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 III:

Arrays, structures and dynamic containers in C++

You will learn:

How to store and process multiple values using arrays and multidimensional arrays. You will practice accessing and modifying array elements and using loops to process collections of data. You will also learn how to analyze memory usage with the sizeof() operator.

The laboratory introduces structures (struct) as a way of grouping related data into a single object. This allows more complex information, such as parameters of robotic components or sensor measurements, to be stored in an organized way.

You will also work with std::vector, a dynamic container from the C++ Standard Library that allows flexible storage. By comparing arrays and vectors, you will better understand when each data structure should be used 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

Array is a collection of elements of the same data type stored in contiguous memory locations. Each element in an array is accessed using an index, starting from zero.

Multidimensional array is an array that contains other arrays as its elements. The most common type is a two-dimensional array, which can be used to represent tables, matrices, or grids.

Structure (struct) is a data type that allows grouping variables of different types into a single unit. Structures are often used to represent complex objects that consist of several related attributes.

std::vector is a dynamic container provided by the C++ Standard Library. Unlike arrays, vectors can automatically resize as elements are added or removed. They provide convenient functions for managing collections of data.

2. Theoretical content

When declaring an array, the number of elements must be specified at compile time.

```
int values[10];
```

Elements are accessed using their index:

```
values[0] = 5;
values[1] = 10;
```

Multidimensional arrays:

```
int grid[3][3];
```

Structures allow programmers to group related variables into a single data type.

```
struct Sensor {
    int id;
    double measurement;
};
```

In modern C++, the std::vector container is often preferred over arrays when the number of elements is not known in advance. Vectors manage memory automatically.

```
std::vector<int> values;
values.push_back(10);
values.push_back(20);
```

The number of elements stored in a vector can be obtained using the size() function.

3. Tasks

TASK 1.

Create a program to store temperature measurements collected during an experiment. The program should begin by declaring an array capable of holding ten temperature values. It should then allow the user to enter these measurements and display all the stored temperatures. Additionally, the program should calculate and display the average temperature, the minimum temperature, and the maximum temperature.

TASK 2.

Create a a 5 × 5 grid map. Each cell in the grid should be assigned a value, where 0 is a free space and 1 is an obstacle. The program should ask the user to enter a value for each cell of the grid, then display the complete grid in matrix form so that it looks like a map. After that, the program should count and display the number of obstacles and free cells.

TASK 3.

Declare several arrays, for example, int numbers[10], double measurements[10], and char symbols[10]. Display the total size of each array in bytes using the sizeof() operator, display the size of a single element, and calculate the number of elements in each array using sizeof(array) / sizeof(array[0]).

Why do arrays with different data types require different amounts of memory?

TASK 4.

Create a structure to represent an environmental sensor in a monitoring system. The program should allow the user to create several sensors, store their information, and display all of them in a readable format.

TASK 5.

Create a program that keeps track of locations explored on a map. Each location should be stored as a pair of coordinates (x, y) using a structure. The program should allow the user to enter as many locations as they want, store them in a std::vector, and then display all visited locations.