|  |
| --- |
| Министерство науки и высшего образования Российской Федерации |
| федеральное государственное автономное образовательное учреждение  высшего образования |
| **«Национальный исследовательский ядерный университет «МИФИ»** |
| ИНСТИТУТ ИНТЕЛЛЕКТУАЛЬНЫХ КИБЕРНЕТИЧЕСКИХ СИСТЕМ |
| КАФЕДРА КИБЕРНЕТИКИ (№ 22) |

«УтверждЕН

на заседании кафедры

«\_\_\_»\_\_\_\_\_\_\_\_\_\_20\_\_ г.,

протокол №\_\_\_\_\_\_\_\_\_

зав.каф.22

\_\_\_\_\_\_\_\_\_\_ / А.М.Загребаев /

**ФОНД ОЦЕНОЧНЫХ СРЕДСТВ**

по дисциплине

**«Функциональное программирование»**

|  |  |
| --- | --- |
| Направление подготовки (специальность) | 09.03.04 Программная инженерия |
|  |  |
| Профиль подготовки (при его наличии) |  |
|  |  |
| Наименование образовательной программы (специализация) | Математическое и программное обеспечение вычислительных машин и компьютерных сетей |
|  |  |
| Квалификация (степень) выпускника | бакалавр |
|  |  |
| Форма обучения | очная |

г. Москва, 2019 г.

**ПАСПОРТ**

**фонда оценочных средств по дисциплине**

**«Функциональное программирование»**

(наименование дисциплины)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Семестр** | **Интерактив** | **Трудоемкость, кред.** | **Общий объем курса, час.** | **Лекции, час.** | **Практич. занятия, час.** | **Лаборат. работы, час.** | **СРС, час.** | **КСР, час.** | **Форма(ы) контроля, экз./зач./КР/КП** |
| 8 |  | 2 | 72 | 10 | 20 | 0 | 42 | 0 | З |
| ИТОГО | 0 | 2 | 72 | 10 | 20 | 0 | 42 | 0 |  |

Группа: Б19-504, Б19-514

**Модели контролируемых компетенций**

В результате освоения дисциплины у выпускника формируются следующие компетенции:

ОПК-1 – Способен применять естественнонаучные и общеинженерные знания, методы математического анализа и моделирования, теоретического и экспериментального исследования в профессиональной деятельности

ОПК-2 – Способен использовать современные информационные технологии и программные средства, в том числе отечественного производства, при решении задач профессиональной деятельности

ОПК-3 – Способен решать стандартные задачи профессиональной деятельности на основе информационной и библиографической культуры с применением информационно-коммуникационных технологий и с учетом основных требований информационной безопасности

ПК-3 – владением навыками использования различных технологий разработки программного обеспечения

**Программа оценивания контролируемых компетенций**

Формирование у студентов компетенций контролируется в течение всего времени освоения дисциплины в рамках:

* текущего контроля на каждом занятии;
* рубежного контроля;
* промежуточного контроля.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№ п.п.** | **Наименование раздела учебного курса** | **Не- дели** | **Лекции час.** | **Практ. зан./ семи-нары, час.** | **Лаб. рабо-ты, час.** | **Обязат. текущий контроль (форма\*, неделя)** | **Аттеста-ция раздела (форма\*, неделя)** | **Макси-мальный балл за раздел \*\*** | **Компетенции по разделам, проверяемые при текущем и рубежном контроле** | **Компетенции, проверяемые на зач. /экз.** |
| 8 семестр | | | | | | | | | | |
| 1 | Раздел 1 | 1-5 | 5 | 10 |  |  | КИ, 5 | 25 | ОПК-1  ОПК-2  ОПК-3  ПК-3 |  |
| 2 | Раздел 2 | 6-10 | 5 | 10 |  |  | КИ, 10 | 25 | ОПК-1  ОПК-2  ОПК-3  ПК-3 |  |
|  | Зачет |  |  |  |  |  | З | 10 |  | ОПК-1  ОПК-2  ОПК-3  ПК-3 |
|  | Итого за 8 семестр |  |  |  |  |  |  | 100 |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Недели** | **Содержание / Темы занятий** | **Лек., час.** | **Пр./сем., час.** | **Лаб., час.** | **Компетенции по темам, проверяемые при текущем контроле** | **Виды тек.контроля по проверке компетенций** | **Компетенции по темам, проверяемые на зач. /экз.** |
| 8 семестр | | | | | | | |
| 1 | **Тема 1. Исходные определения.** Введение. Структуры данных. Типы. Аппликация. Абстракция. Выражение в форме оператор/операнд. | 1 | 2 |  | ОПК-1  ОПК-2  ОПК-3  ПК-3 |  | ОПК-1  ОПК-2  ОПК-3  ПК-3 |
| 2 | **Тема 2. Абстрактный язык функционального программирования.** Нотация (клозы, образцы, охрана). Операционная семантика. Функции высших порядков. | 1 | 2 |  | ОПК-1  ОПК-2  ОПК-3  ПК-3 |  | ОПК-1  ОПК-2  ОПК-3  ПК-3 |
| 3 | **Тема 3. Приемы программирования.** Разбор случаев. Введение подфункций. Накапливающий параметр. Хвостовая рекурсия. Итерационная форма программ. | 1 | 2 |  | ОПК-1  ОПК-2  ОПК-3  ПК-3 |  | ОПК-1  ОПК-2  ОПК-3  ПК-3 |
| 4 | **Тема 4. Введение в Лисп.** Атомы и списки. Точечное представление. Базовые функции обработки данных и примитивы управления. Связывание, деструктивное присваивание. Имя и значение символа. Определение функций. Применяющие функционалы. Замыкания. Макросы. Ввод и вывод. | 1 | 2 |  | ОПК-1  ОПК-2  ОПК-3  ПК-3 |  | ОПК-1  ОПК-2  ОПК-3  ПК-3 |
| 5 | **Тема 5. Синтаксически-ориентированное программирование .** Язык абстрактного описания областей. Построение абстрактного описания (синтаксиса) списков, списочных структур, деревьев, бинарных деревьев. Построение типовых алгоритмов обработки данных для этих областей. | 1 | 2 |  | ОПК-1  ОПК-2  ОПК-3  ПК-3 |  | ОПК-1  ОПК-2  ОПК-3  ПК-3 |
| 6 | **Тема 6. Доказательства свойств программ.** Определение индукции по структурам данных. Списочная индукция, sexpr-индукция. Примеры. Ограничения. | 1 | 2 |  | ОПК-1  ОПК-2  ОПК-3  ПК-3 |  | ОПК-1  ОПК-2  ОПК-3  ПК-3 |
| 7 | **Тема 7. Тематические основания.** Формальная система. Аксиоматезируемость. Синтаксическое и семантическое равенство. Элементы лямбда-исчисления (выражение, подстановка, аксиомы, правила вывода, стратегии редукции, нормальная форма, Теорема Черча-Россера, пародоксальный комбинатор). | 1 | 2 |  | ОПК-1  ОПК-2  ОПК-3  ПК-3 |  | ОПК-1  ОПК-2  ОПК-3  ПК-3 |
| 8 | **Тема 8. Трансформации программ.** Эквивалентные и корректные преобразования. Система преобразований Берсталла и Дарлингтона: instantiation, folding, unfolding, low, abstraction. Трансформационный синтез программ и его проблемы. | 1 | 2 |  | ОПК-1  ОПК-2  ОПК-3  ПК-3 |  | ОПК-1  ОПК-2  ОПК-3  ПК-3 |
| 9 - 10 | **Тема 9. Частичные вычисления.** Остаточная программа. Интерпретатор. Компилятор. Генератор компиляторов. Проекции Футамуры. Принципы построения частичных вычислителей. | 2 | 4 |  | ОПК-1  ОПК-2  ОПК-3  ПК-3 |  | ОПК-1  ОПК-2  ОПК-3  ПК-3 |

