Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования

«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)

Кафедра автоматизации обработки информации

ФУНКЦИОНАЛЬНОЕ И ЛОГИЧЕСКОЕ ПРОГРАММИРОВАНИЕ

Часть 1

(Функциональное программирование)

Методические указания к лабораторным работам и организации самостоятельной работы
для студентов направления 09.03.04
«Программная инженерия»
(уровень бакалавриата)

2018

Салмина Нина Юрьевна
Функциональное и логическое программирование. Часть 1 (Функциональное программирование): Методические указания к лабораторным работам и организации самостоятельной работы для студентов направления «Программная инженерия» (уровень бакалавриата) / Н.Ю. Салмина. – Томск, 2018. – 29 с.

Оглавление

1 Введение

Цель дисциплины – ознакомление студентов с основами функционального программирования и λ-исчистения на примере языка Лисп, формирование у студентов профессиональных знаний и практических навыков по разработке функциональных программ с использованием рекурсии, функционалов.

Изучение данной части дисциплины включает в себя: теоретический раздел (изучение теоретического материала); практический раздел (выполнение лабораторных и контрольных работ); итоговый контроль результата изучения дисциплины. Данное пособие содержит в себе методические указания и варианты заданий для лабораторных работ, вопросы по организации самостоятельной работы по первой части дисциплины: Функциональное программирование.

2 Методические указания к проведению лабораторных работ

2.1 Лабораторная работа «Основы языка Лисп»

Цель работы

Целью данной работы является знакомство со средой работы в LispIDE и получение первичных навыков работы в этой среде путем написания простейших выражений и собственных функций с использованием базовых функций Лисп, λ -выражений.

Рекомендации по подготовке к работе

Для выполнения данной и последующих лабораторных работ прежде всего необходимо ознакомиться с работой в среде LispIDE.

LispIDE является программой, предназначенной для работы с различными реализациями языка Лисп, не имеющими собственных оболочек. При первоначальном запуске программы необходимо произвести следующие действия: зайдите в пункт меню Settings/SetLispPath и выберите ту версию языка Лисп, в которой вы будете работать. Для выполнения практических заданий рекомендуется использовать язык CLisp версия 2.49 (файл clisp-2.49-win32-mingw-big.exe). После установки требуемой версии в нижнем (диалоговом) окне появится надпись, сообщающая о том, что требуемая программа загружена, и вы можете работать.

Нижнее окно предназначено для работы в диалоговом режиме. Этот режим удобен для проверки и выполнения простейших вычислимых выражений, а также для запуска тех функций, которые были уже откомпилированы.

Верхнее окно программы предназначено для работы с файлами: здесь можно открыть существующий файл с программами, создать новый файл, корректировать и сохранять программу. Откомпилировать выбранные функции можно с помощью кнопки «()» или команды Edit/Send to Lisp. После компиляции созданные функции можно вызывать в диалоговом окне.

Внимание!!! Данная среда не понимает пути, в которых используется кириллица! Если вы создаете новый файл, то ВСЕ промежуточные каталоги должны иметь латинские имена. Если вы хотите открыть существующий файл, то проследите, чтобы он находился в каталогах (на всем пути) с латинскими именами!

На данном практическом занятии вам необходимо выполнить три задания. При выполнении первого задания используйте суперпозиции функций CAR и CDR. Помните, что функция CAR определяет голову списка, а функция CDR выделяет хвост списка.

Внимательно следите за балансом скобок!

Для выполнения второго задания используйте лямбда-выражение.

При выполнении третьего задания используйте функцию проверки условий COND, а также, по необходимости, другие базовые функции Лисп.

Порядок проведения работы

1) В первом задании необходимо записать выражение, выбирающее выделенный элемент списка.
2) Во втором задании необходимо написать λ -выражение, описывающее условие, которое проверяет список и выдает истину, если выполняется хотя бы одно из заданных условий. Если ни одно из заданных условий не выполняется, выдать NIL. Проверить работу написанного λ-выражения путем его вызова с различными фактическими параметрами.
   Внимание! Использовать только базовые функции!
3) В третьем задании необходимо написать функцию, выполняющую заданные действия.

Варианты заданий

Задание 1

№ варианта	**Список**
1	(1 2 (3 **4**) 5 6 7)
2	(((q w e) **g** f) n)
3	(a (x c) (s v) **n** q)
4	((**1** 2 3) (4 5 6))
5	(9 8 (**7** 6 5) e w)
6	((**d**) f (h I k) z)
7	(3 (4 5) (**j** l))
8	(h z ((q) d k l) **a**)
9	((1) (**2**) (3) (4) (5))
10	(h ((**a** s d)) l k)
11	(0 (2 g a) **1** 5)
12	((b) **v** (1 2 3) z a)
13	(e ((**1**) (2)) a s)

14	(r f (**b)** c n z)
15	((1 2 3) s (**4** 5) f)
16	((d) f (h I k) **z**)
17	(1 (2 (3)) **4** 5 6)
18	(((q w e) **f**) d)
19	((1 2) (s d) (**w** f) 4 5)
20	(a ((**n** d) q) 5 z)
21	(q ((x)(m)) (**d**) h)
22	(((1 2) 3) e **r**)
23	(q w (**d** (s)) 3 f)

