Минобрнауки России

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

(ФГБОУ ВО «ВГУ»)

УТВЕРЖДАЮ

Заведующий кафедрой

Каменский Михаил Игоревич

Кафедра функционального анализа и операторных уравнений

20.03.2025г.

РАБОЧАЯ ПРОГРАММА УЧЕБНОЙ ДИСЦИПЛИНЫ

Б1.В.06 Формальные грамматики и теория компиляторов

1. Код и наименование направления подготовки/специальности:

10.05.04 Информационно-аналитические системы безопасности

2. Профиль подготовки/специализация:

Автоматизация информационно-аналитической деятельности

3. Квалификация (степень) выпускника:

Специалитет

4. Форма обучения:

Очная

5. Кафедра, отвечающая за реализацию дисциплины: Кафедра функционального анализа и операторных уравнений

6. Составители программы:

Копытин Алексей Вячеславович, доцент кафедры функционального анализа и операторных уравнений

7. Рекомендована:

НМС математического факультета, протокол № 0500-03 от 18.03.2025г.

8. Учебный год:

2027-2028

9.Цели и задачи учебной дисциплины:

Цели изучения дисциплины:

- ознакомление с основными понятиями и методами использования теории формальных грамматик, освоение принципов построения формальных языков программирования, работы компиляторов, разработка алгоритма с использованием грамматических структур формальных грамматик.

Задачи учебной дисциплины:

- Изучение базовых понятий и принципов построения формальных грамматик и компиляторов, работы формальных грамматик и компиляторов, а также области их применения.

10. Место учебной дисциплины в структуре ООП:

дисциплина Формальные грамматики и теория компиляторов относится к части, формируемой участниками образовательных отношений, блока Б1.

11. Планируемые результаты обучения по дисциплине/модулю (знания, умения, навыки),соотнесенные с планируемыми результатами освоения образовательной программы (компетенциями выпускников) и индикаторами их достижения:

Код и название компетенции	Код и название индикатора компетенции	Знания, умения, навыки
ПК-2 Способен организовывать работы по выполнению в информационно-аналитических системах требований защиты информации ограниченного доступа	ПК-2.1 Способен анализировать безопасность информации с помощью формальных моделей	Знать: способы анализа безопасности информации с помощью формальных моделей. Уметь: анализировать безопасность информации с помощью формальных моделей. Владеть: способами анализа безопасности информации с помощью формальных моделей.

12. Объем дисциплины в зачетных единицах/час: 3/108

Форма промежуточной аттестации:

Зачет

13. Трудоемкость по видам учебной работы

Вид учебной работы	Семестр 5	Семестр 6	Всего
Аудиторные занятия	0	48	48
Лекционные занятия		16	16
Практические занятия		32	32
Лабораторные занятия			0
Самостоятельная работа	0	60	60
Курсовая работа			0
Промежуточная аттестация	0	0	0
Часы на контроль			0
Всего	0	108	108

13.1. Содержание дисциплины

п/п	Наименование раздела дисциплины	Содержание раздела дисциплины	Реализация раздела дисциплины с помощью онлайнкурса, ЭУМК
1	Лекции		

п/п	Наименование раздела дисциплины	Содержание раздела дисциплины	Реализация раздела дисциплины с помощью онлайнкурса, ЭУМК
1.1	Реализация лексических и синтаксических анализаторов, применение грамматик для описания синтаксиса формальных языков (неформальное введение в грамматики)	Использование аппарата грамматик для реализации рекурсивных нисходящих синтаксических анализаторов. Демонстрация использования грамматик для проектирования и реализации модуля вычисления математических выражений, принципы формального перевода правил грамматики в код методов на языке программирования. Доработка модуля до простейшего интерпретируемого языка с помощью модификации грамматики и последующей модификации модуля в соответствии с изменениями в грамматике.	https://edu.vsu.ru/course/view.php?id=29819
1.2	Базовая структура транслятора	Структуры данных в трансляторе: ASTдеревья, таблицы идентификаторов, промежуточные представления транслируемой программы. Базовые модули интерпретатора/компилятора. Указания студентам для выполнения практических заданий.	https://edu.vsu.ru/course/view.php?id=29819

