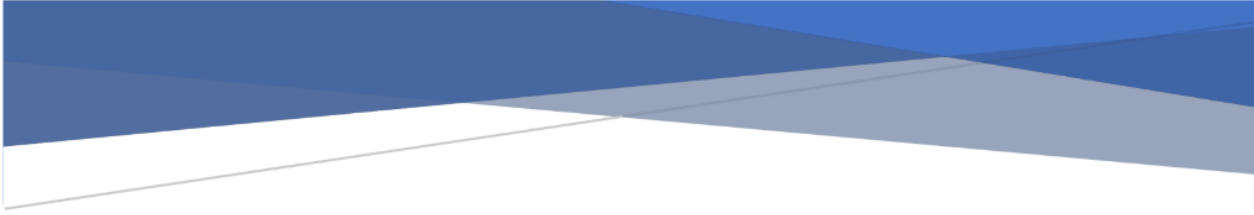


Math SwDv Coding Assignment

by Rhys Whatty

FILE	17862_RHYS_WHATTY_MATH_SWDV_CODING_ASSIGNMENT_250776_673080113.DOCX (802.02K)		
TIME SUBMITTED	22-APR-2020 12:23PM (UTC+0100)	WORD COUNT	2030
SUBMISSION ID	124741039	CHARACTER COUNT	9677



APPLICATION OF MATH FOR SOFTWARE DEVELOPMENT

ACDA4003 - Mathematics

Rhys Whatty (1900045)

Contents

Operation 1 - Vector Length	2
Theory of Operation.....	2
Implementation of Code	2
Testing.....	3
Operation 2 – Normalise a Vector	5
Theory of Operation.....	5
Implementation of Code	5
Testing.....	5
Operation 3 – Distance between 2 points	8
Theory of Operation.....	8
Implementation of Code	8
Testing.....	9
Operation 4 – Dot Product of 2 vectors	11
Theory of Operation.....	11
Implementation of Code	11
Testing.....	12
Operation 5 – Cross Product of 2 vectors	15
Theory of Operation.....	15
Implementation of Code	15
Testing.....	16
Operation 6 – Rotation around a Y-Axis	18
Theory of Operation.....	18
Implementation of Code	18
Testing.....	19
Operation 7 – Mean and Standard Deviation.....	21
Theory of Operation.....	21
Implementation of Code	21
Testing.....	22
Operation 8 – Union and Intersection of 2 sets.....	24
Theory of Operation.....	24
Implementation of Code	24
Testing.....	26
References	28

Operation 1 - Vector Length

Theory of Operation

Vector Length can be calculated as the hypotenuse of a right-angled triangle, and so can be calculated using Pythagoras' theorem. The theorem is $c^2 = a^2 + b^2$, with c being the hypotenuse, and a and b being the 2 lines connect by a right angle. [1]

Implementation of Code

The code has the user enter an X and Y value and then offers the option of a Z value to be entered.

```
//User enters X Value of the vector
cout << "Enter the X value of the vector: ";
getline(cin, input);
stream.str(input);
stream >> vectorAX;
stream.clear();

//User enters Y Value of the vector
cout << "Enter the Y value of the vector: ";
getline(cin, input);
stream.str(input);
stream >> vectorAY;
stream.clear();

cout << endl << endl << "Enter a Z value? (Y/N)" << endl;
getline(cin, strOption);
if (strOption == "Y")
{
    //User enters Z Value of the vector
    cout << "Enter the Z value of the vector: ";
    getline(cin, input);
    stream.str(input);
    stream >> vectorAZ;
    stream.clear();
}
```

The code will then use the specified formula to calculate the vector length with the entered values, and outputs the answer to the user.

```

//User enters Z Value of the vector
cout << "Enter the Z value of the vector: ";
getline(cin, input);
stream.str(input);
stream >> vectorAZ;
stream.clear();

//Code calculates and outputs resulting vector length
vectorLength = sqrt((vectorAX * vectorAX) + (vectorAY * vectorAY) + (vectorAZ * vectorAZ));
cout << "The Length of the vector (" << vectorAX << ", " << vectorAY << ", " << vectorAZ << ") is " << vectorLength << endl << endl;
cout << "Press Enter to return to the main menu" << endl << endl << endl;
cin.get();
}
else if (strOption == "N")
{
    //Code calculates and outputs resulting vector length
    vectorLength = sqrt((vectorAX * vectorAX) + (vectorAY * vectorAY));
    cout << "The Length of the vector (" << vectorAX << ", " << vectorAY << ") is " << vectorLength << endl << endl;
    cout << "Press Enter to return to the main menu" << endl << endl << endl;
    cin.get();
}

