Использование динамических структур обработки данных на примере задач преобразования списков

Д.В. Здор

Аннотация*— Обработка информации является ключевым информационным процессом. В связи с этим актуализируется вопрос выбора структурной организации данных для целей организации процесса обработки. Динамические структуры данных применяются в случаях, когда при решении задачи отсутствует ясность относительно требуемого размера используемой структуры. Другим направлением в использовании динамических структур выступают задачи на обработку больших массивов данных, в том числе задачи на преобразование больших массивов данных. В такой ситуации практический смысл приобретает использование списков. Указанные обстоятельства актуализируют проблему поиска эффективных методов обработки списков. Отдельного внимания в этом аспекте заслуживают задачи на преобразование списков. Особенности использования динамических структур обработки данных на основе рекурсивных методов является одной из актуальных задач в контексте применения систем с элементами искусственного интеллекта. Цель работы заключается в анализе применения рекурсивных методов в задачах на преобразование списков. В задачах обработки списков эффективное применение находят рекурсивные методы. Использование рекурсии возможно не только в стандартных задачах обработки элементов одного списка, но и в задачах на преобразование списков, например, в задачах объединения двух списков в один, разделения одного списка на два. На примере данных задач проведен детальный анализ применения рекурсивных методов на уровне сопоставления предикатов и конкретизации переменных в программе на Прологе. В статье на примере задач объединения двух списков в один, разделения одного списка на два подробно описан способ использования рекурсивных правил для обработки списков. Описание решения задач, анализ выполнения рекурсии в рассмотренных примерах может служить технологической основой для решения других задач на преобразование списков. Полученные результаты могут быть использованы в дальнейшей разработке вопросов использования рекурсивных методов в задачах обработки списков, а также использоваться в учебном процессе при изучении теоретических основ информатики и логического программирования.

***Ключевые слова*—структуры данных, динамическая структура, обработка списков, рекурсивные методы.**

I. Введение

Обработка информации является ключевым информационным процессом в любой информационной структуре. Непосредственно обработка данных реализуется с помощью программ в соответствии с заданным алгоритмом. При этом процесс проектирования программы начинается с постановки задачи, анализа начальных условий и формализации. Этот этап предполагает построение соответствующих математических конструкций, которые составят основу алгоритма обработки данных. В связи с этим актуализируется вопрос выбора структурной организации данных, так как от этого напрямую будет зависеть применение в алгоритме соответствующих команд обработки данных. Следует отметить, что в большинстве случаев задача может иметь несколько способов решения, каждый из способов будет иметь свою специфику, использовать различные методы, поэтому выбор используемых структур данных будет, в конечном итоге, определять и эффективность построенного алгоритма.

Типовые структуры данных могут быть статическими (последовательными) и динамическими (связными). При связном распределении для хранения массива данных не требуется выделения непрерывной области памяти. Информация о порядке следования элементов задается с помощью указателя на следующий элемент [1]. Динамические структуры данных применяются в случаях, когда при решении задачи отсутствует ясность относительно требуемого размера используемой структуры. Другим направлением в использовании динамических структур выступают задачи на обработку больших массивов данных, в том числе задачи на преобразование больших массивов данных. В такой ситуации практический смысл приобретает использование списков. Таким образом, в списке элементы связаны между собой каким-либо образом логически, но главным является тот факт, что список представляет собой множество элементов, в котором в самом простом случае каждый элемент содержит ссылку на следующий элемент [2-4]. Указанные обстоятельства актуализируют проблему поиска эффективных методов обработки список в задачах формирования списка, определения длины списка, добавления элемента в конец списка, добавление элемента в заданную позицию списка, поиска элемента в списке и т.д. Отдельного внимания в этом аспекте заслуживают задачи на преобразование списков, к которым относятся объединения двух списков в один, разделения одного списка на два и т.д. [5; 6].

