МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Севастопольский государственный университет»

кафедра Информационные системы

Бариев Эмин Юсуфович

Институт информационных технологий и управления в технических системах

курс 4 группа ИС/б-16-2

09.03.02 Информационные системы и технологии (уровень бакалавриата)

ОТЧЕТ

по лабораторной работе №4

По дисциплине: «Методы и средства искусственного интеллекта»

По теме: «Ея доступ к базе данных на основе алгоритма сопоставления с образцом»

|  |  |  |
| --- | --- | --- |
| Отметка о зачете | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | \_\_\_\_\_\_\_\_\_\_\_ |
|  |  | (дата) |
|  | Руководитель практикума |  |
| ст. преподаватель | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Сметанина Т.И. |
| (должность) | (подпись) | (инициалы, фамилия) |

Севастополь 2019

# **ЦЕЛЬ РАБОТЫ**

Исследование алгоритма сопоставления с образцом и особенностей его применения для формирования запросов к базам данных, а также для организации доступа к базам данных на ограниченном подмножестве естественного языка.

# **ПОСТАНОВКА ЗАДАЧИ**

Для базы данных, созданной в лабораторной работе 3, необходимо написать на языке Лисп интерфейс, который позволяет выполнять ЕЯ-запросы с помощью алгоритма сопоставления с образцом. Ответ на запрос должен также представляться на естественном языке в виде списка слов предложения. Предусмотреть 5-6 различных запросов.

# **ХОД РАБОТЫ**

# Согласно с вариантом задания была разработана следующая программа:

(defvar \*database\* ())

(defvar dbFileName "lr3\_db.dat")

(defun Insert(fio group MATH PHYS HIST REFCT OOP)(

PUSH (LIST :FIO fio :GROUP group :PERFOMANCE (LIST :MATH MATH :PHYS PHYS :HIST HIST :REFCT REFCT :OOP OOP)) \*database\*

))

(defun ShowDb()

(format t "~%~{ ~{~a:~12t~a ~}~%~}" \*database\*)

)

(defun SaveToFile(fileName)(

with-open-file (out fileName :if-does-not-exist :create :direction :output :if-exists :supersede)

(with-standard-io-syntax (print \*database\* out))

))

(defun ConnectToFile(fileName)

(setf \*database\* ())

(with-open-file (in fileName :direction :input)

(with-standard-io-syntax (setf \*database\* (read in))))

)

(defun fio-selector(fio)

#'(lambda(x) (equal (getf x :FIO) fio))

)

(defun FirstOrDefaul(selector-func)(

remove-if-not selector-func \*database\*

))

(defun Update(selector-func &KEY group MATH PHYS HIST REFCT OOP)

(setf \*database\*

(mapcar #'(lambda (x)

(when (funcall selector-func x)

(if group (setf (getf x :group) group))

(if (OR MATH PHYS HIST REFCT OOP)

;(lambda(perfomance)())

(SETF (GETF x :PERFOMANCE)

(LIST

:MATH (if (not (equal MATH nil)) MATH (GETF (GETF x :PERFOMANCE) :MATH))

:PHYS (if (not (equal PHYS nil)) PHYS (GETF (GETF x :PERFOMANCE) :PHYS))

:HIST (if (not (equal HIST nil)) HIST (GETF (GETF x :PERFOMANCE) :HIST))

:AIP (if (not (equal REFCT nil)) AIP (GETF (GETF x :PERFOMANCE) :AIP))

:LITR (if (not (equal OOP nil)) LITR (GETF (GETF x :PERFOMANCE) :LITR))

)

)

)

)

x)

\*database\*)

)

)

(defun PrintAllWith4And5()

(let ((tempList (remove-if-not #'(lambda(x)(

OR (MEMBER 4 (GETF x :PERFOMANCE)) (MEMBER 5 (GETF x :PERFOMANCE)))

) \*database\*))

)

(IF (null tempList) (PRINT "Я НИЧЕГО НЕ НАШЕЛ")

