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

Department of Computer Engineering

Data Structures and Algorithms

Syllabus

Requisites of the Course	
Cycle of Higher Education	***First cycle of higher education (Bachelor's degree)***
Field of Study	*12 Information Technologies*
Speciality	*123 Hardware engineering*
Education Program	*Computer Systems and Networks*
Type of Course	*Normative*
Mode of Studies	*full-time*
Year of studies, semester	*1 year (1 semester)*
ECTS workload	*5 credits (ECTS). Time allotment -150 hours,*
Testing and assessment	*Final test*
Course Schedule	*http://rozklad.kpi.ua/*
Language of Instruction	*English*
Course Instructors	Lecturer: *D.Sc., Seniour Scientist, Anatoliy Sergiyenko, mobile +380688123376, email anat.srg@gmail.com, personal web page http://kanyevsky.kpi.ua* Teacher of practical work: *M.Sc., Anastasiia Molchanova, email an.ser.313kpi@gmail.com*
Access to the course	*https://bbb.comsys.kpi.ua/b/ana-3nr-kpr*

Outline of the Course

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

The teaching of data structures and algorithms at the university has the following aspects:

— Academic aspect. The data structures and algorithms is a science that gives fundamental knowledge. The aim is to familiarize the students with the basic concepts and methods of the data representation and performing the algorithms with them, measuring the algorithm complexity.

— Professional aspect. Knowledge of the data structures and algorithms is obligatory for the engineers who design the computer hardware, software, and firmware for it.The purpose of this course is to form the skills in the use of the C language for programming the basic algorithms which are used in most of software and firmware systems, and embedded programming.

— Intellectual and educational aspects. Studying the data structures and algorithms contributes to the development of cognitive skills, which is an essential intellectual factor in the process of creating both hardware and software.

As a result of studying this discipline, the following learning outcomes are achieved.

Competencies:

- ability to abstract thinking, analysis and synthesis;
- ability to learn and acquire modern knowledge;
- ability to apply knowledge in practical situations;

• ability to draw up the obtained work results in the form of presentations, scientific and technical reports.

Knowledge:

• basic concepts and methods of the algorithm theory;

• basic data structures and algorithms;

• basic knowledge about the computer architecture, operation system, program compilation, computational processes in computers;

• methods of the algorithm and program complexity measurement.

Skills:

• programming the basic algorithms in C language with different data structures;

• solving problems of the complexity analysis of the algorithms and programs;

• think systematically and apply creative abilities to form new ideas.

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

Prerequisites, ie., disciplines, the study of which must precede the study of this discipline:

• basic level of English language proficiency not lower than B1.

Postrequisites, ie., disciplines, the study of which must be preceded by the study of this discipline:

• object oriented programming;

• computer architecture;

• computer graphics;

• numerical algorithms and methods;

• program engineering;

• system programming;

• computer networks;

• basics of parallel programming;

• signal processing systems;

• mobile system programming;

• modern programming technologies.

3. Content of the course

Theme 1: C language basics.

Subject and goal of the data structures and algorithms. Algorithm concept. Architecture from programmer's point of view. C language history. Model of the C language. C language objects. C language operations and operators. Type conversion in expressions.

Theme 2: Algorithms and programs.

Constructions for the algorithm design. Block constructions. Control constructions: branch, repeat, jump constructions. Conditional, unconditional, and selection branches. One- multilevel jump constructions. Algorithm representations and operators.Activity diagrams. Algorithm block diagrams. Basics of the structural programming.

Fundamentals of the algorithm evaluation. Time complexity. Spatial complexity. Evaluation of the order of complexity. Iterative algorithms. Algorithm types: polynomial complex and NP-complete

algorithms. Complexity measurement in the program. Algorithm optimizations. The simplest algorithmic techniques. Dynamic programming method.

Subroutines. Functions and procedures. Subroutine context, interface, and implementation. Subroutine implementation in an architecture. Methods of the parameter passing. C program structure. Programming the functions in C. Methods of passing the parameters in the subroutine.

Theme 3: Data structures.

An abstract data type concept. Functions over data structures. Data structure classifications. Static and dynamic data structures. Scalar data structures. Combined data structures. Uniform and nonuniform data structures. Arrays.

Shift array algorithms. One- and multidimensional array declaration and storing. Programming the console display. Algorithms for traversing a 2-D array. Features of working with square matrices.

Theme 4: Algorithms with arrays.

