Министерство науки и высшего образования

Российской Федерации

Федеральное государственное бюджетное образовательное

учреждение высшего образования

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра Теоретической и Прикладной Информатики

Лабораторная работа №1 по предмету

«**Интеллектуальные системы»**

Факультет: прикладной математики и информатики

Группа: ПМИ-12

Бригада: 10

Студенты: Швадченко А. В.  
 Попов С. Н.  
 Кораблёв К. В.

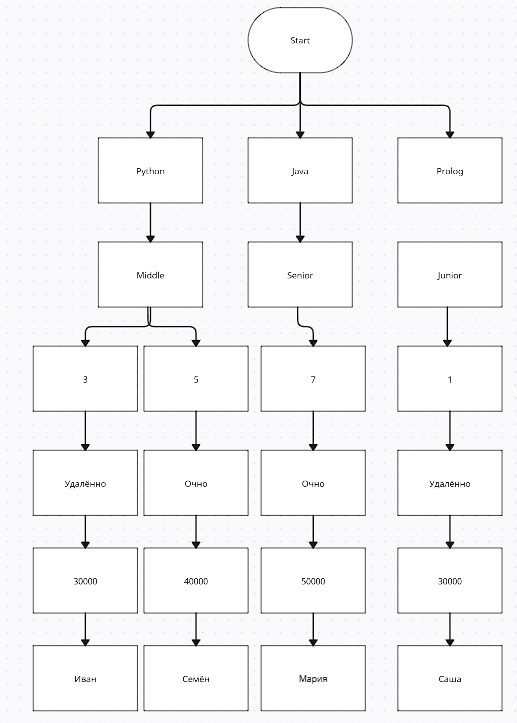
Преподаватели: Дворецкая В. К.

Новосибирск, 2025

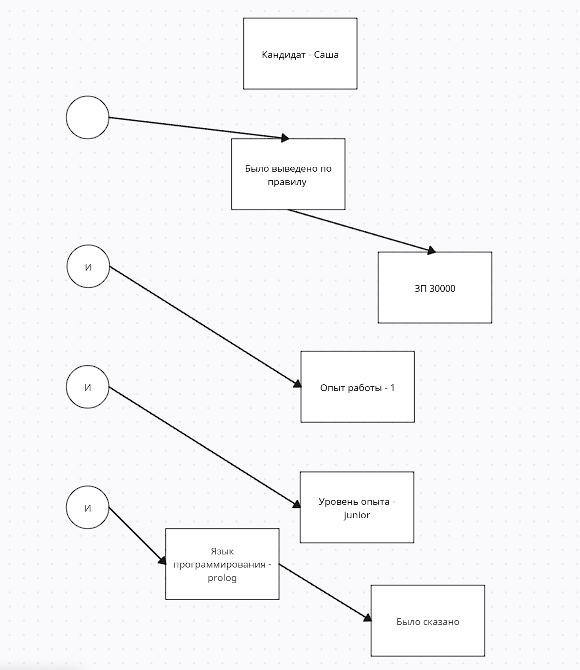
1. Предметная область:  
   Программа реализует экспертную систему для подбора кандидатов на вакантные позиции в IT-сфере. Основная идея заключается в том, чтобы на основе набора характеристик кандидатов принимать решения о соответствии требованиям вакансии. Вот основные аспекты предметной области:

