


МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «ВГУ»)

УТВЕРЖДАЮ

Заведующий кафедрой
Программирования и информационных технологий


проф. Махортов С.Д,
05.03.2024

РАБОЧАЯ ПРОГРАММА УЧЕБНОЙ ДИСЦИПЛИНЫ

Б1.В.04 Теория компиляторов

1. Код и наименование направления подготовки/специальности:

09.03.04 Программная инженерия

2. Профиль подготовки/специализация:

Информационные системы и сетевые технологии

3. Квалификация (степень) выпускника: Бакалавр

4. Форма обучения: Очная

5. Кафедра, отвечающая за реализацию дисциплины:

Программирования и информационных технологий

6. Составители программы:

ст. преподаватель каф. ПиИТ Соломатин Дмитрий Иванович

e-mail: solomatin@cs.vsu.ru

факультет: Компьютерных наук

кафедра: Программирования и информационных технологий

7. Рекомендована:

НМС ф-та компьютерных наук, протокол № 5 от 05.03.2024

8. Учебный год: 2026/2027

Семестр(ы): 6

9. Цели и задачи учебной дисциплины:

Изучение студентами математических основ трансляции программ, принципов построения компиляторов, а также овладение практическими навыками реализации синтаксических анализаторов, интерпретаторов и трансляторов.

10. Место учебной дисциплины в структуре ООП:

Учебная дисциплина относится к вариативной части блока Б1.

Для успешного освоения дисциплины необходимы знания дискретной математики, архитектуры ЭВМ, а также практический опыт программирования на объектно-ориентированном языке программирования.

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

Код	Название компетенции	Код(ы)	Индикатор(ы)	Планируемые результаты обучения
ПК-3	Способен выполнять проектирование ПО	ПК-3.1	Разрабатывает и согласовывает архитектуру ПО с системным аналитиком.	Знать: архитектуру компиляторов, алгоритмические основы компиляторов Уметь: разрабатывать архитектуру компиляторов Владеть: навыками разработки архитектуры компиляторов
		ПК-3.2	Проектирует структуры данных и баз данных.	Знать: структуры данных, применяемых в реализации компиляторов Уметь: Применять структуры данных в реализации компиляторов Владеть: навыками применения структур данных, применяемых в реализации компиляторов
ПК-7	Способен разрабатывать компоненты инструментальных средств программирования	ПК-7.1	Разрабатывает исходный код и создает бинарные файлы инструментальных средств программирования	Знать: основы теории синтаксического анализа, этапы трансляции программ, принципы построения компиляторов и генерации исполняемого кода Уметь: описывать грамматики для формальных языков, использовать инструментальные средства для построения синтаксических анализаторов на основе грамматик, реализовать модули семантического анализа и кодогенерации для подмножества языка программирования
		ПК-7.2	Сопровождает инструментальные средства программирования	Владеть: базовыми практическими навыками разработки и сопровождения инструментальных средств программирования

ПК-13	Способен обрабатывать, анализировать и оформлять результаты исследований и разработок под руководством специалиста более высокой квалификации	ПК-13.1	Обрабатывает полученные результаты исследований с использованием стандартных методов (методик).	Знать: основы теории синтаксического анализа, этапы трансляции программ, принципы построения компиляторов и генерации исполняемого кода
		ПК-13.2	Применяет при обработке данных стандартное и оригинальное программное обеспечение.	Уметь: описывать грамматики для формальных языков, использовать инструментальные средства для построения синтаксических анализаторов на основе грамматик, реализовать модули семантического анализа и кодогенерации для подмножества языка программирования Владеть: базовыми практическими навыками разработки и сопровождения инструментальных средств программирования

12. Объем дисциплины в зачетных единицах/час. (в соответствии с уч. планом) – 2 / 72.

Форма промежуточной аттестации – Зачет

13. Виды учебной работы

Вид учебной работы	Трудоемкость			
	Всего	По семестрам		
		6 сем.	–	–
Аудиторные занятия	48	48	–	–
в том числе:	лекции	–	–	–
	практические	32	32	–
	лабораторные	16	16	–
Самостоятельная работа	24	24	–	–
в том числе: курсовая работа (проект)	–	–	–	–
Форма промежуточной аттестации (зачет – 0 час. / экзамен – 0 час.)	–	–	–	–
Итого:	72	72	–	–

13.1. Содержание дисциплины

№ п/п	Наименование раздела дисциплины	Содержание раздела дисциплины	Реализация раздела дисциплины с помощью онлайн-курса, ЭУМК
1. Лекции			
1.1	нет		
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.	

