Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования «Новосибирский национальный исследовательский
государственный университет» (Новосибирский государственный университет, НГУ)

Факультет информационных технологий

СОГЛАСОВАНО
Декан ФИТ НГУ
_____ М.М. Лаврентьев
«31» марта 2026 г.

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

Трансляция и верификация предметно-ориентированных языков

Направление подготовки: 09.03.01 ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА
Направленность (профиль): Компьютерные науки и системотехника

Форма обучения: очная
Год обучения: 4, семестр: 7

№	Вид деятельности	Семестр
		7
1	Лекции, час.	32
2	Практические занятия, час.	
3	Лабораторные занятия, час.	32
4	Занятий в контактной форме без учета промежуточной аттестации, час, из них	64
5	в электронной форме, час.	
6	из них аудиторных занятий, час.	64
7	из них в активной и интерактивной форме, час.	30
8	консультаций, час.	0
9	Самостоятельная работа, час.	78
10	в том числе на выполнение письменных работ, час	40
11	Форма аттестации (экзамен, зачет, дифференцированный зачет), час	ДЗ 2
12	Всего зачетных единиц[1]	4

Новосибирск 2026

[1] С учетом выделенных часов на промежуточную аттестацию

Рабочая программа дисциплины составлена на основании федерального государственного образовательного стандарта (ФГОС) высшего образования - бакалавриат по направлению подготовки 09.03.01 ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА.

Федеральный государственный образовательный стандарт (ФГОС) высшего образования - бакалавриат по направлению подготовки 09.03.01 ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА введен в действие приказом Минобрнауки от 19.09.2017 № 929.

Место дисциплины в структуре учебного плана: Блок 1 Дисциплины (модули); часть, формируемая участниками образовательных отношений, дисциплина по выбору.

Рабочая программа дисциплины утверждена решением Ученого совета факультета информационных технологий от 30.03.2026, протокол № 103.

Программу разработали:

Доцент кафедры систем информатики ФИТ,
кандидат физико-математических наук — И. В. Марьясов

Доцент кафедры общей информатики ФИТ — А. И. Злыгостев

Заведующий кафедрой систем информатики ФИТ,
доктор физико-математических наук — М.М. Лаврентьев

Ответственный за образовательную программу:

доцент кафедры систем информатики ФИТ,
кандидат физико-математических наук — Д.С. Мигинский

Аннотация к рабочей программе дисциплины
«Трансляция и верификация предметно-ориентированных языков»

Дисциплина «Трансляция и верификация предметно-ориентированных языков» реализуется в рамках образовательной программы высшего образования – программы бакалавриата 09.03.01 ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА, направленность (профиль): КОМПЬЮТЕРНЫЕ НАУКИ И СИСТЕМОТЕХНИКА по очной форме обучения на русском языке.

Место в образовательной программе: Дисциплина «Трансляция и верификация предметно-ориентированных языков» развивает знания, умения и навыки, сформированные у обучающихся по результатам изучения следующих дисциплин: «Введение в дискретную математику и математическую логику», «Введение в алгебру и анализ», «Модели вычислений»

Дисциплина «Трансляция и верификация предметно-ориентированных языков» реализуется в 7 семестре в рамках вариативной части дисциплин (модулей) Блока 1 и является дисциплиной по выбору.

Дисциплина «Трансляция и верификация предметно-ориентированных языков» направлена на формирование компетенций:

Способен разрабатывать требования и проектировать программное обеспечение (ПКС-1), в части следующих индикаторов достижения компетенции:

ПКС-1.5 уметь использовать программные средства для решения прикладных задач

Способен осуществлять концептуальное, функциональное и логическое проектирование систем среднего и крупного масштаба и сложности (ПКС-3), в части следующих индикаторов достижения компетенции:

ПКС-3.1 проводить эксперименты по заданной методике и анализировать результаты

Перечень основных разделов дисциплины:

1. Введение
2. Методы трансляции предметно-ориентированных языков:
   a. Понятие формальных языков
   b. Лексический анализ
   c. Синтаксический анализ
   d. Семантический анализ
   e. Промежуточное представление и оптимизация
   f. Порождение целевого кода
3. Методы верификации программ
   a. Виды формальных спецификаций
   b. Темпоральные логики и методы проверки моделей
   c. Дедуктивная верификация и SAT/SMT солверы
   d. Связь семантического анализа и верификации в современных языках программирования

При освоении дисциплины студенты выполняют следующие виды учебной работы: лекции, лабораторные занятия, самостоятельная работа. В учебном процессе предусматривается использование активных и интерактивных форм проведения занятий.

Самостоятельная работа включает: подготовку к лабораторным занятиям по разделам дисциплины, выполнение домашних работ, реализация учебного проекта.

Общий объем дисциплины – 4 зачетных единицы (144 часа).

Правила аттестации по дисциплине. Текущий контроль по дисциплине «Трансляция и верификация предметно-ориентированных языков» осуществляется на лабораторных занятиях и заключается в выполнении домашних заданий, за которые начисляются баллы. Дополнительные баллы могут быть начислены за активность на лабораторных занятиях. На 8 неделе проводится первая контрольная работа, в конце семестра проводится вторая контрольная работа. Обе работы выполняются в письменной форме по темам лекций. В течение всего семестра студенты самостоятельно реализуют учебный проект – компилятор модельного языка. Помимо защиты проекта в финале семестра, предусмотрено две промежуточных контрольных точки на 5й и 9й неделях семестра. За каждую из оцениваемых активностей начисляются баллы.

В 7 семестре результат аттестации по дисциплине оценивается по шкале «неудовлетворительно» (менее 80 баллов), «удовлетворительно» (80 и более баллов), «хорошо» (120 и более баллов), «отлично» (160 и более баллов). Оценки «отлично», «хорошо», «удовлетворительно» означают успешное прохождение промежуточной аттестации.

Оценка «отлично» соответствует продвинутому уровню сформированности компетенции.
Оценка «хорошо» соответствует базовому уровню сформированности компетенции.
Оценка «удовлетворительно» соответствует пороговому уровню сформированности компетенции.

Учебно-методическое обеспечение дисциплины.
Учебно-методический комплекс по дисциплине «Трансляция и верификация предметно-ориентированных языков»:

Шилов, Николай Вячеславович. Основы синтаксиса, семантики, трансляции и верификации программ : учебное пособие : [для студентов Фак. информ. технологий НГУ] / Н.В. Шилов ; М-во образования и науки РФ, Новосиб. гос. ун-т, Фак. информ. технологий, Каф. парал. вычислений. Новосибирск : Новосибирский государственный университет, 2011. 292 с. : ил., табл. ; 20 см. ISBN 978-5-94356-707-0.

1. Внешние требования к дисциплине

Таблица 1.1

Компетенция ПКС-1 **Способен разрабатывать требования и проектировать программное обеспечение, *в части следующих индикаторов достижения компетенции:***	
ПКС-1.5	уметь использовать программные средства для решения прикладных задач
Компетенция **ПКС-3 - Способен осуществлять концептуальное, функциональное и логическое проектирование систем среднего и крупного масштаба и сложности, *в части следующих индикаторов достижения компетенции:***	
ПКС-3.1	проводить эксперименты по заданной методике и анализировать результаты

2. Требования к результатам освоения дисциплины

Таблица 2.1

Результаты изучения дисциплины по уровням освоения (иметь представление, знать, уметь, владеть)	**Формы организации занятий**		
	Лекции	**Практики / семинары**	**Самостоятельная работа**
ПКС-1.5 - уметь использовать программные средства для решения прикладных задач			
1. Уметь формализовать язык, предложить механизм трансляции и исполнения для этого языка	+	+	+
ПКС-3.1 проводить эксперименты по заданной методике и анализировать результаты			
2. Уметь: применять методы разработки синтаксических анализаторов, трансляторов, виртуальных машин	+	+	+

3. Содержание и структура учебной дисциплины

Таблица 3.1

Темы лекций	**Активные формы, час. (входит в общее кол-во часов)**	**Часы**	**Ссылки на результаты обучения**
Семестр: 7			
1. Введение в трансляцию и компиляторы a. **Основные понятия**: Определение и назначение трансляторов и компиляторов. b. **Структура компилятора**: Обзор этапов (лексический, синтаксический, семантический анализ, генерация кода, оптимизация). c. **Классификация**: Компиляторы, интерпретаторы, ассемблеры, линковщики, загрузчики. d. **Исторический контекст**: Краткая история и эволюция методов трансляции.	0	2	1
2. Формальные языки и грамматики a. **Основы**: Алфавиты, строки, языки. Операции над языками. b. **Иерархия Хомского**: Типы грамматик (регулярные, контекстно-свободные, контекстно-зависимые, неограниченные). **c. Регулярные языки**: Регулярные грамматики и регулярные выражения, их эквивалентность.	0	2	1
3. Лексический анализ	0	2	1

a. Задачи анализатора: Токены, лексемы, шаблоны. b. Конечные автоматы: Построение НКА и ДКА по регулярным выражениям. c. Оптимизация: Минимизация ДКА (алгоритмы Хопкрофта, Бржозовского). d. Практика: Инструменты для генерации лексических анализаторов (например, Lex/Flex).			
4. Синтаксический анализ a. Цели и вызовы: Деревья разбора. Проблемы неоднозначности в КС-грамматиках. b. Классические подходы: c. Нисходящий (Top-Down) анализ: Метод рекурсивного спуска, LL-грамматики. d. Восходящий (Bottom-Up) анализ: Механизм "сдвиг-свертка", LR-грамматики (Yacc/Bison). e. Современные подходы: Обзор грамматик с упорядоченным выбором (Parsing Expression Grammars, PEG) и их преимущества (отсутствие неоднозначности). IF THEN ELSE	0	2	1
5. Семантический анализ a. Назначение: Контроль смысловой корректности программы. b. Таблица символов: Структура, управление областями видимости. c. Системы типов: d. Проверка типов: Совместимость и приведение. e. Вывод типов (Хиндли-Милнер, Rust, Swift). f. Разрешение перегрузок: Алгоритмы поиска наилучшего соответствия.	0	2	1
6. Промежуточное представление a. Необходимость и виды (трехадресный код, деревья, CFG). b. Генерация на основе синтаксического дерева и атрибутные грамматики.	0	2	1
7. Основы оптимизации a. Цели и виды (машинно-независимые и зависимые). b. Анализ потока управления: Базовые блоки и графы потока управления. c. Простейшие методы оптимизации.	0	2	1
8. Порождение целевого кода a. Выбор инструкций целевой платформы. b. Распределение регистров: Основные алгоритмы и подходы. c. Финальные шаги: Сборка и компоновка программы. d. Обзор платформы LLVM	0	2	1
9. Введение в верификацию a. Понятие корректности: Виды ошибок. Ограничения тестирования и отладки. b. Формальная верификация: Определение, цели, преимущества и недостатки. c. Динамическая верификация: Обзор методов: runtime verification, фаззинг, символическое и конколическое выполнение. d. Классификация методов: Статические и динамические подходы.	0	2	1

