Дана база данных вопросов и ответов. Написать программу, которая выводит вопросы на экран в заранее заданном порядке, фиксирует ответы пользователя и проверяет их правильность.

Вариант 13

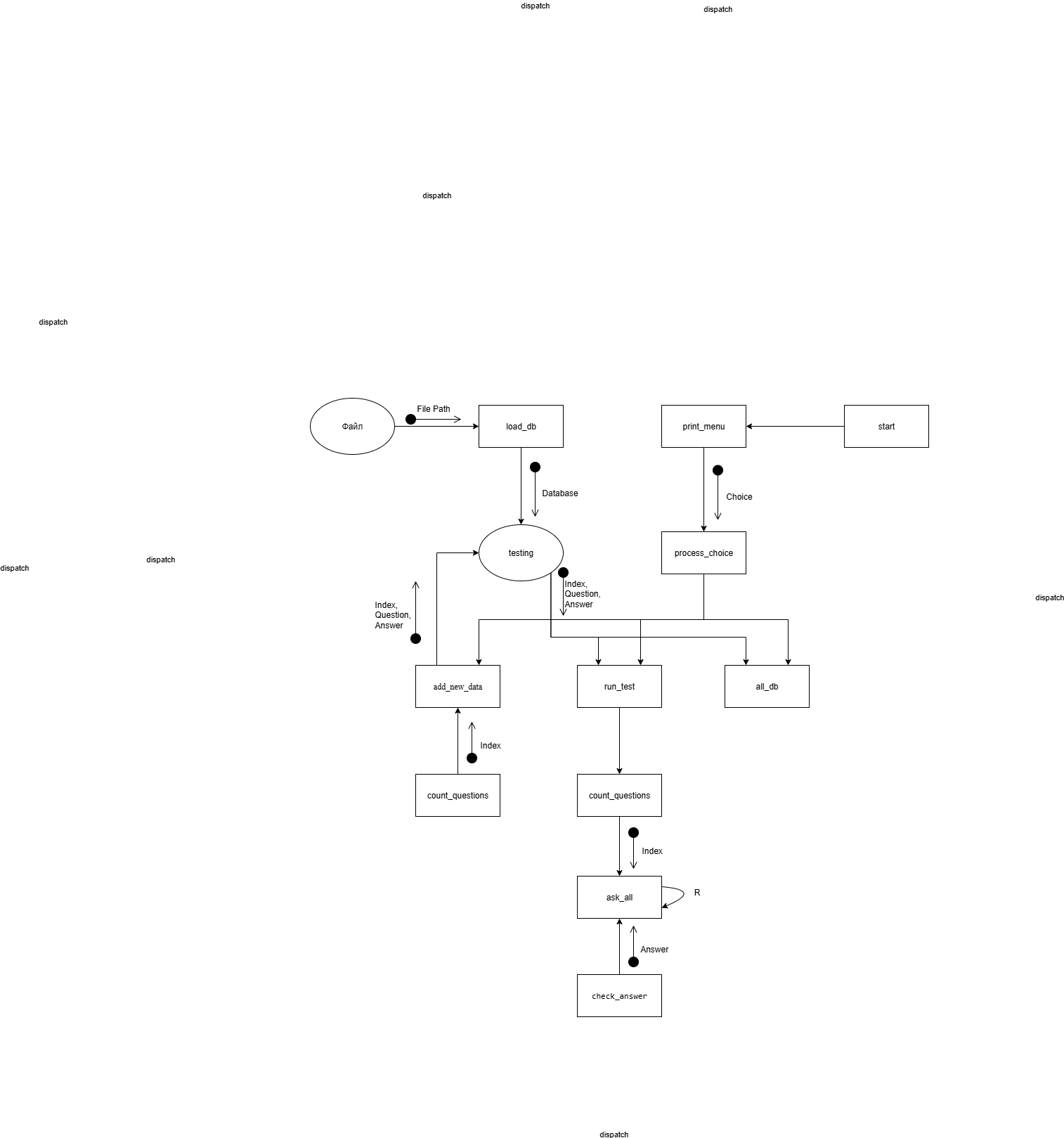


Рисунок 1.Структурная карта Константайна

Domains

question = string % Тип для хранения текста вопроса

answer = string % Тип для хранения текста ответа

index = integer % Тип для хранения номера (индекса) вопроса

% Динамическая база данных для хранения вопросов и ответов

Database

testing(index, question, answer).

