Использование Рефала в учебном процессе

Александр Коновалов, МГТУ имени Н.Э. Баумана, Москва

VI совместное рабочее совещание ИПС имени А.К. Айламазяна РАН и МГТУ имени Н.Э. Баумана
по функциональному языку программирования Рефал,
17 июня 2023

Потребность в новом курсе

Направление подготовки в бакалавриате (1)

Направление подготовки в бакалавривате — «Анализ, порождение и преобразование программного кода»:

Факультет — Информатика и системы управления
Кафедра — Теоретическая информатика и компьютерные технологии, ИУ9

УЧЕБНЫЙ ПЛАН

Направление подготовки — 01.03.02, Прикладная математика и информатика
Направленность — Анализ, порождение и преобразование программного кода
Квалификация — Бакалавр
Срок обучения — 4 года. Год начала обучения — 2022
Форма обучения — очная

Figure 1: Скриншот из учебного плана

Направление подготовки в бакалавриате (2)

Какие курсы студенты слушают в рамках данного направления подготовки:

- Низкоуровневое программирование (3 семестр) — Царёв А.С.
- Теория формальных языков (5 семестр) — Непейвода А.Н.
- Конструирование компиляторов (6 семестр) — Коновалов А.В.
- Генерация оптимального кода (6 семестр) — Синявин А.В.
- Курсовая работа по конструированию компиляторов/численным методам (7 семестр)

Компиляторные курсы на кафедре (1)

Курс «Конструирование компиляторов» (КК) создан Скоробогатовым С.Ю. в 2009 году, с 2014 года докладчик ведёт лабораторные работы, с 2017 года — курс целиком (лекции и лабы). В курсе изучается стадия анализа компилятора: лексический, синтаксический и семантический анализ.

Курс «Генерация оптимального кода» (ГОК) ведёт Синявин А.В. В курсе изучаются аспекты оптимизации на стадии синтеза: граф потока управления, анализ потока управления, анализ потока данных, SSA-форма, распределение регистров. Компиляция рассматривается не в машинный код, а в LLVM.

Компиляторные курсы на кафедре (2)

Вне курсов КК и ГОК оказывается несколько интересных тем:

- Компиляция типичных императивных конструкций в машинный код: фреймы стека и локальные переменные в них, компиляция условных выражений AND и OR.
- Компиляция высокоуровневых абстракций: классы, обработка исключений.
- Управление памятью и сборка мусора.
- Понятие библиотеки поддержки времени выполнения («рантайма») как таковой — зачем она нужна и что в ней есть.

Нужен новый курс (1)

По мнению автора слайдов, студенты направления подготовки «Анализ, порождение и преобразование программного кода» должны рассмотренные ранее темы знать.

Но куда их включить? Если в курс КК, то из него придётся выкидывать что-то в нём имеющееся, а жалко. А курс ГОК читает не докладчик, поэтому вообще не в праве что-либо менять, и, по мнению докладчика, ГОК хорош сам по себе и трогать его не надо.

Нужен новый курс (2)

Вставить новый курс в учебный план можно только переписав весь учебный план, что бюрократически слишком масштабно. Технически более реально переименовать один из имеющихся курсов и утвердить у него новую программу.

Например, у нас на 7 семестре был курс физики, вёл его Каганов Ю.Т. В прошлом году появилось требование читать на всех кафедрах факультета ИУ нейросети. Курс физики превратили в курс «Теория искусственных нейронных сетей», его тоже ведёт Каганов Ю.Т.

Нужен новый курс (3)

Значит, нужно какой-то другой курс заменить на новый.

Скоробогатов С.Ю. вёл курс «Объектно-функциональное программирование» (6 семестр), на котором студенты изучали язык Scala. Курс маленький: одна пара в неделю, семинары чередовались с лабораторными, семинары использовались как лекции. За курс ставится зачёт.

Сейчас этот курс ведёт докладчик.

По мнению докладчика, для студентов нашего направления подготовки полезнее изучить темы, не попавшие в курсы КК и ГОК, чем осваивать ещё один язык программирования (в дополнение к C, Scheme, Go, Java, C++, Kotlin).

