Федеральное государственное автономное образовательное учреждение высшего образования «Московский физико-технический институт (национальный исследовательский университет)»

УТВЕРЖДЕНО
Директор физтех-школы радиотехники и компьютерных технологий
Е.А. Белянко

Рабочая программа дисциплины (модуля)

по дисциплине: Современные методы разработки компиляторов
по направлению: Информатика и вычислительная техника
профиль подготовки: Компьютерные технологии и вычислительная техника
Физтех-школа Радиотехники и Компьютерных Технологий
кафедра микропроцессорных технологий в интеллектуальных системах управления
курс: 4
квалификация: бакалавр

Семестры, формы промежуточной аттестации:
7 (осенний) - Дифференцированный зачет
8 (весенний) - Экзамен

Аудиторных часов: 60 всего, в том числе:
лекции: 30 час.
семинары: 30 час.
лабораторные занятия: 0 час.

Самостоятельная работа: 45 час.

Подготовка к экзамену: 30 час.

Всего часов: 135, всего зач. ед.: 3

Программу составил: С.А. Лисицын

Программа обсуждена на заседании кафедры микропроцессорных технологий в интеллектуальных системах управления 29.04.2021

Аннотация

В курсе рассматриваются ключевые понятия и методы оптимизирующей трансляции кода, а также тестирования и верификации компилятора. Прежде всего, рассматривается процесс компиляции вцелом, после чего производится детальный разбор работы фаз компилятора. Отдельные лекции и семинары посвящены тестированию и верификации работы трансляторов.

Курс содержит в себе практические задания, основанные на использовании современных компиялторных архитектур. Для успешного освоения слушателю желательно знать курсы по программироанию: «Введение в программирование», «Алгоритмы и структуры данных», «Архитектура компьютера», «Операционные системы».

1. Цели и задачи

Цель дисциплины

Целью курса является представление теоретических принципов и практических подходов разработки компиляторов и их составных частей, оптимизаций, используемых в компиляторах, методологии и практическое применение тестирования и верификации трансляторов.

Задачи дисциплины

- Понимание построения компилятора, принципов использования генераторов синтаксических анализаторов (Flex/Bison), устройства современных компиляторов (LLVM и других компипляторов).
- Програмной реализации построения промежуточного представления компилятора с использованием грамматики языка;
- Программной реализации сложных алгоритмов оптимизаций програмного кода;
- Програмной реализации генератора бинарного кода компиялтора;
- Тестирование и верификация компилятора;
- Анализ производительности низкоуровнего кода програмного обеспечения (ПО).

2. Перечень формируемых компетенций

Освоение дисциплины направлено на формирование следующих компетенций:

Код и наименование компетенции	Индикаторы достижения компетенции
УК-1 Способен осуществлять поиск, критический анализ и синтез информации, применять системный подход для решения поставленных задач	УК-1.2 Находит, критически анализирует и выбирает информацию, необходимую для решения поставленной задачи
ОПК-2 Способен использовать современные информационные технологии и программные средства при решении задач профессиональной деятельности, соблюдая требования информационной безопасности	ОПК-2.1 Способен применять современные вычислительную технику и сервисы сети Интернет в области (сфере) профессиональной деятельности
ОПК-3 Способен составлять и оформлять научные и (или) технические (технологические, инновационные) отчеты (публикации, проекты)	ОПК-3.3 Владеет методами визуального и графического представления результатов научной (научно-технической, инновационной технологической) деятельности в виде отчетов, научных публикаций
ПК-1 Способен ставить, формализовывать и решать задачи, в том числе разрабатывать и исследовать математические модели изучаемых явлений и процессов, системно анализировать научные проблемы, получать новые научные результаты	ПК-1.1 Способен находить, анализировать и обобщать информацию об актуальных результатах исследований в рамках тематической области своей профессиональной деятельности
	ПК-1.2 Способен выдвигать гипотезы, строить математические модели для описания изучаемых явлений и процессов, оценивать качество разработанной модели
	ПК-1.3 Способен применять теоретические и (или) экспериментальные методы исследований к конкретной научной задаче и интерпретировать полученные результаты

