Выполнила студентка группы 09-832  
 Кузьмина Виктория

**Инструкция по запуску приложения:**

1. Разархивировать папку “*SportForKid\_ES”*

2. В ней зайти в папку “*exe*” и запустить приложение “*SportForKid\_ES.exe”*  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Про интерфейс:**

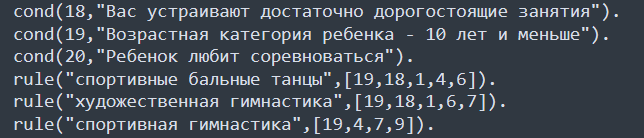
Экспертная система оформлена в виде классического консольного приложения, написанного на языке Prolog, а именно Visual Prolog 10. Среда разработки этой новой 10 версии (в отличие от всех предыдущих) дает возможность создать и графический интерфейс. Но инструменты управления его компонентами достаточно сложные и неочевидные, да и примеров их использования практически нет, поэтому я решила свои силы направить больше на содержание программы, а не на ее оформление.

**Для чего нужна эта экспертная система:**

Данная ЭС помогает родителям выбрать вид спорта для их ребенка, если они сами не могут определиться с подходящим занятием. Пользователю приложения предлагается отвечать на задаваемые вопросы (да или нет), после чего система может предложить подходящий под ответы вид спорта. Если родитель не согласен с предложением, то опрос продолжается, чтобы попробовать подобрать новый вариант.

**Описание базы знаний:**

База знаний находится в файле “*SportForKid\_ES\SportForKid\_catalog.dba”.* Именно этот файл использует приложение. База состоит из двух предикатов:  
- cond хранит порядковый номер свойства и его описание,  
- rule хранит название вида спорта и список номеров соответствующих ему свойств.

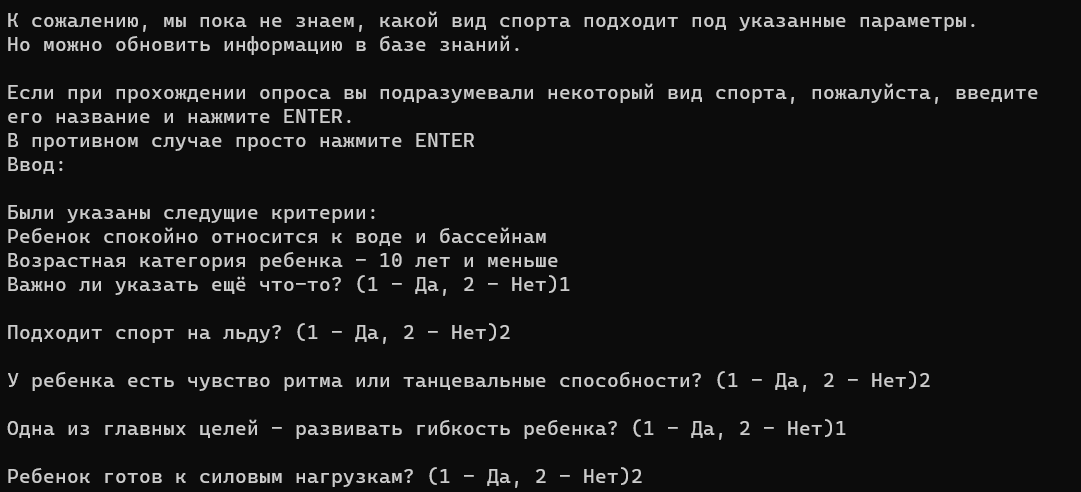


ЭС предлагает определенный вид спорта в качестве результата своей работы, только если на все вопросы о его свойствах пользователь ответил положительно.

