Національний технічний університет України
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Computer Systems Software Department

ALGORITHMS AND DATA STRUCTURES

Syllabus

Requisites of the Course

Cycle of Higher Education	***First cycle of higher education (Bachelor's degree)***
Field of Study	*12 Information Technologies*
Speciality	*121 Software engineering*
Education Program	*Software Engineering of Multimedia and Information Retrieval Systems*
Type of Course	*Normative*
Mode of Studies	*full-time*
Year of studies, semester	*1 year (1, 2 semester)*
ECTS workload	*8 credits (ECTS). Time allocation: 72 hours for lectures, 54 hours for programming assignment, 114 hours for self-study.*
Testing and assessment	*Final test*
Course Schedule	*According to rozklad.kpi.ua*
Language of Instruction	*English*
Course Instructors	*Senior lecturer, Olga Sulema, PhD* *olga.sulema@pzks.fpm.kpi.ua*
Access to the course	Google classroom at https://classroom.google.com/u/0/c/Mzg5MzUwMzI2NzEw

Outline of the Course

1. Course description, goals, objectives, and learning outcomes

The study of the Algorithms and Data Structures course allows students to acquire competencies necessary for solving practical problems related to the development and use of algorithms and data structures.

The purpose of studying the Algorithms and Data Structures course is to build capacity to independently implement well-known algorithms; to develop, analyze complexity and implement their own algorithms; to use most common data structures.

The subject of the Algorithms and Data Structures course is the process of developing and applying algorithms and data structures.

After the course, students will:

know:

- *features and fundamental control structures of algorithms;*
- *basics of theory of algorithm complexity;*
- *methods of developing algorithms;*
- *most common data structures;*
- *search and sorting algorithms;*
- *hashing methods;*
- *algorithms on trees and graphs;*

be able to:

- *analyze algorithm complexity;*
- *generalize algorithms for various data structures;*
- *develop their own algorithms and data structures;*

have experience in:

- *designing algorithms using flowchart;*
- *programming developed algorithms;*
- *implement abstract data types using various data structures;*
- *using algorithms and data structures in real-life problems.*

2. **Prerequisites and post-requisites of the course (the place of the course in the scheme of studies in accordance with curriculum)**

The Algorithms and Data Structures course is a normative discipline and students do not need any specific initial knowledge for its study.

Theoretical knowledge and practical skills acquired in the Algorithms and Data Structures course provide the necessary background for studying programming disciplines in Bachelor and Master programs of 121 Software Engineering specialty.

3. **Content of the course**

Credit Module 1. Basics of Algorithmizing

Section 1. Introduction to Algorithms

Topic 1.1 Algorithms. Basic terms

Topic 1.2 Fundamental Control Structures of Algorithms

Topic 1.3 Complexity of Algorithms

Section 2. Array

Topic 2.1 Introduction to Data Structures. Array as a Linear Data Structure

Topic 2.2 Algorithms on Array

Topic 2.3 Search Algorithms

Topic 2.4 Sorting Algorithms

Section 3. Linked Lists

Topic 3.1 Linked List as a Linear Data Structure

Topic 3.2 Algorithms on Linked Lists

Topic 3.3 Sorting Algorithms with Linked Lists

Credit Module 2. Data Structures

Section 4. Recursive Algorithms

Topic 4.1 Recursion. Recursive Algorithms

Topic 4.2 Merge Sort

Topic 4.3 Quick Sort

Section 5. Linear Data Structures

Topic 5.1 Stack

Topic 5.2 Queue

Section 6. Hashtable

Topic 6.1 Dictionary. Introduction to Hashtable

Topic 6.2 Hashing Methods

Topic 6.3 Resolving Collisions in Hashtable

Section 7. Non-Linear Data Structures

Topic 7. Tree. Algorithms on Trees

Topic 8. Priority Queue

Topic 9. Graph. Algorithms on Graph

4. Coursebooks and teaching resources

Main literature:

1. Sulema O. Guidelines for Programming Assignment in the Algorithms and Data Structures course. KPI, 2021.

2. Bhasin H. Algorithms: Design and Analysis. Oxford University Press, 2015.

3. Roughgarden T. Algorithms Illuminated. SoundLikeYourself Publishing, 2017.

4. Cormen T. H., Leiserson Ch. E., Rivest R. L., Stein C. Introduction to Algorithms. MIT Press, 2009.

5. Stephens R. Essential Algorithms: a Practical Approach to Computer Algorithms using Python and C#. John Wiley & Sons, 2019.