ПК-2 Способен самостоятельно или в качестве члена (руководителя) малого коллектива организовывать и проводить научные исследования и их апробацию	ПК-2.1 Знает принципы построения научной работы, методы сбора и анализа полученного материала, способы аргументации
	ПК-2.2 Способен планировать и проводить научные исследования самостоятельно или в качестве члена (руководителя) малого научного коллектива
	ПК-2.3 Способен проводить апробацию результатов научно-исследовательской работы посредством публикации научных статей и участия в конференциях

3. Перечень планируемых результатов обучения по дисциплине (модулю)

В результате освоения дисциплины обучающиеся должны

знать:

- основные методы программирования, применяемые при разработке сложных компиляторных систем;
- методы програмной реализации сложных алгоритмов оптимизаций кода;
- методы тестирования и верификации трансляторов

уметь:

- разрабатывать и реализовывать компилятор, отдельные его части и системы его тестирования и верификации

владеть:

- теоретическими знаниями в области компиляторов, оптимизаций и тестирования трансляторов.

4. Содержание дисциплины (модуля), структурированное по темам (разделам) с указанием отведенного на них количества академических часов и видов учебных занятий

4.1. Разделы дисциплины (модуля) и трудоемкости по видам учебных занятий

№	Тема (раздел) дисциплины	Трудоемкость по видам учебных занятий, включая самостоятельную работу, час.			
		Лекции	Семинары	Лаборат. работы	Самост. работа
1	Введение в компиляцию. Обзор архитектур существующих компиялторов.	1	1		1
2	Обзор принципов работы компиляторного фронтенда.	1	1		1
3	Генераторы синтаксических анализаторов Flex/Bison.	1	1		1
4	Обзор принципов работы компиляторного мидленда. Промежуточное представление компилятора. LLVM IR.	3	3		3
5	LLVM API. Програмная генерация прромежуточного представляения LLVM.	2	2		2
6	Тестирование и верификация компиляторного фронтенда.	1	1		1

7	Интерпретация промежуточного представления LLVM.	1	1		1
8	Кодогенерация из промежуточного представления LLVM.	1	1		1
9	Оптимизация промежуточного представления LLVM.	4	4		4
10	Обзор принципов работы бинарной трансляции.	3	3		6
11	Динамическая бинарная трансляция.	2	2		4
12	Статическая бинарная трансляция.	2	2		4
13	Тестирование и верификация компилятора.	4	4		8
14	Обзор принципов работы компиляторного бэкенда.	4	4		8
Итого часов		30	30		45
Подготовка к экзамену		30 час.			
Общая трудоёмкость		135 час., 3 зач.ед.			

4.2. Содержание дисциплины (модуля), структурированное по темам (разделам)

Семестр: 7 (Осенний)

1. Введение в компиляцию. Обзор архитектур существующих компиялторов.

Языковые средства разработки. Типы компиляторов. Синтаксические и динамические трансляторы. Профилирование. Языковые и бинарные трансляторы. Нативная компиляция. Кросс-компиляция. Компиляция в байт-код. Структура языкового компилятора. Трёхстадийный трансялтор.

2. Обзор принципов работы компиляторного фронтенда.

Работа анализатора исходного кода. Лексический анализ. Синтаткский анализ. Семантический анализ. Составление грамматики языка. Методы разбора файла программы. Метод рекурсивного спуска. Таблично-управляемый нисхоящий синтаксический анализ. Восходящий синтаксический анализ.

3. Генераторы синтаксических анализаторов Flex/Bison.

Принципы работы синтаксических анализаторов. Основны работы с Flex. Основы работы с Bison. Правила типов, вычислений и трансялций. Устранения синтаксических ошибок.

4. Обзор принципов работы компиляторного мидленда. Промежуточное представление компилятора. LLVM IR.

Сравнительный анализ промежуточных представлений компиляторов. Анализ зависимости промежуточного представления от целевой архитектуры. Промежуточное представление LLVM. Организация программ, типы, инструкции, переменные промежуточного представления LLVM IR.

5. LLVM API. Програмная генерация пррромежуточного представляения LLVM.

Основные инструменты для работы с промежуточным представлением компиялтора. Принцип менеджера проходов LLVM. Построитель промежуточного представления компиялтора – IR Builder. Понятие Value в промежуточном представлении LLVM. Методы генерации модулей, функций, базовых блоков и инструкций.

6. Тестирование и верификация компиляторного фронтенда.

