

**Федеральное государственное автономное образовательное
учреждение высшего образования
«Московский физико-технический институт
(национальный исследовательский университет)»**

УТВЕРЖДЕНО

**Директор физтех-школы
прикладной математики и
информатики**

А.М. Райгородский

	Рабочая программа дисциплины (модуля)
по дисциплине:	Программирование на языке Rust
по направлению:	Прикладная математика и информатика
профиль подготовки:	Проектирование и разработка комплексных бизнес-приложений Физтех-школа Прикладной Математики и Информатики кафедра алгоритмов и технологий программирования
курс:	3
квалификация:	бакалавр

Семестр, формы промежуточной аттестации: 6 (весенний) - Дифференцированный зачет

Аудиторных часов: 60 всего, в том числе:

лекции: 30 час.

семинары: 30 час.

лабораторные занятия: 0 час.

Самостоятельная работа: 75 час.

Всего часов: 135, всего зач. ед.: 3

Программу составили:

А.Д. Становой, учебный ассистент

В.В. Яковлев, канд. физ.-мат. наук, заведующий кафедрой

Программа обсуждена на заседании кафедры алгоритмов и технологий программирования 12.02.2024

Аннотация

В курсе изучается современное системное программирование на новом языке Rust. Это - естественное продолжение курсов Программирования на C++, Архитектуры Компьютерных и Операционных Систем, и Теории, и Практики Многопоточной Синхронизации, преподаваемых в МФТИ. Рассматривается история появления языка Rust, изучаются основы синтаксиса, стандартная библиотека, трейты, метапрограммирование, параллельные вычисления в языке, асинхронное программирование, тулинг вокруг языка, system safety на основе теории типов, примененных в языке для избежания memory unsafety, а также небезопасное подмножество языка. Активно производится сравнительный анализ с языками C++, Go, Python, Java.

В курс входит несколько десятков задач, простых и углубленных, призванных разобрать на практике разные аспекты данного языка. Также есть 3 проекта, предназначенных для написания цельного продукта на языке.

1. Цели и задачи

Цель дисциплины

- овладение студентами правил языка программирования Rust и приемами использования языка Rust в практике программирования.

Задачи дисциплины

- изучение базовых и продвинутых возможностей языка Rust;
- распространение его среди молодых разработчиков.

2. Перечень формируемых компетенций

Освоение дисциплины направлено на формирование следующих компетенций:

Код и наименование компетенции	Индикаторы достижения компетенции
УК-1 Способен осуществлять поиск, критический анализ и синтез информации, применять системный подход для решения поставленных задач	УК-1.1 Анализирует задачу, выделяя этапы ее решения, действия по решению задачи
	УК-1.2 Находит, критически анализирует и выбирает информацию, необходимую для решения поставленной задачи
	УК-1.3 Рассматривает различные варианты решения задачи, оценивает их преимущества и недостатки
	УК-1.4 Грамотно, логично, аргументированно формирует собственные суждения и оценки
	УК-1.5 Определяет и оценивает практические последствия возможных вариантов решения задачи
ОПК-1 Способен применять фундаментальные знания, полученные в области физико-математических и (или) естественных наук и использовать их в профессиональной деятельности	ОПК-1.1 Способен анализировать поставленную задачу, намечать пути ее решения
	ОПК-1.2 Способен строить математические модели, производить количественные расчеты и оценки
	ОПК-1.3 Способен определять границы применимости полученных результатов
ОПК-2 Способен использовать современные информационные технологии и программные средства при решении задач профессиональной деятельности, соблюдая требования информационной безопасности	ОПК-2.1 Способен применять современные вычислительную технику и сервисы сети Интернет в области (сфере) профессиональной деятельности
	ОПК-2.2 Знает и умеет применять численные математические методы и прикладное программное обеспечение для решения научных задач в профессиональной области
	ОПК-2.3 Знает основные требования информационной безопасности
ОПК-3 Способен составлять и оформлять	ОПК-3.1 Знает основные правила оформления научных публикаций и научно-технической документации, в том числе с использованием прикладного программного обеспечения

