KHMELNYTSKYI NATIONAL UNIVERSITY

APPROVED
Dean of IT Faculty
T. HOVORUSHCHENKO
"07" 09 2025.

WORKING PROGRAMME OF THE EDUCATIONAL COMPONENT
Programming

Field of Study: F – Information Technology
Specialty: F2 – Software Engineering
Level of Higher Education: First (Bachelor's) Level
Educational and Professional Programme: Software Engineering
Course Load: 8 ECTS credits **Course Code:** CPT.03
Language of Instruction: English
Status of the Educational Component: Compulsory (Professional Training)
Faculty: Faculty of Information Technology
Department: Department of Software Engineering

Form of Study	Year	Semester	Total Credits		Number of hours						Course project	Coursework	Semester control form	
					Contact Hours					Independent Work (incl. Individual Tasks)				
			ECTS credits	hours	Total	Lectures	Laboratory works	Practical classes	Seminar classes				pass/ fail test	Exam
D	1	1	8	180	82	32	50			158				+

The working programme is based on the Educational and Professional Programme "Software Engineering" within the specialty F2 "Software Engineering".

Program's author _____ DSc, Prof. V.V. Martynyuk

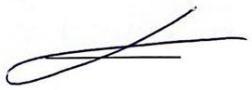
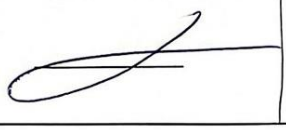
Approved at the meeting of the Department of Software Engineering
Minutes No. 1 dated August 28, 2025
Head of the Department _____ L.P. Bedratyuk

The working programme was reviewed and approved by the Academic Council of the Faculty of Information Technology

Chair of the Academic Council _____ Tetiana HOVORUSHCHENKO

Khmelnytskyi 2025

LETTER OF APPROVAL

Position	Department Name	Signature	First Name, LAST NAME
Head of *D*epartment DSc, Prof.	Software Engineering		Leonid BEDRATIUK
Programme Guarantor DSc, Prof.	Software Engineering		Leonid BEDRATIUK

PROGRAMMING

Type of Educational Component	Compulsory
Level of Higher Education	First (Bachelor's) Level
Language of Instruction	English
Semester	First
Number of ECTS Credits Assigned	8
Forms of Study the Course is Designed For	Full-time

Learning Outcomes. Upon successful completion of the course, the student should be able to: *understand* the principles of structured programming and the syntax of the C programming language; *declare and use* variables, constants, data types, and operators in C; *apply* control structures (conditional statements, loops, switch) to implement program logic; *design and implement* functions with parameters and return values, and apply scope and storage classes; *work* with arrays, strings, and pointers, including pointer arithmetic and dynamic memory allocation; *manipulate* data using structures, unions, and enumerations; *perform* input/output operations using standard C libraries; *implement* recursion and iterative algorithms in C; *design* modular programs by separating code into multiple source and header files; *use* preprocessing directives (macros, conditional compilation) effectively; *debug*, test, and document C programs according to good programming practices, and *use* tools such as compilers, linkers, and debuggers within an integrated development environment.

Course Content. Fundamentals of programming in C. Control structures. Functions, parameter passing, recursion, and modular programming. Arrays, strings, and pointers. Structures, unions, enumerations, and typedef. File input/output and error handling. Preprocessor directives: macros, conditional compilation, header files.

Planned Learning Activities. The minimum amount of classroom-based learning activities in one ECTS credit for a course at the first (Bachelor's) level of higher education in full-time study mode is 10 hours per 1 ECTS credit.

Forms (Methods) of Instruction: Lectures (using problem-based learning and visualisation methods), Laboratory works, Independent work

Assessment Methods: Laboratory work defence, Testing

Form of Final Assessment: Exam

Learning Resources:

1. Orhan Gazi. Modern C Programming: Including Standards C99, C11, C17, C23. Springer, 2023. – 405 p.
2. Robert C. Seacord. Effective C, 2nd Edition: An Introduction to Professional C Programming. No Starch Press, 2024. – 312 p.
3. Anthony Wallit. Learning C Programming. Independently published, 2022. – 250 p.
4. Srihari Radhakrishnan. C Programming: A Modern Approach to Learn the Fundamentals. Independently published, 2021. – 310 p.
5. University Electronic Library. [Electronic resource]. – Available at: http://library.khmnu.edu.ua/
6. Institutional Repository of Khmelnytskyi National University. [Electronic resource]. – Available at: http://elar.khmnu.edu.ua/jspui/?locale=en
7. Modular Learning Environment. [Electronic resource]. – Available at: https://msn.khmnu.edu.ua/

Lecturer: Doctor of Technical Sciences, Full Professor Martynyuk V.V.

