INTERNATIONAL EUROPEAN UNIVERSITY
Education and Research Institute "European Business School"
Department of Information Technology

Approved by
The Scientific and Methodical Council of the University

Chair of SMC _____

PO WORKING PROGRAM OF THE ACADEMIC DISCIPLINE:

BASICS OF PROGRAMMING

Knowledge area: 12 Information Technology

Specialty: 121 Software Engineering

Educational program: 121 Software Engineering

Discipline status: Compulsory

Kyiv – 2023

The working program of the Fundamentals of programming academic discipline is based on the 121 Software Engineering educational and professional program for the first (Bachelor) level of the 121 Software Engineering specialty approved by the University Academic Council on May 30, 2023, protocol No. 4.

Developer: Zoia Sherman, PhD in Physics and Mathematics, associate professor

Reviewers: Oleksandr Nesterenko, Doctor of Science (Techn.), professor
Oleksandr Falovskyi, PhD in Technology

Guarantor of the educational program: Oleksandr Nesterenko, Doctor of Science (Techn.), professor

The working program of the academic discipline is reviewed and approved by the Department of Information Technology, protocol dd. August 31, 2023, No. 1.

Head of the Department
Doctor Science (Techn.),
professor _____ O.V. Nesterenko,

The program is reviewed and approved by the Academic Council of the European Business School, protocol dd. September 11, 2023, No. 1.

Chair of the Academic Council
PhD in Economics, associate professor,
Acting Director
of the European Business School _____ Y.S. Remyha,

Registration No. _9/23__

INTRODUCTION

The **program of the Basics of programming academic discipline** is designed according to the Higher Education Standard of Ukraine (hereinafter referred to as the Standard) of the knowledge area: 12 Information Technology, specialty: 121 Software Engineering.

Discipline description (annotation). This academic discipline is one of the professional disciplines for future software developers.

Table 1

<table>
<tr><td rowspan="2">Criteria</td><td rowspan="2">Knowledge area, training program, educational level</td><td colspan="2">Discipline characteristics</td></tr>
<tr><td>full-time mode of study</td><td>part-time mode of study</td></tr>
<tr><td>Number of credits – 4</td><td>Knowledge area:
12 INFORMATION TECHNOLOGY</td><td colspan="2"><u>Compulsory</u></td></tr>
<tr><td>Sections – 1</td><td rowspan="8">Specialty:
121 SOFTWARE ENGINEERING</td><td colspan="2">Year of training</td></tr>
<tr><td>Content sections – 1</td><td>2023</td><td></td></tr>
<tr><td rowspan="3">Individual research task:</td><td colspan="2">Semester</td></tr>
<tr><td>1st</td><td>2nd</td></tr>
<tr><td colspan="2">Lectures</td></tr>
<tr><td>Total amount of hours – 120</td><td>16 hours</td><td>16 hours</td></tr>
<tr><td colspan="2">Practical and laboratory classes</td></tr>
<tr><td>32 hours</td><td>32 hours</td></tr>
<tr><td rowspan="3">Weekly load:
class hours – 3
independent work of students – 4</td><td rowspan="3">Educational level:
Bachelor</td><td colspan="2">Independent work</td></tr>
<tr><td>12 hours</td><td>12 hours</td></tr>
<tr><td colspan="2">Type of control:</td></tr>
<tr><td>exam</td><td>exam</td></tr>
</table>

Subject matter of the academic discipline: theory and practice of applying basic algorithmic and basic data structures in programming based on state-of-the art software development technologies.

Interdisciplinary links: The academic discipline is related to such disciplines as Operating and software systems, Computer systems and networks.

1. GOAL AND OBJECTIVES OF THE ACADEMIC DISCIPLINE

1.1. The **goal** of the Fundamentals of programming discipline is to provide future specialists with the knowledge of basic concepts of algorithmization and techniques for applying basic algorithmic structures (software organization) and basic data structures (data organization) in programming.