Семинарские занятия (1 семестр)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Недели** | **Темы занятий** | **Компетенции по темам, проверяемые при текущем контроле** | **Виды тек.контроля по проверке компетенций** | **Компетенции по темам, проверяемые на зач. /экз.** |
| 1 | **Элементы программирования на абстрактном функциональном языке.** Элементы программирования на абстрактном функциональном языке. | ОПК-1  ОПК-2  ОПК-3  ПК-3 |  | ОПК-1  ОПК-2  ОПК-3  ПК-3 |
| 2 | **Программирование функций для работы со списками на Haskell.** Программирование функций для работы со списками на Haskell. | ОПК-1  ОПК-2  ОПК-3  ПК-3 |  | ОПК-1  ОПК-2  ОПК-3  ПК-3 |
| 3 | **Программирование функций вычисления сумм рядов** Программирование функций вычисления сумм рядов | ОПК-1  ОПК-2  ОПК-3  ПК-3 |  | ОПК-1  ОПК-2  ОПК-3  ПК-3 |
| 4 - 5 | **Программирование аналитических преобразований.** Программирование аналитических преобразований. | ОПК-1  ОПК-2  ОПК-3  ПК-3 |  | ОПК-1  ОПК-2  ОПК-3  ПК-3 |
| 6 | **Программирование аналитических преобразований.** Программирование аналитических преобразований. | ОПК-1  ОПК-2  ОПК-3  ПК-3 |  | ОПК-1  ОПК-2  ОПК-3  ПК-3 |
| 7 | **Программирование синтаксических анализаторов** Программирование синтаксических анализаторов | ОПК-1  ОПК-2  ОПК-3  ПК-3 |  | ОПК-1  ОПК-2  ОПК-3  ПК-3 |
| 8 | **Доказательство свойств функций.** Доказательство свойств функций. | ОПК-1  ОПК-2  ОПК-3  ПК-3 |  | ОПК-1  ОПК-2  ОПК-3  ПК-3 |
| 9 - 10 | **Реализация семестровых проектов по индивидуальным заданиям.** Реализация семестровых проектов по индивидуальным заданиям. | ОПК-1  ОПК-2  ОПК-3  ПК-3 |  | ОПК-1  ОПК-2  ОПК-3  ПК-3 |

**Соответствие оценочных средств видам контроля**

|  |  |
| --- | --- |
| **Вид контроля** | **Наименование оценочного средства (способ оценки: устно/ письменно /комп.технолог.)** |
| КР | Выполнение контрольно-тестовой работы (письменно) |
| ТДЗ | Выполнение тематического домашнего задания (дистанционно) |
| КИ | Контроль по итогам выполнения (интегральная оценка без проведения дополнительного контроля) |
| З | Зачет по курсу «Функциональное программирование» (устно, с применением персональных компьютеров) |

|  |
| --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ |
| федеральное государственное автономное образовательное учреждение  высшего образования |
| **«Национальный исследовательский ядерный университет «МИФИ»** |
| ИНСТИТУТ ИНТЕЛЛЕКТУАЛЬНЫХ КИБЕРНЕТИЧЕСКИХ СИСТЕМ |
| КАФЕДРА КИБЕРНЕТИКИ (№ 22) |

**Задачи, предлагаемые на зачете по дисциплине**

**«Функциональное программирование»**

|  |  |  |
| --- | --- | --- |
| Составитель | Демидов Дмитрий Витальевич |  |
| Учебный год | 2019 |  |

1. Написать функцию, которая из списка английских символов удаляет все гласные буквы

2. Написать две функции. Первая из списка символов составляет строку. Вторая – из строки

составляет список символов

3. Создать бесконечный список из чисел: [0, 1, 0, 2, 0, 3, 0, 4, ...]

