Санкт-Петербургский Политехнический Университет Петра Великого
Институт компьютерных наук и технологий
Кафедра компьютерных систем и программных технологий

Лабораторная работа №4. Язык искусственного интеллекта PROLOG

Дисциплина: Интеллектуальные системы

Выполнил студент гр. 13541/3 _____ Д.В. Круминьш
(подпись)

Руководитель _____ Е.Н. Бендерская
(подпись)

”__”_____ 2017 г.

Санкт-Петербург
2017

Содержание

Лабораторная работа №4

Лабораторная работа №4

4.1 Получите начальное представление о синтаксисе и семантике базовых конструкций языка PROLOG, ознакомившись с разделами 1-5 методического пособия

- **Бураков С. В. «Язык логического программирования PROLOG», СПбГУАП, 2003. ДОПОЛНИТЕЛЬНАЯ ЛИТЕРАТУРА, НО ОЧЕНЬ ПОЛЕЗНАЯ (!) Середа С.Н. «Методичка по языку Prolog», Муромский университет. 2003г.**

Ознакомлен.

4.2 Создайте проект в оболочке Visual Prolog 7.3., как это показано в примере

http://wikiru.visual-prolog.com/index.php?title=%D0%9E%D1%81%D0%BD%D0%BE%D0%B2%D1%8B_%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B_Visual_Prolog

Создан.

4.3 Удалите проект, созданный в п. 2 и запустить демонстрационный проект family1 в оболочке Visual Prolog 7.3.

Запущен.

4.4 Постройте генеалогическое дерево для данного примера на основе результатов выполнения программы и исходного кода программы.

По результатам выполнения программы, было построено следующее генеалогическое дерево.

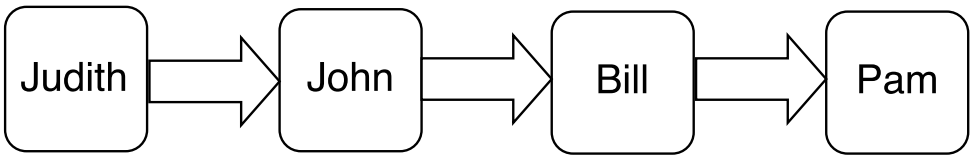

Рис. 4.1: Генеалогическое дерево

4.5 Построить описание онтологии из данного примера на естественном языке.

Необходимо определить степень родства между некоторыми людьми, то есть построить генеалогическое дерево.

Входными данными являются люди, для которых указаны имена и пол, и указаны родственные связи (кто чей родитель).

Определяем, кто чей отец – должен быть родителем и мужчиной.

Определяем, кто чей дедушка – должен быть родителем родителя и мужчиной.

Определяем предка по восходящей линии (ancestor) для конкретного человека (для Pam).

В результате получаем результаты отцовских тестов, тестов на дедушку, и тестов на старших родственников для указанного человека.

4.6 Построить концептуальную карту (семантическую сеть), описывающую данный пример

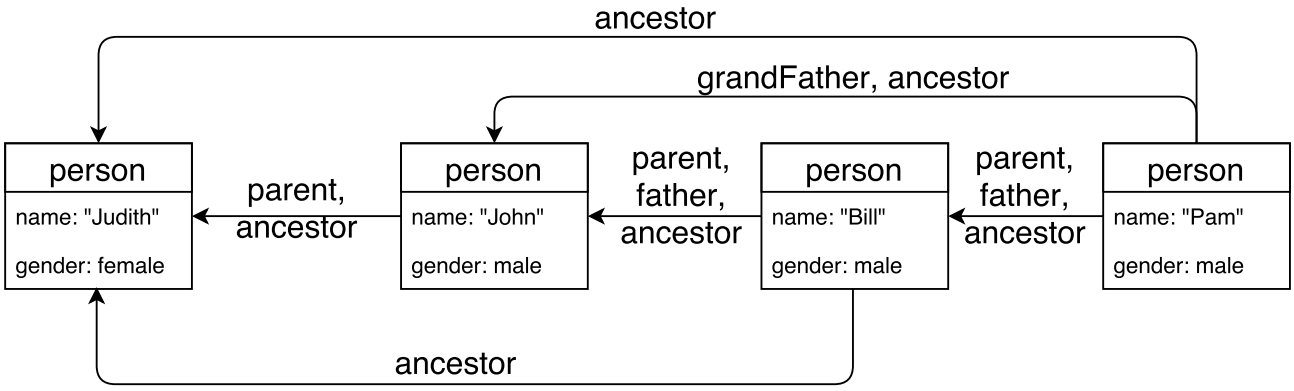

Рис. 4.2: Концептуальная карта

4.7 Создать проекты 1-21 для каждого из примеров в пособии из п.1

4.7.1 Программа 1

```
1 implement main
2     open core
```

```
3
4  class predicates
5      situ : (string Gorod, string Strana) nondeterm anyflow.
6  clauses
7      situ("london", "england").
8      situ("petersburg", "russia").
9      situ("kiev", "ukraine").
10     situ("pekin", "asia").
11     situ("warszawa", "poland").
12     situ("berlin", "europe").
13     situ(X, "europe") :-
14         situ(X, "russia").
15     situ(X, "europe") :-
16         situ(X, "poland").
17
18 clauses
19     run() :-
20         console::init(),
21         situ(X, Y),
22         stdIO::writef("% - %\n", X, Y),
23         fail.
24
25     run().
26
27 end implement main
28
29 goal
30     console::run(main::run).
```

Листинг 4.1: main.pro

```
1  C:\Users\Tom\Documents\Visual Prolog Projects\program1\Exe>"C:\
       ↪ Users\Tom\Documents\Visual Prolog Projects\program1\Exe\
       ↪ program1.exe"
2  london - england
3  petersburg - russia
4  kiev - ukraine
5  pekin - asia
6  warszawa - poland
7  berlin - europe
8  petersburg - europe
```

```
9 warszawa – europe
```

Листинг 4.2: Результат выполнения программы 1

Программа успешно вывела все данные, относящиеся к предикату situ.

4.7.2 Программа 2

```
1  implement main
2      open core
3
4  domains
5      collector = symbol.
6      title = symbol.
7      author = symbol.
8      publisher = symbol.
9      year = integer.
10     personal_library = book(title, author, publication).
11     publication = publication(publisher, year).
12
13 class predicates
14     collection : (collector [out], personal_library [out]).
15 clauses
16     collection("Иванов", book("Война и мир", "Лев Толстой",
       ↪ publication("Просвещение", 1990))).
17
18 clauses
19     run() :-
20         console::init(),
21         collection(X, Y),
22         stdIO::writef("% – %\n", X, Y),
23         fail.
24
25     run().
26
27 end implement main
28
29 goal
30     console::run(main::run).
```

Листинг 4.3: main.pro

```
C:\Users\Tom\Documents\Visual Prolog Projects\program2\Exe>"C:\
    ↪ Users\Tom\Documents\Visual Prolog Projects\program2\Exe\
    ↪ program2.exe"
Иванов – book("Война и мир", "Лев Толстой", publication("Просвещение"
    ↪ ,1990))
```

Листинг 4.4: Результат выполнения программы 2

В данной программе было показано использование составных объектов. Такие объекты особенно полезны при описании иерархических структур.

