ЦЕНТРАЛЬНЫЙ УНИВЕРСИТЕТ

**УТВЕРЖДЕНА**

Решением Ученого совета
АНО ВО «Центральный университет»
«07» марта 2024 г.
Протокол №1

**Рабочая программа дисциплины (модуля)**
**«Дизайн компиляторов»**

**Направление подготовки:** *02.03.01 Математика и компьютерные науки*

**Направленность (профиль) подготовки:** *Программа двух дипломов НИУ ВШЭ и ЦУ «Прикладная математика и информатика»*

**Квалификация (степень) выпускника:** *бакалавр*

**Форма обучения:** *очная*

**Срок освоения программы:** *4 года*

**Год набора:** *2024*

**Москва**
**2024**

Электронный документ

**Содержание**

## 1. Краткая характеристика дисциплины (модуля)

Рабочая программа дисциплины (модуля) «Дизайн компиляторов» составлена в соответствии с федеральным государственным образовательным стандартом высшего образования – бакалавриат по специальности 02.03.01 Математика и компьютерные науки, профиль «Программа двух дипломов НИУ ВШЭ и ЦУ «Прикладная математика и информатика», утвержденный приказом Министерства науки и высшего образования Российской Федерации № 807 от 23.08.2017 года.

Изучение дисциплины (модуля) «Дизайн компиляторов» важно для понимания принципов преобразования высокоуровневого кода в эффективный машинный код, что способствует оптимизации работы программ и систем. Это знание позволяет создавать новые языки программирования и инструменты разработки, повышая производительность и надежность программного обеспечения.

**Место дисциплины (модуля) в структуре образовательной программы**

Настоящая дисциплина (модуль) включена в учебный план по программе подготовки бакалавриата по направлению 02.03.01 Математика и компьютерные науки, профиль «Программа двух дипломов НИУ ВШЭ и ЦУ «Прикладная математика и информатика» и входит в вариативную часть Блока 1, формируемую участниками образовательных отношений.

Дисциплина (модуль) является выборной и доступна для изучения на 2, 3 или 4 курсе в 4, 5, 6, 7 или 8 семестрах на выбор.

**Цель изучения дисциплины (модуля):** освоение методов и принципов разработки компиляторов для преобразования и оптимизации программного кода.

**Задачи изучения дисциплины (модуля)** направлены на формирование у студентов следующий знаний, умений и навыков:

— знание принципов компиляции кода и устройства компиляторов;
— знание техник оптимизации сгенерированного кода;
— знание алгоритмов вывода типов для динамических языков программирования;
— умение описывать формальную грамматику языка;
— умение использовать современные инструменты для генерации кода компилятора;
— умение анализировать внутреннее представление программ;
— умение выполнять преобразования внутреннего представления программ;
— навык реализации статического анализатора кода;
— навык реализации автоматического перевода кода из одного языка программирования в другой.

## 2. Перечень планируемых результатов обучения

Компетенции, формируемые в результате освоения дисциплины (модуля) при проведении учебных занятий в форме контактной работы обучающихся с педагогическими работниками Университета и в форме самостоятельной работы обучающихся:

| Компетенция | Содержание компетенции | Индикатор компетенции | Перечень планируемых результатов обучения по дисциплине (модулю) |
|---|---|---|---|
| УК-1. | Способен осуществлять поиск, критический анализ и синтез информации, применять системный подход для решения поставленных задач | УК-1.1. | Знает методы поиска и анализа информации в области искусственного интеллекта, основные принципы критической оценки источников информации и их релевантности |
| | | УК-1.2. | Умеет критически оценивать источники информации и синтезировать данные из различных источников для решения задач, применять системный подход к анализу и решению комплексных проблем |
| | | УК-1.3. | Имеет практический опыт работы с современными инструментами и технологиями для обработки информации, формулировании и структурировании задач на основе полученной информации |
| ОПК-1. | Способен консультировать и использовать фундаментальные знания в области математического анализа, комплексного и функционального анализа алгебры, аналитической геометрии, дифференциальной геометрии и топологии, дифференциальных уравнений, дискретной математики и математической логики, теории вероятностей, математической статистики и случайных процессов, численных методов, теоретической механики в профессиональной деятельности | ОПК-1.1. | Знает основные концепции и теории в области математического анализа и смежных дисциплин; методы и подходы, используемые в различных областях математики |
| | | ОПК-1.2. | Умеет применять математические методы для решения профессиональных задач |
| | | ОПК-1.3. | Имеет практический опыт разработки и реализация математических моделей в профессиональной деятельности |
| ОПК-6 | Способен разрабатывать алгоритмы и компьютерные программы, пригодные | ОПК-6.1. | Знает алгоритмы разработки, компьютерные программы, а также алгоритмы вычислительной математики в области |