1.2. Key objectives of the Fundamentals of programming discipline:

- key stages of software design;
- typical algorithmic constructions;
- principles of procedural and structured programming; particularities of applying advanced basic instrumental software tools intended for solving tasks related to crew preparation for flight;
- basic data types;
- derived data types: lists, pointers, references, arrays, structures, unions;
- program control operators; pre-processing commands;
- rules of working with functions;
- C++ input/output system;
- basic principles of working with files;
- the basis of designing software programs in controlled C++.

1.3. **Competencies and learning outcomes** encouraged by the discipline (interrelation with the statutory content of student training stipulated in learning outcome terms of the Standard).

According to the Standard requirements, the discipline provides students with the following ***competencies:***

Table 2

Integral competence	Ability to solve complicated specialized tasks and practical problems in software development characterized by complexity and uncertainty of conditions.
General competencies	Understanding of the processes occurring in the computer's software environment. Ability to apply knowledge in practical programming situations.
Specialized (professional, subject) competencies	Understanding of the interactions in the computer's software and operating environment. Ability to think algorithmically and logically. Understanding of the possibilities of practical application of operating systems methods and tools in developing application software.

Specification of competencies according to the National Qualifications Framework descriptors in the Competency matrix form is given in Table 3.

Competency matrix

No.	Competence	Knowledge	Skills / Abilities	Communication	Autonomy and responsibility
Integral competence					
1.	Ability to solve complicated specialized tasks and practical problems in software engineering characterized by complexity and uncertainty of conditions.	Theories of designing operating systems and interacting with application software	To use information technologies, basic system and application software to solve practical problems	Software interaction	Independent design and testing on the production site
General competencies					
2.	Understanding of the processes occurring in the operating system when processing information with application software. Ability to use knowledge in practical programming situations. Ability to search, process and analyze information for application in programming.	structure of operating systems, general principles of their functioning	to apply operations of interaction with the OS environment when developing application programs	Relation between theoretical and practical knowledge	Monitoring of information processing processes
Specialized (professional, subject) competencies					
3.	Understanding of the concept of processes and multitasking. Ability to think algorithmically and logically. Understanding of the practical application of operating system tools.	fundamentals of multiprogramming virtualization and distributed computing, relevant rules and functions of application programming	To use software tools in the operating environment	Application of parallel working technologies	Description of information processes

Integrated final program learning outcomes encouraged by the academic discipline:

Program learning outcomes Bachelor's qualifying paper

Learning outcomes:

After learning the discipline, students should

know:

- key stages of software design;
- typical algorithmic constructions;
- principles of procedural and structured programming; particularities of applying advanced basic instrumental software tools;
- basic data types;
- derived data types: lists, pointers, references, arrays, structures, unions;
- program control operators; pre-processing commands;
- rules of working with functions;
- C++ input/output system;
- basic principles of working with files;
- the basis of designing software programs in controlled C++.

be able to:

– use programming tools, basic system challenges to solve practical application programming problems;

– use functions and libraries of software tools for application development;

– apply technologies of working with modern basic instrumental software tools.

2. INFORMATION CAPACITY OF THE ACADEMIC DISCIPLINE

The Fundamentals of programming academic discipline consists of 120 hours / 4 ECTS credits.

Topic 1. Introduction. Basic concepts and definitions
Topic 2. C++ syntax and semantics.
Topic 3. C++ software structure. Declaring variables
Topic 4. Controlling input-output streams. Expressions
Topic 5. C++ language tools for implementing basic algorithm structures. The if operator.
Topic 6. Loop operators. While Loop.
Topic 7. Do-While Loops.
Topic 8. For Loops.
Topic 9. Complex data types in C++. One-dimensional arrays.
Topic 10. Processing one-dimensional arrays.
Topic 11. Complex data types in C++. Two-dimensional arrays.
Topic 12. Processing two-dimensional arrays.
Topic 13. Functions. Pointers and references
Topic 14. Line processing
Topic 15. Working with std: vector, list containers.
Topic 16. Working with std: degue, stack, gueue containers.
Topic 17. Working with std:map, multimap containers.
Topic 18. File input/output. Working with eof.

