Syllabus
Course Program

Compiler Design Theory

Specialty
123 – Computer engineering

Institute
Institute of Computer Science and Information Technology

Educational program
Modern programming, mobile devices, and computer games

Department
Computer Engineering and Programming (326)

Level of education
Bachelor's level
Master's level
Ph.D.

Course type
Professional, Elective

Semester
1

Language of instruction
English, Ukrainian

Lecturers and course developers

Gavrylenko Svitlana

Svitlana.Gavrylenko@khpi.edu.ua
Doctor of Technical Sciences, Professor, Professor at the Computer Engineering and Programming Department NTU "KhPI".
Has over 250 publications. Teaches courses: "Machine Learning," " Compiler Design Theory," "Formal Languages, Grammars, and Automata."
For more information about the professor, please visit the department's website.
https://web.kpi.kharkov.ua/cep/2021/09/03/gavrylenko-svitlana-yuriyivna/

General information

Summary

The course "Compiler Design Theory " focuses on the syntactic-oriented processing of sentences in the input language. Within the course, various phases of compiler are discussed: lexical, syntax, and semantic analysis, error handling, symbol table generation, code optimization and generation target code. Understanding these stages helps in comprehending the tasks and algorithms of the functional blocks in modern compilers, regardless of the technology and tools used to create them.

Course objectives and goals

The purpose of studying this discipline is to acquaint students with the theoretical foundations of Compiler Design that can translate high-level programming languages into executable machine code or interpreted language and to develop relevant competencies in this field.

Format of classes

The course includes lectures, laboratory and practical assignments, a course project, independent work, and consultations. The final assessment is conducted through an examination.

Competencies

ZK2. Ability for abstract thinking, analysis, and synthesis.

SK5. Ability to design architecture and create system and application software for computer systems and networks.

Learning outcomes

RN1. Apply general approaches of knowledge, mathematical methods, natural and engineering sciences to solve complex problems in computer engineering.
RN3. Build and investigate models of computer systems and networks, evaluate their adequacy, and determine the limits of their applicability.

Student workload

The total volume of the course is 150 hours (5 ECTS credits): lectures - 32 hours, practical classes - 16 hours, laboratory classes - 32 hours, self-study - 70 hours.

Course prerequisites

To successfully complete the course, it is necessary to have knowledge of formal languages, grammars, and automata, as well as proficiency in at least one programming language.

Features of the course, teaching and learning methods, and technologies

Lectures are conducted interactively using multimedia technologies. Practical sessions employ a project-based learning approach and incorporate gamification methods.

Program of the course

Topics of the lectures

Topic 1. Introduction. The typical phases of a compiler.
Basic concepts. Definition of the subject " Compiler Design Theory". Interpreter Vs Compiler. Architecture of a Compiler. Phases of a Compiler. Pass of a Compiler: Front-end, Middle End , Back-end. Modern concepts of a compiler construction.
Topic 2. Lexical analysis.
Lexical Scanner. Token, Pattern, Lexeme. Advantages and Dias advantages of Lexical analysis. Finite Automata. Transition table and State Diagram.
Topic 3. Formal languages and grammars.
Definition of formal languages and grammars. Types of formal languages and grammars. Context-free grammar (CFG). Parse tree. Building production rules. Reduced grammars. Non-generating and unreachable symbols.
Topic 4. Pushdown automata (PDA).
Pushdown automata (PDA). Automaton specification. Automaton configuration. Regular Expressions.
Topic 5. Parsers (syntax analyzers). Strategies for syntax parsing. Top-down parsing strategy.
LL(k) grammars. SLL(1) grammars. WSLL(1) grammars. Set FIRST(μ), FOLLOW(μ), and SELECT(μ). Top-down parsing algorithms for SLL(1), WSLL(1), LL(1) grammars.
Topic 6. Bottom-up parsing strategy. LR(k) grammars
LR(k) grammars. Grammar symbols and grammar entrance. Set F_FIRST(μ), F_FOLLOW(μ). Action table and Goto table. Bottom-up parsing algorithm for LR(k) grammars.
Topic 7. Bottom-up parsing strategy. Simply Operator Precedence Grammars.
Precedence relations between two adjacent symbols. Construction of the set of extreme left and extreme right symbols. Precedence matrix algorithm. Bottom-up parsing algorithm for Simply Operator. Precedence grammar.
Topic 8. Bottom-up parsing strategy. Operator Precedence Grammar.
Construction of the set of extreme left and extreme right terminal symbols. Precedence matrix algorithm. Bottom-up parsing algorithm for Operator Precedence grammar.
Topic 9. Parse Tree.
Parse Tree - Tree of Operations Transformation.
Topic 10. Symbol Table.
Symbol Table in Compiler. Items stored in Symbol table. Binary Search Tree. Linked List. Hash Table.
Topic 11. Code generation and optimization.

