**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА**

**ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК ТА КІБЕРНЕТИКИ**

**Кафедра інформайційних систем**

**«ЗАТВЕРДЖУЮ»**

Заступник декана
з навчальної роботи
_____ Кашпур О.Ф.
«26» 03 2018 року

# РОБОЧА ПРОГРАМА НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

# SOFTWARE FOUNDATIONS / ТЕОРЕТИЧНІ ОСНОВИ ПРОГРАМУВАННЯ

**для здобувачів освітньо-наукового рівня «доктор філософії»**

| | |
|---|---|
| галузь знань | **12 Інформаційні технології** |
| спеціальність | **121 Інженерія програмного забезпечення** |
| освітній рівень | **третій (освітньо-науковий)** |
| освітня програма | **Інженерія програмного забезпечення** |
| вид дисципліни | **вибіркова** |

| | |
|---|---|
| Форма навчання | **денна** |
| Навчальний рік | **2018/2019** |
| Рік навчання | **2** |
| Кількість кредитів ECTS | **4** |
| Мова викладання, навчання та оцінювання | **англійська** |
| Форма заключного контролю | **екзамен** |

Викладач: **к.ф.-м.н., доцент Ченцов О.І.** (лекції, практичні заняття)

Пролонговано: на 2019/2020 н.р. _____ (прот. №9) «15» 04 2019 р.

на 2020/2021 н.р. _____ (протокол №8 «30» 03 2020 р.

**КИЇВ – 2018**

Розробник: Ченцов Олексій Ілліч, к. ф.-м. н., доцент, доцент кафедри інформаційних систем.

ЗАТВЕРДЖЕНО

Завідувач кафедри інформаційних систем

_____ О.І. Провотар

Протокол № 4 від «21» 12 2017 р.

Схвалено науково-методичною комісією факультету комп'ютерних наук та кібернетики

Протокол № 6 від «14» 02 2018 року

Голова науково-методичної комісії _____ д.ф.-м.н., пофесор Д.Я. Хусаінов

**1. Мета дисципліни** – формування компетентностей, необхідних для формальної перевірки коректності програм за допомогою сучасних автоматизованих засобів. Розуміння принципів побудови таких систем, основних прийомів при доведенні властивостей алгоритмічних систем та програмного забезпечення.

/

**Discipline aim.** The main aim of the course is to introduce PhD student to the field of formal software verification using modern automated tools. Also to form understanding of principles on which such systems are based, to master main techniques used to prove properties of software systems.

**2. Попередні вимоги до опанування або вибору навчальної дисципліни:**

1. **Мати уявлення:** про верифікацію та атестацію програмного забезпечення

**Знати:** основи функціонального програмування, основи математичної логіки.

/

**Prerequisites**

1. **To have notion of:** software verification and validation.
2. **To have knowledge of**: functional programming fundamentals, mathematical logic
2.

**3. Анотація навчальної дисципліни**: Навчальна дисципліна «Теоретичні основи програмування» є складовою освітньо-наукової програми підготовки фахівців за третім (освітньо-науковим) рівнем галузі знань 12 – «Інформаційні технології» зі спеціальності 121 «Інженерія програмного забезпечення», за освітньою програмою «Інженерія програмного забезпечення».

Дана дисципліна є вибірковою навчальною дисципліною в рамках освітньої програми «Інженерія програмного забезпечення». Вивчається на 2-му році навчання за освітньо-науковим рівнем «доктор філософії» в **обсязі – 120 год. (4 кредити ECTS)** зокрема: лекції – 18 год., практичні – 4 год., самостійна робота – 96 год., консультації – 2 год. У курсі передбачено 2 змістові частини, 2 контрольні роботи та тест. Завершується дисципліна – **екзаменом.**

Дисципліна «Теоретичні основи програмування» спрямована на поглиблення розуміння семантики та синтаксису мов програмування. Для цього на знайомих прикладах доводяться властивості програмного забезпечення. Робиться наголос на практичному аспекті доведення коректності програмних систем, а саме на використанні існуючих автоматизованих засобів. Вона забезпечує особистісний і професійний розвиток аспіранта та спрямована на формування досконалого володіння теоретичними знаннями для вирішення практичних завдань.

/

**Course description**. Course “Software foundations” is an elective course. Course “Software foundations” aims to deepen the understanding of syntax and semantics of programming languages. It emphasizes the practical aspect of proving software correctness, specifically on using existing state-of-the-art automated tools for proving correctness. This way it strives to develop strong theoretical knowledge based on solution of practical tasks.