3. EXPLANATORY NOTE

The course "Programming" is one of the general training courses and occupies a leading place in the training of students of the first (Bachelor's) level of higher education, full-time mode of study (hereinafter – full-time), who study under the Educational and Professional Programme "Software Engineering" within the specialty F2 "Software Engineering".

Prerequisites – Introductory
Postrequisites – CPT.04 Object-Oriented Programming, CPT.05 Software Engineering Basics, CPT.06 Databases

In accordance with the educational programme, the course contributes to the development of: **competences:** GC1. Ability for abstract thinking, analysis and synthesis. PC10. Ability to accumulate, process, and systematise professional knowledge regarding the creation and maintenance of software and recognise the importance of lifelong learning. PC13. Ability to reasonably choose and master the toolkit for software development and maintenance. PC14. Ability for algorithmic and logical thinking.

programme learning outcomes: PLO1 Analyse, purposefully search for and select information and reference resources and knowledge necessary to solve professional problems, taking into account modern advances in science and technology. PLO07 To understand and apply in practice the fundamental concepts, paradigms, and basic principles of functioning linguistic, instrumental, and computational tools of software engineering. PLO10 To conduct a pre-project survey of the subject area and system analysis of the design object. PLO11 To select initial data for design, guided by formal methods of requirement descriptions and modelling. PLO12 To apply effective software design approaches in practice. PLO13 Know and apply methods for developing algorithms, designing software, and data and knowledge structures. PLO15 Select programming languages and development technologies in a well-grounded manner to solve tasks related to the creation and maintenance of software.

Purpose of the course. To develop in students the competences in structured and modular programming using the C language, including problem analysis, algorithm design, and implementation of programs with attention to efficiency, correctness, and good programming practices.
Subject of the course. Fundamentals of structured programming in the C language, including program design, implementation of algorithms, use of basic data structures, and application of modern programming tools.
Course objectives. To provide students with knowledge and practical skills in designing, implementing, and debugging programs in C, using structured and modular programming techniques, basic data structures, and standard libraries to solve typical computational problems efficiently.
Learning Outcomes. Upon successful completion of the course, the student should be able to: understand the principles of structured programming and the syntax of the C programming language; declare and use variables, constants, data types, and operators in C; apply control structures (conditional statements, loops, switch) to implement program logic; design and implement functions with parameters and return values, and apply scope and storage classes; work with arrays, strings, and pointers, including pointer arithmetic and dynamic memory allocation; manipulate data using structures, unions, and enumerations; perform input/output operations using standard C libraries; implement recursion and iterative algorithms in C; design modular programs by separating code into multiple source and header files; use preprocessing directives (macros, conditional compilation) effectively; debug, test, and document C programs according to good programming practices, and use tools such as compilers, linkers, and debuggers within an integrated development environment

4. STRUCTURE OF THE COURSE CREDITS

Topic Title	Number of hours allocated to:		
	Lectures	Lab work	Indep endent work
Topic 1. Fundamentals of programming in C	4	6	24
Topic 2. Control structures	6	8	24
Topic 3. Functions, parameter passing, recursion, and modular programming	6	8	24
Topic 4. Arrays, strings, and pointers	6	8	24
Topic 5. Structures, unions, enumerations, and typedef	6	8	24
Topic 6. File input/output and error handling	2	6	24
Topic 7. Preprocessor directives	2	6	14
Total	32	50	158

5.1. CONTENT OF THE LECTURE COURSE

Lecture No.	**List of Lecture Topics and Annotations**	**Hours**
Topic 1. Fundamentals of programming in C		
1	Overview of the C programming language. History and characteristics of C. Program structure, basic syntax, keywords, and identifiers. Variables, constants, and data types. Operators and expressions. Compilation and execution process. Introduction to debugging and testing techniques. Writing and running simple programs. Ref.: [1] pp. 1–60; [2] pp. 10–50; [3] pp. 5–45; [4] pp. 1–55	4
Topic 2. Control structures		
2	Conditional statements: if, if-else, nested conditions. Switch-case statements. Loops: while, do-while, for. Nested loops and flow control. Using break, continue, and goto. Designing control flow for program logic. Ref.: [1] pp. 61–100; [2] pp. 51–90; [3] pp. 46–80; [4] pp. 56–90	8
Topic 3. Functions, parameter passing, recursion, and modular programming		
3	Definition and declaration of functions. Parameter passing: by value and by reference. Return values. Scope and lifetime of variables. Recursive functions and their applications. Modular programming and code organization. Using multiple source and header files. Ref.: [1] pp. 101–150; [2] pp. 91–140; [3] pp. 81–120; [4] pp. 91–140	8