научные и (или) технические (технологические, инновационные) отчеты (публикации, проекты)	ОПК-3.2 Владеет на практике методологией составления научно-технических отчетов (проектов)
	ОПК-3.3 Владеет методами визуального и графического представления результатов научной (научно-технической, инновационной технологической) деятельности в виде отчетов, научных публикаций
ПК-1 Способен ставить, формализовывать и решать задачи, в том числе разрабатывать и исследовать математические модели изучаемых явлений и процессов, системно анализировать научные проблемы, получать новые научные результаты	ПК-1.3 Способен применять теоретические и (или) экспериментальные методы исследований к конкретной научной задаче и интерпретировать полученные результаты
	ПК-1.2 Способен выдвигать гипотезы, строить математические модели для описания изучаемых явлений и процессов, оценивать качество разработанной модели
	ПК-1.1 Способен находить, анализировать и обобщать информацию об актуальных результатах исследований в рамках тематической области своей профессиональной деятельности
ПК-2 Способен самостоятельно или в качестве члена (руководителя) малого коллектива организовывать и проводить научные исследования и их апробацию	ПК-2.3 Способен проводить апробацию результатов научно-исследовательской работы посредством публикации научных статей и участия в конференциях
	ПК-2.2 Способен планировать и проводить научные исследования самостоятельно или в качестве члена (руководителя) малого научного коллектива
	ПК-2.1 Знает принципы построения научной работы, методы сбора и анализа полученного материала, способы аргументации

3. Перечень планируемых результатов обучения по дисциплине (модулю)

В результате освоения дисциплины обучающиеся должны

знать:

- принцип исполнения программ на Rust.

уметь:

- реализовывать библиотеку общего назначения по заданным интерфейсам.

владеть:

- навыками работы с объектами и потоками и кругозором в выборе архитектурного решения поставленной задачи.

4. Содержание дисциплины (модуля), структурированное по темам (разделам) с указанием отведенного на них количества академических часов и видов учебных занятий

4.1. Разделы дисциплины (модуля) и трудоемкости по видам учебных занятий

№	Тема (раздел) дисциплины	Трудоемкость по видам учебных занятий, включая самостоятельную работу, час.			
		Лекции	Семинары	Лаборат. работы	Самост. работа
1	Обсуждение языка. Сравнение с C++. Основы языка.	2	2		6
2	Стандартная библиотека и коллекции.	2	2		6
3	Трейты. Функциональные возможности языка. Итераторы.	2	2		6
4	Таблица виртуальных методов. Управление памятью и формальные корни system safety.	2	2		6

5	Метапрограммирование в Rust. Написание идиоматичного кода.	2	4		6
6	Работа с файловой системой. Формальные корни system safety: исследования RustBelt.	2	4		6
7	Пакетный менеджер языка Rust: Cargo. Обработка ошибок.	2	2		6
8	Автоматические средства верификации и поддержки Rust кода.	2	2		5
9	Многопоточное программирование в Rust.	2	2		5
10	Асинхронный Rust и нетворкинг.	2	2		5
11	Unsafe Rust. Репрезентация типов в памяти.	4	2		6
12	Rust и взаимодействие с другими языками и операционной системой.	2	2		6
13	Техники ускорения Rust кода.	4	2		6
Итого часов		30	30		75
Подготовка к экзамену		0 час.			
Общая трудоёмкость		135 час., 3 зач.ед.			

4.2. Содержание дисциплины (модуля), структурированное по темам (разделам)

Семестр: 6 (Весенний)

1. Обсуждение языка. Сравнение с C++. Основы языка.

Почему нам вообще нужен Rust? Какие проблемы Rust решает? Где предполагается использовать Rust? Какие компании уже используют Rust? Где найти работу программистом на Rust? Что значат safe и unsafe. Что значат sound и unsound. Что Rust считает безопасным. Формальная модель RustBelt. Ключевые слова let и mut. Типы i8...i128, u8...u128, isize, usize, f32, f64, bool. Литералы. Shadowing. Ключевое слово as для примитивных кастов. Нетранзитивность кастов. Функции. Ключевое слово return. Выражения и условия. struct. Functional update. self и Self. Ключевое слово impl. Ассоциированные функции. Значение _. enum. size_of enum и дискриминант. std::cmp::Ordering. Сравнение с enum и union из C++. Синтаксис if, while. Именованный break. Синтаксический сахар if/while и let.

