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,*
11.03.2022

РАБОЧАЯ ПРОГРАММА УЧЕБНОЙ ДИСЦИПЛИНЫ

Б1.В.ДВ.04.02 Теория компиляторов

1. Код и наименование направления подготовки/специальности:

09.04.02 Информационные системы и технологии

2. Профиль подготовки/специализация:

Системы прикладного искусственного интеллекта

3. Квалификация (степень) выпускника: магистр

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 № 03 от 25.02.2022

_____
_____
_____

8. Учебный год: 2023-2024 **Семестр(ы):** 3

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, является дисциплиной по выбору.

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-1	Способен организовывать работу программистов в группе по созданию системного ПО	ПК-1.1	Умеет выполнять декомпозицию поставленной задачи и распределение подзадач между программистами	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 Владеть: навыками коллективной работы над сложными проектами с применением формальных подходов к декомпозиции задачи на подзадачи
		ПК-1.2	Умеет определять процессы интеграции разработанных компонентов системного ПО	
		ПК-1.3	Умеет определять задачи для группы стандартов кодирования	
ПК-6	Способен управлять выпуском релизов ИС	ПК-6.1	Умеет определять состав и разрабатывать план выпуска релизов ИС	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 Владеть: навыками коллективной работы над сложными проектами с применением формальных подходов к декомпозиции задачи на подзадачи
		ПК-6.2	Умеет изменять план выпуска релизов ИС на основе одобренных запросов	

12. Объем дисциплины в зачетных единицах/час. (в соответствии с уч. планом) **–** 3 / 108.

Форма промежуточной аттестации – Зачет с оценкой

13. Виды учебной работы

Вид учебной работы		Трудоемкость			
		Всего	По семестрам		
			3 сем.	–	–
Аудиторные занятия		42	42	–	–
в том числе:	лекции	14	14	–	–
	практические	–	–	–	–
	лабораторные	28	28	–	–
Самостоятельная работа		66	66	–	–
в том числе: курсовая работа (проект)		–	–	–	–
Форма промежуточной аттестации (зачет – 0 час. / экзамен – 0 час.)		–	–	–	–
Итого:		108	108	–	–

13.1. Содержание дисциплины

№ п/п	Наименование раздела дисциплины	Содержание раздела дисциплины	Реализация раздела дисциплины с помощью онлайн-курса, ЭУМК
1. Лекции			
1.1	Обзор предметной области	Содержание курса. Критерии оценки. Материалы и источники информации. Терминология: транслятор, компилятор, интерпретатор. Формальное определение компилятора. Методики создания компиляторов: раскрутка, кросс-компиляция, использование виртуальных машин.	
1.2	Фазы трансляции программ	Лексический анализ, синтаксический анализ, семантический анализ, видозависимый анализ, оптимизация, генерация кода. Понятие объектного модуля, сборка исполняемых файлов (линковка).	
1.3	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.	
1.4	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.	
1.5	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.	

1.6	Элементы теории языков (математический подход)	Понятие формального языка, способы задания формальных языков. Математическое определение порождающей грамматики. Соглашения об обозначениях (терминалы, нетерминалы, цепочки символов и т.п.). Примеры грамматик. Понятие выводимости, формальное определение языка. Классификация языков по Хомскому. Разбор цепочек, дерево вывода, понятие неоднозначности грамматик и языков.	
1.7	LL(k)-грамматики	Понятие LL(k)-грамматик. Множества FIRST и FOLLOW, алгоритм построение для k=1. Алгоритм построения управляющей таблицы для LL(1)-разбора. Алгоритм разбора строки с помощью управляющей таблицы.	
1.8	LR(k)-грамматики	Понятие LR(k)-грамматик. Алгоритм построения таблицы Action и Goto. Алгоритм разбора Shift/Reduce.	
1.9	Основы генерации кода	Формальное сопоставление конструкций AST-дерева инструкциям целевой платформы на примере простейшего вычислителя. Разбор примеров. Структура модуля генерации кода.	
1.10	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.	
1.11	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.	
1.12	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.	
1.13	Основы оптимизации кода при компиляции (обзорно)	Виды оптимизирующих преобразований. Фазы компиляции и внутренние представления, на которых выполняются оптимизации. Возможные зависимости между различными оптимизирующими преобразованиями.	
2. Практические занятия			
2.1	нет		
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 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.	
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	Практи-ческие	Лабора-торные	Самостоя-тельная работа	Всего
1	Обзор предметной области	1	–	–	4	5
2	Фазы трансляции программ	1	–	–	4	5
3	Реализация лексических и синтаксических анализаторов, применение грамматик для описания синтаксиса формальных языков (неформальное введение в грамматики)	1	–	4	6	11
4	Базовая структура транслятора	1	–	4	4	9
5	Инструменты для автоматизации построения анализаторов, введение в Antlr	2	–	6	6	14
6	Элементы теории языков (математический подход)	1	–	–	4	5
7	LL(k)-грамматики	1	–	–	4	5
8	LR(k)-грамматики	1	–	–	4	5
9	Основы генерации кода	1	–	2	6	9
10	Генерация байт-кода .NET Framework	2	–	4	6	12
11	Генерация байт-кода Java	1	–	4	6	11
12	Генерация кода для платформы x86	2	–	4	6	12