4.7.3 Программа 3

```
implement main
    open core

domains
    person = symbol.

class predicates
    otec : (person, person) nondeterm anyflow.
    man : (person) nondeterm.
    brat : (person [out], person [out]) nondeterm.

clauses
    man(X) :-
        otec(X, _).

    brat(X, Y) :-
        otec(Z, Y),
        otec(Z, X),
        man(X),
        X <> Y.

    otec("ivan", "igor").
    otec("ivan", "sidor").
    otec("sidor", "lisa").

clauses
    run() :-
        console::init(),
```

```
29          brat(X, Y),
30          stdIO::writef("% is brother of %\n", X, Y),
31          fail.
32
33      run().
34
35  end implement main
36
37  goal
38      console::run(main::run).
```

Листинг 4.5: main.pro

```
1  C:\Users\Tom\Documents\Visual Prolog Projects\program3\Exe>"C:\
       ↪ Users\Tom\Documents\Visual Prolog Projects\program3\Exe\
       ↪ program3.exe"
2  sidor is brother of igor
```

Листинг 4.6: Результат выполнения программы 3

В данной программе, на основании фактов и использовании предикатов решается примитивная задача о семейных отношениях.

4.7.4 Программа 4

```
1  implement main
2      open core
3
4  class predicates
5      add : (integer, integer).
6      fadd : (real, real).
7      maximum : (real, real, real [out]) nondeterm.
8
9  clauses
10     add(X, Y) :-
11         Z = X + Y,
12         stdIO::write("Sum= ", Z),
13         stdIO::nl.
14     fadd(X, Y) :-
15         Z = X + Y,
16         stdIO::write("FSum= ", Z),
17         stdIO::nl.
18     maximum(X, X, X).
```

```
19      maximum(X, Y, X) :-
20          X > Y.
21      maximum(X, Y, Y) :-
22          X < Y.
23
24  clauses
25      run() :-
26          console::init(),
27          add(2, 3),
28          fadd(3.5, 2.56),
29          maximum(2, -4, Z),
30          stdIO::write(Z),
31          fail.
32      run().
33
34  end implement main
35
36  goal
37      console::run(main::run).
```

Листинг 4.7: main.pro

```
1  C:\Users\Tom\Documents\Visual Prolog Projects\program4\Exe>"C:\
       ↪ Users\Tom\Documents\Visual Prolog Projects\program4\Exe\
       ↪ program4.exe"
2  Sum= 5
3  FSum= 6.06
4  2
```

Листинг 4.8: Результат выполнения программы 4

В языке ПРОЛОГ используется ряд встроенных функций для вычисления арифметических выражений. Для описания любых операций арифметики можно также использовать собственные предикаты.

В данной задаче, с помощью предикатов вычисляется сумма целых, вещественных чисел, а также максимум из двух чисел.

4.7.5 Программа 5

```
1  implement main
2      open core
3
4  domains
```

```
5       nazvanie = symbol.
6       stolica = symbol.
7       naselenie = integer.
8       territoria = real.
9
10  class predicates
11      strana : (nazvanie [out], naselenie [out], territoria [out],
        ↪ stolica [out]) multi.
12  clauses
13      strana("kitai", 1200, 9597000, "pekin").
14      strana("belgia", 10, 30000, "brussel").
15      strana("peru", 20, 1285000, "lima").
16
17  clauses
18      run() :-
19          console::init(),
20          strana(X, _, Y, _),
21          Y > 1000000,
22          stdIO::writef("% - %\n", X, Y),
23          fail.
24
25      run().
26
27  end implement main
28
29  goal
30      console::run(main::run).
```

Листинг 4.9: main.pro

```
1  C:\Users\Tom\Documents\Visual Prolog Projects\program5\Exe>"C:\
       ↪ Users\Tom\Documents\Visual Prolog Projects\program5\Exe\
       ↪ program5.exe"
2  kitai - 9597000
3  peru - 1285000
```

Листинг 4.10: Результат выполнения программы 5

В данной программе используются переменные в запросе. В запросе задается один из параметров(площадь больше 1000000), и как результат выводится название страны и её площадь.

4.7.6 Программа 6

```
class main
    open core

predicates
    run : core::runnable.
    hello : ().

end class main
```

Листинг 4.11: main.cl

```
implement main
    open core

clauses
    run() :-
        succeed.

clauses
    hello() :-
        stdIO::write("hello").

end implement main

goal
    console::runUtf8(main::hello).
```

Листинг 4.12: main.pro

```
C:\Users\Tom\Documents\Visual Prolog Projects\program6\Exe>"C:\
    ↪ Users\Tom\Documents\Visual Prolog Projects\program6\Exe\
    ↪ program6.exe"
hello
```

Листинг 4.13: Результат выполнения программы 6

В файле main.cl был добавлен собственный предикат **hello**. Далее он был использован для вывода в консоль сообщения **hello**.

4.7.7 Программа 7

```
implement main
    open core
```

```
3
4  class predicates
5      gorod : (symbol [out]) multi.
6  clauses
7      gorod("Москва").
8      gorod("Минск").
9      gorod("Киев").
10     gorod("Омск").
11     run() :-
12         gorod(X),
13         stdIO::write(X),
14         stdIO::nl(),
15         fail.
16
17     run().
18
19 end implement main
20
21 goal
22     console::init(),
23     stdIO::write("Это города:"),
24     stdIO::nl(),
25     main::run().
```

Листинг 4.14: main.pro

```
1  C:\Users\Tom\Documents\Visual Prolog Projects\program7\Exe>"C:\
       ↪ Users\Tom\Documents\Visual Prolog Projects\program7\Exe\
       ↪ program7.exe"Это
2   города:МоскваМинскКиевОмск
```

Листинг 4.15: Результат выполнения программы 7

Используя предикат fail, было вызывано искусственное не успешное завершение поиска, что позволяет получить все возможные решения задачи.

