**DALYKO APRAŠAS**

| Dalyko pavadinimas | Kodas |
|---|---|
| Tipais grįstas programavimas | **TYDD7134** |

| Dėstytojai | Padalinys |
|---|---|
| **Koordinuojantis:** doc. Karolis Petrauskas<br>**Kiti:** partn. doc. Viačeslav Pozdniakov | Programų sistemų katedra,<br>Informatikos institutas,<br>Vilniaus universitetas |

| Studijų pakopa | Dalyko (modulio) lygmuo | Dalyko (modulio) tipas |
|---|---|---|
| Pirmoji | - | Pasirenkamas |

| Įgyvendinimo forma | Vykdymo laikotarpis | Vykdymo kalbos |
|---|---|---|
| Auditorinė | Pavasario semestras | Lietuvių, anglų |

| Reikalavimai studijuojančiajam | |
|---|---|
| **Išankstiniai reikalavimai:** „Funkcinis programavimas“. | **Gretutiniai reikalavimai (jei yra):** |

| Dalyko (modulio) apimtis ECTS kreditais | Visas studento darbo krūvis valandomis | Kontaktinio darbo valandos | Savarankiško darbo valandos |
|---|---|---|---|
| 5 | 130 | 64 | 66 |

**Dalyko (modulio) tikslas: studijų programos ugdomos kompetencijos**

Supažindinti studentus su išvystytomis programavimo kalbų tipų sistemomis, išnagrinėti jų taikymą programų sistemų inžinerijos uždaviniams spręsti, analizuoti susijusius teorinius pagrindus, ugdyti tipais grįsto programavimo ir verifikavimo įgūdžius.

**Bendrosios kompetencijos:**

- Nuolatinis mokymasis *(BK2)*.
  - Gebės savarankiškai įsisavinti naujas žinias, metodus ir įrankius bei taikyti juos praktikoje *(BK2.3)*.

**Dalykinės kompetencijos:**

- Konceptualių pagrindų žinios ir gebėjimai *(DK4)*:
  - Supras pagrindines programų sistemų inžinerijos koncepcijas bei sąvokas, įskaitant kelias priešakines sritis, suvoks galimas taikymo sritis ir žinos disciplinos aprėptį *(DK4.1)*.
  - Gebės taikyti matematikos pagrindų, mokslo, inžinerijos, kompiuterių mokslo teorines žinias ir algoritminius principus programų sistemų kūrime *(DK4.2)*.
  - Gebės abstrakčiai mąstyti, naudoti formalius aprašymo metodus, įrodinėti jų teisingumą, formalizuoti ir specifikuoti realaus pasaulio problemas *(DK4.3)*.
- Technologinės, metodinės žinios ir gebėjimai, profesinis kompetentingumas *(DK6)*:
  - Gebės derinti teoriją ir praktiką programų sistemų taikymo įvairiose srityse uždavinių sprendimui, įvertinant technologinį, ekonominį, socialinį ir teisinį kontekstą *(DK6.1)*.
  - Gebės parinkti ir panaudoti tinkamus šiuolaikinius metodus, modelius, problemų sprendimo šablonus, įgūdžius bei įrankius, būtinus programų sistemų kūrimui ir priežiūrai, įskaitant naujas taikymo sritis *(DK6.2)*.
  - Gebės panaudoti esamą kompiuterių techninę ir programinę įrangą, identifikuoti, perprasti ir taikyti perspektyvias technologijas *(DK6.3)*.

