**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**

Программирование

Software Design

**Язык(и) обучения**

русский

Трудоемкость в зачетных единицах: 8

Регистрационный номер рабочей программы: 002212

Санкт-Петербург

2020

**Раздел 1. Характеристики учебных занятий**

**1.1. Цели и задачи учебных занятий**

Цель изучения дисциплины: получение обучающимися основных навыков программирования. После изучения дисциплины обучающиеся должны обладать навыками, достаточными для эффективного решения реализационных задач в научно-исследовательских и промышленных проектах под руководством опытного разработчика, а также быть способными выполнять практическую часть учебных и производственных практик, выпускной квалификационной работы на высоком техническом уровне.

Задачи обучения:

1. Приобретение практических навыков программирования в структурном и объектно-ориентированном стиле.
2. Владение классическими алгоритмами и структурами данных, понимание сложности алгоритмов, умение выбрать алгоритм или структуру данных, наиболее пригодные для решения задачи.
3. Приобретение навыков кодирования в соответствии со стилями кодирования, навыков грамотного оформления исходного кода.
4. Умение использовать системы контроля версий, системы сборки, системы непрерывной интеграции, системы и библиотеки модульного тестирования при разработке программных проектов. Понимание назначения и места этих инструментов в процессе разработки.
5. Свободное владение хотя бы одним объектно-ориентированным языком программирования.
6. Навыки отладки, тестирования, статического анализа и документирования программ.
7. Понимание основных концепций архитектуры программного обеспечение, владение ключевыми шаблонами объектно-ориентированного проектирования.
8. Навыки многопоточного программирования, программирования сетевых приложений, веб-приложений, настольных приложений с пользовательским интерфейсом.
9. Владение методологией эксперимента в области программирования, умение корректно выполнить замеры.
10. Понимание гибких методологий разработки программного обеспечения.

Дисциплина реализуется в нескольких вариантах, которые, помимо решения перечисленных тут задач обучения, направлены на удовлетворение индивидуальных образовательных потребностей обучающихся и позволяют им с ранних курсов специализироваться в различных направлениях программной инженерии и компьютерных наук. Варианты реализации читаются на примере разных языков программирования и имеют разные программы, ориентированные на разные уровни подготовленности и способностей обучающихся. Аннотации вариантов реализации и программы каждого варианта приведены в разделе 2.2. Распределение обучающихся по вариантам реализации выполняется на основании их личных предпочтений с учётом результатов входного тестирования.

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

Программа курса предназначена для обучающихся 1 и 2 курса бакалавриата. Предварительные знания не требуются. Для обучающихся, имеющих начальную подготовку в области программирования, существуют варианты реализации, позволяющие им эффективно использовать свои знания, для обучающихся, не имеющих начальных знаний, существуют варианты реализации, начинающие с нуля.

**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-2 – способен применять современный математический аппарат, связанный с проектированием, разработкой, реализацией и оценкой качества программных комплексов в различных областях человеческой деятельности;
* ОПК-4 – способен участвовать в разработке технической документации программных продуктов и программных комплексов;
* ОПК-5 – способен инсталировать и сопровожать программное обеспеченение для информационных систем и баз данных, в том числе отечественного производства;
* ПКП-3 – способен преподавать математику и информатику в средней школе, специальных учебных заведениях на основе полученного фундаментального образования и научного мировоззрения;
* ПКП-4 – способен применять современные информационные технологии при проектировании, реализации, оценке качества и анализа эффективности программного обеспечения для решения задач в различных предметных областях;
* 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 программированию 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 40 часов в первом семестре, 10 часов во втором семестре, 30 часов в третьем семестре.

**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 |  |  |  | 43 | 15 | 4 |  |  | 2 |  |  |  | 75 |  | 5 |  | 40 | 4 |
|  |  |  |  | 2-15 | 2-15 | 2-15 |  |  | 1-15 |  |  |  | 1-1 |  | 1-1 |  |  |  |
| Семестр 2 |  |  |  | 11 | 15 | 4 |  |  | 2 |  |  |  | 32 |  | 8 |  | 10 | 2 |
|  |  |  |  | 2-15 | 2-15 | 2-15 |  |  | 1-15 |  |  |  | 1-1 |  | 1-1 |  |  |  |
| Семестр 3 |  |  |  | 15 | 15 |  |  |  | 2 |  |  |  | 37 |  | 3 |  | 30 | 2 |
|  |  |  |  | 2-15 | 2-15 |  |  |  | 1-15 |  |  |  | 1-1 |  | 1-1 |  |  |  |
| ИТОГО |  |  |  | 69 | 45 | 8 |  |  | 6 |  |  |  | 144 |  | 16 |  |  | 8 |

|  |  |  |  |
| --- | --- | --- | --- |
| 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 |  |
| Семестр 2 |  | зачёт |  |
| Семестр 3 |  | зачёт |  |

**2.2. Структура и содержание учебных занятий**

Структура и содержание учебных занятий зависят от варианта реализации дисциплины. Обучающимся предлагается семь вариантов реализации:

1. **Промышленное программирование, C, C#** — данный вариант ориентирован на обучающихся без предварительных знаний и нацелен на быстрое погружение «с нуля» в программирование с основным упором на индустриальные практики. Курс знакомит с программированием в структурном стиле на языке C, практически в самом начале вводятся системы контроля версий, стиль кодирования, документирование и тестирование ПО. Алгоритмическая теория используется прежде всего как материал, на котором можно «набить руку», однако, несмотря на вспомогательную роль в данном курсе, преподаётся в первом семестре довольно плотно — в результате обучающиеся могут самостоятельно реализовать на C классические структуры данных и алгоритмы, включая списки, хеш-таблицы, деревья (в том числе, самобалансирующиеся), графы. Со второго семестра преподаётся объектно-ориентированное программирование на примере языка C# и различные прикладные технологии, включая модульное тестирование и разработку пользовательских интерфейсов. После этой части курса обучающиеся способны, в частности, разработать несложную игру на движке Unity. В третьем семестре также на примере C# и платформы .NET преподаётся многопоточное программирование, сетевое программирование, работа с базами данных, веб-программирование. После третьего семестра обучающиеся обладают минимальным набором знаний, необходимых для решения практических задач в профессиональной деятельности. Данный трек рассчитан на продолжение в 4-м семестре в рамках дисциплины «Структуры и алгоритмы компьютерной обработки данных», где материал курса дополняется введением в функциональное программирование на примере языка F#, а также рассмотрением дополнительных прикладных задач, что позволяет после второго курса сформировать у обучающихся широкий кругозор, богатый инструментарий навыков и относительно целостную, хоть и неглубокую, картину мира программной инженерии. Данный вариант реализации отличается большим количеством теоретического материала, сообщаемого на занятиях, и большим количеством домашних заданий, проверяемых индивидуально, прежде всего на соответствие хорошим практикам индустриального программирования, и, несмотря на то, что ориентирован на обучающихся без предварительных знаний, требует большого количества усилий и самодисциплины для освоения.
2. **Прикладное и исследовательское программирование на примере F#** – данный вариант ориентирован на обучающихся без предварительных знаний и нацелен на быстрое погружение «с нуля» в программирование с основным упором на исследовательские задачи, с изучением, однако, универсальных и применимых и в промышленном программировании практик. Курс знакомит с программированием на языке F# с использованием основных поддерживаемых парадигм: императивное программирование, ООП и ФП, практически в самом начале вводятся системы контроля версий, стиль кодирования, документирование и тестирование ПО. Алгоритмическая теория используется прежде всего как материал, на котором можно «набить руку», однако, несмотря на вспомогательную роль в данном курсе, преподаётся в первом семестре довольно плотно — в результате обучающиеся могут самостоятельно реализовать на F# классические структуры данных и алгоритмы, включая списки, деревья, графы. Кроме классического взгляда на базовые алгоритмы и структуры данных, делается акцент на особенности, возникающие из-за особенностей функционального программирования, а так же на некоторых математических основах. В частности, взгляд на некоторые задачи через линейную алгебру и обсуждение того, что этот подход даёт на практике. Второй семестр посвящён особенностям дизайна приложений с использованием различных подходов (в основном ООП и ФП), а также основам параллельного программирования. В третьем семестре изучаются специфичные техники программирования: метапрограммирование, программирование на GPGPU. После третьего семестра обучающиеся обладают минимальным набором знаний, необходимых для решения !!!. Данный трек рассчитан на продолжение в 4-м семестре в рамках дисциплины «Структуры и алгоритмы компьютерной обработки данных», где материал курса дополняется !!!, а также рассмотрением дополнительных прикладных задач, что позволяет после второго курса сформировать у обучающихся широкий кругозор, богатый инструментарий навыков и относительно целостную, хоть и неглубокую, картину мира программной инженерии. Данный вариант реализации отличается большим количеством теоретического материала, сообщаемого на занятиях, и большим количеством домашних заданий, проверяемых индивидуально, и, несмотря на то, что ориентирован на обучающихся без предварительных знаний, требует большого количества усилий и самодисциплины для освоения.

Распределение обучающихся по вариантам реализации выполняется на основании их личных предпочтений с учётом результатов входного тестирования.

**Вариант реализации 1: промышленное программирование, C, C#.**

Период обучения (модуль): семестр 1.

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Наименование темы (раздела, части) | Вид учебных занятий | Количество часов |
| I. | Введение в программирование | практические занятия | 13 |
| лабораторные работы | 5 |
| контрольные работы | 2 |
| самостоятельная работа | 20 |
| II. | «Динамические» структуры данных | практические занятия | 20 |
| лабораторные работы | 4 |
| контрольные работы | 2 |
| самостоятельная работа | 40 |
| III. | Парадигмы программирования, дополнительные алгоритмы | практические занятия | 10 |
| лабораторные работы | 6 |
| самостоятельная работа | 15 |
| IV. | Промежуточная аттестация | самостоятельная работа | 5 |
| зачёт | 2 |

Раздел 1: Введение в программирование.

1. Введение в C, структура программы, основные языковые конструкции (функции и рекурсия, переменные, элементарные типы и арифметические операции, массивы, строки, ввод-вывод с консоли), среда разработки.
2. Стиль кодирования. Процесс сборки и запуска программы — компилятор, линковщик, IDE. Практика, написание первых программ на C. После этого занятия следование стилю кодирования обязательно для зачёта заданий.

*Домашняя работа 1.*

1. Понятие сложности алгоритмов, O-символика, основные техники оценки трудоёмкости программы. Примеры задач, имеющих алгоритмы с разной трудоёмкостью: сортировки (пузырьком и подсчётом), вычисление факториала (рекурсивный и итеративный варианты алгоритма), вычисление чисел Фибоначчи (рекурсивно, итеративно и через возведение в степень матрицы 2х2 специального вида).
2. Отладка и тестирование. Практика по пользованию отладчиком. Понятие модульных тестов. После этого занятия написание тестов к программам обязательно.

*Домашняя работа 2.*

1. Сортировки: вставкой, выбором, сортировка Шелла, быстрая сортировка, сортировка кучей. Оценки трудоёмкости. Двоичный поиск.
2. Системы контроля версий, git. Основные команды. Внутреннее устройство репозитория, коммиты, ветки, слияние веток, конфликты, работа с удалёнными репозиториями. Git flow. Хорошие практики использования систем контроля версий. Графические клиенты git. С этого занятия все домашние работы сдаются в виде пуллреквестов на сервисе GitHub.

*Домашняя работа 3.*

1. Внутреннее представление данных. Целые числа, вещественные числа (IEEE 754), строки (ASCII, Unicode и UTF-8).
2. Структуры, указатели, модули и файлы. Общие правила разбиения на модули, модули в C. Практика по разрешению конфликтов в git.

*Домашняя работа 4.*

1. О разработке программных продуктов. Понятие жизненного цикла программного обеспечения, фазы жизненного цикла, водопадная и спиральная модели разработки. Понятие методологии разработки, понятие парадигмы программирования. Git flow.
2. Контрольная работа.

*После этих занятий домашняя работа не задаётся (даётся возможность догнать отстающим).*

Раздел 2: «Динамические» структуры данных.

1. Стек и очередь на указателях. Практика по написанию стека.
2. Списки на указателях, односвязные, двусвязные и циклические. Практика по написанию односвязного списка.

*Домашняя работа 5.*

1. Понятие абстрактного типа данных. Стек и список как абстрактные типы данных. Понятие инварианта. Пример применения АТД для реализации «полиморфного» кода — сортировка слиянием.

*Домашняя работа 6.*

1. Переписывание контрольной работы.
2. Деревья. Дерево как математический объект и как АТД, обходы деревьев. Деревья выражений, двоичные деревья. Пример: алгоритм Хаффмана. Реализация деревьев. Деревья поиска, основные операции.
3. Практика по написанию двоичного дерева поиска в командах с использованием GitHub.

*Домашняя работа 7.*

1. Самобалансирующиеся деревья. АВЛ-дерево, балансировка, основные операции. Красно-чёрные деревья. Splay-деревья. Декартовы деревья.

*Домашняя работа 8.*

1. Переписывание контрольной.
2. Хеш-таблицы. Хеш-функции, их выбор и свойства, совершенные хеш-функции, универсальные хеш-функции, комбинирование хеш-функций. Варианты реализации хеш-таблиц (списки значений и открытая адресация), их свойства.
3. Работа с консолью, системы сборки. Либо командная практика по написанию хеш-таблицы, в зависимости от успехов группы.