Intermediate representation of program code. Reverse Polish notation. Triples. Quadruples. Transformation of typical operation nodes in the operation tree into a list of triples. Transformation of typical operation nodes in the operation tree into assembly code. Optimization of object code using folding and dead code elimination methods.

Topic 12. Compilation of a translator.

Building a translator. Linkers and loaders. Preprocessors. Assemblers. Bootloaders and link editors. Memory allocation. Procedure memory display.

Topic 13. Systems for designing translators.

Building a compiler. Linkers and loaders. Preprocessors. Assemblers. Bootloaders and link editors. Memory allocation. Procedure memory display.

Topic 14. Tools for building compilers.

Parser generator. Scanner generator. Syntax-directed translation mechanisms. Automatic code generators. Data flow analysis mechanisms. Compiler construction toolkits.

Topics of the workshops

Topic 1. Lexical Scanner Construction.

Deterministic finite Automata. Main transition table, transition diagram. Construction of lexical analyzer.

Topic 2. Formal languages and grammars.

Syntax parsing. Left and right derivation. Ambiguous and equivalent grammars.

Topic 3. Grammar Construction. Reduced Grammars.

Grammar construction. Procedure for creating production rules.

Topic 4. Reduced Grammars.

Procedure for definition Non-generating and Unreachable symbols.

Topic 5. Regular Expressions.

Construction of regular expressions. Quantification (searching for sequences). Modifiers. Conditional search.

Topic 6. Pushdown Automata (PDA).

Automaton specification. Automaton configuration.

Topic 7. Top-Down Parsing.

Construction of a deterministic top-down automaton for a SLL(1) grammars.

Topic 8. FIRST(μ), FOLLOW(μ) and SELECT Set.

Construction of FIRST(μ), FOLLOW(μ) and SELECT set.

Topic 9 WSLL(1) and LL(k) Grammars.

Construction of a deterministic top-down parser for weakly separated WSLL(1) and LL(k) grammars.

Topic 10. Bottom-up Parser for LR(0) grammar.

Construction of a deterministic top-down Parser for LR(0) grammar.

Topic 11. Bottom-up Parser for LR(1) grammar.

Construction of a deterministic Bottom-up Parser for LR(1) grammar.

Topic 12. Simply Operator Precedence Grammars and Operator Precedence Grammars.

Construction of a deterministic top-down Parser for Simply Operator Precedence Grammars and Operator Precedence Grammars.

Topic 13. Parse Tree.

Parse tree. Conversion of a parse tree into an operation tree.

Topic 14. Symbol Table Construction.

Construction of symbol table using hashing with chaining method and with hashing and rehashing method.

Topic 15. Intermediate Code Generation.

Intermediate representation of program code. Reverse Polish notation. Triples. Quadruples. Transformation of typical operation tree nodes into a list of triples. Transformation of typical operation tree nodes into assembly code.

Topic 16. Code Optimization.

Optimization of object code using folding method and redundant operation elimination method.

Topics of the laboratory classes

Topic 1. Construction of Lexical Scanner Model.

Development of software that simulates the operation of a lexical scanner.