1.3	Инструменты для автоматизации построения анализаторов, введение в Antlr	Доработка модуля до полноценного интерпретируемого языка, демонстрация сложностей, которые возникают при разработки синтаксического анализатора без использования соответствующих инструментов. Обзор инструментов для построения синтаксических анализаторов (Flex/Bizon, JavaCC, Antlr). Введение в Antlr: возможности, составные части, принцип работы. Грамматики Antlr: лексические и синтаксические правила, управление построением AST-деревьев. Разбор грамматики для реализованного ранее языка.	https://edu.vsu.ru/course/view.php?id=29819
1.4	Основы генерации кода	Формальное сопоставление конструкций ASTдерева инструкциям целевой платформы на примере простейшего вычислителя. Разбор примеров. Структура модуля генерации кода.	https://edu.vsu.ru/course/view.php?id=29819

п/п	Наименование раздела дисциплины	Содержание раздела дисциплины	Реализация раздела дисциплины с помощью онлайнкурса, ЭУМК
1.5	Генерация байткода .NET Framework	Краткий обзор языка MSIL и архитектуры виртуальной машины .NET Framework. Принципы трансляции в MSIL (выделение памяти и т.п.) Генерация кода MSIL для основных типов узлов ASTдерева (арифметические операции, вызов функций, условные операторы, циклы). Разбор примеров.	https://edu.vsu.ru/course/view.php?id=29819
1.6	Генерация байткода Java	Краткий обзор Java байт-кода и архитектуры виртуальной машины Java. Принципы трансляции в Java байт-код (выделение памяти и т.п.) Генерация байт-кода для основных типов узлов AST-дерева (арифметические операции, вызов функций, условные операторы, циклы). Разбор примеров.	https://edu.vsu.ru/course/view.php?id=29819

1.7	Генерация кода для платформы x86	Краткий обзор архитектуры x86. Принципы генерации исполняемого кода под x86: возникающие сложности и варианты их преодоления. Генерация кода для основных типов узлов AST-дерева (арифметические операции, вызов функций, условные операторы, циклы). Разбор примеров.	https://edu.vsu.ru/course/view.php?id=29819
2	Практические занятия		
2.1	Реализация лексических и синтаксических анализаторов, применение грамматик для описания синтаксиса формальных языков (неформальное введение в грамматики)	Использование аппарата грамматик для реализации рекурсивных нисходящих синтаксических анализаторов. Демонстрация использования грамматик для проектирования и реализации модуля вычисления математических выражений, принципы формального перевода правил грамматики в код методов на языке программирования. Доработка модуля до простейшего интерпретируемого языка с помощью модификации грамматики и последующей модификации модуля в соответствии с изменениями в грамматике.	https://edu.vsu.ru/course/view.php?id=29819
2.2	Базовая структура транслятора	Структуры данных в трансляторе: ASTдеревья, таблицы идентификаторов, промежуточные представления транслируемой программы. Базовые модули интерпретатора/компилятора. Указания студентам для выполнения практических заданий.	https://edu.vsu.ru/course/view.php?id=29819

п/п	Наименование раздела дисциплины	Содержание раздела дисциплины	Реализация раздела дисциплины с помощью онлайнкурса, ЭУМК