Searching algorithms. Linear search algorithm and its programming. Sorting algorithms and their use. Evaluation of sorting algorithms. Sorting by the direct insertion. Sorting algorithms by the direct selection. Fast sorting algorithm. Merge sorting. Sorting with counting. Hardwired sorting. Algorithms with strings. Functions with strings. Structures. Sets. Unions.

Coursebooks and teaching resources

Base text books

1. B. Kernighan and D. Ritchie, “The C Programming Language”. 272 p.

2. J. McConnell, “Analysis Algorithms. An Active Learning Approach”. 383 p.

Additional text books

1. Drozdek A. Data Structures and Algorithms in C++. 4th Ed. Cengage Learning Pub. 2013. 818 p.

2. McDowell G. L. Cracking the Coding Interview: 189 Programming Questions and Solutions. 6-th ed. Career Cup. 2016.

3. Loudon K. Mastering Algorithms with C. O’Reilly. 1999.

Lesson slides and Laboratory exercise manual are stored in https://kanyevsky.kpi.ua/en/for-students/

Educational content

4. Methodology

The educational content of the discipline consists of lessons and laboratory exercises.

Lessons:

Theme 1: C language basics.

Lesson 1. Introduction.

Subject and goal of the data structures and algorithms. State of the art of the computer engineering. Educational materials. Software to use. Rating system. History of the algorithms and calculations. Algorithm concept. Computational process. Algorithm diagram. Computer architecture and its history. Architecture Hierarchy. Architecture from programmer’s point of view.

Lesson 2. Model of the C language.

Von Neumann model. C language history. C language standards. C language for the PDP-11 computer. Model of the C language. C language objects: comments, symbols, keywords, identifiers, literals. Data and their types. Variables. Local and global objects.

Lesson 3. C language operations

Operations and operators. Arithmetic and logic operations. Assignment operations. Operation priorities. Type conversion in expressions.

Theme 2: Algorithms and programs.

Lesson 4. Constructions for the algorithm design

Block constructions. Control constructions: branch, repeat, jump constructions. Conditional, unconditional, and selection branches. One- multilevel jump constructions.

Lesson 5. Algorithm representations and operators.

Activity diagrams. Algorithm block diagrams. Conditional and unconditional jump operators. Input-output functions. Theory of the program schemas. Basics of the structural programming.

Lesson 6. Fundamentals of the algorithm evaluation.

Time complexity. Spatial complexity. Standard models for estimating the time complexity. Calculating the time complexity in the C programs. Evaluation of the order of complexity. Iterative algorithms. Example of an algorithm and its complexity calculation.

Lesson 7. Fundamentals of the algorithm evaluation (cont.)

Simplified complexity evaluation. Algorithm types: polynomial complex and NP-complete algorithms. Complexity measurement in the program. Example of an algorithm and its complexity calculation.

Lesson 8. Algorithm optimizations.

The simplest algorithmic techniques. Use of a multiplier with the swapping sign. Even number check. Using sign flag. Loop unfolding. Substitution of multiplication to additions. Dynamic programming method. Conditional branch minimization.

Lesson 9. Subroutines.

Functions and procedures. Subroutine context, interface, and implementation. Subroutine call. Subroutine implementation in an architecture. Methods of the parameter passing. Functions in C Language. C program structure. Programming the functions in C. Passing the parameters by the value. Parameter passing by a reference. Parameter passing by a link. Constant parameter. Keyword static. Calling convention in C.

Theme 3: Data structures.

Lesson 10. Data structures.

Data Structures. An abstract data type concept. Functions over data structures. Data structure classifications. Static and dynamic data structures. Scalar data structures. Combined data structures. Uniform and nonuniform data structures. Arrays. One dimensional array, its storing and access.

Lesson 11. Arrays and their shift.

Shift array algorithms. Cyclic shift of the array to one position. Cyclic shift of the array to k positions. Block swapping. Cyclic shift preserving the middle part of the array. Cyclic shift with reversing.

Lesson 12. 2-D arrays

Multidimensional array declaration and storing. 2-D array initilialization. Programming the console display. Windows library and its functions to program the direct access to the console buffer. Algorithms for traversing a 2-D array. Features of working with square matrices.

Theme 4: Algorithms with arrays.

Lesson 13. Searching algorithms.

The searching problem. Linear search algorithm and its programming. Classic linear search algorithm. Linear search with a barrier. Binary search and its two variations. Direct string searching. KMP-search. Rabin-Carp algorithm. Hash search algorithms.