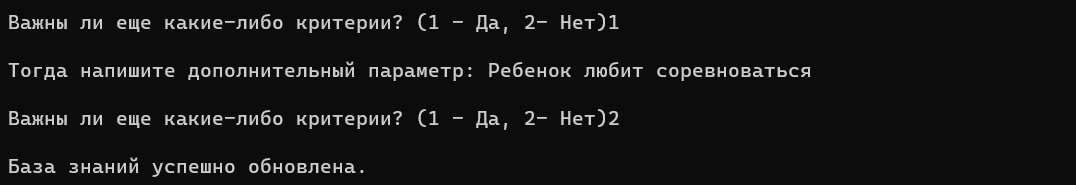
**Про пополнение базы знаний:**

Особенностью системы является то, что если в базе нет информации о виде спорта, который бы подходил под указанные критерии, то у пользователя есть возможность пополнить базу знаний новым видом спорта, который будет соответствовать текущему набору критериев. Для этого система предлагает либо вписать свое название спорта, либо оставить это поле пока без названия (тогда система автоматически вписывает в поле «no\_name»). После чего система предлагает добавить к новому спорту дополнительных свойств: сначала с помощью таких же ответов на вопросы, а потом с помощью ручного ввода.

Пример пополнения базы после опроса:



……………[ряд других вопросов о возможных свойствах]



Таким образом, в процессе использования приложения пользователи автоматически могут пополнять базу знаний новыми видами спорта и новыми свойствами.

**Работа программы на примере:**\*(без объяснений синтаксиса и механизма поиска языка Prolog)

При запуске приложения вызывается предикат run(), который инициализирует консоль, загружает данные в базу (из указанного файла), выводит вводную информацию… Первое существенное действие – вызов предиката choiceSport ().

choiceSport() :-  
        rule(X, L),  
        check(L),  
        stdio::write("\nНам кажется, что для вашего ребенка подойдет следующий вид спорта: ", X),  
        stdio::write("\nА вы согласны с таким выбором? (1 - Да, 2 - Нет)"),  
        read\_true\_char(C),  
        C = '1',  
        !.

Этот предикат отвечает непосредственно за подбор вида спорта. Приложение выдаст пользователю свой вердикт только в том случае, если для вида спорта X будут положительно одобрены все его свойства L.

rule(X, L),  
     check(L),

Перебор видов спорта программа начинает сверху вниз по загруженной базе данных. В текущем файле с данными первым видом спорта являются спортивные бальные танцы.



Затем для списка порядковых номеров свойств L этого вида спорта вызывается предикат check.

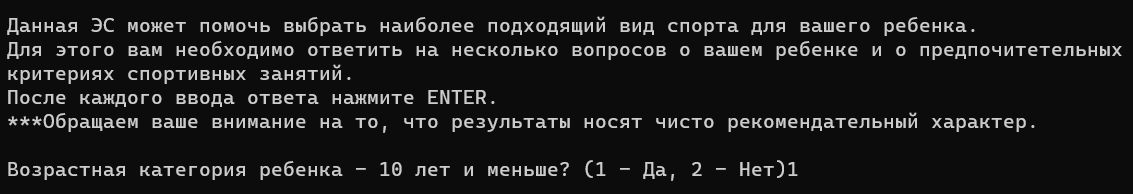
check([H | T]) :-  
        test\_cond(H),  
        check(T).  
    check([]).

Рекурсивно от списка свойств отделяется текущий первый элемент, для которого проверяется условие test\_cond. В нашем примере отделяется номер свойства 19.

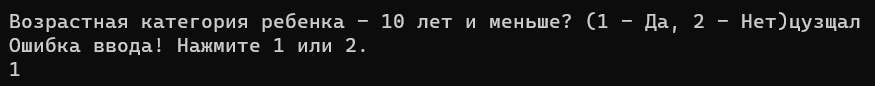
test\_cond(H) :-  
        cond\_is(H, '1'),  
        !. /\* в базе имеется инфа о наличии критерия\*/  
    test\_cond(H) :-  
        cond\_is(H, '2'),  
        !,  
        fail. /\* в базе имеется инфа об отсутствии критерия \*/  
    test\_cond(H) :-  
    /\* в базе нет никакой инфы о критерии, поэтому спрашиваем \*/  
        cond(H, S),  
        stdio::writef("\n%? (1 - Да, 2 - Нет)", S),  
        read\_true\_char(A),  
        assert(cond\_is(H, A)),  
        test\_cond(H).