*Домашняя работа 9.*

1. Графы. Граф как математический объект и как АТД, способы представления графа: матрица смежности, матрица инцидентности, список смежности. Задача достижимости, обходы в глубину и в ширину. Проверка на ацикличность, глубинное остовное дерево. Задача поиска кратчайшего пути, алгоритм Дейкстры. Алгоритм Флойда-Уоршелла. О связи графов и бинарных отношений над множеством, топологическая сортировка.
2. Практика: реализация обхода графа в ширину.

*Домашняя работа 10.*

1. Контрольная работа.

Раздел 3: Парадигмы программирования, дополнительные алгоритмы.

1. Парадигмы программирования. Структурное программирование: машины Тьюринга, архитектура фон Неймана, языки-представители, подробнее про Ada. Объектно-ориентированное программирование, основные понятия, инкапсуляция, наследование, полиморфизм. Языки-представители. Пример объектно-ориентированного кода на C++. Функциональное программирование. Понятие лямбда-исчисления, основные принципы и особенности функционального программирования. Языки представители, подробнее Haskell, F#.
2. Доклады: алгоритмы поиска подстроки в строке (Кнута-Морриса-Пратта, Рабина-Карпа, Бойера-Мура), алгоритм A\*, пакет визуализации графов GraphViz и язык dot.

*Домашняя работа 11.*

1. Парадигмы программирования-2. Логическое программирование, Пролог. Рекурсивное программирование, Рефал. Стековое программирование, Форт. Визуальное программирование, визуальное моделирование, UML, предметно-ориентированное моделирование.
2. Автоматы, лексический анализ. Назначение лексических анализаторов, языки и регулярные выражения, диаграммы переходов, построение кода по диаграммам, НКА, ДКА, построение НКА по регулярному выражению, работа ДКА с таблицей переходов.
3. Практика, написание ДКА для несложного регулярного выражения.

*Домашняя работа 12.*

1. Переписывание контрольной работы.
2. Первая попытка написания зачётной работы.
3. Вторая попытка написания зачётной работы.

Период обучения (модуль): семестр 2.

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Наименование темы (раздела, части) | Вид учебных занятий | Количество часов |
| I. | Объектно-ориентированное программирование | практические занятия | 4 |
| лабораторные работы | 0 |
| самостоятельная работа | 6 |
| II. | Качество программного обеспечения: модульные тесты, обработка ошибок, непрерывная интеграция, инструменты разработки и управления проектами | практические занятия | 6 |
| лабораторные работы | 0 |
| самостоятельная работа | 8 |
| III. | Аспекты современных языков программирования: событийно-ориентированное программирование, генерики | практические занятия | 1 |
| лабораторные работы | 5 |
| контрольные работы | 2 |
| самостоятельная работа | 8 |
| IV. | Элементы архитектуры программного обеспечения | практические занятия | 0 |
| лабораторные работы | 10 |
| контрольные работы | 2 |
| самостоятельная работа | 10 |
| V. | Промежуточная аттестация | самостоятельная работа | 8 |
| зачёт | 2 |

Раздел 1: Объектно-ориентированное программирование.

1. Введение в C#, понятие байт-кода и виртуальной машины, CIL, сборка мусора. Основы синтаксиса C# — структура программы, методы, элементарные типы, массивы, перечисления, структуры, стиль кодирования.

*Домашняя работа 1.*

1. Объектно-ориентированное программирование: основные понятия, объекты и классы, инварианты. Абстракция, инкапсуляция, наследование, полиморфизм, типы времени компиляции и времени выполнения. Абстрактные классы. ООП в C#: ссылочные типы и типы-значения, передача параметров по ссылке, конструкторы, наследование, интерфейсы, абстрактные классы, виртуальные методы, абстрактные методы, перевведение методов. Модификаторы видимости, модификаторы partial, sealed, static. Вложенные классы, преобразования типов, иерархия основных классов стандартной библиотеки. Представление объектов в памяти.

*Домашняя работа 2.*

Раздел 2: Качество программного обеспечения: модульные тесты, обработка ошибок, непрерывная интеграция, инструменты разработки и управления проектами.

1. Комментирование кода на C#, XML Documentation. Модульное тестирование. Тестовые сценарии, примеры. Модульные тесты в C#: NUnit, Microsoft Unit Test Framework, демонстрация по использованию NUnit. Хорошие практики модульного тестирования. Data-driven-тесты.

*Домашняя работа 3.*

1. Исключения и обработка ошибок. Бросание и обработка исключений, try-catch-finally, фильтры исключений. Иерархия классов-исключений стандартной библиотеки. Свойства класса Exception. Перебрасывание исключений. Объявление своих классов-исключений. «Интересные» классы-исключения стандартной библиотеки.

*Домашняя работа 4.*

1. Экосистема проектов с открытым исходным кодом. Непрерывная интеграция: задачи, облачный сервис AppVeyor, настройка сборки, матрица сборки. Облачный сервис Travis. Инструменты анализа качества: CodeCov, Codacy. Инструменты планирования и управления проектом: Trello, Pivotal Tracker. Средства коммуникации: Slack, Gitter. Багтрекер GitHub Issues. Другие средства управления проектом GitHub. Авторское право и лицензии.

*Домашняя работа 5.*

Раздел 3: Аспекты современных языков программирования: событийно-ориентированное программирование, генерики.

1. Событийно-ориентированное программирование. Паттерн «Наблюдатель». События в .NET: делегаты, обработчики, генерики Func и Action. События. Анонимные методы и замыкания. Лямбда-выражения. Объявление событий в .NET. Ручное управление подпиской на события (ключевые слова add и remove).

*Домашняя работа 6.*

1. Пользовательский интерфейс. Библиотека Windows Forms: обзор библиотеки, класс Control, обработка пользовательского ввода, валидация, привязка данных, хорошие практики. Мастер-класс по разработке приложения на Windows Forms.

*Домашняя работа 7.*

1. Контейнеры и генерики. Контейнеры стандартной библиотеки .NET, паттерн «Итератор», энумераторы. Типизация, понятия ad-hoc и универсального полиморфизма, полиморфизм подтипов и параметрический полиморфизм. Генерики в .NET, внутреннее устройство и использование. Генерик-методы и генерик-классы. Открытые и закрытые типы, генерики и вложенные классы. Ограничения на типы. Понятие вариантности, ковариантность и контравариантность, примеры.

*Домашняя работа 8.*

1. Контрольная работа.

Раздел 4: Элементы архитектуры программного обеспечения.

1. Визуальное моделирование, UML. Метафора моделирования, цель моделирования. Диаграммы UML. Диаграмма классов: синтаксис, синтаксис свойств, агрегация и композиция. Диаграмма компонентов. Диаграмма случаев использования. Диаграммы активностей, последовательностей, конечных автоматов. Генерация кода по диаграммам конечных автоматов. Диаграммы развёртывания. Примеры CASE-инструментов. Предметно-ориентированные визуальные языки.

*Домашняя работа 9.*

1. Переписывание контрольной.
2. Доклады.

*Домашняя работа 10.*

1. Правила написания хорошего кода в ООП. Сложность ПО, сопряжение и связность, модульность, свойства модулей. Понятие объекта с точки зрения архитектуры ПО, выделение объектов. Наследование и композиция. Принципы SOLID, закон Деметры. Примеры абстракций: принцип единственности ответственности, уровень абстракции, общие рекомендации по дизайну абстракций. Инкапсуляция: принцип минимизации доступности, общие рекомендации. Наследование, общие рекомендации, конструкторы. Мутабельность. Преждевременная оптимизация. Принцип Fail Fast. Другие рекомендации.
2. Первая попытка написания зачётной работы.
3. Вторая попытка написания зачётной работы

Период обучения (модуль): семестр 3.

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Наименование темы (раздела, части) | Вид учебных занятий | Количество часов |
| I. | Многопоточное программирование | практические занятия | 4 |
| лабораторные работы | 4 |
| самостоятельная работа | 10 |
| II. | Сетевое программирование | практические занятия | 3 |
| лабораторные работы | 3 |
| самостоятельная работа | 8 |
| III. | Рефлексия | практические занятия | 2 |
| лабораторные работы | 2 |
| самостоятельная работа | 5 |
| IV. | Базы данных | практические занятия | 1 |
| лабораторные работы | 1 |
| самостоятельная работа | 3 |
| V. | Современные пользовательские интерфейсы | практические занятия | 2 |
| лабораторные работы | 2 |
| самостоятельная работа | 5 |
| VI. | Веб-программирование | практические занятия | 3 |
| лабораторные работы | 3 |
| самостоятельная работа | 6 |
| VII. | Промежуточная аттестация | самостоятельная работа | 3 |
| зачёт | 2 |

Раздел 1: Многопоточное программирование.

1. Многопоточное программирование: предназначение, потенциальные проблемы. Понятия процесса и потока, многопроцессной и многопоточной программы. Планировщик, пример: планировщик и представление потока в ОС Windows. Поток как абстракция длительной задачи и поток как абстракция вычислителя. Потоки в .NET: Thread, пример гонки по данным. Race condition и deadlock. Понятия конвейера процессора, процессорного кеша.

*Домашняя работа 1.*

1. Примитивы синхронизации, синхронизация режима ядра и режима пользователя. Атомарные операции, Volatile. Понятие модели памяти. Критические области. Активное ожидание. Задача производителя и потребителя. Семафоры, мьютексы, мониторы, conditional variables (события).
2. Практическое занятие: моделирование классической проблемы «Обедающие философы».

*Домашняя работа 2.*

1. Высокоуровневая многопоточность в .NET. Foreground- и Background-потоки, пул потоков, классы Task, CancellationToken, TaskScheduler. Модель асинхронного программирования C#: async/await, примеры, потенциальные проблемы, связь с Task. Task Parallel Library. Потокобезопасные и немутабельные коллекции .NET.

Раздел 2: Сетевое программирование.

1. Работа с сетью, низкий уровень. Архитектура глобальных сетей, модель OSI, стек протоколов TCP/IP, обзор уровней модели OSI. Протокол IP: IP-адреса, формат пакета. DNS, NAT. Порты и сокеты. Консольные утилиты для работы с сетью. Работа с сетью в .NET, минимальный пример клиент-серверного приложения на сокетах, асинхронный клиент и сервер. Работа с UDP.

*Домашняя работа 3.*

1. Работа с сетью в .NET на высоком уровне. Протоколы прикладного уровня, HTTP. Класс HttpClient. Веб-сервисы, архитектурный стиль REST, пример: Google Drive API. Отладка веб-сервисов. Основы сетевой безопасности.
2. Практическое занятие: клиент для социальной сети «ВКонтакте».

Раздел 3: Рефлексия.

1. Рефлексия, вообще и в .NET. Сборки в .NET, сильные и слабые имена сборок. Получение информации о сборках, типах, полях, методах и т.д., создание экземпляров объектов. Атрибуты. Пример использования рефлексии: библиотеки сериализации (System.Xml.Serialization, Newtonsoft.Json). Ключевое слово dynamic. Генерация кода посредством ILGenerator.

*Домашняя работа 4.*

1. Проверочная работа.

Раздел 4: Базы данных.

1. Базы данных, виды СУБД. Реляционная модель данных, язык SQL, ключи и ограничения, основные операции над данными. Низкоуровневая работа с данными из .NET, ADO.NET. Высокоуровневая работа с данными, Entity Framework.

Раздел 5: Современные пользовательские интерфейсы.

1. Библиотека WPF, обзор библиотеки. Язык разметки интерфейсов XAML. Архитектура WPF. Зависимые свойства и маршрутизируемые события. Привязка данных, свойство DataContext, конвертеры. Паттерн «Model-View-ViewModel». Геометрия элементов пользовательского интерфейса.

*Домашняя работа 5.*

1. Паттерн «Команда», команды в WPF. Валидация пользовательского ввода. Шаблоны и стили визуального оформления, триггеры. Ресурсы. Практика: GUI для клиента «ВКонтакте».

Раздел 6: Веб-программирование.

1. Веб-программирование. Архитектура типичных веб-приложений, веб-приложения и веб-сервисы. Фронтенд: язык HTML, DOM, CSS, азы JavaScript, асинхронные запросы. Бэкенд: механизм обработки HTTP-запросов в Windows, ASP.NET MVC. Язык описания правил генерации Razor. Роутинг запросов. Привязка моделей.

*Домашняя работа 6.*

1. Практический пример: приложение для регистрации на конференцию. Модель, вёрстка, работа с СУБД, Bootstrap, валидация.
2. Проверочная работа.

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: Прикладное и исследовательское программирование на примере F#.**

Период обучения (модуль): семестр 1.

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Наименование темы (раздела, части) | Вид учебных занятий | Количество часов |
| I. | Введение | практические занятия | 16 |
| лабораторные работы | 4 |
| контрольные работы | 4 |
| самостоятельная работа | 20 |
| II. | Структуры данных и алгоритмы | практические занятия | 15 |
| лабораторные работы | 5 |
| контрольные работы | 2 |
| самостоятельная работа | 38 |
| III. | Парадигмы программирования | практические занятия | 12 |
| лабораторные работы | 6 |
| контрольные работы | 0 |
| самостоятельная работа | 17 |
| IV. | Промежуточная аттестация | самостоятельная работа | 5 |
| зачёт | 2 |

Раздел 1: Введение

1. Программирование – не только написание кода. Инфраструктура проекта, рабочее окружение, система контроля версий, непрерывная сборка. Соответствующие решения на примере инфраструктуры вокруг GitHub. Практика развёртывания соответствующей инфраструктуры. С этого момента домашние работы только через GitHub с налаженной сборкой.

