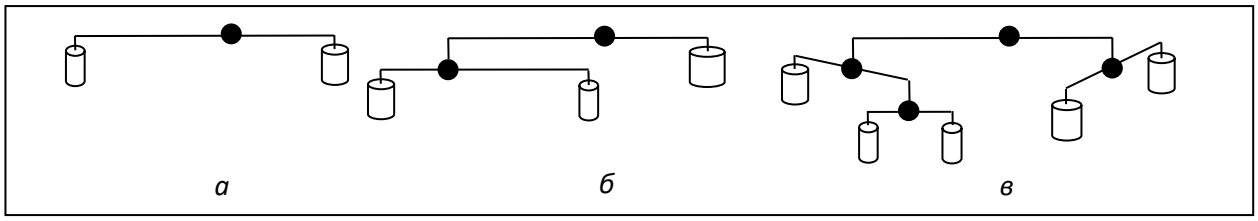
Задания (1-5) про бинарное коромысло.

Бинарное коромысло устроено так, что у него есть два *плеча*: *левое* и *правое*. Каждое плечо представляет собой (невесомый) стержень определенной *длины*, с которого свисает либо *гирька*, либо еще одно бинарное коромысло, устроенное таким же образом.

Можно (но не обязательно) представлять себе *бинарное коромысло*, чем-то похожим на конструкции, изображенные на рисунке.



В соответствии с данным выше рекурсивным определением бинарного коромысла представим бинарное коромысло (БинКор) списком из двух элементов

БинКор ::= (Плечо Плечо),

где первое плечо является левым, а второе – правым. В свою очередь Плечо будет представляться списком из двух элементов

Плечо ::= (Длина Груз),

где Длина есть натуральное число, а Груз представляется вариантами

Груз ::= Гирька | БинКор,

где в свою очередь Гирька есть натуральное число. Таким образом, БинКор есть специального вида иерархический список из натуральных чисел. Например, следующие списки представляют бинарные коромысла, изображенные выше на рисунке (вместо натуральных чисел здесь для общности и удобства восприятия представлены их обозначения с учетом места появления в списке):

а) $((l_1\ m_1)\ (l_2\ m_2))$;

б) $((l_1\ ((l_{11}\ m_{11})\ (l_{12}\ m_{12})))\ (l_2\ m_2))$;

в) $((l_1\ ((l_{11}\ m_{11})\ (l_{12}((l_{121}\ m_{121})\ (l_{122}\ m_{122})))))\ (l_2\ ((l_{21}\ m_{21})\ (l_{22}\ m_{22}))))$.

Для работы с бинарными коромыслами в таком представлении следует использовать базовые функции для работы с иерархическими списками.

Задания

1) Подсчитать общий вес заданного бинарного коромысла bk, т. е. суммарный вес его гирек. Для этого ввести рекурсивную функцию

unsigned int W (**const** БинКор bk).

2) Подсчитать число всех гирек заданного бинарного коромысла bk. Для этого ввести рекурсивную функцию

unsigned int numbers (**const** БинКор bk).

3) Подсчитать общую длину всех плеч заданного бинарного коромысла bk. Для этого ввести рекурсивную функцию

short Length (**const** БинКор bk).

4) Говорят, что бинарное коромысло *сбалансировано*, если момент вращения, действующий на его левое плечо, равен моменту вращения, действующему на правое плечо (то есть длина левого стержня, умноженная на вес груза, свисающего с него, равна соответствующему произведению для правой стороны), и если все подкоромысла, свисающие с его плеч, также сбалансированы.

Написать рекурсивную функцию или процедуру Balanced, которая проверяет заданное коромысло на сбалансированность (выдает значение true, если коромысло сбалансировано, и false в противном случае).

5) Написать рекурсивную функцию или процедуру, формирующую линейный список номеров всех вхождений одного бинарного коромысла в другое.

Примечание. Во всех заданиях для ввода и вывода списков, можно использовать функции ввода-вывода, показанные на лекции. Можно усовершенствовать их, например, функции ввода, добавив синтаксический анализ.

