МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ ТЕХНОЛОГІЧНИЙ УНІВЕРСИТЕТ

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І СИСТЕМ

**МЕТОДИЧНІ РЕКОМЕНДАЦІЇ**

**до самостійної роботи**

з дисципліни «Функціональне програмування»

для здобувачів освітнього ступеня «бакалавр»

зі спеціальності 126 Інформаційні системи та технології

(освітньої програми «Web-технології, Web-дизайн»)

усіх форм навчання

Черкаси

2020

|  |  |
| --- | --- |
| УДК 004.42(07)  М 54 | *Затверджено вченою радою ФІТІС,*  *протокол № 4 від 14.02.2020 р.,*  *згідно з рішенням кафедри*  *інформаційних технологій проектування,*  *протокол № 8 від 10.01.2020 р.* |

|  |  |
| --- | --- |
| Упорядник: | Тарасенко Я. В., к.т.н. |
| Рецензент: | Миронець І. В., к.т.н., доцент |

|  |  |
| --- | --- |
| М 54 | **Методичні** рекомендації до самостійної роботи з дисципліни «Функціональне програмування» для здобувачів освітнього ступеня бакалавра спеціальності № 126 «Інформаційні системи та технології» (освітня програма «Web-технології, Web-дизайн») усіх форм навчання [Електронний ресурс] / [Упоряд.: Я.В. Тарасенко]; М-во освіти і науки України, Черкас. держ. технол. ун-т. – Черкаси : ЧДТУ, 2020. – 15 с. – Назва з титульного екрана. |

Методичні рекомендації містять основні теоретичні відомості відповідної теми, які закріплюються та поглиблюються самостійною роботою, перелік запитань та завдань для самостійного опрацювання, перелік літератури, яку здобувачі можуть використовувати при вивченні дисципліни «Функціональне програмування» та методичні вказівки для виконання завдань самостійного опрацювання. Особлива увага приділяється формуванню навичок надання програмній системі чіткого математичного тлумачення та застосування функціонального підходу програмування в професійній діяльності на основі мови програмування Haskell.

Для здобувачів освітнього ступеня бакалавр спеціальності № 126 «Інформаційні системи та технології» (освітньої програми «Web-технології, Web-дизайн») усіх форм навчання.

УДК 004.42(07)

Виробничо-практичне

електронне видання

комбінованого використовування

**МЕТОДИЧНІ РЕКОМЕНДАЦІЇ**

**до самостійної роботи**

з дисципліни «Функціональне програмування»

для здобувачів освітнього ступеня «бакалавр»

зі спеціальності 126 Інформаційні системи та технології

(освітньої програми «Web-технології, Web-дизайн»)

усіх форм навчання

Упорядник **Тарасенко** Ярослав Володимирович

*В авторській редакції.*

ЗМІСТ

[ВСТУП 4](#_Toc34041582)

[Порядок виконання роботи та звітність 5](#_Toc34041583)

[Самостійна робота № 1.](#_Toc34041584) [Модульність в функціональному програмуванні 6](#_Toc34041585)

[Самостійна робота № 2.](#_Toc34041586) [Визначення нових типів даних 7](#_Toc34041587)

[Самостійна робота № 3.](#_Toc34041588) [Замикання (локальні визначення) 8](#_Toc34041589)

[Самостійна робота № 4.](#_Toc34041590) [Часткове використання функцій 9](#_Toc34041591)

[Самостійна робота № 5.](#_Toc34041592) [Налагодження програм на Haskell 10](#_Toc34041593)

[Самостійна робота № 6.](#_Toc34041594) [Подання програм при розширеному лямбда-численні 11](#_Toc34041595)

[Самостійна робота № 7.](#_Toc34041596) [Перетворення графів при виконанні програм 12](#_Toc34041597)

[Самостійна робота № 8.](#_Toc34041598) [Подання рекурсивних функцій при редукції на графах 13](#_Toc34041599)

[ЛІТЕРАТУРА 14](#_Toc34041600)

# ВСТУП

Навчальна дисципліна «Функціональне програмування» належить до циклу дисциплін професійної підготовки і має міждисциплінарні зв’язки з такими дисциплінами, як «Web-програмування», «Алгоритмізація та програмування», «Технології обробки природо мовної інформації», «Вища математика», «Теорія ймовірності та математична статистика», «Методи та системи штучного інтелекту», «Технології та системи підтримки прийняття рішень», «Прикладні інтелектуальні системи обробки даних», «Професійний практикум».

