**УДК 519.876.5:004.81:378.147**

**О. В. Бісікало, к.т.н., доц.**

**О. В. Бисикало, к.т.н., доц.**

**O. V. Bisikalo, Ph. D., Associate Professor**

# ВИЗНАЧЕННЯ РЕЗУЛЬТАТІВ ТЕСТОВОГО КОНТРОЛЮ ЗАСОБАМИ НЕЛІНІЙНИХ СПИСКОВИХ СТРУКТУР

# ОПРЕДЕЛЕНИЕ РЕЗУЛЬТАТОВ ТЕСТОВОГО КОНТРОЛЯ НА ОСНОВЕ НЕЛИНЕЙНЫХ СПИСОЧНЫХ СТРУКТУР

# THE DETERMINATION OF TEST CONTROL RESULTS BY MEANS OF THE NON-LINEAR LIST STRUCTURES

Анотація

В статті розглядається актуальна проблема розробки адаптивних електронних підручників з можливостями визначення індивідуальної стратегії навчання. З цією метою запропоновано формальний підхід для дослідження результатів тестового контролю на основі використання нелінійних спискових структур. Досягнутий аналітичний розв’язок задачі реалізовано на практиці за допомогою мови програмування Common Lisp.

Аннотация

В статье рассматривается актуальная проблема разработки адаптивных електронных учебников с возможностями определения индивидуальной стратегии обучения. С этой целью предложен формальный подход для исследования результатов тестового контроля на основе использования нелинейных списочных структур. Достигнутое аналитическое решение задачи практически реализовано на языке программирования Common Lisp.

Annotation

Development of the methodology of designing of electronic textbooks with an opportunity of individual learning strategy’s identification is considered in the article. In this regard a formal approach aimed at research of testing control results is suggested on the basis of the non-linear list structures. Analytical decision of the objective has been elaborated and realised in practice in Common Lisp language.

# Вступ

Важливою задачею тестового контролю знань та вмінь студентів є забезпечення можливості поєднання результатів тесту у вигляді підсумкової (синтетичної) оцінки та аналітичного представлення складових цієї оцінки з метою побудови подальшої стратегії навчання [1]. Така постановка задачі фактично дозволяє закласти динамічну модель студента в базу знань тестового комплексу, яка, в свою чергу, може вважатися моделлю предметної області. Ефективність використання в навчальному процесі електронних підручників (ЕП), які, власне, об'єднують моделі предметної області та студента, в першу чергу залежить від ступені адаптивності навчального матеріалу до індивідуальних особливостей студента. Певного рівня адаптивності ЕП можна досягти на основі використання моделей та алгоритмів, що базуються на теорії кінцевих автоматів і Марківских ланцюгів. В роботі [2] для побудови діючої інформаційно-навчальної системи "Гефест" було застосовано модель управління адаптивним навчанням на основі теорії кінцевих автоматів Мура. Цим самим в основу системи було закладено сценарій "доки не навчу, то не відпущу", а сутність адаптації полягає в підготовці додаткових роз'яснювальних матеріалів для студентів у випадку недостатнього засвоєння ними основного навчального матеріалу.

Необхідно зауважити, що в реальній практиці навчального процесу, особливо дистанційного, досить часто виникають ситуації, коли студент має певні знання та навички з дисципліни, що викладається. Реальне зменшення часу на засвоєння навчального матеріалу в цілому та підвищення мотивації до навчання з ЕП для таких студентів можна досягти, якщо провести попереднє загальне тестування та, на його основі, визначити індивідуальну програму навчання [3]. Математична модель з [2] цього не дозволяє, хоча застосована в системі "Гефест" додаткова об'єктна модель трьохрівневої побудови навчального матеріалу частково закладає основу для розв'язання проблеми.

Таким чином, постановкою задачі будемо вважати визначення аналітичної оцінки тестового контролю студента з метою побудови індивідуальної стратегії навчання.

