**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

Programming

**Язык(и) обучения**

русский

Трудоемкость в зачетных единицах: 11

Регистрационный номер рабочей программы: 002212

Санкт-Петербург

2024

**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 к 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)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Наименование категории (группы) компетенций | Код и наименование компетенции | Планируемые результаты обучения, обеспечивающие формирование компетенции | Код индикатора и индикатор достижения универсальной компетенции |
|  | 1 | 2 | 3 | 4 |
| 1 | Теоретические и практические основы профессиональной деятельности | ОПК-1 — способен применять фундаментальные знания, полученные в области математических и (или) естественных наук, и использовать их в профессиональной деятельности. | Понимает и может применять формализации понятия «Вычисление». | ОПК-1.002212.1. Пишет программы императивном и функциональном стиле. |
| 2 | Теоретические и практические основы профессиональной деятельности | ОПК-2. Способен применять современный математический аппарат, связанный с проектированием, разработкой, реализацией и оценкой качества программных комплексов в различных областях человеческой деятельности. | Владеет методами оценки асимптотической трудоёмкости алгоритмов  Владеет основами теории графов и дискретной математики в объёмах, достаточных для реализации классических алгоритмов и структур данных. | ОПК-2.002212.1. Оценивает асимптотическую трудоёмкость алгоритма.  ОПК-2.002212.2. Применяет основы теории графов и дискретной математики для реализации классических алгоритмов и структур данных. |
| 3 | Информационно-коммун икационные технологии для профессиональной деятельности | ОПК-3 — способен применять современные информационные технологии, в том числе отечественные, при создании программных продуктов и программных комплексов различного назначения. | Владеет инструментами разработки. | ОПК-3.002212.1. Ведёт разработку программного обеспечения с использованием интегрированных сред разработки, систем контроля версий, отладчиков. |
| 4 | Информационно-коммун икационные технологии для профессиональной деятельности | ОПК-4. Способен участвовать в разработке технической документации программных продуктов и программных комплексов. | Понимает важность технического документирования программного кода.  Владеет русскоязычной и англоязычной терминологией программирования. | ОПК-4.002212.1. Пишет комментарии к программному коду.  ОПК-4.002212.2. Правильно использует терминологию. |
| 5 | Информационно-коммун икационные технологии для профессиональной деятельности | ОПК-5. Способен инсталлировать и сопровождать программное обеспечение для информационных систем и баз данных, в том числе отечественного производства. | Умеет развернуть рабочее окружение.  Понимает важность качественного инструментария и автоматизации при разработке программного обеспечения. | ОПК-5.002212.1. Может развернуть рабочее окружение: среду разработки, клиент системы контроля версий, вспомогательные инструменты. |
| 6 | Профессиональные компетенции (академические) | ПКА-1 — способен демонстрировать базовые знания математических и естественных наук, программирования и информационных технологий. | Владеет навыками разработки программ, в том числе для решения задач математики. | ПКА-1.002212.1. Разрабатывает программы для работы с математическими структурами. |
| 7 | Профессиональные компетенции (практические) | ПКП-3. Способен преподавать математику и информатику в средней школе, специальных учебных заведениях на основе полученного фундаментального образования и научного мировоззрения. | Имеет навыки подготовки учебных материалов в области программирования, в том числе по ранее незнакомому материалу. | ПКП-3.002212.1. Самостоятельно находит, изучает и представляет коллегам в удобной для восприятия форме информацию об алгоритмах, структурах данных или технологиях. |
| 8 | Профессиональные компетенции (практические) | ПКП-4. Способен применять современные информационные технологии при проектировании, реализации, оценке качества и анализа эффективности программного обеспечения для решения задач в различных предметных областях. | Владеет интегрированными средами разработки, понимает важность их использования для повышения продуктивности.  Владеет командной строкой и консольными утилитами, необходимыми для разработки.  Владеет системами контроля версий, понимает важность версионирования при разработке. | ПКП-4.002212.1. Пользуется интегрированной средой разработки для реализации программного обеспечения.  ПКП-4.002212.2. Способен собрать программу из командной строки без использования интегрированной среды разработки.  ПКП-4.002212.3. Пользуется системой контроля версий при разработке программного обеспечения. |
| 9 | Профессиональные компетенции (практические) | ПКП-5 — способен использовать основные методы и средства автоматизации проектирования, реализации, испытаний и оценки качества при создании конкурентоспособного программного продукта и программных комплексов, а также способен использовать методы и средства автоматизации, связанные с сопровождением, администрированием и модернизацией программных продуктов и программных комплексов. | Владеет системами непрерывной интеграции. | ПКП-5.002212.1. Настраивает системы непрерывной интеграции для кода своих проектов. |
| 10 | Профессиональные компетенции (практические) | ПКП-6 — способен использовать знания направлений развития компьютеров с традиционной (нетрадиционной) архитектурой; современных системных программных средств: операционных систем, операционных и сетевых оболочек, сервисных программ; тенденции развития функций и архитектур проблемно-ориентированных программных систем и комплексов в профессиональной деятельности. | Понимает архитектуру современных процессоров. | ПКП-6.002212.1. Пишет корректные многопоточные программы с учётом модели памяти процессора. |
| 11 | Профессиональные компетенции (практические) | ПКП-7 — способен использовать основные концептуальные положения функционального, логического, объектно-ориентированного и визуального направлений программирования, методы, способы и средства разработки программ в рамках этих направлений. | Владеет парадигмами структурного, объектно-ориентированного и функционального программирования.  Владеет основами языка UML. | ПКП-7.002212.1. Пишет программы в структурном стиле.  ПКП-7.002212.2. Пишет программы в объектно-ориентированном стиле.  ПКП-7.002212.3. Пишет программы в функциональном стиле.  ПКП-7.002212.4. Способен описать на UML структуру классов несложной системы. |
| 12 | Профессиональные компетенции (практические) | ПКП-8 — способен использовать современные методы разработки и реализации конкретных алгоритмов математических моделей на базе языков программирования и пакетов прикладных программ моделирования. | Владеет навыками разработки и реализации основных математических алгоритмов. | ПКП-8.002212.1. Способен реализовать на каком-либо языке программирования математический алгоритм по приведённому описанию алгоритма. |
| 13 | Работа с информацией | 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, искусственного интеллекта и науки о данных, а также информационной безопасности. | Умеет работать с поисковыми сервисами сети «Интернет».  Владеет навыками поиска и анализа технической информации.  Умеет выбирать источники с достоверной информацией, ранжировать источники по степени достоверности. | УКБ-3.1. Находит и использует различные источники информации.  УКБ-3.2. Точно определяет тип и форму необходимой информации.  УКБ-3.3. Получает информацию и сохраняет ее в удобном для работы формате.  УКБ-3.4. Проверяет достоверность собранной информации. |

**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 50 часов в первом семестре, 52 часа во втором семестре, 22 часа в третьем семестре, 26 часов в четвёртом 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 |  |  |  | 56 |  | 6 |  |  | 2 |  |  |  | 75 |  | 5 |  | 50 | 4 |
|  |  |  |  | 2-15 |  | 2-15 |  |  | 1-15 |  |  |  | 1-1 |  | 1-1 |  |  |  |
| Семестр 2 |  |  |  | 56 |  | 4 |  |  | 2 |  |  |  | 38 |  | 8 |  | 52 | 3 |
|  |  |  |  | 2-15 |  | 2-15 |  |  | 1-15 |  |  |  | 1-1 |  | 1-1 |  |  |  |
| Семестр 3 |  |  |  | 26 |  | 4 |  |  | 2 |  |  |  | 37 |  | 3 |  | 22 | 2 |
|  |  |  |  | 2-15 |  | 2-15 |  |  | 1-15 |  |  |  | 1-1 |  | 1-1 |  |  |  |
| Семестр 4 |  |  |  | 26 |  | 4 |  |  | 2 |  |  |  | 32 |  | 8 |  | 26 | 2 |
|  |  |  |  | 2-15 |  | 2-15 |  |  | 1-15 |  |  |  | 1-1 |  | 1-1 |  |  |  |
| ИТОГО |  |  |  | 164 |  | 18 |  |  | 8 |  |  |  | 182 |  | 24 |  |  | 11 |

|  |  |  |  |
| --- | --- | --- | --- |
| 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 |  |
| Семестр 4 |  | зачёт |  |

**2.2. Структура и содержание учебных занятий**

Структура и содержание учебных занятий зависят от варианта реализации дисциплины. Обучающимся предлагается пять вариантов реализации (некоторые из которых могут не реализовываться для конкретного года поступления — список активных вариантов доводится до обучающихся до начала входного тестирования):

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. В 4-м семестре 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. Особенностью выбранного языка является его универсальность: он изначально создавался как язык для промышленной платформы .NET, сочетающий в себе мощную систему типов, объектно-ориентированный и функциональный подход к дизайну программ и их разработки, продвинутые средства метапрограммрования. Таким образом, то, что в другие языки (например C#, Java, C++) сейчас активно добавляется по ходу их развития (например, функции высшего порядка, сопоставление с образцом) и потому часто выглядит по меньшей мере неестественно и сложно осознаваемо, в F# заложено изначально и потому осваивается существенно проще. 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 также 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 исследовательских и базовых прикладных задач, а также достаточных для самостоятельного дальнейшего углублённого изучения специфичных областей программирования и разработки ПО. В 4-м семестре материал курса дополняется углублённым изучением некоторых областей с акцентом на групповую проектную деятельность, а также рассмотрением дополнительных задач, что позволяет после второго курса сформировать у обучающихся широкий кругозор, богатый инструментарий навыков и подготовить его к дальнейшей работе как в исследовательских, так и в прикладных проектах. Данный вариант реализации отличается большим количеством теоретического материала достаточно специального характера, сообщаемого на занятиях, характерен большим количеством домашних заданий, проверяемых индивидуально, и требующих значительных самостоятельных усилий. Несмотря на то, что данный вариант ориентирован на обучающихся без предварительных знаний, требует большого количества усилий, самодисциплины и внутренней мотивации для освоения.
3. **Промышленное программирование, C#, С++** — данный вариант ориентируется на обучающихся без предварительных знаний, готовых много читать и учиться самостоятельно. Отличительной особенностью данного варианта является очная проверка домашних заданий, прямо на занятии, тогда как изучение теории выполняется дома (образовательная технология «перевёрнутый класс»). В ходе обучения предлагается под руководством преподавателя познакомиться с рядом фундаментальных книг по технологиям программирования, научиться решать типичные для индустриального программирования задачи и, с помощью преподавателя, освоить технологии и инструменты, необходимые любому программисту: отладчики, системы сборки, профиляторы, системы контроля версий и т.д. Также уникальной особенностью этого варианта является «ветвистость» образовательной траектории – начинается курс с объектно-ориентированного программирования на C#, далее, в зависимости от желания и возможностей группы, можно углублённо изучать C# или перейти к системному программированию на C++, включая или не включая продвинутую многопоточность. Вариант также отличается от остальных большим количеством своих для каждого обучающегося задач и индивидуальным подходом.
4. **Программирование на C и C++** — очень подробное изучение понятий, лежащих в основе языков программирования C и C++. Не требует предварительных знаний по программированию, но могут использоваться знания по школьному курсу математики в объеме математических школ. Целью данной траектории является развитие у обучающихся способности разрабатывать адекватные алгоритмы для несложных задач программирования, выбирать подходящие для них структуры данных и реализовывать их, применяя языки программирования C и C++. В четвёртом семестре в курсе подробно рассматриваются средства автоматической обработки текстов: конечные автоматы и формальные грамматики.
5. **Прикладное программирование на языке C#** — создание законченных приложений с изучением инструментов языка C#. Кроме настольных предложений преподавателем (в зависимости от успеваемости обучающихся) может быть предложено к реализации веб-приложение на базе Blazor или мобильное на базе Xamarin. Разработанные приложения следуют объектно-ориентированным принципам SOLID и чистой архитектурой, которую можно продемонстрировать в дальнейшем на собеседованиях. Предварительно рассматриваются оптимальные алгоритмы для ряда задач, что также облегчает подготовку к собеседованиям. На занятиях практикуется индивидуальный подход и ревью кода.

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.

Период обучения (модуль): семестр 4.

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

Раздел 1: Функциональное программирование.

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

*Домашняя работа 1.*

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

*Домашняя работа 2.*

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

*Домашняя работа 3.*

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

*Домашняя работа 4.*

Раздел 2: F# как мультипарадигменный язык программирования.

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

*Домашняя работа 5.*

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

*Домашняя работа 6.*

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

*Домашняя работа 7.*

1. Контрольная работа.

Раздел 3: Синтаксический анализ на F#.

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

*Домашняя работа 8.*

Раздел 4: Приложения F#.

1. Доклады.
2. Контрольная работа.

Раздел 5: Особенности платформы .NET.

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

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

Раздел 6: Доклады по учебным практикам

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

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

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

**Вариант реализации 2: 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 и особенности их применения: модульные, интеграционные, unit и т.д. 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, логгирование, 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.

*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*

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*

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*

1. Проверочная работа.

Период обучения (модуль): семестр 4.

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Наименование темы (раздела, части) | Вид учебных занятий | Количество часов |
| I. | Основы цифровой обработки изображений | практические занятия | 4 |
| самостоятельная работа | 8 |
| II. | Структуры данных и алгоритмы линейной алгебры | практические занятия | 10 |
| контрольные работы | 2 |
| самостоятельная работа | 8 |
| III. | Основы анализа сложности алгоритмов | практические занятия | 6 |
| контрольные работы | 2 |
| самостоятельная работа | 6 |
| IV. | Приложения F# | практические занятия | 2 |
| самостоятельная работа | 4 |
| V. | Доклады по учебным практикам | практические занятия | 4 |
| самостоятельная работа | 4 |
| VI. | Промежуточная аттестация | самостоятельная работа | 8 |
| зачёт | 2 |

Раздел 1: Основы цифровой обработки изображений.

1. Введение. Основы растрового представления изображений. Форматы хранения. Кодировка цвета, оттенки серого*.*

*Домашняя работа 1. Приложение для визуализации картинок*

1. Матричные фильтры для обработки изображений: размытие, усреднение, выделение границ (Гаусс, Собель, прочие специфичные фильтры). Их реализация на GPGPU. Свёрточные фильтры, нейронные сети, и обработка изображений.  
    *Домашняя работа 2. Реализация фильтров в приложении из прошлой домашней работы.*
2. Проверочная работа

Раздел 2: Структуры данных и алгоритмы линейной алгебры

1. Основы линейной алгебры: примитивы (матрицы, вектора, поля, кольца, полукольца) и их свойства (конечность и идемпотентность, коммутативность и т.д.), операции над матрицами и векторами: поэлементные, умножение матриц, умножение матрицы на вектор, тензорное произведение, транспонирование.
2. Разреженное представление матриц и векторов. Основные форматы разреженного представления матриц: покоординатный, CSR, Quad-tree. Специализированные форматы: диагональные матрицы, HiCOO, и др. Их преимущества и недостатки.  
    *Домашняя работа 3. Сравнение pygraphblas и scipy.*
3. Параллельная обработка разреженных матриц и векторов. Особенности соответствующих алгоритмов для GPGPU.
4. Прикладные задачи, сводимые к линейной алгебре. Обработка графов, GraphBLAS API. BFS, транзитивное замыкание, кратчайшие пути, подсчёт треугольников, минимальное остовное дерево. Пересечение автоматов, объединение автоматов.  
    *Домашняя работа 4. Разреженные матрицы, вектора, задачи на графе*
5. Контрольная работа.
6. Разбор домашних работ и контрольной работы.