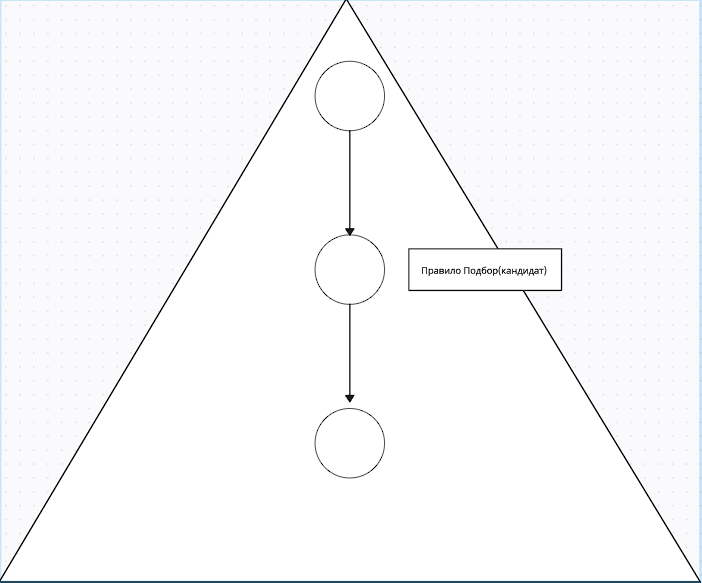
* **Характеристики кандидатов:** Каждый кандидат описывается набором атрибутов, таких как:  
  + **Язык программирования:** например, Python, Java, Prolog.
  + **Уровень опыта:** категории вроде junior, middle, senior.
  + **Опыт работы:** количество лет практического опыта.
  + **Формат работы:** условия, например, удалённый или очное.
  + **Зарплата:** ожидаемая зарплата (например, 30000, 40000 и т.д.).
* **База знаний:** Информация о кандидатах хранится во внешнем файле candidates.pl, а возможные значения для каждого атрибута – в файле options\_db.pl. Это позволяет динамически расширять базу и подстраиваться под изменяющиеся требования.
* **Механизм вывода заключений:** Программа использует механизм доказательства (предикат prove) для логического вывода, проверяя, соответствует ли кандидат заданным условиям. При этом используется метод обратного вызова (backward chaining) для анализа фактов и правил.
* **Интерактивное взаимодействие:** Система запрашивает у пользователя значения для различных атрибутов с помощью предиката ask\_attribute. Это позволяет получать дополнительную информацию или подтверждение от пользователя в процессе подбора кандидата. Взаимодействие сопровождается возможностью задать вопросы типа «почему?» или «как?», что выводит объяснения принятых решений.
* **Добавление новых кандидатов:** Программа предусматривает возможность ввода нового кандидата через меню, после чего его данные сохраняются в базу знаний и в соответствующий файл. Это облегчает расширение системы новыми записями.
* **Пользовательский интерфейс:** Реализовано простое консольное меню, через которое пользователь может выбрать добавление нового кандидата, запуск эксперта для подбора кандидата или завершение работы программы.

1. Роли:
   1. Эксперт - Попов Семён
   2. Инженер по знаниям - Кораблёв Кирилл
   3. Программист - Швадченко Артём
2. Полное дерево всех решений:



1. Дерево “как?” и “почему?”



Цель пользователя кандидат со знанием python, уровнем опыта middle

Текущая цель: Семён

пространство поиска

Ответы пользователя python, middle, 5, очно, 40000

1. Использованные правила:

Правило для кандидата ИВАН:  
 кандидат(иван) :- язык\_программирования(python), уровень\_опыта(middle), опыт\_работы(3), формат\_работы(удаленка), зп(30000).

Если кандидат владеет языком программирования Python, имеет уровень опыта middle, 3 года опыта работы, предпочитает формат работы "удаленка" и ожидает зарплату 30000, то выбирается кандидат ИВАН.

Правило для кандидата МАРИЯ:  
 кандидат(мария) :- язык\_программирования(java), уровень\_опыта(senior), опыт\_работы(7), формат\_работы(очно), зп(50000).

Если кандидат владеет языком программирования Java, имеет уровень опыта senior, 7 лет опыта работы, предпочитает очный формат работы и ожидает зарплату 50000, то выбирается кандидат МАРИЯ.

Правило для кандидата СЕМЕН:  
 кандидат(семен) :- язык\_программирования(python), уровень\_опыта(middle), опыт\_работы(5), формат\_работы(очно), зп(40000).