Lecture No.	List of Lecture Topics and Annotations	Hours
Topic 4. Arrays, strings, and pointers		
4	Declaring and initializing arrays. Multidimensional arrays. String handling and manipulation. Pointer concepts and arithmetic. Pointers to arrays, functions, and structures. Dynamic memory allocation and deallocation. Ref.: [1] pp. 151–200; [2] pp. 141–190; [3] pp. 121–170; [4] pp. 141–190	8
Topic 5. Structures, unions, enumerations, and typedef		
5	Defining and using structures. Nested structures. Unions and their memory allocation. Enumerations and symbolic constants. Typedef for custom types. Organizing complex data in C programs. Ref.: [1] pp. 201–250; [2] pp. 191–240; [3] pp. 171–210; [4] pp. 191–240	8
Topic 6. File input/output and error handling		
6	Working with text and binary files. fopen, fclose, fread, fwrite, fprintf, fscanf. File pointers and file positioning. Error detection and handling. Practical examples of reading and writing data. Using standard C I/O library. Ref.: [1] pp. 251–300; [2] pp. 141–290; [3] pp. 211–250; [4] pp. 241–290	6
Topic 7. Graph algorithms		
7	Using macros and constants (#define). Conditional compilation (#if, #ifdef, #ifndef, #else, #endif). Including header files (#include). Compiler directives and code organization. Practical applications of the preprocessor in program development. Ref.: [1] pp. 301–350; [2] pp. 291–312; [3] pp. 251–280; [4] pp. 291–310	6
	Total	**32**

5.2. CONTENT OF LABORATORY WORKS

Topic No.	Laboratory Session Topic	Hours
1	Linear algorithms. Ref.: [1] pp. 30–40; [2] pp. 6–9; [3] pp. 18–28	6
2	Branched algorithms. Ref.: [4] pp. 25–35	6

Topic No.	Laboratory Session Topic	Hours
3	Programming of cyclic algorithms. Ref.: [4] pp. 65–78	6
4	Arrays. Work with one-dimensional arrays. Ref.: [1] pp. 123–144; [4] pp. 93–99	8
5	Two-dimensional arrays. Ref.: [1] pp. 166–184	8
6	Functions. Ref.: [2] pp. 132–138	8
7	Strings. Ref.: [1] pp. 453–474	6
8	Streams and files. Ref.: [1] pp. 476–507	6
Total for the semester		**50**

5.3. CONTENT OF INDEPENDENT WORK

Independent work of students of all forms of study involves systematic processing of the course material from relevant sources, preparation for laboratory works and testing. Students have access to the course page in the Modular Learning Environment, which contains the Working Programme of the course and the necessary teaching and learning materials.

Week No.	Type of Independent Work	Hours
1	Study of theoretical material from T1, preparation for Laboratory Work No. 1	9
2	Study of theoretical material from T1, preparation for Laboratory Work No. 1	9
3	Study of theoretical material from T2, preparation for Laboratory Work No. 2	9
4	Study of theoretical material from T2, preparation for Laboratory Work No. 2	9
5	Study of theoretical material from T3, preparation for Laboratory Work No. 3	9
6	Study of theoretical material from T3, preparation for Laboratory Work No. 3	9
7	Study of theoretical material from T4, preparation for Laboratory Work No. 4	9
8	Study of theoretical material from T4, preparation for Laboratory Work No. 4. Preparation for TC No. 1	9
9	Study of theoretical material from T5, preparation for Laboratory Work No. 5	9
10	Study of theoretical material from T5, preparation for Laboratory Work No. 5	9

Week No.	Type of Independent Work	Hours
11	Study of theoretical material from T6, preparation for Laboratory Work No. 6	9
12	Study of theoretical material from T6, preparation for Laboratory Work No. 6	9
13	Study of theoretical material from T7, preparation for Laboratory Work No. 7	9
14	Study of theoretical material from T7, preparation for Laboratory Work No. 7	9
15	Study of theoretical material from T7, preparation for Laboratory Work No. 8	9
16	Study of theoretical material from T7, preparation for Laboratory Work No. 8. Preparation for TC No. 2	9
17	Study of theoretical material from T7	14
Total:		**158**

Notes: TC – Test Control, T1–T7 – Topics of the course.