Так что курсом ОФП надо пожертвовать.

Курс «Реализация абстракций в языках программирования»

Факультатив — альфа-версия курса (1)

Изначально планировалась бета-версия, но получилась альфа ☺

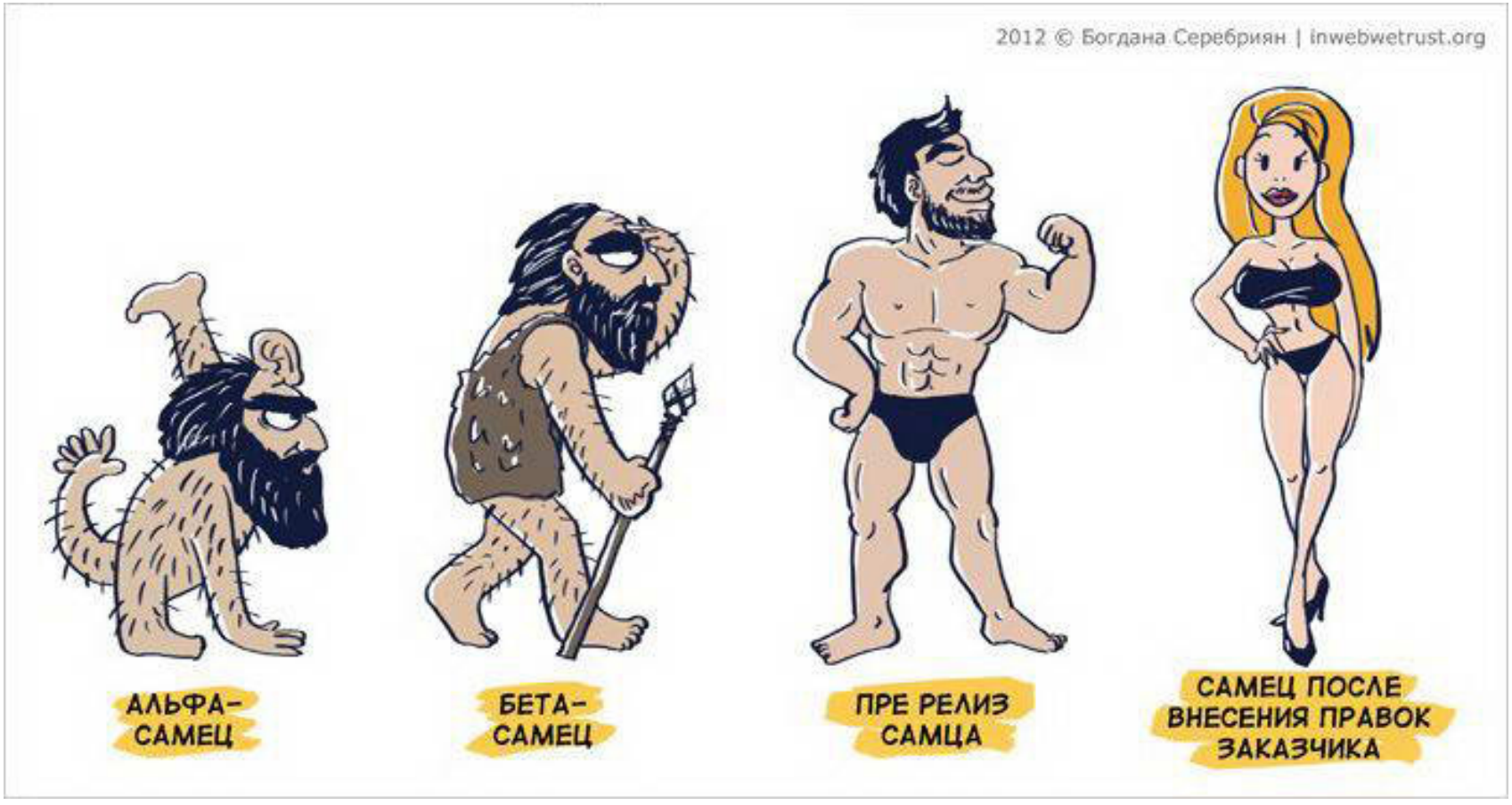

Figure 2: Стадии выпуска ПО

Факультатив — альфа-версия курса (2)

Курс читался по субботам. Как и ОФП, на него отводилась одна пара в неделю.