| Dalyko (modulio) studijų siekiniai | Studijų metodai | Vertinimo metodai |
|---|---|---|
| Studentai supras priklausomus tipus, gebės taikyti juos programų sistemų inžinerijos uždaviniams spręsti. | Paskaitos, į problemas orientuotas dėstymas, atvejų analizė, informacijos paieška, literatūros skaitymas, individualus darbas, pratybos, laboratorinių darbų atlikimas. | Laboratorinių darbai ir jų rezultatų pristatymas, egzaminas raštu (atviri, pusiau atviri ir uždari klausimai ir užduotys). |
| Gebės formuluoti dalykinės srities savybes priklausomais tipais ir įgyvendinti tas savybes atitinkančias programas. | | |
| Gebės analizuoti mokslinius straipsnius susijusius su tipų sistemomis, įvertinti jų indėlį mokslo srityje. | | |

| | | |
|---|---|---|
| Taikyti tipais grįstą programavimą Idris2 kalboje, suprasti konceptų sąsajas su kitų panašių kalbų konstrukcijomis. | | |

| **Temos** | **Kontaktinio darbo valandos** | | | | | | | **Savarankiškų studijų laikas ir užduotys** | |
|---|---|---|---|---|---|---|---|---|---|
| | Paskaitos | Konsultacijos | Seminarai | Pratybos | Laboratoriniai darbai | Praktika | **Visas kontaktinis darbas** | **Savarankiškas darbas** | **Užduotys** |
| 1. Įvadas ir pagrindai. | 2 | | | | | 2 | **4** | **4** | Užduotys pratybų metu.<br>Laboratoriniai darbai.<br>Susijusių straipsnių aptarimas.<br>Savarankiškas literatūros studijavimas. |
| 2. Vektorius, kaip ilgiu indeksuotas tipas, jo panaudojimas, savybių verifikavimas. | 2 | | | | | 2 | **4** | **5** | |
| 3. Naudotojo apibrėžti tipai. | 2 | | | | | 2 | **4** | **5** | |
| 4. Tipais grįstas programavimas, kalbos lygmens ir priklausomi tipai. | 4 | | | | | 4 | **8** | **6** | |
| 5. Lygybė, perrašymas tipuose, pilnumas ir išsprendžiamumas. | 2 | | | | | 2 | **4** | **4** | |
| 6. Curry-Howard atitikimas, įrodymai programuojant. | 2 | | | | | 2 | **4** | **3** | |
| 7. Sudėtingų potipių išreiškimas predikatais. | 2 | | | | | 2 | **4** | **5** | |
| 8. Alternatyvus duomenų struktūrų dekonstravimas taikant rodinius. | 2 | | | | | 2 | **4** | **5** | |
| 9. Srautai ir begalinis programų vykdymas. | 2 | | | | | 2 | **4** | **5** | |
| 10. Tipizuoti automatai kaip protokolų specifikacijos. | 4 | | | | | 4 | **8** | **6** | |
| 11. Sesijų tipai ir lygiagretumas. | 2 | | | | | 2 | **4** | **5** | |
| 12. Tiesiniai tipai ir resursų valdymas. | 4 | | | | | 4 | **8** | **6** | |
| 13. Apibendrinimas ir sąsajos su kitomis programavimo kalbomis. | 2 | | | | | 2 | **4** | **3** | |
| 14. Pasiruošimas egzaminui ir jo laikymas (raštu). | | | | | | | | **4** | |
| **Iš viso** | **32** | | | | | **32** | **64** | **66** | |

| **Vertinimo strategija** | **Svoris, %** | **Atsiskaitymo laikas** | **Vertinimo kriterijai** |
|---|---|---|---|
| Praktinės užduotys | 30 | Kiekvieną savaitę | Studentai atliks praktines užduotis susijusias su paskaitos tema. Bet kuris studentas turi gebėti klasei pristatyti savo sprendimą, paaiškinti kaip ir kodėl toks sprendimas priimtas. |
| Laboratoriniai darbai | 30 | Progresas deklaruojamas kas savaitę. Galutinis pristatymas 16-tą semestro savaitę. | Studentai pasirenka dalykinę sritį ir sudaro programą taikant tipais grįsto programavimo metodus. Programa turi panaudoti visus pagrindinius tipais grįsto programavimo elementus, tokius kaip priklausomi tipai, tipizuoti automatai, sesijų ir tiesiniai tipai ir kitus. Taip pat, turi būti apibrėžtos ir įrodytos su dalykine sritimi susijusios programos savybės. Studentai turi gebėti paaiškinti visus sprendimus ir pakartotinai įgyvendinti nedideles funkcijas paaiškinant visus žingsnius. Darbo progresas turi būti deklaruojamas kiekvieną savaitę. |