**4. Завдання (навчальні цілі).** Основними завданнями дисципліни «Теоретичні основи програмування» є набуття знань, умінь та навичок (компетентностей) на рівні новітніх досягнень у області розробки інформаційних технологій, відповідно науково-освітньої кваліфікації «Доктор філософії». Зокрема, розвивати:

Розуміння теоретичних засад, що лежать в основі методів досліджень інформаційних систем та програмного забезпечення, методології проведення досліджень та обчислювальних експериментів (ФК-8).

/

**Objectives of study**. Primary objective of the "Software foundations" course is to obtain state-of-the-art knowledge, skills and competences in the field of information technology that corresponds to the PhD qualification level. Particularly, obtain understanding of theoretical foundations that underlie information system and software research techniques, methodology of research conduction and computational experiments. In particular, to develop understanding of theoretical foundations that are in the basis of investigation of information technologies and software systems, and computational experiments, methodology of the implementation of the environmental and enumeration experiments.

**5. Discipline learning outcomes / Результати навчання за дисципліною.**

| **Результат навчання (1. знати; 2. вміти; 3. комунікація; 4. автономність та відповідальність)** | | **Форми (та/або методи і технології) викладання і навчання** | **Методи оцінювання та пороговий критерій оцінювання (за необхідності)** | **Відсоток у підсумковій оцінці з дисципліни** |
|---|---|---|---|---|
| **Код** | **Результат навчання** | | | |
| РН1.1 | Знати основні принципи роботи засобів перевірки доведень, прийоми міркування щодо властивостей класів програм /<br>To know: fundamental principles of proof assistant functioning, strategies, tactics and techniques for reasoning about program properties | Лекції, самостійна робота / Lectures, individual work | Контрольна робота 1, 50% балів, виступ на лекції, тест, екзамен / Class exercises 1, 50% of points, oral presentation, test, exam | 25% |
| РН1.2 | Знати семантичні стилі мов програмування, поняття еквівалентності програм, «легковагові» формальні методи, що забезпечують безпеку мов програмування /<br>To know: programming languages semantic styles, program equivalence, "lightweight" formal methods used to provide for programming language safety | Лекції, самостійна робота / Lectures, individual work | Контрольна робота 1, 50% балів, виступ на лекції, тест, екзамен / Class exercises 2, 50% of points, oral presentation, test, exam | 25% |
| РН2.1 | Вміти використовувати функціональну мову автоматизованих засобів | Практичні заняття, самостійна робота / | Здача індивідуального завдання, контрольні роботи, екзамен / | 50% |

| | | | | |
|---|---|---|---|---|
| | перевірки доведень, специфікувати конструкції імперативних та/або функціональних мов, специфікувати логіку Хоара як окремий модуль системи перевірки доведень, добирати тактику для доведення властивостей програм./<br>To be able to: use functional language of automated proof assistant, write specification for the elements of functional and/or imperative languages, specify and apply Hoar logic as a separate module of proof assistant, choose appropriate tactics to prove program properties. | Practicals, individual work | Completed take-home exercises, class exercises, exam | |

**6. Correspondence between discipline learning outcomes and curriculum learning outcomes / Співвідношення результатів навчання дисципліни із програмними результатами навчання.**

| **Discipline learning outcomes**<br>**Curriculum learning outcomes** | **РН1.1** | **РН1.2** | **РН2.1** |
|---|---|---|---|
| **ПРН-11.** Розробляти засоби реалізації інформаційних технологій (методичні, інформаційні, математичні, алгоритмічні, технічні і програмні).<br>/<br>**ПРН-11.** Develop implementation tools for information technology (methodical, informational, mathematical, algorithmic, technical and software tools). | + | + | + |

| **Результати навчання дисципліни**<br>**Програмні результати навчання** | **РН1.1** | **РН1.2** | **РН2.1** |
|---|---|---|---|
| | + | + | + |

**7. Схема формування оцінки.**

**7.1 Форми оцінювання здобувачів освітньо-наукового рівня:**

**Оцінювання впродовж навчального періоду:**

1. Class exercises 1: РН 1.1, РН2.1 – **10 points/5 points**.

2. Class exercises 2: PH 1.2, PH2.1 – **10 points/5 points**.
3. Oral presentation: PH 1.1, PH 1.2 – **8 points/4 points**.
4. Test: PH 1.1, PH 1.2 – **12 points/6 points**.
5. Take-home exercise 1, 2: PH2.1 – **20 points/12 points**.

