Національний технічний університет «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ імені ІГОРЯ СІКОРСЬКОГО»

Емблема кафедри (за наявності)

Department of computer engineering

DEVELOPMENT OF SYSTEM PROGRAMS

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 Computer Engineering*
Education Program	*Computer Systems and Networks*
Type of Course	*Selective*
Mode of Studies	*full-time*
Year of studies, semester	*3 year (5 semester)*
ECTS workload	*1 credits (ECTS). Time allotment - 120 hours, including 54 hours of classroom work, and 66 hours of self-study*
Testing and assessment	*5 semester – Final test*
Course Schedule	*//rozklad/kpi.ua*
Language of Instruction	*English*
Course Instructors	PhD, Associate Professor, Valerii Pavlov, pavlovvg@ukr.net
Access to the course	*https://campus.kpi.ua/tutor/index.php?mode=mob&show&irid=218710*

Outline of the Course

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

*Academic discipline "**Development of System Programs**" belongs to the selective educational components of the education program, namely, to the cycle "Selective Educational components from the Faculty/Department-Catalog". It has the code **12** in the list of components of the educational program.*

***Reason and motivations for studying**: the need to study the academic discipline "**Development of System Programs**" is to consolidate, deepen and generalize theoretical knowledge and practical skills, which students receive during the study of the discipline «**System Programming**», understanding of the principles of software control of the processor itself.*

***The goal of the "Development of System Programs" course is**: Obtaining skills in the development of system programs in accordance with the requirements of the specification or technical requirement and the preparation of a set of documents for programs for their customer or users. Study of the basic requirements for the basic documents of system programs and obtaining skills in the design of such documents on examples of the implementation of system programs.*

***The purpose of the discipline** is the formation of a number of competencies among students, namely:*

***CAPACITY**:*

- *analyze the processes that are carried out during the translation and compilation;*
- *develop system software with the help of modern software development tools;*
- *debug system programs.*

1.2. The main tasks of the academic discipline.

After mastering the academic discipline, students must demonstrate such learning outcomes:

***KNOWLEDGE**:*

- *interaction of system programs during their execution;*
- *methods for describing languages and programming languages in particular;*
- *models and implementations of context-free (CF) and automatic grammars;*
- *features and stages of development of system programs;*
- *methods for solving problems of lexical, syntactic and semantic processing of input codes of programs.*

***ABILITY**:*

- *use the necessary tools to solve problems;*
- *develop or configure a syntax analysis algorithm,*
- *describe the internal form of the presentation of programs in the form of graphs;*
- *describe and process specific computer language operators.*

***SKILLS**:*

- *implementation experiments, data collection and modeling in computer systems;*
- *communicate verbally and in writing on professional matters in Ukrainian and at least one of the official EU languages (English, Germany, Italian, French, Spanish etc.);*
- *adapt to new situations, to justify, make and implement decisions within the competence.*

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

*Interdisciplinary Connections: To successfully study the "**Development of System Programs**" students must master the material and have certain knowledge, skills and abilities in such disciplines:*

- *GM9 - «Programming»,*
- *PM1 - «Computer Logic»,*
- *PM6 - «Computer Architecture»,*
- *PM13 - «Algorithms and Methods of Computation»,*
- *PM7- «System Programming».*

*Knowledge and skills acquired during the study of the discipline «**Development of System Programs**», can be used in the future when mastering the following courses:*

- *PM6 – «Computer Architecture»,*
- *PM9 - «System software»,*
- *PM17 – «Course work on Computer Architecture»,*
- *PM18 – «Course work on System software»,*

as well as during diploma design.

3. Content of the course

Section 1. Methods of processing and transformations in translation programs

Topic 1.1. System software structure

- *Subject and tasks of the course.*
- *Composition and purpose of system software (SysS).*
- *The place and role of translators in SysS.*

Topic 1.2. Principles of construction and types of compilers.

- *Types of translators.*
- *Compilers and interpreters.*
- *Compiler classification.*
- *The main components of compilers and the compilation sequence.*
- *Principles of construction of lexical tables and trees of grammatical analysis.*

Topic 1.3. Automation of compiler creation

- *Description of compilers using T-diagrams.*
- *Front-end and Back-end interfaces.*
- *Promotion method, cross-compilation, virtual machines, JIT compilers.*
- *YACC system.*

Section 2. Methods for formalizing the description of programming languages

Topic 2.1. Methods of describing natural and artificial languages. Metalanguages.