2. Стандартная библиотека и коллекции.

Compound типы array и tuple. Что изменяется, когда их размер большой.

Result и его интерфейс.

Option, его интерфейс и оптимизации компилятора.

VecDeque и его интерфейс.

HashMap, HashSet и их интерфейс. Асимптотики.

BTreeMap, BTreeSet и их интерфейс. Почему B-дерево не может быть полной заменой std::map.

LinkedList, BinaryHeap и их интерфейс.

String. Случайный доступ. UTF-8. &str и напоминание о string_view из C++. char и значение понятия Unicode scalar value.

Аллокации на куче: Box, Rc, Cow и их интерфейс. Упоминание Arc.

Почему Rc немутабелен.

Модуль std::cell. Interior mutability. Cell, RefCell. Reentrancy.

std::mem модуль и его безопасные функции: size_of, swap, replace, forget, drop.

Основы Drop checker. Drop flags. Стабильность порядка drop и причины. Порядок инициализации.

Exotically sized types: ZST, DST, Empty. Контейнеры, особенно Vec, когда T - это ZST.

NonNull, NonZero.

print!, println!, eprint!, eprintln!, write!, writeln! и блокирование IO.

BufReader и BufWriter. Их интерфейс и уменьшение числа аллокаций.

3. Трейты. Функциональные возможности языка. Итераторы.

Трейты. Return type polymorphism. Автоматические трейты. Ключевое слово where. Extension traits.

Основные трейты стандартной библиотеки и их возможности. Default, Clone, Copy. Почему о и не генерируются по умолчанию? Ord, PartialOrd. Eq, PartialEq. Hash, Hasher. Drop, ManuallyDrop и идиома RAII. Почему нельзя полагаться на drop order. Deref, DerefMut, Borrow. Index and IndexMut.

Модуль std::ops. Трейты Add, Sub, Mul, Div, Rem, BitAnd, BitOr, BitXor, Shl, Shr и их -Assign варианты. Not, Neg.

Трейты Debug и Display. Formatter. Мотивация их дизайна. Трейт ToString.

Трейты FnOnce, Fn, FnMut. Замыкания. Capture clause. Ключевое слово move. Variable rebinding в отдельной области видимости.

Модуль std::convert. Трейты From и Into, TryFrom и TryInto, AsRef и AsMut. Функция identity.

Ассоциированные типы и константы.

Итераторы. Ленивость итераторов. Трейты Iterator, IntoIterator. Имплементации итераторов в std. Итераторы в рантайме.

API итераторов: map, filter, fold, flatten и другие.

Мотивация дизайна трейта Iterator.

Инваляция итераторов в C++ и Rust.

Итераторы и векторизация. Как вернуть итератор и замыкание из функции: ключевое слово impl.

Полезные функции из модуля std::iter: from_fn, empty, once, repeat, repeat_with.

Трейты FromIterator, ExactSizeIterator, DoubleEndedIterator, Index, IndexMut.

collect, flatten и их имплементация.

4. Таблица виртуальных методов. Управление памятью и формальные корни system safety.

Таблица виртуальных методов. Толстый указатель. Ключевое слово dyn. Динамическая диспетчеризация. Динамическая диспетчеризация на стеке. Dynamically sized types.

Модуль std::any. Трейт Any.

Type coercion и subtyping. Fully Qualified Syntax и когда его нужно использовать.

Object Safety. Generics в таблице виртуальных методов. Hash и инлайнинг.

Как Rust управляет памятью: aliasing и правило “Aliasing XOR Mutability” (AXM).

Borrow checker, affine система типов.

Lifetimes. Именованные ссылки. Lifetime elision. Reborrowing.

Неограниченный 'static lifetime: почему он нам нужен и какое он имеет отношение к остальным лайфтаймам.

Higher-Rank Trait Bounds (HRTB). Variance.

Ключевое слово ref и match. Two phase borrows.

Drop checker. Связь между PhantomData и variance inference.

Оператор точка и правила автоматического вывода типов.