Активное применение систем с элементами искусственного интеллекта существенным образом преобразует рассматриваемую проблему, так как основу

любой интеллектуальной системы составляют база знаний и заложенный в систему механизм получения решения. Эти компоненты определяют две основные интеллектуальные характеристики системы: способность хранить знания и умение оперировать этими знаниями. К инструментальным средствам разработки интеллектуальных систем относят основные базовые логические языки. При этом в связи с имеющейся спецификой логических языков значимость приобретают методы использования типовых структур данных, в том числе списков. Рассмотрим особенности использование динамических структур обработки данных на примере задач преобразования списков в контексте программной реализации средствами логического языка Пролог. Различным аспектам составления программ обработки информации на языке Пролог посвящены многие работы ученых.

II. Материалы и методы

Теория исчисления высказываний и предикатов как математическая основа логического программирования рассмотрена в работах в работах Г. Метакидес [2], Л. Стерлинга [3], Н.И. Цукановой [4]. Основы логического программирования и особенности языка Пролог, такие как, основные конструкции языка, виды предложений, механизм выполнения программы, методика проектирования логических программ, рассмотрены в работах У. Клоксина [5], Б. С. Хусаинов [6]. Решение логических задач на языке программирования Пролог стало предметом изучения исследователей А.Н. Адаменко [7].

Л. Стерлинг [3] прибегает к рекурсивному программированию в контексте обработки особого типа данных – списков [3]. Списки, представление списков, реализация некоторых операций со списками с помощью рекурсивных правил также является предметом изучения в работе И. Братко [8]. Примеры программ с использованием рекурсивных правил обработки списков представлены в работе Н.И. Цукановой [4] и В.Н. Агафонова [9]. Анализ литературы показал, что вопрос решения задач обработки списков нашел достаточное освещение в научной литературе. Приведены примеры программ, в которых реализованы операции обработки списков с помощью рекурсивных правил. При этом в работах отсутствует анализ выполнения рекурсивных правил в контексте решения задачи обработки списка на уровне сопоставления предикатов и конкретизации переменных, что обнаруживает с технологической точки зрения неполноту знаний о применении рекурсивных методов в задачах обработки списков.

Полученные результаты позволили сформулировать цель, заключающуюся в анализе применения рекурсивных методов в задачах преобразования списков. Структура список содержит упорядоченный набор элементов. Элементы, как правило, связаны между собой по смыслу, например, может быть составлен список дат, список событий, список признаков и т.д. Элементами списка могут быть целые и действительные числа, символы, строки, списки. По аналогии с массивами для списков выделяется такая характеристика, как длина списка, под которой понимается количество элементов списка. Напомним при этом, что для списка отличительной особенностью является то, что длина заранее может быть неизвестна. Отметим основные моменты относительной требуемых описаний при использовании структуры список в программе на логическом языке Пролог. При использовании в программе структуры список, ее необходимо объявить в разделе используемых типов данных, а в разделе объявления предикатов нужно указать предикат, аргументом которого будет являться объявленный тип, относящийся к списку.

III. Результаты и обсуждение

В разделе фактов и правил программы задается сам список, элементы которого указываются в квадратных скобках через запятую, например:

[13, 1, 5, -3, 0, -7] – список целых чисел;

[1.3, 1.8, 5.5, -3.5, -7.7] – список действительных чисел;

[‘Пролог’, ‘Prolog’] – список строк;

[[1,2,3], [6,7,8,9], [-1,-2]] – список списков.

В Прологе, кроме уже указанного стандартного способа путем перечисления элементов, имеет место несколько иных способов представления списков.

Представление списка структурой имеет вид:

.(a,[]), что соответствует списку [a];

Структура .(a ,.(b,.(c,[]) – соответствует списку [a, b, c].

Однако принципиальное значение в контексте рассмотрения списков в соответствии с целью исследования имеет представление списка путем выделения головы и хвоста: [Head | Tail] или [H | T].