6. Standard ECMA-4, Flow Charts. European Computer Manufacturers Association, 2nd ed., 1966.

Additional literature:

1. Mehlhorn K. Data structures and algorithms 1: Sorting and searching. Vol. 1. Springer Science & Business Media, 2013.

2. Mueller J. P., Massaron L. Algorithms for Dummies. John Wiley & Sons, 2017.

3. Erickson J. Algorithms. 2019.

4. Wengrow J. A Common-Sense Guide to Data Structures and Algorithms. Pragmatic Bookshelf, 2020.

Educational content

5. Methodology

Credit Module 1. Basics of Algorithmizing

No	*Type of a class*	*Materials for self-studying*
Section 1. Introduction to Algorithms		
1.	*Lecture 1. Introduction to Algorithms*	*6, №1*
2.	*Lecture 2. Ways of describing algorithms*	*6, №2*
3.	*Lecture 3. Algorithm Control Structures*	*6, №3*
4.	*Lecture 4. Algorithm Complexity*	*6, №4, 24*
5.	*Programming Assignment 1. Loop and Mixed Algorithms*	*Task: Develop and implement loop and mixed algorithms according to the variant.* *6, №5*
Section 2. Array		
6.	*Lecture 5. Introduction to Data Structures. Array as a Linear Data Structure*	*6, №6*
7.	*Lecture 6. Algorithms on Array*	*6, №7, 25*
8.	*Lecture 7. Two-Dimensional Array. Matrix Traversal*	*6, №8*

9.	Programming Assignment 2. Matrix Traversal	Task: Traverse a matrix according to a path defined by the variant. 6, №9
10.	Lecture 8. Search Algorithms – 1	6, №10, 25
11.	Lecture 9. Search Algorithms – 2	6, №11, 25
12.	Lecture 10. Sorting Algorithms – 1	6, №12
13.	Lecture 11. Sorting Algorithms – 2	6, №13
14.	Programming Assignment 3. Sorting Algorithms	Task: Sort elements according to a sorting algorithm defined by the variant. 6, №14
Section 3. Linked Lists		
15.	Lecture 12. Linked List as a Linear Data Structure	6, №15
16.	Lecture 13. Algorithms on Linked Lists	6, №16
17.	Programming Assignment 4. Linked Lists	Task: Implement a linked list according to the variant. 6, №17, 26
18.	Lecture 14. Sorting Algorithms – 3	6, №18
19.	Lecture 15. Sorting Algorithms – 4	6, №19
20.	Lecture 16. Real-life Problems – 1	6, №20, 27
21.	Lecture 17. Real-life Problems – 2	6, №20, 27
22.	Midterm Test 1	6, №22

Credit Module 2. Data Structures

No	Type of a class	Materials for self-studying
Section 4. Recursive Algorithms		
23.	Programming Assignment 5. Sorting Algorithms – 2	Task: Sort elements according to an algorithm defined by the variant. 6, №28
24.	Lecture 18. Recursion. Recursive Algorithms	6, №29
25.	Lecture 19. Merge Sort Algorithm	6, №30
26.	Lecture 20. Quick Sort Algorithm	6, №31
Section 5. Linear Data Structures		
27.	Lecture 21. Stack	6, №32, 52
28.	Lecture 22. Queue	6, №33, 52
29.	Programming Assignment 6. Linear Data Structures	Task: Implement a data structure