3. STRUCTURE OF THE ACADEMIC DISCIPLINE

Content modules and topics	Amount of hours				
	Total	including			
		l	p. c.	lab.	i. w.
1	2	3	4	5	6
Content module 1. Basic concepts of programming.					
Topic 1. Introduction. Basic concepts and definitions	4	1	2	-	1
Topic 2. C++ syntax and semantics.	4	1	2	-	1
Topic 3. C++ software structure. Declaring variables	4	1	2	-	1
Topic 4. Controlling input-output streams. Expressions	4	1	2	-	1
Topic 5. C++ language tools for implementing basic algorithm structures. The if operator.	5	2	2	-	1
Topic 6. Loop operators. While Loop.	5	2	2	-	1
Total per content module	26	8	12	-	6
Content module 2. Means of implementing basic structures in programming.					
Topic 7. Do-While Loops.	12	2	4	-	1
Topic 8. For Loop.	12	2	4	-	1
Topic 9. Complex data types in C++. One-dimensional arrays.	12	2	4	-	1
Topic 10. Processing one-dimensional arrays.	12	2	4	-	1
Topic 11. Complex data types in C++. Two-dimensional arrays.	12	2	4	-	1
Topic 12. Processing two-dimensional arrays.	12	2	4	-	1
Total per content module	42	12	24	-	6

Content module 3. Working with I/O functions and term variables.					
Topic 13. Functions. Pointers and references.	8	2	4	-	2
Topic 14. Line processing	10	2	6	-	2
Topic 15. Working with std: vector, list containers.	8	2	4	-	2
Topic 16. Working with std: degue, stack, gueue containers.	10	2	6	-	2
Topic 17. Working with std:map, multimap containers.	8	2	4	-	2
Topic 18. File input/output. Working with eof.	8	2	4	-	2
Total per content module	52	12	28	-	12
Total hours	120	32	64		24

4. TOPICS OF LECTURES

No.	Topic (brief content)	Amount of hours
1	Topic 1. Introduction. Basic concepts and definitions. Classification and particularities of modern programming languages; programming environments and elements of the Microsoft Visual Studio programming environment window;	1
2	Topic 2. C++ syntax and semantics. The set of C++ symbols and the set of represented symbols. Rules of forming constants, identifiers. Keywords. Using comments in programs. The concept of a lexeme.	1
3	Topic 3. C++ software structure. Declaring variables. Composition of a C++ software. Software output files. Software execution. Particularities of the main() function. The concept of lifetime and scope. Basic data types. Rules of conversion of basic data types. Variable modifiers. Automatic variables. Registry variables. External variables and functions of static variables.	1
4	Topic 4. Controlling input-output streams. Expressions. Basic input and output operators in C++, in particular, using printf() and scan() as examples. The format of input-output of different data types (numbers, strings, pointers, etc.). Arithmetic operations. Assignment operator. Expression concept. Increment and decrement operators. The sizeof operator. Bitwise logical operations. Left and right shift operations. Comparison	1