Head (Голова) – это всегда первый элемент списка.

Tail (Хвост) – это часть списка, остальные его элементы, полученные исключением головы.

Рассмотрим примеры представления списков через голову и хвост:

список [1, 2, 3] можно представить [1| [2,3]]. Для общего случая:

$$[a, b, c, d] = [a \mid [b, c, d]] = [a \mid [b \mid [c, d]]] = [a \mid [b \mid [c \mid [d]]]] = [a \mid [b \mid [c \mid [d \mid []]]]]$$

В связи с этим можно сделать вывод, что список является рекурсивной структурой, поэтому для обработки списков можно использовать рекурсивные методы обработки. Рассмотрим на примере задач преобразования списков использование рекурсивных методов обработки списков, реализация которых средствами языка логического программирования Пролог осуществляется с помощью рекурсивных правил [10]. Рассмотрим программную реализацию решения задачи объединения двух списков в один.

Постановка задачи. Объединить два непустых списка в один.

Программная реализация на языке Пролог.

domains

spisok = ***integer**** /* тип – список целых чисел */

predicates

append (spisok, spisok, spisok) /* трехместный предикат */

clauses

append ([ ], P,P). /* Объединение пустого списка со списком Р дает список Р*/

append ([X | Y], Q, [X | T]):- append(Y, Q, T).

goal

append ([9,15], [3,5], Z), ***write*** («Объединенный список =», Z).

Проведем детальный анализ применения рекурсивных методов на уровне сопоставления предикатов и конкретизации переменных.

1. Целевой предикат, append ([9,15], [3,5], Z), где Z-свободная переменная, значение которой должно определиться в ходе доказательства, сначала сопоставится с фактом, т.к. он первый в БЗ, но сопоставление терпит неуспех, так как списки не совпадают.

2.Тогда предикат append ([9,15], [3,5], Z) сопоставится с головным предикатом правила append ([X | Y], Q, [X | T]), они сопоставимы, если:

– $X^{(1)}$ конкретизируется значением [9];

– $Y^{(1)}$ конкретизируется значением [15];

– $Q^{(1)}$ конкретизируется значением [3,5];

Список [9 | $T^{(1)}$] сцепляется с Z, Z и $T^{(1)}$ – свободные. Теперь нужно доказать истинность головного предиката append ([9| 15], [3,5], [9 | $T^{(1)}$]).

3. Чтобы доказать истинность головного предиката append ([9| 15], [3,5], [9 | $T^{(1)}$]) (первый виток рекурсии), нужно доказать истинность предиката в его теле append ([15], [3,5], [$T^{(1)}$]).

Сопоставление с фактом терпит неуспех.

Предикат сопоставим с головой правила, если:

– $X^{(2)}$ конкретизируется значением [15];

– $Y^{(2)}$ конкретизируется значением [ ];

– $Q^{(2)}$ конкретизируется значением [3,5];

Список [15| $T^{(2)}$] сцепляется с $T^{(1}$, $T^{(1)}$ и $T^{(2)}$ – свободны. Теперь нужно доказать истинность головного предиката append ([15| [ ]], [3,5], [15 | $T^{(2)}$]).

4. Чтобы доказать истинность головного предиката append ([15| [ ]], [3,5], [15 | $T^{(2)}$]) (второй виток рекурсии), нужно доказать истинность предиката в его теле append ([ ], [3,5], [$T^{(2)}$]).

Он уже сопоставим с фактом, если:

– Р конкретизируется значением [3,5];

– $T^{(2)}$ конкретизируется значением Р =[3,5].

Значит, предикат тела append ([ ], [3,5], [3,5]) – истинен. Тогда начинается обратный ход рекурсии.

5. Головной предиката второго витка рекурсии append ([15| [ ]], [3,5], [15 | [3,5]]) – истинен. Предикат тела первого витка рекурсии append ([15], [3,5], [15| [3,5]) – истинен. Головной предиката первого витка рекурсии append ([9 |15], [3,5], [9| [15| [3,5]]]) – истинен. Целевой предикат append ([9 |15], [3,5], [9| [15| [3,5]]]) – истинен.