*Домашняя работа 1*

1. Первое знакомство с .NET и F#. Общие сведения о платформе. F# – как один из языков платформы. Основные особенности F#, примеры кода, базовые языковые конструкции и типы (примитивные, массивы), структура программы.
2. Тестирование программ: ручное, автоматизированное, автоматическое. Доказательство корректности vs тестирование. Типы тестов: модульные, интеграционные, unit. Автоматизация создания тестов. Примеры инструментов: FsUnit, NUnit. С этого момента все домашние работы должны быть снабжены автоматически запускаемыми при сборке тестами.

*Домашняя работа 2*

1. Отладка кода. Некоторые методы отладки: отладочная печать, логирование, использование пошаговых отладчиков. Некоторые шаги отладки: формулировка гипотезы и её проверка, локализация ошибки, работа с тестами. Практика по использованию отладчика.
2. Структура проекта вообще и на языке F# в частности. Разбиение кода на модули, файлы, библиотеки. Переиспользование кода. Зависимости между модулями, библиотеками. Пространства имён.
3. Базовые структуры данных, алгоритмы и их выражение в F#. Функция. Ветвления, рекурсия и итерация. Изменяемость/неизменяемость, понятие переменной, понятие связывания. Базовые типы и основы работы с ними: численные типы, логические, строки, матрицы, массивы, списки, структуры.

*Домашняя работа 3*

1. Сортировки: пузырьком, вставкой, Хоара. Различные сценарии использования: поддержание отсортированного набора, сортировка всего набора целиком. Некоторые особенности реализации: наивная функциональная реализация Хоара, реализация на массиве.
2. Основы машинного представления данных. Представления чисел. Представление чисел с плавающей точкой. Проблемы переполнения. Битовые операции. Строки, кодировки.

*Домашняя работа 4*

1. Основы анализа алгоритмов. Модель вычислителя. Понятие элементарной операции. Асимптотика, «О»-символика.
2. Постановка эксперимента и оформление результатов. Эксперимент по сравнению и анализу производительности. Точность проведения замеров. Особенности работы с управляемыми средами (JIT, сборка мусора). «Масштабы времени», цель эксперимента и точность измерений, инструменты измерений. Базовая статистическая обработка данных. Способы визуализации результатов.

*Домашняя работа 5*

1. Контрольная работа.
2. Разбор домашних работ, разбор контрольной.

Раздел 2: Структуры данных и алгоритмы.

1. Понятие типа данных. Системы типов: статические, динамические, строгие, нестрогие. Примеры языков с разными системами типов. Понятие о разной выразительности («мощности») систем. Приведение типов: автоматическое, ручное. Вывод типов по Хиндли-Милнеру.
2. Алгебраические типы данных: кортежи, DU. Примеры на F#. Единицы измерения.

*Домашняя работа 6*

1. Обобщённые типы данных. Понятие о полиморфизме. Ad-hoc полиморфизм и бинарные операции. Типовые параметры и ограничения на них в F#. Структурный полиморфизм в Ocaml.
2. Списки, деревья: как формальные объекты, структуры данных и как примеры алгебраических обобщённых типов. Реализация списка и дерева. Обходы списков и деревьев.

*Домашняя работа 7*

1. Длинная арифметика. Практика работы со списками. Ещё раз о проблеме переполнения. Целочисленная арифметика на списках.
2. Граф как формальный объект и как структура данных. Понятие о бинарном отношении и его свойствах: транзитивность, рефлексивность. (Не)Ориентированные, (не)помеченные графы. Способы представления графов: список смежности, матрица смежности.

*Домашняя работа 8*

1. Базовые алгоритмы на графах. Обходы в глубину и ширину, построение транзитивного замыкания, поиск кратчайшего пути.
2. Регулярные выражения и конечные автоматы. Определения. Построение автомата по регулярному выражению. Применения регулярных выражений (поиск, анализ текста, моделирование систем, анализ программного кода).
3. Линейная алгебра. Основы: матрица, вектор, полукольцо, кольцо, поле. Сведение некоторых задач к операциям линейной алгебры (транзитивное замыкание, кратчайшие пути, пересечение автоматов). Особенности практического использования такого подхода: разреженные структуры данных, абстрактность, композициональность. Quad-tree

*Домашняя работа 9*

1. Контрольная работа.
2. Разбор домашних работ, разбор контрольной.

Раздел 3: Парадигмы программирования

1. Устройство языков программирования: лексика, синтаксис, семантика. Определения, примеры. Шаги обработки кода: лексический и синтаксический анализы, «семантический» анализ.
2. Лексический и синтаксический анализ. Введение в формальные языки как способ описания синтаксиса. Способы реализации лексических и синтаксических анализаторов. ANTLR, fslex+fsyacc, FParsec, YaccConstructor.

*Домашняя работа 10*

1. Устройство сред разработки и компиляторов, интерпретаторов: общие шаги, классические возможности, JIT/AOT. Примеры из .NET, F#, JVM, LLVM.
2. Интерпретация и компиляция. Особенности, разновидности интерпретаторов, основные шаги. Особенности, разновидности компиляторов, основные шаги. Примеры, пример реализации простого интерпретатора.

*Домашняя работа 11*

1. Парадигмы программирования-1. Структурное программирование: машины Тьюринга, архитектура фон Неймана, языки-представители. Объектно-ориентированное программирование, основные понятия, инкапсуляция, наследование, полиморфизм. Языки-представители. Пример объектно-ориентированного кода на F#. Функциональное программирование. Понятие лямбда-исчисления, основные принципы и особенности функционального программирования. Языки представители, Haskell, F#, Ocaml. Программирование в зависимых типах.
2. Парадигмы программирования-2. Логическое программирование, Пролог. Рекурсивное программирование, Рефал. Стековое программирование, Форт. Визуальное программирование, визуальное моделирование, UML, предметно-ориентированное моделирование.

Период обучения (модуль): семестр 2.

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Наименование темы (раздела, части) | Вид учебных занятий | Количество часов |
| I. | Программный продукт, проект | практические занятия | 4 |
| лабораторные работы | 4 |
| контрольные работы | 0 |
| самостоятельная работа | 8 |
| II. | Объектно-ориентированное программирование | практические занятия | 4 |
| лабораторные работы | 4 |
| контрольные работы | 0 |
| самостоятельная работа | 8 |
| III. | Функциональное программирование | практические занятия | 2 |
| лабораторные работы | 2 |
| контрольные работы | 2 |
| самостоятельная работа | 6 |
| IV. | Параллельное программирование | практические занятия | 1 |
| лабораторные работы | 7 |
| контрольные работы | 2 |
| самостоятельная работа | 10 |
| V. | Промежуточная аттестация | самостоятельная работа | 8 |
| зачёт | 2 |

Раздел 1: Программный продукт, проект.

1. Программа, проект, продукт – что есть что, различия. Жизненный цикл продукта.
2. Открытый исходный код: окружение, инструменты, лицензии. Экосистема проектов с открытым исходным кодом. Непрерывная интеграция: задачи, облачный сервис AppVeyor, настройка сборки, матрица сборки. Облачный сервис Travis. Инструменты анализа качества, линтеры, покрытие тестами. Инструменты планирования и управления проектом: Trello, Pivotal Tracker. Средства коммуникации: Slack, Gitter. Багтрекер GitHub Issues. Другие средства управления проектом GitHub. Авторское право и лицензии.
3. Документация, комментирование, автоматическая генерация документации по комментариям. Публикация документации на gh-pages.
4. Визуальное моделирование, UML. Метафора моделирования, цель моделирования. Диаграммы UML. Диаграмма классов: синтаксис, синтаксис свойств, агрегация и композиция. Диаграмма компонентов. Диаграмма случаев использования. Диаграммы активностей, последовательностей, конечных автоматов. Генерация кода по диаграммам конечных автоматов. Диаграммы развёртывания. Примеры CASE-инструментов. Предметно-ориентированные визуальные языки.

*Домашняя работа 1*

Раздел 2: Объектно-ориентированное программирование

1. Основы. Инкапсуляция и наследование. Интерфейс. Множественное наследование и множественная реализация интерфейса. Абстрактный класс. Примеры ООП на F#.
2. Некоторые базовые паттерны проектирования.
3. Исключения и обработка ошибок. Бросание и обработка исключений. Перебрасывание исключений. Объявление своих классов-исключений. Некоторые особенности использования исключений (легковесность в OCaml vs тяжеловесность в .NET).
4. GUI (winforms, GTK и т.д). Основы событийно-ориентированного программирования. Основы разработки GUI (вёрстки).

*Домашняя работа 2*

Раздел 3: Функциональное программирование

1. Основы проектирования с использованием ФП. Сравнение ФП и ООП дизайна и их совмещение. Railway programming как способ жить без исключений.
2. Особенности чистого функционального программирования. Неизменяемые структуры данных, проблемы с эффективностью и их возможные решения. Отсутствие побочных эффектов.

*Домашняя работа 3*

1. Контрольная работа.

Раздел 4: Параллельное программирование

1. Архитектуры, подходы, парадигмы. SIMD, MIMD, SPMD. Асинхронное программирование, параллельное программирование. Процессы и потоки: многопроцессорность и многопоточность. Гонки по данным, блокировки.
2. Базовые примитивы работы с потоками и разделяемыми ресурсами в F#. Функция lock. Запуск функции в отдельном потоке. Особенности работы с исключениями. Общее состояние. Плюсы и минусы неизменяемости.
3. Array.Parallel, ParallelSeq и другие высокоуровневые средства параллельного программирования на F#. Линейная алгебра и параллелизм: бонусы, проблемы, возможные решения.

*Домашняя работа 4*

1. Actor-ориентированное программирование как реализация асинхронного (concurrent) программирования. Коммуникация на сообщениях. Mailbox processor и Hopac как реализации. Примеры использования Mailbox processor и Hopac.

*Домашняя работа 5*

1. Контрольная работа.

Период обучения (модуль): семестр 3.

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Наименование темы (раздела, части) | Вид учебных занятий | Количество часов |
| I. | Особенности исследовательских проектов | практические занятия | 4 |
| лабораторные работы | 4 |
| самостоятельная работа | 10 |
| II. | Продвинутые техники программирования | практические занятия | 7 |
| лабораторные работы | 7 |
| самостоятельная работа | 15 |
| III. | Программирование на GPGPU | практические занятия | 4 |
| лабораторные работы | 4 |
| самостоятельная работа | 12 |
| IV. | Промежуточная аттестация | самостоятельная работа | 3 |
| зачёт | 2 |

Раздел 1: Особенности исследовательских проектов.

1. Жизненный цикл исследовательских проектов и возможные пути развития. Отличие от прикладных/промышленных проектов и сходство с ними. Цели и задачи исследовательских проектов.
2. Эксперименты: воспроизводимость, анализ результатов, оформление результатов. Примеры соответствующих инструментальных средств. Цели и задачи экспериментов. Экспериментальное исследование, сравнение, проверка гипотезы.

*Домашняя работа 1*

1. Оформление результатов в виде текста: статья, технический отчёт и другие типы текстов. Особенности процесса их написания. Типичная структура и особенности. Курсовая/диплом как научная работа и отчёт по ним как научный текст.
2. Представление результатов в виде презентации, доклада. Различные виды докладов, презентаций. Типичные структуры. Пример презентации для курсовой.

*Домашняя работа 2*

Раздел 2: Продвинутые техники программирования

1. Метапрограммирование вообще и в F# в частности. Понятие метапрограммирования. Подходы к реализации техник метапрограммирования. Программирование времени выполнения и времени компиляции. Примеры: шаблоны в С++ (тьюринг-полнота), системы макросов, вычислимые выражения. Встраивание языков.
2. Рефлексия, вообще и в .NET. Сборки в .NET, сильные и слабые имена сборок. Получение информации о сборках, типах, полях, методах и т.д., создание экземпляров объектов, вызов методов. Компиляция F#-кода во время выполнения.

*Домашняя работа 3 логгер, листбилдер*

1. Разбор домашней работы 2
2. Поставщики типов: их типы, решаемые с их помощью задачи. Особенности использования. Особенности создания. Примеры готовых поставщиков и их использования.
3. F# quotations. Трансформация кода во время выполнения. Возможности и ограничения. Типизированный и нетипизированный варианты. Примеры использования.
4. Событийно-ориентированное программирование, реактивное программирование RxExtension.

*Домашняя работа 4* *Встраивание языка, RX для IDE.*

1. Проверочная работа

Раздел 3: Программирование на GPGPU

1. Основы. Архитектура, логическая модель вычислителя, плюсы/минусы, сферы применения.
2. OpenCL, логическая и физическая модели, переносимость, ядра, атомарные операции, сравнение с CUDA. Основы языка OpenCL C. Примеры простых ядер.
3. Программирование на GPGPU с использованием высокоуровневых средств. Программирование GPGPU на F# как пример метапрограммирования. Особенности и проблемы использования высокоуровневых средств.

*Домашняя работа 5 GPU!!!!*

1. Проверочная работа.

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.

**Вариант реализации 3: …**

Период обучения (модуль): семестр 1.

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Наименование темы (раздела, части) | Вид учебных занятий | Количество часов |
| I. |  | практические занятия |  |
| лабораторные работы |  |
| контрольные работы |  |
| N. | Промежуточная аттестация | зачёт | 2 |

Раздел 1:

Раздел 2:

Период обучения (модуль): семестр 2.

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Наименование темы (раздела, части) | Вид учебных занятий | Количество часов |
| I. |  | практические занятия |  |
| лабораторные работы |  |
| контрольные работы |  |
| N. | Промежуточная аттестация | зачёт | 2 |

Раздел 1:

Раздел 2:

Период обучения (модуль): семестр 3.