6. TECHNOLOGIES AND TEACHING METHODS

The learning process for the course is based on the use of both traditional and modern teaching technologies and methods, in particular: lectures (using visualisation methods, problem-based and interactive learning, motivational techniques, and information and communication technologies); laboratory works (using training exercises, problem situation analysis, explanation, discussions, etc.); independent work (study of theoretical material, preparation for laboratory works, ongoing and final assessment), with the use of information and computer technologies and distance learning technologies.

7. METHODS OF ASSESSMENT

Ongoing assessment is carried out during practical classes, as well as on the days of control activities established by the working programme and the academic schedule.
The following methods of ongoing assessment are used:
– test-based assessment of theoretical material;
– evaluation of the results of laboratory work defence.
When determining the final semester grade, the results of both ongoing assessment and final assessment are taken into account. The final assessment is conducted on all the material of the course according to examination papers prepared in advance and approved at the meeting of the department.
A student who has scored less than 60 percent of the maximum score for any type of academic work is not allowed to undergo the semester assessment until the amount of work stipulated by the Working Programme is completed. A student who has achieved a positive weighted average score (60 percent or more of the maximum score) for all types of ongoing assessment but has failed the examination is considered to have an academic debt.
Elimination of academic debt for the semester assessment is carried out during the examination session or according to the schedule set by the dean's office in accordance with the *Regulation on Control and Assessment of Learning Outcomes of Students at Khmelnytskyi National University*.

8. COURSE POLICY

The policy of the academic course is generally determined by the system of requirements for the student as stipulated by the current University regulations on the organisation and teaching and learning support of the educational process. In particular, this includes completing the safety briefing; attendance at course classes is compulsory. For valid reasons (documentarily confirmed), theoretical training may, with the lecturer's approval, take place online. Successful completion of the course and the formation of professional competences and programme learning outcomes require preparation for each laboratory work (studying the theoretical material for the topic of the work), active participation during the class, thorough preparation of the report, defence of the results, participation in discussions regarding the constructive decisions made during the laboratory works, etc.

Students must meet the established deadlines for completing all types of academic work in accordance with the Working Programme of the course. A missed laboratory class must be completed within the deadline set by the lecturer, but no later than two weeks before the end of the theoretical classes in the semester.

The student's mastery of the theoretical material of the course is assessed through testing.

When performing laboratory work, the student must comply with the policy of academic integrity (cheating, plagiarism — including with the use of mobile devices — is prohibited). If a violation of academic integrity is detected in any type of academic work, the student receives an unsatisfactory grade and must re-do the task on the relevant topic (type of work) as stipulated by the Working Programme. Any form of academic dishonesty is unacceptable.

Within the framework of studying the course, students are provided with recognition and crediting of learning outcomes acquired through non-formal education, available on accessible platforms (https://prometheus.org.ua/, https://www.coursera.org/), which contribute to the formation of competences and the deepening of learning outcomes defined in the Working Programme of the course, or ensure the study of a relevant topic and/or type of work from the course syllabus (for more details, see the *Regulation on the Procedure for Recognition and Crediting of Learning Outcomes of Students at Khmelnytskyi National University*).

9. ASSESSMENT OF STUDENTS' LEARNING OUTCOMES DURING THE SEMESTER

Assessment of a student's academic achievements is carried out in accordance with the *Regulation on the Control and Assessment of Students' Learning Outcomes at Khmelnytskyi National University*. During the ongoing assessment of the work performed by the student for each structural unit and the results obtained, the lecturer awards a certain number of points as set out in the Working Programme for that type of work.

Each structural unit of academic work may be credited only if the student has scored at least 60 percent (the minimum level for a positive grade) of the maximum possible points assigned to that structural unit.