```

Testing

Test Number	Description	Expected Result	Actual Result	Pass/Fail
1	Begin the process of solving Length	When user enters 1 into the option, the code for solving length triggers	Code for solving length triggers when 1 is entered	Pass
2	Test code with user-entered values	Code can calculate length with values entered	Code successfully calculates length	Pass
3	Optional third value	Code should solve the length while offering the user the option of a Z value	Code still works with and without a third Z value that is optional to the user	Pass

Here is a screengrab of the finished code and of it working in a console.

```

case 1: //1. Vector Length

    cout << "Calculate Length of a Vector" << endl << endl;

    //User enters X Value of the vector
    cout << "Enter the X value of the vector: ";
    getline(cin, input);
    stream.str(input);
    stream >> vectorAX;
    stream.clear();

    //User enters Y Value of the vector
    cout << "Enter the Y value of the vector: ";
    getline(cin, input);
    stream.str(input);
    stream >> vectorAY;
    stream.clear();

    cout << endl << endl << "Enter a Z value? (Y/N)" << endl;
    getline(cin, strOption);
    if (strOption == "Y")
    {
        //User enters Z Value of the vector
        cout << "Enter the Z value of the vector: ";
        getline(cin, input);
        stream.str(input);
        stream >> vectorAZ;
        stream.clear();

        //Code calculates and outputs resulting vector length
        vectorLength = sqrt((vectorAX * vectorAX) + (vectorAY * vectorAY) + (vectorAZ * vectorAZ));
        cout << "The Length of the vector (" << vectorAX << ", " << vectorAY << ", " << vectorAZ << ") is " << vectorLength << endl << endl;
        cout << "Press Enter to return to the main menu" << endl << endl << endl;
        cin.get();
    }
    else if (strOption == "N")
    {
        //Code calculates and outputs resulting vector length
        vectorLength = sqrt((vectorAX * vectorAX) + (vectorAY * vectorAY));
        cout << "The Length of the vector (" << vectorAX << ", " << vectorAY << ") is " << vectorLength << endl << endl;
        cout << "Press Enter to return to the main menu" << endl << endl << endl;
        cin.get();
    }
    else
    {
        cout << "Error, returning to menu" << endl << endl << endl;
        cin.get();
    }
    break;

```

Calculate Length of a Vector

Enter the X value of the vector: 3
Enter the Y value of the vector: 4

Enter a Z value? (Y/N)

Y

Enter the Z value of the vector: 5
The Length of the vector (3, 4, 5) is 7.07107
Press Enter to return to the main menu

Operation 2 – Normalise a Vector

Theory of Operation

A vector can be normalised by dividing the vector by its length. This formula is often expressed as $\hat{a} = a / ||a||$, with $||a||$ being the vector length. The normalized vector will be displayed as a vector with an equal amount of points to the original vector. [2]

Implementation of Code

The code first makes the user enter an X, Y and optional Z value, like on Operation 1. The code then solves each normal value separately and outputs them as a vector.

```
cout << endl << endl << "Enter a Z value? (Y/N)" << endl;
getline(cin, strOption);
if (strOption == "Y")
{
    //User enters Z Value of the vector
    cout << "Enter the Y value of the vector: ";
    getline(cin, input);
    stream.str(input);
    stream >> vectorAZ;
    stream.clear();

    //Code calculates and outputs resulting normalized vector
    vectorLength = sqrt((vectorAX * vectorAX) + (vectorAY * vectorAY) + (vectorAZ * vectorAZ));
    normalVectorX = vectorAX / vectorLength;
    normalVectorY = vectorAY / vectorLength;
    normalVectorZ = vectorAZ / vectorLength;

    cout << "The Normalized Vector is (" << normalVectorX << ", " << normalVectorY << ", " << normalVectorZ << ")" << endl << endl;
    cout << "Press Enter to return to the main menu" << endl << endl << endl;
    cin.get();
}
else if (strOption == "N")
{
    //Code calculates and outputs resulting normalized vector
    vectorLength = sqrt((vectorAX * vectorAX) + (vectorAY * vectorAY));
    normalVectorX = vectorAX / vectorLength;
    normalVectorY = vectorAY / vectorLength;

    cout << "The Normalized Vector is (" << normalVectorX << ", " << normalVectorY << ")" << endl << endl;
    cout << "Press Enter to return to the main menu" << endl << endl << endl;
    cin.get();
}
```

Testing

Test Number	Description	Expected Result	Actual Result	Pass/Fail
1	Begin the process of normalizing a vector	When user enters 2 into the option, the code for solving normalized vectors triggers	Code for solving normalized vectors triggers when 2 is entered	Pass

2	Code solves Vector Length	When user enters data, the code should be able to solve the vector length	Vector Length is successfully solved	Pass
3	Code solves normalized values	After length is solved, the code should then solve the values for the normalized vector	Normalized vector is solved successfully	Pass