Определение сценариев для тестирования и верефикации. Внутренние средства LLVM для тестирования и верификации. Генерация тестов для проверки компиляторного фронтенда.

7. Интерпретация промежуточного представления LLVM.

Принцип работы интерпретаторов. Встроенный интерпретатор LLVM. Класс Execution Engine LLVM. Использование Execution Engine для интерпретации промежуточного представления компилятора.

8. Кодогенерация из промежуточного представления LLVM.

Структура генераторов кода. Основные этапы кодогенерации. Выбор инструкций. Планирование инструкций. Машинные инструкции. Распределение регистров. Пролог и эпилог. Реализация собственного прохода генератора кода.

9. Оптимизация промежуточного представления LLVM.

Оптимизации на промежутончном представлении. Зависимости между проходами. Оптимизации времени компиляции и времени компановки. Определение проходов, имеющих значение. Прикладной интерфейс проходов LLVM. Реализация собственного прохода LLVM.

Семестр: 8 (Весенний)

10. Обзор принципов работы бинарной трансляции.

Бинарная трансляция. Описание используемой архитектуры. Принцип работы декодера. Генерация промежуточного представления с помощью декодера.

11. Динамическая бинарная трансляция.

Основы механизма динамической комипляции. Типы динмаической компиляции. Динамическая бинарная трансляция по инструкциям, по базовым блокам и по функциям. Динамическая компиляции промежуточного предсталения, полученного с помощью декодера. Оптимизирующие проходы, применимые при динамической компиляции.

12. Статическая бинарная трансляция.

Механизм статической компиляции. Генерация нативного приложения из промежуточного представления компиялтора. Оптимизирующие проходы, применимые при статической компиляции.

13. Тестирование и верификация компилятора.

Определение сценариев для тестирования и верефикации бинарной трансляции. Использование внутренних средств LLVM для тестирования и верификации бинарной трансляции. Генерация тестов для проверки бинарной трансляции. Использование интерпретации промежуточного представления для верификации компилятора.

14. Обзор принципов работы компиляторного бэкенда.

Инструменты генераторов кода. Язык TableGen. Использование файлов .td с генераторами кода. Инфраструктура описания архитектуры в LLVM. Реализация собственного бэкенда.

5. Описание материально-технической базы, необходимой для осуществления образовательного процесса по дисциплине (модулю)

- необходимое оборудование для лекций : компьютер и проектор
- необходимое программное обеспечение: MS Office Power Point.

6.Перечень рекомендуемой литературы

Основная литература

1. Н. Вирт: Построение компиляторов.
2. Б. К. Лопес, Р. Аулер: LLVM: инфраструктура для разработки компиляторов.
3. А.В.Галатенко, Э.Э.Гасанов, В.Б.Кудрявцев, П.А.Пантелеев: Оптимизирующие компиляторы.

Дополнительная литература

1. А. Ахо, Р. Сети, Д. Ульман, М. Лам: Компиляторы. Принципы, технологии, инструменты.
2. S. Muchnick: Advanced Compiler Design and Implementation.
3. K. D. Cooper, L. Torczon: Engineering a Compiler.
4. С. Свердлов: Конструирование компиляторов.
5. S. Sarda, M. Pandey: LLVM Essentials.

7. Перечень ресурсов информационно-телекоммуникационной сети "Интернет", необходимых для освоения дисциплины (модуля)

1. Примеры разработки компиляторов языков высокого уровня с помощью LLVM: http://llvm.org/docs/tutorial/
2. Создание языка программирования с использованием LLVM: https://habr.com/ru/post/337240/
3. Creating an LLVM Backend for the Cpu0 Architecture:
4. http://jonathan2251.github.io/lbd/

8. Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине (модулю), включая перечень необходимого программного обеспечения и информационных справочных систем (при необходимости)

На лекционных занятиях используются мультимедийные технологии, включая демонстрацию презентаций.

9. Методические указания для обучающихся по освоению дисциплины (модуля)

Посещение семинаров, работа над учебным проектом и самостоятельная работа с литературой. Самостоятельная работа включает в себя: чтение и конспектирование рекомендованной литературы, просмотр интернет-ресурсов по тематике курса, подготовку к ответам на контрольные вопросы.

