

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

УТВЕРЖДЕНО

**Директор физтех-школы
прикладной математики и
информатики**

А.М. Райгородский

	Рабочая программа дисциплины (модуля)
по дисциплине:	Языки программирования и теория компиляции
по направлению:	Информатика и вычислительная техника
профиль подготовки:	Физтех-школа Прикладной Математики и Информатики кафедра алгоритмов и технологий программирования
курс:	3
квалификация:	бакалавр

Семестр, формы промежуточной аттестации: 6 (весенний) - Дифференцированный зачет

Аудиторных часов: 60 всего, в том числе:

лекции: 30 час.

семинары: 30 час.

лабораторные занятия: 0 час.

Самостоятельная работа: 75 час.

Всего часов: 135, всего зач. ед.: 3

Количество контрольных работ, заданий: 1

Программу составил: П.И. Ахтямов, ассистент

Программа обсуждена на заседании кафедры алгоритмов и технологий программирования 04.06.2020

Аннотация

Курс “Языки программирования и теория компиляции” посвящен рассмотрению современных концепции и проектировании и реализации компиляторов. В ходе курса будут рассмотрены три основные стадии компиляции:

- * фронтенд - анализ кода с лексической, синтаксической и семантической точки зрения
- * работа с промежуточным представлением кода - низкоуровневый язык с жестко структурой программных инструкций
- * бэкенд - конвертация промежуточного представления с представлением кода под определенную машинную архитектуру

В лекциях будет сделан акцент на теоретические знания (алгоритмическая база построения стадий компиляции), на семинарах будут разобраны конкретные примеры использования знаний, полученных на лекциях. В качестве семестрового проекта студенты реализуют компилятор для подмножества высокоуровневого языка программирования

1. Цели и задачи

Цель дисциплины

Изучение основных принципов компиляции языков программирования, выявление особенностей компиляции для разных языков программирования.

Задачи дисциплины

- Изучение стадий компиляции программного кода;
- освоение математического аппарата лингвистического анализа программного кода;
- освоение основных алгоритмов предобработки языка программного кода, построения дерева промежуточного представления, и трансляции промежуточного представления в машинный код.

2. Перечень формируемых компетенций

Освоение дисциплины направлено на формирование следующих компетенций:

Код и наименование компетенции	Индикаторы достижения компетенции
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-2.2 Знает и умеет применять численные математические методы и прикладное программное обеспечение для решения научных задач в профессиональной области
	ОПК-2.3 Знает основные требования информационной безопасности

3. Перечень планируемых результатов обучения по дисциплине (модулю)

В результате освоения дисциплины обучающиеся должны

знать:

- Постановку задачи построения компиляторов;
- основные стадии компиляции, основные подстадии компиляции.

уметь:

- Формулировать задачи компиляции программного кода;
- реализовывать подходящий алгоритм выделения синтаксических, семантических сущностей программного кода;
- решать задачи оптимизации деревьев выполнения программ, составления оптимального машинного кода.

владеть:

- Основными программными системами для выделения грамматических сущностей программного кода.

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 теорию компиляции	2	2		5
2	Методы выделения слов программного кода	2	2		5
3	Выделение синтаксической конструкции программного кода. Методы нисходящего анализа текста	2	2		5
4	Выделение синтаксической конструкции программного кода. Методы восходящего анализа текста. Обработка ошибок компиляции	2	2		5
5	Построение промежуточного представления. Системы типов. Семантическая проверка программного кода	2	2		5
6	Семантический анализ дерева разбора	2	2		5
7	Основы механизмов вызова процедур	2	2		5
8	Построение промежуточного представления по синтаксическому дереву разбора	2	2		5
9	Оптимизация промежуточного представления. Понятие о каноническом дереве промежуточного представления	2	2		5
10	Преобразование промежуточного представления в низкоуровневый язык. Механизм выбора инструкций	2	2		5
11	Методы анализа графа исполнения машинного кода	2	2		5
12	Механизм распределения регистров	2	2		5
13	Механизмы сборки мусора	2	2		5
14	Особенности реализации компиляторов ООП-языков	2	2		5
15	Особенности реализации компиляторов для языков, основанных на функциональной парадигме	2	2		5
Итого часов		30	30		75
Подготовка к экзамену		0 час.			
Общая трудоёмкость		135 час., 3 зач.ед.			

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

1. Введение в теорию компиляции

Устройство компиляторов, история компиляции. Отличие компиляции от интерпретации. Основные составляющие компилятора: frontend, IR, backend. Компилятор как транслятор.

2. Методы выделения слов программного кода

Основная задача лексического анализа. Принципы построения лексических анализаторов. Интерфейс задания шаблонов при построении анализаторов - регулярные выражения. Одноразовый проход анализатора - построение НКА по регулярному выражению, ДКА, минимизация ДКА. Построение анализаторов на основе ДКА. Типы сканеров - табличные, генеративные и эвристические.