Програма навчальної дисципліни складається з двох частин:

1. Вступ до Haskell. Основи функціонального програмування.

2. Застосування функціонального підходу в програмуванні.

Методичні рекомендації налічують опис процесу самостійного опрацювання 8 тем з уточненням мети роботи, основних теоретичних відомостей відповідної теми, які закріплюються та поглиблюються самостійною роботою, переліку запитань та завдань для самостійного опрацювання, переліку літератури, яку здобувачі можуть використовувати при вивченні дисципліни та виконанні завдань самостійного опрацювання.

Метою самостійної роботи є закріплення теоретичних основ функціональної парадигми програмування, а також поглиблення практичних навичок проведення лінивих обчислень засобами функціонального програмування, застосування функціонального підходу при програмуванні мовою Haskell, компілювання програм на Haskell, проведення лямбда-числення і забезпечення представлення, виконання та інтерпретації програм мовою Haskell.

Самостійна робота спроектована таким чином, що по завершенню курсу, здобувачі вищої освіти закріплюють набуті компетенції, які формують здатність проектувати, розробляти та використовувати засоби реалізації інформаційних систем, технологій та інфокомунікацій (методичні, інформаційні, алгоритмічні, технічні, програмні та інші), здатність до аналізу, синтезу і оптимізації інформаційних систем та технологій з використанням математичних моделей і методів, знання сучасного рівня технологій інформаційних систем, практичні навички програмування та використання прикладних і спеціалізованих комп’ютерних систем та середовищ з метою їх запровадження у професійній діяльності.

Крім того, матеріали методичних вказівок до самостійної роботи можуть бути використані при виконання лабораторних робіт та підготовці до лекційних занять.

При проектуванні самостійної роботи використані такі новітні методи навчання, як модульне та проблемне навчання. При цьому, проблемне інтегроване з модульним.

# Порядок виконання роботи та звітність

Самостійна робота має на меті самостійне поглиблене опрацювання здобувачем вищої освіти тем лекційного курсу використовуючи пропоновану літературу та метод проблемного навчання. Мається на увазі, що в ході роботи здобувач буде зустрічати певні незрозумілі на даному етапі питання. Ці питання формують певну проблематику, обговорення якої під час лекційних занять та лабораторних робіт прибере прогалини у загальних знаннях здобувача із курсу та допоможе у набутті ним відповідних компетенцій, на що розрахований курс.

Самостійна робота проводиться згідно з навантаженням та тематикою, що висвітлені у силабусі та робочій програмі дисципліни. Вона являє собою організовану викладачем діяльність здобувача вищої освіти, направлену на поглиблення набутих умінь та навичок здобувача.

Робота може проводитись у будь-якому приміщенні, обладнаному комп’ютерною технікою та доступом в Інтернет (бібліотека, навчальні аудиторії, домашні умови).

Самостійна робота складається з двох частин:

1) пошук відповіді на запитання, що виносяться на самостійне опрацювання шляхом опрацювання літератури, перегляд пропонованих у силабусі відео-лекцій та електронних ресурсів;

2) вирішення конкретної практичної задачі.

Контроль виконання самостійної роботи здійснюється у формі співбесіди, де розглядаються результати виконання практичного завдання та рівень засвоєння здобувачем теоретичних знань.

Критерії успішного самостійного опрацювання матеріалу:

1) глибокі знання пропонованої літератури;

2) опрацювання додаткових джерел та матеріалів;

3) вирішення сформованої в ході роботи проблематики;

4) набуті практичні навички;

5) системність отриманих знань та інтегрованість у лекційний курс.