2.3	Инструменты для автоматизации построения анализаторов, введение в Antlr	Доработка модуля до полноценного интерпретируемого языка, демонстрация сложностей, которые возникают при разработки синтаксического анализатора без использования соответствующих инструментов. Обзор инструментов для построения синтаксических анализаторов (Flex/Bizon, JavaCC, Antlr). Введение в Antlr: возможности, составные части, принцип работы. Грамматики Antlr: лексические и синтаксические правила, управление построением AST-деревьев. Разбор грамматики для реализованного ранее языка.	https://edu.vsu.ru/course/view.php?id=29819
2.4	Основы генерации кода	Формальное сопоставление конструкций ASTдерева инструкциям целевой платформы на примере простейшего вычислителя. Разбор примеров. Структура модуля генерации кода.	https://edu.vsu.ru/course/view.php?id=29819
2.5	Генерация байткода .NET Framework	Краткий обзор языка MSIL и архитектуры виртуальной машины .NET Framework. Принципы трансляции в MSIL (выделение памяти и т.п.) Генерация кода MSIL для основных типов узлов ASTдерева (арифметические операции, вызов функций, условные операторы, циклы). Разбор примеров.	https://edu.vsu.ru/course/view.php?id=29819
2.6	Генерация байткода Java	Краткий обзор Java байт-кода и архитектуры виртуальной машины Java. Принципы трансляции в Java байт-код (выделение памяти и т.п.) Генерация байт-кода для основных типов узлов AST-дерева (арифметические операции, вызов функций, условные операторы, циклы). Разбор примеров.	https://edu.vsu.ru/course/view.php?id=29819
2.7	Генерация кода для платформы x86	Краткий обзор архитектуры x86. Принципы генерации исполняемого кода под x86: возникающие сложности и варианты их преодоления. Генерация кода для основных типов узлов AST-дерева (арифметические операции, вызов функций, условные операторы, циклы). Разбор примеров.	https://edu.vsu.ru/course/view.php?id=29819

3	Лабораторные работы		
п/п	Наименование раздела дисциплины	Содержание раздела дисциплины	Реализация раздела дисциплины с помощью онлайнкурса, ЭУМК
3.1	нет		

13.2.Темы (разделы) дисциплины и виды занятий

№ п/п	Наименование темы (раздела)	Лекционные занятия	Практические занятия	Лабораторные занятия	Самостоятельная работа	Всего
1	Обзор предметной области				3	3
2	Фазы трансляции программ				3	3
3	Реализация лексических и синтаксических анализаторов, применение грамматик для описания синтаксиса формальных языков (неформальное введение в грамматики)	3	5		6	14

4	Базовая структура транслятора	1	4		4	9
5	Инструменты для автоматизации построения анализаторов, введение в Antlr	5	7		8	20
6	Элементы теории языков (математический подход)				3	3
№ п/п	Наименование темы (раздела)	Лекционные занятия	Практические занятия	Лабораторные занятия	Самостоятельная работа	Всего
7	LL(k)-грамматики				4	4
8	LR(k)грамматики				4	4
9	Основы генерации кода	1	2		3	6
10	Генерация байткода .NET Framework	2	4		6	12
11	Генерация байткода Java	2	5		6	13

12	Генерация кода для платформы x86	2	5		6	13
13	Основы оптимизации кода при компиляции (обзорно)				4	4
		16	32	0	60	108

14. Методические указания для обучающихся по освоению дисциплины

Рекомендуется работа с конспектами лекций, презентационным материалом, выполнение всех лабораторных и контрольных работ.

15. Перечень основной и дополнительной литературы, ресурсов интернет, необходимыхдля освоения дисциплины

№ п/п	Источник
1	Ахо А. Компиляторы: принципы, технологии и инструменты / А.Ахо, Р. Сети, Дж. Ульман : Пер. с англ. – М и др..:Вильямс, 2008. – 767 с.
2	Соломатин Д.И. Основы синтаксического разбора, построение синтаксических анализаторов - Учебно-методическое пособие для вузов / Д.И.Соломатин, А.В. Копытин, А.И. Другалев - ВГУ, 2014 - 57 с.

б) дополнительная литература:

№ п/п	Источник
1	Грис. Конструирование компиляторов/ Грис. – М. : Мир,1980.
№ п/п	Источник
2	Льюис Ф. Теоретические основы проектирования компиляторов/ Ф. Льюис, Д. Розенкранц, Р. Стирнз. – М. : Мир, 1987.
3	Хантер Р. Проектирование и конструирование компиляторов / Р. Хантер. – М.:Мир, 1989.

