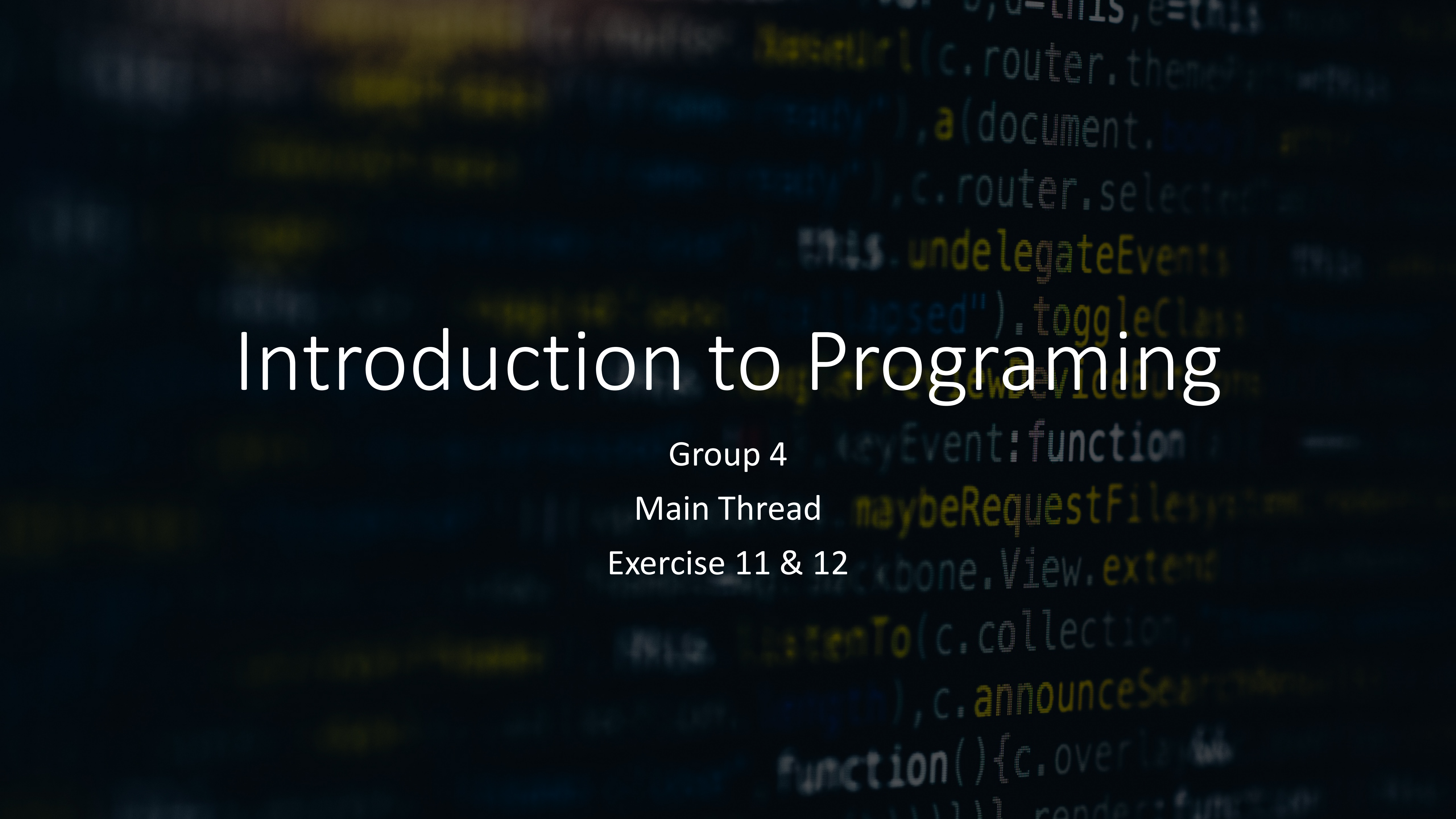


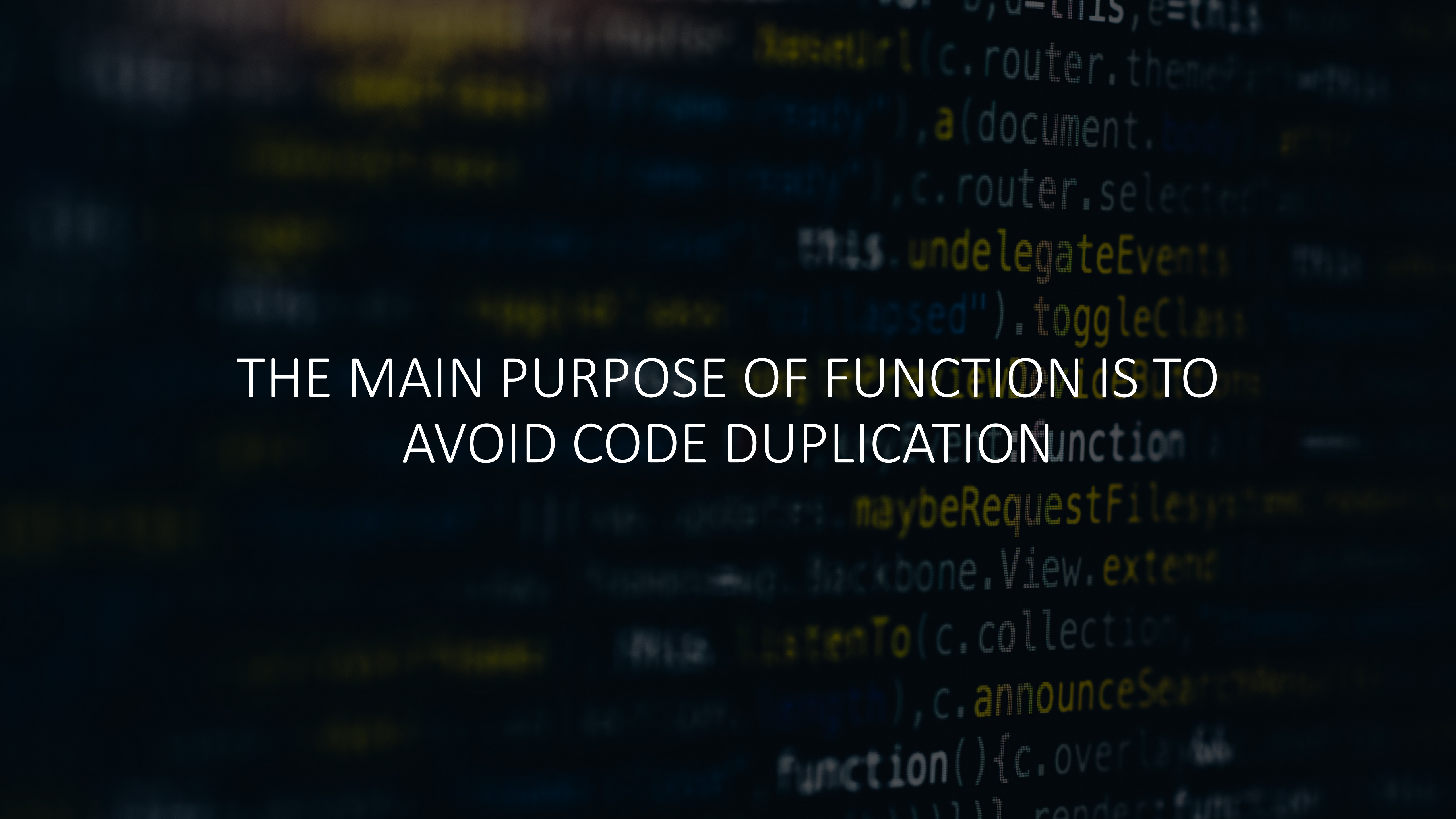
Introduction to Programing

Group 4

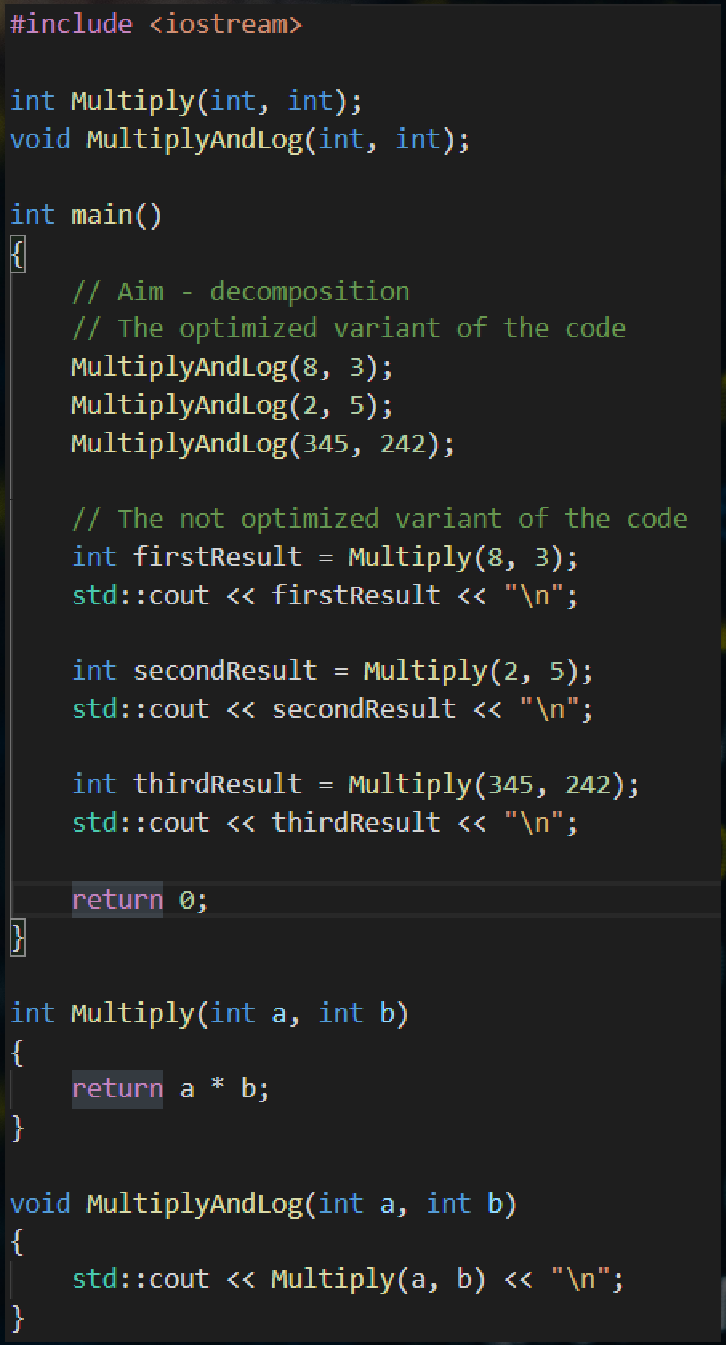
Main Thread

Exercise 11 & 12

FUNCTIONS