На экране появится решение: Объединенный список = [9, 15, 3, 5].

Теперь рассмотрим программную реализацию решения задачи разделения списка на два.

Постановка задачи. Разделить непустой список на два, взяв в первый список элементы меньшие или равные заданному числу М, а во второй – большие М.

Программная реализация на языке Пролог.

domains

spisok = ***integer**** /* тип – список целых чисел */

mid = ***integer***

predicates

delet (mid, spisok, spisok, spisok) /* четырехместный предикат */

clauses

delet (_, [ ], [ ], [ ]).

delet (M, [X | Y], [X | Z], P):- X <=M, delet (M, Y, Z, P).

delet (M, [X | Y], Z, [X | P]):- X > M, delet (M, Y, Z, P).

goal

delet (20, [8, 55, 15, 25], S1, S2), ***write*** («1 список =», S1, «2 список =», S2).

В этой программе процедура delet содержит два рекурсивных правила и факт, ограничивающий рекурсию. Рассмотрим, как Пролог доказывает цель. Проведем детальный анализ применения рекурсивных методов на уровне сопоставления предикатов и конкретизации переменных.

1. Целевой предикат delet (20, [8, 55, 15, 25], S1, S2), где - S1 и S2 свободные переменные, значение которых должно определиться в ходе доказательства, сначала сопоставится с фактом, т.к. он первый в базе знаний, но сопоставление терпит неуспех.

2. Тогда предикат delet (20, [8, 55, 15, 25], S1, S2) сопоставится с головным предикатом первого правила delet (M, [X | Y], [X | Z], P), они сопоставимы, если:

– $M^{(1)}$ конкретизируется значением 20;

– $X^{(1)}$ конкретизируется значением [8];

– $Y^{(1)}$ конкретизируется значением [55,15, 25];

Список [8, $Z^{(1)}$] сцепляется со списком S1; Список $P^{(1)}$ сцепляется со списком S2. Теперь нужно доказать истинность головного предиката delet (20, [8 | [55,15,25]], [8 | $Z^{(1)}$], $P^{(1)}$).

3. Чтобы доказать истинность головного предиката delet (20, [8 | [55,15,25]], [8 | $Z^{(1)}$], $P^{(1)}$) (первый виток рекурсии), нужно доказать истинность предикатов в его теле.

X <= M ↔ 8 <= 20 – истинное утверждение.

Предикат тела delet (20, [55,15,25], $Z^{(1)}$, $P^{(1)}$) с фактом не сопоставим, но с головой первого правила сопоставим, если:

– $M^{(2)}$ конкретизируется значением 20;

– $X^{(2)}$ конкретизируется значением [55];

– $Y^{(2)}$ конкретизируется значением [15, 25];

– Список [55,$Z^{(2)}$] сцепляется со списком $Z^{(1)}$;

– Список $P^{(2)}$ сцепляется со списком $P^{(1)}$.

Теперь нужно доказать истинность головного предиката первого правила delet (20, [55 | [15,25]], [55 | $Z^{(2)}$], $P^{(2)}$) [11-13].

4. Чтобы доказать истинность головного предиката delet (20, [55 | [15,25]], $Z^{(2)}$, [55 | $P^{(2)}$]) (второй виток рекурсии), нужно доказать истинность предикатов в его теле.

X > M ↔ 55 > 20 – истинное утверждение.

Предикат тела delet (20, [15,25], $Z^{(2)}$, $P^{(2)}$) с фактом не сопоставим, но с головой первого правила сопоставим, если:

– $M^{(3)}$ конкретизируется значением 20;

– $X^{(3)}$ конкретизируется значением [15];

– $Y^{(3)}$ конкретизируется значением [25];

– Список [15 | $Z^{(3)}$] сцепляется со списком $Z^{(2)}$;

– Список $P^{(3)}$ сцепляется со списком $P^{(2)}$.