## Вступление

Важной задачей тестового контроля знаний и умений студентов есть обеспечение возможности объединение результатов теста в виде итоговой (синтетической) оценки и аналитического представления составных этой оценки с целью построения дальнейшей стратегии обучение [1]. Такая постановка задачи фактически позволяет заложить динамическую модель студента в базу знаний тестового комплекса, которая, в свою очередь, может считаться моделью предметной области. Эффективность использования в учебном процессе электронных учебников (ЭУ), которые, собственно, объединяют модели предметной области и студента, в первую очередь зависит от степени адаптивности учебного материала к индивидуальным особенностям студента. Определенный уровень адаптивности ЭУ можно достичь на основе использования моделей и алгоритмов, которые базируются на теории конечных автоматов и Марковских цепей. В работе [2] для построения действующей информационно-учебной системы “Гефест” была применена модель управление адаптивным обучением на основе теории конечных автоматов Мура. Этим самым в основу системы был заложен сценарий “пока не научу, то не отпущу”, а сущность адаптации состоит в подготовке дополнительных разъяснительных материалов для студентов в случае недостаточного усвоения ними основного учебного материала. Необходимо заметить, что в реальной практике учебного процесса, в особенности дистанционного, довольно часто возникают ситуации, когда студент имеет определенные знания и навыки по материалу преподаваемой дисциплины. Реальное уменьшение времени на усвоение учебного материала в целом и повышение мотивации к обучению с ЭУ для таких студентов можно достичь, если провести предшествующее общее тестирование и, на его основе, определить индивидуальную программу обучения [3].

Математическая модель с [2] этого не разрешает, хотя примененная в системе “Гефест” дополнительная объектная модель трехуровневого построения учебного материала частично закладывает основу для решения проблемы.

Таким образом, постановкой задачи будем считать определение аналитической оценки тестового контроля студента с целью построения индивидуальной стратегии обучения.

## Introduction

Providing the possibility of test results’ combination as a concluding (synthetic) evaluation and analytical introduction of evaluation constituents aimed at the development of prospective educational strategy is the main objective of a test control of students’ knowledge and skills [1]. The given task actually permits developing a dynamic model within the knowledge base of a test complex. The knowledge base can be regarded as a model of the subject matter area. The efficacy of utilizing electronic textbooks (ET) in the educational process primarily depends on the extent of educational material adaptation towards individual potential of a student. One can achieve a certain extent of electronic textbook adaptation through models and algorithms grounded on the theory of final automatons and the Markov chains. A model for providing management over adaptive training based on the theory of the Moore final automatons was used in the work [2] for the development of “Gefest” information-training system. A scenario “I won’t let you go until teach you” is the core of the system. The essence of adaptation can be defined as the development of additional explanatory materials for students in case of shortcomings in basic knowledge and skills learned.

It should be mentioned that in the practice of the educational process, especially distance education, one could be faced with a problem when a student has already gained some skills and knowledge in the subject matter being taught. Time reduction on learning and motivation increase to work with electronic textbooks can be achieved for such students providing that a preliminary general test is conducted and individual training program is developed according to its results [3].

The mathematic model from [2] does not allow to do it though additional object model of the three-level development of the educational material, used in Gefest system, can partially facilitate to solve the problem.

The identification of analytical evaluation of a student test control aimed at the development of individual learning strategy is supposed to be our major objective.

Побудова алгоритму аналітичної оцінки

В загальному вигляді формально представимо ЕП у вигляді двійки:

$$Et = \langle Sa; Sd \rangle \qquad (1)$$

де Et – модель електронного підручника (модель электронного учебника, electronic textbook model);

Sa – модель предметної області (модель предметной области, subject matter area model);

Sd – модель студента (модель студента, student model).

Будемо вважати побудову ЕП такою, що аналогічна за своєю суттю прикладу змістовно-цільової структури навчального матеріалу [3]. Вважається, що у типовому ЕП можна розрізнити модулі (розділи), що складаються з тем, які, в свою чергу, представляють собою послідовність елементарних навчальних доз. Представимо приклад ієрархічної структури ЕП як модель предметної області у вигляді нелінійного списку:

$$Sa = (1(11(111(1111)(1112))(112(1121)(1122)))(12(121(1211)(1212)(1213))(122(1221))(123(1231)(1232))(13(131(1311)(1312))(132(1321)(1322))) \qquad (2)$$

Більш наглядним список з виразу (2) може бути зображеним у вигляді так званого структурованого друку мови програмування Лісп [4] (Додаток 2).