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Наименование темы (раздела, части) | Вид учебных занятий | Количество часов |
| I. |  | практические занятия |  |
| лабораторные работы |  |
| контрольные работы |  |
| N. | Промежуточная аттестация | зачёт | 2 |

Раздел 1:

Раздел 2:

**Вариант реализации 4: …**

Период обучения (модуль): семестр 1.

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Наименование темы (раздела, части) | Вид учебных занятий | Количество часов |
| I. |  | практические занятия |  |
| лабораторные работы |  |
| контрольные работы |  |
| N. | Промежуточная аттестация | зачёт | 2 |

Раздел 1:

Раздел 2:

Период обучения (модуль): семестр 2.

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Наименование темы (раздела, части) | Вид учебных занятий | Количество часов |
| I. |  | практические занятия |  |
| лабораторные работы |  |
| контрольные работы |  |
| N. | Промежуточная аттестация | зачёт | 2 |

Раздел 1:

Раздел 2:

Период обучения (модуль): семестр 3.

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Наименование темы (раздела, части) | Вид учебных занятий | Количество часов |
| I. |  | практические занятия |  |
| лабораторные работы |  |
| контрольные работы |  |
| N. | Промежуточная аттестация | зачёт | 2 |

Раздел 1:

Раздел 2:

**Вариант реализации 5: …**

Период обучения (модуль): семестр 1.

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Наименование темы (раздела, части) | Вид учебных занятий | Количество часов |
| I. |  | практические занятия |  |
| лабораторные работы |  |
| контрольные работы |  |
| N. | Промежуточная аттестация | зачёт | 2 |

Раздел 1:

Раздел 2:

Период обучения (модуль): семестр 2.

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Наименование темы (раздела, части) | Вид учебных занятий | Количество часов |
| I. |  | практические занятия |  |
| лабораторные работы |  |
| контрольные работы |  |
| N. | Промежуточная аттестация | зачёт | 2 |

Раздел 1:

Раздел 2:

Период обучения (модуль): семестр 3.

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Наименование темы (раздела, части) | Вид учебных занятий | Количество часов |
| I. |  | практические занятия |  |
| лабораторные работы |  |
| контрольные работы |  |
| N. | Промежуточная аттестация | зачёт | 2 |

Раздел 1:

Раздел 2:

**Вариант реализации 6: …**

Период обучения (модуль): семестр 1.

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Наименование темы (раздела, части) | Вид учебных занятий | Количество часов |
| I. |  | практические занятия |  |
| лабораторные работы |  |
| контрольные работы |  |
| N. | Промежуточная аттестация | зачёт | 2 |

Раздел 1:

Раздел 2:

Период обучения (модуль): семестр 2.

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Наименование темы (раздела, части) | Вид учебных занятий | Количество часов |
| I. |  | практические занятия |  |
| лабораторные работы |  |
| контрольные работы |  |
| N. | Промежуточная аттестация | зачёт | 2 |

Раздел 1:

Раздел 2:

Период обучения (модуль): семестр 3.

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Наименование темы (раздела, части) | Вид учебных занятий | Количество часов |
| I. |  | практические занятия |  |
| лабораторные работы |  |
| контрольные работы |  |
| N. | Промежуточная аттестация | зачёт | 2 |

Раздел 1:

Раздел 2:

**Вариант реализации 7: …**

Период обучения (модуль): семестр 1.

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Наименование темы (раздела, части) | Вид учебных занятий | Количество часов |
| I. |  | практические занятия |  |
| лабораторные работы |  |
| контрольные работы |  |
| N. | Промежуточная аттестация | зачёт | 2 |

Раздел 1:

Раздел 2:

Период обучения (модуль): семестр 2.

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Наименование темы (раздела, части) | Вид учебных занятий | Количество часов |
| I. |  | практические занятия |  |
| лабораторные работы |  |
| контрольные работы |  |
| N. | Промежуточная аттестация | зачёт | 2 |

Раздел 1:

Раздел 2:

Период обучения (модуль): семестр 3.

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Наименование темы (раздела, части) | Вид учебных занятий | Количество часов |
| I. |  | практические занятия |  |
| лабораторные работы |  |
| контрольные работы |  |
| N. | Промежуточная аттестация | зачёт | 2 |

Раздел 1:

Раздел 2:

**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 начале курса проводится входное тестирование, по результатам которого и на основании личных предпочтений обучающиеся распределяются на варианты реализации дисциплины. Входное тестирование проводится посредством системы Blackboard либо в аудитории. Обучающимся предлагается ответить на ряд вопросов и решить задачи (примеры заданий приведены в разделе 3.1.4), для решения задач разрешается использовать любой язык программирования или описывать решение словами. Срок выполнения работы — два академических часа. В ходе выполнения работы запрещается пользоваться источниками и общаться.

В формировании итоговой оценки по курсу участвуют домашние задания, контрольные работы, зачётная работа, 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: промышленное программирование, C, 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. Предусмотрено как минимум одно переписывание каждой контрольной, где обучающиеся могут попытаться улучшить свои баллы. Конкретное число переписываний определяется преподавателем, но переписывания проводятся не чаще, чем один раз в неделю.

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 набора файлов, либо 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-исправления повторяется, либо до достижения максимального балла за задачу, либо удовлетворения обучающегося текущими баллами.

Зачётные работы проводятся аналогично контрольным работам, с тем отличием, что они имеют ограниченное количество попыток. Две попытки написания зачётной работы предоставляются в конце каждого семестра, третья попытка — в день промежуточной аттестации по расписанию, последующие попытки — на пересдаче и комиссии.

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: Прикладное и исследовательское программирование на примере F#.**

Контрольные и зачётные 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. Предусмотрено как минимум одно переписывание каждой контрольной, где обучающиеся могут попытаться улучшить свои баллы. Конкретное число переписываний определяется преподавателем, но переписывания проводятся не чаще, чем один раз в неделю.

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). Для каждой домашней работы срок сдачи оговаривается отдельно, но обычно это одна или две недели. Есть три попытки сдать каждую домашнюю работу**.** Соответственно, разбиение задач между попытками ни на что не влияет (можно сперва сдать одну задачу из шести — первая попытка, получить на неё замечания, во вторую исправить замечания и сдать ещё пять задач) за исключением случая если задачи являются зависимыми. Тогда порядок сдачи задаётся зависимостями между задачами. Попытка — это запрос на ревью реквеста или изменения в ветке, выделенной под данную работу. Истечение срока приоритетно: если срок сдачи домашней работы истёк, но ещё не все попытки израсходованы, сдавать задачи после срока нельзя. Важно, что все задачи должны снабжаться тестами, которые запускаются автоматически на сервере. Проверка преподавателем производится только при успешном прохождении сборки и тестирования на сервере.

Все отчёты готовятся в TeX. Публикуются как исходники, так и скомпилированные pdf-файлы.

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.

**Вариант реализации 3: …**

**Вариант реализации 4: …**

**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: промышленное программирование, C, C#.**

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

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. Например, если было изучено модульное тестирование, ожидается, что решение каждой домашней задачи сопровождается модульными тестами.

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: Прикладное и исследовательское программирование на примере F#.**

Итоговый процент освоения дисциплины складывается из баллов за домашние задания, баллов за контрольные работы и баллов за зачётную работу.

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. 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.

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. Например, если было изучено модульное тестирование, ожидается, что решение каждой домашней задачи сопровождается модульными тестами.

Для вычисления итогового процента освоения дисциплины 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 3: …**

**Вариант реализации 4: …**

**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-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.

**ОПК-4** — способен участвовать в разработке технической документации программных продуктов и программных комплексов.

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

**ОПК-5** — способен инсталировать и сопровожать программное обеспеченение для информационных систем и баз данных, в том числе отечественного производства.

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

**ПКП-3** — способен преподавать математику и информатику в средней школе, специальных учебных заведениях на основе полученного фундаментального образования и научного мировоззрения.

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, развиваемые дисциплиной: нет***

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

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

**Контрольно-измерительные материалы входного тестирования**

Примеры вопросов:

* 1. Фамилия, имя, отчество
  2. Электронная почта
  3. На какой из вариантов реализации Вы бы хотели попасть?
     + промышленное программирование, C, C#
     + …
     + …
     + …
     + …
     + …
     + …
     + …
  4. Город, школа, учитель математики, учитель информатики
  5. Как Вы выбрали направление?
     + из материалов в приемной комиссии
     + узнал о специальности от старших знакомых
     + от руководства вашей школы
     + от "агитаторов" СПбГУ в вашем городе
     + из Сети
     + не прошел(а) на другую специальность
     + выбор сделали родители
     + другое \_\_\_\_\_\_\_
  6. Какие вузы и направления ещё рассматривали?
  7. Степень владения языками программирования. Перечислите языки, с которыми знакомы, с указанием, насколько Вы оцениваете свои навыки владения каждым. 1-10, где 1 — слышал(а), 10 – владею виртуозно. Если хочется написать "0", просто не указывайте. Например, “C — 5; C++ — 2; Haskell — 10”.
  8. Общий уровень знания информатики: общеизвестные алгоритмы, умение их адаптировать к задаче и применять, умение придумывать свои. Оцените по десятибалльной шкале, от 0 — не умею включать компьютер, до 10 — знаю и умею всё, что можно знать и уметь.
  9. Знание иностранных языков. Укажите языки, которые Вы знаете на уровне, достаточном, чтобы понимать несложные технические тексты, не обращаясь к словарю слишком часто.
  10. Чем бы вы хотели заняться после окончания Университета? Выберите один-два варианта:
      + Пойду на второе высшее
      + Пойду работать в крупной компании
      + Какое окончание! Пойду в магистратуру, а потом в аспирантуру
      + Сделаю свой стартап
      + Выйду замуж/женюсь
  11. Какими информационными ресурсами пользовались этим летом? Выберите все подходящие варианты:
      + Wikipedia.org
      + Habrahabr.ru / geektimes.ru
      + hackernews.org
      + stackoverflow.com / stackexchange.com
      + другое \_\_\_\_\_\_\_\_\_\_\_\_
  12. Что ещё хотели бы о себе сообщить? Что угодно, что, с вашей точки зрения, нам полезно будет знать о Вас. Например: не пью, не курю, не ем теплокровных, занимаюсь спортом, слушаю только классическую музыку, участвую в благотворительных мероприятиях, обладаю чрезвычайно высокими моральными качествами...

Примеры задач:

1. Написать алгоритм нахождения неполного частного от деления a на b (целые числа), используя только операции сложения, вычитания и умножения.
2. Дан массив целых чисел x[1]…x[m+n], рассматриваемый как соединение двух его отрезков: начала x[1]…x[m] длины m и конца x[m+1]…x[m+n] длины n. Не используя дополнительных массивов, переставить местами начало и конец.

***Критерии оценивания*:** вопросы анкеты не оцениваются. Решение каждой задачи оценивается по шкале от 0 (нет решения) до 10 (решение работоспособно, аккуратно реализовано, алгоритмически оптимально). Оценки за каждую задачу складываются для получения итоговой оценки.

**Контрольно-измерительные материалы текущего контроля и промежуточной аттестации**

Контрольно-измерительные материалы различны для каждого варианта реализации дисциплины.

**Вариант реализации 1: промышленное программирование, C, C#.**

**Семестр 1.**

***Домашняя работа 1:***

1. Написать программу, считающую значение формулы x4 + x3 + x2 + x + 1 за два умножения.
2. Написать программу нахождения неполного частного от деления a на b (целые числа), используя только операции сложения, вычитания и умножения.
3. Дан массив целых чисел x[1] ... x[m + n], рассматриваемый как соединение двух его отрезков: начала x[1] ... x[m] длины m и конца x[m + 1] ... x[m + n] длины n. Не используя дополнительных массивов, переставить начало и конец (обращением двух частей массива, а потом его самого).
4. Посчитать число "счастливых билетов" (билет считается "счастливым", если сумма первых трёх цифр его номера равна сумме трёх последних), подсчётом числа билетов с заданной суммой трёх цифр.
5. Написать программу проверки баланса скобок в исходной строке (т.е. число открывающих скобок равно числу закрывающих и выполняется правило вложенности скобок).
6. Заданы две строки: S и S1. Найти количество вхождений S1 в S как подстроки.
7. Написать программу, печатающую все простые числа, не превосходящие заданного числа.
8. Написать программу, считающую количество нулевых элементов в массиве.

***Проверяемые компетенции*:** ОПК-2, ОПК-5, ПКП-4

***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 вычисление чисел Фибоначчи рекурсивно (и убедиться, что при n ~ 37 уже заметно медленно), реализовать итеративно, почувствовать разницу.
2. Реализовать возведение в степень — в лоб (за линейное время) и за О(log n).
3. Написать сортировки пузырьком и подсчётом.
4. Написать программу, которая заполняет массив случайными значениями (с использованием функции rand из stdlib.h), потом преобразует его без использования дополнительных массивов так, что в начале массива будут элементы, меньшие первого, а в конце — большие либо равные первому. Программа должна работать за линейное время.

***Проверяемые компетенции*:** ОПК-2, ОПК-5, ПКП-4

***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 3:***

1. Реализовать qsort, который для сортировки кусков массива размером меньше 10 использует сортировку вставкой.
2. Получить с клавиатуры 2 числа, n и k, сгенерировать случайно массив из n чисел, сгенерировать k случайных целых чисел, для каждого из них проверить, содержится ли оно в массиве. Надо придумать алгоритм с временной сложностью O(n log n + k log n), или лучший.
3. Найти наиболее часто встречающийся элемент в массиве быстрее, чем за O(n2). Если таких элементов несколько, надо вывести любой из них.
4. Завести аккаунт с разумным именем на https://github.com. Выложить туда в master уже зачтённые задачи (при этом придумав разумную структуру папок), все незачтённые выложить в отдельные ветки и сделать пуллреквесты. Имеет смысл посмотреть презентацию, https://git-scm.com/book/ru/v1 и поставить графический клиент типа https://tortoisegit.org/ для облегчения жизни. Не бояться экспериментировать со своим репозиторием и спрашивать препода

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

***Критерии оценивания*:** решение задач 1-3 оценивается по шкале от 0 (нет решения или решение имеет существенные недостатки) до 3 (решение работоспособно, аккуратно реализовано, алгоритмически аккуратно). Решение задачи 4 оценивается по шкале от 0 (не выполнено) до 2 (выполнено, структура папок адекватна, пуллреквесты сделаны).