Чтобы дополнительно мотивировать студентов поучаствоввать в бетатесте курса, им был предложен зачёт по курсу ОФП + 30 баллов автоматом на экзамене КК.

Преподаватель не всегда успевал готовиться к лекциям и иногда писал конспект лекции одновременно с самой лекцией, поэтому получилась альфа-версия ☺

По факту получилось так: одна или несколько пар — лекция, рассказ нового материала, потом лабораторная работа, на которой выдаётся новое задание и принимается предыдущее.

Содержание курса (1)

В курсе нужно рассаматривать компиляцию конструкций высокоуровнеых императивных языков в язык низкого уровня — ассемблер или машинный код. А это означает, что требуется выбрать три языка:

- Входной язык, который будем компилировать.
- Целевой язык, куда будем компилировать.
- Язык реализации, на котором будем писать компилятор.

Входной и целевой языки должны быть простыми. Во-первых, их рассмотрение не должно требовать много времени, т.к. курс маленький. Во-вторых, детали не должны отвлекать от сути дела.

Язык реализации должен быть удобен для выражения рассматриваемых в курсе алготимов.

Языки курса (1) — целевой язык (1)

В качестве входного языка был выбран ассемблер стековой виртуальной машины. Ассемблер и машина были вдохновлены ими же из книги С.З. Свердлова «Языки программирования и методы трансляции».[1]

В виртуальной машине память представлена массивом целочисленных слов, по младшим адресам располагается программа (стартует со слова 0), в старших адресах расположен стек, который растёт вниз. Пространство между программой и стеком можно использовать как кучу.

---

[1] Свердлов С.З. Языки программирования и методы трансляции: Учебное пособие — СПб.: Питер, 2007. — 638 с.: ил.

Языки курса (1) — целевой язык (2)

Три регистра: `IP`, `SP`, `BP`. Ввод-вывод: инструкция `IN` читает символ в юникоде из `stdin`, инструкция `OUT` — пишет на `stdout`.

Каждая инструкция занимает одно слово. Инструкции безадресные. Положительные числа соответствуют инструкциям помещения константы на стек, отрицательные — прочие инструкции. (Чтобы поместить на стек отрицательное число, нужно поместить положительное и выполнить инструкцию `NEG` с кодом -6.)

Языки курса (1) — целевой язык (3)

В ассемблере можно определять метки как `:‹имя метки›`, константы как целые числа со знаком, мнемоники инструкций и имена меток. Переводы строк считаются пробельными символами, поэтому можно записывать несколько инструкций в одной строке.

```
:factorial      ; n ret
SWAP            ; ret n
DUP factorial_non_zero JNE
DROP 1 SWAP     ; 1 ret
GOTO            ; 1

:factorial_non_zero
DUP 1 SUB       ; ret n (n-1)
factorial CALL ; ret n (n-1)!
MUL SWAP       ; n*(n-1)! ret
GOTO            ; n*(n-1)!
```

Языки курса (1) — целевой язык (4)

Виртуальная машина спроектирована таким образом, чтобы на ней можно было удобно выражать рассматриваемые в курсе понятия: компиляция императивных конструкций (циклы, ветвления), фреймов стека, выражений и т.д.

Особенности реальных машин вроде вещественных чисел, чисел разного размера, байтовая адресация и машинное слово из нескольких байт для курса являются несущественными и в машине отражения не нашли.

Синтаксис ассемблера спроектирован так, чтобы было проще писать интерпретатор, и для удобства небрежной кодогенерации из Рефала-5:

```
<Prout ':_' s.FuncName>
<Prout '_' s.FuncName CALL DROP>
```

Языки курса (2) — входной язык и язык реализации (1)

Поиск синтаксических и семантических ошибок — тема курса КК. В курсе РАЯП можно считать, что на входе мы имеем всегда корректную программу.

Синтаксический разбор — это тоже тема курса КК, поэтому будет удобнее, если входным языком сразу будет абстрактное синтаксическое дерево. Допустимо считать, что в курсе РАЯП мы рассматриваем только стадию синтеза компилятора.