5. Метапрограммирование в Rust. Написание идиоматичного кода.

Generics. Мономорфизация. Статический и динамический полиморфизм.

Trait specialization.

Причина, по которой нет частичной специализации generics: отвратительные последствия SFINAE.

Макросы. macro_rules!. Паттерны, \$crate. Идентификаторы. Гигиеничность. Проблемы макросов. Внутренние макросы.

Основы крейта `serde`.

Атрибуты. `non_exhaustive`, `deprecated`. Макросы `env!`, `option_env!`, `stringify`.

Условная компиляция и крейт `cfg-if`.

Процедурные макросы. `derive`, `cfg`, `test`. `recursion_limit` атрибуты для макросов.

Основы крейта `syn`.

Метапрограммирование. Вычисление констант и ключевое слово `const`. `Const generics`.

Генерация кода макросами.

Паттерны проектирования: `command`, `interpreter`, `newtype idiom`, `strategy`, `visitor`, `builder`, `fold`.

Антипаттерны проектирования: использование `deref polymorphism`.

Советы написания идиоматичного кода.

6. Работа с файловой системой. Формальные корни `system safety`: исследования `RustBelt`.

Работа с файловой системой с модулем `std::fs`. Сравнение дизайна `Rust` и `Go`.

Статья `GhostCell`. Обсуждение силы системы типов и статических проверок в `Rust`.

`Aliasing model`. Статья `stacked borrows`.

7. Пакетный менеджер языка `Rust`: `Cargo`. Обработка ошибок.

`Cargo`. `Crates and modules`. `Compilation unit`. `What's in a crate`. `Coherence`.

Структура `Cargo` пакета. `Cargo.lock`, `semantic versioning`. `Rustup`. `crates.io`. Типы библиотечных крейтов.

`use`, `mod`, `pub`, `super`, `crate`. Где `pub` не работает.

Релизный цикл `Rust`. Сырые идентификаторы. Мигрирование на другую версию.

Обработка ошибок. `Recoverable` и `unrecoverable` ошибки. Паника и `stack unwinding`. `Unwind safety`. `Result<T, E>`. Оператор `?` и устаревший `try!`. Лучшие практики обработки ошибок.

`Exception safety`: минимальный и максимальный уровень.

Управление паникой. `catch_unwind`, `resume_unwind`.

Трейт `Error` и его проблемы.

Основы крейтов `anyhow` и `thiserror`.

8. Автоматические средства верификации и поддержки `Rust` кода.

`Clippy` и линты.

`Rust analyzer`.

MIR интерпретатор `Miri`.

`Rudra`.

Инструменты `Dynamic Symbolic Execution (DSE)`: `Rust verification tools (RVT)`, `Cargo-KLEE`.

Модель чекеры: `Rust Model Checker (RMC)`, `SMACK`

Инструменты верификации: `Haybale`, `Stateright`.

9. Многопоточное программирование в `Rust`.

Модель памяти `Rust`. `Orderings`. `Connection of memory safety and absence of data races`.

Модуль `std::thread`.

Потоки. `Thread builder`. Область видимости и `static`. `thread::scoped` и его проблемы. `Closure scope`. паника в `closure scope`.

Основы крейтов `rayon` и `crossbeam`.

Трейты `Send` и `Sync`. `Unsafe` трейты. `Ord` и `undefined behavior`.

`std::atomic`: `Atomic` и `fence`.

`Arc` и пример `undefined behavior` в `unsafe` коде. `Mutex` и `poisoning`. `RwLock`. `Lazy`.

10. Асинхронный `Rust` и нетворкинг.

Мотивация дизайна асинхронной стороны `Rust`.

async и await.
Stackless coroutines.
Трейт Future. Создание экзекютора.
Pin, Unpin и примеры их использования. PhantomPinned.
Обсуждение устройства различных протоколов нетворкинга.
Модель Open Systems Interconnect (OSI). Напоминание о Ethernet, IP, UDP and TCP.
Нетворкинг в Rust. Модуль std::net. IpAddr, TcpListener, TcpStream и UdpSocket.
Сохранение метрик в Rust.
Основы крейтов tokio и loom.

