Federal State Autonomous Educational Institution of Higher Education
Peoples' Friendship University of Russia named after Patrice Lumumba

Academy of Engineering

(name of the main educational unit (MEU) that developed the educational program of higher education)

WORKING PROGRAM OF THE DISCIPLINE

COMPUTER SCIENCE AND PROGRAMMING

(name of discipline/module)

Recommended for the field of study/specialty:

27.03.04 CONTROL IN TECHNICAL SYSTEMS

(code and name of the field of study/specialty)

The discipline is mastered within the framework of the implementation of the main professional educational program of higher education (EP HE):

DATA SCIENCE AND SPACE SYSTEMS

(name (profile/specialization) of the educational institution of higher education)

2026

1. THE GOAL OF MASTERING THE DISCIPLINE

The "Computer Science and Programming" course is part of the "Data Science and Space Systems" bachelor's program, focusing on 27.03.04 "Control in Technical Systems." It is studied in semesters 2, 3, 4, and 5 of years 1, 2, and 3. The course is taught by the Department of Mechanics and Control Processes. It consists of 29 sections and 101 topics and focuses on the theoretical and practical foundations of information technology and programming. Particular attention is paid to the analysis of methods for solving typical problems and their application in professional activities.

The goal of mastering this course is to develop fundamental knowledge and skills in applying programming technologies to solve a wide range of problems necessary for professional activity and mastering subsequent disciplines.

2. REQUIREMENTS FOR THE RESULTS OF MASTERING THE DISCIPLINE

Mastering the discipline "Computer Science and Programming" aimed at developing the following competencies (parts of competencies) in students:

Table 2.1. List of competencies developed in students while mastering the discipline (results of mastering the discipline)

Cipher	Competence	Indicators of Competency Achievement (within this discipline)
GPC-6	Capable of developing and using algorithms and programs, modern information technologies, methods and means of control, diagnostics and management, suitable for practical application in the field of his professional activity	GPC-6.1 Knows the basic algorithms and programs, modern information technologies, methods and means of control, diagnostics and management, suitable for practical application in the field of his professional activity; GPC-6.2 Able to apply algorithms and programs, modern information technologies, methods and means of control, diagnostics and management, suitable for practical application in the field of his professional activity; GPC-6.3 Confidently uses algorithms and programs, modern information technologies, methods and means of control, diagnostics and management, suitable for practical application in the field of his professional activity;
GPC-9	Able to perform experiments using given methods and process the results using modern information technologies and technical means	GPC-9.1 Knows modern information technologies and technical means; GPC-9.2 Able to apply modern information technologies and technical means to process experimental results; GPC-9.3 Proficient in modern information technologies and technical means for performing experiments and processing results;
PC-1	Capable of collecting, processing and interpreting modern scientific research data necessary to draw conclusions on relevant scientific research, including Earth remote sensing data	PC-1.1 Knows modern methods of collecting, processing and interpreting data from modern scientific research necessary for drawing conclusions on relevant scientific research; PC-1.2 Able to apply modern methods and tools for processing and interpreting scientific research data; PC-1.3 Possesses the basic skills of collecting, processing and interpreting data from modern scientific research, necessary for drawing conclusions on relevant scientific research;

3. PLACE OF THE DISCIPLINE IN THE STRUCTURE OF THE EDUCATIONAL EDUCATIONAL INSTITUTION

Discipline "Computer Science and Programming" refers to the mandatory part of block 1 “Disciplines (modules)” of the educational program of higher education.

As part of the higher education program, students also master other disciplines and/or practices that contribute to the achievement of the planned results of mastering the discipline "Computer Science and Programming".

Table 3.1. List of components of the educational program of higher education that contribute to the achievement of the planned results of mastering the discipline

Cipher	Name of competence	Previous courses/modules, practical training*	Subsequent disciplines/modules, practices*
GPC-6	Capable of developing and using algorithms and programs, modern information technologies, methods and means of control, diagnostics and management, suitable for practical application in the field of his professional activity	Introduction to Computing Science;	Automatic Control Theory; Space Flight Mechanics; Research work / Scientific research work; Undergraduate Training;
GPC-9	Able to perform experiments using given methods and process the results using modern information technologies and technical means	Basic Military Training. Life Safety; Introduction to Computing Science;	Undergraduate Training; Technological Training; Analysis of Geoinformation Data; Optimal Control Methods;
PC-1	Capable of collecting, processing and interpreting modern scientific research data necessary to draw conclusions on relevant scientific research, including Earth remote sensing data	Introduction to Computing Science;	*Virtual and Augmented Reality Technology***; *Virtual and augmented reality technologies***; Optimal Control Methods; Analysis of Geoinformation Data; Space Flight Mechanics; Automatic Control Theory; Research work / Scientific research work; Technological Training; Undergraduate Training;