		defined by the variant. *6, №34, 52*
Section 6. Hashtable		
30.	*Lecture 23. Dictionary. Introduction to Hashtables*	*6, №35, 53*
31.	*Lecture 24. Hashtable and Hashing*	*6, №36, 53*
32.	*Lecture 25. Collision in Hashtable. Methods of Resolving*	*6, №37, 53*
33.	*Lecture 26. Implementation of Hashtable as a Dictionary*	*6, №38, 53*
34.	*Programming Assignment 7. Hashtables*	*Task: Implement a hashtable according to a collision resolving method defined by the variant.* *6, №39, 53*
Section 7. Non-Linear Data Structures		
35.	*Lecture 27. Tree. Binary Tree*	*6, №40, 54*
36.	*Lecture 28. AVL-Tree*	*6, №41, 54*
37.	*Lecture 29. Red-Black Tree*	*6, №42, 54*
38.	*Lecture 30. Binary Heap. Priority Queue*	*6, №43, 54*
39.	*Programming Assignment 8. Trees*	*Task: Implement a tree data structure according to the variant.* *6, №44, 54*
40.	*Lecture 31. Introduction to Graphs*	*6, №45, 54*
41.	*Lecture 32. Graph Traversal Algorithms*	*6, №46, 54*
42.	*Lecture 33. Minimum Spanning Tree*	*6, №47, 54*
43.	*Programming Assignment 9. Graphs*	*Task: Implement a graph data structure according to the variant.* *6, №48, 54*
44.	*Lecture 34. Real-life Problems*	*6, №49, 54*
45.	*Midterm Test 2*	*6, №50*

6. Self-study

Credit Module 1. Basics of Algorithmizing

No	*Topic for self-studying*	*Hours*	*Literature*
1.	*Preparation to a lecture 1*	*1*	*2, 4, 5, extra: 3*
2.	*Preparation to a lecture 2*	*1*	*6, extra: 4*
3.	*Preparation to a lecture 3*	*1*	*2, 4, extra: 4*
4.	*Preparation to a lecture 4*	*1*	*2, 3, 4, extra: 1*

5.	Preparation to a programming assignment 1	1,5	1
6.	Preparation to a lecture 5	1	4, 5, extra: 5
7.	Preparation to a lecture 6	1	4, 5, extra: 5
8.	Preparation to a lecture 7	1	4, 5
9.	Preparation to a programming assignment 2	1,5	1
10.	Preparation to a lecture 8	1	5, extra: 4, 5
11.	Preparation to a lecture 9	1	5, extra: 4, 5
12.	Preparation to a lecture 10	1	4, 5, extra: 1, 5
13.	Preparation to a lecture 11	1	4, 5, extra: 1, 5
14.	Preparation to a programming assignment 3	1,5	1
15.	Preparation to a lecture 12	1	2, 4, 5, extra: 5
16.	Preparation to a lecture 13	1	2, 4, 5, extra: 5
17.	Preparation to a programming assignment 4	1,5	1
18.	Preparation to a lecture 14	1	4, 5, extra: 1, 5
19.	Preparation to a lecture 15	1	4, 5, extra: 1, 5
20.	Preparation to a lecture 16	1	2, extra: 1, 3, 4
21.	Preparation to a lecture 17	1	2, extra: 1, 3, 4
22.	Preparation to a midterm test 1	4	2, 3, 4, 5
23.	Preparation to a final test 1	6	2, 3, 4, 5
24.	Algorithm complexity	4	2, 3, 4, extra: 1
25.	Algorithms on arrays	4	4, 5, extra: 5
26.	Algorithms on linked lists	4	4, 5, extra: 5
27.	Real-life problems	6	2, extra: 1, 3, 4

Credit Module 2. Data Structures

1.	Preparation to a programming assignment 5	1,5	1
2.	Preparation to a lecture 1	1	3, extra: 2, 4
3.	Preparation to a lecture 2	1	3, extra: 2, 4
4.	Preparation to a lecture 3	1	3, extra: 4
5.	Preparation to a lecture 4	1	4, 5, extra: 5
6.	Preparation to a lecture 5	1	4, 5, extra: 5
7.	Preparation to a programming assignment 6	1,5	1
8.	Preparation to a lecture 6	1	4, 5, extra: 1, 5

