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

Computer Systems Software Department

Fundamentals of Programming. Part 2. Programming Methodologies 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 (2 semester)*
ECTS workload	*5.5 credits (ECTS). Time allotment - 165 hours, including 90 hours of classroom work, and 75 hours of self-study.*
Testing and assessment	*2 semester – Exam*
Course Schedule	*2(3) classes per week by the timetable http://roz.kpi.ua/*
Language of Instruction	*English*
Course Instructors	Lecturer: PhD, Associate Professor, Yuliia Boiarinova, mobile +380671751308, email ub@ua.fm Teacher of practical work: PhD, Associate Professor, Yuliia Boiarinova, mobile +380671751308, email ub@ua.fm Teacher of laboratory work: PhD, Associate Professor, Yuliia Boiarinova, mobile +380671751308, email ub@ua.fm
Access to the course	https://t.me/+zub7kDn0N2g0Zjc6 https://classroom.google.com/c/NTQ2MDgwNTcwMjA3?cjc=nvsujjp

Outline of the Course

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

The discipline "Fundamentals of Programming. Part 2. Programming Methodologies" is aimed at studying the theoretical and methodological foundations of building programs in the programming language C, mastering the means of creating software, gainingpractical skills in software development in solving practical problems. Such theoretical and practical training forms basic skills in programming and is the basis for successful mastering of professional disciplines.

***The purpose** of the discipline is to form students' ability to develop software for solving applied problems of varying complexity in the C programming language.*

*Studying the discipline "Fundamentals of Programming. Part 2. Programming Methodologies" generates **general competence (GC) and professional competence (PC):***
***GC 01** Ability to abstract thinking, analysis and synthesis.*
***GC 06** Ability to search, process and analyze information from various sources.*
***PC 01** Ability to identify, classify and formulate software requirements.*
***PC 02** Ability to participate in software design, including its structure, behavior and functioning processes modeling (formal description).*

PC 03 *Ability to develop software systems architectures, modules and components.*
PC07 *Knowledge of information data models, the ability to create software for data storage, retrieval and processing.*
PC 08 *Ability to apply fundamental and interdisciplinary knowledge to successfully solve software engineering problems.*
PC 10 *Ability to accumulate, process and systematize professional knowledge about software creation and maintenance, and determination of the importance of lifelong learning.*
PC 11 *Ability to implement phases and iterations of the life cycle of the software systems and information technology based on appropriate models and approaches to software development.*
PC 13 *Ability to reasonably select and master software development and maintenance tools.*
PC 14 *Ability to algorithmic and logical thinking.*

Programming Learning Outcomes (PLO) *of the discipline "Fundamentals of Programming. Part 2. Programming Methodologies" under the educational program:*
PLO 01 *To analyze, purposefully search and select the necessary information and reference resources and knowledge to solve professional problems, taking into account modern advances in science and technology.*
PLO03 *To know the software life cycle basic processes, phases and iterations.*
PLO06 *Ability to select and use the appropriate task of software development methodology.*
PLO07 *To know and to apply in practice the fundamental concepts, paradigms and basic principles of the functioning of language, instrumental and computational tools of software engineering.*
PLO13 *To know and apply methods of developing algorithms, designing software and data and knowledge structures.*
PLO15 *To choose programming languages and development technologies to solve the problems of creating and maintaining software.*
PLO18 *To know and be able to apply information technology of processing, storage and transmission of data.*
PLO38 *To be able to apply programming technologies for multimedia and information retrieval systems software development.*

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

To successfully master the discipline "Fundamentals of Programming. Part 2. Programming Methodologies" it is necessary and sufficient to have training at the secondary school level, basic knowledge of working with a PC, if possible, the basics of programming in the amount provided by the high school program.

To the successful study of the discipline "Fundamentals of Programming. Part 2. Programming Methodologies" precedes the study of the disciplines of the school course of mathematics (for the analysis of numerical data, which are described by mathematical laws) and computer science (for means of processing and storing data on a personal computer).

Received during the assimilation of the discipline"Fundamentals of Programming. Part 2. Programming Methodologies" theoretical knowledge and practical skills are necessary for most disciplines of the curriculum and the educational program as a whole, in particular, the study of the disciplines "Programming", "Databases", "Software Engineering Components" and other disciplines of the curriculum of undergraduate training in the specialty 121 Software Engineering, as well as for successful completion of pre-diploma practice, course and diploma projects in the specialty 121 Software Engineering.