4. Создать бесконечный список нечѐтных чисел: [1, 3, 5, 7, 9, ... ]

5. Создать бесконечный список: [“a”, “aa”, “aaa”, “aaaa”, ... ]

6. Написать функцию, которая проверяет два списка на идентичность

7. Написать функцию, которая для элементов из списка чисел произвольной длины [a, b, c, d,

e, f, g, h, ...] вычисляет следующее выражение: sin a - cos b + sin c - cos d + sin e - cos f + ...

8. Написать функцию, которая для элементов из списка чисел произвольной длины [a, b, c, d,

e, f, g, h, ...] вычисляет следующее выражение: (a \* b) + (c \* d) + (e \* f) + (g \* h) + ...

9. Написать функцию, которая для элементов из списка чисел произвольной длины [a, b, c, d,

e, f, g, h, ...] создаѐт ноый список: [ a+b, b+c, c+d, d+e, e+f, f+g, ...]

10. Написать функцию, которая для элементов из списка чисел произвольной длины [a, b, c, d, e, f, g, h, ...] вычисляет следующее выражение: (a \* c) + (b \* d) + (e \* g) + (f \* h) + ...

11. Написать функцию, которая ищет в списке первые два элемента, большие 0 и возвращает

их сумму

12. Написать функцию, которая находит максимальный и минимальный элемент в списке чисел и меняет их местами

13. Дан список списков целых чисел. Написать функцию, которая проверяет, верно ли утверждение, что в этом списке существует подсписок, в котором все элементы делятся на 10.

14. Дан список списков целых чисел. Написать функцию, которая проверяет, верно ли,

утверждение, что в этом списке все подсписки содержат как чѐтные, так и нечѐтные числа.

15. Написать функцию, которая бы удаляла в списке повторяющиеся подряд элементы. Например, для списка ["a", "b", "b", "c", "b", "b", "b"], результатом работы был бы список ["a", "b", "c", "b"]

16. Написать функцию умножения элементов двух списков, которая возвращает список,

составленный из произведения элементов списков-параметров. Учесть, что переданные списки могут быть разной длины.

17. Написать функцию, которая удаляет из заданного списка целых чисел все четные числа.

18. Написать функцию, которая удаляет из заданного списка целых чисел все числа, меньшие

заданного.

19. Написать функцию, которая меняет знак всех положительных элементов списка чисел.

Например: по [-1, 0, 5, -10, -20] дает [-1, 0, -5, -10, -20]

20. Написать функцию, которая дублирует элементы списка. Например, для списка ["a", "b", "c"] результатом должен быть список ["a", "a", "b", "b", "c", "c"]

21. Написать функцию конкатенации списка списков, т.е. для списка [[“a”, ”b”, ”c”], [“d”, ”e”, ”f”], [“g”, ”h”]] результат должен быть следующим: [“a”, ”b”, ”c”, “d”, ”e”, ”f”, “g”, ”h”]

22. Написать функцию, сортирующую список строк по длине строки.

23. Написать функцию вычисления геометрического среднего элементов списка вещественных чисел

24. Написать функцию, которая ищет в списке элемент, который в данном списке встречается

три раза подряд.

25. Написать функцию, которая создаѐт новый список на основе существующего, только без

повторяющихся элементов

26. Написать функцию, которая на основе существующего списка создаѐт новый, в котором

находятся повторяющиеся элементы исходного

27. Пусть в списке символов содержатся только символы „[„, „]„, „(„ и „)„. Опишите функцию, которая для данного списка проверяет, что в ней находится правильная скобочная

последовательность, и возвращает true или false. Например, для списка [„[„; „[„; („; „)‟; „]‟; „]‟; „(„; „)‟] возвращает true, а для [„(„; „[„; „]‟] или для [„[„; „(„; „]‟; „)‟] - false.

28. Пусть мы представляем матрицы с помощью списков из списков. Описать функцию,

которая для данного n возвращает такой список для матрицы n на n, содержащий числа от 1 до n, расположенные "уголком". Например, для n = 4 надо вернуть:

[[1; 2; 3; 4];

[2; 2; 3; 4];

[3; 3; 3; 4];

[4; 4; 4; 4]]

29. Написать функцию, которая возвращает список из всех способов выдать сумму в n рублей

монетами достоинством в 1, 2 и 5 рублей. В списке находятся кортеж из трѐх чисел, где первое отвечает за кол-во монет в 1 рубль, второе – за кол-во в два рубля и третье – за кол-во в 5 рулей.

30. Опишите функцию, которая для данного числа n возвращает список из всех строк длины n, состоящих из чисел 1,2,3. причем сумма двух соседний чисел не больше 4. Например, при n=2 функция должна вернуть список [[1,1], [1,2], [1,3], [2,1], [2,2], [3,1]]

31. Дан тип, описывающий версии программы:

type version = {major: int; minor: int; release: int; build: int}

Написать функцию, которая бы сортировала список версий программ. Сначала по старшей версии, потом по младшей, затем по номеру релиза, и под конец по номеру билда. Например для такого списка:

let list = [{major = 2; minor = 3; release = 12; build = 1182};

{major = 2; minor = 1; release = 22; build = 1382};

{major = 1; minor = 7; release = 52; build = 5182};

{major = 2; minor = 1; release = 21; build = 1223};

{major = 2; minor = 3; release = 12; build = 1196}]

правильным результатом сортировки являлся бы такой:

[{major = 1; minor = 7; release = 52; build= 5182};

{major = 2; minor = 1; release = 21; build = 1223};

{major = 2; minor = 1; release = 22; build = 1382};

{major = 2; minor = 3; release = 12; build = 1182};

{major = 2; minor = 3; release = 12; build = 1196}]

32. Написать функцию, которая для данного списка [x1, x2, x3, x4, ...] возвращает список из

частичных сумм: [x1; x1+x2; x1+x2+x3; x1+x2+x3+x4; ... ]

33. Написать функцию, которая из списка чисел из n элементов выбирает n-1 элемент, которые при перемножении дают максимальный результат

34. Создать бесконечный список: [1; 2; 2; 3; 3; 3; 4; 4; 4; 4; ...]