| | | | |
|---|---|---|---|
| | для практического применения | | искусственного интеллекта |
| | | ОПК-6.2. | Умеет разрабатывать математические программные продукты и комплексы с использованием современных технологий программирования в области искусственного интеллекта |
| | | ОПК-6.3. | Имеет практический опыт разработки интеллектуальных информационных систем для визуализации результатов исследований в области искусственного интеллекта |
| ПК-1. | Способен формулировать задачи с математической точностью, обосновывать утверждения строго и анализировать полученные результаты в области математики и компьютерных наук | ПК-1.1. | Знает методы и подходы к формулированию задач, а также основные принципы математического доказательства и анализа результатов |
| | | ПК-1.2. | Умеет корректно ставить и формулировать математические задачи, применять строгие методы доказательства и анализировать полученные результаты |
| | | ПК-1.3. | Имеет опыт работы с задачами в области математики и компьютерных наук, включая применение математических методов для решения практических задач |

## 3. Тематический план

| №п/п | Наименование раздела дисциплины (модуля) | Трудоемкость, академические часы | | | | ТКУ (текущий контроль успеваемости) |
|---|---|---|---|---|---|---|
| | | *Очная форма* | | | | |
| | | Контактная работа | | Контроль | Самостоятельная работа | |
| | | Лекции | Семинары (практические занятия) | | | |
| 1 | Формальные грамматики и описание структуры языков | 5 | 5 | | 22 | Кейс Коллоквиум |
| 2 | Инструменты разбора грамматик и абстрактные синтаксические деревья (AST) | 6 | 6 | | 23 | Кейс Коллоквиум |
| 3 | Построение и анализ потоков данных (DFG) | 3 | 3 | | 22 | Кейс Коллоквиум |
| 4 | Статическая и динамическая типизация, алгоритмы вывода типов | 5 | 5 | | 23 | Кейс Коллоквиум |
| 5 | Преобразования AST | 4 | 4 | | 22 | Кейс Коллоквиум |
| 6 | Генерация кода из AST | 3 | 3 | | 22 | Кейс Коллоквиум |
| | ***Зачет с оценкой*** | | | 4 | | Проект |
| | ***Итого:*** | ***26*** | ***26*** | ***4*** | ***134*** | |
| ***Объем дисциплины (в ак. ч.)*** | | ***190*** | | | | |
| ***Объем дисциплины (в зач. ед.)*** | | ***5*** | | | | |

## 4. Содержание дисциплины (модуля)

| №п/п | Наименование раздела дисциплины (модуля) | Содержание дисциплины (модуля) по темам |
|---|---|---|
| 1 | Формальные грамматики и описание структуры языков | Введение в компиляцию; регулярные и КС-грамматики. BNF / EBNF, лексическая структура, конфликтные грамматики. LL- vs LR-подходы; разбор простых грамматик вручную |
| 2 | Инструменты разбора грамматик и абстрактные синтаксические деревья (AST) | Лексер/парсер-генераторы, построение AST. Обработка ошибок парсера, визуализация AST |
| 3 | Построение и анализ потоков данных (DFG) | Промежуточные представления, CFG, базовые DFG. SSA-форма, анализ доступности и живости. Оптимизации по данным: константная свёртка, устранение мёртвого кода |
| 4 | Статическая и динамическая типизация, алгоритмы вывода типов | Статические типовые системы, вывода Hindley–Milner. Динамическая / гибридная типизация, проверки во время исполнения. Практика вывода типов и аннотаций в IR |
| 5 | Преобразования AST | Десугаринг, рефакторинг и упрощения высокого уровня. Инлайнинг, частичная оценка, подготовка к кодогенерации |
| 6 | Генерация кода из AST | Выбор инструкций, регистровое распределение. Формирование объектных файлов, линковка и запуск |

## 5. Учебно-методическое обеспечение

Университет располагает полным набором лицензионного и свободно распространяемого программного обеспечения, включая продукты отечественного производства.

Каждый студент в течение всего периода обучения получает индивидуальный неограниченный доступ к электронно-библиотечной системе и электронной информационно-образовательной среде университета. Эти системы предоставляют возможность доступа к ресурсам из любой точки, где есть подключение к сети Интернет, как на территории университета, так и за его пределами.

