**Правительство Российской Федерации**

**Санкт-Петербургский государственный университет**

**Р А Б О Ч А Я П Р О Г Р А М М А**

**УЧЕБНОЙ ДИСЦИПЛИНЫ**

Структуры и алгоритмы компьютерной обработки данных

Computer Data Processing Algorithms

**Язык(и) обучения**

русский

Трудоемкость в зачетных единицах: 2

Регистрационный номер рабочей программы: 002187

Санкт-Петербург

2020

**Раздел 1. Характеристики учебных занятий**

**1.1. Цели и задачи учебных занятий**

Курс «Структуры и алгоритмы компьютерной обработки данных» является органическим продолжением курса «Программирование» и имеет целью углубить представление обучающихся о рассматриваемых в курсе «Программирование» темах, а также расширить кругозор обучающихся и продемонстрировать им типичные задачи разных разделов теоретической информатики и программной инженерии, чтобы они могли более осознанно выбирать учебные траектории и элективы на третьем курсе.

Также задачей курса является подготовка обучающихся к практической деятельности в рамках стажировок, студенческих проектов и летних школ в период летних каникул, в которых рекомендуется принять участие всем обучающимся.

Отличительной особенностью данного курса является наличие нескольких вариантов реализации, на которые обучающиеся могут распределиться в зависимости от индивидуальных предпочтений и выбранного ими варианта реализации курса «Программирование». Это позволяет обучающимся уже на втором курсе специализироваться, способствует росту вариативности и индивидуализации учебного плана, а также, с учётом горизонтального распространения знаний в студенческой среде, поощряемого групповыми работами, способствует существенному расширению кругозора и суммы знаний у потока в целом.

**1.2. Требования подготовленности обучающегося к освоению содержания учебных занятий (пререквизиты)**

Программа курса предназначена для обучающихся второго курса бакалавриата, прослушавших курс «Программирование» или аналогичные ему. Требуется сформированность хотя бы на базовом уровне компетенций ОПК-2, ОПК-4, ОПК-5, ПКП-4, УКБ-3.

**1.3. Перечень результатов обучения (learning outcomes)**

Результатами обучения являются знания, умения и навыки, соответствующие задачам обучения, а также представление о возможностях применения этих знаний, умений и навыков в профессиональной деятельности.

Курс способствует формированию следующих компетенций:

* ОПК-1 — способен применять фундаментальные знания, полученные в области математических и (или) естественных наук, и использовать их в профессиональной деятельности;
* ОПК-2 — способен применять современный математический аппарат, связанный с проектированием, разработкой, реализацией и оценкой качества программных комплексов в различных областях человеческой деятельности;
* ОПК-3 — способен применять современные информационные технологии, в том числе отечественные, при создании программных продуктов и программных комплексов различного назначения;
* ПКА-1 — способен демонстрировать базовые знания математических и естественных наук, программирования и информационных технологий;
* ПКП-4 — способен применять современные информационные технологии при проектировании, реализации, оценке качества и анализа эффективности программного обеспечения для решения задач в различных предметных областях;
* ПКП-5 — способен использовать основные методы и средства автоматизации проектирования, реализации, испытаний и оценки качества при создании конкурентоспособного программного продукта и программных комплексов, а также способен использовать методы и средства автоматизации, связанные с сопровождением, администрированием и модернизацией программных продуктов и программных комплексов;
* ПКП-6 — способен использовать знания направлений развития компьютеров с традиционной (нетрадиционной) архитектурой; современных системных программных средств: операционных систем, операционных и сетевых оболочек, сервисных программ; тенденции развития функций и архитектур проблемно-ориентированных программных систем и комплексов в профессиональной деятельности;
* ПКП-7 — способен использовать основные концептуальные положения функционального, логического, объектно-ориентированного и визуального направлений программирования, методы, способы и средства разработки программ в рамках этих направлений;
* ПКП-8 — способен использовать современные методы разработки и реализации конкретных алгоритмов математических моделей на базе языков программирования и пакетов прикладных программ моделирования;
* УКБ-3 — способен понимать сущность и значение информации в развитии общества, использовать основные методы получения и работы с информацией с учетом современных технологий цифровой экономики и информационной безопасности.

**1.4. Перечень и объём активных и интерактивных форм учебных занятий**

Основной формой обучения структурам и алгоритмам компьютерной обработки данных являются практические занятия в аудитории, проводимые в активной форме: в форме работы над задачами, в том числе групповой, в форме мозгового штурма, деловых игр, а также в форме активных лекций, предполагающих дискуссию с преподавателем. Также дисциплина предполагает занятия в виде докладов, делаемых обучающимися.

Общий объём активных и интерактивных форм учебных занятий составляет 34 часа (при 30 часах суммарной аудиторной работы, так что, по всей видимости, имеет место ошибка в учебном плане).

**Раздел 2. Организация, структура и содержание учебных занятий**

**2.1. Организация учебных занятий**

**2.1.1 Основной курс**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Трудоёмкость, объёмы учебной работы и наполняемость групп обучающихся | | | | | | | | | | | | | | | | | | |
| Код модуля в составе дисциплины,  практики и т.п. | Контактная работа обучающихся с преподавателем | | | | | | | | | | | | Самостоятельная работа | | | | Объём активных и интерактивных  форм учебных занятий | Трудоёмкость |
| лекции | семинары | консультации | практические  занятия | лабораторные работы | контрольные работы | коллоквиумы | текущий контроль | промежуточная  аттестация | итоговая аттестация | под руководством преподавателя | в присутствии  преподавателя | сам. раб. с использованием  методических материалов | текущий контроль (сам.раб.) | промежуточная аттестация (сам.раб.) | итоговая аттестация  (сам.раб.) |
| ОСНОВНАЯ ТРАЕКТОРИЯ | | | | | | | | | | | | | | | | | | |
| Форма обучения: очная | | | | | | | | | | | | | | | | | | |
| Семестр 1 |  |  |  | 26 |  | 4 |  |  | 2 |  |  |  | 30 |  | 8 |  | 34 | 2 |
|  |  |  |  | 2-15 |  | 2-15 |  |  | 1-15 |  |  |  | 1-1 |  | 1-1 |  |  |  |
| ИТОГО |  |  |  | 26 |  | 4 |  |  | 2 |  |  |  | 30 |  | 8 |  | 34 | 2 |

|  |  |  |  |
| --- | --- | --- | --- |
| Виды, формы и сроки текущего контроля успеваемости и промежуточной аттестации | | | |
| Период обучения (модуль) | Формы текущего контроля успеваемости | Виды промежуточной аттестации | Виды итоговой аттестации  (только для программ итоговой аттестации и дополнительных образовательных программ) |
| ОСНОВНАЯ ТРАЕКТОРИЯ | | | |
| Форма обучения: очная | | | |
| Семестр 1 |  | зачёт |  |

**2.2. Структура и содержание учебных занятий**

Структура и содержание учебных занятий зависят от варианта реализации дисциплины. Обучающимся предлагаются следующие варианты реализации, являющиеся продолжением соответствующих вариантов реализации курса «Программирование»:

1. **Функциональное программирование на F#** (продолжение варианта «Промышленное программирование, C, C#») — первая половина курса посвящена изучению основ функционального программирования на примере языка F#, как ещё одного языка для платформы .NET, с которой обучающиеся познакомились в рамках курса «Программирование». Вторая половина курса рассматривает дополнительные темы промышленного программирования, при этом дополнительные темы также подаются через призму функционального программирования, демонстрируются преимущества (и недостатки) функциональной парадигмы по сравнению с объектно-ориентированной. Выбор F# в качестве языка обучения функциональному программированию довольно нетипичен в силу «идеологической нечистоты» языка, но обусловлен возможностью его практического применения в реальных проектах, тесной интеграцией с платформой .NET (и, соответственно, возможностью использования библиотек, написанных под эту платформу) и тем фактом, что многие особенности языка F# постепенно переносятся и в язык C#, так что, в каком-то смысле, F# можно рассматривать как C# недалёкого будущего.

Курс рассматривает такие темы, как парадигма и основные приёмы функционального программирования, лямбда-исчисление (только нетипизированное), особенности систем типов функциональных языков, монады (вычислительные выражения в F#), особенности взаимодействия объектно-ориентированного и функционального подхода, применение функциональных паттернов для прикладных задач: асинхронное программирование и реактивное программирование, реализованные на уровне стандартной библиотеки, синтаксический анализ с использованием парсер-комбинаторного подхода. Также кратко рассматривается теория формальных языков, есть занятие, посвящённое сборке мусора в платформах с виртуальной машиной на примере .NET, которое напрямую к F# не относится, но необходимо для углубления знаний обучающихся о платформе .NET.

Особенностью курса является большое количество домашних заданий, сдаваемых через системы контроля версий с использованием средств непрерывной интеграции, большое количество теоретического материала, сообщаемого на занятиях. Курс построен так, чтобы постепенно уменьшать самостоятельную работу к концу семестра, чтобы оставить обучающимся время на активную работу над учебной практикой. Также в конце курса выделяется два занятия на доклады по результатам учебных практик, с целью подготовить обучающихся к зачёту по учебной практике, развить у них компетенции, связанные с подготовкой презентаций и расширить их кругозор путём слушания чужих докладов.