***Домашняя работа 4:***

1. Ввести два числа, перевести в двоичное представление в дополнительном коде и напечатать, сложить в столбик в двоичном представлении, вывести сумму, перевести в десятичное, вывести сумму в десятичном виде. Все сообщения писать по-русски (рекомендуется использовать функцию setlocale, чтобы сообщения выводились по-русски и под Windows тоже).
2. Переделать задачу 3 из прошлого задания так, чтобы сортировка была в отдельном модуле и читала входные данные из файла.
3. Написать программу — телефонный справочник. Она должна уметь хранить имена и номера телефонов, в интерактивном режиме осуществлять следующие операции:
   * 0 — выйти
   * 1 — добавить запись (имя и телефон)
   * 2 — распечатать все имеющиеся записи
   * 3 — найти телефон по имени
   * 4 — найти имя по телефону
   * 5 — сохранить текущие данные в файл

При запуске программа должна читать данные из файла (того самого, в который сохраняет данные), если файла нет — начинать с пустой базы номеров. Размер базы ограничен сотней записей.

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

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

***Домашняя работа 5:***

1. Написать программу для вычисления арифметического выражения в постфиксной форме. С клавиатуры вводится последовательность цифр (для простоты) и операций +, -, \*, /, представляющая выражение в постфиксной форме, должен выводиться результат вычисления. Например, на тесте 9 6 - 1 2 + \* должно получиться 9.
2. Написать программу проверки баланса скобок в строке, скобки могут быть трёх видов: (), [], {}. Скобочная последовательность вида ({)} считается некорректной, ({}) — корректной.
3. Написать программу, преобразующую выражение из инфиксной формы в постфиксную. В выражении могут быть знаки +, -, \*, /, скобки и цифры. Пример: (1 + 1) \* 2 должно преобразовываться в 1 1 + 2 \*. Алгоритм перевода предлагается найти самостоятельно (алгоритм «сортировочной станции» Э. Дейкстры).

Задачи 1, 2 и 3 решаются с помощью стека — его надо реализовать единожды в отдельном модуле, и использовать во всех этих задачах. Комментарии ко всем функциям из заголовочного файла обязательны.

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

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

***Домашняя работа 6:***

1. Написать программу, которая в диалоговом режиме позволяет осуществлять следующие операции:
   * 0 — выйти
   * 1 — добавить значение в сортированный список
   * 2 — удалить значение из списка
   * 3 — распечатать список

Все операции должны сохранять сортированность. Начинаем с пустого списка.

1. "Считалочка" — отряд из 41-го сикария, защищавший галилейскую крепость Массада, не пожелал сдаваться в плен блокировавшим его превосходящим силам римлян. Сикарии стали в круг и договорились, что каждые два воина будут убивать третьего, пока не погибнут все. Самоубийство – тяжкий грех, но тот, кто в конце концов останется последним, должен будет его совершить. Иосиф Флавий, командовавший этим отрядом, якобы быстро рассчитал, где нужно стать ему и его другу, чтобы остаться последними, но не для того, чтобы убить друг друга, а чтобы сдать крепость римлянам. В нашем случае участвует n воинов и убивают каждого m-го. Требуется определить номер k начальной позиции воина, который должен будет остаться последним. Считать с помощью циклического списка.
2. Реализовать сортировку слиянием. Во входном файле последовательность записей «имя–номер телефона». Программа должна отсортировать эти записи либо по имени, либо по номеру телефона, в зависимости от выбора пользователя, и вывести результат на экран. Количество записей заранее неизвестно, так что надо реализовывать списками на указателях.

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

***Критерии оценивания*:** решение задач 1 и 2 оценивается по шкале от 0 (нет решения или решение имеет существенные недостатки) до 3 (решение работоспособно, аккуратно реализовано, алгоритмически аккуратно), решение задачи 3 оценивается по шкале от 0 (нет решения или решение имеет существенные недостатки) до 4 (решение работоспособно, аккуратно реализовано, отсутствуют утечки памяти).

***Домашняя работа 7:***

1. Реализовать АТД "словарь" на основе двоичного дерева поиска, хранящий ключи типа int и связанные с ними значения типа char \*. Должны поддерживаться следующие операции:
   * Добавить значение по заданному ключу в словарь. Если такой ключ уже есть, значение заменяется на новое.
   * Получить значение по заданному ключу из словаря. Если такого ключа нет, возвращается nullptr.
   * Проверить наличие заданного ключа в словаре.
   * Удалить заданный ключ и связанное с ним значение из словаря. Если такого ключа нет, функция ничего не делает.

Программа должна работать в интерактивном режиме, то есть пользователю должно предоставляться меню, позволяющее выполнить эти операции

1. По дереву разбора арифметического выражения вычислить его значение. Дерево разбора хранится в файле в виде (<операция> <операнд1> <операнд2>), где <операнд1> и <операнд2> сами могут быть деревьями, либо числами. Например, выражение (1 + 1) \* 2 представляется в виде (\* (+ 1 1) 2). Должны поддерживаться операции +, -, \*, / и целые числа в качестве аргументов. Требуется построить дерево в явном виде, распечатать его (не обязательно так же, как в файле), и посчитать значение выражения обходом дерева. Может быть полезна функция ungetc (если не '(', возвращаем символ в поток и читаем число fscanf-ом). Можно считать, что входной файл корректен. Пример - по входному файлу (\* (+ 1 1) 2) может печататься ( \* ( + 1 1 ) 2 ) и выводиться 4.

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

***Критерии оценивания*:** решение каждой задачи оценивается по шкале от 0 (нет решения или решение имеет существенные недостатки) до 5 (решение работоспособно, аккуратно реализовано, алгоритмически аккуратно, отсутствуют утечки памяти).

***Домашняя работа 8:***

1. Реализовать словарь с ключами и значениями типа char \* на основе АВЛ-дерева. Должны поддерживаться следующие операции.
   * Добавить значение по заданному ключу в словарь. Если такой ключ уже есть, значение заменяется на новое.
   * Получить значение по заданному ключу из словаря. Если такого ключа нет, возвращается nullptr.
   * Проверить наличие заданного ключа.
   * Удалить заданный ключ и связанное с ним значение из словаря. Если такого ключа нет, функция ничего не делает.

Программа должна работать в интерактивном режиме, то есть пользователю должно предоставляться меню, позволяющее выполнить эти операции.

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

***Критерии оценивания*:** решение задачи оценивается по шкале от 0 (нет решения или решение имеет существенные недостатки) до 8 (решение работоспособно, аккуратно реализовано, алгоритмически аккуратно, отсутствуют утечки памяти).

***Домашняя работа 9:***

1. Посчитать частоты встречаемости слов в тексте с помощью хеш-таблицы. На входе файл с текстом, вывести на консоль все слова, встречающиеся в этом тексте с количеством раз, которое встречается каждое слово. Словом считается последовательность символов, разделённая пробелами, разные словоформы считаются разными словами. Хеш-таблицу реализовать в отдельном модуле, использующем модуль «Список». Подсчитать и вывести также коэффициент заполнения хеш-таблицы, максимальную и среднюю длину списка в сегменте таблицы.

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

***Критерии оценивания*:** решение задачи оценивается по шкале от 0 (нет решения или решение имеет существенные недостатки) до 5 (решение работоспособно, аккуратно реализовано, алгоритмически аккуратно, отсутствуют утечки памяти).

***Домашняя работа 10:***

1. Есть множество городов и дороги, связывающие эти города. Для каждой дороги задана её длина. Задача — распределить города между государствами по такому алгоритму: задаются k столиц каждого государства, далее по очереди каждому государству добавляется ближайший незанятый город, непосредственно связанный дорогой с каким-либо городом, уже принадлежащим государству (столицей или каким-либо городом, добавленным на одном из предыдущих шагов). Процесс продолжается до тех пор, пока все города не будут распределены. Граф дорог связный. Во входном файле: n — число городов и m — число дорог. Далее следуют сами дороги в формате: i j len, i и j — номера городов, len — длина дороги. Далее задано число k — число столиц, далее — k чисел — номера столиц. Надо вывести на консоль номера государств и списки городов, принадлежащих государствам.

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

***Критерии оценивания*:** решение задачи оценивается по шкале от 0 (нет решения или решение имеет существенные недостатки) до 8 (решение работоспособно, аккуратно реализовано, алгоритмически аккуратно, отсутствуют утечки памяти).

***Домашняя работа 11:***

1. Реализовать поиск подстроки любым из следующих алгоритмов: Бойера-Мура, Кнута-Моррисса-Пратта, Рабина-Карпа. Из файла читается текст, с консоли — строка, программа должна выводить на консоль позицию первого вхождения введённой строки в файле.

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

***Критерии оценивания*:** решение задачи оценивается по шкале от 0 (нет решения или решение имеет существенные недостатки) до 4 (решение работоспособно, аккуратно реализовано, алгоритмически аккуратно, отсутствуют утечки памяти).

***Домашняя работа 12:***

1. Реализовать с помощью switch по состоянию лексический анализатор, проверяющий, является ли введённая последовательность символов вещественным числом (вещественное число задаётся регулярным выражением digit+ (. digit+)? (E(+ | -)? digit+)?, где digit - [0..9]).
2. С помощью ДКА с явной таблицей состояний, заданной в файле, вывести на консоль все комментарии С++ вида /\* комментарий \*/ из входного файла (вместе с символами "/\* \*/").

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

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

***Темы докладов***

1. Алгоритм Бойера-Мура
2. Алгоритм Кнута-Морриса-Пратта
3. Алгоритм поиска А\*
4. Алгоритм Рабина-Карпа
5. Визуализатор графов GraphViz и язык dot

***Проверяемые компетенции*:** OПК-4, ПКП-3, УКБ-3

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

***Примеры условий контрольных работ:***

***Контрольная работа 1***

1. Реализовать сортировку вставками только чётных элементов массива. Например, массив [6, 4, 3, 2, 1] должен быть отсортирован как [2, 4, 3, 6, 1].
2. Реализовать сортировку Шелла.
3. В некоторых языках программирования однострочные комментарии задаются не //, как в С++, а символом ";" (комментарий начинается с ; и заканчивается концом строки). Задача — вывести на консоль все комментарии такого вида из входного файла (вместе с символом ";"). До комментария в строке может быть значимый текст, его выводить не надо. Конец строки представляется символом \n, могут быть полезны функции fgetc и feof.

***Контрольная работа 2***

1. Дан файл, в котором встречаются даты. Каждая дата — это число, месяц и год (например, 09.11.2009). Найти наибольшую дату.
2. Написать программу, которая в диалоговом режиме позволяет осуществлять следующие операции:
   * 0 — выйти
   * 1 — добавить число на вершину стека
   * 2 — удалить число с вершины стека
   * 3 — дублировать число на вершине стека, то есть положить на стек его копию
   * 4 — распечатать стек

***Зачётная работа***

1. Дана строка, изображающая двоичную запись целого положительного числа. Вывести строку, изображающую десятичную запись этого же числа
2. 1953 год. В связи с усилением влияния Запада и необходимостью предпринять ответные меры, Иосиф Сталин разрабатывает новый эффективный алгоритм внутрипартийной чистки: сначала расстреливаются n самых опасных членов, а затем оставшиеся упорядочиваются в алфавитном порядке и первые m отправляются в Сибирь. Л.П. Берия уже успел провести предварительные работы: определил степень лояльности к СССР среди членов партии и предоставил списки генеральному секретарю. На входе файл формата «фамилия — степень лояльности». n и m вводятся с клавиатуры. Задача: определить, кто будет расстрелян, кого отправят в Сибирь, кто останется невредим. Пока невредим.
3. Реализовать конечный автомат, принимающий строки, задаваемые следующим регулярным выражением: [A-Za-z] ([A-Za-z] | [0-9] | \_ )\*

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

***Критерии оценивания*:** решение задачи оценивается от 0 до 10 баллов, где 0 — полное отсутствие решения, 10 — правильное решение, демонстрирующее все ожидаемые навыки (комментарии, тесты, навык использования системы контроля версий, отсутствие утечек памяти).

**Семестр 2.**

***Домашняя работа 1:***

1. Посчитать факториал
2. Посчитать числа Фибоначчи
3. Отсортировать массив какой-либо из сортировок
4. Дан массив размерностью N x N, N — нечетное число. Вывести элементы массива при обходе его по спирали, начиная с центра
5. Отсортировать столбцы матрицы по первым элементам

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

***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. Написать связный список в виде класса. От списка хочется:
   * добавлять/удалять элемент по произвольной позиции, задаваемой целым числом;
   * узнавать размер, проверять на пустоту;
   * получать или устанавливать значение элемента по позиции, задаваемой целым числом.
2. Написать хеш-таблицу в виде класса с использованием класса-списка из первой задачи. Должно быть можно добавлять значение в хеш-таблицу, удалять и проверять на принадлежность.
3. Реализовать стековый калькулятор (класс, реализующий выполнение операций +, -, \*, / над арифметическим выражением в виде строки в постфиксной записи). Строка уже дана в обратной польской записи (например, 1 2 3 + \*). Стек реализовать двумя способами (например, массивом или списком) в двух разных классах на основе одного интерфейса. Стековый калькулятор должен знать только про интерфейс стека. В Main надо спросить у пользователя, какой из вариантов стека он хочет, в зависимости от выбора создаётся объект одной из двух реализаций и передаётся калькулятору.