10. Формальная спецификация: Логические основы a. Зачем нужны формализмы: Необходимость точного описания требований. b. Базовый аппарат: Введение в логику высказываний и логику предикатов первого порядка. c. Синтаксис и семантика: Построение и интерпретация логических формул. d. пропозициональная и предикатная логики e. Системы доказательства: Общее представление об исчислении секвенций.	0	2	1
11. Темпоральные логики a. Моделирование времени: Представление поведения систем во времени. b. Линейная темпоральная логика (LTL): Операторы (Next, Finally, Globally), синтаксис, семантика. c. Логика ветвящегося времени (CTL): Операторы, синтаксис, семантика. d. Практика: Примеры спецификации свойств (безопасность, живость) с использованием темпоральных логик.	0	2	1
12. Проверка моделей (Model Checking) a. Основная идея: Автоматическая проверка выполнения спецификаций на модели системы. b. Структуры Крипке: Формальное представление систем для верификации. c. Алгоритмы: Обзор алгоритмов проверки моделей для CTL и LTL. d. Проблема взрыва состояний: Основные подходы к ее решению	0	2	1
13. Инструменты для проверки моделей a. Обзор: Популярные инструменты (Spin, NuSMV). b. Языки моделирования: Пример языка Promela для Spin. c. Практические примеры: Моделирование и верификация простых протоколов и алгоритмов с помощью инструментов.	0	2	1
14. Дедуктивная верификация a. Аксиоматическая семантика: Описание поведения программ с помощью логических аксиом. b. Логика Хоара: Тройки {P}C{Q}. Правила вывода для основных операторов языка. c. Метод наиболее слабого предусловия: Вычисление *wp(C, Q)* и его связь с логикой Хоара. d. Обзор модельного языка	0	2	1
15. Применение SAT/SMT решателей в верификации a. Введение: Что такое SAT/SMT решатели и как они работают. b. Bounded Model Checking (BMC): Применение решателей для проверки моделей на ограниченной глубине. c. Symbolic Execution: Использование символьных переменных для исследования путей выполнения программы и поиска ошибок.	0	2	1

16. Обзор передовых методов и заключение a. Абстрактная интерпретация: Принцип аппроксимации семантики программы для статического анализа. b. Верификация с помощью систем типов: Как продвинутые системы типов помогают предотвращать ошибки. c. Перспективы развития: Обзор текущих трендов в трансляции и верификации. d. Подведение итогов: Обсуждение ключевых идей курса.	0	2	1
Итого:	**0**	**32**	

Таблица 3.2

Темы лабораторных занятий	Активные формы, час. (входит в общее кол-во часов)	Часы	Ссылки на результаты обучения	Учебная деятельность
Семестр: 7				
Тема 1. Регулярные выражения и конечные автоматы	2	2	1,2	Освоение перехода от регулярных выражений к НКА/ДКА. Отработка минимизации ДКА и проверка эквивалентности.
Тема 2. Основы лексического анализа	2	2	1,2	Получение рабочей программы лексического разбора из регулярных выражений. Добавление к ней фильтров/предикатов для пробелов и комментариев. Получение и подготовка к выполнению ДЗ1.
Тема 3. Основы синтаксического анализа: LL-разбор	2	2	1,2	Преобразование грамматик для получения LL(1). Построение таблицы предсказаний и отладка рекурсивного спуска. Встраивание лексера из ДЗ1 в LL(1)-парсер.
Тема 4. Основы синтаксического анализа: LR-разбор	2	2	1,2	Построение LR(0)/SLR таблиц. Понимание и устранение конфликтов сдвиг/свёртка, свёртка/свёртка. Подготовка к ДЗ2.
Тема 5: Основы семантического анализа: таблицы имен	2	2	1,2	Построение AST для учебного языка Funny. Организация таблиц символов, учёт областей видимости.
Тема 6: Основы семантического анализа: проверка и вывод типов	2	2	1,2	Построение системы типов языка Funny. Реализация проверки типов. Обсуждение вывода типов и базовой верификации. Подготовка к ДЗ3.
Тема 7: Граф потока управления, локальные оптимизации	2	2	1,2	Построение CFG из промежуточного представления. Оценка живости/достижимости. Приведение к SSA форме. Локальные оптимизации – распространение констант, устранение общих подвыражений, устранение недостижимого кода Подготовка к ДЗ4
Промежуточная контрольная работа	0	2	1	Контроль освоения тем лексического, синтаксического, и семантического разбора текстов на формальных языках

Темы лабораторных занятий	Активные формы, час. (входит в общее кол-во часов)	Часы	Ссылки на результаты обучения	Учебная деятельность
Тема 8: Формализм требований	2	2	1,2	Перевод текстовых требований в формальные предикаты и инварианты. Определение наблюдаемых переменных и начальных/конечных условий Подготовка свойств для темпоральных спецификаций.
Тема 9: Темпоральные спецификации	2	2	1,2	Запись свойств живости и безопасности в формализмах CTL и LTL. Подготовка спецификаций для использования в проверке моделей (ДЗ5) Проверка корректности формул на примерах трасс.
Тема 10: Проверка моделей при помощи Spin	2	2	1,2	Описание модели системы на языке Promela. Проверка LTL-свойств в Spin. Анализ контрпримеров
Тема 11: Проверка моделей при помощи NuSMV	2	2	1,2	Перенос модели из предыдущей темы в NuSMV и проверка свойств LTL/CTL. Обсуждение отличий BDD/SAT от bitstate Spin и их влияние на производительность.
Тема 12: Логика Хоара и слабейшие предусловия	2	2	1,2	Применение троек Хоара к простым программам. Вычисление wp для базовых конструкций языка Funny и циклов с инвариантами. Подготовка к ДЗ6
Тема 13: Применение SMT для автоматизированной верификации корректности	2	2	1,2	Перевод свойств программы в SMT-ограничения. Получение и интерпретация контрпримеров Подготовка к ДЗ7
Тема 14: Символьное и конколическое вычисление	2	2	1,2	Разбор символьного/конколического выполнения на малых примерах. Генерация входов, воспроизводящих ветви и ошибки вычисления. Обсуждение ограничений этих подходов и их связь с фаззингом
Финал	2	2	1,2	Защита проекта, ответы на вопросы перед итоговой контрольной работой, обсуждение отчётов
Итого:	**30**	**32**		

4. Самостоятельная работа студентов

Таблица 4.1

№	Виды самостоятельной работы	Ссылки на результаты обучения	Часы на выполнение	Часы на консультации
Семестр: 7				
1	Выполнение домашних заданий	1,2	40	0
	Самостоятельное выполнение заданий в соответствии с настоящей программой			
2	Работа над учебным проектом	1,2	20	0
	Реализация компилятора модельного языка самостоятельно либо в микрогруппах (2-3 человека)			
3	Подготовка к контрольным и чтение литературы	1,2	18	0
	Повторение теоретического материала по вопросам, совпадающим с темами лекций			
	Итого		**78**	**0**

5. Образовательные технологии

В ходе реализации учебного процесса по дисциплине применяются лекционные и лабораторные занятия, а также применяются следующие интерактивные формы обучения (таблица 5.1).

Таблица 5.1

1	Портфолио	ПКС-1.5, 3.1
Формируемые умения: - **Знать:** базисные концепции и основные принципы формальных языков, и языков программирования, в частности. - **Уметь:** формализовать язык, предложить механизм трансляции и исполнения для этого языка.		
Краткое описание применения: бакалавры ведут портфолио (оценки за задания), которое является основой для проведения аттестации по дисциплине		

Для организации и контроля самостоятельной работы бакалавров, а также проведения консультаций применяются информационно-коммуникационные технологии (таблица 5.2).

Таблица 5.2

Информирование	Адрес почты – сообщается бакалаврам на первом занятии.
Консультирование	Адрес почты – сообщается бакалаврам на первом занятии.
Контроль	Адрес почты – сообщается бакалаврам на первом занятии.
Размещение учебных материалов	-

6. Правила аттестации студентов по учебной дисциплине

По дисциплине «Трансляция и верификация предметно-ориентированных языков» проводится текущая и промежуточная аттестация (итоговая по дисциплине).

Текущая аттестация по дисциплине «Трансляция и верификация предметно-ориентированных языков»:

Текущий контроль по дисциплине «Трансляция и верификация предметно-ориентированных языков» осуществляется на лабораторных занятиях и заключается в выполнении домашних заданий, за которые начисляются баллы. Дополнительные баллы могут быть начислены за активность во время лабораторных занятий. На 8 неделе проводится промежуточная контрольная работа, в конце семестра проводится итоговая контрольная работа. Обе работы выполняются в письменной форме по темам лекций. В течение всего семестра студенты самостоятельно реализуют учебный проект – компилятор модельного языка. Помимо защиты проекта в финале семестра, предусмотрено две промежуточных контрольных точки на 5й и 9й неделях семестра. За каждую из оцениваемых активностей начисляются баллы.

Промежуточная аттестация (итоговая по дисциплине) проводится в форме дифференцированного зачета на основании результатов портфолио и итоговой контрольной работы.

Итоговая оценка по дисциплине вычисляется на основе суммы баллов по всем видам контроля.

Таблица 6.1

№	Вид контроля	Максимальная сумма баллов	Доля от общей суммы баллов	Минимальное количество баллов, необходимое для успешного прохождения промежуточной аттестации
1	Домашние задания:	70	35%	28
1.1	Домашнее задание №1	10	5%	4
1.2	Домашнее задание №2	10	5%	4
1.3	Домашнее задание №3	10	5%	4
1.4	Домашнее задание №4	10	5%	4
1.5	Домашнее задание №5	10	5%	4
1.6	Домашнее задание №6	10	5%	4
1.7	Домашнее задание №7	10	5%	4
2	Учебный проект:	70	35%	32
2.1	Этап 1	20	10%	8
2.2	Этап 2	20	10%	8
2.3	Финальный этап	30	15%	16
3	Контрольные работы:	40	20%	20
3.1	Промежуточная контрольная работа	20	10	10
3.2	Итоговая контрольная работа	20	10	10
4	Активность на лабораторных занятиях	20	10%	0
	Итого	**200**	**100%**	**80**

Таблица 6.2

Сумма баллов по всем видам контроля	Итоговая оценка по дисциплине
< 80	неудовлетворительно
80 – 119	удовлетворительно

120 – 159	хорошо
160 – 200	отлично

Оценки «отлично», «хорошо», «удовлетворительно» означают успешное прохождение промежуточной аттестации.