1. **...** (продолжение варианта«Прикладное и исследовательское программирование на примере F#») —
2. **Углублённый C++** (продолжение варианта«Промышленное программирование, C#, С++») — данный вариант может преподаваться по различным траекториям, в зависимости от успехов группы в курсе «Программирование». В случае, если в третьем семестре только началось изучение C++, программа данного варианта полностью повторяет углублённую программу изучения C++ третьего семестра программирования. Если в третьем семестре C++ уже изучался достаточно углублённо, в данном варианте рассматривается многопоточное программирование на C++, включая std::thread, примитивы синхронизации из стандартной библиотеки, атомарные операции, понятие thread-safe кода, RAII в многопоточном программировании на C++, conditional-ы, std::future/ std::promise; также уделяется много внимания работе с инструментом valgrind в ОС Linux. Курс ориентирован на выполнение объёмных домашних заданий по многопоточному программированию с обсуждением и проверкой задач прямо на занятиях, может сильно помочь знакомство с книгой A. Williams «C++ Concurrency in Action: Practical Multithreading».
3. **Методы и средства автоматической обработки текстов** (продолжение варианта«Программирование на C и C++»)— в курсе очень подробно рассматриваются такие понятия, как конечные автоматы и формальные грамматики. В первой части курса обсуждаются вопросы проектирования и оптимизации конечных автоматов, во второй – использование понятия формальной грамматики для решения задач по распознаванию и преобразованию текста. Достаточно подробно представлены темы использования контекстно-свободных грамматик для определения принадлежности строки определяемому ими языку (представлены алгоритмы Кока-Янгера-Касами и Эрли), а также построения деревьев разбора.
4. … (продолжение варианта«Промышленное программирование на C, Kotlin») —

Распределение обучающихся по вариантам реализации выполняется прежде всего в соответствии с их выбором варианта реализации курса «Программирование», но возможна смена варианта реализации по инициативе обучающегося и по согласованию с преподавателями.

**Вариант реализации 1: функциональное программирование на F#.**

Период обучения (модуль): семестр 4.

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Наименование темы (раздела, части) | Вид учебных занятий | Количество часов |
| I. | Функциональное программирование | практические занятия | 8 |
| самостоятельная работа | 8 |
| II. | F# как мультипарадигменный язык программирования | практические занятия | 6 |
| контрольные работы | 2 |
| самостоятельная работа | 8 |
| III. | Синтаксический анализ на F# | практические занятия | 4 |
| самостоятельная работа | 4 |
| IV. | Приложения F# | практические занятия | 2 |
| контрольные работы | 2 |
| самостоятельная работа | 6 |
| V. | Особенности платформы .NET | практические занятия | 2 |
| самостоятельная работа | 0 |
| VI. | Доклады по учебным практикам | практические занятия | 4 |
| самостоятельная работа | 4 |
| VII. | Промежуточная аттестация | самостоятельная работа | 8 |
| зачёт | 2 |

Раздел 1: Функциональное программирование.

1. Введение. Функциональное программирование как парадигма. Программирование без изменяемого состояния и явного управления потоком исполнения. Преимущества и недостатки функционального подхода. Язык F#: назначение, инструменты. Структура программы, двухмерный синтаксис, let-определения, функции, автоматический вывод типов, элементарные типы, кортежи, лямбда-функции, списки, Option. *После этого занятия и до раздела 2 домашние работы сдаются на F# без использования мутабельного состояния.*

*Домашняя работа 1.*

1. Модульное тестирование в F#, библиотеки FsUnit, FsCheck, Unquote, Foq. Особенности F#: каррирование, функции высших порядков, операторы конвейера и композиции, сопоставление шаблонов. Последовательности, записи, размеченные объединения, деконструкция, взаимная рекурсия в типах, особенности использования стандартной библиотеки .NET из кода на F#. Приёмы функционального программирования: одноэлементные объединения, хвостовая рекурсия, паттерн «Аккумулятор», Continuation Passing Style.

*Домашняя работа 2.*

1. Нетипизированное лямбда-исчисление: интуиция за лямбда-исчислением, лямбда-исчисление как формальная система: лямбда-нотация, свободные и связанные переменные, подстановка, альфа-, бета- и эта-преобразования, бета-редукция. Редэксы, стратегии редукции. Комбинаторы, комбинатор неподвижной точки, его связь с рекурсией в лямбда-исчислении. Лямбда-исчисление как универсальный вычислитель: булевые выражения, нумералы Чёрча, арифметические операции, пары, примитивная рекурсия, списки.

*Домашняя работа 3.*

1. Генерики в F#. Автоматическое обобщение, встроенные шаблонные операции (генерик-сравнение, генерик-печать, boxing/unboxing). Приёмы обобщения кода, словари операций. Генерики и наследование, приведение типов, гибкие ограничения. Потенциальные проблемы вывода типов и методы их решения, Value Restriction. Point-free-стиль программирования.

*Домашняя работа 4.*

Раздел 2: F# как мультипарадигменный язык программирования.

1. Объектно-ориентированное программирование в F#. Методы у разных типов данных, методы-расширения, статические методы, методы и каррирование, передача параметров в виде кортежа или каррированием, преимущества и недостатки, именованные и опциональные аргументы, перегрузка методов. Классы. Основной конструктор, методы и свойства, модификаторы видимости, мутабельные свойства, автоматические свойства, индексеры, операторы. Дополнительные конструкторы. Наследование, абстрактные классы, реализация по умолчанию, интерфейсы, явное приведение. Наследование интерфейсов, объектные выражения, приёмы, с ними связанные (частичная реализация, делегирование вложенному объектному выражению). Модули и пространства имён. *Начиная с этого занятия в задаваемых после него домашних работах допустимо, хотя и не поощряется, использование мутабельного состояния.*

*Домашняя работа 5.*

1. Вычислительные выражения, что это и зачем нужно, мотивирующий пример. Монадические типы, функции bind и return, пример: Option.bind. Связь с CPS, CPS и let-определения. WorkflowBuilder. Композиция вычислительных выражений, вложенные вычислительные выражения, законы монад. Другие методы WorkflowBuilder, синтаксический сахар. Связь с алгеброй и приёмы композиционального программирования: моноиды, эндоморфизмы, монады как моноиды.

*Домашняя работа 6.*

1. Многопоточное программирование в F#. Монада async, связь с пулом потоков, обработка исключений в async и отмена вычисления. Низкоуровневые примитивы синхронизации, мониторы и функция lock, EventHandle-ы, мьютексы, семафоры. Ручное управление планировщиком. BackgroundWorker. События в F#, реактивное программирование. Атомарные операции, класс Volatile. Модель памяти, понятие relaxed ordering. Класс Interlocked. Введение в lock-free-программирование.

*Домашняя работа 7.*

1. Контрольная работа.

Раздел 3: Синтаксический анализ на F#.

1. Синтаксический анализ вообще. Фазы компиляции, место синтаксического анализа в компиляции программы. Понятие формальной грамматики, иерархия языков Хомского. Вывод в формальных грамматиках, левая рекурсия, неоднозначность вывода. Кратко про алгоритмы разбора: нисходящий (рекурсивный спуск, LL-анализ), восходящий (LR-анализ). Множества FIRST и FOLLOW. Форма Бэкуса-Наура, EBNF. Понятие парсер-комбинаторного подхода.
2. Синтаксический анализ на F#. Реализация интерпретатора арифметических выражений в парсер-комбинаторном стиле с помощью библиотеки FParsec. Представление AST, позитивное замыкание, рекурсивные правила, факторизация грамматики для избавления от левой рекурсии. Промежуточное представление дерева (Parse tree). Приоритет операций. Построение AST по Parse Tree. Реализация того же интерпретатора арифметических выражений с помощью внешнего DSL на примере инструментов FsLex/FsYacc. Подготовка проекта. Описание грамматики в формате FsYacc, описание лексического анализатора в формате FsLex. Пропуск токенов на примере пробелов. Семантические действия. Приоритет операций. Сравнение получившихся решений.

*Домашняя работа 8.*

Раздел 4: Приложения F#.

1. Доклады.
2. Контрольная работа.

Раздел 5: Особенности платформы .NET.

1. Сборка мусора в .NET. Представление объекта в памяти, выделение памяти под объект. Алгоритм mark and sweep. Поколения. Large Object Heap. Режимы сборки мусора, многопоточная сборка. Динамическая настройка сборщика мусора, ручная сборка. Мониторинг сборки мусора. Финализаторы и их взаимодействие со сборщиком мусора. Класс SafeHandle. Детали реализации IDisposable, using, using var. Freachable queue. Ручное управление жизнью объекта, класс GCHandle, ключевое слово fixed. WeakReference.

*Это занятие предполагается не проводить в первую очередь, если фактических часов на реализацию курса меньше, чем предполагается в учебном плане (например, за счёт праздников или эпидемий).*

Раздел 6: Доклады по учебным практикам

1. Представление результатов учебных практик.
2. Представление результатов учебных практик.

В целях оперативной актуализации программы обучения допустимы отклонения от обозначенного здесь плана занятий (вплоть до полной замены темы тех или иных занятий) при условии сохранения общей структуры курса и следования учебному плану.

Домашние работы могут выдаваться несколько до соответствующего им занятия, с целью мотивировать обучающихся самостоятельно искать информацию и прийти на занятие подготовленными (технология «перевёрнутый класс»).