35. Создать бесконечный список: [1; cos 1; cos cos 1; cos cos cos 1; cos cos cos cos 1; ... ]

36. Создать бесконечный список простых чисел: [1; 2; 3; 5; 7; 11; ...]

37. Опишите функцию, которая для данных чисел n и k возвращает список из всех последовательностей длины k, таких, что каждая последовательность состоит из целых чисел, все числа не меньше 1 и не больше n. И в последовательности чередуются четные и нечетные числа - т.е. в ней не должно быть рядом двух четных или двух нечетных чисел.

Например, для n = 3 и k = 3 результатом должно быть: [[1; 2; 1]; [1; 2; 3]; [2; 1; 2]; [2; 3; 2]; [3; 2; 1]; [3; 2; 3]]

38. Написать функцию, которая для данного числа n возвращает список из всех строк длины n, состоящих из символов 'а', 'b' и 'c'. Например, при n=2 функция должна вернуть список ["aa", "ab", "ac", "ba", "bb", "bc", "ca", "cb", "cc"].

39. Написать функцию, которая для данного списка [x1, x2, x3, x4, ...] возвращает список из

частичных средних: [x1; (x1+x2)/2; (x1+x2+x3)/3; (x1+x2+x3+x4)/4; ... ]

40. Создать список из всех возможных перестановок таких, что каждая из этих перестановок

состоит из 4 чисел, а каждое из этих чисел лежит в диапазоне от 1 до 6

41. Даны два типа:

type name = {id: int; name: string}

type surname = {id: int; surname: string }

Составить список имѐн и фамилий с одинаковым идентификатором

42. Написать функцию, вычисляющую скалярное произведение двух списков

43. Написать функцию quicksort, осуществляющая быструю сортировку списка, которая

принимает на вход, кроме исходного списка ещѐ и функцию сравнения типа 'a -> 'a -> bool и

сортирует список в соответствие с нею

44. Создать бесконечный список из чисел: [1; -1; 1; -1; 1; -1; 1; -1; ...]

С помощью этого списка и списка seq [1; 1..] создать бесконечный список: [1; -2; 3; -4; 5; -6 ...]

45. Описать функцию, которая ищет в списке первый положительный элемент, и возвращает

пару из его значения и оставшегося (не просмотренного) куска списка. С помощью этой функции найти сумму всех положительных элементов данного списка.

46. Создать список из всех возможных перестановок таких, что каждая из этих перестановок

состоит из 4 чисел, сумма которых не превышает 30

47. Создать список из всех возможных перестановок таких, что каждая из этих перестановок

состоит из 4 чисел, а каждое последующее число в перестановке больше предыдущего

48. Какой будет результат выполнения функции: fmap (+2) (\*3)?

|  |
| --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ |
| федеральное государственное автономное образовательное учреждение  высшего образования |
| **«Национальный исследовательский ядерный университет «МИФИ»** |
| ИНСТИТУТ ИНТЕЛЛЕКТУАЛЬНЫХ КИБЕРНЕТИЧЕСКИХ СИСТЕМ |
| КАФЕДРА КИБЕРНЕТИКИ (№ 22) |

**Варианты тематических домашних заданий (ТДЗ)**

**по дисциплине**

**«Функциональное программирование»**

|  |  |  |
| --- | --- | --- |
| Составитель | Демидов Дмитрий Витальевич |  |
| Учебный год | 2019 |  |

**Введение**

Приём и контроль сдачи ведётся по адресу: http://mephi.eu01.aws.af.cm/labs. Для регистрации требуется выполнить задание:

Следующая последовательность чисел называется треугольником Паскаля:

1

1 1

1 2 1

1 3 3 1

1 4 6 4 1

Числа по краям всегда 1, а каждое число внутри треугольника — сумма двух вышестоящих чисел. Нужно написать функцию, которая рекурсивно вычисляет элементы треугольника Паскаля. Она принимает номер колонки и номер строки, считая от нуля, и возвращает число на их пересечении в треугольнике. К примеру, pascal 0 2 = 1, pascal 1 2 = 2 и pascal 1 3 = 3.

В файле шаблона нужно также вписать адрес своей почты и фамилию с инициалами. Посмотреть, что у вас получается можно с помощью функции printIt (int -> string), которая выводит n-строчек треугольника в списке. После того как вы всё напишете и проверите, вызывайте функцию main, которая отправит постом результаты вычисления первых 20 строчек. Отправлять можно неограниченное число раз, адрес почты является первичным ключом.