* - filled in accordance with the competency matrix and the SUP EP HE

** - elective courses/practices

4. SCOPE OF THE DISCIPLINE AND TYPES OF EDUCATIONAL WORK

The total workload of the discipline "Computer Science and Programming" is 17 credit units.

Table 4.1. Types of educational work by periods of mastering the educational program of higher education for full-time education.

Type of academic work	**TOTAL,**academic hours		**Semester(s)**			
			2	**3**	**4**	**5**
Contact work, academic hours	*263*		*68*	*72*	*51*	*72*
Lectures (LC)	123		34	36	17	36
Laboratory work (LW)	140		34	36	34	36
Practical/seminar classes (SC)	0		0	0	0	0
Independent work of students, academic hours	*241*		*94*	*81*	*21*	*45*
Control (exam/test with assessment), academic hours	*108*		*18*	*27*	*36*	*27*
Total complexity of the discipline	**academic hours**	***612***	*180*	*180*	*108*	*144*
	credit	***17***	*5*	*5*	*3*	*4*

The total workload of the discipline "Computer Science and Programming" is 17 credit units.

Table 4.2. Types of educational work by periods of mastering the educational program of higher education for full-time education.

Type of academic work	**TOTAL,**academic hours		**Semester(s)**				
			1	**2**	**3**	**4**	**5**
Contact work, academic hours	*299*		*36*	*68*	*72*	*51*	*72*
Lectures (LC)	141		18	34	36	17	36
Laboratory work (LW)	158		18	34	36	34	36
Practical/seminar classes (SC)	0		0	0	0	0	0
Independent work of students, academic hours	*313*		*72*	*94*	*81*	*21*	*45*
Control (exam/test with assessment), academic hours	*108*		*0*	*18*	*27*	*36*	*27*
Total complexity of the discipline	**academic hours**	***720***	*108*	*180*	*180*	*108*	*144*
	credit	***20***	*3*	*5*	*5*	*3*	*4*

The total workload of the discipline "Computer Science and Programming" is 17 credit units.

Table 4.3. Types of educational work by periods of mastering the educational program of higher education for full-time education.

Type of academic work	**TOTAL,**academic hours		**Semester(s)**			
			2	**3**	**4**	**5**
Contact work, academic hours	*263*		*68*	*72*	*51*	*72*
Lectures (LC)	123		34	36	17	36
Laboratory work (LW)	140		34	36	34	36
Practical/seminar classes (SC)	0		0	0	0	0
Independent work of students, academic hours	*241*		*94*	*81*	*21*	*45*
Control (exam/test with assessment), academic hours	*108*		*18*	*27*	*36*	*27*
Total complexity of the discipline	**academic hours**	***612***	*180*	*180*	*108*	*144*
	credit	***17***	*5*	*5*	*3*	*4*

5. CONTENT OF THE DISCIPLINE

Table 5.1. Content of the discipline (module) by types of academic work

Section number	Name of the discipline section	Topic Title		Topic Contents	Type of academic work*
Section 1	Information and computer science	1.1	Basic concepts. Subject and objectives of computer science	Computer science is defined as the science of the structure and properties of information, as well as methods for collecting, processing, storing, transmitting, and using it. The main objectives of computer science are the automation of information processes and the creation of information systems and technologies.	OK
		1.2	Information and its properties	The concept of information. Properties of information: objectivity, reliability, completeness, relevance, usefulness, comprehensibility. Forms of information presentation. Measures of information: syntactic, semantic, pragmatic.	OK
		1.3	Arithmetic and logical principles of computer operation	Number systems: positional and non-positional. Binary, octal, and hexadecimal systems. Conversion of numbers between number systems. Logical foundations: Boolean algebra, basic logical operations (conjunction, disjunction, negation). Logic elements and circuits.	OK
		1.4	Information coding	Principles of computer information encoding. Number encoding: integer and real types. Text encoding: encoding tables. Encoding of graphic, audio, and video information. The concept of data compression.	OK
		1.5	Prospects for the development of computer science	Current trends in computer science and computing: artificial intelligence, machine learning, big data, cloud computing, and the Internet of Things. Directions for further development.	OK
		1.6	Modern aspects of programming. Classification and areas of application of modern programming languages	The evolution of programming languages. Classification: machine-oriented, procedural, object-oriented, functional, and logical. Application areas: systems programming, web development, scientific computing, mobile development, and data analysis.	OK
Section 2	Computer technology	2.1	History of development and classification of computers	The stages of computing technology development: from mechanical devices to modern computers. Computer generations. Classification: supercomputers, servers, personal computers, embedded systems, mobile devices.	OK
		2.2	Computer architecture. Composition of a computing system	Von Neumann's principles. Basic components: processor, memory, input/output devices. Bus-modular architecture. Concepts of modern architectures.	OK
		2.3	Operating principles of computing system	Central processing unit: control unit, arithmetic logic unit, reg-	LC, LW