**Вариант реализации 2:**

**Вариант реализации 3: Углублённый C++**

Период обучения (модуль): семестр 4, вариант 1: углублённый C++, основы архитектуры программного обеспечения

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Наименование темы (раздела, части) | Вид учебных занятий | Количество часов |
| I. | Углублённый C++ | практические занятия | 20 |
| контрольные работы | 2 |
| самостоятельная работа | 20 |
| II. | Паттерны проектирования | практические занятия | 6 |
| контрольные работы | 2 |
| самостоятельная работа | 10 |
| III. | Промежуточная аттестация | самостоятельная работа | 8 |
| зачёт | 2 |

Раздел 1: Углублённый C++

1. Стандартная библиотека: ввод-вывод, контейнеры, алгоритмы. Потоки, модификаторы потоков, связывание потоков. Контейнеры стандартной библиотеки. Заголовочный файл algorithm, алгоритмы стандартной библиотеки. Лямбда-выражения. Ассоциативные контейнеры.
2. Динамическая память: умные указатели, аллокаторы. Move-семантика.
3. Перегрузка операторов, std::function, объекты-функции.
4. Шаблоны в C++. Понятие, устройство и синтаксис шаблонов. type\_traits, forward, перегрузка, вариадические шаблоны.
5. Обработка исключений, пространства имён, множественное и виртуальное наследование. Перегрузка new/delete, placement new, RTTI, enum, pointer to member, вложенные и локальные классы, union, битовые поля, volatile, extern c.

Раздел 2: Паттерны проектирования

1. Понятие паттерна проектирования, пример использования паттернов при создании архитектуры программ: разработка текстового редактора. Model-View-Controller как пример паттерна.
2. Доклады по основным паттернам проектирования.

Период обучения (модуль): семестр 4, вариант 2: многопоточное программирование на C++

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Наименование темы (раздела, части) | Вид учебных занятий | Количество часов |
| I. | Базовое многопоточное программирование | практические занятия | 16 |
| контрольные работы | 2 |
| самостоятельная работа | 18 |
| II. | Асинхронное программирование, модель памяти | практические занятия | 10 |
| контрольные работы | 2 |
| самостоятельная работа | 12 |
| III. | Промежуточная аттестация | самостоятельная работа | 8 |
| зачёт | 2 |

Раздел 1: Базовое многопоточное программирование

1. Зачем нужна многопоточность, плюсы и минусы. Простая параллельная программа на C++. Понятие критической секции, гонки, взаимная блокировка, понятие расписания.
2. Использование Valgrind. Параллельные программы с mutex. Взаимные блокировки и способы их избежания: trylock, диспетчер, мьютексы большей гранулярности, замки в правильном порядке. Атомарные операции, thread-safe-код.
3. RAII в многопоточном программировании: концепция, lock\_guard, unique\_lock. conditional-ы, thread pool. Проектирование интерфейсов потокобезопасных абстракций.

Раздел 2: Асинхронное программирование, модель памяти

1. Более подробно conditional variables, проблема сочетания с atomic.
2. Асинхронное программирование на C++: std::future/std::promise.
3. Понятие модели памяти.

**Вариант реализации 4: Методы и средства автоматической обработки текстов.**

Период обучения (модуль): семестр 4.

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Наименование темы (раздела, части) | Вид учебных занятий | Количество часов |
| I. | Конечные автоматы | практические занятия | 14 |
| контрольные работы | 2 |
| самостоятельная работа | 18 |
| II. | Формальные грамматики | практические занятия | 12 |
| контрольные работы | 2 |
| самостоятельная работа | 20 |
| III. | Промежуточная аттестация | самостоятельная работа | 8 |
| зачёт | 2 |

Раздел 1: Конечные автоматы.

1. Детермнированные конечные автоматы.
2. Минимизация детерминированных конечных автоматов.
3. Использование детерминированных конечных автоматов для поиска и замены большого количества фрагментов текста.
4. Недетерминированные конечные автоматы.
5. Автоматическая обработка недетерминированных конечных автоматов.
6. Использование конечных автоматов для лексического анализа. Пример программы, использующей конечный автомат для транслитерации.
7. Средства для автоматического построения лексических анализаторов.
8. Контрольная работа.
9. Переписывание контрольной работы.

Раздел 2: Формальные грамматики.

1. Введение*.*
2. Деревья разбора.
3. Методы построения деревьев разбора.
4. Алгоритм Кока-Янгера-Касами.
5. Алгоритм Эрли.
6. Средства для автоматического построения синтаксических анализаторов.
7. Контрольная работа.
8. Переписывание контрольной работы.

**Вариант реализации 5:**

**Вариант реализации 6:**

**Вариант реализации 7:**

В целях оперативной актуализации программы обучения допустимы отклонения от обозначенных здесь планов занятий (вплоть до полной замены темы тех или иных занятий) при условии сохранения общей структуры курса и следования учебному плану.

**Раздел 3. Обеспечение учебных занятий**

**3.1. Методическое обеспечение**

**3.1.1 Методические указания по освоению дисциплины**

Для успешного усвоения дисциплины необходима самостоятельная работа над материалами курса и домашними заданиями. Сдавать домашние задания нужно своевременно. Также рекомендовано знакомиться с дополнительными материалами (литературой, веб-источниками, онлайн-курсами и т.п.), рекомендуемыми преподавателем.

**3.1.2 Методическое обеспечение самостоятельной работы**

Для самостоятельной работы рекомендуется использовать рекомендованную литературу и материалы курса (презентации, конспекты), размещаемые в системе поддержки обучения.

**3.1.3 Методика проведения текущего контроля успеваемости и промежуточной аттестации и критерии оценивания**

***3.1.3.1. Методика проведения текущего контроля успеваемости и промежуточной аттестации***

В формировании итоговой оценки по курсу участвуют домашние задания, контрольные работы, оценки за работы на практиках (включая доклады). Конкретные критерии оценивания и методики проведения текущего контроля и промежуточной аттестации варьируются в зависимости от варианта реализации дисциплины, чтобы учесть особенности программы и индивидуальные особенности обучающихся. Методики проведения текущего контроля и промежуточной аттестации по вариантам реализации представлены ниже.

**Вариант реализации 1: функциональное программирование на F#.**

Контрольные работы выполняются в аудитории (при этом в исключительных случаях допустимо удалённое проведение контрольной работы), в течение двух академических часов. Обучающимся предлагается от одной до трёх задач, которые необходимо сделать в отведённое время, продемонстрировав навыки использования всех изученных технологий и техник разработки. Обучающимся рекомендуется использовать свою вычислительную технику, в случае её отсутствия предоставляются компьютеры компьютерных классов. Использовать любые источники на контрольной разрешается, общаться (в том числе, с использованием электронных средств коммуникации) — нет.

Домашние работы выполняются индивидуально, при их выполнении разрешается пользоваться любыми источниками и задавать вопросы преподавателю, однако запрещается непосредственно заимствовать чужой код. Выполненные работы сдаются через систему поддержки электронного обучения в виде ссылки на публичный репозиторий в одном из облачных хостингов систем контроля версий (например, GitHub). Домашние работы состоят из нескольких задач, допустимо сдавать задачи отдельно. Каждая задача из домашней работы имеет мягкие ограничения по времени выполнения и по количеству попыток (устанавливаемые для каждой работы отдельно), при превышении лимитов уменьшаются максимальные баллы за задачи. После проверки обучающимся выдаётся список замечаний и текущий балл за задачу, обучающиеся вправе исправить замечания и сделать ещё одну попытку сдачи, после чего процесс проверки, выдачи замечаний и исправления повторяется, либо до достижения максимального балла за задачу, либо удовлетворения обучающегося текущими баллами.

При выявлении заимствований кода в домашних или контрольных работах задача не засчитывается полностью (ставится 0 баллов), в случае обнаружения похожих фрагментов кода у двух обучающихся задача полностью не засчитывается обоим обучающимся. Что считать похожими фрагментами кода, определяет преподаватель.

В случае, если на момент конца семестра максимальное возможное количество баллов не позволяет обучающемуся получить зачёт даже при условии выполнения всех заданий, такому обучающемуся выдаются дополнительные задачи с числом баллов, достаточным, чтобы в случае успешного их решения быть аттестованным. В случае, если к моменту промежуточной аттестации или пересдачи такая ситуация возникает снова, выдаются ещё задачи.

Темы докладов доводятся до сведения обучающихся не позднее чем за две недели до даты докладов, после чего обучающиеся могут указать темы докладов, которые им интересны (на это выделяется один-два дня). Если одну тему указало больше одного обучающегося, докладчик назначается случайно. На доклад отводится 15 минут, предполагается выступление с заранее подготовленными слайдами и, если это необходимо, демонстрацией.

Помимо докладов и контрольных, некоторые занятия включают в себя задания, за которые в случае успешного выполнения можно получить небольшие мотивирующие баллы. Подобные задания могут быть как индивидуальными, так и командными. Форму заданий и критерии оценивания определяет преподаватель согласно текущим педагогическим целям (например, допускается поощрить +1 баллом выход к доске, если нет желающих), однако суммарное количество полученных таким образом баллов для каждого обучающегося не может превосходить 20% от максимально возможного количества баллов за семестр.

**Вариант реализации 2:**

**Вариант реализации 3: Углублённый C++**

Контрольные работы выполняются в аудитории (при этом в исключительных случаях допустимо удалённое проведение контрольной работы), в течение двух академических часов. Обучающимся предлагается от одной до трёх задач, которые необходимо сделать в отведённое время, продемонстрировав навыки использования всех изученных технологий и техник разработки. Обучающимся рекомендуется использовать свою вычислительную технику, в случае её отсутствия предоставляются компьютеры компьютерных классов. Использовать любые источники на контрольной разрешается, общаться (в том числе, с использованием электронных средств коммуникации) — нет.