Студентам обеспечен удаленный доступ к современным профессиональным базам данных и информационным справочным системам.

***Основная литература:***

1. Наке, К. LLVM 17: инфраструктура для разработки компиляторов. Руководство для начинающих по LLVM – системе разработки компиляторов и сопутствующих библиотек на C++ : практическое руководство / К. Наке, Э. Кван ; пер. с англ. А. А. Слинкина. – Москва : ДМК Пресс, 2024. - 370 с. – ISBN 978-5-93700-303-4. - Текст : электронный. - URL: https://znanium.ru/catalog/product/2205082.

2. Лопес, Б. LLVM: инфраструктура для разработки компиляторов. Знакомство с основами LLVM и использование базовых библиотек для создания продвинутых инструментов : практическое руководство / Б. Лопес, Р. Аулер ; пер. с англ. А. Н. Киселёва. — 2-е изд. - Москва : ДМК Пресс, 2023. - 343 с. - ISBN 978-5-89818-603-6. - Текст : электронный. - URL: https://znanium.com/catalog/product/2108482.

3. Вирт, Н. Построение компиляторов : практическое пособие / Н. Вирт ; пер. с англ. Е. В. Борисова, Л. Н. Чернышова. - 2-е изд. - Москва : ДМК Пресс, 2023. - 193 с. - ISBN 978-5-89818-573-2. - Текст : электронный. - URL: https://znanium.com/catalog/product/2107934.

4. Клинтон, Л. Дж. Создай свой собственный язык программирования. Руководство программиста по разработке компиляторов, интерпретаторов и доменно-ориентированных языков для решения современных вычислительных задач : практическое руководство / Л. Дж. Клинтон ; пер. с англ. С. В. Минца. - Москва : ДМК Пресс, 2023. - 408 с. - ISBN 978-5-93700-140-5. - Текст : электронный. - URL: https://znanium.com/catalog/product/2109499.

***Дополнительная литература:***

5. Новожилов О. П. Архитектура ЭВМ и вычислительных систем : учебник для вузов / О. П. Новожилов. — 2-е изд., испр. и доп. — Москва : Издательство Юрайт, 2025. — 505 с. — (Высшее образование). — ISBN 978-5-534-20365-3. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/568920.

6. Толстобров А. П. Архитектура ЭВМ : учебник для вузов / А. П. Толстобров. — 3-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2025. — 162 с. — (Высшее образование). — ISBN 978-5-534-16839-6. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/566711.

7. Малявко А. А. Формальные языки и компиляторы : учебник для вузов / А. А. Малявко. — Москва : Издательство Юрайт, 2025. — 429 с. — (Высшее образование). — ISBN 978-5-534-04288-7. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/562822.

## 6. Материально-техническое обеспечение

Университет располагает материально-технической базой, соответствующей действующим противопожарным правилам и нормам и обеспечивающей проведение всех

видов дисциплинарной и междисциплинарной подготовки, практической и научно-исследовательской работ обучающихся, предусмотренных учебным планом.

Помещения, которые представляют собой учебные аудитории для проведения занятий лекционного типа, занятий семинарского (практического) типа, групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации, а также помещения для самостоятельной работы и помещения для хранения и профилактического обслуживания учебного оборудования. Помещения укомплектованы специализированной мебелью и техническими средствами обучения, служащими для представления учебной информации большой аудитории.

Изучение дисциплины (модуля) обеспечивается в учебных аудиториях, оснащенных:

— столами и стульями;
— компьютерной техникой;
— механическими калькуляторами;
— специализированным оборудованием, включая демонстрационное оборудование.

Помещения для самостоятельной работы обучающихся, в том числе приспособленные для использования инвалидами и лицами с ограниченными возможностями здоровья, оснащены компьютерной техникой с возможностью подключения к сети «Интернет» и обеспечением доступа в электронную информационно-образовательную среду Университета.