13	Основы оптимизации кода при компиляции (обзорно)	1	–	–	6	7
	Итого:	16	–	28	66	108

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 - беплатен
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, видеокоммутатор, 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)	Компетен-ция(и)	Индикатор(ы) достижения компетенции	Оценочные средства
1	Обзор предметной области	ПКВ-1 ПКВ-6	ПКВ-1.1, ПКВ-1.2, ПКВ-1.3 ПКВ-6.1, ПКВ-6.2	Практическое задание из пункта 20.1 (контроль и оценка этапов выполнения)
2	Фазы трансляции программ	ПКВ-1 ПКВ-6	ПКВ-1.1, ПКВ-1.2, ПКВ-1.3 ПКВ-6.1, ПКВ-6.2	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)	ПКВ-1 ПКВ-6	ПКВ-1.1, ПКВ-1.2, ПКВ-1.3 ПКВ-6.1, ПКВ-6.2	Практическое задание из пункта 20.1 (контроль и оценка этапов выполнения)
4	Базовая структура транслятора	ПКВ-1 ПКВ-6	ПКВ-1.1, ПКВ-1.2, ПКВ-1.3 ПКВ-6.1, ПКВ-6.2	Практическое задание из пункта 20.1 (контроль и оценка этапов выполнения)
5	Инструменты для автоматизации построения анализаторов, введение в Antlr	ПКВ-1 ПКВ-6	ПКВ-1.1, ПКВ-1.2, ПКВ-1.3 ПКВ-6.1, ПКВ-6.2	Практическое задание из пункта 20.1 (контроль и оценка этапов выполнения)
6	Элементы теории языков (математический подход)	ПКВ-1 ПКВ-6	ПКВ-1.1, ПКВ-1.2, ПКВ-1.3 ПКВ-6.1, ПКВ-6.2	Практическое задание из пункта 20.1 (контроль и оценка этапов выполнения)
7	LL(k)-грамматики	ПКВ-1 ПКВ-6	ПКВ-1.1, ПКВ-1.2, ПКВ-1.3 ПКВ-6.1, ПКВ-6.2	Практическое задание из пункта 20.1 (контроль и оценка этапов выполнения)
8	LR(k)-грамматики	ПКВ-1 ПКВ-6	ПКВ-1.1, ПКВ-1.2, ПКВ-1.3 ПКВ-6.1, ПКВ-6.2	Практическое задание из пункта 20.1 (контроль и оценка этапов выполнения)

9	Основы генерации кода	ПКВ-1 ПКВ-6	ПКВ-1.1, ПКВ-1.2, ПКВ-1.3 ПКВ-6.1, ПКВ-6.2	Практическое задание из пункта 20.1 (контроль и оценка этапов выполнения)
10	Генерация байт-кода .NET Framework	ПКВ-1 ПКВ-6	ПКВ-1.1, ПКВ-1.2, ПКВ-1.3 ПКВ-6.1, ПКВ-6.2	Практическое задание из пункта 20.1 (контроль и оценка этапов выполнения)
11	Генерация байт-кода Java	ПКВ-1 ПКВ-6	ПКВ-1.1, ПКВ-1.2, ПКВ-1.3 ПКВ-6.1, ПКВ-6.2	Практическое задание из пункта 20.1 (контроль и оценка этапов выполнения)
12	Генерация кода для платформы x86	ПКВ-1 ПКВ-6	ПКВ-1.1, ПКВ-1.2, ПКВ-1.3 ПКВ-6.1, ПКВ-6.2	Практическое задание из пункта 20.1 (контроль и оценка этапов выполнения)
13	Основы оптимизации кода при компиляции (обзорно)	ПКВ-1 ПКВ-6	ПКВ-1.1, ПКВ-1.2, ПКВ-1.3 ПКВ-6.1, ПКВ-6.2	Практическое задание из пункта 20.1 (контроль и оценка этапов выполнения)
Промежуточная аттестация форма контроля – зачет с оценкой				Перечень вопросов к зачету с оценкой 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, 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 (по этапам: 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 (по этапам: 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

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.

Для оценивания результатов обучения на зачете с оценкой используется 4-балльная шкала: «отлично», «хорошо», «удовлетворительно», «неудовлетворительно».

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

Критерии оценивания компетенций	Уровень сформированности компетенций	Шкала оценок
Студент владеет основными понятиями учебной дисциплины, может пояснить большинство принципов на примерах; вовремя сдал все практические задания, которые выполнены на высоком уровне, без явных ошибок.	Повышенный уровень	Отлично
Студент владеет основными понятиями учебной дисциплины, однако в ответах на некоторые вопросы допускает неточности; сдал все практические задания, однако к некоторым решениям студента у преподавателя есть замечания.	Базовый уровень	Хорошо
Студент знает основные определения из учебной дисциплины, однако пояснить многие понятия на примерах затрудняется; сдал большую часть практических заданий, однако продемонстрированные решения содержат существенные ошибки.	Пороговый уровень	Удовлетворительно
Студент путается в основных понятиях учебной дисциплины, не может привести примеры; не сдал большую часть практических заданий.	–	Неудовлетвори-тельно

Перечень вопросов к зачету с оценкой:

№ 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 синтаксическо 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 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, 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 построение 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 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).