RTU studiju kurss "Paralēlā funkcionālā programmēšana"

33000 Datorzinātnes, informācijas tehnoloģijas un enerģētikas fakultāte

Vispārējā informācija

Kods	DE0944
Nosaukums	Paralēlā funkcionālā programmēšana
Studiju kursa statuss programmā	Obligātais/Ierobežotās izvēles; Brīvās izvēles
Atbildīgais mācībspēks	Pāvels Rusakovs - Doktors, Asociētais profesors
Apjoms daļās un kredītpunktos	1 daļa, 6.0 kredītpunkti
Studiju kursa īstenošanas valodas	LV, EN
Anotācija	Studiju kursā tiek aplūkota paralēlā programmēšana modernajās funkcionālajās programmēšanas valodās. pārsvarā Erlang (Elixir) un F# valodās. Sniegts apskats par funkcionālo un loģisko programmēšanu, pavedieniem un dalītājiem aprēķiniem. Aplūkoti matemātiski balstīto valodu teorētiskie pamati. Programmēšanas līmenī detalizēti izskatīta pavedienu veidošana un pavedienu savstarpējā iedarbība, resursu aizsardzība, dalītie aprēķini un dalīto sistēmu modelēšana. Analizēti ar funkcionālo paralēlo programmēšanu saistīti papildu satvari un bibliotēkas. Laboratorijas darbos studentiem ir nodrošināta iespēja pilnveidot studiju kursā apgūtās teorētiskās zināšanas un programmēšanas prasmes.
Mērķis un uzdevumi, izteikti kompetencēs un prasmēs	Studiju kursa mērķis ir sagatavot studentu funkcionālās paradigmas lietošanai paralēlajā/laiksakritīgajā programmēšanā un dalītājiem aprēķiniem. Studiju kursa uzdevumi: - veicināt izpratni par funkcionālo un loģisko programmēšanu; - iemācīt funkcionālo un loģisko programmēšanas valodu matemātisko aparātu; - attīstīt prasmi lietot funkcionālās programmēšanas iespējas imperatīvajās valodās; - attīstīt pavedienu vadības un dalīto aprēķinu prasmes; - iemācīt pielietot funkcionālās valodas dalīto sistēmu izstrādei; - attīstīt prasmes lietot funkcionālo programmēšanu globālajā tīmeklī.
Patstāvīgais darbs, tā organizācija un uzdevumi	Patstāvīgais darbs tiek organizēts, studentiem individuāli izpildot uzdevumus par paralēlās programmēšanas un dalīto aprēķinu izmantošanu funkcionālajās programmēšanas valodās, tās lietošanu globālajā tīmeklī un drošuma paaugstināšanu vairākpavedienošanas kontekstā. Patstāvīga gatavošanās eksāmenam.
Literatūra	Obligātā / Obligatory: 1. Michal Plachta. Grokking Functional Programming. 2023, 520 lpp. https://www.oreilly.com/library/view/grokking-functional-programming/9781617291838/ 2. Simon St. Laurent. Introducing Erlang, 2nd Edition. 2017, 212 lpp. https://www.oreilly.com/library/view/introducing-erlang-2nd/9781491973363/ 3. Sasa Juric. Elixir in Action, Second Edition. 2019, 384 lpp. https://www.oreilly.com/library/view/elixir-in-action/9781617295027/ 4. Svilen Gospodinov. Concurrent Data Processing in Elixir. 2021, 176 lpp. https://www.oreilly.com/library/view/concurrent-data-processing/9781680508956/ Papildu / Additional: 1. Francesco Cesarini, Simon Thompson. Erlang Programming, 1st Edition. 2009, 470 lpp. https://www.oreilly.com/library/view/erlang-programming/9780596803940/ 2. Aditya Iyengar, Eric Sullivan. Build Your Own Web Framework in Elixir. 2023, 274 lpp. https://www.oreilly.com/library/view/build-your-own/9781801812542/ 3. Andrea Leopardi, Jeffrey Matthias. Testing Elixir. 2021, 264 lpp. https://www.oreilly.com/library/view/testing-elixir/9781680508918/ 4. Francesco Cesarini, Steve Vinoski. Designing for Scalability with Erlang/OTP. 2016, 482 lpp. https://www.oreilly.com/library/view/designing-for-scalability/9781449361556/ 5. Ulisses Almeida. Learn Functional Programming with Elixir: New Foundations for a New World (The Pragmatic Programmers). Pragmatic Bookshelf; 1st edition. 6. James Gray II, Bruce Tate. Designing Elixir Systems With OTP: Write Highly Scalable, Self-healing Software with Layers. 2020, 248 lpp. Pragmatic Bookshelf; 1st edition. 7. Saša Juric. Elixir in Action, Third Edition. 2024, 416 lpp. Manning; 3rd edition. 8. James Edward Gray, Bruce A. Tate .Designing Elixir Systems With OTP. 2019, 248 lpp. https://www.oreilly.com/library/view/designing-elixir-systems/9781680507362/ 9. Kit Eason. Stylish F# 6: Crafting Elegant Functional Code for .NET 6. 2021, 546 lpp. Apress; 2nd edition. 10. Isaac Abraham. F# in Action. 2024, 331 lpp. Manning.
Nepieciešamās priekšzināšanas	Procedurālā programmēšana, objektorientētā programmēšana