Теперь нужно доказать истинность головного предиката первого правила delet (20, [15 | [25]], [15 | $Z^{(3)}$], $P^{(3)}$).

5. Чтобы доказать истинность головного предиката delet (20, [15 | [25]], [15 | $P^{(3)}$], $P^{(3)}$) (третий виток рекурсии), нужно доказать истинность предикатов в его теле.

X <= M ↔ 15< 20 – истинное утверждение.

Предикат тела delet (20, [25], $Z^{(3)}$, $P^{(3)}$) с фактом не сопоставим, но с головой первого правила сопоставим, если:

– $M^{(4)}$ конкретизируется значением 20;

– $X^{(4)}$ конкретизируется значением [25];

– $Y^{(4)}$ конкретизируется значением [ ];

– Список [25 | $Z^{(4)}$] сцепляется со списком $Z^{(3)}$;

– Список $P^{(4)}$ сцепляется со списком $P^{(3)}$.

Теперь нужно доказать истинность головного предиката первого правила delet (20, [25 | [ ]], [25 | $Z^{(4)}$], $P^{(4)}$).

6. Чтобы доказать истинность головного предиката delet (20, [25 | [ ]], [25 | $Z^{(4)}$], $P^{(4)}$) (четвертый виток рекурсии), нужно доказать истинность предикатов в его теле.

X <= M↔ 25 <= 20 – ложное утверждение, поэтому осуществляется возврат и сопоставление предиката тела delet (20, [25], $Z^{(3)}$, $P^{(3)}$) из третьего витка рекурсии с головой второго правила [14].

Эти предикаты сопоставимы, если:

– $M^{(4)}$ конкретизируется значением 20;

– $X^{(4)}$ конкретизируется значением [25];

– $Y^{(4)}$ конкретизируется значением [ ];

– Список $Z^{(4)}$ сцепляется со списком $Z^{(3)}$;

– Список [25,$P^{(4)}$] сцепляется со списком $P^{(3)}$.

Теперь нужно доказать истинность головного предиката из второго правила delet (20, [25 | [ ]], $Z^{(4)}$, [25 | $P^{(4)}$]).

7. Чтобы доказать истинность головного предиката delet (20, [25 | [ ]], $Z^{(4)}$, [25 | $P^{(4)}$]) (4-ый виток рекурсии), нужно доказать истинность предикатов в его теле.

X > M ↔ 25 > 20 – истинное утверждение.

Предикат тела delet (20, [ ], $Z^{(4)}$, $P^{(4)}$) уже сопоставим с фактом, если:

– $Z^{(4)}$ конкретизируется значением пустого списка [ ];

– $P^{(4)}$ конкретизируется значением пустого списка [ ].

Значит, предикат тела является истинным, а в этом случае начнется обратный ход рекурсии.

8. Головной предикат delet (20, [25 | [ ]], [], [25, []] – истинен. Предикат тела третьего витка рекурсии delet (20, [25], $Z^{(3)}$, $P^{(3)}$) = delet (20, [25], []], [25 | []]) – истинен. Головной предикат третьего витка рекурсии delet (20, [15 | [25]], [15 | $P^{(3)}$], $P^{(3)}$)=delet (20, [15 | [25]], [15 | []], []).Предикат тела второго витка рекурсии delet(20,[15,25],$Z^{(2)}$,$P^{(2)}$) = delet(20,[15,25], [15 | []], [25 | []]) – истинен. Головной предиката второго витка рекурсииappend (20, [55 |15,25], $Z^{(2)}$, [55 | $P^{(2)}$]) = append (20, [55 |15,25], [15 | []], [55| [25 | []]]) – истинен. Предикат тела первого витка рекурсии delet (20, [55,15,25], $Z^{(1)}$, $P^{(1)}$) = delet (20, [55,15,25], [15 | []], [55, [25 | []]] – истинен. Головной предиката первого витка рекурсии append (20, [8 | [55,15,25]], [8| $Z^{(1)}$, $P^{(1)}$) = append (20, [8 | [55,15,25]], [8 | [15 | []], [55, [25 | []]]] -истинен. Целевой предикат delet (20, [8, 55, 15, 25], S1, S2) = append (20, [8 | [55,15 ,25]], [8 | [15 | []], [55, [25 | []]]] -истинен.