Таблица 6.3

Коды компетенций ФГОС	**Результаты обучения**	**Формы аттестации**	
		Дифференцированный зачёт	
		1 этап - портфолио	**2 этап – Итоговая контрольная**
ПКС-1	**ПКС-1.5** уметь использовать программные средства для решения прикладных задач	+	+
ПКС-3.1	**ПКС-3.1** проводить эксперименты по заданной методике и анализировать результаты	+	+

Требования к структуре и содержанию портфолио, оценочные средства, а также критерии оценки сформированности компетенций и освоения дисциплины в целом, представлены в Фонде оценочных средств, являющемся приложением 1 к настоящей рабочей программе дисциплины.

7. Литература

1. Свердлов, С. З. Языки программирования и методы трансляции : учебное пособие для вузов / С. З. Свердлов. — 5-е изд., стер. — Санкт-Петербург : Лань, 2025. — 564 с. — ISBN 978-5-507-50570-8. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/447398. — Режим доступа: для авториз. пользователей.

2. Старолетов, С. М. Основы тестирования и верификации программного обеспечения / С. М. Старолетов. — 3-е изд., стер. — Санкт-Петербург : Лань, 2023. — 344 с. — ISBN 978-5-507-46773-0. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/319445 . — Режим доступа: для авториз. пользователей.

3. Шорников, Ю. В. Теория языков и языковых процессоров : учебник для вузов / Ю. В. Шорников. — 2-е изд., перераб. и доп. — Санкт-Петербург : Лань, 2024. — 292 с. — ISBN 978-5-507-48427-0. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/380705 . — Режим доступа: для авториз. пользователей.

Интернет-ресурсы

Таблица 7.1

№ п/п	Наименование Интернет-ресурса	Краткое описание
1	http://минобрнауки.рф/. – Загл. с экрана.	Министерство образования и науки Российской Федерации [Электронный ресурс]: официальный ресурс Минобрнауки России. – 2011. –
2	https://www.nsu.ru	Веб-сайт НГУ

8. Учебно-методическое и программное обеспечение дисциплины

8.1. Учебно-методическое обеспечение

Учебно-методический комплекс по дисциплине «Трансляция и верификация предметно-ориентированных языков»:

Шилов, Николай Вячеславович. Основы синтаксиса, семантики, трансляции и верификации программ : учебное пособие : [для студентов Фак. информ. технологий НГУ] / Н.В. Шилов ; М-во образования и науки РФ, Новосиб. гос. ун-т, Фак. информ. технологий, Каф. парал. вычислений. Новосибирск : Новосибирский государственный университет, 2011. 292 с. : ил., табл. ; 20 см. ISBN 978-5-94356-707-0. (40 экз.)

8.2. Программное обеспечение

Для обеспечения реализации дисциплины используется стандартный комплект программного обеспечения (ПО), включающий регулярно обновляемое лицензионное ПО Windows и MS Office.

Перечень специализированного программного обеспечения для изучения дисциплины представлен в таблице 8.1.

Специализированное программное обеспечение

Таблица 8.1

№	Наименование ПО	Назначение
1	Microsoft VS Code 1.90 или новее	Среда разработки приложений

9. Профессиональные базы данных и информационные справочные системы

1. Полнотекстовые журналы Springer Journals за 1997-2025 г., электронные книги (2005 – 2025 гг.)
2. Электронная библиотека диссертаций Российской государственной библиотеки (ЭБД РГБ)
3. Электронные ресурсы Web of Science Core Collection (Thomson Reuters Scientific LLC.), Journal Citation Reports + ESI
4. БД Scopus (Elsevier)

10. Материально-техническое обеспечение

Таблица 10.1

№	Наименование	Назначение
1	Презентационное оборудование (мультимедиа-проектор, экран, компьютер для управления)	Для проведения лекционных занятий
2	Компьютерный класс (с выходом в Internet)	Для проведения лабораторных занятий и организации самостоятельной работы

Материально-техническое обеспечение образовательного процесса по дисциплине для обучающихся из числа лиц с ограниченными возможностями здоровья осуществляется согласно «Порядку организации и осуществления образовательной деятельности по образовательным программам для инвалидов и лиц с ограниченными возможностями здоровья в Новосибирском государственном университете».

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования «Новосибирский национальный исследовательский
государственный университет» (Новосибирский государственный университет, НГУ)

Факультет информационных технологий

СОГЛАСОВАНО
Декан ФИТ НГУ
_____ М.М. Лаврентьев
«31» марта 2026 г.

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ
по дисциплине Трансляция и верификация предметно-ориентированных языков

Направление подготовки: 09.03.01 ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА

Направленность (профиль): Компьютерные науки и системотехника

Квалификация: бакалавр

Форма обучения: очная Год обучения: 4, семестр 7

Форма аттестации	Семестр
Дифференцированный зачёт	7

Новосибирск 2026

Фонд оценочных средств промежуточной аттестации по дисциплине является **Приложением 1** к рабочей программе дисциплины «Трансляция и верификация предметно-ориентированных языков», реализуемой в рамках образовательной программы высшего образования – программы бакалавриата 09.03.01 Информатика и вычислительная техника, направленность (профиль): Компьютерные науки и системотехника

Фонд оценочных средств промежуточной аттестации по дисциплине утвержден решением ученого совета факультета информационных технологий, протокол № 103 от 30.03.2026.

Разработчики:

Доцент кафедры систем информатики ФИТ,
кандидат физико-математических наук — И. В. Марьясов

Доцент кафедры общей информатики ФИТ — А. И. Злыгостев

Заведующий кафедрой систем информатики ФИТ,
доктор физико-математических наук — М.М. Лаврентьев

Ответственный за образовательную программу:

доцент кафедры систем информатики ФИТ,
кандидат физико-математических наук — Д.С. Мигинский

1. Содержание и порядок проведения промежуточной аттестации по дисциплине

1.1. Общая характеристика содержания промежуточной аттестации

Промежуточная аттестация по дисциплине «Трансляция и верификация предметно-ориентированных языков» проводится по завершению периода освоения образовательной программы (семестра) для оценки сформированности компетенций в части следующих индикаторов достижения компетенции (таблица П1.1).

Таблица П1.1

Код	Компетенции, формируемые в рамках дисциплины «Методы трансляции»	Семестр 7	
		Дифференцированный зачёт	
		1 этап – портфолио	2 этап – итоговая контрольная работа
ПКС-1 Способен разрабатывать требования и проектировать программное обеспечение			
ПКС-1.5	уметь использовать программные средства для решения прикладных задач	+	+
ПКС-3 Способен осуществлять концептуальное, функциональное и логическое проектирование систем среднего и крупного масштаба и сложности			
ПКС-3.1	Уметь проводить эксперименты по заданной методике и анализировать результаты	+	+

Промежуточная аттестация (итоговая по дисциплине) проводится в форме дифференцированного зачета на основании результатов портфолио и итоговой контрольной работы. Оценке подлежат следующие аспекты:

1. Выполнение домашних заданий в письменной и электронной формах
2. Выполнение учебного проекта в электронной форме
3. Выполнение контрольных работ в письменной форме
4. Активность при участии в лабораторных занятиях

Тематика вопросов контрольных работ соответствует разделам (темам) дисциплины «Трансляция и верификация предметно-ориентированных языков» (5 семестр):

1. Синтаксис формальных языков
2. Семантика языков программирования
3. Методы трансляции языков программирования
4. Методы верификации программ

1.2. Порядок проведения промежуточной аттестации по дисциплине

Промежуточная аттестация проводится в форме дифференцированного зачёта и включает два этапа: портфолио и итоговая контрольная работа. Итоговая оценка вычисляется на основе суммы баллов по всем видам контроля:

Сумма баллов по всем видам контроля	Итоговая оценка по дисциплине
< 80	неудовлетворительно
80 – 119	удовлетворительно
120 – 159	хорошо
160 – 200	отлично

2. Требования к структуре и содержанию фонда оценочных средств промежуточной аттестации по дисциплине

Перечень оценочных средств, применяемых на каждом этапе проведения промежуточной аттестации по дисциплине, представлен в таблице П1.2.

Таблица П1.2

№ п/п	Наименование оценочного средства	Краткая характеристика оценочного средства	Представление оценочного средства в фонде
Домашние задания			
1.	Задания по темам курса	Набор заданий для самостоятельного выполнения	Комплект заданий
Контрольные работы			
2	Задачи для контрольных работ	Каждая из двух контрольных работ состоит из четырёх задач на различные темы, изучаемые в теоретической части курса	Список вариантов задач по каждой из тем, входящих в контрольные работы
Учебный проект			
3	Спецификация проекта	Учебный проект проверяет способность применять на практике теоретические знания, полученные при освоении курса.	Подробное описание требований к проекту, план его этапов, критерии оценивания результатов по каждому из этапов

2.1. Требования к структуре и содержанию оценочных средств аттестации

2.1.1 **Описание оценочных средств.**

Задания для аттестации представлены в форме текста на русском языке с терминологией из курса, описывающего постановку задачи. В курсе предусмотрены задания трёх типов:

1. Домашние задания для самостоятельного выполнения.
   Предполагают написание программного кода, который можно использовать в учебном проекте, позволяют контролировать освоение обучающимися конкретных практических навыков.
2. Контрольные работы
   Предполагают письменное выполнение без помощи компьютера, позволяют контролировать освоение обучающимися теоретических концепций курса
3. Учебный проект
   Предполагает личную либо командную работу над реализацией полноценного системного инструмента; позволяет контролировать освоение учащимися основной компетенции курса.

Подробные описания заданий каждого типа приведены в следующих пунктах данного раздела.

2.1.2. **Перечень домашних заданий**

2.1.2.1. **Домашнее задание №1** «Регулярные выражения и конечные автоматы»

1. Принять список регулярных выражений для токенов языка Funny (IDENT, INT, WS, COMMENT, OP/разделители/ключевые слова `function/returns/...`).
2. Построить НКА (Томпсон) → ДКА (subset construction) → минимизированный ДКА (Хопкрофт).
3. Сгенерировать таблицу переходов (формат на выбор: JSON/CSV/код).
4. Добавить «ловушку» для непокрытых символов.

5. Подготовить набор положительных и отрицательных тестов.