Here is a screengrab of the finished code and of it working in a console.

```

case 2: //2. Normalize a vector
{
    cout << "Normalize a vector" << endl << endl;

    //User enters X Value of the vector
    cout << "Enter the X value of the vector: ";
    getline(cin, input);
    stream.str(input);
    stream >> vectorAX;
    stream.clear();

    //User enters Y Value of the vector
    cout << "Enter the Y value of the vector: ";
    getline(cin, input);
    stream.str(input);
    stream >> vectorAY;
    stream.clear();

    cout << endl << endl << "Enter a Z value? (Y/N)" << endl;
    getline(cin, strOption);
    if (strOption == "Y")
    {
        //User enters Z Value of the vector
        cout << "Enter the Y value of the vector: ";
        getline(cin, input);
        stream.str(input);
        stream >> vectorAZ;
        stream.clear();

        //Code calculates and outputs resulting normalized vector
        vectorLength = sqrt((vectorAX * vectorAX) + (vectorAY * vectorAY) + (vectorAZ * vectorAZ));
        normalVectorX = vectorAX / vectorLength;
        normalVectorY = vectorAY / vectorLength;
        normalVectorZ = vectorAZ / vectorLength;

        cout << "The Normalized Vector is (" << normalVectorX << ", " << normalVectorY << ", " << normalVectorZ << ")" << endl << endl;
        cout << "Press Enter to return to the main menu" << endl << endl << endl;
        cin.get();
    }
}

else if (strOption == "N")
{
    //Code calculates and outputs resulting normalized vector
    vectorLength = sqrt((vectorAX * vectorAX) + (vectorAY * vectorAY));
    normalVectorX = vectorAX / vectorLength;
    normalVectorY = vectorAY / vectorLength;

    cout << "The Normalized Vector is (" << normalVectorX << ", " << normalVectorY << ")" << endl << endl;
    cout << "Press Enter to return to the main menu" << endl << endl << endl;
    cin.get();
}
else
{
    cout << "Error, returning to menu" << endl << endl << endl;
    cin.get();
}
break;

```

Normalize a vector

Enter the X value of the vector: 3

Enter the Y value of the vector: 4

Enter a Z value? (Y/N)

Y

Enter the Y value of the vector: 5

The Normalized Vector is (0.424264, 0.565685, 0.707107)

Press Enter to return to the main menu

Operation 3 – Distance between 2 points

Theory of Operation

Using a similar method to measuring Vector Length, a distance between 2 points can be calculated as the hypotenuse of a right-angled triangle, and so can be calculated using Pythagoras' theorem. The theorem is $c^2 = a^2 + b^2$, with c being the hypotenuse, a being the difference in x values and b being the difference in y values. [1]

Implementation of Code

First, the user is asked to enter an X and Y co-ordinate for both points.

```
//User enters co-ordinates
cout << "Enter the X value of the first set of co-ordinates: ";
getline(cin, input);
stream.str(input);
stream >> vectorAX;
stream.clear();
cout << "Enter the Y value of the first set of co-ordinates: ";
getline(cin, input);
stream.str(input);
stream >> vectorAY;
stream.clear();

cout << "Enter the X value of the second set of co-ordinates: ";
getline(cin, input);
stream.str(input);
stream >> vectorBX;
stream.clear();
cout << "Enter the Y value of the second set of co-ordinates: ";
getline(cin, input);
stream.str(input);
stream >> vectorBY;
stream.clear();
```

Then the code calculates the differences between the X and Y values, then squares the results.

Testing

Test Number	Description	Expected Result	Actual Result	Pass/Fail
1	Begin the process of solving the distance between 2 points	When user enters 3 into the option, the code for solving distance triggers	Code for solving distance triggers when 3 is entered	Pass
2	User enters 2 X and Y co-ordinates	User can enter the co-ordinates for 2 points to be solved	User can successfully enter 2 points	Pass
3	Code solves distance between points	The code should determine the distance between the points entered	Distance is solved successfully	Pass

Here is a screengrab of the finished code and of it working in a console.

```
case 3: //3. Distance between 2 points

    cout << "Calculate the distance between 2 points" << endl << endl;

    //User enters co-ordinates
    cout << "Enter the X value of the first set of co-ordinates: ";
    getline(cin, input);
    stream.str(input);
    stream >> vectorAX;
    stream.clear();
    cout << "Enter the Y value of the first set of co-ordinates: ";
    getline(cin, input);
    stream.str(input);
    stream >> vectorAY;
    stream.clear();

    cout << "Enter the X value of the second set of co-ordinates: ";
    getline(cin, input);
    stream.str(input);
    stream >> vectorBX;
    stream.clear();
    cout << "Enter the Y value of the second set of co-ordinates: ";
    getline(cin, input);
    stream.str(input);
    stream >> vectorBY;
    stream.clear();

    //Code calculates and outputs resulting distance
    xSq = (vectorBX - vectorAX) * (vectorBX - vectorAX);
    ySq = (vectorBY - vectorAY) * (vectorBY - vectorAY);
    distance = sqrt((xSq * xSq) + (ySq * ySq));
    cout << "The distance between the 2 points is " << distance << endl << endl;
    cout << "Press Enter to return to the main menu" << endl << endl << endl;
    cin.get();

    break;
```