Studiju kursa saturs

Saturs	Pilna un nepilna laika klātienes studijas		Nepilna laika neklātienes studijas	
	Kontakt stundas	Patstāv. darbs	Kontakt stundas	Patstāv. darbs
Matemātiskās programmēšanas valodas. Funkcionālā un loģiskā programmēšana.	4	4	0	0
Imperatīvās un funkcionālās programmēšanas salīdzināšana.	2	2	0	0
Funkcionālās programmēšanas elementi un principi.	6	6	0	0

Funkcijas un predikāti, to līdzība un atšķirības.	4	4	0	0
Funkcionālā programmēšana objektorientētajās valodās.	4	4	0	0
Erlang valodas īpašības funkcionālās programmēšanas kontekstā.	6	6	0	0
Standarta moduļu bibliotēka un šablonu risinājumu bibliotēka – satvars OTP (Open Telecom Platform) Erlang valodā.	6	6	0	0
Pavedienu koncepcija Erlang valodā.	6	6	0	0
Aktoru modelis un tā realizācija Erlang valodā.	4	4	0	0
Savstarpējā pavedienu iedarbība Erlang valodā. Asinhronie ziņojumi.	4	4	0	0
Dalītie aprēķini Erlang valodā.	4	4	0	0
Drošums un drošība Erlang valodā.	6	6	0	0
Elixir valoda kā Erlang valodas uzbūve.	4	4	0	0
F# valoda un funkcionālā paralēlā programmēšana .NET platformā.	6	6	0	0
Funkcionālā pieeja un aktoru modelis pavedienos Scala valodā.	6	6	0	0
Paralēlās dalītās sistēmas globālajā tīmeklī.	4	4	0	0
Erlang un Elixir valodas reāllaika sistēmu kontekstā.	4	4	0	0
Kopā:	80	80	0	0

Sasniedzamie studiju rezultāti un to vērtēšana

Sasniedzamie studiju rezultāti	Rezultātu vērtēšanas metodes
Spēj izstrādāt programmatūru funkcionālajās un loģiskajās valodās.	Laboratorijas darbu izpilde un aizstāvēšana. Kritēriji: spēj veidot programmu no funkcijām, predikātiem un moduļiem.
Spēj uzlabot objektorientētās programmas ar funkcionālās programmēšanas konstrukcijām.	Laboratorijas darbu izpilde un aizstāvēšana. Kritēriji: spēj pielietot funkcionālo pieeju objektorientētajās un procedurālajās programmās.
Spēj izstrādāt paralēlo/laiksakritīgo programmatūru Erlang un Elixir valodās.	Laboratorijas darbu izpilde un aizstāvēšana. Kritēriji: spēj kontrolēt pavedienus un nodrošināt resursu aizsardzību.
Spēj pielietot aktoru modeli funkcionālajās valodās.	Laboratorijas darbu izpilde un aizstāvēšana. Kritēriji: spēj implementēt aktoru modeli neatkarīgi no programmēšanas valodas.
Spēj spriest par vairākiem paralelitātes un dalīto aprēķinu jautājumiem.	Eksāmens. Kritēriji: spēj atbildēt uz teorētiskajiem jautājumiem, rakstīt programmas, atklāt un izskaidrot kļūdas eksistējošās programmās.

Studiju rezultātu vērtēšanas kritēriji

Kritērijs	% no kopējā vērtējuma
Laboratorijas darbi	50
Eksāmens	50
Kopā:	100

Studiju kursa plānojums

Daļa	KP	Stundas			Pārbaudījumi			Brīvās izvēles pārbaudījumi		
		Lekcijas	Prakt d.	Laborat	Ieskaite	Eksām.	Darbs	Ieskaite	Eksām.	Darbs
1.	6.0	32.0	0.0	32.0		*			*	