4.7.8 Программа 8

```
1  implement main
2      open core
3
4  domains
5      person = symbol.
```

```
6
7  class predicates
8      deti : (person [out]) multi.
9      make_cut : (person) determ.
10
11 clauses
12     deti("Петя").
13     deti("Вася").
14     deti("Олег").
15     deti("Маша").
16     deti("Оля").
17     deti("Наташа").
18     run() :-
19         deti(X),
20         stdIO::write(X),
21         stdIO::nl(),
22         make_cut(X),
23         !.
24
25     run().
26
27     make_cut(X) :-
28         X = "Олег".
29
30 end implement main
31
32 goal
33     console::init(),
34     stdIO::write("Это мальчики:"),
35     stdIO::nl(),
36     main::run().
```

Листинг 4.16: main.pro

```
1  C:\Users\Tom\Documents\Visual Prolog Projects\program8\Exe>"C:\
       ↪ Users\Tom\Documents\Visual Prolog Projects\program8\Exe\
       ↪ program8.exe"Это
2   мальчики:ПетяВасяОлег
```

Листинг 4.17: Результат выполнения программы 8

В данной программе используется предикат отсечения cut. Он позволяет получить доступ только к части данных, устраняя дальнейшие поисковые действия. На консоль выводятся только

имена мальчиков, так как после имени Олег мы прекращаем поиск.

4.7.9 Программа 9

```
implement main
    open core

class predicates
    buy_car : (symbol [out], symbol [out]) determ.
    car : (symbol [out], symbol [out], integer [out]) multi.
    color : (symbol, symbol) determ.

clauses
    car("москвич", "синий", 12000).
    car("жигули", "зеленый", 26000).
    car("вольво", "синий", 24000).
    car("волга", "синий", 20000).
    car("ауди", "зеленый", 20000).
    color("синий", "темный").
    color("зеленый", "светлый").
    buy_car(Model, Color) :-
        car(Model, Color, Price),
        color(Color, "светлый"),
        !,
        Price < 25000.
    run() :-
        console::init(),
        buy_car(X, Y),
        stdIO::writef("% - %\n", X, Y),
        fail.
    run().

end implement main

goal
    console::runUtf8(main::run).
```

Листинг 4.18: main.pro

```
C:\Users\Tom\Documents\Visual Prolog Projects\program9\Exe>"C:\
    ↪ Users\Tom\Documents\Visual Prolog Projects\program9\Exe\
    ↪ program9.exe"
```

```
2
3 C:\Users\Tom\Documents\Visual Prolog Projects\program9\Exe>pause
4 Для продолжения нажмите любую клавишу . . .
```

Листинг 4.19: Результат выполнения программы 9

Эта программа не найдет ни одного решения, поскольку после дорогих зеленых «жигулей» поиск заканчивается, и более дешевые «ауди» не будут найдены.

4.7.10 Программа 10

```
1  implement main
2      open core
3
4  domains
5      number = integer.
6
7  class predicates
8      write_number : (number) nondeterm.
9  clauses
10     write_number(10).
11
12     write_number(N) :-
13         N < 10,
14         stdIO::write(N),
15         stdIO::nl(),
16         write_number(N + 1).
17
18     run() :-
19         console::init(),
20         stdIO::write("Это числа"),
21         stdIO::nl(),
22         main::write_number(1),
23         fail.
24
25     run().
26
27 end implement main
28
29 goal
30     console::runUtf8(main::run).
```

Листинг 4.20: main.pro

```
C:\Users\Tom\Documents\Visual Prolog Projects\program10\Exe>"C:\
    ↪ Users\Tom\Documents\Visual Prolog Projects\program10\Exe\
    ↪ program10.exe"
Это числа
1
2
3
4
5
6
7
8
9
```

Листинг 4.21: Результат выполнения программы 10

Данная программа показывает пример использования рекурсии в ПРОЛОГЕ. В разделе clauses даны два описания предиката write_number. Если в процессе решения первое описание не успешно, то используется второе описание. Программа печатает цифры от 1 до 9.

4.7.11 Программа 11

```
implement main
    open core

class predicates
    summa : (integer, integer [out]).
clauses
    summa(X, Y) :-
        X < 10,
        Y = X,
        !.

    summa(X, Y) :-
        X1 = X div 10,
        summa(X1, Y1),
        Y = Y1 + X mod 10.

    run() :-
```

```
18          console::init(),
19          summa(2045, Y),
20          stdIO::write(Y),
21          stdIO::nl().
22
23  end implement main
24
25  goal
26      console::runUtf8(main::run).
```

Листинг 4.22: main.pro

```
1  C:\Users\Tom\Documents\Visual Prolog Projects\program11\Exe>"C:\
       ↪ Users\Tom\Documents\Visual Prolog Projects\program11\Exe\
       ↪ program11.exe"
2  11
```

Листинг 4.23: Результат выполнения программы 11

Данная программа печатает сумму всех цифр введенного числа(2045). Благодаря использованию предиката ! в описании правила позволяет избежать переполнения стека.

4.7.12 Программа 12

```
1  implement main
2      open core
3
4  domains
5      loc = right; middle; left.
6
7  class predicates
8      hanoi : (integer).
9      move : (integer, loc, loc, loc).
10     inform : (loc, loc).
11
12 clauses
13     hanoi(N) :-
14         move(N, left, middle, right).
15
16     move(1, A, _, C) :-
17         inform(A, C),
18         !.
19
```

```
20      move(N, A, B, C) :-
21          move(N - 1, A, C, B),
22          inform(A, C),
23          move(N - 1, B, A, C).
24
25      inform(Loc1, Loc2) :-
26          stdIO::write("Диск с ", Loc1, " на ", Loc2),
27          stdIO::nl().
28
29      run() :-
30          console::init(),
31          hanoi(5).
32
33  end implement main
34
35  goal
36      console::runUtf8(main::run).
```

Листинг 4.24: main.pro

```
1  C:\Users\Tom\Documents\Visual Prolog Projects\program12\Exe>"C:\
       ↪ Users\Tom\Documents\Visual Prolog Projects\program12\Exe\
       ↪ program12.exe"
2  Диск с left на right
3  Диск с left на middle
4  Диск с right на middle
5  Диск с left на right
6  Диск с middle на left
7  Диск с middle на right
8  Диск с left на right
9  Диск с left на middle
10 Диск с right на middle
11 Диск с right на left
12 Диск с middle на left
13 Диск с right на middle
14 Диск с left на right
15 Диск с left на middle
16 Диск с right на middle
17 Диск с left на right
18 Диск с middle на left
19 Диск с middle на right
```

```
Диск с left на right
Диск с middle на left
Диск с right на middle
Диск с right на left
Диск с middle на left
Диск с middle на right
Диск с left на right
Диск с left на middle
Диск с right на middle
Диск с left на right
Диск с middle на left
Диск с middle на right
Диск с left на right
```

Листинг 4.25: Результат выполнения программы 12

В данной программе решается задача «Ханойская башня». Требуется переместить диски с первого на третий стержень за некоторую последовательность ходов, каждый из которых заключается в перекладывания верхнего диска с одного из стержней на другой стержень. При этом больший диск никогда нельзя ставить на меньший диск.

Данная задача была решена с использованием рекурсии.