Задание 2

№ варианта	**Условия проверки**
1	а) список содержит всего два элемента; или б) список содержит три элемента.
2	а) первый и третий элемента списка одинаковы; или б) второй и третий элементы списка одинаковы
3	Хотя бы один из первых трех элементов является списком
4	Первый элемент списка является атомом или числом
5	Второй или третий элемент списка является числом
6	Первый элемент списка равен второму или Второй элемент списка равен третьему
7	второй или третий элементы списка являются списками
8	Первый элемент списка является числом или Третий элемент списка является списком
9	а) первый элемент списка равен 2; или б) второй элемент списка равен 3; или в) 3-й элемент списка = 4
10	Первый или третий элементы списка являются атомами
11	Список содержит ровно четыре элемента или Список пуст
12	Второй или третий элементы списка являются списками
13	Один из первых трех элементов списка является числом
14	Один из первых трех элементов списка является списком

15	а) первый и третий элементы равны; или б) первый и четвертый элементы равны; или в) второй элемент - список
16	Первые три элементы списка равны между собой
17	Первый элемент списка равен 0 или второй элемент списка равен 1 или третий элемент – равен 2
18	Первый и второй элементы списка – числа или второй элемент – список
19	Первый элемент списка атом или второй элемент списка – число
20	Список имеет больше трех элементов или список пуст
21	Первые два элемента списка являются числами ИЛИ в списке больше двух элементов
22	Первый или третий элемент списка является пустым списком
23	Второй элемент списка равен 2 и третий элемент списка является списком

Задание 3

1. Напишите функцию, осуществляющую циклическую перестановку элементов в списке: первый элемент списка функция переставляет в конец списка, т.е. (a s d f) => (s d f a)
2. Написать функцию, которая сравнивает первые два элемента списка. Если эти элементы равны, возвращается исходный список, если разные, то первый и второй элементы меняются местами.
3. Определите функцию (f a b c), которая равна истине тогда и только тогда, когда из отрезков a, b и c можно построить прямоугольный треугольник. Если нельзя, функция должна возвращать число, равное сумме трех чисел: a, b и c.
4. Написать функцию, которая возвращает сумму первого и последнего элементов числового списка, если они равны, а иначе – произведение этих элементов.
5. Определите функцию, которая меняет местами первый и последний элементы списка, оставляя остальные на своих местах, если элементы списка разные, иначе функция возвращает список без первого и последнего элементов.
6. Написать функцию двух аргументов. Если оба аргумента атомы – создать новый список из этих аргументов. Если какой-то аргумент является списком, то в новый список включать только первый элемент списка:

1 2 3 => (1 2)
1 (d f c) => (1 d)

7. Написать функцию с одним аргументом – списком. Если первые два элемента списка – списки, создать список из двух элементов: 1-й элемент 1-го списка + 1-й элемент 2-го списка. Иначе убрать из исходного списка первые два элемента.
8. Написать функцию, на вход которой подается три целых числа – сторона квадрата, длина и ширина прямоугольника. Функция должна определять, сколько квадратов с заданной стороной поместится в этот прямоугольник (например, при стороне=2, ширине=3 и длине=5, можно поместить 2 квадрата).
9. Создать функцию, на вход которой подается список Y и число N. Если первые три элемента списка Y являются числами, то функция возвращает список этих чисел, увеличенных в N раз. Если нет, то функция возвращает просто число N.
10. Напишите функцию двух аргументов (f q w), где аргументы являются списками. Функция возвращает t только в том случае, если первые два элемента этих списков соответственно равны друг другу. В остальных случаях функция должна выдавать nil.
11. Напишите функцию двух аргументов (f q w), где аргументы являются списками. Функция должна создавать список из двух заданных путем слияния, исключив из них первые два элемента.
12. Написать функцию от трех аргументов – списков. Функция должна возвращать 3, если в каждом списке не менее 3-х элементов, 2 – если в каждом списке не менее 2-х элементов и 1 – если в каждом списке есть по крайней мере один элемент. Если среди исходных списков есть пустой – функция возвращает NIL.
13. Написать функцию с одним аргументом – числовым списком, длиной не менее трех. Функция должна возвращать T, если первые три элемента списка – четные и NIL в противном случае.
14. Написать функцию с одним аргументом – числовым списком. Функция должна вернуть исходный список, уменьшив его первые два элемента вдвое, если количество элементов в списке меньше двух, функция должна возвращать пустой список.
15. Написать функцию с одним аргументом – списком. Функция должна возвращать 3, если первые два элемента списка – списки, 1 – если только первый элемент является списком, 2 –

если только второй элемент является списком и 0 – в противном случае.