| | | | |
|---|---|---|---|
| Egzaminas raštu | 40 | Sesijos metu | Egzaminą sudaro atvirojo, pusiau atvirojo ir uždarojo tipo klausimai iš per paskaitas išdėstytų temų. |

| **Autorius** | **Leidimo metai** | **Pavadinimas** | **Leidinio Nr. Ar tomas** | **Leidykla ar internetinė nuoroda** |
|---|---|---|---|---|
| **Privalomi studijų šaltiniai** | | | | |
| Edwin Brady | 2017 | Type-Driven Development with Idris | | Manning.<br>ISBN: 978-1617293023 |
| **Papildomi studijų šaltiniai** | | | | |
| Adam Chlipala | 2013 | Certified Programming with Dependent Types | | The MIT Press.<br>ISBN: 978-0262026659 |
| Samuel Mimram | 2020 | Program = Proof | | Independently published.<br>ISBN: 979-8615591839 |

**COURSE UNIT DESCRIPTION**

| Course unit title | Course unit code |
|---|---|
| Type-driven development | |

| Lecturer(s) | Department where the course unit is delivered |
|---|---|
| **Coordinator:** assoc. prof. Karolis Petrauskas<br>**Other lecturers:** p'ship assoc. prof. Viačeslav Pozdniakov | Department of Software Engineering, Institute of Compute Science, Vilnius University |

| Cycle | Level of course unit | Type of the course unit |
|---|---|---|
| First | - | Elective |

| Mode of delivery | Semester or period when the course unit is delivered | Language of instruction |
|---|---|---|
| Face-to-face | Spring semester | Lithuanian, English |

| Prerequisites and corequisites | |
|---|---|
| **Prerequisites:** "Functional programming". | **Corequisites (if any): -** |

| Number of ECTS credits allocated | Student's workload | Contact hours | Self-study hours |
|---|---|---|---|
| 5 | 130 | 64 | 66 |

**Purpose of the course unit: programme competences to be developed**

Introduce students to elaborated programming language type systems, explore their application to software engineering problems, analyze relevant theoretical background, and develop skills in type-driven development and verification of software.

**Generic competences:**

- Continuous learning *(GC2)*.
  - Will be able to independently acquire new knowledge, methods, and tools and apply them in practice *(GC2.3)*.

**Specific competences:**

- Knowledge and abilities of conceptual foundations *(SC4)*:
  - Students will understand the basic concepts of software engineering, including several frontier areas, potential application domains, and the discipline's scope *(SC4.1)*.
  - Will be able to apply theoretical knowledge in mathematics, software engineering, computer science, and algorithmic principles to build software systems *(SC4.2)*.
  - Will be able to think abstractly, use formal methods, prove correctness, formalize, and model real-world problems *(SC4.3)*.
- Technological, methodological knowledge and abilities, professional competence *(SC6)*:
  - Will be able to combine the theory and practice of software engineering, considering problem-solving techniques and assessing technological, economic, social, and legal contexts *(SC6.1)*.
  - Will be able to select and use appropriate modern methods, models, problem-solving patterns, skills, and tools necessary for developing and maintaining application systems, including new application areas *(SC6.2)*.
  - Will be able to use existing computer hardware and software, identify, understand, and apply frontier technologies *(SC6.3)*.

