Syllabus
Course Program

Object-oriented programming and design

Specialty 113 - Applied mathematics	Institute Institute of Computer Modelling, Applied Physics and Mathematics
Educational program Applied Mathematics. Computer and Mathematical Modeling	Department Mathematical Modeling and Intelligent Computing in Engineering (161)
Level of education Bachelor's degree	Course type Special (Professional), Mandatory
Semester 2	Language of instruction English

Lecturers and course developers

Lyudmyla Rozova

Lyudmyla.Rozova @khpi.edu.ua
Candidate of Technical Sciences, Associate Professor of the Department of Mathematical Modeling and Intelligent Computing in Engineering at NTU 'KPI'.

More about the lecturer on the department's website

Oleksiy D'yomin

oleksii.domin@khpi.edu.ua
Postgraduate student of the Department of Mathematical Modeling and Intelligent Computing in Engineering at NTU 'KPI'

More about the lecturer on the department's website

General information

Summary

The course is aimed at studying the concepts of object-oriented programming using the C++ language as an example. The course is mandatory and professional for students of the specialty 113-Applied Mathematics, consisting of 150 hours. (5 ECTS credits): lectures – 16 hours, laboratory work – 48 hours, independent work – 86 hours. Final assessment - exam.

Course objectives and goals

To develop in students the knowledge, skills, and abilities necessary for understanding and rational use of the concepts of object-oriented programming. Course objectives: to study the fundamental principles of object-oriented programming and to apply these principles during the design and development of programs; to master the object-oriented programming language C++; to develop skills in decomposing the assigned task and its subsequent programming implementation based on object-oriented technologies.

Format of classes

Lectures, laboratory work.

Competencies

PC04. Ability to develop algorithms and data structures, software tools and software documentation.
PC08. Ability to use modern programming and software testing technologies.

Learning outcomes

PO11. Be able to apply modern technologies of programming and software development, software implementation of numerical and symbolic algorithms.

Student workload

The total volume of the discipline is 150 hours (5 ECTS credits): lectures - 16 hours, laboratory work - 48 hours, independent work - 86 hours.

Course prerequisites

Algorithmisation and programming. Introduction to the speciality. Introductory practice.

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

Classes in the Object-Oriented Programming course are conducted interactively, using multimedia technologies. In laboratory classes, a practical approach to learning is applied, with general and individual tasks. For the practical development of the basics of object-oriented programming and individual tasks of laboratory work, a free cross-platform environment for developing programs in C++ Code::Blocks is used. The student is obliged to attend all classes according to the schedule and perform laboratory work. In order to master the required quality of education in the discipline, attendance and regular preparation for classes are required.

Program of the course

Topics of the lectures

Topic 1: Introduction to object-oriented programming (OOP)
Comparison of the structural and object-oriented approach to programming. Basic concepts of OOP. Introduction to the concept of class. Fields and methods of a class. An instance of a class is an object. Basic principles of OOP. Application of the concepts of abstraction and encapsulation to create a class. Access modes in the class. The concept of a UML class diagram. Separation of the class interface from the implementation, multi-file projects.

Topic 2. Class constructors and destructors. Friendly functions and classes.
The concept of class constructor and destructor. Types of class constructors. The default constructor. Constructor with parameters, including default parameters. Delegating constructor. The copy constructor. The ‘this’ pointer. Function overloading. Friendly functions. Friendly classes.

Topic 3. Overloading of operators and operations
The concept of overloading operators and operations. Rules of overloading. The concept of an operator function. Ways to define an operator function.
Overloading operators using class methods. Overloading operators using friendly functions. Overloading operators using external functions

Topic 4. Static elements. Inheritance

Static variables and methods.
The concept of inheritance. Types of inheritance. Inheritance syntax. Controlling access to members of the base class. Simple inheritance. Calling class constructors and passing parameters to base class constructors. Multiple inheritance. Problems of multiple inheritance. Calling destructors during inheritance. Types of relationships between classes.
Topic 5. Virtual functions, abstract classes

The concept of polymorphism. Virtual functions. A polymorphic class. Rules for declaring and using virtual functions. Early static and late dynamic binding. Table of virtual methods. An abstract class. A purely virtual function. Virtual destructor. Virtual inheritance.