Домашние работы выполняются индивидуально, при их выполнении разрешается пользоваться любыми источниками и задавать вопросы преподавателю, однако запрещается непосредственно заимствовать чужой код. Выполненные работы сдаются в виде ссылки на публичный репозиторий в одном из облачных хостингов систем контроля версий (например, GitHub). Домашние работы могут состоять из нескольких задач, допустимо сдавать задачи отдельно. После проверки обучающимся выдаётся список замечаний и текущий балл за задачу, обучающиеся вправе исправить замечания и сделать ещё одну попытку сдачи.

При выявлении заимствований кода в домашних или контрольных работах задача не засчитывается полностью (ставится 0 баллов), в случае обнаружения похожих фрагментов кода у двух обучающихся задача полностью не засчитывается обоим обучающимся. Что считать похожими фрагментами кода, определяет преподаватель.

В случае, если на момент конца семестра максимальное возможное количество баллов не позволяет обучающемуся получить зачёт даже при условии выполнения всех заданий, такому обучающемуся выдаются дополнительные задачи с числом баллов, достаточным, чтобы в случае успешного их решения быть аттестованным. В случае, если к моменту промежуточной аттестации или пересдачи такая ситуация возникает снова, выдаются ещё задачи.

Темы докладов доводятся до сведения обучающихся не позднее чем за две недели до даты докладов. На доклад отводится 15 минут, предполагается выступление с заранее подготовленными слайдами и, если это необходимо, демонстрацией.

**Вариант реализации 4: Методы и средства автоматической обработки текстов.** Текущий контроль успеваемости реализуется при помощи двух контрольных работ, предлагаемых студентам в конце каждого раздела. На контрольной работе допустимо пользоваться дополнительными источниками информации, кроме общения между собой и вопросов на форумах. В случае выявления такого общения, равно как и заимствований из работ других обучающихся, за контрольную работу в целом ставится 0 баллов, причем в случае заимствований 0 баллов ставится всем обучающимся, в тексте работ которых обнаружены совпадающие фрагменты. Обычно, контрольная работа содержит от одной до пяти задач каждому обучающемуся индивидуально, которые нужно решить за отведенное время (2 часа). Каждая отдельная задача имеет определенное, зависящее от нее максимальное количество баллов. В некоторых контрольных работах решения последующих задач могут зависеть от решения предыдущих.

**Вариант реализации 5:**

**Вариант реализации 6:**

**Вариант реализации 7:**

***3.1.3.2. Критерии оценивания итогового процента освоения дисциплины***

Критерии оценивания итогового процента освоения дисциплины варьируются в зависимости от варианта реализации дисциплины, чтобы учесть особенности программы и индивидуальные особенности обучающихся. Критерии по вариантам реализации представлены ниже.

**Вариант реализации 1: функциональное программирование на F#.**

Итоговый процент освоения дисциплины складывается из баллов за домашние задания, баллов за контрольные работы, баллов за работу в аудитории, включая доклады.

Домашние задания оцениваются по шкалам и критериям, индивидуальным для каждой задачи. Критерии и максимальные баллы приведены в разделе 3.1.4 и доводятся до обучающихся вместе с условием задачи. Результирующая оценка находится в диапазоне от 0 до 100 и вычисляется по формуле MAX(0, (n/N – 0.6)) \* 2.5 \* 100, где n – суммарный балл, набранный обучающимся на момент аттестации, N – максимально возможный суммарный балл за семестр по домашним заданиям. Например, обучающийся, успешно сдавший 60% заданий, получает 0 баллов, 80% – 50, 100% – 100 баллов.

Каждое задание имеет срок выполнения (по умолчанию 2 недели, но в разделе 3.1.4 может указываться другой срок для конкретных заданий), при пропуске срока вычитается 0.5 балла за каждую неделю задержки (начиная с -0.5 при первом пропуске срока) из максимального балла за каждую задачу из домашней работы. Каждая задача также имеет фиксированное количество «бесплатных» попыток сдачи (по умолчанию две), после которых каждая следующая попытка сдачи приводит к вычитанию 0.5 балла из максимального балла за задачу. При этом исправления замечаний должны быть сданы за неделю после получения замечаний, иначе за каждую неделю пропуска срока начиная с первой вычитается 0.5 балла из максимального балла. Попытка сдачи, в которой не реализованы все требования условия, уменьшает максимальный балл за задачу пропорционально объёму нереализованных требований. Штрафы к максимальному баллу, полученные таким образом, никогда не могут быть меньше текущего балла, при этом текущий балл никогда не может уменьшаться. В случае, если максимальный балл упал до уровня текущего, балл за задачу фиксируется и дальнейшие исправления к ней не принимаются. Штрафы, полученные таким образом, не могут уменьшить максимальный балл за задачу более чем на 50% от его исходного значения (то есть за задачу всегда можно получить половину баллов, если она полностью сделана).

Например, положим, есть задача, максимальный балл за которую составляет 10 баллов. Положим, обучающийся А сдал сразу правильное решение задачи, но за день после установленного срока сдачи, он получает 9.5 баллов. Положим, обучающийся Б сдал задачу с третьей попытки, при этом первая попытка была сделана в срок, вторая — через две недели и один день с момента получения замечаний к первой попытке, третья — в течение трёх дней после получения замечаний ко второй. Обучающийся Б получает 8.5 баллов (-1 за пропуск сроков исправления второй попытки, -0.5 за третью попытку).

На контрольных работах каждая задача оценивается от 0 до 10 баллов, где 0 — полное отсутствие решения, 10 — правильное решение, демонстрирующее все ожидаемые навыки. Баллы за задачи суммируются и итоговый балл за все контрольные в семестре вычисляется по формуле n/N \* 100, где n – суммарный балл, набранный обучающимся на момент аттестации, N – максимально возможный суммарный балл за семестр по контрольным. На переписывании обучающимся предлагается решить другие задачи взамен тех, которые решены не на максимальные баллы на предыдущей попытке, при этом существует однозначное соответствие между задачей переписывания и исходной задачей, определяемое номером задачи. Например, если обучающийся решил на контрольной задачу 1 на 10 баллов, задачу 2 на 8 баллов и задачу 3 на 2 балла, то на переписывании он может решать из предложенных только задачи под номерами 2 и 3, при этом задачу 2 на переписывании он должен написать не хуже, чем на 8 баллов. Если балл, полученный на переписывании, меньше, чем балл предыдущей попытки, то используется максимальный из баллов (то есть баллы при переписывании «не сгорают»).

Критерии оценивания каждой задачи (и домашних, и контрольных) предполагают использование всего пройденного материала.

Доклады оцениваются по шкале от 0 до 10, где 0 — отсутствие доклада, 10 — очень хороший доклад, полностью раскрывающий тему, хорошо представленный и оформленный.

Для вычисления итогового процента освоения дисциплины оценки за работу в аудитории и оценки за доклады прибавляются к оценке за домашние работы (не меняя при этом максимальный возможный балл за домашние работы, так что возможно получение балла за домашние работы более 100%). Далее пересчитанные таким образом баллы за домашние работы и баллы за контрольные приводятся к шкалам от 0 до 100 по приведённым выше формулам и берётся минимум из получившихся оценок. Этот минимум и становится итоговым процентом освоения дисциплины.

**Вариант реализации 2: …**

**Вариант реализации 3: Углублённый C++**

Итоговый процент освоения дисциплины складывается из баллов за домашние задания, баллов за контрольные работы, баллов за доклады.

Домашние задания оцениваются по шкалам и критериям, индивидуальным для каждой задачи. Критерии и максимальные баллы приведены в разделе 3.1.4 и доводятся до обучающихся вместе с условием задачи. Результирующая оценка находится в диапазоне от 0 до 100 и вычисляется по формуле MAX(0, (n/N – 0.6)) \* 2.5 \* 100, где n – суммарный балл, набранный обучающимся на момент аттестации, N – максимально возможный суммарный балл за семестр по домашним заданиям. Например, обучающийся, успешно сдавший 60% заданий, получает 0 баллов, 80% – 50, 100% – 100 баллов.

Каждое задание имеет срок выполнения (по умолчанию 2 недели, но в разделе 3.1.4 может указываться другой срок для конкретных заданий), при пропуске срока вычитается 0.5 балла за каждую неделю задержки (начиная с -0.5 при первом пропуске срока) из максимального балла за каждую задачу из домашней работы. Каждая задача также имеет фиксированное количество «бесплатных» попыток сдачи (по умолчанию две), после которых каждая следующая попытка сдачи приводит к вычитанию 0.5 балла из максимального балла за задачу. При этом исправления замечаний должны быть сданы за неделю после получения замечаний, иначе за каждую неделю пропуска срока начиная с первой вычитается 0.5 балла из максимального балла. Попытка сдачи, в которой не реализованы все требования условия, уменьшает максимальный балл за задачу пропорционально объёму нереализованных требований. Штрафы к максимальному баллу, полученные таким образом, никогда не могут быть меньше текущего балла, при этом текущий балл никогда не может уменьшаться. В случае, если максимальный балл упал до уровня текущего, балл за задачу фиксируется и дальнейшие исправления к ней не принимаются. Штрафы, полученные таким образом, не могут уменьшить максимальный балл за задачу более чем на 50% от его исходного значения (то есть за задачу всегда можно получить половину баллов, если она полностью сделана).

Например, положим, есть задача, максимальный балл за которую составляет 10 баллов. Положим, обучающийся А сдал сразу правильное решение задачи, но за день после установленного срока сдачи, он получает 9.5 баллов. Положим, обучающийся Б сдал задачу с третьей попытки, при этом первая попытка была сделана в срок, вторая — через две недели и один день с момента получения замечаний к первой попытке, третья — в течение трёх дней после получения замечаний ко второй. Обучающийся Б получает 8.5 баллов (-1 за пропуск сроков исправления второй попытки, -0.5 за третью попытку).