2.2	Базовая структура транслятора	Структуры данных в трансляторе: AST-деревья, таблицы идентификаторов, промежуточные представления транслируемой программы. Базовые модули интерпретатора/компилятора. Указания студентам для выполнения практических заданий.	
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/Bison, 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.	
2.4	Основы генерации кода	Формальное сопоставление конструкций AST-дерева инструкциям целевой платформы на примере простейшего вычислителя. Разбор примеров. Структура модуля генерации кода.	
2.5	Генерация байт-кода .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 AST-дерева (арифметические операции, вызов функций, условные операторы, циклы). Разбор примеров.	
2.6	Генерация байт-кода 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.	
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.	
3. Лабораторные работы			
3.1	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.	
3.2	Базовая структура транслятора	Структуры данных в трансляторе: AST-деревья, таблицы идентификаторов, промежуточные представления транслируемой программы. Базовые модули интерпретатора/компилятора. Указания студентам для выполнения практических заданий.	
3.3	Инструменты для автоматизации построения анализаторов, введение в Antlr	Доработка модуля до полноценного интерпретируемого языка, демонстрация сложностей, которые возникают при разработке синтаксического анализатора без использования соответствующих инструментов. Обзор инструментов для построения синтаксических анализаторов (Flex/Bison, 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.	

3.4	Основы генерации кода	Формальное сопоставление конструкций AST-дерева инструкциям целевой платформы на примере простейшего вычислителя. Разбор примеров. Структура модуля генерации кода.	
3.5	Генерация байт-кода .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 AST-дерева (арифметические операции, вызов функций, условные операторы, циклы). Разбор примеров.	
3.6	Генерация байт-кода 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.	
3.7	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.	

13.2. Темы (разделы) дисциплины и виды занятий

№ п/п	Наименование темы (раздела) дисциплины	Виды занятий (часов)				
		Лекции	Практические	Лабораторные	Самостоятельная работа	Всего
1	Обзор предметной области	–	–	–	1	1
2	Фазы трансляции программ	–	–	–	1	1
3	Реализация лексических и синтаксических анализаторов, применение грамматик для описания синтаксиса формальных языков (неформальное введение в грамматики)	–	5	3	2	10
4	Базовая структура транслятора	–	4	1	2	7
5	Инструменты для автоматизации построения анализаторов, введение в Antlr	–	7	5	2	14
6	Элементы теории языков (математический подход)	–	–	–	1	1
7	LL(k)-грамматики	–	–	–	2	2
8	LR(k)-грамматики	–	–	–	2	2
9	Основы генерации кода	–	2	1	1	4
10	Генерация байт-кода .NET Framework	–	4	2	2	8
11	Генерация байт-кода Java	–	5	2	2	9
12	Генерация кода для платформы x86	–	5	2	2	9
13	Основы оптимизации кода при компиляции (обзорно)	–	–	–	4	4
	Итого:	–	32	16	24	72

14. Методические указания для обучающихся по освоению дисциплины

Рекомендуется работа с конспектами лекций, презентационным материалом, выполнение всех лабораторных и контрольных работ, заданий текущей аттестации. Учебные и методические материалы по дисциплине размещены на сетевом диске, доступным на любом компьютере в локальной сети ФКН.

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

а) основная литература:

№ 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
3	Грис. Конструирование компиляторов/ Грис. – М. : Мир,1980.
4	Льюис Ф. Теоретические основы проектирования компиляторов/ Ф. Льюис, Д. Розенкранц, Р. Стирнз. – М. : Мир, 1987.
5	Хантер Р. Проектирование и конструирование компиляторов / Р. Хантер. – М.:Мир, 1989.
6	Ахо А. Теория синтаксического анализа, перевода и компиляции/ А. Ахо, Дж.Ульман. – М.: Мир, 1978.