3. Content of the course

Topic 1. Elements of structural programming.

Topic 2. Development of programs using pointers

Topic 3. Use functions to work with characters and strings

Topic 4. Files.

Topic 5. Basics of object-oriented programming

4. Coursebooks and teaching resources

Basic

1. The C Programming Language/ Kernighan,Brian; Ritchie, Dennis M. - Englewood Cliffs, NJ:Prentice Hall, 1988 -288p.

2. C: How to Program/ Paul J. Deitel, Harvey M. Deitel,Pearson Prentice Hall, 2010 -998p.

Aditional

3. Sibling rivalry: C and C++/ Stroustrup, Bjarne AT&T Labs. Archived from the original on August 24, 2014.

4. https://www.programiz.com/c-programming

5. https://www.tutorialspoint.com/cprogramming/index.htm

6. https://www.cprogramming.com/

Educational content

5. Methodology

№	Type of study	Description of the lesson
	II semester	
	Topic 1. Elements of structural programming.	
1.	*Lection 1. Elements of structural programming*	*Memory classes. Scope of variables. Global, localvariables Self study: item 6, N1*
2.	*Computer lesson*	*Task: create program with different class of memory Self study:item 6, N2*
3.	*Lection 2. Elements of structural programming*	*Preprocessor. Header files. Creating a library.Self study: item 6, N3*
4.	*Computer lesson*	*Task: creating library Self study: item 6, N4*
5.	*Practical lesson*	*Functions with different number of parameters.Recursions Self study: item 6, N5*
6.	*Lection 3. Structures.*	*Initialization of structures. Access to structure elements. Operations on structure type variables. Self study: item 6, N6*
7.	*Computer lesson*	*Task: create program by variant with structures Self study: item 6, N7*
8.	*Lection 4.Union*	*Union Initialization. Access to union elements. Operations on type variables of unions. Self study: item 6, N8*
9.	*Computer lesson*	*Task: create program by variant with structures Self study: item 6, N9*
10.	*Practical lesson*	*Transfer function by parameter Self study: item 6, N10*
11.	*Lection 5. Bitwise operations*	*Bitwise operations: shift, “and”, “or”, “xor” Self study: item 6, N11*

12.	Computer lesson	Task: create program by variant withstructures (continue) Self study: item 6, N12
Topic 2. Development of programs using pointers		
13.	Lection 6. Pointers	Pointer operations Self study: item 6, N13
14.	Computer lesson	Task: create program by variant with pointers Self study: item 6, N14
15.	Practical lesson	Dynamic memory allocation. Self study: item 6, N15
16.	Lection 7. Pointers	Linking pointers to arrays Self study: item 6, N16
17.	Computer lesson	Task: create program by variant with pointers andarrays Self study: item 6, N17
18.	Lection 8. Dynamic data structures	Dynamic data structures: stacks, queues, lists. Generation and destruction of dynamic objects. Self study: item 6, N18
19.	Computer lesson	Task: create program by variant with dynamic data structures Self study: item 6, N19
20.	Practical lesson	Dynamic memory redistribution. Self study: item 6, N20
21.	Test	Self study: item 6, N21
22.	Lection 9. Dynamic data structures	Dynamic data structures: single-directional and bidirectional lists. Self study: item 6, N22
23.	Computer lesson	Task:create program by variant with dynamic data structures(continue) Self study: item 6, N23
Topic 3. Use functions to work with characters and strings		
24.	Lection 10. Characters in the language of C.	Basic standard character processing functions. ASCII character codes. Self study: item 6, N24
25.	Computer lesson	Task:create program by variant with charters data Self study: item 6, N25
26.	Practical lesson	Working with characters and strings Self study: item 6, N26
27.	Lection 11. Strings	Features of the structure of strings in C. Declaration ofarrays of strings. Operations by strings Self study: item 6, N27
28.	Computer lesson	Task: create program by variant with string Self study: item 6, N28
Topic 4. Files.		
29.	Lection 12. Files.	Files. Logical and physical file. File structure Self study: item 6, N29
30.	Computer lesson	Task:create program by variant with file Self study: item 6, N30
31.	Practical lesson	Using serial access files Self study: item 6, N31