Раздел 3: Основы анализа сложности алгоритмов.

1. Введение. Классическая теория сложности, анализ сложности алгоритмов в теории и что это значит для практики.
2. Основы fine-grained complexity.
3. Основы анализа сложности параллельных алгоритмов.

*Домашняя работа 5. Выбрать любой алгоритм на графах из реализованных и провести его анализ сложности.*

1. Проверочная работа

Раздел 4: Приложения F#.

1. Доклады.
2. Контрольная работа.

Раздел 5: Доклады по учебным практикам

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

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

**Вариант реализации 3: Промышленное программирование, C#, С++**

Период обучения (модуль): семестр 1.

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Наименование темы (раздела, части) | Вид учебных занятий | Количество часов |
| I. | Введение, основные конструкции программирования | практические занятия | 10 |
| лабораторные работы | 6 |
| контрольные работы | 0 |
| самостоятельная работа | 21 |
| II. | Объектно-ориентированное программирование | практические занятия | 10 |
| лабораторные работы | 6 |
| контрольные работы | 2 |
| самостоятельная работа | 22 |
| III. | Дополнительные конструкции C# | практические занятия | 23 |
| лабораторные работы | 5 |
| контрольные работы | 2 |
| самостоятельная работа | 32 |
| IV. | Промежуточная аттестация | самостоятельная работа | 5 |
| зачёт | 2 |

Раздел 1: Введение, основные конструкции программирования

1. Построение проектов с помощью среды разработки Visual Studio.
2. Создание приложений с пользовательским интерфейсом на Windows Forms. Редактор форм, дизайн пользовательских интерфейсов, реакция на события от интерфейса.
3. Метод Main, параметры командной строки, консольный ввод-вывод. Встроенные типы C#, преобразования типов, неявные типы. Строки, массивы и структуры. Ссылочные типы и типы-значения.

Раздел 2: Объектно-ориентированное программирование

* 1. Понятие класса, описание классов в C#. Конструкторы. Поля, свойства, автоматические свойства. Модификаторы доступа. Инкапсуляция, наследование и полиморфизм. Абстрактные и виртуальные методы. Частичные типы. Методы Equals и GetHashCode.
  2. Модульное тестирование. Назначение и преимущества модульного тестирования, основные правила написания модульных тестов. Библиотека Microsoft Unit Test Framework. Класс Assert, основные проверки. Запуск и отладка модульных тестов.
  3. Интерфейсы. Наследование интерфейсов. Реализация интерфейсов, явная реализация. Отличие от абстрактных базовых классов. Стандартные библиотечные интерфейсы: IEnumerable, IEnumerator, ICloneable, IComparable, IComparer. Ключевые слова yield return, yield break. Глубокое и мелкое копирование, метод MemberwiseClone.

Раздел 3: Дополнительные конструкции C#

1. Коллекции и обобщения. Пространства имён System.Collections и System.Collections.Generic. Синтаксис инициализации коллекций. Понятия упаковки и распаковки (boxing/unboxing). Обобщённые методы. Параметры-типы. Ограничения на типы.
2. Версионный контроль. Система контроля версий git, назначение и место в цикле разработки. Основные понятия: коммит, ветка, слияние веток. Работа с удалёнными репозиториями. Хостинг систем контроля версий GitHub: создание репозитория, пуллреквесты.
3. Делегаты, события и лямбда-выражения. Концепция обратного вызова, тип делегата и его реализация в C#, групповой вызов. Базовые классы делегатов. Обобщённые делегаты Action<> и Func<>. Ключевое слово event, регистрация на событие и отписывание от события. Классы System.EventHandler, System.EventArgs. Анонимные методы. Замыкания. Лямбда-выражения.
4. Структурированная обработка исключений. Процесс генерации и обработки исключений, выборочный перехват исключений, повторная генерация исключений, блок finally. Класс System.Exception и его свойства. Исключения, сгенерированные CLR и пользовательские исключения. Определение своих классов-исключений.
5. Жизненный цикл объекта. Классы, объекты и ссылочные переменные. Куча, выделение памяти в CLR. Алгоритм сборки мусора CLR, поколения. Класс System.GC, принудительная сборка мусора, состояние кучи. Метод Finalize, финализаторы, Dispose(). Ленивое создание объектов.

Далее обучающимся предстоит голосованием решить, какую траекторию выбрать для продолжения обучения в следующем семестре: продолжать изучать C# или перейти к программированию на C++.

Период обучения (модуль): семестр 2, вариант 1: продолжение C#

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Наименование темы (раздела, части) | Вид учебных занятий | Количество часов |
| I. | Углублённый C# | практические занятия | 11 |
| лабораторные работы | 15 |
| контрольные работы | 4 |
| самостоятельная работа | 32 |
| II. | Промежуточная аттестация | самостоятельная работа | 8 |
| зачёт | 2 |

Раздел 1: Углублённый C#

1. Углублённая система типов C#. Класс System.Object, приведение типов, операторы as и is. Стек, куча, JIT-компиляция. Ссылочные типы и типы-значения. Пространства имён, using. Ключевые слова checked и unchecked. Ключевые слова abstract, virtual, override, sealed, new. Статические классы. Разница между var и dynamic.
2. Конструкторы ссылочных типов и типов-значений, статические конструкторы. Виртуальные методы и конструктор. Перегрузка операторов, неявное преобразование типа, ключевые слова implicit и explicit. Свойства в C#. Методы-расширения, const-методы, partial-методы. Необязательные и именованные параметры, модификаторы ref и out. Сильные и слабые типы. Кортежные типы.
3. Синтаксис описания обобщённого класса. static-поля в генериках. Обобщённые методы. Открытые и закрытые типы. Ковариантность и контравариантность. Ограничения на параметры-типы: основные, ограничения конструктора. Ключевое слово default. Обобщённые интерфейсы.
4. LINQ. Синтаксический сахар для LINQ в C#. Пространство имён System.Linq, библиотечные методы.
5. Строки, verbatim-строки, интерполяция строк. Конвертирование символа в число, переносы строк. Класс StringBuilder. Культура, региональные стандарты. Unicode. Класс SecureString. Интернирование строк. Перечислимые типы, внутреннее устройство, методы перечислимых типов. Массивы, их реализация, копирование массивов, небезопасный доступ к массиву, многомерные массивы.
6. Делегаты. Ковариантность и контравариантность делегатов. Цепочки делегатов, добавление и удаление делегатов из цепочки. Делегаты и рефлексия. Лямбда-выражения. События. Атрибуты. Nullable-типы: задание, операторы, упаковка и распаковка. Оператор «??».
7. Исключения: семантика try/catch/finally, порядок сопоставления исключений в catch. CLS-(не)совместимые исключения. «throw», особенности перебрасывания исключений. Рефлексия и исключения. Constrained Execution Region.
8. Сборка мусора и финализация в .NET. Метод Dispose(), финализаторы, влияние на время жизни объекта. Большие объекты и сборка мусора. Режимы сборщика мусора.

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

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Наименование темы (раздела, части) | Вид учебных занятий | Количество часов |
| I. | Язык Си | практические занятия | 4 |
| лабораторные работы | 6 |
| контрольные работы | 2 |
| самостоятельная работа | 10 |
| II. | Язык C++ | практические занятия | 7 |
| лабораторные работы | 9 |
| контрольные работы | 2 |
| самостоятельная работа | 22 |
| III. | Промежуточная аттестация | самостоятельная работа | 8 |
| зачёт | 2 |

Раздел 1: Язык Си.

1. Различия Си и C#. Структурное программирование, нативное исполнение кода, низкоуровневая работа с памятью, полный контроль над работой процессора.
2. Типы, операторы и выражения. Преобразования типов. Условные выражения и операторы ветвления, циклы. Функции, внешние переменные, области видимости, статические переменные, регистровые переменные. Препроцессор, заголовочные файлы и файлы с реализацией, включения файлов, модульная структура программы. Макросы и условная компиляция.
3. Указатели и массивы, выделение памяти. Адресная арифметика. Указатели и строки. Указатели на указатели, многомерные массивы. Указатели на функции. Структуры, массивы структур, указатели на структуры, структуры со ссылками на себя. Объединения. Битовые поля. typedef.

Раздел 2: Язык C++

1. Синтаксис C++, встроенные структуры данных, обзор стандартной библиотеки. Ввод-вывод средствами стандартных библиотек Си и C++. Массивы и их инициализация. Класс string, строковые литералы. Перечисления. Управление памятью через new и delete. Цикл for по диапазону. Файловый ввод-вывод на потоках. Прототипы функций. Передача по ссылке и по значению. Аргументы по умолчанию.
2. Шаблоны функций. Явные специализации. Перегрузка функций с шаблонами. decltype и trailing return types. Продолжительность хранения и области видимости. Языковое связывание функций. Пространства имён и using.
3. ООП на C++. Процедурное и объектно-ориентированное программирование, классы и интерфейсы. Модификаторы доступа. Конструкторы и деструкторы, присваивание и инициализация, ключевое слово explicit, функции преобразования типов. const-методы, this. Списковая инициализация. enum class. Перегрузка операций. Ключевое слово friend.
4. Процесс компиляции. Препроцессор. Модульная организация программ на C++.
5. Системы сборки, система сборки make. Синтаксис Makefile.
6. Указатели и ссылки в C++. const-указатели. Ключевое слово auto. Ссылочные переменные и параметры, временные переменные, потенциальные проблемы. Ссылка на rvalue.
7. Утечки памяти. Работа с утилитой valgrind.

Период обучения (модуль): семестр 3, вариант 1: C++, основы архитектуры программного обеспечения

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Наименование темы (раздела, части) | Вид учебных занятий | Количество часов |
| I. | Язык C++ | практические занятия | 11 |
| лабораторные работы | 11 |
| самостоятельная работа | 25 |
| II. | Паттерны проектирования | практические занятия | 4 |
| лабораторные работы | 4 |
| самостоятельная работа | 12 |
| III. | Промежуточная аттестация | самостоятельная работа | 3 |
| зачёт | 2 |

Раздел 1: Язык C++

1. Синтаксис C++, встроенные структуры данных, обзор стандартной библиотеки. Ввод-вывод средствами стандартных библиотек Си и C++. Массивы и их инициализация. Класс string, строковые литералы. Перечисления. Управление памятью через new и delete. Цикл for по диапазону. Файловый ввод-вывод на потоках. Прототипы функций. Передача по ссылке и по значению. Аргументы по умолчанию.
2. Шаблоны функций. Явные специализации. Перегрузка функций с шаблонами. decltype и trailing return types. Продолжительность хранения и области видимости. Языковое связывание функций. Пространства имён и using.
3. ООП на C++. Процедурное и объектно-ориентированное программирование, классы и интерфейсы. Модификаторы доступа. Конструкторы и деструкторы, присваивание и инициализация, ключевое слово explicit, функции преобразования типов. const-методы, this. Списковая инициализация. enum class. Перегрузка операций. Ключевое слово friend.
4. Процесс компиляции. Препроцессор. Модульная организация программ на C++.
5. Системы сборки, система сборки make. Синтаксис Makefile.
6. Указатели и ссылки в C++. const-указатели. Ключевое слово auto. Ссылочные переменные и параметры, временные переменные, потенциальные проблемы. Ссылка на rvalue.
7. Утечки памяти. Работа с утилитой valgrind.

Раздел 2: Паттерны проектирования

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

Период обучения (модуль): семестр 3, вариант 2: углублённый C++, основы архитектуры программного обеспечения

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Наименование темы (раздела, части) | Вид учебных занятий | Количество часов |
| I. | Углублённый C++ | практические занятия | 11 |
| лабораторные работы | 11 |
| самостоятельная работа | 22 |
| II. | Паттерны проектирования | практические занятия | 4 |
| лабораторные работы | 4 |
| самостоятельная работа | 15 |
| III. | Промежуточная аттестация | самостоятельная работа | 3 |
| зачёт | 2 |

Раздел 1: Углублённый C++

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

Раздел 2: Паттерны проектирования

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

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

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

Раздел 1: Углублённый C++

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

Раздел 2: Паттерны проектирования

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

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

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

Раздел 1: Базовое многопоточное программирование

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

Раздел 2: Асинхронное программирование, модель памяти

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

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 4: Программирование на C и C++.**

Период обучения (модуль): семестр 1. Язык C.

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Наименование темы (раздела, части) | Вид учебных занятий | Количество часов |
| I. | Язык C | практические занятия | 22 |
| лабораторные работы | 7 |
| контрольные работы | 2 |
| самостоятельная работа | 37 |
| II. | Структуры данных | практические занятия | 21 |
| лабораторные работы | 8 |
| контрольные работы | 2 |
| самостоятельная работа | 38 |
| III. | Промежуточная аттестация | самостоятельная работа | 5 |
| зачёт | 2 |

Раздел 1:

1. Блок-схемы алгоритмов.
2. Раздельная компиляция. Программы для разработки на C. Препроцессор.
3. Общая структура программы на C. Понятие типа в C. Явно выписанные константы. Переменные. Ввод значений. Арифметические операции. Функции из стандартной математической библиотеки.
4. Параметры вывода. Операции сравнения. Логические операции. Побитовые операции. Условная операция. Операция последования.
5. Оператор, сделанный из выражения. Составной оператор, локальные и глобальные переменные. Условный оператор. Оператор выбора, оператор break. Операторы цикла, операторы break и continue. Метки и оператор goto.
6. Необходимость массивов. Их объявления и определения, доступ к элементам. Указатели. Арифметика указателей. Связь с массивами.
7. Понятие функции. Параметры и результат функции. Передача параметров по значению и по ссылке. Объявление и определение функции. Статические переменные. Возврат указателя в качестве результата.
8. Динамические массивы, разреженные матрицы. Рекурсия. Функции, число и типы параметров которых заранее неизвестны.
9. Сортировка. Указатели на функции.
10. Поиск.
11. Строки.
12. Структурные типы данных. Объединения.
13. Библиотека ввода-вывода.
14. Контрольная работа.
15. Переписывание контрольной работы.