(MAPCAR #'(lambda(x)(format t "~%~{~12t ~a~}" (LIST (GETF x :FIO) (GETF x :GROUP)))) tempList)

)

)

)

(defun match (p d)

(cond

((and (null p) (null d)) t)

((and (null d) (equal (car p) '$) (null (cdr p))) t)

((or (null p) (null d)) nil)

((or (equal (car p) '?) (equal (car p) (car d))) (match (cdr p) (cdr d)))

((equal (car p) '$)

(cond

((match (cdr p) d) t)

((match p (cdr d)) t)

))

((and (not (atom (car p))) (not (atom (car d))) (match (car p) (car d))) (match (cdr p) (cdr d)))

((and (atom (car p)) (equal (car-letter (car p)) #\?) (match (cdr p) (cdr d))) (set (cdr-name (car p)) (car d)) t)

)

)

(defun get-matches(p db)

(cond

((null db) nil)

((match p (car db)) (cons (car db) (get-matches p (cdr db))))

(t (get-matches p (cdr db)))

)

)

(defun car-letter (x) (if (not (numberp x)) (car (coerce (string x) 'list))))

(defun cdr-name (x) (intern (coerce (cdr (coerce (string x) 'list)) 'string)))

(defun getValuesList(propLst)

(let (result (number 0))

(DOLIST (temp (reverse propLst) result)

(IF (evenp number) (setq result (cons temp result)))

(setq number (+ number 1))

)

)

)

(defun average(lst)

(when lst

(FLOAT (/ (apply #'+ lst) (length lst)))

)

)

(defun dialog(fact)

(cond

((match '($ всех $ отличников $) fact)(format t "Я нашел писок квадратноголовых: ~a ~%" (get-matches '($ ($ :MATH 5 :PHYS 5 :HIST 5 :REFCT 5 :OOP 5 $) $) \*database\*)))

((match '($ информацию $ по фамилии ?FIO $) fact) (format t "Данные студента: ~a ~%" (get-matches (LIST '$ :FIO FIO '$) \*database\*)))

((match '($ оценки $ по фамилии ?FIO $) fact)(format t "Вот оценки искомого студента: ~a ~%" (GETF (car (get-matches (LIST '$ :FIO FIO '$) \*database\*)) :PERFOMANCE)) )

((match '($ ?RANK по физике $) fact) (format t "Адепты Эйнштейна: ~a ~%" (get-matches (LIST '$ (LIST '$ :PHYS RANK '$) '$) \*database\*)))

(

(and (match '($ средний бал $ по фамилии ?FIO $) fact) (not (null (get-matches (LIST '$ :FIO FIO '$) \*database\*))))

(format t "Срединий бал этого студента: ~a ~%" (average (getValuesList (GETF (car (get-matches (LIST '$ :FIO FIO '$) \*database\*)) :PERFOMANCE))))

;(print (getValuesList (GETF (car (get-matches (LIST '$ :FIO FIO '$) \*database\*)) :PERFOMANCE)))

)

(t (print "МОЯ ТВОЯ НЕ ПОНИМАЕТ"))

)

)

(Insert 'Симпсон 'ИС-42 5 4 3 2 1 )

(Insert 'Гейтс 'ИС-42 5 5 5 5 5 )

(Insert 'Маск 'ИС-42 5 4 4 3 5 )

(Insert 'Дамблдор 'ИС-42 5 4 4 2 2 )

(Insert 'Джобс 'ИС-42 5 2 4 2 2 )

(setq fact '(найти всех отличников))

(format t "~{ ~a~} ~%" fact )

(dialog fact)

(format t "~%~%")

(setq fact '(покажи студентов с оценками 5 по физике))

(format t "~{ ~a~} ~%" fact )

(dialog fact)

(format t "~%~%")

(setq fact '(найди информацию студента по фамилии Симпсон))

(format t "~{ ~a~} ~%" fact )