4.7.13 Программа 13

```
implement main
    open core

domains
    dog_list = symbol*.

class predicates
    dogs : (dog_list [out]).
    print_list : (dog_list).

clauses
    dogs(["лайка", "борзая", "дог", "болонка"]).

    print_list([]).

    print_list([X | Y]) :-
        stdIO::write(X),
        stdIO::nl(),
```

```
19          print_list(Y).
20
21      run() :-
22          console::init(),
23          dogs(X),
24          stdIO::write(X),
25          stdIO::nl(),
26          print_list(X).
27
28  end implement main
29
30  goal
31      console::runUtf8(main::run).
```

Листинг 4.26: main.pro

```
1  C:\Users\Tom\Documents\Visual Prolog Projects\program13\Exe>"C:\
       ↪ Users\Tom\Documents\Visual Prolog Projects\program13\Exe\
       ↪ program13.exe"
2  ["лайка "борзая "дог "болонка"]
3  лайка
4  борзая
5  дог
6  болонка
```

Листинг 4.27: Результат выполнения программы 13

В данной программе, с помощью списков были описаны породы собак. Операция разделения списка на голову и хвост обозначается с помощью вертикальной черты: [Head|Tail] С помощью этой операции можно реализовывать рекурсивную обработку списка.

4.7.14 Программа 14

```
1  implement main
2      open core
3
4  domains
5      dog_list = symbol*.
6
7  class predicates
8      find_it : (symbol, dog_list) nondeterm.
9  clauses
10     find_it(X, [X | _]).
```

```
11
12      find_it(X, [_ | Y]) :-
13          find_it(X, Y).
14
15      run() :-
16          console::init(),
17          find_it("болонка", ["лайка", "дог"]),
18          stdIO::write("да"),
19          fail.
20
21      run().
22
23  end implement main
24
25  goal
26      console::runUtf8(main::run).
```

Листинг 4.28: main.pro

```
1  C:\Users\Tom\Documents\Visual Prolog Projects\program14\Exe>"C:\
       ↪ Users\Tom\Documents\Visual Prolog Projects\program14\Exe\
       ↪ program14.exe"
2
3  C:\Users\Tom\Documents\Visual Prolog Projects\program14\Exe>pause
4  Для продолжения нажмите любую клавишу . . .
```

Листинг 4.29: Результат выполнения программы 14

Первое правило описывает ситуацию, когда искомый элемент X совпадает с головой списка.

Второе правило используется при неуспехе первого правила и описывает новый вызов первого правила, но уже с усеченным списком, в котором нет первого элемента и т.д. Если в списке нет элементов (пустой список), то второе правило оказывается неуспешным.

Программа не напечатает Yes, поскольку болонки нет в списке собак.

4.7.15 Программа 15

```
1  implement main
2      open core
3
4  domains
5      spisok = integer*.
6
7  class predicates
```

```
8       summa_sp : (spisok, integer [out]).
9   clauses
10      summa_sp([], 0).
11
12      summa_sp([H | T], S) :-
13          summa_sp(T, S1),
14          S = H + S1.
15
16      run() :-
17          console::init(),
18          summa_sp([2, 0, 4, 5], Sum),
19          stdIO::write(Sum),
20          stdIO::nl().
21
22  end implement main
23
24  goal
25      console::runUtf8(main::run).
```

Листинг 4.30: main.pro

```
1   C:\Users\Tom\Documents\Visual Prolog Projects\program15\Exe>"C:\
        ↪ Users\Tom\Documents\Visual Prolog Projects\program15\Exe\
        ↪ program15.exe"
2   20
```

Листинг 4.31: Результат выполнения программы 15

В данной программе происходит подсчет суммы всех элементов списка чисел. Числа, указанные в списке: 2, 0, 4, 5. Их сумма равна 11, что совпадает с результатом, выведенным на консоль.

4.7.16 Программа 16

```
1   implement main
2       open core
3
4   domains
5       loc = east; west.
6       state = state(loc, loc, loc, loc).
7       path = state*.
8
9   class predicates
10      go : (state, state).
```

```
11      path : (state, state, path, path [out]) determ.
12      move : (state, state [out]) nondeterm.
13      opposite : (loc, loc) determ anyflow.
14      unsafe : (state) nondeterm.
15      member : (state, path) nondeterm.
16      write_path : (path) determ.
17      write_move : (state, state) determ.
18
19  clauses
20      go(S, G) :-
21          path(S, G, [S], L),
22          stdIO::write("Решение:"),
23          stdIO::nl(),
24          write_path(L),
25          fail.
26
27      go(_, _).
28
29      path(S, G, L, L1) :-
30          move(S, S1),
31          not(unsafe(S1)),
32          not(member(S1, L)),
33          path(S1, G, [S1 | L], L1),
34          !.
35
36      path(G, G, T, T) :-
37          !.
38
39      move(state(X, X, G, C), state(Y, Y, G, C)) :-
40          opposite(X, Y).
41
42      move(state(X, W, X, C), state(Y, W, Y, C)) :-
43          opposite(X, Y).
44
45      move(state(X, W, G, X), state(Y, W, G, Y)) :-
46          opposite(X, Y).
47
48      move(state(X, W, G, C), state(Y, W, G, C)) :-
49          opposite(X, Y).
50
```

```
51      opposite(east, west).
52
53      opposite(west, east) :-
54          !.
55
56      unsafe(state(F, X, X, _)) :-
57          opposite(F, X).
58
59      unsafe(state(F, _, X, X)) :-
60          opposite(F, X).
61
62      member(X, [X | _]).
63
64      member(X, [_ | L]) :-
65          member(X, L).
66
67      write_path([H1, H2 | T]) :-
68          !,
69          write_move(H1, H2),
70          write_path([H2 | T]).
71
72      write_path([]).
73
74      write_move(state(X, W, G, C), state(Y, W, G, C)) :-
75          !,
76          stdIO::write("Мужик пересекает реку с ", X, " на ", Y),
77          stdIO::nl().
78
79      write_move(state(X, X, G, C), state(Y, Y, G, C)) :-
80          !,
81          stdIO::write("Мужик везет волка с ", X, " берега на ", Y),
82          stdIO::nl().
83
84      write_move(state(X, W, X, C), state(Y, W, Y, C)) :-
85          !,
86          stdIO::write("Мужик везет козу с ", X, " берега на ", Y),
87          stdIO::nl().
88
89      write_move(state(X, W, G, X), state(Y, W, G, Y)) :-
90          !,
```

```
91          stdIO::write("Мужик везет капусту с ", X, " берега на ", Y),
92          stdIO::nl().
93
94  clauses
95      run() :-
96          console::init(),
97          go(state(east, east, east, east), state(west, west, west,
        ↪ west)).
98
99  end implement main
100
101 goal
102     console::runUtf8(main::run).
```

Листинг 4.32: main.pro

```
1  C:\Users\Tom\Documents\Visual Prolog Projects\program16\Exe>"C:\
       ↪ Users\Tom\Documents\Visual Prolog Projects\program16\Exe\
       ↪ program16.exe"
2  Решение:
3  Мужик везет козу с west берега на east
4  Мужик пересекает реку с east на west
5  Мужик везет капусту с west берега на east
6  Мужик везет козу с east берега на west
7  Мужик везет волка с west берега на east
8  Мужик пересекает реку с east на west
9  Мужик везет козу с west берега на east
```

Листинг 4.33: Результат выполнения программы 16

Данная программа показывает, как используются списки и механизм рекурсии при решении известной задачи о мужике, волке, козе и капусте.

