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 V:

References and Pointers, Dynamic Memory Allocation

You will learn:

How references and pointers are used to access and modify data in C++ programs. You will understand how references provide an alternative way to refer to existing variables and how pointers store the memory addresses of variables.

You will also learn how dynamic memory allocation allows programs to create objects and arrays during program execution using operators such as new and delete. These mechanisms are commonly used when the number of elements required by a program is not known in advance.

By the end of the laboratory you will understand how memory is accessed through pointers and how dynamically allocated memory must be properly managed.

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

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

1. Initial information

Reference is an alias for another variable. It does not create a new variable but instead refers to an existing one. Any modification made through a reference affects the original variable.

Pointer is a variable that stores the memory address of another variable. Pointers allow programs to directly access memory locations.

2. Theoretical content

Reference example:

```
int a = 10;
int& ref = a;
ref = 20;
```

After this operation, the value of a becomes 20.

Pointer example:

```
int a = 10;
int* ptr = &a;
```

The pointer ptr stores the address of variable a. The value stored at that address can be accessed using the dereference operator *.

```
*ptr = 20;
```

This modifies the value of a.

Pointers provide more flexibility than references because they can be reassigned to point to different memory locations or set to nullptr.

In C++, dynamic memory is typically allocated using the *new* operator and released using the *delete* operator. Example:

```
int* ptr = new int;
*ptr = 10;
delete ptr;
```

For arrays:

```
int* arr = new int[10];
delete[] arr;
```

3. Tasks

TASK 1.

Create three versions of a function that increases a numeric value: one that uses pass by value, one that uses pass by reference, and one that uses a pointer. Use the same variable as input for each function and observe how its value changes after each call.

TASK 2.

Call the following function multiple times:

```
void allocate() {
    int* arr = new int[100];
    std::cout << "Allocated memory" << std::endl;
}
```

Analyze what happens to the allocated memory after each call. Run the program in a loop and observe its behavior. Then modify the code so that the allocated memory is properly released.

TASK 3.

Design a program that stores a dynamic number of measurements without using std::vector. Ask the user for the number of elements, allocate memory dynamically, compute average and release memory