Section number	Name of the discipline section	Topic Title		Topic Contents	Type of academic work*
			elements	isters. Memory: RAM, ROM, cache. Operating principles of input and output devices.	
		2.4	Computer networks. Client-server architecture	Network classification: local, global, corporate. Network protocols. The OSI model and TCP/IP stack. Client-server architecture: operating principles, server types. Peer-to-peer networks.	LC, LW
Section 3	Software	3.1	System software	Operating systems: purpose, functions, classification. Device drivers. Utilities. Software tools: compilers, interpreters, debuggers, development environments.	LC, LW
		3.2	Application software	Classification of application software: word processors, spreadsheets, database management systems, graphic editors, multimedia applications. Calculation automation systems.	LC, LW
Section 4	Basic concepts of modeling and algorithmization	4.1	Stages of solving a problem using a computer	Key stages: problem definition, formalization, algorithm development, programming, debugging and testing, and results analysis. Correctness is essential at every stage.	OK
		4.2	Models and their classification	The concept of a model. Classification of models: material and informational; static and dynamic; deterministic and stochastic; simulation. Objectives of modeling.	LC, LW
		4.3	The concept and properties of an algorithm. Methods for describing an algorithm	Definition of an algorithm. Properties: discreteness, certainty, efficiency, mass character. Description methods: verbal, graphical (flowcharts), pseudocode, algorithmic languages.	OK
Section 5	Python programming language	5.1	Interpreter. Basic syntax. Memory model. Data types.	Installing and running the Python interpreter. Basic syntax: indentation, comments, operators. Memory model: reference model, objects, and variables. Basic data types: numbers, strings, lists, tuples, dictionaries, sets.	LC, LW
		5.2	Logical constructs. Cycles and branches	Conditional operator: logical expressions, nested conditions. Loops: loop with counter, loop with condition. Loop control statements. List comprehensions.	LC, LW
		5.3	Functions. Passing Arguments. Scope. Call Stack	Defining and calling functions. Parameters and arguments: positional, named, default values. Passing arguments by reference. Variable scopes: local, global, non-local. Call stack and recursion.	LC, LW
		5.4	Working with Files. File Properties and Types. Data Serialization	Opening, reading, and writing files. Text and binary files. Context manager. Data serialization: converting objects to storage or transmission formats. Serialization modules.	LC, LW
		5.5	Block-based organization of programs. Modules and packages. The pip package manager.	Dividing a program into modules. Importing modules. Packages as hierarchical structures of modules. Installing third-party libraries using the pip package manager. Dependency management.	LC, LW

