Syllabus
Course Program

Object oriented programming

Specialty
113 Applied mathematics

Institute
Educational and Scientific Institute of Computer Science and Information Technology

Educational program
Intelligent Data Analysis

Department
Computer Mathematics and Data Analysis

Level of education
Bachelor's level

Discipline type
Special (professional), Mandatory

Semester
3

Language of instruction
Ukrainian

Lecturers and course developers

Vladyslav Kolbasin

vladyslav.kolbasin@khpi.edu.ua
Senior teacher of the Department of Computer Mathematics and Data Analysis

Work experience - 15 years. Leading lecturer in the disciplines: "Object-oriented programming", "Computer geometry and graphics"

More about the lecturer on the website

General information

Summary

The discipline is aimed at mastering the theoretical and practical foundations of object-oriented programming. The main terms from the OOP course are considered: encapsulation, polymorphism, imitation. C++, C# and Java languages are also compared from the point of view of OO concepts.

Course objectives and goals

Formation of modern object-oriented thinking. Learning the object-oriented C++ programming language based on the Microsoft Visual C++ development environment.

Format of classes

Lectures, laboratory works, self-study, consultations. Final control in the form of an exam.

Competencies

GC 1. The ability to learn and master modern knowledge.
GC 2. Ability to apply knowledge in practical situations.
GC 3. The ability to generate new ideas (creativity).
GC 6. Ability to abstract thinking, analysis and synthesis.
GC 7. Ability to search, process and analyze information from various sources.
GC 8. Knowledge and understanding of the subject area and understanding of professional activity.

GC 10. Skills in the use of information and communication technologies.
SC 5. Ability to develop algorithms and data structures, software tools and software documentation.
SC 6. Ability to design databases, information systems and resources.
SC 7. The ability to solve professional tasks using computer equipment, computer networks and the Internet, in the environment of modern operating systems, using standard office applications.
SC 8. Ability to operate and maintain software of automated and information systems for various purposes.
SC 9. Ability to use modern technologies of programming and software testing.
SC 14. The ability to understand the statement of the task, formulated in the language of a certain subject area, to search and collect the necessary initial data.
SC 15. The ability to formulate a mathematical statement of a problem, based on a statement in the language of the subject field, and to choose a method of its solution, which ensures the required accuracy and reliability of the result.

Learning outcomes

LO 1. Demonstrate knowledge and understanding of basic concepts, principles, theories of applied mathematics and use them in practice.
LO 2. To have basic principles and methods of mathematical, complex and functional analysis, linear algebra and number theory, analytical geometry, theory of differential equations, in particular partial differential equations, theory of probabilities, mathematical statistics and random processes, numerical methods.
LO 13. To use specialized software products and software systems of computer mathematics in practical work.
LO 14. Demonstrate the ability to self-study and continue professional development.
LO 15. To be able to organize one's own activity and obtain a result within a limited time.

Student workload

The total volume of the course is 120 hours (4 ECTS credits): lectures – 28 hours, laboratory classes – 32 hours, self-study – 60 hours.

Course prerequisites

"Algorithmization and programming".

Features of the course, teaching and learning methods, and technologies

Programming skills are required. Study materials are available to students on the teacher's website.

Program of the course

Topics of the lectures

Topic 1. Evolution of programming methodologies. Origin of the object model. Components of the object approach. Abstraction, Encapsulation, Modularity, Hierarchy, Typing, Parallelism. C++ as an improved C.
Topic 2. Classes. Structure and organization of the class. Class declaration and definition. Creation and initialization of objects.
Topic 3. Class constructors. Alternative ways to initialize objects. Destructors. Constant class members. Scopes of class members. Static constants, variables, functions. Constant pointer this. Built-in class member functions.
Topic 4. Overloaded functions and class member operations. Overloading of arithmetic operations. Friendly access. Local and nested classes. Copying class objects.
Topic 5. Inheritance and class hierarchies. Polymorphism. Static and dynamic binding. Virtual functions. Tables of virtual functions. Abstract classes. Multiple inheritance. Virtual base classes. Interfaces.
Topic 6. Error handling methods. Error handling based on the use of the exception mechanism. Exceptions in constructors and destructors. Standard exclusions. Features of exception handling in C# and Java.

Topic 7. Dynamic binding and type casting. RTTI: Usage Issues. Type conversion in C++.
Topic 8. Introduction to generalized programming. Template functions. Template classes. Containers and iterators. Standard Template Library (STL)
Topic 9. History of managed code. Architecture and main components of .NET. The main differences between the C++ and C# languages.
Topic 10. Managed and unmanaged code. Built-in, value-abased, and reference-based data types
Topic 11. Delegates, function pointers
Topic 12. Overview of design and programming patterns. UML design language
Topic 13. Serialization and deserialization of objects.
Topic 14. The reflexion mechanism. Attributes of classes.
Topic 15. Regular expressions.
Topic 16. Instruments of large projects development. Code version control methods. Git.