When assessing students' learning outcomes for any type of academic work (structural unit), it is recommended to use the generalised criteria provided below:

Table – Assessment Criteria for Student Learning Outcomes

Grade and Level of Achievement of Intended Learning Outcomes and Competences	General Description of Assessment Criteria
Excellent *(High)*	The student has deeply and fully mastered the course content, confidently navigates it, and skilfully uses the conceptual framework; demonstrates the ability to connect theory with practice, solve practical problems, and clearly express and justify their reasoning. An excellent grade implies a logical presentation of the answer in the language of instruction (oral or written), high-quality formatting of the work, and proficiency in using specialised tools, instruments, or application software. The student demonstrates confidence when answering reformulated questions, is capable of making detailed and summarised conclusions, and shows practical skills in solving professional tasks. The answer may contain two or three minor inaccuracies.
Good *(Average)*	The student has shown full understanding of the course content, possesses the conceptual framework, and navigates the material well; applies theoretical knowledge consciously to solve practical tasks. The answer is generally well-articulated, although some minor inaccuracies or vague formulations of rules or principles may occur. The student's answer is based on independent thinking. Two or three minor mistakes are acceptable.
Satisfactory *(Sufficient)*	The student demonstrates knowledge of the basic course material sufficient for continued learning and practical activity in the profession; is able to complete the practical tasks foreseen by the programme. The answer is usually based on reproductive thinking. The student has limited knowledge of the structure of the discipline, makes inaccuracies and significant errors in the answer, and hesitates when answering reformulated questions. Nevertheless, they possess basic skills to complete simple practical tasks that meet the minimum assessment criteria and, under the lecturer's guidance, can correct their mistakes.
Unsatisfactory *(Insufficient)*	The student demonstrates fragmented, unstructured knowledge, cannot distinguish between main and secondary ideas, makes conceptual errors, misinterprets definitions, presents material in a chaotic and unconfident manner, and cannot apply knowledge to solve practical problems. An unsatisfactory grade is typically given to a student who is unable to continue learning the subject without additional study.

Structuring of the Course by Types of Academic Work and Assessment of Student Learning Outcomes

In-Class Work								***Assessment Activities***		***Semester Final Assessment***	
Laboratory Work №:								Test control:		Exam	Total
1	2	3	4	5	6	7	8	T 1-3	T 4-6		
Number of points per type of academic work (min–max)											
3-5	3-5	3-5	3-5	3-5	3-5	3-5	3-5	6-10	6-10	24-40	**60-100***
24-40								**12-20**		**24-40**	

Notes: If the number of points earned for any type of academic work in the course is below the established minimum, the student receives a failing grade and must retake the work within the deadline set by the lecturer (or dean). The institutional grade is determined in accordance with the table "**Correspondence between the Institutional Grading Scale and the ECTS Grading Scale**".

Assessment of Laboratory Work Defence Results

A laboratory work completed and formatted in accordance with the requirements established in the Methodological Guidelines is comprehensively assessed by the lecturer during its defence based on the following criteria:

– independence and accuracy of execution;

– completeness of the answer and understanding of the principles of building machine learning models;

– ability to justify the choice of algorithm or method;

– correctness of model implementation in the Python programming environment using appropriate libraries;

– ability to interpret the results of modelling and evaluate their suitability for solving the given task.

When assessing a laboratory session, the lecturer uses the generalised criteria outlined in the table "Assessment Criteria for Student Learning Outcomes" (minimum passing score – 3 points, maximum – 5 points).

If the student demonstrates a knowledge level below 60 percent of the maximum score established in the Working Programme for each structural unit, the laboratory work is not credited. In such a case, the student must study the topic more thoroughly, review the methodology, correct major mistakes, and re-defend the work at the time set by the lecturer.

Assessment of Test-Based Control Results

Each test included in the Working Programme consists of 30 test items, each carrying equal weight. According to the table for structuring types of academic work, the student may receive between 3 and 5 points depending on the number of correct answers.

Distribution of points depending on the number of correct answers to test items:

The test duration is 30 minutes. Students complete the test online in the Modular Learning Environment.

If a failing grade is received, the test must be retaken before the next scheduled assessment.

Distribution of points depending on correct answers to test questions

Number of Correct Answers	1-17	18-23	24-26	27-30
Percentage of Correct Answers	0-59	60-79	80-89	90-100
Number of Points	-	3	4	5

The final semester grade according to the institutional grading scale and the ECTS grading scale is determined automatically after the lecturer enters the assessment results in points for all types of academic work into the electronic gradebook. The correspondence between the institutional grading scale and the ECTS grading scale is provided in the table "Correspondence" below.