Если кандидат владеет языком программирования Python, имеет уровень опыта middle, 5 лет опыта работы, предпочитает очный формат работы и ожидает зарплату 40000, то выбирается кандидат СЕМЕН.

Правило для кандидата САША:

кандидат(саша) :- язык\_программирования(prolog), уровень\_опыта(junior), опыт\_работы(1), формат\_работы(удаленка), зп(30000).

Если кандидат владеет языком программирования Prolog, имеет уровень опыта junior, 1 год опыта работы, предпочитает формат работы "удаленка" и ожидает зарплату 30000, то выбирается кандидат САША.

1. Код программы:

Файл lr1.pl  
:- working\_directory(\_, 'C:/users/artem/documents/prolog').

:- dynamic is\_ans/2.

:- dynamic кандидат/1.

:- dynamic answer/2.

:- multifile язык\_программирования/1.

:- multifile уровень\_опыта/1.

:- multifile опыт\_работы/1.

:- multifile формат\_работы/1.

:- multifile зп/1.

:- multifile prove/3.

:- multifile ask\_ans/2.

:- multifile ans\_check/2.

:- multifile askable/1.

:- multifile option\_list/2.

:- multifile update\_option/2.

:- multifile update\_option\_if\_new/2.

:- multifile ensure\_expected/4.

:- multifile ask\_attribute/2.

:- multifile print\_options/2.

:- multifile my\_asserta/1.

:- multifile story/0.

:- multifile conj/2.

:- multifile wrap/3.

:- multifile read\_languages/1.

:- multifile read\_languages\_aux/2.

:- multifile add\_candidate/0.

:- multifile build\_clause\_body/6.

:- multifile write\_candidate\_to\_file/6.

:- multifile write\_list\_lines/3.

:- multifile clear\_history/0.

:- multifile run\_expert/0.

:- multifile menu/0.

:- multifile candidate\_menu/0.

:- op(500, xfy, <==).

:- op(500, xfy, /).

% Объявляем askable-предикаты как динамические

:- dynamic язык\_программирования/1.

:- dynamic уровень\_опыта/1.

:- dynamic опыт\_работы/1.

:- dynamic формат\_работы/1.

:- dynamic зп/1.

% Разрешаем разобщённые определения для этих предикатов

:- discontiguous язык\_программирования/1.

:- discontiguous уровень\_опыта/1.

:- discontiguous опыт\_работы/1.

:- discontiguous формат\_работы/1.

:- discontiguous зп/1.

% "Заглушки" для askable-предикатов, чтобы они существовали и не выдавали ошибку

язык\_программирования(\_) :- fail.

уровень\_опыта(\_) :- fail.

опыт\_работы(\_) :- fail.

формат\_работы(\_) :- fail.

зп(\_) :- fail.

% Загрузка базы знаний кандидатов и базы опций из отдельных файлов

:- consult('candidates.pl').

:- consult('options\_db.pl').

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% МЕХАНИЗМ ОБЪЯСНЕНИЯ И ДОКАЗАТЕЛЬСТВА

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

prove(G, G, \_) :-

predicate\_property(G, built\_in),

call(G), !.

prove(G, true, \_) :-

G = true, !.

prove((G, Gs), (P, Ps), Q) :- !,

prove(G, P, Q),

prove(Gs, Ps, Q), !.

prove(G, (G <== Ps), Q) :-

\+ askable(G),

clause(G, Gs),

prove(Gs, Ps, [G <== Gs | Q]),

my\_asserta(is\_ans(G, true)).

prove(G, G, \_) :-

is\_ans(G, false), !, fail.

prove(G, G, \_) :-

is\_ans(G, true), !, true.

prove(G, G, Q) :-

ask\_ans(G, Q), !.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% МЕХАНИЗМ ЗАПРОСА С ОТЛОЖЕННЫМ СОХРАНЕНИЕМ ОТВЕТОВ