2.1.2.2. **Домашнее задание №2** «Синтаксический анализ»
1. Выбрать ветку: LL(1) (рекурсивный спуск) или LR/SLR (генератор, например bison).
2. Привести грамматику к LL(1) или подготовить LR-таблицы; зафиксировать приоритеты/ассоциативность операторов.
3. Интегрировать лексер из ДЗ№1 (позиции токенов обязательны).
4. Реализовать вывод AST (человеко-читаемый или сериализуемый формат).
5. Добавить обработку ошибок: режим паники (LL) или правила/приоритеты (LR) с читаемыми сообщениями.

2.1.2.3. **Домашнее задание №3** «Семантический анализ и типы»
1. Реализовать построение таблиц символов с учётом областей видимости.
   a. Проверить на повторные объявления, использование до объявления.
2. Реализовать правила типизации для выражений, операторов, вызовов функций;
   a. Проверить сигнатуры и количества аргументов.
3. Реализовать унификацию для простых типов (int/bool/string/func) или выбранного подмножества.
4. Реализовать диагностику: выдачу сообщений об ошибках с позициями токенов/узлов.
5. Добавить аннотации к AST: хранить тип каждого выражения для последующих этапов.

2.1.2.4. **Домашнее задание №4** «Промежуточное представление и базовые оптимизации»
1. Реализовать генерацию IR для основных конструкций (выражения, ветвления, циклы, вызовы/return).
2. Реализовать построение CFG и базовых блоков.
3. Реализовать докальные оптимизации:
   a. распространение констант,
   b. алгебраические упрощения,
   c. локальное устранение недостижимого кода
   d. (опционально) устранение общих подвыражений в пределах блока.
4. Реализовать построение графа конфликтов по живости и жадную раскраску временных переменных.

2.1.2.5. **Домашнее задание №5** «Темпоральные спецификации и проверка моделей»
1. Определить модель (протокол/алгоритм) из практик 9-11 (можно доработать).
2. Записать 4-6 свойств: минимум 3 безопасности (G) и 1-2 живости (F/response). Если используете NuSMV, хотя бы одно свойство формулируется в CTL.
3. Закодировать модель в Promela (Spin) или NuSMV (выбрать основной инструмент).
   a. Опционально: перенести в другой инструмент и сохранить те же свойства для A/B сравнения.
4. Запустить проверки, собрать результаты, при необходимости уточнить модель/свойства
   a. Если делаете A/B: зафиксировать опции (POR/bitstate для Spin, BDD/SAT для NuSMV) и наблюдения по времени/памяти/трассам.
5. Описать найденные контрпримеры или подтвердить отсутствие нарушений.

2.1.2.6. **Домашнее задание №6** «Логика Хоара и слабые предусловия»
1. Выбрать 3–4 процедуры/функции (без сложных структур данных) с ветвлениями и циклами.

2. Задать предусловия и постусловия; для циклов — инварианты и условие завершения (эскиз).
3. Вычислить wp для указанных Q и тел программ, подтвердить P.
4. Описать доказательство корректности (вручную, текстом).

2.1.2.7. **Домашнее задание №7** «SMT/BMC»

4. Выбрать небольшой сценарий: функция с ветвлениями/циклами или упрощенный протокол.
5. Перевести её в набор ограничений SMT-LIB или BMC-формул (ограниченная глубина).
6. Запустить SMT-решатель (Z3 или аналог), получить sat/unsat, модель или unsat core.
5. Описать найденный контрпример (если sat) и его связь с исходной задачей.

2.1.3 **Форма заданий контрольной работы №1**

2.1.3.1. **Задание 1** «Регулярные выражения и автоматы» (5 баллов)

Дано регулярное выражение над алфавитом `{a,b}`.
1. Постройте НКА (любой эквивалентный).
2. Постройте ДКА (построение подмножеств).
3. Минимизируйте ДКА (таблица переходов).
4. Приведите пример принимаемой и отвергаемой строки.

2.1.3.2. **Задание 2** «Грамматика и синтаксический разбор» (5 баллов)

Дана грамматика в РБНФ.
1. Устраните левую рекурсию (если есть) и/или выполните левую факторизацию.
2. Постройте множества FIRST/FOLLOW для всех нетерминалов.
3. Укажите, является ли грамматика LL(1). Если да, приведите предиктивную таблицу; если нет, укажите конфликт.

2.1.3.3. **Задание 3** «Семантика и типы» (5 баллов)

Дана программа/фрагмент.
1. Постройте таблицу символов по областям видимости.
2. Найдите ошибки (если есть) и укажите их место/причину.
3. Выведите типы переменных без явной аннотации.

2.1.3.4. **Задание 4** «IR и оптимизация/кодогенерация» (5 баллов)

Дана программа/фрагмент.
1. Переведите фрагмент в трехадресный код (TAC).
2. Постройте CFG, отметьте базовые блоки.
3. Выполните одну оптимизацию (const prop/DCE/CSE) **или** сгенерируйте стековый байт-код для фрагмента и укажите максимальную глубину стека.

2.1.4. **Форма заданий для контрольной работы №2**

2.1.4.1. **Задание 1** «Формализация свойств программ» (5 баллов)

Дано описание требований для протокола/алгоритма.
1. Запишите свойства безопасности и живости в LTL и/или CTL.
2. Укажите, какое свойство к какому классу относится.

2.1.4.2. **Задание 2** «Проверка моделей» (5 баллов)
Дана Крипке-структура или маленькая модель.

1. Проверьте истинность одной формулы, привести контрпример (если нужно).
2. Коротко объясните, почему формула истинна/ложна.

2.1.4.3. **Задание 3** «Дедуктивная верификация» (5 баллов)
Дана процедура/фрагмент кода.

1. Сформулируйте предусловие/постусловие и один инвариант цикла.
2. Постройте слабейшее предусловие или сформируйте условия корректности для одного шага.

2.1.4.4. **Задание 4** «SMT/BMC/символьное выполнение» (5 баллов)

Дана небольшая программа.

1. Закодируйте условие её корректности в виде ограничений. Покажите, как найти контр-пример через SMT (модель).
2. Для BMC: разверните на k шагов и запишите формулу.

2.1.5. **Перечень вариантов заданий контрольных работ**

Перечень заданий контрольных работ, структурированный по категориям, представлен в таблице П2.1

Таблица П2.4

№п.п.	Категория/задание
Категория 1: Регулярные выражения и автоматы	
1.1	`a*(a|b)b`
1.2	`a(a*|b)b`
1.3	`a(a|b*)b`
1.4	`a(a|b)*ba`
1.5	`a(a|b)b*a`
1.6	`(a*|b)ab`
1.7	`(a|b*)ab`
1.8	`(a|b)*aba`
1.9	`(a|b)a*ba`
1.10	`(a|b)ab*a`
1.11	`a*b(a|b)a`
1.12	`ab*(a|b)a`
1.13	`ab(a*|b)`
1.14	`ab(a|b*)`
1.15	`ab(a|b)*a`
Категория 2: Грамматика и синтаксический разбор	
2.1	`E ::= E "+" T | T` `T ::= "(" E ")" | "id"`
2.2	`S ::= S "a" | T` `T ::= "b" | "c" | "d"`
2.3	`S ::= A B` `A ::= "a" A | ε`

№п.п.	Категория/задание
2.4	B ::= "b" B | "c" S ::= A
2.5	A ::= "a" A | "b" A | ε S ::= "a" A | "a" B
2.6	A ::= "c" | "e" | "f" B ::= "d" | "g"
2.7	S ::= S "b" | T T ::= "a" | "c" | ε
2.8	S ::= A B A ::= "a" | ε
2.9	B ::= "b" | ε S ::= A A
2.10	A ::= "a" A | "b" S ::= "a" S "b" | T
2.11	T ::= "c" T | "d" | ε S ::= A
2.12	A ::= "b" A | "b" | "c" | "d" S ::= A B
2.13	A ::= "a" | ε B ::= "a" | "b"
2.14	E ::= E "or" T | T T ::= "(" E ")" | "id"
2.15	S ::= A A ::= A "+" T | T
Категория 3: Семантика и типы	

3.1	let x:int = 1;
	let arr:int[] = new int[3];
	{
	  let x:int = 2;
	  let t = x + arr[0];
	  {
	  let y:int = t + 1;
	  let ok:int = y > x;
	  }
	}

3.2	let flag:bool = true;
	let n:int = 2;
	{
	  let flag:int = 0;
	  let t = flag + n;
	  {
	  let k:int = t + 1;
	  }
	  let flag:bool = false;
	}

№п.п.	Категория/задание

3.3	`let x:int = 5;`
	`let arr:int[] = new int[4];`
	`{`
	`  let x:int = arr[1];`
	`  let t = x + 1;`
	`}`
	`{`
	`  let y:int = 0;`
	`  let z:int = t;`
	`}`

3.4	`let a:int[] = new int[2];`
	`let i:int = 0;`
	`{`
	`  let i:int = 1;`
	`  let t = a[i];`
	`  {`
	`  let ok:bool = true;`
	`  let v:int = a[ok];`
	`  }`
	`}`

3.5	`let x:int = 1;`
	`let y:int = 2;`
	`{`
	`  let x:int = y;`
	`  let t = x + 1;`
	`}`
	`{`
	`  let y:int = 0;`
	`  let y:bool = false;`
	`}`

3.6	`let arr:int[] = new int[5];`
	`let n:int = 1;`
	`{`
	`  let n:int = arr[0];`
	`  let t = n + 1;`
	`  {`
	`  let k:int = m + 1;`
	`  let m:int = 2;`
	`  }`
	`}`

№п.п.	Категория/задание

3.7	`let x:int = 0;`
	`let flag:bool = true;`
	`{`
	`  let x:int = 3;`
	`  let t = x > 1;`
	`  {`
	`  let y:int = t + 1;`
	`  let z:int = x;`
	`  }`
	`}`

3.8	`let a:int[] = new int[3];`
	`let i:int = 0;`
	`{`
	`  let i:int = 1;`
	`  let t = a[i];`
	`}`
	`{`
	`  let s:int = 0;`
	`  let u:int = t + s;`
	`}`

3.9	`let x:int = 1;`
	`let y:int = 2;`
	`{`
	`  let y:int = x + 1;`
	`  let t = y + 1;`
	`  {`
	`  let z:int = t;`
	`  }`
	`  let y:bool = true;`
	`}`

3.10	`let arr:int[] = new int[4];`
	`let idx:int = 0;`
	`{`
	`  let idx:int = 1;`
	`  let t = arr[idx];`
	`}`
	`{`
	`  let ok:bool = idx > 0;`
	`  let v:int = arr[ok];`
	`}`

№п.п.	Категория/задание

3.11	`let x:int = 1;`
	`let b:bool = false;`
	`{`
	`  let x:bool = true;`
	`  let t = x;`
	`  {`
	`  let y:int = 0;`
	`  y = t;`
	`  }`
	`}`