32.	Lection 13. Files.	General information about streaming I / O libraries. Functions for sharing streams. Serial access files. Creating, reading, writing data Self study: item 6, N32
33.	Computer lesson	Task: create program by variant with file(continue) Self study: item 6, N33
34.	Lection 14. Files.	Direct (random) access files. Creating, reading, writingdata. Self study: item 6, N34
35.	Computer lesson	Task: create program by variant with file(continue) Self study: item 6, N35
36.	Practical lesson	Using files with direct access Self study: item 6, N36
37.	Lection 15. Files.	Text files. Creating, reading, writing data. Self study: item 6, N37
38.	Computer lesson	Task: create program by variant with file(continue) Self study: item 6, N38
39.	Lection 16. Files.	Binary files. Creating, reading, writing data. Self study: item 6, N39
40.	Computer lesson	Task: create program by variant with file(continue) Self study: item 6, N40
41.	Practical lesson	Using text files Self study: item 6, N41
42.	Lection 17. Modular programming.	Modular programming. Dividing the program intoseparate modules Self study: item 6, N42
43.	Computer lesson	Task: create program with modular programming Self study: item 6, N43
Topic 5. Basics of object-oriented programming		
44.	Lection 18. Basics of object-oriented programming	Basic concepts of OOP. Properties and their classification. Encapsulation and inheritance, rules of inheritance. Class as an abstract type data. Self study: item 6, N44
45.	Computer lesson	Task: create program with elements of OOP Self study: item 6, N45
46.	Practical lesson	Generation and destruction of dynamic softwareobjects. Self study: item 6, N46
47.	Exam	Self study: item 6, N47

6. Self-study

The discipline «Fundamentals of Programming. Part 1. Basic structures» is based on independent preparations for classroom on theoreticaland practical topics.

1.	Preparation for the lecture 19	1	1,p.73-81
2.	Preparing for a computer lesson 19	1.5	Create a program
3.	Preparation for the lecture 20	1	2,p.151-152
4.	Preparing for a computer lesson 20	1.5	Create a program
5.	Preparing for a practical lesson 10	1.5	2,p.167-173

6.	Preparation for the lecture 21	1	1,p.114-116
7.	Preparing for a computer lesson 21	1.5	Create a program
8.	Preparation for the lecture 22	1	1,p.131-133
9.	Preparing for a computer lesson 22	1.5	Create a program
10.	Preparing for a practical lesson 11	1.5	1,p.192-194
11.	Preparation for the lecture 23	1	1,175-177
12.	Preparing for a computer lesson 23	1.5	Create a program
13.	Preparation for the lecture 24	1	2,p.254-270
14.	Preparing for a computer lesson 24	1.5	Create a program
15.	Preparing for a practical lesson 12	1.5	2,p.456-457
16.	Preparation for the lecture 25	1	2,p.275-285
17.	Preparing for a computer lesson 25	1.5	Create a program
18.	Preparation for the lecture 26	1	1,p.160-164,2,p.454-477
19.	Preparing for a computer lesson 26	1.5	Create a program
20.	Preparing for a practical lesson 13	1.5	2,p.309-311
21.	Preparing for test	4	Lection 19-26
22.	Preparation for the lecture 27	1	2,p.478-494
23.	Preparing for a computer lesson 27	1.5	Create a program
24.	Preparation for the lecture 28	1	2,p.310-316
25.	Preparing for a computer lesson 28	1.5	Create a program
26.	Preparing for a practical lesson 14	1.5	2,p.421-425
27.	Preparation for the lecture 29	1	2,p.322-355
28.	Preparing for a computer lesson 29	1.5	Create a program
29.	Preparation for the lecture 30	1	2,p.417-419
30.	Preparing for a computer lesson 30	1.5	Create a program
31.	Preparing for a practical lesson 15	1.5	2,p.430-432
32.	Preparation for the lecture 31	1	2,p.420-429
33.	Preparing for a computer lesson 31	1.5	Create a program
34.	Preparation for the lecture 32	1	2,p.420-436
35.	Preparing for a computer lesson 32	1.5	Create a program
36.	Preparing for a practical lesson 16	1.5	1,p.148-152
37.	Preparation for the lecture 33	1	1,p.142-144
38.	Preparing for a computer lesson 33	1.5	Create a program
39.	Preparation for the lecture 34	1	1-22,p. 437-453
40.	Preparing for a computer lesson 34	1.5	Create a program
41.	Preparing for a practical lesson 17	1.5	1,p.153-154
42.	Preparation for the lecture 35	1	1,p.192-194