Lesson 14. Sorting algorithms.

Sorting algorithms and their use. Evaluation of sorting algorithms. Sorting by the direct insertion. Algorithm with a linear search from the inserted element. Algorithm with a linear search from the element to be inserted. Algorithm with linear search from the element to be inserted using the barrier. Algorithm with the binary search. Analysis of the direct insertion sorting algorithms.

Lesson 15. Sorting algorithms (cont.)

Sorting algorithms by the direct selection. Direct exchange sorting algorithm. Bubble sorting algorithm. Shaker-sorting algorithm. Shell's algorithm.

Lesson 16. Sorting algorithms (end)

Fast sorting algorithm. Merge sorting. Sorting with counting. Hardwired sorting. Bitonic sorting network. Bubble sorting network. Sorting network of the even-odd merge.

Lesson 17. Algorithms with strings

Strings. String representation in computers. Functions with strings. Structures. Sets. Unions.

Lesson 18.

Module control work.

Laboratory exercises

Laboratory exercise 1.

Branched algorithms.

Laboratory exercise 2.

Cyclic algorithms and dynamic programming.

Laboratory exercise 3.

Linear search algorithms.

Laboratory exercise 4.

Algorithms of traversing the 2-d matrices.

Laboratory exercise 5.

Binary search algorithms.

Laboratory exercise 6.

Sorting algorithms.

5. Self-study

The self-study includes the independent work of students and is as follows:

• preparation for lectures by studying the previous lecture material as well as literary sources on which the material of previous lectures is based (the list of sources and the list of sections is provided together with the lecture material);

• preparation for the laboratory exercise by getting acquainted with the task and guidelines for laboratory work, including the study of theoretical material needed to answer control questions for laboratory work;

Each laboratory exercise is intended to be prepared and executed for two weeks.

Policy and Assessment

6. Course policy

The system of requirements for students:

• the student is obliged to attend lectures and laboratory classes, and actively work on mastering the material taught at them;

• at the lecture the lecturer uses his own presentation material;

• laboratory works are defended in two stages: the first stage is - students perform tasks, draw up an electronic report and send to the teacher; the second stage is - defence of the laboratory work in the laboratory. The control of knowledge in the laboratory exercises is carried out by checking the report on laboratory work, as well as through the implementation of modular tests.

• modular test is written in a lecture using all available materials;

7. Monitoring and grading policy

At the first class the students are acquainted with the grading policy which is based on Regulations on the system of assessment of learning outcomes https://document.kpi.ua/files/2020_1-273.pdf. The student's rating in the course consists of points that he/she receives for defended laboratory exercises (R1)and the modular control work (R2).

Rs=R1+R2=100 points

As a result, the maximum average weight score is equal to:

6 laboratory exercises x 10 points = 60 points

modular control work = 40 points

According to the university regulations on the monitoring of the student's academic progress (https://kpi.ua/document_control) there are two assessment weeks (attestation), 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 condition of the first attestation is to receive at least 10 points (at the time of certification). The condition of the second attestation is to receive at least 30 points (at the time of certification).

The scoring criteria are:

• Execution of laboratory works:

- impeccable work values 9 points;
- there are certain shortcomings in the decorated work - 8-7 points;
- there are some shortcomings in the implementation of the work program - 6-5 points;

Work not performed or not defended - 0 points.

For work that is submitted on time, there is an incentive - 1 point (total no more - 6 points).

• Modular kontrol work is evaluated with 40 points. The control work consists of 16 test questions, as well as a practical task (to design a program) from the list provided in the appendix to the work program.

For each correct answer to the test question 2 points are awarded. The answer to the practical task is evaluated with 8 points according to the following criteria:

- "excellent" - the correct text of the program with comments - 8 - 7 points;
- "good" - the text of the program, in general, correct, but not used optimization techniques - 6-5 points;

- "satisfactory" - there are some fundamental errors in the text of the program - 4 - 3 points;

- "unsatisfactory" - unsatisfactory answer - 0 points.

The students whose finally score the required number of points (≥60) can:

- get their final grade according to the rating score;

- perform a Fail/Pass test in order to increase the grade.

Students can receive up to 6 incentive points for performing creative works from the credit module (compiling abstracts, participating in competitions, in research, etc.).