Section number	Name of the discipline section	Topic Title		Topic Contents	Type of academic work*
Section 6	Python libraries for solving scientific and applied problems	6.1	Data visualization using the Matplotlib library	Chart creation: line, scatter, and bar charts. Customize axes, labels, and legends. Create multiple charts in a single field. Save images.	LC, LW
		6.2	Solving statistics and linear algebra problems using NumPy and Pandas libraries	NumPy library: multidimensional arrays, array operations, mathematical functions. Pandas library: data structures (Series, DataFrame), data loading and processing, grouping, aggregation, working with missing values.	LC, LW
Section 7	Programming paradigms	7.1	Main paradigms and their features: procedural programming, object-oriented programming, functional programming	Procedural programming: programs as a sequence of instructions, procedures, and functions. Object-oriented programming: objects, classes, and basic principles. Functional programming: functions as first-class objects, no side effects.	LC, LW
		7.2	Object-oriented programming in Python. Encapsulation, polymorphism, inheritance. Classes and objects. Class inheritance.	Defining classes and creating objects. Attributes and methods. Encapsulation: private and protected attributes. Inheritance: base and derived classes, method overriding. Polymorphism: a single interface for different types.	LC, LW
		7.3	Functional Programming in Python. Anonymous Functions: Syntax and Usage. Function Decorators	Higher-order functions. Anonymous functions: syntax and usage with sorting and filtering functions. Function decorators: wrapping functions, adding functionality without changing the source code.	OK
		7.4	Visual block programming as a tool for creating and managing VR worlds	The concept of visual block programming. Drag-and-drop principles. Application in educational environments and for creating virtual worlds. Examples of environments.	LC, LW
Section 8	Data structures	8.1	Basic data structures and their properties	Basic structures: array, list, stack, queue, linked list, tree, graph. Properties: ordering, uniqueness of elements, access and insertion methods.	LC, LW
		8.2	Standard Python data structures and how to work with them	Lists: mutability, methods. Tuples: immutability. Dictionaries: key-value pairs, hashing. Sets: element uniqueness, set operations.	LC, LW
		8.3	Graph data structures. Python libraries implementing graph data structures and their specific features.	Graph concepts: vertices and edges, directed and undirected graphs. Graph representation: adjacency lists, adjacency matrices. Graph libraries, basic graph algorithms.	LC, LW
Section 9	Algorithms	9.1	The concept of computation and computability. Classification of algorithms. Turing machines.	The concept of a computable process. The problem of computability. The Turing machine: formal model of the algorithm, tape, control device. Classification of algorithms by various criteria.	OK
		9.2	Evaluation of algorithm complexity	Time and space complexity. Asymptotic analysis. Notations: upper, lower, and tight bound notations. Comparison of algorithm performance.	OK
		9.3	Sorting algorithms	Basic sorting algorithms: bubble sort, insertion sort, selection sort, quicksort, merge sort. Comparison of their efficiency.	LC, LW

Section number	Name of the discipline section	Topic Title		Topic Contents	Type of academic work*
		9.4	Search algorithms	Linear search. Binary search in an ordered array. Searching for a substring in a string. Hashing as a fast search method.	LC, LW
		9.5	Graph algorithms	Graph traversals: depth-first search and breadth-first search. Finding shortest paths: algorithms for graphs without negative edges. The concept of minimum spanning trees.	LC, LW
Section 10	Python libraries for solving scientific and applied problems	10.1	SciPy library functionality and features of working with them	SciPy as a NumPy extension. Submodules for optimization, integration, interpolation, linear algebra, signal and image processing. Solving differential equations.	LC, LW
		10.2	SymPy library functionality and features of working with it	SymPy as a library for symbolic computing. Define symbols, simplify expressions, differentiate, integrate, and solve equations symbolically.	LC, LW
Section 11	Operating System Basics	11.1	History of development and main functions of operating systems	Stages of operating system development. Key functions: process management, memory management, input/output management, file system management, and user interface management.	OK
		11.2	Basics of working in the command interpreter	Command line: prompt, commands. Navigating the file system, creating and deleting files and directories, running programs. I/O redirection, pipes.	LC, LW
		11.3	Architectural features of operating systems	Monolithic kernel, microkernel, hybrid kernel. Modularity. Main kernel components. Processor operating modes: user and privileged.	LC, LW
		11.4	Process and memory management	Processes and threads: states, context, scheduling. Scheduling algorithms. Memory management: virtual memory, paging and segment organization, paging.	LC, LW
		11.5	Input/output management	Device drivers. Buffering, caching. Interrupt handling. File systems: structure, types, basic operations.	LC, LW
Section 12	Version control systems (VCS)	12.1	History of the development of SLE. Basic concepts and terms. Classification and modern SLE.	The evolution of version control systems: local, centralized, distributed. Key concepts: repository, commit, branch, tag. Modern systems.	LC, LW
		12.2	Using Git and Organizing Your Software Development Workflow	Basic Git commands: initializing a repository, adding changes, committing, viewing history. Working with branches: creating, merging, resolving conflicts. Remote repositories. Collaboration.	LC, LW
Section 13	Basics of the C Programming Language	13.1	History of development, features and scope of application of the C language	Creation of the C language. Features: efficiency, low-level capabilities, portability. Applications: systems programming, embedded systems, operating systems.	OK
		13.2	Declaration and definition of variables. Variable types. Type conversion.	Basic data types: integer, real, character. Variable declarations and definitions. Initialization. Explicit and implicit type conversion.	LC, LW