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

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

***Домашняя работа 3:***

1. Написать юнит-тесты к задаче 3 из предыдущего задания
2. Модифицировать хеш-таблицу из задачи 2 предыдущей работы так, чтобы хеш-функцию можно было менять в зависимости от выбора пользователя, причём хеш-функцию должно быть можно передавать из использующего хеш-таблицу кода в виде объекта некоторого класса, реализующего некоторый интерфейс. Хеш-функцию должно быть можно менять во время работы. Юнит-тесты и комментарии в формате XML Documentation обязательны.

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

***Критерии оценивания*:** решение каждой задачи оценивается по шкале от 0 (нет решения или решение имеет существенные недостатки) до 3 (решение работоспособно, аккуратно реализовано, алгоритмически аккуратно, модульные тесты не содержат дублирующегося кода).

***Домашняя работа 4:***

1. Решить задачу о вычислении выражения по дереву разбора из прошлого семестра. Реализовать иерархию классов, описывающих дерево разбора, используя их, реализовать класс, вычисляющий значение выражения по дереву. Классы, представляющие операнды и операторы, должны сами уметь себя вычислять и печатать.

Исходное условие:

По дереву разбора арифметического выражения вычислить его значение. Дерево разбора хранится в файле в виде (<операция> <операнд1> <операнд2>), где <операнд1> и <операнд2> сами могут быть деревьями, либо числами. Например, выражение (1 + 1) \* 2 представляется в виде (\* (+ 1 1) 2). Должны поддерживаться операции +, -, \*, / и целые числа в качестве аргументов. Требуется построить дерево в явном виде, распечатать его (не обязательно так же, как в файле), и посчитать значение выражения обходом дерева. Может быть полезна функция ungetc (если не '(', возвращаем символ в поток и читаем число fscanf-ом). Можно считать, что входной файл корректен. Пример - по входному файлу (\* (+ 1 1) 2) может печататься ( \* ( + 1 1 ) 2 ) и выводиться 4.

1. Унаследовавшись от класса список, реализовать класс UniqueList, который не содержит повторяющихся значений. Реализовать классы исключений, которые генерируются при попытке добавления в такой список уже существующего или при попытке удаления несуществующего элемента.

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

***Критерии оценивания*:** решение задачи 1 оценивается по шкале от 0 (нет решения или решение имеет существенные недостатки) до 5 (решение работоспособно, аккуратно реализовано, алгоритмически аккуратно), решение задачи 2 оценивается по тем же критериям, но по шкале от 0 до 3.

***Домашняя работа 5:***

1. Настроить сборку на AppVeyor одной из предыдущих задач. В качестве решения приложить ссылку на успешно собранный пуллреквест с appveyor.yml..

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

***Критерии оценивания*:** решение задачи оценивается по шкале от 0 (нет) до 3 (решение работоспособно и адекватно, модульные тесты запускаются в CI-системе).

***Домашняя работа 6:***

1. Реализовать функции Map, Filter и Fold:

* Map принимает список и функцию, преобразующую элемент списка. Возвращаться должен список, полученный применением переданной функции к каждому элементу переданного списка. Например, Map(new List<int>() {1, 2, 3}, x => x \* 2) должен возвращать список [2; 4; 6].
* Filter принимает список и функцию, возвращающую булевое значение по элементу списка. Возвращаться должен список, составленный из тех элементов переданного списка, для которых переданная функция вернула true.
* Fold принимает список, начальное значение и функцию, которая берёт текущее накопленное значение и текущий элемент списка, и возвращает следующее накопленное значение. Сама Fold возвращает накопленное значение, получившееся после всего прохода списка. Например, Fold(new List<int>() {1, 2, 3}, 1, (acc, elem) => acc \* elem) работала бы так: сначала в acc клался бы 1, потом умножался бы на 1, потом результат (1) умножался бы на 2, потом результат (2) умножался бы на 3, потом результат (6) возвращался бы в качестве ответа.

Если умеете генерики, подумайте, как сделать тип функций наиболее общим (за +1 балл к оценке).

1. На базе класса, генерирующего события по нажатию на клавиши управления курсором (EventLoop с пары), реализовать консольное приложение, позволяющее управлять персонажем, перемещающимся по карте. Карта состоит из свободного пространства и стен, и должна грузиться из файла. Приложение должно отображать карту и персонажа (символом @) в окне консоли, и позволять персонажу перемещаться по карте, реагируя на клавиши управления курсором. Будут полезны свойства Console.CursorLeft и Console.CursorTop.

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

***Критерии оценивания*:** решение задачи 1 оценивается по шкале от 0 (нет решения или решение имеет существенные недостатки) до 2 (решение работоспособно, аккуратно реализовано, алгоритмически аккуратно). Решение задачи 2 оценивается по тем же критериям, но по шкале от 0 до 4 (при этом дополнительно проверяется качество тестового покрытия).

***Домашняя работа 7:***

1. Написать калькулятор с пользовательским интерфейсом (по примеру стандартного калькулятора вашей любимой операционной системы)
2. Сделать часы (приложение, показывающее текущее системное время) со стрелками (часовой, минутной и секундной). Здесь юнит-тесты можно не писать.

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

***Критерии оценивания*:** решение задачи 1 оценивается по шкале от 0 (нет решения или решение имеет существенные недостатки) до 5 (решение работоспособно, аккуратно реализовано, отвечает требованиям на качество пользовательских интерфейсов). Решение задачи 2 оценивается по тем же критериям, но по шкале от 0 до 2.

***Домашняя работа 8:***

1. Создать генерик-класс, реализующий АТД "Множество". Множество должно реализовывать интерфейс System.Collections.Generic.ISet. Ожидается асимптотическая трудоёмкость основных операций не хуже в среднем логарифмической (то есть несбалансированное двоичное дерево тоже пойдёт). Проверить в тестах, что по множеству можно ходить foreach.

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

***Критерии оценивания*:** решение задачи оценивается по шкале от 0 (нет решения или решение имеет существенные недостатки) до 6 (решение работоспособно, аккуратно реализовано, соответствует требованиям и рекомендациям по разработке библиотек).

***Домашняя работа 9:***

1. Нарисовать на UML диаграмму классов для игры «реверси», с несколькими видами «ботов», которые могли бы играть вместо человека. Диаграмма должна быть довольно подробным проектом системы. Реализовывать этот проект не нужно. Желательно пользоваться средствами визуального моделирования наподобие Visual Paradigm (https://www.visual-paradigm.com/download/community.jsp), присылать скриншоты с диаграммой и исходники (файлы .vpp в случае Visual Paradigm)

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

***Критерии оценивания*:** решение задачи оценивается по шкале от 0 (нет решения или решение имеет существенные недостатки) до 4 (решение архитектурно корректно, не вызывает существенных вопросов по реализации, аккуратно оформлено).

***Домашняя работа 10:***

1. Сделать с помощью Unity очередной клон компьютерной игры SCP-087 (https://ru.wikipedia.org/wiki/SCP-087). Игра в жанре сурвайвал-хоррор должна представлять собой спуск с фонариком по тёмной бесконечной лестнице, при этом с вероятностью N% (где N передаётся как параметр командной строки) на очередной лестничной клетке появляется ужасное нечто и убивает главного героя, на чём игра заканчивается, показав сообщение, что игрок проиграл. Моделировать главного героя не требуется, "летающей камеры" достаточно, в качестве ужасного нечто можно использовать любую достаточно ужасную 3д-модель (шар, куб или чайник вполне подойдут). Желательно (хотя и не обязательно) звуковое сопровождение, создающее гнятущую атмосферу экзистенциального ужаса, и соответствующие текстуры на стенах. См. также http://scpfoundation.net/scp-087.

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

***Критерии оценивания*:** решение задачи оценивается по шкале от 0 (нет решения или решение имеет существенные недостатки) до 6 (решение работоспособно, играбельно, не вызывает проблем при сборке и запуске, эстетично оформлено). Баллы за данную задачу не учитываются при вычислении максимального балла в семестре, так что эта задача «необязательная».

***Темы докладов***

* Тестирование пользовательского интерфейса, Coded UI и White
* Сравнение платформ .NET Core, .NET Framework, Mono
* yield return, ленивые вычисления
* Unity

***Проверяемые компетенции*:** OПК-4, ПКП-3, УКБ-3

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

***Примеры условий контрольных работ:***

***Контрольная работа***

1. Реализовать программу с пользовательским интерфейсом: на форме есть кнопка и индикатор прогресса, при нажатии на кнопку индикатор прогресса должен начать заполняться (сам увеличивая своё значение через равные промежутки времени), после того как он достигнет 100%, должна появиться кнопка, позволяющая закрыть форму

***Зачётная работа***

1. Реализовать кодирование алгоритмом Move-To-Front строк английского алфавита. На вход программе подаётся строка, на выход — закодированная MTF последовательность чисел. Например, по строке banana должна выдаваться кодовая последовательность [1,1,13,1,1,1]. Обязательны комментарии, CI и юнит-тесты.

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

***Критерии оценивания*:** решение задачи оценивается от 0 до 10 баллов, где 0 — полное отсутствие решения, 10 — правильное решение, демонстрирующее все ожидаемые навыки (комментарии, тесты, навык использования системы контроля версий, модульного тестирования и непрерывной интеграции).

**Семестр 3.**

***Домашняя работа 1, Lazy:***

Реализовать следующий интерфейс, представляющий ленивое вычисление:

public interface ILazy<T> {

T Get();

}

Объект Lazy создаётся на основе вычисления (представляемого объектом Func<T>):

* Первый вызов Get() вызывает вычисление и возвращает результат
* Повторные вызовы Get() возвращают тот же объект, что и первый вызов
* Вычисление должно запускаться не более одного раза

Создавать объекты надо не вручную, а с помощью класса LazyFactory, который должен иметь два метода с сигнатурами наподобие

public static Lazy<T> Create...Lazy<T>(Func<T> supplier)

возвращающих две разные реализации ILazy<T>:

* Простая версия с гарантией корректной работы в однопоточном режиме (без синхронизации)
* Гарантия корректной работы в многопоточном режиме
  + При этом она должна по возможности минимизировать число необходимых синхронизаций (если значение уже вычислено, не должно быть блокировок)
* supplier вправе вернуть null
* Библиотечным Lazy пользоваться, естественно, нельзя

Нужно:

* CI, на котором проходят ваши тесты
* Тесты
  + Однопоточные, на разные хорошие и плохие случаи
  + Многопоточные, на наличие гонок

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

***Критерии оценивания*:** решение задачи оценивается по шкале от 0 (нет решения или решение имеет существенные недостатки) до 5 (решение работоспособно, аккуратно реализовано, отсутствуют ошибки многопоточного программирования, решение оптимально использует память и избегает лишних блокировок).

Данная работа имеет ожидаемый срок выполнения 3 недели.

***Домашняя работа 2, MyThreadPool:***

Реализовать простой пул задач (наподобие https://docs.microsoft.com/en-us/dotnet/api/system.threading.threadpool?view=netframework-4.8 + https://docs.microsoft.com/en-us/dotnet/api/system.threading.tasks.taskfactory?view=netframework-4.8) с фиксированным числом потоков (число задается в конструкторе)

* При создании объекта MyThreadPool в нем должно начать работу n потоков
* У каждого потока есть два состояния: ожидание задачи / выполнение задачи
* Задача — вычисление некоторого значения, описывается в виде Func<TResult>
* При добавлении задачи, если в пуле есть ожидающий поток, то он должен приступить к ее исполнению. Иначе задача будет ожидать исполнения, пока не освободится какой-нибудь поток
* Задачи, принятые к исполнению, представлены в виде объектов интерфейса IMyTask<TResult>
* Метод Shutdown должен завершить работу потоков. Завершение работы коллаборативное, с использованием CancellationToken — уже запущенные задачи не прерываются, но новые задачи не принимаются на исполнение потоками из пула.
  + Возможны два варианта решения — дать всем задачам, которые уже попали в очередь, досчитаться, либо выбросить исключение во все ожидающие завершения задачи потоки. Не должно получиться так, что вызвавший Result поток может при каких-то условиях заблокироваться навсегда.
* IMyTask
  + Свойство IsCompleted возвращает true, если задача выполнена
  + Свойство Result возвращает результат выполнения задачи
    - В случае, если соответствующая задаче функция завершилась с исключением, этот метод должен завершиться с исключением AggregateException, содержащим внутри себя исключение, вызвавшее проблему
    - Если результат еще не вычислен, метод ожидает его и возвращает полученное значение, блокируя вызвавший его поток
  + Метод ContinueWith — принимает объект типа Func<TResult, TNewResult>, который может быть применен к результату данной задачи X и возвращает новую задачу Y, принятую к исполнению
    - Новая задача будет исполнена не ранее, чем завершится исходная
    - В качестве аргумента объекту Func будет передан результат исходной задачи, и все Y должны исполняться на общих основаниях (т.е. должны разделяться между потоками пула)
    - Метод ContinueWith может быть вызван несколько раз
    - Метод ContinueWith не должен блокировать работу потока, если результат задачи X ещё не вычислен
    - ContinueWith должен быть согласован с Shutdown — принятая как ContinueWith задача должна либо досчитаться, либо бросить исключение ожидающему её потоку.