На контрольных работах каждая задача оценивается от 0 до 10 баллов, где 0 — полное отсутствие решения, 10 — правильное решение, демонстрирующее все ожидаемые навыки. Баллы за задачи суммируются и итоговый балл за все контрольные в семестре вычисляется по формуле n/N \* 100, где n – суммарный балл, набранный обучающимся на момент аттестации, N – максимально возможный суммарный балл за семестр по контрольным. На переписывании обучающимся предлагается решить другие задачи взамен тех, которые решены не на максимальные баллы на предыдущей попытке, при этом существует однозначное соответствие между задачей переписывания и исходной задачей, определяемое номером задачи. Например, если обучающийся решил на контрольной задачу 1 на 10 баллов, задачу 2 на 8 баллов и задачу 3 на 2 балла, то на переписывании он может решать из предложенных только задачи под номерами 2 и 3, при этом задачу 2 на переписывании он должен написать не хуже, чем на 8 баллов. Если балл, полученный на переписывании, меньше, чем балл предыдущей попытки, то используется максимальный из баллов (то есть баллы при переписывании «не сгорают»).

Критерии оценивания каждой задачи (и домашних, и контрольных) предполагают использование всего пройденного материала.

Доклады оцениваются по шкале от 0 до 10, где 0 — отсутствие доклада, 10 — очень хороший доклад, полностью раскрывающий тему, хорошо представленный и оформленный.

Для вычисления итогового процента освоения дисциплины оценки за доклады прибавляются к оценке за домашние работы (не меняя при этом максимальный возможный балл за домашние работы, так что возможно получение балла за домашние работы более 100%). Далее пересчитанные таким образом баллы за домашние работы и баллы за контрольные приводятся к шкалам от 0 до 100 по приведённым выше формулам и берётся минимум из получившихся оценок. Этот минимум и становится итоговым процентом освоения дисциплины.

**Вариант реализации 4: Методы и средства автоматической обработки текстов.** Общая оценка степени усвоения дисциплины складывается из результатов контрольных работ (50%, вклад каждой задачи пропорционален максимально возможному количеству баллов за нее) и общему количеству баллов за устный зачет (50%), складывающемуся из оценок за ответы на вопросы зачета (как и в случае контрольных работ, максимальное количество баллов за ответ на разные вопросы может быть разным, и вклад соответствующих оценок в общую сумму пропорционален максимуму баллов за ответ на вопрос). Кроме того, за пропуск каждого занятия могут начисляться отрицательные баллы по усмотрению преподавателя.

**Вариант реализации 5: …**

**Вариант реализации 6: …**

**Вариант реализации 7: …**

Вне зависимости от варианта реализации дисциплины перед вычислением итогового процента освоения баллы за дисциплину приводятся к диапазону 0–100 линейным преобразованием. Далее применяется следующее правило выставления оценки:

|  |  |  |
| --- | --- | --- |
| Итоговый процент  выполнения, % | Оценка СПбГУ при  проведении зачёта | Оценка  ECTS |
| 90-100 | зачтено | A |
| 80-89 | зачтено | B |
| 70-79 | зачтено | C |
| 61-69 | зачтено | D |
| 50-60 | зачтено | E |
| менее 50 | не зачтено | F |

**3.1.4 Методические материалы для проведения текущего контроля успеваемости и промежуточной аттестации (контрольно-измерительные материалы, оценочные средства)**

***3.1.4.1. Формируемые дисциплиной компетенции***

***Компетенции, впервые формируемые дисциплиной:***

**ОПК-3** — способен применять современные информационные технологии, в том числе отечественные, при создании программных продуктов и программных комплексов различного назначения.

Шкала оценивания: линейная, определяется долей успешно выполненных заданий, проверяющих данную компетенцию.

**ПКП-5** — способен использовать основные методы и средства автоматизации проектирования, реализации, испытаний и оценки качества при создании конкурентоспособного программного продукта и программных комплексов, а также способен использовать методы и средства автоматизации, связанные с сопровождением, администрированием и модернизацией программных продуктов и программных комплексов.

Шкала оценивания: линейная, определяется долей успешно выполненных заданий, проверяющих данную компетенцию.

**ПКП-6** — способен использовать знания направлений развития компьютеров с традиционной (нетрадиционной) архитектурой; современных системных программных средств: операционных систем, операционных и сетевых оболочек, сервисных программ; тенденции развития функций и архитектур проблемно-ориентированных программных систем и комплексов в профессиональной деятельности.

Шкала оценивания: линейная, определяется долей успешно выполненных заданий, проверяющих данную компетенцию.

**ПКП-7** — способен использовать основные концептуальные положения функционального, логического, объектно-ориентированного и визуального направлений программирования, методы, способы и средства разработки программ в рамках этих направлений.

Шкала оценивания: линейная, определяется долей успешно выполненных заданий, проверяющих данную компетенцию.

**ПКП-8** — способен использовать современные методы разработки и реализации конкретных алгоритмов математических моделей на базе языков программирования и пакетов прикладных программ моделирования.

Шкала оценивания: линейная, определяется долей успешно выполненных заданий, проверяющих данную компетенцию.

***Компетенции, развиваемые дисциплиной:***

**ОПК-1** — способен применять фундаментальные знания, полученные в области математических и (или) естественных наук, и использовать их в профессиональной деятельности.

Шкала оценивания: линейная, определяется долей успешно выполненных заданий, проверяющих данную компетенцию.

**ОПК-2** — способен применять современный математический аппарат, связанный с проектированием, разработкой, реализацией и оценкой качества программных комплексов в различных областях человеческой деятельности;

Шкала оценивания: линейная, определяется долей успешно выполненных заданий, проверяющих данную компетенцию.

**ПКА-1** — способен демонстрировать базовые знания математических и естественных наук, программирования и информационных технологий;

Шкала оценивания: линейная, определяется долей успешно выполненных заданий, проверяющих данную компетенцию.

**ПКП-4** — способен применять современные информационные технологии при проектировании, реализации, оценке качества и анализа эффективности программного обеспечения для решения задач в различных предметных областях;

Шкала оценивания: линейная, определяется долей успешно выполненных заданий, проверяющих данную компетенцию.

**УКБ-3** — способен понимать сущность и значение информации в развитии общества, использовать основные методы получения и работы с информацией с учетом современных технологий цифровой экономики и информационной безопасности.

Шкала оценивания: линейная, определяется долей успешно выполненных заданий, проверяющих данную компетенцию.

***Компетенции, полностью сформированные по результатам освоения дисциплины: нет***

***3.1.4.2. Контрольно-измерительные материалы (примеры)***

Контрольно-измерительные материалы различны для каждого варианта реализации дисциплины.

**Вариант реализации 1: функциональное программирование на F#.**

***Домашняя работа 1:***

1. Посчитать факториал.
2. Посчитать числа Фибоначчи (за линейное время).
3. Реализовать функцию обращения списка (за линейное время).
4. Реализовать функцию, которая принимает на вход n и m и возвращает список из элементов [2n; 2n + 1; ...; 2n + m].
5. Реализовать функцию, которая выдает первую позицию вхождения заданного числа в список.

***Проверяемые компетенции*:** ОПК-3, ПКА-1, ПКП-4, ПКП-6, ПКП-7, ПКП-8

***Критерии оценивания*:** решение каждой задачи оценивается по шкале от 0 (нет решения или решение имеет существенные недостатки) до 1 (решение работоспособно, аккуратно реализовано, алгоритмически аккуратно).

***Домашняя работа 2:***

1. Реализовать три варианта функции, подсчитывающей количество четных чисел в списке (с использованием стандартных функций map, filter, fold). Использование рекурсии не допускается, зато нужен FsCheck для проверки функций на эквивалентность
2. Реализовать функцию, применяющую функцию к каждому элементу двоичного дерева и возвращающую новое двоичное дерево, каждый элемент которого — результат применения функции к соответствующему элементу исходного дерева (map для деревьев)
3. Посчитать значение дерева разбора арифметического выражения, заданного через вложенные discriminated union-ы
4. Реализовать функцию, генерирующую бесконечную последовательность простых чисел

Ко всем задачам обязательны комментарии, юнит-тесты и CI.

***Проверяемые компетенции*:** ОПК-3, ПКА-1, ПКП-4, ПКП-5, ПКП-6, ПКП-7, ПКП-8

***Критерии оценивания*:** решение каждой задачи оценивается по шкале от 0 (нет решения или решение имеет существенные недостатки) до 1 (решение работоспособно, аккуратно реализовано, алгоритмически аккуратно).

***Домашняя работа 3:***

1. Выполнить бета-редукцию λ-терма ((λa.(λb.b b) (λb.b b)) b) ((λc.(c b)) (λa.a)). Решение (в виде последовательности преобразований) набрать в TeX-е и выложить файл .tex на GitHub. Может помочь <https://www.papeeria.com/> или <https://www.overleaf.com/>.
2. Докажите, что S K K = I. Доказательство набрать в TeX-е и выложить на GitHub. Делать надо самостоятельно, это известное (и несложное) упражнение, так что гуглить нечестно.
3. Реализовать интерпретатор лямбда-выражений, выполняющий бета-редукцию по нормальной стратегии. Лямбда-выражения задаются через размеченные объединения. Должна поддерживаться альфа-конверсия для избежания захвата свободных переменных. Примечание: если не извращаться и делать всё по определению, задача простая.