9.	Preparation to a lecture 7	1	4, 5, extra: 1, 5
10.	Preparation to a lecture 8	1	4, 5, extra: 1, 5
11.	Preparation to a lecture 9	1	4, 5, extra: 1, 5
12.	Preparation to a programming assignment 7	1,5	1
13.	Preparation to a lecture 10	1	2, 4, 5, extra: 1, 5
14.	Preparation to a lecture 11	1	2, 4, 5, extra: 1, 5
15.	Preparation to a lecture 12	1	2, 4, 5, extra: 1, 5
16.	Preparation to a lecture 13	1	2, 4, 5, extra: 1, 5
17.	Preparation to a programming assignment 8	1,5	1
18.	Preparation to a lecture 14	1	2, 4, 5, extra: 1, 2, 3, 4, 5
19.	Preparation to a lecture 15	1	2, 4, 5, extra: 1, 2, 3, 4, 5
20.	Preparation to a lecture 16	1	2, 4, 5, extra: 1, 2, 3, 4, 5
21.	Preparation to a programming assignment 9	1,5	1
22.	Preparation to a lecture 17	1	2, 5, extra: 3, 4
23.	Preparation to a midterm test 2	4	2, 3, 4, 5
24.	Preparation to a final test 2	6	2, 3, 4, 5
25.	Linear data structures	4	4, 5, extra: 5
26.	Hashtables	4	4, 5, extra: 1, 5
27.	Non-linear data structures	4	2, 4, 5, extra: 1, 2, 3, 4, 5
28.	Real-life problems	6	extra: 3, 4

Policy and Assessment

7. Course policy

- *Attending lectures is mandatory.*
- *Attending laboratory classes as necessary.*
- *Rules of conduct: activity, taking part in discussions, respect to teacher and groupmates, muting cellphones.*
- *Compliance with the policy of academic integrity.*
- *Rules of defending programming assignment: assignment has to be completed according to the student's variant defined with a generator of pseudo-random numbers (hereafter – randomizer).*
- *Rules of assigning bonus points and penalty points:*

Bonus points are being assigned for:

- *taking part in discussion during lecture classes;*
- *answering other students' questions on lecture material and programming assignment;*
- *creative approach in programming assignment.*

Max bonus points during the semester: 5 points.

Penalty points might be assigned because of:

- *plagiarism (identical flowcharts and/or programming code in works of different students, programming code from Internet resources): -5 points for every attempt;*
- *uploading programming assignment untimely: -0,5 points for every week after the deadline (max penalty points for one programming assignment: -5 points).*

8. Monitoring and grading policy

In the first lecture, the students are being acquainted with the grading policy which is based on Regulations on the System of Learning Outcomes Assessment (https://document.kpi.ua/files/2020_1-273.pdf).

The student's rating in the course consists of points that they receive for programming assignment (R_1), homework (R_2) and a midterm test (R_3).

$$R_S = R_1 + R_2 + R_3 = 60 + 15 + 25 = 100 \text{ points}$$

According to the university regulations on the monitoring of students' academic progress (https://kpi.ua/document_control), there are two assessment weeks, usually during 7th/8th and 14th/15th week of the semester, when students take the Progress and Module tests respectively, to check their progress against the criteria of the course assessment policy.

The students whose overall points at the end of the semester are more or equal to 60 points can:

- *get their final grade according to the rating system;*
- *pass a final test in order to increase the grade.*

Students whose overall points are less than 60 points have to write a final test for 100 points.

The table of compliance between overall points and the final grade:

Points	*Grade*
95-100	Excellent
85-94	Very good
75-84	Good
64-74	Satisfactory
60-64	Fair
Less than 60	Unsatisfactory
Course requirements are not met	Not Graded

9. Additional information about the course

The list of questions for a final test are adduced in Appendix 1.

Syllabus of the course

Is designed by PhD, senior lecturer, Olga Sulema

Adopted by Computer Systems Software Department (protocol № 12, 26 April 2023)

Approved by the Faculty Board of Methodology (protocol № 10, 26 May 2023)