- *Language concept.*
- *Language description tools.*
- *Natural and artificial languages.*
- *The concept of alphabet and grammar of the language.*
- *Metalanguages BNF and EBNF.*

Topic 2.2. Generating grammars.

- *The concept of generating grammars according to N. Chomsky.*
- *Sensitive output form.*
- *The main components of the description of grammars.*
- *Examples of describing different languages using generating grammars.*

Topic 2.3. Classification of formal grammars. Construction of syntactic trees.

- *Classification of grammars according to N. Chomsky.*
- *Types of recognizers.*
- *Rules for constructing syntactic trees.*

Section 3. Transformations context-free grammars.

Topic 3.1. Grammar transformation methods.

- *The proper context-free (CF) grammar form .*
- *Removing ε-rules and chain rules algorithms.*
- *Removing of non-generating and unattainable non-terminals algorithms.*
- *Mandatory sequence of obtaining the proper form of CF-grammars.*
- *Transforming KV grammarians to the Chomsky Normal Form.*
- *Greibach Normal Form.*
- *Ambiguity of KV – grammar and concepts of guiding symbols.*
- *S- grammars.*
- *Q- grammars.*
- *LL(1)- grammars.*
- *Transform CF - grammar to type LL (1).*
- *Algorithm for eliminating left recursion.*
- *Left factorization.*

Section 4. Classes of syntactic analyzers.

Topic 4.1. Recognizer for regular grammars.

- *Construction of the recognizer based on the Finite Automata (FA).*
- *Types of FA description.*
- *Deterministic (DFA) and non-deterministic (NFA) Finite Automatas*
- *Algorithm for transforming NFA to DFA.*
- *Regular expressions and regular sets.*

- *Algorithm for minimization of Finite Automata.*

Topic 4.2. Recognizer for CF-grammars.

- *Definition of Pushdown Automata.*
- *Deterministic and non-deterministic Pushdown Automatas.*
- *Types of preceding grammars.*

4. Bibliography and resources

4.1. Basic:

1. Abdulaziz Ghuloum. An Incremental Approach to Compiler Construction, Conference: Scheme and Functional Programming Workshop , 2006. URL: http://scheme2006.cs.uchicago.edu/11-ghuloum.pdf.

3. Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman. Compilers. Principles, Techniques, and Tools,– ADDISON-WESLEY, 1986, - 796 p.

4. Alfred V. Aho, Jeffrey D. Ullman. The theory of parsing, translation and compiling. Two-volume series. – Prentice-Hall, 1972, - 2051 p.

5. Wirth Niklaus. Compiler Construction. – ADDISON-WESLEY, 2000, - 132 p.

4.2 Additional:

1. Intel Corporation. Intel 64 and IA-32 Architectures Software Developer's Manual. Volumes 1–3 [Electronic resource]:– 2014. http://www.intel.com/content/www/us/en/processors/architectures-software-developer-manuals.html.

2. Jeff Duntemann. Assembly Language Step-by-Step. – Indianapolis: Wiley, 3rd Edition 2009, - 646 p.

3. Charles W. Kann. Introduction to MIPS Assembly Language Programming. – Gettysburg College, 2015, - 179 p.

4. Dennis Yurichev. Reverse Engineering for Beginners. dennis@yurichev.com:, 2013, - 1083 p.

Educational content

5. Methodology

The study of the discipline consists of four stages, which are shown in Table 5.1.

Table 5.1

Section and topic titles	Number of hours				
	Total	including			
		Lectures	Practical (Seminars)	Laboratory (Computer Workshop)	Self-study
Section 1. Methods of processing and transformations in translation programs					
Topic 1.1. System software structure.	4	2	–	–	2
Topic 1.2. Principles of construction and types of compilers.	8	2	–	2	4
Topic 1.3. Automation of compiler creation.	4	2	–	–	2
Section 2. Methods for formalizing the description of programming languages					
Topic 2.1. Methods of describing natural and artificial languages. Metalanguages.	8	2	–	2	4
Topic 2.2. Generating grammars.	8	4	–	–	4
Topic 2.3. Classification of formal grammars. Construction of syntactic trees.	8	2	–	2	4
Verification work 1	2	–	–	–	2