1. Контрольна робота 1: РН 1.1, РН2.1 – **10 балів/5 балів**.
2. Контрольна робота 2: РН 1.2, РН2.1 – **10 балів/5 балів**.
3. Виступ на лекції: РН 1.1, РН 1.2 – **8 балів/4 бали**.
4. Тест: РН 1.1, РН 1.2 – **12 балів/6 балів**.
5. Індивідуальні завдання 1, 2: РН2.1 – **20 балів/12 балів**.

**Підсумкове оцінювання:** екзамен

- Максимальна кількість балів які можуть бути отримані здобувачем: **40 балів.**
- Результати навчання які будуть оцінюватись: РН1.1, РН1.2, РН2.1.
- Форма проведення і види завдань: письмова робота.
- Види завдань: 4 письмових завдання.

До екзамену можуть бути **не допущені** здобувачі освітньо-наукового рівня, які впродовж навчального періоду набрали **менше ніж 20 балів**.

**7.2 Організація оцінювання:**

**Терміни проведення форм оцінювання:**

1. Контрольна робота 1: до 6 тижня навчального періоду.
2. Контрольна робота 2: до 10 тижня навчального періоду.
3. Виступ на лекції: до 9 тижня навчального періоду.
4. Тест: до 10 тижня навчального періоду.
5. Індивідуальне завдання 1: до 5 тижня навчального періоду.
6. Індивідуальне завдання 2: до 10 тижня навчального періоду.

Якщо здобувач з поважних причин, які підтверджено документально, був відсутній при написанні контрольної роботи, він має право на одне перескладання з можливістю отримання максимальної кількості балів. Термін перескладання визначається викладачем.

Здобувач має право здавати індивідуальні завдання після закінчення визначеного для них терміну, але з втратою одного балу за кожен тиждень, який пройшов з моменту закінчення терміну її здачі.

**7.3 Шкала відповідності оцінок**

| | |
|---|---|
| **Відмінно** / Excellent | 90-100 |
| **Добре** / Good | 75-89 |
| **Задовільно** / Satisfactory | 60-74 |
| **Незадовільно** / Fail | 0-59 |

**8. Lectures / Структура навчальної дисципліни. Тематичний план лекцій і практичних занять.**

| № лекції | Назва теми/лекції | Кількість годин | | |
|---|---|---|---|---|
| | | Лекції | Практичні заняття | Самост. робота |
| **Частина 1. Логічні основи програмування /** | | | | |

| | | | | |
|---|---|---|---|---|
| **Part 1. Logical Foundations** | | | | |
| 1 | **Тема 1.** Функціональне програмування в системі і Coq. /<br>**Topic 1**. Functional Programming in Coq. | 2 | | 10 |
| 2 | **Тема 2.** Індукція та інші важливі тактики доведення. /<br>**Topic 2**. Induction and Other Important Proof Tactics. | 2 | | 10 |
| 3 | **Тема 3.** Логіка у системі Coq. /<br>**Topic 3**. Logic in Coq. | 2 | | 10 |
| 4 | **Тема 4**. Застосування до простих імперативних програм./<br>**Topic 4**. Application to Simple Imperative Programs. | 2 | | 10 |
| 5 | **Тема 5**. Застосування до алгоритмів./<br>**Topic 5**. Application to Functional Algorithms.реалізованих на функціональних мовах. | 2 | | 12 |
| Контрольна робота 1 /Class exercises 1 | | | | 1 |
| Всього по частині 1 /Total (part 1) | | 18 | | 53 |
| **Частина 2. Теоретичні основи мов програмування /<br>Part 2. Programming Language Foundations** | | | | |
| 6 | **Тема 6**. Еквівалентність програм /<br>**Topic 6**. Program Equivalence. | 2 | | 10 |
| 7 | **Тема 7**. Логіка Хоара. /<br>**Topic 7**. Hoare Logic. | 2 | | 10 |
| 8 | **Тема 8**. Лямбда-числення з простою типізацією (ЛЧПТ). /<br>**Topic 8**. The Simply-Typed Lambda Calculus (STLC). | 2 | | 10 |
| 9 | **Тема 9**. Розширення ЛЧПТ: підтипи. /<br>**Topic 9**. SLTC Extensions: Subtyping. | 2 | 2 | 12 |
| Контрольна робота 2 /Class exercises 2 | | | | 1 |
| Тест /Test | | | | |
| Всього по частині 2 /Total (part 2) | | 8 | 2 | 43 |
| **ВСЬОГО /TOTAL** | | **18** | **4** | **96** |

Total hours – **120** hours, including:
Lectures **– 18** hours
Practicals – **4** hours
Individual work – **96** hours
Consultations – **2** hours

Загальний обсяг – **120** год., в тому числі:
Лекцій **– 18** год.
Практичні заняття – **4** год.
Самостійна робота – **96** год.
Консультації – **2** год.