(dialog fact)

(format t "~%~%")

(setq fact '(найди оценки студента по фамилии Дамблдор))

(format t "~{ ~a~} ~%" fact )

(dialog fact)

(format t "~%~%")

(setq fact '(покажи средний бал студента по фамилии Джобс))

(format t "~{ ~a~} ~%" fact )

(dialog fact)

(format t "~%~%")

(setf \*database\* nil)

# **РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ПРОГРАММЫ**

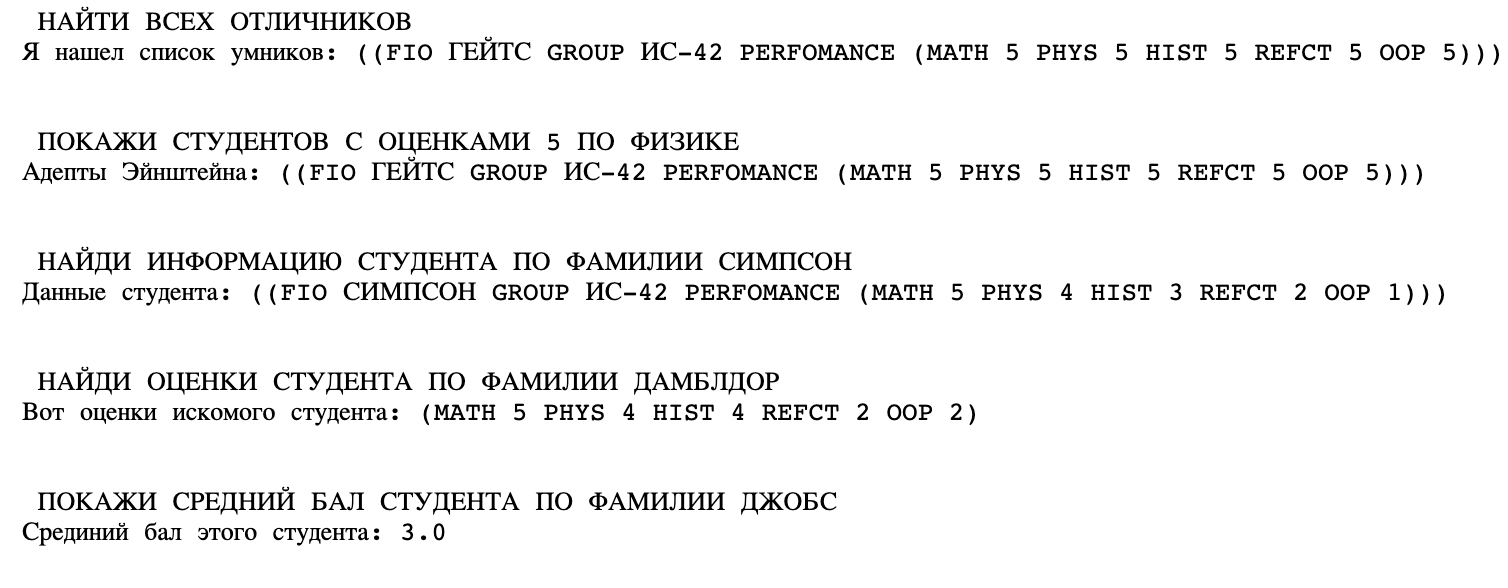


Рисунок 1 – Результат выполнения программы

# **ВЫВОДЫ**

В ходе выполнения лабораторной работы были исследованы технологии подготовки и выполнения Лисп-программ в интегрированной среде разработки LispWorks. Так же были исследованы способы обращения к базе данных на естественном языке, обрабатываемые при помощи алгоритма сопоставления с образцом.

Для закрепления теоретических знаний, была разработана программа, позволяющая выполнять запросы пользователя к базе данных, на ограниченном подмножестве естественного языка.

По результатам выполнения лабораторной работы, можно сделать вывод, что поставленная цель достигнута.