Входное синтаксическое дерево должно быть удобно для загрузки из программы, а также чтения и восприятия.

Языки курса (2) — входной язык и язык реализации (2)

Язык реализации должен быть удобен для выражения рассматриваемых алгоритмов.

Обычно, алгоритмы описываются как правила переписывания синтаксических деревьев в целевой код — узел дерева с неизвестными поддеревьями переписывается по некоторому шаблону, куда также вставляются результаты трансляции поддеревьев.

Поэтому язык реализации должен иметь средства для сопоставления с образцом и лаконичное выражение конкатенации.

Языки курса (2) — входной язык и язык реализации (3)

Для языка реализации был выбран Рефал-5. Его преимущества:

- Простота. Почти весь диалект можно рассказать быстро, за 2 лекции (докладчик рассказал за 3, т.к. не готовил слайды).
- Выразительность — мощное сопоставление с образцом, конкатенация выражается «пробелом».
- Синтаксическое дерево входного языка можно описывать как объектные выражения Рефала — читаемость дерева на уровне Лиспа, при этом можно использовать инфиксную запись (`((L x) "+" (L y))` вместо `("+" (L x) (L y))`).

Входной язык описывается данными Рефала.

Языки курса (2) — входной язык и язык реализации (4)

Входной язык должен быть низкоуровневым на уровне Си и даже ниже — меньше расстояние между ним и целевым языком. Язык не предназначен для написания прикладных программ, поэтому удобство программирования на нём не играет роли. (На нём предполагается писать алгоритм сборки мусора, но можно потерпеть.)

Язык так и называется: НизкоУровневый Императивный Язык Программирования — НУИЯП.

Языки курса (2) — входной язык и язык реализации (5)

Особенности НУИЯПа:

- Нетипизированный. Единственный тип данных — целочисленное слово, массив слов. В частности, значения имён переменных и функций — их адреса, т.е. числа.
- Поддерживаются структуры. Имя структуры — целочисленная константа, её размер. Имена полей — тоже целочисленные константы в глобальной области видимости, их смещения.
- Для переменной и поля структуры просто указывается, сколько слов они занимают.
- Можно определять свои целочисленные константы.

Языки курса (2) — входной язык и язык реализации (6)

Особенности НУИЯПа (продолжение):

- Параметр функции — переменная размером 1.
  Возвращаемое значение — слово.
- Операторы: присваивание, вызов функции, `if`, `while`, `return`.
- Арифметические выражения: имя, число, разыменование, арифметические операции, вызов функции.
- Логические выражения: отношение, `and`, `or`, `not`, `TRUE`, `FALSE`.

Языки курса (2) — входной язык и язык реализации (7)

(Тут докладчик переключается на слайды лекций и иллюстрирует.)

Содержание курса (2)

Получился вот такой курс:

1. **Виртуальная машина и ассемблер** (1-2 лекции). Лабораторная работа — написание интерпретатора виртуальной машины.
2. **Рефал-5** (2 лекции). Лабораторная работа: несколько простых функций на Рефале (например, символьное дифференцирование).
3. **НУИЯП** (2 лекции). Лабораторная работа: написание компилятора базового НУИЯПа в ассемблер + добавление расширения на выбор (например, `for` как в Паскале).
4. **ООП** (2 лекции). Лабораторная работа: расширение базового НУИЯПа объектно-ориентированными средствами на выбор.
5. **Управление памятью и сборка мусора** (2 лекции). Лабораторная работа: расширение НУИЯПа синтаксисом для объектов, рапределяемых в куче + сборка мусора (несколько вариантов на выбор).

Содержание курса (2)

Да, исключения в курс не влезли ☹

Получается, что в среднем (2 лекции + 1 лаба) × 5 тем = 15 занятий. А в семестре 17 занятий, так что всё нормально.

В этом году участникам бетатеста лабораторную по сборке мусора можно было не делать, т.к. времени на неё не осталось.

