{a}| = \sqrt{a_x^2 + a_y^2 + a_z^2} = \sqrt{4^2 + 6^2 + 8^2} = \sqrt{16 + 36 + 64} = \sqrt{116} = 2\sqrt{29} \approx 10.770329614269007$$

[OnlineMSchool.com](https://www.onlimeschool.com)

The table below displays the test schedule which confirms that the program works correctly.

Actual result shows the values from the completed vector length function, while expected results values are provided by the online calculator from the [onlimeschool.com](https://www.onlimeschool.com).

Test number	Vector values	Expected result	Actual result	Pass / Fail
1	1, 2, 3	3.7416	3.7416	Pass
2	4, 5, 6	8.7749	8.7749	Pass
3	1.5, 2.5, 3.5	4.5552	4.5552	Pass
4	4.5, 5.5, 6.5	9.6306	9.6306	Pass
5	8, 5.6, 9	13.2801	13.2801	Pass
6	12, 7.7, 43	45.3022	45.3022	Pass

Figure 3 – Vector length function.

```
int VectorLength(string message) // Function which calculates the length of the Vector
{
    /* CALCULATE THE LENGTH OF THE VECTOR */

    cout << message << endl; // Cout message

    double x = ReadData("Enter x for Vector A: "); // Function to get input from the user for the x for the Vector A
    double y = ReadData("Enter y for Vector A: "); // Function to get input from the user for the y for the Vector A
    double z = ReadData("Enter z for Vector A: "); // Function to get input from the user for the z for the Vector A

    double length; // Double needed for the length of the Vector A

    length = sqrt(pow(x, 2) + pow(y, 2) + pow(z, 2)); // Calculate the length of the Vector A

    cout << "Vector A entered = (" << x << ", " << y << ", " << z << ")" << endl; // Cout the Vector A entered by the user
    cout << "Vector length = " << length << endl; // Cout the length of the Vector A

    cout << endl; // Go to the next line
    return 0; // Return an integer value
}
```

Several steps were taken to calculate the length of a vector:

1. Get input from the user for the vector (x, y, z).
2. Calculate length of a vector by using correct formula.
3. Output vector entered and its length.

Vector normalization

Vector normalization refers to dividing every entry in a vector by its length to create a vector of length equal to 1. [2]

Formula for the vector normalization:

$$\hat{v} = \frac{\vec{v}}{|\vec{v}|} = \frac{(x,y,z)}{\sqrt{x^2+y^2+z^2}} = \left(\frac{x}{\sqrt{x^2+y^2+z^2}}, \frac{y}{\sqrt{x^2+y^2+z^2}}, \frac{z}{\sqrt{x^2+y^2+z^2}} \right)$$

Example of the vector normalization:

Normalize the vector $r = \{12, -3, -4\}$

$$\begin{aligned} |\vec{r}| &= \sqrt{12^2 + (-3)^2 + (-4)^2} \\ |\vec{r}| &= \sqrt{144 + 9 + 16} \\ |\vec{r}| &= \sqrt{169} \\ |\vec{r}| &= 13 \end{aligned}$$

$$\begin{aligned} \hat{r} &= \frac{(12, -3, -4)}{13} \\ \hat{r} &= \left(\frac{12}{13}, -\frac{3}{13}, -\frac{4}{13} \right) \end{aligned}$$

Figure 4 – Completed function which normalizes the vector.

```
You have selected option 2.
Enter x for Vector A: 6
Enter y for Vector A: 3
Enter z for Vector A: 9
Vector A entered = (6, 3, 9)
Vector length = 11.225
Normalized Vector A = (0.534522, 0.267261, 0.801784)
```

Figure 5 – Result for the same vector from the online calculator.

Original vector		Unit vector	
x	6	x3	0.5345
y	3	y3	0.26726
z	9	z3	0.8018
Magnitude	11.225	Magnitude	1

The table below displays the test schedule which confirms that the program works correctly.

Actual result shows the values from the completed vector normalization function, while expected results values are provided by the online calculator from the [omnicalculator.com](https://www.omnicalculator.com/math/unit-vector).

Test number	Vector values	Expected result	Actual result	Pass / Fail
1	3, 8, 3	0.3313, 0.8835, 0.3313	0.3313, 0.8835, 0.3313	Pass
2	5, 3, 7	0.5488, 0.3293, 0.7683	0.5488, 0.3293, 0.7683	Pass
3	4.5, 3.7, 2.5	0.7098, 0.5836, 0.3943	0.7098, 0.5836, 0.3943	Pass
4	6.5, 5.8, 6.9	0.5849, 0.5219, 0.6209	0.5849, 0.5219, 0.6209	Pass
5	3, 4.9, 1	0.5144, 0.8402, 0.1714	0.5144, 0.8402, 0.1714	Pass
6	11.1, 7, 31	0.3297, 0.2079, 0.9209	0.3297, 0.2079, 0.9209	Pass

Figure 6 – Vector normalization function.

```
int VectorNormalization(string message) // Function which normalizes the Vector
{
    /* NORMALIZE THE VECTOR */

    cout << message << endl; // Cout message

    double x = ReadData("Enter x for Vector A: "); // Function to get input from the user for the x for the Vector A
    double y = ReadData("Enter y for Vector A: "); // Function to get input from the user for the y for the Vector A
    double z = ReadData("Enter z for Vector A: "); // Function to get input from the user for the z for the Vector A

    double length; // Double needed for the length of the Vector A

    length = sqrt(pow(x, 2) + pow(y, 2) + pow(z, 2)); // Calculate the length of the Vector A

    cout << "Vector A entered = (" << x << ", " << y << ", " << z << ")" << endl; // Cout the Vector A entered by the user
    cout << "Vector length = " << length << endl; // Cout the length of the Vector A

    double xnor; // Double needed to normalize the x
    double ynor; // Double needed to normalize the y
    double znor; // Double needed to normalize the z

    xnor = x / length; // Normalize x
    ynor = y / length; // Normalize y
    znor = z / length; // Normalize z

    cout << "Normalized Vector A = (" << xnor << ", " << ynor << ", " << znor << ")" << endl; // Cout normalized Vector A

    cout << endl; // Go to the next line

    return 0; // Return an integer value
}
```

Several steps were taken to normalize the vector:

1. Get input from the user for vector (x, y, z).
2. Calculate length of a vector by using correct formula.
3. Output vector entered and its length.
4. Normalize the vector by dividing each of its values by its length.
5. Output normalized vector.