Раздел 2:

1. Связные списки.
2. Стеки и очереди.
3. Двоичные деревья.
4. Произвольные деревья.
5. Обход дерева.
6. Деревья поиска. Деревья поразрядного поиска.
7. Деревья двоичного поиска.
8. АВЛ-деревья.
9. B-деревья.
10. Красно-черные деревья.
11. Гибридные деревья (смесь поразрядного и двоичного поиска).
12. Декартовы деревья.
13. Хэш-таблицы.
14. Контрольная работа.
15. Переписывание контрольной работы.

Период обучения (модуль): семестр 2. Язык C++.

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Наименование темы (раздела, части) | Вид учебных занятий | Количество часов |
| I. | Расширение С | практические занятия | 6 |
| лабораторные работы | 8 |
| контрольные работы | 2 |
| самостоятельная работа | 16 |
| II. | Структурные типы данных в C++ | практические занятия | 5 |
| лабораторные работы | 7 |
| контрольные работы | 2 |
| самостоятельная работа | 16 |
| III. | Промежуточная аттестация | самостоятельная работа | 8 |
| зачёт | 2 |

Раздел 1:

1. Операции new и delete для управления динамической памятью. Автоматическое определение типа переменных и результатов функций, ключевое слово auto. Простейший ввод-вывод, манипуляторы. Цикл по элементам массива. Передача параметров по ссылке. Значения по умолчанию. Перегрузка функций. Возврат результатов по ссылке.
2. Подставляемые функции. Шаблонные функции. Функции, набор типов параметров которых может быть разным (через шаблоны с переменным числом параметров).
3. Функции, значения которых можно вычислять на этапе компиляции. Получение новых функций путем подстановки констант на место некоторых параметров имеющихся функций. Безымянные функции.
4. Тип string.
5. Шаблонные типы vector и list, итераторы.
6. Контрольная работа.
7. Переписывание контрольной работы.

Раздел 2:

1. Права доступа, причины сокрытия полей, ключевое слово class, друзья. Статические поля структур. Методы в структурах, в том числе статические, операция разрешения области видимости.
2. Конструкторы и деструкторы, большие и маленькие объекты, семантика перемещения.
3. Перегрузка операций и методов. Мимикрия, умные указатели, функторы.
4. Указатели на поля и методы. Шаблоны классов, специализация, явное инстанциирование, полиморфизм, обобщенное программирование.
5. Контрольная работа.
6. Переписывание контрольной работы.

Период обучения (модуль): семестр 3. Язык C++. Продолжение.

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Наименование темы (раздела, части) | Вид учебных занятий | Количество часов |
| I. | Наследование | практические занятия | 8 |
| лабораторные работы | 7 |
| самостоятельная работа | 18 |
| II. | Стандартная библиотека | практические занятия | 7 |
| лабораторные работы | 8 |
| самостоятельная работа | 19 |
| III. | Промежуточная аттестация | самостоятельная работа | 3 |
| зачёт | 2 |

Раздел 1:

1. Синтаксис и смысл наследования. Множественное наследование. Права доступа при наследовании. Права доступа protected.
2. Автоматическое преобразование указателя на производный класс к указателю на базовый. Диспетчеризация. Виртуальные деструкторы. Чистые виртуальные методы и абстрактные классы.
3. Явное преобразование указателя на базовый класс к указателю на производный. Определение типа объекта во время выполнения программы (RTTI, run-time type identification). Двойная диспетчеризация и реализация через RTTI. Виртуальные базовые классы.
4. Исключения в C++. Раскрутка стека и принцип «захват ресурса – это инициализация» (RAII, Resource Acquisition Is Initialization). Безопасность исключений.
5. Пространства имен.
6. Контрольная работа.
7. Переписывание контрольной работы.

Раздел 2:

1. Потоковый ввод-вывод.
2. Ассоциативные контейнеры.
3. Многопоточность. OpenMP.
4. async и future. Класс thread.
5. Синхронизация.
6. Структуры данных.
7. Регулярные выражения.
8. Случайные числа.
9. Время.
10. Контрольная работа.
11. Переписывание контрольной работы.

Период обучения (модуль): семестр 4.

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

Раздел 1: Конечные автоматы.

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

Раздел 2: Формальные грамматики.

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











**Вариант реализации 6: Прикладное программирование на языке C#.**

Период обучения (модуль): семестр 1. Введение в язык C#.

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Наименование темы (раздела, части) | Вид учебных занятий | Количество часов |
| I. | Введение в инструменты C# | практические занятия | 16 |
| лабораторные работы | 4 |
| контрольные работы | 4 |
| самостоятельная работа | 20 |
| II. | Алгоритмические задачи | практические занятия | 15 |
| лабораторные работы | 5 |
| контрольные работы | 2 |
| самостоятельная работа | 38 |
| III. | ООП и структуры данных | практические занятия | 12 |
| лабораторные работы | 6 |
| контрольные работы | 0 |
| самостоятельная работа | 17 |
| IV. | Промежуточная аттестация | самостоятельная работа | 5 |
| зачёт | 2 |

Раздел 1:

* + - 1. Цели, задачи и принципы .NET Framework, .NET Core. Этапы развития платформы и языка C#. Современное состояние платформы Microsoft.NET. Сравнительный анализ языков C# и C++. Just-In-Time компиляция в .NET. Механизм сборок.
      2. Среда разработки MS Visual Studio. Первая программа на C#: создание проекта, ввод-вывод на консоль.
      3. Система типов. Объявление и использование переменных. Константы. Условные операторы: if/else, switch. Операторы цикла: while, do..while, for.

*Домашняя работа 1.*

* + - 1. Значимые и ссылочные типы. Понятие о стэке и куче.
      2. Одномерные, двумерные массивы на примере матриц. Оператор цикла foreach.

*Домашняя работа 2.*

* + - 1. Контрольная работа.

Раздел 2:

* + - 1. Применение рекурсии для решения задач методом “разделяй и властвуй”.
      2. Динамическое программирование.
      3. Бинарные деревья.

*Домашняя работа 3.*

* + - 1. Некоторые виды сортировок.
      2. Примеры задач с собеседований.

Раздел 3:

* + - 1. Введение в ООП. Классы, структуры и объекты. Поля, методы и свойства. Создание и уничтожение объектов. Статические поля и методы. Исключения.

*Домашняя работа 4.*

* + - 1. Пространства имен. Использование стандартных классов System.Environment, System.Math, System.Random.
      2. Динамические структуры данных и контейнеры изнутри: список на базе массива, односвязный и двусвязный списки, очередь и стэк.

*Домашняя работа 5.*

* + - 1. Словари изнутри. Реализация словаря методом цепочек.

Период обучения (модуль): семестр 2. Прикладное программирование на языке C#.

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Наименование темы (раздела, части) | Вид учебных занятий | Количество часов |
| I. | Углубленный C# | практические занятия | 5 |
| лабораторные работы | 7 |
| контрольные работы | 2 |
| самостоятельная работа | 16 |
| II. | Проект на Windows Forms | практические занятия | 6 |
| лабораторные работы | 8 |
| контрольные работы | 2 |
| самостоятельная работа | 16 |
| III. | Промежуточная аттестация | самостоятельная работа | 8 |
| зачёт | 2 |

Раздел 1:

* + - 1. Интерфейсы и абстрактные классы.
      2. Итераторы. Создание итерируемой коллекции.
      3. Компараторы. Создание сортируемых коллекций.
      4. Обобщенные типы (генерики).

*Домашняя работа 1.*

Раздел 2:

* + - 1. Создание настольных приложений с Windows Forms.

*Домашняя работа 2.*

* + - 1. Делегаты. События.
      2. Концепция потоков и работа с файлами. Сериализация.
      3. Регулярные выражения. Лямбда-выражения.

*Домашняя работа 3.*

* + - 1. Автоматическое управление памятью. Сборка мусора. Освобождение ресурсов: финализаторы и паттерн IDisposable.

*Домашняя работа 4.*

* + - 1. LINQ и PLINQ.

Период обучения (модуль): семестр 3. Профессиональное программирование на языке C#.

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Наименование темы (раздела, части) | Вид учебных занятий | Количество часов |
| I. | Многопоточное программирование | практические занятия | 8 |
| лабораторные работы | 7 |
| самостоятельная работа | 18 |
| II. | Основы объектно-ориентированного дизайна приложений | практические занятия | 7 |
| лабораторные работы | 8 |
| самостоятельная работа | 19 |
| III. | Промежуточная аттестация | самостоятельная работа | 3 |
| зачёт | 2 |

Раздел 1:

1. Принципы многопоточного программирования, класс Thread и таймеры.
2. Синхронизация потоков.
3. Высокоуровневая работа с задачами: Parallel.For и класс Task.

*Домашняя работа 1.*

1. Асинхронное программирование через async/await.
2. Операции отмены и прогресса на базе CancellationToken и IProgress<T>.

*Домашняя работа 2.*

Раздел 2:

* + - 1. Принципы SOLID.
      2. Паттерн декоратор.
      3. Слоистая архитектура на базе паттерна Декоратор.
      4. Паттерн Фабрика.

*Домашняя работа 3.*

* + - 1. Паттерн Посетитель

*Домашняя работа 4.*

* + - 1. Model View Presenter

*Домашняя работа 5.*

**Период обучения (модуль): семестр 4.**

См. вариант реализации 4.

**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.

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 и 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 неделю 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 преподавателем. 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. Кроме этого, контрольная может содержать от одного до пяти теоретических вопросов, проверяющих владение теоретическим материалом. 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. Переписываний контрольных работ не предусмотрено.

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 запроса на ревью pull-request-а в публичном репозитории обучающегося, опубликованного на облачном хостинге систем контроля версий GitHub. Репозиторий, созданный обучающимся для работы над домашними задачами должен быть снабжён сиситемой автоматической сборки и тестирования (например, с помощью сервиса Travis CI или AppVeyor). Сборка и тестирование должны запускаться на изменения в ветках, а так же на открытие pull request-а. 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 с использованием системы TeX. Публикуются как исходники, так и скомпилированные pdf-файлы. При этом работа сдаётся точно так же, как и в случае программного кода, за исключением того, что сборка на сервере не требуется, однако компилируемость опубликованных исходных файлов в pdf является обязательным требованием для успешной сдачи. Поэтом рекомендуется снабжать отчёты скриптом сборки.

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: Промышленное программирование, C#, С++**

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 преподавателю на занятии, при этом начиная с середины первого семестра требуется выкладывание исходных кодов в один из облачных хостингов систем контроля версий (например, GitHub). После проверки обучающимся выдаётся список замечаний и текущий балл за задачу, обучающиеся вправе исправить замечания и сделать ещё одну попытку сдачи.

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

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

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

Помимо докладов и контрольных, на занятиях проводятся опросы по прочитанному материалу. Каждый опрос длится не более получаса. Обучающимся предлагается набор открытых вопросов, на которые они должны ответить в письменном виде, пользоваться источниками и общаться во время опроса запрещается (в случае нарушения этого правила за опрос выставляется 0 баллов). Переписывание опросов не предполагается.

**Вариант реализации 4: Программирование на C и C++**

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

**Вариант реализации 5: Прикладное программирование на языке C#**

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 преподавателю на занятии, при этом начиная с середины первого семестра требуется выкладывание исходных кодов в один из облачных хостингов систем контроля версий (например, GitHub). После проверки обучающимся выдаётся список замечаний и текущий балл за задачу, обучающиеся вправе исправить замечания и сделать ещё одну попытку сдачи.

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

***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. Обучающийся имеет право один раз продлить срок выполнения задачи, но не более, чем на две недели. Каждая задача также имеет фиксированное количество «бесплатных» попыток сдачи (по умолчанию три), после которых каждая следующая попытка сдачи приводит к вычитанию 0.5 балла из максимального балла за задачу. При этом исправления замечаний должны быть сданы за неделю после получения замечаний, иначе за каждую неделю пропуска срока начиная с первой вычитается 0.5 балла из максимального балла. Попытка сдачи, в которой не реализованы все требования условия, уменьшает максимальный балл за задачу пропорционально объёму нереализованных требований. Штрафы к максимальному баллу, полученные таким образом, никогда не могут быть меньше текущего балла, при этом текущий балл никогда не может уменьшаться. В случае, если максимальный балл упал до уровня текущего, балл за задачу фиксируется и дальнейшие исправления к ней не принимаются. Также преподаватель может доводить до сведения обучающихся типичные ошибки и штрафы за них, после чего новые решения, содержащие такие ошибки, невосстановимо теряют баллы. Штрафы, полученные таким образом, не могут уменьшить максимальный балл за задачу более чем на 50% от его исходного значения (то есть за задачу всегда можно получить половину баллов, если она полностью сделана).

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

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

Критерии оценивания каждой задачи (и домашних, и контрольных) предполагают использование всего пройденного материала. 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, баллы за контрольные и баллы за зачётную работу 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. Шкала оценивания для каждой задачи имеет два значения: минимальное (0, задача не решена) и максимальное (задача решена с соблюдением всех требований, значение сообщается обучающимся вместе с задачей). Результирующая оценка за все задачи находится в диапазоне от 0 до 70 и вычисляется как сумма баллов за все задачи.

Каждое задание имеет срок выполнения (по умолчанию 2 недели, но данный срок может варьироваться и сообщается обучающимся вместе с задачей), после истечения срока задачи не принимаются. Для сдачи каждой работы есть 3 попытки, при этом разбиение на задачи между попытками никак не учитывается (за исключением ситуаций, когда задачи зависимы между собой). Все замечания должны быть исправлены в течении недели с того момента, как были получены. При этом установленный срок сдачи приорететнее количества попыток.

Например, пусть есть домашняя работа из 6 задач и сроком сдачи 3 недели. Обучающийся А в первой попытке сдал первые две задачи через 3 дня после того, как работа была выдана, и через день получи замечания. Через 13 дней он сдал исправленные задачи и ещё 2 новых (сделал вторую попытку), к которым также получил замечания. Теперь у него есть неделя на исправления замечаний, но сдать ещё 2 задачи он не может, так как истёк срок (хотя и осталась ещё одна попытка).

На контрольных работах каждая задача оценивается от 0 до n баллов, где 0 — полное отсутствие решения, n — правильное решение, демонстрирующее все ожидаемые навыки. Максимальное значение баллов за задачу сообщается вместе с задачей. Баллы за задачи суммируются и итоговый балл за все контрольные в семестре вычисляется как сумма баллов за все задачи. Переписываний контрольных не предусмотрено.

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 суммируются, в результате чего получается число от 0 до 100 (это гарантируется распределением баллов за домашние и контрольные: обычно примерно 70 баллов за все домашние задачи в семестре и 30 за все контрольные/проверочные). Это значение и становится итоговым процентом освоения дисциплины.

**Вариант реализации 3: Промышленное программирование, C#, С++**

Итоговый процент освоения дисциплины складывается из баллов за домашние задания, баллов за контрольные работы, баллов за доклады, если они предусмотрены, и баллов за письменные опросы.