Для формалізованого опису моделі студента Sd, що має будуватися на основі тестового контролю змістовно-цільової структури навчального матеріалу Sa введемо алгебраїчну систему:

$$Sd = \langle Basis; Main\_Sign \rangle, \qquad (3)$$

де

$$Basis = \{ListStructures\} \qquad (4)$$

та

$$Main\_Sign = \{IF, OP\}. \qquad (5)$$

Якщо основами Basis алгебраїчної системи можна вважати нелінійні спискові структури, то сигнатуру Main_Sign як сукупність предикатів та операцій будемо розглядати на двох рівнях абстрагування – високому (макро-рівень) та низькому (мікро-рівень). Розглянемо спочатку макро-рівень, формально для множини предикатів, визначених на сукупності даних, що обробляються

$$IF = \{BalanceBracket, RightBracket, LeftAtomBracket\} \quad (6)$$

та для множини операторів, що реалізують обробку даних

$$OP = \{CreateTop, AddMeaning, EraseTop, Shift(QuantitySymbols)\}, \quad (7)$$

де предикати:

BalanceBracket – предикат істинний, якщо кількість відкритих (лівих) скобок дорівнює кількості закритих (правих) скобок у списку, що аналізується;

RightBracket – предикат істинний, якщо у списку, що аналізується наступним символом є права скобка виду “)”;

LeftAtomRight – предикат істинний, якщо у списку, що аналізується наступними символами є ліва скобка – атом – права скобка виду “(“ Atom “)”;

та оператори:

CreateTop – створити вершину стека;

AddMeaning – додати значення у вершину стека;

EraseTop – ліквідувати вершину стека;

Shift(QuantitySymbols) – зсув показника вправо по списку, що аналізується, на кількість символів QuantitySymbols.

Розглянемо тепер двохосновну алгебру

$$Alg ebra = \langle \{IF, OP\}; Sign \rangle, \quad (8)$$

в якій на основах IF і OP визначена сигнатура операцій Sign та в яку включено логічні операції: диз’юнкція, кон’юнкція та заперечення, визначені на основі IF, а також операції композиції, альтернативи та циклу, що приймають свої значення на множині OP. Формально

$$Sign = \langle \{Or, And, Not\}; \{*, ([u]A, B), \{[u]A\}\} \rangle, \quad (9)$$

де { Or, And, Not } – основні логічні операції (основные логические операции, main Boolean operations);

* – композиція A * B представляє собою послідовне виконання операторів: спочатку A, після – B (композиция A * B представляет собою последовательное выполнение операторов: сначала A, после – B, composition A * B is a consecutive implementation of operators: firstly - A, then - B);

([u] A, B) – альтернатива: у випадку якщо [u], то виконання оператора A, інакше – B (альтернатива: в случае если [u], то выполнение оператора A, иначе – B; alternative: in case of [u] operator A implements, in other case – operator B does);

{[u] A} – цикл: ітеративна операція виконання оператора A при u=0 до тих пір, доки предикат [u] не стане істинним (цикл: итеративная операция выполнения оператора A при u=0 до тех пор, пока предикат [u] не станет истинным; cycle: operator A will conduct iterative operation under u=0 until [u] becomes a true predicate).

Треба також зазначити, що для визначеної сигнатури Sign оператори

$$A, B \in OP, \qquad (10)$$

а предикат

$$u \in IF. \qquad (11)$$

Сигнатура Sign включає в себе класичні програмістські структури, закладені Дейкстрою в основу технології структурного програмування.