Topic 6: Exceptional situations
Types of errors that can occur in programs. The concept of an exceptional situation. Handling an exception. Algorithm for handling uncaught exceptions. Specification of exceptions. List of function exceptions. Exceptions in constructors and destructors. Classes of exceptions and their hierarchy. Standard exceptions. The exception class.

Topic 7. Templates of functions and classes
The concept of function templates and their format. Rules for working with function templates. Class templates in C++. Full and partial specialisation of class templates. Static elements of templates. Inheritance of class templates

Topic 8: Namespaces. Preprocessor directives.
Global namespace. Namespace conflict. Declaring namespaces. Nesting of namespaces. Preprocessor directives. Macro definitions. Conditional compilation.

Topics of the workshops

Topics of the laboratory classes

Topic 1: Repetition of the material of the first semester on programming. Structures.
Lab 1: Data structures. Working with binary files

Topic 2. Introduction to classes and objects. Application of abstraction principles and encapsulation on practical examples of creating classes and their objects. Creating multi-file projects.
Laboratory work 2. Classes and objects. Abstraction and encapsulation. UML basics for representing classes. Classes in C++

Topic 3. Creating different types of class constructors and destructors. The work of the 'this' pointer. Creating friendly functions and classes
Laboratory work 3. Class constructors. The class destructor.

Topic 4. Overloading operations and operators in three ways. Types of operations

Laboratory work 4. Overloading of operators.
Laboratory work 5. Overloading of operators

Topic 5. Examples of creating a class hierarchy by inheritance. Constructors and destructors during inheritance, features of passing parameters to base class constructors. Examples and application of static elements
Laboratory work 6. Global variables in OOP: static members of the class. The 'Loner' design pattern.
Laboratory work 7. Inheritance

Topic 6. Creating virtual methods using examples. Development of abstract classes. Implementation of virtual methods. Examples.
Laboratory work 8. Polymorphism in C ++. Virtual functions.

Topic 7. Consideration of the algorithm for handling exceptions on examples. Creating classes and hierarchy of exception classes.
Lab 9: Exceptional situations.

Topic 8: Examples of creating templates of functions and classes.
Laboratory work 10. Templates of classes.

Self-study

Independent work involves studying the lecture material, preparing for laboratory classes, and completing individual calculation tasks for each laboratory work.
Independent study of topics and issues that are not covered in lectures:
Additional tasks for independent work on the topics of lectures and laboratory work:
http://repository.kpi.kharkov.ua/handle/KhPI-Press/52280.

Course materials and recommended reading

The C++ programming language / Bjarne Stroustrup.-4th edition.- Addison-Wesley Professional, 2013. - 1368p.
Object-Oriented Programming with C++/ E Balagurusamy - 8th edition.- McGraw Hill, 2020. - 656p.
Object-oriented programming: a textbook. In 2 parts. 2. An object-oriented approach to software development /S. M. Alkhimova. - Kyiv: Igor Sikorsky Kyiv Polytechnic Institute, Polytechnic Publishing House, 2019.
Fundamentals of C++ programming: A textbook for students majoring in 113 - Applied Mathematics and 122 - Computer Science: a textbook / Vodka O.O., Dashkevych A.O., Ivanchenko K.V., Rozova L.V., Senko A.V. - Kharkiv: NTU 'KhPI', 2021. 114 p. http://repository.kpi.kharkov.ua/handle/KhPI-Press/52280.
Fundamentals of object-oriented programming: a textbook / Hryshanovych T.O., Hlynchuk L.Y.; Lesya Ukrainka National University of Lutsk: Lesya Ukrainka National University, 2022. - 120 p.
C++: The Complete Reference/ Shildt, G.- 4th Edition.-- McGraw Hill Education, 2002, - 1035p.
Additional literature:
Robert Lafore. Object Oriented Programming In C++, 4th edition. - Sams - 1040p.
Design patterns: elements of reusable object-oriented software / Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. - Addison Wesley, 1995. - 395p.

Assessment and grading

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

Points are awarded based on the rating according to the following ratio:
Practical part:
Performing and defending individual tasks for laboratory work - 60 points
(mandatory component, is an admission to the test or exam)
Theoretical part:
1 mark: Test on the theoretical part based on lecture materials - 40 points
2 var: Exam consisting of theoretical questions - 40 points. Admission to the exam is the completion of individual tasks for laboratory work..

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	Head of the department Alexey VODKA
	Date, signature	Guarantor of the educational program Gennadiy LVOV