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.

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.

Опросы оцениваются по линейной шкале от 0 до 10, где 0 — ответы на все вопросы отсутствуют или неправильны, либо выявлено нарушение правил опроса, 10 — все ответы правильные и достаточно полны.

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

**Вариант реализации 4: Программирование на C и C++**

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

**Вариант реализации 6: Прикладное программирование на языке C#**

Итоговый процент освоения дисциплины складывается из баллов за домашние задания и баллов за контрольные работы. Каждое успешное задание оценивается в 1 балл. Кроме того, за пропуск каждого занятия могут начисляться отрицательные баллы по усмотрению преподавателя.

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

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

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

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

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

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

**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.

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

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

**ПКП-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.

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

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

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

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

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

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

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

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

**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. Отсортировать массив какой-либо из квадратичных сортировок
2. Реализовать преобразование Барроуза-Уилера. На вход подаётся строка, на выходе должна получиться строка, преобразованная Барроузом-Уилером, и позиция конца строки в результате преобразования. Реализовать также и обратное преобразование, принимающее преобразованную строку и позицию, и возвращающую исходную строку. Проверить, что исходная строка действительно восстанавливается.
   1. Решение, где в прямом преобразовании явно строится таблица вращений строк («наивный BWT»), оценивается на один балл ниже максимума.
   2. Решение, где в обратном преобразовании не используется построение циклической перестановки, а явно сортируются строки и строится таблица («наивный обратный BWT»), также лишает вас одного балла.
   3. Использование в явном виде символа конца строки также даёт минус балл (поскольку исходная строка, вообще говоря, может содержать любой символ — преобразование Барроуза-Уилера используется и для сжатия бинарных файлов, где может встречаться вообще всё, что угодно)

***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 1 (решение работоспособно, аккуратно реализовано, алгоритмически аккуратно). Решение задачи 2 оценивается по шкале от 0 (нет решения или решение имеет существенные недостатки) до 5 (решение работоспособно, аккуратно реализовано, нет проблем, описанных в пунктах a-c).

***Домашняя работа 2:***

1. Написать в виде класса структуру данных "Бор" ([https://neerc.ifmo.ru/wiki/index.php?title=Бор](https://neerc.ifmo.ru/wiki/index.php?title=%D0%91%D0%BE%D1%80)) и реализовать с его помощью алгоритм Лемпеля — Зива — Уэлча. Требуется написать консольное приложение, которое в качестве аргумента командной строки принимает путь к файлу, который надо сжать или разжать, и ключ -c, означающий, что файл надо сжать, или -u, означающий, что надо разжать. В качестве результата должен создаваться новый файл:
   1. если выполняется сжатие, файл именуется как <имя изначального файла>.zipped
   2. если выполняется разжатие, расширение .zipped отбрасывается.

Вы вправе в сжатом файле хранить любую дополнительную информацию.

Программа после сжатия должна печатать коэффициент сжатия на консоль.

Применение преобразования Барроуза-Уилера из предыдущего задания и исследование его влияния на коэффициент сжатия принесёт вам один дополнительный балл.

1. 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 (то есть вообще в коде класса «Стековый калькулятор» не должно быть ни одного упоминания конкретных реализаций стека). Числа и арифметические знаки разделены пробелами, числа только целые (но могут быть знаковыми, и уж точно не только из одной цифры, используйте int.Parse или int.TryParse). В результате должно получаться число — результат вычислений. Результат может быть дробным. При попытке деления на 0 должна выдаваться ошибка и программа должна корректно заканчивать работу.

Напомним, что в силу особенностей представления вещественных чисел в памяти (вспомните, каких) сравнение с нулём дробного числа оператором == сразу лишает вас одного балла.

***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 7 (решение работоспособно, аккуратно реализовано, алгоритмически аккуратно), выполнение дополнительного условия приносит один дополнительный балл, не считающийся в максимум. Решение задачи 2 оценивается по шкале от 0 (нет решения или решение имеет существенные недостатки) до 4 (решение работоспособно, аккуратно реализовано, правильно с точки зрения ООП и использования C#).

***Домашняя работа 3:***

1. Реализовать словарь, отображающий строковые ключи в строковые значения, на основе B-дерева. Должны быть поддержаны операции добавления пары «ключ-значение», возврата значения по ключу, проверки наличия ключа в дереве, изменения значения по ключу, удаления значения по ключу. Минимальная степень дерева (число ключей в каждом узле) должно приниматься в конструктор.

Юнит-тесты на все операции для разных случаев (операции с корнем, внутренними узлами, листами) обязательны.

***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, алгоритмически аккуратно).

***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. С использованием подхода Data-Driven Testing реализовать модульные тесты к задаче 2 домашней работы 2. Модульные тесты должны проверять работоспособность стекового калькулятора на корректных и некорректных строках, в случае использования разных стеков, при этом не содержать дублирующегося кода. Настроить сборку и запуск модульных тестов на AppVeyor для этой задачи.
2. Есть участок сети, состоящий из роутеров, связанных Ethernet-соединениями. Поскольку разные куски этой сети администрируют разные организации, у сети отсутствует единая архитектура, что часто приводит к избыточным соединениям между роутерами, либо наоборот, изоляции участков сети. Современные сетевые протоколы устроены так, что избыточные соединения почти столь же опасны, как и их отсутствие — если роутер не знает маршрута до целевого узла, он рассылает пакеты по всем портам, кроме того, откуда пакет пришёл, в надежде, что кто-то из роутеров-адресатов сможет его доставить. Поэтому не исключена ситуация, когда пакеты начинают ходить по кругу до тех пор, пока не исчерпается их время жизни (Time To Live, TTL), что нагружает сеть и снижает общую производительность.

Ваша задача — написать утилиту, которая по данной топологии сети генерирует конфигурацию для каждого роутера и проверяет, что все роутеры достижимы. Топология задана в файле в виде списка роутеров и того, к каким другим роутерам они подключены каналами какой пропускной способности, например,

1: 2 (10), 3 (5)

2: 3 (1)

задаёт сеть из трёх роутеров, где первый связан со вторым и третьим, второй с первым и третьим, третий с первым и вторым. Причём, канал между первым и вторым имеет в десять раз большую пропускную способность, чем между вторым и третьим. Вы должны вывести в файл аналогичную таблицу, где оставлены только те соединения, которые необходимы для обеспечения связности сети, без циклов, например,

1: 2 (10), 3 (5)

Причём конфигурация должна быть в каком-то смысле оптимальной: сумма пропускных способностей всех каналов в сети должна быть максимальной.

Если построить такую таблицу невозможно (то есть сеть изначально была не связной), программа должна вывести в поток ошибок (обратите внимание, не в файл и не совсем на консоль) сообщение, что сеть не связна, и завершить работу с ненулевым кодом возврата. Пути до входного и выходного файлов должны приниматься в качестве параметров.

Сборка и запуск юнит-тестов на AppVeyor для этой задачи тоже обязательны.

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

***Критерии оценивания*:** решение задачи 1 оценивается по шкале от 0 (нет) до 3 (решение работоспособно и адекватно, модульные тесты запускаются в CI-системе). Решение задачи 2 оценивается по шкале от 0 до 7 по тем же критериям.

***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. Каждый раз перерисовывать всю карту нельзя.

***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). Калькулятор должен вычислять операторы немедленно, то есть если пользователь нажимает «7», «+», «3», «+», на экране должно отобразиться «10». Ввод кнопочный, то есть разбор и прямое редактирование выражения делать не надо (соответственно, скобки, приоритет операций и т.п. калькулятор не должен поддерживать). Рекомендуется вспомнить про конечные автоматы для упрощения формализации вещей в духе «если оператор нажат первый раз, ждём второй операнд, если второй операнд уже есть, печатаем ответ и запоминаем оператор».

Юнит-тесты здесь обязательны на всю функциональность, не связанную непосредственно со взаимодействием с пользователем, поэтому стоит сразу реализовывать калькулятор правильно — тонкий и глупый слой пользовательского интерфейса и отдельно класс с бизнес-логикой, который даже не знает про существование GUI.

Использование data binding для отображения результатов вычисления принесёт вам один дополнительный балл сверх максимума за эту задачу.

1. 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. Переделать B-дерево из домашней работы 3 так, чтобы оно реализовывало интерфейс System.Collections.Generic.IDictionary. Проверить в тестах, что по словарю можно ходить foreach. Следовать всем правилам реализации структуры данных в библиотеке: проверять все аргументы всех public-методов, реализовать инвалидацию итератора при изменении коллекции.

***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, 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).

Плюс 4 балла сверх максимума за реализацию по созданному проекту (но именно в таком порядке: сначала сдаёте диаграмму, получаете одобрение, только потом садитесь писать код).

***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 должно как-то удобно задаваться) 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 гнетущую 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, Selenium, White
* yield return, ленивые вычисления
* Историческая справка: .NET Framework, .NET Core, Mono
* Обзор библиотеки 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, Параллельное умножение матриц:***

Одна из самых полезных и вместе с тем хорошо параллелящихся задач — умножение матриц. Это часто используется не только в научных расчётах, но и при практически любой работе с графикой, особенно трёхмерной. Фактически, современные видеокарты — это специализированные вычислители, умеющие умножать матрицы (в частности, вектора на матрицы) очень эффективно за счёт большого количества вычислительных узлов (до сотен в современных выделенных видеокартах). Естественно, параллельно. Вам надо попробовать поумножать матрицы с помощью обычного многоядерного процессора.

Требуется реализовать параллельное умножение для плотных целочисленных матриц. На входе программа получает два файла с матрицами (не обязательно квадратными), на выходе должен получиться файл, содержащий матрицу — их произведение. Сравнить скорость работы с последовательным вариантом в зависимости от размеров матриц, причём сделать это по возможности грамотно.

* Для каждого тестового случая выполнить N запусков, посчитать матожидание и среднеквадратичное отклонение.
* Сделать это для нескольких разных и достаточно больших размеров матриц.
* Свести результаты в таблицу.
* +1 балл за построение графика по результатам тестового прогона.

Попробовать получить возможно большее ускорение.

Можно использовать только класс Thread для организации параллельной работы.

Обратите внимание, что распараллеливание имеет смысл только на достаточно больших данных, так что требуется также уметь генерировать большие тестовые данные и найти такие размеры данных, при которых различия в скорости работы будут заметны и значительны.

В качестве решения приложите пуллреквест в собственный репозиторий на GitHub. Как обычно, ожидаются комментарии, юнит-тесты, CI, следование правилам стайлгайда и здравого смысла.

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

***Критерии оценивания*:** решение задачи оценивается от 0 до 4 баллов, где 0 — полное отсутствие решения, 4 — правильное решение, демонстрирующее все ожидаемые навыки (комментарии, тесты, навык использования системы контроля версий, модульного тестирования и непрерывной интеграции). Дополнительно один балл может быть начислен за выполнение дополнительного требования условия.***Домашняя работа 2, 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

Интерфейс должен быть реализован двумя способами:

* 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 3, 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 4, 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 5, 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 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).

**Семестр 4.**

***Домашняя работа 1:***

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

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

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

***Домашняя работа 2:***

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

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

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

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

***Домашняя работа 3:***

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

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

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

***Домашняя работа 4:***

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

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

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

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

***Домашняя работа 5:***

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

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

Требуется периодически выводить состояние сети.

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

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

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

***Домашняя работа 6:***

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

rounding 3 {

let! a = 2.0 / 12.0

let! b = 3.5

return a / b

}

должно возвращать 0.048

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

let result = calculate {

let! x = "1"

let! y = "2"

let z = x + y

return z

}

должно возвращать значение, содержащее 3, а

let res = calculate {

let! x = "1"

let! y = "Ъ"

let z = x + y

return z

}

должно возвращать значение, указывающее на отсутствие результата.

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

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

***Домашняя работа 7:***

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

type ILazy<'a> =

abstract member Get: unit -> 'a

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

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

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

static member CreateSingleThreadedLazy supplier

возвращающих три разные реализации Lazy<'a>:

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

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

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

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

***Домашняя работа 8:***

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

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

let K = \x y.x

S K K

Должно получиться в качестве ответа:

\x.x

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

Для синтаксического анализа использовать FParsec.

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

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

***Темы докладов***

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

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

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

***Примеры условий контрольных работ:***

***Контрольная работа 1***

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

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

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

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

***Контрольная работа 2***

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

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

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

*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-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, ПКП-3, ПКП-4

***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-5, ПКП-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, ПКП-3, ПКП-4

***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, 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-5, ПКП-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, ПКП-3, ПКП-4

***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-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 балла) 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 балла) Реализовать построение автомата по регулярному выражению. В качестве основы использовать представление матриц из предыдущей работы. Разбор входного регулярного выражения осуществлять средствами из предыдущей задачи.
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-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-5, ПКП-3, ПКП-4

***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-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, ПКП-3, ПКП-4

***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-5, ПКП-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 балла) Внедрить RxExtensions для работы с событиями в разработанной ранее IDE.

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

***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 балла) 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.

**Семестр 4.**

***Домашняя работа 1:***

Для выполнения данной работы можно использовать любые технологии, однако нужно предусмотреть возможности использования GPGPU в разрабатываемом приложении. В качестве дефолтного варианта предлагается .NET.

1. Реализовать приложение с графическим интерфейсом пользователя, позволяющее открыть папку с изображениями, выбрать изображение, просмотреть его, просмотреть информацию о нём (размер в пикселях, размер в мегабайтах).

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

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

***Домашняя работа 2:***

Для выполнения данной работы можно использовать любые технологии, однако нужно предусмотреть возможности использования GPGPU в разрабатываемом приложении. В качестве дефолтного варианта предлагается .NET.

1. (1 балл) Расширить приложение из предыдущей домашней работы графической компонентой задания матричного фильтра. Необходимо предусмотреть возможность выбора типа фильтра, дефолтных значений, размера фильтра, корректировку весов.
2. (1 балл) Расширить приложение из предыдущей работы возможностью отображать одновременно два изображения: до и после применения фильтра. Предусмотреть возможность сохранять результат применения фильтра.
3. (3 балла) Реализовать применение матричных фильтров с использованием GPGPU. Интегрировать с разработанным графическим интерфейсом. Предусмотреть возможность применения нескольких фильтров последовательно.
4. (6 баллов) Расширить разрабатываемое приложение возможностью потоковой обработки изображений: выбираем папку с изображениями и ко всем применяем заданный фильтр. Результаты применения фильтров сохраняются.
5. (5 баллов) Подготовить отчёт с анализом производительности и масштабируемости полученного решения.

