МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное

учреждение высшего образования

«Санкт–Петербургский государственный университет  
аэрокосмического приборостроения»

Кафедра проблемно-ориентированных вычислительных комплексов

ОТЧЁТ ПО ПРАКТИКЕ

ЗАЩИЩЁН С ОЦЕНКОЙ

Руководитель

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Доц., к.т.н. |  |  |  | С.Г. Толмачев |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

ОТЧЁТ ПО ПРАКТИКЕ

|  |  |  |
| --- | --- | --- |
| вид практики | Производственная | |
| тип практики |  | |
| на тему индивидуального задания | | «Прототип экспертной системы по выбору литературного произведения» |
|  | | | |
|  | | | |

|  |  |
| --- | --- |
| выполнена | Шубыревой Любовью Артуровной |
| фамилия, имя, отчество обучающегося в творительном падеже | |

|  |  |  |  |
| --- | --- | --- | --- |
| по направлению подготовки | 01.03.02 |  | Прикладная математика и информатика |
|  | Код |  | наименование направления |
| направленности | 01 |  | Исследование операций, системный анализ и управление динамическими системами |
|  | Код |  | наименование направленности |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Обучающийся группы № | 4518 |  |  |  | Т.С.Тимановская |
|  | номер |  | подпись, дата |  | инициалы, фамилия |

Санкт–Петербург 2018

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

на прохождение производственной практики обучающегося направления подготовки

1. Фамилия, имя, отчество обучающегося: Тимановская Татьяна Сергеевна
2. Группа: 4518
3. Тема индивидуального задания:

«Прототип экспертной системы по выбору стиля музыки»

1. Исходные данные:

Факты: литературные жанры, сюжетные повороты.

Правила: выбору стиля музыки в соответствии с алгоритмом .

1. Содержание отчетной документации:
   1. индивидуальное задание;
   2. отчёт, включающий в себя:
   * титульный лист;
   * содержательная часть;
   * выводы по результатам практики;
   * список использованных источников.
   1. Электронная копия документации и *(при необходимости)* файлы, содержащие разработанный программный продукт.
2. Срок представления отчета на кафедру: «        »                             2018       г.

Руководитель практики

Доц., к.т.н..          С.Г. Толмачев

должность, уч. степень, звание подпись, дата инициалы, фамилия

Задание принял к исполнению:

Обучающийся

Т.С.Тимановская

дата подпись инициалы, фамилия

Оглавление

[Введение 4](#_Toc524394199)

[Дерево принятий решений 4](#_Toc524394200)

[Таблица признаков 5](#_Toc524394201)

[Правила и факты 6](#_Toc524394202)

[Результат работы экспертной системы 11](#_Toc524394203)

[Заключение 13](#_Toc524394204)

[Список используемой литературы 13](#_Toc524394205)

[Приложение 14](#_Toc524394206)

# Введение

Среда CLIPS предназначается для создания экспертных систем. В данной работе будет реализована экспертная система, которая предлагает пользователю данной системы возможность выбрать ему книгу, система предложит вам несколько вариантов стилей и жанров книг из которых вы можете подобрать себе подходящую. Реализация данной экспертной системы будет использовать управляющие команды CLIPS, такие как if-then-else и while. Пользователь вручную задает параметры, которые его интересуют, отвечая на представленные вопросы yes/no.