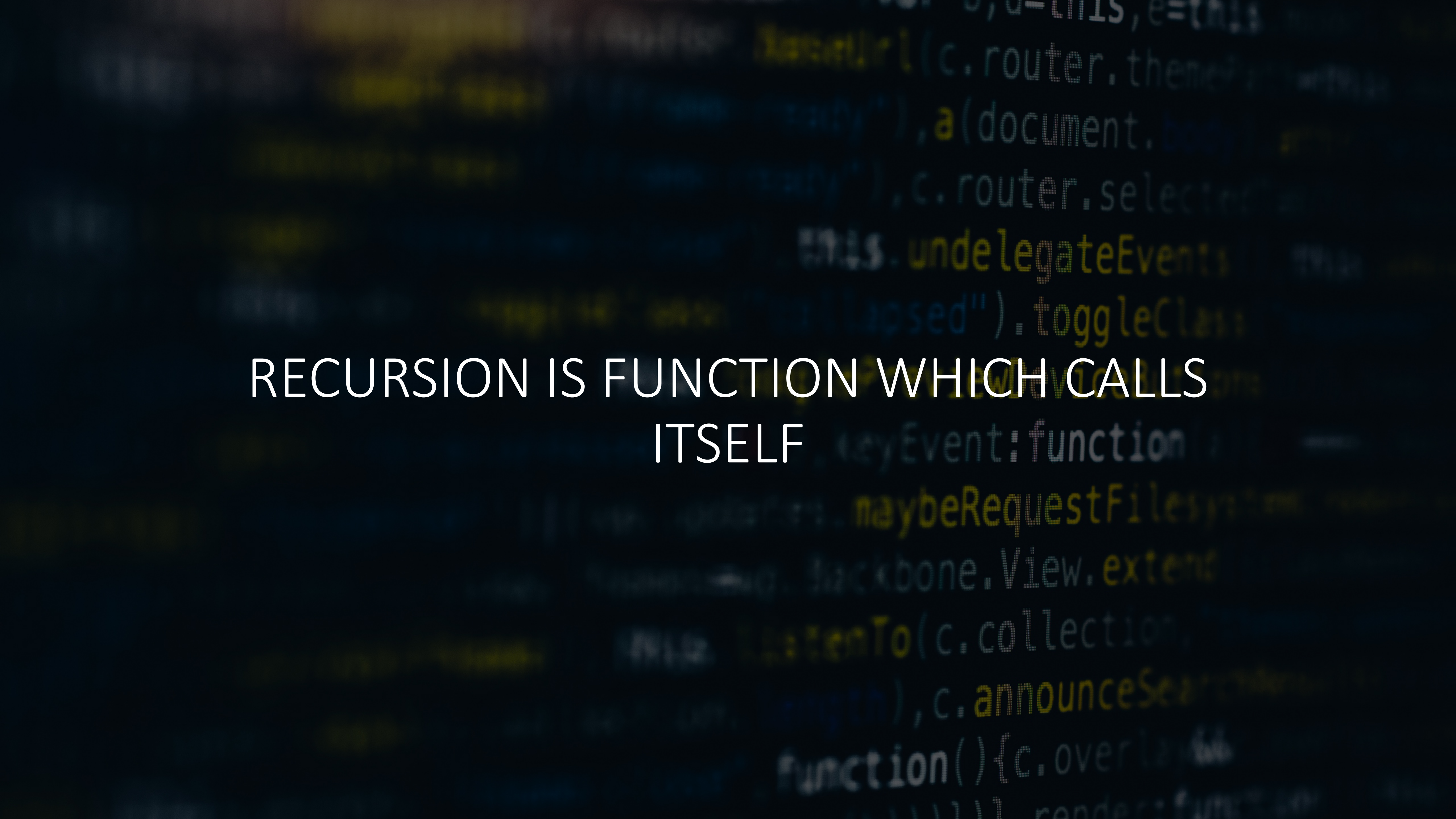


THE MAIN PURPOSE OF FUNCTION IS TO AVOID CODE DUPLICATION



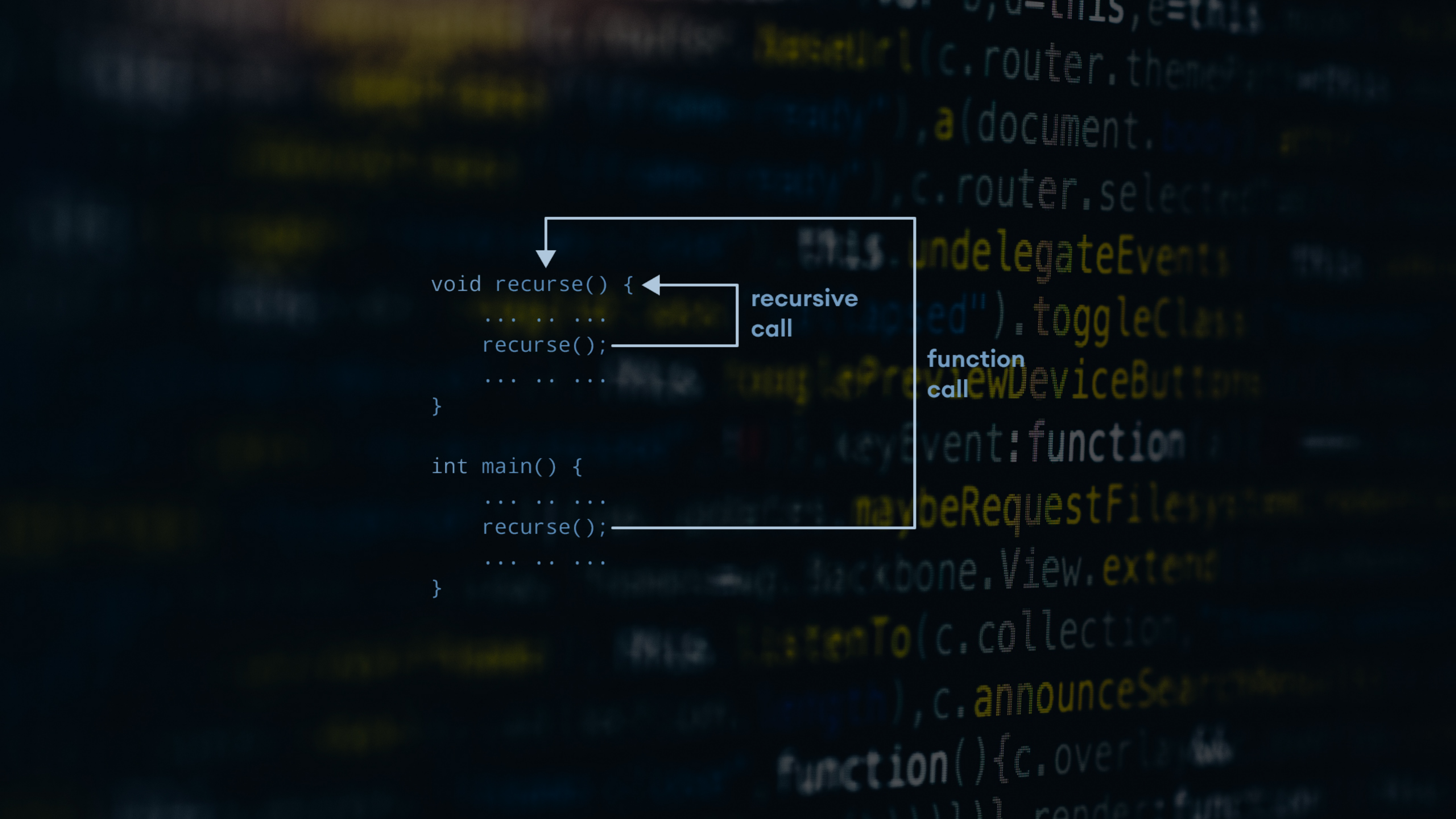
```cpp
#include <iostream>

int Multiply(int, int);
void MultiplyAndLog(int, int);

int main()
{
    // Aim - decomposition
    // The optimized variant of the code
    MultiplyAndLog(8, 3);
    MultiplyAndLog(2, 5);
    MultiplyAndLog(345, 242);

    // The not optimized variant of the code
    int firstResult = Multiply(8, 3);
    std::cout << firstResult << "\n";

    int secondResult = Multiply(2, 5);
    std::cout << secondResult << "\n";

    int thirdResult = Multiply(345, 242);
    std::cout << thirdResult << "\n";

    return 0;
}

int Multiply(int a, int b)
{
    return a * b;
}

void MultiplyAndLog(int a, int b)
{
    std::cout << Multiply(a, b) << "\n";
}
```