Distance between the two points

Distance between the two points can be described as the line segment that connects those points. [3]

Formula for the distance between the two points:

$$D = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Example for the distance between the two points:

Find the distance between the point A = (-1, -2) and B = (2, -3)

$$\begin{aligned} d &= \sqrt{(2 - (-1))^2 + (-3 - (-2))^2} \\ &= \sqrt{(2 + 1)^2 + (-3 + 2)^2} \\ &= \sqrt{(3)^2 + (-1)^2} \\ &= \sqrt{9 + 1} = \sqrt{10} \approx 3.16 \end{aligned}$$

Figure 7 – Completed function which calculates the distance between the two points.

```
You have selected option 3.
Enter x for Point A: 2
Enter y for Point A: -4
Enter x for Point B: 5
Enter y for Point B: 3
Point A entered = (2, -4)
Point B entered = (5, 3)
Distance between points A and B = 7.61577
```

Figure 8 – Result for the same points from the online calculator.

GEOMETRY

Distance between 2 Points :

Coordinates (x_A , y_A)

Coordinates (x_B , y_B)

Length = 7.6158

The table below displays the test schedule which confirms that the program works correctly.

Actual result shows the values from the completed function which calculates the distance between the two points, while expected results values are provided by the online calculator from the ncalculators.com.

Test number	Points values	Expected result	Actual result	Pass / Fail
1	4, 3 and 5, 8	5.0990	5.0990	Pass
2	5, 3.3 and 6, 1	2.5079	2.5079	Pass
3	12, 5 and 2, 3	10.198	10.198	Pass
4	4.4, 3.2 and 1, 9	6.7230	6.7230	Pass
5	21, 32 and 51, 8	38.4187	38.4187	Pass
6	9.2, 3.1 and 3, 1	6.5459	6.5459	Pass

Figure 9 – Distance between the two points function.

```
int PointsDistance(string message) // Function which calculates the distance between the two Points
{
    /* CALCULATE THE DISTANCE BETWEEN THE TWO POINTS */

    cout << message << endl; // Cout message

    double xa = ReadData("Enter x for Point A: "); // Function to get input from the user for the x for the Point A
    double ya = ReadData("Enter y for Point A: "); // Function to get input from the user for the y for the Point A
    double xb = ReadData("Enter x for Point B: "); // Function to get input from the user for the x for the Point B
    double yb = ReadData("Enter y for Point B: "); // Function to get input from the user for the y for the Point B

    double distance; // Double needed for the distance between the two points

    distance = sqrt(pow(xa - xb, 2) + pow(ya - yb, 2)); // Calculate the distance between the two points

    cout << "Point A entered = (" << xa << ", " << ya << ")" << endl; // Cout the Point A entered by the user
    cout << "Point B entered = (" << xb << ", " << yb << ")" << endl; // Cout the Point B entered by the user

    cout << "Distance between points A and B = " << distance << endl; // Cout the distance

    cout << endl; // Go to the next line

    return 0; // Return an integer value
}
```

Several steps were taken to calculate the distance between the two points:

1. Get input from the user for the point A (x, y) and the point B (x, y).
2. Calculate the distance by using correct formula.
3. Output points entered and the distance between them.

Dot product of the two vectors

Dot product can be described as scalar product which is a way to combine two vectors into one, where the result is a scalar, rather than a vector. [4]

Formula for the dot product of the two vectors:

$$\vec{a} \cdot \vec{b} = a_1b_1 + a_2b_2 + a_3b_3$$

Example for the dot product:

Find the dot product for the two vectors $a = \{0, 3, -7\}$ and $b = \{2, 3, 1\}$

$$\vec{a} \cdot \vec{b} = 0 + 9 - 7 = 2$$

Figure 10 – Completed function which calculates the dot product of the two vectors.

```
Enter x for Vector A: 2
Enter y for Vector A: 4
Enter z for Vector A: 6
Enter x for Vector B: 4
Enter y for Vector B: 6
Enter z for Vector B: 2
Vector A entered = (2, 4, 6)
Vector B entered = (4, 6, 2)
Vector A length = 7.48331
Vector B length = 7.48331
Angle between Vector A and Vector B = 38.2132 degrees (0.666946 radians)
Dot Product of Vector A and Vector B = 44
```

Figure 11 – Result for the same vectors from the online calculator.

Calculate [dot product](#):

$$\vec{a} \cdot \vec{b} = a_x \cdot b_x + a_y \cdot b_y + a_z \cdot b_z = 2 \cdot 4 + 4 \cdot 6 + 6 \cdot 2 = 8 + 24 + 12 = 44$$

The table below displays the test schedule which confirms that the program works correctly.

Actual result shows the values from the completed function which calculates the dot product of the two vectors, while expected results values are provided by the online calculator from the [onlinemschool.com](#).