# Дерево принятий решений

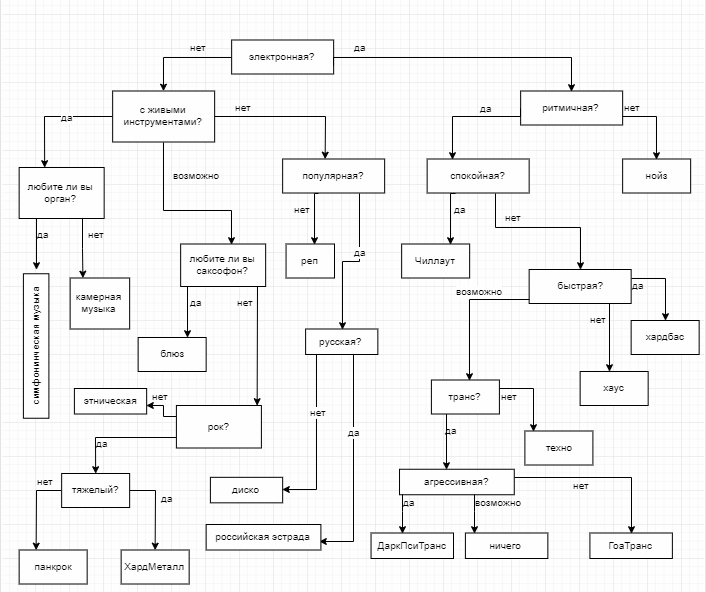


Рис.1 Дерево вопросов

# Таблица признаков

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | объект  Признаки (факты) | направление | | | характер | | | | | | | Страна | Живые инструменты  Отношение к | | | | | Отношение к  органу | | Отношение к быстрой музыке | | | | | Обобщенный жанр | | Отношение к  саксофону | | |
| электронная | популярная | | ритмичная | | спокойная | тяжелая | | | агрессивная | Россия | | да | | возможно | нет | позитивное | негативное | определенно | возможно | | нет | | рок | транс | позитивное | | негативное |
| 1 | Симфоническая музыка |  |  | |  | |  | |  | |  |  | + | |  | |  | + |  |  |  | | |  |  |  | |  |  |
| 2 | Камерная музыка |  |  | |  | |  | |  | |  |  | + | |  | |  |  | + |  |  | | |  |  |  | |  |  |
| 3 | блюз |  |  | |  | |  | |  | |  |  |  | | + | |  |  |  |  |  | | |  |  |  | | + |  |
| 4 | Панкрок |  |  | |  | |  | |  | |  |  |  | | + | |  |  |  |  |  | | |  | + |  | |  | + |
| 5 | ХардМетал |  |  | |  | |  | | + | |  |  |  | | + | |  |  |  |  |  | | |  | + |  | |  | + |
| 6 | Реп |  |  | |  | |  | |  | |  |  |  | |  | | + |  |  |  |  | | |  |  |  | |  |  |
| 7 | Диско |  | + | |  | |  | |  | |  |  |  | |  | | + |  |  |  |  | | |  |  |  | |  |  |
| 8 | Российская эстрада |  | + | |  | |  | |  | |  | + |  | |  | | + |  |  |  |  | | |  |  |  | |  |  |
| 9 | Нойз | + |  | |  | |  | |  | |  |  |  | |  | |  |  |  |  |  | | |  |  |  | |  |  |
| 10 | Чиллаут | + |  | | + | | + | |  | |  |  |  | |  | |  |  |  |  |  | | |  |  |  | |  |  |
| 11 | Хаус | + |  | | + | |  | |  | |  |  |  | |  | |  |  |  |  |  | | | + |  |  | |  |  |
| 12 | Хардбас | + |  | | + | |  | |  | |  |  |  | |  | |  |  |  | + |  | | |  |  |  | |  |  |
| 13 | Техно | + |  | | + | |  | |  | |  |  |  | |  | |  |  |  |  | + | | |  |  |  | |  |  |
| 14 | ДаркТранс | + |  | | + | |  | |  | |  |  |  | |  | |  |  |  |  | + | | |  |  | + | |  |  |
| 15 | ГоаТранс | + |  | + | |  | |  | |  | |  |  | |  | |  |  |  |  | | + | |  |  | + | |  |  |
| 16 | Этническая |  |  |  | |  | |  | |  | |  |  | | + | |  |  |  |  | |  | |  |  |  | |  | + |

# Правила и факты

**Факты, описывающие электронную музыку:**

electr-type ok – включаем электронную музыку в возможные;

electr-type bad – исключаем электронную музыку;

**Факты, описывающие популярную музыку:**

pop-type ok – включаем популярную музыку в возможные;

pop-type bad – исключаем популярную музыку;