11. Unsafe Rust. Репрезентация типов в памяти.

Ключевое слово unsafe. Контракт между safe и unsafe кодом. Что умеет unsafe. Когда нам нужен unsafe. Проблема: безопасный mem::forget. mem::transmute. Ключевое слово static. UnsafeCell. Указатели: *const T, *mut T.
MaybeUninit. Оптимизации компилятора, Container<MaybeUninit<T>>> и Container<T>.
Unsafe трейты. Проблемы с safety у BTreeMap и трейта Ord.
WTF-8. PathBuf и Path.
CString и CStr. OsString и OsStr.
Основы крейтов cbindgen и rust-bindgen.
Drop checker. Атрибут may_dangle.
Имплементация split_at_mut.
Обсуждение имплементации Vec.
Обсуждение имплементации Arc и Mutex.
Модуль std::alloc. Функции alloc, alloc_zeroed, dealloc, Layout. GlobalAlloc.
realloc в Rust и C++. Как move работает в Rust и C++. Когда ломается move в C++. Обсуждение крейта moveit. Копирование и клонирование в Rust.
Leaking. Обсуждение крейта once_cell.
Репрезентация типов в памяти. Exotically sized type Extern. enum в памяти. Гарантии Option.
“Разметка” данных: C, transparent, u*, i*, packed, align(n). Порядок полей.

12. Rust и взаимодействие с другими языками и операционной системой.

Секции .data, .rodata, .bss и .text. Структура кучи и стека. Переполнение буфера.
Использование C/C++ кода из Rust. Использование Rust кода из C/C++.
Состояние Rust ABI.
Советы для написания FFI.
Обработка сигналов.
Вызовы методов ядра.
Состояние Rust в ядре Linux.

13. Техники ускорения Rust кода.

Модуль core::arch и SIMD. core::intrinsics и его использование в стандартной библиотеке.
Модуль std::hint.
Link-time Optimization (LTO).
Profile-guided Optimization (PGO).
Профилирование Rust кода.
Инлайнинг кода. Причины и примеры.
dyn против impl когда оба могут быть использованы.
Заметка о HashSet и HashMap. Siphash и альтернативы.
Удаление и уменьшение числа аллокаций и реаллокаций. Профилирование malloc и free.
Проблема коротких векторов. Обсуждение крейтов smallvec и arrayvec.
Ускорение компиляции и линковки больших проектов на Rust.

5. Описание материально-технической базы, необходимой для осуществления образовательного процесса по дисциплине (модулю)

Компьютер и мультимедийное оборудование (проектор, звуковая система).

6.Перечень рекомендуемой литературы

Основная литература

1. Введение в программирование , учебное пособие / И. Ю. Баженова, В. А. Сухомлин. — Москва, ИНТУИТ, 2016.— URL: <https://e.lanbook.com/book/100695> (дата обращения: 30.12.2020). - Полный текст (Режим доступа : из сети МФТИ / Удаленный доступ)

Дополнительная литература

7. Перечень ресурсов информационно-телекоммуникационной сети "Интернет", необходимых для освоения дисциплины (модуля)

Не используются

8. Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине (модулю), включая перечень необходимого программного обеспечения и информационных справочных систем (при необходимости)

Система GitLab для автоматического тестирования домашней работы.

9. Методические указания для обучающихся по освоению дисциплины (модуля)

Студент, изучающий дисциплину, должен с одной стороны, овладеть общим понятийным аппаратом, а с другой стороны, должен научиться применять теоретические знания на практике. Успешное освоение дисциплины требует:

- посещения студентом всех видов аудиторных занятий;
- ведения конспекта в ходе лекционных занятий;
- качественной самостоятельной подготовки к практическим занятиям, активной работы на них;
- активной самостоятельной и аудиторной работы студента;
- своевременной сдачи преподавателю заданий по аудиторным видам работ.