Calculate the distance between 2 points

Enter the X value of the first set of co-ordinates: 1

Enter the Y value of the first set of co-ordinates: 2

Enter the X value of the second set of co-ordinates: 3

Enter the Y value of the second set of co-ordinates: 4

The distance between the 2 points is 5.65685

Press Enter to return to the main menu

Operation 4 – Dot Product of 2 vectors

Theory of Operation

Dot Product is used to help solve vector length and direction. Dot product can be displayed as a formula as either $a \cdot b = |a| \cdot |b| \cdot \cos \theta$ or, and as the method that will be used in the code, $a \cdot b = a_x b_x + a_y b_y + a_z b_z$. [2]

Implementation of Code

First, the user enters the X and Y value of the first vector, then the second vector and then if offered the option of adding Z values.

```
cout << "Enter the X value of the first vector: ";
getline(cin, input);
stream.str(input);
stream >> vectorAX;
stream.clear();
cout << "Enter the Y value of the first vector: ";
getline(cin, input);
stream.str(input);
stream >> vectorAX;
stream.clear();

cout << "Enter the X value of the second vector: ";
getline(cin, input);
stream.str(input);
stream >> vectorBX;
stream.clear();
cout << "Enter the Y value of the second vector: ";
getline(cin, input);
stream.str(input);
stream >> vectorBY;
stream.clear();

cout << endl << endl << "Enter a Z value? (Y/N)" << endl;
getline(cin, strOption);
if (strOption == "Y")
{
    cout << "Enter the Z value of the first vector: ";
    getline(cin, input);
    stream.str(input);
    stream >> vectorAZ;
    stream.clear();
    cout << "Enter the Z value of the second vector: ";
    getline(cin, input);
    stream.str(input);
    stream >> vectorBZ;
    stream.clear();
}
```

Next the code calculates and outputs the Dot Product using the entered values.

```
//Code calculates and outputs resulting distance
dotProduct = (vectorAX * vectorBX) + (vectorAY * vectorBY) + (vectorAZ * vectorBZ);
cout << "The dot product of the 2 vectors is " << dotProduct << endl << endl;
cout << "Press Enter to return to the main menu" << endl << endl << endl;
cin.get();
}
else if (strOption == "N")
{
    //Code calculates and outputs resulting distance
    dotProduct = (vectorAX * vectorBX) + (vectorAY * vectorBY);
    cout << "The dot product of the 2 vectors is " << dotProduct << endl << endl;
    cout << "Press Enter to return to the main menu" << endl << endl << endl;
    cin.get();
}
```

Testing

Test Number	Description	Expected Result	Actual Result	Pass/Fail
1	Begin the process of solving Dot Product	When user enters 4 into the option, the code for solving dot product triggers	Code for solving dot product triggers when 4 is entered	Pass
2	User enters values for vectors	User should be able to enter X, Y and optional Z values for 2 vectors	Values can be successfully entered	Pass
3	Code calculates Dot Product	The code will take the values entered and solve the Dot Product of the 2 vectors	Dot Product is solved successfully	Pass

Here is a screengrab of the finished code and of it working in a console.

```

case 4: //4. Dot Product of 2 vectors

    cout << "Calculate the Dot Product of 2 vectors" << endl << endl;

    //User enters co-ordinates
    cout << "Enter the X value of the first vector: ";
    getline(cin, input);
    stream.str(input);
    stream >> vectorAX;
    stream.clear();
    cout << "Enter the Y value of the first vector: ";
    getline(cin, input);
    stream.str(input);
    stream >> vectorAY;
    stream.clear();

    cout << "Enter the X value of the second vector: ";
    getline(cin, input);
    stream.str(input);
    stream >> vectorBX;
    stream.clear();
    cout << "Enter the Y value of the second vector: ";
    getline(cin, input);
    stream.str(input);
    stream >> vectorBY;
    stream.clear();

    cout << endl << endl << "Enter a Z value? (Y/N)" << endl;
    getline(cin, strOption);
    if (strOption == "Y")
    {
        cout << "Enter the Z value of the first vector: ";
        getline(cin, input);
        stream.str(input);
        stream >> vectorAZ;
        stream.clear();
        cout << "Enter the Z value of the second vector: ";
        getline(cin, input);
        stream.str(input);
        stream >> vectorBZ;
        stream.clear();

        //Code calculates and outputs resulting distance
        dotProduct = (vectorAX * vectorBX) + (vectorAY * vectorBY) + (vectorAZ * vectorBZ);
        cout << "The dot product of the 2 vectors is " << dotProduct << endl << endl;
        cout << "Press Enter to return to the main menu" << endl << endl << endl;
        cin.get();
    }
}