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

ask\_ans(G, Q) :-

askable(G),

G =.. [Pred, \_Expected],

( answer(Pred, \_) -> true ;

ask\_attribute(G, UserAnswer),

assertz(answer(Pred, UserAnswer))

),

ans\_check(G, Q).

ans\_check(G, Q) :-

G =.. [Pred, Expected],

answer(Pred, A),

( A == почему ->

( Q = [Hypothesis <== Conditions | \_] ->

format("Выдвигается гипотеза: ~w\n", [Hypothesis]),

format("при комплексе условий: ~w\n", [Conditions])

; true ),

retractall(answer(Pred, \_)),

ask\_attribute(G, NewAnswer),

assertz(answer(Pred, NewAnswer)),

ans\_check(G, Q)

; A == как ->

short\_story,

retractall(answer(Pred, \_)),

ask\_attribute(G, NewAnswer),

assertz(answer(Pred, NewAnswer)),

ans\_check(G, Q)

; A == Expected ->

true

;

fail

).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Предикаты для askable-атрибутов

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

askable(t(язык\_программирования/\_)).

askable(t(уровень\_опыта/\_)).

askable(t(опыт\_работы/\_)).

askable(t(формат\_работы/\_)).

askable(t(зп/\_)).

askable(язык\_программирования(\_)).

askable(уровень\_опыта(\_)).

askable(опыт\_работы(\_)).

askable(формат\_работы(\_)).

askable(зп(\_)).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Вспомогательные предикаты для меню

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

option\_list(Attr, Options) :-

findall(Opt, option(Attr, Opt), Options).

update\_option(Attr, Option) :-

open('options\_db.pl', append, Stream),

format(Stream, 'option(~q, ~q).~n', [Attr, Option]),

close(Stream).

update\_option\_if\_new(Attr, Value) :-

( option(Attr, Value) ->

true

;

update\_option(Attr, Value),

assertz(option(Attr, Value))

).

ensure\_expected(Attr, Expected, OptionsTemp, OptionsFinal) :-

( member(Expected, OptionsTemp) ->

OptionsFinal = OptionsTemp

;

update\_option\_if\_new(Attr, Expected),

OptionsFinal = [Expected|OptionsTemp]

).

ask\_attribute(G, Answer) :-

G =.. [Pred, Expected],

( option\_list(Pred, BaseOptions) -> true ; BaseOptions = [Expected] ),

OptionsFiltered = BaseOptions,

ensure\_expected(Pred, Expected, OptionsFiltered, OptionsNoWhy),

append(OptionsNoWhy, [почему, как], Options),

format("Выберите значение для ~w:\n", [Pred]),

print\_options(Options, 1),

format("Ваш выбор (введите номер): "),

read(Choice),

( integer(Choice),

nth1(Choice, Options, Selected) ->

Answer = Selected

; writeln('Неверный выбор, попробуйте снова.'),

ask\_attribute(G, Answer)

).

print\_options([], \_).

print\_options([Opt|Rest], N) :-

format("~w. ~w\n", [N, Opt]),

N1 is N + 1,

print\_options(Rest, N1).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Сохранение информации о доказательствах и история

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

my\_asserta(is\_ans(G, V)) :-

retract(is\_ans(G, V)),

assert(is\_ans(G, V)), !.

my\_asserta(is\_ans(G, V)) :-

assert(is\_ans(G, V)), !.

% Определение списка атрибутов для истории

attributes([язык\_программирования, уровень\_опыта, опыт\_работы, формат\_работы, зп]).

% Модифицированный предикат story/0 для вывода истории в заданном формате

story :-

writeln("История:"),

attributes(Attrs),

forall(member(Pred, Attrs),

( answer(Pred, A) ->

findall(Opt, option(Pred, Opt), Options),

forall(member(Opt, Options),

( (Opt == A -> V = true ; V = false),

G =.. [Pred, Opt],

format("G = ~w V = ~w\n", [G, V])

)

)

; true % Пропускаем атрибуты, которые не были запрошены

)

).