Обучающимся предоставляется доступ (в том числе удаленный) к ресурсам информационно-телекоммуникационной сети «Интернет», электронным ресурсам (в том числе электронным библиотечным системам, современным профессиональным базам данных и информационным справочным системам):

| № | Наименование портала (издания, курса, документа) | Ссылка |
|---|---|---|
| 1. | Научная электронная библиотека elibrary.ru библиотека | https://elibrary.ru/defaultx.asp |
| 2. | База данных для IT-специалистов | https://habr.com |
| 3. | База данных ScienceDirect | https://www.sciencedirect.com |
| 4. | Официальный сайт Министерства науки и высшего образования Российской Федерации | https://minobrnauki.gov.ru/ |
| 5. | Федеральный портал «Российское образование" | https://www.edu.ru/ |
| 6. | Информационная система "Единое окно доступа к образовательным ресурсам" | http://window.edu.ru/ |
| 7. | Единая коллекция цифровых образовательных ресурсов | http://school-collection.edu.ru/ |
| 8. | Федеральный центр информационно - образовательных ресурсов | http://fcior.edu.ru/ |

Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине (модулю), в том числе комплект лицензионного программного обеспечения, современные профессиональные базы данных и информационные справочные системы:

| Наименование ПО | Производство | Лицензионное / свободно распространяемое |
|---|---|---|
| **Операционные системы:** | | |
| Microsoft Imagine (Windows Client, Server) | зарубежное | лицензионное |
| **Браузеры:** | | |
| Яндекс.Браузер | отечественное | свободно распространяемое |
| Google Chrome | зарубежное | свободно распространяемое |
| **Офисные приложения:** | | |

| | | |
|---|---|---|
| Microsoft Imagine (Visio, OneNote) | зарубежное | лицензионное |
| TeXstudio | зарубежное | свободно распространяемое |
| Adobe Acrobat Reader | зарубежное | свободно распространяемое |
| **Программное обеспечение для планирования и учета времени:** | | |
| Toggle app | зарубежное | свободно распространяемое |
| **Системы управления проектами:** | | |
| Microsoft Imagine (Project) | зарубежное | лицензионное |
| **Системы управления базами данных:** | | |
| Microsoft Imagine (SQL Server) | зарубежное | лицензионное |
| **Системы резервного копирования (backup):** | | |
| Acronis Backup Advanced for HyperV | зарубежное | лицензионное |
| **Справочно-правовые системы:** | | |
| КонсультантПлюс: справочно-правовая система | отечественное | лицензионное |
| **Средства антивирусной защиты:** | | |
| Kaspersky Endpoint Security для бизнеса Стандартный Russian Edition | отечественное | лицензионное |
| **Среды разработки:** | | |
| Visual Studio Code | зарубежное | свободно распространяемое |
| Bash (Unix shell) | зарубежное | свободно распространяемое |
| Anaconda | зарубежное | свободно распространяемое |
| Robotic Operating System | зарубежное | свободно распространяемое |
| CopelliaSim | зарубежное | свободно распространяемое |
| Google Colaboratory | зарубежное | свободно распространяемое |
| **Пакеты программных средств и библиотек:** | | |
| AutoPsy | зарубежное | свободно распространяемое |
| Interactive Disassembler (IDA) | зарубежное | свободно распространяемое |
| **Системы управления библиографической информацией:** | | |
| Zotero | зарубежное | свободно распространяемое |
| **Сервисы и службы:** | | |
| Bind | зарубежное | свободно распространяемое |
| Docker | зарубежное | свободно распространяемое |

## 7. Методические и оценочные материалы

### Методические указания для обучающихся по освоению дисциплины (модуля)

В процессе изучения дисциплины (модуля) «Дизайн компиляторов» в рамках текущего контроля успеваемости используются такие виды учебной работы, как лекции, семинары, коллоквиумы, кейсы, проект, а также различные виды самостоятельной работы обучающихся по заданию преподавателя, направленные на развитие навыков профессиональной лексики, закрепление практических профессиональных компетенций, поощрение инициатив.

*Лекция* – систематическое, последовательное, монологическое изложение преподавателем учебного материала, как правило, теоретического характера.

В процессе лекций рекомендуется вести конспект лекций: кратко и схематично фиксировать основные идеи, выводы и обобщения лекции; выделять важные мысли, ключевые слова и термины. Необходимо отметить вопросы или материалы, которые вызывают затруднения, и попытаться найти ответы в рекомендованной литературе. Если разобраться в материале не удается, следует сформулировать вопрос и задать его преподавателю на консультации или во время семинарского (практического) занятия.

*Семинар* — это форма учебной деятельности, проводимая в учебном заведении под

руководством преподавателя, где студенты активно участвуют в обсуждениях, практических заданиях и других формах взаимодействия.

Для успешной подготовки к семинару рекомендуется заранее ознакомиться с темой занятия и основными материалами, чтобы иметь возможность активно участвовать в обсуждении. Также полезно подготовить вопросы и идеи для обсуждения, что поможет глубже понять материал и продемонстрировать заинтересованность.