Section number	Name of the discipline section	Topic Title		Topic Contents	Type of academic work*
		13.3	Arithmetic and logical operators. Bitwise operators. Precedence and order of evaluation.	Arithmetic operations. Logical operators. Bitwise operators: working with individual bits. Operator precedence. Expression evaluation order.	LC, LW
		13.4	Control structures. Branching and loops, unconditional jump and multiple choice operators	Conditional operator. Ternary operator. Multiple-choice operator. Loops with precondition, postcondition, and counter. Unconditional jump operators.	LC, LW
Section 14	Functions and structure of the program	14.1	Functions. Syntactic constructs for working with functions: declaration, definition, call. Recursion. Call stack. Block structure of a program.	Function declaration (prototype) and definition. Argument passing: by value. Return value. Recursive functions. Call stack. Block structure: local blocks.	OK
		14.2	External variables and scope. Static and register variables. Header files.	Global and local variables. Scope and lifetime. Static variables: preserving a value between calls. Register variables. Header files: declaring functions and macros.	LC, LW
		14.3	The process of program compilation. Preprocessor, file inclusion, macro substitution, conditional compilation	Stages: preprocessing, compilation, assembly, linking. Preprocessor directives: file inclusion, macro definitions, conditional compilation.	LW
Section 15	Pointers and arrays	15.1	Pointers and Addresses. Pointers and Function Arguments	The concept of a pointer as a variable storing an address. The address-taking and dereferencing operator. Passing arguments by pointer to modify values within a function.	LC, LW
		15.2	Arrays. Address arithmetic	Defining and initializing arrays. Relationships between arrays and pointers. Address arithmetic: addition and subtraction of pointers. Accessing array elements through a pointer.	OK
		15.3	Pointers to pointers. Multidimensional arrays	Pointer to pointer. Representation of multidimensional arrays in memory. Accessing elements of multidimensional arrays via pointers.	LC, LW
		15.4	Command-line arguments. Function pointers. Complex declarations.	Passing arguments to the main function. Function pointers: declaration, assignment, and call. Rules for reading complex declarations involving pointers and arrays.	LC, LW
Section 16	Structures	16.1	Basics of Working with Structures. Structures and Functions. Pointers to Structures	Defining a structure type. Declaring structure variables. Accessing fields. Passing structures to functions by value and by pointer.	LC, LW
		16.2	Definition of new types	Using a keyword to create type aliases. Improving code readability.	OK
		16.3	Unions and bit fields	Unions: storing different types in a single memory area. Bit fields: saving memory when storing flags.	LC, LW
Section 17	Input/output operations	17.1	Standard input/output facilities	Formatted and unformatted input/output functions. Standard streams: input, output, error output.	LC, LW
		17.2	Variable-length argument lists. Formatted	Creating functions with a variable number of arguments. Macros	LC, LW