4	Ахо А. Теория синтаксического анализа, перевода и компиляции/ А. Ахо, Дж.Ульман. – М.: Мир, 1978.

в) информационные электронно-образовательные ресурсы:

№ п/п	Источник
1	Вирт Н. Построение компиляторов [Электронный ресурс] : . — Электрон. дан. — М. : ДМК Пресс, 2010. — 186 с. — Режим доступа: http://e.lanbook.com/books/element.php?pl1_id=1262
2	Залогова, Л.А. Разработка Паскаль-компилятора [Электронный ресурс] : . — Электрон. дан. — М. : "Лаборатория знаний" (ранее "БИНОМ. Лаборатория знаний"), 2014. — 185 с. — Режим доступа: http://e.lanbook.com/books/element.php?pl1_id=66125

16. Перечень учебно-методического обеспечения для самостоятельной работы

№ п/п	Источник
1	Ахо А. Компиляторы: принципы, технологии и инструменты / А.Ахо, Р. Сети, Дж. Ульман : Пер. с англ. – М и др..:Вильямс, 2008. – 767 с.
2	Соломатин Д.И. Основы синтаксического разбора, построение синтаксических анализаторов - Учебно-методическое пособие для вузов / Д.И.Соломатин, А.В. Копытин, А.И. Другалев - ВГУ, 2014 - 57 с.

17. Образовательные технологии, используемые при реализации учебной дисциплины,включая дистанционные образовательные технологии (ДОТ), электронное обучение (ЭО), смешанное обучение):

OpenJDK - беплатен

Среда разработки NetBeans или Intellij IDEA (академическая лицензия или версия Community) бесплатны

Python версии 3.5 или выше с установленными дополнительными библиотеками (возможен вариант в виде дистрибутива Anaconda) - бесплатен

Среда разработки PyCharm (академическая лицензия или версия Community) - бесплатна

18. Материально-техническое обеспечение дисциплины:

Специализированная мебель, маркерная доска, персональные компьютеры.

19. Оценочные средства для проведения текущей и промежуточной аттестаций

Порядок оценки освоения обучающимися учебного материала определяется содержанием следующих разделов дисциплины:

№ п/п	Разделы дисциплины (модули)	Код компетенции	Код индикатора	Оценочные средства для текущей аттестации
1	1. Реализация лексических и синтаксических анализаторов, применение грамматик для описания синтаксиса формальных языков (неформальное введение в грамматики) 2. Базовая структура транслятора 3. Инструменты для автоматизации построения анализаторов, введение в Antlr 4. Основы генерации кода 5. Генерация байт-кода .NET Framework 6. Генерация байт-кода Java 7. Генерация кода для платформы x86	ПК-2	ПК-2.1	Практическое задание из пункта 20.1 (контроль и оценка этапов выполнения)

Промежуточная аттестация

Форма контроля - Зачет

Оценочные средства для промежуточной аттестации

Для оценивания результатов обучения на зачете используются следующие содержательные показатели (формулируется с учетом конкретных требований дисциплины):

1) знание теоретических основ учебного материала, основных определений, понятий ииспользуемой терминологии;

2) умение проводить обоснование и представление основных теоретических и практических результатов (теорем, алгоритмов, методик) с использованием математических выкладок, блок-схем, структурных схем и стандартных описаний к ним;

3) умение связывать теорию с практикой, иллюстрировать ответ примерами, в том числе, собственными, умение выявлять и анализировать основные закономерности, полученные, в том числе, в ходе выполнения лабораторно-практических заданий;

4) умение обосновывать свои суждения и профессиональную позицию по излагаемому вопросу;

5) владение навыками программирования и экспериментирования в рамках выполняемых лабораторных заданий;

Различные комбинации перечисленных показателей определяют критерии оценивания результатов обучения (сформированности компетенций) на зачете:

– высокий (углубленный) уровень сформированности компетенций;
– повышенный (продвинутый) уровень сформированности компетенций; – пороговый (базовый) уровень сформированности компетенций.