*Кейс* – практическая работа студентов над реальными или смоделированными задачами, что позволяет студенту применять теоретические знания на практике.

Студент самостоятельно разрабатывает стратегию решения поставленной задачи, что способствует развитию навыков критического мышления и самостоятельного принятия решений. Такой подход помогает подготовить будущих специалистов к реальным вызовам в их профессиональной деятельности.

*Коллоквиум* – устные ответы на вопросы, список которых известен студенту заранее.

В процессе подготовки к коллоквиуму необходимо проанализировать учебные материалы, ознакомившись с лекциями, учебниками и дополнительными источниками, акцентируя внимание на ключевых темах. Рекомендуется создать структурированные конспекты, выделяя основные идеи, термины и формулы.

*Проект* – исследовательская работа по курсу и презентация результатов.

Для успешной подготовки к проекту: четко определите цели и задачи проекта, распределите роли и обязанности между участниками, а также установите сроки выполнения каждой части работы. Регулярно проводите встречи для обсуждения прогресса и решения возникающих вопросов.

*Самостоятельная работа* – работа студентов, направленная на углубленное изучение отдельных тем и вопросов учебной дисциплины (модуля).

В процессе самостоятельной работы студенты взаимодействуют с рекомендованными материалами при минимальном участии преподавателя. Задачи студента включают работу с конспектами лекций (обработка текста), повторное изучение учебных материалов, планов и тезисов ответов, изучение дополнительных тем, выполнение учебно-исследовательских заданий и другое.

## Система оценивания результатов обучения по дисциплине (модулю)

**Критерии получения уровня и оценивания сформированности компетенций по дисциплине (модулю) «Дизайн компиляторов»**

Оценивание уровня учебных достижений, обучающихся по дисциплине (модулю), осуществляется в виде текущего контроля успеваемости и промежуточной аттестации.

**Промежуточная аттестация** по дисциплине (модулю) осуществляется в форме ***зачета с оценкой***, при этом проводится оценка компетенций, сформированных по дисциплине.

Для оценивания текущего контроля успеваемости и промежуточной аттестации используется десятибалльная шкала оценивания, которая соотносится с традиционной пятибалльной шкалой следующим образом:

| Десятибалльная оценка | Пятибалльная оценка | Оценка за зачет | Общая характеристика результата обучения по дисциплине (модулю) |
|---|---|---|---|
| 10 | Отлично | Зачтено | Студент полностью владеет знаниями, изложенными в рабочей программе, и |
| 9 | Отлично | Зачтено | |

| Десятибалльная оценка | Пятибалльная оценка | Оценка за зачет | Общая характеристика результата обучения по дисциплине (модулю) |
|---|---|---|---|
| 8 | Отлично | Зачтено | глубоко осмысляет дисциплину. Он самостоятельно и логически последовательно отвечает на все вопросы, акцентируя внимание на наиболее важном. Умеет анализировать, сравнивать, классифицировать, обобщать, конкретизировать и систематизировать изученный материал, выделяя ключевые моменты и устанавливая причинно-следственные связи. Четко формулирует ответы, уверенно интерпретирует результаты анализов и других исследований, а также решает сложные задачи. Студент хорошо знаком с методами исследования, необходимыми для практической деятельности, и умеет связывать теоретические аспекты дисциплины (модуля) с практическими задачами. |
| 7 | Хорошо | Зачтено | Студент обладает знаниями предмета почти в полном объеме рабочей программы и самостоятельно, логически последовательно и всесторонне отвечает на все вопросы, акцентируя внимание на наиболее значимых моментах. Он умеет анализировать, сравнивать, классифицировать, обобщать, конкретизировать и систематизировать изученный материал, выделяя его ключевые аспекты и устанавливая причинно-следственные связи. Формулирует свои ответы, уверенно интерпретирует результаты анализов и других исследований, а также решает сложные ситуационные задачи. Студент хорошо знаком с методами исследования, необходимыми для практической деятельности, и умеет связывать теоретические аспекты предмета с практическими задачами. |
| 6 | Хорошо | Зачтено | |
| 5 | Удовлетворительно | Зачтено | Студент обладает базовыми знаниями по дисциплине (модулю), но испытывает трудности при самостоятельных ответах и использует неточные формулировки. В ходе ответов он допускает ошибки, касающиеся сути вопросов. Студент способен решать только самые простые |
| 4 | Удовлетворительно | Зачтено | |

| Десятибалльная оценка | Пятибалльная оценка | Оценка за зачет | Общая характеристика результата обучения по дисциплине (модулю) |
|---|---|---|---|
| | | | задачи и владеет лишь минимальным набором методов исследования. |
| 3 | Не сдан | Не зачтено | Студент не овладел обязательным минимумом знаний по предмету и не может ответить на вопросы, даже если преподаватель задает дополнительные наводящие вопросы. |
| 2 | Не сдан | Не зачтено | |
| 1 | Не сдан | Не зачтено | |