За кожною темою самостійної роботи коротко наводиться зв'язок із темою лекційного завдання та наводяться рекомендації з напрямку та розширеної тематики лекційного курсу, на які слід звернути особливу увагу в ході виконання завдання, а також при підготовці до лекційного заняття та виконання лабораторної роботи. Практичне завдання направлене на доповнення навичок, здобутих під час виконання лабораторних робіт, а також розширення навичок, шляхом виконання завдань, аналогічних лабораторним роботам, проте, які відрізняються особливими умовами, оточенням, основною метою чи нестандартним підходом до вирішення поставленої перед здобувачем вищої освіти задачі.

# Самостійна робота № 1

# Модульність в функціональному програмуванні

**Мета роботи** – ознайомитися з механізмом модульності у функціональному програмуванні. Розширити навички розподілення функцій між частинами програми. Поглибити базові знання технологій функціонального підходу програмування.

**Теоретичні відомості для поглибленого опрацювання**

Тема лекції №1 «Базові парадигми та математична основа функціонального програмування» має на меті розкриття основних аспектів функціонального підходу програмування в порівнянні з імперативним підходом. Для цього описуються основні поняття та принципи функціонального програмування, наголошується на різниці функціонального та об’єктного підходів та розглядається реалізація основних принципів функціонального програмування засобами різних мов програмування, в тому числі мовою Haskell. На самостійне опрацювання виноситься питання модульності у функціональному програмуванні. Підтримка модульності хоч і не є властивою даному стилю, проте підтримується і покращує розуміння різниці функціонального та імперативного підходів.

Для поглиблення знань в області основ функціонального підходу програмування, слід особливу увагу звернути на різницю реалізації модульності в імперативних мовах програмування та функціональних. Необхідно виявити, що саме дозволяє робити механізм модульності і навіщо застосовується подібний підхід. Крім того, слід розуміти механізми реалізації модульності у функціональних мовах програмування.

**Запитання для самостійного опрацювання:**

1. Що являє собою модульне програмування?
2. Які переваги та недоліки модульного програмування?
3. Яким чином модульність реалізується при застосування функціонального підходу програмування?
4. Що таке інверсія керування та які методи застосовуються при реалізації модульності у функціональному програмуванні?
5. Як реалізовано систему модулів у Haskell?

**Практичне завдання для самостійної роботи:**

Реалізувати програму знаходження суми максимального та мінімального значення серед чисел, які є мінімальними значеннями у кожному зі стовпців матриці, використовуючи функціональний стиль програмування із розподіленням функцій за модульною системою.

# Самостійна робота № 2

# Визначення нових типів даних

**Мета роботи –** ознайомитися з механізмом визначення нових типів даних в Haskell. Розширити навички використання існуючих простих типів даних. Поглибити знання механізмів обчислення з невизначеним результатом.

**Теоретичні відомості для поглибленого опрацювання**

Тема лекції №2 «Основи мови Haskell» має на меті розкриття основних аспектів програмування мовою Haskell. Для цього описуються особливості системи програмування Haskell Platform, процес визначення функцій за допомогою рівнянь, пояснюються типи функцій, основи роботи зі списками та типами даних. На самостійне опрацювання виноситься питання визначення нових типів даних для закріплення та розширення набутих знань з основ Haskell.

Для поглиблення знань в області базового програмування мовою Haskell, а саме визначення нових типів даних, слід особливу увагу звернути на типи і класи, як такі, до яких належать класичні типи даних. Визначення нових типів даних має на увазі і визначення класу, до якого вони будуть належати. Крім того, необхідно опрацювати механізми обчислення з невизначеним результатом, які прямо відносяться до визначення типів даних. Слід дослідити процес виводу класів типів та можливість рекурсивного визначення типів даних (рекурсивні структури даних). Доцільно засвоїти механізм використання синонімів типів та конструкторів типів, а також призначення та особливості кожної технології.

**Запитання для самостійного опрацювання:**

1. Яким чином оголошується новий, не існуючий за замовчуванням тип даних мовою Haskell?
2. Що таке конструктори значень та яку роль вони відіграють у визначенні типів даних?
3. Яким чином використовуються конструктори типів?
4. Яку роль у визначенні нових типів відіграють класи та яким чином відбувається визначення нових класів типів?
5. В чому суть механізму обчислення з невизначеним результатом та в яких випадках його доцільно використовувати?