В нашем случае в базе ответов нет еще никакой информации о свойствах, поэтому первые 2 варианта данного условия не выполняются. 3й вариант по индексу свойства получает его описание S и с помощью консоли выводит вопрос о нем. В нашем примере под номером 19 следующее свойство:



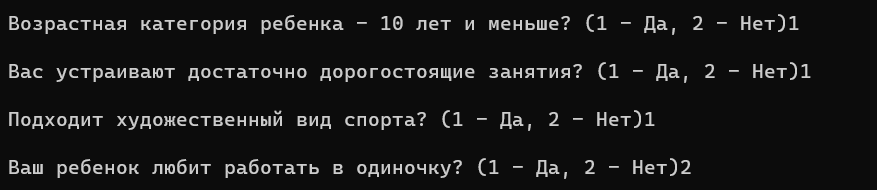


Предикат read\_true\_char(A) отвечает за то, чтобы считанный с консоли введенный символ являлся 1 или 2, в противном случае выведется сообщение об ошибке.



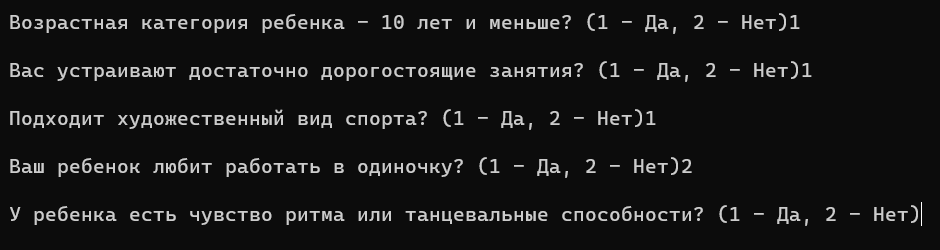
Следующим вызывается предикат assert, он является встроенным и отвечает за пополнение базы знаний текущей программы. В нашем примере в базе появится cond\_is(19, ‘1’), то есть что на вопрос о свойстве 19 пользователь ответил положительно. В конце предиката test\_cond есть рекурсивный вызов, и с помощью оператора отсечения в первом варианте данного предиката программа вернется к условию check(). На этом заканчивается обработка свойства 19.

По точно такой же схеме будут обрабатываться следующие свойства спортивных бальных танцев.

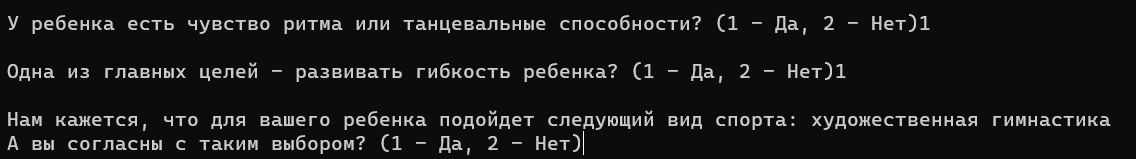


Однако получив отрицательный ответ на вопрос, в предикате test\_cond срабатывает оператор fail, поэтому вся работа с предикатом check для текущего списка свойств обрубается. Тогда choiceSport () берет в качестве спорта следующий по списку – художественная гимнастика.





Первый вопрос задан по свойству с номером 6, потому что про предыдущие критерии (19, 18, 1) в списке уже имеются положительные ответы в текущей базе знаний. При положительных ответах на новые 2 вопроса система выносит вердикт – художественная гимнастика.



**Код программы (язык Visual Prolog 10):**

implement main  
  
domains  
    i = integer.  
    s = string.  
    ch = char.  
    li = i\*.  
  
class facts - knowledge  
    cond : (i, s). /\* критерии \*/  
    rule : (s, li).  
  
class facts - dialog  
    cond\_is : (i, ch).