% Предикат для вывода укороченной истории на данный момент

short\_story :-

findall(P, (answer(P, A), A \== почему, A \== как), AskedAttrs),

list\_to\_set(AskedAttrs, UniqueAttrs),

( UniqueAttrs = [] ->

writeln("Нет запрошенных атрибутов на данный момент.")

;

writeln("Укороченная история на данный момент:"),

forall(member(P, UniqueAttrs),

( answer(P, A),

option\_list(P, Options),

forall(member(Opt, Options),

( (Opt == A -> V = true ; V = false),

G =.. [P, Opt],

format("G = ~w V = ~w\n", [G, V])

)

)

)

)

).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Вспомогательные предикаты для работы с кандидатами

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

conj([], true).

conj([G], G).

conj([G|Gs], (G, Rest)) :-

conj(Gs, Rest).

wrap(P, Arg, Goal) :-

Goal =.. [P, Arg].

read\_languages(Languages) :-

write('Введите язык программирования: '), read(Lang),

read\_languages\_aux([Lang], Languages).

read\_languages\_aux(Acc, Languages) :-

writeln('Добавить еще язык? (1 - Да, 2 - Перейти к следующему пункту)'),

read(Choice),

( Choice = 1 ->

write('Введите язык программирования: '), read(NextLang),

append(Acc, [NextLang], NewAcc),

read\_languages\_aux(NewAcc, Languages)

; Choice = 2 ->

Languages = Acc

; writeln('Неверный выбор, считаем, что больше языков не добавляется.'),

Languages = Acc

).

add\_candidate :-

write('Введите имя кандидата (например, ivan): '), read(Name),

read\_languages(Languages),

forall(member(Lang, Languages), update\_option\_if\_new(язык\_программирования, Lang)),

write('Введите уровень опыта (например, junior, middle, senior): '), read(Level),

update\_option\_if\_new(уровень\_опыта, Level),

write('Введите опыт работы (число лет, например, 5): '), read(WorkExp),

update\_option\_if\_new(опыт\_работы, WorkExp),

write('Введите формат работы (удаленка или очно): '), read(WorkFormat),

update\_option\_if\_new(формат\_работы, WorkFormat),

write('Введите зп (например, 30000 или 50000): '), read(Salary),

update\_option\_if\_new(зп, Salary),

build\_clause\_body(Languages, Level, WorkExp, WorkFormat, Salary, Body),

assertz((кандидат(Name) :- Body)),

writeln('Новый кандидат добавлен в базу знаний.'),

write\_candidate\_to\_file(Name, Languages, Level, WorkExp, WorkFormat, Salary).

build\_clause\_body(Languages, Level, WorkExp, WorkFormat, Salary, Body) :-

maplist(wrap(язык\_программирования), Languages, LanguageGoals),

conj(LanguageGoals, LanguagesConj),

wrap(уровень\_опыта, Level, LevelGoal),

wrap(опыт\_работы, WorkExp, WorkExpGoal),

wrap(формат\_работы, WorkFormat, FormatGoal),

wrap(зп, Salary, SalaryGoal),

Body = (LanguagesConj, LevelGoal, WorkExpGoal, FormatGoal, SalaryGoal).

write\_candidate\_to\_file(Name, Languages, Level, WorkExp, WorkFormat, Salary) :-

open('candidates.pl', append, Stream),

format(Stream, '\n%% Новый кандидат\n', []),

format(Stream, 'кандидат(~w) :-\n', [Name]),

write\_list\_lines(Stream, Languages, 'язык\_программирования'),

format(Stream, ' ~w(~w),\n', ['уровень\_опыта', Level]),

format(Stream, ' ~w(~w),\n', ['опыт\_работы', WorkExp]),

format(Stream, ' ~w(~w),\n', ['формат\_работы', WorkFormat]),

format(Stream, ' ~w(~w).\n', ['зп', Salary]),

close(Stream).