3.12	`let a:int[] = new int[2];`
	`let x:int = 1;`
	`{`
	`  let x:int = a[0];`
	`  let t = x + 1;`
	`}`
	`{`
	`  let y:int = 0;`
	`  let z:int = t + y;`
	`}`

3.13	`let p:int = 1;`
	`let q:int = 2;`
	`{`
	`  let p:int = q + 1;`
	`  let t = p > q;`
	`  {`
	`  let r:bool = t;`
	`  let s:int = r + 1;`
	`  }`
	`}`

3.14	`let n:int = 0;`
	`let flag:bool = true;`
	`{`
	`  let n:int = 1;`
	`  let t = n + 1;`
	`}`
	`{`
	`  let flag:bool = false;`
	`  let k:int = n + 1;`
	`  let flag:bool = true;`
	`}`

№п.п.	Категория/задание

3.15	`let x:int = 0;`
	`let arr:int[] = new int[3];`
	`{`
	`  let x:int = arr[0];`
	`  let t = x + 1;`
	`  {`
	`  let y:int = t + 1;`
	`  let z:bool = y > x;`
	`  }`
	`  let w:int = y;`
	`}`

Категория 4: IR и оптимизация/кодогенерация	

4.1	`t = (a + b) + c;`
	`while (t < d) {`
	`  t = t + 1;`
	`}`
	`y = t + 1;`
	`if (y < e) {`
	`  z = t - (y + 1);`
	`} else {`
	`  z = t + (y + 1);`
	`}`

4.2	`t = (a - b) + c;`
	`while (t < d) {`
	`  t = t + 2;`
	`}`
	`y = t + 2;`
	`if (y > e) {`
	`  z = t - (y + 1);`
	`} else {`
	`  z = t + (y + 1);`
	`}`

4.3	`t = (a + b) * c;`
	`while (t < d) {`
	`  t = t + 1;`
	`}`
	`y = t + 1;`
	`if (y == e) {`
	`  z = t - (y + 2);`
	`} else {`
	`  z = t + (y + 2);`
	`}`

№п.п.	Категория/задание

4.4	`t = (a + b) - c;`
	`if (t > c) {`
	`  z = (t + 1) - a;`
	`} else {`
	`  z = (t + 1) + a;`
	`}`
	`t = t + 1;`
	`while (t < d) {`
	`  t = t + 1;`
	`}`

4.5	`t = (b - a) + c;`
	`if (t < c) {`
	`  z = (t + 2) - b;`
	`} else {`
	`  z = (t + 2) + b;`
	`}`
	`t = t + 2;`
	`while (t < d) {`
	`  t = t + 1;`
	`}`

4.6	`t = (a * b) + c;`
	`if (t == c) {`
	`  z = (t + 1) - b;`
	`} else {`
	`  z = (t + 1) + b;`
	`}`
	`t = t + 1;`
	`while (t < d) {`
	`  t = t + 2;`
	`}`

4.7	`t = (a + b) + c;`
	`while (t < d) {`
	`  y = t + 1;`
	`  if (y < e) {`
	`  z = t - (y + 1);`
	`  } else {`
	`  z = t + (y + 1);`
	`  }`
	`  t = t + 1;`
	`}`

№п.п.	Категория/задание

4.8	`t = (a - b) + c;`
	`while (t < d) {`
	`  y = t + 2;`
	`  if (y > e) {`
	`  z = t - (y + 2);`
	`  } else {`
	`  z = t + (y + 2);`
	`  }`
	`  t = t + 1;`
	`}`

4.9	`t = (a * b) - c;`
	`while (t < d) {`
	`  y = t + 1;`
	`  if (y == e) {`
	`  z = t - (y + 1);`
	`  } else {`
	`  z = t + (y + 1);`
	`  }`
	`  t = t + 2;`
	`}`

4.10	`t = (a + b) - c;`
	`if (t < c) {`
	`  y = t + 1;`
	`  while (y < d) {`
	`  y = y + 1;`
	`  }`
	`  z = t + (y + 1);`
	`} else {`
	`  y = t + 2;`
	`  z = t - (y + 1);`
	`}`

4.11	`t = (a - b) + c;`
	`if (t > c) {`
	`  y = t + 2;`
	`  while (y < d) {`
	`  y = y + 1;`
	`  }`
	`  z = t + (y + 2);`
	`} else {`
	`  y = t + 1;`
	`  z = t - (y + 2);`
	`}`

№п.п.	Категория/задание

4.12	`t = (a * b) + c;`
	`if (t == c) {`
	`  y = t + 1;`
	`  while (y < d) {`
	`  y = y + 2;`
	`  }`
	`  z = t + (y + 1);`
	`} else {`
	`  y = t + 2;`
	`  z = t - (y + 1);`
	`}`

4.13	`t = (a + b) + c;`
	`if (t < c) {`
	`  y = t + 1;`
	`  z = t - (y + 1);`
	`} else {`
	`  y = t + 2;`
	`  while (y < d) {`
	`  y = y + 1;`
	`  }`
	`  z = t + (y + 1);`
	`}`

4.14	`t = (a - b) + c;`
	`if (t > c) {`
	`  y = t + 2;`
	`  z = t - (y + 2);`
	`} else {`
	`  y = t + 1;`
	`  while (y < d) {`
	`  y = y + 1;`
	`  }`
	`  z = t + (y + 2);`
	`}`

4.15	`t = (a * b) - c;`
	`if (t == c) {`
	`  y = t + 1;`
	`  z = t - (y + 1);`
	`} else {`
	`  y = t + 2;`
	`  while (y < d) {`
	`  y = y + 2;`
	`  }`
	`  z = t + (y + 1);`
	`}`

Категория 5: Формализация свойств программ	
5.1	Атомарные утверждения: `req`, `ack`, `timeout`. Требования: 1. Никогда не бывает одновременно `ack` и `timeout`. 2. Каждый `req` в итоге приводит к `ack` или `timeout`.

№п.п.	Категория/задание
5.2	Атомарные утверждения: `crit1`, `crit2`, `req1`. Требования: 1. Процессы 1 и 2 не могут одновременно находиться в критической секции. 2. Если `req1` поднят, то процесс 1 когда-нибудь войдет в `crit1`.
5.3	Атомарные утверждения: `busy`, `idle`, `reset`. Требования: 1. Состояния `busy` и `idle` не могут быть истинны одновременно. 2. После `reset` система когда-нибудь окажется в `idle`.
5.4	Атомарные утверждения: `token1`, `token2`, `grant`. Требования: 1. Никогда не бывает одновременно `token1` и `token2`. 2. Каждый `grant` когда-нибудь приводит к появлению `token1`.
5.5	Атомарные утверждения: `full`, `empty`, `enq`, `deq`. Требования: 1. Очередь не может быть одновременно `full` и `empty`. 2. Каждый `enq` когда-нибудь приводит к `deq`.
5.6	Атомарные утверждения: `leader1`, `leader2`, `elect`. Требования: 1. Не бывает двух лидеров одновременно. 2. После `elect` когда-нибудь появится `leader1` или `leader2`.
5.7	Атомарные утверждения: `alarm`, `ok`, `reset`. Требования: 1. `alarm` и `ok` не могут быть истинны одновременно. 2. После `reset` когда-нибудь наступит `ok`.
5.8	Атомарные утверждения: `grant`, `deny`, `req`. Требования: 1. Нельзя одновременно `grant` и `deny`. 2. Каждый `req` когда-нибудь ведет к `grant` или `deny`.
5.9	Атомарные утверждения: `sent`, `delivered`, `lost`. Требования: 1. Сообщение не может быть одновременно `delivered` и `lost`. 2. Каждый `sent` когда-нибудь приводит к `delivered` или `lost`.
5.10	Атомарные утверждения: `run`, `ready`, `blocked`. Требования: 1. Процесс не может быть одновременно `run` и `blocked`. 2. Если процесс `ready`, то он когда-нибудь получит `run`
5.11	Атомарные утверждения: `hit`, `miss`, `load`. Требования: 1. `hit` и `miss` не могут быть одновременно истинны. 2. Каждый `miss` когда-нибудь приводит к `load`.
5.12	Атомарные утверждения: `doorOpen`, `moving`, `call`. Требования: 1. `doorOpen` и `moving` не истинны одновременно. 2. После `call` дверь когда-нибудь откроется (`doorOpen`).

<table>
<tr><th>№п.п.</th><th>Категория/задание</th></tr>
<tr><td>5.13</td><td>Атомарные утверждения: <code>paid</code>, <code>shipped</code>, <code>canceled</code>.
Требования:

1. Нельзя одновременно <code>shipped</code> и <code>canceled</code>.
2. После <code>paid</code> заказ когда-нибудь будет <code>shipped</code>.</td></tr>
<tr><td>5.14</td><td>Атомарные утверждения: <code>login</code>, <code>logout</code>, <code>error</code>.
Требования:

1. Нельзя одновременно <code>login</code> и <code>logout</code>.
2. После <code>login</code> когда-нибудь произойдет <code>logout</code> или <code>error</code>.</td></tr>
<tr><td>5.15</td><td>Атомарные утверждения: <code>heartbeat</code>, <code>reboot</code>, <code>fail</code>.
Требования:

1. Нельзя одновременно <code>heartbeat</code> и <code>reboot</code>.
2. После <code>fail</code> система когда-нибудь выполнит <code>reboot</code>.</td></tr>
<tr><td colspan="2">Категория 6: Проверка моделей</td></tr>
<tr><td>6.1</td><td>Крипке-структура:
<table><tr><th>Состояние</th><th>Метки</th></tr><tr><td>s0 (init)</td><td>p</td></tr><tr><td>s1</td><td>p</td></tr><tr><td>s2</td><td>q</td></tr></table>Переходы:
<code>s0 -> s1, s2</code>
<code>s1 -> s1</code>
<code>s2 -> s2</code>
Проверьте формулу CTL: <code>AG p</code>. Если формула ложна, приведите контрпример (путь).</td></tr>
<tr><td>6.2</td><td>Крипке-структура:
<table><tr><th>Состояние</th><th>Метки</th></tr><tr><td>s0 (init)</td><td>p</td></tr><tr><td>s1</td><td>p</td></tr><tr><td>s2</td><td>q</td></tr></table>Переходы:
<code>s0 -> s1, s2</code>
<code>s1 -> s1</code>
<code>s2 -> s2</code>
Проверьте формулу CTL: <code>EF q</code>. Если формула ложна, приведите контрпример (путь).</td></tr>
<tr><td>6.3</td><td>Крипке-структура:
<table><tr><th>Состояние</th><th>Метки</th></tr><tr><td>s0 (init)</td><td>p</td></tr><tr><td>s1</td><td>p</td></tr><tr><td>s2</td><td>q</td></tr></table>Переходы:
<code>s0 -> s1, s2</code>
<code>s1 -> s1</code>
<code>s2 -> s2</code>
Проверьте формулу CTL: <code>AF p</code>. Если формула ложна, приведите контрпример (путь).</td></tr>
</table>