Topics of the laboratory classes

Topic 1. Pointers and references.
Topic 2. Development of a simpler class.
Topic 3. Overloading of class functions and operations.
Topic 4. Error handling methods in classes.
Topic 5. Inheritance and class hierarchies. Template functions. Template classes. Containers and iterators.
Topic 6. Template classes. "Smart" pointers. Function pointers.
Topic 7. Introduction to .NET. Interfaces.
Topic 8. Delegates. Serialization.
Topic 9. Regular expressions.

Self-study

During self-stury, students study lecture material, prepare for tests, and exams.

Non-formal education

In non-formal education according to the relevant Regulation (http://surl.li/pxssv), the educational component or its individual topics can be taken into account in case of independent completion of professional courses/trainings, obtaining civic education, online education, professional internship, etc.

In particular, individual topics of this component may be taken into account upon successful completion of the following courses:

- Topic 1. Evolution of programming methodologies. Origin of the object model. Components of the object approach. Abstraction, Encapsulation, Modularity, Hierarchy, Typing, Parallelism. C++ as an improved C. https://www.coursera.org/learn/cs-fundamentals-1
- Topic 2. Classes. Structure and organization of the class. Class declaration and definition. Creation and initialization of objects.
https://www.coursera.org/learn/cs-fundamentals-1
- Topic 3. Class constructors. Alternative ways to initialize objects. Destructors. Constant class members. Scopes of class members. Static constants, variables, functions. Constant pointer this. Built-in class member functions. https://www.coursera.org/learn/cs-fundamentals-1
- Topic 4. Overloaded functions and class member operations. Overloading of arithmetic operations. Friendly access. Local and nested classes. Copying class objects. https://www.coursera.org/learn/cs-fundamentals-1 https://www.coursera.org/learn/object-oriented-cpp
- Topic 5. Inheritance and class hierarchies. Polymorphism. Static and dynamic binding. Virtual functions. Tables of virtual functions. Abstract classes. Multiple inheritance. Virtual base classes. Interfaces.
https://www.coursera.org/learn/cs-fundamentals-1 https://www.coursera.org/learn/object-oriented-cpp
- Topic 9. History of managed code. Architecture and main components of .NET. The main differences between the C++ and C# languages. https://www.coursera.org/learn/oo-development-using-c-sharp
- Topic 10. Managed and unmanaged code. Built-in, dimensional, and reference data types
https://www.coursera.org/learn/oo-development-using-c-sharp

Course materials and recommended reading

Main literature

1. В. В. Бублик. Об'єктно-орієнтоване програмування. Київ: ІТ-Книга 2015 р.
https://itknyga.com.ua/documents/OOP_final.pdf
2. B. Stroustrup, The C++ Programming Language 4th Edition – 2013, 1281 p.
https://chenweixiang.github.io/docs/The_C++_Programming_Language_4th_Edition_Bjarne_Stroustrup.pdf
3. R. Floyd, Paradigms of Programming, Communications of the ACM, August 1979, vol. 22, № 8, pp. 455-460.
4. Stanley Lippman, Josée Lajoie and Barbara Moo, "C++ Primer", 5th Edition – 2013, 969 p.
https://zhjwpku.com/assets/pdf/books/C++.Primer.5th.Edition_2013.pdf
5. Erich Gamma et al.. Design Patterns Elements of Reusable Object-Oriented Software – 2009. 417 p.
https://www.javier8a.com/itc/bd1/articulo.pdf
6. https://en.cppreference.com/w/
7. https://www.oodesign.com/

Additional literature

8. Brett D. McLaughlin. Head First Object-Oriented Analysis and Design 1st Edition – 2006, 636 p.
ISBN: 9780596008673

Assessment and grading

Criteria for assessment of student performance, and the final score structure

A necessary condition for passing the test or exam is the completion of laboratory work.
30 points are awarded for writing control tests.
Passing laboratory tests - 70 points.

Grading scale

Total points	National	ECTS
90–100	Excellent	A
82–89	Good	B
75–81	Good	C
64–74	Satisfactory	D
60–63	Satisfactory	E
35–59	Unsatisfactory (requires additional learning)	FX
1–34	Unsatisfactory (requires repetition of the course)	F

Norms of academic integrity and course policy

The student must adhere to the Code of Ethics of Academic Relations and Integrity of NTU "KhPI": to demonstrate discipline, good manners, kindness, honesty, and responsibility. Conflict situations should be openly discussed in academic groups with a lecturer, and if it is impossible to resolve the conflict, they should be brought to the attention of the Institute's management.
Regulatory and legal documents related to the implementation of the principles of academic integrity at NTU "KhPI" are available on the website: http://blogs.kpi.kharkov.ua/v2/nv/akademichna-dobrochesnist/

Approval

Approved by	Date, signature		
	29.08.2024		Head of the department Olena AKHIIEZER
	Date, signature 29.08.2024		Guarantor of the educational program Olena AKHIIEZER