ПРИЛОЖЕНИЕ

ОЦЕНОЧНЫЕ МАТЕРИАЛЫ ПО ДИСЦИПЛИНЕ (МОДУЛЮ)

по направлению: Информатика и вычислительная техника
профиль подготовки: Компьютерные технологии и вычислительная техника
Физтех-школа Радиотехники и Компьютерных Технологий
кафедра микропроцессорных технологий в интеллектуальных системах управления
курс: 4
квалификация: бакалавр

Семестры, формы промежуточной аттестации:
7 (осенний) - Дифференцированный зачет
8 (весенний) - Экзамен

Разработчик: С.А. Лисицын

1. Компетенции, формируемые в процессе изучения дисциплины

Код и наименование компетенции	Индикаторы достижения компетенции
УК-1 Способен осуществлять поиск, критический анализ и синтез информации, применять системный подход для решения поставленных задач	УК-1.2 Находит, критически анализирует и выбирает информацию, необходимую для решения поставленной задачи
ОПК-2 Способен использовать современные информационные технологии и программные средства при решении задач профессиональной деятельности, соблюдая требования информационной безопасности	ОПК-2.1 Способен применять современные вычислительную технику и сервисы сети Интернет в области (сфере) профессиональной деятельности
ОПК-3 Способен составлять и оформлять научные и (или) технические (технологические, инновационные) отчеты (публикации, проекты)	ОПК-3.3 Владеет методами визуального и графического представления результатов научной (научно-технической, инновационной технологической) деятельности в виде отчетов, научных публикаций
ПК-1 Способен ставить, формализовывать и решать задачи, в том числе разрабатывать и исследовать математические модели изучаемых явлений и процессов, системно анализировать научные проблемы, получать новые научные результаты	ПК-1.1 Способен находить, анализировать и обобщать информацию об актуальных результатах исследований в рамках тематической области своей профессиональной деятельности
	ПК-1.2 Способен выдвигать гипотезы, строить математические модели для описания изучаемых явлений и процессов, оценивать качество разработанной модели
	ПК-1.3 Способен применять теоретические и (или) экспериментальные методы исследований к конкретной научной задаче и интерпретировать полученные результаты
ПК-2 Способен самостоятельно или в качестве члена (руководителя) малого коллектива организовывать и проводить научные исследования и их апробацию	ПК-2.1 Знает принципы построения научной работы, методы сбора и анализа полученного материала, способы аргументации
	ПК-2.2 Способен планировать и проводить научные исследования самостоятельно или в качестве члена (руководителя) малого научного коллектива
	ПК-2.3 Способен проводить апробацию результатов научно-исследовательской работы посредством публикации научных статей и участия в конференциях

2. Показатели оценивания компетенций

В результате изучения дисциплины «Современные методы разработки компиляторов» обучающийся должен:

знать:

- основные методы программирования, применяемые при разработке сложных компиляторных систем;
- методы програмной реализации сложных алгоритмов оптимизаций кода;
- методы тестирования и верификации трансляторов

уметь:

- разрабатывать и реализовывать компилятор, отдельные его части и системы его тестирования и верификации

владеть:

- теоретическими знаниями в области компиляторов, оптимизаций и тестирования трансляторов.

3. Перечень типовых (примерных) вопросов, заданий, тем для подготовки к текущему контролю

С целью контроля освоения обучающимися учебного материала в течение семестра студентам будет предложено написать семестровые проекты с использованием LLVM.

Тема учебного проекта в 7-ом семестре: Реализация языка программирования с использованием LLVM.

Тема учебного проекта в 8-ом семестре: Реализация бинарного транслятора с использованием LLVM.

4. Перечень типовых (примерных) вопросов и тем для проведения промежуточной аттестации обучающихся

Перечень контрольных вопросов для сдачи дифференцированного зачёта:

1. Языковые средства разработки. Типы компиляторов. Синтаксические и динамические трансляторы. Профилирование. Языковые и бинарные трансляторы. Нативная компиляция. Кросс-компиляция. Компиляция в байт-код. Структура языкового компилятора. Трёхстадийный транслятор.

2. Работа анализатора исходного кода. Лексический анализ. Синтаксический анализ. Семантический анализ. Составление грамматики языка. Методы разбора файла программы. Метод рекурсивного спуска. Таблично-управляемый нисходящий синтаксический анализ. Восходящий синтаксический анализ.