Test number	Vectors values	Expected result	Actual result	Pass / Fail
1	2, 5, 6 and 6, 1, 3	35	35	Pass
2	1, 3, 5 and 2, 2, 2	18	18	Pass
3	4.2, 12, 1 and 6, 7, 0	109.2	109.2	Pass
4	23, 1, 3 and 3, 2, 1	74	74	Pass
5	2.2, 3, 9 and 61, 12, 33	467.2	467.2	Pass
6	8, 2.3, 21 and 2, 11, 23	524.3	524.3	Pass

Figure 12 – Dot product function.

```
int DotProduct(string message) // Function which calculates the Dot Product
{
    /* ASK FOR THE VECTORS */

    cout << message << endl; // Cout message

    double xa = ReadData("Enter x for Vector A: "); // Function to get input from the user for the x for the Vector A
    double ya = ReadData("Enter y for Vector A: "); // Function to get input from the user for the y for the Vector A
    double za = ReadData("Enter z for Vector A: "); // Function to get input from the user for the z for the Vector A
    double xb = ReadData("Enter x for Vector B: "); // Function to get input from the user for the x for the Vector B
    double yb = ReadData("Enter y for Vector B: "); // Function to get input from the user for the y for the Vector B
    double zb = ReadData("Enter z for Vector B: "); // Function to get input from the user for the z for the Vector B

    double dotab; // Double needed for the Dot Product

    double lengtha; // Double needed for the length of the Vector A
    double lengthb; // Double needed for the length of the Vector B

    double angled; // Double needed for the Angle in degrees
    double angler; // Double needed for the Angle in radians
    double cosangle; // Double needed for the Cosinus Angle

    /* CALCULATE THE DOT PRODUCT */

    lengtha = sqrt(pow(xa, 2) + pow(ya, 2) + pow(za, 2)); // Add the all values from the Vector A which are raised to the second power and square the result
    lengthb = sqrt(pow(xb, 2) + pow(yb, 2) + pow(zb, 2)); // Add the all values from the Vector B which are raised to the second power and square the result
    cout << "Vector A entered = (" << xa << ", " << ya << ", " << za << ")" << endl; // Cout the Vector A entered by the user
    cout << "Vector B entered = (" << xb << ", " << yb << ", " << zb << ")" << endl; // Cout the Vector B entered by the user
    cout << "Vector A length = " << lengtha << endl; // Cout the lenght of the Vector A
    cout << "Vector B length = " << lengthb << endl; // Cout the lenght of the Vector B

    dotab = xa * xb + ya * yb + za * zb; // Calculate the Dot Product of the Vector A and the Vector B needed for the Cosinus Angle
    cosangle = dotab / (lengtha * lengthb); // Calculate the Cosinus Angle between the Vector A and the Vector B
    angled = acos(cosangle) * 180.0 / M_PI; // Remove the cosinus to calculate the Angle in degrees
    angler = acos(cosangle); // Remove the cosinus to calculate the Angle in radians

    cout << "Angle between Vector A and Vector B = " << angled << " degrees (" << angler << " radians)" << endl; // Cout the Angle
    cout << "Dot Product of Vector A and Vector B = " << dotab << endl; // Cout the Dot Product

    cout << endl; // Go to the next line

    return 0; // Return an integer value
}
```

Several steps were taken to calculate the dot product:

1. Get input from the user for the vector A (x, y, z) and the vector B (x, y, z).
2. Calculate the lengths of those two vectors.
3. Output vectors entered and their lengths.
4. Calculate the dot product using correct formula.
5. Calculate the angle between the two vectors in radians and degrees.
6. Output angle between the two vectors and the dot product.

Cross product of the two vectors

Cross product can be described as a vector perpendicular to the two given vectors which have the length equal to the product of the lengths of the two given vectors that are multiplied by the sine of the angle between the two vectors. [3]

Formula for the cross product of the two vectors:

$$\mathbf{a} \times \mathbf{b} = \|\mathbf{a}\| \|\mathbf{b}\| \sin(\theta) \mathbf{n}$$

$$c_x = a_y b_z - a_z b_y$$

$$c_y = a_z b_x - a_x b_z$$

$$c_z = a_x b_y - a_y b_x$$

Example for the cross product:

Find the cross product for the two vectors $\mathbf{a} = \{2, 3, 4\}$ and $\mathbf{b} = \{5, 6, 7\}$

- $c_x = a_y b_z - a_z b_y = 3 \times 7 - 4 \times 6 = -3$
- $c_y = a_z b_x - a_x b_z = 4 \times 5 - 2 \times 7 = 6$
- $c_z = a_x b_y - a_y b_x = 2 \times 6 - 3 \times 5 = -3$

Answer: $\mathbf{a} \times \mathbf{b} = (-3, 6, -3)$

Figure 13 – Completed function which calculates the cross product of the two vectors.

```
You have selected option 5.
Enter x for Vector A: 1
Enter y for Vector A: 2
Enter z for Vector A: 3
Enter x for Vector B: 4
Enter y for Vector B: 5
Enter z for Vector B: 6
Vector A entered is = (1, 2, 3)
Vector B entered is = (4, 5, 6)
Vector A length = 3.74166
Vector B length = 8.77496
Angle between Vector A and Vector B = 12.9332 degrees (0.225726 radians)
Cross Product of Vector A and Vector B = (-3, 6, -3)
The length of the Vector Cross Product = 7.34847
```

Figure 14 – Result for the same vectors from the online calculator.

Solution:

$$\vec{a} \times \vec{b} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ a_x & a_y & a_z \\ b_x & b_y & b_z \end{vmatrix} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ 1 & 2 & 3 \\ 4 & 5 & 6 \end{vmatrix} = \mathbf{i}(2 \cdot 6 - 3 \cdot 5) - \mathbf{j}(1 \cdot 6 - 3 \cdot 4) + \mathbf{k}(1 \cdot 5 - 2 \cdot 4) =$$

$$= \mathbf{i}(12 - 15) - \mathbf{j}(6 - 12) + \mathbf{k}(5 - 8) = \{-3; 6; -3\}$$

OnlineMSchool.com

The table below displays the test schedule which confirms that the program works correctly.

Actual result shows the values from the completed function which calculates the cross product of the two vectors, while expected results values are provided by the online calculator from the onlinemschool.com.