Дисциплина (модуль) «Дизайн компиляторов» оценивается следующим образом:

| Активность | Вес | Описание |
|---|---|---|
| Кейс | 40% | Практическая работа студентов над реальными или смоделированными задачами, что позволяет студенту применять теоретические знания на практике |
| Коллоквиумы | 30% | Устные ответы на вопросы, список которых известен студенту заранее |
| Проект | 30% | Защита итогового проекта |

**Формула расчёта итоговой оценки по дисциплине (модулю) «Дизайн компиляторов»:** «0,4 × среднее за кейсы + 0,3 × среднее за коллоквиумы + 0,3 × проект».

## Текущий контроль успеваемости обучающихся по дисциплине (модулю)

### Примерные задания для кейсов

### Кейс 1: Разработка грамматики для арифметического выражения

**Описание кейса:** Вы разрабатываете простую грамматику для подмножества арифметических выражений, поддерживающего операции +, -, *, /, скобки и переменные. Грамматика должна быть безконфликтной и подходящей для LL(1)-разбора. Анализируйте потенциальные конфликты (сдвиги/сводки) и преобразуйте в EBNF.

**Задание 1: Анализ грамматики**
Определите контекстно-свободную грамматику (КС-грамматику) в BNF для выражений вида "a + b * (c - d)", где a–d — переменные или числа. Вычислите FIRST и FOLLOW множества для нетерминалов. Выявите возможные конфликты (левосторонняя рекурсия или неоднозначность) и предложите устранение (например, факторизация).

**Задание 2: Ручной разбор**
Ручным способом (таблица разбора или рекурсивный спуск) разберите выражение "2 * (3 + 4) - 1" по вашей грамматике. Постройте стек вызовов для LL(1)-парсера и обработайте случай с ошибкой (например, "2 + * 3"). Используйте Python для симуляции таблицы PREDICT.

**Задание 3: Отчет и сравнение**
Составьте отчет (1–2 страницы): опишите грамматику в EBNF, результаты анализа FIRST/FOLLOW, таблицу разбора и обработку ошибок. Сравните LL(1) и LR(0) подходы для этой грамматики, обсудите преимущества для компилятора (например, скорость vs. мощность). Прикрепите код симуляции.

## Кейс 2: Построение парсера для простого языка конфигурации

**Описание кейса:** Создайте лексер и парсер для мини-языка конфигурации (типа INI-файла), поддерживающего секции [section], ключи key=value и комментарии #. Используйте генераторы (ANTLR или Flex/Bison) для генерации AST, визуализируйте дерево и обработайте ошибки (например, незакрытые скобки).

**Задание 1: Анализ и генерация грамматики**
Разработайте грамматику в EBNF для языка: пример — "[database]\nhost=localhost\nport=5432\n#comment". Определите токены (ID, STRING, EQUALS и т.д.) и правила. Сгенерируйте лексер/парсер с помощью ANTLR (или JFlex/CUP для Java). Выявите потенциальные ошибки парсинга (например, дубликаты секций).

**Задание 2: Построение и визуализация AST**
Реализуйте парсер на Python/Java, интегрируя ANTLR. Для входного файла конфигурации постройте AST (узлы: SectionNode, KeyValueNode). Визуализируйте дерево с помощью Graphviz или print-структуры. Обработайте ошибки: добавьте recovery (пропуск неверных строк) и протестируйте на 3 примерах (валидный, с ошибкой, с комментариями).

**Задание 3: Отчет и анализ**
Подготовьте отчет (1–2 страницы): включите грамматику, сгенерированный код (фрагменты), AST для примера и визуализацию. Проанализируйте обработку ошибок (точность восстановления) и производительность (время парсинга для 100 строк). Обсудите, как AST упрощает дальнейшую обработку (например, в компиляторе).