Total by sections 1-2	*42*	*14*	*–*	*8*	*22*
Section 3. Transformations context-free (CF) grammars.					
Topic 3.1. Grammar transformation methods.	*28*	*10*	*–*	*4*	*14*
Section 4. Classes of syntactic analyzers.					
Topic 4.1. Recognizer for regular grammars.	*20*	*6*	*–*	*4*	*10*
Topic 4.2. Recognizer for CF-grammars.	*20*	*6*	*–*	*4*	*10*
Verification work 2	*2*	*–*	*–*		*2*
Total by sections 3-4	*70*	*22*	*–*	*12*	*36*
Preparation for the final test	*8*	*–*	*–*	*–*	*8*
Total hours	*120*	*36*	*–*	*18*	*66*

Laboratory works

The main tasks of the cycle of laboratory classes are to provide students with the necessary practical skills in developing and analyzing algorithms for system programs, development and debugging of system programs using high-level languages and Assembler.

N in/o	***Name of laboratory work (computer workshop)***	***Number of classroom hours***
1	*Development and processing of the basic structure of the compiler.*	*2*
2	*Unary and binary operations.*	*2*
3	*Processing variables.*	*2*
4	*Nested constructions.*	*4*
5	*Ternary operator, branching operators.*	*4*
6	*Cycles.*	*4*
	Total:	*18*

1. **Self-study**

N in/o	***Names of topics and questions that are submitted for independent study***	***Number of hours of self-study***
1	*Performing tasks on the topic of each lecture (look at Section 4) – 1 hour on 1 lection hour*	*36*
2	*Preparation for laboratory classes (look at Section 5) – 1 hour on 1 laboratory work hour*	*18*
3	*Preparation for verification works (look at Section 9) – 2 hours on every MVR.*	*4*
4	*Preparation for the final test*	*8*
	Total:	*66*

Policy and Assessment

2. Course policy

When counting and evaluating laboratory work, the following factors are taken into account:

- *Completeness of the task on laboratory work on the individual variant;*
- *Timeliness of laboratory work according to the schedule;*
- *Autonomy of laboratory work and no indications of plagiarism;*
- *Answers to questions on the content of laboratory work during its protection.*

When evaluating control works, next consideration is taken into account:

- *Correctness and completeness of tasks;*
- *Number of completed tasks in conditions of limited time;*
- *Autonomy of tasks and no indications of plagiarism;*
- *Number of attempts to run controls that precede the one that is estimated.*
- *To prepare for the tests students receive a list of theoretical questions and the content of typical exercises that will be in the tasks on the test.*
- *At the first and second attestation, the number of laboratory works and tests that were passed at the time of the attestation is taken into account.*

3. Monitoring and grading policy

*The system of assessing the grading policy in the discipline "Development of System Programs" is based on the "Regulations on the system of assessment of learning outcomes in the «Igor Sikorsky Kyiv Polytechnic Institute» (*https://document.kpi.ua/files/2020_1-273.pdf), namely the Rating System of Assessment (**RSA**) of the first type (**RSA-1**).

RSA in the discipline, semester control of which is provided in the form of a test, is developed according to the type of RSA-1 and includes evaluation of current control measures in the discipline during the semestery.

*The rating score of the applicant is formed as the sum of points that are obtained as a result of current control activities (laboratory works (**R_L**) and tests (**R_T**)), incentive (**R_I**) and penalty (**R_P**) points:*

$$R_S = R_L + R_T + R_I + R_P,$$

*where R_L for 6 laboratory works is **6 X 10 = 60** points,*

*R_T for 2 tests is **2 X 20 = 40** points.*

In this way, the maximum basic amount of points for the semester is 60 + 40 = 100 points.

*Behind the main assessment scale there are incentive and penalty points, which are taken into account in the total amount of points, but are not included in the main **RSA** scale.*

Incentive points take into account the answers to questions and the performance of tasks in lecture classes, the quality of the notes.

Penalty points are provided for late performance of laboratory work, that is, with a delay relative to the schedule.

Assessment of learning outcomes is carried out on a 100-point scale with further conversion of grades into a university scale in accordance with the table:

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

Syllabus of the course

Is designed by teacher PhD, Associate Professor, Valerii Pavlov

Adopted by Department of Computing Technics (protocol #10 , May 25, 2022)

Approved by the Faculty Board of Methodology (protocol #10 , June 09, 2022)

…