***Проверяемые компетенции*:** ОПК-1, ОПК-2, ОПК-3, ПКА-1, ПКП-4, ПКП-5, ПКП-6, ПКП-7, ПКП-8

***Критерии оценивания*:** решение задач 1 и 2 оценивается по шкале от 0 (нет решения или решение имеет существенные недостатки) до 1 (решение правильно и аккуратно свёрстано в TeX). Решение задачи 3 оценивается по шкале от 0 (нет решения или решение имеет существенные недостатки) до 3 (решение алгоритмически правильно, корректно обрабатывает случай захвата переменных, аккуратно реализовано и покрыто тестами).

***Домашняя работа 4:***

1. Реализовать функцию, которая по произвольной строке проверяет корректность скобочной последовательности в этой строке. Скобки бывают трёх видов.
2. Записать в point-free стиле «func x l = List.map (fun y -> y \* x) l». Выписать шаги вывода и проверить с помощью FsCheck корректность результата.
3. Написать программу — телефонный справочник. Она должна уметь хранить имена и номера телефонов, в интерактивном режиме осуществлять следующие операции:
   1. выйти;
   2. добавить запись (имя и телефон);
   3. найти телефон по имени;
   4. найти имя по телефону;
   5. вывести всё текущее содержимое базы;
   6. сохранить текущие данные в файл;
   7. считать данные из файла.

При этом бизнес-логика должна быть отделена от пользовательского интерфейса и покрыта тестами.

***Проверяемые компетенции*:** ОПК-3, ПКА-1, ПКП-4, ПКП-5, ПКП-6, ПКП-7, ПКП-8

***Критерии оценивания*:** решение каждой задачи оценивается по шкале от 0 (нет решения или решение имеет существенные недостатки) до 2 (решение работоспособно, аккуратно реализовано, алгоритмически аккуратно).

***Домашняя работа 5:***

1. В объектно-ориентированном стиле смоделировать работу локальной сети:

* в сети есть несколько компьютеров, связанных друг с другом (каким образом — можно задавать, например, матрицей смежности);
* на каждом компе стоит ОС (Windows, Linux, etc...);
* в сети гуляют вирусы, так что для каждой машины есть ненулевая вероятность заразиться (вероятность зависит от типа ОС), заражаются компьютеры, непосредственно соединённые с заражёнными;
* заражения (и проверки, заразился компьютер или нет) происходят дискретно — по ходам.

Требуется периодически выводить состояние сети.

Необходимы также юнит-тесты, проверяющие корректность работы алгоритма заражения: если вероятность заражения всегда 1, вирус должен вести себя как обход в ширину, если вероятность 0 — никто не должен заражаться. Могут быть полезны mock-объекты.

***Проверяемые компетенции*:** ОПК-3, ПКА-1, ПКП-4, ПКП-5, ПКП-6, ПКП-7, ПКП-8

***Критерии оценивания*:** решение задачи оценивается по шкале от 0 (нет решения или решение имеет существенные недостатки) до 3 (решение работоспособно, аккуратно реализовано, алгоритмически аккуратно, хорошо покрыто тестами).

***Домашняя работа 6:***

1. Реализовать Workflow, выполняющий математические вычисления с заданной (как аргумент Builder-а) точностью. Например,

rounding 3 {

let! a = 2.0 / 12.0

let! b = 3.5

return a / b

}

должно возвращать 0.048

1. Реализовать Workflow, выполняющий вычисления с числами, заданными в виде строк. Например,

let result = calculate {

let! x = "1"

let! y = "2"

let z = x + y

return z

}

должно возвращать значение, содержащее 3, а

let res = calculate {

let! x = "1"

let! y = "Ъ"

let z = x + y

return z

}

должно возвращать значение, указывающее на отсутствие результата.

***Проверяемые компетенции*:** ОПК-3, ПКА-1, ПКП-4, ПКП-5, ПКП-6, ПКП-7, ПКП-8

***Критерии оценивания*:** решение всех задач оценивается по шкале от 0 (нет решения или решение имеет существенные недостатки) до 1 (решение работоспособно, аккуратно реализовано, алгоритмически аккуратно).

***Домашняя работа 7:***

1. Переделать на F# задачу Lazy из предыдущего семестра (с некоторой дополнительной функциональностью), то есть реализовать интерфейс, представляющий ленивое вычисление:

type ILazy<'a> =

abstract member Get: unit -> 'a

Объект Lazy создаётся на основе вычисления (представляемого лямбда-функцией supplier : unit -> 'a).

* Первый вызов Get() вызывает вычисление и возвращает результат.
* Повторные вызовы Get() возвращают тот же объект, что и первый вызов.
* В однопоточном режиме вычисление должно запускаться не более одного раза, в многопоточном — как получится (см. далее).

Создавать объекты надо не вручную, а с помощью класса LazyFactory, имеющего три метода наподобие

static member CreateSingleThreadedLazy supplier

возвращающих три разные реализации Lazy<'a>:

* простая версия с гарантией корректной работы в однопоточном режиме (без синхронизации);
* гарантия корректной работы в многопоточном режиме; вычисление не должно производиться более одного раза (что-то наподобие многопоточного синглтона, без лишних блокировок);
* то же, что и предыдущее, но lock-free; вычисление может производиться более одного раза, но при этом Lazy.Get всегда должен возвращать один и тот же объект (то есть результаты «лишних» вычислений должны теряться).

1. Написать функцию, принимающую адрес веб-страницы, скачивающую все веб-страницы, на которые есть ссылки с указанной, и печатающую информацию о размере каждой в формате "адрес страницы — число символов". Ссылки нужно обрабатывать только заданные в форме <a href="http://...">. Для поиска ссылок на странице могут быть полезны регулярные выражения (класс System.Text.RegularExpressions.Regex). Качать страницы надо параллельно.

***Проверяемые компетенции*:** ОПК-3, ПКА-1, ПКП-4, ПКП-5, ПКП-6, ПКП-7, ПКП-8

***Критерии оценивания*:** решение задачи 1 оценивается по шкале от 0 (нет решения или решение имеет существенные недостатки) до 3 (решение работоспособно, аккуратно реализовано, отсутствуют ошибки многопоточного программирования и не делается лишних синхронизаций), решение задачи 2 оценивается по тем же критериям по шкале от 0 до 2.

***Домашняя работа 8:***

1. Разработать синтаксический анализатор для интерпретатора лямбда-выражений из домашней работы 3, с дополнительной функциональностью: возможностью именовать лямбда-выражения. Интерпретатор должен принимать на вход строку или файл (должно поддерживаться и то и другое) с набором именованных определений и лямбда-выражением, в котором именованные определения используются. Должно выдаваться в виде строки лямбда-выражение, полученное бета-редукцией входного лямбда-выражения. Лямбда во входных данных обозначается символом \, именованное определение начинается с let и может быть только одно на строке. Должны поддерживаться лямбда-абстракции с несколькими «параметрами»". Пример:

let S = \ x y z.x z (y z)

let K = \x y.x

S K K

Должно получиться в качестве ответа:

\x.x

(или любой другой терм, альфа-эквивалентный этому).

Для синтаксического анализа использовать FParsec.

***Проверяемые компетенции*:** ОПК-1, ОПК-2,ОПК-3, ПКА-1, ПКП-4, ПКП-5, ПКП-6, ПКП-7, ПКП-8

***Критерии оценивания*:** решение задачи оценивается по шкале от 0 (нет решения или решение имеет существенные недостатки) до 4 (решение работоспособно, аккуратно реализовано, не содержит ошибок в грамматике).

***Темы докладов***

1. FAKE, Scaffold
2. Веб-приложения в комбинаторном стиле
3. Type Providers, F# Data
4. Дополнительные возможности F#
5. F# и анализ данных

***Проверяемые компетенции*:** ПКА-1,УКБ-3

***Критерии оценивания*:** Доклады оцениваются по шкале от 0 до 4, где 0 — отсутствие доклада, 4 — очень хороший доклад, полностью раскрывающий тему, хорошо представленный и оформленный.

***Примеры условий контрольных работ:***

***Контрольная работа 1***

1. Найти сумму всех чётных чисел Фибоначчи, не превосходящих миллиона.
2. Описать функцию, которая работает как map, только для каждого значения исходного списка можно задать не одно, а несколько значений, на которые его надо заменить. Пример вызова: «supermap [1.0; 2.0; 3.0] (fun x -> [sin x; cos x])» должно возвращать «[sin 1.0; cos 1.0; sin 2.0; cos 2.0; sin 3.0; cos 3.0]».
3. Реализовать функцию, возвращающую все элементы двоичного дерева, удовлетворяющие переданному как параметр условию.

При решении задач контрольной работы 1 конструкции императивного программирования и мутабельное состояние использовать нельзя.

***Проверяемые компетенции*:** ОПК-3, ПКА-1, ПКП-4, ПКП-5, ПКП-6, ПКП-7, ПКП-8

***Критерии оценивания*:** решения задач оцениваются от 0 до 3 баллов, где 0 — полное отсутствие решения, 3 — правильное решение, демонстрирующее все ожидаемые навыки (комментарии, тесты, навык использования системы контроля версий, навыки функционального программирования).

***Контрольная работа 2***

1. Реализовать класс «Очередь». При попытке получить значение из пустой очереди должно бросаться исключение. Использовать классы из пространства имён System.Collections и вложенных в него — нельзя.
2. Реализовать потокобезопасный стек. Стек должен иметь методы Push и TryPop, который возвращает Some <значение в вершине> или None, если стек пуст.