Section number	Name of the discipline section	Topic Title		Topic Contents	Type of academic work*
			input	for working with such lists. Formatted input.	
		17.3	Reading and writing files	Opening and closing files. Access modes. Reading and writing characters, strings, and data blocks. Positioning within a file.	LC, LW
		17.4	Error handling	Error checking when performing input/output operations. Error codes.	LC, LW
Section 18	Standard Library	18.1	String Operations: Character Analysis, Classification, and Conversion	String functions: copying, concatenation, comparison, searching. Character classification functions: checking for letters, numbers, etc. Case conversion.	LC, LW
		18.2	Command Execution. Memory Management	Executing system commands from a program. Dynamic memory allocation and deallocation functions. Dynamic memory management.	OK
		18.3	Mathematical functions. Random number generator	Mathematical functions: trigonometric, exponential, logarithmic. Pseudo-random number generation.	LC, LW
Section 19	Basics of the C++ programming language	19.1	The history, features, and scope of the C language. Differences between C and C++	The creation of the C++ language. Support for object-oriented programming. Differences from C: input/output, references, namespaces, function overloading.	OK
		19.2	Types and Declarations. Namespaces. Pointers, References, Arrays, and Structures.	Basic data types. Variable declarations. Namespaces to avoid name conflicts. References as an alternative to pointers. Arrays and structures in C++.	LC, LW
		19.3	Expressions and Operators. Functions	C++ language operators. Control structures. Functions: function overloading, default values, passing by reference.	LC, LW
		19.4	Exceptions. Keyword throw, catch	Exception handling mechanism: blocks for error tracking, exception generation, exception interception.	LC, LW
		19.5	Source files and programs. Separate compilation	Splitting a program into multiple source files. Header files. The process of separate compilation and linking.	LC, LW
Section 20	Abstraction mechanisms (OOP)	20.1	Classes and objects. Class members. Constructors and destructors. Class composition. Access modifiers. Overloading class methods.	Class definition. Object creation. Fields and methods. Constructors: default, parameterized, copy. Destructor. Object nesting (composition). Access modifiers. Method overloading.	LC, LW
		20.2	Operator overloading. Operator functions. Type casting operators. Class friends	Overloading operators as member functions or external functions. Overloading type cast operators. Friend functions and classes for accessing private members.	LC, LW
		20.3	Class inheritance. Derived classes. Virtual functions. Class hierarchies and abstract classes.	Inheritance: base and derived classes. Constructor and destructor call order. Virtual functions for late binding. Polymorphism. Abstract classes.	LC, LW
		20.4	Templates. Definition of a template. Specification of templates. Type checking.	The concept of templates for creating generic code. Function templates. Class templates. Concretization of templates with specific	LC, LW

Section number	Name of the discipline section	Topic Title		Topic Contents	Type of academic work*
			Function templates. Specialization	types. Partial and full specialization.	
Section 21	Exception handling	21.1	Error Handling: Exception Grouping	Traditional error handling methods and their shortcomings. Exceptions as an alternative. Exception class hierarchy.	LC, LW
		21.2	Exception Handling. Resource Management	Catch blocks. Catch order. Exception rethrow. The resource acquisition idiom during initialization for safe resource management.	LC, LW
		21.3	Exception specification	Specifying what exceptions a function can throw. Specifications.	LC, LW
		21.4	Exceptions and Efficiency: Alternatives to Error Handling	The impact of exceptions on performance. Comparison of exceptions with other error handling mechanisms.	LC, LW
Section 22	Class hierarchies	22.1	Designing a Class Hierarchy. Traditional Class Hierarchies	Principles of designing hierarchies. Abstract base classes. Examples of classical hierarchies.	OK
		22.2	Multiple inheritance and access control	Inheritance from multiple base classes. The diamond inheritance problem. Virtual inheritance. Access control in inheritance.	OK
Section 23	Standard Library (STL)	23.1	Standard containers	Sequential containers: vector, list, dec. Associative containers: set, map, multiset, multimap. Unordered containers.	LC, LW
		23.2	Algorithms and classes of functional objects	Generalized algorithms: sorting, searching, copying, transformation. Functional objects (functors). Lambda expressions.	LC, LW
		23.3	Iterators and allocators	Iterators as a generalization of pointers. Iterator categories. Allocators for memory management in containers.	LC, LW
		23.4	Strings and Streams	A class for working with strings. Stream I/O: string streams.	LC, LW
		23.5	Classes for mathematical calculations	Complex numbers. Pairs. Functions for working with sequences.	LC, LW
Section 24	Programming technology	24.1	Basic concepts and approaches	Software as a product. Programming technology: goals and principles. Software engineering.	OK
		24.2	Problems of developing complex software systems	Complexity, consistency, variability, invisibility. Challenges in managing deadlines, budgets, and quality.	OK
		24.3	Block-hierarchical approach to creating complex systems	Decomposition principle. Division into subsystems, modules, components. Hierarchical structure.	OK
		24.4	Life cycle and stages of development	Key stages: requirements analysis, design, implementation, testing, deployment, and maintenance. Life cycle models.	OK
		24.5	Evaluation of the quality of software development processes	Quality standards and metrics. Performance, reliability, maintainability.	OK
Section 25	Techniques for ensuring the technological effectiveness of software products	25.1	Software Manufacturability. Modules and Their Properties	The concept of manufacturability. Modules: coherence and cohesion. Advantages of modules.	OK
		25.2	Top-down and bottom-up development	Development strategies: from general to specific (top-down) and from specific to general (bottom-up). A combination of both.	OK
		25.3	Structured and "Unstructured" Programming: Tools for Describing Structured Algorithms	Principles of structured programming: prohibition of unconditional jumps. Description tools: flowcharts, Nassi-Schneiderman	LC, LW