4.7.17 Программа 17

```
1  implement main
2      open core
3
4  class predicates
5      name : (symbol) determ.
6      name : (symbol [out]) multi.
7      mesto : (symbol) determ.
8      mesto : (symbol [out]) multi.
```

```
9       prizer : (symbol, symbol) nondeterm.
10      solution : (symbol [out], symbol [out], symbol [out], symbol [
        ↪ out], symbol [out], symbol [out]) determ.
11
12  clauses
13      name("Alex").
14      name("Pier").
15      name("Nike").
16      mesto("first").
17      mesto("second").
18      mesto("third").
19      prizer(X, Y) :-
20          name(X),
21          mesto(Y),
22          X = "Pier",
23          not(Y = "second"),
24          not(Y = "third")
25          or
26          name(X),
27          mesto(Y),
28          X = "Nike",
29          not(Y = "third")
30          or
31          name(X),
32          mesto(Y),
33          not(X = "Pier"),
34          not(X = "Nike").
35
36      solution(X1, Y1, X2, Y2, X3, Y3) :-
37          name(X1),
38          name(X2),
39          name(X3),
40          mesto(Y1),
41          mesto(Y2),
42          mesto(Y3),
43          prizer(X1, Y1),
44          prizer(X2, Y2),
45          prizer(X2, Y3),
46          Y1 <> Y2,
47          Y2 <> Y3,
```

```
48          Y1 <> Y3,
49          X1 <> X2,
50          X2 <> X3,
51          X1 <> X3,
52          !.
53
54      run() :-
55          console::init(),
56          solution(X1, Y1, X2, Y2, X3, Y3),
57          stdIO::writef("% - %\n% - %\n% - %\n", X1, Y1, X2, Y2, X3,
    ↪  Y3),
58          fail.
59
60      run().
61
62  end implement main
63
64  goal
65      console::runUtf8(main::run).
```

Листинг 4.34: main.pro

```
1  C:\Users\Tom\Documents\Visual Prolog Projects\program17\Exe>"C:\
     ↪ Users\Tom\Documents\Visual Prolog Projects\program17\Exe\
     ↪ program17.exe"
2  Alex - third
3  Nike - first
4  Pier - second
```

Листинг 4.35: Результат выполнения программы 17

С помощью ПРОЛОГА была успешно решена следующая задача:

«В велосипедных гонках три первых места заняли Алеша, Петя и Коля. Какое место занял каждый из них, если Петя занял не второе и не третье место, а Коля – не третье?»

4.7.18 Программа 18

```
1  implement main
2      open core
3
4  domains
5      name = symbol.
```

```
6
7  class predicates
8      student : (name) determ.
9      student : (name [out]) multi.
10     gorod : (name) determ.
11     gorod : (name [out]) multi.
12     velo : (name, name) determ.
13     fact : (name, name) determ anyflow.
14     fact1 : (name, name) determ anyflow.
15     rodom : (name, name) nondeterm.
16     rodom : (name [out], name) nondeterm.
17     rodom_penza : (name) nondeterm.
18
19 clauses
20     student(X) :-
21         X = "Сергей"
22         or
23         X = "Борис"
24         or
25         X = "Виктор"
26         or
27         X = "Григорий"
28         or
29         X = "Леонид".
30
31     gorod(Y) :-
32         Y = "Пенза"
33         or
34         Y = "Львов"
35         or
36         Y = "Москва"
37         or
38         Y = "Харьков"
39         or
40         Y = "Рига".
41
42     fact("Сергей", "Рига").
43     fact("Борис", "Пенза").
44     fact("Виктор", "Москва").
45     fact("Григорий", "Харьков").
```

```
46      velo(X, Y) :-
47          student(X),
48          gorod(Y),
49          fact(X, Y),
50          !
51          or
52          student(X),
53          gorod(Y),
54          not(fact(X, _)),
55          not(fact(_, Y)).
56      fact1("Борис", "Рига").
57      fact1("Виктор", "Львов").
58
59      rodom_penza(X) :-
60          student(X),
61          not(fact1(X, _)),
62          gorod(U),
63          not(U = "Пенза"),
64          velo(X, U),
65          rodom("Леонид", U).
66
67      rodom(X, Z) :-
68          student(X),
69          gorod(Z),
70          fact1(X, Z),
71          !
72          or
73          student(X),
74          not(X = "Леонид"),
75          Z = "Пенза",
76          rodom_penza(X),
77          !
78          or
79          student(X),
80          gorod(Z),
81          not(fact1(_, Z)),
82          X = "Леонид",
83          not(Z = "Пенза"),
84          student(K),
85          not(fact1(K, _)),
```

```
86              velo(K, Z)
87              or
88              student(X),
89              not(X = "Леонид"),
90              gorod(Z),
91              not(Z = "Пенза"),
92              not(fact1(_, Z)),
93              not(fact1(X, _)),
94              gorod(Y),
95              not(Y = Z),
96              velo(X, Y),
97              not(rodom("Леонид", Z)),
98              not(rodom("Леонид", Y)).
99      run() :-
100             console::init(),
101             rodom(X, "Москва"),
102             stdIO::writef("% родом из Москвы", X),
103             fail.
104
105     run().
106
107 end implement main
108
109 goal
110     console::runUtf8(main::run).
```

Листинг 4.36: main.pro

```
1 C:\Users\Tom\Documents\Visual Prolog Projects\program18\Exe>"C:\
    ↪ Users\Tom\Documents\Visual Prolog Projects\program18\Exe\
    ↪ program18.exe"
2 Сергей родом из Москвы
```

Листинг 4.37: Результат выполнения программы 18

Была решена ещё одна логическая задача:

«Пятеро студентов едут на велосипедах. Их зовут Сергей, Борис, Леонид, Григорий и Виктор. Велосипеды сделаны в пяти городах: Риге, Пензе, Львове, Харькове и Москве. Каждый из студентов родился в одном из этих городов, но ни один из студентов не едет на велосипеде, сделанном на его родине. Сергей едет на велосипеде, сделанном в Риге. Борис родом из Риги, у него велосипед из Пензы. У Виктора велосипед из Москвы. У Григория велосипед из Харькова. Виктор родом из Львова. Уроженец Пензы едет на велосипеде, сделанном на родине

Леонида. Кто из студентов родом из Москвы?»