Test number	Vectors values	Expected result	Actual result	Pass / Fail
1	3, 1, 6 and 4, 1, 3	-3, 15, -1	-3, 15, -1	Pass
2	1, 1, 2 and 2, 1, 2	0, 2, -1	0, 2, -1	Pass
3	1.2, 32, 21 and 1, 3, 2	1, 18.6, -28.4	1, 18.6, -28.4	Pass
4	32, 4, 9 and 1, 3, 2	-19, -55, 92	-19, -55, 92	Pass
5	3.1, 4, 1 and 21, 11, 13	41, -19.3, -49.9	41, -19.3, -49.9	Pass
6	2, 5.3, 2 and 2, 10, 3	-4.1, -2, 9.4	-4.1, -2, 9.4	Pass

Figure 15 – Cross product function.

```
int CrossProduct(string message) // Function which calculates the Cross Product
{
    /* ASK FOR THE VECTORS */

    cout << message << endl; // Cout message

    double xa = ReadData("Enter x for Vector A: "); // Function to get input from the user for the x for the Vector A
    double ya = ReadData("Enter y for Vector A: "); // Function to get input from the user for the y for the Vector A
    double za = ReadData("Enter z for Vector A: "); // Function to get input from the user for the z for the Vector A
    double xb = ReadData("Enter x for Vector B: "); // Function to get input from the user for the x for the Vector B
    double yb = ReadData("Enter y for Vector B: "); // Function to get input from the user for the y for the Vector B
    double zb = ReadData("Enter z for Vector B: "); // Function to get input from the user for the z for the Vector B

    double xc; // Double needed for the x of the Cross Product
    double yc; // Double needed for the y of the Cross Product
    double zc; // Double needed for the z of the Cross Product

    double crossab; // Double needed for the Cross Product
    double dotab; // Double needed for the Dot Product

    double lengtha; // Double needed for the length of the Vector A
    double lengthb; // Double needed for the length of the Vector B

    double angled; // Double needed for the Angle in degrees
    double angler; // Double needed for the Angle in radians
    double cosangle; // Double needed for the Cosinus Angle

    /* CALCULATE THE CROSS PRODUCT */

    lengtha = sqrt(pow(xa, 2) + pow(ya, 2) + pow(za, 2)); // Add the all values from the Vector A which are raised to the second power and square the result
    lengthb = sqrt(pow(xb, 2) + pow(yb, 2) + pow(zb, 2)); // Add the all values from the Vector B which are raised to the second power and square the result

    xc = ya * zb - za * yb; // Calculate x for the Cross Product of the Vector A and the Vector B
    yc = za * xb - xa * zb; // Calculate y for the Cross Product of the Vector A and the Vector B
    zc = xa * yb - ya * xb; // Calculate z for the Cross Product of the Vector A and the Vector B

    dotab = xa * xb + ya * yb + za * zb; // Calculate the Dot Product of the Vector A and the Vector B needed for the Cosinus Angle

    cosangle = dotab / (lengtha * lengthb); // Calculate the Cosinus Angle between the Vector A and the Vector B
    angled = acos(cosangle) * 180.0 / M_PI; // Remove the cosinus to calculate the Angle in degrees
    angler = acos(cosangle); // Remove the cosinus to calculate the Angle in radians

    crossab = lengtha * lengthb * sin(angler); // Calculate the length of the Vector Cross Product

    cout << "Vector A entered is (" << xa << ", " << ya << ", " << za << ")" << endl; // Cout the Vector A entered by the user
    cout << "Vector B entered is (" << xb << ", " << yb << ", " << zb << ")" << endl; // Cout the Vector B entered by the user
    cout << "Vector A length = " << lengtha << endl; // Cout the length of the Vector A
    cout << "Vector B length = " << lengthb << endl; // Cout the length of the Vector B

    cout << "Angle between Vector A and Vector B = " << angled << " degrees (" << angler << " radians)" << endl; // Cout the Angle
    cout << "Cross Product of Vector A and Vector B = (" << xc << ", " << yc << ", " << zc << ")" << endl; // Cout the Cross Product
    cout << "The length of the Vector Cross Product = " << crossab << endl; // Cout the length of the Vector Cross Product

    cout << endl; // Go to the next line

    return 0; // Return an Integer value
}
```

Several steps were taken to calculate the cross product:

1. Get input from the user for the vector A (x, y, z) and the vector B (x, y, z).
2. Calculate the lengths of those two vectors.
3. Calculate the cross product using correct formula.
4. Calculate the dot product needed for the angle between the two vectors.
5. Calculate the angle between the two vectors in radians and degrees.
6. Calculate the length of the vector cross product.
7. Output vectors entered and their lengths.
8. Output angle between the two vectors, cross product and the length of the vector cross product.

Rotation about the y axis transformation

Rotation about the y axis transformation can be described as basic rotation about y axis of a coordinate system. [5]

Formula for the rotation about the y axis:

- Counterclockwise rotation around y-axis

$$R_y(\beta) = \begin{bmatrix} \cos \beta & 0 & \sin \beta \\ 0 & 1 & 0 \\ -\sin \beta & 0 & \cos \beta \end{bmatrix}$$

Example for the rotation about the y axis:

Rotate the point $A = \{3, 4\}$ with an angle of 30 radians, anti-clockwise about the y axis

$$(3, 4) \rightarrow (3 \cos(30) - 4 \sin(30), 4 \cos(30) + 3 \sin(30))$$

$$(3.00, 4.00) \rightarrow (4.41, -2.35)$$

Figure 16 – Completed function which rotates the point.

```
You have selected option 6.
Enter x for point A: 4
Enter y for point A: 3.3
Enter angle for rotation of point A in radians: 34
Point A entered is: A = (4, 3.3)
New point after anti-clockwise rotation about the y axis transformation A' = (-5.14025, -0.683951)
```

Figure 17 – Result for the same point and angle from the online calculator.

Input interpretation:		
rotation	point	(4, 3.3)
	angle	34 radians (counterclockwise)
	center	(0, 0)
Transformed point:		
(4, 3.3)→(-5.14025, -0.683951)		

The table below displays the test schedule which confirms that the program works correctly.