16. Написать функцию с одним аргументом – списком. Функция должна возвращать список из трех элементов, выбираемых из первых трех элементов исходного списка следующим образом: если элемент исходного списка – атом, он добавляется в результирующий список. Если элемент исходного списка – список, из него выбирается первый элемент:
    ((a s) 2 3 4) ==> (a 2 3)
    ((q w e) (1 2 3) 4 5 (a)) ==> (q 1 4)
    (1 2 (a 3)) ==> (1 2 a)
17. Определить функцию, которая меняет местами элементы в исходном списке: 1-й и 2-й, 3-й и 4-й, 5-й и 6-й. Если количество элементов меньше 6, то менять местами 1-2, 3-4 (или только 1-2, если количество элементов меньше 4-х).
18. Написать функцию с одним аргументом - списком, которая бы меняла местами последние два элемента списка.
19. Определить функцию, на вход которой подается список и число. Если длина списка меньше заданного числа, то число добавляется в список в качестве первого элемента. Если больше, то просто возвращается исходный список.
20. Написать функцию, имеющую два аргумента: X – S-выражение и Y – список. Если X равен первому элементу Y, то он добавляется в конец списка Y, если нет – в начало.
21. Функция имеет три аргумента – действительные числа, которые являются параметрами квадратного уравнения. Функция должна возвращать nil, если уравнение не имеет корней и значение дискриминанта, если решение существует. Функция SQRT извлекает квадратный корень из аргумента.
22. Определить функцию, имеющую один аргумент – числовой список. Функция должна изменять исходный список следующим образом: первый элемент списка увеличивается в три раза, второй элемент уменьшается в два раза, остальные элементы остаются неизменными.
23. Написать функцию, имеющую три числовых аргумента A, B и C – коэффициенты квадратного уравнения. Функция должна находить корни квадратного уравнения и возвращать список из этих корней. Если уравнение не имеет корней, возвращать NIL.

2.2 Лабораторная работа «Рекурсивные функции»

Цель работы

Целью данной работы является знакомство с рекурсией и принципами построения рекурсивных функций.

Рекомендации по подготовке к работе

В процессе выполнения работы необходимо написать две функции, описанные в вашем варианте задания. При создании функций используйте рекурсию. Для этого вам необходимо сначала определить терминальную ветвь (при необходимости терминальных ветвей может быть несколько), которая определяет правило останова. Правило останова может определять две ситуации: если мы выполнили все возможные действия и не достигли нужного результата; если мы достигли нужного результата.

На втором этапе спланируйте рекурсивную ветвь (ветви), которая определяет рекурсивный вызов функции с упрощением аргументов. При каждом рекурсивном обращении мы должны приближаться к условию останова!

Формирование результата может быть осуществлено двумя способами: в процессе рекурсивных вызовов (при спуске «вниз»), или в процессе подъема «наверх». Если рекурсия вызывает у вас затруднения, то, как правило, проще формировать результат при спуске «вниз». Для этого введите вспомогательные параметры, в которых и будет формироваться результат. При достижении правила останова сформированный результат и будет результатом работы функции.

Внимание! Циклы, функционалы и функцию присвоения не использовать!

Варианты заданий.

1. а) напишите функцию, которая проверяет, являются ли все элементы списка положительными числами. Функция возвращает NIL, если хотя бы один элемент списка НЕ является числом, или если в списке есть хотя бы один отрицательный элемент.

   б) определите функцию, аргументом которой является числовой список z, и которая формирует новый список, удаляя из z на всех уровнях положительные элементы (z может быть многоуровневым).

2. а) определите функцию умножения двух целых чисел через сложение и вычитание.
   б) напишите функцию, которая увеличивает все элементы числового списка в два раза (список может быть многоуровневым).

3. а) определите функцию от двух аргументов x и n, которая создает список из n раз повторенных элементов x.
   б) напишите функцию, аналогичную встроенной функции subst замены в списке s выражения x на y, но производящую взаимную замену y на x: x -> y, y -> x.
   Например: (f 2 a `(s d 2 f a 4 2)) ==> (s d a f 2 4 a)

4. а) Определите функцию (f n) (n кратное 3), вычисляющую сумму: 1*2*3+4*5*6+…+(n-2)*(n-1)*n
   б) определите функцию, на вход которой подается список (он может быть многоуровневым). Функция должна возвращать T, если все атомы списка (на всех уровнях) являются числами, и NIL – в противном случае.

5. а) последовательность чисел Фибоначчи 1,1,2,3,5,8,13… строится по следующему закону: первые два числа – единицы; любое следующее число есть сумма двух предыдущих. Напишите функцию (f n f1 f2) с накапливающимися параметрами f1 и f2, которая вычисляет n-е число Фибоначчи.
   б) напишите функцию, которая считает полное количество атомов (на всех уровнях и не равных NIL) во входном списке.

6. а) определите функцию, аргументом которой является список с четным количеством элементов – атомов. Функция должна менять местами каждую пару элементов списка (1-й элемент со 2-м, 3-й с 4-м и т.д.). Если количество элементов списка является нечетным, функция должна возвращать NIL.
   б) Определите функцию, на вход которой подаются два списка – множества. Функция должна выдавать объединение этих множеств

7. а) напишите функцию, имеющую два аргумента: числовой список и целое число N. Функция должна возвращать список, в котором элементы, стоящие на четных позициях, увеличены в N раз.

б) Определите функцию, на вход которой подаются два списка – множества. Функция должна искать пересечение этих множеств

8. а) напишите функцию, аргументом которой является целое число, которая бы вычисляла факториал этого числа.
   б) напишите функцию, аргументом которой является список с любыми s-выражениями. Функция должна возвращать линейный список из всех атомов исходного. Например, список (q (s d) (f (x c)) b) должен преобразовываться функцией в список (q s d f x c b).

9. а) Написать функцию, на вход которой подается список из двух списков чисел. Результатом работы функции должен стать список, элементами которого являются поэлементные суммы этих двух подсписков. Например: ((1 3 2)(4 6 3)) ==> (5 9 5)
   б) Определить функцию, которая определяет, является ли данное натуральное число простым

10. а) Написать функцию, на вход которой подается список атомов. Функция должна формировать список из двух подсписков следующим образом: в 1-й подсписок включаются все элементы исходного списка, стоящие на нечетных позициях, во 2-й – на четных. Например: (ф ы в а п р) ==> ((ф в п)(ы а р))
    б) Определите функцию (f s), вычисляющую знакочередующую сумму a1-a2+a3-a4+…+ak*(-1)^(k+1) для списка s, имеющего вид (a1 a2 a3 … ak).