20 Типовые оценочные средства и методические материалы, определяющие процедурыоценивания

20.1 Текущий контроль успеваемости

Контроль успеваемости по дисциплине осуществляется с помощью контроля выполнения обязательных практических заданий. Перечень заданий:

1. Реализация учебного компилятора для подмножества языка C в MSIL байт-код (по этапам:

синтаксический анализ, семантичский анализ, кодогенерация)

2. Реализация учебного компилятора для подмножества языка C в Java байт-код (по этапам: синтаксический анализ, семантичский анализ, кодогенерация)
3. Реализация учебного компилятора для подмножества языка C в промежуточный код IR LLVM (поэтапам: синтаксический анализ, семантичский анализ, кодогенерация)
4. Реализация учебного компилятора для подмножества языка Pascal в MSIL байт-код (по этапам:

синтаксический анализ, семантичский анализ, кодогенерация)

5. Реализация учебного компилятора для подмножества языка Pascal в Java байт-код (по этапам: синтаксический анализ, семантичский анализ, кодогенерация)
6. Реализация учебного компилятора для подмножества языка Pascal в промежуточный код IR LLVM

(по этапам: синтаксический анализ, семантичский анализ, кодогенерация)

7. Реализация учебного компилятора для подмножества языка Go в MSIL байт-код (по этапам:

синтаксический анализ, семантичский анализ, кодогенерация)

8. Реализация учебного компилятора для подмножества языка Go в Java байт-код (по этапам:

синтаксический анализ, семантичский анализ, кодогенерация)

9. Реализация учебного компилятора для подмножества языка Go в промежуточный код IR LLVM (поэтапам: синтаксический анализ, семантичский анализ, кодогенерация)
10. Реализация учебного компилятора для подмножества языка Swift в MSIL байт-код (по этапам:

синтаксический анализ, семантичский анализ, кодогенерация)

11. Реализация учебного компилятора для подмножества языка Swift в Java байт-код (по этапам: синтаксический анализ, семантичский анализ, кодогенерация)
12. Реализация учебного компилятора для подмножества языка Swift в промежуточный код IR LLVM

(по этапам: синтаксический анализ, семантичский анализ, кодогенерация)

13. Реализация транслятора подмножества языка Python в JavaScript
14. Реализация прототипа высокоуовней виртуальной машины для JavaScript и компилятора JavaScript под эту машину
15. Реализация транслятора подмножества JavaScript в Python
16. Прототип модуля исполнения SQL-запросов

20.2 Промежуточная аттестация

Перечень вопросов к зачету:

1. Общее устройство компиляторов, фазы компиляции (обзорно).