Actual result shows the values from the completed function which rotates the point, while expected results values are provided by the online calculator from the wolframalpha.com.

Test number	Point values and angle	Expected result	Actual result	Pass / Fail
1	5, 6 and 2	-7.536, 2.049	-7.536, 2.049	Pass
2	2, 1 and 3	-2.121, -0.707	-2.121, -0.707	Pass
3	1.1, 7 and 3.3	0.017, -7.085	0.017, -7.085	Pass
4	15, 4 and 6	15.52, -0.350	15.52, -0.350	Pass
5	9, 3 and 4.6	1.971, -9.279	1.971, -9.279	Pass
6	2.8, 3.7 and 5	4.342, -1.635	4.342, -1.635	Pass

Figure 18 – Rotation about the y axis transformation function.

```
int PointRotation(string message) // Function which rotates the Point
{
    /* ROTATE THE POINT */

    cout << message << endl; // Cout message

    double x = ReadData("Enter x for point A: "); // Function to get input from the user for the x
    double y = ReadData("Enter y for point A: "); // Function to get input from the user for the y
    double angle = ReadData("Enter angle for rotation of point A in radians: "); // Function to get input from the user for the angle

    double xnew; // Double needed for the new x
    double ynew; // Double needed for the new y

    xnew = x * cos(angle) - y * sin(angle); // Function which calculates the new x
    ynew = y * cos(angle) + x * sin(angle); // Function which calculates the new y

    cout << "Point A entered is: A = (" << x << ", " << y << ")" << endl; // Cout the Point A
    cout << "New point after anti-clockwise rotation about the y axis transformation A' = (" << xnew << ", " << ynew << ")" << endl; // Cout the new Point A'
    cout << endl; // Go to the next line
    return 0; // Return an integer value
}
```

Several steps were taken to rotate the point about the y axis:

1. Get input from the user for the point A (x, y).
2. Get input from the user for the angle needed for rotation.
3. Calculate new coordinates using correct formula.
4. Output point entered.
5. Output new point after anti-clockwise rotation about the y axis.

Mean, mode and standard deviation

Mean is the sum of all the data entries divided by the number of those entries, can be described as average.

Mode is the data entry that occurs with the greatest frequency.

Standard Deviation measure variability and consistency of the sample population. [6]

Formula for the Mean:

$$\text{Population Mean: } \mu = \frac{\sum x}{N}$$

Formula for the Mode:

$$\text{Mode} = l + h \left(\frac{f_m - f_1}{2f_m - f_1 - f_2} \right)$$

Formula for the Standard Deviation:

$$\text{Population Standard Deviation} = \sigma = \sqrt{\frac{\sum (x - \mu)^2}{N}}$$

Example for the Mean, Mode and Standard Deviation:

Calculate Mean, Mode and Standard Deviation for the numbers 10, 2, 38, 23, 38, 23, 21

Standard Deviation, σ : 12.298996142875

Mean $\frac{10 + 2 + 38 + 23 + 38 + 23 + 21}{7} = \frac{155}{7} = 22.143$

Count, N: 7
Sum, Σx : 155
Mean, μ : 22.142857142857
Variance, σ^2 : 151.26530612245

Mode | 38, 23, each appeared 2 times

Steps

$$\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^N (x_i - \mu)^2}$$

$$\begin{aligned} \sigma^2 &= \frac{\sum (x_i - \mu)^2}{N} \\ &= \frac{(10 - 22.142857142857)^2 + \dots + (21 - 22.142857142857)^2}{7} \\ &= \frac{1058.8571428571}{7} \\ &= 151.26530612245 \\ \sigma &= \sqrt{151.26530612245} \\ &= 12.298996142875 \end{aligned}$$

Figure 19 – Completed function which calculates Mean, Mode and Standard Deviation.

```
You have selected option 7.
Enter array size: 6
Enter number 1: 12
Enter number 2: 13
Enter number 3: 2
Enter number 4: 5
Enter number 5: 7
Enter number 6: 4
Numbers entered for the first array in the ascending order are = { 2 4 5 7 12 13 }
Mean = 7.16667
Mode = { NO MODE }
Mode occurrence = 1 times
Standard Deviation = 4.0586
```

Figure 20 – Results for the same values from the online calculator.

Mean (Average)	7.166666666667	Standard Deviation, σ : 4.058598553962
Median	6	Count, N: 6
Range	11	Sum, Σx : 43
Mode	All values appeared just once.	Mean, μ : 7.166666666667
Geometric Mean	5.93444494969594	Variance, σ^2 : 16.472222222222
Largest	13	Steps
Smallest	2	$\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^N (x_i - \mu)^2}$
Sum	43	$\sigma^2 = \frac{\sum (x_i - \mu)^2}{N}$
Count	6	$= \frac{(2 - 7.166666666667)^2 + \dots + (13 - 7.166666666667)^2}{6}$
Sorted Data Set: 2, 4, 5, 7, 12, 13		$= \frac{98.833333333333}{6}$
		$= 16.472222222222$
		$\sigma = \sqrt{16.472222222222}$
		$= 4.058598553962$

The table below displays the test schedule which confirms that the program works correctly.

Actual result shows the values from the completed functions, while expected results values are provided by the online calculator from the calculator.net.