11. а) Определите функцию, аргументом которой является одноуровневый список, элементами которого могут быть как числа, так и символьные атомы. Функция должна формировать новый список, состоящий из двух подсписков: в 1-й подсписок включаются числа, во 2-й – все символьные атомы.
    Например: (в а 1 3 ы 5 ф ы) ==> ((1 3 5)(в а ы ф ы))
    б) Напишите функцию, аргументом которой является список, удаляющую из этого списка все не числовые атомы на всех уровнях (список может быть многоуровневым).

12. а) Определите функцию, аргументом которой является многоуровневый список, атомы которого могут быть как числа, так и символьные атомы. Функция должна менять все числа на атом NUM.

Например: (a s 4 (s 1 2) c) ==> (a s num (s num num) c)
б) напишите функцию, определяющую глубину первого вхождения элемента y в список w.

13. а) определите функцию, аргументом которой является целое число n, результатом которой является сумма цифр натурального числа n. (встроенная функция MOD дает остаток от деления).
    б) Определите функцию, аргументом которой является одноуровневый список. Функция должна менять все отрицательные числа на атом MINUS, положительные – на атом PLUS, нули – на атом ZERO и символьные атомы – на атом SMV.
    Например: (a -4 s 1 0 c) ==> (smv minus smv plus zero smv)

14. а) Напишите функцию, которая делает из списка множество, т.е. удаляет все повторяющиеся элементы.
    б) Определите функцию (f s n), где s – список, n – число. Функция должна возвращать n-й атом из списка s. Список s может быть многоуровневым.

15. а) функция (f l) имеет единственный аргумент – список, состоящий из четного количества элементов – чисел. Каждая пара чисел в списке задает стороны прямоугольника. Функция должна возвращать список из периметров прямоугольников.
    б) Напишите функцию, на вход которой подается многоуровневый список. Функция должна подсчитывать количество всех подсписков на всех уровнях.
    Например: ((2 3) 4 ((3 (4 (2))) 1)) ==> 5

16. а) Определите функцию, реверсирующую список и все его подсписки на любом уровне.
    Например, (1 (2 3) 4) ==> (4 (3 2) 1).
    б) Напишите функцию, зависящую от двух аргументов x и y, удаляющую все вхождения x в список y на всех уровнях. X может быть атомом или списком.

17. а) напишите функцию, на вход которой подается список и два числа: n < k. Функция должна формировать новый список, включая в него только те элементы исходного списка, которые

являются числами. При этом выбираемые числа должны быть или меньше n, или больше k.
б) на вход функции подается список, элементами которого являются списки из двух атомов. Функция должна менять местами элементы в каждом из подсписков: ((1 2) (ф ы) (3 в)) ==> ((2 1) (ы ф) (в 3)). Функцию reverse не использовать!

18. а) напишите функцию, которая бы подсчитывала количество пустых списков на всех уровнях в единственном аргументе функции – списке.
б) На вход функции подается числовой список с четным количеством элементов. Определить функцию, которая бы проводила расчеты по следующей схеме: (x1 x2 x3 x4 …) ==> x1*x2+x3*x4+…
Например: (1 2 3 4 5 6) ==> 44

19. а) Определить функцию, которая изменяет входной список, переставляя местами 1-й и 2-й элементы, 3-й и 4-й элементы и т.д. Например: (a s d f g h) ==> (s a f d h g)
б) определить функцию, которая находит максимальную сумму двух соседних чисел в заданном списке. Единственный аргумент функции является одноуровневым числовым списком

20. а) Определить функцию, на вход которой подается два числа K, L и числовой список, на выходе формируется список, положительные элементы которого увеличиваются в K раз, а отрицательные элементы увеличиваются в L раз. Например:
(2 3 (-1 2 3 -4 5)) ==> (-3 4 6 -12 10)
б) Написать функцию, на вход которой подается натуральное положительное число. Функция должна возвращать список его делителей.

21. а) На вход функции подается одноуровневый список чисел. Необходимо найти номер первого максимального элемента списка.
б) написать функцию, удаляющую из исходного списка все числовые атомы. Исходный список может быть многоуровневым.

22. а) На вход функции подается одноуровневый список. Сформировать новый список, удалив из исходного все элементы, стоящие на нечетных позициях.

б) написать функцию, имеющую два аргумента Y и N, где Y – список, N – атом. Функция должна удалять из Y все элементы, совпадающие с N. Исходный список может быть многоуровневым.

23. а) На вход функции подается два числовых списка $[a_1, a_2, \ldots, a_n]$ и $[b_1, b_2, \ldots, b_n]$. Функция должна формировать новый список $[\min(a_1,b_1), \min(a_2,b_2), \ldots, \min(a_n,b_n)]$.
    б) написать функцию, изменяющую исходный список следующим образом: после каждого встреченного нуля добавляется единица. Исходный список может быть многоуровневым.

2.3 Лабораторная работа «Разработка функциональных программ»

Цель работы

Целью проведения данной работы является знакомство с принципами построения сложных рекурсивных функций, лямбда-выражений, модульной структуры программы.