На экране появится решение:

1 список = [8,15]

2 список = [25,55].

Заметим, что в рассматриваемых задачах завершение рекурсивного вызова предикатов происходит при сопоставлении с фактом, содержащемся в качестве первого утверждения в базе знаний [15].

IV. Заключение

Итак, для обработки списков в задачах эффективное применение находят рекурсивные методы. Использование рекурсии возможно не только в стандартных задачах обработки элементов одного списка, но и в задачах на преобразование списков, что было показано в рассмотренных примерах объединения двух списков в один, разделения одного списка на два. На примере данных задач проведен детальный анализ применения рекурсивных методов на уровне сопоставления предикатов и конкретизации переменных. Примеры рассмотрены достаточно подробно, представлены необходимые теоретические пояснения.

Описание решения задач, анализ выполнения рекурсии в рассмотренных примерах может служить технологической основой для решения других задач на преобразование списков. Полученные результаты могут быть использованы в дальнейшей разработке вопросов использования рекурсивных методов в задачах обработки списков, а также использоваться в учебном процессе при изучении теоретических основ информатики и логического программирования.

Библиография

[1] Павлов Л.А. Структуры и алгоритмы обработки данных. – Санкт-Петербург: Лань, 2020.

[2] Метакидес Г. Принципы логики и логического программирования. – Москва: Факториал, 1998.

[3] Стерлинг Л. Искусство программирования на языке пролог. – Москва: Мир, 1990.

[4] Цуканова Н.И. Теория и практика логического программирования на языке Визуал пролог 7. – Москва: Горячая линия – Телеком, 2013.

[5] Клоксин У. Программирование на языке Пролог. – Москва: Мир, 1987.

[6] Хусаинов Б. С. Структуры и алгоритмы обработки данных. Примеры на языке Си. – Москва: Финансы и статистика, 2004.

[7] Адаменко А.Н. Логическое программирование и Визуальный пролог. – Санкт-Петербург: БХВ-Петербург, 2003.

[8] Братко И. Алгоритмы искусственного интеллекта на языке Пролог. – Москва: Вильямс, 2004.

[9] Агафонова В.Н. Логическое программирование. – Москва: Мир, 1988.

[10] Марков В.Н. Современное логическое программирование на языке Визуал Пролог 7/5. – Санкт-Петербург: БХВ-Петербург, 2016.

[11] Тарушкин В.Т., Тарушкин П.В., Тарушкина Л.Т., Юрков А.В. Логика предикатов и язык Пролог // Современные наукоемкие технологии. – 2010. – № 4. – С. 62-63.
[12] Коста Э. 2010. Визуальный пролог 7.3 для Tyros [Электронный ресурс] – Режим доступа: http://visual-prolog.com/download/73/books/tyros/tyros73.pdf (дата обращения 24.03.2021).
[13] Вирт Н. Алгоритмы и структуры данных. – Москва: ДМК Пресс, 2010.
[14] Ахо А. Структуры данных и алгоритмы. – Москва: Вильямс, 2016.
[15] Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. Алгоритмы: построение и анализ. – Москва: Вильямс, 2014.

Дмитрий Валерьевич ЗДОР,
кандидат педагогических наук, доцент Инженерно-технологического института Приморской государственной сельскохозяйственной академии, 692510, пр. Блюхера, 44, Уссурийск, Российская Федерация.
E-mail: zdor7440-1@unesp.co.uk.