	operators. The coma operation. Priority and order of execution of operations.	
5	Topic 5. C++ language tools for implementing basic algorithm structures. The if operator. Conditional operators. If operators. If-else operators. Conditional operator ?:. Switch operator.	2
6	Topic 6. Loop operators. While Loop. The concept of a loop: While, Do While, For loops. Control operators in loops: break operator, continue operator.	2
7	Topic 7. Do-While Loops. Embedded loops. Microsoft Visual Studio integrated debugger.	2
8	Topic 8. For Loop. Means of processing elements of the For loop.	2
9	Topic 9. Complex data types in C++. One-dimensional arrays. The concept of data array. Declaration, initialization and output of arrays. Typical algorithms of array processing. Determination of maximum and minimum elements and their numbers.	2
10	Topic 10. Processing one-dimensional arrays. Algorithm for composing the elements of a one-dimensional array. Other methods of array ordering.	2
11	Topic 11. Complex data types in C++. Two-dimensional arrays. Initialization of two-dimensional arrays. Typical tasks in which two-dimensional arrays are used. Arrays of structures. Pointers to structures. Passing by reference members of arrays of structures. Merging and operations with them.	2
12	Topic 12. Processing two-dimensional arrays. Algorithms for processing elements of the two-dimensional array.	2
13	Topic 13. Functions. Pointers and references. The concept of structural programming. Declaring and calling functions. Passing arguments. Function prototypes. Scope of visibility. Local and global variables. Default arguments. Function overloads and templates. Recursion. General overview. Pointer naming. Pointer arithmetic. Pointers. to pointers. Pointers. to functions. References. Passing parameters by reference and value. Using pointers and references with the const keyword.	2
14	Topic 14. Line processing. Overview of the string class. Performing basic operations on string objects. Working with string objects using iterators.	2
15	Topic 15. Working with std: vector, list containers. Performing basic operations on vector and list objects. Working with vector and list objects using iterators.	2
16	Topic 16. Working with std: degue, stack, gueue containers. Performing basic operations on deque, stack, queue objects. Working with degue, stack, gueue objects using iterators.	2
17	Topic 17. Working with std:map, multimap containers. Performing basic operations on map and multimap objects. Working with map and multimap objects using iterators.	2
18	Topic 18. File input/output. Working with eof.	2

	I/O overview. Function settings and arguments. Default arguments. Namespace. Passing arguments. Function prototypes. Scope of visibility. Function overloads and templates. Recursion. Necessary steps for working with eof. Options and alternatives.	

5. TOPICS OF LABORATORY AND PRACTICAL CLASSES

No.	Topic (brief content)	Amount of hours
1	Topic 1. Introduction. Basic concepts and definitions. Classification and particularities of modern programming languages; programming environments and elements of the Microsoft Visual Studio programming environment window;	2
2	Topic 2. C++ syntax and semantics. The set of C++ symbols and the set of represented symbols. Rules of forming constants, identifiers. Keywords. Using comments in programs. The concept of a lexeme.	2
3	Topic 3. C++ software structure. Declaring variables. Composition of a C++ software. Software output files. Software execution. Particularities of the main() function. The concept of lifetime and scope. Basic data types. Rules of conversion of basic data types. Variable modifiers. Automatic variables. Registry variables. External variables and functions of static variables.	2
4	Topic 4. Controlling input-output streams. Expressions. Basic input and output operators in C++, in particular, using printf() and scan() as examples. The format of input-output of different data types (numbers, strings, pointers, etc.). Arithmetic operations. Assignment operator. Expression concept. Increment and decrement operators. The sizeof operator. Bitwise logical operations. Left and right shift operations. Comparison operators. The coma operation. Priority and order of execution of operations.	2
5	Topic 5. C++ language tools for implementing basic algorithm structures. The if operator. Conditional operators. If operators. If-else operators. Conditional operator ?:. Switch operator.	2
6	Topic 6. Loop operators. While Loop. The concept of a loop: While, Do While, For loops. Control operators in loops: break operator, continue operator.	4
7	Topic 7. Do-While Loops. Embedded loops. Microsoft Visual Studio integrated debugger.	4
8	Topic 8. For Loop. Means of processing elements of the For loop.	4
9	Topic 9. Complex data types in C++. One-dimensional arrays. The concept of data array. Declaration, initialization and output of arrays. Typical algorithms of array processing. Determination of maximum and minimum elements and their numbers.	4
10	Topic 10. Processing one-dimensional arrays.	4