Рекомендации по подготовке к работе

В процессе выполнения лабораторной работы необходимо написать программу, выполняющую действия, описанные в вашем варианте задания. Количество необходимых функций, их входные и выходные параметры определить самостоятельно, исходя из задания.

При написании программы декомпозируйте задачу: модульная структура программы облегчает как написание, так и понимание рассматриваемых действий. Всегда легче писать более мелкие функции, из которых можно компоновать более сложные задачи.

Внимание! Циклы и функцию присвоения не использовать!

Варианты заданий

1. Определите функцию, зависящую от одного аргумента – многоуровневого списка, которая по данному списку формирует одноуровневый список его элементов, встречающихся в нем более одного раза.

2. Определите функцию (f a n), которая от двух числовых аргументов вычисляет величину a+a*(a+1)+a*(a+1)*(a+2)+…+a*(a+1)*(a+2)*…*(a+n).

3. Напишите функцию, на вход которой подается атом Y и список W. Функция должна заменять Y на число, равное глубине вложения Y в W, например, Y=a, W=((a b) a (c (a (a d)))) -> ((2 b) 1 (c (3 (4 d)))).

4. Напишите функцию, которая сортирует список чисел по возрастанию. Список может быть многоуровневым. Тогда вес (значение) любого подсписка для сортировки определяется суммой его элементов. Любой подсписок внутри тоже должен быть отсортирован.
   Например: (9 2 (1 2) 5 (3 6 2) 12 (6 (3 1))) =>
   (2 (1 2) 5 9 ((1 3) 6) (2 3 6) 12).

5. Написать программу , которая возвращает список (m1 m2 m3), состоящий из трех наибольших элементов исходного числового списка s: m1>=m2>=m3. Исходный список содержит не менее трех элементов.

6. Написать программу, результатом работы которой является список, получающийся из списка списков s после удаления всех подсписков, содержащих числа.

7. Определите функцию (f s), которая в многоуровневом списке s переставляет все отрицательные элементы в начало списка, положительные – в конец, и делает его одноуровневым. Например,
   (f `(4 -2 (3 (-1 -9)) 8)) -> (-2 -1 -9 4 3 8)

8. Определить функцию с одним аргументом – одноуровневым списком. Функция должна подсчитывать количество вхождений в список каждого атома и выдавать результат в виде списка списков: (a s 3 d s s a 3) ==> ((a 2) (s 3) (3 2) (d 1))

9. Написать программу, осуществляющую перевод десятичного числа в любую заданную систему счисления (двоичную, восьмеричную и т.п.)

10. Определить функцию с двумя аргументами A и B – числовыми списками. Функция должна проверять, выполняется ли следующее соотношение между элементами списков: bi=ai+1; $a_{i+1}=2*b_i$

11. Определить функцию трех аргументов: a, b и c. Первым аргументом является множество – список целых чисел, b и c – целые числа. Функция должна формировать список трех подмножеств данного множества, определяемых следующим образом:
    a. Четные числа множества;
    b. Числа множества, являющиеся квадратами числа;
    c. Все числа b<=ai<=c.

    Например: (f `(1 2 3 4) 0 2) => ((2 4) (1 4) (1 2))

12. Написать функцию, которая сортирует список чисел, используя алгоритм простой вставки.

13. На вход функции подается многоуровневый список чисел. Необходимо найти номер первого экстремального (максимального или минимального) элемента списка. Нумерация должна идти по атомам.

14. Напишите программу, формирующую список простых чисел на заданном интервале.

15. Задан числовой список. Написать программу, подсчитывающую среднее значение элементов списка, за исключением максимального и минимального элементов.

16. Написать программу, осуществляющую перевод числа, представленного в любой заданной системе счисления (двоичной, восьмеричной и т.п.), в десятичную систему счисления.

17. Написать функцию, которая сортирует список чисел методом пузырька.

18. Написать программу, определяющую дизъюнктивную сумму между двумя множествами: A+B=(A U B) – (A ∩ B)

19. Задан многоуровневый список чисел. Написать программу, возвращающую список из двух чисел – атомов исходного с максимальной разницей между ними.

20. Задан список вида ((A1 N1)(A2 N2)…(AK NK)), где Ai – символьный атом, Ni – число. Написать программу, возвращающую список из двух символьных атомов исходного списка (Ai Aj) с максимальным произведением Ni*Nj.

21. Написать функцию, упорядочивающую список, заданный в качестве ее первого аргумента, переставляя его элементы в той последовательности, в какой они встречаются в списке, являющемся значением второго аргумента.

22. Напишите функцию трех аргументов: X, Y – числовые списки, Q – число. Функция должна находить сумму S вида xi+yj, наиболее близкую к числу Q.

23. Написать функцию, имеющую два аргумента: V – список, N – число. Функция должна формировать список элементов из V, встречающихся в нем не менее N раз.

2.4 Лабораторная работа «Функционалы»

Цель работы

Целью проведения данной работы является знакомство с функционалами и принципами их использования при написании новых функций.

Рекомендации по подготовке к занятию

В процессе выполнения лабораторной работы необходимо написать функцию, описанную в вашем варианте задания, с использованием функционалов. В случаях, когда обрабатывается многоуровневый список, может потребоваться использование косвенной рекурсии.