The use of dynamic data processing structures on the example of list transformation tasks

Dmitry Zdor

***Abstract*— Information processing is a key operation with information. In this regard, the study mainstreams the issue of choosing the structural organisation of data for the purposes of organising the processing operation. Dynamic data structures are used in cases where there is no clarity regarding the required size of the structure used when solving a problem. Another area of applying the dynamic structures is the task of processing large amounts of data, including the task of converting large amounts of data. In such a situation, the use of lists becomes practical. These circumstances actualise the problem of finding effective methods for processing lists. Tasks for transforming lists deserve special attention in this aspect. Features of the use of dynamic data processing structures based on recursive methods is one of the urgent issues in the context of the use of systems with elements of artificial intelligence. The purpose of the study is to analyse the use of recursive methods in tasks for transforming lists. Recursive methods are effectively used in list processing tasks. The use of recursion is possible not only in standard tasks for processing elements of one list, but also in tasks for transforming multiple lists, for example, in the tasks of combining two lists into one, dividing one list into two. Using these tasks as an example, the authors of the study perform a detailed analysis of the use of recursive methods at the level of predicate comparison and concretisation of variables in a Prolog programme. Using the example of the tasks of combining two lists into one, dividing one list into two, the study thoroughly describes the method of using recursive rules for processing lists. The description of the solution of problems, the analysis of the execution of recursion in the considered examples can serve as a technological basis for solving other problems for the transformation of lists. The results obtained can be used in the further investigation of issues of applying recursive methods in problems of processing lists, as well as in the educational process upon studying the theoretical foundations of computer science and logical programming.**

***Keywords*—data structures, dynamic structure, list processing, recursive methods.**

REFERENCES

[1] Pavlov, L.A. 2020. Structures and algorithms for data processing. St. Petersburg: Lan.

[2] Metakides, G. 1998. Principles of logic and logical programming. Moscow: Factorial.

[3] Sterling, L. 1990. The art of programming in the prologue language. Moscow: Mir.

[4] Tsukanova, N.I. 2013. Theory and practice of logical programming in the Visual Prolog 7 language. Moscow: Hotline-Telecom.

[5] Kloksin, W. 1987. Programming in the Prolog language. Moscow: Mir.

[6] Khusainov, B.S. 2004. Structures and algorithms for data processing. Examples in C language. Moscow: Finance and statistics.

[7] Adamenko, A. N. 2003. Logical programming and Visual Prolog. St. Petersburg: BHV-Petersburg.

[8] Bratko, I. 2004. Algorithms of artificial intelligence in the Prolog language. Moscow: Williams.

[9] Agafonova, V.N. 1988. Logic programming. Moscow: Mir.

[10] Markov, V.N. 2016. Modern logic programming in the Visual Prolog 7/5 language. St. Petersburg: BHV-Petersburg.

[11] Tarushkin, V.T., Tarushkin, P.V., Tarushkina, L.T., Yurkov, A.V. 2010. Predicate Logic and the Prologue Language. Modern Science-Intensive Technologies, 4, 62-63.

[12] Costa E. 2010. Visual Prolog 7.3 for Tyros. http://visual-prolog.com/download/73/books/tyros/tyros73.pdf

[13] Virt, N. 2010. Algorithms and data structures. Moscow: DMK Press.

[14] Aho, A. 2016. Data structures and algorithms. Moscow: Williams.

[15] Cormen, T., Leiserson, C., Rivest, R., Stein, K. 2014. Algorithms: construction and analysis. Moscow: Williams.

Dmitry ZDOR,
Candidate of Pedagogical Sciences, Associate Professor at the Institute of Engineering and Technology of the Primorskaya State Academy of Agriculture, 692510, 44 Bluhera st., Ussuriisk, Russian Federation.
E-mail: zdor7440-1@unesp.co.uk.