	Algorithm for composing the elements of a one-dimensional array. Other methods of array ordering.	
11	Topic 11. Complex data types in C++. Two-dimensional arrays. Initialization of two-dimensional arrays. Typical tasks in which two-dimensional arrays are used. Arrays of structures. Pointers to structures. Passing by reference members of arrays of structures. Merging and operations with them.	4
12	Topic 12. Processing two-dimensional arrays. Algorithms for processing elements of the two-dimensional array.	4
13	Topic 13. Functions. Pointers and references. The concept of structural programming. Declaring and calling functions. Passing arguments. Function prototypes. Scope of visibility. Local and global variables. Default arguments. Function overloads and templates. Recursion. General overview. Pointer naming. Pointer arithmetic. Pointers. to pointers. Pointers. to functions. References. Passing parameters by reference and value. Using pointers and references with the const keyword.	4
14	Topic 14. Line processing. Overview of the string class. Performing basic operations on string objects. Working with string objects using iterators.	6
15	Topic 15. Working with std: vector, list containers. Performing basic operations on vector and list objects. Working with vector and list objects using iterators.	4
16	Topic 16. Working with std: degue, stack, gueue containers. Performing basic operations on deque, stack, queue objects. Working with degue, stack, gueue objects using iterators.	6
17	Topic 17. Working with std:map, multimap containers. Performing basic operations on map and multimap objects. Working with map and multimap objects using iterators.	4
18	Topic 18. File input/output. Working with eof. I/O overview. Function settings and arguments. Default arguments. Namespace. Passing arguments. Function prototypes. Scope of visibility. Function overloads and templates. Recursion. Necessary steps for working with eof. Options and alternatives.	4

6. INDEPENDENT WORK

No.	Topic (brief content)	Amount of hours
1	Topic 1. Introduction. Basic concepts and definitions. Classification and particularities of modern programming languages; programming environments and elements of the Microsoft Visual Studio programming environment window;	1
2	Topic 2. C++ syntax and semantics. The set of C++ symbols and the set of represented symbols. Rules of forming constants, identifiers. Keywords. Using comments in programs. The concept of a lexeme.	1
3	Topic 3. C++ software structure. Declaring variables.	1

	Composition of a C++ software. Software output files. Software execution. Particularities of the main() function. The concept of lifetime and scope. Basic data types. Rules of conversion of basic data types. Variable modifiers. Automatic variables. Registry variables. External variables and functions of static variables.	
4	Topic 4. Controlling input-output streams. Expressions. Basic input and output operators in C++, in particular, using printf() and scan() as examples. The format of input-output of different data types (numbers, strings, pointers, etc.). Arithmetic operations. Assignment operator. Expression concept. Increment and decrement operators. The sizeof operator. Bitwise logical operations. Left and right shift operations. Comparison operators. The coma operation. Priority and order of execution of operations.	1
5	Topic 5. C++ language tools for implementing basic algorithm structures. The if operator. Conditional operators. If operators. If-else operators. Conditional operator ?:. Switch operator.	1
6	Topic 6. Loop operators. While Loop. The concept of a loop: While, Do While, For loops. Control operators in loops: break operator, continue operator.	1
7	Topic 7. Do-While Loops. Embedded loops. Microsoft Visual Studio integrated debugger.	1
8	Topic 8. For Loop. Means of processing elements of the For loop.	1
9	Topic 9. Complex data types in C++. One-dimensional arrays. The concept of data array. Declaration, initialization and output of arrays. Typical algorithms of array processing. Determination of maximum and minimum elements and their numbers.	1
10	Topic 10. Processing one-dimensional arrays. Algorithm for composing the elements of a one-dimensional array. Other methods of array ordering.	1
11	Topic 11. Complex data types in C++. Two-dimensional arrays. Initialization of two-dimensional arrays. Typical tasks in which two-dimensional arrays are used. Arrays of structures. Pointers to structures. Passing by reference members of arrays of structures. Merging and operations with them.	1
12	Topic 12. Processing two-dimensional arrays. Algorithms for processing elements of the two-dimensional array.	1
13	Topic 13. Functions. Pointers and references. The concept of structural programming. Declaring and calling functions. Passing arguments. Function prototypes. Scope of visibility. Local and global variables. Default arguments. Function overloads and templates. Recursion. General overview. Pointer naming. Pointer arithmetic. Pointers. to pointers. Pointers. to functions. References. Passing parameters by reference and value. Using pointers and references with the const keyword.	2