```

```

else if (strOption == "N")
{
    //Code calculates and outputs resulting distance
    dotProduct = (vectorAX * vectorBX) + (vectorAY * vectorBY);
    cout << "The dot product of the 2 vectors is " << dotProduct << endl << endl;
    cout << "Press Enter to return to the main menu" << endl << endl << endl;
    cin.get();
}
else
{
    cout << "Error, returning to menu" << endl << endl << endl;
    cin.get();
}
break;

```

Calculate the Dot Product of 2 vectors

Enter the X value of the first vector: 1

Enter the Y value of the first vector: 2

Enter the X value of the second vector: 3

Enter the Y value of the second vector: 4

Enter a Z value? (Y/N)

N

The dot product of the 2 vectors is 14

Press Enter to return to the main menu

Operation 5 – Cross Product of 2 vectors

Theory of Operation

The cross product is used to generate a new vector that runs orthogonally to 2 vectors at the point where they meet. It can be calculated with the formula $\mathbf{a} \times \mathbf{b} = ((a_y b_z - a_z b_y), (a_x b_z - a_z b_x), (a_x b_y - a_y b_x))$. [2]

Implementation of Code

First, the user enters the X, Y and Z values for the 2 vectors.

```
//User enters co-ordinates
cout << "Enter the X value of the first vector: ";
getline(cin, input);
stream.str(input);
stream >> vectorAX;
stream.clear();
cout << "Enter the Y value of the first vector: ";
getline(cin, input);
stream.str(input);
stream >> vectorAX;
stream.clear();
cout << "Enter the Z value of the first vector: ";
getline(cin, input);
stream.str(input);
stream >> vectorAZ;
stream.clear();

cout << "Enter the X value of the second vector: ";
getline(cin, input);
stream.str(input);
stream >> vectorBX;
stream.clear();
cout << "Enter the Y value of the second vector: ";
getline(cin, input);
stream.str(input);
stream >> vectorBY;
stream.clear();
cout << "Enter the Z value of the second vector: ";
getline(cin, input);
stream.str(input);
stream >> vectorBZ;
stream.clear();
```

Then, the code calculates and outputs the Cross Product in the form of a vector.

```
//Code calculates and outputs resulting distance
cout << "The Cross Product of the 2 vectors is (" << (vectorAY * vectorBZ) - (vectorAZ * vectorBY) << ", "
<< (vectorAZ * vectorBX) - (vectorAX * vectorBZ) << ", " << (vectorAX * vectorBY) - (vectorAY * vectorBX) << ")" << endl << endl;
cout << "Press Enter to return to the main menu" << endl << endl << endl;
cin.get();
break;
```

Testing

Test Number	Description	Expected Result	Actual Result	Pass/Fail
1	Begin the process of solving Cross Product	When user enters 5 into the option, the code for solving Cross Product triggers	Code for solving Cross Product triggers when 5 is entered	Pass
2	User enters 3D vectors	User should be able to enter and X, Y and Z co-ordinates for 2 vectors	All co-ordinates are successfully entered.	Pass
3	Code solves Cross product	Code should solve the cross product of the 2 entered vectors.	Cross product is solved successfully and outputted as a vector.	Pass  2

Here is a screengrab of the finished code and of it working in a console.

```

case 5: //5. Cross Product of 2 vectors

    cout << "Calculate the Cross Product of 2 vectors" << endl << endl;

    //User enters co-ordinates
    cout << "Enter the X value of the first vector: ";
    getline(cin, input);
    stream.str(input);
    stream >> vectorAX;
    stream.clear();
    cout << "Enter the Y value of the first vector: ";
    getline(cin, input);
    stream.str(input);
    stream >> vectorAY;
    stream.clear();
    cout << "Enter the Z value of the first vector: ";
    getline(cin, input);
    stream.str(input);
    stream >> vectorAZ;
    stream.clear();

    cout << "Enter the X value of the second vector: ";
    getline(cin, input);
    stream.str(input);
    stream >> vectorBX;
    stream.clear();
    cout << "Enter the Y value of the second vector: ";
    getline(cin, input);
    stream.str(input);
    stream >> vectorBY;
    stream.clear();
    cout << "Enter the Z value of the second vector: ";
    getline(cin, input);
    stream.str(input);
    stream >> vectorBZ;
    stream.clear();

    //Code calculates and outputs resulting distance
    cout << "The Cross Product of the 2 vectors is (" << (vectorAY * vectorBZ) - (vectorAZ * vectorBY) << ", "
        << (vectorAZ * vectorBX) - (vectorAX * vectorBZ) << ", " << (vectorAX * vectorBY) - (vectorAY * vectorBX) << ")" << endl << endl;
    cout << "Press Enter to return to the main menu" << endl << endl << endl;
    cin.get();
    break;