**Факты, описывающие ритмичную музыку:**

rithm-type ok – включаем ритмичную музыку в возможные;

rithm -type bad – исключаем ритмичную музыку;

**Факты, описывающие спокойную музыку:**

quiet-type ok – включаем спокойную музыку в возможные;

quiet -type bad – исключаем спокойную музыку;

**Факты, описывающие тяжелую музыку:**

heavy-type ok – включаем тяжелую музыку в возможные;

heavy -type bad – исключаем тяжелую музыку;

**Факты, описывающие агрессивную музыку:**

agr-type ok – включаем агрессивную музыку в возможные;

agr -type bad – исключаем агрессивную музыку;

**Факты, описывающие российскую музыку:**

rus-type ok – включаем российскую музыку в возможные;

rus -type bad – исключаем российскую музыку;

**Факты, описывающие наличие живых инструментов в музыке:**

instr-type yes – включаем наличие живых инструментов в возможные;

instr-type maybe – наличие живых инструментов возможно;

instr-type no – исключаем наличие живых инструментов;

**Факты, описывающие отношение к органу в музыке:**

org-type ok – включаем наличие органа в возможные;

org -type bad – исключаем наличие органа;

**Факты, описывающие отношение к органу в музыке:**

sax-type ok – включаем наличие органа в возможные;

sax -type bad – исключаем наличие органа;

**Факты, описывающие скорость музыки:**

speed-type yes – включаем быструю музыку в возможные;

speed-type maybe – предпочтителен средний темп;

speed-type no – исключаем быструю музыку;

**Факты, описывающие стиль рок:**

rock-type ok – включаем стиль рок в возможные;

rock-type bad – исключаем стиль рок

**Факты, описывающие стиль транс:**

trance-type ok – включаем стиль транс в возможные;

trance-type bad – исключаем стиль транс;

**Правила**

**Функция ask-question**

*Функция, задающая пользователю произвольный вопрос и получающая ответ из заданного набора корректных ответов.*

(deffunction ask-question (?question $?allowed-values)

(printout t ?question)

(bind ?answer (read))

(if (lexemep ?answer)

then

(bind ?answer (lowcase ?answer)))

(while (not (member ?answer ?allowed-values)) do

(printout t ?question)

(bind ?answer (read))

(if (lexemep ?answer)

then

(bind ?answer (lowcase ?answer))))

?answer)

**Функция yes-or-no-p**

*Функция, задающая пользователю вопрос и допускающий ответ в виде да/нет, т. к. это один из самых распространенных типов вопросов.*

(deffunction yes-or-no-p (?question)

(bind ?response (ask-question ?question yes no у n))

(if (or (eq ?response yes) (eq ?response y))

then

TRUE

else

FALSE))

*Функция yes-or-no-p вызывает функцию ask-question с постоянным набором допустимых ответов: yes, no, у и n. В случае если пользователь ввел ответ yes или у, функция возвращает значение true, иначе - false. Поскольку функция yes-or-no-p использует функцию ask-question, то она должна быть определена после нее.*

**Правило, которое определяет, нравится ли пользователю электронная музыка**

(defrule determine-electr-type ""

(not (electr-type ?))

(not (music ?))

=>

(if (yes-or-no-p "The music should be elecronical(yes/no)? ")

then

(assert (electr-type ok))

else

(assert (electr-type bad))))

**Правило, которое определяет, нравится ли пользователю ритмичная музыка**

(defrule determine-rithm-type ""

(electr-type ok)

(not (rithm-type ?))

(not (music ?))

=>

(if (yes-or-no-p "Do you like rithmyc music (yes/no)? ")

then

(assert (rithm-type ok))

else

(assert (music “liten to noise style”))))

**Правило, которое определяет, нравится ли пользователю спокойная музыка**

(defrule determine-quite-type ""

(electr-type ok)

(rithm-type ok)

(not (quite-type ?))

(not (music ?))

=>

(if (yes-or-no-p "Do you like quite music? ")

then

(assert (music “try listen to chillout music"))

else

(assert (quite-type bad))))

**Правило, которое определяет, нравится ли пользователю быстрая музыка**

(defrule determine-speed-type ""

(electr-type ok)

(rithm-type ok)

(quite-type bad)

(not (speed-type ?))