3. Выделение синтаксической конструкции программного кода. Методы нисходящего анализа текста

Основная задача синтаксического анализа кода. Понятие контекстно-свободной грамматики. Однозначные контекстно-свободные грамматики. Предиктивный (нисходящий) алгоритм LL(1) разбора грамматик: определение множеств FIRST, FOLLOW, избавление от левосторонней рекурсии в правилах грамматики. Недостатки нисходящего разбора. Обработка ошибок компиляции при нисходящем анализе.

4. Выделение синтаксической конструкции программного кода. Методы восходящего анализа текста. Обработка ошибок компиляции

Алгоритмы восходящего анализа: LR(0), SLR, LR(1). Понятие LR-таблицы. Обработка левоассоциативных правил. Оптимизация LR-таблицы. Обработка ошибок компиляции при восходящем анализе.

5. Построение промежуточного представления. Системы типов. Семантическая проверка программного кода

Понятие синтаксического дерева разбора. Сохранение позиций кодовых элементов при построении синтаксического дерева разбора. Обход синтаксического дерева разбора: паттерн проектирования Посетитель.

6. Семантический анализ дерева разбора

Понятие таблицы символов. Структуры данных для хранения таблицы символов: хэш-таблицы, деревья поиска. Понятие персистентных структур данных, персистентное дерево поиска. Системы типов, обработка ошибок, связанных с неправильной типизацией переменных.

7. Основы механизмов вызова процедур

Понятие стека вызова функции, указателя на стек. Механизм передачи параметров в функцию. Механизм возвращения результатов вычисления функции. Хранение локальных переменных, статических переменных.

8. Построение промежуточного представления по синтаксическому дереву разбора

Деревья промежуточного представления кода. Преобразование примитивных типов, списочных типов, строковых типов. Преобразование арифметических выражений, условных выражений, циклов, вызовов функций. Преобразование объявления переменных, конструкторов классов.

9. Оптимизация промежуточного представления. Понятие о каноническом дереве промежуточного представления

Понятие канонического дерева промежуточного представления. Правила преобразований дерева промежуточного представления в каноническое дерево: поднятие операций вызова функций, линеаризация, обработка условных операций.

10. Преобразование промежуточного представления в низкоуровневый язык. Механизм выбора инструкций

Паттерны преобразования деревьев в набор машинных инструкций. Понятие виртуального регистра. Алгоритмы выбора инструкций: максимальное совпадение, динамическое программирование, быстрое нахождение паросочетаний. Особенности выбора инструкции для CISC архитектур.

11. Методы анализа графа исполнения машинного кода

Понятие графа исполнения кода. Понятие живой переменной. Математическая модель анализа потока данных выполнения программ. Решение уравнений анализа потока данных для нахождения оптимального способа исполнения кода.

12. Механизм распределения регистров

Задача преобразования виртуальных регистров в физические регистры. Сложность решения задачи преобразования регистров. Эвристические алгоритмы на графах для распределения регистров.

13. Механизмы сборки мусора

Определение задачи сборки мусора. Подсчет количества ссылок на объект. Алгоритм пометок сборки мусора. Поколенческие сборщики мусора. Инкрементальные сборщики мусора.

14. Особенности реализации компиляторов ООП-языков

Реализация полиморфизма, реализация областей видимости методов и полей при наследовании. Реализация множественного наследования, порядок разрешения методов.

15. Особенности реализации компиляторов для языков, основанных на функциональной парадигме

Реализация замыканий (лямбда-функций), неизменяемых переменных. Развертывание inline-инструкций. Обработка хвостовой рекурсии, механизмы ленивого вычисления функций.

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

Учебная аудитория, оснащенная мультимедийным оборудованием (проектор или плазменная панель), доской.

6. Перечень рекомендуемой литературы

Основная литература

Не предусмотрено

Дополнительная литература

Не предусмотрено

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

Не используются

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)

Дополнительная литература:

1. Andrew W. A., Jens P. Modern compiler implementation in Java //ISBN 0–521–58388–8. – Cambridge University Press, 2002.
2. Cooper K., Torczon L. Engineering a compiler. – Elsevier, 2011.
3. Pandey M., Sarda S. LLVM cookbook. – Packt Publishing Ltd, 2015.