3. Принципы работы синтаксических анализаторов. Основы работы с Flex. Основы работы с Bison. Правила типов, вычислений и трансляций. Устранения синтаксических ошибок.

4. Сравнительный анализ промежуточных представлений компиляторов. Анализ зависимости промежуточного представления от целевой архитектуры. Промежуточное представление LLVM. Организация программ, типы, инструкции, переменные промежуточного представления LLVM IR.

5. Основные инструменты для работы с промежуточным представлением компилятора. Принцип менеджера проходов LLVM. Подход строителя промежуточного представления компилятора – IR Builder. Понятие Value в промежуточном представлении LLVM. Методы генерации модулей, функций, базовых блоков и инструкций.

6. Определение сценариев для тестирования и верефикации. Внутренние средства LLVM для тестирования и верификации. Генерация тестов для проверки компиляторного фронтенда.

7. Принцип работы интерпретаторов. Встроенный интерпретатор LLVM. Класс Execution Engine LLVM. Использование Execution Engine для интерпретации промежуточного представления компилятора.

8. Структура генераторов кода. Основные этапы кодогенерации. Выбор инструкций. Планирование инструкций. Машинные инструкции. Распределение регистров. Пролог и эпилог. Реализация собственного прохода генератора кода.

9. Оптимизации на промежуточном представлении. Зависимости между проходами. Оптимизации времени компиляции и времени компоновки. Определение проходов, имеющих значение. Прикладной интерфейс проходов LLVM. Реализация собственного прохода LLVM.

Перечень контрольных вопросов для сдачи экзамена:

1. Языковые средства разработки. Типы компиляторов. Синтаксические и динамические трансляторы. Профилирование. Языковые и бинарные трансляторы. Нативная компиляция. Кросс-компиляция. Компиляция в байт-код. Структура языкового компилятора. Трёхстадийный транслятор.

2. Работа анализатора исходного кода. Лексический анализ. Синтаксический анализ. Семантический анализ. Составление грамматики языка. Методы разбора файла программы. Метод рекурсивного спуска. Таблично-управляемый нисходящий синтаксический анализ. Восходящий синтаксический анализ.

3. Принципы работы синтаксических анализаторов. Основы работы с Flex. Основы работы с Bison. Правила типов, вычислений и трансляций. Устранения синтаксических ошибок.

4. Сравнительный анализ промежуточных представлений компиляторов. Анализ зависимости промежуточного представления от целевой архитектуры. Промежуточное представление LLVM. Организация программ, типы, инструкции, переменные промежуточного представления LLVM IR.

5. Основные инструменты для работы с промежуточным представлением компилятора. Принцип менеджера проходов LLVM. Подход строителя промежуточного представления компилятора – IR Builder. Понятие Value в промежуточном представлении LLVM. Методы генерации модулей, функций, базовых блоков и инструкций.

6. Определение сценариев для тестирования и верефикации. Внутренние средства LLVM для тестирования и верификации. Генерация тестов для проверки компиляторного фронтенда.

7. Принцип работы интерпретаторов. Встроенный интерпретатор LLVM. Класс Execution Engine LLVM. Использование Execution Engine для интерпретации промежуточного представления компилятора.

8. Структура генераторов кода. Основные этапы кодогенерации. Выбор инструкций. Планирование инструкций. Машинные инструкции. Распределение регистров. Пролог и эпилог. Реализация собственного прохода генератора кода.

9. Оптимизации на промежутончном представлении. Зависимости между проходами. Оптимизации времени компиляции и времени компановки. Определение проходов, имеющих значение. Прикладной интерфейс проходов LLVM. Реализация собственного прохода LLVM.

10. Бинарная трансляция. Описание используемой архитектуры. Принцип работы декодера. Генерация промежуточного представления с помощью декодера.

11. Основы механизма динамической компиляции. Типы динамической компиляции. Динамическая бинарная трансляция по инструкциям, по базовым блокам и по функциям. Динамическая компиляции промежуточного представления, полученного с помощью декодера. Оптимизирующие проходы, применимые при динамической компиляции.

12. Механизм статической компиляции. Генерация нативного приложения из промежуточного представления компилятора. Оптимизирующие проходы, применимые при статической компиляции.

