Session 6

Revisiting 2D Matrix Multiplication with OOP mindset

Agenda:

1. Discussing matrix.cc file from last session (We discussed main.cc & matrix.hpp but we only had matrix.o, the binary file, not the implementation of it, which is matrix.cc).
2. + Discussing some minor changes that I have made in matrix.cc file.
   + Implementing a second version of matrix multiplication "Matrix Matrix::multiply_second_version(const Matrix& B) const "
3. Discussing the behavior of the new version of matrix multiplication.
4. Making improvements on the previous part.

Part 1, Discussing matrix.cc file

```
class Matrix {
public:
    // Constructor
    Matrix(int rows, int cols);

    // Destructor
    ~Matrix();

    // Function to set an entry in the matrix
    void set_entry(int row, int col, int value);

    // Function to get an entry from the matrix
    int get_entry(int row, int col) const;

    // Function to get the number of rows in the matrix
    int get_rows() const;

    // Function to get the number of columns in the matrix
    int get_cols() const;

    // Function to print the entire matrix
    void print() const;

    // Function to prompt the user to enter values for the matrix
    void receive();

    // Function to perform matrix multiplication with another matrix
    Matrix* multiply(const Matrix& B) const;
private:
    int row;
    int col;
    int** mat;

    // Helper function to allocate memory for the matrix
    void allocate_memory();

    // Helper function to deallocate memory for the matrix
    void deallocate_memory();
};
```

Notice that helper functions are private.

Part 1, Discussing matrix.cc file

Sample output (expected behavior)

```
!./program

Enter the number of rows for Matrix A: 2
Enter the number of columns for Matrix A: 2
Enter values for the matrix:
Mat[0][0]: 1
Mat[0][1]: 2
Mat[1][0]: 3
Mat[1][1]: 4
Matrix A:
1 2
3 4
Enter the number of rows for Matrix B: 2
Enter the number of columns for Matrix B: 1
Enter values for the matrix:
Mat[0][0]: 3
Mat[1][0]: 7
Matrix B:
3
7
Result of Matrix Multiplication (C = A * B):
17
37


The deallocation of memory function is triggered! The matrix that is being deallocated is as follows:
17
37
The deallocation of memory function is triggered! The matrix that is being deallocated is as follows:
3
7
The deallocation of memory function is triggered! The matrix that is being deallocated is as follows:
1 2
3 4
```

Part 2, Discussing some minor changes that I have made in matrix.cc file.

```
private:
    int row;
    int col;
    string name;
    int** mat;
```

```
// Constructor
Matrix::Matrix(int rows, int cols) : row(rows), col(cols), name("NotGivenName") {
    allocate_memory();
}

// **** Recently added function ****
// Overloading Constructor function
Matrix::Matrix(int rows, int cols, string n) : row(rows), col(cols), name(n) {
    allocate_memory();
}

// Destructor
Matrix::~Matrix() {...
}

// Function to allocate memory for the matrix
void Matrix::allocate_memory() {...
}

// **** Recently modified function  *****
// Function to deallocate memory for the matrix
void Matrix::deallocate_memory() {
    cout << "The deallocation_of_memory function is triggered! Matrix "<< this->name << " is being deallocated.\n";
    //cout << "The deallocation of memory function is triggered! The matrix that is being deallocated is as follows:\n";
    //this->print()
    for (int i = 0; i < row; i++) {
        delete[] mat[i];
    }
    delete[] mat;
}
```

Part 2, Implementing a second version of matrix multiplication

```
131  // Function to perform matrix multiplication with another matrix
132  Matrix Matrix::multiply_second_version(const Matrix& B) const {
133      int rows_B = B.get_rows();
134      int cols_B = B.get_cols();
135
136      if (col != rows_B) {
137          cout << "Matrix dimensions are not compatible for multiplication.\n";
138          return Matrix(0, 0);
139      }
140
141      Matrix result(row, cols_B, "Matrix_D_created_inside_multiply_Second_version_func");
142
143      // Complete here (only 9 lines of code. Hint: You can see how previous function, multiply, is implemented.)
144      // write your code between START and END.
145      // START writing your code.
146
147
148      // END.
149
150      return result;
151  }
```

+ Read the main.cc (in part2 folder) and guess what would be the output.
+ Then run the code to compare the output you get and what you guessed.