RECURSION



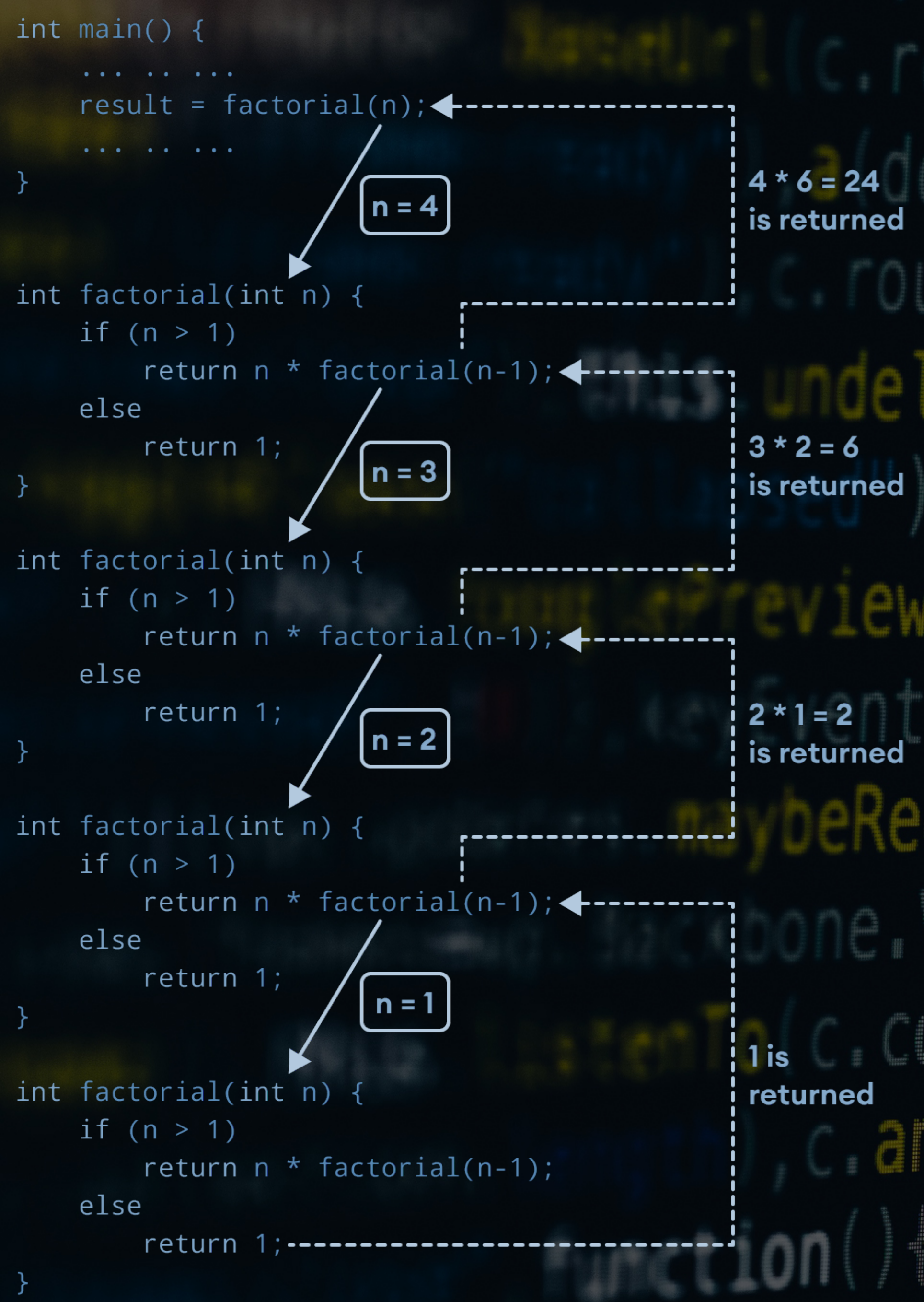
RECURSION IS FUNCTION WHICH CALLS ITSELF

```
void recurse()
{
    ... .. ...
    recurse();
    ... .. ...
}

int main()
{
    ... .. ...
    recurse();
    ... .. ...
}
```



```
void recurse() {
    ... .. ...
    recurse();
    ... .. ...
}

int main() {
    ... .. ...
    recurse();
    ... .. ...
}
```

```cpp
// Factorial of n = 1*2*3*...*n

#include <iostream>
using namespace std;

int factorial(int);

int main() {
    int n, result;

    cout << "Enter a non-negative number: ";
    cin >> n;

    result = factorial(n);
    cout << "Factorial of " << n << " = " << result;
    return 0;
}

int factorial(int n) {
    if (n > 1) {
        return n * factorial(n - 1);
    } else {
        return 1;
    }
}
```



```
int main() {
    ... .. ...
    result = factorial(n);
    ... .. ...
}

int factorial(int n) {
    if (n > 1)
        return n * factorial(n-1);
    else
        return 1;
}

int factorial(int n) {
    if (n > 1)
        return n * factorial(n-1);
    else
        return 1;
}

int factorial(int n) {
    if (n > 1)
        return n * factorial(n-1);
    else
        return 1;
}

int factorial(int n) {
    if (n > 1)
        return n * factorial(n-1);
    else
        return 1;
}
```

DISADVANTAGEOUS

- It takes a lot of stack space compared to an iterative program.
- It uses more processor time.

Thank you for watching