<table>
<tr><th>№п.п.</th><th>Категория/задание</th></tr>
<tr><td>6.4</td><td>Крипке-структура:
<table><tr><th>Состояние</th><th>Метки</th></tr><tr><td>s0 (init)</td><td>-</td></tr><tr><td>s1</td><td>p</td></tr><tr><td>s2</td><td>-</td></tr><tr><td>s3</td><td>q</td></tr></table>Переходы:
<code>s0 -> s1, s2</code>
<code>s1 -> s3</code>
<code>s2 -> s3</code>
<code>s3 -> s3</code>
Проверьте формулу CTL: <code>AF q</code>. Если формула ложна, приведите контрпример (путь).</td></tr>
<tr><td>6.5</td><td>Крипке-структура:
<table><tr><th>Состояние</th><th>Метки</th></tr><tr><td>s0 (init)</td><td>-</td></tr><tr><td>s1</td><td>p</td></tr><tr><td>s2</td><td>-</td></tr><tr><td>s3</td><td>q</td></tr></table>Переходы:
<code>s0 -> s1, s2</code>
<code>s1 -> s3</code>
<code>s2 -> s3</code>
<code>s3 -> s3</code>
Проверьте формулу CTL: <code>AG (p -> AF q)</code>. Если формула ложна, приведите контрпример (путь).</td></tr>
<tr><td>6.6</td><td>Крипке-структура:
<table><tr><th>Состояние</th><th>Метки</th></tr><tr><td>s0 (init)</td><td>-</td></tr><tr><td>s1</td><td>p</td></tr><tr><td>s2</td><td>-</td></tr><tr><td>s3</td><td>q</td></tr></table>Переходы:
<code>s0 -> s1, s2</code>
<code>s1 -> s3</code>
<code>s2 -> s3</code>
<code>s3 -> s3</code>
Проверьте формулу CTL: <code>EG q</code>. Если формула ложна, приведите контрпример (путь).</td></tr>
<tr><td>6.7</td><td>Крипке-структура:
<table><tr><th>Состояние</th><th>Метки</th></tr><tr><td>s0 (init)</td><td>-</td></tr><tr><td>s1</td><td>p</td></tr><tr><td>s2</td><td>-</td></tr><tr><td>s3</td><td>q</td></tr></table>Переходы:
<code>s0 -> s1, s2</code>
<code>s1 -> s3</code>
<code>s2 -> s3</code>
<code>s3 -> s3</code>
Проверьте формулу CTL: <code>EG p</code>. Если формула ложна, приведите контрпример (путь).</td></tr>
<tr><td>6.8</td><td>Крипке-структура:
<table><tr><th>Состояние</th><th>Метки</th></tr><tr><td>a0 (init)</td><td>-</td></tr><tr><td>a1</td><td>p</td></tr><tr><td>a2</td><td>p</td></tr><tr><td>a3</td><td>q</td></tr></table>Переходы:
<code>a0 -> a1, a2</code>
<code>a1 -> a1, a3</code>
<code>a2 -> a2</code>
<code>a3 -> a3</code>
Проверьте формулу CTL: <code>AF q</code>. Если формула ложна, приведите контрпример (путь).</td></tr>
</table>

<table>
<tr><th>№п.п.</th><th>Категория/задание</th></tr>
<tr><td>6.9</td><td>Крипке-структура:
<table><tr><th>Состояние</th><th>Метки</th></tr><tr><td>a0 (init)</td><td>-</td></tr><tr><td>a1</td><td>p</td></tr><tr><td>a2</td><td>p</td></tr><tr><td>a3</td><td>q</td></tr></table>Переходы:
<code>a0 -> a1, a2</code>
<code>a1 -> a1, a3</code>
<code>a2 -> a2</code>
<code>a3 -> a3</code>
Проверьте формулу CTL: <code>EF q</code>. Если формула ложна, приведите контрпример (путь).</td></tr>
<tr><td>6.10</td><td>Крипке-структура:
<table><tr><th>Состояние</th><th>Метки</th></tr><tr><td>t0 (init)</td><td>-</td></tr><tr><td>t1</td><td>p</td></tr><tr><td>t2</td><td>q</td></tr></table>Переходы:
<code>t0 -> t1</code>
<code>t1 -> t2</code>
<code>t2 -> t1</code>
Проверьте формулу CTL: <code>AG (p -> AF q)</code>. Если формула ложна, приведите контрпример (путь).</td></tr>
<tr><td>6.11</td><td>Крипке-структура:
<table><tr><th>Состояние</th><th>Метки</th></tr><tr><td>t0 (init)</td><td>-</td></tr><tr><td>t1</td><td>p</td></tr><tr><td>t2</td><td>q</td></tr></table>Переходы:
<code>t0 -> t1</code>
<code>t1 -> t2</code>
<code>t2 -> t1</code>
Проверьте формулу CTL: <code>EG p</code>. Если формула ложна, приведите контрпример (путь).</td></tr>
<tr><td>6.12</td><td>Крипке-структура:
<table><tr><th>Состояние</th><th>Метки</th></tr><tr><td>t0 (init)</td><td>-</td></tr><tr><td>t1</td><td>p</td></tr><tr><td>t2</td><td>q</td></tr></table>Переходы:
<code>t0 -> t1</code>
<code>t1 -> t2</code>
<code>t2 -> t1</code>
Проверьте формулу CTL: <code>AF p</code>. Если формула ложна, приведите контрпример (путь).</td></tr>
<tr><td>6.13</td><td>Крипке-структура:
<table><tr><th>Состояние</th><th>Метки</th></tr><tr><td>u0 (init)</td><td>-</td></tr><tr><td>u1</td><td>p</td></tr><tr><td>u2</td><td>r</td></tr><tr><td>u3</td><td>p, q</td></tr></table>Переходы:
<code>u0 -> u1, u2</code>
<code>u1 -> u3</code>
<code>u2 -> u2</code>
<code>u3 -> u3</code>
Проверьте формулу CTL: <code>AX p</code>. Если формула ложна, приведите контрпример (путь).</td></tr>
</table>

<table>
<tr><th>№п.п.</th><th>Категория/задание</th></tr>
<tr><td>6.14</td><td>Крипке-структура:
<table><tr><th>Состояние</th><th>Метки</th></tr><tr><td>u0 (init)</td><td>-</td></tr><tr><td>u1</td><td>p</td></tr><tr><td>u2</td><td>r</td></tr><tr><td>u3</td><td>p, q</td></tr></table>Переходы:
<code>u0 -> u1, u2</code>
<code>u1 -> u3</code>
<code>u2 -> u2</code>
<code>u3 -> u3</code>
Проверьте формулу CTL: <code>EX r</code>. Если формула ложна, приведите контрпример (путь).</td></tr>
<tr><td>6.15</td><td>Крипке-структура:
<table><tr><th>Состояние</th><th>Метки</th></tr><tr><td>u0 (init)</td><td>-</td></tr><tr><td>u1</td><td>p</td></tr><tr><td>u2</td><td>r</td></tr><tr><td>u3</td><td>p, q</td></tr></table>Переходы:
<code>u0 -> u1, u2</code>
<code>u1 -> u3</code>
<code>u2 -> u2</code>
<code>u3 -> u3</code>
Проверьте формулу CTL: <code>AG EF q</code>. Если формула ложна, приведите контрпример (путь).</td></tr>
<tr><td colspan="2">Категория 7: Дедуктивная верификация</td></tr>
<tr><td>7.1</td><td><pre>x = n;
y = 0;
while (x > 0) {
  y = y + m;
  x = x - 1;
}</pre>Спецификация: при неотрицательном n после выполнения программы значение y должно равняться произведению n и m, а x - нулю.</td></tr>
<tr><td>7.2</td><td><pre>x = k;
y = 0;
while (x > 0) {
  y = y + b;
  x = x - 1;
}</pre>Спецификация: при неотрицательном k после выполнения программы значение y должно равняться произведению k и b, а x - нулю.</td></tr>
<tr><td>7.3</td><td><pre>x = p;
y = 0;
while (x > 0) {
  y = y + q;
  x = x - 1;
}</pre>Спецификация: при неотрицательном p после выполнения программы значение y должно равняться произведению p и q, а x - нулю.</td></tr>
</table>

№п.п.	Категория/задание

7.4	`i = 0;`
	`s = 0;`
	`while (i < n) {`
	`  i = i + 1;`
	`  s = s + i;`
	`}`
	Спецификация: при неотрицательном `n` после выполнения программы значение `i` должно равняться `n`, а `s` – половине произведения `n` на `n - 1`.

7.5	`i = 0;`
	`s = 0;`
	`while (i < k) {`
	`  i = i + 1;`
	`  s = s + i;`
	`}`
	Спецификация: при неотрицательном `k` после выполнения программы значение `i` должно равняться `k`, а `s` – половине произведения `k` на `k - 1`.

7.6	`i = 0;`
	`s = 0;`
	`while (i < m) {`
	`  i = i + 1;`
	`  s = s + i;`
	`}`
	Спецификация: при неотрицательном `m` после выполнения программы значение `i` должно равняться `m`, а `s` – половине произведения `m` на `m - 1`.

7.7	`i = 0;`
	`p = 1;`
	`while (i < n) {`
	`  p = p * a;`
	`  i = i + 1;`
	`}`
	Спецификация: при неотрицательном `n` после выполнения `i` должно равняться `n`, а `p` равняться `a` в степени `n`.

7.8	`i = 0;`
	`p = 1;`
	`while (i < k) {`
	`  p = p * b;`
	`  i = i + 1;`
	`}`
	Спецификация: при неотрицательном `k` после выполнения `i` должно равняться `k`, а `p` равняться `b` в степени `k`.

7.9	`i = 0;`
	`p = 1;`
	`while (i < m) {`
	`  p = p * c;`
	`  i = i + 1;`
	`}`
	Спецификация: при неотрицательном `m` после выполнения `i` должно равняться `m`, а `p` равняться `c` в степени `m`.

№п.п.	Категория/задание

7.10	`x = n;`
	`y = c;`
	`while (x > 0) {`
	`  y = y + d;`
	`  x = x - 1;`
	`}`
	Спецификация: при неотрицательном `n` после выполнения `y` должно равняться сумме `c` и произведения `n` на `d`, а `x` - нулю.