| Learning outcomes of the course unit: students will be able to | Teaching and learning methods | Assessment methods |
|---|---|---|
| Understand dependent types and apply them to software engineering problems. | Lectures, problem-oriented teaching, case studies, information retrieval, literary reading, individual work, tutorials, lab assignments. | Lab assignments and presentation of their results, written exam (open, semi-open and close-ended questions and tasks). |
| Formulate desired software properties as dependent types and develop implementations conforming to those properties. | | |

| | | |
|---|---|---|
| Analyze scientific papers in the domain of type systems and understand their contribution in the context of core knowledge. | | |
| Apply type-driven development in the Idris2 programming language and understand conceptual relations to other languages with similar type systems. | | |

| **Course content: breakdown of the topics** | **Contact hours** | | | | | | | **Self-study work: time and assignments** | |
|---|---|---|---|---|---|---|---|---|---|
| | Lectures | Tutorials | Seminars | Practice | Lab assignmens | Practical training | **Contact hours** | **Self-study hours** | **Assignments** |
| 1. Introduction and fundamentals. | 2 | | | | | 2 | **4** | **4** | Tasks at practical classes.<br>Laboratory assignments.<br>Discussion on related papers.<br>Self-study of literature. |
| 2. A vector as a length indexed type, its use cases, verifying its properties. | 2 | | | | | 2 | **4** | **5** | |
| 3. User-defined datatypes. | 2 | | | | | 2 | **4** | **5** | |
| 4. Type-driven development, programming with first-class and dependent types. | 4 | | | | | 4 | **8** | **6** | |
| 5. Equality, rewrites in types, totality and decidability. | 2 | | | | | 2 | **4** | **4** | |
| 6. Curry-Howard correspondence, proving by programming. | 2 | | | | | 2 | **4** | **3** | |
| 7. Expressing complex subtypes with predicates. | 2 | | | | | 2 | **4** | **5** | |
| 8. Decomposing data structures in alternative ways using views. | 2 | | | | | 2 | **4** | **5** | |
| 9. Streams and infinite execution. | 2 | | | | | 2 | **4** | **5** | |
| 10. Typed state machines as protocol specifications. | 4 | | | | | 4 | **8** | **6** | |
| 11. Session types and concurrency. | 2 | | | | | 2 | **4** | **5** | |
| 12. Linear types and resource management. | 4 | | | | | 4 | **8** | **6** | |
| 13. Summary and relation to other programming languages. | 2 | | | | | 2 | **4** | **3** | |
| 14. Preparing for the exam and taking the final exam (written). | | | | | | | | **4** | |
| **Total** | **32** | | | | | **32** | **64** | **66** | |

| **Assessment strategy** | **Weight,%** | **Deadline** | **Assessment criteria** |
|---|---|---|---|
| Practical assignments | 30 | Each week. | Students perform small practical assignments related to the topic of the lecture. Any student should be able to present his work for a class, explaining how and why he came to the solution. |
| Lab assignment | 30 | Progress declared each week.<br>Final presentation – 16th week of the semester. | Students choose a domain and develop a program by applying type-driven development practices. The program has to use all the main type-driven constructs like dependent types, typed state machines, session, and linear types, and others. Also, domain-specific properties have to be stated and proved. The student has to be able to explain all the decisions and re-implement a small function explaining all the steps. Progress has to be declared every week. |

| | | | |
|---|---|---|---|
| Exam (written) | 40 | Exam session. | Exam consists of open, semi-open and close-ended questions from the topics covered in lectures. |

| Author | Year | Title | Number or volume | Publisher or URL |
|---|---|---|---|---|
| **Required reading** | | | | |
| Edwin Brady | 2017 | Type-Driven Development with Idris | | Manning.<br>ISBN: 978-1617293023 |
| **Recommended reading** | | | | |
| Adam Chlipala | 2013 | Certified Programming with Dependent Types | | The MIT Press.<br>ISBN: 978-0262026659 |
| Samuel Mimram | 2020 | Program = Proof | | Independently published.<br>ISBN: 979-8615591839 |