4.7.19 Программа 19

```
implement main
    open core

domains
    s = symbol.

class predicates
    st_A : (s [out]) multi.
    st_D : (s [out]) multi.
    st_B : (s [out]) multi.
    st_V : (s [out]) multi.
    st_G : (s [out]) multi.
    ogr1 : (s, s, s, s, s) determ.
    ogr2 : (s, s, s, s, s) determ.
    spisok : (s [out], s [out], s [out], s [out], s [out])
  ↪ nondeterm.
    norm1 : (s, s, s, s, s) determ.
    norm2 : (s, s, s, s, s) determ.
    norm3 : (s, s, s, s, s) determ.
    norm4 : (s, s, s, s, s) determ.
clauses
    st_A(A) :-
        A = "Андрей"
        or
        A = "нет".
    st_D(D) :-
        D = "Дмитрий"
        or
        D = "нет".
    st_B(B) :-
        B = "Борис"
        or
        B = "нет".
    st_V(V) :-
        V = "Виктор"
        or
```

```
36          V = "нет".
37      st_G(G) :-
38          G = "Григорий"
39          or
40          G = "нет".
41      ogr1("Андрей", _, _, "нет", _).
42      ogr1("нет", _, _, "Виктор", _).
43      ogr2(_, "Дмитрий", _, _, "нет").
44      ogr2(_, "нет", _, _, _).
45      norm1("Андрей", "Дмитрий", "нет", _, _).
46      norm2("Андрей", "нет", "Борис", "нет", _).
47      norm3(_, "Дмитрий", "нет", "нет", _).
48      norm4(_, "нет", "нет", "Виктор", "Григорий").
49      spisok(A, D, B, V, G) :-
50          st_A(A),
51          st_D(D),
52          st_B(B),
53          st_V(V),
54          st_G(G),
55          norm1(A, D, B, V, G),
56          ogr1(A, D, B, V, G),
57          ogr2(A, D, B, V, G)
58          or
59          st_A(A),
60          st_D(D),
61          st_B(B),
62          st_V(V),
63          st_G(G),
64          norm2(A, D, B, V, G),
65          ogr1(A, D, B, V, G),
66          ogr2(A, D, B, V, G)
67          or
68          st_A(A),
69          st_D(D),
70          st_B(B),
71          st_V(V),
72          st_G(G),
73          norm3(A, D, B, V, G),
74          ogr1(A, D, B, V, G),
75          ogr2(A, D, B, V, G)
```

```
76          or
77          st_A(A),
78          st_D(D),
79          st_B(B),
80          st_V(V),
81          st_G(G),
82          norm4(A, D, B, V, G),
83          ogr1(A, D, B, V, G),
84          ogr2(A, D, B, V, G)
85          or
86          st_A(A),
87          st_D(D),
88          st_B(B),
89          st_V(V),
90          st_G(G),
91          not(norm1(A, D, B, V, G)),
92          not(norm2(A, D, B, V, G)),
93          not(norm3(A, D, B, V, G)),
94          not(norm4(A, D, B, V, G)),
95          ogr1(A, D, B, V, G),
96          ogr2(A, D, B, V, G).
97
98  clauses
99      run() :-
100         console::init(),
101         spisok(A, D, B, V, G),
102         stdIO::writef("%%%%%\n", A, D, B, V, G),
103         fail.
104
105     run().
106
107 end implement main
108
109 goal
110     console::runUtf8(main::run).
```

Листинг 4.38: main.pro

```
1  C:\Users\Tom\Documents\Visual Prolog Projects\program19\Exe>"C:\
   ↪ Users\Tom\Documents\Visual Prolog Projects\program19\Exe\
   ↪ program19.exe"
```

```
2  Андрей Дмитрий нет нет нет
3  Андрей нет Борис нет Григорий
4  Андрей нет Борис нет нет
5  Андрей Дмитрий нет нет нет
6  нет нет нет Виктор Григорий
7  Андрей Дмитрий Борис нет нет
8  Андрей нет нет нет Григорий
9  Андрей нет нет нет нет
10 нет Дмитрий Борис Виктор нет
11 нет Дмитрий нет Виктор нет
12 нет нет Борис Виктор Григорий
13 нет нет Борис Виктор нет
14 нет нет нет Виктор нет
```

Листинг 4.39: Результат выполнения программы 19

Была решена ещё одна логическая задача:

«Пять студентов должны посещать лекции всю неделю, но по определенным ими установленным правилам, а именно:

1. Если пришли Андрей и Дмитрий, то Бориса быть не должно, но если Дмитрий не пришел, то Борис должен быть, а Виктор быть не должен.
2. Если Виктор пришел, то Андрея быть не должно и наоборот.
3. Если Дмитрий пришел, то Григория быть не должно.
4. Если Бориса нет, то Дмитрий должен быть, но если нет также и Виктора, а если Виктор есть, Дмитрия быть не должно, но должен быть Григорий.
5. Каждый день студенты должны приходить в разных сочетаниях.

Какие это сочетания?»

4.7.20 Программа 20

```
1  implement main
2      open core
3
4  domains
5      name = symbol.
6      rost = integer.
7      ves = integer.
8
```

```
9  class facts
10     dplayer : (name, rost, ves).
11
12 class predicates
13     player : (name [out], rost [out], ves [out]) multi.
14     assert_database : ().
15
16 clauses
17     player("Михайлов", 180, 87).
18     player("Петров", 187, 93).
19     player("Харламов", 177, 80).
20     assert_database() :-
21         player(N, R, V),
22         assertz(dplayer(N, R, V)),
23         fail.
24     assert_database().
25 clauses
26     run() :-
27         console::init(),
28         assert_database(),
29         dplayer(N, R, V),
30         R > 180,
31         stdIO::writef("%% см %% кг\n", N, R, V),
32         fail.
33
34     run().
35
36 end implement main
37
38 goal
39     console::runUtf8(main::run).
```

Листинг 4.40: main.pro

```
1  C:\Users\Tom\Documents\Visual Prolog Projects\program20\Exe>"C:\
   ↪ Users\Tom\Documents\Visual Prolog Projects\program20\Exe\
   ↪ program20.exe"
2  Петров 187 см 93 кг
```

Листинг 4.41: Результат выполнения программы 20

Факты, описанные в разделе clauses, можно рассматривать, как статическую базу данных (БД). Эти факты являются частью кода программы и не могут быть оперативно изменены.

В данной программе осуществляется поиск человека, ростом выше 180 см.

