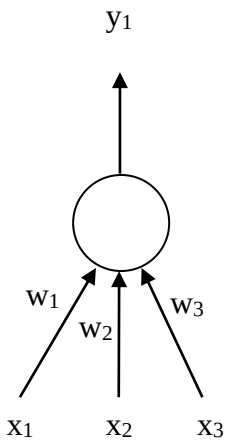
Semester work No. 1
Programming and modelling

Implement McCulloch–Pitts's model of the neuron (without training). The weight vector and the threshold are given by the user and they are parameters of the constructor. Define common ancestor and several descendants (minimally 2) implementing neurons having different output functions. Verify your implementation in a simple program: insert input values x_i and print output value of the neuron. Draw a class diagram in UML.

Output function: $y = \Phi(x_1 w_1 + \cdots x_n w_n + \Theta)$

Φ ... output (step function, sigmoid, ..)
Θ ... threshold



Semester work No. 2
Programming and modelling

Implement an object-oriented model of the fuzzy set. Create at least two types of the fuzzy sets with several types of membership functions (trapezoid, triangular, ...). Design suitable attributes and methods, implement mandatory methods returning membership of the real number to the set in the range <0,1>. Verify your implementation in a simple program: insert input values *x* and print its membership. Draw a class diagram in UML.

(Help: Define an abstract class as a common ancestor and drive descendants for each type of the specific fuzzy set. Use virtual methods.)

Semester work No. 3
Programming and modelling

Implement a complex-calculator using an object-oriented approach. Define a class that represents a complex number and methods implementing basic arithmetic operations (addition, subtraction, multiplication, division) - the methods will return a new object - result (complex number), as well as methods for size determination and calculation of angle (argument). You do not have to implement operator overloading. Draw a class diagram in UML.

Semester work No. 4
Programming and modelling

Program a text-conversational game that asks the player about the property of the animals and tries to guess the animal that the player thinks according to the property. For simplicity, consider only 1 property for each animal. Implement the knowledge base at your own consideration (linked list, array), you can use the STL library. You will receive bonus points for saving / loading the database to / from a text file.

Example: (C is a computer, P is a player)

```
C: Think animal.
C: Does the animal growl?
P: yes
C: That must be a bear!
C: Do you want to play again?
P: yes
C: Think animal.
C: Does the animal growl?
P: no
C: I don't know such animal. What is it?
P: bird
C: What does the bird do?
P: Flies
C: So, the bird flies.
C: Do you want to play again?
P: yes
C: Think animal.
C: Does the animal growl?
P: no
C: Does the animal fly?
P: yes
C: That must be a bird!
C: Do you want to play again?
etc.
```

Don’t solve the problem with the irregular verbs.

Semester work No. 5
Programming and modelling

Implement a class to represent a polyline in a plane with a maximum of 50 points. Design suitable attributes and methods, it is obligatory to implement the methods returning the total line length and number of segments and the length of the *i*-th segment. Verify your implementation in a simple program. Draw a class diagram in UML.

Semester work No. 6
Programming and modelling

Implement a class to represent a triangle in 2D space in the sense of analytic geometry. Design suitable attributes and methods, it is obligatory to implement the methods returning the area and the circumference of the triangle and the method moving each point. Check in the constructor if three point can define a triangle (they don’t lay in line). Verify your implementation in a simple program. Draw a class diagram in UML.

Semester work No. 7
Programming and modelling

Design an object model of an application that simulates the control of the opening of the entrance gate according to the card number read by the reader. You have two prepared classes at your disposal: `CardReader`, whose methods return the number of the card read and the identification of the sensor that read the card, as well as `GateControl`, which is used to control each individual barrier and display messages on the display (i.e., there will be one instance in the application for each gate).
Design a class that implements the card database (you can use classes from the STL) and algorithm of deciding if the card has the right to open the gate. Design methods that insert gate opening rights records into the database.

Draw a class diagram in UML.

Sensor numbers and gate numbers are in the range 1 - 10, card numbers in the range 300,000 - 300,500.

Classes simulate the operation of individual parts of the system.

The `CardReader::is_Read_any_Card()` method returns true if any card has been read by any card reader (it returns true with a probability of 50%). The method `CardReader::get_Card_and_Reader(int &card, int &gate)` returns the number of the card read and the number of the gate the card was attached to (again randomly generated). The correct use of the functions is that the program cyclically asks if the card has been read and then retrieves the card and rack number.

The method `GateControl::open_Gate(int gate)` simulates the opening the barier at gate no *gate*. It writes a text on the screen that a command is being sent to open the corresponding gate. The method `GateControl::display_message(std::string &m)` simulates sending text to the information panel at the gate (prints the text on the screen). It is up to you how you use this function, e.g. to display the text "You can enter", "Your card is not in the database", etc.