ОЦЕНОЧНЫЕ МАТЕРИАЛЫ ПО ДИСЦИПЛИНЕ (МОДУЛЮ)

по направлению:	Прикладная математика и информатика
профиль подготовки:	Проектирование и разработка комплексных бизнес-приложений Физтех-школа Прикладной Математики и Информатики кафедра алгоритмов и технологий программирования
курс:	<u>3</u>
квалификация:	бакалавр

Семестр, формы промежуточной аттестации: 6 (весенний) - Дифференцированный зачет

Разработчики:

А.Д. Становой, учебный ассистент

В.В. Яковлев, канд. физ.-мат. наук, заведующий кафедрой

1. Компетенции, формируемые в процессе изучения дисциплины

Код и наименование компетенции	Индикаторы достижения компетенции
УК-1 Способен осуществлять поиск, критический анализ и синтез информации, применять системный подход для решения поставленных задач	УК-1.1 Анализирует задачу, выделяя этапы ее решения, действия по решению задачи
	УК-1.2 Находит, критически анализирует и выбирает информацию, необходимую для решения поставленной задачи
	УК-1.3 Рассматривает различные варианты решения задачи, оценивает их преимущества и недостатки
	УК-1.4 Грамотно, логично, аргументированно формирует собственные суждения и оценки
	УК-1.5 Определяет и оценивает практические последствия возможных вариантов решения задачи
ОПК-1 Способен применять фундаментальные знания, полученные в области физико-математических и (или) естественных наук и использовать их в профессиональной деятельности	ОПК-1.1 Способен анализировать поставленную задачу, намечать пути ее решения
	ОПК-1.2 Способен строить математические модели, производить количественные расчеты и оценки
	ОПК-1.3 Способен определять границы применимости полученных результатов
ОПК-2 Способен использовать современные информационные технологии и программные средства при решении задач профессиональной деятельности, соблюдая требования информационной безопасности	ОПК-2.1 Способен применять современные вычислительную технику и сервисы сети Интернет в области (сфере) профессиональной деятельности
	ОПК-2.2 Знает и умеет применять численные математические методы и прикладное программное обеспечение для решения научных задач в профессиональной области
	ОПК-2.3 Знает основные требования информационной безопасности
ОПК-3 Способен составлять и оформлять научные и (или) технические (технологические, инновационные) отчеты (публикации, проекты)	ОПК-3.1 Знает основные правила оформления научных публикаций и научно-технической документации, в том числе с использованием прикладного программного обеспечения
	ОПК-3.2 Владеет на практике методологией составления научно-технических отчетов (проектов)
	ОПК-3.3 Владеет методами визуального и графического представления результатов научной (научно-технической, инновационной технологической) деятельности в виде отчетов, научных публикаций
ПК-1 Способен ставить, формализовывать и решать задачи, в том числе разрабатывать и исследовать математические модели изучаемых явлений и процессов, системно анализировать научные проблемы, получать новые научные результаты	ПК-1.3 Способен применять теоретические и (или) экспериментальные методы исследований к конкретной научной задаче и интерпретировать полученные результаты
	ПК-1.2 Способен выдвигать гипотезы, строить математические модели для описания изучаемых явлений и процессов, оценивать качество разработанной модели
	ПК-1.1 Способен находить, анализировать и обобщать информацию об актуальных результатах исследований в рамках тематической области своей профессиональной деятельности
ПК-2 Способен самостоятельно или в качестве члена (руководителя) малого коллектива организовывать и проводить научные исследования и их апробацию	ПК-2.3 Способен проводить апробацию результатов научно-исследовательской работы посредством публикации научных статей и участия в конференциях
	ПК-2.2 Способен планировать и проводить научные исследования самостоятельно или в качестве члена (руководителя) малого научного коллектива

2. Показатели оценивания компетенций

В результате изучения дисциплины «Программирование на языке Rust» обучающийся должен:

знать:

- принцип исполнения программ на Rust.

уметь:

- реализовывать библиотеку общего назначения по заданным интерфейсам.

владеть:

- навыками работы с объектами и потоками и кругозором в выборе архитектурного решения поставленной задачи.

3. Перечень типовых (примерных) вопросов, заданий, тем для подготовки к текущему контролю

1. Система типов.
2. Числовые типы.
3. Коллекции.
4. Строковые типы.
5. Константы.
6. Управление памятью.
7. Синтаксис языка.
8. Объектная система.
9. Особенности программирования.
10. Сравнение с другими языками.