При этом:

* В данной работе запрещено использование TPL, PLINQ и библиотечных классов Task и ThreadPool.
* Все интерфейсные методы должны быть потокобезопасны (то есть сам пул потоков может разделяться между несколькими потоками, и на это должны быть тесты).
* Для каждого базового сценария использования должен быть написан несложный тест.
* Также должен быть написан тест, проверяющий, что в пуле действительно не менее n потоков.

Подсказка: задачи могут быть разных типов (например, можно var myTask = myThreadPool.Submit(() => 2 \* 2).ContinueWith(x => x.ToString());). Хранить такие задачи в очереди можно, обернув их в Action.

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

***Критерии оценивания*:** решение задачи оценивается по шкале от 0 (нет решения или решение имеет существенные недостатки) до 15 (решение работоспособно, аккуратно реализовано, отсутствуют ошибки многопоточного программирования).

Данная работа имеет ожидаемый срок выполнения 4 недели.

***Домашняя работа 3, SimpleFTP:***

Требуется реализовать сервер, обрабатывающий два запроса.

* List — листинг файлов в директории на сервере
* Get — скачивание файла с сервера

И клиент, позволяющий исполнять указанные запросы.

List, формат запроса:

* <1: Int> <path: String>
* path — путь к директории относительно того места, где запущен сервер

Например, "1 ./Test/Files".

Формат ответа:

* <size: Int> (<name: String> <isDir: Boolean>)\*,
* size — количество файлов и папок в директории
* name — название файла или папки
* isDir — флаг, принимающий значение True для директорий

Например, "2 ./Test/files/file1.txt false ./Test/files/directory true"

Если директории не существует, сервер посылает ответ с size = -1

Get, формат запроса:

* <2: Int> <path: String>
* path — путь к файлу

Формат ответа:

* <size: Long> <content: Bytes>,
* size — размер файла,
* content — его содержимое

Если файла не существует, сервер посылает ответ с size = -1

Сервер должен иметь возможность обслуживать несколько клиентов одновременно (например, в ситуации, когда три клиента одновременно скачивают большой файл).

Обратите внимание на строгое следование протоколу. Сервер любого правильного решения должен уметь взаимодействовать с клиентом любого другого правильного решения.

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

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

Данная работа имеет ожидаемый срок выполнения 2 недели.

***Домашняя работа 4, MyNUnit:***

Реализовать command-line приложение, принимающее на вход путь и выполняющее запуск тестов, находящихся во всех сборках, расположенных по этому пути:

* тестом считается метод, помеченный аннотацией Test
  + у аннотации может быть два аргумента — Expected для исключения, Ignore (со строковым параметром) — для отмены запуска и указания причины
* перед и после запуска каждого теста в классе должны запускаться методы, помеченные аннотациями Before и After
* перед и после запуска тестов в классе должны запускаться методы, помеченные аннотациями BeforeClass и AfterClass
* BeforeClass и AfterClass должны быть статическими методами, при их запуске объект создаваться не должен (поскольку они могут устанавливать окружение, необходимое для выполнения конструктора объекта)

Тесты должны запускаться возможно более параллельно.

Приложение должно выводить в стандартный поток вывода отчет:

* о результате и времени выполнения прошедших и упавших тестов
* о причине отключенных тестов

Юнит-тесты на систему тестирования обязательны (при этом они должны быть написаны не на ней самой, а на чём-то более отлаженном, типа NUnit).

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

***Критерии оценивания*:** решение задачи оценивается по шкале от 0 (нет решения или решение имеет существенные недостатки) до 10 (решение работоспособно, аккуратно реализовано, выдаёт подробную диагностику в случае ошибки написания юнит-теста, работает многопоточно).

Данная работа имеет ожидаемый срок выполнения 3 недели.

***Домашняя работа 5, GUI для FTP:***

Сделать на WPF GUI для FTP-клиента из домашней работы 3.

Нужно:

* иметь возможность задать адрес и порт сервера;
* при подключении получить список файлов и подпапок папки, на которую "смотрит" сервер;
* иметь возможность перемещаться по папкам (переходить в подпапки и возвращаться на уровень выше, если он есть — выходить выше "корневой" папки нельзя);
* иметь возможность указать папку в файловой системе клиента для скачивания файлов;
* иметь возможность скачать один файл или все файлы в текущей папке сразу;
  + при этом скачивание нескольких файлов должно происходить параллельно, в клиенте должен как-нибудь отображаться статус файла — скачивается (и на сколько процентов скачался) или уже скачан.

При этом:

* надо активно пользоваться Data Binding и паттерном Model-View-ViewModel;
* юнит-тесты на GUI можно не писать, но вся нетривиальная функциональность «бэкенда» должна быть протестирована.

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

***Критерии оценивания*:** решение задачи оценивается по шкале от 0 (нет решения или решение имеет существенные недостатки) до 10 (решение работоспособно, аккуратно реализовано, соответствует требованиям на качественные пользовательские интерфейсы, отвечает хорошим практикам использования библиотеки WPF).

Данная работа имеет ожидаемый срок выполнения 3 недели.

***Домашняя работа 6, MyNUnitWeb:***

Реализовать веб-интерфейс для системы юнит-тестирования MyNUnit из 4-й домашней работы. Требуется:

* Форма для загрузки на сервер сборок, которые надо тестировать
  + Это может быть несколько файлов, например, .dll-ка с тестируемыми классами и .dll-ка с юнит-тестами к ним, их можно загружать по одному
* Кнопка «Начать тестирование», запускающая юнит-тесты, по завершении которых должен отобразиться результат тестового прогона
  + +2 балла контрольным за отображение результатов без перезагрузки страницы
* Форма истории запусков, где можно просмотреть результаты всех тестовых прогонов, когда-либо запускавшихся на сервере:
  + Список всех сборок с тестами, общее количество успешных, проваленных и проигнорированных тестов по каждой сборке
  + Список всех тестов в сборке (возможно, появляющийся при выборе сборки из первого списка), со статусом теста и временем его выполнения
    - Если тест проигнорирован, с сообщением о причине

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

***Критерии оценивания*:** решение задачи оценивается по шкале от 0 (нет решения или решение имеет существенные недостатки) до 10 (решение работоспособно, аккуратно реализовано, аккуратно свёрстано, соответствует хорошим практикам разработки веб-приложений).

Данная работа имеет ожидаемый срок выполнения 3 недели.

***Примеры условий контрольных работ:***

***Контрольная работа***

Реализовать потокобезопасную блокирующую очередь с приоритетами. Очередь должна иметь следующие методы:

* Enqueue(value, priority) — ставит значение с заданным приоритетом в очередь.
* Dequeue() — возвращает значение с максимальным приоритетом и удаляет его из очереди. Если очередь пуста, вызвавший поток блокируется до тех пор, пока в очереди не появится значение.
* Size() — возвращает размер очереди в некоторый момент времени в прошлом.

Значения с одинаковыми приоритетами должны извлекаться из очереди в порядке «первый пришёл – первый ушёл».

Могут быть полезны методы Monitor.Wait и Monitor.PulseAll

***Зачётная работа***

Реализовать на WPF игру крестики-нолики (в классическом варианте, с полем 3 на 3), позволяющую играть двум игрокам в режиме hot seat. После победы одного из игроков или ничьи должно быть можно начать новую партию. Нужны юнит-тесты для всего, что не относится напрямую к GUI.

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

***Критерии оценивания*:** решение задачи оценивается от 0 до 10 баллов, где 0 — полное отсутствие решения, 10 — правильное решение, демонстрирующее все ожидаемые навыки (комментарии, тесты, навык использования системы контроля версий, модульного тестирования и непрерывной интеграции).

*Условия, состав и количество домашних работ, а также шкалы оценивания могут меняться в течение семестра в зависимости от успехов обучающихся или в ходе текущей актуализации программы курса, с последующим обновлением данной РПД (это необходимо из-за динамичного развития технологий, на примере которых преподаются концепции в этом курсе).*

**Вариант реализации 2: Прикладное и исследовательское программирование на примере F#.**

**Семестр 1.**

***Домашняя работа 1***

1. (1 балл) Инициализировать рабочее окружение: репозиторий на GitHub, CI, readme, лицензия. Добавить преподавателя в совладельцы. Оформить тестовый pull request: например, оформленное readme. Запросить ревью у преподавателя.

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

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

***Домашняя работа 2***

1. (1 балл) Реализовать функцию, вычисляющую значение выражения x^4+x^3+x^2+x+1 "наивным" способом. Реализовать чтение x из консоли и вывод результата в консоль.
2. (1 балл) Реализовать функцию, вычисляющую значение выражения x^4+x^3+x^2+x+1, применив минимальное число умножений и сложений. Реализовать чтение x из консоли и вывод результата в консоль.
3. (1 балл) Вычислить индексы элементов массива, не больших, чем заданное число. Реализовать чтение входных данных из консоли и печать результата в консоль.
4. (1 балл) Вычислить индексы элементов массива, лежащих вне диапазона, заданного двумя числами. Реализовать чтение входных данных из консоли и печать результата в консоль.
5. (1 балл) Дан массив длины 2. Поменять местами нулевой и первый элементы, не используя дополнительной памяти/переменных. Реализовать чтение входных данных из консоли и печать результата в консоль.
6. (1 балл) Поменять местами i-й и j-й элементы массива не используя дополнительной памяти/переменных. Реализовать чтение входных данных из консоли и печать результата в консоль.

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

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

***Домашняя работа 3***

1. (1 балл) Реализовать вычисление n-ого числа Фибоначчи рекурсивным методом. Обеспечить чтение n из консоли и печать результата в консоль.
2. (1 балл) Реализовать вычисление n-ого числа Фибоначчи итеративным методом. Обеспечить чтение n из консоли и печать результата в консоль.
3. (1 балл) Реализовать вычисление n-ого числа Фибоначчи итеративным методом, но не используя ref-переменных (и mutable) и других изменяемых структур. Подсказка: нужно использовать рекурсию с аккумулятором. Обеспечить чтение n из консоли и печать результата в консоль.
4. (2 балла) Реализовать вычисление n-ого числа Фибоначчи через перемножение матриц «наивным» методом. Обеспечить чтение n из консоли и печать результата в консоль.
5. (2 балла) Реализовать вычисление n-ого числа Фибоначчи через перемножение матриц за логарифм. Обеспечить чтение n из консоли и печать результата в консоль.
6. (1 балл) Реализовать вычисление всех чисел Фибоначчи до n-ого включительно. Обеспечить чтение n из консоли и печать результата в консоль.

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

***Критерии оценивания*:** решения задач 1, 2, 3, 6 оцениваются по шкале от 0 (нет решения или решение имеет существенные недостатки) до 1 (решение работоспособно, аккуратно реализовано). Решения задач 4, 5 оцениваются по аналогичным критериям по шкале от 0 до 2.

***Домашняя работа 4***

1. (1 балл) Реализовать сортировку пузырьком массива. Реализовать чтение массива из файла и печать результата в файл. Функции чтения и записи необходимо переиспользовать.
2. (1 балл) Реализовать сортировку пузырьком списка. Реализовать чтение списка из файла и печать результата в файл. Функции чтения и записи необходимо переиспользовать.
3. (1 балл) Реализовать быструю сортировку для списка. Реализовать чтение списка из файла и печать результата в файл. Функции чтения и записи необходимо переиспользовать.
4. (1 балл) Реализовать быструю сортировку для массива. Реализовать чтение массива из файла и печать результата в файл. Функции чтения и записи необходимо переиспользовать.
5. (1 балл) Реализовать запаковку двух 32-битных чисел в одно 64-битное и распаковку обратно. Реализовать чтение входных данных с консоли и печать результата в консоль.
6. (1 балл) Реализовать запаковку четырёх 16-битных чисел в одно 64-битное и распаковку обратно. Реализовать чтение входных данных с консоли и печать результата в консоль.

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

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

***Домашняя работа 5***

1. (5 баллов) Провести сравнительное исследование реализованных в предыдущей домашней работе сортировок и стандартных реализаций сортировок соответствующих коллекций. Оформить отчёт: постановка эксперимента, результаты экспериментов, анализ результатов. Отчёт оформляется в TeX, исходники выкладываются так же как и обычный код, снабжаются скриптом сборки.

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

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

***Домашняя работа 6***

1. (1 балл) Предположим, что мы храним булевы матрицы в виде списка координат ячеек, значение которых «true». Необходимо реализовать соответствующие типы: единицы измерения для строк и столбцов, пара «строка-столбец», список пар «строка-столбец».
2. (2 балла) Реализовать функцию, перемножающую две матрицы, заданных в формате, описанном в предыдущей задаче. Не забыть проверку корректности входных данных. Реализовать подгрузку матриц из файла и запись результата в файл. Файлы с данными и результатом указываются через консоль. Формат хранения матрицы из m строк и n столбцов: в файле m строк, каждая строка состоит из n символов 0 или 1.

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

***Критерии оценивания*:** решения задачи 1 оцениваются по шкале от 0 (нет решения или решение имеет существенные недостатки) до 1 (решение работоспособно, аккуратно реализовано), решения задачи 2 оцениваются по тем же критериям, но по шкале от 0 до 2.

***Домашняя работа 7***

1. (1 балл) Реализовать самостоятельно полиморфный список (далее будем называть этот тип MyList). Реализовать для него функции сортировки, вычисления длины, конкатенации.
2. (1 балл) На основе MyList реализовать тип MyString, представляющий строку как список символов. Реализовать преобразование стандартной строки в MyString и конкатенацию строк для MyString.
3. (1 балл) Реализовать тип дерева с произвольным количеством потомков в каждом узле (использовать MyList) MyTree. Каждый узел должен хранить данные произвольного типа. Реализовать функцию обхода в глубину для такого дерева.
4. (1 балл) Пусть есть MyTree, хранящий в узлах целые числа. Реализовать функции, которые находят максимальный хранимый элемент, среднее значение всех хранимых элементов.