Ідея алгоритму: список аналізується за допомогою оператора Shift() зліва направо від початку до кінця, тобто до того моменту, коли кількість відкритих (лівих) скобок дорівнює кількості закритих (правих) скобок. Робота алгоритму основана на використанні двоелементного n-адресного стеку. Перший елемент стеку використовується для зберігання атомів, а другий – для рекурсивного визначення їх оцінок на основі вкладених підсписків та/або підлеглих атомів. В процесі рекурсивного аналізу в стек додаються підлеглі підсписки для того, щоб оцінити їх на основі відповідних атомів та/або вкладених підсписків; рекурсивна оцінка завершується з

відповідним вилученням елементу стека тоді, коли підлеглими на даному вузлі залишаються лише атоми.

Рекурсивний алгоритм оцінки дерева змістовного матеріалу, що має сім термінальних вершин, у формульному вигляді можна представити наступною структурною схемою:

$$EVAL\_LIST ::= \{[BalanceBracket]([RightBracket]EraseTop, ([LeftAtomRight]AddMeaning, CreateTop))^* Shift(QuantitySymbols)\}. \quad (12)$$

Якщо вважати системою утворюючих алгебри

$$S = Su \cup So, \quad (13)$$

де

$$Su \subset IF, So \subset OP, \quad (14)$$

тоді алгоритм EVAL_LIST по суті представляє собою складений оператор у вигляді терма – суперпозиції елементів з S та операцій з Sign.

В графічному вигляді рекурсивний алгоритм можна зобразити наступним деревом операцій (рис.1):

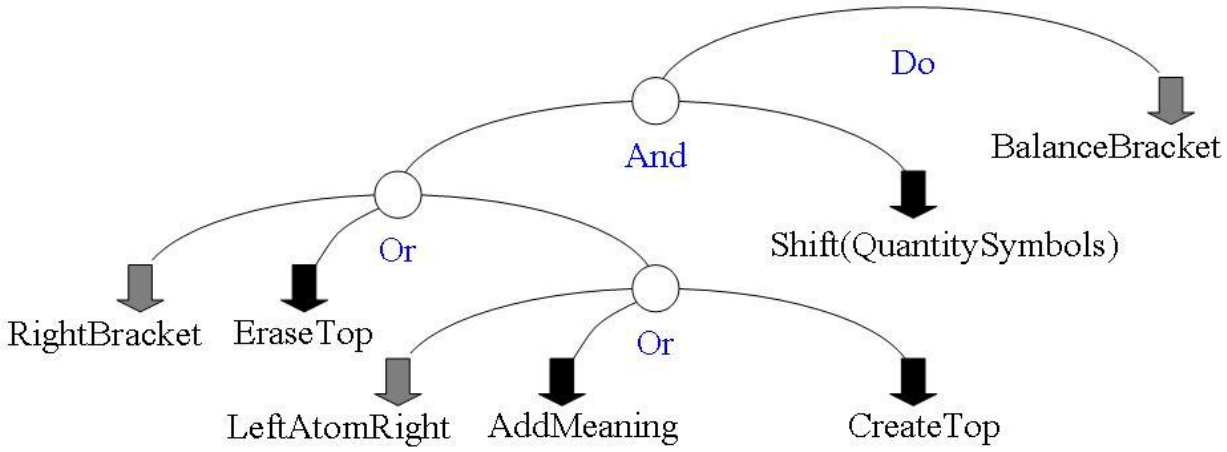


Рис.1.

Макро-оператори (7), в свою чергу, можна представити на мікро-рівні елементарними операторами обробки нелінійних спискових структур:

$$CreateTop ::= \{Push^* Ev(Atom) \to LS^* Atom \to ALS^* QuantitySymbols() = Atom + 1^* BalanceBracket() = BalanceBracket() + 1\}, \quad (15)$$

де Push – стиск стека донизу (сдвиг стека вниз, moving a stack down);

Ev(Atom) – визначення оцінки атома (навчальної дози) Atom згідно з результатами тестового контролю (определение оценки атома (учебной дозы)

Atom в соответствии с результатами тестирования, atom evaluation identification (educational dose) according to test control results);

→ – оператор передачі значення при роботі з полями стеку (оператор передачи значения при работе с полями стека, operator for meaning transfer while working with stack fields);