в) информационные электронно-образовательные ресурсы (официальные ресурсы интернет):

№ п/п	Источник
7	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
8	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. Информационные технологии, используемые для реализации учебной дисциплины, включая программное обеспечение и информационно-справочные системы (при необходимости)

№ п/п	Наименование
1	OpenJDK - бесплатен
2	Среда разработки NetBeans или IntelliJ IDEA (академическая лицензия или версия Community) - бесплатны
3	Python версии 3.5 или выше с установленными дополнительными библиотеками (возможен вариант в виде дистрибутива Anaconda) - бесплатен
4	Среда разработки PyCharm (академическая лицензия или версия Community) - бесплатна

18. Материально-техническое обеспечение дисциплины:

№ п/п	Наименование
1	Мультимедийная лекционная аудитория (корп. 1а, ауд. № 479 или другая подходящая): рабочее место преподавателя: ПК-Intel-i3, проектор, видеоконмутатор, микрофон, аудиосистема, специализированная мебель: доски меловые 2 шт., столы и стулья/лавки в количестве, достаточном для размещения потока студентов; выход в Интернет, доступ к фондам учебно-методической документации и электронным изданиям.

2	Компьютерный класс (корп. 1а, ауд. № 382-385 или другие подходящие): ПК-Intel-i3 16 шт., специализированная мебель: доска маркерная 1 шт., столы и стулья в количестве, достаточном для размещения академической группы (подгруппы) студентов; выход в Интернет, доступ к фондам учебно-методической документации и электронным изданиям.
---	---

19. Оценочные средства для проведения текущей и промежуточной аттестаций

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

№ п/п	Наименование раздела дисциплины (модуля)	Компетенция(и)	Индикатор(ы) достижения компетенции	Оценочные средства
1	Обзор предметной области	ПК-3 ПК-7 ПК-13	ПК-3.1, ПК-3.2 ПК-7.1, ПК-7.2 ПК-13.1, ПК-13.2	Практическое задание из пункта 20.1 (контроль и оценка этапов выполнения)
2	Фазы трансляции программ	ПК-3 ПК-7 ПК-13	ПК-3.1, ПК-3.2 ПК-7.1, ПК-7.2 ПК-13.1, ПК-13.2	Практическое задание из пункта 20.1 (контроль и оценка этапов выполнения)
3	Реализация лексических и синтаксических анализаторов, применение грамматик для описания синтаксиса формальных языков (неформальное введение в грамматики)	ПК-3 ПК-7 ПК-13	ПК-3.1, ПК-3.2 ПК-7.1, ПК-7.2 ПК-13.1, ПК-13.2	Практическое задание из пункта 20.1 (контроль и оценка этапов выполнения)
4	Базовая структура транслятора	ПК-3 ПК-7 ПК-13	ПК-3.1, ПК-3.2 ПК-7.1, ПК-7.2 ПК-13.1, ПК-13.2	Практическое задание из пункта 20.1 (контроль и оценка этапов выполнения)
5	Инструменты для автоматизации построения анализаторов, введение в Antlr	ПК-3 ПК-7 ПК-13	ПК-3.1, ПК-3.2 ПК-7.1, ПК-7.2 ПК-13.1, ПК-13.2	Практическое задание из пункта 20.1 (контроль и оценка этапов выполнения)
6	Элементы теории языков (математический подход)	ПК-3 ПК-7 ПК-13	ПК-3.1, ПК-3.2 ПК-7.1, ПК-7.2 ПК-13.1, ПК-13.2	Практическое задание из пункта 20.1 (контроль и оценка этапов выполнения)
7	LL(k)-грамматики	ПК-3 ПК-7 ПК-13	ПК-3.1, ПК-3.2 ПК-7.1, ПК-7.2 ПК-13.1, ПК-13.2	Практическое задание из пункта 20.1 (контроль и оценка этапов выполнения)
8	LR(k)-грамматики	ПК-3 ПК-7 ПК-13	ПК-3.1, ПК-3.2 ПК-7.1, ПК-7.2 ПК-13.1, ПК-13.2	Практическое задание из пункта 20.1 (контроль и оценка этапов выполнения)
9	Основы генерации кода	ПК-3 ПК-7 ПК-13	ПК-3.1, ПК-3.2 ПК-7.1, ПК-7.2 ПК-13.1, ПК-13.2	Практическое задание из пункта 20.1 (контроль и оценка этапов выполнения)
10	Генерация байт-кода .NET Framework	ПК-3 ПК-7 ПК-13	ПК-3.1, ПК-3.2 ПК-7.1, ПК-7.2 ПК-13.1, ПК-13.2	Практическое задание из пункта 20.1 (контроль и оценка этапов выполнения)
11	Генерация байт-кода Java	ПК-3 ПК-7 ПК-13	ПК-3.1, ПК-3.2 ПК-7.1, ПК-7.2 ПК-13.1, ПК-13.2	Практическое задание из пункта 20.1 (контроль и оценка этапов выполнения)
12	Генерация кода для платформы x86	ПК-3 ПК-7 ПК-13	ПК-3.1, ПК-3.2 ПК-7.1, ПК-7.2 ПК-13.1, ПК-13.2	Практическое задание из пункта 20.1 (контроль и оценка этапов выполнения)
13	Основы оптимизации кода при компиляции (обзорно)	ПК-3 ПК-7 ПК-13	ПК-3.1, ПК-3.2 ПК-7.1, ПК-7.2 ПК-13.1, ПК-13.2	Практическое задание из пункта 20.1 (контроль и оценка этапов выполнения)