Predicates

start

load\_db

print\_menu

process\_choice(integer)

run\_test

count\_questions(integer)

count\_up(integer, integer, integer)

ask\_all(index)

check\_answer(index, answer)

add\_new\_data

all\_db

Clauses

start :-

makewindow(1,27,57,"Testing",0,0,25,80),

print\_menu,

readint(Choice),

process\_choice(Choice).

load\_db :-

existfile("test\_base.dat"), !,

consult("test\_base.dat"),

write("The database is loaded"), nl.

load\_db :-

write("The file 'test\_base.dat' was not found. A new database is being created."), nl.

print\_menu :- nl,

write("=== Menu ==="), nl,

write("1. Take the test"), nl,

write("2. Add a question to the database"), nl,

write("3. Exit"), nl,

write("4. All"), nl,

write("Select an item: ").

process\_choice(1) :- run\_test, start.

process\_choice(2) :- add\_new\_data, start.

process\_choice(3) :- save("test\_base.dat"), retractall(testing(\_, \_, \_)), write("GoodBye!"), nl, !.

process\_choice(4) :- all\_db, readchar(\_), start.

process\_choice(\_) :- write("Wrong Choice! Pick the right number"), readchar(\_), nl, start.

run\_test :-

count\_questions(N),

N > 0, !,

ask\_all(1).

run\_test :- write("The Database is empty"), readchar(\_), nl.

ask\_all(I) :-

count\_questions(N),

I <= N, !,

testing(I, Q, \_),

write("Question number "), write(I), write(": "),

write(Q), nl,

write("Your answer: "),

readln(UserAnswer),

check\_answer(I, UserAnswer),

I1 = I + 1,

ask\_all(I1).

ask\_all(\_).

check\_answer(I, UserAnswer) :-

testing(I, \_, Correct),

upper\_lower(Correct,Correct1),

upper\_lower(UserAnswer,UserAnswer1),

UserAnswer1 = Correct1, !,

write("Correctly!"), readchar(\_), nl, nl.

check\_answer(I, \_) :-

testing(I, \_, Correct),

write("Wrong. Right answer: "), write(Correct), readchar(\_), nl, nl.

add\_new\_data :-

write("Ask a question: "),

readln(Question),

write("Enter the correct answer: "),

readln(Answer),

count\_questions(N),

N1 = N + 1,

assertz(testing(N1, Question, Answer)),

write("Question added."), readchar(\_), nl.

count\_questions(N) :-

count\_up(1, 0, N).

count\_up(I, Acc, N) :-

testing(I, \_, \_), !,

Acc1 = Acc + 1,

I1 = I + 1,

count\_up(I1, Acc1, N).

count\_up(\_, Acc, Acc).

all\_db :-

testing(I, Q, A),

write("testing: "), write(I), write(", "), write(Q), write(", "), write(A), nl,fail.

all\_db :-

write("End").

Goal

load\_db, start.

% Функция создания базы данных

create\_db :-

FileName = "text.txt",

existfile(FileName), % Проверка на существование файла

file\_str(Filename, String), % Чтение файла как строки

makewindow(1,27,57,"",0,0,25,80), % Создание окна на весь экран

string\_to\_words(String, Words), % Разбиение строки на слова

words\_to\_sentence(Words, "", Sentences), % Объединение слов в предложение

retractall(sentence\_db(\_,\_)), % Удаление всех фактов в базе данных

sentences\_into\_db(Sentences, 1), % Заполнение базы данных фактами

write("Data base created"), nl.

create\_db :-

write("File not found"), nl.

string\_to\_words("", []).

string\_to\_words(String, [Word|RestWords]) :- % Разделение строки на слова

fronttoken(String, Word, RestString), % Извлечение слова и остатка строки

string\_to\_words(RestString, RestWords). % Рекурсия

if\_operator(S) :- % Проверка символа

S = "."; S = "!"; S = "?".

words\_to\_sentence([], "", []).

words\_to\_sentence([Word|RestWords], Stroka, [Text|RestText]) :- % Объединение слов в предложение

if\_operator(Word), % Проверка на символ конца предложения

concat(Stroka, Word, Text), % Объединение строки и знака

words\_to\_sentence(RestWords, "", RestText). % Рекурсия

words\_to\_sentence([Word|RestWords], Stroka, Text) :- % Добавление слова в предложение

concat(Stroka, " ", Dop),

concat(Dop, Word, Dop1), % Добавление слова в строку

words\_to\_sentence(RestWords, Dop1, Text). % Рекурсия

sentences\_into\_db([], \_).

sentences\_into\_db([Sentence|RestSentences], N) :- % Запись предложения в базу данных вместе с его номером

assert(sentence\_db(N,Sentence)),

N1 = N + 1,

sentences\_into\_db(RestSentences,N1). % Рекурсия со следущим номером

task:- % Получение интересующего пользователя предложения

write("Enter the sentence number: "),

readint(N1), nl, !,

sentence\_number(N1).

sentence\_number(N1):- % Проверка на наличие предложения с заданным номером

sentence\_db(N1,Sentence), % Получение сохранённого списка слов

write("Sentence number ", N1, ": "), write(Sentence), nl.

sentence\_number(N1):-

write("Sentence number ", N1, " dont exist."), nl.