Задания из Учебного пособия

Решить следующие задачи с использованием базовых функций рекурсивной обработки списков:

6) проверить иерархический список на наличие в нем заданного элемента (атома) *x*;

7) удалить из иерархического списка все вхождения заданного элемента (атома) *x*;

8) заменить в иерархическом списке все вхождения заданного элемента (атома) *x* на заданный элемент (атом) *y*;

9) подсчитать число атомов в иерархическом списке; сформировать линейный список атомов, соответствующий порядку подсчёта;

10) подсчитать число различных атомов в иерархическом списке; сформировать из них линейный список;

11) сформировать линейный список атомов исходного иерархического списка таким образом, что скобочная запись полученного линейного списка будет совпадать с сокращённой скобочной записью исходного иерархического списка после устранения всех внутренних скобок;

12) проверить идентичность двух иерархических списков;

13) вычислить глубину (число уровней вложения) иерархического списка как максимальное число одновременно открытых левых скобок в сокращённой скобочной записи списка; принять, что глубина пустого списка и глубина атомарного *S*-выражения равны нулю; например, глубина списка (***a*** (***b*** ( ) ***c***) ***d***) равна двум;

14) обратить иерархический список на всех уровнях вложения; например, для исходного списка (***a*** (***b c***) ***d***) результатом обращения будет список (***d*** (***c b***) ***a***).

I. Дополнения

Дополнение 1.

15) проверить *структурную* идентичность двух иерархических списков (списки структурно идентичны, если их устройство (скобочная структура и количество элементов в соответствующих (под)списках) одинаково, при этом атомы могут отличаться);

Дополнение 2 – группа заданий 16-24.

Пусть выражение (логическое, арифметическое, алгебраическое*) представлено иерархическим списком. В выражение входят константы и переменные, которые являются атомами списка. Операции представляются в префиксной форме ((<операция> <аргументы>)), либо в постфиксной форме (<аргументы> <операция>)). Аргументов может быть 1, 2 и более. Например (в префиксной форме): (+ a (* b (- c))) или (OR a (AND b (NOT c))).

В задании даётся один из следующих вариантов требуемого действия с выражением: *проверка синтаксической корректности*, *упрощение* (преобразование), *вычисление*.

Пример *упрощения*: (+ 0 (* 1 (+ a b))) преобразуется в (+ a b).

В задаче *вычисления* на входе дополнительно задаётся список значений переменных

$$((x_1\ c_1)\ (x_2\ c_2)\ \dots\ (x_k\ c_k)),$$

где x_i – переменная, а c_i – её значение (константа).

В индивидуальном задании указывается: тип выражения (возможно дополнительно - состав операций), вариант действия и форма записи. Всего 9 заданий.

---

* - здесь примем такую терминологию: в *арифметическое* выражение входят операции +, -, *, /, а в *алгебраическое* – +, -, * и дополнительно некоторые функции.

16) логическое, проверка синтаксической корректности, добавить 4-ую операцию (которая может принимать 2 аргумента), префиксная форма

17) логическое, упрощение, префиксная форма

18) логическое, вычисление, добавить 4-ую операцию (которая может принимать 2 аргумента), префиксная форма

19) арифметическое, проверка синтаксической корректности и деления на 0 (простая), постфиксная форма

20) арифметическое, упрощение, проверка деления на 0, префиксная форма

21) арифметическое, вычисление, постфиксная форма

22) алгебраическое (+, -, *, sqrt(), log(,)), проверка синтаксической корректности, простая проверка log(,), префиксная форма

23) алгебраическое (+, -, *, sin(), cos()), упрощение, постфиксная форма

24) алгебраическое (+, -, *, power(,)), вычисление, префиксная форма

В заданиях 22 – 24 функции sqrt(), log(,), sin(), cos(), power(,) используются в классической форме (аргумент(ы) в скобках, а слева от скобок – название функции), а не в префиксной или постфиксной!

Константами могут быть положительные, отрицательные числа и 0: 5, +5, -5, +0, -0, 0.

(- -2) корректное выражение в префиксной форме

(-2 -) корректное выражение в постфиксной форме

b корректное выражение

(- b) корректное выражение в префиксной форме

(b -) корректное выражение в постфиксной форме

Некорректные выражения:

-b

+b

(- -b)

(-b -)

Константы и переменные – это атомы. Атом следует считать особым случаем выражения.

Дополнение 3 – группа заданий 25-27.

Символьное дифференцирование алгебраического выражения, рассматриваемого как функция от одной из переменных. На входе выражение в виде иерархического списка и переменная, по которой следует дифференцировать. На выходе – производная исходного выражения. После дифференцирования возможно упрощение выражения. Набор операций (функций), которые могут входить в выражение, определяется вариантом.

25) +, -, *, упростить после дифференцирования

26) +, -, *, ^, exp(), упрощать не требуется

27) +, -, *, /, sin(), cos(), упрощать не требуется

Примечание. Во всех заданиях для ввода и вывода списков, можно использовать функции ввода-вывода, показанные на лекции. Можно усовершенствовать их, например, функции ввода, добавив синтаксический анализ.

Дополнение 4 – задания про иерархическое содержание.

Будут высланы позже.