

# Session 10

## **Operator Overloading (w/ Member Function)**

**Session #10 is the continuation of Session 8, Part 2**

Session 8 / Part 2

**Operator Overloading (w/ Friend keyword)**

## 1) What is the concept?

If you overload an operator w/ a Class Member Function, then compiler translate " $A + B$ " to " $A.operator+(B)$ ". That means, the operator overloaded member function gets invoked on the first operand. Consequently, you can only overload an operator for a class with Member function if the left side of the operand is not from that class type. For instance, for the expression " $10 + A$ " you cannot overload  $+$  operator with member functions. You have to use non-member function and make it friend to that class, if necessary.

[Read more about Concept.](#)

## 2) Why is it Necessary?

There are some cases that you have to overload an operand using member functions. Using non-member friend function might cause issues.

[Read More: The copy assignment operator must be overloaded as a member function.](#)

## 3) How does the syntax work?

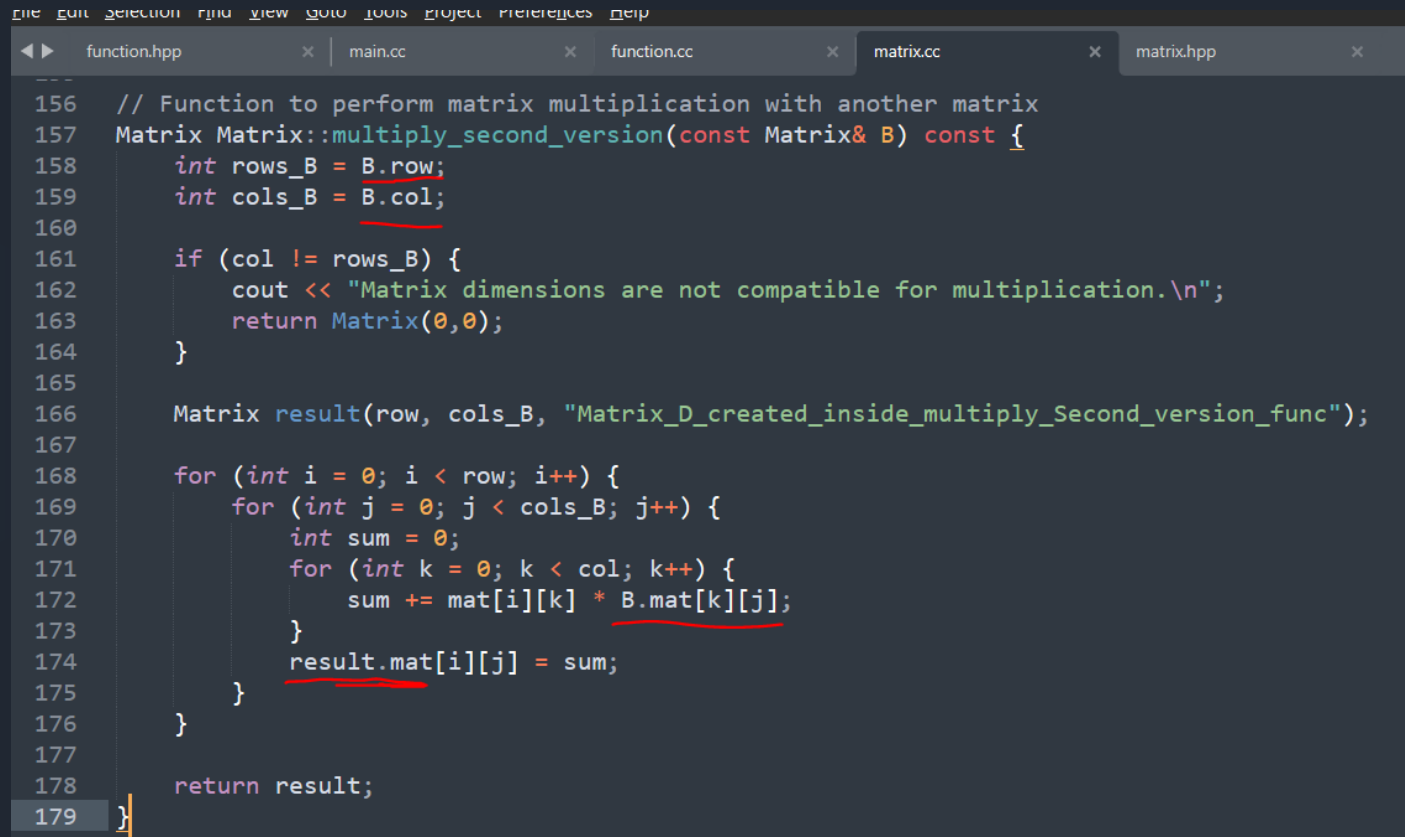
Nothing new.

## Lab Exercises:

- 1) (will be implemented by me) **Overload + with member function for Matrix class ( A + B )**
- 2) (will be implemented by me) **Overload + with non-member friend function for Matrix class ( A + B )**
- 3) (You will implement this) **Overload << with non-member friend function for Matrix class ( cout << A ; )**

(note that you cannot overload << as a member function, why?)

Don't forget that a class itself has access to the private attributes of the object that it is called on (using this keyword) as well as having access to the private attributes of the objects from the same class type.



```
156 // Function to perform matrix multiplication with another matrix
157 Matrix Matrix::multiply_second_version(const Matrix& B) const {
158     int rows_B = B.row;
159     int cols_B = B.col;
160
161     if (col != rows_B) {
162         cout << "Matrix dimensions are not compatible for multiplication.\n";
163         return Matrix(0,0);
164     }
165
166     Matrix result(row, cols_B, "Matrix_D_created_inside_multiply_Second_version_func");
167
168     for (int i = 0; i < row; i++) {
169         for (int j = 0; j < cols_B; j++) {
170             int sum = 0;
171             for (int k = 0; k < col; k++) {
172                 sum += mat[i][k] * B.mat[k][j];
173             }
174             result.mat[i][j] = sum;
175         }
176     }
177
178     return result;
179 }
```

Compare this function w/ the function we wrote in session 8. They are the same. But, but, we better use set/get functions to work with attributes, if provided (implemented).