4. Перечень типовых (примерных) вопросов и тем для проведения промежуточной аттестации обучающихся

1. Реализовать граф объектов на сырых указателях.
2. В файле `src/lib.rs` реализуйте функцию `add`, которая складывает два числа.
3. Напишите несколько функций для работы с итераторами:
`count()` - создаёт итератор, который возвращает по порядку числа от 0 до `u64::MAX`. При вызове `.next()` после того, как вернулся `u64::MAX`, можно паниковать.
4. Напишите несколько функций для работы с итераторами:
`cycle(into_iter)` - создаёт бесконечный итератор, который возвращает за цикленно последовательность элементов, полученную из `into_iter`. Заметьте, что нижележащий итератор должен истощаться лениво, а не сразу при вызове `cycle`. `Item` обязан быть `Clone`.
5. Напишите несколько функций для работы с итераторами:
`extract(into_iter, index)` - возвращает пару, первый элемент которой - `Option<Item>`, где `Item` - элемент итератора под номером `index` (начиная с 0), если таковой есть. Второй элемент пары - итератор, который должен возвращать элементы исходного итератора в той же последовательности, но без извлечённого элемента. Заметьте, что исходный итератор должен быть истощён не более, чем на `index + 1` элемент.
6. Напишите несколько функций для работы с итераторами:
`tee(into_iter)` - возвращает пару из двух независимых итераторов, каждый из которых возвращает ровно ту же последовательность элементов, что исходный итератор. Исходный итератор должен при этом

истощаться лениво: из него всегда должно быть извлечено кол-во элементов, равное максимуму из

кол-ва извлечённых элементов у двух итераторов, которые tee вернул. Item обязан быть Clone.

7. Напишите несколько функций для работы с итераторами:

`group_by(into_iter, f)` - объединяет все идущие подряд элементы, для которых `f` возвращает одинаковое значение, в группы. `group_by` возвращает итератор по парам, где первый элемент - это `f(item)`,

а второй элемент - это вектор из идущих подряд элементов, таких, что `f(item)` для каждого из этих

элементов равен первому элементу пары.

8. Напишите библиотеку ORM (Object-relational mapping).

9. Реализуйте интерпретатор игрового языка программирования Polka.

10. Напишите распараллеленный грер.

Критерии оценивания

Оценка "Отлично" (10) - полностью и вовремя решены все задачи без ошибок. Продemonстрирован грамотный подход к решению задач, реализованы оптимальные алгоритмы, код оформлен в едином удобочитаемом стиле.

Оценка "Отлично" (9) - полностью и вовремя решены все задачи без ошибок. Продemonстрирован грамотный подход к решению задач, реализованы оптимальные алгоритмы.

Оценка "Отлично" (8) - полностью и вовремя решены все задачи без ошибок. Продemonстрирован грамотный подход к решению задач.

Оценка "Хорошо" (7) - полностью решены все задачи. Допущены несущественные ошибки.

Оценка "Хорошо" (6) - полностью решено большинство задач. В некоторых задачах допущены и не исправлены ошибки, либо некоторые задачи решены частично.

Оценка "Хорошо" (5) - полностью решено две трети задач. В некоторых задачах допущены и не исправлены ошибки, либо некоторые задачи решены частично.

Оценка "Удовлетворительно" (4) - полностью решено более половины задач. В остальных задачах допущены и не исправлены ошибки, либо некоторые задачи решены частично.

Оценка "Удовлетворительно" (3) - полностью решено более половины задач.

Оценка "Неудовлетворительно" (2) - решено менее половины задач.

Оценка "Неудовлетворительно" (1) - не решено ни одной задачи.

5. Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности

Дифференцированный зачет может проводиться по итогам текущей успеваемости и сдачи заданий и других видов работ, предусмотренных программой дисциплины и (или) путем организации специального опроса, проводимого в устной и (или) письменной форме.

При проведении устного дифференцированного зачета обучающемуся предоставляется 30 минут на подготовку. Опрос обучающегося не должен превышать одного астрономического часа.

Во время проведения дифференцированного зачета обучающиеся могут пользоваться программой дисциплины, а также справочной литературой, конспектами лекций или другими материалами.