Assessment of the Final Semester Control (Exam)

The educational programme provides for a final semester control in the form of an examination, the purpose of which is to systematically and objectively assess both the theoretical and practical preparation of the student in the course. The examination is conducted according to examination papers prepared in advance and approved at the meeting of the department. In accordance with this, the examination paper contains a combination of both theoretical questions (including in test form) and practical tasks.

Table – Assessment of Final Semester Examination Results *for full-time students (40 points allocated for final control)*

Type of Task	**For each individual type of task**		
	Minimum (Satisfactory) Score	***Potential Positive Score (Good)****	**Maximum (Excellent) Score**
Theoretical Question № 1	3	4	5
Theoretical Question № 2	3	4	5
Practical Tasks *(6 tasks worth 3 points each)*	18	24	30
Total:	**24**		**40**

Note. ***A passing score for the exam, different from the minimum (24 points) and the maximum (40 points), falls within the range of 25–39 points and is calculated as the sum of points for all structural elements (tasks) of the exam.***

For each individual type of task in the final semester assessment, the assessment criteria for student learning outcomes provided above (see **Table – Assessment Criteria for Student Learning Outcomes**) are applied.

The final semester grade according to the institutional grading scale and the ECTS grading scale is determined automatically after the lecturer enters the assessment results in points for all types of academic work into the electronic gradebook. The correspondence between the institutional grading scale and the ECTS grading scale is shown below in the **Correspondence Table**.

The final examination grade is recorded if the total number of points accumulated by the student in the course as a result of ongoing assessment falls within the range of 60 to 100 points. In this case, a grade of *Excellent/Good/Satisfactory* is assigned according to the institutional scale, and a letter grade is assigned according to the ECTS scale, corresponding to the total number of points earned by the student as specified in the **Correspondence Table**.

Table – Correspondence between the Institutional Grading Scale and the ECTS Grading Scale

ECTS Grade	**Rating Scale (Points)**	**Institutional Grade(Level of Achievement of the Intended Learning Outcomes in the Course)**	
		Pass/ Fail	**Exam / Graded Credit**
A	90-100	Pass	**Excellent** – a high level of achievement of the intended learning outcomes in the course, indicating the learner's full readiness for further study and/or professional activity in the field.
B	83-89		**Good** – an average (maximally sufficient) level of achievement of the intended learning outcomes in the course and readiness for further study and/or professional
C	73-82		

			activity in the field.
D	66-72		**Satisfactory** – the student has demonstrated a minimally sufficient level of achievement of the learning outcomes required for further study and/or professional activity in the field.
E	60-65		
FX	40-59	Fail	**Fail** – several intended learning outcomes in the course have not been achieved. The level of acquired learning outcomes is insufficient for further study and/or professional activity in the field.
F	0-39		**Fail** – no learning outcomes have been achieved.

10. SELF-ASSESSMENT QUESTIONS ON LEARNING OUTCOMES

1. History and key features of the C programming language.
2. Structure of a C program, compilation and execution process.
3. Basic syntax, keywords, and identifiers in C.
4. Data types, variables, and constants in C.
5. Operators and expressions in C programs.
6. Control structures: if, if-else, switch-case.
7. Control structures: loops (for, while, do-while).
8. Using break, continue, and goto in program flow control.
9. Functions: declaration, definition, and invocation.
10. Parameter passing mechanisms in C.
11. Recursion: definition, properties, and examples.
12. Modular programming: header files and multiple source files.
13. Arrays: declaration, initialization, and operations.
14. Multidimensional arrays and their applications.
15. Strings: declaration, initialization, and basic operations.
16. String handling functions from <string.h>.
17. Pointers: concept, declaration, and basic operations.
18. Pointer arithmetic and pointers to arrays.
19. Pointers to functions and their applications.
20. Dynamic memory allocation: malloc, calloc, realloc, free.
21. Memory leaks and strategies to avoid them.
22. Structures: declaration, initialization, and access.
23. Nested structures and arrays of structures.
24. Unions: definition, properties, and applications.
25. Enumerations: purpose and examples.
26. Typedef: creating new type names.
27. File handling: text file operations.
28. File handling: binary file operations.
29. File error handling and EOF detection.
30. Standard I/O library functions.
31. Preprocessor directives: #define and macros.
32. Preprocessor directives: #include and header files.
33. Conditional compilation: #if, #ifdef, #ifndef.
34. Inline functions and macro functions.