2. Методики создания компиляторов: раскрутка, кросс-компиляция, использование виртуальныхмашин.
3. Понятие объектного модуля, сборка исполняемых файлов (линковка).
4. Лексический анализ, реализация лексического анализатора.
5. Синтаксический анализ, применение грамматик реализация синтаксическо анализатора. Дереворазбора и AST-дерево.
6. Внутреннее представление разбираемых программ. Понятие входной строки, токенов, ASTдерева, таблиц идентификаторов.
7. Семантический и видозависимый анализ, вычисление типов выражений. Реализациясемантического анализатора.
8. Разбор математичеких выражений (в качестве практического примера).
9. Разбор XML (в качестве практического примера).
10. Разбор JSON (в качестве практического примера).
11. Генераторы синтаксисечких анализаторов (обзорно).
12. Применение ANTLR для построения анализаторов, структура грамматики ANTLR, структуратранслятора с использованием ANTLR.
13. Возможности ANTLR для построения AST-деревьев.
14. Формальные языки, классификация языков.
15. Математическое определение порождающей грамматики, типы грамматик, примеры грамматик.
16. Понятие выводимости, формальное определение языка.
17. Разбор цепочек, дерево вывода, понятие неоднозначности грамматик и языков.
18. LL(k)-грамматики. Множества FIRST и FOLLOW, алгоритм построение для k=1.
19. Алгоритм построения управляющей таблицы для LL(1)-разбора. Алгоритм разбора строки спомощью управляющей таблицы.
20. LR(k)-грамматики. Алгоритм построения таблицы Action и Goto. Алгоритм разбора Shift/Reduce.21. Формальное сопоставление конструкций AST-дерева инструкциям целевой платформы на примере простейшего вычислителя. Структура модуля генерации кода.
22. Генерация кода для стековой и регистровой машины, сравнение.
23. Краткий обзор языка MSIL и архитектуры виртуальной машины .NET Framework.
24. Генерация кода MSIL для основных типов узлов AST-дерева.
25. Краткий обзор Java байт-кода и архитектуры виртуальной машины Java.
26. Генерация Java байт-кода для основных типов узлов AST-дерева.
27. Принципы генерация кода для регистровых архитектур. Двухадресный и трехадресный код.
28. Генерация кода для x86.
29. Сложности трансляции кода блочных языков (с произвольной вложенностью блоков процедур/функций), используемые решения.
30. Оптимизация кода в процессе компиляции (обзорно).

20.3 Фонд оценочных средств сформированности компетенций студентов, рекомендуемый для проведения диагностических работ

1) закрытые задания (тестовые, средний уровень сложности):

1) Объединение линейных языков

(1) является линейным языком

(2) не является линейным языком

(3) не имеет смысла, как действие

2) Автоматы с магазинной памятью соответствуют

(1) контекстно-свободным грамматикам

(2) праволинейным грамматикам

(3) как контекстно-свободным, так и праволинейным грамматикам

3) Используемые в приложениях формальные языки

(1) являются конечными

(2) являются бесконечными

(3) как правило, не определяются понятием конечности

4) Контекстно-свободным грамматикам соответствуют

(1) конечные автоматы

(2) автоматы с магазинной памятью

(3) бесконечные детерминированные автоматы

5)Процесс нахождения дерева вывода слова в заданной контекстно-свободной грамматике называется

(1) синтаксическим анализом

(2) синтаксической интерполяцией

(3) синтаксическим разбором

2) открытые задания (тестовые, повышенный уровень сложности):

1) Вставьте два слова. К наиболее распространенным способам конечного задания формального языка относят … и …

Ответ: грамматики, автоматы

2) Вставьте слово. Объединение линейных языков является … языком

Ответ: линейным

3) Вставьте слово. Используемые в приложениях формальные языки являются ….

Ответ: бесконечными

4) Вставьте два слова. Контекстно-свободным грамматикам соответствуют автоматы с …

Ответ: магазинной памятью

5) Вставьте слово. Если соответствующее отношение взаимозаменяемости разбивает множество всех слов рассматриваемого алфавита на конечное число классов эквивалентности, то язык является …

Ответ: автоматным

Критерии и шкалы оценивания заданий ФОС:

1) Задания закрытого типа (выбор одного варианта ответа, верно/неверно):

- 1 балл – указан верный ответ;
- 0 баллов – указан неверный ответ.

2) Задания закрытого типа (множественный выбор):

- 2 балла – указаны все верные ответы;
- 0 баллов — указан хотя бы один неверный ответ.

3) Задания закрытого типа (на соответствие):

- 2 балла – все соответствия определены верно;
- 0 баллов – хотя бы одно сопоставление определено неверно.

4) Задания открытого типа (короткий текст):

- 2 балла – указан верный ответ;
- 0 баллов – указан неверный ответ.

5) Задания открытого типа (число):

- 2 балла – указан верный ответ;

0 баллов – указан неверный ответ.

Задания раздела 20.3 рекомендуются к использованию при проведении диагностических работ с целью оценки остаточных результатов освоения данной дисциплины (знаний, умений, навыков).