4.7.21 Программа 21

```
implement main
    open core

class facts
    xpositive : (string, string).
    xnegative : (string, string).

class predicates
    expertiza : ().
    vopros : (string, string) determ.
    fish_is : (string [out]) nondeterm.
    positive : (string, string) determ.
    negative : (string, string) determ.
    remember : (string, string, string) determ.
    clear_facts : ().

clauses
    expertiza() :-
        fish_is(X),
        !,
        stdIO::write("ваша рыба это ", X, " "),
        stdIO::nl,
        clear_facts().
    expertiza() :-
        stdIO::write("это неизвестная рыба!"),
        stdIO::nl,
        clear_facts().
    vopros(X, Y) :-
        stdIO::write("вопрос – ", X, " ", Y, "? данет(/)"),
        R = stdIO::readLine(),
        remember(X, Y, R).
    positive(X, Y) :-
        xpositive(X, Y),
        !.
    positive(X, Y) :-
        not(negative(X, Y)),
```

```
37          !,
38          vopros(X, Y).
39      negative(X, Y) :-
40          xnegative(X, Y),
41          !.
42      remember(X, Y, "да") :-
43          assertz(xpositive(X, Y)).
44      remember(X, Y, "нет") :-
45          assertz(xnegative(X, Y)),
46          fail.
47      clear_facts() :-
48          retract(xpositive(_, _)),
49          fail.
50      clear_facts() :-
51          retract(xnegative(_, _)),
52          fail.
53      clear_facts().
54      fish_is("сом") :-
55          positive("у рыбы", "вес > 40 кг").
56      fish_is("сом") :-
57          positive("у рыбы", "вес < 40 кг"),
58          positive("у рыбы", "есть усы").
59      fish_is("щука") :-
60          positive("у рыбы", "вес < 20 кг"),
61          positive("у рыбы", "длинное узкое тело").
62      fish_is("окунь") :-
63          positive("у рыбы", "вес < 20 кг"),
64          positive("у рыбы", "широкое тело"),
65          positive("у рыбы", "темные полосы").
66      fish_is("плотва") :-
67          positive("у рыбы", "вес < 20 кг"),
68          positive("у рыбы", "широкое тело"),
69          positive("у рыбы", "серебристая чешуя").
70  clauses
71      run() :-
72          console::init(),
73          expertiza().
74
75  end implement main
76
```

```
77 goal
78     console::runUtf8(main::run).
```

Листинг 4.42: main.pro

```
1 C:\Users\Tom\Documents\Visual Prolog Projects\program21\Exe>"C:\
    ↪ Users\Tom\Documents\Visual Prolog Projects\program21\Exe\
    ↪ program21.exe"
2 вопрос – у рыбы вес > 40 кг? (да/нет)нет
3 вопрос – у рыбы вес < 40 кг? (да/нет)да
4 вопрос – у рыбы есть усы? (да/нет)нет
5 вопрос – у рыбы вес < 20 кг? (да/нет)да
6 вопрос – у рыбы длинное узкое тело? (да/нет)нет
7 вопрос – у рыбы широкое тело? (да/нет)да
8 вопрос – у рыбы темные полосы? (да/нет)да
9 ваша рыба это окунь
```

Листинг 4.43: Результат выполнения программы 21

В данной программе реализован пример простой экспертной системы, которая решает задачу определения вида экземпляра пойманной рыбы.

Программа реализует заданное дерево поиска решения. Ответы на заданные вопросы позволяют продвигаться по ветвям этого дерева к одному из вариантов решения.

4.8 Выполнить одно из индивидуальных заданий

Вариант - 11.

Пять пионеров Алик, Боря, Витя, Лена и Даша приехали в лагерь из 5 разных городов: Харькова, Умани, Полтавы, Славянска и Краматорска. Есть 4 высказывания:

1. Если Алик не из Умани, то Боря из Краматорска.
2. Или Боря, или Витя приехали из Харькова.
3. Если Витя не из Славянска, то Лена приехала из Харькова.
4. Или Даша приехала из Умани, или Лена из Краматорска.

Кто откуда приехал?

```
1 implement main
2     open core
3
4 class predicates
5     pioner : (symbol) determ.
6     pioner : (symbol [out]) multi.
```

```
7       gorod : (symbol) determ.
8       gorod : (symbol [out]) multi.
9       fact : (symbol, symbol) nondeterm.
10      rodina : (symbol, symbol) nondeterm anyflow.
11      sequence : (symbol [out], symbol [out], symbol [out], symbol [
      ↪ out], symbol [out], symbol [out], symbol [out], symbol [out
      ↪ ], symbol [out],
12          symbol [out]) determ.
13
14  clauses
15      pioner("Alik").
16      pioner("Boris").
17      pioner("Vitya").
18      pioner("Lena").
19      pioner("Dasha").
20      gorod("Harkov").
21      gorod("Umani").
22      gorod("Poltava").
23      gorod("Slavyansk").
24      gorod("Kramatorsk").
25      fact(X, Y) :-
26          pioner(X),
27          gorod(Y),
28          X = "Alik",
29          not(Y = "Umani")
30          or
31          X = "Boris",
32          Y = "Kramatorsk"
33          or
34          pioner(X),
35          X = "Boris"
36          or
37          X = "Vitya",
38          gorod(Y),
39          Y = "Harkov"
40          or
41          pioner(X),
42          gorod(Y),
43          X = "Vitya",
44          not(Y = "Slavyansk")
```

```
45          or
46          X = "Dasha",
47          Y = "Harkov"
48          or
49          pioner(X),
50          gorod(Y),
51          X = "Dasha",
52          Y = "Umani"
53          or
54          X = "Lena",
55          Y = "Kramatorsk".
56
57  clauses
58      rodina(X, Y) :-
59          pioner(X),
60          gorod(Y),
61          fact(X, Y).
62      sequence(X, XX, Y, YY, Z, ZZ, T, TT, M, MM) :-
63          rodina(X, XX),
64          rodina(Y, YY),
65          rodina(Z, ZZ),
66          rodina(T, TT),
67          rodina(M, MM),
68          not(X = Y),
69          not(X = Z),
70          not(X = T),
71          not(X = M),
72          not(Y = Z),
73          not(Y = T),
74          not(Y = M),
75          not(Z = T),
76          not(Z = M),
77          not(T = M),
78          not(XX = YY),
79          not(XX = ZZ),
80          not(XX = TT),
81          not(XX = MM),
82          not(YY = ZZ),
83          not(YY = TT),
84          not(YY = MM),
```

```
85          not(ZZ = TT),
86          not(ZZ = MM),
87          not(TT = MM),
88          !.
89      run() :-
90          console::init(),
91          sequence(X, XX, Y, YY, Z, ZZ, T, TT, M, MM),
92          stdIO::writef("====Result:\n% - %\n% - %\n% - %\n% - %\n%
    ↪ - %\n", X, XX, Y, YY, Z, ZZ, T, TT, M, MM),
93          fail.
94
95      run().
96
97  end implement main
98
99  goal
100     console::runUtf8(main::run).
```

Листинг 4.44: main.pro

```
1  C:\Users\Tom\Documents\Visual Prolog Projects\varian11\Exe>"C:\
       ↪ Users\Tom\Documents\Visual Prolog Projects\varian11\Exe\
       ↪ varian11.exe"
2  ====Result:
3  Alik - Harkov
4  Boris - Slavyansk
5  Vitya - Poltava
6  Lena - Kramatorsk
7  Dasha - Umani
```

Листинг 4.45: Результат выполнения variant11

Были написаны следующие предикаты:

- pioner (Y)
- gorod(X)
- fact(X,Y)
- rodina(X,Y,)

Первые два используются для перечисления имеющихся в задаче объектов.