14	Topic 14. Line processing. Overview of the string class. Performing basic operations on string objects. Working with string objects using iterators.	2
15	Topic 15. Working with std: vector, list containers. Performing basic operations on vector and list objects. Working with vector and list objects using iterators.	2
16	Topic 16. Working with std: degue, stack, gueue containers. Performing basic operations on deque, stack, queue objects. Working with degue, stack, gueue objects using iterators.	2
17	Topic 17. Working with std:map, multimap containers. Performing basic operations on map and multimap objects. Working with map and multimap objects using iterators.	2
18	Topic 18. File input/output. Working with eof. I/O overview. Function settings and arguments. Default arguments. Namespace. Passing arguments. Function prototypes. Scope of visibility. Function overloads and templates. Recursion. Necessary steps for working with eof. Options and alternatives.	2

7. TRAINING METHODS

Teaching the Fundamentals of programming discipline, one uses information and practical training methods: classical lectures, laboratory and practical classes using simulation laboratory workshops, as well as consultations on the accomplishment of independent work of students, written assignments.

Methods of learning and cognitive activity: explanatory and illustrative method, reproductive method, problem presentation method, partially exploratory or heuristic method, research method.

Methods of stimulation and motivation of learning and cognitive activity: inductive and deductive teaching methods; methods of stimulation and motivation of learning.

8. CONTROL METHODS

The plan of the Operating systems discipline implies carrying out of current and final control.

Current control is the assessment of the level of knowledge, skills and abilities of students carried out during the educational process by conducting a written survey at the end of sections (module colloquium). Final control is carried out in the form of an exam.

9. FORM OF STUDENT PERFORMANCE FINAL CONTROL

The form of final control is the **exam** taken on-campus (or in the form of computer test in case of a specific situation) in the period stipulated by the Dean's office or according to the individual schedule stipulated by the curriculum.

10. SCORING SYSTEM

Scoring during the semester

No.	Type of activity	Number of points per didactic unit	Number	Total points
1	Accomplishment of tests	2	8	16
2	Accomplishment of laboratory works	4	8	32
3	Accomplishment of independent tasks	1.5	8	12
Maximum grade				60

General assessment of student knowledge due to current control

The results of current control of student knowledge are assessed in general ranging from **0** to **60** points.

Students are allowed to final control if they fulfil the requirements of the training program and obtain at least **36** points for the current learning activity.

Final assessment of student knowledge

Final assessment of student knowledge is conducted in the form of **exam**.

Allocation of assessment points during final control in the academic discipline

Grade in points for final assessment	Grade according to the national scale
35-40	Excellent
21-34	Good
10-20	Satisfactory
less than 10	Fail

Assessing the answer to the particular question, one takes into account the following gaps and mistakes:

– untidy preparation of work (nonconventional abbreviations, unclear handwriting, use of pencils instead of clear inks) (minus **2** points);

– incorrectness in certain economic categories and definitions (minus **4** points).

Assessment criteria for answers to theoretical questions of the exam card:

1. The full answer to the question rated as ***excellent (40 points)*** should correspond to the following requirements:

– detailed, comprehensive representation of the content of the given problem;

– full list of economic categories and laws required to reveal the question;

– ability to carry out a comparative analysis of various theories, concepts, approaches and make logical conclusions and generalizations;

– ability to apply methods for the scientific analysis of economic phenomena, processes and characterize their features and forms of appearance;

– demonstration of the ability to express and reason your own attitude to alternative views on this question;

– use of relevant actual and statistical data, knowledge of dates and historical periods that prove key points of the answer.

2. The answer to the question is rated as ***good (30 points)*** if:

– the answer for the highest grade does not reveal at least one of the above-mentioned points (if it is definitely required to reveal the question comprehensively), or if:

– revealing the question correctly in general according to the above-mentioned requirements, one makes some mistakes while using digital materials.