Students whose final performance score Rs is below 60 points but more than 30 are required to complete a Fail/Pass test. If the grade for the test is lower than the grade, which the student gets for his semester activity, a strict requirement is applied - the student's previous rating is canceled and he receives a grade based on the results of the Fail/Pass test. Students whose score is below 30 are not allowed to take the Fail/ Pass Test.

The Fail/Pass test is estimated at 60 points. The control task of this work consists of three questions. Each of the 3 questions is evaluated with 20 points according to the following criteria:

- "excellent" - a complete answer (at least 90% of the required information), provided appropriate justifications and personal opinion - 20 - 18 points;

- "good" - a fairly complete answer (at least 75% of the required information), performed in accordance with the requirements for the level of "skills", or minor inaccuracies) - 17… 15 points;

- "satisfactory" - incomplete answer (not less than 60% of the required information, which is performed in accordance with the requirements for the "stereotypical" level and some errors) - 14… 12 points;

- "unsatisfactory" - unsatisfactory answer - 0 points.

The final performance score R_S is adopted by university grading system as follows:

Score	*Grade*
100-95	Excellent
94-85	Very good
84-75	Good
74-65	Satisfactory
64-60	Sufficient
Below 60	Fail
Course requirements are not met	Not Graded

8. Additional information about the course

• Questions in the module control work
look like following.

I 1.The hierarchy of computer architectures (A) is as follows: a) micro-A, instruction set A, A of the OS level, assembler level A, A of the problem-oriented languages, b) micro-A, instruction set A, assembler level A, A of OS level, A of the problem-oriented languages, c) micro-A, , assembler level A, instruction set A, A of OS level , A of the problem-oriented languages.

2. In the C language a phrase \n means — a) print control, so that the output continues from the next line, b) the character of jump to the next line, c) returns the division to n?

3. A string ”5” in C language means a) the symbol of the digit 5, b) number 5, c) a string with the character 5.

4. Expression: a++ *b && c * d; is equivalent to the expression — a) (a++)* (b && c)* d; , b) ((a++)* b) && (c * d); , c) a++*(b && (c * d));

5. The result of the operation 15 && 6 is a) 6 , b) 1 , c) 0 .

6. The result of the operation if(12 & 3) then a=1; else if(15 && 6) a=2; else a = 3; is — a) 1 , b) 2 , c) 3.
7. The block is: a) part of the program between the brackets{ and }, b) the selected part of a program, which has the interface for input and output variables, c) the selected part of a program, which variables have local influence.
8. The branch constructions are — a) conditional operator or the construction of choice, b) conditional or unconditional ones, c) conditional, protected, loops.
9. The preconditioned loop has форму – a) do {…} while (condition), b) while (condition) {…} do , c) while (condition){…} .
10. The unconditional loop is: a) loop with a post condition, when the condition is allways — true, b) loop of any kind, for which the continue condition is allways — true, c) such a loop does not exist.
11. In the UML activity diagram there are: a) one vertex of the initial state and one vertex of the end state, b) one vertex of the initial state and one or more vertices of the initial state, c) one or more vertices of the initial state and one or more vertices of the final state.
12. An operator for (;;){...} — implements a loop, – a), if its block contains an break operator, b) if its block contains an operator of the loop breaking, otherwise it is an infinite loop, c) such an operator is not correct.
13. The code: char A[3] = “abc”; printf(“%c”,A[3]); outputs – a) c, b) 0 , c) empty space.
14. An array is the data structure, which represents a) a homogeneous set of elements, which has fixed size and configuration, b) a renumbered set of integers, c) a homogeneous set of elements, which are fixed in size and configuration, and its elements are ordered by numbers.
15. The direct swapping sorting is based on – a) comparison and swapping the pairs of adjacent elements, b) exchange of the leftmost element and the minimum element in the unsorted part of the array, c) exchange of the rightmost element of the sorted part and the minimal element in the unsorted part of the array.
16. Which of the sorting algorithm is quicker – a) direct selection sorting, b) bubble sorting, c) sorting by direct swapping?

II Develop an algorithm and program, which solves the problem. There is a floating point number X . calculate an array X[n], which is X[0] = X; $X[i] = \sum_{k=1}^{i}(-X)^k$ (k+1); Note that the operation number has to be minimized.

Syllabus of the course

Is designed by teacher D.Sc., Senior Scientist, Anatolii Sergiyenko

Adopted by Department of Computing Engineering (protocol № 25, 10 May 2022)

Approved by the Faculty Board of Methodology (protocol № 10 , 09 June 2022)

….