# Шаблон задания

|  |
| --- |
| {-# LANGUAGE OverloadedStrings #-}  import Control.Monad  import Network.HTTP.Conduit  import Data.Text.Encoding  import qualified Data.ByteString.Lazy as L  import qualified Data.ByteString.Char8 as C  import Network (withSocketsDo)  (email, name) = ("", encodeUtf8 "") -- адрес почты и фамилия с инициалами  pascal :: Int -> Int -> Int  pascal c r = 1 -- а тут решение  printIt :: Int -> C.ByteString  printIt n = C.pack $ show $ [pascal y x | x <- [0..n], y <- [0..x]]  main :: IO()  main =  withSocketsDo $ do  initReq <- parseUrl "http://mephi.eu01.aws.af.cm/lab0"  let req = urlEncodedBody [("email", email), ("name", name), ("lang", "haskell"), ("content", printIt 20)] $ initReq { method = "POST" }  response <- withManager $ httpLbs req  L.putStr $ responseBody response |

**ТДЗ № 1**

# Задание 1

Составить программу на функциональном языке, которая печатает таблицу значений элементарной функции, вычисленной двумя способами: по формуле Тейлора и с помощью встроенных функций языка программирования. В качестве аргументов таблицы взять точки разбиения отрезка [a,b] на n равных частей (n + 1 точка, включая концы отрезка), находящихся в рекомендованной области хорошей точности формулы Тейлора. Вычисления по формуле Тейлора проводить двумя способами: по экономной в сложностном смысле схеме (с сохранением и домножением предыдущего слагаемого) и «в лоб» вычислением n-го коэффициента. Число слагаемых ряда определяется достижением заданной точности вычислений. Результат должен быть напечатан в виде таблицы примерно следующего вида:

| x | Первый способ | Кол-во итераций | Второй способ | Кол-во итераций | Значение функции |
| --- | --- | --- | --- | --- | --- |
| 0.00 | ... |  | 0.0 | ... |  |
| 0.05 |  |  | 0.0008 | ... |  |
| ... | ... |  | ... | ... |  |
| 0.50 | ... |  | ... | ... |  |

Варианты задания в соответствии с номером в группе указаны в таблице 1.

# Задание 2

Составить программу на функциональном языке для решения трансцендентных алгебраических уравнений различными численными методами (итераций, Ньютона и половинного деления). Нелинейные уравнения оформить как параметры-функции, разрешив относительно неизвестной величины в случае необходимости. Применить каждую процедуру к решению трех уравнений, заданных тремя строками таблицы, начиная с варианта с заданным номером. Каждое уравнение решать всеми применимыми методами. Варианты задания содержатся в таблице 2. Нужно подститать количество итераций для каждого метода.

# Варианты задания

Таблица 1

Функции и их разложение в ряд Тейлора на указанном отрезке

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **ряд** | **a** | **b** | **функция** |
| 1 |  | -1.0 | 1.0 |  |
| 2 |  | 0.0 | 0.5 |  |
| 3 |  | -0.2 | 0.3 |  |
| 4 |  | -1.0 | 1.0 |  |
| 5 |  | 0.0 | 0.5 |  |
| 6 |  | 0.0 | 1.0 | sh *x* |
| 7 |  | 0.0 | 0.5 |  |
| 8 |  | 0.0 | 2.0 |  |
| 9 |  | 0.0 | 0.5 |  |
| 10 |  | 0.0 | 1.0 |  |
| 11 |  | 0.1 | 0.6 |  |
| 12 |  | 0.0 | 1.0 |  |
| 13 |  | 0.0 | 1.0 |  |
| 14 |  | 0.1 | 0.6 |  |
| 15 |  | 0.0 | 1.0 |  |
| 16 |  | 0.0 | 1.0 |  |
| 17 |  | 0.2 | 0.7 |  |
| 18 |  | 0.1 | 0.6 |  |
| 19 |  | 0.1 | 0.6 | ch *x* |
| 20 |  | 0.1 | 0.6 |  |
| 21 |  | 0.1 | 0.6 |  |
| 22 |  | 0.0 | 1.0 |  |
| 23 |  | 0.0 | 0.5 |  |
| 24 |  | 0.0 | 1.0 |  |
| 25 |  | 0.0 | 1.0 |  |
| 26\* |  |  |  |  |
| 27\* |  | 0.1 | 0.6 |  |
| 28**\*** |  |  |  |  |

Таблица 2

Трансцендентные алгебраические уравнения и рекомендуемые методы решения

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Уравнение | Отрезок, содержащий корень | Базовый метод | Приближенное значение корня |
| 1 |  | [3, 4] | Ньютона | 3.5265 |
| 2 |  | [1, 2] | дихотомии | 1.0804 |
| 3 |  | [1, 1.5] | итераций | 1.1474 |
| 4 |  | [1, 3] | Ньютона | 2.0692 |
| 5 |  | [0, 1] | дихотомии | 0.5768 |
| 6 |  | [0.5, 1] | итераций | 0.9892 |
| 7 |  | [1, 3] | Ньютона | 1.8832 |
| 8 |  | [2, 3] | дихотомии | 2.4200 |
| 9 |  | [2, 3] | итераций | 2.0267 |
| 10 |  | [0.4, 1] | Ньютона | 0.6533 |
| 11 |  | [-1, 0] | дихотомии | -0.2877 |
| 12 |  | [2, 3] | итераций | 2.8459 |
| 13 |  | [0.2, 1] | Ньютона | 0.5472 |
| 14 |  | [1, 2] | дихотомии | 1.0769 |
| 15 |  | [1, 2] | итераций | 1.2388 |
| 16 |  | [2, 3] | итераций | 2.2985 |
| 17 |  | [0, 2] | Ньютона | 1.0001 |
| 18 |  | [0.4, 1] | дихотомии | 0.7376 |
| 19 |  | [0, 0.85] | итераций | 0.2624 |
| 20 |  | [1, 2] | Ньютона | 1.1183 |
| 21 |  | [0, 0.8] | дихотомии | 0.3333 |
| 22 |  | [0, 1] | итераций | 0.5629 |
| 23 |  | [2, 4] | Ньютона | 3.23 |
| 24 |  | [1, 2] | дихотомии | 1.8756 |
| 25 |  | [0, 1] | итераций | 0.7672 |
| 26 |  | [0, 1] | Ньютона | 0.8814 |
| 27 |  | [1, 3] | дихотомии | 1.3749 |
| 28 |  | [1.2, 2] | итераций | 1.3077 |

