

Lab 3.2.2 Matrices and pointers – a step inside

Objectives

Improve the student's skills in:

- understanding internal matrix representation;
- using pointers to access a desired memory location;
- combining **for** loops in order to manipulate two-dimensional arrays.

Scenario

Forgive us for asking – did you successfully complete the previous lab? If not, please go back to it and try again. Believe us – you won't be able to solve this problem if you didn't solve the previous one. Sorry, but such is the tough life of a coder.

Take a look at the code. It does almost nothing – it simply outputs the contents of a 10×10 matrix. As the matrix is initially filled with zeros (why?), the output doesn't look very attractive.

This is why we need you here – your task is to fill the matrix with values that will turn it into a multiplication table. Easy? Too easy to be true. Let's raise the bar, just like we did before. You mustn't use brackets. You mustn't use indexing. Ergo, you must use pointers.

Okay, you can use brackets – once. And only in the declaration. Nowhere else. It is possible. Really.

Anyway, this is exactly what we want to see on your screen:

```
1  2  3  4  5  6  7  8  9 10
2  4  6  8 10 12 14 16 18 20
3  6  9 12 15 18 21 24 27 30
4  8 12 16 20 24 28 32 36 40
5 10 15 20 25 30 35 40 45 50
6 12 18 24 30 36 42 48 54 60
7 14 21 28 35 42 49 56 63 70
8 16 24 32 40 48 56 64 72 80
9 18 27 36 45 54 63 72 81 90
10 20 30 40 50 60 70 80 90 100
```

And this is the snippet you should expand on.

```
#include <iostream>

using namespace std;

int main(void) {

    int matrix[10][10] = { };

    // Insert your code here

    for(int i = 0; i < 10; i++) {
        for(int j = 0; j < 10; j++) {
            cout.width(4);
            cout << matrix[i][j];
        }
        cout << endl;
    }
    return 0;
}
```