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

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

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

***Домашняя работа 3:***

1. (3 балл) Выбрать минимум три различных сценария использования разреженных матриц (различные матричные операции и различные представления матриц), реализуемые одновременно в библиотеках pygraphblas и scipy. Провести сравнение производительности этих библиотек на выбранных сценариях. Подготовить отчёт.
2. (6 баллов) Выбрать две различные библиотеки линейной алгебры на GPGPU. Выбрать минимум три различных сценария использования разреженных матриц (различные матричные операции и различные представления матриц), реализуемые одновременно в выбранных библиотеках. Провести сравнение производительности этих библиотек на выбранных сценариях. Подготовить отчёт.

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

***Критерии оценивания*:** решение задач 1 оценивается по шкале от 0 (нет решения или решение имеет существенные недостатки) до 3 (решение правильно, отчёт аккуратно свёрстан в TeX). Решение задачи 6 оценивается по шкале от 0 (нет решения или решение имеет существенные недостатки) до 3 (решение правильно, отчёт аккуратно свёрстан в TeX).

***Домашняя работа 4:***

В данной работе допускается использование любых языков программирования на GPGPU.

1. (3 балла) Выбрать задачу анализа графов, легко формулируемую в терминах линейной алгебры (BFS, минимальное остовное дерево, подсчёт треугольников). Провести анализ алгоритмов решения этой задачи в терминах линейной алгебры. Выбрать лучший, выяснить, какие примитивы и операции над ними нужны для реализации выбранного алгоритма. Внимание, наивный алгоритм есть в каждой задаче, но его выбирать нельзя. Он почти всегда самый медленный и обосновать выбор не получится.
2. (6 баллов) Реализовать необходимые для реализации алгоритма из предыдущей задачи примитивы и операции над ними на GPGPU. Необходимо использовать разреженные представления матриц и векторов. Решение должно быть оформлено в виде самостоятельной библиотеки.
3. (3 балла) Реализовать выбранный в задаче 1 алгоритм на основе библиотеки из задачи 2.
4. (5 балла) Провести анализ производительности полученного решения и его сравнение с решением на pygraphblas. Оформить отчёт.

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

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

***Домашняя работа 5:***

1. (7 баллов) Провести анализ временной сложности алгоритма из предыдущей домашней работы. Учесть его параллельную реализацию и особенности представления структур данных.

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

***Критерии оценивания*:** решение задачи оценивается по шкале от 0 (нет решения или решение имеет существенные недостатки) до 7 (решение корректно, грамотно изложено, аккуратно оформлено).

***Темы докладов***

1. FAKE, Scaffold
2. Веб-приложения в комбинаторном стиле
3. Type Providers, F# Data
4. MBrace
5. Alea CUDA

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

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

***Условия контрольных работ:***

Задачи для контрольных работ основаны на изученном материале, связаны с домашними заданиями и подразумевают владение как практическими, так и теоретическими навыками.

**Вариант реализации 3: Промышленное программирование, C#, С++**

***Примеры вопросов опроса:***

**Семестр 1**

1. Опишите сигнатуру метода Main
2. Приведите код на С#, позволяющий разобрать параметры командной строки
3. Приведите и кратко охарактеризуйте методы консольного ввода-вывода
4. Опишите в общих словах (или в виде небольшой схемы) иерархию системных
5. типов.
6. Что значит, что строки в С# неизменяемы?
7. Что такое сужающее/расширяющее преобразование типов? Какие ошибки можно
8. допустить?
9. В какой момент происходит выведение типа неявно типизированной переменной?
10. Что такое необязательные аргументы? Какие ошибки возникнут при компиляции
11. метода static void EnterLogData ( string message, DateTime timestamp =
12. DateTime.Now, string owner ) и почему?
13. Приведите код объявления массива, в котором можно хранить все что угодно.
14. Что такое структура (struct) в С#? Для чего используется?
15. Опишите особенности передачи ссылочного типа по значению и по ссылке.
16. Объясните семантику операции ??.

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

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

**Семестр 2, вариант 1, C#.**

1. Назовите и опишите открытые методы System.Object.
2. Что будет, если мы попытаемся явно привести объект к некоторому типу, пари этом объект не принадлежит к типу, его наследникам, его предкам?
3. Опишите как использовать операторы as и is, как сделать приведение эффективно.
4. Что такое пространства имен, зачем нужны?
5. Что такое объект-тип и объект? Как связаны?
6. Опишите тип данных System.Single.
7. Каково назначение ключевых слов checked и unchecked?
8. Когда следует использовать ссылочный и значащий типы в ЯП C#? Что нужно учитывать?
9. Как осуществить приведение упакованного типа в C# с Int32 в Int16 (распакованный)?
10. Какие 4 характеристики, присущие равенству, должны обеспечиваться при перегрузке операции Equals?
11. Опишите смысл использования объявления partial (2-3 случая).
12. Опишите влияние ключевого слова ЯП C# (abstract, virtual, override, sealed, new) на (тип, метод/свойство/событие, константу/поле). Выберите любые две комбинации с разным характером влияния.
13. Зачем пытаются минимизировать количество виртуальных методов?
14. Назовите правила проектирования классов, 3-4 штуки.

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

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

**Семестр 2 вариант 2, С++.**

1. Сравните составные типы «указатель» и «ссылка».
2. Можно ли «перенаправить» ссылку на новый объект? Что произойдет в следующем коде:
3. int &ref1 = …;
4. int &ref2 = …;
5. int x = 5;
6. ------
7. ref1 = x;
8. ref1 = ref2;
9. В чем отличие модификаторов const верхнего и нижнего уровней? На константность чего указывают разные уровни? Какие могут иметь ссылка и указатель?
10. Что такое константное выражение? Как сделать переменную или функцию константным выражением?
11. Какой тип будет иметь var и почему:
12. ```
13. // pseudo\_type объявлен как псевдоним int \*
14. const pseudo\_type var;
15. ```
16. Опишите основные отличие auto и decltype при наличии модификатора const, ссылок и указателей (в плане того, какой тип будет выведен)
17. Почему заголовки не должны содержать объявления using namespace?
18. Файлы-заголовки библиотеки C имеют C++ аналоги. В чем их отличие? Какие и почему следует использовать в программах на C++?
19. Для чего в языке C++ используются итераторы? Какие основные операции поддерживают? Приведите пример.
20. При передаче в функцию массивы преобразуются в указатели и теряют свой размер. Как справиться с этим (хотя бы два способа)?

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

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

**Семестр 3, вариант 1, начальный C++**

1. Какие парадигмы программирования представлены в ЯП C++?
2. Опишите суть нисходящего проектирования.
3. Что такое переносимая программа? Какие существуют проблемы?
4. Опишите основные вехи стандартизации языка C++.
5. Объясните смысл директивы using (поверхностно). Частичный импорт.
6. endl и “\n”: концептуальная разница и соглашения.
7. Вложенные функции в C и C++ по сравнению с другими ЯП.
8. Чем плоха идея использовать имена переменных наподобие \_x и \_\_y?
9. Зачем нужны объявления наподобие int numbers = {24}? Какая цель преследуется?
10. Как задать константу в C++. Чем отличается от define?
11. Списковая инициализация и безопасность типов.

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

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

**Семестр 3, вариант 2, углублённый C++**

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

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

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

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

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

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

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

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

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

***Примеры домашних задач***

Порядок выдачи домашних задач не привязан к конкретным занятиям, задачи выбираются самими обучающимися индивидуально и выполняются по мере изучения соответствующего материала (в том числе, самостоятельно или на курсе «Дискретная математика»). К концу каждого семестра для аттестации требуется набрать определённое минимальное количество баллов.

***Тема «Сортировки»***

1. Реализовать быструю сортировку и сортировку пузырьком. Сравнить производительность на различных размерах данных. Найти размер данных, когда разница времени исполнения становится видна.
2. Реализовать сортировку вставками и сортировку Шелла. Сравнить производительность на различных размерах данных. Найти размер данных, когда разница времени исполнения становится видна.
3. Реализовать сортировку слиянием для N отсортированных массивов.
4. Реализовать на массиве сортировку пирамидой (heapsort).

***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).

***Тема «Рекурсия»***

1. Вычислить рекурсивно детерминант матрицы N\*N.
2. Вычислить N-ю строку треугольника Паскаля рекурсивным и итеративным способом.
3. Сгенерировать все перестановки из набора 1, 2, 3, ..., N с помощью рекурсивного алгоритма.
4. Есть наборы бусинок различных цветов: Красный (K штук), Синий (L штук), Зеленый (M штук). Существуют правила: синяя не может идти за красной, красная не может быть на четной позиции. С помощью рекурсивной функции подсчитать сколько может быть собрано различных бус длины M, по данным правилам. На место склейки правила не распространяется.

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

***Критерии оценивания*:** решения задач 1 и 2 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 по шкале от 0 до 2, решения задачи 3 оцениваются по тем же критериям по шкале от 0 до 3.

***Тема «Двоичное представление»***

1. Вычислить количество единиц в двоичной записи числа N, стоящих на четных позициях.
2. Дано кольцо, состоящее из нулей и единиц, длины M. Вывести все числа, которые можно получить с помощью разрезания, в десятичной системе счисления.
3. Сгенерировать все числа менее N, в двоичной записи которых есть две единицы подряд.
4. Реализовать функцию, которая генерирует все числа меньшие N, в двоичном представлении которых есть заданный образец.

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

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

***Тема «Матрицы и массивы»***

1. Вычислить сумму следа квадратной матрицы.
2. Реализовать функцию перемножения матрицы на вектор.
3. Вычислить разницу между наддиагональной и поддиагональной частями матрицы.
4. Реализовать функцию транспонирования матрицы.

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

***Критерии оценивания*:** решения задач 1 и 2 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 по шкале от 0 до 2.

***Тема «Файлы и строки»***

1. Во входном файле задан текст. Вывести в выходной файл сроки текста в обратном порядке. Обратить так же и содержимое каждой строки.
2. Во входном файле задан текст. Совершить замену всех вхождений одной подстроки на другую, вывести в файл. Считать что шаблон помещается в строку целиком.
3. Во входном файле задан текст. Вывести в выходной файл сроки текста в обратном порядке. Обратить так же и содержимое каждой строки. При этом буфер для хранения строк ограничен несколькими строками.
4. Во входном файле задан текст. Вывести в выходной файл строки текста в обратном порядке. Обратить так же и содержимое каждой строки. При этом буфер для хранения строк ограничен, помещается только часть строки.

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

***Критерии оценивания*:** решения задач 1 и 2 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 по шкале от 0 до 2, решения задачи 3 — по шкале от 0 до 3.

***Тема «Перебор»***

1. Расставить по шахматной доске 8 ферзей так чтобы они не били друг друга, генерировать их таким образом, чтобы первый ферзь ставился случайно (можно в первый ряд/строку).
2. Магический квадрат — это матрица размера 3 \* 3, где в ячейках стоят целые числа из интервала [1..9] плюс выполняются определенные правила. С помощью перебора необходимо сформировать магический квадрат. Первая цифра ставится в середину, выбирается случайным образом.
3. Дан случайный набор чисел, разбить их на две группы минимизировав разницу суммарного веса каждой.
4. Решить перебором задачу о рюкзаке: дан список вещей, имеющих вес и стоимость. Набрать в лимит веса L вещей так, чтобы сформированный рюкзак имел максимальную стоимость. «Пилить» вещь нельзя.

***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 задачи 4 оцениваются по тем же критериям по шкале от 0 до 2.

***Тема «Бинарные деревья и поиск»***

1. Реализовать бинарный поиск в отсортированном массиве. Так же реализовать операции вставки и удаления с сохранением порядка.
2. Реализовать бинарный поиск в массиве с использованием компаратора (корректную вставку, удаление).
3. Реализовать объект «куча» — в массиве записано двоичное дерево, на позиции 2 \* n и 2 \* n + 1 стоят соответственно левый и правый ребенок. Должны быть операции добавления, удаления, поиск и обход (любой).
4. В объекте «куча» реализовать три обхода, описанных в википедии.

***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 по шкале от 0 до 2, решения задачи 4 — по шкале от 0 до 3.

***Тема «Перестановки»***

1. Найти ранг (степень) перестановки.
2. Сгенерировать перестановку по номеру.
3. Получить номер по перестановке.
4. Найти количество циклов в перестановке и вывести их.

***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).

***Тема «Графы»***

1. Написать программу, конвертирующую граф, представленный матрицей смежности, в матрицу инцидентности.
2. Написать программу, конвертирующую граф, представленный матрицей инцидентностей, в представление списком ребер.
3. Написать программу, которая строит матрицу инцидентности по заданному списку ребер.
4. Степенью вершины называется количество ребер, ей инцидентных. Вывести все вершины, имеющую четную степень у заданного графа. Метод задания графа на выбор.

***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).

***Тема «Алгоритмы обхода графов»***

1. С помощью модифицированного алгоритма Флойда-Уоршелла, для заданной вершины X найти, сколько шагов потребуется для достижения любой другой вершины (шаг — это переход по ребру).
2. Дан неориентированный граф, ребра содержат веса. Найти кратчайший путь между двумя заданными вершинами. Однако, если степень вершины четная — через нее нельзя пройти.
3. Дан неориентированный граф, проверить является ли он связным. Метод задания графа на выбор.
4. Модифицировать алгоритм нахождения минимального остовного дерева в графе таким образом, чтобы на шаге добавления четного ребра в расчет принималось первое значение из пары, а при нечетном шаге — второе. Пара представляет собой вес ребра.

***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 2 (решение работоспособно, аккуратно реализовано).

***Тема «Стек и очередь»***

1. Реализовать стек или очередь с использованием массива фиксированной длины.
2. Реализовать очередь или стек на линейных списках.
3. Смоделировать очередь с использованием двух стеков.
4. Смоделировать стек с использованием двух очередей.

***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 1 (решение работоспособно, аккуратно реализовано), решения задач 2-3 – по тем же критериям, но по шкале от 0 до 2.

***Тема «Списки, хеш-таблицы, сбалансированные деревья»***

1. Удалить из заданного списка за один проход все подряд идущие повторяющиеся элементы.
2. Реализовать операцию «фильтрация по предикату», удаляющую за один проход из списка все элементы, не удовлетворяющие ему.
3. Дано 2−3-дерево. Напечатать все его элементы из интервала [A, B] без лишнего просмотра.
4. Даны два бинарных дерева поиска. За O(M + N) проверить, содержат ли они в себе одинаковый набор элементов, новые структуры данных заводить нельзя.

***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).

***Тема «Объектно-ориентированное программирование»***

*Задача «Комплексные числа»*

Реализовать комплексные числа как объект. Должен поддерживать методы:

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

***Проверяемые компетенции*:** ОПК-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).

*Задача «Стек»*

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

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

***Проверяемые компетенции*:** ОПК-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).

*Задача «Компьютерная сеть»*

Реализовать в объектах симулятор распространения вирусов в компьютерной сети.

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

***Проверяемые компетенции*:** ОПК-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).

*Задача «Кинотеатр»*

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

Запросы:

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

Посчитать некоторую программу, поданную на вход.