```

Calculate the Cross Product of 2 vectors

```

Enter the X value of the first vector: 3
Enter the Y value of the first vector: 2
Enter the Z value of the first vector: 5
Enter the X value of the second vector: 1 3
Enter the Y value of the second vector: 2
Enter the Z value of the second vector: 3
The Cross Product of the 2 vectors is (-4, -1, 2)

Press Enter to return to the main menu

```

Operation 6 – Rotation around a Y-Axis

Theory of Operation

Rotating a point around a Y-Axis takes an x and y co-ordinate and rotates each co-ordinate a certain number of degrees around an origin. It can be calculated with the formula $X_{new} = X_{old} \cos \theta - Y_{old} \sin \theta$ and $Y_{new} = X_{old} \sin \theta + Y_{old} \cos \theta$. [2]

Implementation of Code

First, the user enters the X and Y values of the co-ordinate they wish to rotate and the angle that they wish to rotate the co-ordinates.

```
//User enters co-ordinates
cout << "Enter the X value of the first set of co-ordinates: ";
getline(cin, input);
stream.str(input);
stream >> vectorAX;
stream.clear();
cout << "Enter the Y value of the first set of co-ordinates: ";
getline(cin, input);
stream.str(input);
stream >> vectorAY;
stream.clear();

//User enters rotation
cout << "Enter the Angle you wish to rotate the shape: ";
getline(cin, input);
stream.str(input);
stream >> rotationAngle;
stream.clear();
```

Then the code transforms the 2 points to solve the location of the rotated co-ordinate.

```
//Code calculates and outputs the new co-ordinates
vectorBX = vectorAX * cos(rotationAngle) - vectorAY * sin(rotationAngle);
vectorBY = vectorAX * sin(rotationAngle) + vectorAY * cos(rotationAngle);
cout << "The new co-ordinates after being rotated are (" << vectorBX << ", " << vectorBY << ")" << endl << endl;
cout << "Press Enter to return to the main menu" << endl << endl << endl;
cin.get();
break;
```

Testing

Test Number	Description	Expected Result	Actual Result	Pass/Fail
1	Begin the process of solving rotation of a point	When user enters 6 into the option, the code for solving Rotation triggers	Code for solving Rotation triggers when 6 is entered	Pass
2	User enters values	User should be able to enter X and Y co-ordinates of the point to rotate, and the angle they wish to rotate	All co-ordinates and rotation are entered successfully	Pass
3	Code produced new co-ordinates	Code should calculate the new co-ordinates after the old ones have been rotated around the Y-Axis by the amount entered.	Points are correctly and successfully calculated and outputted.	Pass

Here is a screengrab of the finished code and of it working in a console.

```
case 6: //6. Rotation around the Y-Axis

    cout << "Rotating a point around the Y-Axis" << endl << endl;

    //User enters co-ordinates
    cout << "Enter the X value of the first set of co-ordinates: ";
    getline(cin, input);
    stream.str(input);
    stream >> vectorAX;
    stream.clear();
    cout << "Enter the Y value of the first set of co-ordinates: ";
    getline(cin, input);
    stream.str(input);
    stream >> vectorAY;
    stream.clear();

    //User enters rotation
    cout << "Enter the Angle you wish to rotate the shape: ";
    getline(cin, input);
    stream.str(input);
    stream >> rotationAngle;
    stream.clear();

    //Code calculates and outputs the new co-ordinates
    vectorBX = vectorAX * cos(rotationAngle) - vectorAY * sin(rotationAngle);
    vectorBY = vectorAX * sin(rotationAngle) - vectorAY * cos(rotationAngle);
    cout << "The new co-ordinates after being rotated are (" << vectorBX << ", " << vectorBY << ")" << endl << endl;
    cout << "Press Enter to return to the main menu" << endl << endl << endl;
    cin.get();
    break;
```

Rotating a point around the Y-Axis

Enter the X value of the first set of co-ordinates: 1

Enter the Y value of the first set of co-ordinates: 2

Enter the Angle you wish to rotate the shape: 60

The new co-ordinates after being rotated are (-0.342792, 1.60002)

Press Enter to return to the main menu

Operation 7 – Mean and Standard Deviation

Theory of Operation

The mean (or average) of a series of numbers is calculated by dividing the sum of the numbers with the amount of individual values. To solve the Standard Deviation, the mean of the values must be calculated first. Then take away the mean from each of the entered values and square each result. Next, calculate the mean of the new figures, and finally calculate the square root of that value to discover the Standard Deviation. [2]

Implementation of Code

First, the user enters 5 numbers into the system that stores the values in an array.

```
//User inputs values
counterX = 0;
while (counterX < 5)
{
    cout << "Enter a number: ";
    getline(cin, input);
    stream.str(input);
    stream >> stdDevArray[counterX];
    stream.clear();
    counterX = counterX + 1;
}
```