Отображающие функционалы работают с функциями, обрабатывающими единственный аргумент – s-выражение (список или атом). В случае, если вам необходимо использовать дополнительные

параметры, опишите требуемую функцию как лямбда-выражение. Лямбда-выражение внутри функции видит все локальные переменные этой функции.

Варианты заданий

1. Напишите функцию (exist p x), которая проверяет «Существует ли элемент списка x, удовлетворяющий предикату p?» (p – функция или функциональное имя).

2. Написать функцию, аргументом которой является числовой список. Функция должна возвращать список сумм: 1-го элемента, 2-х первых элементов, 3-х первых элементов и т.д. Например: (f `(1 2 3 4)) => (1 3 6 10).

3. Определите функцию (f s), которая из списка чисел s создает новый список, меняя знак у каждого атома. Исходный список не предполагается одноуровневым.

4. Напишите функцию (sum_list s), аргументом которой является список арифметических выражений. Функция должна возвращать сумму вычисленных арифметических выражений. Например:
(sum_list `((+ 1 2) (* 3 4) (- 8 4))) => 19

5. Напишите функцию (all p x), которая проверяет «Для всех ли элементов списка x выполняется предикат p?» (p – функция или функциональное имя).

6. Напишите функцию (count p x), которая подсчитывает, сколько атомов в списке x удовлетворяет предикату p (p – функция или функциональное имя). Список x не предполагается одноуровневым.

7. Определите функцию (f s n), которая из списка чисел s создает новый список, прибавляя к каждому атому число n. Исходный список не предполагается одноуровневым.

8. Напишите функцию (filter p x), которая создает список из элементов списка x, удовлетворяющим предикату p (p – функция или функциональное имя).

9. Напишите функцию, строящую список всех подмножеств данного множества.

10. Напишите функцию (break p x), которая бы формировала список, состоящий из двух подсписков: в 1-й подсписок должны попадать все элементы x, удовлетворяющие предикату p, во 2-й – не удовлетворяющие.

11. Напишите функцию (f s n), которая из многоуровневого списка чисел s создает новый список, исключив из него все элементы, которые превышают заданное число n.

12. Напишите функцию (f-n p x n), которая выдает истину только если ровно для n элементов списка x выполняется предикат p (p – функция или функциональное имя).

13. Напишите функцию (f n l_f x), где n – число, l_f – список арифметических операций, x – числовой список. Функция должна возвращать число – результат применения n-й операции к элементам списка x.

14. Напишите функцию, на вход которой подается многоуровневый список. Функция должна преобразовывать этот список в одноуровневый.

15. Напишите функцию, на вход которой подается многоуровневый список и атом. Функция должна удалять из списка все элементы, совпадающие с указанным атомом.

16. Написать функцию, удаляющую из исходного списка все числовые атомы. Исходный список может быть многоуровневым.

17. Определить функцию, которая находит максимальную сумму двух соседних чисел в заданном списке. Единственный аргумент функции является одноуровневым числовым списком.

18. Напишите функцию, которая считает полное количество атомов (на всех уровнях и не равных NIL) во входном списке.

19. Напишите функцию (f s n), которая в многоуровневом списке чисел s суммирует только те числа, которые превышают заданное число n.

20. напишите функцию, которая бы подсчитывала количество пустых списков на всех уровнях в единственном аргументе функции – списке.

21. Напишите функцию (new_list p x), которая создает список из элементов списка x, НЕ удовлетворяющих предикату p (p – функция или функциональное имя).

22. Напишите функцию (p L P), которая в качестве результата выдает список, состоящий из всех элементов списка L, начиная с начала списка и до первого элемента, не удовлетворяющего данному предикату P. Например:
    (p `(2 4 6 a 8 9) `numberp) ==> (2 4 6)

23. Напишите функцию (p L P), которая в качестве результата выдает список, состоящий из всех элементов списка L, начиная с первого элемента, не удовлетворяющего данному предикату P. Например:
    (p `(2 4 6 a 8 9) `numberp) ==> (a 8 9)

2.5 Лабораторная работа «Графы и деревья»

Цель работы

Целью данной работы является работа с графами, представленными различными способами и написание функций для работы с ними.

Рекомендации по подготовке к работе

В процессе выполнения лабораторной работы необходимо написать программу определяющую для графа (дерева) заданную характеристику. При необходимости используйте локальные или вспомогательные функции.

Решение различных задач обработки графов напрямую зависит от способа их представления. Так, для поиска пути на графе удобнее использовать вектора смежности. Если же мы решаем задачу построения остовного дерева, граф удобнее представлять в виде списка ребер и множества узлов графа.

Представление деревьев зависит от вида дерева: обычное или бинарное, и от вида решаемой задачи.

Продумайте алгоритм, который позволит решить поставленную задачу. При необходимости вы можете воспользоваться существующими алгоритмами просмотра, изменения, корректировки графов и деревьев.

Если в вашем варианте не оговорен способ представления графа (дерева), вы можете выбрать тот способ представления, который будет боле удобен для решения поставленной задачи.

