AGH University of Krakow | Faculty of Mechanical Engineering and Robotics
| Department of Robotics and Mechatronics

MECHATRONIC ENGINEERING | I cycle | semester VI | 2025/2026

Object oriented programming and software engineering

Instruction II:

Functions, Function Overloading, Global Variables, Recursion

You will learn:

How to define and use functions in C++ in order to organize code into smaller, reusable parts. You will explore how functions can accept parameters and return values, and how the same function name can be reused through function overloading. You will learn the concept of variable scope and the role of global variables, including the limitations associated with their use. You will also implement simple recursive algorithms to understand how recursion operates in practice.

Course supervisor:
dr inż. Lucjan Miękina, miekina@agh.edu.pl

Instruction author:
mgr inż. Joanna Koszyk, jkoszyk@agh.edu.pl

1. Initial information

Function is a named block of code that performs a specific task. Functions help organize programs into smaller, reusable components, making code easier to read, maintain, and reuse. A function can receive input values through parameters and may return a result to the part of the program that called it.

Function overloading allows multiple functions to have the same name as long as their parameter lists differ in number, type, or order. The compiler determines which version of the function to call based on the arguments provided when the function is used.

Global variable is a variable declared outside of all functions, typically at the beginning of a program. Because of its scope, it can be accessed and modified by any function within the program. While global variables can simplify access to shared data, their use should be limited because they can make programs harder to understand and maintain.

Recursion is a programming technique in which a function calls itself in order to solve a problem. A recursive function typically breaks a problem into smaller instances of the same problem and includes a base case that stops the recursion when a specific condition is met.

2. Theoretical content

Example of a function declaration (prototype) and definition:

```
int add(int a, int b); // function declaration

int add(int a, int b) {
    return a + b;
}
```

Functions can return values using the return statement. If a function does not return a value, its return type should be declared as void.

```
void printMessage() {
    std::cout << "Hello" << std::endl;
}
```

Example of function overloading:

```
int add(int a, int b);
double add(double a, double b);
```

Example of recursion:

```
int factorial(int n) {
    if (n == 0) {
        return 1;
    }
    return n * factorial(n - 1);
}
```

3. Tasks

TASK 1.

Write a function that returns 1 if the number is prime and 0 if it is not.

TASK 2.

Write a program that demonstrates function overloading by creating three versions of a function named maxValue: one that takes two integers, one that takes two doubles, and one that takes three integers. In main(), call maxValue with different types and numbers of arguments and print the results to see which version of the function is executed for each case.

TASK 3.

Write a program that uses a global counter to keep track of how many times different functions are called.
Also add a counter based on local variable.

TASK 4.

Write a program that prints a countdown from a random number to 1 using recursion.

TASK 5.

Write a program that simulates a robot using the Fibonacci sequence. The program should ask the user to enter a number and then use a recursive function to calculate the Fibonacci number at that position. This number will be the robot's energy. Create at least three functions named move. Each time the robot acts, pick one of these move functions randomly. The robot should keep acting until it runs out of energy. At the end, print a summary with the Fibonacci number, the moves the robot made, the energy countdown, and the total number of moves.