LS – поле зі значенням вершини стеку (поле значения вершины стека, field with a stack top meaning);

ALS – поле з адресою вершини стеку (поле адреса вершины стека, field with a stack top address);

QuantitySymbols () – функція для визначення кількості елементів списку при зсуві (функция для определения количества элементов списка при сдвиге, function for identifying a number of symbols while shifting);

BalanceBracket () – функція для визначення балансу скобок (функция для определения баланса скобок, function for identifying balance brackets);

$$AddMeaning ::= \{Ev(Atom) : LS = LS + Ev(Atom)^* \\ QuantitySymbols() = Atom + 2\}; \qquad (16)$$

$$EraseTop ::= \{ALS \rightarrow Atom^* Ev(Atom) : LS = LS + \\ EV(Atom)^* PLS = PLS + LS^* Pop^* QuantitySymbols(), \qquad (17) \\ = 1^* BalanceBracket() = BalanceBracket() - 1\}$$

де Pop – витиск стека уверх (сдвиг стека вверх, moving a stack up).

Построение алгоритма аналитической оценки

В общем виде формально представим ЕП в виде двойки (1). Будем считать построение ЭУ аналогичным по своей сути примеру содержательно-целевой структуры учебного материала из [3]. Считается, что в типичном ЭУ можно различать модули (разделы), которые состоят из тем, и которые, в свою очередь, представляют собою последовательность элементарных учебных доз. Представим пример иерархической структуры ЕП как модель предметной области в виде нелинейного списка (2). Более наглядно список из (2) может быть представлен в виде так называемой структурированной печати языка программирования Лисп [4] (Приложение 2).

Для формального описания модели студента Sd, которая может строиться на основе тестового контроля содержательно-целевой структуры учебного материала Sa введем алгебраическую систему (3), (4), (5).

Если основами Basis алгебраической системы можно считать нелинейные списочные структуры, то сигнатуру Main_Sign как совокупность предикатов и операций будем рассматривать на двух уровнях абстрагирования – верхнем (макро-уровень) и нижнем (микро-уровень). Рассмотрим сначала макро-уровень, формально для множества предикатов, определенных на совокупности данных, которые обрабатываются (6) и для множества операторов, которые реализуют обработку данных (7), где предикаты:

BalanceBracket – предикат истинный, если количество открытых (левых) скобок равно количеству закрытых (правых) скобок в анализируемом списке;

RightBracket – предикат истинный, если в анализируемом списке следующим символом является правая скобка вида “)”;

LeftAtomRight – предикат истинный, если в анализируемом списке следующими символами являются левая скобка – атом – правая скобка вида “(“ Atom “)”;

и операторы:

CreateTop – создать вершину стека;

AddMeaning – прибавить значение в вершину стека;

EraseTop – ликвидировать вершину стека;

Shift(QuantitySymbols) – сдвиг показателя вправо по анализируемому списку на количество символов QuantitySymbols.

Рассмотрим теперь двухосновную алгебру (8), в которой на основах IF и OP определена сигнатура операций Sign и в которую включены логические операции: дизъюнкция, конъюнкция и отрицание, определенные на основе IF, а также операции композиции, альтернативы и цикла, принимающие свои значения на множестве OP. Формально – (9).

Необходимо отметить, что для сигнатуры Sign операторы – (10), а предикат – (11). Сигнатура Sign содержит классические структуры, заложенные Дейкстрой в основу технологии структурного программирования.

Идея алгоритма: список анализируется с помощью оператора Shift() слева направо с начала до конца, то есть к тому моменту, когда количество открытых (левых) скобок стает равным количеству закрытых (правых) скобок. Работа алгоритма основана на использовании двухэлементного n-адресного стека. Первый элемент стека используется для сохранения атомов, а второй – для рекурсивного определения их оценок на основе вложенных подсписков и/или подчиненных атомов. В процессе рекурсивного анализа в стек прибавляются подчиненные подсписки для того, чтобы оценить их на основе соответствующих атомов и/или вложенных подсписков; рекурсивная оценка завершается при соответствующем изъятии элемента стека тогда, когда подчиненными на данном узле остаются лишь атомы.