/\* номер критерия; '1' - есть, '2' - нет\*/  
  
class predicates  
    choiceSport : () nondeterm.  
clauses  
    choiceSport() :-  
        rule(X, L),  
        check(L),  
        stdio::write("\nНам кажется, что для вашего ребенка подойдет следующий вид спорта: ", X),  
        stdio::write("\nА вы согласны с таким выбором? (1 - Да, 2 - Нет)"),  
        read\_true\_char(C),  
        C = '1',  
        !.  
    choiceSport() :-  
        stdio::write("\nК сожалению, мы пока не знаем, какой вид спорта подходит под указанные параметры. \n"),  
        stdio::write("Но можно обновить информацию в базе знаний.\n"),  
        update.  
  
class predicates  
    update : () nondeterm.

/\* добавляет в базу информацию о новом спорте \*/  
clauses  
    update() :-  
        stdio::write("\nЕсли при прохождении опроса вы подразумевали некоторый вид спорта, пожалуйста, введите его название и нажмите ENTER. "),  
        stdio::write("\nВ противном случае просто нажмите ENTER"),  
        stdio::write("\nВвод: "),  
        S = stdio::readLine(),  
        add\_cond(L),  
        S\_len = string::length(S),  
        if S\_len = 0 then  
            assert(rule("no\_name", L))

/\* добавляем вид спорта без названия\*/  
        else  
            assert(rule(S, L))

/\* добавляем вид спорта с введенным названием\*/  
        end if,  
        file::saveUtf8(@"..\SportForKid\_catalog.dba", knowledge),    
        stdio::write("\nБаза знаний успешно обновлена.").  
  
class predicates  
    test\_cond : (i) nondeterm.  
clauses  
    test\_cond(H) :-  
        cond\_is(H, '1'),  
        !. /\* в базе имеется инфа о наличии критерия\*/  
    test\_cond(H) :-  
        cond\_is(H, '2'),  
        !,  
        fail. /\* в базе имеется инфа об отсутствии критерия \*/  
    test\_cond(H) :-  
    /\* в базе нет никакой инфы о критерии, поэтому спрашиваем \*/  
        cond(H, S),  
        stdio::writef("\n%? (1 - Да, 2 - Нет)", S),  
        read\_true\_char(A),  
        assert(cond\_is(H, A)),  
        test\_cond(H).  
  
class predicates  
    check : (li) nondeterm.  
clauses  
    check([H | T]) :-  
        test\_cond(H),  
        check(T).  
    check([]).  
  
class predicates  
    add\_cond : (li) nondeterm anyflow.

/\* возвращает список критериев, имеющихся у нового спорта \*/  
clauses  
    add\_cond(L) :-  
        cond\_is(\_, '1'),  
        !,  
        stdio::write("\nБыли указаны следущие критерии: "),  
        print\_cond(1, [], L1),    
        stdio::write("\nВажно ли указать ещё что-то? (1 - Да, 2 - Нет)"),  
        read\_true\_char(C),  
        read\_cond(C, L1, L).  
    add\_cond(L) :-  
        read\_cond('1', [], L).  
  
class predicates  
    print\_cond : (i, li, li) nondeterm anyflow.

/\* добавляет в список номера критериев с ответами «да» \*/  
clauses  
    print\_cond(H, L, L) :-  
        not(cond(H, \_)),  
        !.  
    print\_cond(H, L, L1) :-  
        cond\_is(H, '1'),  
        !,  
        cond(H, T),  
        H1 = H + 1,  
        stdio::writef("\n%", T),  
        print\_cond(H1, [H | L], L1).  
    print\_cond(H, L, L1) :-  
        H1 = H + 1,  
        print\_cond(H1, L, L1).  
  
class predicates  
    read\_cond : (ch, li, li) anyflow.