ОЦЕНОЧНЫЕ МАТЕРИАЛЫ ПО ДИСЦИПЛИНЕ (МОДУЛЮ)

по направлению: Информатика и вычислительная техника

профиль подготовки: Физтех-школа Прикладной Математики и Информатики
кафедра алгоритмов и технологий программирования

курс: 3

квалификация: бакалавр

Семестр, формы промежуточной аттестации: 6 (весенний) - Дифференцированный зачет

Разработчик: П.И. Ахтямов, ассистент

1. Компетенции, формируемые в процессе изучения дисциплины

Код и наименование компетенции	Индикаторы достижения компетенции
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-2.2 Знает и умеет применять численные математические методы и прикладное программное обеспечение для решения научных задач в профессиональной области
	ОПК-2.3 Знает основные требования информационной безопасности

2. Показатели оценивания компетенций

В результате изучения дисциплины «Языки программирования и теория компиляции» обучающийся должен:

знать:

- Постановку задачи построения компиляторов;
- основные стадии компиляции, основные подстадии компиляции.

уметь:

- Формулировать задачи компиляции программного кода;
- реализовывать подходящий алгоритм выделения синтаксических, семантических сущностей программного кода;
- решать задачи оптимизации деревьев выполнения программ, составления оптимального машинного кода.

владеть:

- Основными программными системами для выделения грамматических сущностей программного кода.

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

Устройство компиляторов, история компиляции.
Основная задача лексического анализа.
Понятие контекстно-свободной грамматики.
Понятие LR-таблицы.

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

1. Постройте лексический анализатор для команд движения робота по клетчатой доске: задайте регулярным выражением, постройте минимальный детерминированный конечный автомат для обработки команд
2. Постройте восходящий и нисходящий парсер для синтаксического разбора арифметических выражений.
3. По участку кода определите, на каком уровне вложенности находятся переменные. Покажите ход изменения персистентного дерева поиска для таблицы символов.
4. Покажите участки кода промежуточного представления IR/LLVM, где происходит передача параметров функции, возвращение результатов исполнения.
5. Перечислите основные требования, предъявляемые к языку промежуточного представления.
6. Предъявите пример кода на языке C++, промежуточное представление которого отличается в зависимости от уровня оптимизации. Объясните основные различия.
7. Постройте каноническое IR-дерево по имеющемуся IR-дереву.

8. По имеющемуся каноническому IR-дереву постройте набор инструкций для низкоуровневого кода.
9. Постройте граф исполнения кода по представленному примеру IR-дерева.
10. Продемонстрируйте ход действия алгоритма распределения физических регистров, написанному на уровне виртуальных регистров.
11. Опишите основные различия в реализации алгоритмов для хранения таблицы символов в языках без использования высокоуровневых функций и с использованием языка высокоуровневых функций.

Пример экзаменационного билета:

1. Постройте граф исполнения кода по представленному примеру IR-дерева.
2. По участку кода определите, на каком уровне вложенности находятся переменные. Покажите ход изменения персистентного дерева поиска для таблицы символов.

Критерии оценивания

оценка «отлично (10)» 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 (9)» выставляется студенту, показавшему всесторонние, систематизированные, глубокие знания учебной программы дисциплины и умение применять их на практике при решении конкретных задач, свободное и правильное обоснование принятых решений;

- оценка «отлично (8)» выставляется студенту, показавшему всесторонние систематизированные, глубокие знания учебной программы дисциплины и умение применять их на практике при решении конкретных задач, и правильное обоснование принятых решений;

- оценка «хорошо (7)» 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 (6)» выставляется студенту, если он 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 знает материал, и по существу излагает его, умеет применять полученные знания на практике, но допускает в ответе или в решении задач некоторые неточности;

- 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 владеет основными разделами учебной программы, необходимыми для дальнейшего обучения и может применять полученные знания по образцу в стандартной ситуации;

- оценка «удовлетворительно (3)» выставляется студенту, показавшему фрагментарный, разрозненный характер знаний, недостаточно правильные формулировки базовых понятий, нарушения логической последовательности в изложении программного материала, но при этом он владеет фрагментарно основными разделами учебной программы, необходимыми для дальнейшего обучения и может применять полученные знания по образцу в стандартной ситуации;

- оценка «неудовлетворительно (2)» выставляется студенту, который не знает большей части основного содержания учебной программы дисциплины, допускает грубые ошибки в формулировках основных понятий дисциплины и не умеет использовать полученные знания при решении типовых практических задач;

- оценка «неудовлетворительно (1)» выставляется студенту, который не знает формулировок основных понятий дисциплины.

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

Дифференцированный зачёт ставится по сумме баллов, полученных за семестр

При проведении устного дифференцированного зачёта обучающемуся предоставляется 30 минут на подготовку.

Результирующая оценка рассчитывается по формуле:

$$O_{\text{итоговая}} = O_{\text{накопл}}$$

Накопленная оценка рассчитывается как среднее значение оценок за все выданные домашние задания и самостоятельные работы. В конце семестра разрешается переписать все самостоятельные работы, пропущенные по уважительной причине.

Накопленная и итоговая оценки округляются арифметически.