Test number	Values	Expected result	Actual result	Pass / Fail
1	2, 2, 3, 5	3, 2, 1.224	3, 2, 1.224	Pass
2	3, 7, 8, 1, 1	4, 1, 2.966	4, 1, 2.966	Pass
3	1, 1, 1, 3, 4, 5	2.5, 1, 1.607	2.5, 1, 1.607	Pass
4	12, 3, 12, 4, 5	7.2, 12, 3.969	7.2, 12, 3.969	Pass
5	1.2, 4, 1.6, 1.2	2, 1.2, 1.166	2, 1.2, 1.166	Pass
6	4, 6, 6, 4.4	5.1, 6, 0.911	5.1, 6, 0.911	Pass

Figure 21 – Mean, Mode and Standard Deviation function.

```
int MeanModeDeviation(string message) // Function which calculates Mean, Mode and Standard Deviation
{
    /* ASK FOR THE SIZE AND THE VALUES OF THE ARRAY */

    cout << message << endl; // Cout message

    double number; // Double needed for the array allocation
    double sum = 0; // Initialize sum element to 0
    double mean = 0; // Initialize mean element to 0
    int maxcount = 0; // Initialize maxcount element to 0
    double mode = 0; // Initialize mode element to 0
    double subpow = 0; // Initialize subpow element to 0
    double stddev = 0; // Initialize stddev element to 0

    double* ReadArray = 0; // Pointer to double which initialize to nothing

    int ArraySize = ReadData("Enter array size: "); // Function to get input from the user for the array size
    ReadArray = new double[(int)ArraySize]; // Allocate size of the array inputed by the user and save the pointer in ReadArray
    for (int i = 0; i < ArraySize; i++) // Loop to get input from the user for each array allocation
    {
        cout << "Enter number " << (i + 1); // Cout number
        number = ReadData(""); // Get input from the user for each number
        *(ReadArray + i) = number; // Ask for numbers until the array is full which was defined by the user for the array size
    }

    /* SORT THE ARRAY IN THE ASCENDING ORDER */

    SortArray(ReadArray, ArraySize);

    cout << "Numbers entered for the first array in the ascending order are = { "; // Cout second set
    CoutNumbers(ReadArray, ArraySize);
    cout << "}" << endl; // Cout and go to the next line

    /* CALCULATE THE MEAN */

    for (int i = 0; i < ArraySize; i++) // Loop for the sum of the numbers in the array
    {
        sum = sum + ReadArray[i]; // Sum calculation
    }
    mean = sum / ArraySize; // Mean calculation

    /* CALCULATE THE STANDARD DEVIATION */

    for (int i = 0; i < ArraySize; i++) // Loop which subtracts the mean from each number and square the result
    {
        subpow = subpow + pow(ReadArray[i] - mean, 2); // Calculation which subtracts the mean from each number and square the result
    }
    stddev = sqrt(subpow / ArraySize); // Standard Deviation calculation

    /* CALCULATE THE MODE */

    int display = 1; // Set count to 1 for each number
    cout << "Mean = " << mean << endl; // Cout Mean
    cout << "Mode = { "; // Cout Mode

    for (int i = 0; i < ArraySize; i++) // Loop which checks each number entered by the user
    {
        int count = 1; // Set count to 1 to avoid display numbers where there is no mode
        for (int j = i + 1; j < ArraySize; j++) // Loop which checks each next number entered by the user with the previous number
        {
            if (ReadArray[i] == ReadArray[j]) // Loop which checks if the numbers are equal
            {
                count++; // Increment count by 1 if the numbers are equal
                if (count > maxcount) // Loop which checks if the count is greater than the maxcount
                {
                    maxcount = count; // If yes then maxcount is replaced with count
                }
            }
        }

        for (int i = 0; i < ArraySize; i++) // Loop which checks each number entered by the user
        {
            int count = 1; // Set count to 1 to avoid display numbers where there is no mode
            for (int j = i + 1; j < ArraySize; j++) // Loop which checks each next number entered by the user with the previous number
            {
                if (ReadArray[i] == ReadArray[j]) // Loop which checks if the numbers are equal
                {
                    for (int k = j; k < ArraySize; k++) // Loop which continues to check each next number
                    {
                        ReadArray[k] = ReadArray[k + 1]; // Move pointer forward one position as number was removed
                    }
                    ArraySize--; // Move the pointer back as array can not be checked for the numbers greater than the ThirdArraySize
                    j--; // Move the pointer back to check if equal numbers occur in the same array
                    count++; // Increment count by 1 if the numbers are equal
                    if (count == maxcount) // Loop which checks if the count is equal to the maxcount
                    {
                        cout << ReadArray[i] << " "; // If yes then cout each Mode
                        display = count; // Keeps track of how many times number occurs
                    }
                }
            }
        }

        if (display == 1) // Check if each number occurs only once
        {
            cout << "NO MODE "; // Cout no Mode
        }

        cout << "}" << endl; // Cout and go to the next line

        cout << "Mode occurrence = " << display << " times" << endl; // Cout occurrence and go to the next line
        cout << "Standard Deviation = " << stddev << endl; // Cout Standard Deviation

        cout << endl; // Go to the next line

        return 0; // Return an integer value
    }
}
```

Several steps were taken to calculate Mean, Mode and Standard Deviation:

1. Get input from the user for the array size and its values.
2. Sort the array in the ascending order.
3. Output numbers entered.
4. Calculate the Mean using correct formula.
5. Calculate the Standard Deviation using correct formula.
6. Calculate the Mode using correct formula.
7. Check how many times number occurs.
8. Check if Mode exist and remove duplicates.
9. Output Mean, Mode, Standard Deviation and Mode occurrence.

Union and intersection of the two sets

Union of the two sets is a new set that contains all of the elements of both sets.