Промежуточная аттестация форма контроля – зачет	Перечень вопросов к зачету из пункта 20.2
--	--

20. Типовые оценочные средства и методические материалы, определяющие процедуры оценивания

20.1 Текущий контроль успеваемости

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

№ п/п	Задание
1	Реализация учебного компилятора для подмножества языка C в MSIL байт-код (по этапам: синтаксический анализ, семантический анализ, кодогенерация)
2	Реализация учебного компилятора для подмножества языка C в Java байт-код (по этапам: синтаксический анализ, семантический анализ, кодогенерация)
3	Реализация учебного компилятора для подмножества языка C в промежуточный код IR LLVM (по этапам: синтаксический анализ, семантический анализ, кодогенерация)
4	Реализация учебного компилятора для подмножества языка Pascal в MSIL байт-код (по этапам: синтаксический анализ, семантический анализ, кодогенерация)
5	Реализация учебного компилятора для подмножества языка Pascal в Java байт-код (по этапам: синтаксический анализ, семантический анализ, кодогенерация)
6	Реализация учебного компилятора для подмножества языка Pascal в промежуточный код IR LLVM (по этапам: синтаксический анализ, семантический анализ, кодогенерация)
7	Реализация учебного компилятора для подмножества языка Go в MSIL байт-код (по этапам: синтаксический анализ, семантический анализ, кодогенерация)
8	Реализация учебного компилятора для подмножества языка Go в Java байт-код (по этапам: синтаксический анализ, семантический анализ, кодогенерация)
9	Реализация учебного компилятора для подмножества языка Go в промежуточный код IR LLVM (по этапам: синтаксический анализ, семантический анализ, кодогенерация)
10	Реализация учебного компилятора для подмножества языка Swift в MSIL байт-код (по этапам: синтаксический анализ, семантический анализ, кодогенерация)
11	Реализация учебного компилятора для подмножества языка Swift в Java байт-код (по этапам: синтаксический анализ, семантический анализ, кодогенерация)
12	Реализация учебного компилятора для подмножества языка Swift в промежуточный код IR LLVM (по этапам: синтаксический анализ, семантический анализ, кодогенерация)
13	Реализация транслятора подмножества языка Python в JavaScript
14	Реализация прототипа высокоуровневой 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

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 и используемой 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.

Соотношение показателей, критериев и шкалы оценивания результатов обучения на зачете представлено в следующей таблице.

Критерии оценивания компетенций	Уровень сформированности компетенций	Шкала оценок
Студент владеет основными понятиями учебной дисциплины, может пояснить большинство принципов на примерах; сдал большую часть практических заданий.	Пороговый уровень	Зачет
Студент путается в основных понятиях учебной дисциплины, не может привести примеры; не сдал большую часть практических заданий.	–	Незачет

Перечень вопросов к зачету:

№ п/п	Вопрос
1	Общее устройство компиляторов, фазы компиляции (обзорно).
2	Методики создания компиляторов: раскрутка, кросс-компиляция, использование виртуальных машин.
3	Понятие объектного модуля, сборка исполняемых файлов (линковка).
4	Лексический анализ, реализация лексического анализатора.
5	Синтаксический анализ, применение грамматик реализация синтаксического анализатора. Дерево разбора и AST-дерево.
6	Внутреннее представление разбираемых программ. Понятие входной строки, токенов, AST-дерева, таблиц идентификаторов.
7	Семантический и видозависимый анализ, вычисление типов выражений. Реализация семантического анализатора.
8	Разбор математических выражений (в качестве практического примера).
9	Разбор XML (в качестве практического примера).
10	Разбор JSON (в качестве практического примера).
11	Генераторы синтаксических анализаторов (обзорно).
12	Применение ANTLR для построения анализаторов, структура грамматики ANTLR, структура транслятора 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 построения для k=1.
19	Алгоритм построения управляющей таблицы для LL(1)-разбора. Алгоритм разбора строки с помощью 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 генерации 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).