Варианты заданий

1. Дан граф. Написать функцию, которая находит в графе максимальный цикл и выдает его в виде списка вершин. Если в графе нет циклов, функция должна возвращать nil.
2. Написать функцию, на вход которой подается граф, определяющую, связан ли граф.
3. Задан граф, у которого для каждой дуги задана ее длина: ((a b 12) (s d 3) …). Написать функцию, определяющую кратчайший путь между указанными двумя вершинами.
4. Написать функцию, подсчитывающую количество циклов в графе.
5. Определить функция, на вход которой подается граф. Функция должна определять, является ли он гамильтоновым, и если да, то найти гамильтонов цикл. Цикл в графе называется гамильтоновым, если он содержит все вершины графа ровно по одному разу; граф с таким циклом называется гамильтоновым.
6. На вход функции подается ориентированный граф. Функция должна определять, существует ли путь из A в B. Если путь существует, выдать его в качестве результата работы, если пути нет, выдать пустой список.
7. Написать функцию, на вход которой подается граф в виде (a (b) b (a c d) c (b e f g) …). Функция должна преобразовывать граф в вид: ((a b) (b c) (b d) …).
8. Написать функцию, на вход которой подается граф в виде (a (b) b (c d) c (e f g) …). Функция должна проверять, является ли граф ориентированным.
9. Определите функцию, аргументом которой является дерево. Функция должна вернуть ветвь с максимальным количеством листьев.
10. Определить функцию, которая ищет заданную вершину в дереве и возвращает список, содержащий предка искомой вершины и ее потомков: (предок (потомок1 потомок2 …)).
11. Написать функцию, которая проверяет, является ли заданный граф деревом (имеет ли циклы)?

12. На вход функции подается дерево. Функция должна добавить к каждой вершине информацию: глубину ветки, исходящей из данной вершины.
    Например, дерево вида (1 (2 (3 (4))(5)) (6 (7))) должно преобразоваться в ((1 3) ((2 2) ((3 1) ((4 0)))((5 0))) ((6 1) ((7 0))))
13. Стяжение ветви. Определить функцию, аргументами которой является дерево и две его вершины. Функция должна склеивать два заданных узла, если они соседние и выдавать NIL в противном случае.
14. Напишите программу, определяющую эйлеровый путь, начинающийся с заданной вершины неориентированного графа. Путь называется эйлеровым, если проходит все ребра графа по одному разу. Если такой путь не существует – программа должна возвращать NIL.
15. Напишите программу, определяющую компоненты связности данного графа. Результатом работы программы должен быть список списков (компонента связности – список вершин).
16. Напишите программу, на вход которой подаются два дерева. Программа должна проверять, изоморфны ли данные деревья (Два дерева называются изоморфными, если можно отобразить одно из них в другое, изменением порядка ветвей в поддеревьях).
17. Напишите программу, которая будет строить список смежностей по представлению графа, заданному в виде матрицы смежностей.
18. Определить функцию, аргументом которой является дерево, подсчитывающую количество листьев в дереве.
19. Написать функцию, на вход которой подается дерево, определяющую максимальную глубину дерева.
20. Задан граф, у которого для каждой дуги задана ее длина: ((a b 12) (s d 3) …). Определить функцию, которая находит две самые удаленные друг от друга вершины.
21. Определить функцию, которая ищет середину кратчайшего пути между двумя заданными вершинами. Функция должна возвращать: атом-имя вершины (если между исходными вершинами нечетное количество вершин), список из двух вершин (если четное) и NIL, если заданные вершины – соседи.
22. Напишите программу, формирующую для ориентированного графа G его транзитивное замыкание G*. Дуга (a b) принадлежит G*, если существует ориентированный путь из a в b.
23. Определите для бинарного дерева отношение ordered(Tree), выполненное, если дерево Tree является упорядоченным деревом целых чисел, т. е. число, стоящее в любой вершине дерева, больше любого элемента в левом поддереве и меньше любого элемента в

правом поддереве. Например, дерево tree(nil, 5, tree(nil, 6, tree(tree(nil, 8, nil), 10, nil))) является упорядоченным.

2. 6 Лабораторная работа «Циклы и блочные функции»

Цель работы

Целью проведения данной работы является знакомство с циклами и блочными функциями и принципами их использования при написании новых функций.

Рекомендации по подготовке к работе

В процессе выполнения работы необходимо написать функцию с использованием циклов и блочных функций, описанную в вашем варианте задания.

Зачастую рекурсивная обработка списков требует большого объема оперативной памяти для организации стека. В таком случае использование циклов позволяет строить более эффективные программы, минимизирующие затраты ресурсов.

Выбор требуемого цикла или блочной функции осуществлять самостоятельно. В данной лабораторной работе возможно использование функции присвоения.

Задание на лабораторную работу

Напишите функцию с использованием циклов и блочных функций. При необходимости используйте локальные или вспомогательные функции.

Внимание! Рекурсию не использовать!

Использование рекурсии позволительно только при обработке многоуровневых списков.

В качестве заданий используйте свои варианты из лабораторной работы «Рекурсивные функции».

3 Методические указания для организации самостоятельной работы

3.1 Общие положения

Цель самостоятельной работы по дисциплине – проработка лекционного материала, самостоятельное изучение некоторых разделов курса, подготовка к лабораторным работам, тестам и контрольной работе.

Самостоятельная работа студента по дисциплине «Функциональное и логическое программирование» включает следующие виды его активности:

1. проработка лекционного материала;
2. изучение тем теоретической части дисциплины, вынесенных для самостоятельной проработки;
3. подготовка к лабораторным работам;
4. подготовка к тестам;
5. подготовка к контрольной работе;
6. подготовка к экзамену.