/\* добавляет в список номера критериев, о которых еще не спрашивалось \*/  
clauses  
    read\_cond('1', L, L2) :-  
        ex\_cond(1, L, L1, N),  
        new\_cond(N, L1, L2),  
        !.  
    read\_cond(\_, L, L) :-  
        !.  
  
class predicates  
    ex\_cond : (i, li, li, i) nondeterm anyflow.

/\* добавляет в список номера имеющихся в базе критериев\*/  
clauses  
    ex\_cond(N, L, L, N) :-  
        not(cond(N, \_)),  
        !.  
    ex\_cond(N, L, L1, N2) :-  
        cond\_is(N, \_),  
        !,  
        N1 = N + 1,  
        ex\_cond(N1, L, L1, N2).  
    ex\_cond(N, L, L1, N2) :-  
        cond(N, S),  
        stdio::writef("\n%? (1 - Да, 2 - Нет)", S),  
        read\_true\_char(A),  
        wr\_cond(A, N, L, L2),  
        N1 = N + 1,  
        ex\_cond(N1, L2, L1, N2).  
  
class predicates  
    wr\_cond : (ch, i, li, li) nondeterm anyflow.  
clauses  
    wr\_cond('1', N, L, [N | L]) :-  
        !.  
    wr\_cond('2', \_, L, L) :-  
        !.  
  
class predicates  
    new\_cond : (i, li, li) anyflow.

/\* добавляет номера и описания новых критериев в базу знаний \*/  
clauses  
    new\_cond(N, L, L1) :-  
        stdio::write("\nВажны ли еще какие-либо критерии? (1 - Да, 2- Нет)"),  
        read\_true\_char(A),  
        A = '1',  
        !,  
        stdio::write("\nТогда напишите дополнительный параметр: "),  
        S = stdio::readLine(),  
        assert(cond(N, S)),   
        N1 = N + 1,  
        new\_cond(N1, [N | L], L1).  
    new\_cond(\_, L, L).  
  
class predicates  
    read\_true\_char : (ch [out]) determ.

/\* читает символ с консоли, пока он не равен '1' или '2'\*/  
clauses  
    read\_true\_char(C) :-  
        Cstr = stdio::readLine(),  
        C1 = string::subChar(Cstr, 0),  
        test(C1, C).  
  
class predicates  
    test : (ch, ch [out]) determ.  
clauses  
    test(C, C) :-  
        '1' <= C,  
        C <= '2',  
        '1' <= C,  
        !.  
    test(\_, C) :-  
        stdio::write("Ошибка ввода! Нажмите 1 или 2. \n"),  
        Cstr = stdio::readLine(),  
        C1 = string::subChar(Cstr, 0),  
        test(C1, C).  
  
clauses  
    run() :-  
        console::init(stream::unicode),  
        file::consult(@"..\SportForKid\_catalog.dba", knowledge),   
        stdio::write("\nДанная ЭС может помочь выбрать наиболее подходящий вид спорта для вашего ребенка.\n"),  
        stdio::write("Для этого вам необходимо ответить на несколько вопросов о вашем ребенке и о предпочитетельных критериях спортивных занятий.\n"),  
        stdio::write("После каждого ввода ответа нажмите ENTER.\n"),  
        stdio::write("\*\*\*Обращаем ваше внимание на то, что результаты носят чисто рекомендательный характер.\n"),  
        choiceSport, /\* попытка определить \*/  
        retractFactDB(dialog), /\* очищаем текущую информацию \*/  
        retractFactDB(knowledge), /\* удаляем инфу из базы \*/  
        stdio::write("\n\nПопробуем ещё раз? (1 - Да, 2 - Нет)"),  
        read\_true\_char(C),  
        C = '1',  
        !,  
        run.  
    run() :-  
        stdio::write("\nНадеемся, что смогли вам помочь!"),  
        \_ = stdio::readChar().  
  
end implement main  
  
goal  
    mainExe::run(main::run).