Рекурсивный алгоритм оценки дерева содержательного материала, который имеет семь терминальных вершин, можно представить в формульном виде структурной схемой (12). Если считать системой образующих алгебры (13), где (14), тогда алгоритм EVAL_LIST по сути представляет собою составной оператор в виде терма – суперпозиции элементов из S и операций из Sign. В графическом виде рекурсивный алгоритм можно представить деревом операций, изображенным на рис.1.

Макро-операторы (7), в свою очередь, можно представить на микро-уровне элементарными операторами обработки нелинейных списочных структур (15), (16) и (17).

## Development of analytical evaluation algorithm

An electronic textbook can be represented as a combination of two constituents in the following formula (1).

Let us assume that the structure of the electronic textbook is similar to the content-target structure of the educational material presented in [3].

It is believed that a typical electronic textbook consists of modules (sections) composed of topics, which in turn demonstrate a sequel of elementary educational doses. It is an example of the hierarchic structure as a model of the subject matter area in the form of a non-linear list (2). The list (2) can be demonstrated more visually in the form of the so-called structured print in Lisp programming language [4] (Addition 2).

In order to develop a formula of a student model Sd, that is to be created on the basis of the content-target structure test control of the educational material Sa, we will introduce the following algebraic system (3), (4), (5).

If the list structures are regarded as the basis of the algebraic system, then a signature Main_Sign as a combination of predicates and operations will be considered on two levels of abstraction, namely: high (macro-level) and low (micro-level). At first we will pay attention to a macro-level. Predicates' multiplication, determined by the sum of data being processed (6) and for operators' multiplication implemented data processing (7), where

a. Predicates:

BalanceBracket – a true predicate if a number of open (left) brackets is equal to a number of closed (right) brackets in the list being analyzed;

RightBracket – a true predicate if the right bracket ")" is the next symbol in the list being analyzed;

LeftAtomRight – a true predicate in case when the left bracket – atom – the right bracket "(" Atom ")" are next symbols in the list being analyzed;

b. Operators:

CreateTop – to create a stack top;

AddMeaning – to add the meaning to a stack top;

EraseTop – to erase a stack top;

Shift (QuantitySymbols) – to shift an indicator to the right according to a number of QuantitySymbols in the list being analyzed.

Let us examine now the two-based algebra (8), where there are two bases IF i OP which determine an operation signature Sign, the latter has been included logic

(Boolean) operations, namely disjunction, conjunction, and negation, identified on IF base, as well as operations of composition, alternative and cycle, grounded on OP multiplication. The formula is as follows – (9).

It should be stressed upon, that operators – (10) and a predicate – (11) are meant for the above-mentioned Sign signature. Sign signature includes classic programming structures, which have been supplemented by Dijkstra E. W. in the technology of structured programming.

Algorithm idea: a list is analyzed with the assistance of Shift () operator from the left to the right from the beginning to the end, i.e. until the moment when a number of open (left) brackets becomes equal to a number of closed (right) brackets. Algorithm's operation based on the use of the two-element n-address stack. As the recursive analysis goes, subordinate sublists are submitted to a stack in order to evaluate them on the basis of appropriate atoms and/or sublists included. The recursive evaluation is finished with corresponding extraction of a stack element at the time when only atoms are left to be subordinate on the given hub.

The first element of the stack is used for atoms' preservation, the second one for recursive identification of their evaluation on the basis of sublists included and/or subordinate atoms.

A recursive evaluation algorithm of the subject matter tree having seven terminal tops can be expressed by the structured scheme (12). If (13) is regarded as a system of algebra constituents, where (14) then EVAL_LIST algorithm represents itself a composed operator in the form of term – superposition of elements with S and operations with Sign. The recursive algorithm can be graphically demonstrated the operation tree in figure 1.

Macro-operators (7) can be in turn represented on the micro-level as elementary operators of processing non-linear list structures (15), (16), (17).