**Практичне завдання для самостійної роботи:**

Перевизначити існуючий клас простих типів даних, визначений за замовчуванням та дати їм нові імена, використовуючи механізм синонімів типів із застосуванням параметрів типів.

# Самостійна робота № 3

# Замикання (локальні визначення)

**Мета роботи –** ознайомитися з особливостями механізму функціонального програмування – замикання. Розширити навички використання анонімних функцій. Поглибити знання в області написання нетривіальних функцій вищого порядку.

**Теоретичні відомості для поглибленого опрацювання**

Тема лекції №3 «Функції вищого порядку» має на меті розкриття основних особливостей утворення та використання функцій вищого порядку. Для цього розглядається відображення і згортка, лямбда-вирази та застосування функцій вищого порядку для обробки списків та складних структур. На самостійне опрацювання виноситься питання застосування механізму функціонального програмування – замикання для кращого засвоєння нетривіальних функцій вищого порядку.

Для поглиблення знань в області функцій вищого порядку, особливо при реалізації нетривіальних функцій, слід особливу увагу звернути на анонімні функції взагалі, одним з видів якої на ряду з лямбда-функцією стоїть замикання. Необхідно розглянути особливості локальних визначень та сфери застосування, а також умови використання локальних визначень лише всередині функцій. Крім того, необхідно проаналізувати види замикань та характерні риси кожного з двох видів, а також процес оптимізації обрахунків на основі замикання, що зумовлений можливістю обрахунку замикання лише один раз. Слід визначити схожі риси та різницю між двома видами локальних визначень.

**Запитання для самостійного опрацювання:**

1. Що являє собою замикання (closure) та яким чином воно використовується для оптимізації визначень функцій?
2. Що витікає з твердження, що замикання знаходиться в області імен основної функції?
3. У яких випадках використовується замикання?
4. Яким чином замикання використовується в написання нетривіальних функцій вищого порядку?
5. Які існують види замикання та в чому їх суть?

**Практичне завдання для самостійної роботи:**

Реалізувати простий лічильник, який зчитує свої виклики та повертає їх поточну кількість використовуючи замикання (локальні визначення) мовою Haskell.

# Самостійна робота № 4

# Часткове використання функцій

**Мета роботи –** ознайомитися з відмінностями між каррінгом та частковим використанням функцій. Розширити навички каррінгу та часткового використання функцій. Поглибити знання в області частково параметризованих функцій.

**Теоретичні відомості для поглибленого опрацювання**

Тема лекції №4 «Часткова параметризація функцій» має на меті розкриття основних аспектів написання програм з використанням частково параметризованих функцій. Для цього описуються особливості каррінгу, розглядається функціональне представлення даних та позиціонування у списках. На самостійне опрацювання виноситься питання часткового використання функцій, оскільки каррінг часто плутають з частковим використанням і необхідно розуміти різницю.

Для поглиблення знань в області частково параметризованих функцій, а саме в частковому використанні функцій слід особливу увагу звернути на різницю та спільні риси понять каррінг та часткове використання функцій. Необхідно розглянути Часткові (обмежені) функції та механізм реалізації часткового використання функцій в різних мовах програмування та математичні основи цього процесу. Слід також приділити увагу сферам практичного програмування, де застосовується каррінг та часткове використання функцій та розглянути типові користувацькі сценарії, де використовується кожне з явищ.

**Запитання для самостійного опрацювання:**

1. Які спільні та відмінні риси мають каррінг та часткове застосування функцій мовою Haskell?
2. В чому полягають особливості застосування часткового використання функцій?
3. Який зв'язок мають явища каррінг та часткове використання функції?
4. Яким чином реалізується часткове використання функцій мовою програмування Haskell?
5. У яких випадках застосовується часткове використання функцій при вирішенні практичних задач?

**Практичне завдання для самостійної роботи:**

Реалізувати функцію обчислення добутку п’яти натуральних чисел. Застосувати часткове використання функції до 1-го, 2-х, 3-х та 4-х аргументів, які задати за замовчуванням.