***Проверяемые компетенции*:** ОПК-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).

***Темы докладов по паттернам проектирования (3 семестр):***

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

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

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

**Семестр 4, вариант 1, углублённый C++**

***Пример теоретического опроса:***

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

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

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

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

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

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

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

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

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

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

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

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

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

***Примеры домашних заданий:***

*Задача «Комплексные числа»*

Реализовать комплексные числа как объект. Должен поддерживать методы:

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

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

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

*Задача «Стек»*

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

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

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

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

*Задача «Компьютерная сеть»*

Реализовать в объектах симулятор распространения вирусов в компьютерной сети.

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

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

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

*Задача «Кинотеатр»*

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

Запросы:

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

Посчитать некоторую программу, поданную на вход.

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

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

**Семестр 4, вариант 2, многопоточное программирование на C++**

***Примеры домашних заданий***

*Задача 1*:

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

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

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

*Задача 2*:

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

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

Сравнить производительность в каждом случае на нормальном количестве потоков.

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

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

*Задача 3*:

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

1. Put
2. Remove
3. Check

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

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

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

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

*Задача 4, «Логгер»*:

Для реального программного обеспечения характерна следующая ситуация:

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

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

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

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

Вам необходимо рассмотреть 4 отдельных случая:

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

Примечания:

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

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

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

**Вариант реализации 4: Программирование на C и C++.**

Примерные варианты контрольных работ и вопросов к зачету:

**Семестр 1.**

***Контрольная работа 1:***

1. Придумать алгоритм для следующей задачи и представить его в виде диаграммы: ввести 20 вещественных чисел и вывести среднее арифметическое максимального и минимального из них.
2. Написать 3 заголовочных файла так, чтобы можно было включить не более двух из них любое количество раз, а включение всех трех приводило к ошибке.
3. Написать программу, вводящую число лет и начальную сумму вклада и выводящую минимальное значение процентной ставки, при которой через указанное число лет сумма вклада превысит миллион (без циклов).
4. Определить, сколько раз встречается 101 в двоичном представлении целого положительного числа (в двоичном представлении 1010110111 оно встречается 3 раза).
5. Ввести значение параметра и вывести число решений уравнения на интервале .
6. Ввести 20 вещественных чисел и посчитать, сколько раз сумма трех рядом стоящих чисел больше 100 (без массивов).

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

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

***Контрольная работа 2:***

1. Написать функцию dg с параметром-указателем на корень дерева, возвращающую указатель на узел, полученный из корня перемещением на один шаг в сына с максимальными данными (если есть всего один сын, перемещаемся в него, а если данные одинаковы — годится любой из сыновей), пока не окажемся в листе, указатель на который и надо вернуть.
2. Написать функцию addGR с параметром-указателем на корень дерева, добавляющую новый узел с данными 0 на место правого сына самого правого узла.
3. Написать функцию down\_max с параметром-указателем на корень дерева, возвращающую указатель на лист, полученный перемещениями в сына с максимальными данными (если у текущего узла есть только один сын, перемещаемся в него).
4. Написать функцию insertXY с параметром-указателем на корень дерева и двумя параметрами x и y типа T и вставляющую нового сына корня (без потомков) с данными x непосредственно перед первым сыном корня со значением данных y (можно предполагать, что такой сын корня существует).
5. Написать функцию sum\_alt с параметром-указателем на корень дерева, возвращающую сумму данных в узлах дерева, в которую правые сыновья своих отцов входят со знаком «+», а левые — со знаком «––». Данные корня в этой сумме не участвуют.

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

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

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

1. Блок-схемы алгоритмов.
2. Раздельная компиляция. Программы для разработки на C. Препроцессор.
3. Общая структура программы на C. Понятие типа в C. Явно выписанные константы. Переменные. Ввод значений. Арифметические операции. Функции из стандартной математической библиотеки.
4. Параметры вывода. Операции сравнения. Логические операции. Побитовые операции. Условная операция. Операция последования.
5. Оператор, сделанный из выражения. Составной оператор, локальные и глобальные переменные. Условный оператор. Оператор выбора, оператор break. Операторы цикла, операторы break и continue. Метки и оператор goto.
6. Необходимость массивов. Их объявления и определения, доступ к элементам. Указатели. Арифметика указателей. Связь с массивами.
7. Понятие функции. Параметры и результат функции. Передача параметров по значению и по ссылке. Объявление и определение функции. Статические переменные. Возврат указателя в качестве результата.
8. Динамические массивы, разреженные матрицы. Рекурсия. Функции, число и типы параметров которых заранее неизвестны.
9. Сортировка. Указатели на функции.
10. Поиск.
11. Строки.
12. Структурные типы данных. Объединения.
13. Библиотека ввода-вывода.
14. Связные списки.
15. Стеки и очереди.
16. Двоичные деревья.
17. Произвольные деревья.
18. Обход дерева.
19. Деревья поиска. Деревья поразрядного поиска.
20. Деревья двоичного поиска.
21. АВЛ-деревья.
22. B-деревья.
23. Красно-черные деревья.
24. Гибридные деревья (смесь поразрядного и двоичного поиска).
25. Декартовы деревья.
26. Хэш-таблицы.

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

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

**Семестр 2.**

***Контрольная работа 1*** (максимальное число баллов за любую задачу указано в скобках после номера задачи):

1. (7). Написать шаблонную функцию scf с произвольным набором параметров (через шаблоны с переменным числом параметров), вводящую с клавиатуры информацию по строке формата и дополнительным параметрам, переданным по ссылке (ввод очередного дополнительного параметра в строке формата обозначается знаком %, обычные символы должны соответствовать тем, которые вводит пользователь). Например, если переменная s имеет тип string, d – double и n – int, то вызов scf("name=% weight=% age=%", s, d, n) при введенной строке «name=Joe weight=85.5 age=52» должен присвоить переменной s значение "Joe" (строка вводится до первого символа, идущего после соответствующего % в строке формата, в данном случае до первого пробела), переменной d значение 85.5, переменной n значение 52.
2. (5). Написать функцию, строящую в окне терминала при помощи символов \* график другой функции, переданной в качестве параметра. В функции main вызвать эту функцию с параметром-безымянной функцией, построив таким образом график функции а) sqrt(100-x\*x) б) 7arctg(x/7).
3. (5). Написать функцию, возводящую в куб квадратную матрицу, переданную ей как параметр по ссылке, представленную как вектор из векторов.
4. (7). Написать функцию, получающую в качестве параметра функцию f типа double(double, double), которую мы будем рассматривать как функцию от точки на плоскости, и возвращающую другую функцию g от угла t в радианах, равную f(cos(t),sin(t)).

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

***Критерии оценивания*:** решения задач оцениваются по шкале от 0 (нет решения или решение имеет существенные недостатки) до количества баллов, указанного в скобках перед условием задачи (решение работоспособно, аккуратно реализовано).

***Контрольная работа 2:***

1. Написать шаблонную структуру Set с параметром «тип элемента». Она должна описывать конечные множества из элементов целого типа (элементы хранятся в связном списке шаблонного типа list; библиотечным шаблоном set пользоваться нельзя). Должны быть реализованы операции объединения, пересечения, разности множеств, построения множества из одного указанного элемента, проверка принадлежности элемента множеству, проверка включения одного множества в другое (операции сравнения).

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

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

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

1. Операции new и delete для управления динамической памятью. Автоматическое определение типа переменных и результатов функций, ключевое слово auto. Простейший ввод-вывод, манипуляторы. Цикл по элементам массива. Передача параметров по ссылке. Значения по умолчанию. Перегрузка функций. Возврат результатов по ссылке.
2. Подставляемые функции. Шаблонные функции. Функции, набор типов параметров которых может быть разным (через шаблоны с переменным числом параметров).
3. Функции, значения которых можно вычислять на этапе компиляции. Получение новых функций путем подстановки констант на место некоторых параметров имеющихся функций. Безымянные функции.
4. Тип string.
5. Шаблонные типы vector и list, итераторы.
6. Контрольная работа.
7. Переписывание контрольной работы.
8. Права доступа, причины сокрытия полей, ключевое слово class, друзья. Статические поля структур. Методы в структурах, в том числе статические, операция разрешения области видимости.
9. Конструкторы и деструкторы, большие и маленькие объекты, семантика перемещения.
10. Перегрузка операций и методов. Мимикрия, умные указатели, функторы.
11. Указатели на поля и методы. Шаблоны классов, специализация, явное инстанциирование, полиморфизм, обобщенное программирование.

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

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

**Семестр 3.**

***Контрольная работа 1:***

1. Упрощенный JSON использует для описания пользовательских данных структуру, определяемую рекурсивно как один из следующих вариантов:
   1. Целые числа.
   2. Строки в двойных кавычках (для простоты, сам символ двойной кавычки внутри строки содержаться не может).
   3. Список из структур в квадратных скобках через запятую.

Для представления такого рода структур из чисел и строк написать базовый класс Item (всякая структура будет представляться как указатель на Item) и 3 производных от него:

* Integer (его описание в текстовом виде выглядит как явно указанное число, гарантированно влезающее в int),
* String (строка в двойных кавычках) и
* List (вектор указателей на Item, каждый из которых представляет один из элементов списка; его текстовое представление выглядит как список текстовых представлений его элементов через запятую в квадратных скобках).

Написать также функцию read(имя\\_файла), считывающую из указанного текстового файла описание структуры, строящую ее как структуру из объектов классов, производных от Item, и возвращающую представляющий ее указатель на Item.

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

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

***Контрольная работа 2:***

1. (7) Написать функцию, принимающую в качестве параметров две строки — имя входного файла и префикс, и записывающую каждую строку входного файла в свой отдельный файл, имя которого образуется конкатенацией префикса, номера соответствующей строки из входного файла и расширения «.txt».
2. (7) Проверить правильность скобочной структуры в строке (параллельно двумя потоками).
3. (7) Дана функция f с двумя целыми параметрами и целым результатом. Написать функцию, принимающую в качестве параметра множество целых чисел и возвращающую замыкание этого множества относительно данной функции, т. е. наименьшее по включению множество, содержащее множество-параметр и такое, что если два числа принадлежат этому множеству, то и результат применения функции f к ним тоже принадлежит этому множеству.

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

***Критерии оценивания*:** решения задач оцениваются по шкале от 0 (нет решения или решение имеет существенные недостатки) до количества баллов, указанного в скобках перед условием задачи (решение работоспособно, аккуратно реализовано).

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

1. Синтаксис и смысл наследования. Множественное наследование. Права доступа при наследовании. Права доступа protected.
2. Автоматическое преобразование указателя на производный класс к указателю на базовый. Диспетчеризация. Виртуальные деструкторы. Чистые виртуальные методы и абстрактные классы.
3. Явное преобразование указателя на базовый класс к указателю на производный. Определение типа объекта во время выполнения программы (RTTI, run-time type identification). Двойная диспетчеризация и реализация через RTTI. Виртуальные базовые классы.
4. Исключения в C++. Раскрутка стека и принцип «захват ресурса – это инициализация» (RAII, Resource Acquisition Is Initialization). Безопасность исключений.
5. Пространства имен.
6. Контрольная работа.
7. Переписывание контрольной работы.
8. Потоковый ввод-вывод.
9. Ассоциативные контейнеры.
10. Многопоточность. OpenMP.
11. async и future. Класс thread.
12. Синхронизация.
13. Структуры данных.
14. Регулярные выражения.
15. Случайные числа.
16. Время.

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

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

**Семестр 4.**

Примерные варианты контрольных работ и вопросов к зачету:

***Контрольная работа 1:***

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

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

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

***Контрольная работа 2:***

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

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

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

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

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

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

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

**Вариант реализации 5: Прикладное программирование на языке C#**

Ниже приведены примеры заданий, которые предлагались обучающимся в прошлом. С целью актуализации заданий текущие варианты могут отличаться путем полной замены, исключения или добавления пунктов.

**Семестр 1.**

***Домашняя работа 1:***

1. Реализовать в консольном виде калькулятор, который выполняет вычисления в стандартной математической нотации над аргументом вида: 1+2\*3
2. Добавить константы e, pi.
3. Реализуйте в своем калькуляторе вычисление ln(x), sqrt(x), pow(x,y), а также скобочных выражений. Первоначально необходимо конвертировать выражение в постфиксную запись, а затем ее вычислить.

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

***Критерии оценивания*:** решения пунктов задачи оцениваются от 0 (нет решения или решение имеет существенные недостатки) до 1 балла (решение работоспособно, аккуратно реализовано, а обучающийся может полностью пояснить детали реализации).

***Домашняя работа 2:***

* + - 1. Построить генератор случайного лабиринта в виде матрицы 0 и 1 (размером N на M) с существующим путем из (x1, y1) до (x2, y2).
      2. В случайно сгенерированном лабиринте разместите героя X и определите, сколько шагов нужно сделать до границы матрицы. Если выхода нет, то выведите фразу “C’est la vie…”
      3. Напишите программу, которая строит матрицу (размером N на M) с элементами от 0 до 4 и распечатывает количество одинаковых элементов, расположенных рядом по вертикали или горизонтали.

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

***Критерии оценивания*:** решения пунктов задачи оцениваются от 0 (нет решения или решение имеет существенные недостатки) до 1 балла (решение работоспособно, аккуратно реализовано, а обучающийся может полностью пояснить детали реализации).

***Домашняя работа 3:***

* + - 1. Напишите программу для решения следующей задачи: у вас есть два кувшина - на N и M галлонов (M>N). Но ни на одном из них нет маркировки. Также имеется насос, с помощью которого можно набирать воду. Как получить ровно X галлонов в кувшине c M галлонами?
      2. Напишите программу для решения следующей задачи: три миссионера и три каннибала подошли к берегу реки, возле которого привязана лодка, вмещающая только двух человек. Каждому нужно перебраться на другой берег, чтобы продолжить путешествие. Однако, если на каком-нибудь из берегов каннибалов окажется больше, чем миссионеров, то миссионеры будут съедены. Найдите такую последовательность перевозок, чтобы все безопасно оказались на другом берегу реки. Введите параметры N – число каннибалов, M – число миссионеров, C – вместимость лодки.
      3. Реализуйте задачу о “рюкзаке”: дано *N* предметов, *ni* предмет имеет массу *wi*>0 и стоимость *pi*>0. Необходимо выбрать из этих предметов такой набор, чтобы суммарная масса не превосходила заданной величины *W* (вместимость рюкзака), а суммарная стоимость была максимальна.
      4. Реализуйте игру “Угадай животное”, построенную на бинарном дереве решений. Пользователь загадывает животное, а программа его отгадывает, задавая пользователю наводящие вопросы. В её базе хранится начальное количество названий животных и вопросов. Если же пользователь загадывает такое животное, которое программа отгадать не может (в её базе его нет), то она "просит" пользователя написать животное, которое он загадал, а также написать наводящий вопрос к нему.