13. Определение сценариев для тестирования и верефикации бинарной трансляции. Использование внутренних средств LLVM для тестирования и верификации бинарной трансляции. Генерация тестов для проверки бинарной трансляции.

Структура и инструменты генераторов кода. Основные этапы кодогенерации. Язык TableGen. Использование файлов .td с генераторами кода. Инфраструктура описания архитектуры в LLVM.

Примеры билетов к экзамену

Билет №1

1. Языковые средства разработки. Типы компиляторов. Синтаксические и динамические трансляторы. Профилирование. Языковые и бинарные трансляторы. Нативная компиляция. Кросс-компиляция. Компиляция в байт-код. Структура языкового компилятора. Трёхстадийный транслятор.

2. Структура и инструменты генераторов кода. Основные этапы кодогенерации. Язык TableGen. Использование файлов .td с генераторами кода. Инфраструктура описания архитектуры в LLVM.

Билет №2

1. Бинарная трансляция. Описание используемой архитектуры. Принцип работы декодера. Генерация промежуточного представления с помощью декодера.

2. Принципы работы синтаксических анализаторов. Основы работы с Flex. Основы работы с Bison. Правила типов, вычислений и трансляций. Устранения синтаксических ошибок.

Критерии оценивания

Оценка отлично (10) выставляется студенту, показавшему всесторонние, систематизированные, глубокие знания учебной программы дисциплины, проявляющему интерес к данной предметной области, продемонстрировавшему умение уверенно и творчески применять их на практике при решении конкретных задач, свободное и правильное обоснование принятых решений.

Оценка отлично (9) выставляется студенту, показавшему всесторонние, систематизированные, глубокие знания учебной программы дисциплины и умение уверенно применять их на практике при решении конкретных задач, свободное и правильное обоснование принятых решений.

Оценка отлично (8) выставляется студенту, показавшему систематизированные, глубокие знания учебной программы дисциплины и умение уверенно применять их на практике при решении конкретных задач, правильное обоснование принятых решений, с некоторыми недочетами.

Оценка хорошо (7) выставляется студенту, если он твердо знает материал, грамотно и по существу излагает его, умеет применять полученные знания на практике, но недостаточно грамотно обосновывает полученные результаты.

Оценка хорошо (6) выставляется студенту, если он твердо знает материал, грамотно и по существу излагает его, умеет применять полученные знания на практике, но допускает в ответе или в решении задач некоторые неточности.

Оценка хорошо (5) выставляется студенту, если он в основном знает материал, грамотно и по существу излагает его, умеет применять полученные знания на практике, но допускает в ответе или в решении задач достаточно большое количество неточностей.

Оценка удовлетворительно (4) выставляется студенту, показавшему фрагментарный, разрозненный характер знаний, недостаточно правильные формулировки базовых понятий, нарушения логической последовательности в изложении программного материала, но при этом он освоил основные разделы учебной программы, необходимые для дальнейшего обучения, и может применять полученные знания по образцу в стандартной ситуации.

Оценка удовлетворительно (3) выставляется студенту, показавшему фрагментарный, разрозненный характер знаний, допускающему ошибки в формулировках базовых понятий, нарушения логической последовательности в изложении программного материала, слабо владеет основными разделами учебной программы, необходимыми для дальнейшего обучения и с трудом применяет полученные знания даже в стандартной ситуации.

Оценка неудовлетворительно (2) выставляется студенту, который не знает большей части основного содержания учебной программы дисциплины, допускает грубые ошибки в формулировках основных принципов и не умеет использовать полученные знания при решении типовых задач.

Оценка неудовлетворительно (1) выставляется студенту, который не знает основного содержания учебной программы дисциплины, допускает грубейшие ошибки в формулировках базовых понятий дисциплины и вообще не имеет навыков решения типовых практических задач.

5. Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности

Дифференцированный зачет и экзамен проводятся в устной форме.

При проведении дифференцированного зачета обучающемуся предоставляется 20 минут на подготовку. Опрос обучающегося проводится в течение 30 минут.

При проведении устного экзамена обучающемуся предоставляется 1 час на подготовку. Опрос обучающегося по билету на устном экзамене не должен превышать двух астрономических часов.

Во время проведения дифференцированного зачета и экзамена обучающиеся могут пользоваться программой дисциплины, а также справочной литературой, вычислительной техникой и проч.