Section number	Name of the discipline section	Topic Title		Topic Contents	Type of academic work*
				diagrams.	
		25.4	Program design style. Efficiency and technology	Code formatting, variable naming, and commenting. The impact of style on maintainability. The tradeoff between efficiency and maintainability.	LC, LW
Section 26	Defining software requirements	26.1	Classification of software products by functionality. Basic operational requirements	Classification: system, application, and tool software. Requirements: functional and non-functional (reliability, performance, security).	OK
		26.2	Development of technical specifications. Fundamental decisions for the initial design stages	Composition and structure of the technical specifications. Selection of architecture, platform, and technologies.	OK
Section 27	Structural approach	27.1	Requirements analysis and specification definition using a structured approach. State transition diagrams, functional diagrams, and data flow diagrams. Data structures and component data relationship diagrams. Mathematical models of problems.	Methods of structural analysis. State transition diagrams. Functional diagrams. Data flow diagrams. Data structure modeling.	LC, LW
		27.2	Software design using a structured approach. Structural and functional diagrams. Step-by-step detailing. Constantine maps. Data structure design. Data decomposition-based design. Case technologies.	Construction of structural and functional diagrams. Step-by-step detailing method. Constantine maps for structure visualization. Data decomposition as a design basis.	OK
Section 28	Object-oriented approach	28.1	Requirements analysis and specification using the object-oriented approach. UML. Use case definition. Conceptual domain modeling. Behavior description.	Unified Modeling Language. Use case diagrams. Conceptual models. Activity and sequence diagrams for describing behavior.	LC, LW
		28.2	Object-oriented software design. Developing a structure. Defining relationships between objects and classes. Designing classes. Layout. Layout of distributed software systems. Spiral development model.	Class diagrams. Relationships: association, aggregation, composition, inheritance. Design of classes and components. Deployment diagrams. Spiral life cycle model.	OK
Section 29	Software testing	29.1	Types of quality control. Manual control. Structural and functional testing.	Quality assurance methods: code reviews, inspections. Manual testing. Structural (white-box) testing. Functional (black-box) testing.	LC, LW
		29.2	Modular, comprehensive and evaluation testing	Testing of individual modules. Integration testing of module interactions. System testing. Acceptance testing. Evaluation testing of characteristics.	LC, LW

* - to be completed only for FULL-TIME education: LC – lectures; LW – laboratory work; SC – practical/seminar classes.

6. LOGISTIC AND TECHNICAL SUPPORT OF DISCIPLINE

Table 6.1. Material and technical support for the discipline

Audience type	Equipment of the auditorium	Specialized educational/laboratory equipment, software and materials for mastering the discipline (if necessary)
Lecture	A lecture hall equipped with specialized furniture, a whiteboard (screen), and multimedia presentation equipment.	Projector
Computer class	A computer room for conducting classes, group and individual consultations, ongoing monitoring and midterm assessment, equipped with personal computers (14 in total), a board (screen) and technical means for multimedia presentations.	MS Visual Studio Code, NotePad++, Python, GIT, MS Visual Studio Community Edition, MinGW, Varwin Education, Open VR, Windows Subsystem for Linux (WSL2), Glasgow Haskell Compiler, Cabal, Haskell Tool Stack, haskell-language-server, PostgreSQL.
Seminar	An auditorium for conducting seminar-type classes, group and individual consultations, ongoing monitoring and midterm assessment, equipped with a set of specialized furniture and technical means for multimedia presentations.	
For independent work	A classroom for independent student work (can be used for seminars and consultations), equipped with a set of specialized furniture and computers with access to the Electronic Information System.	

* - the classroom for independent work of students MUST be indicated!