## Висновки

В роботі досягнуто загального аналітичного розв'язку задачі визначення складових тестового контролю (побудови моделі студента) згідно з нелінійною списковою структурою навчального матеріалу (моделлю

предметної області). Практичну реалізацію алгоритму на мові Common Lisp представлено у Додатку 1, а результат роботи програму – у Додатку 2.

## Выводы

В работе достигнуто общее аналитическое решение задачи определения составляющих тестового контроля (построения модели студента) в соответствии с нелинейной списочной структурой учебного материала (моделью предметной области). Практическая реализация алгоритма на языке Common Lisp представлена в Приложении 1, а результат работы программы – в Приложении 2.

## Conclusion

General analytical decision of the objective concerning the determination of test control constituents (a student model development) according to the non-linear list structure of the educational material (a model of the subject matter area) has been elaborated in the work. Algorithm’s practical realization in Common Lisp programming language is demonstrated in Addition 1. The result of algorithm’s work is shown in Addition 2.

## Література (Список литературы, Literature)


1. F. T. Tschang and T. Della Senta (eds.): Access to Knowledge: New Information Technologies and the Emergence of the Virtual University. Amsterdam: Elsevier Science and International Association of Universities, 2001, 167-206.
2. Минасов Ш.М., Тархов С.В. Проект "Гефест" как вариант практической реализации технологий электронного обучения в вузе в условиях интеграции традиционного и дистанционного обучения / Educational Technology & Society, 8(1) 2005, с. 134-147.
3. Бісікало О.В. Підхід до створення електронних підручників з тестуючими компонентами на основі моделі адаптивного навчання. В збірнику “Контроль та управління в складних системах. (КУСС-2003). Матеріали сьомої міжнародної науково-технічної конференції. м. Вінниця, 8-11 жовтня 2003 року.” – Вінниця: УНІВЕРСУМ-Вінниця, 2003. с.227-234.

4. Хювенен Э., Сеппянен Й. Мир Лиспа. В 2-х т. Т.1: Введение в язык Лисп и функциональное программирование. Пер. с финск. – М.: Мир, 1990. – 447с.

Відомості про авторів (Сведения про авторов, Authors information)

Бісікало Олег Володимирович,
к.т.н., доц.,
доцент кафедри економічної кібернетики та інформатики
Вінницького державного аграрного університету,
+38 (0432) 53-22-61, +38 (067) 580-04-19,
inter@vsau.com , agoffice@svitonline.com

Бисикало Олег Владимирович,
к.т.н., доц.,
доцент кафедры экономической кибернетики и информатики
Винницкого государственного аграрного университета,
+38 (0432) 53-22-61, +38 (067) 580-04-19,
inter@vsau.com , agoffice@svitonline.com

Oleg V. Bisikalo,
Ph.D., Associate Professor,
Department of Economic Cybernetics and Informatics
Vinnytsia State Agricultural University,
+38 (0432) 53-22-61, +38 (067) 580-04-19,
inter@vsau.org , agoffice@svitonline.com

Додаток 1 (Приложение 1, Addition 1)

```
(defun dft (tree n)
   (cond ((null tree) ())
         ((not (consp tree)) (format t "~a~%" (pad n tree)))
         (t (dft (car tree) (1+ n))
           (dft (cdr tree) n))))
(defun pad (n s)
  (format nil "~a~a" (make-string n :initial-element #\Space) s))

(defparameter *tt* '(1 (11 (111 (1111) (1112)) (112 (1121) (1122))) (12 (121
(1211) (1212) (1213)) (122 (1221)) (123 (1231) (1232))) (13 (131 (1311) (1312))
(132 (1321) (1322)))))
```

Додаток 2 (Приложение 2, Addition 2)

```
CL-USER> (dft *tt* 0)
 1
  11
   111
    1111
    1112
   112
    1121
    1122
  12
   121
    1211
    1212
    1213
   122
    1221
   123
    1231
    1232
  13
   131
    1311
    1312
   132
    1321
    1322
NIL
CL-USER>
```