# Шаблон кода для первого и второго заданий:

|  |
| --- |
| {-# LANGUAGE OverloadedStrings #-}  module Lab1 (fTailor, tailor, tailorA, fSolve, iter, newton, dichotomy) where  import System.IO  import System.Environment  import System.Directory  import Control.Monad  import Network.HTTP.Conduit  import qualified Data.ByteString.Lazy as L  import qualified Data.ByteString.Char8 as C  import Network (withSocketsDo)  -- почтовый адрес  email = ""  -- общий тип для возвращаемых вашими функциями значений, где первая часть кортежа это само значение функции, вторая - кол-во операций  type Result = (Float, Integer)  fTailor x = x -- функция, которую раскладываем  delta = 1e-10  (n, a, b) = (20, -1, 1) -- интервал  tailor x = (x, 0)  tailorA :: Float -> Result  tailorA x = (x, 0)  printTailor = mapM\_ putStrLn $  map  (\ x ->  let ((firstRes, firstCou), (secondRes, secondCou)) = (tailor x, tailorA x)  in show x ++ "\t" ++ show firstRes ++ "\t" ++ show firstCou ++ "\t" ++ show secondRes ++ "\t" ++ show secondCou ++ "\t" ++ show (fTailor x))  [a, a + (b - a) / n .. b]  -- \*\*\* Вторая часть  fSolve = \x -> x -- функция, решение которой ищем  iter :: (Float -> Float) -> Float -> Float -> Result  iter f a b = (42, 0)  newton :: (Float -> Float) -> Float -> Float -> Result  newton f a b = (42, 0)  dichotomy =  --для функций с аккумулятором удобно ставить его в начало  let dichotomyA i f a b = (42, 0)  in dichotomyA 0 -- чтобы воспользоваться каррированием  printSolve =  mapM\_ putStrLn $ map (\f -> show $ f fSolve a b) [iter, newton, dichotomy]  main = do  withSocketsDo $ do  dir <- getCurrentDirectory  initReq <- parseUrl "http://mephi.eu01.aws.af.cm/lab1"  handle <- openFile (dir ++ "/Lab1.hs") ReadMode  hSetEncoding handle utf8\_bom  content <- hGetContents handle  let req = urlEncodedBody [("email", email), ("content", C.pack content)] $ initReq { method = "POST" }  response <- withManager $ httpLbs req  hClose handle  L.putStrLn $ responseBody response |

ТДЗ № 2

# Описание

В этой лабораторной предстоит работать со списками, полученными в результате разбора веб-страниц. Для этих целей используются библиотеки [FSharp.Data](https://fsharp.github.io/FSharp.Data/library/HtmlParser.html) и [\*-conduit](https://github.com/snoyberg/xml). Они позволяют изящно разбирать и запрашивать данные из таких иерархических структур как json, xml, html. В нашем случае мы будем работать с html-страницами, запрашивать содержимое тэгов с помощью селекторов и получать нужные нам данные в последовательностях и списках соответственно.

Примеры работы с библиотеками для каждого из языков находятся в соответствующих boilerplate-файлах: [Lab2.fs](https://github.com/agsh/mipt15-1/blob/master/lab2/Lab2.fs) и [Lab2.hs](https://github.com/agsh/mipt15-1/blob/master/lab2/Lab2.hs). И в интернете: [FSharp.Data](https://fsharp.github.io/FSharp.Data/library/HtmlParser.html), [html-conduit](https://www.fpcomplete.com/school/starting-with-haskell/libraries-and-frameworks/text-manipulation/tagsoup)