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

***Критерии оценивания*:** решения пунктов задачи оцениваются от 0 (нет решения или решение имеет существенные недостатки) до 1 балла (решение работоспособно, аккуратно реализовано, а обучающийся может полностью пояснить детали реализации).

***Домашняя работа 4:***

1. Компания планирует создать систему управления для компании из сферы недвижимости. У компании есть имя, владелец, налоговый идентификатор, сотрудники и список объектов на продажу. Запись сотрудника состоит из имени, места работы и опыта. Компания продает несколько типов недвижимости: квартиры, дома, неразработанные участки и магазины. Все они характеризуются площадью, стоимостью квадратного метра и местом нахождения. Для некоторых из них есть дополнительная информация. Для квартир — это данные о количестве этажей, наличие лифта, и есть ли обстановка. Для домов — квадратные метры нежилых помещений и кол-во квадратных метров во дворе, сколько этажей у него есть и есть ли обстановка. Разработайте набор классов с отношениями между ними, которые моделируют данные для компании. Реализуйте тестовый класс, который демонстрирует работу всех классов.

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

***Критерии оценивания*:** решения пунктов задачи оцениваются от 0 (нет решения или решение имеет существенные недостатки) до 1 балла (решение работоспособно, аккуратно реализовано, а обучающийся может полностью пояснить детали реализации).

***Домашняя работа 5:***

1. Реализуйте класс односвязного списка на базе массива.
2. Реализуйте класс односвязного списка на базе ссылок.
3. Разработайте метод сортировки односвязного списка.
4. Реализуйте класс двусвязного списка.

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

***Критерии оценивания*:** решения пунктов задачи оцениваются от 0 (нет решения или решение имеет существенные недостатки) до 1 балла (решение работоспособно, аккуратно реализовано, а обучающийся может полностью пояснить детали реализации).

**Семестр 2.**

***Домашняя работа 1:***

1. Адаптировать разработанный односвязный список для применения генериков.
2. Адаптировать разработанный односвязный список так, чтобы он реализовывал интерфейсы ICollection<T>, IEnumerable<T>, а также все сопутствующие интерфейсы, чтобы этим списком можно было подменить стандартный List<T> и LinkedList<T>.
3. Реализуйте тестовую функцию, которая проверяет все публичные методы ICollection<T> и запустите ее на разработанном списке, List<T> и LinkedList<T>. Сравните быстродействие их операций и объясните результат.
4. Реализуйте интерфейс IComparable сравнения списков по длине.

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

***Критерии оценивания*:** решения пунктов задачи оцениваются от 0 (нет решения или решение имеет существенные недостатки) до 1 балла (решение работоспособно, аккуратно реализовано, а обучающийся может полностью пояснить детали реализации).

***Домашняя работа 2:***

1. Реализуйте на Windows Forms игру с моделью жизни по таким правилам: есть поле (напр. 50 на 50), оно поделено на клетки. В каждой клетке может жить инфузория. Изначально какое-то их число, (напр. 50) разбрасывается случайным образом по этому полю. Каждый ход пробегается все поле и для каждой клетки проверяются правила жизни:

Правило 1. Если клетка жива и у нее 2-3 живых соседа, то она остается живой, иначе умирает.

Правило 2. Если клетка мертва и у нее 3 живых соседа, то она становится живой, иначе остается мертвой.

После каждого хода поле перерисовывается и начинается следующий ход. Должна быть кнопка запуска и остановки игры.

1. Настало время для новых требований. Переработайте свой объектно-ориентированный дизайн для их учета.

Вначале особи рождаются и умирают, как обычно. Назовем их белыми. Ход передается от первой до последней ячейки, как и раньше. Когда появляется устойчивый паттерн, то они самоорганизуются “в колонию” и становятся черными. Цель этой колонии – двигаться, захватывать белых и объединяться с другими колониями. Колония определяет сторону, в которую идти шаг за шагом, и ее особи идут туда до тех пор, пока не наткнутся на белых, стену или черных.

a) Белая особь становится черной и присоединяется к колонии, если число ее белых соседей плюс один (она сама) меньше черных соседей.

b) Черная особь становится белой, по аналогичному правилу.

c) Черная особь больше не рождается, но умирает, как и белая.

d) При столкновении колония пропускает ход. После завершения захвата, объединения или столкновения с препятствием, определяется новое случайное направление движения.

e) При встрече двух колоний, они объединяются в одну.

f) В неопределенных случаях придумайте свои правила.

Вопрос: Могут ли черные не захватить весь мир в принципе?

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

***Критерии оценивания*:** решения пунктов задачи оцениваются от 0 (нет решения или решение имеет существенные недостатки) до 1 балла (решение работоспособно, аккуратно реализовано, а обучающийся может полностью пояснить детали реализации).

***Домашняя работа 3:***

1. В рамках гит-хаб проекта необходимо написать личный файловый менеджер. У него может быть две панели, или одна. Минимальный набор операций:

1) Навигация по дискам, файлам и папкам.

2) Копирование и перемещение файлов (через Stream) и папок.

3) Переименование, удаление.

4) Архивация файлов и папок.

В выбранной в нем папке мы отслеживаем изменения с помощью класса FileSystemWatcher.

2. Необходимо придумать и реализовать как можно больше настроек для файлового менеджера (логин, пароль, картинка фона, шрифт и все, что придумаете). Настройки сериализуем в файл в директории пользовательских документов. С помощью атрибутов OnSerialized, OnDeserialized необходимо обеспечить шифрование и восстановление логина и пароля при записи и чтении с диска. Они должны храниться в зашифрованном виде и восстанавливаться в процессе сериализации. Способ шифрования выбирается самостоятельно.

3. Реализуйте отдельный форк своего файлового менеджера, который отображает не файлы и папки, а книги по нужной тематике с сайта Amazon, полученные с помощью регулярных выражений.

В поле, где выбираете диск, сделайте пункты с известными вам языками программирования. С помощью автокомплита дайте возможность вводить в нем же собственный язык и поле для количества книг для вывода (напр. 100).

Если число обнаруженных книг меньше заданного, найдите регулярными выражениями ссылку на следующую страницу с книгами и обработайте ее так же. Необходимо в списке вывести название, автора, оценку, дату выхода (при наличии) и цену. При щелчке на книге, ее страница открывается в браузере (аналогично запуску файла в вашем менеджере).

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

***Критерии оценивания*:** решения пунктов задачи оцениваются от 0 (нет решения или решение имеет существенные недостатки) до 1 балла (решение работоспособно, аккуратно реализовано, а обучающийся может полностью пояснить детали реализации).

***Домашняя работа 4:***

1. Проведите эксперимент с измерением накладных расходов на сборку мусора. Напишите программу, которая создает множество мусорных объектов. Определите (с помощью финализатора) кол-во этих объектов, после которого начинается сборка мусора. Замерьте время выполнения программы со сборкой мусора.

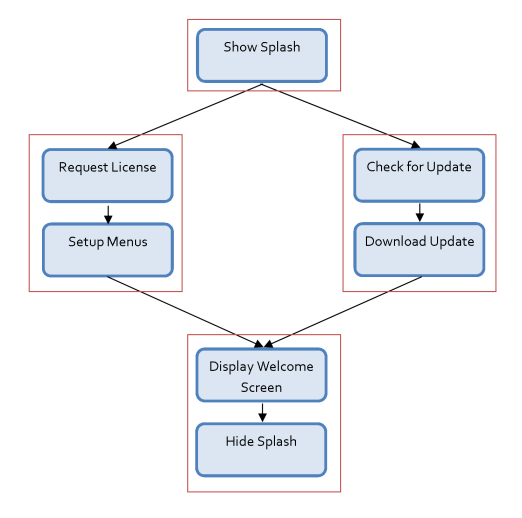
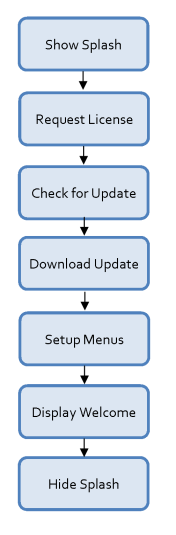
***Проверяемые компетенции:*** ОПК-2, ОПК-4, ОПК-5, ПКП-4, УКБ-3.

***Критерии оценивания*:** решения пунктов задачи оцениваются от 0 (нет решения или решение имеет существенные недостатки) до 1 балла (решение работоспособно, аккуратно реализовано, а обучающийся может полностью пояснить детали реализации).

**Семестр 3.**

***Домашняя работа 1:***

1. Реализуйте в файловом менеджере параллельную архивацию файлов в директории с помощью Parallel.For.
2. Необходимо написать консольное приложение, которое имитирует этап загрузки любого коммерческого приложения из следующих стадий:



На каждой стадии наша программа просто выводит ее название. С помощью продолжений Tasks (continuations) перепроектируйте его на параллельное выполнение. Учтите, что некоторые стадии могут выбросить исключения (напр. нет доступа в сеть, не действует лицензия и пр.) Пусть они генерируются случайным образом в коде, и в случае их появления, программа пишет соответствующее сообщение и, стадии, зависимые от ошибочной – не выполняются. В самом конце выводится сообщение, успешно ли загрузилась программа или нет.

1. Реализуйте механизм поиска названий файлов и папок в текущей директории по маске (выражение, где "\*" обозначает любое кол-во любых символов, "?" - один любой символ. Например, "MyLog\*.txt" должно обнаружить все txt-файлы, чьи имена начинаются с MyLog). Поиск происходит также во вложенных папках и в именах самих папок. Маску нужно вводить в отдельное поле и, как только в нем происходит изменение, поиск запускается заново (аналогично проводнику Windows). Обнаруженные названия файлов и папок отображаются в панели текущей папки. Используйте многопоточность, метод List<T>.FindAll(Predicate<T>) и соответствующее лямбда выражение.

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

***Критерии оценивания*:** решения пунктов задачи оцениваются от 0 (нет решения или решение имеет существенные недостатки) до 1 балла (решение работоспособно, аккуратно реализовано, а обучающийся может полностью пояснить детали реализации).

***Домашняя работа 2:***

1. Реализуйте операцию архивации асинхронным образом через async/await.
2. Реализуйте асинхронный менеджер множественной закачки файлов по введенным URL’ам.
3. Они оба должны поддерживать операции отмены и прогресса на базе CancellationToken и IProgress<T>. Для этого необходимо реализовать асинхронную функцию закачки на основе FileWebRequest.GetResponse.

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

***Критерии оценивания*:** решения пунктов задачи оцениваются от 0 (нет решения или решение имеет существенные недостатки) до 1 балла (решение работоспособно, аккуратно реализовано, а обучающийся может полностью пояснить детали реализации).

***Домашняя работа 3:***

1. С помощью паттерна (Decorator) введите дополнительный уровень с классами Entry, Folder, File, изолирующий файловый менеджер от классов System.IO: Directory, DirectoryInfo, File, FileInfo. Перенесите в методы новых классов обработку каких-нибудь ситуаций, которые можно локально обработать, напр. исключений.

2. С учетом наличия в текущей архитектуре классов Entity, Folder и File, cпроектируйте и реализуйте поиск личных данных, файлов и папок по имени также и внутри zip-архивов. Попробуйте достичь этого созданием классов ZippedFolder и ZippedFile в нашей иерархии, так чтобы код файлового менеджера был изолирован от того, с какими папками и файлами мы работаем на самом деле. Используйте для этого DotNetZip. Операции по распаковке производите в памяти, не создавая временных файлов.  
  
3. Необходимо реализовать просмотр содержимого и навигацию по zip-файлу. А также создание, удаление и копирование файлов и папок внутри zip-архива и вне его.

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

***Критерии оценивания*:** решения пунктов задачи оцениваются от 0 (нет решения или решение имеет существенные недостатки) до 1 балла (решение работоспособно, аккуратно реализовано, а обучающийся может полностью пояснить детали реализации).

***Домашняя работа 4:***

1. Переделайте с помощью Visitor реализацию подсчета MD5-хэша файла, так чтобы было возможно посчитать и хэш значение директории. Для этого производится обход ее файлов и поддиректорий и на основе содержимого всех файлов вычисляется общий MD5-хэш.

2. Реализуйте с помощью паттерна Visitor возможность шифрования выбранного файла или папки. При шифровании папки будут просто шифроваться все ее файлы (аналогично нашему сжатию). Перед процессом шифрования показывайте форму, в которой можно ввести ключ. Можно использовать любой метод шифрования. Естественно, нужен и механизм расшифровки по введенному ключу с помощью Visitor.

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

***Критерии оценивания*:** решения пунктов задачи оцениваются от 0 (нет решения или решение имеет существенные недостатки) до 1 балла (решение работоспособно, аккуратно реализовано, а обучающийся может полностью пояснить детали реализации).

***Домашняя работа 5:***

1. Разорвите зависимость между бизнес-логикой файлового менеджера и формами с помощью паттерна MVP (Passive View). Для нашей задачи лучше объединить Presenter с моделью. т.е. раньше код формы содержал бизнес-логику, а теперь выделяем из каждой формы интерфейс с методами и событиями IView, пишем класс формы, который его реализует. В его обработчиках мы просто вызываем события. Наконец, пишем Presenter формы, который подписывается на события IView, их ловит, отрабатывает и вызывает методы IView для отображения. Если необходимо, добавьте события в Presenter, на которые будет подписываться форма и отображать содержимое. Presenter и форма не должны общаться напрямую, только через события и методы IView.

2. У нашего файлового менеджера есть навигация по zip-папкам. Presenter стоит изолировать от конкретного вида папки, т.к. возможные будущие требования коснутся добавления работы с другими архивами, ftp-папки и пр. Поэтому выражений в коде вида

if x.Contains(".zip") в нем быть не должно. Вынесите их в отдельный класс – фабрику. Она принимает на вход путь и выдает соответствующий Entry. Теперь Presenter работает только с этим Entry, а какой класс его фактически реализует - знает только фабрика. Поэтому в Entry необходимо вынести все нужные операции напр.:

...  
List<Entry> GetAll();  
...  
  
Далее, нам нужно уметь создавать файлики и папки внутри папок и zip-папок. Аналогично, if-ы не нужны здесь. Самое простое решение - в Entry добавить методы CreateFolder и CreateFile и реализовать их в наследниках. Наследники будут также создаваться по их пути фабрикой.

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

***Критерии оценивания*:** решения пунктов задачи оцениваются от 0 (нет решения или решение имеет существенные недостатки) до 1 балла (решение работоспособно, аккуратно реализовано, а обучающийся может полностью пояснить детали реализации).

**Семестр 4.**

См. вариант реализации 4.