7.11	`x = k;`
	`y = p;`
	`while (x > 0) {`
	`  y = y + q;`
	`  x = x - 1;`
	`}`
	Спецификация: при неотрицательном `k` после выполнения `y` должно равняться сумме `p` и произведения `k` на `q`, а `x` - нулю.

7.12	`x = m;`
	`y = r;`
	`while (x > 0) {`
	`  y = y + s;`
	`  x = x - 1;`
	`}`
	Спецификация: при неотрицательном `m` после выполнения `y` должно равняться сумме `r` и произведения `m` на `s`, а `x` - нулю.

7.13	`i = 0;`
	`s = 0;`
	`while (i < n) {`
	`  s = s + (2 * i + 1);`
	`  i = i + 1;`
	`}`
	Спецификация: при неотрицательном `n` после выполнения `i` должно равняться `n`, а `s` – квадрату `n`.

7.14	`i = 0;`
	`s = 0;`
	`while (i < k) {`
	`  s = s + (2 * i + 1);`
	`  i = i + 1;`
	`}`
	Спецификация: при неотрицательном `k` после выполнения `i` должно равняться `k`, а `s` – квадрату `k`.

7.15	`i = 0;`
	`s = 0;`
	`while (i < m) {`
	`  s = s + (2 * i + 1);`
	`  i = i + 1;`
	`}`
	Спецификация: при неотрицательном `m` после выполнения `i` должно равняться `m`, а `s` – квадрату `m`.

Категория 8: SMT/BMC/символьное выполнение	

№п.п.	Категория/задание	

8.1	`x = a + 1;`
	`y = b - 1;`
	`if (x > y) {`
	`  z = x - y;`
	`} else {`
	`  z = y - x;`
	`}`
	`assert(z >= 2);`
	Запишите SMT-формулу для поиска контрпримера к `assert` и приведите пример модели
		`i = 0;`
		`x = a;`
		`while (i < 2) {`
		`  x = x + 1;`
		`  i = i + 1;`
		`}`
		`assert(x <= 1);`
		Разверните цикл на `k=2` и запишите BMC-формулу для нарушения `assert`, укажите пример модели.

8.2	`x = a + 2;`
	`y = b - 0;`
	`if (x > y) {`
	`  z = x - y;`
	`} else {`
	`  z = y - x;`
	`}`
	`assert(z >= 3);`
	Запишите SMT-формулу для поиска контрпримера к `assert` и приведите пример модели
		`i = 0;`
		`x = a;`
		`while (i < 2) {`
		`  x = x + 1;`
		`  i = i + 1;`
		`}`
		`assert(x <= 2);`
		
		Разверните цикл на `k=2` и запишите BMC-формулу для нарушения `assert`, укажите пример модели.

8.3	`x = a + 0;`
	`y = b - 2;`
	`if (x > y) {`
	`  z = x - y;`
	`} else {`
	`  z = y - x;`
	`}`
	`assert(z >= 2);`
	Запишите SMT-формулу для поиска контрпримера к `assert` и приведите пример модели
		`i = 0;`
		`x = a;`
		`while (i < 2) {`
		`  x = x + 2;`
		`  i = i + 1;`
		`}`
		`assert(x <= 3);`
		
		Разверните цикл на `k=2` и запишите BMC-формулу для нарушения `assert`, укажите пример модели.

8.4	`x = a + 1;`
	`y = b - 2;`
	`if (x > y) {`
	`  z = x - y;`
	`} else {`
	`  z = y - x;`
	`}`
	`assert(z >= 3);`
	Запишите SMT-формулу для поиска контрпримера к `assert` и приведите пример модели
		`i = 0;`
		`x = a;`
		`while (i < 2) {`
		`  x = x + 1;`
		`  i = i + 1;`
		`}`
		`assert(x <= 0);`
		
		Разверните цикл на `k=2` и запишите BMC-формулу для нарушения `assert`, укажите пример модели.

№п.п.	Категория/задание	

8.5	`x = a + 2;`
	`y = b - 1;`
	`if (x > y) {`
	`  z = x - y;`
	`} else {`
	`  z = y - x;`
	`}`
	`assert(z >= 2);`
	Запишите SMT-формулу для поиска контрпримера к `assert` и приведите пример модели
		`i = 0;`
		`x = a;`
		`while (i < 2) {`
		`  x = x + 2;`
		`  i = i + 1;`
		`}`
		`assert(x <= 2);`
		
		Разверните цикл на `k=2` и запишите BMC-формулу для нарушения `assert`, укажите пример модели.

8.6	`x = a + 3;`
	`y = b - 0;`
	`if (x > y) {`
	`  z = x - y;`
	`} else {`
	`  z = y - x;`
	`}`
	`assert(z >= 2);`
	Запишите SMT-формулу для поиска контрпримера к `assert` и приведите пример модели
		`i = 0;`
		`x = a;`
		`while (i < 2) {`
		`  x = x + 1;`
		`  i = i + 1;`
		`}`
		`assert(x <= 3);`
		
		Разверните цикл на `k=2` и запишите BMC-формулу для нарушения `assert`, укажите пример модели.

8.7	`x = a + 0;`
	`y = b - 1;`
	`if (x > y) {`
	`  z = x - y;`
	`} else {`
	`  z = y - x;`
	`}`
	`assert(z >= 1);`
	Запишите SMT-формулу для поиска контрпримера к `assert` и приведите пример модели
		`i = 0;`
		`x = a;`
		`while (i < 2) {`
		`  x = x + 2;`
		`  i = i + 1;`
		`}`
		`assert(x <= 1);`
		
		Разверните цикл на `k=2` и запишите BMC-формулу для нарушения `assert`, укажите пример модели.

8.8	`x = a + 2;`
	`y = b - 2;`
	`if (x > y) {`
	`  z = x - y;`
	`} else {`
	`  z = y - x;`
	`}`
	`assert(z >= 3);`
	Запишите SMT-формулу для поиска контрпримера к `assert` и приведите пример модели
		`i = 0;`
		`x = a;`
		`while (i < 2) {`
		`  x = x + 1;`
		`  i = i + 1;`
		`}`
		`assert(x <= 4);`
		
		Разверните цикл на `k=2` и запишите BMC-формулу для нарушения `assert`, укажите пример модели.

№п.п.	Категория/задание	

8.9	`x = a + 1;`
	`y = b - 0;`
	`if (x > y) {`
	`  z = x - y;`
	`} else {`
	`  z = y - x;`
	`}`
	`assert(z >= 2);`
	Запишите SMT-формулу для поиска контрпримера к `assert` и приведите пример модели
		`i = 0;`
		`x = a;`
		`while (i < 2) {`
		`  x = x + 2;`
		`  i = i + 1;`
		`}`
		`assert(x <= 1);`
		
		Разверните цикл на `k=2` и запишите BMC-формулу для нарушения `assert`, укажите пример модели.

8.10	`x = a + 3;`
	`y = b - 1;`
	`if (x > y) {`
	`  z = x - y;`
	`} else {`
	`  z = y - x;`
	`}`
	`assert(z >= 3);`
	Запишите SMT-формулу для поиска контрпримера к `assert` и приведите пример модели
		`i = 0;`
		`x = a;`
		`while (i < 2) {`
		`  x = x + 1;`
		`  i = i + 1;`
		`}`
		`assert(x <= 2);`
		
		Разверните цикл на `k=2` и запишите BMC-формулу для нарушения `assert`, укажите пример модели.

8.11	`x = a + 0;`
	`y = b - 3;`
	`if (x > y) {`
	`  z = x - y;`
	`} else {`
	`  z = y - x;`
	`}`
	`assert(z >= 2);`
	Запишите SMT-формулу для поиска контрпримера к `assert` и приведите пример модели
		`i = 0;`
		`x = a;`
		`while (i < 2) {`
		`  x = x + 2;`
		`  i = i + 1;`
		`}`
		`assert(x <= 0);`
		
		Разверните цикл на `k=2` и запишите BMC-формулу для нарушения `assert`, укажите пример модели.

8.12	`x = a + 2;`
	`y = b - 1;`
	`if (x > y) {`
	`  z = x - y;`
	`} else {`
	`  z = y - x;`
	`}`
	`assert(z >= 3);`
	Запишите SMT-формулу для поиска контрпримера к `assert` и приведите пример модели
		`i = 0;`
		`x = a;`
		`while (i < 2) {`
		`  x = x + 1;`
		`  i = i + 1;`
		`}`
		`assert(x <= 1);`
		
		Разверните цикл на `k=2` и запишите BMC-формулу для нарушения `assert`, укажите пример модели.

№п.п.	Категория/задание	

8.13	`x = a + 1;`
	`y = b - 3;`
	`if (x > y) {`
	`  z = x - y;`
	`} else {`
	`  z = y - x;`
	`}`
	`assert(z >= 4);`
	Запишите SMT-формулу для поиска контрпримера к `assert` и приведите пример модели
		`i = 0;`
		`x = a;`
		`while (i < 2) {`
		`  x = x + 2;`
		`  i = i + 1;`
		`}`
		`assert(x <= 3);`
		
		Разверните цикл на `k=2` и запишите BMC-формулу для нарушения `assert`, укажите пример модели.

8.14	`x = a + 3;`
	`y = b - 2;`
	`if (x > y) {`
	`  z = x - y;`
	`} else {`
	`  z = y - x;`
	`}`
	`assert(z >= 4);`
	Запишите SMT-формулу для поиска контрпримера к `assert` и приведите пример модели
		`i = 0;`
		`x = a;`
		`while (i < 2) {`
		`  x = x + 1;`
		`  i = i + 1;`
		`}`
		`assert(x <= 5);`
		
		Разверните цикл на `k=2` и запишите BMC-формулу для нарушения `assert`, укажите пример модели.

8.15	`x = a + 0;`
	`y = b - 0;`
	`if (x > y) {`
	`  z = x - y;`
	`} else {`
	`  z = y - x;`
	`}`
	`assert(z >= 1);`
	Запишите SMT-формулу для поиска контрпримера к `assert` и приведите пример модели
		`i = 0;`
		`x = a;`
		`while (i < 2) {`
		`  x = x + 2;`
		`  i = i + 1;`
		`}`
		`assert(x <= 2);`
		
		Разверните цикл на `k=2` и запишите BMC-формулу для нарушения `assert`, укажите пример модели.

Набор заданий контрольных работ формируется и утверждается в установленном порядке в начале учебного года при наличии контингента обучающихся, завершающих освоение дисциплины «Трансляция и верификация предметно-ориентированных языков» в текущем учебном году.

2.1.6. **Спецификация учебного проекта**

Цель: собрать компилятор языка Funny в байткод виртуальной машины Java. Верификационные расширения (wp/VC/SMT, контракты времени выполнения) относятся к бонусным возможностям. Обязательные части оцениваются автотестами/контрольными точками, бонусы дают дополнительные баллы.