7. EDUCATIONAL, METHODOLOGICAL AND INFORMATIONAL SUPPORT OF THE DISCIPLINE

Main literature:

1. Computer science. Basic course. Simonovich S.V., St. Petersburg: Peter, 2011 - 640 p.

2. Learning Python. Volume 1. 5th edition. M. Lutz, St. Petersburg: Dialectika, 2019. — 832 p.

3. Python 3. The Essentials. Prokhorenok N., Dronov V., St. Petersburg: BHV-Petersburg, 2019 — 610 p.

4. C Programming Language Brian W. Kernighan, D.M. Ritchie, M.: Williams, 2019 - 288 p.

5. How to Program in C. 7th edition. X. Deitel, P. Deitel, Moscow: BINOM, 2017 — 1000 p.

6. The C Programming Language. Lectures and Exercises. Stephen Prata. Moscow: Williams, 2015 — 928 p.

7. Algorithms. Handbook with examples in C, C++, Java, and Python. Heineman J., Pollis G., Selkov S., St. Petersburg: Alfa-Kniga LLC, 2017 — 432 p.
8. C++ programming language. Stroustrup B., Martynov N.N., M: Binom, 2011. - 1135 p.
9. How to Program in C++. 8th edition. X. Deitel, P. Deitel, Moscow: Binom, 2020 — 1032 p.
10. C++. Sacred knowledge. Dewhurst S., St. Petersburg: Symbol Plus, 2012 – 240 p.
11. Object-Oriented Design Patterns. Gamma E., Helm R., Johnson R., Vlissides J., St. Petersburg: Piter, 2020 — 448 p.
12. Algorithms. Handbook with examples in C, C++, Java, and Python. Heineman J., Pollis G., Selkov S., St. Petersburg: Alfa-Kniga LLC, 2017 — 432 p.

Further reading:

1. Practical Python Programming

- The Computer Science Book: A complete introduction to computer science in one book. Johnson Thomas, Canada: Leanpub, 2020, - 410 p.
- Automating Routine Tasks with Python: A Practical Guide for Beginners. Sweirart, E., Moscow: Williams Publishing House, 2017. 592 p.
- Classic Computer Science Problems in Python. D. Kopets. St. Petersburg: Piter, 2020. — 256 p.
- The Big Book of Python Projects. Sweigart El. St. Petersburg: Piter, 2022 — 432 p.;
- Learning Python: Game Programming, Data Visualization, Web Applications. Matiz, E. St. Petersburg: Piter, 2020 — 512 p.

2. Algorithms, tools and theory

- Algorithms: construction, analysis and implementation in the C programming language. Vorozhtsov A.V., Vinokurov N.A., Moscow: MIPT, 2007 — 452 p.
- Programming and computer science. Antonyuk V.A., Ivanov A.P., Moscow: Faculty of Physics. Moscow State University named after M. V. Lomonosova, 2015 - 64 p.
- Pro Git. Version 2.1.x. Scott Chacon, Ben Straub, USA, New York: Apress, 2020 - 506 p. URL: https://git-scm.com/book/en/v2

3. Object-oriented programming

- Object-oriented thinking. Weisfeld M., St. Petersburg: Piter, 2014 — 304 p.
- Object-oriented programming: Workshop. Pavlovskaya T.A., Shchupak Yu.A., St. Petersburg: Piter, 2006. - 265 p.
- Structures and algorithms for data processing: object-oriented approach and implementation in C++. Kubensky A.A. St. Petersburg: BHV-Petersburg, 2004 — 464 p.

Resources of the information and telecommunications network "Internet":

1. RUDN University Electronic Library System and third-party electronic library systems to which university students have access based on concluded agreements

- Electronic library system of RUDN - ELS RUDN http://lib.rudn.ru/MegaPro/Web
- Electronic Library System "University Library Online" http://www.biblioclub.ru
- EBS Yurayt http://www.biblio-online.ru
- Electronic Library System "Student Consultant" www.studentlibrary.ru
- Electronic Library System "Troitsky Bridge"

2. Databases and search engines

- electronic fund of legal and regulatory documentation http://docs.cntd.ru/
- Yandex search engine https://www.yandex.ru/
- Google search engine https://www.google.ru/
- SCOPUS abstract database http://www.elsevierscience.ru/products/scopus/

Educational and methodological materials for independent work of students in mastering a discipline/module:*

1. Lecture course on the subject "Computer Science and Programming".

* - all teaching and methodological materials for independent work of students are posted in accordance with the current procedure on the discipline page in TUIS!

DEVELOPER:

Associate Professor		Saltykova Olga Alexandrovna
Position, DEPARTMENT	*Signature*	*Surname I.O.*

HEAD OF THE DEPARTMENT:

Head of Department		Razumny Yuri Nikolaevich
Position of the DEPARTMENT	*Signature*	*Surname I.O.*

HEAD OF THE EP HE:

Professor		Razumny Yuri Nikolaevich
Position, DEPARTMENT	*Signature*	*Surname I.O.*