(not (music ?))

=>

(bind ?response (ask-question "do u like fast music (yes/no/maybe)? " no yes maybe)

(if (eq ?response no)

then (assert (music “u will enjoy HOUSE music”))

else

(if (eq ?respone yes)

then (assert (music “u will enjoy HARDBASS music”))

else (assert (speed-type maybe))

)

))))

**Правило, которое определяет, нравится ли пользователю трансовая музыка**

(defrule determine-trance-type ""

(electr-type ok)

(rithm-type ok)

(quite-type bad)

(speed-type maybe)

(not (trance-type ?))

(not (music ?))

=>

(if (yes-or-no-p "Do you like trance music? ")

then

(assert (trance-type ok))

else

(assert (music “Try listen to Techno music"))))

**Правило, которое определяет, нравится ли пользователю агрессия в музыке**

(defrule determine-agr-type ""

(electr-type ok)

(rithm-type ok)

(quite-type bad)

(speed-type maybe)

(trance-type ok)

(not (agr-type ?))

(not (music ?))

=>

(bind ?response (ask-question "Do you like aggression in music (no/yes/maybe)? " no yes maybe))

(assert (music (switch ?response

(case no then "u should try GOA-Trance")

(case yes then "U will enjoy DarkPsyTrance")

(case maybe then "ur taste is anormal, dude")

)))

)

**Правило, которое определяет, нравится ли пользователю живые инструменты в музыке**

(defrule determine-instr-type ""

(electr-type bad)

(not (instr-type ?))

(not (music ?))

=>

(bind ?response (ask-question "do u like the real sound of instruments (yes/no/maybe)? " no yes maybe))

(assert (instr-type ?response))

)

**Правило, которое определяет, нравится ли пользователю звучание органа**

(defrule determine-org-type ""

(electr-type bad)

(instr-type maybe)

(not (org-type ?))

(not (music ?))

=>

(if (yes-or-no-p "Do you like sound of organ? ")

then

(assert (music “u need to listen symphonic music”))

else

(assert (music “Try to listen some camer music"))))

**Правило, которое определяет, нравится ли пользователю звучание саксофона**

(defrule determine-sax-type ""

(electr-type bad)

(instr-type maybe)

(not (sax-type ?))

(not (music ?))

=>

(if (yes-or-no-p "Do you like sound of sax? ")

then

(assert (music “u need to listen blues music”))

Else (assert (sax-type bad))))

**Правило, которое определяет, нравится ли пользователю рок музыка**

(defrule determine-rock-type ""

(electr-type bad)

(instr-type maybe)

(sax-type bad)

(not (rock-type ?))

(not (music ?))

=>

(if (yes-or-no-p "Do you like rock style? ")

then

(assert (rock-type ok))

Else (assert (music “try to listen ethnic music”))))

**Правило, которое определяет, нравится ли пользователю тяжелая музыка**

(defrule determine-heavy-type ""

(electr-type bad)

(instr-type maybe)

(sax-type bad)

(rock-type ok)

(not (heavy-type ?))

(not (music ?))

=>

(if (yes-or-no-p "Do you like heavy music? ")

then

(assert (music “for u – the best is HardMetall”))

Else (assert (music “u would enjoy PunkRock”))))

**Правило, которое определяет, нравится ли пользователю популярная музыка**

(defrule determine-pop-type ""

(electr-type bad)

(instr-type no)

(not (pop-type ?))

(not (music ?))

=>

(if (yes-or-no-p "Do you like popular music? ")

then

(assert (pop-type ok))

Else (assert (music “try Rap music”))))

**Правило, которое определяет, нравится ли пользователю российская музыка**

(defrule determine-rus-type ""

(electr-type bad)

(instr-type no)

(pop-type ok)

(not (rus-type ?))

(not (music ?))