2.1.6.1. Целевая платформа и ABI (обязательные требования)

- Формат: JVM `.class`, генерация любым байткод-эмиттером (ASM/Jasmin/аналог). На основе одного модуля Funny должен порождаться один `public final` класс.
- Для каждой функции модуля должен порождаться один `public static` метод; имя метода должно совпадать с именем функции Funny.

- Типы: `int` → JVM `int`; массивы Funny отображаются в предоставленный рантайм-класс `ImmutableIntArray` (см. ниже).
- Неизменяемость параметров: параметры (в т.ч. массивы) читать можно, писать нельзя. Обновление массива через `b = a; b[i] = v;` генерирует новый экземпляр `ImmutableIntArray` (функциональное обновление, см. обессахаривание).
- Множественные возвраты (обязательный вариант): кортеж как `int[]` фиксированной длины; порядок элементов соответствует порядку переменных в блоке `returns`.
- Исключения: можно полагаться на JVM (`ArithmeticException`, `ArrayIndexOutOfBoundsException`) или вставлять явные проверки – выбранный вариант описать в отчёте.

2.1.6.2 Допустимые бонус-варианты ABI
- Возвраты через сгенерированный record/POJO вместо `int[]`.
- Параллельный вывод в альтернативный целевой язык/платформу (LLVM IR/регистровая VM).
- Альтернатива массивам: можно использовать голые `int[]` + хелперы копирования/обновления вместо `ImmutableIntArray`.

2.1.6.3 Предоставляемый рантайм-класс

Готовый `ImmutableIntArray` (часть окружения проекта):
- `private final int[] data;`
- `public int length()`
- `public int get(int idx)` (бросает исключение при неверном индексе).
- `public ImmutableIntArray with(int idx, int value)` – копирует, меняет элемент, возвращает новый объект.
- `public static ImmutableIntArray of(int... elems)`
  `public static ImmutableIntArray (int[] src)` – фабрики.

Порождаемый код по умолчанию обязан использовать этот класс для семантики массивов.

2.1.6.4 Язык реализации
- Обязательный минимум: Java/Kotlin/TypeScript/Node/Python (или иной), если есть путь до генерации `.class` (ASM/Jasmin/генерация `.j` + jasmin).
- Бонус: Rust/Go с обёрткой над ASM или генерацией `.j`.

2.1.6.5 Контрольные точки

1. T0 (неделя 5)
   - Лексер/парсер для актуальной спецификации Funny, включая обессахаривание массивов (`a[i]=v` → `a=a.with(i,v)`).
   - AST + вывод типов локальных (тип по первому присваиванию; параметры/возвраты — явный тип).
   - Тесты: эталонные токены/AST/обессахаривание, отрицательные синтаксические сценарии.

2. T1 (неделя 9)
   - Семантика: проверки имён/типов, запрет записи в параметры/параметры-массивы; ошибки с позициями.
   - IR (TAC) + CFG + базовые оптимизации: распространение констант/упрощения, локальный DCE (минимум); опционально локальный CSE.
   - Маппинг типов на JVM (`int`/`ImmutableIntArray`), схема многозначного возврата через `int[]`.

- Прототип кодогенерации: выражения, присваивания, `if`/`while` → валидный байткод (не обязательно всё покрывать, но связка AST → байткод должна работать).
- Тесты: типовые ошибки, позитивные примеры → байткод → запуск JVM, сверка вывода/исключений.

3. Финал (неделя 16)
   - Полная генерация кода для поддерживаемого подмножества Funny (кортежные вызовы, массивы, обессахаривание обновлений).
   - Исполнитель/рантайм: CLI/скрипт, принимающий `.funny`, выдающий `.class` и запускающий для проверки результатов/исключений.
   - Тесты: корректные программы (результат совпадает с эталоном), отрицательные сценарии на ошибки типов/семантики, проверки времени выполнения JVM (деление на 0, выход за границы).
   - Отчёт: как реализованы копирование массивов (`ImmutableIntArray`), многозначный возврат, стратегия проверок (компилятор/рантайм).

2.1.6.6 Бонусные возможности (добавляют баллы)

1. VC/wps + SMT (верификация контрактов)
   - **Что сделать**: научиться порождать условия корректности (слабейшее предусловие) по `requires`/`ensures`/`invariant`/`assert`/`assume`, конвертировать их в формат SMT-LIB и вызывать Z3; sat → контрпример, unsat → корректность доказана.
   - **Критерий успеха**: 2-3 корректных программы дают unsat, 2-3 некорректных аннотации дают sat с моделью; скрипт для массовой проверки корректности нескольких программ.

2. Верификация трансляции (локальных оптимизаций)
   - **Что сделать**: выбрать 1-2 оптимизации (например, `add x,0`→`x`, `mul x,1`→`x`, удаление `mov r,r`) и проверять эквивалентность промежуточного представления до/после через SMT.
   - **Критерий успеха**: утилита/скрипт, генерирующая SMT-LIB для пары блоков до/после; Z3 возвращает unsat для корректных правил и sat с моделью для намеренно испорченного. Приложить 2-3 демонстрационных примера.

3. Runtime `assert/assume` по флагу
   - **Что сделать**: добавить флаг сборки (`--runtime-asserts` или аналог); при включении вставлять проверки предикатов в байткод (бросок исключения при нарушении), при выключении – не вставлять.
   - **Критерий успеха**: тесты показывают, что с флагом нарушение `assert`/`assume` вызывает исключение, без флага — не влияет на выполнение (верификация при этом остаётся).

4. Record/POJO для множественного возврата

   - **Что сделать**: генерировать record/POJO-класс для функций с несколькими возвращаемыми значениями и использовать его вместо `int[]`.
   - **Критерий успеха**: корректный байткод, вызывающая сторона распаковывает поля; тесты на функции с 2-3 возвратами.

5. Проверка модели стека/фрейма (Spin/NuSMV)

- **Что сделать**: смоделировать push/pop фреймов и возвраты с ограниченной глубиной; задать свойства safety (<длина стека ≤ MaxDepth>, <нет выхода за рамки массива кадров>) и liveness (<возврат на корректный адрес>).
- **Критерий успеха**: файл модели + спецификации (LTL/CTL), запуск Spin/NuSMV с одним корректным и одним нарушающим сценарием (контрпример предъявлен).

6. Альтернативные целевые платформы
   - **Что сделать**: поддержать дополнительный вывод в регистровую схему или LLVM IR для подмножества языка (арифметика, присваивания, if/while).
   - **Критерий успеха**: генератор альтернативного вывода + запуск/интерпретация примеров с совпадающим результатом; задокументированы отличия ABI.

7. Расширение набора типов
   - **Bool**: добавить `bool` как полноценный тип переменных/параметров/возвратов; обновить типизацию, VC, генерация кода (`boolean` в JVM).
   - **Параметризованный `array<T>`**: минимум `array<int>`, корректная типизация и генерация кода, обновлённые правила VC.
   - **Критерий успеха**: тесты, покрывающие новые типы, и описание изменений в правилах/кодогенерации.

2.1.6.7 Минимальный набор тестов

- **Положительные примеры**: арифметика, if/while, копирование массивов, функциональное обновление `a[i]=v`, многозначный возврат, контракты функций/циклов, `assert/assume` (демонстрация), рекурсия.
- **Негативные примеры**: синтаксические ошибки, ошибки типов, модификация параметров/массивов, неверные кортежи/аргументы, нарушения аннотаций (только если делаете генерацию условий корректности), выход за границы/деление на 0 (исключения JVM).
- **Бонус**: сценарии для VC/SMT, валидация трансляции, утверждения (assert) времени выполнения, модели Spin/NuSMV.

2.1.6.8. Оценивание
- **Обязательная часть**: успешные контрольные точки (T0/T1/финал) + прохождение тестов.
- **Бонусы**: явно перечислить в отчёте, что реализовано (VC/SMT, контракты времени выполнения, валидация трансляции, альтернативные платформы/типы и т.д.).
- **Качество**: читаемость кода, модульность, наличие CLI/скриптов для сборки/запуска, документация по форматам (AST/IR/SMT/байткод).

3. Критерии оценки сформированности компетенций в рамках промежуточной аттестации по дисциплине

Таблица П1.5

Шифр компетенций	Структурные элементы оценочных средств	Показатель сформированности	Не сформирован (2 балла)	Пороговый уровень (3 балла)	Базовый уровень (4 балла)	Продвинутый уровень (5 баллов)
ПКС-1.5	Домашние задания Вопросы контрольных работ Спецификация учебного проекта	**ПКС-1.5** уметь использовать программные средства для решения прикладных задач	Не имеет представления об основных типах языков программирования, средствах исполнения программ, принципах работы трансляторов и компиляторов.	Имеет представление об основных типах языков программирования, средствах исполнения программ, принципах работы трансляторов и компиляторов. Умеет самостоятельно изучать базовый материал, решать задачи начального уровня.	Знает основные типы языков программирования, принципы работы сред исполнения, алгоритмы синтаксического и семантического анализа, трансляции языков. Умеет реализовывать компоненты трансляторов и инструментальной поддержки языков.	Знает основные типы языков программирования, принципы работы сред исполнения, алгоритмы синтаксического и семантического анализа, трансляции языков. Умеет реализовывать такие комплексные системы как трансляторы языков высокого уровня.
ПКС-3.1	Домашние задания Вопросы контрольных работ Спецификация учебного проекта	**ПКС-3.1** проводить эксперименты по заданной методике и анализировать результаты	Не умеет применять методы разработки синтаксических анализаторов, трансляторов, виртуальных машин	Слабо умеет применять методы разработки синтаксических анализаторов, трансляторов, виртуальных машин, допускает грубые ошибки	Умеет применять методы разработки синтаксических анализаторов, трансляторов, виртуальных машин в рамках учебных задач, допускает незначительные ошибки	Умеет грамотно применять методы разработки синтаксических анализаторов, трансляторов, виртуальных машин

4. Критерии выставления оценок по результатам промежуточной аттестации по дисциплине

В 7 семестре результаты промежуточной аттестации определяются оценками «отлично», «хорошо», «удовлетворительно», «неудовлетворительно». Оценки «отлично», «хорошо», «удовлетворительно» означают успешное прохождение промежуточной аттестации.

Оценка «отлично» соответствует продвинутому уровню сформированности компетенции.

Оценка «хорошо» соответствует базовому уровню сформированности компетенции.

Оценка «удовлетворительно» соответствует пороговому уровню сформированности компетенции.

Оценка «неудовлетворительно» выставляется, если хотя бы одна компетенция не сформирована.