# Самостійна робота № 5

# Налагодження програм на Haskell

**Мета** – ознайомитися з процесом налагодження програм на Haskell. Розширити навички використання функторів та монад при програмуванні мовою Haskell. Поглибити знання технологій компіляції програм на Haskell.

**Теоретичні відомості для поглибленого опрацювання**

Тема лекції №5 «Ліниві обчислення. Функтори та монади» має на меті розкриття основних аспектів складу стандартних класів та використання нескінченних структур даних для програмування мовою Haskell. Для цього описуються особливості енергійної та лінивої схеми обчислень, розглядаються нескінченні структури даних, функтори, монади та послідовні обчислення, а також процес введення, виведення і компіляції програм на Haskell. На самостійне опрацювання виноситься питання налагодження програмного коду на Haskell.

Для поглиблення знань в області функціонального програмування на Haskell, а саме в налагодженні програм, слід особливу увагу звернути на різновиди методик налагодження. Необхідно розглянути інструментальні засоби налагодження програм, написаних функціональними мовами взагалі та мовою Haskell зокрема. Слід приділити увагу текстовому представленню даних, що спрощує процедуру налагодження програм, а також на процес віддаленого налагодження.

**Запитання для самостійного опрацювання:**

1. В чому полягає особливість налагодження функціональних програм?
2. Які інструментальні засоби забезпечують процес налагодження програмного коду на Haskell?
3. Які характерні особливості механізму декларативного налагодження програм на Haskell?
4. Як можливо використати традиційну методику налагодження при використанні лінивих обчислень?
5. Які існують методики налагодження програм функціональних мов?

**Практичне завдання для самостійної роботи:**

Використовуючи мову Haskell, реалізувати програма «вгадування» задуманого числа (у вигляді: «Задумайтесь число від 0 до 99 Задумане менше, ніж 50? (Yes, по)». Якщо ні, то «Задумане менше, ніж 75? (Yes, по)» і т.д.). Обрати найбільш вдалий інструмент та методику налагодження. Провести налагодження реалізованої програми.

# Самостійна робота № 6

# Подання програм при розширеному лямбда-численні

**Мета** – ознайомитися з поданням програм при розширеному лямбда-численні. Розширити навички застосування чистого лямбда-числення. Поглибити знання технологій представлення конструкцій мовою програмування Haskell.

**Теоретичні відомості для поглибленого опрацювання**

Тема лекції №6 «Лямбда-числення в аналізі та синтезі інформаційних систем» має на меті розкриття основних аспектів застосування математичного підґрунтя функціональної парадигми програмування. Для цього описуються особливості подання виразів у лямбда-численні, рекурсії в ньому та розглядається чисте лямбда-числення, а також наводяться відомості про нормальну та слабку заголовну нормальну форму. На самостійне опрацювання виноситься питання розширеного лямбда-числення та особливостей подання програм у ньому, як такому, що базується на принципах чистого лямбда-числення та дозволяє закріпити і поглибити знання цих принципів.

Для поглиблення знань в області лямбда-числення, а саме в області розширеного лямбда-числення слід особливу увагу звернути на загальні принципи та основи розширеного лямбда-числення, застосування правил редукції в умовах існування рекурсивних визначень. Також необхідно звернутися до питань класифікації конструкцій в розширеному лямбда-численні (особливо виділяють константи) і можливості зручного представлення конструкцій розширеного лямбда-числення у функціональних програмах

**Запитання для самостійного опрацювання:**

1. Які переваги має розширене лямбда-числення в порівнянні із чистим в контексті вирішення практичних задач?
2. Яким чином можливо більш точно дослідити семантику виконання функціональних програм, використовуючи розширене лямбда-числення?
3. Які види конструкцій існує в розширеному лямбда-численні?
4. В чому полягає особливість вираження конструкцій функціональних мов програмування шляхом застосування розширеного лямбда-числення?
5. Яким чином проводиться опис конструкцій синтаксичного дерева, використовуючи новий тип даних мовою Haskell?