**Topics for individual work / Теми, винесені на самостійне вивчення:**

1. Working with Structured Data.
2. Polymorphism and Higher-Order Functions.
3. The Curry-Howard Correspondence.
4. Lexing and Parsing in Coq.
5. Small-step Operational Semantics.
6. Type Systems.
7. A Typechecker for STLC.
8. STLC extensions: Records.
9. Normalization of STLC.

1. Міркування при роботі з структурами даних.
2. Поліморфізм та функції вищих порядків.
3. Відповідність Карі-Говарда.
4. Лексичний та семантичний аналіз у системі Coq.
5. Операційна семантика малих кроків.
6. Системи типів.
7. Перевірка типів для ЛЧПТ.
8. Розширення ЛЧПТ: записи.
9. Нормалізація ЛЧПТ.

**9. Recommended reading / Рекомендовані джерела:**

**Main / Основні:**


1. *Pierce B.* Software Foundations series. Vol. 1. Logical Foundations / Pierce B. et al. [online] — 2018. — Available: http://www.cis.upenn.edu/~bcpierce/sf/
2. *Pierce B.* Software Foundations series. Vol. 2. Programming Language Foundations / Pierce B. et al. [online] — 2018. — Available: https://softwarefoundations.cis.upenn.edu
3. *Lampropoulos L.* Software Foundations series. Vol. 4. Property-based Testing in Coq / Lampropoulos L., Pierce B. [online] — 2018. — Available: https://softwarefoundations.cis.upenn.edu
4. *Pierce B.* Software Foundations / Pierce B. et al. [online] — 2018. Available: http://www.cis.upenn.edu/~bcpierce/sf/
5. *Pierce B.* Types and Programming Languages / Pierce B. — Massachusetts: MIT Press, 2002. — 645p.
6. *Stamp A*. Verified functional programming in Agda. — London: Morgan et Claypool Publishers, 2016. — 258p.


**Additional / Додаткові:**


1. *Friedman D*. Essentials of Programming Languages, 3rd ed. / Friedman D., Wand M. — MIT Press, 2008. — 432p.

2. *Nordstrom B*. Programming in Martin-Lof's Type Theory: An Introduction / Nordstrom B., Petersson K., Smith J. — Oxford University Press, 1990. — 211p.
3. *Hehner E*. A Practical Theory of Programming. — Springer, 1993. — 250p. [Online] — Available: http://www.cs.toronto.edu/~hehner/aPToP/aPToP.pdf
4. *McAdam B.J*. Repairing Type Errors in Functional Programs: Ph.D. thesis / The University of Edinburgh. — Edinburgh, 2002. — 166pp.
5. *Taylor P*. An Exact Interpretation of while // Proceedings of the First Imperial College Department of Computing Workshop on Theory and Formal Methods. — London, UK: Springer-Verlag, 1993. — Pp.302-313.
6. *Dehnert J*. Fundamentals of Generic Programming / Dehnert J.C., Stepanov A.A. // Selected Papers from the International Seminar on Generic Programming. — London, UK: Springer-Verlag, 2000. — Pp. 1-11.
7. *Backhouse R*. Generic Programming: an Introduction / Backhouse R., Jansson P., Jeuring J., Meertens L. // Advanced Functional Programming Third International School. — Braga, Portugal. 1998. — Pp. 28–115.
8. *Rydeheard D*. Computational Category Theory / Rydeheard D., Burstall R. — New York, London: Prentice Hall, 1988. — 257pp.
9. *Bruce K*. Provable Isomorphisms of Types / Bruce K., Cosmo R., Longo G // Mathematical Structures in Computer Science. — 1992. — no. 2. — Pp. 231–247.
10. *Di Cosmo R*. Isomorphisms of Types: from Lambda-calculus to Information Retrieval and Language Design. — Birkhauser, 1995. — 235pp.