# Установка библиотек

С использованием cabal: cabal install xml-conduit http-conduit html-conduit

# Задание

| Вариант | Задание |
| --- | --- |
| 1 | По списку языков программирования википедии (http://en.wikipedia.org/wiki/List\_of\_programming\_languages) составить список императивных, не функциональных ЯП. |
| 2 | По списку языков программирования википедии (http://en.wikipedia.org/wiki/List\_of\_programming\_languages) составить список кортежей: год появления, названия. Языки без указания годов появления исключить. |
| 3 | По списку телефонных номеров МИФИ (http://mephi.ru/about/governance.php) выяснить, кто делит один номер с коллегами. Телефонные номера нормализовать |
| 4 | Узнать, в каком году было больше и меньше всего выпускников факультета КиБ |
| 5 | Составить список 50 самых умных и самых строгих преподавателей (http://www.mephist.ru/mephist/prepods.nsf/teachers) |
| 6 | Составить список из ФИО преподавателей (http://cyber.mephi.ru/Faculty.html) и их страниц в соц.сетях (linkedin, facebook, vk) |
| 7 | Составить список выпускников МИФИ, которые любят КВН (https://vk.com/vmephi) |
| 8 | Составить список 50 самых комментируемых преподавателей (http://www.mephist.ru/mephist/prepods.nsf/teachers) |
| 9 | Составить список 50 комментариев для преподавателей с самой высокой оценкой (http://www.mephist.ru/mephist/prepods.nsf/teachers) |
| 10 | Найти средний возраст участников студсовета МИФИ (https://vk.com/vmephi) |
| 11 | Отсортировать список git-проектов Haskell, которые проходят тесты на Travis CI по количеству веток (https://github.com/haskell) |
| 12 | Выяснить, какой процент из топ-500 проектов на языках Haskell и F# на github с наибольшим количеством звёзд составляют проекты на F# (https://github.com/search?utf8=%E2%9C%93&q=language%3AF%23&type=Repositories&ref=advsearch&l=F%23) |
| 13 | Составить частоту обновлений проектов на F# на github (в течении последнего месяца, двух, и т. д.) (https://github.com/search?utf8=%E2%9C%93&q=language%3AF%23&type=Repositories&ref=advsearch&l=F%23) |
| 14 | Узнать, от какой библиотеки зависит больше всего запрещённых пакетов на hackage (http://hackage.haskell.org/packages/deprecated) |
| 15 | Найти пять самых скачиваемых пакетов за всё время на hackage (http://hackage.haskell.org/packages/top) |
| 16 | Какой процент составляют issue с комментариями от общего количества issue для NuGet? (https://github.com/nuget/home/issues?page=1&q=is%3Aissue+is%3Aopen) |
| 17 | Какое количество разработчиков NuGet уже перебазировались на github, если судить по их никам? (https://nuget.codeplex.com/team/view) |
| 18 | Кто из разработчиков Microsoft на github ведёт блог на blogs.msdn.com и указал его в профиле? (https://github.com/Microsoft) |
| 19 | Попытаться найти по списку языков программирования википедии человека, который разработал больше всего ЯП (http://en.wikipedia.org/wiki/List\_of\_programming\_languages) |
| 20 | Какое количество ссылок на странице о LISP ведут на англоязычные ресурсы, а какие - на русскоязычные? (https://www.linux.org.ru/wiki/en/%D0%A7%D0%B0%D1%81%D1%82%D1%8C\_1.\_%D0%9E%D0%B1%D1%89%D0%B8%D0%B5\_%D0%B2%D0%BE%D0%BF%D1%80%D0%BE%D1%81%D1%8B\_%D0%BE\_Lisp) |
| 21 | Под какой лицензией выпущено большинство проектов на F# на github, если судить по файлу LICENSE? (https://github.com/search?utf8=%E2%9C%93&q=language%3AF%23&type=Repositories&ref=advsearch&l=F%23) |
| 22 | Сколько пакетов в Hackage (http://hackage.haskell.org/packages/) относятся к нескольким категориям? |
| 23 | У какого факультета МИФИ (http://mephi.ru/about/faculty/) больше всего кафедр? |
| 24 | Сколько музыкальных групп всего с одним треком находится в чарте last.fm top100 композиций по РФ? (http://www.last.fm/charts/tracks/top/place/Russian+Federation?limit=100) |
| 25 | Какие группы появлялись в чартах last.fm с 2008 по 2013 года (исключая 2012) больше всего раз? (http://www.last.fm/bestof/2008 - http://www.last.fm/bestof/2013) |
| 26 | Кто, кроме anonymous'а оставил больше всего комментариев в теме https://www.linux.org.ru/news/google/11404954? |
| 27 | Какой тэг самый популярный среди ста самых популярных пакетов на NuGet? (https://www.nuget.org/stats/packages) |
| 28 | В каком месяце какого года было больше всего вопросов в рассылке эрланга? (http://erlang.org/pipermail/erlang-questions/) |

# Шаблон кода для задания

|  |
| --- |
| {-# LANGUAGE OverloadedStrings #-}  import Network.HTTP.Conduit (simpleHttp) -- из пакета http-conduit  import qualified Data.ByteString.Lazy.Char8 as L -- если захотим работать с данными, полученными из simpleHttp  import qualified Data.Text as T -- представление юникодной строки в хаскелле  import Data.Text (replace) -- берём из модуля Data.Text только функцию replace (многие ф-ии из него перекрывают станартные из Prelude)  import Text.HTML.DOM (parseLBS) -- из пакета html-conduit  import Text.XML.Cursor (Cursor, attributeIs, content, element, fromDocument, child, ($//), (&|), (&//), (&/), (>=>)) -- из пакета xml-conduit, для работы с DOM-деревом документа  import Network (withSocketsDo) -- workaround для windows  n = T.pack "\r\n" -- перевод строки в плохой вёрстке сайта cyber.mephi.ru  url = "http://cyber.mephi.ru/Faculty.html" -- преподаватели кафедры  {-  Аналог XPath записи для поиска нужной нам информации. Выбираются все теги td с бэкграундом, внутри которых находятся ссылки  &// Ищет среди всех потомков, а &/ - в детях  -}  findNodes :: Cursor -> [Cursor]  findNodes = element "TD" >=> attributeIs "bgcolor" "#e1e4e6" &// element "B" &// element "A" >=> child  {-  Извлечь контент из узла. Для извлечения ссылок на страницы преподавателей нужно использовать функцию attribute  -}  extractData :: Cursor -> T.Text  extractData = T.concat . content  {-  Извлекаем содержимое страницы, парсим её и возвращаем курсор на корень DOM-дерева  -}  cursorFor :: String -> IO Cursor -- тут тип важен  cursorFor u = do  page <- withSocketsDo $ simpleHttp u  return $ fromDocument $ parseLBS page  main :: IO()  main = withSocketsDo $ do  nodes <- lab2  dir <- getCurrentDirectory  initReq <- parseUrl "http://mephi.eu01.aws.af.cm/lab2"  handle <- openFile (dir ++ "/Lab2.hs") ReadMode  hSetEncoding handle utf8\_bom  content <- hGetContents handle  let req = urlEncodedBody [("email", email), ("result", encodeUtf8 $ T.concat $ nodes), ("content", encodeUtf8 $ T.pack content) ] $ initReq { method = "POST" }  response <- withManager $ httpLbs req  hClose handle  L.putStrLn $ responseBody response |