Next, the code calculates, stores and outputs the mean of the entered numbers.

```
//Code calculates mean
mean = (stdDevArray[0] + stdDevArray[1] + stdDevArray[2] + stdDevArray[3] + stdDevArray[4]) / 5;
cout << endl << "The Mean of the entered values is " << mean << endl;
```

Finally, the code calculates the Standard Deviation of the entered numbers.

```
//Code calculates Standard Deviation (stores values in same array)
counterX = 0;
while (counterX < 5)
{
    stdDevArray[counterX] = (stdDevArray[counterX] - mean) * (stdDevArray[counterX] - mean);
}

standDev = sqrt((stdDevArray[0] + stdDevArray[1] + stdDevArray[2] + stdDevArray[3] + stdDevArray[4]) / 5);
cout << "The Standard Deviation of the entered values is " << standDev << endl << endl;
cout << "Press Enter to return to the main menu" << endl << endl << endl;
cin.get();
break;
```

Testing

Test Number	Description	Expected Result	Actual Result	Pass/Fail
1	Begin the process of solving Mean and Standard Deviation	When user enters 7 into the option, the code for solving mean and Standard Deviation triggers	Code for solving mean and Standard Deviation triggers when 7 is entered	Pass
2	User enters values	User should be able to enter 5 values to be used.	All values are entered successfully	Pass
3	Code calculates mean	Code should calculate the mean of the 5 entered values and display it	Mean was successfully outputted.	Pass
4	Code calculates Standard Deviation	Code should calculate the standard Deviation of the 5 entered values and display it	Program froze after solving the Mean of the entered values	Fail – During the while loop, there was no line of code to increase the counter after each loop and so was stuck in an infinite loop.
5	Code calculates Standard Deviation	Code should calculate the standard Deviation of the 5 entered values and display it	Standard deviation was successfully outputted.	Pass

Here is a screengrab of the finished code and of it working in a console.

```

case 7: //7. Mean, Mode and Standard Deviation of a list of integers

    cout << "Calculate Mean and Standard Deviation of a list of integers" << endl << endl;

    //User inputs values
    counterX = 0;
    while (counterX < 5)
    {
        cout << "Enter a number: ";
        getline(cin, input);
        stream.str(input);
        stream >> stdDevArray[counterX];
        stream.clear();
        counterX = counterX + 1;
    }

    //Code calculates mean
    mean = (stdDevArray[0] + stdDevArray[1] + stdDevArray[2] + stdDevArray[3] + stdDevArray[4]) / 5;
    cout << endl << "The Mean of the entered values is " << mean << endl;

    //Code calculates Standard Deviation (stores values in same array)
    counterX = 0;
    while (counterX < 5)
    {
        stdDevArray[counterX] = (stdDevArray[counterX] - mean) * (stdDevArray[counterX] - mean);
        counterX = counterX + 1;
    }

    standDev = sqrt((stdDevArray[0] + stdDevArray[1] + stdDevArray[2] + stdDevArray[3] + stdDevArray[4]) / 5);
    cout << "The Standard Deviation of the entered values is " << standDev << endl << endl;
    cout << "Press Enter to return to the main menu" << endl << endl << endl;
    cin.get();
    break;

```

Calculate Mean and Standard Deviation of a list of integers

Enter a number: 1
Enter a number: 2
Enter a number: 3
Enter a number: 4
Enter a number: 5

The Mean of the entered values is 3
The Standard Deviation of the entered values is 1.41421
Press Enter to return to the main menu



Operation 8 – Union and Intersection of 2 sets

Theory of Operation

Union and Intersection, which are represented by $\cup$ and $\cap$ respectively, are used to identify data within multiple sets. Union highlights all the unique values stored in all the selected sets, not outputting duplicate numbers in the listing, while Intersection only highlights those values which are common between the selected sets. [3]

Implementation of Code

First the code allows the user to enter 4 values each into 2 sets.

```
//User inputs values
counterX = 0;
while (counterX < 4)
{
    cout << "Enter a number into the first set: ";
    getline(cin, input);
    stream.str(input);
    stream >> setA[counterX];
    stream.clear();
    counterX = counterX + 1;
}
counterX = 0;
while (counterX < 4)
{
    cout << "Enter a number into the second set: ";
    getline(cin, input);
    stream.str(input);
    stream >> setB[counterX];
    stream.clear();
    counterX = counterX + 1;
}
```

Next, the code calculates the values that are in union between the 2 sets.

```

//Code solves Union
cout << "The values in Union of Set A and B are: ";
counterX = 0;
counterY = 0;
while (counterX < 4)
{
    cout << setA[counterX] << " ";
    while (counterY < 4)
    {
        if (setA[counterX] == setB[counterY])
        {
            counterY = counterY + 1;
        }
        else
        {
            cout << setB[counterY] << " ";
            counterY = counterY + 1;
        }
    }
    counterX = counterX + 1;
}

