

Certainly! Here's a simple C++ program that fulfills the given requirements.

```
include<iostream>
const int rows = 2;
const int cols = 2;

// Function to initialize the matrix with zeros
void MatrixInitializer(int matrix[][cols]) {
    for (int i = 0; i < rows; ++i) {
        for (int j = 0; j < cols; ++j) {
            matrix[i][j] = 0;
        }
    }
}

// Function to add two matrices
void AddMatrix(int result[][cols], int matrix1[][cols], int matrix2[][cols]) {
    for (int i = 0; i < rows; ++i) {
        for (int j = 0; j < cols; ++j) {
            result[i][j] = matrix1[i][j] + matrix2[i][j];
        }
    }
}

// Function to multiply a matrix by a scalar
void MultMatrixByScalar(int result[][cols], int scalar, int matrix[][cols]) {
    for (int i = 0; i < rows; ++i) {
        for (int j = 0; j < cols; ++j) {
            result[i][j] = scalar * matrix[i][j];
        }
    }
}

// Function to display a matrix
void DisplayMatrix(int matrix[][cols]) {
    for (int i = 0; i < rows; ++i) {
        for (int j = 0; j < cols; ++j) {
            std::cout << matrix[i][j] << " ";
        }
        std::cout << std::endl;
    }
    std::cout << std::endl;
}

int main() {
```

```

int A[rows][cols] = {{1, 2}, {3, 4}};
int B[rows][cols] = {{1, 0}, {0, 1}};
int result[rows][cols];

// Initialize matrices
MatrixInitializer(result);

// Display matrices A and B
std::cout << "Matrix A:" << std::endl;
DisplayMatrix(A);
std::cout << "Matrix B:" << std::endl;
DisplayMatrix(B);

// Compute 2*B
MultMatrixByScalar(result, 2, B);
std::cout << "2 * B:" << std::endl;
DisplayMatrix(result);

// Add matrices A and 2*B
AddMatrix(result, A, result);
std::cout << "Resultant matrix [A + 2 * B]:" << std::endl;
DisplayMatrix(result);

return 0;
}

```

This C++ program defines functions to initialize, add, multiply by scalar, and display matrices. In the main function, it initializes matrices A and B, performs the specified operations, and displays the results.