write\_list\_lines(\_, [], \_).

write\_list\_lines(Stream, [X], Predicate) :-

format(Stream, ' ~w(~w),\n', [Predicate, X]).

write\_list\_lines(Stream, [X|Xs], Predicate) :-

format(Stream, ' ~w(~w),\n', [Predicate, X]),

write\_list\_lines(Stream, Xs, Predicate).

clear\_history :-

retractall(is\_ans(\_,\_)),

retractall(answer(\_,\_)).

% Модифицированный run\_expert с выбором после поиска

run\_expert :-

clear\_history,

( prove(подбор(Кандидат), Proof, [])

-> ( format("Результат подбора кандидата: ~w\n", [Кандидат]),

writeln("Доказательство:"),

writeln(Proof)

)

; writeln('Не найден кандидат, удовлетворяющий условиям.')

),

writeln('1. Показать историю'),

writeln('2. Выход в меню'),

read(Choice),

( Choice = 1 ->

story

; Choice = 2 ->

true

; writeln('Неверный выбор, возвращаемся в меню.')

).

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Меню программы

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

menu :-

writeln('Меню:'),

writeln('1. Ввести нового кандидата'),

writeln('2. Выполнить подбор кандидата'),

writeln('3. Выход'),

writeln('Выберите опцию (1, 2 или 3):'),

read(Choice),

( Choice = 1 ->

candidate\_menu

; Choice = 2 ->

run\_expert,

menu

; Choice = 3 ->

writeln('Выход из программы.'),

halt

; otherwise ->

writeln('Неверный выбор, попробуйте снова.'),

menu

).

candidate\_menu :-

add\_candidate,

writeln('Добавить еще кандидата? (1 - Да, 2 - Вернуться в главное меню)'),

read(Choice),

( Choice = 1 ->

candidate\_menu

; Choice = 2 ->

menu

; writeln('Неверный выбор, возвращаем в главное меню.'),

menu

).

% Определение предиката подбора кандидата

подбор(Кандидат) :-

кандидат(Кандидат).

:- initialization(menu).

Файл candidates.pl

:- dynamic кандидат/1.

% Пример кандидата ИВАН:

кандидат(иван) :-

язык\_программирования(python),

уровень\_опыта(middle),

опыт\_работы(3),

формат\_работы(удаленка),

зп(30000).

% Пример кандидата МАРИЯ (с двумя языками):

кандидат(мария) :-

язык\_программирования(java),

уровень\_опыта(senior),

опыт\_работы(7),

формат\_работы(очно),

зп(50000).

% Пример кандидата СЕМЕН (несколько языков):

кандидат(семен) :-

язык\_программирования(python),

уровень\_опыта(middle),

опыт\_работы(5),

формат\_работы(очно),

зп(40000).

% Пример кандидата САША:

кандидат(саша) :-

язык\_программирования(prolog),

уровень\_опыта(junior),

опыт\_работы(1),

формат\_работы(удаленка),

зп(30000).

Файл options\_db.pl  
:- dynamic option/2.

% Опции для языка программирования (оставляем без изменений)

option(язык\_программирования, python).

option(язык\_программирования, java).

% Опции для уровня опыта

option(уровень\_опыта, junior).

option(уровень\_опыта, middle).

option(уровень\_опыта, senior).

% Опции для опыта работы (числовые значения)

option(опыт\_работы, 1).

option(опыт\_работы, 3).

option(опыт\_работы, 5).

option(опыт\_работы, 7).

option(опыт\_работы, 9).

option(опыт\_работы, 10).

% Опции для формата работы

option(формат\_работы, удаленка).

option(формат\_работы, очно).

% Опции для зп (числовые значения)

option(зп, 30000).

option(зп, 40000).

option(зп, 50000).

option(зп, 60000).