35. Bitwise operators and their applications.
36. Working with command-line arguments (argc, argv).
37. Using standard library functions from <stdlib.h>.
38. Random number generation in C.
39. Sorting arrays using library functions (qsort).
40. Searching in arrays using library functions (bsearch).
41. Error handling and return codes in C programs.
42. Debugging techniques in C programming.
43. Testing simple C programs.
44. Writing structured, modular, and maintainable C code.
45. Code documentation and commenting best practices.
46. Using makefiles for program compilation.
47. Linking with static and dynamic libraries.
48. Introduction to C standard (C99, C11, C17, C23).
49. Best practices for secure and portable C programming.

11. EDUCATIONAL AND METHODOLOGICAL SUPPORT

The educational process for the course *"*Algorithms and Data Structures*"* is supported with all necessary instructional and methodological materials, which are available in the Modular Learning Environment MOODLE:

1. Course *"*Programming*"*: https://msn.khmnu.edu.ua/course/view.php?id=8709
2. Methodological Guidelines for Laboratory Sessions: https://msn.khmnu.edu.ua/course/view.php?id=8709

12. RECOMMENDED LITERATURE

Primary

1. Orhan Gazi. Modern C Programming: Including Standards C99, C11, C17, C23. Springer, 2023. – 405 p.
2. Robert C. Seacord. Effective C, 2nd Edition: An Introduction to Professional C Programming. No Starch Press, 2024. – 312 p.
3. Anthony Wallit. Learning C Programming. Independently published, 2022. – 250 p.
4. Srihari Radhakrishnan. C Programming: A Modern Approach to Learn the Fundamentals. Independently published, 2021. – 310 p.
5. Nora Sandler. Writing a C Compiler: Build a Real Programming Language from Scratch. No Starch Press, 2024. – 792 p.

Supplementary

6. Jens Gustedt. Modern C (3rd ed., C23-ready). Manning, 2024.
7. Yung-Hsiang Lu, George K. Thiruvathukal. Intermediate C Programming (2nd ed.). CRC Press, 2024.
8. Christopher Preschern. Fluent C: Principles, Practices, and Patterns. O'Reilly, 2023.
9. Jeff Szuhay. Learn C Programming (2nd ed.). Packt, 2022.
10. Lewis Van Winkle. Hands-On Network Programming with C. Packt, 2019.
11. Kamran Amini. Extreme C: Taking You to the Limit in Concurrency, OOP, and the Most Advanced Capabilities of C. Packt, 2019.
12. Peter Prinz, Tony Crawford. C in a Nutshell (2nd ed.). O'Reilly, 2015.
13. Форкун Ю. Метрика диференційованої цикломатичної складності аналізу програмнго коду з використанням систем керування базами даних / Ю. Форкун, В. Мартинюк, Н.

Праворська, О. Лучицький // Вимірювальна та обчислювальна техніка в технологічних процесах. – 2023. – № 3. – С. 100-105.
14. Форкун Ю. Метод розробки та проектування архітектурної складової програмного застосунку / Форкун Ю., Мартинюк В, Яшина О.. Measuring and computing devices in technological processes, (4), 2023, С. 87–93.
15. Праворська Н. Конструювання програмного забезпечення за допомогою синхронного підходу: основні процеси та інструменти для ефективної реалізації DevOps / Н. Праворська, В. Мартинюк // Вісник Хмельницького національного університету. Технічні науки. – 2023. – Т. 1, № 5. – С. 182-191.
16. Олійник П. Удосконалений метод роботи з метриками покриття коду для забезпечення ефективного оцінювання результатів тестування програмного забезпечення / П. Олійник, В. Мартинюк // Вимірювальна та обчислювальна техніка в технологічних процесах. – 2023. – № 3. – С. 138-143.

13. INFORMATION RESOURCES

1. Electronic Library of the University. [Electronic resource]. – Access: http://library.khmnu.edu.ua/
2. Institutional Repository of Khmelnytskyi National University. [Electronic resource]. – Access: http://elar.khmnu.edu.ua/jspui/?locale=uk
3. Modular Learning Environment. [Electronic resource]. – Access: https://msn.khmnu.edu.ua/
4. Online course “Програмування на С++” – Prometheus URL: https://prometheus.org.ua/prometheus-plus/programming-c