3. The answer to the question is rated as ***satisfactory (20 points)*** if:

– the answer for the highest grade does not reveal four and more points specified in its requirements (if they are required to reveal the question comprehensively);

– there are four or more gaps characterizing individually assessment criteria;

– conclusions made during the answer do not correspond to correct or generally defined ones with the absence of evidence for opposite facts given in the answer;

– the character of the answer gives reason to state that persons fail to understand the question properly or do not know the correct answer, and that is why fail to answer in actual fact, making serious mistakes.

National and ECTS grading scale

<table>
<tr><td rowspan="2">Sum of points for all types of educational activities</td><td rowspan="2">ECTS grade</td><td colspan="2">Grade according to the national scale</td></tr>
<tr><td>for exam, term paper, practical training</td><td>for Pass/Fail test</td></tr>
<tr><td>90-100</td><td>A</td><td>excellent</td><td rowspan="5">pass</td></tr>
<tr><td>82-89</td><td>B</td><td rowspan="2">good</td></tr>
<tr><td>74-81</td><td>C</td></tr>
<tr><td>66-73</td><td>D</td><td rowspan="2">satisfactory</td></tr>
<tr><td>60-65</td><td>E</td></tr>
<tr><td>30-59</td><td>FX</td><td>fail with possible repeated pass</td><td>fail with possible repeated pass</td></tr>
<tr><td>1-29</td><td>F</td><td>fail with obligatory repeated learning of the discipline</td><td>fail with obligatory repeated learning of the discipline</td></tr>
</table>

The overall final grade in points according to the national and ECTS scales is put into the examination and test register, academic card and credit book of students.

11. METHODICAL SUPPORT:

- working program of the discipline;
- electronic course on the e-learning platform;

- plans of lectures, practical classes and independent work of students;
- key points of discipline lectures;
- methodical guidelines to laboratory and practical classes for students;
- methodical materials for independent work of students;
- test tasks for lecture topics;
- list of questions for the exam.

12. RECOMMENDED READING

Primary:

1. Pekarskyi B.H. Fundamentals of programming: Study guide. Condor, 2018. 364 p.
2. Vasiliev O.N. C++ tutorial with tasks and examples (+ virtual CD). Science and Technology, 2016. 480 p.
3. Sutter H. Solving complex problems in C++. Williams, 2015. 400 p.
4. George Heinemann, Gary Pollis, Stanley Selkow. Algorithms. Reference book with examples in C, C++, Java and Python. Translated from English. Dialectics, 2017. 432 p.
5. Fundamentals of programming: study guide / M.F. Bondarenko, O.H. Kachko. Kh.: SMIT Company, 2008. 432 p.
6. Habrusiev V.Y., Lapinskyi V.V., Nesterenko O.V. Fundamentals of operating systems: Core, process, thread. Ternopil: Bohdan, 2007. 94 p.

Additional:

1. C++ Crash Course: A Fast-Paced Introduction. / Lospinoso Josh. ISBN 1593278885. - 2019.- 792c.
2. International Standard ISO/IEC 14882:2014(E) – Programming Language C++, ISBN-13: 978- 0321563842: [Electronic resource]. – Available at: https://isocpp.org/std/the-standard.
3. C/C++ language and standard libraries reference: [Electronic resource]. – Available at: https://msdn.microsoft.com/en-us/library/hh875057.aspx. Fundamentals of programming: study guide for students in the 123 Computer engineering specialty / V.H. Zaitsev, I.P.Drobiazko; Kyiv: Igor Sikorsky Kyiv Polytechnic Institute. 2019. – 240 p. [Electronic resource]
4. Fundamentals of programming: study guide. [edited by V.M. Rudnytskyi] / I.M. Fedotova-Piven, I.V. Myronets, O.B. Piven, S.V. Sysoienko, T.V. Myroniuk; Cherkasy State Technological University. – Kharkiv.: DISA PLUS LLC, 2019. 216 p.

Information resources

1. https://www.microsoft.com/uk-ua/
2. https://stud.com.ua/informatika/
3. https://dou.ua/

4. http://it.ridne.net/
5. https://www.kernel.org/