ICS0017 Fundamentals of C/C++ Programming

Course code:	ICS0017
Course title:	Fundamentals of C/C++ Programming
Credits:	6 ECTS
Language of instruction:	English
Assessment form	Pass/fail assessment
Prerequisite:	C programming (ICS0004)

Course aims

This course focuses on the principles of object-oriented programming and software design in C++. The program combines technical development with design methodology, emphasizing the integration of programming concepts and structured design practices. In this course, theoretical principles are consistently paired with practical C++ mechanisms, so that abstract concepts are reinforced through hands-on programming.

Learning outcomes

1. Student understands Abstraction and Encapsulation through the use of classes, constructors, and access modifiers.
2. Student applies Generalization and Polymorphism using inheritance and abstract classes.
3. Student understands Information Hiding and modular program structure using interfaces and well-defined class responsibilities.
4. Student applies basic quality considerations through file I/O, exception handling, and error management.
5. Student integrates acquired programming concepts in the implementation of a final course project.

Topics

1. Introduction to Programming and OOP Thinking
2. Variables, Control Flow, Namespaces, and Basic Types
3. Functions, Classes, Objects, and Access Modifiers
4. Constructors, Destructors, and Object Lifetime
5. Inheritance and Class Hierarchies

6. Virtual Functions and Polymorphism
7. Abstract Classes and Interfaces
8. Static Members, Const Correctness, and Operator Overloading
9. Templates and Generic Programming
10. Pointers and Dynamic Memory Allocation
11. File Input/Output and Exception Handling
12. Integration of OOP Concepts
13. Introduction to Multithreading
14. Final Project Integration and Review

Teaching methods and workload

Students must register for the course in ÕIS according to the academic calendar.

Lectures are held every week and introduce object-oriented programming concepts together with software design principles, illustrated by C++ language mechanisms.

Students work in small groups on a course project that integrates programming and design aspects. Continuous feedback is provided through instructor comments, intermediate reviews, and project releases.

Students need a personal computer with a C++ development environment (compiler and IDE) to complete programming assignments and the course project.

• Lectures: 32 hours

• Practice (practices / project work): 32 hours

Homework deadlines and point distribution

Weekly practical assignments are provided during practical sessions to support continuous learning and hands-on practice. **Weekly practice assignments are not graded and are intended solely for skill development and preparation for assessed activities.**

Student performance is evaluated through project releases and control tests, which are conducted once per month. Each review and test covers the topics studied during the corresponding period.

Assessed reviews and control tests

Release 1 / Control Test 1 -Month 1 (Weeks 1–4)

Topics:

1. C++ basics: variables, control flow, namespaces
2. Functions and basic types
3. Classes, objects, access modifiers
4. Constructors and encapsulation
5. Introduction to object-oriented thinking

Release 2 / Control Test 2 - Month 2 (Weeks 5–8)

Topics:

1. Inheritance and class hierarchies
2. Virtual functions and polymorphism
3. Abstract classes and interfaces
4. Static members, const correctness
5. Operator overloading and templates

Release 3 / Control Test 3 - Month 3 (Weeks 9–12)

Topics:

1. Pointers and dynamic memory allocation
2. Object lifetime and basic memory management
3. File input/output (ifstream, ofstream)
4. Exception handling and error management

Release 4 / Control Test 4 - Month 4 (Weeks 13–16)

Topics:

1. Integration of object-oriented concepts
2. Combining polymorphism and templates
3. Basic multithreading concepts (introductory level)
4. Final project integration and system validation

Detailed point distribution, exact deadlines, and assessment criteria for each review and control test are announced at the beginning of the semester in Moodle and are aligned with the course project structure

Assessment and requirements

1. Project assessment

Assessment period	Assessment activity	Topics / focus	Points
Month 1	Project release 1	Requirements analysis, CRC cards, conceptual UML diagrams, basic C++ classes and encapsulation	10

Assessment period	Assessment activity	Topics / focus	Points
Month 2	Project release 2	Inheritance, polymorphism, templates, technical UML class diagram, design trade-offs	15
Month 3	Project release 3	Dynamic memory, interfaces, file I/O, operator overloading, exception handling, UML sequence and state diagrams, tests	15
Month 4	Final project release 4	Integrated C++ system, final UML documentation, architectural integration, system demonstration	20
Total project points			**60**

2. Control tests

Test	Period	Topics covered	Points
Control test 1	Month 1	C++ basics, control flow, functions, classes, constructors, encapsulation	10
Control test 2	Month 2	Inheritance, polymorphism, abstract classes, templates, operator overloading	10
Control test 3	Month 3	Dynamic memory, file I/O, exceptions	10
Control test 4	Month 4	Integration of OOP concepts, basic architecture, testing and validation	10
Total test points			**40**

3. General assessment rules

1. Weekly practical programming exercises **are not graded** and are intended for practice and preparation.
2. Control tests are conducted **once per month**.
3. **Total maximum score: 100 points**.
4. **Minimum requirement to pass the course: at least 80 points out of 100** (project + control tests combined).

5. Project work is carried out in **groups of students**.
6. The **final grade** is based on the **combined result** of project assessment and control tests: **60 project points+ 40 control tests points**.
7. AI-based tools may be used for learning and coding, provided that students understand and can explain all submitted work. All work must comply with **TalTech rules of academic honesty**.
8. Late submission rules and exact deadlines are announced in Moodle at the beginning of the semester.

Late penalties

1. −5 points for each started week of delay for project submissions.
2. The maximum penalty is up to 50% of the points for the respective deliverable.
3. Control tests must be taken at the scheduled time; missed tests may be completed at the end of the course with instructor approval.

Study materials

• Stroustrup, B. *The C++ Programming Language*, 4th Edition, Addison-Wesley
• Lippman, S., Lajoie, J., Moo, B. *C++ Primer*, 5th Edition, Addison-Wesley
• Meyers, S. *Effective C++*, Addison-Wesley
• Meyers, S. *More Effective C++*, Addison-Wesley
• Josuttis, N. *The C++ Standard Library*, Addison-Wesley
• Gamma, E., Helm, R., Johnson, R., Vlissides, J. *Design Patterns*
• Booch, G. *Object-Oriented Analysis and Design*
• Williams, A. *C++ Concurrency in Action*, 2nd Edition, Manning
• Pressman, R. *Software Engineering: A Practitioner's Approach*
• Sommerville, I. *Software Engineering*, 10th Edition
• ISO/IEC 14882 — C++ Standard

• Moodle (course materials and announcements)
• C++ reference documentation: https://en.cppreference.com

Instructor contact

Email: anna.kyselova@taltech.ee

General competences

The course supports the development of object-oriented thinking, analytical reasoning, and problem-solving skills. Students learn to design, implement, test, and document software systems in C++ using structured and object-oriented approaches, as well as to communicate technical solutions clearly in written and oral form.

Students with special needs

Students with special needs are encouraged to contact the instructor at the beginning of the course. If special needs may affect participation or assessment, the instructor should be informed in advance so appropriate accommodation can be arranged. Students may also contact university support services for accessibility arrangements. All requests will be handled confidentially.