С помощью предиката fact(x,y) записываются имеющиеся условия задачи.

С помощью предиката rodina(X,Y) обозначается логическую связь между пионерами, городами и фактами: для пионера X город Y является родиной, если X – пионер, Y – город, и выполняется предикат fact(X,Y).

Также с помощью предиката sequence были указаны правила на ограничения задачи: пионеры из разных городов, это разные пионеры и у каждого есть город, из которого он приехал.

4.9 Изучить 1-2 лабы по методичке Седана С.Н. (доп литература). Согласно своему варианту решить задачу с помощью PROLOG

Вариант - 3.

Лабиринт представляет собой систему комнат, соединенных между собой переходами. В лабиринте имеется вход и выход, а также комната с золотым кладом. Кроме того, имеются комнаты, запрещенные для посещений: комната монстров и комната разбойников.

1. Найди путь в лабиринте от входа до входа, не посещая дважды одной и той же комнаты;
2. Найти путь с посещением золотой комнаты;
3. Найти путь, избегающий запрещенных к посещению комнат.

Решение задачи:

Структура лабиринта была описана схематичным образом, также был написан алгоритм обхода в ширину и соответствующие запросы, согласно заданию.

```
1  edge(in, out).
2  edge(in, gold).
3  edge(in, monster).
4  edge(in, robber).
5
6  edge(gold, out).
7  edge(gold, monster).
8  edge(gold, robber).
9
10 edge(monster, out).
11 edge(monster, gold).
12 edge(monster, robber).
13
14 edge(robber, out).
15 edge(robber, gold).
16 edge(robber, monster).
17
18 findWay(From, To, _, [edge(From, To)]):-
19   edge(From, To).
```

```
20 findWay(From, To, VisitedNodes, [(From, X)|TailPath]):-
21   edge(From, X),
22   not(member(X, VisitedNodes)),
23   findWay(X, To, [From|VisitedNodes], TailPath).
```

Листинг 4.46: variant3.pl

В структуре описано из какой комнаты в какую можно попасть, так из комнаты входа, можно сразу попасть в выход. С помощью findWay ищутся варианты маршрутов. Первым параметром является начальная точка, вторым – конечная, третьим – та комната, которую необходимо посетить.

```
1 findWay(in, out, [], Path). %1 – путь от входа до выхода
2 findWay(in, out, [], Path), member((_, gold), Path). %2 –
    ↪ путь с посещением золотой комнаты
3 findWay(in, out, [], Path), not(member((_, robber), Path)), not(
    ↪ member((_, monster), Path)). %3 –
    ↪ путь без посещения монстров и грабителей.
```

Листинг 4.47: Запросы

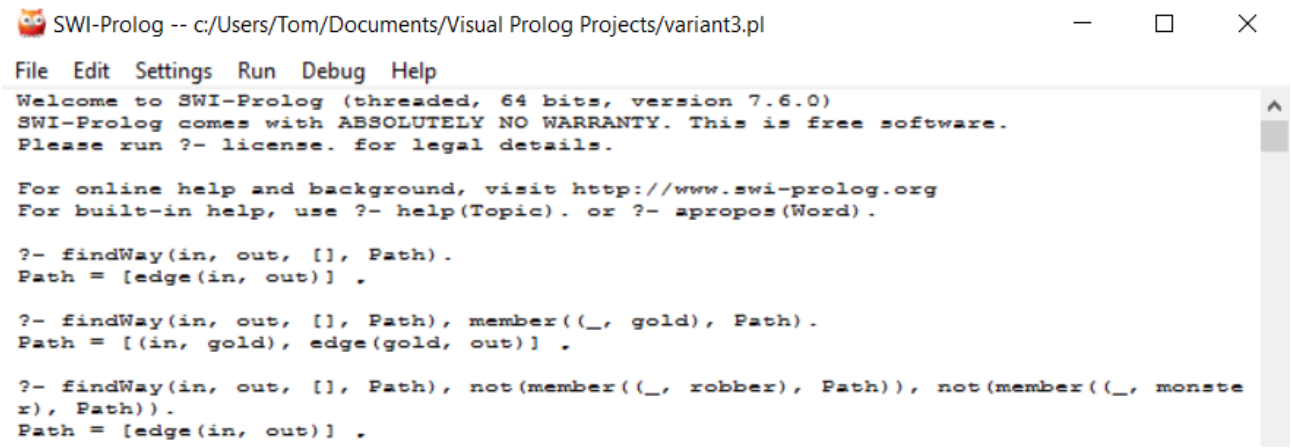

Рис. 4.3: Результат выполнения

Запросы возвращают наиболее простые проходы, из-за неоднозначной структуры лабиринта.

4.10 Вывод

Как итог данной работы, я впервые познакомился с логическим языком ПРОЛОГ. Были изучены базовые операции, синтаксис и т.д. Как итог, мне трудно представить проект(кроме ЭС) на данном языке, не связанный с научной деятельностью.

1. В чем Плюсы и минусы языка Prolog?

Данный язык является логическим, из-за этого имеются трудности при решении каких-либо

комплексных задач, вычислительных операциях, в следствии чего наблюдается недостаток инвестиций и внимания. Свою сильные стороны язык проявляет в:

- доказательстве теорем и вывода решений в задачах;
- создания пакетов символьной обработки при решении уравнений, дифференцировании, интегрировании и т. д.;
- разработки упрощенных версий систем ИИ.

2. Какие еще языки используются для разработки ИИ, приведите примеры (НЕ МЕНЕЕ 2-х) проектов, языков и краткое описание проектов. (Альтернативы PROLOG)

Для создания ИИ можно использовать любой язык.

Проект	Язык	Описание
OpenWorm	NeuroML	Компьютерная модель червя на клеточном уровне
DeepMind	?	Наиболее перспективный(на мой взгляд) проект по ИИ

3. Решаема ли проблема комбинаторного взрыва, пути решения?

Не решаема, кроме перебора всех решений.

4. Корректно ли по-вашему в принципе разработка языка ИИ? Что он должен из себя представлять?

Да, корректна, это ускорит еще сильнее создание полноценного ИИ. Как минимум он должен содержать в себе, уже реализованные базовые функции при разработке ИИ.

5. Можно ли разработать ИИ не понимая, как он работает, должны ли мы понимать, как он работает, думает, рассуждает?

Точно так-же как и разрабатывают квантовый компьютер. Точка где человек, в деталях понимает как работает его творение уже пройдена. Стоит задуматься о контроле таких творений, в особенности ИИ.

Литература

[1] Бураков С.В. Язык логического программирования PROLOG [Электронный ресурс]. — СПбГУАП, 2003.

[2] Середа С.Н. Методичка по языку Prolog [Электронный ресурс]. — Муромский университет, 2003.