Отзывы студентов

Список участников бета-теста

- Дельман Александр Дмитриевич (ИУ9-62Б)
- Котов Андрей Андеевич (ИУ9-61Б)
- Петряев Юрий Вадимович (ИУ9-61Б)
- Степанов Борис Валерьевич (ИУ9-62Б)
- Терюха Михаил Романович (ИУ9-62Б)

БОЛЬШОЕ ВАМ СПАСИБО!

Отчёт бетатестера (1)

Классический бетатест подразумевает написание бетатестером обстоятельного об использованной программе. Был отчёт и в этом курсе. На следующем слайде он будет показан как скришот и мелко, но это не важно.

Далее я просто выберу наиболее интересные цитаты.

Цитаты упорядочены по фамилиям авторов.

Отчёт бетатестера (2)

Отчёт бетатестера курса

Цели курса:

- Ликвидировать разрыв между курсами «Конструирование компиляторов» и «Генерация оптимального кода». В курсе КК рассматривается только стадия анализа, в курсе ГОК — некоторые аспекты стадии синтеза (в частности, генерация кода). В курсе РАЯП предполагается рассмотреть стадию синтеза «с птичьего полёта».
- Познакомиться с реализацией высокоуровневых языковых абстракций. В этом семестре удалось рассмотреть локальные переменные и фреймы стека, ООП и автоматическое управление памятью.
- Познакомиться с понятием библиотеки поддержки времени выполнения (жарг. «рантайм») — набора средств (как правило, подпрограмм), которые используются в скомпилированном коде.

Вопросы к участникам:

1. На сколько, на ваш взгляд, удалось добиться поставленных целей:
   - более разносторонне рассмотреть стадию синтеза,
   - узнать о реализации высокоуровневых абстракций,
   - понять, что такое «рантайм» и зачем он нужен.
2. На сколько материал был новым:
   - какие темы из рассказанных вам уже были хорошо известны,
   - какие темы вы знали поверхностно,
   - какой материал был новым?
3. РАЯП тематически близок к курсу ГОК, как связаны эти курсы:
   - было ли перекрытие по материалу, изучаемому в обоих курсах,
   - были ли темы, которые рассказывались в обоих курсах, но по-разному,
   - было ли расхождение в используемой терминологии,
   - не обнаружились ли новые зазоры — материал, который предполагался известным при изложении обоих курсов, но фактически незнакомый?
4. РАЯП тематически близок к курсу «Низкоуровневое программирование», вопросы те же:
   - было ли перекрытие по материалу, изучаемому в обоих курсах,
   - были ли темы, которые рассказывались в обоих курсах, но по-разному,
   - было ли расхождение в используемой терминологии,
   - не обнаружились ли новые зазоры — материал, который предполагался известным при изложении обоих курсов, но фактически незнакомый (например, в РАЯП между делом упомянулась защита страниц и страничное прерывание, материал, который студентами забылся)?
5. В курсе рассматривалась компиляция в модельный ассемблер виртуальной стековой машины. Как, на ваш взгляд, такое решение:
   - ☐ на рассмотренной модели удобно показывать реализацию конструкций императивных языков программирования,
   - ☐ модельный ассемблер не совсем удачный — его надо переделать в следующих аспектах *(перечислите)*,
   - ☐ лучше использовать ассемблер реальной машины — укажите какой.
6. В курсе рассматривалась компиляция из абстрактного синтаксического дерева, описанного как структура данных Рефала-5. По мнению автора курса, Рефал-5 позволяет описывать синтаксические деревья в наглядной форме, почти приближенной к исходному тексту программы (например, `(x "=" ((L x) "+" 1))`), а Рефал позволяет лаконично описывать алгоритмы, рассмотренные в курсе. На сколько выбор Рефала оказался оправдан:
   - ☐ Рефал-5 хорошо подходит для описания рассмотренных алгоритмов,
   - ☐ больше подошёл бы другой диалект Рефала *(укажите и обоснуйте)*,
   - ☐ больше подошёл бы другой язык программирования *(укажите и обоснуйте)*.
7. Другие дополнения, мнения, впечатления...

На вопросы 5 и 6 тоже нужно дать развёрнутые ответы.