Clauses

start :-

makewindow(1,27,57,"Testing",0,0,25,80), % Создание окна

print\_menu, % Вывод меню

readint(Choice),

process\_choice(Choice). % Выбор варианта в главном меню

load\_db :- % Загрузка файла, в котором хранится факты базы данных

existfile("test\_base.dat"), !, % Проверка на существование файла

consult("test\_base.dat"), % Если базы фактов

write("The database is loaded"), nl.

load\_db :-

write("The file 'test\_base.dat' was not found. A new database is being created."), nl.

print\_menu :- nl, % Вывод главного меню программы

write("=== Menu ==="), nl,

write("1. Take the test"), nl,

write("2. Add a question to the database"), nl,

write("3. Exit"), nl,

write("4. All"), nl,

write("Select an item: ").

% Обработка выбора пользователя в меню

process\_choice(1) :- run\_test, start. % Запуск опроса и возвращение в меню

process\_choice(2) :- add\_new\_data, start. % Добавить нового вопроса и возврат

process\_choice(3) :- save("test\_base.dat"), retractall(testing(\_, \_, \_)), write("GoodBye!"), nl, !. % Сохранение базы в файл и завершение программу

process\_choice(4) :- all\_db, readchar(\_), start. % Вывод всей базы данных

process\_choice(\_) :- write("Wrong Choice! Pick the right number"), readchar(\_), nl, start.

run\_test :- % Запуск опроса, если в базе есть хотя бы один вопрос

count\_questions(N), % Подсчёт количества вопросов

N > 0, !,

ask\_all(1).

run\_test :- write("The Database is empty"), readchar(\_), nl.

ask\_all(I) :- % Рекурсивно задаёт вопросы, начиная с номера I

count\_questions(N),

I <= N, !,

testing(I, Q, \_),

write("Question number "), write(I), write(": "),

write(Q), nl,

write("Your answer: "),

readln(UserAnswer),

check\_answer(I, UserAnswer), % Проверка ответа

I1 = I + 1,

ask\_all(I1). % Рекурсивный вызов

ask\_all(\_).

check\_answer(I, UserAnswer) :- % Проверка правильности ответа пользователя

testing(I, \_, Correct),

upper\_lower(Correct,Correct1), % Приведение к нижнему регистру

upper\_lower(UserAnswer,UserAnswer1),

UserAnswer1 = Correct1, !,

write("Correctly!"), readchar(\_), nl, nl.

check\_answer(I, \_) :- % Случай: ответ неверный

testing(I, \_, Correct),

write("Wrong. Right answer: "), write(Correct), readchar(\_), nl, nl.

add\_new\_data :-

write("Ask a question: "),

readln(Question),

write("Enter the correct answer: "),

readln(Answer),

count\_questions(N),

N1 = N + 1,

assertz(testing(N1, Question, Answer)),

write("Question added."), readchar(\_), nl.

count\_questions(N) :-

count\_up(1, 0, N).

count\_up(I, Acc, N) :-

testing(I, \_, \_), !,

Acc1 = Acc + 1,

I1 = I + 1,

count\_up(I1, Acc1, N).

count\_up(\_, Acc, Acc).

all\_db :-

testing(I, Q, A),

write("testing: "), write(I), write(", "), write(Q), write(", "), write(A), nl,fail.

all\_db :-

write("End").

Goal

load\_db, start.

|  |
| --- |
| add\_new\_data :- % Добавление нового вопроса и ответа в базу  write("Ask a question: "),  readln(Question),  write("Enter the correct answer: "),  readln(Answer),  count\_questions(N),  N1 = N + 1,  assertz(testing(N1, Question, Answer)), % Добавляем факт в конец базы  write("Question added."), readchar(\_), nl.  count\_questions(N) :- % Возвращает общее число вопросов в базе  count\_up(1, 0, N).  count\_up(I, Acc, N) :- % Рекурсивно считает количество фактов в базе  testing(I, \_, \_), !,  Acc1 = Acc + 1,  I1 = I + 1,  count\_up(I1, Acc1, N).  count\_up(\_, Acc, Acc).  all\_db :- % Выводит всё содержимое базы данных  testing(I, Q, A),  write("testing: "), write(I), write(", "), write(Q), write(", "), write(A), nl,fail.  all\_db :-  write("End").    Goal  load\_db, start. |

В ходе работы программы были использованы основные механизмы языка Prolog, а именно:

1. Динамическая база данных – во время работы программы в неё можно добавлять новые утверждения. В динамической БД содержаться только факты, но не правила.
2. Рекурсия – основной приём программирования, который позволяет использовать в процессе определения предиката его самого. Рекурсию компилятор Prolog должен оптимизировать таким образом, чтобы не использовать стек в пустую. Для этого испльзуется “хвостовая рекурсия” - частный случай рекурсии, при котором любой рекурсивный вызов является последней операцией перед возвратом из функции.
3. Бэктрекинг – в языке программирования Turbo Prolog это возврат к предыдущему вызову процедуры, откат, который обеспечивает полный обход дерева логического вывода, что является основой для решения интеллектуальных задач.