## Примерные вопросы для коллоквиума

1. **Теория:** Определите разницу между регулярными грамматиками и контекстно-свободными (КС-грамматиками). Приведите пример регулярной грамматики для языка идентификаторов (буква, за которой следуют буквы/цифры).
2. **Практика:** Запишите КС-грамматику в BNF для арифметических выражений с операциями +, *, скобками и переменными. Преобразуйте её в EBNF, устранив левостороннюю рекурсию.
3. **Анализ:** Для грамматики S → A | B, A → aA | ε, B → bB | ε вычислите множества FIRST и FOLLOW. Объясните, почему она подходит для LL(1)-разбора.
4. **Сравнение:** Опишите ключевые различия между LL- и LR-подходами к разбору. Приведите пример грамматики, где LL(1) терпит неудачу, но LR(1) succeeds (например, неоднозначные выражения).
5. **Теория:** Что такое лексер-генератор (например, Flex) и парсер-генератор (например, Bison)? Опишите этапы их работы на примере простого токенизатора для ключевых слов и чисел.
6. **Практика:** Для входной строки "if (x > 0) { y = 1; }" постройте AST вручную: укажите узлы (IfNode, BinaryOpNode и т.д.) и их связи. Объясните, как AST отличается от конкретного синтаксического дерева (CST).
7. **Обработка ошибок:** Как парсер ANTLR обрабатывает синтаксические ошибки (например, пропуск токена)? Приведите пример recovery-стратегии для незакрытой скобки в выражении.
8. **Визуализация:** Опишите, как визуализировать AST с помощью Graphviz. Нарисуйте схематично AST для цикла "for i in 1..10 { print(i); }".
9. **Теория:** Что такое CFG (Control Flow Graph) и DFG (Data Flow Graph)? Объясните, как CFG используется для анализа потоков в процедуре с ветвлениями и циклами.

10. **Практика:** Для кода на псевдокоде:

1: a = 5;
2: if (a > 0) b = a + 1; else b = 0;
3: c = b * 2;

Постройте CFG и DFG. Вычислите доступные определения (reaching definitions) для переменной b в узле 3.

11. **Анализ:** Опишите SSA-форму (Static Single Assignment). Преобразуйте приведенный выше код в SSA, введя φ-функции для слияния путей.

12. **Оптимизации:** Как анализ живости переменных (liveness analysis) помогает в устранении мертвого кода? Приведите пример: если переменная x присваивается, но не используется после, покажите оптимизацию.

13. **Теория:** Сравните статическую и динамическую типизацию. Приведите преимущества статической (например, в Java) и динамической (например, в Python) типизации для компиляторов.

14. **Практика:** Используя алгоритм Hindley–Milner, выведите тип для λ-выражения: let id = λx. x in id (id true). Укажите полиморфный тип (forall α. α → α).

15. **Гибридная типизация:** Объясните, как гибридная типизация (статическая + динамическая) работает в языках вроде TypeScript. Приведите пример аннотации типа и runtime-проверки.

16. **IR-практика:** В промежуточном представлении (IR) для "let y = x + 1" добавьте типовые аннотации. Что происходит при выводе типов, если x — int, a + — перегрузка?

17. **Теория:** Что такое десугаринг (desugaring)? Объясните, как он упрощает AST, превращая сахар (syntactic sugar) в базовые конструкции (например, for-loop в while).

18. **Практика:** Для AST выражения "if (cond) { a = b + c; } else { a = 0; }" выполните инлайнинг, если b и c — константы. Покажите преобразованный AST.

19. **Оптимизации:** Опишите частичную оценку (partial evaluation). Примените её к коду "f(x) = if (true) x*2 else x/2; call f(5)", вычислив статически.

20. **Теория и практика:** Опишите регистровое распределение (register allocation) в генерации кода. Для AST простого выражения "a = b + c" сгенерируйте машинный код (x86-подобный), предполагая регистры EAX, EBX, и укажите, как граф интерференции помогает распределению.

## Примерное описание и критерии оценивания к итоговому проекту

**Описание проекта:**

В рамках итогового проекта студентам предлагается разработать прототип компилятора для простого императивного или функционального языка программирования, включающий следующие ключевые этапы:

1. **Определение формальной грамматики языка**
   - Описание синтаксиса с использованием контекстно-свободной грамматики (КС-грамматики).
   - Формализация правил синтаксиса и допустимых конструкций.
2. **Построение парсера и синтаксического анализа**
   - Реализация парсера с применением методов нисходящего (LL) или восходящего (LR) разбора.
   - Построение абстрактного синтаксического дерева (AST).
3. **Анализ потоков данных (DFG)**
   - Построение графа потоков данных на основе AST.
   - Анализ и оптимизация промежуточного представления.
4. **Типизация и вывод типов**
   - Внедрение статической типизации с использованием алгоритма вывода типов (например, Хиндли-Милнера).