***Проверяемые компетенции*:** ОПК-3, ПКА-1, ПКП-4, ПКП-5, ПКП-6, ПКП-7, ПКП-8

***Критерии оценивания*:** решения задач оцениваются от 0 до 5 баллов, где 0 — полное отсутствие решения, 5 — правильное решение, демонстрирующее все ожидаемые навыки (комментарии, тесты, навык использования системы контроля версий, навыки функционального программирования).

**Вариант реализации 3: Углублённый C++**

**Вариант 1, углублённый C++**

***Пример теоретического опроса:***

1. Что такое вызываемый объект, лямбда-выражение, обобщенный алгоритм?
2. В чем отличие array от других последовательных контейнеров стандартной библиотеки?
3. Какое требование может возникнуть к типу, содержащемуся в контейнере?
4. Что может произойти с итераторами, если вставить элемент в середину вектора? Почему?
5. Возможна ли следующая инициализация:

std::vector&lt;const int&gt; vec1{10, 0};

std::vector&lt;int&gt; vec2(vec1);

1. Можно ли уменьшить объем памяти, которую занимает вектор?
2. Опишите идею адаптера: что это такое и что позволяет делать.
3. Что позволяет обобщенным алгоритмам быть применимыми для любого типа контейнера?
4. Как безопасно вызвать fill\_n для пустого вектора?
5. Перепишите использование bind с помощью лямбда-выражения

int f(int a, int b, int c);

auto f1 = std::bind(f, 1, std::placeholders::\_3, std::placeholders::\_1);

1. Что именно делают с элементами unique(), remove() и могут ли понадобиться какие-либо дополнительные действия?
2. С помощью потоковых итераторов считать значения последовательности чисел со стандартного ввода, и вывести их на экран консоли по одному на строке.

***Проверяемые компетенции*:** ОПК-1, ПКА-1, ПКП-7, ПКП-8, УКБ-3

***Критерии оценивания*:** Каждый ответ оценивается по шкале от 0 (нет ответа или неправильный ответ) до 2 (правильный и полный ответ), далее баллы складываются и приводятся в диапазон от 0 до 10 линейным преобразованием.

***Темы докладов по паттернам проектирования:***

1. Паттерн «Компоновщик».
2. Паттерн «Декоратор».
3. Паттерн «Стратегия».
4. Паттерн «Адаптер».
5. Паттерн «Прокси».
6. Паттерн «Фасад».
7. Паттерн «Мост».
8. Паттерн «Приспособленец».
9. Паттерн «Спецификация».
10. Паттерн «Фабричный метод».
11. Паттерн «Шаблонный метод».
12. Паттерн «Абстрактная фабрика».
13. Паттерн «Одиночка».
14. Паттерн «Прототип».
15. Паттерн «Строитель».
16. Паттерн «Посредник».
17. Паттерн «Команда».
18. Паттерн «Цепочка ответственности».
19. Паттерн «Наблюдатель».
20. Паттерн «Состояние».
21. Паттерн «Посетитель».
22. Паттерн «Хранитель».

***Проверяемые компетенции*:** ПКА-1, УКБ-3

***Критерии оценивания*:** решения задачи оцениваются по шкале от 0 (нет решения или решение имеет существенные недостатки) до 1 (решение работоспособно, аккуратно реализовано).

***Примеры домашних заданий:***

*Задача «Комплексные числа»*

Реализовать комплексные числа как объект. Должен поддерживать методы:

* CompareTo(Complex);
* Add;
* Multiply;
* Print;
* Если язык позволяет, перегрузить арифметические операторы.

***Проверяемые компетенции*:** ОПК-1,ОПК-2, ОПК-3, ПКА-1, ПКП-4, ПКП-5, ПКП-6, ПКП-7, ПКП-8, УКБ-3

***Критерии оценивания*:** решения задачи оцениваются по шкале от 0 (нет решения или решение имеет существенные недостатки) до 1 (решение работоспособно, аккуратно реализовано).

*Задача «Стек»*

Реализовать стек как объект. Хорошо продумать контракт. Должен поддерживать методы:

* Push;
* Pop;
* Peek;
* Clone;
* Length;
* CompareTo(Stack) — равны или нет;
* Если язык позволяет, перегрузить арифметические операторы.

***Проверяемые компетенции*:** ОПК-1,ОПК-2, ОПК-3, ПКА-1, ПКП-4, ПКП-5, ПКП-6, ПКП-7, ПКП-8, УКБ-3

***Критерии оценивания*:** решения задачи оцениваются по шкале от 0 (нет решения или решение имеет существенные недостатки) до 10 (решение работоспособно, аккуратно реализовано).

*Задача «Компьютерная сеть»*

Реализовать в объектах симулятор распространения вирусов в компьютерной сети.

* Есть компьютеры, есть N типов операционных систем, каждый компьютер имеет Li уровень защиты от вируса;
* Компьютеры соединены с другими компьютерами сетью;
* Каждый вирус может существовать на N-ом типе операционной системы, имеет уровень атаки K;
* Если компьютеры соединены и уровень атаки вируса выше уровня защиты компьютера, то компьютер заражается;
* Вирусы «ходят» последовательно, по кругу. За ход вирус может заразить только один из компьютеров, соединенных непосредственно с уже зараженным. Либо затереть все остальные заражения на одном зараженном компьютере;
* После заражения вирус может включать или выключать подконтрольные компьютеры по желанию в течении своего хода;
* Просимулировать развитие ситуации и выявить, какие компьютеры вирус, стартуя из данного узла, может выключить на момент хода X. Так же, вывести все «пограничные» компьютеры.

***Проверяемые компетенции*:** ОПК-1,ОПК-2, ОПК-3, ПКА-1, ПКП-4, ПКП-5, ПКП-6, ПКП-7, ПКП-8, УКБ-3

***Критерии оценивания*:** решения задачи оцениваются по шкале от 0 (нет решения или решение имеет существенные недостатки) до 10 (решение работоспособно, аккуратно реализовано).

*Задача «Кинотеатр»*

Реализовать симулятор сервиса по продаже билетов в кинотеатре. Кинотеатр — это поле N \* M. Есть L зон, непересекающихся, каждая со своей ценой. Зона — последовательный набор рядов.

Запросы:

* Купить X мест, в линию или прямоугольник, бронь идет на группу с названием C;
* Купить X мест, чем ближе — тем лучше, но важно сесть вместе;
* Купить одно место;
* Снять бронь, взятую группой с названием C;
* Напечатать, инициализировать, обнулить зал;
* Подсчитать прибыль;

Посчитать некоторую программу, поданную на вход.

***Проверяемые компетенции*:** ОПК-1,ОПК-2, ОПК-3, ПКА-1, ПКП-4, ПКП-5, ПКП-6, ПКП-7, ПКП-8, УКБ-3

***Критерии оценивания*:** решения задачи оцениваются по шкале от 0 (нет решения или решение имеет существенные недостатки) до 10 (решение работоспособно, аккуратно реализовано).

**Вариант 2, многопоточное программирование на C++**

***Примеры домашних заданий***

*Задача 1*:

На pthreads или std::thread написать параллельную программу вычисляющую детерминант матрицы по методу миноров. Пул потоков не используем, лимитируем уровень на котором запускаются потоки. Потом померить время в зависимости от количества потоков. Объяснить результаты.

***Проверяемые компетенции*:** ОПК-1,ОПК-2, ОПК-3, ПКА-1, ПКП-4, ПКП-5, ПКП-6, ПКП-7, ПКП-8, УКБ-3

***Критерии оценивания*:** решения задачи оцениваются по шкале от 0 (нет решения или решение имеет существенные недостатки) до 10 (решение работоспособно, аккуратно реализовано).

*Задача 2*:

на pthreads или std::thread написать несколько вариантов для параллельной программы из примера, приведенного на занятии (номер 3):

1. Наивная реализация без mutex, показать что valgrind ругается
2. Реализация a), показать что valgrind не ругается
3. Реализация b), показать что valgrind не ругается
4. Реализация d), показать что valgrind не ругается

Сравнить производительность в каждом случае на нормальном количестве потоков.

***Проверяемые компетенции*:** ОПК-1,ОПК-2, ОПК-3, ПКА-1, ПКП-4, ПКП-5, ПКП-6, ПКП-7, ПКП-8, УКБ-3

***Критерии оценивания*:** решения задачи оцениваются по шкале от 0 (нет решения или решение имеет существенные недостатки) до 10 (решение работоспособно, аккуратно реализовано).

*Задача 3*:

на pthreads или std::thread реализовать многопоточную хеш-таблицу с методом разрешения коллизий в виде цепочек. Она должна поддерживать следующий набор операций:

1. Put
2. Remove
3. Check

Для простоты считать что ключи int, а значения — класс с двумя полями (строкой и числом). Списки должны быть свои, мутекс не должен быть на целую цепочку или на всю структуру. Максимизируем параллельность. Для тестирования реализовать несколько сценариев:

1. Последовательно добавляем 1...1000 при наличии N потоков. Потом, проверить что все положилось.
2. Стучимся из N потоков случайно добавляя/удаляя/проверяя случайное число из диапазона [1..M]. Продемонстрировать на valgrind что есть чистая выдача.

***Проверяемые компетенции*:** ОПК-1,ОПК-2, ОПК-3, ПКА-1, ПКП-4, ПКП-5, ПКП-6, ПКП-7, ПКП-8, УКБ-3

***Критерии оценивания*:** решения задачи оцениваются по шкале от 0 (нет решения или решение имеет существенные недостатки) до 10 (решение работоспособно, аккуратно реализовано).