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

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

***Домашняя работа 8***

1. (4 балла) Используя тип MyTree из предыдущей домашней работы, реализовать целочисленную длинную арифметику: операции сложения, умножения, вычитания, целочисленного деления.

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

***Критерии оценивания*:** решения задачи оцениваются по шкале от 0 (нет решения или решение имеет существенные недостатки) до 4 (решение работоспособно, аккуратно реализовано).

***Домашняя работа 9***

1. (1 балл) Реализовать представление разреженных матриц в виде дерева квадрантов. Реализовать функцию поэлементного сложения двух матриц в таком формате.
2. (1 балл) Реализовать функцию умножения двух матриц в формате дерева квадрантов.
3. (1 балл) Реализовать функцию тензорного умножения двух матриц в формате дерева квадрантов.
4. (3 балла) Реализовать построение транзитивного замыкания ориентированного графа через произведение матриц. Использовать представление матриц из первой задачи. Визуализировать результат с помощью GraphViz: исходный граф и выделенные рёбра, появившиеся в результате транзитивного замыкания. Для задания графа использовать формат из задачи 2 6-й домашней работы.
5. (3 балла) Реализовать вычисление кратчайших путей между всеми парами вершин в ориентированном графе. Использовать представление матриц из первой задачи. Визуализировать результат с помощью GraphViz: исходный граф и выделенные рёбра со значением кратчайшего пути между соответствующей парой вершин. Для задания графа использовать формат, аналогичный формату из задачи 2 6-й домашней работы.

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

***Критерии оценивания*:** решения задач 1-3 оцениваются по шкале от 0 (нет решения или решение имеет существенные недостатки) до 1 (решение работоспособно, аккуратно реализовано), решения задач 4-5 оцениваются по тем же критериям, но по шкале от 0 до 3.

***Домашняя работа 10***

1. (3 балл) Реализовать с помощью одного из инструментов ANTLR, FParsec, fslex+fsyacc синтаксический анализ языка регулярных выражений и операций над ними. Регулярные выражения используют алфавит латинских символов и операции конкатенации, |, \*, группирующие скобки. Можно добавить + и ?. Операции над выражениями – пересечение и объединение. Обеспечить чтение входа из файла и визуализацию дерева разбора в GraphViz.
2. (3 балл) Реализовать построение автомата по регулярному выражению. В качестве основы использовать представление матриц из предыдущей работы.
3. (5 баллов) Реализовать вычислитель для длинной арифметики. Реализовать синтаксический анализатор арифметических выражений: бинарные операции +, -, \*, /; унарный минус; возведение в степень; группирующие скобки; возможность именовать подвыражения и использовать имена в других выражениях. Числа только целые и могут быть «очень большими». Обеспечить чтение входа из файла и визуализацию дерева разбора в GraphViz. Реализовать вычисление значения выражения на основе реализованной ранее длинной арифметики.

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

***Критерии оценивания*:** решения задач 1-2 оцениваются по шкале от 0 (нет решения или решение имеет существенные недостатки) до 3 (решение работоспособно, аккуратно реализовано), решения задачи 3 оцениваются по тем же критериям, но по шкале от 0 до 5.

***Домашняя работа 11***

1. (1 балл) Расширить язык из предыдущей работы командой, которая будет принимать выражение над регулярками (автомат) и строку и проверять, принимается ли строка автоматом.
2. (2 балл) Расширить язык возможностью именовать подвыражения и использовать имена в других выражениях.
3. (5 баллов) Реализовать интерпретатор получившегося языка. Для построения автоматов по регулярным выражениям использовать результаты предыдущей работы. Для операций над автоматами использовать тензорное произведение из работы 9. Вероятно, нужно будет определить абстракцию полукольца. Снабдить интерпретатор возможностями обрабатывать файл и работать в интерактивном режиме.

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

***Критерии оценивания*:** решения задачи 1 оцениваются по шкале от 0 (нет решения или решение имеет существенные недостатки) до 1 (решение работоспособно, аккуратно реализовано), решения задачи 2 оцениваются по тем же критериям, но по шкале от 0 до 2, решения задачи 3 оцениваются по тем же критериям, но по шкале от 0 до 5.

**Семестр 2.**

***Домашняя работа 1***

1. (1 балл) Оформление калькулятора или регулярных выражений как отдельного проекта. Создать репозиторий, снабдить всеми необходимыми элементами экосистемы: сборка, тесты, лицензия, readme.
2. (2 балла) Создать документацию. Описать цели и задачи проекта, конкретный синтаксис языка, привести примеры.
3. (2 балла) Создать диаграмму (наиболее подходящего типа), описывающую структуру проекта, выбранного выше. Добавить её в документацию.

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

***Критерии оценивания*:** решения задачи 1 оцениваются по шкале от 0 (нет решения или решение имеет существенные недостатки) до 1 (решение работоспособно, аккуратно реализовано), решения задач 2 и 3 оцениваются по тем же критериям, но по шкале от 0 до 2.

***Домашняя работа 2.*** Мини-IDE для языка из предыдущей работы.

1. (2 балла) Расширить синтаксис соответствующего языка логическими выражениями (с переменными) и условными операторами.
2. (5 баллов) Разработать среду разработки для полученного языка. Среда должна предоставлять следующие возможности:
   1. Редактировать код.
   2. Работать с файлами: создать новый, открыть существующий, сохранить изменения.
   3. Выводить сообщения о (синтаксических) ошибках.
   4. Запустить программу на исполнение.
   5. Видеть результат исполнения в «консоли».
3. (3 балла) Расширить IDE возможностью подсветки синтаксиса.
4. (5 баллов) Расширить IDE возможностью устанавливать точки останова. В момент остановки должна быть возможность просмотреть значения всех «живых» переменных.

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

***Критерии оценивания*:** решения задачи 1 оцениваются по шкале от 0 (нет решения или решение имеет существенные недостатки) до 2 (решение работоспособно, аккуратно реализовано), решения задач 2 и 4 оцениваются по тем же критериям, но по шкале от 0 до 5, решения задачи 3 оцениваются по тем же критериям, но по шкале от 0 до 3.

***Домашняя работа 3***

1. (1 балл) Реализовать список с конкатенацией за константу. Реализовать функцию конкатенации для соответствующего типа.
2. (2 балла) Реализовать функции вычисления длины, взятия головы и хвоста, добавления элемента в голову для списка из предыдущего пункта.
3. (2 балла) Реализовать сортировку для списка из первого пункта.
4. (3 балла) Реализовать map, foldr, foldl, rev для списка из первого пункта.
5. (3 балла) Реализовать zipper для стандартного списка.
6. (4 балла) Реализовать zipper для списка из первого пункта.

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

***Критерии оценивания*:** решения задачи 1 оцениваются по шкале от 0 (нет решения или решение имеет существенные недостатки) до 1 (решение работоспособно, аккуратно реализовано), решения задач 2 и 3 оцениваются по тем же критериям, но по шкале от 0 до 2, решения задач 4 и 5 оцениваются по тем же критериям, но по шкале от 0 до 3, решение задачи 6 оценивается по тем же критериям, но по шкале от 0 до 4.

***Домашняя работа 4***

1. Реализовать консольный генератор матриц. На вход принимается размер матрицы, тип данных, количество матриц, метрика разреженности, возможно другие необходимые параметры. В результате генерируется набор файлов с матрицами в формате из первого семестра.
2. Реализовать параллельное умножение для плотных матриц. Исследовать варианты с распараллеливанием различных циклов. Для исследования использовать созданный ранее генератор. Оформить соответствующий отчёт.
3. Реализовать параллельное умножение матриц, представленных в виде дерева квадрантов.
4. Сравнить производительность решений из первых двух пунктов на разных типах матриц. Для исследования использовать созданный ранее генератор. Оформить соответствующий отчёт.

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

***Критерии оценивания*:** решения всех задач оцениваются по шкале от 0 (нет решения или решение имеет существенные недостатки) до 4 (решение работоспособно, аккуратно реализовано).

***Домашняя работа 5***

1. На основе Mailboxprocessor или Hopac реализовать решение, в котором есть следующие конкурентно выполняющиеся типы задачи:
   1. Подгрузка пар матриц из файлов (для генерации использовать генератор из предыдущей работы)
   2. Различные алгоритмы перемножения матриц (для разреженных, для плотных параллельно и последовательно)
   3. Балансировщик, знающий, кому какие матрицы отправлять для обработки.

Предусмотреть два режима работы:

* + 1. Обработать все матрицы, доступные на входе
    2. Обработать заданное количество пар матриц

1. Проанализировать масшатабируемость полученной системы. Какое количество конкурентных задач оптимально для определённой конфигурации системы? Оформить соответствующий отчёт.

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

***Критерии оценивания*:** решения всех задач оцениваются по шкале от 0 (нет решения или решение имеет существенные недостатки) до 4 (решение работоспособно, аккуратно реализовано).

**Семестр 3.**

***Домашняя работа 1***

1. (5 баллов) Сформулировать гипотезы относительно зависимости времени работы алгоритмов сортировок относительно размера входных данных. Проверить их. Обосновать полученные результаты. Оформить соответствующий отчёт. Должны быть проанализированы следующие алгоритмы.
   1. Стандартная сортировка для List
   2. Различные варианты реализации быстрой сортировки для MyList.
   3. Сортировка пузырьком для List.
   4. Сортировка пузырьком для Array.
   5. Можно включить сортировки для списка с конкатенацией за константу, а также другие реализации быстрой сортировки (на массиве, например).
2. (7 баллов) Сформулировать гипотезы относительно зависимости времени работы матричных алгоритмов относительно размера входных матриц. Проверить их. Обосновать полученные результаты. Оформить соответствующий отчёт. Должны быть проанализированы следующие алгоритмы.
   1. Последовательный для дерева квадрантов
   2. Параллельный для дерева квадрантов
   3. Последовательный для плотных матриц
   4. Параллельный для плотных матриц

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

***Критерии оценивания*:** решения задачи 1 оцениваются по шкале от 0 (нет решения или решение имеет существенные недостатки) до 5 (решение работоспособно, аккуратно реализовано), решения задачи 2 оцениваются по тем же критериям, но по шкале от 0 до 7.

***Домашняя работа 2***

1. (6 баллов) Подготовить презентацию «Анализ времени работы алгоритмов сортировок» или «Анализ времени работы алгоритмов умножения матриц». Презентация готовится в TeX, по установленному шаблону.
2. (8 баллов) Подготовить отчёт на тему «Анализ времени работы алгоритмов сортировок» или «Анализ времени работы алгоритмов умножения матриц». Отчёт готовится в TeX, по установленному шаблону.

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

***Критерии оценивания*:** решения задачи 1 оцениваются по шкале от 0 (нет презентации или презентация имеет существенные недостатки) до 6 (презентация полностью раскрывает тему и аккуратно оформлена), решения задачи 2 оцениваются по тем же критериям, но по шкале от 0 до 8.

***Домашняя работа 3***

1. (3 балла) Реализовать «логгер» с использованием workflow builder. Необходимо предоставить возможность логгировать входные аргументы некоторых функций. Предусмотреть возможность указывать, куда печатать вывод: в консоль, файл, куда-то ещё.
2. (5 баллов) Реализовать list builder для типа MyList или списка с конкатенацией за константу. Ориентироваться на билдер seq для лучшего понимания ожидаемой функциональности.

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

***Критерии оценивания*:** решения задачи 1 оцениваются по шкале от 0 (нет решения или решение имеет существенные недостатки) до 3 (решение работоспособно, аккуратно реализовано), решения задачи 2 оцениваются по тем же критериям, но по шкале от 0 до 5.

***Домашняя работа 4***

1. (10 баллов) Встроить на основе F# quotations разработанный ранее язык арифметики или регулярных выражений. Обеспечить вычисление задаваемых выражений и возможность работы с полученными значениями на стороне F#. Необходимо максимальное переиспользование готовых компонент.
2. (4 баллов) Внедрить RxExtensions для работы с событиями в разработанной ранее IDE.

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

***Критерии оценивания*:** решения задачи 1 оцениваются по шкале от 0 (нет решения или решение имеет существенные недостатки) до 10 (решение работоспособно, аккуратно реализовано), решения задачи 2 оцениваются по тем же критериям, но по шкале от 0 до 4.

***Домашняя работа 5***

1. (4 баллов) Реализовать функцию перемножения двух плотных матриц с использованием Brahma.FSharp.
2. (9 баллов) Расширить решение из предыдущего семестра возможностью конкурентно умножать матрицы на GPGPU (использовать предыдущее решение).
3. (9 баллов) Проанализировать полученное в предыдущем пункте решение и оформить соответствующий отчёт.
   1. В каких случаях лучше использовать CPU, а в каких GPGPU?
   2. Какая конфигурация конкурентно выполняющихся задач оптимальна? Имеет ли смысл поддерживать больше одного агента, работающего с GPGPU?
   3. Имеет ли смысл использовать Brahma.FSharp для выполнения кода на CPU?

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

***Критерии оценивания*:** решения задачи 1 оцениваются по шкале от 0 (нет решения или решение имеет существенные недостатки) до 4 (решение работоспособно, аккуратно реализовано), решения задач 2 и 3 оцениваются по тем же критериям, но по шкале от 0 до 9.

**Вариант реализации 3: …**

**Вариант реализации 4: …**

**Вариант реализации 5: …**

**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

Тут надо вписать себя