Part 3, Discussing matrix.cc file

MINGW64:/f/6_s6

```
ASUS@MyLaptop MINGW64 /f/6_s6
$ ./exe.exe
Enter the number of rows for Matrix A: 2
Enter the number of columns for Matrix A: 2
Enter values for the matrix:
Mat[0][0]: 1
Mat[0][1]: 2
Mat[1][0]: 3
Mat[1][1]: 4
Matrix A:
1 2
3 4
Enter the number of rows for Matrix B: 2
Enter the number of columns for Matrix B: 2
Enter values for the matrix:
Mat[0][0]: 1
Mat[0][1]: 0
Mat[1][0]: 0
Mat[1][1]: 1
Matrix B:
1 0
0 1
Result of Matrix Multiplication (C = A * B):
1 2
3 4


The deallocation_of_memory function is triggered! Matrix matrix_C_created_inside_multiply is being deallocated.
The deallocation_of_memory function is triggered! Matrix Matrix_D_created_inside_multiply_Second_version_func is being deallocated.
Result of Matrix Multiplication (D = A * B):
16430304 16428592
16428592 16387168


The deallocation_of_memory function is triggered! Matrix Matrix_D_changed_name_inside_main is being deallocated.
The deallocation_of_memory function is triggered! Matrix B is being deallocated.
The deallocation_of_memory function is triggered! Matrix A is being deallocated.

ASUS@MyLaptop MINGW64 /f/6_s6
$
```

Output on my windows operating system

Part 3, Discussing matrix.cc file

```
ardestani@17ninja:~/summer2620/6_s6$ ls
main.cc  matrix.cc  matrix.hpp
ardestani@17ninja:~/summer2620/6_s6$ g++ -Wall -std=c++11 -o exe main.cc matrix.cc
ardestani@17ninja:~/summer2620/6_s6$ ./exe
Enter the number of rows for Matrix A: 2
Enter the number of columns for Matrix A: 2
Enter values for the matrix:
Mat[0][0]: 1
Mat[0][1]: 2
Mat[1][0]: 3
Mat[1][1]: 4
Matrix A:
1 2
3 4
Enter the number of rows for Matrix B: 2
Enter the number of columns for Matrix B: 2
Enter values for the matrix:
Mat[0][0]: 1
Mat[0][1]: 0
Mat[1][0]: 0
Mat[1][1]: 1
Matrix B:
1 0
0 1
Result of Matrix Multiplication (C = A * B):
1 2
3 4


The deallocation_of_memory function is triggered! Matrix matrix_C_created_inside_multiply is being deallocated.
The deallocation_of_memory function is triggered! Matrix Matrix_D_created_inside_multiply_Second_version_func is being deallocated.
Result of Matrix Multiplication (D = A * B):
-669833104 21851
131075 1


The deallocation_of_memory function is triggered! Matrix Matrix_D_changed_name_inside_main is being deallocated.
free(): double free detected in tcache 2
Aborted (core dumped)
ardestani@17ninja:~/summer2620/6_s6$
```

Result on University's Linux systems

Part 3, Discussing matrix.cc file

```
[5]  !g++ -Wall -std=c++11 -o exe main.cc matrix.cc
```

```
[6]  !./exe

Enter the number of rows for Matrix A: 2
Enter the number of columns for Matrix A: 2
Enter values for the matrix:
Mat[0][0]: 1
Mat[0][1]: 2
Mat[1][0]: 3
Mat[1][1]: 4
Matrix A:
1 2
3 4
Enter the number of rows for Matrix B: 2
Enter the number of columns for Matrix B: 2
Enter values for the matrix:
Mat[0][0]: 1
Mat[0][1]: 0
Mat[1][0]: 0
Mat[1][1]: 1
Matrix B:
1 0
0 1
Result of Matrix Multiplication (C = A * B):
1 2
3 4


The deallocation_of_memory function is triggered! Matrix matrix_C_created_inside_multiply is being deallocated.
The deallocation_of_memory function is triggered! Matrix Matrix_D_created_inside_multiply_Second_version_func is being deallocated.
Result of Matrix Multiplication (D = A * B):
```

Result on Google Colab Linux systems

Unexpected termination of the program!

Part 3, Why did we get that wild behavior from the code?

Elaborated on in Miro board

Part 4, Final Version

+ Now that you know why when and why a Copy Constructor is being called, and now that you know that the "Default Copy Constructor" does not work in our case, Implement a Copy Constructor yourself.