**Практичне завдання для самостійної роботи:**

Необхідно реалізувати функцію, що за лямбда-виразом рекурсивної функції будує нерекурсивну, яка є еквівалентною їй за допомогою Y-комбінатору, що є вбудованою функцією.

# Самостійна робота № 7

# Перетворення графів при виконанні програм

**Мета** – ознайомитися з процесом перетворення графів при виконанні функціональних програм. Розширити навички представлення функціональних програм. Поглибити знання SECD-машини та графічного представлення функціональних програм.

**Теоретичні відомості для поглибленого опрацювання**

Тема лекції №7 «Представлення, інтерпретація та виконання функціональних програм. SECD-машина» має на меті розкриття основних аспектів представлення та виконання функціональних програм на базі мови Haskell. Для цього описуються особливості представлення функціональних програм, розглядається Eval/apply-інтерпретатор Маккарті, архітектура SECD-машини та особливості компіляції функціональних програм у ній. На самостійне опрацювання виноситься питання перетворення графів при виконанні програм, що відноситься до розділу графічного представлення функціональних програм.

Для поглиблення знань в області представлення функціональних програм, а саме в графічному представленні, слід особливу увагу, перш за все звернути на графічне представлення конструкцій розширеного лямбда-числення взагалі. Необхідно розглянути особливості процесу перетворення графів при виконанні функціональних програм. Для вірної побудови графу слід з’ясувати правила виконання редукцій при побудові графа а також розглянути функції-проектори та фіктивні вузли.

**Запитання для самостійного опрацювання:**

1. Чим відрізняється модель виконання функціональних програм у вигляді обчислень на SECD-машині від моделі графічного представлення?
2. Яким чином лямбда-вираз може бути представлений у вигляді графа?
3. В чому сутність правил виконання редукцій при побудові графа та яким чином проходить процес перетворення графу виразу?
4. Яку необхідність зумовлює вирішення проблеми поділу виразів в представленні виразів у вигляді графа?
5. Що являють собою функції-проектори та навіщо утворюються фіктивні вузли?

**Практичне завдання для самостійної роботи:**

Представити Y-комбінатор Каррі в графічному вигляді. Виконати редукції графа.

# Самостійна робота № 8

# Подання рекурсивних функцій при редукції на графах

**Мета** – ознайомитися з правилами подання рекурсивних функцій при редукції на графах. Розширити навички графічного представлення функціональних програм. Поглибити знання в області абстрагування та виявити різницю абстрагування в імперативному та функціональному програмуванні.

**Теоретичні відомості для поглибленого опрацювання**

Тема лекції №8 «Абстрагування у функціональному програмуванні. Довговічність коду» має на меті розкриття основних аспектів абстракції у функціональному програмуванні. Для цього описуються особливості використання імперативних програм, як функцій, розглядається абстрагування від змінних, оптимізація Каррі та збереження аплікативних під виразів при абстрагуванні. На самостійне опрацювання виноситься питання подання рекурсивних функцій при редукції на графах у зв’язку з необхідність засвоєння різниці абстракції в імперативному і функціональному підході програмування, зважаючи на те, що застосування методу процедурної абстракції передбачає використання рекурсивних функцій в імперативному підході.

Для поглиблення знань в області абстрагування та в поданні рекурсивних функцій слід особливу увагу звернути на правила перетворення графів для основних комбінаторів. Крім того, слід розглянути послідовність перетворення рекурсивного блоку в аплікативний вираз, особливості перетворення графа для Y-комбінатора та спосіб уникнення застосування Y-комбінатора, що має на увазі утворення циклічних посилань в графі.

**Запитання для самостійного опрацювання:**

1. Як підвищується ефективність роботи по перетворенню графів?
2. Як можливо позбутися від рекурсії при, якщо вираз містить рекурсивні блоки?
3. Як виглядає правило, що визначає Y-комбінатор?
4. Яким чином операції, необхідні, щоб позбутися рекурсії можливо представити у програмному вигляді?
5. Яким чином можна зобразити графічно правило перетворення графа для Y-комбінатора?