Intersection of the two sets is a new set that contains all of the elements from the first set that also belong to the second set. [7]

Formula for the Union:

$$A \cup B = \{x : x \in A \text{ or } x \in B\}.$$

Formula for the Intersection:

$$A \cap B = \{x : x \in A \text{ and } x \in B\}.$$

Example for the Union of the two sets:

Calculate Union and Intersection for the sets $A = \{10, 8, 3, 21\}$ and $B = \{1, 21, 3, 4, 23, 11, 5\}$

Result

You have entered the following sets:

$$A = \{10, 8, 3, 21\}$$

$$B = \{1, 21, 3, 4, 23, 11, 5\}$$

The union of these sets is:

$$A \cup B = \{10, 8, 3, 21, 1, 4, 23, 11, 5\}$$

Result

You have entered the following sets:

$$A = \{10, 8, 3, 21\}$$

$$B = \{1, 21, 3, 4, 23, 11, 5\}$$

The intersection of these sets is:

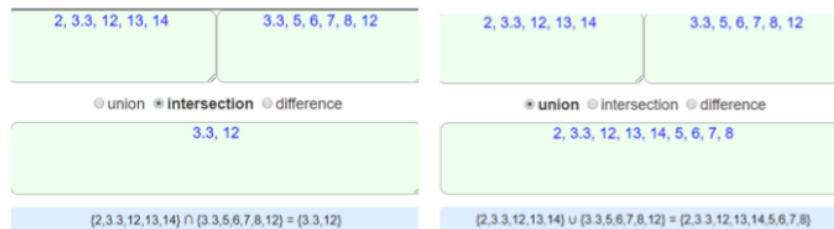
$$A \cap B = \{3, 21\}$$

Figure 22 – Completed function which calculates Union and Intersection.

```

You have selected option 8.
Enter size for the first set: 5
Enter number 1: 12
Enter number 2: 13
Enter number 3: 14
Enter number 4: 3.3
Enter number 5: 2
Enter size for the second set: 6
Enter number 1: 3.3
Enter number 2: 12
Enter number 3: 5
Enter number 4: 6
Enter number 5: 7
Enter number 6: 8
Numbers entered for the first set in the ascending order are = { 2 3.3 12 13 14 }
Numbers entered for the second set in the ascending order are = { 3.3 5 6 7 8 12 }
Union of the first and the second set in the ascending order is = { 2 3.3 5 6 7 8 12 13 14 }
Intersection of the first and the second set in the ascending order is = { 3.3 12 }
    
```

Figure 23 – Results for the same values from the online calculator.



The table below displays the test schedule which confirms that the program works correctly.

Actual result shows the values from the completed functions, while expected results values are provided by the online calculator from the [mathsisfun.com](https://www.mathsisfun.com).

Test number	Values	Expected result	Actual result	Pass / Fail
1	2, 3, 4 and 5, 6, 1, 6, 2	1, 2, 3, 4, 5, 6 and 2	1, 2, 3, 4, 5, 6 and 2	Pass
2	12, 7, 2, 5 and 3, 1, 1, 1, 8	1, 2, 3, 5, 7, 8, 12	1, 2, 3, 5, 7, 8, 12	Pass
3	2.2, 10, 3 and 2.2, 3, 3, 1, 2	1, 2, 2.2, 3, 10 and 2.2, 3	1, 2, 2.2, 3, 10 and 2.2, 3	Pass
4	9, 9, 9 and 3, 3, 3, 4, 1	1, 3, 4, 9	1, 3, 4, 9	Pass
5	3, 3.3, 4 and 4, 4, 12, 1, 4	1, 3, 3.3, 4, 12 and 4	1, 3, 3.3, 4, 12 and 4	Pass
6	1, 2, 3 and 3, 2, 1, 8, 9	1, 2, 3, 8, 9 and 1, 2, 3	1, 2, 3, 8, 9 and 1, 2, 3	Pass

