**Visual Programming**

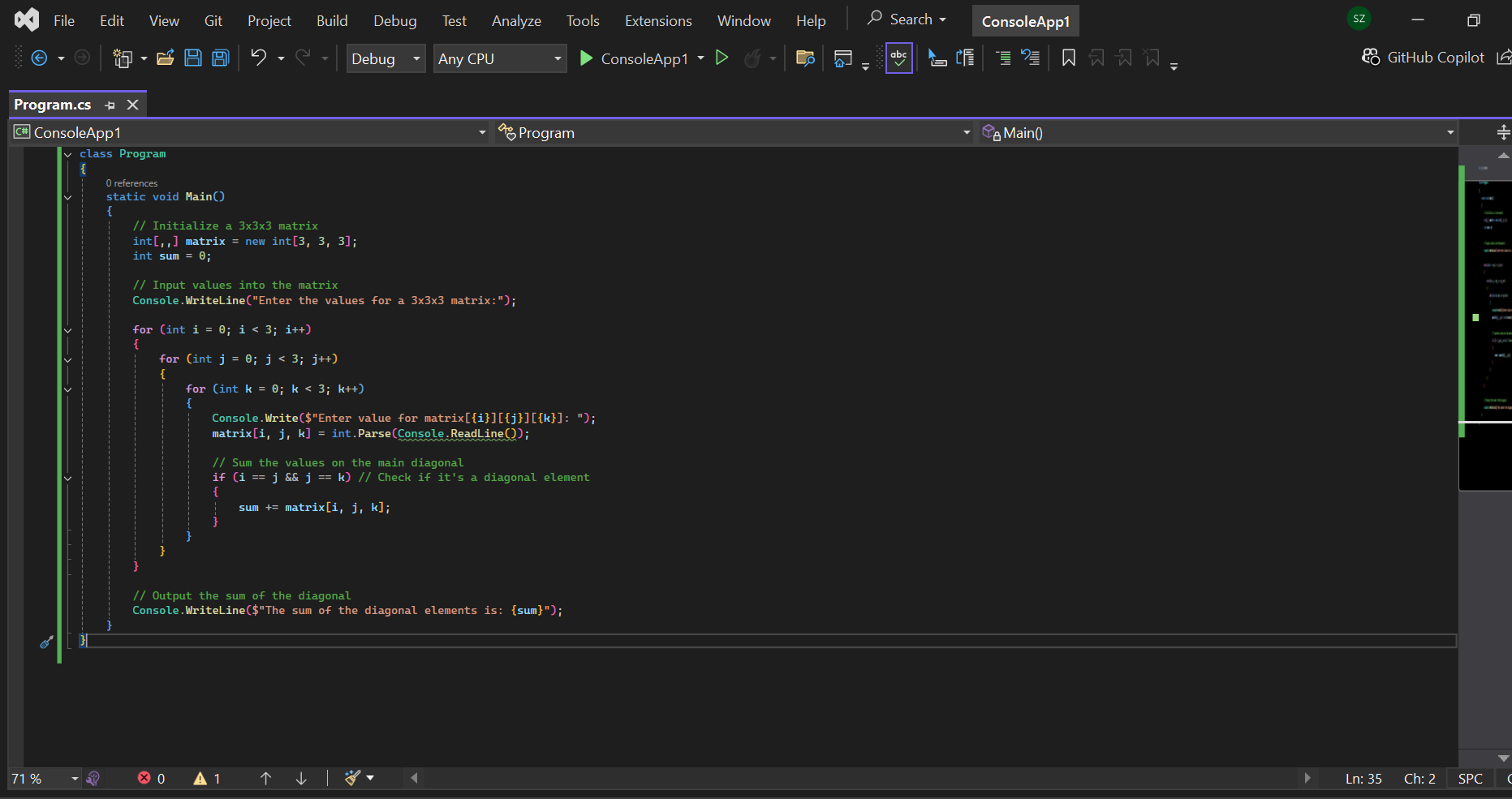
# **Assignment no. 1**

***Roll No. 233530***

***Submit to: Mam Atka Ali***

***Submit by: Somal Atta Baloch***

**Task:**



**Report:**

**1. Introduction**

This report provides an overview and explanation of a C# program designed to compute the sum of diagonal elements in a three-dimensional (3x3x3) matrix. The program prompts the user to input values for the matrix and then sums the values of the elements that lie on the main diagonal.

**2. Problem Statement**

The task involves reading integer inputs to populate a 3x3x3 matrix and summing only the values of the elements along the main diagonal. In a three-dimensional matrix, the main diagonal is characterized by elements where all three indices (i, j, k) are equal, i.e., matrix[i][i][i].

**3. Algorithm Outline**

The algorithm performs the following steps:

1. **Matrix Initialization**:
   * A 3x3x3 matrix is declared and initialized using an integer array.
   * An integer variable sum is initialized to zero for storing the sum of the diagonal elements.
2. **Input Values into the Matrix**:
   * The program prompts the user to input values for each element in the matrix using three nested loops, where i, j, and k represent the indices of the matrix.
   * For each element matrix[i][j][k], the user is prompted to enter an integer value.
3. **Diagonal Check and Summation**:
   * While populating the matrix, the program checks if the current element is part of the main diagonal by verifying that i == j == k.
   * If this condition is met, the value of matrix[i][j][k] is added to the sum variable.
4. **Output the Result**:
   * After all elements are entered, the program outputs the sum of the diagonal elements.

**4. Explanation of Key Components**

* **Matrix Initialization**: The matrix matrix[,,] is a three-dimensional array with dimensions [3, 3, 3]. The variable sum is initialized to store the sum of diagonal elements.
* **Input Handling**: The nested for loops iterate over each index of the matrix. The Console.ReadLine() function is used to take input for each element, which is then parsed into an integer using int.Parse().
* **Diagonal Check**: The condition if (i == j && j == k) ensures that only the elements where all three indices are equal are included in the sum. This condition identifies the main diagonal of the 3D matrix.
* **Output**: The final sum is printed using Console.WriteLine().

**5. Example of Program Execution**

##### **Input**:

The user enters the following values for the matrix:

| matrix[0][0][0] = 1 | matrix[0][0][1] = 2 | matrix[0][0][2] = 3 | | matrix[1][1][1] = 5 | matrix[1][1][0] = 4 | matrix[1][1][2] = 6 | | matrix[2][2][2] = 9 | matrix[2][2][1] = 7 | matrix[2][2][0] = 8 |

##### **Output**:

The program calculates the sum of the diagonal elements:  
1 (from matrix[0][0][0]) + 5 (from matrix[1][1][1]) + 9 (from matrix[2][2][2]) = 15

**6. Conclusion**

The C# program efficiently calculates the sum of diagonal elements of a 3x3x3 matrix. By leveraging nested loops and conditions to identify diagonal elements, the program processes user input and provides the correct result. This code serves as an introduction to working with three-dimensional arrays and applying conditional logic to matrix operations.