*Задача 4, «Логгер»*:

Для реального программного обеспечения характерна следующая ситуация:

1. Имеется k работающих потоков, выполняющие однородные или кардинально разные задачи;
2. Необходимо фиксировать деятельность всех потоков в файле или на консоли, т.е. вести лог (журнал) действий.

Ведение журнала может быть полезны как для отладки программы, так и для ее нормального функционирования. Например, журналируемые файловые системы используют лог для восстановления данных после ошибок (e.g. внезапного отключения питания и соответственно диска).

Вам предлагается реализовать подобную функциональность для контролируемой (искусственной) среды:

* Имеется k потоков-рабочих. С некоторой случайной периодичностью эти потоки выражают желания сделать запись в лог (запись — строка текста, содержание неважно).
* Непосредственно запись сообщения в консоль или на диск — дорогая операция. Поэтому потоки просто складывают свои «заявки» в специальную очередь.
* Выделенный (единственный) поток-логгер при необходимости/готовности забирает заявку с очереди и осуществляет реальную запись в консоль или на диск.

Вам необходимо рассмотреть 4 отдельных случая:

* Использовать std;:condition\_variable можно, размер очереди неограничен
* Использовать std;:condition\_variable можно, размер очереди ограничен (потоки должны ждать пока место в очереди освободится, чтобы оставить заявку и продолжить работу)
* Использовать std;:condition\_variable нельзя, размер очереди неограничен
* Использовать std;:condition\_variable нельзя, размер очереди ограничен

Примечания:

1. Можно выбрать, писать в файл или консоль. С файлом helgrind отрабатывает нормально, с консолью может ругаться на «возможные ошибки» при вызове std::cout.
2. В очереди запрещено хранить строки в чистом виде (std::string или char \*), используйте простую структуру/класс с полем-строкой. Обратите внимание на слово explicit для конструкторов
3. Для имитации «случайной» деятельности потоков, можно при их создании выбрать случайный размер задержки. Можно проверить два варианта: задержка одна у всех или же все потоки работают в разном ритме. Также, можно менять размер задержки в зависимости от порядкового номера записи в данном потоке.
4. В случае ограниченной очереди, убедитесь, что потоки достаточно быстро добавляют заявки и очередь успевает переполниться.
5. Пункт 2 позволяет расширить ваше решение. Достаточно заменить строку на «описание задачи», а «поток-логгер» на несколько «потоков-исполнителей» и у вас получится неплохой пул потоков. Единственный недостаток — после выполнения задачи не будет «обратного отклика», например возвращения результата тому, кто задачу поставил. Это можно обойти, добавляя в описание задачи ссылку или указатель на слот для результата.

***Проверяемые компетенции*:** ОПК-1,ОПК-2, ОПК-3, ПКА-1, ПКП-4, ПКП-5, ПКП-6, ПКП-7, ПКП-8, УКБ-3

***Критерии оценивания*:** решения задачи оцениваются по шкале от 0 (нет решения или решение имеет существенные недостатки) до 10 (решение работоспособно, аккуратно реализовано).

**Вариант реализации 4: Методы и средства автоматической обработки текстов.**

Примерные варианты контрольных работ и вопросов к зачету:

***Контрольная работа 1:***

1. Построить детерминированный конечный автомат, распознающий строки, состоящие из некоторого числа повторений строки 101 (пустая строка, т. е. ноль повторений, тоже годится). Написать функцию на C++, распознающую такие строки при помощи построенного конечного автомата.
2. Построить недетерминированный конечный автомат, распознающий строки, начинающиеся на три нуля и кончающиеся на три единицы. Написать функцию на C++, распознающую такие строки при помощи построенного конечного автомата.
3. Написать функцию, принимающую по ссылке конечный автомат и меняющую его следующим образом: из каждого допускающего состояния добавляется переход в начальное состояние по символу '0', если из этого состояния еще не было такого перехода (а если он был, такой переход не меняется).

***Проверяемые компетенции:*** ОПК-2, ОПК-4, ОПК-5, ПКП-4, УКБ-3.

***Критерии оценивания*:** решения задач оцениваются по шкале от 0 (нет решения или решение имеет существенные недостатки) до 5 (решение работоспособно, аккуратно реализовано).

***Контрольная работа 2:***

1. Написать грамматику для следующего языка: алфавит <<(>>, <<)>>, <<[>>, <<]>>. Языку принадлежат те правильные скобочные последовательности, которые удовлетворяют двум условиям: а) рядом стоят скобки одного вида (<<()[]>> нельзя, <<()()>> --- можно) и б) непосредственно вложенные скобки имеют другой вид (<<(())>> нельзя, <<([])>> --- можно).
2. Написать по грамматике из задачи 1 функцию, распознающую, принадлежит ли строка-параметр этому языку.

***Проверяемые компетенции:*** ОПК-2, ОПК-4, ОПК-5, ПКП-4, УКБ-3.

***Критерии оценивания*:** решения задач оцениваются по шкале от 0 (нет решения или решение имеет существенные недостатки) до 5 (решение работоспособно, аккуратно реализовано).

***Примерный список вопросов к зачету:***

1. Детерминированные конечные автоматы.
2. Минимизация детерминированных конечных автоматов.
3. Использование детерминированных конечных автоматов для поиска и замены большого количества фрагментов текста.
4. Недетерминированные конечные автоматы.
5. Автоматическая обработка недетерминированных конечных автоматов.
6. Использование конечных автоматов для лексического анализа. Пример программы, использующей конечный автомат для транслитерации.
7. Средства для автоматического построения лексических анализаторов.
8. Понятие формальной грамматики*.*
9. Деревья разбора.
10. Методы построения деревьев разбора.
11. Алгоритм Кока-Янгера-Касами.
12. Алгоритм Эрли.
13. Средства для автоматического построения синтаксических анализаторов.

***Проверяемые компетенции***:ОПК-2, ОПК-4, ОПК-5, ПКП-4, УКБ-3.

***Критерии оценивания*:** ответы на вопросы оцениваются по шкале от 0 (нет ответа или неправильный ответ) до 10 (правильный и полный ответ).

**Вариант реализации 6: …**

**Вариант реализации 7: …**

**3.1.5 Методические материалы для оценки обучающимися содержания и качества учебного процесса**

Для оценки обучающимися содержания и качества учебного процесса применяется анкетирование в соответствии с методикой и графиком, утвержденными в установленном порядке.

**3.2. Кадровое обеспечение**

**3.2.1 Образование и (или) квалификация штатных преподавателей и иных лиц, допущенных к проведению учебных занятий**

К проведению лекционных занятий должны привлекаться преподаватели, имеющие диплом о высшем образовании по соответствующему направлению.

**3.2.2 Обеспечение учебно-вспомогательным и (или) иным персоналом**

Учебно-вспомогательный и инженерно-технический персонал должен иметь соответствующее образование и обладать навыками организации работы с пользовательскими программными продуктами в локальной сети компьютерного класса и в Интернете.

Также рекомендовано обеспечить курс ассистентами (teaching assistants) для поддержки обучающихся и помощи им в выполнении домашних заданий. В качестве ассистентов допустимо привлекать обучающихся старших курсов (например, в рамках педагогической практики).

**3.3. Материально-техническое обеспечение**

**3.3.1 Характеристики аудиторий (помещений, мест) для проведения занятий**

Требуется стандартно оборудованная аудитория с проектором.

Также для самостоятельной работы и выполнения контрольных работ требуется доступ в компьютерные классы обучающимся, не имеющим собственного оборудования, подходящего для данных видов деятельности.

**3.3.2 Характеристики аудиторного оборудования, в том числе неспециализированного компьютерного оборудования и программного обеспечения общего пользования**

Доска для письма маркером, мультимедийный проектор с возможностью подключения личного ноутбука преподавателя (снабжённый разъёмами D-SUB или HDMI).

**3.3.3 Характеристики специализированного оборудования**

Не требуется.

**3.3.4 Характеристики специализированного программного обеспечения**

Интегрированная среда разработки Microsoft Visual Studio версии не ниже 15.0 с установленными оснастками для программирования на языках C++, C#, F#, консольный клиент системы контроля версий git (например, Git for Windows).

**3.3.5 Перечень и объёмы требуемых расходных материалов**

Маркеры для доски, губка.

**3.4. Информационное обеспечение**

Не требуется.

**Раздел 4. Разработчики программы**

Литвинов Юрий Викторович, кандидат технических наук, доцент кафедры системного программирования, [y.litvinov@spbu.ru](mailto:y.litvinov@spbu.ru)

Брыксин Тимофей Александрович, кандидат технических наук, доцент кафедры системного программирования, [t.bryksin@spbu.ru](mailto:t.bryksin@spbu.ru)

Григорьев Семён Вячеславович, кандидат физико-математических наук, доцент кафедры информатики, [s.v.grigoriev@spbu.ru](mailto:s.v.grigoriev@spbu.ru)

Чернышев Георгий Алексеевич, ассистент кафедры информационно-аналитических систем, [g.chernyshev@spbu.ru](mailto:g.chernyshev@spbu.ru)

Лебединский Дмитрий Михайлович, кандидат физико-математических наук, доцент кафедры параллельных алгоритмов, [d.lebedinsky@spbu.ru](mailto:d.lebedinsky@spbu.ru)

Григорьев Дмитрий Алексеевич, кандидат физико-математических наук, доцент кафедры информатики, [d.a.grigoriev@spbu.ru](mailto:d.a.grigoriev@spbu.ru)

Тут надо вписать себя