=>

(if (yes-or-no-p "Do you understand English songs? ")

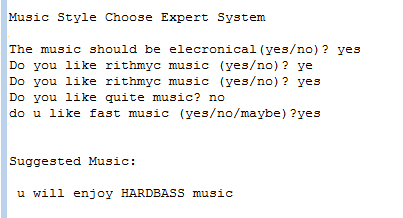
then

(assert (music “u should listen to Disco”))

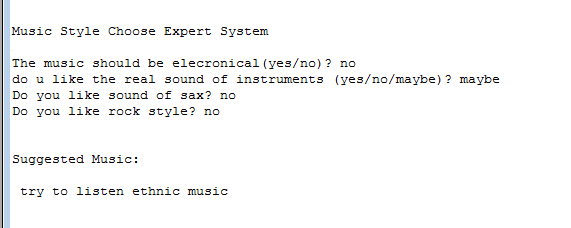
Else (assert (music “try russian estrade seems u will like it”))))

# Результат работы экспертной системы

Хардбасс

  
Рис.2 Пример 1

Этническая музыка

  
Рис.3 Пример 2

Disco

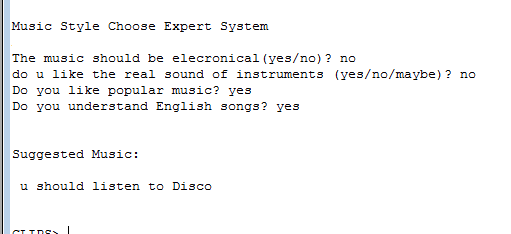


Рис.4 Пример 3

Пользователь не попал ни в одну из категорий:

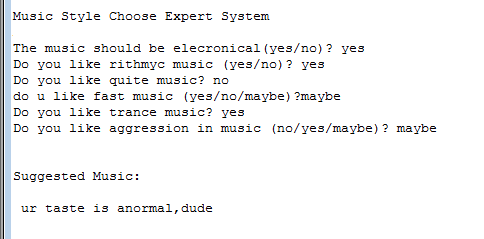


Рис.5 Пример 4

# Заключение

В результате выполнения данного задания были получены базовые навыки программирования в системе CLIPS 6.24. Более того, была создана экспертная система, которая позволяет пользователю выбрать книгу для чтения.

Проделав эту работу, можно заключить, что, используя факты, функции и правила, возможно создание экспертной системы, позволяющей свести к минимуму участие пользователя в решении глобальных задач, путем разбиения таковых на вопросы, что в значительной степени упрощает решение задачи выбора перед пользователем.

# Список используемой литературы

1. Джарратано Дж. Экспертные системы: принципы разработки и программирование/ Дж. Джарратано, Г. Райли. – Москва: Вильямс, 2007. – 1152 с.

2.Частиков А.П. Разработка экспертных систем. Среда CLIPS/ А.П. Частиков, Т.А. Гаврилова, Д.Л. Белов. – СПб.: БХВ-Петербург, 2003. — 608 с.

# Приложение

(deffunction ask-question (?question $?allowed-values)

(printout t ?question)

(bind ?answer (read))

(if (lexemep ?answer)

then (bind ?answer (lowcase ?answer)))

(while (not (member ?answer ?allowed-values)) do

(printout t ?question)

(bind ?answer (read))

(if (lexemep ?answer)

then (bind ?answer (lowcase ?answer))))

?answer)

(deffunction yes-or-no-p (?question)

(bind ?response (ask-question ?question yes no y n))

(if (or (eq ?response yes) (eq ?response y))

then TRUE

else FALSE))

(defrule determine-electr-type ""

(not (electr-type ?))

(not (music ?))

=>

(if (yes-or-no-p "The music should be elecronical(yes/no)? ")

then

(assert (electr-type ok))

else

(assert (electr-type bad))))

(defrule determine-rithm-type ""

(electr-type ok)

(not (rithm-type ?))

(not (music ?))

=>

(if (yes-or-no-p "Do you like rithmyc music (yes/no)? ")

then

(assert (rithm-type ok))

else

(assert (music "liten to noise style"))))

(defrule determine-quite-type ""

(electr-type ok)

(rithm-type ok)

(not (quite-type ?))