Topic 2. Grammar Rule Construction.
Development of software for checking production rules for the presence of Non-generating and Unreachable symbols and performing syntax parsing of input strings.
Topic 3. Construction of FIRST(μ), FOLLOW(μ) and SELECT sets.
Development of software for constructing FIRST(μ), FOLLOW(μ), and SELECT(μ) sets. Checking grammar for LL(k) property.
Topic 4. Syntax Parsing using Bottom-up Strategy.
Development of software for generating transition functions of a deterministic top-down parser. Testing the operation of the parser using example of input string.

Self-study

The subject includes a course project (course assignment, coursework project) on building a bottom-up parser. Students are also recommended to use additional materials such as videos, articles, and textbooks for self-study and analysis.

Course materials and recommended reading

Main literature:
1. Gavrylenko S.Yu. Formalni movy, hramatyky ta avtomaty. Navch. posibnyk// S.Yu. Gavrylenko // Kharkiv: NTU "KhPI", 2021. - 133 p.
2. Gavrylenko S.Yu., Klymenko A.M., Liubchenko N.Yu. Teoriia tsyfrovykh avtomativ ta formalnykh mov. Rekomendovano MON Ukrainy - Kharkiv, NTU "KhPI", 2010. - 176 p.
3. Gavrylenko S.Yu. Proektuvannia systemnykh program. Laboratornyi praktykum. Kharkiv: NTU "KhPI", 2013, - 76 p.
4. Gavrylenko S.Yu., Filonenko A.M., Glavchev M.I. Osnovy pobudovi kompiliatoriv. - Kharkiv: Kursor, 2005. - 107 p.
5. Gavrylenko S.Yu. Metodychni vkazivky do vykonannia kursovoho proektu z kursu "Proektuvannia systemnykh program". Kharkiv, NTU "KhPI", 2009, - 16 p.
6. Spektorskyi I.Ya., Statkevych V.M. Formalni movy ta avtomaty. - Kyiv: KPI im. Ihoria Sikorskoho, 2019. - 167 p.
7. Zakhiariia L.M., Zaiats M.M. Formalni mova, hramatyky ta avtomaty. Lvivska politekhnika, 2016. - 196p.

Additional literature:
8. Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman. Compilers: Principles, Techniques, & Tools. Pearson Education, Inc., 2007. - 1009 p.
9. Proektuvannia transliatoriv [Text]: navch. posibnyk dlia studentiv spetsialnosti "Prohramne zabezpechennia avtomatyzovanykh system" / V. Kh. Vlasiuk; Vinnitskyi derzh. tekhnichnyi un-t. - Vinnytsia: VDTU, 2002. - 108 p.
10. Alfred V. Aho, Jeffrey D. Ullman. The Theory of Parsing, Translation and Compiling. Volume 1. Parsing. Prentice-Hall, Inc., Englewood Cliffs, N.J., 1972. - 582 p.

Assessment and grading

Criteria for assessment of student performance, and the final score structure

The final grade is calculated based on the evaluation results from the exam (40%) and continuous assessment (60%). Exam: written assignment (2 theory questions + problem solving) and an oral presentation. Continuous assessment: 4 online tests (20%), practical sessions (40%).

Grading scale

Total points	National	ECTS
90–100	Excellent	A
82–89	Good	B
75–81	Good	C
64–74	Satisfactory	D
60–63	Satisfactory	E
35–59	Unsatisfactory (requires additional learning)	FX
1–34	Unsatisfactory (requires repetition of the course)	F

Norms of academic integrity and course policy

The student must adhere to the Code of Ethics of Academic Relations and Integrity of NTU "KhPI": to demonstrate discipline, good manners, kindness, honesty, and responsibility. Conflict situations should be openly discussed in academic groups with a lecturer, and if it is impossible to resolve the conflict, they should be brought to the attention of the Institute's management.
Regulatory and legal documents related to the implementation of the principles of academic integrity at NTU "KhPI" are available on the website: http://blogs.kpi.kharkov.ua/v2/nv/akademichna-dobrochesnist/

Approval

Approved by	Date, signature	Head of the department Oleksandr ZAKOVOROTNYI
	Date, signature	Guarantor of the educational program Svitlana GAVRYLENKO