- Проверка корректности типов в программе.

5. **Преобразования AST**
   - Реализация локальных и глобальных преобразований AST для оптимизации и упрощения.
   - Удаление избыточных конструкций и применение правил преобразования.
6. **Генерация кода**
   - Генерация промежуточного или целевого кода с учётом архитектурных особенностей.
   - Обеспечение корректности и эффективности сгенерированного кода (управление регистрами, памятью и потоками).

**Цели проекта:**

- Продемонстрировать понимание и применение базовых концепций формальных грамматик и синтаксического анализа.
- Использовать инструменты и методы построения парсеров и AST.
- Реализовать анализ потоков данных для оптимизации кода.
- Применить алгоритмы вывода типов для обеспечения безопасности типов.
- Провести преобразования AST для улучшения качества промежуточного представления.
- Сгенерировать корректный и работоспособный код.

**Критерии оценки:**

- Корректность и полнота описания формальной грамматики языка.
- Правильная реализация парсера и построение AST без ошибок.
- Качество построения и анализа графа потоков данных (DFG), применение оптимизаций.
- Реализация статической типизации и корректный вывод типов.
- Эффективность и корректность преобразований AST.
- Корректность, эффективность и архитектурная адаптация генерируемого кода.
- Качество документации и презентации проекта, ясность пояснений и демонстраций.

**Задания для промежуточной аттестации по дисциплине (модулю)**

| № п/п | Задание | Ответ | Компетенция |
|---|---|---|---|
| 1. | Что из перечисленного является примером контекстно-свободной грамматики?<br>a) Регулярное выражение<br>b) Производственные правила вида A → α, где A — нетерминал, α — строка терминалов и нетерминалов<br>c) Контекстно-зависимая грамматика<br>d) Грамматика с ограничениями на длину правой части | b | УК-1 |
| 2. | Какой метод разбора является нисходящим?<br>a) LR-парсинг<br>b) SLR-парсинг<br>c) LALR-парсинг<br>d) LL-парсинг | d | ОПК-1 |
| 3. | Что такое абстрактное синтаксическое дерево (AST)?<br>a) Упрощённое дерево, отражающее синтаксическую структуру программы без лишних деталей<br>b) Полное дерево разбора с терминалами и нетерминалами<br>c) Граф потоков данных<br>d) Набор производственных правил | a | УК-1 |

| | | | |
|---|---|---|---|
| 4. | Для чего используется граф потоков данных (DFG)?<br>a) Для описания синтаксиса языка<br>b) Для вывода типов<br>c) Для анализа и оптимизации промежуточного представления программы<br>d) Для генерации машинного кода | c | ОПК-6 |
| 5. | Что характеризует статическую типизацию?<br>a) Типы определяются во время выполнения<br>b) Типы проверяются во время компиляции<br>c) Отсутствие контроля типов<br>d) Автоматическое преобразование типов во время исполнения | b | ОПК-6 |
| 6. | Какой алгоритм используется для вывода типов в функциональных языках?<br>a) Алгоритм Дейкстры<br>b) Алгоритм Кнута-Морриса-Пратта<br>c) Алгоритм Форда-Беллмана<br>d) Алгоритм Хиндли-Милнера | d | ОПК-1 |
| 7. | Что из перечисленного относится к локальным преобразованиям AST?<br>a) Оптимизация циклов<br>b) Анализ потоков данных<br>c) Удаление избыточных операторов<br>d) Генерация кода | c | ПК-1 |
| 8. | При генерации кода важно учитывать:<br>a) Архитектурные особенности целевой платформы<br>b) Только синтаксис исходного языка<br>c) Только правила типизации<br>d) Только структуру AST без оптимизаций | a | ПК-1 |
| 9. | Как называется тип грамматик, для которых справедлива иерархия Хомского и которые широко применяются для описания синтаксиса языков программирования? | Контекстно-свободные грамматики | УК-1 |
| 10. | Как называется метод нисходящего синтаксического анализа, основанный на предсказании? | LL-парсинг | ОПК-1 |
| 11. | Как называется алгоритм вывода типов, применяемый в функциональных языках? | Хиндли-Милнер | ОПК-1 |
| 12. | Как называется набор правил, описывающих синтаксис языка программирования? | Формальная грамматика | УК-1 |