43.	Preparing for a computer lesson 35	1.5	Create a program
44.	Preparation for the lecture 36	1	2,p. 528-600
45.	Preparing for a computer lesson 36	1.5	Create a program
46.	Preparing for a practical lesson 18	1.5	2,p.454-494
47.	Preparing for exam	6	Lection 1-18

Policy and Assessment

7. Course policy

- *Attendance at lectures is mandatory.*
- *Attendance at computer lesson can be sporadic and if necessary to protect the work of the computerlesson.*
- *Rules of conduct in the classroom: activity, respect for those present, turning off the phones.*
- *Adherence to the policy of academic integrity.*
- *Rules for the protection of computer work: work must be done according to the option of the student,which is determined by his number in the list of the group*

8. Monitoring and grading policy

During the semester, students complete 5 computer works. Maximum number of points for each computer workshop: 8 points.

Points are awarded for:

- quality of laboratory work (computer work): 0-3 points;

- answer during the defense of laboratory work (computer workshop): 0-4 points;

- timely submission of work to the defense: 0-1 points.

-

Performance evaluation criteria:

3 points - the work is done qualitatively, in full;

2 points - the work is done qualitatively, in full, but has shortcomings;

1 point - the work is done in full, but contains minor errors;

0 points - the work is not performed in full, or contains significant errors.

Response evaluation criteria:

4 points - the answer is complete, well-argued;

3-2 points - in general the answer is correct, but has shortcomings or minor errors;

1 point - there are significant errors in the answer;

0 points - no answer or the answer is incorrect.

Criteria for assessing the timeliness of submission of work to the defense:

1 points - the work is submitted for defense no later than the specified period;

0 points - the work is submitted for defense later than the specified deadline.

Maximum number of points for performing and defending computer workshops:

R1=8 points × 5 lab. works = 40 points

The test consists of 1 practical task. The answer is evaluated by 8 points.

R2=10

10-9 points - the answer is correct, complete, well-argued;

7-8- points - the answer is correct, detailed, but not very well reasoned;
5-6 points - in general the answer is correct, but has shortcomings;
3-4 points - there are minor errors in the answer;
1-2 points - there are significant errors in the answer;
0 points - no answer or the answer is incorrect.

The task for the exam consists of 2 questions - 1 theoretical and 2 practical. The answer to theoretical question is evaluated by 10 points, and the answer to the practical question is evaluated by 20 points.

Criteria for evaluating each theoretical question of the test:
9-10 points - the answer is correct, complete, well-argued;
7-8 points - the answer is correct, detailed, but not very well reasoned;
5-6 points - in general the answer is correct, but has shortcomings;
3-4 points - there are minor errors in the answer;
1-2 points - there are significant errors in the answer;
0 points - no answer or the answer is incorrect.

Criteria for evaluating the practical question of the test:
18-20 points - the answer is correct, the calculations are performed in full;
14-17 points - the answer is correct, but not very well supported by calculations;
9-13 points - in general the answer is correct, but has shortcomings;
5-8 points - there are minor errors in the answer;
1-4 points - there are significant errors in the answer;
0 points - no answer or the answer is incorrect.

Maximum number of points for exam:
R3=10 points × 1 theoretical question + 20 points × 2 practical questions = 50 points.
The rating scale for the discipline is equal to:

Rs = R1 + R2 + R3 = 40 point + 10 point + 50point = 100 points.
Calendar control: conducted twice a semester as a monitoring of the current state of compliance with the requirements of the syllabus.

At the first attestation (8th week) the student receives "credited" if his current rating is not less than 15 points (50% of the maximum number of points that a student can receive before the first attestation).
At the second attestation (14th week) the student receives "credited" if his current rating is not less than 25 points (50% of the maximum number of points that a student can receive before the second attestation).

Semester control: exam
Conditions of admission to semester control: with a semester rating at least 30 points and enrollment in all computer works.

The final performance score or the results of the Fail/ Pass Exam are 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*

9. Additional information about the course

It is possible to enroll in certificates of distance or online courses on the relevant topic - programming in C.

Syllabus of the course

Is designed by teacher PhD, Associate Professor, Yuliia Boiarinova
Adopted by Computer Systems Software Department (protocol № 12 from 26.04.23)

Approved by the Faculty Board of Methodology (protocol № 10 from 26.05.23)