# Assignment Description

Write an application that solves a system of linear equations. Have the user input the coefficients of the two lines. Remember that the definition of a line is:  Ax + By = C.  You can use any method you wish to solve. The program should output the intersection of the two lines if they do indeed intersect, indicate if the two lines are parallel, or if they are the same line.

# GitHub URL (optional)

<https://github.com/wesleyhixon/Programming-Assignments/tree/4aaea7fdb66f08ec43d1cb2d05deb9a2ee8a50a6/M08%20Final%20Project>

# Readme Documentation

Input Information: Input is the coefficients of two equations of form Ax + By = C

Output Information: Output is the intersection of those equations if they do intersect, or whether they are parallel, perpendicular, or the same line.

# Flowchart Screen Shots (optional)

Screen shot(s) here

# UML and Use Case Diagrams (optional)

Screen shot(s) here

# Source Code of All files (.h, .cpp)

1. #include <iostream>
2. using namespace std;
3. /\*
4. Name: System of Linear Equations
5. Author: Wesley Hixon
6. Due Date: 07/25/2024
7. Purpose: Make a program which solves a system of linear equations given 2 lines of form Ax+By=C
8. \*/
10. void solveSystem(double equation1[], double equation2[]);
11. void copyEquation(const double equation[], double copiedEquation[]);
12. void multiplyEquation(double equation[], double factor);
13. void addEquations(double equation1[], double equation2[], double outputEquation[]);
14. double validateInput(string prompt);
15. const int xCoefficient = 0;     // Const variables to represent the coefficient array indexes
16. const int yCoefficient = 1;
17. const int zCoefficient = 2;
19. int main(){
20. double equation1[3], equation2[3];  // Declaring equation arrays
22. cout << "Welcome to the system of equations solver!" << endl    // Welcoming user to my humble program
23. << "This program will solve any system of 2 equations in the form: Ax + By = C" << endl;
24. // Getting input for the first equation
25. equation1[xCoefficient] = validateInput("Enter A for the first equation: ");
26. equation1[yCoefficient] = validateInput("Enter B for the first equation: ");
27. equation1[zCoefficient] = validateInput("Enter C for the first equation: ");
28. // Getting input for the second equation
29. equation2[xCoefficient] = validateInput("Enter A for the second equation: ");
30. equation2[yCoefficient] = validateInput("Enter B for the second equation: ");
31. equation2[zCoefficient] = validateInput("Enter C for the second equation: ");
33. cout << "The lines you entered are:" << endl
34. << equation1[xCoefficient] << "x + " << equation1[yCoefficient] << "y = " << equation1[zCoefficient] << endl
35. << equation2[xCoefficient] << "x + " << equation2[yCoefficient] << "y = " << equation2[zCoefficient] << endl;
36. // Solving the system
37. solveSystem(equation1, equation2);
39. return 0;
40. }
41. // This function solves a system of equations given 2 equations in form Ax + By = C
42. // It uses the elimination method through cancelling one variable to find the other
43. void solveSystem(double equation1[], double equation2[]){
44. double equation3[3];                                                // This equation is for multiplying while preserving original equations
45. double ySolution, xSolution;                                        // Solutions
46. double slope1 = equation1[xCoefficient] / equation1[yCoefficient];  // Slopes
47. double slope2 = equation2[xCoefficient] / equation2[yCoefficient];
48. bool x1Smaller;                                                     // My method requires knowing which x coefficient is smaller
49. double factorToMultiply;                                            // This is the multiplication factor to cancel out the x coefficients
50. if(slope1 == -(1 / slope2)){
51. cout << "The lines are perpendicular" << endl;                  // If the slopes are opposite reciporicals, the lines are perpendicular
52. }
53. else if(slope1 == slope2){
54. if(equation1[zCoefficient] == equation2[zCoefficient]){
55. cout << "The lines are the same";                           // If the equations are the same, the lines are the same
56. return;
57. }
58. else{
59. cout << "The lines are parallel" << endl;                       // If the slopes are the same, the lines are parallel
60. return;
61. }
62. }

65. if(equation1[xCoefficient] < equation2[xCoefficient]){                      // Determining which X-Coefficient is smaller
66. factorToMultiply = equation2[xCoefficient] / equation1[xCoefficient];   // The factor to multiply is the bigger X divided by the smaller X
67. copyEquation(equation1, equation3);
68. x1Smaller = true;
69. }
70. else{
71. factorToMultiply = equation1[xCoefficient] / equation2[xCoefficient];
72. copyEquation(equation2, equation3);
73. x1Smaller = false;
74. }
75. if(equation1[xCoefficient] > 0 && equation2[xCoefficient] > 0
76. || equation1[xCoefficient] < 0 && equation2[xCoefficient] < 0){
77. factorToMultiply \*= -1;                                                 // If both x coefficients are positive or negative, invert the factor so the x coefficients cancel out
78. }
79. multiplyEquation(equation3, factorToMultiply);                              // Multiplying the copied equation to make the xCoefficient inverse
80. if(x1Smaller){                                                              // Adding the new equation to the larger X equation to create a third equation with a cancelled out X
81. addEquations(equation2, equation3, equation3);
82. }
83. else{
84. addEquations(equation1, equation3, equation3);
85. }
86. ySolution = equation3[zCoefficient] / equation3[yCoefficient];              // Solving for Y solution
87. copyEquation(equation1, equation3);                                         // Copying equation 1 to solve for X
88. equation3[yCoefficient] \*= ySolution;                                       // Multiplying Y coefficient by Y solution
89. equation3[zCoefficient] -= equation3[yCoefficient];                         // Subtracting Y coefficient to leave X on its own
90. xSolution = equation3[zCoefficient] / equation3[xCoefficient];              // Dividing Z coefficient by X coefficient to get X solution
91. cout << "The lines intersect at (" << xSolution << ", " << ySolution << ")";// Outputting solution
92. return;
93. }
94. // This function copies an equation into another
95. void copyEquation(const double equation[], double copiedEquation[]){
96. for(int i=0; i<3; i++){
97. copiedEquation[i] = equation[i];
98. }
99. }
100. // This function multiplies an entire equation by a factor
101. void multiplyEquation(double equation[], double factor){
102. for(int i=0; i<3; i++){
103. equation[i] \*= factor;
104. }
105. }
106. // This function adds two equations and outputs it to outputEquation[]
107. void addEquations(double equation1[], double equation2[], double outputEquation[]){
108. for(int i = 0; i < 3; i++){
109. outputEquation[i] = equation1[i] + equation2[i];
110. }
111. }
112. // Function to get valid double input
113. double validateInput(string prompt){
114. double input;
116. bool valid = false;
117. while(!valid){
118. cout << prompt;
119. cin >> input;
120. if(!cin){
121. // If user inputs a letter or a symbol
122. cout << "Please enter a valid number. Try again." << endl;
123. cin.clear();
124. cin.ignore(100000, '\n');
125. }
126. else{
127. // If user inputs a valid number
128. valid = true;
129. return input;
130. }
131. }
132. return input;
133. }

# Three Use Case Screen Shots

The can all be in a single screen shot

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

A computer screen shot of a black screen

Description automatically generated