(not (music ?))

=>

(if (yes-or-no-p "Do you like quite music? ")

then

(assert (music "try listen to chillout music"))

else

(assert (quite-type bad))))

(defrule determine-speed-type ""

(electr-type ok)

(rithm-type ok)

(quite-type bad)

(not (music ?))

=>

(bind ?response (ask-question "do u like fast music (yes/no/maybe)?" yes no maybe))

(if (eq ?response no)

then (assert (music "u will enjoy HOUSE music"))

else

(if (eq ?response yes)

then (assert (music "u will enjoy HARDBASS music"))

else (assert (speed-type maybe)))))

(defrule determine-trance-type ""

(electr-type ok)

(rithm-type ok)

(quite-type bad)

(speed-type maybe)

(not (trance-type ?))

(not (music ?))

=>

(if (yes-or-no-p "Do you like trance music? ")

then

(assert (trance-type ok))

else

(assert (music "Try listen to Techno music"))))

(defrule determine-agr-type ""

(electr-type ok)

(rithm-type ok)

(quite-type bad)

(speed-type maybe)

(trance-type ok)

(not (agr-type ?))

(not (music ?))

=>

(bind ?response (ask-question "Do you like aggression in music (no/yes/maybe)? " no yes maybe))

(assert (music (switch ?response

(case no then "u should try GOA-Trance")

(case yes then "U will enjoy DarkPsyTrance")

(case maybe then "ur taste is anormal,dude")

)))

)

(defrule determine-instr-type ""

(electr-type bad)

(not (instr-type ?))

(not (music ?))

=>

(bind ?response (ask-question "do u like the real sound of instruments (yes/no/maybe)? " no yes maybe))

(assert (instr-type ?response))

)

(defrule determine-org-type ""

(electr-type bad)

(instr-type ok)

(not (org-type ?))

(not (music ?))

=>

(if (yes-or-no-p "Do you like sound of organ? ")

then

(assert (music "u need to listen symphonic music"))

else

(assert (music "Try to listen some camer music"))))

(defrule determine-sax-type ""

(electr-type bad)

(instr-type maybe)

(not (sax-type ?))

(not (music ?))

=>

(if (yes-or-no-p "Do you like sound of sax? ")

then

(assert (music "u need to listen bluse music"))

else (assert (sax-type bad))))

(defrule determine-rock-type ""

(electr-type bad)

(instr-type maybe)

(sax-type bad)

(not (rock-type ?))

(not (music ?))

=>

(if (yes-or-no-p "Do you like rock style? ")

then

(assert (rock-type ok))

else (assert (music "try to listen ethnic music"))))

(defrule determine-heavy-type ""

(electr-type bad)

(instr-type maybe)

(sax-type bad)

(rock-type ok)

(not (heavy-type ?))

(not (music ?))

=>

(if (yes-or-no-p "Do you like heavy music? ")

then

(assert (music "for u - the best is HardMetall"))

else (assert (music "u would enjoy PunkRock"))))

(defrule determine-pop-type ""

(electr-type bad)

(instr-type no)

(not (pop-type ?))

(not (music ?))

=>

(if (yes-or-no-p "Do you like popular music? ")

then

(assert (pop-type ok))

else (assert (music "try Rap music"))))

(defrule determine-rus-type ""

(electr-type bad)

(instr-type no)

(pop-type ok)

(not (rus-type ?))

(not (music ?))

=>

(if (yes-or-no-p "Do you understand English songs? ")

then

(assert (music "u should listen to Disco"))

else (assert (music "try russian estrade seems u will like it"))))

(defrule no\_music ""

(declare (salience 10))

(not (music?))

=>

(accert (music "ur taste is anormal, dude"))

(defrule print-music ""

(declare (salience 10))

(music ?item)

=>

(printout t crlf crlf)

(printout t "Suggested Music:")

(printout t crlf crlf)

(format t " %s%n%n%n" ?item))

(defrule system-banner ""

(declare (salience 10))

=>

(printout t crlf crlf)

(printout t "Music Style Choose Expert System")

(printout t crlf crlf))