```

Finally, the code calculates the values that are in Intersection between the 2 sets.

```

//Code solves Intersection
cout << "The values in Intersection of Set A and B are: ";
counterX = 0;
counterY = 0;
while (counterX < 4)
{
    while (counterY < 4)
    {
        if (setA[counterX] == setB[counterY])
        {
            cout << setA[counterX] << " ";
            counterY = counterY + 1;
        }
        else
        {
            counterY = counterY + 1;
        }
    }
    counterX = counterX + 1;
}

```

Testing

Test Number	Description	Expected Result	Actual Result	Pass/Fail
1	Begin the process of solving Union and Intersection	When user enters 8 into the option, the code for solving Union and Intersection triggers	Code for solving Union and Intersection triggers when 7 is entered	Pass
2	User enters values	User should be able to enter 8 values, 4 values on each set.	All values are entered successfully	Pass
3	Code calculates union of 2 sets	Code should calculate the union of the 4 entered values and display it	Union was successfully outputted.	Pass
4	Code calculates Intersection of 2 sets	Code should calculate the Intersection of the 2 sets and display it	Code did not output any values, even though there were common values	Fail – The counter of the nested while loop was not being reset when the outer while command looped
5	Code calculates Intersection of 2 sets	Code should calculate the Intersection of the 2 sets and display it	Intersection was successfully outputted.	Pass

Here is a screengrab of the finished code and of it working in a console.

```

case 8: //8. Union and Intersection of 2 sets

    cout << "Calculate Union and Intersection of 2 sets" << endl << endl;

    //User inputs values
    counterX = 0;
    while (counterX < 4)
    {
        cout << "Enter a number into the first set: ";
        getline(cin, input);
        stream.str(input);
        stream >> setA[counterX];
        stream.clear();
        counterX = counterX + 1;
    }
    counterX = 0;
    while (counterX < 4)
    {
        cout << "Enter a number into the second set: ";
        getline(cin, input);
        stream.str(input);
        stream >> setB[counterX];
        stream.clear();
        counterX = counterX + 1;
    }

    //Code solves Intersection
    cout << "The values in Intersection of Set A and B are: ";
    counterX = 0;
    while (counterX < 4)
    {
        counterY = 0;
        while (counterY < 4)
        {
            if (setA[counterX] == setB[counterY])
            {
                cout << setA[counterX] << " ";
                counterY = counterY + 1;
            }
            else
            {
                counterY = counterY + 1;
            }
        }
        counterX = counterX + 1;
    }

    break:

```

Calculate Union and Intersection of 2 sets

```

Enter a number into the first set: 1
Enter a number into the first set: 2
Enter a number into the first set: 3
Enter a number into the first set: 4
Enter a number into the second set: 4
Enter a number into the second set: 5
Enter a number into the second set: 6
Enter a number into the second set: 7
The values in Intersection of Set A and B are: 4
Press Enter to return to the main menu

```

References

- [1] J. Bird, Basic Engineering Mathematics, Routledge Ltd, 2014.
- [2] B. & V. Verth, Essential Mathematics for Games and interactive Applications, Burlington: Morgan Kaufmann, 2008.
- [3] S. Doyle, WJEC AS Mathematics Statistics 1 (S1), Essex: Illuminate, 2013.

Math SwDv Coding Assignment

GRADEMARK REPORT

FINAL GRADE

GENERAL COMMENTS

Instructor

60/100

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13



Comment 1

no, wrong

PAGE 14

PAGE 15



Comment 2

no, wrong



Comment 3

error



Comment 4

ok but limited to 5 elements. Generalize if you can.



Comment 5

Well done for reporting this error.



Comment 6

Mode??



Comment 7

No union tests?

GRADING FORM: SMU GENERIC_GJ

RHYS WHATTY**60****RESEARCH-INFORMED LITERATURE**

Most Rfs appropriate. Some issues with coded calculations. Do detailed calc in report and use this to compare.

3**KNOWLEDGE AND UNDERSTANDING**

Distance between two points -you squared the difference twice! Test values leading to $x_{diff}=y_{diff}=1$ will not reveal the error! Dot product - test reveals problem $(1,2) \cdot (3,4)=11$ not 14! console output- cross product calc wrong! Some operations missing/flawed.

12**ANALYSIS**

It would help you to show the detailed calculations done for all tasks and use this to test.

12**PRACTICAL APPLICATION AND DEP.**



Most attempted - some errors in testing not identified/corrected. Otherwise a fair attempt.

30

SKILLS FOR PROFESSIONAL PRAC.



Report structured. Screen shots presented and refs included.

3