3.2 Проработка лекционного материала

При проработке лекционного материала необходимо:

а) отработать прослушанную лекцию (прочитать конспект, прочитать учебное пособие и сопоставить записи с конспектом, просмотреть слайды) и восполнить пробелы, если они имелись (например, если вы что-то не поняли или не успели записать);

б) перед каждой последующей лекцией прочитать предыдущую, дабы не тратилось много времени на восстановление контекста изучения дисциплины при продолжающейся теме.

Данный вид деятельности ориентирован как на закрепление материала, так и на подготовку к тестовым заданиям и контрольным работам.

Содержание лекций:

Концепция и особенности функционального программирования. Свойства функциональных языков. Основные особенности Лиспа, достоинства языка.

Элементарные понятия языка Лисп: атомы и списки. Программа на языке Лисп. Вычислимые выражения. Понятие функции, префиксная нотация.

Вычисление лямбда-выражений. Определение функций в Лиспе. Базовые функции языка, предикаты.

Понятие рекурсии. Правила записи рекурсивной функции. Терминальная ветвь, рекурсивная ветвь. Прямая и косвенная рекурсия. Рекурсия с несколькими терминальными ветвями, рекурсивными ветвями.

Внутреннее представление списков. Вспомогательные функции над списками.

Глобальные и локальные переменные. Изменение значений переменных. Диалоговый режим работы. Функции ввода-вывода. Циклы и блочные функции. Обработка текстовых данных. Работа с файлами. Массивы.

Разрушающие функции. Рассматриваются функции, изменяющие структуру списков. Обычные функции не изменяют структуру данных в Лиспе: при формировании новых структур выделяется новая область памяти, существующие же списки сохраняются неизменными.

Важно! При использовании разрушающих функций мы меняем значения одних переменных, и при этом могут измениться значения других переменных, связанных с теми же структурами.

Применение разрушающих функций удобно для изменения данных в Базах Знаний, когда данные должны быть связаны на глобальном, смысловом уровне.

Для лучшего понимания работы разрушающих функций необходимо разобраться с внутренним представлением списков. Обратите внимание, как изменяется внутреннее представление списков при их формировании базовыми функциями, и при их изменении с помощью разрушающих функций.

Функции высших порядков. Различие между данными и функциями. Функционалы.

Работа с графами и деревьями: представление, обработка, поиск пути на графе.

3.3 Самостоятельное изучение тем теоретической части курса

3.3.1 Циклы и блочные функции.

Несмотря на достоинства рекурсии (краткость и наглядность записи), у нее есть существенный недостаток: при работе рекурсивных функций требуется дополнительная память, объем которой пропорционален глубине рекурсии. При обработке длинных списков рекурсивными функциями может просто не хватить памяти.

В противовес рекурсии у циклов есть следующие преимущества:

1) объем памяти, используемый в цикле, без учета памяти, занимаемой значениями переменных, не зависит от числа итераций;

2) циклические алгоритмы, как правило, выполняются быстрее рекурсивных.

В Лиспе существуют следующие циклы: LOOP, DO и DOTIMES. Рассмотрите самостоятельно работу и организацию указанных функций.

Обратите внимание, что значения локальных переменных в цикле DO меняются на каждом шаге *одновременно*! Особенно это важно для тех переменных, которые меняют значение в зависимости от значений других переменных.

Цикл DOTIMES используется только в том случае, если мы знаем заранее, сколько раз должен повториться цикл.

Кроме циклов могут быть использованы и блочные функции LET и PROG. Функция LET, по сути, является аналогом лямбда-выражение с тем отличием, что значения локальным переменным присваиваются в начале выражения, а не в конце.

Обратите внимание, что в функции PROG начальные значения локальных переменных равны NIL, а значит с ними можно работать как со списками. Две дополнительные функции, используемые внутри конструкции PROG, позволяют осуществлять переходы по меткам и выходить из выражения в любом требуемом месте.

3.3.2 Обработка и хранение знаний: свойства символов, ассоциативные списки.

Для обработки и хранения знаний в Лиспе существует два типа списков: списки свойств и ассоциативные списки.

В Лиспе с символом можно связывать, не только значение, но и информацию, называемую списком свойств (property list).

Списки свойств широко используются для представления смысловой информации, когда с атомом необходимо связать список определенных свойств и связанных с ними значений. Существуют специальные функции языка, позволяющие выполнять различные действия с такими списками: присвоить, прочитать, заменить, удалить свойство. Кроме этого, всегда можно просмотреть информацию о списке свойств.

Ассоциативный список или просто а-список (a-list) есть основанная на списках и точечных парах структура данных, описывающая связи наборов данных и ключевых полей, для работы с которой существуют готовые функции.

Ассоциативные списки применяются при работе с динамическими базами данных в оперативной памяти. Ассоциативный список можно рассматривать как отображение множества ключей на множество соответствующих им объектов. Для работы с такими списками существуют функции создания ассоциативного списка и поиска элементов в ассоциативном списке. К сожалению, не существует готовой функции, позволяющей изменить значение ключа в таком списке. При необходимости функцию изменения значений в ассоциативном списке необходимо писать самостоятельно.

Рекомендуемые источники

Салмина Н. Ю., Функциональное программирование и интеллектуальные системы: учебное пособие [Электронный ресурс] / Салмина Н. Ю. — Томск: ТУСУР, 2016 . — 100 с. — Режим доступа: https://edu.tusur.ru/publications/63572).