***3.1.4.3. Соответствие индикаторов достижения компетенций и контрольно-измерительных материалов***

|  |  |  |
| --- | --- | --- |
| № | Код индикатора и индикатор достижения универсальной компетенции | Контрольно-измерительные материалы (КИМ) (тестовые вопросы, контрольные задания, кейсы и пр.) |
|  | 1 | 2 |
| 1 | ОПК-1.002212.1. Пишет программы императивном и функциональном стиле. | *Вариант реализации 1:*  Все домашние, контрольные и зачётные работы.  *Вариант реализации 2:*  Все домашние работы.  *Вариант реализации 3:*  Все домашние работы.  *Вариант реализации 4:*  Все контрольные работы.  *Вариант реализации 5:*  Все домашние работы. |
| 2 | ОПК-2.002212.1. Способен оценить асимптотическую трудоёмкость алгоритма. | *Вариант реализации 1:*  Семестр 1, домашние работы 2, 3, 6-12, все контрольные и зачётная работы. Семестр 2, домашние работы 1-4, 5.2, 6.1, контрольная и зачётная работы.  *Вариант реализации 2:*  Семестр 1, домашние работы 3-11. Семестр 2, домашние работы 3-5. Семестр 3, домашние работы 1, 3.  *Вариант реализации 3:*  Домашние задачи по темам «Сортировки», «Рекурсия», «Перебор», «Бинарные деревья и поиск», «Перестановки», «Графы», «Алгоритмы обхода графов», «Стек и очередь».  *Вариант реализации 4:*  Семестр 1, контрольная работа 2. Семестр 2, контрольная работа 2. Семестр 3, контрольная работа 2.  *Вариант реализации 5:*  Семестр 1, домашние работы 3, 5. |
| 3 | ОПК-2.002212.2. Применяет основы теории графов и дискретной математики для реализации классических алгоритмов и структур данных. | *Вариант реализации 1:*  Семестр 1, домашние работы 2-11, все контрольные и зачётная работы. Семестр 2, домашние работы 1-5, контрольная и зачётная работы.  *Вариант реализации 2:*  Семестр 1, домашние работы 3-9. Семестр 2, домашние работы 3, 4. Семестр 3, домашние работы 1, 5.  *Вариант реализации 3:*  Домашние задачи по темам «Сортировки», «Рекурсия», «Двоичное представление», «Матрицы и массивы», «Перебор», «Бинарные деревья и поиск», «Перестановки», «Графы», «Алгоритмы обхода графов», «Стек и очередь».  *Вариант реализации 4:*  Семестр 1, контрольная работа 2. Семестр 2, контрольная работа 2. Семестр 3, контрольные работы 1 и 2.  *Вариант реализации 5:*  Семестр 1, домашние работы 1-3, 5. Семестр 2, домашняя работа 1. |
| 4 | ОПК-3.002212.1. Ведёт разработку программного обеспечения с использованием интегрированных сред разработки, систем контроля версий, отладчиков. | *Вариант реализации 1:*  Все домашние, контрольные и зачётные работы.  *Вариант реализации 2:*  Все домашние работы.  *Вариант реализации 3:*  Все домашние работы.  *Вариант реализации 4:*  Все контрольные работы.  *Вариант реализации 6:*  Все домашние работы. |
| 5 | ОПК-4.002212.1. Пишет комментарии к программному коду. | *Вариант реализации 1:*  Все домашние, контрольные и зачётные работы, начиная с домашней работы 2 первого семестра, требующие написания кода.  *Вариант реализации 2:*  Все домашние работы, требующие написания кода.  *Вариант реализации 3:*  Все домашние работы.  *Вариант реализации 4:*  Все контрольные работы.  *Вариант реализации 5:*  Все домашние работы. |
| 6 | ОПК-4.002212.2. Правильно использует терминологию. | *Вариант реализации 1:*  Все домашние, контрольные и зачётные работы, начиная с домашней работы 2 первого семестра.  *Вариант реализации 2:*  Все домашние работы.  *Вариант реализации 3:*  Все домашние работы, опросы и доклады.  *Вариант реализации 4:*  Все контрольные работы и списки вопросов к устному зачёту.  *Вариант реализации 5:*  Все домашние работы. |
| 7 | ОПК-5.002212.1. Может развернуть рабочее окружение: среду разработки, клиент системы контроля версий, вспомогательные инструменты. | *Вариант реализации 1:*  Все домашние работы.  *Вариант реализации 2:*  Все домашние работы.  *Вариант реализации 3:*  Все домашние работы.  *Вариант реализации 4:*  Все контрольные работы.  *Вариант реализации 5:*  Все домашние работы. |
| 8 | ПКА-1.002212.1. Разрабатывает программы для работы с математическими структурами. | *Вариант реализации 1:*  Семестр 1, домашние работы 2, 5.  Семестр 3, домашняя работа 1.  Семестр 4, домашние работы 3, 6.  *Вариант реализации 2:*  Семестр 1, домашние работы 3, 6, 8-10.  Семестр 2, домашние работы 4, 5.  Семестр 3, домашние работы 1, 5.  *Вариант реализации 3:*  Темы «Рекурсия», «Матрицы и массивы», «Перестановки», задача «Комплексные числа».  *Вариант реализации 4:*  Семестр 2, контрольные работы 1, 2.  *Вариант реализации 5:*  Семестр 1, домашняя работа 1. |
| 9 | ПКП-3.002212.1. Самостоятельно находит, изучает и представляет коллегам в удобной для восприятия форме информацию об алгоритмах, структурах данных или технологиях. | *Вариант реализации 1:*  Все доклады.  *Вариант реализации 2:*  Семестр 1, домашняя работа 5. Семестр 2, домашние работы 1, 4.4, 5.2. Семестр 3, домашние работы 1, 2, 5.3.  *Вариант реализации 3:*  Доклады по паттернам проектирования.  *Вариант реализации 4:*  Все списки вопросов к устному зачёту.  *Вариант реализации 5:*  Все домашние работы, начиная с домашней работы 3 второго семестра. |
| 10 | ПКП-4.002212.1. Пользуется интегрированной средой разработки для реализации программного обеспечения. | *Вариант реализации 1:*  Все домашние, контрольные и зачётные работы.  *Вариант реализации 2:*  Все домашние работы.  *Вариант реализации 3:*  Все домашние работы.  *Вариант реализации 4:*  Все контрольные работы.  *Вариант реализации 5:*  Все домашние работы. |
| 11 | ПКП-4.002212.2. Способен собрать программу из командной строки без использования интегрированной среды разработки. | *Вариант реализации 1:*  Семестр 2, домашняя работа 5.  *Вариант реализации 2:*  Все домашние работы.  *Вариант реализации 3:*  Все домашние работы, начиная со второго семестра.  *Вариант реализации 4:*  Все контрольные работы, начиная со второго семестра.  *Вариант реализации 5:*  Все домашние работы, начиная со второго семестра. |
| 12 | ПКП-4.002212.3. Пользуется системой контроля версий при разработке программного обеспечения. | *Вариант реализации 1:*  Все домашние, контрольные и зачётные работы, начиная с домашней работы 3 первого семестра.  *Вариант реализации 2:*  Все домашние работы.  *Вариант реализации 3:*  Все домашние работы.  *Вариант реализации 4:*  Все контрольные работы, начиная со второй контрольной работы первого семестра.  *Вариант реализации 5:*  Все домашние работы, кроме домашней работы 1 первого семестра. |
| 13 | ПКП-5.002212.1. Настраивает системы непрерывной интеграции для кода своих проектов. | *Вариант реализации 1:*  Все домашние, контрольные и зачётные работы, начиная с домашней работы 5 семестра 2.  *Вариант реализации 2:*  Все домашние работы.  *Вариант реализации 3:*  Все домашние работы, начиная со второго семестра.  *Вариант реализации 4:*  Все контрольные работы. начиная с четвёртого семестра.  *Вариант реализации 5:*  Все домашние работы, начиная со второго семестра. |
| 14 | ПКП-6.002212.1. Пишет корректные многопоточные программы с учётом модели памяти процессора. | *Вариант реализации 1:*  Все домашние, контрольные и зачётные работы, начиная с третьего семестра.  *Вариант реализации 2:*  Семестр 2, домашние работы 4, 5.  Семестр 3, домашняя работа 5.  Все домашние работы семестра 4.  *Вариант реализации 3:*  Все домашние работы семестра 4.  *Вариант реализации 4:*  Контрольная работа 2 семестра 3.  *Вариант реализации 5:*  Семестр 3, домашняя работа 2. |
| 15 | ПКП-7.002212.1. Пишет программы в структурном стиле. | *Вариант реализации 1:*  Все домашние работы семестра 1.  *Вариант реализации 2:*  Все домашние работы семестра 1.  *Вариант реализации 3:*  Темы «Сортировки», «Рекурсия», «Двоичное представление», «Матрицы и массивы», «Файлы и строки», «Перебор», «Бинарные деревья и поиск», «Перестановки», «Графы», «Алгоритмы обхода графов», «Стек и очередь», «Списки, хеш-таблицы, сбалансированные деревья».  *Вариант реализации 4:*  Все контрольные работы семестров 1 и 2.  *Вариант реализации 5:*  Домашние работы 1-3 семестра 1. |
| 16 | ПКП-7.002212.2. Пишет программы в объектно-ориентированном стиле. | *Вариант реализации 1:*  Все домашние работы семестров 2 и 3.  *Вариант реализации 2:*  Семестр 1, домашние работы 7, 9.  Семестр 2, домашние работы 1, 2.  *Вариант реализации 3:*  Задачи темы «Объектно-ориентированное программирование», задачи варианта 1 семестра 4.  *Вариант реализации 4:*  Семестр 2, контрольная работа 2.  Семестр 3, контрольная работа 1.  Семестр 4, контрольные работы 1 и 2.  *Вариант реализации 5:*  Домашние работы 4 и 5 семестра 1, |
| 17 | ПКП-7.002212.3. Пишет программы в функциональном стиле. | *Вариант реализации 1:*  Все домашние работы семестра 4.  *Вариант реализации 2:*  Все домашние работы.  *Вариант реализации 3:*  Темы «Рекурсия», «Перебор».  *Вариант реализации 4:*  Семестр 2, контрольная работа 1.  *Вариант реализации 5:*  Семестр 3, домашняя работа 1. |
| 18 | ПКП-7.002212.4. Способен описать на UML структуру классов несложной системы. | *Вариант реализации 1:*  Семестр 2, домашняя работа 9.  *Вариант реализации 2:*  Семестр 2, домашняя работа 1.  *Вариант реализации 3:*  Тема «Объектно-ориентированное программирование», доклады по паттернам проектирования.  *Вариант реализации 4:*  Семестр 3, вопросы к зачёту.  *Вариант реализации 5:*  Семестр 2, домашняя работа 4.  Семестр 3, домашние работы 3-5. |
| 19 | ПКП-8.002212.1. Способен реализовать на каком-либо языке программирования математический алгоритм по приведённому описанию алгоритма. | *Вариант реализации 1:*  Семестр 1, домашние работы 2, 3, 5, 7.  Семестр 2, домашние работы 2, 4.  Семестр 3, домашняя работа 1.  Семестр 4, домашние работы 1–3, 6, контрольные работы.  *Вариант реализации 2:*  Семестр 1, домашние работы 3, 6, 8, 9.  Семестр 2, домашние работы 4, 5.  Семестр 3, домашняя работа 5.  Семестр 4, домашние работы 3, 4.  *Вариант реализации 3:*  Темы «Рекурсия», «Матрицы и массивы», «Перестановки», «Объектно-ориентированное программирование», задачи четвёртого семестра.  *Вариант реализации 4:*  Семестр 2, контрольные работы 1 и 2.  Семестр 3, контрольная работа 2.  *Вариант реализации 5:*  Семестр 1, домашняя работа 1. |
| 20 | УКБ-3.1. Находит и использует различные источники информации. | *Вариант реализации 1:*  Все домашние и контрольные работы, доклады.  *Вариант реализации 2:*  Все домашние работы.  *Вариант реализации 3:*  Все домашние работы, опросы и доклады.  *Вариант реализации 4:*  Все контрольные работы и списки вопросов к устному зачёту.  *Вариант реализации 5:*  Все домашние работы. |
| 21 | УКБ-3.2. Точно определяет тип и форму необходимой информации. | *Вариант реализации 1:*  Все домашние, контрольные и зачётные работы, доклады.  *Вариант реализации 2:*  Все домашние работы.  *Вариант реализации 3:*  Все домашние работы, опросы и доклады.  *Вариант реализации 4:*  Все контрольные работы и списки вопросов к устному зачёту.  *Вариант реализации 5:*  Все домашние работы. |
| 22 | УКБ-3.3. Получает информацию и сохраняет ее в удобном для работы формате. | *Вариант реализации 1:*  Все домашние, контрольные и зачётные работы, доклады.  *Вариант реализации 2:*  Все домашние работы.  *Вариант реализации 3:*  Все домашние работы, опросы и доклады.  *Вариант реализации 4:*  Все контрольные работы и списки вопросов к устному зачёту.  *Вариант реализации 5:*  Все домашние работы. |
| 23 | УКБ-3.4. Проверяет достоверность собранной информации. | *Вариант реализации 1:*  Все домашние, контрольные и зачётные работы, доклады.  *Вариант реализации 2:*  Все домашние работы.  *Вариант реализации 3:*  Все домашние работы, опросы и доклады.  *Вариант реализации 4:*  Все контрольные работы и списки вопросов к устному зачёту.  *Вариант реализации 5:*  Все домашние работы. |

**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**

Учебные аудитории для проведения учебных занятий, оснащенные стандартным оборудованием, используемым для обучения в СПбГУ в соответствии с требованиями материально-технического обеспечения.

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

Стандартное оборудование, используемое для обучения в СПбГУ.

MS Windows, MS Office, Mozilla FireFox, Google Chrome, Acrobat Reader DC, WinZip, Антивирус Касперского.

**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**

**3.4.1 Список обязательной литературы**

Не требуется

**3.4.2 Список дополнительной литературы**

Не требуется

**3.4.3 Перечень иных информационных источников**

* Сайт Научной библиотеки им. М. Горького СПбГУ: <http://www.library.spbu.ru/>
* Электронный каталог Научной библиотеки им. М. Горького СПбГУ:

<http://www.library.spbu.ru/cgi-bin/irbis64r/cgiirbis_64.exe?C21COM=F&I21DBN=IBIS&P21DBN=IBIS>

* Перечень электронных ресурсов, находящихся в доступе СПбГУ:

<http://cufts.library.spbu.ru/CRDB/SPBGU/>

* Перечень ЭБС, на платформах которых представлены российские учебники, находящиеся в доступе СПбГУ: <http://cufts.library.spbu.ru/CRDB/SPBGU/browse?name=rures&resource_type=8>

**Раздел 4. Разработчики программы**

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

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

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

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

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

Григорьева Анастасия Викторовна, кандидат технических наук, доцент кафедры информатики, [a.v.grigorieva@spbu.ru](mailto:a.v.grigorieva@spbu.ru)

При использовании материалов Брыксина Тимофея Александровича, кандидата технических наук