**Практичне завдання для самостійної роботи:**

Функція gcd обчислення найбільшого спільного дільника двох натуральних чисел може бути представлена рівняннями gcd m 0 = m gcd m n = gcd n (m 'mod' n). Зробіть графічне представлення виразу gcd 9 10 і проведіть редукції в отриманому графі до приведення його до нормальної форми.

# ЛІТЕРАТУРА

1. Конспект лекцій з дисципліни «Функціональне та логічне програмування» «Функціональне програмування» ч. 1 для студентів спеціальності 121 «Інженерія програмного забезпечення» освітня програма «Програмне забезпечення систем» усіх форм навчання / уклад.: І. В. Левада. – Запоріжжя : ЗНТУ, 2016. – 90 с.
2. Карнаухов О.К., Карнаухов А.К., Прошина М.Ю. Функціональне програмування у задачах наукових досліджень. Тернопільський національний технічний університет ім. Івана Пулюя , 2011. URL: http://elartu.tntu.edu.ua/handle/123456789/1004
3. Кубенский А.А.Функциональное программирование : учебник и практикум для академического бакалавриата. М. : Издательство Юрайт, 2017. 348 с.
4. Душкин Р.В. Функциональное программирование на языке Haskell. М.: ДМК Пресс, 2006. 608 с.
5. Городняя Л.В. Основы функционального программирования. М.: Интуит, 2016. 247 с.
6. Душкин Р.В. Текст лекций по курсу Функциональное программирование. М.: МИФИ, 2001. 69 с.
7. Роганова Н. Функциональное программирование. МГИУ, 2007. 215 с.
8. Харрисон Д. Введение в функциональное программирование (пер. с англ.). Кембридж, 1997. 170 с.
9. Lemmer R. Haskell Design Patterns. Birmingham: Packt Publishing, 2015. 166 p.
10. Allen C., Moronuki J. Haskell Programming from first principles. New York: Christopher Allen and Julie Moronuki, 2016. 1228 p.
11. Хендерсон П. Функциональное программирование. Применение и реализация: Пер. с англ. М.: Мир, 1983. 349 с.
12. Мена А. Серано. Изучаем Haskell. Библиотека программиста. СПб.: Питер, 2015. 464 с.
13. Заяць В.М. Функціональне програмування: навчальний посібник. Львів: Бескид-Біт, 2003. 160 с.
14. Шалимов П.Ю. Функциональное Программирование на языках Lisp и Erlang: Учебное пособие. Брянск: БГТУ, 2010. 191 с.
15. Душкин Р.В. Квантовые вычисления и функциональное программирование. М.: ДМК Пресс, 2014. 318 с.
16. Зыков С. Современные языки программирования и .NET. Основы функционального программирования и computer science. Учебный курс. М.: Учебный Центр безопасности информационных технологий Microsoft МИФИ (ГУ), 2003. 219 с.
17. Галкін О.В., Верес М.М. Функціональне програмування. Лямбда числення в мовах FP та F#. *Вісник Київського національного університету імені Тараса Шевченка. Серія: фізико-математичні науки*. 2011. Вип. 1. С. 103-107.
18. Миран Л. Изучай Haskell во имя добра! М.: ДМК Пресс, 2012. 490 с.
19. Марлоу С. Параллельное и конкурентное программирование на языке Haskell. М.: ДМК Пресс, 2014. 372 с.
20. Душкин Р.В. 14 занимательных эссе о языке Haskell и функциональном программировании. М.: ДМК Пресс, 2011. 140 с.
21. Джонс С.П. Язык и библиотеки Haskell 98. 2-е изд. испр. НОУ «Интуит», 2016. 323 c.
22. Душкин Р.В. Практика работы на языке Haskell. М.: ДМК Пресс, 2010. 288 с.
23. Bragilevsky V. Haskell in Depth. NY: Manning, 2020. 625 p.
24. O'Sullivan B., Goerzen J., Stewart D. Real World Haskell. Beijing, Cambridge, Farnham, Koln, Sebastopol, Taipei, Tokyo: O'Reilly Media, 2008. 714 p.