Figure 24 – Union and Intersection of the two sets function.

```

int UnionIntersection(string message) // Function which calculates Union and Intersection
{
    /* ASK FOR THE SIZE AND THE VALUES OF THE FIRST AND THE SECOND ARRAY */

    cout << message << endl; // Cout message

    double ReadThirdArray[1000]; // Array with maxium size of 1000
    int ThirdArraySize; // Size of the array determined by the user
    double numbers1; // Double needed for the array allocation
    double numbers2; // Double needed for the array allocation

    double* ReadFirstArray = 0; // ReadFirstArray pointer to double which initialize to nothing

    int FirstArraySize = ReadData("Enter size for the first set: "); // Function to get input from the user for the array size
    ReadFirstArray = new double[(int)FirstArraySize]; // allocate size of the array inputed by the user and save the pointer in ReadFirstArray
    for (int i = 0; i < FirstArraySize; i++) // Loop to get input from the user for each array allocation
    {
        cout << "Enter number " << i + 1 << ": "; // Cout number
        numbers1 = ReadData(""); // Get input from the user for each number
        *ReadFirstArray + i = numbers1; // Ask for numbers until the array is full which was defined by the user for the array size
    }

    double* ReadSecondArray = 0; // ReadSecondArray pointer to double which initialize to nothing

    int SecondArraySize = ReadData("Enter size for the second set: "); // Function to get input from the user for the array size
    ReadSecondArray = new double[(int)SecondArraySize]; // allocate size of the array inputed by the user and save the pointer in ReadSecondArray
    for (int i = 0; i < SecondArraySize; i++) // Loop to get input from the user for each array allocation
    {
        cout << "Enter number " << i + 1 << ": "; // Cout number
        numbers2 = ReadData(""); // Get input from the user for each number
        *ReadSecondArray + i = numbers2; // Ask for numbers until the array is full which was defined by the user for the array size
    }

    /* SORT THE FIRST AND THE SECOND ARRAY IN THE ASCENDING ORDER */

    SortArray(ReadFirstArray, FirstArraySize);

    cout << "Numbers entered for the first set in the ascending order are = { "; // Cout second set
    CoutNumbers(ReadFirstArray, FirstArraySize);
    cout << "}" << endl; // Cout and go to the next line

    SortArray(ReadSecondArray, SecondArraySize);

    cout << "Numbers entered for the second set in the ascending order are = { "; // Cout second set
    CoutNumbers(ReadSecondArray, SecondArraySize);
    cout << "}" << endl; // Cout and go to the next line

    /* MERGE FIRST AND SECOND ARRAY INTO THIRD ARRAY */

    for (int i = 0; i < FirstArraySize; i++) // Loop which checks each number entered by the user
    {
        ReadThirdArray[i] = ReadFirstArray[i]; // Move all the numbers from ReadFirstArray[i] to ReadThirdArray[i]
    }

    ThirdArraySize = FirstArraySize + SecondArraySize; // Update the size of the ThirdArraySize to match FirstArraySize + SecondArraySize
    for (int i = 0, k = FirstArraySize; k < ThirdArraySize && i < SecondArraySize; i++, k++) // Loop which checks each number in the ReadThirdArray after all the numbers already added from ReadFirstArray
    {
        ReadThirdArray[k] = ReadSecondArray[i]; // Move all the numbers from ReadSecondArray[i] to ReadThirdArray[k] after all the numbers already added from ReadFirstArray
    }

    /* SORT THE THIRD ARRAY IN THE ASCENDING ORDER */

    SortArray(ReadThirdArray, ThirdArraySize);

    /* REMOVE DUPLICATES FROM THE THIRD ARRAY */

    for (int i = 0; i < ThirdArraySize; i++) // Loop which checks each number entered by the user
    {
        for (int j = i + 1; j < ThirdArraySize; j++) // Loop which checks each next number entered by the user with previous number
        {
            if (ReadThirdArray[i] == ReadThirdArray[j]) // Loop which checks if the numbers are equal and removes second one
            {
                for (int k = j; k < ThirdArraySize; k++) // Loop which continues to check each next number
                {
                    ReadThirdArray[k] = ReadThirdArray[k + 1]; // Move pointer forward one position as number was removed
                }
                ThirdArraySize--; // Move the pointer back as array can not be checked for the numbers greater than the ThirdArraySize
                j--; // Move the pointer back to check if equal numbers occur in the same array
            }
        }
    }

    cout << "Union of the first and the second set in the ascending order is = { "; // Cout Union
    CoutNumbers(ReadThirdArray, ThirdArraySize);
    cout << "}" << endl; // Cout and go to the next line

    /* REMOVE DUPLICATES FROM THE FIRST ARRAY */

    RemoveDuplicates(ReadFirstArray, FirstArraySize);

    /* REMOVE DUPLICATES FROM THE SECOND ARRAY */

    RemoveDuplicates(ReadSecondArray, SecondArraySize);

    /* CHECK IF THE NUMBERS FROM THE FIRST AND THE SECOND ARRAY ARE EQUAL */

    cout << "Intersection of the first and the second set in the ascending order is = { "; // Cout Intersection
    for (int i = 0; i < FirstArraySize; i++) // Loop which checks each number entered by the user
    {
        for (int j = 0; j < SecondArraySize; j++) // Loop which checks each number entered by the user
        {
            if (ReadFirstArray[i] == ReadSecondArray[j]) // Check if ReadFirstArray[i] and ReadSecondArray[j] numbers are equal
            {
                cout << ReadFirstArray[i] << " "; // Cout equal numbers of the Intersection in the ascending order
            }
        }
    }

    cout << "}" << endl; // Cout and go to the next line

    cout << endl; // Go to the next line

    return 0; // Return an integer value
}

```

Several steps were taken to calculate Union and Intersection of the two sets:

1. Get input from the user for the arrays sizes and their values.
2. Sort the arrays in the ascending order.
3. Output numbers entered.
4. Merge two arrays into one.
5. Sort the merged array in the ascending order.
6. Remove duplicates from the all arrays.
7. Check if numbers from the first and the second array are equal.
8. Output Union and Intersection of the two sets.

References

- [1 D. A. S. Will H Moore, in *A Mathematics Course for Political and Social Research*, New Jersey, Princeton University Press, 2013, pp. 278-280.
- [2 L. L. Pierre Legendre, in *Numerical Ecology Third Edition*, Amsterdam, Elsevier B.V., 2012, pp. 90-100.
- [3 M. R. C. Dennis G Zill, in *Advanced Engineering Mathematics Third Edition*, Canada, Jones and Bartlett Publishers, 2006, pp. 300-320.
- [4 B. G. R. I C Jong, in *Engineering Mechanics: Statics*, Orlando, Saunders College, 1991, pp. 113-114.
- [5 H. Jones, in *Computer Graphics through Key Mathematics*, London, Springer Science & Business Media, 2001, pp. 167-172.
- [6 college.cengage, [Online]. Available:
] https://college.cengage.com/mathematics/brase/understandable_statistics/9780618949922_ch03.pdf.
- [7 R. R. Dhruv Mubayi, "Set systems with union and intersection constraints," *Journal of Combinatorial Theory*, vol. 99, no. 3, pp. 639-642, 2009.

Application of Maths for Software Development

GRADEMARK REPORT

FINAL GRADE

GENERAL COMMENTS

Instructor

92/100

PAGE 1

PAGE 2

PAGE 3



Comment 1

we use (..) brackets typically. avoid braces here.

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16

PAGE 17

PAGE 18

PAGE 19

PAGE 20

PAGE 21

PAGE 22

PAGE 23

PAGE 24

PAGE 25

PAGE 26



Comment 2

only one instance of an element - no repetitions.

PAGE 27

PAGE 28

PAGE 29

GRADING FORM: SMU GENERIC_GJ

DOMINIK KANIA

91.50

RESEARCH-INFORMED LITERATURE



Excellent refs.

4.5

KNOWLEDGE AND UNDERSTANDING



Excellent understanding demonstrated here. See comments below re set intersections.

18.5

ANALYSIS



Code and test design excellent. Calculations and methods demonstrated very well.

18.5

PRACTICAL APPLICATION AND DEP.



Excellent application - a good set of results and test schedule complete. Comments appropriate. Try to provide a description of what a function (or method, should you use OOP) does at the beginning of the function definition. Intersections of sets should have no repetition of elements common to both - code appears to work here but fig 24 appears to contradict. Unclear?.

46

SKILLS FOR PROFESSIONAL PRAC.



Excellent. good structure - try to be consistent in the number of d.p. s used and use (..) brackets for vectors (in line with the text books and conventions adopted here).

4