ТДЗ № 3

# Реализовать

1. Тип данных для представления JSON-объектов. Будем ещё его называть деревом, так как по сути это оно и есть, и задания больше относятся к работе с деревом, нежели с JSON
2. Разбор JSON-строки в объект
3. Функцию обработки древовидной структуры в соответствии с вариантом задания
4. Функцию генерации случайного JSON-объекта
5. Сериализацию объекта обратно в строку

Для получения последовательности случайных чисел можно использовать генератор псевдослучайных чисел типа StdGen и функцию random, или функции получения случайных значений из внешнего мира randomIO, randomRIO и т.д. В первом случае сложность может возникнуть в правильной передаче нового состояния вашего генератора дальнейшим вычислениям. В этом случае удобно обернуть генерирующие функции в монаду State, маленький пример есть в шаблоне. Во втором случае придётся работать внутри монады IO, что может быть неприятно. Для многих заданий полезно причислить тип JSON к классу типов Monoid

# Тесты

Нужно написать тесты, в которых проверить работу программы на нескольких модельных примерах, показывающих правильность реализованного алгоритма, а также на случайно сгенерированном дереве.

# Определения

JSON-объекты представлены следующими типами (json.org):

* объект, содержащий пары ключ - значение (другой JSON-объект)
* массив из JSON-объектов
* притивные типы: null, number, string, true, false

Глубиной вершины дерева называется длина пути в эту вершину из корня. Глубиной (высотой) дерева называется максимальная глубина его вершин. Листом или терминальной вершиной дерева называется вершина, не имеющая поддеревьев. Степенью вершины называется число исходящих из неё ветвей. Степенью дерева называется максимальная степень его вершин. Шириной уровня дерева называется число вершин на данной глубине. Шириной дерева называется максимальная ширина по всем уровням. Подобие деревьев отличается от равенства возможным несовпадением значений данных, хранящихся в узлах.

# Варианты заданий

1. Найти количество элементов во всех массивах
2. Построить список, содержащий все ключи в объектах дерева
3. Определить ширину дерева
4. Определить максимальную глубину у числовых элементов
5. Определить глубину дерева
6. Определить степень дерева
7. Определить число нетерминальных вершин дерева (объекты и массивы)
8. Определить число вершин дерева, степень которых совпадает со степенью дерева
9. Определить уровень дерева, на котором находится максимальное число вершин
10. Определить значение листа дерева (примитвный тип), имеющего минимальную глубину
11. Определить значение нетерминальной вершины дерева с максимальной глубиной
12. Проверить, все ли числа в JSON находятся в заданном диапазоне
13. Узнать, подобны ли два JSON-объекта
14. Проверить монотонность возрастания ширины уровня дерева
15. Проверить монотонность возрастания длины массивов в зависимости от уровня их вложенности
16. Найти сумму всех чисел в JSON
17. Проверить, находятся ли все числа на одном уровне
18. Увеличить значения всех чисел в JSON на их уровень
19. Увеличить на единицу значения листьев дерева, находящихся на нечётных уровнях
20. Добавить произвольный примитивный элемент в дерево
21. Найти количество вхождений элемента примитивного типа в дерево
22. Удалить элемент примитивного типа из дерева
23. Проверить, является ли дерево симметричным (равным своему отражению, объекты в JSON не отражаются)
24. Проверить, является ли дерево самоподобным (подобным своему отражению, объекты в JSON не отражаются)
25. Написать функцию, которая объединяет два JSON-объекта или JSON-массива
26. Написать функцию, которая принимает JSON-объект и возвращает новый, где ключи — значения, а значения — ключи
27. Написать функцию, которая принимает список ключей и проверяет, все ли из них есть в JSON
28. Составить хеш: значение примитивного типа в JSON, сколько раз оно встречается

# Шаблон для задания

|  |
| --- |
| {-# LANGUAGE OverloadedStrings #-}  import System.IO  import System.Environment  import System.Directory  import System.Random  import Control.Monad  import Control.Monad.State  import Network.HTTP.Conduit  import qualified Data.ByteString.Lazy.Char8 as L  import Data.Text.Encoding  import qualified Data.Text as T  import Network (withSocketsDo)  -- почтовый адрес  email = ""  newtype JSON = Object [(String, JSON)]  -- добавим сответствующие классы типов для JSON  instance Show JSON where  show = stringify  instance Read JSON where  readsPrec \_ x = parse x  parse :: ReadS (JSON)  parse "{}" = [(Object [], "")]  lab3 (Object list) = 0  stringify (Object list) = "{}"  -- вариант с монадой IO  generateIO :: IO JSON  generateIO = do  num <- randomRIO (1, 2) :: IO Int  let json = case num of  1 -> Object [];  2 -> Object [("io", Object [])]  return json  -- чистый вариант с генератором, заключённым в состояние  -- мы храним в состоянии генератор, каждый раз используя  -- его, возвращаем в состояние новый  type GeneratorState = State StdGen  generate' :: GeneratorState JSON  generate' = do  gen <- get  let (num, newGen) = randomR (1, 2) gen :: (Int, StdGen)  let json = case num of  1 -> Object [];  2 -> Object [("pure", Object [])]  put newGen  return json  generate :: JSON  generate = evalState generate' (mkStdGen 0)  main :: IO()  main = withSocketsDo $ do  dir <- getCurrentDirectory  initReq <- parseUrl "http://mephi.eu01.aws.af.cm/lab3"  handle <- openFile (dir ++ "/Lab3.hs") ReadMode  hSetEncoding handle utf8\_bom  content <- hGetContents handle  let req = urlEncodedBody [("email", email), ("content", encodeUtf8 $ T.pack content) ] $ initReq { method = "POST" }  response <- withManager $ httpLbs req  hClose handle  L.putStrLn $ responseBody response |