Цитаты (1)

На сколько выбор Рефала оказался оправдан:

Александр Дельман:

> *Признаться честно, выбор этого языка для курса сильно повлиял на мое желание поучаствовать в его тесте, так как было интересно познакомиться с Рефалом. И полученный опыт я могу считать интересным. Понимаю, что в курсе рассмотрены лишь основы, но и их достаточно для решения лабораторных работ, к тому же, изучить их не так сложно. Формат описания синтаксиса НИУЯПА грамматикой типов на языке Рефал-5 понятен и удобен. Поэтому считаю выбор оправданным.*

Цитаты (2)

РАЯП тематически близок к курсу ГОК, как связаны эти курсы:

Андрей Котов:

> *Перекрытия по материалу с ГОК не было. Стадия синтеза в РАЯП была рассмотрена на более низком уровне, чем в курсе генерации оптимального кода: прямой работы с памятью в ГОК не было. Расхождения в терминологии с ГОК также не было. И в РАЯП, и в ГОК я не встречал ситуации, когда какой-то материал предполагался известным, но я его не изучал.*

Цитаты (3)

На сколько выбор Рефала оказался оправдан:

Андрей Котов:

> *Рефал-5 хорошо подошел для этого курса из-за своей системы сопоставления с образцом, конечно, можно было бы реализовывать все на другом языке с ООП, но пришлось бы писать больше кода и уделять больше времени реализации, но я не вижу смысла без надобности усложнять задачи.*

Цитаты (4)

РАЯП тематически близок к курсу ГОК, как связаны эти курсы:

Юрий Петряев:

> *Перекрытия по материалу не было, в ГОК рассматривалась только оптимизация кода с построением CFG и SSA и написание бэкенда с помощью gcc и llvm, которые абстрагируют от прямого синтеза в машинный код. Также довольно поверхностно было рассказано про давление на регистры (возможно про это не успели рассказать). Расхождения в терминологии и зазоров не было.*

Цитаты (5)

На сколько выбор Рефала оказался оправдан:

Юрий Петряев:

> *Рефал-5 очень хорошо подходит для описания рассмотренных алгоритмов. Механизм сопоставления с образцом сэкономил много времени для более детального изучения реализации высокоуровневых абстракций.*

Цитаты (6)

Другие дополнения, мнения, впечатления…

Юрий Петряев:

Курс вышел довольно обширным и очень интересным. Также круто, что все лабораторные работы взаимосвязаны и в итоге позволяют написать синтез ЯП с ООП, аллокаторами и сборкой мусора в ассемблер, который исполняется в реализованной виртуальной машине. Очень обрадовало большое количество разнообразных заданий, включающие в себя и задания повышенной сложности (к сожалению, к ним не нашлось времени приступить).

Цитаты (7)

На сколько выбор Рефала оказался оправдан:

Борис Степанов:

> *Рефал-5 подходит для этой задачи, но большинству студентов его придется учить с нуля. При этом по нему не так много документации, как по более распространенным языкам (C++, Python). Возможно стоит оставить возможность самому выбирать язык, на котором будешь писать компилятор, и привести пару примеров построения case-классов для них.*

Цитаты (8)

Другие дополнения, мнения, впечатления…

Борис Степанов:

> *Можно добавить в case-классы ассемблерные вставки (как в Си). При написании функций для работы с динамической памятью иногда чувствовалось, что процесс можно было бы упростить вставкой кода на ассемблере. Если ещё и добавить регистры в стековую машину, то это будет очень мощным инструментом. Или можно сделать это вариантом ЛР.*

Цитаты (9)

На сколько выбор Рефала оказался оправдан:

Михаил Терюха:

> *Я считаю что Рефал-5 лучше всего подходит для этого курса, более того для части курса по ТФЯ тоже лучше использовать Рефал, поэтому возможно часть ребят уже будет кое-как уметь писать на этом языке. <…> На рефале наглядно и просто использовать сопоставление с образцом, что является с плюсом, да и на самом деле значительная часть курса отведена под изучение Рефала.*

Выводы

Выводы

- Получился черновик курса с важным для направления подготовки материалом.
- Найдена сфера применения, для которой Рефал очень хорошо подходит.
- Студентам понравилось.